Die Transportgleichung

Dieter Glaser 9630590
Peter Janach 9912935
Susanne Koch 9911414
Markus Sölkner 9973027

Februar 2004

1 Einleitung

Praktische Probleme in vielen Bereichen können auf die Lösung von partiellen Differentialgleichungen zurückgeführt werden. Die einfachsten Beispiele erscheinen in klassischen Bereichen der Physik, so zum Beispiel die Wärme-, die Wellen- oder die Transportgleichung.
In unserer Arbeit werden wir zunächst allgemein eine partielle Differentialgleichung definieren und dann eine typische Klassifikationsmöglichkeit für diese Gleichungen vorstellen. Im Anschluss daran beschäftigen wir uns im Speziellen mit der linearen Transportgleichung, wobei wir sowohl den homogenen als auch den inhomogenen Fall behandeln. Einen sehr wichtigen Teil unserer Arbeit nimmt die Numerische Lösung der Transportgleichung ein, wobei wir zuerst die Stabilität der Transportgleichung nachweisen und dann eine Fehlerabschätzung angeben werden. Abschließend werden wir die unterschiedlichen Ergebnisse verschiedener numerischer Algorithmen genauer untersuchen. In diesem Zusammenhang fügen wir unserer Arbeit auch den Source-Code eines von uns implementierten Programms bei. Im Speziellen behandeln wir zwei numerische Algorithmen, nämlich das Verfahren der Finiten Differenzen und das Upwind-Verfahren.

2 Definition

Eine partielle Differentialgleichung (PDE) ist eine Gleichung für eine Funktion von zwei oder mehreren Variablen und einige ihrer Ableitungen.

Der Ausdruck

$$F(D^k u, D^{k-1}u, ..., Du, u, x) = 0,\ x \in U \subset \mathbb{R}^n \tag{1}$$

ist eine PDE der Ordnung k.

$$F : (\mathbb{R}^n)^k \times (\mathbb{R}^n)^{k-1} ... \times (\mathbb{R}^n) \times \mathbb{R} \times U \to \mathbb{R} \tag{2}$$

Wir lösen (1), wenn wir ein u bestimmen, das (1) verifiziert und möglicherweise gegebene Bedingungen auf $\Gamma \subset \partial U$ erfüllt.
Dabei kann 'eine Lösung finden' folgendes bedeuten:

1. u in expliziter Form angeben
2. die Existenz von u zeigen

Auch die Eigenschaften der Lösungen sind für uns von großer Bedeutung.

3 Klassifizierung

Es gibt einige Möglichkeiten PDEs zu klassifizieren:

1. PDEs können an Hand ihrer Struktur klassifiziert werden: Sie sind
   (a) **linear**, wenn sie von der Form

   $$\sum_{|\alpha| \leq k} a_\alpha(x)(D^\alpha)u(x) = f(x)$$

   sind, wobei a_α, die Abbildung $f : U \to \mathbb{R}$ und $|\alpha| \leq k$ gegeben sind. Falls $f = 0$ ist, nennt man die PDE homogen.
   (b) **semilinear**, wenn sie von der Form

   $$\sum_{|\alpha|=k} a_\alpha(x) D^\alpha u(x) + G_0(D^{k-1}u(x), ...Du(x), u(x), x) = 0$$

   sind.

(c) **quasilinear**, wenn sie von der Form

$$\sum_{|\alpha|=k} a_\alpha(D^{k-1}u(x),...,Du(x),u(x),x)D^\alpha u(x)+G_0(D^{k-1}u(x),...,Du(x),u(x),x)=0$$

sind.

(d) **nicht linear**, wenn G_0 eine nichtlineare Funktion ist.

2. Die meisten partiellen Differentialgleichungen, mit denen wir uns beschäftigen sind PDEs der zweiten Ordnung:

$$\sum_{i,j=1}^{n} a_{ij}(x)u_{x_ix_j}(x)+\sum_{i=1}^{n} a_i(x)u_{x_i}(x)+a_0(x)u(x)=f(x),\quad x\in U$$

Da $u_{x_ix_j}=u_{x_jx_i}$ für alle $u\in C^2(u)$ gilt, können wir auch vermuten, dass $a_{ij}(x)=a_{ji}(x)$ für alle i,j = 1,. . . ,n gilt. Daher bilden die Koeffizienten $a_{ij}(x)$ eine symmetrischen Matrix $A(x)=(a_{ij})_{1\le i,j\le n}$.
Eine sehr brauchbare Klassifizierung kann durch die Eigenschaften der quadratischen Form $\xi\in\mathbb{R}^n\to(\xi^t)A(x)\xi=\sum_{i,j=1}^n a_{ij}(x)\xi_i\xi_j$ erreicht werden.

(a) Die lineare PDE zweiter Ordnung nennt man elliptisch, falls A(x) positiv oder negativ definit ist.

(b) Sie ist parabolisch, wenn A(x) singulär ist, und sie ist

(c) hyperbolisch, wenn ein Eigenwert von A(x) ein anderes Vorzeichen hat, als alle anderen Eigenwerte dieser Matrix.

Zu den elliptischen Gleichungen gehört die Laplace-Gleichung $u_{x_1x_1}=+\ldots+u_{x_nx_n}=0$, da $A(x)=I$.

Zu den parabolischen Gleichungen zählt die Wärmeleitgleichung $u_t-\Delta u=0$, da $A(t,x)=\begin{pmatrix}0&0\\0&-I\end{pmatrix}$.

Zu den hyperbolischen Gleichungen zählen die Wellengleichung $u_{tt}-\Lambda u=0$ und die Transportgleichung $u_t+bDu=0$.

Wir wollen uns in unserer Arbeit auf hyperbolische Gleichungen, im Speziellen auf das Lösen des Transportproblems konzentrieren.

4 Die lineare Transportgleichung

1. **Der homogene Fall**

Eine der einfachsten partiellen Differentialgleichungen ist die lineare Transportgleichung mit konstanten Koeffizienten

$$u_t(t,x) + \sum_{i=1}^{n} b_i u_{x_i}(t,x) = 0 \qquad \forall (t,x) \in (0,\infty) \times \mathbb{R}^n, \tag{3}$$

wobei $b = (b_1, \ldots, b_n)^T \in \mathbb{R}^n$ ein fixer Vektor ist. In kompakter Notation können wir das schreiben als:

$$u_t + b \cdot (Du) = 0 \qquad auf (0,\infty) \times \mathbb{R}^n, \tag{4}$$

wobei Du der von x abhängige Gradient ist, und $\cdot$ ein Euklidisches Skalarprodukt darstellt. Üblicherweise beschreibt t die Zeit und x einen Punkt im Raum.
In der Physik wird die Transportgleichung bei den verschiedensten Transporten von Stoffen verwendet, wie zum Beispiel Gase in einem Auspuff oder das Versickern von Wasser im Erdreich. Da Messungen in vielen Bereichen schwierig bis unmöglich sind, kann die Transportgleichung sehr hilfreich sein, um zum Beispiel die Qualität des Grundwassers abzuschätzen. Durch die Komplexität der meisten Transportprobleme muss man meistens auf ein numerisches Verfahren zurück greifen.
Um eine Formel zur Lösung dieser PDE abzuleiten, beobachten wir, dass $u_t + b \cdot (Du)$ die direkte Ableitung von u in Richtung $(1,b)^T$ ist.
Sei nun u irgendeine Lösung von (4) und sei $(t,x) \in (0,\infty) \times \mathbb{R}^n$ ein beliebiger Punkt. Dann definieren wir eine Funktion ν

$$\nu(s) := u(t+s, x+sb), s > -t.$$

Mit Hilfe der Kettenregel ergibt sich

$$\frac{d}{ds}\nu(s) = \frac{d}{ds}u(t+s,x+sb) = u_t(t+s,x+sb) + \sum_{i=1}^{n} u_{x_i}(t+s,x+sb)b_i = 0,$$

wobei wir (3) in der letzten Gleichung verwendet haben.

Daraus folgt:

$$\nu(s) = u(t+s,x+sb) = const \qquad \forall s > -t.$$

Das heißt: Der Wert u(t,x) wird entlang der Linie $(t+s, x+sb)$, $s > -t$ durch (t,x) transportiert.
Daher gilt, wenn ein $u \in C^1((0,\infty)\times\mathbb{R}^n) \cap C\overline{((0,\infty)\times\mathbb{R}^n)}$ zusätzlich zu (4) auch die Bedingung $u(0,x) = u_0(x) \quad \forall x \in \mathbb{R}^n$ mit den Daten $u_0 \in C^1(\mathbb{R}^n)$ erfüllt, dann erhalten wir:

$$u(t,x) = u(0, x-tb) = u_0(x-tb) \qquad \forall (t,x) \in (0,\infty)\times\mathbb{R}^n. \tag{5}$$

Dies führt uns zu folgendem Resultat:

Theorem 1: Wenn man die Gleichung $u_t + bDu = 0$ mit konstanten Koeffizienten $\underline{b} \in \mathbb{R}^n$ betrachtet, gelten folgende zwei Behauptungen:

(a) Ist u eine klassische Lösung von (4), dann erfüllt diese Lösung die Gleichung:

$$u(t+s, x+sb) \equiv const, s > -t \quad \forall (t,x) \in (0,\infty)\times\mathbb{R}^n. \tag{6}$$

(b) Seien $u_0 \in C^1(\mathbb{R}^n)$ gegebene Daten (Anfangswerte). Dann hat das Anfangswertproblem

$$\begin{cases} u_t + bDu &= 0 & \text{auf } (0,\infty)\times\mathbb{R}^n \\ u(0,x) &= u_0(x) & \forall x \in \mathbb{R}^n \end{cases} \tag{7}$$

eine eindeutige Lösung $u \in C^1((u,\infty)\times\mathbb{R}^n) \cap C\overline{((0,\infty)\times\mathbb{R}^n)}$, die durch

$$u(t,x) = u_0(x-tb), (t,x) \in (0,\infty)\times\mathbb{R}^n$$

gegeben ist.

Beweis:

(a) Wurde bereits gezeigt.

(b) Eindeutigkeit:
Wie wir schon gezeigt haben, erfüllt jede klassische Lösung von (7) die Gleichung $u(t,x) = u_t(x-tb)$ Für diese Gleichung ist u eindeutig.

Existenz:

Sei u definiert durch $u(t,x) = u_t(x - tb)$. Da $u_0 \in C^1(\mathbb{R}^n)$ ist, gilt: $u \in C^1((u,\infty) \times \mathbb{R}^n) \cap C(\overline{(0,\infty) \times \mathbb{R}^n})$. Weiters erfüllt u $u(0,x) = u_0(x)$ und wir haben:

$$u_t(t,x) = -b \cdot Du_0(x - tb), \qquad und \qquad Du(t,x) = Du_0(x - tb).$$

Dies zeigt, dass $u_t + b \cdot Du = 0$.

2. **Der inhomogene Fall**

Wir betrachten nun das allgemeine inhomogene Problem:

$$\begin{cases} u_t + b \cdot Du &= f & \text{auf } (0,\infty) \times \mathbb{R}^n \\ u(0,\cdot) &= u_0 & \text{auf } \mathbb{R}^n, \end{cases} \tag{8}$$

wobei $f \in C^1((0,\infty) \times \mathbb{R}^n)$.
Wie vorhin, ist die linke Seite in (8) die Ableitung in Richtung $(1,b)^T$. Daher erfüllt die Funktion $\nu(s) := u(t+s, x+sb), s > -t$ für jeden Punkt $(t,x) \in (0,\infty) \times \mathbb{R}^n$ die Gleichung

$$\frac{d}{ds}\nu(s) = u_t(t+s, x+sb) + \sum_{i=1}^{n} u_{x_i}(t+s, x+sb)b_i = f(t+s, x+sb). \tag{9}$$

Integration von (9) für $s \in [-t, 0]$ ergibt mit (8)

$$u(t,x) - u_0(x-tb) = \nu(0) - \nu(-t) = \int_{-t}^{0} f(t+s, x+sb)ds = \int_0^t f(s, x+(s-t)b)ds. \tag{10}$$

Daraus folgt das folgende Theorem:

Theorem 2:
Für alle $b \in \mathbb{R}^n$, $f \in C^1((0,\infty) \times \mathbb{R}^n)$ und $u_0 \in C^1(\mathbb{R}^n)$ hat das Problem

$$\begin{cases} u_t + b \cdot Du &= f & \text{auf } (0,\infty) \times \mathbb{R}^n \\ u(0,\cdot) &= u_0 & \text{auf } \mathbb{R}^n \end{cases}$$

eine klassische Lösung $u \in C^1((u,\infty)\times\mathbb{R}^n) \cap C(\overline{(0,\infty)\times\mathbb{R}^n})$, die gegeben ist durch:

$$u(t,x) = u_0(x-tb) + \int_0^1 f(s, x+(s-t)b)ds, \quad (t,x) \in (0,\infty)\times\mathbb{R}^n \quad (11)$$

Beweis:
Eindeutigkeit:
Wir haben gezeigt, dass jede klassische Lösung u von (8) die Gleichung (10), und somit auch (11) erfüllt. Daher bietet (11) die eindeutige Lösung von (8) und (10).

Existenz:
Es seien $f \in C^1((0,\infty)\times\mathbb{R}^n)$ und $u_0 \in C^1(\mathbb{R}^n)$. Das u in (11) ist offensichtlich $\in C^1((u,\infty)\times\mathbb{R}^n) \cap C(\overline{(0,\infty)\times\mathbb{R}^n})$ und erfüllt (8).
Weiters haben wir:

$$u_t = -b\cdot Du_0(x-tb) + f(t,x) - \int_0^t b\cdot D_x f(s, x+(s-t)b)ds$$

$$Du(t,x) = Du_0(x-tb) + \int_0^t D_x f(s, x+(s-t)b)ds$$

und daher gilt $u_t + b\cdot Du = f$, wie notwendig.

Wir haben somt gezeigt:

- Wir können (8) in ein System von gewöhnlichen Differentialgleichungen umwandeln. Diese Technik nennt man: 'Methode der Charakteristik'.
- Die Gleichung (11) macht auch Sinn für nicht-differenzierbare, ja sogar unstetige Daten. Dies führt zu einer sinnvollen Definition der sogenannten 'schwachen Lösungen' für (8), welche eine Ausbreitung der Konzepts der klassischen Lösungen hin zum Fall der weniger regulären Daten führt.

5 Numerische Lösung der Transportgleichung

Ein numerisches Verfahren zur Lösung der Transportgleichung ist unumgänglich, da man zwar in der Theorie exakte Lösungen angeben kann, jedoch in der Praxis Integrale auftreten können die nicht lösbar sind. Also möchten wir in diesem Abschnitt zunächst eine Lösung approximieren, und diese Approximation dann auch abschätzen, um eine bestimmte Genauigkeit der Lösung zu erreichen.

1. **Stabilität**
   Zuerst analysieren wir die Transportgleichung:

$$\begin{cases} \frac{\partial u}{\partial t} + b\frac{\partial u}{\partial x} &= f & \text{in } U_t \\ u &= 0 & \text{auf } \partial U \\ u &= g & \text{auf } U \times \{t = 0\} \end{cases} \tag{12}$$

   wobei $b > 0$ und $\partial U_- = \{x \in \partial U | b \cdot \nu < 0\}$, was man üblicherweise auch als 'boundary' bezeichnet. Im Gegensatz dazu bezeichnet ∂U_+ den 'outflow'.
   Man kann also durch die Randbedingung sagen, wieviel hineinströmt, aber nicht wieviel herausströmt.

   In einer Dimension gilt also: $U = (0, 1)$ bzw. $\partial U_- = \{0\}$ und somit $U(0, t) = 0$.
   Die Finite Differenzen Diskretisierung von (12) liefert:

$$\begin{cases} \frac{u_i^{m+1} - u_i^m}{\delta t} + b\frac{u_i^m - u_{i-1}^m}{h} &= f(x_i, t_m) \\ u_0^m &= 0 \\ u_i^0 &= g \end{cases} \tag{13}$$

   Somit haben wir:

$$u_i^{m+1} - u_i^m + (\frac{b\delta t}{h})(u_i^m - u_{i-1}^m) = \delta t f(x_i, t_m)$$

   Sei nun $\mu = \frac{b\delta t}{h}$ und nehmen wir an, dass $0 \leq \mu \leq 1$ ist, dann bezeichnet μ die COURANT-Number (garantiert die Stabilität für dieses Schema)

   Also gilt:
   $u_i^{m+1} = u_i^m - \mu u_i^m + \mu u_{i-1}^m + \delta t f(x_i, t_m)$

Daraus folgt:

$$\begin{aligned} |u_i^{m+1}| &\leq (1-\mu)|u_i^m| + \mu|u_{i-1}^m| + \delta t max_i|f_i^m| \\ &\leq (1-\mu)\ max_i|u_i^m| + \mu\ max_i|u_{i-1}^m| + \delta t\ max_i|f_i^m| \\ &\leq max_i|u_i^m| + \delta t\ max_i|f_i^m| \end{aligned}$$

$$\Longrightarrow max_i|u_i^{m+1}| \leq max_i|u_i^m| + \delta t\ \max_i|f_i^m| \tag{14}$$

Definieren wir nun: $\|u_i^m\|_\infty = max_{0\leq i\leq N}|u_i^m|$, dann gilt:
$\|u^{m+1}\|_\infty \leq \|u^m\|_\infty + \delta t\|f(\cdot, t_m)\|_\infty$

Mit Rekursion: $\|u^{m+1}\|_\infty \leq \|u^0\|_\infty + \sum_{k=0}^{M-1} \delta t\|f(\cdot, t_m)\|_\infty$
also: $max_{1\leq m\leq M}\|u^m\|_\infty \leq \|u^0\|_\infty + \sum_{k=0}^{M-1} \delta t\|f(\cdot, t_m)\|_\infty$

Somit haben wir die Stabilität für $\delta t \leq \frac{h}{b}$ gezeigt.

2. **Fehlerabschätzung**
Für die Fehlerabschätzung für

$$e_i^m = u(x_i, t_m) - u_i^m$$

haben wir

$$\frac{e_i^{m+1} - e_i^m}{\delta t} + bD_x^- e_i^m = \varphi_i^m \tag{15}$$

Wobei aus der Taylor Entwicklung folgt:

$\varphi_i^m = \frac{1}{2}\delta t\frac{\partial^2 u}{\partial t^2}(x_i, \tau_m) + \frac{1}{2}bh\frac{\partial^2 u}{\partial x^2}(\xi_i, t_m)$,
mit $\tau_m \in (t_m, t_{m+1})$ und $\xi_i \in (x_{i-1}, x_i)$

Dann gilt:
$|\varphi_i^m| \leq \frac{1}{2}\delta t M_t + \frac{b}{2}hM_x$
wobei $M_t = max_{x_i}|\frac{\partial^2 u}{\partial t^2}|$ und $M_x = max_{x_i}|\frac{\partial^2 u}{\partial x^2}|$,

also $\|\varphi^m\|_\infty \leq \frac{1}{2}M(\delta t + bh)$
Wir definieren nun noch: $M = max(M_t, M_x)$.

Jetzt wenden wir (14) auf (15) an und nehmen weiters an, dass
$\|e^0\|_\infty = max_i|g(x_i) - u_0| = 0$.

Dann gilt: $max_{1\leq m\leq M}\|e^m\|_\infty \leq \frac{MT}{2}(\delta t + bh)$ also:

$$max_{1\leq m\leq M}\|u(\cdot, t_m) - u^m\|_\infty \leq \frac{MT}{2}(\delta t + bh) \tag{16}$$

Somit ist der Fehler: 0 $(\delta t + bh)$

3. **System von Transportgleichungen**

Sei $A \in \mathbb{R}^{k\times k}$ eine Matrix mit k reellen Eigenwerten $\{\lambda_i\}_{i=1}^k$ und k linear unabhängigen Eigenvektoren. Das System

$$\frac{\partial u}{\partial t} + A\frac{\partial u}{\partial x} = 0 \quad \underline{u} = (u_1, ..., u_k) \tag{17}$$

ist hyperbolisch.
Unter diesen Voraussetzungen existiert eine nicht singuläre Matrix P, sodass $PAP^{-1} = D$, wobei D eine Diagonalmatrix ist: $D = diag(\lambda_1 ... \lambda_k)$.

$$\begin{aligned} \tfrac{\delta}{\delta t}(P\underline{u}) + \underbrace{PAP^{-1}}P\tfrac{\delta u}{\delta x} &= 0 \\ \tfrac{\delta}{\delta t}(P\underline{u}) + D(\tfrac{\delta}{\delta x})(P\underline{u}) &= 0 \\ \underline{v} &= P\underline{u} \end{aligned}$$

4.
$$\begin{aligned} \tfrac{\delta v}{\delta t} + D\tfrac{\delta \underline{v}}{\delta x} &= 0 \\ \tfrac{\delta v_i}{\delta t}v + \lambda_i\tfrac{\delta v_i}{\delta x} &= 0 \quad i = 1, ..., k \end{aligned}$$

Dies entspricht $u_t + bu_x = 0$; das heißt man hat hier mehrere Transportgleichungen.

$$\begin{cases} \tfrac{\delta u}{\delta t} + b\tfrac{\delta u}{\delta x} = f & in\ U_t \\ u = 0 & \text{auf } \delta U \\ u = g & \text{auf } U \times \{t = 0\} \end{cases}$$

$$\begin{aligned} u_t + bDu &= f \\ u &= 0 \end{aligned}$$

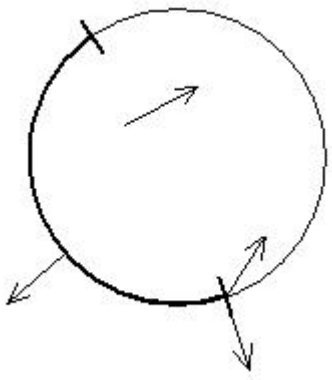

Abbildung 1: Kreis

$\delta u_- = \{x \in \delta u | bv < 0\}$
δu_t.....outflow

Man kann also sagen wieviel hineinströmt, aber nicht weiviel herausströmt.
$Dx^- : \frac{u_i^{m+1}-u_i^m}{\delta t} + b\frac{u_i^m - u_{i-1}^m}{h} = f(x_i, t_m)$

$u_i^{m+1} - u_i^m + (\frac{b\delta t}{h})(u_i^m - u_{i-1}^m) = \delta t f(x_i, t_m)$

$u = \frac{b\delta t}{h}$...COURANT Number (garantiert die Stabilität für dieses Schema)

$u_i^{m+1} = u_i^m - \mu u_i^m + \mu u_{i-1}^m + \delta t f(x_i, t_m) \qquad 0 < \mu \leq 1$

$$\begin{aligned} |u_i^{m+1}| &\leq (1-\mu)|u_i^m| + \mu|u_{i-1}^m| + \delta t|f_i^m| \\ &\leq (1-\mu)\ max_i|u_i^m| + \mu\ max_i|u_{i-1}^m| + \delta t\ max_i|f_i^m| \\ &\leq max_i|u_i^m| + \delta t\ max_i|f_i^m| \\ \\ &\Longrightarrow \quad max_i|u_i^{m+1}| \leq max_i|u_i^m| + \delta t\ \max_i|f_i^m| \end{aligned}$$

$\|u_i^m\|_\infty = max|u_i^m|$
$\|u^{m+1}\|_\infty \leq \|u^m\|_\infty + \delta t\|f(\cdot, t_m)\|_\infty$

Rekursiv: $\|u^{m+1}\|_\infty \leq \|u^0\|_\infty + \sum_{k=0}^m \delta t\|f(\cdot, t_m)\|_\infty$
$max_{1\leq m\leq M}\|u^m\|_\infty \leq \|u^0\|_\infty + \sum_{k=0}^{M-1} \delta t\|f(\cdot, t_m)\|_\infty$

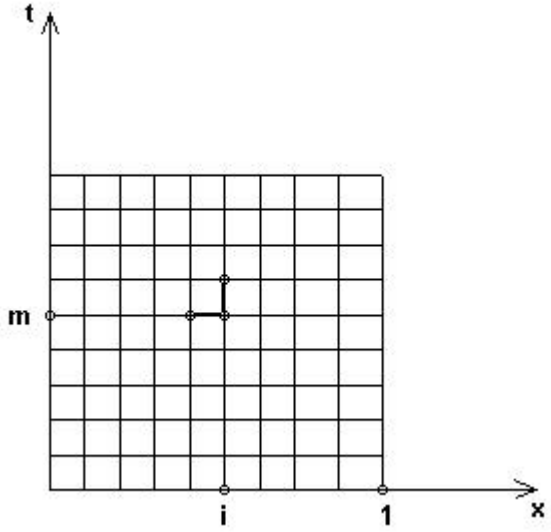

Abbildung 2: Raster

1. System von Transportgleichungen

$$\frac{\partial u}{\partial t} + A\frac{\partial u}{\partial x} = 0 \qquad \underline{u} = (u_1, ..., u_k)$$

$A \in \mathbb{R}^{k\times k}$, ß; $\{\lambda_i\}_{i=1}^k$, $\lambda_i \in \mathbb{R}, i = 1, ..., k$ Eigenwerte. Die k Eigenwerte sind linear unabhängig und es existiert eine Matrix P, die nicht singulär ist, mit folgender Eigenschaft:
$\exists P, P^{-1}$, sodass $PAP^{-1} = D$, mit $D = diag(\lambda_1, ...\lambda_k)$

$$\begin{aligned} \tfrac{\delta}{\delta t}(P\underline{u}) + \underbrace{PAP^{-1}}P\tfrac{\delta u}{\delta x} &= 0 \\ \tfrac{\delta}{\delta t}(P\underline{u}) + D(\tfrac{\delta}{\delta x})(P\underline{u}) &= 0 \\ \underline{v} &= P\underline{u} \end{aligned}$$

2.

$$\begin{aligned} \tfrac{\delta v}{\delta t} + D\tfrac{\delta \underline{v}}{\delta x} &= 0 \\ \tfrac{\delta v_i}{\delta t}v + \lambda_i\tfrac{\delta v_i}{\delta x} &= 0 \quad i = 1, ..., k \end{aligned}$$

Dies entspricht $u_t + bu_x = 0$; das heißt man hat hier mehrere Transportgleichungen.

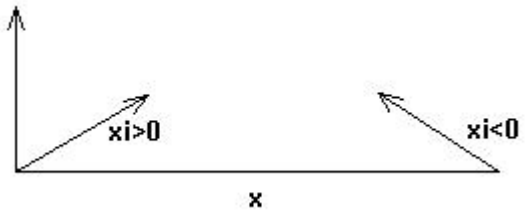

Abbildung 3: x_i

6 Unterschiedliche Ergebnisse numerischer Algorithmen

6.1 Das Verfahren der Finiten Differenzen

Zur numerischen Ermittlung der Lösung einer partiellen Differentialgleichung verwendetn wir das Verfahren der Finiten Differenzen: Dabei wird ein Gitter über den zu behandelnden Raum $U \subseteq \mathbb{R}$ gelegt. Dadurch wird erreicht, dass nur mehr mit endlich vielen Werten gerechnet werden muss. Je feiner man das Netz wählt, das heißt je kleiner die räumliche Schrittweite $\triangle x_i$ gewählt wird, desto näher wird die numerische an der exakten Lösung liegen.
Mit Hilfe der Taylor-Reihen-Entwicklung erhält man eine Approximation der Ableitungen:

$$\begin{aligned} u(x_{i+1}) &= u(x_i) + u'(x_i) \cdot h + \frac{1}{2}u''(x_i) \cdot h^2 + \frac{1}{6}u'''(x_i) \cdot h^3 + \dots \\ \frac{u(x_{i+1}) - u(x_i)}{h} &= u'(x_i) + \frac{1}{2}u''(x_i) \cdot h + \frac{1}{6}u'''(x_i) \cdot h^2 + \dots \\ D_x^+ &= \frac{u(x_{i+1}) - u(x_i)}{h} \end{aligned}$$

und in weiterer Folge kann man etwa $\frac{\partial}{\partial x_i}$ bzw. $\frac{\partial}{\partial t}$ durch die Funktionen $D_x^{\pm}$ bzw. $d_t^{\pm}$ ersetzen.

6.2 Lösung mit Hilfe des Upwind-Verfahrens

Wir werden in der Folge das zweiparametrige Transportproblem

$$u(x,y)_t + \lambda \cdot u(x,y)_x = 0 \tag{18}$$

des dreidimensionalen Raums mit Hilfe der im vorigen Kapitel vorgestellten Methode der Finiten Differenzen lösen.

Zunächst wählen wir einen geeigneten Unterraum U_{xy} der xy-Ebene des kartesischen Normalkoordinatensystems. Für jeden Punkt dieses Raums U_{xy} ist die z-Koordinate einer Ausgangsfläche $\Phi \mathrel{\hat{=}} f(x,y)$ definiert. In weiterer Folge wird ein Gitter über U_{xy} gelegt, die Gitterpunkte (x,y) definieren eine endliche Anzahl von Punkten $(x,y,f(x,y))$ der Ausgangsfläche Φ mit deren Hilfe die Lösung der partiellen Differentialgleichung approximiert wird.

Ersetzt man, wie im vorigen Kapitel vorgestellt, die partiellen Ableitungen in (18) durch die Approximationen d_t^+ bzw. D_x^+, so erhält man:

$$d_t^+ u_i + \lambda_i D_x^+ u_i = 0 \qquad i = 1,2 \tag{19}$$

bzw. für $t \in [0,T]$

$$\frac{u_i^{t+\delta t} - u_i^t}{\delta t} + \lambda_i \frac{u_{i+1}^t - u_i^t}{\delta x_i} = 0 \qquad i = 1,2 \tag{20}$$

Definiert man das für die Stabilität des Algorithmus wesentliche Verhältnis $\mu := \frac{\delta t}{\delta x_i}$ erhält man schließlich nach kurzen Umformungen

$$u_i^{t+\delta t} = u_i^t - p\lambda_i (u_{i+1}^t - u_i^t) \tag{21}$$

Es ist also möglich, mit Hilfe der für jede Koordinate entsprechenden 'rechten' Partnerpunkte ($\rightarrow$ Upwindverfahren) die neuen Koordinaten eines Punktes nach einem vorgegebenen Zeitintervall zu bestimmen.
Abbildung 4 zeigt die Lage bzw. Gestalt eines zu bewegenden Quaders zu angegebenen Zeitpunkten. Dabei ist zu beachten, dass wir den Algorithmus so implementiert haben, dass der bewegte Körper periodisch wiederkehrt und nicht verschwindet. Der folgende Text zeigt den Source-Code des von uns implementierten Programms:

```
%function [time]=trans_equ(u0,l,r,delta_x,delta_t,tmax,vers)

% Gruppe Transportgleichung, PDE, Februar 2004
%
% Eingabeparameter:
```

```
% u0.........Anfangsbedingung (Ausgangsfunktion)
% l..........linke Intervallgrenze
% r..........rechte Intervallgrenze
% delta_x....raeuumliche Schrittweite
% delta_t....zeitliche Schrittweite
% tmax.......Zeit, bis zu der iteriert wird
% vers.......gibt an, welche Diskretisierung verwendet werden soll
%
% Ausgabeparamter:
% time.......benoetigte CPU-time

u0 = 'quader'; l = 0.1; r = 10; delta_x = 0.1; delta_t = 0.05;
tmax = 10; vers = 'periodisch'; paus=0;

figure(1) time=cputime; t=0; p=delta_t / delta_x; weite=r-l;

gitter_x=l:delta_x:r;                       % Gitterpunktematrix
gitter_y=l:delta_x:r;

for i=1:length(gitter_x)                    % Startmatrix wird erstellt
   for j=1:length(gitter_y)
      valt(i,j)=feval(u0,gitter_x(i),gitter_y(j),weite);
   end
end

vneu=valt; colormap('winter'); surf(vneu); title(t);
axis([0,100,0,100,0,30]); pause(2); valt = valt;

%-------------------------------------------------------------------------
% forward forward periodisch Diskretisierung
if vers=='periodisch'
   while t<tmax
      t = t + delta_t;

      for i=1:length(gitter_x)
          for j=1:length(gitter_y)
             if i==length(gitter_x)
                 vneu(i,j)=valt(i,j)+p*(valt(1,j)-valt(i,j));
             else
```

```
                vneu(i,j) = valt(i,j) + p*(valt(i+1,j) - valt(i,j));
            end
        end
    end

    valt=vneu;

    for i=1:length(gitter_x)
        for j=1:length(gitter_y)
            if j==length(gitter_y)
                vneu(i,j)=valt(i,j)+p*(valt(i,1)-valt(i,j));

            else
                vneu(i,j) = valt(i,j) + p*(valt(i,j+1) - valt(i,j));
            end
        end
    end

    valt=vneu;
    surf(valt);
    title(t);
    axis([0,100,0,100,0,30]) ;

    pause(paus);
  end

end
%-------------------------------------------------------------------------

time=cputime-time;
```

Wie man nur unschwer erkennen kann, ist dieser Algorithmus nur eingeschränkt verwendbar, da sich der zu bewegende Körper bereits nach kurzer Zeit sehr stark verformt, was nicht unbedingt dem erwarteten Ergebnis eines Transports entspricht. Außerdem ist der Algortithmus nicht für beliebige Werte stabil und in Abbildung ist ein solcher Fall dargestellt.

7 TVD (Total variation diminishing) Verfahren

Wie man in unserer Graphik sehen kann, liefert das Upwind Verfahren schon nach kurzer Zeit einen großen Fehler. Um den geringer zu halten kann man das TVD Verfahren verwenden. Das TVD Verfahren braucht dafür um einiges mehr an Rechnerleistung.

$$\frac{u_i^{m+1} - u_i^m}{\delta t} = -\frac{b}{\delta x}(1 + \frac{1}{2}\chi(r^+_{i-\frac{1}{2}}) - \frac{1}{2}\frac{\chi(r^+_{i-\frac{3}{2}})}{r^+_{i-\frac{3}{2}}})(u_i - u_{i-1})$$

$$mit \qquad r^+_{i-\frac{1}{2}} = \frac{(u_{i+1} - u_i)}{u_i - u_{i-1})}, \quad r^+_{i-\frac{3}{2}} = \frac{(u_i - u_{i-1})}{(u_{i-1} - u_{i-2})}$$

$$und \qquad \chi(r) = max[0, min(2r, 1), min(r, 2)]$$

Der folgende Text zeigt den Source-Code des von uns implementierten Programms:

```
%  TVD - Verfahren   periodisch

% Gruppe Transportgleichung, PDE, Februar 2004
%
% Eingabeparameter:
% u0.........Anfangsbedingung (Ausgangsfunktion)
% l..........linke Intervallgrenze
% r..........rechte Intervallgrenze
% delta_x....raeuumliche Schrittweite
% delta_t....zeitliche Schrittweite
% tmax.......Zeit, bis zu der iteriert wird
% vers.......gibt an, welche Diskretisierung verwendet werden soll
%
% Ausgabeparamter:
% time.......benoetigte CPU-time

u0 = 'quader'; l = 0.1; r = 10; delta_x = 0.1; delta_t = 0.05;
tmax = 0.4;

figure(1)
bx=1;
by=1;
```

```
t=0;
p=delta_t / delta_x;
weite=r-l;

gitter_x=l:delta_x:r;                          % Gitterpunktematrix
gitter_y=l:delta_x:r;

for i=1:length(gitter_x)                       % Startmatrix wird erstellt
   for j=1:length(gitter_y)
      valt(i,j)=feval(u0,gitter_x(i),gitter_y(j),weite);
   end
end

vneu=valt; colormap('winter'); surf(vneu); title(t);
axis([0,100,0,100,0,30]); pause(0); valt = valt; dieter=0;
%-------------------------------------------------------------------------
% TVD - Scheme

while t<tmax

t = t + delta_t;

for i=1:(length(gitter_x))
  for j=1:(length(gitter_y))

  if i==1
   rpim12=(valt(i+1,j)-valt(i,j)) / (0.00001 + valt(i,j)-
    valt(length(gitter_x),j));
     rpim32=(valt(i,j)-valt(length(gitter_x),j)) / (0.00001 +
      valt(length(gitter_x),j)-valt((length(gitter_x)-1),j));

       vneu(i,j)=valt(i,j)-delta_t/delta_x * bx*(1+1/2*chi(rpim12)-
        1/2*chi(rpim32)/(rpim32 + 0.00001) ) * (valt(i,j)-valt(length(gitter_x),j));

         else
          if i==2
            rpim12=(valt(i+1,j)-valt(i,j)) / (0.00001 + valt(i,j)-valt(i-1,j));
              rpim32=(valt(i,j)-valt(i-1,j)) / (0.00001 + valt(i-1,j)
                -valt(length(gitter_x),j));
```

```
            vneu(i,j)=valt(i,j)-delta_t/delta_x * bx*(1+1/2*chi(rpim12)
            -1/2*chi(rpim32)/(rpim32 + 0.00001) ) * (valt(i,j)-valt(i-1,j));

            else
            if i==(length(gitter_x))
            rpim12=(valt(1,j)-valt(i,j)) / (0.00001 + valt(i,j)-valt(i-1,j));
            rpim32=(valt(i,j)-valt(i-1,j)) / (0.00001 + valt(i-1,j)-valt(i-2,j));

            vneu(i,j)=valt(i,j)-delta_t/delta_x * bx*(1+1/2*chi(rpim12)
            -1/2*chi(rpim32)/(rpim32 + 0.00001) ) * (valt(i,j)-valt(i-1,j));

            else
            rpim12=(valt(i+1,j)-valt(i,j)) / (0.00001 + valt(i,j)-valt(i-1,j));
            rpim32=(valt(i,j)-valt(i-1,j)) / (0.00001 + valt(i-1,j)-valt(i-2,j));

            vneu(i,j)=valt(i,j)-delta_t/delta_x * bx*(1+1/2*chi(rpim12)
            -1/2*chi(rpim32)/(rpim32 + 0.00001) ) * (valt(i,j)-valt(i-1,j));

            end
          end
        end



   end
end

    valt=vneu;

 for i=1:(length(gitter_x))
 for j=1:(length(gitter_y))
 if j==1
  rpim12=(valt(i,j+1)-valt(i,j)) / (0.00001 + valt(i,j)
   -valt(i,length(gitter_y)));
    rpim32=(valt(i,j)-valt(i,length(gitter_y))) / (0.00001 + valt(i,length(gitter_y))
     -valt(i,(length(gitter_y)-1)));
```

```
 vneu(i,j)=valt(i,j)-delta_t/delta_x * by*(1+1/2*chi(rpim12)
 -1/2*chi(rpim32)/(rpim32 + 0.00001) ) * (valt(i,j)-valt(i,length(gitter_y)));

else
if j==2
rpim12=(valt(i,j+1)-valt(i,j)) / (0.00001 + valt(i,j)-valt(i,j-1));
rpim32=(valt(i,j)-valt(i,j-1)) / (0.00001 + valt(i,j-1)-valt(i,length(gitter_y)));


vneu(i,j)=valt(i,j)-delta_t/delta_x * by*(1+1/2*chi(rpim12)
-1/2*chi(rpim32)/(rpim32 + 0.00001) ) * (valt(i,j)-valt(i,j-1));

else
if j==length(gitter_y)
rpim12=(valt(i,1)-valt(i,j)) / (0.00001 + valt(i,j)-valt(i,j-1));
rpim32=(valt(i,j)-valt(i,j-1)) / (0.00001 + valt(i,j-1)-valt(i,j-2));


vneu(i,j)=valt(i,j)-delta_t/delta_x * by*(1+1/2*chi(rpim12)
-1/2*chi(rpim32)/(rpim32 + 0.00001) ) * (valt(i,j)-valt(i,j-1));

else
rpim12=(valt(i,j+1)-valt(i,j)) / (0.00001 + valt(i,j)-valt(i,j-1));
rpim32=(valt(i,j)-valt(i,j-1)) / (0.00001 + valt(i,j-1)-valt(i,j-2));


vneu(i,j)=valt(i,j)-delta_t/delta_x * by*(1+1/2*chi(rpim12)
-1/2*chi(rpim32)/(rpim32 + 0.00001) ) * (valt(i,j)-valt(i,j-1));

end
end



 end
 end
end
```

```
      valt=vneu;
      colormap=('winter');
      surf(valt);
      title(t);
      axis([0,100,0,100,0,30]) ;


   end
```

Die Funtion chi:

```
function [chi]=chivon(r)

a=[2*r,1]; b=[r,2];

m1=min(a); m2=min(b);

c=[0,m1,m2];

chi=max(c);
```

Die Anfangsbedingungen werden werden in diesem Fall über die Funktion 'quader' geladen:

```
function [fij]=quader(i,j,l)

fij = 0;

if i > l * 0.3
   if i < l * 0.6
      if j > l * 0.4
         if j < l * 0.7
            fij = 30; % ändern, verallgemeinern
         end
      end
   end
end
```

Wie man in den folgenden Graphiken sehen kann, ist die Glättung bzw. der numerische Fehler wesentlich kleiner.

8 Berechnung des Fehlers

Um das Verhalten des Fehlers bezüglich der verschiedenen Verfahren, Zeitschritten und Gitterschritten zu beobachten, haben wir ein kleines Programm geschrieben, das die Werte der Transportgleichung berechnet und haben sie mit den numerisch berechneten Werten verglichen.

Verfahren	δ x	δt	Zeitpunkt	Fehler (Maximumsnorm)
upwind	0,1	0,05	0,1	0,0070
upwind	0,05	0,025	0,1	0,0051
upwind	0,025	0,0125	0,1	0,0037
upwind	0,1	0,05	0,3	0,0013
upwind	0,05	0,025	0,3	0,0009
upwind	0,025	0,0125	0,3	0,0006
TVD	0,1	0,05	0,1	0,0012
TVD	0,05	0,025	0,1	0,0006
TVD	0,025	0,0125	0,1	0,0004
TVD	0,1	0,05	0,3	0,0032
TVD	0,05	0,025	0,3	0,0028
TVD	0,025	0,0125	0,3	0,0018

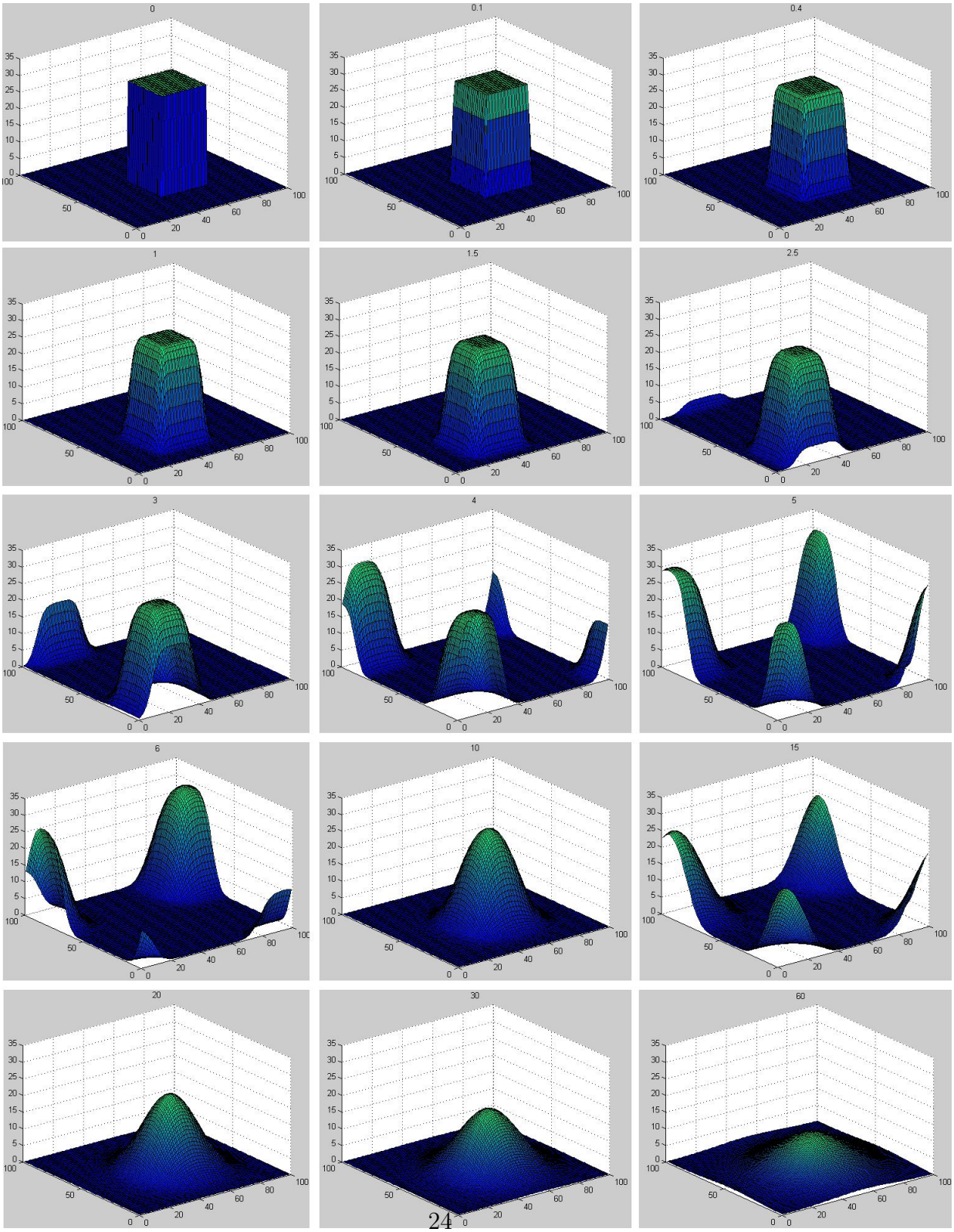

Abbildung 4: Quader wird transportiert

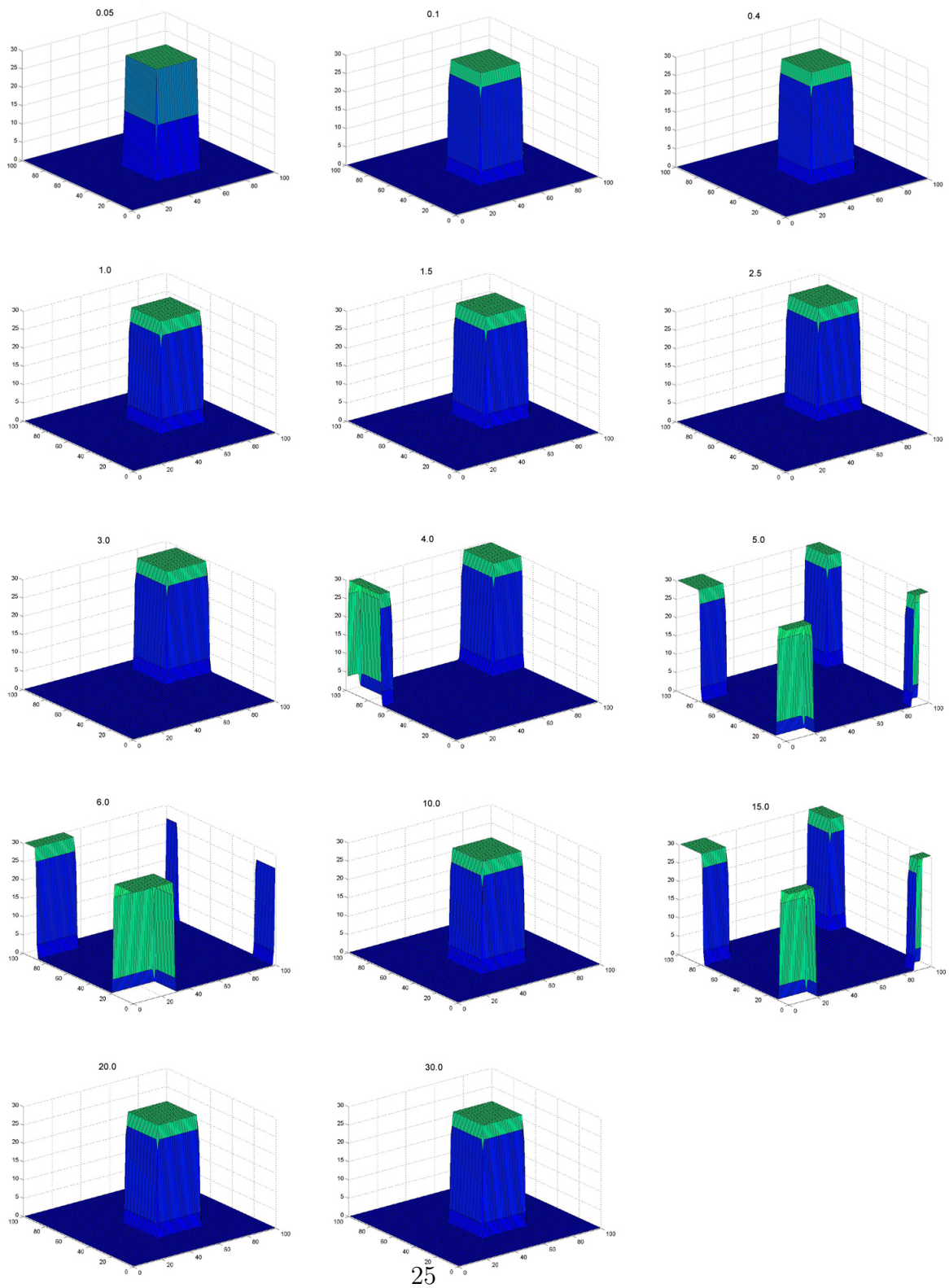

Abbildung 5: TVD Quader wird transportiert