noch in Bearbeitung

Wissenschaftliches Rechnen

Vorlesungsskript WS 2015/16

Prof. Dr. Christian Engwer

15. Januar 2018

Vielen Dank an Sophie Schrader für die Mühe meinen handschriftlichen Notizen in LaTeXzu gießen.
Dank gilt darüber hinaus Marcel Koch für seine Unterstützung bei der Vorlesungsdurchführung.
Für die Unterstützung bei den vorangegangenen Vorlesungen bedanke ich mich bei Liesel Schumacher, Markus Knappitsch und Sebastian Westerheide.

Inhaltsverzeichnis

Todo list

1. Einführung

Wir möchten in diesem ersten Kapitel uns anhand eines einfachen Experiments den Prozess der Modellierung, numerischen Diskretisierung und Implementierung in einem Computerprogramm vor Augen führen.

1.1. Das mathematische Pendel

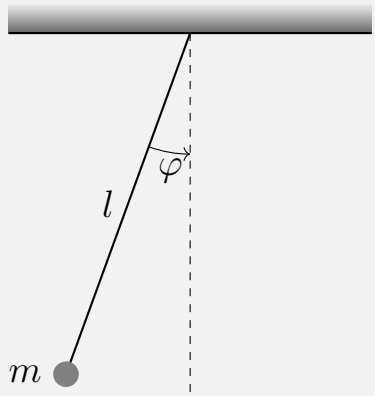
Wir betrachten ein ideales Pendel. In der Realität werden einige weitere Effekte zu beobachten sein, die wir hier zunächst ignorieren wollen.

1.1.1. Experiment

1.1.2. Beobachtung

- Gewicht pendelt um Ruheposition
- Periode unabhängig von Gewicht, Auslenkung (annähernd)
- Periodendauer abhängig von Fadenlänge l ($T \sim \sqrt{l}$)

1.1.3. Modell



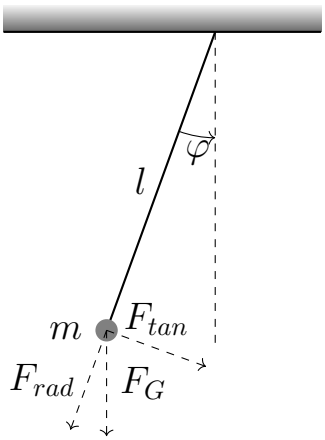
Systemeigenschaften

- Gewicht m
- Länge l
- Auslenkung φ

Mögliche Untersuchungen

- Schwingungsdauer einer Periode
- Abhängigkeit von
  - Auslenkung
  - Gewicht
  - Länge

Abbildung 1.1.: Aufbau des Pendel Experiments



Modellannahmen

- Reibungsfreiheit
- Punktmasse
- Faden massefrei, nicht elastisch

Newtonsche Mechanik

- Gewichtskraft: $F_G = m \cdot g$,
  Erdbeschleunigung g (vektorwertig)
- Beschleunigung (2. Newtonsches Gesetz): $F_a(t) = m \cdot a(t)$

$\rightarrow$ Rückstellkraft: $F_R(t) = -|F_{tan}(t)| = -m \cdot |g| \cdot sin(\varphi(t))$

$\rightarrow$ Beschleunigung in Polarkoordinaten $a(t) = l \cdot \ddot{\varphi}(t)$

→ Kräftegleichgewicht:

$$\begin{aligned} & F_R(t) = F_a(t) \\ \Leftrightarrow \quad & -m \cdot |g| \cdot sin(\varphi(t)) = m \cdot l \cdot \ddot{\varphi}(t) \\ \Leftrightarrow \quad & 0 = \frac{|g|}{l} sin(\varphi(t)) + \ddot{\varphi}(t) \end{aligned}$$

1.2. Grundlegende Begriffe

Wir betrachten

- Modelle zur Beschreibung von (realen) Systemen, sowie die
- Simulation dieser Modelle auf einem Computer

Definition 1.1 System: Unter einem *System* versteht man allgemein Komponenten, die über Signale interagieren. Dabei unterscheiden wir offene und geschlossene Systeme. Zustandsgrößen unterscheiden eindeutig den Zustand der Komponenten.

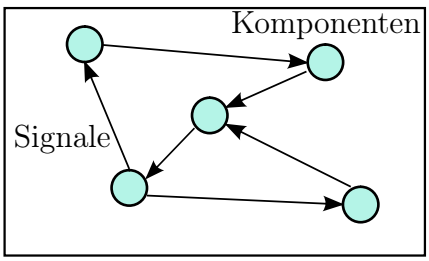


Abbildung 1.2.: geschlossenes System

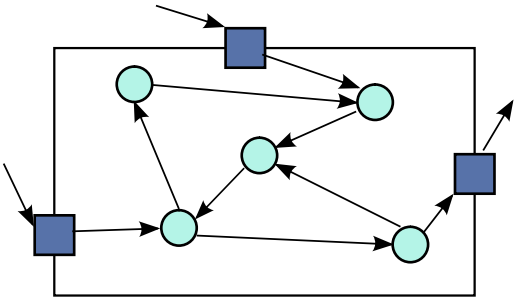


Abbildung 1.3.: offenes System

In einem geschlossenen System ist das Verhalten komplett durch seine Komponenten bestimmt. Ein offenes System interagiert mit der Umwelt, es erfordert zur Beschreibung zusätzliche Informationen an den Systemgrenzen.

Definition 1.2 Modell: Ein *Modell* ist eine Repräsentation eines Systems, mit dem Ziel dieses zu erfassen. Das Modell ist eine Vereinfachung des Systems, welches aber zur Erfüllung des Zweckes ausreicht. Modelle können hierarchisch sein, da auch Modelle ein System darstellen.

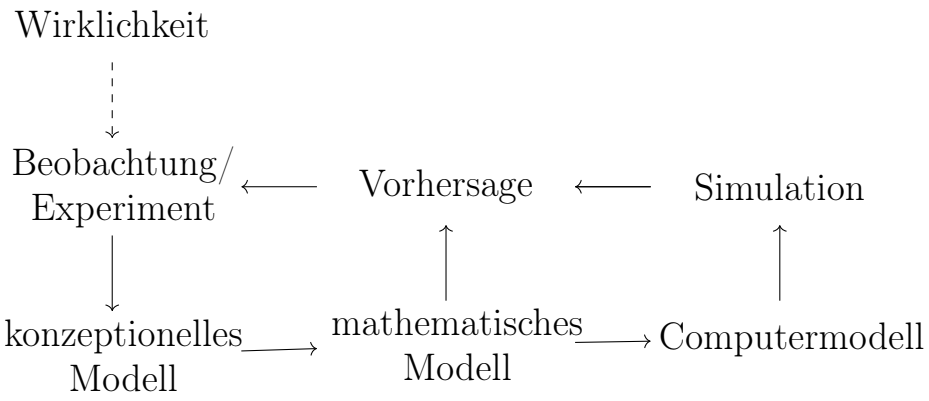
Ergänzung: **Mathematisches Modell**

Das Modell wird formalisiert mit den Methoden der Mathematik, z. B.

- Prädikatenlogik
- Regelsysteme (Automaten, formale Spachen,...)
- Gleichungen (algebraische Gleichungen, Differentialgleichungen)

In den Natur- und Ingenieurwissenschaften ist dieses Vorgehen natürlich, da Sachverhalte bereits mathematisch formuliert werden. In anderen Bereichen, wie z. B. der Medizin, ist dies schwieriger.

1.2.1. Allgemeines Vorgehen



Nach Modellbildung und Simulation wird die Vorhersage mit der Wirklichkeit verglichen. Abweichungen, die über dem erwarteten Maß liegen, sind den einzelnen Teilschritten zuzuordnen, insbesondere unterscheidet man:

- Modellfehler: nicht erfasste (physikalische) Prozesse, Annahmen über Homogenität, Form von Abhängigkeiten, etc.
- Datenfehler: ungenügend genau erfasste Daten an den Systemgrenzen (Rand- und Anfangsbedingungen) oder Parameter
- Simulationsfehler: (Approximationsfehler): Fehler, die durch die Approximation des Modells zum Zwecke der numerischen Lösung entstehen (Diskretisierungsfehler, Rundungsfehler, etc.)

1.2.2. Forderungen an ein mathematisches Modell

1. Hypothesen sollten eindeutig und überprüfbar sein
2. Übersetzung in die Sprache der Mathematik sollte überzeugend sein

   Beispiel: Bevölkerungswachstum $= c \cdot$ Bevölkerung (wirklich linear?)

3. Lösungsmethoden sollten sich eng an das mathematische Modell anpassen

1.2.3. Eigenschaften eines mathematischen Modells

Mathematische Modelle können wir in Klassen unterteilen:

a) dynamische Modelle: Zustandsgrößen und Signale hängen von der Zeit ab. Gegenteil: statische Modelle

b) deterministische Modelle: Zu jedem Zeitpunkt sind Zustände und Signale eindeutig bestimmt. Gegenteil: stochastische Modelle; Größen können zufällig gewählt werden

c) diskrete Modelle: Werte der Zustandsgrößen nur diskret / nur zu einem diskreten Zeitpunkt gegeben. Gegenteil: kontinuierliche Modelle

d) hierarchische Modelle: Viele Modelle können auf unterschiedlichen Skalen betrachtet werden.

Beispiel: Atommodelle

- Demokrit (400 v. Chr.)
- Bohr: diskrete Bahnen von Elektronen
- Orbitalmodell: Aufenthaltswahrscheinlichkeit
- Quarkmodell: Protonen/Neutronen bestehen aus Quarks, starke Wechselwirkung

Welche Skala sich am besten eignet, hängt vom Problem ab. Makroskopische Modelle können bspw. nur mit sehr hohem Aufwand auf der Mikroskala simuliert werden. In *Mehrskalensimulationen* werden Modelle auf verschiedenen Skalen gekoppelt. Die Trennung in verschiedene Skalen ist allerdings nicht immer möglich (kontinuierliche Skalen, z. B. turbulente Strömungen).

1.3. Numerisches Modell

2. Newtonsche Mechanik

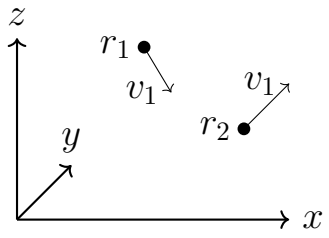
$$\text{Beobachtung} \longrightarrow \begin{matrix}\text{konzept.}\\ \text{Modell}\end{matrix} \longrightarrow \begin{matrix}\text{math.}\\ \text{Modell}\end{matrix} \longrightarrow \begin{matrix}\text{num.}\\ \text{Modell}\end{matrix} \longrightarrow \text{Simulation}$$

a) Beobachtung

- 2 Körper üben eine anziehende Kraft aufeinander aus
- die Kraft ist proportional zu den Massen der Körper
- die Kraft führt zu einer Bewegungsänderung / Beschleunigung

Beispiel:
- Apfel & Erde
- Erde & Mond
- Sonnensystem
- Galaxien

b) Modell



Gegeben 2 Körper im Raum mit Positionen r_1, r_2, Geschwindigkeiten v_1, v_2 und Massen m_1, m_2. Die Kraft, die Körper 1 auf Körper 2 auswirkt wird durch das Newtonsche Gravitationsgesetz beschrieben:

$$F_{21} = G\frac{m_1 m_2}{||r_2 - r_1||^3}(r_1 - r_2),$$

wobei $G = 6,67384 \cdot 10^{-11} \frac{m^3}{kg \cdot s^2}$ die Gravitationskonstante bezeichnet. Aufgrund des 3. Newtonschen Gesetzes wissen wir, dass die entgegengesetzte Kraft auf Körper 1 wirkt:

$$F_{12} = -F_{21}$$

Modellannahmen

(phys. Modell)

- nur 2 Körper $\rightarrow$ leicht erweiterbar
- keine relativistischen Effekte
- keine Strahlung
- Punktmassen
- keine Reibung

c) Erweiterung auf N Körper

Für $N \in \mathbb{N}$ Körper ergibt sich die Gesamtkraft, die auf Körper $i \in \{0, ..., N-1\}$ wirkt als

$$F_i = \sum_{j=0, j \neq i}^{N-1} F_{ij} = \sum_{j=0, j \neq i}^{N-1} G \frac{m_i m_j}{||r_j - r_i||^3}(r_j - r_i) \qquad \forall i = 0, ..., N-1.$$

d) Bewegungsgleichungen

Das 2. Newtonsche Gesetz setzt die Kraft in Relation zur Beschleunigung des Körpers

$$F_i = m_i a_i \qquad \forall i = 0, ..., N-1,$$

wobei gilt

$$a_i = \frac{dv_i}{dt}, \qquad v_i = \frac{dr_i}{dt}.$$

Hieraus ergibt sich folgendes System gewöhnlicher Differentialgleichungen:

$$\begin{aligned} \frac{dr_i}{dt} &= v_i \\ \frac{dv_i}{dt} = \frac{F_i}{m_i} &= \sum_{j=0, j \neq i}^{N-1} G \frac{m_j}{||r_j - r_i||^3}(r_j - r_i) \end{aligned}$$

$\Rightarrow$ math. Modell

e) Zur Simulation wollen wir in das Schwerpunktsystem wechseln

$$R = \frac{1}{M}\sum_{i=0}^{N-1} m_i r_i, \qquad V = \frac{1}{M}\sum_{i=0}^{N-1} m_i v_i, \qquad M = \sum_{i=0}^{N-1} m_i.$$

Ohne externe Kräfte ist $\frac{dV}{dt} = 0$ und wir korrigieren

$$\begin{aligned} \tilde{r}_i|_{t=0} &= r_i|_{t=0} - R|_{t=0} \\ \tilde{v}_i|_{t=0} &= v_i|_{t=0} - V|_{t=0}\,. \end{aligned}$$

OBdA nehmen wir an, dass $R = V = 0$ gilt und lassen die Tilde weg.

f) Das Modell schlägt fehl, wenn 2 Körper kollidieren, da $F_{ij} \to \infty$ für $||r_j - r_i|| \to 0$. Daher wird die Beschleunigung regularisiert und wir erhalten

$$a_{ij} = G\frac{m_i}{(||r_j - r_i||^2 + \epsilon^2)^{\frac{3}{2}}}(r_j - r_i), \quad \epsilon > 0$$

Anmerkung
Man kann hierzu auch eine passende Kraft konstruieren, die zu einem räumlich ausgedehnten Körper mit variabler Dichte passt.

g) numerische Näherungslösung

- Masse, Geschwindigkeit, Position als Fließkommazahlen
- aktuelle Position und Geschwindigkeit durch Zeitintegration

$$\begin{aligned} r_i(t_2) &= r_i(t_1) + \int_{t_1}^{t_2} v_i(\tau)d\tau \\ v_i(t_2) &= v_i(t_1) + \int_{t_1}^{t_2} a_i(\tau)d\tau \end{aligned}$$

Das einfachste Verfahren zur Zeitintegration ist das explizite Eulerverfahren

$$\begin{aligned} r_i(t_2) &\approx r_i(t_1) + v_i(t_1)\cdot(t_2 - t_1) \\ v_i(t_2) &\approx v_i(t_1) + a_i(t_1)\cdot(t_2 - t_1) \end{aligned}$$

Es gibt bessere Verfahren, die wir in den Übungsaufgaben betrachten werden.

h) Anfangswerte

Nur wenige Anfangswerte führen zu "interessanten" Simulationen. Geeignete Anfangswerte werden durch das Plummerpotential beschrieben

$$\Phi(r) = GM\frac{1}{(r^2+a^2)^{\frac{1}{2}}}.$$

Alle Körper haben die gleiche Masse $m_i = \frac{1}{N}$. Hieraus werden Verteilungen von r_i und v_i bestimmt.

2.1. Diskretisierungsfehler der Lösung der GDGL

2.1.1. Euler-Verfahren

$$\begin{aligned} t^k &= k \cdot \Delta t \\ v_i^k &= v_i(t^k) \\ r_i^k &= r_i(t^k) \\ v_i^{k+1} &= v^k + \Delta t a_i^k \\ r_i^{k+1} &= r^k + \Delta t v_i^k \end{aligned}$$

Fehlerabschätzung durch die Taylorreihe:

$$\begin{aligned} r_i^{k+1} &= r_i(t^k + \Delta t) \\ &= r_i(t^k) + \Delta t\frac{d}{dt}r_i(t^k) + \frac{1}{2}\Delta t^2\frac{d^2}{dt^2}r_i(t^k) + \frac{1}{6}\Delta t^3\frac{d^3}{dt^3}r_i(t^k) + \mathcal{O}(\Delta t^4) \end{aligned}$$

Taylorreihe für expliziten Euler:

$$\begin{aligned} r_i^{k+1} &= r_i^k + \Delta t v^k \\ &= r_i^k + \Delta t\frac{d}{dt}r_i^k \end{aligned}$$

Daraus ergibt sich der Fehler

$$r_i(t^k + \Delta t) - r_i^{k+1} = \frac{1}{2}\Delta t^2\frac{d^2}{dt^2}r_i(t^k) + \mathcal{O}(\Delta t^3)$$

d.h.

$$\begin{aligned} \text{err}(r_i(\Delta t)) &= \mathcal{O}(\Delta t^2) && \text{pro Zeitschritt} \\ \text{err}(r_i(t)) &= \mathcal{O}(\Delta t) && \text{insgesamt (akum. Fehler)} \end{aligned}$$

Analog folgt für v: $\text{err}(v_i(t)) = \mathcal{O}(\Delta t)$.

Anmerkung
stückweise konstant

$\Rightarrow$ 0. Ordnung Interpolation

$\Rightarrow$ 1. Ordnung Interpolationsfehler

Frage: Verfahren höherer Genauigkeit?

2.1.2. Aufwand des Verfahrens

- Berechnung der Beschleunigung $a_i^k = G \sum_{j \neq i} \frac{m_j}{|r_j^k - r_i^k|}(r_j^k - r_i^k)$ hängt von Position aller Körper $j \neq i$ ab und hat den Aufwand $\mathcal{O}(N-1) = \mathcal{O}(N)$
- Berechnung Gesamtbeschleunigung a^k: $\mathcal{O}(N^2)$
- Berechnung Geschwindigkeit v^{k+1}: $\mathcal{O}(N)$
- Berechnung Position r^{k+1}: $\mathcal{O}(N)$

$\Rightarrow$ Gesamtaufwand: $\mathcal{O}(N^2)$ pro Zeitschritt, $\mathcal{O}(\frac{N^2}{\Delta t})$ insgesamt

2.2. Höhere Ordnung Zeitschritt-Verfahren

Ziel: Verfahren 2. Ordnung

Erste Idee: weitere Taylorglieder

Anwenden auf N Körper:

- neue Position:

$$\begin{aligned} r_i^{k+1} &= r_i^k + \Delta t \frac{d}{dt} r_i^k + \Delta t^2 \frac{d^2}{dt^2} r_i^k \qquad \forall i = 0, ..., N-1 \\ &= r_i^k + \Delta t v_i^k + \Delta t^2 a_i^k \end{aligned}$$

- neue Geschwindigkeit:

$$\begin{aligned} v_i^{k+1} &= v_i^k + \Delta t \frac{d}{dt} v_i^k + \Delta t^2 \frac{d^2}{dt^2} v_i^k \qquad \forall i = 0, ..., N-1 \\ &= v_i^k + \Delta t a_i^k + \Delta t^2 \ ???\ldots \end{aligned}$$

$\Rightarrow$ komplizierter Δt^2 Ausdruck, d.h. der Fehler ist immer noch $\mathcal{O}(\Delta t^2)$.

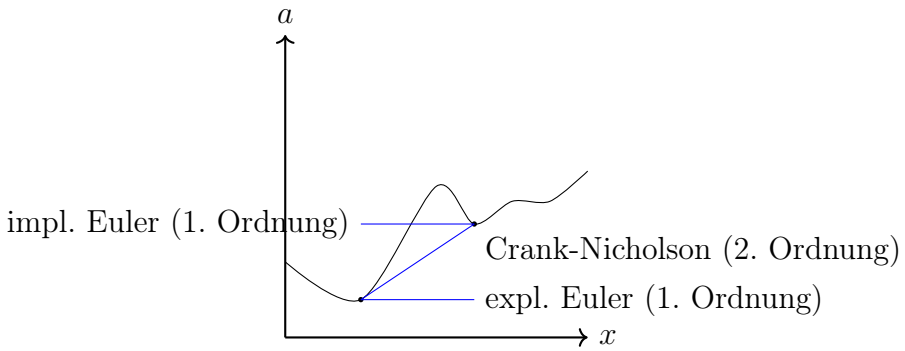
$$\begin{aligned} \frac{d}{dt} a_i^k = G \sum_{i \neq j} & \frac{m_{ij}}{|r_j^k - r_i^k|^3} \left(\frac{d}{dt} r_j^k - \frac{d}{dt} r_i^k \right) \\ & - \frac{3 m_j (r_j^k - r_i^k) \cdot \operatorname{sign}(r_i^k - r_j^k)}{|r_j^k - r_i^k|^4} \left(\frac{d}{dt} r_i^k - \frac{d}{dt} r_j^k \right) \end{aligned}$$

$\Rightarrow$ nicht praktikabel

Alternativer Ansatz: höhere Ordnung Integrationsregeln

Mittelpunktsregel:

$$v_i(t + \Delta t) = v(t) + \int_t^{t+\Delta t} a(\tau) d\tau$$



Crank-Nicolson ist 2. Ordnung exakt, benötigt aber das Lösen eines (nichtlinearen) Gleichungssystems um $a_i(t+\Delta t)$ zu bestimmen. Das Problem ist die Kopplung von r, v und a.
$\Rightarrow$ Entkopplung der Gleichungen

2.2.1. Heun-Verfahren

Berechne zunächst analog zum expliziten Euler-Verfahren

$$\tilde{r}_i^{k+1} = r_i^k + \Delta t v(t^k, r_i^k)$$

alte Geschwindigkeit und extrapolierte (geschätzte) Geschwindigkeit zum neuen Zeitschritt

$$\begin{aligned} r_i^{k+1} &= r_i^k + \frac{1}{2}\Delta t \left(v(t^k, r_i^k) + v(t^{k+1}, \tilde{r}_i^{k+1})\right) \\ &= \frac{1}{2} r_i^k + \frac{1}{2}\left(\tilde{r}_i^{k+1} + \Delta t v(t^{k+1}, \tilde{r}_i^{k+1})\right) \end{aligned}$$

Die r_i^k sind Näherungswerte für die tatsächliche Position $r(k\Delta t)$. Die Werte werden iterativ eingesetzt, zunächst wird die Position $\tilde{r}_i^{k+1}$ in Abhängigkeit von r_i^k berechnet, um dann v_i^{k+1} zu berechnen und anschließend r_i^{k+1} zu bestimmen. Der globale Fehler des Heun-Verfahrens ist $\mathcal{O}(\Delta t^2)$, allerdings gibt es einen doppelten Rechenaufwand, da zweimal v und a berechnet werden.

2.3. Symplektische Verfahren

Im folgenden wollen wir uns nochmal auf das einfachere Beispiel zu Beginn beschränken und einige weitere Eigenschaften der besprochenen Zeitschrittverfahren betrachten.

Wir betrachten erneut das Modellproblem des Fadenpendels.

Bewegungsgleichungen:

$$-m \cdot g \cdot \sin(\varphi(t)) = m \cdot l \cdot \ddot{\varphi}(t), \quad \text{bzw.} -\sin(\varphi(t)) = \ddot{\varphi}(t) \quad \text{(entdimensinalisiert)}$$

Diese führen auf das System gekoppelter Differentialgleichungen erster Ordnung:

$$\dot{\omega}(t) = -\frac{g}{l}\sin(\varphi(t)), \tag{2.1}$$
$$\dot{\varphi}(t) = \omega(t), \tag{2.2}$$

mit der Winkelgeschwindigkeit ω.

Energie:

$$E_{pot} = m \cdot g \cdot h = m \cdot g \cdot l \cdot (1 - \cos(\varphi(t)))$$
$$E_{kin} = \frac{1}{2} m \cdot v^2 = \frac{1}{2} m \cdot l^2 \cdot \dot{\varphi}^2(t) = \frac{1}{2} m \cdot l^2 \cdot \omega^2(t)$$

mit Winkelgeschwindigkeit ω

$$E = E_{pot} + E_{kin} = \frac{1}{2}\omega^2 - \cos(\varphi(t)) \qquad \text{(entdimensinalisiert)}$$
$$= \text{const} \iff \partial_t E = 0$$

Vereinfachen wir das System mit Hilfe der Näherung für kleine Winkel

$$sin(x) \approx x,$$

so erhalten wir

$$\dot{\omega}(t) = -\frac{g}{l}\varphi(t), \tag{2.3}$$
$$\dot{\varphi}(t) = \omega(t), \tag{2.4}$$

und die Energien

$$E_{pot} = m \cdot g \cdot h = \frac{1}{2} m \cdot g \cdot l \cdot \varphi^2(t)$$
$$E_{kin} = \frac{1}{2} m \cdot v^2 = \frac{1}{2} m \cdot l^2 \cdot \dot{\varphi}^2(t) = \frac{1}{2} m \cdot l^2 \cdot \omega^2(t)$$

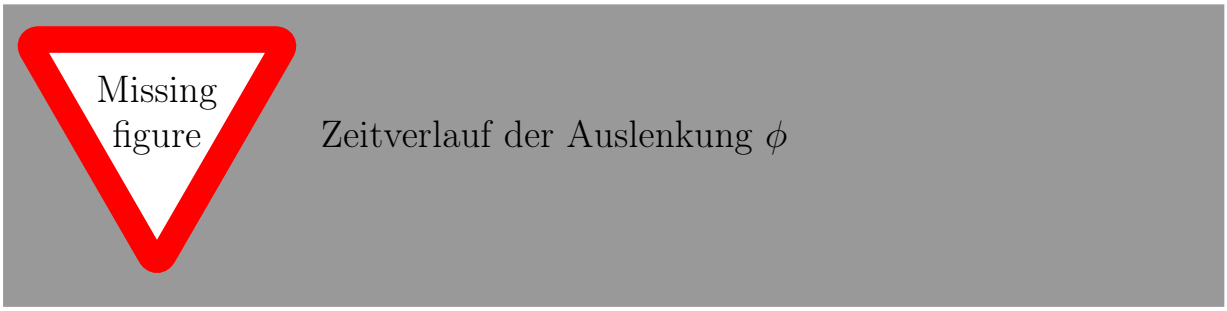


Abbildung 2.1.: Exakte Lösung der Auslenkung ϕ und unterschiedliche numerische Lösungen.

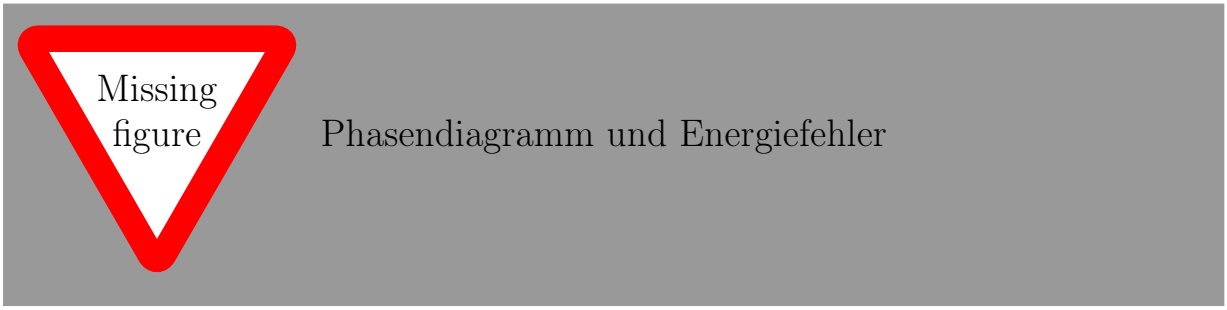


Abbildung 2.2.: Phasendiagramm und Energiefehler unterschiedliche numerische Lösungen.

Betrachten wir das Phasendiagramm, d.h. den Graphen, der Auslenkung und Geschwindigkeit gegen einander aufträgt, so erwarten wir, eine geschlossen, kreisförmige Trajektorie (siehe Übungsblatt 1). Betrachten wir unterschiedliche numerische Zeitschrittverfahren (siehe 2.3), so sehen wir jedoch, dass diese nicht eine kreisförmige Trajektorie beschreiben, sondern eine Spirale. Im Falle des expliziten Eulerverfahrens nimmt die Amplitude mit längerer Simulationszeit zu, für das implizite Eulerverfahren nimmt die Amplitude ab und ähnlich verhält es sich für das Crank-Nicolson- und das Heun-Verfahren.

2.3.1. Hamilton-System

Weiterlesen

Für weitere Details verweisen wir auf [Hairer et al., 2006]

Das Problem lässt sich als sich über das zugehörige Hamilton-System interpretieren.

Definition 2.1 Hamilton-System: Der Hamilton-Operator H beschreibt die totale Energie eines Systems. Die Dynamik, d.h. die zeitliche Entwicklung des Systems ist gegeben durch den Gradienten der Energie. Dies entspricht dem Prinzip der Energie-Minimierung.

Wir entdimensionalisieren, d. h. $m := 1$, $g := 1$, $l := 1$. Der zugehörige *Hamilton-Operator* ist gegeben als

$$H(\varphi, \omega) = \frac{1}{2}\omega^2 - \cos(\varphi).$$

Hieraus erhalten wir die *Hamiltongleichung* als

$$(\dot{\varphi}, \dot{\omega}) = -J\nabla H(\varphi, \omega) = (\omega, -sin(\varphi)),$$

mit

$$J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad \text{bzw.} \qquad J = \begin{pmatrix} 0 & \mathbb{1}_d \\ -\mathbb{1}_d & 0 \end{pmatrix}.$$

Diese entsprecht gerade dem System gewöhnlicher Differentialgleichungen erster Ordnung, welches wir bereits zuvor hergeleitet hatten.

Anmerkung
Für die Matrix J gelten die folgenden Rechenregeln:

$$J^{-1} = J^T = -J \tag{2.5}$$

$$\text{und} \quad J^{-T}J = JJ = -\mathbb{1}. \tag{2.6}$$

Definition 2.2 Erstes Integral/Erhaltungsgröße: Eine nicht-konstante Funktion $I(y)$ ist ein erstes Integral von $\dot{y} = f(y)$ wenn $I'(y)f(y) = 0$ gilt $\forall y$.

Das ist äquivalent zu der Erhaltungseigenschaft, dass für jede Lösung $y(t)$ von $\dot{y} = f(y)$ gilt $I(y(t)) = const.$

Beispiel 2.1 Energierhaltung: Für Hamilton-Systeme ist der Hamilton-Operator $H(p, q)$ ein erstes Integral.

Betrachten wir die zeitliche Entwicklung des Hamilton-Operators, so ist diese durch einen *Hamiltonschen Fluss* Φ_t beschrieben, d. h.

$$\Phi_t : (\varphi(0), \omega(0)) \to (\varphi(t), \omega(t))$$

Es lässt sich zeigen , dass die Energieerhaltung mit der Eigenschaft

$$\det \frac{\partial \Phi_t}{\partial(\varphi(0), \omega(0))} = 1 \quad \forall t$$

korrespondiert. Man spricht davon, dass Φ_t symplektisch (flächenerhaltend) ist.

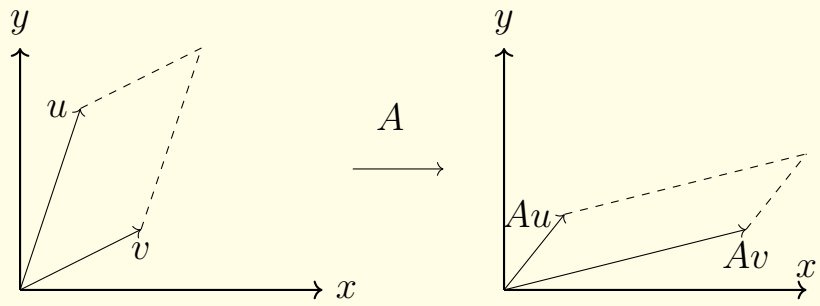
Definition 2.3 Symplektische Abbildung: Eine lineare Abbildung $A : \mathbb{R}^{2d} \to \mathbb{R}^{2d}$ heißt *symplektisch*, wenn gilt

$$A^T J A = J$$

Anmerkung
Für $d = 1$ gilt:

$$A \text{ symplektisch} \iff \det(A) = 1$$

und die Abbildung lässt sich geometrisch interpretieren:



Die Fläche des von u und v aufgespannten Parallelogramms bleibt unter der Transformation A erhalten.

Definition 2.4 Symplektische Transformation: Eine differenzierbare Funktion $g : U \to \mathbb{R}^{2d}$, $U \subset R^{2d}$ heißt *symplektisch*, falls ihre Jacobimatrix $g'(v)$ $\forall v \in U$ symplektisch ist, d. h.

$$g'(v)^T J g'(v) = J \quad \forall v \in U.$$

Satz 2.1. *Satz von Poincaré Sei $H(p,q)$ eine zweimal stetig differenzierbare Funktion auf $U \subset \mathbb{R}^{2d}$ (Hamiltonfunktion). Dann ist für jedes feste t, der Fluss Φ_t eine symplektische Transformation.*

Beweis. Wir schreiben $y = (p,q)$ bzw. $y_0 = (p_0, q_0)$.

Die Ableitung $\frac{\partial \Phi_t(y)}{\partial y_0}$ ist eine Lösung der dazugehörigen Hamiltongleichung und hat die Form

$$\dot{\Psi} = -J\nabla^2 H(\Phi_t(y_0))\Psi,$$

wobei $\nabla^2(p,q)$ die Hesse-Matrix (symmetrisch) von $H(p,q)$ beschreibt.

Wir wählen

$$\Psi = \frac{\partial \Phi_t(y)}{\partial y_0} = \begin{pmatrix} \frac{\partial \Phi_t(y)}{\partial p_0} \\ \frac{\partial \Phi_t(y)}{\partial q_0} \end{pmatrix}$$

und betrachten die Zeitableitung von $\Psi^T J \Psi$. Unter Nutzung der Rechenregeln (2.5) und (2.6) erhalten wir:

$$\begin{aligned}
\frac{d}{dt}\left(\left(\frac{\partial \Phi_t}{\partial y_0}\right)^T J \left(\frac{\partial \Phi_t}{\partial y_0}\right)\right) &= \left(\frac{\partial \Phi_t}{\partial y_0}\right)'^T J \left(\frac{\partial \Phi_t}{\partial y_0}\right) + \left(\frac{\partial \Phi_t}{\partial y_0}\right)^T J \left(\frac{\partial \Phi_t}{\partial y_0}\right)' \\
&= \left(-J\nabla^2 H(\Phi_t)\left(\frac{\partial \Phi_t}{\partial y_0}\right)\right)^T J \left(\frac{\partial \Phi_t}{\partial y_0}\right) + \left(\frac{\partial \Phi_t}{\partial y_0}\right)^T J \left(-J\nabla^2 H(\Phi_t)\left(\frac{\partial \Phi_t}{\partial y_0}\right)\right) \\
&= -\left(\frac{\partial \Phi_t}{\partial y_0}\right)^T \nabla^2 H(\Phi_t) J^T J \left(\frac{\partial \Phi_t}{\partial y_0}\right) - \left(\frac{\partial \Phi_t}{\partial y_0}\right)^T J J \nabla^2 H(\Phi_t)\left(\frac{\partial \Phi_t}{\partial y_0}\right) = 0
\end{aligned}$$

Da zum Zeitpunkt $t = 0$ noch kein Transport stattgefunden hat, gilt $\frac{\partial \Phi_0(y)}{\partial y_0} = \mathbb{1}$ und somit ist für $t = 0$ die Relation

$$\left(\frac{\partial \Phi_t}{\partial y_0}\right)^T J \left(\frac{\partial \Phi_t}{\partial y_0}\right) = J$$

erfüllt.

Wie wir nachgerechnet hatten, verschwindet die Zeitableitung von $\left(\frac{\partial \Phi_t}{\partial y_0}\right)^T J \left(\frac{\partial \Phi_t}{\partial y_0}\right)$ und mit gilt die Relation auch für alle Zeiten t und Startwerte y_0.

□

Ziel: Wir suchen Zeitschrittverfahren, die angewandt auf unser/ein Hamilton-System eine symplektische Abbildung beschreiben.

Satz 2.2. *Das implizite/explizite Euler-Verfahren ist nicht symplektisch.*

Beweis.

explizit:

$$\begin{aligned}(\varphi^{k+1}, \omega^{k+1}) &= (\varphi^k, \omega^k) + \Delta t(\omega^k, -\sin(\varphi^k)) \\ \det \frac{\partial \Phi_{ex,t}}{\partial(\varphi^0, \omega^0)} &= \begin{vmatrix} 1 & k\Delta t \\ -k\Delta t \ \cos(\varphi^0) & 1 \end{vmatrix} = 1 + (k\Delta t)^2 \ \cos(\varphi^0)\end{aligned}$$

implizit:

$$\begin{aligned}(\varphi^{k+1}, \omega^{k+1}) &= (\varphi^k, \omega^k) + \Delta t(\omega^{k+1}, -\sin(\varphi^{k+1})) \\ \det \frac{\partial \Phi_{im,t}}{\partial(\varphi^0, \omega^0)} &= (1 + (k\Delta t)^2 \cos(\varphi^1))^{-1}\end{aligned}$$

□

2.3.2. Energieerhaltung für diskrete Verfahren

Explizites Euler-Verfahren

$$\begin{aligned}\varphi^{k+1} &= \varphi^k + \Delta t \omega^k \\ \omega^{k+1} &= \omega^k + \Delta t a^k,\end{aligned}$$

mit $a^k = -g \cdot \sin(\varphi^k)$.

$$\begin{aligned}
E^{k+1} =& -m \cdot g \cdot l \cdot \cos(\varphi^{k+1}) + \frac{1}{2} m \cdot l^2 \cdot \omega^{(k+1)^2} \\
=& -m \cdot g \cdot l \cdot \cos(\varphi^k + \Delta t \omega^k) + \frac{1}{2} m \cdot l^2 \cdot (\omega^k - \Delta t g \cdot \sin(\varphi^k)) \\
=& -m \cdot g \cdot l \cdot (\cos(\varphi^k) \cdot \cos(\Delta t \omega^k) - \sin(\varphi^k) \cdot \sin(\Delta t \omega^k)) \\
&+ \frac{1}{2} m \cdot l^2 \cdot (\omega^k - \Delta t \cdot g \cdot \sin(\varphi^k)) \\
\approx& E^k (1 + \Delta t^2)
\end{aligned}$$

für kleine Winkel, bei denen $\sin(\varphi) \approx \varphi$ und $\cos(\varphi) \approx 1$.

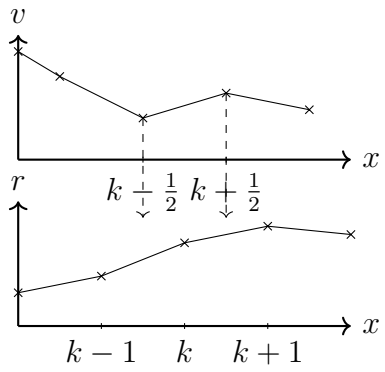
Heun-Verfahren

Analog erhalten wir für das Heun-Verfahren

$$E^{k+1} \approx E^k (1 + \frac{\Delta t^4}{4})$$

2.3.3. Leapfrog-Verfahren

Alternativer Ansatz für 2. Ordnung Verfahren (ähnlich zum Euler-Collatz-Ansatz) durch Entkopplung auf zeitversetzten Gittern:



$$\begin{aligned}
v_i^{k+\frac{1}{2}} &= v_i^{k-\frac{1}{2}} + \Delta t a_i^k \\
r_i^{k+1} &= r_i^k + \Delta t v_i^{k+\frac{1}{2}}
\end{aligned}$$

Umformulierung zu diskreten Zeitschritten:

$$\begin{aligned}
v_i^{k+1} &= v_i^k + \Delta t \frac{a_i^k + a_i^{k+1}}{2} \quad \text{(Mittelpunktsregel)} \\
r_i^{k+1} &= r_i^k + \Delta t v_i^k + \frac{\Delta t^2}{2} a_i^k
\end{aligned}$$

Taylorreihe liefert $\text{err}(r(t)) = \mathcal{O}(\Delta t^2)$ insgesamt.

Satz 2.3. *Das Leapfrog-Verfahren angewendet auf ein Hamilton-System ist symplektisch.*

Beweis. Übungsaufgabe. □

Anmerkung
Weitere Eigenschaften des Leap-Frog Verfahrens:

- Das Verfahren ist 2. Ordnung exakt,
- es benötigt eine Kraftauswertung pro Zeitschritt,
- das Verfahren ist reversibel / zeitinvertierbar, d.h. dass man damit auch rückwärts in der zeit rechnen kann und tatsächlich wieder bei der Startlösung angelangt.
- Das Verfahren ist explizit
- und es ist symplektisch, d.h. der Energiefehler beschränkt.

2.4. Implementierung

Komponenten des Algorithmus:

- Kraft auf Körper i zum Zeitpunkt t
- Änderung der Position von Körper i von Zeit t bis $t + \Delta t$
- Änderung der Geschwindigkeit von Körper i von Zeit t bis $t + \Delta t$

Algorithmus 2.1: Leapfrog-Verfahren

```
Leapfrog (Particles p, dt){
  a = computeAccel(r(p),m(p))
          // aus vorherigem Zeitschritt
  for(i in p)
  r_i += dt*v_i + dt*dt/2*a_i
  a' = computeAccel(r(p), m(p))
  for (i in p)
  v_i += dt/2*(a_i + a'_i)
}

computeAccel(r(p),m(p)){
  a = vector(vector(3), sizeof(r));
  a = 0.0;
  for(i in r) {
        for(j in r, j>i){
          da = m_j * (r_j-r_i) * (|r_j-r_i|^2 + eps^2 )^(3/2);
          a_i += da;
          a_j -= da; // Symmetrie der Kraefte
        }
  }
  return a_i;
}
```

Anmerkung

In C++ funktionieren vektorwertige Operationen nicht direkt:

```
v_i += dt/2 * (a_i + a'_i)
```

$\longrightarrow$

```
for(n = 0; n < 3; n++)
    v[i][n] += dt/2 * (a[i][n] + aprime[i][n]);
```

Anmerkung

Große Speicherflächen nicht kopieren und außerhalb anlegen:

```
return a;
```

$\longrightarrow$ besser Call-by-Reference verwenden:

```
computeAccel(vector<array<double,3>>& a,...)
```

Lemma 2.4 (Komplexität des Leapfrog Verfahrens)**.** *Das Berechnen eines Zeitschritts des N-Körperproblems mit Hilfe des Leapfrog Verfahrens hat die Komplexität* $\mathcal{O}(N^2)$.

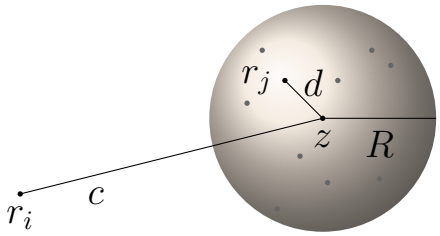
Beweis. Nach Konstruktion gilt für N Partikel:

Leapfrog:	$N \times 6 \times 3$	FLOPS (Position)	
	$+N \times 4 \times 3$	FLOPS (Geschw.)	$\to \mathcal{O}(N)$
ComputeAccel:	$N \times \frac{N-1}{2} \times 23$	FLOPS	$\to \mathcal{O}(N^2)$
Gesamt:	$N \times \left(30 + \frac{N-1}{2} \times 23\right)$	FLOPS	$\to \mathcal{O}(N^2)$

□

2.5. Barnes-Hut-Algorithmus

1. Beobachtung: Die Kräfte weit entfernter, benachbarter Massen lassen sich zusammenfassen und durch die Gesamtmasse im Schwerpunkt beschreiben.



2. Modellierung: Gewichtskraft als Gradient eines Potentials

$$F_{ij} = m_i \frac{G m_j (r_j - r_i)}{|r_j - r_i|^3} = m_i \nabla \left(\frac{G m_j}{|r_j - r_i|} \right) = m_i \nabla \phi_j(r_i)$$

$$\overset{\text{Linearität}}{\Rightarrow} F_i = m_i \nabla \Phi(r_i) = m_i \nabla \sum_{j \neq i} \phi_j(r_i)$$

Wir setzen eine Reihenentwicklung an und wollen höhere Ordnung Terme vernachlässigen. Dazu definieren wir eine Funktion des Abstandes:

$$\tilde{\phi_j}(r_i) = \phi_j(r_i)\frac{1}{Gm_j} = \frac{1}{|r_j - r_i|} := f(r_j - r_i)$$

Wir betrachten jetzt eine Taylorentwicklung von $f(r_j - r_i)$ um z:

$$\begin{aligned} f(r_j - r_i) &= f(\underbrace{(z - r_i)}_{=:c} + \underbrace{(r_j - z)}_{=:d}) \\ &= \sum_{\alpha} \frac{D^\alpha f(c)}{|\alpha|!} d^{|\alpha|} \quad \text{mit Multiindex } \alpha \\ &= \sum_{|\alpha| \leq k} \frac{D^\alpha f(c)}{|\alpha|!} d^{|\alpha|} + \mathcal{O}(d^{k+1}) \quad \text{mit } k > 0, k \in \mathbb{N} \end{aligned}$$

Damit erhalten wir das **Gesamtpotential**:

$$\begin{aligned} \Phi(r_i) &= \sum_{j \neq i} G m_j f(r_j - r_i) \\ &\approx \sum_{j \neq i} G m_j \sum_{|\alpha| \leq k} \frac{D^\alpha f(c)}{|\alpha|!} d^{|\alpha|} \\ &= \sum_{|\alpha| \leq k} \frac{D^\alpha f(c)}{|\alpha|!} G \underbrace{\sum_{j \neq i} m_j d^{|\alpha|}}_{=:M_\alpha} \\ &= \sum_{|\alpha| \leq k} \frac{D^\alpha f(z - r_i)}{|\alpha|!} M_\alpha \end{aligned}$$

Fehler: Der relative Fehler ist $\mathcal{O}\left(\left(\frac{R}{c}\right)^{k+1}\right)$, die Genauigkeit steigt bei

- kleinerem $\frac{R}{c}$
- größerem Entwicklungsgrad k

Gradient:

Kraft als Gradient des Potentials

$$F_i = \nabla\Phi(r_i)$$
$$\frac{\partial}{\partial\omega}\Phi(r_i) \approx \sum_{|\alpha|\leq k} \frac{D^\alpha \partial_\omega f(z-r_i)}{|\alpha|!} M_\alpha, \ \omega \text{ Raumausrichtung}$$

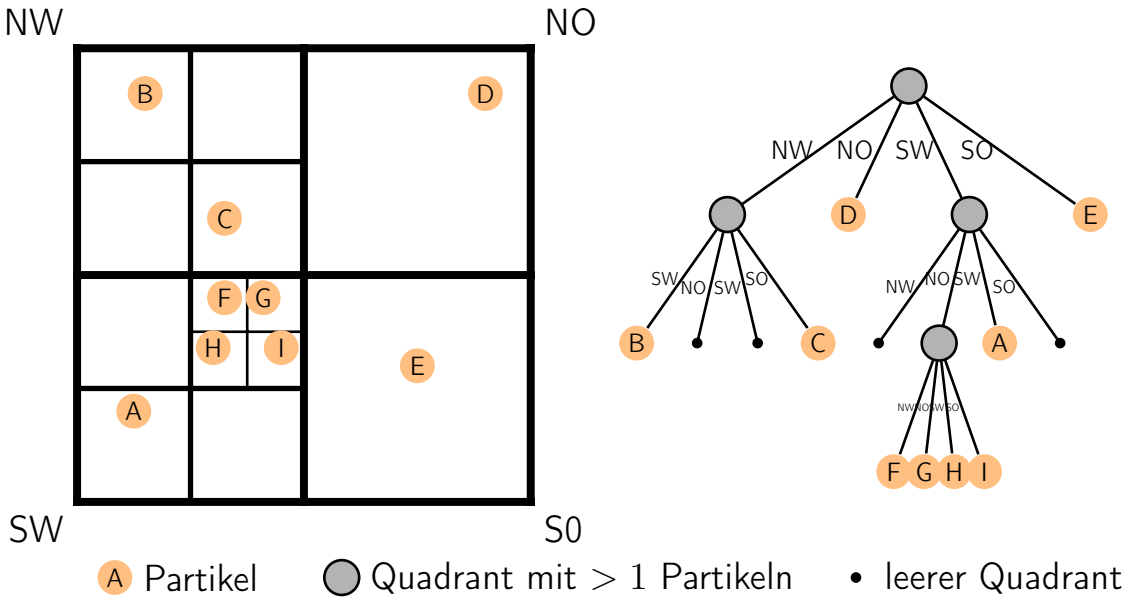
Multipolentwicklung:

Normalerweise Multipol- statt Taylorentwicklung, der einfachster Fall ist ein Monopol

$$\Phi(r_i) \approx \frac{1}{||z-r_i||} G \underbrace{\sum_{j\neq i} m_j}_{\text{Gesamtmasse}} ,$$

wobei z den Schwerpunkt bezeichnet.

3. Baumalgorithmus: [Barnes and Hut, 1986]
   - Octree-Struktur
   - rekursive Adaption, bis es ein Objekt pro Zelle gibt



(1) Position & Masse aller Einträge im Baum berechnen: $\mathcal{O}(N)$ (proportional zur Anzahl der Knoten im Baum)

(2) Kraft auf einen Knoten berechnen: $\mathcal{O}(\log N)$

- Baum von der Wurzel ablaufen
- Traversierung beenden, wenn $\frac{R}{c} \leq \theta$, $0 < \theta \leq 1$

$\Rightarrow$ Gesamtaufwand $\mathcal{O}(N \log N)$

Algorithmus 2.2: Barnes-Hut-Algorithmus

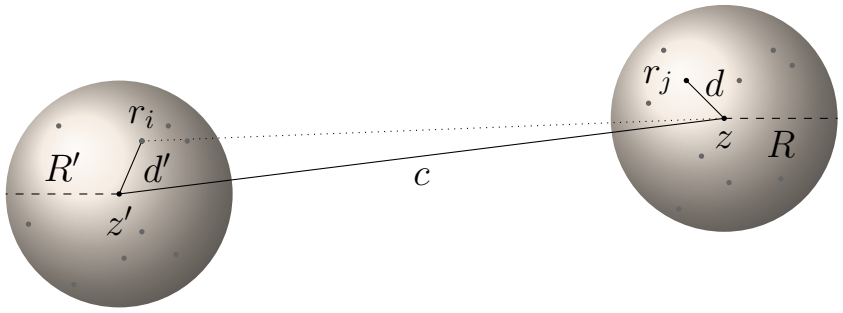
```
struct node {
  bool isLeaf;
  node* children;
  Body b;
};

Potential(Body b, Node n){
  double p = 0;
  if(n.isLeaf && b = n.b)
    // i ≠ j
    return 0;
  if(n.isLeaf)
    // direkte Auswertung
    return Φ(n.b, b);
  if(R(n)/c(n,b) ≤ θ)
    // Fernfeld bzgl. Pseudoteilchen
    return Φ(n.b, b);
  for(n2 in n.children)
    p += potential(b, n2);
  return p;
}
```

(3) Auswerten des Zeitschrittverfahrens: $\mathcal{O}(N)$

$\Rightarrow$ Barnes-Hut-Algorithmus hat Komplexität $\mathcal{O}(N \log N)$

2.6. Fast-Multipol Methode Greengard and Rokhlin [1987]

Einer der Top-10 Algorithmen des 20. Jahrhunderts.



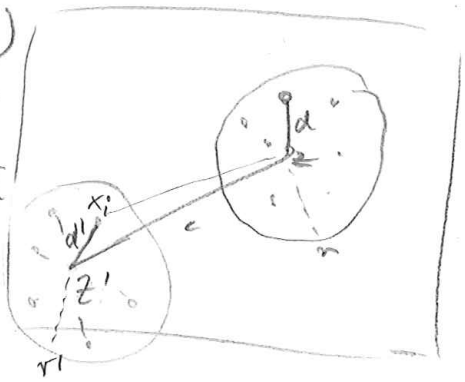
Siehe handschriftliche Notizen...

Fast-Multipol-Methode

a) Darstellung als Multipol-~~Reihe~~ Entwicklung

(Kugelflächenfunktionen)

$$\phi(x_i) = M_0 \log(z-x) + \sum_{\alpha=1}^{k} \frac{M_\alpha}{(z-x)^\alpha}$$



mit Momenten M_k

b) Verschieben einer MPE

Das Potential an x_i, z' im Ursprung, d.h. $t_i := d'$

$\phi(x)$ wie oben

lässt sich für entfernte Punkte z' entwickeln als

$$\tilde{\phi}(x_i) = M_0 \log(x_i) \sum_{\alpha=1}^{k} \frac{\tilde{M}_\alpha}{x_i^\alpha}$$

mit neuen Koeffizienten $\tilde{M}_k$

Der Fehler konvergiert mit $\left|\frac{|z|+R}{x_i}\right|^{\alpha+1} = \mathcal{O}\left(\frac{1}{d'^{\alpha+1}}\right)$

a) ~~Translation der Multipolentwicklung~~

Umwandeln in eine lokale Reihenentwicklung

innerhalb der Kugel r' um den Ursprung (z')

kann das Potential

$\phi(x_i)$ als Potenzreihe entwickelt werden

$$\tilde{\phi}(x_i) \approx \sum_{l=0}^{\infty} b_l x_i^l$$

Algorithmus

1) Upward-Pass

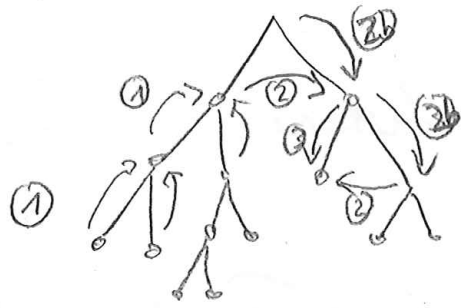
Hierarchisch, depth-first im Baum MPE in den Blättern nach oben aufsummieren

2) Downward-Pass

Statt direkt Interaktion einer Box als Anwendung ihrer MPE auf Nachbarn zu berechnen!

2a) Pro Level verschieben der MPE in Nachbarn als lokale Entwicklung um die Zellmitte.

2b) Verschieben der LE in Kindzellen & Addition der Potentiale



3. Musterbildung

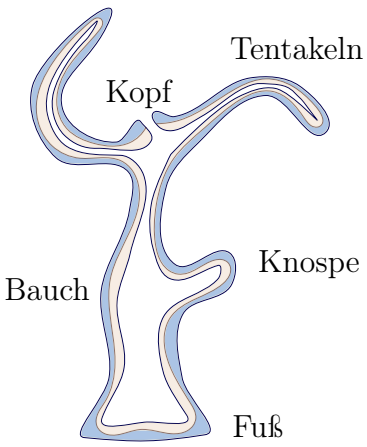
Definition 3.1 Musterbildung: Musterbildung ist ein Prozeß, bei dem ein räumlich homogener Zustand instabil wird und einem inhomogenen Zustand, also einem Muster weicht.

- viele Musterbildungsprozesse in der Natur lassen sich mit Hilfe von Reaktions-Diffusions-Systemen beschreiben
- grundlegendes Prinzip mit Anwendungen in der Biologie, Chemie, Geographie etc.
- erstmals vorgeschlagen von Alan Turing (1952)
- Beispiel: Entwicklung der Hydra

3.1. Die Hydra

Beobachtung:

- Süßwasserpolypen, Familie der Nesseltiere (kein Medusenstadium)
- vielzelliger Organismus ohne Organe
- Hohltier, besteht aus Kopf, Rumpf, Fuß
- hat 4 bis 12 Tentakeln
- bis zu 3 cm groß
- lebt in Seen und Flüssen bis zu 300 m Tiefe
- ernährt sich von kleinen Krebsen, Larven, Wasserflöhen, etc.

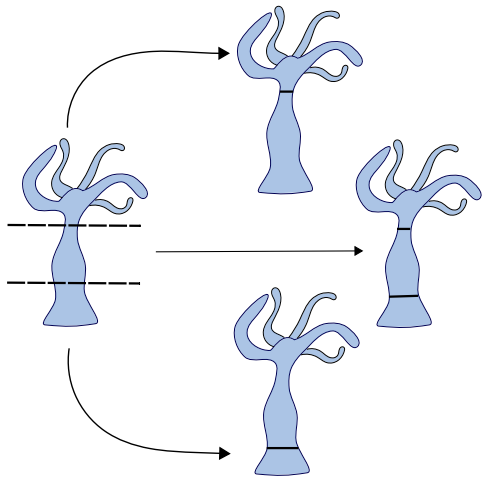


- sehr starke Regenerationsfähigkeit (→ interessant für Entwicklungsbiologen)
- stationäres Embryonenstadium
- Zellen werden ständig erneuert
  - Bildung von Stammzellen im Bauchbereich
  - Zellen wandern nach außen und differenzieren sich aus
  - auch Nervenzellen werden erneuert

  → wird theoretisch "unendlich alt"
- mechanische Zerteilung führt zu Bildung neuer Hydren aus den Stücken

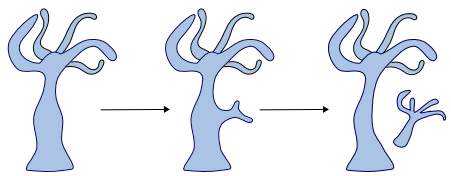
Experimente:

a) Regeneration



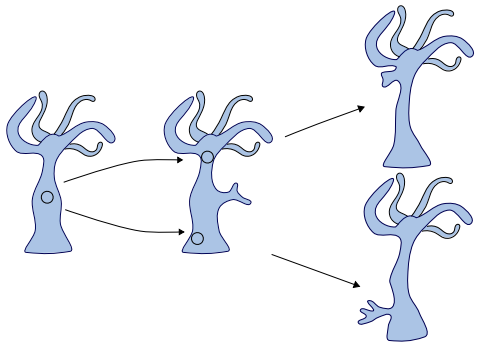
- Neubildung amputierter Extremitäten
- Orientierung (Kopf ↔ Fuß) bleibt erhalten

b) Sprossung



- Hydra kann sich sowohl geschlechtlich als auch durch Sprossung fortpflanzen
- ständig werden neue Zellen produziert
- im Bauch entsteht ein neuer Kopf
- Spross spaltet sich ab

c) Verpflanzung



- Zellen werden am Bauch entnommen
- Verpflanzungen an Kopf/Fuß führen zur Entwicklung des Gegenstücks

3.2. Morphogenese, morphogenes Feld

(phänomenologisches Modell)

Definition 3.2 Morphogenese: Morphogene sind Signalmoleküle, welche die Musterbildung während der Entwicklung vielzelliger Lebewesen steuern.

- lokale Konzentrationsunterschiede der Morphogene führen zu unterschiedlichen Genexpressionen (welche Gene sind aktiv)
- ursprünglich gleiche Zellen übernehmen unterschiedliche Funktionen
- lokale Konzentrationsunterschiede entstehen durch Reaktion und Diffusion der Morphogene und anderer chemischer Strukturen

3.3. Reaktions-Diffusions-Systeme

Ursprünglich wurde angenommen, dass Reaktions-Diffusions-Probleme keine Fluktuationen erzeugen können, da Diffusion ausgleichend (stabilisierend) wirkt. [Turing 1952] postulierte ein Reaktions-Diffusions-System, welches lokale Konzentrationsunterschiede für Morphogene der Hydra erzeugt. Ohne Diffusion ist das Reaktionssystem stabil gegenüber kleiner Störungen, mit Diffusion ist es instabil.

3.3.1. Aktivator-Inhibitor-System

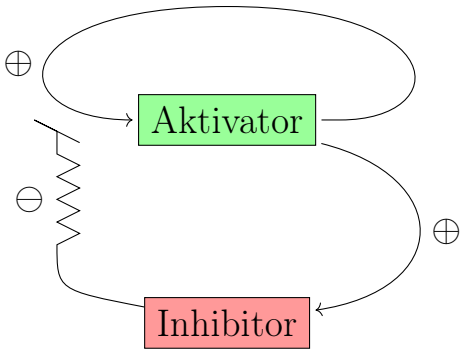
Definition 3.3 Turing-Modell für 2 Chemikalien: Gegeben seien zwei Konzentrationen a und b, so beschreibt das *Turing-Modell* ein Anfangswertproblem

$$\begin{aligned} \frac{\partial a}{\partial t} &= D_a \Delta a + f(a,b) \quad \text{auf } \Omega \subset \mathbb{R}^n \\ \frac{\partial b}{\partial t} &= D_b \Delta b + g(a,b) \end{aligned}$$

$$\begin{aligned} n \cdot \nabla a = n \cdot \nabla b &= 0 \quad \text{auf } \partial\Omega \\ a(t=0) &= a_0 \\ b(t=0) &= b_0 \end{aligned}$$

mit äußerem Normalenvektor n, nichtlinearen Funktionen f, g und Diffusionskonstanten D_a, D_b.

Idee:



Beispiel: Gierer-Meinhard-Modell

$$\begin{aligned} f(a,b) &= s\left(\frac{a^2}{b} + c_a\right) - r_a a \\ g(a,b) &= sa^2 - r_b b + c_b, \end{aligned}$$

wobei
$s\frac{a^2}{b}$: Produktionsrate Aktivator (Autokatalyse),
sa^2: Produktionsrate Inhibitor,

c_a, c_b: Grundproduktion von Aktivator/Inhibitor,
$-r_a a$, $-r_a b$: Zerfall der Aktovator-/Inhibitor-Moleküle.

Beispiel: Schnakenberg-System

$$f(a,b) = c_a - r_a a + s a^2 b$$
$$g(a,b) = c_b - s a^2 b$$

oftmals wird auch die vereinfachte und entdimensionlisierte Version

$$f(a,b) = \alpha - a + a^2 b$$
$$g(a,b) = \beta - a^2 b$$

betrachtet. Hierbei sind $\alpha, \beta \in \mathbb{R}$ Konstanten, die die Grundproduktion beschreiben.

3.3.2. Stabilitätsanalyse

Die Argumentation lautete ja, dass wir

ohne Diffusion eine stabile konstante Lösung haben.

ohne Reaktion einen stabilen Gleichgewichtszustand haben.

im System der Gleichgewichtszustand nicht mehr stabil ist.

Ein Zustand ist im Gleichgewicht, wenn die Zeitableitungen verschwinden. In der Natur kommen stets kleine Störungen vor. Ob dieser Zustand robust gegen solche kleine Störungen ist, schliessen wir durch eine lineare Stabilitätsanalyse. Hierbei untersuchen wir die Eigenwerte der Jakobimatrix am Gleichgewichtspunkt.

Satz 3.1 (Stabilität)**.** *Sei u^* Lösung eines (nichtlinearen) Gleichungssystems. Der Zustand u^* ist stabil gegen kleine Störungen, falls für die Eigenwerte λ_i der Jakobimatrix J am Linearisierungspunkt u^* gilt:*

$$\mathrm{re}(\lambda_i) < 0 \qquad \forall \lambda_i$$

Beweis. Ohne Beweis □

Satz 3.2 (Spur-/Determinanten-Kriterium)**.** *Für $J \in \mathbb{R}^{2\times 2}$ ist eine hinreichende Bedingung für Stabilität in Satz 3.1*

$$\text{spur}(J) < 0 \qquad \textit{und} \qquad \det(J) > 0$$

Beweis. Beweis für Spezialfall als Übungsaufgabe □

Anmerkung
Satz 3.2 ist eine hinreichende aber keine notwendige Bedingung. Das heißt, dass im Falle dass das Spur-/Determinanten-Kriterium verletzt ist, immer noch die Eigenwerte untersucht werden müssen um sicher sagen zu können, ob eine Lösung stabil ist.

Handschriftliche Analyse hinzufügen

Universität Stuttgart
Germany

SimTech
Cluster of Excellence

Modell 2

Schnakenberg-System

$$f(a,b) = \alpha - a + a^2 b$$

$$g(a,b) = \beta - a^2 b$$

Modell 1

Gierer-Meinhardt

$$f(a,b) = s\left(\frac{a^2}{b} + b_a\right) - r_a a$$

$$g(a,b) = s a^2 + b_b - r_b b$$

a) Stabilität ohne Reaktion

$$s = r_a = r_b = b_a = b_b = 0$$

$\Rightarrow$ Stabile Lösung $a = \text{const}$, $b = \text{const}$

einfach bestimmt für G_2 auf Konstante

Universität Stuttgart
Germany

SimTech
Cluster of Excellence

b) Stabilität ohne Diffusion

zweifacher Gleicher Maßstab

$s = 1$

$r_a = \alpha > 0$

$r_b = 1$

$b_a = b_b = 0$

$D_a = D_b = 0$

$f(a,b) = \frac{a^2}{b} - r_a a$

$g(a,b) = a^2 - b$

$\Rightarrow$ (1) $\partial_t a = \frac{a^2}{b} - r_a a$

(2) $\partial_t b = a^2 - b$

(1) Gleichgewichtszustand

$\partial_t a = 0 = \partial_t b$

(2) $a^2 = b$ und (1) $0 = \frac{b}{b} - \alpha a$

$\Rightarrow \bar{a} = \frac{1}{\alpha}$ und $\bar{b} = \frac{1}{\alpha^2}$

Eigenwertanalyse: Entwicklung um Gl.g.w. Punkt

$$J_{b_s} = \begin{pmatrix} \frac{2a}{b} - \alpha & -\frac{a^2}{b^2} \\ 2a & -1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{\partial f}{\partial a} & \frac{\partial f}{\partial b} \\ \frac{\partial g}{\partial a} & \frac{\partial g}{\partial b} \end{pmatrix} =$$

www.simtech.uni-stuttgart.de

Universität Stuttgart
Germany

SimTech
Cluster of Excellence

Stabil falls für EW λ_1, λ_2 gilt $\mathrm{Re}(\lambda_i) < 0$

für $\mathbb{R}^2$ ⟺

$\lambda_1 + \lambda_2 = \mathrm{Spur}(J) < 0$

$\lambda_1 \cdot \lambda_2 = \det J > 0$

~~Betrachte~~ Umformung auf Gl.s.w. Pkt $\bar{a} = \frac{1}{\alpha}, \bar{b} = \frac{1}{\alpha^2}$

$$\Rightarrow J = \begin{pmatrix} \alpha & -\alpha^2 \\ \frac{2}{\alpha} & -1 \end{pmatrix}$$

~~det~~

$\mathrm{Spur}\, J = \alpha - 1 \overset{!}{<} 0$

$\det J = \alpha - \alpha + 2\frac{\alpha^2}{\alpha} = \alpha \overset{!}{>} 0$

Stabil für

$\Rightarrow \quad 0 < \alpha < 1$

www.simtech.uni-stuttgart.de

Universität Stuttgart
Germany

SimTech
Cluster of Excellence

C) lokale Instabilität, globale Stabilität

Reaktion + Diffusion

$D_a > D_b$, eindimensional, ohne RB

Setze Wellen als Störung an, welche Wellenlängen werden verstärkt, welche gedämpft?

Entwickeln als $a = \bar{a} + \tilde{a}$, $b = \bar{b} + \tilde{b}$

um $(\bar{a}, \bar{b})$ und linearisiern:

$$\frac{\partial \tilde{a}}{\partial t} = D_a \frac{\partial^2 \tilde{a}}{\partial x^2} + \left(\frac{\partial f}{\partial a}\right) \tilde{a} + \frac{\partial f}{\partial b} \tilde{b} + \mathcal{O}\left(\tilde{a}^2 + \tilde{b}^2\right)$$

$$\frac{\partial \tilde{b}}{\partial t} = D_b \frac{\partial^2 \tilde{b}}{\partial x^2} + \frac{\partial g}{\partial a} \tilde{a} + \frac{\partial g}{\partial b} \tilde{b} + \mathcal{O}\left(\tilde{a}^2 + \tilde{b}^2\right)$$

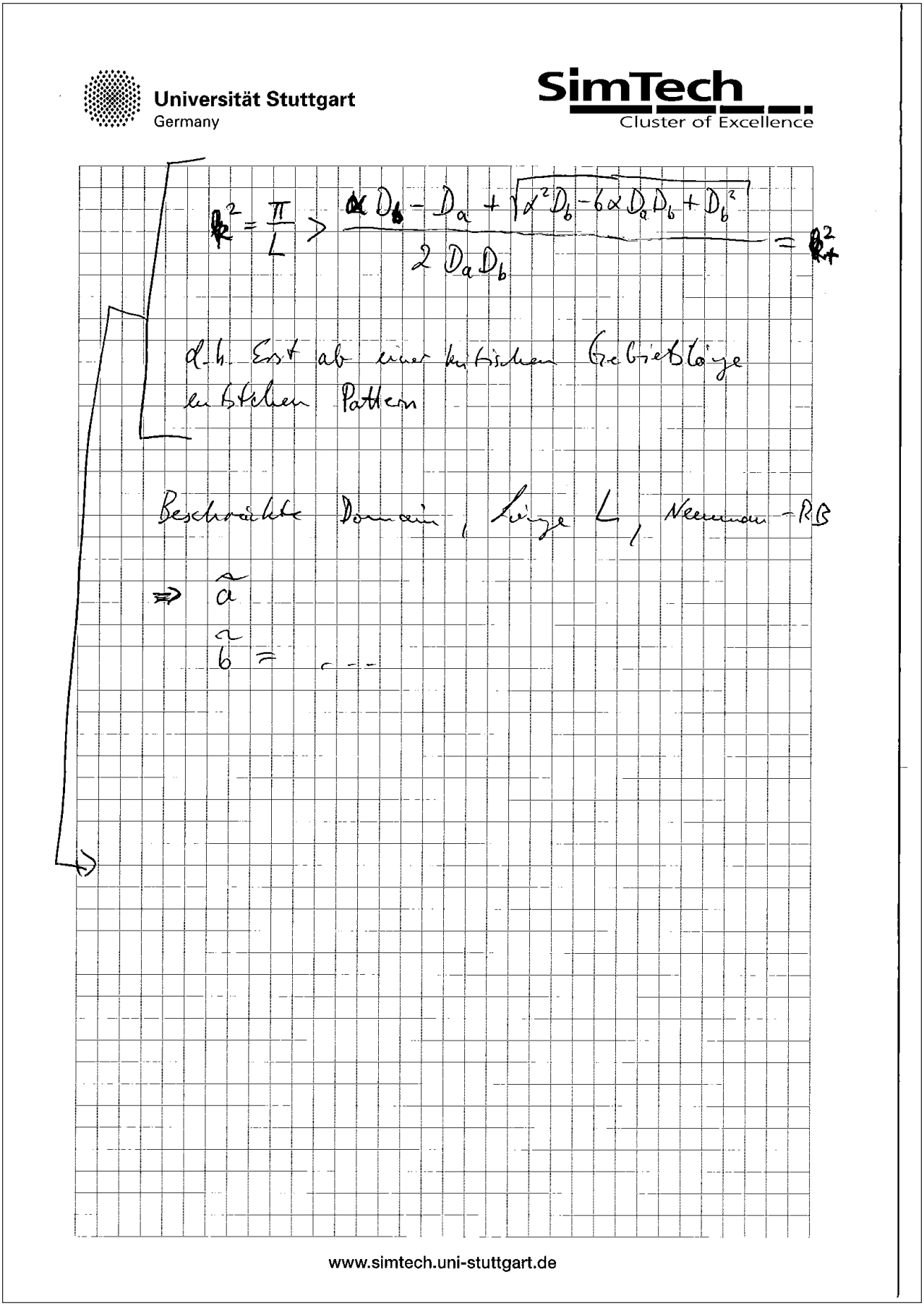
Setze Störung als Welle an (Fourierreihe)

$$\tilde{a}(x,t) = A(t) e^{ikx}$$

$$\tilde{b}(x,t) = B(t) e^{ikx}$$

mit Wellenzahl $k > 0$

www.simtech.uni-stuttgart.de


Universität Stuttgart
Germany
SimTech
Cluster of Excellence
$k^2 = \frac{\pi}{L} > \frac{\alpha D_b - D_a + \sqrt{\alpha^2 D_b - 6\alpha D_a D_b + D_b^2}}{2 D_a D_b} = k_+^2$
d.h. Erst ab einer kritischen Gebietslänge entstehen Pattern
Beschränkte Domain, Länge L, Neumann-RB
$\Rightarrow \hat{a}$
$\hat{b} = \ldots$
www.simtech.uni-stuttgart.de

Universität Stuttgart
Germany

SimTech
Cluster of Excellence

$\Rightarrow$ Diffusion

$$D_a \partial_x^2 \underbrace{A(t) e^{ikx}}_{\widetilde{a}} = -q^2 D_a \underbrace{A(t) e^{ikx}}_{\widetilde{a}}$$

analog für b

$$\Rightarrow \quad J = \begin{pmatrix} f_a - q^2 D_a & f_b \\ g_a & g_b - q^2 D_b \end{pmatrix}$$

für unser Modellproblem:

$$J = \begin{pmatrix} \alpha - q^2 D_a & -\alpha^2 \\ 2 & -1 - q^2 D_b \end{pmatrix}$$

Spur $J = \underbrace{\alpha}_{<1} - 1 - \underbrace{k^2 (D_a + D_b)}_{>0} < 0 \quad (\checkmark)$

$$\det J = (\alpha - k^2 D_a)(-1 - k^2 D_b) - 2\alpha \overset{!}{>} 0$$

$\Rightarrow$ Bedingung an D_a, D_b

$q \in \mathbb{R}^+$ nur erfüllt, falls

$$-k \alpha D_b + D_a > 2\sqrt{D_a D_b \alpha}$$

~~wenn $q = \frac{n\pi}{L}$ folgt für kleines q $n \in \mathbb{N}^+$~~

Auch hier wollen wir mit Hilfe der linearen Stabilitätsanalyse untersuchen, was passiert.

Wir entwickeln die Störung als Fourier-Reihe und können damit untersuchen, welche Moden verstärkt und welche gedämpft werden.

Wir betrachten den Fall in 1D mit $\Omega = [0, \pi]$, andere Längen erhält man durch skalieren. Aufgrund der Neumann Randbedingungen muss die Ableitung am Rand 0 sein und die Lösung läßt sich als Cosinus-Reihe entwickeln:

$$a = \sum_{k=0}^{\infty} a_k(t) cos kx$$

und analog für b.

Wir nehmen nun eine kleine Störung der Form

$$\delta_{a/b} = \alpha_{a/b} \exp(\sigma t) cos kx$$

an, d.h. eine periodische Funktion mit Wellenzahl k.

Annahme: kleine Störung → Linearisieren um Gleichgewichtszustand → nur Entwicklung der Störung betrachten.

Wir betrachten nun die Entwicklung der Störung

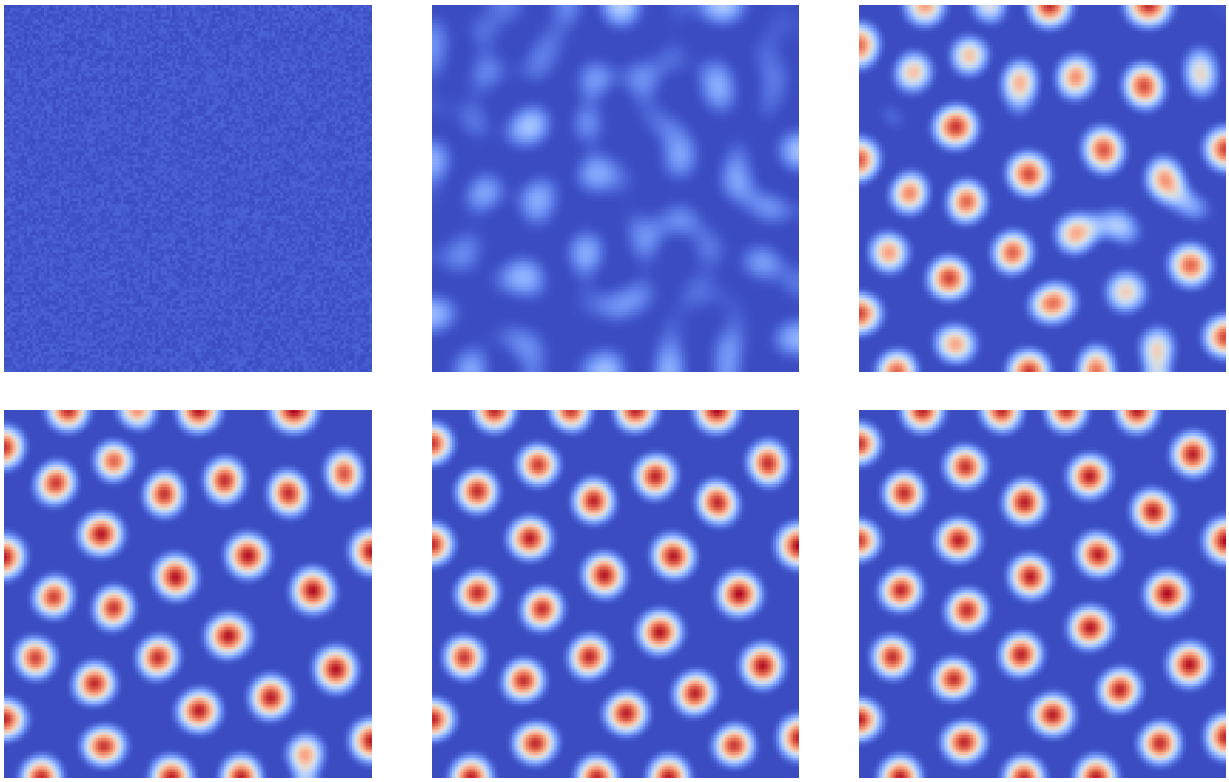


Abbildung 3.1.: Lösung des Gierer-Meinhard-Modells, $t = 0, t = 50, t = 100, t = 200, t = 500, t = 1000$.

3.4. Diskretisierung des Turing-Modells

Wir betrachten zwei gekoppelte PDGL der Form

$$\left.\begin{aligned} \frac{\partial a}{\partial t} &= D_a \Delta a + f(a,b) \\ \frac{\partial b}{\partial t} &= D_b \Delta b + g(a,b) \end{aligned}\right\} \text{ auf } \Omega \times [0,T]$$

mit Neumann-Randbedingungen

$$\left.\begin{array}{l} \nabla a \cdot n = 0 \\ \\ \nabla b \cdot n = 0 \end{array}\right\} \quad \text{auf } \partial\Omega.$$

und Anfangwerten a_0, b_0.

- räumliche Kopplung nur durch den Laplace-Operator
- Reaktion nur lokal, dafür nicht linear

3.4.1. Linienmethode

Das Gesamtproblem auf $\Omega \times [0, T]$ mit $\Omega \subset \mathbb{R}^n$ ist sehr aufwendig zu lösen.

Ansatz: Entkoppeln von Orts- und Zeitproblem

Zuerst findet eine Diskretisierung im Ort statt, dies führt auf ein System von gewöhnlichen Differentialgleichungen, welche mithilfe eines Zeit-Integrationsschemas diskretisiert werden.

3.4.2. Diskretisierung des Ortsproblems

Das räumliche Problem (Laplace-Operator) wird mit Finiten-Differenzen diskretisiert. (Wahlfreiheit! – wir könnten auch ein anderes Verfahren verwenden)

Wir betrachten das 1D-Problem:
Gegeben $\Omega \subset \mathbb{R}$ beschreibt $\mathcal{T}(\Omega)$ eine Partition von Ω in N äquidistante Intervalle der Länge h.

Die Intervallgrenzen werden konsekutiv durchnummeriert. Das definiert die Knoten $i \in [0, N-1]$ des Gitters an den Positionen x_i. Die unbekannte Lösung $a(x)$, $x \in \Omega$ wird an den diskreten Knoten i durch $a_i = a(x_i)$ beschrieben. Idee: Die Ableitungen werden mit Differenzenquotienten approximiert:

$$\nabla a(x) \approx \frac{a(x+h) - a(x-h)}{2h} \quad \text{(zentrale Differenzen)},$$

die zweite Ableitung erhält man durch doppeltes Differenzieren

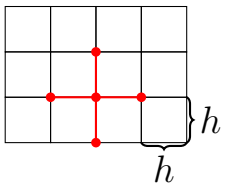
$$\begin{aligned}
\Delta a(x) &= \nabla \cdot (\nabla a)(x) \\
&\approx \frac{\nabla a(x + \frac{h}{2}) - \nabla(x - \frac{h}{2})}{h} \\
&\approx \left(\frac{a(x+h) - a(x)}{h} - \frac{a(x) - a(x-h)}{h} \right) \cdot \frac{1}{h} \\
&= \frac{a(x+h) + a(x-h) - 2a(x)}{h^2}
\end{aligned}$$

$$\Rightarrow \text{diskret:} \quad \Delta a_i = \frac{a_{i+1} + a_{i-1} - 2a_i}{h^2} \quad \text{für } i \in]0, N-1[.$$

Wir betrachten nun die Randbedingung: $\nabla a(x) \cdot n(x) = 0$, oBdA beschränken wir uns auf den linken Rand, d. h. $i = 0$.

$$\begin{aligned}
\Delta a(x_0) &= \nabla \cdot (\nabla a)(x_0) \\
&\approx \frac{\nabla a(x_0 + h) - \nabla a(x_0)}{h} && \text{(rechtsseitiger Differenzenquotient)} \\
&= \frac{\nabla a(x_1) - 0}{h} && \text{(Randbedingung)} \\
&\approx \frac{a_1 - a_0}{h^2} && \text{(linksseitiger Differenzenquotient)}
\end{aligned}$$

Erweiterung auf 2D/3D:



Wir betrachten ein strukturiertes Gitter mit Schrittweite h und $N \times M$ Knoten, die lexikographisch nummeriert werden. Jeder Knoten hat die Nachbarknoten N_i, in 2D: $N_i = \{i-1, i+1, i-M, i+M\}$.

Damit erhalten wir den bekannten 5-Punkt-Stern

$$\Delta a_i = \sum_{j \in \mathcal{N}_i} a_j - a_i \quad \forall i \in [0, (N+1)(M+1) - 1]$$

Anmerkung
Wie wir noch sehen werden ist diese Diskretisierung auf strukturierten Gittern äquivalent zu Finiten Elementen und Finiten Volumen 1. Ordnung.

3.4.3. Zeitdiskretisierung

Die Ableitung in der Zeit wird duch das explizite Euler-Verfahren diskretisiert. An jedem Punkt i des Gitters aus 3.4.2 ergibt sich eine ODE in der Zeit

$$\frac{da_i}{dt} = A(a, t)$$

mit einem nicht linearen Operator $A(\cdot, \cdot)$. Das Zeitintervall $[0, T]$ wird in Intervalle der Länge Δt unterteilt (nicht notwendigerweise äquidistant), $\frac{da_i}{dt}$ wird als rechtsseitiger Differenzenquotient approximiert. Es ergibt sich:

$$\frac{da_i(t^k)}{dt} \approx \frac{a_i(t^k + \Delta t^k) - a_i(t^k)}{\Delta t^k} \approx A(a^k, t^k)$$

$$\begin{aligned} \Rightarrow a_i^{k+1} &\approx \Delta t^k \cdot A(a^k, t^k) + a_i^k \\ &= D \frac{\Delta t^k}{h^2} \sum_{j \in \mathcal{N}_i} (a_j^k - a_i^k) + \Delta t^k f(a_i^k, b_i^k) + a_i^k \end{aligned}$$

3.4.4. CFL-Bedingung/Stabilität

Wir erinnern uns: Das explizite Euler-Verfahren ist nur für kleine Zeitschritte stabil.
$\longrightarrow$ Zeitschrittschranke, benannt nach [Courant et al., 1928]

Satz 3.3. *Gegeben sei eine FD-Semi-Diskretisierung der Wärmeleitungsgleichung*

$$\partial_t a = -D\Delta a \qquad \text{auf } \Omega \subset \mathbb{R}^n \times [0,T]$$

mit der unbekannten Funktion a, skalarem $D > 0$ und geeigneten Randbedingungen. Die Zeitdiskretisierung mit dem expliziten Euler-Verfahren ist stabil für

$$\Delta t \leq \frac{h^2}{2D}$$

Beweis. Den Beweis führen mit Hilfe der *Neumann-Analyse*.

Wir betrachten die Fehlerfortpflanzung für den Fehler $\epsilon_j^k = e^{iqx(j)}e^{iry(j)}$ im Knoten j zum Zeitpunkt $k\Delta t$.

$$\begin{aligned}
\tilde{a}_j^k &= a_j^k + \epsilon_j^k \\
\tilde{a}_j^{k+1} &= \Delta t \cdot \frac{D}{h^2} \cdot \sum_{l \in \mathcal{N}_j} (\tilde{a}_l^k - \tilde{a}_j^k) + \tilde{a}_j^k \\
\Rightarrow \epsilon_j^{k+1} &= \tilde{a}_j^{k+1} - a_j^{k+1} \\
&= \epsilon_j^k (1 - \Delta t \cdot \frac{D}{h^2} \cdot \sum_{\mathcal{N}} 1) \\
&= \epsilon_j^k (1 - \Delta t \cdot \frac{D}{h^2} \cdot \sum_{\mathcal{N}} 1) \\
&= \epsilon_j^k \underbrace{(1 - \frac{\Delta t \cdot 2 \cdot n \cdot D}{h^2})}_{\text{Verstärkungsfaktor}}
\end{aligned}$$

$$\Rightarrow \Delta t \leq \frac{h^2}{2nD}$$

Der Fehler läßt sich schärfer abschätzen, indem man die kartesischen Richtungen einzeln betrachtet. Der Fehler besteht aus Anteilen in x- und y-Richtung

und entsprechend findet auch nur die Kopplung statt. Damit summiert man nur über den linken und rechten Nachbarn und wir erhalten

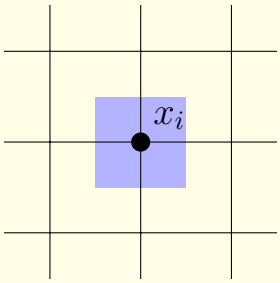
$$\Delta t \leq \frac{h^2}{2D}$$

□

Anmerkung

Für die Diffusionsgeschwindigkeit gilt $\frac{\Delta x}{\Delta t} \leq \frac{2D}{\Delta x}$, da auf einem diskreten Gitter nur Kopplungen mit den Nachbarzellen existieren

Betrachten wir die Zellen des duale Gitter, so ist der Fluss von a aus einer dualen Zelle / dem Kontrollvolumen beschränkt, sodass im Intervall Δt nicht mehr Stoff aus der Zelle austritt als zum Zeitpunkt t^k vorhanden war.



3.4.5. Splitting-Verfahren

Ziel: Umgehung der Zeitschrittschranke (für schnelle Reaktionen)

Implizite Verfahren, wie z. B. das implizite Euler-Verfahren, sind sehr teuer ($\rightarrow$ Newton-Iterationen $\rightarrow$ große GLS).

Definition 3.4: Gegeben sei ein Anfangswertproblem mit additiv zerlegter rechter Seite

$$\frac{\partial a}{\partial t} = f(a) + g(a), \quad a(0) = a_0 \tag{3.1}$$

Angenommen wir kennen die Lösungen

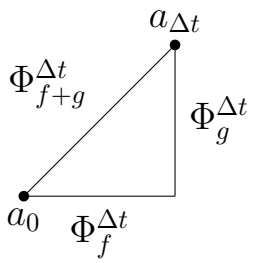
$$\frac{\partial a}{\partial t} = f(a), \quad \frac{\partial a}{\partial t} = g(a),$$

und können diese durch die Entwicklungsoperatoren

$$\Phi_f^{\Delta t}, \quad \Phi_g^{\Delta t}$$

beschreiben ($\Phi^{\Delta t}(a_0) = a_{a_0}(t + \Delta t)$). So liefert das *Lie-Trotter-Splitting* eine Näherung

$$\Phi_{f+g}^{\Delta t} \approx \Phi_g^{\Delta t} \circ \Phi_f^{\Delta t}$$



Anmerkung
Algorithmisch lässt sich der Ansatz folgendermaßen beschreiben:

Gesucht ist $a(t + \Delta t)$ als Lösung für (3.1) mit $a(t) = a_t$.

a) Berechne $\tilde{a}(t + \Delta t)$ für $\frac{\partial \tilde{a}}{\partial t} = f(\tilde{a})$ und $\tilde{a}(t) = a_t$.

b) Berechne $\hat{a}(t + \Delta t)$ für $\frac{\partial \hat{a}}{\partial t} = f(\hat{a})$ und $\hat{a}(t) = \tilde{a}(t)$.

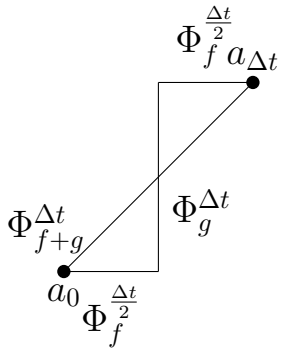
$\hat{a}(t + \Delta t) \approx a(t + \Delta t)$ liefert dann ein Näherungsergebnis.

$\longrightarrow$ Splitting-Fehler zusätzlich zum Fehler der FD-Verfahrens und des Zeitintegrationsschemas.

Alternativ definieren wir:

Definition 3.5 Strang-Splitting: Das *Strang-Splitting* wird definiert durch

$$\Phi_{f+g}^{\Delta t} \approx \Phi_f^{\frac{\Delta t}{2}} \circ \Phi_g^{\frac{\Delta t}{2}} \circ \Phi_f^{\frac{\Delta t}{2}}.$$



Satz 3.4. *Das Lie-Trotter- bzw. das Strang-Splitting hat die Konsistenzordnung 1 bzw. 2.*

Korollar 3.5. *Die Taylorentwicklung der Evolution $\Phi^{\Delta t}$ um $t = 0$ ist für $\frac{d}{dt}a = f(a)$, $a(0) = a_0$ gegeben durch*

$$\Phi^{\Delta t}a_0 = a_0 + \Delta t f(a_0) + \frac{1}{2}\Delta t^2(Df(a_0) \cdot f(a_0)) + \mathcal{O}(\Delta t^3)$$

Beweis. Es gilt

$$\Phi^{\Delta t}a_0 = a(\Delta t) = a(0) + \Delta t \cdot \frac{d}{dt}a(0) + \frac{1}{2}\Delta t^2 \cdot \frac{d^2}{dt^2}a(0) + \mathcal{O}(\Delta t^3),$$

sowie

$$\begin{aligned} a(0) &= a_0 \\ \frac{d}{dt}a(0) &= f(a_0) \\ \frac{d^2}{dt^2}a(0) &= \frac{d}{dt}f(a_0) = Df(a_0) \cdot \frac{d}{dt}a(0) = Df(a_0)f(a_0). \end{aligned}$$

Die Behauptung folgt. □

Beweis (Satz 3.4, Konvergenzorderung des Strang-Splittings).
Mit Korollar 3.5 folgt für die Taylorreihe von $\Phi^{\Delta t}_{f+g}$:

$$\Phi^{\Delta t}_{f+g}a = a + \Delta t(f(a) + g(a)) + \frac{\Delta t^2}{2}(Df(a) + Dg(a)) \cdot (f(a) + g(a)) + \mathcal{O}(\Delta t^3).$$

Für die Taylorreihe von $\Phi^{\frac{\Delta t}{2}}_f$, $\Phi^{\Delta t}_g$ gilt:

$$\begin{aligned} \Phi^{\frac{\Delta t}{2}}_f a &= a + \frac{\Delta t}{2}f(a) + \frac{\Delta t^2}{8}Df(a) \cdot f(a) + \mathcal{O}(\Delta t^3) \\ \Phi^{\Delta t}_g a &= a + \Delta t g(a) + \frac{\Delta t^2}{2}Dg(a) \cdot g(a) + \mathcal{O}(\Delta t^3). \end{aligned}$$

Für das Strang-Splitting gilt:

$$\begin{aligned}
\Phi_{f+g}^{\Delta t} &= \Phi_f^{\frac{\Delta t^2}{2}} \left(\Phi_g^{\Delta t} \left(\Phi_f^{\frac{\Delta t}{2}} a \right) \right) \\
&= \Phi_f^{\frac{\Delta t}{2}} \left(\Phi_g^{\Delta t} \left(a + \frac{\Delta t}{2} f(a) + \frac{\Delta t^2}{8} Df(a) \cdot f(a) + \mathcal{O}(\Delta t^3) \right) \right) \\
&= \Phi_f^{\frac{\Delta t}{2}} \Bigg(a + \frac{\Delta t}{2} f(a) + \frac{\Delta t^2}{8} Df(a) \cdot f(a) + \mathcal{O}(\Delta t^3)) \\
&\quad + \Delta t g(a + \frac{\Delta t}{2} f(a) + \mathcal{O}(\Delta t^2)) \\
&\quad + \frac{\Delta t^2}{2} Dg(a + \mathcal{O}(\Delta t)) \cdot g(a + \mathcal{O}(\Delta t)) + \mathcal{O}(\Delta t^3) \Bigg).
\end{aligned}$$

Taylorn der beiden farbig markierten Terme ■ und ■ liefert:

$$\begin{aligned}
&= \Phi_f^{\frac{\Delta t}{2}} \Bigg(a + \frac{\Delta t}{2} f(a) + \Delta t g(a) + \frac{\Delta t^2}{8} Df(a) \cdot f(a) \\
&\quad + \frac{\Delta t^2}{2} Dg(a) \cdot f(a) + \frac{\Delta t^2}{2} Dg(a) \cdot g(a) + \mathcal{O}(\Delta t^3) \Bigg) \\
&= a + \frac{\Delta t}{2} f(a) + \Delta t g(a) + \frac{\Delta t^2}{8} Df(a) \cdot f(a) \\
&\quad + \frac{\Delta t^2}{2} Dg(a) \cdot f(a) + \frac{\Delta t^2}{2} Dg(a) \cdot g(a) + \mathcal{O}(\Delta t^3) \\
&\quad + \frac{\Delta t}{2} f \left(a + \frac{\Delta t}{2} f(a) + \Delta t g(a) + \mathcal{O}(\Delta t^2) \right) + \frac{\Delta t^2}{8} f(a + \mathcal{O}(\Delta t)) + \mathcal{O}(\Delta t^3)
\end{aligned}$$

Taylorn der beiden nun hervorgehobenen Terme ■ und ■ führt auf

$$\begin{aligned}
&= a + \Delta t (f(a) + g(a)) \\
&\quad + \frac{\Delta t^2}{2} (Dg(a) f(a) + Dg(a) g(a) + Df(a) g(a) + Df(a) f(a)) + \mathcal{O}(\Delta t^3)
\end{aligned}$$

$\longrightarrow$ Vergleich liefert Behauptung. □

3.4.6. Physikalische Eigenschaften

Wir betrachten das vereinfachte Modell ohne Reaktionsterm (d.h. die Wärmeleitungsgleichung). So haben wir zwei zentrale physikalische Eigenschaften:

a) Massenerhaltung

b) Monotonie (Maximumsprinzip)

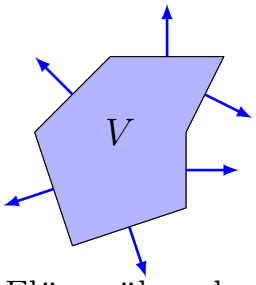
Massenerhaltung:

Unser Modell beschreibt ein abgeschlossenes System. Aus der Randbedingung $\nabla a \cdot n = 0$ folgt, dass die Gesamtenergie von Stoff a im Gebiet Ω konstant ist

$$\frac{d}{dt}\left(\int_\Omega a dx\right) = D\int_\Omega \Delta a dx = D\int_{\partial\Omega} \nabla a \cdot n dS = 0$$

Für ein beliebiges Kontrollvolumen $V \subset \Omega$ folgt die lokale Massenerhaltung

$$\frac{d}{dt}\left(\int_V a dx\right) = D\int_{\partial V} \nabla a \cdot n dS$$



d.h. die Massenänderung entspricht dem Integral der Flüsse über den Rand des Kontrollvolumens. Nehmen wir die Reaktion hinzu, folgt

$$\begin{aligned}\frac{\partial}{\partial t}\int_V a dx &= \int_V D\Delta a dx + \int_V f(a) dx \\ &= D\int_{\partial V} \nabla a \cdot n dS + \int_V f(a) dx\end{aligned}$$

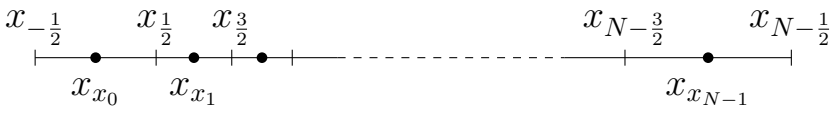
Die Reaktion wirkt als Quell-/ Senkterm im Volumen.

Anmerkung
Im Allgemeinen garantiert die FD-Diskretisierung keine lokale Massenerhaltung, da die differentielle Formulierung und nicht die Integral-Formulierung diskretisiert wird.
$\longrightarrow$ Alternativer Ansatz: Finite-Volumen-Verfahren (Integrierte Finite Differenzen)

3.4.7. Finite-Volumen-Diskretisierung der Differentialgleichung

Zell-zentrierte Finite-Volumen-Verfahren:

Wir betrachten (wieder) das 1D-Problem. Gegeben $\Omega \subset \mathbb{R}$ beschreibt $\mathcal{T}(\Omega)$ eine Partition in n Intervalle i der Länge h_i. Die Intervalle werden konsektiv durchnummeriert.



Das definiert Zellen $E_i \in [0, N-1]$ mit Zellmittelpunkten x_i. Wir betrachten das Diffusionsproblem in integrierter Form

$$\frac{\partial}{\partial t} \int_V a dx = D \int_{\partial V} \nabla a \cdot n dS$$

Annahmen:

- Die Lösung ist konstant auf jeder Zelle E_i.
- Der Gradient wird auf dem dualen Gitter der Zellmittelpunkte als FD-Approximation bestimmt, d. h.

$$\nabla a(x_{i+\frac{1}{2}}) = \nabla a(x_{(i+1)-\frac{1}{2}}) \approx \frac{a_{i+1} - a_i}{x_{i+1} - x_i} = \frac{a_{i+1} - a_i}{\frac{1}{2}(h_{i+1} + h_i)}.$$

- Als Kontrollvolumen wählen wir die Zellen E_i.

Dies führt auf

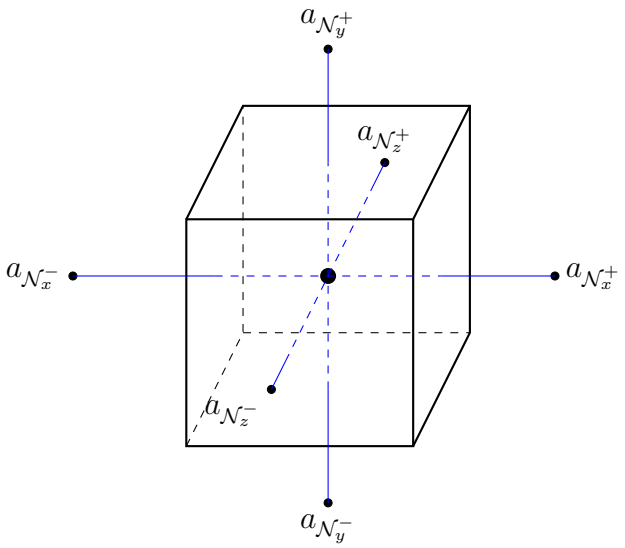
$$\frac{\partial}{\partial t} a_i = \frac{1}{h_i} D \left(- \underbrace{\frac{a_i - a_{i-1}}{\frac{1}{2}(h_i + h_{i-1})}}_{\text{linker Rand: } n=-1} + \underbrace{\frac{a_{i+1} - a_i}{\frac{1}{2}(h_{i+1} + h_i)}}_{\text{rechter Rand: } n=1} \right) \quad \forall i \in]0, N-1[,$$

sowie für die Randzellen links

$$\frac{\partial}{\partial t} a_0 = \frac{1}{h_0} D \left(0 + \frac{a_{i+1} - a_i}{\frac{1}{2}(h_1 + h_0)} \right),$$

rechts analog.

Erweiterung auf 2D/3D: Durch das Tensorprodukt konstruieren wir ein Gitter $\mathcal{T}(\Omega)$, welches auf rechteckigen Zellen E_i besteht mit den Nachbarn $\mathcal{N} = \{\mathcal{N}_x^+, \mathcal{N}_x^-, \mathcal{N}_y^+, \mathcal{N}_y^-, \mathcal{N}_z^+, \mathcal{N}_z^-\}$.



$$\begin{aligned}\partial_t a_i &= \frac{1}{h_{ix}h_{iy}h_{iz}} \cdot D \sum_{d\in\{x,y,z\}} \sum_{s\in\{+,-\}} n_{\mathcal{N}_{ds}} \frac{h_{ix}h_{iy}h_{iz}}{h_{id}} \frac{a_{\mathcal{N}_{ds}} - a_i}{x_{\mathcal{N}_{ds}} - x_i} \\ &= D \sum_{d\in\{x,y,z\}} -\frac{a_i - a_{\mathcal{N}_{d-}}}{\frac{1}{2}(h_{\mathcal{N}_{d-}} + h_{id})h_{id}} + \frac{a_{\mathcal{N}_{d+}} - a_i}{\frac{1}{2}(h_{\mathcal{N}_{d+}} + h_{id})h_{id}}\end{aligned}$$

Für äquidistante Gitter gilt

$$\partial_t a_i = \frac{1}{h^2} d \sum_{j\in\mathcal{N}} (a_j - a_i)$$

→ das Verfahren ist identisch zu FD, bis auf Randterme.

3.5. Finite Elemente Verfahren

In 1D lassen sich FD, FV, FE algebraisch äquivalent formulieren. FE-Verfahren bieten einen allgemeineren analytischen Zugang.

3.5.1. Variationsproblem

Der Einfachheit halber betrachten wir das elliptische Problem: Gegeben $f \in L^2(\Omega)$ suchen wir $u \in H^1(\Omega)$, sodass

$$-\Delta u = f \qquad \text{in } \Omega \tag{3.2}$$
$$\nabla u \cdot n = 0 \qquad \text{auf } \partial\Omega, \tag{3.3}$$

bzw. im diskreten Fall:
Wir suchen $u_h \in V_h \subset H^1(\Omega)$, sodass $u_h \approx u$ im Unterraum. Es ergibt sich eine Folge $u_h \to u$ für die Folge $V_h \to H^1(\Omega)$. Die schwache Formulierung von (3.2) lautet

$$\int_\Omega (-\Delta u) v dx = \int_\Omega f v dx \quad \forall v \in H^1(\Omega).$$

Partielle Integration liefert

$$\int_\Omega \nabla u \nabla v dx = \int_\Omega f v dx + \int_{\partial\Omega} \underbrace{(n \cdot \nabla u)}_{=0} v dS.$$

Das Problem in schwacher Formulierung lautet nun:
Gegeben $f \in L^2(\Omega)$ suchen wir $u \in H^1(\Omega)$, sodass gilt:

$$\int_\Omega \nabla u \nabla v dx = \int_\Omega f v dx \quad \forall v \in H^1(\Omega) \tag{3.4}$$

oder $a(u, v) = (f, v)_{L^2}$ mit der Bilinearform $a : H^1 \times H^1 \to \mathbb{R}$. Zur Lösung wählen wir einen diskreten Unterraum (endlich-dimensional) U_h, V_h mit Folgen $U_h \to H^1$, $V_h \to H^1$ um Näherungslösungen $u_h \to u$ zu bestimmen. (3.4) ergibt damit das diskrete Variationsproblem:
Gegeben $f \in L^2(\Omega)$, suchen wir $u_h \in U_h(\Omega)$, sodass gilt

$$a_h(u_h, v_h) = (f, v)_{L^2} \quad \forall v_h \in V_h(\Omega)$$

mit der diskreten Bilinearform $a_h : U_h \times V_h \to \mathbb{R}$ und dem Ansatzraum U_h, sowie dem Testraum V_h.

Anmerkung
Die Wahl der Räume U_h, V_h führt auf eine große Klasse unterschiedlicher Verfahren, z. B.

- konforme FE-Verfahren
- spektrale FE-Verfahren
- FV-Verfahren
- Discontinuous Galerkin-Verfahren

Anmerkung
Für einen symmetrischen Operator $a_h : U_h \times V_h \to \mathbb{R}$ lässt sich ein zu (3.4) äquivalentes Minimierungsproblem formulieren:
Das Energiefunktional $J(v) := \frac{1}{2}a(v,v) - (f,v)_{L^2}$ nimmt in H^1 sein Minimum bei u an, genau dann, wenn

$$a(u,v) = (f,v)_{L^2} \quad \forall v \in H^1(\Omega).$$

Um dünnbesetzte Gleichungssysteme (ähnlich zu FD, FV) zu erhalten, versuchen wir U_h, V_h so zu konstruieren, dass wir eine Basis finden, deren Basisfunktionen fast überall Null sind. In der Regel basiert die Konstruktion auf einer Triangulierung $\mathcal{T}(\Omega)$ und die Basisfunktionen sind nur auf wenigen Zellen ungleich Null.

3.5.2. Galerkin-Verfahren

Wählen wir $V_h = U_h$, so sprechen wir von Galerkin-Verfahren. Gesucht ist $u_h \in V_h$, sodass

$$a_h(u_h, v_h) = (f, v_h)_{L^2} \quad \forall v_h \in V_h \tag{3.5}$$

Weiter sei $\{\varphi_1, \varphi_2, ..., \varphi_N\}$ eine Basis von V_h, dann ist (3.5) äquivalent zu

$$a_h(u_h, \varphi_i) = (f, \varphi_i)_{L^2} \quad \forall i \in [1, N]$$

Wir setzen $u_h = \sum_{k=1}^{N} x_k \varphi_k$. Dann erhalten wir ein lineares Gleichungssystem

$$\sum_{k=1}^{N} a(\varphi_k, \varphi_i) x_k = (f, \varphi_i)_{L^2}, \quad i = 1, ..., N,$$

welches wir in Matrix-Vektor-Schreibweise als

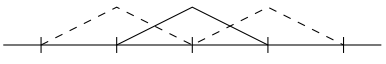
$$Ax = b \quad \text{mit } A_{ik} = a(\varphi_k, \varphi_i)$$
$$b_i = (f, \varphi_i)_{L^2}$$

schreiben können. Der "klassische" Ansatz sind stückweise polygoniale Funktionen.

Beispiel 3.1 Lagrange-Ansätze 1. Ordnung: Sei $\mathcal{T}(\Omega)$ eine geeignete Partition von Ω, so definieren wir Ansatzräume für stückweise lineare Funktionen

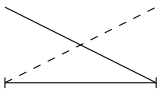
$$V_h = \{v_h \in C(\bar{\Omega})|\ v_h|_E \in P_1(E), E \in \mathcal{T}(\Omega)\},$$

wobei P_k der Raum der Polynome bis Grad k ist. Eine kanonische Wahl der Basis ist die sogenannte "Knotenbasis" in Analogie zur Lagrangebasis der Lagrange-Interpolation. Dabei wird jedem Knoten a_i eine Basisfunktion φ_i zugeordnet, für die $\varphi_i(a_j) = \delta_{ij}$, $\varphi_i \in V_h$ gilt.



E

1D: Auf jedem Element lassen sich diese zusammensetzen aus



$$\tilde{\varphi}_0 = 1 - x \qquad \nabla\tilde{\varphi}_0 = -1$$
$$\tilde{\varphi}_1 = x \qquad \nabla\tilde{\varphi}_1 = 1.$$

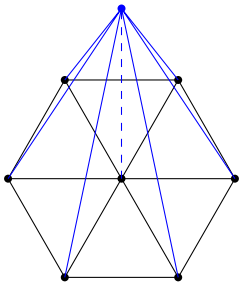
Es gibt dabei keine Kopplungen über die Elmenentgrenzen hinaus:

$$a_h(\varphi_k, \varphi_i) = \begin{cases} 0 & \text{für } i \notin \{k-1, k, k+1\} \\ -\int_{h(i-1)}^{h(i+1)} \left(\frac{1}{h}\right)^2 dx = -\frac{2}{h} & \text{für } i = k \\ -\int_{hi}^{hk} -\frac{1}{h}\frac{1}{h} dx = \frac{1}{h} & \text{für } i = k-1 \\ -\int_{hk}^{hi} \frac{1}{h}\left(-\frac{1}{h}\right) dx = \frac{1}{h} & \text{für } i = k+1, \end{cases}$$

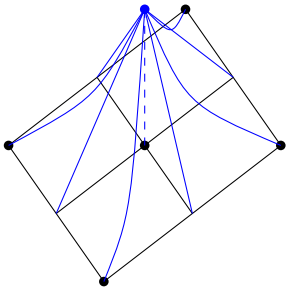
d. h. auf äquidistanten Gittern erhalten wir die gleiche Diskretisierung wie für die FD-Methode.

2D: Es werden Lagrangeansätze in höheren Raumdimensionen verwendet mit Basisfunktionen analog basierend auf der Basis der Lagrangeinterpolation.

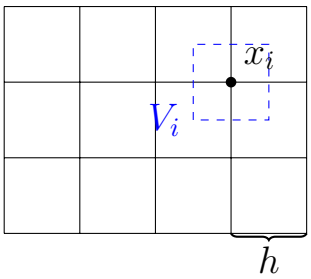
Dreieck:
lineare "P1"-Funktion



Viereck:
bilineare "Q1"-Funktion



3.5.3. Finite Volumen als Finite Elemente Verfahren



Gegeben $\Omega \subset \mathbb{R}^n$ konstruieren wir ein strukturiertes Gitter $\mathcal{T}(\Omega)$ und das zugehörige duale Gitter als Voronoi-Gitter $\mathcal{V}(\mathcal{T}(\Omega))$. Zu jedem Knoten $x_i \in \mathcal{T}(\Omega)$ assoziieren wir eine Voronoi-Zelle $V_i = \{x|\ |x-x_i| < |x-x_j|$ für alle Knoten $x_j\}$. Als Ansatzfunktion U_h wählen wir stückweise lineare Funktionen

$$U_h = \{u \in \mathbb{C}^0|\ u|_E \in Q_1, E \in \mathcal{T}(\Omega)\}$$

Als Testfunktionen wählen wir stückweise konstante Funktionen auf dem dualen Gitter $\mathcal{V}$

$$V_h = \{v \in L^2|\ v|_{V_i} \text{ const}, v_i \in \mathcal{V}(\Omega)\}.$$

Als Basis wählen wir

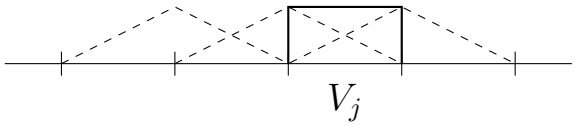
$$U_h = \text{span}\{\varphi_i\}, \varphi_i \text{ Knotenfunktion zum Knoten } x_i \text{ mit } \varphi_i(x_j)\delta_{ij}$$
$$V_h = \text{span}\{\psi_i\}, \psi_i = \begin{cases} 1 \text{ auf } V_i \\ 0 \text{ sonst.} \end{cases}$$

Wir erhalten folgende Bilinearform:

$$a(\varphi_i, \psi_j) = -\int_\Omega (\Delta\varphi_i)\psi_j dx = -\int_{V_j} \Delta\varphi_i dx = -\int_{\partial V_j} u\nabla\varphi_i dS.$$

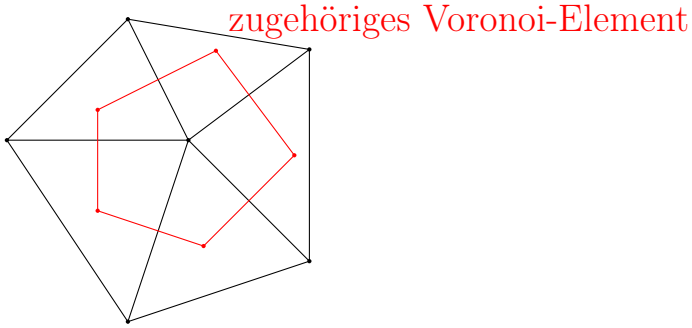
Erinnern wir uns an das FV-Verfahren: Konstruktion über die integrierte Form

1D:



- $\nabla\varphi_i$ entspricht der FD-Konstruktion
- RHS: $(f, \psi_j)_{L^2} = \int f\psi_j dx = \int_{V_j} f dx$
- Auswertung über Mittelpunktsregel liefert $|V_j| \cdot f(x_j)$

2D:
allgemeiner: Delaunay-Triangulierung $\longrightarrow$ knotenkonzentrierte FV



Anmerkung
Ist $\mathcal{T}(\Omega)$ ein strukturiertes Gitter, so ist $\mathcal{V}(\mathcal{T})$ ein strukturiertes Gitter und damit ein gültiges FE-Gitter. Wir konstruieren U_h bzgl. $\mathcal{V}(\mathcal{T})$, V_h bzgl. $\mathcal{T}$, Integration über die Mittelpunktsregel führt dann auf das bekannte zellzentrierte FV-Verfahren.

3.5.4. Massenerhaltung im Finite Elemente-Kontext

lokale Massenerhaltung:

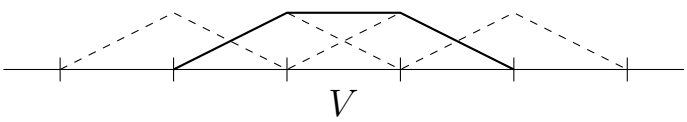
Mit der Flussbilanz

$$-\int_{\partial V} n \cdot \nabla u dx = -\int_{\partial V} n \cdot j dx = \int_V q dx$$

erhält man durch Testen mit $v \equiv 1$ auf V, 0 sonst, sowie partieller Integration

$$\begin{aligned} a(u,v) = -\int_V \Delta u v dx &= \int_V \nabla u \underbrace{\nabla v}_{=0} dx - \int_{\partial V} n \cdot \nabla u \underbrace{v}_{=1} dS \\ &= -\int_{\partial V} n \cdot \nabla u dS \\ (f,v)_{L^2} = \int_V f v dx &= \int_V f dx. \end{aligned}$$

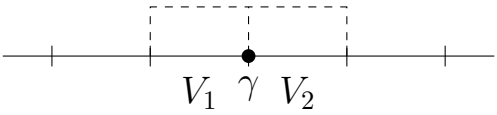
Lokale Massenerhaltung wird gewährleistet, wenn die 1 in unserem Testraum ist (d. h. eine Testfunktion, die konstant auf dem Kontrollvolumen ist und kompakten Träger hat).



globale Massenerhaltung(1):

Aus der lokalen Massenerhaltung folgt globale Massenerhaltung genau dann, wenn die Flussberechnung von links und rechts an ein Face gleich ist, d. h.

$$\nabla u(\gamma)|_{V_1} = \nabla u(\gamma)|_{V_2}$$



$\longrightarrow V = V_1 + V_2$ Flussbilanz folgt als Summe (Übung)

globale Massenerhaltung(2):

Wir wählen als Kontrollvolumen $V = \Omega$ und erhalten nach Konstruktion die Kontinuitätsgleichung, falls die Funktion $v \equiv 1$ auf Ω im Testraum V_h ist.

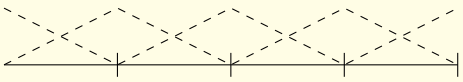
Anmerkung

Die Basisfunktionen auf den Randelementen unterscheiden sich abhängig von der Randbedingung:

a) Neumann-Randbedingungen :

$$-\Delta u = f \text{ auf } \Omega$$
$$n \cdot \nabla u = j \text{ auf } \partial\Omega$$

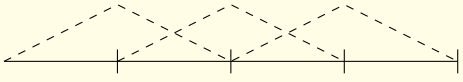
$\longrightarrow$ auch mit den Knoten am Rand assoziieren wir Basisfunktionen $V_h = U_h \subset H^1(\Omega)$



b) Dirichlet-Randbedingungen:

$$-\Delta u = f \text{ auf } \Omega$$
$$u = g \text{ auf } \partial\Omega$$

$\longrightarrow$ keine Basisfunktionen für Randknoten $V_h = U_h \subset H_0^1(\Omega)$



$\longrightarrow$ Für Neumann-Randbedingungen erhalten wir globale Massenerhaltung, nicht jedoch für Dirichlet-Randbedingungen.

3.5.5. Diskretes Maximumsprinzip für FE-Verfahren

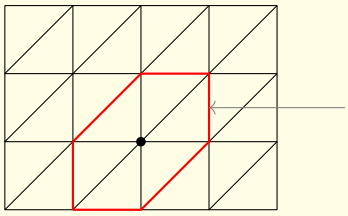
Das diskrete Maximumsprinzip ist erfüllt, wenn die Steifigkeitsmatrix A eine M-Matrix ist:

$$\begin{aligned} A_{ii} &> 0 \\ A_{ij} &\leq 0 \quad \text{für} j \neq i \\ A_{ii} &\geq \sum_{j \neq i} |A_{ij}| \end{aligned}$$

Satz 3.6 (Maximumsprinzip für Finite Elemente)**.** *Wenn alle Innenwinkel des Gitters $\mathcal{T}(\Omega)$ kleiner oder gleich $\frac{\pi}{2}$ sind, genügt das FE-Verfahren 1. Ornund (Lagrange-FE) einem diskreten Maximumsprinzip, d. h. die Steifigkeitsmatrix A ist eine M-Matrix.*

Beweis. Idee: Man betrachtet eine einzelne Zelle aus $\mathcal{T}(\Omega)$ und berechnet die lokalen Einträge der Bilinearform. □

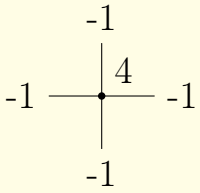
Anmerkung



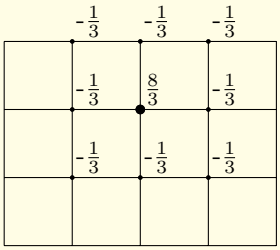
Im Spezialfall eines strukturierten kartesischen Dreiecks-Gitters erhalten wir folgende explizite Darstellung der Matrix A:

$$a_{ij} = a(\nabla\varphi_i, \nabla\varphi_j),$$

mit $a_{ii} = 4, a_{i,i\pm1} = a_{i,i\pm N} = -1$, alle anderen Elemente sind 0.



Das entspricht genau den Einträgen der FD-Diskretisierung bis auf einen Faktor h^2
$\longrightarrow A^{FE} = h^2 A^{FD}$.
Die Viereckskonstruktion liefert einen "9-Punkte-Stern".



3.5.6. Praktische Askekte

Gegeben sei $\Omega \subset \mathbb{R}^n$ und eine Triangulierung $\mathcal{T}(\Omega)$. Wir betrachten das elliptische Problem

$$\begin{aligned} Lu = -\Delta u = f \quad & \text{auf } \Omega \\ \nabla u \cdot n = j \quad & \text{auf } \partial\Omega \\ (u = 0 / u = g \quad & \text{auf } \partial\Omega) \end{aligned}$$

mit den Test- und Ansatzräumen U_h, V_h. Die schwache Formulierung/Diskretisierung führt auf

$$a(u, \varphi) = \ell(\varphi) \quad \forall \varphi \in V_h$$

mit der Bilinearform

$$a(u, \varphi) = \int_\Omega \nabla u_h \nabla \varphi dx$$

und der Linearform

$$\ell(\varphi) = \int_\Omega f\varphi dx - \int_{\partial\Omega} j\varphi dS.$$

Dies führt auf LGS, welches sich in der Matrix-Vektor-Schreibweise darstellt als

$$Ax = b,$$

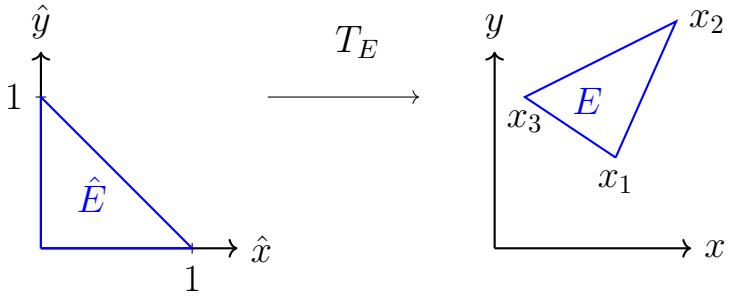
wobei x der Koeffizientenvektor der Lösung $u = \sum_i x_i \varphi_i$ ist.

Wie geht man praktisch vor?

Für beliebige Elemente ist eine diskrete Darstellung der Basis i. d. R. schwer anzugeben, stattdessen verwenden wir lokale Darstellungen.

1. Darstellung der Elemente $E \in \mathcal{T}(\Omega)$ als Referenzelement $\hat{E}$ und Transformation T_E

   Beispiel: Dreiecksgitter & Referenzelement mit den Eckpunkten (0,0), (1,0), (0,1)



2. Auf jedem Element betrachten wir jene Basen, welche Träger auf dem Element haben. Dies liefert die lokale Basis

$$\{\phi_i\} = \{\varphi_j | \operatorname{supp}(\varphi_j) \cap E \neq \emptyset\}.$$

   Diese lokale Basis korrespondiert zu einer lokalen Basis auf dem Referenzelement. Üblicherweise wird die Basis auf dem Referenzelement definiert und die globale Basis entsteht durch tranformation und zusammensetzen dieser lokalen Basen $\{\hat{\phi}_i\}$.

   Beispiel: lineare Ansätze 1. Ordnung:

$$\begin{aligned} \hat{\phi}_0 &= 1 - \hat{x} - \hat{y} \\ \hat{\phi}_1 &= \hat{x} \\ \hat{\phi}_2 &= \hat{y} \end{aligned}$$

3. FE-Ansätze auf $E \in \mathcal{T}(\Omega)$ ergeben sich durch Transformation der lokalen Basis mit $T_E : \hat{E} \to E \subset \Omega$, sodass für $x \in E$ gilt:

$$\varphi_i(x)|_E = \hat{\phi}_i \left(T_E^{-1}(x)\right)$$

Anmerkung
Solche Ansätze nennt man parametrisch. Falls T_E aus dem gleichen Polynomraum wie die lokale Basis $\{\hat{\phi}_i\}$ kommt, spricht man von iso-parametrischen Ansätzen.

TODO: $\hat{\phi}$ vs. φ, x vs. $\hat{x}$ (eventuell durcheinander)

4. Für iso-parametrische Dreieckselemente verwenden wir eine affin-lineare Transformation T_E, diese lässt sich mit Hilfe der Basis $\{\hat{\phi}_i\}$ schreiben als

$$T_E(\hat{x}) = x_1\hat{\phi}_1(\hat{x}) + x_2\hat{\phi}_2(\hat{x}) + x_3\hat{\phi}_3(\hat{x}),$$

mit den Ecken des Dreiecks x_i. Alternativ gilt:

$$T_E(\hat{x}) = \underbrace{B_E}_{\text{Verzerrung}}\hat{x} + \underbrace{b_E}_{\text{Verschiebung}},$$

wobei B_E der Jacobimatrix J_{T_E} der Transformation T_E entspricht.

Das Aufstellen des Gleichungssystems, d. h. der Matrix $A_{ij} = a(\varphi_i, \varphi_j)$ und der rechten Seite $b_i = (f, \varphi_i)_{L^2}$ erfordert Integration der Testfunktionen über Ω. Die Basis φ_i (bzw. ψ_i) ist nur auf wenigen Elementen von 0 verschieden, d. h.

$$\text{supp}(\varphi_i) \cap E \neq \emptyset$$

Darstellung: Es gilt:

$$b_i = \int_\Omega f\varphi_i dx = \sum_{E\subset\mathcal{T}(\Omega)} \int_E f\varphi_i dx = \sum_{E\in\text{supp}(\varphi_i)} \int_E f\varphi_i dx,$$

das Ausnutzen der lokalen Darstellung liefert

$$b_i = \sum_{E\in\mathcal{T}(\Omega)} \sum_{k=1}^{|\{\phi\}|} P_{E,i} \cdot b_{E,j}(k)$$

mit der Prolongationsmatrix $P_{i,j}$ und den lokalen Beiträgen

$$b_{E,j} = (f, \phi_j)_{L^2,E} = \int_E f(x)\phi_j(x)dx.$$

Die Prolongationsmatrix P_E beschreibt eine Index-Abbildung zwischen der lokalen Basis $\{\phi\}$ und der globalen Basis $\{\varphi\}$, d.h. die j-te lokale Basis entspricht der i-ten globalen Basis. Die Matrix P_E ist eine Rechteckmatrix, welche

einen Nicht-Null Eintrag pro Spalte hat, in Zeile j, mit Wert 1 (Details folgen später).

Den lokalen Beitrag kann man auf dem Referenzelement berechnen als

$$b_{E,j} = \int_{\hat{E}} \det(J_{T_E}(\hat{x}) f(T_E(\hat{x}))\hat{\phi}_j(\hat{x}) d\hat{x},$$

wobei für den affin linearen Fall $\det(J_{T_E}) = \det(B_E)$ gilt, was der Fläche von E entspricht.

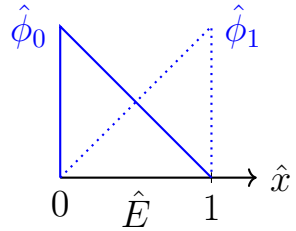
Analog konstruieren wir für die Matrix Darst

$$A_{ij} = \sum_{E \in \mathcal{T}(\Omega)} \sum_{k=1}^{|\{\hat{\phi}\}|} \sum_{l=1}^{|\{\hat{\phi}\}|} A_{E,i(k),j(l)}$$

mit

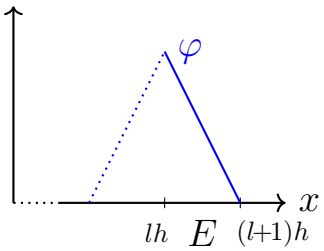
$$\begin{aligned} A_{E,i,j} &:= \int_E \nabla\varphi_j(x)\nabla\varphi_i(x)dx \\ &= \int_{\hat{E}} \det(J_{T_E})(\hat{x}) \cdot J_{T_E}^{-T} \hat{\nabla}\hat{\phi}_i(\hat{x}) d\hat{x} \end{aligned}$$

Beispiel: Transormationsgradient (1D)



$$\begin{aligned} \hat{\phi}_0 = 1 - \hat{x}, \quad \nabla\hat{\phi}_0 &= -1 \\ \hat{\phi}_1 = \hat{x}, \quad \nabla\hat{\phi}_1 &= 1 \end{aligned}$$

$$\begin{aligned} T_E(\hat{x}) &= h \cdot \hat{x} + l \cdot h, \\ T_E^{-1}(x) &= \frac{x - l \cdot h}{h} = \frac{x}{h} - l \end{aligned}$$



$$\varphi|_E = 1 + l - \frac{x}{h}, \quad \nabla\varphi|_E = -\frac{1}{h} = \left(\frac{1}{h}\right)(-1)$$

Die numerische Auswertung der lokalen Integrale erfolgt mithilfe einer Quadraturformel.

Beispiel 3.2 Quadraturen basierend auf Lagrange-Interpolation: Auf dem Referenzelement $\hat{E}$ sei ein Polynomraum $P(\hat{E})$ mit Stützstellen $\hat{x}_s \in \hat{E}$, $s = 1, ..., S$ definiert.
(Stützstellen müssen nicht Knotenpunkte der FE-Basis sein!)
Seien $L_s \in P(\hat{E})$ die entsprechenden Lagrangepolynome mit $L_s(\hat{x}_r) = \delta_{sr}$. Gegeben $\hat{v}(\hat{x})$ eine stetige Funktion, ergibt sich das Interpolationspolynom

$$p(\hat{x}) = \sum_{s=1}^{S} \hat{v}(\hat{x}_s) L_s(\hat{x}).$$

Dies führt zur Quadratur

$$Q_{\hat{E}}(\hat{v}) = \sum_{s=1}^{S} w_s \hat{v}(\hat{x}_s)$$
$$w_s = \int_E L_s(\hat{x}) d\hat{x}$$

Die Konstruktion über Stützstellen einer Interpolation erlaubt es somit, Polynome bis zum Grad der Interpolation exakt zu integrieren.

Definition 3.6 Quadraturformel : Eine interpolatorische Quadraturformel (wie oben) auf dem Referenzelement $\hat{E}$ heißt von Ordnung r, wenn Polynome bis Grad $r - 1$ (und nicht höher) integriert werden. Sie ist "zulässig", wenn ihre Stützstellen ausreichen, sodass

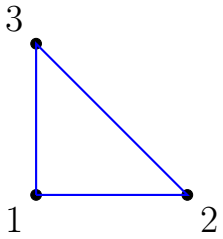
$$q \in P(\hat{E}): \quad \nabla q(x_s) = 0 \text{ für } s = 1, ..., S \quad \Rightarrow \quad q \equiv \text{const}$$

Speichern in globalen Matrizen & Vektoren:

Im Referenzelement werden die Entitäten (Ecken, Kanten,...) nummeriert und Basisfunktionen mit diesen assoziiert.

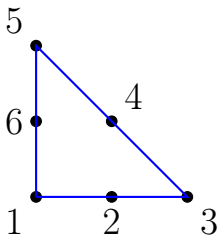
Beispiel:

Knotenbasis:



$\hat{\phi}_1, ..., \hat{\phi}_3$

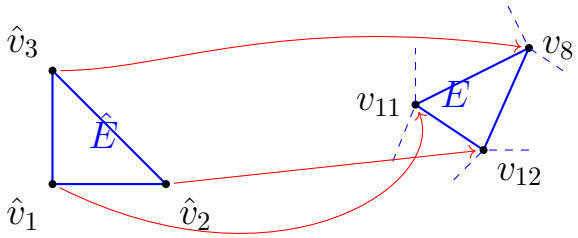
Lagrange 2. Ordnung:



$\hat{\phi}_1, ..., \hat{\phi}_6$

Beispiel:

Bezüglich eines Elementes $E \subset \mathcal{T}(\Omega)$ werden Gitterentitäten zugeordnet.



Prolongation

Beispiel:

Auf jeder Zelle werden lokale Matrizen $A_{E,i,j}$ und Vektoren $b_{E,j}$ berechnet und mithilfe der Zuordnung der Entitäten in globale Matrizen und Vektoren eingepflegt (aufaddiert).

$$\begin{pmatrix} \hat{a}_{11} & \hat{a}_{12} & \hat{a}_{13} \\ \hat{a}_{21} & \hat{a}_{22} & \hat{a}_{23} \\ \hat{a}_{31} & \hat{a}_{32} & \hat{a}_{33} \end{pmatrix} \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \end{pmatrix} \xrightarrow[\oplus]{} \begin{pmatrix} \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ b_8 \\ \vdots \\ b_{11} \\ b_{12} \\ \vdots \end{pmatrix}$$

Handhabung von Randbedingungen:

a) Neumann-Randbedingung ($\nabla u \cdot n = j$ auf $\partial\Omega$):
Diese Bedingung liefert einen Beitrag zur rechten Seite:

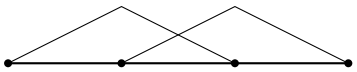
$$-\int_{\partial\Omega} j\varphi dS.$$

Das Zerlegen des Randes in Teile $\Gamma = \{\Gamma_E = \partial\Omega \cap \partial E, E \in \mathcal{T}(\Omega)\}$ erlaubt die lokale Berechnung der Beiträge pro Segment Γ_E, mit der lokalen Basisfunktion auf E.

b) Dirichlet-Randbedingungen ($u = 0$ auf $\partial\Omega$):
Diese Randbedingungen werden oft direkt in den Ansatzraum eingebaut, d. h. alle Basisfunktionen erfüllen $u = 0$ auf $\partial\Omega$.

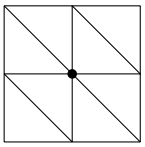
Beispiel: Knotenbasis

Es gibt keine Unbekannten auf dem Rand.



Das lokale Assemblieren (lokale Integration A_E, b_E) erfolgt ohne Berücksichtigung der Dirichlet-Randbedingung, mache Einträge werden nicht in die globale Matrix bzw. den globalen Vektor aufaddiert.

Extrembeispiel:



Matrix Strukturen, siehe Folien

Dichtbesetzte Matrizen

$A \in \mathbb{R}^{N \times N}$

- Alle Einträge werden konsekutiv im gespeichert.

- Es wird (in unseren Fällen) zeilenweise nummeriert
- Speicherbedarf N^2
- Nicht optimal für dünnbesetzte Matrizen

Beispiel:

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

0,0	0,1	0,2	0,3	1,0	1,1	1,2	1,3	2,0	···	3,2	3,3

COO-Format

Koordinatenformat

$A \in \mathbb{R}^{N \times N}$, nnz : Anzahl Einträge $\neq 0$

- einfachstes Speicherformat für dünnbesetzte Matrizen.
- 3 eindimensionale Arrays der Länge $nnz(A)$:
  - double-Array `value`: Nichtnullelemente von A (bel. Reihenfolge)
  - integer-Arrays `row-index` und `col-index`: Koordinaten der Nichtnullelemente
- Speicherbedarf $3 \times nzz$

Beispiel: $A = \begin{pmatrix} 1 & 3 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ 0 & 4 & 8 & 0 \\ 0 & 0 & 9 & 1 \end{pmatrix}$

value	1	-1	3	2	4	8	9	1
row-index	0	1	0	1	2	2	3	3
col-index	0	0	1	1	1	2	2	3

CSR-Format (oder auch CRS)

Compressed Row Storage

- Kompakt, ohne eine spezielle Struktur vorauszusetzen
- 3 eindimensionale Arrays: 2 der Länge $nnz(A)$ und 1 der Länge $N + 1$:
  - double-Array `value`: Nichtnullelemente von A (zeilenweise, von links nach rechts)

  - integer-Array `col-index`: Spaltenindex der Nichtnullelemente
  - integer-Array `row-ptr`: Beginn der jeweiligen Matrixzeile im `value` Array
- Speicheraufwand $2 \times nnz + N$

Beispiel: $A = \begin{pmatrix} 1 & 3 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ 0 & 4 & 8 & 0 \\ 0 & 0 & 9 & 1 \end{pmatrix}$

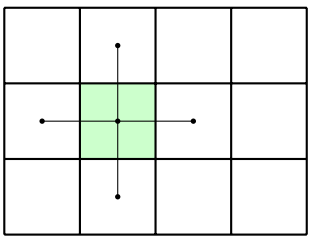
value	1	3	-1	2	4	8	9	1
col-index	0	1	0	1	1	2	2	3

row-ptr	0	2	4	6

3.5.7. Zusammenfassung

Simulation des Turingproblems:

- Turingpattern sind eine Klasse von Modellen, welche Musterbildung beschreiben
- Turingpattern basieren auf gekoppelten Diffusions-Reaktionsgleichungen
- Entkopplung von Orts- und Zeitoperator führt auf
  - GDGL in der Zeit
  - PDGL im Ort
- Operator "im Ort" enthält Reaktionssystem (GDGL, nicht linear) und mehrere Diffusionsoperatoren (PDGL, linear)
- Zeitskalen können sehr verschieden sein
  → Ansatz: Splitting-Verfahren, z. B. Strang-Splitting (2. Ordnung)
- Zeitdiskretisierung auf adaptivem Zeitgitter, die Zeitschrittweite wird bestimmt durch die CFL-Bedingung
- Splitting-Verfahren erlaubt unterschiedliche Verfahren für Reaktion und Diffusion, z. B. das explizite oder implizite Eulerverfahren
- Ortsdiskretisierung des Diffusionsoperators: allgemeiner Zugang über Finite Elemente-Verfahren → FV-, (FD-) Verfahren als Spezialfall
- wichtige (physikalische) Eigenschaften
  - "Massenerhaltung", Kontinuitätsgleichung
  - Maximumsprinzip bzw. Nicht-Linearität
- unsere Wahl: zellzentrierte Finite Volumen-Verfahren



  - 1. Ordnung exakt, 2. Ordnung Konvergenz

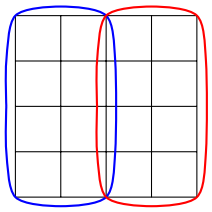
- lokal massenerhaltend
- diskretes Maximumsprinzip
- Splitting-Verfahren
- implizites Eulerverfahren für die Reaktion
- explizites Eulerverfahren für die Diffusion

→Diffusion: linear, räumlich
Reaktion: lokal, nicht linear

Ausblick

Parallelisierung mit getrenntem Speicher:

- parallele Datenaufteilung: eindeitige Zuordnung aller Freiheitsgrade (d. h. Gitterzellen) zu den Prozessoren, z. B: gegeben ein Gitter Ω mit $N \times M$ Zellen, sowie ein Feld von Prozessoren $P \times Q$, erhält jeder Prozessor $\frac{N}{P} \times \frac{M}{Q}$ Zellen



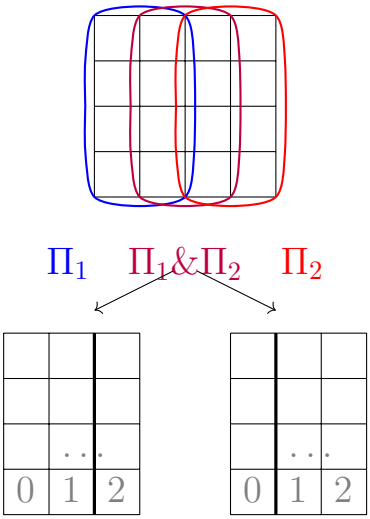
Π_1 Π_2

- eindeutige Aufteilung der Berechnung der Updates pro Zelle

Algorithmus 3.1: Turin Patter Hauptalgorithmus (sequentiell)

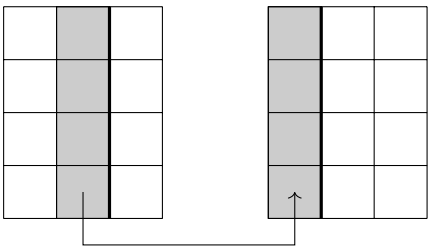
```
Grid grid
Vec x, xold(grid.size)  ⟵   andere Vektoren im Parallelen
Vec update(grid.size)   ⟵   andere Vektoren im Parallelen
double T, dt, w;
double t, told;
while(t<T) {
  w = 0.9, dt = 0;
  xold = x, told = t;
  // try timestep
  while( t == told) {
    (update,dt) = computeDiffusion(x);
                                ⟵   communicate update
        double sugg_dt = w * dt
        x += sugg_dt * update
        update = computeReaction(x,sugg_dt * 2)
                                ⟵   communicate update
        x += udpate
        (update, dt) = computeDiffusion(x)
                                ⟵   communicate update
        if(dt >= sugg_dt) {
          x += sugg_dt * update
          t += 2 * sugg_dt
        }
        else {
          x = xold
          w = w/2
        }
  }
}
```

- zusätzlich einen Layer “Ghost”-Zellen, d. h. Kopien der Daten des Nachbarprozesses

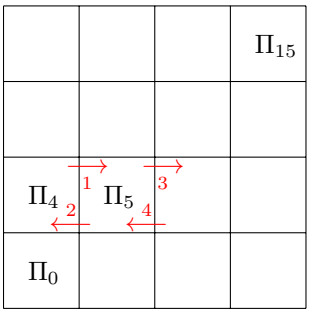


Änderungen am Gitter:

1. Berechnen der globalen Position aus Prozessornummern
   - $\longrightarrow$ virtuelle Nummerierung Touis $\longrightarrow$ Abb. $\Pi \longrightarrow (p, q)$
   - innere Zellen $\frac{N}{P} \times \frac{M}{Q}$,
   - sowie Kopien $\left[p \cdot \frac{N}{P} + 2\right] \times \left[q \cdot \frac{N}{Q} + 2\right]$ für alle innere Prozessoren
2. "for each cell"
   - iteriert über innere Zellen
3. communicate
   - mit welchem Prozessor werden die Daten getauscht?
     $\longrightarrow$ Nachbarn wie im Gitter



   - lokale Kenntnis über Überlappbereich
     $\Pi_0 : \texttt{send}(\texttt{recv}[\texttt{x}_1, \texttt{x}_4, \texttt{x}_7]) \rightarrow \Pi_2\texttt{recv}[\texttt{x}_1, \texttt{x}_3, \texttt{x}_6])$
   - wechselseitige Blockade verhindern (*dead-lock*)




```
for each dir{
   if even coord[dir]{
      send(right(dir))
      recv(right(dir))
      send(left(dir))
      recv(left(dir))
   }
   else{
      recv(left(dir))
      send(left(dir))
      recv(right(dir))
      send(right(dir))
   }
}
```

4. Strömungsmechanik

Ausgangspunkt für die Modellierung sind folgende Erhaltungseigenschaften in einem Massevolumen als abgeschlossenes System:

- Massenerhaltung: Masse wird nicht erzeugt oder vernichtet
- Impulserhaltung (2. Newtonsches Gesetz): Die Änderung des (Dreh-)Impulses ist gleich der einwirkenden Kraft (Dreh-Moment).
- Energieerhaltung: Energie wird nicht erzeugt oder vernichtet.

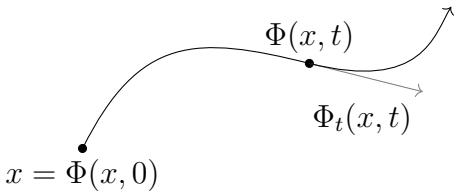
4.1. Erhaltungsgleichungen

4.1.1. Eulersche Beschreibung der Bewegung

Die Trajektorien von idealisierten Flüssigkeitspunkten werden durch eine Flussfunktion beschrieben:

$$\Phi : \quad \begin{cases} \Omega \times [0, \infty) & \to \quad \Omega \\ (x, t) & \to \quad \Phi(x, t) \end{cases}$$

mit $\Phi(x, 0) = 0$. $\Phi(x, t)$ ist die Raumkoordinate zur Zeit t des Teilchens, welches zur Zeit $t = 0$ am Punkt x war.



Die Geschwindigkeit u ergibt sich als zeitliche Ableitung des Ortes des Teilchens als

$$\Phi_t(x,t) = u(\Phi(x,t),t).$$

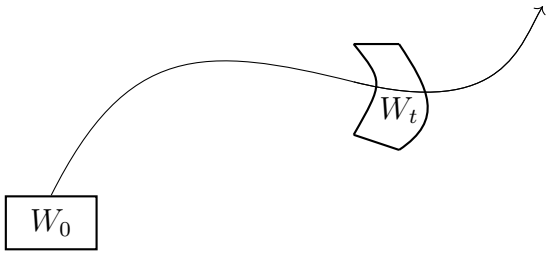
Ist das Geschwindigkeisfeld $u(x,t)$ bekannt, kann Φ mithilfe dieser Gleichung durch das Lösen von Anfangwertaufgaben gefunden werden. Ist hingegen $\Phi(x,t)$ bekannt, so lässt sich u folgendermaßen bestimmen:

$$u(y,t) = u(\Phi(x,t)t,t) = \Phi_t(x,t))\Phi_t(\Phi^{-1}(y,t),t),$$

für ein gegebenes (y,t) wird also ein x mit $\Phi(x,t) = y$, bzw. $x = \Phi^{-1}(y,t)$ gesucht. Dabei wird $\Phi(.,t) : \Omega \to \Omega$ für festes t als bijektiv vorausgesetzt.

4.1.2. Massenerhaltung & Kontinuitätsgleichungen

Sei $W_0 \subset \Omega$ ein beschränktes Teilgebiet, sowie $W(t) = \Phi(.,t)(W_0) = \{\Phi(x,t)|x \in W_0\}$ das Bild von W_0 unter der Abbildung $\Phi(.,t)$.



Wir beschreiben weiter

$$M(t) = \int_{W(t)} \rho(x,t)dx$$

als Masse in $W(t)$ zum Zeitpunkt t, wobei ρ die lokale Dichte beschreibt. Die Massenerhaltung besagt, dass $M(t)$ konstant ist, d. h. $\frac{d}{dt}M(t) = 0$. Um die Ableitung des Integrals zu berechnen, benötigen wir folgenden Satz:

Satz 4.1 (Reynoldsches Transporttheorem)**.** *Seien u, Φ, W_0 und $W(t)$ gegeben und sei $f(x,t)$ eine C^1-Funktion. Dann gilt:*

$$\frac{d}{dt}\int_{W(t)} f(x,t)dV = \int_{W(t)} \{f_t + \nabla \cdot (fu)\}(x,t)dV$$

Aus der Massenerhaltung folgert man mithilfe des Transporttheorems mit $f = \rho$

$$0 = \frac{d}{dt} \int\limits_{W(t)} \rho dV = \int\limits_{W(t)} \{\rho_t + \nabla \cdot (\rho u)\} dV.$$

Da $W(t)$ ein beliebiges beschränktes Teilgebiet ist, folgt für ein genügend glattes ρ (Lokalisierung) die **Kontinuitätsgleichung**

$$\rho_t + \nabla \cdot (\rho u) = 0. \tag{4.1}$$

4.1.3. Impulserhaltung

Seien $u, \Phi, W_0, W(t)$ und ρ wie oben gegeben, dann ist $I(t) = \int_{W(t)} (\rho u)(y,t) dV$ der Impuls der Masse $M(t)$ in $W(t)$. Nach dem 2. Newtonschen Gesetz gilt: Änderung des Impulses = Kraft, und somit

$$\frac{d}{dt} I(x) = K(t), \tag{4.2}$$

mit der Kraft $K(t)$, die auf die Masse $W(t)$ einwirkt. Wir nehmen an, dass sich $K(t)$ aufteilen lässt in

- eine Volumenkraft
- eine Kraft, die auf den Rand $\partial W(t)$ wirkt.

Die Volumenkraft lasse sich darstellen als

$$K_1(t) = \int\limits_{W(t)} \rho(u,t) F(y,t) dV,$$

wobei $F(y,t)$ ein gegebenes Feld bezeichnet, bei der Gravitation gilt z. B. $F = (0,0,-g)^T$. Dabei hat ρdV die Dimension einer Masse, daher ergibt sich für F die Dimension einer Beschleunigung. Für die Randkraft gelte

$$K_2(t) = \int\limits_{\partial W(t)} \sigma(y,t) n(y,t) dS,$$

mit der äußeren Einheitsnormalen $n(y,t)$ auf $\partial W(t)$ im Punkt y und dem Spannungstensor $\sigma(x,t) \in \mathbb{R}^{3\times 3}$. Da dS die Dimension einer Fläche hat, gilt für die Dimension des Spannungstensors

$$[\sigma] = \left[\frac{\text{Kraft}}{\text{Fläche}}\right] = [\text{Druck}].$$

Analog zur Kontinuitätsgleichung folgt aus der Impulserhaltung (4.2) und dem Reynoldschen Transporttheorem die **Impulsgleichung**

$$(\rho u)_t + \nabla \cdot (\rho u u^T) = \rho F + \nabla \cdot \sigma. \tag{4.3}$$

Anmerkung
Aus der Dreh-Impulserhaltung folgt, dass σ symmetrisch ist.
$\longrightarrow$ Übungsaufgabe

4.2. Navier-Stokes-Gleichungen

4.2.1. Berechnung von σ, u, p

Üblicherweise nimmt der Spannungstensor die Form

$$\sigma = -pI + \tau$$

an, mit dem viskosen Spannungstensor τ und dem Druck $p = p(x,t)$. Wir definieren den Deformationstensor def, bzw. den symmetrischen Gradienten ϵ

$$(\mathrm{def}(u))_{ij} = \partial_i u_j + \partial_j u_i, \qquad \epsilon_{ij} = \frac{1}{2}(\partial_i u_j + \partial_j u_i)$$

bzw. in anderer Notation:

$$\mathrm{def} = \nabla u + \nabla u^T, \qquad \epsilon = \frac{1}{2}(\nabla u + \nabla u^T).$$

Die Navier-Stokes-Gleichungen gehen von linearem mechanischem Verhalten aus (kleine Verzerrungen). Aus der Annahme der Linearität und der Symmetrie des viskosen Spannungstensors folgt der Ansatz Navier-Stokes-Gleichungen

$$\begin{aligned}\tau &= \lambda(\nabla \cdot u)I + 2\mu\epsilon \\ &= \lambda\ \mathrm{spur}(\epsilon)I + 2\mu\epsilon,\end{aligned}$$

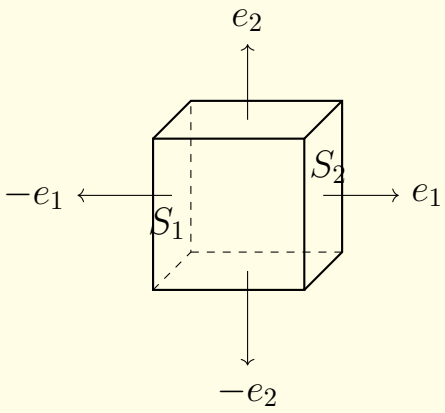
mit den von u unabhängigen Materialeigenschaften μ, λ. Einsetzen in die Impulsgleichung (4.3) liefert

$$(\rho u)_t + \nabla \cdot (\rho u u^T) + \nabla p = \rho F + \nabla(\lambda \nabla \cdot u) + \nabla \cdot (2\mu\epsilon).$$

Anmerkung

Zur Berechnung von $\sigma = -pI + \tau$:

a) Zur Veranschaulichung des Terms $-pI$ betrachten wir einen Kubus.

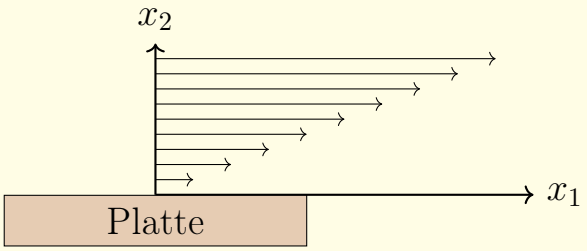


An der Fläche S_1 wirke der Druck p_1, an S_2 der Druck $p_2 < p_1$. Dann gilt:

$$\int\limits_{S_1 \cup S_2} -pn dS = |S|\{-p_1(-e_1) - p_2 e_1\} = |S|(p_1 - p_2)e_1,$$

der Druckabfall führt somit zu einer Kraft nach rechts in Richtung e_1.

b) Um den Anteil von τ zu veranschaulichen, betrachten wir folgendes Geschwindigkeitsfeld



$$u_1(x_1, x_2, x_3) = kx_2, \quad u_2 = u_3 = 0, \quad k > 0.$$

Auf eine Platte wirke eine Reibungskraft in Richtung e_1. Es gilt:

$$\nabla u = \begin{pmatrix} 0 & k & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \text{somit } \epsilon = \frac{1}{2}\begin{pmatrix} 0 & k & 0 \\ k & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

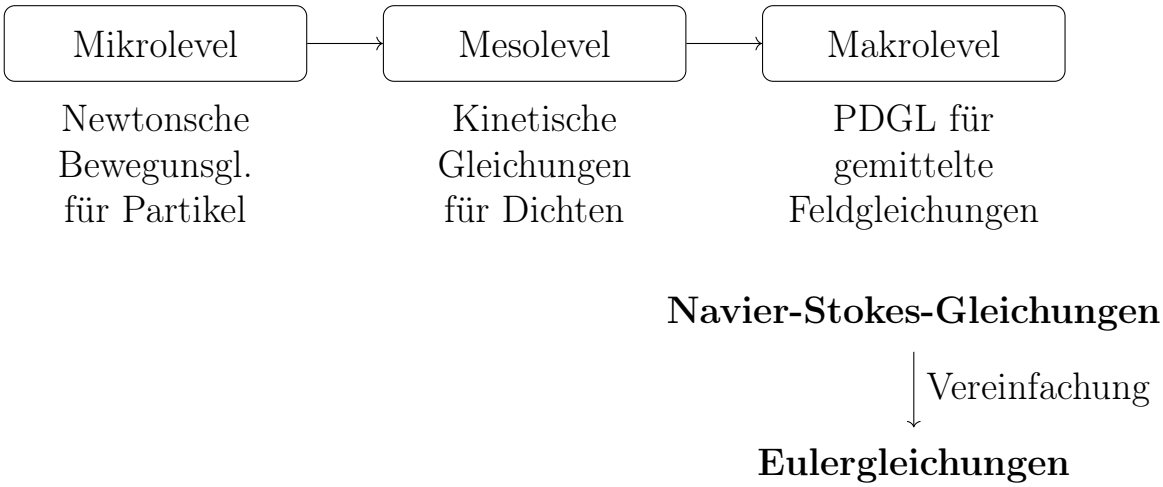
Für τn folgt mit $n = e_2$ nun:

$$\tau n = \lambda(\nabla \cdot u)n + \mu 2\epsilon u n = \mu k e_1.$$

Dies passt zu der Betrachtung, dass eine Kraft in Richtung e_1 auf die Platte wirkt.

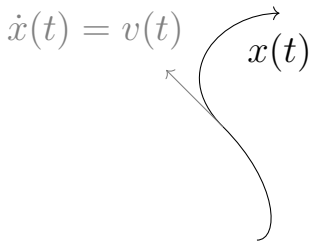
Extra: Vom Partikel zur Flüssigkeit: Fluiddynamik

Idee:



4.2.2. Erinnerung: Bewegungsgleichungen für Teilchen im Erdschwerefeld

Die Teilchenbewegung wird beschrieben durch den Ort $x : \mathbb{R}^+ \to \mathbb{R}^3$ und die Geschwindigkeit $v : \mathbb{R}^+ \to \mathbb{R}^3$.
Mit Newton folgt:



$$\dot{x}(t) = v(t)$$
$$\dot{v}(t) = g = \begin{pmatrix} 0 \\ 0 \\ \bar{g} \end{pmatrix}, \quad \text{mit } \bar{g} = 9.80665 \frac{m}{s^2}.$$

Darüber hinaus bezeichne $p(t) = m \cdot v(t)$ den Impuls eines Teilchens der Masse m. Für große Teilchenzahlen ist diese Darstellung sehr unangenehm, da man pro Teilchen 6 Gleichungen benötigt. Daher geht man zur Beschreibung durch Verteilungsdichten über.

4.2.3. Boltzmann-Gleichung

ein paar kleine Namenskollisionen mit der Notation von Markus: F vs. g, K vs. W, ρ vs. M (hier erstmal Christians Namen)

Betrachte die Verteilungsdichte $f(t, x, v)$ mit $f : \mathbb{R}^+ \times \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^+$ der Partikel, die zum Zeitpunkt t am Ort x die Geschwindigkeit v haben. Es gilt

$$\int \int f(t, x, v) dx dv = 1$$

Um zu verstehen, wie sich f mit der Zeit verändert, betrachten wir die Zeitableitung:

$$\frac{d}{dt} f = \partial t f + \dot{x} \nabla_x f + \dot{v} \nabla_v f.$$

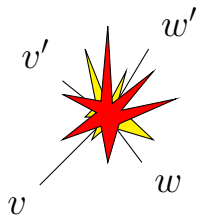
Ist die freie Weglänge groß gegenüber der Partikelgröße, erhält man daraus die klassische *Boltzmanngleichung*

$$\partial_t f + v \nabla_x f + g \cdot \nabla_v f = Q(f, f), \tag{4.4}$$

mit dem (nichtlinearen) Kollisionsoperator $Q(f, f) = Q^+(f, f) - Q^-(f, f)$ wobei

$$Q^+(f, f) = \int \int \int W(v', w', v, w) f(v') f(w') dv' dw' dw$$
$$Q^-(f, f) = \int \int \int W(v, w, v', w') f(v) f(w) dv' dw' dw.$$

Dabei ist $W(v, w, v', w')$ die Wahrscheinlichkeit dafür, dass 2 Teilchen mit Geschwindigkeit v und w kollidieren und danach die Geschwindigkeiten v' und w' haben ($(v, w) \to (v', w')$ bzw. $(v', w') \to (v, w)$).



4.2.4. Momentengleichung

Problem der Boltzmann-Gleichung: die hohe Dimensionalität ist schwierig zu handhaben.
Idee: Mittelung, Momentenentwicklung

0. Moment:

Wir suchen nun eine Gleichung für die Massendichte

$$M(x, t) = \int m f(t, x, v) dv$$

Durch Multiplikation von (4.4) mit m und Integration bezüglich v erhält man

$$\underbrace{\int m \partial_t f dv}_{(a)} + \underbrace{\int m v \cdot \nabla_x f dv}_{(b)} + \underbrace{\int m g \cdot \nabla_v f dv}_{(c)} = \underbrace{\int m Q(f, f) dv}_{(d)},$$

dessen Terme wir nun im Einzelnen betrachten:

a) $\to \partial_t M$

b)

$$\int m v \cdot \nabla_x f dv = \nabla_x \cdot (Mu), \text{ mit}$$

$$u = \frac{1}{M} \int m v f dv \quad \text{(mittlere Feldgeschwindigkeit in } (t, x))$$

c)

$$\int mg \cdot \nabla_v f dv = m \int \nabla_v \cdot (gf) dv \overset{\text{Satz von Gauß}}{=} \int (gf) \cdot n dA = 0,$$

da f kompakten Träger hat.

d)

$$\begin{aligned}\int mQ(f,f)dv &= \int mQ^+(f,f)dv - \int mQ^-(f,f)dv \\ &= m \iiiint W(v',w',v,w) f(v') f(w') dv' dw' dw dv \\ &\quad - m \iiiint W(v,w,v',w') f(v) f(w) dv' dw' dw dv \\ &= 0,\end{aligned}$$

wobei es sich um eine Konsequenz der physikalischen Massenerhaltung handelt.

Insgesamt erhalten wir für M so die partielle Differentialgleichung

$$\partial_t M + \nabla_x \cdot (Mu) \quad \text{(Massentransport)}. \tag{4.5}$$

Jetzt brauchen wir eine Gleichung für u.

1. Moment:

$u^T u$ vs. uu^T überprüfen

Multipliziere Gleichung (4.4) mit mv, dann integriere bzgl. $v = (v_1, v_2, v_3)^T$

$$\underbrace{\int mv \partial_t f dv}_{a)} + \underbrace{\int mvv^T \nabla_x f dv}_{b)} + \underbrace{\int mvg^T \nabla_v f dv}_{c)} = \underbrace{\int mvQ(f,f)dv}_{d)},$$

wobei für die einzelnen Terme gilt:

a)

$$\int mv \partial_t f dv = \partial_t (Mu)$$

b)

$$\int mvv^T \nabla_x f dv = \nabla_x \cdot \left[\int mvv^T f dv \right]$$

Setze nun $w = v - u$ ($w \approx$ *random motion*) und betrachte

$$\begin{aligned} \int w^T w m f dv &= \int (v-u)^T (v-u) m f dv \\ &= \int (v^T v - v^T u - u^T v + u^T u) m f dv \\ &= \underbrace{\int vv^T v m f dv}_{=Mu^T u} - \underbrace{\int v^T u m f dv}_{=Mu^T u} - \underbrace{\int u^T u m f dv}_{=Mu^T u} \end{aligned}$$

$$\rightarrow \int v^T v m f dv = Mu^T u + \int w^T w m f dv.$$

Daher schreiben wir für (b)

$$\nabla_x \cdot (Mu^T u + M\langle w, w \rangle) \quad \text{mit } \langle w, w \rangle = \frac{1}{M} \int u^T w m f dv.$$

Beachte: $u^T u$ und $\langle u, w \rangle$ sind Matrizen (Tensoren).

c) Diesen Term schreiben wir mit partieller Integration um zu

$$\begin{aligned} m \int vg \nabla_v f dv &= m \int vgf dA - m \int \nabla_v (vg) f dv \\ &= 0 - g \int m f dv \\ &= -gM. \end{aligned}$$

d)

$$\int vmQ(f,f) dv = 0 \quad \text{(Impulserhaltung)}$$

Insgesamt erhalten wir somit

$$\partial_t (Mu) + \nabla_x \cdot (Mu^T u + M\langle w, w \rangle) = g \cdot M$$

Den Tensor $\langle w, w \rangle$ zerlegen wir nun in zwei Anteile:

- den Druck $P = \frac{1}{3}M\langle w, w\rangle$
- den viskosen Spannungstensor $\Pi = \delta_{ii}P - M\langle w, w\rangle$.

Damit erhalten wir

$$\partial_t(Mu) + \nabla_x \cdot (Mu^T u + P - \Pi) = gM. \tag{4.6}$$

Wir können nun die Kontinuitätsgleichung (4.5) benutzen, um (4.6) umzuschreiben in eine Gleichung für u. (4.6) schreiben wir mit der Produktregel um zu

$$(\partial_t M)u + M\partial_t u + [\nabla \cdot (Mu)]u + Mu \cdot \nabla \cdot u + \nabla \cdot (P - \Pi) = gM$$

Aus (4.5) haben wir: $\partial_t M = -\nabla \cdot (Mu)$

$$\partial_t u + (u \cdot \nabla)u + \frac{1}{M}\nabla \cdot (P - \Pi) = g. \tag{4.7}$$

Das ist die *Navier-Stokes-Gleichung*!
In der Regel sieht man diese in der inkompressiblen Form mit der Zusatzbedingung $\nabla \cdot u = 0$. Vernachlässigt man nun den Effekt innerer Reibung kann man auf den Viskositätsterm $\nabla \cdot \Pi$ verzichten und erhält als Vereinfachung die *Euler-Gleichungen*

$$\partial_t u + (u \cdot \nabla)u + \frac{1}{M}\nabla \cdot p = g. \tag{4.8}$$

4.2.5. Existenz & Eindeutigkeit von Lösungen

Die Existenz von Lösungen der Navier-Stokes-Gleichungen ist schwer zu beweisen und eines der Millenium-Probleme für dessen Lösung das CMI einen Preis von einer Million US-Dollar ausgeschrieben hat.

4.3. Navier-Stokes-Gleichungen für inkompressible Fluide

Zur Erinnerung: Wir betrachten

$$\begin{aligned}\partial_t \rho + \nabla \cdot (\rho u) &= 0\\ \partial_t(\rho u) + \nabla \cdot (\rho u u^T) + \nabla p &= \rho F + \nabla(\lambda \nabla \cdot u) + \nabla \cdot (2\mu\epsilon).\end{aligned}$$

Wir verwenden

$$\partial_t \rho u = u_t \partial_t \rho + \rho \partial_t u \qquad \text{und} \quad \nabla \cdot (\rho u u^T) = u \nabla \cdot (\rho u) + \rho u \cdot \nabla u$$

und erhalten mit der Kontinuitätsgleichung $\partial_t \rho + \nabla \cdot (\rho u) = 0$

$$\partial_t \rho + \nabla \cdot (\rho u) = 0 \tag{4.9a}$$

$$\partial_t u + (u \cdot \nabla)u + \frac{1}{\rho}\nabla p = F + \frac{\lambda + \mu}{\rho}\nabla(\nabla \cdot u) + \frac{\mu}{\rho}\Delta u \tag{4.9b}$$

mit F, μ, λ gegeben
und ρ, p, u unbekannt $\Big\}$ $\rightarrow$ unterbestimmt!

Für inkompressible Fluide gilt $\rho = \rho_0 = \text{const}$, dadurch vereinfacht sich die Kontinuitätsgleichung (4.9a) zu

$$\nabla \cdot u = 0$$

und für die Impulserhaltung folgt

$$\begin{aligned}\partial_t u + (u \cdot \nabla)u + \frac{1}{\rho_0}\nabla p &= F + \frac{\mu}{\rho_0}\Delta u\\ \nabla \cdot u &= 0\end{aligned} \tag{4.10}$$

Wir nennen $\nu = \frac{\mu}{\rho_0}$ kinematische Viskosität und normieren $\rho_0 = 1$

$$\begin{aligned}\partial_t u + (u \cdot \nabla)u + \nabla p &= F + \nu\Delta u\\ \nabla \cdot u &= 0\end{aligned} \tag{4.11}$$

Anmerkung

- $\nu > 0$: inkompressible Navier-Stokes-Gleichung
- $\nu = 0$: inkompressible Euler-Gleichung
- p ist ein Lagrangemultiplikator für die Nebenbedingung $\nabla \cdot u = 0$.

4.3.1. Stokes-Gleichungen

Für kleines u können wir annähernd lineares Verhalten annehmen, d. h. $(u \cdot \nabla) \approx 0$, was auf die *Stokes-Gleichungen*

$$\begin{aligned} \partial_t u - \nu \Delta u + \nabla p &= f \\ \nabla \cdot u &= 0 \end{aligned}$$

4.4. Euler-Gleichungen und volle Navier-Stokes-Gleichungen

4.4.1. Navier-Stokes-Gleichungen für kompressible Fluide

Wenn Kompressibilität berücksichtigt werden muss, wir aber Temperatureffekte vernachlässigen können, vervollständigen wir (4.9a),(4.9b) durch eine Zustandsgleichung

$$p = \bar{p}(\rho, T),$$

oft wird das System als iso-therm angenommen und

$$p = p_0 \left(\frac{\rho}{\rho_0} \right)^\gamma$$

gewählt, wobei es verschiedene Modelle zur Wahl von γ gibt, z. B. für Luft $\gamma \approx \frac{7}{5}$. Mit $p = \bar{p}(\rho)$ folgt für die Unbekannten ρ, u das System

$$\begin{aligned} \partial_t \rho + \nabla \cdot (\rho u) &= 0 \\ \partial_t u + (u \cdot \nabla) u + \frac{1}{\rho} \bar{p}'(\rho) \nabla \rho_i &= F + \frac{\lambda + \mu}{\rho} \nabla (\nabla \cdot u) + \frac{\mu}{\rho} \Delta u, \end{aligned}$$

für $\mu = \lambda = 0$ ergibt sich eine Form der Euler-Gleichungen.

4.4.2. Energieerhaltung

Betrachte die Dichte ρ, den Druck p, die Temperatur T und die innere Energie e mit

$$p = p(\rho, T), \quad e = e(\rho, T).$$

Dann kann die Energie im System beschrieben werden durch

$$E(t) = \int_{W(t)} \underbrace{\rho e}_{pot.Energie} + \underbrace{\frac{1}{2}\rho|u|^2}_{kin.Energie} \, dx$$

Eine Änderung der Energie kann durch verschieden Faktoren bewirkt werden:

- kinetische Energie ändert sich durch die gleichen Kräfte wie in der Impulsgleichung
- Änderung der inneren Energie folgt aus 1. Hauptsatz der Thermodynamik.

Die zeitliche Änderung der inneren Energie e ist i. A. gleich der eingebrachten Wärme Q und der verrichteten Arbeit A

$$\mathrm{d}_t e = Q + A,$$

wobei hier A der Expansion oder Kompression des Fluids entspricht. Es folgt die Erhaltungslgleichung der totalen Energie E

$$\begin{aligned}\frac{d}{dt}\int_W \rho e - \frac{1}{2}\rho|u|^2 dx = &\int_W \underbrace{\rho F \cdot u}_{\text{Volumenkraft}} \, dx + \int_{\partial W} \underbrace{(\sigma u)\cdot n}_{\text{Oberflächenkraft}} \, dS \\ &+ \int_W \underbrace{\rho Q}_{\text{Wärmezufuhr}} \, dx + \int_{\partial W} \underbrace{K\nabla T \cdot n}_{\text{Wärmefluss}} dS,\end{aligned}$$

wobei K den Wärmeleitungskoeffizienten bezeichnet. Durch Lokalisierung erhalten wir nun die **Energiegleichung**

$$\partial_t(\frac{1}{2}\rho|u|^2 + \rho e) + \nabla\cdot((\frac{1}{2}\rho|u|^2 + \rho e)u) = \rho F\cdot u + \rho Q + \nabla\cdot(\sigma u) + \nabla\cdot(K\nabla T),$$

mit $\sigma = -pI + \tau$, $\tau = \lambda(\nabla\cdot u)I + \frac{1}{2}\epsilon$

Zusätzlich benötigen wir Zustandsgleichungen für $p(\rho, T)$ und $e(T)$, z. B.

- ideales Gasgesetz $p(\rho, T) = \rho RT$
- $e(T) = c_\nu T$

Welche Wahl wir hier treffen ist keine mathematische Frage, sondern eine Modellierungsfrage!

Anmerkung
Mit $u = 0$, $F = 0$, ρ konstant folgt die Wärmeleitungsgleichung

$$\begin{aligned} \partial_t(\rho c_\nu T) &= \nabla \cdot (K \nabla T) + \rho Q \\ \Rightarrow \partial_t T - \frac{K}{\rho c_\nu} \Delta T &= \frac{Q}{c_\nu} \qquad \text{(Wärmeleitungsgleichung)} \end{aligned}$$

4.4.3. Reibungsfreier Fall für dünne Gase

Für $\nu = 0$ gilt bei Vernachlässigung der kinetischen Energie (kaum Masse)

$$\underbrace{(F = 0), Q = 0, \nabla T = 0}_{\text{abgeschlossenes System, konstante Temperatur}} .$$

Es folgt die

allgemeine Form der Eulergleichungen

$$\begin{aligned} \partial_t \rho + \nabla \cdot (\rho u) &= 0 \\ \partial_t (\rho u) + \nabla \cdot (\rho u u^T) &= -\nabla p + (\rho F) \\ \partial_t E + \nabla \cdot (Eu) &= -p \nabla \cdot u, \end{aligned}$$

wobei $E = \rho \cdot e$ mit der spezifischen Energie e.

4.4.4. Ausbreitung akustischer Wellen

Akustische Wellen breiten sich in Form von Druckschwankungen aus, die mit Dichteschwankungen korrespondieren. Wir formulieren p, ρ, u als Konstanten/mittelere Grundwerte + Störung

$$\begin{aligned} \rho(t) &= \bar{\rho} + \tilde{\rho}(t) \\ p(t) &= \bar{p} + \tilde{p}(t) \\ v(t) &= \bar{v} + \tilde{v}(t). \end{aligned}$$

Wir setzen $\bar{u} \equiv 0$ (Schwerpunktsystem), $\nabla \bar{p} = 0$. Das Verwenden des idealen Gasgesetztes

$$p = \rho RT$$
$$c = \sqrt{RT}$$

liefert

$$p = c^2 \rho = c^2(\bar{\rho} + \tilde{\rho}) = \bar{p} + \tilde{p}.$$

Linearisieren um $\bar{\rho}$ und $\bar{p}$ und Einsetzen in die Eulergleichungen liefert

$$\partial_t \tilde{\rho} + \nabla \cdot (\bar{\rho}\tilde{u}) = 0$$
$$\partial_t (\bar{\rho}\tilde{u}) + \nabla \tilde{p} = 0$$

Nicht konservative Form der linearen Akustik:

Sei c konstant in Ω, mit $\tilde{\rho} = \frac{\tilde{p}}{c^2}$ folgt dann

$$\partial_t \tilde{p} + c^2 \bar{\rho} \nabla \cdot \tilde{u} = 0 \tag{4.12}$$
$$\bar{\rho} \partial_t \tilde{u} + \nabla \tilde{p} = 0 \tag{4.13}$$

mit Materialkonstanten c, $\bar{\rho}$. Die Zeitableitung von (4.12) und die Divergenz von (4.13) liefern die *Wellengleichung*

$$\partial_t^2 \tilde{p} - c^2 \Delta \tilde{p} = 0.$$

Anmerkung
Herleitung

$$\begin{aligned} & \partial_t^2 \tilde{p} + \partial_t c^2 \bar{\rho} \nabla \cdot \tilde{u} = 0 \\ & \nabla \cdot (\bar{\rho} \partial_t \tilde{u} + \nabla \tilde{p}) = 0 \\ \iff \quad & \partial_t^2 \tilde{p} + \bar{\rho} \nabla \cdot \partial_t c^2 \tilde{u} = 0 \\ & \bar{\rho} \nabla \cdot \partial_t \tilde{u} = -\nabla \cdot \nabla \tilde{p} \\ \implies \quad & \partial_t^2 \tilde{p} - c^2 \nabla \cdot \nabla \tilde{p} = 0 \end{aligned}$$

4.5. Typeinteilung von partiellen Differentialgleichungen 2. Ordnung

Wir betrachten im Folgenden (lineare) PDGL 2. Ordnung, welche sich mit Hilfe eines Differentialoperators L schreiben lassen in der Form

$$Lu = f, \quad L := \sum_{i,j=1}^{d} a_{ij}\partial_i\partial_j + \sum_{j=1}^{d} a_j\partial_j + a.$$

Da die partiellen Ableitungen vertauscht werden können ($\partial_i\partial_j = \partial_j\partial_i$), können die Koeffizienten $a_{ij} = a_{ji}$ als symmetrisch angenommen werden.

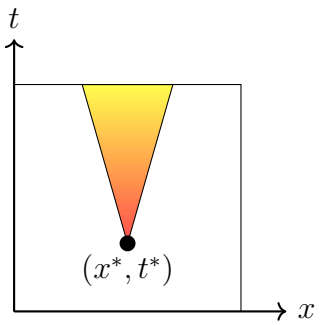
Beispiel 4.1 Transportgleichung: Wir betrachten das eindimensionale Transportproblem auf dem Gebiet $\Omega \times I \subset \mathbb{R} \times \mathbb{R}_+$

$$\partial_t\rho(x,t) + a\partial_x\rho(x,t) = 0$$

mit Geschwindigkeit a und Anfangswerten $\rho(x,0) = \rho_0(x)$. Die eindeutige Lösung nimmt die Form

$$\rho(x,t) = \rho_0(x - at), \quad t \geq 0$$

an.



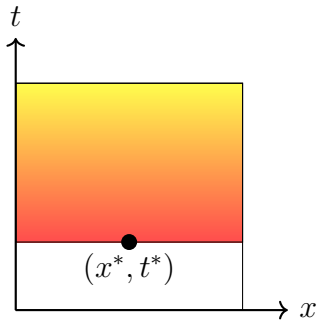
Angenommen, die Lösung wird in einem Punkt $(x^*, t^*) \in \Omega \times I$ gestört, wird diese Störung mit der Geschwindigkeit v weitertransportiert. Die Störung hat nur einen Einfluss auf Punkte, für die $t \geq t^*$ gilt, und da die Ausbreitung mit endlicher Geschwindigkeit v stattfindet, liegt der Einflussbereich der Störung innerhalb eines Kegels.

Beispiel 4.2 Wärmeleitungsgleichung: Wir betrachten die eindimensionale Wärmeleitungsgleichung auf $\Omega \times I \subset \mathbb{R} \times \mathbb{R}_+$

$$\partial_t u(x,t) - \lambda\partial_x^2 u(x,t) = f(x,t)$$

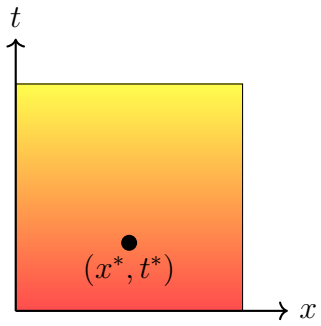
mit den Anfangswerten und Randbedingungen

$$u(x,0) = u_0(x)$$
$$\partial_n u(x,t) = 0 \quad \text{auf } \partial\Omega$$



Hier wirkt sich eine Störung im Punkt (x^*, t^*) unmittelbar auf alle nachfolgenden Zeit-/Ortspunkte, d. h. alle Punkte (x, t) mit $t \geq t^*$ aus.

Beispiel 4.3 Stationäre Wärmeleitungsgleichung:



Im Falle der stationären Wärmeleitungsgleichung hat eine Störung in einem beliebigen Punkt $x \in \Omega$ Einfluss auf alle Punkte im Gebiet Ω.

4.6. Burgersgleichung

Die Burgersgleichung ist ein typisches Modellproblem zur Untersuchung numerischer Probleme bei hyperbolischen Systemen. Wir gehen von den kompressiblen Eulergleichungen aus und betrachten die Impulsgleichung

$$\partial_t \rho u + \nabla \cdot (\rho u u^T) = -\nabla p + \rho F.$$

Für den Fall $\nabla p = 0, F = 0, \rho = 1$ erhalten wir

$$\partial_t u + \nabla \cdot (u u^T) = 0,$$

im eindimensionalen Fall mit dem "unmotivierten" Faktor $\frac{1}{2}$ erhalten wir die nicht viskose Burgersgleichung

$$\begin{aligned} \partial_t u + \frac{1}{2}\partial_x u^2 &= 0 \qquad\qquad \text{in } \mathbb{R} \times \mathbb{R}_+, \quad u = u(x,t) \\ u(x,0) &= u_0(x). \end{aligned}$$

Anmerkung
Eigentlich betrachtete Burgers die viskose Gleichung

$$\partial_t u + \frac{1}{2}\partial_x u^2 = \epsilon \partial_x^2 u.$$

Diese Gleichung lässt sich über die kompressiblen Navier-Stokes-Gleichungen motivieren und erlaubt die Untersuchung einiger zentraler Eigenschaften der Gasdynamik.

4.6.1. Schwache Lösung

Gegeben sei eine nichtlineare hyperbolische Gleichung

$$\begin{aligned} \partial_t u + \partial_x f(u) &= 0 \quad \text{in } \mathbb{R} \times \mathbb{R}_+ \\ u(x,0) &= u_0(x), \end{aligned} \tag{4.14}$$

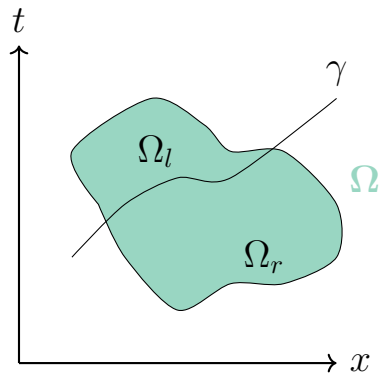
ein Beispiel ist die Burgersgleichung mit $f(u) = \frac{u^2}{2}$. u ist eine schwache Lösung von (4.14), wenn

$$\int_0^\infty \int_{-\infty}^\infty [u(x,t) \cdot \partial_t \Phi(x,t) + f(u(x,t)) \cdot \partial_x \Phi(x,t)]\, dxdt + \int_{-\infty}^\infty u_0(x,t)\Phi(x,0)dx = 0 \tag{4.15}$$

für alle Testfunktionen $\Phi \in C_0^1(\mathbb{R} \times \mathbb{R}_+)$. Man zeigt: Ist u glatte Lösung von (4.15), dann ist u auch eine Lösung von (4.14).

4.6.2. Rankine-Hugoniot-Bedingung

Bei der *Rankine-Hugoniot-Bedingung* handelt es sich um eine notwendige Bedingung, die eine schwache Lösung an einer Sprungstelle erfüllen muss. Wir betrachten ein offenes Teilgebiet $\Omega \subset \mathbb{R} \times \mathbb{R}_+$ und eine Kurve $\gamma = (\hat{x}(t), t)$, die Ω in die beiden Teilgebiete Ω_l und Ω_r zerlegt.



Definition 4.1 Sprung: Für einen Punkt $(x,t) \in \gamma$ definieren wir den *Sprung* von u als

$$[u](x,t) = \lim_{(x',t') \to ((x,t) \text{ in } \Omega_l} u(x',t') - \lim_{(x',t') \to ((x,t) \text{ in } \Omega_r} u(x',t')$$

Satz 4.2. *Sei u eine schwache Lösung von (4.14) im Sinne von (4.15) mit folgenden zusätzlichen Bedingungen:*

1. *u ist eine klassische Lösung in Ω_l, Ω_r*
2. *u ist unstetig entlang γ, d. h. $[u](\hat{x}(t),t) \neq 0$ auf γ*
3. *$[u]$ ist stetig entlang von γ, d. h. $[u](\hat{x}(t),t)$ ist stetig in t.*

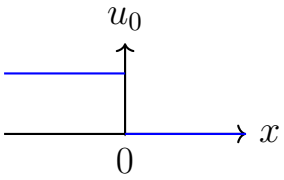
Dann gilt

$$\frac{d}{dt}\hat{x}(t) \cdot [u](\hat{x}(t),t) = [f(u)](\hat{x}(t),t) \quad \forall (\hat{x}(t),t) \in \gamma$$

Beispiel 4.4: linearer Fall:

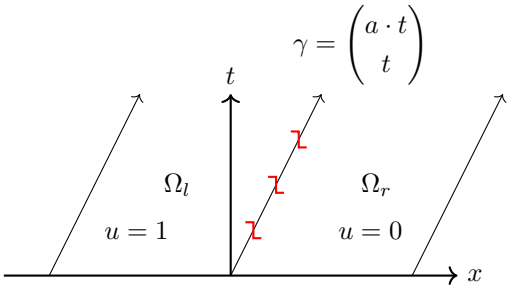
Wir betrachten

$$\partial_t u + a\partial_x u = 0, \quad u_0(x) = \begin{cases} 1, & x \leq 0 \\ 0, & \text{sonst} \end{cases}$$



Dann ist $\hat{x}'(t) = a$ die Geschwindigkeit, mit der sich der Sprung bewegt. Dann gilt

$$\frac{[f(u)](\hat{x}(t),t)}{[u](\hat{x}(t),t)} = \frac{a \cdot 1 - a \cdot 0}{1 - 0} = a$$



nichtlinearer Fall:

Betrachte

$$\partial_t u + \partial_x f(u) = 0 \quad \text{mit } f(u) = \frac{u^2}{2}.$$

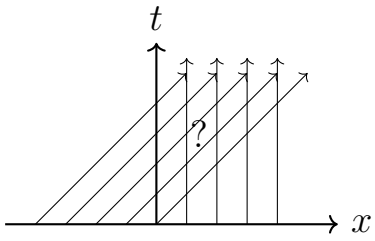
Mit $f'(u) = u$ gilt dann:

$$\partial_t u + \partial_x f(u) = \partial_t u + f'(u(x,t))\partial_x u = \partial_t u + u\partial_x u = 0$$

mit der Steigung $u(x,t)$.

Fall 1: Die Anfangswerte seien gegeben durch

$$u_0(x) = \begin{cases} 1, & x \leq 0 \\ 0, & \text{sonst.} \end{cases}$$



Für die Schockgeschwindigkeit gilt dann

$$\hat{x}'(t) = \frac{[f(u)]}{[u]} = \frac{\frac{1^2}{2} - \frac{0^2}{2}}{1-0} = \frac{1}{2}.$$

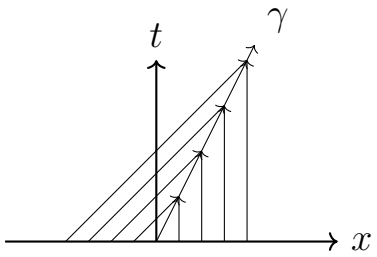
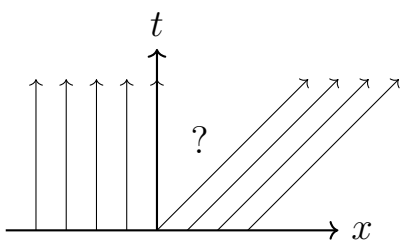


Abbildung 4.1.: schwache Lösung

Fall 2: Seien die Anfangswerte gegeben sein durch

$$u_0(x) = \begin{cases} 0, & x \leq 0 \\ 1, & \text{sonst.} \end{cases}$$



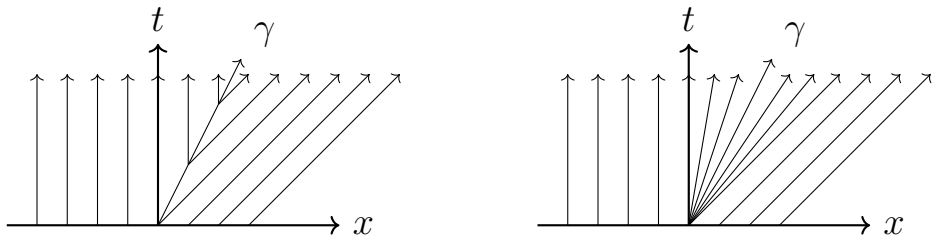
Dann gilt für die Schockgeschwindigkeit

$$\hat{x}'(t) = \frac{[f(u)]}{[u]} = \frac{\frac{0^2}{2} - \frac{1^2}{2}}{0-1} = \frac{1}{2}.$$

Zwei Beispiel für schwache Lösungen sind die Funktionen u_1 und u_2 mit

$$u_1(x,t) = \begin{cases} 0, & x < \frac{t}{2} \\ 1, & x > \frac{t}{2} \end{cases}, \quad u_2(x,t) = \begin{cases} 0, & x < 0 \\ \frac{x}{t}, & 0 \leq \frac{x}{t} \leq t \\ 1, & x > t \end{cases}.$$

Abbildung 4.2.: schwache Lösungen: links: u_1, physikalisch inkorrekt; rechts: u_2, Verdünnungswelle



Anmerkung

- Die Rankine-Hugoniot-Sprungbedingung ist eine notwendige Bedingung, die Auskunft über die Schockgeschwindigkeit gibt.
- Schwache Lösungen sind nicht notwendigerweise eindeutig.

4.6.3. Viskositätslimes

Die kompressiblen Navier-Stokes-Gleichungen in einer Raumdimension sind gegeben durch

$$\begin{aligned} \partial_t \rho_\epsilon + \partial_x(\rho_\epsilon u_\epsilon) &= 0 \quad \text{in } \mathbb{R} \times \mathbb{R}_+ \\ \partial_t(\rho_\epsilon u_\epsilon) + \partial_x(\rho_\epsilon u_\epsilon^2 + p_\epsilon) &= \epsilon \partial_x^2 u_\epsilon \quad \text{in } \mathbb{R} \times \mathbb{R}_+. \end{aligned} \tag{4.16}$$

Wie betrachten nun eine Folge von Lösungen $(\rho_\epsilon, u_\epsilon, p_\epsilon)$ bezüglich der Viskosität ϵ für $\epsilon \to 0$. Für $\epsilon \to 0$ folgen die Eulergleichungen

$$\begin{aligned} \partial_t \rho + \partial_x(\rho u) &= 0 \quad \text{in } \mathbb{R} \times \mathbb{R}_+ \\ \partial_t(\rho u) + \partial_x(\rho u^2 + p) &= 0 \quad \text{in } \mathbb{R} \times \mathbb{R}_+ \end{aligned} \tag{4.17}$$

Für $\epsilon > 0$ besitzt (4.16) eine eindeutige Lösung, (4.17) hingegen ist nicht mehr eindeutig lösbar. Wir suchen eine Lösung von (4.17), welche dem Limes $\epsilon \to 0$ der Lösung von (4.16) entspricht. Der Übergang $u_\epsilon \to u$ nennt sich Viskositätslimes. Mathematisch führt dies auf

Definition 4.2 Lax-Entropiebedingung: Sei u eine schwache Lösung von (4.14) und sei γ eine glatte Kurve in $\mathbb{R} \times \mathbb{R}_+$, entlang der u unstetig ist. Sei $(x_0, t_0) \in \gamma$,

$$u_l := \lim_{(x',t') \to (x,t) \text{in} \Omega_l} (x', t'), \quad u_r := \lim_{(x',t') \to (x,t) \text{in } \Omega_r} (x', t'), \quad s := \frac{f(u_l) - f(u_r)}{u_l - u_r}.$$

Dann erfüllt u die *Lax-Entropiebedingung* in (x_0, t_0) genau dann, wenn gilt:

$$f'(u_r) < s < f'(u_l).$$

Eine Unstetigkeit, die sowohl die Lax-Entropiebedingung als auch die Rankine-Hugoniot-Bedingung erfüllt, heißt *Schock*, s nennt sich *Schockgeschwindigkeit*.

4.6.4. Finite Differenzen-Verfahren für die Burgersgleichung

Wir betrachten folgendes FD-Verfahren :
Für genügend glattes u gilt:

$$\partial_t u + \partial_x \left(\frac{u^2}{2} \right) = \partial_t u + u \partial_x u = 0, \quad u_0(x) = \begin{cases} 1, & x \leq 0 \\ 0, & \text{sonst} \end{cases}$$

Mit dem expliziten Eulerverfahren und der Finite Differenzen-Diskretisierung gilt:

$$\begin{aligned} & \frac{u_i^{k+1} - u_i^k}{\tau} + u_i^k \cdot \frac{u_i^k - u_{i-1}^k}{h} = 0 \\ \iff \quad & u_i^{k+1} = u_i^k - \frac{\tau}{h} u_i^k (u_i^k - u_{k-1}^k) \end{aligned}$$

mit den diskreten Schrittweiten τ, h. Das Verfahren findet nicht die richtige Lösung:

- falsche Schockgeschwindigkeit (0 statt $\frac{1}{2}$) $\rightarrow u(t) = \text{const}$
- unabhängig von τ, $h \rightarrow$ nicht konvergent
- in Abhängigkeit von den Startwerten erhalten wir (andere) falsche Schockgeschwindigkeiten.

$\rightarrow$ Naive Verfahren können "konvergente" falsche Lösungen liefern, die nicht einmal schwache Lösungen sind.

4.6.5. Godunov-Verfahren für die Burgersgleichung

Das explizite Eulerverfahren führt mit der FV-Diskretisierung zu

$$u_i^{k+1} = u_i^k + \frac{\tau}{h}\left(\mathcal{F}(u_i^k, u_{i+1}^k) - \mathcal{F}(u_{i-1}^k, u_i^k)\right)$$

mit der Flussfunktion

$$\mathcal{F}(u_i^k, u_i^{k+1}) = \begin{cases} f(u_i^k), & s = f'(u_{i+\frac{1}{2}}^k) \geq 0 \iff \frac{f(u_i^k) - f(u_{i+1}^k)}{u_i^k - u_{i+1}^k} \geq 0 \\ f(u_{i+1}^k), & \text{sonst.} \end{cases}$$

Anmerkung
Im linearen Fall $f(u) = au$ gilt

$$s = \frac{au_i^k - au_{i+1}^k}{u_i^k - u_{i+1}^k} = a,$$

für allgemeinere Anfangsbedingungen wird die Fallunterscheidung komplizierter. Für das Verfahren gilt

- liefert korrekte Schockgeschwindigkeit
- scharfe Front für $\tau = 2h$.

Anhang A.

Grundlagen der Programmierung

Ziel ist es, Aufgaben vom Computer ausführen zu lassen. In Form von Algorithmen lassen sich Aufgaben beschreiben, diese werden als Computerprogramme umgesetzt.

Definition A.1 Algorithmus: Ein *Algorithmus* ist eine Rechenvorschrift zur Lösung eines Problems. Solch ein Problem hat i. A. unendlich viele Instanzen. Die Probleme können Entscheidungs- oder Berechnungsprobleme sein. Zu den Anforderungen gehört:

Finitheit Die Beschreibung des Algorithmus ist endlich.

Determiniertheit Das Ergebnis hängt ausschließlich von der Eingabe ab, erneute Anwendung liefert die gleichen Ergebnisse.

Effektivität Die Ausführung des Algorithmus besteht aus einer Folge von elementaren Schritten, die effektiv (=tatsächlich) ausführbar sind.

Terminiertheit Die Ausführung endet nach endlich vielen Schritten (für jede endliche Eingabe).

Beispiel A.1 Integration mit Hilfe der Trapezregel: Sei $f : \mathbb{R} \to \mathbb{R}$, $f \in L^2(\mathbb{R})$, sowie $a, b \in \mathbb{R}$, $a < b$. Es liefert $T : L^2(\mathbb{R}) \to \mathbb{R}$ eine Näherung

des Integrals $\int_a^b f(x)dx \approx T(a,b,f,n)$:

$$\begin{aligned} T(a,b,f,n) &= \sum_{i=0}^{n-1} \frac{h}{2}\left(f(a+(i+1)h) - f(a+ih)\right) \\ &= h\left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{i=1}^{n-1} f(a+ih)\right] \end{aligned}$$

mit der Schrittweite $h = \frac{b-a}{n}$.

Zwei Bestandteile:

1. Definition der Variablen
2. Anweisungen, bzw. Algorithmus

Bestandteile der Programmierung:

1. Variablen deklarieren
2. Anweisungen

Algorithmus A.1: Integration mit Trapezregel

```
trapezoidal(a,b,f,n)
    h ⟵ (b-a)/n
    x ⟵ a
    s ⟵ 0
    for (i = 1,...,n-1)
        x ⟵ x + h
        s ⟵ s + f(x)
    end for
    s ⟵ s + 0.5(f(a) + f(b))
    s ⟵ hs
    return s
```

A.1. Datentypen & Typisierung

Die Typisierung dient der möglichst genauen Beschreibung von Wertebereichen.

Beispiel: $x \in \mathbb{R}^2$, $||x|| \leq 1$ beschreibt alle Punkte des Einheitskreises.

Durch die Typisierung soll sichergestellt werden, dass auf Inhalten von Variablen keine Operationen ausgeführt werden, die syntaktisch oder semantisch falsch sind. Ein Typsystem wird durch folgende Bestandteile gebildet:

- die Typen selbst
- Regeln um Programmelemente (z.B. Variablen) einem bestimmten Typ zuzuordnen
- Regeln um Werte eines Ausdrucks einem Typ zuzuordnen
- Regeln um Typkorrektheit bei Zuordnungen zu prüfen

A.1.1. Klassifizierung von Typsystemen

a) dynamische / statische Typisierung: Der Typ einer Variable lässt sich zur Laufzeit ändern (dynamisch) oder wird schon während der Kompilierung festgelegt (statisch).

Beispiel:

```
a = 2      → a ist integer
b = ''z''  → b ist string
b = a      → b ist integer (dynamisch)
                ↯          (statisch), Alternative: b = str(a)
```

b) implizite / explizite Typisierung: Der Typ einer Variablen wird explizit festgelegt oder automatisch (implizit) festgelegt.

Beispiel:

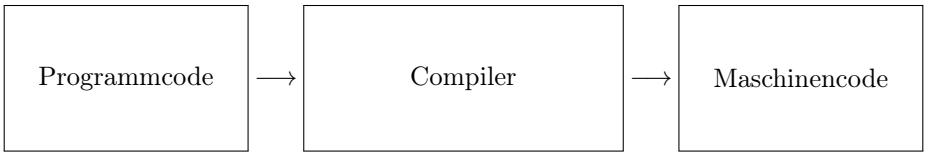
```
        a = 2      → implizit ist a ein integer
float b = 3.0      → explizite Typdeklaration
```

c) schwache / starke Typisierung: Operationen dürfen nur zwischen kompatiblen Typen durchgeführt werden (stark).

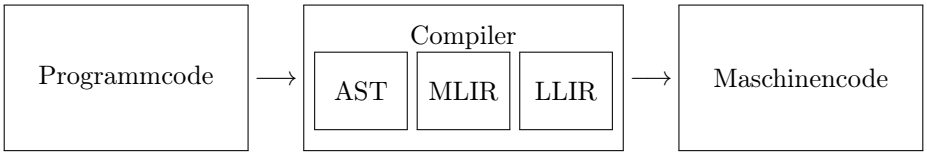
Beispiel: `a = 2`
`b = ''x''`
`c = b + a` → neuer String "`x2`" (schwach)
↯ (stark), Alternative: `c = b + str(a)`

A.2. Arten der Programmausführung (Programmgeschwindigkeit)

Klassischer Ansatz:



Der Compiler geht durch Zwischenstufen, welche unterschiedliche Optimierungen erlauben.



AST: *(abstract syntax tree)*

Optimierungen:

- direktes Auswerten statischer Ausdrücke

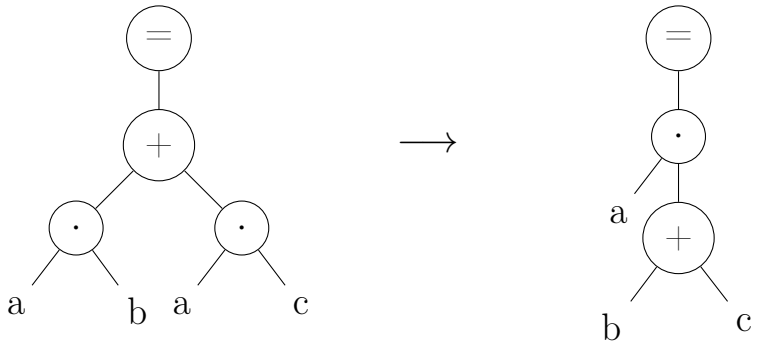
  Beispiel: `5 · 4 = 20`

- Entfernen von "totem" Code

  Beispiel: `if(1==2)` → eliminieren

- "einfache" Umsortierung

  Beispiel: `x = a·b + a·c` → `x = a(b+c)`



MLIR: *(medium level instruction representation)*

Optimierungen:

- verwendete Variablen werden analysiert
- Subausdrücke werden zusammengefasst

  Beispiel: `int j = (ab + ab)` → `int _temp = ab`
  `int j = _tmp + _tmp`

LLIR: *(low level instruction representation)*

Optimierungen:

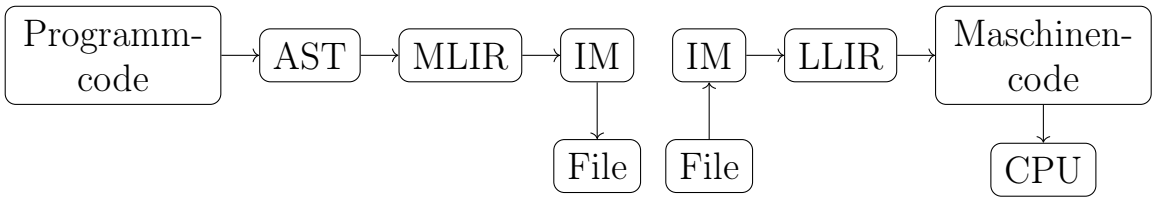
Beispiel: `MULT REG1, 2` → `SHIFTLEFT REG1`
`x*2` → `x<<1`

Intermediate Language

Eine Alternative zum vollstandigen Compilieren ist die Verwendung von eines „Intermediate Language" (IM). Dieser Ansatz wird beispielsweise bei den Sprachen Java der Python verwendet. Der IM Ansatz erlaubt es einige der Optimierungen bereits beim vorkompilieren durchzuführen und beschleunigt somit die Ausführung der Programms.

Vollständig kompiliert:

Mit Intermediate Language:



Das andere Extrem sind vollständig interpretierte Sprachen bei denen alle Schritte erst zur Laufzeit durchgeführt werden.

A.2.1. Vergleich C++ / Matlab / python

	C++	**Matlab**	**python**
Typisierung	stark	schwach	schwach
	explizit	implizit	implizit
	(implizit via `auto`)		(unterstützt auch explizit)
	statisch	dynamisch	dynamisch
Ausführung	voll kompiliert	voll dynamisch	dynamisch + IR

Tabelle A.1.: Vergleich der Typisierung und der Ausführung von C++, Matlab und python.

Algorithmus A.2: Zusammengestzte Trapezregel, Implementierung in C++

```
using fnkt = std::function<double(double)>;

double Trapezoidal(double a, double b, fnkt f, int n)
{
    double h = (b-a)/double(n);
    double x = a;
    double s = 0;
    for(int i=1; i<=n-1; i++){
        x = x+h;
        s = s+f(x);
    }
    s = 0.5*(f(a)+f(b)) + s;
    return h*s;
}
```

Algorithmus A.3: Zusammengestzte Trapezregel, Implementierung in Matlab

```
function r=Trapezoidal(a,b,f,n)
    f = fcnchk(f);
    h = (b-a)/n;
    x = a;
    s = 0;
    for(i=1:n-1)
        x = x+h;
        s = s+feval(f,x);
    end
    s = 0.5*(feval(f,a)+feval(f,b)) + s;
    r = h*s;
end
```

Algorithmus A.4: Zusammengestzte Trapezregel, Implementierung in python

```
def Trapezoidal(a,b,f,n):
    h = (b-a)/float(n)
    x = a
    s = 0
    for i in range(1,n):
        x = x+h
        s = s+f(x)
    end
    s = 0.5*(f(a)+f,(b)) + s
    s = h*s
    return s
end
```

Anhang B.

Genauigkeit, Fehlerquellen

B.1. Überblick

$$\underbrace{\text{Beobachtung}}_{\substack{\text{Messfehler,}\\ \text{z.B. Parameter}}} \longrightarrow \underbrace{\text{Modell} \longrightarrow \text{math. Modell}}_{\substack{\text{Modellfehler,}\\ \text{z.B. Modellannahmen, Vereinfachungen}}} \longrightarrow \underbrace{\text{Simulation}}_{\substack{\text{Diskretisierungsfehler,}\\ \text{Rundungsfehler}}}$$

Ziel ist es, Fehler auszubalancieren.

B.2. Datenrepräsentation & Arithmetrik

Computer (und Programmiersprachen) kennen elementare Datentypen zur Repräsentation von Zahlen:

$\mathbb{N}_0$	`unsigned int`	exakt, aber eingeschränkter
$\mathbb{Z}$	`int`	Wertebereich
$\mathbb{R}$	`float, double`	auch innerhalb des Wertebereichs oft
$\mathbb{C}$	`std::complex<double>`	nur näherungsweise dargestellt

Definition B.1 normierte Fließkommazahl: $\mathbb{F}(\beta, r, s) \subset \mathbb{R}$ beteht aus den Zahlen mit folgenden Eigenschaften:

- $\forall x \in \mathbb{F}(\beta, r, s)$ gilt $x = m\beta^e$ mit $m = \pm \sum_{i=1}^{r} m_i \beta^{-i}$, $e = \pm \sum_{j=0}^{s} e_j \beta^{+i}$. Dabei heißt m Mantisse, e ist der Exponent.

- $\forall x \in \mathbb{F}(\beta, r, s)$ gilt $x = 0 \vee m_1 \neq 0$ (Normierung), d. h. $|x| = 0 \vee \beta^{-1} \leq |m| < 1$

Beispiel:

- $\mathbb{F}(10, 3, 1)$ besteht aus Zahlen der Form

$$x = \pm(m_1 \cdot 0.1 + m_2 \cdot 0.01 + m_3 \cdot 0.001) \cdot 10^{\pm e_0}$$

$(m_1 \neq 0 \vee m_1 = m_2 = m_3 = 0)$, z.B. $0.999 \cdot 10^1$, $0.123 \cdot 10^{-1}$, 0.

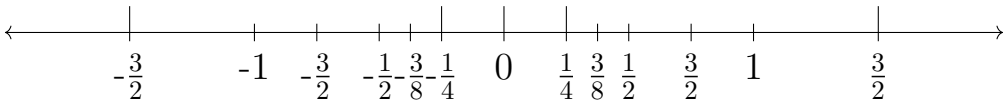
  - $0.014 \in \mathbb{F}(10, 3, 1)$, da $0.014 = 0.14 \cdot 10^{-1}$

  - $0.000000000014 \notin \mathbb{F}(10, 3, 1)$, da $0.000000000014 = 0.14 \cdot 10^{-10}$

- $\mathbb{F}(2, 2, 1)$ besteht aus Zahlen der Form $(x = 0 \vee m_1 \neq 0)$

$$x = \pm\left(m_1 \cdot \frac{1}{2} + m_2 \cdot \frac{1}{4}\right) \cdot 2^{\pm e_0}.$$

Es gilt also $\mathbb{F}(2, 2, 1) = \left\{-\frac{3}{2}, -1, -\frac{3}{4}, -\frac{1}{2}, -\frac{3}{8}, , -\frac{1}{4}, 0, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{3}{4}, 1, \frac{3}{2}\right\}$, bzw. graphisch:



Definition B.2 Runden: Gegeben $x, y \in \mathbb{R}$, sollen diese in Fließkommazahlen umgewandelt werden. Das *Runden* beschreibt eine Abbildung

$$\mathrm{rd} : D \to \mathbb{F}(\beta, r, s),$$

wobei $D = [X_-, x_-] \cup \{0\} \cup [x_+, X_+] \subset \mathbb{R}$ der Bereich der darstellbaren Zahlen $X_{+/-}$ der größten / kleinsten darstellbaren Zahl in $\mathbb{F}$ und $x_{+/-}$ der kleinsten positiven / größten negativen Zahl ist. Die Abbildung rd erfülle die Bedingung

$$|x - \mathrm{rd}(x)| \leq \min_{y \in \mathbb{F}} |x - y|, \quad \forall x \in D$$

Anmerkung
Eine Schwierigkeit bei Fließkommazahlen ist das Problem der Auslöschung. Durch diesen Effekt können beliebig große Fehler entstehen. Wir illustrieren das an einem Beispiel.

Beispiel B.1 Auslöschung bei gerundeter Subtraktion: Es seien $x, y \in \mathbb{F}$, $x \neq y$. Die Operation $\ominus : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ sei exakt gerundet, d. h. $x \ominus y = \mathrm{rd}(x - y)$. Dann gilt für den relativen Fehler

$$\frac{(x \ominus y) - (x - y)}{(x - y)} = \frac{\mathrm{rd}(x - y) - (x - y)}{(x - y)} \leq \epsilon.$$

Betrachten wir $x, y \in \mathbb{F}$ als gerundete Eingaben von $\hat{x}, \hat{y} \in \mathbb{R}$. Bezüglich des exakten Ergebnisses gilt dann:

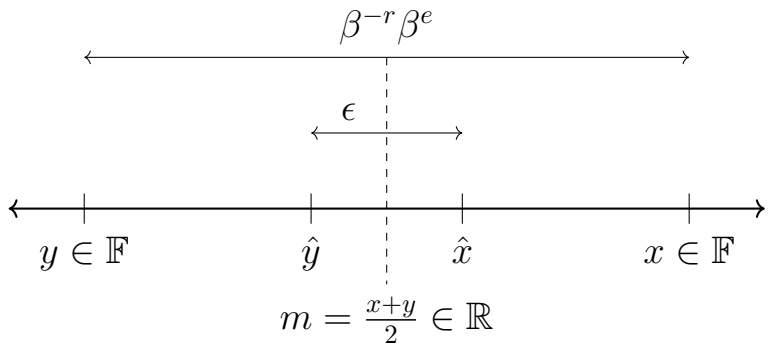
$$\frac{(x \ominus y) - (\hat{x} - \hat{y})}{(\hat{x} - \hat{y})} \sim \frac{1}{\epsilon} \quad \text{für spezielle } |\hat{x} - \hat{y}| = \epsilon$$

Beweis. Wir betrachten $\mathbb{F}(\beta, r, s)$. Wähle

$$x - y = \beta^{-r}\beta^{e}, \quad m = \frac{x + y}{2}, \quad \hat{y} = m - \frac{\epsilon}{2}, \quad \hat{x} = m + \frac{\epsilon}{2}.$$

Dann gilt:

$$\frac{(x \ominus y) - (\hat{x} - \hat{y})}{(\hat{x} - \hat{y})} = \frac{\beta^{-r}\beta^{e} - \epsilon}{\epsilon} = \frac{\beta^{-r}\beta^{e}}{\epsilon} - 1.$$



□

Beispiel: Sei $\mathbb{F} = \mathbb{F}(10, 4, 1)$.

$$\begin{array}{rcl@{\quad\longrightarrow\quad}rcl} \hat{x} & = & 0,11258762 \cdot 10^{2} & x = \mathrm{rd}(\hat{x}) & = & 0,1126 \cdot 10^{2} \\ \hat{y} & = & 0,11244891 \cdot 10^{2} & y = \mathrm{rd}(\hat{y}) & = & 0,1124 \cdot 10^{2} \\ \hline \hat{x} - \hat{y} & = & 0,13871 \cdot 10^{-1} & x - y & = & 0,200 \cdot 10^{-1}, \end{array}$$

und damit

$$\frac{0,2 \cdot 10^{-1} - 0,13871 \cdot 10^{-1}}{0,13871 \cdot 10^{-1}} \approx 0,44 \approx 883\epsilon$$

bei $\epsilon = 0.0005$. Der Fehler in der Subtraktion bei gerundeten Eingaben kann also beliebig groß werden.

B.3. Stabilität & Kondition

Definition B.3 Konditionsanalyse: Bei der *Konditionsanalyse* wird die Sensitivität einer Operation $y = f(x)$ bezüglich Störungen der Eingabedaten untersucht. Sei $f : \mathbb{R}^n \to \mathbb{R}^n$ eine zweimal stetig differenzierbare Abbildung, so gilt die Taylorentwicklung

$$f_i(x + \Delta x) = f_i(x) + \sum_{j=1}^{m} \frac{\partial f_i(x)}{\partial x_j} \Delta x_j + \mathcal{O}(|\Delta x|^2) \quad i = 1, ..., n.$$

Es ergibt sich der relative Fehler

$$\frac{\Delta y_i}{y_i} = \frac{f_i(x + \Delta x) - f_i(x)}{f_i(x)} = \sum_{j=1}^{m} \frac{\partial f_i(x)}{\partial x_j} \frac{\Delta x_j}{f_i(x)}$$

$$= \sum_{j=1}^{m} \underbrace{\frac{\partial f_i(x)}{\partial x_j} \frac{x_j}{f_i(x)}}_{=:\kappa_{ij}(x)} \underbrace{\frac{\Delta x_j}{x_j}}_{\substack{\text{rel. Fehler} \\ \text{von} x_j}} .$$

Definition B.4 Kondition: Die Zahlen κ_{ij} heißen *Konditionszahlen*. Eine Operation $y = f(x)$ heißt schlecht konditioniert, wenn ein $|\kappa_{ij}| \gg 1$ ist, andernfalls gut konditioniert.
Ist $|\kappa_{ij}| < 1$ werden Fehler gedämpft, bei $|\kappa_{ij}| > 1$ verstärkt.

Beispiel: Betrachte $y = f(x_1, x_2) = x_1^2 - x_2^2$.

$$\frac{\Delta y}{y} = 2x_1 \cdot \frac{x_1}{x_1^2 - x_2^2} \cdot \frac{\Delta x_1}{x_1} + (-2x_2) \cdot \frac{x_2}{x_1^2 - x_2^2} \cdot \frac{\Delta x_2}{x_2}$$

$$= \underbrace{\frac{2}{1 - \left(\frac{x_2}{x_1}\right)^2}}_{=:\kappa_1} \cdot \frac{\Delta x_1}{x_1} + \underbrace{\frac{2}{1 - \left(\frac{x_1}{x_2}\right)^2}}_{=:\kappa_2} \cdot \frac{\Delta x_2}{x_2}$$

κ_1 und κ_2 sind die Verstärkungsfaktoren.

Definition B.5 Stabilität eines numerischen Verfahrens: Ein Verfahren heißt numerisch *stabil*, wenn die im Laufe der Rechnung akkumulierten Rundungsfehler (Eingabe aus $\mathbb{F}$) den durch die Konditionierung der numerischen Aufgabe unvermeidbaren Problemfehler nicht übersteigen.

Beispiel: Sowohl $(x_1 \odot x_1) \ominus (x_2 \odot x_2)$, als auch die ausmultiplizierte Variante $(x_1 \ominus x_2) \odot (x_1 \oplus x_2)$ sind stabil, denn die Fehlerfortpflanzung hat die Form $\frac{1}{1-\left(\frac{x_1}{x_2}\right)^2}$, bzw. $\frac{1}{1-\left(\frac{x_2}{x_1}\right)^2}$.

Beispiel: Explizites Euler-Verfahren Wir betrachten die Anfangswertaufgabe

$$\begin{aligned} u'(t) &= \lambda u(t), \quad 0 \leq t \leq T \\ u(0) &= u_0 \end{aligned}$$

mit $\lambda \in \mathbb{R}$, $\lambda \leq 0$.Die Lösung ist $u(t) = \exp(\lambda t)$. Das explizite Euler-Verfahren liefert

$$\begin{aligned} u^{k+1} &= u^k + \lambda \Delta t u^k \\ &= (1 + \lambda \Delta t) u^k \\ &= (1 + \lambda \Delta t)^{k+1} u_0. \end{aligned}$$

Für gestörte Anfangswerte $\tilde{u}_0 = u_0 + \Delta u_0$ ergibt sich:

$$\begin{aligned} \tilde{u}^{k+1} &= (1 + \lambda \Delta t)^{k+1} \tilde{u}_0 \\ &= (1 + \lambda \Delta t)^{k+1} (u_0 + \Delta u_0) \\ \Rightarrow \Delta u^{k+1} &= (1 + \lambda \Delta t)^{k+1} \Delta u_0 \end{aligned}$$

Die Fehlerverstärkung der Rundungsfehler ist $|1 + \lambda \Delta t|^k$. Die Kondition der AWA ergibt sich für $\tilde{u}_0$ wie folgt:

$$\begin{aligned} & \tilde{u}(t) - u(t) = \exp(\lambda t)(\tilde{u}_0 - u_0) \\ \iff \quad & \Delta u(t) = \exp(\lambda t) \Delta u_0 \\ \implies \quad & \frac{\Delta u}{u} = \frac{\exp(\lambda t) \Delta u_0}{\exp(\lambda t) u_0} = \kappa \underbrace{\frac{\Delta u_0}{u_0}}_{\substack{\text{rel.} \\ \text{Fehler}}} \end{aligned}$$

mit $\kappa = 1$.
Das Verfahren ist nur stabil für

$$\begin{aligned} & |1 + \lambda\Delta t| \leq 1 \\ \iff \quad & -1 \leq 1 + \lambda\Delta t \leq 1 \\ \iff \quad & 0 \leq \Delta t \leq -\frac{2}{\lambda} = \Delta t_{\max} \end{aligned}$$

also für genügend kleine Schrittweiten.

Anhang C.

Rechnerarchitekturen

C.1. Sequentielle Rechnerarchitekturen

Sequentielle Rechner sind solche, die nur einen Instruktions- und Datenstrom besitzen.

Definition C.1 Von-Neumann-Architektur: (John von Neumann: 1903 - 1957)
Rechnerarchitektur, bei der Programme wie Daten gespeichert werden.

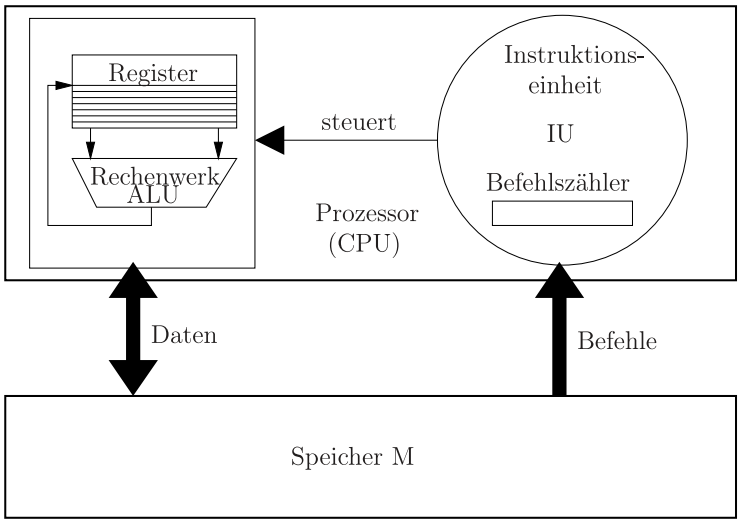


Abbildung C.1.: Von-Neumann Architektur

- Speicher M (*memory*) enthält Daten und Programme
- Instruktionseinheit IU (*instruction unit*)

  - liest nächsten Befehl aus M, an Befehlszählerstelle
  - dekodiert den Befehl
  - steuert das Rechenwerk ALU (*arithmetric logic unit*)
  - Befehlszähler wird erhöht
  - Ergebnis wird in Register geschrieben
  - einzelne Phasen werden in festem Takt abgearbeitet

- Rechenwerk ALU
  - Operanden werden aus Registern gelesen
  - Ergebnisse werden in Register geschrieben (Registerdaten können auch aus dem Speicher geladen & zurück geschrieben werden)
  - führt arithmetrische & logische Operationen aus, mindestens

- Addition	(arithmetrisch)	`integer`
- Negation	(logisch)	`not`
- Konjugation	(logisch)	`and`

  - einzelne Phasen werden mit festem Takt abgearbeitet
  - Datenaustausch zwischen Speicher und Registern durch einen Bus (Leitungsbündel)

C.2. Pipelining

Es gibt 5 Einheiten des MIPS Prozessors:

- Instruction Fetch (IF)
- Instruction Decode (ID)
- Execute (EX)
- Memory access (MEM)
- Write Back (WB)

Bei sequentiellem Ablauf ist immer nur eine Funktionseinheit aktiv:

OP1	IF	ID	EX	MEM	WB									
OP2						IF	ID	EX	MEM	WB				
OP3											IF	...		

Bessere Hardwarenutzung durch Pipelining:

OP1	IF	ID	EX	MEM	WB			
OP2		IF	ID	EX	MEM	WB		
OP3			IF	ID	EX	MEM	WB	

Definition C.2 Pipeline: Zur Beschleunigung werden Maschinenbefehle in Teilaufgaben zerlegt, welche für mehrere Befehle parallel ausgeführt werden. Voraussetzungen für die Anwendbarkeit von Pipelining sind:

- eine Operation OP(x) muss auf viele Operanden $x_1, x_2, ...$ angewandt werden
- die Operation kann in $m > 1$ Teiloperationen zerlegt werden, die in gleicher Zeit bearbeitet werden können
- ein Operand x_i darf nur unter Einschränkung von früheren Operationen abhängen.

Lemma C.1. *Der Zeitbedarf für N Operationen bei einer m-stufigen Pipeline beträgt*

$$T_P(N) = (m + N - 1) \cdot \frac{T_{OP}}{m}$$

Die Beschleunigung beträgt somit

$$S(N) = \frac{T_S(N)}{T_P(N)} = \frac{N \cdot T_{OP}}{(m + N - 1)\frac{T_{OP}}{m}} = \frac{m \cdot N}{m + N - 1}$$

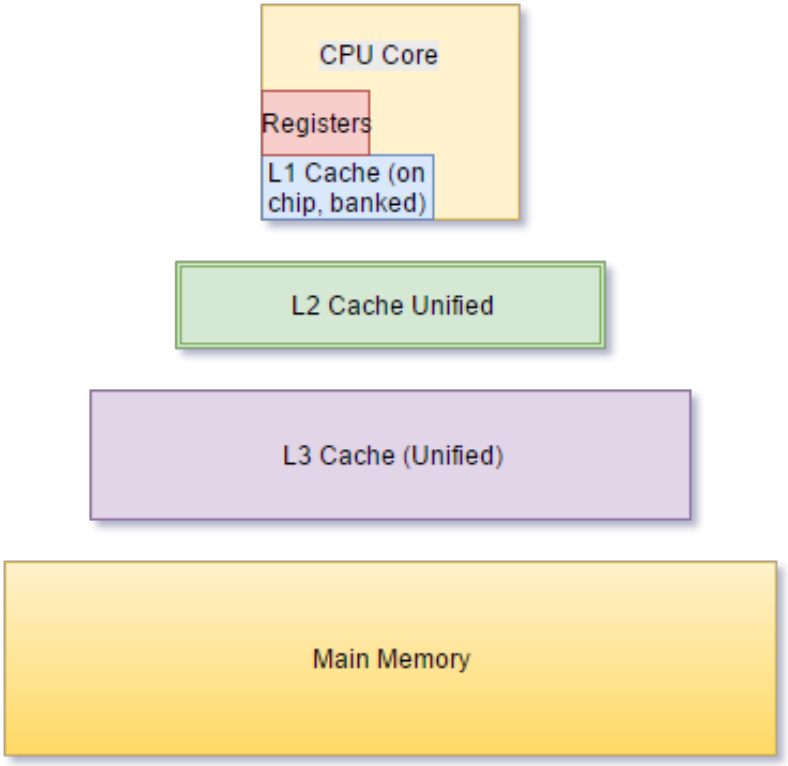
$$\lim_{N \to \infty} S(N) = m$$

Anmerkung
Das diskutierte Pipelining des MIPS-Prozessors wird als Instruction-Pipeline bezeichnet.

C.3. Caches

- moderne Prozessoren sind sehr viel schneller als der Hauptspeicher
- schnellerer Speicher ist sehr teuer, z. T. technisch nicht möglich

→ Lösung: hierarchischer Speicheraufbau mit unterschiedlicher Größe & Geschwindigkeit



Annahmen

- ein Datum, welches einmal verwendet wurde, wird bald wieder verwendet
- Daten, welche gemeinsam gespeichert sind, werden auch gemeinsam verwendet (Lokalitätsprinzip)

Vorgehen

- Prozessor greift auf eine Adresse zu
- ist die Adresse bereits im Cache, wird diese Kopie verwendet

- ansonsten wird der Inhalt des gesamten Blocks, in dem sich die Adresse befindet (Cacheline) kopiert

Der Nutzen des Caches hängt von 4 Eigenschaften ab:

- Platzierungsstrategie
- Identifikationsstrategie
- Ersetzungsstrategie
- Schreibstrategie

Gegeben: Hauptspeicher mit N Blöcken, Cache mit M Blöcken } Blockgröße gleich (=Größe einer Cacheline)

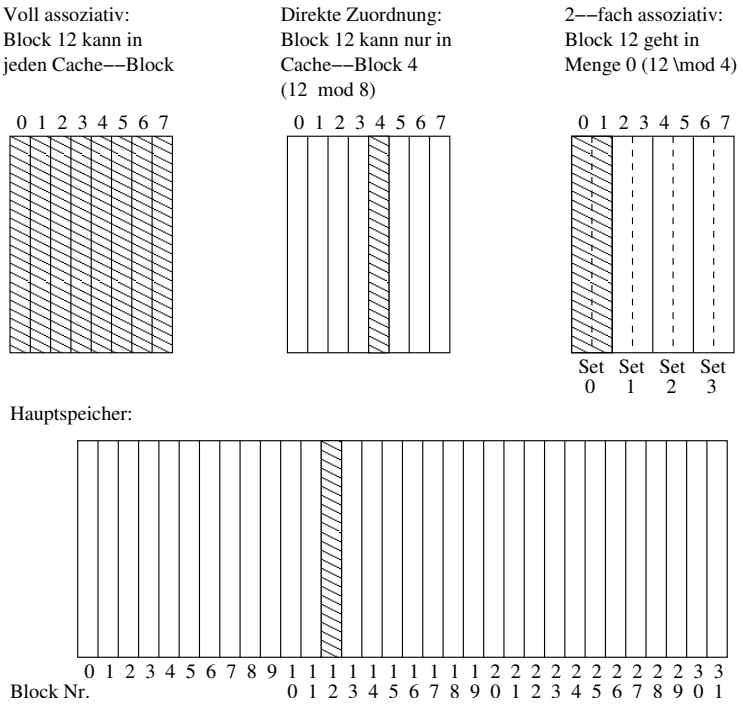


Abbildung C.2.: Unterschieliche Cache-Strategien

a) Platzierung

In welchem Cache-Block wird ein Speicherblock i gespeichert? Es gibt 3 mögliche Strategien:

- *direct mapped* Cache: Block i des Hauptspeichers wird in Block $j = i \mod M$ des Caches geladen

- voll assoziativer Cache: Block i darf in jeden beliebigen Block j des Caches geladen werden
- k-fach assoziativer Cache:
  - M Cacheblöcke werden in $\frac{M}{k}$ Mengen unterteilt
  - Block i wird in einem der Blöcke der Menge $j = i \mod \frac{M}{k}$ gespeichert
  - typische Werte für k: 2-4 für Level-1-Cache, 8-32 für höhere Level

b) Identifizierung

Ist / Wo ist ein bestimmter Block im Cache gespeichert?

- zu jedem Block im Cache wird die *block frame address* gespeichert
- im k-fach assoziativen Cache: k parallele Vergleiche der Adresse
- zusätzlich hat jeder Block eine *valid*-Flag

c) Ersetzung

Welcher alte Eintrag wird überschrieben?

- *direct mapped*: nur 1 Block möglich
- (k-fach)-assoziativ:
  - am sinnvollsten der Wert/Block, der am längsten nicht verwendet wurde (LRU-Cache)
  - praktisch zeigt sich, dass für große Caches eine zufällige Auswahl genauso gut ist

d) Schreiben

- 2 Strategien:
  a) konsistenter Cache und Hauptspeicher (*write through*)
  b) Hauptspeicher wird erst geschrieben, wenn Cache-Block verdrängt wird (*write back*)
- wenn die Adresse nicht im Cache ist, kann a) direkt zurück schreiben, b) muss erst Block laden (*cold write*/*write miss*)

- bei b) ist eine zusätzliche *dirty*-Flag nötig, um anzuzeigen, welche Daten noch nicht geschrieben sind

C.4. Vektorisierung

C.4.1. Klassifizierung nach Flynn (1972)

Betrachtet werden die Anzahl der Instruktions- und der Datenströme. Wir unterscheiden lediglich einfach (1) und mehrfach (>1). Es ergeben sich 4 Klassen von Rechnern:

SISD *single instruction, single data*
⟶ klassischer sequentieller Rechner, z. B. von Neumann

SIMD *single instruction, multiple data*
Eine Instruktionseinheit (IU) steuert viele Rechenwerke. Insbesondere heißt das, dass alle Rechenwerke die gleiche Operation ausführen müssen
⟶ wenig flexibel, aber effizient

MIMD *multiple instruction, multiple data*
viele unabhängige Rechner, bzw. CPUs
⟶ echte Großrechner, Multi-Core CPUs, *shared memory* Systeme

MISD gibt es nicht

C.4.2. SIMD

Ursprünglich gab es spezielle Vektor-Rechner, z.B. die NEC SX Reihe, oder der *Earth-Simulator*.

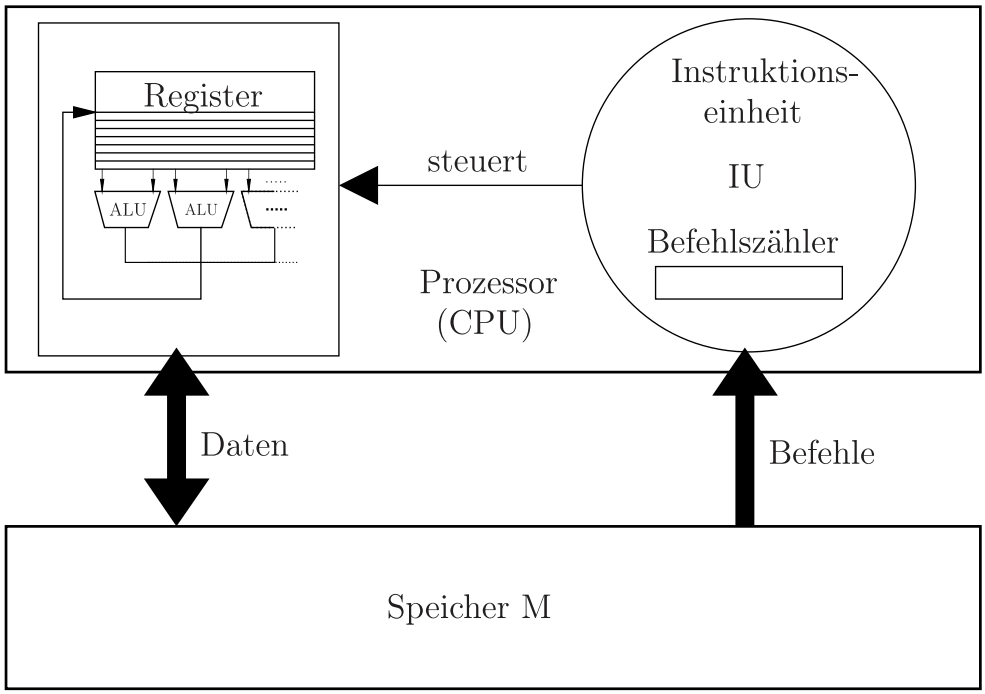


Abbildung C.3.: SIMD Architekturen enthalten mehrere ALUs, welche (single instruction) stets die gleiche Operation ausführen.

Heute finden sich SIMD Einheiten in jedem normalen Computer. Meist wird es als Erweiterung zu klassischen CPUs verwendet:

- Vektoreinheiten der CPU (SSE, AVX, AltiVec)
  - wenige Fließkommadaten(z. B. 4 float bei SSE) werden parallel bearbeitet
  - gesonderte Register
- Grafikkarten
  - sehr breite Vektoreinheiten in wenigen SIMD Einheiten (viele hundert float parallel)

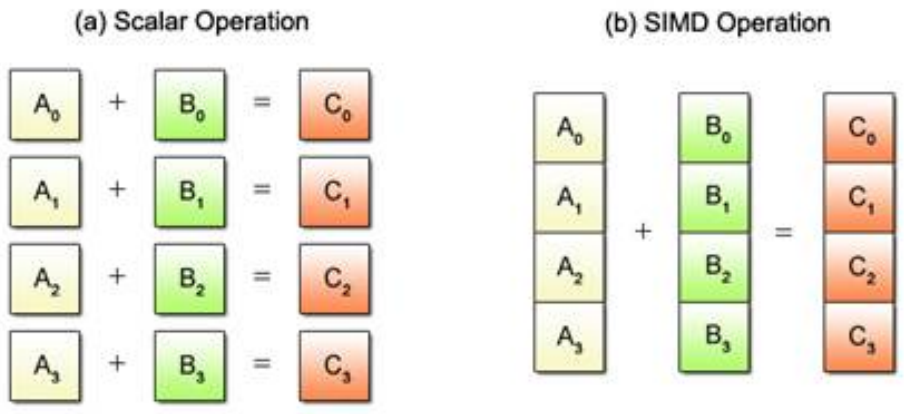


Abbildung C.4.: SIMD Einheiten führen die gleiche arithmetische Operation auf mehreren Datenströmen parallel aus. Wenn die Daten sich entsprechend sortieren lassen, sind somit erhebliche Programmbeschleunigungen möglich (Quelle: `https://www.kernel.org/`).

Der Compiler besitzt einen Auto-Vektorisierer, der die Erweiterungen der CPU unterstützt.
Notwendige Eigenschaften:

a) Vektoren dürfen nicht überlappen

b) große Register laden geht nur dann parallel, wenn die Daten *aligned* sind
→ Startadresse ein Vielfaches der *alignments*, i. d. R. Breite der Register

C.5. Weitere Beschleunigungstechniken

superskalare Systeme

- mehrere Recheneinheiten
- mehrere Befehler pro Takt abarbeiten
- in Verbindung mit Instructionpipelining

→ größere Gefahr von Pipelineproblemen durch Datenabhängigkeiten

out-of-order execution

- soll Abhängigkeitsprobleme bei superskalaren Systemen reduzieren
- einzelne Befehle werden vom Prozessor umsortiert

branch-prediction, speculative execution

- Verzweigungen werden versucht vorherzusagen
- soll die Pipelineausnutzung erhöhen

Beispiel: `for (i=0; i < 10000; i++)` $\longrightarrow$ 9999-mal `true`, 1-mal `false`

- vermuteter Zweig wird ausgerechnet
- bei falscher Vorhersage wurde zuviel berechnet

RISC/CISC, Micro-Ops RISC (*reduced instruction set computer*) als Ersatz von CISC (*complex instruction set computer*) um Pipelines effizient zu nutzen

- alle Befehle in Hardware dekodieren
- ein Befehl pro Takt $\rightarrow$ homogener Befehlssatz
- explizite *load*/*store* Befehle für Speicherzugriff
- viele Regisster & große Caches

Heute oft ein Mittelweg

- nach außen CISC
- Übersetzung der Hardware in μ-Ops
- Berechnung im RISC-Kern

Anhang D.

Paralleles Rechnen

Definition D.1 Paralleles Programm: Ein *paralleles Programm* besteht aus einer Menge interagierender sequentieller Prozesse. Die Ausführung kann auf einem oder vielen Prozessen erfolgen. Die Interaktion erfolgt über gemeinsame Variablen (*shared memory*) oder durch Nachrichtenaustausch (*distributed memory/message passing*).

Beispiel D.1 Ein einfaches Problem: Wir betrachten die Berechnung des Skalarprodukts zweier Vektoren $x, y \in \mathbb{R}^n$ gegeben durch $s = x \cdot y = \sum_{i=0}^{N-1} x_i y_i$. Welche Schritte können bei dieser Berechnung parallelisiert werden?

a) Berechnung der Summanden x_i, y_i kann für alle i gleichzeitig erfolgen.

b) Da die Anzahl der Indizes N größer als die Anzahl der Prozessoren sein darf, weisen wir jedem Prozessor eine Teilmenge $I_p \subseteq \{0, ..., N-1\}$ der Indizes zu. Jeder Prozessor berechnet dann die Teilsumme

$$s_p = \sum_{i \in I_p} x_i y_i.$$

c) Bei der Berechnung der Gesamtsumme aus den Prozessorteilsummen können wir die Assoziativität nutzen und schreiben z. B. für $P = 8$

$$s = \underbrace{\underbrace{\underbrace{s_0 + s_1}_{s_{01}} + \underbrace{s_2 + s_3}_{s_{23}}}_{s_{0123}} + \underbrace{\underbrace{s_4 + s_5}_{s_{45}} + \underbrace{s_6 + s_7}_{s_{67}}}_{s_{4567}}}_{s},$$

d. h. die Summe kann in $3 = \mathrm{ld}8 := \log_2 x$ aufeinander folgenden, jeweils parallelen Schritten berechnet werden.

D.1. Kommunikation über gemeinsame Variablen

→ PR Skript

D.2. Kommunikation über Nachrichtenaustausch

Systeme mit privatem (verteiltem) Speicher benötigen dedizierte Befehle zum Kopieren von Daten aus dem Speicher eines Prozesses in den eines anderen (*send*/*receive*).

D.2.1. Technik

- Nachrichten werden in Pakete fester Länge zerteilt
- Pakete werden von Knoten zu Knoten gesendet
- Paket besteht aus 3 Teilen:

  - Kopf: Statusinformationen der Nachricht, z. B. Ziel
  - Daten
  - Nachspann: z. B. Checksumme

- viele Möglichkeiten des Netzwerks, z. B. Ethernet, Myrinet, Infiniband

D.2.2. Kommunikation

P, N sind nur als globale Konstanten erlaubt, eine Interaktion über gemeinsame Variablen ist nicht möglich.
$\longrightarrow$ Interaktion zwischen Prozessen durch Nachrichten

Syntax

- send (<Prozess>, <Variablen>)
- receive(<Prozess>, <Variablen>)

Diese Operationen dauern bis die Nachricht gesendet und empfangen wurde (blockierend).

Algorithmus D.1: Parallel message passing scalar product

```
const int d, P = 2^d, N; //Konstanten!
process Π [int p ∈ {0,...,P-1}]{
   double x[N/P], y[N/P]; //lokales Ausschneiden der Vektoren
   int i, r, m, k;
   double s = 0, ss;
   for(int i = 0; i < (p + 1) * N/P - p * N/P; i++)
      s = s + x[i] * y[i];
   for(i = 0; i < d; i++)
      m = (p >> i+1) << i+1;
      r = m | 1 << i;
      if(p==r) send (Π_r, s)
      if(p==m){
         receive (Π_r, ss)
         s = s + ss;
      }
   }
}
```

Man beachte, dass die Prozesse über den Nachrichtenaustausch implizit synchronisiert werden. Am Ende des Programms enthält die Variable s des Prozesses 0 das Endergebnis. Sowohl *send* als auch *receive* blockieren, da die Kommunikation abgeschlossen ist.
$\longrightarrow$ Sende- und Empfangsprozesse *src*/ *dest* müssen ein passendes *send*/ *recv*-Paar aufrufen, sonst resultiert dies in einer Verklemmung.

Beispiel D.2 Prozesse, die Daten im Ring schicken:

Π_0:	Π_1 :	Π_2 :
send(Π_1, i)	send(Π_2, i)	send(Π_0, i)
recv(Π_2, i)	recv(Π_0, i)	recv(Π_1, i)

Es ist nicht immer klar, von welchem Prozessor die nächste Nachricht kommt.
$\longrightarrow$ Beispiel: Master-Worker-Pattern

Lösung

`bool sprobe (dest), bool rprobe (src)`

- Funktionen, die testen, ob ein Prozess bereit ist zu empfangen/senden
- `sprobe`, `rprobe` sind selbst nicht blockierend

Die Implementierung von `sprobe` wäre sehr kompliziert $\longrightarrow$ in der Regel steht nur `rprobe` zur Verfügung.

Alternative

`receive_any (who, var`$_0$`,..., var`$_n$ werden von einem beliebigen Prozess empfangen.

D.2.3. Asynchrone Kommunikation

```
msgid asend (dest, val_0, ... ,val_n)
msgid arecv (src, val_0, ... , val_n)
```

- Semantik wie bei `send,recv`
- nicht blockierend
  → Problem: Wann ist ein Befehl abgeschlossen? (besonders beim Empfang)
- `bool success(msgid)`
- asynchrones und synchrones (=blockierendes) Empfangen und Senden können beliebig gemischt werden

D.3. Leistungsanalyse

Wie leistungsfähig ist ein paralleles Programm? Um das zu beurteilen, ist ein Vergleich zwischen dem sequentiellen und dem parallelen Programm nötig. Dabei stellen sich insbesondere zwei Fragen:

- Was ist der Einfluss des Algorithmus?
- Was ist der Einfluss der Hardware?

D.3.1. Leistungsfähigkeit eines Netzwerkes

- Nachrichtenaustausch zwischen 2 Prozessen
- das Übertragen von n Bytes dauert $t_{\text{send}}(n) = t_s + n \cdot t_b$ (linearer Zusammenhang) mit
  t_s: Setup-Zeit (Latenz)
  t_0: Durchsatz ($\frac{1}{t_b}$ heißt Bandbreite).

D.3.2. Leistungsanalyse von Algorithmen

Für das Beispiel des Skalarproduktes gilt:

sequentiell: $T_S(N) = 2Nt_a$, wobei t_a die Zeit für eine arithmetrische Operation (FLOP) ist.

parallel: $T_P(N,P) = 2\frac{N}{P}t_a + \text{ld}P(t_m + t_a)$, wobei t_m die Zeit für das Versenden einer Fließkommazahl ist.

Definition D.2 Speedup: Der *Speedup* ist das Verhältnis von sequentieller und paralleler Laufzeit

$$S(N,P) := \frac{T_S(N)}{T_P(N,P)},$$

hier bedeutet dies konkret

$$S(N,P) = \frac{2Nt_a}{2\frac{N}{P}t_a + \text{ld}(P(t_m + t_a)} = \frac{P}{1 + \frac{P}{N}\text{ld}P\frac{t_m+t_a}{2t_a}}$$
$$\Rightarrow S(N,P) \leq P.$$

Definition D.3 Effizienz: Die *Effizienz* eines Algorithmus ist definiert als

$$E(N,P) = \frac{S(N,P)}{P},$$

hier konkret

$$E(N,P) = \frac{1}{1 + \frac{P}{N}\mathrm{ld}P\frac{t_m+t_a}{2t_a}}$$

$$\overset{S(N,P)\leq P}{\Longrightarrow} E(N,P) \leq 1.$$

Anmerkung

- für festes N: $E(N,P) \geq E(N,P+1)$
- für festes P: $\lim_{N\to\infty} E(N,P) = 1$
- Skalierbarkeit für $N = k \cdot P$, k konstant:

$$E(kP,P) = \frac{1}{1 + \mathrm{ld}\frac{t_m+t_a}{2t_a k}}$$

Literaturverzeichnis

J. Barnes and P. Hut. A hierarchical o (n log n) force-calculation algorithm. *nature*, 324(6096):446–449, 1986.

R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische annalen*, 100(1): 32–74, 1928.

L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

Index