DIPLOMARBEIT

Numerische Simulation von chemischen Reaktionen in Flüssigkeiten

Angefertigt am
Institut für Numerische Simulation

Vorgelegt der
Mathematisch-Naturwissenschaftlichen Fakultät der
Rheinischen Friedrich-Wilhelms-Universität Bonn

September 2013

Von

Katharina Schinagl

geboren am 25.02.1985
in
Rosenheim

Inhaltsverzeichnis

1. Einleitung **1**

2. Grundlagen **5**
2.1. Chemische Reaktionen 5
2.1.1. Oszillierende Reaktionen 5
2.2. Reaktions-Diffusions-Modelle 5
2.2.1. Der Brusselator 6
2.3. Alternative Modelle 7
2.3.1. Das Gray-Scott-Modell 7
2.3.2. Der Oregonator 8
2.3.3. Das Arrhenius-Modell 8
2.4. Die Navier-Stokes-Gleichungen und NaSt3DGPF 9
2.4.1. Das mathematische Modell: die Navier-Stokes-Gleichungen 9
2.4.2. Diskretisierung in NaSt3D 10
2.4.3. Zusammenfassung des Algorithmus 17
2.4.4. Überblick über NaSt3DGPF 18

3. Numerische Umsetzung des Chemiemodells und der Kopplung **19**
3.1. Ortsdiskretisierung des Brusselators 19
3.1.1. Finite Differenzen - Methode 19
3.1.2. Ortsdiskretes Gleichungssystem (2D) 22
3.1.3. Gesplittetes Gleichungssystem 27
3.1.4. Operatorsplitting für das ortsdiskrete Brusselator-System 29
3.2. Zeitdiskretisierung des Brusselators 32
3.3. Implementierung 35
3.4. 2D-Wärmeleitungsgleichung in MATLAB 35
3.4.1. Explizit 36
3.4.2. Implizit 36
3.5. 2D-Brusselator in MATLAB 38
3.5.1. Explizites Verfahren zur Zeitdiskretisierung der Reaktion 38
3.5.2. Implizites Verfahren zur Zeitdiskretisierung der Reaktion 38
3.6. Erweiterung des Brusselators auf 3D 41
3.7. Kopplung des Brusselators an NaSt3DGPF 43
3.7.1. Kopplungsmodell 43

3.7.2. Zeitskalen für Chemie und Strömung 45
3.7.3. CFL-Bedingungen für das Chemiemodell 45
3.7.4. Gekoppelter Algorithmus 48

4. Konvergenzanalysen und Simulationsergebnisse 51
4.1. Vereinfachte Systeme 51
4.2. Konvergenzanalyse des 2D-Brusselators 56
4.2.1. Explizites Verfahren - Zeitanalyse 57
4.2.2. Explizites Verfahren - Ortanalyse 59
4.2.3. Implizites Verfahren - Zeitanalyse 62
4.2.4. Implizites Verfahren - Ortanalyse 65
4.2.5. Gleichzeitige Verfeinerung - explizites Verfahren 67
4.2.6. Gleichzeitige Verfeinerung - implizites Verfahren 70
4.3. Vergleich mit weiteren 2D-Literaturbeispielen 73
4.4. 3D - Simulationsergebnisse 78
4.4.1. 3D: Oszillierende Reaktion 78
4.4.2. 3D: Oszillierende Reaktion mit aktiver Kopplung an das Strömungsfeld 85
4.4.3. 3D: Einfluß eines Rührers in einer oszillierenden Reaktion mit anfangs getrennten Chemikalien 89

5. Zusammenfassung und Ausblick 99
5.1. Zusammenfassung 99
5.2. Ausblick 101

A. Anhänge 103
A.1. Aktive Kopplung - Parameterset II 103
A.2. Aktive Kopplung - Parameterset III 106
A.3. 3D-Reaktion mit getrennten Chemikalien ohne Rührfisch 108
A.4. 3D-Reaktion mit getrennten Chemikalien, Rührfisch nur passiv gekoppelt . 111

1. Einleitung

Motivation

Bei einer chemischen Reaktion handelt es sich um einen Vorgang, bei dem eine oder mehrere chemische Verbindungen in andere umgewandelt werden. Sie sind allgegenwärtig sowohl im täglichen Leben als auch in der Industrie. Einige Beispiele für Gelegenheiten, bei denen chemische Reaktionen auftreten, sind Kochen, Waschvorgänge, die Lebensmittelherstellung, Wasserverschmutzung Verbrennunsgvorgänge, chemische Trennverfahren und viele mehr.

Nicht zu den chemischen Reaktionen zählen physikalische Vorgänge, bei denen sich nur der Aggregatszustand ändert wie z.B. Schmelzen, Verdampfen oder die Diffusion.

Weil viele der Reaktionen - auch von den genannten Beispielen - in Fluiden, also Gasen oder Flüssigkeiten stattfinden und Vorgänge wie das Lösen von Chemikalien oder die Bewegung der Substanzen (z.B. durch Rühren) dabei eine Rolle spielen, ist eine Verbindung zur Fluiddynamik naheliegend.

Da ein realler Versuchsaufbau zum Durchführen von Experimenten häufig sehr aufwändig, kostspielig und gefährlich sein kann, sind hier natürlich numerische Simulationen von Interesse. Diese würden es ermöglichen wesentlich mehr und größere Experimente durchzuführen, als es bei realen Versuchen allein der Fall wäre. So ließen sich viele Erkenntnisse gewinnen, die für den ingeneurstechnischen Einsatz bedeutsam wären. Eine wichtige mögliche Anwendung ist etwa das Design und die Optimierung von chemischen Reaktoren.

Problemstellungen

Zur Umsetzung der Simulation von chemischen Reaktionen stellen sich einige Probleme und Fragen.
Zunächst ist zu beachten, dass der verwendete Strömungslöser, NaSt3DGPF, inkompressible Strömungen behandelt, das heißt solche, bei denen sich die Dichte nicht ändert. Da dies bei vielen chemischen Reaktionen aber der Fall ist, müssen wir uns auf solche einschränken, wo die Dichteänderung vernachlässigbar ist. Daher gehen wir im Folgenden

davon aus, dass die Reaktionen verdünnt in einer Lösung stattfinden. Dies schließt zum Beispiel die Betrachtung von Verbrennungsreaktionen aus.

Weiterhin muss berücksichtigt werden, dass zwischen Strömung, Transport und Reaktion Kopplungen und Rückkopplungen stattfinden. Hier wird die Umsetzung der numerischen Simulation sehr komplex.

Auch hängen die Konzentrationen der verschiedenen an der Reaktion beteiligten chemischen Spezies wechselseitig voneinander ab. Insbesondere soll darauf hingeweisen werden, dass an einer Reaktion meist viele Spezies beteiligt sind - mehr als man zunächst annehmen würde, da es viele Zwischenspezies geben kann. Dadurch haben wir es im Grunde auch mit einer hochdimensionalen Problemstellung zu tun, wofür man auch eine Vielzahl entsprechender Ansätze verwenden könnte. In der Literatur findet sich viel hierzu. In dieser Arbeit wollen wir uns jedoch vordergründig mit der Kopplung einer Simulationsmöglichkeit für chemische Reaktionen an den Strömungslöser beschäftigen und beschränken uns zu diesem Zweck auf vereinfachte Chemie-Modelle mit wenigen Spezies.

Grundsätzlich hat man es aber, auch wenn nur wenige Spezies vorhanden sind, bei chemischen Reaktionen mit nichtlinearen Vorgängen zu tun, was die numerische Simulation erschwert. Diese Tatsache begründet sich darin, dass sich der Reaktionsablauf - abhängig von der Aktivierungsenergie - auf einer sehr kleinen Zeitskala stark verändern kann.

Lösungsansätze

Wir haben uns verschiedene Modelle angesehen, die zur Simulation von chemischen Reaktionen eingesetzt werden können.
Davon wurde schließlich das Brusselator-Modell implementiert. Um eine größere numersiche Stabilität zu erhalten, wurde neben einem expliziten Lösungsverfahren auch ein implizites Verfahren programmiert.
Das Modell wurde in einer dreidimensionalen Version an den Strömungscode NaSt3DGPF gekoppelt. Die Kopplung wurde sowohl in einer passiven Variante als auch in einer aktiven Variante durchgeführt. Bei der passiven Variante werden die reagierenden Spezies lediglich mit der Strömung transportiert. Für die aktive Kopplung -bei der zusätzlich zum Transport auch eine Rückwirkung der Reaktion auf das Strömungsfeld stattfindet- wurde ein neues Modell entwickelt und getestet.
Zusätzlich setzen wir die Fluid-Struktur-Interaktion von NaSt3DGPF ein, um im Rechner einen magnetischen Rührer (auch: “Rührfisch”) zu erzeugen. Derartige Apparate werden bei realen Experimenten häufig zum Vermischen der Stoffe eingesetzt.

Eigene Beiträge

Wir fassen noch einmal die in dieser Arbeit geleisteten eigenen Beiträge zusammen:

- Implementierung von expliziten und impliziten Verfahren für den Brusselator in 2D und 3D
- Konvergenzanalyse des 2D-Codes
- Kopplung in 3D an den Strömungscode NaSt3DGPF
- insbesondere aktive Rückkopplung der Reaktion an das Strömungsfeld
- erfolgreiche Simulation von oszillierenden Reaktionen in 3D
- Simulation einer durch einen "Rührfisch" angetriebenen chemischen Reaktion

Zu dieser Arbeit

Nun wollen wir noch einen kurzen Überblick geben, wie diese Arbeit strukturiert ist.

Kapitel 2 beginnt mit einem kurzen Überblick über die theoretischen Grundlagen, die zum Verstehen verschiedener Modelle für chemische Reaktionen notwendig sind und stellt einige davon vor. Weiterhin befassen wir uns mit den physikalischen/mathematischen Gleichungen, die die Strömungsvorgänge beschreiben, und dem Strömungslöser NaSt3DGPF.

Kapitel 3 erläutert die numerische Umsetzung des Brusselator-Modells und die Kopplung dessen an den Strömungscode.

Kapitel 4 stellt anschliessend die mit unserer Implementierung erhaltenen Simulationsergebnisse sowie Konvergenzbetrachtungen und Fehleranalysen der Algorithmen vor.

Kapitel 5 schließlich beendet diese Arbeit mit einer Zusammenfassung der Ergebnisse und einem Ausblick auf mögliche weitere Entwicklungen und noch offene Fragestellungen.

2. Grundlagen

2.1. Chemische Reaktionen

Eine chemische Reaktion besteht darin, dass mehrere Stoffe umgewandelt werden. Es gibt viele sehr unterschiedliche Arten der Reaktionen. Generell lassen sie sich schreiben als

$$A + B \to C \tag{2.1}$$

Derartige Reaktionsschemata können allerdings sehr komplex werden. Häufig gibt es, bei der Umwandlung etwa von A zu B noch viele verschiedene Zwischenreaktionen und -spezies.
Meist lässt sich eine Reaktion jedoch durch einige wenige "Bausteine" von Elementarreaktionen beschreiben (vgl. [DB08]). Daraus lassen sich Modelle von Differentialgleichungen ableiten. Diese können wir numerisch behandeln.

2.1.1. Oszillierende Reaktionen

Ein Sonderfall von chemischen Reaktionen ist die Klasse der *oszillierenden Reaktionen*. Diese wurden zufällig experimentell entdeckt. Da sie häufig besonders komplex sind, sind wenige dieser Reaktionen genau erforscht. Ein gutes Beispiel, eine der am meisten und besten erforschten oszillierenden Reaktionen ist die *Belousov-Zhabotinsky-Reaktion* (BZR). Sie ist der Prototyp der dem in dieser Arbeit betrachteten Brusselator-Modell zugrundeliegenden Reaktionen.

2.2. Reaktions-Diffusions-Modelle

Sei $\Omega \subset \mathbb{R}^3$ ein Gebiet.
Wir betrachten nun *Reaktions-Diffusions-Modelle* . Diese beschreiben die Konzentrationsänderungs einer oder mehrerer Größen unter dem Einfluss von Reaktion und Diffusion.

$$\partial_t C(t,x) = \underbrace{D_C \cdot \Delta C(t,x)}_{Diffusion} + \underbrace{Q\left(C(t,x),\ldots\right)}_{Reaktion} \tag{2.2}$$

wobei $C : \Omega \times [0, t_{end}] \to \mathbb{R}$ die Konzentration einer Spezies darstellt.
Unter *Diffusion* verstehen wir den physikalischen Prozess des Vermischens zweier oder mehrerer Stoffe, die miteinander in Berührung sind, aufgrund ihrer Eigenbewegung (im Gegensatz zu einer Mischung durch externe Einflüsse wie Strömung oder Mischapparaturen). Die *Diffusionskonstante* (oder auch *Diffusionskoeffizient*) D_C hängt unter anderem von der Temperatur des Systems ab und davon um welchen Stoff es sich handelt.
Die Diffusion führt zu einem Ausgleich der Konzentrationen der vorhandenen Spezies. Damit wirkt sie der Reaktion entgegen, die im Allgemeinen eher für Konzentrationsänderungen sorgt.
Ein solches Modell kann eine oder auch mehrere Spezies beinhalten. Dabei kann es sich um unterschiedliche Dinge handeln - in unserem Zusammenhang gehen wir von chemischen Spezies aus und werden auch im Folgenden nur davon sprechen. Es sei aber erwähnt dass in anderen Zusammenhängen beziehungsweise bei anderen Reaktions-Diffusions-Modellen als den von uns betrachteten, die modellierten Größen etwas völlig anderes darstellen können. Bei dem *FitzHugh-Nagumo-Modell* etwa wird ein Neuron beschrieben.

2.2.1. Der Brusselator

Sei wieder $\Omega \subset \mathbb{R}^2$ oder $\Omega \subset \mathbb{R}^3$. Wir betrachten ein bekanntes Modell für zwei Spezies, den Brusselator [LN71]:

$$\partial t C_1(\vec{x}, t) = D_{C_1} \Delta C_1(\vec{x}, t) + k_1 A - k_2 B C_1(\vec{x}, t) + k_3 C_1(\vec{x}, t)^2 C_2(\vec{x}, t) - k_4 C_1(\vec{x}, t) \quad \forall x \in \Omega, \quad t \in (0, t_{end}] \tag{2.3}$$

$$\partial_t C_2(\vec{x}, t) = D_{C_2} \Delta C_2(\vec{x}, t) + k_2 B C_1(\vec{x}, t) - k_3 C_1(\vec{x}, t)^2 C_2(\vec{x}, t) \quad \forall x \in \Omega, \quad t \in (0, t_{end}] \tag{2.4}$$

mit den Rand- sowie Anfangsbedingungen:

$$\begin{aligned} &C_1(\vec{x}, t) = f^{\mathcal{D}}(\vec{x}, t) \text{ und } C_2(\vec{x}, t) = g^{\mathcal{D}}(\vec{x}, t) \qquad \forall x \in \delta\Omega, \quad t \in (0, t_{end}] \\ &\quad \text{oder} \\ &\frac{\partial C_1}{\partial n}(\vec{x}, t) = f^{\mathcal{N}}(\vec{x}, t) \text{ und } \frac{\partial C_2}{\partial n}(\vec{x}, t) = g^{\mathcal{N}}(\vec{x}, t) \qquad \forall x \in \delta\Omega, \quad t \in (0, t_{end}], \end{aligned} \tag{2.5}$$

wobei $f^{\mathcal{D}}$ und $f^{\mathcal{N}}$ Funktionen sind, die Dirichlet- oder Neumann-Randbedingungen beschreiben.

$$\begin{aligned} C_1(\vec{x}, t) &= C_1(\vec{x}) \qquad \forall x \in \overline{\Omega} \\ C_2(\vec{x}, t) &= C_2(\vec{x}) \qquad \forall x \in \overline{\Omega} \end{aligned} \tag{2.6}$$

Es handelt sich um ein System zweier nichtlinearer, gewöhnlicher Differentialgleichungen[1] erster Ordnung.

[1] auch: *ODE* von englisch: ordinary differential equation

Das Modell basiert auf dem vereinfachten Reationsschema (vgl. etwa [HNW00]):

$$A \xrightarrow{k_1} C_1$$

$$B + C_1 \xrightarrow{k_2} C_2 + D$$

$$2C_1 + C_2 \xrightarrow{k_3} 3C_1$$

$$C_1 \xrightarrow{k_4} E$$

Das Schema beschreibt zwei Spezies, C_1 und C_2, die miteinander in einer oszillierenden Reaktion stehen. Vorbild für dieses vereinfachte Modell sind Reaktionen wie etwa die Belousov-Zhabotinsky-Reaktion .
Dabei handelt es sich um eine der bekanntesten und am besten erforschten oszillierenden Reaktionen. Dennoch ist die genaue Reaktionskinetik schwierig zu bestimmen, da es, wie bei den meisten Reaktionen, viele Prozesse gibt, die gleichzeitig stattfinden, und Zwischenprodukte, die sich bilden. Ein weit verbreitetes und anerkanntes Modell für den zugrundeliegenden Mechanismus ist der *FKN-Mechanismus* - vorgestellt von Field, Körös und Noyes in [FKN72]. Auf diesem basiert der Brusselator sowie auch der weiter unten kurz vorgestellte Oregonator.
Für einen Ausblick auf die über dieses Modell hinausgehende detaillierte Erforschung der vielen kinetischen Abläufe bei der Belousov-Zhabotinsky-Reaktion sei etwa auf [GTF90] verwiesen, wo ein Mechanismus bestehend aus 80 Reaktionen mit 26 Spezies vorgeschlagen wird.

Der Brusselator hat den kritischen Punkt $\left(A, \frac{B}{A}\right)$.
Eine Analyse der Stabilitätsregionen findet sich in [SMM13].

Leider gibt es bislang wenig numerische Betrachtungen zum Brusselator-Modell in der wissenschaftlichen Literatur.

2.3. Alternative Modelle

Wir wollen zunächst beispielhaft zwei weitere Reaktions-Diffusions-Modelle vorstellen:

2.3.1. Das Gray-Scott-Modell

Das Gray-Scott Reaktions-Diffusions-Modell beruht auf dem Mechanismus:

$$\begin{aligned} C_1 + 2C_2 &\xrightarrow{f} 3C_2 \\ C_2 &\xrightarrow{k} P \end{aligned} \tag{2.7}$$

Hieraus ergeben sich zwei Ratengleichungen:

$$\begin{aligned}\partial_t C_1(\vec{x},t) &= D_{C_1}\nabla^2 C_1(\vec{x},t) - C_1(\vec{x},t)C_2(\vec{x},t)^2 + f(1-C_1(\vec{x},t)) \\ \partial_t C_2(\vec{x},t) &= D_{C_2}\nabla^2 C_2(\vec{x},t) + C_1C_2(\vec{x},t)^2 - (f+k)C_2(\vec{x},t)\end{aligned} \tag{2.8}$$

2.3.2. Der Oregonator

Ein anderes Modell, das ebenfalls wie der Brusselator auf dem FKN-Mechanismus beruht, jedoch etwas komplizierter ist, ist der *Oregonator*.
Dieser modelliert 5 chemische Reaktionen, die 7 chemische Spezies beinhalten:

$$\begin{aligned} A + C_2 &\xrightarrow{r_1} C_1 \\ C_1 + C_2 &\xrightarrow{r_2} P \\ B + C_1 &\xrightarrow{r_3} 2C_1 + C_3 \\ 2C_1 &\xrightarrow{r_4} Q \\ C_3 &\xrightarrow{r_5} fC_2, \end{aligned} \tag{2.9}$$

wobei f ein stöchiometrischer Parameter ist.
Daraus leiten sich die Ratengleichungen ab, wenn man -ähnlich wie beim Brusselator- annimt, dass A, B, Q und P konstant bleiben:

$$\begin{aligned}\partial_t C_1(\vec{x},t) &= r_1AC_2(\vec{x},t) - r_2C_1(\vec{x},t)C_2(\vec{x},t) + r_3AC_1(\vec{x},t) - 2r_4C_1^2(\vec{x},t) \\ \partial_t C_2(\vec{x},t) &= -r_1AC_2(\vec{x},t) - k_2C_1(\vec{x},t)C_2(\vec{x},t) + \frac{1}{2}r_5BC_3(\vec{x},t) \\ \partial_t C_3(\vec{x},t) &= 2r_3AC_1(\vec{x},t) - k_5BC_3(\vec{x},t)\end{aligned} \tag{2.10}$$

Das Oregonator-Modell ist physikalisch realistischer als der Brusselator, jedoch ist es auch komplexer. Da der Brusselator die wesentlichen Charakteristiken der BZ-Reaktion ebenfalls aufweist und zudem einfacher ist, haben wir diesen vorgezogen.

2.3.3. Das Arrhenius-Modell

Ein häufig verwandtes Modell für chemische Reaktionen ist auch das *Arrhenius-Modell* oder auch *laminar finite rate model*. Dieses wird zum Beispiel in dem kommerziellen Code *FLUENT* benutzt. Es eignet sich besonders für Verbrennungsreaktionen und wird vorwiegend für derartige Simulationen eingesetzt. Das Modell benötigt einige empirische thermodynamische Werte als Eingabe, die Paketen wie *CHEMKIN* [Lab13] oder *JANAF* zu entnehmen sind. Das Modell lässt sich sehr gut für Verbrennungsreaktionen einsetzen, da dafür eine Breite an empirischen Daten vorliegt. Wir wollen jedoch, wie schon angesprochen, in dieser Arbeit keine Verbrennungsreaktionen betrachten, da wir diese in Verbindung mit NaSt3D nicht gut behandeln können.

Wir haben uns entschieden, für diese Arbeit das Brusselator-Modell zu verwenden. Dieses ist von den Reaktions-Diffusions-Modellen das am besten untersuchte und ist mit zwei variablen Spezies noch vergleichsweise einfach gehalten. Als modellhaftes Problem lässt sich dieses einfacher behandeln als zum Beispiel das Arrhenius-Modell, welches Daten erfordert, die für viele der uns interessanten Anwendungen nicht vorhanden sind. Durch die enthaltene Nichtlinearität wird das Modell interessant in der numerischen Umsetzung. Es gibt noch nicht viele numerische Behandlungen des Brusselators, so dass eine Implementierung und Kopplung an einen Strömungscode lohnenswert erschien.

2.4. Die Navier-Stokes-Gleichungen und NaSt3DGPF

2.4.1. Das mathematische Modell: die Navier-Stokes-Gleichungen

Wie eingangs erwähnt, sollen in dieser Arbeit die chemischen Reaktionen, in verdünnter Form, in Verbindungen mit Fluidströmungen betrachtet werden.
Die Strömungsdynamik von Fluiden ist ein wichtiges und interessantes Gebiet für Ingenieure ebenso wie Naturwissenschaftler. Unter Fluiden verstehen wir unter Flüssigkeiten oder Gase.
Wir benutzen in dieser Arbeit den numerischen Strömungslöser NaSt3DGPF. Dieser beruht auf den mathematischen beziehungsweise physikalischen Modell der Navier-Stokes-Gleichungen. Es handelt sich dabei um partielle Differentialgleichungen, die zur Beschreibung inkompressibler Fluidströmungen dienen.

Dazu betrachten wir auf einem dreidimensionalen Gebiet $\Omega \subset \mathbb{R}^3$ für $0 \leq t \leq t_{end}$ das System:

$$\frac{\partial}{\partial t}\vec{u} + (\vec{u} \cdot \mathrm{grad})\,\vec{u} + \mathrm{grad}p = \frac{1}{Re}\Delta\vec{u} + \vec{G} \tag{2.11}$$

$$\nabla_x \cdot \vec{u} = 0 \tag{2.12}$$

Bei (2.11) handelt es sich um die *Impulsgleichung* (*momentum equation*), bei (2.12) um die *Kontinuitätsgleichung* (*continuity equation*). Hierbei bezeichnen:

$\vec{u} : \Omega \times [0, t_{end}] \to \mathbb{R}^3$ das Geschwindigkeitsfeld,

$p : \Omega \times [0, t_{end}] \to \mathbb{R}$ das Druckfeld,

$\rho : \Omega \times [0, t_{end}] \to \mathbb{R}$ die Dichte,

$\vec{G} \in \mathbb{R}^3$ die Volumenkräfte,

$Re \in \mathbb{R}$ die dimensionslose Reynoldszahl.

Um das System zu vervollständigen seien auch gewisse Anfangsbedingungen und Randbedingungen vorgegeben.
Das Gleichungssystem ist im allgemeinen analytisch nicht eindeutig lösbar. (Für bestimmte zwei-dimensionale Fälle gibt es eine eindeutige Lösung, vgl. [GDN98] beziehungsweise den Verweis dort.) Deshalb ist eine numerische Berechnung der Lösungen von hoher Bedeutung.

Zusätzlich zu (2.11) und (2.12) werden wir weiterhin Gleichungen benötigen, die den Transport von skalaren Werten modellieren, in unserem Fall der Konzentrationen der Chemikalien:

$$\partial_t C + \vec{u}\nabla_x C_i = D_{C_i}\Delta C_i \qquad \forall i = 1 \dots S \tag{2.13}$$

$C_i : \Omega \times [0, t_{end}] \to \mathbb{R}$ die Chemikalien

D_{C_i} Diffusionskonstanten

Dabei handelt es sich um klassische *Konvektions-Diffusions-Gleichungen*, die über die jeweiligen Größen aussagen, dass sie sich konvektiv und diffusiv ausbreiten.
Diese können auch für beliebige andere skalare Größen verwendet werden.

Die Navier-Stokes-Gleichungen leiten sich her aus den Sätzen der Massenerhaltung und der Impulserhaltung. Dazu sei hier auf [GDN98] verwiesen.

2.4.2. Diskretisierung in NaSt3D

Wir wollen nun im Detail darauf eingehen, wie die Navier-Stokes-Gleichungen in NaSt3DGPF diskretisiert sind. Dazu wollen wir uns der Es sei im Folgenden

$$\Omega = [0, a] \times [0, b] \times [0, c] \subset \mathbb{R}^3$$

das Strömungsgebiet. Es hat eine Quaderform, was uns ermöglicht es mit einem äquidistanten Gitter zu diskretisieren.

Wir können (2.11) und (2.12) in eine komponentenweise Notation umformulieren, welche für die Implementierung einfacher zu handhaben ist.

$$\begin{aligned}
\frac{\partial u}{\partial t} + \frac{\partial p}{\partial x} &= \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) - \frac{\partial(u^2)}{\partial x} - \frac{\partial(uv)}{\partial y} - \frac{\partial(uw)}{\partial z} + G_x \\
\frac{\partial v}{\partial t} + \frac{\partial p}{\partial y} &= \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) - \frac{\partial(uv)}{\partial x} - \frac{\partial(v^2)}{\partial y} - \frac{\partial(vw)}{\partial z} + G_y \\
\frac{\partial w}{\partial t} + \frac{\partial p}{\partial z} &= \frac{1}{Re}\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) - \frac{\partial(uw)}{\partial x} - \frac{\partial(vw)}{\partial y} - \frac{\partial(w^2)}{\partial z} + G_z
\end{aligned} \tag{2.14}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{2.15}$$

wobei wir nun mit u und v die Geschwindigkeitskomponenten meinen:

$$\begin{aligned} u &: \Omega \times [0, t_{end}] \to \mathbb{R} \text{ die Geschwindigkeit in x-Richtung} \\ v &: \Omega \times [0, t_{end}] \to \mathbb{R} \text{ die Geschwindigkeit in y-Richtung} \\ w &: \Omega \times [0, t_{end}] \to \mathbb{R} \text{ die Geschwindigkeit in z-Richtung.} \end{aligned}$$

Wir haben also Zeitableitungen $\frac{\partial}{\partial t}$ wie auch Ortsableitungen $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$ und $\frac{\partial}{\partial z}$ zu diskretisieren.

Zur Diskretisierung im Ort ist es üblich für die Navier-Stokes-Gleichungen ein *versetztes Gitter* (*staggered grid*) zu verwenden. Dabei liegen die verschiedenen zu berechnenden Größen u, v und p nicht in den gleichen Gitterpunkten. Der Grund dafür liegt in der Stabilität der numerischen Berechnung.
Man kann sich das so vorstellen, dass die Werte eigentlich zu drei verschiedenen Gittern gehören. Diese liegen, versetzt um eine halbe Maschenweite, übereinander.

Es liegen also für die Zelle (i, j, k) der dazugehörige Druck $p_{i,j,k}$ im Zellmittelpunkt und die Geschwindigkeiten $u_{i,j,k}$, $v_{i,j,k}$ und $w_{i,j,k}$ an den Kanten.
Wir haben also:

$$\begin{aligned} p_{i,j,k} \quad &\to \quad ((i-0.5)dx, (j-0.5)dy, (k-0.5)dz) \\ u_{i,j,k} \quad &\to \quad (idx, (j-0.5)dy, (k-0.5)dz) \\ v_{i,j,k} \quad &\to \quad ((i-0.5)dx, jdy, (k-0.5)dz) \\ w_{i,j,k} \quad &\to \quad ((i-0.5)dx, (j-0.5)dy, kdz) \end{aligned} \tag{2.16}$$

Für skalare Größen, wie die chemischen Konzentrationen, gilt im Allgemeinen (und insbesondere in dieser Arbeit) dass diese in den Zellmittelpunkten liegen, also an den gleichen Stellen wie die Druckwerte.

$$C_{i,j,k} \quad \to \quad ((i-0.5)dx, (j-0.5)dy, (k-0.5)dz) \tag{2.17}$$

Diese Art der Gebietsdiskretisierung hat zur Folge, dass direkt auf dem Rand des Gebietes nicht alle Werte verfügbar sind. Wir benötigen diese jedoch zur Erfüllung der Randbedingungen. Deshalb verfährt man so, dass noch eine Randschicht von Gitterzellen eingeführt wird, die sogenannten *Ghostzellen*. Die Zellen liegen eigentlich außerhalb des Gebietes. So ist es möglich, die Randbedingungen durch eine Mittelung der nächstliegenden Werte zu implementieren.

Randbedingungen

Wie schon erwähnt müssen noch Randbedingungen gesetzt werden. Diese gelten jeweils über die Zeit $[0, t_{end}]$ hinweg für $\vec{u}$ auf dem Gebietsrand $\delta\Omega$. Die Randbedingungen für

den Druck p werden aus den Geschwindigkeiten errechnet, worauf später noch eingegangen wird.
Es gibt zwei verschiedene Formen der Randbedingungen: *Dirichletbedingungen* und *Neumannbedingungen*. Bei den Dirchletbedingungen wird die entsprechende Variable an der Randstelle direkt auf einen festen Wert gesetzt. Bei den Neumannbedingungen wird dagegen der Gradient vorgegeben.

Wir können an den einzelnen Rändern unterschiedliche Randbedingungen setzen. Es ist auch möglich nur einen Teil eines Randes zum Beispiel als Ausströmbereich zu definieren, während am Rest des Randes eine slip-Bedingung gilt.

In dem Fall teilt sich dann ein Gebietsrand Γ auf in

$$\Gamma = \Gamma_{in/out} \cup \Gamma_s$$

Für Strömungen haben wir folgende physikalisch sinnvolle Arten von Randbedingungen:

- **Haftbedingung Γ_h (no-slip)**: Das Fluid "haftet" an diesem Rand. Dies ist der Fall wenn der Geschwindigkeitsvektor verschwindet:

$$u\Big|_{\Gamma_h} = 0$$

- **Ein-/Ausströmbedingungen $\Gamma_{in/out}$ (in-/outflow)**: Es wird eine bestimmte Geschwindigkeit vorgegeben, die das Fluid bei der Ein- oder Ausströmung erfüllt. Dies kann man sich so vorstellen, dass es zum Beispiel durch eine Schleuse oder ähnliches geleitet wird.

$$u\Big|_{\Gamma_{in/out}} = u_0$$

- **Rutschbedingung Γ_s (slip)**: Das Fluid gleitet ohne Reibungsverluste diesen Rand entlang.

$$(u \cdot \vec{n})\Big|_{\Gamma_s} = 0, \qquad \partial_{\vec{n}}(u \cdot t)\Big|_{\Gamma_s} = 0, \qquad \partial_{\vec{n}}(u \cdot s)\Big|_{\Gamma_s} = 0$$

- **Ausströmbedingung Γ_{out} (outflow)**: Die Fluidgeschwindigkeit ändert sich am Ausströmrand nicht.

$$\partial_{\vec{n}} u\Big|_{\Gamma_{out}} = 0$$

- **Periodische Randbedingung Γ_p**: Es müssen die Geschwindigkeiten an beiden Rändern in einer Koordinatenrichtung übereinstimmen.

Dabei soll die *Kompatibititätsbedingung* erfüllt sein:

$$\int_\Gamma \begin{pmatrix} u \\ v \end{pmatrix} \cdot n ds = 0 \tag{2.18}$$

Chorin'sche Projektionsmethode

Wir wollen nun auf die Methode eingehen, nach der die Navier-Stokes-Gleichungen (2.11) und (2.12) gelöst werden. Hierbei halten wir uns an die Darstellungen von [Cro10], [Cro02] und [GDN98].
Zunächst werden wir die Impuls- und die Kontinuitätsgleichungen entkoppeln, um so Druck und Geschwindigkeiten getrennt zu berechnen. Bei einem solchen Vorgehen, in dem ein Problem in zwei Teilprobleme zerlegt wird handelt es sich um eine Splittingmethode. Diese Art von numerischen Verfahren wird uns in 3.1.4 noch genauer beschäftigen.
Die für NaSt3D verwendete Methode ist die *Chorin'sche Projektionsmethode*, oder auch gelegentlich als *Chorin-Themam-Projektionsmethode* bezeichnet, die unabhängig voneinander in [Cho68] und [Té69] entwickelt wurde.
Wir betrachten das Verfahren hier im einphasigen Fall. Für eine Beschreibung des Zweiphasen-Systems sei auf [Cro10] verwiesen.

Wir wollen, ausgehend von den bekannten Werten des Zeitschrittes k, nun die Geschwindigkeiten und den Druck für den Zeitschritt $k+1$ berechnen.
Zuerst berechnen wir Schätzgeschwindigkeiten $\vec{u}^*$:

$$\vec{u}^* = \vec{u}^k + \delta t(-\nabla \cdot (\vec{u}^k \otimes \vec{u}^k)) + \vec{G} + \frac{1}{\rho}(\nabla \cdot (\mu D^k)) \tag{2.19}$$

Dabei handelt es sich um ein temporäres Geschwindigkeitsfeld. Dieses ist noch divergenzfrei.
Wir haben dann:

$$\frac{\vec{u}^{k+1} - \vec{u}^*}{\delta t} + \frac{\nabla p^{k+1}}{\rho} = 0 \tag{2.20}$$

$$\nabla \cdot \vec{u}^{k+1} = 0 \tag{2.21}$$

Daraus erhalten wir indem in (2.20) die Divergenz gebildet wird zusammen mit (2.21):

$$\nabla \cdot \left(\frac{1}{\rho}\nabla p^{k+1}\right) = \nabla \cdot \frac{\vec{u}^*}{\delta t}, \tag{2.22}$$

die *Druck-Poisson-Gleichung*.
Nachdem diese gelöst wurde, ergibt sich die gesuchte Geschwindigkeit $\vec{u}^{k+1}$ aus der Schätzgeschwindigkeit $\vec{u}^*$ und einer Druckkorrektur mit den neuen Druckwerten:

$$\vec{u}^{k+1} = \vec{u}^* - \frac{\delta t}{\rho}\nabla p^{k+1} \tag{2.23}$$

Wir müssen noch Randbedingungen für (2.22) angeben: falls man am Rand $\vec{u}^*_\Gamma = \vec{u}^{k+1}_\Gamma$ setzt, ergibt sich

$$\left.\frac{\partial p^{k+1}}{\partial \vec{n}}\right|_\Gamma = 0, \tag{2.24}$$

eine homogene Neumann-Randbedingung.

Ortsdiskretisierung

Wir haben noch die

- diffusiven Terme: $\frac{\partial^2 u}{\partial x}$, $\frac{\partial^2 u}{\partial y}$, $\frac{\partial^2 u}{\partial z}$, $\frac{\partial^2 v}{\partial x}$, $\frac{\partial^2 v}{\partial y}$, $\frac{\partial^2 v}{\partial z}$, $\frac{\partial^2 w}{\partial x}$, $\frac{\partial^2 w}{\partial y}$ und $\frac{\partial^2 w}{\partial z}$

und die

- konvektive Terme: $\frac{\partial(u^2)}{\partial x}$, $\frac{\partial(v^2)}{\partial y}$, $\frac{\partial(w^2)}{\partial z}$, $\frac{\partial uv}{\partial y}$, $\frac{\partial uv}{\partial x}$, $\frac{\partial uw}{\partial z}$, $\frac{\partial vw}{\partial z}$, $\frac{\partial uw}{\partial x}$ und $\frac{\partial vw}{\partial y}$

zu diskretisieren.

Für die diffusiven Terme benutzen wir zentrale Differenzen mit halber Schrittweite:

$$\begin{aligned}\left[\frac{\partial^2 u}{\partial x^2}\right]_{i,j,k} &= \frac{1}{\Delta x_{i+\frac{1}{2}}}\left(\frac{u_{i+1,j,k}-u_{i,j,k}}{\Delta x_i} - \frac{u_{i,j,k}-u_{i-1,j,k}}{\Delta x_{i-1}}\right)\\ \left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j,k} &= \frac{1}{\Delta y_j}\left(\frac{u_{i,j+1,k}-u_{i,j,k}}{\Delta y_{j+\frac{1}{2}}} - \frac{u_{i,j,k}-u_{i,j-1,k}}{\Delta y_{j-\frac{1}{2}}}\right)\end{aligned} \tag{2.25}$$

mit

$$\Delta x_i := x_i - x_{i-1} \qquad \text{und} \qquad \Delta x_{i+\frac{1}{2}} := \frac{\Delta x_i + \Delta x_{i+1}}{2}\ . \tag{2.26}$$

Entsprechend gehen wir auch für die anderen diffusiven Terme vor.

Für die konvektiven Terme haben wir verschiedene Verfahren zur Auswahl, darunter VONOS, ENO und WENO. Für eine eingehendere Beschreibung dieser Methoden siehe etwa [Cro02].

Zeitdiskretisierung

Nachdem alle Ortsableitungen diskretisiert wurden, verbleiben noch die Zeitableitungen $\frac{\partial u}{\partial t}$ und $\frac{\partial v}{\partial t}$. Es ist also notwendig auch in der Zeit zu diskretisieren.

Wir teilen das Intervall $[0, t_{end}]$ a uf in die Abschnitte

$$[k\delta t, (k+1)\delta t] \qquad\qquad k = 0, \ldots, k_{max}$$

mit $k_{max} = \frac{t_{end}}{\delta t} - 1$ und $\delta t \in \mathbb{R}^+$.
Dabei wird δt in jedem Schritt neu bestimmt, worauf wir in 2.4.2 noch eingehen.

Wir betrachten also die Werte u, v und p nur an den Zeitpunkten $k\delta t$. Wir bezeichnen die diskreten Zeitpunkte im Folgenden auch als t_k und mit u^k, v^k etc. die Werte der Variablen zum Zeitpunkt t_k.

Wir können dann die Zeitableitungen für t_{k+1} diskretisieren als:

$$\left[\frac{\partial u}{\partial t}\right]^{(k+1)} := \frac{u^{k+1} - u^k}{\delta t} \qquad \text{und} \qquad \left[\frac{\partial v}{\partial t}\right]^{(k+1)} := \frac{v^{k+1} - v^k}{\delta t} \tag{2.27}$$

Dies entspricht dem *explizitem Euler-Verfahren.*
Alternativ bieten sich in NaSt3D auch *Adams-Bashforth* oder *Runge-Kutta-Methoden.*

Zeitschrittweitensteuerung

NaSt3dGPF verwendet eine adaptive Zeitschrittweitensteuerung. Es wird also nicht in jedem Zeitschritt k die gleiche Schrittweite δt benutzt. Aus Gründen der Stabilität muss diese angepasst werden.
Die Bedingungen, die wir stellen werden, besagen, dass sich die Informationen pro Zeitschritt nicht weiter als eine Gitterweite bewegen dürfen. Sonst würden Informationen verloren gehen.
Beschränkungen für die Zeitschrittweite ergeben sich jeweils aus den konvektiven und den viskosen Termen, sowie durch die Volumenkräfte. Später werden wir noch eine Beschränkung aufgrund des Chemiemodells beziehungsweise der Kopplung mit diesem hinzufügen. Dies wird in 3.7.3 behandelt.
Durch die konvektiven Terme ergibt sich:

$$\delta t \leq \min_{\Omega} \left(\frac{\delta x}{|u_{max}|}, \frac{\delta y}{|v_{max}|}, \frac{\delta z}{|w_{max}|} \right). \tag{2.28}$$

Diese Bedingungen sind bekannt als die *Courant-Friedrichs-Lewy-Bedingung* oder kurz: *CFL-Bedingung.* Sie besagen, dass kein Fluidpartikel in der Zeit δt mehr als um eine Gitterweite konvektiv bewegt werden darf.
Aufgrund der viskosen Terme haben wir die Zeitschrittweitenbeschränkung:

$$\delta t \leq \left(\frac{\mu}{\rho} \left(\frac{2}{(\delta x)^2} + \frac{2}{(\delta y)^2} + \frac{2}{(\delta z)^2} \right) \right)^{-1} \tag{2.29}$$

Insgesamt ergibt sich:

$$\delta t := \tau \; min \left(\frac{\mu}{\rho} \left(\frac{2}{(\delta x)^2} + \frac{2}{(\delta y)^2} + \frac{2}{(\delta z)^2} \right)^{-1}, \frac{\delta x}{|u_{max}|}, \frac{\delta y}{|v_{max}|}, \frac{\delta z}{|w_{max}|} \right) \tag{2.30}$$

Dabei stellt $\tau \in (0, 1]$ einen Sicherheitsfaktor dar. Das ergibt sich daraus, dass wir immer nur die Geschwindigkeiten aus dem vorangegangen Zeitschritt für die Abschätzungen nutzen können. Deshalb kann die Abschätzung etwas daneben liegen und wird durch τ sicherheitshalber etwas verringert. Wenn die Strömung relativ dynamisch ist, sich also die Geschwindigkeitswerte schnell ändern, sollte τ klein gewählt werden.

Für den Einfluß der Volumenkräfte auf die Zeitschrittweitenbeschränkung siehe siehe [Cro10].

Weiterhin bietet der Code auch noch die Möglichkeit, bei den Simulationen eine feste Schranke $delt_{max}$ vorzugeben, so dass dann $\delta t := min(delt_{max}, \tilde{\delta t})$ gewählt wird, wenn $\tilde{\delta t}$ gemäß (2.30) bestimmt wurde.

Fluid-Struktur-Interaktionen

Weiterhin verfügt NaSt3DGPF über das Feature der Fluid-Struktur-Interaktion, welches wir später verwenden werden eingehen. Dies ermöglicht die Verwendung von beweglichen Starrkörpern (Hindernissen) in den Simulationen. Die komplexen Geometrien werden mittels einer Level-Set-Methode beschrieben. Die Wechselwirkung zwischen dem Starrkörper und dem Fluid erfolgt bidirektional, also zum einen wird der Starrkörper mit dem Fluid transportiert und zum anderen erfolgte eine Einbindung in die Geschwindigkeitsberechnung der Strömung.
Für eine genauere Beschreibung der verwendeten Methodik siehe [Cro10].

2.4.3. Zusammenfassung des Algorithmus

Wir fassen nun noch den vollständigen Algorithmus zusammen, nach [Cro10]:

Algorithmus 1: Lösen der inkompressiblen Navier-Stokes-Gleichungen

Input : Ω, t_{end}, Maschenweiten dx, dy, dz, $\vec{G}$, μ, ρ, $\vec{u}_0$, $\vec{p}_0$
Output : Druck, Geschwindigkeit
Setze $t = 0$, $k = 0$
Setze Anfangswerte $\vec{u}^k = \vec{u}_0$, $\vec{p}^k = \vec{p}_0$
Setze Randwerte für $\vec{u}^k$ gemäß Abschnitt 2.4.2
while $t \leq T_{end}$ **do**
  Berechne Zeitschrittweite:
  gemäß (2.30)
  Berechne Schätzgeschwindigkeiten:
  berechne $\vec{u}^*(\vec{u}^k, t^k)$ gemäß (2.19)
  setze Randwerte für $\vec{u}^*(\vec{u}^k, t^k)$
  Löse Druck-Poisson-Gleichung:
  setze rechte Seite in Abhängigkeit von $\vec{u}^*(\vec{u}^k, t^k)$
  berechne iterativ p^{k+1} gemäß (2.22)
  Führe Druck-Korrektur der Geschwindigkeiten durch:
  berechne $\vec{u}^{k+1}$ aus $\vec{u}^*(\vec{u}^k, t^k)$ und p^{k+1} gemäß (2.23)
  setze Randwerte für $\vec{u}^{k+1}$
  Setze $t = t + \delta t$, $k = k + 1$
end

2.4.4. Überblick über NaSt3DGPF

Wir haben der Einfachheit halber nur die Grundfunktionalität des Programmpaketes NaSt3DGPF erklärt, das Berechnen von dreidimensionalen inkompressiblen Strömungen. Für die Durchführung der Kopplung unseres Chemiemodelles an den Code benötigen wir nur diese Teile.
Jedoch soll betont werden, dass viele weitere Features zur Verfügung stehen, welche den Einsatz des Codes gerade auch in Zusammenhang mit chemischen Reaktionen interessant machen, unter anderem

- **zweiphasige Strömungen** basierend auf einer Level-Set-Methode
- **freie Oberflächen mit Level-Set-Methode** (Roberto Croce, [Cro10])
- **Temperaturmodellierung** mittels einer Boussinesq-Approximation
- **komplexe dreidimensionale Hindernisgeometrien** (Peter Zaspel, [Zas09])
- **parallelisierte Berechnung** der Strömung basierend auf einer MPI-Implementierung
- diverse **Schemata für die konvektiven Terme**, auch höherer Ordnung
- **bewegbare Geometrien** (Roberto Croce, [Cro10])
- **Sedimentsimulationen** (Markus Burkow, [Bur10])
- verschiedene Verfahren zur Lösung der Druck-Poisson-Gleichung
- **Kontaktwinkelmodelle** (Margrit Klitz, [GK13])

3. Numerische Umsetzung des Chemiemodells und der Kopplung

Wir beschreiben nun, welche Schritte zur numerischen Umsetzung des im vorherigen Kapitel beschriebenen Modelles und seiner Verbindung mit dem Strömungscode erfolgt sind.

Neben der dreidimensionalen Implementierung des Brusselators in C++, zur Kopplung an den NaSt3D-Code, wurde zunächst auch ein 2D-Code für das Modell in MATLAB implementiert, welcher der einfacheren Analyse diente.

Zunächst erklären wir das grundsätzliche Vorgehen bei der Kopplung der Programmteile, dann wird im Detail auf die einzelnen Komponenten eingegangen.

3.1. Ortsdiskretisierung des Brusselators

3.1.1. Finite Differenzen - Methode

Die Brusselator-Gleichungen (2.3) und (2.4) beschreiben ein in Zeit und Ort kontinuierliches Gleichungssytem. Zur numerischen Lösung ist es notwendig, in eine diskrete Form überzugehen.

Wir diskretisieren nun das System zuerst im Ort, und zwar mit der Methode der *finiten Differenzen*. Dazu werden statt dem kontinuierlichen Gebiet Ω nur noch einzelne, endlich viele Punkte in Ω betrachtet. An diesen werden dann die benötigten Funktionen ausgewertet und die Differenzenquotienten approximiert. Die Diskretisierungspunkte bilden ein Gitter, das Ω überdeckt. Wir sprechen auch von einem *Diskretisierungsgitter*. Das so diskretisierte Gebiet bezeichen wir hier mit Ω_h, wobei h für die Maschenweite des Gitters, also den Abstand zwischen zwei Punkten, steht.

Diskretisierungsgitter

Wir betrachten im Folgenden ein rechteckiges Gebiet, $\Omega = [x_{min}, x_{max}] \times [y_{min}, y_{max}]$, wobei wir uns ohne Einschränkung auf $[0, 1] \times [0, 1]$ beziehen wollen.

Hierfür wollen wir ein in jeder Raumrichtung äquidistantes Gitter mit $N \times N$ inneren Punkten aufstellen. Es sei

$$h = \frac{x_{max} - x_{min}}{N}$$

die *Maschenweite* des Gitters.
In unserem Fall gilt

$$h = \frac{1}{N}.$$

Damit haben wir

$$\Omega_h := \{(ih, jh) | i, j = 0 \dots (N+1)\}$$

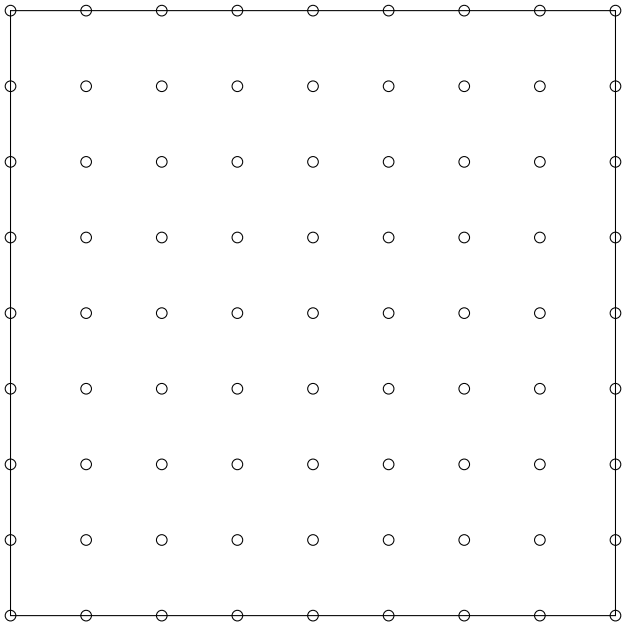


Abbildung 3.1.: Beispiel für ein ortsdiskretes Gitter

Das Gitter Ω_h hat somit $N \cdot N$ innere Punkte. Die gesamten Gitterpunkte, also inklusive derjenigen, die direkt auf dem Rand liegen, sind $(N+2) \cdot (N+2)$ viele.

Wir interpretieren hier immer die Größen C_i als in den Mittelpunkten der kontinuierlichen Zellen liegend, wie es typischerweise für alle skalaren Größen gemacht wird.

Wir gehen also über von den kontinuierlichen zu den ortsdiskreten Größen:

$$\begin{aligned} \Omega &\implies \Omega_h, \\ C_i(\vec{x}, t) &\implies C_{i,h}(x_{ij}, t), \end{aligned}$$

wobei:

$$C_i(\vec{x}, t) : \Omega \times [t_0, t_{end}] \to \mathbb{R}$$

$$C_{i,h}(\vec{x},t) : \Omega_h \times [t_0, t_{end}] \to \mathbb{R} \text{ für } i = 1 \dots S.$$

Wir wollen nun der Übersichtlichkeit halber in der Notation das h weglassen. Wir bezeichen also vorerst mit C das diskrete C_h und so weiter.

Zu beachten ist, dass es sich hierbei bis jetzt nur um eine Diskretisierung im Ort handelt. Dies reicht noch nicht aus, um das System am Rechner lösen zu können. Wir werden in 3.2 noch eine Diskretisierung von $[t_0, t_{end}]$ vornehmen.

Differenzenquotienten

Die Approximation der Differenzenquotienten geht auf die Definition einer Ableitung zurück:

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{3.1}$$

Lassen wir hierbei die Grenzwertbildung weg, haben wir eine gewisse Annäherung an den Wert der Ableitung. Wenn wir das Gitter fein genug wählen, also h (fest gewählt) genügend klein ist, können wir somit die Ableitung numerisch approximieren.

Bilden wir also die *Vorwärts-Differenz*, haben wir für $\frac{df}{dx}$ an der Stelle x:

$$\frac{df}{dx}(x) \approx \frac{f(x+h) - f(x)}{h}, \tag{3.2}$$

beziehungsweise in Gitternotation

$$\frac{df}{dx}(x_{ij}) = \frac{f(x_{i(j+1)}) - f(x_{ij})}{h}. \tag{3.3}$$

Als alternative Möglichkeiten für Differenzenquotienten haben wir die *Rückwärts-Differenz*

$$\frac{df}{dx} \approx \frac{f(x) - f(x-h)}{h} \tag{3.4}$$

sowie die *zentrale Differenz*:

$$\frac{df}{dx} \approx \frac{f(x+h) - f(x-h)}{2h}. \tag{3.5}$$

Anhand einer Taylorentwicklung kann man die Fehlerordnung der Formeln bestimmten. Diese beträgt für die Vorwärts- und Rückwärtsdifferenzen $O(h)$, für die zentrale Differenz $O(h^2)$.

Dabei handelt es sich um Approximationen in einem eindimensionalen Gebiet - wir ziehen nur die “linken” und “rechten” Nachbarn $x + h$ beziehungsweise $x - h$ zur Berechnung heran. Das Verfahren lässt sich entsprechend auch auf 2D oder 3D erweitern.

Im zweidimensionalen Fall benötigen wir die vier Nachbarn $x_{i(j+1)}$, $x_{i(j-1)}$, $x_{(i+1)j}$, $x_{(i-1)j}$, wodurch sich bildhaft ein sogenannter *5-Punkte Differenzenstern* ergibt.

Für unsere Implementierung der 2D-Diffusionsgleichung in MATLAB werden wir eine solche finite-Differenze-Methode verwenden.
Für den dreidimensionalen Fall werden wir später einen Programmteil von NaSt3DGPF verwenden, der die Diffusion im dreidimensionalen durchführt. Auch dieser basiert auf finiten Differenzen. Zu beachten ist allerdings bei der Kopplung an NaSt3D, dass in diesem Code die Geschwindigkeitswerte auf einem *staggered grid* liegen, wie in 2.4.2 beschrieben.

3.1.2. Ortsdiskretes Gleichungssystem (2D)

Nachdem wir das Prinzip unserer Ortsdiskretisierung vorgestellt haben, soll diese nun an den konkreten Gleichungen ausgeführt werden.

Wir wollen das ortsdiskrete Gleichungssystem für den zweidimensionalen Fall ausschreiben.
Dabei gehen wir zusätzlich von der häufig angenommenen (vgl. etwa [HNW00], [SMM13]) vereinfachenden Annahme, dass $k_1 = k_2 = k_3 = k_4 = 1$ gilt, aus.
Damit haben (2.3) und (2.4) die vereinfachte Gestalt:

$$\partial t C_1(\vec{x},t) = D_{C_1}\Delta C_1(\vec{x},t) + A - (B+1)C_2(\vec{x},t) + C_1(\vec{x},t)^2 C_2(\vec{x},t) \qquad \forall x \in \Omega, \quad t \in (0, t_{end}] \tag{3.6}$$

$$\partial_t C_2(\vec{x},t) = D_{C_2}\Delta C_2(\vec{x},t) + BC_1(\vec{x},t) - C_1(\vec{x},t)^2 C_2(\vec{x},t) \qquad \forall x \in \Omega, \quad t \in (0, t_{end}], \tag{3.7}$$

wobei die Rand- und Anfangsbedingungen bleiben wie in (2.6) und (2.5) angegeben.
Unsere Überlegungen lassen sich auch auf den allgemeinen und auf den dreidimensionalen Fall erweitern.

Zur Diskretisierung unseres Gleichungssystems (2.3) muss nun insbesondere der Diffusionsanteil mit dem *Laplaceoperator* Δ diskretisiert werden.
Dieser ist definiert als:

$$\Delta = \sum_{k=1}^{n} \frac{\partial^2}{\partial^2 x_k}$$

Im Falle von $\mathbb{R}^2$ ist

$$\Delta C(\vec{x},t) = \frac{\partial^2 C(\vec{x},t)}{\partial^2 x_1} + \frac{\partial^2 C(\vec{x},t)}{\partial^2 x_2}$$

Wir erhalten gemäß 3.1.1 als im Ort diskretes Δ_h:
Für $x_{ij} \in \Omega_h$:

$$\Delta_h C_l(x_{ij},t) = \frac{C_l(x_{i(j+1)},t) + C_l(x_{i(j-1)},t) + C_l(x_{(i+1)j},t) + C_l(x_{(i-1)j},t) - 4C_l(x_{ij},t)}{h^2} \tag{3.8}$$

für $l \in \{1,2\}$.

Indem wir die Diskretisierung (3.8) in die Brusselator-Gleichungen (3.6) einsetzen, erhalten wir für die Konzentration C_1 am Zeitpunkt t:
Für $x_{ij} \in \Omega_h$:

$$\frac{d}{dt}C_1(x_{ij},t) = A + C_1(x_{ij},t)^2 C_2(x_{ij},t) - (A+1)C_1(x_{ij},t) + \frac{D_{C_1}}{h^2}\Big(C_1(x_{i(j+1)},t) + C_1(x_{i(j-1)},t) + C_1(x_{(i+1)j},t) + C_1(x_{(i-1)j},t) - 4C_1(x_{ij},t)\Big),$$

also:

$$\begin{aligned}\frac{d}{dt}C_1(x_{ij},t) = A + C_1(x_{ij},t)^2 C_2(x_{ij},t) - (A+1+4\frac{D_{C_1}}{h^2})C_1(x_{ij},t) + \\ \frac{D_{C_1}}{h^2}\Big(C_1(x_{i(j+1)},t) + C_1(x_{i(j-1)},t) + C_1(x_{(i+1)j},t) + C_1(x_{(i-1)j},t)\Big)\end{aligned} \tag{3.9}$$

Für C_2 ergibt sich entsprechend:
Für $x_{ij} \in \Omega_h$:

$$\frac{d}{dt}C_2(x_{ij},t) = BC_1(x_{ij},t) - C_1(x_{ij},t)^2 C_2(x_{ij},t) + \frac{D_{C_2}}{h^2}\Big(C_2(x_{i(j+1)},t) + C_2(x_{i(j-1)},t) + C_2(x_{(i+1)j},t) + C_2(x_{(i-1)j},t) - 4C_2(x_{ij},t)\Big) \tag{3.10}$$

Diese beiden Gleichungen sind jeweils an jedem der inneren Gitterpunkte zu lösen. Insgesamt haben wir also $2 \cdot N \cdot N$ Gleichungen zu lösen. Es ist zu beachten, dass für den Diffusionsanteil auch Werte an Punkten außerhalb des Gebiets benötigt werden, während sich der Reaktionsteil nur “lokal” in einem Punkt errechnet.

Wir können die Konzentrationen C_1 und C_2, in lexikografischer Anordnung, in einen

Vektor zusammenfassen:

$$\vec{C}(t) = \begin{pmatrix} C_1(t) \\ C_2(t) \end{pmatrix} = \begin{pmatrix} C_1(x_{11},t) \\ C_1(x_{12},t) \\ C_1(x_{13},t) \\ \vdots \\ C_1(x_{i(j-1)},t) \\ C_1(x_{ij},t) \\ C_1(x_{i(j+1)},t) \\ \vdots \\ C_1(x_{N(N-1)},t) \\ C_1(x_{NN},t) \\ C_2(x_{11},t) \\ C_2(x_{12},t) \\ C_2(x_{13},t) \\ \vdots \\ C_2(x_{i(j-1)},t) \\ C_2(x_{ij},t) \\ C_2(x_{i(j+1)},t) \\ \vdots \\ C_2(x_{N(N-1)},t) \\ C_2(x_{NN},t) \end{pmatrix} \tag{3.11}$$

Es gilt $\vec{C} \in \mathbb{R}^{2 \cdot N \cdot N}$, da wir an den $N \cdot N$ inneren Gitterpunkten jeweils beide Konzentrationswerte haben.

Damit können wir (3.9) und (3.10) zusammenfassen in das Gleichungssystem:

$$\frac{d}{dt}C(x_{ij},t) = \underbrace{MC(x_{ij},t) + \vec{a}}_{\text{linearer Teil}} + \underbrace{b(C_1(x_{ij},t), C_1(x_{ij},t), C_2(x_{ij},t))}_{\text{nichtlinearer Teil}} \qquad \text{für } x_{ij} \in \Omega_h, \quad t \in (0, t_{end}] \tag{3.12}$$

wobei $b : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ eine Trilinearform ist und M die Matrix:

$$M = \left(\begin{array}{ccccc|ccccc} D_0 & I_0 & & & & & & & & \\ I_0 & \ddots & \ddots & 0 & & & & & & \\ & \ddots & \ddots & \ddots & & & & 0 & & \\ & 0 & \ddots & \ddots & I_0 & & & & & \\ & & & I_0 & D_0 & & & & & \\ \hline B & & & & & D_1 & I_1 & & & \\ & \ddots & & 0 & & I_1 & \ddots & \ddots & 0 & \\ & & \ddots & & & & \ddots & \ddots & \ddots & \\ & 0 & & \ddots & & & 0 & \ddots & \ddots & I_1 \\ & & & & B & & & & I_1 & D_1 \end{array}\right) \in \mathbb{R}^{2\cdot N\cdot N} \times \mathbb{R}^{2\cdot N\cdot N} \tag{3.13}$$

wobei D_0, D_1, I_0 und I_1 die Blockmatrizen sind:

$$D_0 = \begin{pmatrix} -(B+1+4\frac{D_{C_1}}{h^2}) & \frac{D_{C_1}}{h^2} & & & \\ \frac{D_{C_1}}{h^2} & \ddots & \ddots & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & \ddots & \ddots & \frac{D_{C_1}}{h^2} \\ & & & \frac{D_{C_1}}{h^2} & -(B+1+4\frac{D_{C_1}}{h^2}) \end{pmatrix} \in \mathbb{R}^N \times \mathbb{R}^N,$$

$$D_1 = \begin{pmatrix} (B-4\frac{D_{C_2}}{h^2}) & \frac{D_{C_2}}{h^2} & & & \\ \frac{D_{C_2}}{h^2} & \ddots & \ddots & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & \ddots & \ddots & \frac{D_{C_2}}{h^2} \\ & & & \frac{D_{C_2}}{h^2} & (B-4\frac{D_{C_2}}{h^2}) \end{pmatrix} \in \mathbb{R}^N \times \mathbb{R}^N,$$

$$I_1 = \frac{D_{C_1}}{h^2} \cdot E_N \quad \text{und}$$

$$I_2 = \frac{D_{C_2}}{h^2} \cdot E_N,$$

wobei E_N jeweils eine $N \times N$ Einheitsmatrix darstellt.

Der Vektor $\vec{a} \in \mathbb{R}^{2\cdot N\cdot N}$ enthält die Randwerte und den konstanten Teil von Gleichung (3.12). Dabei korrespondieren wieder die ersten $N \cdot N$ Einträge zu C_1 und die letzten $N \cdot N$ Einträge zu C_2.

$$
\vec{a} \quad = (a_i)_{i=1\ldots 2\cdot N\cdot N} \quad = \left(\begin{array}{c}
A + ul + ub \\
A + ul \\
\vdots \\
A + ul \\
A + ul + ut \\
A + ub \\
A \\
\vdots \\
A \\
A + ut \\
A + ub \\
A \\
\vdots \\
A \\
A + ut \\
A + ur + ub \\
A + ur \\
\vdots \\
A + ur \\
A + ur + ut \\
ul + ub \\
ul \\
\vdots \\
ul \\
ul + ut \\
ub \\
0 \\
\vdots \\
0 \\
ut \\
ub \\
0 \\
\vdots \\
0 \\
ut \\
ur + ub \\
ur \\
\vdots \\
ur \\
ur + ut
\end{array}\right)
\begin{array}{l}
\left.\begin{array}{l}\\ \\ \\ \\ \end{array}\right\} \text{für } i = 1 \ldots N \\
\text{für } i = N + 1 \\
\\ \\ \\
\text{für } i = 2N \\
\text{für } i = 2N + 1 \\
\\ \\ \\
\text{für } i = (N-1) \cdot N \\
\left.\begin{array}{l}\\ \\ \\ \\ \end{array}\right\} \text{für } i = ((N-1) \cdot N + 1) \ldots N \\
\\ \\ \\
\left.\begin{array}{l}\\ \\ \\ \\ \\ \\ \\ \\ \end{array}\right\} \text{für } i = (N \cdot N) + 1 \ldots (N+1) \cdot N \\
\\ \\ \\
\text{für } i = (N+1) \cdot N + 1 \\
\\ \\ \\ \\ \\
\text{für } i = (N+2) \cdot N \\
\text{für } i = (N+2) \cdot N + 1 \\
\\ \\ \\ \\ \\
\text{für } i = (N-1) \cdot 2N \\
\left.\begin{array}{l}\\ \\ \\ \\ \\ \\ \\ \\ \end{array}\right\} \text{für } i = (N-1) \cdot 2N + 1 \ldots 2 \cdot N \cdot N
\end{array}
$$

3.1.3. Gesplittetes Gleichungssystem

Wir haben jetzt das gesamte Gleichungssystem (3.12) betrachtet. Zur leichteren Beschreibung der eingesetzten Verfahren, wollen wir dieses nochmal in einer Form darstellen, die den linearen Teil des Gleichungssystems in den vom Diffusionsteil herrührenden Term und den restlichen, von den linearen Komponenten des Brusselators kommenden, Term unterscheidet.

Wir teilen M also auf:

$$M = M_1 + M_2$$

mit $M_1 \in \mathbb{R}^{2\cdot N\cdot N} \times \mathbb{R}^{2\cdot N\cdot N}, \quad M_2 \in \mathbb{R}^{2\cdot N\cdot N} \times \mathbb{R}^{2\cdot N\cdot N}$.

Dabei ist

$$M_1 = \left(\begin{array}{ccccc|ccccc} -(B+1) & & & & & & & & & \\ & \ddots & & 0 & & & & & & \\ & & \ddots & & & & & & 0 & \\ & 0 & & \ddots & & & & & & \\ & & & & -(B+1) & & & & & \\ \hline B & & & & & & & & & \\ & \ddots & & 0 & & & & & & \\ & & \ddots & & & & & & 0 & \\ & 0 & & \ddots & & & & & & \\ & & & & B & & & & & \end{array}\right) \tag{3.14}$$

$$M_2 = \begin{pmatrix} -4 & 1 & & & & & & & & \\ 1 & \ddots & \ddots & 0 & & & & & & \\ & \ddots & \ddots & \ddots & & & & 0 & & \\ & 0 & \ddots & \ddots & 1 & & & & & \\ & & & 1 & -4 & & & & & \\ & & & & & -4 & 1 & & & \\ & & & & & 1 & \ddots & \ddots & 0 & \\ & & 0 & & & & \ddots & \ddots & \ddots & \\ & & & & & & 0 & \ddots & \ddots & 1 \\ & & & & & & & & 1 & -4 \end{pmatrix} \tag{3.15}$$

Der Randvektor $\vec{a}$ teilt sich folgendermaßen auf:

$$
\vec{a} = \vec{a_1} + \vec{a_2} = \begin{pmatrix} ul + ub \\ ul \\ \vdots \\ ul \\ ul + ut \\ ub \\ 0 \\ \vdots \\ 0 \\ ut \\ ub \\ 0 \\ \vdots \\ 0 \\ ut \\ ur + ub \\ ur \\ \vdots \\ ur \\ ur + ut \\ ul + ub \\ ul \\ \vdots \\ ul \\ ul + ut \\ ub \\ 0 \\ \vdots \\ 0 \\ ut \\ ub \\ 0 \\ \vdots \\ 0 \\ ut \\ ur + ub \\ ur \\ \vdots \\ ur \\ ur + ut \end{pmatrix} + \begin{pmatrix} A \\ \\ \vdots \\ \\ \\ \\ \\ \vdots \\ \\ \\ \\ \\ \vdots \\ \\ \\ \\ \\ \vdots \\ A \\ 0 \\ \\ \vdots \\ \\ \\ \\ \\ \vdots \\ \\ \\ \\ \\ \vdots \\ \\ \\ \\ \\ \vdots \\ 0 \end{pmatrix}
$$

$\left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\}$ für $i = 1 \ldots N$

für $i = N + 1$

für $i = 2N$

für $i = 2N + 1$

für $i = (N - 1) \cdot N$

$\left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array}\right\}$ für $i = ((N - 1) \cdot N + 1) \ldots N$

$\left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array}\right\}$ für $i = (N \cdot N) + 1 \ldots (N + 1) \cdot N$

für $i = (N + 1) \cdot N + 1$

für $i = (N + 2) \cdot N$

für $i = (N + 2) \cdot N + 1$

für $i = (N - 1) \cdot 2N$

$\left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array}\right\}$ für $i = (N - 1) \cdot 2N + 1 \ldots 2 \cdot N \cdot N$

(3.16)

Wir erhalten somit das gesplittete Gleichungssystem:

$$\frac{d}{dt}C(x_{ij},t) = (M_1+M_2)C(x_{ij},t) + \vec{a_1} + \vec{a_2} + b(C_1(x_{ij},t), C_1(x_{ij},t), C_2(x_{ij},t)) \quad \text{für } x \in \Omega_h, \quad t \in (0, t_{end}] \tag{3.17}$$

Dieses können wir aufteilen in die beiden Systeme:

$$\frac{d}{dt}C(x_{ij},t) = M_1C(x_{ij},t) + \vec{a_1} \qquad \text{für } x \in \Omega_h, \quad t \in (0, t_{end}] \tag{3.18}$$

$$\frac{d}{dt}C(x_{ij},t) = M_2C(x_{ij},t) + \vec{a_2} + b(C_1(x_{ij},t), C_1(x_{ij},t), C_2(x_{ij},t)) \qquad \text{für } x \in \Omega_h, \quad t \in (0, t_{end}] \tag{3.19}$$

3.1.4. Operatorsplitting für das ortsdiskrete Brusselator-System

Wir haben also in (3.12) eine ortsdiskrete Formulierung des Systems aufgestellt. Dieses ist sehr groß, da es sich um ein $2 \cdot N \cdot N \times 2 \cdot N \cdot N$ Gleichungssystem handelt. Bereits für nicht allzu großes N sind das sehr viele zu lösende Gleichungen. Wir haben einen linearen Teil - der die Diffusion enthält sowie die linearen und konstanten Terme des Brusselators - und einen nichtlinearen Teil.

Es gibt nun verschiedene Wege, ein solches kompliziertes System von ODEs zu lösen. Was sich zur Vereinfachung anbietet, ist insbesondere das Vorgehen des *Operatorsplittings*. Dieses funktioniert allgemein wie folgt:

Sei ein System von Differentialgleichungen gegeben durch:

$$\frac{\delta u}{\delta t} + \mathcal{L} = 0, \qquad \mathcal{L} = \sum_{k=1}^{S} \mathcal{L}_k \tag{3.20}$$

Wir haben also eine Summe von Operatoren. Die Idee des Splittings besteht nun darin, die Operatoren $\mathcal{L}_k$ einzeln nacheinander - und somit unabhängig voneinander - abzuarbeiten.

Dieses Vorgehen wird häufig für große Systeme von DGLen eingesetzt. Auch die in NaSt3D verwendete Chorin'sche Projektionsmethode ist ein Operatorsplitting.
Es handelt sich dabei um eine *divide and conquer*-Strategie.
Dadurch wird das zu lösende Problem übersichtlicher und man kann einige Vorteile ausnutzen. So können unterschiedliche Verfahren für die Teilprobleme eingesetzt werden, zum Beispiel kann ein Teil explizit und ein Teil implizit behandelt werden. Durch die Verwendung verschiedener Algorithmen für die Teilprobleme kann man in einigen Fällen eine bessere Stabilität erzielen und den Rechenaufwand geringer halten im Vergleich zu

der Lösung des Gesamtsystem mit einem aufwändigeren Verfahren. Weiterhin können bei Bedarf verschiedene Zeitschrittweiten für die Teilprobleme eingesetzt werden.

In unserem Fall wollen wir das Problem in zwei Teile splitten also ist $S = 2$. Wir setzen $\mathcal{L}_1 = M_1 + b$ (Reaktion) und $\mathcal{L}_2 = M_2$ (Diffusion).

Ein großer Vorteil unseres Vorgehens ist die entkoppelte Behandlung von Reaktion und Diffusion. Insbesondere können wir bei der Implementierung in NaSt3D dann für die Diffusion auf eine bereits vorhandene Diskretisierung zurückgreifen.

Yanenko-Splitting

Die in dieser Arbeit verwendete Methode ist ein *Yanenko-Splitting* für $S = 2$, also für eine Aufteilung in zwei Operatoren. Dabei handelt es sich um eine *fractional step*-Methode.

Wir beschreiben das Vefahren nach [Kuz].
Für $S = 2$ haben wir das System:

$$\frac{\partial u}{\partial t} + (\mathcal{L}_1 + \mathcal{L}_2)u = f \tag{3.21}$$

Die Idee ist nun, eine "künstliche Zeitschrittweite" $\frac{\delta t_C}{2}$ einzuführen.
Aus

$$\frac{\partial u}{\partial t} = \frac{u\left(\cdot, t + \delta t_C\right) - u\left(\cdot, t\right)}{\delta t_C} \tag{3.22}$$

wird, indem wir $u\left(\cdot, t + \frac{\delta t_C}{2}\right)$ einfügen (Nulladdition):

$$\frac{\partial u}{\partial t} = \frac{u\left(\cdot, t + \delta t_C\right) - u\left(\cdot, t + \frac{\delta t_C}{2}\right) + u\left(\cdot, t + \frac{\delta t_C}{2}\right) - u\left(\cdot, t\right)}{\delta t_C}. \tag{3.23}$$

Indem wir dies in (3.21) einsetzen, erhalten wir:

$$\frac{u\left(\cdot, t + \delta t_C\right) - u\left(\cdot, t + \frac{\delta t_C}{2}\right) + u\left(\cdot, t + \frac{\delta t_C}{2}\right) - u\left(\cdot, t\right)}{\delta t_C} + (\mathcal{L}_1 + \mathcal{L}_2)u = f \tag{3.24}$$

Das lässt sich aufteilen in:

$$\begin{aligned} \frac{u\left(\cdot, t+\frac{\delta t_C}{2}\right) - u\left(\cdot, t\right)}{\delta t_C} + \mathcal{L}_1 u\left(\cdot, t+\frac{\delta t_C}{2}\right) &= 0 \\ \frac{u\left(\cdot, t+\delta t_C\right) - u\left(\cdot, t+\frac{\delta t_C}{2}\right)}{\delta t_C} + \mathcal{L}_2 u\left(\cdot, t+\delta t_C\right) &= f(\cdot, t) \end{aligned} \tag{3.25}$$

Das bedeutet in unserem Fall zum Beispiel für C_1 ausgeschrieben, dass zu lösen ist:

$$\frac{C_1(\cdot, t+\frac{\delta_{t_C}}{2}) - C_1(\cdot, t)}{\delta_{t_C}} - D_{C_1}\Delta C_1(\cdot, t+\frac{\delta_{t_C}}{2}) = 0 \tag{3.26}$$

$$\frac{C_1(\cdot, t+\delta_{t_C}) - C_1(\cdot, t+\frac{\delta_{t_C}}{2})}{\delta t} + (k_2 B + k_4) C_1(\cdot, t+\delta_{t_C}) - k_3 C_1^2(\cdot, t+\delta t) C_2(\cdot, t+\delta_{t_C}) = k_1 A \tag{3.27}$$

Mit einem impliziten Verfahren führt uns das also auf folgendes. Es muss zuerst berechnet werden:

$$C_1\left(\cdot, t+\frac{\delta_{t_C}}{2}\right) = C_1\left(\cdot, t\right) + \delta_{t_C}\mathcal{L}_1\left(\cdot, t+\frac{\delta_{t_C}}{2}\right) \tag{3.28}$$

Mit dem so berechneten Wert lösen wir dann:

$$C_1\left(\cdot, t+\delta_{t_C}\right) = C_1\left(\cdot, t+\frac{\delta_{t_C}}{2}\right) + \delta_{t_C}\mathcal{L}_2\left(\cdot, t+\delta_{t_C}\right) \tag{3.29}$$

Wir haben also allgemein:

(1) berechne:

$$C_i\left(\cdot, t+\frac{\delta_{t_C}}{2}\right) = C_i\left(\cdot, t\right) + \delta_{t_C}\mathcal{L}_1\left(\cdot, t+\frac{\delta_{t_C}}{2}\right) \tag{3.30}$$

(2) berechne:

$$C_i\left(\cdot, t+\delta_{t_C}\right) = C_i\left(\cdot, t+\frac{\delta_{t_C}}{2}\right) + \delta_{t_C}\mathcal{L}_2\left(\cdot, t+\delta_{t_C}\right) \tag{3.31}$$

für $i = 1, 2$.

3.2. Zeitdiskretisierung des Brusselators

Das bisher beschriebene System ist diskret nur im Ort, aber zur numerischen Lösung müssen wir auch noch eine Zeitdiskretisierung durchführen. Dazu kommen prinzipiell verschiedene Verfahren in Frage.

Sei das Anfangswertproblem (AWP)

$$\begin{aligned} \frac{du}{dt} &= f(t,u) \\ u(t_0) &= u_0 \end{aligned} \tag{3.32}$$

mit $t_0, u_0 \in \mathbb{R}$ gegeben.
Für eine fest gewählte Zeitschrittweite[1] $\delta_{t_C} \in \mathbb{R}^+$ teilen wir das Zeitintervall $[t_0, t_{end}]$ in die diskreten Zeitpunkte

$$t_k = t_0 + k \cdot \delta_{t_C} \qquad \text{für } k = 1, 2, \ldots k_{max} \quad ,$$

auf. Wir betrachten hier oBdA $t_0 = 0$.

Die erste und einfachste Wahl zur Diskretisierung ist das *explizite Euler-Verfahren* . Für (3.32) berechnet sich die diskreten Lösung u_{k+1} zum Zeitpunkt t_{k+1} durch:

$$u_{k+1} = u_k + \delta_{t_C} \cdot f(t_k, u_k) \tag{3.33}$$

Dieses ist einfach zu implementieren, jedoch nicht sehr stabil.
Gerade für steife Differentialgleichungen ist das problematisch. Um ein explizites Verfahren benutzen zu können müssen wir gewissen Stabilitätsbedingungen erfüllen, die sogenannten CFL-Bedingungen. Wir gehen in 3.7.3 noch genauer darauf ein, wie diese für unser problem aussehen. Grob gesagt stellen sie sicher, dass sich keine der zu berechnenden Größen pro Zeitschritt um mehr als eine Maschenweite verändern kann. Häufig führt dies dazu, dass bei einem expliziten Verfahren eine sehr kleine Zeitschrittweite gewählt werden muss. Das macht die Berechnungen jedoch langsam.

Eine Alternative bietet das *implizite Euler-Verfahren* :

$$u_{k+1} = u_k + \delta_{t_C} \cdot f(t_{k+1}, u_{k+1}) \tag{3.34}$$

Allerdings sind die Werte $f(t_{k+1}, u_{k+1})$ bei der Berechnung von u_{k+1} nicht bekannt. Man sagt daher, dass sie nicht explizit gegeben sind sondern nur implizit. Daher muss in jedem Rechenschritt ein Gleichungssystem gelöst werden. Deshalb ist der Rechenaufwand groß, jedoch bietet sich das Verfahren aufrgund seiner Stabilität für steife Gleichungssyteme an.

[1] δ_{t_C} kann zwischen den einzelnen Teilprobleme auch variieren, ist jedoch für jedes Teilproblem fest gewählt.

Mit der "künstlichen Zeitschrittweite" $\frac{\delta_{t_C}}{2}$ haben wir nun die Bezeichnungen:

$$\begin{aligned} t_k &\rightarrow C_k \\ t_k + \frac{\delta_{t_C}}{2} &\rightarrow C_{k+\frac{1}{2}} \\ t_{k+1} = (t_k + \delta_{t_C}) &\rightarrow C_{k+1} \end{aligned} \tag{3.35}$$

Mit diesen beiden Verfahren bieten sich folgende Möglichkeiten, wie wir sie zur Lösung des gesamten oder des gesplitteten Gleichungssystems kombinieren können:
(1) Gesamtes System: explizit lösen
(2) Gesamtes System: implizit lösen
(3) Splitting: Diffusion explizit, Reaktionsteil implizit lösen
(4) Splitting: Diffusion implizit, Reaktionsteil explizit lösen
(5) Splitting: Diffusion explizit, Reaktionsteil explizit lösen
(6) Splitting: Diffusion implizit, Reaktionsteil implizit lösen

Für (1) lautet das Verfahren:
Für $k = 0, 1, 2, \ldots$

$$\text{berechne} \quad C_{k+1}(\cdot, t_k + \delta_{t_C}) = C_k(\cdot, t_k) + \delta_{t_C} \cdot (MC_k(\cdot, t_k + \delta_{t_C}) + \vec{a} + b(C_k(\cdot, t_k + \delta_{t_C}))$$

Für (2), wobei das gesamte System implizit gelöst wird, haben wir:
Für $k = 0, 1, 2, \ldots$

$$\text{berechne} \quad C_{k+1}(\cdot, t_k + \delta_{t_C}) = C_k(\cdot, t_k) + \delta_{t_C} \cdot (MC_{k+1}(\cdot, t_{k+1} + \delta_{t_C}) + \vec{a} + b(C_{k+1}(\cdot, t_{k+1} + \delta_{t_C}))$$

Bei dem Splitting (3), welches den Diffusionsteil explizit, den Reaktionsteil aber implizit löst, berechnen wir:
Für $k = 0, 1, 2, \ldots$

$$\text{berechne} \quad C_{k+\frac{1}{2}}\left(\cdot, t_k + \frac{\delta_{t_C}}{2}\right) = C_k\left(\cdot, t_k\right) + \delta_{t_C} \cdot \left(M_1 C_k\left(\cdot, t_k\right) + \vec{a}_1\right)$$

$$C_{k+1}\left(\cdot, t_k + \delta_{t_C}\right) = C_{k+\frac{1}{2}}\left(\cdot, t_k + \frac{\delta_{t_C}}{2}\right) + \delta_{t_C} \cdot \left(M_2 C_{k+1}\left(\cdot, t_k + \delta_{t_C}\right) + \vec{a}_2 + b\left(C_{k+1}\left(\cdot, t_k + \delta_{t_C}\right)\right)\right)$$

Für (4) haben wir:
Für $k = 0, 1, 2, \ldots$

$$\text{berechne} \quad C_{k+\frac{1}{2}}\left(\cdot, t_k + \frac{\delta_{t_C}}{2}\right) = C_k\left(\cdot, t_k\right) + \delta_{t_C} \cdot \left(M_1 C_{k+\frac{1}{2}}\left(\cdot, t_k + \frac{\delta_{t_C}}{2}\right) + \vec{a}_1\right)$$

$$C_{k+1}\left(\cdot, t_k+\delta_{t_C}\right) = C_{k+\frac{1}{2}}\left(\cdot, t_k+\frac{\delta_{t_C}}{2}\right)+\delta_{t_C}\cdot\left(M_2 C_k\left(\cdot, t_k+\frac{\delta_{t_C}}{2}\right)+\vec{a}_2+b\left(C_{k+\frac{1}{2}}\left(\cdot, t_k+\frac{\delta_{t_C}}{2}\right)\right)\right)$$

Bei (5) verwenden wir ein Splitting, in dem beide Programmteile explizit berechnet werden:
Für $k = 0, 1, 2, \ldots$

$$\text{berechne} \quad C_{k+\frac{1}{2}}\left(\cdot, t_k + \frac{\delta_{t_C}}{2}\right) = C_k\left(\cdot, t_k\right) + \delta_{t_C} \cdot \left(M_1 C_k\left(\cdot, t_k\right) + \vec{a}_1\right)$$

$$C_{k+1}\left(\cdot, t_k+\delta_{t_C}\right) = C_{k+\frac{1}{2}}\left(\cdot, t_k+\frac{\delta_{t_C}}{2}\right)+\delta_{t_C}\cdot\left(M_2 C_k\left(\cdot, t_k+\frac{\delta_{t_C}}{2}\right)+\vec{a}_2+b\left(C_{k+\frac{1}{2}}\left(\cdot, t_k+\frac{\delta_{t_C}}{2}\right)\right)\right)$$

Als letztes haben wir mit (6) die Variante, die beide Teile implizit berechnet:
Für $k = 0, 1, 2, \ldots$

$$\text{berechne} \quad C_{k+\frac{1}{2}}\left(\cdot, t_k + \frac{\delta_{t_C}}{2}\right) = C_k\left(\cdot, t_k\right) + \delta_{t_C} \cdot \left(M_1 C_k\left(\cdot, t_k + \frac{\delta_{t_C}}{2}\right) + \vec{a}_1\right)$$

$$C_{k+1}\left(\cdot, t_k+\delta_{t_C}\right) = C_{k+\frac{1}{2}}\left(\cdot, t_k+\frac{\delta_{t_C}}{2}\right)+\delta_{t_C}\cdot(M_2 C_{k+1}\left(\cdot, t_k+\delta_{t_C}\right)+\vec{a}_2+b\left(C_{k+1}\left(\cdot, t_k+\delta_{t_C}\right)\right)$$

Wir haben auf eine Implementierung von (1) und (2) verzichtet, da wir durch das Operatorsplitting eine Entkopplung der Reaktions- und Diffusionsterme erreichen wollten. Für das gesplittete Gleichungssystem (3.17) haben wir (3) - (6) implementiert. In Kapitel 4 finden sich ausführliche Betrachtungen zur Eignung der Verfahren für einzelne Probleme.

Daneben wäre es auch noch möglich, andere Methoden zur Zeitdiskretisierung einzusetzen, zum Beispiel das Von-Heun-Verfahren oder das Crank-Nicholson-Verfahren.

Der Algorithmus für das gesplittete Verfahren lautet:

Algorithmus 2: Gesplittete Berechnung des Brusselators

Setze $t = 0$, $k = 0$
Setze Anfangswerte $C_1^k = C_1(0), C_2^k(0) = C_2(0)$
while $t \leq T_{end}$ **do**
  Berechne Zeitschrittweite δ_{t_C}**:**
  gemäß (3.69)
  Setze Randwerte:
  für $C_i^{k+\frac{1}{2}}(t_C + \frac{\delta t_C}{2})$
  Berechne Diffusion:
  bestimme $C_i^{k+\frac{1}{2}}(t_C + \delta_{t_C}/2)$
  Setze Randwerte:
  für $C_i^{k+1}(t_C + \delta t_C)$
  Berechne Reaktion:
  berechne $C_i^{l+1}(t_C + \delta_{t_C})$
  Setze $t = t + \delta t_C$, $k = k + 1$
end

3.3. Implementierung

Wir haben zunächst die Codes in einer 2-dimensionalen Version in *Matlab* implementiert, was ein einfacheres Testen ermöglichte, als wenn das Modell direkt, in einer 3D-Variante, an den NaSt-Code gekoppelt worden wäre.

Da wir wie beschrieben ein Operatorsplitting einsetzen wollen, haben wir einzeln den Diffusionsanteil und den Reaktionsanteil implementiert. Beide wollen wir nun im Detail beschreiben.

3.4. 2D-Wärmeleitungsgleichung in MATLAB

Wir lösen nun auf Ω die *Diffusionsgleichung* oder auch *Wärmeleitungsgleichung*:

$$\begin{aligned} \frac{\partial C(\vec{x},t)}{\partial t} &= D_C \Delta C(\vec{x},t) \qquad \forall \vec{x} \in \Omega, t \in [0, t_{end}] \\ C(\vec{x},t) &= f \qquad \forall \vec{x} \in \Gamma\Omega, t \in [0, t_{end}] \\ C(\vec{x},0) &= C_0(\vec{x}) \qquad \forall \vec{x} \in \Omega \end{aligned} \tag{3.36}$$

Diese beschreibt den diffusiven Transport des skalaren C.

Im Rahmen des Operatorsplittings gemäß 3.1.4 behandeln wir damit den Term $\mathcal{L}_1$.

3.4.1. Explizit

Mit einem einfachen expliziten Euler-Verfahren, welches oben bereits beschrieben wurde, erhalten wir:

für $i = 2 \dots imax + 1$
für $j = 2 \dots jmax + 1$

berechne für $k = 1, 2, \dots$:

$$C^{k+1}(x_{ij}) = C^k(x_{ij}) + \delta_{t_C} \cdot \Delta_h C^{k+1}(x_{ij}), \tag{3.37}$$

also:

$$\begin{aligned} C^{k+1}(x_{ij}) = C^k(x_{ij}) \\ + \delta_{t_C} \cdot \frac{D_C}{h^2} \left(C^k(x_{i(j+1)}) + C^k(x_{i(j-1)}) + C^k(x_{(i+1)j}) + C^k(x_{(i-1)j}) - 4C^k(x_{ij}) \right) \end{aligned} \tag{3.38}$$

Dabei kommen zur Berechnung von C^{k+1} auf der rechten Seite für $i = 2$, $i = imax + 1$, $j = 2$ oder $j = jmax+1$ Werte von C^k vor, die auf dem Rand $\Gamma\Omega$ liegen. Diese sind gemäß den Randbedingungen aus (3.36) bekannt und werden entsprechend eingesetzt. Deshalb enthält unser Diskretisierungsgitter 3.1, auch wenn wir nur auf den inneren Punkten rechnen, auch Gitterpunkte auf dem Rand bereit, in denen die Randwerte gespeichert sind. Die Werte C^k auf den inneren Gitterpunkte sind bekannt aus dem jeweils letzten Schritt beziehungsweise für $k = 1$ als die Anfangskonzentrationen C_0, so dass wir die Werte C^{k+1} unmittelbar ausrechnen können.

3.4.2. Implizit

Schwieriger zu implementieren ist das implizite Verfahren.

Hier müssen wir lösen:
für $i = 2 \dots imax + 1$
für $j = 2 \dots jmax + 1$

berechne für $k = 1, 2, \dots$:

$$\begin{aligned} C^{k+1}(x_{ij}) = C^k(x_{ij}) \\ + \delta_{t_C} \cdot D_C \frac{C^{k+1}(x_{i(j+1)}) + C^{k+1}(x_{i(j-1)}) + C^{k+1}(x_{(i+1)j}) + C^{k+1}(x_{(i-1)j}) - 4C^{k+1}(x_{ij})}{h^2} \end{aligned} \tag{3.39}$$

Es kommen also für C^{k+1} auch auf der rechten Seite die Größen C^{k+1} vor und es ergibt sich somit ein Gleichungssystem, das zu lösen ist, wobei die Randbedingungen gemäß (3.36) einzuhalten sind. Für C_1 gehen wir folgendermaßen vor (C_2 analog):

Wie schon in (3.11) schreiben wir das Konzentrationsfeld $C_1(\vec{x}, t)$, das wir normalerweise als Matrix notieren würden, in einer lexikografischen Reihenfolge in einen Vektor:

$$\vec{C_1}(t) = \begin{pmatrix} C_1(x_{11}, t) \\ C_1(x_{12}, t) \\ \vdots \\ C_1(x_{N(N-1)}, t) \\ C_1(x_{NN}, t) \end{pmatrix} \tag{3.40}$$

Dabei nehmen wir nur die $N \cdot N$ inneren Punkte, für die das System zu lösen ist.
Wir wollen also lösen:

$$M \cdot \vec{C_1} = b, \tag{3.41}$$

wobei wir noch die Systemmatrix M und die rechte Seite b aufstellen müssen.

Wie in 3.1.2 ergibt sich M zu einer Blocktridiagonalmatrix:

$$M = \begin{pmatrix} 1+4s & -s & & & \\ -s & \ddots & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & -s \\ & & & -s & 1+4s \end{pmatrix} \quad \in \mathbb{R}^{\cdot N \cdot N} \times \mathbb{R}^{\cdot N \cdot N}, \tag{3.42}$$

wobei wir

$$s := \frac{D_{C_1}}{h^2}$$

gesetzt haben. Die rechte Seite ergibt sich aus den Werten C_1^k und den Randwerten. Analaog gehen wir für C_2 vor, hier ändert sich nur die Diffusionskonstante.

3.5. 2D-Brusselator in MATLAB

3.5.1. Explizites Verfahren zur Zeitdiskretisierung der Reaktion

Wir wollen mit einem expliziten Euler-Verfahren den Reaktionsteil des Brusselators lösen:

Für $x_{ij} \in \Omega_h$:

$$\frac{d}{dt}\begin{pmatrix} C_1(x_{ij},t) \\ C_2(x_{ij},t) \end{pmatrix} = \begin{Bmatrix} k_1 A - k_2 B C_1(x_{ij}) + k_3 C_1(x_{ij})^2 C_2(x_{ij}) - k_4 C_1(x_{ij}) \\ k_2 B C_1(x_{ij}) - k_3 C_1(x_{ij})^2 C_2(x_{ij}) \end{Bmatrix} \tag{3.43}$$

Für $t_k = t_0 + k \cdot \delta_{t_C}$ für $k = 1, 2, \ldots$ und $C^k(x_{ij}) := C(x_{ij}, t_k)$ lautet das Verfahren:
Berechne für $k = 1, 2, \ldots$

$$\begin{aligned} \begin{pmatrix} C_1^{k+1}(x_{ij}) \\ C_2^{k+1}(x_{ij}) \end{pmatrix} = & \begin{pmatrix} C_1^{k}(x_{ij}) \\ C_2^{k}(x_{ij}) \end{pmatrix} \\ & + \delta_{t_C} \cdot \begin{Bmatrix} k_1 A - k_2 B C_1^k(x_{ij}) + k_3 C_1^k(x_{ij})^2 C_2^k(x_{ij}) - k_4 C_1^k(x_{ij}) \\ k_2 B C_1^k(x_{ij}) - k_3 C_1^k(x_{ij})^2 C_2^k(x_{ij}) \end{Bmatrix} \end{aligned} \tag{3.44}$$

Dieses können wir direkt implementieren. Das explizite Euler Verfahren ist eines der einfachsten, jedoch nicht sehr stabil.

3.5.2. Implizites Verfahren zur Zeitdiskretisierung der Reaktion

Wir wollen nun genauer die Implemetierung eines impliziten Verfahrens zur Lösung des Teilproblems der Reaktion, also (3.19),

$$\frac{d}{dt}C(x_{ij},t) = M_2 C(x_{ij},t) + \vec{a_2} + b(C_0(x_{ij},t), C_0(x_{ij},t), C_1(x_{ij},t)) \qquad \text{für } x \in \Omega_h, \quad t \in (0, t_{end}]$$

In der ursprünglichen, matrixfreien, Schreibweise lautete die Gleichung:

$$\frac{d}{dt}\begin{pmatrix} C_1(x_{ij},t) \\ C_2(x_{ij},t) \end{pmatrix} = \begin{Bmatrix} k_1 A - k_2 B C_1(x_{ij}) + k_3 C_1(x_{ij})^2 C_2(x_{ij}) - k_4 C_1(x_{ij}) \\ k_2 B C_1(x_{ij}) - k_3 C_1(x_{ij})^2 C_2(x_{ij}) \end{Bmatrix} \tag{3.45}$$

Den Reaktionsterm auf der rechten Seite wollen wir wieder mit $g_3(C(x_{ij}))$ bezeichnen. Wir haben hier, um flexibler in der Wahl der Parameter zu sein, auch die Paramter k_i als frei wählbar implementiert und lassen die häufige Annahme, dass $k_1 = k_2 = k_3 = k_4 = 1$ gilt, vorerst unberücksichtigt.

Wir bezeichnen für die Zeitschritte $t_k = t_0 + k \cdot \delta_{t_C}$ für $k = 1, 2, \ldots$ mit

$$C^k(x_{ij}) := C(x_{ij}, t_k)$$

die diskrete Lösung des k-ten Zeitschrittes t_k.
Das implizite Euler-Verfahren für (3.47) lautet nun:
Berechne für $k = 1, 2, \ldots$

$$\begin{pmatrix} C_1^{k+1}(x_{ij}) \\ C_2^{k+1}(x_{ij}) \end{pmatrix} = \begin{pmatrix} C_1^{k}(x_{ij}) \\ C_2^{k}(x_{ij}) \end{pmatrix} \tag{3.46}$$

$$+ \delta_{t_C} \cdot \begin{Bmatrix} k_1 A - k_2 B C_1^{k+1}(x_{ij}) + k_3 C_1^{k+1}(x_{ij})^2 C_2^{k+1}(x_{ij}) - k_4 C_1^{k+1}(x_{ij}) \\ k_2 B C_1^{k+1}(x_{ij}) - k_3 C_1^{k+1}(x_{ij})^2 C_2^{k+1}(x_{ij}) \end{Bmatrix} \tag{3.47}$$

Das implizite Euler-Verfahren ist stabiler, jedoch auch aufwändiger in der Berechnung. Hier ergibt sich in jedem Zeitschritt ein Gleichungssystem, das gelöst werden muss. In unserem Fall ist dieses nichtlinear, was einen besonderen Aufwand erfordert.

Wir verwenden zur Lösung des nichtlinearen Systems ein *Newton-Verfahren.*

Wir können (3.47) umschreiben:

$$0 = \begin{pmatrix} C_1^{k+1}(x_{ij}, t) \\ C_2^{k+1}(x_{ij}, t) \end{pmatrix} + \begin{pmatrix} C_1^{k}(x_{ij}, t) \\ C_2^{k}(x_{ij}, t) \end{pmatrix} \tag{3.48}$$

$$+ \delta_{t_C} \cdot \begin{Bmatrix} k_1 A - k_2 B C_1^{k+1}(x_{ij}) + k_3 C_1^{k+1}(x_{ij})^2 C_2^{k+1}(x_{ij}) - k_4 C_1^{k+1}(x_{ij}) \\ k_2 B C_1^{k+1}(x_{ij}) - k_3 C_1^{k+1}(x_{ij})^2 C_2^{k+1}(x_{ij}) \end{Bmatrix} \tag{3.49}$$

Somit wird unsere Aufgabe dazu, die Nullstellen des Terms auf der rechten Seite zu bestimmen. Dies müssen wir allerdings in *jedem Punkt* x_{ij} tun.

Für $F = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$ bezeichnen wir:

$$F \begin{pmatrix} C_1^{k+1}(x_{ij}, t) \\ C_2^{k+1}(x_{ij}, t) \end{pmatrix} := \begin{pmatrix} C_1^{k+1}(x_{ij}, t) \\ C_2^{k+1}(x_{ij}, t) \end{pmatrix} + \begin{pmatrix} C_1^{k}(x_{ij}, t) \\ C_2^{k}(x_{ij}, t) \end{pmatrix} \tag{3.50}$$

$$+ \delta_{t_C} \cdot \begin{Bmatrix} k_1 A - k_2 B C_1^{k+1}(x_{ij}) + k_3 C_1^{k+1}(x_{ij})^2 C_2^{k+1}(x_{ij}) - k_4 C_1^{k+1}(x_{ij}) \\ k_2 B C_1^{k+1}(x_{ij}) - k_3 C_1^{k+1}(x_{ij})^2 C_2^{k+1}(x_{ij}) \end{Bmatrix}, \tag{3.51}$$

wobei $F : \mathbb{R}^{2 \cdot N \cdot N} \to \mathbb{R}^{2 \cdot N \cdot N}$ eine diskrete Funktion ist.
Es gilt also nun zu lösen:

$$F \begin{pmatrix} C_1^{k+1}(x_{ij}, t) \\ C_2^{k+1}(x_{ij}, t) \end{pmatrix} = 0 \tag{3.52}$$

Newtonverfahren

Dafür verwenden wir ein mehrdimensionales *Newtonverfahren* . Es funktioniert nach der Iterationsvorschrift:
für $it = 1, 2, \ldots itmax$ *oder Abbruchkriterium*

$$x^{it+1} := x^{it} + \Delta x^{it} \tag{3.53}$$

$$\text{mit} \quad \Delta x^{it} = -F(x^{it}) \cdot J_F^{-1}(x^{it}), \tag{3.54}$$

wobei $itmax \in \mathbb{N}$ eine Schranke für die maximale Anzahl der Iterationen ist. Als weiteres Abbruchkriterium sollte eine Fehlerschätzung verwendet werden, zum Beispiel

$$x^{k+1^{it}}(x_{ij}) - x^{k+1^{it-1}}(x_{ij}) < TOL \tag{3.55}$$

Wird der Wert auf der linken Seite sehr klein, verändert sich die iterierte Lösung nicht mehr stark und wenn eine gewisse Toleranzgrenze unterschritten ist, kann abgebrochen werden.
Für das Verfahren müssen wir also die *Jacobi-Matrix* J_F bestimmen.
Es ist allgemein allgemein, für eine Funktion $h : \mathbb{R}^m \to \mathbb{R}^n$ die Jacobi-Matrix J_h an einem Punkt $a \in R^m$, definiert als:

$$J_h(a) := \begin{pmatrix} \frac{\partial h_1}{x_1}(a) & \frac{\partial h_1}{x_2}(a) & \ldots & \frac{\partial h_1}{x_n}(a) \\ \vdots & \vdots & \ldots & \vdots \\ \frac{\partial h_m}{x_1}(a) & \frac{\partial h_m}{x_2}(a) & \ldots & \frac{\partial h_m}{x_n}(a) \end{pmatrix} \tag{3.56}$$

In unserem Fall also:

$$J_F(x_{ij}) = \begin{pmatrix} \frac{\delta f_1}{\delta x_1}(x_{ij}) & \frac{\delta f_1}{\delta x_2}(x_{ij}) \\ \frac{\delta f_2}{\delta x_2}(x_{ij}) & \frac{\delta f_2}{\delta x_2}(x_{ij}) \end{pmatrix} \tag{3.57}$$

Für das durch (3.51) gegebene F gilt:

$$J_F \begin{pmatrix} C_1^{k+1^{it}}(x_{ij}) \\ C_2^{k+1^{it}}(x_{ij}) \end{pmatrix} = \tag{3.58}$$

$$= \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} + \delta_{t_C} \cdot \begin{pmatrix} -k_2 B - k_4 + 2k_3 C_1^{k+1^{it}}(x_{ij}) C_2^{k+1^{it}}(x_{ij}) & k_3 C_1^{k+1^{it}}(x_{ij})^2 \\ k_2 B - 2k_3 C_1^{k+1^{it}}(x_{ij}) C_2^{k+1^{it}}(x_{ij}) & -k_3 C_1^{k+1^{it}}(x_{ij})^2 \end{pmatrix} \tag{3.59}$$

Hier bezeichnet $C^{k^{it}}(x_{ij})$ die aktuelle Iterierte innerhalb einer Schleife des Newtonverfahrens im k-ten Zeitschritt.

Wir benötigen die Inverse J_F^{-1}. Da wir - für 2 chemische Spezies - eine 2×2 Matrix haben, kann diese direkt invertiert werden nach der Formel:

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix}^{-1} = \frac{1}{ad - cb} \begin{pmatrix} d & -c \\ -b & a \end{pmatrix}$$

Im Falle dass $N > 2$ gilt und JF also größer ist muss hier allerdings auch wieder ein Gleichungssystem gelöst werden.

Algorithmus - Zusammenfassung

Insgesamt lautet also unser Verfahren:

Algorithmus 3: Implizite Berechnung der Reaktion (2D).

Input : $C_1^k(x_{ij}), C_2^k(x_{ij})$

setze $k := 0$, $it := 0$

while $it < itmax$ *oder Abbruchkriterium* **do**

  for $i := 1$ **to** N **do**

    for $j := 1$ **to** N **do**

      löse $\quad J_F \begin{pmatrix} C_1^{k+1^{it}}(x_{ij}) \\ C_1^{k+1^{it}}(x_{ij}) \end{pmatrix} s^{it} = -F \begin{pmatrix} C_2^{k+1^{it}}(x_{ij}) \\ C_2^{k+1^{it}}(x_{ij}) \end{pmatrix},$

      dazu setze:

      $\begin{pmatrix} C_1^{k+1^{it+1}}(x_{ij}) \\ C_2^{k+1^{it+1}}(x_{ij}) \end{pmatrix}$

      $= \begin{pmatrix} C_1^{k+1^{it}}(x_{ij}) \\ C_2^{k+1^{it}}(x_{ij}) \end{pmatrix} - \left\{ F_{Disk} \begin{pmatrix} C_1^{k+1^{it}}(x_{ij}) \\ C_2^{k+1^{it}}(x_{ij}) \end{pmatrix} \cdot JF_{Disk}^{-1} \begin{pmatrix} C_1^{k+1^{it}}(x_{ij}) \\ C_2^{k+1^{it}}(x_{ij}) \end{pmatrix} \right\}$

    end

  end

  setze it = it+1;

end

3.6. Erweiterung des Brusselators auf 3D

Zur Erweiterung unserer Implementierung auf drei Dimensionen gehen wir analog wie bei dem 2D-Fall vor.

Die Berechnung der Reaktion ändert sich im Prinzip nicht, weil sie bei Reaktions-Diffusions-Modellen lokal stattfindet. Wir müssen nun nur bei der Ortsdiskretisierung in noch einer weiteren Raumrichtung diskretisieren, verwenden also ein dreidimensionales Gitter. Im Code erhalten wir dadurch noch eine weitere Schleife beim Durchgehen der Punkte. Das bedeutet natürlich, dass sich der Rechenaufwand erhöht.

Für die Berechnung des Diffusionsteils können wir auf eine bereits vorhandene skalare Transportroutine von NaSt3D zurückgreifen. Für den 3D-Fall lautet dann unser Algorithmus, mit der impliziten Methode:

Algorithmus 4: Implizites Verfahren für Brusselator in 3D

setze $k := 0$, $it := 0$
while $it < itmax$ *oder Abbruchkriterium* **do**
  for $i := 1$ **to** N **do**
    for $j := 1$ **to** N **do**
      for $k := 1$ **to** N **do**
        löse $\quad J_F \begin{pmatrix} C_1^{k+1^{it}}(x_{ijk}) \\ C_1^{k+1^{it}}(x_{ijk}) \end{pmatrix} s^{it} = -F \begin{pmatrix} C_2^{k+1^{it}}(x_{ijk}) \\ C_2^{k+1^{it}}(x_{ijk}) \end{pmatrix},$
        dazu setze:
        $\begin{pmatrix} C_1^{k+1^{it+1}}(x_{ijk}) \\ C_2^{k+1^{it+1}}(x_{ijk}) \end{pmatrix}$
        $= \begin{pmatrix} C_1^{k+1^{it}}(x_{ijk}) \\ C_2^{k+1^{it}}(x_{ijk}) \end{pmatrix} - \left\{ F_{Disk} \begin{pmatrix} C_1^{k+1^{it}}(x_{ijk}) \\ C_2^{k+1^{it}}(x_{ijk}) \end{pmatrix} \cdot JF_{Disk}^{-1} \begin{pmatrix} C_1^{k+1^{it}}(x_{ijk}) \\ C_2^{k+1^{it}}(x_{ijk}) \end{pmatrix} \right\}$
      end
    end
  end
  setze it = it+1;
end

Wir verzichten hier darauf, das explizite Verfahren nochmals in 3D anzugeben, die Erweiterung des 2D-Codes hierfür geschieht analog durch eine Anpassung der Ortsdiskretisierung.

3.7. Kopplung des Brusselators an NaSt3DGPF

3.7.1. Kopplungsmodell

Wir haben in Kapitel 2.4 den Algorithmus zur Strömungssimulation beschrieben und uns schließlich in den vorangegangenen Kapiteln mit dem in dieser Arbeit umgesetzten Reaktionsmodell selbst befasst. Es gilt nun, diese beiden Programmteile zu verbinden.
Auf die Durchführung dieser sogenannten Kopplung wollen wir in diesem Kapitel eingehen. Es gibt dabei verschiedenes zu beachten und mehrere Möglichkeiten, die man in Betracht ziehen kann.

Zunächst wollen wir unterscheiden zwischen einer *aktiven* Kopplung und einer *passiven* Kopplung. Dabei verstehen wir unter der passiven Kopplung, dass die regierenden Spezies mit der Strömung bewegt werden. Es wirkt sich also die Strömung auf die Reaktion aus, nicht jedoch umgekehrt. Damit sind wir bei der aktiven Kopplung: hier entsteht auch eine Einwirkung der Reaktion auf die Strömung. Man spricht auch von Rückkopplung.
Prinzipiell ist beides auch einzeln möglich. In dieser Arbeit soll sowohl die aktive als auch die passive Kopplung simuliert werden. Der aktive Teil ist dabei der schwierige, da es eine offene Frage ist, wie man dies für ein Reaktions-Diffusions-Modell wie den Brusselator auf eine physikalisch beziehungsweise chemisch richtige Weise bewerkstelligen kann.
Wir gehen im folgenden noch auf unser eigenes Modell hierzu ein.

Wie schon erklärt, werden wir den Reaktions- und den Diffusionsteil des Brusselators gemäß einer Splittingmethode getrennt voneinander lösen. Dies ermöglicht uns insbesondere, für die Diffusion auf eine im NaSt3D-Code bereits vorhandene Routine zum Transport von allgemeinen skalaren Feldern zurückzugreifen.

Die Hauptkomponenten, die wir zu verbinden haben, sind also: das Lösen der Impulsgleichung zur Bestimmung der Schätzgeschwindigkeiten, der diffusive Transport der Konzentrationsfelder, die Berechnung des Reaktionseinflusses, das Lösen der Druckgleichung und die Korrektur der Geschwindigkeiten.
Dabei wird zur Berechnung der Reaktion der eigene, in dieser Arbeit entwickelte Algorithmus eingesetzt.
Besonders muss beachtet werden, welcher Wert gerade an welchem Zeitschritt berechnet wird, und zu welchem Zeitschritt die Werte, von denen dieser abhängt gehören, so dass das Vorgehen konsistent ist.

Wir wollen ein boussinesqartiges Modell zur Kopplung von Chemie und Strömung verwenden. Dieses ist also angelehnt an das Boussinesq-Verfahren der Kopplung einer Temperaturgleichung an die Navier-Stokes-Gleichungen, welches etwa in [GDN98] beschrieben wurde. Dafür gehen wir folgendermaßen vor:

$$\partial_t \vec{u} + (\vec{u} \cdot \nabla_x)\, \vec{u} = \frac{\mu}{\rho} \nabla_x \cdot \left(\nabla_x \vec{u} + \{\nabla_x \vec{u}\}^T \right) - \frac{1}{\rho} \nabla_x p + \varrho(\mathbf{C_1}, \ldots, \mathbf{C_N}) \vec{G} \tag{3.60}$$

$$\partial_t C_i - D_{C_i}\Delta C_i + (u \cdot \nabla_x)\, C_i = \mathbf{Q_i}\left(\mathbf{C_1}, \ldots, \mathbf{C_N}\right) \qquad \forall i = 1 \ldots N \tag{3.61}$$

$$\nabla_x \cdot \vec{u} = 0 \tag{3.62}$$

Wir erweitern also die Impulsgleichung (2.11) um den Term:

$$\varrho(C_1, \ldots, C_N).$$

(3.62) ist die unveränderte Kontinuitätsgleichung (2.12). Dem System aus modfizierter Impulsgleichung und Kontinuitätsgleichung wurden noch die N Reaktions-Diffusionsgleichungen (3.61) hinzugefügt. Darin beschreibt der Quellterm

$$Q_i\left(C_1, \ldots, C_N\right) \qquad \forall i = 1 \ldots N$$

jeweils die Reaktion.

In unserem Fall für das Brusselator-Modell ist $N = 2$ und die Reaktionsquellterme Q_i haben die Gestalt:

$$\begin{aligned} Q_1(C_1, C_2) &= k_1 A - k_2 B C_1 + k_3 C_1^2 C_2 - k_4 C_1 \\ Q_2(C_1, C_2) &= C_2 + k_2 B C_1 - k_3 C_1^2 C_2 \end{aligned} \tag{3.63}$$

Der Term $\varrho(C_1, \ldots, C_N)$ führt die aktive Kopplung des Chemiemodells an die Navier-Stokes-Gleichungen ein. Dieser hat eine boussinesqartige Gestalt. Wir verwenden:

$$\varrho(C_1, \ldots, C_N) = (1 - \gamma_1 C_1^{ref}), \tag{3.64}$$

wobei γ_1 eine thermodynamische Konstante in Bezug auf C_1 ist, und C_1^{ref} eine Referenzkonzentration für C_1.

Zur Umsetzung des Modelles müssen wir nun noch die in (2.19) beschriebene Berechnung des temporären Geschwindigkeitsfeldes so modfizieren, dass unser Chemie-Boussinesq-Term eingebunden wird. Damit werden die diskreten Impulsgleichungen zu:

$$\begin{aligned} \tilde{F}_{i,j,k}^{(n)} &:= F_{i,j,k}^{(n)} - \gamma_1 \frac{\delta t}{2} \left(C_1(x_{ijk})^{(n+1)} + C_1(x_{ijk})^{(n+1)} \right) G_x \\ \tilde{G}_{i,j,k}^{(n)} &:= G_{i,j,k}^{(n)} - \gamma_1 \frac{\delta t}{2} \left(C_1(x_{ijk})^{(n+1)} + C_1(x_{ijk})^{(n+1)} \right) G_y \\ \tilde{H}_{i,j,k}^{(n)} &:= H_{i,j,k}^{(n)} - \gamma_1 \frac{\delta t}{2} \left(C_1(x_{ijk})^{(n+1)} + C_1(x_{ijk})^{(n+1)} \right) G_z \end{aligned} \tag{3.65}$$

Damit haben wir eine Kopplung an das Geschwindigkeitsfeld errreicht.
Wir wollen abschließend noch betonen, dass es sich hierbei um ein modellhaftes Vorgehen handelt. Eine passende Wahl der Parameter ist noch eine offene Frage, die in dieser Arbeit ohne weiteres umfassendes Wissen über Chemie nicht abschließend geklärt werden konnte.

Für unsere in Kapitel 4 vorgestellten Berechnungen haben wir gute Resultate mit folgenden Parametern erhalten:

$\gamma_1 = 3.8$

$C_1^{ref} = 0$

Im Anhang finden sich noch weitere Simulationsresultate für andere Parameter.

3.7.2. Zeitskalen für Chemie und Strömung

Es ist möglich, dass eine Strömung und eine chemische Reaktion unterschiedlich schnell ablaufen. Insbesondere gibt es auch sehr langsame oder sehr schnelle Reaktionen. Wir betrachten zwar mit unserem Modell keinen von diesen Extremfällen: bei der BZR (bei Raumtemperatur) läuft ein Oszillationszyklus in ca. 20-30 Sekunden ab (siehe [Con82]). Dennoch soll es prinzipiell möglich sein, das Chemiemodell u d die Strömung mit verschiedenen Zeitschrittweiten zu berechnen. Dies kann für einige Fälle interessant sein.

Deshalb haben wir für die Zeitdiskretisierung des Brusselator Raktions-und Diffusionsmodelles eine andere Zeitschrittweite, δt_C verwendet als für die Diskretisierung der Strömung, δt_S.
Konkret wirkt sich das darin aus, dass wir in einem Zeitschritt der Strömungsberechnung ein- oder mehrmals die Simulation der Reaktion durchführen, falls $\delta t_S \leq \delta t_C$. Der Reaktions-Algorithmus wird dann in einer Schleife entsprechend oft aufgerufen. Falls $\delta t_S > \delta t_C$ gewählt ist, wird nicht in jedem Strömungsschritt die Reaktion durchgeführt, sondern erst wieder wenn δt_C viel Zeit vergangen ist.

3.7.3. CFL-Bedingungen für das Chemiemodell

Wir müssen zur Stabilität noch CFL-Bedingungen berücksichtigen. In 2.4.2 wurde bereits beschrieben, wie die Stabilitätsbedingungen für den Strömungscode allein aussehen. Wir werden diese Beschränkungen an die Strömungs-Zeitschrittweite δt_S noch ergänzen um eine Bedingung, die sich aus unserem Kopplungsmodell ergibt. Desweiteren muss auch die Chemie-Zeitschrittweite δt_C bestimmte Konditionen erfüllen.
Grundsätzlich stellen sich zwei Arten von Bedingungen, einmal an die diffusiven und einmal an die konvektiven Terme.

Diffusive Terme

Da wir die zu lösenden ODEs in Diffusion und Reaktion aufgesplittet haben, haben wir zwei Teile, aus denen CFL-Bedingungen entstehen können.

(1) Diffusion
Da bei unserem Vorgehen die Diffusion immer explizit gelöst wird, müssen wir immer die CFL-Bedingung für skalaren Transport erfüllen:

$$\delta t_C := \min_{\Omega} \left(\delta t_C, \tau \cdot \frac{1}{D_{C_i}} \cdot \frac{1}{2} \cdot \left(\frac{1}{x_{min}^2} + \frac{1}{y_{min}^2} + \frac{1}{z_{min}^2} \right)^{-1} \right), \tag{3.66}$$

wobei $\tau \in (0, 1]$ ein Sicherheitsfaktor ist (vgl. [GDN98], [Cro02]).
Diese Bedingung müssen wir immer dann berücksichtigen, wenn Transport der Chemikalien stattfindet, wenn also $D_{C_i} \neq 0$ für $i \in 1, 2$ gilt.

(2) Reaktion
Hinzu kommt, bei Einsatz des expliziten Verfahrens für die Reaktionsberechnung die Bedingung:

$$\frac{C_i(\vec{x}, t) \cdot \delta t_C}{\delta x} < 1 \qquad \forall \vec{x} \in \Omega, t \in [0, T_{end}], i \in 1, 2$$

für die x-Richtung und analog auch in die y- und z-Richtung.

Daraus ergibt sich:

$$\delta t_C < \min_{\Omega} \left(\frac{\delta x}{|C_{i,max}|}, \frac{\delta y}{|C_{i,max}|}, \frac{\delta z}{|C_{i,max}|} \right) \qquad \text{für } i \in 1, 2 \tag{3.67}$$

Für das implizite Verfahren benötigen wir an dieser Stelle keine CFL-Bedingung.

Konvektive Terme

Es ist zu beachten, dass wir außerdem, in dem Fall, dass eine Rückkopplung auf das Strömungsfeld besteht, den Einfluß der Kopplung in die Stabilitätsbedingung für die konvektiven Terme miteinberechnen müssen. Die Kopplung geschieht bei uns über die Volumenkräfte. Die Erweiterung der CFL-Bedingung um die Volumenkräfte wurde in [Cro02] ausgeführt. Wir gehen ebenso vor, nur dass bei uns die Kräfte $\vec{G}$ die modifizierte Form $\vec{G}(1 - \gamma_1(C_1 - C_1^{ref})$ haben.
Wir schreiben $\vec{G} = \begin{pmatrix} g_1 \\ g_2 \\ g_3 \end{pmatrix}$.

Damit ergibt sich aus den CFL-Bedingungen (2.30) :

$$\delta t_S^u \leq 2 \left(\left(\frac{|u_{max}|}{\delta x} + V \right) + \sqrt{\left(\frac{|u_{max}|}{\delta x} + V \right)^2 + \frac{4|g_1|(1 - \gamma_1(|C_{1,max}| - C_1^{ref})}{\delta x}} \right)^{-1},$$

für die x-Richtung, wobei wir die Abkürzung

$$V := \frac{\mu}{\rho}\left(\frac{2}{(\delta x)^2} + \frac{2}{(\delta y)^2} + \frac{2}{(\delta z)^2}\right)$$

verwendet haben.

Entsprechend haben wir für die v- und w-Komponenten:

$$\delta t_S^v \leq 2\left(\left(\frac{|v_{max}|}{\delta y} + V\right) + \sqrt{\left(\frac{|v_{max}|}{\delta y} + V\right)^2 + \frac{4|g_2|(1-\gamma_1(|C_{1,max}| - C_1^{ref})}{\delta y}}\right)^{-1},$$

$$\delta t_S^w \leq 2\left(\left(\frac{|w_{max}|}{\delta z} + V\right) + \sqrt{\left(\frac{|w_{max}|}{\delta z} + V\right)^2 + \frac{4|g_3|(1-\gamma_1(|C_{1,max}| - C_1^{ref})}{\delta z}}\right)^{-1},$$

Und es ergibt sich

$$\delta t_S \leq \tau \min_{\Omega}(\delta t_S^u, \delta t_S^v, \delta t_S^w) \tag{3.68}$$

als gesamte Stabilitätsbedingung für die konvektiven Terme unter Berücksichtigung der reaktionsbehafteten Volumenkräfte.

Wir fassen nochmal zusammen:

$$\begin{aligned}
&\text{Falls Diffusion:} && \delta t_C \leq \min_{\Omega}\left(\delta t_C, \tau \cdot \frac{1}{D_{C_i}} \cdot \frac{1}{2} \cdot \left(\frac{1}{x_{min}^2} + \frac{1}{y_{min}^2} + \frac{1}{z_{min}^2}\right)^{-1}\right) \\
&&& \text{für } i \in 1,2 \\
&\text{Falls Reaktion explizit:} && \delta t_C < \min_{\Omega}\left(\frac{\delta x}{|C_{i,max}|}, \frac{\delta y}{|C_{i,max}|}, \frac{\delta z}{|C_{i,max}|}\right) \\
&&& \text{für } i \in 1,2 \\
&\text{Falls Kopplung:} && \delta t_S \leq \tau \min_{\Omega}(\delta t_S^u, \delta t_S^v, \delta t_S^w)
\end{aligned} \tag{3.69}$$

3.7.4. Gekoppelter Algorithmus

Die Implementierung des Reaktions-Quellterms wurde oben ausführlich beschrieben. Die geeignete Erweiterung des Strömungscodes um die neuen Bestandteile haben wir soeben beschrieben. Insgesamt ergibt sich folgender Algorithmus. Dabei handelt es sich um eine Kombination von dem ursprünglichen NaSt-Algorithmus 1 und 2, wobei zusätzlich das unter 3.7 beschriebene Kopplungsmodell verwandt wird.

Es ist auch möglich, das Programm in der nur passiv gekoppelten Variante zu nutzen - dann wird in der Berechnung der Schätzgeschwindigkeiten der Chemie-Kopplungsterm deaktiviert und als CFL-Bedingung für δt_S gilt die normale NaSt-CFL-Bedingung ohne Kopplungseinfluß.

Algorithmus 5: NaSt mit aktiv gekoppelter Reaktion

Setze $t = 0$, $k = 0$
Setze Anfangswerte $u^k = u_0$, $p^k = p_0, C_1^k = C_1(0), C_2^k(0) = C_2(0)$
while $t \leq T_{end}$ **do**
  Berechne Zeitschrittweite (Strömung) δ_{t_S}**:**
  gemäß (3.69)
  Setze Randwerte:
  für u^{k+1}, p^{k+1} gemäß Abschnitt 2.4.2
  Berechne Reaktion:
  setze Chemie-Zeitschritt $t_C = t$
  setze $l = 0$
  while $t_C \leq (t + \delta_{t_S})$ **do**
    Berechne Zeitschrittweite (Chemie) δ_{t_C}:
    gemäß (3.69)
    Setze Randwerte:
    für $C_i^{l+\frac{1}{2}}(t_C + \frac{\delta t_C}{2})$
    Berechne Diffusion:
    bestimme $C_i^{l+\frac{1}{2}}(t_C + \delta_{t_C}/2)$ gemäß (3.30):
    Setze Randwerte:
    für $C_i^{l+1}(t_C + \delta t_C)$
    Berechne Reaktion:
    berechne $C_i^{l+1}(t_C + \delta_{t_C})$ mittels Algorithmus 4:
    setze $t_C = t_C + \delta_{t_C}$, $l = l + 1$
  end
  setze $C_i^{k+1} := C_i^{l+1}(t + \delta_{t_S})$
  `/* wenn` $t_C = t + \delta_{t_S}$ `erreicht ist, haben wir` C_i^{k+1} `*/`
  Berechne Schätzgeschwindigkeiten:
  berechne $u^*(u^k, t^k)$ gemäß (3.65)
  Löse Druck-Poisson-Gleichung:
  setze rechte Seite in Abhängigkeit von $u^*(u^k, t^k)$
  berechne iterativ p^{k+1} gemäß (2.22)
  Führe Druck-Korrektur der Geschwindigkeiten durch:
  berechne u^{k+1} aus $u^*(u^k, t^k)$ und p^{k+1} gemäß (2.23)
  Setze $t = t + \delta t_S$, $k = k + 1$
end

4. Konvergenzanalysen und Simulationsergebnisse

In diesem Kapitel prüfen wir unseren Code an einigen Beispielen, etwa aus [SMM13].
Im Allgemeinen ist das System nicht analytisch lösbar. Für bestimmte Parameterwahlen kann jedoch eine Lösung angegeben werden. Dies ermöglicht uns, unsere numerischen Resultate zu verifizieren und Konvergenzanalysen durchzuführen. Auch werden wir unsere Ergebnisse mit numerischen Ergebnissen aus der Literatur vergleichen.
Es war uns in diesem Kapitel wichtig, eine ansteigende Folge der Komplexität in unseren Untersuchungen zu wählen. Auf diese Art ließen sich Fehler in der Implementierung leichter finden und eliminieren. Wir betrachten zunächst zweidimensionale Beispiele, die von einfachsten Fällen bis hin zu komplexeren Literaturbeispielen gehen. Nachdem wir eine gute Verifikation im Zweidimensionalem erreicht haben, wenden wir uns den anspruchsvolleren dreidimensionalen Simulationen zu.

4.1. Vereinfachte Systeme

Das volle Brusselator-System mit den beiden gekoppelten Spezies, der Diffusion und der Nichtlinearität ist bereits von einer nicht zu unterschätzenden Schwierigkeit. Deshalb war unsere erste Aufgabe, ein einfacheres System zu betrachten, in welchem weniger beziehungsweise einfachere Terme vorkommen.

Die Parameter k_i, A und B ermöglichen uns, aus dem Brusselator ein solches System zu erzeugen, indem wir einige der Parameter als Null wählen. Dies haben wir im Folgenden getan. Zuerst betrachten wir ein Beispiel, in dem wir das System auf nur eine Spezies reduziert haben. Anschließend wenden wir uns einem Fall zu, in dem für beide Spezies einfache Gleichungen gelöst werden, jedoch ohne den nichtlinearen Term.
Zusätzlich ist es gelungen, für diese beiden Fälle eine analytische Lösung anzugeben, anhand welcher wir unsere Ergebnisse verfizieren konnten.

Beispiel a - nur eine Spezies

Wir betrachten als erstes folgendes Beispiel:

$k_i =$	0	$i = 1 \dots 3$
$k_4 =$	1	
$B =$	1	
$A =$	1	
$D_{C_1} =$	0	
$D_{C_2} =$	0	

Damit haben wir ein diffusionsfreies System und die Gleichungen lauten ausgeschrieben:

$$\begin{aligned} \frac{dC_1(\vec{x},t)}{dt} &= -C_1(\vec{x},t) \\ \frac{dC_2(\vec{x},t)}{dt} &= 0 \end{aligned} \tag{4.1}$$

Dazu können wir eine analytische Lösung angeben:

$$\begin{aligned} C_1(\vec{x},t) &= c_1(\vec{x},t) \cdot \exp(-t) \\ C_2(\vec{x},t) &\equiv const, \end{aligned} \tag{4.2}$$

wobei $c_1(\vec{x},t) := C_1(\vec{x},0)$ sei.

Wir führen eine Berechnung durch für:

$\Omega =$	$[0,1] \times [0,1]$
$T =$	$[0, 0.01]$
$N =$	40, also 40×40 Gitter in MATLAB und 40×40 mit NaSt
$delt_t$	siehe unten
$N_{max} =$	300 (implizites Verfahren)
$TOL =$	1.0e-9 (implizites Verfahren)

Anhand dieses vereinfachten Systems konnten wir eine erste Überprüfung des Reaktionsteils unseres 2D-MATLAB-Codes durchführen. Wir wählen die Parameter wie oben angeben und können so das System mit unserem Code berechnen.

Die erhaltenen Ergebnisse haben wir mit der analytischen Lösung (4.2) verglichen. Angegeben ist hier jeweils der Fehler:

$$C_{abs} = max_{i,j} \left(C_t^{num}(i,j) - C_t^{ana}(i,j) \right),$$

wobei wir mit $C_t^{num}(i,j)$ unsere numerische Lösung und mit $C_t^{ana}(i,j)$ die analytische Lösung zum Zeitpunkt t an der Stelle x_{ij} meinen.

Die nachfolgenden Tabellen zeigen diesen Fehler für verschiedene Zeitschrittweiten.

für $\mathbf{delt_t = 0.002}$ nach $\mathbf{t = 0.01}$		
Konzentration C_1	explizit MATLAB	implizit MATLAB
	1.0902e-04	1.0902e-04
Konzentration C_2	explizit MATLAB	implizit MATLAB
	0	0

für $\mathbf{delt_t = 1.0e-5}$ nach $\mathbf{t = 0.01}$		
Konzentration C_1	explizit MATLAB	implizit MATLAB
	5.4437e-07	4.4547e-07
Konzentration C_2	explizit MATLAB	implizit MATLAB
	0	0

für $\mathbf{delt_t = 1.0e-7}$ nach $\mathbf{t = 0.01}$		
Konzentration C_1	explizit MATLAB	implizit MATLAB
	5.4437e-09	4.4546e-09
Konzentration C_2	explizit MATLAB	implizit MATLAB
	0	0

Wir können hier sehen, dass wir eine gute Näherung an die analytische Lösung der Gleichung für C_1 erzielt haben. Da C_2 in diesem Beispiel konstant bleibt, ist hier der Fehler immer Null gewesen.
Wie wir sehen, verbessert sich der Fehler hier auch mit einer Verkleinerung der Zeitschrittweite. Allerdings führen wir erst in 4.2 eine eingehendere Konvergenzanalyse durch, welche neben einer Verfeinerung in der Zeit auch eine Verfeinerung im Ort berücksichtigt.

Beispiel b - zwei Spezies ohne Nichtlinearität

Da wir schrittweise die Komplexität unseres Vorgehens erhöhen wollten, nehmen wir nun zu dem vorherigen Beispiel einen modellhaften Term für eine Reaktion in der zweiten Spezies hinzu. Dadurch ergibt sich eine Kopplung von C_2 an C_1. Hier haben wir jedoch noch keine Rückkopplung von C_2 auf C_1, auch der Diffusionsterm und der nichtlineare Term kommen hier noch nicht vor.

Mit Wahl der Parameter als:

$k_1 = k_3 = k_4 = 0$
$k_2 = 1$
$B = 1$
$A = 0$
$D_{C_1} = 0$
$D_{C_2} = 0$
$N = 40$

ergibt sich:

$$\begin{aligned} \frac{dC_1(\vec{x},t)}{dt} &= -C_1(\vec{x},t) \\ \frac{dC_2(\vec{x},t)}{dt} &= C_1(\vec{x},t) \end{aligned} \tag{4.3}$$

Die analytische Lösung für das System lautet:

$$\begin{aligned} C_1(\vec{x},t) &= c_1(\vec{x},t) \cdot \exp(-t) \\ C_2(\vec{x},t) &\equiv -c_1(\vec{x}) \cdot \exp(-t) + c_1(\vec{x}) + c_2(\vec{x}) \end{aligned} \tag{4.4}$$

Dieses System haben wir mit den gleichen Simulationsparametern wie unter a) berechnet. Dabei ergaben sich als Fehler:

für $\mathbf{delt_t = 0.002}$ nach $\mathbf{t = 0.01}$		
Konzentration C_1	explizit MATLAB	implizit MATLAB
	1.0902e-04	8.8977e-05
Konzentration C_2	explizit MATLAB	implizit MATLAB
	8.9213e-05	1.0873e-04

für $\mathbf{delt_t = 1.0e-5}$ nach $\mathbf{t = 0.01}$		
Konzentration C_1	explizit MATLAB	implizit MATLAB
	5.4437e-07	4.4547e-07
Konzentration C_2	explizit MATLAB	implizit MATLAB
	4.4548e-07	5.4437e-07

für $\mathbf{delt_t = 1.0e-7}$ nach $\mathbf{t = 0.01}$		
Konzentration C_1	explizit MATLAB	implizit MATLAB
	5.4437e-09	4.4546e-09
Konzentration C_2	explizit MATLAB	implizit MATLAB
	4.4547e-09	5.4437e-09

Da die Konzentrationsgleichung für C_1 die gleiche war wie in Beispiel a), haben wir für C_1 die gleichen Ergebnisse erhalten wie dort. C_2 ist nun nicht mehr konstant gewesen. Der Fehler in C_2 ist ebenfalls klein. Mit diesen beiden einfachen Beispielen haben wir also eine erste Validierung der expliziten und impliziten Berechnung der Reaktion (ohne Diffusionsanteil) durchführen können.
Somit wollen wir uns jetzt einigen komplexeren Beispielen aus der Literatur zuwenden.

4.2. Konvergenzanalyse des 2D-Brusselators

Wir betrachten zunächst Beispiel 1 aus [SMM13]. Hier sind auch der nichtlineare Term und die Diffusion vorhanden, aber es lässt sich trotzdem eine analytische Lösung angeben, was sonst im Allgemeinen nicht der Fall ist. Mit einem Vergleich von unseren berechneten Ergebnissen mit einer analytisch berechneten Lösung können wir eine gute Verifikation unserer Implementierung erzielen.

Wir wollen weiterhin dabei auch eine ausführliche Konvergenzanalyse durchführen. Dies tun wir für die explizite und die implizite Implementierung mit dem 2D-Prototypen des Codes in MATLAB. Da uns leider kein Beispiel für eine analytische Lösung des Brusselators in 3D bekannt ist, können wir die genaue Konvergenzanalyse nur am 2D-Code durchführen.

Um ein besseres Verständins für das Konvergenzverhalten der implementierten Verfahren zu gewinnen, führen wir verschiedene Betrachtungen durch. Als Erstes wollen wir für beide Verfahren eine getrennte Studie jeweils in Zeit und Ort vornehmen. Das heißt, wir halten die Ortsauflösung fest und verkleinern die Zeitauflösung - beziehungsweise umgekehrt. Die Frage die sich stellt, ist ob wir so eine Konvergenz in Zeit und Ort beobachten können. Weiterhin werden wir Berechnungen durchführen, bei denen gleichzeitig die Auflösungen in Zeit und Ort verkleinert wurden.

Wir betrachten also folgendes Setting:

$\Omega =$ $[0,1] \times [0,1]$
$T =$ $[0,2]$
$k_i =$ $1 \quad i = 1 \dots 4$
$B =$ 1
$A =$ 0
$D_{C_1} =$ 0.25
$D_{C_2} =$ 0.25

Wobei die Anfangsbedingungen und die Dirichlet-Randbedingungen gemäß der analytischen Lösung gesetzt werden, die für diese Konfiguration lautet:

$$\begin{aligned} C_1(x_1, x_2, t) &= exp(-x_1 - x_2 - 0.5t) \\ C_2(x_1, x_2, t) &= exp(x_1 + x_2 + 0.5t) \end{aligned} \tag{4.5}$$

Wir betrachten jeweils den relativen Fehler in der L_2-Norm:

$$C_{rel} = \frac{1}{N} \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} \frac{C_t^{num}(i,j) - C_t^{ana}(i,j)}{C_t^{ana}(i,j)}}$$

für $C = C_1$ oder $C = C_2$ und N die Anzahl der Gitterpunkte.
Mit $C^{num}(i,j)$ meinen wir wieder unsere berechnete numerische Lösung, mit C^{ana} die

analytische.
Sofern nichts anderes angegeben ist, beziehen wir uns in diesem Abschnitt immer auf diese Art der Fehlermessung.

4.2.1. Explizites Verfahren - Zeitanalyse

Wir haben zuerst mit einer hohen Auflösung von 100×100 Gitterpunkten die Konvergenz des expliziten Verfahrens bei kleiner werdenden Zeitschrittweiten betrachtet. Als gröbste Zeitschrittweite haben wir $\delta t = 1.0e-4$ gewählt. Davon ausgehend haben wir für jede Berechnung die Zeitschrittweite halbiert.

Die Zeitschrittweiten für die verschiedenen Rechnungen waren also:

δt
$1.0e-4$
$5.0e-5$
$2.5e-5$
$1.25e-5$
$6.25e-6$
$3.125e-6$
$1.5625e-6$
$7.8125e-7$

Dazu haben wir uns jeweils den Fehler C_{rel} zum Zeitpunkt $t = 1$ angesehen.
Die folgenden beiden Plots zeigen für C_1 und C_2 jeweils diesen Fehler für die mit den verschiedenen Verfeinerungen von δt berechneten Resultate.

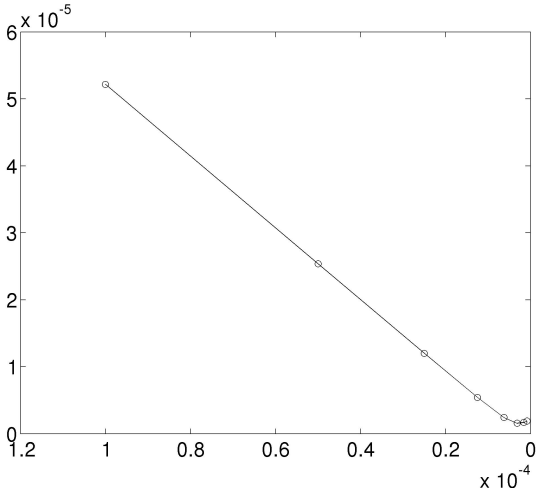


Abbildung 4.1.: Zeitanalyse für C_1, explizites Verfahren, $N = 100$

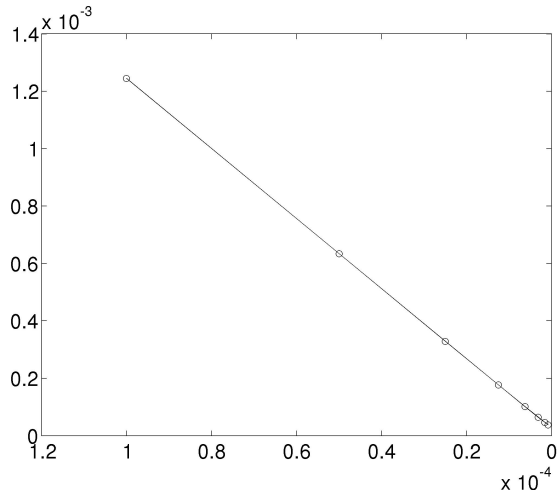


Abbildung 4.2.: Zeitanalyse für C_2, explizites Verfahren, $N = 100$

Zum Bestimmen der Konvergenzrate haben wir eine doppelt logarithmische Version der Plots erstellt. Aufgetragen ist also $log(C_{Rel})$ gegen $log(\delta t)$. Zu diesen Werten haben wir eine Ausgleichsgerade bestimmt, ihre Steigung s gibt uns die Konvergenzrate an. Zur besseren Übersicht haben wir die Gerade nicht eingezeichnet und nur s angegeben.

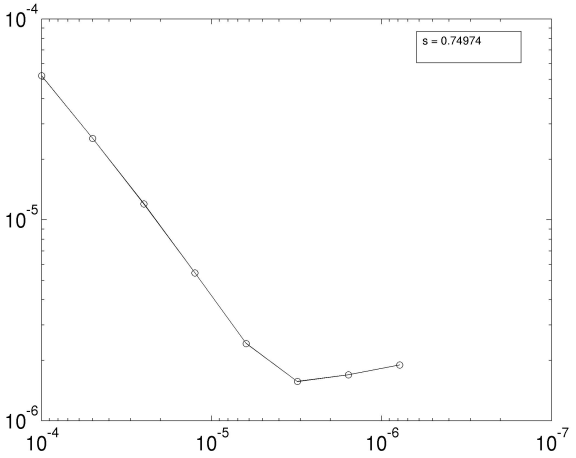


Abbildung 4.3.: Zeitanalyse für C_1, explizites Verfahren, $N = 100$ – doppelt logarithmisch

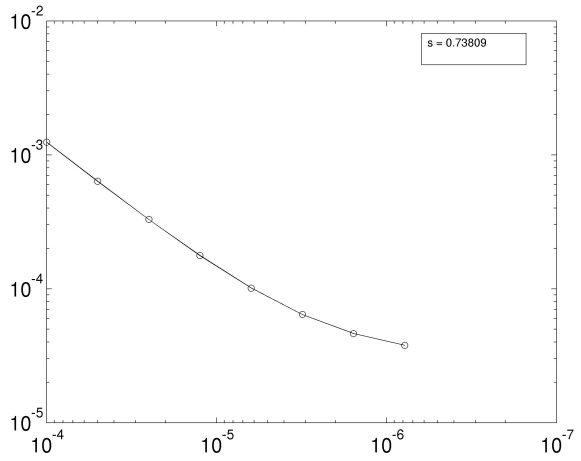


Abbildung 4.4.: Zeitanalyse für C_2, explizites Verfahren, $N = 100$ – doppelt logarithmisch

Anhand dieser Plots kann man erkennen, dass sich, wenn wir die Zeitschrittweite verkleinern und alle anderen Simulationsparameter belassen, der Fehler verkleinert und (in C_1) ab einem gewissen Punkt stagniert.
Die Stagnation, die wir hier in C_1 sehen, erklärt sich dadurch, dass wir bei der Diskretsierung auch einen Ortsfehler machen. Deshalb kann - bei festgehaltener Ortsauflösung - der Fehler nicht unbegrenzt immer weiter abnehmen. Dementsprechend wäre auch für C_2 zu erwarten, dass der Fehler irgendwann stagniert. Dies konnte in dieser Rechnung nicht beobachtet werden, würde jedoch bei noch weiterer Verfeinerung eintreten. Der Plot für C_2 ist deshalb auf den ersten Blick etwas irreführend, diente uns aber in erster Linie auch dazu, die Konvergenzordnung zu bestimmten.
Die logarithmischen Plots lassen uns für C_1 und C_2 eine Rate von circa 0,7 ablesen. Diese Rate erklärt sich dadurch, dass bei der Bestimmung der Ausgleichsgeraden alle Punkte berücksichtigt wurden, auch diejenigen im Bereich in dem der Fehler stagnierte. Wir hätten auch diesen Bereich bei der Bestimmung der Ausgleichsgeraden unberücksichtigt lassen können und hätten dann eine Rate von etwa 1 erhalten.
Die Konvergenzordnung 1 war zu erwarten gewesen, da es sich bei dem expliziten Eulerverfahren um ein Verfahren erster Ordnung handelt.
Es ist noch zu bemerken, dass eigentlich auch der Fehler durch das Splittingverfahren eine Rolle spielt. Vergleiche dazu die Anmerkungen unter 4.2.6.

4.2.2. Explizites Verfahren - Ortanalyse

Desweiteren haben wir eine Konvergenzanalyse im Ort für das explizite Verfahren vorgenommen. Hier haben wir die Anzahl der Gitterpunkte N sukzessive verfeinert. Als Zeitschrittweite haben wir für diese Berechnungen $\delta t = 1.0e - 6$ gewählt. Aufgrund dessen, dass sich die CFL-Bedingungen (siehe Abschnitt 3.7.3) mit zunehmender Verfeinerung

des Gitters verschärfen, war es notwendig eine solch geringe Zeitschrittweite zu wählen. Als Ortsauflösungen wurden gewählt:

N
20
40
80
160
320

Für C_1 und C_2 verhält sich der Fehler folgendermaßen:

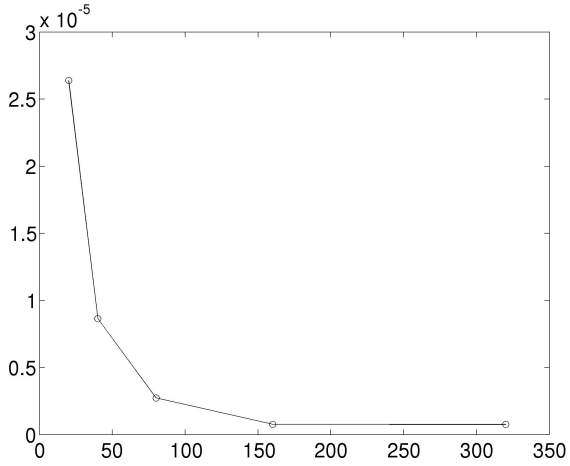


Abbildung 4.5.: Ortanalyse für C_1, explizites Verfahren, $\delta_t = 1.0 \times 10^{-6}$

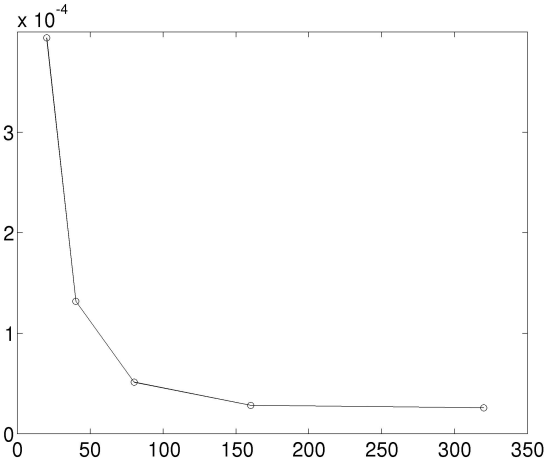


Abbildung 4.6.: Ortanalyse für C_2, explizites Verfahren, $\delta_t = 1.0 \times 10^{-6}$

Wir sind hier wieder vorgegangen wie oben und haben eine doppelt logarithmische Variante der Plots erstellt:

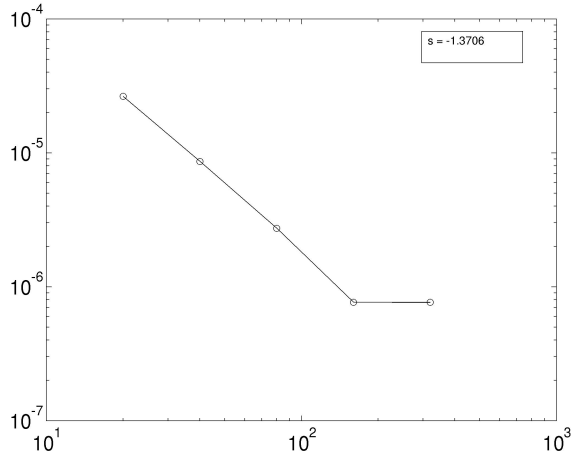


Abbildung 4.7.: Ortanalyse für C_1, explizites Verfahren, $\delta_t = 1.0 \times 10^{-6}$ – doppelt logarithmisch

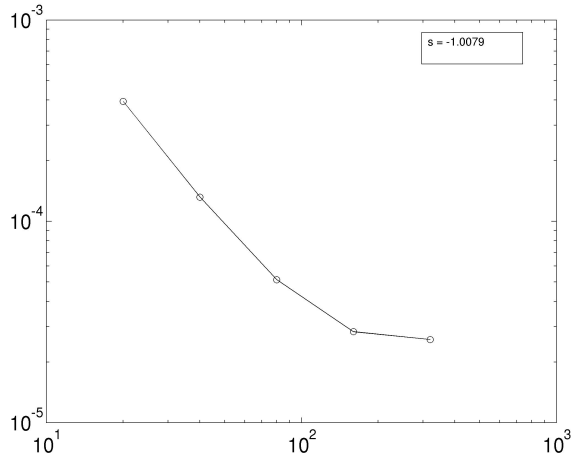


Abbildung 4.8.: Ortanalyse für C_2, explizites Verfahren, $\delta_t = 1.0 \times 10^{-6}$ – doppelt logarithmisch

Wir sehen in den Plots ähnlich wie bei der vorherigen Zeitanalyse wieder, dass der Fehler mit der Verfeinerung abnimmt, jedoch später abflacht. Dies ist zu erwarten und geschieht aus analogen Gründen wie oben beschrieben. Eine weitere Verfeinerung des Ortes kann hier keine Verkleinerung des Fehlers mehr erzielen, da man durch den Zeitfehler beschränkt ist.

Die Raten sind hier also bei -1.3 und -1. Das negative Vorzeichen liegt dabei nur an der umgekehrt angeordneten Darsetlltung der Werte.[1] Die Konvergenzraten sind also circa bei 1. Die Abweichung erklärt sich wie schon im Fall der Zeitanalyse dadurch, dass das Verfahren ab einer gewissen Verfeinerung stagniert.
Im Bereich der Konvergenz haben wir also auch hier wieder Ordnung 1 erreicht, wie beim expliziten Eulerverfahren zu erwarten.

4.2.3. Implizites Verfahren - Zeitanalyse

Wir führen jetzt für das implizte Verfahren die gleichen Betrachtungen durch wie soeben für das explizite.
Zunächst halten wir die Ortsauflösung bei $N = 100$ fest und verfeinern in der Zeit. Bei dem impliziten Verfahren konnten wir hier größere Zeitschrittweiten wählen als zuvor. Verwendet wurden:

δt
0.01
0.005
0.0025
0.00125
$6.25e-4$
$3.125e-4$
$1.5625e-4$
$7.8125e-5$

Damit haben wir zum Zeitpunkt $t = 1$ folgende Resultate für den relativen Fehler C_{rel} in der L_2-Norm erhalten:

[1] Wir haben der Übersicht halber alle Plots so angeordnet, dass auf der x-Achse (die Orts- bzw. Zeitfeinheit angibt) die Werte nach rechts hin verfeinert werden. Somit sind in den Plots der Zeitanalysen die Werte auf der x-Achse etwa zwischen $1.0e-4$ und $7.8125e-7$ nach rechts kleiner werdend. In den Plots der Zeitanalysen werden die Werte zwischen 20 und 350 de facto größer, was jedoch einer Verfeinerung entspricht, da eine größere Anzahl der Gitterpunkte einer feineren Maschenweite entspricht. Entsprechendes gilt auch für die logarithmischen Plots, und für alle nachfolgenden.

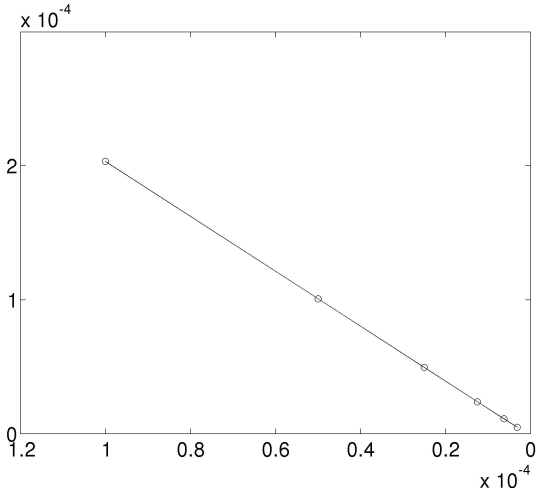


Abbildung 4.9.: Zeitanalyse für C_1, implizites Verfahren, $N = 100$

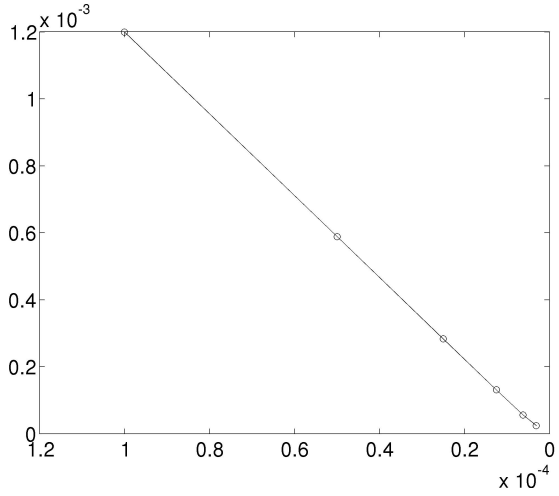


Abbildung 4.10.: Zeitanalyse für C_2, implizites Verfahren, $N = 100$

Wir sehen hier wieder, dass der Fehler immer weiter abzunehmen scheint. Allerdings müssen wir berücksichtigen - wie oben schon bemerkt - dass dieses in den Plots zu sehende Verhalten sich nicht immer weiter so fortsetzen würde. Tatsächlich ist zu erwarten, dass der Fehler auch hier mit zunehmender Verfeinerung der Zeitschrittweite irgendwann abflacht. Der Grund ist wieder, dass bei einer fest bleibenden Ortsauflösung irgendwann der Ortsfehler erricht sein würde. Dies könnte man für eine andere Wahl der Zeitschrittweiten oder der festen Ortsauflösung auch beobachten. Die Plots sind deshalb zwar durchaus richtig, aber etwas irreführend.

Als Konvergenzraten erhalten wir in den logarithmischen Plots:

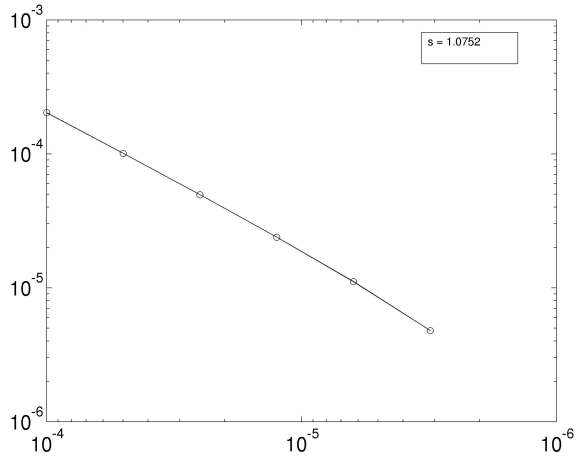


Abbildung 4.11.: Zeitanalyse für C_1, implizites Verfahren, $N = 100$ – doppelt logarithmisch

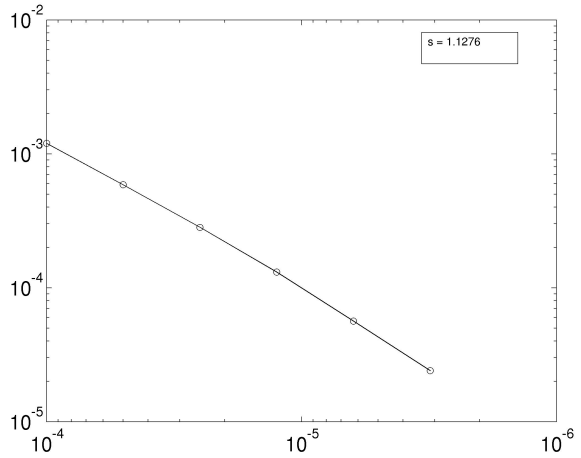


Abbildung 4.12.: Zeitanalyse für C_2, implizites Verfahren, $N = 100$ – doppelt logarithmisch

Wir haben also erwartungsgemäß eine Konvergenz der Ordnung 1.

4.2.4. Implizites Verfahren - Ortanalyse

Auch hier führen wir wieder eine Konvergenzanalyse im Ort durch.
Als Ortsauflösungen wurden gewählt:

N
20
40
80
160
320
640
1280

Betrachtet wurde hier der Fehler nach einem Zeitschritt, also zum Zeitpunkt $t = 1.0 \times 10^{-6}$.

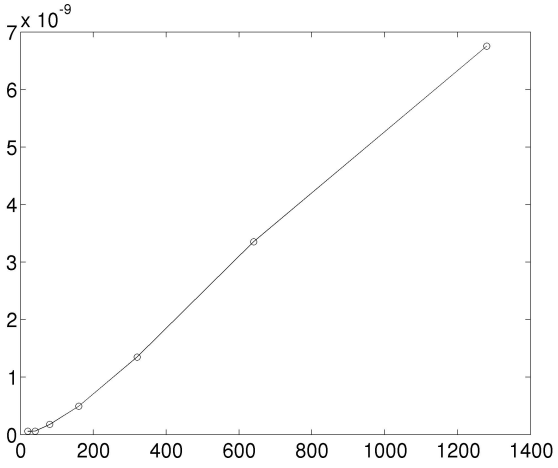


Abbildung 4.13.: Ortanalyse für C_1, implizites Verfahren, $\delta_t = 1.0 \times 10^{-6}$ – nach einem Zeitschritt

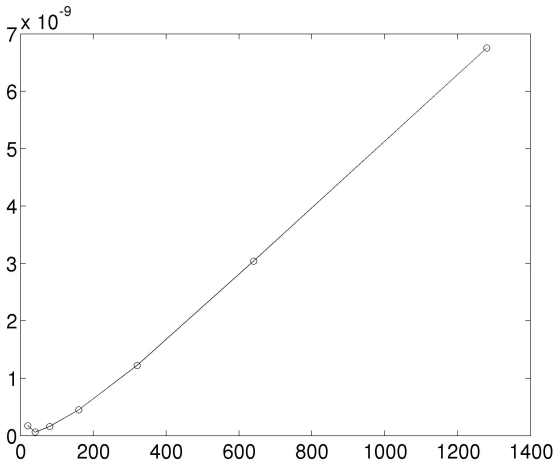


Abbildung 4.14.: Ortanalyse für C_2, implizites Verfahren, $\delta_t = 1.0 \times 10^{-6}$ – nach einem Zeitschritt

Hier lässt sich für das implizite Verfahren im Ort keine Verkleinerung des Fehlers feststellen. Der Grund dafür ist, dass man eigentlich gleichzeitig die Orts- und die Zeitauflösung verkleinern sollte. Siehe dazu auch die Anmerkungen unter 4.2.6.
Erst wenn beide Größen gleichzeitig verkleinert werden, müssen wir mit Konvergenz rechnen. Diesem Fall wenden wir uns als Nächstes zu.

4.2.5. Gleichzeitige Verfeinerung - explizites Verfahren

Nachdem wir in den vorangehenden Abschnitten Zeit und Ort getrennt verfeinert haben, soll nun beides gleichzeitig verkleinert werden. Dann ist zu erwarten, dass die Verfahren konvergieren.
Wir haben zunächst für das explizite Verfahren folgende Verfeinerungslevel gewählt:

N	δt	Level
32	4^{-5}	5
64	4^{-6}	6
128	4^{-7}	7
256	4^{-8}	8

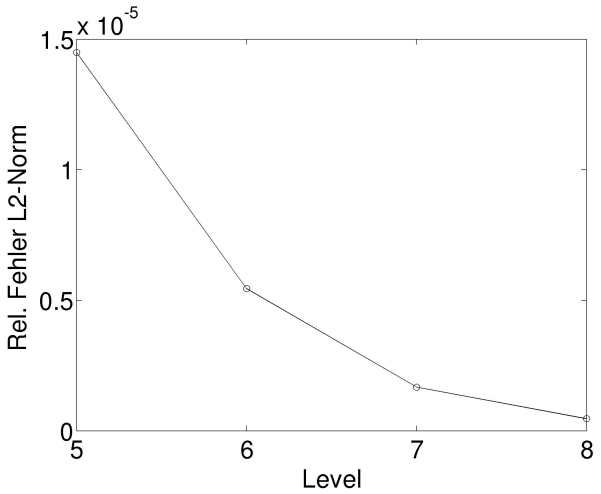


Abbildung 4.15.: C_1, explizites Verfahren, nach $t = 0.00097656$

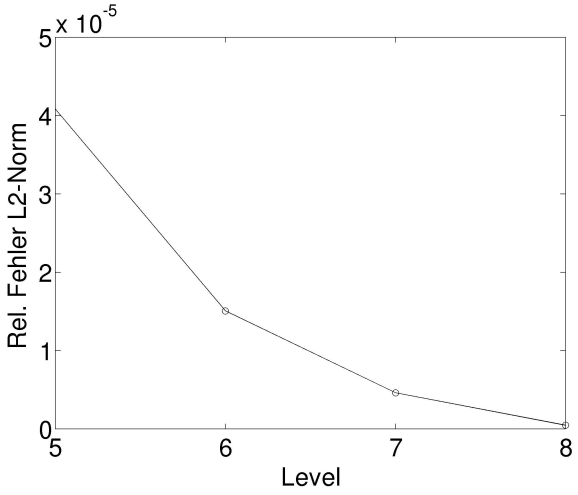


Abbildung 4.16.: C_2, explizites Verfahren, nach $t = 0.00097656$

Als Konvergenzraten bekommen wir:

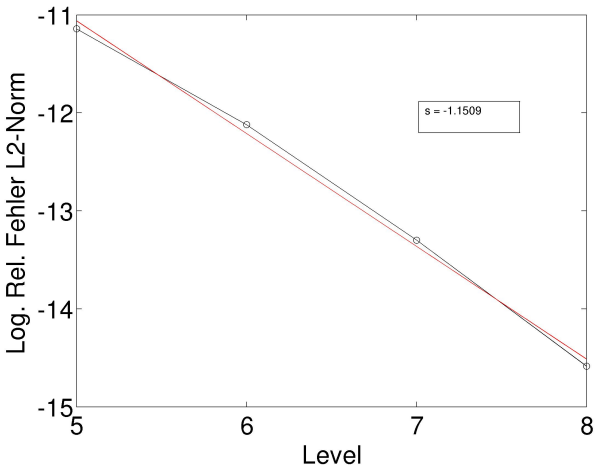


Abbildung 4.17.: C_1, explizites Verfahren, nach $t = 0.00097656$ - logarithmisch

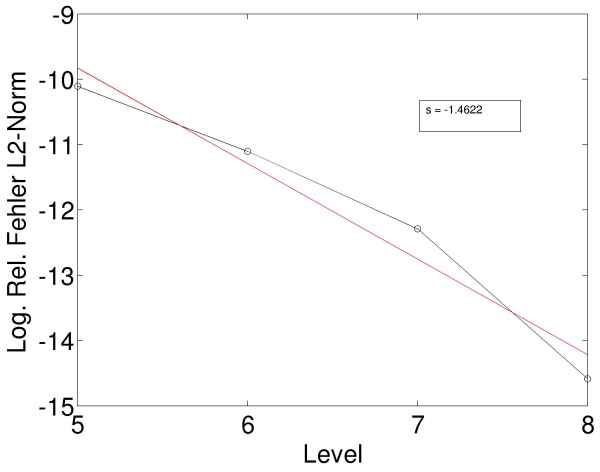


Abbildung 4.18.: C_2, explizites Verfahren, nach $t = 0.00097656$ - logarithmisch

Wir sehen hier also wiederum, dass das explizite Verfahren die Konvergenzordnung 1 hat.

4.2.6. Gleichzeitige Verfeinerung - implizites Verfahren

Das Gleiche führen wir nun für das implizite Verfahren durch.
Wir haben wieder verwendet:

N	δt	Level
32	4^{-5}	5
64	4^{-6}	6
128	4^{-7}	7
256	4^{-8}	8

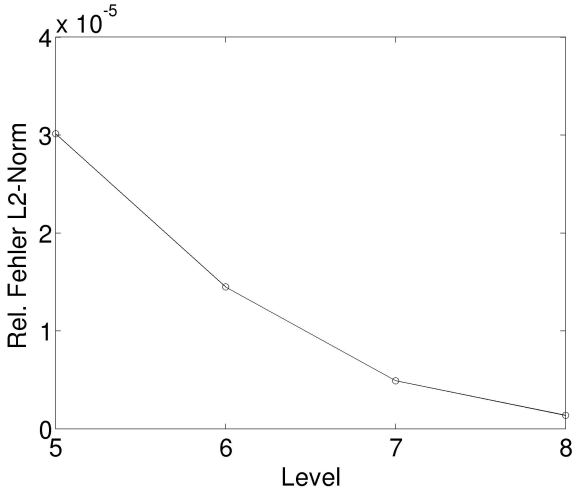


Abbildung 4.19.: C_1, implizites Verfahren, nach $t = 0.00097656$

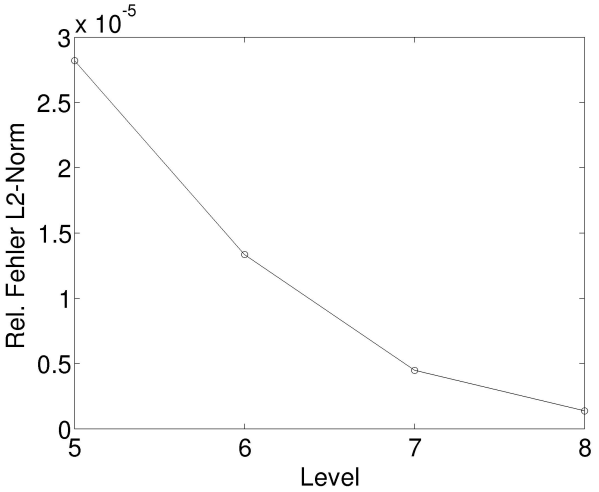


Abbildung 4.20.: C_2, implizites Verfahren, nach $t = 0.00097656$

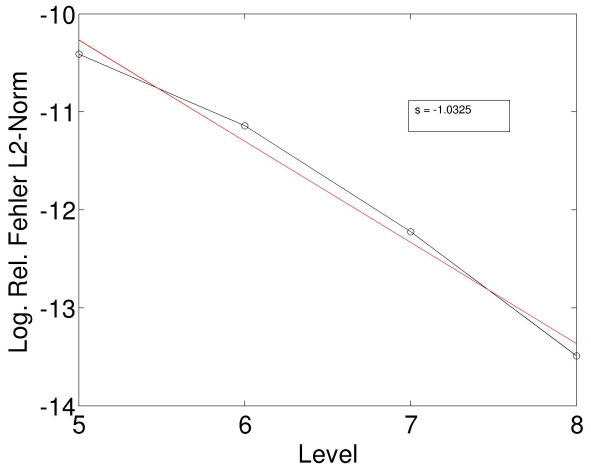


Abbildung 4.21.: C_1, implizites Verfahren, nach $t = 0.00097656$ - logarithmisch

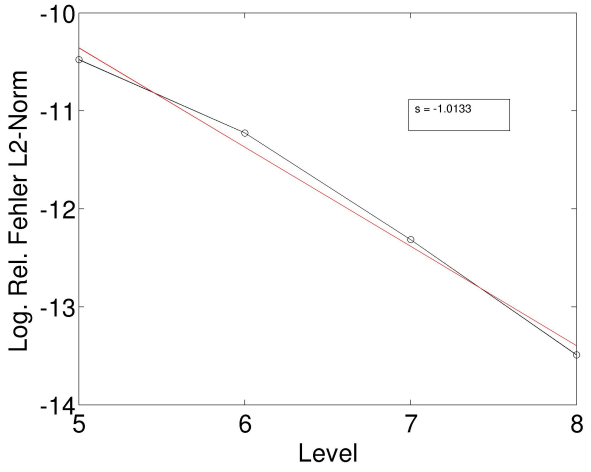


Abbildung 4.22.: C_2, implizites Verfahren, nach $t = 0.00097656$ - logarithmisch

Wie man an den Plots sehen kann, bekommen wir im Fall der gleichzeitigen Verfeinerung von Ort und Zeit für beide Verfahren Konvergenz, wie zu erwarten war. Der Grund dafür ist, dass man eigentlich gleichzeitig die Orts- und die Zeitauflösung verkleinern sollte. Bei unserem Verfahren ergibt sich der Fehler zwischen der analytischen und der numerischen Lösung nicht nur aus der Diskretisierung, sondern auch aus dem Splittingverfahren. Wir haben also neben dem Fehler in Zeit und Ort auch den Splittingfehler.
Wenn man also nur Zeit oder Ort verkleinert, ist nicht unbedingt zu erwarten, dass dies zu einer Verbesserung des Fehlers führt. Dies haben wir auch im Fall der Ortsanalyse des impliziten Verfahrens bestätigt gesehen, in welchem das Verfahren im Ort nicht konvergiert.
Beim expliziten Verfahren haben wir in beiden Fällen der separaten Verfeinerung von Zeit und Ort Konvergenz gesehen. Es ist davon auszugehen, dass dies daran liegt, dass sich

beim expliziten Verfahren der Splittingfehler aufhebt.
Es wäre möglich, anhand des Abflachens der Konvergenz, nun zu bestimmen wie groß jeweils der Fehler in Zeit und Ort ist. Man könnte dann über eine Kosten-Nutzen-Analyse eine Optimierung der Wahl der Zeitschrittweiten und Gitterweiten gegeneinander durchführen. Jedoch ist zu beachten, dass wir bei unserem Problem durch die in 2.4.2 und 3.7.3 beschriebenen CFL-Bedingungen ohnehin in der Wahl der Zeitschrittweite eingeschränkt sind, so dass der Gewinn dadurch nicht groß wäre. Deshalb haben wir in unseren Berechnungen Zeitschrittweite und Gitterweite nur gemäß den CFL-Bedingungen gewählt.

4.3. Vergleich mit weiteren 2D-Literaturbeispielen

Beispiel 2

In diesem und den beiden folgenden Abschnitten wollen wir noch die Übereinstimmung unserer Ergebnisse mit den weiteren Simulationsergebnissen aus [SMM13] zeigen, um die Programm zu überprüfen.

In diesem Beispiel wurden folgende Parameter verwendet:

$\Omega =$ $[0,1] \times [0,1]$
$T =$ $[0,2]$
$k_i =$ $1 \quad i = 1 \dots 4$
$B =$ 2
$A =$ 1
$D_{C_1} =$ 0.25
$D_{C_2} =$ 0.25

mit den Anfangsbedingungen:

$$C_1(\vec{x}, 0) = 0$$
$$C_2(\vec{x}, 0) = 0$$

Die Randwerte wurden gemäß gemäß $(C_1, C_2) = (A, \frac{B}{A})$ als Dirichtlet-Randwerte gesetzt.

MATLAB - explizit

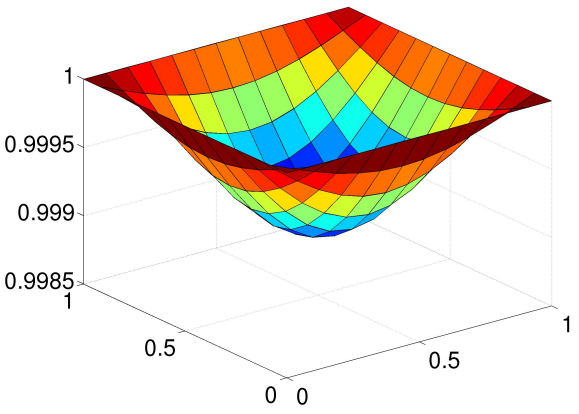


Abbildung 4.23.: C_1 nach $T = 2.0$

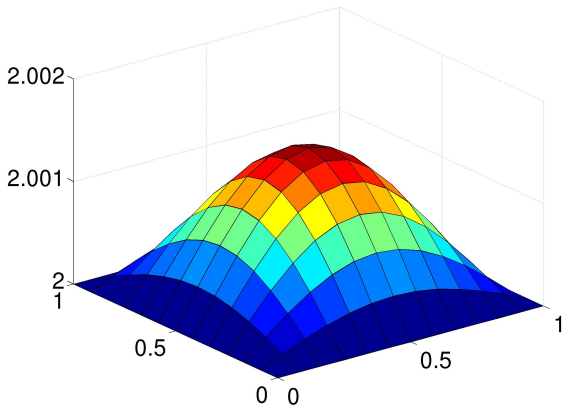


Abbildung 4.24.: C_2 nach $T = 2.0$

MATLAB - implizit

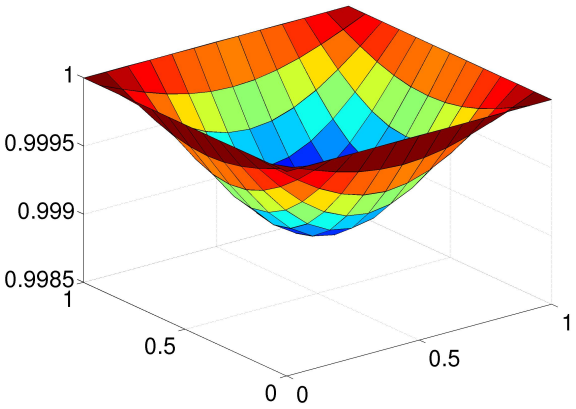


Abbildung 4.25.: C_1 nach $T = 2.0$

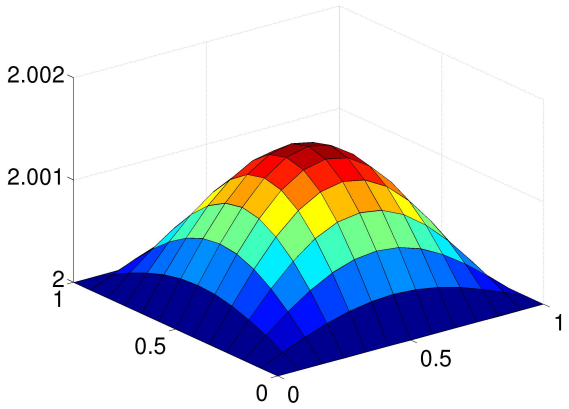


Abbildung 4.26.: C_2 nach $T = 2.0$

Diese Ergebnisse zeigen für das explizite und für das implizite Verfahren eine gute Übereinstimmung mit den Resultaten von [SMM13].

Beispiel 3

Ebenfalls haben wir Beispiel 3 aus [SMM13] betrachtet:

$\Omega =$ $[0,1] \times [0,1]$
$T =$ $[0,5]$
$k_i =$ $1 \quad i = 1 \ldots 4$
$B =$ 1
$A =$ 2
$D_{C_1} =$ 0.002
$D_{C_2} =$ 0.002

mit den Anfangsbedingungen:

$$\begin{aligned} C_1(x_1, x_2, 0) &= 2 + 0.25x_2 \\ C_2(x_1, x_2, 0) &= 1 + 0.8x_1 \end{aligned} \tag{4.6}$$

und Neumann-Randbedingungen:

$$\begin{aligned} &\left.\frac{C_1(x_1,x_2,t)}{\partial x_1}\right|_{x_1=0} = \left.\frac{C_1(x_1,x_2,t)}{\partial x_1}\right|_{x_1=1} = \left.\frac{C_1(x_1,x_2,t)}{\partial x_2}\right|_{x_2=0} = \left.\frac{C_1(x_1,x_2,t)}{\partial x_2}\right|_{x_2=1} = 0 \\ &\left.\frac{C_2(x_1,x_2,t)}{\partial x_1}\right|_{x_1=0} = \left.\frac{C_2(x_1,x_2,t)}{\partial x_1}\right|_{x_1=1} = \left.\frac{C_2(x_1,x_2,t)}{\partial x_2}\right|_{x_2=0} = \left.\frac{C_2(x_1,x_2,t)}{\partial x_2}\right|_{x_2=1} = 0 \end{aligned} \tag{4.7}$$

MATLAB - explizit

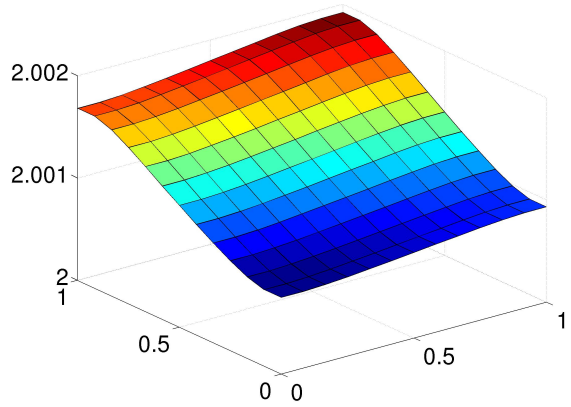


Abbildung 4.27.: C_1 nach $T = 2.0$

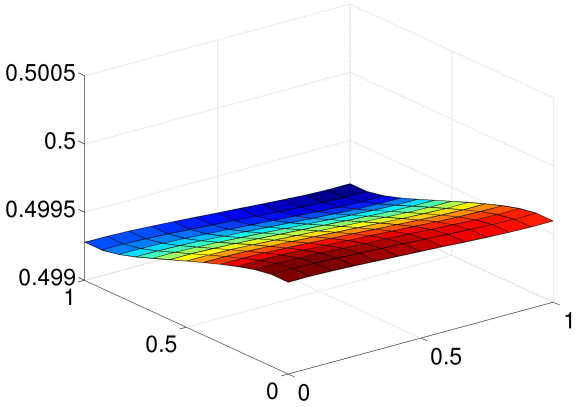


Abbildung 4.28.: C_2 nach $T = 2.0$

MATLAB - implizit

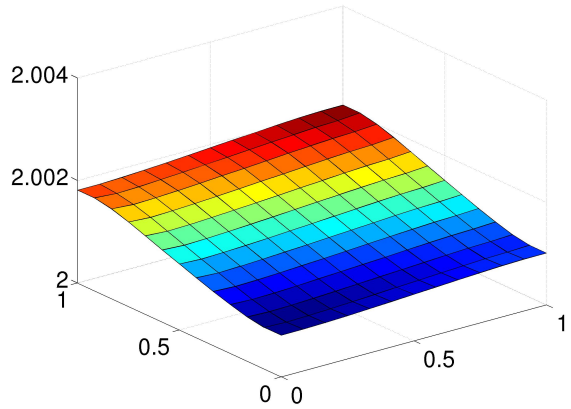


Abbildung 4.29.: C_1 nach $T = 2.0$

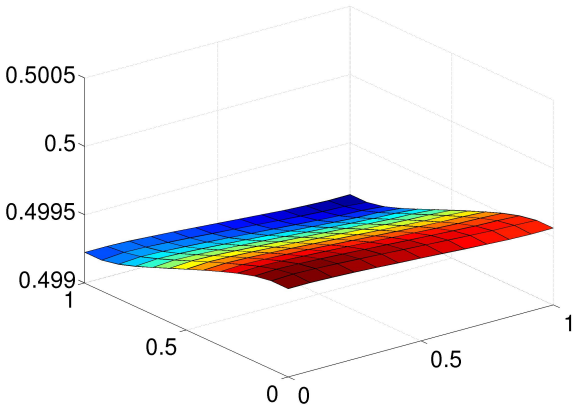


Abbildung 4.30.: C_2 nach $T = 2.0$

Wir haben auch in diesem Fall die Ergebnisse aus der Literatur reproduzieren können. Als Nächstes wenden wir uns einer leichten Modifikation dieses Beispiels zu.

Beispiel 4

Für diese Berechnungen wurden die gleichen Parameter verwendet wie unter Beispiel 3, jedoch mit

$T = [0, 40]$
$B = 3.4$
$A = 1$

MATLAB - explizit

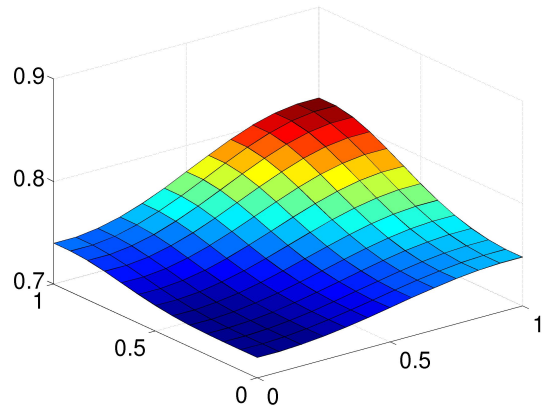


Abbildung 4.31.: C_1 nach $T = 2.0$

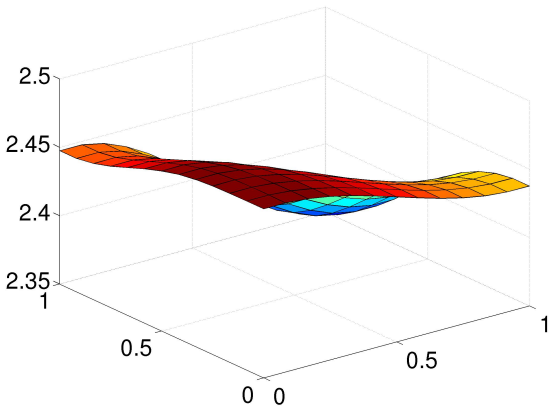


Abbildung 4.32.: C_2 nach $T = 2.0$

MATLAB - implizit

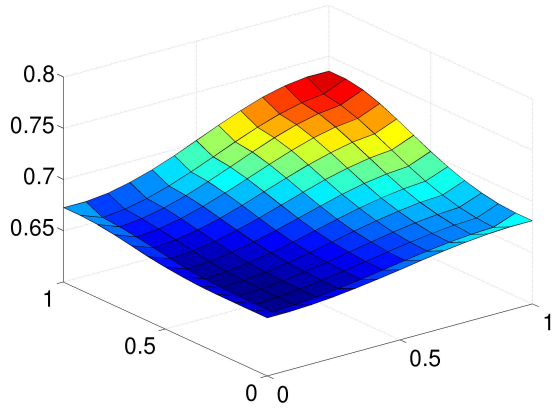


Abbildung 4.33.: C_1 nach $T = 2.0$

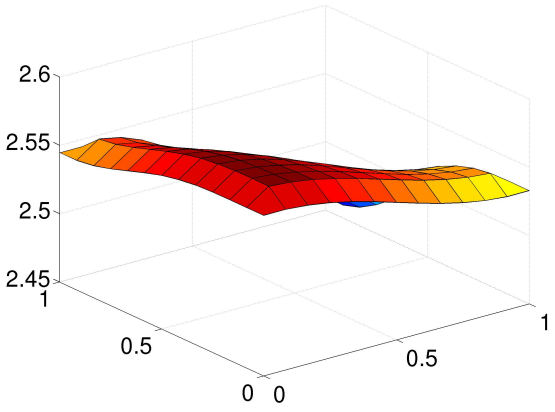


Abbildung 4.34.: C_2 nach $T = 2.0$

Auch hier sind die Ergebnisse übereinstimmend mit der Literatur.
Dieses Beispiel haben wir verwendet um mit der an NaSt3DGPF gekoppelten 3D-Variante Simulationen zu erstellen. Darauf wird nun als Nächstes eingegangen.

4.4. 3D - Simulationsergebnisse

Im folgenden zeigen wir einige der Ergebnisse, die wir mit dem an den Strömungscode gekoppelten Brusselator berechnet haben.
Die Visualisierung der Konzentrations- und Strömungsfelder wurde mit Paraview durchgeführt.

4.4.1. 3D: Oszillierende Reaktion

Wir beginnen zunächst mit einer Simulation, die nur eine dreidimensionale Reaktion zeigt. Wir wollen vorerst zeigen, dass wir mit unserem dreidimensionalen Modell tatsächlich eine realistische Reaktion erzeugen können, die mit Literaturangaben über die BZ-Reaktion konform geht.
Eine Strömung findet deshalb in dieser Simulation noch nicht statt, alle Geschwindigkeiten sind auf 0 gesetzt.

Wir haben das Beispiel 4 aus [SMM13] auf ein dreidimensionales Gebiet erweitert:

Nun auf $\Omega = [0,1] \times [0,1] \times [0,1]$ wurden also wieder die Parameter verwendet:
$T = \quad [0, 40]$
$k_i = \quad 1 \qquad i = 1 \ldots 4$
$B = \quad 3.4$
$A = \quad 1$
$D_{C_1} = \quad 0.002$
$D_{C_2} = \quad 0.002$
Als Anfangskonzentration haben wir in 3D verwendet:

$$\begin{aligned} C_1(x_1, x_2, x_3, 0) &= 2 + 0.25x_2 \\ C_2(x_1, x_2, x_3, 0) &= 1 + 0.8x_1 \end{aligned} \tag{4.8}$$

Die Randwerte wurden an jedem Rand als Neumann-0-Randbedingungen gesetzt.

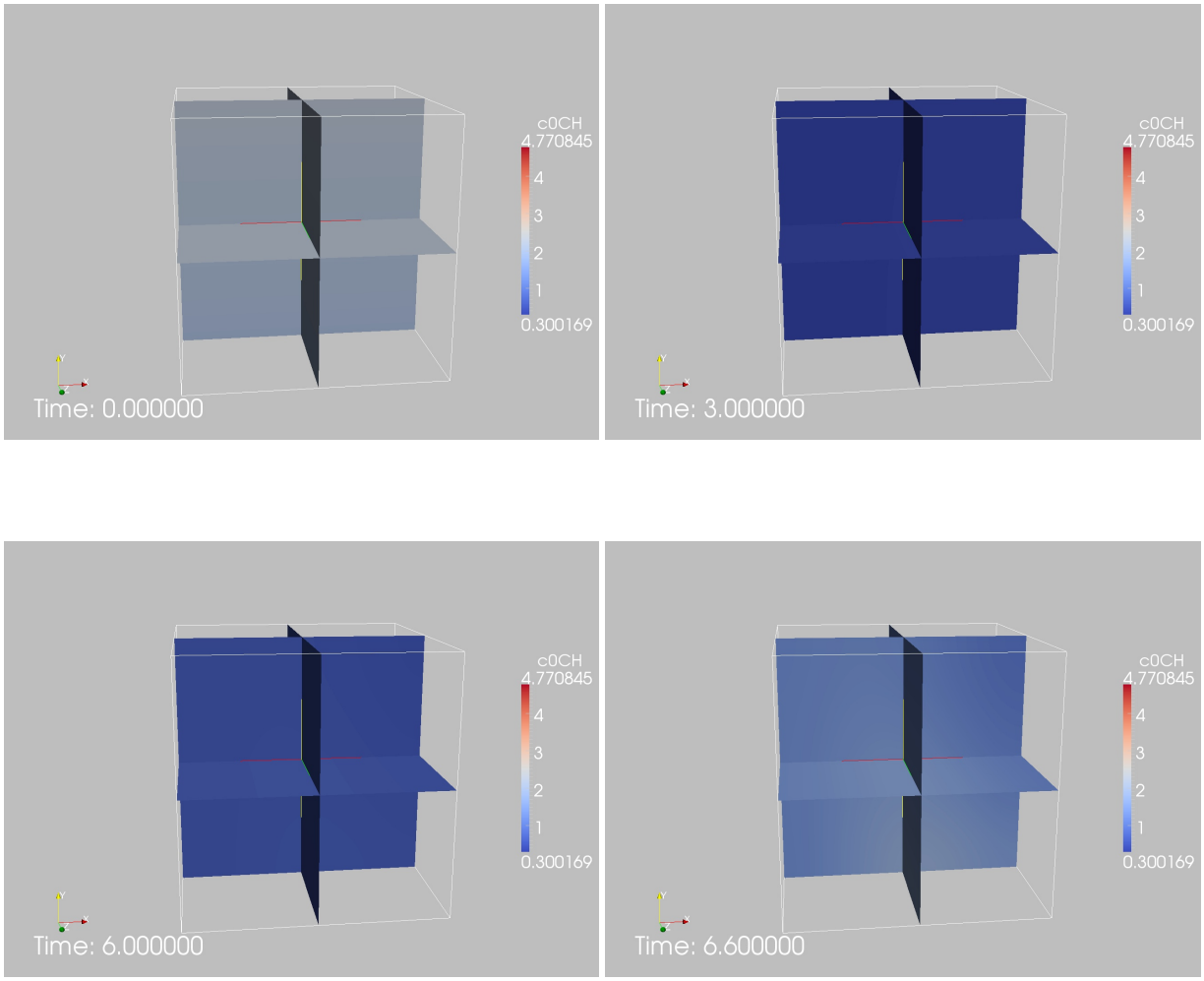
Die Parameter für den Löser sind in Tabelle 4.1 dargestellt.

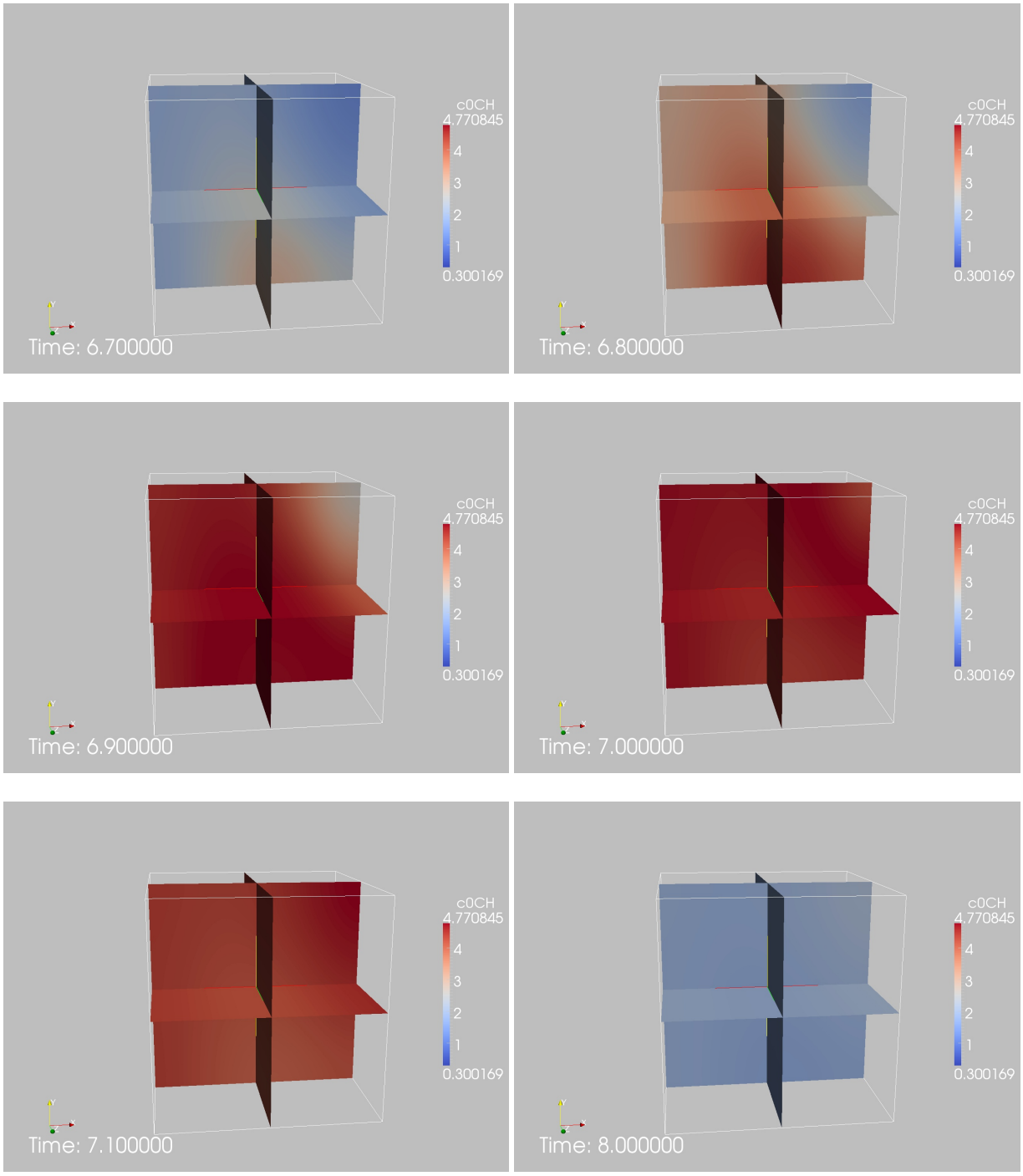
Simulation

Die folgenden Abbildungen zeigen die Veränderung der Konzentration C_1.
Dargestellt sind x-, y- und z-Schnitte (*slices*) durch das 3D-Gebiet, welches mit einem $40 \times 40 \times 40$ Gitter aufgelöst wurde:

	Auflösung	$40 \times 40 \times 40$
δt_S	Zeitschrittweite Strömung	0.002
ϵ	Genauigkeit des Poissonlösers	1.0e-10
$itermax$	Druckiterationen	300
T_{end}	Simulationsende (sec)	40
$\vec{G}$	Volumenkräfte	$\vec{0}$
	Diskretisierung der konvektiven Terme	VONOS
	Drucklöser	PCG
	Zeitdiskretisierung (Strömung)	Euler 1.Ordnung
	Reaktionslöser	implizit
	TOL	1.0e-9
	Nmax	300
δt_C	Zeitschrittweite Chemie-Berechnung	0.002

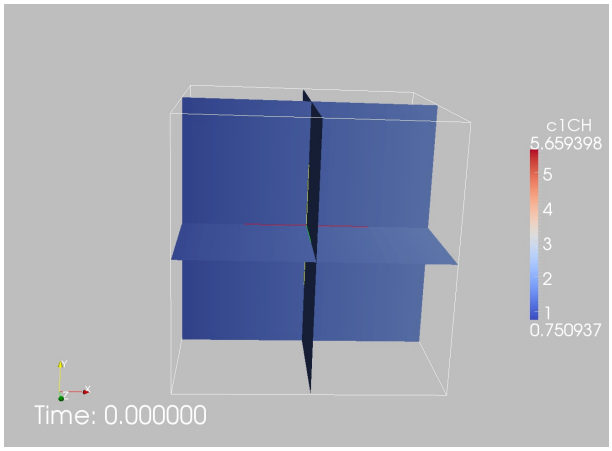
Tabelle 4.1.: Parameter zur Berechnung der 3D-Reaktion

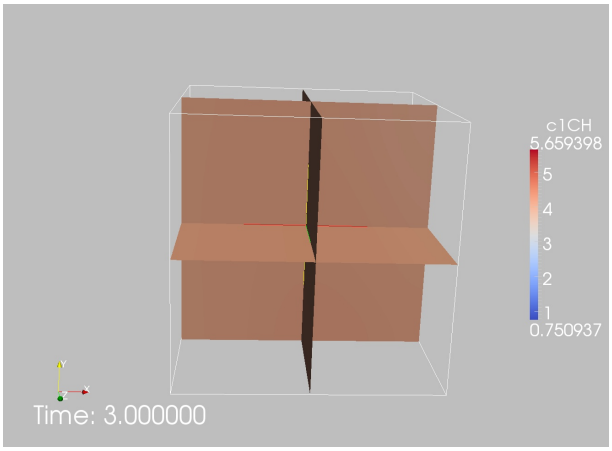


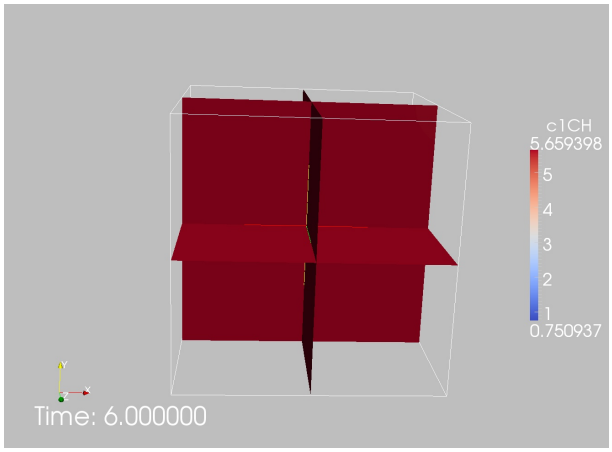


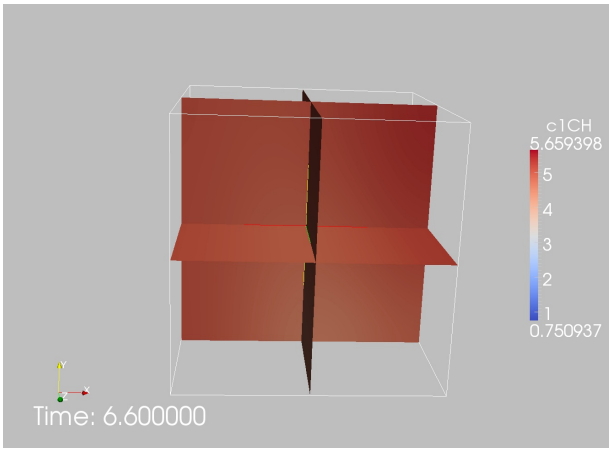
Wir sehen hier eine deutliche Oszilliation in der Konzentration C_1. Etwa zwischen $T = 6$ und $T = 7$ ändert sich die Konzentration komplett.

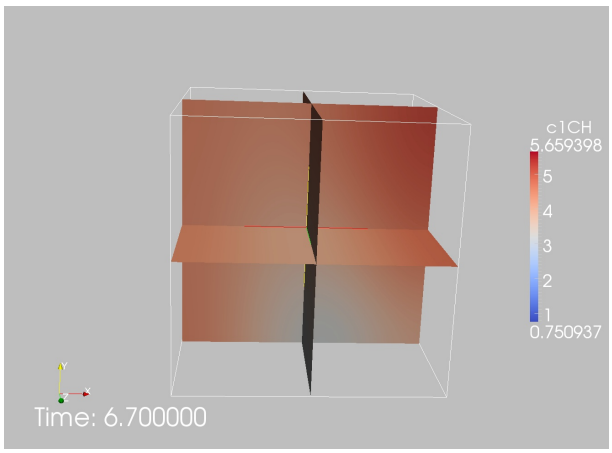
Das Feld C_2 verhält sich entsprechend. Während C_1 abnimmt, nimmt C_2 zu - beziehungsweise umgekehrt. Das Maximum/Minimum wird etwa an den gleichen Zeitpunkten erreicht:

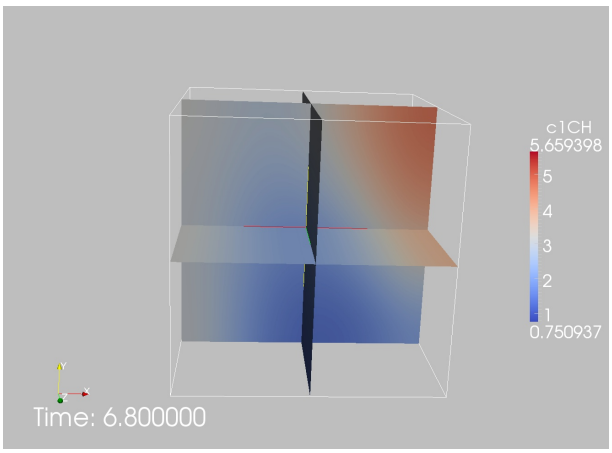

c1CH
5.659398
5
4
3
2
1
0.750937
Time: 0.000000

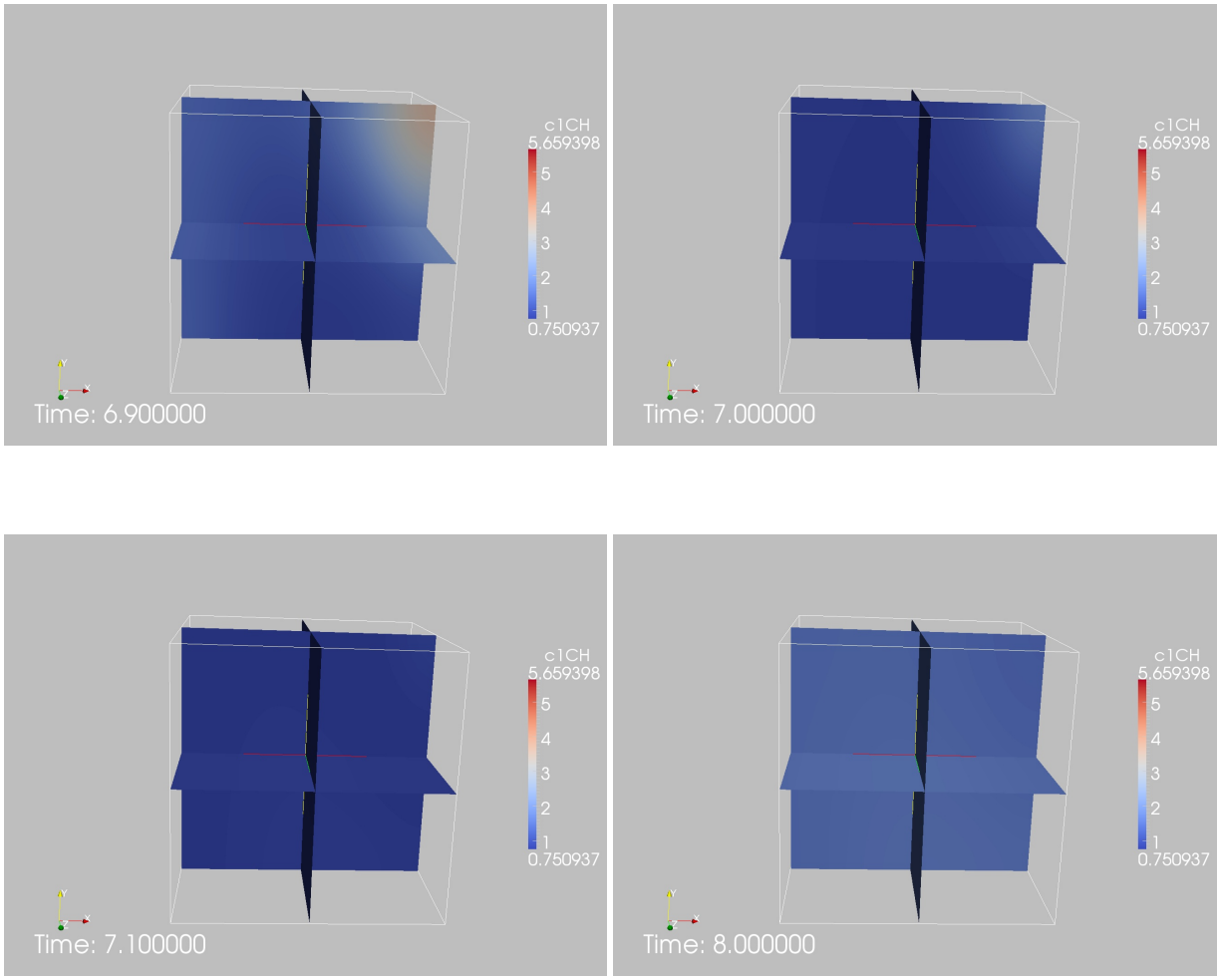

c1CH
5.659398
5
4
3
2
1
0.750937
Time: 3.000000


c1CH
5.659398
5
4
3
2
1
0.750937
Time: 6.000000

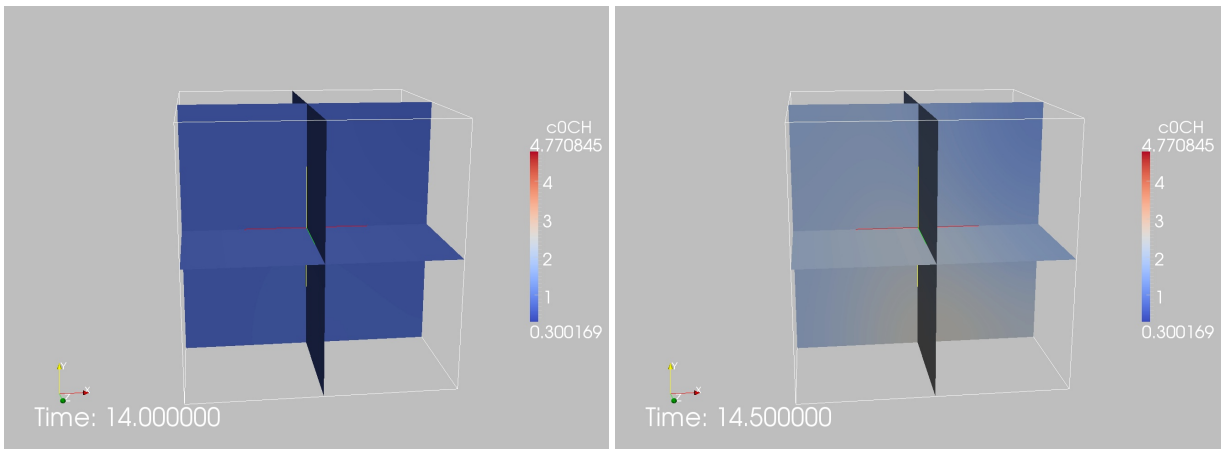

c1CH
5.659398
5
4
3
2
1
0.750937
Time: 6.600000

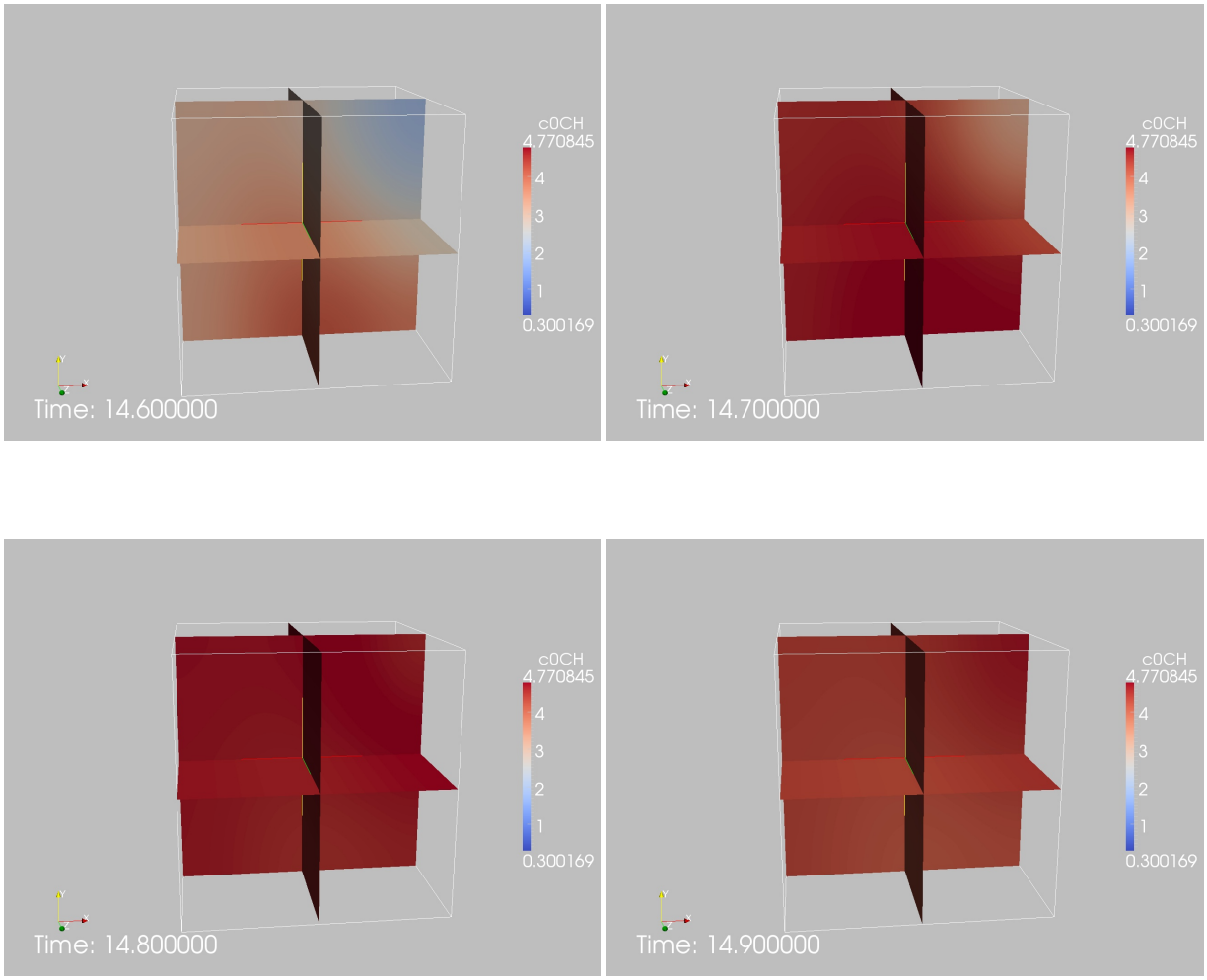

c1CH
5.659398
5
4
3
2
1
0.750937
Time: 6.700000


c1CH
5.659398
5
4
3
2
1
0.750937
Time: 6.800000

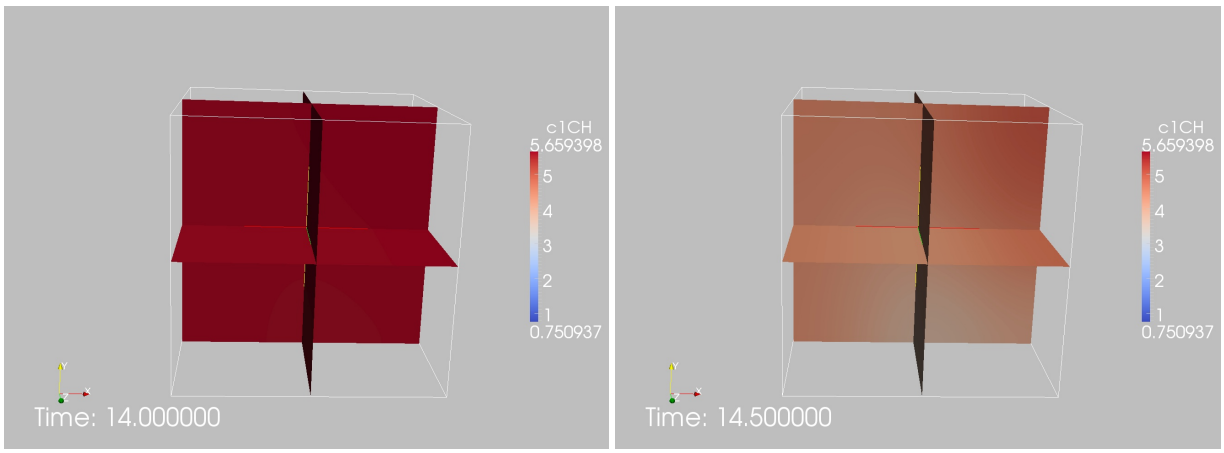


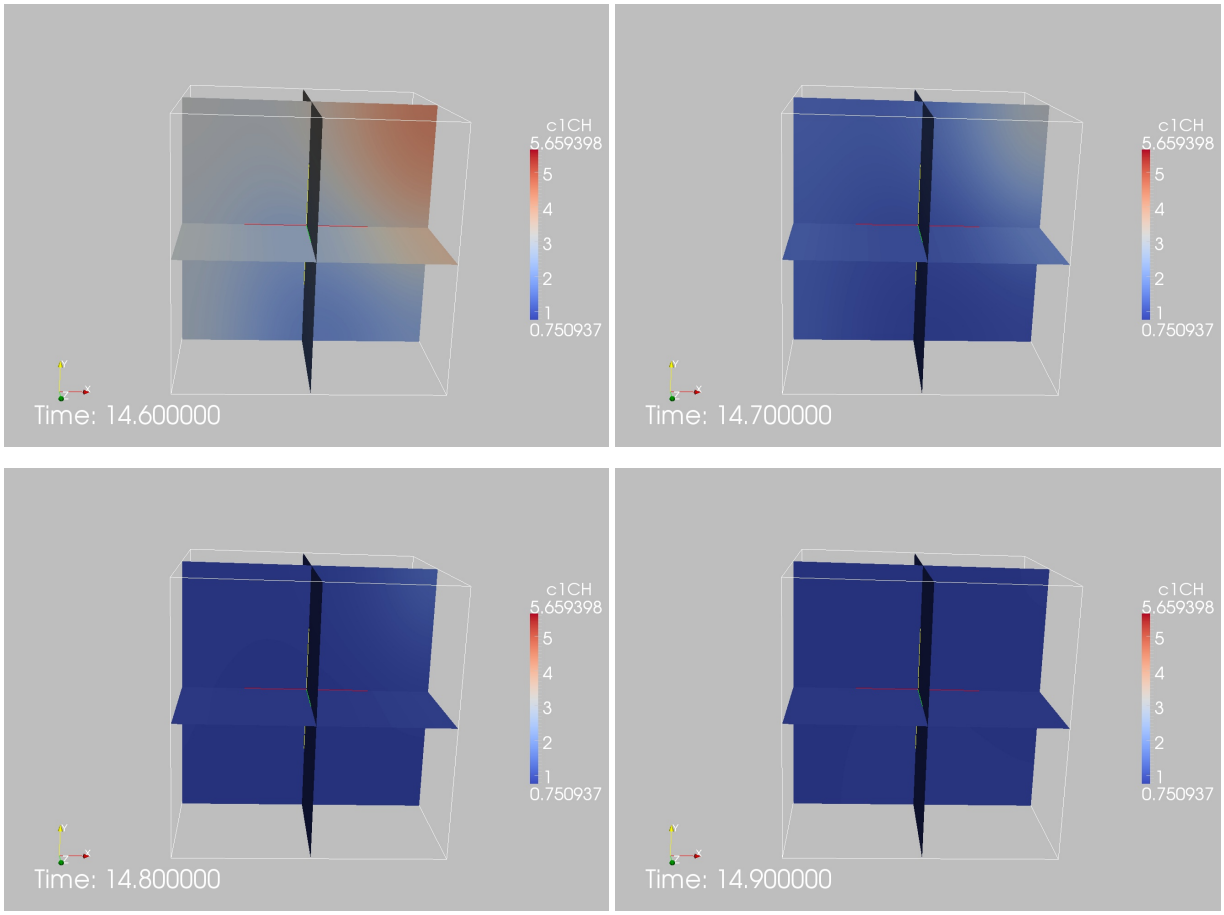
Dieses Verhalten von C_1 und C_2 hat sich in den Simulationen periodisch gezeigt. Hier sehen wir eine weitere Oszillation zwischen $T = 14$ und $T = 15$:





Ebenso konnten wir bei C_2 zur gleichen Zeit wieder eine Oszillation beobachten:





Die Rechnung wurde bis $T = 40$ durchgeführt und es konnten zu den etwa gleichen Zeitabständen, jeweils alle 6-7 Sekunden, weitere Oszillationszyklen in C_1 und C_2 beobachtet werden.

Dieses auf 3D erweiterte Setting zeigt also ein ähnliches Verhalten wie das 2D-Beispiel aus der Literatur:
Der Vergleich mit Figure 10 aus [SMM13] zeigt, dass sich die Oszillationen etwa an den gleichen Zeitpunkten befinden. Auch bewegen sich die Werte von C_1 und C_2 im gleichen Wertebereich.

4.4.2. 3D: Oszillierende Reaktion mit aktiver Kopplung an das Strömungsfeld

Nachdem wir in 4.4.1 erfolgreich eine dreidimensionale Brusselator-Reaktion simulieren konnten, bestand das nächste Ziel darin, diese mit einer Strömung zu verbinden.
Die Ergebnisse dazu stellen wir im Folgenden vor.

Wir haben wieder auf $\Omega = [0,1] \times [0,1] \times [0,1]$ das gleiche Setting wie in 4.4.1 verwendet:
$T = \quad [0, 40]$
$k_i = \quad 1 \qquad i = 1 \dots 4$
$B = \quad 3.4$
$A = \quad 1$
$D_{C_1} = \quad 0.002$
$D_{C_2} = \quad 0.002$
mit den Anfangskonzentrationen:

$$\begin{aligned} C_1(x_1, x_2, x_3, 0) &= 2 + 0.25x_2 \\ C_2(x_1, x_2, x_3, 0) &= 1 + 0.8x_1 \end{aligned} \tag{4.9}$$

und Neumann-0-Randbedingungen für die Chemikalien an jedem Rand.

Hinzu kommt nun unser in 3.7 beschriebenes Kopplungsmodell. Wir haben dafür

$$\gamma_1 = 3.8$$

$$C_1^{ref} = 0$$

als Parameter verwendet.

Als Volumenkräfte haben wir

$$\vec{G} = \begin{pmatrix} 0 \\ -9.81 \\ 0 \end{pmatrix}$$

gesetzt.

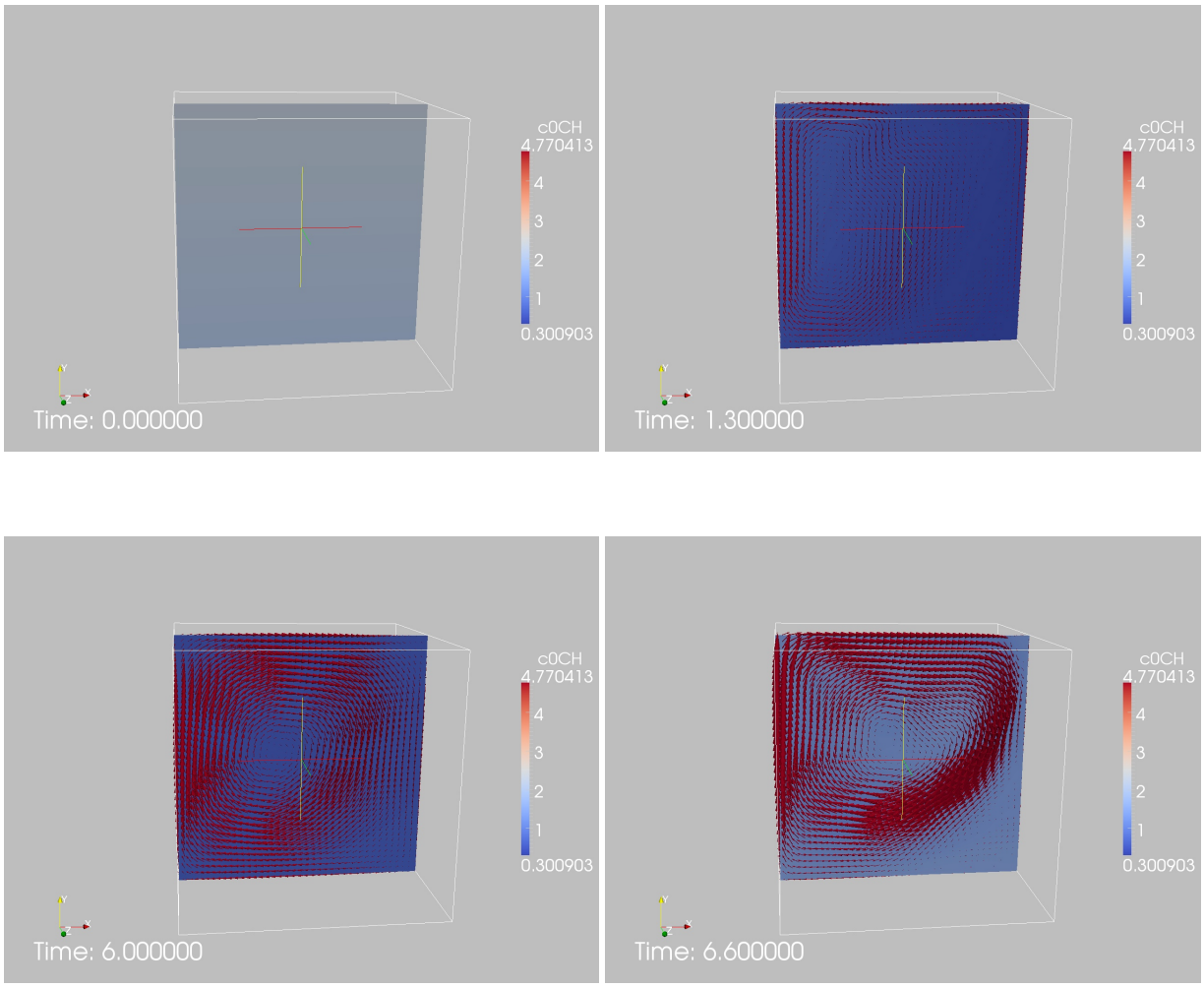
Für das Fluid galt an allen Rändern die slip-Randbedingung.

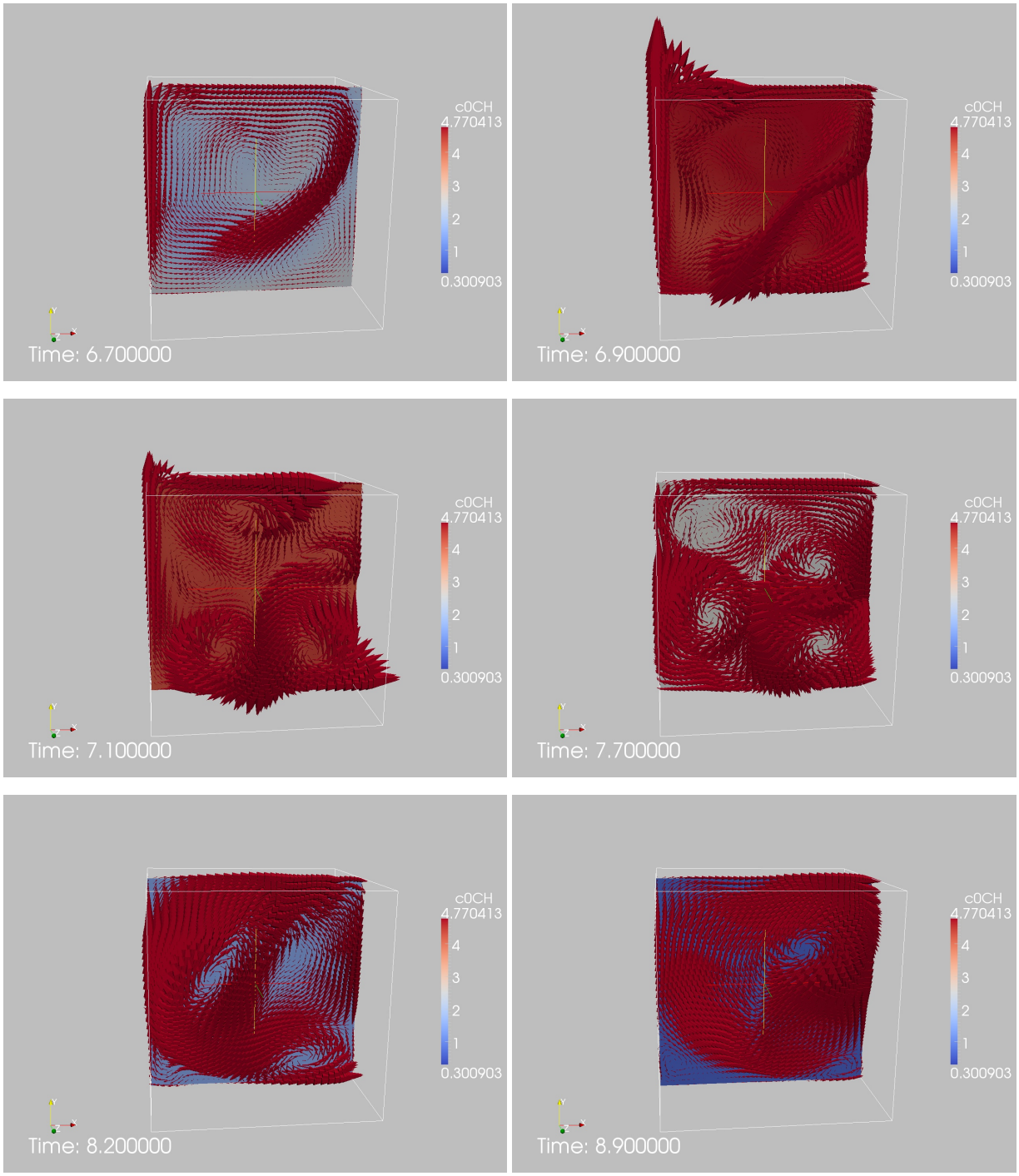
Die weiteren Parameter für die Berechnung sind in 4.2 dargestellt.

Nachfolgend sind unsere Simulationsergebnisse hierzu dargestellt. Zu sehen ist das Geschwindigkeitsfeld vor dem Hintergund eines slices durch die Konzentration C_1 in der x-y-Ebene:

	Auflösung	$40 \times 40 \times 40$
δt_S	Zeitschrittweite Strömung	0.002
ϵ	Genauigkeit des Poissonlösers	1.0e-10
$itermax$	Druckiterationen	300
T_{end}	Simulationsende (sec)	40
$\vec{G}$	Volumenkräfte	$\vec{0}$
	Diskretisierung der konvektiven Terme	VONOS
	Drucklöser	PCG
	Zeitdiskretisierung (Strömung)	Euler 1.Ordnung
	Reaktionslöser	implizit
	TOL	1.0e-9
	Nmax	300
δt_C	Zeitschrittweite Chemie-Berechnung	0.002

Tabelle 4.2.: Parameter zur Berechnung der 3D-Reaktion mit aktiver Kopplung





Wir initialisieren alle Geschwindigkeiten mit Null und haben slip-Randbedingungen - also insbesondere keine inflow/outflow-Bedingungen. Daher wäre das Geschwindigkeitsfeld ohne die Wirkung der Kopplung konstant Null.
Wie man sieht, erzeugen die Änderungen der Konzentration jedoch bereits nach kurzer Simulationszeit einen Einfluß auf das Geschwindigkeitsfeld. Dieser nimmt nochmals deutlich zu, wenn wir uns den Zeitpunkten nähern, an denen die Konzentrationen C_1 und C_2

ihr Maximum beziehungsweise Minimum erreichen und umschwanken.
Je nach Wahl der Kopplungsparameter gestaltet sich das Geschwindigkeitsfeld unterschiedlich. In den Anhängen A.1 und A.2 sind weitere Ergebnisse für andere Parameter zu finden.
Somit haben wir das Ziel, eine Rückkopplung der Reaktion auf die Strömung zu simulieren, erreicht.

4.4.3. 3D: Einfluß eines Rührers in einer oszillierenden Reaktion mit anfangs getrennten Chemikalien

Wir haben nun zusätzlich das in NaSt3DGPF bereits bestehende Feature der Fluid-Struktur-Interaktionen verwendet, um einen bewegten, sich drehenden Stab in das Gebiet zu setzen. Dadurch erzielen wir die Simulation eines Magnetrührers, wie sie in chemischen Laboren verwendet werden.

Wir haben die gleichen Bedingungen und Parameter verwendet wie in der vorherigen Simulation, 4.4.2, bis darauf, dass die Anfangskonzentrationen "getrennt" wurden:

$$\begin{aligned} C_1(x_1, x_2, x_3, 0) &= 2 + 0.25x_2 \qquad \text{für } 0.5 < x_2 <= 1.0 \\ C_1(x_1, x_2, x_3, 0) &= 0 \qquad\qquad\quad \text{sonst} \end{aligned} \tag{4.10}$$

$$\begin{aligned} C_2(x_1, x_2, x_3, 0) &= 1 + 0.8x_1 \qquad \text{für } 0 <= x_2 <= 0.5 \\ C_2(x_1, x_2, x_3, 0) &= 0 \qquad\qquad\quad \text{sonst} \end{aligned} \tag{4.11}$$

Deshalb sind die Ergebnisse in diesem Abschnitt nicht direkt mit den Simulationen aus den beiden vorherigen Abschnitten vergleichbar. Im Anhang A.3 ist für die hier betrachtete Reaktion mit getrennten Chemikalien nochmal eine Simulation ohen Rührer zu sehen.

Hier wurde auch wieder die aktive Kopplung mit den Parametern

$\gamma_1 = 3.8$

$C_1^{ref} = 0$

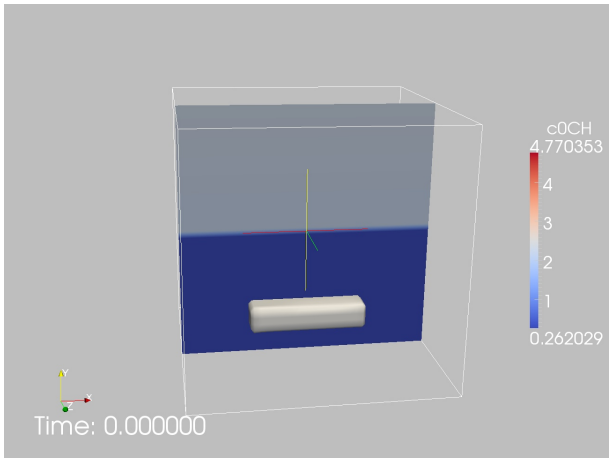
verwendet.
Im Anhang A.4 ist eine weitere Simulation des magnetischen Rührers mit nur passiver Kopplung an das Strömungsfeld zu sehen.

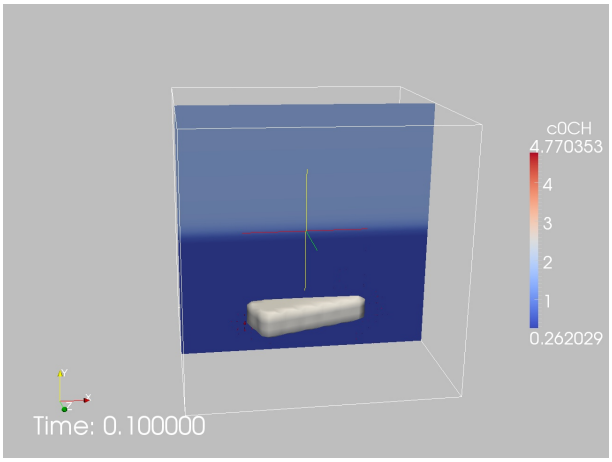
Als Rührer haben wir einen Stab mit den Ausmaßen

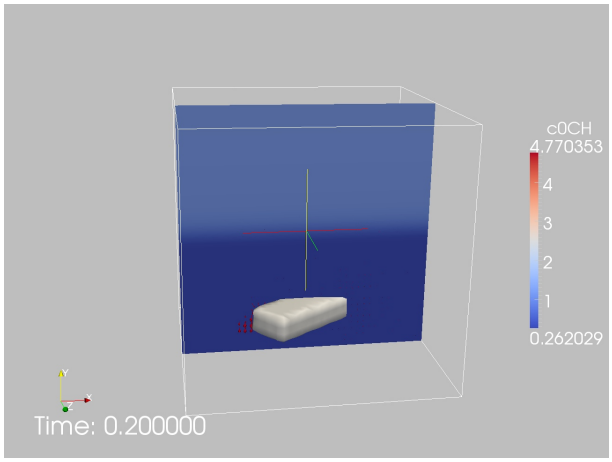
$0.1 \times 0.5 \times 0.2$ (Höhe $\times$ Breite $\times$ Tiefe)

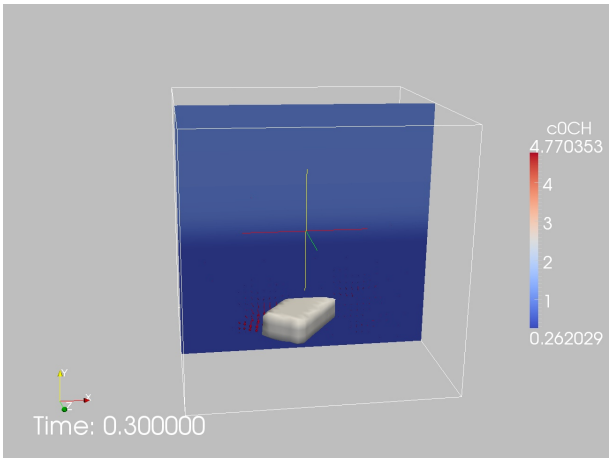
gewählt.

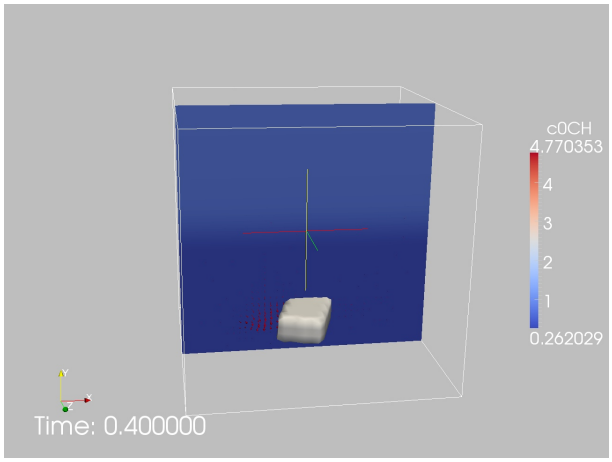
Für die so beschriebene Situation haben wir in C_1 erhalten:

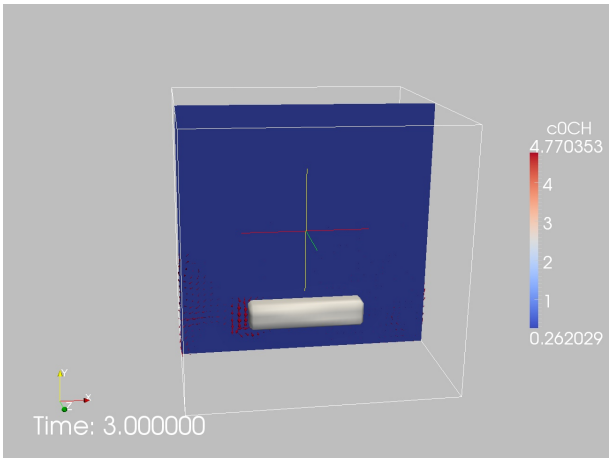

c0CH
4.770353
4
3
2
1
0.262029
Time: 0.000000

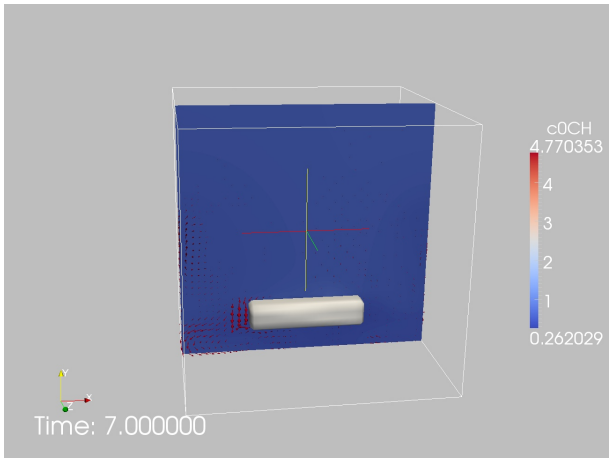

c0CH
4.770353
4
3
2
1
0.262029
Time: 0.100000

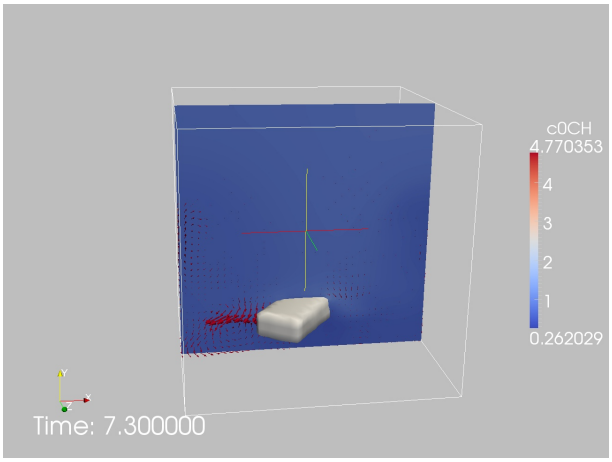

c0CH
4.770353
4
3
2
1
0.262029
Time: 0.200000

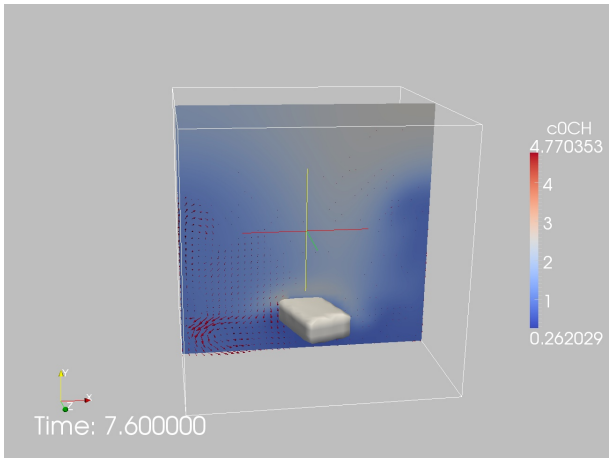

c0CH
4.770353
4
3
2
1
0.262029
Time: 0.300000

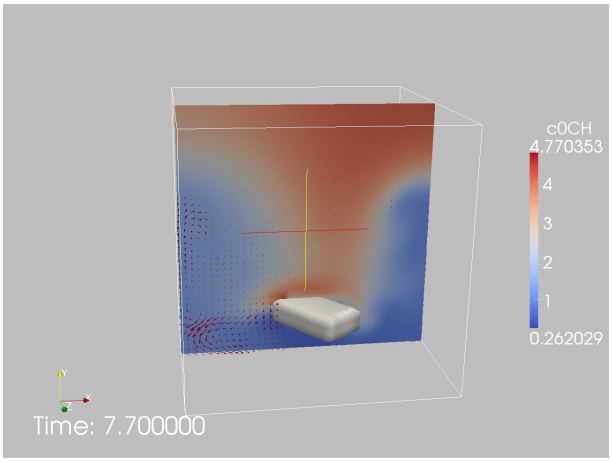

c0CH
4.770353
4
3
2
1
0.262029
Time: 0.400000

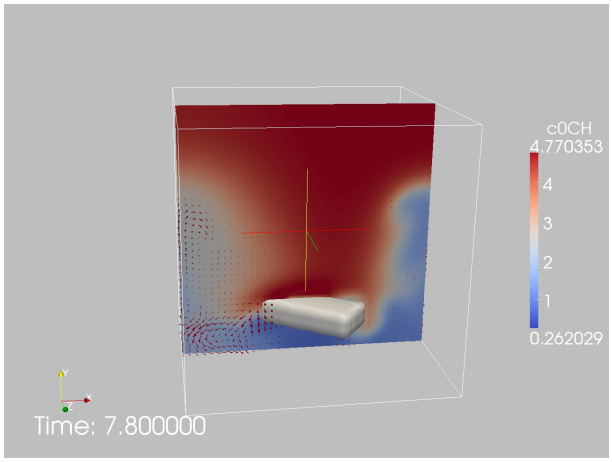

c0CH
4.770353
4
3
2
1
0.262029
Time: 3.000000

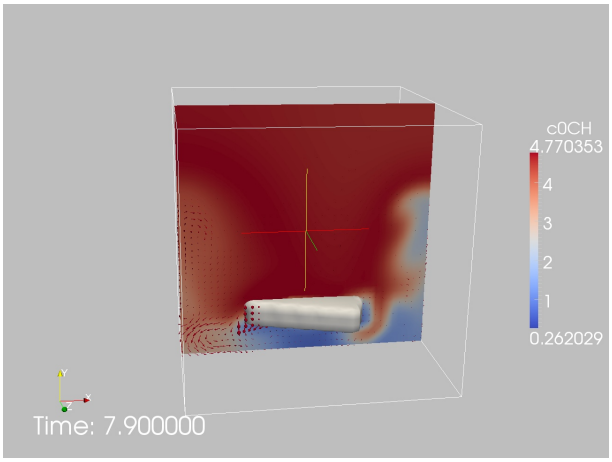

c0CH
4.770353
4
3
2
1
0.262029
Time: 7.000000

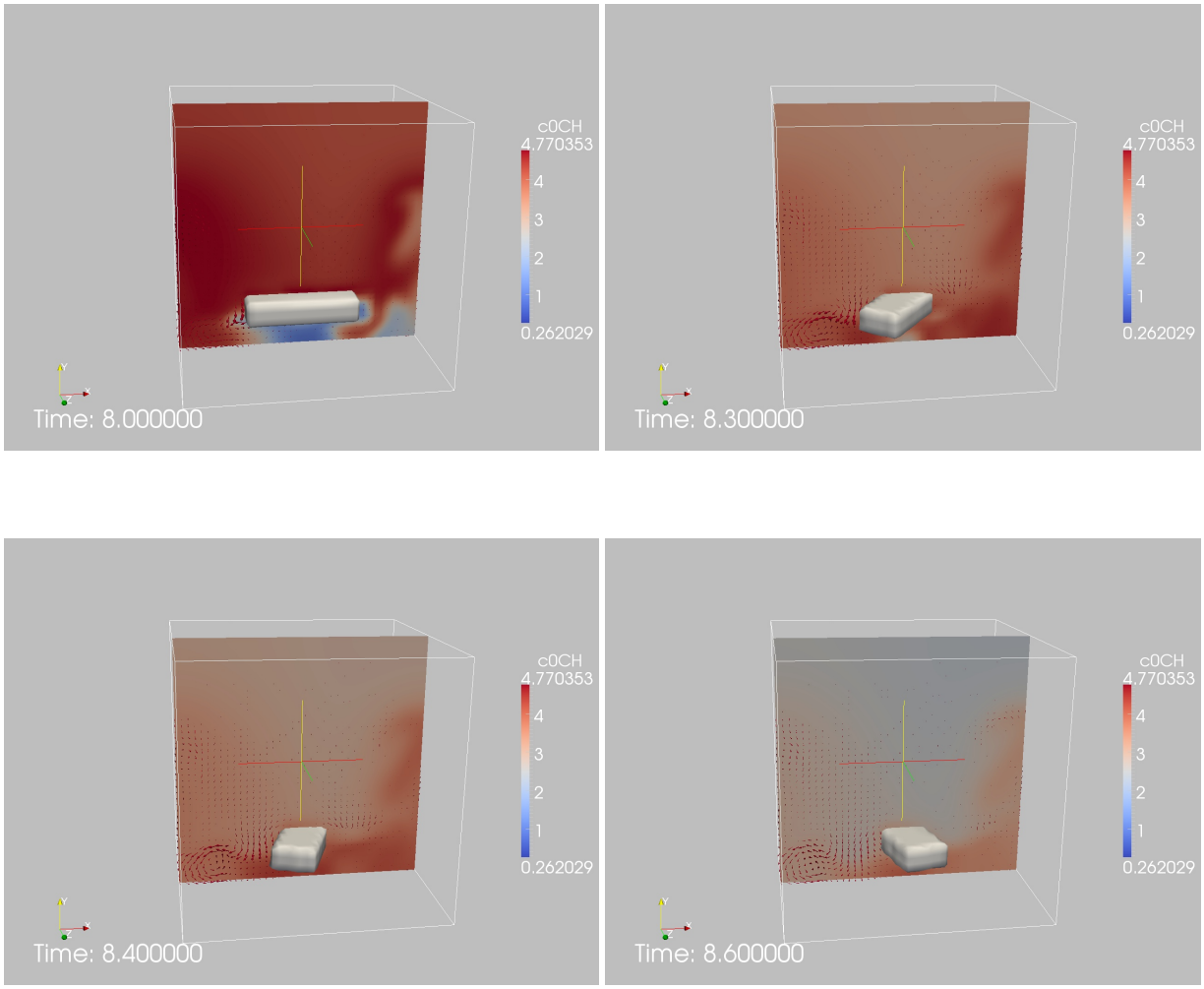

c0CH
4.770353
4
3
2
1
0.262029
Time: 7.300000


c0CH
4.770353
4
3
2
1
0.262029
Time: 7.600000

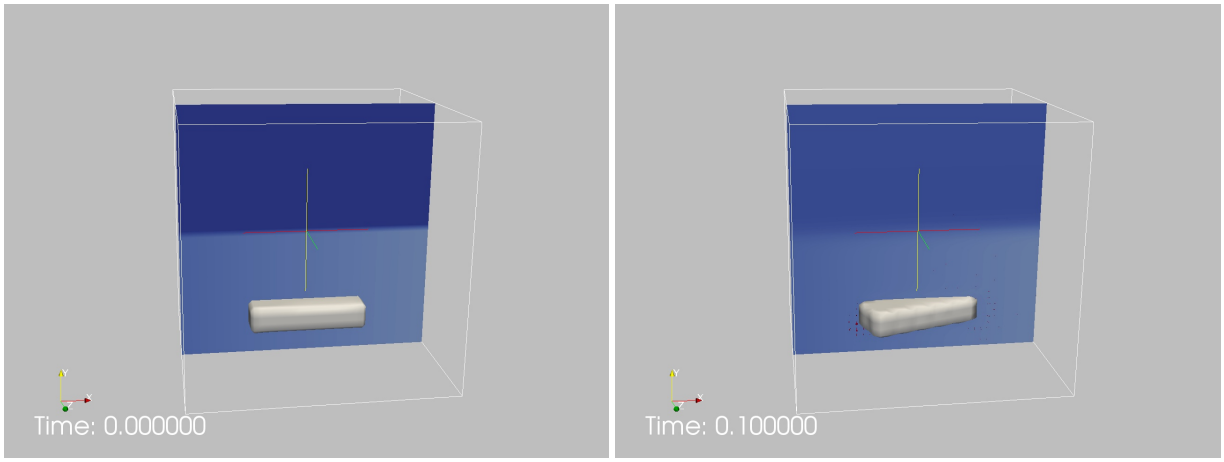

c0CH
4.770353
4
3
2
1
0.262029
Time: 7.700000

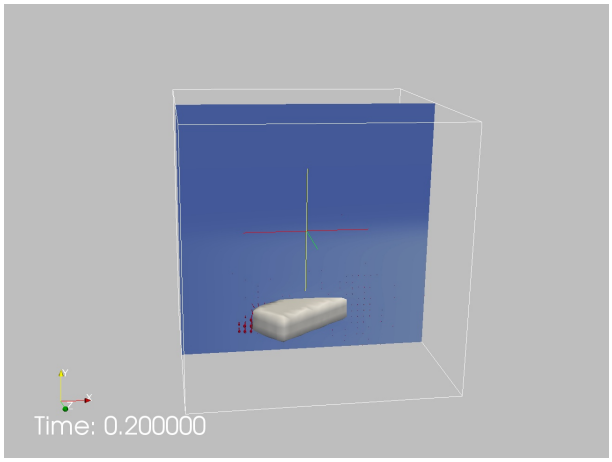

c0CH
4.770353
4
3
2
1
0.262029
Time: 7.800000

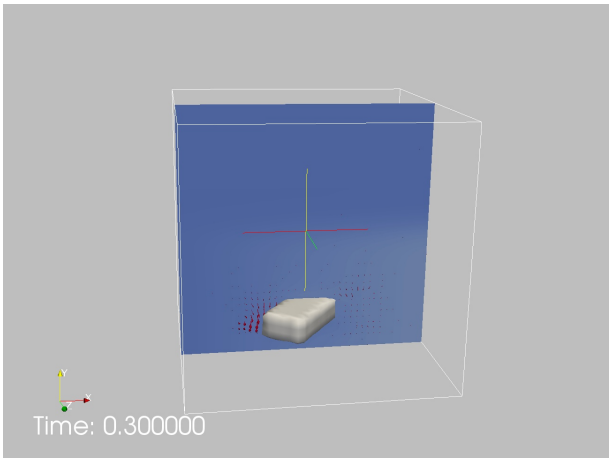

c0CH
4.770353
4
3
2
1
0.262029
Time: 7.900000

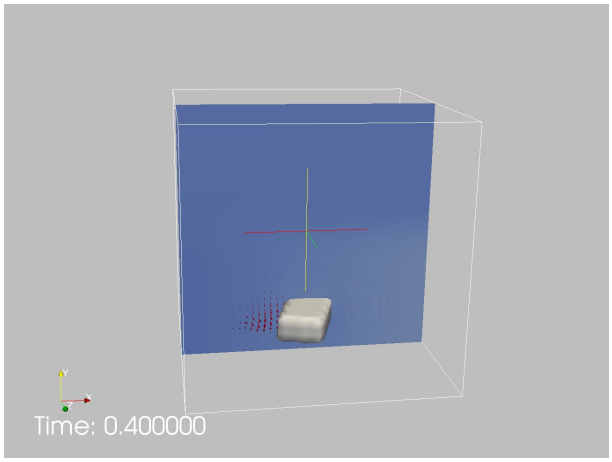


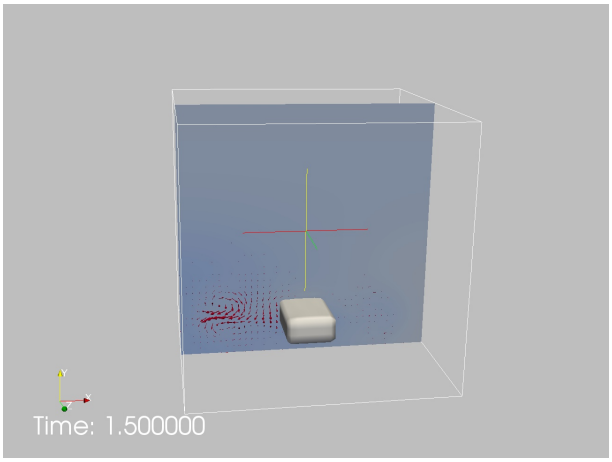
Die Konzentration C_2 verhält sich wie folgt:

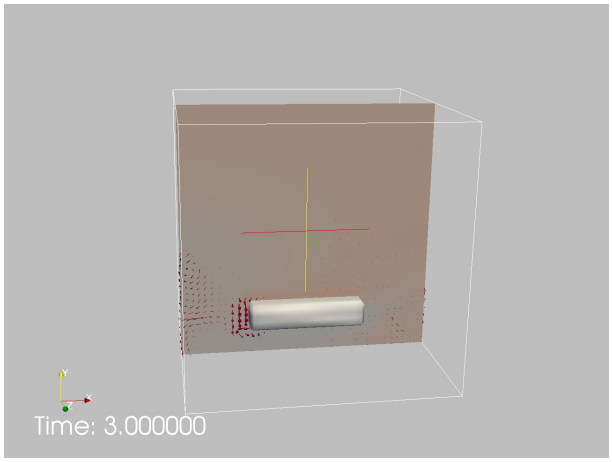


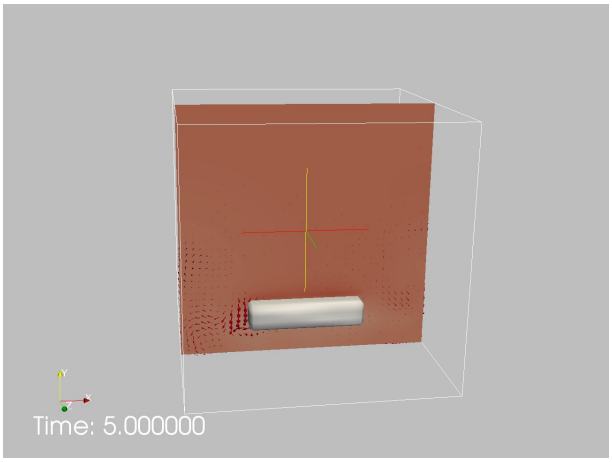

Time: 0.200000

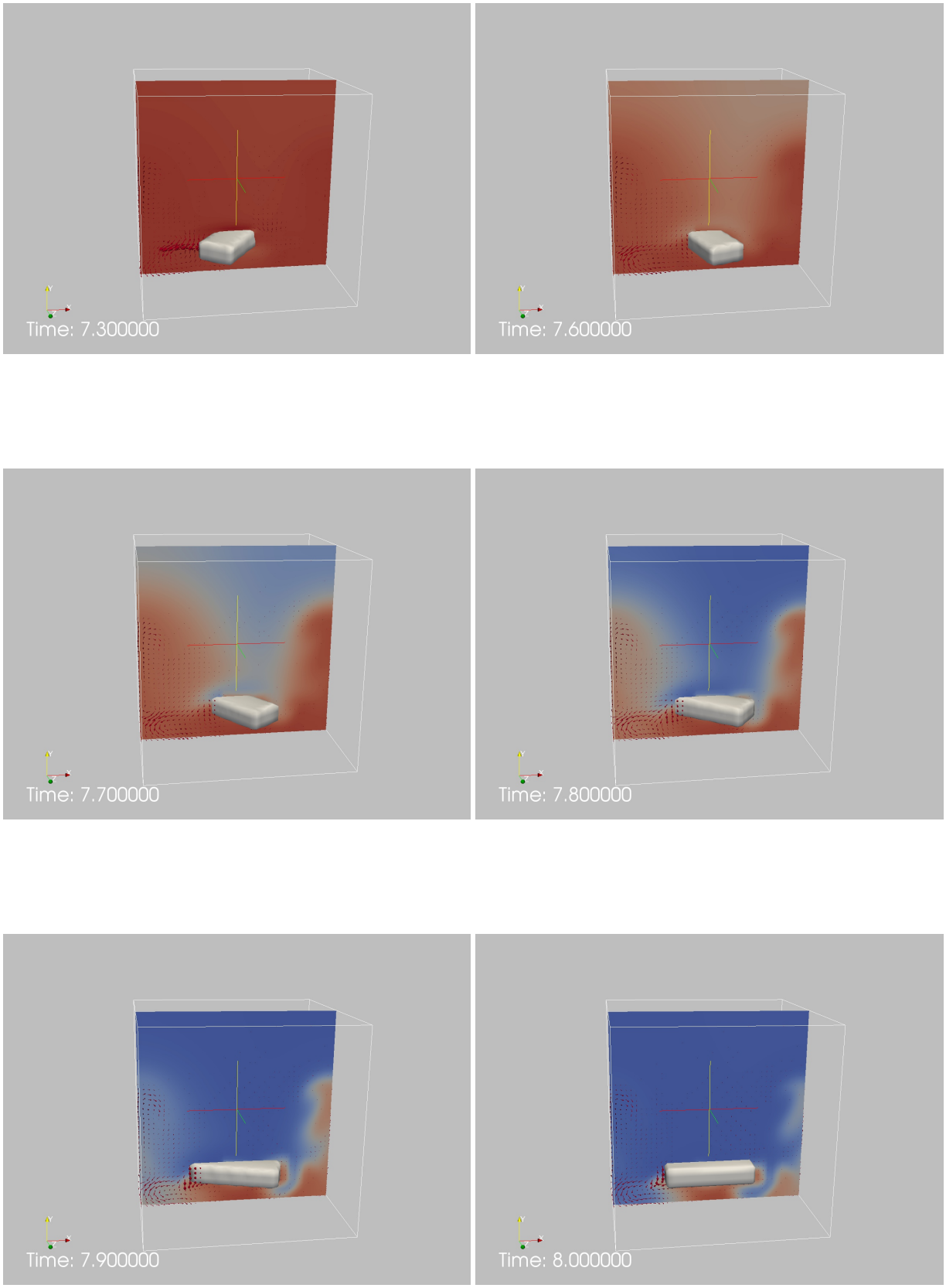

Time: 0.300000

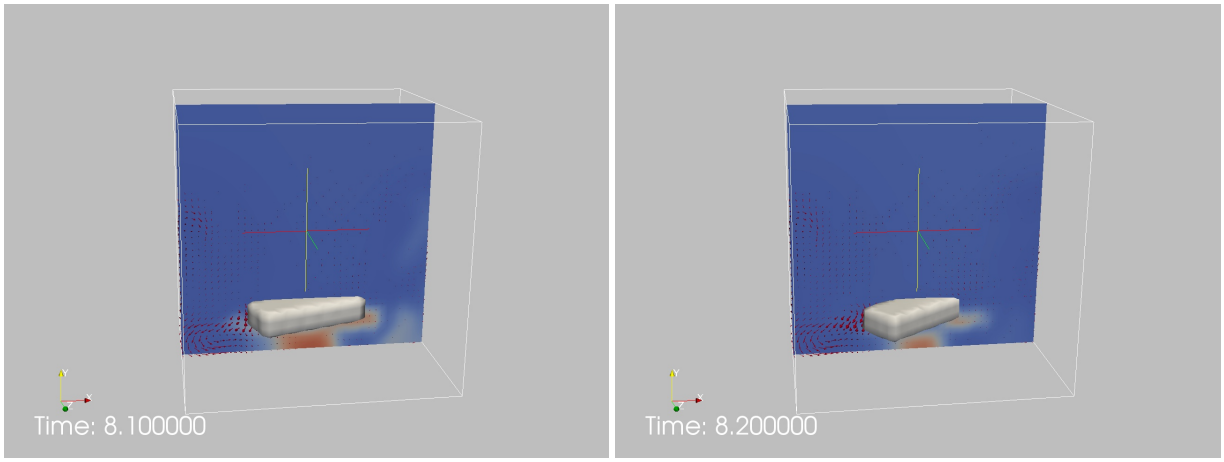

Time: 0.400000


Time: 1.500000


Time: 3.000000


Time: 5.000000


Time: 7.300000
Time: 7.600000
Time: 7.700000
Time: 7.800000
Time: 7.900000
Time: 8.000000



Diese Ergebnisse zeigen, dass es mit NaSt3DGPF und unserer Implementierung des Brusselators prinzipiell möglich ist, einen Rührfisch zu simulieren.

Vergleich: getrennte Chemikalien - Reaktion mit und ohne Rührer

Der Vergleich dieser Ergebnisse mit einer Simulation dieses Beispieles ohne die Rührgeometrie zeigt, dass sich bei unserer Wahl der Parameter die Reaktion nicht beschleunigt hat. Die Ausbreitung des Konzentrationsprofils ist jedoch anders.

Zur Gegenüberstellung zeigen wir hier zum Zeitpunkt $T = 7.7$ zweimal die Chemikalie C_1. Zum einen sehen wir die unbeeinflußte Reaktion, ohne Rührer und nur passiv an den Strömungscode gekoppelt. Hier findet Transport nur aufgrund der Diffusion statt. Zum anderen sehen wir die gleiche Reaktion mit dem Rührer, wobei auch die aktive Kopplung eingesetzt war.[2]

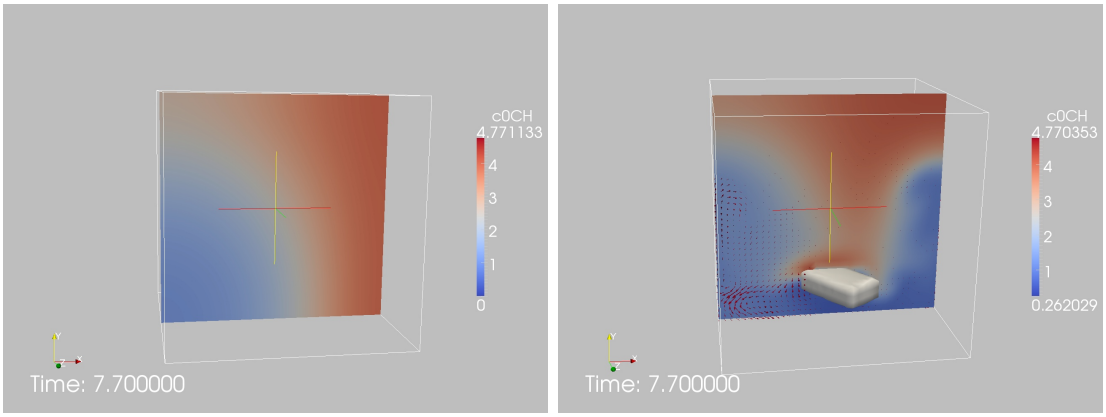


Abbildung 4.35.: Links: unbeinflusste Reaktion; Rechts: mit Rührfisch (aktiv gekoppelt)

Für eine andere Wahl der Reaktionsparameter wäre jedoch zu erwarten, dass die Reaktion mit Rührer schneller vonstatten geht als ohne.

Ein Vergleich des Verlaufs am Beginn der Reaktion zeigt, dass sich die in unserem Beispiel anfangs getrennten die Konzentrationen in beiden Fällen fast gleich schnell über das Gebiet hinweg angleichen (siehe Abbildung).

Deshalb wäre es hier sicherlich ein sinnvoller Ansatz, die Anfangswerte der Konzentrationen so zu gestalten, dass sich eine stärkere Trennung ergibt oder die Diffusion (hier: $D_{C_1} = 0.002$) weiter zu verkleinern. Weiterhin kann man die Bewegungsgeschwindigkeit

[2]Eine Darstellung weiterer Simulationszeitpunkte für diese Reaktion ohne Rührer sowie mit Rührer aber nur passiv gekoppelt findet sich in den Anhängen A.3 und .

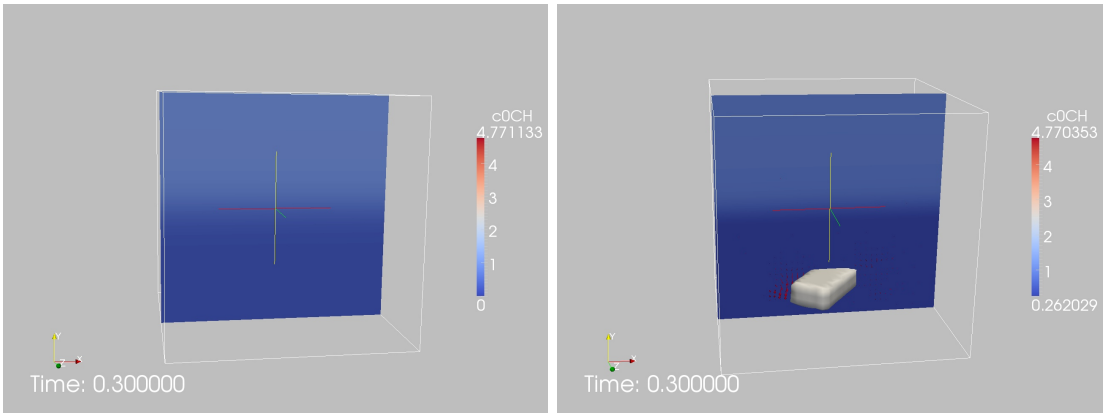


Abbildung 4.36.: Links: unbeinflusste Reaktion; Rechts: mit Rührfisch (aktiv gekoppelt)

des Rührers variieren und diesen nach der Vermischung auch anhalten. Wir haben für unsere Berechnungen eine relativ schnelle Geschwindigkeit gewählt, den realen Apparaturen nachempfunden.

Zusammenfassend zu diesem Kapitel können wir jedoch sagen, dass sich in allen Fällen ein realistisches Verhalten zeigt, wie es bei einer Reaktion von Art der BZ-Reaktion zu erwarten ist.

5. Zusammenfassung und Ausblick

5.1. Zusammenfassung

In dieser Arbeit beschäftigten wir uns mit der Modellierung und Simulation chemischer Reaktionen in Verbindung mit Fluidströmungen.

Unser Hauptziel war die aktive und passive Kopplung eines Modelles für chemische Reaktionen an den Strömungslöser. Hinzu kam als weiteres Ziel das Simulieren eines Rührers, einer Vorrichtung wie sie bei realen chemischen Experimenten zum Vermischen von Stoffen eingesetzt wird.

Wir haben zunächst einen Überblick über verschiedene Modelle zur Beschreibung chemischer Reaktionen gegeben und sind dann genauer auf das für unsere Zwecke ausgewählte Brusselator-Modell eingegangen.

Das bei der Diskretisierung des Brusselators in Ort und Zeit entstehende Gleichungssystem wurde in Kapitel 3 analysiert. Es wurde eine Splitting-Methode gewählt um das Gleichungssysytem numerisch zu lösen. Dies ermöglichte uns die entkoppelte Behandlung von Reaktion und Diffusion. Dadurch wurde eine größere Flexibilität in der Wahl der Methode zur Zeitdiskretisierung erreicht. Zur Ortsdiskretsierung setzten wir ein äquidistantes Gitter in zwei beziehungsweise drei Raumdimensionen ein.

Weiterhin haben wir unsere Implementierung eingehend genauer beschrieben. Für den Reaktionsteil und für den Diffusionsteil wurde zunächst jeweils ein explizites und ein implizites Eulerverfahren in MATLAB umgesetzt. Dies wurde für den zweidimensionalen Fall getan.
Anschließend wendeten wir uns der dreidimensionalen Implementierung des Modelles in Verbindung mit NaSt3DGPF zu. Das gewählte Splittingverfahren ermöglichte uns, für die Diffusion auf einen bestehenden Programmteil von NaSt3DGPF zurückzugreifen. Somit verblieb noch die Erweiterung der Löser der Reaktionsberechnung. Dies geschah, in dem wir das Diskretisierungsgitter in natürlicher Weise auf die zusätzliche Raumdimension erweitert haben. Wir haben also unser explizites und implizites Verfahren derart angepasst und in den NaSt3DGPF-Code eingebaut.

Um die Stabilität zu gewährleisten, war es erforderlich für die explizite Reaktionsberechnung eine zusätzliche Chemie-CFL-Bedingung einzuführen, die die maximale Konzentrationszunahme in einer Zelle beschränkt. Bei Verwendung des impliziten Verfahrens war dies

nicht notwendig. Zu beachten war allerdings auch, dass die vorhandene skalare Transportroutine, die explizit berechnet wird, ebenfalls bereits einer CFL-Bedingung unterliegt.

Desweiteren wurde eine separate Zeitschrittweite für die Berechnung der Konzentrationen eingeführt. Dies gab uns die wichtige Möglichkeit, die Reaktion auf anderen Zeitskalen zu berechnen als die Strömung. Das heißt, im Falle dass wir die Reaktion zeitlich feiner auflösen wollen als die Strömung, dass mehrmals hintereinander eine Reaktionsberechnung durchgeführt wird, bevor wieder ein Strömungsschritt berechnet wird. Der umgekehrte Fall - mehrere Strömungsschritte bevor erneut die Reaktion betrachtet wird - ist auch möglich. Die Verwendung verschiedener Zeitschrittweiten war uns wichtig, da sie für viele Anwendungen sehr sinnvoll sein kann.

Nachdem wir das Brusselator-Modell auf diese Weise zuerst passiv an den Strömungscode gekoppelt haben, bestand jetzt die zentrale Aufgabe darin, eine Auswirkung der Konzentrationsänderungen auf das Geschwindigkeitsfeld zu erarbeiten.
Dazu wurde ein Modell ausgearbeitet, welches an die Boussinesq-Approximation für die Temperatur angelehnt ist. Da es sich bei den Chemikalien ebenso wie bei der Temperatur um skalare Werte handelt war dies ein vielversprechender Ansatz.
Da nun die Strömung einem Einfluß durch die Konzentrationen unterliegt, war es notwendig, die CFL-Bedingung für die konvektiven Terme anzupassen. Durch Hinzufügen eines entsprechenden Termes in die bestehende Bedingung für die konvektiven Terme haben wir den Einfluß der Konzentrationsänderungen berücksichtigt.

In Kapitel 4 wurden Konvergenzanalysen und Simulationsergebnisse vorgestellt. Dazu haben wir unsere zweidimensionale Implementierung anhand von verschiedenen Beispielen untersucht. Für den dreidimensionalen Fall war das in dieser Form nicht möglich, da uns dafür kein Beispiel bekannt ist, für das es eine analytische Lösung gibt. Wir gehen aber aufgrund der guten Ergebnisse im 2D-Fall davon aus, dass der 3D-Code ebenfalls gut funktioniert.

Dies wird untermauert durch die überzeugenden Simulationsergebnisse, die wir im zweiten Teil von 4 gezeigt haben. Wir konnten realistische Simulationen erzielen, die eine oszillierende Reaktion zeigen, welche der Beschreibung der realen BZ-Reaktion nahe kommt (vgl. etwa [Con82]).
Weiterhin haben wir unser Rückkopplungsmodell für verschiedene Parameter eingesetzt. Damit wurde erfolgreich eine Rückkopplung der Reaktion auf das Geschwindigkeitsfeld simuliert. Wir haben hier vorerst nur als einen modellhaften Ansatz die Machbarkeit anhand einiger Parametersätze gezeigt, das Auffinden von realistischen Materialparametern verbleibt noch als eine interessante Aufgabe für die Zukunft.

Abschließend wurde die Fluid-Struktur-Interaktion von NaSt3DGPF genutzt, um einen magnetischen Rührer zu simulieren. Somit kommen wir auf unser ursprüngliches Ziel zurück, welches erfolgreich erfüllt werden konnte.

5.2. Ausblick

Es gibt auf dem Gebiet der Simulation von chemischen Reaktionen viele weitere interessante Ansatzmöglichkeiten.

Wie in Kapitel 2 vorgestellt, gibt es verschiedene Modelle, die man zur Beschreibung chemischer Reaktionen benutzen kann. Mit unserer Implementierung des Brusselators können wir oszillierende Reaktionen simulieren, welche von hohem Interesse für die Industrie sind. Für noch breitere Anwendungsmöglichkeiten wäre es denkbar, Modelle für weitere Reaktionsarten zu implementieren.

Insbesondere wäre es auch interessant, Reaktionen mit mehr als zwei Spezies zu betrachten. Für Reaktionen mit einer größeren Anzahl (Zwischen-)Spezies könnten auch hochdimensionale Methoden zum Einsatz kommen. Hierzu gibt es bereits Arbeiten zur Reduktion von detaillierten chemischen Reaktionsmechanismen auf einfachere Systeme. Besonders hervorzuheben ist hier die ILDM-Methode (*intrinsic low dimensional manifold* - [MP92]) und ihre Weiterentwicklungen. Wir haben uns in der vorliegenden Arbeit für ein einfacher umzusetzendes Modell entschieden, da unser Schwerpunkt auf der Kopplung des Reaktionsmodelles mit dem CFD-Löser liegt. Aufgrund des hohen Aufwandes der Implementierung, der mit Modellen wie ILDM verbunden ist, wurde dieser Ansatz hier nicht verfolgt, wäre jedoch ein lohnenswertes Ziel.

Zur Verbesserung unseres im Rahmen dieses Arbeit erstellten Codes könnte eine Überlegung sein, ein implizites Verfahren für das Gesamtsystem zu implementieren. In der aktuellen Implementierung wird immer eine explizite Transportroutine eingesetzt, die einer CFL-Bedingung unterliegt, so dass die Wahl unserer Chemie-Zeitschrittweite beschränkt ist. Dies könnte durch Einsatz eines impliziten Verfahrens vermieden werden. Zu beachten ist allerdings, dass auch wenn die Chemie-CFL-Bedingungen wegfallen, wir weiterhin die CFL-Bedingungen für die Strömungsberechnung haben (welche im Falle der aktiven Kopplung zusätzlich durch die Chemikalien beeinflußt werden).
Dadurch sind wir also mit der Schnelligkeit der Gesamtsimulation begrenzt. Hier wäre noch genauer abzuwägen ob eine weitere Verbesserung der Chemie-Zeitschrittweiten einen tatsächlichen Vorteil bringen würde.

Bezüglich unseres Modells zur aktiven Rückkopplung wären weitere Untersuchungen dazu, wie man die Wahl der Parameter realistischer gestalten kann eine nützliche Aufgabe.

In Kapitel 4.4.3 wurde ein magnetischer Rührer, wie er tatsächlich bei chemischen Experimenten eingesetzt wird, simuliert. Es gibt sicher noch viele verschiedene Aufbauten, deren Simulation auch von Interesse wäre.

Ein spannender Fall wäre insbesondere auch der häufig - gerade auch im Großmaßstab - verwendete Blasenreaktor. Dieser stellt allerdings eine Herausforderung an die Numerik dar. Bereits die physikalisch realistische Simulation einer einzelnen Blase in 3D ist sehr schwierig.

Da NaSt3DGPF bereits das Simulieren von Sedimenttransport beherrscht, wäre eine Verbindung hiermit naheliegend. Dafür gibt es viele Anwendungen, der Bereich des Umweltschutzes (Wasserverschmutzung) ist nur eine davon.
In diesem Zusammenhang könnte man auch das Phänomen der Ausfällung einbeziehen: aus den gelösten chemischen Stoffen könnte sich bei der Reaktion - wenn etwa eine bestimmte Konzentration erreicht ist - ein Feststoff bilden. Dessen Partikel können dann als Sedimente vom Geschwindigkeitsfeld transportiert werden.
Die Realisierung von Ausfällung bei chemischen Reaktionen ist eine schwierige, aber lohnenswerte Aufgabe.

Somit gibt es noch viele interessante und lohnenswerte Möglichkeiten, die Arbeiten zu diesem Thema zu verbessern und fortzusetzen.

Danksagung

Ich bedanke mich bei Herrn Prof. Dr. Griebel für die Übergabe des spannenden Themas und seine Betreuung und viele Anregungen. Bei Herrn Prof. Dr. Schweitzer möchte ich mich für die Übernahme der Zweitkorrektur der Arbeit bedanken.

Ein weiteres Dankeschön geht an Dr. Christian Rieger und Peter Zaspel für ihre geduldige Hilfe und zahlreiche wertvolle Ideen und Denkanstöße.

Außerdem bedanke ich mich bei Alisa Maloglazova für ausführliches Korrekturlesen. Dank für weitere Korrekturen geht auch an Michelle Steimels.

A. Anhänge

Wir wollen ergänzend noch einige weitere Simulationsergebnisse aufführen.

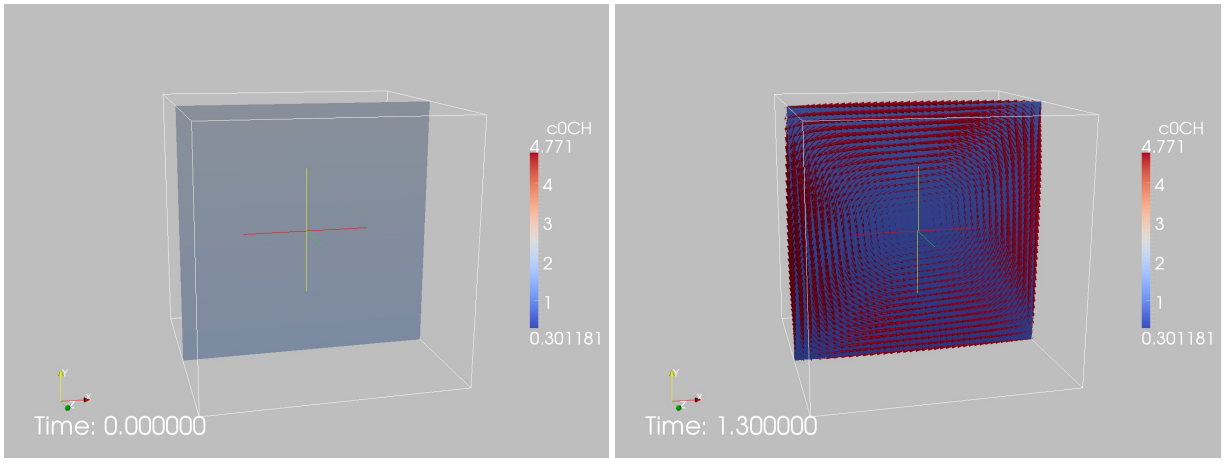
A.1. Aktive Kopplung - Parameterset II

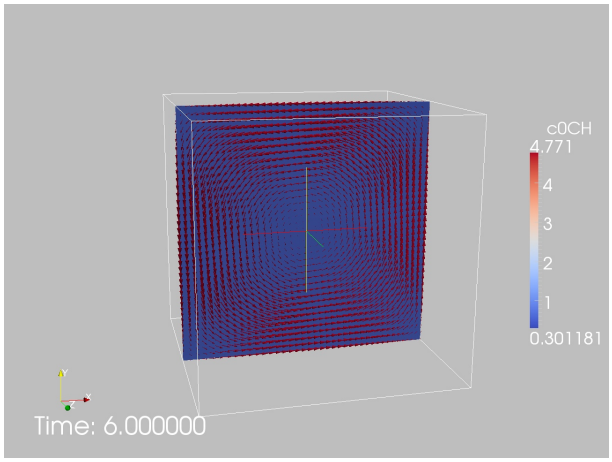
Zusätzlich zu der in Kapitel 4.4.2 abgebildeten Rechnung haben wir noch weitere Simulationen zur Rückkopplung durchgeführt.
Mit

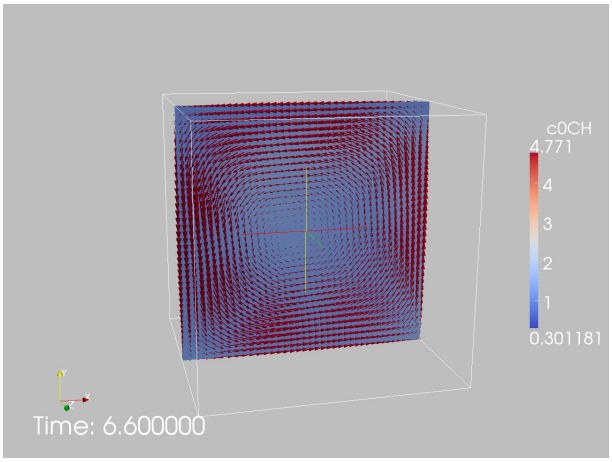
$$\gamma_1 = 0.8$$

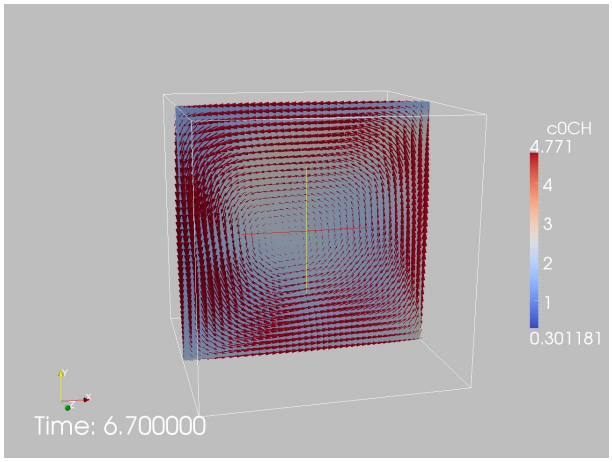
$$C_1^{ref} = 0$$

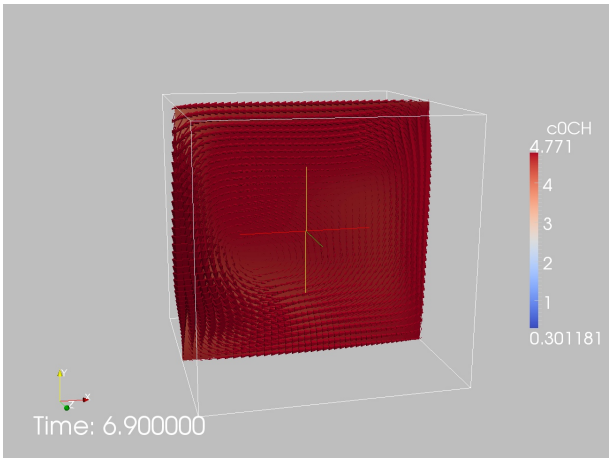
und ansonsten gleichen Parametern wie unter 4.4.2 haben wir folgende Ergebnisse erhalten. Dargestellt ist wieder das Geschwindigkeitsfeld mit einem Slice durch C_1:

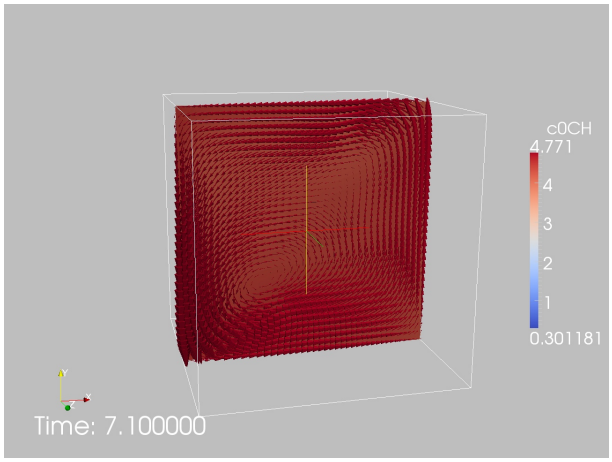


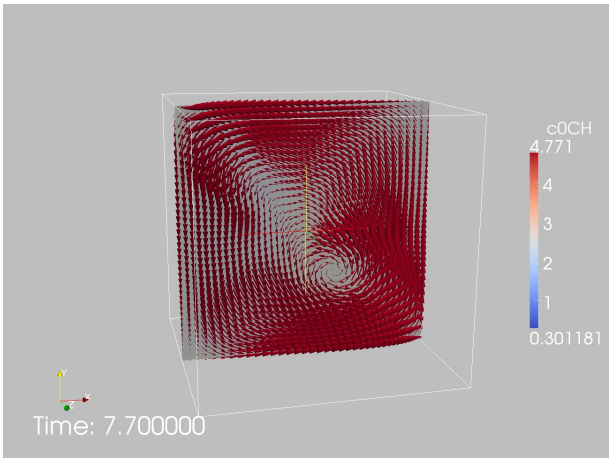

c0CH
4.771
4
3
2
1
0.301181
Time: 6.000000

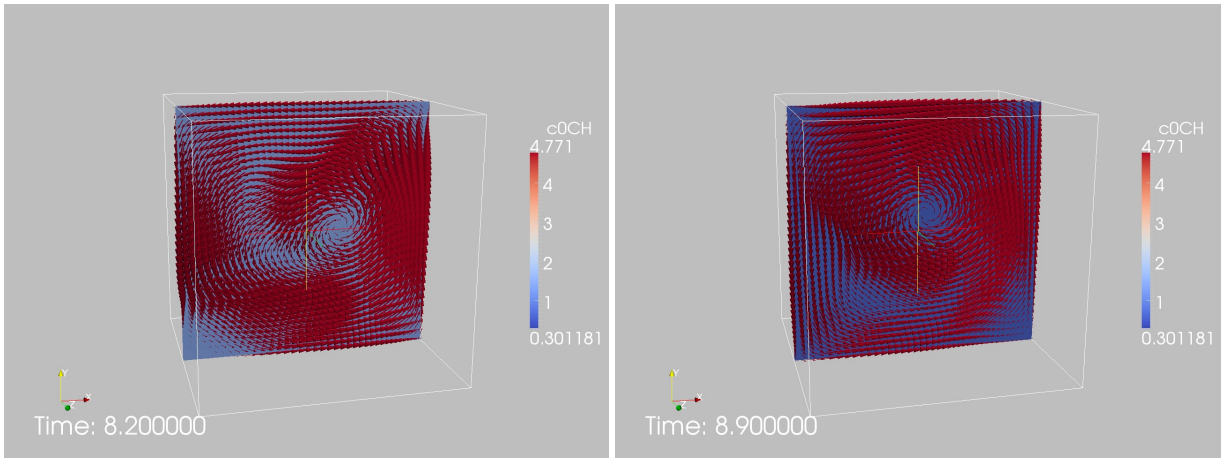

c0CH
4.771
4
3
2
1
0.301181
Time: 6.600000


c0CH
4.771
4
3
2
1
0.301181
Time: 6.700000


c0CH
4.771
4
3
2
1
0.301181
Time: 6.900000


c0CH
4.771
4
3
2
1
0.301181
Time: 7.100000


c0CH
4.771
4
3
2
1
0.301181
Time: 7.700000


c0CH
4.771
4
3
2
1
0.301181
Time: 8.200000
c0CH
4.771
4
3
2
1
0.301181
Time: 8.900000

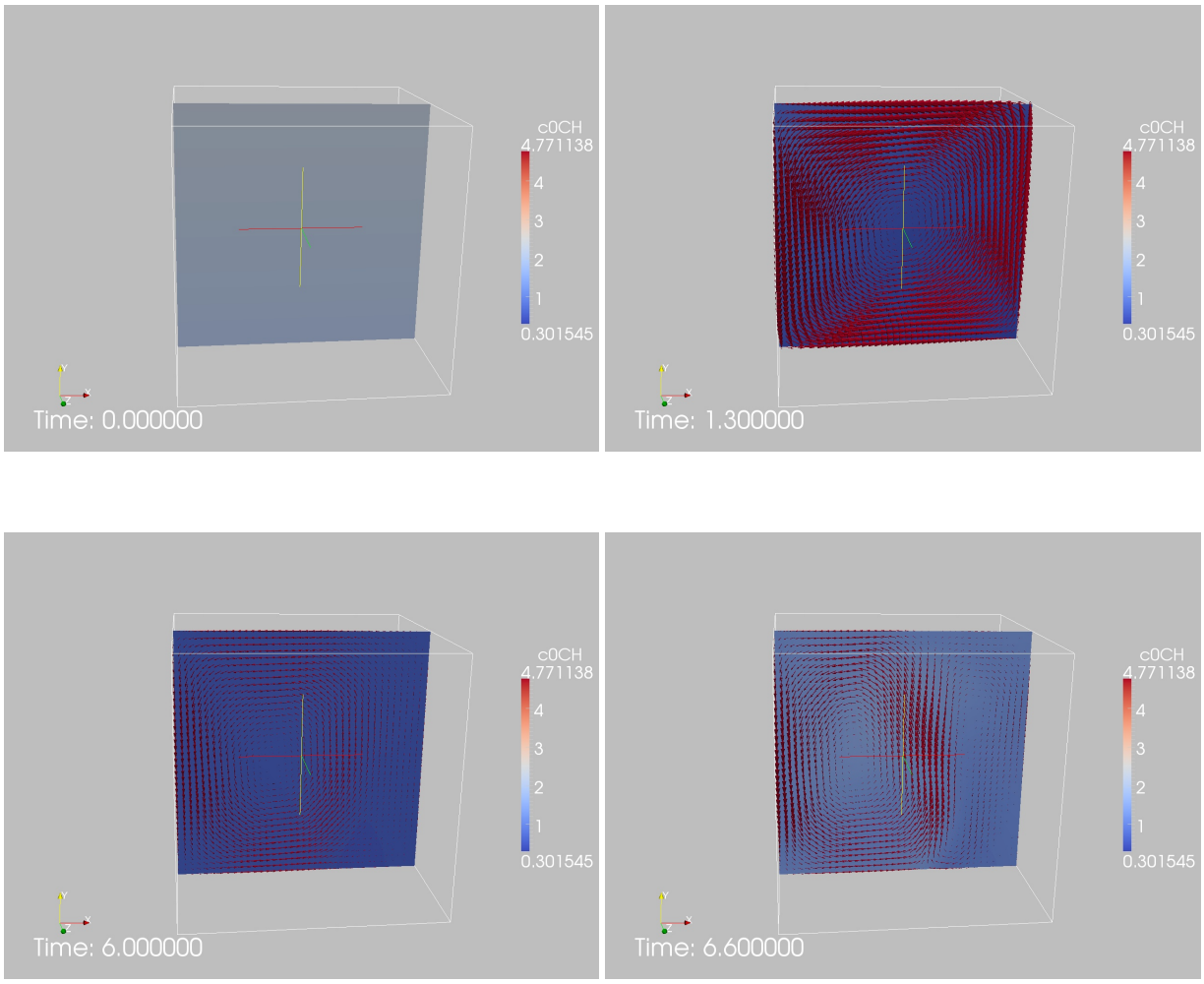
A.2. Aktive Kopplung - Parameterset III

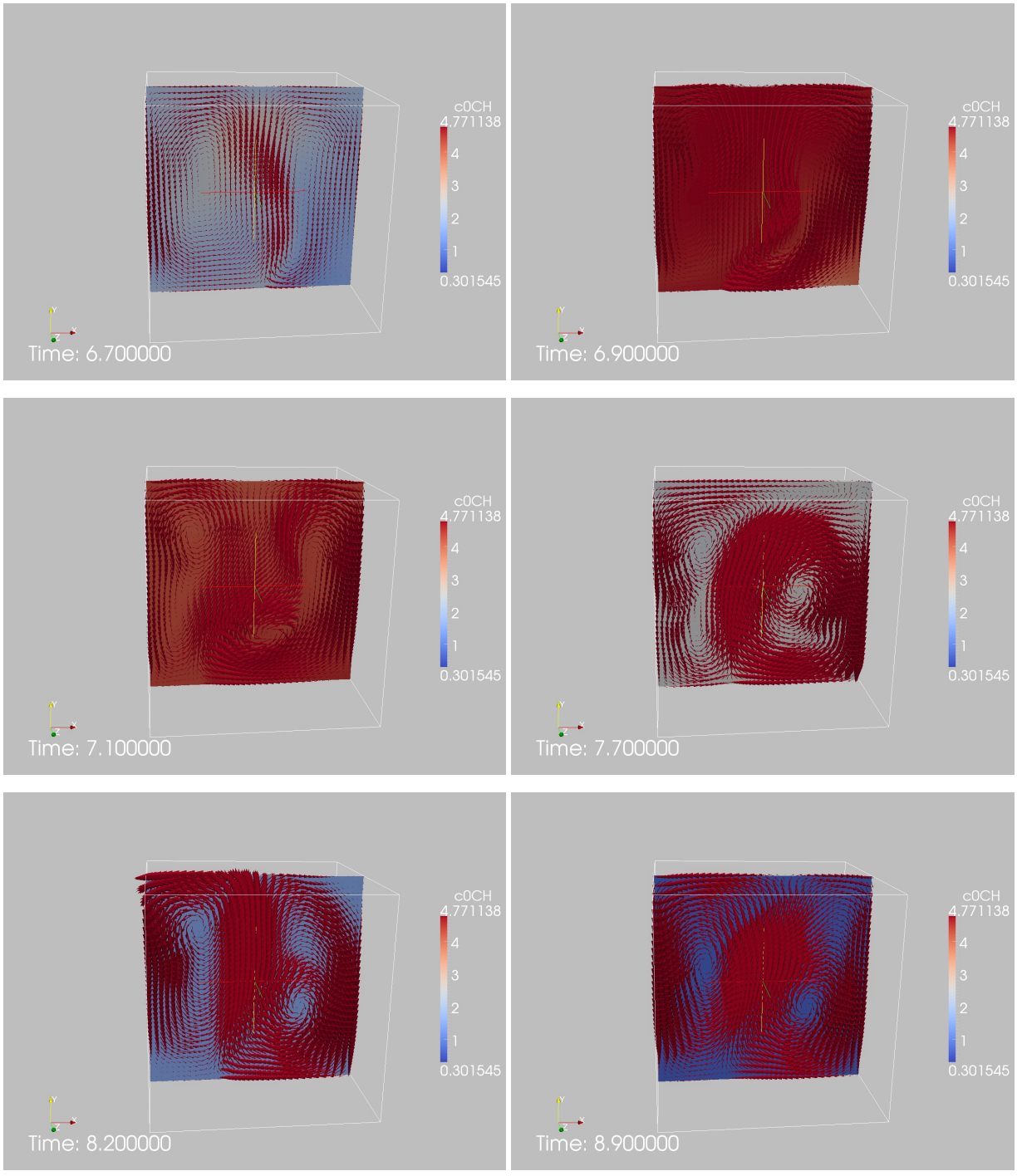
Ein weiteres Ergebnis, das wir basieren auf 4.4.2 durch Veränderung der Kopplungsparameter als

$$\gamma_1 = 1.7$$

$$C_1^{ref} = 0$$

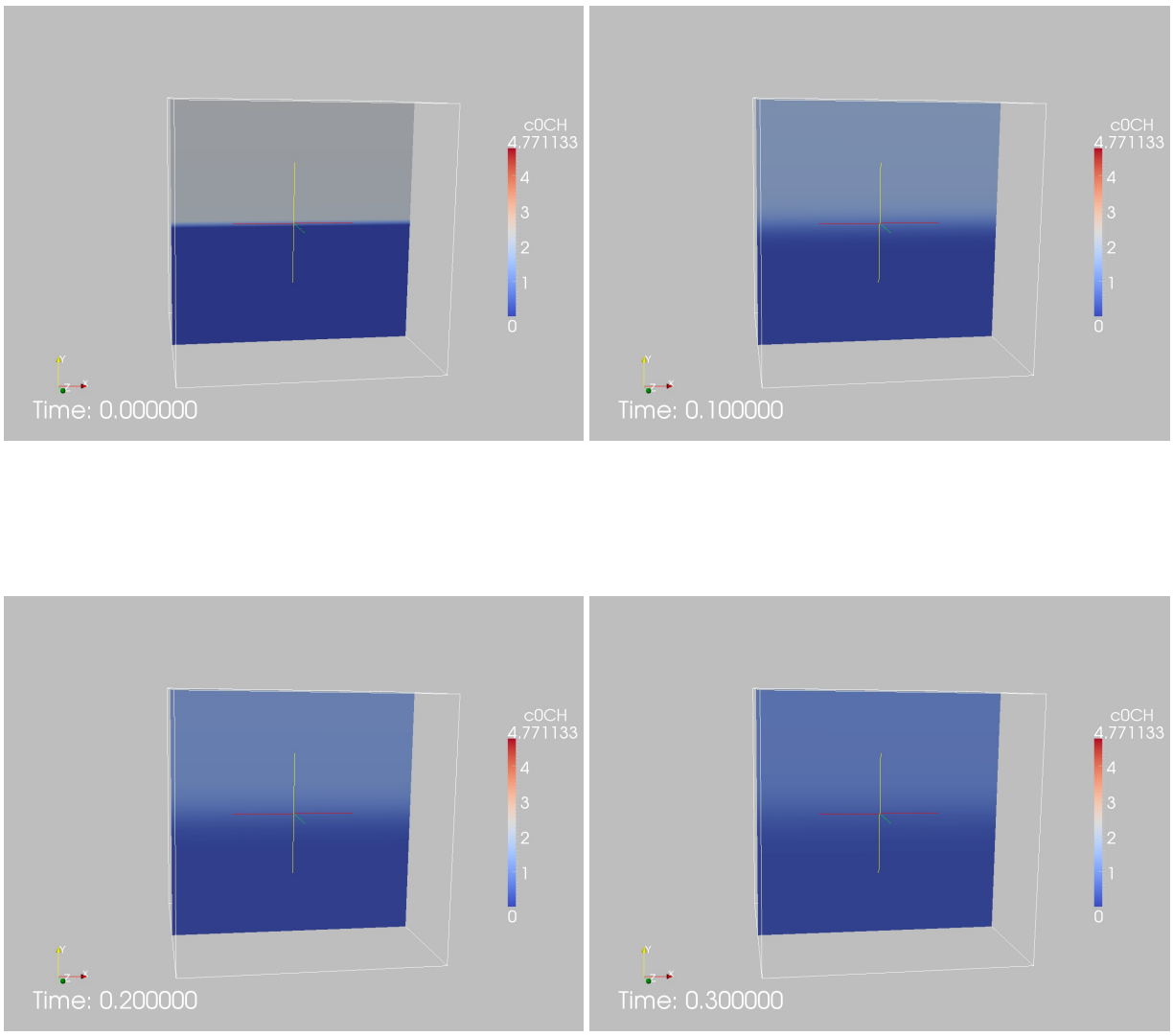
erhalten haben:

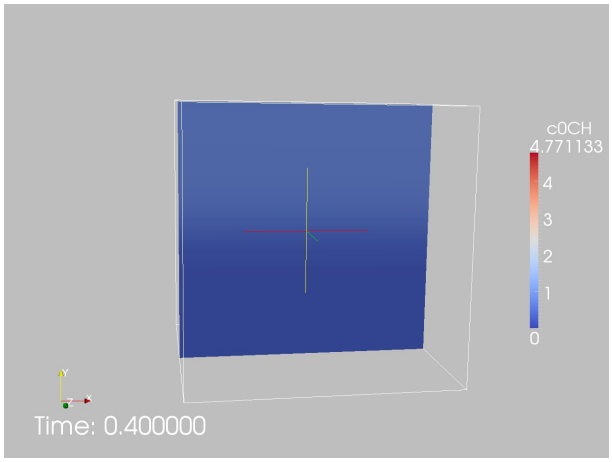


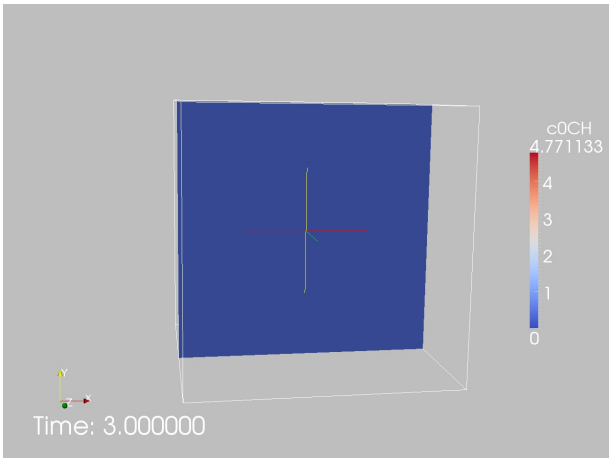

c0CH
4.771138
4
3
2
1
0.301545
Time: 6.700000
c0CH
4.771138
4
3
2
1
0.301545
Time: 6.900000
c0CH
4.771138
4
3
2
1
0.301545
Time: 7.100000
c0CH
4.771138
4
3
2
1
0.301545
Time: 7.700000
c0CH
4.771138
4
3
2
1
0.301545
Time: 8.200000
c0CH
4.771138
4
3
2
1
0.301545
Time: 8.900000

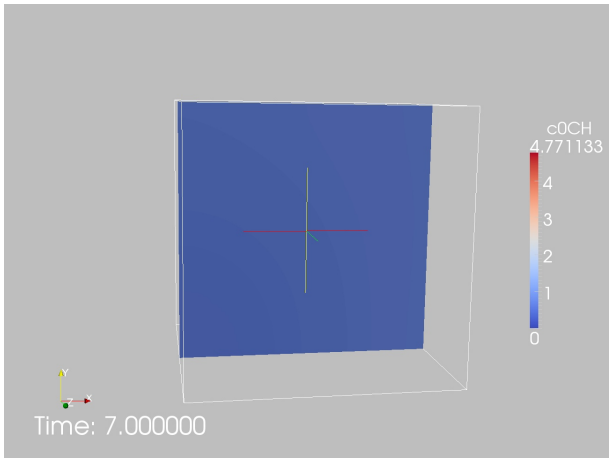
A.3. 3D-Reaktion mit getrennten Chemikalien ohne Rührfisch

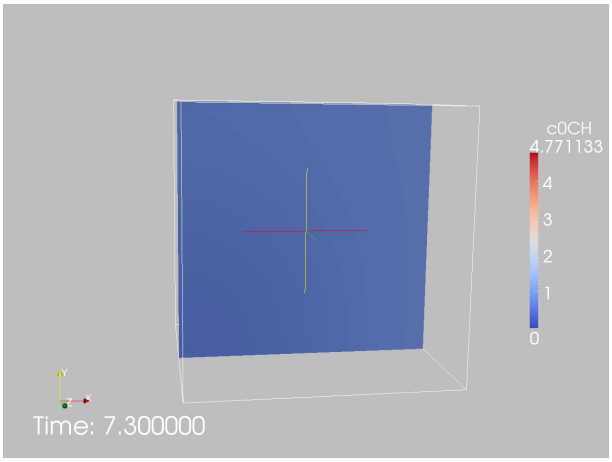
Zum besseren Vergleich haben wir die Simulation mit getrennten Anfangskonzentrationen aus Kapitel 4.4.3 auch ohne den Rührfisch durchgeführt. Wir zeigen dazu hier die Ergebnisse für C_1 (C_2 verhält sich analaog).

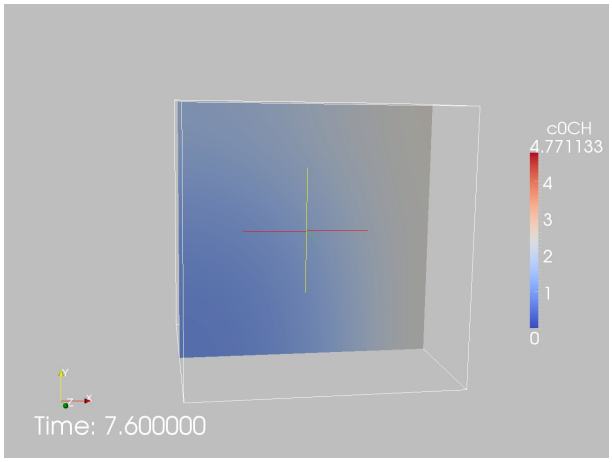


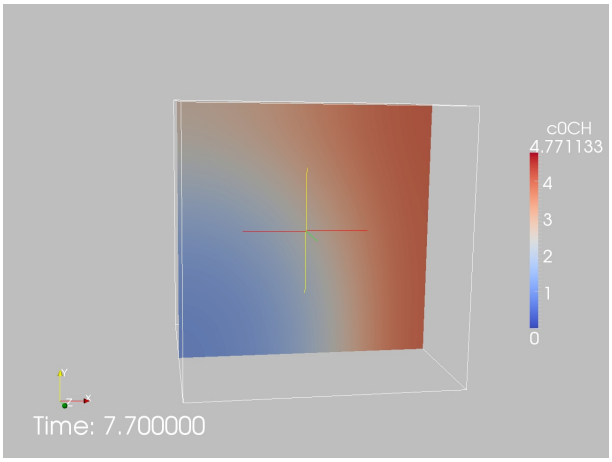

c0CH
4.771133
4
3
2
1
0
Time: 0.400000

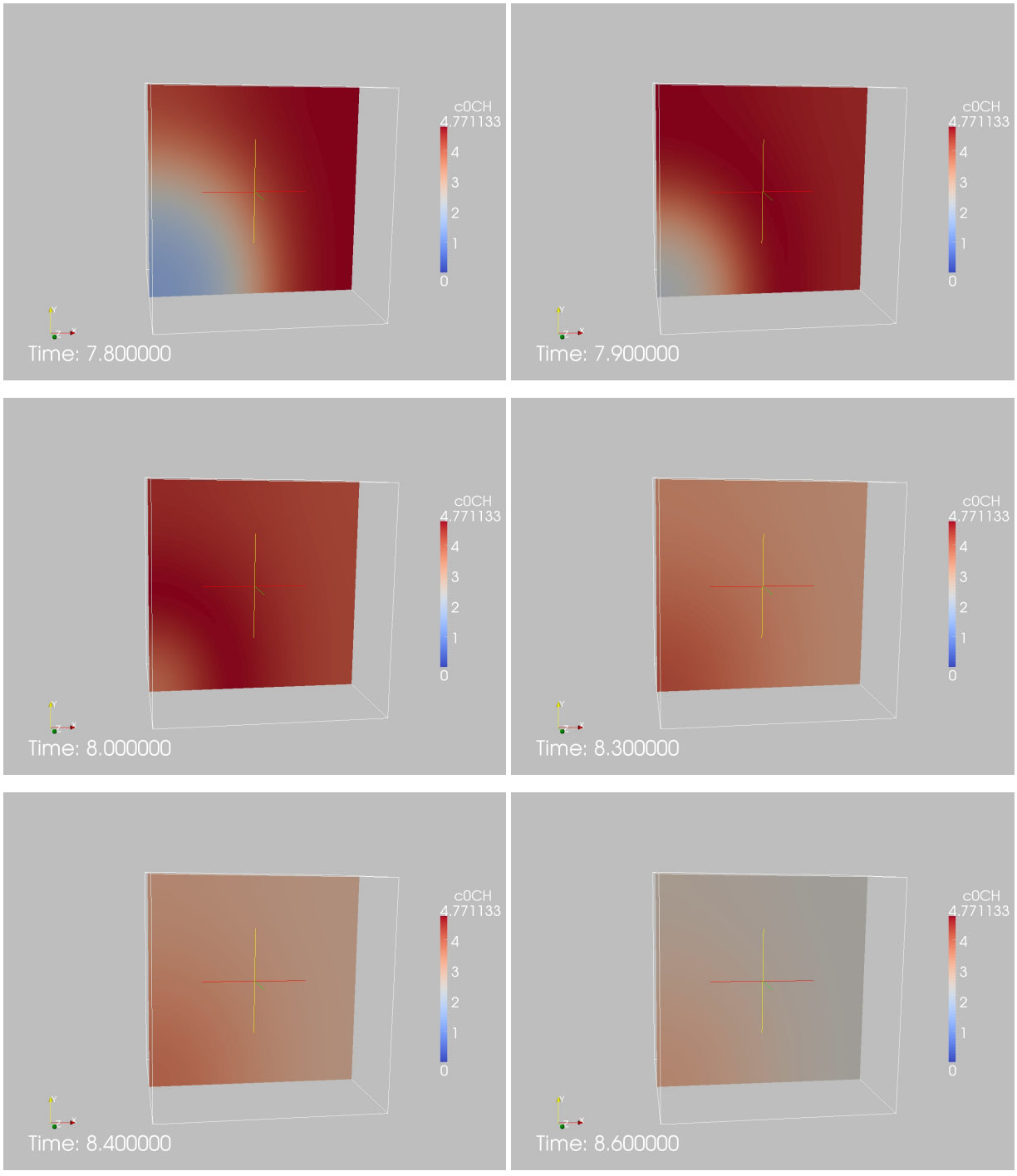

c0CH
4.771133
4
3
2
1
0
Time: 3.000000


c0CH
4.771133
4
3
2
1
0
Time: 7.000000


c0CH
4.771133
4
3
2
1
0
Time: 7.300000

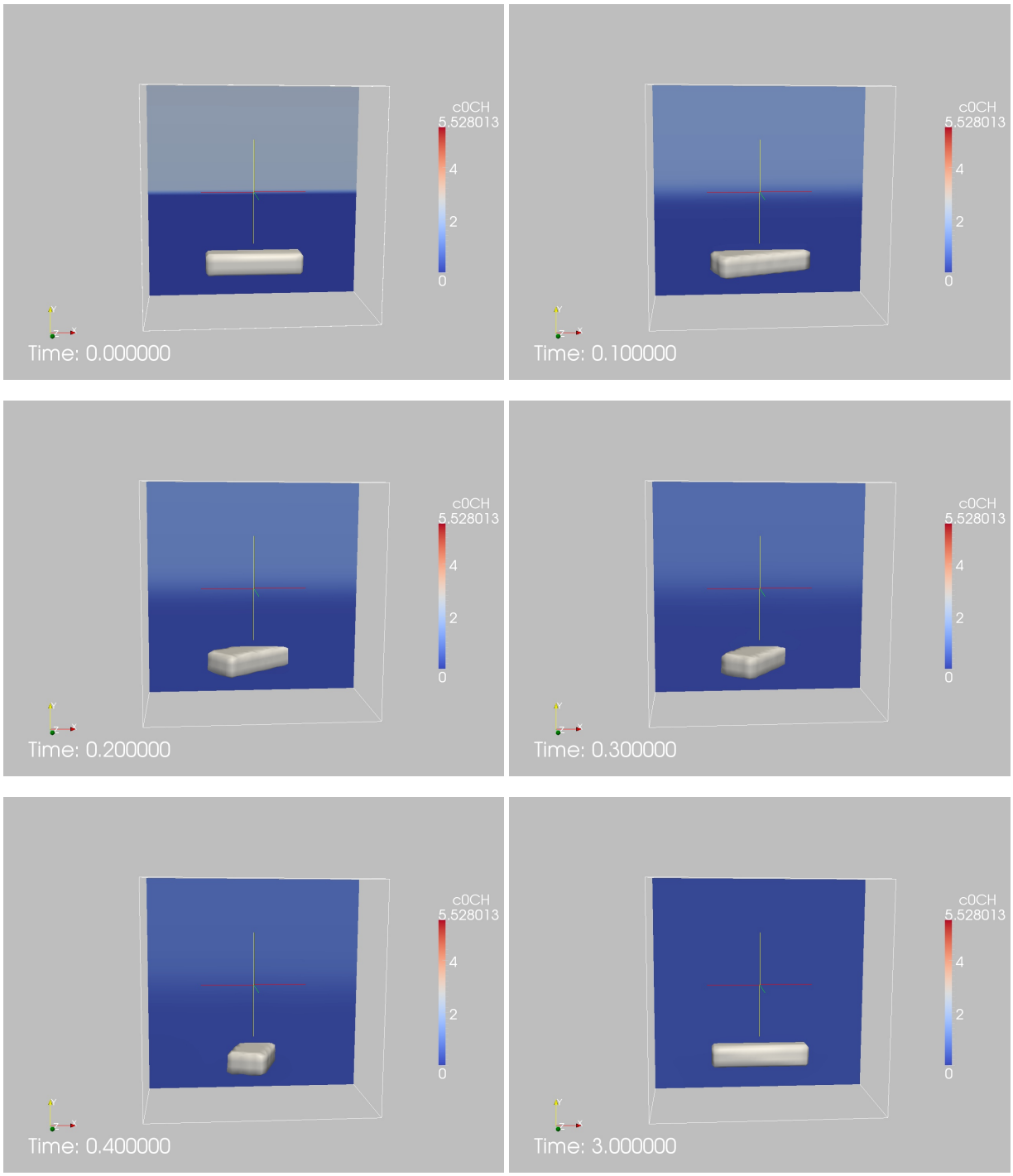

c0CH
4.771133
4
3
2
1
0
Time: 7.600000

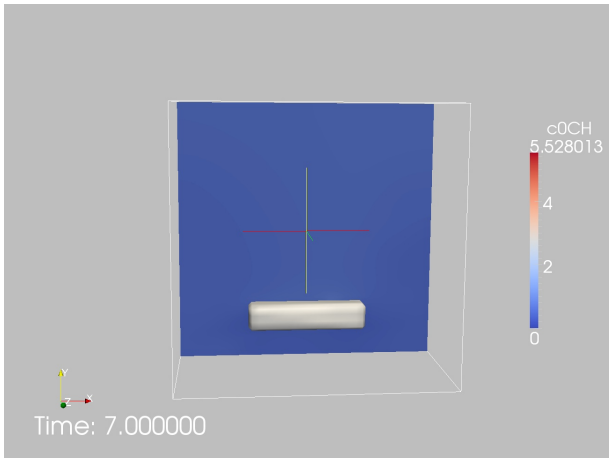

c0CH
4.771133
4
3
2
1
0
Time: 7.700000

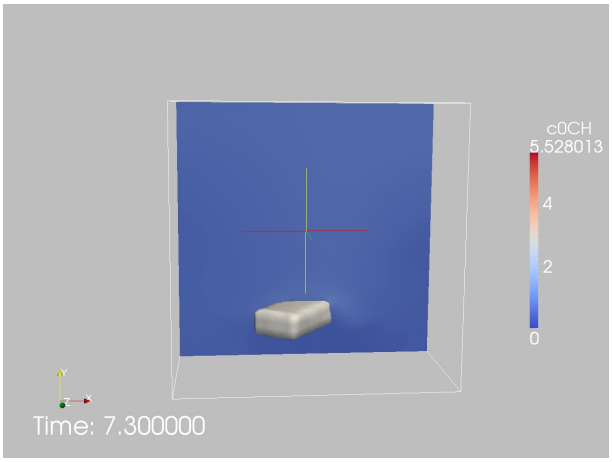

c0CH
4.771133
4
3
2
1
0
Time: 7.800000
Time: 7.900000
Time: 8.000000
Time: 8.300000
Time: 8.400000
Time: 8.600000

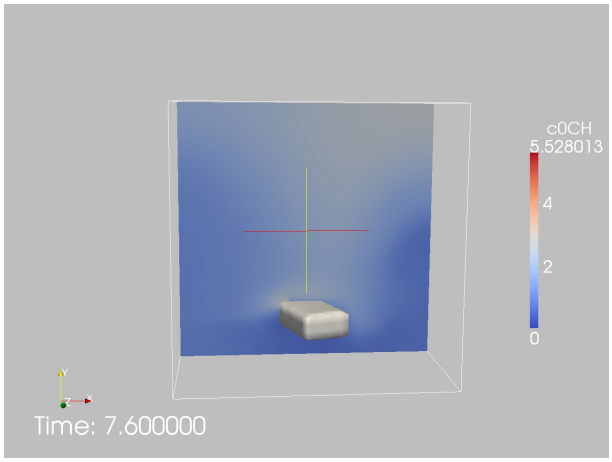
A.4. 3D-Reaktion mit getrennten Chemikalien, Rührfisch nur passiv gekoppelt

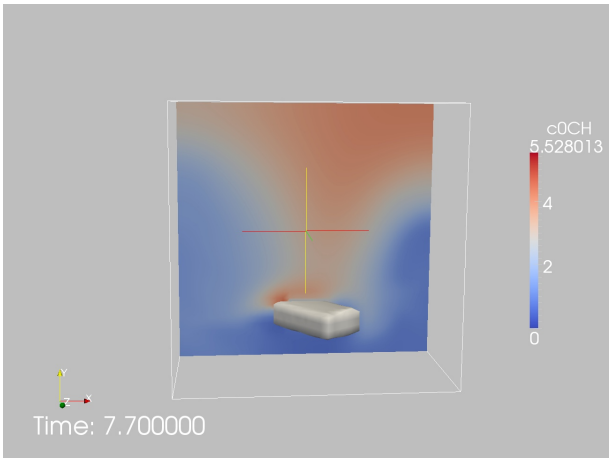
C_1

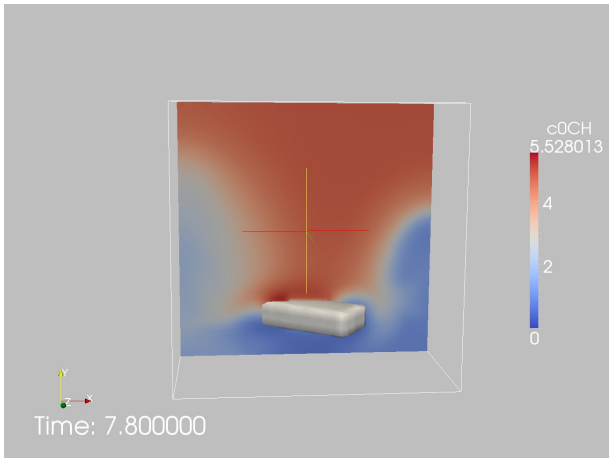


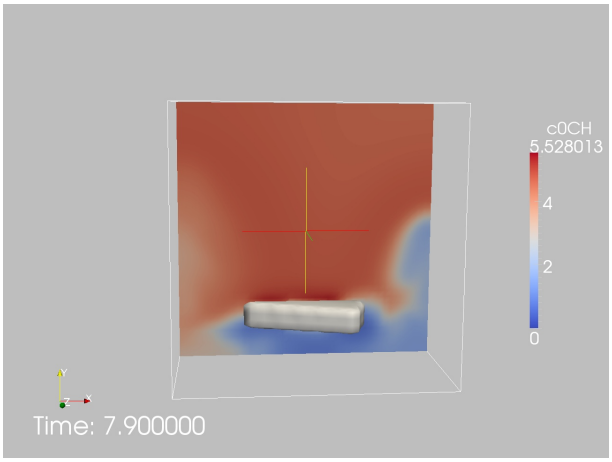

c0CH
5.528013
4
2
0
Time: 7.000000

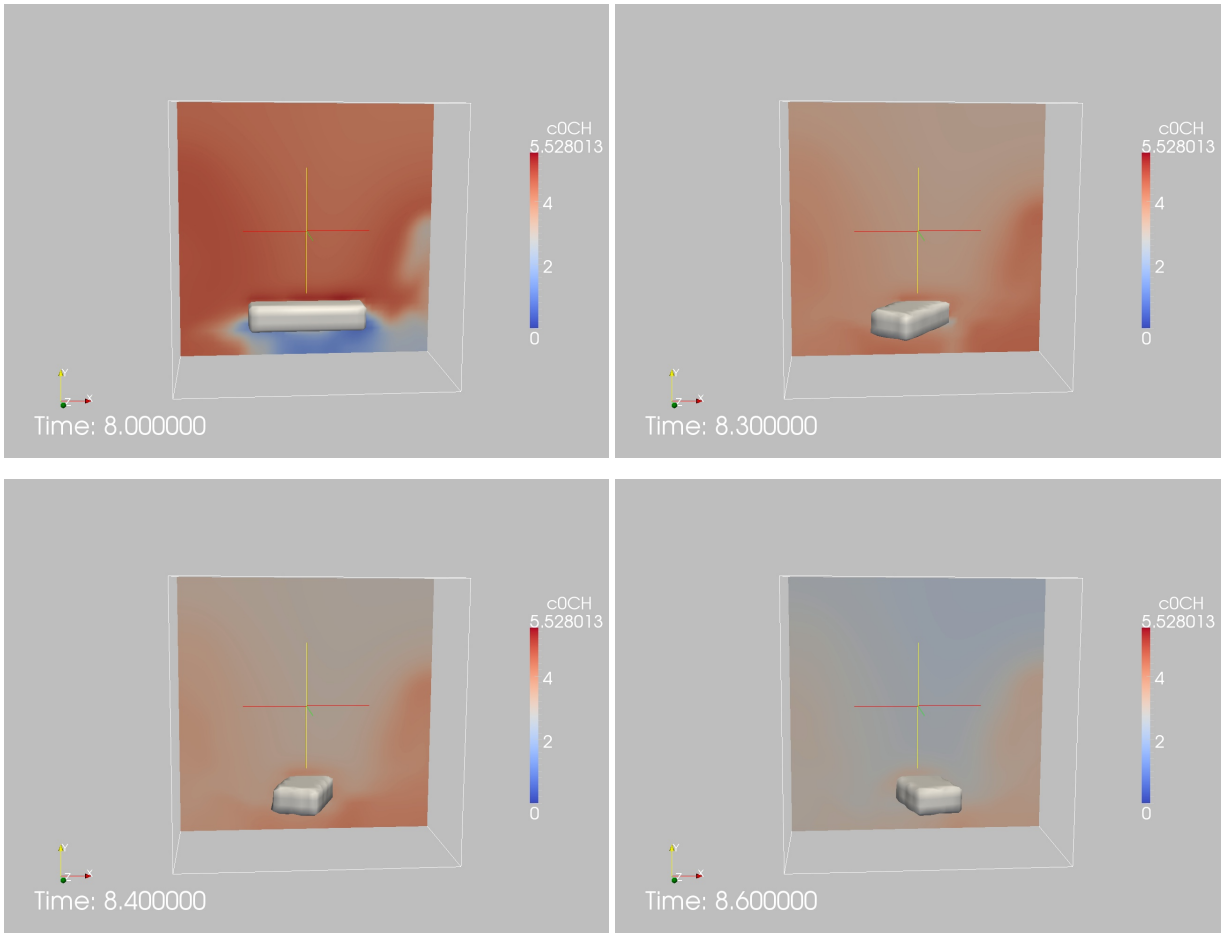

c0CH
5.528013
4
2
0
Time: 7.300000


c0CH
5.528013
4
2
0
Time: 7.600000


c0CH
5.528013
4
2
0
Time: 7.700000


c0CH
5.528013
4
2
0
Time: 7.800000


c0CH
5.528013
4
2
0
Time: 7.900000


c0CH
5.528013
4
2
0
Time: 8.000000
c0CH
5.528013
4
2
0
Time: 8.300000
c0CH
5.528013
4
2
0
Time: 8.400000
c0CH
5.528013
4
2
0
Time: 8.600000

Literaturverzeichnis

[Bur10] Burkow, M.: *Numerische Simulation stroemungsbedingten Sedimenttransports und der entstehenden Gerinnebettformen.* Diplomarbeit, Institut für Numerische Simulation, Universität Bonn, 2010.

[Cho68] Chorin, Alexandre Joel: *Numerical solution of the Navier-Stokes equations.* Mathematics of computation, 22(104):745–762, 1968.

[Con82] Conradi, Elke: *Oszillierende chemische Reaktionen. Experimentalvortrag.* Protokoll nach: `http://www.chids.de/dachs/expvortr/215OszillierendeRkt_Conradi_Scan.pdf`, 1982.

[Cro02] Croce, Roberto: *Ein paralleler, dreidimensionaler Navier-Stokes-Löser für inkompressible Zweiphasenströmungen mit Oberflächenspannung, Hindernissen und dynamischen Kontaktflächen.* 2002.

[Cro10] Croce, Roberto: *Numerische Simulation der Interaktion von inkompressiblen Zweiphasenströmungen mit Starrkörpern in drei Raumdimensionen.* Dissertation, University of Bonn, 2010.

[DB08] Deuflhard, P. und F. Bornemann: *Numerische Mathematik II: Integration gewöhnlicher Differentialgleichungen, 3.Auflage.* De Gruyter Lehrbuch. Walter de Gruyter, 2008, ISBN 9783110203561.

[FKN72] Field, Richard J, Endre Koros und Richard M Noyes: *Oscillations in chemical systems. II. Thorough analysis of temporal oscillation in the bromate-cerium-malonic acid system.* Journal of the American Chemical Society, 94(25):8649–8664, 1972.

[GDN98] Griebel, Michael, Thomas Dornsheifer und Tilman Neunhoeffer: *Numerical simulation in fluid dynamics: a practical introduction*, Band 3. Siam, 1998.

[GK13] Griebel, M. und M. Klitz: *Simulation of Droplet Impact with Dynamic Contact Angle Boundary Conditions.* 2013. INS Preprint No. 1302.

[GTF90] Gyorgyi, Làszlo, Tamàs Turànyi und Richard J Field: *Mechanistic details of the oscillatory Belousov-Zhabotinskii reaction.* Journal of physical chemistry, 94(18):7162–7170, 1990.

[HNW00] Hairer, Ernst, SP Noersett und Gerhard Wanner: *Solving ordinary differential equations*. Springer-Vlg, 2000.

[Kuz] Kuzmin, Dmitri: *Introduction to computational fluid dynamics. skript.* `http://www.mathematik.uni-dortmund.de/~kuzmin/cfdintro/cfd.html`.

[Lab13] Laboratories, Sandia National: *Chemkin reaction design software*. `http://www.sandia.gov/chemkin/index.html`, 2013.

[LN71] Lefever, René und Grégoire Nicolis: *Chemical instabilities and sustained oscillations*. Journal of theoretical Biology, 30(2):267–284, 1971.

[MP92] Maas, Ulrich und Stephen B Pope: *Simplifying chemical kinetics: intrinsic low-dimensional manifolds in composition space*. Combustion and Flame, 88(3):239–264, 1992.

[SMM13] Schaback, Robert, Maryam Mohammadi und Reza Mokhtari: *Simulating the 2d brusselator system in reproducing kernel hilbert space*. `http://num.math.uni-goettingen.de/schaback/research/papers/St2DBSiRKHS.pdf`, 2013.

[Té69] Témam, R.: *Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (II)*. Archive for Rational Mechanics and Analysis, 33:377–385, 1969.

[Zas09] Zaspel, P.: *Zweiphasige Navier-Stokes Fluidsimulationen in Maya: Konfiguration, Visualisierung und Animation*. Diplomarbeit, Institut für Numerische Simulation, Universität Bonn, April 2009.