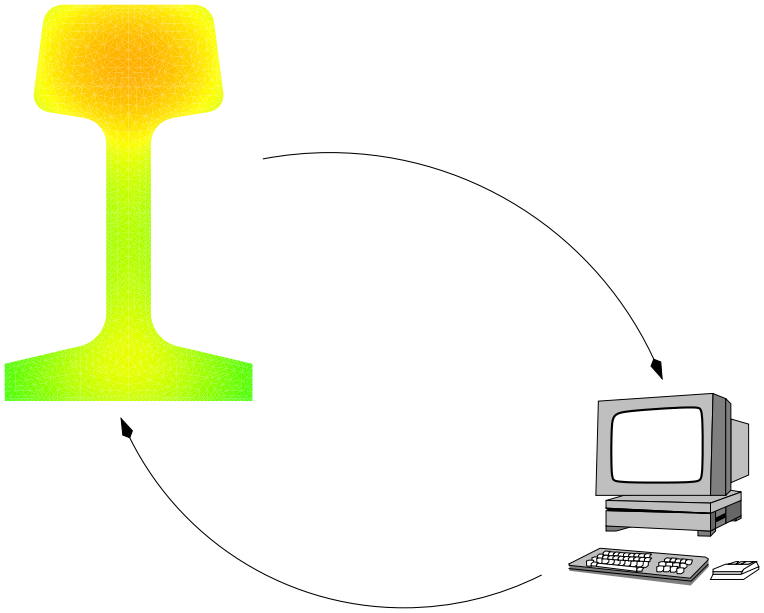
Universität Bremen

Zentrum für Technomathematik

Effiziente numerische Lösung eines Optimalsteuerungsproblems für die Abkühlung von Stahlprofilen

Diplomarbeit

im Studiengang Technomathematik



eingereicht von: Jens Saak

1. Gutachter: Dr. Peter Benner (TU Berlin)

2. Gutachter: Prof. Dr. Alfred Schmidt

Bremen, den 2. September 2003

Inhaltsverzeichnis

Einleitung

Diese Arbeit macht im Gegensatz zur derzeitigen allgemeinen Mediensituation nicht heiß, sondern kalt, denn es geht hier wie der Titel schon unschwer vermuten läßt, um die geregelte Abkühlung von Stahlprofilen. Am Beispiel eines Gleisprofiles soll dazu ein MATLAB–Programm entwickelt werden, welches es erlaubt, diesen Prozess am Computer zu simulieren und verschiedene Parameter und Kostenfunktionen zu testen. Dabei werden zwei Programmpakete eingesetzt, welche die beiden wesentlichen dabei auftretenden Teilaufgaben implementieren. Einerseits wird die C-Bibliothek ALBERT zum Einsatz kommen. Hierbei handelt es sich um eine Sammlung von Funktionen, die ein Entwickler benötigt, um das Wärmeleitungsproblem (und andere Probleme, welche durch partielle Differentialgleichungen beschrieben werden) auf dem Computer in der Programmiersprache C umzusetzen. Andererseits wird das LyaPack benutzt, wobei es sich um eine Sammlung von MATLAB–Routinen handelt, die es ermöglichen, das auftretende Steuerungsproblem mit einer großen Zahl an Unbekannten effizient zu lösen.

Das Beispielproblem, welches hier betrachtet werden soll, orientiert sich an einem Problem, das an der Mathematischen Fakultät der TU Chemnitz in Zusammenarbeit mit Mannesmann Demag [15] bearbeitet wurde. Damit drängt sich natürlich die Frage auf: Warum sollte man dieses Problem bearbeiten, wenn es doch bereits erforscht ist? Die Antwort ist sehr einfach. In Chemnitz wurde das Wärmeleitungsproblem nichtlinear betrachtet. In dieser Arbeit soll das Modell nun dahingehend vereinfacht werden, daß das Problem hier linearisiert wird. Dies erlaubt die Benutzung von Regelungsmechanismen, die auf das nichtlineare Problem nicht angewendet werden können. Die Hoffnung ist dann, daß diese Methoden eine effizientere Lösung des Problems bei vertretbaren Genauigkeitsverlusten ermöglichen.

Die verwendete Regelungsmethode läßt dabei in der Implementierung verschiedene Herangehensweisen zu. Ein Schwerpunkt im fortgeschrittenen Teil dieser Arbeit wird es sein, die verschiedenen Herangehensweisen aufzuzeigen und in Hinsicht auf ihre Effizienz miteinander zu vergleichen. Einen anderen Schwerpunkt stellt das Testen der Implementierung mit verschiedenen Anforderungen an die Optimalität der Lösung, sowie der Vergleich der erlangten Ergebnisse dar.

Zielsetzung

Ziel dieser Arbeit ist es nicht, eine mathematisch penible und detailgenaue Betrachtung des zugrundeliegenden Randkontrollproblems zu erlangen. Es geht vielmehr darum eine Implementierung[1] der numerischen Methoden zu erhalten, die in Hinsicht auf Effizienz der Nutzung von zur Verfügung stehenden Rechenressourcen *optimal* oder wenigstens gut ausgereift ist. Dabei darf natürlich in einer mathematischen Arbeit auch die Betrachtung des mathematischen Hintergrundes nicht zu kurz kommen. Die tieferliegenden Resultate werden jedoch in dieser Arbeit nur zitiert und die Gültigkeit ihrer Voraussetzungen geprüft. Eine Ausführung oder gar ein Beweis dieser Resultate würde bereits allein den Umfang einer Diplomarbeit füllen.

Da die Theorie der Feedbacksteuerung von gewöhnlichen Differentalgleichungssystemen in endlichen Dimensionen eine wesentliche Rolle für die numerische Umsetzung spielt, wird diese in einer Übersicht im Anhang der Arbeit beleuchtet. Um einen Überblick bezüglich der Methoden zu gewinnen, verzichten wir auch dort auf das Ausführen der Beweise, da dies vom eigentlichen Schwerpunkt ablenken würde. Ebenso sind auch andere für die Arbeit wichtige Hilfsresultate in den Anhang verschoben worden, um im Hauptteil der Arbeit einen besseren Blick auf die wesentlichen Zusammenhänge zu gewährleisten.

Übersicht

Diese Diplomarbeit gliedert sich im Hauptteil in drei Bereiche:

- Modellentwicklung,
- Theoretischer Hintergrund und Implementierung,
- Numerische Tests.

Einen vierten Bereich bilden die im Anhang aufgeführten Hilfsresultate, welche an entsprechenden Stellen im Hauptteil zitiert werden. Die Modellentwicklung beschränkt sich auf das Kapitel 1. Dort wird das im weiteren Verlauf der Arbeit verwendete Modell aus dem oben angesprochenen Chemnitzer Modell abgeleitet. Aktuelle Erkentnissen aus dem Sonderforschungsbereich 570: “Distortion Engineering - Verzugsbeherrschung in der Fertigung”, an dem auch das Zentrum für Technomathematik der Universität Bremen mit zwei Arbeitsgruppen beteiligt ist, ergänzen das in dieser Arbeit verwendete Modell.

Der Bereich “Theoretischer Hintergrund und Implementierung” erstreckt sich über die Kapitel 2 und 3. Dabei geht es im Kapitel 2 darum, ob und wie man das Kontrollproblem aus Kapitel 1, das in einem unendlich dimensionalen Hilbertraum formuliert

[1] Angesichts der verschiedenen betrachteten Herangehensweisen und Randwerte, die im Modellkapitel sowie im Implementationskapitel erläutert werden sogar mehrere Implementierungen

ist, durch Probleme in endlich dimensionalen Teilräumen dieses Hilbertraumes approximieren kann. Damit ist dann der Grundstock für die Implementierung gelegt, welche das Kapitel 3 behandelt. Dazu werden zunächst einige theoretische Hilfmittel eingeführt. Der zweite Abschnitt des Kapitels widmet sich dann den Resultaten zur Approximation von abstrakten linearen Kontrollproblemen. Die dort aus [2] zitierten Ergebnisse gehen auf Resultate zurück, die Gibson Ende der siebziger Jahre in [12] vorgestellt hat. Die Neuerung, welche die von Banks und Kunisch präsentierten Resultate gegenüber der Theorie von Gibson aufweisen ist, daß sie nicht mehr darauf angewiesen sind, die approximierenden Probleme in dem zu approximierenden Hilbertraum selbst zu formulieren. Sie betrachten die approximierenden Probleme in unabhängigen endlich dimensionalen Räumen, die nicht einmal zwingend Teilräume des Hilbertraumes sein müssen. Die vorliegende Arbeit beschränkt sich allerdings auf die Approximation des Hilbertraumes durch endlich dimensionale Teilräume seinerselbst. Im vierten Abschnitt des Kapitels wird dann die Konvergenz der Approximation des Randkontrollproblemes dieser Arbeit durch Methoden entsprechend denen von Banks und Kunisch auf die Konvergenz der Approximation des beschriebenen abstrakten Kontrollproblems zurückgeführt. Außerdem wird im dritten Abschnitt des Kapitels die schwache Formulierung des betrachteten Problems hergeleitet, die insbesondere bei der Implementierung in Kapitel 3 eine zentrale Rolle spielt, aber auch für die Definition der approximierenden Probleme eine wichtige Voraussetzung darstellt.

Im Kapitel 3 geht es dann um die softwaretechnische Umsetzung des Problems. Dazu werden die beiden verwendeten Programmpakete kurz vorgestellt und ihr Funktionsumfang umrissen. Danach wird dann jeweils aufgezeigt, wie sich das Paket in die Aufgabenstellung einfügt und wo es zum Einsatz kommt. Der erste Abschnitt des Kapitels erläutert zuvor aber die beiden Herangehensweisen, die bei der Implementierung verfolgt wurden, und zeigt ihre Unterschiede auf. Dabei wird insbesondere auch darauf eingegangen, welche Teilprobleme es bei der Implementierung zu verknüpfen galt.

Das Ergebniskapitel 4 geht insbesondere auch auf den direkten Vergleich der verschiedenen Herangehensweisen ein. Diese werden aber auch ins Verhältnis zu ungesteuerten Referenzrechnungen gesetzt, bei denen das Abkühlen an der Luft simuliert wird.

Der Bereich “Numerische Tests” widmet sich im Kapitel 5 den Auswirkungen verschiedener Kostenfunktionale. Dabei werden unterschiedliche Gewichtungen der Referenzpunkte in den Ausgängen des Systems, sowie verschiedene Schwerpunkte im Bezug auf die Kosten der Steuerung und hoher Temperaturgradienten untersucht.

Danksagungen

Mein Dank gilt all denen, die mich bei der Anfertigung dieser Arbeit sowohl tatkräftig, als auch seelisch und moralisch unterstützt haben. Besonders hervorheben möchte ich an dieser Stelle zunächst meinen Diplombetreuer Peter Benner, dem ich speziell für seine große Geduld bei der Begleitung meiner Arbeit danke. Desweiteren gilt mein besonderer Dank Rolf Krahl. Ohne dessen Unterstützung bei der Einarbeitung in ALBERT und die vielen fruchtbaren Diskussionen hätte die Programmierphase wohl noch um einiges länger gedauert. Insbesondere die Bereitstellung seiner ALBERT–Toolbox hat

mir einiges an Zeit verschafft. Bei Alfred Schmidt bedanke ich mich für die Hilfestellungen bei der Arbeit mit ALBERT und insbesondere die vielen Tipps zur Visualisierung der Daten mit GRAPE. Nicht vergessen möchte ich an dieser Stelle Kristian Bredies und Malte Peter, die mir gerade in der Endphase der Arbeit immer wieder ein offenes Ohr geschenkt haben und mit wichtigen Ratschlägen aufwarten konnten.

Unter den Mitarbeitern des ZeTeM möchte ich besonders Ronald Stöver danken, der über mein gesamtes Studium immer ein freundlicher und kompetenter Ansprechpartner bei Studienproblemen aller Art war, sowie bei Elmar "Postscript" Plischke, ohne dessen fundierte Postscriptkentnisse so manche Grafik in dieser Arbeit nicht in der vorliegenden Form möglich gewesen wäre.

Nicht zuletzt möchte ich mich bei meiner Familie und meinen Freunden bedanken, die mich besonders in den Prüfungszeiten ertragen und immer wieder zum Weitermachen ermutigt haben. Ein besonders großes Dankeschön geht hier an meine Freundin Şenay Yavuz, die mich aus mancher Sinnkrise gerissen und immer wieder von der Richtigkeit meines Tuns überzeugt hat.

Bezeichnungen

$\mathbb{N}$	ist die Menge der natürlichen Zahlen beginnend bei 1,
$\mathbb{N}_0$	$= \mathbb{N} \cup \{0\}$,
$\mathbb{R}$	bezeichnet den Körper der reellen Zahlen,
$\mathbb{C}$	bezeichnet den Körper der komplexen Zahlen,
$\mathbb{R}_{\geq 0}$	bezeichnet die Halbachse der nichtnegativen reellen Zahlen,
$\mathbb{C}_{\geq 0}$	bezeichnet die Halbebene der komplexen Zahlen mit nichtnegativem Realteil,
$\mathbb{R}_{>0}$	bezeichnet die Halbachse der echt positiven reellen Zahlen,
$\mathbb{C}_{>0}$	bezeichnet die Halbebene der komplexen Zahlen mit echt positivem Realteil,
$\mathbb{R}^n := \bigtimes_{i=1}^{n} \mathbb{R}$	
$\mathbb{C}^n := \bigtimes_{i=1}^{n} \mathbb{C}$	sind die n–dimensionalen Vektorräume über $\mathbb{R}$ bzw. $\mathbb{C}$
$\mathbb{R}^{m,n}$	bezeichnet den Vektorraum der $m \times n$ Matrizen über dem Körper $\mathbb{R}$

Sind X, Y normierte $\mathbb{K}$–Vektorräume, dann bezeichnen:[2]

$\mathcal{L}(X, Y)$	die Menge der *linearen* und *stetigen* Operatoren auf X mit Bild in Y,
$\mathcal{L}(X)$	$:= \mathcal{L}(X, X)$ mit *Identität* $I \in \mathcal{L}(X)$,
$\mathcal{K}(X, Y)$	die Menge der *kompakten* Operatoren aus $\mathcal{L}(X, Y)$,
$\mathcal{K}(X)$	$:= \mathcal{K}(X, X)$,
$X' := \mathcal{L}(X, \mathbb{K})$	den *Dualraum* von X, seine Elemente heißen *lineare Funktionale* auf X.

Ist $v \in X$ und $f \in X'$, so meint $< f, v >$ die Dualpaarung, also die Anwendung des Funktionals f auf v. Zur deutlichen Unterscheidung werden Skalarprodukte stets in der Form $(\cdot, \cdot)$ mit runden Klammern geschrieben.

[2] siehe dazu auch [1] Kapitel 3

Das Kronecker–Symbol δ_{ij} ist für $i, j \in \mathbb{N}$ definiert durch

$$\delta_{ij} = \begin{cases} 1 & \text{falls } i = j \\ 0 & \text{sonst} \end{cases}$$

Ist $x = (x_1, \ldots, x_n) \in \mathbb{R}^n = \mathbb{R}^{1,n}$, dann bezeichnet

$$x^T = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^{n,1}$$

den transponierten Vektor zu x.

Sei $A \in \mathbb{R}^{n,n}$, dann bezeichnet $\sigma(A)$ das Spektrum von A, d.h.:

$$\sigma(A) := \{\lambda \in \mathbb{C} | \exists x \in \mathbb{R}^n \text{ mit } Ax = \lambda x\}.$$

A heißt *positiv definit*, falls stets gilt $x^T Ax \geq 0$ und $x^T Ax = 0 \Leftrightarrow x = 0$. Wir schreiben dann auch kurz $A > 0$. A^T bezeichnet die Transponierte zu A und A heißt *symmetrisch*, falls gilt $A = A^T$. Den Rang (definiert als die Dimension des Bildes der durch A gegebenen linearen Abbildung) der Matrix A bezeichnen wir mit $\mathrm{rg}(A)$. Die Einheitsmatrix $(\delta_{ij})_{i,j=1,\ldots,n}$ schreiben wir kurz als I_n, oder einfach I sofern die Dimension aus dem Zusammenhang klar ist.

Sei $M \subset \mathbb{R}^n$, dann bezeichnet

$\mathring{M}$	das Innere von M, sowie
$\bar{M}$	den Abschluß von M.

Die Menge M heißt *zusammenhängend* genau dann, wenn sie *wegzusammenhängend* ist, d.h. wenn es zu je zwei Punkten $x_0, x_1 \in M$ eine stetige Abbildung $\gamma : [0,1] \to M$ mit $\gamma(0) = x_0$ und $\gamma(1) = x_1$ gibt. γ heißt dann *(stetiger) Weg* von x_0 nach x_1 in M. Falls nichts anderes gesagt wird bezeichnet Ω ein Gebiet im R^n. Das heißt $\Omega \subset \mathbb{R}^n$ ist offen und zusammenhängend und sein Rand $\Gamma = \partial\Omega$ ist Lipschitzstetig. Dabei gilt im Fall des Anwendungsproblems, welches in dieser Arbeit diskutiert wird, stets $n = 2$.

Ableitungen

Für den Rest dieses Kapitels bezeichnen k, m und p natürliche Zahlen. Sei $f : T \times \Omega \to \mathbb{R}$ eine von der Zeit $t \in T := (t_0, t_{end}) \subset \mathbb{R}_{\geq 0}$ und dem Ort $x = (x_1, \ldots, x_n)^T \in \Omega$ abhängige Funktion. Außerdem sei $\nu \in \mathbb{R}^n$ ein Vektor (gewöhnlich die äußere Normale an den Rand von Ω). $\alpha := (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}_0^n$ heißt *Multi–Index* mit Betrag $|\alpha| := |\alpha_1| + \cdots + |\alpha_n|$.

$\partial_t f := \frac{\partial}{\partial t} f$	heißt die Zeitableitung von f. Wir werden häufig auch einfach $\dot{f} = \frac{\partial}{\partial t} f$ verwenden.
$\partial_j f := \frac{\partial}{\partial x_j} f$	heißt die j–te partielle Ableitung oder j–te Ortsableitung von f.
$\partial_j^k f := \underbrace{\partial_j \dots \partial_j f}_{k\text{–mal}}$	heißt die k–fache j–te partielle Ableitung,
$\partial^\alpha f := \partial_1^{\alpha_1} \dots \partial_n^{\alpha_n} f$	ist die α–te partielle Ableitung von f,
$\nabla f := (\partial_1 f, \dots, \partial_n f)^T$	heißt der Gradient von f.
$\partial_\nu f := \nu \cdot \nabla f$	ist die Ableitung von f in Richtung ν. Im Fall der äußeren Normalen nennen wir diese auch einfach die Normalenableitung von f.
$\Delta f := \sum_{i=1}^{n} \partial_i^2 f$	bezeichnet den *Laplaceoperator* angewandt auf f.

Dabei ist hier stets die schwache Ableitung gemeint, wie sie z.B. in [1] eingeführt wird, sofern keine (klassische) Ableitung existiert. Anderenfalls stimmen beide überein.

Funktionenräume und ihre Normen und Skalarprodukte

Die Räume der stetigen, bzw. k–mal stetig differenzierbaren Funktionen auf Ω werden wie üblich mit $C(\Omega)$ und $C^k(\Omega)$ bezeichnet. Entsprechend sind $C^\infty(\Omega)$ und $C_0^\infty(\Omega)$ die Räume der unendlich oft stetig differenzierbaren Funktionen, bzw. unendlich oft stetig differenzierbaren Funktionen mit kompaktem Träger.

Lebesgue–Räume nennen wir für $1 \le p \le \infty$ die Räume:

$$L^p(\Omega) = \{f : \Omega \to \mathbb{R} \ : \ \|f\|_p < \infty\}$$

mit der Norm

$$\|f\|_p := \left(\int_\Omega |f(x)|^p dx \right)^{\frac{1}{p}} \quad \text{für } 1 \le p < \infty$$

bzw.

$$\|f\|_\infty := \operatorname*{ess\,sup}_{x \in \Omega} := \inf_{\substack{\mathcal{N} \subset \Omega \\ \text{Nullmenge}}} \sup_{x \in \Omega \setminus \mathcal{N}} |f(x)| < \infty.$$

Mit der Äquivalenzrelation $f = g$ in $L^p(\Omega)$, falls gilt $f = g$ fast überall in Ω, erhalten wir jeweils einen Banachraum. Im Fall $p = 2$ handelt es sich mit dem Skalarprodukt

$$(f, g)_2 := \left(\int_\Omega f(x) g(x) dx \right)^{\frac{1}{2}},$$

sogar um einen Hilbertraum. Sollen Funktionen mit Bild in $\mathbb{C}$ anstatt in $\mathbb{R}$ betrachtet werden, dann muß im Integralterm anstelle von $g(x)$ die konjugiert komplexe Funktion $\overline{g(x)}$ stehen.

Damit definieren wir weiter die Sobolev–Räume $H^{m,p}(\Omega)$ durch

$$H^{m,p}(\Omega) = \{u \in L^p(\Omega) : |\alpha| \le m, \ \partial^\alpha u \in L^p(\Omega)\}.$$

Die Norm

$$\|u\|_{m,p} := \left(\sum_{|\alpha| \leq m} \int_\Omega |\partial^\alpha u(x)|^p dx \right)^{\frac{1}{p}}$$

macht $H^{m,p}(\Omega)$ zum Banachraum. Im Fall $p = 2$ ist $H^{m,2}(\Omega)$ mit dem Skalarprodukt

$$(u, v)_{m,2} := \left(\sum_{|\alpha| \leq m} \int_\Omega \partial^\alpha u(x) \partial^\alpha v(x) dx \right)^{\frac{1}{2}}$$

ein Hilbertraum. Im Falle komplexwertiger Funktionen muß hier wieder wie oben die zweite Komponente konjugiert werden.

Außerdem werden wir Sobolev–Räume mit erweiterten Nullrandwerten betrachten. Diese sind definiert als

$\mathring{H}^{m,p}(\Omega) := \{u \in H^{m,p}(\Omega) : \text{ Es existiert eine Folge } f_k \in C_0^\infty(\Omega) \text{ mit } \|u - f_k\|_m \to 0 \text{ für } k \to \infty\}$.

Da wir es in dieser Arbeit nur mit Sobolev–Räumen über $L^2(\Omega)$ zu tun haben, führen wir außerdem die vereinfachenden Schreibweisen $H^m(\Omega) := H^{m,2}(\Omega)$ und $\mathring{H}^m(\Omega) := \mathring{H}^{m,2}(\Omega)$ ein. Wir schreiben bei den Normen dennoch weiterhin $\|.\|_{m,2}$, um die Unterscheidung zu den Lebesgue–Raum–Normen zu gewährleisten.

Bisher haben wir stets Integrale im Lebesgue–Sinne betrachtet. Für die Behandlung von zeitabhängigen Problemen, benötigen wir außerdem einen vektorwertigen Integralbegriff. Diesen liefert uns das *Bochner–Integral*.

Sei $[a, b] \subset \mathbb{R}$ und $u : [a, b] \to X$ mit einem reellen Banachraum $(X, \|\cdot\|)$. u heißt *einfach*, falls $[a, b]$ in endlich viele paarweise disjunkte, Lebesgue–meßbare Teilmengen B_i zerfällt, derart daß $u(t) = \sum_{i=1}^{n} \alpha_i \chi_{B_i}(t)$ gilt für alle $t \in [a, b]$ und $\alpha_1, \ldots, \alpha_n \in X$. Das Bochner–Integral von u ist dann definiert als

$$\int_a^b u(t)dt = \sum_{i=1}^{n} \text{vol}(B_i)\alpha_i \quad \in X.$$

$\text{vol}(B_i)$ ist dabei ganz das Volumen der Menge B_i im Lebesque–Sinne, also $\int_{B_i} 1$. u heißt *Bochner–meßbar*, falls es punktweise fast überall Grenzwert einer Folge einfacher Funktionen u_k ist. Gilt außerdem

$$\lim_{k \to \infty} \int_a^b \|u(t) - u_k(t)\| dt = 0,$$

so heißt u *Bochner–integrierbar* und

$$\int_a^b u(t)dt := \lim_{k \to \infty} \int_a^b u_k(t)dt$$

heißt das Bochner–Integral von u.[3]

[3] Ausführlichere Einführungen des Bochner Integrals finden sich z.B. in [9, 13, 29]

Analog zu den Lebesgue-Räumen können wir nun bezüglich dieses Integrals die Räume

$$L^p(a,b;X) := \{u : [a,b] \to X : u \text{ Bochner–messbar}, \|u\|_{L^p(a,b;X)} < \infty\}$$

mit den Normen

$$\|u\|_{L^p(a,b;X)} := \left(\int_a^b \|u(t)\|_X^p dt \right)^{\frac{1}{p}}$$

bzw.

$$\|u\|_{L^\infty(a,b;X)} := \inf_{\substack{\mathcal{N}\subset[a,b] \\ \mathcal{N} \text{ Nullmenge}}} \sup_{t\in[a,b]\setminus\mathcal{N}} \|u(t)\|_X,$$

sowie den Raum

$$C^0([a,b];X) := \{u : [a,b] \to X : u \text{ stetig}\}$$

mit der Norm

$$\|u\|_{C^0([a,b];X)} = \max_{t\in[a,b]} \|u(t)\|_X$$

definieren. Gilt $-\infty < a < b < \infty$ und ist X ein Banachraum, so sind auch die so definierten Räume Banachräume.

Ist $(X, (.,.))$ ein Hilbertraum und $0 < T < \infty$ sowie $u \in L^1(0,T;X)$, so definieren wir die *schwache Ableitung* $v \in L^1(0,T;X)$ von u durch:

$$\int_0^T \varphi'(t)u(t)dt = -\int_0^T \varphi(t)v(t)dt \qquad \text{für alle } \varphi \in C_0^\infty(0,T)$$

und schreiben wieder kurz $u' = v$. Die schwache Ableitung ist eindeutig bestimmt, wenn sie existiert.

Damit sind wir nun in der Lage, den Raum zu definieren, in dem wir die Lösung des Wärmeleitungsproblems suchen wollen. Sei wie oben $\Omega \subset \mathbb{R}^d$ offen und beschränkt, $0 < T < \infty$ ($d = 2$ im Fall des vorliegenden Anwendungsproblems.). Wir definieren

$$W_2^1(0,T) := \{u \in L^2(0,T;\mathring{H}^1(\Omega)) : u' \in L^2(0,T;\mathring{H}^1(\Omega))\}$$

mit der Norm

$$\|u\|_{W_2^1(0,T)} = \left(\int_0^T \|u(t)\|^2_{\mathring{H}^1(\Omega)} + \int_0^T \|u'(t)\|^2_{H^1(\Omega)'} \right)^{\frac{1}{2}}.$$

Eigenschaften von linearen Operatoren

Seien $A, B : H \to H$ Operatoren auf dem Hilbertraum H. $A^* : H \to H$ bezeichnet die Hilbertraumadjungierte zum Operator A, definiert durch

$$(x, Ay)_H = (A^*x, y)_H \text{ mit } x,\ y \in H.$$

Wenn gilt $A^* = A$, dann nennen wir A *selbstadjungiert*.

Ein Operator A heißt *nichtnegativ* oder *positiv semidefinit*, wenn stets gilt $(x, Ax)_H \geq 0$. Wir schreiben dafür auch $A \geq 0$. Gilt außerdem $(x, Ax)_H = 0 \Leftrightarrow x = 0$ so nennen wir A *positiv definit* und schreiben $A > 0$.

Die gewöhnliche Ordnung von nichtnegativen selbstadjungierten Operatoren in Hilberträumen wird definiert durch $A > B :\Leftrightarrow A - B > 0$.

Sei X ein Banachraum, sowie X' sein Dualraum. Zu $x \in X$ definieren wir die *Dualitätsmenge* $F(x) \subset X'$ durch

$$F(x) = \{x' \in X' \ : < x', x >= \|x\|^2 = \|x'\|^2\}.$$

Der lineare Operator A heißt *dissipativ* , falls es zu jedem $x \in \operatorname{dom} A \subset X$ ein Element x' aus der Dualitätsmenge $F(x)$ gibt, derart das

$$\operatorname{Re} < x', Ax > \leq 0$$

Dazu wird beispielsweise in [28] gezeigt, daß ein linearer Operator A genau dann dissipativ ist, wenn $\|(\lambda I - A)x\| \geq \lambda \|x\|$ für alle $x \in \operatorname{dom} A$ und $\lambda > 0$. Engel und Nagel verwenden diesen Zusammenhang in [10] direkt zur Definition dieser Eigenschaft.

Kapitel 1

Das Modell

In diesem Kapitel werden wir das verwendete Modell aus einem im Jahre 1999 von Tröltzsch und Unger vorgestellten Modell ableiten. In den ersten Abschnitten soll jedoch zunächst das technische Umfeld aufgezeigt und damit das Modell motiviert werden.

1.1 Zusammenhang von Temperaturentwicklung und Produkteigenschaften

Eine etwas ausführlichere Behandlung, sowie umfangreiche Literaturhinweise zum Zusammenhang von Temperaturentwicklung und Produkteigenschaften finden sich in [15]. An dieser Stelle soll nur stichpunktartig auf einige Details eingegangen werden, um zu vermitteln, welche Fragestellungen bei der Abkühlung von Stahlprofilen während des Walzens bzw. zwischen den einzelnen Walzvorgängen eine Rolle spielen. Die zunächst wichtigsten Fragen bei, bzw. vor der aktiven Kühlung der Profile ist: Welche Temperaturverteilung liegt im Profil vor und wie ist sie entstanden? Dabei hängt die Temperaturverteilung wesentlich davon ab, welche Geometrie der Formstahl vor, während und nach dem Walzvorgang hat. Eine gewisse Vorstellung machen wir uns davon im täglichen Leben, wenn wir ein metallenes Messer aus heißem Wasser heben. Der massive Griff kühlt hier deutlich langsamer ab als die dünnere Klinge. Natürlich hängt die Temperaturverteilung ebenso davon ab, welche Teile des Profils während des Walzvorgangs Kontakt mit den Walzen hatten und wo sich der Stahl durch Umformwärme aufgeheizt hat. Nicht zuletzt spielen auch die Pausen zwischen den einzelnen Produktionsschritten im Kühlbett eine entscheidende Rolle. Diese sollen in erster Linie eine gleichmäßigere Verteilung der Temperatur über den Querschnitt bewirken und für ein deutliches Abkühlen des Profils vor dem nächsten Walzvorgang sorgen. Dabei ist Ersteres je nach Profilgeometrie nur sehr bedingt möglich, da hier Wärmeabstrahlung und Konvektion eine wesentliche Rolle spielen, die bei ungünstiger Geometrie den Ausgleichsprozess eher behindern, als unterstützen.

Eine gleichmäßige Temperaturverteilung über den Profilquerschnitt durch selektive Kühlung vor den letzten Walzvorgängen bewirkt gegenüber konventioneller Walzung ohne selektive Kühlung gleichmäßigere mechanische Eigenschaften des Endproduktes.

Diese Tatsache soll als Ansatzpunkt für die Problemstellung dieser Arbeit dienen.

1.2 Zielsetzung für den betrachteten Fall

Ein Gleisprofil verläßt eine Walzstraße des Walzwerkes mit einer Temperatur von etwa 1000°C und soll nun zunächst auf ca. 700°C abgekühlt werden. Dabei sind verschiedene Faktoren zu berücksichtigen:

- Das Abkühlen darf aus Kostengründen nicht zu lange dauern.
- Es soll nach dem Kühlvorgang eine möglichst gleichmäßige Temperaturverteilung vorliegen und eine deutliche Senkung der Temperatur insgesamt erreicht werden.
- Es gibt mehrere Kühldüsen, die unabhängig voneinander angesprochen werden können, um ein gezieltes Abkühlen (z.B. des dickeren Gleiskopfes) zu ermöglichen.

Phasenübergänge, wie sie in [5] behandelt werden, bleiben unberücksichtigt, da der kritische Temperaturbereich unterhalb von 723°C nur knapp erreicht wird. Wir halten an dieser Stelle fest, daß das in dieser Arbeit vorgestellte Modell nur solche Walzvorgänge berücksichtigt, die der Formgebung der Profilstähle dienen. Sollen sich durch das Walzen die Materialeigenschaften verändern, so müssen Phasenübergänge natürlich berücksichtigt werden, da diese die Materialeigenschaften entscheidend beeinflussen.

1.3 Mathematische Formulierung

1.3.1 Das Rechengebiet

Wir betrachten das Gleisprofil als, gegenüber seiner Ausdehnung in horizontaler (x–Richtung) und vertikaler (y–Richtung) Richtung, unendlich lang. Damit rechtfertigt sich die Modellannahme, daß ein Wärmefluß in z–Richtung praktisch nicht auftritt. Wir ziehen uns daher bei den Rechnungen auf einen 2–dimensionalen Querschnitt des Gleisprofils zurück. Desweiteren nutzen wir die Symmetrie des Gleises bezüglich der Mittelsenkrechten, um das Rechengebiet zu halbieren. Damit wird insbesondere die Dimension der Systemmatrizen etwa halbiert, was den Aufwand speziell bei der Lösung des Steuerungsproblems erheblich verringert. Dazu führen wir auf der Mittelsenkrechten des Querschnitts einen künstlichen Rand ein, auf dem eine Isolationsrandbedingung vorgegeben wird, um die Symmetrie zu simulieren[1].

Wir werden nun zwei verschiedene Probleme betrachten. Einerseits geben wir an den übrigen Rändern eine Randbedingung vor, die den Wärmeausfluss entlang der äußeren Normalen proportional zur Differenz zwischen der Temperatur des Kühlfluids und der Temperatur der Profiloberfläche angibt. Andererseits geben wir an den übrigen

[1] d.h. wir verwenden eine Neumann–Randbedingung mit rechter Seite 0

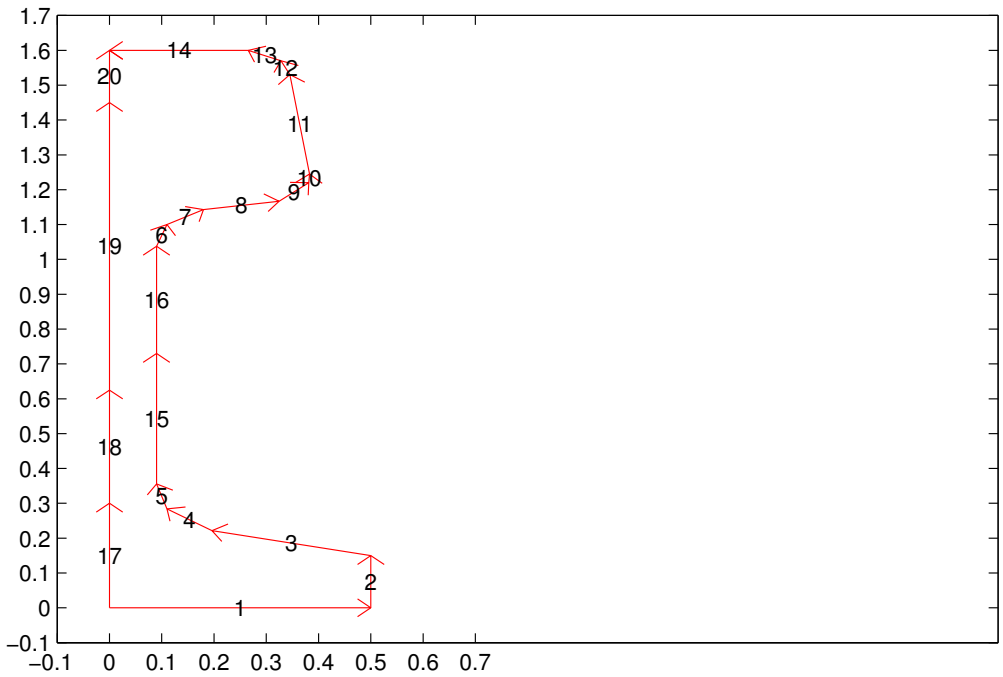


Abbildung 1.1: Das Rechengebiet, wie es mit MATLABS PDETool erstellt wurde. (Die angegebenen Randnummern werden im Kapitel 4 verwendet um zu erläutern, welcher Kühlparameter welchen Randstücken zugeordnet ist.)

Rändern eine gemischte Randbedingung (Randbedingung dritter Art) vor, Diese gibt das Ausfließen der Wärme entlang der äußeren Normalen aus dem Rechengebiet heraus, proportional zur Temperatur des Kühlfluides und abhängig von der Temperatur des Werkstückes am Rand an. Der wesentliche Unterschied zwischen beiden Fällen wird bei der mathematischen Formulierung augenscheinlich werden.

Das für die numerische Implementation verwendete Programmpaket ALBERT [2] bietet die Möglichkeit, parametrisierte Ränder zu verwenden. Dies nutzen wir aus, um die gekrümmten Randstücke als Kreissegmente zu modellieren, auf die jeder neue Randknoten bei seiner Erschaffung projiziert wird. Damit erhalten wir bei hoher Knotendichte am Rand eine bessere Approximation der realen Geometrie[3].

1.3.2 Die mathematischen Gleichungen

Tröltzsch und Unger haben in [25] das folgende Modell für den Wärmetransport in Stahl vorgestellt:

$$\begin{aligned} c(u)\varrho(u)\frac{\partial u(t,x)}{\partial t} &= \nabla.(\lambda(u)\nabla u(t,x)), && x \in \Omega,\ t \in (0,T], \\ u(0,x) &= u_0(x), && x \in \Omega, \\ \lambda(u)\partial_\nu u(t,x) &= g_i(t,x), && t \in (0,T],\ x \in \Gamma_i, i = 0,\ldots,7,\ g_0 \equiv 0. \end{aligned} \tag{1.1}$$

[2] siehe auch Kapitel 3.2 sowie [23, 24]

[3] Nähere Informationen zu den gekrümmten Rändern befinden sich Anhang C

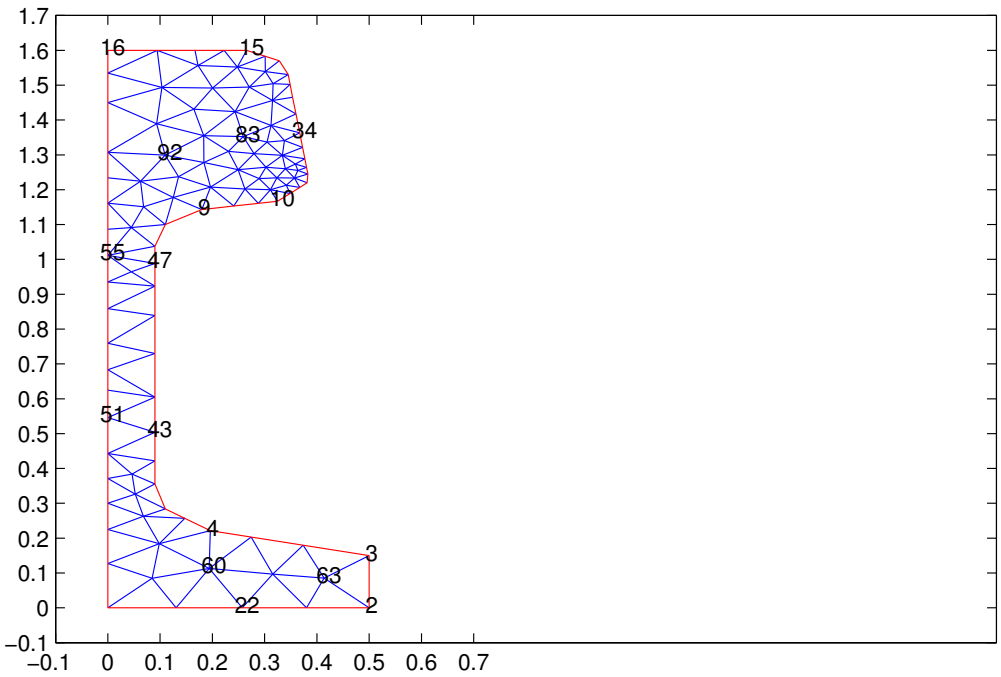


Abbildung 1.2: Das Startgitter, wie es MATLAB's PDETool erstellt hat. (Die ausgezeichneten Knoten werden in den Kapiteln 4 und 5 verwendet um Ausgangsmatrizen für die Testprobleme zu definieren.)

Dabei sind

u Wärmeverteilung ($\in W_2^1(0,T)$)

$c(u)$ Wärmekapazität ($: \Omega \to \mathbb{R}_{>0}$)

$\lambda(u)$ Wärmeleitfähigkeit ($: \Omega \to \mathbb{R}_{>0}$)

$\varrho(u)$ Dichte des Stahls ($: \Omega \to \mathbb{R}_{>0}$)

Wir linearisieren dieses Modell durch die Annahme, daß die Wärmekapazität c, die Wärmeleitfähigkeit λ und die Dichte ϱ im betrachteten Temperaturbereich von ca. 700-1000°C als konstant genähert werden können. Damit vereinfacht sich (1.1) zu

$$\begin{aligned} \partial_t u(t,x) &= \alpha \Delta u(t,x) && \text{in } \Omega \times (0,T] \text{ mit } \alpha := \tfrac{\lambda}{c\varrho}, \\ u(0,x) &= u_0(x) && \text{in } \Omega, \\ \partial_\nu u(t,x) &= \tfrac{1}{\lambda} g_i && \text{auf } \Gamma_i \times (0,T],\ i = 0,\dots,7, \end{aligned} \tag{1.2}$$

wobei wiederum $g_0 \equiv 0$ gilt.

Dabei sind die g_i jeweils Funktionen des Ortes, der Zeit, der Wärmeleitfähigkeit, der Kontrollparameter und nicht zuletzt auch der Lösung auf dem jeweiligen Rand. Die ursprüngliche Fassung in [25] enthällt außerdem einen Wärmeaustauschkoeffizienten. Diesen fassen wir in dieser Arbeit mit den Kontrollparametern zusammen.

Wie bereits angesprochen, werden wir zwei Sorten von Randbedingungen betrachten. Einerseits verwenden wir wie Tröltzsch und Unger eine Proportionalitätsrandbedingung[4] der Form

$$\lambda \partial_\nu u(t,x) = q_i(t,x)(u_{ext} - u(t,x)) \qquad t \in (0,T],\ i = 0,\dots,7,\ q_0 \equiv 0. \tag{1.3}$$

Wir werden diese Randbedingung im weiteren Verlauf dieser Arbeit als Abstrahlrandbedingung oder Robin–Randbedinung bezeichnen.

Im anderen Fall "linearisieren"[5] wir auch diese Randbedingungen, indem wir die Klammer auf der rechten Seite auflösen und den Koeffizienten von u, der hier die Gestalt eines Wärmeaustauschkoeffizienten hat, als konstant wählen. Wir erhalten damit eine Randbedingung dritter Art

$$\gamma_i(t,x)u + \lambda \partial_\nu u(t,x) = q_i(t,x)u_{ext}, \qquad t \in (0,T],\ i = 0,\dots,7,\ q_0 \equiv 0. \tag{1.4}$$

Diese Randbedingung bezeichnen wir fortan als gemischte Randbedinung , da sie eine Kombination aus einer Dirichlet– und einer Neumannrandbedingung darstellt. Im Folgenden wird angenommen, daß die Kühldüsen jeweils ein gesamtes Randstück gleichmäßig kühlen, also $q_i(t,x) = q_i(t)$ gilt. Wir verwenden hier nicht die Temperatur u_{ext} selbst als Kontrollparameter, sondern die eingeführten Koeffizienten q_i, da man ggf. nicht nur die Temperatur des Kühlfluides beeinflussen möchte, sondern auch die Intensität des aufgebrachten Strahles. Zu den berechneten Kontrollparametern muß noch diskutiert werden, welche Bedeutung sie für die technische Umsetzung haben. Insbesondere ist dabei zu betrachten, wie sich der in den q_i versteckte Intensitätsparameter aus dem mit der Strahlintensität variierenden Wärmeaustauschkoeffizienten ableitet.

1.3.3 Diskussion des mathematischen Modells

Die Schwächen dieses Modells liegen einerseits in der Linearisierung der Gleichung durch Wahl der konstanten Koeffizienten. Andererseits ist die gemischte Randbedingung nicht optimal, in dem Sinne, daß man den Temperaturausfluß über den Rand eigentlich proportional zur Temperaturdifferenz von Kühlfluid und Rand beschreiben möchte [6]. Dies wird daher für die Anlauf– und Referenzrechnungen in der Implementierung berücksichtigt. In den gesteuerten Rechnungen wird zunächst dennoch mit den gemischten Randbedingungen gerechnet, da die Formulierung mit der Abstrahlrandbedingung eine zusätzliche Koppelung der Koeffizientenmatrix B (vgl. (2.3)) mit der Lösung u bedeuten würde, die mit den verwendeten Steuerungsmethoden theoretisch nicht vereinbar ist. Dies nehmen wir hier explizit in Kauf, da wir nach einem effizienten und vor allem schnellen Löser suchen. Die Beurteilung der Qualität, der hiermit erzeugten Resultate orientiert sich an den Ergebnissen aus [15]. In den numerischen Experimenten werden wir auch Rechnungen mit der Proportionalitätsrandbedingung

[4] $\partial_\nu u = c(u - u_{ext})$ Diese Randbedingung wird auch als Ofenbedingung bezeichnet, da sie auch zur Simulation des Wärmeabstrahlverhaltens eines Ofens verwendet wird.

[5] "linearisieren" ist hier in dem Sinne zu verstehen, das wir durch dieses Vorgehen die Koeffizientenmatrix B im resultierenden Kontrollsystem von der Lösung u entkoppeln und damit die Kontrolle linearisieren. Dieses Problem wird in 1.3.3 erneut thematisiert.

[6] siehe z.B.: [5]

durchführen und die Ergebnisse mit denen aus den theoretisch abgesicherten Rechnungen vergleichen. Der Schwäche bezüglich der Linearisierung können wir in der PDE–Löser basierten Implementierung[7] dadurch Rechnung tragen, daß wir die Koeffizienten weiterhin in jedem Zeitschritt als konstant betrachten, um der Theorie gerecht zu werden, diese jedoch zu Beginn jedes Zeitschrittes gemäß der Gleichungen für die Materialparameter in der Austenitphase aus [5] Kapitel 4.

$$\begin{aligned} \varrho(u) &= -0.4553u + 7988 \\ \lambda(u) &= 0.0127u + 14.6 \\ c(u) &= 0.1756u + 454.4 \end{aligned} \tag{1.5}$$

mit der Lösung u_{old} aus dem vorhergehenden Zeitschritt aktualisieren. Ein ähnliches Vorgehen ist mit dem Koeffizienten γ aus der gemischten Randbedingung ebenfalls denkbar, wurde in der vorliegenden Arbeit aber aus Zeitgründen nicht mehr implementiert. In einer ungesteuerten Referenzrechnung wird das Vorgehen bzgl. der Koeffizienten dazu verwendet zu Prüfen, wie sich die Linearisierung durch globale Mittelung auf das Endergebnis auswirkt.

[7] PDE–Zugang (siehe Abschnitt 3.1.2)

Kapitel 2

Mathematische Grundlagen

An dieser Stelle werden zunächst die Begriffe definiert, die nötig sind, um die Stabilisierbarkeit des Problems zu untersuchen. Später folgt ein Resultat von Banks und Kunisch[1], welches Bedingungen für die Stabilisierbarkeit von parabolischen partiellen Differentialgleichungen liefert. Abschließend wird daraus die Stabilisierbarkeit des vorliegenden Problems gefolgert. Der vorletzte Abschnitt dieses Kapitels leitet dazu aus der schwachen Formulierung der Modellgleichung die diskrete Gleichung her.

2.1 Approximation eines abstrakten linear–quadratischen Optimierungsproblems in Hilberträumen

2.1.1 Einführung und grundlegende Definitionen

Halbgruppen von Operatoren

In diesem Abschnitt werden wir in einigen Definitionen die Begriffe bereitstellen, die wir benötigen um festzulegen, was wir unter einem linear–quadratischen Optimierungsproblem in Hilberträumen verstehen wollen. Desweiteren halten wir wichtige Eigenschaften der definierten Begriffe fest, welche im weiteren Verlauf von Belang sind. Die hier angegebenen Definitionen orientieren sich an [29], [19]. Eine lehrbuchartige Einführung in diese Thematik geben Dunford und Schwartz in [9]. Es werden aber auch Resultate aus [10] und [21] zitiert.

Definition 2.1.1 (Stark stetige Halbgruppe):

Sei Z ein Hilbertraum. Eine *stark stetige Halbgruppe von linearen Operatoren* wird definiert durch $T : \mathbb{R}^+ \to \mathcal{L}(Z)$ mit

$$\begin{aligned} T(0) &= I \\ T(s+t) &= T(s)T(t) \\ \|T(t)z - z\| &\to 0 \qquad \text{für } t \to 0 \text{ und alle } z \in Z \end{aligned} \tag{2.1}$$

[1] (1984 in [2])

Im Folgenden werden wir den Begriff der stark stetigen Halbgruppe häufig durch $\mathcal{C}_0$-Halbgruppe abkürzen. Gilt die Stetigkeitsaussage sogar bezüglich der Operatornorm und damit bezüglich der gleichmäßigen Operatortopologie, wie sie in [9] eingeführt wird, so sprechen wir von einer *gleichmäßig stetigen Halbgruppe*.

Definition 2.1.2 (infinitesimaler Generator):

Sei $T(t)$ eine $\mathcal{C}_0$–Halbgruppe. Zu $h > 0$ definieren wir den linearen Operator A_h durch

$$A_h z = \frac{T(h)z - z}{h} \text{ für } z \in Z$$

Sei $\mathcal{D}(A)$ die Menge aller $z \in Z$ für die der Grenzwert $\lim_{h\to 0} A_h z$ existiert. Wir definieren damit den Operator A mit $\operatorname{dom} A = \mathcal{D}(A)$ durch

$$Az = \lim_{h\to 0} A_h z \text{ für } z \in \mathcal{D}(A) \tag{2.2}$$

Der auf diese Weise definierte Operator A heißt *infinitesimaler Generator*[2]der Halbgruppe $T(\cdot)$

Per Definition ist klar, daß $\mathcal{D}(A)$ ein linearer Teilraum von Z ist und A ein auf $\mathcal{D}(A)$ linearer Operator. Die Bezeichnung als *infinitesimaler* Generator wird aus der Definition sofort klar. Allerdings bleibt zu klären inwiefern A die Halbgruppe $T(t)$ *generiert*. Dazu betrachten wir folgenden

Satz 2.1.3:

Sei $T(t)$ eine stark stetige Halbgruppe von Operatoren $t \geq 0$, seien weiter A_h wie in der Definition gegeben durch

$$A_h = \frac{T(h) - I}{h}$$

Dann gilt

$$T(t)z = \lim_{h\to 0} e^{tA_h} z, \text{ für } z \in Z$$

gleichmäßig in t auf jedem endlichen Intervall.

Dabei ist e^{tA} für lineare und stetige Operatoren A definiert durch

$$e^{tA} := \sum_{i=0}^{\infty} \frac{t^k A^k}{k!}.$$

In diesem Sinne generiert A also die Halbgruppe $T(t)$. Eine naheliegende Frage ist jetzt natürlich, welche Eigenschaften der in (2.2) definierte Operator A besitzt, bzw. wann es sich bei A um einen beschränkten Operator handelt oder wir eine Darstellung $T(t) = e^{tA}$ erhalten. Diese Frage beantwortet folgender

Satz 2.1.4:

Jede gleichmäßig stetige Halbgruppe $(T(t))_{t\geq 0}$ auf einem Banachraum X hat die Form

$$T(t) = e^{tA} \text{ mit } t \geq 0$$

für einen beschränkten Operator $A \in \mathcal{L}(X)$.

[2]Bei einigen Autoren wird dieser Operator auch als *Erzeuger* der Halbgruppe bezeichnet.

Zum Beweis siehe [10], Kapitel 1.

Ein nützliches Resultat zur Charakterisierung der Generatoren von exponentiell beschränkten $\mathcal{C}_0$–Halbgruppen liefert der folgende Satz aus [21].

Satz 2.1.5:

Ein linearer Operator A ist der infinitesimale Generator einer $\mathcal{C}_0$–Halbgruppe $T(t)$ auf dem Banachraum X, die $\|T(t)\| \leq Me^{\omega t}$ erfüllt, genau dann wenn

1. A ist abgeschlossen und $\operatorname{dom} A$ ist dicht in X,
2. Die Resolventenmenge $\varrho(A)$ von A umfaßt $(\omega, \infty) \subset \mathbb{R}$ und $\|(\lambda I - A)^{-n}\| \leq \frac{M}{(\lambda-\omega)^n}$ für $\lambda > \omega$ und $n = 1, 2, \ldots$

Linear–quadratische Optimierungsprobleme in Hilberträumen

Im Folgenden wird stets angenommen, daß der Zustandsraum H und der Eingangsraum U Hilberträume sind. Wir betrachten das System

$$\begin{aligned} \dot{u}(t) &= Au(t) + Bq(t), \quad \text{mit } t > 0, \\ u(0) &= u_0, \end{aligned} \tag{2.3}$$

wobei $B \in \mathcal{L}(U, H)$ und $A : \operatorname{dom} A \subset H \to H$ infinitesimaler Generator der stark stetigen Halbgruppe $T(t)$ auf H ist. Weiter betrachten wir das zugehörige Kostenfunktional

$$\mathcal{J}(u_0, q) = \int_0^\infty (Qu(t), u(t)) + (Rq(t), q(t))dt \tag{2.4}$$

Dabei seien $Q \in \mathcal{L}(H)$, $R \in \mathcal{L}(U)$ selbstadjungierte Operatoren mit $Q \geq 0$, $R > 0$. Damit lautet unser Optimierungsproblem: Minimiere $\mathcal{J}(u_0, q)$ über $q \in L^2(0, \infty; U)$ unter der Nebenbedingung (2.3) an u.

Definition 2.1.6 (zulässige Steuerung):

Wir nennen $q(t) \in L^2(0, \infty; U)$ eine *zulässige Steuerung zum Anfangswert* u_0, falls gilt $\mathcal{J}(u_0, q) < \infty$.

Wie im endlich dimensionalen Fall[3] ist auch hier eine algebraische Riccatigleichung zu lösen. Der lineare Operator $\Pi \in \mathcal{L}(H)$ heißt *Lösung der algebraischen Riccatigleichung* (ARE), falls Π: $\operatorname{dom} A \to \operatorname{dom} A^*$ erfüllt und auf H die Gleichung

$$\mathfrak{R}(\Pi) := Q + A^*\Pi + \Pi A - \Pi B R^{-1} B^* \Pi = 0 \tag{2.5}$$

löst. D.h. $(u, \mathfrak{R}(\Pi)v) \equiv 0$.

[3] vgl. Anhang A

2.1.2 Die Resultate von Banks und Kunisch

Der folgende Satz ist eine zusammenfassende Folgerung aus Resultaten, die Gibson in [12] entwickelt hat.

Satz 2.1.7:

Seien A, B, Q, R wie in (2.3), (2.4) gefordert. Dann existiert genau dann eine nichtnegative selbstadjungierte Lösung Π der algebraischen Riccatigleichung, wenn es zu jedem $u_0 \in H$ eine zulässige Steuerung gibt. Die optimale Steuerung q_* und die zugehörige Lösungstrajektorie u_* zum obigen Optimierungsproblem sind in diesem Fall gegeben durch

$$q_*(t) = -R^{-1}B^*\Pi_\infty u_*(t) \tag{2.6}$$
$$u_*(t) = S(t)u_0 \tag{2.7}$$

Hier ist Π_∞ die minimale nichtnegative selbstadjungierte Lösung der ARE (2.5) und $S(t)$ die von $A - BR^{-1}B^*\Pi_\infty$ erzeugte stark stetige Halbgruppe.
Gilt darüberhinaus $|u(t;q)| \to 0$ falls $t \to \infty$ für jede zulässige Steuerung (z.B. für $Q > 0$), dann ist Π_∞ die eindeutige nichtnegative selbstadjungierte Lösung der ARE (2.5). Ist außerdem $Q > 0$ so ist $S(t)$ gleichmäßig exponentiell stabil.

Wir wollen nun sehen, daß die minimale Lösung Π_∞ durch eine Folge von Lösungen von Problemen in endlich dimensionalen, linearen Teilräumen H^N von H gewonnen werden kann. Dazu formulieren wir zunächst eine Folge approximierender Optimierungsprobleme in endlichen Dimensionen und werden später ein Konvergenzresultat für die zugehörigen Riccatioperatoren finden. Seien dazu H^N für $N = 1, 2, \ldots$ endlichdimensionale lineare Teilräume von H, $P^N : H \to H^N$ die zugehörigen kanonischen, orthogonalen Projektionen. Sei $T^N(t)$ eine Folge von $\mathcal{C}_0$-Halbgruppen auf den Räumen H^N mit infinitesimalen Generatoren $A^N \in \mathcal{L}(H^N)$. Zu gegebenen Operatoren $B^N \in \mathcal{L}(U, H^N)$ und $Q^N \in (H^N)$ betrachten wir eine Familie von Optimierungsproblemen definiert durch:

Minimiere $\mathcal{J}^N(u^N(0), q)$ über $q \in L^2(0, \infty; U)$,

wobei

$$\begin{aligned} \dot{u}^N(t) &= A^N u^N(t) + B^N q(t), \text{ für } t > 0 \\ u^N(0) &= u_0^N := P^N u_0 \end{aligned} \tag{2.8}$$

und

$$\mathcal{J}^N(u^N(0), q) = \int_0^\infty (Q^N u^N(t), u^N(t)) + (Rq(t), q(t))dt \tag{2.9}$$

Wir halten fest, daß mit $B^N : U \to H^N$ die Lösungstrajektorien zum linearen, zeitinvarianten System (*LTI-System*) (2.8) in H^N verlaufen. Daher dürfen wir die obigen Probleme als Optimierungsprobleme in den endlich dimensionalen Teilräumen H^N betrachten. Es ist damit die im Anhang A betrachtete endlich dimensionale Theorie anwendbar, falls wir zu H^N eine Basis wählen können, bzgl. derer die Operatoren eine reelle Matrixrepräsentation besitzen.

Wir werden nun einige Voraussetzungen formulieren, die man benötigt, um eine Konvergenzaussage über die Folge der endlich dimensionalen Probleme zu treffen.

i.) Für jedes $u_0^N \in H^N$ existiert eine zulässige Steuerung $q^N \in L^2(0,\infty,U)$ zum endlich dimensionalen Optimierungsproblem (2.8),(2.9) und jede zulässige Steuerung für (2.8),(2.9) läßt den Zustand u^N asymptotisch gegen 0 streben.

ii.) Für alle $z \in H$ gilt $T^N(t)P^Nz \to T(t)z$ gleichmäßig in t auf jedem beschränkten Teilintervall von $[0,\infty)$.

iii.) Für alle $z \in H$ gilt $T^N(t)^*P^Nz \to T(t)^*z$ gleichmäßig in t auf jedem beschränkten Teilintervall von $[0,\infty)$.

iv.) Für alle $v \in U$ gilt $B^Nv \to Bv$ und für alle $z \in H$ gilt $B^{N*}z \to B^*z$.

v.) Für alle $z \in H$ gilt $Q^NP^Nz \to Qz$.

Können wir nun zu H^N eine Basis wählen, bezüglich derer die gegebenen Operatoren eine reelle Matrixrepräsentation besitzen[4] und identifizieren wir die Operatoren und Vektoren mit ihren Darstellungen in dieser Basis, so ist i. eine direkte Folge aus der Theorie für endlich dimensionale Probleme aus Anhang A. Bedingung ii.) bedeutet insbesondere (mit $t = 0$), daß $P^Nz \to z$ für alle $z \in H$. In diesem Sinne approximieren die endlich dimensionalen Teilräume H^N unseren Hilbertraum H.[5] Falls i.) gilt so wissen wir nach Satz A.2.4, daß die optimale Steuerung q_* zum Problem (2.8),(2.9) in Feedbackform gegeben ist durch

$$q_*(t) = -R^{-1}(B^N)^T\Pi^N u_*^N(t), \tag{2.10}$$

wobei $\Pi^N \in \mathbb{R}^{N,N}$ die eindeutige nichtnegative symmetrische Lösung der algebraischen Riccatigleichung auf H^N ist:

$$0 = Q^N + (A^N)^T\Pi^N + \Pi^N A^N - \Pi^N(B^N)^T\Pi^N \tag{2.11}$$

und $u_*(t)$ die gemäß (2.8) zu $q_*^N(t)$ gehörende Lösungstrajektorie. Desweiteren sind nach Satz A.2.4 die optimalen Kosten gegeben durch

$$\mathcal{J}^N(u_0^N, q_*^N) = (\Pi^N u_0^N, u_0^N). \tag{2.12}$$

Existiert also eine Folge solcher Basen zu den H^N, die ii.) erfüllen, dann können wir folgenden Satz formulieren.

Satz 2.1.8 (Konvergenz der endlich dimensionalen Approximationen):

Seien die Voraussetzungen i.)–v.) erfüllt. Seien außerdem $R > 0$, $Q \geq 0$ und $Q^N \geq 0$, sowie Π^N die Lösungen der ARE zu den endlich dimensionalen Problemen (2.8),(2.9). Weiter existiere ein selbstadjungierter nichtnegativer Riccatioperator Π auf H zum linear–quadratischen Optimierungsproblem (2.3),(2.4). Seien $S(t)$ und $S^N(t)$ die von $A - BR^{-1}B^*\Pi$ bzw. $A^N - B^NR^{-1}B^{N^*}\Pi^N$ auf H bzw. H^N generierten Halbgruppen und

[4] Wie im vorliegenden Fall durch die Approximation des Problem mit der Methode der finiten Elemente

[5] Die Finite–Element–Approximation ist gerade so beschaffen, daß sie diese Bedingung in natürlicher Art und Weise erfüllt.

es gelte $|S(t)z| \to 0$ falls $t \to \infty$ für alle $z \in H$.
Falls es positive Konstanten M_1, M_2 und ω unabhängig von N und t gibt, so daß

$$|S^N(t)|_{H^N} \leq M_1 e^{-\omega t} \quad \text{für } t \geq 0 \text{ und } N = 1, 2, \ldots \tag{2.13}$$
$$|\Pi^N|_{H^N} \leq M_2. \tag{2.14}$$

Dann konvergieren

$$\Pi^N P^N z \to \Pi z \qquad \text{für alle } z \in H, \tag{2.15}$$
$$S^N(t) P^N z \to S(t) z \qquad \text{für alle } z \in H \tag{2.16}$$

gleichmäßig in t auf beschränkten Teilmengen von $[0, \infty)$ und es ist

$$|S(t)| \leq M_1 e^{-\omega t} \text{ für } t \geq 0. \tag{2.17}$$

2.2 Schwache Formulierung und Diskretisierung des Wärmeleitungsproblems

In diesem Abschnitt werden wir die Finite–Elemente–Diskretisierung von Gleichung (1.2) herleiten, die wir in Kapitel 3.2 für die numerische Implementierung benötigen. [6] Aus der Variationsformulierung dieser Gleichung mit orts– und zeitunabhängigen Koeffizienten erhalten wir für alle $v \in H^1(\Omega)$

$$\begin{aligned}
(\partial_t u, v) &= (\frac{1}{c\varrho} \nabla . \lambda \nabla u, v) \\
&= \int_\Omega \frac{1}{c\varrho} (\nabla . \lambda \nabla u) v \, dx = -\int_\Omega \alpha \nabla u . \nabla v \, dx + \int_\Gamma \alpha \partial_\nu u \, v \, d\sigma \\
&= -\int_\Omega \alpha \nabla u \nabla v dx + \sum_{k=1}^{7} \int_{\Gamma_k} \frac{1}{c\varrho} \left(q_k u_{ext} - \gamma_k u \right) v \, d\sigma \\
&= -\int_\Omega \alpha \nabla u \nabla v dx + \sum_{k=1}^{7} \left(q_k u_{ext} \int_{\Gamma_k} \frac{1}{c\varrho} v \, d\sigma - \int_{\Gamma_k} \frac{\gamma_k}{c\varrho} u v \, d\sigma \right).
\end{aligned} \tag{2.18}$$

Die diskrete Formulierung von (2.18) lautet dann

$$\begin{aligned}
\sum_{i=1}^{n_d} (\partial_t u)_i \int_\Omega \varphi_i . \varphi_j \, dx &= -\sum_{i=1}^{n_d} u_i \int_\Omega \alpha \nabla \varphi_i . \nabla \varphi_j \, dx \\
&\quad + \sum_{k=1}^{7} q_k u_{ext} \int_{\Gamma_k} \tfrac{1}{c\varrho} \varphi_j \, d\sigma \\
&\quad - \sum_{k=1}^{7} \sum_{i=1}^{n_d} u_i \int_{\Gamma_k} \tfrac{\gamma_k}{c\varrho} \varphi_i \varphi_j \, d\sigma \quad \text{für alle } \varphi_j \in X_h
\end{aligned} \tag{2.19}$$

[6] Eine für eine Diplomarbeit angemessene Einführung in die Theorie der finiten Elemente befindet sich in [14]. In dieser Arbeit soll auf eine solche Einführung verzichtet werden, da die finiten Elemente hier nur als Werkzeug verwendet werden. Eine ausführliche Einführung in diese Theorie liefert z.B. [6]. Die benötigten Grundlagen aus dem Bereich der linearen Funktionalanalysis stellen [1, 28] zur Verfügung.

mit $j = 1, \ldots, n_d$, wobei n_d die Anzahl der Freiheitsgrade ist, also die Anzahl der Ansatzfunktionen φ_i in der Basis des Finite– Elemente–(FE-)–Raumes X_h. Die φ_i sind dabei als polynomiale Ansatzfunktionen mit $\varphi_i(x_j) = \delta_{ij}$ gewählt. $(\partial_t u)_i$ und die $(u)_i$ sind jeweils die Auswertungen von $\partial_t u$ bzw. u im i–ten Freiheitsgrad, also in x_i.[7]

Definiert man nun

$$\begin{aligned} S &:= \left(\alpha \int_\Omega \nabla\varphi_i \nabla\varphi_j \, dx \right)_{i,j=1,\ldots,n_d} && \in \mathbb{R}^{n_d,n_d}, \\ M &:= \left(\int_\Omega \varphi_i \varphi_j \, dx \right)_{i,j=1,\ldots,n_d} && \in \mathbb{R}^{n_d,n_d}, \\ M_\Gamma &:= \sum_{k=1}^{7} \left(\int_{\Gamma_k} \frac{\gamma_k}{c\varrho} \varphi_i \varphi_j \, d\sigma \right)_{i,j=1,\ldots,n_d} && \in \mathbb{R}^{n_d,n_d}, \\ B &:= u_{ext} \left(\int_{\Gamma_k} \frac{1}{c\varrho} \varphi_j \, d\sigma \right)_{j=1,\ldots,n_d \; ; \; k=1,\ldots,7} && \in \mathbb{R}^{n_d,7}, \end{aligned} \tag{2.20}$$

dann läßt sich (2.19) schreiben als lineares gewöhnliches Differentialgleichungssytem

$$M\dot{u} + Su = Bq - M_\Gamma u \Leftrightarrow M\dot{u} + (S + M_\Gamma)u = Bq \tag{2.21}$$

2.3 Existenz der Lösung zum vorliegenden Steuerungsproblem

Jetzt wollen wir die schwache Formulierung aus dem vorigen Abschnitt verwenden, um die approximierenden endlichdimensionalen Probleme aus der "Banks und Kunisch"–Theorie näher zu spezifizieren und damit Aussagen über die Gültigkeit der Forderungen i-v zu gewinnen.

Das vorliegende Steuerungsproblem

Wir betrachten die schwache Formulierung aus (2.18), um das vorliegende Problem dieser Arbeit in eine Gleichung der Form (2.3) zu überführen und damit die Theorie von Banks und Kunisch verwenden zu können. Dazu betrachten wir wieder die linearisierte Gleichung, in der wir die Koeffizienten c, ϱ und λ als Konstanten ansehen und in der Praxis durch „gute“[8] Mittelwerte ersetzen. Wir schreiben dazu Gleichung (2.18)

[7] siehe auch [14] im Kapitel 3.3 Polynomiale Finite–Elemente–Räume, oder [6] in Kapitel 2 §5 "Einige gebräuchliche Finite–Elemente".

[8] "gut" ist hier in dem Sinne zu verstehen, daß wir auf Messungen und Erfahrungswerte der Anwender zurückgreifen, aus deren Fachgebiet dieses Problem stammt.

wiederum für alle $v \in H^1(\Omega)$ in der Form[9]

$$(\partial_t u, v)_{L^2(\Omega)} + \alpha\left((\nabla u, \nabla v)_{L^2(\Omega)} + \frac{\gamma}{\lambda}(\mathrm{Sp}\, u, \mathrm{Sp}\, v)_{L^2(\partial\Omega)}\right) - \frac{1}{c\varrho}\sum_{i=1}^{7} q_i(\chi_{\Gamma_i} u_{ext}, \mathrm{Sp}\, v)_{L^2(\partial\Omega)} = 0. \tag{2.22}$$

Definieren wir nun:

$$\begin{array}{rccl} A: & H^1(\Omega) & \to & H^1(\Omega)', \\ & u & \mapsto & \alpha\left((\nabla u, \nabla\cdot)_{L^2(\Omega)} + \frac{\gamma}{\lambda}(\mathrm{Sp}\, u, \mathrm{Sp}\,\cdot)_{L^2(\partial\Omega)}\right), \\ B: & U & \to & H^1(\Omega)', \\ & & \mapsto & \frac{1}{c\varrho}\sum_{i=1}^{7} q_i(\chi_{\Gamma_i} u_{ext}, \mathrm{Sp}\,\cdot)_{L^2(\partial\Omega)}, \\ M: & H^1(\Omega) & \to & H^1(\Omega)', \\ & u & \mapsto & (\partial_t u, \cdot)_{L^2(\Omega)}, \end{array} \tag{2.23}$$

und damit

$$\sigma_A(v, w) :=< Av, w > +\alpha(v, w)_{L^2(\Omega)}, \tag{2.24}$$

dann erhalten wir mit σ_A eine stetige und koerzive[10] Sesquilinearform auf $H^1(\Omega)$. Denn $\sigma_A(v,v) = \alpha(\|v\|^2_{1,2} + \gamma\|v\|^2_{L^2(\partial\Omega)})$, also gilt insbesondere $\sigma_A(v,v) \geq \alpha\|v\|^2_{1,2}$. Außerdem gilt $\sigma_A(u,v) = \alpha((\nabla u, \nabla v)_2 + \gamma(\mathrm{Sp}\, u, \mathrm{Sp}\, v)_{L^2(\partial\Omega)} + (u,v)_2)$ und mit der Cauchy–Schwarz–Ungleichung und Stetigkeit der Spur folgt die Stetigkeit von σ_A. Damit liefert uns der Satz von Lax–Milgram in der Version aus [1] die Existenz von invertierbaren linearen beschränkten Operatoren $A_\alpha \in \mathcal{L}(H^1(\Omega))$, $A^*_\alpha \in \mathcal{L}(H^1(\Omega))$, so daß gilt

$$\begin{array}{rcl} \sigma_A(v,w) & = & (-A_\alpha v, w)_{H^1(\Omega)}, \\ \overline{\sigma_A(v,w)} & = & (-A^*_\alpha w, v)_{H^1(\Omega)}. \end{array} \tag{2.25}$$

Wir erhalten damit ein Problem der Form (2.3), denn (2.22) wird mit der Anfangsbedingung des Modells zu

$$\begin{array}{rcll} \partial_t u & = & -A_\alpha u + B\tilde{q} & \text{in } \Omega, \\ u(0,\cdot) & = & u_0 & \text{in } \Omega. \end{array} \tag{2.26}$$

Die Koerzivität der Sesquilinearform liefert nun mit (2.25)

$$\begin{array}{rcl} \mathrm{Re}(A_\alpha z, z) & \leq & -c_1\|z\|^2_{1,2}, \\ \mathrm{Re}(A^*_\alpha z, z) & \leq & -c_1\|z\|^2_{1,2}. \end{array} \tag{2.27}$$

Damit ist A_α ein in $H^1(\Omega)$ dicht definierter dissipativer linearer Operator. Ebenso ist A^*_α ein dicht definierter und dissipativer[11] linearer Operator. Nach Korollar 4.4 aus Kapitel 1 in [21] sind A_α, A^*_α damit die infinitesimalen Generatoren von $\mathcal{C}_0$–Halbgruppen $T_\alpha(t)$

[9] Zum Spuroperator und schwachen Randwerten von Sobolev–Funktionen siehe [1] im Anhang zu Kapitel 6.

[10] Eine Sesquilinearform $\sigma(x,y)$ heißt koerziv auf X, falls es eine Konstante $c > 0$ gibt, so daß für alle $x \in X$ $\sigma(x,x) \geq c\|x\|^2$

[11] siehe Definition in der Einleitung

und $T^*_\alpha(t)$. Wir halten noch fest, daß die Lösungs–Halbgruppe $T(t)$ zum ungesteuerten Problem dann gegeben ist durch

$$T(t) = e^{\alpha t} T_\alpha(t) \tag{2.28}$$

und ihr infinitesimaler Generator A durch $A = A_\alpha + \alpha I$ mit dem gleichen Definitionsbereich wie schon A_α. Analog erhalten wir für $T^*(t)$:

$$T^*(t) = e^{\alpha t} T^*_\alpha(t)$$

generiert durch $A^* = A^*_\alpha + \alpha I$. Außerdem gilt mit (2.27) offenbar

$$\|T(t)\| \leq e^{(\alpha - c_1)t} \text{ für } t \geq 0.$$

Die approximierenden, endlich dimensionalen Probleme

Wir widmen uns nun der Approximation durch endlich dimensionale Probleme. Dazu wollen wir zunächst die in [2] geforderte Approximationsbedingung (C1) wiederholen. Sie lautet in der auf unser Problem angepassten Formulierung: "Zu jedem $v \in H^1(\Omega)$ gibt es ein $v^N \in H^N$ derart, daß $\|v - v^N\|_{1,2} \leq \varepsilon(N)$ mit $\varepsilon(N) \to 0$ für $N \to \infty$."

Offenbar wird diese Bedingung von der Galerkinapproximation, wie sie der verwendeten FE-Methode zugrundeliegt, bereits erfüllt.

Schränken wir nun (2.24) auf die approximierenden Teilräume H^N ein, so erhalten wir analog zum Vorgehen in (2.24) bis (2.27) lineare beschränkte Operatoren A^N_α und A^{N*}_α mit

$$\begin{aligned} \sigma_A|_{H^N \times H^N}(v, w) &= (-A^N_\alpha v, w) \\ \overline{\sigma_A|_{H^N \times H^N}(v, w)} &= (-A^{*N}_\alpha w, v) \end{aligned} \tag{2.29}$$

Ebenso analog zur Vorgehensweise in (2.23)-(2.27) sind $A^N : H^N \to H^{N\prime} \cong H^N$ und $A^{N*} : H^N \to H^N$ definiert und generieren die $\mathcal{C}_0$–Halbgruppen $T^N(t)$ bzw. $T^N(t)^*$.

Die Approximationen B^N von B und Q^N von Q aus Satz 2.1.8 und davor sind definiert durch

$$B^N = P^N B, \qquad Q^N = P^N Q, \tag{2.30}$$

wobei $P^N : L^2(\Omega) \to H^N$ die kanonische orthogonale Projektion ist, wie sie auch im Galerkin–Verfahren verwendet wird. Wie bereits im Abschnitt 2.1.1 diskutiert, ergeben sich mit diesen Setzungen, durch die Wahl unserer Basis aus dem Galerkinverfahren, endlich dimensionale Probleme, die mit der Theorie aus Anhang A behandelt werden können.

Verifikation der Konvergenzvoraussetzungen

Die folgende Definition liefert einige Begriffe, die wir für die weitere Diskussion des Problems benötigen werden.

Definition 2.3.1:

Das Operatorpaar (A, B) heißt *exponentiell stabilisierbar*, falls es Konstanten $M \geq 1$ und $\omega \geq 0$ sowie einen Operator $K : \operatorname{dom} A \to \operatorname{dom} B$ gibt, derart das die von $A - BK$ generierte Operatorhalbgruppe $T(t)$ die Bedingung $\|T(t)\| \leq Me^{-\omega t}$ erfüllt.

Die Familie $(A^N, P^N B)_{N\in\mathbb{N}}$ heißt *gleichmäßig exponentiell stabilisierbar* falls M, ω und K unabhängig von N gewählt werden können.

In diesem Falle heißt die Halbgruppe $T(t)$ *exponentiell stabil*, die Familie $(T^N(t))_{N\in\mathbb{N}}$ von Halbgruppen heißt entsprechend *gleichmäßig exponentiell stabil*.

Die Anwendbarkeit der Theorie aus Anhang A liefert uns mit den Eigenschaften der Approximation sofort die Erfüllung der Bedingung i.) aus Abschnitt 2.1. Wir wollen uns nun also der Verifikation der darauffolgenden Bedingungen ii.)-v.) widmen. iv.) und v.) ergeben sich hier sofort aus (2.30), da die Galerkinapproximation insbesondere $P^N v \to v$ für $N \to \infty$ und $v \in L^2(\Omega)$ liefert. Die Bedingungen ii.) und iii.) werden wir nun wie folgt angehen. Zunächst zeigen wir mit Hilfe eines Lemmas aus [11] die Konvergenzen

$$\begin{aligned} T_\alpha^N(t)P^N v &\to T_\alpha(t)v, \\ T_\alpha^N(t)^* P^N v &\to T_\alpha(t)^* v, \end{aligned} \tag{2.31}$$

wobei $v \in L^2(\Omega)$ lokal gleichmäßig in t. Zusammen mit den Definitionen von $T(t)$ und $T^*(t)$ erhalten wir dann die gewünschten Bedingungen. Um diese Konvergenzen nachzuweisen, halten wir zunächst fest, daß (2.27) ebenso auch für die A_α^N und A_α^{N*} und alle $v \in H^N$ gilt. Gelingt es uns nun

$$\begin{aligned} (I - A_\alpha^N)^{-1}P^N v &\to (I - A_\alpha)^{-1}v \text{ für } v \in L^2(\Omega) \text{ und} \\ (I - A_\alpha^{N*})^{-1}P^N v &\to (I - A_\alpha{}^*)^{-1}v \text{ für } v \in L^2(\Omega) \end{aligned} \tag{2.32}$$

nachzuweisen, so können wir mit dem Trotter–Kato Theorem [12] die Konvergenz in (2.31) folgern.

Wir haben also zu zeigen, das (2.32) gilt. Dazu bedienen wir uns des bereits angesprochenen Resultats aus [11][13].

Lemma 2.3.2:

Es gibt eine Konstante δ_1, sowie ein $\theta_1 \in (0, \pi/2)$, so daß

$$|\lambda|\|v\|_2^2 + \|v\|_1^2 \leq \delta_1|\lambda|\|v\|^2 - \sigma_A(v, v)| \tag{2.33}$$

für alle $v \in H^1(\Omega)$ und $\lambda \in \{\zeta \in \mathbb{C} : |\arg \zeta| \leq \pi\}$.

Wir definieren nun $w := (I - A_k)^{-1}v$ und $w^N := (I - A_k^N)^{-1}P^N v$. Dann gilt für alle $v^N \in H^N$:

$$\begin{array}{rcccccl} (w, v^N) + \sigma_A(w, v^N) & = & (w, v^N) + (-A_\alpha w, v^N) & = & ((I - A_\alpha)w, v^N) & = & (v, v^N) \\ (w^N, v^N) + \sigma_A(w^N, v^N) & = & (w^N, v^N) + (-A_\alpha^N w^N, v^N) & = & ((I - A_\alpha^N)w^N, v^N) & = & (v, v^N) \end{array} \tag{2.34}$$

[12] siehe [21] Kapitel 3 Theorem 4.4.

[13] Die Version in [11] Lemma 3.3 ist zwar für $\mathring{H}^1(\Omega)$ formuliert, nutzt die Nullrandwerte aber nirgendwo aus.

Definieren wir nun $e^N := w - w^N$, so ergibt sich daraus

$$(e^N, v^N) + \sigma_A(e^N, v^N) = 0 \text{ für alle } v^N \in H^N. \tag{2.35}$$

Setzen wir nun in (2.33) $\lambda = -1$ und $v = e^N$ so erhalten wir mit (2.35)

$$\|e^N\|_2^2 + \|e^N\|_{1,2}^2 \leq \delta_1 | - \|e^N\|_2^2 - \sigma_A(e^N, e^N)| = \delta_1 | - (e^N, e^N + v^N) - \sigma_A(e^N, e^N + v^N)| \tag{2.36}$$

für alle $v^N \in H^N$. Sei nun $\tilde{w}^N \in H^N$ eine Approximation an w, die (C1) erfüllt. Wählen wir in (2.36) $v^N = w^N - \tilde{w}^N$, dann folgt

$$\begin{aligned} \|e^N\|_2^2 + \|e^N\|_{1,2}^2 &\leq \delta_1 |(e^N, w - \tilde{w}^N) + \sigma_A(e^N, w - \tilde{w}^N)| \\ &\leq c_2 \delta_1 \varepsilon(N) \left(\|e^N\|_2 + \|e^N\|_{1,2} \right) \end{aligned} \tag{2.37}$$

Dabei ist c_2 die Konstante aus der Stetigkeitsbedingung an σ_A, welche o.B.d.A als größer 1 angenommen werden kann. Damit wissen wir $e^N \to 0$ für $N \to \infty$. Somit gilt Gleichung 1 aus (2.32) und damit Bedingung ii.). Um die zweite Gleichung in (2.32) zu erhalten, definieren wir $\tau : H^1(\Omega) \times H^1(\Omega) \to \mathbb{C}$ durch $\tau(v, w) = \overline{\sigma_A(v, w)} = (-A_\alpha^* w, v)$. Dann ist τ ebenso stetig und koerziv wie σ_A und wir erhalten Bedingung iii durch analoges Vorgehen.

Die bisherigen Ergebnisse zusammenfassend, erhalten wir damit

Lemma 2.3.3:

Für die gewählte Approximation gelte (C1). Außerdem seien B^N, Q^N, $T^N(t)$, sowie $T^N(t)^*$ definiert wie in und direkt vor (2.30). Dann gelten auch die Konvergenzbedingungen ii.)-v.) aus Abschnitt 2.1.

Um die Sätze 2.1.7 und 2.1.8 anwenden zu können, benötigen wir eine Bedingung unter der eine zulässige Steuerung zu jedem Anfangswert existiert. Dies sichern wir durch die Annahme (C2), daß das Paar (A, B) exponentiell stabilisierbar ist.

Die wichtigste Bedingung für die Konvergenz der endlich dimensionalen Probleme bezeichnen Banks und Kunisch als *preservation of exponential stability under approximation condition*, kurz (POES). Sie besagt, daß es ein $N_0 \in \mathbb{N}$ gibt, so das für alle $N \geq N_0$ die Paare $(A^N, P^N B)$, durch den Operator K zu (C2) aus Definition 2.3.1 gleichmäßig exponentiell stabilisierbar sind. Banks und Kunisch bewiesen in [2] Lemma 3.3, daß bei Gültigkeit der Annahmen (C1) und (C2) die Approximationsbedingung (POES) für die durch (2.29) und (2.30) gegebenen Approximationen erfüllt ist.

Seien also (C1) und (C2) erfüllt. Dann liefert (POES) mit Satz 2.1.7 die Existenz von nichtnegativen selbstadjungierten Riccati Operatoren Π und Π^N (für N groß genug) zu (2.26) bzw. seinen endlich dimensionalen Approximationen (vgl. (2.8)).

Sicherung der Voraussetzungen von Satz 2.1.8

Wir wollen nun einsehen, daß die Voraussetzungen aus Satz 2.1.8 im vorliegenden Problem erfüllt sind. Wir betrachten zunächst Gleichung (2.14). Nach (2.12) gilt:

$$\|\Pi^N\|_{H^N} = \sup_{z \in H^N,\ \|z\|=1} (H^N z, z) = \sup_{\|z\|=1} \mathcal{J}^N(z, u_*^N) \tag{2.38}$$

Seien nun $T^N(t)$ die Operatorhalbgruppen aus Definition 2.3.1, dann gilt nach Satz 2.1.7

$$\begin{aligned} q_*^N(t) &= -R^{-1}B^{N^*}\Pi^N u_*^N(t) \\ u_*^N(t) &= T^N(t)u_0^N \end{aligned} \tag{2.39}$$

Zu beliebigem $v \in H$ mit $\|v\| = 1$ definieren wir die Steuerung $q^N(t) := KT^N(t)v$.

Dann gilt:

$$\begin{aligned} \mathcal{J}^N(v, q_*^N) &\leq \mathcal{J}^N(v, q^N) \\ &= \int_0^\infty (Q^N T^N(t)v, T^N(t)v) + (RKq^N(t), Kq^N(t))dt \\ &= \int_0^\infty (Q^N T^N(t)v, T^N(t)v) + (RKT^N(t)v, KT^N(t)v)dt \\ &\leq (\|Q^N\| + \|R\|\|K\|^2)M^2\omega^2 =: M_2 \end{aligned} \tag{2.40}$$

Mit (2.38) folgt also das gewünschte Ergebnis.

Wir definieren zu den $S^N(t)$ aus Satz 2.1.8 die Generatoren G^N durch

$$G^N := A^N - P^N BR^{-1}B^*P^N\Pi^N.$$

Da $P^N BR^{-1}B^*P^N\Pi^N \geq 0$ gilt nach (2.27) ebenfalls

$$\mathrm{Re}(-G^N u, u) \geq c_1\|u\|_{1,2}^2.$$

Wenden wir nun Satz 2.1.5 auf G an, so erhalten wir sofort:

$$\|S(t)\|_{H^N} \leq e^{-c_1 t} \text{ unabhängig von } N \tag{2.41}$$

Und damit liefert uns Satz 2.1.8 die Konvergenzen in (2.15) und (2.16). Diese ergeben nun mit $P^N v \to v$ für $N \to \infty$ die Konvergenz der Kontrollen zu den approximierenden Problemen gegen die Kontrolle zum betrachteten unendlich dimensionalen Kontrollproblem dieser Arbeit. Fassen wir also zusammen:

Das Finite–Elemente–Verfahren erfüllt die Approximationbedingung (C1). Das betrachtete PDE–Problem erfüllt die Stabilisierbarkeitsbedingung (C2). Wählen wir nun Q, R so, daß die Definitheitsanforderungen $Q > 0$ und $R \geq 0$ nicht verletzt werden, dann existieren eindeutige Riccati–Operatoren Π und Π^N zum betrachteten Kontrollproblem, bzw. seinen endlich dimensionalen Approximationen und es gelten die Konvergenzen

$$\Pi^N P^N v \to \Pi v \qquad \text{für } v \in L^2(\Omega), \tag{2.42}$$

$$S^N(t)P^N v \to S(t)v \qquad \text{für } v \in L^2(\Omega) \text{ und} \tag{2.43}$$

$$q_*^N(t) \to q_*(t). \tag{2.44}$$

Die letzten beiden Konvergenzen gelten lokal gleichmäßig in t auf $[0, \infty)$ und $S^N(t)$, bzw. $S(t)$ sind die von $A^N - P^N BR^{-1}B^*P^N\Pi^N$, respektive $A - BR^{-1}B^*\Pi$ generierten Halbgruppen. Die eindeutigen optimalen Feedbacksteuerungen sind gegeben durch $q_*^N(t) = -R^{-1}B^*P^N\Pi^N S^N(t)P^N u_0$ bzw. $q_*(t) = -R^{-1}B^*\Pi S(t)u_0$ und überdies gilt

$$\|S(t)\| \leq e^{-c_1 t},$$

wobei $c_1 > 0$ die Koerzivitätskonstante zur Sesquilinearform aus der schwachen Formulierung der PDE ist.

Kapitel 3

Implementierung mit ALBERT und LyaPack

Dieses Kapitel gibt in kurzen Einführungen wieder, was die beiden Programmpakete ALBERT und LyaPack sind und wie sie arbeiten. Nach der jeweiligen Einführung wird dann erläutert, wie das entsprechende Teilproblem damit implementiert ist. Außerdem wird in den späteren Abschnitten erläutert, wie sich die Teilprobleme zusammenfügen. Dabei wurden insbesondere zwei grundsätzlich unterschiedliche Herangehensweisen verfolgt, die zunächst in einem einführenden Abschnitt erläutert werden sollen.

3.1 Einführung in die beiden Herangehensweisen

Im einfachen Fall des ODE-Ansatzes wurde zunächst in ALBERT eine Anlaufrechnung bis zu einer Starttemperaturverteilung implementiert. Diese gibt dann ihre Daten an MATLAB und LyaPack Routinen ab, um später die Lösung der gewöhnlichen Differentialgleichung mit Feedbacksteuerung zurückzuerhalten und mit entsprechenden Gitterdaten zur Visualisierung in Dateien zu speichern. Der PDE–Zugang gibt die Daten nach derselben Anlaufrechnung an das LyaPack, um von dort einen neuen Satz von Steuerparametern für die nächsten Zeitschritte zu erhalten und nach der Berechnung dieser Zeitschritte einen neuen Satz von Steuerparametern vom LyaPack anzufordern.

In den Begriffen und Methoden der Numerik partieller Differentialgleichungen ausgedrückt heißt das, daß der ODE–Ansatz eine Semidiskretisierung nach der (vertikalen) Linienmethode verwendet, während der PDE–Ansatz eine Rothe–Methode, oder auch horizontale Linienmethode benutzt.

3.1.1 ODE–Zugang

Der ODE–Zugang ist von den beiden in dieser Arbeit implementierten Ansätzen der klassische. Es wird zu dem gegebenen Regelungssystem die Lösung mittels Zustands-

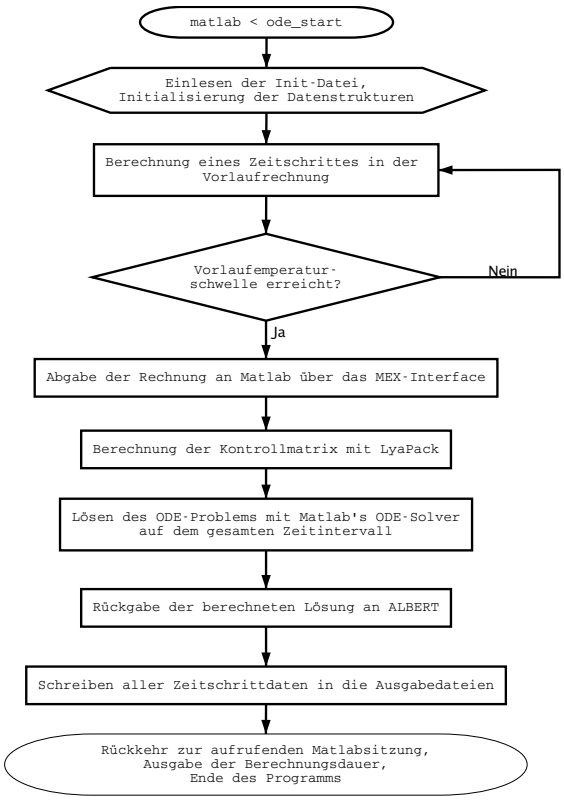


Abbildung 3.1: Flußdiagramm zum ODE–Zugang

rückführung[1] gesucht. Der ODE–Zugang verwendet dabei ALBERT um die Systemmatrizen aufzustellen und rechnet mit diesen die Feedbackmatrix K aus[2], um dann mittels eines MATLAB–ODE–Lösers die Gleichung

$$\dot{\tilde{x}} = -\tilde{N} - K\tilde{B}^T\tilde{x} \tag{3.1}$$

zu lösen. Dabei wird als Anfangswert x_0 die Lösung der Anlaufrechnung in ALBERT verwendet. Die Vorlaufrechnung in ALBERT zieht sich dabei über mehrere Zeitschritte hin, da hier eine Abkühlung von konstanten 1000°C, auf eine Maximaltemperatur von unter 990°C mit der, dem Problem zugrundeliegenden Wärmeleitungsgleichung gerechnet wird. Obwohl ALBERT prinzipiell die Möglichkeit bietet im Ort adaptiv zu rechnen, also das Rechengitter und damit die Systemmatrix an die Temperaturverteilung anzupaßen, werden in dieser Arbeit nur Rechnungen auf festen global verfeinerten Gittern vorgestellt. Der Grund dafür liegt darin, daß der in ALBERT mitgelieferte Fehlerschätzer für Wärmeleitungsgleichungen mit den hier verwendeten Randbedingungen nur unzureichend arbeitet. Ein problemangepaßter Fehlerschätzer für Wärmeleitungsgleichungen mit den vorliegenden Randbedingungen wurde im Rahmen dieser Arbeit aus Zeitgründen nicht mehr implementiert. Nach dem Lösen von (3.1) erhält ALBERT die Daten wieder zur Endverarbeitung zurück. Algorithmisch kann der Prozess also wie folgt beschrieben werden.

[1] oder auch Feedbacksteuerung
[2] vgl. auch Anhang A sowie Kapitel 2

Algorithmus 3.1.1 (ODE–Zugang):

1. Initialisierung der Parameter
2. Beginn der Vorlaufrechnung in ALBERT
3. Sobald $\max_{x_i \in \Omega_h} u_h < u_{init}$ erreicht:
4. Senden der Daten an MATLAB.
5. Transformation der Gleichung in LyaPack-Form.
6. Berechnung der Kontrollmatrix K.
7. Lösen von (3.1) auf $[0, t_{end}]$.
8. Rücksendung der berechneten Daten an ALBERT.
9. Ausgabe der Lösung in für `albert_movi` (GraPE) lesbare Dateien.

Wie dabei die einzelnen Schritte dieses Algorithmus arbeiten, wird in den weiteren Abschnitten dieses Kapitels detailierter erläutert.

3.1.2 PDE–Zugang

Der PDE–Zugang stellt die wesentliche neue Idee in dieser Arbeit dar. Während der ODE–Zugang zunächst eine Ortsdiskretisierung macht und dann auf dem entstandenen Gitter eine gewöhnliche Differentialgleichung löst, wird hier zunächst in der Zeit diskretisiert. In den einzelnen Zeitschritten ist dann jeweils nur noch eine elliptische partielle Differentialgleichung zu lösen. Die Steuerung wird hier in Form der Steuerparameter in den Randbedingungen eingebracht, d.h. zu Beginn wird ein Satz von Steuerparametern vorgegeben. Dies können z.B. die Grundeinstellungen der Kühlmaschinerie nach dem Einschalten derselbigen sein. Diese werden dann im Verlauf der Rechnung immer wieder an die derzeit vorliegende Wärmeverteilung angepaßt. Wir erhalten damit insbesondere die Möglichkeit, die Fähigkeit von ALBERT auszunutzen, den Ort adaptiv zu diskretisieren. Somit ist es im Gegensatz zum ODE–Ansatz auch während der Rechnung möglich, das Gitter weiter an die aktuell vorliegende Temperaturverteilung und ihre kritischen Punkte anzupassen. Aus den bereits beim ODE–Ansatz diskutierten Gründen, ist dies aber in der vorliegenden Implementierung noch nicht umgesetzt. Im PDE–Zugang wird die im Abschnitt 3.1.1 beschriebene Anlaufrechnung ebenfalls gemacht. Damit liegen in beiden Ansätzen die gleichen Startvoraussetzungen vor. Allerdings läuft hier die Berechnung der Lösung komplett in ALBERT, respektive C ab. MATLAB, bzw. LyaPack, wird in diesem Ansatz zum Starten und zur Berechnung der Steuerparameter benutzt. Der Benutzer kann bei der Initialisierung angeben wie häufig[3] die Steuerparameter neu berechnet werden sollen. Das Programm überprüft dann je-

[3] „wie häufig“ ist an dieser Stelle so zu verstehen, das der Benutzer die Möglichkeit hat anzugeben, nach wievielen Zeitschritten jeweils ein neuer Satz von Steuerparametern berechnet werden soll. Die zwischen den Aufdatierungen der Parameter verstrichene Zeit kann dabei sehr unterschiedlich sein, da ALBERT die Zeitschrittweite ggf. adaptiv berechnet

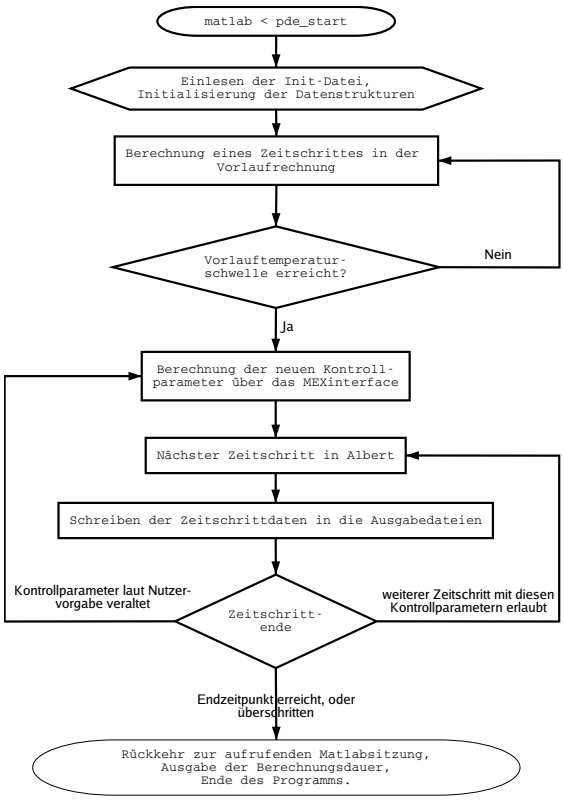


Abbildung 3.2: Flußdiagramm zum PDE–Zugang

weils am Ende eines Zeitschrittes, ob wieder neue Steuerparameter berechnet werden sollen. Ist dies der Fall, wird das LyaPack aufgerufen und ein neuer Satz von Steuerparametern berechnet, die dann wieder für die folgenden Zeitschritte die Randbedingungen bestimmen. Für die Berechnung der neuen Steuerparameter wird jeweils eine Riccati–Gleichung mit den aktuellen Systemmatrizen gelöst, um die neue Feedbackmatrix zu berechnen. Das macht natürlich eigentlich nur dann Sinn, wenn sich die beteiligten Matrizen geändert haben. Dies ist immer dann der Fall, wenn sich auch das unterliegende Gitter geändert hat, oder die Zeitschrittweite verändert wurde. Bleibt das Gitter erhalten, so wird bei fester Zeitschrittweite jedes Mal die gleiche Feedbackmatrix berechnet. Daher ist dieser Ansatz nur dann wirklich effizient, wenn ALBERT adaptiv[4] rechnet. Da sich mindestens die Zeitadaption für die normierte Gleichung anbietet[5], stellt dies aber im vorliegenden Fall kein Problem dar. Algorithmus 3.1.2 zeigt die Arbeitsweise dieses Zugangs nocheinmal in algorithmischer Form auf.

[4] im Ort oder in der Zeit

[5] bei der Normierung der Temperatur von 1000°C auf 1 wird der Faktor 1000 auf die Zeitskala umgelegt. Eine Rechnung für eine Dauer von 45 Sekunden wird also auf dem Interval $[0, 45000]$ stattfinden.

Algorithmus 3.1.2 (PDE–Zugang):

1. Initialisierung der Parameter.
2. Beginn der Vorlaufrechnung in ALBERT.
3. Sobald $\max_{x_i \in \Omega_h} u_h < u_{init}$ erreicht:
4. Wiederholung der Sequenz
5. Senden der Daten an MATLAB.
6. Transformation der Gleichung in LyaPack-Form.
7. Berechnung der Kontrollmatrix K.
8. Berechnung der neuen Steuerparameter.
9. Rücksenden der neuen Steuerparameter an ALBERT.
10. Berechnung der nächsten Zeitschritte bis neue Steuerparameter erforderlich.
11. Solange $t < t_{end}$.

Im Gegensatz zum ODE–Zugang werden hier die Daten bereits am Ende jedes Zeitschrittes auf den Datenträger geschrieben, was zusätzlich eine Beobachtung des Fortschrittes bereits während der Rechnung ermöglicht.

3.2 ALBERT

3.2.1 Einführung in ALBERT

Für ausführliche Informationen über ALBERT wird auf [23] und [24] verwiesen. An dieser Stelle sollen lediglich die Arbeitsweise und die Möglichkeiten kurz skizziert werden, um aufzuzeigen, wie die Bibliothek im vorliegenden Fall angewendet werden kann.

ALBERT ist die Implementierung einer adaptiven hierarchischen Finite–Elemente Toolbox. Entwickelt wurde ALBERT von Alfred Schmidt [6] und Kunibert G. Siebert [7] am Institut für Angewandte Mathematik an der Albert–Ludwigs–Universität Freiburg. Die Stärke der Implementierung liegt in ihrem hohen Abstraktionslevel. Dieses erlaubt es sehr einfach, auch eigene Probleme in die Bibliothek einzupflegen, indem man z.B. eigene Finite–Elemente implementiert[8], eigene Löser entwickelt, oder angepasste Fehlerschätzer für die eigenen Probleme vorgibt. Dabei liefert ALBERT eine sehr kompakte und dennoch äußerst flexible Grundimplementation und einige typische Beispiele. In der vorliegenden Implementierung verwendet ALBERT Bisektionsverfeinerung und

[6] schmidt@math.uni-bremen.de
[7] kunibert@mathematik.uni-freiburg.de
[8] siehe z.B. [14]

Residualfehlerschätzer. Es stehen polynomiale Lagrangeelemente zur Verfügung und im Fall der linearen Lagrangeelemente gibt es ebenfalls eine Implementierung hierarchischer Basen.

Der Ablauf eines Standardprogramms für instationäre Probleme in ALBERT läßt sich wie folgt zusammenfassen. Zunächst werden die Parameter (wie z.B. maximale Iterationszahlen, Anzahl der initialen Verfeinerungen des Makrogitters, die Koeffizienten für die Fehlerschätzer, Start– und Endzeitpunkt, . . .) und die Makrotriangulierung aus Dateien eingelesen und die Datenstruktur für das Gitter sowie die Struktur mit den Adaptionsdaten initialisiert und mit den eingelesenen Werten, bzw. mit Standardwerten gefüllt. Die Adaptionsdaten umfassen dabei sowohl echte Daten wie Start– und Endzeitpunkt, als auch Verweise auf die zu verwendenden Funktionen zum Lösen der Gleichungssysteme, zum Aufstellen der Systemmatrix usw. Außerdem werden die Datenstrukturen für die Systemmatrix, die rechte Seite und die Lösung vorbereitet. Nachdem die Initialisierung beendet ist, übernimmt die Funktion `adapt_method_instat()` die Arbeit. Diese ruft zunächst im 0–ten Zeitschritt die Funktion `adapt_method_stat()` für stationäre Probleme auf. Damit wird nun das zugehörige stationäre Problem gelöst. Dabei werden die in der Init–Datei vorgegebenen Daten für den Initialschritt (Anzahl der Verfeinerungen, Adaptionskoeffizienten, . . .) verwendet. Danach startet in der Funktion `adapt_method_instat()` eine Schleife, die solange läuft, bis der aktuelle Zeitpunkt größer oder gleich dem vorgegebenen Endzeitpunkt ist. In dieser Schleife werden nacheinander die Funktionen `init_timestep(), one_timestep()` und `close_timestep()` aufgerufen.

`init_timestep()` übernimmt die Sicherung der Lösung aus dem letzten Zeitschritt, die für die Zeitdiskretisierung benötigt wird, sowie alle weiteren zur Initialisierung eines Zeitschrittes nötigen Vorgänge. In `one_timestep()` werden die Routinen zum Aufstellen und Lösen des Gleichungssystems für den aktuellen Zeitschritt aufgerufen und damit die neue Lösung berechnet. Desweiteren wird hier ggf. die Fehlerschätzung vorgenommen. Die Funktion`close_timestep()` wird nun verwendet um die Daten auszugeben, d.h. zur späteren Anzeige mit den GraPE–Tools `albert_movi`, bzw. `albert_grape` in Dateien zu speichern, oder sofort mittels der integrierten GLTools–Grafikroutinen zu visualisieren. Außerdem können hier Fortschrittsinformationen, bzw. Daten über die geschätzten Fehler ausgegeben werden. Natürlich ist es möglich alle drei Funktionen selbst vorzugeben werden. Ist dies bei `one_timestep()` nicht der Fall, so verwendet ALBERT eine Standard–Bibliotheksfunktion. Die beiden anderen Funktionen bleiben leer, falls sie nicht vom Benutzer implementiert werden.

3.2.2 Die Implementierung des Wärmeleitungsproblems in ALBERT

Den Startpunkt für die Implementierung stellte die als Beispiel vorliegende Implementierung eines Lösers zur instationären Wärmeleitungsgleichung dar. Diese rechnet das Problem

$$\begin{aligned} \frac{\partial u(t)}{\partial t} &= \Delta u(t,x) && x \in \Omega,\ t \in (0,T], \\ u(0,x) &= u_0(x) && x \in \Omega, \\ \frac{\partial u}{\partial \nu} &= 0 && t \in (0,T]. \end{aligned}$$

Das Gebiet Ω wird dabei aus einer Makrotriangulierungsdatei eingelesen. Es mußten also folgende Punkte implementiert werden:

1. Anpassung des Operators; d.h Einbau geeigneter Datenstrukturen zur Beschreibung des Operators unter Berücksichtigung der Materialkonstanten aus Gleichung (1.2),
2. Vorbereitung eines Makrogitters für das Gleisprofil und Bereitstellung dieses Gitters als Makrotriangulierungsdatei,
3. Erstellung einer Funktion für die Zuweisung der Randwerte zu den einzelnen Randstücken,
4. Vorgabe der Projektionsfunktionen, welche die neu eingefügten Randknoten ggf. auf die parametrisierten Ränder (Kreissegmente) verschieben,
5. Erschaffung einer Funktion, die die Randbedingungen implementiert.

Operator und Makrogitter

Die Anpassung des Operators ließ sich relativ leicht umsetzen, da es in der Funktion `assemble()`, die die Systemmatrix in jedem Zeitschritt aufstellt, bereits eine Struktur `op_info` mit Daten über den Operator gibt, welche um die Felder für Wärmekapazität, Wärmeleitfähigkeit und Dichte des Stahls ergänzt werden mußte. Diese wurden in der angepaßten Funktion `LALt()` verwendet, welche die Aufstellung der Elementmatrixen implementiert, um den Operator korrekt wiederzugeben.

Das Makrogitter wurde mit Hilfe des MATLAB–PDETools erzeugt. Die damit erzeugten Daten wandelt ein zu diesem Zweck implementiertes MATLAB–Skript in das von ALBERT für Makrogitter verwendete Format um. Auf diese Weise lassen sich leicht auch andere Profilquerschnitte erzeugen und für die Verwendung in ALBERT vorbereiten, da das Umwandlungsskript strikt auf den Datenstrukturen agiert und keine weiteren Kenntnisse über das Rechengebiet verwendet. Einzig die Randnummern, anhand derer ALBERT die einzelnen Randstücke identifiziert und anhand deren Vorzeichen es standardmäßig die Randbedingungen zuordnet, müssen ggf. noch per Hand korrigiert werden.

Behandlung des Randes

Auswahl der Randstücke und parametrisierter Rand Die Zuweisung der Randwerte zu den einzelnen Randstücken übernimmt in ALBERT die Funktion `ibdry()`. Diese liefert zu einer Randnummer ein Tupel aus einer Projektionsfunktion, für die Projektion neuer Knoten auf den parametrisierten Rand (bzw. einen Nullpointer falls dies nicht nötig ist) und einer Konstante, die angibt um welche "Art" Rand es sich hier handelt. Damit kann die Bisektionsroutine entscheiden, ob ein neu eingefügter Randpunkt auf eine Randkurve projiziert werden muß. Die Randbedingungsroutine kann entscheiden, welche Randbedingung für dieses Randstück zu verwenden ist. Ist die übergebene

Randnummer negativ, oder gleich 0, so interpretiert die Standard `ibdry()`-Funktion aus der Bibliothek diesen Rand als Neumann–Null–Rand. Ist der Wert größer als 0, so wird das Randstück als Dirichlet–Rand interpretiert. Diese Routine wurde durch eine eigene ersetzt, welche die negativen Übergabeparameter als Robin–Rand interpretiert und im Spezialfall des linken Randes die Konstante `NEUMANN` zurückgibt. Durch einen entsprechenden Verweis in der `adapt_instat`–Struktur wurde festgelegt, daß ALBERT diese Funktion anstelle der Standardfunktion verwendet.

Für die Projektion der neuen Randknoten auf die krummen Ränder wurden einige Projektionsroutinen implementiert, die diese, bei der Gitterverfeinerung, auf die passenden Kreissegmente verschieben. Wie diese genau aussehen und arbeiten, wird im Anhang C näher erläutert.

Implementierung der Randbedingungen Für die Behandlung der Randwerte wurden verschiedene Funktionen entwickelt, die unterschiedliche Formulierungen von Randbedingungen abdecken.

Um zu verstehen, wie die Funktionen zu den Randbedingungen arbeiten, betrachten wir zunächst die Finite–Elemente–Diskretisierung (FEM–Diskretisierung) der Gleichung (1.2), wie wir sie in Abschnitt 2.2 hergeleitet haben.

Zunächst gibt es natürlich eine Funktion, die die Randbedingung aus Gleichung (1.2) implementiert. Diese nutzt `ibdry`, um zu entscheiden, welcher der Steuerparameter zur Anwendung kommt. Liefert `ibdry` die Konstante `NEUMANN` so beendet sich diese Funktion sofort, um die Neumann–Null–Randwerte zu implementieren. Anderenfalls werden die Elementintegrale ausgewertet, die Elementmatrizen berechnet und die Systemmatrix sowie die rechte Seite aufdatiert. Neben dieser Funktion gibt es noch eine weitere, die die in der Anlaufrechnung und in der Referenzrechnung verwendete Randbedingung gemäß

$$\partial_\nu u = \frac{q_i}{\lambda}(u - u_{ext})$$

implementiert. Hier wird die Neumann–Randbedingung durch die Setzung des Steuerparameters auf 0 wiedergegeben. Eine analoge Rechnung wie oben liefert für diese Formulierung das Differentialgleichungssystem

$$M\ddot{u} + Su = Bq - M_\alpha u,$$

wobei B hier von u und von u_{ext} abhängt. Daher arbeitet diese Funktion bis auf die Konstanten wie die obige. Die Funktionen werden jeweils nach dem Aufstellen der Systemmatrix von der Funktion `assemble()` aufgerufen, um die Systemmatrix und die rechte Seite aufzudatieren.

3.3 LyaPack

3.3.1 Was ist das LyaPack?

Das LyaPack ist eine MATLAB–Toolbox für hochdimensionale Lyapunov– und Riccati–Gleichungen, sowie Modell–Reduktionsprobleme und linear–quadratische Optimalsteuerungsprobleme. Genaue Informationen zur Benutzung und Anwendbarkeit des LyaPack finden sich in [22]. Wie bereits für die ALBERT Bibliothek im Abschnitt 3.2.1, soll auch hier ein kurzer überblick über den Funktionsumfang und die Arbeitsweise gegeben werden. Das LyaPack (Akronym für „Lyapunov Package“) ist geeignet, drei Klassen von Problemen zu bearbeiten.

- Lösung von Lyapunov–Gleichungen
- Modellreduktion
- Lösung von Riccati-Gleichungen und linear–quadratischen Optimalsteuerungsproblemen

Generell gilt dabei, daß sich das LyaPack an Benutzer richtet, die mit Problemgrößen etwa ab Dimension 500 arbeiten müssen. Die enthaltenen Routinen und die zugrundeliegenden Algorithmen sind dabei ebenso effizient in Hinsicht auf Speichernutzung, wie auch CPU–Auslastung ausgelegt. Dabei sind einige Voraussetzungen an die zu lösenden Probleme gestellt, wie z.B.:

- Die Systemmatrix muß stabil sein, oder es muß ein stabilisierendes Initialfeedback vorgegeben werden.
- Die Massenmatrix muß regulär sein.
- Effizienz wird nur gewährleistet, wenn die Anzahl der Ein– und Ausgänge sehr viel kleiner ist, als die Dimension des Problems

Da das LyaPack auf iterativen Methoden aufbaut, ist es besonders anfällig bei schlecht konditionierten Problemen.

Nach einigen Bemerkungen zum Aufbau und Design des LyaPack werden wir uns im folgenden auf die Betrachtung der Methoden zur Lösung der linear–quadratischen Optimierungsprobleme beschränken, da diese für das vorliegende Problem relevant sind.

Aufbau und Design

LyaPack ist seitens der Routinen wie folgt organisiert. Es gibt zunächst einige Systemfunktionen[9], die das Kernstück des Paketes bilden. Da im LyaPack großer Wert darauf

[9] im Unterverzeichnis `routines`

gelegt wird, daß man am Namen einer Funktion bereits ihren Zweck und die Zugehörigkeit erkennen kann, beginnen diese Funktionen sämtlich mit `lp_`. Unter anderem gibt es hier Routinen für die low rank ADI–Methode[10], Hilfsfunktionen für die Berechnung der ADI–Shift Parameter, die Auswertung der involvierten Minimax–Probleme usw. Den zweiten großen Block von Funktionen bilden die user–supplied–functions[11]. Diese sind Funktionen für die Matrixmultiplikationen und das Lösen der linearen Gleichungssysteme, die vom Benutzer bereitgestellt werden um bestmögliche Ausnutzung der problembezogenen Struktur der Matrizen zu gewährleisten. Diese sind so organisiert, daß jede Routine einen Basisnamen besitzt, der anzeigt, zu welchem Problem sie gehört. Außerdem erhält jede Routine eine Endung, die anzeigt, welches Teilproblem sie dabei löst. So gibt es Funktionen, die die Daten für eine Matrixmultiplikation vorbereiten, also beispielsweise Matrixzerlegungen, die häufig benötigt werden, einmal berechnen und als globale Daten ablegen. Dazu gibt es dann noch weitere Funktionen, die die Matrixprodukte selbst ausrechnen und wieder andere Funktionen, die Daten wieder aus dem globalen Speicher entfernen, um die effiziente Speichernutzung zu gewährleisten. Das Paket stellt dabei bereits einige Standard user–supplied–functions für symmetrische, unsymmetrische und Systeme mit Massematrix bereit. Diese sind für typische Anwendungen meist schon ausreichend und können andererseits als gute Vorlage, bzw. zur Verdeutlichung des Konzeptes dienen, das hinter diesen Funktionen steckt.

Arbeitsweise

Die betrachteten Probleme sind von der Form: Minimiere das Kostenfunktional

$$\mathcal{J}(u, y, x_0) = \frac{1}{2} \int_0^\infty y(t)^T Q y(t) + u(t)^T R u(t) dt \tag{3.2}$$

unter der Bedingung, daß y der Ausgang und u der Eingang des dynamischen Systems

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) \end{aligned} \tag{3.3}$$

sind. LyaPack läßt aber auch die Verallgemeinerung

$$\begin{aligned} M\dot{\tilde{x}}(t) &= N\tilde{x}(t) + \tilde{B}u(t), \\ y(t) &= \tilde{C}\tilde{x}(t) \end{aligned} \tag{3.4}$$

zu und stellt, wie bereits geschildert, sogar die nötigen user–supplied–functions für solche Probleme zur Verfügung. Dabei sind $A, M, N \in \mathbb{R}^{n,n}$, $B, \tilde{B} \in \mathbb{R}^{n,m}$, $C, \tilde{C} \in \mathbb{R}^{q,n}$, $Q \in \mathbb{R}^{q,q}$, $R \in \mathbb{R}^{m,m}$ und $t \in \mathbb{R}$. Zur Betrachtung der Arbeitsweise wird stellvertretend das Demo–Skript zu linear–quadratischen Optimierungsproblemen vorgestellt[12]. Das Skript verwendet als Beispielproblem eine Finite–Differenzen–Diskretisierung der Poisson–Gleichung. Zunächst wird das Problem initialisiert.

[10] vergleiche dazu auch Abschnitt B.2 im Anhang
[11] im Unterverzeichnis `usfs`
[12] aus dem Unterverzeichnis `demos`

```
n0 = 50;   % n0 = number of grid points in either space direction;
           % n = n0^2 is the problem dimension!
           % (Change n0 to generate problems of different size.)

A = fdm_2d_matrix(n0,'0','0','0');
B = fdm_2d_vector(n0,'.1<x<=.3');
C = (fdm_2d_vector(n0,'.7<x<=.9'))';

Q0 = 10   % Q = Q0*Q0' = 100
R0 = 1   % R = R0*R0' = 1

K_in = [];   % Initial feedback K is zero (Note that A is stable).

disp('Problem dimensions:')

n = size(A,1)   % problem order (number of states)
m = size(B,2)   % number of inputs
q = size(C,1)   % number of outputs
```

Die Funktionen `fdm_*` stellen die Finite–Differenzen–Diskretisierung zur Verfügung. Zu beachten ist hier, daß mit einem leeren initial–Feedback begonnen werden kann, da die Systemmatrix stabil ist. Der folgende Abschnitt soll verdeutlichen, wie die user–supplied–functions verwendet werden. Zunächst wird in `name` der Basisname gesetzt, der angibt, welche Funktionenklasse verwendet werden soll. Im Beispiel werden die Funktionen für unsymmetrische Matrizen verwendet, obwohl die Systemmatrix eigentlich symmetrisch ist. Das ist nötig, da im weiteren Verlauf geshiftete Systeme berechnet werden, bei denen die Symmetrie nicht mehr sichergestellt werden kann. Nach diese Setzung werden die vorbereitenden Routinen aufgerufen.

```
name = 'au';

[A0,B0,C0,prm,iprm] = au_pre(A,B,C);   % preprocessing (reordering
                                       % for bandwidth reduction)

% Note that K_in is zero.
% Otherwise it needs not be transformed as well.

au_m_i(A0);   % initialization for matrix multiplications with A0

au_l_i;   % initialization for solving systems with A0 (This is
          % needed in the Arnoldi algorithm w.r.t. inv(A0).
          % The Arnoldi algorithm is part of the algorithm in
          % 'lp_para', which in turn will be invoked in each Newton
          % step in the routine 'lp_lrnm'.)

% Note that 'au_s_i' will be invoked repeatedly in 'lp_lrnm'.

```

```
disp(...
'Parameters for heuristic algorithm which computes ADI parameters:')
l0 = 15   % desired number of distinct shift parameters
kp = 50   % number of steps of Arnoldi process w.r.t. A0-B0*K0'
km = 25   % number of steps of Arnoldi process w.r.t. inv(A0-B0*K0')
```

Die letzten Zeilen geben die Parameter für die Berechnung der Shift–Parameter an. Danach werden noch die diversen Parameter für die Newtonmethode zur Berechnung des Choleskyfaktors `Z0` der Lösung X der Riccati–Gleichung

$$C^T QC + A^T X + XA - XBR^{-1}B^T X = 0$$

gesetzt und schließlich der entscheidende Aufruf der low–rank–Newtonmethode abgesetzt.

```
[Z0, flag_r, res_r, flp_r, flag_l, its_l, res_l, flp_l] = lp_lrnm(...
zk, rc, name, B0, C0, Q0, R0, K_in, max_it_r, min_res_r, with_rs_r,...
min_ck_r, with_ks_r, info_r, kp, km, l0, max_it_l, min_res_l,...
with_rs_l, min_in_l, info_l );
```

Hier sollte besonders beachtet werden, daß einer der Parameter die Variable `name` ist, über welche die intern verwendeten Subroutinen erfahren, welchen Satz von user–supplied-functions sie verwenden sollen. Im weiteren Verlauf berechnet das Skript auch noch direkt die Feedbackmatrix `K0` und vergleicht die Ergebnisse. Dafür ist nur ein Aufruf der Routine `lp_lrnm()` mit leicht abgewandelten Parametern nötig. Im folgenden Block werden die entsprechenden user–supplied-functions zum Beseitigen der nicht mehr benötigten globalen Daten aufgerufen und die Ergebnisse auf das Ausgangsproblem rücktransformiert.

```
Z = au_pst(Z0,iprm);
K = au_pst(K0,iprm);

% Note that 'au_s_d' has already been invoked in 'lp_lrnm'.

au_l_d;   % clear global variables initialized by au_l_i
au_m_d;   % clear global variables initialized by au_m_i
```

Die Möglichkeit, die Feedbackmatrix direkt zu berechnen, wurde bei der Implementierung des vorliegenden Problems ausgenutzt, da gerade diese von Interesse und die implizite Berechnung effizienter ist, als zunächst die Cholesky–Faktoren zu X zu berechnen und damit die Feedbackmatrix aufzustellen.

3.3.2 Implementierung des Steuerungsproblems mit dem LyaPack

Die Basis zur Implementierung des Steuerungsproblems liefert bereits der oben angesprochene zweite Teil des Skriptes `demo_r1.m` aus dem LyaPack. Für die Umsetzung wird

lediglich der Initialisierungsteil durch einen MATLAB–Funktionenheader ersetzt, der die benötigten Matrizen `M, N, B` und `C` aus Übergabeparametern bereitstellt. Die Parameter `Q0, R0, l0, kp` und `km` werden weiterhin direkt hier gesetzt. Wir halten an dieser Stelle fest, daß die Steifigkeitsmatrix $S + M_\Gamma$ aus der Finite–Elemente–Diskretisierung in Gleichung (2.21) positiv definit ist. In der Formulierung des Kontrollproblems gemäß (2.3) steht sie aber mit negativem Vorzeichen auf der rechten Seite der Gleichung, ist also insbesondere stabil. Wir können folglich analog zum obigen Beispiel das initial–Feedback auf 0 setzen.

Außerdem wird am Ende der Funktion der Steuerungsvektor $q = (q_1, \ldots, q_7)$ aus (1.2) ausgerechnet. Damit haben wir eine MATLAB–Funktion, die aus den Systemmatrizen und der aktuellen Lösung die neuen Steuerparameter berechnet. Diese wird im PDE–Zugang verwendet. Eine leicht abgewandelte Version, die nicht den Steuervektor berechnet, dafür aber den ODE–Löser aufruft, arbeitet im ODE–Zugang.

3.4 Zusammenspiel von ALBERT und LyaPack

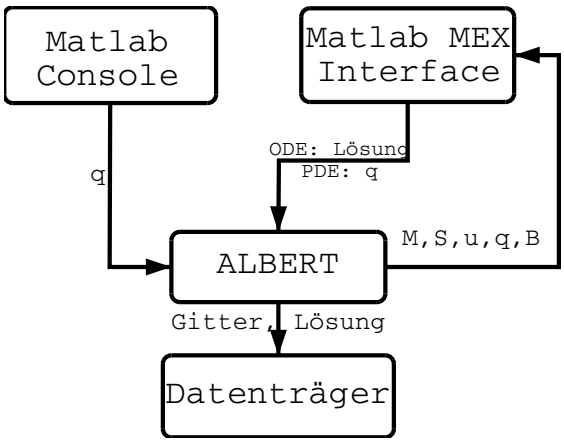


Abbildung 3.3: Schematische Darstellung des Datenflußes

Die Abbildungen 3.1 und 3.2 zeigen die Programmabläufe in den beiden Zugängen als Flußdiagramme. Dort wird unter anderem auch anschaulich gemacht, wann und wie die Pakete die Arbeit aneinander abgeben. Die Hauptschwierigkeit bei der Verknüpfung der beiden Pakete bestand darin, das ALBERT in C und das LyaPack in MATLAB implementiert wurden. Es mußte also irgendeine Verbindung zwischen beiden geschaffen werden. Es ist wegen des starken Skriptcharakters des LyaPack nicht möglich, dieses mittels des MATLAB–Compilers in eine von MATLAB unabhängige C–Bibliothek zu übersetzen, weil hier die interpretersprachenspezifische Funktion `eval` zur Auswertung eines Strings als Kommandozeile im Interpreter verwendet wird, welche der Compiler in der vorliegenden Version nicht in gültigen C–Quellcode konvertieren kann. Daher wurde in dieser Arbeit der Weg über das MEX-File gewählt. Das bedeutet, daß einerseits zu der ALBERT–Implementation des Wärmeleitungsproblems eine MATLAB–MEXRoutine bereitgestellt wurde, die den Aufruf dieses C–Programms

als MATLAB–Funktion erlaubt. Andererseits heißt es, daß Datenumwandlungsroutinen implementiert wurden, die aus den Datenstrukturen, die ALBERTfür die Speicherung dünn besetzter Matrizen verwendet, die Daten auslesen und in das vom MATLAB–C–Interface verwendete Format für dünn besetzt Matrizen umwandeln. Da sich die von MATLAB zum Einlesen dünn besetzter Matrizen über das MEX–Interface bereitgestellten Funktionen als äußerst träge erwiesen wurde diese Umwandlung direkt in den ALBERT–Quellcode integriert. Damit könnte das Interface zwar in zukünftigen Versionen von MATLAB unbrauchbar werden, wenn in diesen die Speicherformate für dünn besetzte Matrizen geändert werden, jedoch rechtfertigt der hierdurch erzielte Geschwindigkeitszuwachs dieses Risiko in jedem Fall. Außerdem dürfte die Gefahr der Änderung der Speicherformate als äußerst gering eingestuft werden, da die Datentypen sehr ausführlich dokumentiert sind. Abbildung 3.3 stellt den Weg der Daten während des gesamten Programmablaufs im Überblick schematisch dar.

Kapitel 4

Ergebnisse der ersten Testrechungen

Wir wollen uns nun den numerischen Tests widmen. Dazu werden zunächst die Ergebnisse der ungesteuerten Referenzrechnung zusammengestellt. Danach werden nacheinander die Ergebnisse zu den verschiedenen Rechenansätzen vorgestellt und miteinander verglichen.

Zu allen hier vorgestellten Ergebnisse wurden die Rechnungen mit linearen Lagrange–Elementen durchgeführt. Für die Rechnungen mit einer Auflösung von bis zu zwei globalen Verfeinerungsstufen, sowie die Rechnungen auf adaptiven Gittern wurden SUN Blade 100 Workstations mit 512MB Hauptspeicher und einer UltraSparcII CPU mit 500MHz Taktfrequenz verwendet. Bei den Rechnungen für die Gitter mit drei globalen Verfeinerungsstufen kamen zwei SUN Ultra 60 elite3D mit je 2GB Hauptspeicher und 360MHz UltraSparc CPUs, sowie eine SUN Enterprise 450 mit 4GB Hauptspeicher und 400MHz UltraSparc CPU zum Einsatz, um sicherzustellen, daß genügend Hauptspeicher für die insbesondere beim Lösen der gewöhnlichen Differentialgleichungssyteme sehr großen Systeme zur Verfügung steht. Die Laufzeiten sind hier leider nicht zu 100% vergleichbar, da es sich mit Ausnahme der Enterprise 450, bei sämtlichen Rechnern um Arbeitsplatzrechner handelt, die natürlich unterschiedlicher Auslastung durch weitere Benutzer unterlagen.

In einem ersten Abschnitt widmen wir uns aber zunächst der Skalierung der Größen in der Gleichung auf dem Rechner und den damit verbundenen Konsequenzen für die Ausgabedaten.

4.1 Maßeinheiten und Skalierungen

In den numerischen Tests wurde jeweils die Temperatur so skaliert, daß ein Wert von 1.000 für die Temperatur einer realen Temperatur von 1000°C entspricht. Außerdem ist, wie in Abbildung 1.1 zu sehen, das Gebiet mit einer Höhe von 1.6 angenommen. Gehen wir von einem 16 cm hohen Gleisprofil aus, so müssen wir Distanzen offenbar in dm angeben. Wir erhalten also für die Temperatur einen Skalierungsfaktor von $\frac{1}{1000}$ und für die Distanzen einen Faktor von 10. Wie wirken sich nun diese Skalierungen auf die

Koeffizienten in der Gleichung und den Randbedingungen aus?

Um diese Frage zu beantworten, betrachten wir die Maßeinheiten, in welchen die Koeffizienten angegeben werden:

- spezifische Dichte ϱ in $\frac{kg}{m^3}$,
- spezifische Wärmekapazität c in $\frac{m^2}{s^2\,°C}$,
- Wärmeleitfähigkeit λ in $\frac{kg\ m}{s^3\,°C}$,
- Wärmeaustauschkoeffizient κ in $\frac{kg}{s^3\,°C}$.

Für $\alpha = \frac{\lambda}{c\,\varrho}$ erhalten wir damit

$$\frac{\frac{kg\ m}{s^3\,°C}}{\frac{m^2}{s^2\,°C}\ \frac{kg}{m^3}} = \frac{m^2}{s},$$

Beim Übergang von Metern zu Dezimetern ergibt sich hier folglich ein Faktor 100 für α. Die Temperatur kürzt sich hier heraus, so daß eine Temperaturskalierung in α also nicht berücksichtigt werden muß. Reskalieren wir daher die Zeit um den Faktor 100, so können wir im Code mit dem gleichen Wert für α rechnen, wie in der Theorie. Die Zeit im Programm ist also auf Centisekunden skaliert. Ein Realzeitintervall $[0, 45]$ in Sekunden transformiert sich in der numerischen Formulierung auf das Intervall $[0, 4500]$ in Centisekunden.

Für die Proportionalitätskonstante in der Robin–Randbedingung $\partial_\nu u = -\frac{\kappa}{\lambda}(u_{ext} - u)$ ergibt sich

$$\frac{\frac{kg}{s^3\,°C}}{\frac{kg\ m}{s^3\,°C}} = \frac{1}{m}.$$

Da sich die Normale ν auf der linken Seite der Gleichung mit dem gleichen Faktor skaliert, muß diese Änderung im Code nicht berücksichtig werden. Mit der oben gewählten Skalierung der Zeit können wir also die Modellgleichung direkt für die numerische Umsetzung auf dem Computer übernehmen, wenn wir berücksichtigen, das gerechnete Zeitintervalle zu hundertsteln sind um die Daten auf die reale Situation abzubilden.

4.2 Die ungesteuerte Referenzrechnung

Die ungesteuerte Referenzrechnung wurde, wie bereits weiter oben erläutert, mit einer Robin–Randbedingung durchgeführt, welche die Normalenableitung der Temperatur als proportional zur Temperaturdifferenz von Rand– und Umgebungstemperatur angibt. Dabei wurden die Parameter $q_k = \frac{\kappa}{\lambda} = \frac{50}{26.4} \approx 1.89394$ (für k=1,. . . ,7) so gewählt, daß ein einfaches Abkühlen des Stahlprofils an Luft wiedergegeben wird[1]. Das Programm liefert nach einer Kühldauer von etwa 45 Sekunden die in Abbildung 4.1 dargestellte

[1] Die Größe dieses Parameters orientiert sich an der in [5] angegebenen.

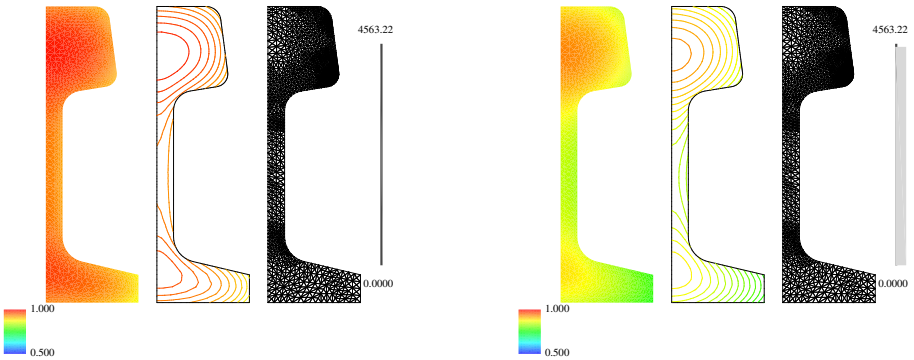


Abbildung 4.1: Anfangs- und Endtemperaturverteilung und Gitter zur Referenzrechnung mit konstanten Koeffizienten.

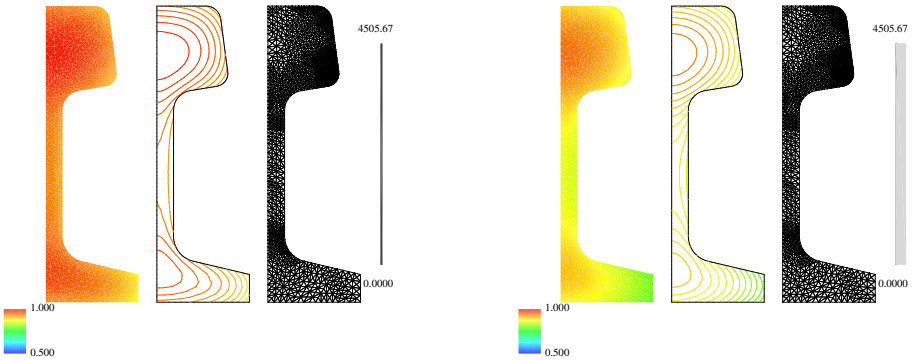


Abbildung 4.2: Anfangs- und Endtemperaturverteilung und Gitter zur Referenzrechnung mit nichtlinearen Koeffizienten.

Temperaturverteilung. Die maximale Temperatur im Rechengebiet war nach dieser Zeit 934°C. Das Minimum der Temperatur lag bei 756°C. Die Isothermen in den Abbildungen in diesem und im folgenden Kapitel sind in 10 °C–Abständen gewählt. Abbildung 4.2 zeigt die gleiche Rechnung für den Löser mit Nichtlinearität, d.h. mit den am Anfang jedes Zeitschrittes gemäß der im Abschnitt 1.3.3 angegebenen Gleichungen an die Lösung des vorhergehenden Zeitschritts angeglichenen Koeffizienten. Hier lag die Endtemperatur im Maximum bei 950°Cund im Minimum bei 781°C. Es sieht also so aus, als hätte die Linearisierung der Koeffizienten bereits in diesem Temperaturbereich einige Auswirkungen. Die Isolinienplots zeigen aber deutlich, das wir zwar quantitativ eine merklichen Unterschied feststellen können, die beiden Rechnungen aber qualitativ zum selben Ergebnis führen. Der qunatitative Unterschied von nahezu genau 16°C bestätigt sich bei der Rechnung mit dreifach global verfeinertem Gitter und stark verringerter Toleranz für die Zeitschrittweitensteuerung ebenso wie die Absolutwerte selbst. Allerdings erhöhen sich dabei bereits beim einfachen Lösen der Wärmeleitungsgleichung, ohne die im Vergleich erheblich teurere Berechnung der Steuerung, die Rechenzeiten von wenig mehr als einer halben Minute auf fast zehn Minuten. Es gibt also praktisch keinen Grund höhere Gitterauflösungen als zwei globale Verfeinerungsstufen zu wählen.

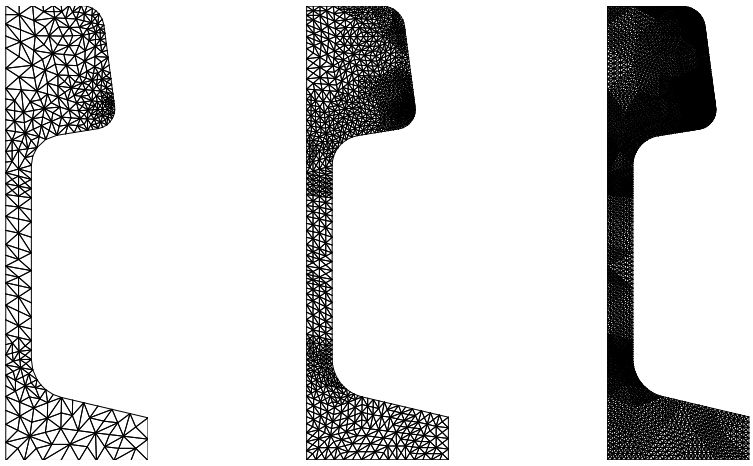


Abbildung 4.3: Das einfach/zweifach/dreifach global verfeinerte Gitter.

4.3 Ergebnisse mit hochauflösenden festen Gittern

Die Rechnungen, die diesem Abschnitt zugrunde liegen, wurden auf festen global-verfeinerten Gittern ausgeführt. Wie im vorigen Abschnitt diskutiert zeigte sich, daß es keinen Sinn hat eine höhere Auflösung als zwei globale Verfeinerungsstufen zu wählen. Es wurde daher ein zeitinvariantes zweifach global verfeinertes Gitter verwendet, wie es in Abbildung 4.3 in der Mitte dargestellt ist. Diese Auflösung entspricht 1357 Freiheitsgraden (bzw. Gitterknoten im Fall der hier verwendeten linearen Lagrange–Elemente). Die im Folgenden aufgeführten Testrechnungen wurden sämtlich mit $Q = R = I_{1357}$ und C gemäß folgenden MATLAB–Codes ($n = 1357$) durchgeführt.

```
C=sparse([],[],[],6,n,0);
C(1,60)=3; C(1,22)=-1; C(1,4)=-1;
C(2,63)=2; C(2,3)=-1; C(2,2)=-1;
C(3,51)=1; C(3,43)=-1;
C(4,55)=1; C(4,47)=-1;
C(5,92)=2; C(5,9)=-1; C(5,16)=-1;
C(6,83)=3; C(6,34)=-1; C(6,10)=-1; C(6,15)=-1;
```

Die Ausgangsmatrix C liefert also Temperaturdifferenzen zwischen Referenzpunkten im Innern der Geometrie und Meßpunkten auf der Oberfläche. Die Meß– und Referenzpunkte sind in Analogie zu den in [15] verwendeten gewählt. Die Punkte mit den Knotennummern 60 und 63 liegen im Fuß, die Punkte 50 und 55 liegen im Steg und dir Knoten 83 sowie 92 liegen im Kopf.[2] Das es zu den Punkten im Kopf mehr Vergleiche gibt, soll ein zusätzliches Gewicht auf die Kühlung des Kopfes legen, da zu erwarten ist, daß sich dort die hohen Temperaturen wegen seiner relativ zur Oberfläche verhältnismäßig großen Masse sehr viel länger halten, als im dünnen Steg, bzw. dem Fuß, der im Verhältnis eine sehr viel gößere Oberfläche gemessen an seiner Masse aufweist. Man beachte, daß der Knoten mit der Nummer 60 selbst ebenfalls im Ausgang

[2]Die genaue Lage der aufgeführtren Gitterknoten kann in Abbildung 1.2 abgelesen werden.

vorkommt. Zusätzlich zu den oben beschriebenen Temperaturdifferenzen liefert der Ausgang also auch die Temperatur selbst im diesem Punkt.

Die linearisierten Parameter, die für die Rechnungen verwendet wurden sind im einzelnen (ohne Maßeinheiten, siehe dazu oben den Abschnitt 4.1)

- $\lambda = 26.4$,
- $c = 7620.0$,
- $\varrho = 654.0$,
- $\gamma_k = 2.64\lambda \approx 69,7$.

4.3.1 ODE–Zugang

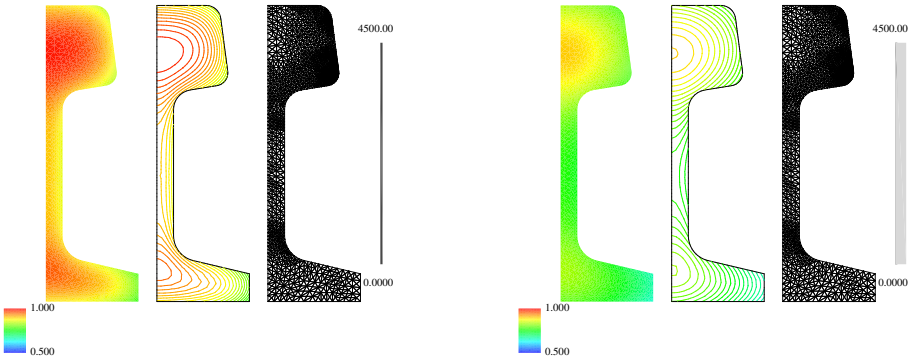


Abbildung 4.4: Die Rechenergebnisse zur gemischten Randbedingung im ODE–Ansatz.

Die Testrechnung zu den obigen Daten dauert laut MATLAB 5Std.39min 15.85s. Das Ergebnis ist in Abbildung 4.4 dargestellt. Der linke Teil der Abbildung zeigt die Anfangswerte, die nach der Vorlaufrechnug in ALBERT vorlagen. Rechts ist das Endergebnis nach der Rechnung abgebildet. Der Zeitpunkt, den das rechte Bild darstellt, ist $t = 45$s, wenn $t = 0$ bei den Daten aus dem linken Bild gesetzt wird. Abbildung 4.5 zeigt die zeitlich Entwicklung der sieben verschiedenen Kontrollparameter. Diese Grafiken sollen als Referenz für das Ergebnis der Rechnung im PDE–Ansatz dienen. In dieser sowie allen weiteren Grafiken sind die sieben Kontrollparameter q_1 bis q_7 der Reihe nach von oben links nach unten rechts aufgetragen. In Abbildung 1.1 im Kapitel 1 sind die von MATLAB vergebenen Randnummern eingezeichnet. Anhand dieser soll hier kurz tabellarisch festgehalten werden, welcher der Kontrollparameter auf welches Randstück wirkt.

Kontrollparameter	Randstücke
q_1	12, 13, 14
q_2	11
q_3	6, 7, 8, 9, 10
q_4	15, 16
q_5	3, 4, 5
q_6	2
q_7	1

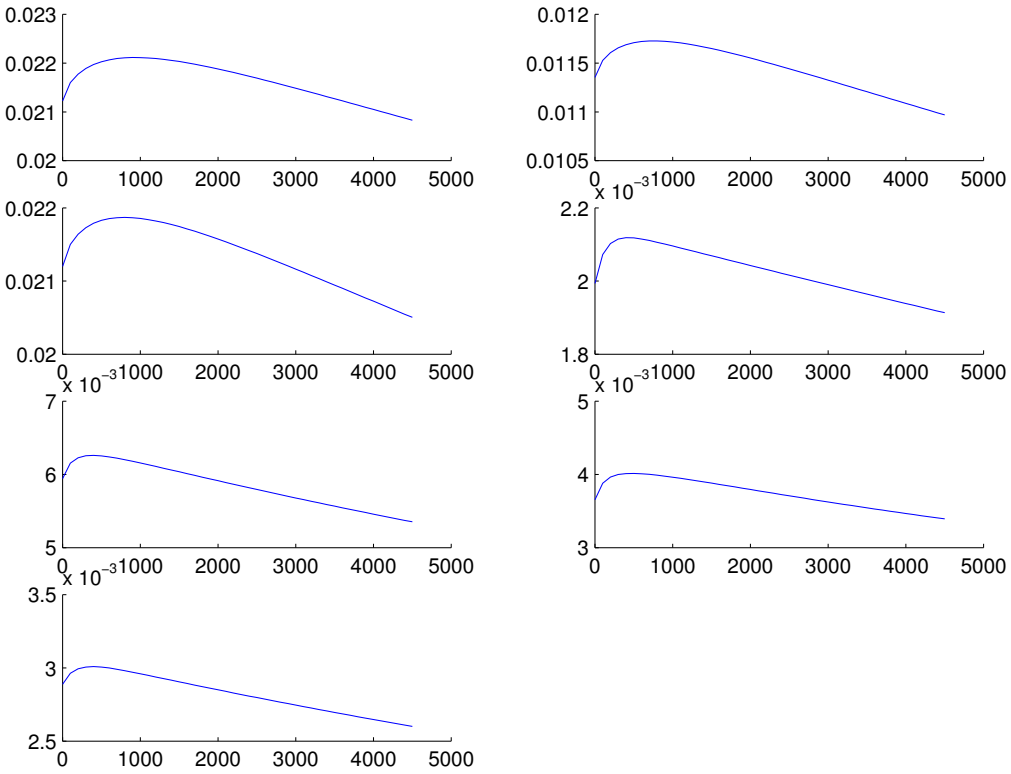


Abbildung 4.5: Die Kontrollparameter zur gemischten Randbedingung im ODE–Ansatz als Funktionen der Zeit

Außerdem werden hier als zusätzliche Referenzwerte noch die maximale, sowie die minimale Temperatur bei $t = 45$s angegeben. Die maximale Temperatur betrug 901°C, die minimale Temperatur lag bei 652°C.

An dieser Stelle soll sofort auf zwei Bedenken an diesem Ergebnis eingegangen werden. Einerseits erhalten wir hier eine extrem lange Rechenzeit. Andererseits liefert das Ergebnis ein deutlich ausgeglicheneres Bild, als es z.B. in [15] der Fall ist. Die dort erzielten Ergebnisse liegen im Maximum knapp unter den hier erreichten Werten. Allerdings liegt die Minimaltemperatur dort nach 50 Sekunden Kühldauer im Bereich von 300-500°C, abhängig von der Startverteilung. Also mindestens 150°C unter dem hier erreichten Resultat. Desweiteren zeigt sich die hier verwendete Anordnung äußerst resistent gegen Änderungen der Parameter. Auf Änderungen der Gewichtung der Kosten für die Steuerung, bzw. für den Ausgang des Systems, also einer Verlagerung der Gewichtung zwischen R und Q im Kostenfunktional, reagiert das System nur mit minimalen Abweichungen in der Endtemperaturverteilung. Selbst dann, wenn ein Gewichtsunterschied von mehreren Größenordnungen eingesetzt wird, weichen die Temperaturen in der Endverteilung bei $t = 45$s um weniger als 20°C von den zuvor berechneten ab. Ähnliches gilt bei Veränderung der Ausgangsmatrix C. Läßt man dort die Differenzen außer Acht und verwendet direkt die Temperatur in den Knoten: 51, 55, 60, 63, 83, 92[3] als Ausgangsdaten, weist das System damit also zur Temperaturminimierung an, so bleibt das Endergebnis nahezu gleich. Die maximale Temperatur liegt stets in einer

[3]Entsprechend der Ausgangsmatrix C_{temp} aus Kapitel 5

Umgebung von etwa 10°Cum 900°C, das Minimum findet man immer in ähnlichem Abstand von etwa 650°C.

4.3.2 PDE–Zugang

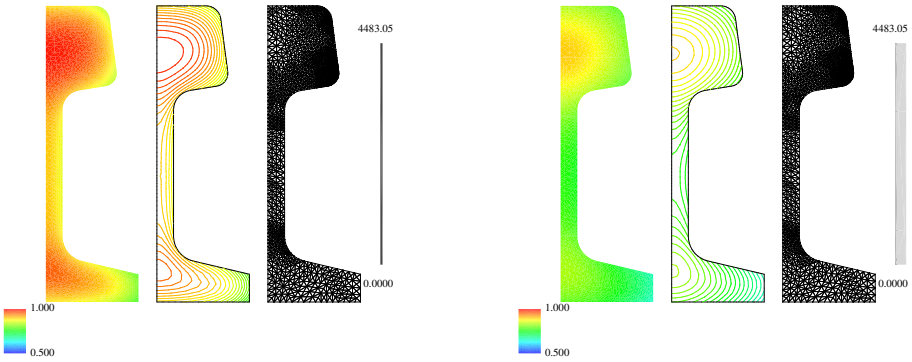


Abbildung 4.6: Die Rechenergebnisse zur gemischten Randbedingung bei Aufdatierung der Kontrollparameter in jedem 2,5,10–ten Schritt.

Zu den oben vorgegebenen Bedingungen wurden Rechnungen mit unterschiedlichen Aufdatierungshäufigkeiten bzgl. der Kontrollparameter gemacht. Ein erster Programmlauf rechnete in jedem zehnten Zeitschritt einen neuen Satz Kontrollparameter aus. Im Vergleich zu dieser Rechnung wurden dann noch weitere Rechnungen durchgeführt, welche die Kontrollparameter bereits nach jeweils 5 Zeitschritten, oder sogar in jedem zweiten Zeitschritt neu berechneten. Es zeigte sich dabei, daß alle Rechnungen im Rahmen der Darstellungsgenauigkeit exakt auf das gleiche Endergebnis führten. In der Abbildungen 4.6 sind die Ergebnisse dieser Rechnungen dargestellt. Die linke Grafik zeigt dabei die Temperaturverteilung, Isolinien (in 10°CAbstand) und das Gitter zum Beginn der Rechnung, also nach der Vorlaufrechnung. Die Grafik rechts zeigt das entsprechende am Ende der Rechnung, also nach gut 45 Sekunden. Es wurde hier darauf verzichtet die Grafiken für die verschiedenen Aufdatierungshäufigkeiten alle in die Arbeit aufzunehmen, da die Ergebnisse bis auf Hundertstelgrade identisch waren.

Die Abbildung 4.7 zeigt den Verlauf der sieben Kontrollparameter in diesen Rechnungen. Die Grafiken zeigen je einen der sieben Kontrollparameter für alle drei Aufdatierungshäufigkeiten. Auch hier sieht man wieder, das in allen Fällen identische Ergebnisse erzielt wurden. Die folgende Tabelle zeigt die Testläufe im drekten Vergleich. Die maximalen und minimalen Temperaturen beziehen sich hier jeweils auf den Endzeitpunkt[4], n ist die Anzahl der Zeitschritte, die jeweils mit einem Satz von Kühlparametern gerechnet wurde:

[4]Die Werte in der Tabelle stimmen hier und auch in den anderen Fällen nicht mit denen in den Grafiken angegebenen Daten überein, da sich in die Routine zur Setzung der Zeit in den Ausgabedateien ein Fehler eingeschlichen hatte, der dafür sorgt, daß der erste Zeitschritt mit Kühlung dort in der Zeitrechnung nicht berücksichtigt wird. Addiert man diesen auf die Werte in den Grafiken, so erhält man die in den Tabellen angegebenen Endzeitpunkte.

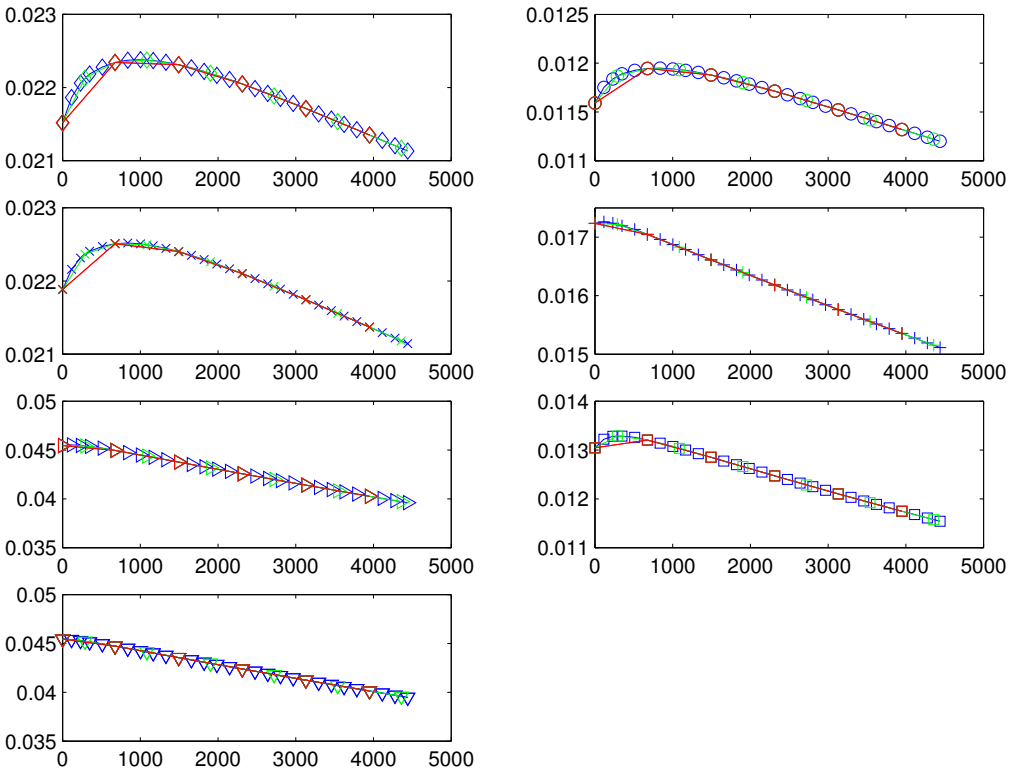


Abbildung 4.7: Die Kontrollparameter zur gemischten Randbedingung im PDE–Ansatz als Funktionen der Zeit: rot 10, grün 5 und blau 2 Zeitschritte zwischen den Aufdatierungen der Parameter.

n	max. Temp.	min. Temp	Endzeit	# Zeitschritte	Rechenzeit
2	900.07°C	650.72°C	48.93s	57	31min 25.68s
5	900.07°C	650.72°C	48.93s	57	13min 21.07s
10	900.07°C	650.72°C	48.93s	57	6min 46.67s

Die berechneten Werte weichen also jeweils um etwa 1°C von den im ODE–Ansatz erhaltenen Werten ab. Angesichts der Tatsache, daß die Werte etwa 4 Sekunden später einzuordnen sind als das Ergebnis der obigen Rechnung, können wir also davon ausgehen, daß der ODE–Ansatz und der PDE–Ansatz nicht nur qualitativ sondern auch quantitativ das selbe Ergebnis hervorbringen. Wenn sie dies auch nicht so exakt tun, wie es bei den unterschiedlichen Rechnungen im PDE–Ansatz der Fall ist. Die Ergebnisse bezüglich der Änderung der Temperatur zum Endzeitpunkt, welche im ODE–Ansatz erzielt wurden bestätigen sich auch im PDE–Ansatz. Aus diesem Grund wurden zusätzliche Rechnungen mit der bereits in der Referenzrechnung verwendeten Abstrahlrandbedingung angestellt.

4.3.3 PDE–Zugang mit alternativer Randbedingung

Wir betrachten nun Ergebnisse zu Rechnungen mit der Randbedingung gemäß Gleichung (1.3). Diese sind zwar wegen der zusätzlichen Kopplung von B an u die aus der Multiplikation der Kontrollparameter mit der Lösung u erwächst, durch die Theorie im Kapitel 2 nicht abgedeckt, sollen hier aber dennoch mit aufgeführt werden, da sie sich als deutlich geeigneter erwiesen haben, das Problem zu beeinflussen. Die von der Lösung abhängige Matrix $B(u)$ wurde, ähnlich zur Vorgehensweise bei den nichtlinearen Koeffizienten in der Referenzrechnung, dazu jeweils mit der Lösung aus dem vorhergehenden Zeitschritt aufgestellt. Auch hier wurden wieder Rechnungen mit 2, 5 und

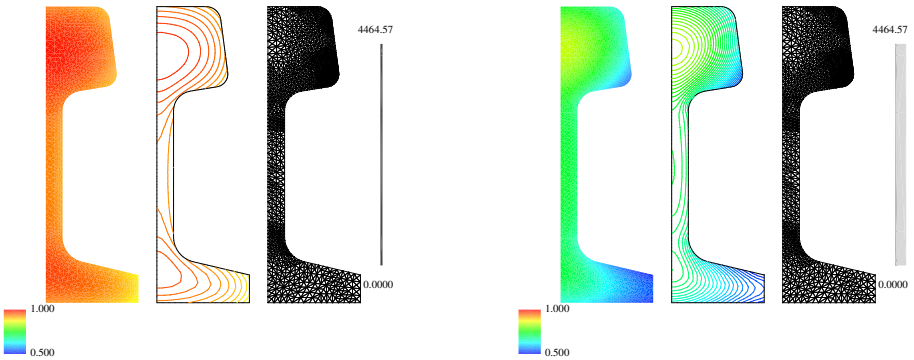


Abbildung 4.8: Die Rechenergebnisse zur Abstrahlrandbedingung bei Aufdatierung der Kontrollparameter in jedem 10–ten Schritt

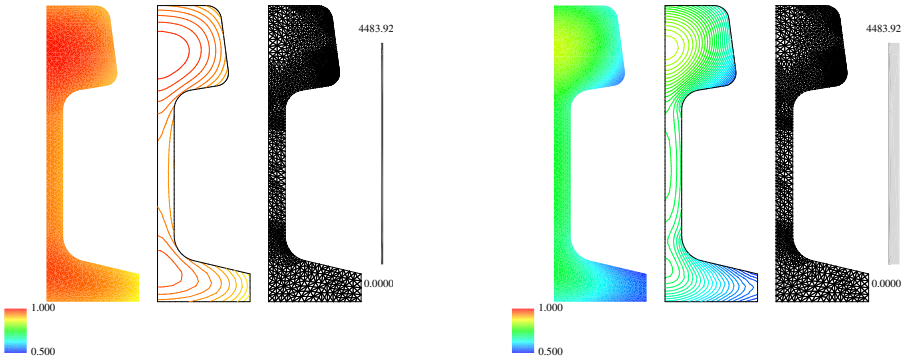


Abbildung 4.9: Die Rechenergebnisse zur Abstrahlrandbedingung bei Aufdatierung der Kontrollparameter in jedem 5–ten Schritt

10 Zeitschritten zwischen den Aufdatierungen der Kontrollparameter unternommen. Es zeigte sich dabei, daß alle Rechnungen auch hier qualitativ auf das gleiche Endergebnis führten. In den Abbildungen 4.8 bis 4.10 sind die Ergebnisse dieser Rechnungen dargestellt. Hier sind alle Grafiken aufgenommen, da die Endergebnisse nicht so exakt übereinstimmten, wie noch beim PDE–Ansatz mit der gemischten Randbedingung.

Die Abbildung 4.11 zeigt wieder den Verlauf der sieben Kontrollparameter in diesen Rechnungen. Es sind auch hier jeweils die Parameter zu den verschiedenen Aufdatierungshäufgkeiten in einem Diagramm aufgetragen um zu zeigen, daß sich auch

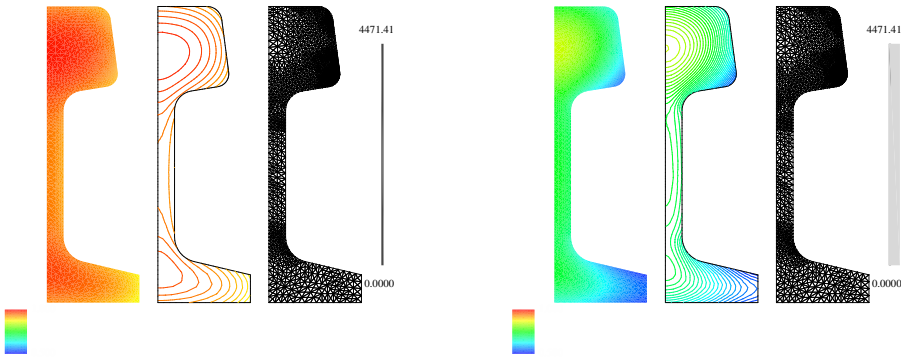


Abbildung 4.10: Die Rechenergebnisse zur Abstrahlrandbedingung bei Aufdatierung der Kontrollparameter in jedem 2–ten Schritt

diese im Langzeitverhalten identisch zeigen. Es ist allerdings im Gegensatz zu obigen Verlaufsgrafen ein deutliches Einschwingen der Kontrollparameter zu beobachten, bevor sie sich im Langzeitverhalten aneinander angleichen. Wenn wir also mit dieser Randbedingung rechnen wollen, dann sollten wir entweder für eine geeignet geringe Zeitschrittweite sorgen, oder mit häufigen Aufdatierungen der Kontrollparameter rechnen. Die hier aufgezeigten Rechnungen erfüllen offenbar bereits das erste dieser beiden Kriterien, so daß das Rechnen mit 5-10 Zeitschritten zwischen den Aufdatierungen der Kontrollparameter durchaus gerechtfertigt ist. Die folgende Tabelle zeigt, genau wie die Tabelle für den PDE–Zugang mit gemischter Ransbedingung weiter oben, die Testläufe im direkten Vergleich:

n	max. Temp.	min. Temp	Endzeit	# Zeitschritte	Rechenzeit
2	850°C	516°C	52.90 s	129	10Std. 24min 25.06s
5	848°C	513°C	53.02 s	134	4Std. 30min 44.57s
10	845°C	508°C	53.54 s	133	2Std. 1min 18.88s

Es sind hier keine Nachkommastellen der Temperaturen in der Tabelle aufgeführt, da bedingt durch die Modellgenauigkeit die Temperaturen ohnehin nur mit einer Genauigkeit von $+/-10$°C zu berechnen sein sollten. In der Tabelle für den PDE–Ansatz mit gemischter Randbedingung sind die Nachkommastellen nur aufgeführt um zu betonen, daß die verschiedenen Programmläufe dort alle exakt das gleiche Endergebniss berechnet haben.

4.4 Vergleich der Ergebnisse

Es hat sich gezeigt, das der ODE- und PDE–Ansatz mit gemischten Randbedingungen in der zu erwartenden Genauigkeit auf das gleiche Ergebnis führen. Allerdings scheint dieses Ergebnis wenig nützlich für die technische Anwendung, da sich die Rechnungen wie bereits weiter oben erläutert sehr resistent gegen Änderungen in den q_k zeigten, auf Änderungen von γ dagegen sehr viel stärker reagierten[5]. Das das System

[5]Siehe dazu auch die Ergebnisse in Abschnitt D.3 in Anhang D

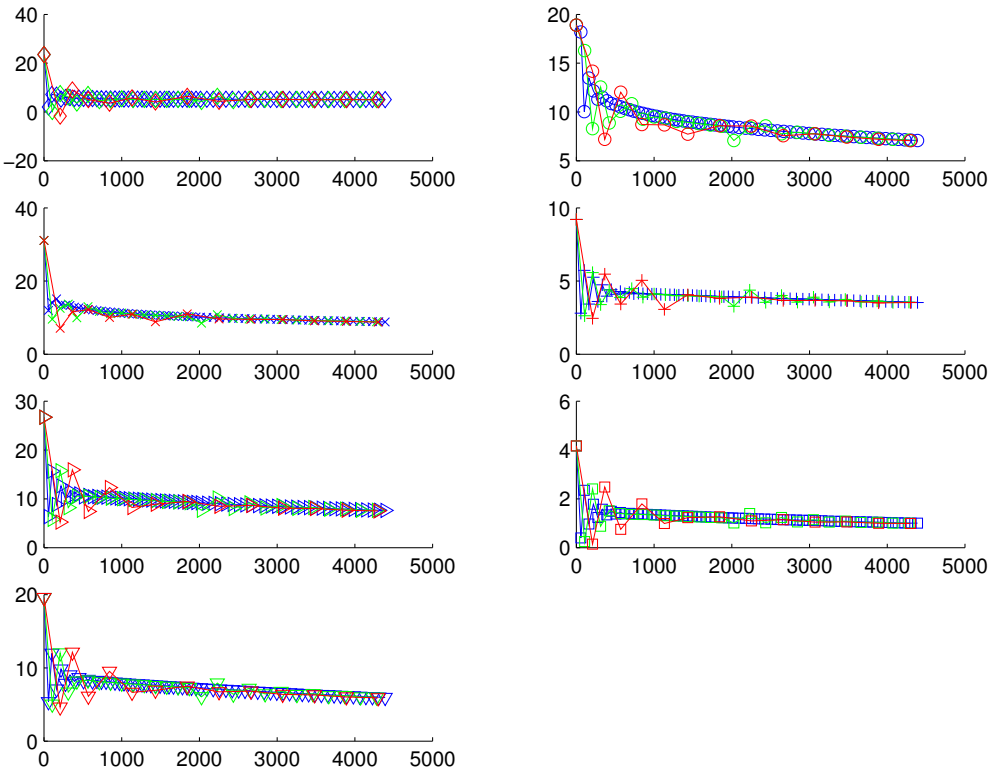


Abbildung 4.11: Die Kontrollparameter als Funktionen der Zeit: rot 10, grün 5 und blau 2 Zeitschritte zwischen den Aufdatierungen der Parameter

auf Änderungen in γ empfindlicher reagiert erscheint andererseits sehr plausibel, wenn man sich die Ergebnisse der Rechnungen mit der Abstrahlrandbedingung im Vergleich dazu ansieht. Im letzten Kapitel dieser Arbeit sollen nun diese Ergebnisse erweitert werden. Es soll dort untersucht werden, auf welche Änderungen das System in welcher Weise reagiert. Diese Rechnungen werden sämtlich mit der Abstrahl–Randbedingung durchgeführt, da es offenbar keinen Sinn macht, dies mit der gemischten Randbedingung zu versuchen.

Kapitel 5

Auswirkungen der Wahlen von Kostenfunktional und Ausgangsmatrix

Als letztem wichtigen Bestandteil dieser Arbeit widmen wir uns der Analyse der Auswirkungen verschiedener Wahlen von Kostenfunktionalen und Ausgangsmatrizen. Dabei soll insbesondere untersucht werden, wie sich unterschiedliche Gewichtungen der Anteile der Kosten für die Kontrollparameter, bzw. der Kosten für die Lösung auf den Endzustand auswirken. In anderen Rechnungen soll betrachtet werden, wie sich Änderungen der Ausgangsmatrix auf das Ergebnis, sowie die Kontrollparameter auswirken. Als Referenz für die hier erstellten Ergebnisse dienen die Ergebnisse aus dem vorigen Kapitel.

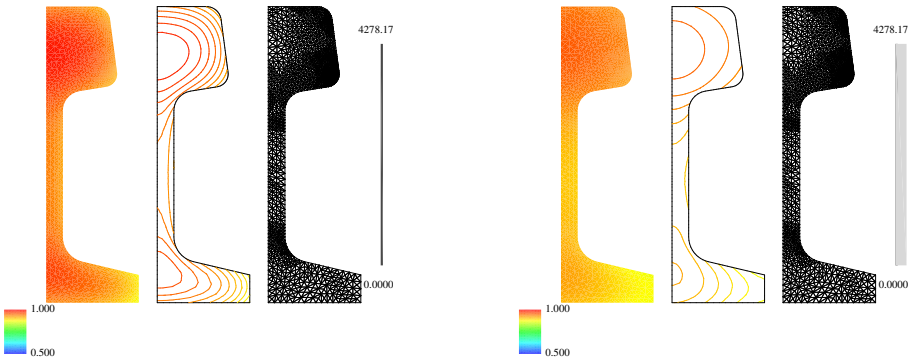


Abbildung 5.1: Die Rechenergebnisse für $Q = I$, $R = 10I$ und $C = C_{60}$

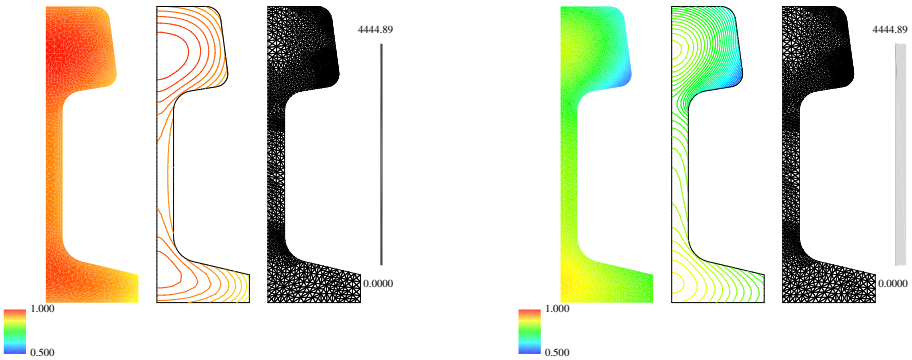


Abbildung 5.2: Die Rechenergebnisse $Q = I$, $R = I$ und $C = C_{diff}$

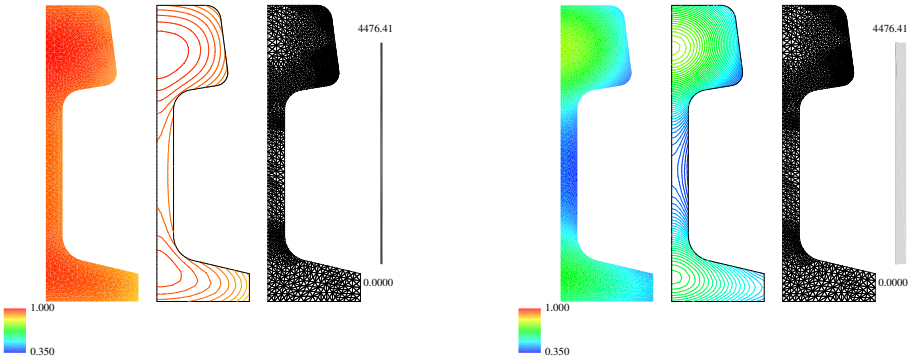


Abbildung 5.3: Die Rechenergebnisse $Q = I$, $R = I$ und $C = C_{temp}$

5.1 Auswirkungen verschiedener Gewichtungen im Kostenfunktional

Zunächst werden wir die Ergebnisse der Rechnungen mit unterschiedlichen Gewichtungen der Kosten für den Ausgang und der Kosten für die Kontrollparameter betrachten. Gegenüber den Rechnungen aus dem letzten Kapitel wird hier nur die Gewichtung im Kostenfunktional, also die Faktoren an den beiden Matrizen Q und R verändert. Die Matrizen werden weiterhin als Vielfache der Einheitsmatrix angesehen.

Wir betrachten dazu nocheinmal die Verlaufsgrafen zu den Kontrollparametern in Abbildung 4.11. Eine Beobachtung, die bei den Rechnungen mit der Abstrahlrandbedingung immer wieder gemacht wird ist, daß die Kontrollparameter, die sich nach der ersten Berechnung ergeben, relativ groß sind. Dieses Verhalten führt nun bei einer Verlagerung des Gewichts auf den Ausgang des Systems, z.B. durch die Wahl: $R = 10^{-4}I$ und $Q = I$ dazu, das im ersten Schritt deutlich zu stark gekühlt wird, so daß selbst bei nur zwei Zeitschritten bis zur Neuberechnung der Kontrollparameter die Gradienten so stark angewachsen sind, daß die nächsten Kontrollparameter sogar negative Werte annehmen. Es wird also in den weiteren Schritten zunächst an einigen Rändern geheizt. Dadurch wird wiederum der Algorithmus zur Lösung der Riccatigleichung derart destabilisiert, das er nach wenigen Iterationen zum Absturz führt. Das die Stabilität derart

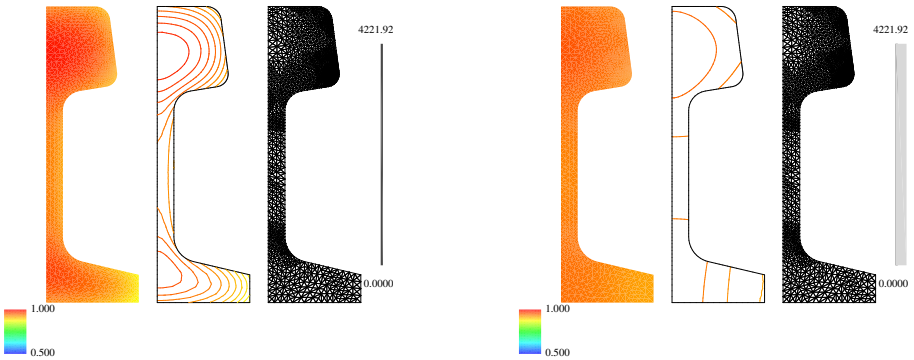


Abbildung 5.4: Die Rechenergebnisse $Q = I$, $R = 10I$ und $C = C_{diff}$

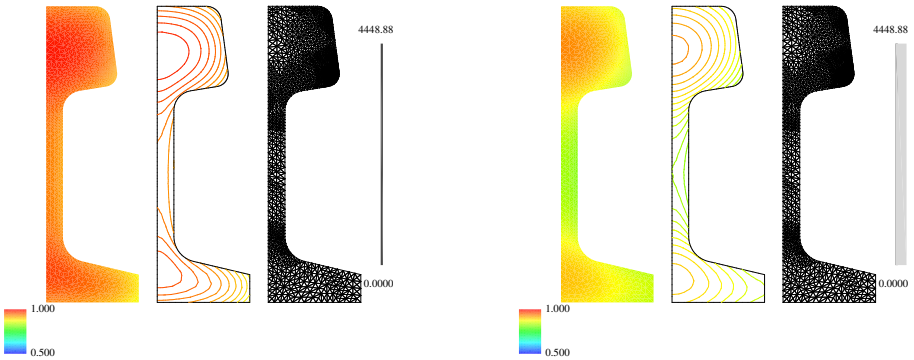


Abbildung 5.5: Die Rechenergebnisse $Q = I$, $R = 10I$ und $C = C_{temp}$

leidet ist angesichts der im Löser verwendeten Newtonmethode[1] wenig verwunderlich, wenn man bedenkt, daß durch die starken Schwankungen in der Lösung u, die mit dem starken Kühlen in den ersten Schritten und dem "Gegenheizen" in den darauffolgenden Schritten einher gehen, ein sehr negativer Einfluß in die Kondition des Problems unternommen wird, welcher für iterative Methoden allgemein sehr negative Auswirkungen mit sich bringt. Mit dieser Randbedingung sind also Gewichtsverschiebungen hin zum Ausgang in der aktuellen Implementation nicht behandelbar. Allerdings scheint dies auch nicht schlimm zu sein, da mit der im Kapitel 4 gerechneten Gewichtung schon sehr gute Kühlergebnisse erzielt worden sind, die jedoch noch starke Temperaturunterschiede in der Verteilung über die Geometrie zulassen. Festzuhalten ist hier noch die Tatsache, daß dieses Verhalten offenbar nichts mit der Gewichtsverlagerung um mehrere Größenordnungen zu tun hat, da es auch schon bei einer Wahl von $Q = I$ und $R = 4I$ auftritt.

Betrachten wir nun das Ziel des Ingenieurs, eine möglichst gleichmäßige Temperaturverteilung zu erhalten, so scheint es doch sinnvoll das Gewicht zur Kontrolle hin zu verschieben. Der Kühleffekt wird so mit möglichst geringen Kühlparametern erzielt, damit die großen Gradienten in der Temperatur gar nicht erst auftreten können. Genau dieses Verhalten läßt sich mit einer Wahl von $Q = I$, $R = 10I$ auch bestätigen, wie man in Abbildung 5.1 sehen kann. Allerdings ist eine Wahl von $R = 10I$ auch nicht sinnvoll, da in diesem Fall die Gesamtkühlwirkung deutlich geringer ist, als bei Abkühlung an

[1] Siehe dazu auch Anhang B

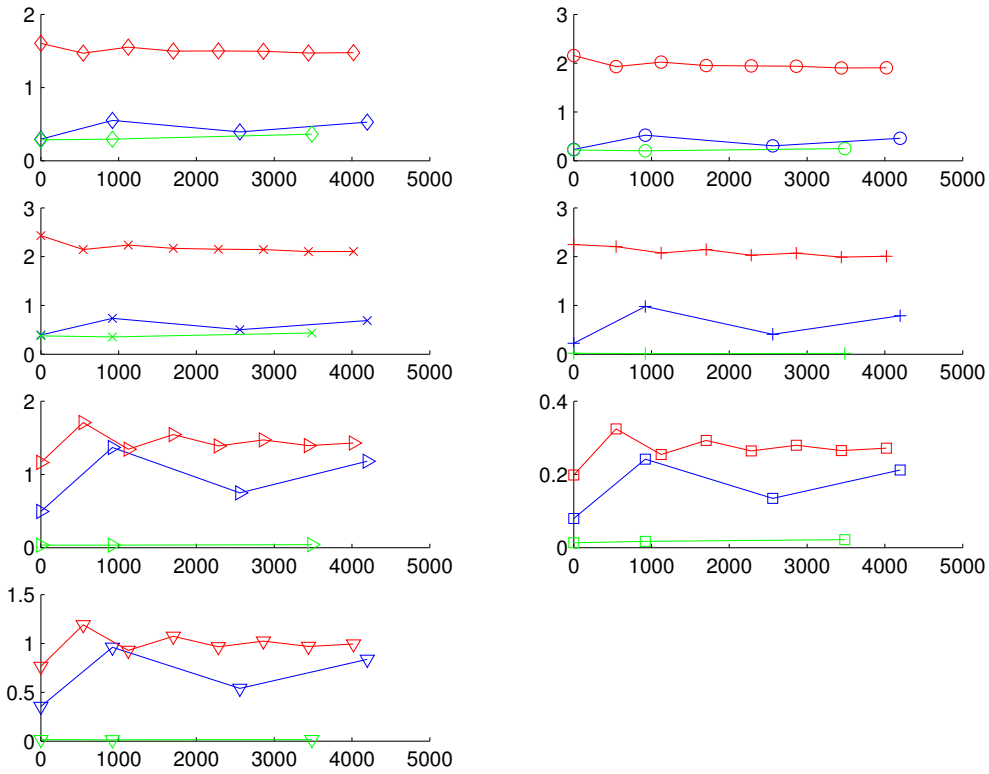


Abbildung 5.6: Die Kontrollparameter als Funktionen der Zeit für $Q = I$, $R = 10I$. rot:C_{temp}, grün:C_{diff}, blau:C_{60}

Luft. Ziel dieser Rechnung war es aber weniger einen "optimalen" Gewichtungsfaktor zu finden, als die Bestätigung der Anschauung durch das verwendete Modell, die man hier klar als erfolgrreich ansehen kann. Was genau ein optimaler Gewichtungsfaktor ist, sollte wohl ohnehin der anwendende Ingenieur mit dem entsprechenden Fachwissen in Materialkunde entscheiden.

5.2 Auswirkungen der Wahl der Ausgangsmatrix

Dieser Abschnitt wird für verschiedene Ausgangsmatrizen beobachten, wie sich die Wahl des Ausgangs auf die Temperaturverteilung am Ende der Rechnung auswirkt. Um die Sprechweise etwas zu vereinfachen, definieren wir zunächst Bezeichnungen für drei verschiedene Ausgangsmatrizen. Die Ausgangsmatrix C, die wir im vorigen Kapitel definiert haben, nennen wir wegen der zusätzlichen Aufnahme des Knotenpunktes mit der Nummer 60 die Matrix C_{60}. Die Ausgangsmatrix aus der der zusätzliche Punkt entfernt ist, d.h. bei der gilt $C(1, 60) = 2$, nennen wir C_{diff}, da hier nur noch *Diff*erenzen von Temperaturwerten in gewissen Gitterknoten der Triangulierung betrachtet werden. Die dritte Matrix nennen wir C_{temp} und definieren sie wie folgt

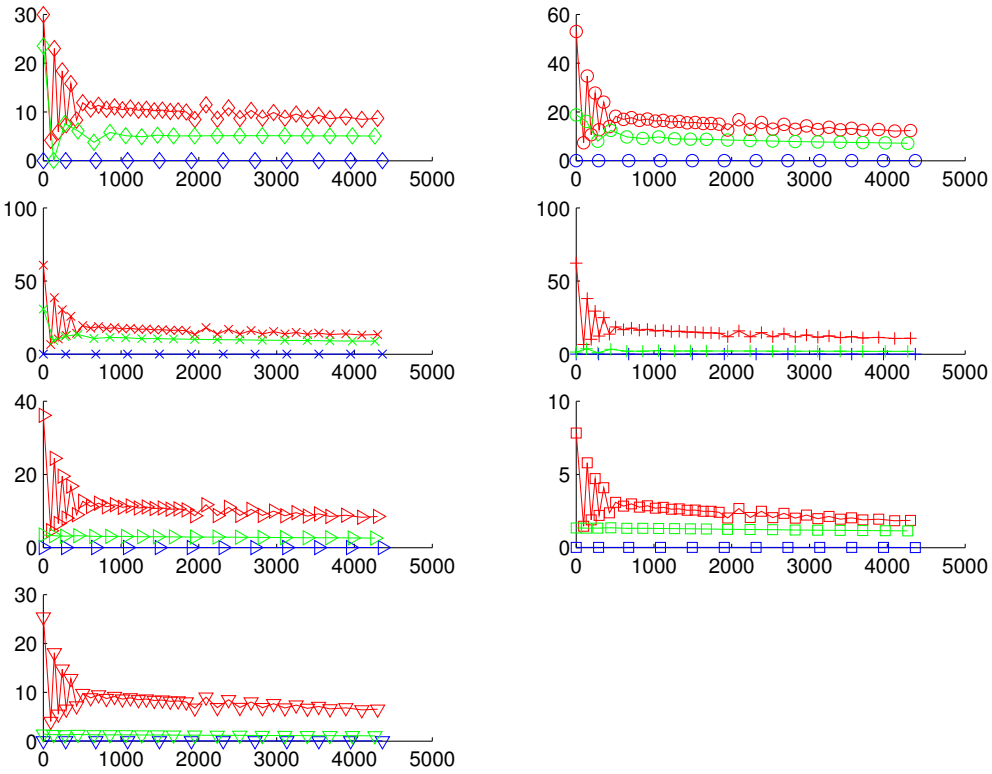


Abbildung 5.7: Die Kontrollparameter als Funktionen der Zeit für $Q = I$, $R = I$. rot:C_{temp}, grün:C_{diff}, blau:C_{60}

```
C=sparse([],[],[],6,n,0);
C(1,60)=1;
C(2,63)=1;
C(3,51)=1;
C(4,55)=1;
C(5,92)=1;
C(6,83)=1;
```

In der ersten Rechnung verwenden wir C_{diff}. Es werden also nur Temperaturdifferenzen im Ausgang betrachtet. Die Anschauung legt hier Nahe, das dieses Vorgehen dazu führen sollte, daß im Profilfuß gegenüber der Rechnung aus dem vorigen Kapitel eine gleichmäßigere Temperaturverteilung eingestellt wird, um die Temperaturunterschiede klein zu halten, da die Bedingung der Temperaturminimierung im Gitterknoten mit der Nummer 60 wegfällt. In Abbildung 5.2 ist das Ergebnis zu dieser Rechnung dargestellt. Dort ist deutlich die weniger starke Abkühlung des Profilfußes gegenüber den Ergebnissen in den Abbildungen 4.8–4.10 zu erkennen.

Die Abbildung 5.3 zeigt die Ergebnisse zu einer entsprechenden Rechnung mit der Ausgangsmatrix C_{temp}. Dort ist sehr schön zu erkennen, wie sich eine erheblich stärkere Kühlung des gesamten Profils einstellt, sobald man im Kostenfunktional die Temperatur selbst als Minimierungsgröße einführt. Man beachte besonders, das hier die Temperaturskala bereits bei 350°C beginnt, da die minimale Temperatur hier im Bereich von

etwa 360-370°C liegt und damit deutlich unter der bisher ausreichenden Grenze von 500°C. Allerdings sollte auch bedacht werden, daß das Modell in diesem Bereich wegen seiner linearisierten Koeffizenten nicht mehr in hinreichendem Maße korrekt arbeiten dürfte. Dazu müssen die Koeffizienten entsprechend angepaßt, d.h. durch geeignetere Mittelwerte ersetzt werden. Wir können also höchstens qualitative Aussagen aus dieser Grafik ziehen. In Abbildung 5.5 ist die Rechnung nocheinmal mit der Gewichtung $R = 10I$ wie in den Rechnungen aus Abschnitt 5.1 dargestellt. Dort ist wieder die erheblich geringere Abkühlung zu sehen, die aus dieser Gewichtung resultiert. Allerdings sehen wir auch hier gegenüber den entsprechenden Rechnungen mit C_{60} eine erheblich Verringerung der Temperaturen im gesamten Profil.

Ausblick

Abschließend soll nun aufgezeigt werden, wo noch Verbesserungsmöglichkeiten zum vorgestellten Verfahren liegen. Dabei wird unterschieden zwischen solchen Verbesserungvorschlägen, die sich auf das Modell und die mathematisch relevanten Anteile beziehen und solchen, die eine Bereitstellung des Verfahrens für die technische Anwendung in der Arbeitswelt eines Stahlwerkers erfordern würde.

Zum mathematischen Modell

Das mathematische Modell kann in folgende Richtungen weiterentwickelt werden:

- Anpassung auf andere Temperaturbereiche. z.B.: durch Berücksichtigung der Phasenumwandlung im Stahl
- Modellierung des Aufheizens der Umgebung des Profils und dessen Einflußes auf die Kühltemperatur und –wirkung, speziell in der Vorlaufrechnung
- Optimierung der gemischten Randbedingung in Bezug auf realistischeren Wärmeaustausch mit dem Kühlfluid, also Bereitstellung eine "optimalen" Parameters γ

Zur Implementierung

Die Implementierung sollte grundsätzlich in folgenden Punkten verbessert, bzw. weiterentwickelt werden:

- Entwicklung eines problemangepassten Fehlerschätzers, der ggf. auch die nichtlineare Gleichung abdeckt und damit adaptive Rechnungen mit linearisierten und ggf. sogar nichtlinearen Koeffizienten erlaubt.
- Stabilitätstests mit realistischen Anfangs– und Randdaten aus der Anwendung
- Beschleunigung der Berechnung der neuen Kühlkoeffizienten, bzw. der Steuerungsmatrix z.B.: durch Implementierung einer dem LyaPack ähnlichen C-Programmbibliothek und damit Entkopplung von MATLAB

- Implementierung bzw. Einbindung eines geeigneten ODE-Lösers zur Beschleunigung des ODE–Ansatzes.
- Beschleunigung der ADI–Iteration bei der Berechnung der Steuerungsmatrix, durch Implementation einer optimaleren Shiftparameterfunktion.
- Stabilisierung der Newtonmethode bei der Lösung der algebraischen Riccatigleichung. z.B. durch Vorgabe eines stabilisierenden Initialfeedbacks, falls die betragsmäßig kleinsten Eigenwerte von A besonders klein werden.

Für die Anwendung im Arbeitsalltag eines Technikers im Walzwerk sollte das Programm außerdem mit einer graphischen Benutzerschnittstelle ausgestattet werden, in der unter anderem auch die Kennwerte der relevanten Stähle einfach ausgewählt werden können.

Anhang A

Linear–quadratische Optimierung in n Dimensionen

Dieser erste Teil des Anhangs widmet sich der Zusammenfassung von Ergebnissen zur Linear–quadratischen Optimierung in n Dimensionen. Es wird dabei auf Beweise verzichtet, um eine bessere Übersichtlichkeit zu gewährleisten. Diese Zusammenstellung legt dabei wenig Wert auf Vollständigkeit der Ausführung. Vielmehr soll eine gute Übersicht über die wichtigsten Methoden und Resultate geliefert werden, die in der Numerik dieser Probleme relevant sind. Eine etwas ausführlichere Darstellung findet sich in [3]. Eine ähnliche Darstellung mit vielen Beweisen und ausführlichen Herleitungen liefert Mehrmann in [20]. Eine umfassende Ausführung der Theorie stellt [8] bereit.

A.1 Linear–quadratische Optimierung auf dem endlichen Zeitintervall $[t_0, t_{end}]$

Wir wiederholen zunächst die allgemeine Definition eines zeitinvarianten dynamischen Systems (LTI-System).

Definition A.1.1 (LTI-System):
Sei $A \in \mathbb{R}^{n,n}, B \in \mathbb{R}^{n,m}, C \in \mathbb{R}^{p,n}, D \in \mathbb{R}^{p,m}$

$$\begin{aligned} \text{Zustandsgleichung:} \quad & \dot{x}(t) &&= Ax(t) + Bu(t) \text{ für } t > t_0 (= 0) \\ \text{Anfangsbedingung:} \quad & x(t_0) &&= x^0 \\ \text{Ausgangsgleichung:} \quad & y(t) &&= Cx(t) (+Du(t)) \end{aligned} \tag{A.1}$$

Dabei ist $x(t)$ der Zustand des Systems (A.1), $y(t)$ ist das Ausgangssignal und $u(t)$ ist das Eingangssignal.

Unter der optimalen Steuerung von (A.1) wollen wir nun diejenige verstehen, welche ein vorgegebenes Kostenfunktional minimiert. Wir definieren dazu das Standardproblem

der klassischen Kotrolltheorie[1].

Definition A.1.2 (linear–quadratisches Optimierungsproblem):

Seien $Q = Q^T \in \mathbb{R}^{p,p}, S \in \mathbb{R}^{p,m}, R = R^T \in \mathbb{R}^{m,m}$ sowie $R \geq 0$.
Minimiere das Kostenfunktional

$$\mathcal{J}(x,u) := \frac{1}{2}\int_0^{t_{end}} y(t)^T Q y(t) + 2x(t)^T S u(t) + u(t)^T R u(t) dt \tag{A.2}$$

unter der Nebenbedingung (A.1)

Einige Autoren fordern in der Definition nur die Symmetrie von Q, R, mit dem Verweis darauf, dass stärkere Anforderungen in den Existenz– und Eindeutigkeitsaussagen folgen. Eine verbreitete Vereinfachung, auf die auch wir uns später zurückziehen werden, ist $S = 0$.

Mit dieser Definition formulieren wir den folgenden

Satz A.1.3:

Sei die Lösung u_* zu (A.1),(A.2) stückweise stetig, x_* die zugehörige Lösungstrajektorie zum Anfangswertproblem (AWP) (A.1). Dann existiert eine *Kozustandsfunktion* $\mu(t) \in \mathbb{R}^n$, so daß x_*, u_*, μ das lineare Randwertproblem (RWP)

$$\begin{pmatrix} A & 0 & B \\ Q & A^T & S \\ S^T & B^T & R \end{pmatrix}\begin{pmatrix} x \\ \mu \\ u \end{pmatrix} = \begin{pmatrix} I_n & 0 & 0 \\ 0 & -I_n & 0 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} \dot{x} \\ \dot{\mu} \\ \dot{u} \end{pmatrix} \tag{A.3}$$

$$x(t_0) = x^0 \quad , \quad \mu(t_{end}) = 0 \tag{A.4}$$

lösen.

Dabei beachten wir, dass (A.3) keine zusätzlichen Regularitätsanforderungen an u mit sich bringt, da $\dot{u}$ hier nur formal auftaucht. Zu diesem Satz läßt sich ein Optimalitätsnachweis führen.

Satz A.1.4:

Seien x_*, u_*, μ_* Lösung des obigen RWP und sei

$$\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \geq 0$$

Dann gilt:

$$\mathcal{J}(x,u) \geq \mathcal{J}(x_*, u_*) \text{ für alle Paare } (x,u), \text{ die (A.1) lösen.}$$

Definition A.1.5 (Hamiltonische Matrizen):

$H \in \mathbb{R}^{2n,2n}$ heißt *Hamiltonisch*, falls gilt

$$HJ = (HJ)^T \text{ mit } J = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix}$$

[1] siehe dazu auch [18, 7, 3, 20, 8]

Bemerkung A.1.6:

Hieraus folgt insbesondere, daß alle Hamiltonischen Matrizen die folgende Form haben

$$H = \begin{pmatrix} A & G \\ Q & A^T \end{pmatrix} \text{ für } A, G, Q \in \mathbb{R}^{n,n} \text{ mit } G = G^T, Q = Q^T$$

Im Folgenden verwenden wir die Vereinfachung $S = 0$ und nehmen an, daß P positiv definit ist. Diese Voraussetzung an P bedeutet für das Optimierungsproblem lediglich, daß es keine kostenfreien Steuerungsparameter geben darf. Aus der letzten Zeile von (A.3) erhalten wir jetzt den Zusammenhang von u und μ gemäß $u(t) = -R^{-1}B^T\mu(t)$. Damit ist (A.3) äquivalent zu

$$\begin{pmatrix} \dot{x} \\ \dot{\mu} \end{pmatrix} = \underbrace{\begin{pmatrix} A & BR^{-1}B^T \\ Q & -A^T \end{pmatrix}}_{=:H} \begin{pmatrix} x \\ \mu \end{pmatrix} \tag{A.5}$$

Wir machen nun den Ansatz $\mu(t) = X(t)x(t)$ dann ergibt sich mit (A.4) sofort $X(t_{end}) = 0$. Setzen wir außerdem $G := BR^{-1}B^T$ so folgt aus (A.5)

$$\dot{x} = (A - GX)x, \tag{A.6}$$

und mit $\dot{\mu} = -\dot{X}x - X\dot{x}$

$$\begin{aligned} & (Q + A^T X)x = -\dot{X}x - X\dot{x} = -\dot{X}x - X(A - GX)x \\ \Leftrightarrow \quad & (\dot{X}(t) + X(t)A + A^T X(t) - X(t)GX(t) + Q)x(t) = 0 \text{ für alle } t \in [t_0, t_{end}] \end{aligned} \tag{A.7}$$

Da im Allgemeinen $x \not\equiv 0$ gilt, erfüllt X die *Matrix-Riccati-Differentialgleichung*

$$\begin{aligned} \dot{X}(t) = -\mathcal{R}(X(t)) := Q + A^T X(t) + X(t)A - X(t)GX(t) \\ \text{mit } X(t_{end}) = 0 \text{ für alle } t \in [0, t_{end}] \end{aligned} \tag{A.8}$$

Wir formulieren damit das folgende abschließende Resultat

Satz A.1.7 (Existenz der eindeutigen Feedbacksteuerung u_*):

Für $Q \geq 0, R > 0$ und $t_{end} < \infty$ besitzt das linear quadratische Optimierungsproblem (A.2) genau eine Lösung, die durch die *Feedbacksteuerung*

$$u_*(t) = -R^{-1}B^T X_*(t)x(t) \tag{A.9}$$

gegeben ist. Dabei ist X_* die eindeutige symmetrische Lösung der Matrix-Riccati-Differentialgleichung (A.8). Außerdem sind zu jedem Anfangswert x^0 die optimalen Kosten gegeben durch

$$\mathcal{J}(x^0, u_*) = \frac{1}{2}(x^0)^T X_*(0)x^0.$$

A.2 Linear–quadratische Optimierung auf dem unbeschränkten Zeitintervall

Bisher haben wir immer $t_{end} \in \mathbb{R}_{>0}$ betrachtet. Wir wollen uns nun die veränderte Situation im Fall des unbeschränkten Zeitintervalles ansehen.

Sei also $t_{end} = \infty$. Wir betrachten $R > 0$ sowie $Q \geq 0$. Die Kosten für die Steuerung sollen beschränkt sein, aus $\lim_{t\to\infty} u(t) \neq 0$ folgen aber sofort unbeschränkte Kosten, da R positiv definit ist. Das gleiche gilt falls $\lim_{t\to\infty} x(t)^T C^T Q C x(t)(t) > 0$. Daher erhalten wir

$$\lim_{t\to\infty} u(t) = 0 \tag{A.10}$$

$$\lim_{t\to\infty} x(t)^T C^T Q C x(t) = 0 \tag{A.11}$$

Sicher gilt nach (A.4)

$$\lim_{t\to\infty} \mu(t) = 0 \tag{A.12}$$

Und damit ebenfalls

$$\lim_{t\to\infty} x(t) = 0 \tag{A.13}$$

falls in unserem obigen Ansatz $X(t) \equiv X$ gilt. Das ist aber klar. Um dies einzusehen, wählen wir zunächst $t_0 < t_1 < t_2$ und betrachten die zugehörigen Lösungen X_1 und X_2 der Matrix-Riccati-Differentialgleichung. Wegen der Eindeutigkeit der Lösung muß dann X_1 durch Reskalierung der Zeit in X_2 auf das Existenzintervall von X_1 hervorgehen, d.h. $X_2(t) = X_1(ct)$ mit $c = \frac{t_1}{t_2}$. Offensichtlich gilt dann

$$\dot{X}_2(t) = c\dot{X}_1(ct).$$

Und damit ergibt sich durch Grenzübergang $t_2 \to \infty$ mit $t_2 = t_{end}$

$$\lim_{t_{end}\to\infty} \dot{X}_{t_{end}} = \lim_{t_{end}\to\infty} \frac{t_1}{t_{end}} \dot{X}_1(t) = 0.$$

Gehen wir nun in (A.8) zum Grenzwert über, so erhalten wir die *algebraische Riccatigleichung* (ARE)

$$0 = \mathcal{R}(X_\infty) = C^T Q C + X_\infty A + A^T X_\infty - X_\infty B R^{-1} B^T X_\infty \tag{A.14}$$

Diese hat im Gegensatz zur Matrix-Riccati-Differentialgleichung keine eindeutige Lösung. Zwar soll X_∞ als Grenzwert symmetrischer Lösungen $X_{t_{end}}$ wieder symmetrisch sein, aber auch das schränkt den Kreis der möglichen Lösungen noch nicht genügend ein. Wir betrachten daher das dynamische System mit optimaler Feedbackkontrolle, also den geschlossenen Kreis, bei dem wir das optimale Feedback u_* direkt als Eingang verwenden.

$$\dot{x}(t) = (A - BR^{-1}B^T X_*)x(t) =: A_* x(t), \text{ mit } x(t_0) = x^0$$

Die eindeutige Lösung dieses linearen Anfangswertproblems ist offensichtlich $x_*(t) := e^{A_* t} x_0$. Da wir bereits wissen, das $\lim_{t\to\infty} x(t) = 0$ gelten muß, folgt nun die Bedingung $\sigma(A_*) \subset \mathbb{C}^-$. Bevor wir nun damit die abschließenden Resultate formulieren, benötigen wir noch zwei kurze Definitionen.

Definition A.2.1:
Sei$A \in \mathbb{R}^{n,n}$, $B \in \mathbb{R}^{n,m}$, $C \in \mathbb{R}^{k,n}$.

i) (A, B) heißt *stabilisierbar*, falls

$$\operatorname{rg}[\lambda I - A, B] = n$$

für alle $\lambda \in \mathbb{C}_{\geq}0$.

ii) (A, C) heißt *entdeckbar*, falls

$$\operatorname{rg}\begin{bmatrix} \lambda I - A \\ C \end{bmatrix} = n$$

für alle $\lambda \in \mathbb{C}_{\geq}0$.

Damit läßt sich der folgende Satz beweisen.

Satz A.2.2 (Existenz einer eindeutigen Lösung der ARE):
Seien $F \geq 0$, $G \geq 0$, sowie (A, G) stabilisierbar und (F, A) entdeckbar, dann hat die algebraische Riccatigleichung

$$0 = F + A^T X + XA - XGX \tag{A.15}$$

eine eindeutige symmetrische stabilisierende Lösung $X_* \geq 0$, es gilt also insbesondere $\sigma(A - GX_*) \subset \mathbb{C}^-$.

Bemerkung A.2.3:
Unter wenig stärkeren Voraussetzungen läßt sich ebenfalls erreichen, daß $X_* > 0$. Sind die geforderten Definitheitsvoraussetzungen nicht erfüllt, so kann trotzdem eine stabilisierende Lösung X existieren, diese ist dann aber ggf. indefinit.

Satz A.2.4 (Existenz der eindeutigen Feedbacksteuerung u_*):
Für $Q \geq 0, R > 0$ sowie (A, B) stabilisierbar und $(C^T QC, A)$ entdeckbar besitzt das linear–quadratische Optimierungsproblem (A.2) mit $t_{end} = \infty$ genau eine Lösung

$$u_*(t) = -R^{-1}B^T X_*(t)x(t) \tag{A.16}$$

Dabei ist X_* die eindeutige symmetrische stabilisierende Lösung der algebraischen Riccatigleichung (A.15), mit $F = C^T QC$ und $G = BR^{-1}B^T$.

A.3 Systeme mit Masse

Wir haben in den vorigen Abschnitten stets LTI–Systeme der Form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \\ x(0) &= x_0 \end{aligned} \tag{A.17}$$

betrachtet. Im Kapitel 3 haben wir aber gesehen, daß das betrachtete Problem auf ein System der Form

$$\begin{aligned} M\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \\ x(0) &= x_0 \end{aligned} \tag{A.18}$$

führt. Dabei ist M symmetrrisch und positiv definit. In diesem Abschnitt wollen wir uns daher mit solchen Systemen auseinandersetzen. Wir werden hier die Methode betrachten, die auch im LyaPack zum Lösen solcher Systeme verwendet wird.

Dazu zerlegen wir die Matrix M in $M = M_L M_U$ und definieren die Variable z als

$$z(t) := M_U x(t).$$

Damit gilt offensichtlich

$$\dot{z}(t) = \dot{M}_U x(t) + M_U \dot{x}(t) = M_U \dot{x}(t).$$

Definieren wir nun weiter

$$\begin{aligned} \tilde{A} &:= M_L^{-1} A M_U^{-1}, \\ \tilde{B} &:= M_L^{-1} B, \\ \tilde{C} &:= C M_U^{-1}, \end{aligned}$$

dann gilt für die Zustandsgleichung

$$\dot{z}(t) = M_U \dot{x}(t) = M_L^{-1} A x(t) + M_L^{-1} B u(t) = M_L^{-1} A M_U^{-1} M_U x(t) + M_L^{-1} B u(t) = \tilde{A} z(t) + \tilde{B} u(t), \tag{A.19}$$

und für die Ausgangsgleichung

$$y(t) = Cx(t) = C M_U^{-1} M_U x(t) = \tilde{C} z(t). \tag{A.20}$$

Es ergibt sich also das System

$$\begin{aligned} \dot{z}(t) &= \tilde{A} z(t) + \tilde{B} u(t) \\ y(t) &= \tilde{C} z(t) \end{aligned}$$

mit dem Kostenfunktional (A.2) und $S = 0$. Die resultierende ARE ist:

$$0 = \tilde{F} + \tilde{A}^T \tilde{X} + \tilde{X}^T \tilde{A} + \tilde{X}^T \tilde{G} \tilde{X} \tag{A.21}$$

mit $\tilde{F} = \tilde{C}^T Q \tilde{C}$ und $\tilde{G} = \tilde{B} R^{-1} \tilde{B}^T$.

Damit haben wir das Problem mit Massenmatrix in ein äquivalentes Problem in Standardform überführt. LyaPack fordert die Voraussetzungen, das M invertierbar und $M^{-1}A$ stabil ist, damit solche Systeme behandelt werden können. Das ist aber auch leicht einzusehen, da das System:

$$\begin{aligned} \dot{x}(t) &= M^{-1} A x(t) + M^{-1} B u(t), \\ y(t) &= Cx(t) \end{aligned}$$

ebenfalls äquivalent zu (A.18) ist.

Anhang B

Lösung der algebraischen Riccatigleichung in n Dimensionen

B.1 Die Newton Methode zur Lösung algebraischer Riccati–Gleichungen

Wir haben im vorigen Kapitel gesehen, daß zum Lösen des linear quadratischen Optimalsteuerungsproblems eine algebraische Riccati–Gleichung gelöst werden muß. Dieses Kapitel soll sich nun mit der numerischen Lösung eben dieser Gleichungen befassen.

Wir betrachten also die algebraische Riccati–Gleichung:

$$\mathcal{R}(X) := F + A^T X + XA - XGX = 0 \tag{B.1}$$

mit den oben gefundenen Anforderungen an A, B, C, Q, R, sowie F, G definiert durch $F := C^T QC$ und $G := BR^{-1}B^T$. Die Idee ist nun, die ARE als nichtlineares Gleichungssystem zu betrachten und mit einem Newtonverfahren zu lösen. Wir suchen also formal nach einer Darstellung $X_{k+1} = X_k - \mathcal{R}^{-1}(X_k)\mathcal{R}(X_k)$. Dazu betrachten wir die Frechét–Ableitung von $\mathcal{R} : \mathcal{S}_{\backslash} \to \mathcal{S}_{\backslash} := \{S \in \mathbb{R}^{n,n} : S = S^T\}$:

$$\begin{aligned}
\mathcal{R}'_X(Z) &:= \lim_{t\to 0} \tfrac{1}{t} \left(\mathcal{R}(X + tZ) - \mathcal{R}(X)\right) \\
&= \lim_{t\to 0} \tfrac{1}{t} \left(A^T(X + tZ) + (X + tZ)A - (X + tZ)G(X + tZ) - A^T X - XA + XGX\right) \\
&= \lim_{t\to 0} \left((A - GX)^T Z + Z(A - GX) - ZGtZ\right) \\
&= (A - GX)^T Z + Z(A - GX)
\end{aligned} \tag{B.2}$$

und erhalten damit die gewünschte Darstellung:

$$X_{k+1} = X_k - \left(\mathcal{R}'_{X_k} \left(\mathcal{R}(X_k)\right)^{-1}\right)$$

oder

$$\begin{aligned}
\mathcal{R}'_{X_k}(N_k) &= -(X_k), \\
X_{k+1} &= X_k + N_k.
\end{aligned} \tag{B.3}$$

Damit können wir nun den folgenden Algorithmus formulieren.

Algorithmus B.1.1 (Newtonverfahren für ARE's):

1 Wahl eines Startwertes X_0
2 FOR k=1,2,... UNTIL $||N_k|| \leq tol$ OR $||\mathcal{R}(X_k)|| \leq tol$
3 Löse die Lyapunov-Gleichung
4 $(A - GX_k)^T N_k + N_k(A - GX_k) = \mathcal{R}(X_k)$ bzgl. N_k
5 Datiere auf: $X_{k+1} = X_k + N_k$
6 END FOR

In der Sprechweise der LyaPack Dokumentation heisst das: Löse die Lyapunov–Gleichung

$$(A^T - K_{k-1}B^T)X_k + X_k(A - BK_{k-1}^T) = -C^TQC - K_{k-1}RK_{k-1}^T$$

für X_k und datiere K gemäß $K_k = X_kBR^{-1}$ auf. Bezüglich der Umsetzung der Abbruchbedingung im Lyapack sei hier wieder auf die Dokumentation [22] verwiesen. Aussagen zur Konvergenz liefern [3] und [20]. Lancaster und Rodman gehen in [16] auch auf Konvergenzraten ein. Sie zeigen, für einen Startwert X_0, für welchen $A - GX_0$ stabil ist, quadratische Konvergenz. In dieser Arbeit ist bereits A selbst stabil. Wir können also bereits zum Startwert $X_0 = 0$ quadratische Konvergenz erwarten. Wie man die Lyapunov–Gleichungen numerisch löst wird im Kapitel B.2 behandelt.

B.2 Alternating Directions Implicit (ADI) Methode

Im vorigen Abschnitt haben wir das Lösen einer algebraischen Riccatigleichung auf das Lösen einer Lyapunov–Gleichung zurückgeführt. In diesem Kapitel wollen wir nun sehen, wie man diese tatsächlich löst. Wir werden uns dazu zunächst die allgemeine Idee der Alternating–Directions–Implicit–Methode anschauen und dann in einem späteren Abschnitt sehen, wie das Lösen der Lyapunov–Gleichungen in dieses Schema passt.

B.2.1 Die Grundidee (Peaceman–Rachford ADI Iteration)

Sei A reell symmetrisch und positiv definite (spd) Matrix aus dem $\mathbb{R}^{n,n}$ und s ein bekannter Vektor aus dem $\mathbb{R}^n$. Wir betrachten das lineare Gleichungssystem

$$Au = s \tag{B.4}$$

und suchen seine Lösung $u \in \mathbb{R}^n$. Die Idee der ADI Iteration ist nun A als Summe zweier Matrizen H und V darzustellen, für welche sich die Gleichungssysteme $(H + pI)v = r$ und $(V + pI)w = t$ besonders effizient lösen lassen. Dabei sind die rechten Seiten r und t bekannt, p ist ein noch näher zu untersuchender Iterationsparameter[1] und I die

[1] siehe auch B.2.4

Einheitsmatrix im $\mathbb{R}^{n,n}$. Sind H und V spd Matrizen, so existieren positive Parameter p_j, für welche die Peaceman–Rachford Alternating Directions Implicit Iteration

$$\begin{aligned}(H + p_j I)u_{j-\frac{1}{2}} &= (p_j I - V)u_{j-1} + s,\\ (V + p_j I)u_j &= (p_j I - H)u_{j-\frac{1}{2}} + s\end{aligned} \tag{B.5}$$

für $j = 1, 2, \ldots$ konvergiert. Dabei ist u_0 ein vorgegebener Startvektor, der oft als der Null–Vektor gewählt wird. Wir nehmen nun also Konvergenz der u_j gegen ein $u \in \mathbb{R}^n$ an und betrachten inwiefern damit eine Lösung von (B.4) gefunden ist. Wir betrachten zunächst die Differenz der Gleichungen (B.5),

$$V(u_{j-1} - u_j) - p_j(u_{j-1} + u_j) + 2p_j u_{j-\frac{1}{2}} = 0. \tag{B.6}$$

Offensichtlich konvergiert für $j \to \infty$ der erste Summand gegen 0 und der zweite gegen $-2p_j u$. Damit muß der letzte Summand gegen $2p_j u$ konvergieren. Also konvergiert mit $u_j \to u$ auch die Folge der Zwischenschritte $u_{j-\frac{1}{2}} \to u$. Bilden wir nun die Summe obiger Gleichungen, so erhalten wir

$$2Hu_{j-\frac{1}{2}} + V(u_{j-1} + u_j) + p_j(u_j - u_{j-1}) = 2s. \tag{B.7}$$

Im Grenzübergang ergibt sich:

$$2Hu + 2Vu = 2s$$

Da nach Voraussetzung $H+V = A$ gilt, löst u also Gleichung B.4. In [26] zeigt Wachspress, daß die (sub)optimale Berechnung der Iterationsparameter p_j eng verbunden ist mit der Konvergenz des Algorithmus. Diesen Umstand beleuchten wir im folgenden Abschnitt.

B.2.2 Konvergenz der ADI Methode und Wahl der Shiftparameter

Zur Konvergenz zeigen wir zunächst, daß sich der Fehler $e_j := u_j - u$ im j–ten Iterationsschritt als $e_j = R_j e_{j-1}$ darstellen läßt. Dazu definieren wir

$$R_j := (V + p_j I)^{-1}(H - p_j I)(H + p_j I)^{-1}(V - p_j I),$$

sowie

$$\begin{aligned}B_j &:= H + p_j I,\\ C_j &:= p_j I - V,\\ D_j &:= V + p_j I,\\ E_j &:= p_j I - H.\end{aligned}$$

Damit gilt offensichtlich

$$R_j = D_j^{-1}(-E_j)B_j^{-1}(-C_j)$$

und aus (B.5) ergibt sich:

$$\begin{aligned}u_{j-\frac{1}{2}} &= B_j^{-1}C_j u_{j-1} + B_j^{-1}s\\ u_j &= D_j^{-1}E_j B_j^{-1}C_j u_{j-1} + D_j^{-1}E_j B_j^{-1}s + D_j^{-1}s\\ &= R_j u_{j-1} + D_j^{-1}E_j B_j^{-1}s + D_j^{-1}s\end{aligned}$$

Es bleibt also zu zeigen:

$$D_j^{-1}E_jB_j^{-1}s + D_j^{-1}s - u = -R_ju$$

Dazu verwenden wir (B.4) und $A = H + V$:

$$\begin{aligned} D_j^{-1}E_jB_j^{-1}s + D_j^{-1}s - u &= -R_jC_j^{-1}s + R_jC_j^{-1}B_jE_j^{-1}s - u \\ &= -R_j(C_j^{-1}A - C_j^{-1}B_jE_j^{-1}A + R_j^{-1})u \\ &= -R_j(C_j^{-1}(A - B_jE_j^{-1}A + B_jE_j^{-1}D_j))u \\ &= -R_j(C_j^{-1}(A - B_jE_j^{-1}(A - D_j)))u \\ &= -R_j(C_j^{-1}(A - B_jE_j^{-1}E_j))u \\ &= -R_jC_j^{-1}(A - B_j)u \\ &= -R_jC_j^{-1}C_ju \end{aligned}$$

Der Fehler nach im j–ten Iterationsschritt ist also

$$e_J = \prod_{j=1}^{J} R_je_0 =: G_Je_0.$$

Dabei ist G_J symmetrisch, da die spd Matrizen H und V kommutieren. Bezeichnen wir nun mit $\|\cdot\|$ die 2–Norm, so gilt wegen der Symmetrie $\|G_J\| = \rho(G_J)$ und damit $\|e_J\| =\leq \rho(G_J)\|e_0\|$. Das führt uns zu dem Problem: Minimiere das Maximum der Eigenwertsbeträge zu Eigenwerten von G_J. Um einzusehen, wie dies zu bewerkstelligen ist, suchen wir nach einer Darstellung für die Eigenwerte von G_J aus bekannten Größen. Dazu seien λ, γ Eigenwerte von H, bzw. V zu einem gemeinsamen Eigenvektor[2]. Offenbar lassen sich die Eigenwerte μ_j von R_j damit darstellen als:

$$\mu_j = \frac{(p_j - \lambda)(p_j - \gamma)}{(p_j + \lambda)(p_j + \gamma)}$$

Es gilt also außerdem:

$$\rho(G_J) = \max_{\lambda\in\sigma(H),\gamma\in\sigma(V)} \left| \prod_{j=1}^{J} \frac{(p_j - \lambda)(p_j - \gamma)}{(p_j + \lambda)(p_j + \gamma)} \right|$$

Damit erhalten wir das minimax–Problem für die Peaceman–Rachford–ADI Iteration:
Zu gegebenem J finde $\{p_j \in \mathbb{R} : j = 1, \ldots, J\}$, so daß $\rho(G_J)$ minimiert wird.

B.2.3 Anwendung auf Lyapunov–Gleichungen

Bisher haben wir die ADI–Methode nur als einen Löser für Gleichungsysteme der Form (B.4) betrachtet. In diesem Abschnitt wollen wir nun zeigen, inwiefern dadurch auch eine Möglichkeit gegeben ist, die Lösung einer Lyapunov–Gleichung

$$AX + XA^T = C \tag{B.8}$$

[2]die kommutierenden spd Matrizen H und V besitzen eine gemeinsame Basis aus Eigenvektoren, die damit natürlich auch Eigenvektoren von R_j sind

zu berechnen. Die Idee ist dabei, die Lyapunov–Gleichung als äquivalent zu einem Operator $\mathcal{A}$ zu betrachten, der X auf C abbildet und als Summe aus Linksmultiplikation mit A und Rechtsmultiplikation mit A^T gebildet wird. Die beiden Teiloperatoren sind offensichtlich linear und kommutieren mit der Matrixaddition. Damit haben wir eine Zerlegung des Operators $\mathcal{A}$ in zwei Operatoren analog zur Zerlegung der Matrix A aus (B.4) in die Matrizen H und V. Analog zu (B.5) findet man damit die zweischrittige Iterationsvorschrift:

$$\begin{aligned} X_0 &= 0 \\ (A + p_j I)X_{j-\frac{1}{2}} &= C - X_{j-1}(A^T - p_j I) \\ (A + p_j I)X_j &= C - X_{j-1}^T(A^T - p_j I) \end{aligned} \tag{B.9}$$

Im LyaPack wird eine davon abgewandelte Version verwendet, die eine Choleskyfaktorisierung $X_j = Z_j Z_j^H$ verwendet und in Termen von Z_j arbeitet. Gewöhnlich haben die Z_j nur $t_j \ll n$ Spalten und damit einen sehr viel kleineren Rang. Man spricht daher auch von der Cholesky factor ADI Methode, wie sie in [17] vorgestellt wird. Die algorithmische Umsetzung erläutert die LyaPack–Dokumentation [22].

B.2.4 Berechnung der (sub)–optimalen ADI-shift Parameter

In Abschnitt B.2.2 haben wir gesehen, daß die optimalen Shiftparameter für die Peaceman–Rachford–ADI Methode gegeben sind als Lösung des minimax–Problems

$$\min_{\{p_j\}} \max_{\lambda\in\sigma(H),\gamma\in\sigma(V)} \left| \prod_{j=1}^{J} \frac{(p_j - \lambda)(p_j - \gamma)}{(p_j + \lambda)(p_j + \gamma)} \right|.$$

Für die Anwendung auf Lyapunov–Gleichungen gilt offenbar $H = A$ und $V = A^T$, also $\sigma(H) = \sigma(V) = \sigma(A)$. Es reicht also, das Maximum über $\lambda, \gamma \in \sigma(A)$ zu betrachten. Ausserdem gilt o.b.d.A für $\lambda, \gamma \in \sigma(A)$

$$\frac{(p_j - \lambda)}{(p_j + \lambda)} \geq \frac{(p_j - \gamma)}{(p_j + \gamma)}.$$

Dann gilt aber auch

$$\frac{(p_j - \lambda)(p_j - \lambda)}{(p_j + \lambda)(p_j + \lambda)} \geq \frac{(p_j - \lambda)(p_j - \gamma)}{(p_j + \lambda)(p_j + \gamma)}.$$

Es reicht also aus das minimax–Problem

$$\min_{\{p_j\}} \max_{\lambda\in\sigma(A)} \left| \prod_{j=1}^{J} \frac{(p_j - \lambda)}{(p_j + \lambda)} \right| \tag{B.10}$$

zu lösen. An dieser Stelle eröffnen sich jedoch zwei Probleme:

1. Das Spektrum $\sigma(A)$ ist a priori nicht bekannt und läßt sich für sehr große A auch nicht ohne großen Rechenaufwand berechnen.

2. Selbst wenn das Spektrum bekannt wäre, gäbe es keine Möglichkeit die optimalen Parameter p_j zu berechnen.

Man beschränkt sich daher darauf, suboptimale Parameter zu berechnen. Die dabei vom LyaPack verfolgte Idee ist es, zunächst das Spektrum durch sogenannte Ritz-Werte zu „approximieren“ und dann eine Teilmenge dieser Werte als Näherungslösung für B.10 zu wählen. Näheres dazu findet der geneigte Leser in [22] im Kapitel “Lyapunov equations”. Der verwendete Ansatz ist hier sehr heuristischer Natur, so daß sich die Konvergenz der ADI Iteration ggf. durch eine andere Wahl der suboptimalen ADI–Shifts noch verbessern läßt. [4, 17, 26, 27] sind nur einige Arbeiten, die sich mit der Bereitstellung (sub)optimaler ADI Shiftparameter befassen. [26] gibt dabei außerdem einen sehr guten Überblick zur Anwendung und Geschichte der ADI-Methoden allgemein.

Anhang C

Berechnung der Projektionen für die gekrümmten Ränder

C.1 Erläuterung der Vorgehensweise

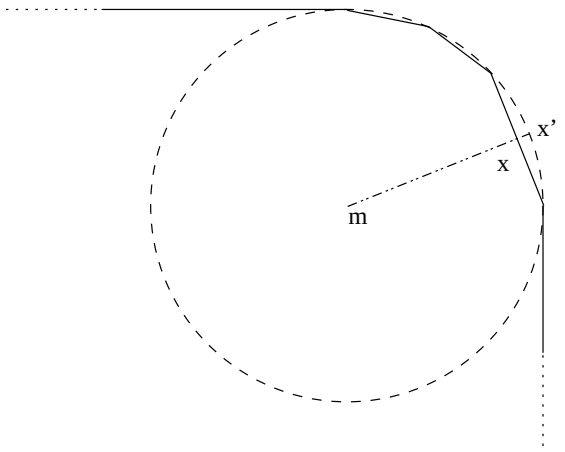


Abbildung C.1: Projektion des einzufügenden Randpunktes x auf den gekrümmten Rand.

Die gekrümmten Ränder des Gleisprofils sollen als Kreissegmente angesehen werden. Diese Kreissegmente werden derart gesucht, daß die beiden geraden Randstücke, die den gekrümmten Rand einschließen, Tangenten an den Kreis bilden. Das Kreissegment ist in der Makrotriangulierung durch einen Streckenzug aus Sekantenstücken beschrieben. Die Idee ist nun, Gitterknoten, die auf diesen Sekanten neu eingefügt werden sollen, entlang der Verbindungsgeraden zwischen den neuen Knoten und dem Kreismittelpunkt auf den Kreisrand zu verschieben.

Zu den geraden Rändern des Profils sind dabei jeweils mindestens der Anfangs- und der Endpunkt bekannt. Diese beiden Punkte legen eindeutig Geradengleichungen fest, die wir als Tangenten an einen Kreis ansehen wollen, welcher das zu berechnende Kreisseg-

ment angibt. Nun wählen wir einen in der Makrotriangulierung vorhandenen Punkt des gekrümmten Randes aus. Zu den beiden Randgeraden und dem ausgewählten Punkt berechnen wir jetzt den Kreis, der die Geraden als Tangenten besitzt, durch diesen Punkt verläuft und außerdem die "Krümmung in die richtige Richtung hat". Die beiden Tangenten und der ausgewählte Punkt liefern uns zwei mögliche Kreise (siehe Abbildung C.2). Allerding verläuft nur einer der beiden Kreise durch die Berührpunkte der Tangenten, die durch die Punkte aus der Makrotriangulierung festgelegt sind.

Da es rechnerisch einen unangemessen großen Aufwand bedeutet, für jeden neuen Randpunkt, die beiden Kreise durch den gewählten Punkt zu berechnen und anhand der Berührpunkte zu entscheiden welcher der beiden der gesuchte Kreis ist, wird die Berechnung für jedes gekrümmte Randstück vorab in MATLAB durchgeführt und nur die Werte für den Mittelpunkt des Kreises und dessen Radius in die ALBERT–Routinen eingesetzt. Außerdem können wir so sicherstellen, daß die Punkte der Makrotriangulierung, die auf den gekrümmten Rändern liegen alle auf die Kreisbögen passen und damit Fehler in der Triangulierung von vornherein ausschliessen. Soll nun in ALBERT ein neuer Randknoten auf dem gekrümmten Rand eingefügt werden, so wird der Differenzvektor zwischen dem Mittelpunkt des Kreises und den aktuellen Koordinaten des neuen Punktes berechnet. Die Länge des Differenzvektors wird dann gemäß des Radius des Kreises gestreckt und der neue Punkt an dessen neuem Ende eingefügt. Der neue Randpunkt wird folglich in Richtung der Differenzvektors auf den Rand des Kreises nach außen verschoben (Siehe Abbildung C.1. Einzelne Dreieckselemente der Triangulierung sind hier zur Förderung der Übersichtlichkeit nicht dargestellt.).

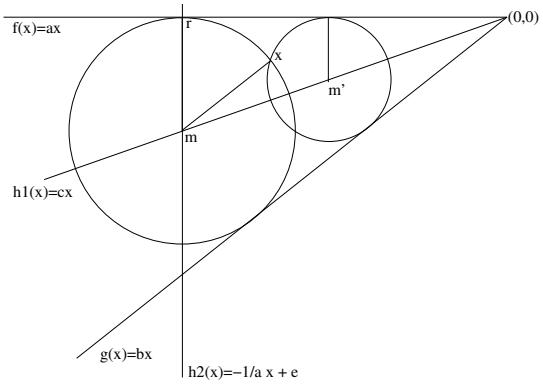


Abbildung C.2: Skizze.

C.2 Berechnung der benötigten Daten

Kommen wir nun also dazu, wie sich der Mittelpunkt und der Radius aus den gegebenen Daten berechnen lassen. Abbildung C.2 zeigt eine Referenzkonfiguration der jeweils gegebenen Situation, bei welcher der Schnittpunkt der Tangenten f und g in den Koordinatenursprung verschoben ist. Die eingezeichneten Punkte x, m, r seien dargestellt als $x = (x_1, x_2)$, $m = (m_1, m_2)$ und $r = (r_1, r_2)$. Da r der Schnittpunkt der Geraden f

und $h2$ ist, gilt

$$ar_1 = r_2 = -\frac{1}{a}r_1 + e \tag{C.1}$$

und somit

$$e = \left(a + \frac{1}{a}\right)r_1. \tag{C.2}$$

Desweiteren gilt offensichtlich

$$cm_1 = m_2 = -\frac{1}{a}m_1 + \left(a + \frac{1}{a}\right)r_1 \tag{C.3}$$

und damit

$$m_1 = \frac{a + \frac{1}{a}}{c + \frac{1}{a}}r_1 \Leftrightarrow r_1 = \frac{c + \frac{1}{a}}{a + \frac{1}{a}}m_1. \tag{C.4}$$

Sind nun also r, x und die beiden Tangentengleichungen bekannt, so benötigen wir für die Berechnung von m noch eine Darstellung von c in bekannten Größen. Damit läßt sich dann m_1 aus der Kreisgleichung bestimmen, indem wir die gefunden Gleichungen in $(x_1 - m_1)^2 + (x_2 - m_2)^2 = (r_1 - m_1)^2 + (r_2 - m_2)^2$ einsetzen. Offensichtlich ist $h1$ die Winkelhalbierende des Winkels zwischen f und g. Betrachten wir das Problem vektoriell, so ist der Richtungsvektor der Geraden $h1$ gerade die Summe der Richtungsvektoren der Geraden f und g, falls $(f(x_1) - x_2)(x_2 - g(x_1)) > 0$[1] und ihre Differenz, falls $(f(x_1) - x_2)(x_2 - g(x_1)) < 0$[2].

Wir betrachten zunächst den ersten Fall. Dazu normieren wir die beiden Richtungsvektoren

$$\begin{pmatrix} 1 \\ a \end{pmatrix} \mapsto \begin{pmatrix} \frac{1}{\sqrt{a^2+1}} \\ \\ \frac{a}{\sqrt{a^2+1}} \end{pmatrix}, \tag{C.5}$$

sowie

$$\begin{pmatrix} 1 \\ b \end{pmatrix} \mapsto \begin{pmatrix} \frac{1}{\sqrt{b^2+1}} \\ \\ \frac{b}{\sqrt{b^2+1}} \end{pmatrix}. \tag{C.6}$$

Addition liefert nun

$$\tilde{c} := \begin{pmatrix} \frac{1}{\sqrt{a^2+1}} + \frac{1}{\sqrt{b^2+1}} \\ \\ \frac{1}{\sqrt{a^2+1}} + \frac{b}{\sqrt{b^2+1}} \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{a^2+1}+\sqrt{b^2+1}}{\sqrt{a^2+1}\sqrt{b^2+1}} \\ \\ \frac{a\sqrt{b^2+1}+b\sqrt{a^2+1}}{\sqrt{a^2+1}\sqrt{b^2+1}} \end{pmatrix} \tag{C.7}$$

und durch Reskalierung erhalten wir

$$\hat{c} = \begin{pmatrix} 1 \\ c \end{pmatrix} = \tilde{c}\frac{\sqrt{a^2+1}+\sqrt{b^2+1}}{\sqrt{a^2+1}\sqrt{b^2+1}} = \begin{pmatrix} 1 \\ \\ \frac{a\sqrt{b^2+1}+b\sqrt{a^2+1}}{\sqrt{a^2+1}+\sqrt{b^2+1}} \end{pmatrix}. \tag{C.8}$$

[1] x liegt „zwischen“ den Geraden f und g.
[2] x liegt entweder oberhalb beider Geraden, oder unterhalb beider Geraden.

Es gilt also

$$c = \frac{a\sqrt{b^2+1} + b\sqrt{a^2+1}}{\sqrt{a^2+1} + \sqrt{b^2+1}}. \tag{C.9}$$

Für den zweiten Fall erhalten wir durch analoge Rechnung

$$c = \frac{a\sqrt{b^2+1} - b\sqrt{a^2+1}}{\sqrt{b^2+1} - \sqrt{a^2+1}}. \tag{C.10}$$

Betrachten wir nun die Kreisgleichung

$$0 = \left((x_1 - m_1)^2 + (x_2 - m_2)^2\right) - \left((r_1 - m_1)^2 + (r_2 - m_2)^2\right). \tag{C.11}$$

Darin ersetzen wir zunächst r_2 und m_2 nach den Gleichungen (C.1) und (C.3)

$$0 = \left((x_1 - m_1)^2 + (x_2 - cm_1)^2\right) - \left((r_1 - m_1)^2 + (ar_1 - cm_1)^2\right). \tag{C.12}$$

Nun definieren wir $d := \frac{ac+1}{a^2+1}$ und ersetzen r_1 nach (C.4)

$$0 = \left((x_1 - m_1)^2 + (x_2 - cm_1)^2\right) - \left((dm_1 - m_1)^2 + (adm_1 - cm_1)^2\right) \tag{C.13}$$

und rechnen weiter:

$$\begin{aligned}
0 \quad &= x_1^2 + x_2^2 - 2(x_1 + cx_2)m_1 - \left(c^2 + 1 - (d-1)^2 - (ad-c)^2\right)m_1^2 \\
&= x_1^2 + x_2^2 - 2(x_1 + cx_2)m_1 - \left(c^2 + 1 - d^2 + 2d - 1 - a^2d^2 + 2adc - c^2\right)m_1^2 \\
&= x_1^2 + x_2^2 - 2(x_1 + cx_2)m_1 + \left((1+a^2)d^2 - 2d - 2adc\right)m_1^2 \\
&= x_1^2 + x_2^2 - 2(x_1 + cx_2)m_1 + \left(\tfrac{(ac+1)^2}{a^2+1} - 2\tfrac{(ac+1)^2}{a^2+1}\right)m_1^2 \\
&= x_1^2 + x_2^2 - 2(x_1 + cx_2)m_1 - \tfrac{(ac+1)^2}{a^2+1}m_1^2 \\
\Leftrightarrow \quad m_1^2 \quad &= \tfrac{a^2+1}{(ac+1)^2}\left(x_1^2 + x_2^2 - 2(x_1 - cx_2)m_1\right).
\end{aligned} \tag{C.14}$$

Diese quadratische Gleichung in m_1 können wir nun zu gegebenen a, b, x_1 und x_2 leicht numerisch lösen. Aus den beiden resultierenden Lösungen für m wählen wir, durch Einsetzen der Endpunkte[3] der geraden Randkanten, die passende aus. Den jeweiligen Radius $\hat{r}$ der beiden Kreise erhalten wir aus der Kreisgleichung als:

$$\hat{r} = \sqrt{(x_1 - m_1)^2 + (x_2 - m_2)^2} \tag{C.15}$$

C.3 Die verwendeten Daten im Überblick

In diesem Abschnitt sollen kurz die verwendeten Daten zusammengefasst werden. Die folgende Tabelle stellt dabei zunächst die Steigungen der geraden Randkanten zusammen, die aus jeweils zwei Punkten auf den Kanten (den Endpunkten) berechnet wurden.

[3] Die Endpunkte kennen wir bereits aus der Makrotriangulierung. Sie ändern sich durch das Bisektionsverfahren nicht.

Kante	Punkt 1	Punkt 2	Steigung	Achsenabschnitt
Kopf oben	(0.2655,1.6)	(0,1.6)	0	1.6
Kopf rechts	(0.3826,1.245)	(0.345,1.5303)	-7.58777	4.14808
Kopf unten	(0.1797,1.14319)	(0.32441,1.16666)	0.16219	1.11404
Steg rechts	(0.09,0.345)	(0.09,1.03778)	-	-
Fuß oben	(0.5, 0.15)	(0.19607, 0.22151)	-0.23528	0.26765

Für die Kante am Steg können wir keine endliche Steigung angeben, da diese parallel zur y-Achse im gewählten Koordinatensystem verläuft. Um die Berechnungen hier durchzuführen, wurden zusätzlich zur oben angesprochenen Verschiebung in die Referenzsituation die Achsen vertauscht und so Steigung 0 erreicht. Die folgende Tabelle listet die Werte für die Mittelpunkte m und die Radien $\hat{r}$ zu den vier vorkommenden gekrümmten Rändern auf.

Randstück	m	$\hat{r}$
Steg/Fuß	(0.22758, 0.35544)	0.13758140667677
Steg/Kopf	(0.19680, 1.03778)	0.10679592032499
Kopf unten	(0.3132, 1.23581)	0.07005044877845
Kopf oben	(0.2655, 1.5198)	0.08019162478295

Eine beispielhafte Projektionsroutine in ALBERT sieht damit wie folgt aus:

```
1   static void ball_project1(REAL_D p){
2     FUNCNAME("ballproject1");
3     REAL norm;
4     int k;
5
6     const REAL   rad=0.08019162478295;
7     const REAL_D m={0.2655, 1.5198};
8
9     TEST_EXIT(p)("p=nil\n");
10
11    for (k=0;k<DIM_OF_WORLD;k++) p[k]-=m[k]; /* p=p-m */
12
13    norm = NORM_DOW(p);
14    norm = rad/norm;
15
16    for (k=0;k<DIM_OF_WORLD;k++)
17      p[k]=m[k]+norm*p[k]; /* p=m+Skalierung*Richtung*/
18
19    return;
20  }
```

Anhang D

Referenzdaten

Die hier aufgeführten Rechnungen wurden auf einer Sun Ultra 10 Elite3D mit 768MB Hauptspeicher und einer Utlrasparc II CPU mit 440 MHz durchgeführt. Die Laufzeiten ermittelten MATLABS `tic/toc`–Befehle. Für die Darstellungen in den Grafiken gilt wieder, daß die Isothermen in 10°C Abstand gewählt sind. Die im Abschnitt D.2 dargestellten Ergebnisse beziehen sich auf Rechnungen mit nichtlinearen Koeffizienten. Diese wurden ebenso wie die Rechnung im darauffolgenden Abschnitt mit fünf Zeitschritten zwischen den Aufdatierungen der Kühlparameter gerechnet. Es wird an den Ergebnissen auch noch einmal deutlich, daß die gemischte Randbedingung offenbar zur Steuerung der vorliegenden Wärmeleitungsgleichung nicht geeignet ist.

D.1 Zehn Minuten Abkühlung an Luft

Hier werden die Daten zu einer Rechnung aufgeführt, die eine zehn minütige Abkühlung an Luft wiedergibt. Die Rechnung wurde auf einem dreifach global verfeinerten Gitter durchgeführt und dauerte 7 Minuten und 18 Sekunden. Diese Rechnung verwendete eine etwas gröbere Toleranz (und daraus resultierend größere Zeitschritte), als die Referenzrechnungen in Kapitel 4, da der Schwerpunkt hier auf das qualitative Aussehen der Temperaturverteilung bei langer Kühldauer gelegt war. In den Rechnungen im Kapitel 4 ging es insbesondere auch darum, eine Referenz für die quantitative Qualität der Ergebnisse auf den gröberen Gittern zu erhalten. Zu beachten ist außerdem, das das Ergebnis weniger als Abbild der Realität, sondern vielmehr als Verhalten des Modells angesehen werden muß. Besonders durch die Linearisierung, aber auch durch das vernachlässigen der Phasenumwandlungsprozesse, das Modell in Temperturbereichen, die so weit unter 700°C liegen, wie es hier der Fall ist, nur noch sehr eingeschränkt gültig ist.

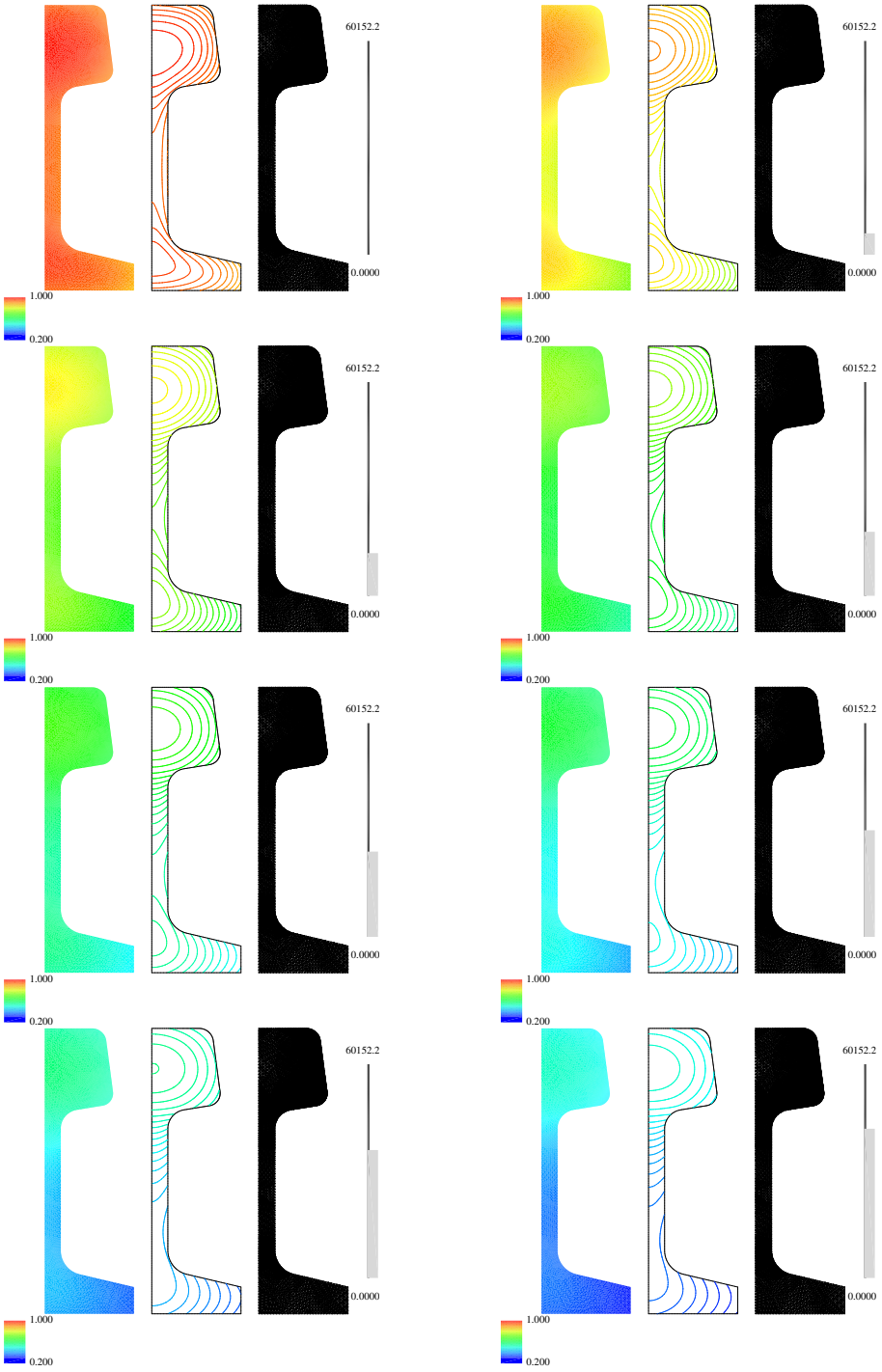


Abbildung D.1: Temperaturverteilung und Gitter zur Referenzrechnung mit konstanten Koeffizienten auf dreifach global verfeinertem Gitter. Zwischenergebnisse bei $t = 0, 1, 2, 3, 4, 5, 6, 7$ Minuten

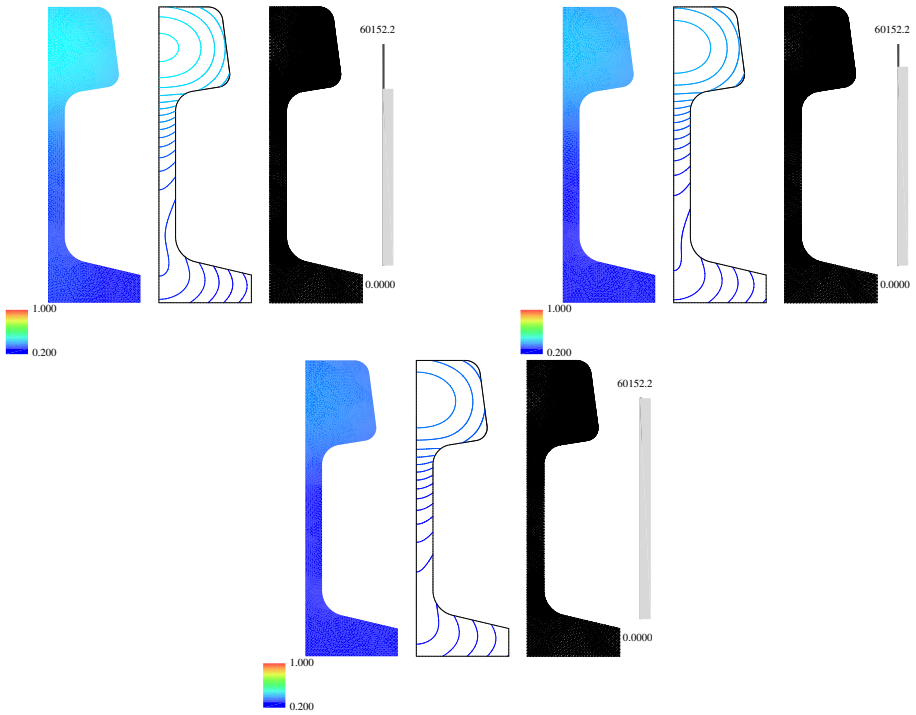


Abbildung D.2: Temperaturverteilung und Gitter zur Referenzrechnung mit konstanten Koeffizienten auf dreifach global verfeinertem Gitter bei $t = 8,9$Minuten am Ende der zehn minütigen Abkühlung an Luft.

D.2 Vergleichsrechnungen mit nichtlinearen Koeffizienten

In den vorderen Kapiteln der Arbeit wurde angedeutet, daß das in dieser Arbeit erstellte Programm auch mit nichtlinearen Koeffizienten umgehen kann. Diese werden am Anfang jedes Zeitschritte mit der Lösung aus dem vorhergehenden Zeitschritt gemäß der Gleichungen aus 1 aufdatiert. Während des Zeitschrittes werden sie dann aber genau wie die linearen Koeffizienten behandelt. Im Gegensatz zu den linearen sind die so behandelten Koeffizienten aber ortsabhängig und passen genauer zu der gerade vorliegenden Temperaturverteilung. Die Ergebnisse sind hier nur der Vollständigkeit halber aufgeführt. Sie konnten aus Zeitgründen nicht mehr ausführlich diskutiert werden.

Ausgang	max. Temp.	mn. Temp	Endzeit	# Zeitschritte	Rechenzeit
C_{60}	921.47	679.58	45.4s	28	6Min 43.93s
C_{temp}	921.47	679.52	45.4s	28	8Min 14.41s
C_{diff}	921.50	679.66	45.4s	28	11Min 52.95s

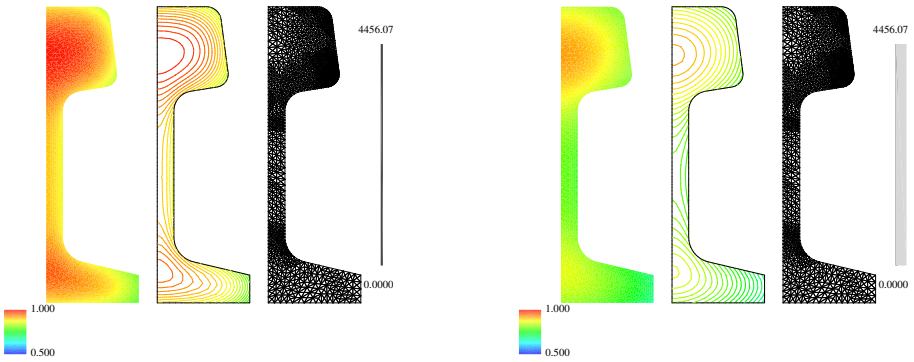


Abbildung D.3: Vergleichsrechnung mit nichtlinearen Koeffizienten und gemischten Randbedingungen zu $Q = R = I$ und $C = C_{60}$ im PDE–Zugang mit Aufdatierung der Kontrollparameter nach jedem fünften Zeitschritt.

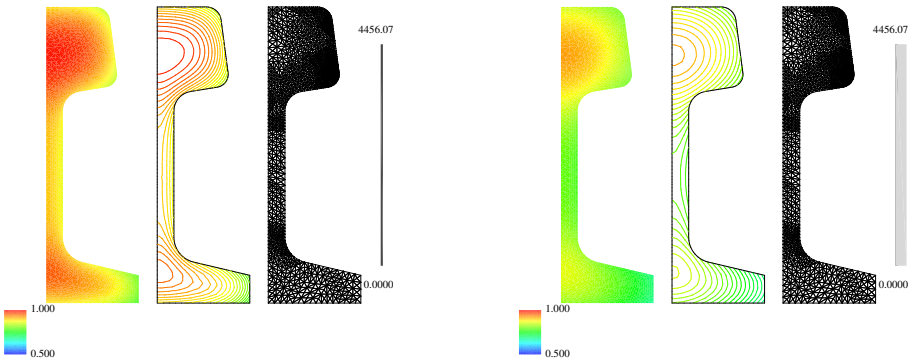


Abbildung D.4: Vergleichsrechnung mit nichtlinearen Koeffizienten und gemischten Randbedingungen zu $Q = R = I$ und $C = C_{diff}$ im PDE–Zugang mit Aufdatierung der Kontrollparameter nach jedem fünften Zeitschritt.

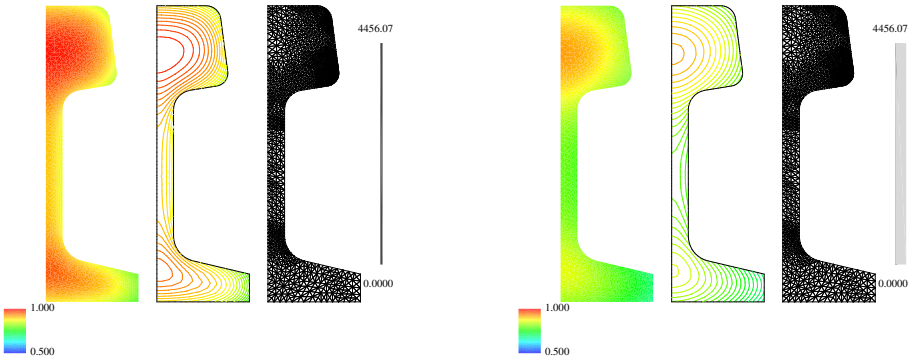


Abbildung D.5: Vergleichsrechnung mit nichtlinearen Koeffizienten und gemischten Randbedingungen zu $Q = R = I$ und $C = C_{temp}$ im PDE–Zugang mit Aufdatierung der Kontrollparameter nach jedem fünften Zeitschritt.

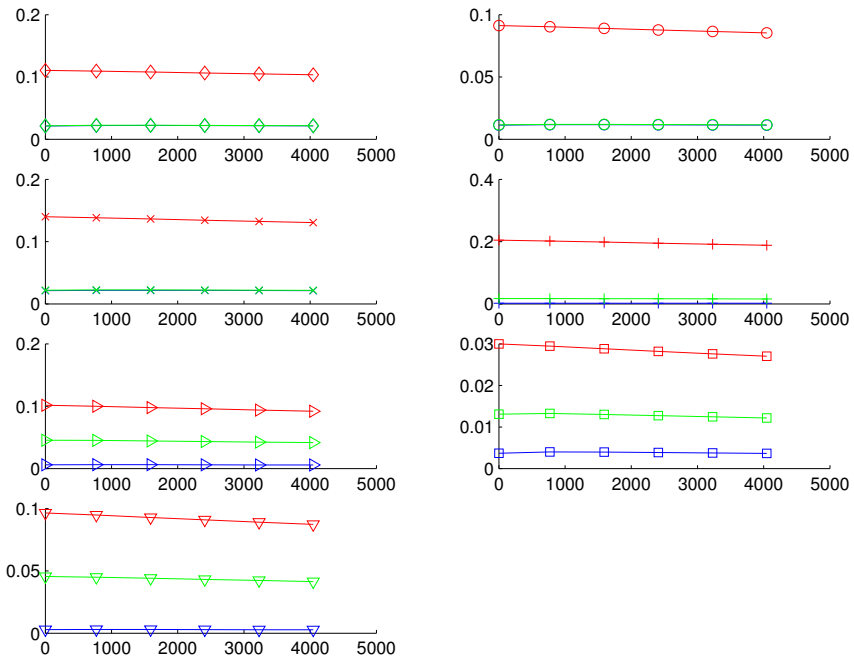


Abbildung D.6: Die Kontrollparameter zur Rechnung mit nichtlinearen Koeffizienten als Funktionen der Zeit.

D.3 Vergleichsrechnungen mit verändertem Parameter γ in der gemischten Randbedingung

Die hier aufgeführten Daten sollen die in Kapitel 4 aufgestellte Behauptung untermauern, daß das System sehr viel stärker auf Änderungen in γ reagiert, als auf Änderungen in den Kühlparametern. Die Daten sprechen dabei für sich und sind bereits in Kapitel 4 diskutiert worden.

Ausgang	max. Temp.	mn. Temp	Endzeit	# Zeitschritte	Rechenzeit
C_{diff}	930.23	773.40	45.5s	20	6Min 50.48s

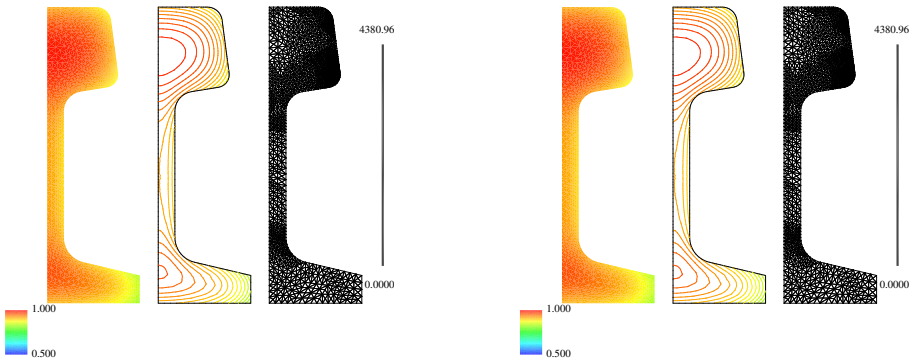


Abbildung D.7: Vergleichsdaten mit $\gamma = 1.32$ im Gegensatz zu $\gamma = 2.64$ in Abbildung 4.6

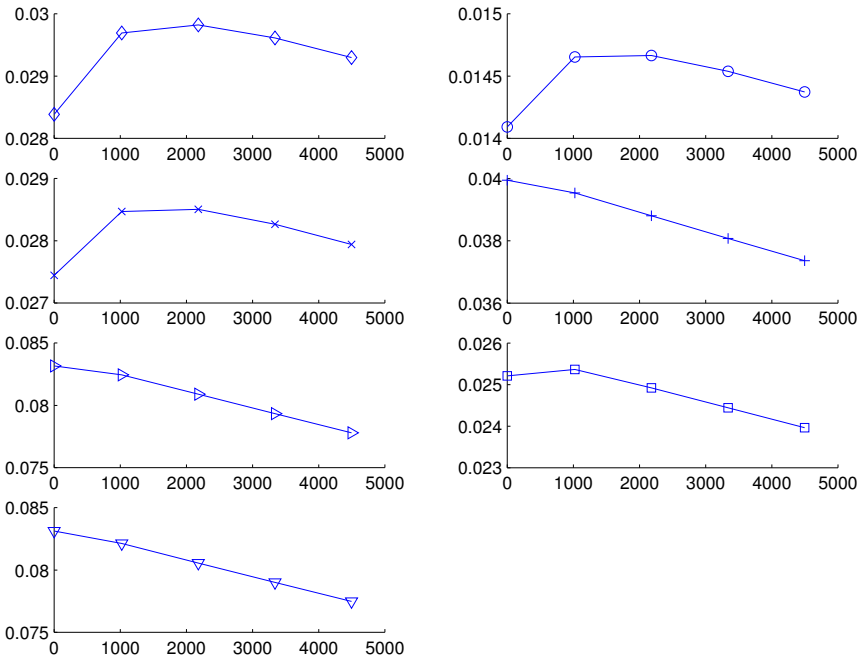


Abbildung D.8: Die Kontrollparameter als Funktionen der Zeit zur Rechnung mit $\gamma = 1.32$ im Vergleich zu $\gamma = 2.64$ in Abbildung 4.7

Literaturverzeichnis

[1] Alt, Hans W.: *Lineare Funktionalanalysis*. 3. Auflage. Springer-Verlag Berlin Heidelberg, 1999 8, 10, 25, 27

[2] Banks, H. T. ; Kunisch, Karl: The Linear Regulator Problem for Parabolic Systems. In: *SIAM J. Cont. Optim.* Bd. 22. Society for Industrial and Applied Mathematics, 1984, S. 684–698 7, 20, 28, 30

[3] Benner, Peter: Computaional methods for linear-quadratic optimization, Report 98-04 / Zentrum für Technomathematik. Universität Bremen, 1998. – Forschungsbericht. http://www.math.uni-bremen.de/zetem/reports/reports-psgz/report9804.ps.gz 65, 66, 72

[4] Benner, Peter ; Li, Jing-Rebecca ; Penzl, Thil. *Numerical solutuion of large Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems*. February 2002 76

[5] Böhm, M. ; Wolff, M. ; Bänsch, E. ; Davis, D.: Modellierung der Abkühlung von Stahlbrammen, Berichte aus der Technomathematik, Bericht 00-07 / Zentrum für Technomathematik. Universität Bremen, 2000. – Forschungsbericht. http://www.math.uni-bremen.de/zetem/reports/reports-psgz/report0007.ps.gz 15, 18, 19, 47

[6] Braess, Dietrich: *Finite Elemente - Theorie, schnelle Löser und Anwendung in der Elastizitätstheorie*. 1. Auflage. Springer-Verlag Berlin Heidelberg, 1992 25, 26

[7] Casti, John L.: *Dynamical systems and their applications: linear theory.* first edition. Mathematics in Science and Engineering, Vol. 135. New York - London: Academic Press (Harcourt Brace Jovanovich, Publishers). XV, 240 p. , 1977 66

[8] Crouch, P.E. ; Hinrichsen, D. ; Pritchard, A.J. ; e.a., D. S.: *Introduction to Mathematical System Theory*. 1988. – Lecture notes for a joint course at the Universities of Warwick and Bremen 65, 66

[9] Dunford, Nelson ; Schwartz, Jacob T.: *Linear Operators*. Bd. 1: *General Theory*. first edition. Interscience Publishers, Inc. New York, 1957 11, 20, 21

[10] Engel, Klaus-Jochen ; Nagel, Rainer: *One-parameter semigroups for linear evolution equations.* Graduate Texts in Mathematics. 194. Berlin: Springer., 2000 13, 20, 22

[11] Fujita, H. ; Mizutani, A.: On the finite element method for parabolic equations: Approximation of holomorphic semigroups. In: *Journal of Math. Soc. Japan* Bd. 28. Mathematical Society Japan, 1976, S. 749–770 29

[12] Gibson, J.S.: The Riccati integral equation for optimal control problems on Hilbert spaces. In: *SIAM J. Cont. Optim.* Bd. 17. Society for Industrial and Applied Mathematics, 1979, S. 537–565 7, 23

[13] Hille, Einar ; Phillips, Ralph S.: *Functional analysis and semi-groups. 3rd printing of rev. ed. of 1957.* Colloquium Publications, 31. Providence, R.I.: American Mathematical Society (AMS). XII, 1974. – available online: http://www.ams.org/online_bks/coll31/ 11

[14] Krahl, Rolf: *Das Crouzeix–Raviart–Element bei der numerischen Lösung der Navier–Stokes–Gleichung mit FEM*. Oktober 2002. – Diplomarbeit an der Universität Bremen, http://www.wias-berlin.de/people/krahl/pub/diplom.pdf 25, 26, 36

[15] Krengel, R. ; Standke, R. ; Tröltzsch, F. ; Wehage, H.: Mathematisches Modell einer optimal gesteuerten Abkühlung von Profilstählen in Kühlstrecken / Fakultät für Mathematik TU Chemnitz. 1997. – Forschungsbericht. Preprint im SFB 393 5, 14, 18, 49, 51

[16] Lancaster, Peter ; Rodman, Leiba: *Algebraic Riccati Equations*. 1. Oxford Science Publications, 1995 72

[17] Li, Jing-Rebecca ; White, Jacob: Low rank solutuion of Lyapunov Equations. In: *SIAM Journal on Matrix Analysis and Applications* 24 (2002), Nr. 1, S. 260–280 75, 76

[18] Luenberger, David G.: *Introduction to dynamic systems. Theory, models, and applications.* first edition. New York etc.: John Wiley & Sons. XIV, 446 p., 1979 66

[19] McBride, A.C.: *Semigroups of linear operators: an introduction*. first edititon. Longman Scientific & Techical, 1987 20

[20] Mehrmann, Volker: *Numerische Methoden in der Steuerungstheorie*. 1995. – Vorlesungsskript TU Chemnitz-Zwickau, http://ftp.tu-chemnitz.de/ftp-home/pub/Local/mathematik/Skripte/Mehrmann/nmst/vorles.ps.gz 65, 66, 72

[21] Pazy, A.: *Semigroups of linear operators and applications to partial differential equations.* Applied Mathematical Sciences, 44. New York etc.: Springer-Verlag. VIII, 1983 20, 22, 27, 29

[22] Penzl, Thilo: *LYAPACK: A MATLAB Toolbox for Large Lyapunov and Riccati Equation, Model Reduction Problems, and Linear-Quadratic Optimal Control Problems*. Version 1.0. TU Chemnitz: SFB 393 Fakultät für Mathematik, 1999. – http://www.tu-chemnitz.de/sfb393/lyapack/guide.pdf 40, 72, 75, 76

[23] Schmidt, A. ; Siebert, K.: *ALBERT: An adaptive hierarchical finite element toolbox*. Edition: ALBERT-1.0. Albert-Ludwigs-Universität Freiburg: Preprint 06/2000 / Institut für Angewandte Mathematik, 2000. – http://www.mathematik.uni-freiburg.de/IAM/ALBERT/doc.html 16, 36

[24] Schmidt, Alfred ; Siebert, Kunibert G.: ALBERT - Software for scientific computations and application. In: *Acta mathematica Universitatis Comenianae* 70 (2001), Nr. 1, S. 105–122 16, 36

[25] Tröltzsch, F. ; Unger, A.: Fast solution of optimal control problems in the selective cooling of steel. In: *ZAMM Z. Angew. Math. Mech.* 81 (2001), Nr. 7, S. 447–456 16, 17

[26] Wachspress, Eugene L.: *The ADI model problem*. first edition. E.L. Wachspress, 1995 73, 76

[27] Wachspress, Eugene L. *ADI iteration parameters for the Sylvester equation*. August 2000 76

[28] Werner, Dirk: *Funktionalanalysis. (Functional analysis). 3., neu bearb. und erweiterte Aufl.* Springer-Lehrbuch. Berlin: Springer., 2000 13, 25

[29] Yosida, Kosaku: *Functional Analysis*. 5. edition. Springer-Verlag Berlin Heidelberg New York, 1978 11, 20

Index