Universität Bayreuth
Fakultät für Mathematik und Physik

Diplomarbeit

Numerische Berechnung der optimalen Wertefunktion mit Approximation zweiter Ordnung

STEFANIE ÖHRLEIN

17. Januar 2007

Betreut durch
PROF. DR. LARS GRÜNE

Inhaltsverzeichnis

1 Einleitung **1**

2 Theorie der Verfahren **5**
2.1 Theoretische Grundlagen 5
2.1.1 Diskrete stochastische Prozesse 6
2.1.2 Theorie der Kontrollsysteme 8
2.1.3 Notwendige Optimalitätsbedingungen 10
2.2 Das Perturbations-Modell 12
2.2.1 Das Modell für die Kontrollfunktion und die Dynamik 14
2.2.2 Das Modell für die Wertefunktion 15
2.3 Die Fixpunkt-Iteration 18
2.3.1 Die Iterationsvorschrift 18
2.3.2 Konvergenz der Fixpunkt-Iteration 21

3 Herleitung der Koeffizienten für die Perturbations-Methoden **23**
3.1 Koeffizienten für die Kontrollfunktion und die Dynamik 23
3.1.1 Lineare Approximation 23
3.1.2 Quadratische Approximation 25
3.1.3 Approximation höherer Ordnung 26
3.2 Koeffizienten für die Wertefunktion 27
3.2.1 Lineare Approximation 27
3.2.2 Quadratische Approximation 30
3.2.3 Approximation höherer Ordnung 34

4 Implementierung der Verfahren **37**
4.1 Implementierung der Perturbations-Methode der Kontrollfunktion . . . 37
4.2 Implementierung der Perturbations-Methode der Wertefunktion 38
4.2.1 Implementierung im Programm compute_Vcoeff.mw 39
4.2.2 Auswertung anhand des Programms show51.m 41
4.3 Implementierung der Fixpunkt-Iteration 42
4.3.1 Speicherung im Programm compute.m 42
4.3.2 Implementierung im Programm fixed_point.cpp 43
4.3.3 Auswertung anhand des Programms show51.m 45

5 Anwendung der Verfahren auf das Modell-Beispiel **47**
5.1 Darstellung des Modells 47
5.1.1 Exakte Lösung des Grundmodells 47
5.1.2 Berechnung des Gleichgewichtspunktes 49
5.2 Auswertung des Modells 51
5.2.1 Numerische Approximation für die Kontrollfunktion 51
5.2.2 Numerische Approximation für die Wertefunktion 54
5.2.3 Auswertung der Approximationen 57
5.3 Abwandlung des Modells 70
5.3.1 Numerische Approximationen 71
5.3.2 Auswertung für $\kappa = 2$ 74
5.3.3 Auswertung für unterschiedliche κ und Standardabweichungen 81

6 Fazit der praktischen Ergebnisse und Ausblick **93**

A Notation **97**

B Matlab-Quelltexte **99**
B.1 Die Routine compute_Vcoeff.mw 99
B.2 Die Routine compute.m 104
B.3 Die Routine fixed_point.cpp 113
B.4 Die Routine show51.m 125

C Material auf der beiliegenden CD **147**

Literaturverzeichnis **149**

Abbildungsverzeichnis

5.1 Die exakte Wertefunktion V(x) 49
5.2 Fehler der Perturbations-Methode für die Kontrollfunktion 58
5.3 Fehler der Perturbations-Methode für die Wertefunktion 59
5.4 Fehler der Fixpunkt-Iteration 59
5.5 Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration . . . 61
5.6 Exakte Wertefunktion und Perturbations-Methode 64
5.7 Exakte Wertefunktion und Fixpunkt-Iteration 64
5.8 Fehler der Perturbations-Methode für die Wertefunktion 65
5.9 Fehler der Fixpunkt-Iteration 66
5.10 Fehler der Fixpunkt-Iteration 69
5.11 Fehler der Perturbations-Methode für die Kontrollfunktion 70
5.12 Perturbations-Methode für die Wertefunktion 73
5.13 Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration . . . 75
5.14 Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration . . . 78
5.15 Perturbations-Methode für die Wertefunktion zu unterschiedlichen κ . . 82
5.16 Fixpunkt-Iteration zu unterschiedlichen κ 83
5.17 Differenz beider Approximationen unter 0.1, $\kappa = 1$ 85
5.18 Differenz beider Approximationen unter 0.1, $\kappa = 10$ 85
5.19 Differenz beider Approximationen unter 0.1, $\kappa = 15$ 86
5.20 Differenz beider Approximationen unter 0.1, $\kappa = 15$ 87
5.21 Differenz beider Approximationen unter 0.1, $\kappa = 5$, $\sigma = 0.012$ 89
5.22 Differenz beider Approximationen unter 0.1, $\kappa = 1$, $\sigma = 0.01$ 89
5.23 Differenz beider Approximationen unter 0.1, $\kappa = 1$, $\sigma = 0.012$ 90
5.24 Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration, $\kappa = 15$ 91
5.25 Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration, $\kappa = 1$, $\sigma = 0.008$ 91
5.26 Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration, $\kappa = 1$, $\sigma = 0.012$ 92

Tabellenverzeichnis

4.1 Bezeichnungen im Programm compute_Vcoeff.mw 41

5.1 Fehler von V_pert und V_fix auf unterschiedlichen Gebietsabschnitten, $\kappa = 0$ 62
5.2 Fehler von V_pert und V_fix auf unterschiedlichen Gebietsabschnitten, $\kappa = 0$ 67
5.3 Fehler von V_pert und V_fix auf unterschiedlichen Gebietsabschnitten, $\kappa = 0$, 251 Gitterpunkte 68
5.4 Differenz zwischen V_pert und V_fix auf unterschiedlichen Gebietsabschnitten, $\kappa = 2$ 76
5.5 Differenz zwischen V_pert und V_fix auf unterschiedlichen Gebietsabschnitten, $\kappa = 2$ 79
5.6 Differenz zwischen V_pert und V_fix auf unterschiedlichen Gebietsabschnitten, $\kappa = 2$, 251 Gitterpunkte 80

1 Einleitung

In der heutigen Makroökonomie wird häufig eine Modell-Wirtschaft verwendet, um die Auswirkungen der Wirtschaftspolitik oder Änderungen in der Marktstruktur zu untersuchen. Leider ist es meist unmöglich, dies analytisch zu tun, weshalb man sich numerischer Methoden bedient.
Da das Interesse daran steigt, das Risiko in die Betrachtungen einzubeziehen oder die Beziehungen zwischen steigender Wohlfahrt und beispielsweise Konsumverhalten der Wirtschaftssubjekte zu untersuchen, werden nach COLLARD und JUILLARD [7] die Modelle der Makroökonomie zunehmend komplexer. Deshalb wurden in den letzten Jahren eine Reihe von Verfahren entwickelt, die dem Rechnung tragen sollen.
Hierzu zählen unter anderem die so genannten Perturbations-Methoden. Nach JUDD [16] liegt die Idee darin, für ein gegebenes Problem einen Spezialfall zu finden, für den die Lösung bekannt ist. Dieser Fall und die zugehörige Lösung werden dann verwendet, um Lösungen für Probleme zu finden, die nur geringfügig vom Spezialfall abweichen.
In den Wirtschaftswissenschaften finden dabei meist bestimmte Perturbations-Methoden, die Linearisierungs-Methoden, Verwendung, die lediglich eine lineare Taylor-Entwicklung zur Approximation zu Grunde legen. Leider weist die Linearisierung etliche Nachteile auf, die sich nach SCHMITT-GROHÉ [25] unter anderem bei der Bewertung der Wohlfahrt bemerkbar machen. Warum derartige Methoden dennoch sehr häufig herangezogen werden, liegt daran, dass man die Berechnung von Taylor-Entwicklungen höherer Ordnung als wesentlich aufwändiger glaubte. Hierin belehren JIN und JUDD [14] eines besseren, denn,

> Contrary to conventional wisdom, the higher-order terms are conceptually no more difficult to compute than the conventional deterministic linear approximations.

Da man zudem an Gleichgewichten innerhalb der Modell-Wirtschaft interessiert ist, und die Wirtschaftssubjekte in der Lage sind ihr Verhalten zu steuern, um ein für sie optimales Gleichgewicht zu erreichen, beschäftigen wir uns im Folgenden mit Perturbations-Methoden für Kontrollsysteme beziehungsweise für optimale Steuerung.

In der vorliegenden Arbeit werden zwei Verfahren aus dem Bereich der Perturbations-Methoden beschrieben. Eine in diesem Rahmen entstandene Dipomarbeit von BECKER [2] stellt die Grundlage dar. Sie beinhaltet die Herleitung eines der Verfahren zur numerischen Berechnung der Kontrolle sowie der Dynamik. Da einige Approximationen hierbei sehr ungenau werden, diskutiert die folgende Arbeit weitere Möglichkeiten, die numerisch ermittelten Werte zu verbessern.
Die so erhaltenen Approximationen werden analog zur Arbeit von BECKER [2] an einem volkswirtschaftlichen Beispiel getestet. Da hier für einen speziellen Fall eine exakte Lösung berechnet werden kann, sind direkte Vergleiche bezüglich der Genauigkeit der numerischen Lösungen möglich. Zusätzlich sollen aber auch alternative Fälle betrachtet und hierfür mögliche Schlüsse aus den vorherigen Resultaten gezogen werden.

Die Arbeit ist wie folgt aufgebaut:
Das nächste Kapitel nimmt Bezug auf die theoretischen Grundlagen der verwendeten Verfahren. Dabei werden zunächst die notwendigen stochastischen Begriffe, die Theorie von Kontrollsystemen und die Optimalitätsbedingungen eingeführt. Anschließend folgt die Vorstellung des Perturbations-Modells für die Kontrolle und die Dynamik nach SCHMITT-GROHÉ und URIBE, was ausführlich in BECKER [2] beschrieben wird und auch maßgeblich als Referenzliteratur verwendet wurde. Zur numerischen Berechnung der Wertefunktion wird ein Perturbations-Verfahren nach COLLARD und JUILLARD [6] verwendet, dessen theoretisches Vorgehen hier entsprechend aufgeführt wird. Eine zweite Möglichkeit zur Approximation der Wertefunktion benötigt die Grundlagen der Fixpunkt-Iteration.

Da die Perturbations-Methoden stets mit Taylor-Approximationen arbeiten, ist eine Herleitung der dabei auftretenden Koeffizienten nötig. Dies geschieht im dritten Kapitel separat für die Kontrolle und die Wertefunktion. In einem ersten Schritt werden die linearen Terme für die Entwicklung der Kontrolle und der Dynamik ermittelt und in einem weiteren die quadratischen Terme. Eine ausführliche Diskussion findet sich wieder in BECKER [2]. Für die Wertefunktion wird zunächst das Prinzip der hier verwendeten Perturbations-Methode von COLLARD und JUILLARD anhand der linearen Approximation beschrieben. Dass sich eine quadratische Approximation auf gleiche einfache Weise berechnen lässt, wird im darauffolgenden Abschnitt aufgezeigt. Zudem kann die Vorgehensweise leicht auf Approximationen höherer Ordnung übertragen werden, worauf kurz in einem weiteren Abschnitt eingegangen wird.

Für eine spätere Auswertung der numerischen Approximationen wurden die angesprochenen Verfahren implementiert. Die Beschreibung der Programme findet im vierten Kapitel statt. Hierbei wird gänzlich auf eine Ausführung der Routinen verzichtet, die die Perturbations-Methode für die Kontrolle und die Dynamik realisieren. Es sei lediglich auf die Arbeit von BECKER [2] verwiesen, die sich vollständig auf diese Perturbations-Methode konzentriert. Für die Taylor-Entwicklung der Wertefunktion wurde eine Routine in Maple geschrieben, die die notwendigen Koeffizienten ermittelt. Sie stützt sich dabei auf die im vorherigen Kapitel hergeleitete Theorie. Ein weiteres Programm in C++ realisiert die zweite Approximation der Wertefunktion durch die Fixpunkt-Iteration. Da hier als Grundlage die approximative Kontrolle und Wertefunktion benötigt werden, erfolgt deren Speicherung durch eine Matlab-Routine. Zur Auswertung der numerisch erhaltenen Werte existiert ein weiteres Programm in Matlab, deren Beschreibung sich ebenfalls in diesem Kapitel findet.

Anhand eines Modell-Beispiels aus der Volkswirtschaft soll im fünften Kapitel die Genauigkeit und das Verhalten der verwendeten Verfahren getestet werden. Hierbei wird zunächst auf einen Spezialfall des Modells eingegangen. Da für diesen Fall eine exakte Lösung bekannt ist, kann sie berechnet und anschließend die Approximationen di-

rekt damit verglichen werden. Die dadurch erhaltenen Aussagen über die Genauigkeit der verwendeten Methoden sollen Aufschluss über die abgeänderte Version des Modell-Beispiels, die stärker auf die wirtschaftliche Unsicherheit eingeht, geben. Es wird dabei versucht, Parallelen im Verhalten der Näherungs-Lösungen zu ziehen und somit die vorherigen Ergebnisse auf das erweiterte Modell zu übertragen. Ebenso wird das Verhalten bei Variation der stochastischen Größen untersucht.

Im letzten Kapitel werden die erhaltenen Resultate nochmals aufgegriffen. Dabei wird gegebenenfalls ein Ausblick auf Verbesserungen und eine Wertung über die Verwendbarkeit des Verfahrens in der Praxis gegeben.

An dieser Stelle möchte ich mich bei Herrn Prof. Dr. Grüne für seine engagierte und lehrreiche Betreuung und die ständige Bereitschaft zur Diskussion und Hilfe während meiner Diplomarbeit bedanken. Die kritischen und konstruktiven Gespräche und die interessante Themenstellung waren immer Ansporn und Inspiration.
Außerdem möchte ich all denen danken, die mich während der Entstehungszeit dieser Arbeit begleitet und unterstützt haben.

2 Theorie der Verfahren

Zur approximativen Berechnung der Kontrolle und der Dynamik wird, wie bereits erwähnt, das in BECKER [2] beschriebene Perturbations-Verfahren verwendet. Die wichtigsten mathematischen Voraussetzungen des Verfahrens werden im Folgenden beschrieben. Ebenso werden die Grundlagen für die Perturbations-Methode und die Fixpunkt-Iteration zur Approximation der Wertefunktion erläutert.
Zunächst werden diskrete stochastische Prozesse als Grundlage für das Perturbations-Verfahren der Kontrolle u und der Dynamik x sowie der Wertefunktion V definiert. Des Weiteren führen wir Kontrollsysteme zur Erläuterung der Begriffe Kontrolle und Wertefunktion ein. Schließlich sind noch Optimatiltätsbedingungen notwendig, die für die Perturbations-Methode der Kontrolle hergeleitet werden.
Im zweiten Abschnitt werden die Approximation der Kontrolle und der Dynamik nach SCHMITT-GROHÉ und URIBE und der Wertefunktion nach COLLARD und JUILLARD beschrieben. Wenn die Verfahren sich auch unterscheiden, so verwenden beide als Grundlage eine Taylor-Entwicklung, wie wir in diesem Abschnitt sehen werden.
Zuletzt wird die Theorie von Fixpunkt-Iterationen erläutert. Hierbei wird die verwendete Iterationsvorschrift hergeleitet und ein Beweis der Konvergenz und der Eindeutigkeit der dadurch erhaltenen Lösung gegeben.

2.1 Theoretische Grundlagen

In den Wirtschaftswissenschaften werden häufig Linearisierungs-Methoden zur Lösung von Optimierungsproblemen verwendet, vor allem dann, wenn noch keine Informationen über die Lösung vorhanden sind. Das Potenzial von Perturbations-Methoden liegt laut COLLARD und JUILLARD [6] darin, dass sie einerseits genauso einfach sind wie die Linearisierungs-Methoden, andererseits aber deren größten Nachteil ausgleichen. So

sind Perturbations-Methoden im Gegensatz zur Linearisierung in der Lage, auf hohe Standardabweichung bei der Zufallsvariablen und starke Krümmung der Nutzenfunktion zu reagieren.

2.1.1 Diskrete stochastische Prozesse

In den meisten wirtschaftlichen Modellen sollte der Faktor Unsicherheit nicht vernachlässigt werden. So können sich nach COLLARD und JUILLARD [6] neben der Wirtschaftspolitik auch andere Gegebenheiten im Umfeld der Wirtschaftssubjekte ändern, wie beispielsweise ein Wandel der Marktstruktur. Werden derartige Auswirkungen auf das System bei den Auswertungen vernachlässigt, kann dies leicht zu falschen Ergebnissen führen.
Ein Teil des Zustandsvektors x_t soll deshalb einer stochastischen Dynamik folgen, um die Unsicherheit wiederzugeben. Um dies mathematisch zu fassen, werden einige theoretische Gegebenheiten aus der Stochastik benötigt. Bezüglich ausführlicher Grundlagen der Maßtheorie sei auf CAPIŃSKI und KOPP [5] und ELSTRODT [8] verwiesen.
Zunächst erfolgt die Definition des Wahrscheinlichkeitsraumes.

Definition 2.1 *Ein Tripel $(\Omega, \mathcal{A}, P)$ heißt Wahrscheinlichkeitsraum, wobei Ω der Ereignisraum ist, $\mathcal{A}$ eine entsprechende $\sigma-$Algebra und P ein geeignetes Wahrscheinlichkeitsmaß.*

Für die betrachteten Verfahren ist außerdem der Erwartungswert unabdingbar.

Definition 2.2 *Sei X eine reelle Zufallsvariable auf $(\Omega, \mathcal{A}, P)$. Die Verteilung von $\mathbf{X}$ sei diskret, also getragen von einer abzählbaren Wertemenge $\{x_i | i \in I\}$. Falls die Reihe*

$$\sum_{i \in I} x_i \cdot P(\mathbf{X} = x_i)$$

absolut konvergent ist, so sagt man, der Erwartungswert $E[X]$ existiert und man setzt

$$E[X] := \sum_{i \in I} x_i \cdot P(X = x_i).$$

Zusätzlich benötigen wir die Definition der Varianz beziehungsweise der Standardabweichung.

Definition 2.3 *Es sei X eine reelle Zufallsvariable auf $(\Omega, \mathcal{A}, P)$ mit existierendem zweiten Moment, d.h. $E[X^2] < \infty$. Dann existieren auch die Erwartungswerte $E[X]$ und $E[(X - E[X])^2] =: Var[X]$ und letzterer heißt Varianz von X. Der Parameter $\sigma = \sqrt{Var[X]}$ wird als Standardabweichung oder Streuung bezeichnet.*

Wie bereits erwähnt, soll ein Teil des Zustandsvektors einer stochastischen Dynamik folgen. Das mathematische Pendant hierzu ist der stochastische Prozess. Wir beschränken uns dabei auf den diskreten Fall, da für die numerische Realisierung besonders dieser von Interesse ist.
Ein stochastischer Prozess entspricht einer gewöhnlichen Differenzengleichung, erweitert um einen so genannten Störungsterm, der ausschließlich vom aktuellen Zustand abhängt. Mathematisch erhält man:

Definition 2.4 *Ein stochastischer Prozess für Systeme in diskreter Zeit kann durch eine stochastische Differenzengleichung ausgedrückt werden:*

$$x_{t+1} = \zeta(x_t, t) + z(x_t, t),\ t \in \mathbf{T}, \tag{2.1}$$

mit $\mathbf{T} := \mathbb{N}_0 := \{k | k \in \mathbb{N}_0\}$.
Hierbei ist x_{t+1} eine Zufallsvariable, die nur von x_t und t abhängt, ζ ist der bedingte Mittelwert von x_{t+1}, gegeben x_t, und $z(.)$ ist eine d-dimensionale Zufallsvariable mit Mittelwert Null, deren bedingte Verteilung bei gegebenem x_t nicht von x_s für $s < t$ abhängt.

Bemerkung 2.5 *Bei der obigen Definition wird von einer exogenen Störung ausgegangen, also nicht von einer Störung durch Messfehler, bei der die Zustandsvariablen nicht genau beobachtet werden können.*

Im Folgenden soll stets gelten, dass der Störungsterm $z(x_t) \sim N(\mu, \sigma)$-verteilt ist. Dadurch ist es möglich, ihn zu normalisieren. Mit $\frac{z(x_t)}{\sigma(x_t)}$ erhält man eine $N(0, 1)$-verteilte Zufallsvariable $\varepsilon_t, t \in \mathbf{T}$. Diese ist unabhängig von x und kann als Folge von unabhängigen und identisch $N(0, 1)$-verteilten Zufallsvariablen betrachtet werden.
Mit

$$z = \sigma(x_t) \cdot \varepsilon_t,$$

erhält man schließlich für den stochastischen Teil des Zustandsvektors (2.1):

$$x_{t+1} = \zeta(x_t) + \sigma(x_t) \cdot \varepsilon_t. \tag{2.2}$$

Für einen derartigen Zustandsvektor mit deterministischem und stochastischem Anteil und einer unabhängig und identisch $N(0,1)$-verteilten Zufallsvariablen wollen wir Kontrollsysteme einführen.

2.1.2 Theorie der Kontrollsysteme

Zur Klärung der Basis-Begriffe Kontrolle und Wertefunktion benötigen wir die Theorie von Konrollsystemen. Ein kurzer Einblick in die wichtigsten Definitionen und Sätze ist in diesem Abschnitt gegeben.

Bemerkung 2.6 *Die Begriffe „Steuerung“ und „Kontrolle“ werden im Folgenden synonym verwendet.*

Zunächst muss eingegrenzt werden, in welchem Bereich die Werte der Kontrolle liegen dürfen.

Definition 2.7 *Die nichtleere und kompakte Menge $\mathbf{U} \subset \mathbb{R}^m$ wird als Kontrollbereich festgelegt und beschreibt die zulässige Wertemenge für den Kontrollvektor u_t, d.h.*

$$u_t \in \mathbf{U}, \quad \forall\, t \in \mathbf{T}$$

mit $\mathbf{T} := \mathbb{N}_0 := \{k | k \in \mathbb{N}_0\}$.

Da wir ein volkswirtschaftliches Modell betrachten werden und hierbei die einzelnen Wirtschaftssubjekte in der Lage sind, ihr Verhalten zu steuern beziehungsweise zu kontrollieren, muss der Begriff des Kontrollsystems eingeführt werden:

Definition 2.8 *Ein Kontrollsystem in diskreter Zeit $\mathbf{T}$ im $\mathbb{R}^d$, $d \in \mathbb{N}$, ist gegeben durch die Differenzengleichung*

$$x_{t+1} = \varphi(x_t, u_t, z_t), \tag{2.3}$$

wobei $\varphi : \mathbb{R}^d \times \mathbf{U} \times \mathbb{R}^d \to \mathbb{R}^d$ eine stetige Abbildung und z_t eine unabhängig und identisch verteilte Zufallsvariable ist.

Unter bestimmten Voraussetzungen an das Kontrollsystem kann für jeden beliebigen Anfangswert eine eindeutige Lösung der Differenzengleichung (2.3) gefunden werden.

Definition 2.9 *Eine eindeutige Funktion x_t, die Lösung von (2.3) ist, zum Anfangswert $x_0 \in \mathbb{R}^d$, zur Kontrollfunktion $u_t \in \mathbf{U}$ und zur unabhängig und identisch verteilten Zufallsvariable $z_t \in \mathbb{R}^d$ wird mit $\Phi(t, x_0, u_t, z_t)$ bezeichnet .*

Durch Induktion lässt sich zeigen, dass zu jedem Anfangswert x_0, jeder Kontrolle $u_t \in \mathbf{U}$ und jeder Folge von Realisierungen eine eindeutige Lösung $\Phi(t, x_0, u_t, z_t)$, $\Phi : \mathbf{T} \times \mathbb{R}^d \times \mathbf{U} \times \mathbb{R}^d \to \mathbb{R}^d$, mit

$$\Phi(0, x_0, u_t, z_t) = x_0 \quad \text{und} \quad \Phi(t+1, x_0, u_t, z_t) = \varphi(\Phi(t, x_0, u_t, z_t), u_t, z_t)$$

existiert. Das bedeutet letztendlich, dass im diskreten Fall stets eine eindeutige Lösung vorhanden ist.
Ausgangspunkt der Perturbations-Methoden ist ein Gleichgewicht:

Definition 2.10 *Ein Zustand $\bar{x}$ aus einer Menge $\mathbf{X} \subset \mathbb{R}^d$ und eine Kontrolle $\bar{u} \in \mathbf{U}$ heißen Gleichgewicht eines Kontrollsystems φ, falls $\varphi(\bar{x}_t, \bar{u}_t, 0) = \bar{x}_t$.*

Nach den bisherigen Definitionen können unendlich viele Lösungen für ein Kontrollsystem gefunden werden. Unser Ziel ist dabei die optimale Lösung zu berechnen. Um den Begriff der Optimalität mathematisch zu fassen, definiert man eine Zielfunktion,

$$\psi : \mathbb{R}^d \times \mathbf{U} \to \mathbb{R},$$

die man entlang der Trajektorie Φ aufsummiert. Die Kontrolle $u_t \in \mathbf{U}$, für die die Summe maximal (beziehungsweise alternativ minimal) wird, bezeichnet die optimale Kontrolle. Man sagt, man löst das optimale Kontrollproblem:

Definition 2.11 *Betrachte ein Kontrollsystem, gegeben durch (2.3). Für eine Funktion $\psi : \mathbb{R}^d \times \mathbf{U} \to \mathbb{R}$, einen Parameter $\delta > 0, \delta \in \mathbb{R}$ und eine unabhängig und identisch verteilte Zufallsvariable $z_t \in \mathbb{R}^d$, definieren wir das diskontierte Funktional auf unendlichem Zeithorizont in diskreter Zeit als*

$$J(x_t, u_t, z_t) := \sum_{t=0}^{\infty} \beta^t \cdot \psi(\Phi(t, x_t, u_t, z_t), u_t) \tag{2.4}$$

mit $\beta := (1 - \delta)$.

Das optimale stochastische Kontrollproblem lautet dann: Bestimme die optimale Wertefunktion

$$V(x) := \sup_{u_t \in \mathbf{U}} E\left[J(x_t, u_t, z_t)\right]. \tag{2.5}$$

Dabei sei u_t *von* $z_0, \ldots, z_t$ *abhängig und es gelte*

(i) $\mathbf{U}$ *ist kompakt.*

(ii) Die Funktion ψ *sei stetig und erfülle*

$$|\psi(x, u)| \leq M_\psi \quad \textit{und} \quad |\psi(x_1, u) - \psi(x_2, u)| \leq L_\psi \|x_1 - x_2\|$$

für alle $x, x_1, x_2 \in \mathbb{R}^d$*, alle* $u \in \mathbf{U}$ *und geeignete Konstanten* $M_\psi, L_\psi > 0$*, mit* $M_\psi, L_\psi \in \mathbb{R}$.

Nachdem die Begriffe der Kontrolle und Wertefunktion nun geklärt sind, wollen wir uns mit den Optimalitätsbedingungen beschäftigen, die im Vorfeld für die Approximations-Methode der Kontrolle u und der Dynamik x benötigt werden.

2.1.3 Notwendige Optimalitätsbedingungen

Für die Perturbations-Verfahren ist es notwendig, bereits zu Beginn eine Lösung zu kennen. In unserem Fall stellt diese Lösung der Gleichgewichtspunkt dar. Die Idee besteht nun darin, die Lösung zu stören und somit Approximationen in deren Umgebung zu erhalten. Diverse Schwierigkeiten können dann auftreten, wenn zum Beispiel kein eindeutiges Gleichgewicht existiert, oder das System in eine periodische Lösung statt einen einzelnen Punkt läuft.

Da bei den verwendeten Verfahren zunächst das Gleichgewicht als Ausgangspunkt benötigt wird, ist eine Gleichgewichtsbedingung für allgemeine dynamische, stochastische Gleichgewichtsmodelle zu formulieren:

$$\boxed{E_t f(u_{t+1}, u_t, x_{t+1}, x_t) = 0 \qquad \forall\, t \in \mathbb{N}_0.} \tag{2.6}$$

Hierbei bezeichnet E_t den Erwartungswert in Abhängigkeit von allen zum Zeitpunkt t vorhandenen Informationen. Analog sind u_t und x_t der Kontrollvektor beziehungsweise der Zustandsvektor zum Zeitpunkt t. Bleibt noch eine Funktion f zu definieren, so dass die Gleichgewichtsbedingung erfüllt ist. Um das Verständnis der obigen Gleichung zu verbessern, erfolgt in diesem Abschnitt eine schrittweise Herleitung der Funktion f.
Die Optimalitätsbedingungen müssen dazu in eine Form gebracht werden, wie sie im Algorithmus von SCHMITT-GROHÉ und URIBE verwendet werden. Hierfür ist die so genannte Hamilton-Funktion nötig.

Definition 2.12 *Sei $\lambda_0 \in \mathbb{R}$ und $\lambda \in \mathbb{R}^n$. Dann heißt*

$$H^t(x_t, u_t, \lambda_t) := \lambda_0 \cdot \beta^t \psi(x_t, u_t) + \lambda_{t+1} \cdot \varphi(t, x_t, u_t) \tag{2.7}$$

die HAMILTON*-Funktion zum Kontrollproblem aus Definiton 2.11.*
Der Vektor $\lambda \in \mathbb{R}^n$ wird adjungierte Variable oder Kozustand genannt.

Bemerkung 2.13 *λ_0 kann unter zusätzlichen Bedingungen zu $\lambda_0 = 1$ gewählt werden und tritt daher später nicht in der* HAMILTON*-Funktion auf.*

Damit lassen sich nun die Optimalitätsbedingungen folgendermaßen darstellen:

Satz 2.14 ***(Diskretes Maximumprinzip)*** *Es sei u_t^* eine optimale Entscheidungsfolge und x_t^* die zugehörige Zustandsfolge. Dann existiert eine Kozustandsfolge λ_t, so dass gilt*

$$\boxed{\begin{aligned} x_{t+1}^* &= \left(\frac{\partial H^t}{\partial \lambda_{t+1}}\right) && \textit{Zustandsgleichung} \\ \lambda_t &= \left(\frac{\partial H^t}{\partial x_t^*}\right) && \textit{Kozustandsbedingung} \\ 0 &= \left(\frac{\partial H^t}{\partial u_t^*}\right) && \textit{Maximumbedingung} \end{aligned}} \tag{2.8}$$

Dies gilt allerdings nur, wenn keine Beschränkung der Steuervariablen und ein unendlicher Zeithorizont vorliegen. Im Allgemeinen reicht uns die Formulierung aber für das

verwendete Verfahren aus. Ist H zusätzlich konkav in (x, u), sind die obigen Bedingungen auch hinreichend. Ein Beweis des Satzes 2.14 und nähere Erläuterungen zur Herleitung der Gleichungen finden sich in FEICHTINGER und HARTL [9].

Bemerkung 2.15 *In dem untersuchten Verfahren wird zunächst $\sigma = 0$ gesetzt. Für die resultierende Situation ohne Störung von außen wird der Gleichgewichtspunkt berechnet. Der stochastische Anteil entfällt also und man betrachtet ein deterministisches Problem, weshalb im Gleichgewicht $E_t f = f = 0$ gilt.*

Aus dem diskreten Maximumprinzip erhält man schließlich die Funktion f aus der Gleichgewichtsbedingung (2.6):

Definition 2.16 *Für die Funktion f aus Gleichung (2.6) gilt.*

$$f := \begin{pmatrix} x^*_{t+1} - \frac{\partial H^t}{\partial \lambda_{t+1}} \\ \lambda_t - \frac{\partial H^t}{\partial x^*_t} \\ \frac{\partial H^t}{\partial u^*_t} \end{pmatrix} = 0 \tag{2.9}$$

Wie bereits erwähnt fällt der Erwartungswert bei Gleichung (2.6) im Gleichgewicht weg, da die stochastische Störung auf 0 gesetzt wurde. Zudem soll die Gleichgewichtsbedingung stets in der Form wie in Gleichung (2.6) vorliegen.

Für das Perturbations-Verfahren der Kontrolle u und der Dynamik x ist es notwendig, die Funktion f, wie eben gezeigt, mit Hilfe der Hamilton-Funktion im Voraus aufzustellen. Als Voraussetzung für den Algorithmus kommen also neben der Angabe des Gleichgewichts die entsprechend umformulierten Optimalitätsbedingungen hinzu.

2.2 Das Perturbations-Modell

Ein Grund für die bevorzugte Verwendung von Linearisierungs-Methoden in den Wirtschaftswissenschaften ist darin zu sehen, dass Entwicklungen höherer Ordnung meist

mit einem erheblichen numerischen Aufwand verbunden wurden. Dass es Möglichkeiten gibt Approximationen höherer Ordnung mit vergleichbarem Aufwand wie lineare Approximationen zu erhalten, zeigen die verwendeten Perturbations-Methoden. Durch die höhere Ordnung sind sie allerdings im Gegensatz zur Linearisierung in der Lage, die vorhandene Unsicherheit stärker in das Modell einzubinden und somit eine größere Genauigkeit zu erreichen.
Grundlage für die Perturbations-Verfahren ist eine Taylor-Entwicklung. Diese lässt sich formal folgendermaßen fassen:

Satz 2.17 *Sei* $\xi : \mathbb{R}^d \to \mathbb{R}$ *und* $\xi \in C^{k+1}$. *Dann gilt für* $x^0 \in \mathbb{R}^d$ *und* $x, x^0 \in [a, b]$

$$\begin{aligned}\xi(x) &= \xi(x^0) + \sum_{i=1}^{d} \frac{\partial \xi}{\partial x_i}(x^0)(x_i - x_i^0) \\ &\quad + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial^2 \xi}{\partial x_i \partial x_j}(x^0)(x_i - x_i^0)(x_j - x_j^0) \\ &\quad \vdots \\ &\quad + \frac{1}{k!}\sum_{i_1=1}^{d} \dots \sum_{i_k=1}^{d} \frac{\partial^k \xi}{\partial x_{i_1} \dots \partial x_{i_k}}(x^0)(x_{i_1} - x_{i_1}^0) \dots (x_{i_k} - x_{i_k}^0) \\ &\quad + O(\|x - x^0\|^{k+1}).\end{aligned}$$

In den verwendeten Verfahren wird dabei der Zustandsvektor in einen deterministischen und einen stochastischen Teil zerlegt:

$$x_t = [x_t^1; x_t^2]^T$$

mit $x_t^1 \in \mathbb{R}^{d_1}, x_t^2 \in \mathbb{R}^{d_2}, d = d_1 + d_2$.
Der erste Teil x_t^1 folgt einer rein deterministischen Dynamik. Er enthält hierbei nur endogene, das heißt sich aus dem Modell ergebende, Zustandsvariablen. Der zweite Teil x_t^2 folgt einem exogenen stochastischen Prozess, das heißt mit einer Störung von außen.
Wie die Koeffizienten für die jeweilige Taylor-Reihe ermittelt werden, geschieht bei der Perturbations-Methode der Kontrolle und der Wertefunktion auf unterschiedliche Weise, weshalb eine getrennte Betrachtung der Modelle erfolgt.

2.2.1 Das Modell für die Kontrollfunktion und die Dynamik

Wie bereits erwähnt, muss bei der Perturbations-Methode nach SCHMITT-GROHÉ und URIBE das Gleichgewicht $(\bar{x}, \bar{u})$ als Ausgangspunkt bekannt sein. Zur Berechnung des Gleichgewichts wurde $\sigma = 0$ gesetzt. Weiterhin werden noch eine Reihe wichtiger Annahmen getroffen, die im Detail in BECKER [2] nachzulesen sind. Unter anderem zählen dazu:

- x_t^2 ist gegeben durch

  $$x_{t+1}^2 = \widetilde{h}(x_t^2, \sigma) + \widetilde{\eta}\sigma\varepsilon_{t+1}$$

  mit $x_t^2, \varepsilon \in \mathbb{R}^{d_2}, \widetilde{\eta} \in \mathbb{R}^{d_2 \times d_2}, \sigma \geq 0$ und $\widetilde{h} : \mathbb{R}^{d_2} \times \mathbb{R} \to \mathbb{R}^{d_2}$

- Der Vektor ε_t ist unabhängig identisch verteilt mit Erwartungswert Null und Varianz/Kovarianzmatrix I.

- Die Standardabweichung $\sigma \geq 0$ und die Matrix $\widetilde{\eta}$ sind bekannt.

- Es wird angenommen, dass Funktionen $g := \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^m$ und $h := \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ existieren, die mindestens zwei mal bezüglich aller Variablen stetig differenzierbar sind, so dass für den Kontrollvektor

  $$u_t = g(x_t, \sigma) \tag{2.10}$$

  und für den Zustandsvektor

  $$x_{t+1} = h(x_t, \sigma) + \eta \cdot \sigma \cdot \varepsilon_{t+1} \tag{2.11}$$

  gilt, mit $\eta \in \mathbb{R}^{d_1 \times d_2}$ und $\eta := \begin{bmatrix} O \\ \widetilde{\eta} \end{bmatrix}$.

- f ist mindestens zweimal bezüglich aller Variablen differenzierbar und die Ableitungen von f sind bekannt.

Da hier eine Approximation der Kontrolle u_t und der Dynamik x_t gesucht ist, sind nach obigen Annahmen die Funktionen g und h zu approximieren. Dies geschieht mit Hilfe einer Taylor-Entwicklung nach Satz 2.17.

Um nun für die Entwicklung höherer Ordnung einen vergleichbaren Aufwand wie bei der Linearisierung zu erhalten, bedient sich die Methode von SCHMITT-GROHÉ und URIBE der folgenden Funktion:

$$F(x, \sigma) := E_t f(u_{t+1}, u_t, x_{t+1}, x_t) = 0. \tag{2.12}$$

Setzt man die Dynamiken (2.10) und (2.11) in Gleichung (2.12) ein, so erhält man:

$$F(x, \sigma) = E_t f(g(h(x_t, \sigma) + \eta\sigma\varepsilon_{t+1}, \sigma), g(x_t, \sigma), h(x_t, \sigma) + \eta\sigma\varepsilon_{t+1}, x_t) = 0. \tag{2.13}$$

Die Funktion F hat einige nützliche Eigenschaften, die für die Berechnung der Koeffizienten ausgenutzt werden können. Zum einen gilt, dass

$$F(x, \sigma) \equiv 0 \quad \forall x \in \mathbb{R}^d, \sigma \geq 0.$$

Daraus folgt aber sogleich, dass:

$$F_{x^k\sigma^j}(x, \sigma) \equiv 0 \quad \forall x \in \mathbb{R}^d, \sigma \in \mathbb{R}, k, j \in \mathbb{N}, \tag{2.14}$$

wobei $F_{x^k\sigma^j}$ die k-malige Ableitung in x-Richtung und die j-malige Ableitung in σ-Richtung bezeichnet.
Das heißt, nicht nur die Funktion F an sich ist stets Null, sondern auch alle Ableitungen erster oder höherer Ordnung. Wie diese Tatsache im Speziellen verwendet wird, um die gewünschten Koeffizienten für die Funktionen g und h zu erhalten, wird in Kapitel 3.1 erläutert.

Bemerkung 2.18 *Der Unterschied zwischen den Funktionen F und f liegt darin, dass für f lediglich im Gleichgewichtspunkt die Identität $f(\bar{u}, \bar{u}, \bar{x}, \bar{x}) = 0$ gilt. Außerhalb des Gleichgewichts kann f beliebige Werte annehmen. Deshalb müssen auch die Ableitungen von f nicht unbedingt Null sein.*

2.2.2 Das Modell für die Wertefunktion

Die für die Wertefunktion verwendete Perturbation-Methode basiert auf einem Verfahren von COLLARD und JUILLARD [6]. Sie verwendet ebenso wie das Modell für die

Kontrolle eine Taylor-Entwicklung gemäß Satz 2.17 mit dem Gleichgewicht als Entwicklungspunkt. Dieser wird mit Hilfe der Optimalitätsbedingungen aus Satz 2.14 aus dem vorherigen Abschnitt berechnet.
Grundlage ist weiterhin die Definition 2.11 der Wertefunktion für diskrete stochastische Kontrollprobleme:

$$V(x) = \sup_{u_t \in \mathbf{U}} E\left[\sum_{t=0}^{\infty} \beta^t \cdot \psi(x_t, u_t)\right], \tag{2.15}$$

mit $x_{t+1} = \varphi(x_t, u_t, z_t)$ und z_t unabhängig und identisch verteilte Zufallsvariable.
Zudem wird das Prinzip der Dynamischen Programmierung, auch Bellman'sches Optimalitätsprinzip genannt, ausgenutzt:

Satz 2.19 *Betrachte das diskrete stochastische Kontrollproblem 2.11. Dann erfüllt die optimale Wertefunktion $V(x)$ für jedes $x \in \mathbb{R}^d$ und jedes $k \in \mathbb{N}$ die Gleichung:*

$$V(x) = \sup_{u_t \in \mathbf{U}} E\left[\sum_{t=0}^{k} \beta^t \cdot \psi(x_t, u_t) + \beta^{k+1} \cdot V(x_{k+1})\right]. \tag{2.16}$$

Beweis: Der Beweis teilt sich in zwei Schritte. Zunächst wird (2.16) für "$\leq$" gezeigt, anschließend für "$\geq$", woraus die Gleichheit folgt.

(i) "$\leq$": Seien $x \in \mathbb{R}^d$, $k \in \mathbb{N}$ und $u_t \in \mathbf{U}$ beliebig. Dann gilt

$$\begin{aligned} E\left[\sum_{t=0}^{\infty} \beta^t \psi(x_t, u_t)\right] &= E\left[\sum_{t=0}^{k} \beta^t \psi(x_t, u_t) + \sum_{t=k+1}^{\infty} \beta^t \psi(x_t, u_t)\right] \leq \\ &\leq E\left[\sum_{t=0}^{k} \beta^t \psi(x_t, u_t) + \beta^{k+1} V(x_{k+1})\right]. \end{aligned}$$

Da $u_t \in \mathbf{U}$ beliebig gewählt wurde, gilt dies auch für das Supremum und somit für $V(x)$.

(ii) "$\geq$": Seien $x \in \mathbb{R}^d$, $k \in \mathbb{N}$ und $\varepsilon > 0$ beliebig. Wähle ein $\breve{u}_t \in \mathbf{U}$ so, dass

$$\begin{aligned} &\sup_{u_t \in \mathbf{U}} E\left[\sum_{t=0}^{k} \beta^t \psi(x_t, u_t) + \beta^{k+1} V(x_{k+1})\right] \\ &\leq E\left[\sum_{t=0}^{k} \beta^t \psi(x_t, \breve{u}_t) + \beta^{k+1} V(x_{k+1}) + \varepsilon\right] \end{aligned}$$

Dadurch ist $\check{u}_t$ auf [0,k] festgelegt. Wähle nun $\check{u}_t\big|_{(k,\infty)}$ so, dass

$$E\left[\sum_{t=0}^{\infty}\beta^t\psi(x_{t+k+1},\check{u}_{t+k+1})\right] \geq V(x_{k+1})-\varepsilon.$$

Damit ergibt sich

$$\begin{aligned}
&\sup_{u_t\in\mathbf{U}} E\left[\sum_{t=0}^{k}\beta^t\psi(x_t,u_t)+\beta^{k+1}V(x_{k+1})\right]\leq\\
&\leq E\left[\sum_{t=0}^{k}\beta^t\psi(x_t,\check{u}_t)+\sum_{t=0}^{\infty}\beta^{t+k+1}\psi(x_{t+k+1},\check{u}_{t+k+1})+(1+\beta^{k+1})\varepsilon)\right]\\
&\leq E\left[\sum_{t=0}^{k}\beta^t\psi(x_t,\check{u}_t)+\sum_{t=k+1}^{\infty}\beta^t\psi(x_t,\check{u}_t)+(1+\beta^{k+1})\varepsilon\right]\\
&= E\left[\sum_{t=0}^{\infty}\beta^t\psi(x_t,\check{u}_t)+(1+\beta^{k+1})\varepsilon\right]\leq V(x)+(1+\beta^{k+1})\varepsilon
\end{aligned}$$

und da $\varepsilon>0$ beliebig war, folgt somit die Behauptung.

□

Das Optimalitätsprinzip besagt anschaulich, dass die Endstücke optimaler Trajektorien wiederum optimale Trajektorien darstellen. Anders gesagt: Steuert man bis zu einem gewissen Punkt in endlicher Zeit optimal und verwendet den dabei erreichten Wert von V, so erhält man V in einem beliebigen Punkt.
Für eine bessere Übersicht werden im Folgenden Variablen aus der Periode $(t+1)$ stets mit einem Strich gekennzeichnet, während Variablen aus der Periode t keine Markierung erhalten. Für k=0, die optimale Kontrolle $u(x)$ und die optimale Dynamik $h(x) = (x_1', x_2')$ ergibt sich aus (2.16) die folgende Gleichung:

$$V(x) = E\left[\psi(x,u(x))+\beta\cdot V(h(x))\right]. \tag{2.17}$$

Das Verfahren besteht nun darin, eine Taylor-Reihe gleicher Ordnung $n\in\mathbb{N}$ für jede Seite der Gleichung (2.17) zu entwickeln. Dadurch findet sich auf beiden Seiten jeweils ein Term mit x^i, σ^j beziehungsweise $x^i\sigma^j$, $i,j=1,\ldots,n$ mit entsprechenden Koeffizienten. Da diese Koeffizienten für die gleichen Terme übereinstimmen müssen, erhält man ein lineares Gleichungssystem, das zum Beispiel in Matlab gelöst werden kann und

die Koeffizienten für die Taylor-Entwicklung der Wertefunktion V liefert.
Die genaue Herleitung der Berechnung der Koeffizienten erfolgt im anschließenden Kapitel.

2.3 Die Fixpunkt-Iteration

Für eine weitere Approximation der Wertefunktion V wird die Fixpunkt-Iteration verwendet. Sie bedient sich einer Iterationsvorschrift $\Psi : \mathbb{R}^N \to \mathbb{R}^N$, durch die eine Folge von Vektoren $\widetilde{V}^j \in \mathbb{R}^N$ durch $\widetilde{V}^{j+1} = \Psi(\widetilde{V}^j)$ berechnet wird. Diese Vorschrift erhalten wir aus der Gleichung (2.15), die der Definition 2.11 der Wertefunktion für diskrete stochastische Kontrollprobleme entspricht:

$$V(x) = \sup_{u_t \in \mathbf{U}} E\left[\sum_{t=0}^{\infty} \beta^t \cdot \psi(x_t, u_t)\right],$$

mit $x_{t+1} = \varphi(x_t, u_t, z_t)$ und z_t unabhängig und identisch verteilte Zufallsvariable.
Wie man daraus eine implementierbare Iterationsvorschrift erhält, wird im Folgenden hergeleitet. Der abschließende Abschnitt beschäftigt sich mit der Konvergenz des Iterationsverfahrens.

2.3.1 Die Iterationsvorschrift

Zur Herleitung der Rechenvorschrift setzt man in (2.15) die approximativ optimale Kontrolle $\widehat{u}$, $x_0 = x$ und $x_{t+1} = \varphi(x_t, \widehat{u}(x_t), z_t)$ mit φ aus Definition 2.8 ein. Daraus ergibt sich durch einige Umformungen die Gleichung (2.18):

$$\begin{aligned} V_{\widehat{u}}(x) &= E\left[\sum_{t=0}^{\infty} \beta^t \cdot \psi(x_t, \widehat{u}(x_t))\right] \\ V_{\widehat{u}}(x) &= E\left[\psi(x_0, \widehat{u}(x_0)) + \sum_{t=0}^{\infty} \beta^{t+1} \cdot \psi(x_{t+1}, \widehat{u}(x_{t+1}))\right] \end{aligned}$$

$$V_{\widehat{u}}(x) = E\left[\psi(x, \widehat{u}(x)) + \beta \cdot V_{\widehat{u}}(\varphi(x, \widehat{u}(x), z))\right] \tag{2.18}$$

In Anlehnung an das später verwendete Modell-Beispiel sei $x \in \mathbb{R}^2$. Für die Realisierung muss für x ein Gitter mit n äquidistanten Gitterpunkten je Koordinatenrichtung definiert werden, worin die Funktionen auszuwerten sind.

Definition 2.20 *Sei $\Omega \subset \mathbb{R}^2$ gegeben durch $\Omega = [a_1, b_1] \times [a_2, b_2]$ mit Werten $a_1 < b_1$ und $a_2 < b_2$. Ein (regelmäßiges) Rechteckgitter Γ auf Ω ist eine Menge von Rechtecken R_i, $i = 1, \ldots, P$, $P = (n-1)(n-1)$, so dass*

$$\bigcup_{i=1}^{P} R_i = \Omega \quad \text{und} \quad intR_i \cap intR_j = \emptyset, \text{ für alle } i, j = 1, \ldots, P, \ i \neq j.$$

Mit E_i, $i = 1, \ldots, N$, $N = n \cdot n$ bezeichnen wir die Knotenpunkte des Gitters.

Somit stellt $V_{\widehat{u}}(E_i)$ einen N-dimensionalen Vektor dar, mit $N = n^2$ und mit den Werten $\widetilde{V}_i = V_{\widehat{u}}(E_i),\ i = 1, \ldots, N$.

Um den Erwartungswert bilden zu können, muss die Zufallsvariable z ebenfalls räumlich diskretisiert werden. Sei hierzu analog zum späteren Modell-Beispiel $z \in \mathbb{R}$. Weiterhin wählen wir ein Gitter mit m äquidistanten Gitterpunkten z_k, $k = 1, \ldots, m$. Wertet man die Funktion φ in diesen Punkten aus, so ergeben sich die Werte $\varphi_k(x) = \varphi(x, \widehat{u}(x), z_k)$, $k = 1, \ldots, m$. Auf dieser Grundlage wird der Erwartungswert in (2.18) folgendermaßen gebildet:

$$\begin{aligned} \widetilde{V}_i &= E\left[\psi(E_i, \widehat{u}(E_i)) + \beta \cdot V_{\widehat{u}}(\varphi(E_i, \widehat{u}(E_i), z))\right] \\ &= \psi(E_i, \widehat{u}(E_i)) + \beta \cdot \sum_{k=1}^{m} V_{\widehat{u}}(\varphi_k(E_i)) \cdot p_k, \end{aligned} \tag{2.19}$$

mit $i = 1, \ldots, N$ und p_k der Wahrscheinlichkeit im Gitterknoten z_k, $k = 1, \ldots, m$.

Da $\widetilde{V}_i$ nur für diskrete Werten ermittelt wird, müssen die Werte $V_{\widehat{u}}(\varphi_k(E_i))$ nicht notwendigerweise gegeben sein. Dies ist allerdings zur Berechnung des Erwartungswertes nötig, weshalb man in den entsprechenden Punkten interpolieren muss, was mit stetigen und stückweise affin bilinearen Funktionen geschieht.

Definition 2.21 *(i) Sei $A \subset \mathbb{R}^2$. Eine Funktion $w : A \to \mathbb{R}$ heißt affin bilinear, falls es Konstanten $\alpha_0, \ldots, \alpha_3$ gibt, so dass für alle $x = (x_1, x_2) \in A$ die Identität $w(x) = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_1 x_2$ gilt.*

(ii) Betrachte eine rechteckförmige Menge $\Omega \subset \mathbb{R}^2$ mit Rechteckgitter Γ. Wir definieren den Raum der stetigen und stückweise affin blinearen Funktion auf Ω bezüglich des Gitters Γ als

W:=$\{w : \Omega \to \mathbb{R} \mid w$ *ist stetig und* $w|_{R_i}$ *ist affin bilinear für jedes* $i = 1, \ldots, P\}$.

Eine für uns wichtige Eigenschaft von **W** beschreibt das folgende Lemma.

Lemma 2.22 *Für jedes Rechteck $R_i = [c_1, d_1] \times [c_2, d_2]$ mit den Eckpunkten*

$$E_{i_1} = (c_1, c_2), E_{i_2} = (d_1, c_2), E_{i_3} = (c_1, d_2), E_{i_4} = (d_1, d_2)$$

lässt sich $w|_{R_i}$ für $x = (x_1, x_2) \in R_i$ schreiben als

$$w(x) = \sum_{j=1}^{4} \mu_j(x) w(E_{i_j})$$

mit

$$\mu_1(x) = (1 - y_1(x))(1 - y_2(x)), \qquad \mu_2(x) = y_1(x)(1 - y_2(x)),$$
$$\mu_3(x) = (1 - y_1(x))y_2(x), \qquad \mu_4(x) = y_1(x)y_2(x)$$

und

$$y_l(x) = \frac{x_l - c_l}{d_l - c_l} \quad \text{für } l = 1, 2.$$

Insbesondere gilt hierbei $\mu_j(x) \geq 0$ für $j = 1, \ldots, 4$ und $\sum_{j=1}^{4} \mu_j(x) = 1$.

Beweis: Hierfür ist zu zeigen, dass die Funktion w eindeutig durch ihre Werte $w(E_i)$ in den Eckpunkten des Gitters bestimmt ist. Dies ist auf Grund der obigen Konstruktion der Funktion einfach nachzuweisen. Ebenso lässt sich leicht zeigen, dass $\mu_j(x) \geq 0$ für $j = 1, \ldots, 4$ und $\sum_{j=1}^{4} \mu_j(x) = 1$ gilt.

□

Die in dieser Arbeit verwendete Implementierung benutzt das folgende Gesamtschrittverfahren.

Definition 2.23 *Betrachte ein diskretes stochastisches Kontrollproblem 2.11 und ein Rechteckgitter Γ mit N Eckpunkten. Zu jedem $i = 1, \ldots, N$ sei $B_k(i)$ der N-dimensionale Zeilenvektor, für den für jedes $w \in$ **W** und $W = (w(E_1), \ldots, w(E_N)) \in \mathbb{R}^N$ mit der üblichen Matrixmultiplikation gilt*

$$w(\varphi_k(E_i)) = B_k(i)W.$$

(Dabei hat $B(i)$ gemäß des vorherigen Lemmas höchstens 4 Einträge $\neq 0$, die sich außerdem zu 1 aufsummieren.) Des Weiteren sei $G(i) = \psi(E_i, \widehat{u}(E_i))$. Dann berechnen wir iterativ Vektoren $\widetilde{V}^j$ mit Startvektor $\widetilde{V}^0$ durch das Gesamtschrittverfahren

$$\widetilde{V}_i^{j+1} := G(i) + \beta \sum_{k=1}^{m} B_k(i) \widetilde{V}^j \cdot p_k, \quad \text{für } i = 1, \ldots, N,\ j = 0, 1, 2, \ldots . \tag{2.20}$$

Durch die Diskretisierung im Ort entsteht ein Fehler, der sich mittels affin bilinearer Funktionen abschätzen lässt. Näheres dazu ist in GRÜNE [12] in Kapitel 3.2.3 zu finden. Da die Iterationsvorschrift nur in seltenen Fällen eine exakte Lösung liefert, ist es nötig, ein Abbruchkriterium zu finden. Es bietet sich an, die Iteration bis zu einer bestimmten Genauigkeit auszuführen und das erreichte $\widetilde{V}^j$ als Näherungslösung anzugeben.

2.3.2 Konvergenz der Fixpunkt-Iteration

Als nächstes muss untersucht werden, ob die Lösung aus dem vorher beschriebenen Verfahren gegen die exakte Lösung von $V(x)$ konvergiert. Der folgende Satz 2.24 belegt die Konvergenz und die Eindeutigkeit der Lösung.

Satz 2.24 *Betrachte die Iterationsvorschrift aus Definition 2.23 und sei $\beta < 1$. Dann konvergieren die Vektoren $\widetilde{V}^j$ für $j \to \infty$ komponentenweise gegen den Vektor $\widetilde{V}$, der eindeutig bestimmt ist durch*

$$\widetilde{V}_i = G(i) + \beta \sum_{k=1}^{m} B_k(i) \widetilde{V} \cdot p_k \quad \text{für } i = 1, \ldots, N.$$

Beweis: Für beliebige Vektoren $V, W \in \mathbb{R}^N$ und alle $i = 1, \ldots, N$ gilt die Ungleichung

$$\left| G(i) + \beta \sum_{k=1}^{m} B_k(i) V \cdot p_k - G(i) - \beta \sum_{k=1}^{m} B_k(i) W \cdot p_k \right| \leq \beta ||V - W||_\infty, \tag{2.21}$$

mit $|| \cdot ||_\infty$ der L_∞-Norm im $\mathbb{R}^N$. Hierbei wird ausgenutzt, dass nach Lemma 2.22 $\sum\limits_{i=1}^{N} B_k(i) = 1$ für $k = 1, \ldots, m$. Da zudem p_k ein Wahrscheinlichkeitsmaß ist, gilt, dass $\sum\limits_{k=1}^{m} p_k = 1$ ist, und man erhält $\sum\limits_{k=1}^{m} \sum\limits_{i=1}^{N} B_k(i) p_k = 1$. Somit definiert (2.21) eine Kontraktion auf dem $\mathbb{R}^N$ bezüglich der L_∞-Norm, weswegen der Vektor $\widetilde{V}$ existiert. Dieser ist zudem eindeutig, denn für jeden weiteren solchen Vektor W gilt mit (2.21)

$$||\widetilde{V} - W||_\infty \leq \beta ||\widetilde{V} - W||_\infty < ||\widetilde{V} - W||_\infty,$$

woraus $W = \widetilde{V}$ folgt.

□

Wir erhalten mit diesem Verfahren also eine Näherungslösung mit einer gewissen Genauigkeit, bei deren Erreichen das Verfahren abbricht. Dennoch kann es auf Grund von Rundungsfehlern dazu führen, dass die Iteration keinen Fortschritt mehr zeigt, obwohl die gewünschte Genauigkeit noch nicht erreicht ist. Es gibt einige Möglichkeiten, das Verfahren dahingehend zu verbessern, worauf wegen des begrenzten Rahmens der Arbeit jedoch nicht näher eingegangen wird.

3 Herleitung der Koeffizienten für die Perturbations-Methoden

Da sowohl die Kontrolle und die Dynamik als auch die Wertefunktion durch eine Taylor-Reihe approximiert werden, behandelt dieses Kapitel die Herleitung der Koeffizienten der jeweiligen Reihe. Wir wollen hierbei zunächst die Terme für die Approximation der Kontrolle und der Dynamik nach SCHMITT-GROHÉ und URIBE [24] ermitteln. Anschließend folgt die Entwicklung für die Wertefunktion nach COLLARD und JUILLARD [6].

3.1 Koeffizienten für die Kontrollfunktion und die Dynamik

Bei der Taylor-Entwicklung für die Kontrolle u und die Dynamik x beginnt man mit der Berechnung der linearen Terme. Da eine lineare Approximation aber große Ungenauigkeiten aufweist, ist weiterhin die Berechnung der quadratischen Terme der Taylor-Reihe nötig. Zuletzt wird noch kurz auf Approximationen höherer Ordnung eingegangen, deren Berechnung in ebenso einfacher Weise vollzogen werden kann.

3.1.1 Lineare Approximation

Der Gleichgewichtspunkt sei dabei durch $(\bar{x}, \bar{u})$ gegeben. Gesucht ist nun eine Approximation von g und h aus (2.10) und (2.11) um den Entwicklungspunkt $(x, \sigma) = (\bar{x}, 0)$.

Bemerkung 3.1 *Zu Beginn sei auf die verwendete Schreibweise bei den Ableitungen hingewiesen. In dieser Arbeit wird die Notation aus* COLLARD *und* JUILLARD [7] *übernommen. Als Beispiel seien die Ableitung erster Ordnung mit* $[h_x(\bar{x}, 0)]^j_a = \dfrac{\partial h_j(\bar{x}, 0)}{\partial x_a}$ *und die Ableitung zweiter Ordnung mit* $[h_{xx}(\bar{x}, 0)]^j_{ab} = \dfrac{\partial^2 h_j(\bar{x}, 0)}{\partial x_a \partial x_b}$ *gegeben.*

Die linearen Taylor-Entwicklungen schreiben sich somit in der Form

$$[g(x,\sigma)]^i = [g(\bar{x},0)]^i + [g_x(\bar{x},0)]^i_a \left[(x-\bar{x})\right]_a + [g_\sigma(\bar{x},0)]^i [\sigma] \tag{3.1}$$

$$[h(x,\sigma)]^j = [h(\bar{x},0)]^j + [h_x(\bar{x},0)]^j_a \left[(x-\bar{x})\right]_a + [h_\sigma(\bar{x},0)]^j [\sigma], \tag{3.2}$$

mit $i = 1, \ldots, m$ und $a, j = 1, \ldots, d$.

Zur Ermittlung des Gleichgewichts wurde $\sigma = 0$ verwendet, das heißt man bedint sich des Ansatzes, dass keine Störung von außen existiert. Deshalb fällt der Erwartungswert in (2.6) weg und es gilt die Identität $f(\bar{u}, \bar{u}, \bar{x}, \bar{x}) = 0$, woraus man

$$g(\bar{x}, 0) = \bar{u},$$
$$h(\bar{x}, 0) = \bar{x}.$$

erhält.

Wir erinnern daran, dass die Funktion F in Kapitel 2.2.1 definiert wurde, um die fehlenden Terme zu berechnen. Dies ist mit Hilfe der folgenden Gleichungen möglich, wobei wiederum ausgenutzt wird, dass zur Berechnung des Gleichgewichts die Störung auf 0 gesetzt wurde, somit also ein deterministisches Problem vorliegt, und dass $E[\varepsilon'] = 0$ gilt, da $\varepsilon \sim N(0,1)$-verteilt ist:

$$F_\sigma(\bar{x}, 0) = 0 \tag{3.3}$$

$$F_x(\bar{x}, 0) = 0 \tag{3.4}$$

Auch hier werden aus Gründen der Übersichtlichkeit bei der Berechnung der Ableitungen Variablen aus der Periode $(t+1)$ mit einem Strich gekennzeichnet. Bei Variablen aus Periode t entfällt der Index.

Aus der ersten Gleichung erhalten wir die Werte $h_\sigma = g_\sigma = 0$. Die zweite Gleichung liefert mit (2.12) beziehungsweise (2.13):

$$\begin{pmatrix} f_{x'} & f_{u'} \end{pmatrix} \cdot \begin{pmatrix} I \\ g_x \end{pmatrix} h_x = - \begin{pmatrix} f_x & f_u \end{pmatrix} \cdot \begin{pmatrix} I \\ g_x \end{pmatrix}. \tag{3.5}$$

Aus dieser Gleichung (3.5) lassen sich durch geeignete Zerlegungen h_x und g_x bestimmen. Eine genauere Herleitung der Ergebnisse, sowie die Klärung der Existenz und Eindeutigkeit der Lösungen sind in BECKER [2] ausgeführt.

3.1.2 Quadratische Approximation

Die eben hergeleitete lineare Approximation liefert nur sehr ungenaue Ergebnisse. Nach JIN und JUDD [14] kann dies bisweilen sogar zu völlig falschen Resultaten führen. Als Beispiel wird eine Studie von TESAR [27] genannt, die die generelle Verwendbarkeit der Standard-Methode von KYDLAND und PRESCOTT [17] widerlegt. Deshalb wird es nötig, die quadratischen Terme der Taylor-Approximation zu berechnen, was auf ebenso einfache Weise geschieht, wie bei den linearen Termen.
Gesucht ist nun eine Approximation von g und h aus (2.10) und (2.11) um den Punkt $(x, \sigma) = (\bar{x}, 0)$ der folgenden Form:

$$\begin{aligned}
[g(x,\sigma)]^i &= [g(\bar{x},0)]^i + [g_x(\bar{x},0)]^i_a[(x-\bar{x})]_a + [g_\sigma(\bar{x},0)]^i[\sigma] \\
&\quad + \frac{1}{2}[g_{xx}(\bar{x},0)]^i_{ab}[(x-\bar{x})]_a[(x-\bar{x})]_b \\
&\quad + \frac{1}{2}[g_{x\sigma}(\bar{x},0)]^i_a[(x-\bar{x})]_a[\sigma] \\
&\quad + \frac{1}{2}[g_{\sigma x}(\bar{x},0)]^i_a[(x-\bar{x})]_a[\sigma] \\
&\quad + \frac{1}{2}[g_{\sigma\sigma}(\bar{x},0)]^i[\sigma][\sigma]
\end{aligned}$$

und

$$\begin{aligned}
[h(x,\sigma)]^j &= [h(\bar{x},0)]^j + [h_x(\bar{x},0)]^j_a[(x-\bar{x})]_a + [h_\sigma(\bar{x},0)]^j[\sigma] \\
&\quad + \frac{1}{2}[h_{xx}(\bar{x},0)]^j_{ab}[(x-\bar{x})]_a[(x-\bar{x})]_b \\
&\quad + \frac{1}{2}[h_{x\sigma}(\bar{x},0)]^j_a[(x-\bar{x})]_a[\sigma] \\
&\quad + \frac{1}{2}[h_{\sigma x}(\bar{x},0)]^j_a[(x-\bar{x})]_a[\sigma] \\
&\quad + \frac{1}{2}[h_{\sigma\sigma}(\bar{x},0)]^j[\sigma][\sigma]
\end{aligned}$$

mit $i = 1, \ldots, m$ und $a, b, j = 1, \ldots, d$.

Die darin enthaltenen unbekannten Terme sind $[g_{xx}]^i_{ab}$, $[g_{x\sigma}]^i_a$, $[g_{\sigma x}]^i_a$, $[g_{\sigma\sigma}]^i$, $[h_{xx}]^j_{ab}$, $[h_{x\sigma}]^j_a$, $[h_{\sigma x}]^j_a$, $[h_{\sigma\sigma}]^j$. Die restlichen Terme sind zum einen aufgrund der linearen Approximation, zum anderen aufgrund der bekannten Ableitungen von f bereits gegeben.

Wiederum ist es möglich, die fehlenden Terme mit Hilfe der Funktion F aus Gleichung (2.12) zu berechnen. Die dafür notwendigen Gleichungen sind gegeben durch:

$$[F_{xx}(\bar{x},0)]^i_{jk} = 0 \quad \text{mit} \quad i=1,\dots,n; \quad j,k=1,\dots,d, \tag{3.6}$$

$$[F_{\sigma\sigma}(\bar{x},0)]^i = 0 \quad \text{mit} \quad i=1,\dots,n, \tag{3.7}$$

$$[F_{\sigma x}(\bar{x},0)]^i_{jk} = 0 \quad \text{mit} \quad i=1,\dots,n; \quad j,k=1,\dots,d \tag{3.8}$$

Aus der ersten Gleichung (3.6) erhält man h_{xx} und g_{xx}, die zweite (3.7) liefert $h_{\sigma\sigma}$ und $g_{\sigma\sigma}$ und die letzte (3.8) schließlich $h_{\sigma x} = g_{\sigma x} = 0$. Für eine ausführlichere Berechnung, sowie den Nachweis der Existenz und Eindeutigkeit der Lösungen sei auf BECKER [2] verwiesen.

Wie wir bei der linearen und quadratischen Approximation gesehen haben, ergibt sich $h_\sigma = g_\sigma = 0$ und $h_{x\sigma} = g_{x\sigma} = 0$. Das bedeutet für die lineare Approximation, dass der stochastische Anteil keine Bedeutung hat, weshalb es auch oft zu Falschaussagen bei rein linearer Approximation kommen kann. Der Grund, dass derartige Approximationen dennoch häufig verwendet wurden, liegt darin, dass man eine Taylor-Entwicklung zweiter Ordnung als viel aufwändiger einschätzte.

Das Verfahren von SCHMITT-GROHÉ und URIBE [24] zeigt hingegen, wie man eine quadratische Approximation erhält, in der man, wie bei der linearen Entwicklung, nur lineare Gleichungssysteme zu lösen braucht, wodurch der Rechenaufwand überschaubar bleibt.

Bei der Taylor-Reihe zweiter Ordnung werden die Kontrolle u und die Dynamik x im Vergleich zu ihrem deterministischen Gegenpart lediglich um einen jeweils konstanten Term erweitert, der durch $\frac{1}{2} g_{\sigma\sigma}\, \sigma^2$ beziehungsweise $\frac{1}{2} h_{\sigma\sigma}\, \sigma^2$ gegeben ist.

3.1.3 Approximation höherer Ordnung

Ist man an noch genaueren Ergebnissen interessiert, kann man eine Taylor-Entwicklung höherer Ordnung wählen. Die hinzukommenden Terme sind dabei analog der gezeigten Methode zu berechnen. Es wird wiederum ausgenutzt, dass alle Ableitungen der Funktion F identisch Null sind. Zudem sind bereits die Terme von h und g für die lineare und quadratische Approximation, und die Ableitungen von f an der Stelle $(\bar{x}, \bar{u})$ bekannt. So

können die Terme höherer Ordnung berechnet werden, wobei lediglich lineare Gleichungen zu lösen sind. Eine nähere Ausführung der Vorgehensweise ist in dieser Arbeit nicht vorgesehen, da wir uns mit der quadratischen Approximation begnügen werden.

3.2 Koeffizienten für die Wertefunktion

Da auch die Wertefunktion durch eine quadratische Taylor-Entwicklung approximiert werden soll, befasst sich dieser Abschnitt mit der Herleitung der dazu benötigten Koeffizienten.
Hierbei ist zunächst das Aufstellen des Gleichungssystems notwendig, aus dem die entsprechenden Koeffizienten ermittelt werden können. Dies geschieht für lineare und quadratische Approximationen auf die gleiche Weise. Anschließend wird kurz auf Approximationen höherer Ordnung eingegangen.

3.2.1 Lineare Approximation

Die Herleitung des angesprochenen Gleichungssystems zur linearen Entwicklung

$$\widehat{V}(x,\sigma) = V_0(\bar{x},0) + [V_x(\bar{x},0)]_a\,[(x-\bar{x})]_a + V_\sigma(\bar{x},0)\sigma$$

erfolgt analog einer Arbeit von COLLARD und JUILLARD [6]. Grundlage ist, wie im vorherigen Kapitel gezeigt, eine lineare Taylor-Entwicklung der beiden Seiten der Gleichung (2.17), die man aus der Definition 2.11 der Wertefunktion für diskrete stochastische Kontrollprobleme mit Hilfe des Bellman'schen Optimalitätsprinzips erhält:

$$V(x) \;=\; E\left[\psi(x,u(x)) + \beta \cdot V(h(x))\right].$$

Zunächst werden wir die Gleichung wie folgt umformulieren.
Es wird eine Funktion $G(x,\sigma)$ definiert, so dass

$$V(x) \;=\; E\left[G(x,\sigma)\right]$$

gilt. Eine Taylor-Entwicklung der beiden Seiten dieser Gleichung führt schließlich zu der Form:

$$\begin{aligned} V_0(\bar{x},0) &+ [V_x(\bar{x},0)]_a [(x-\bar{x})]_a + V_\sigma(\bar{x},0)\sigma \\ &= E\left[G_0(\bar{x},0) + [G_x(\bar{x},0)]_a [(x-\bar{x})]_a + G_\sigma(\bar{x},0)\sigma\varepsilon\right]. \end{aligned} \tag{3.9}$$

mit $a = 1, \ldots, d$, wobei stets in dem Punkt $(x, \sigma) = (\bar{x}, 0)$ ausgewertet wird, da das Gleichgewicht $\bar{x}$ für $\sigma = 0$ ermittelt wurde. Die Funktionen $V_{x^k\sigma^j}$ beziehungsweise $G_{x^k\sigma^j}$ beschreiben dabei die k-malige Ableitung in x-Richtung und j-malige Ableitung in σ-Richtung.
Da unsere Zufallsvariable $\varepsilon \sim N(0,1)$-verteilt ist, gilt für den Erwartungswert $E[\varepsilon] = 0$. Daraus folgt, dass der Term $E[G_\sigma(\bar{x},0)\sigma\varepsilon]$ verschwindet, weshalb auch $V_\sigma(\bar{x},0)\sigma = 0$ gelten muss. Wir erhalten also $V_\sigma(\bar{x},0) = 0$, da im Allgemeinen $\sigma \neq 0$ ist. Dies bedeutet wiederum, dass der stochastische Anteil bei der linearen Approximation komplett entfällt. Ein vergleichbares Verhalten zeigte sich bereits bei der linearen Approximation der Kontrolle und der Dynamik nach SCHMITT-GROHÉ und URIBE.
Damit verbleibt von (3.9) die Taylor-Entwicklung

$$\begin{aligned} V_0(\bar{x},0) &+ [V_x(\bar{x},0)]_a [(x-\bar{x})]_a \\ &= G_0(\bar{x},0) + [G_x(\bar{x},0)]_a [(x-\bar{x})]_a, \end{aligned}$$

aus der sich ein lineares Gleichungssystem ergibt:

$$\begin{aligned} I) \quad & V_0(\bar{x},0) = G_0(\bar{x},0) \\ II) \quad & [V_x(\bar{x},0)]_a = [G_x(\bar{x},0)]_a, \end{aligned}$$

wobei $a = 1, \ldots, d$.
Dieses Gleichungssystem kann zum Beispiel in Maple mit Hilfe bereitgestellter Routinen wie `linsolve(A,b)` zur Berechnung von $Ay = b$, mit A Matrix der Dimension $n \times n$ und b Vektor der Dimension n, gelöst werden.
Da ψ aus (2.17) in unserem späteren Modell nur von $u(x)$ abhängt, sei zur besseren Übersichtlichkeit auf die Ableitung von ψ nach x im Folgenden verzichtet. Für eine

zweidimensionale Dynamik $h(x) = (x_1', x_2')$ hat obiges Gleichungssystem damit die folgende Gestalt:

$$\begin{aligned}
V_0 &= \psi(u(\bar{x})) + \beta V(h(\bar{x})) \\
V_{x_1} &= \frac{\partial \psi(u(x))}{\partial u(x)} \cdot \frac{\partial u(x)}{\partial x_1}\bigg|_{x=\bar{x},\sigma=0} + \beta \frac{\partial V(h(x))}{\partial x_1'} \cdot \frac{\partial x_1'}{\partial x_1}\bigg|_{x=\bar{x},\sigma=0} \\
&\quad + \beta \frac{\partial V(h(x))}{\partial x_2'} \cdot \frac{\partial x_2'}{\partial x_1}\bigg|_{x=\bar{x},\sigma=0} \\
V_{x_2} &= \frac{\partial \psi(u(x))}{\partial u(x)} \cdot \frac{\partial u(x)}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0} + \beta \frac{\partial V(h(x))}{\partial x_1'} \cdot \frac{\partial x_1'}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0} \\
&\quad + \beta \frac{\partial V(h(x))}{\partial x_2'} \cdot \frac{\partial x_2'}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0}
\end{aligned}$$

Um das Gleichungssystem etwas einfacher und übersichtlicher zu gestalten, notieren wir die Ableitungen wieder durch entsprechende Indizes. Dabei soll ein Index von $x_1^i x_2^j \sigma^k$ die i-malige Ableitung nach x_1, j-malige Ableitung nach x_2 und k-malige Ableitung in σ-Richtung, ausgewertet im Gleichgewicht $(x, \sigma) = (\bar{x}, 0)$, bezeichnen. Da der Gleichgewichtspunkt für $\sigma = 0$ berechnet wurde, also für den Fall, dass keine Störung von außen existiert, werden die Ableitungen für dieses σ ausgewertet.
Damit erhalten wir das folgende Gleichungssystem, aus dem die fehlenden Koeffizienten V_0, V_{x_1} und V_{x_2} bestimmt werden können.

$$\begin{aligned}
V_0 &= \psi(u) + \beta V_0 \\
V_{x_1} &= \psi_u \cdot u_{x_1} + \beta V_{x_1} \cdot (x_1')_{x_1} + \beta V_{x_2} \cdot (x_2')_{x_1} \\
V_{x_2} &= \psi_u \cdot u_{x_2} + \beta V_{x_1} \cdot (x_1')_{x_2} + \beta V_{x_2} \cdot (x_2')_{x_2}
\end{aligned}$$

Dieses Gleichungssystem wird derart umformuliert, dass sich die Gleichung $Ay = b$ mit einer 3×3-Matrix A, einem 3-dimensionalen Vektor b und der Lösung $y = (V_0, V_{x_1}, V_{x_2})^T$ ergibt. Für die eindeutige Lösbarkeit dieser Gleichung muss die Matrix A vollen Rang besitzen, da die Tatsache nicht ausreicht, dass die Anzahl der Gleichungen der Anzahl der Unbekannten entspricht. Die Matrix A muss also invertierbar sein. Im späteren Programm wird dies nur indirekt überprüft. Die Maple-Routine `linsolve(A,b)` würde dabei mit Hilfe globaler Variablen die Familie der Lösungen angeben beziehungsweise

einen Null-Vektor liefern, falls die Gleichung nicht lösbar ist. Dadurch fließt indirekt die Eindeutigkeit und Existenz der Lösung des Gleichungssystems in die Berechnungen ein. Eine derartige Approximation der Wertefunktion entspricht der in den Wirtschaftswissenschaften häufig verwendeten Standard-Linearisierung. Die Linearisierung kann somit als spezieller Fall der Perturbations-Methoden gesehen werden. Ihr großer Nachteil besteht nach COLLARD und JUILLARD [6] darin, dass,

> especially in the case of asset pricing models, [...] it does not take advantage of any piece of information contained in the distribution of the shocks. In particular, assuming certainty equivalence, this solution leads to a solution that ignores risk.

Deshalb ist es sinnvoll, Approximationen höherer Ordnung zu betrachten, um die dadurch zusätzlichen Terme höherer Momente einzubeziehen und somit eine vom Risiko abhängige numerische Lösung zu erhalten.

3.2.2 Quadratische Approximation

Die quadratische Taylor-Entwicklung der Wertefunktion

$$\begin{aligned}\widehat{V}(x,\sigma) &= V_0(\bar{x},0) + [V_x(\bar{x},0)]_a [(x-\bar{x})]_a + V_\sigma(\bar{x},0)\sigma \\ &\quad + \frac{1}{2}[V_{xx}(\bar{x},0)]_{ab}[(x-\bar{x})]_a[(x-\bar{x})]_b + \frac{1}{2}[V_{x\sigma}(\bar{x},0)]_a[(x-\bar{x})]_a\sigma \\ &\quad + \frac{1}{2}[V_{\sigma x}(\bar{x},0)]_a\sigma[(x-\bar{x})]_a + \frac{1}{2}V_{\sigma\sigma}(\bar{x},0)\sigma^2\end{aligned}$$

berücksichtigt dieses angesprochene Risiko. Sie verläuft analog der linearen Approximation.
Wieder werden beide Seiten der Gleichung (2.17)

$$V(x) = E[\psi(x,u(x)) + \beta \cdot V(h(x))]$$

gemäß Taylor entwickelt, diesmal durch Polynome zweiter Ordnung:

$$
\begin{aligned}
V_0(\bar{x},0) \;+\; & [V_x(\bar{x},0)]_a\,[(x-\bar{x})]_a + V_\sigma(\bar{x},0)\sigma \\
& + \frac{1}{2}\,[V_{xx}(\bar{x},0)]_{ab}\,[(x-\bar{x})]_a\,[(x-\bar{x})]_b \\
& + \frac{1}{2}\,[V_{x\sigma}(\bar{x},0)]_a\,[(x-\bar{x})]_a\,\sigma \\
& + \frac{1}{2}\,[V_{\sigma x}(\bar{x},0)]_a\,\sigma\,[(x-\bar{x})]_a + \frac{1}{2}V_{\sigma\sigma}(\bar{x},0)\sigma^2 \\
= \; & E[G_0(\bar{x},0) + [G_x(\bar{x},0)]_a\,[(x-\bar{x})]_a + G_\sigma(\bar{x},0)\sigma\varepsilon \\
& + \frac{1}{2}\,[G_{xx}(\bar{x},0)]_{ab}\,[(x-\bar{x})]_a\,[(x-\bar{x})]_b \\
& + \frac{1}{2}\,[G_{x\sigma}(\bar{x},0)]_a\,[(x-\bar{x})]_a\,\sigma\varepsilon \\
& + \frac{1}{2}\,[G_{\sigma x}(\bar{x},0)]_a\,\sigma\varepsilon\,[(x-\bar{x})]_a + \frac{1}{2}G_{\sigma\sigma}(\bar{x},0)\sigma^2\varepsilon^2],
\end{aligned}
\tag{3.10}
$$

mit $a, b = 1, \ldots, d$, wobei wiederum in $(x, \sigma) = (\bar{x}, 0)$ ausgewertet wird. Zur Erinnerung bezeichnet $V_{x^k\sigma^j}$ beziehungsweise $G_{x^k\sigma^j}$ die k-malige Ableitung nach x und die j-malige Ableitung nach σ.

Auf Grund der Verteilung von $\varepsilon \sim N(0,1)$ gilt, dass $E[\varepsilon] = E[x\varepsilon] = E[\varepsilon x] = 0$ ist. Dadurch erhalten wir mit $a = 1, \ldots, d$ die Identitäten

$$E[G_\sigma(\bar{x},0)\sigma\varepsilon] = E[[G_{x\sigma}(\bar{x},0)]_a\,[(x-\bar{x})]_a\,\sigma\varepsilon] = E[[G_{\sigma x}(\bar{x},0)]_a\,\sigma\varepsilon\,[(x-\bar{x})]_a] = 0.$$

Somit müssen auch die Terme $V_\sigma(\bar{x},0)\sigma$, $[V_{x\sigma}(\bar{x},0)]_a\,[(x-\bar{x})]_a\,\sigma$ und $[V_{\sigma x}(\bar{x},0)]_a\,\sigma\,[(x-\bar{x})]_a$ verschwinden, weshalb gilt:

$$
\begin{aligned}
V_\sigma(\bar{x},0) &= 0 \\
[V_{x\sigma}(\bar{x},0)]_a &= [V_{\sigma x}(\bar{x},0)]_a = 0,
\end{aligned}
$$

für $a = 1, \ldots, d$.

Zudem ist $E[\varepsilon^2] = 1$ wegen $\varepsilon \sim N(0,1)$ und deshalb $E[\frac{1}{2}G_{\sigma\sigma}(\bar{x},0)\sigma^2\varepsilon^2] = \frac{1}{2}G_{\sigma\sigma}(\bar{x},0)\sigma^2$.

Die damit von (3.10) verbleibende Taylor-Entwicklung

$$
\begin{aligned}
V_0(\bar{x},0) \;+\; & [V_x(\bar{x},0)]_a\,[(x-\bar{x})]_a + \frac{1}{2}\,[V_{xx}(\bar{x},0)]_{ab}\,[(x-\bar{x})]_a\,[(x-\bar{x})]_b + \frac{1}{2}V_{\sigma\sigma}(\bar{x},0)\sigma^2 \\
= \; & G_0(\bar{x},0) + [G_x(\bar{x},0)]_a\,[(x-\bar{x})]_a + \frac{1}{2}\,[G_{xx}(\bar{x},0)]_{ab}\,[(x-\bar{x})]_a\,[(x-\bar{x})]_b \\
& + \frac{1}{2}G_{\sigma\sigma}(\bar{x},0)\sigma^2
\end{aligned}
$$

liefert uns folgendes lineares Gleichungssystem, das wieder in Maple mit der gegebenen Routine `linsolve(A,b)` zu lösen ist:

$$
\begin{aligned}
I) \quad & V_0(\bar{x},0) = G_0(\bar{x},0) \\
II) \quad & [V_x(\bar{x},0)]_a = [G_x(\bar{x},0)]_a \\
III) \quad & [V_{xx}(\bar{x},0)]_{ab} = [G_{xx}(\bar{x},0)]_{ab} \\
IV) \quad & V_{\sigma\sigma}(\bar{x},0) = G_{\sigma\sigma}(\bar{x},0),
\end{aligned}
$$

mit $a, b = 1, \ldots, d$. Analog zur Kontrolle und Dynamik wird auch die Wertefunktion im Vergleich zu ihrem deterministischen Gegenpart lediglich um einen konstanten Term, nämlich $\frac{1}{2}V_{\sigma\sigma}\sigma^2$, werweitert.
Für eine zweidimensionale Dynamik $h(x) = (x_1', x_2')$ und ψ aus (2.17) wiederum nur von $u(x)$ abhängig, gestaltet sich dieses Gleichungssystem folgendermaßen:

$$
\begin{aligned}
V_0 &= \psi(u(\bar{x})) + \beta V(h(\bar{x})) \\
V_{x_1} &= \frac{\partial\psi(u(x))}{\partial u(x)} \cdot \frac{\partial u(x)}{\partial x_1}\bigg|_{x=\bar{x},\sigma=0} + \beta\frac{\partial V(h(x))}{\partial x_1'} \cdot \frac{\partial x_1'}{\partial x_1}\bigg|_{x=\bar{x},\sigma=0} \\
&\quad + \beta\frac{\partial V(h(x))}{\partial x_2'} \cdot \frac{\partial x_2'}{\partial x_1}\bigg|_{x=\bar{x},\sigma=0} \\
V_{x_2} &= \frac{\partial\psi(u(x))}{\partial u(x)} \cdot \frac{\partial u(x)}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0} + \beta\frac{\partial V(h(x))}{\partial x_1'} \cdot \frac{\partial x_1'}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0} \\
&\quad + \beta\frac{\partial V(h(x))}{\partial x_2'} \cdot \frac{\partial x_2'}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0} \\
V_{x_1x_1} &= \frac{\partial^2\psi(u(x))}{(\partial u(x))^2} \cdot \left(\frac{\partial u(x)}{\partial x_1}\right)^2\bigg|_{x=\bar{x},\sigma=0} + \frac{\partial\psi(u(x))}{\partial u(x)} \cdot \frac{\partial^2 u(x)}{(\partial x_1)^2}\bigg|_{x=\bar{x},\sigma=0} \\
&\quad + \beta\frac{\partial^2 V(h(x))}{(\partial x_1')^2} \cdot \left(\frac{\partial x_1'}{\partial x_1}\right)^2\bigg|_{x=\bar{x},\sigma=0} + \beta\frac{\partial^2 V(h(x))}{\partial x_1'\partial x_2'} \cdot \frac{\partial x_1'}{\partial x_1} \cdot \frac{\partial x_2'}{\partial x_1}\bigg|_{x=\bar{x},\sigma=0} \\
&\quad + \beta\frac{\partial V(h(x))}{\partial x_1'} \cdot \frac{\partial^2 x_1'}{(\partial x_1)^2}\bigg|_{x=\bar{x},\sigma=0} + \beta\frac{\partial^2 V(h(x))}{\partial x_2'\partial x_1'} \cdot \frac{\partial x_2'}{\partial x_1} \cdot \frac{\partial x_1'}{\partial x_1}\bigg|_{x=\bar{x},\sigma=0} \\
&\quad + \beta\frac{\partial^2 V(h(x))}{(\partial x_2')^2} \cdot \left(\frac{\partial x_2'}{\partial x_1}\right)^2\bigg|_{x=\bar{x},\sigma=0} + \beta\frac{\partial V(h(x))}{\partial x_2'} \cdot \frac{\partial^2 x_2'}{(\partial x_1)^2}\bigg|_{x=\bar{x},\sigma=0} \\
V_{x_1x_2} &= \frac{\partial^2\psi(u(x))}{(\partial u(x))^2} \cdot \frac{\partial u(x)}{\partial x_1} \cdot \frac{\partial u(x)}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0} + \frac{\partial\psi(u(x))}{\partial u(x)} \cdot \frac{\partial^2 u(x)}{\partial x_1\partial x_2}\bigg|_{x=\bar{x},\sigma=0}
\end{aligned}
$$

$$
\begin{aligned}
&+\beta\frac{\partial^2 V(h(x))}{(\partial x_1')^2}\cdot\frac{\partial x_1'}{\partial x_1}\cdot\frac{\partial x_1'}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0}+\beta\frac{\partial^2 V(h(x))}{\partial x_1'\partial x_2'}\cdot\frac{\partial x_1'}{\partial x_1}\cdot\frac{\partial x_2'}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0}\\
&+\beta\frac{\partial V(h(x))}{\partial x_1'}\cdot\frac{\partial^2 x_1'}{\partial x_1\partial x_2}\bigg|_{x=\bar{x},\sigma=0}+\beta\frac{\partial^2 V(h(x))}{\partial x_2'\partial x_1'}\cdot\frac{\partial x_2'}{\partial x_1}\cdot\frac{\partial x_1'}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0}\\
&+\beta\frac{\partial^2 V(h(x))}{(\partial x_2')^2}\cdot\frac{\partial x_2'}{\partial x_1}\cdot\frac{\partial x_2'}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0}+\beta\frac{\partial V(h(x))}{\partial x_2'}\cdot\frac{\partial^2 x_2'}{\partial x_1\partial x_2}\bigg|_{x=\bar{x},\sigma=0}
\end{aligned}
$$

$$
\begin{aligned}
V_{x_2x_2} \;=\;& \frac{\partial^2\psi(u(x))}{(\partial u(x))^2}\cdot\left(\frac{\partial u(x)}{\partial x_2}\right)^2\bigg|_{x=\bar{x},\sigma=0}+\frac{\partial\psi(u(x))}{\partial u(x)}\cdot\frac{\partial^2 u(x)}{(\partial x_2)^2}\bigg|_{x=\bar{x},\sigma=0}\\
&+\beta\frac{\partial^2 V(h(x))}{(\partial x_1')^2}\cdot\left(\frac{\partial x_1'}{\partial x_2}\right)^2\bigg|_{x=\bar{x},\sigma=0}+\beta\frac{\partial^2 V(h(x))}{\partial x_1'\partial x_2'}\cdot\frac{\partial x_1'}{\partial x_2}\cdot\frac{\partial x_2'}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0}\\
&+\beta\frac{\partial V(h(x))}{\partial x_1'}\cdot\frac{\partial^2 x_1'}{(\partial x_2)^2}\bigg|_{x=\bar{x},\sigma=0}+\beta\frac{\partial^2 V(h(x))}{\partial x_2'\partial x_1'}\cdot\frac{\partial x_2'}{\partial x_2}\cdot\frac{\partial x_1'}{\partial x_2}\bigg|_{x=\bar{x},\sigma=0}\\
&+\beta\frac{\partial^2 V(h(x))}{(\partial x_2')^2}\cdot\left(\frac{\partial x_2'}{\partial x_2}\right)^2\bigg|_{x=\bar{x},\sigma=0}+\beta\frac{\partial V(h(x))}{\partial x_2'}\cdot\frac{\partial^2 x_2'}{(\partial x_2)^2}\bigg|_{x=\bar{x},\sigma=0}
\end{aligned}
$$

$$
\begin{aligned}
V_{\sigma\sigma} \;=\;& \frac{\partial^2\psi(u(x))}{(\partial u(x))^2}\cdot\left(\frac{\partial u(x)}{\partial \sigma}\right)^2\bigg|_{x=\bar{x},\sigma=0}+\frac{\partial\psi(u(x))}{\partial u(x)}\cdot\frac{\partial^2 u(x)}{(\partial \sigma)^2}\bigg|_{x=\bar{x},\sigma=0}\\
&+\beta\frac{\partial^2 V(h(x))}{(\partial x_1')^2}\cdot\left(\frac{\partial x_1'}{\partial \sigma}\right)^2\bigg|_{x=\bar{x},\sigma=0}+\beta\frac{\partial^2 V(h(x))}{\partial x_1'\partial x_2'}\cdot\frac{\partial x_1'}{\partial \sigma}\cdot\frac{\partial x_2'}{\partial \sigma}\bigg|_{x=\bar{x},\sigma=0}\\
&+\beta\frac{\partial V(h(x))}{\partial x_1'}\cdot\frac{\partial^2 x_1'}{(\partial \sigma)^2}\bigg|_{x=\bar{x},\sigma=0}+\beta\frac{\partial^2 V(h(x))}{\partial x_2'\partial x_1'}\cdot\frac{\partial x_2'}{\partial \sigma}\cdot\frac{\partial x_1'}{\partial \sigma}\bigg|_{x=\bar{x},\sigma=0}\\
&+\beta\frac{\partial^2 V(h(x))}{(\partial x_2')^2}\cdot\left(\frac{\partial x_2'}{\partial \sigma}\right)^2\bigg|_{x=\bar{x},\sigma=0}+\beta\frac{\partial V(h(x))}{\partial x_2'}\cdot\frac{\partial^2 x_2'}{(\partial \sigma)^2}\bigg|_{x=\bar{x},\sigma=0}
\end{aligned}
$$

Auf Grund einer besseren Übersichtlichkeit notieren wir im Folgenden die Ableitungen wieder durch entsprechende Indizes. Zur Erinnerung bezeichnet ein Index von $x_1^i x_2^j \sigma^k$ die i-malige Ableitung nach x_1, j-malige Ableitung nach x_2 und k-malige Ableitung in σ-Richtung, ausgewertet im Gleichgewicht $(x, \sigma) = (\bar{x}, 0)$.

Es ergibt sich das folgende Gleichungssystem, aus dem wir die Koeffizienten V_0, V_{x_1}, V_{x_2}, $V_{x_1x_1}$, $V_{x_1x_2}$, $V_{x_2x_2}$ und $V_{\sigma\sigma}$ erhalten.

$$
\begin{aligned}
V_0 &= \psi(u) + \beta V_0 \\
V_{x_1} &= \psi_u \cdot u_{x_1} + \beta V_{x_1}(x_1')_{x_1} + \beta V_{x_2}(x_2')_{x_1} \\
V_{x_2} &= \psi_u \cdot u_{x_2} + \beta V_{x_1}(x_1')_{x_2} + \beta V_{x_2}(x_2')_{x_2} \qquad (3.11) \\
V_{x_1x_1} &= \psi_{uu}\left(u_{x_1}\right)^2 + \psi_u \cdot u_{x_1x_1} + \beta V_{x_1x_1}\left((x_1')_{x_1}\right)^2 + 2\beta V_{x_1x_2}(x_1')_{x_1}(x_2')_{x_1} \\
&\quad + \beta V_{x_1}(x_1')_{x_1x_1} + \beta V_{x_2x_2}\left((x_2')_{x_1}\right)^2 + \beta V_{x_2}(x_2')_{x_1x_1} \\
V_{x_1x_2} &= \psi_{uu} \cdot u_{x_1} \cdot u_{x_2} + \psi_u \cdot u_{x_1x_2} + \beta V_{x_1x_1}(x_1')_{x_1}(x_1')_{x_2} + \beta V_{x_1x_2}(x_1')_{x_1}(x_2')_{x_2} \\
&\quad + \beta V_{x_1}(x_1')_{x_1x_2} + \beta V_{x_2x_1}(x_2')_{x_1}(x_1')_{x_2} + \beta V_{x_2x_2}(x_2')_{x_1}(x_2')_{x_2} + \beta V_{x_2}(x_2')_{x_1x_2} \\
V_{x_2x_2} &= \psi_{uu}\left(u_{x_2}\right)^2 + \psi_u \cdot u_{x_2x_2} + \beta V_{x_1x_1}\left((x_1')_{x_2}\right)^2 + 2\beta V_{x_1x_2}(x_1')_{x_2}(x_2')_{x_2} \\
&\quad + \beta V_{x_1}(x_1')_{x_2x_2} + \beta V_{x_2x_2}\left((x_2')_{x_2}\right)^2 + \beta V_{x_2}(x_2')_{x_2x_2} \\
V_{\sigma\sigma} &= \psi_{uu}\left(u_{\sigma}\right)^2 + \psi_u \cdot u_{\sigma\sigma} + \beta V_{x_1x_1}\left((x_1')_{\sigma}\right)^2 + 2\beta V_{x_1x_2}(x_1')_{\sigma}(x_2')_{\sigma} \\
&\quad + \beta V_{x_1}(x_1')_{\sigma\sigma} + \beta V_{x_2x_2}\left((x_2')_{\sigma}\right)^2 + \beta V_{x_2}(x_2')_{\sigma\sigma}
\end{aligned}
$$

Die Eindeutigkeit und Existenz der Lösung dieses Gleichungssystems wird in der Implementierung wieder indirekt geprüft. So würde die Maple-Routine `linsolve(A,b)` den Null-Vektor oder eine Familie von Lösungen liefern, wenn das Gleichungssystem über- oder unterbestimmt wäre.

Die quadratischen Terme, die bei dieser Approximation einbezogen werden, bewirken, dass die Lösung die Krümmung der Wertefunktion besser wiedergeben kann. Wie in COLLARD und JUILLARD [6] erwähnt, ist dies besonders dann wichtig, wenn eine starke Krümmung vorhanden ist beziehungsweise die wirtschaftliche Unsicherheit sehr groß ist. Die approximative Wertefunktion hängt also vom Risiko ab, da die Standardabweichung der Wahrscheinlichkeitsverteilung in die Berechnung eingebunden wird. Dadurch erhöht sich die Genauigkeit der gegebenen Lösung.

3.2.3 Approximation höherer Ordnung

Um die Fehlergröße weiter zu senken, ist es möglich, analog zur Entwicklung zweiter Ordnung, das Verfahren für beliebige Ordnungen durchzuführen. Dies impliziert, wie

eben erwähnt, auch genauere Approximationen, da die zusätzlichen Terme dem auftretenden Risiko besser gerecht werden können. Nach SCHMITT-GROHÉ [25] ist es ein genereller Vorteil der Perturbations-Methoden, dass der Aufwand dabei in einem überschaubaren Rahmen bleibt, da stets nur lineare Gleichungssysteme zu lösen sind.
Taylor-Entwicklungen zweiter und besonders vierter Ordnung zeigen in der Fehler-Genauigkeit eine höhere Präzision gegenüber den ursprünglich verwendeten Linearisierungs-Methoden. Wie im Einzelnen die Entwicklungen höherer Ordnung errechnet werden, wird in COLLARD und JUILLARD [6] ausgeführt. Dort findet auch eine Auswertung der Approximationen höherer Ordnung statt, die bestätigen, dass dem wirtschaftlich auftretenden Risiko eine entsprechende Bedeutung widerfährt.
Auf eine Ausarbeitung höherer Entwicklungen sei hier verzichtet, da es den Rahmen dieser Arbeit sprengen würde und zudem die durch quadratische Approximation erreichten Genauigkeiten unseren angestrebten Zielen Rechnung tragen.

4 Implementierung der Verfahren

Zur Realisierung der einzelnen Berechnungen existieren unterschiedliche Programme. Die Approximation der Kontrollfunktion und der Dynamik findet dabei ausschließlich in Matlab statt. Für die Taylor-Entwicklung der Wertefunktion wurde Maple verwendet, während die Fixpunkt-Iteration in C++ implementiert wurde. Zudem wurden weitere Programme in Matlab zur Auswertung und zum Vergleich der einzelnen Ergebnisse geschrieben.
Die Matlab-Programme verwenden die Toolbox `Symbolic Math` zur symbolischen Darstellung von Variablen, Matrizen und anderen Ausdrücken. Ebenso ist damit ein direkter Zugang zu Funktionen aus Maple möglich.
Eine Beschreibung der einzelnen Programme findet in diesem Kapitel statt.

4.1 Implementierung der Perturbations-Methode der Kontrollfunktion

In dieser Arbeit sei lediglich auf eine genaue Beschreibung der Programme für die Perturbations-Methode für u und x in BECKER [2] verwiesen. Die ersten fünf der hier aufgeführten, zur Berechnung der Koeffizienten notwendigen Programme sind im Internet unter SCHMITT-GROHÉ und URIBE [26] zu finden.
Zu den wichtigsten Programmen zählen die folgenden m-Files:

- `anal_deriv.m`: Dieses Programm dient der analytischen Berechnung der ersten und zweiten Ableitungen der Funktion f aus der Gleichgewichtsbedingung (2.6) in x- und u-Richtung. Dabei werden die Ableitungen als symbolische Objekte behandelt.

- `num_eval.m`: Hier erfolgt die Berechnung der numerischen Werte der Ableitungen von f im Gleichgewichtspunkt $(\bar{x}, \bar{u})$. Um die Routine ausführen zu können, muss vorher `anal_deriv.m` aufgerufen werden. Dadurch wird sichergestellt, dass die benötigten Variablen für `num_eval.m` zur Verfügung stehen.

- `gxhx.m`: Wie bereits aus dem Programmnamen ersichtlich, werden hier die Matrizen g_x und h_x für die lineare Approximation berechnet.

- `gxx_hxx.m`: Analog werden in diesem Programm die dreidimensionalen Felder g_{xx} und h_{xx} für die quadratische Approximation ermittelt.

- `gss_hss.m`: Zuletzt fehlen noch, ebenfalls für die quadratische Approximation, die Vektoren $g_{\sigma\sigma}$ und $h_{\sigma\sigma}$, deren Berechnung in diesem Programm erfolgt.

- `Beispiel.m`: Mit den entsprechenden Daten für das Beispiel-Modell werden die analytischen Ableitungen berechnet.

- `Beispiel_Zahlen.m`: Diese Routine dient der Festlegung der Parameter und der Gleichgewichtswerte für das Beispiel-Modell, auf die nach Ausführung alle Routinen Zugriff haben.

- `Beispiel_Run.m`: Hiermit werden schließlich die Koeffizienten für die Taylor-Approximationen für das verwendete Beispiel ermittelt.

Für das erweiterte Modell existieren die letzten drei Routinen in etwas abgeänderter Version unter den Namen `Beispiel2.m`, `Beispiel2_Zahlen.m` und `Beispiel2_Run.m`.
Die für die Berechnungen entwickelten Programme bauen auf der in den beiden vorherigen Kapiteln hergeleiteten Theorie auf und wurden dementsprechend implementiert.

4.2 Implementierung der Perturbations-Methode der Wertefunktion

Für die Taylor-Entwicklung der Wertefunktion müssen ebenfalls die notwendigen Koeffizienten ermittelt werden. Dies erfolgt durch Erstellen und Lösen des linearen Gleichungs-

systems (3.11) aus Kapitel 3.2.2, was in dem Maple-Programm `compute_Vcoeff.mw` realisiert wurde. Zur Auswertung der erhaltenen Approximationen dient die Matlab-Routine `show51.m`. Beide Programme werden im Folgenden beschrieben.

4.2.1 Implementierung im Programm compute_Vcoeff.mw

Wie in Kapitel 3.2.2 gezeigt, werden die Koeffizienten der Taylor-Reihe der Wertefunktion durch die Gleichung (2.17)

$$V(x) \quad = \quad E\left[\psi(x, u(x)) + \beta \cdot V(h(x))\right]$$

berechnet, die man aus der Definition 2.11 der Wertefunktion für diskrete stochastische Kontrollprobleme erhält.
In Anlehnung an das später verwendete Modell-Beispiel werden wir eine zweidimensionale Zustandsvariable $x = (x_1, x_2)$ wählen. Es erfolgt eine Taylor-Entwicklung beider Seiten der Gleichung (2.17), die für die linke Seite einfach mit den entsprechenden unbekannten Koeffizienten von V aufgestellt werden kann:

$$\begin{aligned} \widehat{V}(x) \quad = \quad & V_0 + V_{x_1} \cdot x_1 + V_{x_2} \cdot x_2 + \\ & \frac{1}{2} V_{x_1 x_1} \cdot x_1^2 + V_{x_1 x_2} \cdot x_1 x_2 + \frac{1}{2} V_{x_2 x_2} \cdot x_2^2 + \frac{1}{2} V_{\sigma\sigma} \cdot \sigma^2 \end{aligned} \tag{4.1}$$

Dabei stehen die Indizes für die jeweilige Ableitung von V, ausgewertet im Gleichgewichtspunkt $(x, \sigma) = (\bar{x}, 0)$, da die Standardabweichung zur Berechnung des Gleichgewichts auf $\sigma = 0$ gesetzt wurde. Um die späteren Rechnungen zusätzlich zu vereinfachen, wird eine Substitution von (x_1, x_2) durchgeführt, so dass der Entwicklungspunkt, also das Gleichgewicht, zum Ursprung verschoben wird. Dies begründet auch die nachfolgende Änderung von Kontrolle und Dynamik und die obige Reihen-Entwicklung von V, in der der Entwicklungspunkt (x_1, x_2) bei $(0, 0)$ liegt. Da wir die Koordinatenverschiebung sowohl für die Kontrolle und die Dynamik, als auch für die Wertefunktion durchführen, liefert uns die Rechnung das gewünschte Ergebnis.
Auf der rechten Seite der Gleichung (2.17) finden sich die Kontrolle u und die Dynamik $h(x) = (x_1', x_2')$ wieder. Da im Allgemeinen keine exakten Lösungen für u und (x_1', x_2')

bekannt sind, verwenden wir die durch die Perturbations-Methode von SCHMITT-GROHÉ und URIBE erhaltenen Approximationen. Die Taylor-Reihen von (x_1', x_2') und u sind somit nach obiger Transformation durch folgende Polynome gegeben, wobei $[h]^i$, $i = 1, 2$, jeweils die i-te Zeile der Matrix h bedeutet:

$$\begin{aligned}
\widehat{x}_1' &= [h_{x_1}]^1 \cdot x_1 + [h_{x_2}]^1 \cdot x_2 + \frac{1}{2}[h_{x_1x_1}]^1 \cdot x_1^2 + [h_{x_1x_2}]^1 \cdot x_1x_2 \\
&\quad + \frac{1}{2}[h_{x_2x_2}]^1 \cdot x_2^2 + \frac{1}{2}[h_{\sigma\sigma}]^1 \cdot \sigma^2 \\
\widehat{x}_2' &= [h_{x_1}]^2 \cdot x_1 + [h_{x_2}]^2 \cdot x_2 + \frac{1}{2}[h_{x_1x_1}]^2 \cdot x_1^2 + [h_{x_1x_2}]^2 \cdot x_1x_2 \\
&\quad + \frac{1}{2}[h_{x_2x_2}]^2 \cdot x_2^2 + \frac{1}{2}[h_{\sigma\sigma}]^2 \cdot \sigma^2 \\
\widehat{u} &= g_0 + g_{x_1} \cdot x_1 + g_{x_2} \cdot x_2 + \frac{1}{2}g_{x_1x_1} \cdot x_1^2 + g_{x_1x_2} \cdot x_1x_2 + \frac{1}{2}g_{x_2x_2} \cdot x_2^2 + \frac{1}{2}g_{\sigma\sigma} \cdot \sigma^2
\end{aligned}$$

Mit Hilfe dieser Approximationen ist die rechte Seite der obigen Gleichung in eine Taylor-Reihe entwickelbar. Die entsprechenden Koeffizienten der einzelnen Terme erhält man in Maple mit Hilfe der Funktion `coeftayl`, ohne dass die Taylor-Entwicklung dabei berechnet werden muss. Maple verwendet Differenzierungen und Substitutionen, um die entsprechenden Koeffizienten zu ermitteln.

Da auf beiden Seiten der Gleichung Terme mit x_1, x_2, x_1^2, x_2^2 und σ^2 auftreten, müssen die jeweiligen Koeffizienten gleich gesetzt werden, woraus man das gewünschte lineare Gleichungssystem (3.11) erhält. Dieses kann in Maple nun durch die Funktion `linsolve(A,b)` gelöst werden.

Die einzelnen Schritte des Programms lassen sich wie folgt kurz zusammenfassen:

INPUT: Die Koeffizienten für die Taylor-Reihen von x_1, x_2 und u müssen bekannt sein.

- Stelle die Taylor-Reihe V von $V(x)$ gemäss (4.1) auf.
- Stelle die Taylor-Reihe Vh auf, indem die Approximationen $\widehat{x}_1'$ und $\widehat{x}_2'$ in die Taylor-Reihe V für x1 und x2 eingesetzt werden.
- Ersetze u durch die entsprechende Approximation $\widehat{u}$ in $\psi(u)$.

- `Ermittle die Koeffizienten der Taylor-Reihe von` $\psi(\widehat{u}) + \beta V h - V$
- `Stelle das lineare Gleichungssystem auf und löse es.`

OUTPUT: $V0, V1, V2, V11, V12, V22, Vss$

In dem Programm werden für die Bezeichnung der Variablen bisweilen andere Namen verwendet. Die folgende Tabelle 4.1 gibt Aufschluss über die verwendeten Synonyme.

Name im Theorieteil	Name im Programm
x_1, x_1	x1, x2
V_0, V_{x_1}, V_{x_2}	V0, V1, V2
$V_{x_1x_1}$, $V_{x_2x_2}$	V11, V22
$V_{x_1x_2}$, $V_{\sigma\sigma}$	V12, Vss
$\widehat{V}(x)$	V
$V(\widehat{h}(x))$	Vh

Tabelle 4.1: Bezeichnungen im Programm `compute_Vcoeff.mw`

Um unsere Resultate grafisch zu veranschaulichen, benötigen wir eine weitere Routine, die anschließend beschrieben wird.

4.2.2 Auswertung anhand des Programms show51.m

Zur Auswertung der Approximationen wurde das Programm `show51.m` in Matlab geschrieben. Es werden darin nicht nur die numerischen Werte der durch `compute_Vcoeff.mw` erhaltenen Wertefunktion visualisiert, sondern auch die Approximation durch die Fixpunkt-Iteration, die Approximation der Kontrolle und, falls vorhanden, die exakten Lösungen.
Die Berechnungen erfolgen stets für ein Rechteckgitter aus Definition 2.20, wobei die Approximations-Intervalle in x_1- und x_2-Richtung immer die gleiche Anzahl n von äquidistanten Knoten enthalten sollen. Für die entsprechenden Funktionen erhält man somit

eine $n \times n$-Matrix, indem an besagten Gitterknoten ausgewertet wird. Die Fixpunkt-Iteration liefert ebenfalls eine $n \times n$-Matrix, da sie zur Ausführung der Iteration eine derartige Matrix verwendet. Im Programm `show51.m` wurden dabei 51 Gitterpunkte verwendet. Dieses Programm existiert auch in etwas abgeänderter Version mit 251 Gitterpunkten unter dem Namen `show251.m`. Die erstellten Matrizen werden in Matlab eingelesen und können durch die Funktion `interp2` linear interpoliert werden, um zum Beispiel den Wert im Gleichgewichtspunkt zu erhalten.
Für die späteren Auswertungen soll die Differenz zwischen der Approximation der Wertefunktion durch die Perturbations-Methode und durch die Fixpunkt-Iteration ermittelt werden. Falls die exakte Lösung angegeben werden kann, wird ebenfalls der Fehler der einzelnen Methoden berechnet und grafisch dargestellt. Außerdem werden die beiden Methoden und die exakte Lösung einzeln und jeweils paarweise visualisiert.
Der Vollständigkeit halber wird zudem eine Grafik der exakten und approximierten Kontrolle ausgegeben und der entstandene Fehler berechnet. Eine ausführliche Darstellung und Auswertung findet man in BECKER [2].

4.3 Implementierung der Fixpunkt-Iteration

Zur Durchführung der Fixpunkt-Iteration wurde das Programm `fixed_point.cpp` in C++ geschrieben. Es verwendet als Ausgangs-Matrix für die Wertefunktion die von der Matlab-Routine `compute.m` gespeicherten Werte der Perturbations-Methode. Die erhaltenen Approximationen werden wiederum in der Matlab-Methode `show51.m` ausgewertet.

4.3.1 Speicherung im Programm compute.m

Diese Routine dient lediglich zur Speicherung der Werte der exakten Lösungen der Kontrolle und der Wertefunktion, für den Fall, dass diese vorhanden sind, und der approximativen Werte durch die Perturbations-Methoden nach SCHMITT-GROHÉ und URIBE beziehungsweise COLLARD und JUILLARD.
Die benötigten Koeffizienten für die Berechnung der Taylor-Entwicklungen werden zu

Beginn des Programms vorbelegt. Ebenso werden die exakten Lösungen angegeben. Zum späteren Einlesen und Auswerten der approximativen und exakten Werte von u und V werden diese in entsprechenden ASCII-Dateien gespeichert.
Die Routine lässt sich in folgenden Schritten kurz zusammenfassen:

INPUT: Die benötigten Koeffizienten der Taylor-Entwicklungen müssen im Vorfeld berechnet werden; zusätzlich müssen σ und der gewünschte Approximations-Bereich angegeben werden.

- Belege die Variablen für die Koeffizienten
- Berechne die Werte von u und V in den Gitterpunkten mit Hilfe der Taylor-Entwicklungen
- Falls eine exakte Lösung berechenbar ist: Werte die exakte Formel für u und V in den Gitterpunkten aus.
- Speichere die so erhaltenen $n \times n$-Matrizen in separaten Dateien.

OUTPUT: ASCII-Dateien für U_exakt, V_exakt, U_pert und V_pert

Wie die Namensgebung vermuten lässt, enthalten die Dateien U_exakt und V_exakt die Werte für die exakte Kontrolle beziehungsweise Wertefunktion, die nur für den Spezialfall angegeben werden können. Die Dateien U_pert und V_pert beinhalten die Matrizen, die sich bei der approximativen Berechnung von u und V durch die Perturbations-Methoden ergeben.

4.3.2 Implementierung im Programm fixed_point.cpp

In diesem Programm wird die in Kapitel 2.3 eingeführte Fixpunkt-Iteration implementiert. Da für den Algorithmus eine Start-Matrix benötigt wird, muss zunächst die Approximation V_pert aus der entsprechenden Datei geladen werden. Mit diesen Werten wird

die Iteration durchgeführt.
Dazu wird für jeden Gitterknoten $x = ((x_1)_i, (x_2)_j),\ i,j = 1,\cdots,n$ die rechte Seite der Gleichung (2.19)

$$E\left[\psi(x,\widehat{u}(x)) + \beta \cdot V_{\widehat{u}}(\varphi(x,\widehat{u}(x),z))\right] = \psi(x,\widehat{u}(x)) + \beta \cdot \sum_{k=1}^{m} V_{\widehat{u}}(\varphi_k(x)) \cdot p_k, \quad (4.2)$$

mit p_k der Wahrscheinlichkeit im Punkt z_k, ausgewertet.
Die dabei erhaltenen Werte werden mit den vorherigen verglichen. Überschreitet die maximal auftretende Differenz einen gewissen Schwellwert, so wird die Iteration wiederholt. Andernfalls wird die erhaltene Matrix mit der Bezeichnung V_fix in einer ASCII-Datei als Approximation gespeichert.
Der Algorithmus stellt sich folgendermaßen dar:

INPUT: Die Approximationen U_pert und V_pert müssen in ASCII-Dateien gespeichert vorliegen; zusätzlich müssen σ, der gewünschte Approximations-Bereich und die gewünschte Genauigkeit angegeben werden.

- Belege die Matrizen U_pert und V_pert mit den gespeicherten Werten.
- Setze $V\_fix = V\_pert$.
- Solange die gewünschte Genauigkeit nicht erreicht ist, wiederhole:

  Für $i = 1,\ldots,n$ wiederhole:

  Für $j = 1,\ldots,n$ wiederhole:

  · Setze x= $((\text{x1})_i, (\text{x2})_j)$

  · Berechne $\varphi_k = \varphi(\text{x}, U\_pert[j][i], z_k)$ für $k = 1,\ldots,m$

  · Interpoliere V_fix in φ_k

  · Berechne (4.2):

$$V\_fix[j][i] = \psi(\text{x}, U\_pert[j][i]) + \beta \cdot \sum_{k=1}^{m} V\_fix(\varphi_k) \cdot p_k$$

`STOP` j`, STOP` i

OUTPUT: `ASCII-Datei für` V_fix

Da V_fix als $n \times n$-Matrix gespeichert ist, müssen deren Werte nicht notwendigerweise an den Stellen $\varphi_k(x)$ gegeben sein. Deshalb ist, wie bereits erwähnt, eine Interpolation notwendig. Hierzu werden affin bilineare Funktionen verwendet, wie in Kapitel 2.3 erläutert wurde.

4.3.3 Auswertung anhand des Programms show51.m

Die Matlab-Routine `show51.m` dient, wie bereits oben beschrieben, lediglich der Auswertung der berechneten Approximationen. Das Hauptaugenmerk wird dabei auf die beiden Approximationen der Wertefunktion gelegt. Falls möglich, findet aber auch ein Vergleich der approximierten und exakten Kontrolle statt.
Mit den Resultaten, die aus den Visualisierungen und Fehlerberechnungen gezogen werden können, beschäftigt sich das folgende Kapitel.

5 Anwendung der Verfahren auf das Modell-Beispiel

Zur Überprüfung, wie gut die beschriebenen Verfahren die exakten Lösungen approximieren, werden diese nun an einem Beispiel aus der Volkswirtschaft getestet.
Da für das Grundmodell die exakte Lösung berechnet werden kann, was im ersten Abschnitt geschieht, ist ein direkter Vergleich möglich, welcher im zweiten Teil dieses Kapitels angestellt wird. Der dritte Abschnitt befasst sich mit dem erweiterten Modell, bei dem dem stochastischen Anteil eine stärkere Gewichtung widerfährt. Da hier keine exakte Lösung ermittelt werden kann, werden lediglich Vermutungen bezüglich der Fehlergröße geäußert. Gut zu beobachten ist dabei, wie sich Änderungen in der Zielfunktion auf das Verhalten der Approximationen auswirken.

5.1 Darstellung des Modells

Zunächst wird das verwendete Beispiel vorgestellt und die exakte Lösung angegeben. Zudem muss der Gleichgewichtspunkt berechnet werden, da er die Grundlage für die Perturbations-Verfahren bildet. Hierfür bedienen wir uns der Optimalitätsbedingungen aus Satz 2.14, die auch für die Perturbations-Methode der Kontrolle benötigt werden.

5.1.1 Exakte Lösung des Grundmodells

Als Grundmodell dient uns ein stochastisches, volkswirtschaftliches Wachstumsmodell aus GRÜNE [13].
Die Zielfunktion ist hierbei gegeben durch:

$$\max \sum_{t=0}^{\infty} \beta^t \ \log u_t \tag{5.1}$$

und das Kontrollsystem durch:

$$\varphi(x,u,z) \;=\; \begin{pmatrix} A\,e^{x_2}\,x_1^{\alpha} - u \\ \rho x_2 + z \end{pmatrix}. \tag{5.2}$$

Der Kontrollvektor u ist eindimensional und der Zustandsvektor zweidimensional, wobei x_1 der deterministische und x_2 der stochastische Anteil ist.
Das Beispiel ist einerseits deshalb so gut für die Anwendung der Verfahren geeignet, weil die Lösung genügend glatt ist für $x_1 \neq 0$, andererseits, weil, wie bereits erwähnt, eine exakte Lösung für das Modell ermittelt werden kann. Diese ist folgendermaßen gegeben:

$$V(x) = B \;+\; C\;\ln x_1 \;+\; D\,x_2 \qquad \text{und} \tag{5.3}$$

$$u(x) = (1-\alpha\beta)\,A\,e^{x_2}\,x_1^{\alpha}. \tag{5.4}$$

mit

$$\begin{aligned} B &= \frac{\ln((1-\beta\alpha)\,A) + \frac{\beta\alpha}{1-\beta\alpha}\ln(\beta\alpha A)}{(1-\beta)} \\ C &= \frac{\alpha}{1-\alpha\beta} \\ D &= \frac{1}{(1-\alpha\beta)(1-\rho\beta)}. \end{aligned}$$

Um die entsprechenden Optimalitätsbedingungen aufzustellen und das Gleichgewicht und die notwendigen Koeffizienten zu berechnen, verwenden wir folgende Werte in dem Modell-Beispiel: $A = 5$, $\alpha = 0.34$, $\beta = 0.95$ und $\rho = 0.9$.
Außerdem ist die Normalisierung der Zufallsvariablen z nötig, die $N(0,\sigma)$-verteilt ist. Hier wird die Standardabweichung $\sigma = 0.008$ gewählt. Dass die Normalisierung mit Hilfe der Standardabweichung erfolgt, wurde in Kapitel 2.1.1 erläutert, und man erhält mit $\varepsilon = \frac{z}{\sigma}$ eine $N(0,1)$-verteilte Zufallsvariable, die zudem unabhängig von x ist. Für den deterministischen und stochastischen Anteil des Zustandsvektors ergibt sich deshalb aus (5.2):

$$\begin{aligned} x_1' &= A\,e^{x_2}x_1^{\alpha} - u \\ x_2' &= \rho x_2 + \sigma\varepsilon, \end{aligned}$$

wobei wiederum Variablen aus der Zeitperiode $(t+1)$ aus Notationsgründen mit einem $'$ gekennzeichnet sind, während Variablen aus der Zeitperiode t keine Kennzeichnung erhalten.
Die exakte optimale Wertefunktion sieht für diese Werte folgendermaßen aus:

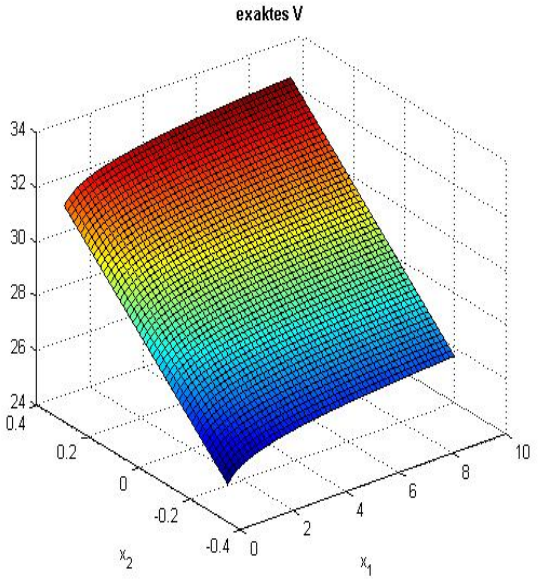

Abbildung 5.1: Exakte Wertefunktion auf dem Rechteck $\Omega = [0.3, 8.8] \times [-0.32, 0.32]$

Da für die Perturbations-Methoden jeweils ein Entwicklungspunkt benötigt wird, berechnen wir hierfür im Folgenden den Gleichgewichtspunkt.

5.1.2 Berechnung des Gleichgewichtspunktes

Dazu ist die Funktion f aus der Gleichgewichtsbedingung (2.6) nötig, die man mit Hilfe der Hamilton-Funktion aus Definition 2.12 erhält. Diese hat für unser Beispiel die folgende Form:

$$H^t(x, u, \lambda) \quad = \quad \beta \log u + \lambda_1^{'} \cdot (Ae^{x_2} x_1^{\alpha} - u) + \lambda_2^{'} \cdot \rho x_2. \tag{5.5}$$

Mittels diverser Umformungen des diskreten Maximumsprinzips aus Satz 2.14, was genauer in BECKER [2] nachzulesen ist, erhält man:

$$\widetilde{f} := \begin{pmatrix} x_1^{'} - A\, e^{x_2}\, x_1^{\alpha} + u \\ x_2^{'} - \rho\, x_2 \\ \beta\, \frac{1}{u'}\, \alpha\, A\, e^{x_2^{'}}\, x_1^{'(\alpha-1)} - \frac{1}{u} \end{pmatrix} = 0 \tag{5.6}$$

Diese Funktion $\widetilde{f}$ entspricht nicht exakt der Funktion f aus Definition 2.16, bezieht aber alle notwendigen Optimalitätsbedingungen ein.
Da im Gleichgewicht

$$\begin{aligned} x_1^{'} &= x_1, \\ x_2^{'} &= x_2, \\ u^{'} &= u \end{aligned}$$

gilt, folgt aus der Gleichung (5.6) aus $x_2^{'} - \rho\, x_2 = 0$, dass $\bar{x}_2 = 0$ ist.
Damit erhalten wir für die erste Zeile der obigen Gleichung (5.6)

$$u = A \cdot 1 \cdot x_1^{\alpha} - x_1, \tag{5.7}$$

was in die verbleibende Zeile eingesetzt die Gleichung

$$\beta\, \alpha\, A\, x_1^{(\alpha-1)}\, \frac{1}{A\, x_1^{\alpha} - x_1} = \frac{1}{A\, x_1^{\alpha} - x_1}$$

liefert. Diese lässt sich nun folgendermaßen umformulieren:

$$\begin{aligned} &\iff && \beta\, \alpha\, A\, x_1^{(\alpha-1)} = 1 \\ &\iff && \beta\, \alpha\, A = x_1^{(1-\alpha)} \\ &\iff && \left(\, \beta\, \alpha\, A \,\right)^{\frac{1}{1-\alpha}} = x_1 \end{aligned}$$

Somit erhalten wir für x_1 durch Einsetzen der entsprechenden Werte:

$$x_1 \approx 2.067344815$$

Mit Hilfe der Gleichung (5.7) lässt sich damit u berechnen:

$$u \approx 4.333103529$$

Das Gleichgewicht ist somit gegeben durch die folgenden Werte:

$$\left[\ \bar{x};\ \ \bar{u}\ \right] \ = \ \left[\ (2.067344815, 0);\ \ 4.333103529\ \right]$$

Um zu verdeutlichen, was mit dem ermittelten Gleichgewicht ausgesagt wird, sollte die Interpretation der Variablen aus SCHMITT-GROHÉ und URIBE [24] herangezogen werden. Die Größe x_1 beschreibt dabei das vorhandene Kapital eines Wirtschaftssubjekts und x_2 die technologische Veränderung. Die Kontrolle u spiegelt das Konsumverhalten der Beteiligten wider. Während sich der Kapitalstock bei ca. 2.07 Einheiten und der Konsum bei ca. 4.33 Einheiten einstellt, darf die Technologie keinen Veränderungen unterliegen. Dies ergibt sich auf Grund unseres Ansatzes, dass zur Berechnung des Gleichgewichtspunktes die Störung von außen, also die technologische Veränderung, auf 0 gesetzt wurde. Deshalb zeigt sich auch bei der späteren Abwandlung des Modells und größerer Bedeutung des stochastischen Anteils keine Änderung im Gleichgewichtspunkt.

5.2 Auswertung des Modells

Dass für unser Modell-Beispiel eine exakte Lösung existiert, eröffnet uns die Möglichkeit, die Fehler der Approximationen direkt betrachten zu können. Dafür benötigen wir für die Perturbations-Methoden noch die entsprechenden Koeffizienten.

5.2.1 Numerische Approximation für die Kontrollfunktion

Zunächst muss das Modell in die Form gebracht werden, die vom Algorithmus von SCHMITT-GROHÉ und URIBE gefordert wird.
Hierzu sind, neben der Normalisierung der Zufallsvariablen, wieder das diskrete Maximumsprinzip aus Satz 2.14 und die Hamilton-Funktion aus Definition 2.12 nötig, welche bereits im vorherigen Abschnitt zur Berechnung des Gleichgewichts in (5.5) angegeben

wurde:

$$H^t(x,u,\lambda) \;=\; \beta \log u + \lambda_1' \cdot (Ae^{x_2} x_1^{\alpha} - u) + \lambda_2' \cdot \rho x_2.$$

Mit der resultierenen Funktion $\widetilde{f}$ aus (5.6) können die Koeffizienten ermittelt werden. Dazu werden die von BECKER [2] bereitgestellten Routinen verwendet. In der angesprochenen Arbeit findet sich auch eine Beschreibung zur genauen Handhabung der Programme `Beispiel.m`, `Beispiel_Zahlen.m` und `Beispiel_Run.m`.
Sie liefern uns die folgenden Werte:

- Koeffizienten der linearen Terme:

$$g_x = \begin{pmatrix} 0.7126 & 4.3331 \end{pmatrix}$$

$$h_x = \begin{pmatrix} 0.3400 & 2.0673 \\ 0 & 0.9 \end{pmatrix}.$$

- Koeffizienten der in x quadratischen Terme:

$$g_{xx}(:,:,1) = \begin{pmatrix} -0.2275 & 0.7126 \end{pmatrix}$$

$$g_{xx}(:,:,2) = \begin{pmatrix} 0.7126 & 4.3331 \end{pmatrix}$$

$$h_{xx}(:,:,1) = \begin{pmatrix} -0.1085 & 0.3400 \\ 0 & 0 \end{pmatrix}$$

$$h_{xx}(:,:,2) = \begin{pmatrix} 0.3400 & 2.0673 \\ 0 & 0 \end{pmatrix}.$$

- Koeffizienten der in σ quadratischen Terme:

$$g_{\sigma\sigma} = \begin{pmatrix} 1.5677e-010 \end{pmatrix}$$

$$h_{\sigma\sigma} = \begin{pmatrix} -0.1568e-009 \\ 0 \end{pmatrix}.$$

Betrachtet man die letzten beiden Terme $g_{\sigma\sigma}$ und $h_{\sigma\sigma}$, so lässt dies vermuten, dass sie in dem Beispiel nur eine sehr untergeordnete Rolle spielen, da alle Werte im Bereich 10^{-10} liegen, während sich die restlichen Werte im ganzzahligen Bereich aufhalten. Deshalb werden später weitere Fälle behandelt, in denen der stochastische Anteil eine stärkere Gewichtung findet.
Mit den oben berechneten Werten erhält man nun für die quadratische Approximation von u die folgende Funktion:

$$\begin{aligned}
\widehat{u} &= 4.333103529 + 0.7126 \cdot [(x_1 - 2.067344815)] + 4.3331 \cdot [(x_2 - 0)] \\
&\quad + \frac{1}{2} \cdot (-0.2275) \cdot [(x_1 - 2.067344815)] \cdot [(x_1 - 2.067344815)] \\
&\quad + \frac{1}{2} \cdot 0.7126 \cdot [(x_2 - 0)] \cdot [(x_1 - 2.067344815)] \\
&\quad + \frac{1}{2} \cdot 0.7126 \cdot [(x_1 - 2.067344815) \cdot [(x_2 - 0)] \\
&\quad + \frac{1}{2} \cdot 4.3331 \cdot [(x_2 - 0)] \cdot [(x_2 - 0)] \\
&\quad + \frac{1}{2} \cdot 0 \cdot [0.008] \cdot [0.008].
\end{aligned}$$

Die Approximation der Dynamik liefert

$$\begin{aligned}
\widehat{x}_1' &= 2.067344815 + 0.3400 \cdot [(x_1 - 2.067344815)] + 2.0673 \cdot [(x_2 - 0)] \\
&\quad + \frac{1}{2} \cdot (-0.1085) \cdot [(x_1 - 2.067344815)] \cdot [(x_1 - 2.067344815)] \\
&\quad + \frac{1}{2} \cdot 0.34 \cdot [(x_2 - 0)] \cdot [(x_1 - 2.067344815)] \\
&\quad + \frac{1}{2} \cdot 0.34 \cdot [(x_1 - 2.067344815) \cdot [(x_2 - 0)] \\
&\quad + \frac{1}{2} \cdot 2.0673 \cdot [(x_2 - 0)] \cdot [(x_2 - 0)] \\
&\quad + \frac{1}{2} \cdot 0 \cdot [0.008] \cdot [0.008],
\end{aligned}$$

$$
\begin{aligned}
\widehat{x}_2' \;=\; & 0 + 0 \cdot [(x_1 - 2.067344815)] + 0.9 \cdot [(x_2 - 0)] \\
& + \frac{1}{2} \cdot 0 \cdot [(x_1 - 2.067344815)] \cdot [(x_1 - 2.067344815)] \\
& + \frac{1}{2} \cdot 0 \cdot [(x_2 - 0)] \cdot [(x_1 - 2.067344815)] \\
& + \frac{1}{2} \cdot 0 \cdot [(x_1 - 2.067344815) \cdot [(x_2 - 0)] \\
& + \frac{1}{2} \cdot 0 \cdot [(x_2 - 0)] \cdot [(x_2 - 0)] \\
& + \frac{1}{2} \cdot 0 \cdot [0.008] \cdot [0.008] \\
& + \sigma\varepsilon'.
\end{aligned}
$$

Bemerkung 5.1

(i) Bei der Approximation von x_2' kann man sehr gut erkennen, dass die eigentliche Approximation ungefähr Null ist und der hintere stochastische Teil die größte Rolle spielt.

(ii) Bei der Approximation von u und x_1' stellt man fest, dass jeweils der erste Term das meiste Gewicht in die Berechnung einbringt, vorausgesetzt die Werte für x_1 und x_2 liegen genügend nahe beim Gleichgewicht. Daraus lässt sich als Vermutung folgern, dass die Approximationen in einer genügend kleinen Umgebung um das Gleichgewicht gute Ergebnisse liefern müssten, weil die Funktion hinreichend glatt ist.

5.2.2 Numerische Approximation für die Wertefunktion

Für die Taylor-Entwicklung der Wertefunktion V wird die Gleichung (2.17)

$$V(x) = E\left[\psi(x, u(x)) + \beta \cdot V(h(x))\right]$$

benötigt, die man aus der Definition 2.11 der Wertefunktion für diskrete stochastische Kontrollprobleme erhält.

Da $\psi(x, u(x)) = logu(x)$, lautet sie für unser Modell-Beispiel:

$$V(x) = E\left[logu(x) + \beta \cdot V(h(x))\right] \tag{5.8}$$

mit $h(x)$ = (x_1', x_2').

Wie bereits erwähnt, werden dabei, da im Allgemeinen keine exakten Lösungen für die Kontrolle u und die Dynamik $h(x)$ vorhanden sind, die Approximationen $\widehat{u}(x)$ und $\widehat{h}(x) = (\widehat{x}_1', \widehat{x}_2')$ durch die Perturbations-Methode nach SCHMITT-GROHÉ und URIBE verwendet.

Für die beidseitige Taylor-Entwicklung der Gleichung (5.8) nutzen wir deshalb aus, dass

$$\begin{aligned} 0 &= \frac{\partial \widehat{u}}{\partial \sigma} = \frac{\partial^2 \widehat{u}}{(\partial \sigma)^2} = \frac{\partial \widehat{x}_1'}{\partial \sigma} = \frac{\partial^2 \widehat{x}_1'}{(\partial \sigma)^2} \\ &= \frac{\partial \widehat{x}_2'}{\partial x_1} = \frac{\partial^2 \widehat{x}_2'}{(\partial x_1)^2} = \frac{\partial^2 \widehat{x}_2'}{\partial x_1 \partial x_2} = \frac{\partial^2 \widehat{x}_2'}{(\partial x_2)^2} = \frac{\partial^2 \widehat{x}_2'}{(\partial \sigma)^2}. \end{aligned}$$

Das resultierende lineare Gleichungssystem im Sinne von (3.11), das aus termweiser Identifikation der Koeffizienten hervorgeht und aus dem sich die fehlenden Werte von $V_0, V_{x_1}, V_{x_2}, V_{x_1x_1}, V_{x_1x_2}, V_{x_2x_2}$ und $V_{\sigma\sigma}$ berechnen lassen, hat damit folgende Gestalt:

$$\begin{aligned} V_0 &= log\widehat{u}_0 + \beta V_0 \\ V_{x_1} &= \frac{1}{\widehat{u}_0} \cdot \widehat{u}_{x_1} + \beta V_{x_1} \cdot (\widehat{x}_1')_{x_1} \\ V_{x_2} &= \frac{1}{\widehat{u}_0} \cdot \widehat{u}_{x_2} + \beta V_{x_1} \cdot (\widehat{x}_1')_{x_2} + \beta V_{x_2} \cdot (\widehat{x}_2')_{x_2} \\ V_{x_1x_1} &= \frac{-1}{\widehat{u}_0^2} \cdot (\widehat{u}_{x_1})^2 + \frac{1}{\widehat{u}_0} \cdot \widehat{u}_{x_1x_1} + \beta V_{x_1x_1} \cdot ((\widehat{x}_1')_{x_1})^2 + \beta V_{x_1} \cdot (\widehat{x}_1')_{x_1x_1} \\ V_{x_1x_2} &= \frac{-1}{\widehat{u}_0^2} \cdot \widehat{u}_{x_1} \cdot \widehat{u}_{x_2} + \frac{1}{\widehat{u}_0} \cdot \widehat{u}_{x_1x_2} + \beta V_{x_1x_1} \cdot (\widehat{x}_1')_{x_1} \cdot (\widehat{x}_1')_{x_2} \\ &\quad + \beta V_{x_1x_2} \cdot (\widehat{x}_1')_{x_1} \cdot (\widehat{x}_2')_{x_2} + \beta V_{x_1} \cdot (\widehat{x}_1')_{x_1x_2} \\ V_{x_2x_2} &= \frac{-1}{\widehat{u}_0^2} \cdot (\widehat{u}_{x_2})^2 + \frac{1}{\widehat{u}_0} \cdot \widehat{u}_{x_2x_2} + \beta V_{x_1x_1} \cdot ((\widehat{x}_1')_{x_2})^2 + 2\beta V_{x_1x_2} \cdot (\widehat{x}_1')_{x_2} \cdot (\widehat{x}_2')_{x_2} \\ &\quad + \beta V_{x_1} \cdot (\widehat{x}_1')_{x_2x_2} + \beta V_{x_2x_2} \cdot ((\widehat{x}_2')_{x_2})^2 \\ V_{\sigma\sigma} &= \beta V_{x_2x_2} \cdot ((\widehat{x}_2')_{\sigma})^2 \end{aligned} \tag{5.9}$$

Die approximativen Funktionen $\widehat{x}_1'$ und $\widehat{x}_2'$ werden für die Taylor-Approximation von $V(\widehat{h}(x))$ wie folgt verwendet:

$$\begin{aligned} V_0 &+ V_{x_1}(0.34x_1 + 2.0673x_2 - 0.05425x_1^2 + 0.34x_1x_2 + 1.03365x_2^2) + 0.9V_{x_2}x_2 \\ &+ \frac{1}{2}V_{x_1x_1}(0.34x_1 + 2.0673x_2 - 0.05425x_1^2 + 0.34x_1x_2 + 1.03365x_2^2)^2 \\ &+ 0.9V_{x_1x_2}(0.34x_1 + 2.0673x_2 - 0.05425x_1^2 + 0.34x_1x_2 + 1.03365x_2^2)x_2 \\ &+ 0.405V_{x_2x_2}x_2^2 + \frac{1}{2}V_{\sigma\sigma}\sigma^2. \end{aligned}$$

Hierbei bedienen wir uns der beschriebenen Substitution, so dass der Entwicklungspunkt im Ursprung liegt. Darin findet sich auch der Grund, dass die Funktionen $\widehat{x}_1'$ und $\widehat{x}_2'$ nicht exakt mit den im vorherigen Abschnitt ermittelten übereinstimmen. Für V erhalten wir dadurch die Reihe

$$\widehat{V}(x) = V_0 + V_{x_1}x_1 + V_{x_2}x_2 + \frac{1}{2}V_{x_1x_1}x_1^2 + V_{x_1x_2}x_1x_2 + \frac{1}{2}V_{x_2x_2}x_2^2 + \frac{1}{2}V_{\sigma\sigma}\sigma^2.$$

Wird die vorherige Substitution wieder rückgängig gemacht, ergibt sich die gewünschte Taylor-Enwicklung von V:

$$\begin{aligned} \widehat{V}(x) &= V_0 + V_{x_1}(x_1 - 2.0673) + V_{x_2}x_2 + \frac{1}{2}V_{x_1x_1}(x_1 - 2.0673)^2 \\ &+ V_{x_1x_2}(x_1 - 2.0673)x_2 + \frac{1}{2}V_{x_2x_2}x_2^2 + \frac{1}{2}V_{\sigma\sigma}\sigma^2 \end{aligned}$$

Aus dem linearen Gleichungssystem (5.9) erhält man mit Hilfe der Maple-Routine `compute_Vcoeff.mw` die folgenden Werte für das Modell-Beispiel:

$$\begin{aligned} V_0 &= 29.32566 \\ V_{x_1} &= 0.24291 \\ V_{x_2} &= 10.18671 \\ V_{x_1x_1} &= -0.11749 \\ V_{x_1x_2} &= V_{x_2x_1} = 0.14047 \cdot 10^{-4} \\ V_{x_2x_2} &= 0.47826 \cdot 10^{-3} \\ V_{\sigma\sigma} &= 0 \end{aligned}$$

Somit ist die Approximation von $V(x)$ gegeben durch

$$\begin{aligned}\widehat{V}(x) &= 29.32566 + 0.24291 \cdot (x_1 - \bar{x}_1) + 10.18671 \cdot (x_2 - \bar{x}_2) \\ &\quad + \frac{1}{2}(-0.11749)(x_1 - \bar{x}_1)^2 + 0.14047 \cdot 10^{-4} \cdot (x_1 - \bar{x}_1)(x_2 - \bar{x}_2) \\ &\quad + \frac{1}{2}0.47826 \cdot 10^{-3} \cdot (x_2 - \bar{x}_2)^2\end{aligned}$$

mit dem Entwicklungspunkt $(\bar{x}_1, \bar{x}_2) = (2.0673, 0)$.

Bemerkung 5.2 *Bei der Approximation von V lässt sich wie bei $\widehat{u}$ und $\widehat{x}_1'$ feststellen, dass sich der erste Term, zumindest in kleiner Umgebung um das Gleichgewicht, am stärksten bemerkbar machen wird. Deshalb ist die Approximation nahe des Gleichgewichts vermutlich sehr genau, da die Funktion hinreichend glatt ist.*

Analog den Approximationen der Kontrolle und Dynamik entfällt hier der stochastische Anteil durch $V_{\sigma\sigma} = 0$. Jedoch bleibt - im Gegensatz zu Kontrolle und Dynamik - auch im erweiterten Modell die Bedeutung relativ gering.
Es wurden nun alle notwendigen Approximationen ermittelt. Widmen wir uns also der Genauigkeit und Verwendbarkeit der beschriebenen Verfahren.

5.2.3 Auswertung der Approximationen

Im Folgenden sollen die Dynamik und die Kontrolle vernachlässigt werden, da wir insbesondere an der Approximation der Wertefunktion interessiert sind.
Hierbei wollen wir die zwei Bereiche $[1.0, 4.0]\times[-0.32, 0.32]$ und $[0.3, 8.8]\times[-0.32, 0.32]$ betrachten.

Das Gebiet $\Omega = [\mathbf{1.0}, \mathbf{4.0}] \times [\mathbf{-0.32}, \mathbf{0.32}]$

Zunächst wählen wir das kleinere Gebiet, auf dem V approximiert werden soll. Dabei sollen die entsprechenden Auswertungen sowohl in x_1- als auch in x_2-Richtung jeweils mit 51 Knoten, also einer Gesamtzahl von 2601 Knoten, durchgeführt werden. Wir erhalten die Schrittweiten $h_1 = 0.06$ und $h_2 = 0.0128$. Zusätzlich brauchen wir für die Fixpunkt-Iteration die Zufallsvariable z. Hierfür wird vorerst ein Intervall von $[-4\sigma, 4\sigma]$

mit 11 Knoten gewählt, woraus eine Schrittweite von $h_z = 0.0064$ resultiert.
Für diese Werte weist die Approximation der Kontrolle u an den Gebietsrändern bereits einen geringen Unterschied zur exakten Kontrolle auf. Der Fehler liegt in einem Bereich von unter 0.3 und wird in der Abbildung 5.2 dargestellt. In einem Bereich nahe des Gleichgewichts ist dabei die Perturbations-Methode für u noch recht genau ist, da zum Beispiel auf dem Bereich $[1.8, 2.5] \times [-0.05, 0.05]$ nur ein Fehler von etwa $3.1 \cdot 10^{-3}$ auftritt. An den Bereichsgrenzen hingegen, an denen sich besagte Werte von ca. 0.3 ergeben, weist die Approximation bereits zu große Ungenauigkeiten auf.

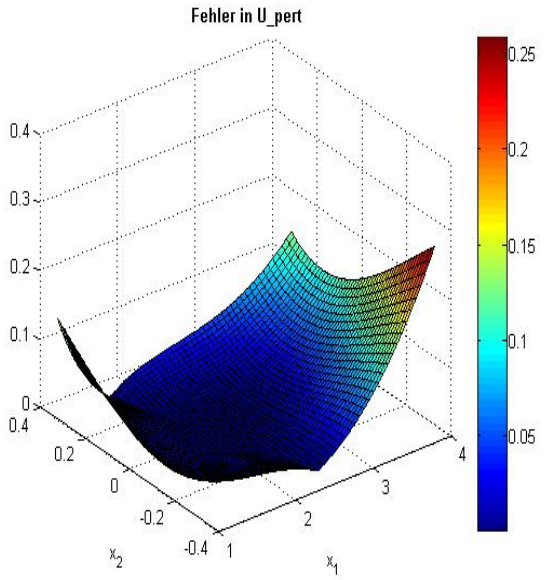

Abbildung 5.2: Fehler der Perturbations-Methode für die Kontrolle auf dem Rechteck $\Omega = [1.0, 4.0] \times [-0.32, 0.32]$

Die Approximationen von V dagegen lassen sich auf dem Rechteck $[1.0, 4.0] \times [-0.32, 0.32]$ von der exakten Lösung mit dem bloßen Auge kaum unterscheiden. Deshalb wollen wir die Abbildungen 5.3 und 5.4 betrachten, die den Fehler der Perturbations-Methode beziehungsweise der Fixpunkt-Iteration in den jeweiligen Gitterpunkten zeigen.
Für die Perturbations-Methode liegt dabei der Fehler stets unterhalb von 0.1. Die Genauigkeit wird also im Vergleich zur approximativen Kontrolle erhöht. Analog zur Berechnung der Kontrolle ist die Approximation nahe des Gleichgewichts sehr genau. Der

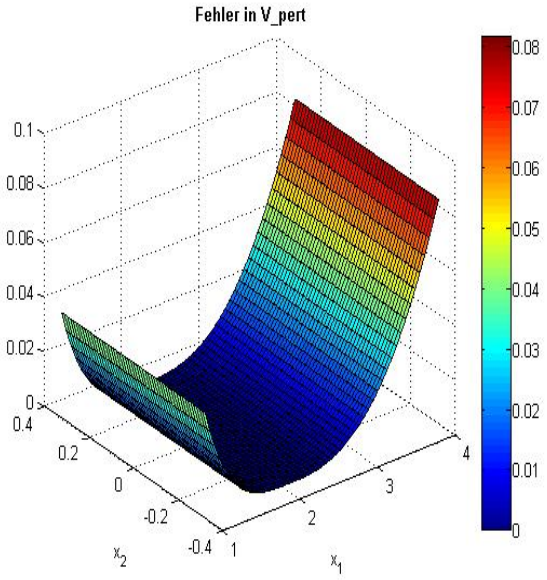

Abbildung 5.3: Fehler der Perturbations-Methode für die Wertefunktion auf dem Rechteck $\Omega = [1.0, 4.0] \times [-0.32, 0.32]$

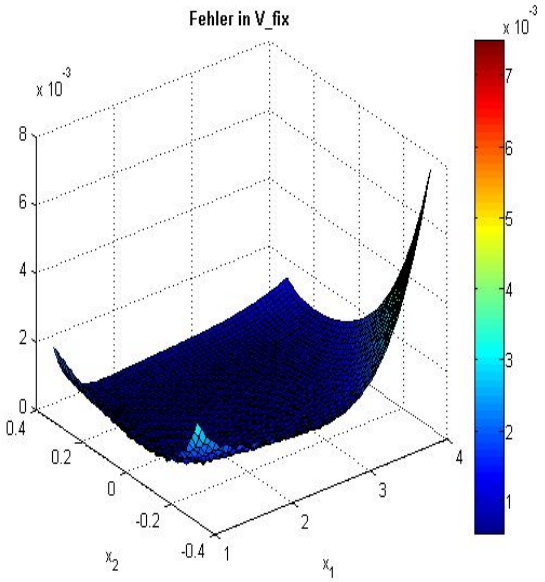

Abbildung 5.4: Fehler der Fixpunkt-Iteration auf $\Omega = [1.0, 4.0] \times [-0.32, 0.32]$

Fehler liegt im Abschnitt $[1.8, 2.5] \times [-0.05, 0.05]$ bei nur etwa $1.4 \cdot 10^{-3}$, im Bereich $[2.0, 2.1] \times [-0.01, 0.01]$ verringert er sich sogar auf $1.7 \cdot 10^{-5}$. Die Approximation der Kontrolle ist auf letzterem Rechteck sogar bis auf $4.7 \cdot 10^{-6}$ genau.
Widmen wir uns der Fixpunkt-Iteration, so erkennt man, dass der Fehler in einem sehr kleinen Bereich von $8 \cdot 10^{-3}$ liegt. Derartig "hohe" Fehler werden allerdings nur an den Gebietsrändern, und zwar inbesondere zum Punkt $(4.0, -0.32)$ hin angenommen. Ein Grund dafür liegt darin, dass zur Durchführung der Iteration die Kontrolle benötigt wird, die ebenfalls zu diesem Punkt hin ihre größte Ungenauigkeit aufweist.
Zudem ist bei der Fixpunkt-Iteration anzumerken, dass der Fehler zum Gleichgewicht hin nicht so stark abnimmt, wie das bei den Perturbations-Methoden der Fall ist. Im Gleichgewichtspunkt selbst ist der Fehler sogar höher als bei der Perturbations-Methode, was bei einer Größe von ca. $7 \cdot 10^{-4}$ aber unwesentlich ist. Natürlich liegt diese Tatsache auch in der Funktionsweise der Perturbations-Methoden begründet, die den Gleichgewichtspunkt immer als Ausgangspunkt nehmen und somit in diesem Punkt und seiner - meist jedoch recht kleinen - Umgebung sehr genaue Werte liefern.
Auch wenn man den Unterschied der beiden Approximationen untersucht, so werden Werte von 0.1 nicht überschritten, siehe Abbildung 5.5. Die Grafik unterscheidet sich dabei unwesentlich von der, die man beim Fehler der Perturbations-Methode erhält. Da diese auch den wesentlichen Fehler verursacht, und durch die Fixpunkt-Iteration nur ein zusätzlicher Unterschied von unter $8 \cdot 10^{-3}$ hinzukommt, ist das Verhalten anschaulich klar.
Im Allgemeinen kann man sagen, dass für dieses Gebiet die Approximationen der Perturbations-Methode und der Fixpunkt-Iteration recht genau sind. In beiden Fällen ist der Fehler um das Gleichgewicht gering. Erst zu den Rändern steigt er an, was sich bei der Perturbations-Methode wesentlich stärker bemerkbar macht als bei der Fixpunkt-Iteration.
Die Tabelle 5.1 zeigt den maximalen Fehler beziehungsweise Unterschied der einzelnen Methoden auf den angegebenen Gebietsabschnitten. Hier erkennt man nochmals recht deutlich, dass die Perturbations-Methode in einer kleinen Umgebung um das Gleichgewicht sehr genaue Ergebnisse liefert. Der Fehler der Fixpunkt-Iteration bleibt jedoch über

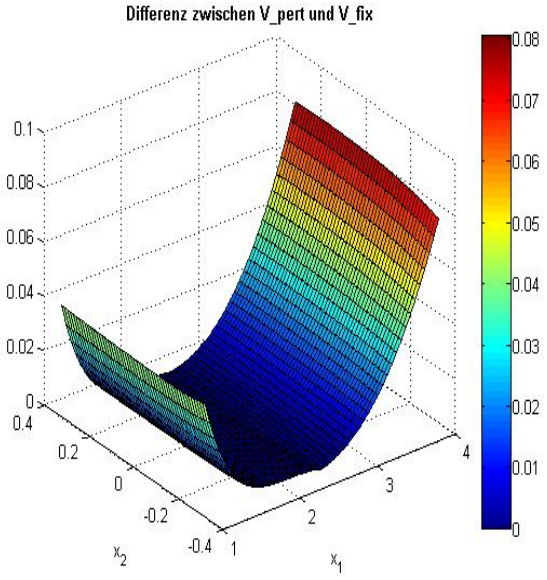

Abbildung 5.5: Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration auf dem Rechteck $\Omega = [1.0, 4.0] \times [-0.32, 0.32]$

---

den gesamten Bereich hinweg konstanter. Lediglich im Gleichgewicht und im kleinsten Abschnitt übersteigt er den der Perturbations-Methode, aber selbst im noch recht kleinen Rechteck $[1.8, 2.5] \times [-0.05, 0.05]$ ist die Fixpunkt-Iteration bereits besser.
Da man für das Rechteck $[1.0, 4.0] \times [-0.32, 0.32]$ stets geringere Fehler als 0.1 erhält, liefern uns die errechneten Werte gute Approximationen. Daher ist es nicht zwingend notwendig, die Schrittweiten durch eine größere Anzahl von Knoten zu verkleinern. Die Genauigkeit insbesondere der Fixpunkt-Iteration würde zwar verbessert, was auf Grund der geringen Fehlergröße aber nicht nötig ist und lediglich die Rechenzeit erhöhen würde.

Das Gebiet $\Omega = [0.3, 8.8] \times [-0.32, 0.32]$

Wir sind allerdings auch an den Lösungen in größerer Entfernung vom Gleichgewichtspunkt interessiert. Dazu bertrachten wir das Rechteck $[0.3, 8.8] \times [-0.32, 0.32]$. Die Anzahl der Gitterpunkte soll dabei nicht verändert werden, weshalb wir Schrittweiten von

Teilgebiet	**Fehler V_pert**	**Fehler V_fix**	**‖V_pert - V_fix‖**
Im Gleichgewicht	$1.629119 \cdot 10^{-5}$	$6.999264 \cdot 10^{-4}$	$6.836352 \cdot 10^{-4}$
$[2.0, 2.1] \times [-0.01, 0.01]$	$1.7 \cdot 10^{-5}$	$6.86 \cdot 10^{-4}$	$6.72 \cdot 10^{-4}$
$[1.8, 2.5] \times [-0.05, 0.05]$	$1.355 \cdot 10^{-3}$	$7.15 \cdot 10^{-4}$	$9.4 \cdot 10^{-4}$
$[1.5, 3.5] \times [-0.1, 0.1]$	$3.4309 \cdot 10^{-2}$	$1.085 \cdot 10^{-3}$	$3.362 \cdot 10^{-2}$
$[1.5, 3.5] \times [-0.3, 0.3]$	$3.4331 \cdot 10^{-2}$	$2.529 \cdot 10^{-3}$	$3.3668 \cdot 10^{-2}$
$[1.0, 4.0] \times [-0.32, 0.32]$	$8.1474 \cdot 10^{-2}$	$7.486 \cdot 10^{-3}$	$8.0453 \cdot 10^{-2}$

Tabelle 5.1: Fehler von V_pert und V_fix auf unterschiedlichen Gebietsabschnitten

$h_1 = 0.186$ und $h_2 = 0.0128$ erhalten.

Im Unterschied zur Referenzliteratur von BECKER, GRÜNE und SEMMLER [3] wurde in dieser Arbeit die untere Grenze auf 0.3 statt 0.1 gesetzt. Dies findet seine Begründung in der Tatsache, dass im erweiterten Modell für die Werte $x_1 = 0.1$ die erste Komponente von φ, also $Ae^{x_2}x_1^{\alpha} - u$, für alle Gitterknoten von x_2 die Intervallgrenze 0.1 unterschreiten würde. Gleiches Phänomen zeigt sich bei einer Untergrenze von 0.2. Somit ist eine Interpolation von V in diesen Punkten nicht möglich. Dies wird bei der Fixpunkt-Iteration aber zur Berechnung des Erwartungswertes in (2.19) beziehungsweise (4.2) benötigt, weshalb man zu einer Extrapolation übergehen müsste. Um das zu vermeiden, wäre als Alternative denkbar, bei Unterschreitung der unteren Intervallgrenze die entsprechenden Werte auf 0.1 zu setzen. Allerdings liefert uns das eine sehr schlechte Approximation von V an der Intervallgrenze, was Fehler im Bereich von 10 bis 15 Einheiten bedeutet.

Auch eine obere Intervallgrenze von 10 kann nicht aufrecht erhalten werden. Die Ursache hierfür liegt in der Berechnung der Kontrolle durch die Perturbations-Methode nach SCHMITT-GROHÉ und URIBE, die wiederum für die Gleichung (4.2) benötigt wird. Das Problem liegt diesmal jedoch im ersten Term $\psi(x, u(x))$, der dem $logu(x)$ in unserem

Modell-Beispiel entspricht. In der Approximation von u im erweiterten Modell ergeben sich an einigen Stellen, zum Punkt $(x_1, x_2) = (10, -0.32)$ hin, negative Werte, was eine Auswertung von $log\widehat{u}(x)$ im Reellen ausschließt. Die Substitution der negativen Werte durch einen kleinen positiven Wert im Bereich von 10^{-6} liefert wiederum Fehler von beinahe 10 Einheiten. Die ursprüngliche Kontrolle wird durch eine derartige Substitution verfälscht und liefert somit falsche Ergebnisse für V. Deshalb wurde die obere Intervallgrenze auf 8.8 herabgesetzt, um die negativen Werte der Kontrolle zu umgehen.
Hierin liegt ein weiterer Nachteil der Approximations-Methode nach SCHMITT-GROHÉ und URIBE, da deren Verwendung in unserem Beispiel bei negativen Werten nicht möglich ist.
Zur Genauigkeit der numerischen Werte lässt sich feststellen, dass auf diesem Bereich bei der Kontrolle u bereits ein größerer Fehler zu erkennen ist. Lediglich im Gleichgewicht scheint die Approximation wieder recht genau. Am Rand des Rechtecks werden dabei Fehler von über 3 gemessen, was nicht mehr als sinnvolle Approximation verwendet werden kann. Wiederum ist es der Abschnitt $[1.8, 2.5] \times [-0.05, 0.05]$, das eine angenehme Fehlergröße von 10^{-3} aufweist. Der geringfügige Unterschied zum vorher ermittelten Fehler auf dem Gebiet $[1.8, 2.5] \times [-0.05, 0.05]$ liegt daran, dass die Gitterpunkte anders gewählt wurden, da ihre Anzahl analog dem kleineren Rechteck bleibt. Somit wurden die Fehler in anderen Punkten ermittelt, was die Differenz begründet. Für das Gleichgewicht erhalten wir dadurch sogar einen Fehler von ca. $2.8 \cdot 10^{-10}$ und auch auf dem Bereichsabschnitt $[1.5, 3.5] \times [-0.3, 0.3]$ ergibt sich nur ein geringer Fehler von etwa 10^{-1}.
Bei den Auswertungen für die Wertefunktion erhalten wir auf dem Gebiet $[0.3, 8.8] \times [-0.32, 0.32]$ die Abbildung 5.6 für die Perturbations-Methode und Abbildung 5.7 für die Fixpunkt-Iteration. Die jeweilige Methode wird dabei immer mit der exakten Lösung abgebildet. Hier kann man in beiden Fällen bereits eine Differenz erkennen.
Die Fixpunkt-Iteration scheint wieder geringfügig genauer als die Perturbations-Methode. Auch der Plot des jeweiligen Fehlers bestätigt diese Aussage. Die Perturbations-Methode liefert Fehler von bis zu ca. 1.7 Einheiten, während bei der Fixpunkt-Iteration der Feh-

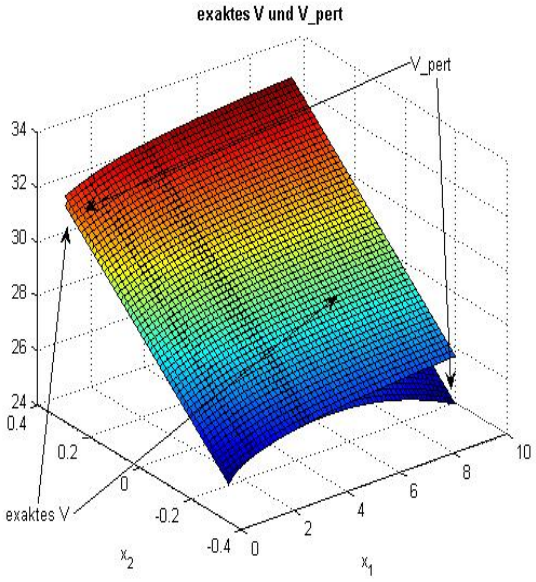

Abbildung 5.6: Exakte Wertefunktion und Perturbations-Methode auf dem Rechteck $\Omega = [0.3, 8.8] \times [-0.32, 0.32]$

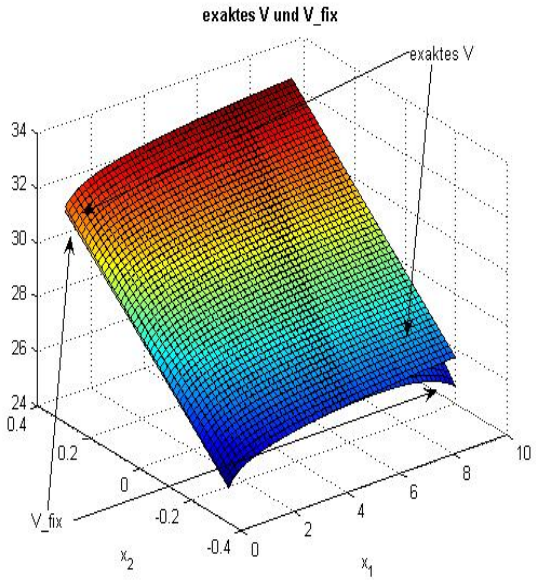

Abbildung 5.7: Exakte Wertefunktion und Fixpunkt-Iteration auf dem Rechteck $\Omega = [0.3, 8.8] \times [-0.32, 0.32]$

ler stets unter 1.0 bleibt. Zudem werden derart hohe Fehler wiederum nur zum Punkt $(8.8, -0.32)$ hin angenommen. Auf dem restlichen Bereich scheint die Approximation noch um einiges genauer zu sein. Die Perturbations-Methode hingegen weist diese Ungenauigkeit für alle Werte von x_2 zur Intervallgrenze $x_1 = 8.8$ hin auf. Siehe hierzu die Abbildungen 5.8 und 5.9.

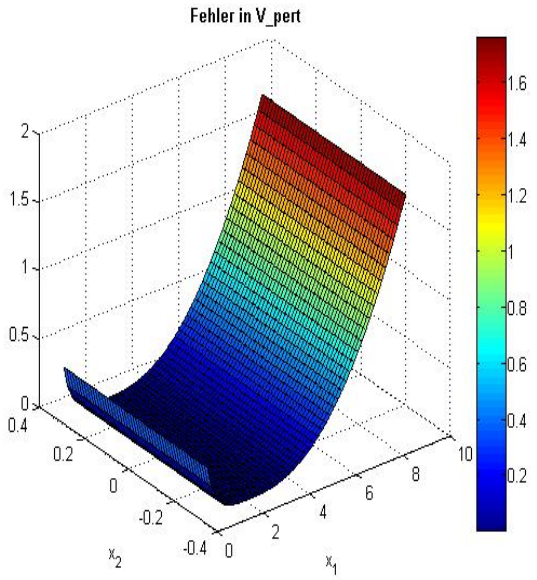

Abbildung 5.8: Fehler der Perturbations-Methode für die Wertefunktion auf dem Rechteck $\Omega = [0.3, 8.8] \times [-0.32, 0.32]$

Von Interesse wäre nun noch, ob die Perturbations-Methode zumindest nahe des Gleichgewichtspunktes genauere Ergebnisse liefert. Hierzu betrachten wir die Tabelle 5.2. Wie bereits auf dem kleineren Gebiet ergeben sich im Gleichgewichtspunkt selbst, aber auch auf dem Abschnitt $[1.8, 2.5] \times [-0.05, 0.05]$ wesentlich geringere Fehler bei der Perturbations-Methode von unter $3.8 \cdot 10^{-4}$. Die Fixpunkt-Iteration hat hier einen vergleichsweise hohen Fehler von etwa $5.4 \cdot 10^{-3}$. Dennoch ist mit einer derartigen Ungenauigkeit die Approximation noch ohne Weiteres verwendbar. Zudem steigt der Fehler in den folgenden Abschnitten kaum an und liegt selbst bei dem Rechteck $[1.0, 4.0] \times [-0.32, 0.32]$ nur unwesentlich über 10^{-2}.

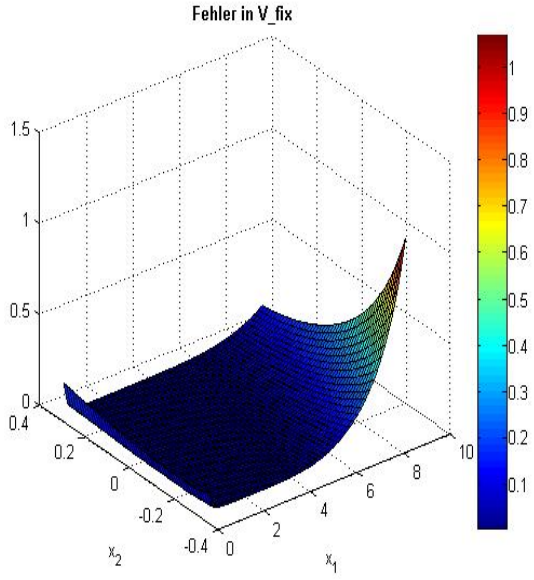

Abbildung 5.9: Fehler der Fixpunkt-Iteration auf $\Omega = [0.3, 8.8] \times [-0.32, 0.32]$

Man sieht recht deutlich, dass die Perturbations-Methode auf kleinen Umgebungen des Gleichgewichts wesentlich genauere Werte liefert, was wiederum in der Funktionsweise des Algorithmus begründet liegt, da die Taylor-Reihe um den Gleichgewichtspunkt entwickelt wird. Der Fehler steigt allerdings zu den Rändern des Gebiets hin deutlicher an als bei der Fixpunkt-Iteration. Diese liefert konstantere Fehlerwerte.
Der Unterschied zwischen den beiden Methoden orientiert sich an dem maximalen Fehler der Perturbations-Methode beziehungsweise der Fixpunkt-Iteration. Deshalb erhalten wir für die Differenz bis zum Itervall $[1.8, 2.5] \times [-0.05, 0.05]$ in etwa den Fehler der Fixpunkt-Iteration. Auf den größeren Bereichsabschnitten entspricht sie ungefähr dem Fehler der Perturbations-Methode, wie uns die Tabelle 5.2 bestätigt. Bei der späteren Betrachtung des erweiterten Modells, können wir aus diesen Erkenntnissen eventuell Aufschluss über die Genauigkeit der Approximationen erhalten.

Da wir am Rand des Rechtecks von steigenden Ungenauigkeiten ausgehen können, wollen wir die Approximationen für eine größere Anzahl von Gitterpunkten durchführen.

Teilgebiet	**Fehler V_pert**	**Fehler V_fix**	**‖V_pert - V_fix‖**
Im Gleichgewicht	$1.629119 \cdot 10^{-5}$	$5.773829 \cdot 10^{-3}$	$5.757538 \cdot 10^{-3}$
$[2.0, 2.1] \times [-0.01, 0.01]$	$9.0 \cdot 10^{-6}$	$5.332 \cdot 10^{-3}$	$5.323 \cdot 10^{-3}$
$[1.8, 2.5] \times [-0.05, 0.05]$	$3.76 \cdot 10^{-4}$	$5.432 \cdot 10^{-3}$	$5.699 \cdot 10^{-3}$
$[1.5, 3.5] \times [-0.1, 0.1]$	$2.8098 \cdot 10^{-2}$	$5.811 \cdot 10^{-3}$	$2.3375 \cdot 10^{-2}$
$[1.5, 3.5] \times [-0.3, 0.3]$	$2.8119 \cdot 10^{-2}$	$8.441 \cdot 10^{-3}$	$2.3918 \cdot 10^{-2}$
$[1.0, 4.0] \times [-0.32, 0.32]$	$6.7935 \cdot 10^{-2}$	$1.2623 \cdot 10^{-2}$	$6.3492 \cdot 10^{-2}$
$[0.5, 6.0] \times [-0.32, 0.32]$	$4.6154 \cdot 10^{-1}$	$9.7826 \cdot 10^{-2}$	$4.47115 \cdot 10^{-1}$
$[0.5, 8.0] \times [-0.32, 0.32]$	1.280376	$5.34061 \cdot 10^{-1}$	1.202499
$[0.3, 8.8] \times [-0.32, 0.32]$	1.75483	1.065386	1.607179

Tabelle 5.2: Fehler von V_pert und V_fix auf unterschiedlichen Gebietsabschnitten

Wir wählen dafür in x_1- und x_2-Richtung jeweils 251 Gitterpunkte, also eine Gesamtzahl von 63001 Gitterpunkten, und für die Zufallsvariable 51 Gitterpunkte. Somit ergeben sich die Schrittweiten $h_1 = 0.0372$, $h_2 = 0.00256$ und $h_z = 0.00128$. Leider erhalten wir dadurch kaum Änderungen in den Approximationen durch die Perturbations-Methoden, was auf Grund der Berechnung einer Taylor-Reihe zweiter Ordnung anschaulich klar ist. Aber auch die Fixpunkt-Iteration scheint keine wesentliche Verbesserungen gegenüber der vorherigen Anzahl von Knoten zu zeigen. Hierzu wollen wir uns aber Tabelle 5.3 betrachten, die wiederum die Fehler und den Unterschied der beiden Approximationen auf unterschiedlichen Abschnitten auflistet.

Man sieht, dass die Fixpunkt-Iteration mit Ausnahme der letzten drei Abschnitte nur Fehler zwischen 10^{-4} und 10^{-2} aufweist. Vor allem auf den ersten vier Bereichen zeigt sich eine größere Genauigkeit, da wir nur Fehler in einem Bereich von 10^{-4} erhalten. So-

Teilgebiet	**Fehler V_pert**	**Fehler V_fix**	**‖V_pert - V_fix‖**
Im Gleichgewicht	$1.629119 \cdot 10^{-5}$	$2.034242 \cdot 10^{-4}$	$1.87133 \cdot 10^{-4}$
$[2.0, 2.1] \times [-0.01, 0.01]$	$1.7 \cdot 10^{-5}$	$2.19 \cdot 10^{-4}$	$2.07 \cdot 10^{-4}$
$[1.8, 2.5] \times [-0.05, 0.05]$	$1.156 \cdot 10^{-3}$	$2.29 \cdot 10^{-4}$	$9.58 \cdot 10^{-4}$
$[1.5, 3.5] \times [-0.1, 0.1]$	$3.6731 \cdot 10^{-2}$	$6.99 \cdot 10^{-4}$	$3.6427 \cdot 10^{-2}$
$[1.5, 3.5] \times [-0.3, 0.3]$	$3.6751 \cdot 10^{-2}$	$2.156 \cdot 10^{-3}$	$3.6451 \cdot 10^{-2}$
$[1.0, 4.0] \times [-0.32, 0.32]$	$7.8437 \cdot 10^{-2}$	$6.524 \cdot 10^{-3}$	$7.7813 \cdot 10^{-2}$
$[0.5, 6.0] \times [-0.32, 0.32]$	$4.81738 \cdot 10^{-1}$	$9.7606 \cdot 10^{-2}$	$4.70039 \cdot 10^{-1}$
$[0.5, 8.0] \times [-0.32, 0.32]$	1.297828	$5.42439 \cdot 10^{-1}$	1.221451
$[0.3, 8.8] \times [-0.32, 0.32]$	1.75483	1.059105	1.610786

Tabelle 5.3: Fehler von V_pert und V_fix auf unterschiedlichen Gebietsabschnitten

mit überwiegt bei der Differenz der beiden Approximationen meist der Fehler durch die Perturbations-Methode. Dies ist auf besagten letzten drei Gebietsabschnitten allerdings weniger ausgeprägt, da hier der Fehler der Fixpunkt-Iteration ebenfalls stark zunimmt. Abermals wird dieser Fehler nur zum Punkt $(8.8, -0.32)$ hin angenommen. Bei den anderen Grenzen des Bereichs ist die Approximation selbst auf dem größten Gebiet noch akzeptabel, da der Fehler sich auf ca. 0.1 beläuft, wie Abbildung 5.10 zeigt.
Sucht man nach einer Ursache für die vergleichsweise hohen Abweichungen der Fixpunkt-Iteration zum Punkt $(8.8, -0.32)$ hin, sollte man den Fehler der Approximation der Kontrolle u, siehe Abbildung 5.11, betrachten. Es ist hier unschwer zu erkennen, dass die Perturbations-Methode für die Kontrolle ihre größte Ungenauigkeit zum Intervallrand $x_1 = 8.8$ aufweist, und hier insbesondere im Punkt $(8.8, -0.32)$. Da für die Fixpunkt-Iteration diese Approximation von u verwendet wird, lässt sich die Ungenau-

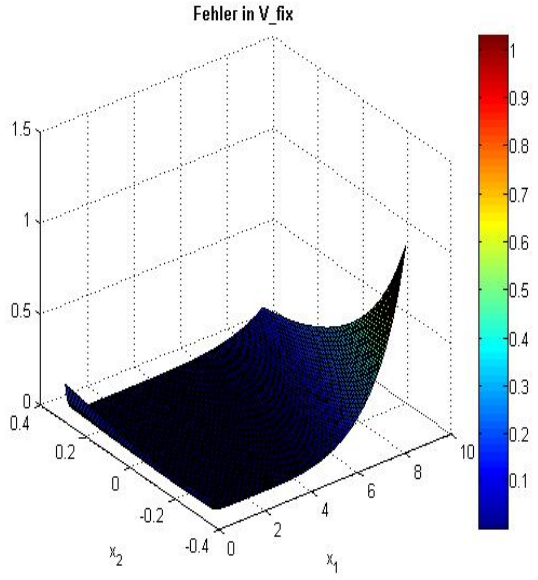

Abbildung 5.10: Fehler der Fixpunkt-Iteration $\Omega = [0.3, 8.8] \times [-0.32, 0.32]$

igkeit wohl auf die Fehler bei der approximativen Kontrolle zurückführen. Vermutlich ist dies unter anderem auch ein Grund der Ungenauigkeit der Perturbations-Methode für die Wertefunktion, die ebenfalls besagte Approximation verwendet.

Im Allgemeinen kann man sagen, dass die Perturbations-Methode für kleine Umgebungen um das Gleichgewicht eine hervorragende Approximation liefert. Diese Umgebungen können unter Umständen recht gering sein, und der Fehler nimmt stark zu, je weiter man sich vom Gleichgewicht entfernt. Hier besticht die Fixpunkt-Iteration, da sie trotz des vergleichsweise höheren Fehlers um den Gleichgewichtspunkt einen konstanteren und dennoch geringen Fehler auch in weiterer Entfernung von diesem Punkt liefert. Durch eine größere Anzahl von Gitterpunkten wird die Genauigkeit zusätzlich auf dem gesamten Gebiet erhöht. Auf Grund dieser Tatsachen ist die Fixpunkt-Iteration wohl als Approximation zu bevorzugen.

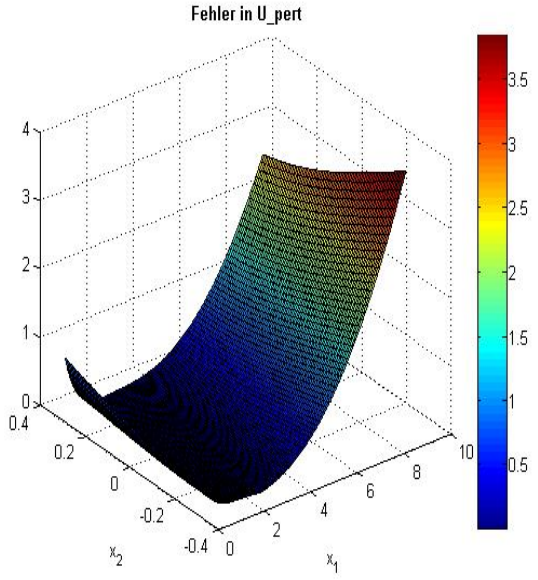

Abbildung 5.11: Fehler der Perturbations-Methode für die Kontrolle auf dem Rechteck $\Omega = [0.3, 8.8] \times [-0.32, 0.32]$

5.3 Abwandlung des Modells

Die Motivation für die Abänderung des vorherigen Beispiels liegt in der geringen Beachtung der stochastischen Anteile $g_{\sigma\sigma}$, $h_{\sigma\sigma}$ und $V_{\sigma\sigma}$. Hierzu wird die Variable κ wie folgt in die Zielfunktion eingebaut:

$$\max \sum_{t=0}^{\infty} \beta^t \ \log u_t \cdot e^{\kappa \cdot x_2} \tag{5.10}$$

Dadurch wird die Unsicherheit stärker in das Modell einbezogen. Zudem kann im Folgenden untersucht werden, wie sich die Änderung auf die Koeffizienten auswirkt. Unser vorheriges Beispiel ist also als Spezialfall des erweiterten Modells für $\kappa = 0$, also Vernachlässigung des Risikos, zu interpretieren.
Das Kontrollsystem ist wiederum durch

$$\varphi(x, u, z) \ = \ \begin{pmatrix} A\, e^{x_2}\, x_1^{\alpha} - u \\ \rho x_2 + z \end{pmatrix} \tag{5.11}$$

gegeben und auch die Werte der Variablen, $A = 5$, $\alpha = 0.34$, $\beta = 0.95$ und $\rho = 0.9$, werden für die späteren Auswertungen übernommen.

5.3.1 Numerische Approximationen

Analog zum Grundmodell muss nun die Funktion $\widetilde{f}$ aus der Gleichgewichtsbedingung (2.6) hergeleitet werden, was genauer in BECKER [2] nachzulesen ist:

$$\widetilde{f} := \begin{pmatrix} x_1^{'} - A\, e^{x_2}\, x_1^{\alpha} + u \\ x_2^{'} - \rho\, x_2 \\ \beta \cdot \frac{1}{u^{'}} \cdot e^{\kappa x_2^{'}} \cdot \alpha \cdot A \cdot e^{x_2^{'}} \cdot x_1^{'(\alpha-1)} - \frac{1}{u} \cdot e^{\kappa x_2^{'}} \end{pmatrix} = 0. \tag{5.12}$$

Daraus lässt sich wieder das Gleichgewicht, der Ausgangspunkt unserer Perturbations-Algorithmen, errechnen.

Obwohl sich die dritte Zeile der Gleichung (5.12) verändert hat, bleiben die Werte für den Gleichgewichtspunkt erhalten. Das Gleichgewicht ist also unabhängig von der Variablen κ. Die einzelnen Schritte zur Ermittlung des Gleichgewichts gestalten sich wie folgt.

Aus der zweiten Zeile erhalten wir wie bereits vorher, dass $\bar{x}_2 = 0$ gilt. Wiederum eingesetzt in die erste Zeile von (5.12), ergibt sich für u analog den vorherigen Berechnungen:

$$u = A \cdot 1 \cdot x_1^{\alpha} - x_1. \tag{5.13}$$

Verwenden wir diese Ergebnisse in Zeile 3 der Gleichung (5.12), so erhalten wir:

$$\begin{aligned} & \beta\, \alpha\, A\, x_1^{(\alpha-1)}\, \frac{1}{A\, x_1^{\alpha} - x_1} = \frac{1}{A\, x_1^{\alpha} - x_1} \\ \Longleftrightarrow \qquad & \beta\, \alpha\, A\, x_1^{(\alpha-1)} = 1. \end{aligned}$$

Da somit die verbleibenden Berechnungen analog des Grundmodells verlaufen, erhalten wir die selben Werte für das Gleichgewicht:

$$\left[\ \bar{x};\ \ \bar{u}\ \right] \ = \ \left[\ (2.067344815, 0);\ \ 4.333103529\ \right].$$

Für die Taylor-Entwicklung der Kontrolle und der Dynamik liefern uns für $\kappa = 2$ die entsprechenden Matlab-Routinen `Beispiel2.m`, `Beispiel2_Zahlen.m` und `Beispiel2_Run.m` folgende Werte:

- Koeffizienten der linearen Terme sind:

$$g_x = \begin{pmatrix} 0.7126 & 4.7277 \end{pmatrix}$$

$$h_x = \begin{pmatrix} 0.3400 & 1.6727 \\ 0 & 0.9000 \end{pmatrix}$$

- Koeffizienten der in x quadratischen Terme:

$$g_{xx}(:,:,1) = \begin{pmatrix} -0.2275 & 0.7775 \end{pmatrix}$$

$$g_{xx}(:,:,2) = \begin{pmatrix} 0.7775 & 5.0563 \end{pmatrix}$$

$$h_{xx}(:,:,1) = \begin{pmatrix} -0.1085 & 0.2751 \\ 0 & 0 \end{pmatrix}$$

$$h_{xx}(:,:,2) = \begin{pmatrix} 0.2751 & 1.3441 \\ 0 & 0 \end{pmatrix}$$

- Koeffizienten der in σ quadratischen Terme:

$$g_{\sigma\sigma} = \begin{pmatrix} -7.5821 \end{pmatrix}$$

$$h_{\sigma\sigma} = \begin{pmatrix} 7.5821 \\ 0 \end{pmatrix}.$$

Im Vergleich zum Basis-Modell sind insbesondere die Werte für $g_{\sigma\sigma}$ und $h_{\sigma\sigma}$ gestiegen. Diese Tatsache reflektiert die gestiegene Bedeutung der wirtschaftlichen Unsicherheit und weist auf die starke Abhängigkeit der optimalen Kontrolle beziehungsweise Trajektorien von σ hin.

Durch die Einbindung der Variablen κ erwarten wir für die Wertefunktion eine stärkere

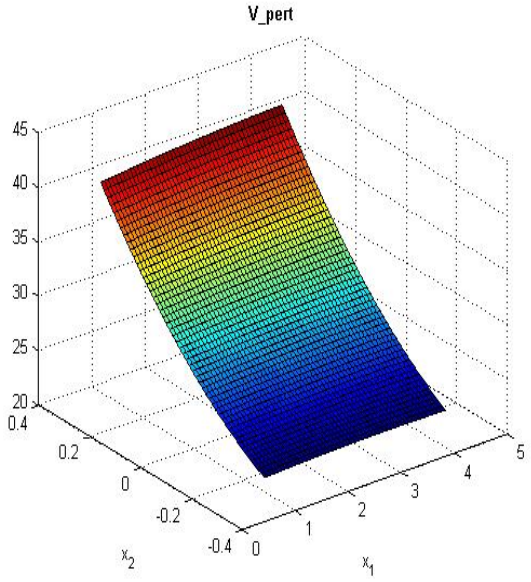

Abbildung 5.12: Perturbations-Methode für die Wertefunktion auf dem Rechteck $\Omega = [0.8, 4.2] \times [-0.32, 0.32]$

Krümmung im betrachteten Gebiet. Dies bestätigt auch die Abbildung 5.12, in der die mit Hilfe der Taylor-Entwicklung und $\kappa = 2$ ermittelte Wertefunktion zu sehen ist.
Löst man für $\kappa = 2$ das sich ergebende lineare Gleichungssystem (5.9), das sich aus beiderseitiger Taylor-Entwicklung der Gleichung (5.8) ergibt, so erhalten wir folgende Werte für die Koeffizienten der Taylor-Reihe von V:

$$\begin{aligned} V_0 &= 29.32566 \\ V_{x_1} &= 0.24291 \\ V_{x_2} &= 28.38888 \\ V_{x_1x_1} &= -0.11749 \\ V_{x_1x_2} &= V_{x_2x_1} = 0.41735 \\ V_{x_2x_2} &= 42.71896 \\ V_{\sigma\sigma} &= -0.15508 \cdot 10^{-2}. \end{aligned}$$

Somit ist nun die Approximation von $V(x)$ gegeben durch

$$\begin{aligned}\widehat{V}(x) &= 29.32566 + 0.24291 \cdot (x_1 - \bar{x}_1) + 28.38888 \cdot (x_2 - \bar{x}_2) \\ &\quad + \frac{1}{2}(-0.11749)(x_1 - \bar{x}_1)^2 + 0.41735 \cdot 10^{-4} \cdot (x_1 - \bar{x}_1)(x_2 - \bar{x}_2) \\ &\quad + 42.71896 \cdot (x_2 - \bar{x}_2)^2 - 0.15508 \cdot 10^{-2}\sigma^2\end{aligned}$$

Während sich die Koeffizienten stark vergrößert haben, in denen Ableitungen nach x_2 vorkommen, spielt $V_{\sigma\sigma}$ immer noch eine relativ geringe Rolle. Durch die Multiplikation mit σ^2 verschwindet der Term fast komplett. Aus der Modellstruktur ist dies aber durchaus zu erwarten, da $V_{x_2x_2}$ und $V_{\sigma\sigma}$ voneinander abhängen. Ein größeres σ verursacht dabei in erster Linie eine größere Schwankung der x_2-Komponente. Diese kann sich nur dann in der Wertefunktion auswirken, wenn V in x_2-Richtung eine stärkere Krümmung aufweist, also $|V_{x_2x_2}| >> 0$ ist.
Da nun für die Approximationen alle Berechnungen durchgeführt wurden, wollen wir uns wieder der Genauigkeit der Lösungen widmen.

5.3.2 Auswertung für $\kappa = 2$

Um die erhaltenen Schlüsse aus dem vorherigen Abschnitt auf das erweiterte Modell übertragen zu können, betrachten wir in etwa die gleichen Gebiete wie zuvor. Damit die errechneten Werte hierzu stets in den angegebenen Bereichen liegen und man keine negativen Werte bei der Approximation von u erhält, müssen für $\kappa = 2$ das kleine Gebiet auf $[0.8, 4.2] \times [-0.32, 0.32]$ vergrößert, und das große auf $[0.3, 8.6] \times [-0.32, 0.32]$ verkleinert werden. Auf Grund der geringen Veränderungen bedeutet dies keine wesentliche Einschränkung bezüglich der Übertragbarkeit der vorherigen Resultate.
Da nun keine exakte Lösung berechnet werden kann, bleibt uns lediglich der Vergleich zwischen den beiden Approximationen der Wertefunktion. Hierfür wählen wir zunächst den kleineren Bereich.

Das Gebiet $\Omega = [0.8, 4.2] \times [-0.32, 0.32]$

Für die Kontrolle u können mangels Vergleichsmöglichkeiten keine Aussagen gemacht werden. Es liegt jedoch die Vermutung nahe, dass auch hier nahe des Gleichgewichts-

punktes eine hohe Genauigkeit erreicht wird, während der Fehler mit größerer Entfernung steigt. Dies findet seine Begründung wiederum in der Funktionsweise des Verfahrens, da der Gleichgewichtspunkt als Entwicklungspunkt verwendet wird. Für genauere Resultate sei auf die Arbeit von BECKER [2] verwiesen, die die Perturbations-Methode mit der Dynamischen Programmierung vergleicht, wobei letztere immer eine gewisse Genauigkeit gewährleistet. Der dort angeführte Vergleich bestätigt unsere Vermutung.
Bei der Wertefunktion scheinen beide Approximationen auf den ersten Blick recht ähnlich. Man erkennt lediglich einen geringen Unterschied zu den Bereichsgrenzen hin. Tatsächlich bewegt sich diese Differenz im ganzzahligen Bereich, wie der Plot des Fehlers in Abbildung 5.13 zeigt. Dabei scheint der Unterschied für x_2 nahe bei Null auf dem gesamten Intervall für x_1 relativ gering zu sein. Betrachtet man die Tabelle 5.4 so wird bestätigt, dass in kleinen Umgebungen um das Gleichgewicht die Differenz sehr gering ist. Sie liegt bei ca. 10^{-2}. Aus den Erfahrungen des letzten Abschnitts legt dies nahe, dass hier beide Approximationen relativ genau sind.

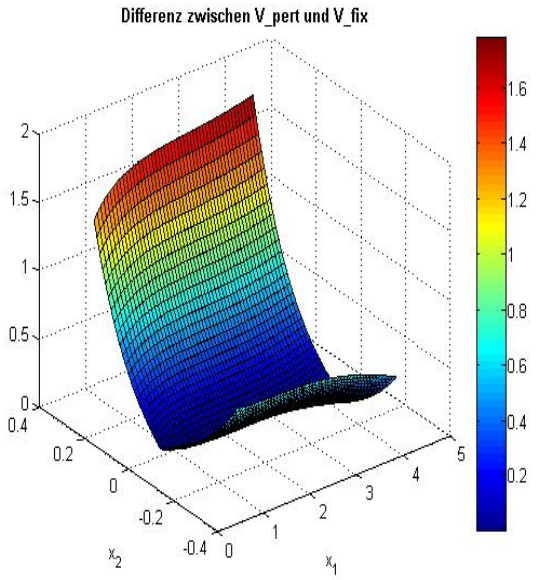

Abbildung 5.13: Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration auf dem Rechteck $\Omega = [0.8, 4.2] \times [-0.32, 0.32]$

Für die Fixpunkt-Iteration lässt sich weiterhin die Vermutung anstellen, dass der Fehler

Teilgebiet	**Differenz zwischen V_pert und V_fix**
Im Gleichgewicht	$4.17527 \cdot 10^{-2}$
$[2.0, 2.1] \times [-0.01, 0.01]$	$4.1824 \cdot 10^{-2}$
$[1.8, 2.5] \times [-0.05, 0.05]$	$1.29151 \cdot 10^{-1}$
$[1.5, 3.5] \times [-0.1, 0.1]$	$2.96746 \cdot 10^{-1}$
$[1.5, 3.5] \times [-0.3, 0.3]$	1.471612
$[0.8, 4.2] \times [-0.32, 0.32]$	1.76091

Tabelle 5.4: Differenz zwischen V_pert und V_fix auf unterschiedlichen Gebietsabschnitten, $\kappa = 2$

relativ konstant bleibt, jedoch nahe des Gleichgewichtspunktes überwiegt, da er merklich größer als der Fehler der Perturbations-Methode ist. Im Gleichgewicht und in kleiner Umgebung stellt sich für $\kappa = 2$ eine Differenz von ca. $4.2 \cdot 10^{-2}$ ein. Man kann also davon ausgehen, dass dies in etwa der Fehler der Fixpunkt-Iteration im Gleichgewicht ist. Analog zu den vorherigen Entwicklungen der Fixpunkt-Iteration kann man annehmen, dass auf dem gesamten Rechteck $[0.8, 4.2] \times [-0.32, 0.32]$ der Fehler Werte von ungefähr $5 \cdot 10^{-1}$ erreichen wird. Die Approximation weist demnach bereits zu große Ungenauigkeiten auf.
Zugleich würde ein derartiger Fehler aber implizieren, dass die Differenz auf dem Bereich $[0.8, 4.2] \times [-0.32, 0.32]$ maßgeblich von der Perturbations-Methode verursacht wird. Man erhält deshalb Fehler von über 1.0, was entschieden höher als für den Fall $\kappa = 0$ und auf Grund der Größe inakzeptabel ist.

Wiederum ist es also die Perturbations-Methode, die nahe des Gleichgewichts sehr geringe Fehler, zu den Rändern des Bereichs hin aber einen inakzeptablen Anstieg auf-

weist. Die Fixpunkt-Iteration hingegen liefert auf dem gesamten Gebiet niedrigere Fehler, approximiert die tatsächliche Wertefunktion aber in geringer Entfernung vom Gleichgewichtspunkt nicht derart akkurat wie die Perturbations-Methode. Eine Entscheidung zwischen beiden Methoden muss daher immer vor dem Hintergrund der gestellten Anforderungen gefällt werden, je nachdem ob die Approximation nahe des Gleichgewichts oder eine konstante Fehlergröße auf dem gesamten Bereich im Vordergrund steht.
Diese Aussagen stützen sich lediglich auf die Vermutung, dass das Verhalten der beiden Approximationen für $\kappa = 2$ in etwa dem für $\kappa = 0$ entspricht. Leider entbehrt diese Annahme derzeit jeglicher Grundlage. Dies müsste zunächst mit einem besseren Verfahren wie beispielsweise der Dynamischen Programmierung getestet werden, da sie einen entsprechend konstanten Fehler aufweist und somit für einen Vergleich herangezogen werden kann, was im begrenzten Rahmen dieser Arbeit jedoch nicht ausgeführt wird.

Das Gebiet $\Omega = [0.3, 8.6] \times [-0.32, 0.32]$

Bei diesem Gebiet werden wir wie oben von den Resultaten des vorherigen Abschnitts ausgehen und versuchen, Parallelen zu dem erweiterten Modell zu ziehen. Für die Kontrolle fehlt uns allerdings wieder jeglicher Vergleich, weshalb nochmals auf die Arbeit von BECKER [2] verwiesen sei.
Abermals zeigt sich, dass die Differenz der Approximationen nahe des Gleichgewichtspunktes recht klein ist und stets bei ca. 10^{-2} liegt. Zu den Bereichsgrenzen hin steigt sie allerdings auf bis zu 3 Einheiten an, siehe Abbildung 5.14.
Die Tabelle 5.5 zeigt uns die Differenz zwischen beiden Methoden auf verschiedenen Gebietsabschnitten. Im oftmals betrachteten Rechteck $[1.8, 2.5] \times [-0.05, 0.05]$ unterscheiden sich beide Verfahren um weniger als $1.3 \cdot 10^{-1}$. Die hier gemessene Differenz wird vermutlich wieder hauptsächlich von der Fixpunkt-Iteration verursacht, da die Perturbations-Methode im Gleichgewicht und kleiner Umgebung bisher immer eine größere Genauigkeit gezeigt hat. Man kann also gemäß der vorherigen Entwicklungen der Fixpunkt-Iteration davon ausgehen, dass der Fehler bereits beim Abschnitt $[1.0, 4.0] \times [-0.32, 0.32]$ ganzzahlige Werte annehmen wird, weshalb die Approximation nicht mehr verwendbar ist.

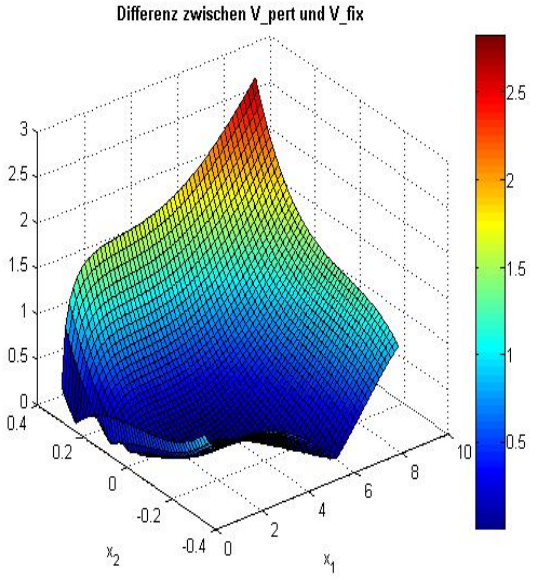

Abbildung 5.14: Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration auf dem Rechteck $\Omega = [0.3, 8.6] \times [-0.32, 0.32]$

Da der gesamte Fehler auf dem Abschnitt $[0.3, 8.8] \times [-0.32, 0.32]$ ca. 3 Einheiten beträgt, scheint auch die Perturbations-Methode keinen wesentlich größeren Fehler aufzuweisen. Allerdings hat hieran auch die Fixpunkt-Iteration einen gewissen Anteil. Man kann also keine genauen Aussagen über den Fehler der Perturbations-Methode treffen, aber vermutlich wird auch dieser im ganzzahligen Bereich verlaufen.

Deshalb wollen wir die Approximationen wieder mit 251 Gitterpunkten je x_1- beziehungsweise x_2-Richtung und 51 Gitterpunkten für die Zufallsvariable berechnen, da wir hier von einer größeren Genauigkeit der Fixpunkt-Iteration ausgehen können.
Am aufschlussreichsten wird die Tabelle 5.6 der Differenz in den angegebenen Bereichsbschnitten sein. Um diese Tabelle zu der erhofften Übertragung der Schlussfolgerungen des vorherigen Abschnitts nutzen zu können, müssen wir folgende Überlegungen anstellen. Zwar vermindert sich die Differenz zwischen beiden Methoden im Gleichgewicht, was man wohl der höheren Genauigkeit der Fixpunkt-Iteration zuschreiben kann.

Teilgebiet	**Differenz zwischen V_pert und V_fix**
Im Gleichgewicht	$3.711163 \cdot 10^{-2}$
$[2.0, 2.1] \times [-0.01, 0.01]$	0.0
$[1.8, 2.5] \times [-0.05, 0.05]$	$1.25052 \cdot 10^{-1}$
$[1.5, 3.5] \times [-0.1, 0.1]$	$2.92966 \cdot 10^{-1}$
$[1.5, 3.5] \times [-0.3, 0.3]$	1.46787
$[1.0, 4.0] \times [-0.32, 0.32]$	1.753805
$[0.5, 6.0] \times [-0.32, 0.32]$	2.038348
$[0.5, 8.0] \times [-0.32, 0.32]$	2.592092
$[0.3, 8.6] \times [-0.32, 0.32]$	2.818611

Tabelle 5.5: Differenz zwischen V_pert und V_fix auf unterschiedlichen Gebietsabschnitten, $\kappa = 2$

Jedoch ist sie in den folgenden Abschnitten sogar höher als bei der vorherigen Approximation mit 51 beziehungsweise 11 Gitterpunkten. Dies steht aber nicht im Gegensatz zur Gültigkeit der Schlüsse des letzten Abschnitts, da die entsprechenden Tabellen 5.2 und 5.3 für $\kappa = 0$ die gleichen Analogien aufweisen. Offensichtlich hat sich der Fehler der Fixpunkt-Iteration hier im Vergleich zur Approximation mit 51 Gitterpunkten weiter verringert, weshalb sich der Fehler der Perturbations-Methode verstärkt bemerkbar macht.

Leider zeigt sich keine derart starke Verminderung der Differenz wie für $\kappa = 0$. Vermutlich hat sich also auch die Fixpunkt-Iteration nicht maßgeblich verbessert. Dennoch wird sich der Fehler auf dem Rechteck $[1.0, 4.0] \times [-0.32, 0.32]$ auf ca. $5 \cdot 10^{-1}$ belaufen, weshalb die Perturbations-Methode hier ebenfalls Ungenauigkeiten im ganzzahligen Be-

Teilgebiet	**Differenz zwischen V_pert und V_fix**
Im Gleichgewicht	$3.070868 \cdot 10^{-2}$
$[2.0, 2.1] \times [-0.01, 0.01]$	$4.6639 \cdot 10^{-2}$
$[1.8, 2.5] \times [-0.05, 0.05]$	$1.42228 \cdot 10^{-1}$
$[1.5, 3.5] \times [-0.1, 0.1]$	$3.18893 \cdot 10^{-1}$
$[1.5, 3.5] \times [-0.3, 0.3]$	1.507376
$[1.0, 4.0] \times [-0.32, 0.32]$	1.745735
$[0.5, 6.0] \times [-0.32, 0.32]$	2.034703
$[0.5, 8.0] \times [-0.32, 0.32]$	2.592182
$[0.3, 8.8] \times [-0.32, 0.32]$	2.807602

Tabelle 5.6: Differenz zwischen V_pert und V_fix auf unterschiedlichen Gebietsabschnitten, $\kappa = 2$

reich aufweist.

Da sich die Approximation von u im Vergleich zum Modell für $\kappa = 0$ nur unwesentlich verändert hat, liegt die Vermutung nahe, dass wir zum Intervallrand $x_1 = 8.6$ beziehungsweise zum Punkt $(8.6, -0.32)$ ebenfalls wieder hohe Fehler erhalten, weshalb hier auch die Fixpunkt-Iteration höhere Fehler liefern wird. Ebenso wird sich dies bei der Perturbations-Methode für die Wertefunktion bemerkbar machen. Dies bedarf aber noch weiterer Untersuchungen beispielsweise mit genaueren Approximationen für u wie durch Dynamische Programmierung, da die Annahmen bisher einer fundierten Grundlage entbehren. Da derartige Untersuchungen jedoch den Rahmen der Arbeit sprengen würden, sei hier darauf verzichtet.

Mit der Annahme, dass das Verhalten der Approximationen für $\kappa = 2$ in etwa dem für $\kappa = 0$ entspricht, würden sich die erhaltenen Ergebnisse mit unseren Vermutungen decken, dass die Fixpunkt-Iteration einen eher konstanten Fehler aufweist. Die Perturbations-Methode hingegen ist zwar um den Gleichgewichtspunkt erheblich besser, wird aber zunehmend ungenauer je größer die Entfernung davon ist und ist somit nur für sehr kleine Umgebungen anwendbar.
Auch die Fixpunkt-Iteration wird, wie wir eben gesehen haben, auf dem großen Gebiet für das erweiterte Modell sehr ungenau. Leider sind zudem die Gebietsgrößen auf Grund der Approximation der Kontrolle durch die Perturbations-Methode beschränkt, da diese einerseits negative Werte annimmt und somit der Iterationsschritt nicht berechnet werden kann, und da sie andererseits an der einen Rechtecksgrenze bereits sehr ungenau wird, was sich auf die Fixpunkt-Iteration überträgt. Will man die Bereiche für die Fixpunkt-Iteration nicht einschränken, so sollte man deshalb als Grundlage in unserem Modell eine Approximation der Kontrolle wählen, die derartige Nachteile nicht mehr aufweist.

5.3.3 Auswertung für unterschiedliche κ und Standardabweichungen

Von großem Interesse bleiben noch die Veränderungen der Wertefunktion und ihrer Approximationen bei unterschiedlichen κ und σ.
Die aus den verschiedenen κ resultierenden Änderungen in den Koeffizienten der Taylor-Approximationen seien hier nicht explizit aufgeführt. Genaue Werte finden sich in den Implementierungen wie `compute_Vcoeff.mw` oder bei Ausführung der entsprechenden Programme. Es wird lediglich darauf hingewiesen, dass mit steigendem κ sich insbesondere die Werte von $g_{\sigma\sigma}$, $h_{\sigma\sigma}$ und $V_{x_2x_2}$ ändern, was auf Grund des verstärkten Einbeziehens der stochastischen Größe anschaulich klar ist. Auch $V_{\sigma\sigma}$ steigt mit zunehmendem κ an, jedoch bleibt der Wert immer deutlich unter dem von $V_{x_2x_2}$, weshalb er kaum ins Gewicht fällt. Als Beispiel seien die Werte für $\kappa = 15$ gegeben, die sich auf 1329.507185 für $V_{x_2x_2}$ und -0.0872324 für $V_{\sigma\sigma}$ belaufen. Wie bereits erwähnt, ist dies zu erwarten, da ein steigendes σ die Schwankung in der x_2-Komponente verstärkt. Dies kann sich in der Wertefunktion nur durch eine stärkerer Krümmung in x_2-Richtung auswirken, wofür

$|V_{x_2x_2}| >> 0$ gelten muss.
Für die folgenden Auswertungen wurden für κ die Werte 1, 2, 5, 10 und 15 gewählt. Der vorherige Wert von $\sigma = 0.008$ wird auf 0.01 beziehungsweise 0.012 erhöht.

Veränderung von κ

Für einen ersten Einblick soll hier die Abbildung 5.15 dienen, in der die Approximation der Wertefunktion durch die Perturbations-Methode für die unterschiedlichen κ zu sehen ist. Da die Werte für $V_{x_2x_2}$ stark ansteigen, verlaufen die Approximationen zunehmend gekrümmter und in gegeben hohen Bereichen. Die Fixpunkt-Iteration übertrifft die Taylor-Entwicklung in dieser Hinsicht, da sie sogar Werte von bis zu beinahe 700 erreicht, siehe Abbildung 5.16. Dies lässt bereits vermuten, dass wir bezüglich Fehlergröße und Genauigkeit keine zutreffenden Aussagen mehr machen können.

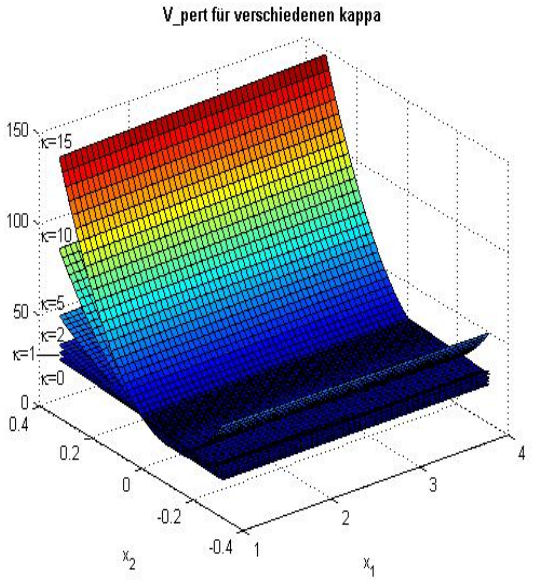

Abbildung 5.15: Perturbations-Methode auf $\Omega = [1.0, 4.0] \times [-0.32, 0.32]$

Beide Approximationen zeigen zwar noch das gleiche tendenzielle Verhalten, dass die Wertefunktion eine stärkere Krümmung in x_2-Richtung aufweist und deshalb die Werte im Bereich $x_2 = 0.32$ extrem ansteigen, jedoch tun sie dies bereits in recht unterschiedlichem Maße. Während bei dem Gebiet $[1.0, 4.0] \times [-0.32, 0.32]$ (beziehungsweise bei

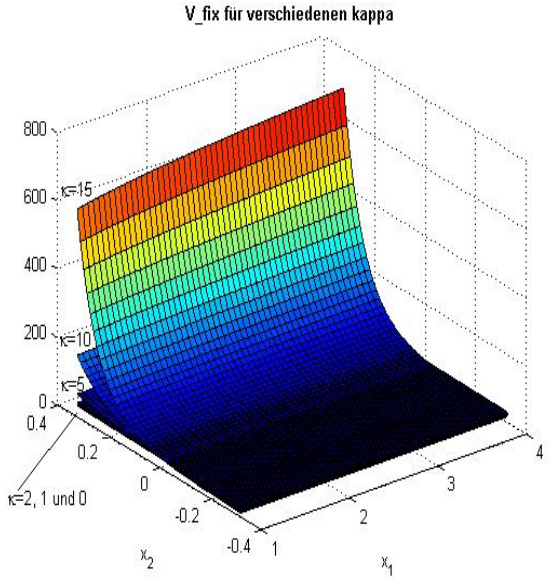

Abbildung 5.16: Fixpunkt-Iteration auf dem Rechteck $\Omega = [1.0, 4.0] \times [-0.32, 0.32]$

geringer Modell-bedingter Einschränkung für $\kappa = 1$ auf $[0.7, 4.5] \times [-0.32, 0.32]$ und für $\kappa = 2$ auf $[0.8, 4.2] \times [-0.32, 0.32]$) die Perturbations-Methode Werte von bis zu 150 erreicht, steigt die Fixpunkt-Iteration hier bereits auf ca. 700. Im vorherigen Abschnitt haben wir gesehen, dass die Fixpunkt-Iteration auf dem gesamten Rechteck meist genauere Werte liefert. Vermutlich bleibt diese Eigenschaft erhalten, weshalb die Perturbations-Methode erhebliche Defizite für große κ, und somit starker Gewichtung des Risikos, aufweist. Über den Fehler der Fixpunkt-Iteration lassen sich an dieser Stelle noch keine Aussagen treffen, es ist jedoch anzunehmen, dass auch dieser sehr groß wird.

Für $\kappa = 1$ und $\kappa = 2$ erhalten wir noch mäßige Differenzen von etwa 1 bis 2, was sich bei $\kappa = 15$ auf Unterschiede von beinahe 600 steigert. Da hierbei vermutlich auch der Fehler der Fixpunkt-Iteration keine unerhebliche Rolle mehr spielt, lässt sich lediglich schließen, dass die Approximationen der stochastischen Unsicherheit nicht gerecht werden können.

Fraglich ist hier, ob die Resultate des vorherigen Abschnitts übertragen werden können. Für $\kappa = 1$ und $\kappa = 2$ ist dies sicherlich in dem Sinne noch möglich, dass die

Perturbations-Methode nahe des Gleichgewichts recht genau zu sein scheint. Vermutlich liefert die Fixpunkt-Iteration wieder die konstanteren Fehlerwerte auf dem gegebenen Gebiet, weshalb hier die Fehlergrenze im Bereich von 0.1 bis 1 liegen wird. Anschaulich ist natürlich klar, dass die Fehler für kleinere κ ebenfalls geringer sind, da das wirtschaftliche Risiko schwächer eingebunden wird.
Für größere κ, insbesondere $\kappa = 10$ und $\kappa = 15$, lassen sich obige Vermutungen kaum mehr aufrecht erhalten. Hier belaufen sich die Differenzen auf bis zu 600. Es ist fragwürdig, dass dieser Fehler alleinig von der Perturbations-Methode verursacht wird und kann wohl verneint werden. Vielmehr wird auch die Fixpunkt-Iteration einen großen Anteil an der Differenz haben, was durch weitere Approximationen, wie beispielsweise durch die Dynamische Programmierung, zu bestätigen wäre.
Betrachtet man die Punkte, bei denen sich die Differenz zwischen beiden Methoden auf unter 0.1 beläuft, so ist dies für $\kappa = 1$ noch für eine verhältnismäßig große Anzahl der Fall, siehe Abbildung 5.17. Mit steigendem κ wird dieser Teil der Gitterpunkte natürlich immer geringer, weshalb man für $\kappa = 10$ nur noch einen schmalen Streifen nahe des Gleichgewichts und bei $\kappa = 15$ lediglich einige derartige Punkte findet. Dies zeigen uns die Abbildungen 5.18 und 5.19, in denen die Punkte mit Differenzen unter 0.1 grün eingefärbt wurden. Dabei lässt sich die Approximation für $\kappa = 2$ noch in gewissen Bereichen verwenden, wie wir im vorherigen Abschnitt gesehen haben.
Bei kleineren Abschnitten, wie beispielsweise $[2.0, 2.1] \times [-0.02, 0.02]$, belaufen sich die Fehler bei $\kappa = 15$ auf ca. 2, siehe Abbildung 5.20. Das heißt bereits in so geringer Entfernung vom Gleichgewichtspunkt sind die Differenzen schon sehr hoch, also vermutlich auch der jeweilige Fehler der Approximationen. Interessanterweise werden die geringsten Differenzen nicht in kleiner Umgebung um das Gleichgewicht angenommen, da sie hier eindeutig über 0.1 liegen, wir aber in Abbildung 5.19 Punkte finden, in denen die Differenz 0.1 unterschreitet. Vielmehr verschiebt sich der Bereich in Richtung der Intervallgrenze $x_2 = -0.32$. Der Grund liegt bei der Approximation durch die Perturbations-Methode. Betrachtet man sich die dadurch berechnete Wertefunktion von der Seite, also in einer zweidimensionalen x_2-$\widehat{V}(x)$-Grafik, so beschreibt die Kurve eine Parabel, deren Scheitel für $x_2 \approx -0.1$ angenommen wird. Da in diesem Bereich der Ab-

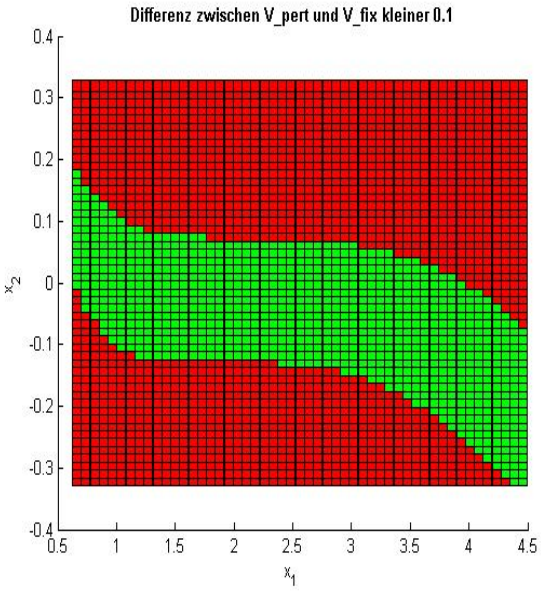

Abbildung 5.17: Differenz unter 0.1, $\kappa = 1$, auf $\Omega = [0.7, 4.5] \times [-0.32, 0.32]$

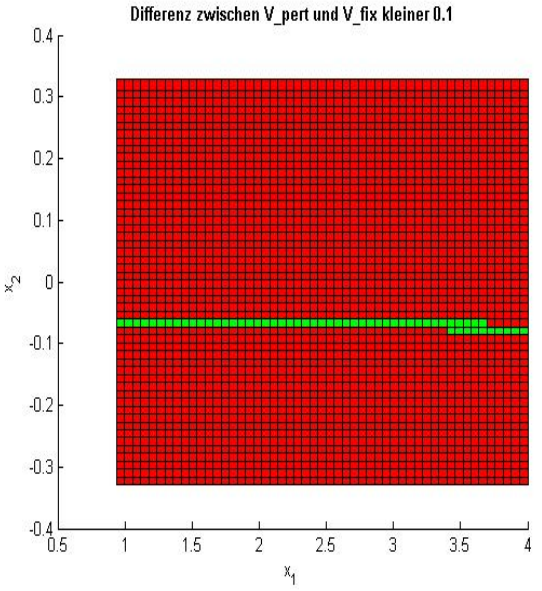

Abbildung 5.18: Differenz unter 0.1, $\kappa = 10$, auf $\Omega = [1.0, 4.0] \times [-0.32, 0.32]$

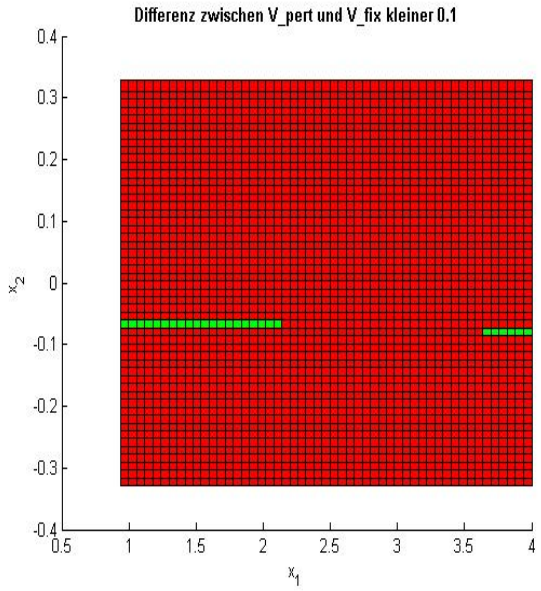

Abbildung 5.19: Differenz unter 0.1, $\kappa = 15$, auf $\Omega = [1.0, 4.0] \times [-0.32, 0.32]$

---

stand zur Approximation durch die Fixpunkt-Iteration am geringsten ist, ergibt sich die beobachtete Verschiebung.

Bis zu $\kappa = 2$ beläuft sich die Differenz auf dem gesamten Gebietsabschnitt $[2.0, 2.1] \times [-0.02, 0.02]$ noch auf unter 0.1. Für $\kappa = 1$ steigt sie sogar nur auf knapp 0.04. Die extrem hohen Fehler für die anderen κ auf dem gesamten Rechteck finden sich jedoch nicht für große Intervallabschnitte von x_1. So zeigen sich im Bereich $[1.1, 3.9] \times [-0.02, 0.02]$ in etwa die gleichen Fehlerwerte wie auf dem vorher betrachteten. Je größer dabei das κ ist, desto geringer ist der prozentuale Fehleranstieg. Deshalb liegen die Fehler für $\kappa = 15$ nur knapp über 2, während sich die Fehler für $\kappa = 1$ verdreifacht haben. Da wir lediglich das Intervall für x_1 vergrößert haben, ist der Grund im Verhältnis der Krümmungen in x_1- und x_2-Richtung zu suchen. Für steigendes κ verstärkt sich die Krümmung in x_2-Richtung, weshalb die in x_1-Richtung verhältnismäßig an Bedeutung verliert. Deshalb ist der Fehleranstieg hier prozentual geringer als für kleine κ, bei denen die Krümmung in x_1-Richtung stärker ins Gewicht fällt. Da durch stärkere Krümmung in x_2-Richtung das wirtschaftliche Risiko an Bedeutung gewinnt, ist das Verhalten auch anschaulich klar.

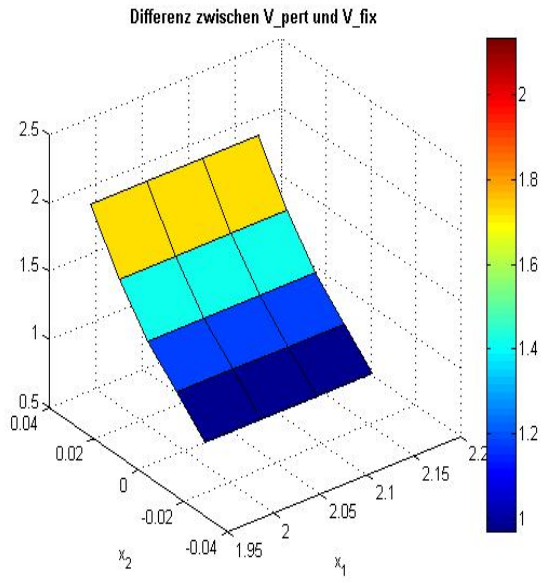

Abbildung 5.20: Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration, $\kappa = 15$, auf dem Abschnitt $\Omega = [2.0, 2.1] \times [-0.02, 0.02]$

Bei der Approximation auf dem großen Gebiet $[0.3, 8.6] \times [-0.32, 0.32]$ ergeben sich keine wesentlichen Verschlechterungen, da die Differenzen bereits auf dem kleinen Bereich sehr hoch sind. Allgemein beobachtet man das gleiche tendenzielle Verhalten, weshalb auf eine separate Betrachtung verzichtet wird.

Veränderung von σ

Als weitere σ-Werte wurden in dieser Arbeit 0.01 und 0.012 gewählt. Dadurch muss das betrachtete Intervall für x_2 auf $[-0.4, 0.4]$ beziehungsweise $[-0.48, 0.48]$ vergrößert werden, um zu gewährleisten, dass die Werte für die zweite Komponente von φ, also $\rho x_2 + z$, immer im gegebenen Intervall liegen. Dadurch erfährt wiederum das im erweiterten Modell gegebene Risiko eine größere Bedeutung.
Natürlich verschlechtern sich somit auch die Approximationen der Wertefunktion. Dabei nimmt die Differenz zwischen beiden Approximation mit steigendem σ immer stärker

zu. Während sie sich für $\kappa = 1$ und $\sigma = 0.008$ auf dem Rechteck $[0.7, 4.5]\times[-0.32, 0.32]$ auf etwa 0.7 belaufen, steigen sie für $\sigma = 0.012$ auf ca. 1.3 an. Bei $\kappa = 15$ auf dem Gebiet $[1.0, 4.0] \times [-0.32, 0.32]$ ergeben sich Unterschiede von etwa 550 bei $\sigma = 0.008$ und ca. 6000 bei $\sigma = 0.012$. Auch hier zeigt sich also, dass die Approximationen der stochastischen Unsicherheit nicht gerecht werden können.

Für $\kappa = 15$ ist zumindest die Perturbations-Methode, aber vermutlich auch die Fixpunkt-Iteration, nicht mehr als Approximation verwendbar. Versucht man wiederum die Punkte, in denen die Differenz unter 0.1 liegt, in grüner Farbe zu plotten, so kann man derartige Punkte für $\sigma = 0.01$ und $\sigma = 0.012$ nicht finden. Auch für $\kappa = 10$ sieht die Situation nicht besser aus. Es existieren lediglich einige wenige Punkte mit der gewünschten Eigenschaft. Selbst $\kappa = 5$ liefert nur einen schmalen Streifen in der Nähe des Gleichgewichts, in dem die Differenz 0.1 unterschreitet, siehe Abbildung 5.21. Dabei finden sich diese Punkte abermals nicht in einer Umgebung um das Gleichgewicht, sondern geringfügig zur Intervallgrenze $x_2 = -0.48$ hin verschoben. Wie bereits erwähnt hängt dies damit zusammen, dass die Perturbations-Methode eine Parabel für x_2 beschreibt, deren Scheitel bei ca. -0.2 liegt, in dessem Bereich sich auch die geringsten Differenzen ergeben.

Wiederum sind es $\kappa = 1$ und 2, die noch in weitgehend großen Bereichen eine Differenz unter 0.1 aufweisen. Um die Entwicklung dieser Bereiche für $\kappa = 1$ unter verschiedenen σ zu beobachten, sind hier zur Abbildung 5.17 für $\sigma = 0.008$ die Abbildungen 5.22 und 5.23 für $\sigma = 0.01$ beziehungsweise $\sigma = 0.012$ beigefügt. Wie man wohl bereits erwartet hat, wird der Bereich, in dem die Differenz 0.1 unterschreitet, mit zunehmendem σ immer geringer. Das zeigt sich durchweg für alle κ und steigendes σ, was mit der erhöhten wirtschaftlichen Unsicherheit durch ein größeres σ zu erklären ist. Da dadurch der stochastische Schock verstärkt wird, kann er in den Berechnungen nicht entsprechend aufgefangen werden. Für $\kappa = 15$ führt dies schließlich dazu, dass für $\sigma = 0.01$ und $\sigma = 0.012$ keinerlei Punkte mehr existieren, bei denen die Differenz kleiner 0.1 ist.

Betrachtet man wiederum das Gebiet $[2.0, 2.1] \times [-0.02, 0.02]$, so erhöht sich der Fehler von ca. 2 für $\kappa = 15$ und $\sigma = 0.008$ auf über 3 bei $\sigma = 0.01$ und sogar auf bis zu 4.5 bei $\sigma = 0.012$, siehe Abbildung 5.24. Für $\kappa = 1$ macht sich ein größeres σ nicht derart

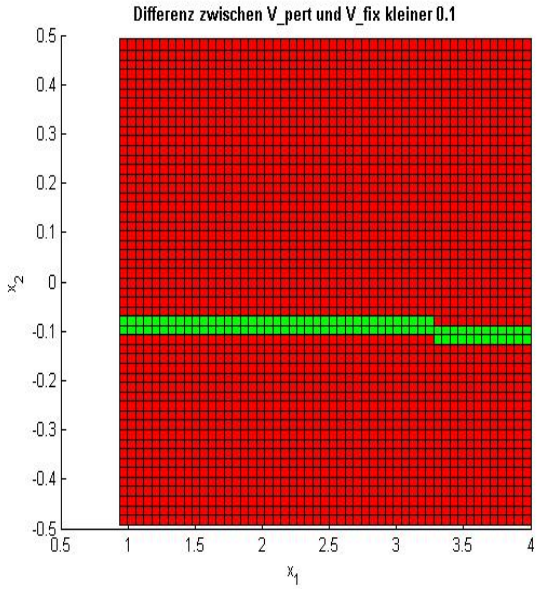

Abbildung 5.21: Differenz unter 0.1, $\kappa = 5$, $\sigma = 0.012$, $\Omega = [1.0, 4.0] \times [-0.48, 0.48]$

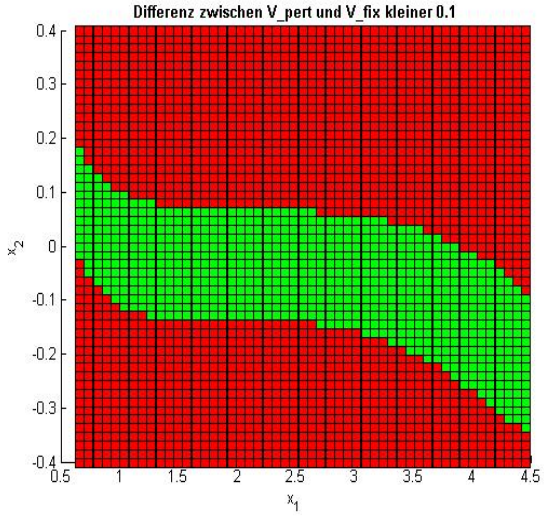

Abbildung 5.22: Differenz unter 0.1, $\kappa = 1$, $\sigma = 0.01$, $\Omega = [0.7, 4.5] \times [-0.4, 0.4]$

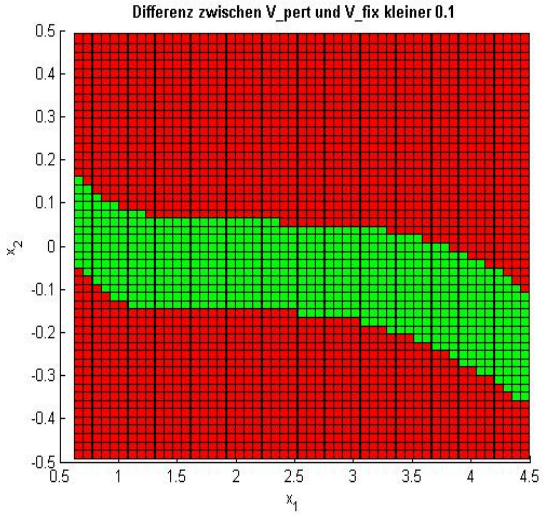

Abbildung 5.23: Differenz unter 0.1, $\kappa = 1$, $\sigma = 0.012$, $\Omega = [0.7, 4.5] \times [-0.48, 0.48]$

---

stark bemerkbar. Hier steigt der Fehler von etwa 0.4 auf nur ca. 0.7. Anschaulich ist dies klar, da bei $\kappa = 1$ dem σ und somit dem wirtschaftlichen Risiko eine weitaus geringer Gewichtung widerfährt, was sich in dem kleinen Wert für $V_{\sigma\sigma}$ zeigt.
Wiederum bleiben die Fehler für den Bereich $[1.1, 3.9] \times [-0.02, 0.02]$ in etwa auf dem gleichen Niveau, wie wir bereits vorher gesehen haben. Dabei erkennt man, dass sich bei $\kappa = 1$ die Fehler für zunehmendes σ wesentlich gerinfügiger vergrößern als bei $\kappa = 15$. Sie steigen lediglich von 0.12 auf 0.15, siehe Abbildungen 5.25 und 5.26, während bei $\kappa = 15$ die Differenz wiederum von ca. 2 auf knapp 5 ansteigt.
Für die Approximation auf dem großen Gebiet von $[0.3, 8.6] \times [-0.32, 0.32]$ ergeben sich abermals keine wesentlichen Verschlechterungen, da die Differenzen ja ohnehin in sehr hohen Bereichen verlaufen. Deshalb sei auch hier auf eine explizite Betrachtung verzichtet.

Abschließend lässt sich feststellen, dass etliche Faktoren die verwendeten Approximationen verschlechtern. Hierzu zählen neben steigendem κ und σ bei beiden Methoden auch eine zunehmende Entfernung vom Gleichgewicht bei der Perturbations-Methode.

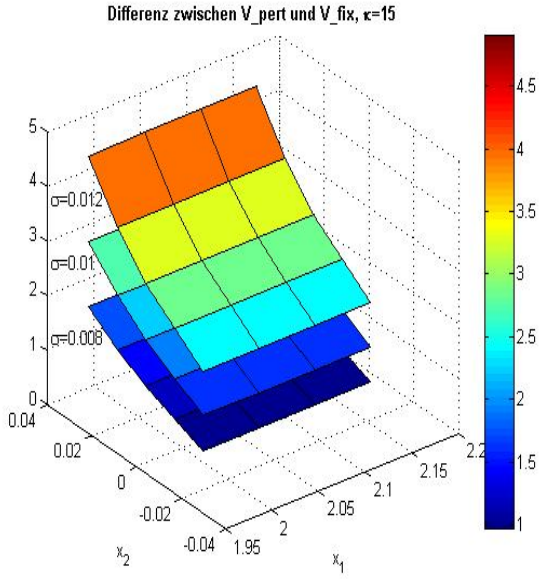

Abbildung 5.24: Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration, $\kappa = 15$, auf dem Rechteck $\Omega = [2.0, 2.1] \times [-0.02, 0.02]$

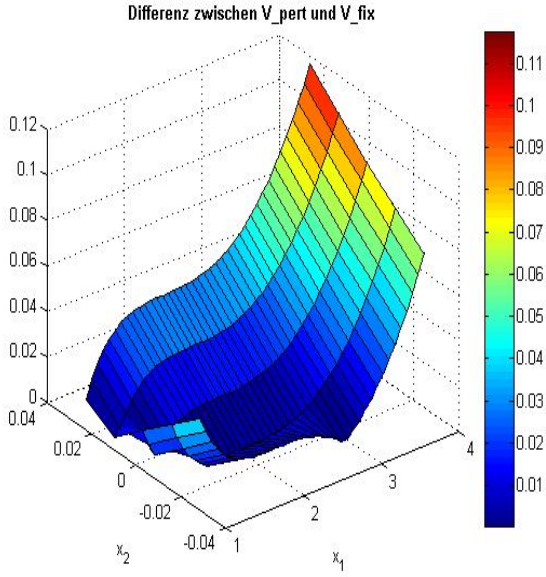

Abbildung 5.25: Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration, $\kappa = 1$, $\sigma = 0.008$, auf dem Rechteck $\Omega = [1.1, 3.9] \times [-0.02, 0.02]$

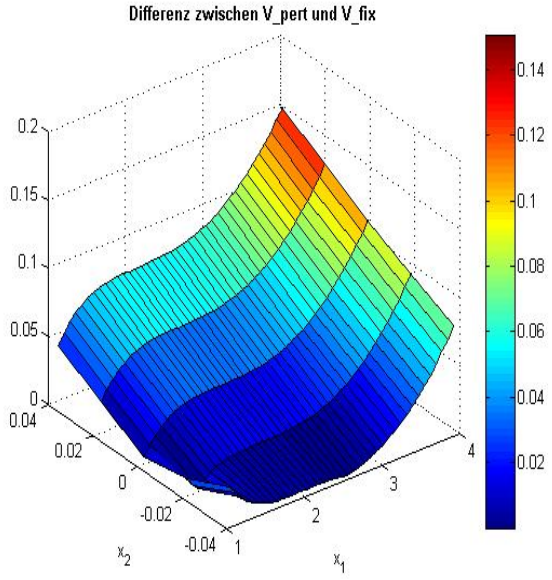

Abbildung 5.26: Differenz zwischen Perturbations-Methode und Fixpunkt-Iteration, $\kappa = 1$, $\sigma = 0.012$, auf dem Rechteck $\Omega = [1.1, 3.9] \times [-0.02, 0.02]$

Für kleine κ liefert insbesondere die Fixpunkt-Iteration gute bis akzeptable Ergebnisse. Die Perturbations-Methode hingegen besticht vor allem in geringer Entfernung vom Gleichgewicht. Man stößt hierbei aber sehr schnell an die Grenzen dieser Methode, da die Umgebungen, in denen der Fehler akzeptabel ist, meist sehr klein sind. Deshalb ist für konstantere Genauigkeiten, zumindest für geringe κ, in jedem Fall die Fixpunkt-Iteration zu bevorzugen.

6 Fazit der praktischen Ergebnisse und Ausblick

Im letzten Kapitel haben wir gesehen, wie sich die Wertefunktion V und die Genauigkeit ihrer Approximationen auf unterschiedlichen Gebieten und für die verschiedenen Parameter verhalten. Dabei hatten wir für $\kappa = 0$ die Möglichkeit eines direkten Vergleichs der Approximationen mit der exakten Lösung.

Daraus wurde ersichtlich, dass die Perturbations-Methode insbesondere nahe des Gleichgewichtspunktes eine hervorragende Approximation liefert. Es werden hier sehr geringe Fehlerwerte gemessen, was in der Funktionsweise des Algorithmus begründet liegt. Dieser verwendet als Ausgangspunkt das Gleichgewicht und entwickelt damit die Approximation durch eine Taylor-Reihe. Leider steigen die Fehler dieser Methode mit zunehmender Entfernung vom Entwicklungspunkt rapide an. Besonders macht sich dies in dem größeren der beiden untersuchten Bereiche bemerkbar.
Die Fixpunkt-Iteration hingegen liefert Fehlerwerte, die über das gesamte Gebiet konstanter sind. Zwar weist sie demnach an den Bereichsgrenzen nicht so hohe Fehler auf wie die Perturbations-Methode, ist dafür jedoch im Gleichgewicht nicht derart genau. Dennoch ergeben sich hier stets nur sehr geringe Fehler. Problematisch ist bei der Fixpunkt-Iteration, dass sie die Approximationen der Kontrolle durch die Perturbations-Methode verwenden muss. Dadurch sind einige Einschränkungen des Bereichs nötig und die gegebene Ungenauigkeit in der approximierten Kontrolle überträgt sich auf die Fixpunkt-Iteration.
Bezüglich des Aufwands beider Methoden sind nur schwer Aussagen zu treffen, da beispielsweise für die Perturbations-Methode etliche Berechnungen bereits im Vorfeld stattfinden. Auf der anderen Seite verwendet die Fixpunkt-Iteration als Ausgangspunkt die

durch die Perturbations-Methode ermittelte Wertefunktion, weshalb sie anschaulich natürlich einen höheren Aufwand aufweisen wird. Nach dem Wesen der Fixpunkt-Iteration könnte hier aber jeglicher Anfangspunkt in einem gewissen Bereich für das gleiche Ergebnis verwendet werden. Zudem erhält man hier insbesondere für das kleinere Gebiet ein größere Genauigkeit. Dies kann noch durch eine höhere Anzahl von Gitterpunkten verbessert werden, wodurch jedoch auch der Aufwand der Methode steigt.

Im erweiterten Modell fehlt uns leider der direkte Vergleich mangels einer exakten Lösung. Jedoch lassen sich zum Teil die vorher getroffenen Schlüsse übertragen. So bleibt die Perturbations-Methode in der Nähe des Gleichgewichts genauer, während die Fixpunkt-Iteration vermutlich einen insgesamt geringeren Fehler aufweist. Obige Probleme durch die Approximation der Kontrolle bleiben jedoch erhalten.
Mit Hilfe einer Erhöhung der Anzahl der Gitterpunkte lässt sich die Fixpunkt-Iteration wiederum verbessern. Insbesondere für kleine κ bestätigt sich dadurch die Vermutung, dass die Perturbations-Methode nahe des Gleichgewichts und die Fixpunkt-Iteration auch noch in größerer Entfernung günstig verlaufen.
Für steigendes κ und σ verschlechtern sich die Approximationen zunehmend. Es wird hier schwer, noch Aussagen über die Fehler der einzelnen Methoden zu treffen, da die Differenzen zwischen den Approximationen auf zu große Werte steigen.

Eine Entscheidung zu Gunsten einer der beiden Methoden muss immer im Hinblick auf die gewünschten Anforderung geschehen. Sind nur kleine Umgebungen um das Gleichgewicht von Bedeutung, so ist in jedem Fall die Perturbations-Methode zu bevorzugen. Da die Umgebungen mit großer Genauigkeit aber sehr klein sind, fällt die Entscheidung wohl meist auf die Fixpunkt-Iteration, da sie eine gewisse Genauigkeit auf einem Großteil des Gebiets liefert.
Wird $\kappa = 1$ oder $\kappa = 2$ gewählt, so sind diese Überlegungen durchaus noch gültig. Zwar fehlt uns hierfür eine zuverlässige Grundlage, jedoch scheinen sich die Vermutungen mit den erhaltenen Resultaten zu decken. Dies ändert sich für höhere Werte von κ. Da die Differenzen hierbei immens groß werden, bedarf es weiterer Untersuchungen, um sich

zu Gunsten einer Approximation zu entscheiden, falls dies überhaupt noch möglich ist. Es ist anzunehmen, dass beide Methoden einen erheblichen Fehler, insbesondere an den Bereichsgrenzen, aufweisen. Über die Genauigkeit der Approximationen in kleiner Umgebung des Gleichgewichtspunktes lässt sich nur schwer etwas aussagen. Dazu wären zusätzliche Vergleichsmöglichkeiten nötig, was Anregungen für weitere Diskussionen in diesem Bereich mit sich zieht.

A Notation

An dieser Stelle werden die wichtigsten Variablenbezeichnungen und Notationen übersichtsartig aufgeführt. Die Reihenfolge entspricht dabei dem erstmaligen Auftreten im Text.

$\mathbf{T}$	Zeitachse in diskreter Zeit
x_t	Zustand zum Zeitpunkt t in diskreter Zeit
u_t	Kontrolle zum Zeitpunkt t in diskreter Zeit
$\mathbf{U}$	Kontrollbereich
φ	Kontrollsystem in diskreter Zeit
Φ	Trajektorie in diskreter Zeit
$\bar{x}$	Zustand im Gleichgewicht
$\bar{u}$	Kontrolle im Gleichgewicht
J	diskontiertes Funktional in diskreter Zeit
β^t	Diskontfaktor in diskreter Zeit
$V(x)$	Optimale Wertefunktion in diskreter Zeit
f	Gleichgewichtsgleichung
H^t	Hamilton-Funktion in diskreter Zeit
λ	Kozustandsvektor
x_1	Zustandsvariable, die deterministischer Dynamik folgt
x_2	Zustandsvariable, die stochastischer Dynamik folgt
g	Kontrollfunktion im Modell
h	Funktion für Dynamik im Modell
$F(x, \sigma)$	Definierte Hilfsfunktion, um Terme für die Approximation zu berechnen
$\widetilde{f}$	äquivalente Form zu f
$\widehat{u}$	Approximation der Kontrolle u
$\widehat{x}_1$	Approximation der Zustandsvariablen x_1
$\widehat{x}_2$	Approximation der Zustandsvariablen x_2

$\widehat{V}$	Approximation der Wertefunktion durch die Perturbations-Methode
$\widetilde{V}$	Approximation der Wertefunktion durch die Fixpunkt-Iteration
Ω	der untersuchte Bereich
Γ	regelmäßiges Rechteckgitter
R_i	Rechtecke des Gitters
E_i	Knotenpunkte des Gitters
W	Raum der stetigen, stückweise affin bilinearen Funktionen

B Matlab-Quelltexte

B.1 Die Routine compute_Vcoeff.mw

```
> restart;
> with(linalg):
> Vcoeff := proc(g::algebraic, h::Vector, r::algebraic,
    beta::algebraic, p::algebraic)

    local V, Vc, Vh, A, Ay, Ayy, dim, z, b, Vcoeff,
        i, j, k, i1, i2, i3;

    # Taylor approximation for V
    V := V0 + V1*x[1] + V2*x[2] + V11*x[1]^2/2 +
         V12*x[1]*x[2] + V22*x[2]^2/2 + Vss*sigma^2/2;

    # list of unknown coefficients in V
    Vc := [V0, V1, V2, V11, V12, V22, Vss];

    # Taylor approximation for V(h)
    Vh := subs(x1=h[1],x2=h[2],subs(x[1]=x1, x[2]=x2, V)):
    print(Vh);

    # Compute the terms for the Taylor approximation
    # of the DP equation
    # r(g,h) + beta*V(h) - V = 0
    # and store the terms in Ay
```

```
Ay := Vector((p+1)^3):
i  := 0;
for i1 from 0 to p do
  for i2 from 0 to p do
    for i3 from 0 to p do
      j := i1 + (p+1)*i2 + (p+1)^2*i3 + 1:
      if (i1+i2+i3<=p) then
        Ayy := coeftayl(r(g,h[1],h[2])+expand(beta*Vh)-
               expand(V),[x[1],x[2],sigma] =
               [0,0,0],[i1,i2,i3]):
        if Ayy<>0 then
          i := i+1;
          Ay[i] := Ayy;
        end;
      end;
    od;
  od;
od;

dim := i;

z  := Vector(dim):
b  := Vector(dim):
A  := matrix(dim,dim):

# convert Ayy=0 into a system of linear equations
# Ax=b with x containing the values of the
# coefficientx of V
for i from 1 to dim do
  b[i] := 0:
```

```
      for k from 1 to dim do
        A[i,k] := simplify(coeff(Ay[i],Vc[k]));
        b[i] := b[i] + A[i,k]*Vc[k];
      od:
      b[i] := b[i] - Ay[i]:
    od:

    # solve the system
    z := linsolve(A,b);

    # output to the screen
    for k from 1 to dim do
      printf("%A = %A\n",Vc[k],z[k]);
    od;

    Vcoeff := z:
  end:

> # base line model

  beta := 0.95;

  g0   :=  4.3331;
  g1   :=  0.7126;
  g2   :=  4.3331;
  g11  := -0.2275;
  g12  :=  0.7126;
  g22  :=  4.3331;
  gss  :=  0.0;
```

```
  h11  :=  0.3400;
  h12  :=  2.0673;
  h21  :=  0.0;
  h22  :=  0.9;
  h111 := -0.1085;
  h112 :=  0.3400;
  h211 :=  0.0;
  h212 :=  0.0;
  h122 :=  2.0673;
  h212 :=  0.0;
  h222 :=  0.0;
  h1ss :=  0.0;
  h2ss :=  0.0;

  h := Vector([h11*x[1] + h12*x[2] + h111*x[1]^2/2 +
               h112*x[1]*x[2] + h122*x[2]^2/2 +
               h1ss*sigma^2/2,
               h21*x[1] + h22*x[2] + h211*x[1]^2/2 +
               h212*x[1]*x[2] + h222*x[2]^2/2 +
               h2ss*sigma^2/2]);

  g := g0 + g1*x[1] + g2*x[2] + g11*x[1]^2/2 +
       g12*x[1]*x[2] + g22*x[2]^2/2 + gss*sigma^2/2;

  r := (u,x1,x2)->ln(u);

  Vcoeff(g,h,r,beta,2):

> # extended model; kappa = 1
  # analog für kappa = 2, 5, 10, 15
```

```
beta := 0.95;

x1g  := 2.067344815;
x2g  := 0.0;

g0   :=  4.3331;
g1   :=  0.7126;
g2   :=  4.5304;
g11  := -0.2275;
g12  :=  0.7451;
g22  :=  4.7112:
gss  := -1.8955;

h11  :=  0.3400;
h12  :=  1.8700;
h21  :=  0.0;
h22  :=  0.9;
h111 := -0.1085;
h112 :=  0.3075;
h211 :=  0.0;
h212 :=  0.0;
h122 :=  1.6892;
h212 :=  0.0;
h222 :=  0.0;
h1ss :=  1.8955;
h2ss :=  0.0;

h := Vector([h11*x[1] + h12*x[2] + h111*x[1]^2/2 +
             h112*x[1]*x[2] + h122*x[2]^2/2 +
```

```
                h1ss*sigma^2/2,
                h21*x[1] + h22*x[2] + h211*x[1]^2/2 +
                h212*x[1]*x[2] + h222*x[2]^2/2 +
                h2ss*sigma^2/2]);

  g := g0 + g1*x[1] + g2*x[2] + g11*x[1]^2/2 +
       g12*x[1]*x[2] + g22*x[2]^2/2 + gss*sigma^2/2;

  kappa := 1;

  r := (u,x1,x2)->ln(u)*exp(kappa*x2);

  Vcoeff(g,h,r,beta,2):
```

B.2 Die Routine compute.m

```
function compute(kappa,domain)

% Call  : compute(kappa, domain)
%
% kappa : model parameter:
%         0 -> base line model
%         1, 2, 5, 10 or 15 -> extended model
%
% domain: computational domain (general intervals -
%         depending on kappa):
%         0 -> [1  ,  4] x [-4sigma, 4sigma]
%         1 -> [0.3, 8.8] x [-4sigma, 4sigma]
```

```
domaintext= ['[1,4]x[-4sigma,4sigma]    '; ...
             '[0.3,8.8]x[-4sigma,4sigma]'];

fprintf('\nComputation for kappa=%1.0f on %s\n\n', ...
        kappa, domaintext(domain+1,:));

% model parameters
beta=0.95;
alpha=0.34;
rho=0.9;
A=5;
sigma=0.008;

% parameters for Taylor approximations
if (kappa==0)
    x1_ggw=2.067344815;
    x2_ggw=0;
    u_ggw=4.333103529;
    gx=[0.7126 4.3331];
    gxx1=[-0.2275 0.7126];
    gxx2=[0.7126 4.3331];
    gss=0;
    V0    = 29.32566442;
    Vx1   =  0.2429172955;
    Vx2   = 10.18671572;
    Vx1x1 = -0.1174896957;
    Vx1x2 =  0.1404732835e-4;
    Vx2x2 =  0.4782692913e-3;
    Vss   =  0.0;
elseif (kappa==1)
```

```
    x1_ggw=2.067344815;
    x2_ggw=0;
    u_ggw=4.333103529;
    gx=[0.7126 4.5304];
    gxx1=[-0.2275 0.7451];
    gxx2=[0.7451 4.7112];
    gss   = -1.8955;
    V0    = 29.32566442;
    Vx1   =  0.2429172955;
    Vx2   = 19.28779790;
    Vx1x1 = -0.1174896957;
    Vx1x2 =  0.2086832225;
    Vx2x2 = 16.18475594;
    Vss   = -0.3876960000e-3;
elseif (kappa==2)
    x1_ggw=2.067344815;
    x2_ggw=0;
    u_ggw=4.333103529;
    gx=[0.7126 4.7277];
    gxx1=[-0.2275 0.7775];
    gxx2=[0.7775 5.0563];
    gss   = -7.5821;
    V0    = 29.32566442;
    Vx1   =  0.2429172955;
    Vx2   = 28.38888008;
    Vx1x1 = -0.1174896957;
    Vx1x2 =  0.4173523960;
    Vx2x2 = 42.71896378;
    Vss   = -0.1550800000e-2;
elseif (kappa==5)
```

```
    x1_ggw=2.067344815;
    x2_ggw=0;
    u_ggw=4.333103529;
    gx=[0.7126 5.3197];
    gxx1=[-0.2275 0.8749];
    gxx2=[0.8749 5.8933];
    gss   = -47.3879;
    V0    = 29.32566442;
    Vx1   =  0.2429172955;
    Vx2   = 55.69212663;
    Vx1x1 = -0.1174896957;
    Vx1x2 =  1.043359919;
    Vx2x2 = 184.4212544;
    Vss   = -0.9692560000e-2;
elseif (kappa==10)
    x1_ggw=2.067344815;
    x2_ggw=0;
    u_ggw=4.333103529;
    gx=[0.7126 6.3063];
    gxx1=[-0.2275 1.0371];
    gxx2=[1.0371 6.6274];
    gss   = -189.5516;
    V0    = 29.32566442;
    Vx1   =  0.2429172955;
    Vx2   = 101.1975376;
    Vx1x1 = -0.1174896957;
    Vx1x2 =  2.086705789;
    Vx2x2 = 627.5901572;
    Vss   = -0.3877040000e-1;
elseif (kappa==15)
```

```
    x1_ggw=2.067344815;
    x2_ggw=0;
    u_ggw=4.333103529;
    gx=[0.7126 7.2929];
    gxx1=[-0.2275 1.1994];
    gxx2=[1.1994 6.5354];
    gss   = -426.4912;
    V0    = 29.32566442;
    Vx1   =  0.2429172955;
    Vx2   = 146.7029485;
    Vx1x1 = -0.1174896957;
    Vx1x2 =  3.130051662;
    Vx2x2 = 1329.507185;
    Vss   = -0.8723240000e-1;
end;


% definition of computational domain
if (domain==1)
    if (kappa==15)
        x1anfang=1.0;
        x1ende=6.4;
    elseif(kappa==10)
        x1anfang=0.6;
        x1ende=7.3;
    elseif(kappa==5)
        x1anfang=0.4;
        x1ende=8.1;
    elseif(kappa==2)
        x1anfang=0.3;
```

```
        x1ende=8.6;
    elseif(kappa==1)
        x1anfang=0.3;
        x1ende=8.7;
    else
        x1anfang=0.3;
        x1ende=8.8;
    end;
    x2anfang=-40*sigma;
    x2ende=40*sigma;
    x1schritt = (x1ende-x1anfang)/50;
    x2schritt = (x2ende-x2anfang)/50;
else
    x1anfang=1.0;
    x1ende=4.0;
    if(kappa==2)
        x1anfang=0.8;
        x1ende=4.2;
    elseif(kappa==1)
        x1anfang=0.7;
        x1ende=4.5;
    end;
    x2anfang=-40*sigma;
    x2ende=40*sigma;
    x1schritt = (x1ende-x1anfang)/50;
    x2schritt = (x2ende-x2anfang)/50;
end;


% definition of grid nodes
```

```
[X1,X2] = meshgrid(x1anfang:x1schritt:x1ende, ...
                   x2anfang:x2schritt:x2ende);
x1 = X1(1,:);
x2 = X2(:,1);
laengex1 = size(x1,2);
laengex2 = size(x2,1);


% computation of exact solution if kappa=0
if (kappa==0)
    % components of formula for V(x)=B+C*ln(x1)+D*x2
    B= (log((1-beta*alpha)*A)+((beta*alpha)/ ...
       (1-beta*alpha))*log(beta*alpha*A))/(1-beta);
    C=alpha/(1-alpha*beta);
    D=1/((1-alpha*beta)*(1-rho*beta));

    % computation of V_exakt and U_exakt at grid nodes
    for i=1:laengex1
        for j=1:laengex2
            V_exakt(j,i)=B +C*log(x1(i)) +D*x2(j);
            U_exakt(j,i)=(1-alpha*beta)*A*exp(x2(j)) ...
                         * x1(i)^alpha;
        end
    end

    % computation of V_exakt and U_exakt in equilibrium
    V_exakt_ggw = B +C*log(x1_ggw) +D*x2_ggw;
    U_exakt_ggw =(1-alpha*beta)*A*exp(x2_ggw)*x1_ggw^alpha;
end
```

```
% computation of optimal control U_pert by Taylor
% approximation
% in the grid nodes ...
for i=1:laengex1
    for j=1:laengex2
        U_pert(j,i)=u_ggw + gx(1)*(x1(i)-x1_ggw)  ...
                    + gx(2)*(x2(j)-x2_ggw)        ...
                    + 0.5*(gxx1(1)*(x1(i)-x1_ggw) ...
                    * (x1(i)-x1_ggw)              ...
                    + gxx1(2)*(x2(j)-x2_ggw)      ...
                    * (x1(i)-x1_ggw))             ...
                    + 0.5*(gxx2(1)*(x1(i)-x1_ggw) ...
                    * (x2(j)-x2_ggw)              ...
                    + gxx2(2)*(x2(j)-x2_ggw)      ...
                    * (x2(j)-x2_ggw))             ...
                    + 0.5*(gss*sigma*sigma);      ...
    end
end
% ... and in the equilibrium
U_pert_ggw = u_ggw + 0.5*(gss*sigma*sigma);


% computation of V_pert by Taylor approximation
%in the grid nodes...
for i=1:laengex1
    for j=1:laengex2
        V_pert(j,i) = V0 + Vx1*(x1(i)-x1_ggw)     ...
                    + Vx2*(x2(j)-x2_ggw)          ...
                    + 0.5*(Vx1x1*(x1(i)-x1_ggw)   ...
```

```
                    * (x1(i)-x1_ggw)                ...
                    + Vx1x2*(x2(j)-x2_ggw)          ...
                    * (x1(i)-x1_ggw))               ...
                    + 0.5*(Vx1x2*(x1(i)-x1_ggw)     ...
                    * (x2(j)-x2_ggw)                ...
                    + Vx2x2*(x2(j)-x2_ggw)          ...
                    * (x2(j)-x2_ggw))               ...
                    + 0.5*(Vss*sigma*sigma);        ...
    end
end
%... and in the equilibrium
V_pert_ggw = V0 + 0.5*(Vss*sigma*sigma);

%saving functions in ASCII-files
if ((kappa==0) & (domain==0))
  save('U_pert0klein.asc', 'U_pert', '-ASCII');
  save('V_pert0klein.asc', 'V_pert', '-ASCII');
  save('U_exakt0klein.asc', 'U_exakt', '-ASCII');
  save('V_exakt0klein.asc', 'V_exakt', '-ASCII');
elseif ((kappa==0) & (domain==1))
  save('U_pert0gross.asc', 'U_pert', '-ASCII');
  save('V_pert0gross.asc', 'V_pert', '-ASCII');
  save('U_exakt0gross.asc', 'U_exakt', '-ASCII');
  save('V_exakt0gross.asc', 'V_exakt', '-ASCII');
elseif ((kappa==1) & (domain==0))
  save('U_pert1klein.asc', 'U_pert', '-ASCII');
  save('V_pert1klein.asc', 'V_pert', '-ASCII');
elseif ((kappa==1) & (domain==1))
  save('U_pert1gross.asc', 'U_pert', '-ASCII');
  save('V_pert1gross.asc', 'V_pert', '-ASCII');
```

```
elseif ((kappa==2) & (domain==0))
  save('U_pert2klein.asc', 'U_pert', '-ASCII');
  save('V_pert2klein.asc', 'V_pert', '-ASCII');
elseif ((kappa==2) & (domain==1))
  save('U_pert2gross.asc', 'U_pert', '-ASCII');
  save('V_pert2gross.asc', 'V_pert', '-ASCII');
elseif ((kappa==5) & (domain==0))
  save('U_pert5klein.asc', 'U_pert', '-ASCII');
  save('V_pert5klein.asc', 'V_pert', '-ASCII');
elseif ((kappa==5) & (domain==1))
  save('U_pert5gross.asc', 'U_pert', '-ASCII');
  save('V_pert5gross.asc', 'V_pert', '-ASCII');
elseif ((kappa==10) & (domain==0))
  save('U_pert10klein.asc', 'U_pert', '-ASCII');
  save('V_pert10klein.asc', 'V_pert', '-ASCII');
elseif ((kappa==10) & (domain==1))
  save('U_pert10gross.asc', 'U_pert', '-ASCII');
  save('V_pert10gross.asc', 'V_pert', '-ASCII');
elseif ((kappa==15) & (domain==0))
  save('U_pert15klein.asc', 'U_pert', '-ASCII');
  save('V_pert15klein.asc', 'V_pert', '-ASCII');
elseif ((kappa==15) & (domain==1))
  save('U_pert15gross.asc', 'U_pert', '-ASCII');
  save('V_pert15gross.asc', 'V_pert', '-ASCII');
end;
```

B.3 Die Routine fixed_point.cpp

```
/*Call  : a kappa, domain
```

```
kappa : model parameter:
        0 -> base line model
        1, 2, 5, 10, 15 -> extended model

domain: computational domain (general intervals -
        depending on kappa):
        0 -> [1  ,  4] x [-4sigma, 4sigma]
        1 -> [0.3,8.8] x [-4sigma, 4sigma]
*/



#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <cmath>
#include <new>
#include <string>

using namespace std;


//model parameters
int const nodes = 51;
int const numpoints = 11;
double sigma = 0.008;

double beta = 0.95, alpha = 0.34, rho = 0.9,
       idelta = 1e-006;
```

```
int A = 5;
double y1[nodes][nodes], y2[nodes][numpoints],
       w[2][numpoints], v[numpoints];


//function varphi
void dynamik (double x[2], double x1start, double x1end,
              double u, double w1[numpoints], int j, int i)
{
  double n, eps;

  y1[j][i] = (double) (A * exp(x[1]) * pow(x[0], alpha) - u);

  for (int k=0; k<numpoints; k++)
  {
    eps = w1[k];
    n = rho * x[1] + eps;
    y2[j][k] = n;
  }
}

//utility function
double ertrag (double x[2], double u, int kappa)
{
  double r;

  if (kappa == 0)
    r = log(u);
  else
    r = log(u) * exp(kappa * x[1]);
```

```
  return r;
}

//distribution function, N(0,sigma)
void appintweight (double sigma)
{
  double xx[numpoints], ww[numpoints];
  double l, m, n = 0;

  for (int i=0; i<numpoints; i++)
  {
    l = -4*sigma + i * (8*sigma / (numpoints-1));
    m = 1 / (sqrt(2*3.141592654) * sigma)
        * exp( -pow(l,2) / (2*pow(sigma,2)));
    n = n + m;

    xx[i] = l;
    ww[i] = m;
  }

  for (int i=0; i<numpoints; i++)
  {
    ww[i] = ww[i] / n;
    w[0][i] = xx[i];
    w[1][i] = ww[i];
  }
}

//interpolation with affin bilinear functions
```

```
void linInterp (double x1[nodes], double x2[nodes],
                double V_pert[nodes][nodes], double y1,
                double y2help[numpoints])
{
  int i, j;
  double h1, h2, z1, z2, m0, m1, m2, m3, help = 0;

  h1 = (x1[nodes - 1] - x1[0]) / (nodes - 1);
  h2 = (x2[nodes - 1] - x2[0]) / (nodes - 1);


  i = (int) ((y1 - x1[0]) / h1);
  z1 = (y1 - x1[i]) / (x1[i+1] - x1[i]);

  for (int m=0; m<numpoints; m++)
  {
    j = (int) ((y2help[m] - x2[0]) / h2);

    z2 = (y2help[m] - x2[j]) / (x2[j+1] - x2[j]);

    m0 = (1 - z1) * (1 - z2);
    m1 =      z1  * (1 - z2);
    m2 = (1 - z1) * z2;
    m3 =      z1  * z2;

    help = m0*V_pert[j][i] + m1*V_pert[j][i+1] +
           m2*V_pert[j+1][i] + m3*V_pert[j+1][i+1];

    v[m] = help;
  }
```

```
}


int main(int argc, char **argv)
{
double x[2], x1[nodes], x2[nodes], y2help[numpoints],
       w1[numpoints], w2[numpoints];
double res, v0, absErr, x1start, x1end, x2start, x2end;
int iter, kappa;
double V_pert[nodes][nodes], U_pert[nodes][nodes];
char s[30], t[30], o[30];


FILE *fp, *fp2, *fp3;

//usage of arguements
if (argc != 3)
{
  cout << endl << "Falsche Anzahl der Parameter!" <<
          endl << endl;
  cout << "Benutzung: " << argv[0] << " kappa domain" <<
          endl << endl;
  cout << "kappa: 0 fuer das Basis-Modell, 1, 2, 5, 10
           oder 15 fuer das erweiterte Modell" << endl;
  cout << "domain: 0 fuer das kleine Gebiet, 1 fuer das
           grosse Gebiet" << endl;
  exit(0);
}
kappa = (int) atof(argv[1]);
if ((kappa!=0 && kappa!=1 && kappa!=2 && kappa!=5 &&
```

```
     kappa!=10 && kappa!=15) || ((int) atof(argv[2]) != 0
     && (int) atof(argv[2]) != 1))
{
  if ((int) atof(argv[2]) != 0 && (int) atof(argv[2]) != 1)
  {
    if (kappa!=0 && kappa!=1 && kappa!=2 && kappa!=5 &&
        kappa!=10 && kappa!=15)
    {
      cout << endl << "Falsche Eingaben fuer das Modell
                       und die Domain!" << endl << endl;
      cout << "Benutzung: " << argv[0] << " kappa domain"
           << endl << endl;
      cout << "kappa: 0 fuer das Basis-Modell, 1, 2, 5, 10
                oder 15 fuer das erweiterte Modell" << endl;
      cout << "domain: 0 fuer das kleine Gebiet, 1 fuer das
                grosse Gebiet" << endl;
      exit(0);
    }
    cout << endl << "Falsche Eingabe fuer die Domain!" <<
                     endl << endl;
    cout << "Richtige Eingabe: 0 fuer das kleine Gebiet,
              1 fuer das grosse Gebiet" << endl;
    exit(0);
  }
  cout << endl << "Falsche Eingabe fuer das Modell!" <<
                   endl << endl;
  cout << "Richtige Eingabe: 0 fuer das Basis-Modell, 1, 2,
           5, 10 oder 15 fuer das erweiterte Modell" << endl;
  exit(0);
}
```

```
//definition of computational domain
if ((int) atof(argv[2]) == 0)
{
  x1start = 1.0;
  x1end = 4.0;
  if (kappa==2)
  {
    x1start = 0.8;
    x1end = 4.2;
  }
  else if (kappa==1)
  {
    x1start = 0.7;
    x1end = 4.5;
  }
  x2start = -40*sigma;
  x2end = 40*sigma;
}
else
{
  if (kappa==15)
  {
    x1start = 1.0;
    x1end = 6.4;
  }
  else if (kappa==10)
  {
    x1start = 0.6;
    x1end = 7.3;
```

```
  }
  else if (kappa==5)
  {
    x1start = 0.4;
    x1end = 8.1;
  }
  else if (kappa==2)
  {
    x1start = 0.3;
    x1end = 8.6;
  }
  else if (kappa==1)
  {
    x1start = 0.3;
    x1end = 8.7;
  }
  else if (kappa==0)
  {
    x1start = 0.3;
    x1end = 8.8;
  }
  x2start = -40*sigma;
  x2end = 40*sigma;
}

//loading files
strcpy(s,"V_pert");
strcpy(t,"U_pert");
strcpy(o,"V_fix");
strcat(s,argv[1]);
```

```
strcat(t,argv[1]);
strcat(o,argv[1]);
if ((int) atof(argv[2])==0)
{
  strcat(s,"klein");
  strcat(t,"klein");
  strcat(o,"klein");
}
else
{
  strcat(s,"gross");
  strcat(t,"gross");
  strcat(o,"gross");
}
strcat(s,".asc");
strcat(t,".asc");
strcat(o,".asc");
fp = fopen(s,"r");
fp2 = fopen(t,"r");

//loading values from files
if (fp == NULL && fp2 == NULL)
  cout << "Datei existiert nicht!" << endl;
else
{
  for (int i=0; i<nodes; i++)
  {
    for (int j=0; j<nodes; j++)
    {
      fscanf(fp,"%lf\n",&V_pert[i][j]);
```

```
      fscanf(fp2,"%lf\n",&U_pert[i][j]);
    }
  }
  fclose(fp);
  fclose(fp2);
}

//definition of grid nodes
for (int i=0; i<nodes; i++)
{
  x1[i] = x1start + i*((x1end-x1start)/(nodes-1));
  x2[i] = x2start + i*((x2end-x2start)/(nodes-1));
  x2[(nodes-1)/2] = 0;
}

res = 2*idelta;
appintweight (sigma);
iter = 0;

//fixed-point iteration
while (res > idelta && iter<300)
{
  res = 0.0;
  iter = iter + 1;
  for (int i=0; i<nodes; i++)
  {
    for (int j=0; j<nodes; j++)
    {
      x[0] = x1[i];
      x[1] = x2[j];
```

```
if (iter == 1)
{
  for (int l=0; l<numpoints; l++)
    w1[l] = w[0][l];
  dynamik (x, x1start, x1end, U_pert[j][i], w1, j, i);
}

for (int l=0; l<numpoints; l++)
  y2help[l] = y2[j][l];

linInterp (x1, x2, V_pert, y1[j][i], y2help);

v0 = V_pert[j][i];
V_pert[j][i] = 0;
for (int l=0; l<numpoints; l++)
  w2[l] = w[1][l];

for (int k=0; k<numpoints; k++)
  V_pert[j][i] = V_pert[j][i] + w2[k] * v[k];

V_pert[j][i] = beta * V_pert[j][i];
V_pert[j][i] = V_pert[j][i] +
               ertrag (x, U_pert[j][i], kappa);

absErr = v0 - V_pert[j][i];
absErr = abs(absErr);

if (absErr > res)
res = absErr;
```

```
    }
  }
  cout << iter << "\t";
}

//saving file
fp3 = fopen(o,"w+");
for (int i=0; i<nodes; i++)
{
  for (int j=0; j<nodes; j++)
  fprintf(fp3,"%f %f %f\n",x1[i],x2[j],V_pert[j][i]);
}
fclose(fp3);

return(0);
}
```

B.4 Die Routine show51.m

```
function show51(kappa,domain,plotdomain)

% Call  : show51(kappa, domain, plotdomain)
%
% kappa : model parameter:
%         0 -> base line model
%         1, 2, 5, 10 or 15 -> extended model
%
% domain: computational domain (general intervals -
%         depending on kappa):
%         0 -> [1  ,  4] x [-4sigma, 4sigma]
```

```
%         1 -> [0.3,8.8] x [-4sigma, 4sigma]
%
% plotdomain: domain for plots:
%         0 -> computational domain
%         1 to 6 -> smaller domains (see line 137 to 178)


% domains for error computation

x1int = [ [ 2.0,  2.1 ] ; ...
          [ 1.8,  2.5 ] ; ...
          [ 1.5,  3.5 ] ; ...
          [ 1.5,  3.5 ] ; ...
          [ 1.0,  4.0 ] ; ...
          [ 0.5,  6.0 ] ; ...
          [ 0.5,  8.0 ] ;
          [ 0.3,  8.8 ] ];

x2int = [ [-0.01, 0.01 ] ; ...
          [-0.05, 0.05 ] ; ...
          [-0.10, 0.10 ] ; ...
          [-0.30, 0.30 ] ; ...
          [-0.32, 0.32 ] ; ...
          [-0.32, 0.32 ] ; ...
          [-0.32, 0.32 ] ; ...
          [-0.32, 0.32 ] ];

domaintext= ['[1,4]x[-4sigma,4sigma]    '; ...
             '[0.3,8.8]x[-4sigma,4sigma]'];
```

```
fprintf('\nComputation for kappa=%1.0f on %s\n\n', ...
        kappa, domaintext(domain+1,:));

% model parameters
beta=0.95;
alpha=0.34;
rho=0.9;
A=5;
sigma=0.008;

% parameters for Taylor approximations
x1_ggw=2.067344815;
x2_ggw=0;
u_ggw=4.333103529;

if (kappa==0)
    gss=0;
    V0    = 29.32566442;
    Vss   =  0.0;
elseif (kappa==1)
    gss   = -1.8955;
    V0    = 29.32566442;
    Vss   = -0.3876960000e-3;
elseif (kappa==2)
    gss   = -7.5821;
    V0    = 29.32566442;
    Vss   = -0.1550800000e-2;
elseif (kappa==5)
    gss   = -47.3879;
    V0    = 29.32566442;
```

```
    Vss   = -0.9692560000e-2;
elseif (kappa==10)
    gss   = -189.5516;
    V0    = 29.32566442;
    Vss   = -0.3877040000e-1;
elseif (kappa==15)
    gss   = -426.4912;
    V0    = 29.32566442;
    Vss   = -0.8723240000e-1;
end;


% definition of computational domain
if (domain==1)
    if (kappa==15)
        x1anfang=1.0;
        x1ende=6.4;
    elseif(kappa==10)
        x1anfang=0.6;
        x1ende=7.3;
    elseif(kappa==5)
        x1anfang=0.4;
        x1ende=8.1;
    elseif(kappa==2)
        x1anfang=0.3;
        x1ende=8.6;
    elseif(kappa==1)
        x1anfang=0.3;
        x1ende=8.7;
    else
```

```
        x1anfang=0.3;
        x1ende=8.8;
    end;
    x2anfang=-40*sigma;
    x2ende=40*sigma;
    x1schritt = (x1ende-x1anfang)/50;
    x2schritt = (x2ende-x2anfang)/50;
else
    x1anfang=1.0;
    x1ende=4.0;
    if(kappa==2)
        x1anfang=0.8;
        x1ende=4.2;
    elseif(kappa==1)
        x1anfang=0.7;
        x1ende=4.5;
    end;
    x2anfang=-40*sigma;
    x2ende=40*sigma;
    x1schritt = (x1ende-x1anfang)/50;
    x2schritt = (x2ende-x2anfang)/50;
end;


% definition of grid nodes
[X1,X2] = meshgrid(x1anfang:x1schritt:x1ende,...
                   x2anfang:x2schritt:x2ende);
x1 = X1(1,:);
x2 = X2(:,1);
laengex1 = size(x1,2);
```

```
laengex2 = size(x2,1);

% extract indices for plotting
if (plotdomain == 0)
    x1a = 1;
    x1e = laengex1;
    x2a = 1;
    x2e = laengex2;
elseif (plotdomain == 1)
    pdomainx1 = [ 1.1, 3.9];
    pdomainx2 = [-0.3, 0.3];
    x1a = fix((pdomainx1(1) - x1anfang)/x1schritt) + 1;
    x1e = fix((pdomainx1(2) - x1anfang)/x1schritt) + 2;
    x2a = fix((pdomainx2(1) - x2anfang)/x2schritt) + 1;
    x2e = fix((pdomainx2(2) - x2anfang)/x2schritt) + 2;
elseif (plotdomain == 2)
    pdomainx1 = [ 1.1, 3.9];
    pdomainx2 = [-0.1, 0.1];
    x1a = fix((pdomainx1(1) - x1anfang)/x1schritt) + 1;
    x1e = fix((pdomainx1(2) - x1anfang)/x1schritt) + 2;
    x2a = fix((pdomainx2(1) - x2anfang)/x2schritt) + 1;
    x2e = fix((pdomainx2(2) - x2anfang)/x2schritt) + 2;
elseif (plotdomain == 3)
    pdomainx1 = [ 1.1, 3.9];
    pdomainx2 = [-0.02, 0.02];
    x1a = fix((pdomainx1(1) - x1anfang)/x1schritt) + 1;
    x1e = fix((pdomainx1(2) - x1anfang)/x1schritt) + 2;
    x2a = fix((pdomainx2(1) - x2anfang)/x2schritt) + 1;
    x2e = fix((pdomainx2(2) - x2anfang)/x2schritt) + 2;
elseif (plotdomain == 4)
```

```
    pdomainx1 = [ 1.5, 2.5];
    pdomainx2 = [-0.3, 0.3];
    x1a = fix((pdomainx1(1) - x1anfang)/x1schritt) + 1;
    x1e = fix((pdomainx1(2) - x1anfang)/x1schritt) + 2;
    x2a = fix((pdomainx2(1) - x2anfang)/x2schritt) + 1;
    x2e = fix((pdomainx2(2) - x2anfang)/x2schritt) + 2;
elseif (plotdomain == 5)
    pdomainx1 = [ 1.5, 2.5];
    pdomainx2 = [-0.1, 0.1];
    x1a = fix((pdomainx1(1) - x1anfang)/x1schritt) + 1;
    x1e = fix((pdomainx1(2) - x1anfang)/x1schritt) + 2;
    x2a = fix((pdomainx2(1) - x2anfang)/x2schritt) + 1;
    x2e = fix((pdomainx2(2) - x2anfang)/x2schritt) + 2;
elseif (plotdomain == 6)
    pdomainx1 = [ 2.0, 2.1];
    pdomainx2 = [-0.02, 0.02];
    x1a = fix((pdomainx1(1) - x1anfang)/x1schritt) + 1;
    x1e = fix((pdomainx1(2) - x1anfang)/x1schritt) + 2;
    x2a = fix((pdomainx2(1) - x2anfang)/x2schritt) + 1;
    x2e = fix((pdomainx2(2) - x2anfang)/x2schritt) + 2;
end

% loading values from files
if ((kappa==0) & (domain==0))
  Value_pert = load('V_pert0klein.asc');
  Value_fix = load('V_fix0klein.asc');
  Ctrl_pert = load('U_pert0klein.asc');
  Value_exakt = load('V_exakt0klein.asc');
  Ctrl_exakt = load('U_exakt0klein.asc');
elseif ((kappa==0) & (domain==1))
```

```
  Value_pert = load('V_pert0gross.asc');
  Value_fix = load('V_fix0gross.asc');
  Ctrl_pert = load('U_pert0gross.asc');
  Value_exakt = load('V_exakt0gross.asc');
  Ctrl_exakt = load('U_exakt0gross.asc');
elseif ((kappa==1) & (domain==0))
  Value_pert = load('V_pert1klein.asc');
  Value_fix = load('V_fix1klein.asc');
  Ctrl_pert = load('U_pert1klein.asc');
elseif ((kappa==1) & (domain==1))
  Value_pert = load('V_pert1gross.asc');
  Value_fix = load('V_fix1gross.asc');
  Ctrl_pert = load('U_pert1gross.asc');
elseif ((kappa==2) & (domain==0))
  Value_pert = load('V_pert2klein.asc');
  Value_fix = load('V_fix2klein.asc');
  Ctrl_pert = load('U_pert2klein.asc');
elseif ((kappa==2) & (domain==1))
  Value_pert = load('V_pert2gross.asc');
  Value_fix = load('V_fix2gross.asc');
  Ctrl_pert = load('U_pert2gross.asc');
elseif ((kappa==5) & (domain==0))
  Value_pert = load('V_pert5klein.asc');
  Value_fix = load('V_fix5klein.asc');
  Ctrl_pert = load('U_pert5klein.asc');
elseif ((kappa==5) & (domain==1))
  Value_pert = load('V_pert5gross.asc');
  Value_fix = load('V_fix5gross.asc');
  Ctrl_pert = load('U_pert5gross.asc');
elseif ((kappa==10) & (domain==0))
```

```
  Value_pert = load('V_pert10klein.asc');
  Value_fix = load('V_fix10klein.asc');
  Ctrl_pert = load('U_pert10klein.asc');
elseif ((kappa==10) & (domain==1))
  Value_pert = load('V_pert10gross.asc');
  Value_fix = load('V_fix10gross.asc');
  Ctrl_pert = load('U_pert10gross.asc');
elseif ((kappa==15) & (domain==0))
  Value_pert = load('V_pert15klein.asc');
  Value_fix = load('V_fix15klein.asc');
  Ctrl_pert = load('U_pert15klein.asc');
elseif ((kappa==15) & (domain==1))
  Value_pert = load('V_pert15gross.asc');
  Value_fix = load('V_fix15gross.asc');
  Ctrl_pert = load('U_pert15gross.asc');
end;

for i=1:laengex1
    for j=1:laengex2
        V_pert(i,j) = Value_pert((j-1)*laengex1 + i);
        V_fix(i,j) = Value_fix((j-1)*laengex1 + i, 3);
        U_pert(i,j) = Ctrl_pert((j-1)*laengex1 + i);
        if (kappa==0)
            V_exakt(i,j) = Value_exakt((j-1)*laengex1 + i);
            U_exakt(i,j) = Ctrl_exakt((j-1)*laengex1 + i);
        end
    end
end
```

```
% computation of exact solution if kappa=0
if (kappa==0)
    % components of formula for V(x)=B+C*ln(x1)+D*x2
    B= (log((1-beta*alpha)*A)+((beta*alpha)/(1-beta*alpha))...
       *log(beta*alpha*A))/(1-beta);
    C=alpha/(1-alpha*beta);
    D=1/((1-alpha*beta)*(1-rho*beta));

    % computation of V_exakt and U_exakt in equilibrium
    V_exakt_ggw = B +C*log(x1_ggw) +D*x2_ggw;
    U_exakt_ggw =(1-alpha*beta)*A*exp(x2_ggw)*x1_ggw^alpha;
end


% computation of optimal control U_pert by Taylor
% approximation
% in the equilibrium
U_pert_ggw = u_ggw + 0.5*(gss*sigma*sigma);

% computation of V_pert by Taylor approximation
% in the equilibrium
V_pert_ggw = V0 + 0.5*(Vss*sigma*sigma);

% interpolation of V_fix in the equilibrium
V_fix_ggw = interp2(X1,X2,V_fix,x1_ggw,x2_ggw,'*linear');


% computation of approximations error on the nodes
for i=1:laengex1
    for j=1:laengex2
```

```
        if (kappa==0)
            VF_ex_pert(i,j)  = V_exakt(i,j)-V_pert(i,j);
            VF_ex_fix(i,j) = V_exakt(i,j)-V_fix(i,j);
            UF_ex_pert(i,j) = U_exakt(i,j)-U_pert(i,j);
        end;
        VF_pert_fix(i,j)= V_pert(i,j)-V_fix(i,j);
    end
end

% computation of modulus of the error
for i=1:laengex1
    for j=1:laengex2
        if (kappa==0)
            VFa_ex_pert(i,j)  = abs(VF_ex_pert(i,j));
            VFa_ex_fix(i,j) = abs(VF_ex_fix(i,j));
            UFa_ex_pert(i,j) = abs(UF_ex_pert(i,j));
        end;
        VFa_pert_fix(i,j)=abs(VF_pert_fix(i,j));
    end
end


% error plots value functions:

if (kappa==0)

    % exact - fixed-point V
    figure
    surf(x1(x1a):x1schritt:x1(x1e), ...
          x2(x2a):x2schritt:x2(x2e), ...
```

```
          VFa_ex_fix(x2a:x2e,x1a:x1e))
    xlabel('x_1');
    ylabel('x_2');
    title('\bf{Fehler in V\_fix}')
    colorbar

    % exact - perturbation V
    figure
    surf(x1(x1a):x1schritt:x1(x1e), ...
          x2(x2a):x2schritt:x2(x2e), ...
          VFa_ex_pert(x2a:x2e,x1a:x1e))
    xlabel('x_1');
    ylabel('x_2');
    title('\bf{Fehler in V\_pert}')
    colorbar
end;

% perturbation - fixed-point V
figure
surf(x1(x1a):x1schritt:x1(x1e),x2(x2a):x2schritt:x2(x2e), ...
          VFa_pert_fix(x2a:x2e,x1a:x1e))
xlabel('x_1');
ylabel('x_2');
title('\bf{Differenz zwischen V\_pert und V\_fix}')
colorbar

% error plots controls

if (kappa==0)
```

```
    % exact - perturbation u
    figure
    surf(x1(x1a):x1schritt:x1(x1e), ...
          x2(x2a):x2schritt:x2(x2e), ...
          UFa_ex_pert(x2a:x2e,x1a:x1e))
    xlabel('x_1');
    ylabel('x_2');
    title('\bf{Fehler in U\_pert}')
    colorbar
end;


% plots of value functions:

if (kappa==0)
    % exact V
    figure
    surf(x1(x1a):x1schritt:x1(x1e), ...
          x2(x2a):x2schritt:x2(x2e), ...
          V_exakt(x2a:x2e,x1a:x1e))
    xlabel('x_1');
    ylabel('x_2');
    title('\bf{exaktes V}')
end;

% fixed-point V
figure
surf(x1(x1a):x1schritt:x1(x1e),x2(x2a):x2schritt:x2(x2e), ...
          V_fix(x2a:x2e,x1a:x1e))
xlabel('x_1');
```

```
ylabel('x_2');
title('\bf{V\_fix}');

% perturbation V
figure
surf(x1(x1a):x1schritt:x1(x1e),x2(x2a):x2schritt:x2(x2e), ...
          V_pert(x2a:x2e,x1a:x1e))
xlabel('x_1');
ylabel('x_2');
title('\bf{V\_pert}');

if (kappa==0)
    % exakt and fixed-point V
    figure
    surf(x1(x1a):x1schritt:x1(x1e), ...
          x2(x2a):x2schritt:x2(x2e), ...
          V_exakt(x2a:x2e,x1a:x1e))
    hold on
        surf(x1(x1a):x1schritt:x1(x1e), ...
          x2(x2a):x2schritt:x2(x2e), ...
          V_fix(x2a:x2e,x1a:x1e))
    xlabel('x_1');
    ylabel('x_2');
    title('\bf{exaktes V und V\_fix}')

    % exakt and perturbation V
    figure
    surf(x1(x1a):x1schritt:x1(x1e), ...
          x2(x2a):x2schritt:x2(x2e), ...
          V_exakt(x2a:x2e,x1a:x1e))
```

```
    hold on
        surf(x1(x1a):x1schritt:x1(x1e), ...
          x2(x2a):x2schritt:x2(x2e), ...
          V_pert(x2a:x2e,x1a:x1e))
    xlabel('x_1');
    ylabel('x_2');
    title('\bf{exaktes V und V\_pert}')
end;

% fixed-point and perturbation V
figure
surf(x1(x1a):x1schritt:x1(x1e), ...
    x2(x2a):x2schritt:x2(x2e), ...
    V_fix(x2a:x2e,x1a:x1e))
hold on
    surf(x1(x1a):x1schritt:x1(x1e), ...
        x2(x2a):x2schritt:x2(x2e), ...
        V_pert(x2a:x2e,x1a:x1e))
xlabel('x_1');
ylabel('x_2');
title('\bf{V\_pert und V\_fix}');

% plots of controls:

if (kappa==0)
    % exact u
    figure
    surf(x1(x1a):x1schritt:x1(x1e), ...
          x2(x2a):x2schritt:x2(x2e), ...
          U_exakt(x2a:x2e,x1a:x1e))
```

```
    xlabel('x_1');
    ylabel('x_2');
    title('\bf{exaktes u}')
end;

% perturbation u
figure
surf(x1(x1a):x1schritt:x1(x1e),x2(x2a):x2schritt:x2(x2e), ...
          U_pert(x2a:x2e,x1a:x1e))
xlabel('x_1');
ylabel('x_2');
title('\bf{U\_pert}');

if (kappa==0)
    % exact and perturbation u
    figure
    surf(x1(x1a):x1schritt:x1(x1e), ...
          x2(x2a):x2schritt:x2(x2e), ...
          U_exakt(x2a:x2e,x1a:x1e))
    hold on
    surf(x1(x1a):x1schritt:x1(x1e), ...
          x2(x2a):x2schritt:x2(x2e), ...
          U_pert(x2a:x2e,x1a:x1e))
    xlabel('x_1');
    ylabel('x_2');
    title('\bf{exaktes u and U\_pert}')
    hold off;
end;

% error computation
```

```
% error in equilibrium

fprintf('\nFehler im Gleichgewicht (%f, %f):\n', ...
            x1_ggw, x2_ggw );
fprintf('------------------------------\n');
 if (kappa==0)
     fprintf('| V_ex - V_pert |:  %e\n', ...
              abs(V_exakt_ggw-V_pert_ggw));
     fprintf('| V_ex - V_fix  |:  %e\n', ...
              abs(V_exakt_ggw-V_fix_ggw));
 end;
 fprintf('| V_pert - V_fix |:  %e\n', ...
          abs(V_fix_ggw-V_pert_ggw));
 if (kappa==0)
     fprintf('| U_ex - U_pert |:  %e\n', ...
              abs(U_exakt_ggw-U_pert_ggw));
 end;


% error in the domains

nint = size(x1int,1);
VFr_ex_pert = zeros(nint,1);
VFr_ex_fix  = zeros(nint,1);
VFr_pert_fix = zeros(nint,1);
UFr_ex_pert = zeros(nint,1);

for i=1:laengex1
    for j=1:laengex2
```

```
        for k=1:nint
            if ((x1(i)>=x1int(k,1)) & (x1(i)<=x1int(k,2)) ...
                 & (x2(j)>=x2int(k,1)) & (x2(j)<=x2int(k,2)))
                if (kappa==0)
                    VFr_ex_pert(k)  = max(VFr_ex_pert(k), ...
                                          VFa_ex_pert(j,i));
                    VFr_ex_fix(k) = max(VFr_ex_fix(k), ...
                                        VFa_ex_fix(j,i));
                    UFr_ex_pert(k) = max(UFr_ex_pert(k), ...
                                         UFa_ex_pert(j,i));
                end;
                VFr_pert_fix(k) = max(VFr_pert_fix(k), ...
                                      VFa_pert_fix(j,i));
            end;
        end;
    end
end


for k=1:nint
    fprintf('\nFehler auf dem Abschnitt ...
            [%f, %f]x[%f,%f]:\n', ...
            x1int(k,1),x1int(k,2),x2int(k,1),x2int(k,2) );
    fprintf('---------------------------------------\n');
    if (kappa==0)
        fprintf('||V_ex - V_pert||:  %e\n',VFr_ex_pert(k));
        fprintf('||V_ex - V_fix ||:  %e\n',VFr_ex_fix(k));
    end;
    fprintf('||V_pert - V_fix||:  %e\n',VFr_pert_fix(k));
    if (kappa==0)
```

```
        fprintf('||U_ex - U_pert||:  %e\n',UFr_ex_pert(k));
    end;
end;

%error plots value functions

if (kappa==0)
    %perturbation V
    figure
    hold on;
    for k=x1a:laengex1
        for l=x2a:laengex2
            if (VFa_ex_pert(l,k)>0.1)
                plot(x1(k)-(x1schritt/2),x2(l),'s', ...
                     'MarkerEdgeColor','k', ...
                     'MarkerFaceColor','r','MarkerSize',6);
            else
                plot(x1(k)-(x1schritt/2),x2(l),'s', ...
                     'MarkerEdgeColor','k', ...
                     'MarkerFaceColor','g','MarkerSize',6);
            end
        end
    end
    xlabel('x_1');
    ylabel('x_2');
    title('\bf{Fehler in V\_pert kleiner 0.1}')
    hold off;

    %fixed-point V
    figure
```

```
    hold on;
    for k=x1a:laengex1
        for l=x2a:laengex2
            if (VFa_ex_fix(l,k)>0.1)
                plot(x1(k)-(x1schritt/2),x2(l),'s', ...
                     'MarkerEdgeColor','k', ...
                     'MarkerFaceColor','r','MarkerSize',6);
            else
                plot(x1(k)-(x1schritt/2),x2(l),'s', ...
                     'MarkerEdgeColor','k', ...
                     'MarkerFaceColor','g','MarkerSize',6);
            end
        end
    end
    xlabel('x_1');
    ylabel('x_2');
    title('\bf{Fehler in V\_fix kleiner 0.1}')
    hold off;
end;

%difference perturbation - fixed-point V
figure
hold on;
for k=x1a:laengex1
    for l=x2a:laengex2
        if (VFa_pert_fix(l,k)>0.1)
            plot(x1(k)-(x1schritt/2),x2(l),'s', ...
                 'MarkerEdgeColor','k', ...
                 'MarkerFaceColor','r','MarkerSize',6);
        else
```

```
            plot(x1(k)-(x1schritt/2),x2(l),'s', ...
                 'MarkerEdgeColor','k', ...
                 'MarkerFaceColor','g','MarkerSize',6);
        end
    end
end
xlabel('x_1');
ylabel('x_2');
title('\bf{Differenz zwischen V\_pert und V\_fix kleiner 0.1}')
hold off;
```

C Material auf der beiliegenden CD

Die beiliegende CD enthält alle benutzten und angesprochenen Routinen im Verzeichnis Programme, alle in der Arbeit verwendeten Abbildungen im Verzeichnis Bilder, alle zur Auswertung verwendeten Datensätze im Verzeichnis Datensätze, sowie die komplette Arbeit als pdf-Datei und als Latex-Datei im Verzeichnis Diplomarbeit.

Das Verzeichnis Programme

Es sind die folgenden Dateien gespeichert

- `anal_deriv.m`, `num_eval.m`, `gxhx.m`, `gxx_hxx.m` und `gss_hss.m` für die Perturbations-Methode nach SCHMITT-GROHÉ und URIBE.
- `Beispiel.m`, `Beispiel_Zahlen.m`, `Beispiel_run.m` für die Implementierung des Modell-Beispiels und `Beispiel2.m`, `Beispiel2_Zahlen.m` und `Beispiel2_run.m` für die Implementierung des erweiterten Modells.
- `compute_Vcoeff.mw` für die Perturbations-Methode nach COLLARD und JUILLARD.
- `compute.m` zur Speicherung der exakten Lösungen und der Approximationen durch die Perturbations-Methoden.
- `fixed_point.cpp` für die Implementierung der Fixpunkt-Iteration.
- `show51.m` beziehungsweise `show251.m` für die praktischen Auswertungen und zur Erzeugung der Abbildungen.

Das Verzeichnis Bilder

Um die Abbildungen besser zuordnen zu können, haben die Datei-Namen die gleiche Bezeichnung wie die entsprechenden Abbildungsnummern im Text.

Das Verzeichnis Datensätze

Die benutzten Datensätze sind in zwei Dateien gespeichert. Die Datei 51-11 enthält die Auswertungen für 51 Gitterpunkte in x_1- beziehungsweise x_2-Richtung, sowie 11 Gitterpunkte für die Zufallsvariable. Analog befinden sich in 251-51 die Auswertungen für 251 beziehungsweise 51 Gitterpunkte. In jeder dieser Dateien gibt es jeweils eine Datei für die unterschiedlichen σ. Darin sind die Datensätze für die approximativen und exakten Lösungen gespeichert, über deren Namensgebung die folgende Tabelle Aufschluss gibt.

Name der Datensätze	**Inhalt**
U_exakt0klein.asc	exakte Kontrollfunktion für $\kappa = 0$ auf dem Gebiet $[1.0, 4.0] \times [-0.32, 0.32]$
U_exakt0gross.asc	exakte Kontrollfunktion für $\kappa = 0$ auf dem Gebiet $[0.3, 8.8] \times [-0.32, 0.32]$
U_pert2klein.asc	Kontrollfunktion mit Perturbations-Methode für $\kappa = 2$ auf dem Gebiet $[0.8, 4.2] \times [-0.32, 0.32]$
V_exakt0klein.asc	exakte Wertefunktion für $\kappa = 0$ auf dem Gebiet $[1.0, 4.0] \times [-0.32, 0.32]$
V_pert10gross.asc	Wertefunktion mit Perturbations-Methode für $\kappa = 10$ auf dem Gebiet $[0.6, 7.3] \times [-0.32, 0.32]$
V_fix15gross.asc	Wertefunktion mit Fixpunkt-Iteration für $\kappa = 15$ auf dem Gebiet $[1.0, 6.4] \times [-0.32, 0.32]$

Literaturverzeichnis

[1] K. J. ÅSTRÖM, *Introduction to Stochastic Control Theory*, in R. Bellmann, Hrsg., *Mathematics in Science and Engineering, Volume 70*, Academic Press, 1970.

[2] S. BECKER, *Numerische Lösung dynamischer Gleichgewichtsmodelle*, Diplomarbeit an der Universität Bayreuth, betreut durch L. Grüne, Oktober 2005.

[3] S. BECKER, L. GRÜNE UND W. SEMMLER, *Comparing Accuracy of Second Order Approximation and Dynamic Programming*, Preprint, Universität Bayreuth, June 2006, Submitted, http://www.math.uni-bayreuth.de/ lgruene/publ/, Stand Dezember 2006.

[4] R. BELLMANN, *Introduction to the Mathematical Theory of Control Processes, Volume I*, in R. Bellmann, Hrsg., *Mathematics in Science and Engineering, Volume 40*, Academic Press, 1967.

[5] M. CAPIŃSKI UND E. KOPP, *Measure, Integral and Probability*, 2. Auflage, Springer-Verlag, 2005.

[6] F. COLLARD UND M. JUILLARD, *Accuracy of stochastic perturbation methods: The case of asset pricing models*, Journal of Economic Dynamics and Control 25, pp. 979-999, 2001.

[7] F. COLLARD UND M. JUILLARD, *A Higher-Order Taylor Expansion Approach to Simulation of Stochastic Forward-Looking Models with an Application to a Nonlinear Phillips Curve Model*, Computational Economics 17, pp. 125-139, 2001.

[8] J. ELSTRODT, *Maß- und Integrationstheorie*, 3. Auflage, Springer-Verlag, 2002.

[9] G. FEICHTINGER UND R. F. HARTL, *Optimale Kontrolle ökonomischer Prozesse*, de Gruyter, 1986.

[10] L. GRÜNE, *Numerische Mathematik 1*, Ausarbeitung einer Vorlesung, gehalten an der Universität Bayreuth im Wintersemester 2002/2003.

[11] L. GRÜNE, *Numerik dynamischer Systeme*, Ausarbeitung einer Vorlesung, gehalten an der Universität Bayreuth im Wintersemester 2004/2005.

[12] L. GRÜNE, *Numerische Dynamik von Kontrollsystemen*, Ausarbeitung einer Vorlesung, gehalten an der Universität Bayreuth im Sommersemester 2005.

[13] L. GRÜNE, *Error estimation and adaptive discretization for the discrete stochastic Hamilton-Jacobi-Bellman equation*, Numerische Mathematik 99, pp.85-112, Springer-Verlag 2004.

[14] H. JIN UND K. L. JUDD, *Perturbation methods for general dynamic stochastic models*, Working Paper, Stanford University, 2002, `http://bucky.stanford.edu/defaultmain.htm`, Stand Januar 2007.

[15] K. L. JUDD, *Approximation, perturbation, and projection methods in economic analyis*, in H. Amman, D. Kendrick, und J. Rust, Hrsg., *Handbook of Computational Economics*, Amsterdam: Elsevier, 1996.

[16] K. L. JUDD, *Numerical Methods in Economics*, The MIT Press, 1998.

[17] F. E. KYDLAND UND E. C. PRESCOTT, *Time to build and aggregate fluctations*, Econometrica 50, pp. 1345-1370, 1982.

[18] A. MAS-COLELL, *The Theory of General Economic Equilibrium - A Differentiable Approach*, Econometric Society Monograph, Cambridge University Press, 1989.

[19] M. MORLOCK UND K. NEUMANN, *Operations Research*, 2. Auflage, Hanser Verlag, 2002.

[20] J. A. MURDOCK, *Perturbations, Theory and Methods*, John Wiley & Sons, 1991.

[21] R. NECK, *Stochastic Control Theory and Operational Research*, European Journal of Operational Research 17, pp. 283-301, 1984.

[22] P. OBERENDER, *Grundbegriffe der Mikroökonomie*, Verlag P.C.O., Bayreuth, 1999.

[23] F. VAN DER PLOEG, Hrsg., *Mathematical Methods in Economics*, John Wiley & Sons, 1986.

[24] S. SCHMITT-GROHÉ UND M. URIBE, *Solving dynamic general equilibrium models using a second-order approximation to the policy function*, Journal of Economic Dynamics and Control 28, pp. 755-775, 2004.

[25] S. SCHMITT-GROHÉ, *Perturbation Methods for the Numerical Analysis of DSGE Models*, Lecture Notes, preliminary draft, Duke University, 2005, `www.econ.duke.edu/~grohe/teaching/perturbationmethods.htm`, Stand Januar 2007.

[26] S. SCHMITT-GROHÉ UND M. URIBE, *Matlab Codes for Second Order Approximation*, `http://www.econ.duke.edu/~uribe/2nd_order.htm`, Stand November 2006.

[27] L. TESAR, *Evaluating the Gains from International Risksharing*, Carnegie-Rochester Conference Series on Public Policy 42, pp. 95-143, 1995; 5, pp. 43-78, 1995.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Diese Arbeit hat in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Bayreuth, den 17. Januar 2007 .....

Stefanie Öhrlein