Fakultät für Mathematik und Physik

Universität Bayreuth
Professur für Angewandte Mathematik
Prof. Dr. Lars Grüne

Stochastische dynamische Portfolio-Optimierung

auf unendlichem Zeithorizont

Diplomarbeit

von

Maik Müller

Datum: 30. März 2009

Aufgabenstellung und Betreuung:
Prof. Dr. Lars Grüne

Danksagung

An dieser Stelle möchte ich mich bei Professor Grüne für die ausgezeichnete Betreuung während der Arbeit, die interessante Aufgabenstellung und die Heranführung an diesen Themenbereich durch seine Vorlesung und sein Seminar *Stochastische Dynamische Optimierung* bedanken. Wann immer ich Fragen hatte, hat er sich Zeit genommen.

Ebenso danke ich Professor Lempio für seine kritische Auseinandersetzung mit meinem Optimierungsalgorithmus.

Bedanken möchte ich hiermit auch bei meiner Familie, die mich während meines gesamten Studium unterstützt hat, und meiner Freundin, die mir immer Rückhalt gegeben hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Eine interessante Aufgabe in der Finanzwirtschaft ist das Lösen des Portfoliooptimierungsproblems. Dieses behandelt die Frage, wie ein Investor sein Vermögen so auf verschiedenste Anlageformen verteilt, dass ein für ihn maximaler Gewinn erwirtschaftet wird. Ein Portfolio ist dabei ein Mix aus Anlagen, wie z.B. Aktien, Wertpapiere, Immobilien, Rohstoffe und auch Bargeld, welche unterschiedlichste Renditen und Risiken besitzen. Im Verlauf der Arbeit wird gezeigt, dass das *optimale Portfolio* ein sehr subjektiver Begriff ist, da dieses von der Risikoneigung des Investors abhängt.

In [14] Caroline Öhrlein, 2006, wurde das Portfolioproblem mit Hilfe der dynamischen Programmierung untersucht. Dem dritten Modell aus [14] wurden zwei Vermögenswerte mit zeitlich variierenden Zinssätzen, welche mit Sinus-Funktionen modelliert wurden, zugrundegelegt. In meinem Seminarvortrag mit dem Thema *Portfolio-Optimierung auf unendlichem Horizont* im Rahmen des Seminars *Stochastische Dynamische Optimierung* habe ich mich mit den mathematischen und portfoliotheoretischen Grundlagen sowie den Ergebnissen zu diesem Modell beschäftigt. Ziel dieser darauf aufbauenden Diplomarbeit ist es, die Ansätze des Modells aus [14] aufzugreifen und sie um einen stochastischen Einfluss zu erweitern. Weiterhin ist dessen mathematische und algorithmische Umsetzung der Schwerpunkt dieser Arbeit.
Anstatt die Zinsen mit Sinus-Funktionen unterschiedlicher Amplitude und Periodendauer zu modellieren, werde ich reale Kursverläufe (Schweizer Aktien- und Obligationenindex sowie Standard&Poor's Indizes) und mittels Zufallszahlen erzeugte miteinander korrelierte Zinsdaten verwenden, um die Zinsen realitätsnah zu modellieren.
Der praktische Teil dieser Arbeit soll die Frage klären, wie sich die numerische Lösung des Modells bei diesen Zinsdaten und unter der Variation der Anlagemöglichkeiten, d.h. ob und wieviel Leerverkäufe sind zulässig, im deterministischen Fall verhält. Wie sich ein stochastischer Einfluss auswirkt, wird danach für zwei verschiedene Zinsverläufe untersucht.

Kapitel 2 soll eine Einführung in die Portfolio-Theorie aus wirtschaftswissenschaftlicher Sicht geben. Dazu wird nach einem kurzen Abriss der Finanzwirtschaft, die einen Überblick über die Märkte, Instrumente und Marktteilnehmer gibt, das Portfolio-Management vorgestellt. Dabei soll das traditionelle vom modernen Portfolio-Management abgegrenzt und zwei Modelle eingeführt werden. Das erste, das Markowitz-Modell, liefert die Herleitung eines effizienten und optimalen Portfolios unter Berücksichtigung des Rendite-Risiko-Verhältnisses sowie Nutzens, wogegen das zweite, das Capital Asset Pricing Modell, auf einem Gleichgewichtsmarkt den Zusammenhang zwischen systematisch eingegangenem Risiko und zusätzlicher Rendite untersucht.

Das dritte Kapitel legt die mathematischen Grundlagen zur Lösung einen Portfoliooptimierungsproblems mit der stochastischen dynamischen Programmierung. Dazu werden im ersten Abschnitt nach einem kurzen Überblick über wichtige verwendete stochastische Begriffe das stochastische Kontrollsystem und stochastische dynamische Optimierungsproblem definiert und die Frage geklärt, wann ein Kontrollprozess zulässig ist. Die Formulierung und der Beweis des Bellman'schen Optimalitätsprinzip auf unendlichem Zeithorizont für stochastische dynamische Optimierungsprobleme bilden den Abschluss dieses Abschnitts.
Die numerische Umsetzung des Prinzips ist Schwerpunkt des zweiten Abschnitts. Nach kurzer Vorstellung des stochastischen Eulerverfahren für die Diskretisierung in der Zeit wird die räumliche, welche die optimale Wertefunktion auf Gittern approximiert, ausführlich beschrieben.

In Kapitel 4 werden die beiden Modelle beschrieben, das deterministische und das stochastische. Im zweiten wird genauer darauf eingegangen, wie der stochastische Einfluss modelliert werden kann. Dazu wird der Wiener Prozess sowie dessen schwache Approximation definiert und dessen algorithmische Umsetzung im stochastischen Eulerverfahren vorgestellt.
Zum Abschluss wird noch die Konkavität der optimalen Wertefunktion in x_0 (Vermögen; die Komponente x_1 beschreibt die Zeit) bewiesen.

Die verwendeten Programme werden im fünften Kapitel beschrieben. Im ersten Abschnitt werden das Hauptprogramm *Portfolio.c*, welches das stochastisch dynamische Optimierungsproblem löst, und der von Professor Grüne entwickelte Gittergenerator *gridgen.c*, welcher die nötigen Gitter für die Diskretisierung im Raum liefert, beschrieben.
Programme für die Erzeugung von zufälligen Zinsen und für die graphische Darstellung und numerische Auswertung der Ergebnisse des Hauptprogramms werden im zweiten Abschnitt vorgestellt.

In Kapitel 6 werden schließlich die numerischen Ergebnisse zu verschiedenen Tests mit unterschiedlichen Parameterkombinationen zu den beiden in Kapitel 4 vorge-

stellten Modellen ausgewertet.
Nach der Festlegung, welche Parameter dabei als Standard gelten sollen, werden im zweiten Abschnitt Vorbetrachtungen zu der Wahl der Parameter des Optimierungsalgorithmus für die optimale Kontrolle an einem Punkt x sowie zu der Wahl des Parameters des maximalen Vermögen (Gittergröße in der Vermögenskomponente) gemacht.
Danach werden im dritten Abschnitt verschiedene Kontrollwertebereiche für das deterministische Modell und im vierten zusätzlich noch die Auswirkungen unterschiedlicher Gewichtungen des stochastischen Einflusses untersucht.

Wichtige Formeln erhalten Formelnummern, die kapitelweise hochgezählt werden, Literaturverweise bestehen aus der Zahl der Quelle aus dem Literaturverzeichnis in eckigen Klammern, dem Autornachnamen, Jahreszahl sowie, falls erforderlich, der Angabe der relevanten Seite(n). Die Fußnotennummerierung beginnt in jedem Kapitel mit der 1.

Kapitel 2

Einführung in die Portfolio-Theorie

Dieses Kapitel soll einen Überblick über die Portfoliotheorie geben um ein grundlegendes Verständnis für die im 4. Kapitel vorgestellten Modelle zu liefern.
Im Abschnitt 2.1, der Grundlagen der Finanzwirtschaft, werden verschiedene Möglichkeiten der Gliederung der Finanzmärkte, die unterschiedlichen Finanzinstrumente und die Rolle der Marktteilnehmer betrachtet.
Im Abschnitt 2.2 wird eine Einführung in die traditionelle und moderne Portfoliotheorie gegeben. Dabei soll zunächst das grundsätzliche Vorgehen beim traditionellen Portfoliomanagement erläutert und danach die Erweiterung dessen, das moderne Portfoliomanagement, vorgestellt werden.
Der Abschnitt 2.3 beschäftigt sich mit dem Markowitz- und Capital Asset Pricing Modell, wobei auf die Gemeinsamkeiten und Unterschiede sowie Vor- und Nachteile eingegangen wird.

2.1 Grundlagen der Finanzwirtschaft

In diesem Abschnitt, welcher auf [2] Auckenthaler, 1994, S.9-60 aufbaut, soll ein Überblick über die wesentlichen Begriffe der Finanzwirtschaft gegeben werden.

2.1.1 Gliederung der Finanzmärkte

Man kann die Finanzmärkte auf verschiedene Art und Weise unterteilen.

Eine Möglichkeit ist die instrumentale Gliederung.
Auf der einen Seite gibt es den Basismarkt, auf dem die Geschäfte direkt nach Abschluss oder auf Termin gemacht werden. Man unterteilt diesen noch weiter in den Geldmarkt, der dem Ausgleich von Liquidität dient, und den Kapitalmarkt, der der mittel- und längerfristigen Kapitalbeschaffung dient.
Auf der anderen Seite gibt es den Markt für Derivate, bei dem der Abschluss und

die Erfüllung immer zeitlich auseinander liegen. Er ist entstanden, um den Risiken bei der Kapitalaufnahme und -anlage entgegenzutreten.

Eine zweite Möglichkeit der Gliederung ist die funktionale.
Man unterscheidet hier den Primärmarkt, der der Ausgabe von Finanzinstrumenten wie zum Beispiel Aktien oder Anleihen dient, und den Sekundärmarkt, auf dem die zuvor am Primärmarkt ausgegebenen Titel gehandelt werden können. Dabei unterscheidet man den Handel standardisierter Titel an einer Börse (Aktien, Wertpapiere) und den nicht standardisierter Finanzinstrumente „Over the Counter "(am Bankschalter).

Weitere Möglichkeiten der Gliederung sind die nach Inlands- und Auslandsmarkt sowie die nach Organisation, bei der man den freien Markt, Vermittlermarkt, Dealermarkt sowie Auktionsmarkt unterscheidet.

2.1.2 Überblick über die wichtigsten Finanzinstrumente

Sämtlichen Finanzinstrumenten gemein ist die Aufgabe, den monetären Transfer zwischen Kapitalanbietern und Kapitalnachfragern zu erleichtern. Des Weiteren sollen sie die Übertragung und Kontrolle der Risiken ermöglichen.
Finanzinstrumente kurzer Laufzeit sind die Geldmarktinstrumente, die auf einem relativ homogenen Markt angeboten werden. Unterschieden werden dabei Geldmarktinstrumente der öffentlichen Hand und von privatwirtschaftlichen Unternehmen.
Die Kapitalmarktinstrumente gliedert man in fremdkapitalbezogene und eigenkapitalbezogene. Erstere unterscheidet man weiter in der Zinsgestaltung (mit Zinscoupon, Zero Bonds), Tilgung (fixierte, variable Tilgung, mit Option für die Emittenten oder den Investor), Laufzeit (Anleihen mit festen Laufzeiten oder geregelter vorzeitiger Rückzahlung, Anleihen mit Optionsrechten für den Emittenten oder Investor), Emissionspreis, Sicherstellung und Art der Märkte (nationale oder internationale).
Zu den eigenkapitalbezogenen Kapitalmarktinstrumenten gehören die Aktien, die man in Stammaktien, welche Stimm- und Vermögensrechte beinhalten und zu denen die Inhaberaktien und Namensaktien zählen, und Vorzugsaktien, welche Vorteile gegenüber den Stammaktien bei Gewinnbeteiligung, Stimm- und Liquidationsrechten haben, unterteilt, und die Fondszertifikate. Nachteil der Fondszertifikate ist das fehlende Mitspracherecht, aber deren Vorteile sind die Diversifikation und das damit verringerte Risiko, das professionelle Portfolio-Management und dass keine Nachteile durch kleine Investitionsbeträge entstehen.
Des Weiteren gibt es noch Finanzderivate, zu denen Financial Futures, Optionen, Swaps und Structured Assets zählen.

2.1.3 Marktteilnehmer und deren Rolle im Finanzsystem

Zu den Nachfragern im Markt gehören die Haushalte, der Staat und die Unternehmen.
Anbieter sind verschiedene Finanzinstitutionen (sogenannte Finanzintermediäre) wie Kommerz- und Investmentbanken, Factoring-, Beteiligungs-, Forfaitierungs-, Finanz- und Leasinggesellschaften sowie Venture Capital Gesellschaften.
Zu den Aufgaben der Anbieter im Finanzsystem gehören die Fristen- und Risikotransformation sowie die Transformation von Informationen.

2.2 Portfolio-Management

In diesem Abschnitt sollen die Grundzüge des traditionellen und modernen Portfolio-Managements erläutert und die portfoliotheoretischen Grundlagen für die praktischen Experimente an den Modellen in Kapitel 4 gelegt werden.
Es werden aktive und passive Portfolio-Management-Techniken unterschieden, wobei die passiven im modernen Portfolio-Management beim Capital Asset Pricing Modell und die aktiven im traditionellen Portfolio-Management sowie im Markowitz-Modell angewendet werden.

2.2.1 Drei Schritte des traditionellen Portfolio-Management

Die drei Schritte des traditionellen Portfolio-Managements (siehe [2] Auckenthaler, 1994, S.63-108 sowie S. 271-273) sind erstens der Vorbereitungsschritt, der die Finanzanalyse beinhaltet, zweitens die Erarbeitung des Anlagekonzepts und drittens die Portfoliobildung und das Portfolio-Management.
Für eine schematische Darstellung des Ablaufprozesses im Portfolio-Management siehe Abb. 2.1[1].

2.2.1.1 Die Finanzanalyse

Die Finanzanalyse dient dazu, die verschiedenen Anlageformen auf ihre Rendite- und Risikoeigenschaften zu untersuchen.
In der Herangehensweise unterscheidet man Aktien und festverzinsliche Anlagen.
Für die Aktienanalyse gibt es im traditionellen Portfolio-Management die Fundamentalanalyse und die technische Analyse.

Fundamentalanalyse

Mit Hilfe der Fundamentalanalyse versucht man die Aktienkursentwicklung anhand von Marktdaten zu prognostizieren. Dabei bedient man sich der Globalanalyse, bei

[1] in Anlehnung an [2] Auckenthaler, 1994, S.64

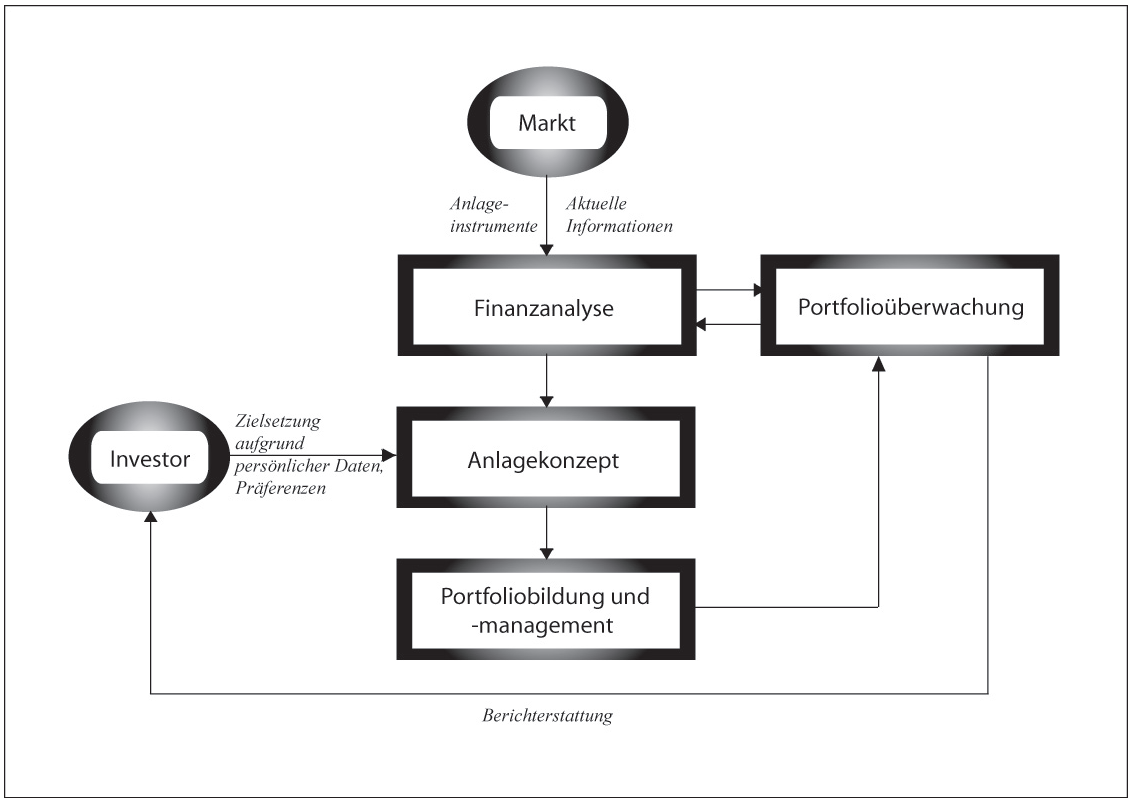

Abbildung 2.1: Ablaufprozess des Portfolio-Managements

der man die politische und volkswirtschaftliche Lage sowie markttechnische Einflussfaktoren, das heißt die gegenseitige Abhängigkeit von Aktienbörsen, analysiert. Weiter bedient man sich der Branchenanalyse und zum Schluss der Unternehmensanalyse, bei der man sich der qualitativen (Managementqualitäten, Ruf und Marktstellung des Unternehmens, Produktbesonderheiten, Forschung und Entwicklung etc.) und der quantitativen (Geschäftsbericht bestehend aus Jahresabschluss sowie Angaben zur Gesellschaft und Erläuterungen zur Geschäftstätigkeit) Analyse bedient. Vorteil der Fundamentalanalyse ist, dass eine Aktie unter Berücksichtigung des gesamten Umfelds eines Unternehmens bewertet wird. Nachteile sind, dass man nicht alle Faktoren, die den Kurs einer Aktie beeinflussen, genau genug bestimmen kann und dass diese Analyse subjektiv ist, da es verschiedene Analystenmeinungen geben kann. Ein weiterer Nachteil ist die problematische Datenbeschaffung, da eine starke Abhängigkeit von der Qualität und Korrektheit der veröffentlichten Daten herrscht. Auch bei korrekten Daten kann es passieren, dass diese schon im Markt eingepreist sind oder dass die Märkte nicht entsprechend der Fundamentalanalyse reagieren (z.B. aufgrund psychologischer Einflüsse).

Technische Analyse

Die technische Analyse steht im Gegensatz zur Random-Walk-Hypothese[2] , welche besagt, dass sich Aktienkurse zufällig verändern. Der Analyst versucht hier mithilfe verschiedener Chartanalysemethoden wie zum Beispiel „Gleitender Durchschnitt“ oder „Kurs/Umsatzanalyse“ die Aktienkursentwicklung vorauszusagen. Vorteil der technischen Analyse ist, dass man kein Problem damit hat, ob bestimmte Informationen über ein Unternehmen korrekt sind oder nicht, da man annimmt, dass sich sämtliche Informationen eines Unternehmens im Börsenkurs widerspiegeln. Dadurch kann man auch emotionale und psychologische Einflüsse in die Analyse mit einbeziehen. Nachteil ist, dass Kaufs- und Verkaufsentscheidungen erst getätigt werden, wenn eine Tendenz aus den Charts ablesbar ist, d.h. man handelt einen Schritt später als jemand, der das Unternehmen auf fundamentaler Ebene beurteilt. Außerdem besteht das Problem, dass in unruhigen Börsenzeiten keine eindeutigen Trends ausgemacht werden können, was zu Fehlallokationen führt. Noch schwerer wiegt der Fakt, dass es ökonomisch wie statistisch fragwürdig ist, dass man ein zukünftiges Kursverhalten aus der Vergangenheit ableiten kann, was eine Abhängigkeit aufeinanderfolgender Kursänderungen voraussetzt. Dies widerspricht aber der Random-Walk-Hypothese.

Beurteilung festverzinslicher Anlagen

Für die Beurteilung von festverzinslichen Anlageinstrumenten zieht man die Kriterien Schuldnerqualität, Laufzeit, Zinssatz, Rendite und Marktgängigkeit heran.
Die Schuldnerqualität gibt Auskunft darüber, ob ein Schuldner fähig ist alle Zins- und Rückzahlungen zu leisten. Kriterien hierfür sind Liquidität, Finanzierungskraft, Ertragskraft und Kapital-/Vermögensstruktur sowie Managementfähigkeiten, Produkte und Marktstellung. Professionelle Analysen hierzu liefern Ratingagenturen wie zum Beispiel Fitch Ratings, Moody's und Standard & Poor's.
Bei der Laufzeit unterscheidet man nach Art und Weise der Tilgung und Zinszahlung sowie ob es ein vorzeitiges Kündigungsrecht gibt.
Bei der Beurteilung des Zinssatzes geht man von einem risikolosen Zinssatz aus, der der Lohn für den Verzicht auf sofortigen Konsum ist, und schlägt darauf die erwartete Inflationsrate um das Realkapital zu erhalten. Weitere Aufschläge auf den Zins ergeben sich durch die Risikoprämie, deren Höhe abhängig von der Schuldnerqualität gewählt wird.
Die Rendite einer Anleihe bestimmt sich aus den Komponenten Zinssatz, Laufzeit und Kurs.
Die Marktgängigkeit gibt an, wie leicht man festverzinsliche Wertpapiere kaufen beziehungsweise verkaufen kann. Beeinflusst wird sie durch die Größe der

[2]Diese besagt, dass sich Aktienkurse zufällig verändern. Die beste Schätzung für den zukünftigen Aktienkurs X_{t+1} erfolgt auf Basis des heutigen Aktienkurs X_t: $X_{t+1} = X_t + \xi_t$ mit ξ_t eine Zufallsvariable und $\mathbb{E}(\xi_t) = 0 \,\forall t$ sowie $Cov(\xi_t, \xi_{t+1}) = 0 \,\forall t$, wobei die Kursveränderungen so sein müssen, dass deren Varianz beschränkt ist.

Anleiheemission, den Marktbedingungen und dem Substitutionsgrad der Anleihe.

2.2.1.2 Das Anlagekonzept

Allein die Finanzanalyse reicht nicht aus um ein Portfolio zusammenzustellen. Deshalb muss der Portfolio-Manager im Rahmen des Anlagekonzeptes die Ziele des Investors und die Anlagevorschriften des Investors berücksichtigen sowie die Anlagepolitik festlegen.
Zu den Investorenzielen gehören Rentabilität, Sicherheit und Liquidität. Die Gewichtung der beiden Teilziele der Rentabilität, der Kapitalzuwachs und die laufenden Erträge, bestimmt, ob ein Investor Aktien oder festverzinsliche Wertpapiere bevorzugt. Die Kaufentscheidung zu Gunsten der Aktie fällt aufgrund des erwarteten Kapitalzuwachses und die zu Gunsten des Wertpapieres aufgrund der laufenden Erträge. Die Dividente der Aktie bzw. die Kursschwankung des Wertpapieres spielt beim Kauf dagegen eine untergeordnete Rolle. Bei der Sicherheit ist von Bedeutung, dass jede Chance auf eine höhere Rendite auch ein Risiko beinhaltet. Im Rahmen der Risikotoleranz unterscheidet man risikofreudiges, risikoindifferentes und risikoaverses Verhalten. Zur Beurteilung der Liquidität muss man die Faktoren Fristigkeit, d.h. die Laufzeit einer Anlage, und Abtretbarkeit, d.h. ob es für die Anlage einen Markt gibt um sie zu Geld zu machen, beachten.
Außer den Investorenzielen müssen noch die Anlagevorschriften beachtet werden. Dazu gehören finanzielle Faktoren (Anlagevolumen, Liquidität, Anlagehorizont), gesetzliche Rahmenbedingungen (Besteuerung von Erträgen und Kapitalgewinnen) und persönliche Wünsche.
Bei der Anlagepolitik unterscheidet man den Bottom-Up- und Top-Down-Ansatz. Der Bottom-Up-Ansatz steht für die Auswahl der Wertpapiere aufgrund fundamentalanalytischer Erkenntnisse. Es werden Titel gesucht, die unterbewertet sind, dabei spielen die gegenwärtige und zukünftige Marktlage und die Herkunft der Titel keine Rolle. Ergebnis ist ein unsystematisch zusammengestelltes Portfolio, bei dem das Risiko keine Beachtung findet. Bei dem Top-Down-Ansatz wird zunächst strategisch abgewogen, welche Anlagemedien sich für welches Investorenziel eignen. Verzinsliche Anlagen eignen sich zum Beispiel für das Ziel Zinsertrag, wogegen ein Wertzuwachs mittels Aktien erreicht werden soll. Strategisch von Bedeutung ist ebenso, in welchen Ländern bzw. Märkten und Fremdwährungen angelegt wird. Bei der taktischen Vorgehensweise des Top-Down-Ansatzes werden die Branchen bestimmt und deren Gewichtung festgelegt, dann werden die einzelnen Titel mit ihren Kaufs- und Verkaufszeitpunkten ermittelt. Dies muss regelmäßig überprüft werden. Ergebnis des Top-Down-Ansatzes ist ein systematisch zusammengestelltes Portfolio, welches den Grundsatz der Diversifikation erfüllt und daher risikobewusster ist sowie die Anlagevorschriften des Investors beachtet.

2.2.1.3 Portfoliobildung und Portfoliomanagement

Im traditionellen Portfolio-Management werden aktive Management-Techniken angewendet, d.h. es wird unterstellt, dass die Schwankungen der Wertpapierkurse tendentiell vorhersehbar und dass die Kapitalmärkte ineffizient sind. Daher legt man den Schwerpunkt auf höhere Renditen, was auch ein höheres Risiko impliziert, welches aber durch die tendenzielle Vorhersehbarkeit der Kurse in Kauf genommen werden kann. Bei den grundlegenden Management-Techniken kann zwar wieder in welche für Aktien und welche für verzinsliche Wertpapiere unterschieden werden, aber da diese vom Prinzip her ähnlich funktionieren, gehe ich nur auf die für die Aktien ein.
Als Ursachen für höhere Renditen kann man die Marktverfassung, die Charakteristiken der einzelnen Aktien und den unterschiedlichen Zustand verschiedener Marktsektoren bzw. Titelgruppen ausmachen. Daraus leitet man dann die Techniken Timing, Selektion und Gruppenrotation zum Optimieren der Renditen ab.
Beim Timing kommt es darauf an die Kaufs- und Verkaufszeitpunkte optimal zu legen. Dazu muss der Manager den Markt genau analysieren. Nutzen kann er dabei die Trend-Methode, bei der der Investor bei optimistischer Börse (Gewinne laufen lassen) kauft und bei pessimistischer (Verluste begrenzen) verkauft, d.h. er verbleibt im Markt, auch wenn andere wegen einer hohen Bewertung der Aktien bereits verkauft haben, und er geht nicht in den Markt, wenn Aktien billig sind, sondern erst, wenn sich ein Aufwärtstrend etabliert hat. Dagegen wird bei der Methode der Contrary Opinion nach dem Prinzip „Kaufe niedrig, verkaufe hoch“ vorgegangen, d.h. man kauft Aktien, die übermäßig gesunken sind und verkauft welche, die stark gestiegen sind, wobei der Grund der niedrigen bzw. hohen Bewertung der Aktien bekannt sein muss. Dies entspricht einem antizyklischen Verhalten des Investors.
Bei der Selektion werden die Kaufs- und Verkaufsentscheidungen aufgrund der Ergebnisse der Fundamentalanalyse gefällt. Wenn der innere Wert einer Aktie über dem gegenwärtigen Kurs liegt, dann ist diese attraktiver, d.h. man kauft sie für das Portfolio. Unattraktive Aktien werden verkauft.
Bei der Gruppenrotation dagegen betrachtet man nicht mehr einzelne Aktien sondern ganze Marktsektoren. Es werden Branchensektoren und Gruppen von Aktientypen gebildet wie z.B. Wachstumsaktien, zyklische Aktien, stabile Aktien und Aktien des Energiesektors.
Wenn man alle drei Techniken anwendet, kann man im traditionellen Portfolio-Management eine höchst mögliche Rendite erreichen, geht aber auch ein höheres Risiko ein.

2.2.1.4 Die Portfolioüberwachung

Drei Gründe sprechen für die Praktizierung einer Portfolioüberwachung: Marktveränderungen, Veränderungen des Anlagekapitals sowie Veränderungen der Ziele und Anlagevorschriften des Investors. Außerdem ist es wichtig die Kapitalströme zu

überwachen, d.h. wenn der Investor Kapital abzieht oder hinzufügt und wenn die anfallenden Zinszahlungen wieder investiert werden sollen, dann ist eventuell eine Portfolioumschichtung notwendig. Dabei ist zu beachten, dass bei einer Portfoliorevision Kosten entstehen, sei es durch Transaktionskosten wie Steuern, Courtagen oder Börsengebühren oder durch Analysekosten. Eine Umschichtung des Portfolios ist nur sinnvoll, falls sich der Nutzen des Portfolios unter Berücksichtigung aller Kosten erhöht.

2.2.2 Modernes Portfolio-Management

Das im Abschnitt 2.2.1 erläuterte traditionelle Portfolio-Management zielt darauf ab die einzelnen Anlagemedien qualitativ zu beurteilen, d.h. der Investor möchte ein Portfolio mit möglichst hoher Rendite zusammenstellen. Das moderne Portfolio-Management (siehe [2] Auckenthaler, 1994, S.117-146 und [1] Auckenthaler, 2001, S.9-89) ergänzt das traditionelle, indem nun das Risiko-Rendite-Verhältnis einer Anlage bzw. eines Portfolios im Vordergrund steht. Zu den wichtigen Begriffen des modernen Portfolio-Managements gehören Nutzen, Risiko und Diversifikation.

2.2.2.1 Definition des Nutzens

Die Rendite, die man basierend auf historischen Daten oder auf Szenarien prognostizieren kann, ist in erster Linie das Anlageziel des Investors. Dem steht das Sicherheitsziel entgegen. Um die beiden Ziele Rendite und Sicherheit in Einklang zu bringen, definiert man sich eine Nutzenfunktion, welche von Investor zu Investor verschieden ist. Ein rational handelnder Investor wird immer den Nutzen maximieren.
Eine Möglichkeit den Nutzen zu definieren ist die Power-Nutzenfunktion[3], welche gegeben ist durch

$$g(x) = \frac{x^{1-\gamma} - 1}{1 - \gamma} \tag{2.1}$$

Dabei ist γ die relative Riskoaversion, für die gilt: $RRA(x) = -x \cdot \frac{g''(x)}{g'(x)}$. Für $\gamma \to 1$ gilt dann $g(x) = ln(x)$.
Der Verlauf der Power-Nutzen-Funktion ist dabei ein Indikator für die Risikoneigung. Ist sie konkav, so ist der Investor risikoscheu, konvex dann risikofreudig und linear dann risikoneutral. Zur graphischen Veranschaulichung siehe Abb. 2.2[4], wobei x für den Input (Rendite, Konsum) und $g(x)$ für den Nutzen des Inputs steht.
Eine konkave Nutzenfunktion modelliert dabei das Prinzip des abnehmenden Grenznutzens, d.h. es tritt ein „Sättigungseffekt“ ein (für $x \to \infty$ gilt $g'(x) \to 0$).

[3] siehe [15] Porembski, 2006, S.81-83
[4] in Anlehnung an [2] Auckenthaler, 1994, S.125

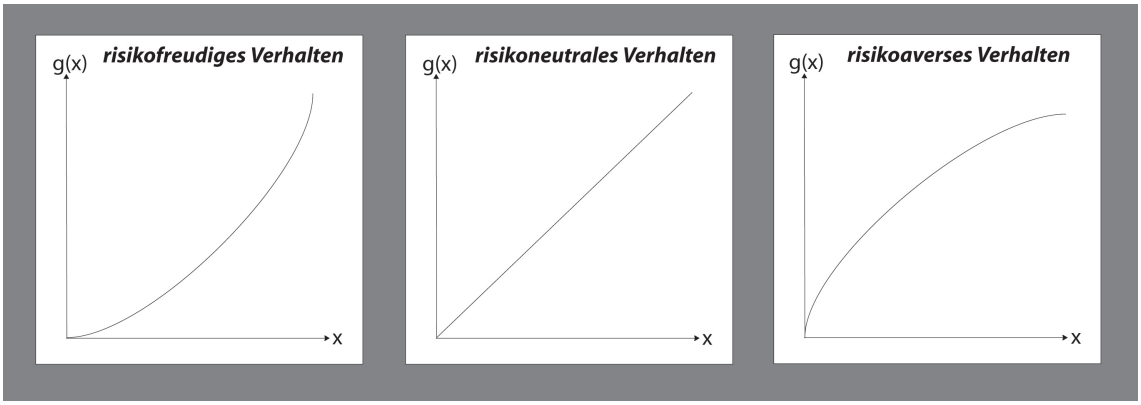

Abbildung 2.2: Verschiedene Nutzenfunktionen im Vergleich

2.2.2.2 Definition des Risikos

Wie am Anfang des Abschnittes 2.2.2 erwähnt, spielt das Risiko eine wichtige Rolle im modernen Portfolio-Management. Die Risiken werden in Firmen- (z.B. Bonität), Branchen- (Konkurrenzdruck, Regulierungsmaßnahmen) und Marktrisiken (Konjunktur) unterteilt.
Verbal kann man es wie folgt beschreiben:

> „Risiko wird somit als Gefahr verstanden, eine erwartete Rendite zu verfehlen.“[5]

Um das Risiko auch analytisch zu quantifizieren, nutzt man die Standardabweichung bzw. die Varianz als Risikomaß. Die Berechnung erfolgt dabei entweder auf Grundlage historischer Renditen oder mittels Szenarien auf Grundlage zukünftiger Renditen.
Da es zwar Renditen von 110% geben kann, aber welche von -110% nicht, transformiert man die Rendite r mit Hilfe des natürlichen Logarithmus in die stetige Rendite $r^s = \ln(1 + r)$. Die Verteilung der stetigen Renditen ist dann durch die Normalverteilung gut darstellbar[6], wodurch der Erwartungswert und die Standardabweichung schon vollständig beschrieben sind.

2.2.2.3 Definition der Diversifikation

Das Grundprinzip der Diversifikation kann man mit dem Satz „Don't put all your eggs into one basket“[7] zusammenfassen.
Mathematisch modellierbar ist die Diversifikation mit Hilfe von Kovarianz und Korrelation, d.h. es wird der Parallelitätsgrad bzw. der Zusammenhang zwischen zwei

[5] [1] Auckenthaler, 2001, S.55
[6] vergleiche Untersuchungen zum Schweizer Aktienindex von 1926-1999 in [1] Auckenthaler, 2001, S.61-64
[7] [2] Auckenthaler, 1994, S.137

Anlageinstrumenten gemessen. Die Kovarianz kann man dabei mit historischen oder mit zukünftigen (auf Szenarien basierenden) Renditen berechnen. Für die Kovarianz zweier Anlagen gilt die Formel[8]:

$$Cov(X,Y) = \sum_{i=1}^{n} [r_{X_i} - \mathbb{E}(r_X)] \cdot [r_{Y_i} - \mathbb{E}(r_Y)] \cdot p_i \tag{2.2}$$

dabei sind:

- n ... Anzahl der Szenarien
- r_{X_i}, r_{Y_i} ... mögliche Renditen von Anlage X bzw. Y
- p_i ... Wahrscheinlichkeit, dass X_i bzw. Y_i eintreten
- $E(r_X)$, $E(r_Y)$ ... erwartete Renditen von Anlage X bzw. Y

Für den Korrelationskoeffizient ρ normiert man die Kovarianz über das Produkt der Standardabweichungen (σ) der zwei Anlagen:

$$\rho_{XY} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{2.3}$$

ρ kann Werte zwischen -1 und $+1$ annehmen, dabei bedeutet ein Korrelationskoeffizient von $+1$ gleichlaufende, einer von -1 gegenläufige Renditen und einer von 0 bedeutet, dass es keinen Zusammenhang zwischen den zwei Anlagen gibt.
Da in einem Portfolio eine Vielzahl von Anlagen gehalten werden, wird die Portfoliorendite durch den gewogenen Durchschnitt der erwarteten Renditen aller im Portfolio enthaltenen Anlagen berechnet:

$$\mathbb{E}(r_P) = \sum_{i=1}^{n} z_i \cdot \mathbb{E}(r_i) \quad mit \quad \sum_{i=1}^{n} z_i = 1,\ z_i \geq 0 \quad \forall i \tag{2.4}$$

Bei dem Portfoliorisiko reicht es dagegen nicht, die einzelnen Varianzen der Anlagen sowie deren Gewichtung für die Berechnung zu benutzen. Es werden zusätzlich noch sämtliche Kovarianzen von jeweils zwei Anlagen benötigt. Daher berechnet man die Varianz des Portfolios (σ_P^2) wie folgt:[9]

$$\sigma_P^2 = \sum_{i=1}^{n} z_i^2 \cdot \sigma_i^2 + \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} z_i \cdot z_j \cdot Cov(i,j) \quad mit \quad \sum_{i=1}^{n} z_i = 1,\ z_i \geq 0 \quad \forall i \tag{2.5}$$

Die Gleichung (2.5) zeigt, dass das Portfoliorisiko stark von den Kovarianzen zwischen den einzelnen Anlagen abhängig ist. Ziel einer optimalen Portfoliostrategie

[8] hier für den Fall zukünftiger erwarteter Renditen
[9] für eine Herleitung der Formel siehe [13] Meintrup/Schäffler, S.129/130

muss es also sein Anlagen zu finden, die bei vorgegebenen Renditeerwartungen möglichst geringfügig oder besser noch negativ miteinander kovariieren. Der Idealfall wäre Anlagen zu finden, die so kovariieren, dass die Portfoliovarianz in (2.5) den Wert 0 ergibt. Dies wird aber in der Praxis kaum erreichbar sein.
Die theoretische Grenze der Diversifikation erkennt man, wenn man annimmt, dass in n Anlagen mit jeweiligem Anteil $1/n$ investiert wird. Dann ergibt sich für die Portfoliovarianz [10]:

$$\sigma_P^2 = \frac{1}{n} \cdot \sigma_*^2 + \frac{n-1}{n} \cdot Cov(i,j)_* \tag{2.8}$$

Wenn man die Anzahl der Anlagen (also n) gegen unendlich gehen lässt, so geht der erste Teil der Gleichung (2.8) gegen 0. Es handelt sich hier um das diversifizierbare, unsystematische Risiko. Der zweite Teil der Gleichung geht dagegen asymptotisch gegen die durchschnittliche Kovarianz, welche das nicht diversifizierbare, systematische Risiko darstellt (siehe Abb. 2.3), d.h. es wird nie gelingen das gesamte Risiko des Portfolios wegzudiversifizieren. Die Größe des Restrisikos hängt davon ab, ob der Investor das Portfolio nur mit Anlagen aus einem Land oder mehreren Ländern zusammenstellt. Man kann durch naive Diversifikation, d.h. durch die zufällige Auswahl verschiedener Anlagen[11], über den gesamten nationalen Markt circa 70% des Risikos wegdiversifizieren. Wenn man über den Weltmarkt diversifiziert, dann sind dies sogar circa 90% des Risikos[12].

[10] [1] Auckenthaler, 2001, S.82/83
Herleitung sowie Notation:

$$Durchschnittliche\, Varianz: \sigma_*^2 = \frac{1}{n} \cdot \sum_{i=1}^{n} \sigma_i^2 \tag{2.6}$$

$$Durchschnittliche\, Kovarianz: Cov(i,j)_* = \frac{1}{n \cdot (n-1)} \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} Cov(i,j) \tag{2.7}$$

Dabei ist $n \cdot (n-1)$ die Anzahl aller Kovarianzen, wobei Cov(i,j)=Cov(j,i).
Einsetzen von (2.6) und (2.7) in (2.5) mit $z_i = \frac{1}{n}\,(\forall i)$ ergibt (2.8).

[11] dabei soll die Diversifikation nicht die Rendite beeinträchtigen

[12] [1] Auckenthaler, 2001, S.87-89

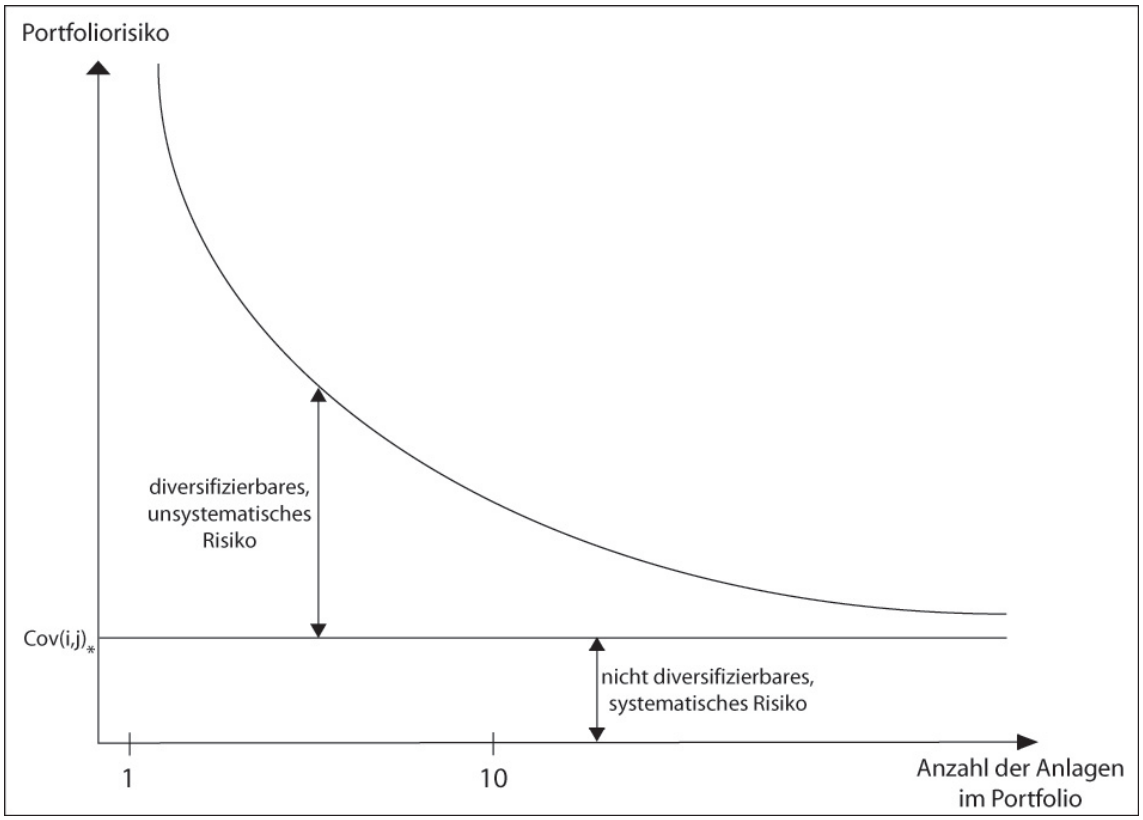

Abbildung 2.3: Systematisches und unsystematisches Risiko

2.3 Modelle zur Portfoliogestaltung

Dieser Abschnitt baut auf [2] Auckenthaler, 1994, S.152-195 sowie S.291-299 und [10] Kremer, 2006, S.87-90 auf.
Es werden die Grundzüge des Markowitz-Modells und des darauf aufbauenden Capital Asset Pricing Modells vorgestellt. Ziel ist es zu verdeutlichen, wie der Investor durch Diversifikation das Risiko des Portfolios senken kann. Da die Modelle für Aktien geschaffen wurden, wird im Folgenden von Aktienportfolios ausgegangen. Dabei ist aber eine Erweiterung der Modelle auf andere, insbesondere auch auf gemischte Portfolios möglich und sinnvoll.

2.3.1 Markowitz-Modell

Die wichtigste Erkenntnis von Markowitz war, dass ein Investor nicht die erwartete Rendite maximieren will, sondern dass er ein effizientes Portfolio erreichen möchte. Dieses kann man entweder über die Maximierung der Rendite bei einem bestimmten Risiko oder über die Minimierung des Risikos bei einer bestimmten Rendite erreichen.

2.3.1.1 Vorraussetzungen des Modells

Bezüglich des Verhaltens der Investoren werden folgende Voraussetzungen gestellt:

1. Entscheidungsparameter: Investoren orientieren sich nur an dem Erwartungswert und der Varianz. Erwartungswert, Varianz und Kovarianz und damit das effiziente Portfolio müssen ermittelbar sein.

2. Risikoaversion

3. Nutzenmaximierung: Investoren wollen mit möglichst wenig Risiko eine maximale Rendite erreichen und damit ihren Nutzen maximieren.

4. Einperiodenmodell: Der Planungshorizont beträgt eine Periode, am Ende dieser wird das Ergebnis geprüft und dann für die nächste Periode geplant.

Des Weiteren muss der Markt folgende Eigenschaften vorweisen:

5. Friktionslose Märkte: Es gibt keine Transaktionskosten und Steuern und die Anlagen sind beliebig teilbar.

6. Vollständige Konkurrenz: Der Investor hat keinen Einfluss auf den Preis. Es gibt keine Zugangsbeschränkungen zum Markt und keine Arbitragemöglichkeit.

7. Leerverkäufe sind ausgeschlossen, d.h. die Gewichte der Anlagen sind positiv.

8. Verhalten der Anlagen: Der Korrelationskoeffizient darf nicht -1 sein, es darf keine risikolose Anlage geben und mindestens zwei Anlagen haben verschiedene Renditen.

2.3.1.2 Herleitung eines effizienten und des optimalen Portfolios

Efficient Frontier

Um ein effizientes Portfolio (Efficient Frontier) zu bestimmen muss zunächst die Menge der zulässigen Portfolios abgegrenzt werden. Markowitz definiert die effizienten Portfolios folgendermaßen ([12] Markowitz, 1987, S.6):

> „An obtainable EV combination is inefficient if another obtainable combination has either higher mean and no higher variance, or less variance and no less mean. [...] An obtainable portfolio is inefficient if its EV combination is inefficient [...]. Efficient portfolios, efficient EV combinations [...] are those which are not inefficient. “[13]

[13] E steht hier für die erwartete Portfoliorendite, vgl. Gleichung (2.4)
V steht für die Portfoliovarianz, vgl. Gleichung (2.5)
für eine graphische Veranschaulichung des Efficient Frontier siehe Abb. 2.4

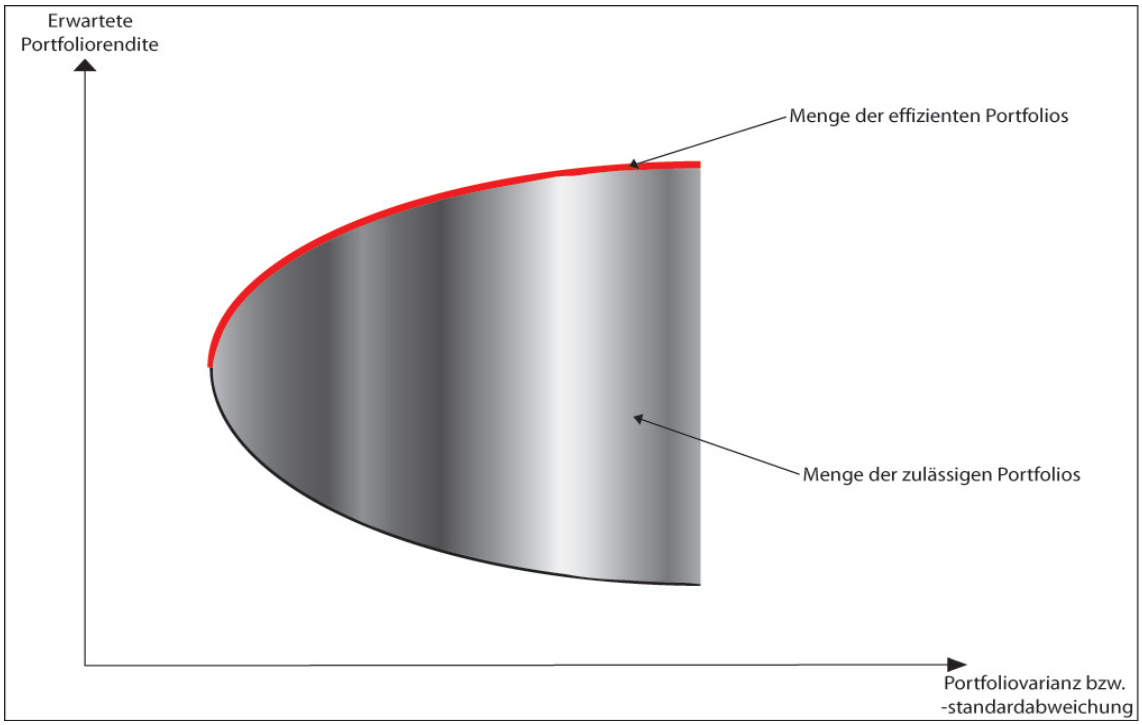

Abbildung 2.4: Efficient Frontier im Rendite-Varianz-Raum

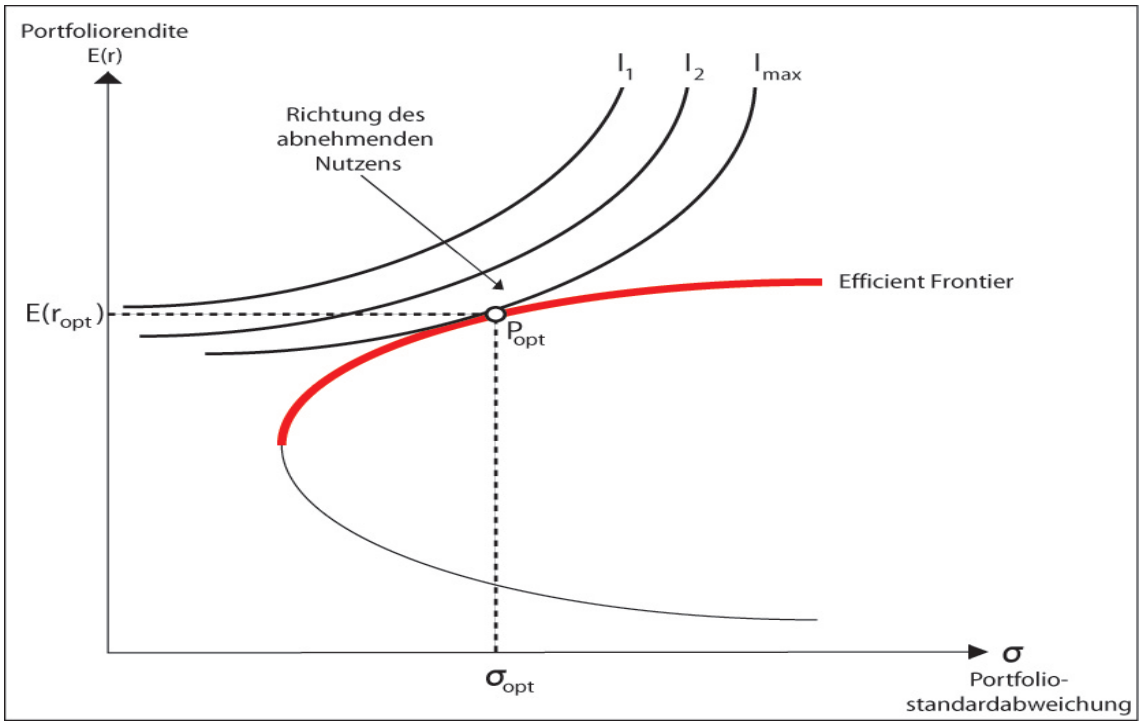

Abbildung 2.5: Das optimale Portfolio

Mathematische Herleitung des risikominimalen Portfolios für den Fall von 2 Anlagen

Ohne Beschränkung der Allgemeinheit seien im Folgenden die erwartete Rendite von Anlage i ungleich der von Anlage j, da ein rationaler Investor sonst das gesamte Kapital in die risikoärmere Anlage investieren würde. Ebenso seien die Varianzen der beiden Anlagen verschieden, da bei Gleichheit derer ein rationaler Investor das gesamte Vermögen in die Anlage mit der höheren Rendite investieren würde.
Für die Portfoliorendite gilt entsprechend Gleichung (2.4) in Abschnitt 2.2.2.3:

$$\mathbb{E}(r_P) = z_i \cdot \mathbb{E}(r_i) + z_j \cdot \mathbb{E}(r_j) \quad mit\, z_i + z_j = 1,\, z_i \geq 0,\, z_j \geq 0, \tag{2.9}$$

Für das Portfoliorisiko der zwei Anlagen i und j gilt dann:

$$\sigma_P^2 = z_i^2 \cdot \sigma_i^2 + z_j^2 \cdot \sigma_j^2 + 2 \cdot z_i \cdot z_j \cdot Cov(i,j) \tag{2.10}$$

wobei $Cov(i,j) = \rho_{i,j} \cdot \sigma_i \cdot \sigma_j$ (gemäß Gleichung (2.3))und $z_j = 1 - z_i$ gilt.
Damit ergibt sich für die Portfoliovarianz (σ_P^2):

$$\sigma_P^2 = z_i^2 \cdot \sigma_i^2 + (1 - z_i)^2 \cdot \sigma_j^2 + 2 \cdot z_i \cdot (1 - z_i) \cdot \rho_{i,j} \cdot \sigma_i \cdot \sigma_j \tag{2.11}$$

Um eine minimale Varianz zu erhalten muss die Gleichung (2.11) einmal differenziert und gleich 0 gesetzt werden. Dann erhält man:

$$0 = 2 \cdot z_i \cdot \sigma_i^2 - 2 \cdot (1 - z_i) \cdot \sigma_j^2 + 2 \cdot \rho_{i,j} \cdot \sigma_i \cdot \sigma_j - 4 \cdot z_i \cdot \rho_{i,j} \cdot \sigma_i \cdot \sigma_j \tag{2.12}$$

Dann gilt für die minimale[14] Varianz:

$$z_i = \frac{\sigma_j^2 - \rho_{i,j} \cdot \sigma_i \cdot \sigma_j}{\sigma_i^2 + \sigma_j^2 - 2 \cdot \rho_{i,j} \cdot \sigma_i \cdot \sigma_j} \quad \forall -1 < \rho_{i,j} < 1 \tag{2.13}$$

und für den Spezialfall $\rho_{i,j} = -1$:

$$z_i = \frac{\sigma_j}{\sigma_i + \sigma_j} \tag{2.14}$$

sowie für $\rho_{i,j} = 0$:

$$z_i = \frac{\sigma_j^2}{\sigma_i^2 + \sigma_j^2} \tag{2.15}$$

Für $\rho_{i,j} = +1$ besteht das risikominimale Portfolio nur aus einer Anlage (im Beispiel in Abb. 2.6 die Anlage j). (2.11) lässt sich dann schreiben als:

$$\sigma_P = z_i \cdot \sigma_i + (1 - z_i) \cdot \sigma_j \tag{2.16}$$

[14] Da nach Definition der Korrelation immer$-1 \leq \rho_{i,j} \leq 1$ gilt, folgt für die zweite Ableitung:

$$(\sigma_P^2)'' = 2 \cdot \sigma_i^2 + 2 \cdot \sigma_j^2 - 4 \cdot \rho_{i,j} \cdot \sigma_i \sigma_j \geq 2 \cdot \sigma_i^2 + 2 \cdot \sigma_j^2 - 4 \cdot \sigma_i \sigma_j = (\sigma_i - \sigma_j)^2 > 0 \quad \forall \rho_{i,j}, \quad \forall \sigma_i \neq \sigma_j$$

d.h. es liegt ein Minimum vor.

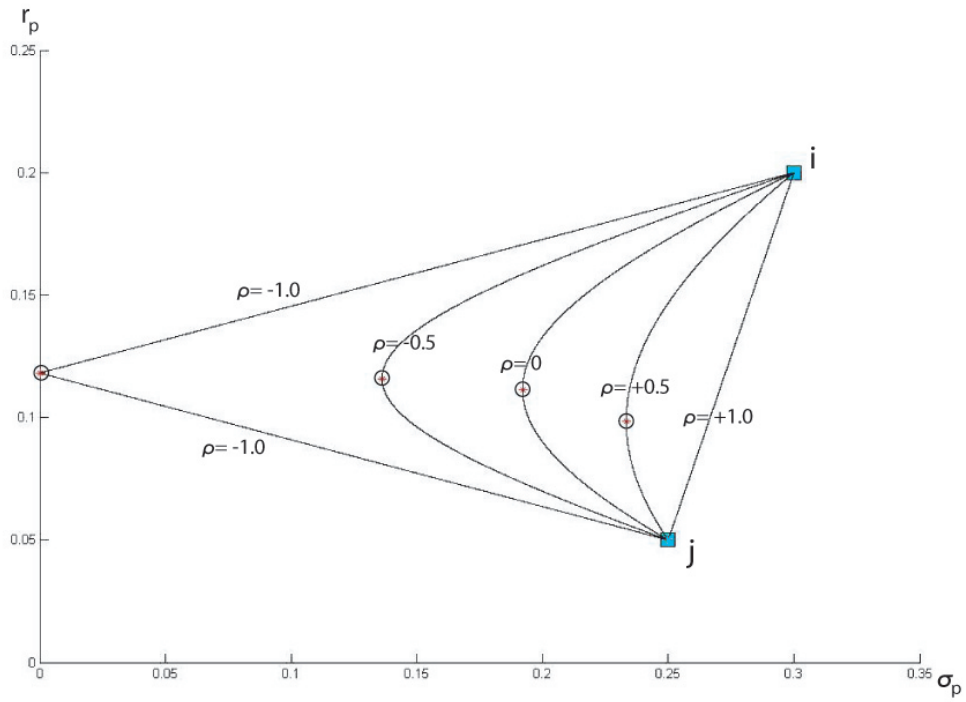

Abbildung 2.6: Portfoliorendite in Abhängigkeit von dem Portfoliorisiko für verschiedene Korrelationskoeffizienten

---

Beispiel

In Abbildung 2.6[15] sieht man ein Beispiel der zwei Anlagen i und j, für die gelten:

- $\sigma_j = 0.25 \quad \sigma_i = 0.3$
- $E(r_j) = 0.05 \quad E(r_i) = 0.2$
- $\rho_{i,j} = -1, -0.5, 0, +0.5, +1$

Die Abbildung zeigt die Rendite-Risiko-Kombination der einzelnen Anlagen i und j (blaue Rechtecke), die der verschiedenen Zusammensetzungen ($0 \leq z_i \leq 1$ und $z_j = 1 - z_i$) der Anlagen i und j sowie die Risikominimale Kombination von i und j (zu erkennen an den schwarzen Kreisen mit rotem Punkt darin) für die vorgegebenen Korrelationskoeffizienten.
Zu erkennen ist, dass bei einem Korrelationskoeffizient von -1 das Risiko vollständig

[15] in Anlehnung an [2] Auckenthaler, 1994, S.158 und [10] Kremer, 2006, S.88:
Geplottet mit MATLAB unter Verwendung der Gleichungen (2.9) und (2.11) sowie (2.13) (für das Finden des risikominimalen Portfolios)
Das zugehörige Programm *Diversifikation.m* befindet sich auf der CD im Verzeichnis:
CD:\Programme\Kapitel 2\Portfoliorendite

wegdiversifiziert werden kann. Wenn dagegen $\rho = +1$ ist, dann ist das risikominimale Portfolio immer das, welches nur aus der Anlage besteht, welche das geringere Risiko aufweist.
In der Abbildung 2.6 liegt das Efficient Frontier (vgl. Abbildung 2.4) auf der Verbindungslinie zwischen dem Punkt des risikominimalen Portfolios und dem Punkt der renditestärkeren Anlage (in der Abbildung 2.6 Anlage i).

Das optimale Portfolio

Das optimale Portfolio wird aus der Menge der effizienten bestimmt, indem das Risikoverhalten des Investors analysiert wird. Dazu gibt der Investor an wie hoch das Risiko bei einer bestimmten Rendite sein darf. Dies wird mathematisch durch die Nutzenfunktion[16] bzw. durch Risiko-Rendite-Indifferenzkurven[17] ausgedrückt.
In [2] Auckenthaler, 1994 wird die Neumann-Morgenstern-Nutzenfunktion benutzt um den Nutzen zu quantifizieren. Sie wird folgendermaßen definiert

$$U(\mathbb{E}(r)) = \mathbb{E}(r) - 0.005 \cdot A \cdot \sigma^2, \tag{2.17}$$

wobei $U(\mathbb{E}(r))$ der Nutzen, σ^2 die Varianz und A der Index der Risikoeinstellung des Investors ist. Wenn man die Gleichung (2.17) nun folgendermaßen umstellt

$$\mathbb{E}(r) = U(\mathbb{E}(r)) + 0.005 \cdot A \cdot \sigma^2, \tag{2.18}$$

dann erhält man für verschiedene Nutzenniveaus $U(\mathbb{E}(r))$ Indifferenzkurven wie in Abbildung 2.5. Dort hat z.B. die Indifferenzkurve I_1 einen höheren Nutzen $U(\mathbb{E}(r))$ als I_2, bildet aber keine zulässige Risiko-Rendite-Kombination ab, da diese nur unterhalb bzw. auf der Efficient Frontier liegen. Das optimale Portfolio liegt dann an dem Punkt, wo die Indifferenzkurve die Efficient Frontier tangiert, also auf I_{max} im Punkt P_{opt} in Abbildung 2.5.
Der Verlauf der Indifferenzkurven erklärt sich aus der Definition der Nutzenfunktion (2.17), d.h. der Investor geht ein erhöhtes Risiko σ bei gleichem Nutzenniveau $U(\mathbb{E}(r))$ nur ein, wenn er mit einer überproportional höheren Rendite $\mathbb{E}(r)$ „belohnt“ wird.
Bei der Power-Nutzenfunktion wird der Nutzen über den Konsum bestimmt, d.h. der Input ist der Konsum und der Output der dazugehörige Nutzen. Die Frage ist nun, ob wir das optimale Portfolio auch mit der Power-Nutzenfunktion genau wie mit der Neumann-Morgenstern-Nutzenfunktion im Markowitz-Modell finden können. Dazu sei folgendes Beispiel betrachtet:

Beispiel zum optimalen Portfolio

Das Programm *optportfolio.m* zu diesem Beispiel befindet sich auf der CD im Verzeichnis *CD:\Programme\Kapitel 2\Optimales Portfolio*. Sämtliche folgende Berechnungen wurden mit Hilfe dieses Programms durchgeführt. Die Daten sind gege-

[16] vergleiche Abschnitt 2.2.2.1
[17] siehe Abb. 2.5

Periode	1	2	3	4	5	$\mathbb{E}$	σ
Anlage A	0.0953	0.079	-0.0408	0.1988	0.077	0.08186	0.076070
Anlage B	-0.0619	0.1655	0.0392	-0.0514	0.2076	0.0598	0.110107

Tabelle 2.1: Daten: Beispiel zum optimalen Portfolio

ben durch Tabelle 2.1, wobei die Erwartungswerte $\mathbb{E}$ und die Standardabweichungen σ berechnet wurden. Für den Korrelationskoeffizienten gilt dann $\rho = -0.313598$. Für das risikominimale Portfolio ergibt sich dann nach (2.13) der Anteil der Anlage A $z_A = 0.636789$ ($z_B = 1 - z_A$), nach (2.9) die erwartete Portfoliorendite $E(r_P) = 0.073848$ und nach (2.11) die Portfoliostandardabweichung $\sigma_P = 0.052257$. Hauptziel des Programms ist es, die Berechnung des optimalen Portfolios mit der Neumann-Morgenstern-Nutzenfunktion im Markowitz-Modell mit der Berechnung mit der Power-Nutzenfunktion durch dynamische Porgrammierung zu vergleichen.

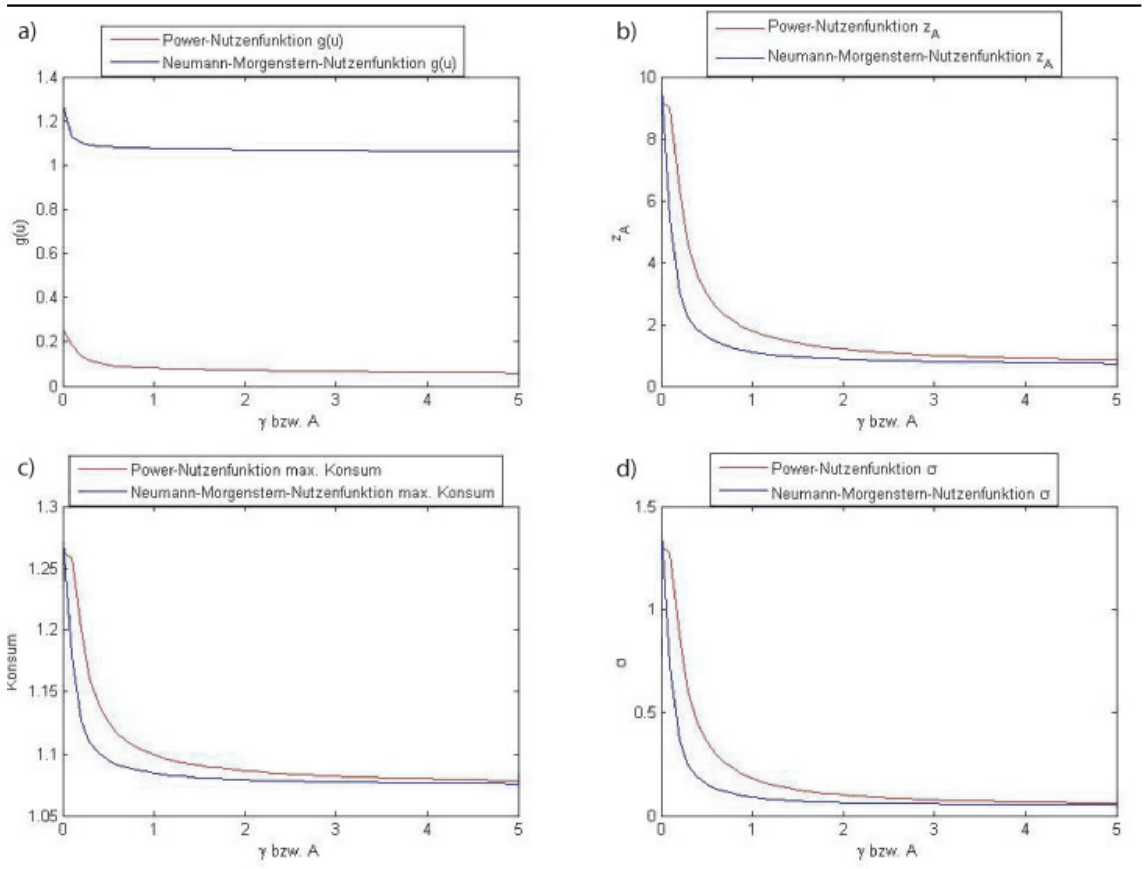

Abbildung 2.7: Vergleich Markowitz mit dynamischer Programmierung

Dazu gelten folgende Annahmen:

1. Es wird nur über eine Periode optimiert und am Ende der Periode wird das gesamte Vermögen=Startkapital·(1+Rendite) konsumiert.

2. Der Parameter A der Neumann-Morgenstern-Nutzenfunktion entspricht im Programm schon dem Term $0.005 \cdot A$ in (2.17). Außerdem wird (2.17) mit dem Startkapitel x_0 multipliziert um zu modellieren, dass man mit einem höheren Startkapital einen höheren Nutzen erhalten kann.

3. Leerverkäufe sind erlaubt. Der Anteil einer Anlage kann zwischen -10 und 10 liegen, d.h. der Investor kann maximal sein zehnfaches Startkapital in eine der beiden Anlagen investieren (möglich durch die Leerverkäufe).

Die Abbildung 2.7 zeigt die Werte der beiden Nutzenfunktionen $g(u)$ (in a) sowie die Anteile z_A der Anlage A (in b), den maximalen Konsum (in c) und das Risiko σ (in d) für beide Nutzenfunktionen für die jeweiligen optimalen Portfolios für die Risikoaversion A bzw. γ im Intervall von 0 bis 5.
Aus der Abbildung kann man ablesen, dass beide Nutzenfunktionen ein simultanes Verhalten in der Abhängigkeit zu ihrem Risikoaversionskoeffizienten A bzw. γ haben. Mit zunehmender Risikoaversion nähert sich der Anteil der Anlage A (sowie als Folge dessen der Konsum und das Risiko) dem des risikominimalen Portfolios an.
In die Power-Nutzen-Funktion fließt zwar im Gegensatz zur Neumann-Morgenstern-Nutzenfunktion die Varianz des Portfolios nicht direkt ein, aber auch sie modelliert, dass der Investor ein risikoärmeres Portfolio wählt, wenn seine Risikoaversion größer ist. Daraus kann man folgern, dass die Power-Nutzen-Funktion genauso gut geeignet ist um das optimale Portfolio zu finden.

Anmerkungen

Es ist möglich die Voraussetzungen des Modells zu modifizieren, um es realitätsnaher zu gestalten. Es können sowohl Leerverkäufe als auch die Existenz einer risikolosen Anlage zugelassen werden.
Für diesen Fall kann man die effizienten Portfolios für eine beliebige Anzahl n von Anlagen berechnen:

$$\textbf{min} \qquad \sum_{i=1}^{n} z_i^2 \cdot \sigma_i^2 + \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} z_i \cdot z_j \cdot Cov(i,j) \tag{2.19}$$

$$\textbf{s.t.} \qquad \sum_{i=1}^{n} z_i \cdot \mathbb{E}(r_i) + (1 - \sum_{i=1}^{n} z_i) \cdot r_f = \mathbb{E}(r_P)$$

$$z_i \in \mathbb{R} \quad \forall i = 1, ...n$$

Die Gleichung (2.19) gibt dabei zu einer vorgegebenen Portfoliorendite $\mathbb{E}(r_P)$ die minimale Varianz dieses Portfolios zurück. Um die Efficient Frontier zu bestimmen,

muss man für alle Portfoliorenditen zwischen der niedrigsten und der höchsten erwarteten Rendite der einzelnen Anlagen das Optimierungsproblem (2.19) lösen.
Will man Leerverkäufe verbieten, so muss man in (2.19) die Gleichungen $z_i \geq 0\,(i = 1, ..., n)$ und, wenn es keine risikolose Anlage gibt, die Gleichung $\sum_{i=1}^{n} z_i = 1$ einfügen (Dann folgt $(1 - \sum_{i=1}^{n} z_i) \cdot r_f = 0$).

2.3.1.3 Kritik am Modell

Ein schwerwiegendes Problem des Markowitz-Modells ist die große Menge an benötigten Daten. Es werden zu jeder Anlage die erwarteten Renditen und das Risiko sowie die Kovarianzen zwischen sämtlichen Anlagen benötigt. Bei n Anlagen bedeutet das, dass jeweils n mal das Risiko und die Rendite sowie $\frac{n \cdot (n-1)}{2}$ verschiedene Kovarianzen bekannt sein müssen.
Ein weiterer Nachteil ist, dass das Modell aufgrund des einperiodigen Zeithorizonts[18] statisch ist. Diesen Nachtteil kann man zwar durch Anwendung der dynamischen Programmierung ausgleichen, aber bei höher dimensionalen Problemen wächst der numerische Aufwand der dynamischen Programmierung schnell an.
Außerdem kann die Annahme des friktionslosen Marktes in der Realität nicht gehalten werden, und eine beliebige Teilbarkeit der Anlagen ist bei geringem Anlagevolumen ein Problem (ohne die beliebige Teilbarkeit wäre die Anwendung von ganzzahliger Optimierung nötig), aber ein bei großen Volumina vernachlässigbares.

2.3.2 Capital Asset Pricing Modell (CAPM)

Das Capital Asset Pricing Modell dient dazu das relevante Anlagerisiko und die Beziehung zwischen erwarteter Rendite und Risiko zu bestimmen. Dabei handelt es sich bei dem CAPM um ein Gleichgewichtsmodell, d.h. der Markt muss im Gleichgewicht sein. Die Idee dahinter ist, dass, wenn der Investor Anlagen mit höherem systematischen Risiko erwirbt (das unsystematische Risiko ist wegdiversifizierbar[19]), die Anlage auch eine höhere Rendite abwerfen sollte, weil sonst das gehaltende Portfolio nicht effizient sein kann.

2.3.2.1 Vorraussetzungen des Modells

Es gelten die Voraussetzungen 1.-6. sowie 8. des Markowitz-Modells[20]. Zusätzlich gelten noch folgende Vorraussetzungen:

1. Risikofreier Zinssatz: Es kann unbeschränkt Kapital zu diesem Zinssatz r_f angelegt und geliehen werden.

2. Homogene Erwartungen: bezüglich Erwartete Rendite, Varianz und Kovarianz von allen Investoren.

[18] siehe Abschnitt 2.3.1.1
[19] vgl. Abschnitt 2.2.2.3
[20] vgl. Abschnitt 2.3.1.1

3. Handelbarkeit der Anlagen: Menge der Anlagen ist vorgegeben und alle Anlagen werden am Markt gehandelt.

4. Informationseffizienz: Informationen sind den Investoren kostenlos, frei und zur gleichen Zeit verfügbar.

5. Kapitalmarktgleichgewicht: Sämtliche Anlagen sind zum Marktpreis im Besitz von Investoren.

2.3.2.2 Capital Market Line (CML)

Aufgrund der Annahme der homogenen Erwartungen werden die Renditen und Risiken sämtlicher Anlagen von allen Investoren gleich eingeschätzt, d.h. es herrscht eine einheitliche Vorstellung über den Verlauf der Efficient Frontier. Dies hat zur Folge, dass jeder Investor in das Marktportfolio investiert, in dem sämtliche risikobehafteten Anlagen proportional zu ihren Marktwerten enthalten sind.
Wenn nun ein risikoloser Zinssatz eingeführt wird, dann verändert sich die Efficient Frontier von AC zu der CML r_fME (siehe Abbildung 2.8), welche wieder wegen Vorraussetzung 2 für alle Investoren gleich ist. Aufgrund Voraussetzung 1 kann jeder Investor individuell nach seiner Risikopräferenz sein optimales Portfolio auf der CML wählen, d.h. er kann Kapital zum Zins r_f aufnehmen und eine höhere erwartete Rendite bei höherem Risiko oder Kapital zum Zins r_f anlegen und eine kleinere erwartete Rendite bei geringerem Risiko erreichen.
Aus der Abbildung 2.8 kann man leicht ablesen, dass die erwartete Rendite jedes effizienten Portfolios eine lineare Funktion der Standardabweichung σ_P ist:

$$\mathbb{E}(r_P) = r_f + \frac{\mathbb{E}(r_M) - r_f}{\sigma_M} \cdot \sigma_P \tag{2.20}$$

Die anderen Daten (r_f, $\mathbb{E}(r_M)$, σ_M) werden vom Markt vorgegeben.

2.3.2.3 Security Market Line (SML)

Im Gegensatz zur CML, welche Rendite-Risiko-Verhältnisse effizienter Portfolios zeigt, wird mit der SML das Rendite-Risiko-Verhältnis einzelner Anlagen und nicht effizienter Portfolios, welche jeweils ebenfalls Bestandteil des Marktportfolios sind, ermittelt.

Herleitung der SML

Angenommen in einem Markt im Gleichgewicht, bei dem die Anlage A mit dem Anteil z_A im Marktportfolio M enthalten ist, wird der Marktanteil von z_A um w_A

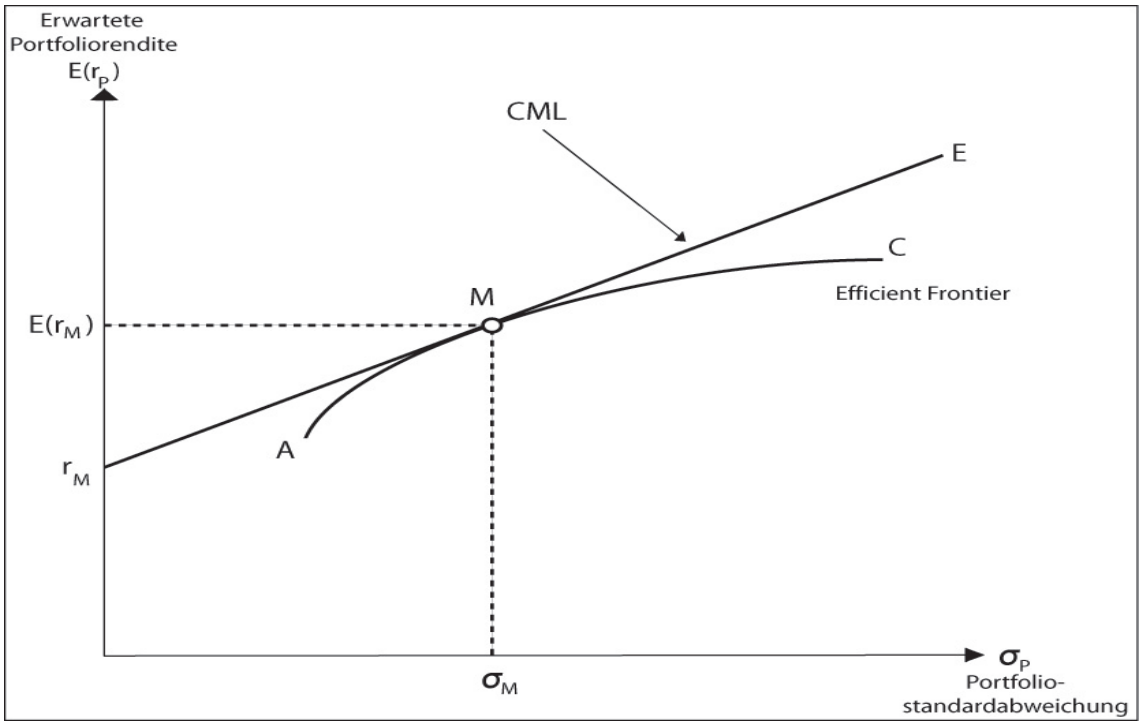

Abbildung 2.8: Capital Market Line

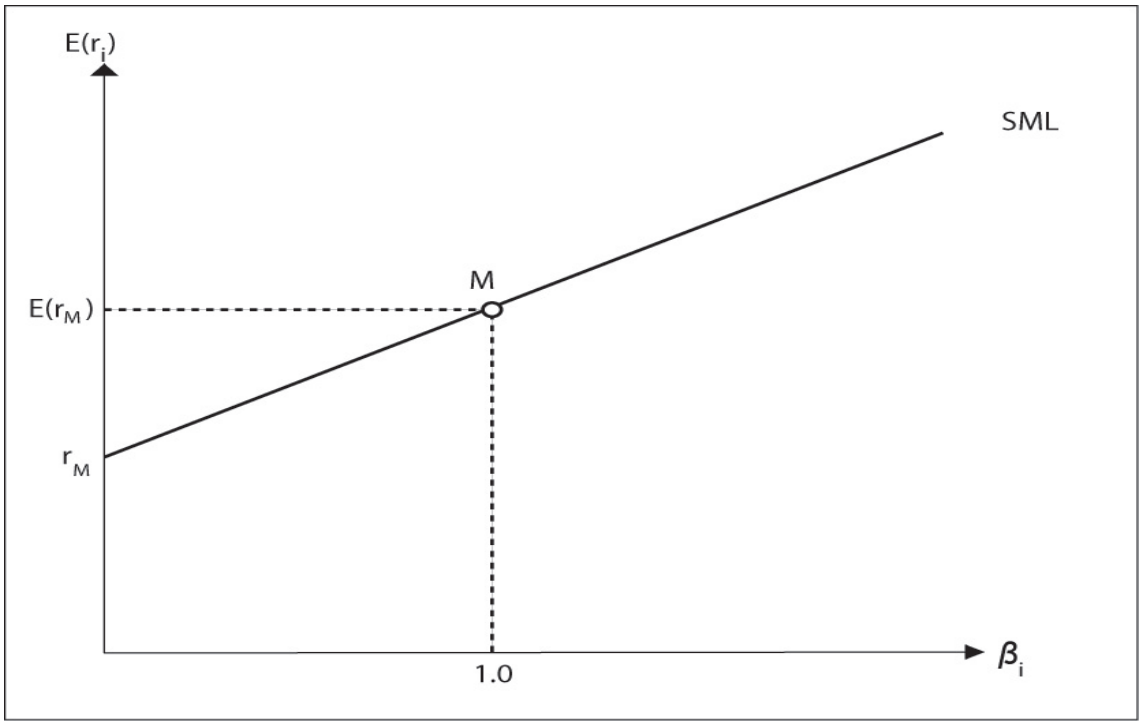

Abbildung 2.9: Security Market Line

verändert. Dann wird ein Nachfrageüberhang erzeugt. Für die Portfoliorendite und -standardabweichung gilt:

$$\mathbb{E}(r_P) = w_A \cdot \mathbb{E}(r_A) + (1 - w_A) \cdot \mathbb{E}(r_M) \tag{2.21}$$

$$\sigma_P = \sqrt{w_A^2 \cdot \sigma_A^2 + (1 - w_A)^2 \cdot \sigma_M^2 + 2 \cdot w_A \cdot (1 - w_A) \cdot Cov(A, M)} \tag{2.22}$$

Im Gleichgewicht muss Anlage A aber mit Anteil z_A in M enthalten sein, d.h. $w_A = 0$. Für die Veränderung der erwarteten Rendite und der Standardabweichung im Punkt $w_A = 0$ gilt dann:

$$\frac{\partial \mathbb{E}(r_P)}{\partial w_A}|_{w_A=0} = \mathbb{E}(r_A) - \mathbb{E}(r_M) \tag{2.23}$$

$$\frac{\partial \sigma_P}{\partial w_A}|_{w_A=0} = \frac{Cov(A, M) - \sigma_M^2}{\sigma_M} \tag{2.24}$$

Damit folgt mit

$$\partial \mathbb{E}(r_P)/\partial \sigma_P = \frac{\partial \mathbb{E}(r_P)/\partial w_A}{\partial \sigma_P/\partial w_A} \tag{2.25}$$

wobei $\partial \mathbb{E}(r_P)/\partial \sigma_P$ für $w_A = 0$ dem Anstieg der CML im Punkt M entspricht, d.h.:

$$\frac{\mathbb{E}(r_A) - \mathbb{E}(r_M)}{\frac{Cov(A,M) - \sigma_M^2}{\sigma_M}} = \frac{\mathbb{E}(r_M) - r_f}{\sigma_M} \tag{2.26}$$

Damit folgt für die erwartete Rendite der Anlage A:

$$\mathbb{E}(r_A) = r_f + (\mathbb{E}(r_M) - r_f) \cdot \beta_A \tag{2.27}$$

wobei $\beta_A = \frac{Cov(A,M)}{\sigma_M^2}$ die Risikohöhe der Anlage A im Vergleich zum Marktrisiko beschreibt. Mithilfe von (2.27) kann nun die SML hergeleitet werden (vgl. Abb. 2.9). Dabei ist offensichtlich, dass der risikolose Zins r_f ein β von 0, das Marktportfolio ein β von 1 hat und dass der Anstieg der SML $(\mathbb{E}(r_M) - r_f)$ beträgt.
Aus der Gleichung (2.27) folgt auch, dass mit zunehmenden β auch die erwartete Rendite steigt. Dabei steht das β aber nur für das systematische Risiko, welches der Investor für eine höhere Rendite in Kauf nimmt, da das unsystematische wegdiversifiziert werden kann.

Anwendung der SML

Aufgrund der Annahme des Marktgleichgewichts liegen alle Anlagen mit „fairem“ Marktpreis auf der SML. Nimmt man nun an, dass kein Marktgleichgewicht vorherrscht, d.h. dass es über- und unterbewertete Anlagen gibt, dann kann man die SML dazu benutzen diese zu erkennen. Dabei liegen überbewertete unterhalb der SML und unterbewertete oberhalb der SML. Dies kann der Investor dazu nutzen um über- bzw. unterbewertete Anlagen zu erkennen und entsprechend zu reagieren, d.h. er kauft die unterbewerteten um die Überschussrendite, die Differenz zwischen erwarteter Rendite und der Rendite der Anlage auf der SML bei gleicher Risikohöhe β, auszunutzen bzw. verkauft die überbewerteten.
Langfristig wird sich durch den Verkauf der überbewerteten Anlagen, wodurch sinkende Kurse und damit ein Ansteigen der effektiven Verzinsung folgt, und den Kauf der unterbewerteten Anlagen, wodurch steigende Kurse und damit eine sinkende effektive Verzinsung folgt, wieder ein Marktgleichgewicht einstellen.

2.3.2.4 Kritik am Modell

Probleme des CAPM sind die Annahmen der homogenen Erwartungen und der Informationseffizients. Wenn diese Vorraussetzungen nicht erfüllt sind, ist ein Test der Resultate des CAPM nicht sinnvoll (vgl.[2] Auckenthaler, 1994, S. 298).

Kapitel 3

Mathematische Grundlagen

In diesem Kapitel werden die mathematischen Grundlagen zur Lösung der stochastischen dynamischen Optimierungsprobleme, welche in den Modellen in Kapitel 4 vorkommen, vorgestellt.
Im Abschnitt 3.1 sollen die Grundlagen der stochastischen dynamischen Programmierung vorgestellt werden. Dazu werden zuerst einige wichtige Begriffe aus der Stochastik, die wesentlich zum weiteren Verständnis sind, eingeführt. Die Abschnitte 3.1.2 und 3.1.3, welche die grundlegenden Begriffe des stochastischen Kontrollsystems bzw. des stochastischen dynamischen Optimierungsproblems definieren sollen, liefern die wesentlichen Definitionen und Sätze für die Formulierung und den Beweis des zur Lösung notwendigen Bellman'schen Optimalitätsprinzips auf unendlichem Horizont. Abschließend wird darauf eingegangen, wie mit dem Optimalitätsprinzip die optimalen Kontrollprozesse charakterisiert werden können.
Im Abschnitt 3.2 soll der Weg zur Implementierung der theoretischen Grundlagen in ein Computerprogramm mithilfe der Diskretisierung bereitet werden. Dazu wird die Diskretisierung in der Zeit und im Raum betrachtet sowie auf Unterschiede in der Anwendung auf Probleme mit unendlichem Zeithorizont eingegangen.

Dieses Kapitel orientiert sich an [4] Grüne, 2004 und an [6] Grüne, 2007 sowie an [5] Grüne, 2008.

3.1 Grundlagen der stochastischen dynamischen Programmierung

In diesem Abschnitt soll nun, nach einer einleitenden Betrachtung einiger stochastischer Grundlagen, auf die zeitdiskreten Varianten des stochastischen Kontrollsystems und des stochastischen dynamischen Optimierungsproblems eingegangen werden, welche für die Modelle in Kapitel 4 wesentlich sind.

3.1.1 Stochastische Grundlagen

Dieser Abschnitt gibt nur einen Überblick über einige wichtige im Folgenden gebrauchte stochastische Begriffe. Für eine detailierte Einführung in die Stochastik sei auf [13] Meintrup, Schäffler, 2005 verwiesen.

Definition 3.1 (Wahrscheinlichkeitsraum)
Ein Wahrscheinlichkeitsraum ist ein Tripel (Ω, Σ, P) *mit:*

- Ω *ist eine Menge von Elementarereignissen* $\omega \in \Omega$
- Σ *ist eine* $\sigma - Algebra$ *auf* Ω.
  Sei $\mathcal{A} \subseteq \mathcal{P}(\Omega)$ *(*$\mathcal{P}(\Omega)$ *steht für die Potenzmenge von* Ω*), dann wird mit* $\sigma(\mathcal{A})$ *die kleinste* σ*-Algebra, die* $\mathcal{A}$ *enthält, bezeichnet.*
- $P : \Sigma \to [0, 1]$ *ist ein Wahrscheinlichkeitsmaß auf* Σ

Definition 3.2 (Borel-σ-Algebra)
Die Borel-σ*-Algebra* $\mathcal{B}$ *ist definiert durch*

$$\mathcal{B} := \sigma\left(\{(a_1, b_1] \times \ldots \times (a_n, b_n] \mid a_i, b_i \in \mathbb{R}, a_i < b_i\}\right)$$

auf $\mathbb{R}^n$*. Mengen B aus der Borel-*σ*-Algebra* $\mathcal{B}$ *heißen Borel-Mengen.*

Definition 3.3 (Stochastischer Prozess)
Ein zeitdiskretes stochastisches System ist durch eine Abbildung

$$f : \mathbb{R}^n \to \mathbb{R}^m, \quad (x, z) \longmapsto f(x, z) \tag{3.1}$$

gegeben. Dabei ist $x \in \mathbb{R}^n$ *der Zustand und* $z \in \mathbb{R}^m$ *der stochastische Einfluss. Für einen Anfangswert* $x_0 \in \mathbb{R}^n$ *und eine Folge von Zufallsvariablen* $Z_0, Z_1, \ldots$ *wird die Lösung* $X_t = X_t(x_0)$ *für* $t > 0$ *induktiv durch*

$$X_0 = x_0, \quad X_{t+1} = f(X_t, Z_t) \tag{3.2}$$

definiert. Dabei ist $X_t(x_0)$ *für jeden Anfangswert* x_0 *ein stochastischer Prozess, d.h. eine zeitabhängige Zufallsvariable.*

Definition 3.4 (Natürliche Filtration)
Gegeben sei ein Wahrscheinlichkeitsraum (Ω, Σ, P).
Eine Filtration $\mathcal{F} = (\mathcal{F}_t, t = 0, 1, \ldots)$ *ist eine Folge von* σ*-Algebren* $\mathcal{F}_t \subseteq \Sigma$*, t=0,1,...,*
für die die Monotonieeigenschaft $\mathcal{F}_s \subseteq \mathcal{F}_t$ *für alle* $0 \leq s \leq t$ *gilt.*
Ein stochastischer Prozess X_t *heißt adaptiert zur Filtration* $\mathcal{F}$*, falls* $(X_0, \ldots, X_t)$ *messbar bezüglich* $\mathcal{F}_t$ *ist, d.h. es gilt* $\sigma((X_0, \ldots, X_t)) \subseteq \mathcal{F}_t$*, t=0,1,...*
Die natürliche Filtration eines stochastischen Prozesses X_t *ist gegeben durch*

$$\mathcal{F}_t = \sigma((X_0, \ldots, X_t)), \quad t = 0, 1, \ldots \tag{3.3}$$

Definition 3.5 (Bedingter Erwartungswert)
Es sei die charakteristische Funktion gegeben durch

$$\chi_A(x) = \begin{cases} 1 & , \quad x \in A \\ 0 & , \quad x \notin A. \end{cases} \tag{3.4}$$

Dann ist der bedingte Erwartungswert einer Zufallsvariable X definiert durch

$$\mathbb{E}(X \mid A) = \frac{\mathbb{E}(X \cdot \chi_A)}{P(A).} \tag{3.5}$$

Definition 3.6 (Verallgemeinerter Erwartungswert)
Für die natürliche Filtration, gegeben durch Gleichung (3.3), und $s \leq t$ definiert man den verallgemeinerten Erwartungswert durch

$$\mathbb{E}_s(X_t) := \mathbb{E}(X_t \,|\, \mathcal{F}_s) \tag{3.6}$$

Bemerkung 3.7 *Folgende Eigenschaften des verallgemeinerten Erwartungswerts sind wichtig für den Beweis des Bellman'schen Optimalitätsprinzip in Abschnitt 3.1.4.2:*

1. $\mathbb{E}_0(X_t) = \mathbb{E}(X_t)$
2. $\mathbb{E}(\mathbb{E}_s(X_t)) = \mathbb{E}(\mathbb{E}(X_t|\mathcal{F}_s)) = \mathbb{E}(\mathbb{E}(X_t|X_s)) = \mathbb{E}(X_t)$

Für weitere Eigenschaften und Beweise sei auf [6] Grüne, 2007, S.5-12 sowie auf [13] Meintrup/Schäffler, 2005, S.219 verwiesen.

Im Folgenden seien $x \in \mathbb{R}^n$ der Zustand, $u \in U \subseteq \mathbb{R}^l$ der Kontrollwert und $z \in \mathbb{R}^m$ der stochastische Einfluss.

3.1.2 Das stochastische Kontrollsystem

In diesem Abschnitt sollen nach der Definition des zeitdiskreten stochastischen Kontrollsystems unterschiedliche Darstellungen für zulässige Kontrollsysteme vorgestellt werden.

Definition 3.8 (Stochastische Kontrollsystem)
Ein zeitdiskretes stochastisches Kontrollsystem ist gegeben durch eine Abbildung

$$\begin{aligned} f &: \mathbb{R}^n \times U \times \mathbb{R}^m \to \mathbb{R}^n \\ &(x, u, z) \longmapsto f(x, u, z). \end{aligned}$$

Die Lösung $X_t = X_t(x_0, \boldsymbol{u})$ des System wird für $t \geq 0$ induktiv durch

$$X_0 = x_0,\ X_{t+1} = f(X_t, u_t, Z_t) \tag{3.7}$$

definiert. Dabei ist $x_0 \in \mathbb{R}^n$ ein Anfangswert, $\boldsymbol{u} = (u_t, t \in \mathbb{N}_0)$ ein bezüglich Σ messbarer Kontrollprozess und Z_0, Z_1, ... eine Folge von Zufallsvariablen.

Zuerst sollen die zulässigen Kontrollprozesse maßtheoretisch definiert werden.

Definition 3.9 (Zulässiger Kontrollprozess)
Gegeben sei ein stochastisches Kontrollsystem, ein Anfangswert x_0 und ein bezüglich Σ messbarer Kontrollprozess $\boldsymbol{u}$. Weiter sei mit $\mathcal{F}_t = \sigma((X_0, ..., X_t))$ die natürliche Filtration $\mathcal{F}$ des entstehenden stochastischen Prozesses $X_t = X_t(x_0, \boldsymbol{u})$ gegeben.
Dann heißt $\boldsymbol{u}$ ein zulässiger Kontrollprozess zum Anfangswert x_0, falls $\boldsymbol{u}$ adaptiert zur Filtration $\mathcal{F}$ ist, d.h. es gilt

$$\sigma(u_t) \subseteq \mathcal{F}_t \quad \forall t \in \mathbb{N}_0.$$

$\mathcal{U}_{x_0}$ ist dabei die Menge aller zulässigen Kontrollprozesse zum Anfangswert x_0.

Diese Definition ist wohldefiniert, da die Zulässigkeit von $\mathbf{u}$ nicht direkt, sondern nur rekursiv, von $\mathcal{F}$ abhängt. Welche Zufallsvariable u_t als t-te Komponente von $\mathbf{u}$ zulässig ist hängt über die Filtration $\mathcal{F}_t$ von $X_0, ..., X_t$ und damit nur von $u_0, ..., u_{t-1}$ ab.
Für die praktische Anwendung ist es sinnvoll noch kontrolltheoretische Formulierungen eines zulässigen Kontrollprozesses zu betrachten.

Satz 3.10
Ein Kontrollprozess $\boldsymbol{u}$ ist genau dann zulässig, wenn eine der folgenden Bedingungen erfüllt ist.

1. *Der Kontrollprozess erfüllt für alle $\omega_1, \omega_2 \in \Omega$ und alle $t \in \mathbb{N}_0$ die Implikation*

$$X_s(\omega_1) = X_s(\omega_2)\, \forall s = 0, ..., t \;\Rightarrow\; u_t(\omega_1) = u_t(\omega_2).$$

2. *Es existieren Abbildungen $h_t : (\mathbb{R}^n)^{t+1} \to U$, $t \in \mathbb{N}_0$, so dass für alle $t \in \mathbb{N}_0$*

$$u_t = h_t(X_0, ..., X_t)\, gilt.$$

Für den Beweis sei auf [6] Grüne, S.14/15, verwiesen.

Die folgenden zwei Lemmas werden für den Beweis des Bellman'schen Optimalitätsprinzip auf unendlichem Horizont benötigt.
Das erste beschreibt, wie aus einem zulässigen Kontrollprozess ein neuer zulässiger Kontrollprozess mittels Konkatenation konstruiert werden kann. Im zweiten wird die Frage beantwortet, wie sich ein zulässiger Kontrollprozess, der mit dem Endstück eines gegebenen Kontrollprozesses übereinstimmt, definieren lässt. Für die Beweise sei auf [6] Grüne, S.16-18, verwiesen.

Lemma 3.11 *Es sei zu jedem Anfangswert x ein zulässiger Kontrollprozess $\boldsymbol{u}^x = (u_t^x)$ gegeben. Weiter sei eine Zeit $s \in \mathbb{N}$, ein weiterer Anfangswert x_0 und ein für x_0 zulässiger Kontrollprozess $\boldsymbol{u}$ gegeben.*
Damit wird nun mittels Konkatenation ein neuer Kontrollprozess $(\widetilde{u}_t) = \boldsymbol{u}\&_s\boldsymbol{u}^x$ konstruiert, wobei $(\widetilde{u}_t)$ definiert wird durch

$$\widetilde{u}_t = \begin{cases} u_t & , \quad t = 0, 1, ..., s-1 \\ u_{t-s}^{X_s} & , \quad t = s, s+1, ... \end{cases}$$

Dann ist $\boldsymbol{u}\&_s\boldsymbol{u}^x$ für die Lösung $\widetilde{X}_t = X_t(x_0, \boldsymbol{u}\&_s\boldsymbol{u}^x)$ zulässig und für $X_t^x = X_t(x, \boldsymbol{u}^x)$ gilt

$$P((u_0^x, ..., u_t^x, X_0^x, ..., X_t^x) \in B) = P((\widetilde{u}_s, ...\widetilde{u}_{s+t}, \widetilde{X}_s, ..., \widetilde{X}_{s+t}) \in B \,|\, \widetilde{X}_s = x) \quad (3.8)$$

für alle $t \in \mathbb{N}_0$ und alle Borel-Mengen B mit passender Dimension.

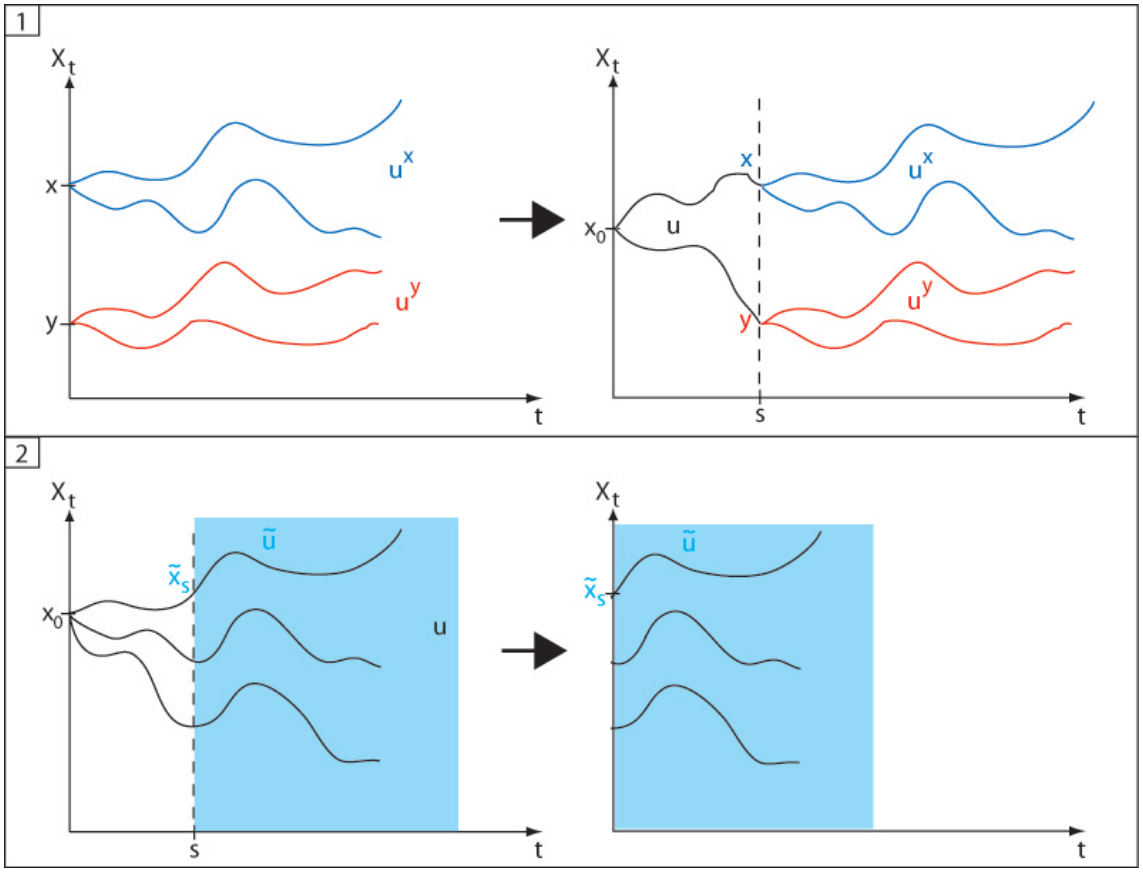

Abbildung 3.1: Graphische Veranschaulichung zu: 1) Lemma 3.11 und 2) Lemma 3.12

Das Lemma 3.11 sagt aus, dass die gemeinsame Verteilung der Prozesse u_t^x und X_t^x mit der mittels $\widetilde{X}_s = x$ bedingten gemeinsamen Verteilung der verschobenen Prozesse $\widetilde{u}_{t+s}$ und $\widetilde{X}_{t+s}$ übereinstimmt. Anschaulich wird dadurch eine Verschiebung des Kontrollprozesses $\mathbf{u}^x$ nach rechts und das Anhängen desselben an einen anderen zulässigen Kontrollprozess $\mathbf{u}$ beschrieben. Die Lösung des entstehenden Kontrollprozess stimmt in Wahrscheinlichkeit mit der um s nach rechts verschobenen Lösung von $\mathbf{u}^x$ überein, siehe auch Abb. 3.1 Teil 1.

Lemma 3.12 *Gegeben sei ein zu einem Anfangswert x_0 zulässiger Kontrollprozess $\boldsymbol{u}$ und ein $s \in \mathbb{N}$. Weiter seien $\widetilde{x}_0, ..., \widetilde{x}_s \in \mathbb{R}^n$ und*

$$A := \{\omega \in \Omega \,|\, X_t(\omega) = \widetilde{x}_t,\ t = 0, ..., s\} \in \mathcal{F}_s.$$

Dann existiert eine Kontrollfunktion $\widetilde{\boldsymbol{u}}$, die zulässig für den Anfangswert $\widetilde{x}_s$ ist und für die und die zugehörige Lösung $\widetilde{X}_t = X_t(\widetilde{x}_s, \widetilde{\boldsymbol{u}})$ die Gleichung

$$P((\widetilde{u}_0, ..., \widetilde{u}_t, \widetilde{X}_0, ..., \widetilde{X}_t) \in B) = P((u_s, ..., u_{s+t}, X_s, ..., X_{s+t}) \in B \,|\, A) \tag{3.9}$$

für alle $t \in \mathbb{N}_0$ und alle Borel-Mengen B passender Dimension erfüllt ist.

Mit anderen Worten besagt das Lemma 3.12, dass die gemeinsame Verteilung der Prozesse $\widetilde{u}_t$ und $\widetilde{X}_t$ mit der mittels A bedingten gemeinsamen Verteilung der verschobenen Prozesse u_{t+s} und X_{t+s} übereinstimmt. Anschaulich wird damit beschrieben, dass ein zulässiger Kontrollprozess $\widetilde{\mathbf{u}}$ definierbar ist, der in Wahrscheinlichkeit mit dem um s nach links verschobenen Lösung des gegebenen zulässigen Kontrollprozesses $\mathbf{u}$ übereinstimmt. Siehe auch Abb. 3.1 Teil 2.

3.1.3 Das stochastische dynamische Optimierungsproblem

In diesem Abschnitt soll die grundlegende Problemklasse für die mathematische Modellierung des Portfolioproblems vorgestellt werden.

Definition 3.13 (Stochastisch dynamisches Optimierungsproblem)
Gegeben sei ein stochastisches Kontrollsystem gemäß Definition 3.8 mit der Lösung $X_t = X_t(x_0, \boldsymbol{u})$, eine laufende Ertragsfunktion $l : \mathbb{R}^n \times U \to \mathbb{R}$ und eine End-Ertragsfunktion $L : \mathbb{R}^n \to \mathbb{R}$ sowie ein Diskontfaktor $\beta \in (0, 1]$.
Dann ist das stochastische dynamische Optimierungsproblem auf endlichem Zeithorizont $\{0, 1, ..., T\}$ definiert als

$$Maximiere \quad J_T(x_0, \boldsymbol{u}) := \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t, u_t) + \beta^T \cdot L(X_T)\right\}. \tag{3.10}$$

Auf unendlichem Zeithorizont wird das Problem für $L \equiv 0$ definiert als

$$Maximiere \quad J_\infty(x_0, \boldsymbol{u}) := \lim_{T \to \infty} J_T(x_0, \boldsymbol{u}). \tag{3.11}$$

Dabei sind $\boldsymbol{u} = (u_t, t \in \mathbb{N}_0) \in \mathcal{U}_{x_0}$ sämtliche für x_0 zulässigen Kontrollprozesse.

Bemerkung 3.14 *Das Problem kann durch Beschränkungen ergänzt werden:*
Kontrollbeschränkungen: $u_t \in U(X_t)$ *für gegebene kompakte Mengen U(x)*
Zustandsbeschränkungen: $X_t \in \mathcal{X}$ *für eine Menge* $\mathcal{X} \subset \mathbb{R}^n$

Definition 3.15 (Optimale Wertefunktion, Kontrollprozess und Lösung)
Gegeben sei ein stochastisches dynamisches Optimierungsproblem mit $T \in \mathbb{N}_0 \cup \{\infty\}$.

- *Dann wird die optimale Wertefunktion* $V_T : \mathbb{R}^n \to \mathbb{R}$ *definiert als*

$$V_T(x_0) := \sup_{\boldsymbol{u} \in \mathcal{U}_{x_0}} J_T(x_0, \boldsymbol{u}). \tag{3.12}$$

  Die Funktion V_T *ordnet jedem Anfangswert den maximal erzielbaren Wert des Funktionals* $J_T(x_0, \boldsymbol{u})$ *zu.*

- *Der Kontrollprozess* $\boldsymbol{u}^T \in \mathcal{U}(x_0)$ *heißt optimal für* J_T*, falls*

$$J_T(x_0, \boldsymbol{u}^T) = V_T(x_0) \quad \textit{gilt.} \tag{3.13}$$

- *Die zugehörige Lösung* $X_t(x_0, \boldsymbol{u}^T)$ *heißt dann optimale Lösung.*

3.1.4 Bellman'sche Optimalitätsprinzip auf unendlichem Horizont

Hier soll nun das Bellman'sche Optimalitätsprinzip auf unendlichem Horizont zur Lösung stochastischer dynamischer Optimierungsprobleme vorgestellt werden.
Dazu soll zuächst die Transversalitätsbedingung definiert werden, welche man für den Beweis der Eindeutigkeit des Prinzips benötigt.

Definition 3.16 (Transversalitätsbedingung)
Eine Funktion $W : \mathbb{R}^n \to \mathbb{R}$ *erfüllt die Transversalitätsbedingung, falls für alle* $x_0 \in \mathbb{R}^n$ *und jede Folge von Kontrollprozessen* $\boldsymbol{u}^T \in \mathcal{U}(x_0)$ *mit den Lösungen* $X_t^T = X_t(x_0, \boldsymbol{u}^T)$ *folgende Implikationen gelten:*

$$(1)\ \limsup_{T\to\infty} \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t^T, u_t^T) + \beta^T \cdot W(X_T^T)\right\} < +\infty \Rightarrow \liminf_{T\to\infty} \mathbb{E}\left\{\beta^T \cdot W(X_T^T)\right\} \geq 0$$

$$(2)\ \liminf_{T\to\infty} \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t^T, u_t^T) + \beta^T \cdot W(X_T^T)\right\} > -\infty \Rightarrow \limsup_{T\to\infty} \mathbb{E}\left\{\beta^T \cdot W(X_T^T)\right\} \leq 0$$

Die Transversalitätsbedingung ist für alle $\mathbf{u}^T$ erfüllt, wenn W beschränkt und $\beta < 1$ ist, da dann gilt: $\beta^T \cdot W(X_T^T) \to 0$. Die optimale Wertefunktion ist beschränkt, falls $\beta < 1$ gilt und l beschränkt ist.

3.1.4.1 Formulierung des Prinzips

Satz 3.17 (Bellman'sche Optimalitätsprinzip auf unendlichem Horizont)

Die optimale Wertefunktion V_∞ erfüllt für alle $x_0 \in \mathbb{R}^n$ und alle $T \in \mathbb{N}$ die Gleichung

$$V_\infty(x_0) = \sup_{\boldsymbol{u} \in \mathcal{U}(x_0)} \mathbb{E}\left\{ \sum_{t=0}^{T-1} \beta^t \cdot l(X_t, u_t) + \beta^T \cdot V_\infty(X_T) \right\} \tag{3.14}$$

mit $X_t = X_t(x_0, \boldsymbol{u})$. Für T=1 gilt insbesondere

$$V_\infty(x_0) = \sup_{u \in U} \mathbb{E}\left\{ l(x_0, u) + \beta \cdot V_\infty(f(x_0, u, Z_0)) \right\}. \tag{3.15}$$

Die optimale Wertefunktion V_∞ erfülle weiterhin die Transversalitätsbedingung. Dann stimmen V_∞ und jede weitere Funktion W, die das Optimalitätsprinzip (3.14) und die Transversalitätsbedingung erfüllt, überein.

3.1.4.2 Beweis

Der Beweis erfolgt in drei Teilen:

1. Nachweis von (3.15)
   (a) "$\leq$"
   (b) "$\geq$"
2. Per Induktion wird mit Hilfe von 1. (3.14) nachgewiesen
3. Nachweis der Eindeutigkeitsaussage

Zu 1.(a):
Sei $x_0 \in \mathbb{R}^n$ und $\mathbf{u} \in \mathcal{U}(x_0)$ beliebig, dann gilt für $X_t = X_t(x_0, \mathbf{u})$ mit Gleichung (3.10) und (3.11):

$$J_\infty(x_0, \mathbf{u}) = \mathbb{E}\left\{ \sum_{t=0}^{\infty} \beta^t \cdot l(X_t, u_t) \right\} = l(x_0, u_0) + \mathbb{E}\left\{ \mathbb{E}_1\left(\sum_{t=1}^{\infty} \beta^t \cdot l(X_t, u_t) \right) \right\} \tag{3.16}$$

Dabei wurde die Eigenschaft 2 aus Bemerkung 3.7 für s=1 sowie die Tatsache, dass $l(x_0, u_0)$ keine Zufallsvariable ist, genutzt. Nach Definition 3.6 und Bemerkung 3.7 gilt für alle $\omega \in \Omega$:

$$\mathbb{E}_1\left(\sum_{t=1}^{\infty} \beta^t \cdot l(X_t, u_t) \right)(\omega) = \mathbb{E}\left(\sum_{t=1}^{\infty} \beta^t \cdot l(X_t, u_t) \,\middle|\, A \right) \tag{3.17}$$

mit $A = \{\widetilde{\omega} \in \Omega \,|\, X_1(\widetilde{\omega}) = X_1(\omega)\}$.
Das A in Lemma 3.12 hat die gleiche Form, da hier $X_0 = x_0$ keine Zufallsvariable, d.h. unabhängig von ω, ist. Unter Anwendung von Lemma 3.12 mit s=1 wird für jedes $\omega \in A$ die Lösung X_t und der Kontrollprozess u_t durch $\widetilde{X}_{t-1}$ und $\widetilde{u}_{t-1}$ ersetzt. Damit folgt unmittelbar, dass $\widetilde{X}_0 = X_1(\omega)$ gilt.
Da die Verteilung der beiden Lösungen bzw. Kontrollprozesse gemäß Lemma 3.12 Gleichung (3.9) identisch ist, folgt damit auch die Gleichheit der Erwartungswerte:

$$(3.17) = \beta \cdot \mathbb{E}\left(\sum_{t=0}^{\infty} \beta^t \cdot l(\widetilde{X}_t, \widetilde{u}_t)\right) = \beta \cdot J_\infty(X_1(\omega), \widetilde{u}_t) \leq \beta \cdot V_\infty(X_1(\omega)) \quad (3.18)$$

Weiter ist $X_1(\omega) = f(x_0, u_0, Z_0(\omega))$ für alle $\omega \in \Omega$. Mit (3.17) und (3.18) folgt:

$$\mathbb{E}_1\left(\sum_{t=1}^{\infty} \beta^t \cdot l(X_t, u_t)\right) \leq \beta \cdot V_\infty(f(x_0, u_0, Z_0))$$

Damit folgt dann für Gleichung (3.16):

$$\begin{aligned} J_\infty(x_0, \mathbf{u}) &\leq l(x_0, u_0) + \mathbb{E}\left\{\beta \cdot V_\infty(f(x_0, u_0, Z_0))\right\} \\ &= \mathbb{E}\left\{l(x_0, u_0) + \beta \cdot V_\infty(f(x_0, u_0, Z_0)\right\} \\ &\leq \sup_{u \in U} \mathbb{E}\left\{l(x_0, u) + \beta \cdot V_\infty(f(x_0, u, Z_0))\right\} \end{aligned} \quad (3.19)$$

Nach Vorraussetzung gilt die Ungleichung (3.19) für alle zulässigen $\mathbf{u}$ und daher auch für

$$V_\infty(x_0) = \sup_{u \in \mathcal{U}(x_0)} J_\infty(x_0, \mathbf{u}) \quad (3.20)$$

Zu 1.(b)
Gegeben sei $\epsilon > 0$ und ein $x_0 \in \mathbb{R}^n$.
Zu jedem $x \in \mathbb{R}^n$ sei ein Kontrollprozess $\mathbf{u}^x \in \mathcal{U}(x)$ gewählt, der

$$J_\infty(x, \mathbf{u}^x) \geq V_\infty(x) - \epsilon \quad (3.21)$$

erfüllt. Außerdem sei ein Kontrollwert $u^* \in U$ gewählt, für den

$$\mathbb{E}\left\{l(x_0, u^*) + \beta \cdot V_\infty(f(x_0, u^*, Z_0))\right\} \geq \sup_{u \in U} E\left\{l(x_0, u) + \beta \cdot V_\infty(f(x_0, u, Z_0))\right\} - \epsilon \quad (3.22)$$

gilt. Dann wählt man den Kontrollprozess $\mathbf{u} \in \mathcal{U}(x_0)$ mit $u_0 = u^*$ so, dass nach Lemma 3.11 $\widetilde{\mathbf{u}} = \mathbf{u}\&_1\mathbf{u}^x$ gesetzt werden kann. $\widetilde{X}_t = X_t(x_0, \widetilde{\mathbf{u}})$ bezeichne hierbei die

dazugehörige Lösung. Dann gilt:

$$\begin{aligned} V_\infty(x_0) &= \sup_{\mathbf{u}\in\mathcal{U}(x_0)} J_\infty(x_0,\mathbf{u}) = \sup_{\mathbf{u}\in\mathcal{U}(x_0)} \mathbb{E}\left\{\sum_{t=0}^{\infty}\beta^t\cdot l(X_t,u_t)\right\} \\ &= \sup_{\mathbf{u}\in\mathcal{U}(x_0)} \mathbb{E}\left\{l(x_0,u_0)+\sum_{t=1}^{\infty}\beta^t\cdot l(X_t,u_t)\right\} \\ &\geq \mathbb{E}\left\{l(x_0,\widetilde{u}_0)+\sum_{t=1}^{\infty}\beta^t\cdot l(\widetilde{X}_t,\widetilde{u}_t)\right\} \\ &= l(x_0,\widetilde{u}_0)+\mathbb{E}\left\{\mathbb{E}_1\left(\sum_{t=1}^{\infty}\beta^t\cdot l(\widetilde{X}_t,\widetilde{u}_t)\right)\right\} \end{aligned} \tag{3.23}$$

Der letzte Schritt folgt dabei analog zu Gleichung (3.16) aus Beweisteil 1(a). Analog zu (3.17) gilt für alle $\omega\in\Omega$:

$$\mathbb{E}_1\left(\sum_{t=1}^{\infty}\beta^t\cdot l(\widetilde{X}_t,\widetilde{u}_t)\right)(\omega) = \mathbb{E}\left(\sum_{t=1}^{\infty}\beta^t\cdot l(\widetilde{X}_t,\widetilde{u}_t)\,\middle|\,\widetilde{X}_1(\omega)=x\right) \tag{3.24}$$

Unter Anwendung von Lemma 3.11 und Gleichung (3.8) kann gefolgert werden:

$$\begin{aligned} (3.24) &= \beta\cdot\mathbb{E}\left(\sum_{t=0}^{\infty}\beta^t\cdot l(\widetilde{X}_{t+1},\widetilde{u}_{t+1})\,\middle|\,\widetilde{X}_1(\omega)=x\right) \\ &= \beta\cdot\mathbb{E}\left(\sum_{t=0}^{\infty}\beta^t\cdot l(X_t^x,u_t^x)\right) = \beta\cdot J_\infty(x,u_t^x) \end{aligned} \tag{3.25}$$

Dabei ist $x=\widetilde{X}_1(\omega)=X_0^x(\omega)$. Aus der Vorraussetzung (3.21) und $\widetilde{X}_1(\omega) = f(x_0,\widetilde{u}_0,Z_0(\omega))$ für alle $\omega\in\Omega$ folgt dann:

$$\beta\cdot J_\infty(x,u_t^x)\geq\beta\cdot(V_\infty(x)-\epsilon)=\beta\cdot(V_\infty(f(x_0,\widetilde{u}_0,Z_0(\omega)))-\epsilon) \tag{3.26}$$

Einsetzen der Ergebnisse von (3.24), (3.25) und (3.26) in (3.23) und die aus der Definition von $\widetilde{\mathbf{u}}=\mathbf{u}\&_1\mathbf{u}^x$ folgenden Gleichheit $\widetilde{u}_0=u_0$ ergibt:

$$V_\infty(x_0)\geq\mathbb{E}\left\{l(x_0,u_0)+\beta\cdot V_\infty(f(x_0,u_0,Z_0(\omega)))\right\}-\beta\cdot\epsilon \tag{3.27}$$

Unter Verwendung von Gleichung (3.22) und $u_0=u^*$ erhält man:

$$V_\infty(x_0)\geq\sup_{u\in U}\mathbb{E}\left\{l(x_0,u)+\beta\cdot V_\infty(f(x_0,u,Z_0(\omega)))\right\}-(1+\beta)\cdot\epsilon \tag{3.28}$$

Da $\epsilon>0$ beliebig gewählt war, folgt die Ungleichung

$$V_\infty(x_0)\geq\sup_{u\in U}\mathbb{E}\left\{l(x_0,u)+\beta\cdot V_\infty(f(x_0,u,Z_0(\omega)))\right\} \tag{3.29}$$

Zu 2.
Um Gleichung 3.14 nachzuweisen, bedient man sich der vollständigen Induktion, wobei der Fall $T = 1$ schon im Teil 1 dieses Beweises behandelt wurde.
Für $T \to T + 1$ gilt dann:

$$\begin{aligned} V_\infty(x_0) &= \sup_{\mathbf{u}\in\mathcal{U}(x_0)} \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t, u_t) + \beta^T \cdot V_\infty(X_T)\right\} \\ &= \sup_{\mathbf{u}\in\mathcal{U}(x_0)} \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t, u_t) + \beta^T \cdot \sup_{\widetilde{u}\in U} \mathbb{E}_T\left\{l(X_T, \widetilde{u}) + \beta \cdot V_\infty(f(X_T, \widetilde{u}, Z_T))\right\}\right\} \\ &= \sup_{\mathbf{u}\in\mathcal{U}(x_0)} \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t, u_t) + \beta^T \cdot (l(X_T, u_T) + \beta \cdot V_\infty(f(X_T, u_T, Z_T)))\right\} \\ &= \sup_{\mathbf{u}\in\mathcal{U}(x_0)} \mathbb{E}\left\{\sum_{t=0}^{T} \beta^t \cdot l(X_t, u_t) + \beta^{T+1} \cdot V_\infty(X_{T+1})\right\} \end{aligned}$$

Zu 3.
Um die Eindeutigkeit zu zeigen, genügt es für zwei die Transversalitätsbedingung und die Gleichung (3.14) erfüllende Funktionen $W_1, W_2 : \mathbb{R}^n \to \mathbb{R}$ die Ungleichung

$$W_1(x) \leq W_2(x) \tag{3.30}$$

für alle $x \in \mathbb{R}^n$ zu zeigen. Damit folgt dann die Behauptung durch Anwendung von (3.30) auf $W_1 = V_\infty$, $W_2 = W$ und $W_1 = W$, $W_2 = V_\infty$.

Gegeben sei ein $\epsilon > 0$ und $x_0 \in \mathbb{R}^n$.
Da W_1 (3.14) erfüllt, existieren Kontrollfunktionen u^T mit zugehörigen Lösungen X_t^T, so dass gilt:

$$W_1(x_0) \leq \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t^T, u_t^T) + \beta^T \cdot W_1(X_T^T)\right\} + \epsilon \tag{3.31}$$

Daraus folgt:

$$\liminf_{T\to\infty} \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t^T, u_t^T) + \beta^T \cdot W_1(X_T^T)\right\} \geq W_1(x_0) - \epsilon > \infty \tag{3.32}$$

W_2 erfüllt ebenfalls (3.14). Daher gilt:

$$\begin{aligned} W_2(x_0) &= \sup_{\mathbf{u}\in\mathcal{U}(x_0)} \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t, u_t) + \beta^T \cdot W_2(X_T)\right\} \\ &\geq \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t^T, u_t^T) + \beta^T \cdot W_2(X_T^T)\right\} \end{aligned} \tag{3.33}$$

Daraus folgt:

$$\limsup_{T\to\infty} \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t^T, u_t^T) + \beta^T \cdot W_2(X_T^T)\right\} \leq W_2(x_0) < +\infty \tag{3.34}$$

Mit den Gleichungen (3.31) und (3.33) und der Linearität des Erwartungswertes folgt für alle $T \in \mathbb{N}$:

$$\begin{aligned} & W_1(x_0) - W_2(x_0) \\ \leq\ & \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t^T, u_t^T) + \beta^T \cdot W_1(X_T^T)\right\} + \epsilon - \mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t \cdot l(X_t^T, u_t^T) + \beta^T \cdot W_2(X_T^T)\right\} \\ =\ & \mathbb{E}\left\{\beta^T \cdot W_1(X_T^T)\right\} - \mathbb{E}\left\{\beta^T \cdot W_2(X_T^T)\right\} + \epsilon \end{aligned} \tag{3.35}$$

Damit folgt dann, da W_1 und W_2 wegen (3.32) bzw. (3.34) die Transversalitätsbedingung erfüllen:

$$\begin{aligned} W_1(x_0) - W_2(x_0) &\leq \limsup_{T\to\infty} \left(\mathbb{E}\left\{\beta^T \cdot W_1(X_T^T)\right\} - \mathbb{E}\left\{\beta^T \cdot W_2(X_T^T)\right\}\right) + \epsilon \\ &\leq \underbrace{\limsup_{T\to\infty} \mathbb{E}\left\{\beta^T \cdot W_1(X_T^T)\right\}}_{\leq 0} - \underbrace{\liminf_{T\to\infty} \mathbb{E}\left\{\beta^T \cdot W_2(X_T^T)\right\}}_{\geq 0} + \epsilon \leq \epsilon \end{aligned}$$

Da $\epsilon > 0$ beliebig gewählt war, folgt die Ungleichung (3.30). □

3.1.4.3 Charakterisierung der optimalen Kontrollprozesse

Satz 3.18 *Für alle $x \in \mathbb{R}^n$ sei das Supremum im Optimalitätsprinzip in Abschnitt 3.1.4.1 ein Maximum.*
Dann existiert eine Funktion $F_\infty : \mathbb{R}^n \to U$, so dass

$$\sup_{u\in U} \mathbb{E}\left\{l(x,u) + \beta \cdot V_\infty(f(x,u,Z_0))\right\} = \mathbb{E}\left\{l(x, F_\infty(x)) + \beta \cdot V_\infty(f(x, F_\infty(x), Z_0))\right\}$$

gilt. Mit der Funktion F_∞ wird für jeden Anfangswert x_0 induktiv die optimale Lösung X_t definiert:

$$X_0 = x_0, \quad X_{t+1} = f(X_t, F_\infty(X_t), Z_t)$$

Für den dazugehörigen optimalen Kontrollprozess $\boldsymbol{u}^\infty = (u_t^\infty, t = 0, 1, ...)$, $u_t^\infty = F_\infty(X_t)$ gilt

$$J_\infty(x_0, u^\infty) = V_\infty(x_0),$$

wobei $V_\infty(x_0)$ die zweite und $J_\infty(x_0, u^\infty)$ die erste Transversalitätsbedingung erfüllt.

Für den Beweis sei auf [6] Grüne, 2007, S.35, verwiesen.
Aus dem Satz folgt, dass die optimalen Kontrollprozesse eine echte Teilmenge der zulässigen Kontrollprozesse $u_t = h_t(X_0, ..., X_t)$ sind, da diese von der aktuellen Zeit und der gesamten Vergangenheit des Prozesses abhängen. Dabei hängt der optimale Kontrollprozess $u_t = F_\infty(X_t)$ nur vom aktuellen Zustand X_t und nicht direkt von der Zeit ab. Weiter gilt das Ergebnis des Satzes auch unter Beschränkungen, wenn diese in die Bestimmung von F_∞ mit einfließen.

3.2 Diskretisierung

In diesem Abschnitt soll gezeigt werden, wie das Prinzip der dynamischen Optimierung numerisch umgesetzt werden kann. Zuerst wird die Umsetzung für den endlichen Zeithorizont gezeigt und dann die Ergebnisse auf den unendlichen angewendet.

3.2.1 Diskretisierung in der Zeit

Da das Ausgangsproblem (3.15) schon diskret in der Zeit ist, muss nur noch überlegt werden, wie (3.15) implementiert werden kann. Für deterministische Probleme ist das einfachste Verfahren dazu das Eulerverfahren. Dieses kommt auch hier zur Anwendung, nur mit dem Unterschied, dass der stochastische Einfluss mit beachtet werden muss.

Definition 3.19 (Stochastisches Eulerverfahren)
Gegeben seien zwei deterministische Funktionen $a, b : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ und die folgende Ito-stochastische Differentialgleichung

$$dX(t) = a(t, X(t))dt + b(t, X(t))dW(t)^1, \tag{3.36}$$

wobei $X \in \mathbb{R}^n$ und W ein Wiener Prozess ist.
Dann ist das stochastische Eulerverfahren gegeben durch

$$X(t+h) = X(t) + h \cdot a(t, X(t)) + \Delta W(t) \cdot b(t, X(t)) \tag{3.37}$$

für eine Schrittweite $h > 0$ mit $\Delta W(t) = W(t+h) - W(t)$.

Wie das stochastische Eulerverfahren algorithmisch umgesetzt wird, wird im Abschnitt 4.2 erläutert.

Für eine detailierte Einführung in stochastische Differentialgleichungen und die Herleitung des stochastischen Eulerverfahrens sei auf [5] Grüne, 2008, S.87-103 verwiesen.

[1] Dies ist nur eine Schreibweise für

$$X(t) = X(t_0) + \int_{t_0}^{t} a(t, X(t))dt + \int_{t_0}^{t} b(t, X(t))dW(t)$$

3.2.2 Diskretisierung im Raum

Ansatz

Bei dem im Folgenden vorgestellten Ansatz verwendet man einen endlichdimensionalen Funktionenraum $\mathcal{W}$ um die Ergebnisse des Optimalitätsprinzips in dem Rechner speichern zu können. Es wird dann nur noch eine Approximation $\widetilde{V}_\infty \in \mathcal{W}$ der optimalen Wertefunktion berechnet.
Vorteil dieses Ansatzes ist, dass er auch auf Probleme komplexer nichtlinearer Dynamik angewandt werden kann. Der größte Nachteil ist dagegen der mit der Dimension n des Zustandsraums schnell anwachsende numerische Aufwand. Daher sind die Algorithmen nur für niedrige Dimensionen effizient.
Es sollen zunächst einige grundlegende Begriffe definiert werden.

Definition 3.20

1. *$F(A,B) := \{W : A \to B\}$ bezeichne die Menge aller Funktionen von einer Menge A in eine Menge B.*

2. *Der Projektionsoperator $\pi : F(A,B) \to \mathcal{W}$ berechne zu einer beliebigen Funktion W die numerische Approximation $\widetilde{W} = \pi W$ im Funktionenraum $\mathcal{W}$.*

3. *Für die rechte Seite des Optinalitätsprinzip definiert man die folgenden Operatoren:*

   (a) $T_u : F(\mathbb{R}^n, \mathbb{R}) \to F(\mathbb{R}^n, \mathbb{R})$

   $$T_u(W)(x) := \mathbb{E}\left\{l(x,u) + \beta \cdot W(f(x,u,Z))\right\} \tag{3.38}$$

   (b) $T : F(\mathbb{R}^n, \mathbb{R}) \to F(\mathbb{R}^n, \mathbb{R})$

   $$T(W)(x) := \sup_{u \in U(x)} T_u(W)(x) \tag{3.39}$$

Folgerung 3.21 *Mit Definition 3.20 folgt für das Optimalitätsprinzip $V_\infty = T(V_\infty)$. Damit kann nun der folgende iterative Algorithmus definiert werden:*

$$\widetilde{V}_\infty = \pi T(\widetilde{V}_\infty). \tag{3.40}$$

Bemerkung 3.22 *Zur algorithmischen Berechnung des approximativen optimalen Feedbacks gibt es zwei Möglichkeiten. Es kann einerseits in der Iteration mitberechnet und als $\pi\widetilde{F}_\infty$ approximiert gespeichert werden. Die numerisch bessere Lösung ist dagegen für einen gegebenen Zustand x den optimalen Kontrollwert $F_\infty(x)$ "online" aus*

$$\widetilde{F}_\infty(x) = \arg\max_{u \in U(x)} T_u(\widetilde{V}_\infty)(x) \tag{3.41}$$

zu berechnen. Vorteile dieser Vorgehensweise sind die Vermeidung des Approximationsfehlers bei der Projektion πF_∞ und das Sparen von Speicherplatz. Ein Nachteil wäre der Rechenaufwand bei der Berechnung von $\arg\max$*, welcher aber vertretbar ist, da* $\arg\max$ *schnell berechenbar ist.*

Im folgenden Abschnitt soll vorgestellt werden, wie man den Operator T(W) numerisch auswerten kann.

Auswertung von $T(W)$

Für die Auswertung von T(W) muss man zwei Teilprobleme lösen:

1. Auswertung von $T_u(W)(x)$
2. Maximierung von $T_u(W)(x)$ über u

Bei 1. besteht die einzige Schwierigkeit darin, den Erwartungswert in $T_u(W)(x)$ zu berechnen, da l, f und W bekannt und numerisch leicht auswertbar sind.
Bei einer kontinuierlichen Zufallsvariable Z muss man zur Berechnung des Erwartungswerts die Dichtefunktion f_Z kennen und

$$\int_{Z(\Omega)} (l(x,u) + \beta \cdot W(f(x,u,z))) \cdot f_Z(z) dz \tag{3.42}$$

zum Beispiel mit der Trapezregel numerisch integrieren. Der bei der Modellierung des Portfolioproblems in Kapitel 4 relevante Fall ist der der Zufallsvariable Z mit endlich vielen Werten $z_1, ..., z_m$. Dann wird der Erwartungswert über folgende Summe berechnet:

$$\sum_{i=1}^{m} (l(x,u) + \beta \cdot W(f(x,u,z_i))) \cdot P_Z(\{z_i\}) \tag{3.43}$$

Bei der Maximierung im 2. Schritt wählt man zunächst den Kontrollwertebereich $U(x)$ kompakt und dann eine endliche Menge $\widetilde{U}(x) \subset U(x)$, weil es numerisch einfacher ist über einer endlichen Menge zu maximieren. T wird dann durch

$$\widetilde{T}(W)(x) = \max_{\widetilde{u} \in \widetilde{U}(x)} T_{\widetilde{u}}(W)(x) \tag{3.44}$$

approximiert.

Lemma 3.23 *Wenn die Abbildung $u \mapsto T_u(W)(x)$ Lipschitz-stetig mit der Konstante L_u ist. Dann gilt*

$$\left| \widetilde{T}(W)(x) - T(W)(x) \right| \leq L_u \cdot \epsilon_u \quad mit\, \epsilon_u := \max_{u \in U(x)} \max_{\widetilde{u} \in \widetilde{U}(x)} \|\widetilde{u} - u\| . \tag{3.45}$$

Für den Beweis siehe [6] Grüne, 2007, S.39. Die Lipschitz-Stetigkeit erhält man zum Beispiel, wenn l und f in u sowie W in x Lipschitz-stetig sind.
Die eigentliche Diskretisierung im Raum erfolgt im folgenden Schritt, der Berechnung von $\pi T(W)$.

Berechnung von $\pi T(W)$

Im Folgenden wird dargelegt, dass es für die Berechnung von $\pi T(W)$ ausreicht die Werte $T(W)(x) \in \mathbb{R}$ für gegebene Punkte $x \in \mathbb{R}^n$ zu berechnen.
Dazu approximiert man V_τ durch ein Element $\widetilde{V}_\tau$ eines endlich-dimensionalen Funktionenraums. Zur Vereinfachung der Darstellung wird im Folgenden die Approximation nur für die Dimension n=2 auf einer rechteckigen Menge $X \subset \mathbb{R}^2$ betrachtet.
Es sollen zunächst die wichtigsten Begriffe definiert werden:

Definition 3.24 (Rechteckgitter)
Gegeben sei eine kompakte Menge $X \subset \mathbb{R}^2$ durch $X = [a_1, b_1] \times [a_2, b_2]$ mit $a_1 < b_1$ und $a_2 < b_2$.
Dann ist ein (regelmäßiges) Rechteckgitter Γ auf X gegeben durch eine Menge von Rechtecken R_i, $i = 0, ..., P-1$, $P = P_1 \cdot P_2$, mit den Kantenlängen $k_1 = (b_1 - a_1)/P_1$ und $k_2 = (b_2 - a_2)/P_2$ und den Eigenschaften

$$\bigcup_{i=0}^{P-1} R_i = X \quad und \quad intR_i \cap intR_j = \emptyset \quad \forall i, j = 0, ..., P-1,\ i \neq j.$$

Dabei bezeichne E_i, $i = 0, ..., N-1$, $N = (P_1 + 1) \cdot (P_2 + 1)$ die Eckpunkte bzw. Knoten des Gitters. Den maximalen Durchmesser k eines Rechtecks berechnet man durch $k = \sqrt{k_1^2 + k_2^2}$.

Ein Beispielgitter ist in Abbildung 3.2 dargestellt.

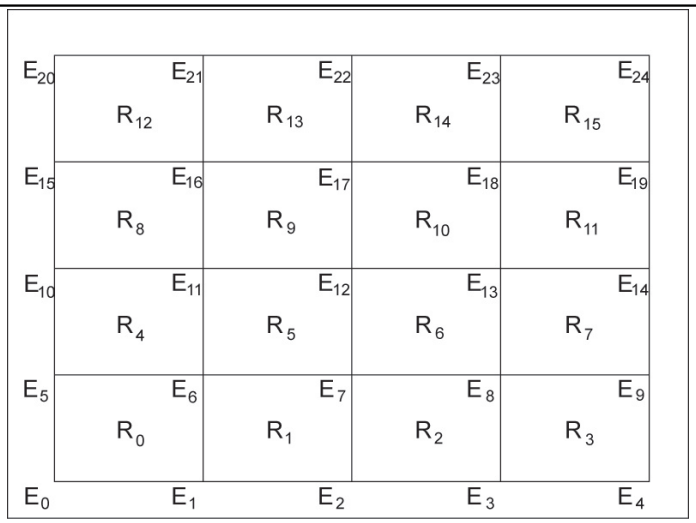

Abbildung 3.2: Aufbau eines Gitters

Der endlich-dimensionale Funktionenraum wird folgendermaßen definiert:

Definition 3.25

1. *Es sei $A \subset \mathbb{R}^2$. Eine Funktion $f : A \to \mathbb{R}$ heißt affin bilinear, falls es Koeffizienten $\alpha_0, ..., \alpha_3 \in \mathbb{R}$ gibt, so dass für alle $x = (x_1, x_2)^T \in A$ die Identität $f(x) = \alpha_0 + \alpha_1 \cdot x_1 + \alpha_2 \cdot x_2 + \alpha_3 \cdot x_1 \cdot x_2$ gilt.*

2. *Sei $X \subset \mathbb{R}^2$ und Γ gegeben wie in Definition 3.24. Dann ist der Raum der stetigen und stückweise affin bilinearen Funktionen auf X bezüglich Γ definiert als* $\mathcal{W} := \{f : X \to \mathbb{R} \mid \ f \text{ ist stetig und } f|_{R_i} \text{ ist affin bilinear für jedes } i = 0, ..., P-1\}$.

Bemerkung 3.26 (Eigenschaften des Funktionenraumes $\mathcal{W}$)

1. *Es kann jede Funktion $f \in \mathcal{W}$ eindeutig durch ihre Werte $f(E_i)$ in den Eckpunkten des Gitters bestimmt werden.*

2. *Für eine Funktion $f \in \mathcal{W}$ lässt sich jeder Punkt aus $f|_{R_i}$ aus den zu R_i gehörigen Eckpunkten bestimmen. Für eine formale Darstellung dieser Punkte sowie den Beweis dazu siehe [6] Grüne, 2007, S. 41.*

3. *$\mathcal{W}$ ist ein N-dimensionaler Vektorraum über $\mathbb{R}$.*

Damit kann nun der Projektionsoperator aus Definition 3.20 formal wie folgt definiert werden.

Definition 3.27 (Projektionsoperator)
Der Projektionsoperator $\pi : F(X, \mathbb{R}) \to \mathcal{W}$ ist definiert als

$$\pi W := \widetilde{W} \in \mathcal{W} \quad mit \quad \widetilde{W}(E_i) = W(E_i) \quad für \quad alle \quad i = 0, ..., N-1. \tag{3.46}$$

Werteiteration

Jetzt soll ein Verfahren zur numerischen Lösung des stochastischen dynamischen Optimierungsproblems, die Werteiteration, vorgestellt werden. Dabei approximiert man $\widetilde{V}_\infty$ durch $\widetilde{V}_T$ für $T = 0, 1, 2, ...$ bis zu einem Abbruchkriterium der Form

$$\left\| \widetilde{V}_T - \widetilde{V}_{T-1} \right\|_X := \sup_{x \in X} \left| \widetilde{V}_T(x) - \widetilde{V}_{T-1}(x) \right|_X \leq \epsilon. \tag{3.47}$$

Das folgende Lemma zeigt nun den Zusammenhang zwischen endlichem und unendlichem Zeithorizont:

Lemma 3.28
Gegeben sei ein optimales Steuerungsproblem gemäß Gleichung (3.15) mit beliebigen laufenden Kosten l, Diskontfaktor $\beta < 1$ und Endkosten $L \equiv 0$. Weiter gelte für die optimale Wertefunktion auf unendlichem Zeithorizont: $\|V_\infty\|_\infty \leq C$ für ein $C \in \mathbb{R}$. Dann gilt für jedes $T \in \mathbb{N}_0$

$$\|V_T - V_\infty\|_\infty \leq \beta^T \cdot C. \tag{3.48}$$

Beweis
Das Lemma wird mit vollständiger Induktion bewiesen:
Für T=0 gilt mit $V_0 = L \equiv 0$: $\|V_0 - V_\infty\|_\infty = \|V_\infty\|_\infty \leq C = \beta^0 \cdot C$
Sei nun (3.48) für T erfüllt. Dann gilt für den Fall $T \to T+1$:

$$\begin{aligned}
\|V_{T+1} - V_\infty\|_\infty &= \sup_{x \in \mathbb{R}^n} |V_{T+1}(x) - V_\infty(x)| \\
&= \sup_{x \in \mathbb{R}^n} \Big| \sup_{u \in U} \{\mathbb{E}\left[l(x,u) + \beta \cdot V_T(f(x,u,z))\right]\} \\
&\quad - \sup_{u \in U} \{\mathbb{E}\left[l(x,u) + \beta \cdot V_\infty(f(x,u,z))\right]\} \Big| \qquad (3.49) \\
&\leq \beta \cdot \sup_{x \in \mathbb{R}^n} \sup_{u \in U} \Big| \mathbb{E}\left[V_T(f(x,u,z)) - V_\infty(f(x,u,z))\right] \Big| \qquad (3.50) \\
&\leq \beta^{T+1} \cdot C \qquad (3.51)
\end{aligned}$$

Dabei wurde die für beliebige Funktionen $a, b : U \to \mathbb{R}$ geltende Ungleichung

$$\Big| \sup_{u \in U} a(u) - \sup_{u \in U} b(u) \Big| \leq \sup_{u \in U} \Big| a(u) - b(u) \Big| \tag{3.52}$$

benutzt um von (3.49) zu (3.50) zu kommen (Für den Beweis siehe [6] Grüne, 2007, S.25/26). Um von (3.50) zu (3.51) zu kommen wurde die Induktionsvoraussetzung, dass die Behauptung schon für T gilt, verwendet. □

Unter den Voraussetzungen des Lemmas 3.28 folgt, dass $V_T \to V_\infty$ gilt. Damit kann man die numerische Approximation theoretisch als

$$\widetilde{V}_\infty = \lim_{T \to \infty} \widetilde{V}_T \tag{3.53}$$

definieren, welches die eindeutige Lösung der Bellman-Gleichung

$$\widetilde{V}_\infty = \pi \widetilde{T} \widetilde{V}_\infty \tag{3.54}$$

ist. Mit der Definition 3.27 des Projektionsoperators und Gleichung (3.53) kann das Iterationsverfahren aus Folgerung 3.21 wie folgt beschrieben werden. Um die Funktion $\widetilde{W} = \pi T(W)$ zu berechnen und im Rechner abzuspeichern genügt es die Werte in den Eckpunkten des Gitters zu berechnen, also $\widetilde{W}(E_i) = \pi T(W)(E_i)$. (3.40) kann (unter Berücksichtigung von (3.44)) numerisch folgendermaßen ausgewertet werden:

$$\begin{aligned} \widetilde{V}_0(E_i) &:= 0 \\ \widetilde{V}_\tau(E_i) &= \pi\widetilde{T}(\widetilde{V}_{\tau-1})(E_i), \quad i = 0, ..., N-1 \quad \textit{und} \quad \tau = 1, 2, ... \end{aligned} \tag{3.55}$$

Wahl der Menge X

Dieses Approximationsverfahren benötigt für das Rechteckgitter eine kompakte Menge, was in der Praxis nicht leicht umzusetzen ist. X ist dabei der Definitionsbereich von $\widetilde{V}_\tau$. Damit $\widetilde{V}_\tau(f(x,u,z))$ für alle x, u und z definiert ist, muss $f(x,u,z) \in X$ für alle $x \in X$, $u \in U(x)$ und $z \in Z(\Omega)$ gelten.
Zum Beispiel kann $Z(\Omega)$ für eine Gauß-verteilte Zufallsvariable unbeschränkt sein. Dagegen kann hilfreich sein $Z(\Omega)$ oder Ω einzuschränken, wobei selbst dann die Wahl einer kompakten Menge oft nicht möglich ist. Außerdem darf bei Problemen auf unendlichem Zeithorizont das Berechnungsgebiet nicht wachsen.
Daher muss man bei der Modellierung darauf achten, dass die optimale Wertefunktion auf einer kompakten nicht wachsenden Menge X ausgewertet werden kann.

Diskretierungsfehler

In [6] Grüne, 2007, S.43-46 wird der Diskretierungsfehler für den endlichen Horizont vorgestellt und bewiesen. Die Mengen und Gitter sind da von der Zeit T abhängig, was auf unendlichem Zeithorizont nicht möglich ist. Daher wird im Folgenden der Satz auf eine Menge X und ein Gitter Γ mit Durchmesser k spezialisiert. Daher ist der Beweis aus [6] Grüne, 2007, S.46 darauf anwendbar.

Satz 3.29 (Diskretierungsfehler)
Gegeben sei eine Menge X mit $f(x,u,z) \in X$ für alle $x \in X$, $u \in U(x)$ und $z \in Z(\Omega)$. Weiter sei auf X ein Gitter Γ mit dem Durchmesser k und diskretisierte Kontrollwertemenge $\widetilde{U}(x)$ mit ϵ_u aus (3.45) gegeben. $T_u V_\tau(x)$ sei Lipschitz mit Konstante $L_{u,\tau}$. Weiter sei
(i) V_τ ist Lipschitz-stetig mit Konstanten L_τ, $\tau = 0, ..., T$ und
(ii) V_τ ist zweimal stetig differenzierbar mit zweiter Ableitung beschränkt durch C_τ, $\tau = 0, ..., T$
Dann gilt für $\tau = 0, ..., T$
(i) $\left\|\widetilde{V}_\tau - V_\tau\right\| \leq \sum_{t=0}^{\tau-1} \beta^{\tau-t-1}(L_{u,t} \cdot \epsilon_u + L_{t+1} \cdot k)$ *und*
(ii) $\left\|\widetilde{V}_\tau - V_\tau\right\| \leq \sum_{t=0}^{\tau-1} \beta^{\tau-t-1}(L_{u,t} \cdot \epsilon_u + C_{t+1} \cdot k^2/2)$.
Dann ist der Fehler in (i) von der Ordnung $\mathcal{O}(\epsilon_u + k)$und in (ii) von der Ordnung $\mathcal{O}(\epsilon_u + k^2)$.

Für den unendlichdimensionalen Fall gelte für die Konstanten:
$L_{u,\tau} \leq L_{u,\infty}$, $L_\tau \leq L_\infty$ und $C_\tau \leq C_\infty$ für alle $\tau = 1, 2, ...$
Dann gilt

$$\begin{aligned} \left\| \widetilde{V}_\infty - V_\infty \right\| &= \left\| \widetilde{V}_\infty - \widetilde{V}_T + \widetilde{V}_T - V_T + V_T - V_\infty \right\| \\ &\leq \left\| \widetilde{V}_\infty - \widetilde{V}_T \right\| + \| V_T - V_\infty \| + \left\| \widetilde{V}_T - V_T \right\|, \end{aligned}$$

wobei der erste und zweite Term nach Lemma 3.28 für $T \to \infty$ gegen 0 konvergiert.
Für den dritten Term gilt für $T \to \infty$ für den Fall
(i) $\left\| \widetilde{V}_T - V_T \right\| \leq \frac{1}{1-\beta} \cdot (L_{u,\infty} \cdot \epsilon_u + L_\infty \cdot k)$ und für den Fall
(ii) $\left\| \widetilde{V}_T - V_T \right\| \leq \frac{1}{1-\beta} \cdot (L_{u,\infty} \cdot \epsilon_u + L_\infty \cdot k^2/2)$,
d.h. auch für den unendlichdimensionalen Fall ist die Ordnung des Fehlers wie in Satz 3.29.

Kapitel 4

Mathematische Modellierungen des Portfolioproblems

In diesem Kapitel soll das Portfolio-Optimimierungsproblem mathematisch formuliert werden. In [9] Korn und Korn, 2001, wird das Portfolioproblem folgendermaßen definiert:[1]

> „Das Portfolioproblem eines Investors in einem Finanzmarkt besteht darin, zu gegebenem Startkapital $x > 0$ eine optimale Investment- und Konsumstrategie zu bestimmen. Das heißt, er muss festlegen [...], **wie viele** Anteile er von **welchem** Wertpapier **wann** halten muss und **wie viel** Vermögen er **wann** konsumieren darf, um seinen **Nutzen** aus **Konsum** im Zeitraum $[0, T]$ [...] zu maximieren."

Bemerkung 4.1 *Zum einfacheren Verständnis und um Verwechslungen bei Betrachtung des Quellcodes zu vermeiden, beginnt im Folgenden die Indizierung immer bei 0, wie es in der Programmiersprache C vorgegeben ist, und nicht bei 1.*
Des Weiteren steht bei z.B. $x_1(4)$ *der Index* 1 *für eine Komponente der zweidimensionalen Variable* x *und die* 4 *in Klammern für den Iterationszähler.*
In Kapitel 5 und 6 wird für alle Variablen des Programms die exakte C-Schreibweise $x[1]$ *genommen. Dabei ist die* 1 *in eckigen Klammern die Komponente, ein Iterationszähler ist dort nicht mehr nötig.*

4.1 Ein deterministisches Modell

Das deterministische Modell orientiert sich an das in [14] Öhrlein, 2006, vorgestellte dritte Modell. Die wesentlichen Änderungen betreffen die Art der verwendeten Zinsen: in [14] wurden geschätzte harmonische Schwingungen, modelliert durch einfache Sinus-Funktionen unterschiedlicher Amplitude und Schwingungsdauer, verwendet.

[1] Auf unendlichem Zeithorizont gilt $T \to \infty$ und es gibt kein Endvermögen.

Die in dieser Arbeit verwendeten Zinsen werden in Abschnitt 4.1.2 vorgestellt.

4.1.1 Grundlagen des Modells

Das Modell hat zwei Vermögenswerte:

- Eine risikoreichere Anlage mit höherer erwarteter Rendite r_e:
  Dazu zählen Aktien, ein Aktienindex wie z.B. der DAX oder der Schweizer Aktienindex oder ein Investmentfond.

- Eine Anlage geringeren Risikos und niedrigerer erwarteter Rendite r_f:
  Dazu zählen Obligationen oder auch ein Obligationsindex.

Des Weiteren wird davon ausgegangen, dass es keine Transaktionskosten (bezüglich der Umschichtung der Anlagen sowie der Liquidierung der Anlagen für den Konsum) gibt.

Anmerkung zu den Zinsverläufen der Obligationen

Bei Obligationen handelt es sich um festverzinsliche Wertpapiere, d.h. der Nominalwert wird über die gesamte Laufzeit mit einem festen Zinssatz verzinst und am Ende der Laufzeit zurückgezahlt. Wenn jetzt der Marktzinssatz (für ein neu emittiertes Wertpapier) z.B. steigt, dann fällt aufgrund der niedrigeren Nachfrage der Marktpreis für Wertpapiere mit geringerer Verzinsung. Dieser Kursverlust kann aber durch Verkauf des Wertpapiers niedrigeren Zinsesniveaus und der Wiederanlage zu höherem Marktzinssatz überkompensiert werden. Für einen fallenden Marktzinssatz beobachtet man den gegenteiligen Effekt.
Daraus kann man schlussfolgern, dass trotz der festen Verzinsung die Rendite des Wertpapiers stark von dem Marktzinssatz abhängt.

Das Kontrollsystem

Es wird nun zuerst ein zweidimensionaler Zustand $X(t) = (x_1(t), x_2(t)) \in \mathbb{R}^2$ gewählt, dessen erste Komponente das gesamte Vermögen des Investors und dessen zweite Komponente die Zeit t modelliert.
Die Kontrolle $u(t) = (u_0(t), u_1(t))^T \in U \subset \mathbb{R}^2$ beschreibt in der Komponente $u_1(t)$ den Anteil des Vermögens, welches zur Zeit t in die risikoreichere Anlage investiert wird, und in $u_0(t)$ den Anteil, der zur Zeit t konsumiert wird.
Damit wird die Lösung des diskretisierten Kontrollsystems induktiv für $t \geq 0$ und die Schrittweite $h > 0$ folgendermaßen definiert:

$$
\begin{aligned}
X(0) &= \begin{pmatrix} x_0(0) \\ x_1(0) \end{pmatrix} \quad und \\
X(t+1) &= f^h(X(t), u(t)) = \begin{pmatrix} f_0^h(X(t), u(t)) \\ f_1^h(X(t), u(t)) \end{pmatrix} \quad mit \\
f_0^h(X(t), u(t)) &= x_0(t) + h \cdot \Bigg(\underbrace{u_1(t) \cdot r_e(t) \cdot x_0(t)}_{Rendite\,der\,risikoreichen\,Anlage} \\
&+ \underbrace{(1 - u_1(t)) \cdot r_f(t) \cdot x_0(t)}_{Rendite\,der\,risikoarmen\,Anlage} - \underbrace{x_0(t) \cdot u_0(t)}_{Konsumausgaben} \Bigg) \\
f_1^h(X(t), u(t)) &= x_1(t) + h
\end{aligned} \tag{4.1}
$$

Ziel

Das mathematische Ziel des Portfolio-Optimierungsproblems ist es nun die optimale Wertefunktion $V_\infty(x)$ zu maximieren.

$$
V_\infty(x) = \max_{\mathbf{u} \in \mathcal{U}_x} J_\infty(x, \mathbf{u}) = \lim_{T \to \infty} \max_{\mathbf{u} \in \mathcal{U}_x} \mathbb{E} \left\{ \sum_{t=0}^{T-1} \beta^t \cdot g(X(t), u(t)) \right\} \tag{4.2}
$$

Dabei ist $\beta < 1$ ein Diskontfaktor und $g(X(t), u(t))$ die Power-Nutzenfunktion, gegeben durch

$$
g(X(t), u(t)) = \frac{(u_0(t) \cdot x_0(t))^{1-\gamma} - 1}{1 - \gamma}. \tag{4.3}
$$

D.h. eine optimale Wertefunktion kann nur durch einen maximalen Nutzen erreicht werden, welcher wiederum nur vom Konsum abhängt. Für den Investor stellt sich die Frage danach, wann er welchen Anteil seines Vermögens konsumiert und wie er den Rest so auf die beiden Anlagemöglichkeiten verteilt, um auch in späteren Perioden konsumieren zu können, wobei durch den Faktor β^t der Nutzen zukünftigen Konsums gemindert wird.

Eindeutigkeit der Lösung durch das Bellman'sche Optimalitätsprinzip

Die Power-Nutzenfunktion $g(X(t), u(t))$ ist für beschränktes $X(t)$ und $u(t)$ (dies ist für die numerischen Tests in Kapitel 6 durch die Optimierung über beschränkte Gitter erfüllt) ebenfalls beschränkt. Wenn $\beta < 1$ gewählt wird, dann ist auch $V_\infty(x)$ beschränkt und damit die Transversalitätsbedingung erfüllt (vergleiche Definition 3.16). Damit ist nach Satz 3.17 die Lösung des Bellman'schen Optimalitätsprinzip eindeutig.

4.1.2 Verwendete Zinsen

Für die verwendeten Zinsen r_e und r_f werden zwei Ansätze verwendet: die Erzeugung von normalverteilten miteinander korrelierten Zinsreihen mittels Zufallszahlen und die Verwendung von historischen Zinsdaten.

Erzeugung mittels Zufallszahlen

Zur Erzeugung der normalverteilten miteinander korrelierten Zufallszahlen wird eine Abwandlung der Box-Muller-Methode aus [8] Jöhnk, 1969, S.18-24, verwendet:
Gegeben seien zwei unkorrelierte, gleichverteilte Zufallszahlen $z_1, z_2 \in [0,1]$, der Korrelationskoeffizient $\rho = \sin\phi$, $\phi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ und die Erwartungswerte μ_1, μ_2 sowie Standardabweichungen σ_1, σ_2. Dann erhält man durch die modifizierte Box-Muller-Methode

$$y_1 = \mu_1 + \sigma_1 \cdot \sqrt{-2 \cdot \ln(z_1)} \cdot \sin(2\pi \cdot z_2) \quad \textit{und} \tag{4.4}$$

$$y_2 = \mu_2 + \sigma_2 \cdot \sqrt{-2 \cdot \ln(z_1)} \cdot \cos(2\pi \cdot z_2 - \phi), \tag{4.5}$$

wobei y_1 (μ_1, σ_1^2)-normalverteilt und y_2 (μ_2, σ_2^2)-normalverteilt mit Korrelationskoeffizient ρ sind.
Die algorithmische Umsetzung erfolgt durch das Programm *Zinserzeugung.c* (siehe Abschnitt 5.2.2)

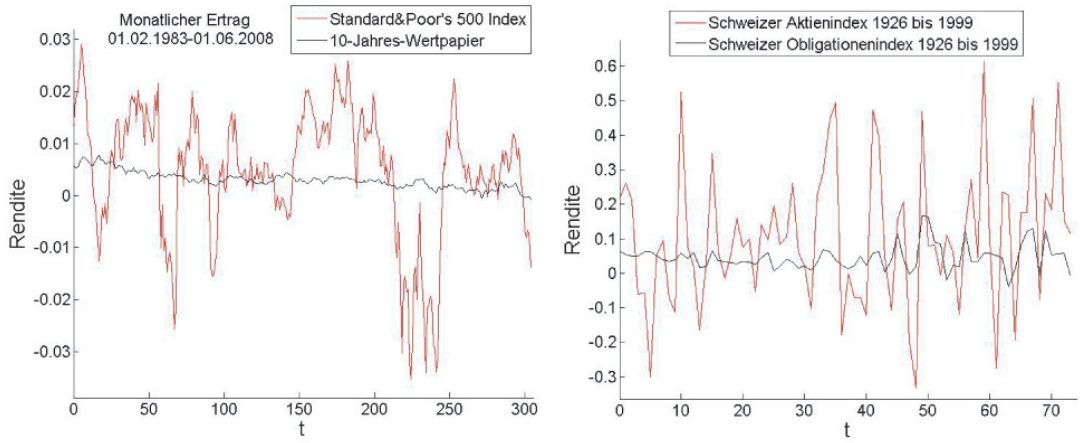

Abbildung 4.1: Graphische Darstellung der historischen Zinsverläufe

Verwendete reale/historische Zinsverläufe

Für eine graphische Darstellung der Zinsverläufe siehe Abbildung 4.1.

1. Monatliche Renditen (Realwerte, also abzüglich der Inflation) von Standard&Poor's 500 Index und 10-Jahres-Wertpapier von 01.02.1983 bis 01.06.2008. (305 Zeitpunkte)

2. Jahresrenditen des Schweizer Aktienindexes und Obligationenindexes von 1926 bis 1999 (74 Zeitpunkte) aus [1] Auckenthaler, 2001, S.53/54

4.2 Erweiterung des Modells durch einen stochastischen Einfluss

4.2.1 Grundlagen des Modells

Es gelten weiterhin die Annahmen des deterministischen Modells. Es wird nun aber das Kontrollsystem zu einem stochastischen Kontrollsystem erweitert. Damit folgt aus (4.1):

$$
\begin{aligned}
X(0) &= \begin{pmatrix} x_0(0) \\ x_1(0) \end{pmatrix} \quad und \\
X(t+1) &= f^h(X(t), u(t), z(t)) = \begin{pmatrix} f_0^h(X(t), u(t), z(t)) \\ f_1^h(X(t), u(t), z(t)) \end{pmatrix} \quad mit \\
f_0^h(X(t), u(t), z(t)) &= x_0(t) + h \cdot \Bigg(\underbrace{u_1(t) \cdot r_e(t) \cdot x_0(t)}_{Rendite\,der\,risikoreichen\,Anlage} \\
&+ \underbrace{(1 - u_1(t)) \cdot r_f(t) \cdot x_0(t)}_{Rendite\,der\,risikoarmen\,Anlage} - \underbrace{x_0(t) \cdot u_0(t)}_{Konsumausgaben} \Bigg) \\
&+ \underbrace{\sigma_{EK} \cdot u_1(t) \cdot x_0(t) \cdot z_0(t)}_{stochastischer\,Einfluss\,auf\,risikoreiche\,Anlage} \\
&+ \underbrace{\sigma_{FK} \cdot (1 - u_1(t)) \cdot x_0(t) \cdot z_1(t)}_{stochastischer\,Einfluss\,auf\,risikoarme\,Anlage} \\
f_1^h(X(t), u(t), z(t)) &= x_1(t) + h
\end{aligned}
\tag{4.6}
$$

Dabei sind $z_0(t)$ und $z_1(t)$ für alle t normalverteilte Zufallszahlen, welche durch den Wiener Prozess erzeugt wurden. Wie stark sich der stochastische Einfluss auf den Anteil des Vermögens, welcher in eine der beiden Anlagen investiert wurde, auswirkt, wird durch die Parameter σ_{EK} für die risikoreiche und σ_{FK} für die risikoarme Anlage beeinflusst.
Im Folgenden soll nun gezeigt werden, wie der stochastische Einfluss algorithmisch umgesetzt werden kann.

4.2.2 Modellierung des stochastischen Einflusses

Ziel dieses Abschnittes ist es, einen Algorithmus für die Approximation des Wiener Prozesses zu finden. Dazu soll dieser zunächst definiert werden.

Definition 4.2 (Wiener Prozess)
Ein an die natürliche Filtration $\mathcal{F}_t$, $t \geq 0$, adaptierter stochastischer Prozess $W(t,\omega)$, $\omega \in \Omega$, $t \geq 0$, auf dem Wahrscheinlichkeitsraum (Ω, Σ, P) heißt Wiener Prozess, falls folgende Bedingungen erfüllt sind:

1. *$W(t,\omega)$ ist Gauß-verteilte Zufallsvariable mit $\mathbb{E}(W(t,\omega)) = 0$ und $Var(W(t,\omega)) = t \ \forall\omega \in \Omega$, d.h. $W(t,\omega) \sim \mathcal{N}(0,t) \ \forall\omega \in \Omega$.*

2. *Für alle $t_2 \geq t_1 \geq 0$ gilt $W(t_2,\omega) - W(t_1,\omega) \sim \mathcal{N}(0, t_2 - t_1) \ \forall\omega \in \Omega$.*

3. *Für alle $s_2 \geq s_1 \geq t_2 \geq t_1$ sind die Inkremente $W(t_2,\omega) - W(t_1,\omega)$ und $W(s_2,\omega) - W(s_1,\omega) \ \forall\omega \in \Omega$ unabhängige Zufallsvariablen.*

Für das stochastische Modell muss der Wiener Prozess nun über ein Gitter approximiert werden. Eine Möglichkeit der Approximation funktioniert über die Erzeugung von $\mathcal{N}(0,h)$-verteilten Zufallsvariablen (h ist dabei die Schrittweite der Diskretisierung).[2] Dieses Verfahren hat aber den Nachteil, dass man sehr viele $\mathcal{N}(0,h)$-verteilte Zufallsvariablen in jedem Gitterpunkt erzeugen muss um eine gute Approximation des normalverteilten stochastischen Prozesses zu erhalten.
Abhilfe schafft dabei die schwache Approximation des Wiener Prozesses[3], da dort in jedem Zeitschritt nur zwei zweipunktverteilte Zufallszahlen $\{x_1, x_2\}$, definiert durch

$$X(\Omega) = \{x_1, x_2\}, \quad \mathbb{P}(\{x_1\}) = \mathbb{P}(\{x_2\}) = \frac{1}{2},$$

erzeugt werden.

Algorithmus 4.3 (Schwache Approximation des Wiener Prozesses)
Gegeben: Schrittweite h, Gitter $\Gamma = \{t_0, ..., t_N\}$ mit $t_i = i \cdot h$
Gesucht: Schwach approximierter Pfad $\widetilde{W}(\cdot,\omega)$ des Wiener Prozesses

1. *Setze die Gitterfunktion $\widetilde{W}(t_0,\omega) = 0$*

2. *Für i=1,...,N*

   (a) *Erzeuge zweipunktverteilte Zufallsvariable $\Delta W_i(\omega)$ mit $y_1 = -\sqrt{h}$ und $y_2 = \sqrt{h}$.*

   (b) *Berechne $\widetilde{W}(t_{i+1},\omega) = \widetilde{W}(t_i,\omega) + \Delta W_i(\omega)$*

[2] Für den Algorithmus dazu siehe [5] Grüne, 2008, S.92/93

[3] vergleiche dazu [3] Camilli und Falcone, 1995. Dort wurde die schwache Approximation für die Berechnung der Hamilton-Jacobi-Bellman-Gleichung verwendet.

Numerisch kann 2.(a) dies wie folgt umgesetzt werden:

1. Für j=1,2
   (a) Erzeuge gleichverteilte Zufallszahl ω auf dem Intervall $[0, 1]$
   (b) Wenn $\omega \leq 0.5$, dann setze $y_j = -\sqrt{h}$, sonst setze $y_j = \sqrt{h}$
2. Berechne $\Delta W_i(\omega) = 0.5 \cdot (y_1 + y_2)$

Mit Hilfe dieser Approximation soll nun das Stochastische Eulerverfahren[4] formuliert werden:

Algorithmus 4.4 (Stochastisches Einschritt-Verfahren für (4.6))
Gegeben: Schrittweite h, Gitter $\Gamma = \{t_0, ..., t_N\}$ mit $t_i = i \cdot h$
Gesucht: Approximierter Pfad $\widetilde{X}(\cdot, \omega)$ für Differenzengleichung in (4.6)

1. *Setze $X_0(t_0) = x_0(t_0)$ und $X_1(0) = x_1(t_0)$*
2. *Für i=1,...,N*
   (a) *Erzeuge zwei zweipunktverteile Zufallsvariablen $\Delta W_i(\omega_0)$ und $\Delta W_i(\omega_1)$ wie in Algorithmus 4.3 in 2.(a).*
   (b) *Berechne*

$$\begin{aligned} X_0(t_{i+1}) &= X_0(t_i) \\ &+ h \cdot (u_1(t_i) \cdot r_e(t_i) \cdot x_0(t_i) + (1 - u_1(t_i)) \cdot r_f(t_i) \cdot x_0(t_i) - x_0(t_i) \cdot u_0(t_i)) \\ &+ \sigma_{EK} \cdot u_1(t_i) \cdot x_0(t_i) \cdot \Delta W_i(\omega_0) \\ &+ \sigma_{FK} \cdot (1 - u_1(t_i)) \cdot x_0(t_i) \cdot \Delta W_i(\omega_1) \\ X_1(t_{i+1}) &= X_1(t_i) + h \cdot 1 \end{aligned}$$

Bemerkung 4.5 *Der Algorithmus des Stochastisches Einschritt-Verfahren für das Modell (4.6) kann in der stochastischen dynamischen Programmierung nicht direkt mit den zwei zweipunktverteilen Zufallsvariablen $\Delta W_i(\omega_0)$ und $\Delta W_i(\omega_1)$ implementiert werden. Grund ist die Verwendung der Werteiteration, da in jedem Iterationsschritt eine neue Zufallsvariable $\Delta W_i(\omega)$ durch den Algorithmus erzeugt werden würde[5]. Während der Werteiteration muss aber für jeden Gitterpunkt x der stochastische Einfluss in jeder Iteration gleich sein, da sonst keine Konvergenz garantiert werden kann.*
*Auf Seite 68 wird vorgestellt, wie in der Berechnung von temp in der Funktion **void finde_max_besser** das Stochastische Einschritt-Verfahren für Modell (4.6) in der Werteiteration umgesetzt wird.*

[4] in Anlehnung an [5] Grüne, 2008, S.102/103, abgewandelt auf das Modell (4.6)
[5] vergleiche Algorithmus 4.3 Schritt 2.(a)

4.3 Eigenschaften der optimalen Wertefunktion für das Modell (4.1) bzw. (4.6)

Lemma 4.6
Es seien $g : \mathbb{R}^k \to \mathbb{R}$ konkav auf einer konvexen Menge $X \subseteq \mathbb{R}^k$ und $f : \mathbb{R}^n \to \mathbb{R}^k$ linear . Dann ist deren Komposition $h = g \circ f : \mathbb{R}^n \to \mathbb{R}$, definiert durch $h(x) = g(f(x))$, konkav in x auf $f^{-1}(X)$.

Beweis:
Es gilt $g(f(x + \lambda(\widetilde{x} - x))) = g(f(x) + \lambda(f(\widetilde{x}) - f(x)))$, da $f(x)$ linear
und $g(f(x)+\lambda(f(\widetilde{x})-f(x))) \geq g(f(x))+\lambda(g(f(\widetilde{x}))-g(f(x)))$, da $g(u)$ mit $u = f(x)$ konkav in u ist. Damit folgt die Behauptung. □

Lemma 4.7
$f^h(x, u, z)$ aus Modell (4.1) bzw. (4.6) ist linear in $u = (u_0, u_1)$.

Beweis:

$$
\begin{aligned}
& f_0^h(x, u + \lambda(\widetilde{u} - u), z) = \\
& \qquad x_0 + x_0 \cdot h \left[u_1 \cdot (r_e + \frac{1}{h} z_0 \sigma_{EK}) + (1 - u_1) \cdot (r_f + \frac{1}{h} z_1 \sigma_{FK}) - u_0 \right] && (4.7) \\
+ \quad & x_0 \cdot h \cdot \lambda \left[\widetilde{u}_1 \cdot (r_e + \frac{1}{h} z_0 \sigma_{EK}) + \widetilde{u}_1 \cdot (r_f + \frac{1}{h} z_1 \sigma_{FK}) - \widetilde{u}_0 \right] && (4.8) \\
- \quad & x_0 \cdot h \cdot \lambda \left[u_1 \cdot (r_e + \frac{1}{h} z_0 \sigma_{EK}) + u_1 \cdot (r_f + \frac{1}{h} z_1 \sigma_{FK}) - u_0 \right] && (4.9)
\end{aligned}
$$

Der Term (4.7) entspricht dabei $f_0^h(x, u, z)$. Wenn man nun $x_0+x_0 \cdot h \cdot \lambda \cdot (r_f+\frac{1}{h} z_1 \cdot \sigma_{FK})$ in Term (4.8) und $-x_0 - x_0 \cdot h \cdot \lambda \cdot (r_f + \frac{1}{h} z_1 \cdot \sigma_{FK})$ in (4.9) einfügt (die Summe beider Terme ergibt 0), so ergibt die Summe aus (4.8) und (4.9) $\lambda \cdot (f_0^h(x, \widetilde{u}, z) - f_0^h(x, u, z))$. Damit folgt die Linearität für $f_0^h(x, u, z)$.
Da $f_1^h(x, u, z)$ nicht von u abhängt, folgt die Linearität trivialerweise. □

Lemma 4.8
Die Power-Nutzenfunktion aus Gleichung (4.3) ist für $\frac{1}{2} \leq \gamma$ konkav in $u_0 \geq 0$ und $x_0 \geq 0$.

Beweis:

$$
\begin{aligned}
g(x, u) &= \frac{(u_0 \cdot x_0)^{1-\gamma} - 1}{1 - \gamma} \\
\nabla g(x, u) &= \begin{pmatrix} u_0^{1-\gamma} \cdot x_0^{-\gamma} \\ u_0^{-\gamma} \cdot x_0^{1-\gamma} \end{pmatrix} && \textit{Gradient} \\
H(g(x, u)) &= \begin{pmatrix} -\gamma \cdot u_0^{1-\gamma} \cdot x_0^{-1-\gamma} & (1-\gamma) \cdot u_0^{-\gamma} \cdot x_0^{-\gamma} \\ (1-\gamma) \cdot u_0^{-\gamma} \cdot x_0^{-\gamma} & -\gamma \cdot u_0^{-1-\gamma} \cdot x_0^{1-\gamma} \end{pmatrix} && \textit{Hessematrix}
\end{aligned}
$$

Berechnung der Eigenwerte von $H(g(x,u))$ (E ist die Einheitsmatrix):

$$\begin{aligned}
0 &= det(H(g(x,u)) - \lambda \cdot E) \Leftrightarrow \\
0 &= (-\gamma \cdot u_0^{1-\gamma} \cdot x_0^{-1-\gamma} - \lambda) \cdot (-\gamma \cdot u_0^{-1-\gamma} \cdot x_0^{1-\gamma} - \lambda) - (1-\gamma)^2 \cdot u_0^{-2\gamma} \cdot x_0^{-2\gamma} \Leftrightarrow \\
0 &= \lambda^2 + \lambda \cdot \gamma \cdot (u_0^{1-\gamma} \cdot x_0^{-1-\gamma} + u_0^{-1-\gamma} \cdot x_0^{1-\gamma}) + (2\gamma - 1) \cdot u_0^{-2\gamma} \cdot x_0^{-2\gamma}
\end{aligned}$$

Damit gilt für $\lambda_{1,2}$:

$$\begin{aligned}
\lambda_{1,2} &= -\frac{1}{2}\gamma \cdot (u_0^{1-\gamma} \cdot x_0^{-1-\gamma} + u_0^{-1-\gamma} \cdot x_0^{1-\gamma}) \\
&\overset{+}{-} \sqrt{\frac{1}{4}\gamma^2 \cdot (u_0^{1-\gamma} \cdot x_0^{-1-\gamma} + u_0^{-1-\gamma} \cdot x_0^{1-\gamma})^2 - (2\gamma - 1) \cdot u_0^{-2\gamma} \cdot x_0^{-2\gamma}} \qquad (4.10)
\end{aligned}$$

$g(x,u)$ ist konkav, genau dann wenn $H(g(x,u))$ negativ semidefinit ist, d.h. alle Eigenwerte (λ_1 und λ_2) kleiner gleich 0 sind.[6]

$$\begin{aligned}
& \lambda_{1,2} \leq 0 \\
\Leftrightarrow\ & \frac{1}{4}\gamma^2 \cdot (u_0^{1-\gamma} \cdot x_0^{-1-\gamma} + u_0^{-1-\gamma} \cdot x_0^{1-\gamma})^2 - (2\gamma - 1) \cdot u_0^{-2\gamma} \cdot x_0^{-2\gamma} \\
& \leq \frac{1}{4}\gamma^2 \cdot (u_0^{1-\gamma} \cdot x_0^{-1-\gamma} + u_0^{-1-\gamma} \cdot x_0^{1-\gamma})^2 \\
\Leftrightarrow\ & u_0^{-2\gamma} \cdot x_0^{-2\gamma} \leq 2\gamma \cdot u_0^{-2\gamma} \cdot x_0^{-2\gamma} \\
\Leftrightarrow\ & \frac{1}{2} \leq \gamma
\end{aligned}$$

Es muss nun nur noch gezeigt werden, dass die Wurzel in Gleichung (4.10) reell ist. Für $\gamma \geq 0$ gilt dann:

$$\begin{aligned}
& \frac{1}{4}\gamma^2 \cdot (u_0^{1-\gamma} \cdot x_0^{-1-\gamma} + u_0^{-1-\gamma} \cdot x_0^{1-\gamma})^2 - 2\gamma \cdot u_0^{-2\gamma} \cdot x_0^{-2\gamma} + u_0^{-2\gamma} \cdot x_0^{-2\gamma} \qquad (4.11) \\
&= \underbrace{\left(\frac{1}{2}\gamma \cdot \left(u_0^{1-\gamma} \cdot x_0^{-1-\gamma} + u_0^{-1-\gamma} \cdot x_0^{1-\gamma}\right) - u_0^{-\gamma} \cdot x_0^{-\gamma}\right)^2}_{\geq 0} \qquad (4.12) \\
&\quad + \underbrace{\gamma}_{\geq 0} \cdot \underbrace{u_0^{-1-2\gamma} \cdot x_0^{-1-2\gamma}}_{\geq 0,\, da\, x_0, u_0 \geq 0} \cdot \underbrace{(u_0 - x_0)^2}_{\geq 0} \qquad (4.13) \\
&\geq 0
\end{aligned}$$

Dabei erhält man Term (4.11), indem man die Summe aus Term (4.12) und (4.13) ausmultipliziert. □

[6] In Gleichung (4.10) wird $\lambda_{1,2}$ für den Fall „$-\sqrt{\cdots}$“immer ≤ 0 sein, da x_0, u_0 und γ alle ≥ 0 sind. Daher wird im Folgenden nur vom Fall „$+\sqrt{\cdots}$“ausgegangen

Satz 4.9 (Konkavität der optimalen Wertefunktion)
Gegeben sei das stochastische Kontrollsystem aus Modell (4.1) bzw. (4.6). Für die Power-Nutzenfunktion (Gleichung (4.3)) gelte $\frac{1}{2} \leq \gamma$. Des Weiteren sei ein kompakter Kontrollwertebereich U und eine Schrittweite $h > 0$ gegeben.
Dann ist in jedem Schritt T der Werteiteration das Funktional $J_T(x, u)$ aus Gleichung (4.2) konkav in u_0 (≥ 0) und u_1 (beliebig) sowie in $x_0 \geq 0$. Die optimale Wertefunktion $V_T(x)$ ist dabei konkav in der Komponente $x_0 \geq 0$.

Beweis:
Dass $g(x, u)$ für $\frac{1}{2} \leq \gamma$ konkav in $u_0 \geq 0$ und $x_0 \geq 0$ ist, folgt aus Lemma 4.8.
Nach Gleichung (4.2) gibt es keine Endkosten, d.h. man setzt $V_0(x) = 0$ für alle $x = (x_0, x_1)$.
Die Behauptung wird nun mit vollständiger Induktion nach den Iterationsschritten der Werteiteration bewiesen, dabei sei immer $x = (x_0, x_1)$ fest gewählt:
Induktionsanfang:

$$J_1(x, u) = h \cdot g(x, u) + \beta \cdot V_0(f^h(x, u, z)) = h \cdot g(x, u)$$

Also ist $J_1(x, u)$ konkav in u und x_0. Weiter gilt für zwei beliebige Punkte $x = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$ bzw. $\widetilde{x} = \begin{pmatrix} \widetilde{x}_0 \\ \widetilde{x}_1 \end{pmatrix}$ sowie den zugehörigen optimalen Kontrollen $\begin{pmatrix} u_0 \\ u_1 \end{pmatrix}^*$ bzw. $\begin{pmatrix} \widetilde{u}_0 \\ \widetilde{u}_1 \end{pmatrix}^*$ und $\lambda \geq 0$ sowie $x_1 = \widetilde{x}_1$ festgehalten:

$$\begin{aligned}
&(1-\lambda) \cdot V_1\left(\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}\right) + \lambda \cdot V_1\left(\begin{pmatrix} \widetilde{x}_0 \\ x_1 \end{pmatrix}\right) \\
&= (1-\lambda) \cdot J_1\left(\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \begin{pmatrix} u_0 \\ u_1 \end{pmatrix}^*\right) + \lambda \cdot J_1\left(\begin{pmatrix} \widetilde{x}_0 \\ x_1 \end{pmatrix}, \begin{pmatrix} \widetilde{u}_0 \\ \widetilde{u}_1 \end{pmatrix}^*\right) && (4.14) \\
&\leq J_1\left((1-\lambda) \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} + \lambda \cdot \begin{pmatrix} \widetilde{x}_0 \\ x_1 \end{pmatrix}, (1-\lambda) \cdot \begin{pmatrix} u_0 \\ u_1 \end{pmatrix}^* + \lambda \cdot \begin{pmatrix} \widetilde{u}_0 \\ \widetilde{u}_1 \end{pmatrix}^*\right) && (4.15) \\
&\leq \max_{u=(u_0,u_1)^T} J_1\left((1-\lambda) \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} + \lambda \cdot \begin{pmatrix} \widetilde{x}_0 \\ x_1 \end{pmatrix}, \begin{pmatrix} u_0 \\ u_1 \end{pmatrix}\right) = V_1\left((1-\lambda) \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} + \lambda \cdot \begin{pmatrix} \widetilde{x}_0 \\ x_1 \end{pmatrix}\right)
\end{aligned}$$

Beim Übergang von (4.14) zu (4.15) benutzt man die Konkavität von $J_1(x, u)$ in u und x_0.
Damit ist auch die Konkaviät von $V_1(x) = \max_{u \in U} J_1(x, u)$ in x_0 bewiesen.
Induktionsvoraussetzung: $V_T(x)$ ist konkav in x_0.
Induktionsschritt: Dann gilt für $J_{T+1}(x, u)$

$$J_{T+1}(x, u) = h \cdot g(x, u) + \beta \cdot V_T(f^h(x, u, z))$$

Bei der Maximierung von $J_{T+1}(x, u)$ über u sind x und z fest gewählt. $f^h(x, u, z)$ wird nur in u verändert, d.h. $f^h(x, u, z)$ ist nach Lemma 4.7 linear. Nach Induktionsvoraussetzung ist $V_T(y)$ in y_0 konkav, wobei $y = f^h(x, u, z)$ gesetzt wurde

($y_1 = f_1^h(x,u,z)$ ist konstant in u). Nach Lemma 4.6 ist $V_T(f^h(x,u,z))$ in u konkav. Da $g(x,u)$ konkav in u ist und die Summe konkaver Funktionen wieder konkav ist, ist $J_{T+1}(x,u)$ ebenfalls konkav in u.
Um die Induktion abzuschließen, muss noch nachgewiesen werden, dass $V_{T+1}(x)$ in x_0 konkav ist (siehe Induktionsvoraussetzung). Es gilt

$$V_{T+1}(x) = \max_{u\in U} J_{T+1}(x,u) = \max_{u\in U} \left\{ h\cdot g(x,u) + \beta \cdot V_T(f^h(x,u,z)) \right\}. \tag{4.16}$$

Für $V_{T+1}(x)$ gilt durch rekursives Anwenden von Gleichung (4.16)

$$\begin{aligned}
V_{T+1}(x) &= \max_{u^{(T+1)}\in U} \Bigg[h\cdot g(x,u^{(T+1)}) \\
&+ \beta \cdot \max_{u^{(T)}\in U} \Bigg[h\cdot g\left(f^h(x,u^{(T+1)},z),u^{(T)}\right) \\
&+ \beta \cdot \max_{u^{(T-1)}\in U} \Big[h\cdot g\left(f^h(f^h(x,u^{(T+1)},z),u^{(T)},z),u^{(T-1)}\right) \\
&+ \beta \cdot \max_{u^{(T-2)}\in U} \big[h\cdot g\left(f^h(f^h(f^h(x,u^{(T+1)},z),u^{(T)},z),u^{(T-1)},z),u^{(T-2)}\right) \\
&+ \cdots \\
&+ \beta \cdot \max_{u^{(1)}\in U} \big[h\cdot g\big(\underbrace{f^h(\cdots(f^h}_{T-mal\ f^h}(x,u^{(T+1)},z),u^{(T)},z \underbrace{)\cdots)}_{(T-1)-mal}, u^{(1)}\big) \big] \cdots \Big] \Big] \Bigg]
\end{aligned} \underbrace{\qquad\qquad}_{(T+1)-mal} \tag{4.17}$$

Im Schritt $T+1$ sind die Maximierung von $V_1, ..., V_{T+1}$ bereits erfolgt, d.h. man kann die Kontrollen $u^{(i)}$, $i = 1, ..., T+1$ fest auf ihren maximierenden Wert wählen. Bei der Überprüfung der Konkavität in x_0 (Komponente von x) muss nur noch beachtet werden, dass mit x auch immer die Kombination von $u^{(T+1)} = (u_0^{(T+1)}, u_1^{(T+1)})$ variiert.
Nach (4.6) gilt

$$\begin{aligned}
f_0^h(x,u,z) &= x_0 \cdot \Big(1 + u_1 \cdot \underbrace{\left(h\cdot(r_e - r_f) + \sigma_{EK}\cdot z_0 - \sigma_{FK}\cdot z_1\right)}_{:=c_1} \\
&- \underbrace{h}_{:=c_2} \cdot u_0 + \underbrace{h\cdot r_f + \sigma_{FK}\cdot z_1}_{:=c_3} \Big)
\end{aligned}$$

c_1 und c_3 hängen von r_e bzw. r_f ab. Diese wiederum hängen von der Zeit, also der x_1-Komponente ab, d.h. bei jeder Anwendung von f^h wird die Zeit um h erhöht (durch $f_1^h(x,u,z) = x_1 + h$). Daher ist $c_1^{(t)}$ bzw. $c_3^{(t)}$ die jeweilige Konstante für den Iterationsschritt t.
$V_T(x)$ ist nun konkav in x_0 wenn in jedem Summand g in x_0 konkav ist. $g(x,u^{(T+1)})$

ist wieder nach Lemma 4.8 konkav in x_0 und $u^{(T+1)}$.

$$g\left(f^h(x,u^{(T+1)},z),u^{(T)}\right) =$$
$$= \frac{1}{1-\gamma}\cdot\left(x_0^{1-\gamma}\left(1+u_1^{(T+1)}c_1^{(T+1)}-u_0^{(T+1)}c_2+c_3^{(T+1)}\right)^{1-\gamma}\cdot\underbrace{(u^{(T)})^{1-\gamma}}_{konstant}-1\right)$$

Dabei ist $f^h(x,u^{(T+1)},z)$ in beiden Komponenten im Algorithmus immer größer gleich 0,[7] und somit ist $(1+u_1^{(T+1)}c_1^{(T+1)}-u_0^{(T+1)}c_2+c_3^{(T+1)}) := \widetilde{u}(x)$[8] ≥ 0 , da $x_0 \geq 0$ nach Voraussetzung gilt. Damit ist $g\left(f^h(x,u^{(T+1)},z),u^{(T)}\right)$ wieder von der Form

$$g\left(f^h(x,u^{(T+1)},z),u^{(T)}\right) = \frac{(x_0\cdot\widetilde{u}(x))^{1-\gamma}-1}{1-\gamma}, \tag{4.18}$$

d.h. nach Lemma 4.8 konkav in x_0. Für die anderen g aus der Maximierung über $u^{(t)}$, $t=1,...,T-1$ aus Gleichung (4.17) gilt dann:

$$g\big(\underbrace{f^h(\cdots(f^h}_{(T-t+1)-mal\ f^h}(x,u^{(T+1)},z),u^{(T)},z\underbrace{)\cdots)}_{(T-t)-mal},u^{(t)}\big) =$$
$$= \frac{1}{1-\gamma}\cdot\Bigg(\underbrace{\left[\prod_{k=t+1}^{T}\left(1+u_1^{(k)}c_1^{(k)}-u_0^{(k)}c_2+c_3^{(k)}\right)^{1-\gamma}\right]\cdot(u_0^{(t)})^{1-\gamma}}_{=:\delta^{(t)}\quad konstant}$$
$$\cdot\quad\Big(\underbrace{1+u_1^{(T+1)}c_1^{(T+1)}-u_0^{(T+1)}c_2+c_3^{(T+1)}}_{=\widetilde{u}(x)}\Big)^{1-\gamma}\cdot x_0^{1-\gamma}\Bigg)-\frac{1}{1-\gamma}$$
$$= \delta^{(t)}\cdot\frac{(x_0\cdot\widetilde{u}(x))^{1-\gamma}-1}{1-\gamma} \tag{4.19}$$

Damit ist Gleichung (4.19) von der Form von (4.18) und damit konkav in x_0, d.h. auch $V_{T+1}(x)$ ist in x_0 konkav, womit der Induktionsbeweis abgeschlossen ist. □

Bemerkung 4.10
Auf dem Gitter bleibt die Konkavität erhalten, d.h. wenn man die Funktionswerte der benachtbarten Gitterpunkte mit Ebenen verbindet, so erhält man ein konkaves Polyeder.

[7] siehe dazu Abschnitt 5.1.2 in der Berechnung von temp, Schritt 2 auf Seite 68

[8] $\widetilde{u}(x)$ drückt aus, dass es für jedes x eine Kombination von $u_0^{(T+1)}$ und $u_1^{(T+1)}$ gibt

Kapitel 5

Vorstellung der verwendeten Programme

Sämtliche Programme befinden sich auf der beiliegenden CD im Unterverzeichnis *CD:\Programme*.

5.1 Das Hauptprogramm für die Portfoliooptimierung

Das Hauptprogramm befindet sich auf der CD in Unterverzeichnis *CD:\Programme\Hauptprogramm*. Es besteht aus *gridgen.c*, welches die benötigten Gitter für die in Abschnitt 3.2.2 mathematisch beschriebene Diskretisierung liefert, und *Portfolio.c*, wo die eigentliche Portfoliooptimierung mit Hilfe der Werteiteration erfolgt.

5.1.1 *gridgen.c*

Das Programm *Portfolio.c* benutzt das Modul *gridgen.c*, Version 0.99t vom 17.03.2008. Ich werde im Folgenden nur die wichtigsten von *Portfolio.c* verwendeten Prozeduren vorstellen. Für eine umfassende Dokumentation zum Gittergenerator gridgen siehe [6] Grüne, 2007, Anhang A.

- **Erzeugen und löschen eines Gitters:**
  - **qgrig *g:**
    Das im Programm erzeugte Gitter g wird durch einen Zeiger vom Typ qgrig* repräsentiert.
  - **qgrig *g=create_grid(double *lo, double *hi, double *Delta, int dim, int level):** Erzeugt Gitter mit Namen g der Dimension *dim* mit der unteren Grenze *lo*, der oberen Grenze *hi* und der Rechteckgröße (eines Gitterrechtecks) *Delta*. Für die in *Portfolio.c* verwendeten regelmäßigen

Gitter setzt man $level = 0$.
lo, hi und $Delta$ sind dabei von der Dimension dim.

- **void delete_grid(qgrig *g):**
  Löscht das Gitter mit dem Namen g.

- **Durchlaufen der Gitterknoten, schreiben und lesen der Gitterwerte:**

  - **int index=first_node(qgrig *g, double *x):**
    Setzt einen internen Zeiger auf den ersten Knoten des Gitters.
  - **int index=next_node(qgrig *g, double *x):**
    Setzt den Zeiger auf den nächsten Gitterknoten. Als Funktionswert wird der Index des aktuellen Knoten ($index \geq 0$) zurückgegeben. Wenn alle Knoten des Gitters abgearbeitet sind, wird $index = -1$ gesetzt (Abbruchkriterium).
  - **int set_current_nodevalue(qgrig *g, double *x):**
    Setzt den Wert des aktuellen Gitterknotens des Gitters g auf den Wert x. Die Funktion gibt im Erfolgsfall 0, ansonsten 1, zurück.
  - **double zahl=value(qgrig *,x,&flag):**
    Liefert für einen beliebigen Gitterpunkt x des Gitters g den Wert $zahl$ zurück. $flag$ gibt des Status der Operation zurück: 0 bedeutet kein Fehler, 1, dass x außerhalb des Gitters liegt, und 2 ein anderer Fehler (z.B. Übergabe eines falschen Zeigers g).

- **Gitter speichern und exportieren sowie laden:**

  - **int save_val(qgrig *g, char *filename):**
    Gibt im Erfolgsfall 0, ansonsten 1, zurück. Sichert das Gitter g mit dem Namen $filename$ so, dass es von load_val wieder eingelesen werden kann.
  - **int export_gridandval(qgrig *g, char *filename):**
    Gibt im Erfolgsfall 0, ansonsten 1, zurück. Speichert das Gitter g mit dem Namen $filename$ so, dass es mit Hilfe von *gridplot2d.m* (siehe Seite 71) in Matlab geplottet werden kann.
  - **qgrig *g=load_val(char *filename):**
    Lädt das Gitter mit dem Namen $filename$ und speichert es in g ab. Im Falle eines Fehlers wird ein Null-Zeiger zurückgegeben und eine Fehlermeldung auf dem Bildschirm ausgegeben.

5.1.2 *Portfolio.c*

Verwaltung der Parameter

Die Parameter (siehe Tabelle 6.1) werden in der Struktur Konstanten , welche wiederrum aus den Strukturen Gitterkonstanten, Problemparameter und Diskretisierung aufgebaut ist, zusammengefasst (vgl. Abbildung 5.1).

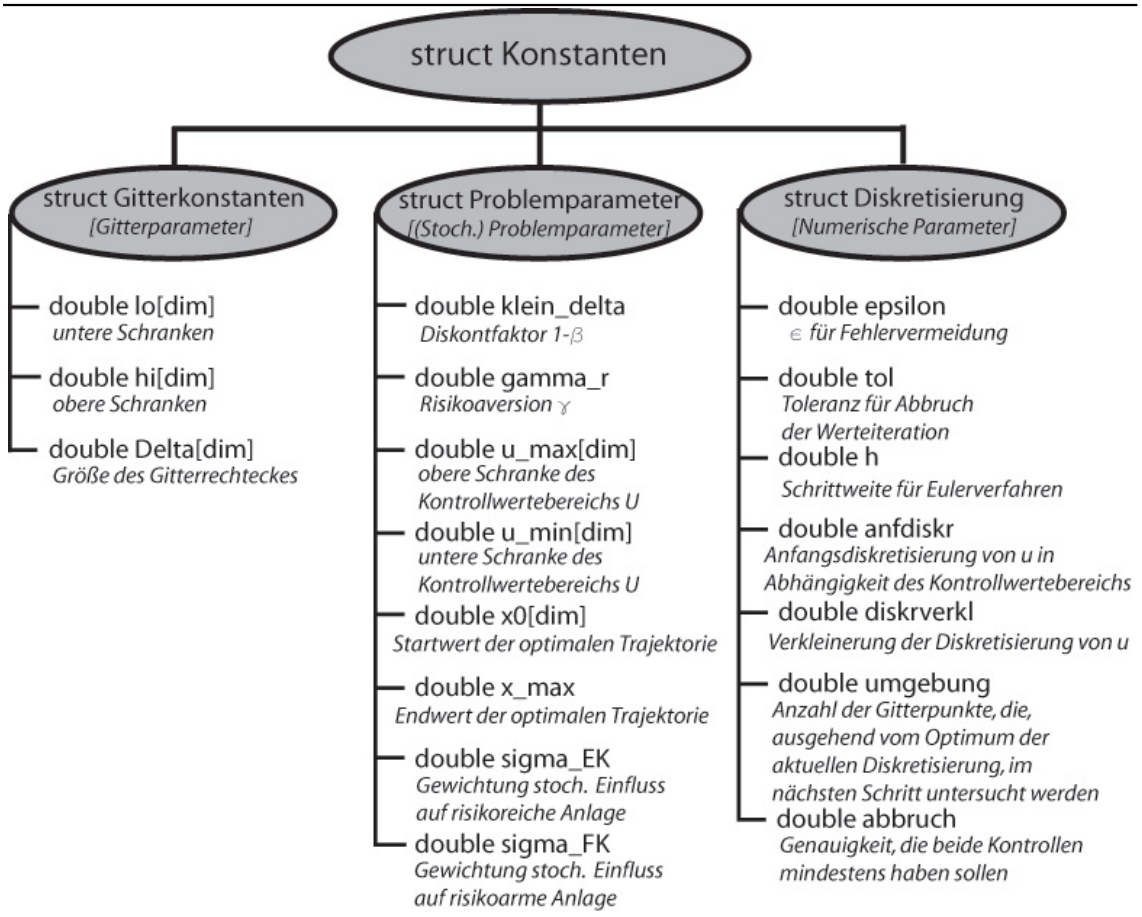

Abbildung 5.1: Aufbau der Struktur der Parameter

Der Aufruf erfolgt mittels

$$Name\_Konstanten\,.\left\{\begin{array}{c} gitter \\ param \\ diskr \end{array}\right\}.Name\_Parameter$$

Dabei ist $Name\_Konstanten$ vom Typ $struct\,Konstanten$ und $Name\_Parameter$ ist ein passender Parametername (z.B. lo, $gamma\_r$ oder h).

Beschreibung der Inputvariablen

In der folgenden Tabelle sind sämtliche Variablen/Parameter aufgeführt, die in den folgenden Routinen als Input verwendet werden.

Typ	Variable	Bedeutung
qgrid	$*Gitter\_V$	zweidimensionales Gitter der optimalen Wertefunktion im letzten Iterationsschritt
qgrid	$*Gitter\_V\_temp$	zweidimensionales Gitter der optimalen Wertefunktion im aktuellen Iterationsschritt
qgrid	$*Gitter\_u0$	zweidimensionales Gitter der optimalen Kontrolle u[0] im aktuellen Iterationsschritt
qgrid	$*Gitter\_u1$	zweidimensionales Gitter der optimalen Kontrolle u[1] im aktuellen Iterationsschritt
qgrid	$*Gitter\_rFK$	eindimensionales Gitter für den Zinsverlauf der risikoarmen Anlage
qgrid	$*Gitter\_rEK$	eindimensionales Gitter für den Zinsverlauf der risikoreichen Anlage
double	$*x$	aktuelle Punkt x aus $*Gitter\_V$, für den die optimale Kontrolle u gefunden werden soll
double	$*u$	ist die aktuell betrachtete Kontrolle
double	$*z$	dient der Modellierung des Wiener Prozesses, wird auf $\sqrt{h}$ oder $-\sqrt{h}$ gesetzt
struct Konstanten	$*K$	Struktur, in der alle Parameter des Problems enthalten sind
float	geszeit	speichert die von *Portfolio.c* benötigte Zeit
int	iter	zählt die Iterationen der Werteiteration

Ein- und Ausgaben

- **struct Konstanten eingabe(FILE *fp):**
  Liest die Konstanten aus der Datei fp ein und speichert diese in der Struktur $Konstanten$ ab.

- **void ausgabe(struct Konstanten *K):**
  Erzeugt die Startausgabe bei Programmaufruf. Dabei werden die verwendeten Konstanten noch einmal zur Kontrolle in der Konsole ausgegeben.

- **void statistiken(struct Konstanten *K, float geszeit, int iter):**
  Schreibt in die Datei *Zusammenfassung.txt* im Ordner \Modell eine Zusammenfassung nach Beendigung des Programms. Dazu werden sämtliche Parameter sowie die Laufzeit und die Anzahl der Iterationen, die das Programm gebraucht hat, aufgelistet.

Routinen, die die Dynamik modellieren

- **double l(double ∗x, double ∗u, struct Konstanten ∗K):**
  Modelliert die Power-Nutzenfunktion (siehe Gleichung (4.3)). Dabei müssen folgende Spezialfälle betrachtet werden.

Output:		
γ	$u[0] \cdot x[0]$	Rückgabewert
≤ 1	beliebig	$\frac{1}{1-\gamma} \cdot \big((u[0] \cdot x[0])^{1-\gamma} - 1\big)$
> 1	$\geq \epsilon$	
$= 1$	$\geq \epsilon$	$\ln(u[0] \cdot x[0])$
$= 1$	$< \epsilon$	-1000000
> 1	$< \epsilon$	

- **void f(qgrid ∗Gitter_rFK, qgrid ∗Gitter_rEK, double ∗x, double ∗u, double *erg, struct Konstanten *K):**

Output:		
$erg[0]$	$=$	$u[1] \cdot r_e(y) \cdot x[0] + (1 - u[1]) \cdot r_f(y) \cdot x[0] - x[0] \cdot u[0]$
$erg[1]$	$=$	1

  Dabei ist $r_e(y) = value(Gitter\_rEK, y, \&flag)$ und
  $r_f(y) = value(Gitter\_rFK, y, \&flag)$ mit $y = x[1]$ der entsprechende Zins.

- **void Stoch_Euler_Verfahren(qgrid ∗Gitter_rFK, qgrid ∗Gitter_rEK, double ∗x, double ∗u, double ∗z, double ∗erg, struct Konstanten ∗K):**

Output:		
$erg[0]$	$=$	$x[0] + h \cdot rs[0] + \sigma_{EK} \cdot u[1] \cdot x[0] \cdot z[0] + \sigma_{FK} \cdot (1 - u[1]) \cdot x[0] \cdot z[1]$
$erg[1]$	$=$	$x[1] + h \cdot rs[1]$

  Dabei entspricht $rs[0]$ dem Ergebnis $erg[0]$ sowie $rs[1]$ dem Ergebnis $erg[1]$ der Funktion *void f* (s.o.).

Routinen zur Berechnung der optimalen Trajektorien

- **double erzeuge_Zufallszahl():**
  Ruft die C-Funktion *rand()* auf und normiert das Ergebnis durch Division durch *RAND_MAX*. Die entstehende Zufallszahl ist aus $[0, 1]$.

- **double Wiener_Prozess(struct Konstanten ∗K):**
  Umsetzung von Schritt 2.(a) aus Algorithmus 4.3 und Rückgabe des $\Delta W_i(\omega)$.

- **void Stoch_trajektorie(qgrid ∗Gitter_u0, qgrid ∗Gitter_u1, qgrid ∗Gitter_rFK, qgrid ∗Gitter_rEK, struct Konstanten ∗K):**
  Berechnet mit Hilfe der optimalen Kontrollen u_0 und u_1 für die dazugehörigen Zinsverläufe die optimale Trajektorie für ein (in struct Konstanten ∗K) vorgegebenes x(0) bis zu einem Endzeitpunkt x_max (vergleiche Tabelle 6.1).

Die Daten werden im Ordner \Modell in der Datei traj_$x_0(0)$_$x_1(0)$.txt gespeichert (für $x_0(0)$ und $x_1(0)$ wird der jeweilige Wert eingesetzt). Gespeichert werden die Zeit $x[1]$, das Vermögen $x[0]$, der absolute Konsum $x[0] \cdot u[0]$[1], die Kontrolle $u[1]$ und die optimale Wertefunktion.
Geplottet werden diese (außer der optimalen Wertefunktion) mit dem Matlab-Programm *Trajekplot.m* (vergleiche Abschnitt 5.2.3).

void finde_max_besser(qgrid *Gitter_V, qgrid *Gitter_rFK, qgrid *Gitter_rEK, double *x, struct Konstanten *K, double *u, double *max_val)

Motivation und Art des Algorithmus:
Dieser Maximierungsalgorithmus ist ein heuristischer. Motivation dafür war der Fakt, dass für eine zweidimensionale Kontrolle bei Verdopplung der Diskretisierungsgenauigkeit sich der Aufwand nahezu vervierfacht.
Wenn man zum Beispiel auf einem Kontrollwertebereich $U = [0,1] \times [0,1]$ beide Kontrollen u_i mit $\Delta u_i = 0.1$ $(i = 0, 1)$ auf einem Gitter approximiert, dann muss man nach Definition 3.24 $(\frac{1-0}{0.1} + 1)^2 = 121$ Eckpunkte betrachten. Für $\Delta u = 0.001$ (wie in finde_max_besser) wären dies schon $(\frac{1-0}{0.001} + 1)^2 = 1002001$ Eckpunkte, also 8281-mal ($\approx 10^4$-mal) soviele.
Da die Optimierung über die Kontrolle u für jeden Punkt x auf dem Gitter *Gitter_V* und dies wiederrum für jeden Iterationsschritt der Werteiteration gemacht werden muss, ist es erforderlich, dass der Algorithmus für die optimale Kontrolle schnell ist, aber trotzdem qualitativ genaue Ergebnisse liefert.

Aufwand des Algorithmus:
Sei $U = [u_{min}[0], u_{max}[0]] \times [u_{min}[1], u_{max}[1]]$.
Dann müssen im ersten Schritt

$$\left(\frac{u_{max}[0] - u_{min}[0]}{anfdiskr \cdot (u_{max}[0] - u_{min}[0])} + 1\right) \cdot \left(\frac{u_{max}[1] - u_{min}[1]}{anfdiskr \cdot (u_{max}[1] - u_{min}[1])} + 1\right)$$
$$= \left(\frac{1}{anfdiskr} + 1\right)^2$$

Eckpunkte berechnet werden.
In jedem folgenden Schritt müssen maximal $(2 \cdot umgebung + 1)^2$ Eckpunkte berechnet werden (wenn das Optimum der derzeitigen Diskretisierung auf oder nahe dem Rand liegt, dann sind es weniger Eckpunkte).
Für den Gesamtaufwand ist nun noch die Anzahl der nötigen Schritte, bis $abstand[i] < abbruch$ $(i = 0, 1)$ ist, interessant. Dabei gilt für Schritt t

$$abstand[i] = (diskrverkl)^{t-1} \cdot anfdiskr \cdot (u_{max}[i] - u_{min}[i]) \quad (i = 0, 1).$$

[1]Wenn durch das stochastische Eulerverfahren ein Vermögen $x[0]$ erreicht wird, welches über der oberen Schranke des Gitters liegt, dann wird die Differenz zwischen $x[0]$ und dieser oberen Schranke zusätzlich konsumiert.

Im folgenden sei $maxdiff = \max_{i=0,1}(u_{max}[i] - u_{min}[i])$.
Dann gilt für t (Anzahl der Schritte):

$$(t-1) \cdot \ln(diskrverkl) + \ln(anfdiskr) + \ln(maxdiff) < \ln(abbruch)$$

$$\Rightarrow \quad t - 1 = \left\lceil \underbrace{\frac{\ln(abbruch) - \ln(anfdiskr) - \ln(maxdiff)}{\ln(diskrverkl)}}_{=\alpha} \right\rceil + \delta,$$

wobei $\ln(diskrverkl) < 0$ ist, da $0 < diskrverkl < 1$ gilt. Wenn α ganzzahlig ist, dann setze $\delta = 1$, sonst auf 0. Insgesamt müssen maximal

$$\left(\frac{1}{anfdiskr} + 1\right)^2 + (2 \cdot umgebung + 1)^2 \cdot \left(\left\lceil \frac{\ln(abbruch) - \ln(anfdiskr) - \ln(maxdiff)}{\ln(diskrverkl)} \right\rceil + \delta \right)$$

Eckpunkte untersucht werden.
Für $anfdiskr = 0.1$, $diskrverkl = 0.5$, $umgebung = 2$, $abbruch = 0.001$ und $U = [0, 0.7] \times [-3, 4.5]$ sind dies maximal 371 Eckpunkte. Bei der Untersuchung auf einem gesamten Gitter der Genauigkeit 0.001 wären dies genau $701 \cdot 7501 = 5258201$ Eckpunkte, also $\approx$ 14000-mal soviele.

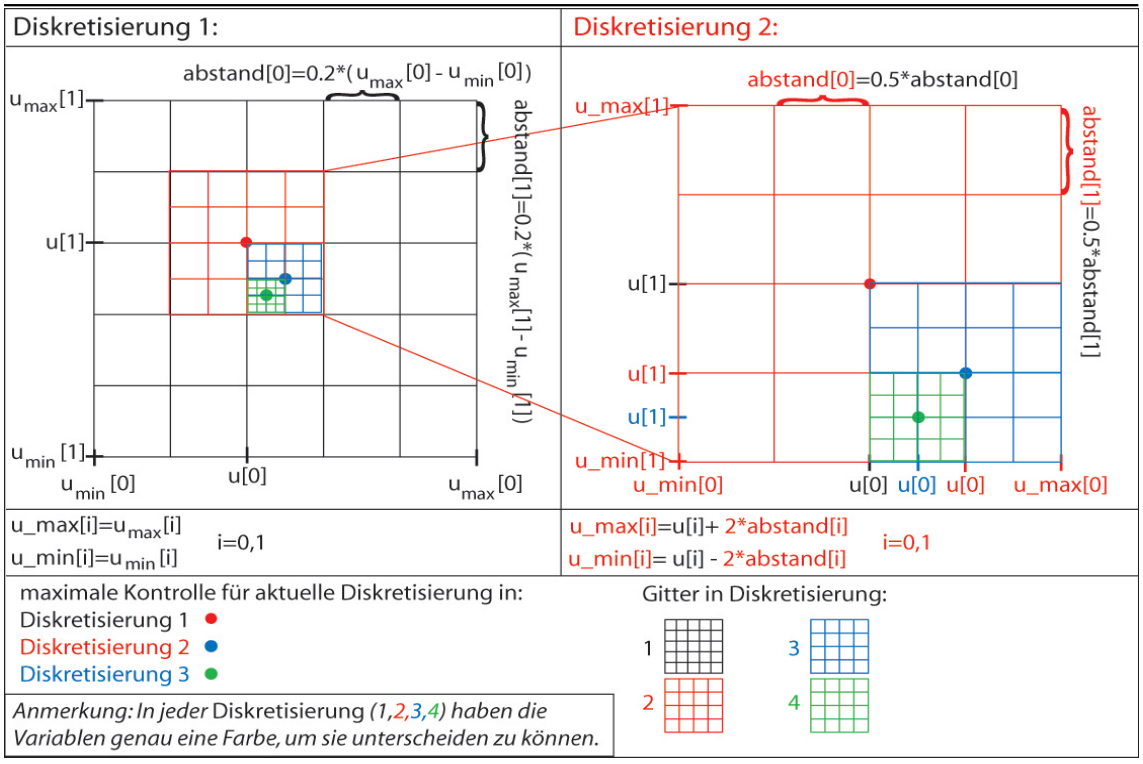

Abbildung 5.2: Beispiel der Funktionsweise von *void finde_max_besser*

Algorithmus:

Output:

Typ	Variable	Bedeutung
double	$*u$	gibt die optimale Kontrolle zurück
double	$*max\_val$	gibt den Wert der optimalen Wertefunktion in x zurück

Ablauf:

1. (a) Setze $current\_max = -\infty$
   (b) Für $i = 0, 1$
      i. Setze die Diskretisierung $abstand[i]$ der Kontrolle $u[i]$ auf $anfdiskr \cdot (u_{max}[i] - u_{min}[i])$. Dabei ist $u_{max}[i]$ die obere und $u_{min}[i]$ die untere Schranke des Kontrollwertebereichs U aus Tabelle 6.1.
      ii. Setze $u\_min[i] = u_{min}[i]$ und $u\_max[i] = u_{max}[i]$

2. Setze $current\_u[0] = u\_min[0] - abstand[0]$
   (a) Berechne $current\_u[0] = current\_u[0] + abstand[0]$
   (b) Setze $current\_u[1] = u\_min[1] - abstand[1]$
   (c) if $current\_u[0] > u\_max[0]$, dann setze $current\_u[0] = u\_max[0]$
      i. Berechne $current\_u[1] = u\_min[1] + abstand[1]$
      ii. if $current\_u[1] > u\_max[1]$, dann setze $current\_u[1] = u\_max[1]$
      iii. Berechnung von *temp*: siehe unten (nach Schritt 5.)
      iv. if $temp > current\_max$, dann setze $u[i] = current\_u\_[i]$ für $i = 0, 1$, und $current\_max = temp$
      v. Wenn $current\_u[1] < u\_max[1]$, dann gehe zu 2.(c) i.
      sonst: wenn $current\_u[0] < u\_max[0]$, dann gehe zu 2.(a)
      sonst: gehe zu 3.

3. if $abstand[0] < abbruch$ und $abstand[1] < abbruch$, dann gehe zu 5.
   sonst: gehe zu 4.

4. Verkleinerung von $abstand[i]$ zur genaueren Suche in einer Umgebung des optimalen $(u[0], u[1])$: Für $i = 0, 1$
   (a) $abstand[i] = diskrverkl \cdot abstand[i]$
   $u\_max[i] = u[i] + umgebung \cdot abstand[i]$
   $u\_min[i] = u[i] - umgebung \cdot abstand[i]$
   (b) wenn $u\_min[i] < u_{min}[i]$, dann setze $u\_min[i] = u_{min}[i]$
   wenn $u\_max[i] > u_{max}[i]$, dann setze $u\_max[i] = u_{max}[i]$
   (c) gehe zu 2.

5. Setze $max\_val = current\_max$

Berechnung von *temp*:

1. Es werden dabei drei Fälle unterschieden (je nach Eingabeparameter):

   (a) $\sigma_{EK} = 0$ und $\sigma_{FK} = 0$
   einmal Aufruf von *Stoch_Euler_Verfahren(Gitter_rFK, Gitter_rEK, x, current_u, z, erg, K)* mit $z[0]$ und $z[1]$ beliebig gewählt

   (b) $\sigma_{EK} \neq 0$ und $\sigma_{FK} = 0$ oder $\sigma_{EK} = 0$ und $\sigma_{FK} \neq 0$
   zweimal Aufruf von *Stoch_Euler_Verfahren(Gitter_rFK, Gitter_rEK, x, current_u, z, erg, K)* mit $(z[0] = \sqrt{h}, z[1] = -\sqrt{h})$ bzw. mit $(z[0] = -\sqrt{h}, z[1] = \sqrt{h})$

   (c) $\sigma_{EK} \neq 0$ und $\sigma_{FK} \neq 0$
   viermal Aufruf von *Stoch_Euler_Verfahren(Gitter_rFK, Gitter_rEK, x, current_u, z, erg, K)* mit $(z[0] = \sqrt{h}, z[1] = -\sqrt{h})$, mit $(z[0] = -\sqrt{h}, z[1] = \sqrt{h})$, mit $(z[0] = \sqrt{h}, z[1] = \sqrt{h})$ und mit $(z[0] = -\sqrt{h}, z[1] = -\sqrt{h})$

2. Die zurückgegebenen Werte erg müssen dann noch darauf überprüft werden, ob sie im Gitter liegen. Bei der Zeitkomponente ($erg[1]$) wird beim Überschreiten der Gittergrenze der Modulowert von $erg[1]$ und $hi[1]$ genommen. Bei der Vermögenskomponente ($erg[0]$) wird $erg[0]$ bei Unterschreiten der Gittergrenze auf $lo[0]$ und bei Überschreiten der Gittergrenze auf $hi[0]$ gesetzt.

3. Berechnung der optimalen Wertefunktion (*temp*)

Dabei sind h, $hi[i]$ und $lo[i]$ $(i = 0, 1)$ sowie σ_{EK} und σ_{FK} Parameter, welche in $*K$ abgespeichert sind, aus Tabelle 6.1.
In Abbildung 5.2 wird die Funktionsweise von *void finde_max_besser* (für die Parameter $anfdiskr = 0.2$, $diskrverkl = 0.5$ und $umgebung = 2$), insbesondere Punkt 4. des Programmablaufes, graphisch veranschaulicht.

Konvergenz des Verfahrens zu dem Maximum:
Die optimale Wertefunktion ist nach Satz 4.9 konkav in der Komponente x_0 und das zu maximierende Funktional $J_T(x, u)$ konkav in x_0 und $u = (u_0, u_1)$, d.h. falls ein lokales maximales u gefunden wird, so ist es auch global (auf dem vorgegebenen Gitter) maximal. Der Algorithmus liefert aber nur das diskrete Maximum. Dieses muss aber nicht mit dem kontinuierlichen übereinstimmen. Man kann konkave Funktionen konstruieren, wo die diskreten Punkte um das kontinuierliche Maximum einen kleineren Wert haben als beim diskreten Maximum, d.h. der Algorithmus verfeinert im nächsten Schritt die Suche beim diskreten Maximum und kann daher das kontinuierliche nicht finden. Dazu vergleiche Abbildung 5.3.[2]
Wie gut der Algorithmus für verschiedene Parameterkonstellationen für die Modelle

[2]Die Abbildung 5.3 beruht auf einer Idee/Skizze von Herrn Professor Lempio.

(4.1) und (4.6) funktioniert, wird in Abschnitt 6.2.1 untersucht.

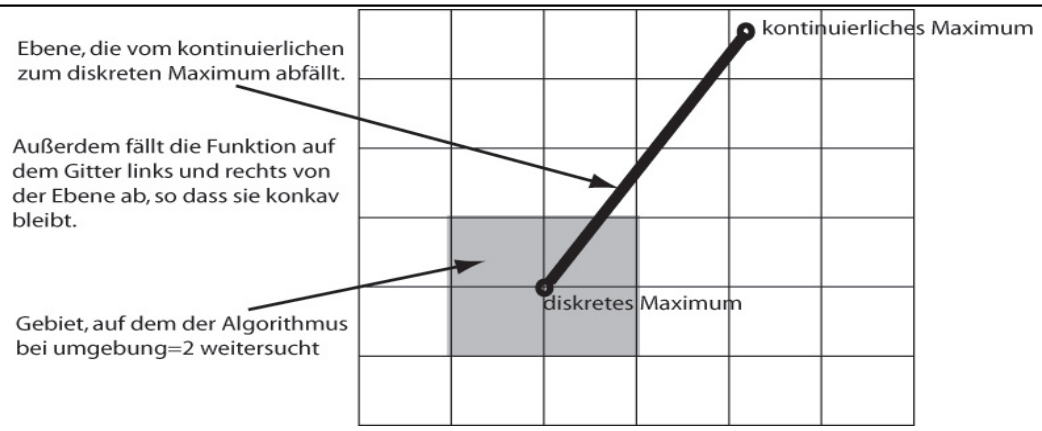

Abbildung 5.3: Beispiel, wo *void finde_max_besser* versagt

void werte_iteration(qgrid *Gitter_V, qgrid *Gitter_V_temp, qgrid *Gitter_u0, qgrid *Gitter_u1, qgrid *Gitter_rFK, qgrid *Gitter_rEK, struct Konstanten *K, float *geszeit, int *iterationen)

In der Werteiteration wird das Startgitter (in Iterationsschritt 0) zunächst in jedem Punkt $x = (x[0], x[1])$ auf 0 gesetzt, da es keine Endkosten gibt (vergleiche Definition 3.13). Das entstehende Gitter wird in den Ordner \Modell\1 mit dem Namen Wertefkt_gitter0.val mit Hilfe der *gridgen*-Funktion *save_val* abgespeichert.
In jedem folgenden Iterationsschritt $i = 1, 2, 3, ...$ wird zunächst mit der *gridgen*-Funktion *load_val* das Gitter $i - 1$ geladen und in der Variable *Gitter_V* abgespeichert. Dann werden für sämtliche Gitterpunkte x von *Gitter_V_temp* die Werte wie folgt berechnet:

1. Aufruf von finde_max_besser(Gitter_V,Gitter_rFK,Gitter_rEK,x,K,u,&max_val) Dabei werden die Werte von Gitter_V bei der Berechnung der optimalen Wertefunktion (siehe Berechnung von *temp*, Seite 68) verwendet.

2. Die zurückgegebenen Werte $u = (u[0], u[1])$ und *max_val* werden mit *set_current_nodevalue* in den Gittern *Gitter_u*0, *Gitter_u*1 und *Gitter_V_temp* gespeichert.

3. Es wird die absolute Differenz der Werte im Punkt x der beiden Gitter *Gitter_V_temp* und *Gitter_V* gebildet und, wenn diese größer als die Differenz der bisher durchgelaufenen Gitterpunkte ist, in der Variable *maximum* abgespeichert.

Nach jedem Iterationsschritt $i > 0$ wird *Gitter_V_temp* mit *save_val* als Wertefkt_gitter**(i)**.val abgespeichert und das zuletzt abgespeicherte Wertefkt_gitter**(i-1)**.val[3]

[3]für **(i)** bzw. **(i-1)** wird die entsprechende Zahl gemäß dem Iterationsschritt i gesetzt

mit der C-Funktion *remove* gelöscht.
Wenn nach durchlaufen aller Gitterpunkte $x\ maximum < tol$ (vergleiche Tabelle 6.1) ist, dann wird die Werteiteration abgebrochen.[4] Dann werden neben $Gitter\_V\_temp$ auch $Gitter\_u0$ und $Gitter\_u1$ mit *save_val* in \Modell\1 als $\left\{\begin{array}{c} Wertefkt \\ u0 \\ u1 \end{array}\right\}$ *_gitter***iter.***val* und mit *export_gridandval* in \Modell\2 als $\left\{\begin{array}{c} Wertefkt \\ u0 \\ u1 \end{array}\right\}$ *_gitter***iter.***asc* abgespeichert, wobei **iter** die Anzahl der benötigten Iterationsschritte angibt.
In der Konsole wird jeder Iterationsschritt mit der gebrauchten Zeit, die Gesamtzeit des Algorithmus bis dahin sowie der Wert von *maximum* ausgegeben.

int main(int args, char $**$argv)

Im Hauptprogramm werden nach dem Einlesen der Parameter mittels der Funktion *eingabe* die Zeiger für die Gitter $Gitter\_V\_temp$, $Gitter\_V$, $Gitter\_u0$, $Gitter\_u1$, $Gitter\_rFK$ und $Gitter\_rEK$ angelegt. $Gitter\_rFK$ und $Gitter\_rEK$ werden dann mit *load_val* aus dem Ordner \Zinsen eingelesen (Dateinamen müssen dabei sein: $rFK\_gitter.val$ und $rEK\_gitter.val$). Dem Anwender stehen dann folgende Möglichkeiten zur Verfügung:

1. Berechnung der optimalen Wertefunktion:
   (a) Eingabe von 1:
   Dann werden mit *create_grid* $Gitter\_V\_temp$, $Gitter\_V$, $Gitter\_u0$ und $Gitter\_u1$ mit den gegebenen Parametern erzeugt und damit die Funktion *werte_iteration* und die Funktion *statistiken* aufgerufen.
   (b) Eingabe von 2 überspringt diesen Schritt
2. Berechnung der optimalen Trajektorien:
   (a) Eingabe von 1:
   Wenn die optimale Wertefunktion zuvor nicht berechnet wurde (Eingabe von 2), muss noch die Nummer **iter** des Gitter eingeben werden (siehe Beschreibung zur Funktion *werte_iteration*).
   Nach Laden von $Gitter\_u0$ und $Gitter\_u1$ wird Stoch_trajektorie mit in den Parametern vorgegebenen $x0[0]$ und $x0[1]$ sowie mit $x0[0] = hi[0]$ und mit $x0[0] = 0.5 \cdot hi[0]$ aufgerufen.
   Anschließend kann der Anwender noch weitere stochastische Trajektorien mit beliebigen manuell eingegebenen $x0[0]$ und $x0[1]$ berechnen lassen.
   (b) Eingabe von 2: Überspringen des Schritts, Ende des Programms *Portfolio*.

[4] Dies entspricht dem Kriterium in Gleichung (3.47)

5.2 Weitere Programme

Alle weiteren Programme befinden sich in den Unterverzeichnissen
CD:\Programme\Weitere Programme\Unterabschnittsname.

5.2.1 *Binde_in_Gitter.c*

Input: Zwei Datenreihen, eine für die Aktienrendite (risikobehaftete Anlage) und eine für den Zinsertrag des Wertpapiers (risikoarme Anlage). Dabei sollten die Daten in einer Spalte stehen, damit sie eingelesen werden können, und die Anzahl bei beiden Datenreihen gleich groß sein.

Output: Im Unterordner *Zinsgitter* werden die zwei Datenreihen in Form eines Gitters als .val-Datei (für das Hauptprogramm) und als .asc-Datei (für die graphische Darstellung der Zinsverläufe mit Hilfe von Matlab) abgespeichert.
Außerdem wird noch die Datei *Statistiken.txt* erzeugt, in der die Anzahl der Zinsdaten und die zugehörige Gitterkonstante $hi[1]$ notiert sind.

5.2.2 *Zinserzeugung.c*

Input: In der Eingabedatei *eingabe.txt* muss die Anzahl der Zinswerte (in der Zeile *Zeit*) und die gewünschten Erwartungwerte (bei *EK_EW* und *FK_EW*), die gewünschten Standardabweichungen (bei *EK_sigma* und *FK_sigma*) und die Korrelation (bei *Korrelation*) der zwei zu erzeugenden Zinsverläufe angeben werden.

Output: Ausgegeben werden zwei normalverteilte Zinsverläufe im Ordner *\Zinsen* mit den Namen *EK_filename* und *FK_filename*, wobei *filename* während des Programmablaufs vom Benutzer eingegeben wurde. Um die Daten aber für das Hauptprogramm verwenden zu können, muss man diese noch in ein Gitter einbinden und als .val-Datei abspeichern. Dies geschieht durch das Programm *Binde_in_Gitter*.

5.2.3 Matlab

Es soll nun im Folgenden ein Überblick der verwendeten Matlab-Programme gegeben werden:

- *Dichtefunktion.m*
  Aufruf: *Dichtefunktion('name.txt')*
  Gibt ein Approximation der Dichtefunktion für eine Menge von Zufallszahlen zurück.
  Als Input kann man die in *Zinserzeugung.c* erzeugte Datei *name.txt* bzw. eine Datei gleichen Aufbaus nehmen.

- *differenz.m*
  Aufruf: *differenz('name1.asc','name2.asc','name3.asc')*
  Berechnet die Differenz der Werte der zwei Gitter (name1.asc-name2.asc).
  Ausgabe: in Datei *name3.asc*
  (name1, name2, name3 können auch die Pfadangabe der Datei enthalten)

- *gridplot2d.m* (basierend auf *gridplot2d.m* von Prof. Grüne, modifiziert in der Darstellung)
  Aufruf: gridplot2d('gitter.asc')
  Dient der graphischen Darstellung einer mittels „export_gridandval“ exportierten Gitterfunktion in 2d.

- *maxmin.m*
  Aufruf: maxmin('gitter.asc')
  Programm um den minimalen und maximalen Wert sowie den Mittelwert und die Standardabweichung auf einem Gitter zu finden.
  Gibt die Ergebnisse in der Matlab-Konsole zurück.

- *plot_konvergenz.m*
  Aufruf: plot_konvergenz('konvergenz.txt')
  Bei der Werteiteration verändert sich das Gitter der optimalen Wertefunktion. Die größte Differenz der Gitterpunkte sollte bei Konvergenz dabei immer kleiner werden. *plot_konvergenz.m* plottet nun diesen Wert und gibt zurück, ob und wie oft er ansteigt (Ein Anstieg ist schlecht für das Konvergenzverhalten).

- *plot_zins.m*
  Aufruf: plot_zins('zins1.asc','zins2.asc')
  Plottet 1 oder 2 Zinsverläufe (bei einem: Aufruf mit plot_zins('zins1.asc',' ').
  Die Zinsen müssen dabei in einem eindimensionalen Gitter abgespeichert sein.

- *Trajekplot.m*
  Aufruf: Trajekplot('trajektorie.txt')
  Plottet die vom Hauptprogramm berechneten optimalen Trajektorien: Vermögen, Konsum, EK-Quote (Anteil des Vermögens, welches in die risikoreiche Anlage investiert wurde).

- *Zinsstatistik.m*
  Aufruf: Zinsstatistik('zins.asc')
  Berechnet zu zwei Zinsverläufen den Erwartungswert, die Varianz und die Kovarianz sowie Korrelation.
  Ausgabe: in Datei *Zinsdaten.txt*

Kapitel 6

Auswertung der numerischen Ergebnisse

	Parameter		**Wert**
	Problemparameter		
•	Diskontfaktor $\beta = 1 - \delta$	δ	0.05
•	Risikoaversion	γ	0.9
•	Kontrollwertebereich	U	$[0, 0.7] \times [-3, 4.5]$
•	Startwert für die optimale Trajektorie	$x0[0] \times x0[1]$	1×0
•	Endwert (Zeit) für die optimale Trajektorie	x_max	200
	Parameter des stochastischen Modells		**Wert**
•	Gewichtungsfaktor für die risikoreiche Anlage	σ_{EK}	0.5
•	Gewichtungsfaktor für die risikoarme Anlage	σ_{FK}	0.0
	Numerische Parameter		
•	Fehlervermeidung/Abbruchkriterium	ϵ	0.000001
•	Toleranz für den Abbruch der Werteiteration	tol	0.1
•	Schrittweite	h	0.1
•	Anfangsdiskretisierung von u	$anfdiskr$	0.1
•	Verkleinerung der Diskretisierung von u	$diskrverkl$	0.5
•	Größe der Umgebung um das Optimum der aktuellen Diskretisierung (in Gitterpunkten der verkleinerten Diskretisierung)	$umgebung$	2
•	Genauigkeit, die die Kontrolle u erreichen soll	$abbruch$	0.001
	Gitterparameter		
•	obere Schranke	$lo[0] \times lo[1]$	0×0
•	untere Schranke	$hi[0] \times hi[1]$	100×73
•	Größe eines Gitterrechteckes	$Delta[0] \times Delta[1]$	1×1

Tabelle 6.1: Parameter im Standardmodell

6.1 Verwendete Standardparameter

Für das deterministische und das stochastische Modell werden die Parameter in Tabelle 6.1 definiert. Dabei erhält man das deterministische, indem man die zusätzlichen Variablen für das stochastische Modell auf 0 setzt. Die Schrittweite h sollte außerdem nicht kleiner als die Toleranz für Abbruch der Werteiteration *tol* sein, da sonst der Fortschritt der Werteiteration so klein ist, dass direkt in der Iteration 1 abgebrochen wird.

6.2 Vorbetrachtungen

Bevor Untersuchungen zu den Modellen (4.1) und (4.6) mit Hilfe des Algorithmus aus Abschnitt 5.1 gemacht werden können, wird zunächst analysiert, welche Parameter optimal für den Optimierungsalgorithmus *finde_max_besser* sind. Dabei soll ein effizientes Verhältnis zwischen Genauigkeit der optimalen Kontrollen u und Laufzeit des Algorithmus gefunden werden.
Danach wird der Einfluss der Wahl des Parameters $hi[0]$ (Vermögen) auf die optimale Wertefunktion und Kontrollen untersucht.
Der in Abbildung 4.1 dargestellte Schweizer Aktien- bzw. Obligationsindex liefert die Daten für r_e bzw. r_f aus Modell (4.1) bzw. (4.6). Sofern nichts anderes angegeben wurde, werden in diesem Abschnitt die Standardparameter aus Tabelle 6.1 verwendet.

6.2.1 Variation der Parameter von *finde_max_besser*

In Tabelle 6.2 sind verschiedene Parameterkombinationen von *finde_max_besser* (anfdiskr, diskrverkl, umgebung) dargestellt. Dabei wird der Fall f als Referenzlösung (wegen der starken Startdiskretisierung $anfdiskr$) für die Fälle 1 bis 11 sowie a bis e genommen.
Der kleinste (min) und größte (max) Wert sowie der Mittelwert (μ) und die Standardabweichung (σ) aus Tabelle 6.3 wurden mit dem Matlab-Programm *maxmin.m* berechnet. In der Teiltabelle *Absolutwerte von a bis f* wurde *maxmin.m* einfach auf das entsprechende Gitter der optimalen Wertefunktion von $a, ..., f$ angewendet. Bei *Vergleich zu der Referenzlösung f* wurde von den Werten der Gitter der optimalen Wertefunktion der Referenzlösung f die Werte aus dem Gitter der optimalen Wertefunktion der Lösung i ($i = 1, ..., 11, a, ..., e$) mit dem Matlabprogramm *differenz.m* subtrahiert und auf das entstehende Gitter (im Folgenden mit $(f - i)$ bezeichnet) *maxmin.m* angewendet.

[1]Sämtliche Parameterkonstellationen wurden mit einem Universitäts-PC (AMD Opteron 254 mit 1.81GHz und 1024kB Cache) gerechnet

Nr.	anfdiskr	diskrverkl	umgebung	Laufzeit des Algorithmus[1]
a	0.075	0.5	2	2574.79s
b	0.05	0.5	2	3579.46s
c	0.025	0.5	2	10250.29s
d	0.01	0.5	2	58244.57s
e	0.005	0.5	2	213954.05s
f	0.0025	0.5	2	865816.94s
1	0.1	0.5	2	2192.34s
2	0.1	0.5	3	5659.15s
3	0.1	0.5	4	5249.11s
4	0.1	0.5	5	7369.82s
5	0.1	0.5	8	15890.64s
6	0.2	0.5	2	1803.19s
7	0.2	0.5	3	3271.70s
8	0.2	0.5	4	5020.18s
9	0.2	0.5	5	7460.21s
10	0.2	0.5	8	16635.12s
11	0.1	0.1	11	9428.35s

Tabelle 6.2: Variation der Parameter von *finde_max_besser*

Die relativen Werte von min, max, μ und σ ergeben sich wie folgt:

$$
\begin{aligned}
rel.\,min_i &= \frac{min_{(f-i)}}{min_f} \quad i = 1, ..., 11, a, ..., e \\
rel.\,max_i &= \frac{max_{(f-i)}}{max_f} \quad i = 1, ..., 11, a, ..., e \\
rel.\,\mu_i &= \frac{\mu_{(f-i)}}{\mu_f} \quad i = 1, ..., 11, a, ..., e \\
rel.\,\sigma_i &= \frac{\sigma_{(f-i)}}{\sigma_f} \quad i = 1, ..., 11, a..., e
\end{aligned}
$$

Dabei stehen die Werte $rel.\,min_i$, $rel.\,max_i$, $rel.\,\mu_i$ bzw. $rel.\,\sigma_i$ sowie $min_{(f-i)}$, $max_{(f-i)}$, $\mu_{(f-i)}$ bzw. $\sigma_{(f-i)}$ in der Tabelle 6.3 in der Zeile *Nr. i* ($i = a, ..., e, 1, ..., 11$).

Fazit:

Die Referenzlösung ist im Mittel die mit der höchsten Approximationsgenauigkeit, aber auch die zeitintensivste. Daher soll nun die Teiltabelle *Vergleich zu der Referenzlösung f* ausgewertet werden um eine zeitsparende ähnlich genaue Parameterkombination zu finden. Wenn die relativen Minimal-, Maximal- und Mittelwerte sowie die Standardabweichung des Differenzgitters $(f-i)$ $(i = 1, ..., 11, a, ..., e)$ nahe bei 0 sind, dann kann davon ausgegangen werden, dass das Gitter *Nr. i* entsprechend genaue Werte wie Gitter f hat. Die kleinsten Werte sind in der Tabelle 6.3 fett markiert.

Absolutwerte von a bis f				
Nr.	min	max	μ	σ
a	-180.161690	45.527823	25.672469	26.683834
b	-180.161690	45.602198	25.717991	26.642774
c	-180.161690	45.585403	25.709293	26.639887
d	-180.161690	45.603561	25.720292	26.643525
e	-180.161690	45.619713	25.727203	26.646934
f	-180.161690	45.624605	25.731460	26.645635

Vergleich zu der Referenzlösung f								
Nr.	min	max	μ	σ	*rel.* min	*rel.* max	*rel.* μ	*rel.* σ
a	0.000000	1.794503	0.058991	0.118062	**0.000000**	0.039332	0.002293	0.004431
b	-0.015103	0.131256	0.013470	0.007759	0.000084	0.002877	0.000523	0.000291
c	-0.014599	0.131645	0.022167	0.011855	0.000081	0.002885	0.000861	0.000445
d	-0.006558	0.049859	0.011168	0.005150	0.000036	0.001093	0.000434	0.000193
e	-0.000391	0.109477	0.004257	0.005717	0.000002	0.002400	**0.000165**	0.000215
1	-0.015593	0.130960	0.007797	0.005343	0.000087	0.002870	0.000303	0.000201
2	-0.018588	0.086817	0.007976	0.005797	0.000103	0.001903	0.000310	0.000218
3	-0.028135	0.068984	0.007440	0.005905	0.000156	0.001512	0.000289	0.000222
4	-0.031717	0.027711	0.005430	0.004474	0.000176	0.000607	0.000211	**0.000168**
5	-0.071429	0.028826	0.004615	0.006668	0.000396	0.000632	0.000179	0.000250
6	-0.015593	0.130960	0.007745	0.005350	0.000087	0.002870	0.000301	0.000201
7	-0.018588	0.086817	0.007969	0.005797	0.000103	0.001903	0.000310	0.000218
8	-0.028135	0.068984	0.007440	0.005905	0.000156	0.001512	0.000289	0.000222
9	-0.031717	0.027711	0.005430	0.004474	0.000176	0.000607	0.000211	**0.000168**
10	-0.071429	0.028826	0.004615	0.006668	0.000396	0.000632	0.000179	0.000250
11	-0.051266	0.024471	0.004893	0.004505	0.000285	**0.000536**	0.000190	0.000169

Tabelle 6.3: Vergleich zu der Referenzlösung aus f (Optimale Wertefunktion)

Zusammenfassend kann man sagen, dass sich sämtliche Lösungen 1 bis 11 bzw. b bis e in ihren relativen Mittelwerten und Standardabweichungen zu der Referenzlösung f (welche als Anfangsgitter eines mit 401×401 Gitterpunkten hat) erst in der 4. Stelle nach dem Komma unterscheiden. Welcher Algorithmus nun genommen wird, hängt von den Präferenzen des Anwenders ab: Für eine höchstmögliche Genauigkeit empfiehlt sich *Nr.4, 5, 9, 10* oder *Nr.11*, welche einen relativen Mittelwert ≤ 0.000211, Standardabweichung ≤ 0.000250, max ≤ 0.000632 und min ≤ 0.000396 haben. Zeitlich am besten schneidet von diesen Varianten 4 und 9 ab, die mit $\approx 7400s$ nur circa 0.85% der Zeit von Variante f brauchen.
Für eine hohe Rechengeschwindigkeit empfiehlt sich Variante 1 bzw. 6, welche mit $\approx 2200s$ nur 0.25% der Zeit von Variante f bzw. nur 30% von der von 4 bzw. 9 benötigen. Dafür büßt man bei der relativen Standardabweichung im Vergleich zu 4 0.000033 (oder $\approx 20\%$) und beim relativen Mittelwert 0.000092 (oder $\approx 44\%$) ein.
Bei den folgenden Anwendungen des Programms werde ich auf Variante 1 zurückgreifen, da es bei hoher Rechengeschwindigkeit trotzdem eine sehr genaue (nahe an Variante f liegende) Lösung liefert.

Variation von hi[0] und Delta[0]				
Nr.	hi[0]	Delta[0]	Anzahl Gitterpunkte in hi[0]	Laufzeit des Algorithmus
a	1	0.001	1000	20920.14s
b	1	0.01	100	2118.21s
c	1	0.1	10	219.07s
d	10	0.01	1000	21065.34s
e	10	0.1	100	2115.56s
f	10	1	10	225.29s
g	100	0.1	1000	20905.96s
h	100	1	100	2102.54s
i	100	10	10	225.81s
j	1000	1	1000	20937.37s
k	1000	10	100	2113.06s
l	1000	100	10	226.25s

Auswertung der verschiedenen Parameterkonstellationen von hi[0] und Delta[0]				
Nr.	min	max	μ	σ
a	0.000000	0.410020	0.089498	0.032900
b	0.000000	0.350342	0.084620	0.036276
c	0.000000	0.211504	0.054758	0.031762
d	0.000000	0.410020	0.089499	0.032895
e	0.000000	0.350342	0.084622	0.036278
f	0.000000	0.211504	0.054760	0.031760
g	0.000000	0.410020	0.089498	0.032894
h	0.000000	0.350342	0.084626	0.036291
i	0.000000	0.211504	0.054760	0.031760
j	0.000000	0.404277	0.089497	0.032912
k	0.000000	0.350342	0.084611	0.036245
l	0.000000	0.211504	0.054757	0.031758

Auswertung ausgewählter Differenzgitter mit gleicher Anzahl von Gitterpunkten					
Nr. der 2 Gitter	Gitterpunkte	min	max	μ	σ
a-d	1000	-0.063984	0.058721	-0.000001	0.000904
a-g	1000	-0.038486	0.066445	-0.000000	0.000953
a-j	1000	-0.047305	0.063984	0.000001	0.001291
b-e	100	-0.002598	0.001914	-0.000002	0.000128
b-h	100	-0.047100	0.005332	-0.000006	0.000567
b-k	100	-0.015381	0.063984	0.000009	0.000831
c-f	10	-0.002051	0.000068	-0.000002	0.000072
c-i	10	-0.002051	0.000068	-0.000002	0.000072
c-l	10	-0.002051	0.002734	0.000001	0.000120

Tabelle 6.4: Variation von hi[0] und Delta[0], Vergleich der Kontrolle u[0]

6.2.2 Variation von hi[0]

In diesem Abschnitt wird untersucht, welche Auswirkung eine Veränderung von $hi[0]$ (Obergrenze Vermögen) auf die optimalen Kontrollen und die optimale Wertefunktion hat. Des Weiteren wird die Größe des Gitterrechteck $Delta[0]$ variiert um den Einfluss der Anzahl der Gitterpunkte[2] auf die Lösung zu untersuchen. In Tabelle 6.4 sind dazu die Ergebnisse der optimalen Kontrolle $u[0]$ für verschiedene $hi[0]$ und $Delta[0]$ zusammengetragen[3]. Die Ergebnisse bezüglich der optimalen Wertefunktion sind in Tabelle 6.5 dargestellt.
Wenn man die optimale Wertefunktion vergleicht, dann muss beachtet werden, dass nur die mit gleicher oberer Schranke $hi[0]$ vergleichbar sind, da ein größeres $hi[0]$ (maximales Vermögen) auch ein höher mögliches $x[0]$ und damit eine höhere Power-Nutzenfunktion bedeutet (vergleiche Gleichung (4.3): $hi[0]$ ist obere Schranke für $x[0]$).
Fazit: Bei Betrachtung von Tabelle 6.4 fällt auf, dass sich die Ergebnisse von Gittern der Kontrolle $u[0]$ mit gleicher Anzahl von Gitterpunkten im Mittelwert um maximal 0.000009 und in der Standardabweichung um maximal 0.001291 unterscheiden. Daraus kann geschlussfolgert werden, dass die Wahl von $hi[0]$ keinen Einfluss auf die Kontrolle $u[0]$ hat, wenn die Anzahl der Gitterpunkte gleich bleibt.
Wenn man andererseits Gitter der Kontrolle $u[0]$ mit unterschiedlicher Anzahl von Gitterpunkten anschaut (z.B. a und b,...), dann erkennt man, dass eine höhere Anzahl von Gitterpunkten zu einem höheren mittleren Wert und Maximalwert (max) von $u[0]$ führt. Das selbe Ergebnis bekommt man, wenn man die optimale Wertefunktion in Tabelle 6.5 für die Fälle mit gleichem $hi[0]$ und unterschiedlichem $Delta[0]$ vergleicht (z.B. $Nr.a$ und $Nr.b$). Besonders groß ist der Unterschied des mittleren Wertes der optimalen Wertefunktion, wenn man die Ergebnisse der Gitter mit 10 und mit 100 Gitterpunkten miteinander vergleicht (z.B. $\mu_h = 25.723663$ und $\mu_i = 2.223853$, wobei $hi[0] = 100$ bei beiden gilt). Beim Vergleich von Fällen mit 100 und 1000 Gitterpunkten ist der maximale Unterschied der mittleren Werte der optimalen Wertefunktion bei $\approx 15\%$ (Vergleich von $Nr.d$ und $Nr.e$).
Um eine gute Approximation der optimalen Wertefunktion zu erhalten, sollte die Anzahl der Gitterpunkte möglichst hoch gewählt werden. Die Verdopplung der Gitterpunkte nur in $x[0]$-Richtung bedeutet aber schon eine Verdopplung des Zeitaufwands (vgl. dazu den Zeitaufwand des Algorithmus in Tabelle 6.4). Daher werden die folgenden Untersuchungen in Abschnitt 6.3 und 6.4 mit 100 Gitterpunkten in $x[0]$-Richtung durchgeführt. $hi[0]$ wird dabei auf 100 und $Delta[0]$ auf 1 gesetzt.
Die Wahl von $hi[0] = 100$ hat den Vorteil, dass man die Werte des Vermögens 1 zu 1 in Prozentwerte übertragen kann. Zusammen mit der Tatsache, dass $hi[0]$ keinen Einfluss auf die Kontrollen u hat (wenn die Anzahl der Gitterpunkte konstant bleibt),

[2]Hier bezeichnet die *Anzahl der Gitterpunkte*, die die sich zwischen lo[0] und hi[0] befinden. Die Zahl der Gitterpunkte in x[1]-Richtung bleibt im Folgenden konstant.

[3]Auf eine Darstellung der Ergebnisse der optimalen Kontrolle $u[1]$ soll hier verzichtet werden, da sie denen von $u[0]$ entsprechen. Die Ergebnisse finden sich auf der CD im Ordner \Ergebnisse\Schweizer Aktienindex\Testhi[0]\vergleichu1\ in der Datei *Ergebnisse.pdf* wieder.

kann das Modell auf verschiedenste Start- und Maximalvermögen angewendet werden. Wenn man z.B. ein Startvermögen von 5000€ und ein Zielvermögen (= $hi[0]$) von 50000€ hat, dann wählt man (für die Berechnung der optimalen Trajektorie) $x0[0] = 10$ mit $hi[0] = 100$.

Auswertung der verschiedenen Parameterkonstellationen von hi[0] und Delta[0]				
Nr.	min	max	μ	σ
a	-180.161690	-37.610998	-48.560671	10.687029
b	-180.161690	-37.709662	-50.257401	16.811640
c	-180.161690	-40.831607	-65.084792	37.766503
d	-180.161690	-0.702330	-14.486666	13.453773
e	-180.161690	-0.824730	-16.621494	21.164626
f	-180.161690	-4.755000	-35.288045	47.545343
g	-180.161690	45.765730	28.411857	16.937542
h	-180.161690	45.610606	25.723663	26.644721
i	-180.161690	40.662916	2.223853	59.856178
j	-180.161690	104.267899	82.417413	21.323370
k	-180.161690	104.069067	79.033059	33.543765
l	-180.161690	97.840256	49.448516	75.354571

Tabelle 6.5: Variation von hi[0] und Delta[0], Vergleich der optimalen Wertefunktion

6.3 Das deterministische Modell

In diesem Abschnitt werden die Standardparameter aus Tabelle 6.1 mit $\sigma_{EK} = 0$ verwendet. Wenn Parameter abweichend davon gewählt werden, dann wird dies explizit angegeben.
Die Untersuchung des deterministischen Falles dient einerseits dazu eine Referenzlösung für die Lösungen der stochastischen Fälle zu haben, und andererseits dazu die Auswirkungen einer Variantion des Kontrollwertebereichs U auf die Lösung zu untersuchen. Dies wird in Abschnitt 6.3.1 bzw. 6.3.2 anhand des Standard&Poor's bzw. Schweizer Aktien-/Obligationenindexes gemacht. In Abschnitt 6.3.3 wird der Einfluss der Korrelation ρ zwischen den zwei Anlagen auf die optimale Wertefunktion und Kontrolle $u[0]$ untersucht.

6.3.1 Indizes von Standard&Poor's

Folgende Fälle werden betrachtet:

Beispiel	$U = [u_{min}[0], u_{max}[0]] \times [u_{min}[1], u_{max}[1]]$	Abbildung
SP_det_a	$[0, 0.7] \times [-3, 4.5]$	6.4 & 6.1 & 6.6
SP_det_b	$[0, 0.7] \times [0, 1]$	6.4 & 6.2 & 6.6
SP_det_c	$[0, 1] \times [-10, 10]$	6.5 & 6.3 & 6.7

Für die Zinsen, welche in Abbildung 4.1 dargestellt sind, gelten folgende Daten:

Zeit	$\mathbb{E}(S\&P500)$	$\sigma(S\&P500)$	$\mathbb{E}(10J-WP)$	$\sigma(10J-WP)$	ρ
305	0.004552	0.012285	0.003025	0.001610	0.267928

Aufgrund der Zahl der Zinsdaten (305) wird $hi[1] = 304$ gesetzt und $x\_max = 600$ gewählt. Da für die Zinsen monatliche Daten gegeben sind, muss der Diskontfaktor $\delta = 0.0043$ betragen.[4]
Auswertung:
In Abbildung 6.4 und 6.5 werden die optimalen Kontrollen $u[1]$, welche den Anteil des investierten Vermögens, hier in den Standard&Poor's 500 Index (risikoreiche Anlage), dargestellt. Wenn man die Abbildungen in der $x[1]$-Richtung mit dem Verlauf der Zinskurven aus Abbildung 4.1 vergleicht, dann sieht man, dass das gesamte Vermögen (entsprechend der oberen bzw. unteren Schranke $u_{max}[1]$ bzw. $u_{min}[1]$) in die Anlage mit der höheren Rendite investiert wird. Ein $u[1] > 1$ bedeutet dabei, dass ein Anteil $u[1] - 1$ des Vermögens von der risikolosen Anlage (hier: 10-Jahres-Wertpapier) leerverkauft werden muss. Das daraus gewonnene Kapital wird zusätzlich in die risikoreiche Anlage investiert, also insgesamt das $u[1]$-fache Vermögen. Für ein negatives $u[1]$ gilt das entsprechende mit dem Leerverkauf der risikoreichen Anlage. Dadurch kann die Rendite zusätzlich erhöht werden.
Bei Betrachtung von Abbildung 6.1, 6.2 und 6.3 kann man erkennen, dass, umso größer bzw. kleiner man die obere bzw. untere Grenze von $u[1]$ wählen kann, desto höher ist der Konsum und damit auch die optimale Wertefunktion.

[4] $(1-\delta)^{12} = 0.9496 \approx 1 - 0.05$

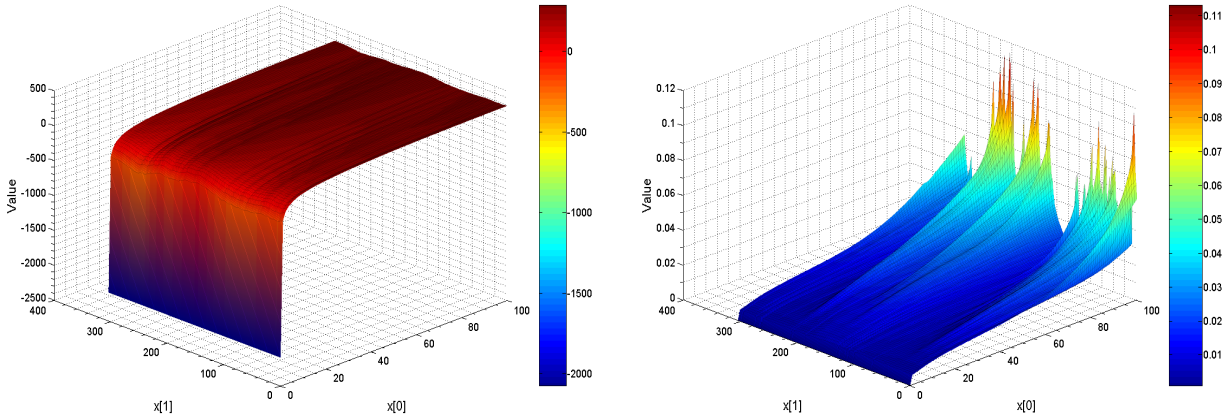

Abbildung 6.1: Optimale Wertefunktion und Kontrolle $u[0]$ für Beispiel SP_det_a

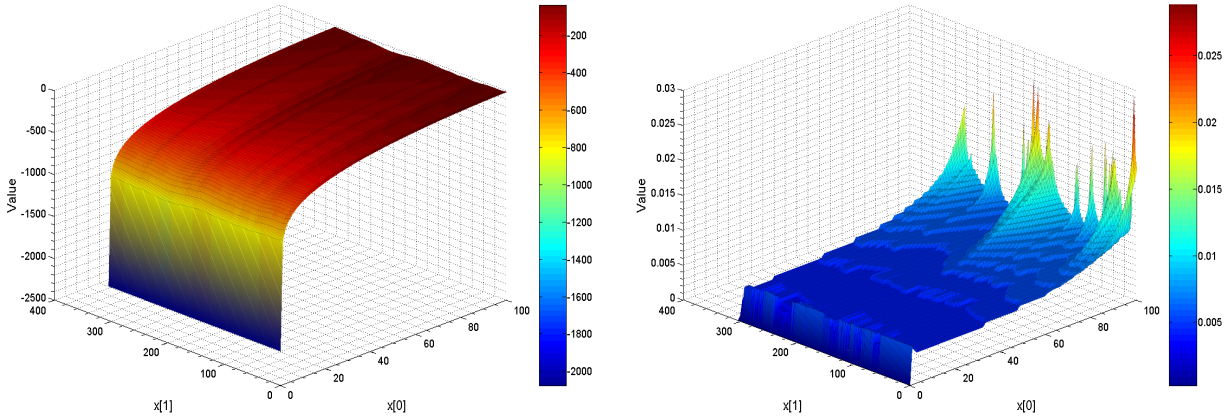

Abbildung 6.2: Optimale Wertefunktion und Kontrolle $u[0]$ für Beispiel SP_det_b

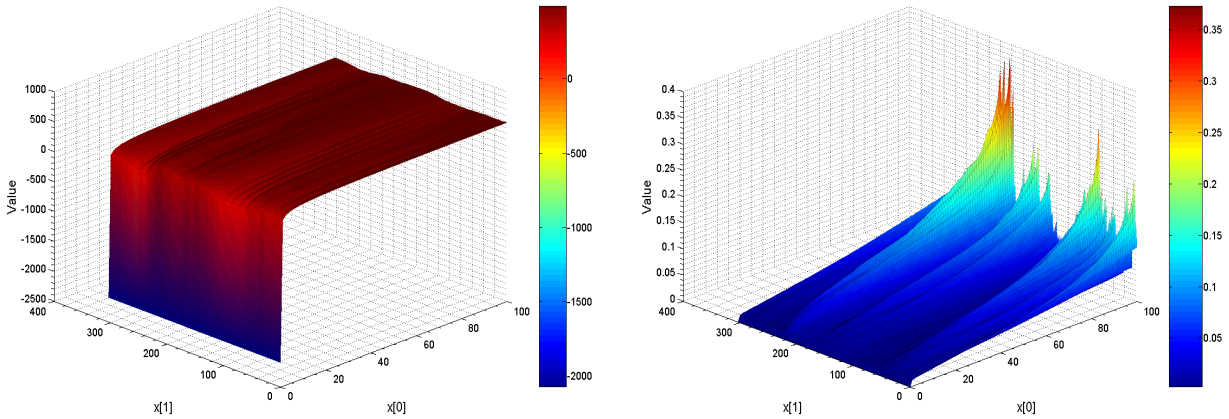

Abbildung 6.3: Optimale Wertefunktion und Kontrolle $u[0]$ für Beispiel SP_det_c

Dazu vergleiche man die folgende Tabelle:

Beispiel	$max_{opt.Wertefkt.}$	$\mu_{opt.Wertefkt.}$	$max_{opt.Kontrolleu[0]}$	$\mu_{opt.Kontrolleu[0]}$
SP_det_a	301.449024	189.042135	0.113887	0.018424
SP_det_b	-23.429586	-215.442357	0.028984	0.005567
SP_det_c	505.780201	416.458806	0.375000	0.042552

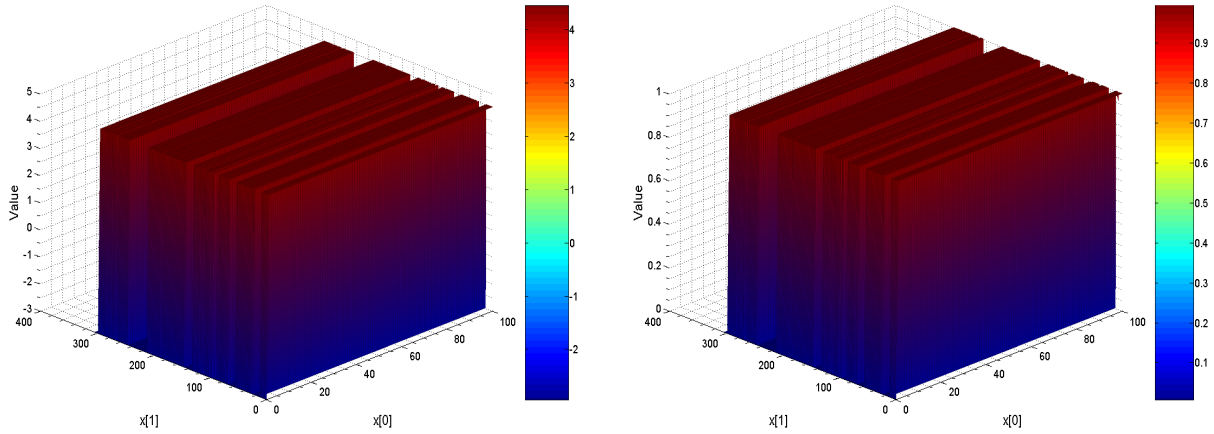

Abbildung 6.4: Optimale Kontrolle $u[1]$ für Beispiel SP_det_a und SP_det_b

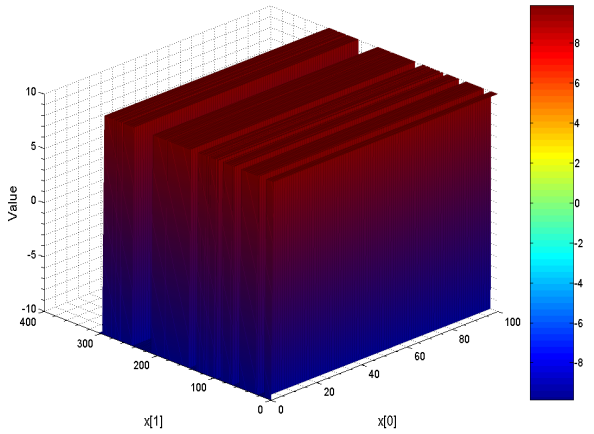

Abbildung 6.5: Optimale Kontrolle $u[1]$ für Beispiel SP_det_c

Dieses Ergebnis folgt daraus, dass die Zinsen im Modell von vorn herein bekannt sind und keine stochastische Störung einwirkt. Daher ist es rational für den Investor, durch die Leerverkäufe der renditeärmeren Anlage sich so viel Kapital zu beschaffen, wie es durch die Kontrolle $u[1]$ zulässig ist, da er durch das Wissen über die Renditen

kein Risiko eingeht.[5]
Die optimalen Trajektorien für den Startwert $x0 = [1, 0]$ in Abbildung 6.6 und 6.7

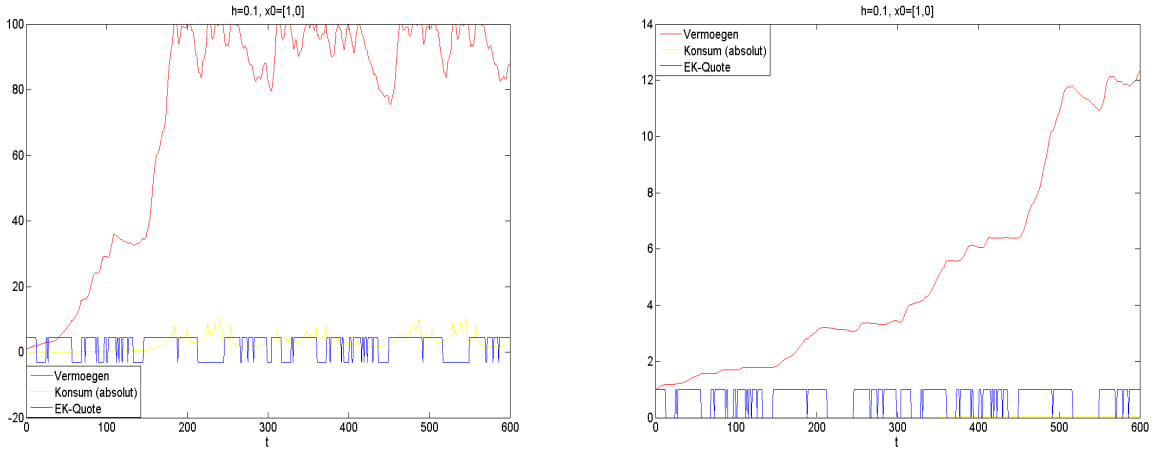

Abbildung 6.6: Optimale Trajektorien für Beispiel SP_det_a und SP_det_b

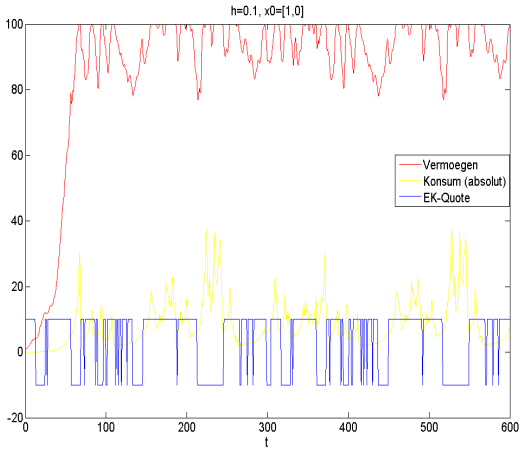

Abbildung 6.7: Optimale Trajektorien für Beispiel SP_det_c

bestätigen die Ergebnisse: Bei Beispiel SP_det_c sieht man ein schnelleres Ansteigen des Vermögens (rot) und einen höheren Konsum (gelb) als bei SP_det_b. Dies ist wiederum durch den größeren zulässigen Kontrollwertebereich in der Komponente $u[1]$, zu erkennen an der *EK_Quote* (blau), erklärbar.

[5]Wenn dagegen bei der Rendite der Anlagen eine Unsicherheit besteht, geht man durch Leerverkäufe ein unbeschränktes Verlustrisiko bei beschränkter Gewinnchance ein.

6.3.2 Schweizer Indizes

Folgende Fälle werden betrachtet:

Beispiel	$U = [u_{min}[0], u_{max}[0]] \times [u_{min}[1], u_{max}[1]]$	Abbildung
SI_det_a	$[0, 0.7] \times [-3, 4.5]$	6.8 & 6.11
SI_det_b	$[0, 0.7] \times [0, 1]$	6.9 & 6.11
SI_det_c	$[0, 1] \times [-10, 10]$	6.10 & 6.12

Für die Zinsen, welche in Abbildung 4.1 dargestellt sind, gelten folgende Daten:

Zeit	$\mathbb{E}(Aktie)$	$\sigma(Aktie)$	$\mathbb{E}(Obligation)$	$\sigma(Obligation)$	ρ
74	0.105727	0.203656	0.046015	0.037076	0.367007

Im Unterschied zu Abschnitt 6.3.1 sind die Zinsen hier jährlich und es gibt 74 Zinsdaten.

Auswertung:

Da sich die optimale Kontrolle $u[1]$ genauso verhält wie in Abschnitt 6.3.1, wird auf deren Abbildung hier und in Abschnitt 6.3.3 verzichtet. Begründed werden kann der Verlauf der Kontrolle genau so wie in Abschnitt 6.3.1.

Der Einfluss des Kontrollwertebereichs U auf die optimale Wertefunktion und Konsumquote für die drei Beispiele hier ist äquivalent zu den Unterschieden der drei Beispiele aus Abschnitt 6.3.1, d.h. die Veränderung von U wirkt sich in der selben Art und Weise aus.

Wenn man die Ergebnisse der Schweizer Indizes mit denen von Standard&Poor's vergleicht, fällt auf, dass sie sich nur in der absoluten Höhe des Konsumniveaus und der optimalen Wertefunktion unterscheiden. Dazu vergleiche die Ergebnisse der folgenden Tabelle mit der auf Seite 82:

Beispiel	$max_{opt.Wertefkt.}$	$\mu_{opt.Wertefkt.}$	$max_{opt.Kontrolleu[0]}$	$\mu_{opt.Kontrolleu[0]}$
SI_det_a	85.146071	74.018214	0.700000	0.311907
SI_det_b	56.410470	36.279519	0.613594	0.088052
SI_det_c	100.175886	92.029216	1.000000	0.627152

Das Konsumniveau liegt in diesem Abschnitt für alle Beispiele höher als bei den Standard&Poor's Indizes. Dies ist darauf zurückzuführen, dass hier Jahresrenditen zugrunde gelegt sind, die im Mittelwert (auch auf den Monat gerechnet) höher sind als die monatlichen der Standard&Poor's Indizes.

Die optimale Wertefunktion in Beispiel SP_det_a bzw. SP_det_c liegt auf einem höheren Niveau als die in SI_det_a bzw. SI_det_c, aber in SP_det_b auf niedrigeren als in SI_det_b ist, obwohl U gleich gewählt wurde.

In Abschnitt 6.3.1 wurde der Diskontfaktor aufgrund der monatlichen Zinsen kleiner gewählt. Dass ein kleineres δ (unter sonst gleichen Bedingungen: Parameter, Zinsen,...) zu einem höheren Niveau der optimalen Wertefunktion führt, wurde in [14] Öhrlein, 2006, S.124-128 ausführlich untersucht. Dass der Mittelwert der optimalen Wertefunktion bei SP_det_b niedriger als bei SI_det_b ist, liegt vorallem daran, dass die Zinsen so niedrig sind, dass nur so wenig Konsum möglich ist, dass die Power-Nutzenfunktion negativ wird (vgl. Gleichung (4.3)), wodurch auch die opti-

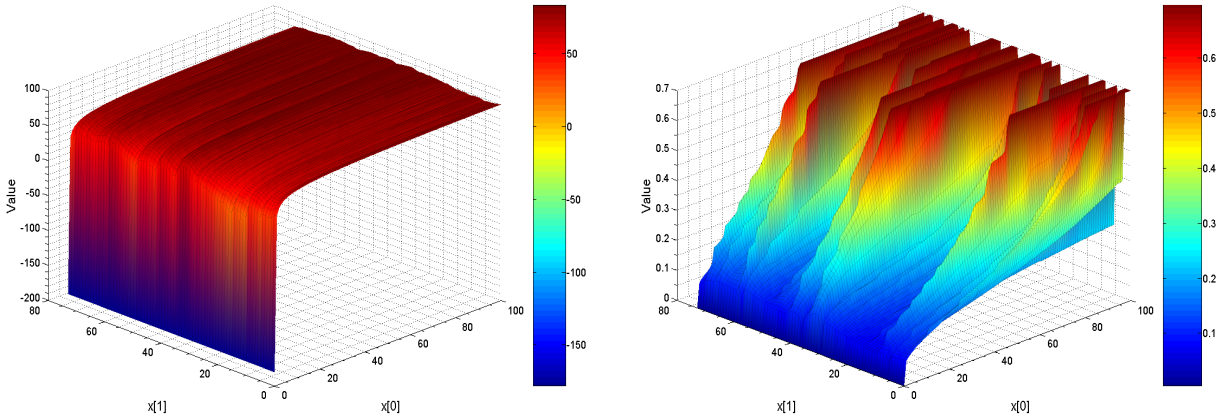

Abbildung 6.8: Optimale Wertefunktion und Kontrolle $u[0]$ für Beispiel SI_det_a

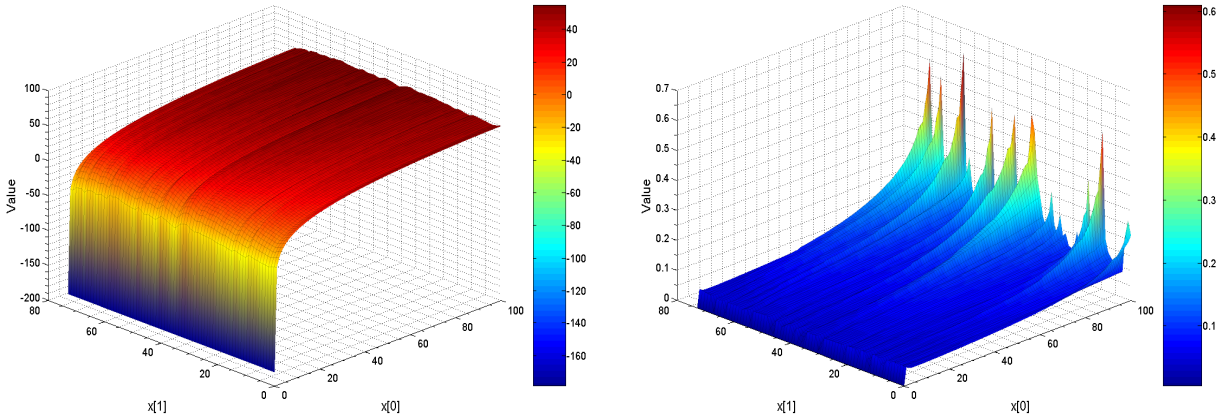

Abbildung 6.9: Optimale Wertefunktion und Kontrolle $u[0]$ für Beispiel SI_det_b

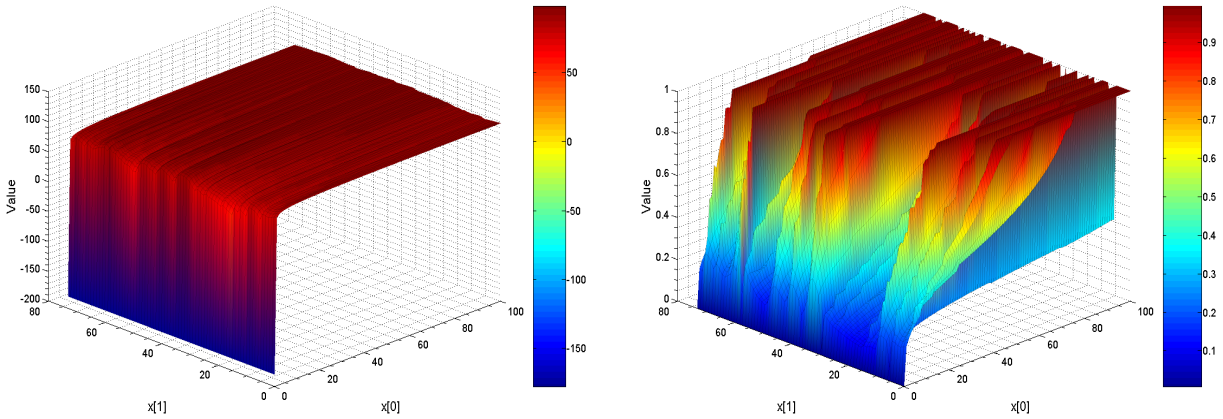

Abbildung 6.10: Optimale Wertefunktion und Kontrolle $u[0]$ für Beispiel SI_det_c

male Wertefunktion negativ wird.
Bei den optimalen Trajektorien in Abbildung 6.11 und 6.12 sieht man, dass das

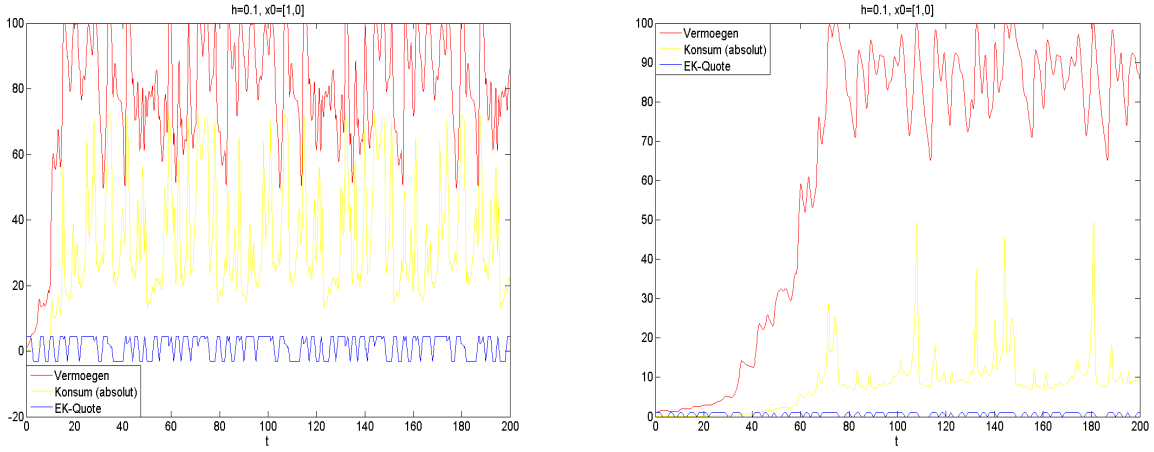

Abbildung 6.11: Optimale Trajektorien für Beispiel SI_det_a und SI_det_b

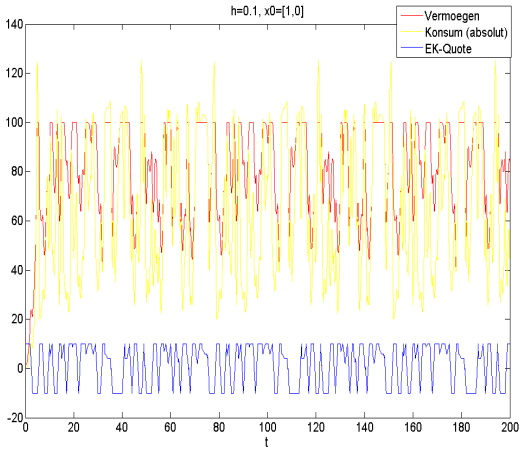

Abbildung 6.12: Optimale Trajektorien für Beispiel SI_det_c

Vermögen (rot) und der absolute Konsum (gelb) stärker als in Abschnitt 6.3.1 schwanken, aber auch, dass der Konsum ein höheres Niveau und das Vermögen den Maximalwert 100 schneller erreicht. Dies lässt sich wiederum auch damit erklären, dass hier die Zinsen stärker schwanken aber auch im Mittel höher sind.

6.3.3 Mittels Zufallszahlen erzeugte Zinsstrukturen

Für den Kontrollwertebereich gelte $U = [0,1] \times [0,1]$ und die optimalen Trajektorien wurden bis $x\_max = 600$ berechnet. Die Zinsen wurden mit folgenden Eingabedaten mit dem Programm *Zinserzeugung.c* erzeugt:[6]

Zeit	EK_EW	EK_sigma	FK_EW	FK_sigma
100	0.1057	0.205	0.046	0.0373

Daher wird hier $hi[1] = 99$ gewählt. Dabei entsprechen EK_EW und EK_sigma dem Erwartungswert bzw. der Standardabweichung des Schweizer Aktienindexes sowie FK_EW und FK_sigma dem des Schweizer Obligationenindexes aus Abbildung 4.1. Als Korrelationen wurden $-1, 0, 1$ gewählt.

Beispiel	Korrelation ρ	Abbildung
ZV_det_a	0	6.13 & 6.16
ZV_det_b	1	6.14 & 6.17
ZV_det_c	-1	6.15 & 6.18

Die zufällig erzeugten Zinsen haben folgende Daten:[7]

Beispiel	EK_EW	EK_sigma	FK_EW	FK_sigma	ρ
ZV_det_a	0.103021	0.237922	0.042928	0.034265	-0.034650
ZV_det_b	0.109087	0.223490	0.046616	0.040664	1.000000
ZV_det_c	0.105215	0.198456	0.046088	0.036109	-1.000000

Ziel dieses Abschnittes ist es, die Auswirkungen der Korrelation auf das Ergebnis der Optimierung zu untersuchen. Daher haben die drei Beispiele ZV_det_a, ZV_det_b und ZV_det_c bis auf eine geringe Abweichung die selben Erwartungswerte und Standardabweichungen wie der Schweizer Aktien- bzw. Obligationenindex, unterscheiden sich aber in der Korrelation ρ.

Auswertung:

Die optimalen Trajektorien in Abbildung 6.16 bis 6.18 zeigen, dass bei einer Korrelation $\rho = -1$ (die Zinsen der zwei Anlagen verlaufen exakt gegensätzlich) das Vermögen nur zwischen ca. 70 und 100 ab $t = 100$ schwankt, wogegen es bei einer von $\rho = 1$ (die Zinsen der zwei Anlagen verlaufen gleichläufig) zwischen ca. 50 und 100 ab $t = 100$ schwankt. Der absolute Konsum erreicht bei $\rho = -1$ einen maximalen Wert von ca. 60, wogegen er bei $\rho = 1$ nur einen Wert von ca. 42 erreicht.
Bei der optimalen Wertefunktion ist der mittlere Wert bei $\rho = -1$ gleich 41.170363 und bei $\rho = 1$ gleich 37.096593 (berechnet durch Anwendung von *maxmin.m* auf das jeweilige Gitter), d.h. ein Investor erreicht einen höheren Nutzen, wenn er zwischen zwei negativ korrelierten Anlagen wählen kann als bei zwei positiv korrelierten.

[6] vgl. Abschnitt 5.2.2

[7] Abweichungen zu den Eingabedaten folgen daraus, dass nur 100 Daten erzeugt wurden und dies (pseudo)zufällig geschah.

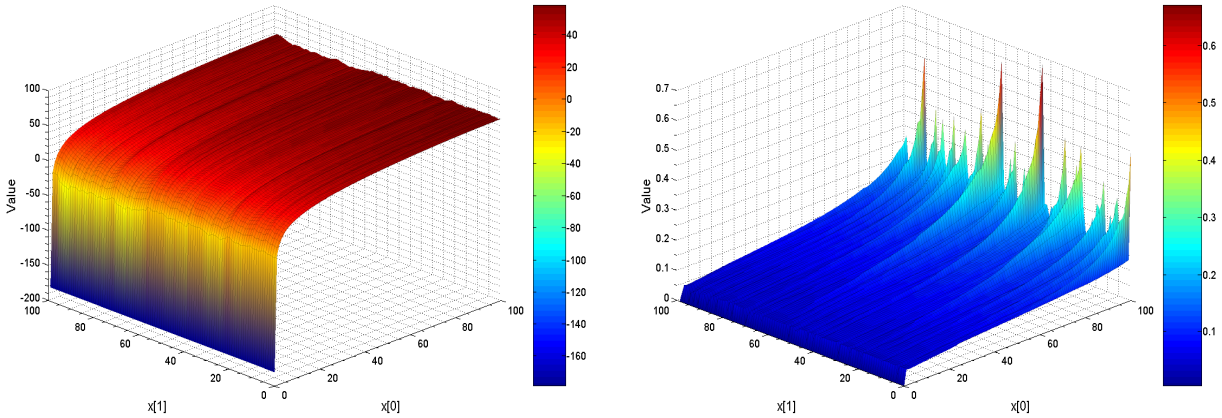

Abbildung 6.13: Optimale Wertefunktion und Kontrolle $u[0]$ für Beispiel ZV_det_a

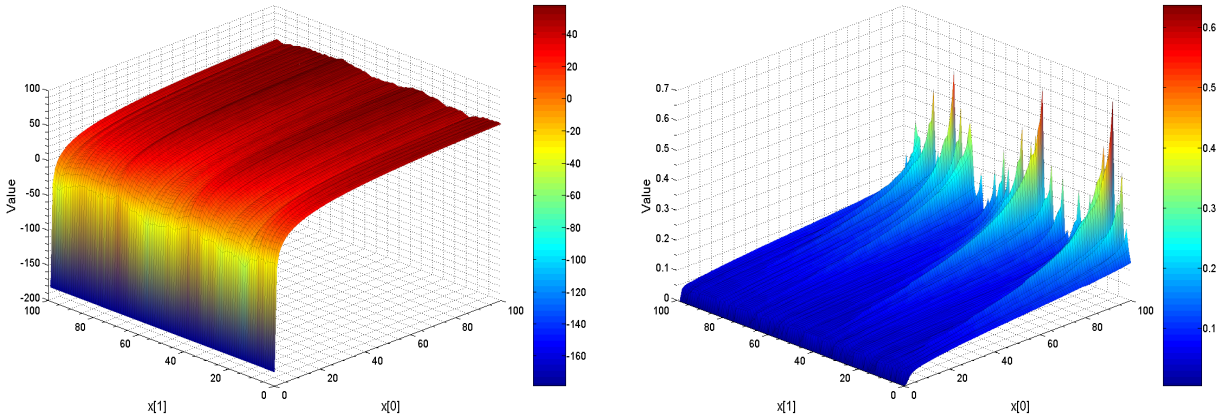

Abbildung 6.14: Optimale Wertefunktion und Kontrolle $u[0]$ für Beispiel ZV_det_b

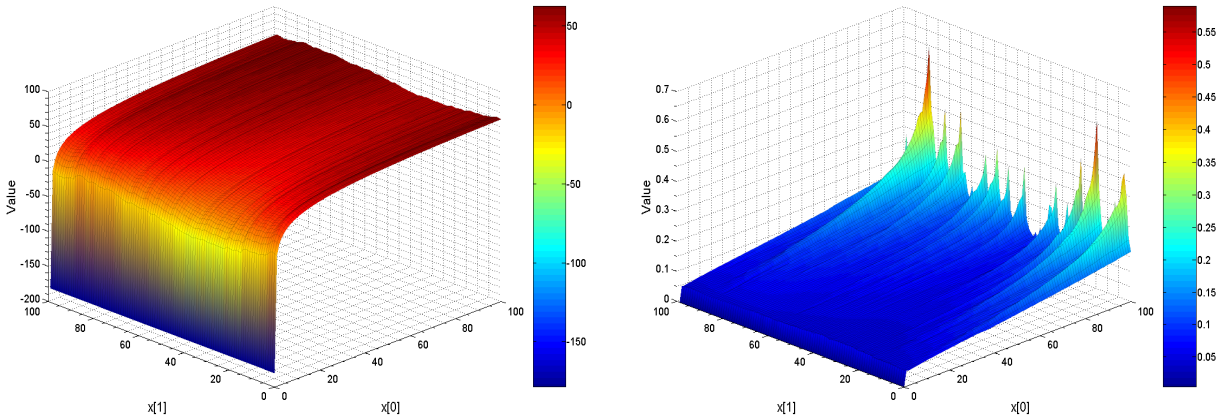

Abbildung 6.15: Optimale Wertefunktion und Kontrolle $u[0]$ für Beispiel ZV_det_c

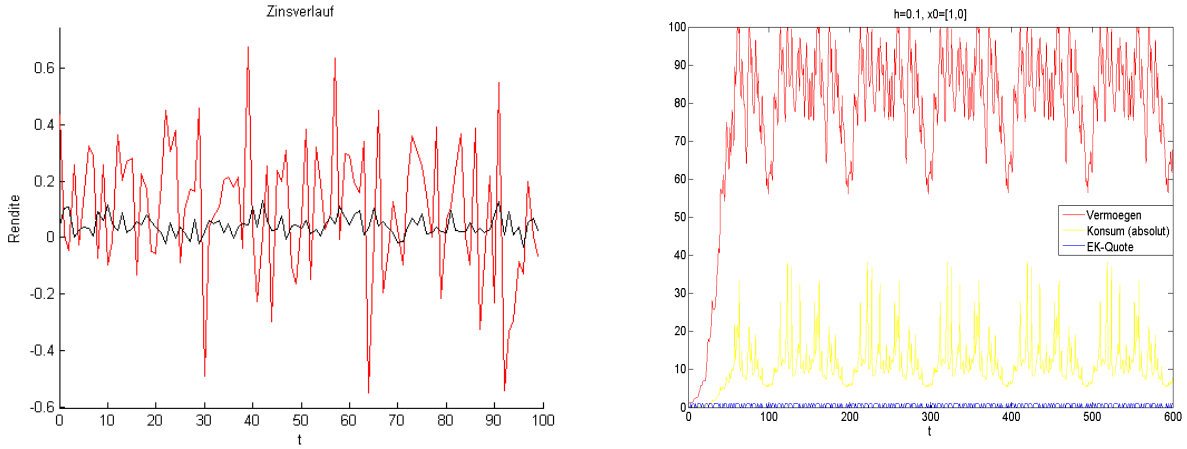

Abbildung 6.16: Zinsen & Optimale Trajektorien für Beispiel ZV_det_a

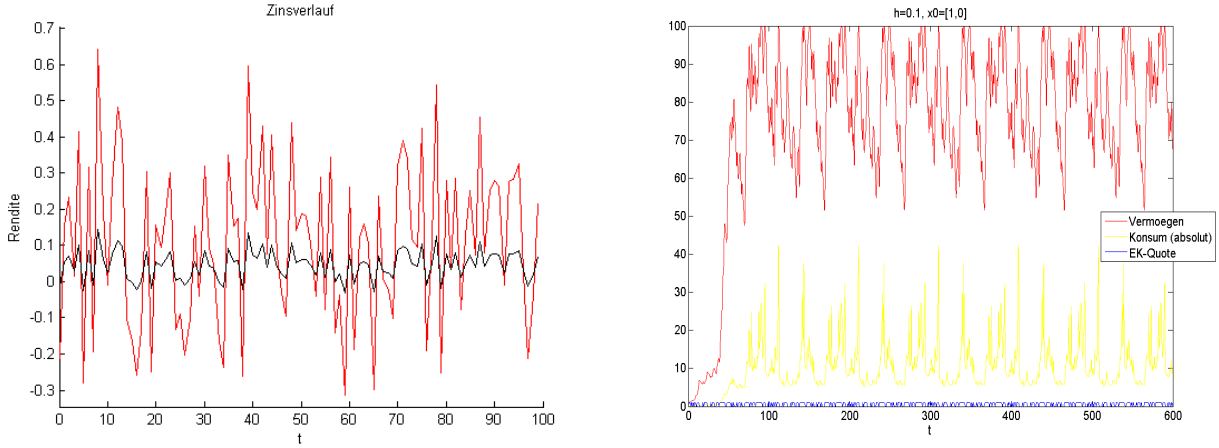

Abbildung 6.17: Zinsen & Optimale Trajektorien für Beispiel ZV_det_b

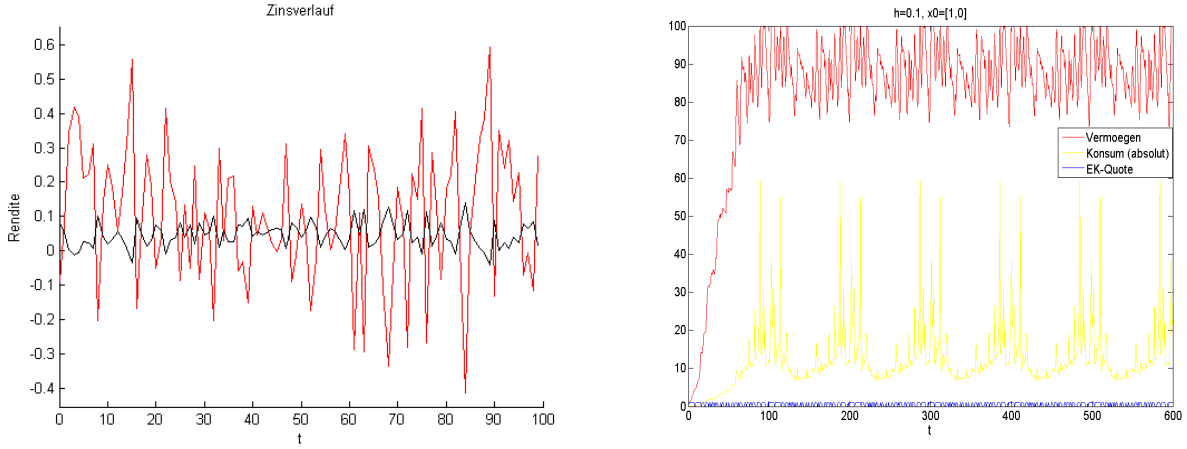

Abbildung 6.18: Zinsen & Optimale Trajektorien für Beispiel ZV_det_c

6.4 Das stochastische Modell

Ziel dieses Abschnitts ist es, die Auswirkungen des stochastischen Einflusses auf die Beispiele der Abschnitte 6.3.1 und 6.3.2 zu untersuchen. Als erstes werden Beispiele untersucht, wo nur einer der Gewichtungsfaktoren (σ_{EK} bzw. σ_{FK}) > 0 und der andere $= 0$ ist. Danach werden auch beide > 0 (aber unteschiedlich groß) gewählt. Im Abschnitt 6.4.2 wird zusätzlich untersucht, wie sich die Lösung verändert, wenn $\sigma_{FK} = 0$ festgehalten und σ_{EK} erhöht wird.
Die Mittelwerte, Standardabweichungen sowie Maximalwerte auf den Gittern im Folgenden wurden mit *maxmin.m* berechnet.

6.4.1 Indizes von Standard&Poor's

Folgende Fälle werden betrachtet:

Beispiel	σ_{EK}	σ_{FK}	$U = [u_{min}[0], u_{max}[0]] \times [u_{min}[1], u_{max}[1]]$	Abbildung
SP_stoch_a	0.2	0	$[0, 0.7] \times [-3, 4.5]$	6.19 & 6.22 & 6.24
SP_stoch_b	0	0.2	$[0, 0.7] \times [-3, 4.5]$	6.20 & 6.22 & 6.24
SP_stoch_c	0.5	0.2	$[0, 0.7] \times [-3, 4.5]$	6.21 & 6.23 & 6.25

Auswertung:
Die Anwendung von *maxmin.m* auf die Gitter der optimalen Wertefunktion sowie der Kontrollen $u[0]$ und $u[1]$ der drei Beispiele liefert folgende Maximal- und Mittelwerte:

Beispiel	opt. Wertefkt.		opt. Kontrolle $u[0]$		opt. Kontrolle $u[1]$	
	max	μ	max	μ	max	μ
SP_stoch_a	-143.005404	-347.158949	0.011621	0.004809	1.523438	-0.016593
SP_stoch_b	-96.909547	-310.689389	0.029668	0.005355	2.233887	0.960994
SP_stoch_c	-1218.509981	-1388.194223	0.071641	0.009866	0.285645	0.122177

Wenn man die optimale Kontrolle $u[1]$ in Beispiel SP_stoch_a in Abbildung 6.19 betrachet, dann erkennt man, dass nicht mehr wie im deterministischen Fall (vgl. Abbildung 6.4) die Kontrolle den Minimal- bzw. Maximalwert annimmt (abhängig davon, welche der beiden Anlagen die höhere Rendite aufweist). Optisch zeichnet sich eine Ebene ab, die nahe bei 0 liegt. Dies bestätigt auch der Mittelwert von $u[1]$ auf dem Gitter, welcher -0.016593 beträgt. In der $x[1]$-Richtung ist dabei ein wellenförmiger Verlauf zu beobachten, wobei sich die meisten Werte im Intervall $[-1.5, 1.5]$ bewegen. Wenn $u[1] = 0$ ist, dann wird das gesamte Vermögen in die risikoarme Anlage investiert[8], welche in diesem Beispiel die mit $\sigma_{FK} = 0$ ist. Daraus kann man folgern, dass tendenziell mehr in die Anlage mit dem kleineren σ investiert wird (die meisten Gitterpunkte sind nahe bei 0), außer wenn die erwartete Rendite der Anlage mit höherem σ groß genug ist (wellenförmiger Verlauf in $x[1]$-Richtung). Wenn man im Verlauf des Standard&Poor's 500 Index in Abbildung 4.1 (rote Kurve) die Zeitpunkte t mit hohen Renditen mit dem Verlauf von $u[1]$ in $x[1]$-Richtung vergleicht, dann sieht man leicht, dass die Zeitpunkte t, wo die Rendite sehr hoch

[8] vgl. Modell (4.6)

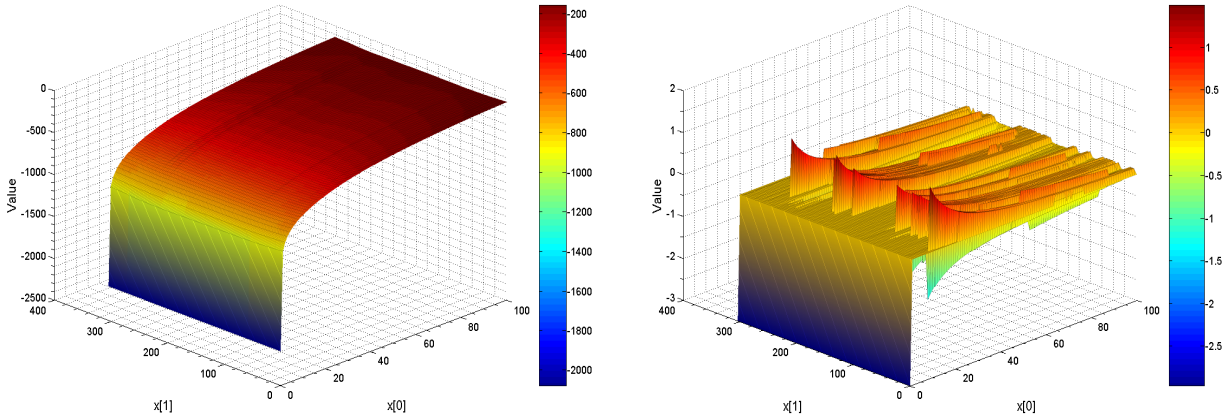

Abbildung 6.19: Optimale Wertefunktion und Kontrolle u[1] für Beispiel SP_stoch_a

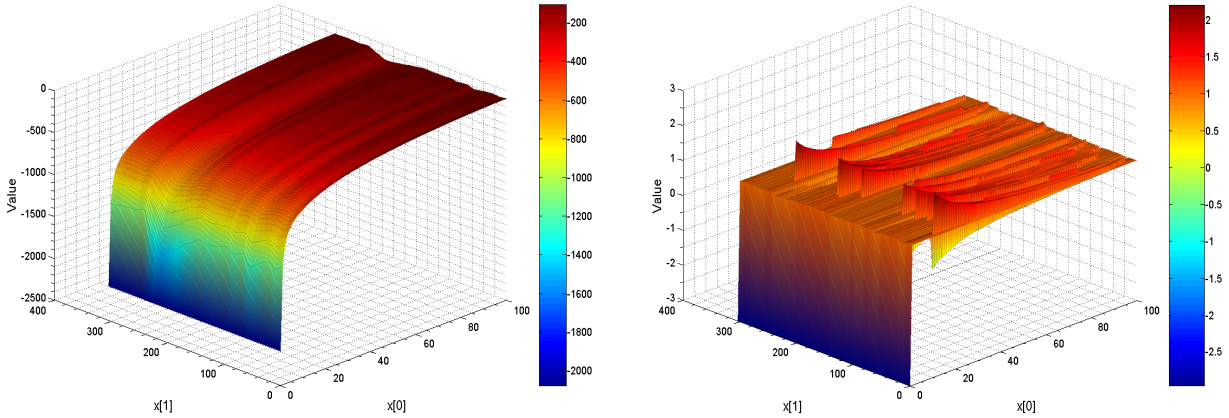

Abbildung 6.20: Optimale Wertefunktion und Kontrolle u[1] für Beispiel SP_stoch_b

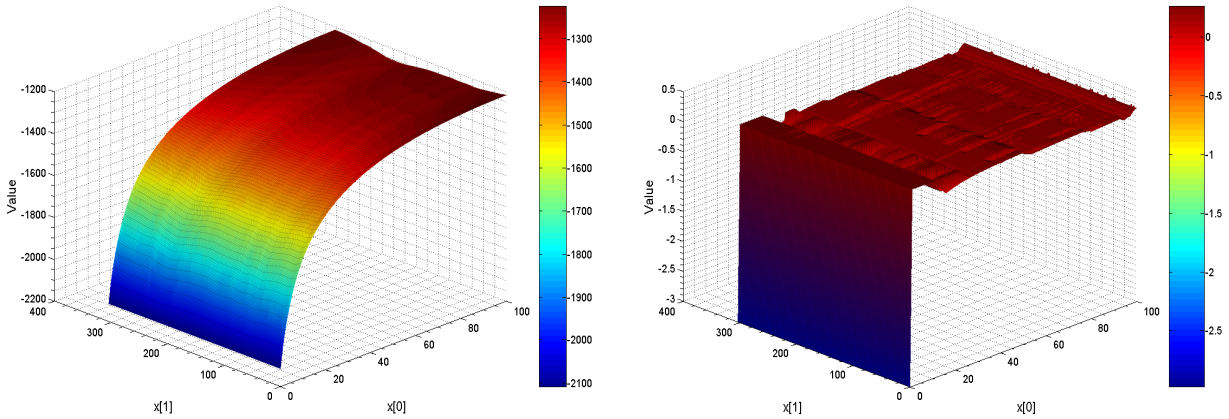

Abbildung 6.21: Optimale Wertefunktion und Kontrolle u[1] für Beispiel SP_stoch_c

ist, mit den $x[1]$-Werten übereinstimmen, wo $u[1] > 0$ ist. Zum Beispiel liegt das größte $u[1]$ am Gitterpunkt $(x[0], x[1]) = (10, 5)$ und hat den Wert 1.523438, die maximale Rendite des Standard&Poor's 500 Index liegt bei $x[1] = t = 5$ mit dem Wert 0.0290667
Das selbe Ergebnis erhält man bei Beispiel SP_stoch_b, nur dass der Mittelwert bei

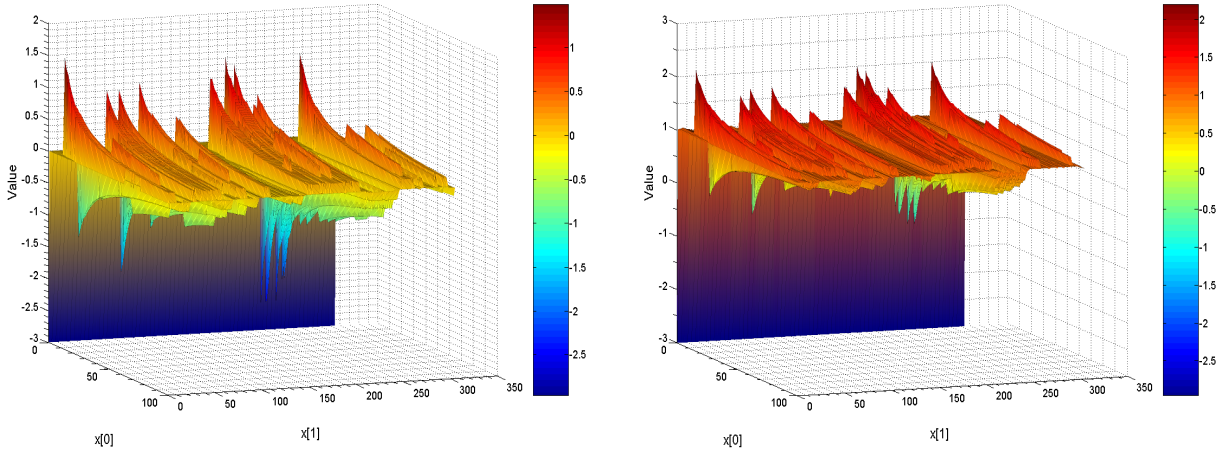

Abbildung 6.22: Optimale Kontrolle u[1] für Beispiel SP_stoch_a und SP_stoch_b

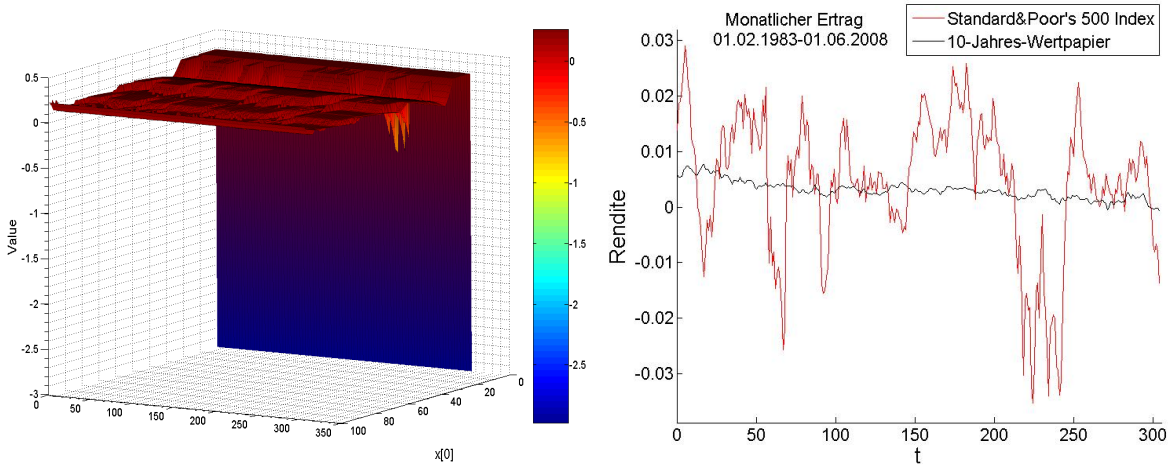

Abbildung 6.23: Optimale Kontrolle u[1] für Beispiel SP_stoch_c / Zinsverlauf aus Abb. 4.1

0.960994 liegt und die Abweichungen nach oben sowie nach unten geringer als bei SP_stoch_a ausfallen, aber an den gleichen Stellen $x[1]$ sind.
Die Abbildungen 6.22 und 6.23 zeigen nochmals die optimale Kontrolle $u[1]$ aus einer anderen Blickrichtung sowie die Zinskurven, die hier verwendet wurden. Wenn man die rote Kurve dabei mit dem Verlauf von $u[1]$ in $x[1]$-Richtung vergleicht, dann sieht man die oben beschriebene Gleichläufigkeit von $u[1]$ und Standard&Poor's 500

Index.
Bei Beispiel SP_stoch_c ist dies nicht mehr so eindeutig feststellbar, da beide Anlagen einem stochastischen Einfluss unterliegen ($\sigma > 0$), wobei der auf den Standard&Poor's 500 Index größer ist. Der Mittelwert beträgt hier 0.122177 und der maximale 0.285645. Im Durchschnitt wird also mehr Kapital in die Anlage mit geringerem stochastischen Einfluss ($\sigma_{FK} = 0.2$) investiert. Außerdem ist die Standardabweichung in SP_stoch_c mit 0.319147 kleiner als bei SP_stoch_a bzw. SP_stoch_b (0.400411 bzw. 0.461730). Einzig bei $x[1] \in [200, 250]$ und $x[0] \approx 10$ ist eine starke Abweichung vom Mittelwert nach unten erkennbar ($u[1]$ ca. -0.5)[9], da auch dort der Standard&Poor's 500 Index sehr klein (bei -0.03) ist.
Weiter fällt auf, dass in allen drei Beispielen die Schwankungen in $x[1]$-Richtung bei kleinem Vermögen $x[0]$ sehr groß sind und mit zunehmendem $x[0]$ abnehmen.
Die optimale Wertefunktion ist im Mittel mit -310.689389 in SP_stoch_b größer als in SP_stoch_a mit -347.158949, obwohl der stochastische Einfluss σ_{EK} bei SP_stoch_a gleich σ_{FK} bei SP_stoch_b ist. Da, nach obigen Betrachtungen, das meiste Kapital in die Anlage mit dem stochastischen Gewichtungsfaktor 0 investiert wird, erreicht man dadurch bei SP_stoch_b höhere Renditen (Investition in den renditestärkeren Standard&Poor's 500 Index), wodurch mehr Konsum möglich ist. Am niedrigsten ist die optimale Wertefunktion mit -1388.194223 in SP_stoch_c, wo beide Anlagen einem stochastischen Einfluss unterliegen.

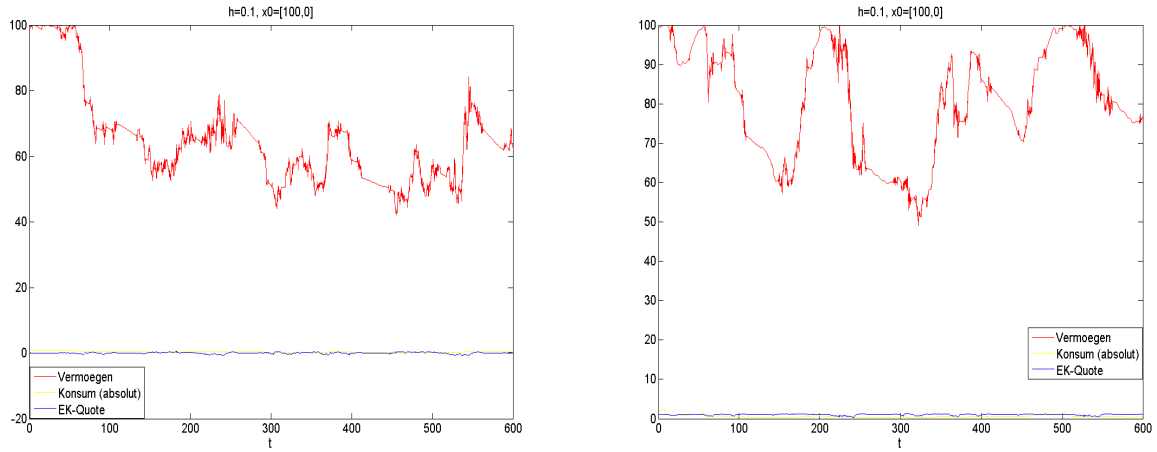

Abbildung 6.24: Optimale Trajektorien für Beispiel SP_stoch_a und SP_stoch_b

In Abbildung 6.24 und 6.25 sind die optimalen Trajektorien dargestellt. Bei den Beispielen SP_stoch_a und SP_stoch_b pendelt sich das Vermögen zwischen 40 und 100 ein, wogegen es bei SP_stoch_c gegen 0 konvergiert. Dies erfolgt bei sämtlichen vier zufällig erzeugten optimalen Trajektorien in Abbildung 6.25, aber der Zeitpunkt, ab dem das Vermögen den Wert 0 annimmt, variiert dabei ($t = 269.5$, $t = 151.8$,

[9] vgl. orange gefärbten Teil der optimalen Kontrolle $u[1]$ der Abbildung 6.23

$t = 998.3$ und $t = 941.9$). Ein Grund für die Konvergenz gegen 0 kann sein, dass der Investor in den ersten beiden Beispielen immer auf eine Anlage ohne stochastischen Einfluss ausweichen kann, was beim dritten nicht möglich ist. Der Grund für die unterschiedlichen Zeitpunkte, wo das Vermögen den Wert 0 annimmt, liegt darin, dass der stochastische Einfluss dem (Pseudo-) Zufall unterliegt und bei jedem Programmstart ein neuer *seed* mit der C-Funktion *srand(time(0))*[10] gesetzt wird, wodurch für jeden der vier optimalen Trajektorien in Abbildung 6.25 andere (Pseudo-) Zufallszahlen erzeugt werden.
Ob das Vermögen immer gegen 0 konvergiert (bei $\sigma_{EK} = 0.5$ und $\sigma_{FK} = 0.2$), soll im folgenden Abschnitt untersucht werden.

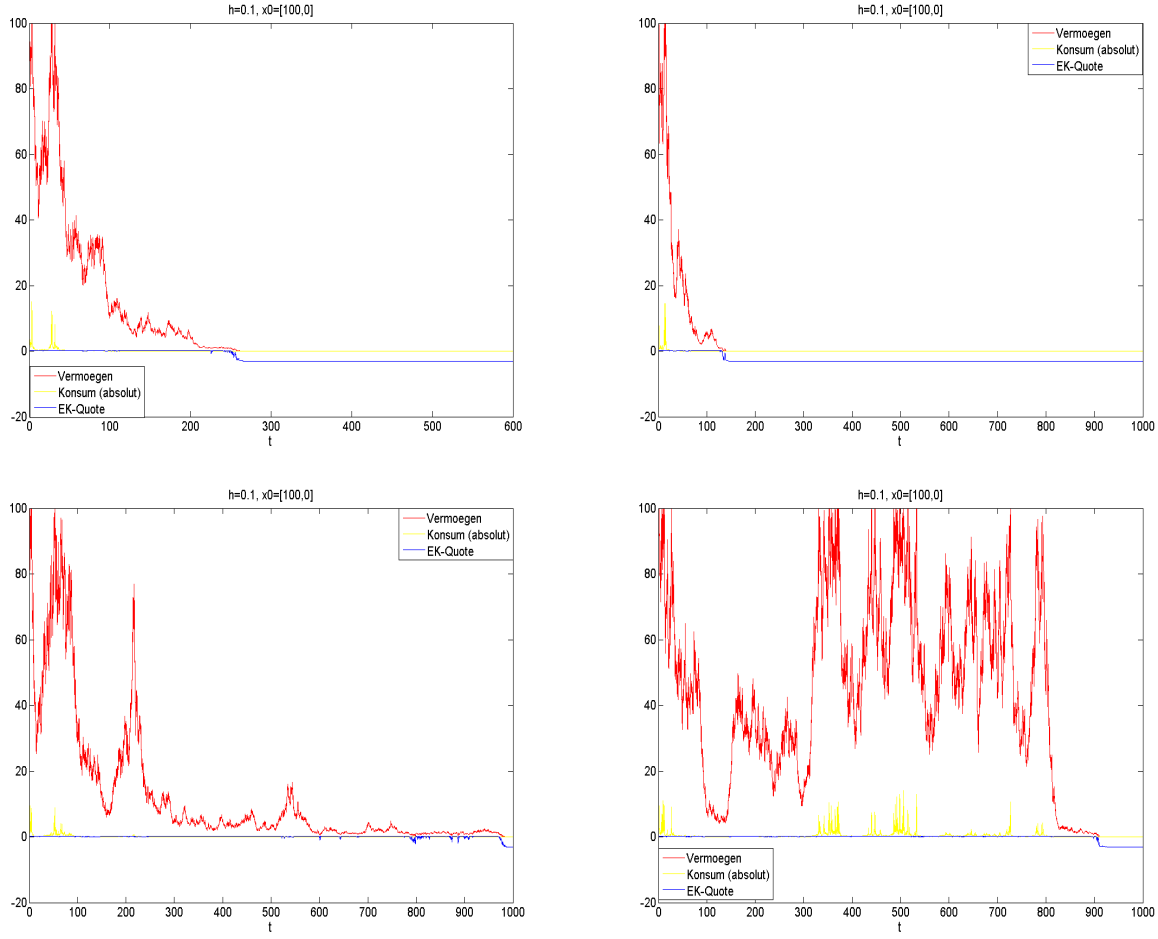

Abbildung 6.25: 4 zufällig erzeugte optimale Trajektorien ($t = 600$ bzw. 1000) für Beispiel SP_stoch_c

[10]dazu vgl. [5] Grüne, 2008, S.91/92; *srand(time(0))* erzeugt den neuen *seed* mit Hilfe der aktuellen Systemzeit

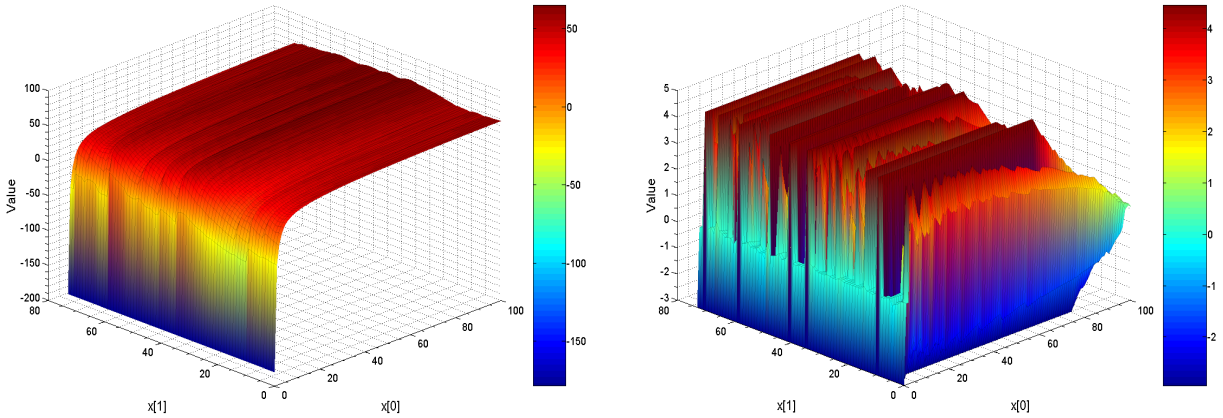

Abbildung 6.26: Optimale Wertefunktion und Kontrolle $u[0]$ für Beispiel SI_stoch_a

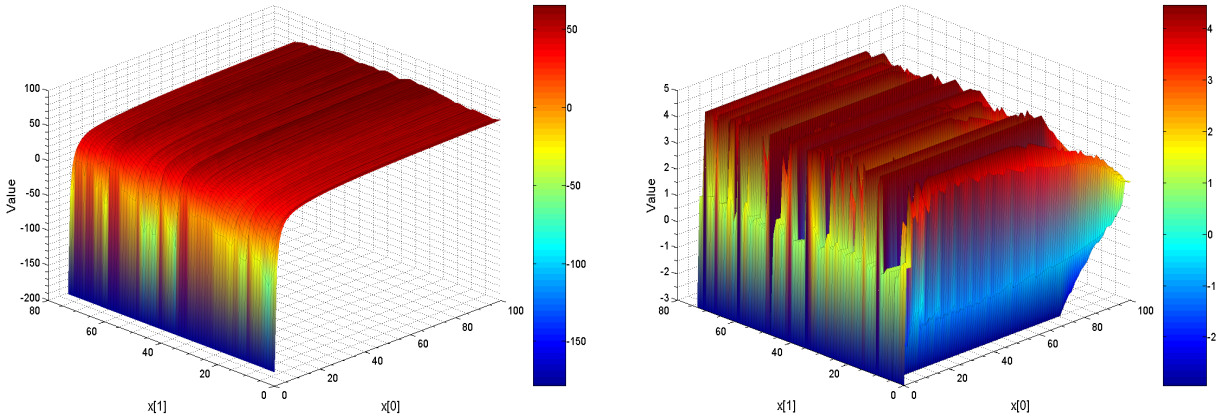

Abbildung 6.27: Optimale Wertefunktion und Kontrolle $u[0]$ für Beispiel SI_stoch_b

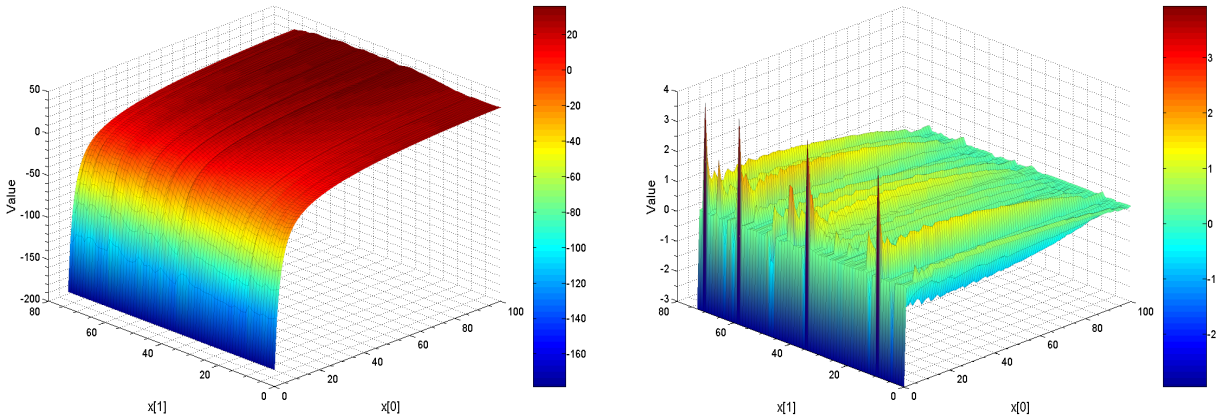

Abbildung 6.28: Optimale Wertefunktion und Kontrolle $u[0]$ für Beispiel SI_stoch_c

6.4.2 Schweizer Indizes

Folgende Fälle werden betrachtet:

Beispiel	σ_{EK}	σ_{FK}	$U = [u_{min}[0], u_{max}[0]] \times [u_{min}[1], u_{max}[1]]$	Abbildung
SI_stoch_a	0.2	0	$[0, 0.7] \times [-3, 4.5]$	6.26 & 6.29 & 6.31
SI_stoch_b	0	0.2	$[0, 0.7] \times [-3, 4.5]$	6.27 & 6.29 & 6.31
SI_stoch_c	0.5	0.2	$[0, 0.7] \times [-3, 4.5]$	6.28 & 6.30 & 6.32

σ_{EK} ist hier der stochastische Einfluss auf den Schweizer Aktienindex und σ_{FK} der auf den Schweizer Obligationenindex.

Auswertung:

Die Anwendung von *maxmin.m* auf die Gitter der optimalen Wertefunktion sowie der Kontrollen $u[0]$ und $u[1]$ der drei Beispiele liefert folgende Maximal- und Mittelwerte:

Beispiel	opt. Wertefkt.		opt. Kontrolle $u[0]$		opt. Kontrolle $u[1]$	
	max	μ	max	μ	max	μ
SI_stoch_a	66.444770	50.315491	0.700000	0.220195	4.500000	0.677267
SI_stoch_b	67.161241	52.552756	0.700000	0.232735	4.500000	1.456400
SI_stoch_c	37.336720	15.794188	0.464092	0.094680	3.973389	0.217805

Genau wie in Abschnitt 6.4.1 verhalten sich hier die Ergebnisse des Schweizer Aktien- und Obligationenindexes im Falle der optimalen Wertefunktion und Kontrolle $u[1]$ (Abbildungen 6.26, 6.27 und 6.28).

Bei SI_stoch_b hat mit einem Mittelwert von 52.552756 die optimale Wertefunktion ein höheres Niveau als bei SI_stoch_a mit 50.315491. Begründbar ist dies wie im vergangenen Abschnitt damit, dass der Investor eher in die Anlage mit dem stochastischen Einfluss $\sigma = 0$ investiert und in SI_stoch_b dies die Anlage mit der höheren erwarteten Rendite ist (Schweizer Aktienindex). Am niedrigsten ist der Mittelwert der optimalen Wertefunktion bei SI_stoch_c mit 15.794188, da dort beide Anlage von dem stochastischen Einfluss abhängen.

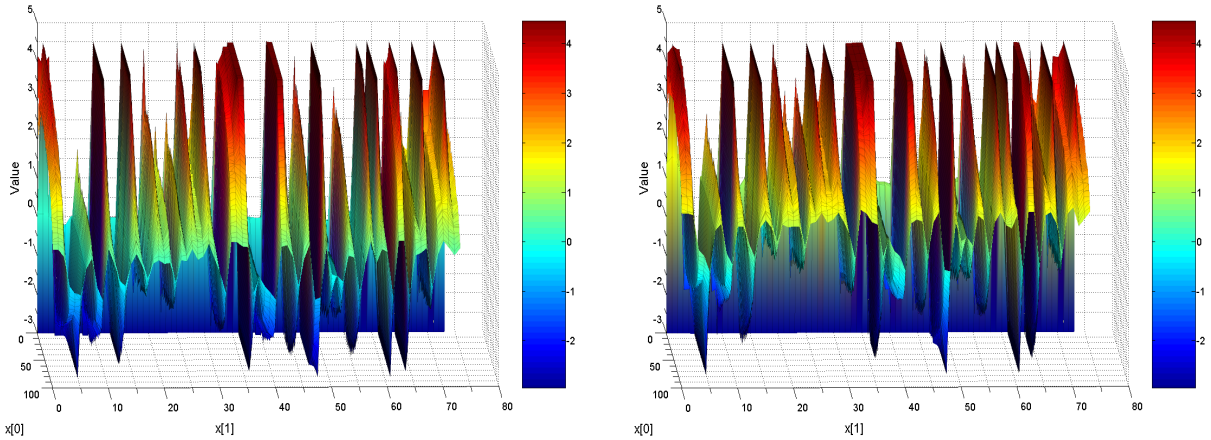

Abbildung 6.29: Optimale Kontrolle $u[1]$ für Beispiel SI_stoch_a und SI_stoch_b

Im wellenförmigen Verlauf der optimalen Kontrolle $u[1]$ in $x[1]$-Richtung erkennt man stark den Verlauf der Renditen des Schweizer Aktienindexes (vgl. Abbildung 6.29 und 6.30) wieder. Die mittleren Werte von $u[1]$ sind bei SI_stoch_a 0.677267, bei SI_stoch_b 1.456400 und bei SI_stoch_c 0.217805. Ein Wert $u[1] = 1$ bedeutet dabei, dass 100% des Vermögens in den Schweizer Aktienindex investiert wird. D.h. bei $\sigma_{EK} = 0$ wird viel in diese Anlage investiert und bei $\sigma_{EK} = 0.2$ weniger.

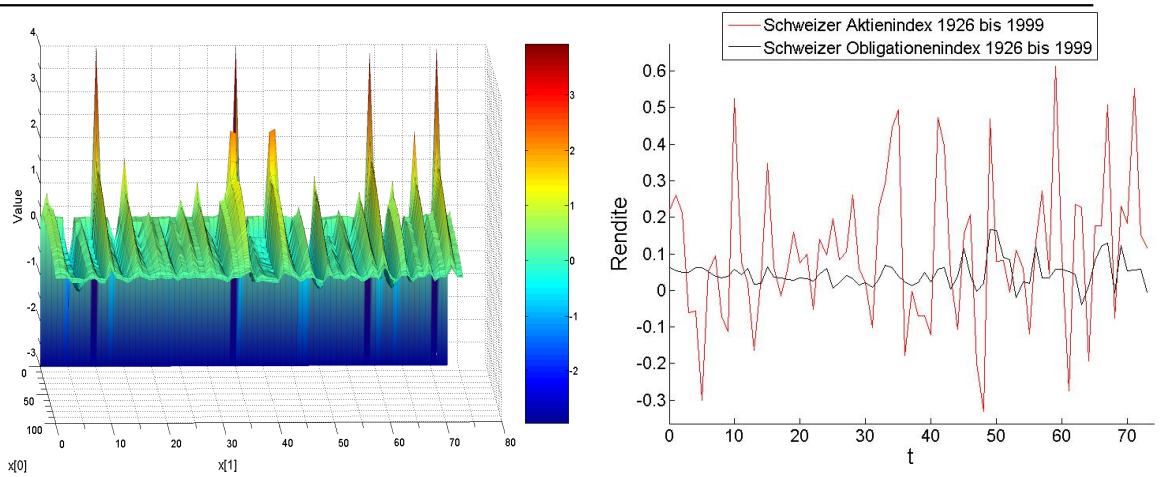

Abbildung 6.30: Optimale Kontrolle $u[1]$ für Beispiel SI_stoch_c / Zinsverlauf aus Abb. 4.1

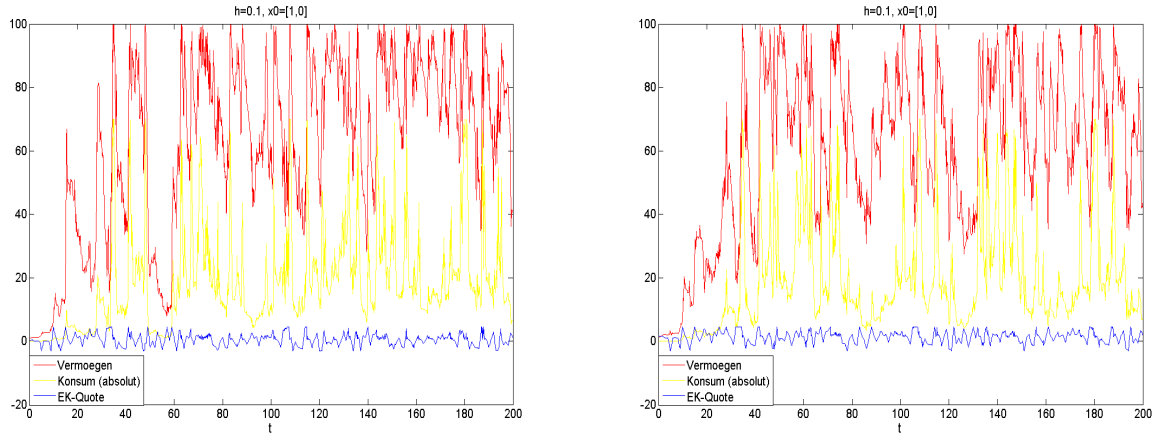

Abbildung 6.31: Optimale Trajektorien für Beispiel SI_stoch_a und SI_stoch_b

Im Gegensatz zum Abschnitt 6.4.1 konvergiert das Vermögen bei den optimalen Trajektorien in Abbildung 6.31 und 6.32 nicht gegen 0. Dafür schwankt das Vermögen hier viel stärker (zwischen ca. 10 und 100), was darauf zurückzuführen ist, dass die Kontrolle $u[1]$ hier in sämtlichen Beispielen größer als in Abschnitt 6.4.1 gewählt

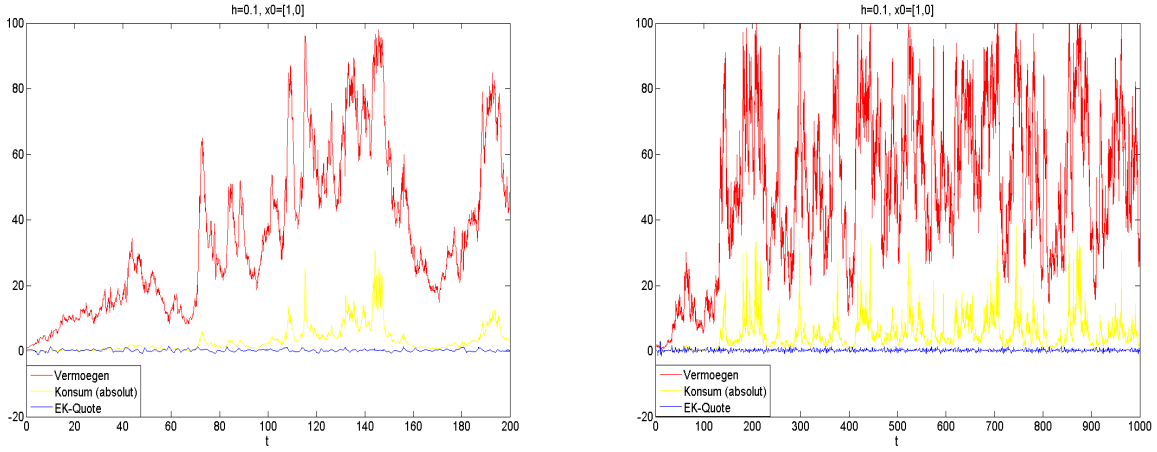

Abbildung 6.32: 2 zufällig erzeugte optimale Trajektorien (mit $t = 200$ bzw. $t = 1000$) für Beispiel SI_stoch_c

---

wurde.
Dass der stochastische Einfluss hier größer ist, wird durch folgende Tabelle verdeutlicht. Dabei sei für $u[1]$ immer der Mittelwert genommen, $x[0]$ sei das Vermögen[11]:

Beispiel	stochast. Einfluss durch σ_{EK}	stochast. Einfluss durch σ_{FK}
SP_stoch_a	$0.2 \cdot (-0.016593) \cdot x[0] = -0.0033186 \cdot x[0]$	0
SP_stoch_b	0	$0.2 \cdot (1 - 0.960994) \cdot x[0] = 0.0078012 \cdot x[0]$
SP_stoch_c	$0.5 \cdot 0.122177 \cdot x[0] = 0.0610885 \cdot x[0]$	$0.2 \cdot (1 - 0.122177) \cdot x[0] = 0.1755646 \cdot x[0]$
SI_stoch_a	$0.2 \cdot 0.677267 \cdot x[0] = 0.1354534 \cdot x[0]$	0
SI_stoch_b	0	$0.2 \cdot (1 - 1.456400) \cdot x[0] = -0.09128 \cdot x[0]$
SI_stoch_c	$0.5 \cdot 0.217805 \cdot x[0] = 0.1089025 \cdot x[0]$	$0.2 \cdot (1 - 0.217805) \cdot x[0] = 0.156439 \cdot x[0]$

Da der stochastische Einfluss sich sowohl positiv als auch negativ auf das Vermögen auswirken kann, muss man die Absolutwerte der Tabelle vergleichen. Dabei ist die Summe des Einflusses durch σ_{EK} und σ_{FK} auf das Vermögen $x[0]$ bei SI_stoch_i größer als bei SP_stoch_i $(i = a, b, c)$. Dies kann die stärkeren Schwankungen des Vermögens bei SI_stoch_i $(i = a, b, c)$ sowohl nach oben wie auch nach unten erklären.

[11] für die Berechnung: siehe Gleichung (4.6)

Jetzt soll untersucht werden, wie sich die optimale Kontrolle $u[1]$ auf dem Gitter verändert, wenn ein stochastischer Einflussfaktor (σ_{EK}) variiert und der andere (σ_{FK}) fest auf 0 gewählt wird. Dazu sei der Kontrollwertebereich im Folgenden $U = [0, 1] \times [-3, 4.5]$ und $\sigma_{FK} = 0$ gewählt. Dabei seien folgende Beispiele betrachtet:

Beispiel	σ_{EK}	$max_{u[1]}$	$\mu_{u[1]}$	Abbildung
SI_det_a	0	4.500000	1.796227	siehe Abschnitt 6.3.2
SI_stoch_d	0.1	4.500000	1.352653	6.34 & 6.39
SI_stoch_e	0.5	2.525391	0.127191	6.35 & 6.39
SI_stoch_f	1	0.933105	0.015263	6.35 & 6.39
SI_stoch_g	1.25	0.747070	0.000475	6.36 & 6.40
SI_stoch_h	1.5	0.481934	-0.008139	6.36 & 6.40
SI_stoch_i	1.75	0.408691	-0.013469	6.37 & 6.40
SI_stoch_j	2	0.292236	-0.017160	6.37 & 6.41
SI_stoch_k	10	0.011719	-0.029213	6.38 & 6.41
SI_stoch_l	100	0.000000	-0.029703	6.38 & 6.41

Hierbei ist $max_{u[1]}$ der Maximalwert und $\mu_{u[1]}$ der Mittelwert von $u[1]$ auf dem Gitter.

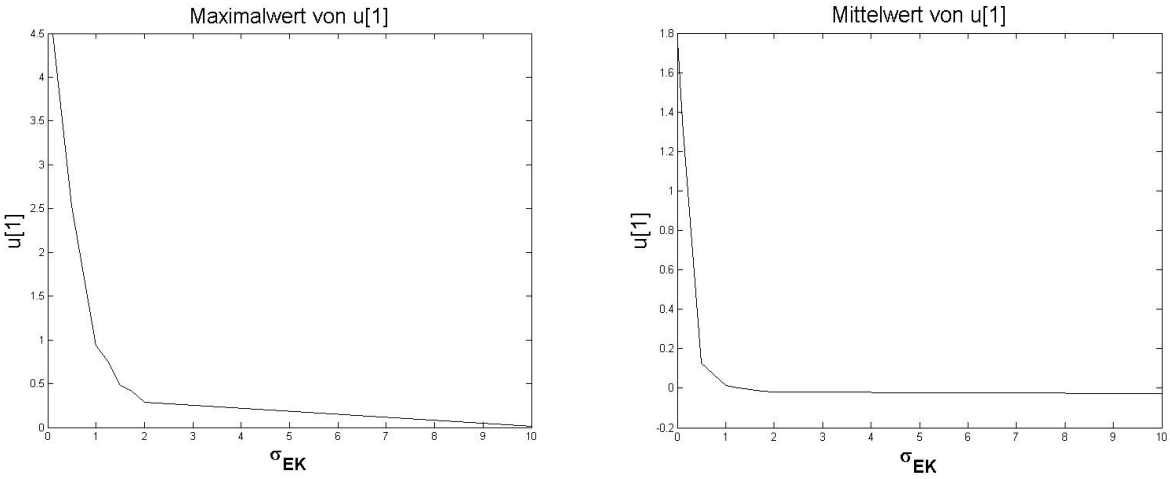

Abbildung 6.33: $max_{u[1]}$ und $\mu_{u[1]}$ in Abhängigkeit von σ_{EK}

In Abbildung 6.33 ist der Zusammenhang zwischen σ_{EK} (bei $\sigma_{FK} = 0$) und $max_{u[1]}$ bzw. $\mu_{u[1]}$ graphisch dargestellt. Man sieht, dass bei einem steigenden σ_{EK} das maximale $u[1]$ gegen 0 konvergiert, d.h. in den Schweizer Aktienindex wird bei $\sigma_{EK} = 10$ kaum noch investiert.
In den Abbildungen 6.34 bis 6.38 sieht man die Veränderung der Kontrollwerte $u[1]$ auf dem Gitter $x = x[0] \times x[1]$ vom deterministischen Fall zu dem mit $\sigma_{EK} = 100$. Gleich ist sämtlichen Beispielen der wellenförmige Verlauf in $x[1]$-Richtung, welcher schon am Anfang dieses Abschnittes erläutert wurde. Auffällig ist, dass die Ampli-

tuden der Wellen umso kleiner werden, desto größer σ_{EK} gewählt wurde.
Bei $\sigma_{EK} = 0$ nimmt $u[1]$ auf dem gesamten Gitter entweder den Wert -3 oder 4.5 an. Außer bei einem Vermögen $x[0]$ nahe der oberen Grenze 100, was aber numerische Effekte sind, die durch die Beschränkung des Vermögens nach oben auftreten.[12] Im Beispiel SI_stoch_d nimmt $u[1]$ ebenfalls die Werte wie in SI_det_a an, nur dass die numerischen Effekte eher auftreten (ab circa $x[0] = 90$). Also wirkt sich ein kleiner stochastischer Einfluss von $\sigma_{EK} = 0.1$ für den hier gewählten Kontrollwertebereich nur gering auf die Kontrolle $u[1]$ aus.
Ab Beispiel SI_stoch_e erkennt man in dem periodischen Verlauf von $u[1]$ in $x[1]$-

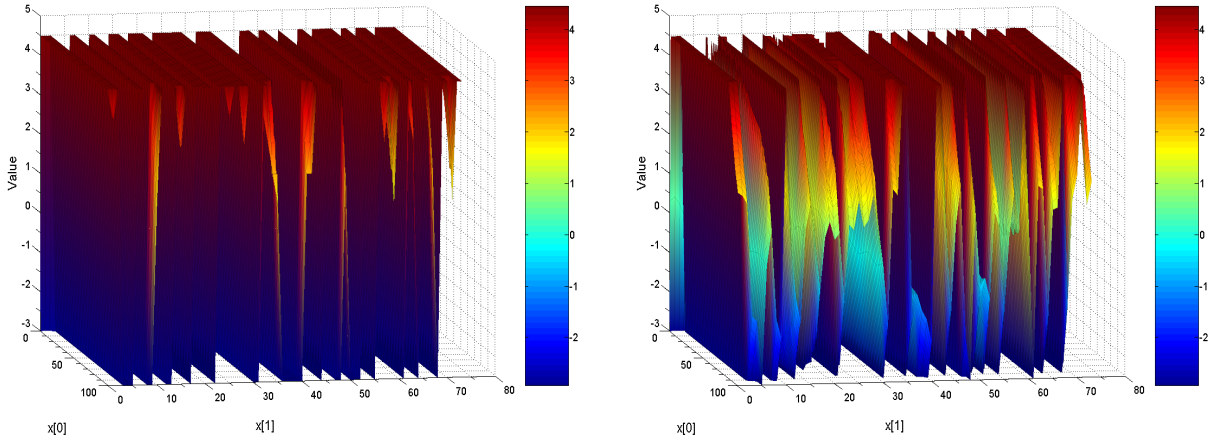

Abbildung 6.34: Optimale Kontrolle $u[1]$ für Beispiel SI_det_a und SI_stoch_d

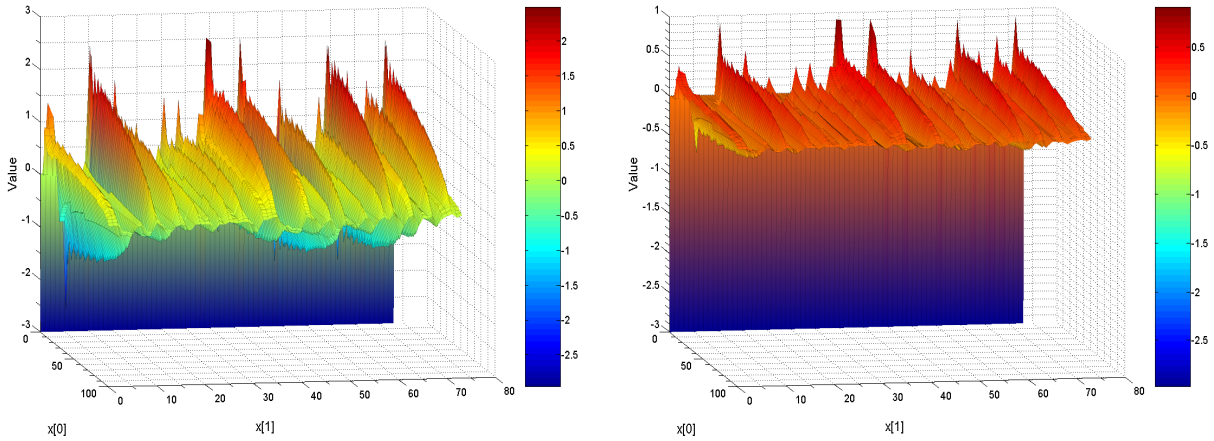

Abbildung 6.35: Optimale Kontrolle $u[1]$ für Beispiel SI_stoch_e und SI_stoch_f

[12]Der Algorithmus setzt den Rückgabewert des stochastischen Eulerverfahrens auf die obere Grenze $hi[0]$ des Vermögens, wenn er größer als $hi[0]$ ist. Daher kann als optimale Kontrolle $u[1]$ ein kleinerer Wert zurückgegeben werden, als wenn $hi[0]$ größer gewählt worden wäre, da bei der Optimierung $u[1]$ von der unteren Grenze zur oberen Grenze in vorgegebenen Diskretierungsabständen durchlaufen wird.

Richtung den Verlauf des Schweizer Aktienindexes wieder (vgl. Abbildung 6.30). Die Amplitude schwächt sich mit zunehmenden σ_{EK} immer weiter ab (vgl. SI_stoch_f bis SI_stoch_j) bis der Einfluss des Schweizer Aktienindexes in SI_stoch_k in Abbildung 6.38 kaum noch erkennbar ist. Bei einem $\sigma_{EK} = 100$ in SI_stoch_l wird das gesamte Kapital in den Schweizer Obligationenindex investiert, da dort $\sigma_{FK} = 0$ ist.

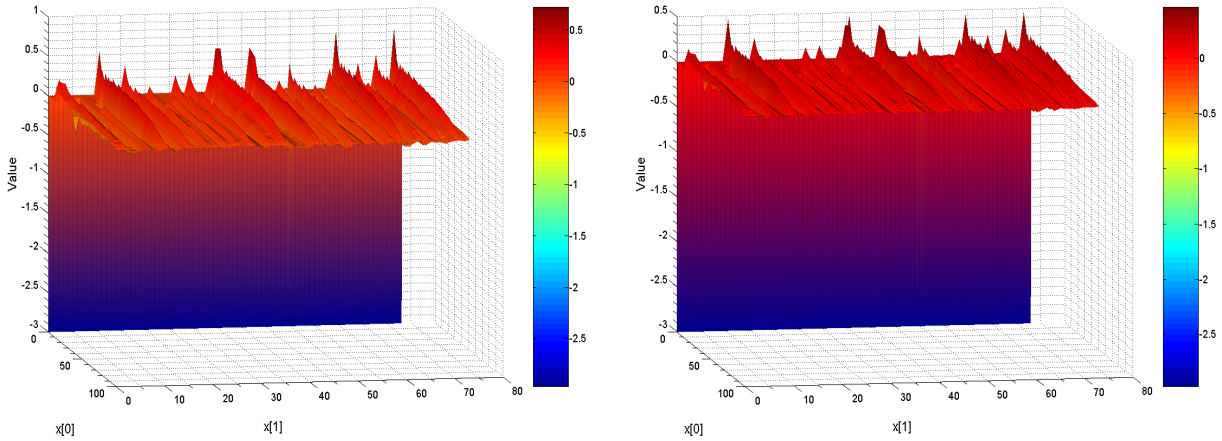

Abbildung 6.36: Optimale Kontrolle $u[1]$ für Beispiel SI_stoch_g und SI_stoch_h

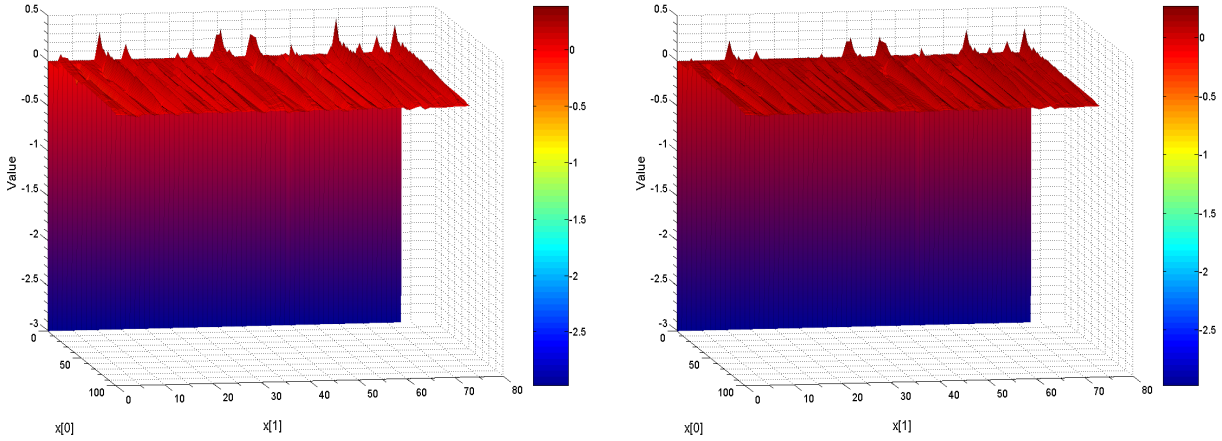

Abbildung 6.37: Optimale Kontrolle $u[1]$ für Beispiel SI_stoch_i und SI_stoch_j

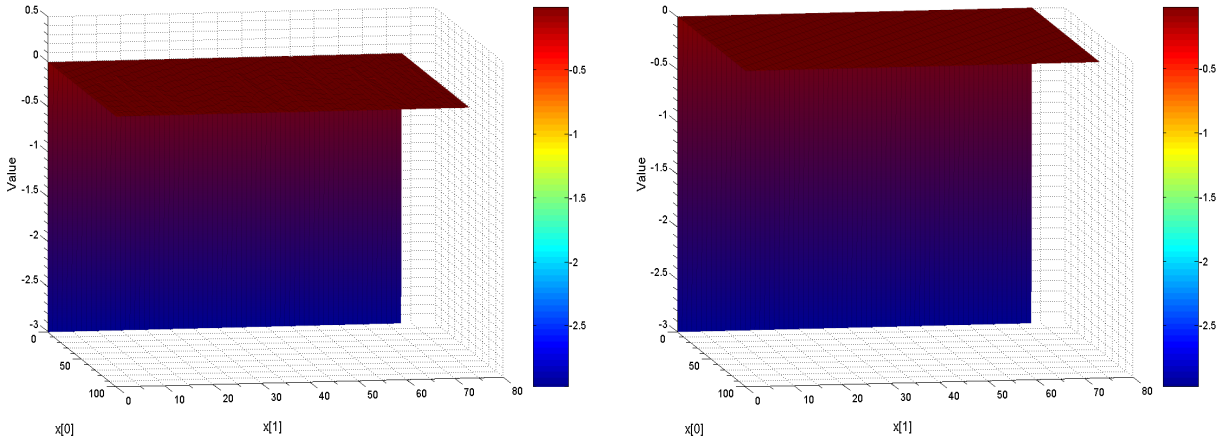

Abbildung 6.38: Optimale Kontrolle $u[1]$ für Beispiel SI_stoch_k und SI_stoch_l

---

In den Abbildungen 6.39 bis 6.41 sind die optimalen Trajektorien dargestellt. Auffällig ist, dass sich das Vermögen (rot) für alle Beispiele auf ein Intervall einpendelt und nicht gegen 0 konvergiert. Bis zu dem $\sigma_{EK} = 1.25$ erreicht es dabei noch die obere Schranke 100. Ab $\sigma_{EK} \geq 2$ pendelt sich das Vermögen auf einem niedrigeren Niveau ein, welches zwischen 2.5 und 3.5 liegt.
Beim absoluten Konsum kann man beobachten, dass dieser bei $\sigma_{EK} = 0.1$ noch sehr hoch liegt und Werte über 100 erreicht. Bei $\sigma_{EK} = 0.5$ beträgt der Konsum nur noch maximal 20, bei $\sigma_{EK} = 100$ bei nur noch maximal 0.2. Bei SI_stoch_l hat die Kontrolle $u[1]$ für alle t den Wert 0, d.h. die Gesamtrendite hängt nur noch vom Obligationenindex ab, welcher keinem stochastischen Einfluss unterliegt. Daher verläuft die Schwankung des Vermögen sehr regelmäßig, wobei die Periodenlänge der Anzahl der Zinsdaten (74) entspricht.

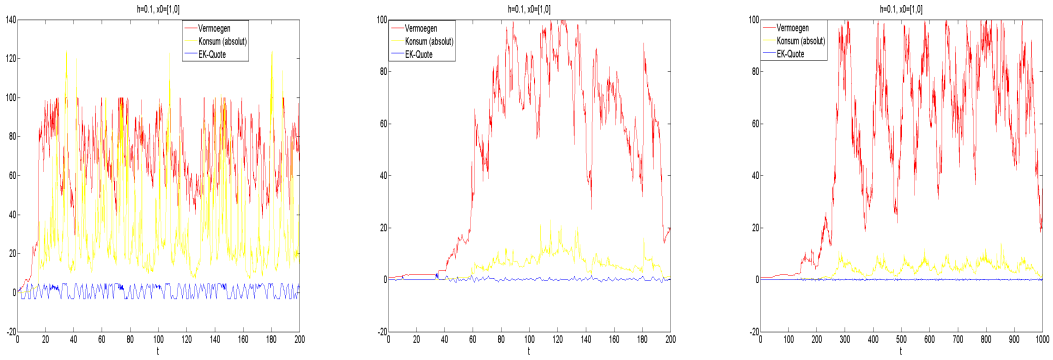

Abbildung 6.39: Optimale Trajektorien für Beispiel SI_stoch_d, SI_stoch_e und SI_stoch_f

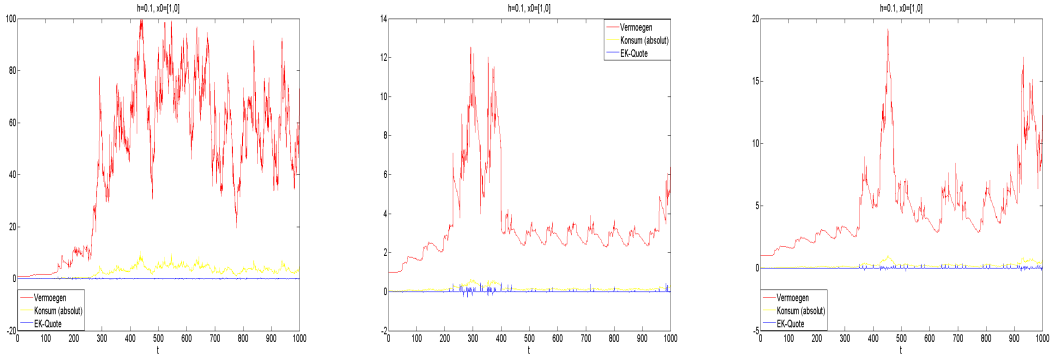

Abbildung 6.40: Optimale Trajektorien für Beispiel SI_stoch_g, SI_stoch_h und SI_stoch_i

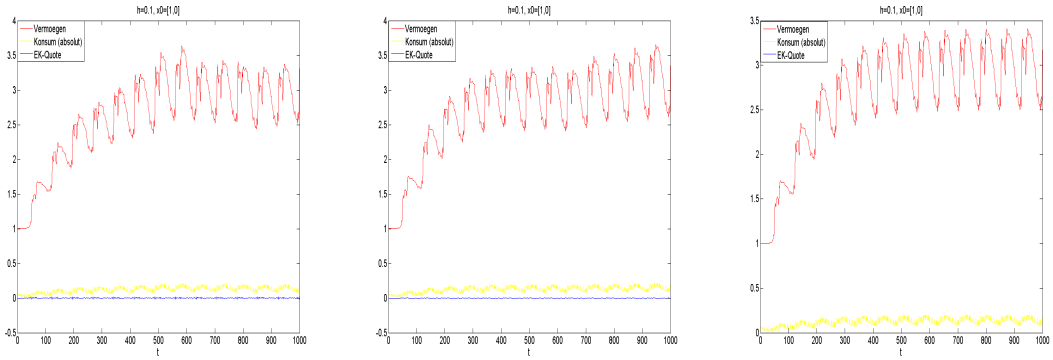

Abbildung 6.41: Optimale Trajektorien für Beispiel SI_stoch_j, SI_stoch_k und SI_stoch_l

Kapitel 7

Zusammenfassung

Ziel dieser Arbeit war es, das um einen stochastischen Einfluss erweiterten Modell für die Optimierung eines Portfolios, bestehend aus zwei Anlagen verschiedener Renditen und Renditeschwankungen, mit dem deterministischen zu vergleichen. Dazu wurde im zweiten Kapitel zunächst ein Überblick über die Portfoliotheorie verschafft um ein grundsätzliches Verständnis für die Finanzwirtschaft, das Portfolio-Management und das Markowitz- sowie CAP- Modell zu vermitteln.

Das dritte Kapitel dagegen legte die mathematischen Grundlagen für die stochastische dynamische Optimierung und die Diskretisierung, damit im vierten die Modellierung des Problems für den deterministischen sowie stochastischen Fall erfolgen konnte. Schwerpunkt dabei war, wie der stochastische Einfluss mit Hilfe der schwachen Approximation des Wiener Prozesses und des stochastischen Einschrittverfahrens numerisch umgesetzt werden kann. Abschließend wurde noch die Konkavität des Funktionals $J_T(x, u)$ für alle T in x_0 und u bewiesen, wodurch jedes lokale Maximum von $J_T(x, u)$ über u auch global ist. Dies half bei der Konstruktion des zugehörigen Optimierungsalgorithmus.

In Abschnitt 5.1.2 auf Seite 65 wurde ein heuristisches Verfahren vorgestellt, welches als Voraussetzung braucht, dass ein lokales Maximum auch global ist. Dass die Konkavität kein hinreichendes Kriterium dafür ist, dass das Verfahren tatsächlich das Maximum findet, wurde anhand eines Gegenbeispiels aufgezeigt. Grund dafür ist, dass über diskrete Punkte optimiert wird. Um zu zeigen, dass er dennoch für das Modell (4.1) und (4.6) gute Lösungen liefert, wurde der Algorithmus in Kapitel 6 für verschiedene Parameter getestet und mit Ergebnissen auf einem stark diskretisierten Startgitter verglichen. Die Resultate eines Startgitters von u mit 11×11 Gitterpunkten wichen im relativen Mittelwert um 0.0003 und in der relativen Standardabweichung um 0.0002 von denen eines Startgitters mit 401×401 Gitterpunkten ab. Daher wurde der Algorithmus mit den Parametern in Tabelle 6.1 verwendet, auch aufgrund der immensen Zeitersparnis. Die Bedeutung dieser war der Hauptgrund nach einem schnelleren Algorithmus zu suchen, da die Optimierung über $u = (u_0, u_1) \in U$ (Kon-

trollraum) für jedes $x = (x_0, x_1) \in X$ (Zustandsraum) gemacht werden muss und dies in jedem Iterationsschritt der Werteiteration. Ein rechnerischer Vergleich auf $U = [0, 0.7] \times [-3, 4.5]$ mit der gewünschten Genauigkeit von 0.001 ergab, dass in einer gewöhnlichen Diskretisierung des gesamten Gitters $\approx$ 14000-mal soviele Gitterpunkte untersucht werden müssten als bei dem Algorithmus mit den Parametern aus Tabelle 6.1. Eine Verdoppelung der Zeit zur Berechnung des optimalen u führt jedoch annähernd zu einer Verdoppelung der Gesamtzeit, den die Werteiteration braucht.
In Abschnitt 3.2.2 wurde schon der stark anwachsende numerische Aufwand in Abhängigkeit der Dimension des Zustandsraums erwähnt. Die Untersuchungen in Kapitel 6 haben gezeigt, dass der numerische Aufwand für die dynamische Optimierung auch stark von der Dimension der Kontrollen abhängt. Außerdem erhöht sich der Zeitaufwand gegenüber dem deterministischen Fall um den Faktor 2^s, wobei s die Anzahl der verschiedenen Wiener Prozesse ist (hier maximal 2, wenn $\sigma_{EK} > 0$ und $\sigma_{FK} > 0$; dazu vgl. Seite 68 in *Berechnung von temp*).

Platz für weitere Untersuchungen bietet hier die Frage, wie man die Optimierung über u noch effizienter, auch für höhere Dimensionen, machen und in Kombination mit der Strategie-Iteration (vgl. [4] Grüne, 2004, S.47-49) eine schnellere Konvergenz als bei der Werteiteration erreichen kann.

Die Untersuchungen in Abschnitt 6.3 und 6.4 zeigten, dass sich ein stochastischer Einfluss immer negativ auf die optimale Wertefunktion, den optimalen Konsum sowie auf das Anlageverhalten auswirkt.
Der Investor wird dabei einen umso höheren Anteil seines Vermögen in eine Anlage investieren, desto geringer sie stochastisch beeinflusst wird und desto höher dessen Rendite ist. Zweiteres beobachtet man dabei auch im deterministischen Fall. Die zusätzliche Erkenntnis ist aber, dass, wenn z.B. Anlage A einen hohen stochastischen Einfluss hat und Anlage B keinen, der Investor tendenziell sein gesamtes Vermögen in B investieren wird. Wenn jedoch die Rendite von A genügend weit über der von B liegt, dann wird er auch einen Teil des Kapitals in A investieren, trotz der stochastischen Unsicherheit. Dies lässt sich aus dem wellenförmigen Verlauf der optimalen Kontrolle $u[1]$ in $x[1]$-Richtung herleiten, z.B. vergleiche man dazu in Abbildung 6.29 und 6.30 den Verlauf von $u[1]$ mit dem des Schweizer Aktienindexes. Dieser Effekt nimmt (bei gleichbleibenden Zinsdaten) mit zunehmenden stochastischen Einfluss auf A immer weiter ab.
Eine stochastische Störung kann das Vermögen entweder zusätzlich erhöhen oder senken. Welche der beiden Fälle eintritt, hängt vom Zufall ab, tritt aber mit gleicher Wahrscheinlichkeit ein. Der Investor muss aber den Konsum so wählen, dass im schlecht möglichsten Fall noch genügend Kapital zum Reinvestieren vorhanden ist. Dies erklärt die gegenüber dem deterministischen Fall niedrigeren Werte des Konsums und damit auch der optimalen Wertefunktion.

Bei der Modellierung des stochastischen Einflusses ergeben sich weitere Möglichkeiten anzuknüpfen. Der Wiener Prozess erzeugt nur $\mathcal{N}(0, h)$-verteilte Zufallsvariablen. Interessant wäre auch die Untersuchung des Problems mit anderen Verteilungen. Man könnte zum Beispiel mit Hilfe der Poisson-Verteilung über Sprung-Diffusion-Stochastische-Differentialgleichungen (vgl. [11] Lasić, 2002, S.17/18) seltene stochastische Ereignisse modellieren. Dazu zählt auch der Totalverlust, d.h. der Investor verliert mit geringer Wahrscheinlichkeit das gesamte Kapital einer der beiden Anlagen. Dies kann z.B. nach dem Kauf von Staatsanleihen in Fremdwährung bei Zahlungsunfähigkeit des entsprechenden Landes passieren. Ein insbesondere in Hinblick auf die aktuelle Finanz- und Weltwirtschaftskrise nicht unmögliches Szenario.

Anhang A

Programm-Quellcodes

A.1 Matlab-Quellcodes

Dichtefunktion.m

```
 1 function Dichtefunktion(name)
 2 %Gibt ein Approximation der Dichtefunktion für eine Menge von
 3 %Zufallszahlen zurück
 4
 5 b=load(name);
 6 sb=size(b);
 7 x=-4:0.1:4;
 8 for i=1:81
 9   y(i)=0;
10 end
11 for j=1:sb(1)
12   for i=1:81
13     if(b(j,1)>x(i)-0.05 && b(j,1)<x(i)+0.05)
14       y(i)=y(i)+1;
15     end
16   end
17 end
18 for i=1:81
19   y(i)=y(i)/sb(1);
20 end
21 plot(x,y,'k-');
```

differenz.m

```
1 function differenz(name1,name2)
2 %Berechnet die Differenz der Werte der beiden Gitter
3 %(gleicher Größe) name1-name2
```

```
4  file=fopen('diff.asc','w');
5
6  a = load(name1);
7  sa = size(a);
8  b = load(name2);
9  sb = size(b);
10
11 for i=1:sa(1)
12   fprintf(file,'%f %f %f %f %f %f %f %f\n',
       a(i,1),a(i,2),a(i,3),a(i,4),
       a(i,5)-b(i,5),a(i,6)-b(i,6),a(i,7)-b(i,7),a(i,8)-b(i,8));
13 end
14
15 fclose(file);
```

gridplot2d.m

```
1  function gridplot2d(name)
2  % gridplot2d(filename)
3  %
4  % graphische Darstellung einer mittels "export_gridandval"
5  % exportierten Gitterfunktion in 2d
6  %
7  % (C) Lars Gruene 2007
8  % modifiziert von Maik Müller 2009
9
10 b = load(name);
11 sb = size(b);
12 set(axes,'FontSize',13);
13 for i=1:sb(1)
14   x=[b(i,1), b(i,3), b(i,3), b(i,1)];
15   y=[b(i,2), b(i,2), b(i,4), b(i,4)];
16   z=[b(i,5), b(i,6), b(i,8), b(i,7)];
17   h=patch(x,y,z,z);
18   set(h,'EdgeAlpha',0.1);
19 end
20 grid minor;
21 xlabel('x[0]','FontSize',16);
22 ylabel('x[1]','FontSize',16);
23 zlabel('Value','FontSize',16);
24 view(-45, 30);
25 colorbar('FontSize',15);
```

maxmin.m

```
 1 function maxmin(name)
 2 %Programm um minimalen und maximalen Wert
   auf einem Gitter zu finden
 3 a = load(name);
 4 sa = size(a);
 5
 6 %Minimum finden
 7 [A(1),I1]=min(a(:,5));
 8 [A(2),I2]=min(a(:,6));
 9 [A(3),I3]=min(a(:,7));
10 [A(4),I4]=min(a(:,8));
11
12 [erg,I]=min(A);
13
14 if I==1
15   index(1)=a(I1,1);
16   index(2)=a(I1,2);
17 end
18 if I==2
19   index(1)=a(I1,3);
20   index(2)=a(I1,2);
21 end
22 if I==3
23   index(1)=a(I1,1);
24   index(2)=a(I1,4);
25 end
26 if I==4
27   index(1)=a(I1,3);
28   index(2)=a(I1,4);
29 end
30
31 fprintf('Der Minimalwert liegt bei (%f,%f) und betraegt
   %f\n',index(1),index(2),erg);
32
33 %Maximum finden
34 [A(1),I1]=max(a(:,5));
35 [A(2),I2]=max(a(:,6));
36 [A(3),I3]=max(a(:,7));
37 [A(4),I4]=max(a(:,8));
38
39 [erg,I]=max(A);
40
```

```
41 if I==1
42   index(1)=a(I1,1);
43   index(2)=a(I1,2);
44 end
45 if I==2
46   index(1)=a(I1,3);
47   index(2)=a(I1,2);
48 end
49 if I==3
50   index(1)=a(I1,1);
51   index(2)=a(I1,4);
52 end
53 if I==4
54   index(1)=a(I1,3);
55   index(2)=a(I1,4);
56 end
57
58 fprintf('Der Maximalwert liegt bei (%f,%f) und betraegt
   %f\n',index(1),index(2),erg);
59
60 %Mittelwert
61 max_x=max(a(:,3))/min(a(:,3));
62 max_y=max(a(:,4))/min(a(:,4));
63 n=size(a(:,5),1);
64 temp=0;
65 for i=1:max_y
66   temp=temp+sum(a( ((i-1)*max_x+1) : i*max_x,5 ))+a(i*max_x,6);
67 end
68 temp=temp+sum(a((n-max_x+1):n,7))+a(n,8);
69 Mittelwert=temp/((max_x+1)*(max_y+1));
70 %Standardabweichung
71 temp=0;
72 for i=1:max_y
73   for j=(i-1)*max_x+1:i*max_x
74     temp=temp+(a(j,5)-Mittelwert)^2;
75   end
76   temp=temp+(a(i*max_x,6)-Mittelwert)^2;
77 end
78 for i=n-max_x+1:n
79   temp=temp+(a(i,7)-Mittelwert)^2;
80 end
81 temp=temp+(a(n,8)-Mittelwert)^2;
```

```
82 temp=temp/((max_x+1)*(max_y+1));
83 Standardabweichung=sqrt(temp);
84
85 fprintf('Mittelwert= %f Standardabweichung=%f\n'
   ,Mittelwert,Standardabweichung);
```

plot_konvergenz.m

```
 1 function plot_konvergenz(name)
 2 %Programm um die Konvergenz der Werteiteration
     nachzuvollziehen
 3
 4 b=load(name);
 5 sb=size(b);
 6 max=0;
 7 zaehler=0;
 8 for i=3:sb(1)
 9   if b(i-1,2)<b(i,2)
10     zaehler=zaehler+1;
11     max2=b(i,2)-b(i-1,2);
12     if max2>max
13       max=max2;
14     end
15   end
16 end
17 plot(b(1:sb(1),1),b(1:sb(1),2),'k-');
18 title(['maximaler Anstieg der maximalen Abweichung in V=',
   'num2str(max),| Anzahl Anstiege=',num2str(zaehler)]);
```

plot_zins.m

```
 1 function plot_zins(name1,name2)
 2 %Plottet 1 oder 2 Zinsverlaeufe
 3 %Input: Gitter, mit gridgen erzeugt
 4 %Wenn nur einer geplottet werden soll, dann den 2.
 5 %mit '' angeben, z.B. plot_zins('1.asc','')
 6
 7 %Daten einlesen
 8 a=load(name1);
 9 sa=size(a);
10 zins_a=[a(:,1),a(:,3)];
```

```
11 zins_a(sa(1)+1,:)=[a(sa(1),2),a(sa(1),4)];
12
13 b=load(name2);
14 sb=size(b);
15 zins_b=[b(:,1),b(:,3)];
16 zins_b(sb(1)+1,:)=[b(sb(1),2),b(sb(1),4)];
17
18 %Ausgabe der Daten
19 hold on;
20 if sa(1)>1
21   plot(zins_a(:,1),zins_a(:,2),'r-');
22 end
23 if sb(1)>1
24   plot(zins_b(:,1),zins_b(:,2),'k-');
25 end
26 hold off;
27 min_y=1.1*min(min(zins_a(:,2)),min(zins_b(:,2)));
28 max_y=1.1*max(max(zins_a(:,2)),max(zins_b(:,2)));
29 axis([0 sb(1)+1 min_y max_y]);
30 title('Zinsverlauf');
31 xlabel('t');
32 ylabel('Rendite');
```

Trajekplot.m

```
1  function Trajekplot(name)
2  %Programm zum Plotten der optimalen Trajektorien
3  %Trajekplot(filename.txt)
4
5  %Daten einlesen
6  b=load(name);
7  sb = size(b);
8  %Ausgabe der Daten
9  set(axes,'FontSize',15);
10 plot(b(2:sb(1),2),b(2:sb(1),1),'r-',b(2:sb(1),2),
     b(2:sb(1),3),'y-',b(2:sb(1),2),b(2:sb(1),4),'b-');
     %,b(2:sb(1),2),b(2:sb(1),5),'k-');
11 legend('Vermoegen','Konsum (absolut)', 'EK-Quote',3);
     % 'optimale Wertefunktion',4);
12 xlabel('t');
13 title(['h=',num2str(b(1,1)),', x0=[',num2str(b(1,2)),
     ',',num2str(b(1,3)),']']);
```

Zinsstatistik.m

```
 1 function Zinsstatistik(input1,input2)
 2 %Berechnet zu zwei Zinsverlaeufen den EW, Varianz, Covarianz
 3 %Input: Gitter in asc-Datei, Gitter dabei gleichgroß
 4 a=load(input1);
 5 sb = size(a);
 6 b=load(input2);
 7
 8 %Erwartungswert berechnen
 9 Er_A=(sum(a(1:sb(1),3))+a(sb(1),4))/(sb(1)+1);
10 Er_B=(sum(b(1:sb(1),3))+b(sb(1),4))/(sb(1)+1);
11
12 %Standardabweichung sigma berechnen
13 Var_A=0;
14 Var_B=0;
15 for i=1:sb(1)
16   Var_A=Var_A+(a(i,3)-Er_A)^2;
17   Var_B=Var_B+(b(i,3)-Er_B)^2;
18 end
19 Var_A=(Var_A+(a(sb(1),4)-Er_A)^2)/(sb(1)+1);
20 Var_B=(Var_B+(b(sb(1),4)-Er_B)^2)/(sb(1)+1);
21 sigma_A=sqrt(Var_A);
22 sigma_B=sqrt(Var_B);
23
24 %Covarianz berechnen
25 Cov_AB=0;
26 for i=1:sb(1)
27   Cov_AB=Cov_AB+(a(i,3)-Er_A)*(b(i,3)-Er_B);
28 end
29 Cov_AB=(Cov_AB+(a(sb(1),4)-Er_A)*(b(sb(1),4)-Er_B))/(sb(1)+1);
30 %Korrelationskoeffizent berechnen
31 rho=Cov_AB/(sigma_A*sigma_B);
32 fp=fopen('Zinsdaten.txt','w');
33 fprintf(fp,'E(r_A)=%f E(r_B)=%f\n',Er_A,Er_B);
34 fprintf(fp,'sigma_A=%f sigma_B=%f\n',sigma_A,sigma_B);
35 fprintf(fp,'Cov_AB=%f rho=%f\n',Cov_AB,rho);
36 fclose(fp);
```

A.2 C-Quellcodes

Zinserzeugung.c

```
1  /*************************************************************/
2  /*                                                           */
3  /* Programm zum Erzeugen von 2 normalverteilten
                                                 Zinsverlaeufen */
4  /* mit vorgegebenen EW, Standardabweichung und Korrelation */
5  /*                                                           */
6  /* gcc -c Zinserzeugung.c                                    */
7  /* gcc Zinserzeugung.o -lm -o Zinserzeugung                  */
8  /* Zinserzeugung eingabe.txt                                 */
9  /* Maik Mueller                                              */
10 /* Stand 19.02.2009                                          */
11 /*                                                           */
12 /*************************************************************/
13 #define pi 3.14159265358979323846264338327  //Kreiszahl Pi
14
15 #include <time.h>
16 #include <math.h>
17 #include <stdio.h>
18 #include <stdlib.h>
19
20 //Zufallszahl zw. 0 und 1 erzeugen
21 double erzeuge_Zufallszahl()
22 {
23   double x;
24   x=rand();
25   return x=x/RAND_MAX;
26 }
27
28 int main(int args, char ** argv)
29 {
30   if(args != 2)
31   {
32     printf("Aufruf: %s Parameterdatei.dat!",argv[0]);
33     exit(1);
34   }
35   FILE *fp,*fp1,*fp2;
36   char str[8];
37   double x,y, muh[2],sigma[2],u[2],rho,phi;
38   char filename[100],file[100];
```

```
39    int i=0,anzahl;
40
41    time_t sekunden;
42    sekunden=time(NULL);
43    printf("Now: %ld\n", sekunden);
44
45    //Daten einlesen
46    fp = fopen( argv[1], "r");
47    fscanf(fp,"%s %d\n",str,&anzahl);
48    //EW und Standardabweichung der 1. Zinsreihe
49    fscanf(fp,"%s %lf\n",str,&muh[0]);
50    fscanf(fp,"%s %lf\n",str,&sigma[0]);
51    //EW und Standardabweichung der 2. Zinsreihe
52    fscanf(fp,"%s %lf\n",str,&muh[1]);
53    fscanf(fp,"%s %lf\n",str,&sigma[1]);
54    //Korrelation
55    fscanf(fp,"%s %lf\n",str,&rho);
56
57    //Kontrollausgabe
58    printf("Anzahl Daten=%d\n",anzahl);
59    printf("EW1=%lf EW2=%lf\n",muh[0],muh[1]);
60    printf("sigma1=%lf sigma2=%lf\n",sigma[0],sigma[1]);
61    printf("rho=%lf\n",rho);
62    fclose(fp);
63
64    printf("Bitte eine Bezeichung 'filename' eingeben?\n
      Ausgegeben werden 2 Dateien: EK_filename und FK_filename.\n");
65    scanf("%s", &filename);
66    sprintf(file,"Zinsen/EK_%s.txt",filename);
67    fp1 = fopen(file, "w");
68    sprintf(file,"Zinsen/FK_%s.txt",filename);
69    fp2 = fopen(file, "w");
70    //Gauss-verteilte Zufallszahlen erzeugen
71    phi=asin(rho);
72    srand(sekunden);
73    for(i=0;i<anzahl;i++)
74    {
75      u[0]=erzeuge_Zufallszahl();
76      u[1]=erzeuge_Zufallszahl();
77      x=muh[0]+sigma[0]*sqrt(-2*log(u[0]))*sin(2*pi*u[1]);
78      fprintf(fp1,"%1.6f\n", x);
79      y=muh[1]+sigma[1]*sqrt(-2*log(u[0]))*cos(2*pi*u[1]-phi);
```

```
80      fprintf(fp2,"%1.6f\n", y);
81    }
82    fclose(fp1);
83    fclose(fp2);
84  }
```

Binde_in_Gitter.c

```
 1  /*************************************************************/
 2  /*                                                           */
 3  /* Programm, welches Zinsdaten aus 2 Dateien einliest und    */
 4  /* diese in 2 Gitter einbindet und diese jeweils als .asc-   */
 5  /* und .val-Datei abspeichert. Dabei muessen die Daten       */
 6  /* in den Eingabedateien in einer Spalte sein.               */
 7  /* Anmerkung: Anzahl Daten muss bei beiden Dateien gleich    */
 8  /* sein, da sonst untersch. grosse Gitter erzeugt werden     */
 9  /*erste Datei enthaelt EK_Zins (hoeheres Risiko+erw. Gewinn)*/
10  /* zweite Datei enthaelt FK_Zins (niedr. Risiko+erw. Gewinn)*/
11  /*                                                           */
12  /* gcc -c Binde_in_Gitter.c                                  */
13  /* gcc Binde_in_Gitter.o gridgen.o -lm -o Binde_in_Gitter    */
14  /* Binde_in_Gitter Datei_1.dat Datei_2.dat                   */
15  /*                                                           */
16  /* Maik Mueller                                              */
17  /* Stand 19.02.2009                                          */
18  /*                                                           */
19  /*************************************************************/
20
21  #include "Portfolio.h"
22
23  int main(int args, char ** argv)
24  {
25    if(args != 3)
26    {
27      printf("Aufruf: %s EK_Zinsdatei1.dat FK_Zinsdatei2.dat!"
          ,argv[0]);
28      exit(1);
29    }
30
31    FILE *fp;
32    qgrid *Gitter;
33    double lo[1],hi[1],Delta[1],y, x[1];
```

```
34    char filename[100];
35    int k, index,i;
36    lo[0]=0;
37    //Erzeugung der .asc- und .val-Dateien
38    for(i=0;i<2;i++)
39    {
40      //Daten einlesen
41      fp = fopen( argv[i+1], "r");
42
43      if(NULL == fp)
44      {
45        fprintf(stderr, "Fehler beim Oeffnen ...\n");
46        return EXIT_FAILURE;
47      }
48      //Anzahl Inputdaten ermitteln
49      k=0;
50      while(fscanf(fp,"%lf\n",&y)!=EOF)
51      {
52        k++;
53      }
54      fclose(fp);
55      //Gitter erzeugen
56      hi[0]=k-1;
57      Delta[0]=1;
58      //printf("%d\n",k);
59      Gitter=create_grid(lo,hi,Delta,1,0);
60
61      fp = fopen( argv[i+1], "r");
62      //Daten einlesen und in Gitter abspeichern
63      index=first_node(Gitter,x);
64      do
65      {
66        fscanf(fp,"%lf\n",&y);
67        set_current_nodevalue(Gitter,y);
68        index=next_node(Gitter,x);
69      }while(index!=-1);
70      printf("%s",argv[i+1]);
71      if(i==0)
72      {
73        sprintf(filename,"Zinsgitter/rEK_gitter.asc");
74        export_gridandval(Gitter,filename);
75        sprintf(filename,"Zinsgitter/rEK_gitter.val");
```

```
76        save_val(Gitter,filename);
77      }
78      else
79      {
80        sprintf(filename,"Zinsgitter/rFK_gitter.asc");
81        export_gridandval(Gitter,filename);
82        sprintf(filename,"Zinsgitter/rFK_gitter.val");
83        save_val(Gitter,filename);
84      }
85      fclose(fp);
86    }
87    //Ausgabe von diversen Statistiken (besonders auch
        der Anzahl der Daten fuer das Hauptprogramm)
88    fp = fopen( "Zinsgitter/Statistiken.txt", "w");
89    fprintf(fp,"Anzahl Daten %d:\n",k);
90    fprintf(fp,"Dies entspricht hi[1]=%d in eingabe.txt
        fuer das Hauptprogramm:\n",k-1);
91    fclose(fp);
92  }
```

Portfolio.h

```
1  /* Einbinden der Header-Datei */
2
3  #include "gridgen.h"
4
5  /* fuer mathematische Operationen */
6
7  #include <math.h>
8
9  #include <stdio.h>
10 #include <stdlib.h>
```

Portfolio.c

```
1  /*************************************************************/
2  /*                                                           */
3  /* Programm zur Loesung eines stochastischen
                Optimierungsproblems                          */
4  /* mit Dynamischer Programmierung                            */
5  /*                                                           */
```

```
 6 /* gcc -c gridgen.c                                                */
 7 /* gcc -c Portfolio.c                                              */
 8 /* gcc Portfolio.o gridgen.o -lm -o Portfolio                      */
 9 /* Portfolio eingabe.txt                                           */
10 /* Maik Mueller                                                    */
11 /* Stand 03.03.2009                                                */
12 /*                                                                 */
13 /*****************************************************************/
14
15 #include "Portfolio.h"
16
17 //-------------Deklaration der Konstanten/Parameter-----------
18 #define dim 2         //Dimension des Problems
19
20 struct Gitterkonstanten {
21   double lo[dim];       //untere Schranken
22   double hi[dim];       //obere Schranken
23   double Delta[dim];   //Groesse eines Gitterrechtecks
24 };
25
26 struct Problemparameter
27 {
28   double klein_delta;  //Abzinsungsfaktor
29   double gamma_r;     //relative Risikoaversion
30   double u_max[dim];   //maximaler [Konsumanteil,Anteil des
                          Vermoegens], investiert im Eigenkapital
31   double u_min[dim];  //minimaler...
32   double x0[dim];     //Startwert fuer opt Trajektorie
33   double x_max;       //x[1]_max fuer opt. Trajektorie
34
35   double sigma_EK;   //Gewichtung des stochastischen
                          Einflusses auf das EK-Vermoegen
36   double sigma_FK;   //Gewichtung des stochastischen
                          Einflusses auf das FK-Vermoegen
37 };
38
39 struct Diskretisierung
40 {
41   double epsilon;  //fuer Fehlervermeidung
42   double tol;   //Toleranz
43   double h;   //Schrittweite
44   //Konstanten fuer Algorithmus finde_max_besser
```

```
45    double anfdiskr;  //Startdiskretisierung (1/anfdiskr
        gibt an, in wieviel Teile das Intervall der
        Kontrollen geteilt wird)
46    double diskrverkl;   //gibt an, wie stark die
        Diskretisierung der Kontrollen verfeinert werden soll
        (0.5 entspricht Halbierung)
47    double umgebung;   //gibt an, in welcher Umgebung vom
        Optimum (Anzahl der Gitterpunkte) der aktuellen
        Diskretisierung weitergesucht werden soll, dabei muss
        umgebung>=(1/diskrverkl) gelten
48    double abbruch;   //Genauigkeit, die die Kontrollen
        erreichen sollen (Stelle nach dem Komma)
49  };
50
51  struct Konstanten
52  {
53    struct Gitterkonstanten  gitter;
54    struct Problemparameter  param;
55    struct Diskretisierung   diskr;
56  };
57
58  //------------Einlesen der Parameter aus einer Datei---------
59  struct Konstanten eingabe(FILE *fp)
60  {
61    struct Konstanten K;
62    char str[8];
63    //Problemparameter einlesen
64    fscanf (fp, "%s %lf", str, &K.param.klein_delta);
65    fscanf (fp, "%s %lf", str, &K.param.gamma_r);
66    fscanf (fp, "%s %lf", str, &K.param.u_max[0]);
67    fscanf (fp, "%s %lf", str, &K.param.u_max[1]);
68    fscanf (fp, "%s %lf", str, &K.param.u_min[0]);
69    fscanf (fp, "%s %lf", str, &K.param.u_min[1]);
70    fscanf (fp, "%s %lf", str, &K.param.x0[0]);
71    fscanf (fp, "%s %lf", str, &K.param.x0[1]);
72    fscanf (fp, "%s %lf", str, &K.param.x_max);
73    fscanf (fp, "%s %lf", str, &K.param.sigma_EK);
74    fscanf (fp, "%s %lf", str, &K.param.sigma_FK);
75    //Numerische Konstanten einlesen
76    fscanf (fp, "%s %lf", str, &K.diskr.epsilon);
77    fscanf (fp, "%s %lf", str, &K.diskr.tol);
78    fscanf (fp, "%s %lf", str, &K.diskr.h);
```

```
79    fscanf (fp, "%s %lf", str, &K.diskr.anfdiskr;
80    fscanf (fp, "%s %lf", str, &K.diskr.diskrverkl);
81    fscanf (fp, "%s %lf", str, &K.diskr.umgebung);
82    fscanf (fp, "%s %lf", str, &K.diskr.abbruch);
83    //Gitterparameter einlesen
84    fscanf (fp, "%s %lf", str, &K.gitter.lo[0]);
85    fscanf (fp, "%s %lf", str, &K.gitter.lo[1]);
86    fscanf (fp, "%s %lf", str, &K.gitter.hi[0]);
87    fscanf (fp, "%s %lf", str, &K.gitter.Delta[0]);
88    fscanf (fp, "%s %lf", str, &K.gitter.Delta[0]);
89    fscanf (fp, "%s %lf", str, &K.gitter.Delta[1]);
90    return K;
91  }
92
93  //--------------Kontrollausgabe bei Programmstart------------
94  void ausgabe(struct Konstanten *K)
95  {
96    printf("*****************************************\n");
97    printf("*\t\t\t\t\t\t\t\t\t*\n");
98    printf("*\tProgramm zur Loesung eines stochastischen
      Optimierungsproblems\t*\n*\t\t\tmit
      Dynamischer Programmierung\t\t\t*\n");
99    printf("*\t\t\t(C)\tMaik Mueller\t2009\t\t\t*\n");
100   printf("*\t\t\t\t\t\t\t\t\t*\n");
101   printf("*\t\traeumliche und zeitliche Diskretisierung auf
      Gittern\t*\n*\t\t\t mit Hilfe von gridgen V0.99.1\t\t\t*\n");
102   printf("*\t\t\t(C)\tLars Gruene\t2007\t\t\t*\n");
103   printf("*\t\t\t\t\t\t\t\t\t*\n");
104   printf("*****************************************\n");
105   printf("*\t\t\t\t\t\t\t\t\t*\n");
106   printf("*\t\tVerwendete Parameter:\t\t\t\t*
      \n*\t\t\t\t\t\t\t\t\t*\n");
107   printf("*\t\tklein_delta\t= %lf\t\t\t\t*\n",
      (*K).param.klein_delta);
108   printf("*\t\tgamma\t\t=%lf\t\t\t\t*\n",(*K).param.gamma_r);
109   printf("*\t\tu_max[0]\t=%lf\t\t\t\t*\n",(*K).param.u_max[0]);
110   printf("*\t\tu_max[1]\t=%lf\t\t\t\t*\n",(*K).param.u_max[1]);
111   printf("*\t\tu_min[0]\t=%lf\t\t\t\t*\n",(*K).param.u_min[0]);
112   printf("*\t\tu_min[1]\t=%lf\t\t\t\t*\n",(*K).param.u_min[1]);
113   printf("*\t\tx0[0]\t\t=%lf\t\t\t\t*\n",(*K).param.x0[0]);
114   printf("*\t\tx0[1]\t\t=%lf\t\t\t\t*\n",(*K).param.x0[1]);
115   printf("*\t\tx_max\t\t=%lf\t\t\t\t*\n",(*K).param.x_max);
```

```
116   printf("*\t\tepsilon\t\t=%lf\t\t\t\t*\n",(*K).diskr.epsilon);
117   printf("*\t\ttol\t\t=%lf\t\t\t\t*\n",(*K).diskr.tol);
118   printf("*\t\th\t\t=%lf\t\t\t\t*\n",(*K).diskr.h);
119   printf("*\t\tanfdiskr\t=%lf\t\t\t\t*\n",(*K).diskr.anfdiskr);
120   printf("*\t\tdiskrverkl\t=%lf\t\t*\n",(*K).diskr.diskrverkl);
121   printf("*\t\tumgebung\t=%lf\t\t\t\t*\n",(*K).diskr.umgebung);
122   printf("*\t\tabbruch\t\t=%lf\t\t\t\t*\n",(*K).diskr.abbruch);
123   printf("*\t\tlo[0]\t\t=%lf\t\t\t\t*\n",(*K).gitter.lo[0]);
124   printf("*\t\tlo[1]\t\t=%lf\t\t\t\t*\n",(*K).gitter.lo[1]);
125   printf("*\t\thi[0]\t\t=%lf\t\t\t\t*\n",(*K).gitter.hi[0]);
126   printf("*\t\thi[1]\t\t=%lf\t\t\t\t*\n",(*K).gitter.hi[1]);
127   printf("*\t\tDelta[0]\t=%lf\t\t\t\t*\n",(*K).gitter.Delta[0]);
128   printf("*\t\tDelta[1]\t=%lf\t\t\t\t*\n",(*K).gitter.Delta[1]);
129   printf("*\t\tsigma_EK\t=%lf\t\t\t\t*\n",(*K).param.sigma_EK);
130   printf("*\t\tsigma_FK\t=%lf\t\t\t\t*\n",(*K).param.sigma_FK);
131   printf("*\t\t\t\t\t\t\t\t\t*\n");
132   printf("****************************************\n\n");
133 }
134 //Zufallszahl zw. 0 und 1 erzeugen
135 double erzeuge_Zufallszahl()
136 {
137   double x;
138   x=rand();
139   return x=x/RAND_MAX;
140 }
141 //----------------------Wiener Prozess--------------------
142 double Wiener_Prozess(struct Konstanten *K)
143 {
144   int i,j,M=2;
145   double x,erg;
146   double W[M];
147 
148   for(j=0;j<M;j++)
149   {
150     x=erzeuge_Zufallszahl();
151     if(x<=0.5)
152     {
153       W[j]=-sqrt((*K).diskr.h);
154     }
155     else
156     {
157       W[j]=sqrt((*K).diskr.h);
```

```
158       }
159     }
160     erg=0.5*(W[1]+W[0]);
161     return erg;
162   }
163
164   //----------Power-Nutzenfunktion entspricht l(x,u)-----------
165
166   double l(double *x, double *u, struct Konstanten *K)
        //u(0) ... Konsum -> bringt den Nutzen l(x,u) mittels
          u(0)*x(0) (x(0)...Vermoegen)
167   {
168     if((*K).param.gamma_r==1) //ln(u)
169     {
170       if((u[0]*x[0])<(*K).diskr.epsilon)
171       {
172         return (-1000000);
173       }
174       else
175       {
176         return (log(u[0]*x[0]));
177       }
178     }
179     else if((*K).param.gamma_r>1)
      //hier haben wir x^(-...)/(-....), ist bei x=0 nicht definiert
180     {
181       if((u[0]*x[0])<(*K).diskr.epsilon)
182       {
183         return (-1000000);
184       }
185       else
186       {
187         return ( (pow((u[0]*x[0]),(1-(*K).param.gamma_r)) -1) /
              (1-(*K).param.gamma_r) );
188       }
189     }
190     else
191     {
192       return ( (pow((u[0]*x[0]),(1-(*K).param.gamma_r)) -1) /
            (1-(*K).param.gamma_r) );
193     }
194   }
```

```
195
196 //-----------------------Dynamik--------------------------
197 //Umformulierung bei Konsum:= u[0]*x[0], u[0] ist nur Anteil
    des Gesamtvermoegens am Konsum, daher hier nicht Konsum=u[0]
198
199 void f(qgrid *Gitter_rFK, qgrid *Gitter_rEK, double *x,
      double *u, double *erg, struct Konstanten *K)
200 {
201   double re,rf,y[1];
202   int flag;
203   y[0]=x[1]; //nur Zeit fuer Zinssatz relevant
204   //-----------------------------------------//
205   if(y[0]>(*K).gitter.hi[1])                  //
206   {    //nur zur Berechnung der optimalen Trajektorie noetig
207     y[0]=fmod(x[1],(*K).gitter.hi[1]);        //
208   }                                           //
209   //-----------------------------------------//
210   re=value(Gitter_rEK,y,&flag);              //EK-Zins
211   rf=value(Gitter_rFK,y,&flag);              //FK-Zins
212
213   erg[0] = u[1]*re*x[0]+(1-u[1])*rf*x[0]-u[0]*x[0];
      //Zinserträge - Konsum
214   erg[1] = 1; //Zeit t
215 }
216
217 //--------------stochastisches Euler-Verfahren-------------
218 //Setze (*K).param.sigma_EK=0 und (*K).param.sigma_FK=0
      fuer deterministisches Euler-Verfahren
219
220 void Stoch_Euler_Verfahren(qgrid *Gitter_rFK, qgrid
      *Gitter_rEK, double *x, double *u, double *z,
      double *erg, struct Konstanten *K)
221 {
222   int i;
223   double rs[dim];
224   f(Gitter_rFK, Gitter_rEK, x, u, rs, K);     //rechte Seite
225
226   erg[0]=x[0]+(*K).diskr.h*rs[0]+ (*K).param.sigma_EK* u[1]*
      x[0]*z[0] + (*K).param.sigma_FK*(1-u[1])*x[0]*z[1];
227   erg[1]=x[1]+(*K).diskr.h*rs[1];
228 }
229
```

```
230 //Berechnung der optimalen Trajektorien fuer x, u[0], u[1]
231
232 void Stoch_trajektorie(qgrid *Gitter_u0, qgrid *Gitter_u1,
    qgrid *Gitter_rFK, qgrid *Gitter_rEK, struct Konstanten *K)
233 {
234   double euler[dim],erg1[dim],erg2[dim];
235   double x[dim],u[dim],y[dim],z[2];
236   double v,diff_konsum;
237   int i,flag,zaehler=0;
238   FILE *fp;
239   char filename[100];
240
241   sprintf(filename,"Modell/traj_%lf_%lf.txt",
        (*K).param.x0[0],(*K).param.x0[1]);
242   fp = fopen(filename , "w");
243
244   v=0;
245   fprintf(fp,"%lf %lf %lf %lf %lf\n",
        (*K).diskr.h,(*K).param.x0[0],(*K).param.x0[1],v,v);
246   for(i=0;i<dim;i++)     //Startwert
247 {
248     x[i]=(*K).param.x0[i];
249   }
250
251   do
252   {
253     y[1]=fmod(x[1],(*K).gitter.hi[1]);
254     diff_konsum=0;
255     if(x[0]>(*K).gitter.hi[0])//Fall, wenn x[0]>hi[0]
256     {
257       diff_konsum=x[0]-(*K).gitter.hi[0];  //Vermoegen,
      welches ueber hi[0] liegt wird zusaetzlich konsumiert
258       x[0]=(*K).gitter.hi[0];
259     }
260     if(x[0]<(*K).gitter.lo[0])
261     {
262       x[0]=(*K).gitter.lo[0];
263     }
264     y[0]=x[0];
265     u[0]=value(Gitter_u0,y,&flag);
266     u[1]=value(Gitter_u1,y,&flag);
267
```

```
268     v=(*K).diskr.h * pow((1.-(*K).param.klein_delta*
        (*K).diskr.h),zaehler) * l(x,u,K)+v;
269     fprintf(fp,"%lf %lf %lf %lf %lf\n",x[0],x[1],
        u[0]*x[0]+diff_konsum,u[1],v);
270
271     //Loesungspfad
272     z[0]=Wiener_Prozess(K);
273     z[1]=Wiener_Prozess(K);
274     Stoch_Euler_Verfahren(Gitter_rFK,Gitter_rEK,x,u,z,euler,K);
275     for(i=0;i<dim;i++)
276     {
277       x[i]=euler[i];
278     }
279
280     zaehler=zaehler+1;
281   }while(x[1]<=(*K).param.x_max+(*K).diskr.epsilon);
282   fclose(fp);
283 }
284
285 //----------Maximum finden---------------------------------
286
287 void finde_max_besser(qgrid *Gitter_V, qgrid *Gitter_rFK,
      qgrid *Gitter_rEK, double *x, struct Konstanten *K,
      double *u, double *max_val)
288 {
289   double current_u[dim], erg1[dim], erg2[dim], erg3[dim],
      erg4[dim], u_max[dim], u_min[dim];
290   double temp,current_max1,z[2];
291   double beta=1.-(*K).param.klein_delta*(*K).diskr.h;
        //Diskontfaktor
292   double abstand[dim];     //derzeitige Diskretisierung
293
294   current_max1=-1.0/0.0;
295
296   int i,flag;
297   int abbruch=0;
298
299   for(i=0;i<dim;i++)
300   {
301     abstand[i]= (*K).diskr.anfdiskr*((*K).param.u_max[i]
          -(*K).param.u_min[i]);
302     u_min[i]=(*K).param.u_min[i];
```

```
303       u_max[i]=(*K).param.u_max[i];
304     }
305
306     do//1. "do"
307     {
308       current_u[0]=u_min[0]-abstand[0]; //aktuelles u[0]
              = untere Schranke
309       do//2."do"
310       {
311         current_u[0]=current_u[0]+abstand[0]; //aktuell
              betrachtetes u[0]
312         current_u[1]=u_min[1]-abstand[1]; //aktuelles u[1]
              = untere Schranke
313         //falls ein current_u groesser als das maximale u wird,
              dann wird es einfach auf dieses gesetzt
314         if(current_u[0]>u_max[0])
315         {
316           current_u[0]=u_max[0];
317         }
318         do//3."do"
319         {
320           current_u[1]=current_u[1]+abstand[1]; //aktuell
              betrachtetes u[1]
321         //falls ein current_u groesser als das maximale u wird,
              dann wird es einfach auf dieses gesetzt
322           if(current_u[1]>u_max[1])
323           {
324           current_u[1]=u_max[1];
325           }
326
327           //zu unterscheiden sind 3 Faelle beim Aufruf von
              Stoch_Euler_Verfahren:
328         //1.) (*K).param.sigma_EK=0 und (*K).param.sigma_FK=0
329         //2.) (*K).param.sigma_EK!=0 und (*K).param.sigma_FK=0
              oder (*K).param.sigma_EK=0 und (*K).param.sigma_FK!=0
330         //3.) (*K).param.sigma_EK!=0 und (*K).param.sigma_FK!=0
331
332           //Fall 1.)
333           if((*K).param.sigma_EK==0 && (*K).param.sigma_FK==0)
334           {
335             z[0]=sqrt((*K).diskr.h);
336             z[1]=sqrt((*K).diskr.h);
```

```
337           Stoch_Euler_Verfahren(Gitter_rFK, Gitter_rEK, x,
              current_u, z, erg1, K);
338           //Fehlerabfang
339           //Falls x[1]>obere Schranke fuer x[1], dann setze
              x[1]=mod(x[1],h[1])
340           if(erg1[1]>(*K).gitter.hi[1])
341           {
342             erg1[1]=fmod(erg1[1],(*K).gitter.hi[1]);
343           }
344 //Falls x[0]>obere Schranke fuer x[0] (bzw.<untere Schranke)
    bei einen oder beiden Fällen sqrt(h) und -sqrt(h),
345 //dann wird x[0] auf die obere bzw. untere Schranke gesetzt
346           if(erg1[0]>(*K).gitter.hi[0])
347           {
348             erg1[0]=(*K).gitter.hi[0];
349           }
350             if(erg1[0]<(*K).gitter.lo[0])
351           {
352             erg1[0]=(*K).gitter.lo[0];
353           }
354           //Berechnung der optimalen Wertefunktion
355           temp=(*K).diskr.h*l(x,current_u,K)+beta*1*
              ( value(Gitter_V,erg1,&flag) );
356         }
357         //Fall 2.)
358     else if( ((*K).param.sigma_EK!=0 && (*K).param.sigma_FK==0)
            || ((*K).param.sigma_EK==0 && (*K).param.sigma_FK!=0) )
359         {
360       //Fall sqrt(h) bei z[0] und -sqrt(h) bei z[1] berechnen
361           z[0]=sqrt((*K).diskr.h);
362           z[1]=-sqrt((*K).diskr.h);
363           Stoch_Euler_Verfahren(Gitter_rFK, Gitter_rEK,
              x, current_u, z, erg1, K);
364       //Fall -sqrt(h) bei z[0] und sqrt(h) bei z[1] berechnen
365           z[0]=-sqrt((*K).diskr.h);
366           z[1]=sqrt((*K).diskr.h);
367           Stoch_Euler_Verfahren(Gitter_rFK, Gitter_rEK,
              x, current_u, z, erg2, K);
368           //Fehlerabfang
369           //Falls x[1]>obere Schranke fuer x[1],
              dann setze x[1]=mod(x[1],h[1])
370           if(erg1[1]>(*K).gitter.hi[1])
```

```
371           {
372             erg1[1]=fmod(erg1[1],(*K).gitter.hi[1]);
373           }
374           if(erg2[1]>(*K).gitter.hi[1])
375           {
376             erg2[1]=fmod(erg2[1],(*K).gitter.hi[1]);
377           }
378 //Falls x[0]>obere Schranke fuer x[0] (bzw. < untere
    Schranke) bei einen oder beiden Fällen sqrt(h) und -sqrt(h),
379 //dann wird x[0] auf die obere bzw. untere Schranke gesetzt
380           if(erg1[0]>(*K).gitter.hi[0])
381           {
382             erg1[0]=(*K).gitter.hi[0];
383           }
384           if(erg1[0]<(*K).gitter.lo[0])
385           {
386             erg1[0]=(*K).gitter.lo[0];
387           }
388           if(erg2[0]>(*K).gitter.hi[0])
389           {
390             erg2[0]=(*K).gitter.hi[0];
391           }
392           if(erg2[0]<(*K).gitter.lo[0])
393           {
394             erg2[0]=(*K).gitter.lo[0];
395           }
396           //Berechnung der optimalen Wertefunktion als
              Erwartungswert der 2 Faelle sqrt(h) und -sqrt(h)
397           temp=(*K).diskr.h*l(x,current_u,K)+beta*0.5*
              ( value(Gitter_V,erg1,&flag) +
              value(Gitter_V,erg2,&flag) );
398         }
399         //Fall 3.)
400     else if((*K).param.sigma_EK!=0 && (*K).param.sigma_FK!=0)
401         {
402     //Fall sqrt(h) bei z[0] und sqrt(h) bei z[1] berechne
403           z[0]=sqrt((*K).diskr.h);
404           z[1]=sqrt((*K).diskr.h);
405           Stoch_Euler_Verfahren(Gitter_rFK, Gitter_rEK, x,
              current_u, z, erg1, K);
406     //Fall -sqrt(h) bei z[0] und sqrt(h) bei z[1] berechne
407           z[0]=-sqrt((*K).diskr.h);
```

```
408             z[1]=sqrt((*K).diskr.h);
409             Stoch_Euler_Verfahren(Gitter_rFK, Gitter_rEK, x,
                current_u, z, erg2, K);
410         //Fall sqrt(h) bei z[0] und -sqrt(h) bei z[1] berechne
411             z[0]=sqrt((*K).diskr.h);
412             z[1]=-sqrt((*K).diskr.h);
413             Stoch_Euler_Verfahren(Gitter_rFK, Gitter_rEK, x,
                current_u, z, erg3, K);
414         //Fall -sqrt(h) bei z[0] und -sqrt(h) bei z[1] berechne
415             z[0]=-sqrt((*K).diskr.h);
416             z[1]=-sqrt((*K).diskr.h);
417             Stoch_Euler_Verfahren(Gitter_rFK, Gitter_rEK, x,
                current_u, z, erg4, K);
418             //Fehlerabfang
419             //Falls x[1]>obere Schranke fuer x[1],
                dann setze x[1]=mod(x[1],h[1])
420             if(erg1[1]>(*K).gitter.hi[1])
421             {
422               erg1[1]=fmod(erg1[1],(*K).gitter.hi[1]);
423             }
424             if(erg2[1]>(*K).gitter.hi[1])
425             {
426               erg2[1]=fmod(erg2[1],(*K).gitter.hi[1]);
427             }
428             if(erg3[1]>(*K).gitter.hi[1])
429             {
430               erg3[1]=fmod(erg3[1],(*K).gitter.hi[1]);
431             }
432             if(erg4[1]>(*K).gitter.hi[1])
433             {
434               erg4[1]=fmod(erg4[1],(*K).gitter.hi[1]);
435             }
436 //Falls x[0]>obere Schranke fuer x[0] (bzw. < untere
    Schranke) bei einen oder beiden Fällen sqrt(h) und -sqrt(h),
437 //dann wird x[0] auf die obere bzw. untere Schranke gesetzt
438             if(erg1[0]>(*K).gitter.hi[0])
439             {
440               erg1[0]=(*K).gitter.hi[0];
441             }
442             if(erg1[0]<(*K).gitter.lo[0])
443             {
444               erg1[0]=(*K).gitter.lo[0];
```

```
445             }
446             if(erg2[0]>(*K).gitter.hi[0])
447             {
448               erg2[0]=(*K).gitter.hi[0];
449             }
450             if(erg2[0]<(*K).gitter.lo[0])
451             {
452               erg2[0]=(*K).gitter.lo[0];
453             }
454             if(erg3[0]>(*K).gitter.hi[0])
455             {
456               erg3[0]=(*K).gitter.hi[0];
457             }
458             if(erg3[0]<(*K).gitter.lo[0])
459             {
460               erg3[0]=(*K).gitter.lo[0];
461             }
462             if(erg4[0]>(*K).gitter.hi[0])
463             {
464               erg4[0]=(*K).gitter.hi[0];
465             }
466             if(erg4[0]<(*K).gitter.lo[0])
467             {
468               erg4[0]=(*K).gitter.lo[0];
469             }
470             //Berechnung der optimalen Wertefunktion
                als Erwartungswert der 4 Faelle
471             temp=(*K).diskr.h*l(x,current_u,K)+beta*0.25*
                ( value(Gitter_V,erg1,&flag) +
                value(Gitter_V,erg2,&flag)+
                value(Gitter_V,erg3,&flag)+
                value(Gitter_V,erg4,&flag) );
472           }
473
474           //Test ob aktuell berechneter Wert optimal ist
475           if(temp>current_max1)
476           {
477             for(i=0;i<dim;i++)
478             {
479               u[i]=current_u[i]; //momentan optimales u
480             }
481             current_max1=temp;
```

```
482           }
483         }while(current_u[1]<u_max[1]); //Ende 3. "do"
484       }while(current_u[0]<u_max[0]); //Ende 2. "do"
485       if(abstand[0]<(*K).diskr.abbruch &&
          abstand[1]<(*K).diskr.abbruch)
486       {
487         abbruch=1;
488       }
489       //abstand[i] verkleineren um in Umgebung vom
          derzeitigen Optimimum eine genauere Loesung zu finden
490       for(i=0;i<dim;i++)
491       {
492         abstand[i]=(*K).diskr.diskrverkl*abstand[i];
493         u_max[i]=u[i]+(*K).diskr.umgebung*abstand[i];
494         u_min[i]=u[i]-(*K).diskr.umgebung*abstand[i];
495         if(u_min[i]<(*K).param.u_min[i])
496         {
497           u_min[i]=(*K).param.u_min[i];
498         }
499         if(u_max[i]>(*K).param.u_max[i])
500         {
501           u_max[i]=(*K).param.u_max[i];
502         }
503       }
504     }while(abbruch!=1); //Ende 1. "do"
505     *max_val=current_max1;
506   }
507
508   //-----------------Werte-Iteration-------------------------
509
510   void werte_iteration(qgrid *Gitter_V, qgrid *Gitter_V_temp,
        qgrid *Gitter_u0, qgrid *Gitter_u1, qgrid *Gitter_rFK,
        qgrid *Gitter_rEK, struct Konstanten *K, float *geszeit,
        int *iterationen)
511   {
512     int abbruch=0;    //fuer Abbruchkriterium
513     int it_zaehler=0;  //Iterationszaehler
514     double maximum,max_val;
515     int index1,index2,index3,flag,saveok;
516     double temp2,temp3;
517     float zeit=0;
518     *geszeit=0;
```

```
519
520   double x[dim], u[dim], erg[dim];
521
522   FILE *fp;
523   fp=fopen("Modell/Konvergenz.txt","w");
524   char filename[100];
525
526   //Start Werteiteration
527   //Gitter_V_temp ist das aktuelle Gitter der optimalen
        Wertefunktion, welches beschrieben wird
528   //Gitter_V ist das Gitter der optimalen Wertefunktion
        aus der vorherigen Iteration
529   do
530   {
531     tic(Gitter_V_temp);  //Start Zeitmessung
532     maximum=-100;
533
534     index1=first_node(Gitter_V_temp,x);
535     index2=first_node(Gitter_u0,x);
536     index3=first_node(Gitter_u1,x);
537     if(it_zaehler==0)  //Startiteration
538     {
539       do
540       {
541         set_current_nodevalue(Gitter_V_temp,0);
542         index1=next_node(Gitter_V_temp,x);
543         index2=next_node(Gitter_u0,x);
544         index3=next_node(Gitter_u1,x);
545       }while(index1!=-1);
546       maximum=1;
547     }
548     else   //Iterationen 1,2,...
549     {
550       sprintf(filename,"Modell/1/Wertefkt_gitter%d.val",
            it_zaehler-1);
551       Gitter_V=load_val(filename);
552       //Wenn Gitter der Wertefunktion nicht geladen
            werden kann => Abbruch
553       if( Gitter_V==NULL )
554       {
555         printf("Fehler beim Laden des Gitters der
              Wertefunktion!");
```

```
556           exit(1);
557         }
558         do
559         {
560           //Optimale Kontrolle bestimmen
561           finde_max_besser(Gitter_V,Gitter_rFK,Gitter_rEK,
                x,K,u,&max_val);
562           //Werte in Gitter besetzen
563           set_current_nodevalue(Gitter_V_temp,max_val);
564           set_current_nodevalue(Gitter_u0,u[0]);
565           set_current_nodevalue(Gitter_u1,u[1]);
566
567           //Abbruchkriterium
568           temp3=value(Gitter_V,x,&flag);
569           temp2=fabs(max_val-temp3);
570           //groesste Abweichung in V berechnen
571           if(temp2>maximum)
572           {
573             maximum=temp2;
574           }
575
576           //Index auf naechsten Knoten setzen
577           index1=next_node(Gitter_V_temp,x);
578           index2=next_node(Gitter_u0,x);
579           index3=next_node(Gitter_u1,x);
580         }while(index1!=-1);
581       }//Ende if(it_zaehler==0)
582
583       //Abbruch?
584       if(maximum<(*K).diskr.tol)
585       {
586         abbruch=1;
587       }
588
589   //altes val-Gitter von optimaler Wertefunktion loeschen
590       //es wird nur immer das jeweils letzte Gitter benoetigt
591       if(it_zaehler>0)
592       {
593         sprintf(filename,"Modell/1/Wertefkt_gitter%d.val",
              it_zaehler-1);
594         remove(filename);
595       }
```

```
596
597 //aktuelles Gitter fuer optimale Wertefunktion abspeichern
598     sprintf(filename,"Modell/1/Wertefkt_gitter%d.val",
          it_zaehler);
599     saveok=save_val(Gitter_V_temp,filename);
600 //Wenn Gitter der Wertefunktion nicht gspeichert werden
      kann => Abbruch
601     if(saveok==1)
602     {
603       printf("Fehler beim Speichern des Gitters der
            Wertefunktion!");
604       exit(1);
605     }
606
607     if(abbruch==1)//Diese Gitter werden nur nach der
          letzten Iteration gespeichert (weil nur da benoetigt)
608     {
609       sprintf(filename,"Modell/2/Wertefkt_gitter%d.asc",
            it_zaehler);
610       saveok=export_gridandval(Gitter_V_temp,filename);
611       if(saveok==1)
612       {
613         printf("Fehler beim Speichern des Gitters der
            Wertefunktion fuer Matlab!");
614         exit(1);
615       }
616     //aktuelles Gitter fuer optimale Kontrollen abspeichern
617       sprintf(filename,"Modell/1/u1_gitter%d.val",it_zaehler);
618       save_val(Gitter_u1,filename);
619       sprintf(filename,"Modell/2/u1_gitter%d.asc",it_zaehler);
620       export_gridandval(Gitter_u1,filename);
621
622       sprintf(filename,"Modell/1/u0_gitter%d.val",it_zaehler);
623       save_val(Gitter_u0,filename);
624       sprintf(filename,"Modell/2/u0_gitter%d.asc",it_zaehler);
625       export_gridandval(Gitter_u0,filename);
626     }
627
628     zeit=toc(Gitter_V_temp); //Ende Zeitmessung fuer
          aktuelle Iteration
629     *geszeit=*geszeit+zeit; //Gesamtzeit bis
          aktueller Iteration
```

```
630     printf("Iteration %d Zeit=%4.2fs Ges.Zeit=%4.2fs\n",
          it_zaehler,zeit,*geszeit);
631     if(it_zaehler>0)
632     {
633       printf("Maximale Abweichung in V: %4.10f, Abbruch?
            %d\n",maximum,abbruch);
634       fprintf(fp,"%d %lf\n",it_zaehler,maximum);
635     }
636     //-----------------------------------
637
638     it_zaehler++;
639
640     delete_grid(Gitter_V);
641   }while(abbruch!=1);
642   *iterationen=it_zaehler-1; //Anzahl Iterationen bis Abbruch
643   printf("Ende! Gesamtzeit=%4.2fs\n",*geszeit);
644   fclose(fp);
645 }
646
647 //-------Statistiken nach Abschluss der Werteiteration------
648
649 void statistiken(struct Konstanten *K, float geszeit, int iter)
650 {
651   FILE *fp;
652   fp = fopen( "Modell/Zusammenfassung.txt", "w");
653
654   fprintf(fp,"Verwendete Konstanten:\n");
655   fprintf(fp,"Dimension des Problems:\n dim\t\t=\t%d\n",dim);
656   fprintf(fp,"Verwendete Toleranz bei Abbruchkriterium
      ||V_t-1-V_t||< tol:\ntol\t\t=\t%4.6f\n", (*K).diskr.tol);
657   fprintf(fp,"Relative Risikoaversion:\n gamma\t=\t%4.6f\n",
      (*K).param.gamma_r);
658   fprintf(fp,"Parameter fuer den stochastischen Einfluss auf
      die Renditen:\n ");
659   fprintf(fp,"sigma_EK\t=\t%4.6f\n ", (*K).param.sigma_EK);
660   fprintf(fp,"sigma_FK\t=\t%4.6f\n", (*K).param.sigma_FK);
661   fprintf(fp,"Abzinsungsfaktor (beta=1-delta): \n
      delta\t=\t%4.6f\n", (*K).param.klein_delta);
662   fprintf(fp,"Schrittweite\n h\t\t=\t%4.6f\n",(*K).diskr.h);
663   fprintf(fp,"Kontrollwertebereich\n U\t\t=\t[%4.6f\t,%4.6f\t]
      x[%4.6f\t, %4.6f\t]\n", (*K).param.u_min[0],
      (*K).param.u_max[0],(*K).param.u_min[1],(*K).param.u_max[1]);
```

```
664   fprintf(fp,"Verwendete Parameter fuer den Algorithmus
        finde_max_besser:\n");
665   fprintf(fp,"Anfangsdiskretisierung\n anfdiskr\t\t=
        \t%4.6f\n", (*K).diskr.anfdiskr);
666   fprintf(fp,"Verfeinerung der Diskretisierung\n diskrverkl
        \t\t= \t%4.6f\n", (*K).diskr.diskrverkl);
667   fprintf(fp,"Umgebung vom Optimum der aktuellen
        Diskretisierung\n umgebung\t\t= \t%4.6f\n",
        (*K).diskr.umgebung);
668   fprintf(fp,"Gewuenschte Genauigkeit der Kontrollen (Stelle
        nach dem Komma)\n abbruch\t\t= \t%4.6f\n",
        (*K).diskr.abbruch);
669   fprintf(fp,"Verwendete Gitterparameter:\n");
670   fprintf(fp,"Untere Schranken: \n lo\t\t=\t[%4.6f\t,
        %4.6f\t]\n",(*K).gitter.lo[0],(*K).gitter.lo[1]);
671   fprintf(fp,"Obere Schranken:\n hi\t\t=\t[%4.6f\t, %4.6f\t]
        \n",(*K).gitter.hi[0],(*K).gitter.hi[1]);
672   fprintf(fp,"Groesse der Gitterrechtecke:\n Delta
        \t=\t[%4.6f\t,%4.6f\t]\n\n",
        (*K).gitter.Delta[0], (*K).gitter.Delta[1]);
673   fprintf(fp,"Benoetigte Zeit fuer die Werteiteration:\n
        Zeit\t=\t%4.2fs\n", geszeit);
674   fprintf(fp,"Iterationen: %d\n", iter);
675
676   fclose(fp);
677 }
678
679 //------------------------------------------------------------
680 //-------------------------Hauptprogramm----------------------
681 //------------------------------------------------------------
682
683 int main(int args, char ** argv)
684 {
685   if(args != 2)
686   {
687     printf("Aufruf: %s Parameterdatei.dat!",argv[0]);
688     exit(1);
689   }
690   float geszeit;
691   int iter, zahl1, zahl2, zinszahl;
692   char filename[100];
693   struct Konstanten K;
```

```
694   FILE *fp;
695
696   //Parameter einlesen
697   fp = fopen( argv[1], "r");
698   if(NULL == fp)
699   {
700     fprintf(stderr, "Fehler beim Oeffnen der
        Parameterdatei\n");
701     return EXIT_FAILURE;
702   }
703   K=eingabe(fp);
704   fclose(fp);
705   //Parameter zur Kontrolle ausgeben
706   ausgabe(&K);
707
708   //qgrid-Struktur fuer die Gitter
709   qgrid *Gitter_V, *Gitter_V_temp; //Gitter fuer die
        optimale Wertefunktion
710   qgrid *Gitter_u0; //Gitter fuer den optimalen Konsum
711   qgrid *Gitter_u1; //Gitter fuer die optimale Kontrolle
712   qgrid *Gitter_rFK,*Gitter_rEK; //Gitter fuer Zinssaetze
        r anlegen
713
714   //sorgt dafuer, dass bei jedem Programmaufruf eine
        neue Zufallsfolge erzeugt wird
715   srand(time(0));
716
717   //Zinsverlaeufe muessen im Ordner "Zinsen" abgelegt werden
718   sprintf(filename,"Zinsen/rEK_gitter.val");
719   Gitter_rEK=load_val(filename);
720   sprintf(filename,"Zinsen/rFK_gitter.val");
721   Gitter_rFK=load_val(filename);
722   //Fehlerabfang, wenn Gitter der Zinsen nicht geoeffnet
        werden koennen
723   if(Gitter_rEK==NULL || Gitter_rFK==NULL)
724   {
725     fprintf(stderr, "Fehler beim Oeffnen der Gitter fuer
        die Zinsen\n");
726     return EXIT_FAILURE;
727   }
728
```

```
729   printf("Soll die optimale Werteiteration berechnet werden?
        ja=(1), nein=(2)\n");
730   scanf("%d", &zahl1);
731   if(zahl1==1)
732   {
733   //--------------------Gitter erzeugen-----------------
734     Gitter_V=create_grid(K.gitter.lo,K.gitter.hi,
        K.gitter.Delta,dim,0);
735     Gitter_V_temp=create_grid(K.gitter.lo,K.gitter.hi,
        K.gitter.Delta,dim,0);
736     Gitter_u0=create_grid(K.gitter.lo,K.gitter.hi,
        K.gitter.Delta,dim,0);
737     Gitter_u1=create_grid(K.gitter.lo,K.gitter.hi,
        K.gitter.Delta,dim,0);
738
739     werte_iteration(Gitter_V, Gitter_V_temp, Gitter_u0,
        Gitter_u1, Gitter_rFK, Gitter_rEK, &K, &geszeit, &iter);
        //Werteiteration
740     statistiken(&K, geszeit, iter); //Statistiken fuer
        besseren Ueberblick ueber die Konstanten, Laufzeit,
        Iterationen
741   }
742   if(zahl1!=1 && zahl1!=2)
743   {
744     printf("Falsche Eingabe");
745     exit(1);
746   }
747   printf("Soll die optimale Tajektorie berechnet werden?
        ja=(1), nein=(2)\n");
748   scanf("%d", &zahl2);
749   if(zahl2==1)
750   {
751     if(zahl1==2)
752     {
753     printf("Nummer des Gitters fuer die optimalen
        Trajektorien eingeben\n");
754     scanf("%d", &iter);
755     }
756     //---------optimale Trajektorien berechnen--------
757     sprintf(filename,"Modell/1/u0_gitter%d.val",iter);
758     Gitter_u0=load_val(filename);
759     sprintf(filename,"Modell/1/u1_gitter%d.val",iter);
```

```
760     Gitter_u1=load_val(filename);
761     //Fehlerabfang, wenn Gitter der optimalen Kontrollen
          nicht geoeffnet werden koennen
762     if(Gitter_u0==NULL || Gitter_u1==NULL)
763     {
764       fprintf(stderr, "Fehler beim Oeffnen der Gitter der
            optimalen Kontrollen bei Berechnung der optimalen
            Trajektorien\n");
765       return EXIT_FAILURE;
766     }
767     printf("Berechnung der optimalen Trajektorien fuer
          x0[0]=%lf,\t x0[1]=%lf\n",K.param.x0[0],K.param.x0[1]);
768     Stoch_trajektorie(Gitter_u0,Gitter_u1, Gitter_rFK,
          Gitter_rEK,&K);
769
770     K.param.x0[0]=K.gitter.hi[0];
771     printf("Berechnung der optimalen Trajektorien fuer
          x0[0]=%lf,\t x0[1]=0\n",K.param.x0[0]);
772     Stoch_trajektorie(Gitter_u0,Gitter_u1, Gitter_rFK,
          Gitter_rEK,&K);
773     K.param.x0[0]=0.5*K.gitter.hi[0];
774     printf("Berechnung der optimalen Trajektorien fuer
          x0[0]=%lf,\t x0[1]=0\n",K.param.x0[0]);
775     Stoch_trajektorie(Gitter_u0,Gitter_u1, Gitter_rFK,
          Gitter_rEK,&K);
776     do
777     {
778       printf("Sollen weitere optimale Tajektorien berechnet
            werden? ja=(1), nein=(2)\n");
779       scanf("%d", &zahl1);
780       if(zahl1==1)
781       {
782         printf("Startvermoegen: x0[0]<%lf eingeben!\n",
              K.gitter.hi[0]);
783         scanf("%lf", &K.param.x0[0]);
784         printf("Startzeit: x0[1]<%d eingeben!\n",
              K.param.x_max);
785         scanf("%lf", &K.param.x0[1]);
786         Stoch_trajektorie(Gitter_u0,Gitter_u1, Gitter_rFK,
              Gitter_rEK,&K);
787       }
788     }while(zahl1==1);
```

```
789     }
790     if(zahl2!=1 && zahl2!=2)
791     {
792       printf("Falsche Eingabe");
793       exit(1);
794     }
795     delete_grid(Gitter_rEK);
796     delete_grid(Gitter_rFK);
797   }
```

Anhang B

Notationen

An dieser Stelle des Anhangs werden die wichtigsten Variablenbezeichnungen und Notationen, geordnet nach ihrem erstmaligen Auftreten (kapitelweise), aufgeführt.

Kapitel 2:

RRA	Abkürzung für relative Risikoaversion
γ	mathematisches Symbol für die relative Risikoaversion
$g^{'}$	1. Ableitung von g
$g^{''}$	2. Ableitung von g
$\mathbb{E}$	Erwartungswert
σ	Standardabweichung
ρ	Korrelation
r_i	Rendite/Zins von Anlage i
EF	Abkürzung für Efficient Frontier
CAPM	Abkürzung für Capital Asset Pricing Modell
CML	Abkürzung für Capital Market Line
SML	Abkürzung für Security Market Line

Kapitel 3:

Ω	Menge von Elementarereignissen
Σ	σ-Algebra auf Ω
P	Wahrscheinlichkeitsmaß auf Σ
$\mathcal{B}$	Borel-σ-Algegra
$\mathbb{R}$	reelle Zahlen
$\mathcal{F}$	Filtration
x	Zustand
u	Kontrolle
z	stochastischer Einfluss

β	Diskontfaktor
l	laufende Kosten in J
J_∞	Zielfunktional des stochastischen Kontrollproblems auf unendlichem Zeithorizont
V_∞	optimale Wertefunktion auf unendlichem Zeithorizont
h	Schrittweite
$\mathcal{W}$	Funktionenraum
$\widetilde{V}_\infty$	numerische Approximation von V_∞
π	Projektionsoperator
$T(W)$	Operator für die rechte Seite des Optimalitätsprinzip
Γ	Gitter

Kapitel 4:

$X(t)$	Zustand zur Zeit t
$u(t)$	Kontrolle zur Zeit t
$z(t)$	stochastischer Einfluss zur Zeit t
$\mathcal{N}(\mu, \sigma^2)$	Normalverteilung mit Erwartungswert μ und Standardabweichung σ, auch bezeichnet mit (μ, σ^2)-normalverteilt
$\widetilde{W}(\cdot, \omega)$	approximierter Pfad des Wiener Prozesses

Anhang C

Grundbegriffe aus der Portfoliotheorie

Allokation: Bezeichnung für den durch Verfügbarkeit und Preis gesteuerten Verteilungsprozess von Gütern und deren Produktionsfaktoren.

Arbitrage: sind Geschäfte, die Preis-, Zins- oder Kursunterschiede, die zu einem bestimmten Zeitpunkt an verschiedenen Märkten bestehen, ausnutzen, um Gewinne zu erzielen oder Verluste zu vermeiden.

Coupon: Zins- oder Dividentenscheine, die festverzinslichen Wertpapieren oder Aktien beigefügt sind.

Courtage: Gebühr, die von Maklern berechnet werden darf.

Derivate: ist ein Produkt, dessen Preis vom Preis anderer Produkte abhängt oder davon abgeleitet wird.

Dividente: ist der auf den Nennwert einer Aktie entfallende Anteil am jährlichen Reingewinn eines Unternehmens.

Emission: Ausgabe und Vertrieb von Aktien und anderen Wertpapieren.

Emittent: Herausgeber der Aktien oder der anderen Wertpapiere.

Factoring: ist ein Finanzierungssystem, bei dem ein Finanzinstitut die Forderung eines Unternehmens aus dem Verkauf von Waren ankauft und das Risiko für den Ausfall der Forderung übernimmt.

Fonds: Geld- oder Vermögensvorratsstelle für bestimmte Zwecke.

Forfaitierung: bezeichnet den Ankauf von Forderungen unter Verzicht auf einen Rückgriff gegen den Verkäufer bei Zahlungsausfall (echte Forfaitierung). Allerdings haftet der Verkäufer für den Rechtsbestand der Forderung. Bei der unechten Forfaitierung ist ein Rückgriff dagegen nicht ausgeschlossen.

Futures: (unbedingtes Termingeschäft) Bezeichnung für Terminkontrakte.

Leerverkauf: Bezeichnung für den Verkauf von Wertpapieren, die der Verkäufer noch nicht besitzt. Der Verkäufer leiht sich die Papiere und gibt sie an den Käufer zu einem bestimmten Preis weiter. Zu einem vereinbarten Zeitpunkt hat der Verkäufer die ausgeliehenen Wertpapiere an den Verleiher zurückzugeben und muss sie vom Markt kaufen.

Obligation: gleich bedeutend wie festverzinsliche Wertpapiere.

Option: (bedingtes Termingeschäft) zeitlich begrenztes Recht, das der Käufer der Option nach freiem Belieben ausüben kann. Der Verkäufer verfügt über kein Wahlrecht. Er hat seiner Liefer- (Call-Option) bzw. Übernahmeverpflichtung (Put-Option) zur vereinbarten Menge, Preis sowie Zeitpunkt oder -raum nachzukommen.

Portfolio: Teil oder Gesamtheit der Anlage in Wertpapieren, die ein Kunde oder ein Unternehmen besitzt (Wertpapierbestand). Es dient primär dem Zweck der Risikostreuung (Risikodiversifikation). Dabei gibt es für jeden Investor ein sogenanntes optimales Portfolio, welches das Risiko-Chancen-Profil bestmöglich abbildet.

Swap: Tauschgeschäft in Devisen bzw. Währungen (Devisenswap), Zinsen (Zinsswap) sowie Kombinationen beider Arten. Dabei handelt es sich um eine Vereinbarung zwischen zwei Vertragspartnern, in der Zukunft Zahlungsströme auszutauschen. Die Vereinbarung definiert dabei, wie die Zahlungen berechnet werden und wann sie fließen.

Venture Capital: (Risiko-, Wagniskapital) Die befristete Bereitstellung von haftendem Kapital an ein kapitalnehmendes junges Unternehmen.

Zero Bonds: (Nullcoupon-Anleihe) sind Anleihen, bei denen die Zinszahlungen nicht in regelmäßigen Abständen erfolgen, sondern mit einem um einen Zinsabschlag verminderten Ausgabekurs ausgegeben oder um einen aufgezinsten Rücknahmekurs fällig werden.

Anhang D

Inhalt der beiliegenden CD

Die beiliegende CD enthält:

1. Die *Diplomarbeit als pdf-file* inklusive der Bilder, welche in der Arbeit eingefügt wurden, in dem Unterordner *Bilder*, geordnet nach Kapiteln
2. Die Ergebnisse aus Kapitel 6 im Ordner *Ergebnisse*, enthält die Ordner:
   (a) *Schweizer Aktienindex* enthält die Ordner:
      i. *deterministisch*: Ergebnisse aus Abschnitt 6.3.2
      ii. *mit stochastischem Einfluss*: Ergebnisse aus Abschnitt 6.4.2
      iii. *Test des Optimierungsalgorithmus*: Ergebnisse aus Abschnitt 6.2.1
      iv. *Testhi[0]*: Ergebnisse aus Abschnitt 6.2.2
      v. *Zinsen*: Zinsdaten des Schweizer Aktien- und Obligationenindex
   (b) *Standard and Poors* enthält die Ordner:
      i. *deterministisch*: Ergebnisse aus Abschnitt 6.3.1
      ii. *mit stochastischem Einfluss*: Ergebnisse aus Abschnitt 6.4.1
      iii. *Zinsen*: Zinsdaten des Standard&Poor's 500 Index und des 10-Jahres-Wertpapiers
   (c) *zufällige Zinsverläufe* enthält den Ordner:
      i. *deterministisch*: Ergebnisse aus Abschnitt 6.3.3
3. Die verwendeten und in Kapitel 5 vorgestellten Programme im Ordner *Programme*:
   (a) Im Ordner *Hauptprogramm* sind *Portfolio.c* und *gridgen.c* inklusive derer Header-Dateien sowie eine Eingabedatei *eingabe.txt* enthalten.
   (b) Im Ordner *Kapitel 2* sind die in Abschnitt 2.3.1 benutzten Programme gespeichert.
   (c) Im Ordner *Weitere Programme* sind in eigenen Unterordnern *Binde_in_Gitter.c*, *Zinserzeugung.c* und die *Matlab*-Programme enthalten.

Literaturverzeichnis

[1] Christoph Auckenthaler, *Mathematische Grundlagen des modernen Portfolio-Managements*, 3. Auflage, Verlag Paul Haupt Bern Stuttgart Wien, 2001

[2] Christoph Auckenthaler, *Theorie und Praxis des modernen Portfolio-Managements*, 2. Auflage, Verlag Paul Haupt Bern Stuttgart Wien, 1994

[3] Fabio Camilli und Maurizio Falcone, *An Approximation Scheme for the Optimal Control of Diffusion Processes*, Mathematical Modelling and Numerical Analysis, Vol. 29, n° 1, 1995, S.97-122

[4] Lars Grüne, *Numerische Dynamik von Kontrollsystemen*, Vorlesungsskript, Universität Bayreuth, Sommersemester 2004

[5] Lars Grüne, *Numerische Methoden für gewöhnliche Differentialgleichungen*, 3. Auflage, Vorlesungsskript, Universität Bayreuth, Sommersemester 2008

[6] Lars Grüne, *Stochastische Dynamische Optimierung*, Vorlesungsskript, Universität Bayreuth, Sommersemester 2007

[7] Lars Grüne, Caroline Öhrlein und Willi Semmler, *Dynamic Consumption and Portfolio Decisions with Time Varying Asset Returns*, Januar 2007

[8] M.D. Jöhnk, *Erzeugen und Testen von Zufallszahlen*, Heft 6, Reihe: Berichte aus dem Institut für Statistik und Versicherungsmathematik und aus dem Institut für Angewandte Statistik der Freien Universität Berlin, Physica-Verlag Würzburg, 1969

[9] Ralf Korn, Elke Korn, *Optionsbewertung und Portfolio-Optimierung*, 2. Auflage, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 2001

[10] Jürgen Kremer, *Einführung in die Diskrete Finanzmathematik*, Springer-Verlag Berlin Heidelberg, 2006

[11] Antonio Lasić, *Parameter Calibration und Simulation von Pfaden für das stochastic-volatility jump diffusion Optionsbewertungsmodell von Bakshi*, Diplomarbeit an der Johann Wolfgang Goethe-Universität Frankfurt am Main,

2002, *http://www.math.uni-frankfurt.de/~numerik/diplom/lasic.pdf*, letzter Abruf: 28.03.2009

[12] Harry M. Markowitz, *Mean-variance analysis in portfolio choice and capital markets*, Blackwell Publishers Oxford, 1987

[13] David Meintrup, Stefan Schäffler, *Stochastik - Theorie und Anwendung*, Springer-Verlag Berlin Heidelberg, 2005

[14] Caroline Öhrlein, *Portfolio-Optimierung mit dynamischer Programmierung*, Diplomarbeit an der Fakultät Mathematik und Physik der Universität Bayreuth 2006, betreut von Prof. Dr. Lars Grüne,

[15] Marcus Porembski, *Financial Optimization*, Vorlesungsskript, Philipps-Universität Marburg, Wintersemester 2005/2006, *http://www.mathematik.uni-marburg.de/Math-Net/v05w/FinOpt.htm*, letzter Abruf: 28.03.2009

[16] Werner Rittershofer, *Wirtschaftlexikon*, 2. Auflage, Deutscher Taschenbuchverlag München, 2002

Erklärung

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

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

Bayreuth, den 30. März 2009 .....

Maik Müller