Gasspeicherbewertung: Strukturelle Analyse und Algorithmen

Zur Erlangung des akademischen Grades eines

DOKTORS DER NATURWISSENSCHAFTEN
(Dr. rer. nat.)

an der Fakultät für Mathematik
des Karlsruher Instituts für Technologie (KIT)
genehmigte

DISSERTATION

von

Dipl.-Math. Viola Sonja Clara Riess
aus Karlsruhe

Tag der mündlichen Prüfung:	21.05.2014
Referentin:	Prof. Dr. Nicole Bäuerle
Korreferent:	Prof. Dr. Alfred Müller

Für meine Eltern.

Danksagung

Nach „lieben“ ist „helfen“ das schönste Zeitwort, der Welt. (Bertha von Suttner)

Ohne die Hilfe und Unterstützung einiger anderer wäre diese Arbeit nicht in dieser Form entstanden. Für all diese Menschen bin ich dankbar.

In besonderem Maße danke ich Frau Prof. Dr. Nicole Bäuerle für ihre intensive Betreuung und stete Förderung dieser Arbeit, für unzählige konstruktive Gespräche, für hilfreiche Kommentare und für wertvolle Hinweise. Ich bedanke mich nicht nur auf wissenschaftlicher Ebene und für das spannende Thema dieser Dissertation, sondern auch für ihre Geduld, ihr offenes Ohr und für den richtigen „Motivationsschub“ zur rechten Zeit. Herrn Prof. Dr. Alfred Müller danke ich für die Übernahme des Korreferats, sowie nützliche, hilfreiche Anmerkungen und Anregungen aus einer weiteren Perspektive.

Diese Arbeit entstand während meiner Zeit als wissenschaftliche Mitarbeiterin am Institut für Stochastik des Karlsruher Institut für Technologie (KIT). Herzlich bedanke ich mich bei Herrn Prof. Dr. Norbert Henze, der mir durch diese Stelle, nicht nur die Finanzierung meiner Promotionszeit ermöglichte, sondern mir auch die Chance gab durch die Zusammenarbeit bei den Lehrveranstaltungen didaktisch wie fachlich viel dazuzulernen und die Freude an der Stochastik zu vermitteln. Dem ganzen Institut danke ich für die außergewöhnlich inspirierende Arbeitsatmosphäre. Danke an meine Bürokollegen, die auch in schwierigen Phasen meiner Dissertation mir halfen die richtige Motivation wiederzufinden. Im speziellen danke ich Dr. Franziska Lindner, auf deren Unterstützung ich immer zählen konnte.

Aus tiefstem Herzen bedanke ich mich bei meiner Familie und Freunden für den Rückhalt und die liebevolle aber gelegentlich auch fordernde Motivationsleistung. Für die Unterstützung durch Korrekturlesen bedanke ich mich bei Prof. Dr. Birgitta König-Ries, Dipl.-Math. techn. Benedikt Galler und Dipl.-Math. techn. Fabian Oboril. Zu guter Letzt danke ich meinen Eltern Roswitha und Erich Riess und meiner Schwester Daniela für die Liebe und den Zusammenhalt, die ich nicht nur während meiner Promotionszeit, sondern schon mein ganzen Leben erfahren durfte.

Karlsruhe, Juni 2014

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Algorithmen

1 Einleitende Betrachtung

1.1. Problemstellung

Die schrittweise Liberalisierung der Energiemärkte mit Beginn in den 1990er Jahren, brachte große Veränderungen im Geschäftsfeld Energie mit sich. Aufgrund des entstandenen Wettbewerbs im Energiemarkt wurden aus einst regulierten, kostenorientierten Preisen für zum Beispiel Strom und Gas, marktbasierte Preise [9]. Es entstand ein Markt für Kassageschäfte und Derivate in vielen Ländern und Regionen in der ganzen Welt [4].

Energieversorgungsunternehmen reagierten auf diese Herausforderungen auf unterschiedlichen Strategieebenen, wobei Energiespeichermöglichkeiten eine Option darstellen. Um durch gezieltes Handeln die fluktuierenden Preise optimal auszunutzen, ist es notwendig die gehandelte „Ware“ zu speichern [11]. Damit ergibt sich für die Energiespeicher neben dem Versorgungsaspekt, die Möglichkeit einen weiteren zusätzlichen Ergebnisbeitrag für die Wertschöpfung im Unternehmen zu generieren [6].

Insbesondere im zunehmend wachsenden Erdgasmarkt sind solche Optionen der Speichermöglichkeiten von Bedeutung. Während früher die Gasspeicher von großen Unternehmen der Gasindustrie vor allem zur Regulierung der Nachfrage genutzt wurden, ist durch die Liberalisierung ein eigenständiger Geschäftsbereich entstanden [5, 24]. Jeder in der Gasindustrie hat derzeit die Möglichkeit, einen Gasspeicher zu nutzen, entweder zu „mieten“ oder zu kaufen, um einen zusätzlichen Ergebnisbeitrag zu erzielen [11]. Preisschwankungen geben den Käufern und Verkäufern die Chance die Ressource gewinnoptimal zu nutzen [24]. Dabei ist zu berücksichtigen, dass die Nachfrage nach Erdgas im Winter während der Heizperiode größer als im Sommer ist, sodass eine saisonale Fluktuation des Preises die Entscheidung mit beeinflusst. Basieren die Entscheidungen bezüglich des Nutzens des Gasspeichers jedoch nur auf den Saisonalitäten, so werden die Chancen der Optimierung des Gewinns nicht vollständig ausgenutzt, da tägliche Schwankungen des Preises ebenfalls mit einbezogen werden müssen [24].

> „To store or not to store.“[24]

Für den Nutzer eines Gasspeichers stellt sich das Problem diese Komplexität für seine

operativen Entscheidungen zu beherrschen. Dies bedeutet konkret, dass jeder Nutzer eines Gasspeichers, aufgrund der bisherigen Marktinformationen die Möglichkeit - die Option - hat, eine gewissen Menge an Gas zu kaufen und in den Speicher zu füllen, eine gewisse Menge an Gas zu verkaufen und aus dem Speicher zu entnehmen, oder weder Gas zu kaufen noch zu verkaufen. Somit wird ein Gasspeicher auch als Realoption interpretiert.

Ein weiterer Aspekt der Gasspeicherbewertung sind nicht nur die besonderen Preischarakteristiken des Gas-Spot-Preises, sondern auch die Rahmenbedingungen des Gasspeichers an sich. Es gibt verschiedene Arten von Gasspeichern: ursprüngliche Gas- oder Erdöllagerstätten, Kavernenspeicher in Salzstöcken, Porenspeicher in Kalk- oder Sandsteinschichten, LNG (liquified natural gas) Speichertanks usw. Jeder Speicher hat seine Charakteristiken, die in die Bewertung mit einbezogen werden müssen. Zum einen kann der Speicher nicht beliebig gefüllt oder meist auch nicht vollständig geleert werden, sodass ein vorgegebenes „base gas level“ erhalten bleiben muss. Eine weitere Besonderheit ist, dass die täglich zum Handeln verfügbare Gasmenge zudem aufgrund begrenzter Einspeicher- und Ausspeicherraten eingeschränkt ist. Diese Raten sind von verschiedenen Faktoren abhängig, wie zum Beispiel der derzeitigen Gassmenge im Speicher, dem Druck im Speicher, der Kompressionsfähigkeit usw. Im Allgemeinen ist die Einspeicher- bzw. Ausspeicherrate direkt vom aktuellen Füllstand, das heißt der derzeitigen Gasmenge im Speicher, abhängig [14]. Die Ausspeicherrate ist am größten, wenn der Gasspeicher voll ist, und nimmt mit abnehmender Gasmenge im Speicher ab. Die Einspeicherrate hingegen ist am größten, wenn der Gasspeicher leer ist, und nimmt mit zunehmender Gasmenge im Speicher ab [14]. Weiterhin ist zu bedenken, dass Transaktionskosten im Markt entstehen oder eine, zum Beispiel technisch bedingte, Gasmenge bei der Einspeisung bzw. Entnahme verloren geht. Ein Bid-Ask-Spread im Gasmarkt ergibt einen zusätzlichen Unterschied in Einkaufs- und Verkaufspreis des Erdgases. Die „Mietzeit“ eines Gasspeichers oder die Bewertung eines Gasspeichers ist in der Regel auf einen gewissen Zeitpunkt begrenzt, oft im Jahresrhythmus, was bei täglichen Entscheidungen einen langen Zeithorizont darstellt. Die Rahmenbedingungen bezüglich der Nutzung eines Gasspeichers werden auch als Gasspeichervertrag bezeichnet.

Grundlegende Fragestellungen zu dieser Situation, die in der vorliegenden Arbeit auch als „das Gasspeicherproblem“ bezeichnet wird, sind:

> Welchen Wert hat der Gasspeicher zu Beginn der Nutzung?
> Wann sollte wie viel Gas in den Speicher gegeben bzw. entnommen werden, um diesen Wert zu erreichen?

In der Literatur finden sich zur Formulierung des Problems sowohl Ansätze in diskreter Zeit [24, 5, 34, 10], als auch Ansätze in stetiger Zeit [11, 37, 32, 15, 23, 22]. Zu beachten ist, dass sich das Ausgangsproblem oft in Details unterscheidet, zum Beispiel im Einbeziehen des Bid-Ask-Spreads oder der Abhängigkeit der Einspeicher-/Ausspeicherraten vom aktuellen Füllstand. Viele Lösungsansätze der Fragestellungen basieren auf einem Stochastischen Entscheidungsproblem in diskreter bzw. stetiger Zeit und damit auf dem Prinzip der dynamischen Programmierung. Eine erste analytische Untersuchung des

Problems in stetiger Zeit in einem Spezialfall ist in Hodges [22] zu finden. Die üblich in der Literatur verwendeten numerischen Methoden in der Optionsbewertung sind: die Monte-Carlo Simulation, Binomial-/Trinomialbäume und numerische Verfahren für Partielle Differentialgleichungen. Letztere Methode verwenden Thompson u. a. [37] und Schlüter u. Davison [32], um das Problem in stetiger Zeit anzugehen. Für eine diskrete Modellierung ist die Methode allerdings nicht anwendbar. Für die stetige Formulierung entwickeln Felix u. Weber [16] eine Methode durch rekombinierende Bäume, Holland [23] und Carmona u. Ludkovski [11] wenden je ein Monte-Carlo Verfahren an. Die Monte-Carlo Methode in Carmona u. Ludkovski [11] bezieht sich auf eine Umformulierung des ursprünglichen Problems als optimales switching Problem. Ein Vergleich basierend auf je einer Methode jeder Art findet sich für die stetige Formulierung in Felix [15]. Da Entscheidungen eher täglich und nicht kontinuierlich in der Zeit getroffen werden, liegt dieser Arbeit eine diskrete Formulierung zu Grunde. Zur numerischen Lösung schlagen de Jong u. Walet [24] und Boogert u. de Jong [5] eine Monte-Carlo Methode vor, die auf dem Least-Square-Algorithmus von Longstaff u. Schwartz [25] basiert. Während de Jong u. Walet [24] nur bestimmte Entscheidungen zulassen, behandeln Boogert u. de Jong [5] das Problem in dieser Hinsicht allgemeiner. Eine weitere Monte Carlo Methode, die mit linearer Optimierung arbeitet, ist in Byers [10] zu finden. Basierend auf einer diskreten Formulierung als Markovsches Entscheidungsproblem, wie auch in [24, 5], wendet Secomandi [34] eine Baum-Methode an. Zudem zeigt Secomandi [34] im Fall mit konstanten Einspeicher-/Ausspeicherraten ausgehend von einer Konkavitätseigenschaft die Struktur der optimalen Entscheidungen.

Ziel dieser Arbeit ist es, ausgehend von einer diskreten Modellierung des Problems als Markovschen Entscheidungsprozess mit endlichem Horizont, eine strukturelle Analyse des Gasspeicherproblems durchzuführen, sowie verschiedene Algorithmen auf dieses Problem anzuwenden und zu vergleichen.

Daraus abgeleitet ergeben sich folgende Teilziele:

Teilziele der strukturellen Analyse sind die Untersuchung auf Struktureigenschaften in der sogenannten Wertefunktion, die den Gasspeicherwert beschreibt, die Beschreibung der optimalen (Handels-)Strategie in Spezialfällen, sowie die Erweiterung der von Secomandi [34] gezeigten Struktur der optimalen Politik auf den Fall füllstand-abhängiger Raten.

Ausgehend von einem zu spezifizierenden Gaspreismodell sollen im Teilziel „Algorithmen“ ein Binomialbaum-Algorithmus, ein Monte-Carlo Algorithmus und der Monte-Carlo Algorithmus mit Least-Square Ansatz analysiert und vergleichend bewertet werden. Auf der Dualitätstheorie für Markovsche Entscheidungsprozesse basierend wird ein Algorithmus auf das Gasspeicherproblem angewendet, um eine obere Schranke für den Gasspeicherwert zu erhalten.

1.2. Gang der Untersuchung

Zur Bearbeitung der Zielsetzung der vorliegenden Arbeit werden nach der einleitenden Betrachtung (Kapitel 1) zunächst die notwendigen Grundlagen über Markovsche Entscheidungsprozesse in diskreter Zeit mit endlichem Horizont gelegt. Kapitel 2 befasst sich demnach mit der Definition eines Markovschen Entscheidungsprozesses, sowie mit dem Prinzip der dynamischen Programmierung. In diesem Zusammenhang werden die Methoden zur Untersuchung des Problems auf Wohlgestelltheit, das heißt insbesondere auf Struktureigenschaften, zielkonform diskutiert.

Mit Hilfe der Grundlagen wird in Kapitel 3 demzufolge das Gasspeicherproblem als Markovscher Entscheidungsprozess in diskreter Zeit mit endlichem Horizont modelliert. Dabei wird nicht nur das Hauptproblem mathematisch beschrieben, sondern auch die im darauffolgenden Kapitel zusätzlich verwendeten Vereinfachungen und Erweiterungen. Das Gasspeicherproblem wird vereinfacht als optimales switching Problem behandelt, Vereinfachungen in der Restriktionenmenge, das heißt in den Einspeicher-/Ausspeicherraten, und des zugrundeliegenden Markov-Prozesses erklärt und die Erweiterung des Markov-Prozesses mit regime switching eingeführt.

Das zentrale Kapitel der strukturellen Analyse des Gasspeicherproblems (Kapitel 4) beginnt mit der Überprüfung der Wohlgestelltheit des Problems, das heißt mit der Überprüfung der sogenannten Integrabilitäts- und der Strukturannahme. Es werden Schrankenfunktionen für das Markovsche Entscheidungsproblem angegeben, sowie Stetigkeits- und Konkavitätsaussagen getroffen. Die darauffolgende Analyse der optimalen Strategie wird zunächst für die vereinfachten Modelle durchgeführt, beginnend mit dem optimal-switching Fall über den Fall vereinfachter Restriktionen mit speziellem Markov-Prozess, bis zum Fall vereinfachter Restriktionen mit allgemeinem Markov-Prozess. Das Kapitel abschließend wird die Struktur der optimalen Politik im allgemeinen Fall mit regime switching angegeben und weiter analysiert.

Zur weiteren Bearbeitung der Zielsetzung, dem Teil über Algorithmen, wird in Kapitel 5 ein Markov-Prozess vorgestellt, der den Gas-Spot-Preis modelliert. Außerdem werden die für die weiteren Betrachtungen notwendigen Eigenschaften des Gas-Preis-Modells diskutiert, sowie eine Simulationsmöglichkeit und eine Möglichkeit zur Approximation des Prozesses durch einen Binomialbaum dargelegt.

Kapitel 6 beschäftigt sich daraufhin mit möglichen Algorithmen zur Gasspeicherbewertung, die auf der Simulation und Approximation aufbauen. Im ersten Teil wird die grundlegende Struktur der Algorithmen unter Verwendung der in Kapitel 4 erhaltenen otpimalen Politik erklärt und die zentrale Frage, der Berechnung des sogenannten Continuation Value erläutert. Das nächste Teilkapitel bezieht sich auf die Approximation des Prozesses als Binomialbaum und entwickelt darauf aufbauend einen Algorithmus. Weiter werden zwei Ansätze für Monte-Carlo Algorithmen basierend auf der Simulation des Preisprozesses vorgestellt. Hierbei wird der Interpolationsansatz, den man wiederum in drei Teilansätze untergliedern kann, entwickelt und der Least-Square-Ansatz unter Verwendung der Struktur der optimalen Politik beschrieben. Zur Schätzung oberer

Schranken des Gasspeicherwertes werden basierend auf der Dualitätstheorie für Markovsche Entscheidungsprozesse Monte-Carlo Algorithmen mittels Information Relaxation auf das Gasspeicherproblem angewandt.

Ausgehend von einem Beispiel eines Gasspeichers werden darauffolgend die Algorithmen in Kapitel 7 numerisch betrachtet. Zunächst wird auf die im Zusammenhang mit der Implementierung entstehende Problematik der Diskretisierung eingegangen, sowie eine Methode zur Diskretisierung des Speicherfüllstandes entwickelt. Daraufhin werden die Algorithmen zunächst in ihren Ansätzen einzeln analysiert und dann auf Laufzeit und Einfluss der Preisparameter hin verglichen. Das Gasspeicherproblem bzw. die Grundfragestellungen werden dann mit Hilfe der Algorithmen veranschaulicht, indem die optimale Strategie, sowie die Gasspeicherentwicklung während der Nutzungsdauer illustriert werden. Abschließend folgt die numerische Betrachtung der Algorithmen zur Schätzung oberer Schranken.

Die abschließende Betrachtung in Kapitel 8 greift die Zielsetzung dieser Arbeit auf, fasst die Ergebnisse zusammen und gibt einen Ausblick auf Erweiterungsmöglichkeiten und weitere Untersuchungsansätze.

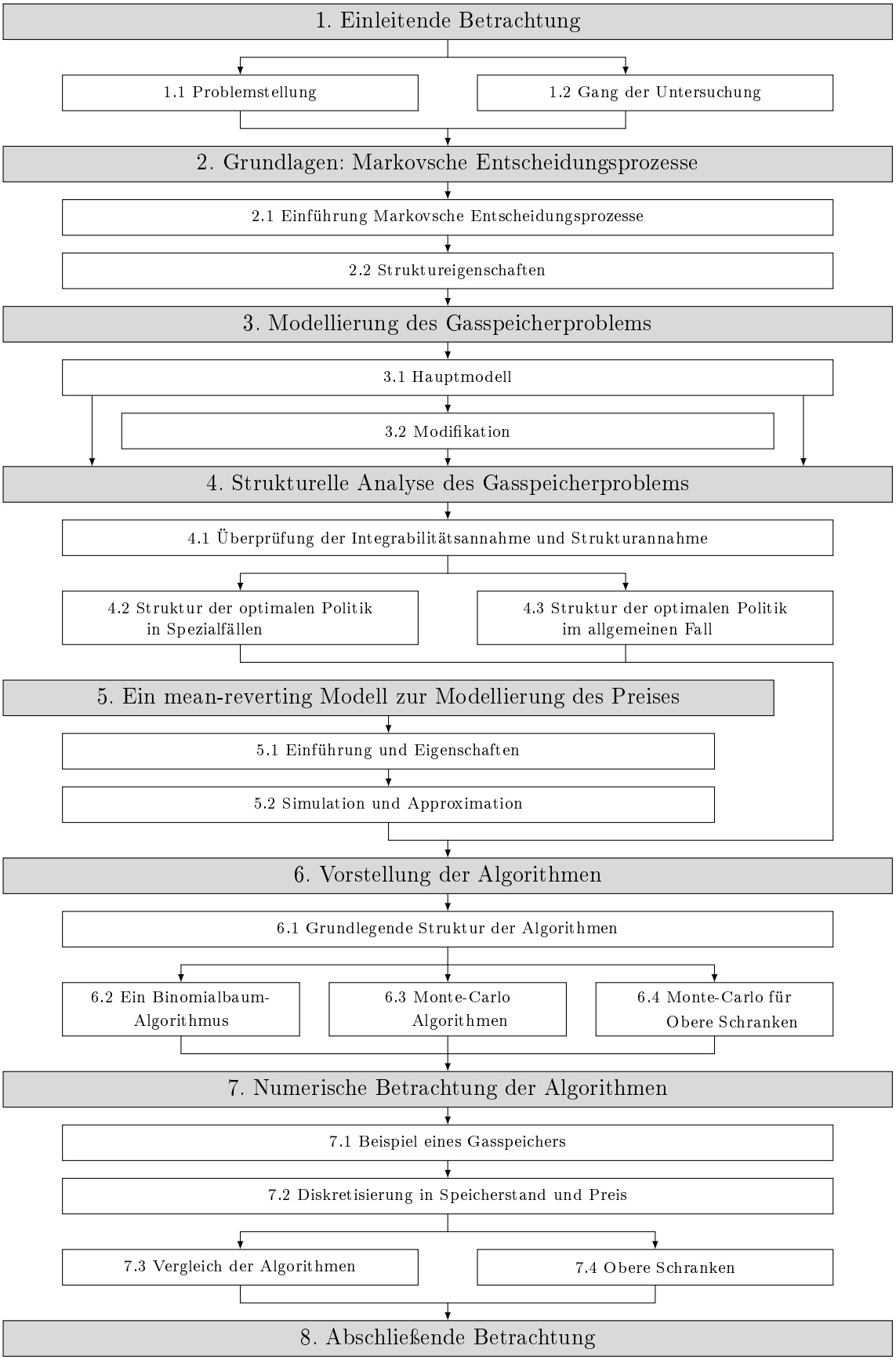


Abbildung 1.1.: Gang der Untersuchung

Teil I.

Strukturaussagen

2 Grundlagen: Markovsche Entscheidungsprozesse in diskreter Zeit mit endlichem Horizont

Um das in Kapitel 1 eingeführte Gasspeicherproblem mathematisch präzise formulieren zu können, werden in diesem Kapitel dazu die Grundlagen gelegt.

So wie in dieser Arbeit dient das stochastische dynamische Programm in vielen zufälligen Entscheidungsprozessen als Grundlage eines mathematischen Modells für dieselben. Da in dieser Arbeit das Problem zeitdiskret und mit endlichem Zeithorizont behandelt wird, beschränkt sich folgendes Kapitel auch auf diesen Fall. Es sei aber bemerkt, dass sich die Konzepte dieses Kapitels auch auf einen unendlichen Horizont und stetige Zeit erweitern lassen (vergleiche hierzu Bäuerle u. Rieder [8]).

Der erste Teil dieses Kapitels befasst sich mit der Definition eines Markovschen Entscheidungsprozesses in diskreter Zeit mit endlichem Horizont. Im zweiten Teil folgen strukturelle Bedingungen an den Markovschen Entscheidungsprozess und an die Eigenschaften des Markovschen Entscheidungsprozess.
Falls nicht anders notiert werden die Definitonen, Lemmata und Sätze aus Bäuerle u. Rieder [8] zitiert.

2.1. Das stochastische dynamische Programm, der Markovsche Entscheidungsprozess und die Bellman Gleichung

Stochastische dynamische Programme sind 6-Tupel bestehend aus einem Zustandsraum, einem Aktionenraum, einer Restriktionenmenge, einem Übergangskern - der Markovsch ist -, einer Gewinnfunktion und einer terminalen Gewinnfunktion. Dies wird in der folgenden Definition präzisiert.

Definition 2.1.1 (Stochastisches dynamisches Programm)
Ein (nicht-stationäres) *stochastisches dynamisches Programm* (engl. Markov Decision Model) mit Planungshorizont $N \in \mathbb{N}$ besteht aus dem Tupel $(E, A, D_n, Q_n, h_n, h_N)$ mit folgender Bedeutung:

- $E \neq \emptyset$ ist der *Zustandsraum*, versehen mit der σ-Algebra $\mathcal{E}$. Elemente des Zustandsraums werden mit $x \in E$ bezeichnet.
- $A \neq \emptyset$ ist der *Aktionenraum*, versehen mit der σ-Algebra $\mathcal{A}$. Elemente des Aktionenraums werden mit $a \in A$ bezeichnet.
- $D_n \subseteq E \times A$ ist eine messbare Teilmenge von $E \times A$. Diese Teilmenge ist die *Restriktionenmenge* zur Zeit n.
  Es wird angenommen, dass D_n den Graph einer messbaren Abbildung $f_n : E \to A$ enthält, das heißt $(x, f_n(x)) \in D_n$ für alle $x \in E$. Die Menge $D_n(x) := \{a \in A \mid (x, a) \in D_n\}$ ist die Menge aller *zulässigen Aktionen* in Zustand x ($x \in E$) zur Zeit n.
- Q_n ist der *stochastische Übergangskern* von D_n nach E auf Stufe n. Das bedeutet, für jedes Paar $(x, a) \in D_n$ (fest) ist $B \mapsto Q_n(B|x, a)$ ein Wahrscheinlichkeitsmaß auf $\mathcal{E}$ und $(x, a) \mapsto Q_n(B|x, a)$ ist messbar für jedes $B \in \mathcal{E}$.
  Befindet sich der Prozess zur Zeit n in Zustand x und wird Aktion a gewählt, so ist $Q_n(B|x, a)$ die Wahrscheinlichkeit, dass der Zustand zur Zeit $n + 1$ in B liegt.
- $h_n : D_n \to \mathbb{R}$ ist eine messbare Funktion, die sogenannte *Gewinnfunktion*. Hierbei gibt $h_n(x, a)$ den Einstufengewinn zur Zeit n an, wenn x der aktuelle Zustand und a die gewählte Aktion ist.
- $h_N : E \to \mathbb{R}$ ist eine messbare Funktion, die sogenannte *terminale Gewinnfunktion*. Hierbei gibt $h_N(x)$ den Gewinn zur Zeit N an, wenn der aktuelle Zustand x ist.

Bemerkung 2.1.2
In einigen Anwendungen, so auch im Gasspeicherproblem, sind der Zustands- und der Aktionenraum Borelteilmengen von Polnischen Räumen, das heißt von vollständig metrisierbaren, separablen Räumen, oder gar endlich oder abzählbare Mengen. In diesem Fall werden oft die σ-Algebren $\mathcal{E}$ und $\mathcal{A}$ als die σ-Algebren der Borelmengen von E bzw. A gewählt und diese $\mathcal{B}(E)$ bzw. $\mathcal{B}(A)$ notiert.

Zu jedem Zeitpunkt entscheidet man sich für eine bestimmte Aktion, die aus der Restriktionenmenge stammt. Diese Entscheidungen setzen sich zur einer sogenannten Strategie bzw. Politik zusammen.

Definition 2.1.3 (Entscheidungsregel, Politik)

a) Eine messbare Abbildung $f_n : E \to A$ mit $f_n(x) \in D_n(x)$ für alle $x \in E$ heißt *Entscheidungsregel* zur Zeit n. Es bezeichne F_n die Menge aller Entscheidungsregeln zur Zeit n.

b) Eine Folge von Entscheidungsregeln $\pi = (f_0, \ldots, f_{N-1})$ mit $f_n \in F_n$ für alle n heißt *N-stufige Politik* oder *N-stufige Strategie*.

Die Menge aller Entscheidungsregeln F_n ist nicht leer, da nach Definition 2.1.1 die Menge D_n den Graphen einer messbaren Abbildung $f_n : E \to A$ enthält.

Ein stochastisches dynamisches Programm soll dazu dienen ein N-stufiges Zufallsexperiment zu beschreiben. Daher wird ein zugrundeliegenden Wahrscheinlichkeitsraum wie folgt konstruiert:

Konstruktion und Definition 2.1.4

Sei $(\Omega, \mathcal{F})$ ein Messraum mit $\Omega = E^{N+1}$ und $\mathcal{F} = \mathcal{E} \otimes \cdots \otimes \mathcal{E}$.
Die Zufallsvariablen $X_0, \ldots, X_N$ seien durch

$$X_n(\omega) = X_n((i_0, i_1, \ldots, i_N)) = i_n$$

definiert, d.h. die n-te Projektion von ω. Die Zufallsvariable X_n beschreibt den Zustand des Systems zur Zeit n und (X_n) heißt *Markovscher Entscheidungsprozess* (engl. Markov Decision Process, kurz: MDP).
Sei $\pi = (f_0, \ldots, f_{N-1}) \in F_0 \times F_1 \times \cdots \times F_{N-1}$ eine Strategie und $x \in E$ ein Anfangswert. Nach dem Satz von Ionescu-Tulcea gibt es dann genau ein Wahrscheinlichkeitsmaß $\mathbb{P}_x^\pi$ auf $(\Omega, \mathcal{F})$ mit

(i) $\mathbb{P}_x^\pi(X_0 \in B) = \delta_x(B)$ für alle $B \in \mathcal{E}$,

(ii) $\mathbb{P}_x^\pi(X_{n+1} \in B | X_0, \ldots, X_n) = \mathbb{P}_x^\pi(X_{n+1} \in B | X_n) = Q_n(B | X_n, f_n(X_n))$.

Für den Satz von Ionesu-Tulcea siehe zum Beispiel Bäuerle u. Rieder [8] Proposition B.2.5.

Die Folge $(X_0, \ldots, X_N)$ ist eine inhomogene Markovkette bezüglich $\mathbb{P}_x^\pi$ (siehe (ii)).

Im Folgenden bezeichnen $\mathbb{E}_x^\pi$ den Erwartungswert bezüglich $\mathbb{P}_x^\pi$, $\mathbb{P}_{nx}^\pi$ die bedingte Wahrscheinlichkeit $\mathbb{P}_{n,x}^\pi(\cdot) := \mathbb{P}^\pi(\cdot | X_n = x)$ und $\mathbb{E}_{n,x}^\pi$ den entsprechenden Erwartungswertoperator.

Aufgrund folgender Annahme ist gesichert, dass alle folgenden Erwartungswerte wohldefiniert sind.

Integrabilitätsannahme (A_N)
Für $n = 0, 1, \ldots, N$ gelte

$$\delta_n^N(x) := \sup_\pi \ \mathbb{E}_{n,x}^\pi \left[\sum_{k=n}^{N-1} h_k^+(X_k, f_k(X_k)) + h_N^+(X_N) \right] < \infty, \ x \in E.$$

Hierbei bezeichne $y^+ := \max\{0, y\}$.

Ziel ist es durch geschickte Wahl der Strategie den erwarteten Gesamtgewinn zu maximieren. Im Folgenden wird das Optimierungsproblem mathematisch formuliert.

Definition 2.1.5
a) Für $\pi \in F_0 \times \cdots \times F_{N-1}$, sei $V_{n\pi} : E \to \mathbb{R}$ definiert durch

$$V_{n\pi}(x) := \mathbb{E}_{n,x}^\pi \left[\sum_{k=n}^{N-1} h_k(X_k, f_k(X_k)) + h_N(X_N) \right].$$

$V_{n\pi}(x)$ ist der *erwartete Gewinn* zur Zeit n über die noch folgenden Stufen n bis N bei Start in $x \in E$ (zur Zeit n) und unter Verwendung der Politik π.

b) Die *Wertfunktion* $V_n : E \to \mathbb{R}$ sei definiert durch

$$V_n(x) = \sup_\pi V_{n\pi}(x).$$

Dabei bezeichnet $V_n(x)$ den *maximalen erwarteten Gewinn* zur Zeit n (über die verbleibenden Stufen n bis N) bei Start in x.

c) Eine Politik $\pi \in F_0 \times \cdots \times F_{N-1}$ heißt *optimal*, falls

$$V_{0\pi}(x) = V_0(x) \ \text{ für alle } x \in E.$$

Bemerkung 2.1.6
Für die terminale Stufe N entspricht die Wertfunktion der terminalen Gewinnfunktion für jede Strategie, das heißt es gilt $V_{N\pi}(x) = V_N(x) = h_N(x)$.
Obwohl die Politik $\pi = (f_0, \ldots, f_{N-1})$ von allen Zeitpunkten abhängt, sind für die Wertefunktion $V_{n\pi}$ zur Zeit n nur noch die folgenden Entscheidungen der Politik, also $(f_n, \ldots, f_{N-1})$, relevant.

Um die Notation zu vereinfachen und letztendlich die Lösung bzw. den Lösungsweg des Optimierungsproblems anzugehen, werden zwei Operatoren eingeführt.

Im Folgenden sei $\mathbb{M}(E) := \{v : E \to [-\infty, \infty) | v \text{ ist messbar}\}$. Nach Voraussetzung gilt $V_{n\pi} \in \mathbb{M}(E)$ für alle π und n.

Definition 2.1.7 (Operatoren)
Auf $\mathbb{M}(E)$ seien die folgenden Operatoren definiert:

a) Für $v \in \mathbb{M}(E)$, $n = 0, 1, \ldots, N-1$ und $(x, a) \in D_n$ sei

$$(L_n v)(x, a) := h_n(x, a) + \int v(x') Q_n(dx'|x, a),$$

falls das Integral existiert.

b) Für $v \in \mathbb{M}(E)$, $n = 0, 1, \ldots, N-1$, $f \in F_n$ und $x \in E$

$$(T_{nf} v)(x) := (L_n v)(x, f(x)).$$

c) Für $v \in \mathbb{M}(E)$, $n = 0, 1, \ldots, N-1$ und $x \in E$ sei

$$(T_n v)(x) := \sup_{a \in D_n(x)} (L_n v)(x, a).$$

falls das Integral exisitiert.
T_n heißt *Maximal-Gewinn-Operator* auf Stufe n.

Zwischen den Operatoren besteht die Beziehung

$$T_n v = \sup_{f \in F_n} T_{nf} v.$$

Es gilt zwar, dass $T_{nf} v \in \mathbb{M}(E)$ liegt für alle $v \in \mathbb{M}(E)$, jedoch ist nicht unbedingt gegeben, dass $T_n v$ zu $\mathbb{M}(E)$ gehört.

Zur Vereinfachung der Notation schreibt man auch $T_n v(x)$ für $(T_n v)(x)$ bzw. $T_n T_{n+1} v$ für $(T_n(T_{n+1} v))$.

Bemerkung 2.1.8
Alle drei Operatoren sind wachsend.
Genaueres hierzu findet sich in Bäuerle u. Rieder [8].

Strategien, die zur Optimierung des gegeben Problems führen, werden Maximisatoren genannt.

Definition 2.1.9 (Maximisator)
Sei $v \in \mathbb{M}(E)$. Eine Entscheidungsregel $f \in F_n$ heißt *Maximisator* von v auf Stufe n, falls

$$(L_n v)(x, f(x)) = T_n v(x) \quad \text{für alle } x \in E,$$

d.h. $f(x)$ ist Maximumstelle von

$$a \mapsto (L_n v)(x, a) \quad a \in D_n(x) \text{ für alle } x \in E.$$

Die sogenannte Gewinniteration bildet die Grundlage zur Lösung des Optimierungsproblems. Sie besagt, dass man nun - mit Hilfe des Operators T_{nf} - den erwarteten Gewinn zu einer Strategie rekursiv berechnen kann.

Satz 2.1.10 (Gewinniteration)
Sei $\pi = (f_0, \dots, f_{N-1})$ eine N-stufige Politik. Dann gilt für $n = 0, 1, \dots N-1$

a) $V_{N\pi} = h_N$ und $V_{n\pi} = T_{nf_n} V_{n+1,\pi}$,

b) $V_{n\pi} = T_{nf_n} \cdots T_{N-1,f_{N-1}} h_N$.

Beweis
Beweis siehe Theorem 2.3.4 in Bäuerle u. Rieder [8]. ∎

Ein weiterer wichtiger Baustein zur Lösung des Optimierungsproblems bildet die Bellman-Gleichung.

Bellman-Gleichung
Die Folge (V_n) erfüllt die sogenannte Bellman-Gleichung

$$\begin{aligned} V_N &= h_N \\ V_n &= T_n V_{n+1}, \quad n = 0, 1, \dots, N-1. \end{aligned} \tag{2.1}$$

Der nächste Satz zeigt, dass sobald eine Lösung der Bellman-Gleichung zusammen mit einer Folge von Maximisatoren existiert, dann führt dies auch zu einer Lösung des Optimierungsproblem.

Somit gibt er auch eine zentrale Lösungsmethode für das Markovsche Entscheidungsproblem an.

Satz 2.1.11 (Verifikationssatz)
Sei $(v_n) \subset \mathbb{M}(E)$ eine Lösung der Bellman-Gleichung. Dann gilt

a) $v_n \geq V_n$ für $n = 0, 1, \dots N$.

b) Falls $f^\star$ ein Maximisator von v_n ist für $n = 0, 1, \dots N-1$, dann ist $v_n = V_n$ und die Politik $\pi^\star = (f_0^\star, \dots, f_{N-1}^\star)$ ist optimal für das N-stufige Markovsche Entscheidungsproblem.

Beweis
Beweis siehe Theorem 2.3.7 in Bäuerle u. Rieder [8]. ∎

Der Verifaktionssatz bietet zwar eine Lösungsmethode, jedoch ist im Allgemeinen nicht garantiert, dass eine optimale Politik existiert. Dafür müssen weitere Annahme über die Struktur des Problems gemacht werden.

Strukturannahme (SA_N)
Es gibt Mengen $\mathbb{M}_n \subseteq \mathbb{M}(E)$ und $\Delta_n \subseteq F_n$, sodass für $n = 0, 1, \dots N-1$ gilt

(i) $h_N \in \mathbb{M}_N$.

(ii) Falls $v \in \mathbb{M}_{n+1}$, dann ist $T_n v$ wohldefiniert und $T_n v \in \mathbb{M}_n$.

(iii) Für alle $v \in \mathbb{M}_{n+1}$ existiert ein $f \in \Delta_n$ für das $T_{nf} v(x) = T_n v(x)$ gilt, d.h. f ist ein Maximisator von v auf Stufe n.

Gilt die Strukturannahme (SA_N), so lässt sich zeigen, dass das Markovsche Entscheidungsproblem gelöst werden kann, indem die N (einstufigen) Optimierungsprobleme rekursiv gelöst werden.

Satz 2.1.12 (Struktursatz)
Gilt (SA_N), so folgt:

a) $V_n \in \mathbb{M}_n$ und die Folge (V_n) erfüllt die Bellman-Gleichung, d.h. für $n = 0, 1, \dots, N-1$ gilt

$$\begin{aligned} V_N(x) &= h_N(x), \\ V_n(x) &= \sup_{a \in D_n(x)} \left\{ h_n(x,a) + \int V_{n+1}(x') Q_n(dx'|x,a) \right\}, \quad x \in E. \end{aligned} \tag{2.2}$$

b) $V_n = T_n T_{n+1} \cdots T_{N-1} h_N$.

c) Für $n = 0, 1, \dots, N-1$ existiert ein Maximisator f_n von V_{n+1} mit $f_n \in \Delta_n$ und jede Folge von Maximisatoren $f_n^\star$ von V_{n+1} definiert eine optimale Politik $(f_0^\star, \dots, f_{N-1}^\star)$ für das N-stufige Markovsche Entscheidungsproblem.

Beweis
Beweis siehe Theorem 2.3.8 in Bäuerle u. Rieder [8]. ∎

Zur Lösung des Optimierungsprolems liefert Satz 2.1.12 folgenden Algorithmus.

Rückwärtsinduktionsalgorithmus

1. Setze $n = N$ und für jedes $x \in E$
$$V_N(x) := h_N(x).$$
2. Setze $n := n - 1$ und berechne für alle $x \in E$
$$V_n(x) = \sup_{a \in D_n(x)} \left\{ h_n(x,a) + \int V_{n+1}(x') Q_n(dx'|x,a) \right\}.$$
Berechne einen Maximisator $f_n^\star$ von V_{n+1}.

3. Ist $n = 0$, so wird die Wertfunktion V_0 berechnet und die optimale Politik $\pi^\star$ ist gegeben durch $\pi^\star = (f_0^\star, \ldots, f_{N-1}^\star)$. Andernfalls gehe zurück zu Schritt 2.

Bemerkung 2.1.13
Satz 2.1.12 besagt, dass die Maximisatoren zu einer optimalen Strategie führen. Jedoch enthalten optimale Strategien nicht unbedingt Maximisatoren.

Ein Beispiel hierfür befindet sich in Bäuerle u. Rieder [8] Beispiel 2.3.10.

2.2. Struktureigenschaften

Im folgenden Teil werden allgemeine Bedingungen angegeben, unter denen die Integrabilitätsannahmen (A_N) und die Strukturannahme (SA_N) erfüllt sind.

2.2.1. Bedingung für (A_N)

Hinreichend für die Erfüllung der Integrabilitätsannahme ist die Existenz einer oberen Schrankenfunktion. Die Schrankenfunktion ist für die weiteren Betrachtungen noch von Bedeutung.

Definition 2.2.1 (Schrankenfunktion)
a) Eine messbare Abbildung $s : E \to \mathbb{R}^+$ heißt *obere Schrankenfunktion* für das stochastische dynamische Prgramm, falls $c_1, c_2, c_3 \in \mathbb{R}^+$ existieren, sodass für alle $n = 0, 1, \ldots N-1$:

(i) $h_n^+(x,a) \leq c_1 s(x)$ für alle $(x,a) \in D_n$

(ii) $h_N^+(x) \leq c_2 s(x)$ für alle $x \in E$

(iii) $\int s(x') Q_n(dx'|x,a) \leq c_3 s(x)$ für alle $(x,a) \in D_n$

b) Eine messbare Funktion $s : E \to \mathbb{R}^+$ heißt *Schrankenfunktion* für das stochastische dynamische Prgramm, falls $c_1, c_2, c_3 \in \mathbb{R}^+$ existieren, sodass für alle $n = 0, 1, \ldots N-1$:

(i) $|h_n(x,a)| \leq c_1 s(x)$ für alle $(x,a) \in D_n$

(ii) $|h_N(x)| \leq c_2 s(x)$ für alle $x \in E$

(iii) $\int s(x') Q_n(dx'|x,a) \leq c_3 s(x)$ für alle $(x,a) \in D_n$

Satz 2.2.2
Hat das stochastische dynamische Programm eine obere Schrankenfunktion s, dann sind die Integrabilitätsbedingungen (A_N) erfüllt.

Beweis
Beweis siehe Proposition 2.4.2 in Bäuerle u. Rieder [8]. ∎

Im Folgenden seien E und A nicht-leere Borel-Teilmengen von Polnischen Räumen. Auf E und A seien $\mathcal{E}$ bzw. $\mathcal{A}$ die σ-Algebren der Borel-Mengen von E und A. Weiter sei D_n eine Borel-Teilmenge von $E \times A$.

Außerdem besitze das stochastische dynamische Programm eine obere Schrankenfunktion s. Für $v \in \mathbb{M}(E)$ bezeichne

$$\|v\|_s := \sup_{x \in E} \frac{|v(x)|}{s(x)}$$

die gewichtete Supremums-Norm, wobei $\frac{0}{0}$ als 0 definiert wird. Weiter definiert man

$$\mathbb{S}_s := \{v \in \mathbb{M}(E) |\ \|v\|_s < \infty\}$$

und

$$\mathbb{S}_s^+ := \{v \in \mathbb{M}(E) |\ \|v^+\|_s < \infty\}.$$

Es gilt

$$\mathbb{S}_s := \{v \in \mathbb{M}(E) |\ |v(x)| \leq c \cdot s(x) \ \text{ für alle } \ x \in E \text{ und ein } c \in \mathbb{R}_+\}$$

bzw.

$$\mathbb{S}_s^+ := \{v \in \mathbb{M}(E) |\ v^+(x) \leq c \cdot s(x) \ \text{ für alle } \ x \in E \text{ und ein } c \in \mathbb{R}_+\}.$$

2.2.2. Stetigkeitsbedingung für (SA_N)

Ein Konzept das letztendlich zur Erfüllung der Strukturannahmen führt, ist das der Stetigkeit. Dazu sei im Folgenden M ein metrischer Raum und $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$.

Definition 2.2.3

a) Eine Abbildung $f : M \to \overline{\mathbb{R}}$ heißt *oben halbstetig* (kurz: ohs), falls für alle Folgen $(x_n) \subset M$ mit $\lim\limits_{n\to\infty} x_n = x \ \in M$ gilt

$$\limsup_{n\to\infty} f(x_n) \leq f(x).$$

b) Eine Abbildung $f : M \to \overline{\mathbb{R}}$ heißt *unten halbstetig* (kurz: uhs), falls $-f$ oben halbstetig ist.

c) Eine Abbildung $f : M \to \overline{\mathbb{R}}$ heißt *stetig*, falls f unten und oben halbstetig ist.

Da in einem stochastischen dynamischen Programm auch mit Mengen gearbeitet wird, ist es notwendig, die Halbstetigkeit bzw. Stetigkeit auch für mengenwertige Funktionen zu definieren.

Definition 2.2.4

a) Eine mengenwertige Abbildung $x \mapsto D(x)$ heißt *oben halbstetig*, falls für alle $x \in E$ gilt:
Falls $\lim_{n\to\infty} x_n = x$ und $a_n \in D(x_n)$ für alle n, dann hat (a_n) einen Häufungspunkt in $D(x)$.

b) Eine mengenwertige Abbildung $x \mapsto D(x)$ heißt *unten halbstetig*, falls für alle $x \in E$ gilt:
Falls $\lim_{n\to\infty} x_n = x$, dann ist jeder Punkt in $D(x)$ ein Häufungspunkt einer Folge von Punkten $a_n \in D(x_n)$ für alle $n \in \mathbb{N}$.

c) Eine mengenwertige Abbildung $x \mapsto D(x)$ heißt *stetig*, falls sie oben und unten halbstetig ist.

Mit diesen Definitionen lässt sich der folgende Struktursatz formulieren, er bildet die Grundlage des Korrollar 2.2.6, das die Erfüllung der Strukturannahme liefert.

Satz 2.2.5

Sei $v \in \mathbb{S}_s^+$ und stetig. Gilt außerdem

(i) $D_n(x)$ kompakt $\forall\, x \in E$ und $x \mapsto D_n(x)$ stetig

(ii) $(x, a) \mapsto L_n v(x, a)$ stetig auf D_n,

dann ist $T_n v$ stetig und es existiert ein Maximisator $f_n \in F_n$ von v. Hat v einen eindeutigen Maximisator $f_n \in F_n$, dann ist f_n stetig.

Beweis

Beweis siehe Proposition 2.4.3 in Bäuerle u. Rieder [8]. ∎

Mit $\mathbb{M}_n := \{v \in \mathbb{S}_s^+ |\ v \text{ stetig}\}$ und $\Delta_n := F_n$ sind die Strukturannahmen (SA_N) für das stochastistische dynamische Programm erfüllt.
Mit Satz 2.2.5, sowie Satz 2.1.12 folgt dann unmittelbar

Korollar 2.2.6
Gegeben sei ein stochastisches dynamisches Programm mit oberer Schrankenfunktion s. Falls für $n = 0, 1, \ldots, N-1$ gilt:

(i) $D_n(x)$ ist kompakt für alle $x \in E$ und $x \mapsto D_n(x)$ ist stetig,

(ii) $(x, a) \mapsto \int v(x')Q_n(dx'|x, a)$ ist stetig für alle stetigen $v \in \mathbb{S}_s$,

(iii) $(x, a) \mapsto h_n(x, a)$ ist stetig,

(iv) $x \mapsto h_N(x)$ ist stetig.

Dann erfüllen die Mengen $\mathbb{M}_n := \{v \in \mathbb{S}_s^+ |\ v \text{ stetig}\}$ und $\Delta_n := F_n$ die Strukturannahme (SA_N).
Es ist $V_n \in \mathbb{M}_n$ und es existiert ein Maximisator $f_n^\star \in F_N$ von V_{n+1}. Die Politik $(f_0^\star, \ldots, f_{N-1}^\star)$ ist optimal.

Einige hilfreiche Aussagen zum Nachprüfen der Stetigkeit liefert das nachfolgende Lemma, siehe Lemma A.2.2 im Anhang von Bäuerle u. Rieder [8].

Lemma 2.2.7
a) Eine mengenwertige Abbildung $x \mapsto D(x)$ ist oben halbstetig genau dann wenn jede Folge $(x_n, a_n) \subset D$ mit der Eigenschaft, dass (x_n) in E konvergiert, einen Häufungspunkt in D hat.
Es folgt: D ist kompakt $\Longrightarrow$ $x \mapsto D(x)$ ist oben halbstetig $\Longrightarrow$ D ist abgeschlossen.

b) Sei A kompakt. Dann ist $x \mapsto D(x)$ oben halbstetig genau dann wenn D abgeschlossen ist.

c) Sei A kompakt und $D(x) = A$ für alle x, dann ist $x \mapsto D(x)$ stetig.

d) Sei $A = \mathbb{R}$ und $D(x) = [\underline{d}(x), \overline{d}(x)]$. Dann ist $x \mapsto D(x)$ stetig, falls $\underline{d} : E \to \mathbb{R}$ stetig und $\overline{d} : E \to \mathbb{R}$ stetig ist.

2.2.3. Konkavitätsbedingung für (SA_N)

Eine weitere hinreichende Bedingung für die Strukturannahmen ist die Konkavität gewisser Operatoren und Funktionen, sowie die Konvexität einiger Mengen. Diese Aussage wird im nachfolgenden Satz und im daraus folgenden Korollar spezifiziert.

Sei $E \subset \mathbb{R}^d$ und $A \subset \mathbb{R}^m$.

Satz 2.2.8
Sei $v \in \mathbb{S}_s^+$. Gilt weiter

(i) D_n konvex in $E \times A$

(ii) $(x, a) \mapsto L_n v(x, a)$ konkav auf D_n.

Dann ist $T_n v$ konkav auf E.

Aus Satz 2.2.8 und einigen weiteren Bedingungen folgt nun unmittelbar, dass die Strukturannahmen erfüllt sind. Die Mengen $\mathbb{M}_n$ und Δ_n werden konkret angegeben.

Korollar 2.2.9
Gegeben sei ein stochastisches dynamische Programm mit oberer Schrankenfunktion s. Falls für $n = 0, 1, \ldots, N-1$ gilt:

(i) D_n ist konvex in $E \times A$.

(ii) die Abbildung $(x, a) \mapsto \int v(x')Q_n(dx'|x, a)$ ist konkav für alle konkaven $v \in \mathbb{S}_s^+$,

(iii) $(x, a) \mapsto h_n(x, a)$ ist konkav,

(iv) $x \mapsto h_N(x)$ ist konkav,

(v) für alle konkaven $v \in \mathbb{S}_s^+$ existiert ein Maximisator $f_n \in \Delta_n$ von v.

Dann erfüllen die Mengen $\mathbb{M}_n := \{v \in \mathbb{S}_s^+ |\ v \text{ konkav}\}$ und $\Delta_n := F_n$ die Strukturannahmen (SA_N).

3 Modellierung des Gasspeicherproblems

Mit dem Konzept des stochastischen dynamischen Programms, vorgestellt in Kapitel 2.1, lässt sich nun das Gasspeicherproblem als ein solches formulieren. Das hier vorgestellte Modell und dessen Modifikationen bilden die Grundlage aller weiteren Untersuchungen in dieser Arbeit.

Das im ersten Teil des Kapitels vorgestellte Modell, hier auch „Hauptmodell“ genannt, versucht einen möglichst allgemeinen Fall des Gasspeicherproblems abzubilden. Während sich die Modifikationen im zweiten Teil auf Spezialfälle, wie etwa den Fall, den Speicher an einem Tag vollständig befüllen bzw. entleeren zu können, beschränken, wird im zweiten Teil eine Erweiterung des Hauptmodells auf eine Markov-Kette mit regime switching vorgestellt.

3.1. Hauptmodell für das Gasspeicherproblem

Folgende Formulierung des Gasspeicherproblems als stochastisches dynamisches Programm mit endlichem Planungshorizont in diskreter Zeit wurde bereits in ähnlicher Weise von Boogert u. de Jong [5] vorgestellt.

Dazu sei $N \in \mathbb{N}$ der endliche Planungshorizont des Gasspeicherproblems, das heißt das Ende des Vertrages. Sei weiter (P_n) ein Markovscher Prozess auf $(\mathcal{P}, \mathcal{B}(\mathcal{P}))$ mit Übergangskern (Q_n), wobei $\mathcal{P} \subseteq \mathbb{R}^+$. Zudem sei

- $E = [b^{\min}, b^{\max}] \times \mathcal{P}$ der Zustandsraum, versehen mit der σ-Algebra $\mathcal{B}([b^{\min}, b^{\max}]) \otimes \mathcal{B}(\mathcal{P})$.

  Hierbei bezeichne $b^{\min}$ bzw. $b^{\max}$ den minimalen bzw. maximal möglichen Speicherstand des Gasspeichers.

  Die Elemente aus E seien durch (x, p) bezeichnet.

- $A = [b^{\min} - b^{\max}, b^{\max} - b^{\min}]$ mit $\mathcal{B}([b^{\min} - b^{\max}, b^{\max} - b^{\min}])$ der Aktionenraum.

  Hierbei bedeutet ein $a < 0$, dass die Menge $|a|$ aus dem Speicher entnommen wird („ausgespeichert wird"); ein $a > 0$, dass die Menge $|a|$ in den Speicher eingefüllt („eingespeichert") wird.

- $D_n(x) = \{a \in A | \max(b^{\min} - x, i_n^{\min}(x)) \leq a \leq \min(b^{\max} - x, i_n^{\max}(x))\}$.
  Hierbei sei $i_n^{\min}$ bzw. $i_n^{\max}$ die maximale Ausspeicher- bzw. Einspeicherrate in Abhängigkeit des aktuellen Speicherstandes x.

  Es wird angenommen, dass $i_n^{\max}$ und $i_n^{\min}$ monoton fallende Funktionen sind, wobei $i_n^{\max}$ nichtnegativ und konkav und $i_n^{\min}$ nichtpositiv und konvex für $b^{\min} \leq x \leq b^{\max}$ ist.

- $\tilde{Q}_n$ sei der Übergangskern mit $\tilde{Q}_n(d(x', p')|x, p, a) = Q_n(dp'|p) \otimes \delta_{x+a}(dx')$, wobei $x \in [b^{\min}, b^{\max}],\ p \in \mathbb{R}^+,\ a \in D_n(x)$.

- Die Gewinnfunktion sei

$$h_n(p, a) = \begin{cases} -k(p) \cdot a, & a > 0 \\ 0, & a = 0 \\ -e(p) \cdot a, & a < 0. \end{cases}$$

  Hierbei ist $k(p) = (1 + w_1)p + z_1$ und $e(p) = (1 - w_2)p - z_2$ mit $w_1, w_2, z_1, z_2 \in \mathbb{R}^+$, sodass $k(p),\ e(p) \geq 0$ und $k(p) \geq e(p)$ für alle $p \in \mathcal{P}$.
  Die Funktion k beschreibt im Prinzip den Kaufpreis, das heißt den Preis, der bezahlt werden muss, um tatsächlich eine Einheit an Gas zu speichern. Die Funktion e beschreibt dementsprechend den Verkaufpreis.

- Die terminale Gewinnfunktion h_N sei zunächst beliebig, einige Beispiele zur Wahl der terminale Gewinnfunktion folgen in Beispiel 3.1.3.

Die Besonderheit an diesem stochastischen dynamischen Programm ist, dass der Übergang des Preises allein vom vorherigen Preis abhängt, nicht aber vom aktuellen Speicherstand oder der gewählten Aktion.
Das heißt letztlich ist der Übergangskern mit dem gearbeitet wird, der des Preisprozesses.

Bemerkung 3.1.1

a) In der allgemeinen Theorie Markovscher Entscheidungsprozesse ist $\tilde{Q}_n$ der stochastische Übergangskern von $\widetilde{D}_n$ nach E auf Stufe n, wobei $\widetilde{D}_n \subseteq [b^{\min}, b^{\max}] \times \mathcal{P} \times A$ mit

$$\widetilde{D}_n(x, p) = \{a \in A | \max(b^{\min} - x, i_n^{\min}(x)) \leq a \leq \min(b^{\max} - x, i_n^{\max}(x))\}$$

ist. Da jedoch $\widetilde{D}_n(x, p') = \widetilde{D}_n(x, p)$ für alle $p, p' \in \mathcal{P}$ ist, wird im Folgenden $D_n \subseteq [b^{\min}, b^{\max}] \times A$ mit $D_n(x)$ wie oben betrachtet.

b) Aufgrund der Definition des Übergangskerns gilt für v messbar und $(x,a) \in D_n,\ p \in \mathcal{P}$

$$\int v(x',p')\tilde{Q}_n(d(x'p')|x,p,a) = \int v(x+a,p')Q_n(dp'|p).$$

Die folgende Skizze (Abbildung 3.1) veranschaulicht beispielhaft die Restriktionenmenge.

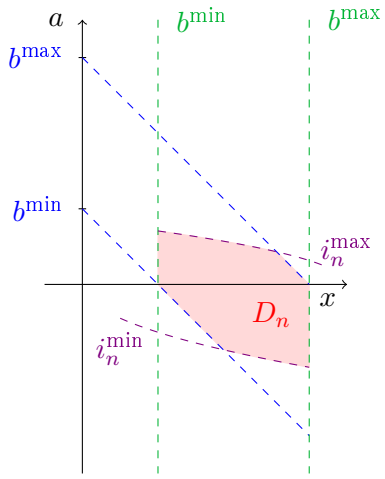


Abbildung 3.1.: Darstellung der Restriktionenmenge

Ist $(\boldsymbol{A_N})$ erfüllt und die Wertfunktion wie in Kapitel 2.1 definiert, so lassen sich die Operatoren wie folgt schreiben:

- $(L_n v)(x,p,a) = h_n(p,a) + \int v(x+a,p')Q_n(dp'|p),\ (x,a) \in D_n,\ p \in \mathcal{P}$
- $(T_n v)(x,p) = \sup_{a \in D_n(x)} (L_n v)(x,p,a),\ (x,p) \in E.$

Sei $\mathbb{E}$ der Erwartungswertoperator zur Markovkette (P_n), dann ist

$$\begin{aligned} T_n V_{n+1}(x,p) &= \sup_{a \in D_n(x)} \{h_n(p,a) + \mathbb{E}[V_{n+1}(x+a,P_{n+1})|P_n = p]\} \\ &= \max \left(\sup_{\max(b^{\min}-x, i_n^{\min}(x)) \le a \le 0} \{-e(p)a + \mathbb{E}[V_{n+1}(x+a,P_{n+1})|P_n = p]\}, \right. \\ &\qquad \left. \sup_{0 \le a \le \min(b^{\max}-x, i_n^{\max}(x))} \{-k(p)a + \mathbb{E}[V_{n+1}(x+a,P_{n+1})|P_n = p]\} \right). \end{aligned}$$

Zur Vereinfachung der Notation schreibt man oft $\mathbb{E}_p[f(P_{n+1})]$ für $\mathbb{E}[f(P_{n+1})|P_n = p]$.

Bemerkung 3.1.2
Aus Gründen der Lesbarkeit und Vereinfachung der Notation wird in dieser Arbeit meist auf eine Diskontierung verzichtet. Die Sätze lassen sich jedoch auf den Diskontierungsfall

übertragen, teilweise mit leicht veränderten Resultaten.

In Kapitel 4.3 wird eine Diskontierung eingeführt, um die Allgemeinheit der Aussagen zu verdeutlichen.

Die Algorithmen im zweiten Teil dieser Arbeit werden ebenfalls ohne Diskontierung vorgestellt, implementiert und analysiert. Auch sie lassen sich ohne Weiteres auf einen Diskontierungsfall übertragen.

In diesem Teilkapitel werden abschließend einige Beispiele für die Wahl terminaler Gewinnfunktionen angegeben, die in Spezialfällen beziehungsweise auch in der Anwendung verwendet werden.

Beispiel 3.1.3 (terminale Gewinnfunktionen)

a) Insbesondere in Kapitel 4.2.2 wird als terminale Gewinnfunktion h_N mit $h_N(x,p) = e(p)(x - b^{\min})$ verwendet. Das heißt am Ende wird im Prinzip das im Speicher verfügbare Gas „verkauft“.

b) Ist h_N durch

$$h_N(x,p) = h_n(p, x_0 - x) = \begin{cases} e(p)(x - x_0), & x \geq x_0 \\ -k(p)(x_0 - x), & x \leq x_0 \end{cases}$$

gegeben, so lässt sich diese terminale Gewinnfunktion wie folgt interpretieren:
Im Gasspeichervertrag ist ein Endzustand x_0 des Speichers festgelegt, dieser könnte zum Beispiel dem Anfangszustand entsprechen. Wird dieser unterschritten, so muss zur „Strafe“, der Betrag bezahlt werden, der nötig wäre, um die fehlende Menge an Gas zu kaufen. Wird der festgelegte Endzustand überschritten, so erhält man zu „Belohnung“ den Betrag, der durch Verkauf der Überflussmenge erzielt würde.

c) Alternativ zum Modell in b), könnte man die Belohnung auch streichen und hier den Gewinn $h_N(x,p) = 0$ setzen, falls $x \geq x_0$.

Eine weitere Alternative die „Strafkosten“ zu wählen, ist es, nicht proportional (mit $k(p)$) zur Entfernung zum geforderten Endzustand, sondern exponentiell oder gestuft proportional mit größer werdenden Faktoren (z.B. $k(p), 2k(p), 3k(p)$ usw.) vorzugehen.

3.2. Modifikationen des Hauptmodells

3.2.1. Das Gasspeicherproblem als optimales switching Problem

Bei einem optimalen switching Problem hat der Entscheider zu jedem Zeitpunkt die Möglichkeit entweder in einem Zustand zu verbleiben, also „nichts zu tun“, oder in einen zweiten Zustand zu wechseln (daher auch „switching“). Das System besteht aus zwei Zuständen.

Für das Gasspeicherproblem wird somit in diesem Spezialfall angenommen, dass es nur zwei Speicherstände gibt: den vollen Speicher ($b^{\max}$) und den leeren Speicher ($b^{\min}$). Der Zustandsraum vereinfacht sich somit zu

$$E = \{b^{\min}, b^{\max}\} \times \mathcal{P}.$$

Wie bereits beschrieben sind zu jedem Zeitpunkt nur zwei Entscheidungsmöglichkeiten gegeben, entweder man entscheidet sich den Speicher voll bzw. leer zu lassen oder man entscheidet sich dafür, den Speicher in den jeweils anderen Zustand zu versetzen. Folglich ist der Aktionenraum

$$A = \{0, 1\},$$

wobei 0 für „nichts tun“ und 1 für „Speicherzustand ändern“ steht.
Die zugrundeliegende Markovkette sei in diesem Fall außerdem gegeben durch

$$P_{n+1} = U(P_n, Y_{n+1}),$$

mit $Y_1, Y_2, \ldots$ unabhängig identisch verteilt.

Aufgrund der Änderung des Aktionenraums wird auch die Notation der Einstufengewinnfunktion geändert. Es bezeichne $h(b^{\max}, p, a)$ den Einstufengewinn im Zustand $(b^{\max}, p)$ bei Entscheidung a und $h(b^{\min}, p, a)$ entsprechend den Einstufengewinn im Zustand $(b^{\min}, p)$ bei Entscheidung a. Es gilt $h(b^{\max}, p, 0) = h(b^{\min}, p, 0) = 0$.
Da es nur zwei Zustände gibt, kann man die Wertefunktion auch vektorwertig schreiben. Es ist

$$\begin{pmatrix} V_n(b^{\max}, p) \\ V_n(b^{\min}, p) \end{pmatrix} = \begin{pmatrix} \max(h(b^{\max}, p, 1) + \mathbb{E}_p[V_{n+1}(b^{\min}, P_{n+1})], \mathbb{E}_p[V_{n+1}(b^{\max}, P_{n+1})]) \\ \max(h(b^{\min}, p, 1) + \mathbb{E}_p[V_{n+1}(b^{\max}, P_{n+1})], \mathbb{E}_p[V_{n+1}(b^{\min}, P_{n+1})]) \end{pmatrix}$$

Man erkennt hier sofort die optimale Politik

$$f_n^\star(b^{\max}, p) = \begin{cases} 1, & h(b^{\max}, p, 1) \geq \mathbb{E}_p[V_{n+1}(b^{\max}, P_n)] - \mathbb{E}_p[V_{n+1}(b^{\min}, P_n)], \\ 0, & h(b^{\max}, p, 1) < \mathbb{E}_p[V_{n+1}(b^{\max}, P_n)] - \mathbb{E}_p[V_{n+1}(b^{\min}, P_n)], \end{cases}$$

$$f_n^\star(b^{\min}, p) = \begin{cases} 1, & h(b^{\min}, p, 1) \geq \mathbb{E}_p[V_{n+1}(b^{\max}, P_n)] - \mathbb{E}_p[V_{n+1}(b^{\min}, P_n)], \\ 0, & h(b^{\min}, p, 1) < \mathbb{E}_p[V_{n+1}(b^{\max}, P_n)] - \mathbb{E}_p[V_{n+1}(b^{\min}, P_n)]. \end{cases}$$

3.2.2. Veränderungen in der Restriktionenmenge

In Kapitel 4.2 werden neben dem optimalen switching Problem auch weitere Spezialfälle betrachtet. Zum einen der Spezialfall des sogenannten schnellen Speichers, das heißt es ist zu jedem Zeitpunkt möglich, den Speicher komplett voll bzw. leer zu machen. Die Restriktionenmenge ist somit

$$D_n(x) = D(x) = \{a \in A | b^{\min} - x \leq a \leq b^{\max} - x\}.$$

Zum anderen wird der Fall angenommen, dass die Aktionen durch eine affine Funktion in x und nur durch diese begrenzt sind, das heißt es gelte $i_n^{\max}$ und $i_n^{\min}$ sind linear für alle n und es gelte zusätzlich $b^{\min} - x \leq i^{\min}(x)$ und $i^{\max}(x) \leq b^{\max} - x$. Die Restriktionenmenge ist somit

$$D_n(x) = \{a \in A | i_n^{\min}(x) \leq a \leq i_n^{\max}(x)\}.$$

Beispiele beider Fälle sind in folgender Skizze (Abbildung 3.2) dargestellt.

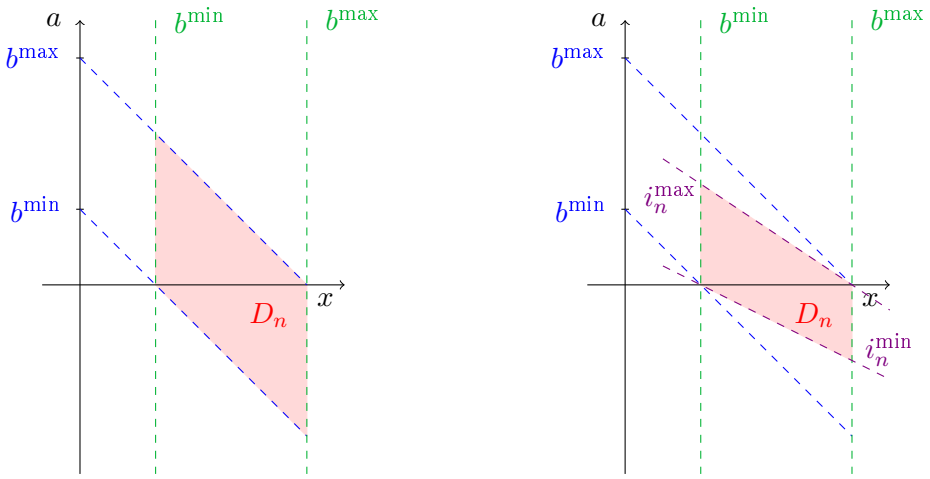


Abbildung 3.2.: Darstellung der Restriktionenmenge in den Vereinfachungen des schnellen Speichers (links) und der Begrenzung durch eine affine Funktion (rechts)

3.2.3. Veränderungen im Preisprozess

Vereinfachung des Preisprozesses

In Kapitel 4.2 wird folgende Vereinfachung des Preisprozesses betrachtet. Es seien $Y_1, Y_2, \ldots$ unabhängig identisch verteilte Zufallsvariablen mit $0 < \mathbb{E}[Y_1] =: \mu < \infty$ und

$$P_n = \prod_{k=1}^{n} Y_k.$$

Dann ist (P_n) eine Markovkette und es gilt

$$\mathbb{E}[P_n | P_{n-1} = p] = \mathbb{E}\left[\prod_{k=1}^{n} Y_k | P_{n-1} = p\right] = \mathbb{E}[P_{n-1} Y_n | P_{n-1} = p] = p\mathbb{E}[Y_n] = p\mu.$$

Verallgemeinerung des Preisprozesses

In Kapitel 4.3 wird das Modell erweitert auf einen Preisprozess mit regime switching. Es wird angenommen, dass eine Markovkette (R_n) mit endlichem Zustandsraum S_R den Preisprozess und damit das Modell beeinflusst.

Sei also (R_n) eine Markovsche Kette mit endlichem Zustandsraum S_R und Übergangsmatrix $(q_{jk})_{j,k\in S_R}$, sodass (P_n, R_n) Markovsch ist. Außerdem gelte

$$P_{n+1} \text{ und } R_{n+1} \text{ sind bedingt unabhängig unter } R_n, P_n, \tag{3.1}$$
$$\mathcal{L}(R_{n+1}|P_n, R_n) = \mathcal{L}(R_{n+1}|R_n). \tag{3.2}$$

Hierbei bezeichne $\mathcal{L}$ die bedingte Verteilung (engl. auch law) der jeweiligen Zufallsvariablen.

Aus diesen Eigenschaften folgt für den Übergangskern

$$\begin{aligned}
&\mathbb{P}(P_{n+1} \in B, R_{n+1} = k | R_n = j, P_n = p) \\
&\overset{(3.1)}{=} \mathbb{P}(P_{n+1} \in B | R_n = j, P_n = p)\mathbb{P}(R_{n+1} = k | R_n = j, P_n = p) \\
&\overset{(3.2)}{=} \mathbb{P}(P_{n+1} \in B | R_n = j, P_n = p)\mathbb{P}(R_{n+1} = k | R_n = j) \\
&= Q_{n,j}(B|p) \cdot q_{jk}.
\end{aligned}$$

Das stochastische dynamische Programm hat zudem einen anderen Zustandsraum, erweitert um eine dritte Komponente:

$$E = [b^{\min}, b^{\max}] \times \mathcal{P} \times S_R.$$

Elemente des Zustandsraum werden im Folgenden oft mit (x, p, r) oder (x, p, l) bezeichnet.

Die Wertfunktion ist nun für $(x, p, r) \in E$ gegeben durch

$$\begin{aligned}
V_N(x, p, r) &= h_N(x, p) \\
V_n(x, p, r) &= \sup_{a \in D_n(x)} \{h_n(p, a) + \mathbb{E}[V_{n+1}(x + a, P_{n+1}, R_{n+1})|P_n = p, R_n = r]\} \\
&= \sup_{a \in D_n(x)} \left\{ h_n(p, a) + \sum_{j \in S_R} q_{rj} \int V_{n+1}(x + a, p', j) Q_{n,r}(dp'|p) \right\}.
\end{aligned}$$

4 Strukturelle Analyse des Gasspeicherproblems

Mit Hilfe der mathematischen Modellierung des Gasspeicherproblems in Kapitel 3 lässt sich das Problem strukturell untersuchen. Dabei stellt sich die Frage, ob die Integrabilitäts- und Strukturannahmen erfüllt bzw. wann sie erfüllt sind. Ein weiteres interessantes Problem ist, ob es Fälle gibt oder Voraussetzungen existieren, sodass die optimale Strategie vom sogenannten „bang-bang“-Typ ist. Mit dem „bang-bang“-Typ werden Strategien bezeichnet, die wie folgt aufgebaut sind: Es gibt Grenzen p_1 und p_2 im Wertebereich des Preises $\mathcal{P}$, sodass die optimale Politik ist, unter der Grenze p_1 die maximal mögliche Gasmenge einzuspeichern, über der Grenze p_2 die maximal mögliche Gasmenge zu entnehmen und zwischen den Grenze weder einzuspeichern, noch zu entnehmen.

Der erste Teil dieses Kapitels beschäftigt sich mit der Frage, unter welchen weiteren Voraussetzungen die Annahmen erfüllt sind. Im zweiten wird die Struktur der optimalen Politik in einigen Spezialfällen untersucht, im dritten Teil wird schließlich in einer recht allgemeinen Situation die Struktur der optimalen Politik bestimmt.

4.1. Überprüfung der Integrabilitätsannahme und Strukturannahme

In diesem Kapitel soll mit Hilfe der Sätze aus Kapitel 2.2 zunächst überprüft werden, ob die Integrabilitätsannahme erfüllt ist. Danach werden Strukturaussagen mit Hilfe der Sätze aus 2.2.2 und 2.2.3 getroffen und weitere direkt gezeigt.

4.1.1. Integrabilitätsannahme, obere Schrankenfunktion, Schrankenfunktion

Die zwei folgenden Sätze zeigen, dass das Gasspeichermodell, vorgestellt in Kapitel 3.1, unter gewissen zusätzlichen Voraussetzungen an die terminale Gewinnfunktion und den Preisprozess eine obere Schrankenfunktion und sogar eine Schrankenfunktion besitzt.

Satz 4.1.1
Gegeben sei das Gasspeichermodell aus Kapitel 3.1. Für die terminale Gewinnfunktion und den Preisprozess gelte mit $c_2, c_3 \in \mathbb{R}^+$

- $h_N^+(x,p) \leq c_2 \cdot p$,
- $\mathbb{E}[P_{n+1}|P_n = p] \leq c_3 \cdot p$.

Dann ist s mit $s(x,p) = p$ eine obere Schrankenfunktion für das stochastische dynamische Programm.

Beweis
Nach Definition 2.2.1 ist s mit $s(x,p) = p$ eine obere Schrankenfunktion, falls $c_1, c_2, c_3 \in \mathbb{R}^+$ exisitieren, sodass für alle $n = 0, 1, \ldots N-1$:

(i) $h_n^+(p,a) \leq c_1 p$ für alle $(x,a) \in D_n$

(ii) $h_N^+(x,p) \leq c_2 p$ für alle $(x,p) \in E$

(iii) $\int p' Q_n(dp'|p) \leq c_3 p$ für alle $(x,a) \in D_n$.

Die beiden Forderungen an die terminale Gewinnfunktion und den Preisprozess sind gerade (ii) und (iii). Es bleibt demnach (i) zu zeigen.
Für $(x,a) \in D_n$ gilt

$$\begin{aligned} h_n^+(p,a) &\leq \max_{a \leq 0, a \in D_n(x)} \{-e(p)a\} = \max_{\max\{b^{\min}-x, i_n^{\min}(x)\} \leq a \leq 0} \{-e(p)a\} \\ &= e(p) \cdot \max\{-(b^{\min}-x), -i_n^{\min}(x)\} = e(p)\max\{x - b^{\min}, -i_n^{\min}(x)\} \\ &\leq e(p)(\max\{b^{\max} - b^{\min}, -i_n(b^{\max})\}) \leq e(p)c = c((1-w_2)p - z_2) \\ &\leq \underbrace{c(1-w_2)}_{\geq 0} p \leq c_1 p. \end{aligned}$$

Somit ist s eine obere Schrankenfunktion. ∎

Satz 4.1.2
Gegeben sei das Gasspeichermodell aus Kapitel 3.1. Für die terminale Gewinnfunktion und den Preisprozess gelte mit $c_2, c_3 \in \mathbb{R}^+$

- $|h_N(x,p)| \leq c_2 \cdot k(p)$,
- $\mathbb{E}[k(P_{n+1})|P_n = p] \leq c_3 \cdot k(p)$.

Dann ist s mit $s(x,p) = k(p)$ eine Schrankenfunktion für das stochastische dynamische Programm.

Beweis
Nach Definition 2.2.1 ist s mit $s(x,p) = k(p)$ eine obere Schrankenfunktion, falls $c_1, c_2, c_3 \in \mathbb{R}^+$ exisitieren, sodass für alle $n = 0, 1, \ldots N-1$:

(i) $|h_n(p,a)| \leq c_1 k(p)$ für alle $(x,a) \in D_n$

(ii) $|h_N(x,p)| \leq c_2 k(p)$ für alle $(x,p) \in E$

(iii) $\int k(p')Q_n(dp'|p) \leq c_3 k(p)$ für alle $(x,a) \in D_n$.

Die beiden Forderungen an die terminale Gewinnfunktion und den Preisprozess sind gerade (ii) und (iii). Es bleibt demnach (i) zu zeigen.
Zum einen gilt für $(x,a) \in D_n$

$$\begin{aligned} h_n(p,a) &\leq \max_{a\leq 0, a\in D_n(x)}\{-e(p)a\} = \max_{\max\{b^{\min}-x, i_n^{\min}(x)\}\leq a\leq 0}\{-e(p)a\} \\ &= e(p)\cdot\max\{-(b^{\min}-x), -i_n^{\min}(x)\} = e(p)\max\{x-b^{\min}, -i_n^{\min}(x)\} \\ &\leq e(p)\max\{b^{\max}-b^{\min}, -i_n^{\min}(b^{\max})\} = e(p)\hat{c}_1 \\ &\leq \hat{c}_1 k(p) \end{aligned}$$

Zum anderen gilt für $(x,a) \in D_n$

$$\begin{aligned} h_n(p,a) &\geq \min_{a\geq 0, a\in D_n(x)}\{-k(p)a\} = \min_{\min\{b^{\max}-x, i_n^{\max}(x)\}\geq a\geq 0}\{-k(p)a\} \\ &= k(p)\cdot\min\{-(b^{\max}-x), -i_n^{\max}(x)\} = k(p)\min\{x-b^{\max}, -i_n^{\max}(x)\} \\ &\geq k(p)\min\{b^{\min}-b^{\max}, -i_n^{\max}(b^{\min})\} \\ &= \bar{c}_1 k(p) \end{aligned}$$

Insgesamt gilt also

$$|h(p,a)| \leq c_1 k(p)$$

mit $c_1 = \max\{|\hat{c}_1|, |\bar{c}_1|\}$.
Somit ist s eine Schrankenfunktion. ∎

Bemerkung 4.1.3
Unter den Voraussetzungen $|h_N(x,p)| \leq \tilde{c}_2(k(p)+\gamma)$ und $\mathbb{E}[k(P_{n+1})+\gamma|P_n = p] \leq \tilde{c}_3(k(p)+\gamma)$, kann auch nachgewiesen werden, dass $k(p)+\gamma$ für ein festes $\gamma \in \mathbb{R}^+$ ebenso eine Schrankenfunktion für das stochastische dynamische Programm ist.

Diese Aussage wird benötigt, wenn sich die Voraussetzungen von Satz 4.1.2 nicht nachweisen lassen.

Hat man bereits $|h_N(x,p)| \leq \tilde{c}_2 k(p)$ gezeigt, so gilt auch $|h_N(x,p)| \leq \tilde{c}_2(k(p)+\gamma)$, genauso folgt aus $|h_n(p,a)| \leq \tilde{c}_1 k(p)$ die Ungleichung $|h_n(p,a)| \leq \tilde{c}_1(k(p)+\gamma)$.

Es folgen Beispiele, bei denen die Forderungen in Satz 4.1.2 bzw. Bemerkung 4.1.3 erfüllt sind.

Beispiel 4.1.4

a) Beispiele für die terminale Gewinnfunktion:

(i) Ist $h_N(x,p) = e(p)x$, so gilt

$$|h_N(x,p)| = e(p)x \leq k(p)x \leq k(p)b^{\max} =: c_2 k(p)$$

(ii) Ist $h_N(x,p) = h_n(p, x_0 - x)$ für zulässiges $x_0 \in [b^{\min}, b^{\max}]$, so gilt im Fall $x_0 \leq x$

$$|h_N(x,p)| = e(p)(x - x_0) \leq k(p)(x - x_0) \leq k(p)(b^{\max} - x_0) =: c_2 k(p)$$

und im Fall $x_0 \geq x$

$$|h_N(x,p)| = k(p)(x_0 - x) \leq k(p)(x_0 - b^{\min}) =: c_2 k(p).$$

b) Beispiele für den Preisprozess:

(i) Betrachtet man eine Markov-Kette (P_n), wie in Kapitel 3.2.3 vorgestellt, das heißt $P_n = \prod_{i=1}^{n} Y_i$ mit Y_i unabhängig, identisch verteilt und $0 < \mathbb{E}[Y_i] =: \mu < \infty$, so ergibt sich

$$\mathbb{E}[k(P_{n+1})|P_n = p] = \mathbb{E}[k(Y_{n+1}p)] = k(\mu p) = \mu(1 + w_1)p + z_1$$

Da $\mu, z_1, p \in \mathbb{R}^+$ existiert eine Konstante $c \in \mathbb{R}^+$, sodass $z_1 \leq c z_1 \mu$ und $p \leq cp$. Damit erhält man

$$\mathbb{E}[k(P_{n+1})|P_n = p] \leq c\mu((1 + w_1)p + z_1) =: c_3 k(p).$$

(ii) Sei $(P_t)_{t \geq 0}$ das Preismodell aus Kapitel 5.1, das heißt

$$\log P_t = e^{-\alpha t}(p_0 + \int_0^t \alpha e^{\alpha x}\mu(x)dx + \int_0^t e^{\alpha x}\sigma(x)dB_x),$$

wobei $(B_t)_{t \geq 0}$ eine Brownsche Bewegung ist. Wie in Kapitel 5.1 Bemerkung 5.1.8 gezeigt wird, gilt für $n \in \{0, 1, \ldots, N-1\}$ und eine Konstante $c(n)$, die von n abhängt,

$$\mathbb{E}[P_{n+1}|P_n = p] = c(n) \cdot p^{e^{-\alpha}} \leq p^{e^{-\alpha}} \cdot \max_{0 \leq n \leq N-1} c(n) =: \bar{c} \cdot p^{e^{-\alpha}}.$$

Hier wird $k(p) + 1$ als Schrankenfunktion gewählt. Somit ist

$$\begin{aligned}\mathbb{E}[k(P_{n+1}) + 1|P_n = p] &= (1 + w_1)\mathbb{E}[P_{n+1}|P_n = p] + z_1 + 1 \\ &\leq (1 + w_1)\bar{c} \cdot p^{e^{-\alpha}} + z_1 + 1.\end{aligned}$$

Für $p \geq 1$ gilt, da $\alpha > 0$

$$\begin{aligned}\mathbb{E}[k(P_{n+1}) + 1|P_n = p] &\leq (1 + w_1)\bar{c} \cdot p^{e^{-\alpha}} + z_1 + 1 \\ &\leq \bar{c}(1 + w_1)p + z_1 + 1 \\ &\leq \tilde{c}(1 + w_1)p + \tilde{c}(z_1 + 1) \\ &= \tilde{c}((1 + w_1)p + z_1 + 1) = \tilde{c}(k(p) + 1),\end{aligned}$$

wobei $\tilde{c} \geq 1$ und $\tilde{c} \geq \bar{c}$ geeignet. Für $0 < p < 1$ gilt

$$\begin{aligned}\mathbb{E}[k(P_{n+1}) + 1 | P_n = p] &\leq (1 + w_1)\bar{c} \cdot p^{e^{-\alpha}} + z_1 + 1 \\ &\leq \bar{c}(1 + w_1) + z_1 + 1 \\ &\leq \bar{c}(1 + w_1) + z_1 + 1 + (1 + w_1)p \\ &\leq \hat{c}(z_1 + 1) + \hat{c}(1 + w_1)p \\ &= \hat{c}((1 + w_1)p + z_1 + 1) = \hat{c}(k(p) + 1),\end{aligned}$$

wobei $\hat{c} \geq 1$ geeignet gewählt wird, sodass $\bar{c}(1 + w_1) + z_1 + 1 \leq \hat{c}(z_1 + 1)$. Wird nun $c_3 := \max(\tilde{c}, \hat{c})$ gewählt, so gilt das Gewünschte.

4.1.2. Strukturannahme: Stetigkeit, Konvexität und weitere Eigenschaften

Zunächst wird gezeigt, dass unter wenigen Zusatzvoraussetzung die Sätze aus Kapitel 2.2.3 und 2.2.2 gelten.

Im Gasspeichermodell überträgt sich die Stetigkeit von v auf $T_n v$.

Satz 4.1.5
Gegeben sei das Gasspeichermodell aus Kapitel 3.1. Es gelte außerdem $i_n^{\max}$, $i_n^{\min}$ stetig für alle n und die Voraussetzungen für die Existenz einer Schrankenfunktion.
Ist $v \in \mathbb{S}_s$ und $x \mapsto v(x, p)$ stetig, so gilt $x \mapsto T_n v(x, p)$ ist stetig für jedes feste $p \in \mathcal{P}$ und es existiert ein Maximisator f_n von v.

Beweis
Die Aussage folgt analog zu Satz 2.2.5, indem die Voraussetzungen geprüft werden, was nun ausgeführt wird.

(i) Offensichtlich ist $D_n(x)$ kompakt für alle $x \in E$.
Es gilt

$$\begin{aligned}D_n(x) &= [\max\{b^{\min} - x, i_n^{\min}(x)\}, \min\{b^{\max} - x, i_n^{\max}(x)\}] \\ &=: [\underline{d}(x), \overline{d}(x)].\end{aligned}$$

Sei $x_k \to x$ $(k \to \infty)$, dann gilt zum einen

$$\lim_{k\to\infty} \overline{d}(x_k) = \lim_{k\to\infty} \min\{b^{\max} - x_k, i_n^{\max}(x_k)\} = \min\{b^{\max} - x, i_n^{\max}(x)\}$$

und zum andern

$$\lim_{k\to\infty} \underline{d}(x_k) = \lim_{k\to\infty} \max\{b^{\min} - x_k, i_n^{\min}(x_k)\} = \lim_{k\to\infty} \max\{b^{\min} - x, i_n^{\min}(x)\}.$$

Diese Aussagen gelten, da $i_n^{\max}, i_n^{\min}$ stetig sind und es gilt: Sind f, g stetig, so ist $\max\{f(x), g(x)\} = \frac{1}{2}(f(x) + g(x) + |f(x) - g(x)|)$, sowie $\min\{f(x), g(x)\} = \frac{1}{2}(f(x) + g(x) - |f(x) - g(x)|)$ ebenfalls stetig.

(ii) Zu zeigen ist $(x,a) \mapsto L_n v(x,p,a)$ stetig auf D_n.
Laut Voraussetzung ist v stetig und in $\mathbb{S}_s$, außerdem ist $a \mapsto h_n(p,a)$ stetig.
Sei $(x_k, a_k) \to (x,a)$, dann gilt

$$\begin{aligned}\lim_{k\to\infty} L_n v(x_k,p,a_k) &= \lim_{k\to\infty}\left(h_n(p,a_k) + \int v(x_k+a_k,p')Q_n(dp'|p)\right)\\ &= \lim_{k\to\infty} h_n(p,a_k) + \lim_{k\to\infty}\int v(x_k+a_k,p')Q_n(dp'|p)\\ &\overset{dom.Konv}{=} h_n(p,a) + \int \lim_{k\to\infty} v(x_k+a_k,p')Q_n(dp'|p)\\ &\overset{v\ stetig}{=} h_n(p,a) + \int v(x+a,p')Q_n(dp'|p) = L_n v(x,p,a)\end{aligned}$$

Die Vertauschung des Grenzwerts mit dem Integral ist möglich, weil eine Schrankenfunktion existiert, daher existieren auch Grenzwerte.
Eine genauere Betrachtung zur Anwendung des Satzes von der dominierten Konvergenz in diesem Fall ist wie folgt:
Es ist $v \in \mathbb{S}_s = \{v \in \mathbb{M}(E) || v(x,p)| \leq c \cdot s(x,p)\ \forall\ (x,p) \in E\}$, wobei hier $s(x,p) = k(p)$, also ist $|v(x_k+a_k,p')| \leq ck(p')$ und da s obere Schrankenfunktion ist gilt, $\int k(p')Q_n(dp'|p) \leq c_3 k(p) < \infty$. ∎

Bemerkung 4.1.6

a) Gilt in Satz 4.1.5 zusätzlich, dass $x \mapsto h_N(x,p)$ stetig, so gilt auch Korollar 2.2.6.

b) Da konkave beziehungsweise konvexe Funktionen im Innern ihres Definitionsbereiches stetig sind, ist die geforderte Stetigkeit in Satz 4.1.5 lediglich eine Forderung an die Randpunkte des Definitionsbereichs von $i_n^{\max}$ und $i_n^{\min}$.

Eine Aussage, die im weiteren Verlauf der Arbeit noch von Bedeutung ist, ist die der Konkavität.

Allerdings ist $(x,p,a) \mapsto L_n v(x,p,a)$ nicht konkav im Gasspeichermodell, da die Abbildung $(p,a) \mapsto h_n(p,a)$ nicht konkav ist. Dennoch kann bei festen p folgender Satz gezeigt werden, der für das weitere Vorgehen relevant ist.

Satz 4.1.7
Gegeben sei das Gasspeichermodell aus Kapitel 3.1. Es gelte außerdem $i_n^{\max}$ konkav und $i_n^{\min}$ konvex für alle n und die Voraussetzungen für die Existenz einer oberen Schrankenfunktion.
Ist $v \in \mathbb{S}_s^+$ und $x \mapsto v(x,p)$ konkav, so ist $x \mapsto T_n v(x,p)$ konkav auf $\mathbb{R}^+$ für jedes feste $p \in \mathcal{P}$.

Beweis
Die Behauptung folgt analog zu Satz 2.2.8, indem die Voraussetzungen nachgeprüft werden.

(i): Sei $(x,a),(\hat{x},\hat{a}) \in D_n$ und $\lambda \in [0,1]$. Dann gilt zum einen

1. $b^{\min} \overset{(2)}{\leq} x+a \overset{(1)}{\leq} b^{\max}$ und $b^{\min} \overset{(2)}{\leq} \hat{x}+\hat{a} \overset{(1)}{\leq} b^{\max}$ und somit

$$\begin{aligned}\lambda x + (1-\lambda)\hat{x} + \lambda a + (1-\lambda)\hat{a} &= \lambda(x+a) + (1-\lambda)(\hat{x}+\hat{a}) \\ &\overset{(1)}{\leq} \lambda b^{\max} + (1-\lambda) b^{\max} = b^{\max} \ \checkmark\end{aligned}$$

bzw.

$$\overset{(2)}{\geq} \lambda b^{\min} + (1-\lambda) b^{\min} = b^{\min} \ \checkmark$$

Zum anderen gilt

2. $i^{\min}(x) \leq a \leq i^{\max}(x)$ und $i^{\min}(\hat{x}) \leq \hat{a} \leq i^{\max}(\hat{x})$ und somit

$$\begin{aligned} i^{\min}(\lambda x + (1-\lambda)\hat{x}) &\overset{i^{\min} konvex}{\leq} \lambda i^{\min}(x) + (1-\lambda) i^{\min}(\hat{x}) \\ &\leq \lambda a + (1-\lambda)\hat{a} \\ &\leq \lambda i^{\max}(x) + (1-\lambda) i^{\max}(\hat{x}) \\ &\overset{i^{\max} konkav}{\leq} i^{\max}(\lambda x + (1-\lambda)\hat{x}) \ \checkmark. \end{aligned}$$

Insgesamt gilt also $(\lambda x + (1-\lambda)\hat{x}, \lambda a + (1-\lambda)\hat{a}) \in D_n$.

(ii): Es wird nur $(x,a) \mapsto L_n v(x,p,a)$ (d.h. p fest) betrachtet. Zu zeigen ist

1. $a \mapsto h_n(p,a)$ ist konkav
2. $(x,a) \mapsto \int v(x+a,p') Q_n(dp'|p)$ ist konkav

zu 1. Es gilt $k(p) \geq e(p)$ bzw. $-k(p) \leq -e(p)$. Sei $a, \hat{a} \in D_n$.

Für $a, \hat{a} \leq 0$ bzw. $a, \hat{a} \geq 0$ gilt die Aussage, da dann $\lambda a + (1-\lambda)\hat{a} \leq 0$ bzw. ≥ 0 und $a \mapsto h_n(p,a)$ stückweise linear ist.

Sei also oBdA $a \leq 0$ und $\hat{a} \geq 0$. Dann gilt

$$\begin{aligned} h_n(p, \lambda a + (1-\lambda)\hat{a}) =& -k(p)(\lambda a + (1-\lambda)\hat{a}) \mathbf{1}_{\{\lambda a + (1-\lambda)\hat{a} > 0\}} \\ & - e(p)(\lambda a + (1-\lambda)\hat{a}) \mathbf{1}_{\{\lambda a + (1-\lambda)\hat{a} < 0\}} \\ =& \lambda \Big(\underbrace{-k(p)a}_{\geq -e(p)a} 1_{\{\lambda a + (1-\lambda)\hat{a} > 0\}} - e(p) a \mathbf{1}_{\{\lambda a + (1-\lambda)\hat{a} < 0\}} \Big) \\ & + (1-\lambda)\Big(-k(p)\hat{a} \mathbf{1}_{\{\lambda a + (1-\lambda)\hat{a} > 0\}} \underbrace{-e(p)\hat{a}}_{\geq -k(p)\hat{a}} \mathbf{1}_{\{\lambda a + (1-\lambda)\hat{a} < 0\}} \Big) \\ \geq & (-\lambda e(p) a - (1-\lambda) k(p) \hat{a}) \mathbf{1}_{\{\lambda a + (1-\lambda)\hat{a} \neq 0\}} \\ \overset{(\star)}{\geq} & \lambda(-e(p)a) + (1-\lambda)(-k(p)\hat{a}) \\ =& \lambda h_n(p,a) + (1-\lambda) h_n(p,\hat{a}). \end{aligned}$$

Die Ungleichung $(\star)$ gilt, da aus $0 = \lambda a + (1-\lambda)\hat{a}$ folgt

$$\begin{aligned} 0 &= -e(p)(\lambda a + (1-\lambda)\hat{a}) = \lambda(-e(p)a) + (1-\lambda)(-e(p)\hat{a}) \\ &\geq \lambda(-e(p)a) + (1-\lambda)(-k(p)\hat{a}). \end{aligned}$$

zu 2. Seien $(x,a),(\hat{x},\hat{a}) \in D_n$ und $\lambda \in [0,1]$. Es ist

$$\begin{aligned} &\int v(\lambda x + (1-\lambda)\hat{x} + \lambda a + (1-\lambda)\hat{a}, p')Q_n(dp'|p) \\ &= \int v(\lambda(x+a) + (1-\lambda)(\hat{x}+\hat{a}), p')Q_n(dp'|p) \\ &\geq \int \lambda v(x+a,p') + (1-\lambda)v(\hat{x}+\hat{a},p')Q_n(dp'|p) \\ &= \lambda \int v(x+a,p')Q_n(dp'|p) + (1-\lambda)\int v(\hat{x}+\hat{a},p')Q_n(dp'|p). \end{aligned}$$

Es folgt aus 1. und 2., dass $(x,a) \mapsto L_n v(x,p,a)$ konkav ist. ∎

Einige dieser Eigenschaften lassen sich auch ohne die Sätze aus Kapitel 2 zeigen, zudem ergeben sich zusätzliche Eigenschaften.

Satz 4.1.8
Gilt $x \mapsto i_n^{\max}(x)$ ist langsamer fallend als $x \mapsto b^{\max} - x$ vor dem Schnittpunkt der beiden Abbildungen, dann gilt:
Ist $x \mapsto v_{n+1}(x,p)$ monoton wachsend, so ist auch $x \mapsto v_n(x,p)$ monoton wachsend.

Beweis
Seien $x_1 \leq x_2$ zulässig und $\varepsilon > 0$. Dann exisitert ein $a_1 \in D_n(x_1)$, sodass

$$Lv_{n+1}(x_1,p,a_1) \geq T_n v_{n+1}(x_1,p) - \varepsilon$$

<u>1. Fall:</u> $a_1 \in D_n(x_2)$

Dann folgt

$$\begin{aligned} Tv_{n+1}(x_2,p) - Tv_{n+1}(x_1,p) &\geq Lv_{n+1}(x_2,p,a_1) - Lv_{n+1}(x_1,p,a_1) - \varepsilon \\ &= \int \underbrace{v_{n+1}(x_2+a_1,p') - v_{n+1}(x_1+a_1,p')}_{\geq 0} Q_n(dp'|p) - \varepsilon \\ &\geq -\varepsilon \end{aligned}$$

<u>2. Fall:</u> $a_1 \notin D_n(x_2)$
Damit folgt aus der Struktur der Resriktionenmenge (vgl. Abbildung 4.1) $a_1 \geq a$ für alle $a \in D_n(x_2)$.

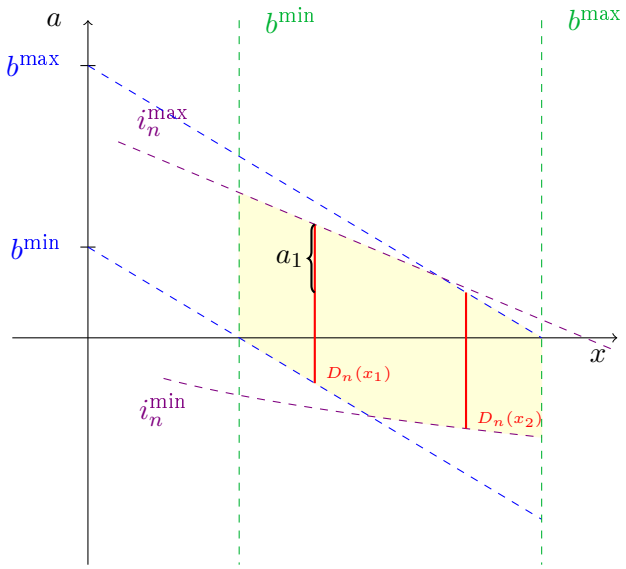


Abbildung 4.1.: Struktur der Restriktionenmenge

Sei nun $a_2 := \min(b^{\max} - x_2, i_n^{\max}(x_2))$. Dann gilt

$$\begin{aligned} a_1 - a_2 &\leq \max_{a \in D_n(x_1) \setminus D_n(x_2)} (a - a_2) = \min(b^{\max} - x_1, i_n^{\max}(x_1)) - \min(b^{\max} - x_2, i_n^{\max}(x_2)) \\ &= \begin{cases} x_2 - x_1, & b^{\max} - x_1 \leq i_n^{\max}(x_1) \text{ und } b^{\max} - x_2 \leq i_n^{\max}(x_2), \\ i_n^{\max}(x_1) - b^{\max} + x_2, & b^{\max} - x_1 \geq i_n^{\max}(x_1) \text{ und } b^{\max} - x_2 \leq i_n^{\max}(x_2), \\ i_n^{\max}(x_1) - i_n^{\max}(x_2), & b^{\max} - x_1 \geq i_n^{\max}(x_1) \text{ und } b^{\max} - x_2 \geq i_n^{\max}(x_2). \end{cases} \end{aligned}$$

Damit ist

$$\begin{aligned} &L_n v_{n+1}(x_1, p, a_1) - L_n v_{n+1}(x_2, p, a_2) \\ &= \underbrace{h_n(a_1, p) - h_n(a_2, p)}_{\leq 0, \text{ da } h_n \text{ fallend in } a} + \int \underbrace{v_{n+1}(x_1 + a_1, p') - v_{n+1}(x_2 + a_2, p')}_{\leq 0, \text{ da } a_1 + x_1 \leq x_2 + a_2 (\star)} Q_n(dp'|p) \\ &\leq 0. \end{aligned}$$

Erklärung zu $(\star)$:
Es gilt

$$\begin{aligned} x_1 + a_1 - x_2 - a_2 &= x_1 - x_2 + a_1 - a_2 \\ &\leq \begin{cases} x_1 - x_2 + x_2 - x_1 = 0, & (1) \\ x_1 - x_2 + i_n^{\max}(x_1) - b^{\max} + x_2 = i_n^{\max}(x_1) - (b^{\max} - x_1) \leq 0, & (2) \\ x_1 - x_2 + i_n^{\max}(x_1) - i_n^{\max}(x_2) \overset{(\star\star)}{\leq} x_1 - x_2 + x_2 - x_1 = 0, & (3) \end{cases} \end{aligned}$$

Erklärung zu $(\star\star)$:
Hierbei ist aufgrund der Voraussetzung gewährleistet, dass bis zum Schnittpunkt von $i_n^{\max}$ und $b^{\max} - x$ die Funktion $i_n^{\max}$ langsamer fällt, d.h. die Sekantensteigung größer

ist. Daher gilt

$$0 \geq \frac{i_n^{\max}(x_2) - i_n^{\max}(x_1)}{x_2 - x_1} \geq \frac{b^{\max} - x_2 - b^{\max} + x_1}{x_2 - x_1} = -1$$

woraus

$$i_n^{\max}(x_1) - i_n^{\max}(x_2) \leq x_2 - x_1$$

folgt. Insgesamt gilt somit im Fall 2

$$\begin{aligned} Tv_{n+1}(x_2, p) - Tv_{n+1}(x_1, p) &\geq Lv_{n+1}(x_2, p, a_2) - Lv_{n+1}(x_1, p, a_1) - \varepsilon \\ &\geq -\varepsilon \end{aligned}$$

Die Behauptung folgt nun mit $\varepsilon \searrow 0$. ∎

Auch die Tatsache, dass sich die Konkavität, sowie die Stetigkeit von v_{n+1} auf v_n überträgt, lässt sich direkt zeigen.

Satz 4.1.9
Seien $i_n^{\max}$ konkav und $i_n^{\min}$ konvex, dann gilt:
Ist $x \mapsto v_{n+1}(x, p)$ konkav, so ist auch $x \mapsto v_n(x, p)$ konkav.

Beweis
Seien x_1, x_2 zulässig und $x = \alpha x_1 + (1-\alpha)x_2$, $\alpha \in [0, 1]$. Sei $\varepsilon > 0$. Dann existieren $a_1 \in D_n(x_1)$ und $a_2 \in D_n(x_2)$ mit

$$\begin{aligned} Lv_{n+1}(x_1, p, a_1) &\geq T_n v_{n+1}(x_1, p) - \varepsilon \\ Lv_{n+1}(x_2, p, a_2) &\geq T_n v_{n+1}(x_2, p) - \varepsilon \end{aligned}$$

Es gilt außerdem $\alpha a_1 + (1-\alpha)a_2 \in D_n(x) = D_n(\alpha x_1 + (1-\alpha)x_2)$, denn

$$\alpha a_1 + (1-\alpha)a_2 \leq \alpha(b^{\max} - x_1) + (1-\alpha)(b^{\max} - x_2) = b^{\max} - (\alpha x_1 + (1-\alpha)x_2)$$

bzw.

$$\alpha a_1 + (1-\alpha)a_2 \geq b^{\min} - (\alpha x_1 + (1-\alpha)x_2)$$

und da $i_n^{\max}$ konkav (1) und $i_n^{\min}$ konvex (2)

$$\alpha a_1 + (1-\alpha)a_2 \leq \alpha i^{\max}(x_1) + (1-\alpha)i^{\max}(x_2) \overset{(1)}{\leq} i^{\max}(\alpha x_1 + (1-\alpha)x_2)$$

bzw.

$$\alpha a_1 + (1-\alpha)a_2 \geq \alpha i^{\min}(x_1) + (1-\alpha)i^{\min}(x_2) \overset{(2)}{\geq} i^{\min}(\alpha x_1 + (1-\alpha)x_2).$$

Damit gilt

$$\begin{aligned}
v_n(x,p) = \quad & T_n v_{n+1}(\alpha x_1 + (1-\alpha)x_2, p) \\
= \quad & \sup_{a \in D_n(\alpha x_1 + (1-\alpha)x_2)} L v_{n+1}(\alpha x_1 + (1-\alpha)x_2, p, a) \\
\geq \quad & L v_{n+1}(\alpha x_1 + (1-\alpha)x_2, p, \alpha a_1 + (1-\alpha)a_2) \\
= \quad & h_n(p, \alpha a_1 + (1-\alpha)a_2) \\
& + \int v_{n+1}(\alpha x_1 + (1-\alpha)x_2 + \alpha a_1 + (1-\alpha)a_2, p') Q_n(dp'|p) \\
\overset{h_n \text{ konkav}}{\geq} \quad & \alpha h_n(p, a_1) + (1-\alpha) h_n(p, a_2) \\
& + \int v_{n+1}(\alpha(x_1 + a_1) + (1-\alpha)(x_2 + a_2), p') Q_n(dp'|p) \\
\overset{Vor}{\geq} \quad & \alpha h_n(p, a_1) + (1-\alpha) h_n(p, a_2) \\
& + \int \alpha v_{n+1}(x_1 + a_1, p') + (1-\alpha) v_{n+1}(x_2 + a_2, p') Q_n(dp'|p) \\
= \quad & \alpha L v_{n+1}(x_1, p, a_1) + (1-\alpha) L v_{n+1}(x_2, p, a_2) \\
\geq \quad & \alpha(T_n v_{n+1}(x_1, p) - \varepsilon) + (1-\alpha)(T_n v_{n+1}(x_2, p) - \varepsilon) \\
= \quad & \alpha v_n(x_1, p) + (1-\alpha) v_n(x_2, p) - \varepsilon
\end{aligned}$$

Die Behauptung folgt mit $\varepsilon \searrow 0$. ∎

Satz 4.1.10
Seien $i_n^{\max}$ stetig und $i_n^{\min}$ stetig, dann gilt:
Ist $x \mapsto v_{n+1}(x,p)$ stetig, so ist auch $x \mapsto v_n(x,p)$ stetig.

Beweis
Seien $w(x,p,a) := L v_{n+1}(x,p,a)$ und $w^\star(x,p) := T_n v_{n+1}(x,p)$. Sei (x_n) eine Folge mit x_n zulässig und $x_n \to x_0$, sodass $\lim_{n\to\infty} w^\star(x_n, p)$ existiert.

Zunächst wird gezeigt, dass $(x,a) \mapsto w(x,p,a)$ und $a \mapsto w(x,p,a)$ stetig. Es ist

$$w(x,p,a) = h_n(p,a) + \int v_{n+1}(x+a, p') Q(dp'|p).$$

Da $a \mapsto h_n(p,a)$ stetig und $x \mapsto v_{n+1}(x,p)$ stetig nach Voraussetzung, ist auch $(x,a) \mapsto v_{n+1}(x+a,p)$ stetig. Damit ist dann $(x,a) \mapsto w(x,p,a)$ stetig, falls

$$\lim_{n\to\infty} \int v(x_n + a_n, p') Q(dp'|p) = \int \lim_{n\to\infty} v(x_n + a_n, p') Q(dp'|p).$$

Analog für $a \mapsto w(x,p,a)$. Mit dem Satz von der monotonen Konvergenz argumentiert man analog zum Beweis von Satz 4.1.5, dass die Aussage richtig ist.

Damit - $a \mapsto w(x,p,a)$ stetig - nimmt w sein Supremum auf $D_n(x)$ an. Seien also a_n Maximumpunkte von $a \mapsto w(x_n,p,a)$ auf $D_n(x_n)$. Da $x \mapsto D_n(x)$ stetig (dies gilt falls $i_n^{\max}$ stetig und $i_n^{\min}$ stetig), existiert eine Teilfolge (a_{n_k}) mit $a_{n_k} \to a_0 \in D_n(x_0)$. Damit ist

$$\lim_{n\to\infty} w^\star(x_n,p) = \lim_{k\to\infty} w^\star(x_{n_k},p) = \lim_{k\to\infty} w(x_{n_k},p,a_{n_k}) \overset{stetig}{=} w(x_0,p,a_0) = w^\star(x_0,p)$$

und $x \mapsto w^\star(x,p)$ stetig. ∎

Neben den bereits gezeigten Strukturaussagen, die auch in Kapitel 2.2 schon erwähnt werden, lässt sich für das Gasspeicherproblem auch eine erste Eigenschaft der Maximisatoren zeigen. Die Maximisatoren auf Stufe n sind fallend im Speicherstand, das heißt, ist mehr Gas im Speicher so ist die optimale Aktion kleiner, man wird demnach weniger Gas einspeichern oder gar Gas aus dem Speicher entnehmen. Ist jedoch weniger Gas im Speicher, so wird der neue Speicherzustand, der nach dem Entnehmen bzw. Befüllen mit Gas entsteht, immer noch geringer sein, als der der mit mehr Gas im Speicher erreicht worden wäre.

Satz 4.1.11
Es seien die Voraussetzungen von Satz 4.1.9 erfüllt.
Für $x_1 \le x_2$ gilt

a) $f_n^\star(x_1) \ge f_n^\star(x_2)$, das heißt der Maximisator ist fallend in x,

b) $x_1 + f_n^\star(x_1) \le x_2 + f_n^\star(x_2)$.

Beweis
Seien $x_1 \le x_2$ zulässig und

$$\begin{aligned}
g_1(a) :=& h_n(p,a) + \int v_{n+1}(x_1+a,p')Q_n(dp'|p) = Lv_{n+1}(x_1,p,a)\\
g_2(a) :=& h_n(p,a) + \int v_{n+1}(x_2+a,p')Q_n(dp'|p) = Lv_{n+1}(x_2,p,a)\\
g_3(a) :=& g_1(a+(x_2-x_1)) = h_n(p,a+(x_2-x_1))\\
&+ \int v_{n+1}(x_1+a+(x_2-x_1),p')Q_n(dp'|p)\\
=& h_n(p,a+(x_2-x_1)) + \int v_{n+1}(x_2+a,p')Q_n(dp'|p).
\end{aligned}$$

a) Da v konkav ist (siehe Satz 4.1.9) ist

$$g_1(a) - g_2(a) = \int \underbrace{v_{n+1}(x_1+a,p') - v_{n+1}(x_2+a,p')}_{\text{wachsend in } a \ (\star_1)} Q_n(dp'|p)$$

wachsend in a und daher gilt

$$g_1(a) = g_2(a) + l_1(a)$$

mit l_1 wachsend und daraus folgt für die Maximalstellen $a_1^\star$ und $a_2^\star$ von g_1 und g_2

$$a_1^\star \geq a_2^\star.$$

Erklärung zu $(\star_1)$:

Sei f konkav, $x_1 \leq x_2$ und $a_1 \leq a_2$, dann ist die Sekantensteigung fallend und es gilt

$$\frac{f(x_2 + a_2) - f(x_1 + a_2)}{x_2 + a_2 - (x_1 + a_2)} \leq \frac{f(x_2 + a_1) - f(x_1 + a_1)}{x_2 + a_1 - (x_1 + a_1)}$$

woraus

$$f(x_1 + a_1) - f(x_2 + a_1) \leq f(x_1 + a_2) - f(x_2 + a_2)$$

folgt.

Also ist $a \mapsto f(x_1 + a) - f(x_2 + a)$ wachsend.

b) Da $g_3(a) = g_1(a + (x_2 - x_1))$ ist, gilt für die Maximalstelle $a_3^\star$ von g_3

$$a_3^\star = a_1^\star - (x_2 - x_1).$$

Desweiteren ist, da h_n konkav in a ist

$$g_3(a) - g_2(a) = h_n(p, a + \underbrace{(x_2 - x_1)}_{\geq 0}) - h_n(p, a)$$

fallend in a $(\star_2)$. Damit gilt

$$g_3(a) = g_2(a) + l_2(a)$$

mit l_2 fallend und daher

$$a_3^\star \leq a_2^\star \implies a_1^\star - (x_2 - x_1) \leq a_2^\star \implies a_1^\star + x_1 \leq a_2^\star + x_2$$

Erklärung zu $(\star_2)$:
Sei f konkav, $a_1 \leq a_2$ und $c > 0$, dann ist die Sekantensteigung fallend und es gilt

$$\frac{f(a_2 + c) - f(a_1 + c)}{a_2 + a - (a_1 + a)} \leq \frac{f(a_2) - f(a_1)}{a_2 - a_1}$$

woraus

$$f(a_2 + c) - f(a_2) \leq f(a_1 + c) - f(a_1)$$

folgt.

Also ist $a \mapsto f(a + c) - f(a)$ fallend. ∎

Bemerkung 4.1.12
Im Beispiel eines Gasspeichers, das in Kapitel 7 betrachtet wird, sind die Voraussetzungen an $i_n^{\max}$ und $i_n^{\min}$ erfüllt. Die Funktionen sind hier als

$$i_n^{\min}(x) = -70.71 \cdot \sqrt{x},$$
$$i_n^{\max}(x) = -0.032 \cdot x + 68170$$

gewählt.

Beide Funktionen erfüllen auf dem gewählten Intervall $[b^{\min}, b^{\max}] = [5000000, 20000000]$ die Stetigkeitseigenschaft. Außerdem ist $i_n^{\min}$ auf dem gewählten Intervall konvex und negativ und $i_n^{\max}$ auf dem gewählten Intervall konkav - sogar affin - und positiv.

Weiter ist festzustellen, dass für $i_n^{\max}$ und $i_n^{\min}$ konstant die Voraussetzungen ebenfalls gelten. Die Annahme konstanter Einspeicher- bzw. Ausspeicherraten ist häufig in der Literatur zu finden, so zum Beispiel in de Jong u. Walet [24] und Secomandi [34], sowie in den Anwendungsbeispielen von Boogert u. de Jong [5] und Felix [15].

4.2. Strukturaussagen zur optimalen Strategie in Spezialfällen

Betrachtet werden in diesem Kapitel die Vereinfachungen in Restriktionen und Preisprozess, siehe auch Kapitel 3.2.

4.2.1. Strukturaussagen bei Betrachtung des Gasspeichermodells als optimales switching Problem

Zunächst wird das Gasspeicherproblem als optimales switching Problem betrachtet. In diesem Fall ist es in jedem Zeitschritt nur möglich, den Speicher komplett zu füllen oder zu leeren oder keine Handlung zu tätigen.
Die Betrachtung dieses Spezialfalls wird insbesondere dann klar, wenn man einen Blick auf die Ergebnisse in Kapitel 4.2.2 wirft. Hier wird gezeigt, dass eine solche Politik auch in dem Fall optimal ist, wenn es möglich, aber nicht nötig ist, den Speicher komplett zu füllen, beziehungsweise zu leeren.

In Kapitel 3.2.1 wird das Modell für das entsprechende optimale switching Problem bereits eingeführt. Es sei also $E = \{b^{\min}, b^{\max}\} \times \mathcal{P}$ und $A = \{0, 1\}$. Der Preisprozess (P_n) sei ein Markovprozess mit $P_n = U(P_{n-1}, Y_n)$ mit $Y_1, Y_2, \ldots$ unabhängig identisch verteilt.

Da der Gasspeicher nur zwei Zustände annehmen kann, definiert man

$$\mathcal{P}^{\max} = \{b^{\max}\} \times \mathcal{P}, \ \mathcal{P}^{\min} = \{b^{\min}\} \times \mathcal{P}$$

sowie

$$p^{\max} = (b^{\max}, p),\ p^{\min} = (b^{\min}, p), \quad p \in \mathcal{P}.$$

Anders als im Hauptmodell in Kapitel 3.1 seien die Gewinnfunktionen allgemeiner für alle $n = 0, 1, \ldots, N-1$ gegeben durch

$$\begin{aligned} h(p^{\max}, 0) &= h(p^{\min}, 0) = 0 \\ h(p^{\max}, 1) &= g^{\max}(p), \quad h(p^{\min}, 1) = -g^{\min}(p). \end{aligned}$$

Die Funktionen $g^{\max}$ und $g^{\min}$ erfüllen die Bedingung

$$g^{\max}(p) < g^{\min}(p) \ \text{ für alle } p \in \mathcal{P}.$$

Die terminale Gewinnfunktion sei nicht weiter spezifiziert. Allerdings werden in den einzelnen Sätzen und Lemmata Voraussetzungen an die Gewinnfunktionen gestellt.

Bemerkung 4.2.1
Wählt man $g^{\max}$ und $g^{\min}$ wie folgt:

$$g^{\max}(p) = e(p) \quad \text{und} \quad g^{\min}(p) = k(p),$$

so sind die bisherigen Bedingungen erfüllt, falls $e(p) < k(p)$. Dies entspricht dem Fall, der in anderen Kapiteln betrachtet wird.

Wie schon in Kapitel 3.2.1 beschrieben ist die Wertfunktion - jetzt mit den Gewinnfunktionen eingesetzt - gegeben durch

$$V_n(b^{\max}, p) = \max\{g^{\max}(p) + \mathbb{E}_p\left[V_{n+1}(b^{\min}, P_{n+1})\right], \mathbb{E}_p\left[V_{n+1}(b^{\max}, P_{n+1})\right]\}$$

und

$$V_n(b^{\min}, p) = \max\{-g^{\min}(p) + \mathbb{E}_p\left[V_{n+1}(b^{\max}, P_{n+1})\right], \mathbb{E}_p\left[V_{n+1}(b^{\min}, P_{n+1})\right]\}.$$

Eine optimale Politik sieht demnach wie folgt aus

$$f_n^\star(b^{\max}, p) = \begin{cases} 1, & g^{\max}(p) \geq \mathbb{E}_p[V_{n+1}(b^{\max}, P_n)] - \mathbb{E}_p[V_{n+1}(b^{\min}, P_n)], \\ 0, & g^{\max}(p) < \mathbb{E}_p[V_{n+1}(b^{\max}, P_n)] - \mathbb{E}_p[V_{n+1}(b^{\min}, P_n)], \end{cases}$$

$$f_n^\star(b^{\min}, p) = \begin{cases} 1, & g^{\min}(p) \leq \mathbb{E}_p[V_{n+1}(b^{\max}, P_n)] - \mathbb{E}_p[V_{n+1}(b^{\min}, P_n)], \\ 0, & g^{\min}(p) > \mathbb{E}_p[V_{n+1}(b^{\max}, P_n)] - \mathbb{E}_p[V_{n+1}(b^{\min}, P_n)]. \end{cases}$$

Das weitere Vorgehen zur näheren Betrachtung der Struktur der optimalen Politik lehnt sich an die Arbeit von Bagus [2] an, der das Problem der Swing Optionen unter anderem

als optimales switching Problem auffasst.

Zunächst sei für $n \in \{0, 1, \ldots, N\}$ die Funktion

$$G_n(p) := \mathbb{E}_p[V_n(b^{\max}, P_n)] - \mathbb{E}_p[V_n(b^{\min}, P_n)] \tag{4.1}$$

definiert, die in der Untersuchung eine große Rolle spielt.

Die optimale Politik lässt sich mit Hilfe der Funktion G_n vereinfachen zu

$$f_n^\star(b^{\max}, p) = \begin{cases} 1, & g^{\max}(p) \geq G_{n+1}(p), \\ 0, & g^{\max}(p) < G_{n+1}(p), \end{cases}$$
$$f_n^\star(b^{\min}, p) = \begin{cases} 1, & g^{\min}(p) \leq G_{n+1}(p), \\ 0, & g^{\min}(p) > G_{n+1}(p). \end{cases}$$

Es zeigt sich, dass die Funktion G_n für die Struktur der optimalen Politik entscheidend ist, daher wird sie im Folgenden untersucht.

Lemma 4.2.2
Es ist

$$\begin{aligned} G_n(p) =& \mathbb{E}_p\left[g^{\min}(\tilde{P})\mathbf{1}_{\{G_{n+1}(\tilde{P}) \geq g^{\min}(\tilde{P})\}}\right] \\ &+ \mathbb{E}_p\left[G_{n+1}(\tilde{P})\mathbf{1}_{\{g^{\max}(\tilde{P}) < G_{n+1}(\tilde{P}) < g^{\min}(\tilde{P})\}}\right] \\ &+ \mathbb{E}_p\left[g^{\max}(\tilde{P})\mathbf{1}_{\{G_{n+1}(\tilde{P}) \leq g^{\max}(\tilde{P})\}}\right], \end{aligned} \tag{4.2}$$

wobei $\tilde{P} = U(p, Y_n)$.

Beweis
Mit der Funktion G_n lässt sich die Wertfunktion V_n schreiben als

$$\begin{aligned} V_n(b^{\max}, p) =& (g^{\max}(p) + \mathbb{E}_p[V_{n+1}(b^{\min}, \tilde{P})])\mathbf{1}_{\{g^{\max}(p) \geq G_{n+1}(p)\}} \\ &+ \mathbb{E}_p[V_{n+1}(b^{\max}, \tilde{P})]\mathbf{1}_{\{g^{\max}(p) < G_{n+1}(p)\}}, \\ V_n(b^{\min}, p) =& (-g^{\min}(p) + \mathbb{E}_p[V_{n+1}(b^{\max}, \tilde{P})])\mathbf{1}_{\{g^{\min}(p) \leq G_{n+1}(p)\}} \\ &+ \mathbb{E}_p[V_{n+1}(b^{\min}, \tilde{P})]\mathbf{1}_{\{g^{\min}(p) > G_{n+1}(p)\}}, \end{aligned}$$

wobei $\tilde{P} = U(p, Y_{n+1})$. Somit ist

$$\begin{aligned} V_n(b^{\max}, p) - V_n(b^{\min}, p) =& g^{\max}(p)\mathbf{1}_{\{g^{\max}(p) \geq G_{n+1}(p)\}} \\ &+ \mathbb{E}_p V_{n+1}(b^{\min}, \tilde{P})\mathbf{1}_{\{g^{\max}(p) \geq G_{n+1}(p)\}} \\ &+ \mathbb{E}_p V_{n+1}(b^{\max}, \tilde{P})\mathbf{1}_{\{g^{\max}(p) < G_{n+1}(p)\}} \\ &- \mathbb{E}_p V_{n+1}(b^{\max}, \tilde{P})\mathbf{1}_{\{g^{\min}(p) \leq G_{n+1}(p)\}} \\ &+ g^{\min}(p)\mathbf{1}_{\{g^{\min}(p) \leq G_{n+1}(p)\}} \\ &- \mathbb{E}_p V_{n+1}(b^{\min}, \tilde{P})\mathbf{1}_{\{g^{\min}(p) > G_{n+1}(p)\}}. \end{aligned} \tag{4.3}$$

Für festes p gilt $g^{\min}(p) > g^{\max}(p)$, daher folgt aus $G_{n+1}(p) > g^{\min}(p)$ auch $G_{n+1}(p) > g^{\max}(p)$ und aus $g^{\max}(p) > G_{n+1}(p)$ auch $g^{\min}(p) > G_{n+1}(p)$.
Mittels Fallunterscheidung erhält man nun

$$\begin{aligned}(4.3) = {} & g^{\min}(p)\mathbf{1}_{\{G_{n+1}(p)\geq g^{\min}(p)\}} + G_{n+1}(p)\mathbf{1}_{\{g^{\max}(p)<G_{n+1}(p)<g^{\min}(p)\}} \\ & + g^{\max}(p)\mathbf{1}_{\{G_{n+1}(p)\leq g^{\max}(p)\}}.\end{aligned}$$

Mit Erwartungswertbildung folgt dann die Behauptung. ∎

Die folgenden Lemmata zeigen, dass G_n zum einen fallend in n, zum andern monoton wachsend in p ist.

Lemma 4.2.3
Aus $G_N(p) \leq G_{N-1}(p)$ folgt für $n \in \{0, 1, \ldots, N-1\}$

$$G_{n+1}(p) \leq G_n(p) \quad \text{für alle } p \in \mathcal{P}.$$

Beweis
Der Beweis erfolgt mittels Rückwärtsinduktion. Dabei ist zu beachten, dass der Induktionsanfang gerade die Voraussetzung darstellt.

Es gelte also (Induktionshypothese) $G_{n+1}(p) \leq G_n(p)$ für ein $n \in \{0, 1, \ldots, N\}$. Dann ist

$$\begin{aligned}G_n(p) = {} & \mathbb{E}_p\left[g^{\min}(U(p,Y_n))\mathbf{1}_{\{G_{n+1}(U(p,Y_n))\geq g^{\min}(U(p,Y_n))\}}\right] \\ & + \mathbb{E}_p\left[G_{n+1}(U(p,Y_n))\mathbf{1}_{\{g^{\max}(U(p,Y_n))<G_{n+1}(U(p,Y_n))<g^{\min}(U(p,Y_n))\}}\right] \\ & + \mathbb{E}_p\left[g^{\max}(U(p,Y_n))\mathbf{1}_{\{G_{n+1}(U(p,Y_n))\leq g^{\max}(U(p,Y_n))\}}\right] \\ \leq {} & \mathbb{E}_p\left[g^{\min}(U(p,Y_n))\mathbf{1}_{\{G_n(U(p,Y_n))\geq g^{\min}(U(p,Y_n))\}}\right] \\ & + \mathbb{E}_p\left[G_n(U(p,Y_n))\mathbf{1}_{\{g^{\max}(U(p,Y_n))<G_n(U(p,Y_n))<g^{\min}(U(p,Y_n))\}}\right] \\ & + \mathbb{E}_p\left[g^{\max}(U(p,Y_n))\mathbf{1}_{\{G_n(U(p,Y_n))\leq g^{\max}(U(p,Y_n))\}}\right] \\ = {} & G_{n-1}(p)\end{aligned}$$

Hierbei gilt das Ungleichheitszeichen aufgrund der Induktionshypothese und einer Fallunterscheidung. Das Gleichheitszeichen am Ende gilt, da die $Y_1, Y_2, \ldots$ identisch verteilt sind. ∎

Lemma 4.2.4
Es sei $p \mapsto G_N(p)$ wachsend. Außerdem gelte

- $g^{\max}$ und $g^{\min}$ monoton wachsend
- $p \mapsto U(p, y)$ monoton wachsend

Dann gilt für $n = 0, 1, \ldots, N-1$

$$p \mapsto G_n(p) \text{ ist wachsend.}$$

Beweis
Der Beweis erfolgt mit Rückwärtsinduktion, wobei der Induktionsanfang gerade die Voraussetzung ist.

Es gelte $p \mapsto G_n(p)$ ist monoton wachsend (Induktionshypothese).
Nach Lemma 4.2.2 gilt

$$G_{n-1}(p) = \mathbb{E}[\varphi(P_{n-1})|P_{n-2} = p]$$

mit

$$\begin{aligned}\varphi(p) = g^{\max}(p)\mathbf{1}_{\{G_n(p)\leq g^{\max}(p)\}} + g^{\min}(p)\mathbf{1}_{\{G_n(p)\geq g^{\min}(p)\}} \\ + G_n(p)\mathbf{1}_{\{g^{\max}(p)<G_n(p)<g^{\min}(p)\}}.\end{aligned} \tag{4.4}$$

Es wird nun gezeigt, dass φ wachsend ist.

Sei dazu $p' \geq p$, $p', p \in \mathcal{P}$. Sind für p' und p die gleichen Indikatoren eins, so gilt $\varphi(p') \geq \varphi(p)$, da $g^{\max}, g^{\min}$ und G_n wachsend sind.

Eine Fallunterscheidung zeigt, dass auch in den anderen Fällen $\varphi(p') \geq \varphi(p)$ gilt.
Alle möglichen Fälle werden nun ausgeführt:

1. Fall: $G_n(p) \leq g^{\max}(p)$, d.h. $\varphi(p) = g^{\max}(p)$. Gilt nun
   (i) $G_n(p') \leq g^{\max}(p')$, so folgt
   $\varphi(p') = g^{\max}(p') \geq g^{\max}(p)$ ✓
   (ii) $G_n(p') > g^{\max}(p')$ und $G_n(p') \geq g^{\min}(p')$, so folgt
   $\varphi(p') = g^{\min}(p') > g^{\max}(p') \geq g^{\max}(p)$ ✓
   (iii) $G_n(p') > g^{\max}(p')$ und $G_n(p') < g^{\min}(p')$, so folgt
   $\varphi(p') = G_n(p') > g^{\max}(p') \geq g^{\max}(p)$ ✓

2. Fall: $G_n(p) \geq g^{\min}(p)$, d.h. $\varphi(p) = g^{\min}(p)$.
   (i) $G_n(p') \geq g^{\min}(p')$, so folgt
   $\varphi(p') = g^{\min}(p') \geq g^{\min}(p)$ ✓
   (ii) $G_n(p') < g^{\min}(p')$ und $G_n(p') > g^{\max}(p')$, so folgt
   $\varphi(p') = G_n(p') \geq G_n(p) > g^{\min}(p)$ ✓
   (iii) $G_n(p') < g^{\min}(p')$ und $G_n(p') \leq g^{\max}(p')$, so folgt
   $\varphi(p') = g^{\max}(p') > G_n(p') \geq G_n(p) > g^{\min}(p)$ ✓

3. Fall: $g^{\max}(p) < G_n(p) < g^{\min}(p)$, d.h. $\varphi(p) = G_n(p)$.
   (i) $g^{\max}(p') < G_n(p') < g^{\min}(p')$, so folgt
   $\varphi(p') = G_n(p') \geq G_n(p)$ ✓
   (ii) $g^{\max}(p') < g^{\min}(p') \leq G_n(p')$, so folgt
   $\varphi(p') = g^{\min}(p') \geq g^{\min}(p) > G_n(p)$ ✓
   (iii) $G_n(p') \leq g^{\max}(p') < g^{\min}(p')$, so folgt
   $\varphi(p') = g^{\max}(p') > G_n(p') \geq G_n(p)$ ✓

Somit ist gezeigt, dass $p \mapsto \varphi(p)$ wachsend ist.

Es gilt, dass

$$\begin{aligned} G_{n-1}(p) &= \mathbb{E}[\varphi(P_{n-1})|P_{n-2} = p] = \mathbb{E}[\varphi(U(P_{n-2}, Y_{n-1}))|P_{n-2} = p] \\ &= \mathbb{E}[\varphi(U(p, Y_{n-1}))|P_{n-2} = p] = \mathbb{E}[\varphi(U(p, Y_{n-1}))] \end{aligned}$$

ebenfalls wachsend in p ist, was zu zeigen war. ∎

Die Frage, die sich nun stellt, ist:
Wann finden sich Schranken $s_n^{\max}$ und $s_n^{\min}$, sodass die optimale Politik durch

$$f_n^\star(b^{\max}, p) = \begin{cases} 1, & p \geq s_n^{\max}, \\ 0, & p < s_n^{\max}, \end{cases} \tag{4.5}$$

bzw.

$$f_n^\star(b^{\min}, p) = \begin{cases} 1, & p \leq s_n^{\min}, \\ 0, & p > s_n^{\min}, \end{cases} \tag{4.6}$$

gegeben ist? In diesem Fall spricht man von einer sogenannten „bang-bang“-Strategie, denn ist der Preis über bzw. unter - je nach Zustand - einer bestimmten Grenze, so wird in den jeweils anderen Zustand gewechselt.

Satz 4.2.5
Es gelte $p \mapsto g^{\max}(p) - G_{n+1}(p)$ und $p \mapsto g^{\min}(p) - G_{n+1}(p)$ wachsend.
Dann ist die optimale Strategie wie in (4.5) bzw. (4.6) mit

$$s_n^{\max} = \begin{cases} \inf\{p \geq 0 : g^{\max}(p) \geq G_{n+1}(p)\}, & \text{falls } \{p \geq 0 : g^{\max}(p) \geq G_{n+1}(p)\} \neq \emptyset \\ \infty, & \text{sonst} \end{cases} \tag{4.7}$$

und

$$s_n^{\min} = \begin{cases} \sup\{p \geq 0 : g^{\min}(p) \leq G_{n+1}(p)\}, & \text{falls } \{p \geq 0 : g^{\min}(p) \leq G_{n+1}(p)\} \neq \emptyset \\ 0, & \text{sonst.} \end{cases} \tag{4.8}$$

Beweis
Zu zeigen ist

$$p \geq s_n^{\max} \iff g^{\max}(p) \geq G_{n+1}(p)$$

bzw.

$$p \leq s_n^{\min} \iff g^{\min}(p) \leq G_{n+1}(p).$$

Gibt es ein p' mit $g^{\max}(p') \geq G_{n+1}(p')$, so gilt für jedes $p \geq p'$ nach Voraussetzung

$$g^{\max}(p) - G_{n+1}(p) \geq g^{\max}(p') - G_{n+1}(p') \geq 0$$

und daher

$$g^{\max}(p) \geq G_{n+1}(p).$$

Aus $p \geq s_n^{\max}$ folgt somit $g^{\max}(p) \geq G_{n+1}(p)$. In ähnlicher Weise zeigt man, dass aus $p < s_n^{\max}$ die Ungleichung $g^{\max}(p) < G_{n+1}(p)$ folgt. Damit ist auch die andere Richtung der Äquivalenz gezeigt.

Gibt es ein p' mit $g^{\min}(p') \leq G_{n+1}(p')$, so gilt für $p \leq p'$:

$$g^{\min}(p) - G_{n+1}(p) \leq g^{\min}(p') - G_{n+1}(p') \leq 0$$

und daher

$$g^{\min}(p) \leq G_{n+1}(p).$$

Analog zum Fall $s_n^{\max}$ zeigt sich nun die Äquivalenz im Fall $s_n^{\min}$. ∎

Nachfolgende Lemmata beschäftigen sich mit Eigenschaften der Schranken für das Wechseln des Zustands.

Lemma 4.2.6
Unter den Voraussetzungen der vorangegangen Sätze und Lemmas gilt

$$s_n^{\max} \leq s_{n-1}^{\max} \quad \text{und} \quad s_n^{\min} \leq s_{n-1}^{\min}.$$

Beweis
Für alle $p \geq s_{n-1}^{\max}$ gilt

$$g^{\max}(p) \geq G_n(p) \geq G_{n+1}(p),$$

woraus

$$p \geq s_n^{\max}$$

folgt.
Analog gilt für alle $p \leq s_n^{\min}$

$$g^{\min}(p) \leq G_{n+1}(p) \leq G_n(p)$$

woraus

$$p \leq s_{n+1}^{\min}$$

folgt. Insgesamt erhält man daraus die Behauptung. ∎

Lemma 4.2.7
Es seien $g^{\min}, g^{\max}$ und G_{n+1} stetig. Dann gilt

$$s_n^{\min} \leq s_n^{\max}.$$

Beweis
Die Aussage ist gültig, falls mindestens eine der Mengen $\{p \geq 0 : g^{\max}(p) \geq G_{n+1}(p)\}$ bzw. $\{p \geq 0 : g^{\min}(p) \leq G_{n+1}(p)\}$ leer ist. Seien also die Mengen nicht leer.

Schneiden sich die Funktionen $g^{\max}$ und G_{n+1} nicht, so gilt (da die Mengen nicht leer sind) $g^{\max}(p) > G_{n+1}(p)$ für alle $p \geq 0$, d.h. $s_n^{\max} = 0$.

Schneiden sich die Funktionen $g^{\min}$ und G_{n+1} nicht so gilt (da die Mengen nicht leer sind) $g^{\min}(p) < G_{n+1}(p)$ für alle $p \geq 0$, d.h. $s_n^{\min} = \infty$.

Gilt beides, so würde dies bedeuten, dass für alle $p \geq 0$: $g^{\min}(p) < G_{n+1}(p) < g^{\max}(p)$ gälte, was wiederum ein Widerspruch zur Voraussetzung $g^{\max} < g^{\min}$ punktweise ist.

Gilt nur eines, so führt die Voraussetzung $g^{\max}(p) < g^{\min}(p)$ für alle $p \geq 0$ in ähnlicher Weise zum Widerspruch.

Es wird angenommen, dass sowohl $\{p \geq 0 : g^{\min}(p) = G_{n+1}(p)\} \neq \emptyset$, also auch $\{p \geq 0 : g^{\max}(p) = G_{n+1}(p)\} \neq \emptyset$ ist.

Da der Abstand zwischen G_{n+1} und $g^{\max}$ bzw. $g^{\min}$ in folgendem Sinne fallend ist, d.h. für $p \geq p'$

$$G_{n+1}(p) - g^{\max}(p) \leq G_{n+1}(p') - g^{\max}(p'),$$
$$G_{n+1}(p) - g^{\min}(p) \leq G_{n+1}(p') - g^{\min}(p')$$

gilt und die Funktionen stetig sind, sind die Mengen $\{p \geq 0 : g^{\min}(p) = G_{n+1}(p)\}$ und $\{p \geq 0 : g^{\max}(p) = G_{n+1}(p)\}$ zusammenhängend.

Eine Fallunterscheidung liefert nun

(i) Es sei $\{p \geq 0 : g^{\min}(p) = G_{n+1}(p)\} = [p_1, \infty)$, dann ist $s_n^{\min} = \infty$.
Für alle $\tilde{p} \geq p_1$ gilt $g^{\min}(\tilde{p}) \leq G_{n+1}(\tilde{p})$ und demnach auch $g^{\max}(\tilde{p}) < G_{n+1}(\tilde{p})$, woraus $0 < G_{n+1}(\tilde{p}) - g^{\max}(\tilde{p})$ folgt.
Für alle $p \leq \tilde{p}$ gilt nach Voraussetzung $0 < G_{n+1}(\tilde{p}) - g^{\max}(\tilde{p}) \leq G_{n+1}(p) - g^{\max}(p)$ und damit $s_n^{\max} = \infty$.

(ii) Es sei $\{p \geq 0 : g^{\min}(p) = G_{n+1}(p)\} = [p_1, p_2]$ und $\{p \geq 0 : g^{\max}(p) = G_{n+1}(p)\} = [\overline{p}_1, \overline{p}_2]$ oder $[\overline{p}_1, \infty)$, dann ist $s_n^{\min} = p_2$ und $s_n^{\max} = \overline{p}_1$.
Da $g^{\max}(p) < g^{\min}(p)$ für alle $p \geq 0$ gilt $\{p : g^{\max}(p) = g^{\min}(p) = G_{n+1}(p)\} = \emptyset$.
Entweder gilt demnach $p_2 < \overline{p}_1$ oder $\overline{p}_2 < p_1$.
Angenommen es gilt $\overline{p}_2 < p_1$. Für alle $p \leq p_1$ gilt $g^{\min}(p) \leq G_{n+1}(p)$, also auch für $\overline{p}_2$. Damit gilt

$$g^{\max}(\overline{p}_2) < g^{\min}(\overline{p}_2) \leq G_{n+1}(\overline{p}_2) = g^{\max}(\overline{p}_2). \text{ Widerspruch.}$$

Folglich gilt $p_2 < \overline{p}_1$. ∎

Es stellt sich das Problem, wann die Voraussetzungen an $p \mapsto g^{\max}(p) - G_{n+1}(p)$ und $p \mapsto g^{\min}(p) - G_{n+1}(p)$ von Satz 4.2.5, die einen essentiellen Teil der Beweise darstellen, erfüllt sind. Die Frage ist, ob es allgemeine Voraussetzungen an z.B. $g^{\max}, g^{\min}$ gibt, damit diese Eigenschaften erfüllt sind.

Satz 4.2.8
Für $g^{\max}$, $g^{\min}$ und U gelten zusätzlich folgende Eigenschaften:

- $p \mapsto g^{\min}(p) - g^{\max}(p)$ monoton wachsend,
- $p \mapsto g^{\max}(p) - g^{\min}(U(p, y))$ monoton wachsend.

Desweiteren gelte für G_N: $p \mapsto g^{\max}(p) - G_N(p)$ wachsend.
Dann gilt für alle n

$$p \mapsto g^{\max}(p) - G_n(p) \text{ und } p \mapsto g^{\min}(p) - G_n(p) \text{ sind monoton wachsend.}$$

Beweis
Es genügt zu zeigen, dass $p \mapsto g^{\max}(p) - G_n(p)$ wachsend ist, denn dann gilt für $p \geq p'$

$$G_n(p) - G_n(p') \leq g^{\max}(p) - g^{\max}(p') \leq g^{\min}(p) - g^{\min}(p').$$

Das letzte Ungleichheitszeichen gilt, da $g^{\min} - g^{\max}$ wachsend ist. Aus dieser Ungleichungskette folgt nun aber auch, dass $p \mapsto g^{\min}(p) - G_n(p)$ wachsend ist.

Behauptung: Für $p \geq p'$ gilt

$$g^{\max}(p) - G_n(p) \geq g^{\max}(p') - G_n(p') \text{ für } n = 0, 1, \ldots, N.$$

Der Beweis erfolgt mittels Rückwärtsinduktion.

1. Induktionsanfang: Sei $n = N$, dann ist $p \mapsto g^{\max}(p) - G_N(p)$ monoton wachsend nach Voraussetzung.
2. Induktionshypothese: Für ein n gelte die Behauptung.
3. Induktionsschritt von n nach $n-1$. Es ist nach Lemma 4.2.2

$$\begin{aligned}
&g^{\max}(p) - G_{n-1}(p) \\
&= \mathbb{E}_p\Big[(g^{\max}(p) - g^{\max}(\tilde{P}))\mathbf{1}_{\{g^{\max}(\tilde{P}) \geq G_n(\tilde{P})\}} \\
&\qquad + (g^{\max}(p) - G_n(\tilde{P}))\mathbf{1}_{\{g^{\max}(\tilde{P}) < G_n(\tilde{P}) < g^{\min}(\tilde{P})\}} \\
&\qquad + (g^{\max}(p) - g^{\min}(\tilde{P}))\mathbf{1}_{\{G_n(\tilde{P}) \geq g^{\min}(\tilde{P})\}}\Big] \\
&= \mathbb{E}_p\Big[(g^{\max}(p) - g^{\max}(U(p, Y_{n-1})))\mathbf{1}_{\{g^{\max}(U(p,Y_{n-1})) \geq G_n(U(p,Y_{n-1}))\}} \\
&\qquad + (g^{\max}(p) - G_n(U(p, Y_{n-1})))\mathbf{1}_{\{g^{\max}(U(p,Y_{n-1})) < G_n(U(p,Y_{n-1})) < g^{\min}(U(p,Y_{n-1}))\}} \\
&\qquad + (g^{\max}(p) - g^{\min}(U(p, Y_{n-1})))\mathbf{1}_{\{G_n(U(p,Y_{n-1})) \geq g^{\min}(U(p,Y_{n-1}))\}}\Big]
\end{aligned}$$

Definiere

$$\begin{aligned}\varphi(p,y) :=&(g^{\max}(p) - g^{\max}(U(p,y)))\mathbf{1}_{\{g^{\max}(U(p,y))\geq G_n(U(p,y))\}}\\ &+ (g^{\max}(p) - G_n(U(p,y)))\mathbf{1}_{\{g^{\max}(U(p,y))<G_n(U(p,y))<g^{\min}(U(p,y))\}}\\ &+ (g^{\max}(p) - g^{\min}(U(p,y)))\mathbf{1}_{\{G_n(U(p,y))\geq g^{\min}(U(p,y))\}}\end{aligned}$$

und zeige nun $p \mapsto \varphi(p,y)$ ist monoton wachsend für alle y. Dann gilt auch $p \mapsto g^{\max}(p) - G_{n-1}(p) = \mathbb{E}[\varphi(p, Y_{n-1})]$ ist wachsend.

Damit $p \mapsto \varphi(p,y)$ wachsend ist, müssen zunächst

a) $p \mapsto g^{\max}(p) - g^{\max}(U(p,y))$

b) $p \mapsto g^{\max}(p) - G_n(U(p,y))$

c) $p \mapsto g^{\max}(p) - g^{\min}(U(p,y))$

wachsend sein.

Die Funktion in c) ist nach Voraussetzung wachsend. Daraus folgt, dass a) wachsend ist, denn sei $p \geq p'$, so gilt

$$g^{\max}(p) - g^{\max}(p') \overset{c)}{\geq} g^{\min}(U(p,y)) - g^{\min}(U(p'y)) \geq g^{\max}(U(p,y)) - g^{\max}(U(p',y)).$$

Hierbei gilt das hintere Ungleichheitszeichen, da $g^{\min} - g^{\max}$ und $p \mapsto U(p,y)$ wachsend sind.

Teil b) folgt aus a) und aus der Induktionshypothese, denn es ist für $p \geq p'$

$$G_n(U(p,y)) - G_n(U(p',y)) \leq g^{\max}(U(p,y)) - g^{\max}(U(p',y)) \overset{a)}{\leq} g^{\max}(p) - g^{\max}(p').$$

Mittels einer Fallunterscheidung wird nun der letzte Teil der Behauptung gezeigt. Sei $p \geq p'$ und

I $g^{\max}(U(p',y)) \geq G_n(U(p',y))$, das heißt $\varphi(p',y) = g^{\max}(p') - g^{\max}(U(p',y))$, und $g^{\max}(U(p,y)) \leq G_n(U(p,y))$ (andernfalls entspräche dies Teil a)). Sei weiter

(i) $G_n(U(p,y)) \leq g^{\min}(U(p,y))$, dann folgt

$$\begin{aligned}\varphi(p,y) &= g^{\max}(p) - G_n(U(p,y))\\ &\overset{b)}{\geq} g^{\max}(p') - G_n(U(p',y))\\ &\geq g^{\max}(p') - g^{\max}(U(p',y)).\ \checkmark\end{aligned}$$

(ii) $G_n(U(p,y)) \geq g^{\min}(U(p,y))$, dann folgt

$$\begin{aligned}\varphi(p,y) &= g^{\max}(p) - g^{\min}(U(p,y))\\ &\geq g^{\max}(p) - G_n(U(p,y))\\ &\overset{b)}{\geq} g^{\max}(p') - G_n(U(p'y))\\ &\geq g^{\max}(p') - g^{\max}(U(p',y))\ \checkmark\end{aligned}$$

II $g^{\min}(U(p',y)) \leq G_n(U(p',y))$, das heißt $\varphi(p',y) = g^{\max}(p') - g^{\min}(U(p',y))$, und $g^{\min}(U(p,y)) \geq G_n(p,y)$ (andernfalls entspräche dies Teil c)). Sei weiter

(i) $G_n(U(p,y)) \leq g^{\max}(U(p,y))$, dann folgt

$$\begin{aligned}\varphi(p,y) &= g^{\max}(p) - g^{\max}(U(p,y)) \\ &\geq g^{\max}(p) - g^{\min}(U(p,y)) \\ &\overset{c)}{\geq} g^{\max}(p') - g^{\min}(U(p',y)) \checkmark\end{aligned}$$

(ii) $G_n(U(p,y)) \geq g^{\max}(U(p,y))$, dann folgt

$$\begin{aligned}\varphi(p,y) &= g^{\max}(p) - G_n(U(p,y)) \\ &\geq g^{\max}(p) - g^{\min}(U(p,y)) \\ &\overset{c)}{\geq} g^{\max}(p') - g^{\min}(U(p',y)) \checkmark\end{aligned}$$

III $g^{\max}(U(p',y)) < G_n(U(p',y)) < g^{\min}(U(p',y))$, das heißt $\varphi(p',y) = g^{\max}(p') - G_n(U(p',y))$. Sei weiter

(i) $g^{\max}(U(p,y)) < G_n(U(p,y)) < g^{\min}(U(p,y))$. Das ist Teil b).

(ii) $g^{\max}(U(p,y)) < g^{\min}(U(p,y)) \leq G_n(U(p,y))$, dann folgt

$$\begin{aligned}\varphi(p,y) &= g^{\max}(p) - g^{\min}(U(p,y)) \\ &\geq g^{\max}(p) - G_n(U(p,y)) \\ &\overset{b)}{\geq} g^{\max}(p') - G_n(U(p',y)) \checkmark\end{aligned}$$

(iii) $G_n(U(p,y)) \leq g^{\max}(U(p,y)) < g^{\min}(U(p,y))$, dann folgt

$$\begin{aligned}\varphi(p,y) &= g^{\max}(p) - g^{\max}(U(p,y)) \\ &\overset{a)}{\geq} g^{\max}(p') - g^{\max}(U(p'y)) \\ &\geq g^{\max}(p') - G_n(U(p',y)). \checkmark\end{aligned}$$

■

Bemerkung 4.2.9
Sind die Funktionen $g^{\max}$ und $g^{\min}$ gegeben durch

$$g^{\max}(p) := ap + b \text{ und } g^{\min}(p) = cp + d,$$

mit $a, c \in \mathbb{R}^+$, so muss $a \leq c$ und $b < d$ gelten, damit $g^{\max}(p) < g^{\min}(p)$ für alle p. In diesem Fall ist die Voraussetzung $g^{\min} - g^{\max}$ wachsend auch erfüllt, da $c \geq a$ ist.

Die anderen Voraussetzungen vereinfachen sich hier zu

- $p \mapsto ap - G_N(p)$ wachsend,

- $p \mapsto \frac{a}{c}p - U(p,y)$ wachsend.

Die zweite Bedingung bedeutet gerade, dass $p \mapsto U(p,y)$ Lipschitzstetig mit Lipschitzkonstante $L \leq \frac{a}{c}$ sein muss.

Es werden Beispiele für $g^{\max}, g^{\min}, g_N$ und U angegeben, sodass die Voraussetzungen von Satz 4.2.8 erfüllt sind.

Beispiel 4.2.10
Im ersten Beispiel wird der Fall wie in Bemerkung 4.2.9, im zweiten wird der allgemeinere Fall betrachtet.

a) Es sei (P_n) eine Markovkette auf dem Zustandsraum $\mathbb{Q} \cap [0,l]$ mit

$$P_n = \min\{(q \cdot P_{n-1} + Y_n)^+, l\},$$

wobei $q \in \mathbb{Q}^+$ und $q \leq \frac{a}{c}$. Die Zufallsvariablen $Y_1, Y_2, \ldots$ seien unabhängig identisch verteilt mit $Y_1(\Omega) = \{-1, 0, 1\}$.

Dann gilt

$$U(p,y) = \min\{(qp + y)^+, l\}$$

und sowohl $p \mapsto U(p,y)$ als auch $p \mapsto \frac{a}{c}p - U(p,y)$ sind wachsend.

Wählt man z.B. konkret $a = c,\ q = 1$ und $\mathbb{P}(Y_1 = 1) = p_1$, $\mathbb{P}(Y_1 = -1) = p_2$ und $\mathbb{P}(Y_1 = 0) = 1 - p_1 - p_2$, $p_1, p_2 \in (0,1)$, so erhält man ein Trinomialmodell, das heißt eine Markovkette mit folgender $(l+1) \times (l+1)$-Übergangsmatrix

$$\begin{pmatrix}
1-p_1 & p_1 & 0 & \cdots & & & 0 \\
p_2 & 1-p_1-p_2 & p_1 & 0 & \cdots & & \vdots \\
0 & p_2 & 1-p_1-p_2 & p_1 & 0 & \cdots & \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
 & & 0 & p_2 & 1-p_1-p_2 & p_1 & 0 \\
 & & & 0 & p_2 & 1-p_1-p_2 & p_1 \\
0 & \cdots & & & 0 & p_2 & 1-p_2
\end{pmatrix}.$$

Es sei $g_N(b^{\max}, p) = g^{\max}(p)$ und $g_N(b^{\min}, p) = 0$. Zu zeigen ist

- $G_{N-1}(p) \geq G_N(p)$,
- G_N ist wachsend,
- $p \mapsto ap - G_N(p)$ ist wachsend.

Es gilt

$$\begin{aligned} G_N(p) &= \mathbb{E}[V_N(b^{\max}, P_N) - V_N(b^{\min}, P_N)|P_{N-1} = p] \\ &= \mathbb{E}[g^{\max}(P_N)|P_{N-1} = p] = \mathbb{E}[g^{\max}(U(p, Y_N)]. \end{aligned}$$

Da $g^{\max}$ und $p \mapsto U(p, y)$ wachsend sind ist auch G_N wachsend.

Außerdem gilt, dass

$$p \mapsto ap - G_N(p) = \mathbb{E}[ap - g^{\max}(U(p, Y_N)] = \mathbb{E}[ap - aU(p, Y_N) - b]$$

wachsend ist, falls $p \mapsto ap - aU(p, y)$ wachsend ist. Das ist aber der Fall, da $p \mapsto \frac{a}{c}p - U(p, y)$ wachsend ist (vgl. Beweis von Satz 4.2.8).

Zuletzt gilt auch

$$\begin{aligned} G_{N-1}(p) = \mathbb{E}_p\Big[& \underbrace{g^{\min}(U(p, Y_{N-1}))}_{\geq g^{\max}(U(p,Y_{N-1}))} \mathbf{1}_{\{G_N(U(p,Y_{N-1})) \geq g^{\min}(U(p,Y_{N-1}))\}} \\ & + g^{\max}(U(p, Y_{N-1}) \mathbf{1}_{\{G_N(U(p,Y_{N-1})) \leq g^{\max}(U(p,Y_{N-1})\}} \\ & + \underbrace{G_N(U(p, Y_{N-1}))}_{\geq g^{\max}(U(p,Y_{N-1}))} \mathbf{1}_{\{g^{\max}(U(p,Y_{N-1})) < G_N(U(p,Y_{N-1})) < g^{\min}(U(p,Y_{N-1}))\}} \Big] \\ \geq \quad & \mathbb{E}[g^{\max}(U(p, Y_{N-1}))] = \mathbb{E}[g^{\max}(U(p, Y_N))] = G_N(p). \end{aligned}$$

b) Es sei $g_N(b^{\max}, p) = g^{\max}(p)$ und $g_N(b^{\min}, p) = 0$, sowie $g^{\max}(p) = \log(p + 1) + c_1$, $g^{\min}(p) = p + c_2$ und $U(p, y) = (c_3 \log(p + 3) + y)^+$, $y \in \{-1, 0, 1\}$. Es gelte $c_1 < c_2$ und $c_3 \leq 1$, dann ist

- $g^{\max}(p) < g^{\min}(p)$ für alle $p \geq 0$.
- $g^{\min} - g^{\max}$ wachsend, denn

$$\frac{d}{dp}(g^{\min} - g^{\max})(p) = 1 - \frac{1}{p+1} \geq 0, \qquad \text{für } p \geq 0.$$

- $p \mapsto g^{\max}(p) - g^{\min}(U(p, y))$ wachsend, denn für $p \geq 2$ gilt

$$\frac{d}{dp}(g^{\min}(U(p, y)) - g^{\max}(p)) = \frac{c_3}{p+1} - \frac{1}{p+1} \leq \frac{1}{p+1} - \frac{1}{p+1} \leq 0$$

  und für $0 \leq p \leq 2$ gilt $p \mapsto g^{\max}(p) - g^{\min}(U(p, y)) = g^{\max}(p) - c_2$ ist wachsend.
- Die Eigenschaften für G_N lassen sich analog zum ersten Beispiel zeigen.

4.2.2. Struktur der optimalen Politik im Fall vereinfachter Restriktionen

Dieser Abschnitt betrachtet die Strukturuntersuchungen im Fall vereinfachter Restriktion, der bereits in Kapitel 3.2.2 vorgestellt wird.

Im ersten Teil wird der Spezialfall des schnellen Speichers betrachtet, das heißt $i_n^{\min}(x) = b^{\min} - x$ und $i_n^{\max}(x) = b^{\max} - x$ für alle $x \in E$ und $n = 0, 1, \ldots, N$, also mit Restriktionenmenge $D_n(x) = D(x) = \{a \mid b^{\min} - x \leq a \leq b^{\max} - x\}$.

Im zweiten Teil wird der Spezialfall betrachtet $i_n^{\min}$ und $i_n^{\max}$ linear fallend und $b^{\min} - x \leq i_n^{\min}(x) \leq i_n^{\max}(x) \leq b^{\max} - x$, die Restriktionenmenge vereinfacht sich zu $D_n(x) = \{a \mid i_n^{\min}(x) \leq a \leq i_n^{\max}(x)\}$.
In beiden Fällen ist D_n also ein „Streifen“, siehe Skizze in Kapitel 3.2.2.

Struktur der optimalen Politik und der Wertfunktion im Fall des „schnellen“ Speichers

Als schneller Speicher wird der Gasspeicher bezeichnet, bei dem es möglich (im Gegensatz zum optimal switching Fall nicht notwendig) ist, an einem Tag den Speicher zu füllen bzw. zu leeren. Das drückt auch die vereinfachte Restriktionenmenge $D(x) = \{a \mid b^{\min} - x \leq a \leq b^{\max} - x\}$ aus.

Desweiteren wird die Wahl des Preisprozesses eingeschränkt, vergleiche hierzu Kapitel 3.2.3. Es wird angenommen, dass $P_n = \prod_{k=1}^{n} Y_k$ mit $Y_1, Y_2, \ldots$ unabhängig identisch verteilt und Erwartungswert $0 < \mathbb{E}Y_1 =: \mu < \infty$. Wie in Kapitel 3.2.3 gezeigt ist dann $\mathbb{E}[P_n|P_{n-1}] = \mu P_{n-1}$. Außerdem gilt

$$\mathbb{E}[k(P_n)|P_{n-1}] = k(\mu P_{n-1}) \quad \text{und} \quad \mathbb{E}[e(P_n)|P_{n-1}] = e(\mu P_{n-1}). \tag{4.9}$$

Betrachtet wird das Problem mit der terminalen Gewinnfunktion

$$h_N(x, p) = e(p)(x - b^{\min}).$$

Damit sind auch nach Satz 4.1.2 die Integrabilitätsannahmen (A_N) erfüllt, denn es gilt mit geeigneten Konstanten $c_1, c_2 \in \mathbb{R}^+$

$$\mathbb{E}[k(P_n)|P_{n-1} = p] = k(\mu p) \leq c_1 k(p) \quad \text{und} \quad h_N(x, p) \leq (b^{\max} - b^{\min})e(p) \leq c_2 \cdot k(p).$$

Die zentrale Frage, mit der sich der folgende Teil beschäftigt, ist, ob es immer optimal ist am Anfang den Speicher zu füllen bzw. zu leeren und dann abzuwarten bis zum Ende oder ob es auch Fälle gibt, bei denen es optimal ist zu keinem Zeitpunkt ein- oder auszuspeichern.

Satz 4.2.11
Gilt $\mu < 1$, so ist für alle $n = 0, 1, \dots N$

$$v_n(x,p) = e(p)(x - b^{\min}) \tag{4.10}$$

und der Maximisator auf Stufe n ist gegeben durch

$$f_n^\star = b^{\min} - x. \tag{4.11}$$

Beweis
Im Fall $\mu < 1$ gilt

$$k(p) \geq e(p) > e(\mu p) > e(\mu p^2) > \cdots$$

Der Beweis wird per Rückwärtsinduktion geführt.

1. Induktionsanfang: Für $\underline{t = N}$ gilt

$$v_N(x,p) = h_N(x,p) = e(p)(x - b^{\min}).$$

2. Induktionshypothese: Für ein $n \in \{0, 1, \dots, N\}$ gilt die Behauptung.

3. Induktionsschritt von $\underline{n \text{ nach } n-1}$: Es gilt

$$\begin{aligned}
v_{n-1}(x,p) &= \sup_{a \in D(x)} \{h(p,a) + \mathbb{E}[v_n(x, P_n) | P_{n-1} = p]\} \\
&\overset{IH}{=} \sup_{a \in D(x)} \{h(p,a) + \mathbb{E}[e(P_n)(x+a) + b^{\min}(-e(P_n)) | P_{n-1} = p]\} \\
&= \sup_{a \in D(x)} \{h(p,a) + e(\mu p)x + ae(\mu p) + b^{\min}(-e(\mu p))\} \\
&= \max \left(\sup_{b^{\min} - x \leq a \leq 0} \{a \underbrace{(e(\mu p) - e(p))}_{<0} + xe(\mu p) + b^{\min}(-e(\mu p))\}, \right. \\
&\qquad \left. \sup_{0 \leq a \leq b^{\max} - x} \{a \underbrace{(e(\mu p) - k(p))}_{<0} + xe(\mu p) + b^{\min}(-e(\mu p))\} \right) \\
&= b^{\min}\,(e(\mu p) - e(p) - e(\mu p)) + x(e(\mu p) - e(\mu p) + e(p)) \\
&= (x - b^{\min})e(p)
\end{aligned}$$

und $f_{n-1}^\star = b^{\min} - x$. ∎

Bemerkung 4.2.12
Im vorherigen Satz wurde somit gezeigt, dass der Wert des Speichers durch $v_0(x_0, p) = e(p)(x_0 - b^{\min})$ gegeben ist und als erste Aktion $f_0^\star = b^{\min} - x_0$ optimal ist. Damit ist der neue Speicherstand zur Zeit 1: $x_0 + b^{\min} - x_0 = b^{\min}$.

Die Strategie ist also zusammengefasst: Am Anfang alles verkaufen und nicht weiter zu reagieren.
Intuitiv würde man diese Strategie auch erwarten, denn bei tendenziell fallenden Preisen ($\mu < 1$), würde man versuchen zu Beginn des Vertrages die höheren Preise zu nutzen und dann alles zu verkaufen.

Satz 4.2.13
Gilt $\mu > 1$ und $e(p) = k(p)$, so gilt für alle $n = 0, 1, \dots N$

$$v_n(x,p) = xe(p) + b^{\max}(e(\mu^{N-n}p) - e(p)) - b^{\min}e(\mu^{N-n}p) \tag{4.12}$$

und der Maximisator auf Stufe n

$$f_n^\star = b^{\max} - x. \tag{4.13}$$

Beweis
In diesem Fall gilt

$$e(p) < e(\mu p) < e(\mu^2 p) < \cdots .$$

Der Beweis wird mittels Rückwärtsinduktion geführt.

1. Induktionsanfang: Für $\underline{t = N}$ gilt

$$v_N(x,p) = h_N(x,p) = e(p)(x - b^{\min}).$$

   Für $\underline{t = N-1}$ gilt zudem

$$\begin{aligned} v_{N-1}(x,p) &= \sup_{a \in D(x)} \{h(p,a) + \mathbb{E}[e(P_N)(x + a - b^{\min}) | P_{N-1} = p]\} \\ &= \sup_{a \in D(x)} \{h(p,a) + e(\mu p)(x - b^{\min}) + e(\mu p)a\} \\ &= \sup_{b^{\min} - x \le a \le b^{\max} - x} \{a \underbrace{(e(\mu p) - e(p))}_{>0} + e(\mu p)(x - b^{\min})\} \\ &= (b^{\max} - x)(e(\mu p) - e(p)) + e(\mu p)(x - b^{\min}) \\ &= xe(p) + b^{\max}(e(\mu p) - e(p)) - b^{\min}e(\mu p) \end{aligned}$$

   und $f_{N-1}^\star = b^{\max} - x$.

2. Induktionshypothese: Die Behauptung gilt für eine $n \in \{0, 1, \dots, N\}$.

3. Induktionsschritt von n nach $n-1$. Es ist

$$\begin{aligned}
v_{n-1}(x,p) &= \sup_{a\in D(x)} \{h(p,a) + \mathbb{E}[\ v_n(x,P_n)|P_{n-1}=p]\} \\
&= \sup_{a\in D(x)} \{h(p,a) + \mathbb{E}[\ e(P_n)(x+a) + b^{\max}(e(\mu^{N-n}P_n) - e(P_n)) \\
&\qquad\qquad - b^{\min}(e(\mu^{N-n}P_n))|P_{n-1}=p]\} \\
&= \sup_{a\in D(x)} \{h(p,a) + e(\mu p)x + ae(\mu p) + b^{\max}(e(\mu^{N-n+1}p) - e(\mu p)) \\
&\qquad\qquad - b^{\min}e(\mu^{N-n+1}p)\} \\
&= \sup_{b^{\min}-x\le a\le b^{\max}-x} \{a\underbrace{(e(\mu p) - e(p))}_{>0} + xe(\mu p) + b^{\max}(e(\mu^{N-n+1}p) - e(\mu p)) \\
&\qquad\qquad - b^{\min}e(\mu^{N-(n-1)}p)\} \\
&= b^{\max}(e(\mu p) - e(p) + e(\mu^{N-(n-1)}p) - e(\mu p)) + x(e(\mu p) - e(\mu p) + e(p)) \\
&\quad - b^{\min}e(\mu^{N-(n-1)}p) \\
&= xe(p) + b^{\max}(e(\mu^{N-(n-1)}p) - e(p)) - b^{\min}e(\mu^{N-(n-1)}p)
\end{aligned}$$

und $f^\star_{n-1} = b^{\max} - x$. ∎

Bemerkung 4.2.14
Im vorherigen Satz wurde somit gezeigt, dass der Wert des Speichers $v_0(x_0,p) = x_0e(p) + b^{\max}(e(\mu^N p) - e(p)) - b^{\min}e(\mu^N p)$ ist und die optimale Strategie zu Beginn $f^\star_0 = b^{\max} - x_0$. Damit ist der neue Speicherstand im nächsten Schritt $x_0 + b^{\max} - x_0 = b^{\max}$.
Die optimale Strategie ist somit: Am Anfang wird der Speicher gefüllt und danach nichts mehr unternommen.
Da im Fall $\mu > 1$ erwartet wird, dass die Preise steigen, ist die Strategie auch intuitiv einsichtig. So nutzt man die am Anfang niedrigeren Preise aus, um günstig Gas zu kaufen, das dann zum Schluss (erwartet) an Wert gewinnt.

Mit den folgenden Beispielen wird gezeigt, dass im Fall $\mu > 1$ und $k(p) > e(p)$ sowohl die Strategie „auffüllen am Anfang und dann nichts tun“, sowie die Strategie „gar nichts tun“ optimal sein kann, je nachdem wie k, e und der Preisprozess (P_n) gewählt werden.

Als erstes wird das Beispiel der optimalen Strategie „nichts tun“ aufgeführt. Danach das für die andere Strategie.

Beispiel 4.2.15
Der Zeithorizont sei $N = 2$ und die Funktionen k und e seien gegeben durch

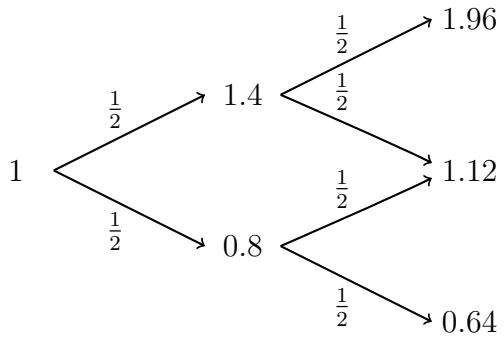
$$\begin{aligned}
k(p) &= (1+w_1)p + z_1 = 1.2p + 0.3 \\
e(p) &= (1-w_2)p - z_2 = 0.9p - 0.2.
\end{aligned}$$

Damit ist die terminale Gewinnfunktion

$$h_2(x,p) = e(p)(x - b^{\min}) = (0.9p - 0.2)(x - b^{\min}).$$

Der Preisprozess sei wie folgt definiert:

Es sei $P_0 = 1$ und $P_{n+1} = YP_n$ für $n = 0, 1$ mit $Y \in \{u, d\}$ und $P(Y = u) = P(Y = d) = \frac{1}{2}$. Die Struktur ist mit $u = 1.4$ und $d = 0.8$ an folgendem Baum dargestellt



Es ergeben sich folgende Werte für k und e

p	0.64	0.8	1	1.12	1.4	1.96
$k(p)$	1.068	1.26	1.5	1.644	1.98	2.652
$e(p)$	0.376	0.52	0.7	1.808	1.06	1.564

Um eine optimale Strategie zu ermitteln wird in der Zeit rückwärts gerechnet:

- Für $\underline{t = 2}$ gilt

$$v_2(x,p) = e(p)(x - b^{\min}) = \begin{cases} 0.376(x - b^{\min}), & p = 0.64, \\ 0.808(x - b^{\min}), & p = 1.12, \\ 1.564(x - b^{\min}), & p = 1.96. \end{cases}$$

- Für $\underline{t = 1}$ unterscheidet man die Fälle
  1. Ist $\underline{p = 0.8}$ so gilt

$$\begin{aligned} v_1(x, 0.8) &= \sup_{a \in D(x)} \{h(0.8, a) + \mathbb{E}[v_2(x + a, P)|0.8]\} \\ &= \sup_{a \in D(x)} \left\{ h(0.8, a) + (x + a - b^{\min}) \left(\frac{1}{2} \cdot 0.808 + \frac{1}{2} \cdot 0.376 \right) \right\} \\ &= \max \left(\sup_{a \le 0, a \in D} \{a \underbrace{(-0.52 + 0.592)}_{>0} + (x - b^{\min}) 0.592\} \right. \\ &\qquad \left. \sup_{a \ge 0, a \in D} \{a \underbrace{(-1.26 + 0.592)}_{<0} + (x - b^{\min}) 0.592\} \right) \\ &= 0.592\,(x - b^{\min}). \end{aligned}$$

2. Ist $\underline{p = 1.4}$ so gilt

$$\begin{aligned} v_1(x, 1.4) &= \sup_{a \in D(x)} \{h(1.4, a) + \mathbb{E}[v_2(x + a, P)|1.4]\} \\ &= \sup_{a \in D(x)} \left\{ h(1.4, a) + (x + a - b^{\min}) \left(\frac{1}{2} \cdot 0.808 + \frac{1}{2} \cdot 1.564 \right) \right\} \\ &= \max \left(\sup_{a \leq 0, a \in D} \{a \underbrace{(-1.06 + 1.186)}_{>0} + (x - b^{\min}) 1.186\} \right. \\ &\qquad \left. \sup_{a \geq 0, a \in D} \{a \underbrace{(-1.98 + 1.186)}_{<0} + (x - b^{\min}) 1.186\} \right) \\ &= 1.186\,(x - b^{\min}). \end{aligned}$$

Zusammengefasst ist zur Zeit $t = 1$ die optimale Strategie $f_1^\star \equiv 0$ und

$$v_1(x, p) = \begin{cases} 1.186(x - b^{\min}), & p = 1.4, \\ 0.592(x - b^{\min}), & p = 0.8. \end{cases}$$

- Für $\underline{t = 0}$ gilt

$$\begin{aligned} v_0(x, 1) &= \sup_{a \in D(x)} \{h(1, a) + \mathbb{E}[v_1(x + a, P)|1]\} \\ &= \sup_{a \in D(x)} \left\{ h(1, a) + (x + a - b^{\min}) \left(\frac{1}{2} \cdot 1.186 + \frac{1}{2} \cdot 0.592 \right) \right\} \\ &= \max \left(\sup_{a \leq 0, a \in D} \{a \underbrace{(-0.7 + 0.889)}_{>0} + (x - b^{\min}) 0.889\} \right. \\ &\qquad \left. \sup_{a \geq 0, a \in D} \{a \underbrace{(-1.5 + 0.889)}_{<0} + (x - b^{\min}) 0.889\} \right) \\ &= 0.889\,(x - b^{\min}). \end{aligned}$$

Die optimale Strategie ist demnach $f_0^\star \equiv 0$ und $v_0 = 0.889(x - b^{\min})$.

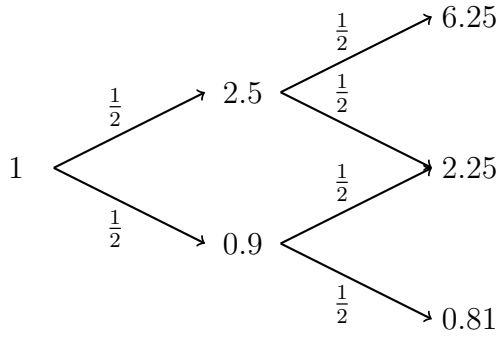
Beispiel 4.2.16
Wie im vorherigen Beispiel sei der Zeithorizont $N = 2$. Die Funktionen k und e seien gegeben durch

$$\begin{aligned} k(p) &= (1 + w_1)p + z_1 = 1.1p + 0.15 \\ e(p) &= (1 - w_2)p - z_2 = 0.8p - 0.05 \end{aligned}$$

und damit ist die terminale Gewinnfunktion

$$h_2(x, p) = e(p)(x - b^{\min}) = (0.8p - 0.05)(x - b^{\min}).$$

Der Preisprozess folge einem ähnlichen Schema, lediglich u und d seien anders gewählt. Es sei also $P_0 = 1$ und $P_{n+1} = YP_n$ für $n = 0, 1$ mit $Y \in \{u, d\}$ und $P(Y = u) = P(Y = d) = \frac{1}{2}$. Die Struktur ist mit $u = 2.5$ und $d = 0.9$ an folgendem Baum dargestellt



Die Werte der Funktionen k und e sind

p	0.81	0.9	1	2.25	2.5	6.25
$k(p)$	1.041	1.14	1.25	2.625	2.9	7.025
$e(p)$	0.598	0.67	0.75	1.75	1.95	4.95

Wieder wird die optimale Strategie mittels Rückwärtsrechnen ermittelt.

- Für $\underline{t = 2}$ gilt

$$v_2(x, p) = e(p)(x - b^{\min}) = \begin{cases} 0.598(x - b^{\min}), & p = 0.81, \\ 1.75(x - b^{\min}), & p = 2.25, \\ 4.95(x - b^{\min}), & p = 6.25. \end{cases}$$

- Für $\underline{t = 1}$ unterscheidet man die Fälle
  1. Ist $\underline{p = 0.9}$ so gilt

$$\begin{aligned} v_1(x, 0.9) &= \sup_{a \in D(x)} \{h(0.9, a) + \mathbb{E}[v_2(x + a, P)|0.9]\} \\ &= \sup_{a \in D(x)} \left\{ h(0.9, a) + (x + a - b^{\min}) \left(\frac{1}{2} \cdot 0.598 + \frac{1}{2} \cdot 1.75 \right) \right\} \\ &= \max \left(\sup_{a \le 0, a \in D} \{a \underbrace{(-0.67 + 1.174)}_{>0} + (x - b^{\min}) 1.174\} \right. \\ &\qquad \left. \sup_{a \ge 0, a \in D} \{a \underbrace{(-1.14 + 1.174)}_{>0} + (x - b^{\min}) 1.174\} \right) \\ &= (b^{\max} - x)\ 0.034 + (x - b^{\min})\ 1.174 \\ &= 1.14\ x + b^{\max}\ 0.034 - b^{\min}\ 1.174. \end{aligned}$$

2. Ist $\underline{p = 2.5}$ so gilt

$$\begin{aligned}
v_1(x, 2.5) &= \sup_{a \in D(x)} \{h(2.5, a) + \mathbb{E}[v_2(x + a, P)|2.5]\} \\
&= \sup_{a \in D(x)} \left\{ h(2.5, a) + (x + a - b^{\min}) \left(\frac{1}{2} \cdot 4.95 + \frac{1}{2} \cdot 1.75 \right) \right\} \\
&= \max \left(\sup_{a \le 0, a \in D} \{ a \underbrace{(-1.75 + 3.35)}_{>0} + (x - b^{\min}) 3.35 \} \right. \\
&\qquad \left. \sup_{a \ge 0, a \in D} \{ a \underbrace{(-2.625 + 3.35)}_{>0} + (x - b^{\min}) 3.35 \} \right) \\
&= (b^{\max} - x)\ 0.725 + (x - b^{\min})\ 3.35 \\
&= 2.625\ x + 0.725\ b^{\max} - 3.35\ b^{\min}.
\end{aligned}$$

Zusammengefasst ist zum Zeitpunkt $t = 1$ die optimale Strategie $f_1^\star = b^{\max} - x$ und damit

$$v_1(x, p) = \begin{cases} 1.14x + b^{\max} 0.034 - b^{\min} 1.174, & p = 0.9, \\ 2.625x + 0.725 b^{\max} - 3.35 b^{\min}, & p = 2.5. \end{cases}$$

- Für $\underline{t = 0}$ gilt

$$\begin{aligned}
v_0(x, 1) &= \sup_{a \in D(x)} \{h(1, a) + \mathbb{E}[v_1(x + a, P)|1]\} \\
&= \sup_{a \in D(x)} \left\{ h(1, a) + \frac{1}{2} v_1(x + a, 0.9) + \frac{1}{2} v_1(x + a, 2.5) \right\} \\
&= \sup_{a \in D(x)} \left\{ h(1, a) + (x + a) 1.8825 + b^{\max} 0.3795 - b^{\min} 2.262 \right\} \\
&= \max \left(\sup_{a \le 0, a \in D} \{ a \underbrace{(-0.75 + 1.8825)}_{>0} + x 1.8825 + b^{\max} 0.3795 - b^{\min} 2.262 \} \right. \\
&\qquad \left. \sup_{a \ge 0, a \in D} \{ a \underbrace{(-1.25 + 1.8825)}_{>0} + x 1.8825 + b^{\max} 0.3795 - b^{\min} 2.262 \} \right) \\
&= (b^{\max} - x)\ \ 0.6325 + x\ 1.8825 + b^{\max}\ 0.3795 - b^{\min}\ 2.262 \\
&= 1.25\ x\ + 1.012\ b^{\max} - 2.262\ b^{\min}.
\end{aligned}$$

Für $t = 0$ gilt also ebenfalls: $f_0^\star = b^{\max} - x$. Für $t = 1$ gilt damit für den Speicherstand $x_1 = x + b^{\max} - x = b^{\max}$.

Folglich ist die optimale Strategie am Anfang den Speicher zu füllen.

Struktur der optimalen Politik im Fall $i_n^{\max}, i_n^{\min}$ linear und Streifen

Es gelte nun $b^{\min} - x \leq i_n^{\min}(x) \leq i_n^{\max}(x) \leq b^{\max} - x$ für alle $n \in \{0, 1, \ldots, N\}$ und $x \in [b^{\min}, b^{\max}]$.

Die Restriktionenmenge sei also gegeben durch

$$D_n(x) = \{a \mid i_n^{\min}(x) \leq a \leq i_n^{\max}(x)\},$$

dabei seien $i_n^{\min}$ und $i_n^{\max}$ lineare fallende Funktionen.

Außerdem sei (P_n) (wieder allgemeiner) ein Markovscher Prozess. Dann existieren für Funktionen c_n und d_n Funktionen g_{n-1} und l_{n-1}, sodass

$$\mathbb{E}[c_n(P_n)|P_{n-1} = p] = g_{n-1}(p) \text{ und } \mathbb{E}[d_n(P_n)|P_{n-1} = p] = l_{n-1}(p).$$

Es wird angenommen, dass die Voraussetzungen aus Satz 4.1.2 oder Satz 4.1.1 erfüllt sind und damit die Integrabilitätsannahme gelte.

Satz 4.2.17
Im oben genannten Fall gilt

$$v_n(x,p) \text{ ist linear in } x, \text{ das heißt } v_n(x,p) = c_n(p)x + d_n(p)$$

und

$$f_n^\star(x,p) = \begin{cases} i_n^{\min}(x), & \gamma_n^1(p) < 0, \\ 0, & \gamma_n^2(p) \leq 0 \leq \gamma_n^1(p), \\ i_n^{\max}(x), & \gamma_n^2(p) > 0. \end{cases} \qquad \text{mit geeignetem } \gamma_n^2(p) \leq \gamma_n^1(p).$$

Beweis
Der Beweis wird mittels Rückwärtsinduktion geführt.

1. Induktionsanfang: Für $\underline{t = N}$ gilt

$$v_N(x,p) = h_N(x,p) = e(p)(x - b^{\min}) = c_N(p)x + d_N(p)$$

mit $c_N(p) = e(p)$ und $d_N(p) = -e(p) \cdot b^{\min}$.

Für $\underline{t = N-1}$ gilt

$$\begin{aligned}
v_{N-1}(x,p) &= \max \Bigg(\sup_{i_{N-1}^{\min}(x) \le a < 0} \{-e(p)a + \mathbb{E}[e(P_N)(x+a)|P_{N-1}=p]\} , \\
&\qquad \sup_{0 < a \le i_{N-1}^{\max}(x)} \{-k(p)a + \mathbb{E}[e(P_N)(x+a)|P_{N-1}=p]\} \Bigg) \\
&= \max \Bigg(\sup_{i_{N-1}^{\min}(x) \le a < 0} \{-e(p)a + (x+a)g_{N-1}(p)\} , \\
&\qquad \sup_{0 < a \le i_{N-1}^{\max}(x)} \{-k(p)a + (x+a)g_{N-1}(p)\} \Bigg) \\
&= \max \Bigg(\sup_{i_{N-1}^{\min}(x) \le a < 0} \{a(g_{N-1}(p) - e(p)) + x g_{N-1}(p))\} , \\
&\qquad \sup_{0 < a \le i_{N-1}^{\max}(x)} \{a(g_{N-1}(p) - k(p)) + x g_{N-1}(p)\} \Bigg) \\
&= x g_{N-1}(p) \; + \max \big(i_{N-1}^{\min}(x)(g_{N-1}(p) - e(p)) \mathbf{1}_{\{g_{N-1}(p)-e(p)<0\}}, \\
&\qquad\qquad i_{N-1}^{\max}(x)(g_{N-1}(p) - k(p)) \mathbf{1}_{\{g_{N-1}(p)-k(p)>0\}} \big) \\
&= x g_{N-1}(p) \quad + i_{N-1}^{\min}(x)(g_{N-1}(p) - e(p)) \mathbf{1}_{\{g_{N-1}(p)-e(p)<0\}} \\
&\qquad\qquad + i_{N-1}^{\max}(x)(g_{N-1}(p) - k(p)) \mathbf{1}_{\{g_{N-1}(p)-k(p)>0\}} \\
&=: x c_{N-1}(p) \quad + d_{N-1}(p)
\end{aligned}$$

Hierbei gilt die letzte Gleichheit, da $i^{\min}$ und $i^{\max}$ linear in x sind und die vorletzte, da nie beide Indikatoren gleichzeitig 1 werden, da sonst $k(p) < g_{N-1}(p) < e(p)$ gelten müsste, es ist aber $k(p) \ge e(p)$ vorausgesetzt.
Desweiteren ist zu erkennen, dass

$$f_{N-1}^{\star} = \begin{cases} i_{N-1}^{\min}(x), & \gamma_{N-1}^{1}(p) < 0, \\ 0, & \text{sonst} \\ i_{N-1}^{\max}(x), & \gamma_{N-1}^{2}(p) > 0 \end{cases}$$

mit $\gamma_{N-1}^{1}(p) = g_{N-1}(p) - e(p)$ und $\gamma_{N-1}^{2} = g_{N-1}(p) - k(p)$.
Es gilt, da $k(p) \ge e(p)$ ist, $\gamma_{N-1}^{1}(p) = g_{N-1}(p) - e(p) \ge g_{N-1}(p) - k(p) = \gamma_{N-1}^{2}(p)$.

2. Induktionshypothese: Die Behauptung gilt für ein $n \in \{0, 1, \ldots, N\}$.

3. Induktionsschritt von $\underline{n \text{ nach } n-1}$. Es gelte $v_n(x,p) = c_n(p)x + d_n(p)$, dann ist

$$\begin{aligned}
v_{n-1}(x,p) &= \max\Bigg(\sup_{i_{n-1}^{\min}(x)\le a\le 0}\{-e(p)a + \mathbb{E}[v_n(P_n, x+a)|P_{n-1}=p]\},\\
&\qquad \sup_{0\le a\le i_{n-1}^{\max}(x)}\{-k(p)a + \mathbb{E}[v_n(P_n, x+a)|P_{n-1}=p]\}\Bigg)\\
&= \max\Bigg(\sup_{i_{n-1}^{\min}(x)\le a\le 0}\{-e(p)a + \underbrace{\mathbb{E}[c_n(P_n)(x+a)|P_{n-1}=p]}_{=:g_{n-1}(p)\cdot(x+a)}\\
&\qquad\qquad + \underbrace{\mathbb{E}[d_n(P_n)|P_{n-1}=p]}_{=:l_{n-1}(p)}\},\\
&\qquad \sup_{0\le a\le i_{n-1}^{\max}(x)}\{-k(p)a + \mathbb{E}[c_n(P_n)(x+a)|P_{n-1}=p]\\
&\qquad\qquad + \mathbb{E}[d_n(P_n)|P_{n-1}=p]\}\Bigg)\\
&= \max\Bigg(\sup_{i_{n-1}^{\min}(x)\le a\le 0}\{a(g_{n-1}(p)-e(p)) + xg_{n-1}(p) + l_{n-1}(p)\},\\
&\qquad \sup_{0\le a\le i_{n-1}^{\max}(x)} a(g_{n-1}(p)-k(p)) + xg_{n-1}(p) + l_{n-1}(p)\Bigg)\\
&= xg_{n-1}(p) + l_{n-1}(p) + \max\big((g_{n-1}(p)-e(p))i_{n-1}^{\min}(x)\mathbf{1}_{\{g_{n-1}(p)-e(p)<0\}},\\
&\qquad\qquad (g_{n-1}(p)-k(p))i_{n-1}^{\max}(x)\mathbf{1}_{\{g_{n-1}(p)-k(p)>0\}}\big)\\
&\overset{s.o.}{=} xg_{n-1}(p) + l_{n-1}(p) + (g_{n-1}(p)-e(p))i_{n-1}^{\min}(x)\mathbf{1}_{\{g_{n-1}(p)-e(p)<0\}}\\
&\qquad + (g_{n-1}(p)-k(p))i_{n-1}^{\max}(x)\mathbf{1}_{\{g_{n-1}(p)-k(p)>0\}}\\
&\overset{s.o.}{=:} xc_{n-1}(p) + d_{n-1}(p)
\end{aligned}$$

und

$$f_{n-1}^{\star} = \begin{cases} i_{n-1}^{\min}(x), & \gamma_{n-1}^1(p) < 0,\\ 0, & \text{sonst}\\ i_{n-1}^{\max}(x), & \gamma_{n-1}^2(p) > 0 \end{cases}$$

mit $\gamma_{n-1}^1(p) = g_{n-1}(p) - e(p)$ und $\gamma_{n-1}^2(p) = g_{n-1}(p) - k(p)$.
Da $e(p) \le k(p)$ gilt auch hier:

$$\gamma_{n-1}^1(p) = g_{n-1}(p) - e(p) \ge g_{n-1}(p) - k(p) = \gamma_{n-1}^2(p). \qquad \blacksquare$$

Bemerkung 4.2.18
Im Beweis von Satz 4.2.17 erkennt man, dass im Fall $k(p) = e(p)$ die Gleichheit $\gamma^1(p) = \gamma^2(p)$ gilt.
Außerdem ist $\mathbb{E}[v_n(x, P_n)|P_{n-1}=p] = xg_{n-1}(p) + l_{n-1}(p)$.

4.3. Strukturaussagen zur Strategie im allgemeinen Fall

In Secomandi [34] wurde bereits gezeigt, dass es im Fall $i^{\max}$ und $i^{\min}$ konstant und (P_n) Markovsch vom aktuellen Preis und Zeitpunkt abhängige Grenzen des Speicherstandes gibt, die eine optimale Strategie maßgeblich bestimmen. Ist die aktuelle Gasmenge im Speicher unterhalb der niedrigeren Grenze, so ist es optimal möglichst bis genau zu dieser Grenze den Speicher zu füllen beziehungsweise, falls dies aufgrund der Restriktionen nicht möglich ist, $i^{\max}$ an Gas zu beziehen. Ähnlich verhält sich die optimale Strategie, wenn die aktuelle Gasmenge oberhalb der oberen Grenze ist - man versucht möglichst bis zu dieser Grenze Gas zu entnehmen. Ist der Füllstand zwischen beiden Grenzen, so wird keine Aktion durchgeführt.

Im ersten Teil dieses Unterkapitel wird die Aussage aus Secomandi [34] erweitert auf den regime switching Fall und $i^{\max}$ konkav und fallend bzw. $i^{\min}$ konvex und fallend. Der darauffolgende Teil beschäftigt sich mit Aussagen zu den oben genannten Grenzen, die in der optimalen Strategie eine wichtige Rolle spielen, nun wieder im Markov-Fall. Wie in Bemerkung 3.1.2 bereits erwähnt, werden alle Aussagen für den Fall mit Diskontierung gezeigt. Sei also im Folgenden $\beta \in (0, 1]$ der Diskontierungsfaktor.

4.3.1. Betrachtungen im Gasspeichermodell mit regime switching

In diesem Kapitel wird das Modell mit regime switching betrachtet, vorgestellt in Kapitel 3.2.3.

Sei also (R_n) eine Markov-Kette mit endlichem Zustandsraum S_R und Übergangsmatrix $(q_{jl})_{j,l\in S_R}$, die den Preisprozess (P_n) beeinflusst, und zwar so, dass (P_n, R_n) immer noch Markovsch ist. Außerdem gelten für den Übergangskern, die in Kapitel 3.2.3 vorgestellten Eigenschaften.

Genau wie im Modell ohne regime switching, vergleiche Kapitel 4.1, lässt sich eine Schrankenfunktion s finden.

Satz 4.3.1
Gegeben sei das Gasspeichermodell aus Kapitel 3.1 mit regime switching. Für die terminale Gewinnfunktion gelte mit $c_2, c_3 \in \mathbb{R}^+$

- $|h_N(x, p, r)| \leq c_2 k(p)$,
- $\mathbb{E}[k(P_{n+1})|P_n = p, R_n = r] \leq c_3 k(p)$.

Dann ist s mit $s(x, p, r) = k(p)$ eine Schrankenfunktion für das stochastische dynamische Programm.

Beweis
Der Beweis wird analog zum Beweis von 4.1.2 geführt. ∎

Auch in diesem Modell gilt eine Konkavitätsaussage.

Satz 4.3.2
Gegeben sei das Gasspeichermodell aus Kapitel 3.1 mit regime switching. Es gelte außerdem $i_n^{\max}$ konkav und $i_n^{\min}$ konvex für alle n und die Voraussetzungen für die Existenz einer Schrankenfunktion.
Ist $v \in \mathbb{S}_s^+$ und $x \mapsto v(x,p,r)$ konkav, so ist auch $x \mapsto T_n v(x,p,r)$ konkav auf $\mathbb{R}^+$ für jedes feste $p \in \mathcal{P}$ und $r \in S_R$.

Beweis
Der Beweis wird analog zum Beweis von 4.1.7 geführt. Beachte, dass D_n konvex ist. ∎

Sei im Folgenden die Funktion $h_N(\cdot,p,r)$ konkav.
Es gelte außerdem

$$D_n(x) = \{a | \max(b^{\min} - x, i_n^{\min}(x)) \leq a \leq \min(b^{\max} - x, i_n^{\max}(x))\},$$

wobei angenommen wird, dass $i_n^{\max}$ konkav und monoton fallend und $i_n^{\min}$ konvex und fallend ist.

Es ist

$$\begin{aligned} V_N(x,p,r) &= h_N(x,p,r), \\ V_n(x,p,r) &= \sup_{a \in D_n(x)} \left\{ h_n(p,r,a) + \beta \sum_{l \in S_R} q_{rl} \int V_{n+1}(x+a,p',l) Q_{n,r}(dp'|p) \right\}. \end{aligned}$$

Definiere weiter

$$\begin{aligned} U_N(x,p,r) &= 0, \\ U_n(x,p,r) &= \beta \sum_{l \in S_R} q_{rl} \int V_{n+1}(x,p',l) Q_{n,r}(dp'|p) \end{aligned}$$

den sogenannten „Continuation Value “.

Korollar 4.3.3
Die Funktionen $V_n(\cdot,p,r)$ und $U_n(\cdot,p,r)$ sind für alle $p \in \mathbb{R}^+, r \in S_R$ und $n = 1,\ldots,N$ konkav.

Beweis
Folgt aus 4.3.2 und der Tatsache, dass $V_N(\cdot,p,r)$ und $U_N(\cdot,p,r)$ konkav ($U_N(\cdot,p,r)$ sogar konstant) sind. ∎

Mit den Voraussetzungen und Korollar 4.3.3 lässt sich nun der Satz zur Struktur der optimalen Politk zeigen.

Wie schon erwähnt bedeutet dies, dass eine der drei Entscheidungen optimal ist: Ist der aktuelle Speicherstand unter einer bestimmten Grenze $\underline{b}_n(p,r)$, so wird eingespeichert bis zu dieser Grenze bzw. $i^{\max}$. Ist der Speicherstand über einer gewissen Grenze $\overline{b}_n(p,r)$ so wird Gas entnommen bis zu eben dieser Grenze oder $i^{\min}$. Ist der Speicherstand zwischen beiden Grenzen so ist es optimal einfach nichts zu tun.

Satz 4.3.4 (Struktur der optimalen Politik)
Gegeben sei das Gasspeichermodell aus Kapitel 3.1 mit regime switching.

Für jedes $n \in \{0,\dots,N-1\}$ existieren $\underline{b}_n(p,r)$ und $\overline{b}_n(p,r)$ mit $\underline{b}_n(p,r) \leq \overline{b}_n(p,r)$, sodass eine optimale Politik im Zustand (x,p,r) wie folgt aussieht

$$f_n^\star(x,p,r) = \begin{cases} \min(\underline{b}_n(p,r)-x, i_n^{\max}(x)), & b^{\min} \leq x < \underline{b}_n(p,r), \\ 0, & \underline{b}_n(p,r) \leq x \leq \overline{b}_n(p,r), \\ \max(\overline{b}_n(p,r)-x, i_n^{\min}(x)), & \overline{b}_n(p,r) < x \leq b^{\max}. \end{cases}$$

Beweis
Der Beweis wird analog zum Beweis von Theorem 1 in Secomandi [34] geführt:

Sei $n \in \{0,\dots,N-1\}$ beliebig aber fest.
Zunächst werden die Einschränkungen der Optimierung im Zeitpunkt n gelockert und es wird über alle $z = a + x \in \left[b^{\min}, b^{\max}\right]$ optimiert. Es wird also folgendes Problem angegangen

$$\max_{z\in[b^{\min},b^{\max}]} h_n(z-x,p,r) + U_n(z,p,r), \tag{1}$$

wobei $p \in \mathcal{P}$ und $r \in S_R$.
Wird das Optimierungsproblem in den Fall $z \geq x$ und $z \leq x$ zerteilt, so erhält man

$$\max\left(\max_{z\in[x,b^{\max}]} U_n(z,p,r) - k(p)z + k(p)x, \max_{z\in[b^{\min},x]} U_n(z,p,r) - e(p)z + e(p)x\right)$$

Die einzelnen Teilprobleme werden mit (2) ($z \geq x$) bzw. (3) ($z \leq x$) bezeichnet.

Betrachte nun die Spezialfälle: (i) $x = b^{\min}$ und (ii) $x = b^{\max}$, dann vereinfacht sich das Problem zu

$$\text{(i)} \max_{z\in[b^{\min},b^{\max}]} \underbrace{U_n(z,p,r) - k(p)z}_{=:f_4(z,p,r)} + k(p)b^{\min} \tag{4}$$

$$\text{(ii)} \max_{z\in[b^{\min},b^{\max}]} \underbrace{U_n(z,p,r) - e(p)z}_{=:f_5(z,p,r)} + e(p)b^{\max} \tag{5}$$

Seien $\underline{b}_n(p,r)$ und $\overline{b}_n(p,r)$ die Maximisatoren von (4) bzw. (5).

Zwischenbehauptung:

$$\underline{b}_n(p,r) \leq \overline{b}_n(p,r)$$

Beweis der Zwischenbehauptung:
Es gilt

$$g(z,p,r) := f_4(z,p,r) - f_5(z,p,r) = \underbrace{(e(p)-k(p))}_{<0} z$$

ist fallend in z. Also ist $f_4(z,p,r)$ gleich $f_5(z,p,r)$ zuzüglich eine in z fallende Funktion und daher ist das Maximum von f_4 kleiner gleich dem Maximum von f_5 für alle p,r fest aber beliebig. Daraus folgt die Zwischenbehauptung.

Betrachtet wird nun die Optimierung für $x \in [b^{\min}, \underline{b}_n(p,r))$.
Offensichtlich gilt, dass $\underline{b}_n(p,r)$ (2) maximiert.
Außerdem gilt, dass x (3) maximiert, denn: Für jedes zulässige z gilt $z \leq x < \underline{b}_n(p,r) \leq \overline{b}_n(p,r)$. Nun maximiert $\overline{b}_n(p,r)$ das Optimierungsproblem (5), was aber nicht zulässig ist, also wird - aufgrund der Konkavität der zu maximierenden Funktion in z - das „nächste“ zulässige z (das ist x) genommen.
Bleibt nun zu bestimmen, ob nun $\underline{b}_n(p,r)$ oder x (1) maximiert.
Da x zulässig für (2) ist und $\underline{b}_n(p,r)$ dieses Problem optimiert, gilt

$$U_n(\underline{b}_n(p,r),p,r) - k(p)\underline{b}_n(p,r) + k(p)x \geq U_n(x,p,r) - k(p)x + k(p)x = U_n(x,p,r).$$

Setzt man andererseits den Maximisator x in (3) ein, so erhält man $U_n(x,p,r)$. Also wird $(1) = \max((2),(3))$ maximiert durch $\underline{b}_n(p,r)$.

Wird die Optimierung für $x \in (\overline{b}_n(p,r), b^{\max}]$ betrachtet, erhält man durch ähnliche Argumentation, dass $\overline{b}_n(p,r)$ (1) maximiert.

Für $x \in [\underline{b}_n(p,r), \overline{b}_n(p,r)]$ gilt (ähnliche Argumentation wie im 1. Fall bei der Optimierung von (3)), dass x sowohl (2) also auch (3) maximiert und demnach auch (1).

Werden nun die zulässigen z in (1) weiter eingeschränkt, genauer gesagt fordert man zusätzlich $i_n^{\min}(x) + x \leq z \leq i_n^{\max}(x) + x$, so ergibt sich das Problem

$$\max_{z \in [(i_n^{\min}(x)+x) \vee b^{\min},\ b^{\max} \wedge (i_n^{\max}(x)+x)]} h_n(z-x,p,r) + U_n(z,p,r) \qquad (1^\star)$$

und man erhält als Maximisator

$$z^\star(x,p,r) = \begin{cases} \underline{b}_n(p,r) \wedge (i_n^{\max}(x)+x), & x \in [b^{\min}, \underline{b}_n(p,r),) \\ x, & x \in [\underline{b}_n(p,r), \overline{b}_n(p,r)], \\ \overline{b}_n(p,r) \vee (i_n^{\min}(x)+x), & x \in (\overline{b}_n(p,r), b^{\max}]. \end{cases}$$

Übertragt man dieses Resultat nun auf das ursprüngliche Problem ($a = z - x$), erhält man

$$f_n^\star(x,p,r) = \begin{cases} \min(\underline{b}_n(p,r) - x, i_n^{\max}(x)), & b^{\min} \leq x < \underline{b}_n(p,r), \\ 0, & \underline{b}_n(p,r) \leq x \leq \overline{b}_n(p,r), \\ \max(\overline{b}_n(p,r) - x, i_n^{\min}(x)), & \overline{b}_n(p,r) < x \leq b^{\max}. \end{cases}$$

■

Bemerkung 4.3.5
Satz 4.3.4 stützt auch das Ergebnis aus Kapitel 4.2 im Spezialfall $b^{\max} - x \leq i^{\max}(x)$ und $i^{\min}(x) \leq b^{\min} - x$ im Fall ohne regime switching. Nach Satz 4.3.4 ist die optimale Politik auf Stufe n

$$f_n^\star(x,p) = \begin{cases} \underline{b}_n(p) - x, & x \in [b^{\min}, \underline{b}_n(p)) \\ 0, & x \in [\underline{b}_n(p), \overline{b}_n(p)] \\ \overline{b}_n(p) - x, & x \in (\overline{b}_n(p), b^{\max}]. \end{cases}$$

Es gilt also $\underline{b}_n(p)$ und $\overline{b}_n(p)$ aus

$$\text{(i)} \quad \max_{z \in [b^{\min}, b^{\max}]} U_n(z,p) - k(p)z = \max_{z \in [b^{\min}, b^{\max}]} \mathbb{E}_n[V_{n+1}(z,P)] - k(p)z$$
$$\text{(ii)} \quad \max_{z \in [b^{\min}, b^{\max}]} U_n(z,p) - e(p)z = \max_{z \in [b^{\min}, b^{\max}]} \mathbb{E}_n[V_{n+1}(z,P)] - e(p)z$$

zu berechnen. Hierbei sei kurz $\mathbb{E}_n[V_{n+1}(z,P)] := \mathbb{E}[V_{n+1}(z,P_{n+1})|P_n = p]$.
Es lässt sich zeigen (siehe Beweis von Satz 4.2.17), dass V_n in x linear ist, das heißt dass gilt

$$V_n(x,p) = c_n(p)x + d_n(p).$$

Außerdem gilt ((P_n) Markovsch, genaueres siehe Bemerkungen vor Satz 4.2.17)

$$\mathbb{E}_n[c_{n+1}(P)] = g_n(p) \text{ und } \mathbb{E}_n[d_{n+1}(P)] = l_n(p).$$

Betrachtet man das Optimierungsproblem (i) so ergibt sich

$$\begin{aligned} \max_{z \in [b^{\min}, b^{\max}]} \mathbb{E}_n[V_{n+1}(z,P)] - k(p)z &= \max_{z \in [b^{\min}, b^{\max}]} \mathbb{E}_n[c_{n+1}(P)z + d_{n+1}(P)] - k(p)z \\ &= \max_{z \in [b^{\min}, b^{\max}]} g_n(p)z + l_n(p) - k(p)z \\ &= \max_{z \in [b^{\min}, b^{\max}]} (g_n(p) - k(p))z + l_n(p) \end{aligned}$$

woraus für (i) folgt

$$\underline{b}_n(p) = \begin{cases} b^{\max}, & g_n(p) - k(p) \geq 0 \\ b^{\min}, & g_n(p) - k(p) < 0 \end{cases}$$

Betrachtet man das Optimierungsproblem (ii) so ergibt sich

$$\begin{aligned} \max_{z \in [b^{\min}, b^{\max}]} \mathbb{E}_n[V_{n+1}(z,P)] - e(p)z &= \max_{z \in [b^{\min}, b^{\max}]} \mathbb{E}_n[c_{n+1}(P)z + d_{n+1}(P)] - e(p)z \\ &= \max_{z \in [b^{\min}, b^{\max}]} g_n(p)z + l_n(p) - e(p)z \\ &= \max_{z \in [b^{\min}, b^{\max}]} (g_n(p) - e(p))z + l_n(p) \end{aligned}$$

woraus für (i) folgt

$$\overline{b}_n(p) = \begin{cases} b^{\max}, & g_n(p) - e(p) \geq 0 \\ b^{\min}, & g_n(p) - e(p) < 0 \end{cases}$$

Da $k(p) \geq e(p)$ werden nun 3 Fallunterscheidungen für $f_n^\star$ betrachtet.

1. Fall: $g_n(p) - k(p) \geq 0$ (hieraus folgt auch $g_n(p) - e(p) \geq 0$)
   und es gilt $\underline{b}_n(p) = \overline{b}_n(p) = b^{\max}$.

2. Fall: $g_n(p) - e(p) < 0$ (hieraus folgt auch $g_n(p) - k(p) \leq 0$)
   und es gilt $\underline{b}_n(p) = \overline{b}_n(p) = b^{\min}$.

3. Fall: $g_n(p) - k(p) < 0 \leq g_n(p) - e(p)$ und es gilt $\underline{b}_n(p) = b^{\min}$, $\overline{b}_n(p) = b^{\max}$.

Hieraus ergibt sich zusammenfassend

$$f_n^\star = \begin{cases} b^{\max} - x, & g_n(p) - k(p) > 0, \\ 0, & \text{sonst}, \\ b^{\min} - x, & g_n(p) - e(p) < 0. \end{cases}$$

Dieses Ergebnis entspricht auch der Aussage von Satz 4.2.17 im Fall $i_n^{\max}(x) = b^{\max} - x$, $i_n^{\min}(x) = b^{\min} - x$.
Festzustellen ist, dass die Strategie hier nur von p, nicht aber von x abhängt.

Interessant ist, dass es nach Satz 4.3.4 nicht immer optimal ist alles was möglich wäre einzuspeichern bzw. auszuspeichern. Ein Beispiel, dass es bereits im Fall $i^{\max}$ konstant ($b^{\min} + i^{\max} < b^{\max}$) solche Fälle gibt, findet sich in Secomandi [34] Beispiel 1.
Dennoch vereinfacht die Kenntnis der Struktur der optimalen Politik die numerische Bewertung eines Gasspeichers. Man muss zwar, um die Grenzen der Politik zu berechnen, für jeden möglichen Preiszustand eine Optimierung über alle Speicherzustände durchführen. Jedoch sind nicht alle möglichen Aktionen für jeden Speicherzustand in Betracht ziehen. Genauer wird hierauf in Kapitel 6 eingegangen.

4.3.2. Betrachtung der Strategieschranken im Markov-Fall

Im Folgenden werden die Schranken $\underline{b}_n$ und $\overline{b}_n$ diskutiert, und zwar im Fall ohne regime switching und mit terminaler Gewinnfunktion $h_N(x,p)$, für die gilt: $x \mapsto h_N(x,p)$ ist konkav auf $(b^{\min} - c, b^{\min} + c)$ für ein $c \in \mathbb{R}^+$. Genauer gesagt, es soll gezeigt werden, dass die Schranken fallend sind. Hierzu sind einige Vorbereitungen nötig.

Es werden die Ableitungen von V als

$$\begin{aligned} V'_{n+1}(x,p) &:= \lim_{\varepsilon \searrow 0} \frac{V_{n+1}(x,p) - V_{n+1}(x-\varepsilon,p)}{\varepsilon} \\ V'_{n+1}(b^{\min},p) &:= \lim_{x \searrow b^{\min}} V'_{n+1}(x,p) \\ V'_{n+1}(b^{\max},p) &:= \lim_{x \nearrow b^{\max}} V'_{n+1}(x,p) \end{aligned}$$

und von U als

$$\begin{aligned} U_n'(x,p) &:= \lim_{\varepsilon \searrow 0} \frac{U_n(x,p) - U_n(x-\varepsilon,p)}{\varepsilon} \\ U_n'(b^{\min},p) &:= \lim_{x \searrow b^{\min}} U_n'(x,p) \\ U_n'(b^{\max},p) &:= \lim_{x \nearrow b^{\max}} U_n'(x,p) \end{aligned}$$

definiert. Es ist zu bemerken, dass die Grenzwerte existieren, da U bzw. V konkav sind. An den Preisprozess werden nun weitere Forderungen gestellt:

Voraussetzungen an den Preisprozess

(V1) Es gilt $\beta\mathbb{E}[P_{n+1}|P_n = p] < \infty$.

(V2) Die bedingte Verteilung von P_{n+1} gegeben $P_n = p$ ist stochastisch wachsend in p. Insbesondere gilt $\mathbb{E}[P_{n+1}|P_n = p]$ ist wachsend in p.

(V3) Es gilt $\beta\mathbb{E}[(1+w_1)P_{n+1}|P_n = p] - (1-w_2)p$ ist fallend in p.

Bemerkung 4.3.6
Die bedingte Verteilung $[Y|X = x]$ ist stochastisch wachsend in x, falls $[Y|X = s] \leq_{st} [Y|X = t]$ für alle $s < t$. Die Relation „$\leq_{st}$“ bezeichnet hierbei die gewöhnliche stochastische Ordnung (vergleiche hierzu zum Beispiel Müller u. Stoyan [27]).

Mit diesen Voraussetzungen lassen sich folgende Lemmata beweisen, siehe online companion /Appendix von Secomandi [34] Lemma 3-7.

Lemma 4.3.7
Unter den Voraussetzungen (V1)-(V3) gelten

a) $p \mapsto \beta\mathbb{E}[P_{n+1}|P_n = p] - p$ ist monoton fallend.

b) $p \mapsto \beta\mathbb{E}[k(P_{n+1})|P_n = p] - k(p)$ ist monoton fallend.

c) $p \mapsto \beta\mathbb{E}[e(P_{n+1})|P_n = p] - k(p)$ ist monoton fallend.

d) Sei g eine wachsende Funktion mit $g(p) \leq k(p)$ für alle $p \in \mathcal{P}$, so gilt
   (i) $p \mapsto \beta\mathbb{E}[g(P_{n+1})|P_n = p] - k(p)$ ist monoton fallend.
   (ii) $p \mapsto \beta\mathbb{E}[g(P_{n+1})|P_n = p] - e(p)$ ist monoton fallend.

Lemma 4.3.8
Unter Voraussetzung (V1) gilt

$$U_n'(x,p) = \beta\mathbb{E}[V_{n+1}'(x, P_{n+1})|P_n = p].$$

Beweis
Lemma 4.3.8 folgt hauptsächlich aus dem Satz von der dominierten Konvergenz.
Falls V_n für alle n in x Lipschitz stetig mit Lipschitz-Konstanten $K_n(p) < \infty$ ist (wird noch gezeigt), so folgt

$$\frac{|V_n(x,p) - V_n(x-\varepsilon,p)|}{\varepsilon} \leq K_j(p)$$

und der Rest folgt mit dominierter Konvergenz (vergleiche Beweis von Lemma 7 im online companion /Appendix von Secomandi [34]).

Zur Lipschitzstetigkeit, vergleiche auch Hinderer [21] (Beweis durch Rückwärtsinduktion):

Sei $\underline{n = N}$. Für jede konkave Funktion f gilt, dass sie auf jeder kompakten Teilmenge des Inneren ihres Definitionsbereiches Lipschitz stetig ist (vgl. Theorem 2.2 in Gruber [19]). Da vorausgesetzt wird, dass $x \mapsto h_N(x,p)$ konkav ist auf $(b^{\min} - c, b^{\max} + c)$ für ein $c \in \mathbb{R}^+$, gilt also $x \mapsto h_N(x,p)$ ist Lipschitz stetig auf $[b^{\min}, b^{\max}]$. Folglich

$$|V_N(x_2,p) - V_N(x_1,p)| = |h_N(x_2,p) - h_N(x_1,p)| \leq K_N(p)|x_2 - x_1|.$$

Für den Induktionsschritt $\underline{n+1 \to n}$ sei oBdA $x_1 \leq x_2$. Dann gilt nach Satz 4.1.11 für die Maximisatoren zum einen

$$f_n^\star(x_1) \geq f_n^\star(x_2) \tag{4.14}$$

und zum anderen

$$x_1 + f_n^\star(x_1) \leq x_2 + f_n^\star(x_2). \tag{4.15}$$

Damit gilt

$$\begin{aligned}
V_n(x_2,p) - V_n(x_1,p) = & \sup_{a \in D_n(x_2)} \{-h_n(p)a + \mathbb{E}[V_{n+1}(x_2+a, P_{n+1})|P_n = p]\} \\
& - \sup_{a \in D_n(x_1)} \{-h_n(p)a + \mathbb{E}[V_{n+1}(x_1+a, P_{n+1})|P_n = p]\} \\
= & - h_n(p, f_n^\star(x_2)) + \beta\mathbb{E}[V_{n+1}(x_2 + f_n^\star(x_2), P_{n+1})|P_n = p] \\
& + h_n(p, f_n^\star(x_1)) - \beta\mathbb{E}[V_{n+1}(x_1 + f_n^\star(x_1), P_{n+1})|P_n = p] \\
= & \beta\ \mathbb{E}[V_{n+1}(x_2 + f_n^\star(x_2), P_{n+1}) - V_{n+1}(x_1 + f_n^\star(x_1), P_{n+1})|P_n = p] \\
& + (h_n(p, f_n^\star(x_1)) - h_n(p, f_n^\star(x_2))) \\
\leq & \beta\ \mathbb{E}[K_{n+1}(P_{n+1})(x_2 + f_n^\star(x_2) - x_1 - f_n^\star(x_1))|P_n = p].
\end{aligned}$$

Hierbei folgt die Abschätzung des ersten Summanden aus der Induktionsvoraussetzung und (4.15), die Abschätzung des zweiten Summanden durch Null folgt aus der Tatsache, dass $a \mapsto h_n(p,a)$ fallend ist und (4.14) gilt.
Durch weitere Abschätzung erhält man

$$\begin{aligned}
V_n(x_2,p) - V_n(x_1,p) \leq & \quad \beta\ \mathbb{E}[K_{n+1}(P_{n+1})(x_2 + f_n^\star(x_2) - x_1 - f_n^\star(x_1))|P_n = p] \\
= & \quad \beta\ \mathbb{E}[K_{n+1}(P_{n+1})|P_n = p](x_2 - x_1 + f_n^\star(x_2) - f_n^\star(x_1)) \\
\overset{(4.14)}{\leq} & \ \overline{K}_n(p)(x_2 - x_1).
\end{aligned}$$

Andererseits gilt

$$\begin{aligned} V_n(x_2,p) - V_n(x_1,p) =& \beta\ \mathbb{E}[V_{n+1}(x_2 + f_n^\star(x_2), P_{n+1}) - V_{n+1}(x_1 + f_n^\star(x_1), P_{n+1})|P_n = p] \\ &+ (h_n(p, f_n^\star(x_1)) - h_n(p, f_n^\star(x_2))) \\ \geq& -\beta\ \mathbb{E}[K_{n+1}(P_{n+1})|P_n = p](x_2 - x_1 + f_n^\star(x_2) - f_n^\star(x_1)) \\ &- L_h(p)(f_n^\star(x_1) - f_n^\star(x_2)) \\ \geq& -\underline{K}_n(p)(x_2 - x_1). \end{aligned}$$

Hierbei gelten die Abschätzungen in der zweiten Zeile zum einen wegen der Induktionsvoraussetzung und wegen (4.15), zum anderen wegen der Lischitzstetigkeit von $a \mapsto h(p,a)$ mit Lipschitz-Konstante $L_h(p)$ und (4.14). Die Abschätzungen in der letzten Zeile folgen wiederum aus (4.14) und (4.15).

Insgesamt gilt also

$$|V_n(x_2,p) - V_n(x_1,p)| \leq K_n(p)(x_2 - x_1),$$

wobei $K_n(p) = \max(\underline{K}_n(p), \overline{K}_n(p))$ und $K_n(p) < \infty$ wegen (V1). ∎

Weitere Voraussetzungen, die an $i^{\max}$ und $i^{\min}$ gestellt werden, sind

Voraussetzungen an den $i^{\max}$, $i^{\min}$

(V4) $i^{\max}$ sei konkav und fallend und differenzierbar und es gelte $|(i^{\max})'(x)| < 1$.

(V5) $i^{\min}$ sei konvex und fallend und differenzierbar und es gelte $|(i^{\min})'(x)| < 1$.

Satz 4.3.9
Unter (V1) - (V5) sind die Funktionen $\underline{b}_n$ und $\overline{b}_n$ aus Satz 4.3.4 monoton fallend.

Beweis
Der Beweis wird in ähnlicher Weise wie der Beweis von Theorem 2 in Secomandi [34] geführt:

Betrachtet werden die Maximierungsprobleme:

$$\max_{z \in [b^{\min}, b^{\max}]} U_n(z,p) - k(p)z$$
$$\max_{z \in [b^{\min}, b^{\max}]} U_n(z,p) - e(p)z$$

Zunächst ist zu bemerken, dass $\overline{b}(p)$ und $\underline{b}_n(p)$ in p fallend sind, falls

$$U_n'(z,p) - k(p) \text{ fallend in } p \text{ bzw.}$$
$$U_n'(z,p) - e(p) \text{ fallend in } p.$$

Der Beweis hierfür erfolgt durch Rückwärtsinduktion. Es werden **2 Behauptungen** aufgestellt:

B1 a) $U'_k(z,p) - k(p)$ ist fallend in p.

b) $U'_k(z,p) - e(p)$ ist fallend in p.

B2 $U'_k(z,p)$ ist wachsend in p.

Es folgt die Induktion für beide Behauptungen gleichzeitig.

IA: Für die Zeiten N und $N-1$.
Es gilt

$$U'_N(z,p) - k(p) = -k(p)$$

ist fallend in p. (Genauso $U'_N(z,p) - e(p)$.)

Für $N-1$ gilt

$$U_{N-1}(z,p) = \beta E[V_N(z,P_N)|P_{N-1}=p] = \beta E[e(P_N)|P_{N-1}=p]z$$

und damit gilt

$$U'_{N-1}(z,p) - k(p) = \beta E[e(P_N)|P_{N-1}=p] - k(p),$$

ist fallend in p wegen Lemma 4.3.7 Teil c). Genauso gilt dies für $U'_{N-1}(z,p) - e(p)$ (Lemma 4.3.7 Teil a)).

Außerdem ist wegen (V2) U'_{N-1} wachsend in p.

IH: B1 und B2 gelten für alle $n+1 \leq k \leq N$.

IS: Für Zeit $n+1$ nach n.

Da

$$U_n(x,p) = \beta E[V_{n+1}(x,P_{n+1})|P_n=p] \text{ und } U'_n(x,p) = \beta E[V'_{n+1}(x,P_{n+1})|P_n=p]$$

wird zunächst $V_{n+1}(x,p)$ betrachtet.

Da es nach Satz 4.3.4 fünf Möglichkeiten für die optimale Entscheidung gibt, lässt sich V_{n+1} wie folgt aufteilen:

$$\begin{aligned} V_{n+1}(x,p) = \quad & \mathbf{1}_{①}\left(-e(p)i^{\min}(x) + U_{n+1}(x+i^{\min}(x),p)\right) \\ + \ & \mathbf{1}_{②}\left(-e(p)(\bar{b}_{n+1}(p)-x) + U_{n+1}(\bar{b}_{n+1}(p),p)\right) \\ + \ & \mathbf{1}_{③}\, U_{n+1}(x,p) \\ + \ & \mathbf{1}_{④}\left(-k(p)(\underline{b}_{n+1}(p)-x) + U_{n+1}(\underline{b}_{n+1}(p),p)\right) \\ + \ & \mathbf{1}_{⑤}\left(-k(p)i^{\max}(x) + U_{n+1}(x+i^{\max}(x),p)\right) \end{aligned}$$

mit

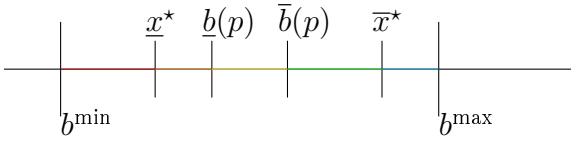
$$
\begin{aligned}
① &= \{\overline{b}_{n+1}(p) < x \leq b^{\max}, x + i^{\min}(x) > \overline{b}_{n+1}(p)\} \\
&= \{x \leq b^{\max}, x + i^{\min}(x) > \overline{b}_{n+1}(p)\} \\
② &= \{\overline{b}_{n+1}(p) < x \leq b^{\max}, x + i^{\min}(x) < \overline{b}_{n+1}(p)\} \\
③ &= \{\underline{b}_{n+1}(p) \leq x \leq \overline{b}_{n+1}(p)\} \\
④ &= \{b^{\min} \leq x < \underline{b}_{n+1}(p), x + i^{\max}(x) > \underline{b}_{n+1}(p)\} \\
⑤ &= \{b^{\min} \leq x < \underline{b}_{n+1}(p), x + i^{\max}(x) < \underline{b}_{n+1}(p)\} \\
&= \{b^{\min} \leq x, x + i^{\max}(x) < \underline{b}_{n+1}(p)\}
\end{aligned}
$$

Ein Exkurs zur Veranschaulichung:

Es gilt ① ∪ ② ∪ ③ ∪ ④ ∪ ⑤ $= [b^{\min}, b^{\max}]$. Gilt zusätzlich ⓘ $\neq \emptyset$ für alle i, so existieren $\overline{x}^\star(p)$ und $\underline{x}^\star(p)$, sodass

$$
⑤ = [b^{\min}, \underline{x}^\star(p)), ④ = [\underline{x}^\star(p), \underline{b}(p)), ③ = [\underline{b}(p), \overline{b}(p)],
$$
$$
② = (\overline{b}(p), \overline{x}^\star(p)], ① = (\overline{x}^\star(p), b^{\max}].
$$

Am Bild:



Außerdem gilt $x^\star(p)$ ist fallend in p, da

$$
x^\star(p) + i(x^\star(p)) = b(p)
$$

woraus

$$
\frac{\partial x^\star(p)}{\partial p} + i'(x^\star(p)) \frac{\partial x^\star(p)}{\partial p} = \frac{\partial b(p)}{\partial p}
$$

folgt und damit

$$
\frac{\partial x^\star(p)}{\partial p} = \underbrace{\frac{1}{1 + i'(x^\star(p))}}_{>0} \underbrace{\frac{\partial b(p)}{\partial p}}_{<0} < 0.
$$

Die Funktion V_{n+1} wird komponentenweise abgeleitet (nach x) und eine 0 wird ergänzt:

$$
\begin{aligned}
V'_{n+1}(x,p) = \quad & \mathbf{1}_{①} \left(-e(p)(i^{\min})'(x) + U'_{n+1}(x + i^{\min}(x), p)(1 + (i^{\min})'(x))\right) \\
+ \ & \mathbf{1}_{②} e(p) + \mathbf{1}_{③} U'_{n+1}(x,p) + \mathbf{1}_{④} k(p) \\
+ \ & \mathbf{1}_{⑤} \left(-k(p)(i^{\max})'(x) + U'_{n+1}(x + i^{\max}(x), p)(1 + (i^{\max})'(x))\right) \\
+ \ & e(p)\mathbf{1}_{①} - e(p)\mathbf{1}_{①} + k(p)\mathbf{1}_{⑤} - k(p)\mathbf{1}_{⑤}
\end{aligned}
$$

Durch Umordnen ergibt sich

$$\begin{aligned} V'_{n+1}(x,p) = \quad & \mathbf{1}_{①}\,(U'_{n+1}(x+i^{\min}(x),p)-e(p))(1+(i^{\min})'(x))) \\ + \ & \mathbf{1}_{⑤}\,(U'_{n+1}(x+i^{\max}(x),p)-k(p))(1+(i^{\max})'(x)) \\ + \ & \mathbf{1}_{③}\,U'_{n+1}(x,p) + \left(\mathbf{1}_{①}+\mathbf{1}_{②}\right)e(p) + \left(\mathbf{1}_{⑤}+\mathbf{1}_{④}\right)k(p). \end{aligned}$$

Die Funktionen

$$\begin{aligned} f^1(x,p) = \ & \mathbf{1}_{①}\,(U'_{n+1}(x+i^{\min}(x),p)-e(p))(1+(i^{\min})'(x))) \\ + \ & \mathbf{1}_{⑤}\,(U'_{n+1}(x+i^{\max}(x),p)-k(p))(1+(i^{\max})'(x)) \\ f^2(x,p) = \ & \mathbf{1}_{③}\,U'_{n+1}(x,p) + \left(\mathbf{1}_{①}+\mathbf{1}_{②}\right)e(p) + \left(\mathbf{1}_{⑤}+\mathbf{1}_{④}\right)k(p) \end{aligned}$$

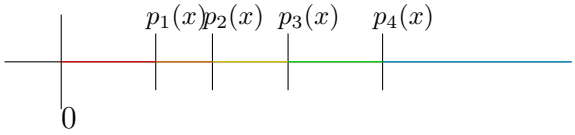
werden später auf ihr Verhalten in p untersucht.

Hierfür sind folgende **Beobachtungen** wichtig:

Da das Verhalten der Funktionen in p bei festem x betrachtet wird, teilt man den Zustandsraum $\mathcal{P}$ des Preisprozessen in 5 Regionen auf. Da die Funktionen $b(p)$ und $x^\star(p)$ fallend sind, ergeben sich zu ⓘ äquivalente „Intervalle“

$$\begin{aligned} ①^\star &= (p_4(x),\infty)\cap\mathcal{P} \\ ②^\star &= (p_3(x),p_4(x)]\cap\mathcal{P} \\ ③^\star &= [p_2(x),p_3(x)]\cap\mathcal{P} \\ ④^\star &= [p_1(x),p_2(x))\cap\mathcal{P} \\ ⑤^\star &= [0,p_1(x))\cap\mathcal{P} \end{aligned}$$

Am Bild



Hierbei entsprechen die Regionen den Aktionen: ①⋆, ②⋆ AUSSPEICHERN (Fall 1), ③⋆ NICHTS TUN (Fall 3), ④⋆, ⑤⋆ EINSPEICHERN (Fall 2).

Fall 1 Hier ist die relevante zu maximierende Funktion (vgl. die entsprechenden Teile von V_{n+1})

$$U_{n+1}(z,p)-e(p)z \tag{F1}$$

wobei über $z\in[b^{\min},x]$ maximiert wird (anderfalls wäre man in einem anderen Fall, da hier ja nur Ausspeichern relevant ist, daher ist das nächste Level auf das man von x aus kommt in $[b^{\min},x]$). Diese Tatsache führt nun zu folgenden Beobachtungen

1. Ist $p \in$ 1⋆, so maximiert $\bar{b}_{n+1}(p)$ (F1), jedoch ist dieser Zustand von x nicht erreichbar. Es gilt also für die Steigung von (F1) im neuen Zustand $x + i^{\min}(x)$

$$U'_{n+1}(x + i^{\min}(x), p) - e(p) \leq 0 \ \forall \ p \in \text{1⋆}$$

2. Ist $p \in$ 1⋆ $\cup$ 2⋆, so maximiert $\bar{b}_{n+1}(p)$ (F1), sodass gilt

$$U'_{n+1}(x, p) - e(p) \leq 0 \ \forall \ p \in \text{2⋆}$$

3. Ist 1⋆ $\neq \emptyset$ und $p \in$ 2⋆, so maximiert $\bar{b}_{n+1}(p)$ (F1) und ist von x auch erreichbar. Dies bedeutet, dass für $x + i^{\min}(x) \leq \bar{b}_{n+1}(p)$ die Steigung von (F1) größer Null ist, also

$$U'_{n+1}(x + i^{\min}(x), p) - e(p) \geq 0 \ \forall \ p \in \text{2⋆}$$

Weitere Erläuterung: Es gilt $b^{\min} \leq x + i^{\min}(x)$, da ein $p' \in$ 1⋆ existiert, sodass $x + i^{\min}(x) > \bar{b}_{n+1}(p') \geq b^{\min}$.

Fall 2 Hier ist die relevante zu maximierende Funktion (vgl. die entsprechenden Teile von V_{n+1})

$$U_{n+1}(z, p) - k(p)z \tag{F2}$$

wobei über $z \in [x, b^{\max}]$ maximiert wird (anderfalls wäre man in einem anderen Fall, da hier ja nur Einspeichern relevant ist, daher ist das nächste Level auf das man von x aus kommt in $[x, b^{\max}]$). Diese Überlegung führt nun zu folgenden Beobachtungen

1. Ist $p \in$ 5⋆, so maximiert $\underline{b}_{n+1}(p)$ (F2), jedoch ist dieser Zustand von x nicht erreichbar. Es gilt also für die Steigung von (F2) im neuen Zustand $x + i^{\max}(x)$

$$U'_{n+1}(x + i^{\max}(x), p) - k(p) \geq 0 \ \forall \ p \in \text{5⋆}$$

2. Ist $p \in$ 4⋆ $\cup$ 5⋆, so maximiert $\underline{b}_{n+1}(p)$ (F2), sodass gilt

$$U'_{n+1}(x, p) - k(p) \geq 0 \ \forall \ p \in \text{4⋆}$$

3. Ist 5⋆ $\neq \emptyset$ und $p \in$ 4⋆, so maximiert $\underline{b}_{n+1}(p)$ (F2) und ist von x auch erreichbar. Dies bedeutet, dass für $x + i^{\max}(x) \geq \underline{b}_{n+1}(p)$ die Steigung von (F2) kleiner Null ist, also

$$U'_{n+1}(x + i^{\max}(x), p) - k(p) \leq 0 \ \forall \ p \in \text{4⋆}$$

Fall 3 Ist $p \in 3^\star$, dann ist nicht handeln optimal, das heißt jede andere Aktion ist nicht optimal. Also gilt

$$U'_{n+1}(x,p) - e(p) \geq 0 \text{ und } U'_{n+1}(x,p) - k(p) \leq 0 \ \forall\ p \in 3^\star,$$

anders ausgedrückt

$$e(p) \leq U'_{n+1}(x,p) \leq k(p) \ \forall\ p \in 3^\star.$$

Jetzt wird das Verhalten der Funktionen $f^1(x,p)$ und $f^2(x,p)$ in p untersucht. Sei nun x fest aber beliebig. Es gilt

$$\begin{aligned} f^1(x,p) = {} & \mathbf{1}_{1} (U'_{n+1}(x + i^{\min}(x), p) - e(p))(1 + (i^{\min})'(x))) \\ & + \mathbf{1}_{5} (U'_{n+1}(x + i^{\max}(x), p) - k(p))(1 + (i^{\max})'(x)). \end{aligned}$$

Da nach Voraussetzung an die Restriktionenmenge $1+(i^{\min})'(x), 1+(i^{\max})'(x) > 0$ gilt erhält man mit Fall 1, 1.

$$f^1(x,p) \leq 0 \ \forall\ p \in 1^\star,$$

mit Fall 2, 1.

$$f^1(x,p) \geq 0 \ \forall\ p \in 5^\star,$$

und

$$f^1(x,p) = 0 \ \forall\ p \in 2^\star \cup 3^\star \cup 4^\star.$$

Die Funktion $f^1(x,p)$ ist also fallend in p und mit (V2) ergibt sich nun

$$\beta\mathbb{E}[f^1(x, P_{n+1}) | P_n = p] \text{ ist fallend in } p.$$

Betrachtung von f^2: Da $e(p), k(p)$ und $U'_{n+1}(x,p)$ wachsend in p sind und mit Fall 1, 2. ($\leq U'_{n+1}(x,p) \leq e(p)$ für $p \in 2^\star$) und Fall 2, 2. ($U'_{n+1}(x,p) \geq k(p)$ für $p \in 4^\star$) erhält man

$$f^2(x,p) = \mathbf{1}_{3} U'_{n+1}(x,p) + \left(\mathbf{1}_{1} + \mathbf{1}_{2}\right) e(p) + \left(\mathbf{1}_{5} + \mathbf{1}_{4}\right) k(p)$$

ist wachsend in p.

Zusätzlich gilt, da $e(p) \leq k(p)$ und $U'_{n+1}(x,p) \leq k(p)$ nach Fall 3, $f^2(x,p) \leq k(p)$. Mit Lemma 4.3.7 Teil d) folgt nun

$$\begin{aligned} &\beta\mathbb{E}[f^2(x, P_{n+1}) | P_n = p] - k(p) \text{ ist fallend in } p, \\ &\beta\mathbb{E}[f^2(x, P_{n+1}) | P_n = p] - e(p) \text{ ist fallend in } p. \end{aligned}$$

Insgesamt gilt damit

$$\begin{aligned} U'n(x,p) - k(p) &= \beta\mathbb{E}[V'_{n+1}(x, P_{n+1})|P_n = p] - k(p) \\ &= \beta\mathbb{E}[f^1(x, P_{n+1})|P_n = p] + \mathbb{E}[f^2(x, P_{n+1})|P_n = p] - k(p) \end{aligned}$$

ist fallend in p. Analog gilt diese Tatsache auch für $U'n(x,p) - e(p)$. Damit ist B1 gezeigt.

Bleibt noch B2 zu zeigen:

Hierzu betrachtet man wieder V'_{n+1}. Zur Erinnerung:

$$\begin{aligned} V'_{n+1}(x,p) = \quad & \mathbf{1}_{①}\left(-e(p)(i^{\min})'(x) + U'_{n+1}(x + i^{\min}(x), p)(1 + (i^{\min})'(x))\right) \\ + \ & \mathbf{1}_{②} e(p) + \mathbf{1}_{③} U'_{n+1}(x,p) + \mathbf{1}_{④} k(p) \\ + \ & \mathbf{1}_{⑤}\left(-k(p)(i^{\max})'(x) + U'_{n+1}(x + i^{\max}(x), p)(1 + (i^{\max})'(x))\right) \end{aligned}$$

Für $p \in$ ⑤* $= [0, p_1(x)) \cap \mathcal{P}$ gilt

$$\begin{aligned} V'_{n+1}(x,p) &= U'_{n+1}(x + i^{\max}(x), p)(1 + (i^{\max})'(x)) - k(p)(i^{\max})'(x) \\ &= \underbrace{(i^{\max})'(x)}_{\leq 0}\underbrace{(U'_{n+1}(x + i^{\max}(x), p) - k(p))}_{\downarrow \text{ in } p \text{ nach B1}} + \underbrace{U'_{n+1}(x + i^{\max}(x), p)}_{\uparrow \text{ in } p \text{ nach IH}} \end{aligned}$$

ist wachsend in p.

Für $p \in$ ④* $= [p_1(x), p_2(x)) \cap \mathcal{P}$ gilt

$$V'_{n+1}(x,p) = k(p)$$

ist wachsend in p. Mit Fall 2, 3. gilt

$$\begin{aligned} V'_{n+1}(x,p) = k(p) &= k(p)(1 + (i^{\max})'(x)) - k(p)(i^{\max})'(x) \\ &\geq U'_{n+1}(x + i^{\max}(x), p)(1 + (i^{\max})'(x)) - k(p)(i^{\max})'(x); \end{aligned}$$

mit Fall 2, 2. gilt

$$V'_{n+1}(x,p) = k(p) \leq U'_{n+1}(x,p).$$

Für $p \in$ ③* $= [p_2(x), p_3(x)] \cap \mathcal{P}$ gilt

$$V'_{n+1}(x,p) = U'_{n+1}(x,p)$$

ist wachsend in p nach IH.

Für $p \in$ ②* $= (p_3(x), p_4(x)] \cap \mathcal{P}$ gilt

$$V'_{n+1}(x,p) = e(p)$$

ist wachsend in p. Mit Fall 1, 2. gilt

$$V'_{n+1}(x,p) = e(p) \geq U'_{n+1}(x,p).$$

mit Fall 1, 3. gilt

$$\begin{aligned} V'_{n+1}(x,p) = e(p) &= e(p)(1 + (i^{\min})'(x)) - e(p)(i^{\min})'(x) \\ &\leq U'_{n+1}(x + i^{\min}(x), p)(1 + (i^{\min})'(x)) - e(p)(i^{\min})'(x); \end{aligned}$$

Für $p \in$ (1*) $= (p_4(x), \infty) \cap \mathcal{P}$ gilt

$$\begin{aligned} V'_{n+1}(x,p) &= U'_{n+1}(x + i^{\min}(x), p)(1 + (i^{\min})'(x)) - e(p)(i^{\min})'(x) \\ &= \underbrace{(i^{\min})'(x)}_{\leq 0} \underbrace{(U'_{n+1}(x + i^{\min}(x), p) - e(p))}_{\downarrow \text{ in } p \text{ nach } \boxed{\text{B1}}} + \underbrace{U'_{n+1}(x + i^{\min}(x), p)}_{\uparrow \text{ in } p \text{ nach } \boxed{\text{IH}}} \end{aligned}$$

ist wachsend in p.

Insgesamt ist also $V'_{n+1}(x,p)$ wachsend in p. Also gilt auch (mit (V2))

$$U'_n(x,p) = \mathbb{E}[V'_{n+1}(x, P_{n+1} | P_n = p]$$

ist wachsend in p, demnach $\boxed{\text{B2}}$. ∎

Teil II.

Algorithmen

5 Ein mean-reverting Modell zur Modellierung des Gas-Spot-Preises

Wie de Jong u. Walet [24] beschreiben, weist der Gas-Spot-Preis einige Besonderheiten auf. So sind Saisonalitäten im Preis dafür verantwortlich, dass Entscheidungen bezüglich des Gasspeichers im Sommer anders ausfallen als im Winter (unter ansonsten gleichen Umständen). Die sogenannte „mean reversion“ Eigenschaft, die zeitabhängige Volatilität und die unregelmäßigen Sprünge verändern täglich die erwartete Entwicklung und die Veränderungsintensität des Preises.

Mit der Wahl des Preisprozesses als Ornstein-Uhlenbeck Prozess mit zeitabhängigen Koeffizienten, siehe Kapitel 5.1, setzt sich dieser Teil der Arbeit mit den drei ersten der vier genannten Eigenschaften auseinander. Vorteile dieser Wahl werden im zweiten Teil dieses Kapitels aufgezeigt: Der gewählte Prozess lässt sich intuitiv simulieren und durch einen rekombinierenden Binomialbaum approximieren.

5.1. Einführung und Eigenschaften des Preisprozesses

Eines der ersten und bekanntesten Modelle zur Modellierung des Gas-Spot-Preises ist das ein-Faktor-Modell von Schwartz [33], das vor allem die mean-reverting Eigenschaft der Gaspreise abbildet. Der Log-Preisprozess folgt einem Ornstein-Uhlenbeck Prozess und erfülle die folgende stochastische Differentialgleichung

$$dX_t = \alpha(\mu - X_t)dt + \sigma dB_t. \tag{5.1}$$

In diesem Modell bezeichnet $\alpha \in \mathbb{R}^+$ den mean-reversion Faktor, das heißt α beeinflusst die Stärke mit der der Prozess zu einem Niveau (Mean) $\mu \in \mathbb{R}$ „zurückgezogen“ wird. Des weiteren ist $\sigma \in \mathbb{R}^+$ die Volatilität des Prozesses, die die Stärke der zufälligen Schwankungen im Preis beeinflusst. Schließlich ist (B_t) eine Brownsche Bewegung.

Um Saisonalitäten und zeitabhängige Schwankungen ebenfalls zu modellieren, wird in dieser Arbeit das Modell von Schwartz der Log-Preise dahingehend verändert, dass sowohl der Mean als auch die Volatilität Funktionen der Zeit darstellen.

Es wird daher angenommen, dass die Log-Preise die folgende Differentialgleichung erfüllen

$$dX_t = \alpha(\mu(t) - X_t)dt + \sigma(t)dB_t. \tag{5.2}$$

Der Preisprozess $(P_t)_{t\geq 0}$ an sich sei gegeben durch $P_t = e^{X_t}$ und X_t erfülle die Gleichung (5.2).

Bemerkung 5.1.1
Das Modell (5.2) findet man im Übersichtskapitel des Artikels von Stoll u. Wiebauer [36] wieder, hier allerdings direkt für den Gas-Preisprozess nicht für die Log-Preise.

Die bis auf Ununterscheidbarkeit eindeutige Lösung der stochastischen Differentialgleichung (5.2) ist

$$X_t = e^{-\alpha t}\left(x_0 + \int_0^t \alpha e^{\alpha x}\mu(x)dx + \int_0^t e^{\alpha x}\sigma(x)dB_x\right). \tag{5.3}$$

Lemma 48.2 in Bauer [3] beziehungsweise Beispiel 4.7.3 in Shreve [35] zeigen, dass der Prozess $(I_t)_{t\geq 0}$ gegeben durch

$$I_t = \int_0^t e^{\alpha x}\sigma(x)dB_x$$

ein Gaußprozess ist.

Somit ist I_t normalverteilt, woraus sich auch die Verteilung von X_t ergibt.

Satz 5.1.2 (Verteilung der Log-Preise)
Für jedes $t > 0$ ist die Zufallsvariable X_t mit (5.3) normalverteilt mit

$$\mathbb{E}(X_t) = e^{-\alpha t}\left(x_0 + \int_0^t \alpha e^{\alpha x}\mu(x)dx\right),$$
$$\mathbb{V}(X_t) = e^{-2\alpha t}\int_0^t e^{2\alpha x}\sigma^2(x)dx.$$

Beweis
Da I_t normalverteilt ist, ist X_t als affine Transformation von X_t ebenfalls normalverteilt. Bezeichnet man mit $m(t) := x_0 + \int_0^t \alpha e^{\alpha x}\mu(x)dx$ und $s(t) := \int_0^t e^{\alpha x}\sigma(x)dB_x$, so ergibt sich, da $\mathbb{E}(s(t)) = 0$, durch Berechnung

$$\mathbb{E}(X_t) = e^{-\alpha t}m(t)$$

und

$$\begin{aligned}\mathbb{V}(X_t) &= \mathbb{E}(X_t^2) - (\mathbb{E}(X_t))^2 = \mathbb{E}(e^{-2\alpha t}(m(t)^2 + 2m(t)s(t) + s(t)^2 - m(t)^2)) \\ &= e^{-2\alpha t}\mathbb{E}(s(t)^2) = e^{-2\alpha t}\mathbb{E}\left(\int_0^t e^{\alpha x}\sigma(x)dB_x\right)^2 = e^{-2\alpha t}\mathbb{E}\left(\int e^{2\alpha x}\sigma^2(x)d < B_x >\right) \\ &= e^{-2\alpha t}\int e^{2\alpha x}\sigma^2(x)dx. \qquad \blacksquare\end{aligned}$$

Da die Verteilung des Log-Preises X_t zur Zeit t eine Normalverteilung besitzt, folgt der eigentliche Preis zur Zeit t $P_t = e^{X_t}$ einer Lognormalverteilung.

Korollar 5.1.3 (Verteilung der Preise)
Für jedes $t > 0$ gilt

$$P_t \sim \mathcal{LN}(\mathbb{E}(X_t), \mathbb{V}(X_t)).$$

Aus der Eigenschaft, dass $(I_t)_{t\geq 0}$ ein Gaußprozess ist, erhält man folgerichtig, dass der Zufallsvektor $(I_t, I_s)^T$ für $0 < s < t$ eine bivariate Normalverteilung besitzt. Durch affine Transformation erhält man diese Verteilungseigenschaft auch für $(X_t, X_s)^T$.

Satz 5.1.4
Sei $0 < s < t$, dann ist

$$\begin{pmatrix} X_t \\ X_s \end{pmatrix} = \mathcal{N}_2 \left(\begin{pmatrix} \mathbb{E}(X_t) \\ \mathbb{E}(X_s) \end{pmatrix}, \begin{pmatrix} \mathbb{V}(X_t) & \mathrm{Cov}(X_t, X_s) \\ \mathrm{Cov}(X_t, X_s) & V(X_s) \end{pmatrix} \right)$$

Die Kovarianz von X_t und X_s für $0 < s < t$ ergibt sich durch Berechnung zu

$$\begin{aligned}
&\mathrm{Cov}(X_t, X_s) \\
&= e^{-\alpha t - \alpha s} \,\mathrm{Cov}\left(x_0 + \int_0^t \alpha e^{\alpha x} dx + \int_0^t e^{\alpha x}\sigma(x) dB_x, x_0 + \int_0^s \alpha e^{\alpha x} dx + \int_0^s e^{\alpha x}\sigma(x) dB_x \right) \\
&= e^{-\alpha(t+s)} \,\mathrm{Cov}\left(\int_0^t e^{\alpha x}\sigma(x) dB_x, \int_0^s e^{\alpha x}\sigma(x) dB_x \right) \\
&= e^{-\alpha(t+s)} \mathbb{E}\left(\int_0^t e^{\alpha x}\sigma(x) dB_x \cdot \int_0^s e^{\alpha x}\sigma(x) dB_x \right) \\
&= e^{-\alpha(t+s)} \left(\mathbb{E}\left(\int_0^s e^{\alpha x}\sigma(x) dB_x \right)^2 + \mathbb{E}\left(\int_s^t e^{\alpha x}\sigma(x) dB_x \cdot \int_0^s e^{\alpha x}\sigma(x) dB_x \right) \right) \\
&= e^{-\alpha(t+s)} \mathbb{E}\left(\int_0^s e^{2\alpha x}\sigma^2(x) d < B_x > \right) \\
&= e^{-\alpha(t+s)} \int_0^s e^{2\alpha x}\sigma^2(x) dx.
\end{aligned} \tag{5.4}$$

Bemerkung 5.1.5
Mit

$$\mathrm{Cov}(X_t - X_s, X_s) = \mathrm{Cov}(X_t, X_s) - \mathbb{V}(X_s) = \int_0^s e^{2\alpha x}\sigma^2(x) dx (e^{-\alpha(t+s)} - e^{-2\alpha s})$$

erkennt man, dass die Zuwächse der Log-Preise korreliert und damit nicht unabhängig sind.

Die folgende Verteilungseigenschaft wird zum Nachweis der Existenz einer Schrankenfunktion und somit zum Nachweis der Integrabilitätsannahme benötigt (vgl. Beispiel 4.1.4).

Satz 5.1.6
Für $0 < s < t$ gilt

$$X_t|X_s = x \sim \mathcal{N}\left(\mathbb{E}(X_t) + \rho\sqrt{\frac{\mathbb{V}(X_t)}{\mathbb{V}(X_s)}}(x - \mathbb{E}(X_s)), \mathbb{V}(X_s)(1-\rho^2)\right),$$

wobei

$$\rho = \frac{\mathrm{Cov}(X_s, X_t)}{\sqrt{\mathbb{V}(X_t)\mathbb{V}(X_s)}}$$

Beweis
Folgt direkt aus Theorem 2.1.1 Tong [38]. ∎

Mit der Verteilung von $X_t|X_s = x$ ist auch die Verteilung von $P_t|P_s = p$ bekannt.

Korollar 5.1.7
Für $0 < s < t$ hat P_t gegeben $P_s = p$ eine Lognormalverteilung.

Da der Erwartungswert $\mathbb{E}[P_t|P_s = p]$ für $0 < s < t$ im Nachweis für die Erfüllung der Integrabilitätsannahme von entscheidender Bedeutung ist, wird dieser in der folgenden Bemerkung genauer untersucht.

Bemerkung 5.1.8
Da die Zufallsvariable P_t gegeben $P_s = p$ eine Lognormalverteilung mit Parametern

$$\begin{aligned}\mu(s,t,p) &:= \mathbb{E}(X_t) + \rho\sqrt{\frac{\mathbb{V}(X_t)}{\mathbb{V}(X_s)}}(\log p - \mathbb{E}(X_s))\\ \sigma^2(s,t) &:= \mathbb{V}(X_s)(1-\rho^2)\end{aligned}$$

besitzt, ist der Erwartungswert gegeben durch

$$\mathbb{E}[P_t|P_s = p] = e^{\mu(s,t,p)+\sigma^2(s,t)/2}.$$

Setzt man die entsprechende Erwartungswerte, Varianzen (Satz 5.1.2) und Kovarianz (5.4) ein, so erhält man

$$\begin{aligned}\mu(s,t,p) =& e^{-\alpha t}\left(x_0 + \int_0^t \alpha e^{\alpha x}\mu(x)dx\right)\\ &+ \frac{e^{-\alpha(t+s)}\int_0^s e^{2\alpha x}\sigma^2(x)dx}{e^{-2\alpha s}\int_0^s e^{2\alpha x}\sigma^2(x)dx}\left(\log p - e^{-\alpha s}(x_0 + \int_0^s \alpha e^{-\alpha x}\mu(x)dx\right).\end{aligned}$$

Da zur Prüfung der Integrabilitätsannahme nur Terme mit p relevant sind, fasst man die restlichen Terme in einer Konstanten $\hat{c}(s,t)$ zusammen, die abhängig von t und s ist. Es gilt also

$$\mu(s,t,p) = \hat{c}(s,t) + e^{-\alpha(t-s)} \log p.$$

Damit ist

$$\mathbb{E}[P_t | P_s = p] = e^{\hat{c}(s,t)+e^{-\alpha(t-s)}\log p+\sigma^2(s,t)/2} = p^{e^{-\alpha(t-s)}} \cdot e^{\hat{c}(s,t)+\sigma^2(s,t)/2}$$

und im Spezialfall $t = n+1$ und $s = n$ gilt für $n = 0, \ldots, N-1$

$$\mathbb{E}[P_{n+1} | P_n = p] = c(n) p^{e^{-\alpha}}.$$

Eine weitere Verteilungseigenschaft, nämlich die Verteilung der bedingten Zuwächse des Prozesses, ist insbesondere für die Simulation des Prozesses wichtig.

Satz 5.1.9
Sei $0 < s < t$, dann hat $X_t - X_s$ gegeben $X_s = x$ eine Normalverteilung mit Erwartungswert

$$\mathbb{E}(X_t) - \mathbb{E}(X_s) + \rho\sqrt{\frac{\mathbb{V}(X_t - X_s)}{\mathbb{V}(X_s)}}(x - \mathbb{E}(X_s))$$

und Varianz

$$\mathbb{V}(X_t - X_s)(1 - \rho^2),$$

wobei

$$\rho = \frac{\mathrm{Cov}(X_t, X_s) - \mathbb{V}(X_s)}{\sqrt{\mathbb{V}(X_t - X_s)\mathbb{V}(X_s)}}$$

Beweis
Folgt durch Rechnung aus Theorem 2.1.1 Tong [38]. ∎

Bemerkung 5.1.10
Da der Preisprozess lognormalverteilt ist, sind die Preise zu jedem Zeitpunkt $t \geq 0$ fast sicher positiv. Allerdings, ist nicht auszuschließen, dass die Wahrscheinlichkeit $\mathbb{P}(e(P_t) < 0)$ positiv ist. Es könnte in Simulationen vorkommen, dass zu einem Zeitpunkt die Forderung $e(p) \geq 0$ nicht mehr erfüllt ist. Je nach Parameterwahl ist die Wahrscheinlichkeit, dass eine solche Situation eintritt, jedoch nahezu Null.

5.2. Simulation und Approximation des Preisprozesses

Simulation

Zunächst wird die Simulation des Log-Preisprozesses betrachtet. Nach Satz 5.1.9 ist, wenn der Log-Preis zu Zeit s bekannt ist, die Verteilung des Zuwachses bis zur Zeit t

eine Normalverteilung. Daher lässt sich ein Pfad des Log-Preisprozesses zu N diskreten Zeitpunkten bei gegebenem Anfangspreis dadurch simulieren, indem nacheinander normalverteilte Zufallsvariablen generiert - mit jeweilig verschiedenen Erwartungswerten und Varianzen - und diese „aneinandergehängt“ werden.

Das Zeitintervall $[0, N]$ wird durch die Zeitpunkte $0, 1, 2, \ldots, N$ diskretisiert, da für die Gasspeicheroptimierung als Zeiteinheit Tage angenommen werden und das Modell dann dementsprechend angepasst wird.

Die Simulation der Log-Preise zu den Zeitpunkten $0, 1, \ldots, N$ - also ein Pfad - bei gegebenem Anfangspreis p_0 beschreibt der folgende Algorithmus 1. Die Funktion `random.normal` beschreibt die Generierung einer normalverteilten Zufallszahl mit gegebenem Erwartungswert und Varianz.

Algorithmus 1 : Simulation eines Pfades der Log-Preise

Data : Koeffizienten des Preisprozesses und gewünschte Länge N.
Result : Ein Vektor pfad der Länge $N + 1$.
$\mathsf{pfad}[0] = \log p_0$;
$\mathsf{pfad}[1] = \texttt{random.normal}(\mathbb{E}(X_1), \mathbb{V}(X_1))$;
for $i \in \{2, \ldots, N\}$ **do**
  $\mu_1 = \mathbb{E}(X_i) - \mathbb{E}(X_{i-1})$;
  $\mu_2 = \mathbb{E}(X_{i-1})$;
  $\sigma_1 = \sqrt{\mathbb{V}(X_i) + \mathbb{V}(X_{i-1}) - 2\,\mathrm{Cov}(X_i, X_{i-1})}$;
  $\sigma_2 = \sqrt{\mathbb{V}(X_{i-1})}$;
  $\rho = \frac{\mathrm{Cov}(X_i, X_{i-1}) - \mathbb{V}(X_{i-1})}{\sqrt{\sigma_1 \sigma_2}}$;
  $\mathsf{mean} = (\mu_1 + \rho \frac{\sigma_1}{\sigma_2}(\mathsf{pfad}[i-1] - \mu_2)$;
  $\mathsf{variance} = {\sigma_1}^2(1 - \rho^2)$;
  $\mathsf{zufallszahl} = \texttt{random.normal}(\mathsf{mean}, \mathsf{variance})$;
  $\mathsf{pfad}[i] = \mathsf{pfad}[i-1] + \mathsf{zufallszahl}$;
end

Betrachtet man den eigentlichen Preisprozess, so ist die Verteilung der „absoluten“ Zuwächse nicht bekannt, allerdings die Verteilung der „relativen“Zuwächse. Aus der Verteilungseigenschaft von Satz 5.1.9 - $X_t - X_s | X_s = x \sim \mathcal{N}(\cdot, \cdot)$ - , erhält man $e^{X_t - X_s} | X_s = x \sim \mathcal{LN}(\cdot, \cdot)$ und damit

$$\left. \frac{P_t}{P_s} \right| P_s = p \sim \mathcal{LN}(\mu, \sigma^2),$$

wobei μ der Erwartungswert und σ^2 die Varianz aus Satz 5.1.9 darstellen, mit $p = e^x$.

Ein Algorithmus zur Simulation eines Pfades würde demnach ähnlich zu Algorithmus 1 verlaufen, mit den Unterschieden, dass zum einen die Zufallsvariablen lognormalverteilt generiert werden und zum anderen die erhaltene Zufallszahl nicht addiert, sondern multipliziert wird. Bei der Generierung der Zufallsvariablen entspricht `mean` und `variance`

immer noch den Parametern der Verteilung mit dem Unterschied, dass der Logarithmus von `pfad`$[i-1]$ verwendet wird, jedoch nicht mehr dem Erwartungswert und der Varianz.

Alternativ generiert man einen Pfad der Log-Preise gemäß Algorithmus 1 und wendet den erhaltenen Vektor komponentenweise auf die Exponentialfunktion an. Das ist die Vorgehensweise, die in dieser Arbeit zur Simulation der Preispfade verwendet wird.

Sobald man an den Punkt gelangt Pfade des Preisprozesses zu simulieren - oder auch Algorithmen zur Bewertung eines Gasspeichers zu betrachten -, muss man sich auch mit der konkreten Parameterwahl beschäftigen. Benth u. a. [4] befassen sich in „Stochastic Modelling of Electricity and Related Markets“ in Kapitel 3 unter anderem damit, ein Modell für einen zeitabhängigen Mean an die Log-Gas-Spot-Preise aus dem UK (National Balancing Point, NBP) im Zeitraum 2. Februar 2001 bis 24. Oktober 2006 anzupassen. Die Einheit der Preise ist in pence pro therm gegeben. Ein pence per therm entspricht 0,1 pence per MMBtu (million British thermal unit).

Es wird davon ausgegangen, dass ein Betrachtungszeitraum von einem Jahr 250 sogenannte Trading Tage enthält. Die saisonale Komponente im Mean wird daher mit einer Periodizität von 250 modelliert. Insgesamt werden der Trend und die saisonale Komponente modelliert durch

$$\mu(t) = a_0 + a_2 t + a_2 \cos(2\pi(t - a_3)/250). \tag{5.5}$$

Mittels einer kleinsten Quadrate Methode passen Benth u. a. [4] die Parameter an die oben genannte Daten an und schätzen wie folgt (vgl. Tabelle 5.1. in [4]); $t = 0$ entspricht hier dem ersten Trading Tag im Februar,

a_0	a_1	a_2	a_3
2.69	0.0007	-0.234	118.1

.

Benth u. a. [4] schätzen ebenfalls den mean reversion Faktor α aus den oben genannten Daten und erhalten

$$\widehat{\alpha} = 0.073. \tag{5.6}$$

De Jong u. Walet [24] nehmen hingegen eine mean reversion Rate von 0.0678 an, die an Daten des Zeebrugge Hub angelehnt ist.

Die Modellierung einer zeitabhängigen Volatilität gestaltet sich problematisch, da in der Literatur, zum Beispiel [4, 5, 24], die Volatilität häufig als konstant oder stochastisch angenommen wird. Daher wird in der vorliegenden Arbeit, für Simulationen und die Durchführung der Algorithmen aus Kapitel 6, der Spezialfall konstanter Volatilität betrachtet, die Benth u. a. [4] auf

$$\widehat{\sigma} = 0.072 \tag{5.7}$$

schätzen. Zum Vergleich: de Jong u. Walet [24] verwenden eine konstante Volatilität von 0.084.

Diese Schätzer bilden den Ausgangspunkt folgender Simulationen und bilden auch die Grundlage der Durchführung der Algorithmen in Kapitel 7. Dennoch wird in Kapitel 7 der Einfluss insbesondere der Parameter α und σ diskutiert, die ausgehend von oben genannten Schätzern variiert werden.

Die Schaubilder 5.1 und 5.2 zeigen zum einen 5 und 150 Pfade des Modells mit oben genannten Parametern der Log-Preise bzw. der Preise im zeitlichen Verlauf. Das obere Bild zeigt jeweils einzelne Pfade, um mögliche Verläufe des Preises zu veranschaulichen, das untere Bild soll die Struktur, insbesondere das „Folgen“ des zeitabhängigen Mean verdeutlichen, der zur visuellen Unterstützung grün eingezeichnet ist. Als Preis zu Beginn wurde $p_0 = e^{\mu(0)}$ gewählt.

Um einen kurzen Einblick darüber zu geben, wie sich die Struktur des Prozesses mit zeitabhängiger Volatilität verändert, zeigt Abbildung 5.3 150 Simulationen der Log-Preise (oben) und der Preise (unten) mit einer Volatiliät von $\sigma(t) = 0.072 + 0.02\sin(2\pi t/250)$.

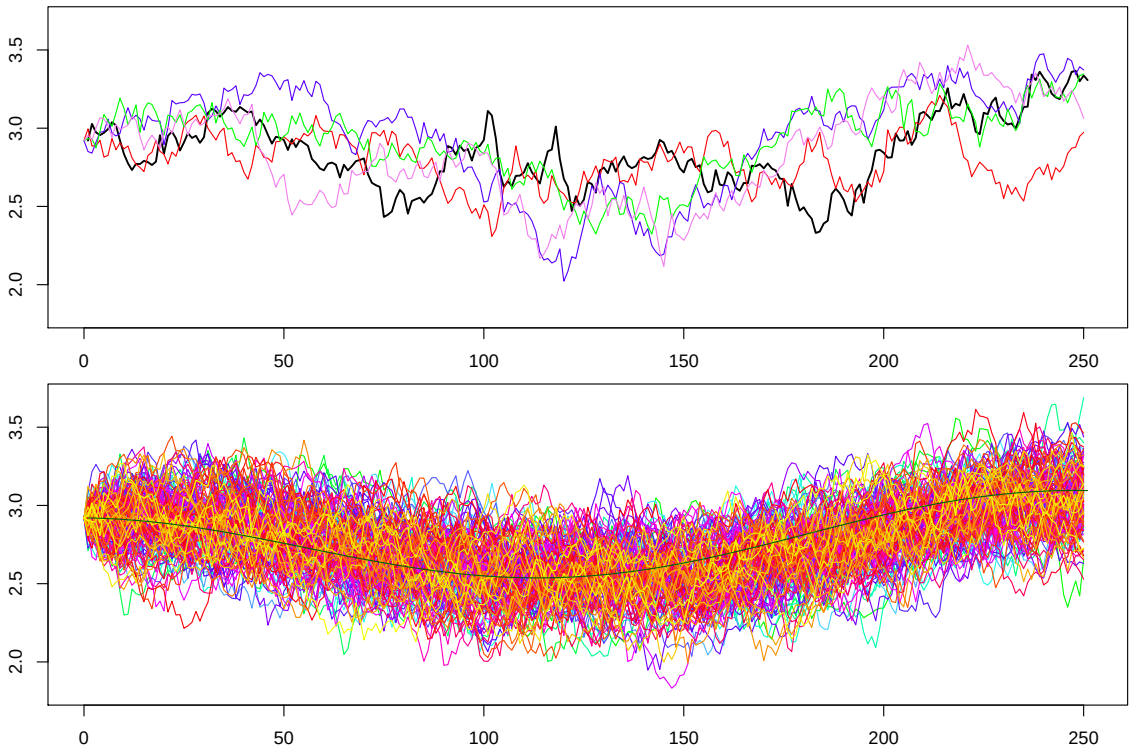


Abbildung 5.1.: 5 bzw. 150 simulierte Pfade der Log-Preise im zeitlichen Verlauf mit $N = 250$

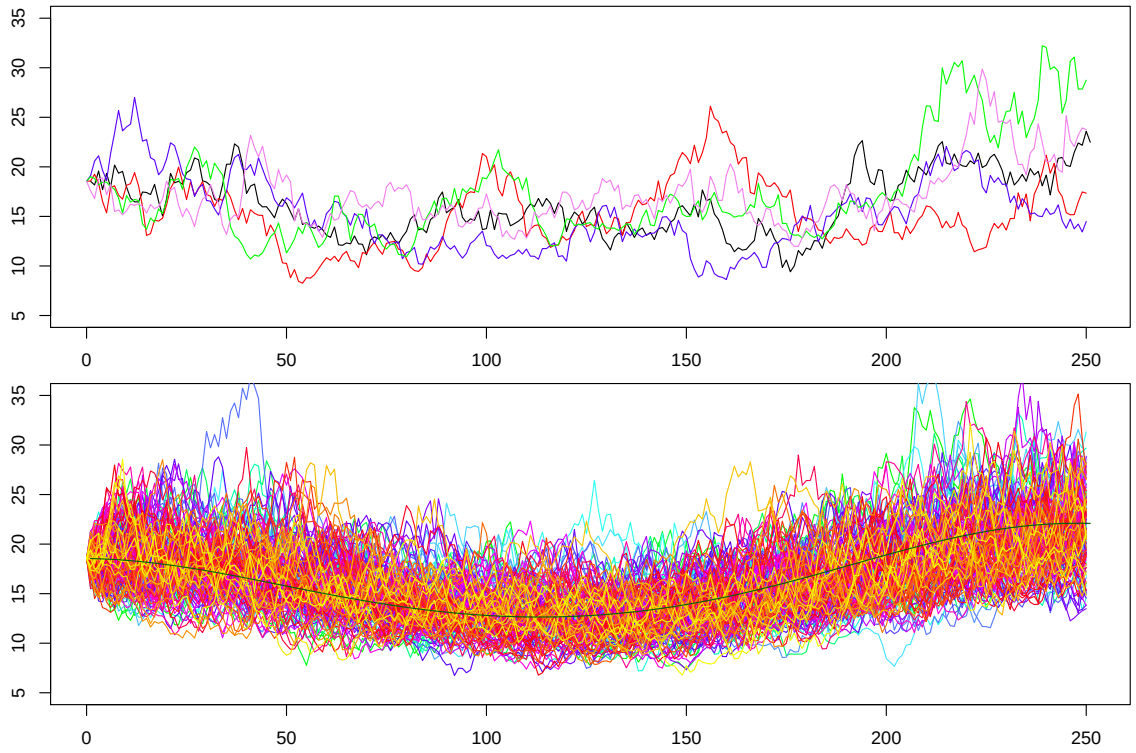


Abbildung 5.2.: 5 bzw. 150 simulierte Pfade der Preise (in pence/therm) im zeitlichen Verlauf mit $N = 250$

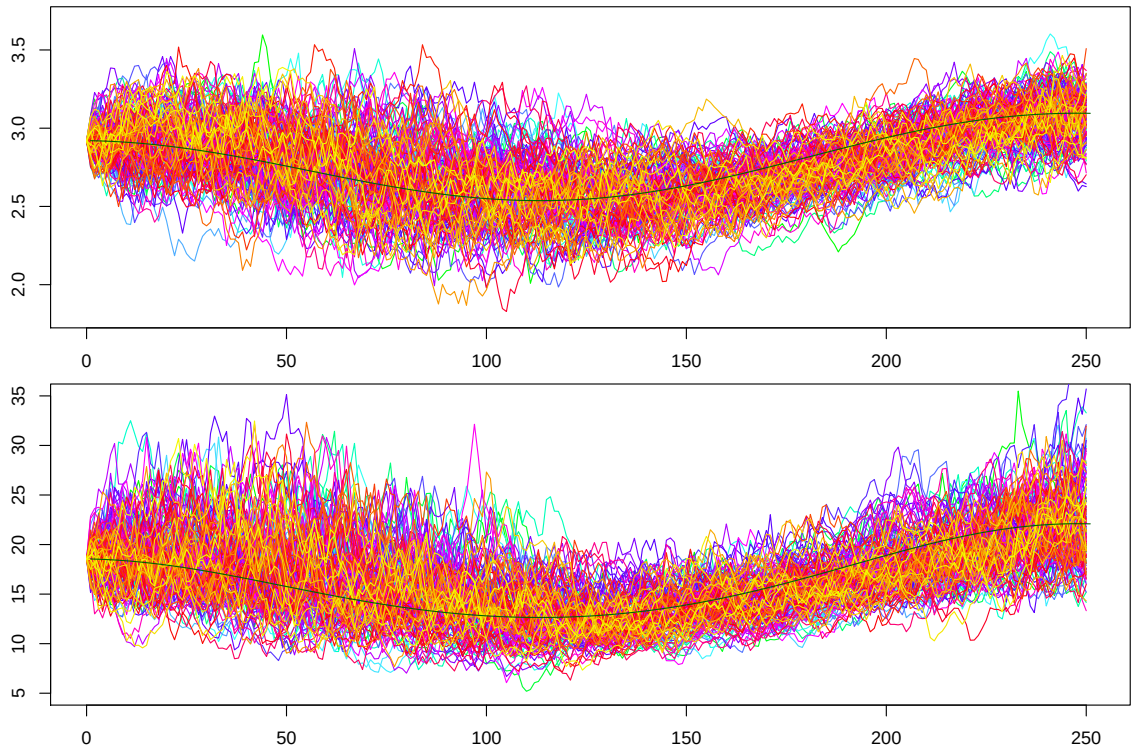


Abbildung 5.3.: 150 simulierte Pfade der Log-Preise (oben) bzw. Preise (unten) im zeitlichen Verlauf mit $N = 250$ und zeitabhängiger Volatilität

Approximation

> „A binomial approximation to a diffusion is defined as "computationally simple" if the number of nodes grows at most linearly in the number of time intervals."[28]

Nelson u. Ramaswamy [28] beschreiben eine Methode, mit der Diffusionsprozesse durch einen rekombinierenden Binomialbaum der Art approximiert werden, dass die Folge der Binomialprozesse unter gewissen Voraussetzungen schwach gegen die Diffusion konvergiert.

Im Folgenden wird zunächst am Beispiel des Log-Preisprozesses (5.3), der einen Diffusionsprozess darstellt, die Konstruktion des Binomialbaums erläutert, später werden dann in Bemerkung 5.2.2 einige der Voraussetzungen unter denen schwache Konvergenz gilt, genannt.

Ausgangspunkt ist ein Diffusionsprozess, der folgender stochastischen Differentialgleichung folgt

$$dX_t = \widetilde{\mu}(t, X_t)dt + \widetilde{\sigma}(t, X_t)dB_t,$$

wobei $\widetilde{\mu}(t,x),\ \widetilde{\sigma}(t,x) \geq 0$ den momentanen Drift und die momentane Standardabweichung darstellen; X_0 ist konstant. In der Situation der Log-Preise ist also

$$\begin{aligned}\widetilde{\mu}(t,x) &= \alpha(\mu(t) - x),\\ \widetilde{\sigma}(t,x) &= \sigma(t).\end{aligned}$$

Gegeben sei nun das Zeitintervall $[0,T]$, das in n äquidistante Teilintervalle der Länge $\Delta t = T/n$ unterteilt wird. Es wird dann der Prozess $(Y_{\Delta t}(t))_{t\geq 0}$ betrachtet, der einem Binomialschema folgt, das heißt zwischen den Intervallknoten $0, \Delta t, 2\Delta t, 3\Delta t, \ldots$ der Unterteilung konstant ist und an diesen mit einer Wahrscheinlichkeit $q^{\uparrow}$ nach oben bzw. $q^{\downarrow} = 1 - q^{\uparrow}$ nach unten springt. Die Wahrscheinlichkeiten der „Up"- bzw. „Down"-Bewegung, sowie die Sprunghöhen selbst, hängen - neben Δt - von der Zeit und dem aktuellen Wert des Prozesses ab. Es seien also

$$\begin{aligned}&q^{\uparrow}_{\Delta t}(y, \Delta tk) \text{ mit } 0 \leq q^{\uparrow}_{\Delta t}(y, \Delta tk) \leq 1,\\ &Y^{\downarrow}_{\Delta t}(y, \Delta tk), Y^{\uparrow}_{\Delta t}(y, \Delta tk) \text{ mit } -\infty < Y^{\downarrow}_{\Delta t}(y, \Delta tk) \leq Y^{\uparrow}_{\Delta t}(y, \Delta tk) < \infty,\end{aligned}$$

Funktionen auf $\mathbb{R} \times [0, \infty)$. Die Bedingungen an die Funktionen gelten für alle $y \in \mathbb{R}$ und $k = 0, 1, \ldots, n$. Der stochastische Prozess $(Y_{\Delta t}(t))_{t\geq 0}$ ist gegeben durch

$$\begin{aligned}&Y_{\Delta t}(0) = X_0,\\ &Y_{\Delta t}(t) = Y_{\Delta t}(k\Delta t), \quad k\Delta t \leq t < (k+1)\Delta t,\\ &\mathbb{P}[Y_{\Delta t}((k+1)\Delta t) = Y^{\uparrow}_{\Delta t}(Y_{\Delta t}(k\Delta t), k\Delta t)|k\Delta t, Y_{\Delta t}(k\Delta t)] = q^{\uparrow}_{\Delta t}(Y_{\Delta t}(k\Delta t), \Delta tk),\\ &\mathbb{P}[Y_{\Delta t}((k+1)\Delta t) = Y^{\downarrow}_{\Delta t}(Y_{\Delta t}(k\Delta t), k\Delta t)|k\Delta t, Y_{\Delta t}(k\Delta t)] = q^{\downarrow}_{\Delta t}(Y_{\Delta t}(k\Delta t), \Delta tk).\end{aligned}$$

Abbildung 5.4 zeigt die Struktur, die der Binomialprozess beschreibt, sowie die spezielle Struktur, die durch die Konstruktion entstehen soll. Dieser Prozess wird so konstruiert,

dass man bei einer „Up“- und dann „Down“-Bewegung den gleichen Wert des Prozesses erreicht, wie bei einer „Down“- und dann „Up“-Bewegung.

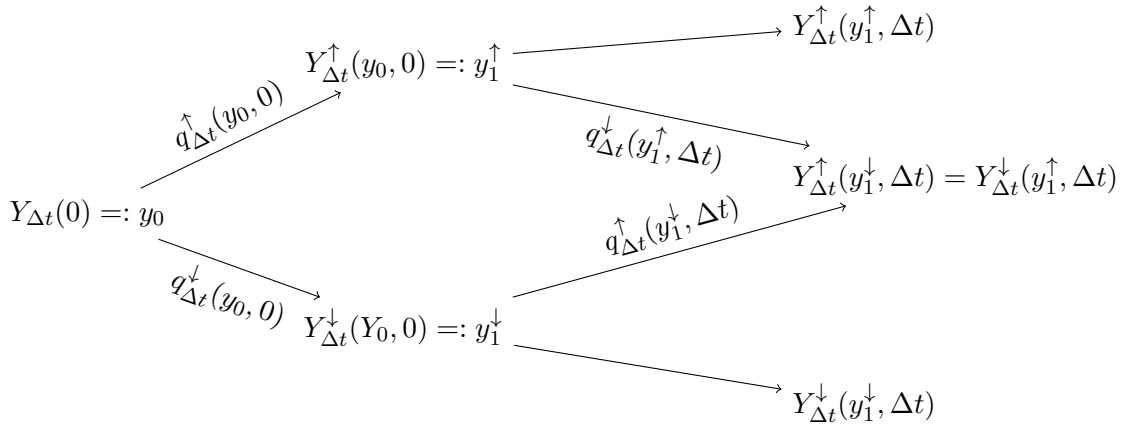


Abbildung 5.4.: Generelle Struktur des Binomialbaums

Damit der Prozess $(Y_{\Delta t}(t))_{t\geq 0}$ in Verteilung gegen (X_t) konvergiert, muss der Binomialprozess außerdem so konstruiert werden, dass der lokale Drift $\mu_{\Delta t}(y,t)$ definiert durch

$$\mu_{\Delta t}(t,y) = \frac{q^{\uparrow}_{\Delta t}(y,t^\star)(Y^{\uparrow}_{\Delta t}(y,t^\star) - y) + q^{\downarrow}_{\Delta t}(y,t^\star)(Y^{\downarrow}_{\Delta t}(y,t^\star) - y)}{\Delta t}$$

und das lokale zweite Moment $\sigma^2_{\Delta t}(y,t)$ definiert durch

$$\sigma_{\Delta t}(t,y) = \frac{q^{\uparrow}_{\Delta t}(y,t^\star)(Y^{\uparrow}_{\Delta t}(y,t^\star) - y)^2 + q^{\downarrow}_{\Delta t}(y,t^\star)(Y^{\downarrow}_{\Delta t}(y,t^\star) - y)^2}{\Delta t},$$

wobei $t^\star = \Delta t \lfloor t/\Delta t \rfloor$, gegen $\widetilde{\mu}$ und $\widetilde{\sigma}^2$ in folgendem Sinne konvergieren.

Für jedes $T > 0$ und $\delta > 0$ gelte

$$\lim_{\Delta t \searrow 0} \sup_{\substack{|y|\leq\delta,\\ 0\leq t\leq T}} |\mu_{\Delta t}(t,y) - \widetilde{\mu}(t,y)| = 0,$$
$$\lim_{\Delta t \searrow 0} \sup_{\substack{|y|\leq\delta,\\ 0\leq t\leq T}} \left|\sigma^2_{\Delta t}(t,y) - \widetilde{\sigma}^2(t,y)\right| = 0.$$

Die Forderung der Rekombiniertheit des Baumes, macht es nicht möglich, die intuitive Wahl $Y^{\uparrow/\downarrow}_{\Delta t}(y, k\Delta t) = y \pm \sqrt{\Delta t}\sigma(k\Delta t)$ zu verwenden. Um zunächst die Zeitabhängigkeit der Volatilität zu umgehen, wird eine Transformation $H(x,t)$ des ursprünglichen Prozesses betrachtet, die zweimal differenzierbar in x und einmal in t sein soll. Man erhält mit der Itô-Formel

$$dH(X_t,t) = \left(\alpha(\mu(t) - X_t)\frac{\partial H(X_t,t)}{\partial x} + \frac{1}{2}\sigma^2(t)\frac{\partial^2 H(X_t,t)}{\partial x^2} + \frac{\partial H(X_t,t)}{\partial t}\right)dt + \left(\sigma(t)\frac{\partial H(X_t,t)}{\partial x}\right)dB_t.$$

Damit die Volatilität des transformierten Prozesses $(H(X_t,t))_t \geq 0$ konstante Volatilität hat, wählt man H so, dass

$$\left(\sigma(t)\frac{\partial H(X_t,t)}{\partial x}\right) dB_t = dB_t$$

gilt; und zwar

$$H(x,t) = \frac{x}{\sigma(t)}.$$

Die Dynamik von $(H(X_t,t))_{t\geq 0}$ ist dann gegeben durch

$$\begin{aligned} dH(X_t,t) &= \left(\frac{\alpha(\mu(t)-X_t)}{\sigma(t)} - \frac{X_t}{\sigma^2(t)}\frac{\partial\sigma(t)}{\partial t}\right) dt + dB_t \\ &= \left(\frac{\alpha\mu(t)}{\sigma(t)} - \alpha H(X_t,t) - \frac{X_t}{\sigma^2(t)}\frac{\partial\sigma(t)}{\partial t}\right) dt + dB_t \end{aligned}$$

So lassen sich die Knoten eines rekombinierenden Binomialbaums für $(H(X_t,t))$ wie in Abbildung 5.5 gezeigt konstruieren.

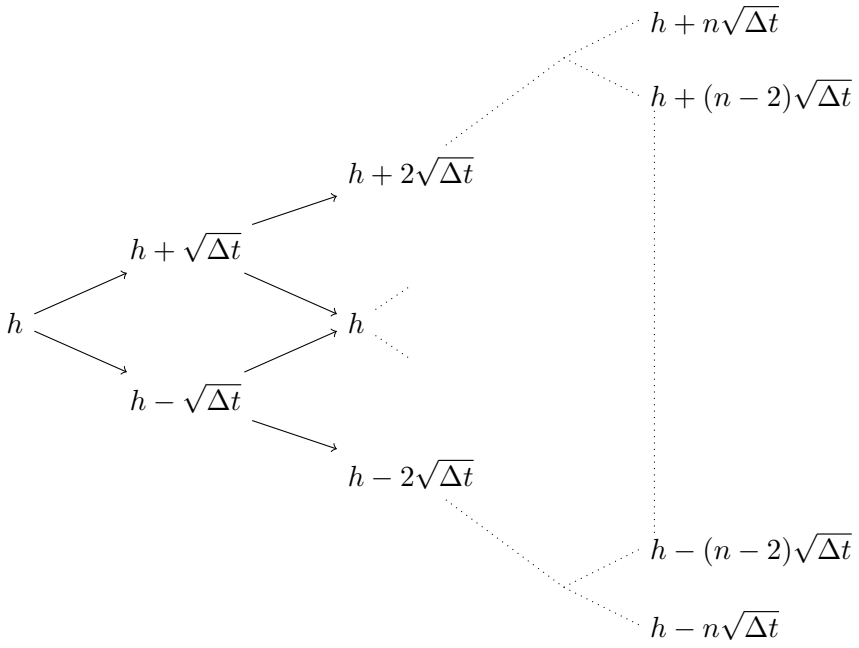


Abbildung 5.5.: Struktur des Binomialbaums für die Transformation

Ausgehend von H wird nun rücktransformiert durch

$$X(h,t) = \{x : H(x,t) = h\},$$

was zur Konstruktion des Binomialbaums des eigentlichen Prozesses verwendet wird. Es ist dann

$$\begin{aligned} Y^{\uparrow}_{\Delta t}(h,t) &= X(h+\sqrt{\Delta t}, t+\Delta t) = (h+\sqrt{\Delta t})\sigma(t+\Delta t), \\ Y^{\uparrow}_{\Delta t}(h,t) &= X(h-\sqrt{\Delta t}, t+\Delta t) = (h-\sqrt{\Delta t})\sigma(t+\Delta t). \end{aligned}$$

Mit

$$\begin{aligned}\widetilde{q^{\uparrow}_{\Delta t}}(h,t) &= \frac{\Delta t\alpha(\mu(t) - X(h,t)) + X(h,t) - Y^{\downarrow}_{\Delta t}(h,t)}{Y^{\uparrow}_{\Delta t}(h,t) - Y^{\downarrow}_{\Delta t}(h,t)} \\ &= \frac{\Delta t\alpha(\mu(t) - \sigma(t)h) + \sigma(t)h - (h - \sqrt{\Delta t})\sigma(t + \sqrt{\Delta t})}{2\sqrt{\Delta t}\sigma(t + \Delta t)}\end{aligned}$$

erhält man

$$q^{\uparrow}_{\Delta t}(h,t) = \max\left(0, \min\left(1, \widetilde{q^{\uparrow}_{\Delta t}}(h,t)\right)\right), \quad q^{\downarrow}_{\Delta t}(h,t) = 1 - q^{\uparrow}_{\Delta t}(h,t).$$

Satz 5.2.1
Unter gewissen Voraussetzungen (vergleiche Nelson u. Ramaswamy [28]), konvergiert der wie oben konstruierte Binomialprozess schwach gegen (X_t) für Δt gegen 0.

Beweis
Vergleiche Theorem 2 in Nelson u. Ramaswamy [28]. ∎

Bemerkung 5.2.2
Einige wichtige Voraussetzungen für Satz 5.2.1 sind Voraussetzungen an den Drift und die Volatilität. Übertragen auf das betrachtete Beispiel bedeuten diese unter anderem

- μ sei stetig
- σ sei stetig, positiv und zweimal differenzierbar.

Für alle weiteren Voraussetzungen wird auf den oben genannten Artikel verwiesen.

Es wird im Folgenden angenommen, dass insbesondere die Voraussetzungen aus Bemerkung 5.2.2 erfüllt sind.

Die Generierung des Baumes der Log-Preise erfolgt gemäß Algorithmus 2. Die Knoten der Bewegungen des Prozesses, sowie die „Up"- und „Down"-Wahrscheinlichkeiten werden in Matrizen gespeichert. Da das Intervall $[0, T]$ in n äquidistante Teilintervalle der Länge Δt zerlegt wird, sind die Knoten-Matrix `X` von Dimension $(n+1) \times (n+1)$ und die der jeweiligen Wahrscheinlichkeit `Up` und `Down` von Dimension $n \times n$. Der Eintrag `X`$[j,k]$ der Matrix `X` enthält den Log-Preis bei j „Down"-Bewegungen und $k-j$ „Up"-Bewegungen des Log-Preises. Der Eintrag `Up`$[j,k]$ der Matrix `Up` enthält die Wahrscheinlichkeit bei einem Log-Preis von `X`$[j,k]$ im nächsten Schritt „nach oben zu springen", der Eintrag `Down`$[j,k]$ enthält dementsprechend die Wahrscheinlichkeit einer „Down"-Bewegung. Damit befinden sich keine Einträge unterhalb der Diagonalen der Matrizen (obere Dreiecksmatrix).

Algorithmus 2 : Generierung des Binomialprozesses der Log-Preise

Data : Koeffizienten des Preisprozesses, Endzeit T, gewünschte Anzahl an Intervallen n.
Result : Drei Matrizen X, Up und Down.
$\Delta t = \frac{T}{n}$;
$\mathsf{X}[0,0] = \log p_0$;
$\mathsf{H}[0,0] = \frac{\mathsf{X}[0,0]}{\sigma(0)}$;
for $j \in \{1, \ldots, n\}$ **do**
  for $k \in \{j, \ldots, n\}$ **do**
    $\mathsf{H}[j,k] = \mathsf{H}[0,0] + (k-2j)\sqrt{\Delta t}$;
    $\mathsf{X}[j,k] = \mathsf{H}[j,k] + \sigma(k\sqrt{\Delta t})\sqrt{\Delta t}$;
  end
end
for $k \in \{1, \ldots, n-1\}$ **do**
  for $j \in \{1, \ldots, k\}$ **do**
    $\mathsf{Up}[j,k] = \frac{\Delta t \alpha(\mu(k\sqrt{\Delta t}) - \mathsf{X}[j,k]) + \mathsf{X}[j,k] - \mathsf{X}[j+1,k+1])}{\mathsf{X}[j,k+1] - \mathsf{X}[j+1,k+1]}$;
    if $\mathsf{Up}[j,k] < 0$ **then**
      $\mathsf{Up}[j,k] = 0$;
    end
    if $\mathsf{Up}[j,k] > 1$ **then**
      $\mathsf{Up}[j,k] = 1$;
    end
    $\mathsf{Down}[j,k] = 1 - \mathsf{Up}[j,k]$;
  end
end

Hat man so den approximierenden Binomialbaum für den Log-Preisprozess generiert, verfährt man ähnlich wie im vorherigen Teilabschnitt Simulation, um einen Binomialbaum für den eigentlichen Preisprozess zu erhalten. Man wendet auf jeden Eintrag der erhaltenen Matrix `X` die Exponentialfunktion an.

In den beispielhaften Schaubilder soll unter anderem veranschaulicht werden, wie sich die Dimension insbesondere der Matrix `X` und dadurch auch `P = exp(X)` verkleinern lässt, indem man Pfade, die Wahrscheinlichkeit 0 haben, nicht berücksichtigt. So wird auch die Anzahl der Rechenknoten reduziert, die bei der Durchführung von Algorithmen berücksichtigt werden müssen (vergleiche Kapitel 6.2).

Für die folgenden Schaubilder werden die Parameter wie im Simulationsteil dieses Kapitels, vergleiche Abbildungen 5.1 und 5.2, verwendet. Das Intervall $[0, N]$, mit $N = 250$, wird unterteilt in N Teilintervalle der Länge 1, jedes Teilintervall entspricht also einem Trading Tag. Die Abbildungen stellen somit die Binomialbäume dar, auf denen unter anderem in Kapitel 7 der Algorithmus aus 6.2 durchgeführt wird.

Schaubild 5.6 zeigt das komplette Gitter, das bei der Berechnung der Matrizen `X` (oben) und `P` (unten) entsteht. Zum Vergleich zeigt 5.7 nur die jeweiligen Knoten, die mit positiver Wahrscheinlichkeit erreicht werden. Außerdem sind jeweils 150 simulierte Pfade (bunt) dargestellt, sowie der zeitabhängige Mean, zur Veranschaulichung der Struktur des approximierten Prozesses.

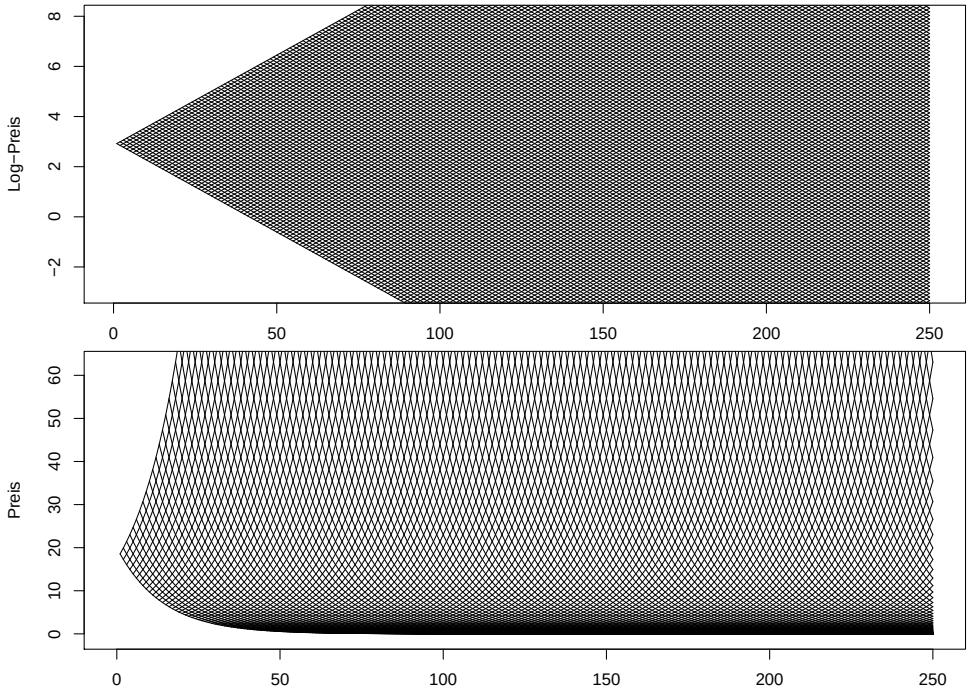


Abbildung 5.6.: Knotennetz des Binomialprozesses der Log-Preise und Preise im zeitlichen Verlauf

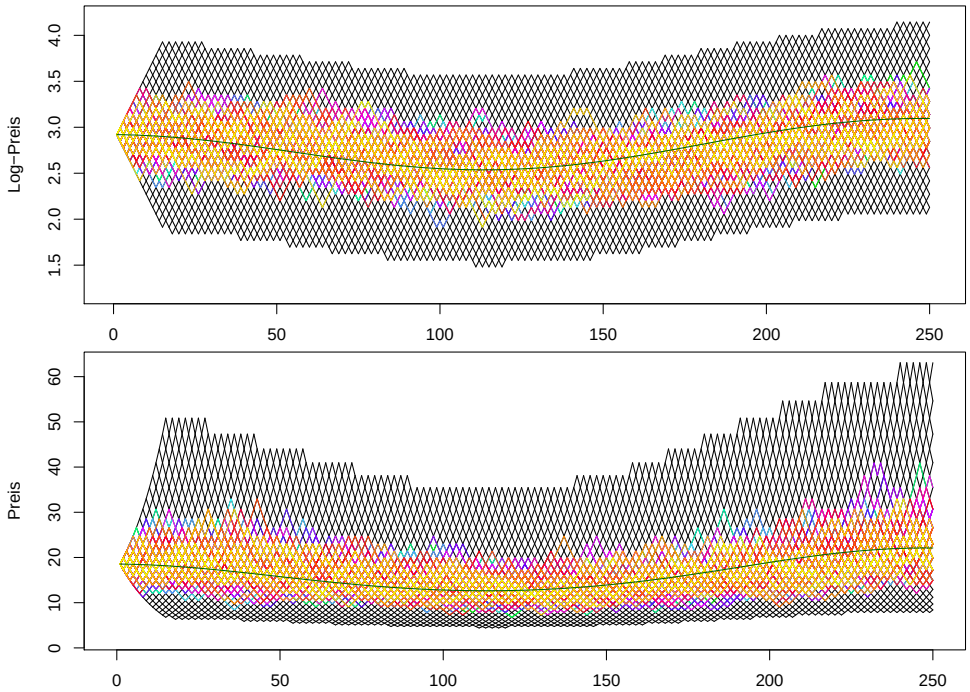


Abbildung 5.7.: Relevantes Knotennetz (schwarz) des Binomialprozesses der Log-Preise und Preise mit Simulationen (bunt) im zeitlichen Verlauf

Die Schaubilder 5.8 und 5.9 veranschaulichen den Einfluss einer zeitabhängigen Volatilität, indem sie die Knoten des Binomialbaumprozesses, sowie simulierte Pfade (bunt) zeigen, ähnlich Abbildung 5.6 und 5.7 für konstante Volatilität.

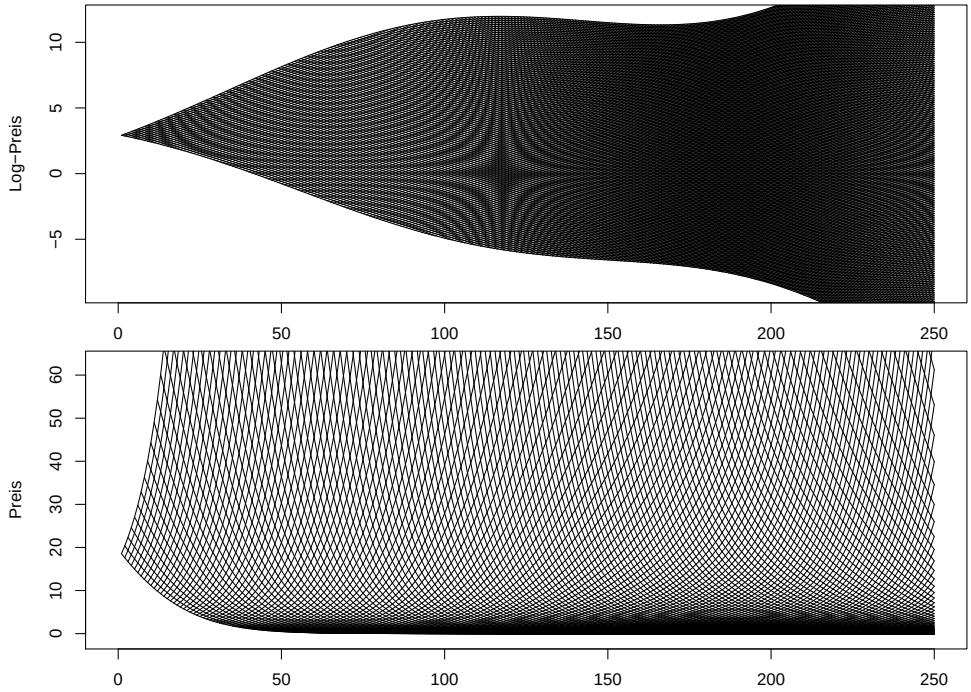


Abbildung 5.8.: Knotennetz des Binomialprozesses der Log-Preise und Preise mit zeitabhängiger Volatilität im zeitlichen Verlauf

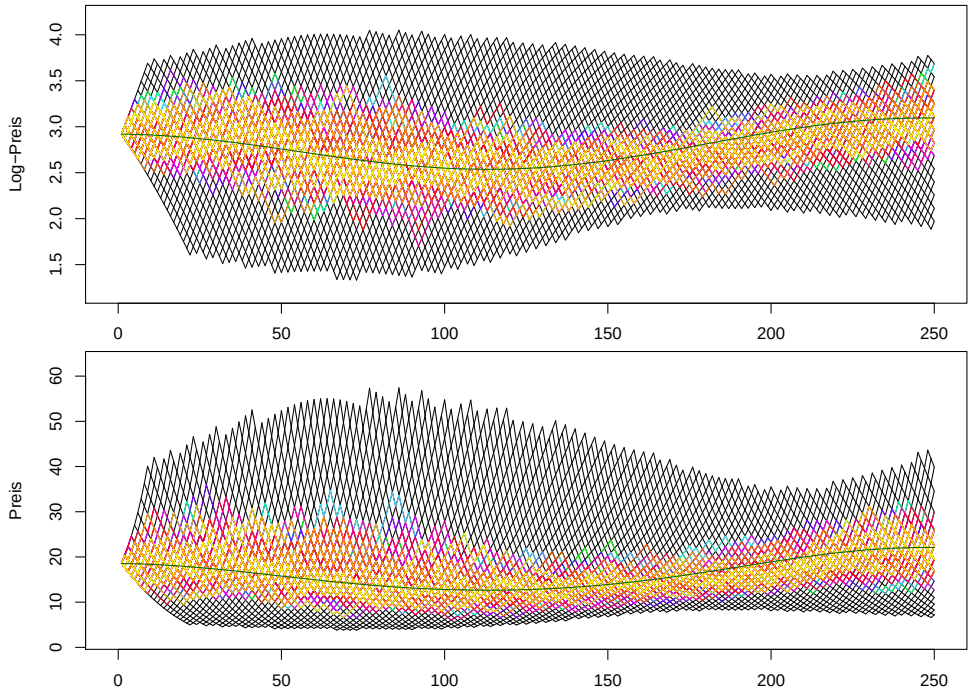


Abbildung 5.9.: Relevantes Knotennetz (schwarz) des Binomialprozesses der Log-Preise und Preise mit Simulationen (bunt) mit zeitabhängiger Volatilität im zeitlichen Verlauf

6 Vorstellung der Algorithmen zur Gasspeicherbewertung

Aufbauend auf dem gewählten Preismodell lassen sich konkrete Algorithmen zur Berechnung beziehungsweise Schätzung des Gasspeicherwertes konstruieren. Die Formulierung als Stochastisches Dynamisches Programm und die Bellman-Gleichung liefern dabei die Grundlage. Ausgangspunkte oder Ideen stammen häufig aus der Optionspreisbewertung, im Speziellen der Bewertung Amerikanischer Optionen, so wie zum Beispiel beim sogenannten LS-Algorithmus.

Im ersten Teil des Kapitels wird die grundlegende Struktur aller Algorithmen besprochen und auf die zentrale Frage hingewiesen, wie der Continuation Value zu berechnen ist. Es folgt die Beschreibung mehrerer Algorithmen zur Gasspeicherbewertung, zum einen ein Binomialbaum-Algorithmus basierend auf der Approximation von Kapitel 5, zum anderen Monte Carlo Algorithmen basierend auf der Simulation von Kapitel 5. Dabei werden Varianten eines Monte Carlo Algorithmus diskutiert. Im letzten Teil werden Methoden vorgestellt, die zur Schätzung einer oberen Schranke des Gaspeichwertes verwendet werden können.

6.1. Die grundlegende Struktur der Algorithmen unter Verwendung der Struktur der optimalen Politik

Beschreibung der Basisstruktur

Bevor in den folgenden Teilkapiteln spezielle Algorithmen vorgestellt werden, soll zunächst in diesem Teilkapitel die grundlegenden Struktur der Algorithmen diskutiert werden.

Der Rückwärtsinduktionsalgorithmus aus Kapitel 2 liefert das Vorgehen zur Berechnung des Speicherwerts V_0 und ist somit auch die Basis jedes Algorithmus. Dieser Basisalgorithmus berechnet sowohl die Wertefunktion V_n zu jedem Zeitpunkt $n \in \{0, 1, \ldots, N\}$,

als auch eine optimale Strategie. Da Satz 4.3.4 jedoch die Struktur der optimalen Politik in dem betrachteten Modell bereits liefert, vereinfacht sich das Vorgehen.

Bemerkung 6.1.1
Die nachfolgenden Betrachtungen basieren auf dem Preismodell aus Kapitel 5, das heißt ohne regime switching Komponente. Es ist dennoch zu bemerken, dass die grundlegende Struktur sich durch eine solche Komponente nur geringfügig ändert.
Es muss allerdings gewährleistet sein, dass ein geeignetes, den jeweiligen Preisprozess approximierendes, Gitter in der Preisdimension erzeugt werden kann.

Zur Veranschaulichung, der im folgenden beschriebenen Bassistruktur, dient Abbildung 6.1. Die Pfeile beschreiben `for`-Schleifen.

Es wird davon ausgegangen, dass sowohl das Intervall $[b^{\min}, b^{\max}]$, welches das jeweilige Speichervolumen (x) angibt, als auch der Wertebereich des Preisprozesses $\mathcal{P}$ in geeigneter Form diskretisiert sind. Für die Diskretisierung in der Preisvariablen p werden zum Beispiel die Methoden aus Kapitel 5.2 verwendet. Zu beachten ist, dass bei der Verwendung dieser Methoden, das Gitter in p zeitlich variiert. Das bedeutet, dass beispielsweise das Gitter zur Zeit n zu dem Gitter zur Zeit $n+1$ (fast sicher) verschieden ist.
In einem ersten Schritt wird die Wertefunktion V_N zum Zeitpunkt N für jedes x und p im jeweiligen Gitter ausgewertet. In der Zeit rückwärtsgehend, gekennzeichnet durch den orangenen Pfeil, werden für jedes $n \in \{N-1, \dots, 0\}$ und für jedes p, gekennzeichnet durch den roten Pfeil, folgende Operationen durchgeführt. Zunächst werden die Strategieschranken $\underline{b}_n(p)$ und $\overline{b}_n(p)$ berechnet. Hierzu muss U_n für alle x berechnet werden, um die gesuchten Maximisatoren der Optimierungsprobleme zu finden. Zuletzt wird dann die optimale Strategie gemäß Satz 4.3.4 bestimmt und der gewünschte Wert V_n für alle x ausgewertet.

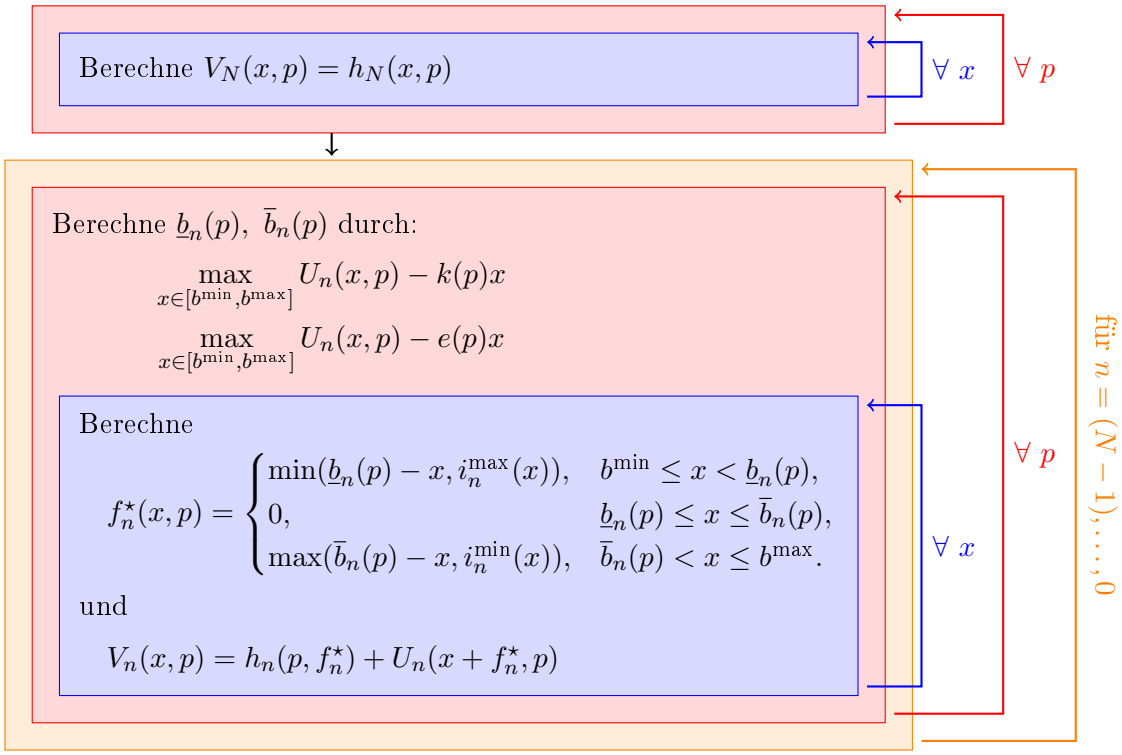


Abbildung 6.1.: Grundlegende Struktur der Algorithmen

Ist man lediglich am Wert des Gasspeichers, das heißt an V_0, interessiert, so ist es nicht nötig V_n für alle $x,\ p$ und n in einem Array zu speichern, es würde genügen V_n für alle x und p zu speichern und im nächsten Schritt mit V_{n-1} zu überschreiben. Genauso verhält es sich mit den Strategieschranken. Ist man an ihnen nicht weiter interessiert, kann man Speicherplatz sparen, indem man sich jeweils nur die zwei Werte $\underline{b}$ und $\overline{b}$ „merkt“ und dann immer wieder überschreibt.
Da in Kapitel 7 unter anderem die Strategieschranken, die letztendlich auch die Struktur der optimalen Politik angeben, untersucht werden, sind die Algorithmen, die in den folgenden Teilkapiteln beschrieben werden, so implementiert, dass diese Werte gespeichert werden. Hierfür werden häufig die Strategie-Schranken für alle p bestimmt, bevor die optimale Strategie und die Wertefunktion berechnet werden. Allerdings muss in diesem Fall auch U_n für alle x und p in einer Matrix gespeichert werden, im anderen Fall wäre es lediglich ein Vektor in x, der im nächsten Schritt wieder überschrieben werden kann. Dennoch hängt es stark von der Berechnung von U_n - und auch von der Programmiersprache - ab, inwiefern dies einen Nachteil darstellt.

Aus Abbildung 6.1 ist das zentrale Problem zu erkennen, das sich bei der Entwicklung geeigneter Algorithmen stellt:

Wie berechnet sich $U_n(x,p) = \mathbb{E}[V_{n+1}(x, P_{n+1})|P_n = p]$?

Alle anderen Größen im Algorithmus sind gegeben, so zum Beispiel die Gitter in $x,\ p$ und n, die Gewinnfunktionen h_n und h_N, sowie die Preisfunktionen k und e , oder lassen sich aus U_n berechnen, so wie $\underline{b}$ und $\overline{b}$, die optimale Politik $f^\star$ und auch die Wertefunktion V_n.

Die in den folgenden Unterkapiteln beschriebenen Algorithmen unterscheiden sich also im Wesentlichen in der Berechnung von U_n.

Bevor nun die Ansätze zur Berechnung von U_n erläutert werden, wird noch das Potential dieser Basisstruktur im Vergleich zur Verwendung des „normalen“ Rückwärtsalgorithmus aus Kapitel 2 in der grundlegenden Berechung der Wertefunktion und auch im Fall, der zusätzlichen Speicherung von $\underline{b}$ und $\overline{b}$ beschrieben.

Potential der Algorithmen unter Verwendung der Struktur der optimalen Politik

Betrachtet man zum Vergleich zur oben beschriebenen Struktur der Algorithmen, die Struktur, die man erhält, wenn man nicht die Struktur der optimalen Strategie verwendet, so würde in Abbildung 6.1 der zweite Block wie in Abbildung 6.2 aussehen.

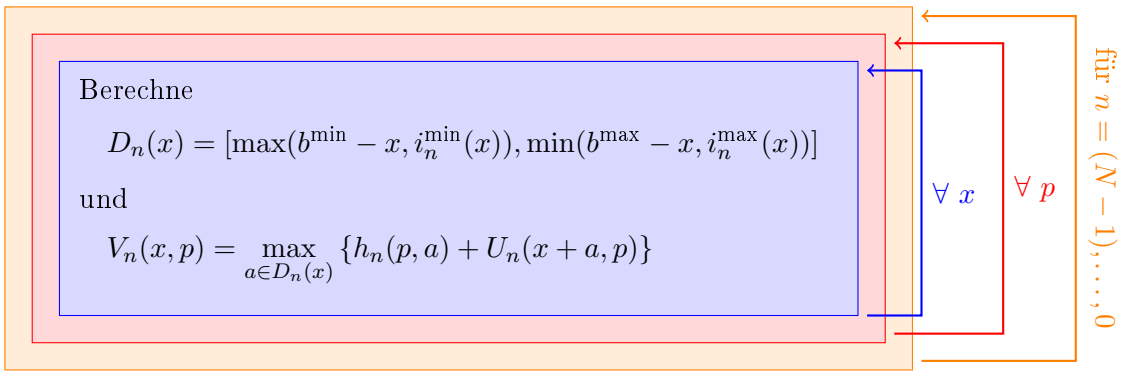


Abbildung 6.2.: Grundlegende Struktur der Algorithmen ohne Verwendung der Struktur der optimalen Strategie

Man müsste demnach für jedes x zuerst das Intervall D_n bestimmen und dann die Optimierung durchführen. Vorteilhaft ist, dass hier die Optimierung (meist) über weniger Gitterpunkte erfolgt und $U_n(\cdot, p)$ nur für die möglichen zukünftigen Zustände berechnet werden muss. Allerdings ist diese Berechnung für jedes x durchzuführen und da $a = 0$ immer zulässig ist, wertet man am Ende doch $U_n(\cdot, p)$ für jedes x aus. Es ist jedoch möglich, dies in einer Weise zu tun, dass weniger Speicherplatz benötigt wird, indem bereits berechnete Werte weiterverwendet und nicht mehr benötigte Werte wieder überschrieben werden. Dennoch bleibt h_n für alle zulässigen Aktionen a für jedes x auszuwerten.

Beim Vergleichen der beiden Abbildungen 6.1 und 6.2 ist zu erkennen, dass die Verwendung der Struktur der optimalen Politik eine Entkopplung der Berechnung von U_n und V_n mit sich bringt. Dies wäre selbstverständlich auch im ursprünglichen Algorithmus möglich, jedoch geht dann der im vorigen Abschnitt beschriebene Speichervorteil wieder verloren.

Des Weiteren haben die Algorithmen mit Verwendung der Struktur der optimalen Politik das Potential für eine weitere Entkopplung. Es können die Strategie-Schranken $\underline{b}_n$ und $\overline{b}_n$, gleich für alle p berechnet werden, bevor man die optimale Strategie $f^\star$ und den Wert V_n für alle x und p ermittelt. Dies geschieht zwar zu Lasten von Speicherplatz, dennoch macht es keinen Unterschied, wenn die Strategie-Schranken zu weiteren Untersuchungen bezüglich der optimalen Strategie gespeichert werden.

Somit ergeben sich bereits auf den ersten Blick - ohne detaillierte Betrachtung der Algorithmen - mehrere Parallelisierungsmöglichkeiten. Es lässt sich zum Beispiel $U_n(\cdot, p)$ für jeden Wert x parallel berechnen, genauso wie die optimale Strategie $f^\star$ und die Wertefunktion $V_n(\cdot, p)$. Je nach Dimension der Gitter in x und p und Beschaffenheit des Rechners lässt sich zusätzlich durch die beschriebene Entkopplung der Strategie-Schranken noch in p parallelisieren.

Verwendet man die Statistik-Software R, so ist es von Vorteil mit Vektoren und Matrizen anstelle der `for`-Schleifen zu arbeiten. Die beschriebene Entkopplung hilft auch hier.

Grundlegendes Vorgehen verbunden mit der Gitterdiskretisierung

Im abschließenden Abschnitt dieses Teilkapitels wird besprochen, wie Probleme, die im Zusammenhang mit der Gitterdiskretisierung entstehen, in allen folgenden Algorithmen behandelt werden.

Sobald mit einem Diskretisierungsgitter im Speicherstand x gearbeitet wird und Funktionen von dieser Variablen auswertet werden, stellt sich die Frage, wie reagiert wird, wenn die Gitterpunkte nicht wieder getroffen werden.

Zum einen ist dies bei der Berechnung möglicher zukünftiger Zustände durch $x+i_n^{\min}(x)$ und $x+i_n^{\max}(x)$ der Fall. Wenn die optimale Strategie $i_n^{\min}(x)$ oder $i_n^{\max}(x)$ ist und der neue Zustand $x+i_n^{\min}(x)$ oder $x+i_n^{\max}(x)$ berechnet wird, ist es nicht unbedingt gewährleistet, dass ein Gitterpunkt getroffen wird.

Zum anderen wird in der Berechnung der Strategie-Schranken $\underline{b}_n(p)$ und $\overline{b}_n(p)$ das Maximum eines Vektors bestimmt bzw. die Stelle an der, der Vektor maximal wird. Es ist möglich, dass zwei Werte des zu maximierenden Vektors gleich sind. In dem Fall muss entschieden werden, welcher dieser Werte zur Berechnung der Strategie-Schranken verwendet wird.

Die Vorgehensweise in diesen Fällen ist in nachfolgender Bemerkung zusammengefasst und die Bezeichnungen werden eingeführt, die in den Algorithmen der Kapitel 6.2 und folgende, verwendet werden.

Bemerkung 6.1.2

a) Der Vektor `x` gebe eine Diskretisierung des Speicherstandes an, wobei `x`$[j]$, den j-ten Eintrag bezeichne.
In dieser Arbeit wurde festgelegt, in dem einen Fall, dass $x+i_n^{\min}(x)$ bzw. $x+i_n^{\max}(x)$ keinen Gitterpunkt treffen, den nächstgrößeren bzw. nächstkleineren Gitterpunkt zu verwenden, sodass der jeweilige entstehende neue Zustand im Intervall $[x+i_n^{\min}(x), x+i_n^{\max}(x)]$ liegt. Der jeweilige Index dieser Gitterpunkte wird im beschriebenen Algorithmus mit `j.min` bzw. `j.max` bezeichnet. Die Funktion `berechnung.index.jneu` berechnet diese Indizes in Abhängigkeit des aktuellen Speicherstands und gibt als erste Komponente `j.min` und als zweite `j.max` zurück.

b) In dieser Arbeit wurde angenommen, dass man bei der Berechnung von $\underline{b}_n(p)$ - Maximierung von $U_n(x,p)-k(p)x$ in x - den kleineren Index und bei der Berechnung $\overline{b}_n(p)$ - Maximierung von $U_n(x,p)-e(p)x$ in x - den größeren Index nimmt. Tendenziell wird also der Bereich, indem nichts unternommen wird, vergrößert. Es ist dennoch zu bemerken, dass dieser Fall selten auftritt. Mit `index.b.u` bzw. `index.b.o` werden die Indizes bezeichnet, an denen der jeweilige Vektor maximal wird. Die Grenzen selbst sind also `x[index.b.u]` bzw. `x[index.b.o]`. Es wird mit `berechnung.index.max` eine Funktion bezeichnet, die den Index eines Vektors bestimmt, an dem der Vektor maximal wird. Aus Lesbarkeitsgründen wird auf eine

Unterscheidung der oben beschriebenen Fälle verzichtet.

Tatsächlich werden in den Algorithmen zunächst die - unter anderem - möglichen neuen Zustände $x + i_n^{\min}(x)$ und $x + i_n^{\max}(x)$, sowie die Strategie-Schranken $\underline{b}_n(p)$ und $\bar{b}_n(p)$, bestimmt und dann die optimale Strategie berechnet.

Bemerkung 6.1.3
In den Algorithmen wird vielmehr der Index des Zustands bestimmt, den man unter Verwendung der Struktur der optimalen Strategie erhält. Dieser Index wird mit `index.fstar` bezeichnet, das heißt der nächste Zustand ist `x[index.fstar]`. Befindet man sich in Zustand `x`$[j]$, so ist die optimale Strategie `x[index.fstar]-x`$[j]$. Es wird mit `berechnung.index.fstar` die Funktion bezeichnet, die in Abhängigkeit von Preis und Zustand (und Zeit) diesen Index bestimmt. Algorithmus 3 zeigt den Verlauf dieser Funktion.

Algorithmus 3 : Funktion `berechnung.index.fstar`

Data : j, index.b.u, index.b.o, j.min,j.max
Result : index.fstar

if $j <$ index.b.u ***and*** index.b.u $\leq$ j.max **then**
  index.fstar = index.b.u ;
else if $j <$ index.b.u ***and*** index.b.u $>$ j.max **then**
  index.fstar = j.max;
else if $j >$ index.b.o ***and*** index.b.o $\geq$ j.min **then**
  index.fstar = index.b.o;
else if $j >$ index.b.o ***and*** index.b.o $<$ j.min **then**
  index.fstar = j.min;
else
  index.fstar = j;
end

6.2. Ein Binomialbaum-Algorithmus

Der „Binomialbaum-Algorithmus“ unterstellt, dass der Preisprozess, wie im Approximationsteil von Kapitel 5.2 beschrieben, durch einen Binomialprozess approximiert wird.

In diesem Fall ist es bestimmend, welche Unterteilung für das Intervalls $[0, N]$ gewählt wird. Da die Wertefunktion des Gasspeicherproblems an diskreten Zeitpunkten ausgewertet werden soll, wird davon ausgegangen, dass $\Delta t \leq 1$ gilt. Außerdem soll ein $l \in \mathbb{N}$ existieren, sodass $l \cdot \Delta t = 1$ gilt. Damit kann die Zufallsvariable P_{n+1} bedingt unter

$P_n = p$, l verschiedene Werte $p_{n+1}^{(p,1)}, \ldots, p_{n+1}^{(p,l)}$ annehmen und U_n berechnet sich als

$$U_n(x,p) = \mathbb{E}[V_{n+1}(x, P_{n+1})|P_n = p] = \sum_{j=1}^{l} V_{n+1}(x, p_{n+1}^{(p,j)}) \cdot \mathbb{P}(P_{n+1} = p_{n+1}^{(p,j)}|P_n = p). \tag{6.1}$$

Da es sich um einen rekombinierenden Binomialbaum handelt, erhält man zum Zeitpunkt n maximal $1 + l \cdot n$ mögliche Preise - und damit Gitterknoten in p. Allerdings ist zusätzlich zu beachten, dass wie in Kapitel 5.2 gesehen, einige Pfade - und damit Knotenpunkte - die Wahrscheinlichkeit 0 besitzen und somit in der Auswertung nicht beachtet werden müssen.

Im Folgenden werden die zwei Spezialfälle $l = 1$ und $l = 2$ betrachtet. Aus diesen ist erkennbar, wie der Algorithmus für allgemeines l umzusetzen ist. In Kapitel 7 werden die Fälle $l = 1,\ l = 2,\ l = 3,\ l = 4$ und $l = 5$ dann genauer untersucht.

Die Basisstruktur aus Abbildung 6.1 bildet die Grundlage der Algorithmen und soll als Übersicht dienen. Ersetzt man hier $U_n(x,p)$ durch den Ausdruck in (6.1) mit $l = 1$ bzw. $l = 2$, so erhält man die Struktur des Algorithmus am Binomialbaum, auf der die folgenden detaillierte Beschreibung aufbaut.

Beschreibung des Binomialbaum-Algorithmus im Spezialfall $\Delta t = 1$

Betrachtet man die Generierung der Matrizen `P = exp(X), Up, Down`, so fällt auf, dass die Einträge unterhalb der Diagonalen nicht besetzt sind. Sie werden mit sogenannten `NA`(=not available) gefüllt. Außerdem sind die Einträge in `P` „unnötig“, die eine Erreichungswahrscheinlichkeit von Null besitzen. All diese Einträge werden im Vorfeld mittels einer Abfrage durch `NA` ersetzt. Nun lässt sich die Dimension der Matrix `P` insofern verkleinern, als dass alle Zeilen mit ausschließlich `NA`-Einträgen gelöscht werden. Die Matrix `P` ist somit nur noch von der Dimension `dim.p` $\times (N+1)$ und es wird weniger Speicherplatz benötigt. Aufgrund der Struktur der Matrix werden allerdings nur „untere“ Pfade gelöscht. Um dennoch diesen Vorteil auszunutzen, werden zwei Indizes `index.u` und `index.o` definiert, welche die Zeilen der Spalte `P`$[,t]$ der Matrix `P` beschreiben, in der zum ersten bzw. zum letzten Mal ein numerischer Eintrag (ungleich `NA`) steht. Es ist zu beachten, dass `P`$[,t]$ die $(t+1)$-te Spalte der Matrix `P` beschreibt. Die Indizes werden für jede Zeit t durch die Funktionen `berechnung.index.unten` bzw. `berechnung.index.oben` abhängig von der Spalte `P`$[,t]$ berechnet. Damit ist auch der Wert des Speichers `NA`, wenn der zugehörige Preis `NA` darstellt.

Das Gitter des Speicherzustandes sei gegeben in einem Vektor `x` der Länge G.

Algorithmus 4 beschreibt das Vorgehen im Fall $\Delta t = 1$. Der Wert des Speichers wird in einem dreidimensionalen Array `Wert` der Dimension $(N+1) \times G \times$ `dim.p` gespeichert. Der Eintrag `Wert`$[t, j, m]$ entspricht $V_t(\texttt{x}[j], \texttt{P}[m, t])$. Die Strategie-Schranken sollen ebenfalls gespeichert werden, und zwar in den $(N-1) \times (\texttt{dim.p} - 1)$-Matrizen `b.unten` und `b.oben`. Hier entspricht der Eintrag `b.unten`$[t, m]$ dem Wert $\underline{b}_t(\texttt{P}[m, t])$ und `b.oben`$[t, m]$ entsprechend $\overline{b}_t(\texttt{P}[m, t])$.

Es wird davon ausgegangen, dass die Funktionen `berechnung.index.max`, `berechnung.index.jneu` und `berechnung.index.fstar`, die in Bemerkung 6.1.2 und Bemerkung 6.1.3 beschrieben sind, bereits implementiert sind und zur Verfügung stehen.

Die Besonderheiten der einzelnen Algorithmen sind blau hervorgehoben. Beispielsweise sind das wichtige Zeilen, die sich zentral von denen in den andern Algorithmen unterscheiden oder hinzugefügt oder weggelassen werden.

Je nach Programmiersprache und deren Umgang mit `NA`, ist die **if**-Abfrage, ob entsprechende Einträge in `Wert` not available sind, nicht notwendig.

Algorithmus 4 : Binomialbaum-Algorithmus im Fall $\Delta t = 1$

Data : Parameter des Gasspeichers, Speichergitter x der Länge G, Preisbaummatrizen P, Up, Down

Result : $3d$-Array Wert des Speicherwerts, Matrizen b.unten und b.oben der Strategie-Schranken.

for $j \in \{1, \ldots, G\}$ **do**

  index.u = berechnung.index.unten(P[, N]);
  index.o = berechnung.index.oben(P[, N]);

  for $m \in \{$index.u, $\ldots$, index.o$\}$ **do**

    Wert$[N, j, m] = h_N(\text{x}[j], \text{P}[m, N])$;

  end

end

for $t \in \{N-1, \ldots, 0\}$ **do**

  index.u = berechnung.index.unten(P[, t]);
  index.o = berechnung.index.oben(P[, t]);

  for $m \in \{$index.u, $\ldots$, index.o$\}$ **do**

    for $j \in \{1, \ldots, G\}$ **do**

      if Wert$[t+1, j, m] ==$ NA ***and*** Wert$[t+1, j, m+1] ==$ NA **then**
      $U_t(\text{x}[j], \text{P}[m, t]) = 0$;
      else if Wert$[t+1, j, m] !=$ NA ***and*** Wert$[t+1, j, m+1] ==$ NA **then**
        $U_t(\text{x}[j], \text{P}[m, t]) = \text{Up}[m, t] \cdot \text{Wert}[t+1, j, m]$;
      else if Wert$[t+1, j, m] ==$ NA ***and*** Wert$[t+1, j, m+1] !=$ NA **then**
        $U_t(\text{x}[j], \text{P}[m, t]) = \text{Down}[m, t] \cdot \text{Wert}[t+1, j, m+1]$;
      else
        $U_t(\text{x}[j], \text{P}[m, t]) = \text{Up}[m, t] \cdot \text{Wert}[t+1, j, m] + \text{Down}[m, t] \cdot \text{Wert}[t+1, j, m+1]$;
      end

      hilfs.vector1 $= U_t(\text{x}[j], \text{P}[m, t]) - k(\text{P}[m, t]) \cdot \text{x}[j])$;
      hilfs.vector2 $= U_t(\text{x}[j], \text{P}[m, t]) - e(\text{P}[m, t]) \cdot \text{x}[j])$;

    end

    index.b.u = berechnung.index.max(hilfs.vector1);
    index.b.o = berechnung.index.max(hilfs.vector2);

    b.unten$[t, m]$ = x[index.b.u];
    b.oben$[t, m]$ = x[index.b.o];

    for $j \in \{1, \ldots, G\}$ **do**

      j.min = berechnung.index.jneu(x$[j]$)[1];
      j.max = berechnung.index.jneu(x$[j]$)[2];

      index.fstar = berechnung.index.fstar(j, index.b.u, index.b.o, j.min, j.max);

      Wert$[t, j, m] = h_n(\text{P}[m, t], \text{x}[\text{index.fstar}] - \text{x}[j]) + U_t(\text{x}[\text{index.fstar}], \text{P}[m, t])$;

    end

  end

end

Beschreibung des Binomialbaum-Algorithmus im Spezialfall $\Delta t = 1/2$

Im Fall $\Delta t = 1/2$, ist die durch Algorithmus 2 erzeugte Matrix `X` von der Dimension $(2N+1) \times (2N+1)$. Allerdings werden N der Spalten nicht benötigt, da die Preise zu den Zeitpunkten $\Delta t, 3\Delta t, 5\Delta t, \ldots$ zur Berechnung des Speicherwertes nicht relevant sind. Werden diese Spalten aus der Matrix `X` gelöscht und die Einträge der daraus gewonnenen Matrix auf die Exponentialfunktion angewendet, so erhält man die relevante Preis-Matrix `P`.

Ist der Preis zum Zeitpunkt t durch `P`$[m,t]$ gegeben, so kann man im nächsten Zeitschritt des Algorithmus - zur Zeit $t+1$ - genau drei Preise erreichen, `P`$[m,t+1]$, `P`$[m+1,t+1]$ und `P`$[m+2,t+1]$. Dementsprechend benötigt man drei Matrizen, die die Wahrscheinlichkeiten speichern. Diese werden hier mit `Up`, `Middle`, `Down` bezeichnet. Es ist zu beachten, dass streng genommen, zwei dieser Matrizen ausreichen, da sich die dritte aus den andern zweien berechnen lässt, denn es gilt `Up` + `Middle` + `Down` = `Einsmatrix`. Es ist festzustellen, dass man im Algorithmus direkt mit den ursprünglich durch Algorithmus 2 berechneten Matrizen $\mathtt{Up}^{\mathrm{alt}}$ und $\mathtt{Down}^{\mathrm{alt}}$ rechnen kann.

Algorithmus 5 zeigt, wie die „neuen“ Wahrscheinlichkeitsmatrizen im Fall $\Delta t = 1/2$ berechnet werden. Ausgehend von `P`$[m,t]$ gelangt man nach `P`$[m,t+1]$ durch eine „Up-Up“-Bewegung, nach `P`$[m+1,t+1]$ durch eine „Up-Down“- oder „Down-Up“-Bewegung und nach `P`$[m+2,t+1]$ durch eine „Down-Down“-Bewegung.

Algorithmus 5 : Berechnung der neuen Wahrscheinlichkeitsmatrizen

Data : Preisbaummatrizen $\mathsf{Up}^{\mathrm{alt}}, \mathsf{Down}^{\mathrm{alt}}$
Result : neue Preisbaummatrizen $\mathsf{Up}, \mathsf{Middle}, \mathsf{Down}$
for $t \in \{0,1,\ldots,N-1\}$ **do**
  for $m \in \{0,1,\ldots,2t\}$ **do**

$$\mathsf{Up}[m,t] = \mathsf{Up}^{\mathrm{alt}}[m,2t] \cdot \mathsf{Up}^{\mathrm{alt}}[m,2t+1];$$

$$\mathsf{Middle}[m,t] = \begin{array}{l} \mathsf{Up}^{\mathrm{alt}}[m,2t] \cdot \mathsf{Down}^{\mathrm{alt}}[m,2t+1] \\ +\mathsf{Down}^{\mathrm{alt}}[m,2t] \cdot \mathsf{Up}^{\mathrm{alt}}[m+1,2t+1] \end{array};$$

$$\mathsf{Down}[m,t] = \mathsf{Down}^{\mathrm{alt}}[m,2t] \cdot \mathsf{Down}^{\mathrm{alt}}[m+1,2t+1];$$

  end
end

Wie schon im Fall $\Delta t = 1$ werden die Einträge der Matrix `P`, die fast sicher nicht erreicht werden, durch `NA` ersetzt und anschließend die Dimension der Matrix `P` verkleinert, sodass sie die Dimension `dim.p` $\times (N+1)$ besitzt. Die Größe `dim.p` ist hier nicht dieselbe wie im Fall $\Delta t = 1$.

Mit den Bezeichnungen der Funktionen und Indizes wie im Fall $\Delta t = 1$ erhält man Algorithmus 6.

Aus Gründen der Lesbarkeit wurde bei der Berechnung von $U_t(\mathtt{x}[j], \mathtt{P}[m,t])$ auf die **if**-Abfrage bzgl. der `NA`-Einträge in `Wert` verzichtet. In der eigentlichen Implementierung muss dies jedoch beachtet werden. Die Zeile, die sich im Vergleich zu Algorithmus 4 ändert ist grün hervorgehoben.

Algorithmus 6 : Binomialbaum-Algorithmus im Fall $\Delta t = \frac{1}{2}$

Data : Parameter des Gasspeichers, Speichergitter x der Länge G, Preisbaummatrizen P, Up, Middle, Down
Result : $3d$-Array Wert des Speicherwerts, Matrizen b.unten und b.oben der Strategie-Schranken.

for $j \in \{1, \ldots, G\}$ **do**
  index.u = `berechnung.index.unten`(P$[, N]$);
  index.o = `berechnung.index.oben`(P$[, N]$);
  for $k \in \{$index.u$, \ldots,$ index.o$\}$ **do**
    Wert$[N, j, k] = h_N($x$[j],$ P$[k, N])$;
  end
end
for $t \in \{N-1, \ldots, 0\}$ **do**
  index.u = `berechnung.index.unten`(P$[, t]$);
  index.o = `berechnung.index.oben`(P$[, t]$);
  for $m \in \{$index.u$, \ldots,$ index.o$\}$ **do**
    for $j \in \{1, \ldots, G\}$ **do**
      $U_t($x$[j],$ P$[m,t]) =$ Up$[m,t] \cdot$ Wert$[t+1, j, m]$
      $+$ Middle$[m,t] \cdot$ Wert$[t+1, j, m+1]$
      $+$ Down$[m,t] \cdot$ Wert$[t+1, j, m+2]$;
      hilfs.vector1 $= U_t($x$[j],$ P$[m,t]) - k($P$[m,t]) \cdot$ x$[j])$;
      hilfs.vector2 $= U_t($x$[j],$ P$[m,t]) - e($P$[m,t]) \cdot$ x$[j])$;
    end
    index.b.u = `berechnung.index.max`(hilfs.vector1);
    index.b.o = `berechnung.index.max`(hilfs.vector2);
    b.unten$[t, m]$ = x[index.b.u];
    b.oben$[t, m]$ = x[index.b.o];
    for $j \in \{1, \ldots, G\}$ **do**
      j.min = `berechnung.index.jneu`(x$[j]$)$[1]$;
      j.max = `berechnung.index.jneu`(x$[j]$)$[2]$;
      index.fstar = `berechnung.index.fstar`$(j,$ index.b.u, index.b.o, j.min, j.max$)$;
      Wert$[t, j, m] = h_n($P$[m,t],$ x[index.fstar] $-$ x$[j]) + U_t($x[index.fstar], P$[m,t])$;
    end
  end
end

6.3. Monte-Carlo Algorithmen

Den folgenden Monte-Carlo Algorithmen liegen M simulierte Preispfade zu Grunde, die mittels Algorithmus 1 erzeugt werden. Das bedeutet zu jedem Zeitpunkt t liegen M realisierte Werte des Preisprozesses vor. Diese Werte seien in einer $M \times (N+1)$-Matrix `P` gespeichert. Der Eintrag `P`$[m,t]$ enthalte den Preis zur Zeit t der m-ten Simulation, der mit p_t^m bezeichnet wird. Demnach liegt m in $\{1,\dots,M\}$ und t wie gehabt in $\{0,1,\dots,N\}$.

Wie beim Binomialbaum-Algorithmus stellt sich hier die Frage, wie sich $U_n(x,p) = \mathbb{E}[V_{n+1}(x,P_{n+1})|P_n = p]$ berechnet. In einer Monte-Carlo Methode ist dies nicht eindeutig bestimmt. Die zwei folgenden Teilkapitel beschreiben verschiedene Ansätze, um den Continuation Value U zu ermitteln.

Im Gegensatz zum Baum-Algorithmus, bei dem das Gitter in der Preisdimension letztendlich fest ist, kann bei jeder neuen Durchführung der Monte-Carlo Simulationen ein anderes Gitter zu Grunde liegen. So erhält man auch verschiedene Werte des Gasspeichers. Letztendlich arbeitet man mit verschiedenen Realisierungen des zugrundeliegenden Stochastischen Prozesses. Aus diesem Grund ist es sinnvoll, eher von der Realisierung eines Schätzers zur Gasspeicherbewertung zu sprechen, als vom Wert des Gasspeichers. Man sollte demnach die Algorithmen mehrfach durchführen und dann Statistiken - wie zum Beispiel Mittelwert, Standardabweichung, Median, Quantile usw. - der erhaltenen Werte studieren.

6.3.1. Interpolationsansatz

Beim Interpolationsansatz wird ausgenutzt, dass die Verteilung von P_{n+1} gegeben $P_n = p$ im betrachteten Modell bekannt ist.
Für jedes feste x ist $V_{n+1}(x,\cdot)$ zur Zeit n eine Funktion des zukünftigen Preises, die durch den zuvor durchgeführten Berechnungsschritt auf dem Gitter $p_{n+1}^1,\dots,p_{n+1}^M$ als bekannt angenommen wird.

Der Ansatz ist, die Punkte $(p_{n+1}^1, V_{n+1}(x,p_{n+1}^1)),\dots,(p_{n+1}^M, V_{n+1}(x,p_{n+1}^M))$ als Stützpunkte für eine Interpolation zu verwenden und so die Funktion $V_{n+1}(x,\cdot)$ zu approximieren. Die Approximation der Funktion wird jeweils mit $\widetilde{V}$ bezeichnet. Es gilt also den Continuation Value durch

$$\widetilde{U}_n(x,p) = \mathbb{E}[\widetilde{V}_{n+1}(x,P_{n+1})|P_n = p] \tag{6.2}$$

mit Hilfe der Verteilung von P_{n+1} gegeben $P_n = p$ zu schätzen.

Im Folgende bezeichne die Funktion `interpolation` eine Routine, die bei Eingabe von Stützpunkten in Form zweier Vektoren die approximierende Funktion zurückgibt.

In der Implementierung in Kapitel 7 werden hierfür entsprechende Routinen aus R genutzt. Dabei werden zum einen Routinen zur linearen Interpolation verwendet, zum anderen natürliche Splines.

Bestimmung des Erwartungswertes durch ein Integral

Die erste Methode diesen Erwartungswert zu bestimmen, ist über die Transformationsformel. Sei dazu $d_p(\cdot)$ die Dichte der Verteilung von P_{n+1} geben $P_n = p$. Dann berechnet sich (6.2) durch

$$\widetilde{U}_n(x,p) = \int_{-\infty}^{\infty} \widetilde{V}_{n+1}(x,p') d_p(p')\ dp' \tag{6.3}$$

mittels numerischer Integration. Nach Korollar 5.1.7 ist im Fall des betrachteten Modells d_p die Dichte einer Lognormalverteilung mit Parameter $\mu_n := \mathbb{E}(X_{n+1}) + \rho\sqrt{\frac{\mathbb{V}(X_{n+1})}{\mathbb{V}(X_n)}}(\log p - \mathbb{E}(X_n))$ und $\sigma_n^2 := \mathbb{V}(X_n)(1-\rho^2)$ (für $n = 1, 2, \ldots, N$), wobei $\rho = \frac{\mathrm{Cov}(X_n, X_{n+1})}{\sqrt{\mathbb{V}(X_{n+1})\mathbb{V}(X_n)}}$. Damit hat d_p die Gestalt

$$d_p(p') = \frac{1}{\sqrt{2\pi}\sigma_n p'} e^{-\frac{(\log p' - \mu_n)^2}{2\sigma_n^2}} \mathbf{1}_{(0,\infty)}(p').$$

Die Zufallsvariable X_n bezeichne wie gehabt $\log P_n$. Für $n = 0$ wird $\mu_0 := \mathbb{E}(X_1)$ und $\sigma_0^2 := \mathbb{V}(X_1)$ gesetzt.

Für die numerische Integration wird in Kapitel 7 eine Routine aus R verwendet, daher bezeichne `integrate`$(f$, `lower, upper`$)$ eine Rountine, die eine gegebene Funktion numerisch von `lower` bis `upper` integriert und den Wert des Integrals zurückgibt. Die Routine der Statistiksoftware R verwendet eine adaptive Quadratur und basiert auf Routinen aus der Bibliothek QUADPACK. Diese Bibliothek ist ursprünglich eine Fortan 77 Bibliothek, die für C reimplementiert wurde. Für nähere Informationen wird auf GNU Scientific library [17] bzw. R user's guide [30] verwiesen.

Desweiteren bezeichnet in Algorithmus 7 `dlnorm` eine Funktion, welche - unter Eingabe der konkreten Parameter der Lognormalverteilung - die Dichte selbiger als Funktion von einer Variablen zurückgibt.

In den Schritten, die sich von den anderen Algorithmen grundlegend unterscheiden, wird also zunächst die Interpolation mit den entsprechenden Stützpunkten in Abhängigkeit des Zeitpunktes und des Speicherzustandes durchgeführt (blau markiert) und dann das entsprechende Integral berechnet (grün markiert). Die grüne Markierung soll verdeutlichen, was sich in den folgenden Abschnitten im Vergleich zur Methode mit Integral verändert.

Genau genommen ist es möglich, da die Interpolation nicht vom konkreten Pfad m abhängt, diese Berechnung vor die „**for** $m \in \{1, \ldots, M\}$ **do**"-Schleife zu stellen. Also zuvor die Interpolationen für alle $j \in \{1, \ldots, G\}$ durchzuführen und die Ergebnisse in einem Vektor zu speichern, der die Funktionen $\widetilde{V}_t(\mathtt{x}[1], \cdot), \ldots, \widetilde{V}_t(\mathtt{x}[G], \cdot)$ enthält.

Algorithmus 7 : Monte-Carlo Algorithmus mit Interpolationsansatz und Integration

Data : Parameter des Gasspeichers, Speichergitter x der Länge G, Preispfadmatrizen P
Result : $3d$-Array Wert des Speicherwerts, Matrizen b.unten und b.oben der Strategie-Schranken.

for $j \in \{1, \ldots, G\}$ **do**
  for $m \in \{1, \ldots, M\}$ **do**
    $\mathsf{Wert}[N, j, m] = h_N(\mathsf{x}[j], \mathsf{P}[m, N])$;
  end
end
for $t \in \{N-1, \ldots, 0\}$ **do**
  for $m \in \{1, \ldots, M\}$ **do**
    for $j \in \{1, \ldots, G\}$ **do**
      $\widetilde{V}_{t+1}(\mathsf{x}[j], \cdot) = \texttt{interpolation}(\mathsf{P}[\ , t+1], \mathsf{Wert}[t+1, j, \ ])$;
      if $t! = 0$ **then**
        $\mu_t = \mathbb{E}(X_{t+1}) + \rho\sqrt{\frac{\mathbb{V}(X_{t+1})}{\mathbb{V}(X_t)}}(\log \mathsf{P}[m, t] - \mathbb{E}(X_t))$;
        $\rho = \frac{\mathrm{Cov}(X_t, X_{t+1})}{\sqrt{\mathbb{V}(X_{t+1})\mathbb{V}(X_t)}}$;
        $\sigma_t^2 = \mathbb{V}(X_t)(1 - \rho^2)$;
      else
        $\mu_t = \mathbb{E}(X_{t+1})$;
        $\sigma_t^2 = \mathbb{V}(X_{t+1})$;
      end
      $d_p = \texttt{dlnorm}(\mu_t, \sigma_t^2)$;
      $\widetilde{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) = \texttt{integrate}(\widetilde{V}_{t+1}(\mathsf{x}[j], \cdot) \cdot d_p, 0, \infty)$;
      $\mathsf{hilfs.vector1} = \widetilde{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) - k(\mathsf{P}[m, t]) \cdot \mathsf{x}[j])$;
      $\mathsf{hilfs.vector2} = \widetilde{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) - e(\mathsf{P}[m, t]) \cdot \mathsf{x}[j])$;
    end
    $\mathsf{index.b.u} = \texttt{berechnung.index.max}(\mathsf{hilfs.vector1})$;
    $\mathsf{index.b.o} = \texttt{berechnung.index.max}(\mathsf{hilfs.vector2})$;
    $\mathsf{b.unten}[t, m] = \mathsf{x}[\mathsf{index.b.u}]$;
    $\mathsf{b.oben}[t, m] = \mathsf{x}[\mathsf{index.b.o}]$;
    for $j \in \{1, \ldots, G\}$ **do**
      $\mathsf{j.min} = \texttt{berechnung.index.jneu}(\mathsf{x}[j])[1]$;
      $\mathsf{j.max} = \texttt{berechnung.index.jneu}(\mathsf{x}[j])[2]$;
      $\mathsf{index.fstar} = \texttt{berechnung.index.fstar}(j, \mathsf{index.b.u}, \mathsf{index.b.o}, \mathsf{j.min}, \mathsf{j.max})$;
      $\mathsf{Wert}[t, j, m] = h_n(\mathsf{P}[m, t], \mathsf{x}[\mathsf{index.fstar}] - \mathsf{x}[j]) + \widetilde{U}_t(\mathsf{x}[\mathsf{index.fstar}], \mathsf{P}[m, t])$;
    end
  end
end

Bestimmung des Erwartungswertes durch eine „nested“ Simulation

Möchte man die Integration in (6.3) vermeiden, ergibt sich zum Beispiel die Möglichkeit, den Erwartungswert durch eine weitere Monte Carlo Simulation zu schätzen. Seien dazu $\widetilde{P}^1_{n+1}, \dots, \widetilde{P}^l_{n+1}$ unabhängig identisch verteilte Zufallsvariablen, die die selbe Verteilung wie P_{n+1} gegeben $P_n = p$ besitzen. Aus dem starken Gesetz großer Zahlen ist bekannt, dass für messbare Funktionen f

$$\frac{1}{l}\sum_{i=1}^{l} f(\widetilde{P}^l_{n+1}) \xrightarrow{f.s.} \mathbb{E}\left[f(\widetilde{P}^1_{n+1})\right] \qquad \text{für } l \to \infty$$

gilt.

Da der Erwartungswert $\mathbb{E}\left[f(\widetilde{P}^1_{n+1})\right]$ mit dem Erwartungswert $\mathbb{E}\left[f(P_{n+1})|P_n = p\right]$ übereinstimmt, ist die Idee zur Zeit n bei gegebenem p und x jeweils l (unabhängig voneinander erzeugte) Realisierungen der Zufallsvariablen mit Verteilung $\mathcal{L}(P_{n+1}|P_n = p)$ zu bestimmen und dann $U_n(x,p)$ durch

$$\widetilde{U}_n(x,p) = \frac{1}{l}\sum_{i=1}^{l} \widetilde{V}_{n+1}(x, \widetilde{p}^i_{n+1}) \tag{6.4}$$

zu approximieren. Hierbei bezeichnen $\widetilde{p}^1_{n+1}, \dots, \widetilde{p}^l_{n+1}$ die Realisierungen der Verteilung $\mathcal{L}(P_{n+1}|P_n = p)$ und $\widetilde{V}_{n+1}$ die V_{n+1}-approximierende Funktion. Aufgrund der erneuten Erzeugung von Realisierungen entsprechender Zufallsvariablen spricht man auch von einer „nested“ Simulation.

Da nach Korollar 5.1.7 bekannt ist, dass $\mathcal{L}(P_{n+1}|P_n = p)$ eine Lognormalverteilung ist, bezeichne `random.lognormal(`l`,` μ`,` σ^2`)` eine Funktion, die l Realisierungen einer lognormalverteilten Zufallsvariablen mit Parametern μ und σ^2 erzeugt. Diese Realisierungen seien im Vektor $\tilde{p}_{n+1}$ gespeichert, der die Einträge $\widetilde{p}^1_{n+1}, \dots, \widetilde{p}^l_{n+1}$ enthält. Ansonsten werden die gleichen Bezeichnungen wie vorher verwendet. Im Vergleich zu Algorithmus 7 sind die grün markierten Zeilen in Algorithmus 8 verändert.

Algorithmus 8 : Monte-Carlo Algorithmus mit Interpolationsansatz und „nested“ Simulation

Data : Parameter des Gasspeichers, Speichergitter x der Länge G, Preispfadmatrizen P, Anzahl der „nested“ Simulationen l

Result : $3d$-Array Wert des Speicherwerts, Matrizen b.unten und b.oben der Strategie-Schranken.

for $j \in \{1, \dots, G\}$ **do**
  for $m \in \{1, \dots, M\}$ **do**
    $\mathsf{Wert}[N, j, m] = h_N(\mathsf{x}[j], \mathsf{P}[m, N])$;
  end
end
for $t \in \{N-1, \dots, 0\}$ **do**
  for $m \in \{1, \dots, M\}$ **do**
    for $j \in \{1, \dots, G\}$ **do**
      $\widetilde{V}_{t+1}(\mathsf{x}[j], \cdot) = \texttt{interpolation}(\mathsf{P}[\ , t+1], \mathsf{Wert}[t+1, j, \ ])$;
      if $t! = 0$ **then**
        $\mu_t = \mathbb{E}(X_{t+1}) + \rho\sqrt{\frac{\mathbb{V}(X_{t+1})}{\mathbb{V}(X_t)}}(\log \mathsf{P}[m, t] - \mathbb{E}(X_t))$;
        $\rho = \frac{\mathrm{Cov}(X_t, X_{t+1})}{\sqrt{\mathbb{V}(X_{t+1})\mathbb{V}(X_t)}}$;
        $\sigma_t^2 = \mathbb{V}(X_t)(1-\rho^2)$;
      else
        $\mu_t = \mathbb{E}(X_{t+1})$;
        $\sigma_t^2 = \mathbb{V}(X_{t+1})$;
      end
      $\widetilde{p}_{t+1} = \texttt{random.lognormal}(l, \mu_t, \sigma_t^2)$;
      $\widetilde{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) = \frac{1}{l}\sum_{i=1}^{l} \widetilde{V}_{t+1}(\mathsf{x}[j], \widetilde{p}_{t+1}^i)$;
      $\mathsf{hilfs.vector1} = \widetilde{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) - k(\mathsf{P}[m, t]) \cdot \mathsf{x}[j])$;
      $\mathsf{hilfs.vector2} = \widetilde{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) - e(\mathsf{P}[m, t]) \cdot \mathsf{x}[j])$;
    end
    $\mathsf{index.b.u} = \texttt{berechnung.index.max}(\mathsf{hilfs.vector1})$;
    $\mathsf{index.b.o} = \texttt{berechnung.index.max}(\mathsf{hilfs.vector2})$;
    $\mathsf{b.unten}[t, m] = \mathsf{x}[\mathsf{index.b.u}]$;
    $\mathsf{b.oben}[t, m] = \mathsf{x}[\mathsf{index.b.o}]$;
    for $j \in \{1, \dots, G\}$ **do**
      $\mathsf{j.min} = \texttt{berechnung.index.jneu}(\mathsf{x}[j])[1]$;
      $\mathsf{j.max} = \texttt{berechnung.index.jneu}(\mathsf{x}[j])[2]$;
      $\mathsf{index.fstar} = \texttt{berechnung.index.fstar}(j, \mathsf{index.b.u}, \mathsf{index.b.o}, \mathsf{j.min}, \mathsf{j.max})$;
      $\mathsf{Wert}[t, j, m] = h_n(\mathsf{P}[m, t], \mathsf{x}[\mathsf{index.fstar}] - \mathsf{x}[j]) + \widetilde{U}_t(\mathsf{x}[\mathsf{index.fstar}], \mathsf{P}[m, t])$;
    end
  end
end

Bestimmung des Erwartungswertes durch Quantisierung

Eine Methode, um die „nested“ Simulation zu umgehen und dennoch eine gute Approximierung des Erwartungswertes durch eine gewichtete Summe zu erhalten, ist die Quantisierungsmethode. Graf u. Luschgy [18] fassen das Ziel dieser Methode wie folgt zusammen:

> „As a mathematical topic quantization for probability distributions concerns the best approximation of a d-dimensional probability distribution P by a discrete probability with a given number l of supporting points or in other words, the best approximation of a d-dimensional random vector X with distribution P by a random vector Y with at most l values in its image.“[18]

Tatsächlich ist es so, dass es für das Fehlermaß, das hier betrachtet wird, immer eine beste Approximation der Form $f(X)$ gibt. [18]

Im Folgenden wird die Quantisierung für $\mathbb{R}^d$-wertige Zufallsvektoren $X \in L^2$ eingeführt. Dabei wird wie in Pagès u. Printems [29] vorgegangen. Für allgemeinere Informationen und allgemeinere Grundlagen wird auf Graf u. Luschgy [18] verwiesen.

Sei X ein quadratisch integrierbarer $\mathbb{R}^d$-wertiger Zufallsvektor auf einem Wahrscheinlichkeitsraum $(\Omega, \mathcal{F}, \mathbb{P})$. Es sei $x := (x_1, \ldots, x_l)$ ein l-Tupel in $\mathbb{R}^d$ und $q : \mathbb{R}^d \to \{x_1, \ldots, x_l\}$ eine messbare Funktion. Der $\{x_1, \ldots, x_l\}$-wertige Zufallsvektor $q(X)$ heißt dann q-Quantisierung von X. Der quadratische q-Quantisierungsfehler ist gegeben durch die Wurzel aus

$$\mathbb{E}|X - q(X)|^2, \tag{6.5}$$

wobei $|\cdot|$ hier, wie auch im Folgenden, die Euklidische Norm in $\mathbb{R}^d$ bezeichnet.
Graf u. Luschgy [18] zeigen, dass unter allen quantisierenden Funktionen q, die sogenannten Voronoi l-Quantizers q_{Vor} den Quantisierungsfehler minimieren, das heißt

$$\mathbb{E}|X - q_{\text{Vor}}(X)|^2 = \min\left\{\mathbb{E}|X - q(X)|^2 : \ q : \mathbb{R}^d \to \{x_1, \ldots x_l\} \text{ messbar } \right\}. \tag{6.6}$$

Die Voronoi l-Quantizer sind definiert durch

$$q_{\text{Vor}}(\xi) = \sum_{i=1}^{l} x_i \mathbf{1}_{C(x_i)}(\xi), \quad \xi \in \mathbb{R}^d, \tag{6.7}$$

wobei $\{C(x_i)\}_{1 \le i \le l}$ eine Borel-Partition von $\mathbb{R}^d$ ist, die für $i = 1, \ldots, l$

$$C(x_i) \subset \{\xi \in \mathbb{R}^d : |\xi - x_i| \le |\xi - x_j|, \ j = 1, \ldots, l\}$$

erfüllt. Eine solche Partition wird auch Voronoi Mosaik des l-Tupels x genannt und $q_{\text{Vor}}(X)$ eine Voronoi l-Quantisierung. Solch eine Voronoi Quantisierung, die abkürzend mit $\widehat{X}$ bezeichnet wird, lässt sich formulieren als

$$\widehat{X}^x := \sum_{i=1}^{l} x_i \mathbf{1}_{C(x_i)}(X). \tag{6.8}$$

Das Ziel der optimalen Quantisierung ist nun, zu vorgegebenem l ein l-Tupel $x^\star = (x_1^\star, \ldots, x_l^\star)$ zu finden, sodass der quadratischen Quantisierungsfehler, also $\mathbb{E}|X - \widehat{X}|^2$, minimal wird.

Die Kerngedanke zur Erwartungswertapproximation ist der, dass man die Verteilung von X durch die von $\widehat{X}$ ersetzt. Sei f eine messbare Funktion, sodass der folgende Erwartungswert existiert. Dann gilt

$$\mathbb{E}f(X) \approx \sum_{i=1}^{l} f(x_i)\mathbb{P}_X(C(x_i)), \tag{6.9}$$

wobei $\mathbb{P}_X$ die Verteilung von X ist. Für Konvergenzbeweise und -raten wird auf oben genannte Literatur verwiesen.

Überträgt man dieses Konzept nun auf die zentrale Berechnung von U in den Algorithmen, so erhält man mittels einer Quantisierung der Verteilung von P_{n+1} gegeben $P_n = p$ und der interpolierenden Funktion $\widetilde{V}$ eine Approximation von U. Es ist

$$U_n(x,p) = \mathbb{E}[V_{n+1}(x, P_{n+1})|P_n = p] \approx \sum_{i=1}^{l} w_i \widetilde{V}_{n+1}(x, \hat{p}_{n+1}^i), \tag{6.10}$$

wobei $\hat{p}_{n+1}^1, \ldots, \hat{p}_{n+1}^l$ die optimalen l-Tupel und $w_i = \mathbb{P}_{P_{n+1}|P_n=p}(C(\hat{p}_{n+1}^i))$ die zugehörigen Gewichte der Voronoi Zellen sind. Es ist zu beachten, dass hier mit Zufallsvariablen gearbeitet wird und daher die Voronoi Zellen Intervallen entsprechen.

Pagès u. Printems [29] beschreiben Algorithmen zur Berechnung der optimalen Quantisierung im Fall der d-dimensionalen Normalverteilung. Auf der Webseite `http://www.quantize.maths-fi.com/` [13] sind unter anderem die damit berechneten l-Tupel und zugehörigen Gewichte der Voronoi Zellen der Standardnormalverteilung für viele verschiedene Werte von l und d verfügbar.

Für die Implementierung soll diese große Datenbank genutzt werden und die optimalen Quantisierungen der Standardnormalverteilung in Quantisierungen der Normalverteilung mit Parametern μ und σ^2 umgeformt werden, die dann für die Berechnung von U in (6.10) benötigt werden. Dazu muss U dahingehend umgeformt werden, dass man eine bedingte Erwartung bezüglich der Log-Preise erhält. Es ist

$$\mathbb{E}[P_{n+1}|P_n = p] = \mathbb{E}[e^{X_{n+1}}|e^{X_n} = p] = \mathbb{E}[e^{X_{n+1}}|X_n = \log p]$$

und damit wird (6.10) zu

$$\begin{aligned} U_n(x,p) &= \mathbb{E}[V_{n+1}(x, P_{n+1})|P_n = p] = \mathbb{E}\left[V_{n+1}\left(x, e^{X_{n+1}}\right)|X_n = \ln p\right] \\ &\approx \sum_{i=1}^{l} w_i \widetilde{V}_{n+1}(x, e^{\hat{x}_{n+1}^i}), \end{aligned} \tag{6.11}$$

wobei $\hat{x}_{n+1}^1, \ldots, \hat{x}_{n+1}^l$ die optimalen l-Tupel und w_i die zugehörigen Gewichte der Voronoi Zellen der entsprechenden Normalverteilung sind, die $\mathcal{L}(X_{n+1}|X_n = \log p)$ entspricht.

Um von einer Quantisierung einer standardnormalverteilten Zufallsvariablen X zu einer Quantisierung einer normalverteilten Zufallsvariablen Y mit μ und σ^2 zu gelangen, nutzt man die Tatsache $Y \stackrel{\mathcal{D}}{=} \sigma X + \mu =: T(X)$ aus und transformiert das gegebene l-Tupel $(x_1, \ldots, x_l)$ mit T. So erhält man das l-Tupel für Y durch $(T(x_1), \ldots, T(x_l))$. Die Voronoi Gewichte bleiben unter dieser affinen Transformation gleich. Da

$$\mathbb{E}|T(X) - T(\hat{X})|^2 = \mathbb{E}|\sigma X + \mu - \sigma\hat{X} - \mu|^2 = \sigma^2\mathbb{E}|X - \hat{X}|^2$$

ist, bleibt in diesem Fall auch die Optimalitätseigenschaft erhalten.

Für die Durchführung von Algorithmus 9 wird die Anzahl der Quantisierungspunkte l festgelegt und entsprechend die Datei von [13] heruntergeladen, welche die Punkte und deren Voronoi-Gewichte der Standardnormalverteilung enthält. Die Dateien sind für l zwischen 1 und 5999 erhältlich. Die Quantisierungstupel werden im Vektor `Quanten` und die Gewichte in einem Vektor `weight` gespeichert. Diese Vektoren werden für den Algorithmus übergeben. Mit Hilfe dieser Quantisierung wird dann in jedem Zeitschritt die Quantisierung für die Normalverteilung mit Parameter μ_t und σ_t^2 gemäß Satz 5.1.6 berechnet. Es beschreiben $\hat{x}_t^1, \ldots, x_t^l$ die entsprechenden l-Tupel zur Quantisierung von $X_t|X_{t-1} = \log p$. Die Berechnung von U_t erfolgt dann mit (6.11).

Veränderungen im Algorithmus im Vergleich zu den anderen Berechnungen des Erwartungswert sind grün markiert.

Algorithmus 9 : Monte-Carlo Algorithmus mit Interpolationsansatz und Quantisierung

Data : Parameter des Gasspeichers, Speichergitter x der Länge G, Preispfadmatrizen P, l Quantisierungspunkte Quanten der Standardnormalverteilung und Gewichte der Voronoi-Zellen weight

Result : $3d$-Array Wert des Speicherwerts, Matrizen b.unten und b.oben der Strategie-Schranken.

for $j \in \{1, \ldots, G\}$ **do**
  for $m \in \{1, \ldots, M\}$ **do**
    $\mathsf{Wert}[N, j, m] = h_N(\mathsf{x}[j], \mathsf{P}[m, N])$;
  end
end
for $t \in \{N-1, \ldots, 0\}$ **do**
  for $m \in \{1, \ldots, M\}$ **do**
    for $j \in \{1, \ldots, G\}$ **do**
      $\widetilde{V}_{t+1}(\mathsf{x}[j], \cdot) = \texttt{interpolation}(\mathsf{P}[\ , t+1], \mathsf{Wert}[t+1, j, \ ])$;
      if $t! = 0$ **then**
        $\mu_t = \mathbb{E}(X_{t+1}) + \rho\sqrt{\frac{\mathbb{V}(X_{t+1})}{\mathbb{V}(X_t)}}(\log \mathsf{P}[m, t] - \mathbb{E}(X_t))$;
        $\rho = \frac{\mathrm{Cov}(X_t, X_{t+1})}{\sqrt{\mathbb{V}(X_{t+1})\mathbb{V}(X_t)}}$;
        $\sigma_t^2 = \mathbb{V}(X_t)(1 - \rho^2)$;
      else
        $\mu_t = \mathbb{E}(X_{t+1})$;
        $\sigma_t^2 = \mathbb{V}(X_{t+1})$;
      end
      for $i \in \{1, \ldots, l\}$ **do**
        $\hat{x}_{t+1}^i = \sigma_t \cdot \mathsf{Quanten}[i] + \mu_t$;
      end
      $\widetilde{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) = \sum_{i=1}^{l} \mathsf{weight}[i] \cdot \widetilde{V}_{t+1}(\mathsf{x}[j], e^{\hat{x}_{t+1}^i})$;
      $\mathsf{hilfs.vector1} = \widetilde{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) - k(\mathsf{P}[m, t]) \cdot \mathsf{x}[j])$;
      $\mathsf{hilfs.vector2} = \widetilde{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) - e(\mathsf{P}[m, t]) \cdot \mathsf{x}[j])$;
    end
    $\mathsf{index.b.u} = \texttt{berechnung.index.max}(\mathsf{hilfs.vector1})$;
    $\mathsf{index.b.o} = \texttt{berechnung.index.max}(\mathsf{hilfs.vector2})$;
    $\mathsf{b.unten}[t, m] = \mathsf{x}[\mathsf{index.b.u}]$;
    $\mathsf{b.oben}[t, m] = \mathsf{x}[\mathsf{index.b.o}]$;
    for $j \in \{1, \ldots, G\}$ **do**
      $\mathsf{j.min} = \texttt{berechnung.index.jneu}(\mathsf{x}[j])[1]$;
      $\mathsf{j.max} = \texttt{berechnung.index.jneu}(\mathsf{x}[j])[2]$;
      $\mathsf{index.fstar} = \texttt{berechnung.index.fstar}(j, \mathsf{index.b.u}, \mathsf{index.b.o}, \mathsf{j.min}, \mathsf{j.max})$;
      $\mathsf{Wert}[t, j, m] = h_n(\mathsf{P}[m, t], \mathsf{x}[\mathsf{index.fstar}] - \mathsf{x}[j]) + \widetilde{U}_t(\mathsf{x}[\mathsf{index.fstar}], \mathsf{P}[m, t])$;
    end
  end
end

6.3.2. Least-Square-Ansatz

Der Least-Square Ansatz geht auf Longstaff u. Schwartz [25] zurück, die diesen zur Bewertung von Amerikanischen Optionen eingeführt haben. Konvergenzresultate für diesen Fall sind in Clément u.a. [12] zu finden. Boogert u. de Jong [5] griffen den Grundgedanken auf, um ihn für einen Algorithmus zur Gasspeicherbewertung zu verwenden. Im Folgenden wird dieser Ansatz erläutert und auf den Algorithmus unter Verwendung der Struktur der optimalen Politik angewendet.

Als bedingter Erwartungswert ist der „Continuation Value“ $U_n(x,\cdot)$ für jedes x eine Funktion des aktuellen Preises p. Die Grundidee ist diesen durch eine Linearkombination aus Basisfunktion $\varphi_1,\ldots,\varphi_l$ zu approximieren. Das bedeutet

$$U_n(x,p) \approx \sum_i^l \beta_i \varphi(p),$$

wobei die β_i, $i=1,\ldots,l$ möglichst optimal zu wählen sind. Die Koeffizienten β_i werden mit Hilfe der Methode der kleinsten Quadrate bestimmt. Es bezeichne p_n^m den Preis zur Zeit n in der m-ten Pfadrealisierung. Als Ausgangspunkt, d.h. als Realisierungen des Continuation Value auf dem Gitter $p_n^1,\ldots p_n^M$, dienen die im vorangegangen Schritt des Algorithmus berechneten Werte $V_{n+1}(x,p_{n+1}^1),\ldots,V_{n+1}(x,p_{n+1}^M)$. Die Koeffizienten β_i werden so bestimmt, dass die Summe der Fehlerquadrate minimiert wird. Auf Stufe n im Zustand x wird demnach

$$\left\| \begin{pmatrix} V_{n+1}(x,p_{n+1}^1) \\ \vdots \\ V_{n+1}(x,p_{n+1}^M) \end{pmatrix} - \begin{pmatrix} \varphi_1(p_n^1) & \cdots & \varphi_l(p_n^1) \\ \vdots & & \vdots \\ \varphi_1(p_n^M) & \cdots & \varphi_l(p_n^M) \end{pmatrix} \cdot \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_l \end{pmatrix} \right\|^2 \tag{6.12}$$

minimiert, wobei $\|\cdot\|$ die Euklid-Norm bezeichne. Man erhält so die Koeffizienten β_i, $i=1,\ldots,l$ und im Algorithmus wird dann mit dem Schätzer

$$\hat{U}_n(x,p_n^m) = \sum_i^l \beta_i \varphi(p_n^m) \tag{6.13}$$

gerechnet.

Um Algorithmus 10 zu beschreiben, seien die Koeffizienten der kleinsten Quadrate Methode im Vektor $\beta = (\beta_1,\ldots,\beta_l)^T$ zusammengefasst. Die Funktion `berechnung.beta.ls` führt die beschriebene Minimierung aus und gibt die Koeffizienten zurück. Eingabeparameter dieser Funktion sind zwei reelle Vektoren, welche die Realisierungen der Variablen p und der abhängigen Variablen $U(x,p)$ angeben, sowie ein Vektor, in dem die Basisfunktionen gespeichert sind. Da in den in Kapitel 7 genutzten Programmiersprachen, das heißt R und C, bereits Routinen zur Verwendung der kleinsten Quadrate Methode zur Verfügung stehen, wird an dieser Stelle auf die nähere Beschreibung der Funktion `berechnung.beta.ls` verzichtet. Die zur Verfügung stehenden Routinen verwenden eine QR-Zerlegung zur Lösung des Ausgleichsproblem. In R ist das die `lm` Routine, die auch im Paket `gputools` als `gpuLM` als GPU-Version vorhanden ist. In C kann zum Beispiel

die Funktion `dgels` des `LAPACK`-Pakets genutzt werden. Für nähere Beschreibung der Routinen wird auf die jeweiligen Dokumentationen, das heißt Anderson u. a. [1] für LAPACK und R Core Team [30] für R, verwiesen.

Zu beachten ist außerdem, wie zum Zeitpunkt $t = 0$ sich die kleinste Quadrate Methode verhält. Da jeder Pfad des Preises mit dem gleichen vorgegebenen p_0 beginnt, ist diese Methode nicht mehr anwendbar. Stattdessen, wird als Continuation Value der Mittelwert aus den Speicherwerten zum Zeitpunkt 1 über die verschiedenen Pfade gebildet.

Es werden wieder die Funktionen aus Bemerkungen 6.1.2 und 6.1.3 verwendet. Mit $\texttt{P}[,t]$ wird die $(t+1)$-te Spalte der Matrix `P` bezeichnet, also alle simulierten Realisierungen $p_t^1, \ldots, p_t^M$ des Preises zur Zeit t. Mit $\texttt{Wert}[t+1, j,]$ wird der Vektor bezeichnet, der $V_{t+1}(\texttt{x}[j], p_{t+1}^1), \ldots, V_{t+1}(\texttt{x}[j], p_{t+1}^M)$ beinhaltet.

Die Zeilen, die sich im Wesentlichen von den anderen Algorithmen unterscheiden, sind blau hervorgehoben.

Algorithmus 10 : Monte-Carlo Algorithmus mit dem Least-Square-Ansatz

Data : Parameter des Gasspeichers, Speichergitter x der Länge G, Preispfadmatrizen P, Basisfunktionen $\varphi_1, \ldots, \varphi_l$

Result : $3d$-Array Wert des Speicherwerts, Matrizen b.unten und b.oben der Strategie-Schranken.

for $j \in \{1, \ldots, G\}$ **do**
  for $m \in \{1, \ldots, M\}$ **do**
    $\mathsf{Wert}[N, j, m] = h_N(\mathsf{x}[j], \mathsf{P}[m, N])$;
  end
end
for $t \in \{N-1, \ldots, 0\}$ **do**
  for $m \in \{1, \ldots, M\}$ **do**
    for $j \in \{1, \ldots, G\}$ **do**
      if $t! = 0$ **then**
        $\beta = \mathtt{berechnung.beta.ls}(\mathsf{P}[, t], \mathsf{Wert}[t+1, j,], \varphi)$;
        $\hat{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) = \sum_{i=1}^{l} \beta_i \varphi_i(\mathsf{P}[m, t])$;
      else
        $\hat{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) = \frac{1}{M} \sum_{m=1}^{M} \mathsf{Wert}[t+1, j, m]$;
      end
      $\mathsf{hilfs.vector1} = \hat{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) - k(\mathsf{P}[m, t]) \cdot \mathsf{x}[j])$;
      $\mathsf{hilfs.vector2} = \hat{U}_t(\mathsf{x}[j], \mathsf{P}[m, t]) - e(\mathsf{P}[m, t]) \cdot \mathsf{x}[j])$;
    end
    $\mathsf{index.b.u} = \mathtt{berechnung.index.max}(\mathsf{hilfs.vector1})$;
    $\mathsf{index.b.o} = \mathtt{berechnung.index.max}(\mathsf{hilfs.vector2})$;
    $\mathsf{b.unten}[t, m] = \mathsf{x}[\mathsf{index.b.u}]$;
    $\mathsf{b.oben}[t, m] = \mathsf{x}[\mathsf{index.b.o}]$;
    for $j \in \{1, \ldots, G\}$ **do**
      $\mathsf{j.min} = \mathtt{berechnung.index.jneu}(\mathsf{x}[j])[1]$;
      $\mathsf{j.max} = \mathtt{berechnung.index.jneu}(\mathsf{x}[j])[2]$;
      $\mathsf{index.fstar} = \mathtt{berechnung.index.fstar}(j, \mathsf{index.b.u}, \mathsf{index.b.o}, \mathsf{j.min}, \mathsf{j.max})$;
      $\mathsf{Wert}[t, j, m] = h_n(\mathsf{P}[m, t], \mathsf{x}[\mathsf{index.fstar}] - \mathsf{x}[j]) + \hat{U}_t(\mathsf{x}[\mathsf{index.fstar}], \mathsf{P}[m, t])$;
    end
  end
end

6.4. Monte-Carlo Algorithmen zur Schätzung oberer Schranken des Gasspeicherwertes

Wenn sich die Frage stellt, ob die bisher beschriebenen Algorithmen tendenziell eine untere oder obere Schranke des Gasspeicherwertes liefern, so muss differenziert werden. Einige Arbeiten gehen von der These aus, dass sich durch den Least-Square-Ansatz eine unter Schranke finden lässt. Allerdings basiert deren Argumentation darauf, dass letztendlich mit einer suboptimalen Strategie gearbeitet wird, während in dieser Arbeit die optimale Strategie verwendet wird. Einige andere Arbeiten wiederum behaupten, dass man aufgrund der Monte-Carlo Simulation letztendlich den Wert eher überschätzt.

Im Folgenden wird eine Methode vorgestellt, die zu einer oberen Schranke des erwarteten Gasspeicherwertes führt. Diese basiert auf der Dualitätstheorie für Markovsche Entscheidungsprozesse, die in der Arbeit von Brown u. a. [7] vorgestellt wird.

6.4.1. Auszug aus der Dualitätstheorie für Markovsche Entscheidungsprozesse

Der Artikel „Information Relaxation and Duality in Stochastic Dynamic Programs“[7] beschreibt einen Dualitätsansatz für stochastische dynamische Programme, der eine obere Schranke des erwarteten optimalen Wertes liefert. In einem ersten Schritt wird die Restriktion gelockert, dass man zum Zeitpunkt n nur aufgrund der Informationen bis zur Zeit n seine Entscheidung für den nächsten Zeitpunkt trifft. Gleichzeitig wird aber in einem zweiten Schritt eine Bestrafung („Penalization“) eingeführt, für den Fall, dass man genau solche Entscheidungen trifft, die den gelockerten Restriktionen entsprechen. Da diese Lockerung („Relaxation“) der Retriktionen einen Informationsgewinn bedeutet, erhält man eine obere Schranke des Optimierungsproblems.

In der Gasspeicherbewertung wird diese Relaxation letztendlich bedeuten, dass die Preise - das einzige zufällige „underlying“ - nicht nur bis zum aktuellen Zeitpunkt n, sondern darüberhinaus bekannt sind. Das wird durch eine Monte-Carlo Simulation erreicht, wie sie schon in Kapitel 5 beschrieben wurde.

Bevor genauer auf die entsprechenden Algorithmen eingegangen wird, sollen die wichtigsten Resultate von Brown u. a. [7] zusammengefasst und erklärt werden. Die Bezeichnungen lehnen sich an den Artikel von Brown u. a. [7] an.

Es sei $(\Omega, \mathcal{F}, \mathbb{P})$ der zugrundeliegende Wahrscheinlichkeitsraum des Stochastischen Dynamischen Programms. Die Entwicklung der Information über die Zeit sei beschrieben in der Filtration $\mathbb{F} = (\mathcal{F}_0, \ldots, \mathcal{F}_N)$, wobei $\mathcal{F}_n \subseteq \mathcal{F}_{n+1} \subseteq \mathcal{F}$ gelte für alle $n < N$ und $\mathcal{F}_n$ beschreibe die Information, die zur Zeit n vorliegt. Es bezeichne $\mathcal{A}$ die Menge aller zulässigen Politiken. Wie bisher wird eine Politik mit π und der Gewinn auf Stufe n mit h_n bezeichnet. Der Gesamtgewinn sei durch die Funktion H beschrieben.

Im ursprünglichen Optimierungsproblem wird vorausgesetzt, dass zulässige Politiken bezüglich der Filtration $\mathbb{F}$ adaptiert sind, das heißt die Teilpolitik der ersten $n+1$ Entscheidungen $(f_0, \ldots, f_n)$ soll messbar bezüglich $\mathcal{F}_n$ sein. Sei $\mathcal{A}_\mathbb{F}$ die Menge all dieser Entscheidungen, dann ist das Primalprogramm (PP) gegeben durch

$$\sup_{\pi \in \mathcal{A}_\mathbb{F}} \mathbb{E}[H(\pi)]. \tag{6.14}$$

Es wird darauf hingewiesen, dass π von der Realisierung des stochastischen „underlying" abhängt. Im Fall des Gasspeicherproblems ist dies der Preis.

Um das Dualprogramm zu formulieren, wird zunächst die Relaxation und die Penalization definiert.
Eine Filtration $\mathbb{G} = (\mathcal{G}_0, \ldots, \mathcal{G}_N)$ ist eine Relaxation der Filtration $\mathbb{F}$, falls $\mathcal{F}_n \subseteq \mathcal{G}_n \subseteq \mathcal{F}$ für alle n gilt; kurz als $\mathbb{F} \subseteq \mathbb{G}$ bezeichnet. Das bedeutet, dass unter Verwendung der Filtration $\mathbb{G}$ mehr Information vorliegt, als unter Verwendung von $\mathbb{F}$. Es bezeichne entsprechend $\mathcal{A}_\mathbb{G}$ die Menge aller $\mathbb{G}$-adaptierten Politiken. Speziell wird die Relaxation $\mathbb{I} = (\mathcal{I}_0, \ldots, \mathcal{I}_N)$ mit $\mathcal{I}_n = \mathcal{F}$ für alle n die Relaxation mit voller Information genannt (kurz: volle Information). Wenn die Filtration gelockert wird, erweitert man die Menge der zulässigen Strategien, das heißt es gilt $\mathcal{A}_\mathbb{F} \subseteq \mathcal{A}_\mathbb{G} \subseteq \mathcal{A}_\mathbb{I} = \mathcal{A}$.
In der Menge der Penalties $\mathcal{Z}$ seien Funktionen z enthalten, die wie H von der gewählten Politk abhängen. Die Menge $\mathcal{Z}_\mathbb{F}$ der dual zulässigen Penalties sei gegeben durch

$$\mathcal{Z}_\mathbb{F} := \{z \in \mathcal{Z} : \mathbb{E}[z(\pi_F)] \leq 0 \text{ für alle } \pi_F \in \mathcal{A}_\mathbb{F}\},$$

das heißt für Politiken aus $\mathcal{A}_\mathbb{F}$ soll die erwartete Bestrafung nicht-positiv sein.

Hiermit lässt sich eine schwache Dualitätsaussage formulieren.

Satz 6.4.1
Sind π_F und z primal und dual zulässig, das heißt $\pi_F \in \mathcal{A}_\mathbb{F}$ und $z \in \mathcal{Z}_\mathbb{F}$, und ist $\mathbb{G}$ eine Relaxation von $\mathbb{F}$ so gilt

$$\mathbb{E}[H(\pi_F)] \leq \sup_{\pi_G \in \mathcal{A}_\mathbb{G}} \mathbb{E}[H(\pi_G) - z(\pi_G)]. \tag{6.15}$$

Beweis
Siehe Lemma 2.1 in Brown u. a. [7]. ∎

Aus diesem Satz erhält man, dass jede Relaxation mit jeder dual zulässigen Penalty eine obere Schranke des Primalprogramms liefert.

Bemerkung 6.4.2
Betrachtet man den Fall voller Information, so besagt Satz 6.4.1, da $\mathcal{A}_{\mathbb{I}} = \mathcal{A}$ ist, dass für jedes $\pi_F \in \mathcal{A}_{\mathbb{F}}$ und $z \in \mathcal{Z}_{\mathbb{F}}$

$$\mathbb{E}[H(\pi_F)] \leq \sup_{\pi \in \mathcal{A}} \mathbb{E}[H(\pi) - z(\pi)] = \mathbb{E}\left[\sup_{\tilde{a} \in \tilde{A}} \{H(\tilde{a}) - z(\tilde{a})\}\right]$$

gilt. Hierbei bezeichnet $\tilde{a}$ einen Vektor aus zulässigen Aktionen und $\tilde{A}$ die Menge dieser Vektoren.

Diese Beobachtung liefert eine Methode, um mit Monte-Carlo Simulationen einen Schätzer für eine obere Schranke des Problems zu erhalten:
Man simuliert Realisierungen des stochastischen „underlying“ und löst das innere deterministische Optimierungsproblem. Anschließend wird der Erwartungswert zum Beispiel durch eine Mittelwertbildung geschätzt.

Die in Bemerkung 6.4.2 beschriebene Monte-Carlo Methode wird in den folgenden Teilkapiteln als Grundlage verwendet.

Auch wenn die Grundlage für die Monte-Carlo Algorithmen zur Schätzung oberer Schranken bereits gelegt ist, sollte man den Satz, der starke Dualität formuliert, erwähnen.

Satz 6.4.3
Sei $\mathbb{G}$ eine Relaxation von $\mathbb{F}$. Dann gilt

$$\sup_{\pi_F \in \mathcal{A}_{\mathbb{F}}} \mathbb{E}[H(\pi_F)] = \inf_{z \in \mathcal{Z}_{\mathbb{F}}} \left\{ \sup_{\pi_G \in \mathcal{A}_{\mathbb{G}}} \mathbb{E}[H(\pi_G) - z(\pi_G)] \right\}. \tag{6.16}$$

Beweis
Siehe Theorem 2.1 in Brown u. a. [7]. ∎

Bemerkung 6.4.4
Brown u. a. [7] zeigen außerdem in Proposition 2.1, dass es sinnvoll ist, Strukturen, die man bereits für die optimale Politk kennt, für die Berechnung der oberen Schranke zu übernehmen.
Das heißt, man kann im Gasspeicherproblem zur Schätzung der oberen Schranke, die in Satz 4.3.4 gefundene Struktur der Strategie verwenden.

6.4.2. Algorithmus mittels Information Relaxation

Wie in Bemerkung 6.4.2 beschrieben, bietet die Dualitätstheorie einen Ausgangspunkt für eine Schätzung der oberen Schranken des Gasspeicherwertes mittels einer Monte Carlo Simulation. In den folgenden Abschnitten, in denen zwei Algorithmen vorgestellt werden, wird davon ausgegangen, dass M mittels Algorithmus 1 simulierte Pfade des Preisprozesses vorliegen. Wie schon in Kapitel 6.3 seien diese Pfade in der Matrix `P` gespeichert. Beide Algorithmen basieren auf einer Relaxation mit voller Information.

Algorithmus mittels Information Relaxation ohne Penalization

Die erste Möglichkeit, die beschrieben wird, ist eine intuitive Vorgehensweise. Dadurch, dass hier die Penalization z auf Null gesetzt wird, besteht das Vorgehen des Algorithmus darin, auf jedem einzelnen erzeugten Pfad das Optimierungsproblem zu lösen und am Ende, als Schätzer des Erwartungswertes, den Mittelwert über alle Pfade zu bilden.

Damit ist auch der erste große Unterschied zu den Algorithmen aus Kapitel 6.3 ersichtlich. Während dort der berechnete Wert zum Zeitpunkt 0 in Speicherstand $\mathtt{x}[j]$ für jeden Pfad gleich war, ist dieser hier verschieden. Dies liegt daran, dass bei den bisherigen Kapiteln in jedem Teilschritt der bedingte Erwartungswert geschätzt wurde, hier wird der Erwartungswert erst am Schluss geschätzt.

Da von voller Information und keiner Penalization ausgegangen wird, nimmt man also an, dass

$$U_n(x, p_n) = \mathbb{E}[V_{n+1}(x, P_{n+1})|\mathcal{I}_n] = V_{n+1}(x, p_{n+1})$$

gilt. Für jeden realisierten Pfad heißt das, dass für alle $m \in \{1, 2, \ldots, M\}$

$$U_n(x, p_n^m) = V_{n+1}(x, p_{n+1}^m) \tag{6.17}$$

gilt.

Algorithmus 11 beschreibt dieses Vorgehen. Bezeichnungen aus den Bemerkungen 6.1.2 und 6.1.3 werden, wie gehabt, verwendet. Der große Unterschied zu den bisherigen Algorithmen, die Mittelwertbildung zur Schätzung des Erwartungswertes am Schluss, ist blau hervorgehoben. Die Berechnung von U_t erfolgt für jeden Pfad mittels (6.17) und ist grün hervorgehoben, um den Unterschied zum Algorithmus mit Penalization zu verdeutlichen.

Algorithmus 11 : Monte-Carlo Algorihtmus zur Schätzung einer oberen Schranke ohne Penalization

Data : Parameter des Gasspeichers, Speichergitter x der Länge G, Preispfadmatrizen P
Result : Schätzer der oberen Schranke Schätzer.obere.Schranke für verschiedene Anfangsvolumina

for $j \in \{1, \ldots, G\}$ **do**
  for $m \in \{1, \ldots, M\}$ **do**
    $\mathsf{Wert}[N, j, m] = h_N(\mathsf{x}[j], \mathsf{P}[m, N])$;
  end
end
for $t \in \{N-1, \ldots, 0\}$ **do**
  for $m \in \{1, \ldots, M\}$ **do**
    for $j \in \{1, \ldots, G\}$ **do**
      $U_t(\mathsf{x}[j], \mathsf{P}[m, t]) = \mathsf{Wert}[t+1, \mathsf{x}[j], m];$
      $\mathsf{hilfs.vector1} = U_t(\mathsf{x}[j], \mathsf{P}[m, t]) - k(\mathsf{P}[m, t]) \cdot \mathsf{x}[j]);$
      $\mathsf{hilfs.vector2} = U_t(\mathsf{x}[j], \mathsf{P}[m, t]) - e(\mathsf{P}[m, t]) \cdot \mathsf{x}[j]);$
    end
    $\mathsf{index.b.u} = \texttt{berechnung.index.max}(\mathsf{hilfs.vector1});$
    $\mathsf{index.b.o} = \texttt{berechnung.index.max}(\mathsf{hilfs.vector2});$
    $\mathsf{b.unten}[t, m] = \mathsf{x}[\mathsf{index.b.u}];$
    $\mathsf{b.oben}[t, m] = \mathsf{x}[\mathsf{index.b.o}];$
    for $j \in \{1, \ldots, G\}$ **do**
      $\mathsf{j.min} = \texttt{berechnung.index.jneu}(\mathsf{x}[j])[1];$
      $\mathsf{j.max} = \texttt{berechnung.index.jneu}(\mathsf{x}[j])[2];$
      $\mathsf{index.fstar} = \texttt{berechnung.index.fstar}(j, \mathsf{index.b.u}, \mathsf{index.b.o}, \mathsf{j.min}, \mathsf{j.max})$;
      $\mathsf{Wert}[t, j, m] = h_n(\mathsf{P}[m, t], \mathsf{x}[\mathsf{index.fstar}] - \mathsf{x}[j]) + U_t(\mathsf{x}[\mathsf{index.fstar}], \mathsf{P}[m, t])$;
    end
  end
end
for $j \in \{1, \ldots, G\}$ **do**
  $\mathsf{Schätzer.obere.Schranke}[j] = \frac{1}{M} \sum_{m=1}^{M} \mathsf{Wert}[0, j, m];$
end

Dieser Algorithmus enthält ganz eindeutig großes Parallelisierungspotential, denn die Größe `Wert` wird für jeden Pfad einzeln unabhängig von den anderen Pfaden bestimmt. Diese Berechnung könnte demnach parallel für alle Pfade durchgeführt werden.

Genauso hat auch der folgende Algorithmus, der eine Penalization einschließt, Parallelisierungspotential.

Algorithmen mittels Information Relaxation mit Penalization

Um eine Methode mit Penalization vorstellen zu können, betrachtet man zunächst den zur Zeit n entstehenden „Bellman-Schritt“ mit Information Relaxation. Es ist

$$\begin{aligned} V_n^{\mathbb{I}}(x, p_n) &= \sup_{a \in D_n(x)} \left\{ h_n(p_n, a) - z_n(x+a, p_n) + \mathbb{E}\left[V_{n+1}^{\mathbb{I}}(x+a, P_{n+1}) | \mathcal{I}_n \right] \right\} \\ &= \sup_{a \in D_n(x)} \left\{ h_n(p_n, a) - z_n(x+a, p_n) + V_{n+1}^{\mathbb{I}}(x+a, p_{n+1}) \right\}. \end{aligned}$$

In Brown u. a. [7] werden einige Methoden zur Konstruktion von Penalties z beschrieben. Eine dieser Methoden, die auf einer Methode von Haugh u. Kogan [20] basiert, wird im Folgenden vorgestellt. Für weitere Möglichkeiten gute Penalties zu konstruieren wird auf Brown u. a. [7] verwiesen.

Ausgehend von einer approximierenden Wertfunktion $\hat{V}_t$ wird die Penalty Funktion

$$\begin{aligned} z_n(x+a, p_n) &= \mathbb{E}[\hat{V}_{n+1}(x+a, P_{n+1}) | \mathcal{I}_n] - \mathbb{E}[\hat{V}_{n+1}(x+a, P_{n+1}) | \mathcal{F}_n] \\ &= \hat{V}_{n+1}(x+a, p_{n+1}) - \mathbb{E}[\hat{V}_{n+1}(x+a, P_{n+1}) | P_n = p_n] \end{aligned}$$

gesetzt.

Für den Continuation Value U bedeutet das für jeden Pfad, also $m = 1, \ldots, M$

$$U_n(x, p_n^m) = V_{n+1}(x, p_{n+1}^m) - \hat{V}_{n+1}(x, p_{n+1}^m) + \mathbb{E}[\hat{V}_{n+1}(x, P_{n+1}) | P_n = p_n^m]. \tag{6.18}$$

Somit muss zunächst eine approximierende Wertfunktion vorliegen. Diese könnte zum Beispiel aus einem der Algorithmen aus Kapitel 6.3 entstehen, indem man die entsprechenden Ergebnisse $\hat{V}_{n+1}(x, \cdot)$, die auf einem Gitter vorliegen - vergleiche die entsprechenden Algorithmen - interpoliert. Hierzu wird lineare Interpolation genutzt, wie unter anderem in Kapitel 6.3.1. Nutzt man für die Berechnung von $\hat{V}$ auf dem Gitter die selbe Monte Carlo Simulation der Preispfade, so muss zumindest für den Teil $\hat{V}_{n+1}(x, p_{n+1}^m)$ in (6.18) nicht die approximierende Funktion genutzt werden. Für die Berechnung von $\mathbb{E}[\hat{V}_{n+1}(x, P_{n+1}) | P_n = p_n^m]$ ist allerdings die approximierende Funktion von Bedeutung.

Im Grunde hat man nun die gleichen Alternativen, wie in Kapitel 6.3.1, um den gesuchten Erwartungswert zu berechnen, beziehungsweise zu schätzen: Integration, Simulation, Quantisierung.

Haugh u. Kogan [20] benutzen für die Bestimmung des Erwartungswertes eine „nested“ Simulationsmethode, die auf neuronalen Netzen basiert. Wie Brown u. a. [7] zeigen, führt diese Methode zur Schätzung einer oberen Schranke, solange der Schätzer des Erwartungswertes erwartungstreu ist.

Zur Beschreibung des Algorithmus wird in dieser Arbeit exemplarisch die Quantisierungsmethode zur Approximation der Erwartungswertes zugrunde gelegt, vergleiche hierzu den entsprechenden Abschnitt in Kapitel 6.3.1. Der Erwartungswert wird somit in jedem Schritt mittels quantisierender l-Tupel $(\hat{x}_{n+1}^1, \ldots, \hat{x}_{n+1}^l)$ und deren Voronoi-Gewichte $w_1, \ldots, w_l$ der jeweiligen Verteilung der Logpreise X_{n+1} gegeben $X_n = \log p$ geschätzt. Ähnlich wie in Kapitel 6.3.1 ist

$$\mathbb{E}[\hat{V}_{n+1}(x, P_{n+1})|P_n = p] \approx \sum_{i=1}^{l} w_i \cdot \hat{V}_{n+1}(x, e^{\hat{x}_{n+1}^i})$$

und damit wird U in jedem Zeitschritt auf jedem Pfad durch

$$U_n(x, p_n^m) = V_{n+1}(x, p_{n+1}^m) - \hat{V}_{n+1}(x, p_{n+1}^m) + \sum_{i=1}^{l} w_i \hat{V}_{n+1}(x, e^{\hat{x}_{n+1}^i}) \tag{6.19}$$

bestimmt.

In Algorithmus 12 wird, wie im Abschnitt davor, ein Schätzer `Schätzer.obere.Schranke` für eine obere Schranke des Gasspeicherwertes für verschiedene Anfangsvolumina berechnet. Wie zuvor wird demnach am Schluss über die auf jedem Pfad berechneten Werte gemittelt (blau markiert).

Es wird davon ausgegangen, dass die approximierende Wertfunktion $\hat{V}$ für jeden Zeitpunkt $t = 1, \ldots, N$ und jeden Speicherzustand im Gitter `x` in Abhängigkeit des Preises vorliegt. Außerdem sei, wie in Algorithmus 9 die Quantisierung der Standardnormalverteilung durch die l-Tupel im Vektor `Quanten` und ihre Gewichte im Vektor `weight` gegeben.
Man muss um U mit Hilfe von (6.19) zu berechnen, zunächst μ_t und σ_t^2 nach Satz 5.1.6 bestimmen und die Quantisierung wie in Algorithmus 9 bereits gesehen, anpassen. Die entsprechenden Zeilen sind grün markiert.

Erkennbar ist, das dieser Algorithmus aufwändiger ist als der zuvor beschriebenen Algorithmus 11. Die Frage ist, ob die bessere, das heißt kleinere, obere Schranke den zusätzliche Aufwand rechtfertigt, siehe dazu Kapitel 7.4.

Algorithmus 12 : Monte-Carlo Algorithmus zur Schätzung einer oberen Schranke mit Penalization

Data : Parameter des Gasspeichers, Speichergitter x der Länge G, Preispfadmatrizen P, approximierende Wertefunktion $\hat{V}$, l Quantisierungspunkte Quanten der Standardnormalverteilung und Gewichte der Voronoi-Zellen weight

Result : Schätzer der oberen Schranke Schätzer.obere.Schranke für verschiedene Anfangsvolumina

for $j \in \{1, \dots, G\}$ **do**
  for $m \in \{1, \dots, M\}$ **do**
    $\mathsf{Wert}[N, j, m] = h_N(\mathsf{x}[j], \mathsf{P}[m, N])$;
  end
end
for $t \in \{N-1, \dots, 0\}$ **do**
  for $m \in \{1, \dots, M\}$ **do**
    for $j \in \{1, \dots, G\}$ **do**
      if $t! = 0$ **then**
        $\mu_t = \mathbb{E}(X_{t+1}) + \rho\sqrt{\frac{\mathbb{V}(X_{t+1})}{\mathbb{V}(X_t)}}(\log \mathsf{P}[m,t] - \mathbb{E}(X_t))$;
        $\rho = \frac{\mathrm{Cov}(X_t, X_{t+1})}{\sqrt{\mathbb{V}(X_{t+1})\mathbb{V}(X_t)}}$;
        $\sigma_t^2 = \mathbb{V}(X_t)(1-\rho^2)$;
      else
        $\mu_t = \mathbb{E}(X_{t+1})$;
        $\sigma_t^2 = \mathbb{V}(X_{t+1})$;
      end
      for $i \in \{1, \dots, l\}$ **do**
        $\hat{x}_{t+1}^i = \sigma_t \cdot \mathsf{Quanten}[i] + \mu_t$;
      end
      $U_t(\mathsf{x}[j], \mathsf{P}[m,t]) = \mathsf{Wert}[t+1, \mathsf{x}[j], m] - \hat{V}_{t+1}(\mathsf{x}[j], \mathsf{P}[m, t+1]) + \sum_{i=1}^{l} \mathsf{weight}[i] \cdot \hat{V}_{t+1}(\mathsf{x}[j], e^{\hat{x}_{t+1}^i})$;
      $\mathsf{hilfs.vector1} = U_t(\mathsf{x}[j], \mathsf{P}[m,t]) - k(\mathsf{P}[m,t]) \cdot \mathsf{x}[j])$;
      $\mathsf{hilfs.vector2} = U_t(\mathsf{x}[j], \mathsf{P}[m,t]) - e(\mathsf{P}[m,t]) \cdot \mathsf{x}[j])$;
    end
    $\mathsf{index.b.u} = \mathtt{berechnung.index.max}(\mathsf{hilfs.vector1})$;
    $\mathsf{index.b.o} = \mathtt{berechnung.index.max}(\mathsf{hilfs.vector2})$;
    $\mathsf{b.unten}[t, m] = \mathsf{x}[\mathsf{index.b.u}]$;
    $\mathsf{b.oben}[t, m] = \mathsf{x}[\mathsf{index.b.o}]$;
    for $j \in \{1, \dots, G\}$ **do**
      $\mathsf{j.min} = \mathtt{berechnung.index.jneu}(\mathsf{x}[j])[1]$;
      $\mathsf{j.max} = \mathtt{berechnung.index.jneu}(\mathsf{x}[j])[2]$;
      $\mathsf{index.fstar} = \mathtt{berechnung.index.fstar}(j, \mathsf{index.b.u}, \mathsf{index.b.o}, \mathsf{j.min}, \mathsf{j.max})$;
      $\mathsf{Wert}[t, j, m] = h_n(\mathsf{P}[m,t], \mathsf{x}[\mathsf{index.fstar}] - \mathsf{x}[j]) + U_t(\mathsf{x}[\mathsf{index.fstar}], \mathsf{P}[m,t])$;
    end
  end
end
for $j \in \{1, \dots, G\}$ **do**
  $\mathsf{Schätzer.obere.Schranke}[j] = \frac{1}{M} \sum_{m=1}^{M} \mathsf{Wert}[0, j, m]$;
end

7 Numerische Betrachtung der Algorithmen zur Gasspeicherbewertung

Hat man Algorithmen für die Gasspeicherbewertung entwickelt, so gilt es diese zu implementieren. Dadurch lassen sich die verschiedenen Ansätze vergleichen. Neben Laufzeitanalysen, Schätzereigenschaften und dem Einfluss der Preisparameter lässt sich ein besserer Eindruck der die optimale Strategie bestimmenden Preisschranken gewinnen.

Die Algorithmen werden in R und C implementiert. Für eine genaue Beschreibung der verwendeten R- und C-Routinen wird auf die entsprechenden Handbücher ([30], [31],[17], [1]) hingewiesen.

Der erste Teil dieses Kapitels befasst sich mit dem konkreten Beispiel eines Gasspeichers, das in den folgenden Teilen betrachtet wird. Anschließend wird auf die mit der Implementierung zusammenhängende Diskretisierung genauer eingegangen. Ein Vergleich der Algorithmen, sowie weitere Untersuchungen mit unterschiedlichen Zielsetzungen folgen. In diesem Kapitel abschließend folgt eine Betrachtung der Algorithmen zur Berechnung einer oberen Schranke für den Gasspeicherwert.

7.1. Beispiel eines Gasspeichers: Eckdaten

Im Folgenden wird das Kernbeispiel vorgestellt, auf das die Algorithmen angewendet werden. Eventuelle Modifikationen werden an den entsprechenden Stellen beschrieben. Andernfalls wird von diesem Beispiel, das heißt insbesondere diesen Parametern ausgegangen.

Parameter im Preis

Wie in Kapitel 5 beschrieben, werden die von Benth u. a. [4] an die britischen Preisdaten angepassten Preisparameter für die numerische Betrachtung der Algorithmen verwendet. Die Log-Preise folgen demnach der stochastischen Differtialgleichung 5.2

$$dX_t = \alpha(\mu(t) - X_t)dt + \sigma(t)dB_t$$

mit

$$\begin{aligned}\alpha &= 0.073,\\ \mu(t) &= 2.69 + 0.0007t - 0.234\cos\left(\frac{2\pi(t-118.1)}{250}\right),\\ \sigma(t) &\equiv 0.072.\end{aligned}$$

Da die Energie in MMBtu (million British thermal units) gemessen werden soll, müssen die Preise noch umgerechnet werden. Die Daten beinhalten als Einheit pence per therm, wobei gilt

$$1 \text{ pence/therm} = 0.1 \text{ £/MMBtu}.$$

Als Anfangspreis (in Pfund) wird

$$p_0 = 0.1 \cdot e^{\mu(0)} \text{ £/MMBtu} \approx 1.855 \text{ £/MMBtu}$$

für die Berechnungen verwendet. Der Kauf- bzw. Verkaufpreis k bzw. e sei gegeben durch

$$\begin{aligned}k(p) &= (1+w_1)p + z_1 = 1.01p + 0.02,\\ e(p) &= (1-w_2)p - z_2 = 0.995p - 0.02.\end{aligned}$$

In Secomandi [34] werden die Parameter z_1 und z_2 als zusätzliche Kosten interpretiert, die zum Beispiel durch Abnutzung der Pumpe oder anderer Geräte entstehen. Sie beziehen sich hierbei auf Maragos [26], der beschreibt, dass diese Kosten normalerweise den Wert 2 Cent pro MMBtu nicht übersteigen. Zum anderen ist es möglich, die Parameter z_1 und z_2 zur Beschreibung eines im Markt vorhandenen Bid-Ask-Spreads zu verwenden. Entsprechende Gaspreise aus de Jong u. Walet [24] lassen hier vermuten, dass sich auch bei dieser Interpretationsmöglichkeit $z_1 = z_2 = 0.02$ £ als sinnvoll erweist.

Die Parameter $w_1 = 0.01$ und $w_2 = 0.005$ zu wählen ist auf den Gasverlust an der Pumpe zurückzuführen, der laut Maragos [26] normalerweise nicht mehr als 1% beträgt. Eine genauere Beschreibung findet man hierzu in Secomandi [34]. Alternativ ist auch eine Interpretation als Transaktionskosten möglich.

Der Gasspeicher

Ein realistisches Beispiel eines Gasspeichervertrags ist zum Beispiel in Thompson u. a. [37] zu finden. Der dort beschriebene Vertrag bezieht sich auf den Stratton Ridge Salzkavernenspeicher in Texas und hat eine Dauer von einem Jahr. In Anlehnung an diesen Gasspeicher werden

$$\begin{aligned}b^{\min} &= 500000 \text{ MMBtu},\\ b^{\max} &= 2000000 \text{ MMBtu},\\ x_0 &= b^{\max}/2 = 1000000 \text{ MMBtu},\end{aligned}$$

sowie

$$N = 250$$

gesetzt. Der Beginn des Vertrages ist Februar, was der Anpassung der Preisdaten entspricht. Mit Hilfe des idealen Gasgesetz und der Bernoulli-Gleichung erhalten Thompson u. a. [37] Gleichungen für $i_n^{\max}$ und $i_n^{\min}$. Angepasst auf die in dieser Arbeit verwendeten Einheiten und das Modell, ergibt sich mit ähnlicher Vorgehensweise

$$i_n^{\min}(x) = i^{\min}(x) = -70.71 \cdot \sqrt{x}.$$

Dies entspricht einer maximalen Ausspeicherrate von ca. 100000 MMBtu pro Tag. Außerdem würde es etwa 20 Tage dauern, bis der Speicher von maximalen Speicherstand $b^{\max}$ auf minimalen Speicherstand $b^{\min}$ geleert ist.

Als Funktion für die Einspeicherrate wird

$$i_n^{\max}(x) = i^{\max}(x) = -0.032 \cdot x + 68170$$

gewählt. Dies entspricht einer maximalen Einspeicherrate von ca. 52000 MMBtu pro Tag bei „leerem“ Speicher. In etwa 78 Tagen lässt sich der Speicher von minimalem Speicherstand $b^{\min}$ auf maximalen Speicherstand $b^{\max}$ befüllen. Die entsprechende Funktion in Thompson u. a. [37] ist konvex und erfüllt daher nicht die Voraussetzungen des Modells dieser Arbeit. Dennoch wird versucht, mit der affinen Funktion, die „Eckdaten“, d.h. maximale Einspeicherrate und Gesamtzeit zur Befüllung, entsprechend einzuhalten.

Die terminale Gewinnfunktion sei gegeben durch

$$h_N(x,p) = \begin{cases} e(p) \cdot (x - x_0), & \text{falls } x > x_0, \\ 0, & \text{falls } x = x_0, \\ k(p) \cdot (x - x_0), & \text{falls } x < x_0. \end{cases}$$

Damit sollte der Gasspeicher am Ende des Vertrages wieder den Anfangsfüllstand erreichen. Falls man darunter bleibt, muss das „fehlende“ Gas finanziert werden. Ist man über dem geforderten Zustand, so fließt der entsprechenden Betrag zurück. Vergleiche hierzu auch Beispiel 4.1.4.

Weitere Bespiele für Gasspeicher finden sich in [5], [24], [34] und [15]. Im Vergleich zum obigen Beispiel sind diese Gasspeicher langsamer in dem Sinne, dass es länger dauert bis der Speicher wieder voll/leer ist. Jedoch sind die Beispiele nicht an einen existierenden Gasspeicher angelehnt. De Jong u. Walet [24] bezeichnen ihr Beispiel als „realistic but stylized example“.

7.2. Diskretisierung in Speicherstand und Preis

Beschäftigt man sich mit Algorithmen, denen kontinuierliche Zustandsräume zu Grunde liegen, so stellen sich folgende Fragen. Wie ist dieser Zustandsraum zu diskretisieren? Welche Wahl und welche Länge eines Gitters ist optimal? Diese Fragen werden im folgende Teilkapitel im Zusammenhang mit der Diskretisierung des Intervalls möglicher Speichervolumen und des möglicher Preise diskutiert.

7.2.1. Diskretisierung des Speichervolumenintervall $[b^{\min}, b^{\max}]$

Motivation

Die naheliegende Wahl eines Diskretisierungsgitters für ein abgeschlossenes Intervall ist ein äquidistantes Gitter. Da die Speicherzustände, die erreicht werden können, im Intervall $[b^{\min}, b^{\max}]$ liegen und ihnen im Vergleich zum Preis kein direkter Stochastischer Prozess zu Grunde liegt, scheint es hier sinnvoll ein äquidistantes Gitter zu wählen.

Wirft man jedoch einen Blick auf die Struktur der optimalen Strategie, so erkennt man, dass in vielen Fällen von einem gegebenen Speicherzustand x aus der nächste Zustand $x + i^{\max}(x)$ oder $x + i^{\min}(x)$ ist.

Abbildung 7.1 zeigt zur Verdeutlichung die Strategieschranken $\underline{b}_n(p)$ (hellblau) und $\overline{b}_n(p)$ (dunkelblau) im Binomialbaum-Algorithmus. Zur Berechnung wurde ein äquidistantes Gitter mit 531 Gitterpunkten verwendet. Man erkennt deutlich, dass lediglich in einem schmalen Übergangsbereich die Strategieschranken wirklich zum Tragen kommen. Oft sind die Schranken gleich $b^{\min}$ oder $b^{\max}$, das heißt an diesen Preisen und Zeitpunkten würde man so viel Gas wie möglich entnehmen beziehungsweise einspeichern.

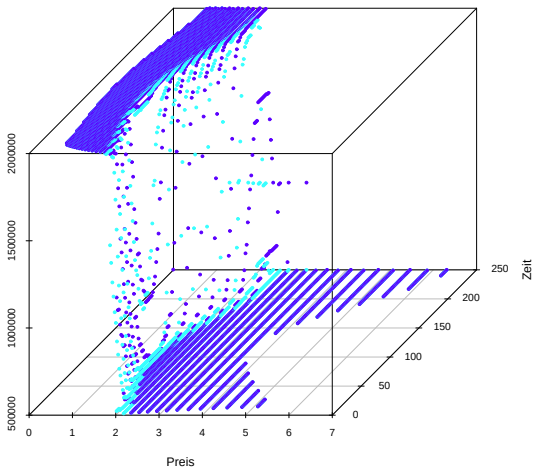


Abbildung 7.1.: Strategieschranken $\underline{b}_n(p)$ (hellblau) und $\overline{b}_n(p)$ (dunkelblau) im Baum-Algorithmus mit äquidistantem Gitter

Wie in Kapitel 6.1 bereits beschrieben, ist es bei allen Diskretisierungen des Speicherzustandes so, dass ein Fehler dadurch entsteht, dass die nächsten Zustände $x + i_n^{\max}(x)$ oder $x + i_n^{\min}(x)$ keine Gitterpunkte „treffen“. Insbesondere in einem äquidistanten Gitter, das nicht fein genug ist, wird dies oft der Fall sein.

Die Vorgehensweise dieser Arbeit ist es, mit Hilfe der Funktionen $i^{\max}$ und $i^{\min}$ das Gitter zu erzeugen.

Erzeugung des Gitters und Wahl der Anzahl der Gitterpunkte

Grundsätzlich sollten zum Gitter, welches den Speicherzustand diskretisiert, $b^{\max}, b^{\min}$ und x_0 gehören. Von $b^{\max}$ ausgehend wird dann $b^{\max} + i^{\min}(b^{\max}), b^{\max} + i^{\min}(b^{\max}) + i^{\min}(b^{\max} + i^{\min}(b^{\max})), \ldots$ berechnet bis $b^{\min}$ erreicht bzw. unterschritten wird. Genauso werden ausgehend von $b^{\min}$ die Gitterpunkte $b^{\min} + i^{\max}(b^{\min}), b^{\min} + i^{\max}(b^{\min}) + i^{\max}(b^{\min} + i^{\max}(b^{\min})), \ldots$ berechnet bis $b^{\max}$ erreicht bzw. überschritten wird. In ähnlicher Weise werden dann von x_0 aus - sofern x_0 nicht $b^{\max}$ oder $b^{\min}$ entspricht - Gitterpunkte $x_0 + i^{\min}(x_0), x_0 + i^{\min}(x_0) + i^{\min}(x_0 + i^{\min}(x_0)), \ldots$ und $x_0 + i^{\max}(x_0), x_0 + i^{\max}(x_0) + i^{\max}(x_0 + i^{\max}(x_0)), \ldots$ berechnet bis der minimale bzw. maximale Speicherstand erreicht wird.

Die so entstehenden Gitterpunkte werden in jedem Fall verwendet und bilden das „Grundgitter“. Im Beispiel dieser Arbeit hat es eine Länge von 173 Gitterpunkten. Das bedeutet demnach, dass mindestens mit einem Gitter dieser Länge gerechnet wird. Algorithmus 13 beschreibt die Erzeugung des Grundgitters, wobei hier davon ausgegangen wird, dass - wie im Beispiel - x_0 weder $b^{\max}$ noch $b^{\min}$ entspricht. Die Funktionen `sort` und `unique` sind Funktionen, die einen Vektor sortieren bzw. doppelte Werte entfernen.

Algorithmus 13 : Erzeugung des Grundgitters im Speicherstand

Data : x.0,b.max,b.min, i.max,i.min
Result : Vektor der das grundgitter enthält

grundgitter $[1]$ = b.min;
$i = 1$;
while grundgitter $[i]$ < b.max **do**
  x = grundgitter$[i]$ + i.max(grundgitter$[i]$);
  grundgitter$[i+1]$ = min$(x$, b.max$)$;
  $i = i + 1$;
end
while grundgitter $[i]$ > b.min **do**
  x = grundgitter$[i]$ + i.min(grundgitter$[i]$);
  grundgitter$[i+1]$ = max$(x$, b.min$)$;
  $i = i + 1$;
end
$j = i + 1$;
grundgitter $[j]$ = x.0;
while grundgitter $[j]$ < b.max **do**
  x = grundgitter$[j]$ + i.max(grundgitter$[j]$);
  grundgitter$[j+1]$ = min$(x$, b.max$)$;
  $j = j + 1$;
end
$k = j + 1$;
grundgitter $[k]$ = x.0;
while grundgitter $[k]$ > b.min **do**
  x = grundgitter$[k]$ + i.min(grundgitter$[k]$);
  grundgitter$[k+1]$ = max$(x$, b.min$)$;
  $k = k + 1$;
end
grundgitter = `unique` (`sort` (grundgitter));

Ist die gewünschte Mindestanzahl an Gitterpunkten mit dem Grundgitter noch nicht erreicht, so wird ausgehend von x_0 der nächstliegende von den Gitterpunkten gesucht, die ausgehend von x_0 entstanden sind. Ausgehend von diesem werden neue Gitterpunkte in ähnlicher Weise wie vorher mit Hilfe von $i^{\max}$ oder $i^{\min}$ berechnet, je nachdem ob der ausgesuchte Gitterpunkt unterhalb oder oberhalb von x_0 liegt. Darauffolgend wird der zweit-nächstliegende Gitterpunkt gesucht und neue Gitterpunkte berechnet. Auf diese Weise wird fortgefahren bis die Mindestgitterlänge erreicht oder übertroffen wird. Für die genauere Beschreibung wird hier auf die zugehörigen R- bzw. C-Dateien verwiesen.

Um einen Eindruck des so entstehenden Gitters zu erhalten, zeigen die Abbildungen 7.2, 7.3, 7.4, wie die Gitterpunkte im Intervall $[b^{\min}, b^{\max}]$ liegen bzw. Histogramme dieser mit Mindestlängen 100, 500 und 1500. Die erste Abbildung zeigt das Grundgitter; die tatsächlichen Gitterlängen der darauffolgenden Abbildungen betragen 530 bzw. 1506.

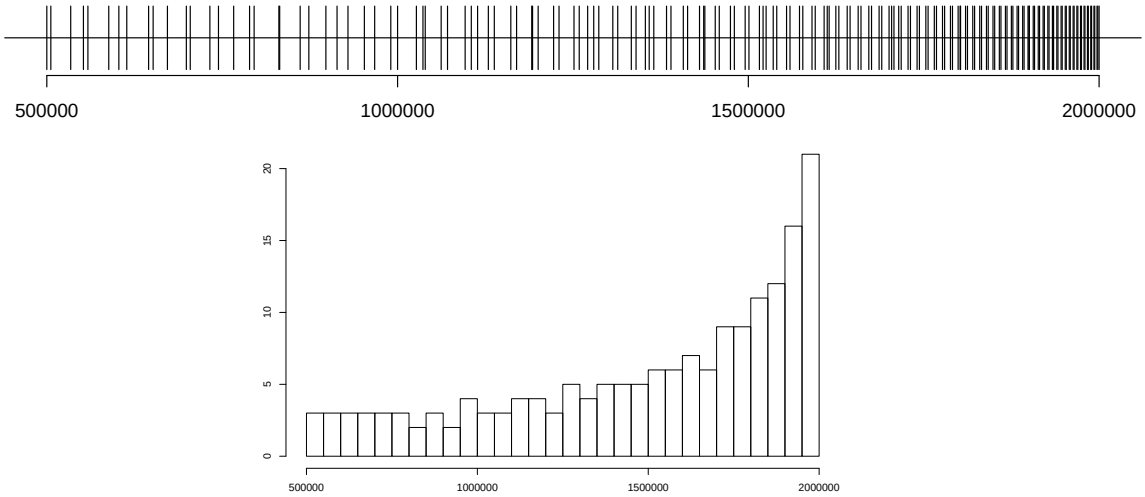


Abbildung 7.2.: Lage und Histogramm des Grundgitters (mit Minimallänge 137)

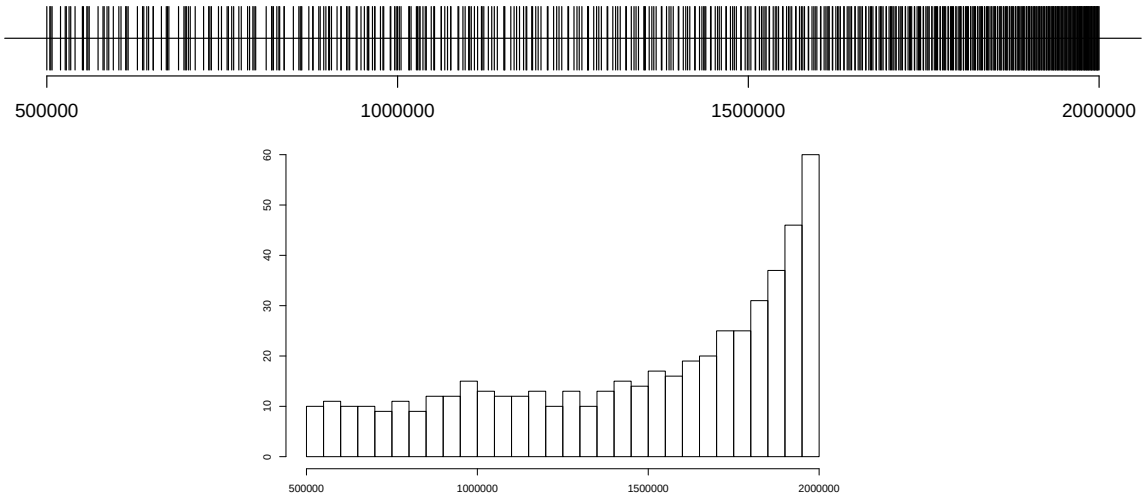


Abbildung 7.3.: Lage und Histogramm des Gitters mit Minimallänge 500

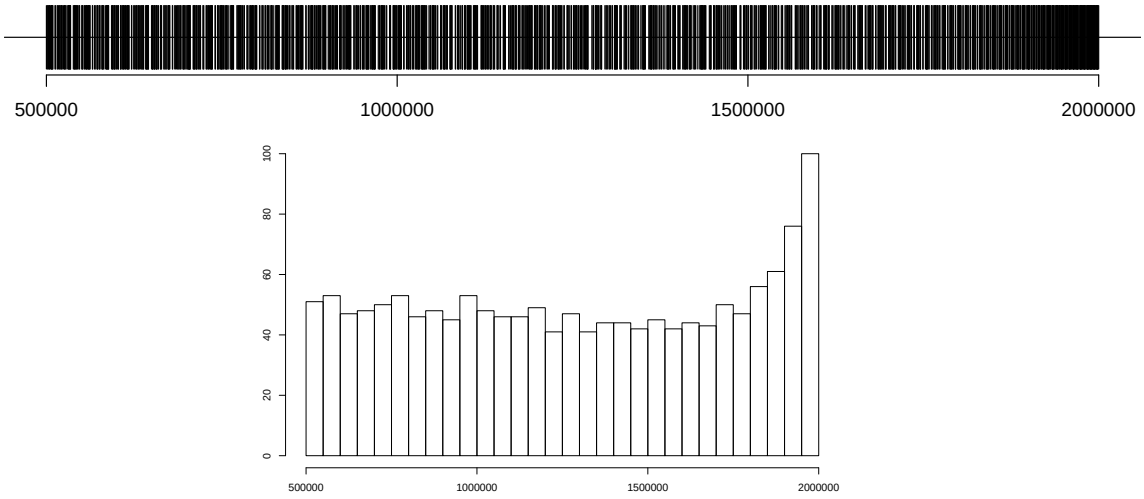


Abbildung 7.4.: Lage und Histogramm des Gitters mit Minimallänge 1500

Vergleicht man die beiden Ansätze - äquidistantes Gitter (Sternsymbole) und Gitter mit Hilfe der Funktionen $i^{\max}$ und $i^{\min}$ (Punkte) - so erkennt man in den Abbildungen 7.5 und 7.6, dass sowohl im Binomialbaum-Algorithmus, als auch im Least-Square-Monte-Carlo Algorithmus, die Methode mit Hilfe der Funktionen $i^{\max}$ und $i^{\min}$, wie sie in dieser Arbeit verwendet wird, schneller konvergiert.

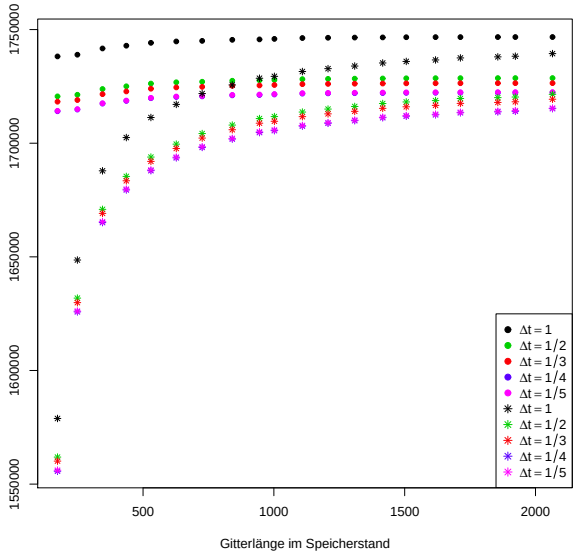


Abbildung 7.5.: Wert des Gasspeichers mit Baumalgorithmus

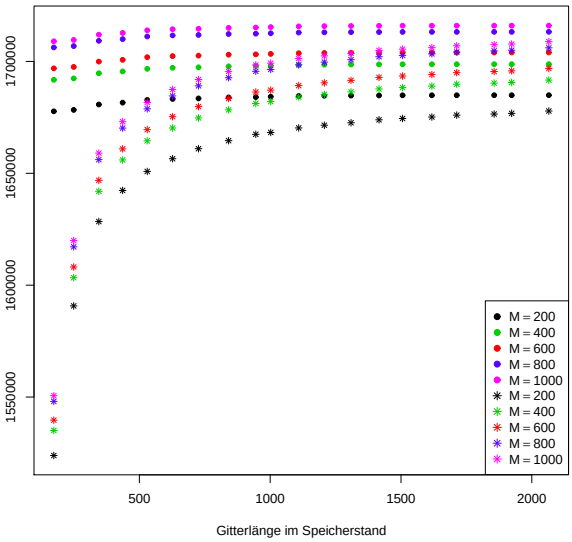


Abbildung 7.6.: Wert des Gasspeichers mit Least-Square Monte Carlo Algorithmus

Eine weitere Frage, die sich stellt ist, welche Gitterlängenwahl möglichst optimal in dem Sinne ist, dass die Gitterlänge, das heißt die Anzahl der Gitterpunkte, möglichst klein und der Wert des Gasspeichers möglichst nahe am „wahren“ Wert liegt.

Die Abbildungen 7.7 bis 7.9 zeigen, wie sich der Wert des Gasspeichers in Abhängigkeit der Gitterlänge verhält. Die beiden Abbildungen 7.7 und 7.9 weisen bei ca. 500 einen ersten größeren und bei ca. 1000 einen kleineren zweiten Knick auf, sodass diese zwei Werte eine gute Wahl darstellen. Auch in Abbildung 7.9 lässt sich ein Knick bei ca. 500 erkennen, insbesondere, wenn man die Werte auf der Skala des LS-Algorithmus betrachtet (rechts). Um nicht zu hohe Laufzeiten der Algorithmen zu erhalten, wird in den folgenden Teilkapiteln aus diesem Grund die Mindestlänge des Gitters im Speicherstand auf 500 gesetzt. Tatsächlich bedeutet dies eine Gitterlänge von 530 Gitterpunkten.

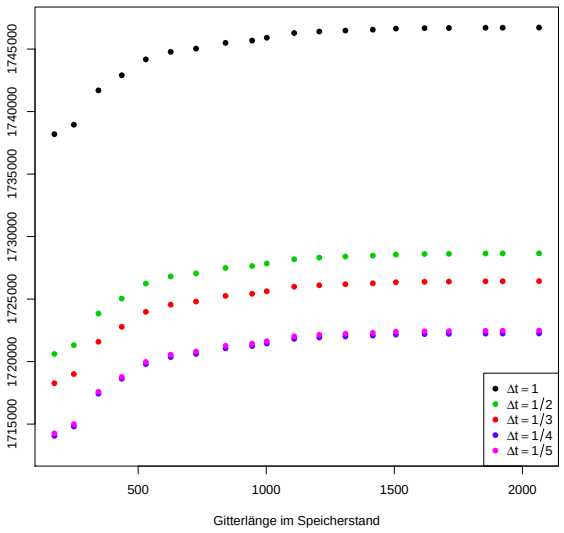


Abbildung 7.7.: Wert des Gasspeichers mit Baumalgorithmus in Abhängigkeit der Gitterlänge im Speicherstand

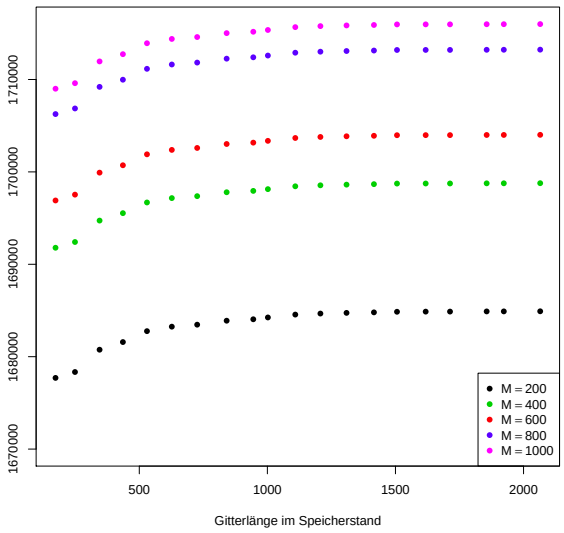
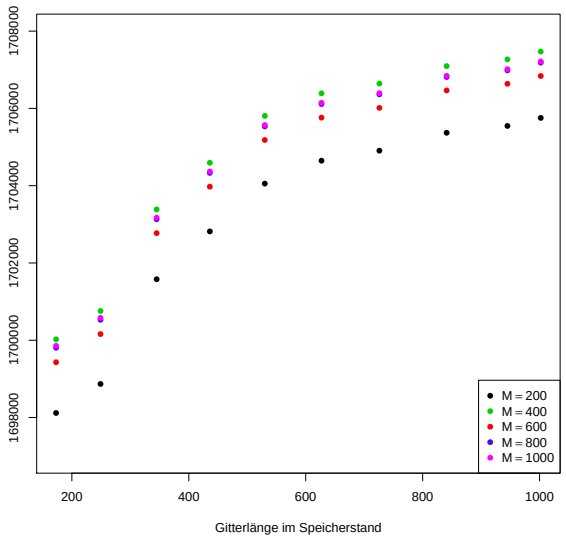


Abbildung 7.8.: Wert des Gasspeichers mit Least-Square-Algorithmus in Abhängigkeit der Gitterlänge im Speicherstand



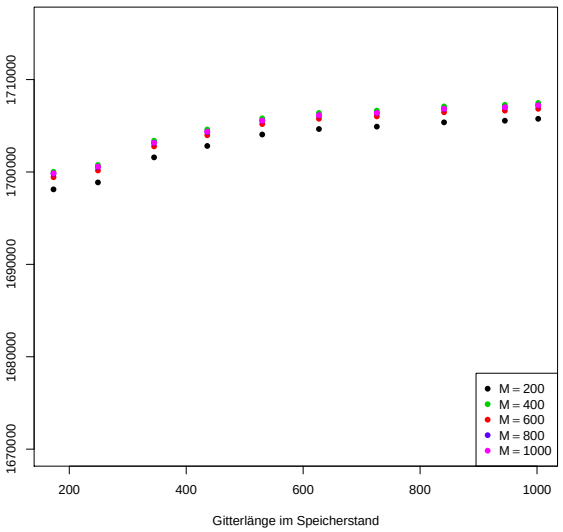


Abbildung 7.9.: Wert des Gasspeichers mit Interpolations-Algorithmus und Quantisierung in Abhängigkeit der Gitterlänge im Speicherstand

7.2.2. Diskretisierung des Preiszustandes

Die Methoden zur Diskretisierung des Preiszustandes sind bereits in Kapitel 5 beschrieben. Zum einen ist dort der approximierende Binomialbaum beschrieben, zum anderen die den Monte Carlo Algorithmen zugrunde liegenden Simulationen der Preispfade.
Im Folgenden stellt sich die Frage, wie im Baum-Algorithmus die Schrittweite der Approximation Δt und in den Monte Carlo Algorithmen die Anzahl der simulierten Preispfade M zu wählen sind.

Abbildung 7.10 zeigt Gasspeicherwerte in Abhängigkeit von Δt. Wie auch in Abbildung 7.7 zu erkennen ist, ist der Wertunterschied von $\Delta t = \frac{1}{4}$ und $\Delta t = \frac{1}{5}$ geringfügig, was die Wahl von $\Delta t = \frac{1}{4}$ für weitere Untersuchungen nahe legt.

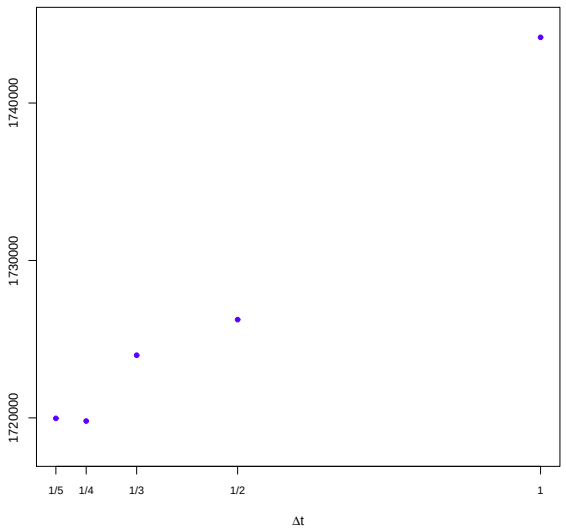


Abbildung 7.10.: Wert des Gasspeichers mit Baumalgorithmus in Abhängigkeit von Δt

Für die Abbildungen der Monte-Carlo Algorithmen wurden 1000 Pfade erzeugt und die entsprechenden Gasspeicherwerte für verschiedene Werte von M berechnet, indem jeweils mehr Pfade hinzugenommen werden. Um einen besseren Eindruck zu erhalten wurde dies 5 mal durchgeführt und durch verschiedene Farben gekennzeichnet.

Abbildung 7.11 zeigt den Speicherwert in Abhängigkeit von M im Interpolationsansatz mit Quantisierung. Es wurden jeweils 100 Quantisierungspunkte aus [13], wie in 6.3.1 beschrieben, verwendet. Für die Approximation der Wertefunktion wurde eine lineare Interpolation (links) und natürliche Splines (rechts) verwendet. Beide Methoden weisen ein ähnliches Verhalten in Abhängigkeit von M auf, wobei die Spline-Methode dahingehend besser erscheint, dass die Abweichung der verschiedenen Durchführungen (Farben) relativ klein ist.

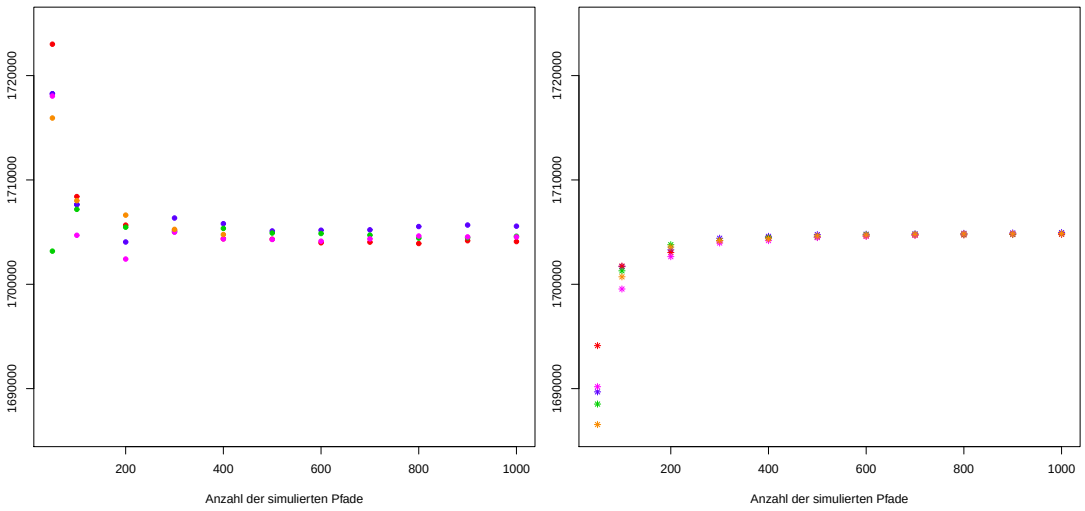


Abbildung 7.11.: Wert des Gasspeichers mit Interpolations-Algorithmus mit Quantisierung und linearer Interpolation (links) bzw. Splines (rechts) in Abhängigkeit der Anzahl der Pfade M

Abbildung 7.12 zeigt, dass die Konvergenzgeschwindigkeit der Methode mit „nested“ Simulation kleiner ist im Vergleich zu der Konvergenzgeschwindigkeit der Methode mit Quantisierung. Die zu erkennenden Unregelmäßigkeiten innerhalb der einzelnen Durchführungen in der linken Grafik, sind auf die unterschiedlichen „nested“ Simulationen zurückzuführen. In der rechten Grafik wird mit gleichen Ausgangsimulationen gerechnet, das heißt innerhalb des Algorithmus ein seed gesetzt. Es wurden jeweils 100 Simulationen zur Berechnung des Erwartungswertes verwendet. Da beide Methoden - mit und ohne seed - jedoch recht ähnliches Verhalten ausweisen, wird in den folgenden Betrachtungen, die Methode ohne seed verwendet. Als Approximation wurde hier eine lineare Interpolation verwendet.

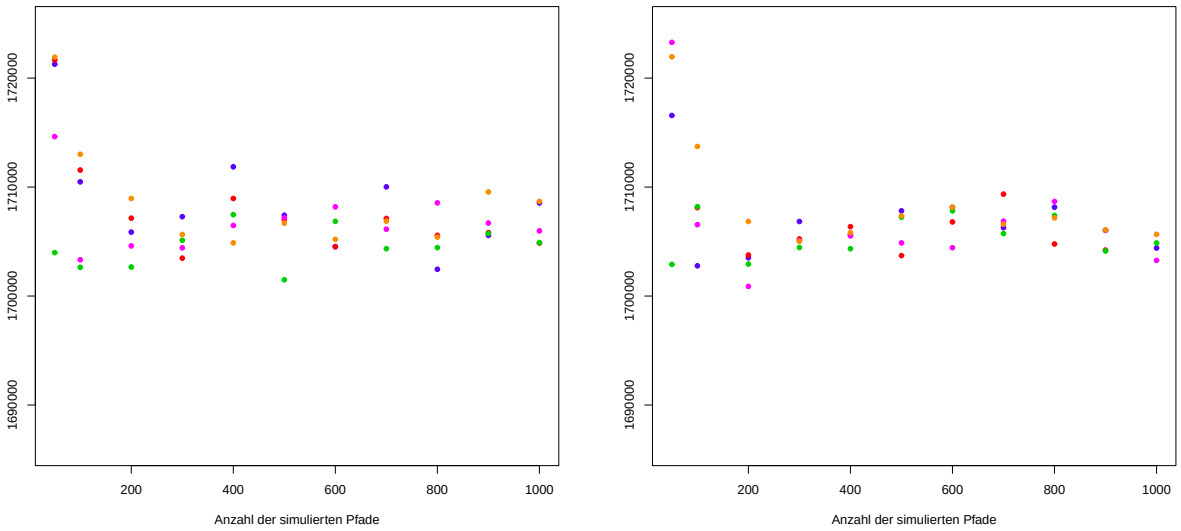


Abbildung 7.12.: Wert des Gasspeichers mit Interpolations-Algorithmus mit Simulation ohne (links) bzw. mit (rechts) gesetztem seed in Abhängigkeit der Anzahl der Pfade M

In Abbildung 7.13 ist erkennbar, dass der Monte-Carlo Algorithmus mit Least-Square-Ansatz, langsamer konvergiert. Als Basisfunktionen der Approximation wurden die Monome $1, x, x^2, x^3$ verwendet.

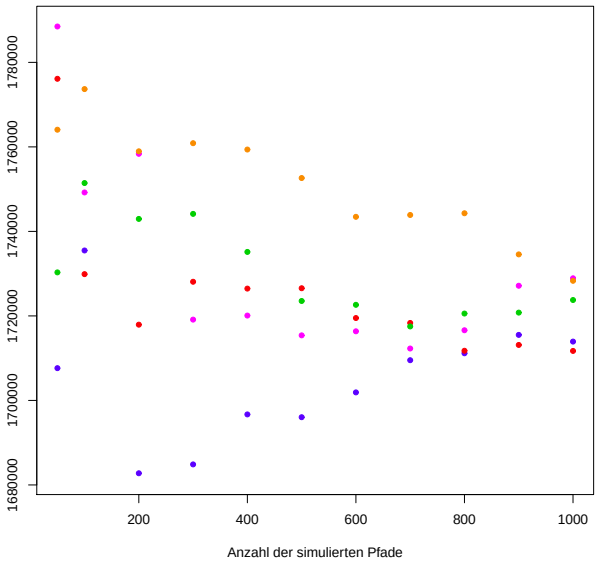


Abbildung 7.13.: Wert des Gasspeichers mit Least-Square-Algorithmus in Abhängigkeit der Anzahl der Pfade M

Im Vergleich der Algorithmen, siehe Abbildung 7.14, ist zu erkennen, dass die Interpolationsalgorithmen, die in der Mitte (Simulation) und Rechts (Quantisierung) gezeigt werden, bereits mit $M = 200$ annehmbare Ergebnisse vorweisen, wohingegen der Least-Square-Algorithmus eine erheblich höhere Anzahl an simulierten Pfaden benötigt. Im folgenden werden für den Least-Square-Algorithmus $M = 1000$ Pfade verwendet. Für den Interpolationsansatz werden die Werte $M = 200$, $M = 500$ und $M = 1000$ verwendet.

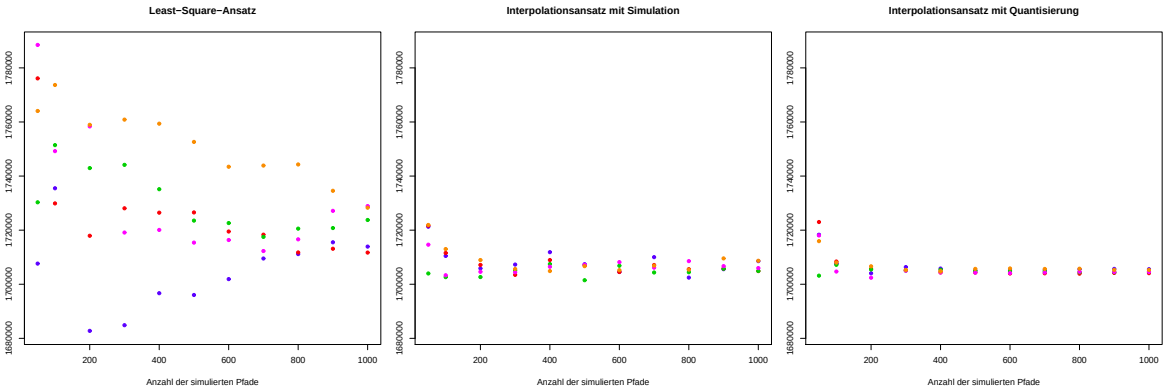


Abbildung 7.14.: Wert des Gasspeichers verschiedener Algorithmen in Abhängigkeit der Anzahl der Pfade M

7.3. Numerischer Vergleich der Algorithmen und weitere Untersuchungen

Einen ersten Eindruck der Unterschiede in den einzelnen Algorithmen liefern bereits die Abbildungen in Kapitel 7.2. Diese Unterschiede werden in diesem Teilkapitel weitergehend diskutiert. Zudem werden die Strategieschranken und die optimale Politik untersucht.

7.3.1. Die Monte-Carlo-Algorithmen: Erste Untersuchung und Vergleiche untereinander

Bevor ein Vergleich zum Binomialbaum-Algorithmus hergestellt wird, werden zunächst die Monte Carlo Algorithmen untereinander untersucht. Aus Lesbarkeitsgründen wird häufig anstelle der genauen Bezeichnungen „Monte Carlo Algorithmus mit Interpolationsansatz und Integration/Simulation/Quantisierung“ beziehungsweisen „Monte Carlo Algorithmus mit Least-Square-Ansatz“ kürzere Bezeichnungen wie zum Beispiel „Interpolations-Algorithmus“ oder „Least-Square-Algorithmus“ verwendet. Aus dem Kontext ist allerdings zu erkennen, welcher Algorithmus verwendet ist.

Der Interpolationsansatz

Bevor im Folgenden der Monte-Carlo-Algorithmus mit Interpolationsansatz mit seinen verschiedenen Möglichkeiten, den bedingten Erwartungswert zu berechnen, genauer betrachtet wird, wird begründet, warum der Teil-Ansatz „Integral“ gänzlich und der Teil-Ansatz „Simulation“ im (natürlichen) Spline-Ansatz nicht weiter betrachtet werden.

Bemerkung 7.3.1
Zum einen zeigt die Tabelle 7.1 die Werte des Interpolationsansatzes mit natürlichem Spline und Berechnung des Erwartungswertes durch ein Integral. Die jeweils 5 berechneten Werte mit 100 bzw. 150 Pfaden deuten ein recht gutes Verhalten des Schätzers an. Jedoch benötigt schon ein Durchlauf mit „nur“ 100 Pfaden ca. 3 Stunden und ein Durchlauf mit 150 Pfaden ca. 4 Stunden, sodass dieser Ansatz sehr viel Zeit in Anspruch nimmt. Das zur Zeitmessung verwendete System wird in 7.3.2 beschrieben. Hinzu kommt, dass bei höherer Anzahl an Pfaden, Probleme bei der Benutzung der Funktion `integrate` zur Berechnung des Integrals in R auftreten. Diese Routine kann das Integral aufgrund der Beschaffenheit des Integranden nicht berechnen. Bei der Verwendung einer linearen Interpolation anstelle eines Splines treten diese Probleme bereits bei 150 Pfaden auf. Desweiteren ist zu bemerken, dass bei den Berechnungen aus Tabelle 7.1 bereits nicht das Integral von 0 bis ∞ berechnet wird, sondern nur auf einem kompakten Intervall, indem möglichst viel der Wahrscheinlichkeitsmasse liegt. Dieses Intervall wurde im Sinne der k-σ-Bereiche der Normalverteilung auf die Log-Normalverteilung angepasst.

Aus diesen Gründen wird im Weiteren auf die Untersuchung dieses Teil-Ansatzes mit Integral verzichtet.

M=100	Wert des Speichers	Laufzeit in Minuten
1. Simulation	1701677	157.7725
2. Simulation	1701739	159.4552
3. Simulation	1701284	157.3015
4. Simulation	1699487	157.3530
5. Simulation	1700698	157.5258
Mittelwert	1700977	157.8816
Standard-abweichung	930.3174	0.8986822

M=150	Wert des Speichers	Laufzeit in Minuten
1. Simulation	1702711	243.2587
2. Simulation	1702033	244.0843
3. Simulation	1703172	243.0338
4. Simulation	1701406	242.0230
5. Simulation	1702026	243.1535
Mittelwert	1702270	243.1107
Standard-abweichung	683.8308	0.7349639

Tabelle 7.1.: Speicherwerte in 5 Simulationsdurchgängen des Interpolations-Algorithmus mit Integration

Zum anderen wird auf die weitere Betrachtung des Interpolationsansatzes mit Spline in der Berechnung des bedingten Erwartungswertes durch Simulation verzichtet. Der Spline-Ansatz hat in diesem Fall keine sinnvollen Werte hervorgebracht. Daher wird hier nur der Ansatz mit linearer Interpolation weiter verfolgt.

Aufgrund der Bemerkung 7.3.1 werden im Folgenden der Interpolationsansatz mit linearer Interpolation sowohl in der Quantisierungsmethode, als auch in der Simulationsmethode betrachtet, der Interpolationsansatz mit natürlichem Spline hingegen nur in der Quantisierungsmethode.

Die erste Frage, die geklärt werden soll, ist die nach der Wahl der Anzahl l der Quantisierungspunkte beziehungweise der Simulationen zur Approximation des bedingten Erwartungswertes $\mathbb{E}[\widetilde{V}_{n+1}(x, P_{n+1})|P_n = p]$.

Hierzu zeigen die Abbildungen 7.15 und 7.16 Werte des Speichers zum Zeitpunkt 0 in Abhängigkeit dieser Anzahl, die mit $M = 500$ Pfaden berechnet werden. In Bild 7.15 erkennt man, dass sowohl bei der Verwendung einer linearen Interpolation (links), als auch bei der Verwendung eines natürlichen Splines (rechts) ein Knick bei $l = 100$ vorliegt. Dies bedeutet, dass für die weiteren Untersuchungen 100 Quantisierungspunkte verwendet werden. Wie in Kapitel 6 bereits erwähnt, werden diese aus der Datenbank auf der Quantization Website [13] entnommen.

Abbildung 7.16 zeigt die Werte durch Simulation, einmal ohne (links) und einmal mit (rechts) gesetztem seed. Im Gegensatz zum linken Bild, ist im rechten eine Konvergenz deutlicher zu erkennen. Im Fall von Simulationen würde man im Vergleich zur Quantisierung eher $l = 150$ wählen, dennoch werden die folgenden Untersuchungen mit $l = 100$ durchgeführt.

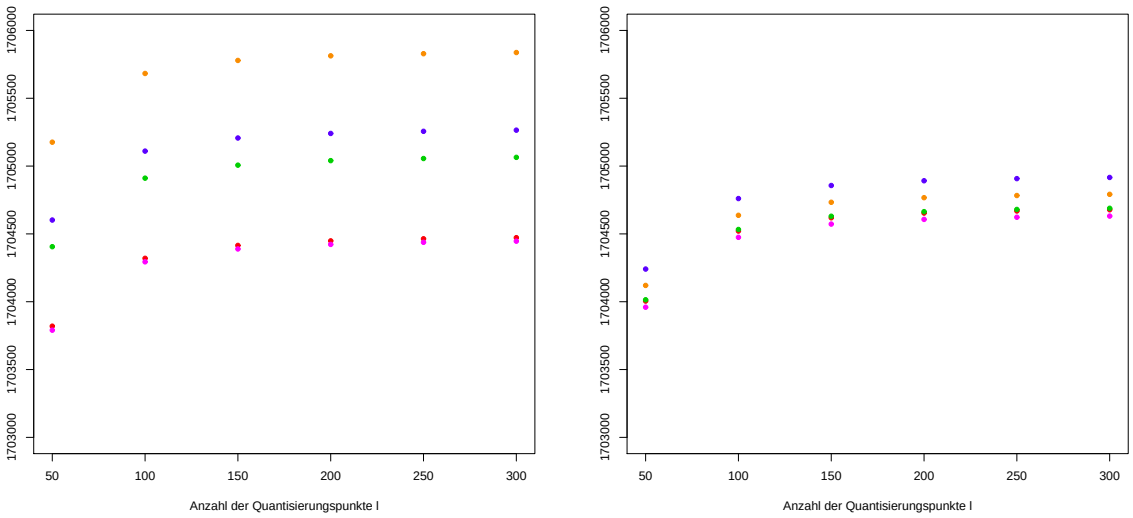


Abbildung 7.15.: Wert des Gasspeichers des Interpolations-Algorithmus mit linearer Interpolation (links) bzw. Splines (rechts) und Quantisierung in Abhängigkeit der Anzahl der Quantisierungspunkte l

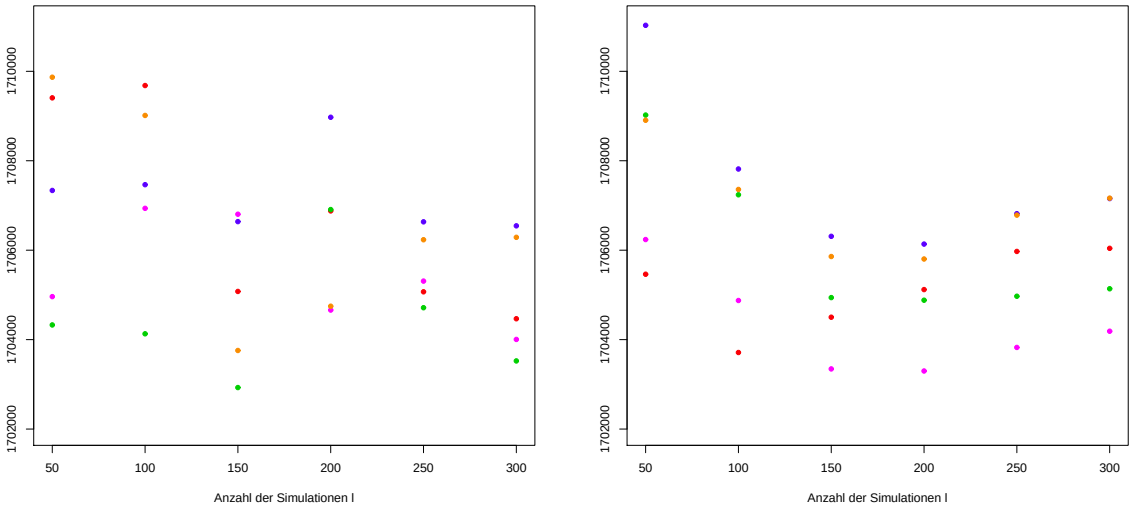


Abbildung 7.16.: Wert des Gasspeichers des Interpolations-Algorithmus mit Simulation ohne (links) bzw. mit (rechts) gesetztem seed in Abhängigkeit der Anzahl der Simulationen ohne und mit seed

Als nächstes wird kurz der Einfluss der „nested“ Simulationen im Ansatz mit Simulationen betrachtet. Abbildung 7.17 zeigt zwei Boxplots, die dadurch entstanden sind, dass der Interpolations-Algorithmus mit „nested“ Simulationen 100 mal mit den selben zugrundeliegenden Pfaden durchgeführt wird, jedoch mit verschiedenen Simulationen zur Berechnung des bedingten Erwartungswertes. Dem linken Boxplot liegen $M = 200$ Pfade, dem rechten $M = 500$ Pfade zugrunde. Neben dem Boxplot ist auch der Mittelwert (roter Punkt) und Mittelwert plus bzw. minus die empirische Standardabweichung (blaues Intervall) mit eingezeichnet. Der Mittelwert beträgt 1705221£ bei $M = 200$ und

1706591£ bei $M = 500$ Pfaden. Die empirische Standardabweichung ist 2670.373 beziehungsweise 2204.73, die Spannweite 11310.26 beziehungsweise 9877.638. Im Vergleich zum Mittelwert sind sowohl empirische Standardabweichung als auch Spannweite nicht hoch und vor allem der Unterschied zwischen den Werten für $M = 200$ und $M = 500$ gering. Dennoch sollte berücksichtigt werden, dass die „innere“ Simulation existiert und damit eine „innere“ Varianz des Schätzers zusätzlich zu der „äußeren“ Varianz aus der Pfad-Simulation entsteht.

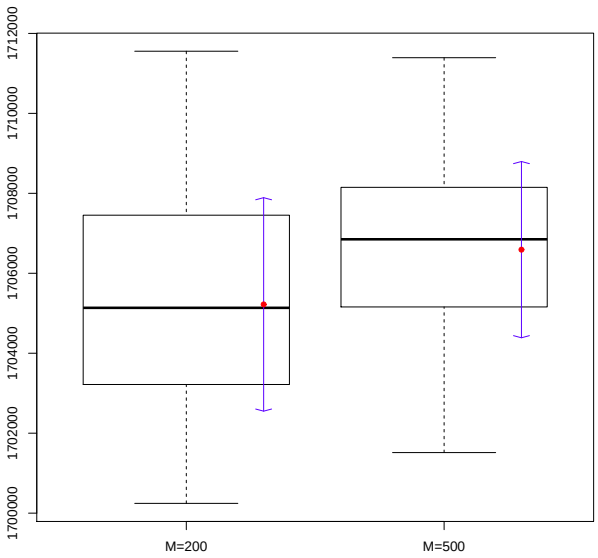


Abbildung 7.17.: Boxplot des Gasspeicherwertes des Interpolations-Algorithmus mit unterschiedlichen Simulation

Im Folgenden werden drei Methoden des Interpolationsansatzes verglichen, diese sind „lineare Interpolation mit Simulation“, „lineare Interpolation mit Quantisierung“ und „Spline-Interpolation mit Quantisierung“. Zur Berechnung des Erwartungswertes werden dazu jeweils 100 Simulationen (ohne seed) bzw. Quantsierungspunkte verwendet. Für $M = 200$, $M = 500$ und $M = 1000$ Pfade werden jeweils 200 Schätzwerte der drei Methoden berechnet. Den Mittelwert, die empirische Standardabweichung und die Spannweite der Schätzwerte zeigt Tabelle 7.2. Mittelwert und Standardabweichung sind, wie im vorherigen Boxplot, auch in die jeweils 3 - für die verschiedenen Pfadanzahlen - entstehenden Boxplots in Abbildung 7.18 mit eingezeichnet. Man beachte hier die verschiedenen Skalen, auf denen die Boxplots der unterschiedlichen Methoden aufgetragen sind.

Es ist zu erkennen, dass sich im Ansatz mit linearer Interpolation die Mittelwerte um 646£ (Simulation) bzw. 248£ (Quantisierung) unterscheiden, wohingegen beim Ansatz mit Spline-Interpolation der Unterschied 1666£ beträgt. Allerdings ist im Ansatz mit Spline-Interpolation die Standardabweichung und die Spannweite geringer. Es ist zudem zu bemerken, dass die Mittelwerte der beiden Quantisierungsansätze bei $M = 1000$ nur um 3£ abweichen. Auch die Boxplots geben dieses Verhalten zu erkennen. Der Median verhält sich hier ähnlich dem Mittelwert im linearen Interpolationsansatz im Vergleich

zum Spline-Interpolationsansatz, bei dem ein größerer Anstieg festzustellen ist. Genauso sind die Quartilsabstände im linearen Interpolationsansatz wesentlich größer als beim Spline-Ansatz. Während man in den linearen Interpolationsansätzen eine größere Anzahl M an Pfaden wählen würde, um vor allem die Standardabweichung zu verkleinern, sollte man beim Spline-Interpolationsansatz eine größere Anzahl M wählen, um den Mittelwert zu „verbessern“.

Insbesondere die Reduktion der Streuung des Schätzers in der Erhöhung von M zeigen die Histogramme in den Abbildungen 7.19, 7.20 und 7.21. Auch die Histogramme unterstreichen die Veränderung im Mittelwert, der dunkelblau eingezeichnet ist, und bekräftigen die Ergebnisse der Boxplots und der Tabelle.

Die Histogramme sind in der jeweiligen Methode auf der gleichen Skala. Vergleicht man die beiden linearen Interpolationsansätze, so zeigt sich, dass die Varianzreduktion im Quantisierungsfall wesentlich größer ist als im Simulationsfall. Hinzu kommt im Simulationsfall, neben der Varianz in der Simulation der Pfade, die bereits angesprochnene Varianz der „inneren“ Simulation zur Berechnung des bedingten Erwartungswertes. Müsste man sich für eine Methode entscheiden, so ist der lineare Interpolationsansatz mit Quantisierung zu wählen, da er trotz höherer Standardabweichung im Mittelwert und Median einen besseren Eindruck macht. Dennoch ist insgesamt festzustellen, dass alle drei Methoden bereits mit $M = 200$ Pfaden einen guten Schätzer ergeben, da die Abweichungen prozentual zum Mittelwert gesehen relativ klein sind. So beträgt im linearen Interpolationsansatz mit Simulation die Standardabweichung vom Mittelwert in etwa 0.23%.

Methode		M	Mittelwert	Standard-abweichung	Spannweite
Simulation	lineare Interpolation	200	1706773	3855.69	17347.41
		500	1706127	2293.382	10930.52
		1000	1706294	1740.8	10692.42
Quantisierung	lineare Interpolation	200	1704608	2185.568	12014.04
		500	1704713	794.143	4003.94
		1000	1704856	408.7203	2244.941
	Spline-Interpolation	200	1703193	398.7616	2522.498
		500	1704531	110.0713	598.8503
		1000	1704859	43.47782	254.3011

Tabelle 7.2.: Mittelwerte, Standardabweichung und Spannweite der Speicherwerte in 200 Simulationen mit dem Interpolationsansatz in verschiedenen Methoden

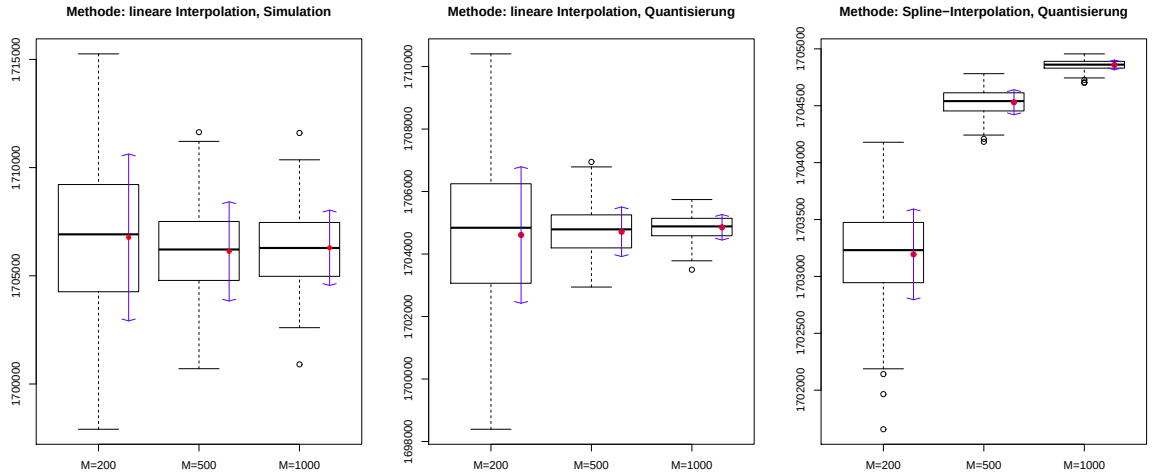


Abbildung 7.18.: Boxplot des Gasspeicherwertes des Interpolations-Algorithmus für verschiedenen M

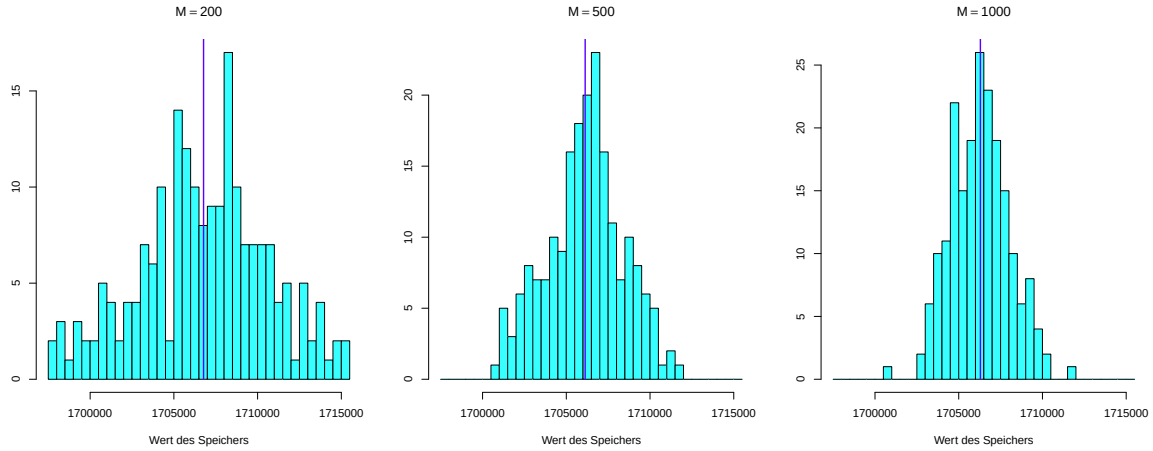


Abbildung 7.19.: Histogramme des Gasspeicherwertes des Interpolations-Algorithmus mit linearer Interpolation „nested“ Simulation für verschiedene M

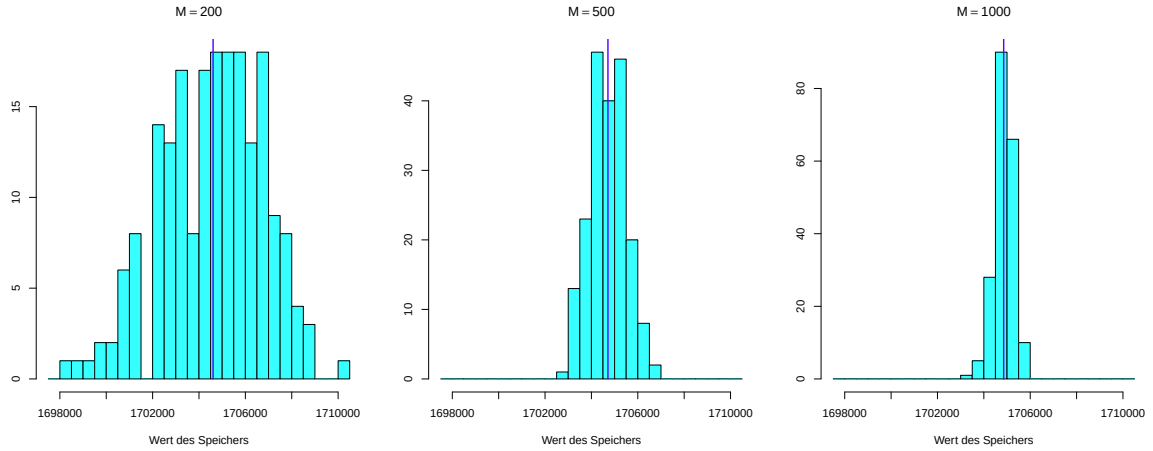


Abbildung 7.20.: Histogramme des Gasspeicherwertes des Interpolations-Algorithmus mit linearer Interpolation und Quantisierung für verschiedene M

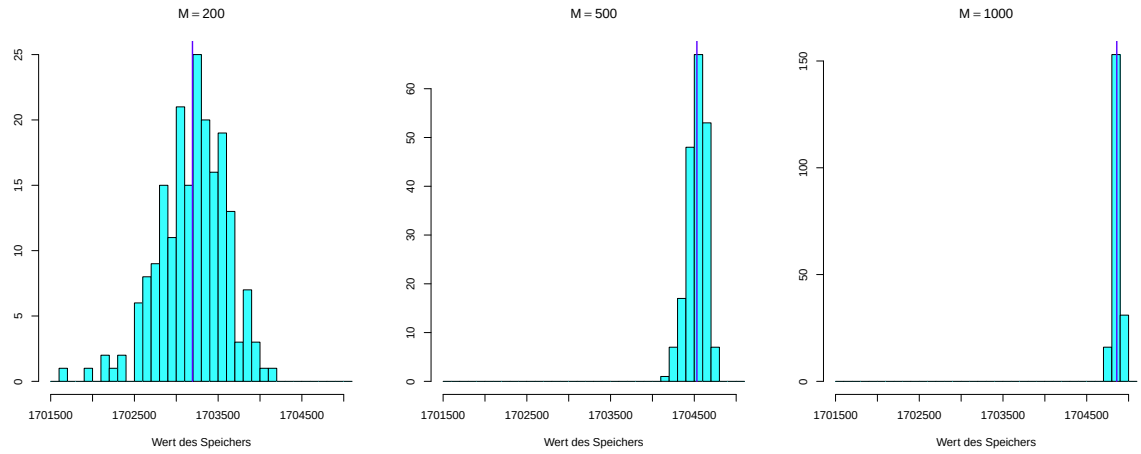


Abbildung 7.21.: Histogramme des Gasspeicherwertes des Interpolations-Algorithmus mit Spline-Interpolation und Quantisierung für verschiedene M

Der Least-Square-Ansatz im Vergleich zum Interpolationsansatz

Betrachtet man Boxplot und Histogramm der mit Hilfe des Least-Square-Ansatzes entstandenen 200 Schätzwerte (bei je $M = 1000$ simulierten Pfaden) in Abbildung 7.22, so fällt, die im Vergleich zum Interpolationsansatz größere Skala auf, auf der die Werte abgetragen sind. Die Skala ist vor allem nach oben hin vergrößert zu sein, was auch der Mittelwert 1718785£ verdeutlicht. Die Standardabweichung liegt hier bei 9611.739£ und die Spannweite bei 48305.08£. Damit sind beide Werte deutlich höher als im Interpolationsansatz. Zu erkennen ist diese Eigenschaft besonders in Abbildung 7.23, in der links der Schätzer mit Least-Square-Ansatz, in der Mitte der Schätzer mit linearem Interpolationsansatz und Simulation und rechts der Schätzer mit linearem Interpolationsansatz und Quantisierung zu sehen sind. Dennoch ist zu erwähnen, dass auch hier im Vergleich zum Mittelwert, die Standardabweichung mit etwa 0.56% des Mittelwertes durchaus annehmbar ist.

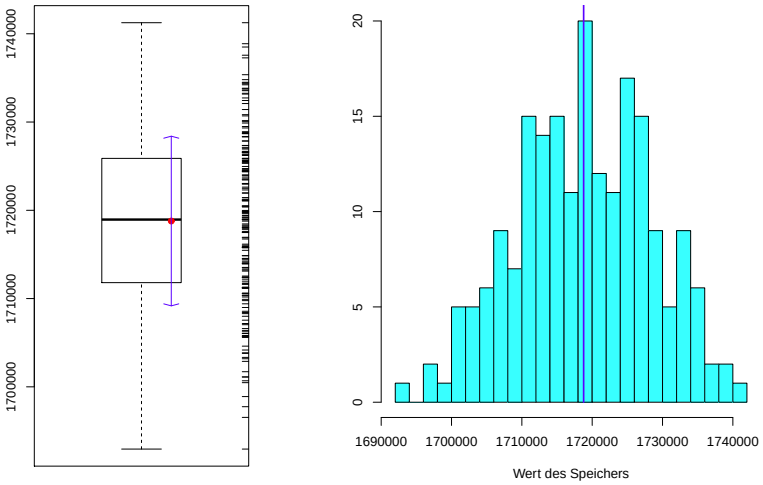


Abbildung 7.22.: Boxplot und Histogramm des Speicherwertschätzers mit Least-Square-Ansatz und $M = 1000$

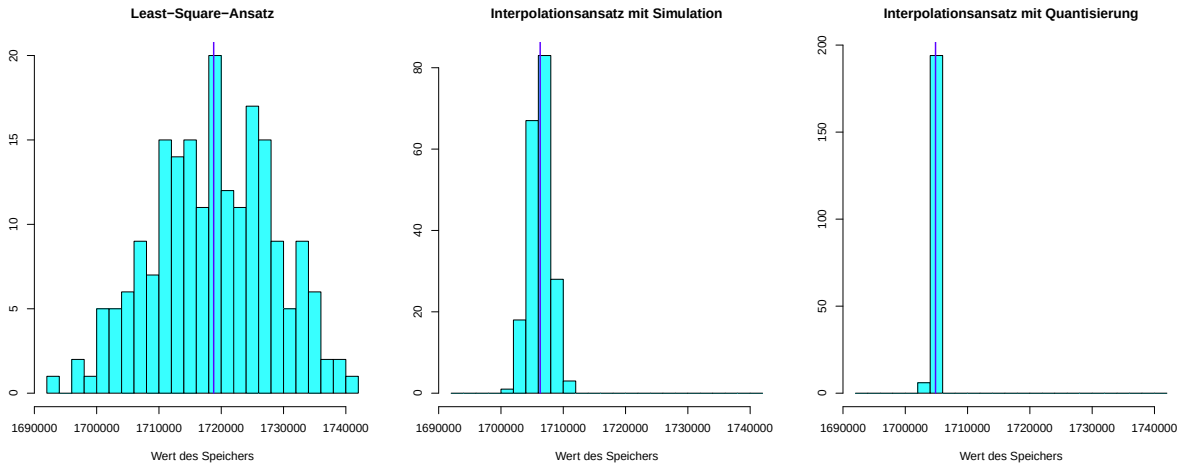


Abbildung 7.23.: Histogramme des Gasspeicherwertes drei verschiedener Monte Carlo Algorithmen mit $M = 1000$ simulierten Pfaden

7.3.2. Weitere Vergleiche: Laufzeitvergleiche und Vergleich des Einflusses einiger Preisparameter

Der Vergleich zwischen den Algorithmen wird in diesem Teilkapitel weiter ausgeführt und auf einen Vergleich mit dem Binomialbaum-Algorithmus erweitert.

Laufzeitvergleiche

Vergleicht man die Laufzeiten verschiedener Algorithmen, ist zu beachten, dass alle Algorithmen möglichst optimal implementiert und Unterschiede analysiert werden. Alle Algorithmen werden primär in R (Version 3.0.2) [30] implementiert und auf einem 64-Bit Windows-System mit einem Intel Core 2 Extreme X9650 Prozessor und 8GB DDR2 RAM ausgeführt. Einige für alle Algorithmen notwendige `for`-Schleifen, die erfahrungsgemäß in R relativ langsam ablaufen, werden dann mit C realisiert [17] und in den R-Code mittels einer sogenannten R Extension [31] eingebunden. Zusätzlich wird darauf geachtet, die Vorteile von R zu nutzen und möglichst umfassend mit Vektoren und Matrizen zu arbeiten. Speziell im Least-Square-Algorithmus wird außerdem die Berechnung des bedingten Erwartungswertes $U_n(x, p)$ in C ausgelagert, da mit Hilfe des LAPACK Paketes [1] die kleinste Quadrate Methode für alle x beziehungsweise Speicherstandgitterpunkte gleichzeitig durchgeführt werden kann. Dies bringt dem Least-Square-Algorithmus einen eindeutigen Laufzeitvorteil. Dennoch ist hier zu bemerken, dass alle Algorithmen noch weiter optimiert werden können, zum Beispiel durch Parallelisierung. Obwohl der Prozessor 4 Kerne besitzt, nutzen alle Algorithmen in dieser Arbeit nur einen Kern.

Tabelle 7.4 zeigt unter anderem die Laufzeiten der verschiedenen Algorithmen mit bereits gegebenen Preissimulationen bzw. generiertem Binomialbaum. Die Laufzeiten der Preisgittergenerierung zeigt Tabelle 7.3, diese müssten jeweils zu den Algorithmen hinzugerechnet werden. Es ist jedoch zu erwähnen, dass gerade im Fall der Monte-Carlo-Simulation sequentiell gearbeitet wird, obwohl hier die einzelnen Pfade auch parallel

berechnet werden können. Daher wird diese Zeit für den Vergleich nicht mit einbezogen.

Betrachtet man Tabelle 7.4 so erkennt man, dass der Binomialbaum-Algorithmus die kürzesten Laufzeiten ergibt und selbst für $\Delta t = 1/5$ mit 1.16 Minuten weit schneller ist als der schnellste Interpolations-Algorithmus mit 6.09 Minuten. Zudem gibt der Binomialbaum-Algorithmus, letztendlich auf dem Approximationsgitter den „exakten" Wert des Speichers an. Das bedeutet auch, dass hier lediglich ein Durchlauf des Algorithmus nötig ist, wohingegen die Monte Carlo Ansätze Schätzwerte liefern und daher mehrere Durchführungen empfehlenswert sind. Man erkennt auch, dass gerade im Vergleich zum Least-Square-Ansatz, der Wert des Speichers nahe am Mittelwert des Least-Square-Schätzers liegt. Dennoch wäre hier zu überlegen, ob eine noch feinere Wahl des Approximationsgitters, das heißt ein kleineres Δt, zu weiteren Verbesserungen im Wert führt. Allerdings ist der Aufwand zu berücksichtigen, der bei der Berechnung der Wahrscheinlichkeiten der möglichen Preiswerte im Zeitschritt t auf $t+1$ entsteht, da sich die Up-Down-Wahrscheinlichkeiten in der Binomialapproximation unter anderem mit der Zeit ändern.

Betrachtet man die Laufzeiten der Monte Carlo Algorithmen so fällt auf, dass die Interpolationsansätze bei $M = 1000$ mit ungefähr 34 Minuten langsamer sind, als der Least-Square-Ansatz mit etwa 15 Minuten. Vergleicht man jedoch zusätzlich das Verhalten des Schätzer mittels Mittelwert und Standardabweichung, so wird deutlich, dass die Interpolationsansätze durchaus mit $M = 500$ oder sogar $M = 200$ „auskommen" und damit mit Laufzeiten von etwa 15 Minuten bzw. etwa 6.5 Minuten. Die Relevanz der Laufzeiten steht im Zusammenhang mit der betrieblichen Entscheidungszeit. Abbildung 7.24 verdeutlicht, dass mit einem Interpolationsansatz und $M = 200$ sich ein vergleichbar guter Schätzer wie mit einem Least-Square-Ansatz und $M = 1000$ ergibt. Die Abbildung zeigt Histogramme der Least-Square-Schätzwerte (links), sowie der Schätzwerte aus dem linearen Interpolationsansatz mit Simulation (mitte) und Quantisierung (rechts).

		Laufzeit in Sekunden
Binomial-baum-generierung	$\Delta t = 1$	0.17
	$\Delta t = 1/2$	0.59
	$\Delta t = 1/3$	1.31
	$\Delta t = 1/4$	2.36
	$\Delta t = 1/5$	3.68
Monte Carlo Simulation	$M = 200$	19.11
	$M = 500$	47.69
	$M = 1000$	95.09

Tabelle 7.3.: Vergleich der Laufzeit der Monte-Carlo-Simulationen und der Binomialbaumgenerierung

		Laufzeit in Sekunden	(Mittel-)Wert in £	Standard-abweichung
Baum	$\Delta t = 1$	14.56	1744174	-
	$\Delta t = 1/2$	28.27	1726241	-
	$\Delta t = 1/3$	41.79	1723980	-
	$\Delta t = 1/4$	55.37	1719799	-
	$\Delta t = 1/5$	69.63	1719969	-
Least-Square	$M = 1000$	890.75	1718785	9611.739
Interpolation linear Simulation	$M = 200$	394.41	1706773	3855.69
	$M = 500$	967.81	1706127	2293.382
	$M = 1000$	1970.8	1706294	1740.8
Interpolation linear Quantisierung	$M = 200$	400.94	1704608	2185.568
	$M = 500$	982.24	1704713	794.143
	$M = 1000$	2000.41	1704856	408.7203
Interpolation Spline Quantisierung	$M = 200$	365.45	1703193	398.7616
	$M = 500$	934.23	1704531	110.0713
	$M = 1000$	2025.26	1704859	43.47782

Tabelle 7.4.: Vergleich der Algorithmen in Wert und Laufzeiten

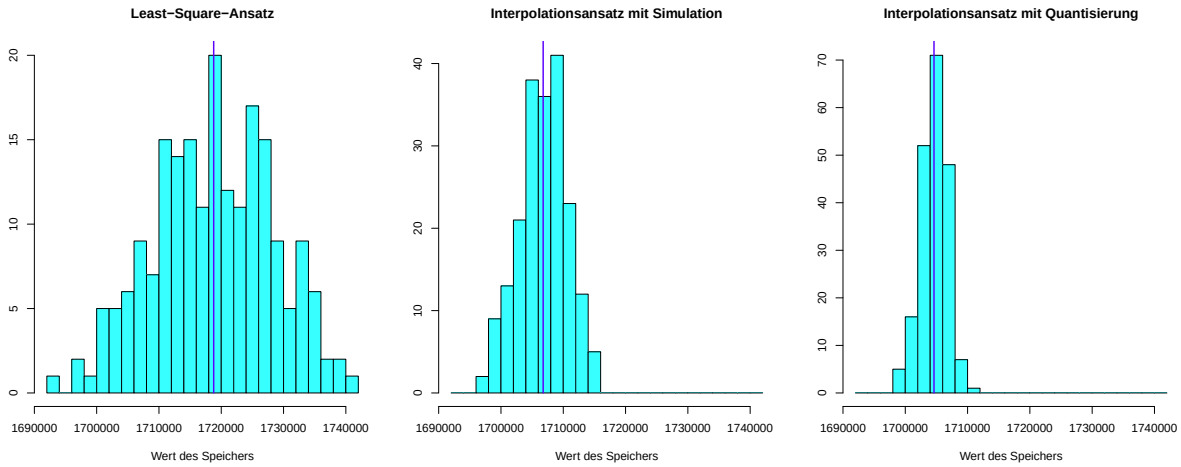


Abbildung 7.24.: Histogramme des Gasspeicherwertes im Least-Square-Ansatz mit $M = 1000$ im Interpolationsansatz mit $M = 200$

Einfluss der Preisparameter ausgewählter Algorithmen

Die folgenden Abbildungen zeigen den Einfluss der Preisparameter α und σ. In jeder Abbildung ist der Wert des Speichers mit Binomialbaum-Algorithmus blau eingezeichnet, der Wert mit Least-Square-Algorithmus rot und der Wert mit linearem Interpolations-Algorithmus mit Quantisierung grün. Die Werte des Binomialbaum-Algorithmus werden mit $\Delta t = 1/4$ berechnet. Die Werte der Monte Carlo Algorithmen werden mit jeweils 1000 simulierten Pfaden und gesetztem seed, das heißt mit gleichen Ausgangssimulationen, ermittelt.

Der Einfluss der Preisparameter wird zum einen „grob“ untersucht (jeweils linkes Bild), um im Überblick zu sehen, wie sich die Abhängigkeit des Gasspeicherwertes von einem Preisparameter darstellt. Zum anderen wird dies im Detail (rechtes Bild) betrachtet, um unter anderem das Ausmaß des Einflusses möglicher Schätzfehler in den Parametern zu erkennen. Die Werte in den linken Bildern werden in 0.01-Schritten, die Werte der rechten Schaubilder in 0.001-Schritten berechnet. Wird der Parameter α verändert, so ist σ konstant auf dem Ausgangswert, wie in den vorherigen Beispielen, gesetzt, und umgekehrt. Insgesamt weisen die Algorithmen ein ähnliches Verhalten in den Preisparametern auf.

In Abbildung 7.25 ist festzustellen, dass die Kurven zum Binomialbaum-Algorithmus und zum Interpolationsansatz nahezu parallel verlaufen, während die Kurve des Least-Square-Algorithmus leicht nach vorne versetzt ist. Dies hat zu Folge, dass gerade im Bereich des im bisherigen Beispiel verwendeten $\alpha = 0.073$ diese Kurve wesentlich „flacher“ ist. In diesem Bereich ist der Einfluss des Parameter α folglich geringer. Der mean reversion Faktor hat letztendlich einen großen Einfluss auf die „Vorhersagbarkeit“ des Prozesses. Erhöht man α so erhöht sich am Anfang der Wert des Speichers. Allerdings wird dann ein Extremwert erreicht, ab dem der Wert des Speichers wieder sinkt. Der Einfluss des mean reversion Faktors wird dann zu hoch, als dass durch Preisfluktionen zusätzliche Gewinnmöglichkeiten entstehen können; der Preisprozess wird sozusagen zu stark zum zeitabhängigen Mean wieder zurückgezogen. Diese Beobachtung machen auch Boogert u. de Jong [5] mit dem Least-Square-Ansatz.

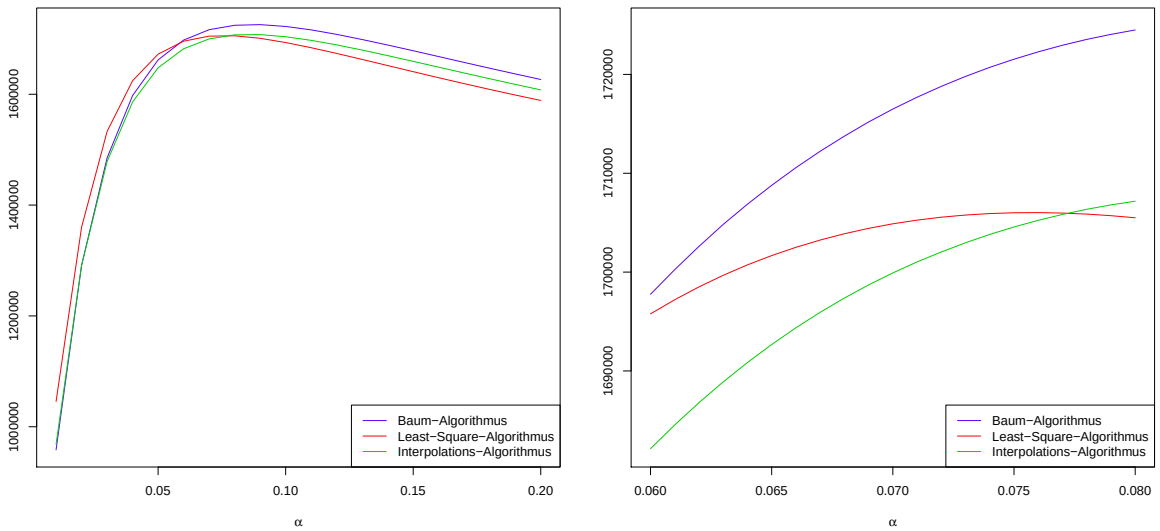


Abbildung 7.25.: Einfluss des mean reverting Faktors α auf den Speicherwert

Der Einfluss der Volatilität σ ist in Abbildung 7.26 dargestellt und ist exponentiell. Hierbei verlaufen die Kurven der drei betrachteten Methoden nahezu parallel, es sind keine großen Unterschiede zu erkennen. Je höher demnach die Volatilität ist, desto größer sind die Schwankungen im Preis und die Chance größeren Gewinn zu erzielen. Damit steigt der Gasspeicherwert.

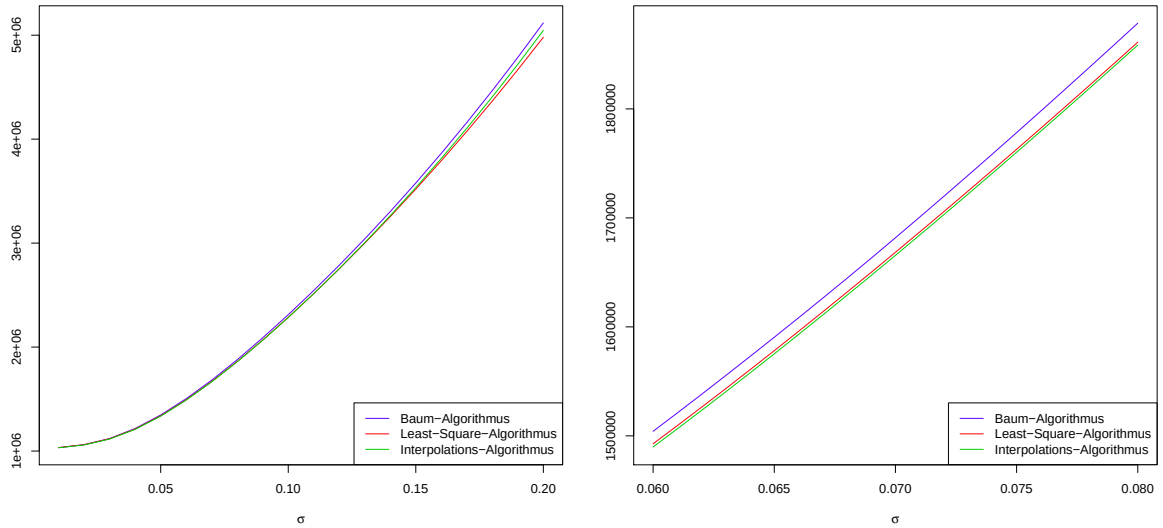


Abbildung 7.26.: Einfluss der Volatiliät σ auf den Speicherwert

Zum Abschluss des Abschnitts über den Einfluss der Preisparameter zeigt Abbildung 7.27 Werte, die mit zeitabhängiger Volatilität entstanden sind. Hier wurde eine von 0.072 ausgehend schwankende Volatilität angenommen, deren Amplitude mit γ bezeichnet wird. Die gestrichelten Linien zeigen jeweils die Werte mit konstanter Volatilität. Man erkennt, wenn die Volatilität zusätzlich zeitlich schwankt, dass der Wert des Gasspeichers mit der Amplitude ansteigt. Bemerkenswert ist, dass der Einfluss der Amplitude auf den Baum-Algorithmus geringer ist.

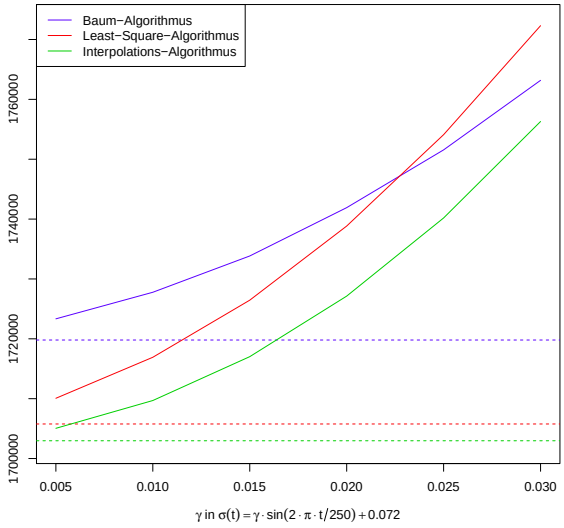


Abbildung 7.27.: Speicherwert mit zeitabhängiger Volatiliät $\sigma(t) = \gamma \sin\left(\frac{2\pi t}{250} + 0.072\right)$ und verschiedenen γ

7.3.3. Strategieschranken und Gasspeicherentwicklung

In diesem Teilkapitel werden zum einen die Strategieschranken $\underline{b}_n$ und $\overline{b}_n$ betrachtet, welche die optimale Politik bestimmenden, zum anderen die Gasmengen- bzw. Speicherstandentwicklung im Speicher bei gegebenem Preispfad unter Verwendung der optimalen Politik. Zuletzt folgt ein Vergleich zu einer anderen - nicht optimalen - Politik.

Die optimale Poltik gemäß Satz 4.3.4 ist gegeben durch

$$f_n^\star(x,p,r) = \begin{cases} \min(\underline{b}_n(p) - x, i_n^{\max}(x)), & b^{\min} \leq x < \underline{b}_n(p), \\ 0, & \underline{b}_n(p) \leq x \leq \overline{b}_n(p), \\ \max(\overline{b}_n(p) - x, i_n^{\min}(x)), & \overline{b}_n(p) < x \leq b^{\max}. \end{cases}$$

Abbildungen 7.28, 7.29 und 7.30 zeigen die mit Hilfe des Binomialbaum-Algorithmus, des Least-Square-Ansatzes und des (linearen) Interpolationsansatzes mit Quantisierung berechneten unteren und oberen Strategieschranken $\underline{b}_n(p)$ und $\overline{b}_n(p)$ für jeden Zeitpunkt und jeden Preis im jeweiligen Gitter. Die Schranken wurden im Binomialbaum-Algorithmus im Fall $\Delta t = 1/4$ berechnet, in den Monte Carlo Algorithmen mit $M = 1000$. In den Abbildungen ist die untere Schranke hellblau und die obere Schranke dunkelbau dargestellt. Gemäß der optimalen Poltik würde man also bei einem Gasspeicherstand unterhalb der unteren Schranke versuchen bis zur unteren Schranke Gas in den Speicher einzufüllen oder zumindest $i^{\max}(x)$. Oberhalb der oberen Grenze würde man Gas entnehmen; möglichst bis zu dieser Schranke, ansonsten $i^{\min}(x)$ und zwischen den Schranken würde der Füllstand nicht verändert.

Zwei wichtige Spezialfälle der Schranken sind $\underline{b}_n(p) = \overline{b}_n(p) = b^{\max}$ und $\underline{b}_n(p) = \overline{b}_n(p) = b^{\min}$. Diese Fälle bedeuten, dass unabhängig wie voll der Gasspeicher derzeit ist, die optimale Strategie bei Preis p ist „alles, was möglich ist, einzuspeichern“ bzw. „alles was möglich ist, auszuspeichern“.

In allen drei Abbildungen ist zu erkennen, dass diese Spezialfälle einen großen Teil des Preisgitters einnehmen und sich die anderen Fälle auf ein „Band“ um den zeitabhängigen Mean erstrecken. Es bildet sich ein Übergangsbereich von dem Bereich des „Nur-Einspeicherns“ bei niedrigem Preis und dem Bereich des „Nur-Ausspeicherns“ bei hohem Preis. Um einen besseren Eindruck dieser Region zu bekommen, zeigen die Abbildungen auch Darstellungen auf einer anderen Preisskala. Jede Grafik „zoomt“ weiter in die vorhergehende Abbildung hinein. Es fällt auf, dass in 7.28 die Skala im ersten Teilbild größer ist als in den Abbildungen 7.29 und 7.30 und auch nicht so weit „hineingezoomt“ wird. Dies liegt daran, dass das Preisgitter im Binomialbaum einerseits alle Pfade mit Wahrscheinlichkeit größer Null enthält und das Preisgitter in den Monte Carlo Algorithmen aus simulierten Pfaden besteht. Zum anderen hat das Gitter im Binomialbaum in der Mitte der Preisskala - gerade um den zeitabhängigen Mean - weniger Gitterpunkte. Um bei den Monte Carlo Algorithmen bessere Übersichtlichkeit in den Abbildungen zu bekommen, werden in den drei Bildern mit kleinerer Preisskala nur die Schranken der ersten 100 Pfade eingezeichnet.

Das Entstehen des Bereiches, in dem „nicht handeln“ optimal ist, geht auf den Unterschied in Einkaufspreis $k(p)$ und Verkaufspreis $e(p)$ zurück. Denn dadurch unterscheiden sich die Optimierungsprobleme, die zu den Strategieschranken führen. Das heißt aber auch, dass dieser Unterschied einen großen Einfluss auf die Bildung und das Aussehen des Übergangsbereiches um den zeitabhängigen Mean hat. Es entsteht - vergleiche die Abbildungen 7.28, 7.29 und 7.30 - eine Art „Wanne“ für beide Schranken um den Mean. Im Preis, zu festem Zeitpunkt sind die Schranken demnach fallend. Dieses Ergebnis konnte bereits unter gewissen Voraussetzungen analytisch in Kapitel 4.3 gezeigt werden und ist hier ebenso der Fall.

Eine Ausnahme bilden die Schranken gegen den Endzeitpunkt hin. Beeinflusst durch die terminale Gewinnfunktion, sollte am Ende möglichst wieder der Anfangszustand erreicht werden.

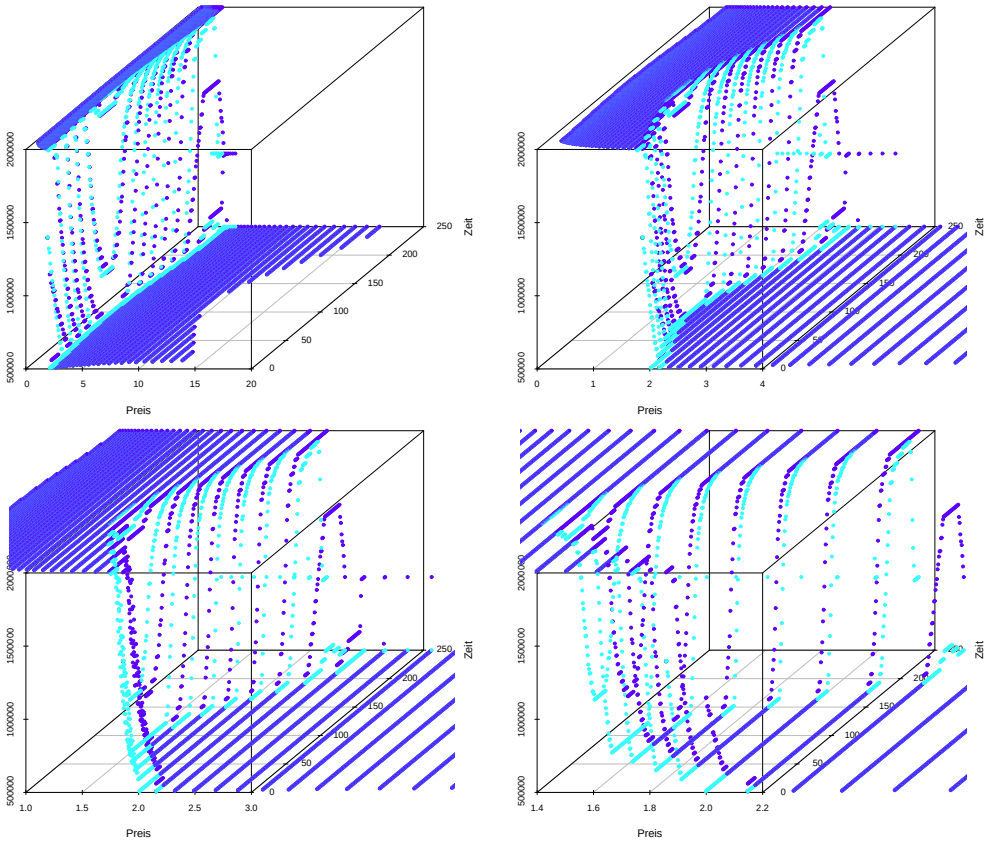


Abbildung 7.28.: Strategieschranken $\underline{b}_n(p)$ (hellblau) und $\overline{b}_n(p)$ (dunkelblau) im Binomialbaum-Algorithmus

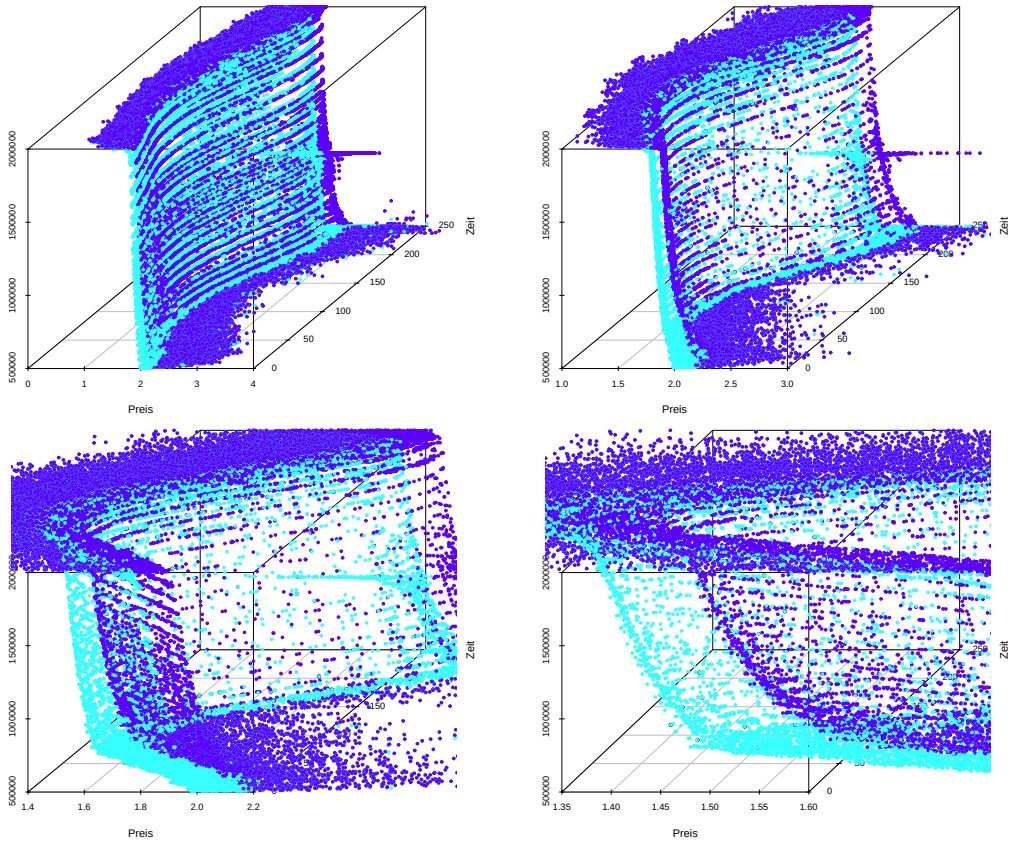


Abbildung 7.29.: Strategieschranken $\underline{b}_n(p)$ (hellblau) und $\overline{b}_n(p)$ (dunkelblau) im Least-Square-Algorithmus

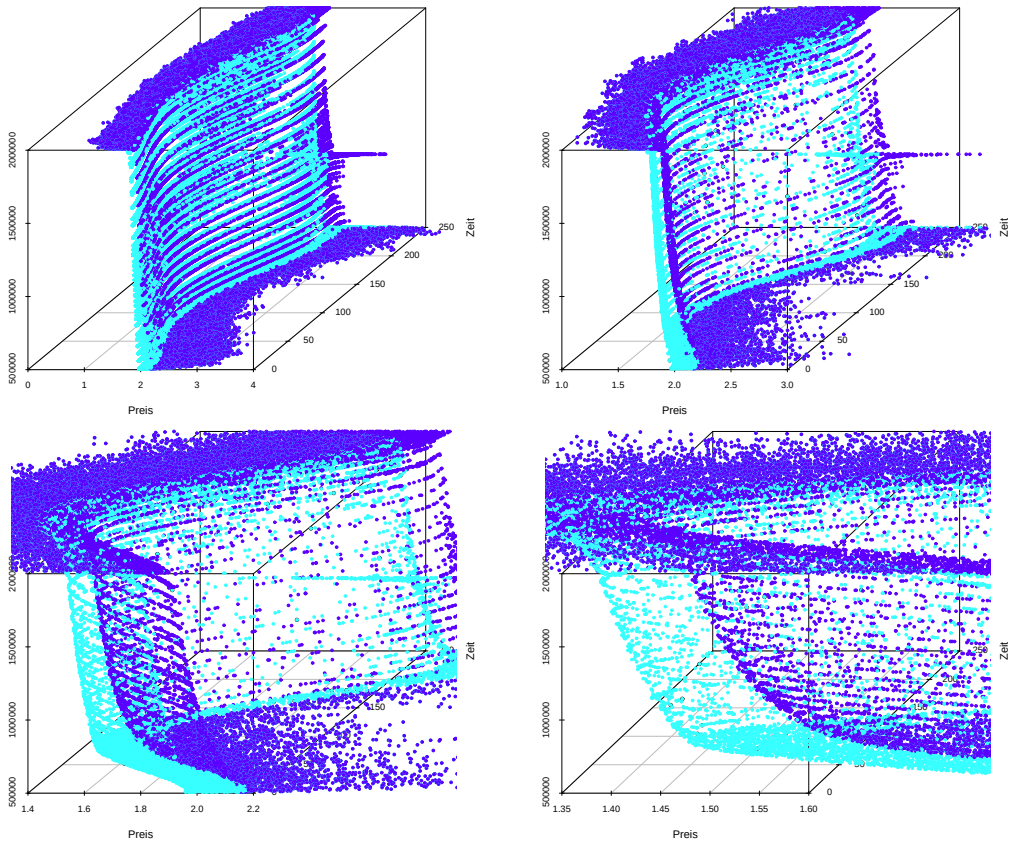


Abbildung 7.30.: Strategieschranken $\underline{b}_n(p)$ (hellblau) und $\overline{b}_n(p)$ (dunkelblau) im Interpolations-Algorithmus

Wendet man dann die optimale Strategie mit den errechneten Schranken auf einen simulierten Preispfad an, so kann man die Entwicklung des Speicherstandes - der Gasmenge im Speicher - über die Zeit beobachten.

Abbildungen 7.31, 7.32 und 7.33 zeigen den jeweiligen Speicherstandverlauf zu den drei im oberen Bild gezeigten simulierten Pfaden, gekennzeichnet durch die verschiedenen Farben.

Der nahezu identische Verlauf der Abbildungen 7.32 und 7.33 ist darauf zurückzuführen, dass die selben simulierten Pfade verwendet werden.

Aufgrund der saisonalen Eigenschaften, repräsentiert durch den zeitabhängigen mean, werden zum Sommer hin die niedrigen Preise erwartet. Daher ist zu vermuten, dass man vor der Sommerzeit versucht, den Speicher zu leeren, um ihn dann, wenn die Preise möglichst niedrig sind, wieder zu befüllen. Zu einem späteren Zeitpunkt im Jahr, wenn die Preise steigen wird dann wieder tendenziell eher geleert. Es ist zu beachten, dass Zeitpunkt 0 Anfang Februar entspricht. Dennoch wird nicht darauf verzichtet durch größere Abweichungen vom Mean einen zusätzlichen Gewinn zu erzielen, indem eher azyklisch gehandelt wird. Tatsächlich führt eine höhere Volatilität im Preispfad eher zu mehr „Handelsspielraum“ und zu höheren Gewinnen, vergleiche hierzu Kapitel 7.3.2, Einfluss der Preisparameter.

Vergleicht man die grün- und rot-markierte Kurve des Preises bzw. der Speicherstandsentwicklung in Abbildung 7.31, so wird beispielsweise in der grünen Kurve die kleine Preisspitze nach Zeitpunkt 50 ausgenutzt und der Speicher früher geleert als bei der roten Kurve, bei der man die recht niedrigen Preise ausnutzt, um noch einmal den Speicher zu füllen bevor er geleert wird. Genauso setzen bei der roten Kurve die niedrigen Preise erst später ein als bei der grünen, sodass der Speicherverlauf der grünen Kurve früher ansteigt.
Vergleicht man die grün- und rot-markierte Kurve der Speicherstandsentwicklung in Abbildung 7.32 bzw. Abbildung 7.33, so erkennt man, dass diese Kurven ab Zeitpunkt 70 fast gegenläufig verlaufen. Die tiefen Preise in der zugehörigen roten Preis-Kurve sind bereits zum Zeitpunkt (etwa) 80 erreicht und werden sofort genutzt, um Gas zu kaufen, das beim Anstieg um Zeitpunkt 150 wieder verkauft wird. Die grüne Kurve erreicht den Zustand des leeren Speichers zwar ebenfalls recht früh, allerdings erst ab dem Preistiefpunkt bei 150 wird wieder Gas eingekauft, bis der Speicher (fast) voll ist - abgesehen von einer kleinen „Zwischenbefüllung“ um 120.

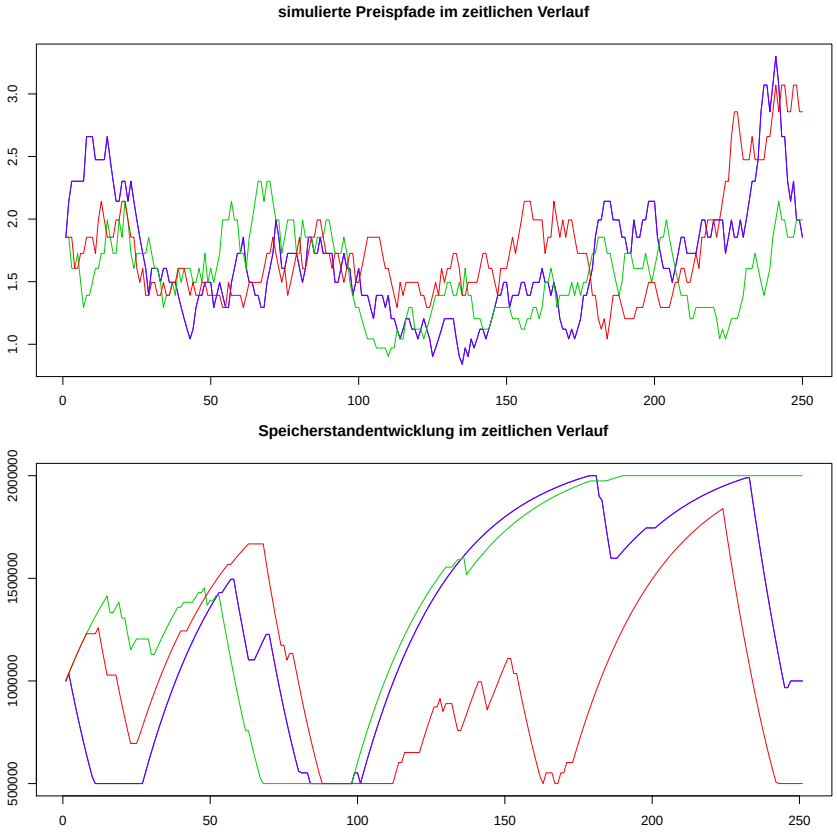


Abbildung 7.31.: 3 simulierte Pfade und der zugehörige zeitliche Speicherverlauf im Binomialbaum-Algorithmus

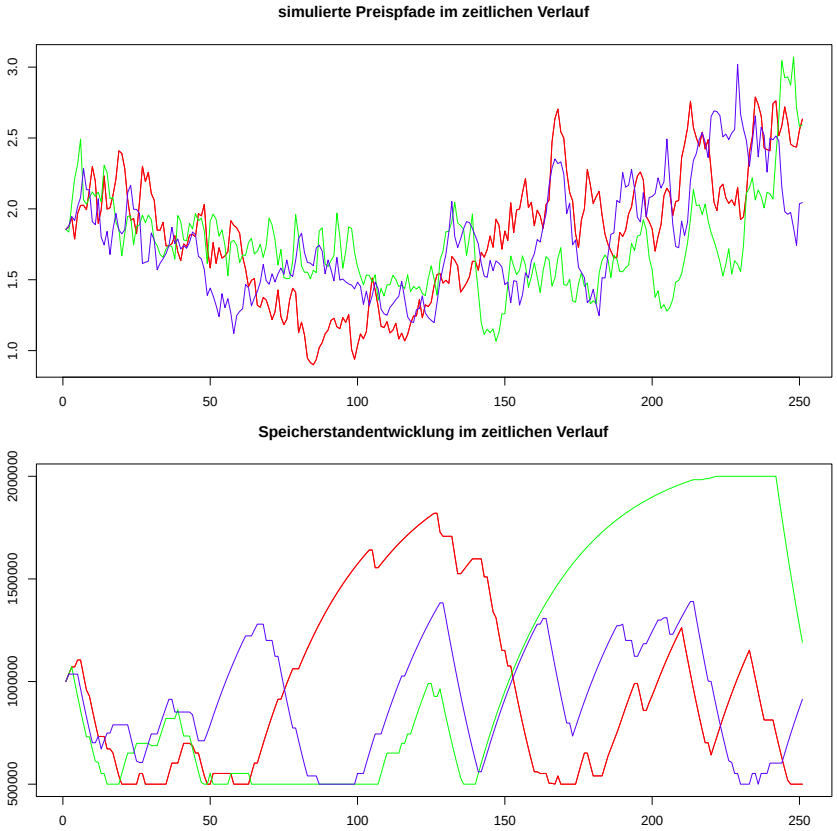


Abbildung 7.32.: 3 simulierte Pfade und der zugehörige zeitliche Speicherverlauf im Least-Square-Algorithmus

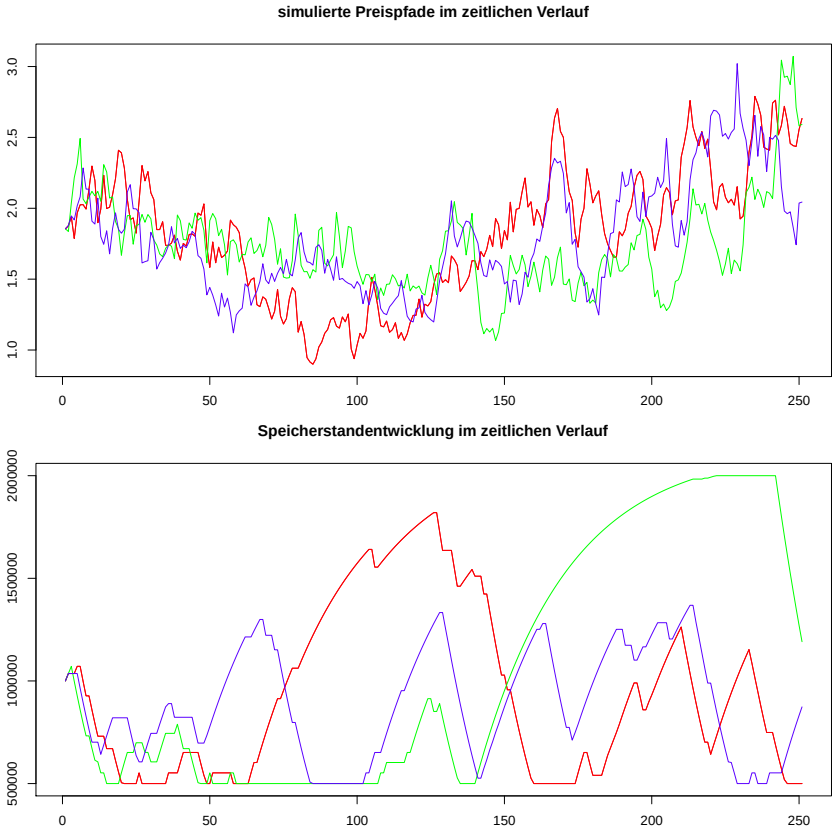


Abbildung 7.33.: 3 simulierte Pfade und der zugehörige zeitliche Speicherverlauf im Interpolations-Algorithmus

Vergleich der optimalen Strategie zu einer bang-bang-Strategie in einigen Beispielen

Mit einem Blick auf die Strategieschranken und die optimale Politik, stellt sich die Frage, inwieweit die Strategieschranken einen Einfluss auf den Gasspeicherwert haben. Oft ist es der Fall, dass die optimale Strategie $i^{\max}(x)$, $i^{\min}(x)$ oder 0 ist. Es stellt sich das Problem, in welchem Ausmaß sich demnach die optimale Politik letztendlich von der bang-bang-Strategie, entweder „$i^{\max}(x)$ einzuspeichern“, „$i^{\min}(x)$ zu entnehmen“ oder „nicht zu handeln“ unterscheidet.

Hierzu zeigt Tabelle 7.5 Gasspeicherwerte, die mit der optimalen Politik $\pi^\star$ erzeugt werden, und Gasspeicherwerte, die mit der bang-bang-Strategie π° erzeugt werden. Zur Berechnung werden jeweils die selben Monte-Carlo-Simulationen zu Grunde gelegt. Es wird außerdem der Interpolations-Algorithmus mit Quantisierung und linearer Interpolation verwendet. Das Ergebnis zeigt, dass alle Algorithmen für die suboptimale Strategie zwar kleinere Werte liefern, der Unterschied zum Wert mit optimaler Strategie ist jedoch relativ gering. Die Differenz des Wertes mit π° zum Wert mit $\pi^\star$ liegt bei etwa 0.03% des Wertes mit $\pi^\star$.

Methode		Gasspeicherwert mit $\pi^\star$	Gasspeicherwert mit π°	Unterschied in Prozent
Baum-Algorithmus	mit $\Delta t = 1/4$	1719799	1719264	0.031
Least-Square-Algorithmus	mit $M = 1000$	1713924	1713307	0.036
Interpolations-Algorithmus	mit $M = 1000$	1705569	1705073	0.029
	mit $M = 500$	1705110	1704611	0.029
	mit $M = 200$	1704053	1703547	0.030

Tabelle 7.5.: Vergleichswerte der optimalen Strategie mit Werten einer bang-bang-Strategie

Da die Werte der beiden Strategien nahe beieinander liegen, gilt es zu untersuchen, in welchem Umfang die Strategieschranken eingesetzt werden. Das heißt wie oft die optimale Entscheidung $\underline{b}_n(p)$ oder $\overline{b}_n(p)$ ist und nicht $i^{\max}$, $i^{\min}$ oder 0.

Tabelle 7.6 zeigt in den drei Beispielpfaden aus Abbildung 7.32 und 7.33 die jeweilige Häufigkeit der optimalen Entscheidungen im Least-Square-Algorithmus und im Interpolations-Algorithmus. Hier wird deutlich, dass die optimale Entscheidung selten den Strategieschranken entspricht.

		optimale Entscheidung				
Methode	Pfad	$\underline{b}_n(p)$	$\overline{b}_n(p)$	$i^{\max}$	$i^{\min}$	0
Least-Square-Algorithmus	rot	1	8	108	58	75
	grün	0	3	102	35	110
	blau	3	5	108	60	74
Interpolations-Algorithmus	rot	0	1	106	59	84
	grün	0	5	100	32	113
	blau	1	3	106	61	79

Tabelle 7.6.: Absolute Häufigkeiten der genutzen optimalen Entscheidungen in drei Beispielpfaden

Insgesamt bedeutet das in den aufgeführten Beispielen, dass der Fehler, der entsteht, wenn statt der optimalen Strategie eine bang-bang-Strategie gewählt wird, relativ gering ist. Oft wird dies als eine intuitive Vorgehensweise interpretiert. Dennoch sollte bewusst werden, dass der Fehler auftritt.

7.4. Obere Schranken für den Gasspeicherwert

Im Folgenden werden die Algorithmen zur Ermittlung von Schätzern für obere Schranken des Gasspeicherwertes genauer untersucht.

Tabelle 7.7 zeigt jeweils 5 Schätzwerte des Schätzers ohne Penalitzation, die mit $M = 500$ beziehungsweise $M = 1000$ Pfade entstanden sind. Im Vergleich zu den errechneten Werten aus den vorigen Kapiteln, die in etwa 1.71 Millionen Pfund betragen, ist die obere Schranke mit etwa 600000 £ über dem Wert relativ groß. Die Frage wäre hier in wieweit M noch zu erhöhen ist, um genauere, das heißt geringere, Werte zu erreichen. Dies wiederum führt jedoch zu einem Laufzeitproblem, obwohl festzustellen ist, dass hier großes Parallelisierungspotential besteht, da die Pfade sich nicht gegenseitig beeinflussen und erst am Ende gemittelt wird. Immerhin scheint dieser Algorithmus eine obere Schranke zu liefern.

$M = 500$	$M = 1000$
2380906.303	2395980.943
2400837.637	2386371.556
2365333.972	2360948.632
2387637.988	2401529.146
2409758.358	2402468.452

Tabelle 7.7.: Einige Schätzwerte der oberen Schranke ohne penalty

Tabelle 7.8 zeigt einige Werte der oberen Schranke mit Penalization mit verschiedenen zugrundeliegenden approximierenden Wertefunktionen. Es bezeichnet $\widetilde{M}$ die Anzahl der Pfade, die zur Berechnung der approximierenden Wertefunktion verwendet werden. Zum Vergleich werden hier einmal der Least-Square-Algorithmus und einmal der (lineare) Interpolationsansatz mit Simulation gewählt. Im Interpolationsansatz werden 50 Simulationen zur Berechnung des bedingten Erwartungswertes verwendet. Für die Berechnung des Schätzwertes der oberen Schranke werden jeweils 500 Simulationspfade verwendet. Zunächst ist festzustellen, dass die gelb markierten Schätzer der oberen Schranke kleiner als ihr Referenzwert sind. Betrachtet man die in Teilkapitel 7.3 berechneten Standardabweichungen der Schätzer, stellt sich außerdem die Frage, ob die Werte, insbesondere bei höherem $\widetilde{M}$, für eine obere Schranke nicht zu gering sind. Intuitiv würde man vermuten, dass je besser die approximierende Funktion ist, desto besser auch der Schätzer der oberen Schranke. Jedoch muss man beachten, dass im Algorithmus selbst und in der Implementierung einige weitere Approximationen festgelegt werden; angefangen von der linearen Interpolation der approximierenden Funktion - da diese nur auf einem Gitter vorhanden ist - bis hin zur Schätzung des Erwartungswertes durch Quantisierung. Daher sind die Ergebnisse in der Praxis kritisch zu bewerten.

$\widetilde{M}$	Wert der approximierende Wertefunktion	Schätzer obere Schranke
50	1707636.477	1770375.976
		1765294.527
		1781960.3015
		1773910.0198
		1775303.304
100	1735484.100	1727417.278
		1721233.775
		1726269.259
		1730833.317
		1730359.961
200	1682763.511	1718891.081
		1715932.0299
		1717961.522
		1723233.331
		1719763.001
500	1696027.820	1709933.470
		1709141.244
		1707543.776
		1714721.015
		1712171.363

$\widetilde{M}$	Wert der approximierende Wertefunktion	Schätzer obere Schranke
50	1719168.093	1756523.656
		1748466.021
		1761997.655
		1757511.582
		1757741.358
100	1713488.601	1730095.210
		1717929.253
		1730212.674
		1729551.901
		1730440.283
200	1704660.666	1720620.724
		1715632.224
		1724897.083
		1718172.192
		1725055.220
500	1712693.772	1717243.530
		1721977.409
		1718897.946
		1719165.017
		1720485.509

Tabelle 7.8.: Einige Schätzwerte der oberen Schranke mit penalty bei zugrundeliegender approximierenden Wertefunktion durch Least-Square-Algorithmus (links) und durch Interpolations-Algorithmus (rechts)

Um dennoch einen Eindruck des Verhaltens des Schätzer der oberen Schranke zu bekommen, zeigt Abbildung 7.34 Histogramme der Methode ohne Penalization (links), mit Penalization und Least-Square-Grundlage (mitte) und mit Penalization und Interpolations-Grundlage (rechts). Die beiden Abbildungen mit Penalization sind auf der gleichen Skala gezeichnet, die Referenzwerte der approximierenden Wertefunktionen betragen 1707636 £ für die Least-Square-Grundlage beziehungsweise 1706998 £ für die Interpolationsgrundlage. Beide approximierenden Wertefunktionen entstanden mit $\widetilde{M} = 50$. Die je 100 Schätzwerte der oberen Schranke wurden auf einer Grundlage von $M = 500$ berechnet. Die Mittelwerte sind ebenfalls dargestellt (dunkelblau). In Varianz und Ab-

stand des Mittelwerts zum Referenzwert weisen beide Methoden mit Penalization ähnliches Verhalten auf. Die Schätzwerte ohne Penalization weisen, wie schon in Tabelle 7.7 gesehen, einen Mittelwert auf, der weit über den Referenzwerten aus Kapitel 7.3 liegt.

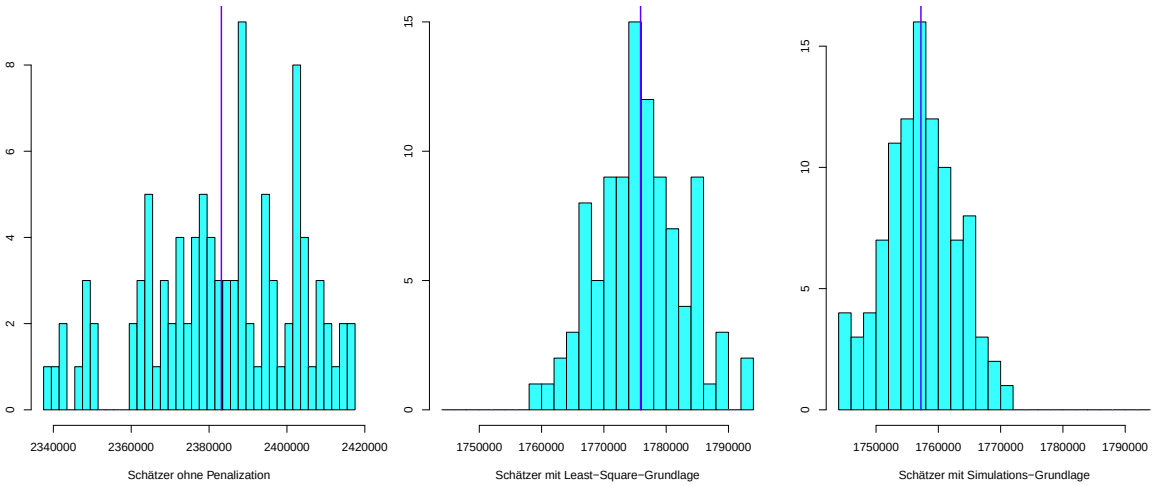


Abbildung 7.34.: Histogramme zur Oberen Schranke mit $M = 500$ zugrundeliegenden Pfaden

8 Abschließende Betrachtung

In der vorliegenden Arbeit wird das Gasspeicherproblem auf seine strukturellen Eigenschaften untersucht, sowie ausgewählte Algorithmen zur Bewertung thematisiert.

Hierzu wird das Problem als zeitdiskreter Markovscher Entscheidungsprozess mit endlichem Horizont formuliert. Dabei wird dem Hauptmodell die Annahme zu Grunde gelegt, dass die Einspeicherrate des Gasspeichers eine konkave Funktion und die Ausspeicherrate eine konvexe Funktion der aktuellen Gasmenge im Speicher darstellt.

In dem vorgestellten Modell lässt sich unter den definierten Voraussetzungen an dem zugrundeliegenden Preisprozess und den Endbedingungen des Speichers eine Schrankenfunktion angeben, die sicherstellt, dass die geforderte Integrabilitätsannahme erfüllt ist. Mit Hilfe der Theorie der Markovschen Entscheidungsprozesse ist es möglich, entsprechend den definierten Voraussetzungen, eine Stetigkeits-, eine Monotonie- und eine Konkavitätsaussage zu treffen, sodass die Strukturannahme gilt und das Problem als „wohlgestellt“ angesehen werden kann. Das bedeutet, dass die Existenz der Maximisatoren des Optimierungsproblems und damit eine optimale Strategie existieren, das Problem also lösbar ist. Insbesondere die Konkavität der Wertfunktion im Speicherstand, das heißt der Gasmenge im Speicher, ist für die allgemeine Struktur der optimalen Politik von entscheidender Bedeutung.

Werden die potentiellen Entscheidungen dahingehend eingeschränkt, dass zu jedem Zeitpunkt die Möglichkeit besteht, von einem gefüllten Speicher zu einem leeren Speicher zu wechseln oder umgekehrt, so lässt sich das Problem als optimales switching Problem formulieren. Daraus ergibt sich eine genauere Untersuchung der Strategie, dahingehend, dass es kritische Preise gibt, die je nach aktuellem möglichen Wechsel angeben, ob dieser optimal ist oder nicht. Es lassen sich Beispiele angeben, die die Voraussetzungen für diese Darstellung der optimalen Politik erfüllen.

Ist es nun nur noch möglich - nicht zwingend - den Speicher täglich zu befüllen oder zu entleeren, so wird dies als ein „schneller“ Gasspeicher bezeichnet. Wird in diesem Fall eine spezielle Struktur eines Markov-Prozess betrachtet, die tendenziell fallende Preise ausweist, lässt sich die Wertfunktion explizit angeben und die aufgestellte These, dass

es optimal ist, zu Beginn der Nutzung den Speicher zu leeren, verifizieren. Bei tendenziell steigenden Preisen und der zusätzlichen Bedingung, dass Verkauf- und Einkaufpreis gleich sind, trifft eine vergleichbare These zu. Mit dem verallgemeinerten Markov-Prozess lässt sich ebenfalls die Affinität in der Speicherstand-Variablen der Wertfunktion zeigen. Die optimale Strategie ist hier von einem speziellen Typus, dass es immer optimal ist, entweder den Speicher zu füllen, zu leeren oder nicht zu handeln. Diese Struktur gilt auch noch, wenn Einspeicher- und Ausspeicherraten eingeführt werden, die linear fallend sind und die einzige Begrenzung der möglichen Entscheidungen bilden. In all diesen Fällen ist die Entscheidung, ob Gas gekauft, verkauft oder keins von beiden zutrifft unabhängig vom aktuellen Füllstand des Speichers und ist damit nur abhängig vom aktuellen Preis.

Diese spezielle Struktur der optimalen Strategie ändert sich sofort, wenn das ursprüngliche Problem mit konkaven bzw. konvexen Einspeicher- bzw. Ausspeicherraten im Preis betrachtet wird. Aufgrund der Tatsache, dass die Wertfunktion konkav im Speicherstand ist, kann die Aussage aus Secomandi [34] auf diesen Fall erweitert und die Struktur der optimalen Politik angegeben werden. Die optimale Entscheidung zu jedem Zeitpunkt wird beeinflusst durch Strategieschranken, die abhängig vom aktuellen Preis sind und angeben, ob und wie viel Gas dem Speicher entnommen bzw. zugeführt wird. Unter weiteren Voraussetzungen sind diese Strategieschranken fallend im Preis.

Unter Verwendung der Struktur der optimalen Politik, lässt sich mit Hilfe von Algorithmen der Wert eines Gasspeichers berechnen. Dabei wird als zugrundeliegender Markov-Prozess ein mean-reverting Modell für die Log-Preise verwendet. Aufgrund der Verteilungseigenschaften dieses Prozesses lassen sich Pfade relativ einfach simulieren. Desweiteren erfüllt der Prozess die Eigenschaften für eine Binomialapproximation nach Nelson u. Ramaswamy [28], die zu einem Binomialbaum-Algorithmus führt. Das zentrale Problem bei der Berechnung des Gasspeicherwertes durch Algorithmen ist, wie der sogenannte Continuation Value, der bedingte Erwartungswert der zukünftigen Wertefunktion unter dem aktuellen Preis, approximiert wird. Hier lässt sich zum einen der Least-Square-Ansatz anwenden, aber auch ein Interpolationsansatz ist möglich. Der letztere lässt sich wiederum unterteilen in drei Teilansätze, um mit Hilfe von Integral, „nested“ Simulation oder Quantisierungsmethode unter Verwendung der bekannten Verteilung der Zuwächse des Preisprozesses den gesuchten Erwartungswert zu approximieren.

In einer numerischen Betrachtung des Beispieles eines Gasspeichers stellt sich heraus, dass der Interpolationsansatz mit Integration jedoch kritisch zu sehen ist, da das Verhalten des Integranden für eine numerische Integration ungeeignet ist. Der numerische Vergleich der Monte-Carlo-Ansätze ergibt, dass die Konvergenzgeschwindigkeit im Interpolationsansatz höher ist als die Konvergenzgeschwindigkeit des Least-Square-Ansatzes, was die höhere Laufzeit des Interpolationsansatzes wieder aufwiegt. Die Quantisierungsmethode weist als Ergebnis eine zusätzliche Reduktion der empirischen Standardabweichung in den durch den Interpolationsansatz berechneten Schätzwerten auf. Der Binomialbaum-Algorithmus hat einen großen Vorteil gegenüber den Monte-Carlo Algorithmen. Es ist nicht notwendig mehrere Male den Algorithmus durchzuführen, um einen Eindruck des Verhaltens des Schätzers und damit des Algorithmus zu erhalten. Der Binomialbaum-Algorithmus liefert auf dem Approximationsgitter des Preises den

„exakten“ Wert. Wie in den Monte-Carlo Algorithmen auch, ist hier durch eine besondere Erzeugung des Diskretisierungsgitters im Speicherstand mittels der Einspeicher- und Ausspeicherraten, ebenfalls eine Konvergenzbeschleunigung zu erkennen. Im Einfluss der Preisparameter, Struktur der Strategieschranken und Gasspeicherentwicklung, weisen alle drei Algorithmen ähnliches Verhalten auf.

Die zwei betrachteten aus der Dualitätstheorie erhaltenen Algorithmen für Schätzer oberer Schranken des Gasspeicherwertes sind in den empirischen Untersuchungen kritisch zu beurteilen. Zwar liefert die Methode der Information Relaxation ohne Penalization eine obere Schranke, allerdings ist diese recht hoch. In der Methode mit Penalization ist je nach Parameterwahl nicht eindeutig bestimmbar, ob die innerhalb des Algorithmus gemachten Approximationen nicht einen zu starken Einfluss aufweisen und daher kleinere Werte als der Referenzwert liefern.

Im weiteren Verlauf einer empirischen Untersuchung wäre weiterhin zu analysieren, inwieweit sich die Algorithmen auf ein anderes Preismodell übertragen lassen. Die einzige theoretische Einschränkung hier ist, dass es sich um einen Markov-Prozess handeln muss, evtl. ergänzt durch regime switching, und die Voraussetzung für die Schrankenfunktion erfüllt sein muss. Eine erste Erweiterung des Prozesses wäre durch zusätzliche zufällige Sprünge im Preismodell. Benth u. a. [4] schlagen hier zum Beispiel ein jump-diffusion Modell vor. Die Binomialapproximationsmethode, die für den entsprechenden Algorithmus verwendet wird, ist hier allerdings beschränkt, da die Methode bisher nur auf Diffusionsprozesse anwendbar ist. Dennoch existieren in der Literatur bereits Ansätze, rekombinierende Binomialbäume für Prozesse mit Sprüngen zu konstruieren. Für die Monte-Carlo Algorithmen muss der Prozess lediglich simulierbar sein. Allerdings ist hier zu differenzieren, denn gerade der Interpolationsansatz verwendet das Wissen der Verteilung der Zuwächse des Preisprozesses, insbesondere für die Quantisierungsmethode. Für die Simulationsmethode innerhalb des Interpolationsansatzes ist jedoch nur die Simulation der Zuwächse nötig.

Im Hinblick auf die Implementierung der Algorithmen, wäre eine mögliche Parallelisierung zu untersuchen und inwieweit diese sich insbesondere auf die Laufzeit und auf die Möglichkeit, feinere Gitter in Speicherstand und Preis zu verwenden, auswirkt.

Aufgrund der aktuellen Entwicklung in Bezug auf die energiepolitischen globalen und lokalen Zielsetzungen, definieren Energieversorger ihren Energiemix neu. Dem Gasspeicherproblem kommt dadurch eine wachsende Bedeutung zu und es wird die strategischen Optionen der Unternehmen in den kommenden Jahren zunehmend beeinflussen.

Literaturverzeichnis

[1] Anderson, E. ; Bai, Z. ; Bischof, C. ; Blackford, S. ; Demmel, J. ; Dongarra, J. ; Du Croz, J. ; Greenbaum, A. ; Hammarling, S. ; McKenney, A. ; Sorensen, D.: *LAPACK Users' Guide*. 3rd Edition. Philadelphia, PA : Society for Industrial and Applied Mathematics, 1999

[2] Bagus, Florian: *Stukturaussagen für die optimalen Ausübungsstrategien bei multiplen Stoppproblemen und Swing Optionen*, Universität Siegen, Diss., 2012

[3] Bauer, Heinz: *Wahrscheinlichkeitstheorie*. 4., völlig überarb. und neugestaltete Aufl. Berlin [u.a.] : de Gruyter, 1991 (De-Gruyter-Lehrbuch)

[4] Benth, Fred E. ; Benth, Jurate S. ; Koekebakker, Steen: *Advanced series on statistical science and applied probability*. Bd. 11: *Stochastic Modelling of electricity and related markets*. Singapore : World Scientific, 2008

[5] Boogert, Alexander ; Jong, Cyriel de: Gas Storage Valuation Using a Monte Carlo Method. In: *The Journal of Derivatives* 15 (2008), Nr. 3, S. 81–98

[6] Breslin, John ; Clewlow, Les ; Elbert, Tobias ; Kwok, Calvin ; Strickland, Chris: Gas Storage: overview and static valuation. (2008), S. 62–68

[7] Brown, David B. ; Smith, James E. ; Sun, Peng: Information Relaxations and Duality in Stochastic Dynamic Programs. In: *Operations Research* 58 (2010), July-August, Nr. 4, S. 785–801

[8] Bäuerle, Nicole ; Rieder, Ulrich: *Markov Decision Processes with Applications to Finance*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011. – ISBN 978–3–642–18324–9

[9] Burger, Markus ; Graeber, Bernhard ; Schindlmayr, Gero: *Managing energy risk : an integrated view on power and other energy markets*. Chichester, England : Wiley, 2007

[10] Byers, Joe W.: Commodity storage valuation: A linear optimization based on traded instruments. In: *Energy Economics* 28 (2006), May, S. 275–287

[11] Carmona, René ; Ludkovski, Michael: Valuation of energy storage: an optimal switching approach. In: *Quantitative Finance* 10 (2010), Nr. 4, S. 359–374

[12] Clément, Emmanuelle ; Lamberton, Damien ; Protter, Philip: An analysis of a least squares regression method for American option pricing. In: *Finance and Stochastics* 6 (2002), Nr. 4, S. 449–471

[13] Corlay, Sylvain ; Pagès, Gilles ; Printems, Jacques: *The optimal quantization website*. `http://www.quantize.maths-fi.com`. Version: 2005

[14] Energy Information Agency, EIA: *The Basics of Underground Natural*. `http://www.eia.gov/`. Version: August 2004

[15] Felix, Bastian: Gas Storage Valuation: A Comparative Simulation Study. In: *EWL Working Paper* (2012), Nr. 1. `http://ssrn.com/abstract=2089268orhttp://dx.doi.org/10.2139/ssrn.2089268`

[16] Felix, Bastian ; Weber, Christoph: Gas storage valuation applying numerically constructed recombining trees. In: *European Journal of Operational Research* (2012), S. 178–187

[17] Galassi, Mark ; Davies, Jim ; Theiler, James ; Gough, Brian ; Jungman, Gerard ; Alken, Patrick ; Booth, Michael ; Rossi, Fabrice: *GNU Scientific Library Reference Manual*. 3rd Edition. Network Theory, 2009 `http://www.gnu.org/software/gsl/`

[18] Graf, Siegfried ; Luschgy, Arald: *Foundations of quantization for probability distributions*. Berlin : Springer, 2000 (Lecture notes in mathematics ; 1730)

[19] Gruber, Peter M.: *Convex and discrete geometry*. Berlin : Springer, 2007 (Grundlehren der mathematischen Wissenschaften ; 336)

[20] Haugh, Martin ; Kogan, Leonid: Pricing American Options: A Duality Approach. In: *Operations Research* 52 (2004), March-April, Nr. 2, S. 258–270

[21] Hinderer, Karl: Lipschitz Continuity of Value Functions in Markovian Decision Processes. In: *Mathematical Methods of Operations Research* 62 (2005), S. 3–22

[22] Hodges, Stewart D.: The value of a storage facility / Warwick Business School. 2004. – Working paper

[23] Holland, Alan: Optimization of Injection/Withdrawal Schedules for Natural Gas Storage Facilities *. In: *Proceedings of Symposium of Applied Computing 2007, ACM Press, Seoul, South Korea*, 2007

[24] Jong, Cyriel de ; Walet, Kasper: To store or not to store. In: *Energy Risk (früher Energy Price Risk Management)* (2003), October, S. 8–11

[25] Longstaff, Francis A. ; Schwartz, Eduardo S.: Valuing American Options by Simulation: A Simple Least-Squares Approach. In: *The Review of Financial Studies* 14 (2001), Nr. 1, S. 113–147

[26] *Kapitel* 14. In: Maragos, Spyros: *Valuation of the Operational Flexibility of Natural Gas Storage Reservoirs*. London : Risk Books, 2004, S. 431–456

[27] Müller, Alfred ; Stoyan, Dietrich: *Comparison Methods for Stochastic Models and Risks*. Chichester : John Wiley and Sons, Ltd, 2002

[28] Nelson, Daniel B. ; Ramaswamy, Krishna: Simple Binomial Processes as Diffusion Approximations in Financial Models. In: *The Review of Financial Studies* 3 (1990), Nr. 3, S. 393–430

[29] Pagès, Gilles ; Printems, Jaques: Optimal quadratic quantization for numerics: the Gaussian case. In: *Monte Carlo Methods and Applications* 9 (2003), S. 135–166

[30] R Core Team: *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2013. `http://www.R-project.org/`

[31] R Core Team: *Writing R Extensions*, 2013. (Version 3.0.1)

[32] Schlüter, Stephan ; Davison, Matt: Pricing an European gas storage facility using a continuous-time spot price model with GARCH diffusion. In: *IWQW Discussion Paper Series, FAU Erlangen-Nürnberg* (2010), Nr. 2. `http://hdl.handle.net/10419/30184`

[33] Schwartz, Eduardo: The Stochastic Behavior of Commodity Prices: Implications for Valuation and Hedging. In: *The Journal of Finance* 52 (1997), Juli, Nr. 3, S. 923–973

[34] Secomandi, Nicola: Optimal Commodity Trading with a Capacitated Storage Asset. In: *Management Science* 56 (2010), März, Nr. 3, S. 449–467. – ISSN 0025–1909

[35] Shreve, Steven E.: *Stochastic Calculus for Finance II: Continuous-Time Models*. New York : Springer, 2004 (Springer Finance Bd. 11)

[36] Stoll, Sven-Olaf ; Wiebauer, Klaus: A spot price model for natural gas considering temperature as an exogenous factor with applications. In: *The Journal of Energy Markets* 3 (2010), Nr. 3, S. 113–128

[37] Thompson, Matt ; Davison, Matt ; Rasmussen, Henning: Natural gas storage valuation and optimization: A real options application. In: *Naval Research Logistics* 56 (2009), S. 226–238

[38] Tong, Yung-Liang: *The multivariate normal distribution*. New York : Springer, 1990 (Springer series in statistics)