Financial Engineering mit R und Rmetrics

Josef Hayden

Jänner 2007

Inhaltsverzeichnis

Vorwort

Diese Arbeit behandelt das Thema Financial Engineering mit `R` und `Rmetrics`. Der Entwicklung von neuen und innovativen Finanzprodukten kommt zunehmend eine immer stärkere Bedeutung zu und damit auch dem Financial Engineering. Für das Financial Engineering spielen leistungsfähige Programmiersprachen eine zentrale Rolle. Hier ergibt sich die Verbindung zu dem Open Source Softwarepaket `R`. Ähnlich wie bei der Programmiersprache `C++` gibt es eine große Anzahl von frei verfügbaren Erweiterungspaketen, wie etwa `Rmetrics`. Der Vorteil von `R` liegt sicher in der freien Verfügbarkeit, Erweiterbarkeit, den enormen grafischen Fähigkeiten und der matrixorientierten Programmiersprache.

Das Ziel dieser Arbeit ist nun einerseits das Softwarepaket und vor allem die Erweiterungen theoretisch zu beleuchten und andererseits die praktische Verwendung im Bereich des Financial Engineering zu demonstrieren. Zusätzlich sollen auch die Möglichkeiten und das Potential von `R` in Hinblick auf die Entwicklung von eigenen Erweiterungen gezeigt werden.

Der Schwerpunkt dieser Arbeit liegt aber nicht in der reinen Auflistung von relevanten Packages und deren Funktionen. Vielmehr soll im ersten Teil der Arbeit ein theoretischer Überblick geschaffen werden um für Fragestellungen des Financial Engineering auf relevante Packages zurückgreifen zu können. Ausführlicher wird dabei die Erweiterung `Rmetrics`, die selbst aus acht Packages besteht, behandelt.

Der zweite Teil dieser Arbeit stellt den Kernteil dar und demonstriert anhand von Fallstudien den Einsatz von `R` für Fragestellungen im Bereich des Financial Engineering. Eine Entwicklung von neuartigen Finanzprodukten ist allerdings nicht das Ziel dieser Arbeit. Stattdessen soll die Funktionalität durch die Implementierung von bestehenden und bewährten Finanzmodellen in `R` demonstriert werden.

Wie bereits angedeutet, gliedert sich die Arbeit in zwei große Bereiche. *Teil I* behandelt das Softwarepaket `R` allgemein und beleuchtet `Rmetrics` etwas genauer. Im *Kapitel 1* wird auf die einzelnen Packages der Erweiterung `Rmetrics` im Detail eingegangen. Es werden die Funktionsgruppen vorgestellt und Funktionsüberschneidungen mit anderen Packages aufgezeigt. Im *Kapitel 2* werden weitere relevante Packages für das Financial Engineering präsentiert.

Teil II rückt die eigene Implementierung von Finanzmodellen und die Anwendung von `R` Packages in den Vordergrund. In jedem Kapitel wird zu Beginn der, für die Lösung der Fallstudien beziehungsweise für die Implementierung benötigte, theoretische Teil beleuchtet. Abgeschlossen wird jedes Kapitel mit einer Präsentation von Beispielen, die mit `R` Packages gelöst wurden.

Kapitel 3 behandelt die Zinsstrukturschätzung nach den Modellen von Svensson (1994) und Nelson und Siegel (1987). Die Schätzung der Zinsstruktur erfolgt dabei mit dem, im Zuge dieser Arbeit entstandenen, Package `termstrc`.

Die Bewertung von Optionen mit numerischen Verfahren, wie mit Binomialbäumen, finiten Differenzen und Monte Carlo Simulation wird in *Kapitel 4* erläutert und demonstriert. Zur Bewertung von amerikanischen Optionen kommt dabei das Package `fOptions` und `RQuantLib` zur Anwendung. Die Monte Carlo Simulation von asiatischen Optionspreisen basiert auf dem Package `fOptions`. Zusätzlich wird ein *jump diffusion* Modell simuliert.

In *Kapitel 5* werden einfache Zinsderivate mittels Binomial- beziehungsweise Trinomialbäumen bewertet. Die Bewertung basiert dabei auf Einfaktor-Short-Rate Modellen. Als Short-Rate Modelle wurden dabei das Modell von Hull und White (1990) und Black, Derman und Toy (1990) implementiert. Die Kalibrierung kann dabei nach der Zinsstruktur und beim Black-Derman-Toy Modell auch zusätzlich nach der Volatilitätsstruktur erfolgen. Im Rahmen dieser Arbeit ist ein weiteres Package entstanden. Das Package `shortrate` umfasst die implementierten Short-Rate Modelle und Funktionen für die Bewertung einfacher Zinsderivate.

Im Anhang ist der vollständige Quellcode der Packages `termstrc` und `shortrate` zu finden.

Teil I

R und Rmetrics

Bei R handelt es sich um eine Open Source Software, die ursprünglich von Ihaka und Gentleman (1996) entwickelt worden ist. Seit Mitte 1997 hat die Entwicklung eine Kerngruppe übernommen, das so genannte R-Core Team (vgl. Hornik, 2006).

Auf der Homepage des R-Projects wird R wie folgt beschrieben wird:

> *"R is a system for statistical computation and graphics. It consists of a language plus a run-time environment with graphics, a debugger, access to certain system functions, and the ability to run programs stored in script files* (Hornik, 2006)."

Die Website von R ist unter `www.R-project.org` zu finden. Der Quellcode und die Binärdateien für die Installation für die verschiedensten Betriebssysteme sind unter `http://CRAN.R-project.org` verfügbar. Bei CRAN (Comprehensive R Archive Network) handelt es sich um ein weltweites Netz von Servern, das die relevanten Installationsdateien, viele Zusatzpackages und Dokumentationen zur Verfügung stellt.

Auf eine Einführung in das Softwarepaket wird in dieser Arbeit allerdings verzichtet, vielmehr soll die Anwendung von R und den relevanten Packages im Vordergrund stehen. Für eine Einführung beziehungsweise für erste Schritte mit R ist Hornik (2006) und Ligges (2005) zu empfehlen.

Der erste Teil dieser Arbeit beschäftigt sich mit dem Softwarepaket R und vor allem mit den Erweiterungen, den so genannten Packages. Der Schwerpunkt liegt in der Vorstellung der für das Financial Engineering relevanten Packages. Die theoretische Beleuchtung der Packages erfolgt in diesem Kapitel. Im zweiten großen Kapitel kommt es zur Lösung von Fallstudien mit R und ausgewählten Packages.

Die Vorstellung der Packages soll aber nicht den Inhalt der vorhandenen Dokumentation kopieren, sondern vielmehr einen Überblick liefern, welche Funktionen die Packages bieten. Ziel soll es sein, die für das Financial Engineering relevanten Packages aus der Vielzahl von verfügbaren Packages herauszufiltern und zu präsentieren, ohne aber die Ausführungen auf eine reine Auflistung von Funktionen zu reduzieren.

Die behandelten Packages werden sich auf die im CRAN Task View Empirical Finance vorgestellten Packages beschränken. Eine spezielle Stellung nimmt dabei `Rmetrics` ein. Dieses Erweiterungspaket zu R besteht selbst, zum Zeitpunkt des Verfassens dieser Arbeit, aus acht Packages, die alle im folgenden Kapitel vorgestellt werden.

Zu beachten ist, dass sich die Informationen über die Packages auf den Zeitpunkt der Verfassung dieser Arbeit beziehen. Durch die rege Weiterentwicklung von R und vielen Packages kann nicht ausgeschlossen werden, dass die angeführten Informationen über die Packages nicht mehr aktuell sind.

Kapitel 1

Rmetrics

Wie bereits erwähnt handelt es sich bei `Rmetrics` um eine Gruppe von `R` Packages, die in diesem Kapitel näher beleuchtet werden. `Rmetrics` wurde von Diethelm Würtz im Rahmen seiner Econophysik Vorlesungen an der ETH Zürich entwickelt. Auf der Homepage von `Rmetrics`[1] ist folgende Beschreibung zu finden:

> *"Rmetrics is the premier open source solution for financial market analysis and valuation of financial instruments. With hundreds of functions build on modern and powerful methods Rmetrics combines explorative data analysis and statistical modeling with object oriented rapid prototyping"* (Wuertz, 2006i).

Rmetrics besteht zum Zeitpunkt des Verfassens dieser Arbeit aus folgenden veröffentlichten Packages:

- `fBasics`
- `fCalendar`
- `fSeries`
- `fExtremes`
- `fMultivar`
- `fOptions`
- `fPortfolio`
- `fEconfin`

[1] http://www.rmetrics.org

Alle genannten Packages sind unter `http://CRAN.R-project.org` frei verfügbar. Die Funktionsgruppen der Packages werden in den folgenden Abschnitten näher erläutert, wobei jeweils auch die Originalnamen der Gruppen verwendet werden um eine Konsistenz mit der Dokumentation von `Rmetrics` herzustellen. Am Ende jedes Abschnitts werden auch Überschneidungen mit anderen Packages aufgezeigt. Als eine Überschneidung werden identische Funktionen gewertet, d.h. diese Funktionen hat in zwei oder mehr Packages den selben Funktionsumfang. Es werden aber auch die Packages angeführt, wenn Funktionen von `Rmetrics` auf diesen basieren.

1.1 fBasics

Die Informationen über das Package `fBasics` stammen aus der Dokumentation und für weiterführende Informationen sei auch auf diese verwiesen (vgl. Wuertz, 2006a).

`fBasics` beinhaltet grundlegende Funktionen zum Analysieren von Finanzdaten. Die Funktionen im Package sind jeweils zu Gruppen von Funktionen zusammengefasst. Diese Gruppen sollen nun vorgestellt werden.

fBasicsUtilities

Diese Gruppe umfasst Hilfsfunktionen auf die andere Funktionen des Packages `fBasics` zurückgreifen. So werden etwa Funktionen für das Farbmanagement geboten.

TailoredReturnPlots

Über die Funktionen dieser Gruppe können verschiedene Grafiken der Renditen von Finanzzeitreihen erzeugt werden. Die Zeitreihe muss allerdings ein Objekt der Klasse `timeSeries` sein. Mehr dazu unter 1.2 beziehungsweise 1.3. Als Plots stehen im speziellen Histogramm -, Dichte- und Quantilplots zur Verfügung.

StableDistribution

Über Funktionen dieser Gruppe können die Dichte-, Verteilungs- und Quantilfunktion von stabilen Verteilungen berechnet und Zufallsvariablen generiert werden. Es werden Funktionen für stabile symmetrische und stabile schiefe Verteilungen zur Verfügung gestellt.

HyperbolicDistribution

Für drei Arten von hyperbolischen Verteilungen (generelle hyperbolische, hyperbolische und die normale inverse Gaussche Verteilung) werden Funktionen zur Erzeugung von Zufallszahlen und zur Berechnung der Dichte-, Verteilungs- und Quantilfunktion zur Verfügung gestellt.

DistributionFits

Diese Gruppe bietet Funktionen mit Maximum-Likelihood Schätzern, welche die Parameter von Verteilungen aus empirischen Daten schätzt. Schätzer sind dabei für die Normal-, Normal-Inverse-, Student-t, stabile, generalisierte hyperbolische Verteilung und für empirische Verteilungen inkludiert.

BasicStatistics

Die Funktionen dienen zum Berechnen von statistischen Eigenschaften (Mittelwert, Varianz, Schiefe, Kurtosis) von Finanzzeitreihen.

StylizedFacts

Die Funktionen bieten die Möglichkeiten sich Eigenschaften von Finanzzeitreihen grafisch zu veranschaulichen. Es werden Plot-Funktionen für die Autokorrelations-Funktionen, Quantile und den Taylor Effekt zur Verfügung gestellt.

PortableInnovations

Diese Funktionen erzeugen Zufallszahlen, die Zufallszahlen unter S-Plus entsprechen. Dies ermöglicht die Vergleichbarkeit von Ergebnissen zwischen R und S-Plus, die auf Zufallszahlen basieren. Unterstützt werden Zufallszahlen der Gleich-, Normal- und t-Verteilung.

HypothesisTesting

Für statistische Tests wurde in `fBasics` die S4 -Klasse `fHtest` definiert. Es steht eine `print` Methode für die Objekte der Klasse bereit.

Testbezeichnung
Kolmogorov-Smirnov
Shapiro-Wilk
Jarque-Bera
D'Agostino
Anderson-Darling
Cramer von Mises
Lillifors
Pearson χ^2
Shapiro-Francia

Tabelle 1.1: fBasics -Test auf Normalverteilung

OneSampleTests

Für univariate Zeitreihen werden verschiedene Tests auf Normalverteilung zur Verfügung gestellt. Die implementierten Tests auf Normalverteilung sind Tabelle 1.1 zu entnehmen.

Weiters ist auch der Runs Test für das Aufzeigen von Nicht-Zufälligkeit implementiert.

TwoSampleTests

Die für zwei-Stichproben konzipierten Tests bieten Möglichkeiten Verteilungen, Mittelwerte und Varianzen, Korrelationen von Stichproben auf ihre Gleichheit zu testen. Die in Tabelle 1.2 aufgezählten Tests sind verfügbar.

Überschneidungen mit anderen Packages

Tabelle 1.3 zeigt die Gruppen von Funktionen des Packages `fBasics` und etwaige Überschneidungen mit Funktionen von anderen Packages.

Testbezeichnung
Kolmogorov-Smirnov Test für zwei Stichproben
t-Test
Kurskal-Wallis
F-Test für die Varianz
Bertlett
Fligner-Killeen
Ansari-Bradley
Mood
Person‘s Koeffizient
Kendall‘s tau
Spearman´s rho

Tabelle 1.2: fBasics - zwei-Stichprobentests

Gruppe von Funktionen	Package
fBasicsUtilities	-
TailoredReturnPlots	-
StableDistribution	-
HyperbolicDistribution	-
DistributionFits	-
BasicStatistics	`base`
StylizedFacts	-
PortableInnovations	-
HypothesisTesting	-
OneSampleTests	`ctest,nortest,tseries`
TwoSampleTests	`ctest`

Tabelle 1.3: fBasics-Funktionsüberschneidungen

1.2 fCalendar

Die Informationen über das Package `fCalendar` stammen aus der Dokumentation und für weiterführende Informationen sei auch auf diese verwiesen (vgl. Wuertz, 2006b).

`fCalendar` vereinigt Funktionen zum Management von Zeitzonen und beinhaltet Klassen für Datum und Zeitreihen. Im Detail gliedert sich das Package in folgend angeführte Gruppen von Funktionen.

DayligthSavingTime

Die Funktionen liefern für 92 Länder und Regionen die Abweichung der Zeit von einer Referenzzeit (z.B. GMT) und für alle Jahre bis zum Jahr 2030 das Datum und die Uhrzeit der Umstellung auf die Sommerzeit.

TimeDateClass

In dieser Gruppe stehen Gruppen und Methoden zum Management von Zeitdaten aus verschiedensten Zeitzonen zur Verfügung. Für die S4 Klasse `timeDate` wurden eigene Methoden definiert. Über die Definition eines eigenen Finanzzentrums mit allen dazugehörigen Regeln (Urlaubstage, Sommerzeit) können problemlos Berechnungen von zeitabhängigen Daten aus verschiedenen Zeitzonen durchgeführt werden.

TimeDateSubsets

Diese Gruppe stellt Funktionen für Objekte der Klasse `timeDate` zur Verfügung. So kann ein Datum überprüft werden ob es etwa ein Wochentag oder Feiertag (für ein gegebenes Finanzzentrum) ist. Zur Handhabung dieser Objekte gibt es auch Funktionen um Teilmengen aus den Objekten der Klasse `timeDate` extrahieren zu können.

TimeMathOps

Diese Funktionen bieten Methoden für mathematische und logische Operationen für Objekte der Klasse `timeDate`.

TimeDateSpecDates

Über diese Funktionen können Zeitdaten gemäß bestimmter Konvention erzeugt werden. So kann etwa das Datum des ersten Tages in einem bestimmten Monat berechnet werden.

TimeDateCoercion

Diese Gruppe bietet Funktionen um Objekte der Klasse `timeDate` in eine andere Klasse umzuwandeln. Es können `timeDate` Objekte in Objekte der Klasse `POSIX`, `data frame` oder `character string` umgewandelt werden.

TimeSeriesClass

Wie bereits mehrfach ausgeführt gibt es in `Rmetrics` eine eigene Klasse für Zeit- und Datumsobjekte. Auch für Zeitreihen ist eigene Klasse definiert worden. Für die Objekte der Klasse `timeSeries` stehen wiederum eine Vielzahl von Methoden und Funktionen zur Verfügung.

TimeSeriesData

Diese Gruppe umfasst Funktionen für Zeitreihenobjekte der Klasse `timeSeries` zum Modifizieren und Klassifizieren der Zeitreihen sowie mathematische Operatoren und Funktionen für Teilmengen.

TimeSeriesPositions

Es werden Methoden und Funktionen zum Extrahieren und Modifizieren von Objekten der Klasse `timeSeries` geboten.

TimeSeriesCoercion

Diese Gruppe bietet Funktionen um Objekte der Klasse `timeSeries` in eine andere Klasse umzuwandeln beziehungsweise Objekte der Klasse `timeSeries` aus anderen Objekten zu erzeugen. Es können `timeSeries` Objekte in Objekte der Klasse `ts,zoo` oder auch in ein `matrix` oder ein `data frame` Objekt umgewandelt werden beziehungsweise umgekehrt.

HolidayDates

Diese Gruppe ermöglicht die Berechnung von Feiertagen für alle G7 Länder und die Schweiz.

HolidayCalendars

Es können die von Ostern abhängigen Feiertage für die G7 Länder, die Schweiz und den New York Stock Exchange berechnet werden.

TimeSeriesImport

Über die Funktionen, die zu dieser Gruppe zusammengefasst sind können Finanzmarktdaten und auch ökonomische Marktdaten vom Internet direkt in `R` geladen werden. Tabelle zeigt dabei die Internetportale für die Importfunktionen implementiert sind [2] .

Hompage	Bezeichnung
www.economagic.com	Economagic bietet ökonomische Zeitreihen
finance.yahoo.com	Yahoo bietet eine große Anzahl an Finanzdaten
research.stlouisfed.org	Zeitreihen von der Federal Reserve Bank von St.Louis
www.forecasts.org	Bietet monatliche Forecasts von Finanzzeitreihen

Tabelle 1.4: fCalender - Importfunktionen für Internetportale

Überschneidungen mit anderen Packages

Das Package `fCalendar` stellt neben dem Package `chron`, das auf ökonomische Zeitreihen maßgeschneidert ist, einen eigenen Ansatz dar. Das `base` Package bietet als Standard für das Management von Zeitdaten den `POSIXt` Standard, der jedoch nicht komplett betriebssystemunabhängig ist.

1.3 fSeries

Die Informationen über das Package `fSeries` stammen aus der Dokumentation und für weiterführende Informationen sei auch auf diese verwiesen (vgl. Wuertz, 2006h).

`fSeries` bietet Funktionen für die Analyse von Zeitreihen, die nun vorgestellt werden.

ArmaModelling

Die Funktionen dieser Gruppe ermöglichen das Modellieren von univariaten autoregressiven Moving Average Zeitreihenprozessen. Die Funktionen bieten auch die Möglichkeit Zeitreihen zu simulieren, Parameter der Prozesse zu schätzen und Vorhersagen für die Prozesse zu treffen. Ebenfalls wurde eine S4 Klasse mit dem Namen `fARMA` definiert. Folgende Modelle werden von den Funktionen abgedeckt:

[2] Es gibt in Rmetrics die Funktion zum Import von www.forecasts.org, allerdings wird der Zugriff auf die Webpage verweigert.

Kürzel	Bezeichnung
ar	autoregressiver Prozess
ma	Moving Average
arma	autoregressiver Moving Average
arima	autoregressiver integrierter Moving Average
arfima	partiell integrierter ARMA Prozess

Tabelle 1.5: fSeries - Zeitreihenmodelle

ArfimaOxInterface

Die Anpassung einer univariaten Zeitreihe an ein ARFIMA Modell wird durch die Funktionen dieser Gruppe unterstützt. Es erfolgt dabei eine Anbindung an das ARFIMA Ox Package. Wobei Ox eine objektorientierte Matrix-Programmiersprache ist. Für weitere Informationen sei auf die Homepage `www.timberlake.co.uk` der Vertreiberfirma verwiesen.

UnitrootDistribution

Die Funktionen bieten die Möglichkeit der Berechnung der Verteilungs- und Quantilfunktion von Unit-root Teststatistiken.

UnitrootTests

Die Gruppe von Funktion stellt eine Reihe von Unit-Root Test zur Verfügung. In Tabelle 1.6 sind die Tests aufgelistet. Alle Tests liefern ein Objekt der S4 Klasse `fHTEST` zurück.

Testbezeichnung
Dickey-Fuller
Elliot-Rothenberg-Stock
KPSS Test auf Stationarität
Phillips-Perron
Schmidt-Phillips
Zivot-Andrews

Tabelle 1.6: fSeries - Unit-root Tests

LongRangeDependence

Die Funktionen dieser Gruppe ermöglichen das Untersuchen des Langzeitverhaltens von univariaten Zeitreihen. So besteht die Möglichkeit den Hurst-Exponenten zu berechnen, die tatsächliche Autokorrelation zu modellieren und partielle ARMA Prozesse und partielles Gaussches Rauschen zu simulieren. Auch ein Wavelet Schätzer wird zur Verfügung gestellt.

GarchDistribuations

Für die Berechnung der Verteilungs-, Dichte- und Quantilfunktion der schiefen Normalverteilung, der schiefen t-Verteilung und schiefen allgemeinen Error-Verteilung werden Funktion geboten. Zusätzlich können für die genannten Verteilungen Zufallszahlen erzeugt und Parameter über Maximum Likelihood Schätzer an die Verteilungen angepasst werden.

HeavisideFunction

Für das Modellieren von GARCH Prozessen stehen die Heaviside Funktion und davon abhängig Funktionen zur Verfügung.

GarchModelling

Die Funktionen dieser Gruppe ermöglichen das Simulieren von GARCH (*generalized autoregressive conditional heteroskedasticity*) Modellen, die Anpassung der Parameter von univariaten Zeitreihen und die Vorhersage von zukünftigen Werten von Zeitreihen. Als S4 Klasse wurde `fGARCH` definiert.

GarchOxInterface

Über eine Anbindung an das G@RCH Ox Package können Parameter einen univariaten Zeitreihe an GARCH Modelle angepasst werden.

ChaoticTimeSeries

Die Gruppe enthält Funktionen zum Untersuchen des Verhaltens von chaotischen Zeitreihen und zum Simulieren von verschiedenen Typen von chaotischen Zeitreihen.

TimeSeriesTests

Die in Tabelle 1.7 dargestellten Tests stehen für univariate Zeitreihen zur Verfügung. Die Test liefern ein Object der S4 Klasse `fHTEST`.

Testbezeichnung
Brock-Dechert-Scheinkman Test auf iid
Teraesvirta Neural Network Test auf vernachlässigte Nicht-Linearität
White Neural Network Test auf vernachlässigte Nicht-Linearität

Tabelle 1.7: fSeries - Tests für univariate Zeitreihen

Überschneidungen mit anderen Packages

Tabelle 1.8 zeigt die Gruppen von Funktionen des Packages `fSeries` und etwaige Überschneidungen mit Funktionen von anderen Packages.

Gruppe von Funktionen	Package
ARMAModelling	`stats,fracdiff`
ArfimaOxInterface	Arfima Ox Package
UnitrootDistribution	-
UnitrootTests	`urca,tseries`
LongRangeDependence	`wavetresh`
GarchDistributions	-
HeavisideFunction	-
GarchModelling	-
GarchOxInterface	G@RCH Ox Package
ChaoticTimeSeries	-
TimeSeriesTests	`tseries`

Tabelle 1.8: fSeries - Funktionsüberschneidungen mit anderen Packages

1.4 fExtremes

Die Informationen über das Package `fExtremes` stammen aus der Dokumentation und für weiterführende Informationen sei auch auf diese verwiesen (vgl. Wuertz, 2006d).

`fExtremes` beinhaltet Funktionen für Verteilungen die für die Extremwerttheorie von Relevanz sind. Im Detail gliedert sich das Package in folgend angeführte Gruppen von Funktionen.

ExtremesData

Diese Gruppe beinhaltet Funktionen zur explorativen Datenanalyse inklusive der Handhabung von Extremwerten. Es werden verschiedene Plotfunktionen geboten, wie etwa für empirische Verteilungen oder Quantile.

GevModelling

Es stehen Funktionen zur Berechnung der allgemeinen Extremwertverteilung und zur Schätzung ihrer Parameter zur Verfügung. So können die Dichte, die Verteilungsfunktion, die Quantilfunktion berechnet und Zufallszahlen für die Frechet-, Gumbel- und Weibullverteilung erzeugt werden. Das Modellieren der allgemeinen Extremwertverteilung kann über drei verschiedene Schätzansätze erfolgen (Maximum Likelihood Schätzer, wahrscheinlichkeitsgewichtete Momentenmethode, MDA Ansatz).

GpdDistribution

Mit Hilfe der Funktionen können die Dichte, die Verteilungsfunktion, die Quantilsfunktion der allgemeinen Pareto Verteilung berechnet und Zufallszahlen erzeugt werden.

GpdFit

Diese Funktionen dienen zum Modellieren der allgemeinen Pareto Verteilung. Für das Schätzen der Paramter stehen zwei Ansätze zur Verfügung (Maximum Likelihood Schätzer, wahrscheinlichkeitsgewichtete Momentenmethode). Die Funktionen basieren auf dem Package `evir`.

PotFit

Es werden Funktionen zum Modellieren von stochastischen Prozessen, so genannten *point processes* aus dem Package `evir` zur Verfügung gestellt.

GevGlmFit

Funktionen aus dem Package `ismev` ermöglichen das Modellieren der allgemeinen Extremwertverteilung durch Maximum Likelihood Schätzung. Jeder Parameter der allgemeinen Extremwertverteilung kann auch über ein allgemeines lineares Modell geschätzt werden.

GpdGlmFit

Funktionen aus dem Package `ismev` ermöglichen das Modellieren der allgemeinen Pareto Verteilung durch Maximum Likelihood Schätzung. Jeder Parameter der allgemeinen Pareto Verteilung kann auch über ein allgemeines lineares Modell geschätzt werden.

PPFit

Es werden Funktionen zum Modellieren von stochastischen Prozessen, so genannten *point processes* aus dem Package `ismev` zur Verfügung gestellt.

RlargFit

Diese Gruppe bietet die Möglichkeit mittels Funktionen aus dem Package `ismev` das *Order Statistic Model* mittels Maximum Likelihood zu approximieren.

ExtremeIndexPlots

Diese Gruppe umfasst Funktionen zur Berechnung des *extremal index* basierend auf drei verschiedenen Methoden (*blocks*, *reciprocal mean cluster size* und *runs* Methode)

Überschneidungen mit anderen Packages

Tabelle 1.9 zeigt die Gruppen von Funktionen des Packages `fExtremes` und etwaige Überschneidungen mit Funktionen von anderen Packages.

1.5 fMultivar

Die Informationen über das Package `fMultivar` stammen aus der Dokumentation und für weiterführende Informationen sei auch auf diese verwiesen (vgl. Wuertz, 2006e).

`fMultivar` besteht hauptsächlich aus Funktionen für das Modellieren von Regressionsmodellen und das Testen dieser Modelle. Im Detail gliedert sich das Package in folgend angeführte Gruppen von Funktionen.

Gruppe von Funktionen	Package
ExtremesData	`evir,ismev`
GevModelling	`evir`
GpdDistribution	`evd`
GpdFit	`evir`
PotFit	`evir`
GevGlmFit	`ismev`
GpdGlmFit	`ismev`
PPFit	`ismev`
RlargFit	`ismev`
ExtremeIndexPlots	-

Tabelle 1.9: fExtremes - Funktionsüberschneidungen mit anderen Packages

TechnicalAnalysis

Diese Gruppe beinhaltet Funktionen zur technischen Analyse von Aktienmärkten. Implementiert sind dabei gängige technische Indikatoren wie Moving Average, Oszillationsindikatoren und viele mehr.

BenchmarkAnalysis

Die Funktionen dieser Gruppe ermöglichen das Berechnen von Benchmarkanalysen. So werden Funktionen zum Berechnen des Sterling- und des Sharperatios angeboten.

RollingAnalysis

Es werden Funktionen zur Durchführung einer *rolling analysis* geboten. Ähnlich wie beim Moving Average werden für Zeitreihen basierend auf einem sich bewegenden Zeitfenster statistische Kennzahlen berechnet.

RegressionModelling

Es stehen Funktionen für verschiedenste Modelle der Regressionsanalyse zur Verfügung. Die Modelle sind in Tabelle 1.10 zusammengefasst.

Regressionsmodell
lineares Modell allgemeines lineares Modell
lineares additives Modell
projection pursuit Modell
multivariate adaptive regression splines
polytochomouns multivariate adaptive regression splines
feedforward neural network modelling

Tabelle 1.10: fMultivar - Regressionsmodelle

RegressionTests

Die Funktionen diese Gruppe ermöglichen das Testen von linearen Regressionsmodellen auf serielle Korrelation, Heteroskedastizität, Autokorrelation des Störterms, Linearität und funktionale Abhängigkeiten. Die zur Verfügung stehenden statischen Tests sind in Tabelle 1.11 aufgelistet.

Testbezeichnung
Breusch-Godfrey
Breusch-Pagan
Durbin-Watson
Goldfeld-Quandt
Harvey-Collier
Harrison-McCabe
Rainbow Test
Ramsey´s RESET Test

Tabelle 1.11: fMultivar - Tests für Regressionsanalysen

EquationsModelling

Diese Gruppe beinhaltet Funktionen zum Anpassen von Systemen von Regressionsgleichungen. Dabei sind die in Tabelle 1.12 aufgelisteten Modelle verfügbar.

VectorMatrixAddon

Die bestehenden Funktionen von `R` im Bereich der Matrixalgebra werden durch Funktionen des Packages `fMultivar` ergänzt.

Modellbezeichnung
ordinary least square modelling
weighted least square modelling
seemingly unrelated regression
two-stage least squares weighted two-stage least squares
three-stage least squares

Tabelle 1.12: fMultivar - Erweiterte Regressionsmodelle

MissingValues

Für Objekte der Klasse `timeSeries` stehen Funktionen zur Handhabung von fehlenden Werten (*missing values*) zur Verfügung.

BivariateTools

Diese Gruppe enthält Funktionen zum Management von bivariaten Datensätzen. Inkludiert sind Schätzverfahren für die Kern-Dichte und eine Auswahl von bivariaten elliptischen Verteilungsfunktionen.

MultivariateDistribution

Es stehen Funktionen zum Berechnen der Werte von multivariaten Dichte- und Verteilungsfunktionen der schiefen Normalverteilung und schiefen t-Verteilung zur Verfügung. Über die Funktionen können auch multivariate Zufallszahlen der Verteilungen erzeugt werden. Für multivariate Daten können die Parameter der zu grundeliegenden Verteilung über Maximum Log-Likelihood Methoden geschätzt werden.

Überschneidungen mit anderen Packages

Tabelle 1.13 zeigt die Gruppen von Funktionen des Packages `fMultivar` und etwaige Überschneidungen mit Funktionen von anderen Packages.

1.6 fOptions

Die Informationen über das Package `fOptions` stammen aus der Dokumentation und für weiterführende Informationen sei auch auf diese verwiesen (vgl. Wuertz, 2006f).

Gruppe von Funktionen	Package
TechnicalAnalysis	-
BenchmarkAnalysis	`tseries`
RollingAnalysis	-
RegressionModelling	`stats,mgcv,mda,polyclass,nnet,modreg`
RegressionTests	`lmtest`
EquationsModelling	`systemfit`
VectorMatrixAddon	`base,mexp`
MissingValues	`base,EMV`
BivariateTools	`sn,gregmisc`
MultivariateDistribution	`mvtnorm,sn`

Tabelle 1.13: fMultivar - Funktionsüberschneidungen mit anderen Packages

`fOptions` bietet Funktionen zur Bewertung von Optionen verschiedenster Art. Es werden analytische, approximativ analytische und numerische Bewertungsverfahren zur Verfügung gestellt. Die analytischen und approximativ analytischen Modelle stützen sich dabei auf Haug (1997). Die Funktionsgruppen sollen nun näher vorgestellt werden. An dieser Stelle soll noch angemerkt werden, dass es keine bekannten Überschneidungen mit anderen `R`-Packages gibt.

PlainVanillaOptions

Diese Gruppe stellt Funktionen zur Bewertung von *plain vanilla* Optionen zur Verfügung. Die Funktionen umfassen dabei das allgemeine Black-Scholes-Merton Optionsbewertungsmodell. Neben der Berechnung des Optionspreises besteht auch die Möglichkeit die Sensitivitäten (*the greeks*) zu berechnen.

BasicAmericanOptions

Die in Tabelle 1.14 aufgelisteten approximativen analytischen Modelle ermöglichen das Bewerten von amerikanischen Optionen.

BinomialTreeOptions

Es werden Funktionen zum Bewerten von Optionen mittels Binomialbäumen geboten, zusätzlich stehen Funktionen zum Plotten von Binomialbäumen zur Verfügung. Im Abschnitt 4.3.2 wird die Verwendung dieser Funktionen anhand eines praktischen Beispiels demonstriert.

Modellbezeichnung
Roll,Geske und Whaley
Barone-Adesi und Whaley
Bjerksund und Stensland

Tabelle 1.14: fOptions - approximativ analytische Bewertungsmodelle für amerik. Optionen

MultipleExerciseOptions

Die Funktionen dieser Gruppe ermöglichen die Bewertung von so genannten *multiple exercise* Optionen mit Hilfe von analytischen Modellen. Die Funktionen, die zur Verfügung stehen, sind in Tabelle 1.15 aufgelistet. Zusätzlich ist der Name des Bewertungsmodells angeführt beziehungsweise der Name des Entwicklers.

Option	Bewertungsmodell
Executive Stock Option	Jennergren and Naslund
Forward Start Option	Rubinstein
Ratchet Option	Rubinstein bzw. Binomialbäume
Time Switch Option	Pechtl
Simple Chooser Option	Rubinstein
Complex Chooser Option	Rubinstein
Option on Options	Geske
Writer Extendible Option	Longstaff
Holder Extendible Option	Longstaff

Tabelle 1.15: fOptions - Bewertungsmodelle für Multiple Exercise Optionen

MultipleAssetsOptions

Die Optionen der Tabelle 1.16 können mit Funktionen dieser Gruppe bewertet werden. In der Tabelle sind auch die entsprechenden Bewertungsmodelle angeführt.

LookbackOptions

Diese Gruppe beinhaltet Funktionen zur Bewertung von pfadabhängigen *lookback* Optionen. Die Tabelle 1.17 zeigt die zur Verfügung stehenden Bewertungsmodelle.

Option	Bewertungsmodell
Two Assest Correlation Option	Zhang
Exchange-One-Assest-For-Another-Asset Option	Margrabe bzw. Binomialbäume
Exchange-On-Exchange Options	Carr
Portfolio Options	Binomialbäume
Rainbow Options	Rubinstein bzw. Binomialbäume
Spread Options	Binomialbäume
Dual Strike Option	Binomialbäume

Tabelle 1.16: fOptions - Bewertungsmodelle für Multiple Assets Optionen

Option	Bewertungsmodell
Floating Strike Lookback Options	Goldman,Sosin und Gatto
Fixed Strike Lookback Options	Conze und Viswanathan
Partial-Time Floating Strike Options	Heynen und Kat
Partial-time Fixed Strike Options	Heynen und Kat
Extreme Spread Options	Bermin

Tabelle 1.17: fOptions - Bewertungsmodelle für Lookback Optionen

BarrierOptions

Für die in Tabelle 1.18 aufgelisteten *barrier* Optionen stehen Funktionen zur Bewertung zur Verfügung.

Option	Bewertungsmodell
Single Barrier Option	Reiner und Rubinstein
Double Barrier Option	Ikeda und Kunitomo
Partial-Time Barrier Option	Heynen und Kat
Two-Asset Barrier Option	Heynen und Kat
Lookback Barrier Option	Bermin
Partial-Time-Two-Assest Option	Bermin
Soft Barrier Option	Hart und Ross

Tabelle 1.18: fOptions - Bewertungsmodelle für Barrier Optionen

BinaryOptions

Die in Tabelle 1.19 aufgelisteten Arten von *binary* Optionen können mit Funktionen dieser Gruppe bewertet werden. In der Tabelle ist ebenfalls das Bewertungsmodell angeführt.

Option	Bewertungsmodell
Gap Option	Reiner und Rubinstein
Cash-or-Nothing Option	Reiner und Rubinstein
Two-Asset-Cash-Or-Nothing Option	Heynen und Kat
Asset-Or-Nothing Option	Cox und Rubinstein
Supershare Option	Hakansson
Binary Barrier Option	Reiner und Rubinstein

Tabelle 1.19: fOptions - Bewertungsmodelle für Binary Optionen

AsianOptions

Für eine weitere Form von pfadabhängigen Optionen, den asiatischen Optionen, sind Funktionen für approximative Bewertungsmodelle nach Turnbull,Wakeman und Levy implementiert.

CurrencyTranslatedOptions

In dieser Gruppe stehen Funktionen zur Bewertung von *currency translated* Optionen zur Verfügung. Die Tabelle 1.20 fasst die Funktionen mit den jeweiligen Bewertungsmodellen zusammen.

Option	Bewertungsmodell
Equity Linked Foreign Exchange Option	Reiner
Quanto Option	Dravid, Richardson und Sun
Foreign Equity Option	Reiner
Takeover Foreign Exchange Option	Schnabel und Wei

Tabelle 1.20: fOptions - Bewertungsmodelle für Currency Translated Optionen

EBMDistribution

Diese Gruppe beinhaltet Funktionen für die Theorie der exponentiellen Brownschen Bewegung. So lassen sich Werte der Dichte- und Verteilungsfunktion der Log-Normalverteilung,

der Gammaverteilung, der reziproken Gammaverteilung und der Johnson Type-I Verteilung berechnen.

GammaFunctions

Es werden spezielle mathematische Funktionen zur Verfügung gestellt. So ist eine entsprechende Funktion für die Error-Funktion, Psi-Funktion, nicht vollständige Gammafunktion und das Pachhammersymbol implementiert.

HypergeometricFunctions

Die Gruppe umfasst Funktionen zum Berechnen von konfluenten hypergeometrischen Funktionen.

BesselFunctions

Mit den Funktionen dieser Gruppe kann die modifizierte Besselfunktion und ihre Ableitung berechnet werden.

HestonNandiGarchFit

Die Funktionen dieser Gruppe bieten die Möglichkeit die Preispfade des Heston und Nandi Optionsbewertungsmodells mittels einem GARCH(1,1) zu modellieren.

HestonNandiOptions

Es stehen Funktionen zur Optionsbewertung und Berechnung der Sensitivitäten mit Hilfe des Heston-Nandi Modells zur Verfügung.

LowDiscrepancy

Diese Gruppe fasst Funktionen zur Erzeugung von Zufallszahlen gemäß Sobol beziehungsweise Halton zusammen.

MonteCarloOptions

Es steht Funktionen zur Bewertung von Optionen mittels Monte Carlo Simulation zur Verfügung. Die Funktionen dieser Gruppe werden unter Abschnitt 4.4.1 zur Bewertung einer arithmetischen asiatischen Option verwendet.

1.7 fPortfolio

Die Informationen über das Package `fPortfolio` stammen aus der Dokumentation und für weiterführende Informationen sei auch auf diese verwiesen (vgl. Wuertz, 2006g).

`fPortfolio` umfasst Funktionen zur Berechnung von Portfolios gemäß Markowitz. Die Funktionsgruppen sollen nun näher charakterisiert werden.

AssetsModelling

Die Funktionen dieser Gruppe ermöglichen das Generieren von künstlichen multivariaten Datensätzen von Assets gemäß einer multivariaten Normalverteilung oder t-Verteilung. Zusätzlich wird eine Funktion geboten, die die Auswahl von individuellen Assets aus einem Portfolio mittels hierarchischen oder k-means Cluster ermöglicht.

DrawdownStatistics

Mit Hilfe der Funktionen dieser Gruppe lassen sich *drawdown* Statistiken berechnen. So können die Dichte, Verteilungsfunktion der maximalen *drawdown* Verteilung berechnet und Zufallszahlen erzeugt werden.

VaRModelling

Es stehen Funktionen zur Berechnung des Value-at-Risk (VaR) und davon abhängigen Risikomaßen für Portfolios zur Verfügung. Zusätzlich stehen Hilfsfunktionen für das Berechnen des maximalen Verlusts, der Gesamtrendite und zum Plotten der Renditen zur Verfügung.

TwoAssetsPortfolio

Die Funktionen ermöglichen das Berechnen der effizienten Hüllkurve für Portfolios aus zwei Assets, wobei kein Leerverkauf möglich ist.

MarkowitzPortfolio

Ähnlich wie die Funktionen der vorherigen Gruppe, können mit den Funktionen dieser Gruppe alle effizienten Portfolios von mehr als zwei Assets bei gegebener Rendite und Varianz berechnet und auch geplottet werden.

Überschneidungen mit anderen Packages

Nur bei der ersten Funktionsgruppe (AssestsModelling) sind Überschneidungen mit den Packages `sn,mtvnorm,corpcov` und `energy` bekannt.

1.8 fEconfin

Das Package `fEconfin` besteht aus Datensätzen, die in Beispielen der `Rmetrics` Dokumentation Verwendung finden. Auf eine Auflistung und Erläuterung der Datensätze wird aber verzichtet. Für weiterführende Informationen sei auf die Dokumentation verwiesen (vgl. Wuertz, 2006c).

Kapitel 2

Weitere ausgewählte R Packages

Neben den sehr umfangreichen `Rmetrics` Packages gibt es noch eine Vielzahl an Packages die für das Financial Engineering von Relevanz sein können. Nachfolgend werden relevante `R` Packages gruppenweise vorgestellt. Die Gruppen und die darin enthaltenen Packages stützen sich auf eine Auflistung von der `Empirical Finance CRAN Task View`, die auf der offiziellen `CRAN` Hompage [1] veröffentlicht ist.

2.1 Standard-Regresssions Modelle

- `nlme`
- `car`
- `lmtest`
- `strucchange`
- `urca`
- `uroot`
- `sandwich`
- `Rcmdr`
- `Zelig`
- `fMultivar`

[1] http://cran.r-project.org

2.2 Zeitreihenanalyse

- dse
- fracdiff
- longmemo
- FracSim
- tseries
- urca
- uroot
- ArDec
- MSBVAR
- vars
- dyn
- dynlm
- rwt
- wavelets
- waveslim
- wavethresh
- tseriesChaos
- tsDyn
- forecasting
- fSeries

2.3 Portfolio Management

- portfolio
- portfolioSim
- backtest
- fPortfolio

2.4 Optionsbewertung

- RQuantLib
- fOptions

2.5 Risiko Management

- VaR
- evd
- evdbayes
- evir
- extRemes
- ismec
- POT
- CreditMetrics
- mvtnorm
- copula
- fgac
- actuar
- fExtremes

2.6 Management und Import von Finanz- und Zeitdaten

- its
- zoo
- RBloomberg
- fCalendar

Teil II

Fallstudien

Im zweiten großen Teil dieser Arbeit soll anhand von Fallstudien der Einsatz des Softwarepakets `R` im Bereich des Financial Engineering demonstriert werden. Es kommen einige der in Teil I beschriebenen Packages zur Anwendung. Der Schwerpunkt liegt jedoch auf der selbständigen Implementierung von gängigen Finanzmodellen.

Dabei werden folgende Themengebiete behandelt: Zinsstruktur- und Credit-Spread-Schätzung, Optionsbewertung, Zinsderivate und Short-Rate Modelle. Um die Aktualität und die Relevanz der Fallbeispiele im Bereich des Financial Engineering zu gewährleisten, wurden die Gebiete in Absprache mit Professoren [2] der Wirtschaftsuniversität Wien getroffen.

In jedem Kapitel wird die relevante Theorie erarbeitet und auch möglichst auf eine Implementierung zugeschnitten. Werden in einem Kapitel selbst entwickelte Packages verwendet, wird auf die Funktionalität näher eingegangen. Bei den anderen verwendeten Packages wird auf diese Implementierungsdetails verzichtet und es steht die Verwendung im Vordergrund. Abgeschlossen wird jedes Kapitel mit einer Präsentation von Anwendungsbeispielen. Innerhalb der Kapitel sind bei den Anwendungsbeispielen jeweils die verwendeten `R` Funktionsaufrufe abgebildet. Im Anhang findet man zudem den Quellcode, für die im Rahmen dieser Arbeit entwickelten Packages.

Kapitel 3 behandelt die Zinsstrukturschätzung nach den Modellen von Svensson (1994) und Nelson und Siegel (1987). Die Bewertung von Optionen mit numerischen Verfahren, wie Binomialbäumen, finiten Differenzen und Monte Carlo Simulation wird in *Kapitel 4* erläutert und demonstriert. In *Kapitel 5* werden einfache Zinsderivate, mit im Rahmen dieser Arbeit implementierten Short-Rate Modelle, bewertet.

Alle Berechnungen wurden mit `R` Version 2.31 unter Mac OS X 10.4 durchgeführt.

[2] Univ.-Prof. Dr. Gregor Dorfleitner - Abteilung für Financial Engineering und Derivate, ao.Univ.Prof.Dipl.-Ing.Dr.techn. Klaus Pötzelberger - Department of Statistics and Mathematics, ao.Univ.Prof.Dipl-Ing.Dr.techn. Michael Hauser -Department of Statistics and Mathematics, Univ. Ass. Dr. Rainer Jankowitsch, Institut für Kreditwirtschaft

Kapitel 3

Zinsstruktur und Credit-Spread Schätzung

Im Bereich der festverzinslichen Wertpapiere kommt der Zinsstrukturkurve eine zentrale Bedeutung zu. So benötigen viele Preis- und Risikomanagementmodelle die Zinsstruktur beziehungsweise die Credit-Spread-Kurven als Input. Für keinen Anleihenmarkt lässt sich die komplette Zinsstrukturkurve für beliebige Laufzeiten beobachten. Daher besteht die Notwendigkeit die Zinsstruktur über Schätzmethoden zu generieren. Die Schätzung basiert dabei auf den beobachtbaren Charakteristika von Anleihen auf den Märkten.

Im folgenden Punkt werden die verwendeten Modelle vorgestellt, wichtige Informationen zur Implementierung gegeben und die Ergebnisse präsentiert. Hervorzuheben ist, dass diesem Kapitel ein Projekt im Rahmen der Lehrveranstaltung "Management Science Lab -Analyse und Entscheidung im Finanzmanagement", welches im Wintersemester 2005/2006 in Kooperation mit Raiffeisen Capital Management durchgeführt wurde, zugrunde liegt. Dabei wurde ein `R` Package erstellt, das in weiterer Folge von Mag. Robert Ferstl und im Rahmen dieser Diplomarbeit erweitert und verbessert wurde.

Die implementierten Modelle beschränken sich auf die weit verbreiteten Ansätze von Nelson und Siegel und Svensson. Abgesehen von diesen gibt es eine große Anzahl von Modellen für die Zinsstrukturschätzung. Für weitere Modelle sei auf den Artikel von Bliss (1996) verwiesen, der in seinem Artikel "Testing term structure estimation methods" unter anderem die Modelle von Fama-Bliss, McCulloch Cubic Spline, Fisher- Nychka-Zervos Cubic Spline und das erweiterte Nelson Siegel Modell behandelt.

3.1 Allgemeine Grundlagen

Die Bestimmung der Spot-Rates von Kuponanleihen könnte theoretisch über Nullkupon-Anleihen erfolgen (*zero coupon bonds*), die genau die Lauzeiten haben, die den Fristigkeiten der Zahlungen der Kuponanleihe entsprechen. Diese Situation trifft jedoch sehr selten zu. Vielmehr kann die Zinsstruktur über beobachtbare Daten von Kuponanleihen auf Märkten geschätzt werden. Beobachtbar sind die Fristigkeiten der Kuponzahlungen m_t, die Laufzeit der Anleihe m_n, der Betrag der Kuponzahlungen c_t und der quotierte Preis (*clean price*) p^c (wobei $t = 1...n$). Ein Käufer der Anleihe hat zusätzlich zum quotierten Preis noch die anteiligen Stückzinsen a (*accrued interest*) zu zahlen, die seit der letzten Zahlung angefallen sind. Dieser Preis wird dann als *dirty price* p^d bezeichnet. Die weiteren Ausführungen stüzten sich auf Bolder und Streliski (1999).

Die Zinsstruktur kann überlicherweise mittels einer Diskontierungsfunktion, Spot-Rate Funktion oder einer Forward-Rate Funktion beschrieben werden. Für die weiteren Ausführungen sind die ersten beiden Funktionen von Bedeutung.

Der Preis einer Anleihe j ergibt als (stetig) diskontierter Cashflow:

$$p^c + a = \sum_{t=1}^{n} c_t e^{-s_t m_t} \tag{3.1}$$

Wobei die Spot-Rate s_t der Zinssatz ist, der für die Abzinsung des Cashflows einer Nullkupon-Anleihe mit der Laufzeit von m_t verwendet wird.

Eine weitere Beziehung zwischen den Cashflows und dem Preis einer Anleihe ergibt sich, wenn alle Zahlungen c_t mit dem selben Zinssatz abgezinst werden. Dieser Zinssatz wird als Rendite y beziehungsweise *yield to maturity* bezeichnet.

$$p^c + a = \sum_{t=1}^{n} c_t e^{-y m_t} \tag{3.2}$$

Eine äquivalente Darstellung des Preises einer Anleihe ergibt unter der Verwendung von Diskontierungsfaktoren $d_t = \delta(m_t) = e^{-s_t m_t}$. Die kontinuierliche Diskontierungsfunktion $\delta(\cdot)$ erhält man durch Interpolation der Diskontierungsfaktoren.

$$p^c + a = \sum_{t=1}^{n} c_t \delta(m_t) \tag{3.3}$$

3.2 Verwendete Modelle

3.2.1 Das Modell von Nelson und Siegel

Wird zwischen den Spot-Rates und den Fristigkeiten eine Funktion unterstellt kann die Zinsstruktur geschätzt werden. Nelson und Siegel gehen in ihrem Modell von der grundlegenden Annahme aus, dass Spot-Rates durch Differentialgleichungen errechnet werden können. Führt man diese Überlegung weiter, sind die Forward-Rates die Lösung dieser Differentialgleichungen (vgl. Nelson und Siegel, 1987, S.475).

Nelson und Siegel definieren die Spot-Rates $s(\mathbf{m};\mathbf{b})$ und Forward-Rates $f(\mathbf{m};\mathbf{b})$ als Differentialgleichung zweiter Ordnung in Abhängigkeit von der Fristigkeit m und einem Parametervektor $\mathbf{b}$.

$$s(\mathbf{m};\mathbf{b}) = \beta_0 + \beta_1 \frac{1 - e^{(-\frac{m}{\tau_1})}}{\frac{m}{\tau_1}} + \beta_2 \left(\frac{1 - e^{(-\frac{m}{\tau_1})}}{\frac{m}{\tau_1}} - e^{(-\frac{m}{\tau_1})} \right) \tag{3.4}$$

$$f(\mathbf{m};\mathbf{b}) = \beta_0 + \beta_1 e^{(-\frac{m}{\tau_1})} + \beta_2 \frac{m}{\tau_1} e^{(-\frac{m}{\tau_1})} \tag{3.5}$$

Der Verlauf der Spot-Rate beziehungsweise Forward-Rate Kurve wird durch den Parametervektor $\mathbf{b} = (\beta_0, \beta_1, \beta_2, \tau_1)$ determiniert (wobei β_0 und $\tau_1 > 0$) . Die Parameter können dabei wie folgt interpretiert werden.

Geht die Fristigkeit gegen unendlich ist der Grenzwert der Spot-Rate Funktion β_0. Geht sie gegen 0 konvergiert die Spot-Rate Funktion gegen die Summe aus β_0 und β_1. Analoges gilt für die Forward-Rate Funktion. Nelson und Siegel (1987) interpretieren die Parameter $\beta_0, \beta_1, \beta_2$ als lang-, mittel- und kurzfristige Komponente der Spot-Rate beziehungsweise Forward-Rate Kurve. Diebold und Li (2003) erweitern die gängige Interpretation von Nelson und Siegel. So bestimmt der Parameter β_0 das Level der Zinsstrukturkurve und β_1 den Anstieg. Einige Autoren wie Frankel und Lown (1994) definieren den Anstieg der Zinsstrukturkurve mit $-\beta_1$ (vgl. auch Diebold und Li, 2003, S.5).

Die Abbildung 3.1 soll den Beitrag der einzelnen Terme der Spot-Rate Funktion zur gesamten Forward-Rate Kurve zeigen (vgl. Nelson und Siegel, 1987, S.477 - figure 2). Die Elemente des Parametervektors $\mathbf{b}$ wurden alle auf 1 gesetzt.

3.2.2 Das Modell von Svensson

Svensson (1994) erweiterte das Modell von Nelson und Siegel durch die Aufnahme von weiteren Parametern hinsichtlich der Flexibilität und der Anpassungsfähigkeit (vgl. Svensson, 1994, S.8).

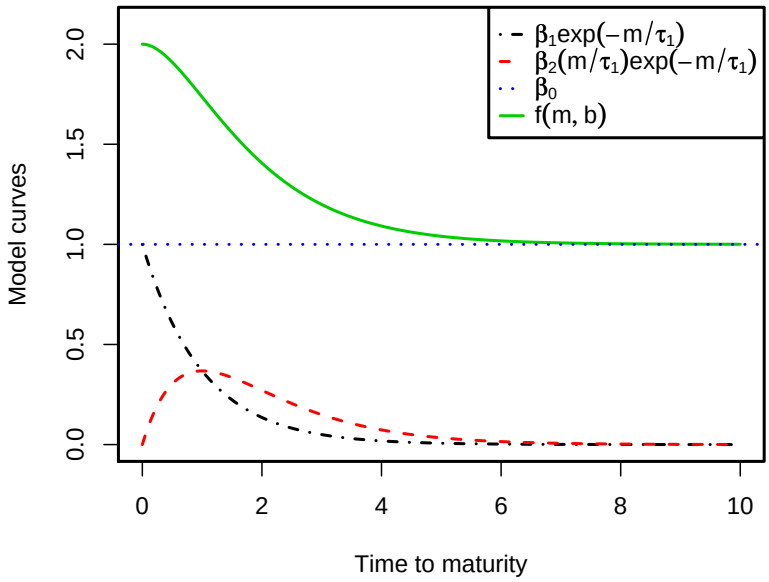

Abbildung 3.1: Beitrag der einzelnen Parameter zur Nelson-Siegel Forward-Rate Funktion

Die Spot-Rate- $s(\mathbf{m};\mathbf{b})$ und die Forward-Rate Funktion $f(\mathbf{m};\mathbf{b})$ sind in diesem Modell wie folgt definiert:

$$s(\mathbf{m};\mathbf{b}) = \beta_0 + \beta_1\left(\frac{1-e^{(-\frac{m}{\tau_1})}}{\frac{m}{\tau_1}}\right) + \beta_2\left(\frac{1-e^{(-\frac{m}{\tau_1})}}{\frac{m}{\tau_1}} - e^{(-\frac{m}{\tau_1})}\right) + \beta_3\left(\frac{1-e^{(-\frac{m}{\tau_2})}}{\frac{m}{\tau_2}} - e^{(-\frac{m}{\tau_2})}\right), \quad (3.6)$$

$$f(\mathbf{m};\mathbf{b}) = \beta_0 + \beta_1 e^{(-\frac{m}{\tau_1})} + \beta_2\frac{m}{\tau_1}e^{(-\frac{m}{\tau_1})} + \beta_3\frac{m}{\tau_2}e^{(-\frac{m}{\tau_2})}, \quad (3.7)$$

mit $\mathbf{b} = (\beta_0, \beta_1, \beta_2, \beta_3, \tau_1, \tau_2)$.

Die Parameter haben folgende Bedeutung (vgl. Bolder und Streliski, 1999, S.7):

- β_0 ist der Grenzwert der Spot-Rate beziehungsweise Forward-Rate Funktion, wenn die Laufzeit gegen ∞ geht ($\beta_0 > 0$).
- β_1 bestimmt den Startwert der Kurve hinsichtlich der Abweichung vom Grenzwert.

Auch wird die Geschwindigkeit mit der sich die Kurve dem langfristigen Trend anpasst, determiniert.

- τ_1 bestimmt den ersten Buckel oder die U-Form der Kurve ($\tau_1 > 0$).
- β_2 bestimmt das Ausmaß und die Richtung des Buckels. Wenn $\beta_2 > 0$ ist, tritt der Buckel zu τ_1 auf. Ist $\beta_2 < 0$, hat die Kurve bei τ_1 eine U-Form.
- τ_2 bestimmt den zweiten Buckel oder die U-Form der Kurve ($\tau_2 > 0$).
- β_3 kann analog zu β_2 interpretiert werden und bestimmt das Ausmaß und die Richtung des zweiten Buckels.

Die folgende Abbildung zeigt den Beitrag der einzelnen Terme zur Svensson Forward-Rate Funktion (vgl. Bolder und Streliski, 1999, S.8 - figure 1):

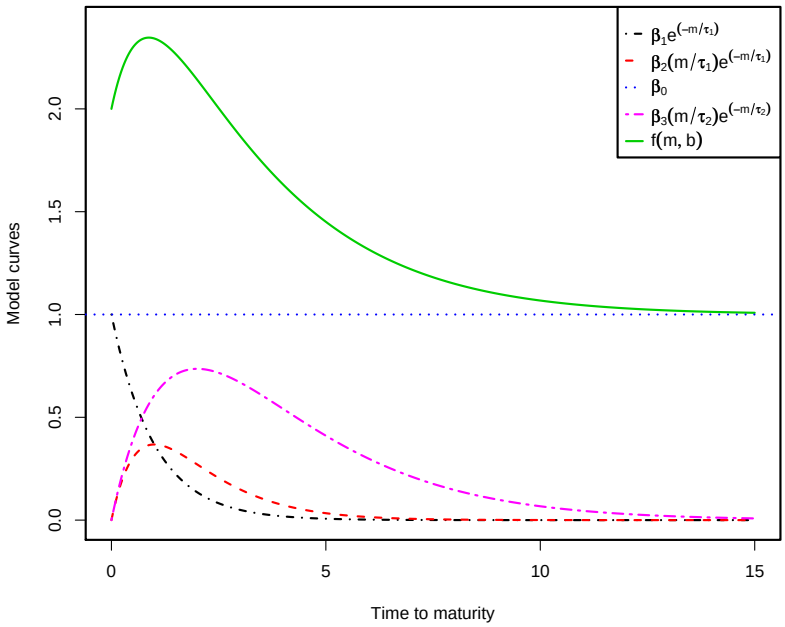

Abbildung 3.2: Beitrag der einzelnen Parameter zur Svensson Forward-Rate Funktion

3.2.3 Modelle zur Credit-Spread Schätzung

Wie bereits in der Einleitung dieses Kapitels erwähnt, benötigen viele Credit-Risk Modelle eine präzise Zinsstrukturkurve, meist der verschiedenen Ratingklassen , als Input. Traditionell gesehen werden Credit-Spreads durch die Subtraktion von unabhängig geschätzten risikolosen und risikobehafteten Bestandteilen der Zinsen gewonnen. In dem Artikel "Parsimonious

estimation of credit spreads" von Jankowitsch und Pichler (2004) werden im wesentlichen zwei Modelle zur Berechnung von Credit-Spreads vorgestellt. Das in dieser Arbeit verwendete Modell wird in dem Artikel als Singlecurve Ansatz bezeichnet (vgl. Jankowitsch und Pichler, 2004, S.8f).

Bei dem Singlecurve Ansatz werden die Spot-Rate Kurven gemäß dem Modell von Nelson und Siegel oder Svensson getrennt geschätzt. Anschließend wird die Referenzkurve von den übrigen Spot-Rate-Kurven subtrahiert. Daraus resultiert jeweils eine Credit-Spread Kurve. Die Spot-Rate Kurve kann dabei zum Beispiel für Anleihen von Ländern, Ländergruppen oder Ratingklassen geschätzt werden.

Der Credit-Spread wird durch die Subtraktion der Referenzkurve von der Spot-Rate Kurve einer Vergleichskurve (eines Landes, einer Ratingklasse) errechnet.

$$cs_j(\mathbf{m}) = s_j(\mathbf{m}, \mathbf{b}) - s_{ref}(\mathbf{m}, \mathbf{b}) \tag{3.8}$$

Wobei:
$cs_j(\mathbf{m}) \ldots$ Credit-Spread zum Beispiel zwischen Land j und Referenzland ref
$s_j(\mathbf{m}, \mathbf{b}) \ldots$ Spot-Rate Kurve für Land j mit der Fristigkeit m
$s_{ref}(\mathbf{m}, \mathbf{b}) \ldots$ Spot Rate Kurve für das Referenzland mit der Fristigkeit m

Die Spot-Rate Funktionen entsprechen der unter Gleichung 3.4 beziehungsweise 3.6 definierten Form.

3.3 Das Optimierungsproblem

Das Ziel der Optimierung ist ein optimales Parameterset $\mathbf{b}$ zu finden, mit dessen Hilfe zero-coupon yield Kurven aus Marktdaten von Anleihen einer bestimmten Gruppe (zum Beispiel Länder oder Ratingklassen) konstruiert werden können. Die Optimierung erfolgt über die Minimierung der Preis- oder Yieldabweichung. Um eine leichte Implementierung zu gewährleisten wird in weiterer Folge das Optimierungsproblem in einer Matrixnotation dargestellt.

3.3.1 Formulierung der Matrizen und Vektoren

Fristigkeitsmatrix M:

$$\mathbf{M}_{[n \times m]} = \{m_{ij}\} \tag{3.9}$$

Die Anzahl der Zeilen n der Matrix ergibt sich als die Anzahl der Cashflows der Anleihe j mit der längsten Restlaufzeit. Für jede Anleihe j existiert eine Spalte mit den entsprechenden

Fristigkeiten der Kuponzahlungen. Fristigkeiten nach der Fälligkeit der Anleihe j werden mit Nullen aufgefüllt, um eine gleiche Länge der einzelnen Fristigkeitsvektoren zu gewährleisten. Ein Element m_{ij} der Matrix repräsentiert somit die Fristigkeit des i-ten Cashflows der j-ten Anleihe.

Cashflow Matrix C:

$$\mathbf{C}_{[n\times m]} = \{c_{ij}\} \tag{3.10}$$

Die Cashflow Matrix ist analog zur Fristigkeitsmatrix aufgebaut. Ein Element c_{ij} der Matrix repräsentiert somit den i-ten Cashflow der j-ten Anleihe. Zu beachten ist, dass der letzte Cashflow der Anleihe j auch die Tilgung umfasst.

Diskontfaktorenmatrix D:

$$\mathbf{D}_{[n\times m]} = \{d_{ij}\} \tag{3.11}$$

Auch die Diskontfaktorenmatrix ist analog zur Fristigkeitsmatrix definiert. Das Element d_{ij} repräsentiert den Diskontfaktor für den i-ten Cashflow für die j-te Anleihen. Ein Diskontfaktor d_{ij} ergibt sich wie folgt:

$$d_{ij} = e^{-m_{ij}s(m_{ij},b)}$$

Dabei ist $s(m_{ij}, \mathbf{b})$ die unter Gleichung 3.4 beziehungsweise 3.6 definierte Spot-Rate Funktion gemäß Nelson und Siegel beziehungsweise Svensson.

Quotierter Preis Vektor (clean price vector) $\mathbf{p_c}$:

$$\mathbf{p^c}_{[1\times m]} = (p_j^c) \tag{3.12}$$

p_j^c ist der quotierte Preis der j-ten Anleihe ($j = 1...m$).

Vektor der Stückzinsen a:

$$\mathbf{a}_{[1\times m]} = (a_j) \tag{3.13}$$

Für die Berechnung der Stückzinsen sind verschiedene Konventionen üblich. Eine einfache Möglichkeit ist die Stückzinsen wie folgt zu berechnen:

$$a_j = \frac{Anzahl\ der\ Tage\ seit\ der\ letzten\ Kuponzahlung}{Anzahl\ der\ Tage\ der\ Kuponzahlungsperiode} c_j.$$

a_j = Stückzinsen für Bond j, die seit der letzten Kuponzahlung angefallen sind.

Dirty Price Vektor $\mathbf{p^d}$:

$$\mathbf{p^d}_{[1\times m]} = (p_j^d) \tag{3.14}$$

Der Dirty Price Vektor setzt sich aus der Summe des quotierten Preis Vektors und dem Stückzinsen Vektor zusammen. Der Dirty Price Vektor beeinhaltet somit alle quotierten Preise plus den anteiligen Stückzinsen für alle Anleihen j.

$$\mathbf{p^d}_{[1\times m]} = \mathbf{p^c}_{[1\times m]} + \mathbf{a}_{[1\times m]}$$

Vektor mit Gewichten ω:

$$\omega_{[1\times m]} = (\omega_j) \tag{3.15}$$

ω_j ist das Gewicht für die Anleihe j mit der MacCauly Duration D_j:

$$\omega_j = \frac{\frac{1}{D_j}}{\sum_{i=1}^{m} \frac{1}{D_i}}.$$

Die Duration einer Anleihe j ist der folgende gewichtete Durchschnitt:

$$D_j = \frac{\mathbf{C}_{[n\times j]} \left(\mathbf{D}_{[n\times j]} \cdot \mathbf{M}_{[n\times j]}\right)^t}{\mathbf{C}_{[n\times j]} \left(\mathbf{D}_{[n\times j]}\right)^t} \tag{3.16}$$

Wobei $(\cdot)$ eine elementweise Multiplikation der Matrizen und $()^t$ eine transponierte Matrix beziehungsweise einen transponierten Vektor bezeichnet.

3.3.2 Formulierung der Zielfunktion und Nebenbedingungen

Die Zielfunktion ergibt sich über die Minimierung der Summe der quadrierten Preisabweichungen. Die Optimierung wäre auch über die Minimierung der Yieldabweichungen möglich. Diese Variante wird allerdings nicht berücksichtig, da der Rechenaufwand bei der Optimierung wesentlich steigt (vgl. Bolder und Streliski, 1999, S.10 ff). Exakter formuliert erfolgt die Optimierung über die Minimierung der gewichteten quadrierten Preisabweichungen. Die Gewichte sind unter Gleichung 3.15 definiert worden und vermeiden die Heteroskedastizität der Preisabweichungen (*price errors*) (vgl. Bolder und Streliski, 1999, S. 11).

Die Zielfunktion F ergibt sich mit:

$$F = \left(\left(\iota_{[1\times n]} \left[\mathbf{C}_{[n\times m]} \cdot \mathbf{D}_{[n\times m]} \right] - \mathbf{p^d}_{[1\times m]} \right)^2 \ \omega_{[1\times m]} \ \iota_{[m\times 1]} \right). \tag{3.17}$$

Wobei $(\cdot)$ eine elementweise Multiplikation der Matrizen beziehungsweise Vektoren bezeichnet und der Vektor ι aus lauter Einsen besteht. Der optimale Parametervektor $\mathbf{b}$ wird über die Minimierung der Zielfunktion bestimmt. Das Optimierungsproblem lässt sich wie folgt definieren:

$$\min_{\mathbf{b}} F(\mathbf{b}). \tag{3.18}$$

Der Parametervektor unterscheidet sich je nach gewählter Spot-Rate Funktion. Im Falle von Nelson Siegel : $\mathbf{b} \in \mathbb{R}^4$ beziehungsweise $\mathbf{b} \in \mathbb{R}^6$ im Falle von Svensson. Allerdings müssen auch Nebenbedingungen der Parameter beachtet werden. Diese wurden bereits unter Abschnitt 3.2.1 und 3.2.2 ausgeführt. So muss bei der Svensson Spot-Rate Funktion $\beta_0, \tau_1, \tau_2 > 0$, bei der Nelson Siegel Spot-Rate Funktion $\beta_0, \tau_1 > 0$ sein. Für alle Parameter scheinen Einschränkungen nach beiden Seiten sinnvoll um den Lösungsraum auf möglichst realistische Ergebnisse einzuschränken. Diese Nebenbedingungen sind stark vom Datensatz abhängig und deshalb wird an dieser Stelle nur die allgemeine Formulierung angeführt. Bolder und Streliski (1999) führen in ihrem Artikel "Yield Curve Modelling at the Bank of Canada" für kanadische Staatsanleihen die Parametergrenzen explizit an.

Das Optimierungsproblem kann damit wie folgt definiert werden:

$$\min_{\mathbf{b}\in G} F(\mathbf{b}), \tag{3.19}$$

$$\mathbf{b} \in G = \{\mathbf{b} \in \mathbb{R}^4 \text{bzw.} \mathbb{R}^6 : \mathbf{ug} \leq \mathbf{b} \leq \mathbf{og}\}. \tag{3.20}$$

Wobei **ug** die untere und **og** die obere Grenze für die Optimierung der Parameter bezeichnet.

3.4 Spezialprobleme der Optimierung

Bei der numerischen Optimierung kommt der Wahl der Startparameter eine zentrale Rolle zu. Bolder und Streliski (1999) weisen darauf hin, dass bei der Optimierung neben dem globalen Minimum noch multiple lokale Minima existieren. Wie anhand der folgenden Ausführungen gezeigt werden soll, ist die Lösung der Optimierung äußesrt sensibel auf die Wahl der Startparameter.

Eine Analyse von italienischen Staatsanleihen zeigt, dass die Startparameter einen wesentlichen Einfluss auf die Optimierung haben. Die verwendeten Daten sind im `R` Package `termstrc` inkludiert. Mehr Details zum Package und der Handhabung werden unter Punkt 3.5 präsentiert. An dieser Stelle soll noch erwähnt werden, dass die Optimierung in `R` mittels der Funktion `nlminb` durchgeführt wird. Diese Funktion basiert auf PORT Routinen (vgl. Gray, 1990).

Die folgende Abbildung der Zinsstruktur für italienische Staatsanleihen, geschätzt nach der Methode von Svesson, zeigt zwei Schätzungen die auf unterschiedlichen Startparametern basieren, die Punkte stellen die berechneten Renditen dar. Die schwarze Kurve wurde mit Startparametern geschätzt die auf einen Ansatz von Geyer und Mader (1999) basieren. Für die rote Kurve wurden Startparameter gewählt, die auf einer rechenaufwändigen Analyse basieren. Mögliche Startparameteralgorithmen werden in Folge noch behandelt. Abbildung 3.4 lässt allerdings schon erkennen, dass ein entsprechender Aufwand in die Wahl der Startparameter investiert werden muss und diese für jeden Datensatz entsprechend kalibriert werden müssen.

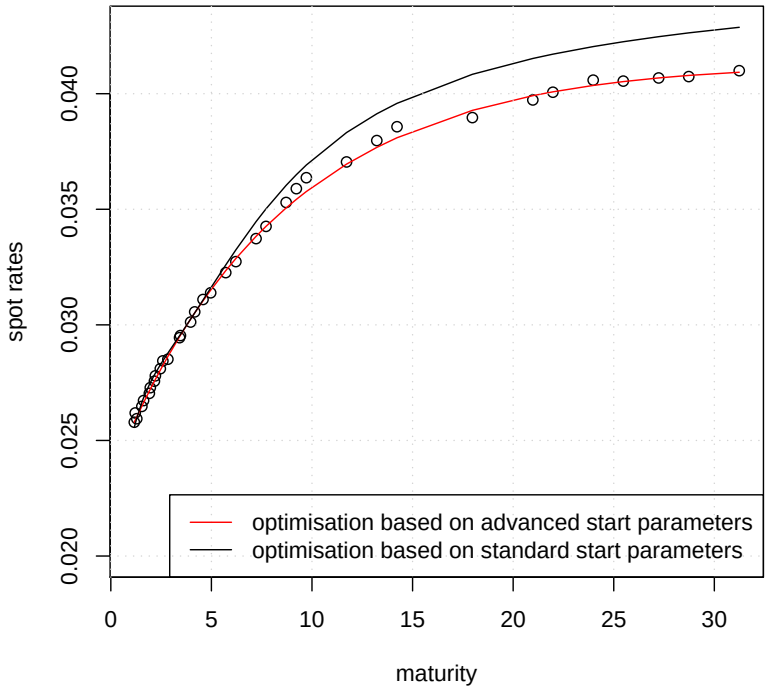

Abbildung 3.3: Zinsstrukturschätzung in Abhängigkeit von der Wahl der Startparameter

3.4.1 Startparameteransätze

Wie bereits dargestellt, kann durch eine gute Startparameterwahl die Güte der Schätzung wesentlich verbessert werden. Auch unter dem Gesichtspunkt, dass die Zielfunktion nicht streng konvex ist, muss der Startparameterwahl besondere Aufmerksamkeit gewidmet werden. Allerdings kann durch die Wahl eines guten Optimierungsalgorithmus eventuell die Sensitivität der Schätzung bezüglich der Startparameterwahl reduziert werden. In `R` stehen allerdings nur wenige Optimierer zur Verfügung.

Die folgenden Ansätze sollen einen kurzen Überblick geben, welche Möglichkeiten bestehen gute Startparameter zu generieren.

Bolder und Streliski (1999) stellen in ihrem Artikel zwei Startparameteransätze vor. Beim ersten wird für eine Matrix von Startparametern für jedes Parameterset eine Optimierung durchgeführt. Es können allerdings, aufgrund des erheblichen Rechenaufwands, nicht alle Parameterkombinationen berücksichtigt werden. So werden β_0 und β_1 nicht variiert, sondern auf fixe Werte gesetzt. Der Startwert für β_0 wird auf die Rendite, der Anleihe mit der längsten Restlaufzeit im Datensatz gesetzt. Der Startwert für β_1 wird mit der Renditendifferenz zwischen der Anleihe mit der längsten und kürzesten Restlaufzeit im Datensatz festgelegt. Alle anderen Parameter werden variiert. Zur Optimierung werden zwei verschiedene Algorithmen verwendet (*SQP-sequential quadratic programming* und die Nelder/Mead Simplex Methode). Beim zweiten Ansatz wird der Parametervektor in zwei Gruppen eingeteilt, in die β‘s (die linearen Parameter) und die τ‘s (die nicht-linearen Parameter). Bei der Optimierung wird nun jeweils eine Gruppe der Parameter konstant gehalten und die andere geschätzt. Dadurch kann die Konvergenzgeschwindigkeit des Optimierungsalgorithmus wesentlich verbessert werden.

Ferenczi (2005) entwickelte einen Algorithmus zur Optimierung unter Nebenbedingungen mit dünnen Gittern der auf dem HCP Algorithmus von Novak und Ritter (1996) basiert und für die Wahl der Startparameter verwendet werden kann. Der große Vorteil dieses Algorithmus ist, dass er zum globalen Minimum der Zielfunktion konvergiert.

Geyer und Mader (1999) verfolgen in ihrer Analyse einen ähnlichen Ansatz wie Bolder und Streliski (1999). So werden die Startparameter β_0 und β_1 analog gewählt. Die übrigen Parameter werden variiert. Der Startparametervektor mit dem besten Ergebnis, d.h. der geringsten Preis- beziehungsweise Yieldabweichung wird für die Optimierung als Startparametervektor herangezogen. Geyer und Mader weisen darauf hin, dass speziell, wenn optimale Parameter von vorangegangenen Schätzungen existieren, es sinnvoll ist, diese als Startparameter zu verwenden.

3.5 Schätzen der Zinsstruktur und der Credit Spreads mit dem R Package termstrc

In diesem Abschnitt wird das Package `termstrc` vorgestellt, das im Rahmen dieser Arbeit entstanden ist. Es sind alle in diesem Kapitel vorgestellten Modelle implementiert worden. Der Quellcode befindet sich im Anhang A. Die Implementierung stützt sich auf die Ausführungen in Abschnitt 3.3. Es werden die benötigten Daten aus dem Datensatz eingelesen und die entsprechenden Berechnungen durchgeführt. Als Ergebnis werden die geplotteten Kurven und die Parameterschätzung ausgegeben.

Im Folgenden wird das Package näher vorgestellt, es werden Details zur Implementierung präsentiert und die Funktionalität erklärt. Abschließend wird anhand von Beispielen die Verwendung demonstriert.

3.5.1 Informationen über das Package termstrc

Für die Implementierung wurde eine objektorientierter Ansatz gewählt und auf die `S3` Klassen von `R` zurückgegriffen. So wurde die Klasse `termstrc_singlecurve` definiert. Als Methoden stehen `print, summary` und `plot` zur Verfügung. Nähere Informationen zu `S3` Klassen und Methoden sind unter R Development Core Team (2006) beziehungsweise Ligges (2005) zu finden. Nähere Informationen zur Verwendung werden unter dem Punkt Funktionalität gegeben. Nun soll die Struktur des Datensatzes behandelt werden, der für die Schätzung benötigt wird.

Struktur der Daten

Das Package `termstrc` enthält zwei Datensätze in einer Listenstruktur. Sollen Schätzungen mit anderen Daten durchgeführt werden, muss die Struktur dieses Datensatzes dieser spezifischen Listenstruktur entsprechen. Der Datensatz `eurobonds` enthält Daten der Staatsanleihen von Österreich, Deutschland, Italien und Ungarn. Im Datensatz `corpbonds` sind Daten von Anleihen verschiedenster Ratingklassen enthalten (AAA, AA+, AA, AA-,A+, A,A-, BBB+, BBB, BBB-). Für jede Anleihe muss die ISIN Nummer, das Emissionsdatum, das Ende der Laufzeit, die Kuponrate, der quotierte Preis, die Stückzinsen, der Betrag der Cashflows, der Zeitpunkt der Cashflows und eine Zeitangabe für die quotierten Preise vorhanden sein.

Funktionalität

Die komplette Funktionalität des Packages wird im Prinzip durch die Funktion `termstrc_estim` zur Verfügung gestellt. Deshalb soll diese genauer erläutert werden. Die Funktion ist dabei

wie folgt aufgebaut:

```
termstrc_estim(countries, bonddata, maturity_spectrum, method, fit,
                          weights, startparam,control)
```

Die Bedeutung der Parameter ist der Tabelle 3.1 zu entnehmen.

Parameter	Bedeutung	Optionen/Beispiel
`countries`	Vektor mit Gruppen von Anleihen	`c("AUSTRIA", "ITALY")`
`bonddata`	Angabe des Datensatzes	`eurobonds`
`maturity_spectrum`	Laufzeitenspektrum in Jahren	`"all"` oder z.B. `c(1,20)`
`method`	Methode von Nelson/Siegel od.Svensson	`"Nelson/Siegel" od. "Svensson"`
`fit`	Optimierungsart	`"prices"` od. `"yields"`
`weights`	Gewichtung mit der Duration	`"none"` od. `"duration"`
`startparam`	Startparametervektor	`b<- matrix(rep(1,8),nrow=2)`
`control`	Parameter für Funktion `nlminb`	`control=list(eval.max=1000)`

Tabelle 3.1: Bedeutung der Parameter

Das erste Land im `countries` Vektor wird als Referenzland für die Credit-Spread Berechnung herangezogen. Der Parameter `fit` ist auch dafür vorgesehen, dass eine Optimierung über die Minimierung der Renditenabweichungen durchgeführt werden kann. Ein spezifischer Startparameteransatz wurde nicht implementiert. Die Angabe der Startparameter erfolgt über eine Matrix, wobei eine Zeile die Startparameter einer Gruppe von Anleihen (Länder, Ratingklasse) repräsentiert.

Programmoutput

Das Ergebnis der Schätzung besteht primär aus den geplotteten Grafiken für die Zinsstrukturkurven der spezifizierten Länder (Ratingklassen) und der Grafik mit den Credit-Spreads, sowie den ausgegebenen optimalen Parametern der Schätzung. Zusätzlich sind über die `summary` Methode Fehlerstatistiken und Informationen zur Konvergenz des Optimierungsalgorithmus ersichtlich. Als Fehlerstatistiken stehen der RSME[1](*root mean square error*) und AABSE [2](*average absolute error*) der Preis- beziehungsweise Yieldabweichung zur Verfügung.

Mittels der `print` Methode erfolgt die Ausgabe der optimalen Parameter pro Land oder Ratingklasse. Die `plot` Methode ist für den entsprechenden grafischen Output notwendig.

[1] $RSME = \sqrt{\sum_{i=1}^{N} \frac{(\hat{P}_i - P_i)^2}{N}}$

[2] $AABSE = \sum_{i=1}^{N} \frac{|\hat{P}_i - P_i|}{N}$

Zusätzlich stehen für weitere Berechnungen die Cashflows, die Laufzeiten, die quotierten Preise, die geschätzten Preise und die berechneten Renditen, in kompakter Listenform, zur Verfügung.

3.5.2 Anwendungsbeispiele

An dieser Stelle soll anhand einiger Anwendungsbeispiele die Funktionalität des Packages `termstrc` demonstriert und die Ergebnisse interpretiert werden. Die Schätzungen werden jeweils nach der Nelson und Siegel und Svensson Methode durchgeführt. Beim ersten Beispiel wird die Eingabe der Parameter explizit angegeben, in weiterer Folge wird darauf verzichtet und es werden nur die Ergebnisse der Schätzung präsentiert.

Schätzen der Zinsstruktur für die Länder Deutschland, Österreich und Italien

Die Eingabe in `R` sieht wie folgt aus:

```
data(eurbonds)
countries <- c("GERMANY", "AUSTRIA", "ITALY")
bonddata <- eurobonds
maturity_spectrum <- "all"
method = "Nelson/Siegel"
fit = "prices"
weights = "duration"
control=list(eval.max=100000)

startparam <- matrix(c(0.02547394, -0.012162592, -0.02547394,    1,
                       0.02611532, -0.011367422, -0.02611532,    1,
                        0.02578871, -0.015207250, -0.02578871,    1 ),
                       nrow=3,ncol=4,byrow=T)

myres<- termstrc_estim(countries, bonddata, maturity_spectrum,
    method, fit, weights, startparam=startparam,control)
```

Die Schätzung nach der Methode von Nelson und Siegel liefert folgende optimale Parameter und Fehlerstatistiken.

```
----------------------------------------------------
Parameters for single-curve estimation:

Method: Nelson/Siegel
Fitted: prices
Weights: duration
```

```
-----------------------------------------------------

           GERMANY     AUSTRIA       ITALY
beta_0  0.04153832  0.04228676  0.04599300
beta_1 -0.01699020 -0.01813795 -0.02089901
beta_2 -0.01326502 -0.01208279 -0.01874076
tau_1   2.20819586  2.48874374  2.36039421

-----------------------------------------------------
Goodness of fit tests:
-----------------------------------------------------

                  GERMANY      AUSTRIA        ITALY
RMSE-Prices  0.2852092109 0.0606758664 0.1588754369
AABSE-Prices 0.1629764768 0.0416447015 0.1041235559
RMSE-Yields  0.0004808922 0.0003668975 0.0007122635
AABSE-Yields 0.0004177424 0.0002694297 0.0006036436


        Convergence info
GERMANY "relative convergence (4)"
AUSTRIA "relative convergence (4)"
ITALY   "relative convergence (4)"
```

Nachfolgend ist der grafische Output dargestellt. Die gemeinsame Grafik aller Zinsstrukturkurven (Abbildung 3.4) zeigt sehr schön, dass die deutsche und österreichische Kurve beinahe deckungsgleich sind. Bis zu einer Laufzeit von 8 Jahren kann bei einer Schätzung nach der Methode von Nelson und Siegel von einem konstanten Spread ausgegangen werden. Ab einer Laufzeit von mehr als 8 Jahren steigt der Spread leicht an. Der Spread für italienische Anleihen steigt hingegen beinahe linear mit der Laufzeit an.

Die Schätzung nach der Methode von Svensson liefert ein ähnliches Ergebnis (vgl. Abbildung 3.5) Allerdings ist die deutsche und österreichische Zinsstrukturkurve nur in wenigen Laufzeitbereichen deckungsgleich. Dementsprechend gestaltet sich auch der Spread. Die italienische Zinsstrukturkurve hat unter beiden Ansätzen die gleiche Gestalt.

Ein Vergleich der Fehlerstatistiken zeigt (Tabelle 3.2), dass speziell der RMSE und AABSE der Preisabweichungen bei der Methode nach Svensson geringer ist, bei den Yieldabweichungen ist der Unterschied zwichen den zwei Methoden minimal.

Die Abbildungen 3.6 und 3.7, bei denen die Zinsstruktur der Länder einzeln mit den jeweiligen Renditen geplottet ist, lassen auch keinen gravierenden Unterschied in der Kurvenanpassung erkennen.

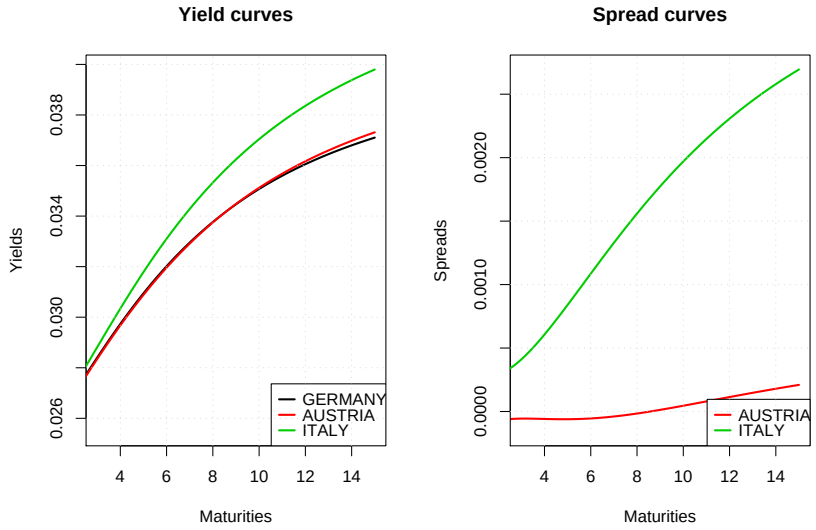

Abbildung 3.4: Zinsstrukturkurven und Spreads - Nelson/Siegel Methode

	Deutschland	Österreich	Italien
RMSE of bond prices - Nelson/Siegel	0.285	0.061	0.159
RMSE of bond prices - Svensson	0.091	0.049	0.085
AABSE of bond prices - Nelson/Siegel	0.163	0.042	0.104
AABSE of bond prices - Svensson	0.058	0.034	0.060
RMSE of yields - Nelson/Siegel	0.00048	0.00036	0.00071
RMSE of yields - Svensson	0.00044	0.00037	0.00069
AABSE of yields- Nelson/Siegel	0.00041	0.00026	0.00060
AABSE of yields- Svensson	0.00039	0.00026	0.00058

Tabelle 3.2: Fehlerstatistiken

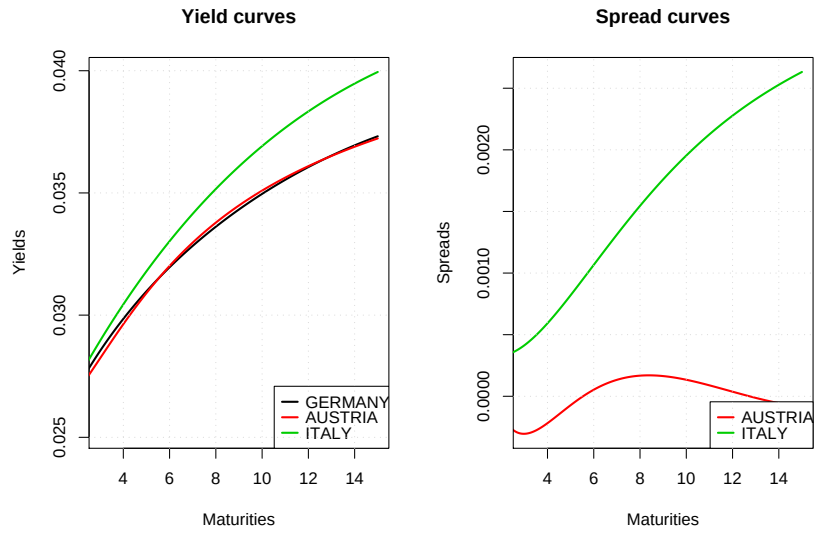

Abbildung 3.5: Zinsstrukturkurven und Spreads- Svensson Methode

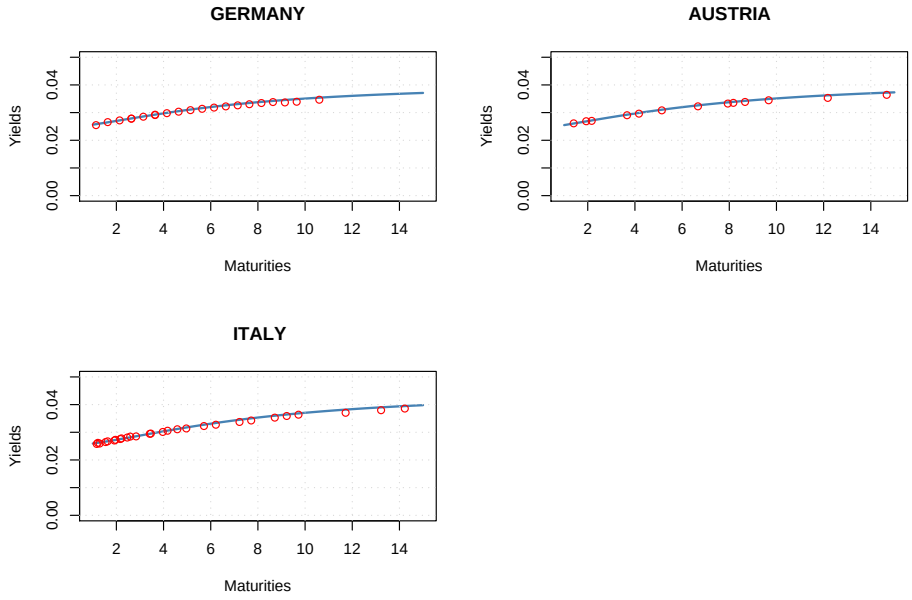

Abbildung 3.6: Zinsstrukturkurven - Nelson u. Siegel Methode

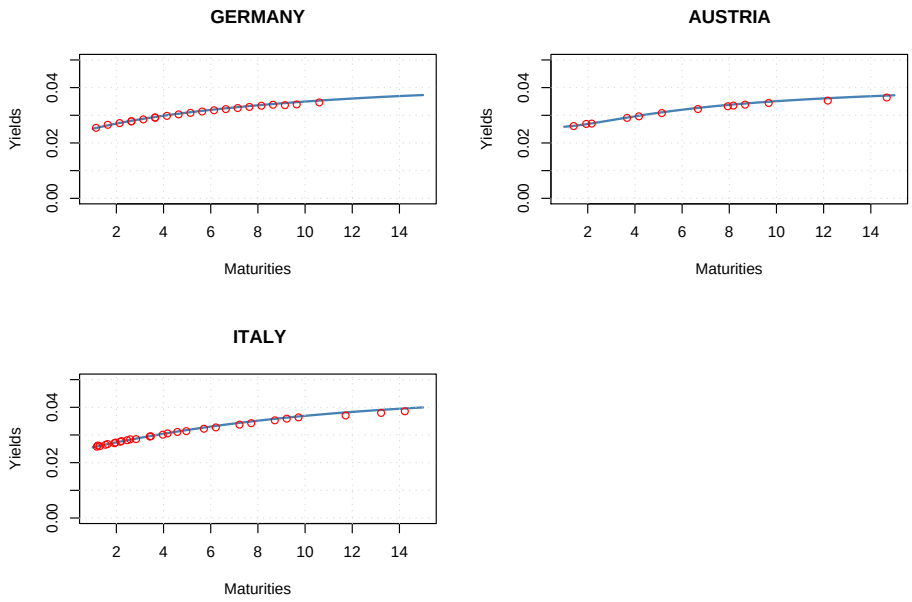

Abbildung 3.7: Zinsstrukturkurven - Svensson Methode

Schätzen der Zinsstruktur für die Ratingklassen AAA, A+ und BBB

Die Betrachtung der Abbildungen 3.8 und 3.9 liefert einen nicht zu verachtenden Unterschied zwichen den Schätzmethoden, vor allem wenn man die Spreads betrachtet. Bei der Methode nach Svensson ist der Spread zwischen BBB und A+ wesentlich konstanter. Interessanterweise kommt es zu einem Absinken der Spreads für die Ratingklasse A+ ab einer Lauzeit von 12 Jahren bei der Schätzung nach Nelson und Siegel.

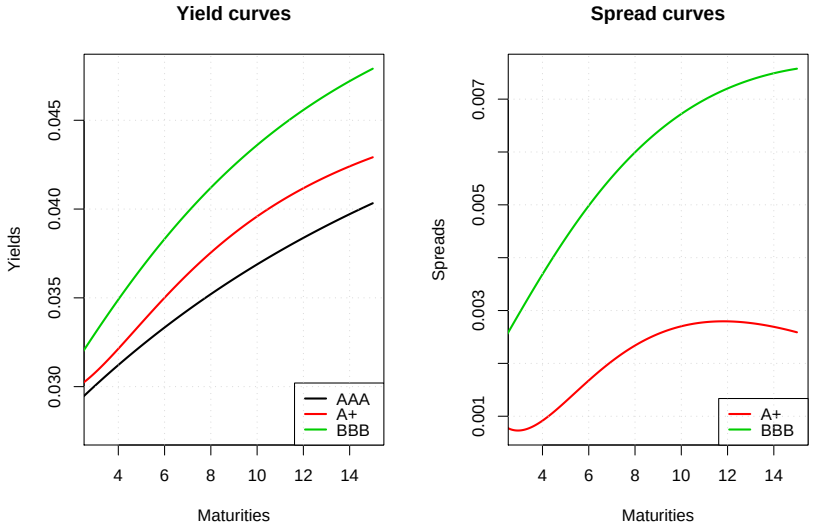

Abbildung 3.8: Zinsstrukturkurven - Nelson u. Siegel Methode

Die Fehlerstatistiken zeigen für beide Methoden in etwa das gleiche Bild (Tabelle 3.3). Nur der RSME und AABSE der Preisabweichung sind bei der Methode von Svensson etwas geringer. Im Vergleich zu dem zuvor präsentierten Beispiel sind die Fehlerstatistiken wesentlich größer und somit ist die Schätzung wesentlich schlechter. Ein Blick auf die Abbildungen 3.10 und 3.11 bestätigt die höheren Fehlerstatistiken. Die Datenlage, mit vielen Ausreißern, scheint die Ursache dafür zu sein.

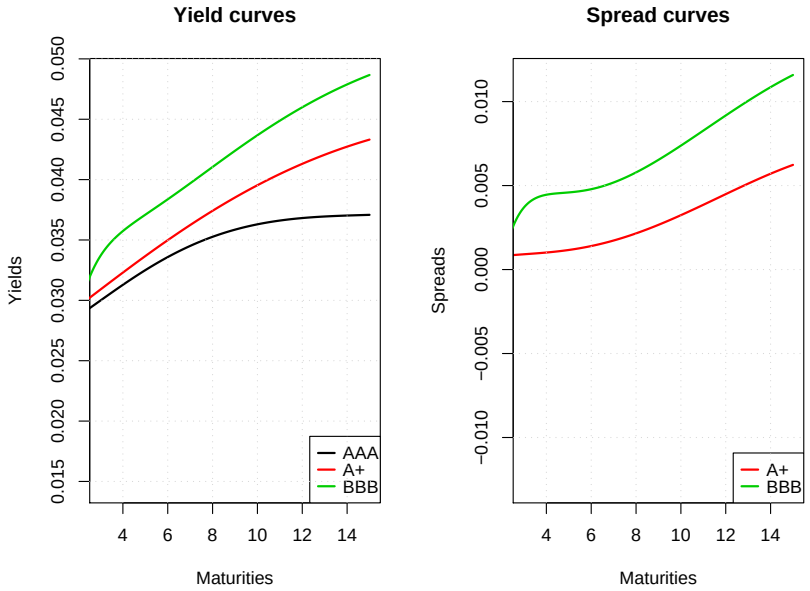

Abbildung 3.9: Zinsstrukturkurven- Svensson Methode

	AAA	A+	BBB
RMSE of bond prices - Nelson/Siegel	0.234	0.696	1.943
RMSE of bond prices - Svensson	0.210	0.662	1.799
AABSE of bond prices - Nelson/Siegel	0.180	0.542	1.062
AABSE of bond prices - Svensson	0.160	0.514	1.003
RMSE of yields - Nelson/Siegel	0.00088	0.00107	0.00227
RMSE of yields - Svensson	0.00136	0.00105	0.0021
AABSE of yields- Nelson/Siegel	0.00063	0.00088	0.00165
AABSE of yields- Svensson	0.00078	0.00088	0.00159

Tabelle 3.3: Fehlerstatistiken

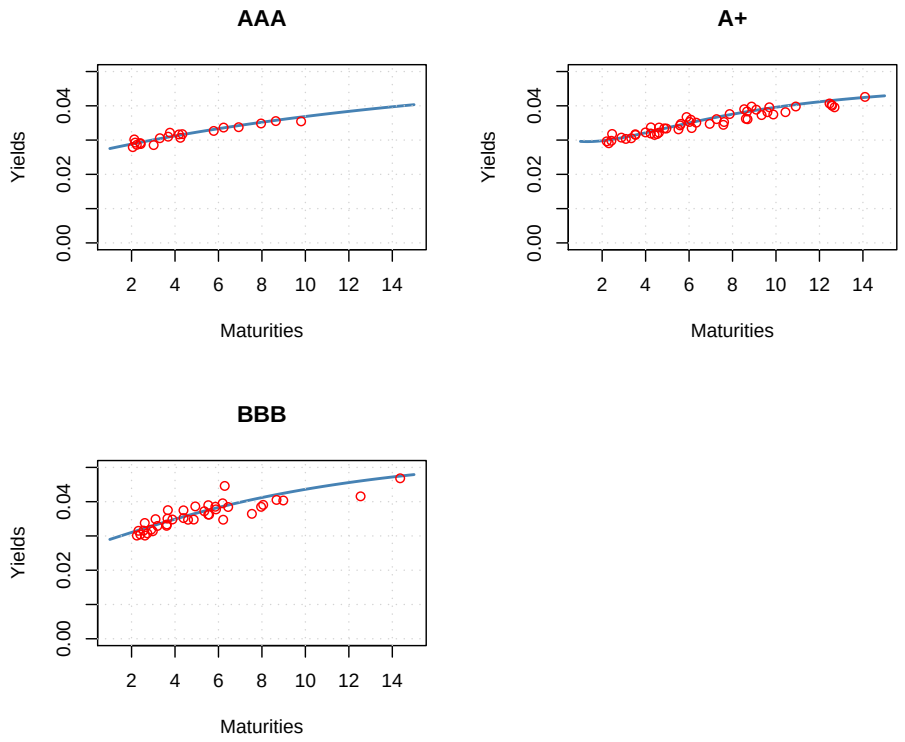

Abbildung 3.10: Zinsstrukturkurven - Nelson u. Siegel Methode

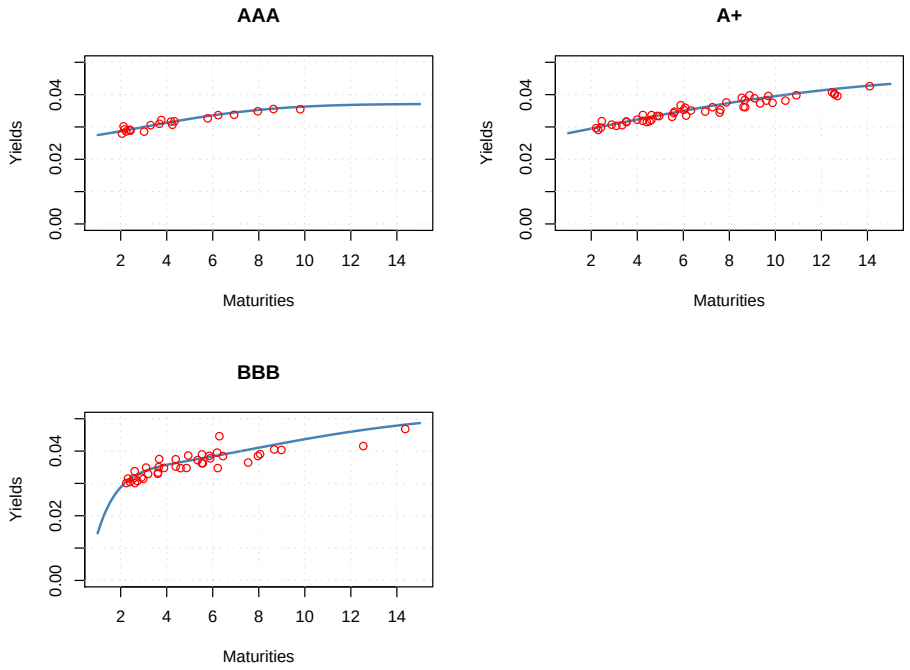

Abbildung 3.11: Zinsstrukturkurven - Svensson Methode

Kapitel 4

Optionsbewertung

Optionen wurden erstmalig auf der Terminbörse CBOE (Chicago Board Options Exchange) im Jahre 1973 standardisiert gehandelt und seither gab es einen massiven Zuwachs. Gehandelt werden Optionen mittlerweile auf verschiedenste Basiswerte (*underlying*) auf den verschiedensten Märkten. Ziel dieses Teils der Arbeit ist es wichtige Modelle zur Bewertung von Optionen auf Aktien sowohl von der theoretischen Seite vorzustellen, als auch die Anwendung unter `R` mit den relevanten Packages zu zeigen.

Zur Bewertung für die unterschiedlichsten Typen von Optionen stehen verschiedenste analytische und numerische Methoden zur Verfügung. Black und Scholes (1973) und Merton (1973) veröffentlichten jeweils im Jahre 1973 einen Artikel mit einem Modell zur exakten Bewertung von Optionen europäischen Typs. Merton und Scholes wurden für diese Entwicklung 1997 auch mit dem Nobelpreis für Wirtschaftswissenschaften ausgezeichnet, Black war zu diesem Zeitpunkt schon verstorben.

Die folgenden theoretischen Ausführungen und die anschließende Präsentation von Beispielen klammert die analytischen Modelle, wie das Black-Scholes-Merton Modell und andere analytische Approximationsmodelle aus. Vielmehr wird der Schwerpunkt auf numerische Methoden zur Optionsbewertung gelegt. In dem Punkt Allgemeine Grudlagen werden kurz die, für die weiteren Modelle und Beispiele relevanten Arten von Optionen behandelt. Nachfolgend werden die verwendeten Bewertungsmodelle und Ansätze vorgestellt. Als Methoden zur Optionsbewertung werden dabei Binomialbäume , Monte Carlo Simulation und finite Differenzen vorgestellt. Abgeschlossen wird dieses Kapitel mit einer Demonstration von Anwendungsbeispielen unter Verwendung der relevanten `R` Packages.

4.1 Allgemeine Grundlagen

An dieser Stelle werden kurz die wesentlichen Arten von Optionen dargestellt, wobei sich die Auswahl auf die später zu bewertenden Optionen beschränkt. Die Grundlagen basieren auf den entsprechenden Ausführungen in den Büchern von Hull (2000) und Wilmott (2001).

4.1.1 Europäische Optionen

Unter einer europäischen Option versteht man eine Option, bei der der Käufer das Recht hat einen bestimmten Basiswert (Underlying S) zu einem bestimmten Zeitpunkt (Ausübungszeitpunkt T) zu einem vorher bestimmten Preis (Ausübungspreis, Strikepreis X) zu kaufen (Call-Option) beziehungsweise zu verkaufen (Put-Option). Der Wert einer Call-Option (Put-Option) zum Zeipunkt t wird mit C_t (P_t) bezeichnet, der Wert des Underlyings mit S_t. Die Call-Option wird nur ausgeübt, wenn $S_T > X$. Der Wert zum Zeitpunkt T ergibt sich somit folgendermaßen:

$$C_T = (S_T - X)_+ = max\{S_T - X, 0\}. \tag{4.1}$$

Für die Put-Option gilt analoges in umgekehrter Folge, d.h. die Option wird ausgübt wenn $S_T < X$, der Wert ergibt sich mit:

$$P_T = (X - S_T)_+ = max\{X - S_T, 0\} \tag{4.2}$$

Die folgenden zwei Abbildungen zeigen die Auszahlungsfunktion (*payoff*) einer Call- und Put-Option.

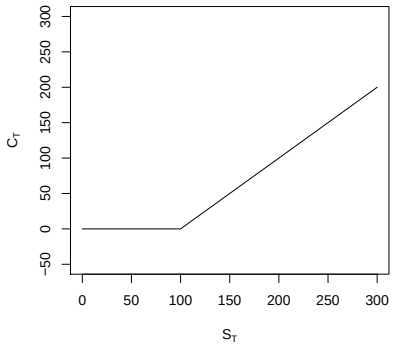

Abbildung 4.1: Payoff Call-Option

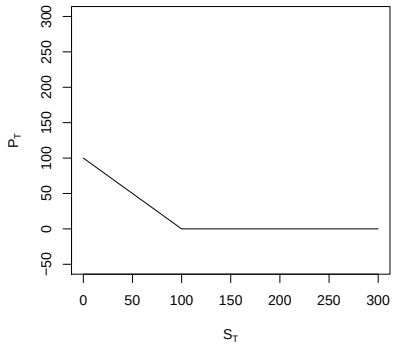

Abbildung 4.2: Payoff Put-Option

4.1.2 Amerikanische Optionen

Eine amerikanische Option ist der europäischen Option sehr ähnlich. Der Unterschied besteht im Ausübungszeitpunkt. Die Ausübung der amerikanischen Option ist zu jedem Zeitpunkt t innerhalb der Laufzeit $0 \dots T$ erlaubt. Wird die Option nicht ausgeübt verfällt sie. Der Wert zum Zeitpunkt t ergibt sich für eine amerikanische Call-Option mit:

$$C_t = (S_t - X)_+ = max\{S_t - X, 0\}, \tag{4.3}$$

und für eine amerikanische Put-Option mit:

$$P_t = (X - S_t)_+ = max\{X - S_t, 0\}. \tag{4.4}$$

4.1.3 Asiatische Optionen

Die asiatische Option kann zur Gruppe der exotischen Optionen subsummiert werden. Der Payoff einer arithmetischen asiatischen Option hängt vom Durchschnitt des Underlyings für eine bestimmte definierte Zeitspanne ab. Der Durchschnitt des Underlyings errechnet sich mit:

$$\bar{S}_T = \frac{1}{n} \sum_{i=1}^{n} S_i. \tag{4.5}$$

Die Auszahlungsfunktion einer Call-Option, Put-Option ergibt sich dann mit $(\bar{S}_T - X)_+$ beziehungsweise mit $(X - \bar{S}_T)_+$.

4.2 Verwendete Bewertungsansätze

4.2.1 Binomialbäume

Die Bewertung von Optionen mithilfe von Binomialbäumen geht zurück auf Cox, Ross und Rubinstein (October 1979). Das Modell geht von der Annahme aus, dass die Änderungen des Underlyings einem Binomialprozess folgen.

Anhand der Bewertung einer amerikanischen Option auf eine Aktie ohne Dividendenzahlungen soll das Modell vorgestellt werden. Die folgenden Ausführungen basieren auf den Büchern von Hull (2000) und Wilmott (2001). Die Laufzeit T der Option wird in eine große Anzahl von kleinen Zeitintervallen Δt eingeteilt. Der Kurs der Aktie kann in jedem dieser Zeitintervalle von seinem Ausgangswert S_0 entweder auf $S_0 u$ steigen oder auf $S_0 d$ fallen (wobei in der

Regel $u > 1$ und $d < 1$). Die Wahrscheinlichkeit mit der der Aktienkurs auf S_0u steigt, wird mit p bezeichnet, die Wahrscheinlichkeit eines Sinkens des Aktienkurses auf S_0d mit $(1-p)$. Abbildung 4.3 veranschaulicht diese Zusammenhänge grafisch für ein Zeitintervall Δt.

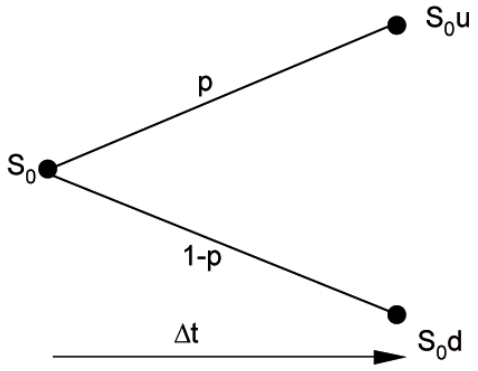

Abbildung 4.3: Eine Stufe im Binomialbaum

Unter der Verwendung der Prinzipien der risikoneutralen Bewertung kann die Option bewertet werden. Die Annahmen für die risikoneutrale Bewertung lauten:

- der erwartete Ertrag aller gehandelten Wertpapiere entspricht dem risikolosen Zinssatz r und
- alle zukünftigen Cashflows können durch Diskontieren der Erwartungswerte mit dem risikolosen Zinssatz bewertet werden.

Die Parameter des Binomialmodells p, u, d müssen nun während des Zeitintervalls Δt den Mittelwert und die Varianz der Änderung des Aktienkurses exakt wiedergeben. Durch die Annahmen der risikoneutralen Bewertung ergibt sich, dass der erwartete Aktienwert am Ende des Zeitintervalls Δt $Se^{r\Delta t}$ sein muss. Somit gilt:

$$Se^{r\Delta t} = pSu + (1-p)Sd, \tag{4.6}$$

$$e^{r\Delta t} = pu + (1-p)d. \tag{4.7}$$

Die Varianz der Preisänderung der Aktie in einem Zeitintervall ist gegeben mit $\sigma^2\Delta t$. Daher muss gelten:

$$pu^2 + (1-p)d^2 - (pu + (1-p)d)^2 = \sigma^2\Delta t. \tag{4.8}$$

Durch Substituieren von $p = \frac{e^{r\Delta t}-d}{u-d}$ in Gleichung 4.8 ergibt sich:

$$e^{r\Delta t}(u+d) - ud - e^{2r\Delta t} = \sigma^2 \Delta t. \tag{4.9}$$

Die Gleichungen 4.8 und 4.9 gemeinsam mit der von Cox, Ross und Rubinstein verwendeten Beziehung $u = \frac{1}{d}$ implizieren:

$$p = \frac{e^{r\Delta t} - d}{u - d}, \tag{4.10}$$

$$u = e^{\sigma\sqrt{\Delta t}}, \tag{4.11}$$

$$d = e^{-\sigma\sqrt{\Delta t}}. \tag{4.12}$$

In Abbildung 4.3 ist eine Stufe im Binomialbaum dargestellt. Diese Überlegungen sollen nun auf die gesamte Baumstruktur ausgedehnt werden. Zum Zeitpunkt 0 ist der Aktienpreis bekannt(S_0). Zum Zeitpunkt Δt gibt es zwei mögliche Aktienpreise Su oder Sd. Zum Zeitpunkt $2\Delta t$ gibt es 3 mögliche Aktienpreise Su^2,S,Sd^2 usw. . Allgemein müssen zu einem Zeitpunkt $i\Delta t$ $i+1$ Aktienpreise berücksichtigt werden:

$$S_0 u^j d^{i-j}, \qquad j = 0, ..., i. \tag{4.13}$$

Die Option kann nun bewertet werden indem, beginnend vom Zeitpunkt der Fälligkeit T bis zum Beginn, der Baum von hinten aufgelöst wird. Der Wert der Option zum Zeitpunkt T ist bekannt und ergibt sich für eine Call-Option mit $(S-X)_+$ beziehungsweise für eine Put-Option mit $(X-S)_+$. Der Wert einer Option zum Zeitpunkt $T - \Delta t$ ergibt sich basierend auf der Annahme der risikoneutralen Bewertung als der diskontierte Erwartungswert des Optionspreises zum Zeitpunkt T. Entsprechend kann der Wert der Option zum Zeitpunkt $T - \Delta t$ berechnet werden. Für eine amerikanische Option muss dabei beachtet werden, dass eine frühzeitige Ausübung bevorzugt werden kann.

Nun soll eine allgemein gültige Formel für den Wert einer Option in einem bestimmten Knoten des Baumes gefunden werden. Die folgenden Überlegungen gelten für amerikanische Put-Optionen. Die Laufzeit der Put-Option wird in N Intervalle mit einer Länge von Δt eingeteilt. Der j-te Knoten zum Zeitpunkt $i\Delta t$ wird in Folge als der $(i,j) - te$ Knoten bezeichnet ($0 \leq i \leq N, 0 \leq j \leq i$). Der Wert der Option bei einem Knoten (i,j) wird mit $f_{i.j}$, der Aktienkurs mit $S_0 u^j d^{i-j}$ bezeichnet. Da der Wert einer amerikanischen Put-Option zum Zeitpunkt T mit $(X - S_T)_+$ definiert ist, ergibt sich:

$$f_{N,j} = (X - S_0 u^j d^{N-j})_+ = max\{X - S_0 u^j d^{N-j}, 0\}, \qquad j = 0, \ldots, N. \tag{4.14}$$

Mit der Wahrscheinlichkeit von p wird ausgehend vom Knoten (i, j) zum Zeitpunkt $i\Delta t$ der Knoten $(i+1, j+1)$ zum Zeitpunkt $(i+1)\Delta t$ erreicht. Der Knoten $(i+1, j)$ wird mit Wahrscheinlichtkeit $(1-p)$ erreicht. Wird eine frühzeitige Ausübung der Option vernachlässigt ergibt sich der Wert der Option für einen Knoten (i, j) mit:

$$f_{i,j} = e^{-r\Delta t} \left(p f_{i+1,j+1} + (1-p) f_{i+1,j} \right), \tag{4.15}$$

wobei $0 \leq i \leq N-1$ und $0 \leq j \leq i$.

Bei der Möglichkeit der frühzeitigen Ausübung der Option muss der Optionswert zu einem bestimmten Zeitpunkt mit dem Wert der Auszahlungsfunktion zu diesem Zeitpunkt verglichen werden.

$$f_{i,j} = max\{X - Su^j d^{i-j}, e^{-r\Delta t} \left(p f_{i+1,j+1} + (1-p) f_{i+1,j} \right)\} \tag{4.16}$$

Geht N gegen unendlich, d.h Δt geht gegen 0, dann kann der exakte Wert der Option berechnet werden. Mit $N = 30$ werden bereits gute Ergebnisse erziehlt (vgl. Hull, 2000, S.392).

Unter Verwendung von relevanten `R` Packages werden mit Hilfe von Binomialbäumen amerikanische Optionen bewertet (siehe Abschnitt 4.3.2).

4.2.2 Finite Differenzen

Die folgenden Ausführungen stützen sich auf die Bücher von Hull und Willmott (vgl. Hull, 2000, S.415 ff) und (vgl. Wilmott, 2006, S. 1199 ff).

Methoden die auf finiten Differenzen basieren, sind numerische Verfahren zur Lösung von partiellen Differentialgleichungen. Die Methode ist mit dem Binomialbaumansatz vergleichbar, jedoch wesentlich allgemeiner und flexibler einsetzbar. Die Methode der finiten Differenzen kann zur Optionsbewertung eingesetzt werden, wenn es keine analytische Lösungsmöglichkeiten gibt. Die Bewertung erfolgt durch das Lösen der spezifischen Differentialgleichung der Option. Die Differentialgleichung wird dabei in eine große Anzahl von Differenzengleichungen aufgespalten. Diese werden schrittweise gelöst und zur Approximation der Differentialgleichung herangezogen.

Im Folgenden soll die Methode der finiten Differenzen näher erläutert werden, dabei wird speziell auf die Bewertung von amerikanischen Optionen eingegangen. Die Differentialgleichung die eine amerikanische Option erfüllt, hat dabei folgendes Aussehen:

$$\frac{\delta f}{\delta t} + rS\frac{\delta f}{\delta S} + \frac{1}{2}\sigma^2 S^2 \frac{\delta^2 f}{\delta S^2} = rf. \tag{4.17}$$

Wobei f der Wert der Option, S der Wert des Underlyings und r der risikolose Zinssatz ist.

Bei der Methode der finiten Differenzen wird die Baumstruktur durch eine Gitterstruktur ersetzt. Die Laufzeit T der Option wird in I Intervalle konstanter Länge ($\Delta t = \frac{T}{I}$) unterteilt und der Aktienkurs in J Intervalle konstanter Länge ($\Delta S = \frac{S_{max}}{J}$). Anzumerken bleibt, dass die Black Scholes Gleichung für $S \leq S < \infty$ gelöst wird und daher stellt S_{max} die Approximation für den unendlichen Aktienkurs dar. Praktisch gesehen, soll dieser Wert so gewählt werden, dass er drei bis viermal dem Ausübungspreis X entspricht (vgl. Wilmott, 2006, S. 1202). Das Gitter besteht aus $(I + 1) \times (J + 1)$ Punkten.

Abbildung 4.4 zeigt eine schematische Darstellung eines Gitters, das zur Berechnung der finiten Differenzen verwendet werden kann.

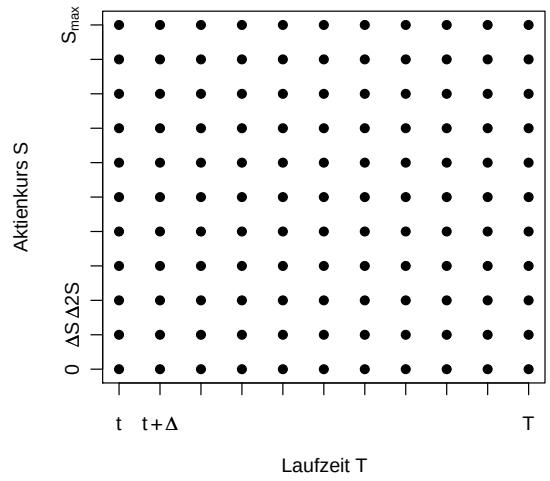

Abbildung 4.4: Gitter für den Ansatz der finiten Differenzen

Ein Punkt (i, j) im Gitter bezieht sich auf den Zeitpunkt $i\Delta t$ und den Aktienkurs $j\Delta S$. Der Wert der Option an diesem Punkt wird wie folgt bezeichnet:

$$f_{i,j} = f(i\Delta t, j\Delta S), \tag{4.18}$$

wobei $0 \leq i \leq I$ und $0 \leq j \leq J$.

Mittels der Methode der finiten Differenzen soll nun gezeigt werden, wie die einzelnen Terme der unter Gleichung 4.17 angeführten Differentialgleichung approximiert werden können.

Approximation von $\theta = \frac{\delta f}{\delta t}$:

Die Definition der ersten Ableitung vom Wert der Option nach der Zeit lautet:

$$\frac{\delta f}{\delta t} = \lim_{n \to \infty} \frac{f(t+h, S) - f(t, S}{h}. \tag{4.19}$$

Diese kann durch Werte im Gitter

$$\frac{\delta f}{\delta t}(t, S) \approx \frac{f_{i+1,j} - f_{i,j}}{\Delta t} \tag{4.20}$$

approximiert werden.

Der Fehler, der durch diese Approximation gemacht wird, kann mithilfe von Taylorreihen quantifiziert werden. Die Funktion des Werts der Option zum Zeitpunkt $t - \delta t$ mit einem Aktienkurs von S lässt sich durch eine Taylor Entwicklung an der Stelle (t, S) als

$$f(t - \delta t, S) = f(t, S) - \delta t \frac{\delta f}{\delta t}(S, t) + O(\delta t^2) \tag{4.21}$$

darstellen.

In Bezug auf das Gitter lässt sich obige Gleichung anschreiben als:

$$f_{i,j} = f_{i+1,j} - \delta t \frac{\delta f}{\delta t}(S, t) + O(\delta t^2), \tag{4.22}$$

$$\frac{\delta f}{\delta t}(S, t) = \frac{f_{i+1,j} - f_{ij}}{\delta t} + O(\delta t). \tag{4.23}$$

Der Fehler, der durch die Approximation von θ durch Gleichung 4.20 gemacht wird, beträgt damit $O(\delta t)$.

Approximation von $\Delta = \frac{\delta f}{\delta S}$:

Für die Approximaiton von Δ können die selben Überlegungen, wie zur Approximation von θ verwendet werden. Die folgenden drei Gleichungen stellen drei verschiedene Methoden der Approximation dar.

$$\frac{\delta f}{\delta S} \approx \frac{f_{i,j+1} - f_{i,j}}{\Delta S} \tag{4.24}$$

$$\frac{\delta f}{\delta S} \approx \frac{f_{i,j} - f_{i,j-1}}{\Delta S} \tag{4.25}$$

$$\frac{\delta f}{\delta S} \approx \frac{f_{i,j+1} - f_{i,j-1}}{2\Delta S} \tag{4.26}$$

Die Möglichkeiten werden als Vorwärtsdifferenz (Gleichung 4.24), Rückwertsdifferenz (Gleichung 4.25) und als mittlere Differenz (Gleichung 4.26) bezeichnet.

Es lässt sich zeigen, dass die mittlere Differenz (*central difference*) die Methode mit den kleineren Fehler O ist (vgl. Wilmott, 2006, S. 1204). Die Herleitung basiert wieder auf einer Taylor Entwicklung.

Approximation von $\Gamma = \frac{\delta^2 f}{\delta S^2}$:

Die Approximation von Γ erfolgt über die Differenz der Approximation von Δ nach der Methode der Vorwärts- und Rückwertsdifferenz. Die Approximation ergibt sich mit:

$$\frac{\delta^2 f}{\delta S^2}(S,t) \approx \frac{(\frac{f_{i,j+1}-f_{i,j}}{\Delta S} - \frac{f_{i,j}-f_{i,j-1}}{\Delta S})}{\Delta S} \approx \frac{f_{i,j+1} + f_{i,j-1} - 2f_{i,j}}{\Delta S^2}. \tag{4.27}$$

Randbedingungen

Eine numerische Lösung der unter Gleichung 4.17 angeführten Differentialgleichung setzt die Spezifikation von Grenzen voraus. Diese unterscheiden sich je nach Charakteristika der zugrunde liegenden Option. Die weiteren Ausführungen sind im speziellen auf eine amerikanische Put-Option maßgeschneidert.

Der Wert einer amerikanischen Put-Option zum Zeitpunkt T ist mit $(X - S_T)_+$ definiert. Daher gilt:

$$f_{I,j} = (X - j\Delta S)_+ = max\{X - j\Delta S, 0\} \qquad j = 0, 1, \ldots, J. \tag{4.28}$$

Der Wert der Put-Option, wenn der Aktienkurs 0 ist, ergibt sich mit X.

$$f_{i,0} = X \qquad i = 0, 1, \ldots, I \tag{4.29}$$

Unter, der Annahme, dass $S = S_{max}$ passend gewählt wurde, hat die Put-Option einen Wert von 0.

$$f_{i,J} = 0 \qquad i = 0, 1, \ldots, I \tag{4.30}$$

Die Gleichungen 4.28,4.29 und 4.30 definieren somit den Wert der Put-Option an drei Kanten des Gitters.

Die Differentialgleichung 4.17 kann nun durch verschiedene Methoden approximiert werden, zwei davon sollen nun vorgestellt werden.

Die implizite Methode

Durch Substituieren der Gleichungen 4.20, 4.26 und 4.27 in die Differentialgleichung 4.17 und unter Verwendung des Zusammenhangs $S = j\Delta S$ ergibt sich:

$$\frac{f_{i+1,j} - f_{i,j}}{\Delta t} + rj\Delta S \frac{f_{i,j+1} - f_{i,j-1}}{2\Delta S} + \frac{1}{2}\sigma^2 j^2 \Delta S^2 \frac{f_{i,j+1} + f_{i,j-1} - 2f_{i,j}}{\Delta S^2} = r f_{i,j}, \tag{4.31}$$

mit $j = 0, 1, \ldots, J-1$ und $i = 0, 1 \ldots, I-1$.

Durch Vereinfachen erhält man:

$$a_j f_{i,j-1} + b_j f_{i,j} + c_j f_{i,j+1} = f_{i+1,j}, \tag{4.32}$$

mit

$$a_j = \frac{1}{2} rj\Delta t - \frac{1}{2}\sigma^2 j^2 \Delta t, \tag{4.33}$$

$$b_j = 1 + \sigma^2 j^2 \delta t + r\Delta t, \tag{4.34}$$

$$c_j = -\frac{1}{2} rj\Delta t - \frac{1}{2}\sigma^2 j^2 \Delta t. \tag{4.35}$$

Die schematische Darstellung in Abbildung 4.5 erläutert die implizite Methode. Die verbundenen blauen Punkte sollen das Prinzip der impliziten Methode darstellen. Bei der impliziten Methode werden für die Berechnung von $f_{i+1,j}$ die Werte $f_{i,j-1}, f_{i,j}, f_{i,j+1}$ herangezogen (vgl. Gleichung 4.32).

Durch die drei Randbedingungen (4.28,4.29,4.30) ist an den Randpunkten des Gitters die Gleichung 4.32 lösbar. Die Berechnung der übrigen Werte der Option erfolgt schrittweise.

Schritt 1:

Die Gitterpunkte zum Zeitpunkt $T - \Delta t$ werden durch die folgende Gleichung berechnet:

$$a_j f_{I-1,j-1} + b_j f_{I-1,j} + c_j f_{I-1,j+1} = f_{I,j}, \tag{4.36}$$

mit $j = 1, \ldots, J-1$. Die rechte Seite der obigen Gleichung ist durch Gleichung 4.29 und 4.30 gegeben.

Schritt 2:

Lösen der Gleichung 4.36 mit $J-1$ Unbekannten.

Schritt 3:

Jeder Wert der Option $f_{I-1,j}$ wird mit $X - j\Delta S$ verglichen. Ist $f_{I-1,j} < X - j\Delta S$, dann ist eine frühzeitige Ausübung der Option zum Zeitpunkt $T - \Delta t$ von Vorteil und $f_{I-1,j}$ wird auf auf $X - j\Delta S$ gesetzt.

Schritt 4:

Alle anderen Knoten des Gitters für den Zeitpunkt $T - i\Delta t$ mit $0 \leq i \leq I-1$ werden in analoger Weise berechnet. Schlussendlich werden die Werte $f_{0,1}, f_{0,2}, f_{0,3}, \ldots, f_{0,J-1}$ berechnet, unter denen sich der gesuchte Wert der Option befindet.

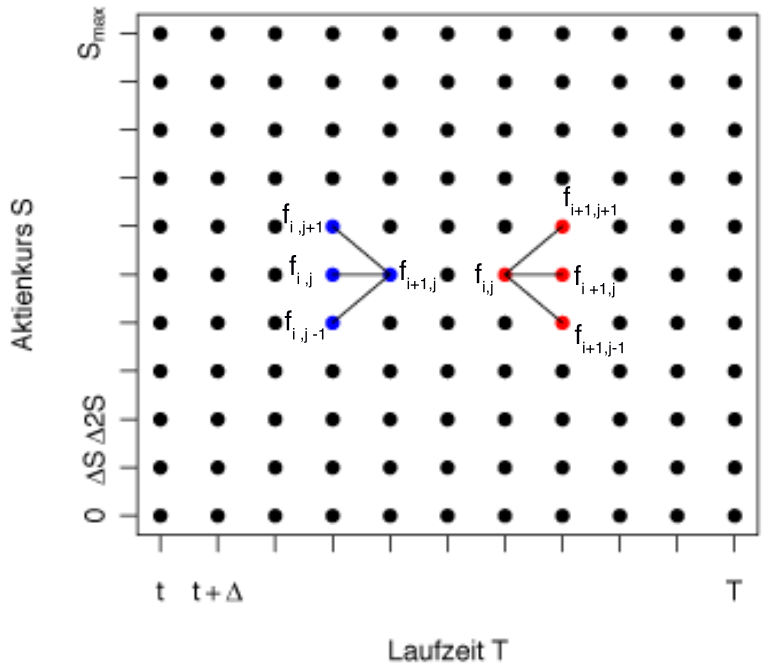

Abbildung 4.5: Implizite (blau) versus explizite (rot) finite Differenzen Methode

Die explizite Methode

Die implizite Methode hat den großen Vorteil, dass sie immer zur Lösung der zugrunde liegenden Differentialgleichung konvergiert, wenn ΔS und Δt gegen Null gehen (vgl. Hull, 2000, S. 419). Der Nachteil bei der impliziten Methode ist, dass jeweils $J-1$ Gleichungen gelöst werden müssen um den Wert $f_{i,j}$ zu berechnen. Bei der expliziten Methode geht man nun von der Überlegung aus, dass die Werte $\frac{\delta f}{\delta S}$ und $\frac{\delta^2 f}{\delta S^2}$ am Gitterpunkt (i,j) gleich den Werten am Gitterpunkt $(i+1,j)$ sind.

Die Gleichungen 4.26 und 4.27 ändern sich somit auf:

$$\frac{\delta f}{\delta S} \approx \frac{f_{i+1,j+1} - f_{i+1,j-1}}{2\Delta S}, \tag{4.37}$$

$$\frac{\delta^2 f}{\delta S^2}(S,t) \approx \frac{f_{i+1,j+1} + f_{i+1,j-1} - 2f_{i+1,j}}{\Delta S^2}. \tag{4.38}$$

Dies führt wiederum zu einer veränderten Gleichung zur Approximation der Differentialgleichung 4.17:

$$f_{i,j} = a_j^* f_{i+1,j-1} + b_j^* f_{i+1,j} + c_j^* f_{i+1,j+1}, \tag{4.39}$$

mit

$$a_j^* = \frac{1}{1+r\Delta t}\left(-\frac{1}{2}rj\Delta t + \frac{1}{2}\sigma^2 j^2 \Delta t\right), \tag{4.40}$$

$$b_j^* = \frac{1}{1+r\Delta t}\left(1 - \sigma^2 j^2 \Delta t\right), \tag{4.41}$$

$$c_j^* = \frac{1}{1+r\Delta t}\left(\frac{1}{2}rj\Delta t + \frac{1}{2}\sigma^2 j^2 \Delta t\right). \tag{4.42}$$

Die explizite Methode verwendet daher zur Berechnung des Wertes $f_{i,j}$ die Werte zum Zeitpunkt $((i+1)\Delta t)$ $f_{i+1,j-1}, f_{i+1,j}, f_{i+1,j+1}$. In Abbildung 4.5 ist dieser Zusammenhang grafisch dargestellt. Die roten verbundenen Gitterpunkte veranschaulichen die explizite Methode. Die Berechnung des Optionspreis erfolgt analog zur impliziten Methode.

Es gibt noch weitere Methoden, die für finite Differenzen verwendet werden, als Beispiel sei die Crank-Nicolson Methode erwähnt, die man sich als Durchschnitt der impliziten und expliziten Methode vorstellen kann. Auf eine detaillierte Darstellung dieser Methode wird aber verzichtet, für Details sei verwiesen auf (Wilmott, 2006, S.1129).

Ein Beispiel, bei dem eine Option unter Verwendung der finiten Differenzen Methode bewertet wird, ist unter Punkt 4.3.2 zu finden.

4.2.3 Monte Carlo und Quasi Monte Carlo Simulation

Einleitung

Die Ausführungen in diesem Abschnitt stützen sich auf Glassermann (2005). Die Monte Carlo Simulation basiert auf der Monte Carlo Integration. Grundlage der Monte Carlo Simulation ist das Gesetz der großen Zahlen: Sind $X_1, X_2, \ldots$ unabhängige, identisch verteilte Zufallsvariablen mit endlichem Erwartungswert μ, dann konvergiert die endliche Folge der arithmetischen Mittel $(\bar{X}_n)$ fast sicher gegen μ.

Sei $f(x)$ die Dichte von X mit dem Erwartungswert

$$\mathbb{E}[g(X)] = \int g(x)f(x)dx. \tag{4.43}$$

Besteht die Möglichkeit unabhängige Zufallsvariablen X_i zu erzeugen, mit derselben Verteilung wie X, dann kann der Erwartungswert durch

$$\mathbb{E}[g(X)] = \frac{g(X_1) + \cdots + g(X_n)}{n} \tag{4.44}$$

geschätzt werden.

Analoges kann auf Integrale angewendet werden, sei $\mu = \int g(x)dx$ und f eine Dichte mit $\{x|g(x) \neq 0\} \subseteq \{x|f(x) > 0\}$, dann gilt

$$\mathbb{E}\left[\frac{g(X)}{f(X)}\right] = \int g(x)dx = \int \frac{g(x)}{f(x)}f(x)dx. \tag{4.45}$$

Man generiert eine f-verteilte Reihe von Zufallsvariablen $X_1, X_2, \ldots$ mit einer Dichte $f(x)$ und schätzt $\int g(x)dx$ durch:

$$\hat{\mu}_n = \frac{1}{n}\left(\frac{g(X_1)}{f(X_1)} + \cdots + \frac{g(X_n)}{f(X_n)}\right). \tag{4.46}$$

Sind $X_1, X_2, \ldots$ unabhängig und identisch verteilt mit einem endlichen Erwartungswert μ und einer endlichen Varianz σ^2, dann lässt sich mit Hilfe des zentralen Grenzwertsatzes zeigen, dass

$$\frac{(\bar{X}_n - \mu)}{\sqrt{\frac{\sigma^2}{n}}} \sim N(0,1). \tag{4.47}$$

Die Güte der Schätzung nimmt damit mit steigendem n zu. $\frac{\sigma}{\sqrt{n}}$ wird dabei als Standardfehler bezeichnet. Soll der Standardfehler um die Hälfte reduziert werden, muss n um den Faktor vier erhöht werden.

Grundlegende Schritte zur Bewertung von Optionen

Zur Bewertung von Optionen wird wieder das Prinzip der risikoneutralen Bewertung herangezogen. In einer risikoneutralen Welt ergibt sich der faire Preis einer einfachen europäischen Option als der diskontierte Erwartungswert des Payoffs zum Ende der Laufzeit der Option. Für das Underlying wird dabei eine risikoneutrale Irrfahrt (*random walk*) vorausgesetzt (vgl. Wilmott, 2006, S.1263).

Als risikoneutrale Irrfahrt kann etwa die folgende angenommen werden:

$$dS = rSdt + \sigma SdW. \tag{4.48}$$

Wobei W eine Brownsche Bewegung und σ die zu S gehörige Volatilität ist. Der Optionswert kann als $e^{-r(T-t)}\mathbb{E}[\text{payoff}(S)]$ geschrieben werden. Wird der risikolose Zinssatz r als konstant angenommen, kann der Optionswert nach folgendem Schema berechnet werden (vgl. Wilmott, 2006, S.1263), (vgl. Hull, 2000, S.407):

1. Simuliere den Random Walk für das benötigte Zeitintervall. Dadurch erhält man einen Preispfad des Underlyings.
2. Berechne den Payoff vom simulierten Preispfad.
3. Wiederhole die Schritte 1 und 2 um eine große Stichprobe von berechneten Payoffs zu erhalten.
4. Berechne den Mittelwert aller simulierten Payoffs.
5. Der Barwert dieses Mittelwerts ist der gesuchte Wert der Option.

Details zum Preispfad des Underlyings

Das Standardmodell mit dem die stochastischen Prozesse von Aktienkursen beschrieben werden, ist die geometrische Brownsche Bewegung. Ein stochastischer Prozess (S_t) folgt einer

geomtrischen Brownschen Bewegung, wenn $log(S_t)$ einer standardisierten Brownschen Bewegung (auch Wiener Prozess genannt) mit dem Startwert $log(S_0)$ folgt (vgl. Glassermann, 2005, S.93). Im Gegensatz zur standardisierten Brownschen Bewegung kann die geometrische Brownsche Bewegung keine negativen Werte annehmen.

Unter einer standardisierten Brownschen Bewegung versteht man einen stochastischen Prozess $(W_t)_{t\in[0,T]}$ mit stetigen Pfaden, der folgende Eigenschaften erfüllt (vgl. Sandmann, 1999, S. 244):

- $W_0 = 0$
- die Zuwächse $(W_t - W_s)$ und $(W_v - W_u)$ sind paarweise stochastisch unabhängig verteilt , $\forall s < t \leq u < v$
- die Zuwächse $(W_u - W_t)$ für $u > t$ sind normalverteilt mit:

$$(W_u - W_t) \sim N\left(0, (u-t)\right). \tag{4.49}$$

Sei W eine geometrische Brownsche Bewegung und S kann durch folgende stochastische Differentialgleichung beschrieben werden (vgl. Glassermann, 2005, S.93):

$$\frac{dS_t}{S_t} = rdt + \sigma dW_t. \tag{4.50}$$

Mit Hilfe des Lemmas von Itô lässt sich diese stochastische Differentialgleichung lösen. Die folgenden Ausführungen stützen sich auf (Sandmann, 1999, S. 251). Ausgangspunkt ist dabei der stochastische Prozess $(X_t)_{t=t_0}^T$, der durch

$$dX_t = \left(\mu - \frac{1}{2}\sigma^2\right) dt + \sigma dW_t \tag{4.51}$$

gegeben ist.

Die Lösung der stochastischen Differentialgleichung 4.51 ergibt sich mit:

$$X_t - X_{t_0} = \int_{t_0}^{t} \left(\mu - \frac{1}{2}\sigma^2\right) ds + \int_0^t \sigma dW_s = \left(\mu - \frac{1}{2}\sigma^2\right)(t - t_0) + \sigma(W_t - W_{t_0}). \tag{4.52}$$

Für $S_t = e^{(X_t)} \equiv f(X_t)$ und der Anfangsbedingung $S_{t_0} = e^{X_{t_0}}$ gilt damit nach dem Itô Lemma,

$$dS_t = df = \frac{\delta f}{\delta t}dt + \frac{\delta f}{\delta x}dX_t + \frac{1}{2}\frac{\delta^2 f}{\delta x^2}(dXt)^2$$

$$= 0dt + e(X_t)dX_t + \frac{1}{2}e^{(X_t)}(dX_t)^2$$

$$= S_t\left(\left(\mu - \frac{1}{2}\sigma^2\right)dt + \sigma dW_t\right) + \frac{1}{2}S_t\sigma^2 dt$$

$$= S_t\mu dt + S_t\sigma dW_t. \tag{4.53}$$

Die Lösung der stochastischen Differentialgleichung 4.50 ergibt sich mit:

$$S_t = S_0 e^{\left((r-\frac{1}{2}\sigma^2)t+\sigma W_t\right)}. \tag{4.54}$$

Etwas allgemeiner formuliert, wenn $u < t$, gilt:

$$S_t = S_u e^{\left((r-\frac{1}{2}\sigma^2)(t-u)+\sigma(W_t-W_u)\right)}. \tag{4.55}$$

Da die Zuwächse von W unabhängig und normalverteilt sind (vgl. Gleichung 4.49) ergibt sich ein einfacher rekursiver Ansatz zur Berechnung des Aktienkurses S zum Zeitpunkt 0 $(t_0 < t_1 < t_2 \cdots < t_n)$:

$$S_{t_{i+1}} = S_{t_i} e^{\left((r-\frac{1}{2}\sigma^2)(t_{i+1}-t_i+\sigma\sqrt{t_{i+1}-t_i}Z_{t_{i+1}}\right)}. \tag{4.56}$$

Wobei $Z_1, Z_2, \ldots, Z_n$ unabhängige standardnormalverteilte Zufallszahlen sind. Wie Glassermann (2005) zeigt, ist diese Methode exakt und generiert keinen Diskretisierungfehler.

Varianzreduzierende Verfahren

Wie bereits erwähnt, beträgt der Standardfehler bei der Monte Carlo Simulation $\frac{\sigma}{\sqrt{n}}$, wobei n die Stichprobengröße der simulierten Zufallsvariablen ist. Soll der Fehler möglichst klein gehalten werden sind oft sehr große Stichprobenumfänge n notwendig. Varianzverringerende Verfahren liefern eine geringere Varianz der Schätzung bei gleichem Stichprobenumfang. Im folgenden soll das Verfahren der *antithetischen Variable* vorgestellt werden. Die Ausführungen basieren auf (Glassermann, 2005, S. 205 ff).

Die grundlegende Idee der antithetischen Variable ist die Reduktion der Varianz durch eine negative Abhängigkeit in den Replikationen der Zufallszahlen. Wird beispielsweise eine Sequenz

von unabhängig und identisch verteilten Zufallszahlen $Z_1, Z_2, \ldots$ erzeugt, wobei $Z \sim N(0,1)$, können die antithetischen Zufallszahlen durch $-Z_1, -Z_2, \ldots$ aus den ursprünglichen Zufallszahlen erzeugt werden. Werden die Zufallszahlen Z_i verwendet um die Zuwächse einer Brownschen Bewegung zu simulieren, dann werden mit $-Z_i$ die Zuwächse der Spiegelung des Pfades der ursprünglichen Brownschen Bewegung simuliert.

Um im Detail zu zeigen, dass die Verwendung einer antithetischen Variable zu geringerer Varianz führt, soll der Erwartungswert $\mathbb{E}[Y]$ mit Hilfe der Paare $(Y_1, \tilde{Y}_1), (Y_2, \tilde{Y}_2), \ldots, (Y_n, \tilde{Y}_n)$ approximiert werden. Wobei gilt, dass

- die Paare $(Y_1, \tilde{Y}_1), (Y_2, \tilde{Y}_2), \ldots, (Y_n, \tilde{Y}_n)$ unabhängig und identisch verteilt sind $(i.i.d)$ und
- $\forall\ i$, haben Y_i und $\tilde{Y}_i$ die selbe Verteilung, wobei für gewöhnlich keine Unabhängigkeit gilt.

Als Schätzwert für den Erwartungswert wird das arithmetische Mittel

$$\tilde{Y}_{AV} = \frac{1}{2n}\left(\sum_{i=1}^{n} Y_i + \sum_{i=1}^{n} \tilde{Y}_i\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i + \tilde{Y}_i}{2}\right) \tag{4.57}$$

herangezogen.

Mit Hilfe des zentralen Grenzwertsatzes lässt sich zeigen, dass

$$\frac{\tilde{Y}_{AV} - \mathbb{E}[Y]}{\sigma_{AV}/\sqrt{n}} \Rightarrow N(0,1), \tag{4.58}$$

mit

$$\sigma^2_{AV} = \mathbb{V}\left[\frac{Y_i + \tilde{Y}_i}{2}\right]. \tag{4.59}$$

Unter der Annahme, dass zur Generierung von einem Paar $(Y_i, \tilde{Y}_i)$ in etwa zweimal so viel Rechenaufwand notwendig ist wie zur Generierung von Y_i, dann ist die Verwendung der antithetischen Variable gegenüber der gewöhnlichen Monte Carlo Simulation vorzuziehen, wenn

$$\mathbb{V}[\tilde{Y}_{AV}] < \mathbb{V}\left[\frac{1}{2n}\sum_{i=1}^{n} Y_i\right],$$

das heißt

$$\mathbb{V}[Y_i + \tilde{Y}_i] < 2\mathbb{V}[Y_i]. \tag{4.60}$$

Wobei die linke Seite der Ungleichung 4.60, unter der Verwendung der Tatsache, dass Y_i und $\tilde{Y}_i$ identisch verteilt sind, auch geschrieben werden kann als

$$\mathbb{V}[Y_i\tilde{Y}_i] = \mathbb{V}[Y_i] + \mathbb{V}[\tilde{Y}_i] + 2\mathbb{COV}[Y_i, \tilde{Y}_i] = 2\mathbb{V}[Y_i] + 2\mathbb{COV}[Y_i, \tilde{Y}_i].$$

Somit ergibt sich die Bedingung, die erfüllt werden muss, damit die antithetische Variable zu einer Varianzreduktion führt, als

$$\mathbb{COV}[Y_i, \tilde{Y}_i] < 0. \tag{4.61}$$

Daher muss zwischen Z und $-Z$ eine negativer Zusammenhang, in Form einer negativen Kovarianz, bestehen. Eine hinreichende Bedingung ist die Monotonie der Abbildung der Inputs auf die Outputs durch den Simulationsalgorithmus. Glassermann (2005) zeigt, wenn der Output Y eine lineare Funktion der Inputs $(Z_1, \ldots, Z_n)$ ist, dann führt das Verfahren zu einer Schätzung mit einer Varianz von 0. Allerdings wäre in diesem Fall eine Schätzung nicht notwendig. Es zeigt sich aber, dass bei einer approximativ linearen Abbildung, antithetische Variablen sehr effektiv sein können.

Quasi Monte Carlo Methoden

An dieser Stelle soll kurz die quasi Monte Carlo Methode (*low-discrepancy methods*) vorgestellt werden. Auf eine detaillierte Darstellung wird allerdings verzichtet, vielmehr soll nur die Grundidee erläutert werden, da in den Beispielen unter Punkt 4.4.2 solche Methoden verwendet werden. Für detailliertere Ausführungen sei auf (Glassermann, 2005, S. 280 ff) verwiesen.

Im Gegensatz zur Monte Carlo Simulation, die auf Zufallszahlen basiert, bauen quasi Monte Carlo Methoden auf *low discrepancy* Reihen auf. Darunter versteht man Reihen von quasi Zufallszahlen, die aber keiner Zufälligkeit folgen. Vielmehr versuchen quasi Monte Carlo Methoden durch das gezielte Generieren von beinahe identisch verteilten Zufallsvariablen die Genauigkeit der Schätzung zu erhöhen. Bekannte Methoden zur Generierung von solchen Reihen von quasi Zufallszahlen wurden von Halton (1960) beziehungsweise Halton und Smith (1964) und Sobol‘ (1967) entwickelt.

Die Abbildung 4.6 veranschaulicht den Unterschied zwischen normalverteilten und quasi Zufallszahlen (Sobol) grafisch.

´

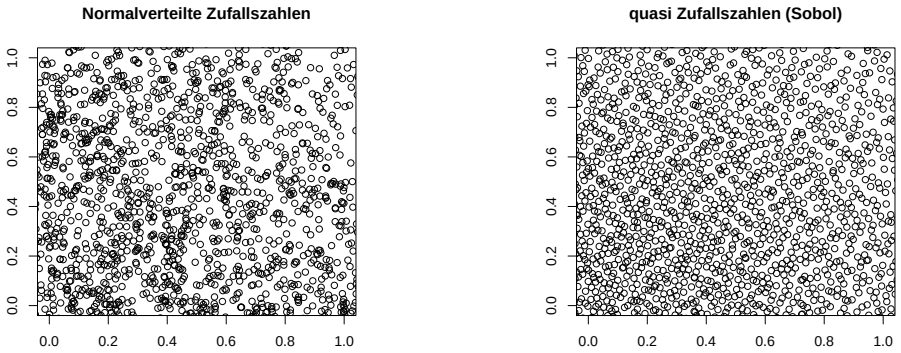

Abbildung 4.6: Zufallszahlen

Sprung - Diffusions Modelle (*jump diffusion models*)

Die Ausführungen zu diesem Thema stützen sich auf die relevanten Kapitel in (Glassermann, 2005, S. 134 ff).

Ein Großteil der verwendeten Modelle zur Preisberechnung von Derivaten geht von der Annahme aus, dass das Underlying stetigen Preispfaden folgt. Viele Studien haben Hinweise auf Preissprünge ergeben. Typischerweise haben Preisänderungsprozesse mit Sprüngen eine leptokurtische Verteilung. Im folgenden Abschnitt soll ein Modell vorgestellt werden, das solche Sprünge bei der Bewertung berücksichtigt.

Eines der ersten *jump diffusion* Modelle wurde von Merton für die Bewertung von Optionen auf Aktien entwickelt. Wobei er die Sprünge als idiosynkratische Schocks interpretierte, die nur ein Unternehmen aber nicht den gesamten Markt beeinträchtigen (vgl. Merton, 1976).

Das von Merton vorgestellte jump diffusion Modell kann durch folgende stochastische Differentialgleichung beschrieben werden:

$$\frac{dS_t}{S_{t-}} = \mu dt + \sigma W_t + dJ_t. \tag{4.62}$$

Wobei μ und σ Konstanten sind, W eine standardisierte Brownsche Bewegung ist und J ein von W unabhängiger Prozess mit stückweisen konstanten Sample-Pfaden. J ist gegeben durch:

$$J_t = \sum_{j=1}^{N_t} (Y_j - 1), \tag{4.63}$$

mit den Zufallszahlen $Y_1, Y_2, \ldots$ und dem Zählprozess N_t.

$$N_t = sup\{n : \tau_n \leq t\}$$

zählt die Anzahl der zufälligen Ankünfte $(0 < \tau_1 < \tau_2 < \dots)$ innerhalb des Intervalls $[0, t]$.

dJ_t aus Gleichung 4.64 bezeichnet einen Sprung des Prozesses J zum Zeitpunkt t. Die Größe des Sprungs ergibt sich mit:

$$\begin{cases} Y_j & \text{wenn } t = \tau_j \\ 0 & \text{wenn } t \neq \tau_j \qquad \forall j \end{cases}$$

Aufgrund des nicht stetigen Prozesses J muss S_t genauer spezifiziert werden, d.h. ob S_t den Wert vor beziehungsweise nach einem Sprung definiert. Geht man von einer rechtsseitigen Stetigkeit von S_t aus, werden alle Sprünge zum Zeitpunkt t berücksichtigt.

$$S_t = \lim_{u \downarrow t} S_u$$

Der linksseitige Limes beinhaltet die Werte gerade vor einem Sprung:

$$S_{t-} = \lim_{u \uparrow t} S_u.$$

Die stochastische Differentialgleichung 4.64 kann somit geschrieben werden als

$$dS_t = \mu S_{t-} dt + \sigma S_{t-} W_t + S_{t-} dJ_t. \tag{4.64}$$

dS_t hängt daher vom Wert S zum Zeitpunkt vor einem Sprung ab. Der Sprung in S zum Zeitpunkt t ist $S_t - S_{t-}$. Er ist 0, ausgenommen der Prozess J springt zum Zeitpunkt $t = \tau_j$ für einige j. Der Sprung zum Zeitpunkt τ_j kann somit formuliert werden als

$$S_{\tau_j} - S_{\tau_j -} = S_{\tau_j -}(J_{\tau_j} - J_{\tau_j -}) = S_{\tau_j -}(Y_j - 1) = S_{\tau_j} Y_j.$$

Y_j ist dabei das Verhältnis des Aktienkurses vor und nach einem Sprung, wobei die Sprünge multiplikativ sind. Wird Y_j auf positive Zufallszahlen beschränkt, so kann S_t nicht negativ werden, daher gilt:

$$logS_{\tau_j} = logS_{\tau_j} + logY_j,$$

und die Sprünge sind additiv. In weiterer Folge wird die geometrische Brownsche Bewegung als Prozess für die Preisänderungen und als logische Fortführung mit muliplikativen Sprüngen verwendet.

Die Lösung der Gleichung 4.64 ergibt sich mit:

$$S_t = S_0 e^{\left((\mu - \frac{1}{2}\sigma^2)t + \sigma W_t\right)} \prod_{j=1}^{N_t} Y_j. \tag{4.65}$$

Es wird angenommen, dass N_t einem Poissonprozess mit einer Rate von λ folgt und die Y_j selbst unabhängig, identisch verteilt und auch unabhängig von N sind. Die Zwischenankunftszeiten $\tau_{j+1} - \tau_j$ sind dann unabhängig und identisch verteilt mit

$$P(\tau_{j+1} - \tau_j) = 1 - e^{\lambda t},\ t \geq 0.$$

Auch wird für das Modell angenommen, dass Y_j lognormalverteilt ist ($Y_j \sim LN(a, b^2)$). Für das Produkt der Y_j ergibt sich:

$$\prod_{j=1} nY_j \sim LN(an, b^2 n).$$

Unter der Annahme, dass Y_j und W voneinander unabhängig sind und $N_t = n$, ergibt sich für S_t folgende Verteilung $F_{n,t}$:

$$S_t \sim LN\left(logS_0 + (\mu - \frac{1}{2}\sigma^2)t + an, \sigma^2 t + b^2 n\right).$$

Die unbedingte Verteilung von S_t unter der Verwendung, dass N_t poissonverteilt ist, lässt sich schreiben als :

$$P(S_t \leq x) = \sum_{n=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} F_{n,t}(x). \tag{4.66}$$

Merton (1976) drückte den Preis einer Option basierend auf Gleichung 4.66 als endliche Reihe aus, wobei jeder Term die Summe aus einer Poissonwahrscheinlichkeit und der Black-Scholes Formel ist.

In einer risikoneutralen Welt und ohne die Existenz von Sprüngen wird der Driftparameter μ in Formel 4.64 durch den risikolosen Zinssatz r ersetzt. Wird dieser Zinssatz als konstant angenommen, kann der Driftparameter über die Bedingung, dass $S_t e^{-rt}$ ein Martingal ist, hergeleitet werden. Merton (1976) dehnte diese Überlegungen auf sein jump diffusion Modell aus. Die wesentlichen Überlegungen zur Bestimmung des Driftparameters sollen an dieser Stelle kurz erläutert werden. Der Poissonprozess hat als allgemeine Eigenschaft, dass $N_t - \lambda t$ ein Martingal ist. Etwas allgemeiner formuliert ist

$$\sum_{i=1}^{N_t} h(Y_j) - \lambda \mathbb{E}[h(Y)]t$$

ein Martingal für unabhängig und identisch verteilte $Y, Y_1, Y_2, \ldots$ und jede Funktion h, für die $\mathbb{E}[h(Y)]$ endlich ist.

Der Prozess $J_t - \lambda mt$ ist ein Martingal, wenn $m = \mathbb{E}[Y_j - 1]$. Der Driftparameter μ muss somit gleich $r - \lambda m$ sein.

Simulation zu fixen Zeitpunkten

Die Simulation des Modells erfolgt zu fixen Zeitpunkten $t_0 = 0 < t_1 < t_2 < \cdots < t_n$. Um S_t zu den Zeitpunkten $t_1, \ldots, t_n$ simulieren zu können, wird Gleichung 4.65 zu

$$S_{t_{i+1}} = S_{t_i} e^{\left((\mu - \frac{1}{2}\sigma^2)(t_{i+1} - t_i) + \sigma(W_{t_{i+1}} - W_{t_i})\right)} \prod_{j=N_{t_{i+1}}}^{N_{t_{i+1}}} Y_j$$

umformuliert.

Für die Simulation kann auch folgende Formulierung verwendet werden:

$$X_{t_{i+1}} = X_{t_i} + \left(\mu - \frac{1}{2}\sigma^2\right)(t_{i+1} - t_i) + \sigma(W_{t_{i+1}} - W_{t_i}) \sum_{j=N_{t_{i+1}}}^{N_{t_{i+1}}} logY_j. \tag{4.67}$$

Die folgenden Schritte sind notwendig um Gleichung 4.67 für t_{i+1} ausgehend von t_i zu simulieren:

1. Generiere $Z_x \sim N(0, 1)$
2. Generiere $N \sim \text{Poisson}(\lambda(t_{i+1} - t_i))$; wenn $N = 0$ setze $M = 0$ und gehe zu Schritt 4.
3. Generiere $logY_1, \ldots, logY_N$ basierend auf der jeweiligen Verteilungsfunktion und setze $M = logY_1 + \cdots + logY_n$
4. Setze:
$$X_{t_{i+1}} = X_{t_i} + \left(\mu - \frac{1}{2}\sigma^2\right)(t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i}Z + M$$
5. Berechne $S_{t_i} = e^{X_{t_i}}$

Diese Methode basiert auf der Eigenschaft, dass die Zuwächse $N_{t_{i+1}} - N_{t_i}$ poissonverteilt mit Mittelwert $\lambda(t_{i+1} - t_i)$ und die Zuwächse unabhängig voneinander sind.

Unter der Annahme, dass $Y_j \sim LN(a, b^2)$ ($logY_j \sim N(a, b^2)$) kann die vorgestellte Methode noch vereinfacht werden. Da

$$\sum_{j=1} nlogY_j \sim N(an, b^2) = an + b\sqrt{n}N(0, 1),$$

kann der dritte Schritt durch

Generiere $Z_y \sim N(0, 1)$; setzte $M = aN + b\sqrt{N}Z_y$

ersetzt werden.

Im Abschnitt 4.4.2 wird eine Option basierend auf einem Sprung Diffusions Modell mittels Monte Carlo Simulation bewertet.

4.3 Bewertung von Optionen mit Binomialbäumen und finiten Differenzen

In diesem Abschnitt sollen Beispiele die theoretischen Ausführungen zur Optionsbewertung ergänzen. Um die praktische Bedeutung der Methoden zu veranschaulichen wird versucht eine gehandelte Option mit der Methode der Binomialbäume und finiten Differenzen zu bewerten. Dabei wird zuvor aus Marktdaten des Underlyings die Volatilität und aus Daten von US-amerikanischen Staatsanleihen der risikolose Zinssatz geschätzt.

Die Eigenschaften der zu bewertenden Option sind in Tabelle 4.1[1] aufgelistet.

Bezeichnung	*QAADQ.X*
Art	*amerikanische Call-Option*
Underlying	*Apple Computers Inc. - AAPL*
Underlying Preis	*87.72*
quotierter Preis	*10.30 USD*
Strike-Preis	*85 USD*
Ende der Laufzeit	*2007-04-20*
Abfragedatum	*2007-12-17*

Tabelle 4.1: Optionscharakteristika

[1] Quelle: finance.yahoo.com ,17.12.2006

4.3.1 Schätzen des risikolosen Zinssatzes und der Volatilität

Für die Bewertung der Option sind der risikolose Zinssatz und die Volatilität erforderlich. Die Volatilität wird aus Marktdaten der Aktie von Apple Computers Inc. geschätzt. Dazu wurden die jährlichen Aktienkurse von der Website *finance.yahoo.com* in R geladen. Verwendet wurden dabei die Funktionen des Rmetrics-Packages `fBasics` und `fCalendar`.

```
#load fBasics and fSeries library
library(fBasics)
library(fCalendar)

#Import of the stock quotes using the Rmetrics function yahooImport
query="s=AAPL&a=11&b=15&c=2005&d=11&q=15&f=2006&z=AAPL&x=.csv"

#set import function
aapl_import<-yahooImport(query, file = "tempfile",
    source = "http://chart.yahoo.com/table.csv?", save = FALSE,
    sep = ";", swap = 20, try = TRUE)

#print stock data
 head(aapl_import@data)
```

```
        DATE  Open  High   Low Close   Volume Adj..Close.
1 2005-12-15 72.68 72.86 71.35 72.18 20041500       72.18
2 2005-12-16 72.14  72.3 71.06 71.11 23970400       71.11
3 2005-12-19 71.11  72.6 71.04 71.38 18903400       71.38
4 2005-12-20 71.63 72.38 71.12 72.11 17111000       72.11
5 2005-12-21  72.6 73.61 72.54  73.5 16990600        73.5
6 2005-12-22 73.91 74.49  73.6 74.02 13236100       74.02
```

Nach der Umwandlung in ein Zeitreihenobjekt können die täglichen Renditen berechnet werden.

```
#alter aapl_import into a timeSeries object
aapl_ts<- timeSeries(as.numeric(as.vector(aapl_import@data[,7])),
charvec=as.character(aapl_import@data[,1])
format = "%Y-%m-%d",zone = "GMT", FinCenter = "GMT",units="AAPL")

#calculate daily log returns
aapl_returns_ts <- returnSeries(aapl_ts)

#print daily returns
head(aapl_returns_ts)
```

```
               AAPL
2005-12-16 -0.0149
2005-12-19  0.0038
2005-12-20  0.0102
2005-12-21  0.0191
2005-12-22  0.0070
2005-12-23 -0.0091
```

Basierend auf den täglichen Returns können nun die Momente berechnet werden.

```
#mean
colAvgs(aapl_returns_ts)
0.0005054475

#standard deviation
sd(aapl_returns_ts)
0.02370607

#skewness
colSkewness(aapl_returns_ts)
0.6944513

#Kurtosis
colKurtosis(aapl_returns_ts)
2.101957
```

Die berechnete Standardabweichung der täglichen Renditen vom Zeitraum 15.12.2005 bis 15.12.2006 kann als Schätzwert für die Volatilität verwendet werden. Wobei bemerkt werden muss, dass es sich um die tägliche Volatilität handelt.

Als risikoloser Zinssatz für die Optionsbewertung wird die Rendite einer Nullkuponanleihe, deren Laufzeit der Option entspricht, verwendet (vgl. Hull, 2000, S.389). Da am Markt kaum Nullkuponanleihen mit der passenden Laufzeit quotieren, müssen Daten anderer Anleihen zur Schätzung herangezogen werden. Dies führt uns wieder zu Kapitel 3, wo solche Fragestellungen ausführlich behandelt wurden. Die Ermittlung des risikolosen Zinssatzes erfolgt dabei mit dem Package `termstrc`. Als Input dienen die Daten der US -Treasury TNote Bonds mit halbjährigen Kuponzahlungen, die in der Tabelle 4.3.1 dargestellt sind. Die Namen der Bonds wurden dabei willkürlich gewählt und entsprechen keiner Marktkonvention.

Name	Preis	Kupon	Ende der Laufzeit	Yield-to-maturity in %	Emissionsdatum
TNOTE01	100.05	3.375	2007-01-15	2.64	1997-07-15
TNOTE02	100.28	3.125	2007-01-31	0.675	2005-07-31
TNOTE03	100.68	6.25	2007-02-15	1.857	1997-08-15
TNOTE04	100.07	2.25	2007-02-15	1.816	2004-08-15
TNOTE05	110.18	3.375	2007-02-28	2.407	2005-08-31
TNOTE06	100.13	3.75	2007-03-31	3.244	2005-11-30
TNOTE07	100	3.625	2007-04-30	3.596	2005-10-31
TNOTE08	101.14	6.625	2007-05-15	3.758	1997-11-15
TNOTE09	100.25	4.375	2007-05-15	3.744	2002-11-15
TNOTE10	99.73	3.125	2007-05-15	3.796	2004-11-15
TNOTE11	99.84	3.5	2007-05-31	3.856	2005-11-30
TNOTE12	99.77	3.625	2007-06-30	4.058	2005-12-31
TNOTE13	99.83	3.875	2007-07-31	4.154	2006-01-31
TNOTE14	101.19	6.125	2007-08-15	4.252	1998-01-15
TNOTE15	99.34	3.25	2007-08-15	4.266	2003-02-15
TNOTE16	99.03	2.75	2007-08-15	4.261	2005-02-15

Tabelle 4.2: Anleihencharakteristika

Basierend auf den Daten in Tabelle 4.1 beträgt die Laufzeit der Option 125 Tage. Die Yield-to-Maturity für diese Laufzeit kann aus der Grafik 4.7 abgelesen werden beziehungsweise mit Funktionen aus dem Package `termstrc` exakt für diese Laufzeit berechnet werden. Der risikolose Zinssatz für eine Laufzeit von 125 Tagen ergibt sich mit 0.03493302.

4.3.2 Bewertung von amerikanischen Optionen mit den Packages fOptions und RQuantLib

Die in Tabelle 4.1 vorgestellte Option soll nun bewertet werden. Als Volatilität und risikoloser Zinssatz werden die ermittelten Werte aus dem Abschnitt 4.3.1 verwendet. Für die Bewertungsmethode der Binomialbäume werden Funktionen aus dem Package `fOptions` verwendet. Das Package `RQuantLib` bietet die Möglichkeit den Optionspreis mittels finiten Differenzen zu berechnen.

Die folgenden Funktionsaufrufe zeigen die Vorgehensweise in `R`. Zu Beginn werden die Parameter spezifiziert. Die Angabe der Laufzeit, des risikolosen Zinssatzes und der Volatilität erfolgen dabei auf jährlicher Basis, wobei von 252 Handelstagen pro Jahr ausgegangen wird.

```
#load package fOptions
library(fOptions)
```

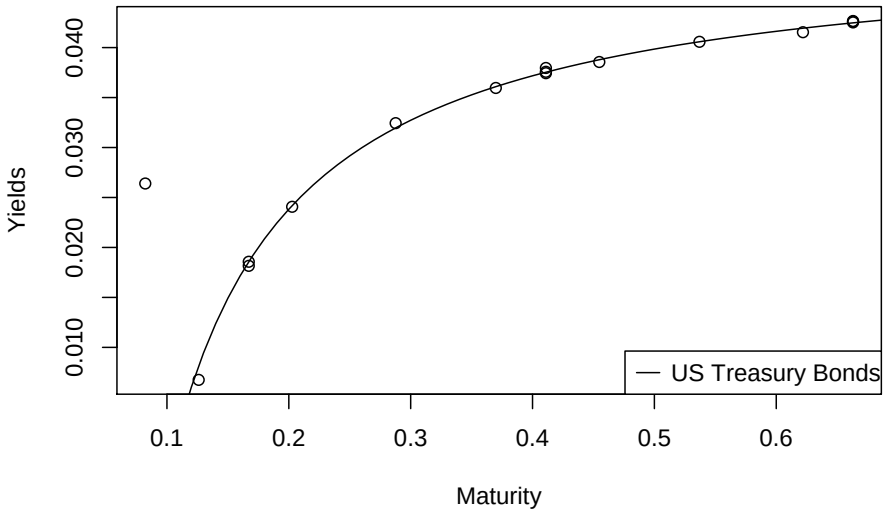

Abbildung 4.7: Zinsstruktur - US Treasury Tnote Bonds

```
##set paramters
#price an american call option
TypeFlag="ca"

#underlying price
S=87.72

#strike price
X=85
#time to maturity
Time=86/252
#risk free rate
r=0.03493302*252/86
#drift
b=r
#volatility
sigma=0.02379498*sqrt(252)
```

Zusätzlich muss noch die Anzahl der Schritte, in welche die Laufzeit unterteilt werden soll, angegeben werden. Um eine grafische Darstellung zu ermöglichen, wird die Schrittanzahl auf 6 gesetzt. Bei großen Schrittzahlen verliert die Grafik an Übersicht.

```
#binomial tree for demonstration
```

```
n=6

Tree<-BinomialTreeOption(TypeFlag , S, X,
        Time, r,b, sigma , n, title = NULL, description = NULL)


#plot tree
BinomialTreePlot(Tree,xlab="Time steps",ylab="")
```

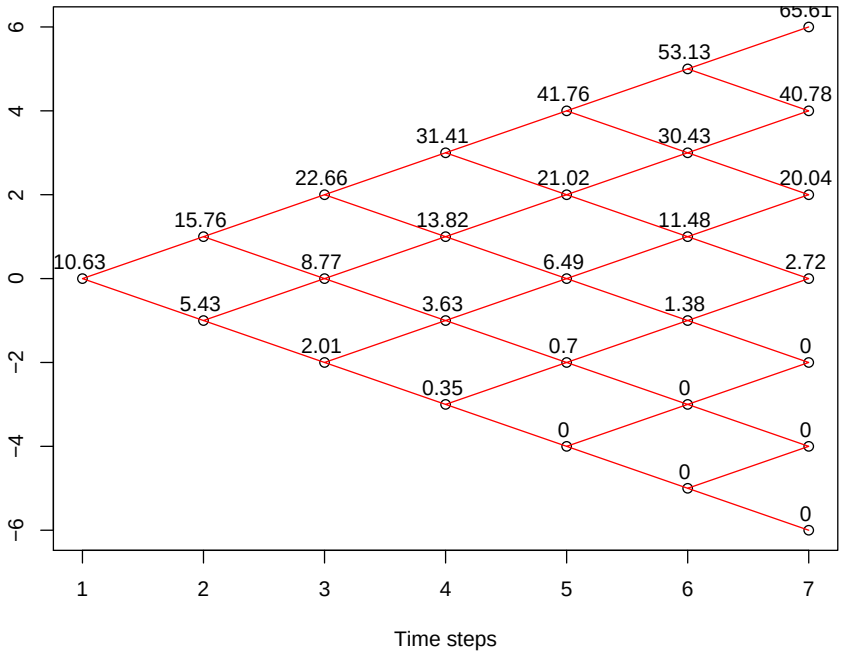

Abbildung 4.8: Amerikanische Optionswerte - Binomialbaum

Der Preis der Option bei dieser geringen Schrittzahl ergibt sich mit 10.63 USD. Neben jedem Knoten ist der Wert der Option zum jeweiligen Zeitpunkt vermerkt. Für aussagekräftigere Bewertungen macht es jedoch Sinn die Anzahl der Intervalle zu erhöhen. Der folgende R-Code erhöht die Schrittanzahl der Berechnung laufend. Die Abbildung 4.9 zeigt das, daraus resultierende, Konvergenzverhalten.

```
#convergence behaviour
price<-rep(NA,90)

for( i in 10:100) {

        price[i-10] <- CRRBinomialTreeOption(TypeFlag , S, X,
```

```
        Time, r,b, sigma , n=i, title = NULL, description = NULL)@price
   }

plot(11:100,price,type="l",col="blue",ylab="Option price", xlab="Index")
```

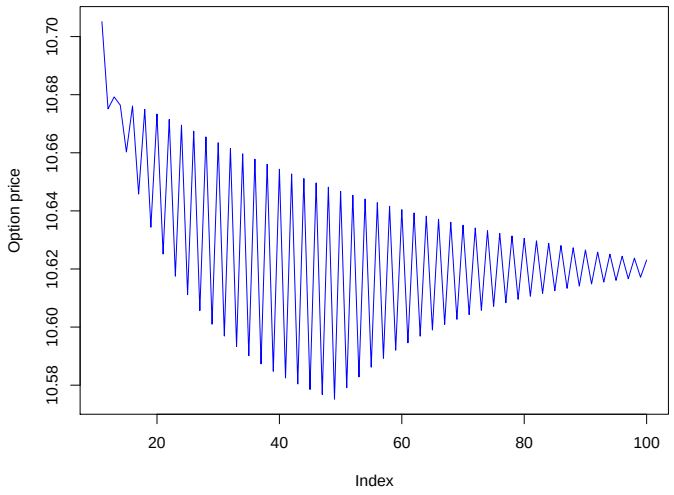

Abbildung 4.9: Konvergenzverhalten der Binomialbaummethode

Die Berechnung mit 50 Schritten ergibt einen Optionspreis von 10.65 USD und liefert damit schon eine sehr gute Annäherung an den quotierten Preis der Option (10.3 USD vgl. Tabelle 4.1). Wie Abbildung 4.9 zeigt, konvergiert der Preis der Option bei der Bewertung mittels Binomialbäumen in diesem Fall gegen 10.62.

```
#option price with time step = 50
n=50
CRRBinomialTreeOption(TypeFlag , S, X,
        Time, r,b, sigma , n, title = NULL, description = NULL)

Parameters:
          Value:
TypeFlag  ca
S         87.72
X         85
Time      0.341269841269841
r         0.102361872558140
b         0.102361872558140
sigma     0.377733597190534
n         50
```

```
Option Price:
10.64675
```

Die Berechnung des Optionspreises mit der entsprechenden Funktion aus dem Package `RQuantLib`, basierend auf der Methode der finiten Differenzen, ergibt einen Optionspreis von 10.618. Beide Bewertungsmethoden liefern dabei den selben Preis für die Option.

```
#load package
library(RQuantLib)
##set paramters
#price a american call option
type="call"

#underlying price
S=87.72

#strike price
X=85
#time to maturity
Time=86/252
#risk free rate
r=0.03493302*252/86

#volatility
sigma=0.02379498*sqrt(252)


AmericanOption("call",S,X,0,r,Time,sigma)

Concise summary of valuation for AmericanOption
 value  delta  gamma   vega  theta    rho divRho
10.618     NA     NA     NA     NA     NA     NA
```

4.4 Bewertung von Optionen mittels (quasi) Monte Carlo Simulation

Dieser Abschnitt gliedert sich in zwei Bereiche. Es wird jeweils die gleiche arithmetische asiatische Option mittels Monte Carlo Simulation bewertet. Die Preispfade, die für das Underlying angenommen werden, unterscheiden sich dabei grundlegend. Unter Punkt 4.4.1 wird für das Underlying ein Preispfad angenommen der durch eine geometrische Brownsche Bewegung beschrieben wird. Unter Punkt 4.4.2 wird der Preispfad mit einem Jump-Diffusion Modell modelliert.

Die arithmetische asiatische Option ist durch die Eigenschaften in Tabelle 4.3 gekennzeichnet.

Art	*arithmetische asiatische Call-Option*
Underlying Preis	*100*
Strike-Preis	*100*
Laufzeit	*1/12 Jahr*
Volatilität	*0.4*
risikoloser Zinssatz	*0.1*

Tabelle 4.3: Optionscharakteristika

4.4.1 Bewertung von arithmetischen asiatischen Optionen mit dem Package fOptions

Wie bereits angemerkt, wird in diesem Abschnitt der Preispfad durch eine geometrische Brownsche Bewegung beschrieben. Die Berechnung erfolgt mit den Funktionen des Packages `fOptions`, wobei verschiedene Zufallszahlen beziehungsweise quasi Zufallszahlen zur Verwendung kommen.

Für die Simulation muss zuerst der Zufallszahlengenerator, der Preispfad und die Payoff-Funktion spezifiert werden.

```
#load package fOptions
library(fOptions)

##generate sobol quasi random numbers
sobol<-function(mcSteps,pathLength,init,...){
        inov = rnorm.sobol(mcSteps,pathLength,init,...)
        inov
        }

##generate normal distributed random numbers
normal<-function(mcSteps,pathLength,init,...){
        inov=rnorm.pseudo(mcSteps,pathLength,init,...)
        inov
        }

##generate halton quasi random numbers
halton<-function(mcSteps,pathLength,init,...){
        inov = rnorm.halton(mcSteps,pathLength,init,...)
        inov
```

```
        }

##wiener path
wienerpath <- function(eps) {

        path=(b-sigma*sigma/2 )*delta.t + sigma*sqrt(delta.t)*eps
        path

        }

##payoff asian aritmetic call or put
aAsian<-function(path){
        St<-mean(S*exp(cumsum(path)))

        if (TypeFlag =="c") payoff = exp(-r*Time)*max(St-X,0)
        if (TypeFlag == "p") payoff = exp(-r*Time)*max(0,X-St)

        payoff
        }
```

Die Parameter werden gemäß Tabelle 4.3 definiert.

```
##define global parameters
#type of option
TypeFlag<<-"c"
#underlying price
S<<- 100
#strike price
X<<-100
#time to maturity
Time<<-1/12
#volatility
sigma<<- 0.4
#risk free rate
r <<- 0.1
#drift
b<<- 0.1
```

Über die Hauptfunktion wird die Simulation fertig spezifiziert. So muss noch der Simulationshorizont (*delta.t*), die Pfadlänge (*pathLength*), die Anzahl der Zufallsvariablen (*mcSteps*) und die Anzahl der Berechnungsschritte (*mcLoops*) angegeben werden. Alle zuvor getätigten Spezifikationen werden an die Funktion übergeben.

```
mc_sobol = MonteCarloOption(delta.t = 1/360, pathLength = 30,
                  mcSteps = 5000,  mcLoops = 50, init = TRUE,
```

```
innovations.gen = sobol,  path.gen = wienerpath,
payoff.calc = aAsian, antithetic = TRUE,
standardization = FALSE, trace = TRUE,
scrambling = 2, seed = 4711)
```

Die Tabelle 4.4 gibt den Mittelwert und das 95% Konfidenzintervall der Simulation über 500 Simulationsläufe wieder. Es zeigt sich, dass die Variante mit den quasi-Zufallszahlen gemäß Sobol das kleinste Konfidenzintervall liefert.

Zufallszahlen	Mittelwert	95% Konfidenzintervall
Sobol	2.9615	2.8346 3.0884
Halton	2.7700	2.5175 3.0225
normalverteilt	2.9188	2.6977 3.1398

Tabelle 4.4: Simulationsergebnisse

Die Abbildung 4.10 veranschaulicht die Simulationsergebnisse grafisch in Abhängigkeit von den gewählten (quasi) Zufallszahlen. Bei einer hohen Anzahl von Simulationsläufen liefert die Simulationen mit normalverteilten Zufallszahlen beinahe die selben Werte Simulationen mit Simulationen mit quasi- Zufallszahlen, die nach der Methode von Sobol erzeugt wurden.

Abbildung 4.10: Preis in Abhängigkeit der verwendeten (quasi) Zufallszahlen

4.4.2 Bewertung von Optionen unter der Verwendung eines Jump Diffusion Modells

Das unter Punkt 4.2.3 vorgestellte Modell kommt nun bei der Bewertung von Optionen zur Anwendung. Bewertet wird die Option aus Tabelle 4.3. Zur Simulation werden nicht die Funktionen des Packages `fOptions` verwendet. Vielmehr wurden eigene Funktionen geschrieben, die teilweise auf den Funktionen des Packages basieren, allerdings für die Problemstellung maßgeschneidert wurden. Der Source-Code ist im Anhang C zu finden.

Bevor die Simulation gestartet werden kann, müssen wieder diverse Parameter spezifiziert werden. Die Angaben unterscheiden sich nicht wesentlich von der zuvor durchgeführten Simulation, abgesehen vom Preispfad, der nun einem Jump Diffusion Prozess folgt. Die Eigenschaften dieses Prozesses können über die Parameterwahl $(a, b, lambda)$ gesteuert werden.

```
#characteristics of the option
optioncharac=list(Type="c",S=100,X=100,Time=1/12,sigma=0.4,r=0.1,mu=0.1)

#settigns of the Monte Carlo simulation
mcsettings=list(deltat=1/360,pathlength=30,mcsteps=5000,mcloops=50,
                        antithetic=FALSE,standardization=FALSE,trace=FALSE,
                        scrambling=2,seed=4711)

#settings of jump diffusion model
jumpsettings=list(a=-0.003,b=0.05,lambda=15)

#define path of the underlying
pathgen=jumppath

#specify payoff-function
payofffunc=aAsian
```

Über den Aufruf der Hauptfunktion wird die Simulation gestartet und liefert als Ergebnis eine Liste mit den simulierten Optionspreisen und Konfidenzintervallen der einzelnen Durchgänge.

```
#run mc simulation
mysim<-MCOptionJD(optioncharac,mcsettings,pathgen,payofffunc,jumpsettings)
```

Die Abbildung 4.11 veranschaulicht das Ergebnis einer Simulation mit 50 Durchgängen. Das Konvergenzverhalten nach 500 Simulationsläufen ist in Abbildung 4.12 dargestellt. Der Preis pendelt sich auf 9.28 ein.

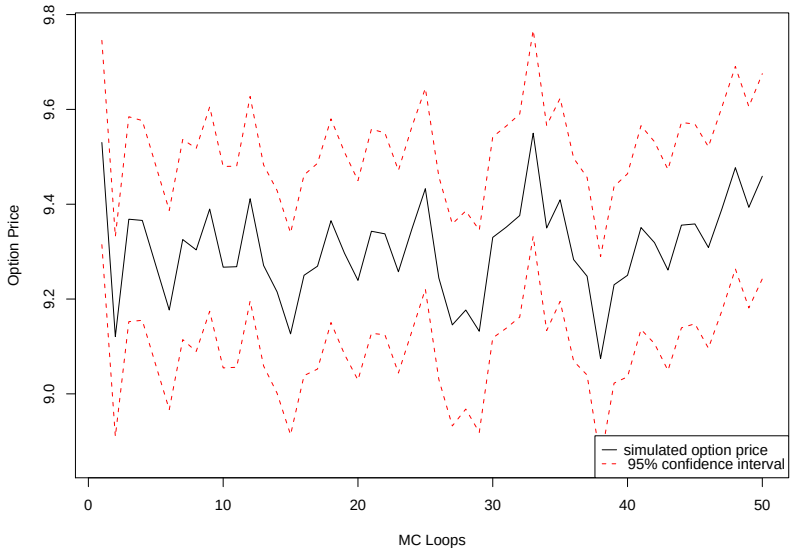

Abbildung 4.11: Simulierter Optionspreis mit Konfidenzintervallen

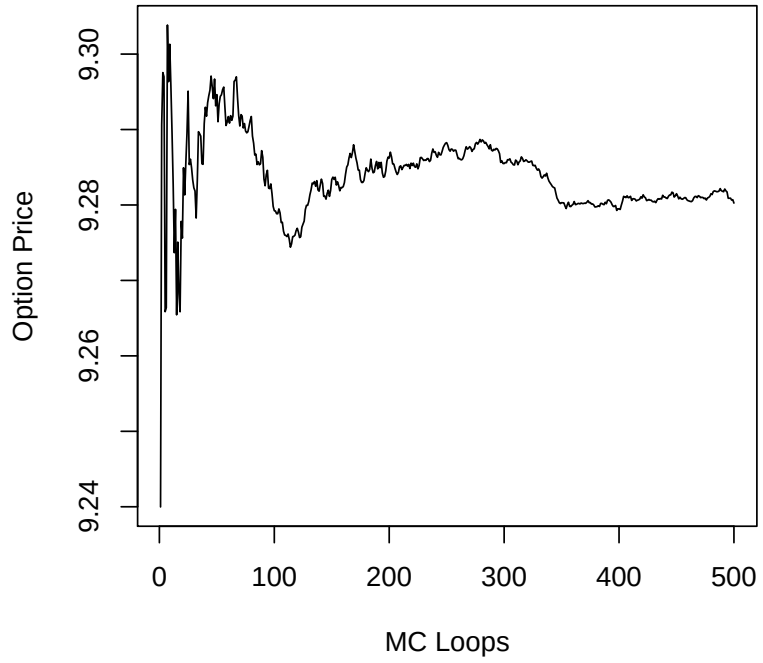

Abbildung 4.12: Konvergenzverhalten einer MC - Simulation

Kapitel 5

Zinsderivate und Short-Rate Modelle

In Kapitel 4 wurden Derivate auf Aktien behandelt. An dieser Stelle wird die Bewertung von Zinsderivaten näher beleuchtet. Unter einem Zinsderivat kann man ganz allgemein ein Finanzinstrument verstehen, bei dem die Auszahlung von der Höhe des Zinssatzes abhängt.

Die Bewertung von Zinsderivaten unterscheidet sich grundlegend von der Bewertung von Derivaten auf Aktien. Es lassen sich folgende Gründe für eine schwierigere Bewertung finden (vgl. Hull, 2000, S.508):

- das Verhalten von Zinssätzen ist wesentlich komplexer als das von Aktienkursen
- für die Bewertung von Zinsderivaten muss ein Modell zur Beschreibung des Verhaltens der Nullkupon-Zinsstrukturkurve *(zero coupon yield curve)* gefunden werden
- die Volatilität an verschiedenen Punkten der Zinsstrukturkurve (*yield curve*) kann sich wesentlich unterscheiden
- Zinssätze werden sowohl zum Diskontieren als auch zur Definition der Auszahlung eines Derivats verwendet.

Ähnlich wie in Kapitel 4 gibt es eine Reihe von analytischen Modellen für die Bewertung von Zinsderivaten. Die Ausführungen werden sich allerdings auf numerische Verfahren und Modelle beschränken. Der Schwerpunkt wird nicht auf der Vorstellung von Zinsderivaten liegen. Im Zuge der Beispiele werden einige wenige Zinsderivate kurz charakterisiert. Vielmehr sollen Modelle für die Short-Rate theoretisch vorgestellt und die Implementierung solcher Modelle in `R` gezeigt werden. Zuerst werden die theoretischen Grundlagen der Modelle behandelt und

aufbauend darauf die Implementierung allgemein erläutert. Abgeschlossen wird dieses Kapitel mit einer Präsentation von Beispielen, bei denen die implementierten Modelle zur Bewertung einfacher Zinsderivate zur Anwendung kommen.

5.1 Allgemeine Grundlagen

Die Ausführung stützen sich, sofern nicht anders angegeben, auf (Clewlow und Strickland, 1998, S.181 ff).

Für Derivate auf Aktien, Aktienindizes, Wechselkurse und auch Waren kann der Level des Underlyings und die damit verbundene Volatilität durch eine einzige Variable und einen Parameter repräsentiert werden. Für Zinsderivate besteht dieses Underlying allerdings aus der gesamten Zinsstruktur. Für die Bewertung von Zinsderivaten ist daher die Bestimmung der Zinsstruktur und der damit verknüpften Volatilität von großer Wichtigkeit.

Sei $P(t,u)$ der Preis einer Nullkupon-Anleihe zum Zeitpunkt t, die zum Zeitpunkt u (am Ende der Laufzeit) einen Betrag von einer Geldeinheit ausbezahlt. Die Zinsstruktur kann dann etwa über Spot-Rates beschrieben werden:

$$P(t,u) = e^{s(t,u)(u-t)}, \tag{5.1}$$

und es gilt

$$s(t,u) = -\frac{1}{u-t} lnP(t,u). \tag{5.2}$$

Eine wichtige Rolle wird in weiterer Folge die Rendite (*yield*) von Anleihen mit minimaler Restlaufzeit spielen. Dieser Zinssatz wird als momentaner kurzfristiger Zinssatz (*short rate*) r bezeichnet.

Eine äquivalente Formulierung des Nullkupon-Anleihenpreises ergibt sich über die Forward-Rate Funktion $f(t,u)$. Sie gibt zu jedem Zeitpunkt t den in u gültigen momentanen kurzfristigen Zinssatz an. Der Preis einer Nullkupon-Anleihe lässt sich über

$$f(t,u) = -\frac{\delta}{\delta u} lnP(t,u) \tag{5.3}$$

als der mit den Forward-Rates abdiskontierte Cashflow darstellen

$$P(t,u) = e^{-\int_t^u f(t,\tau)d\tau}. \tag{5.4}$$

Die Spot-Rate ergibt sich damit als Durchschnitt der Forward-Rates

$$s(t,u) = -\frac{1}{u-t}\left(\int_t^u f(t,\tau)d\tau\right). \tag{5.5}$$

Bei der Bewertung von Zinsderivaten kann man grundsätzlich zwei Varianten unterscheiden. Die erste basiert auf der Modellierung von Anleihenpreisen und beruht auf dem Modell von Black und Scholes. Wie Abbildung 5.1 zeigt, beruht der zweite Ansatz auf der Modellierung von Zinssätzen. In weiterer Folge werden nur Modelle dieser Variante behandelt.

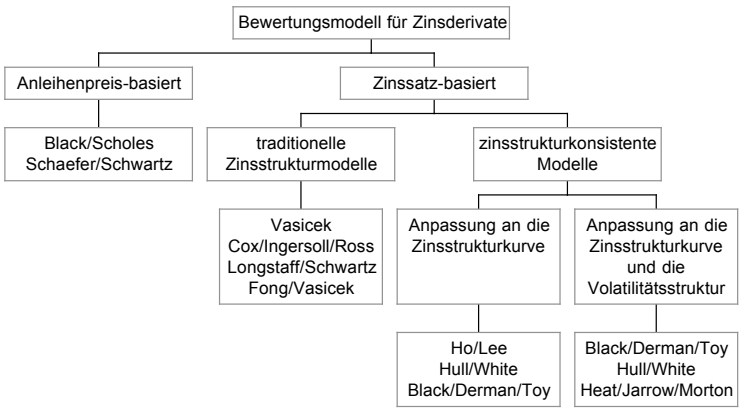

Abbildung 5.1: Short-Rate Modelle (vgl. Clewlow und Strickland, 1998, S.189 Figure 6.3)

Die Herleitung eines Modells für den Zinssatz, das keine Abitrage zulässt, kann über den Prozess, dem die Short-Rate folgt, erfolgen. Es lässt sich zeigen, dass der Prozess der Short-Rate in einer risikoneutralen Welt die Zinsstruktur determiniert. Der Preis einer Nullkupon-Anleihe ergibt sich mit:

$$P(t,u) = \hat{\mathbb{E}}_t\left[e^{-\int_t^u r(\tau)d\tau}\right]. \tag{5.6}$$

Wobei $\hat{\mathbb{E}}_t$ den Erwartungswert in einer risikoneutralen Welt mit der Informationsmenge t und $r(\tau)$ den Pfad der Short-Rate von t nach u bezeichnet. Der Wert eines Zinsderivats mit einer Laufzeit von T kann über den Erwartungswert

$$\hat{\mathbb{E}}_t\left[e^{-\int_t^T r(\tau)d\tau}\text{Payoff}\right] \tag{5.7}$$

bestimmt werden. Payoff steht dabei für eine beliebige Auszahlungsfunktion. Für die Bewertung muss nun der Prozess der Short-Rate r vollständig festgelegt sein.

5.2 Short-Rate Modelle

Abbildung 5.1 zeigt die gängigen Short-Rate Modelle, wobei diese Grafik keinen Anspruch auf Vollständigkeit erhebt. Die Modelle, die in weiterer Folge in diesem Abschnitt vorgestellt werden, können zu den zinsstrukturkonsistenten Modellen gezählt werden. So wird das Black-Derman-Toy Modell und das Modell von Hull-White vorgestellt. Zinsstrukturkonsistente Modelle versuchen die Zinsstruktur so zu modellieren, dass sie möglichst konsistent mit den ursprünglichen, also beobachteten, Marktdaten sind.

5.2.1 Das Modell von Black, Derman und Toy

Black, Derman und Toy (1990) entwickelten ein Einfaktormodell für die Short-Rate, das die Zinsstruktur und auch die Volatilitätsstruktur modelliert. In weiterer Folge wird dieses Modell als BDT Modell bezeichnet. Ursprünglich beschreibt dieses Modell die Entwicklung der Zinsstruktur in einem Binomialbaum. Der Baum wird dabei so konstruiert, dass er automatisch die beobachtete Yield- und Volatilitätsstruktur nachbildet. Im Abschnitt 5.3 wird die Methode der Binomialbäume für das BDT Short-Rate Modell näher erläutert.

Es wurde gezeigt, wenn die Größe des Zeitintervalls im Binomialbaum gegen Null geht, kann die Short-Rate durch folgende stochastische Differentialgleichung beschrieben werden:

$$d\ lnr(t) = \left(\theta(t) + \frac{\sigma'(t)}{\sigma(t)} lnr(t)\right) dt + \sigma(t) dz. \tag{5.8}$$

Das BDT Modell beinhaltet zwei zeitunabhängige Funktionen ($\theta(t)$ und σ), die so gewählt werden, dass die beobachtete Zinsstruktur und Volatilitätsstruktur nachgebildet wird. Der Zinssatz kann im BDT Modell nicht negativ werden, da die Änderungen der Short-Rate lognormalverteilt sind. Nachteilig wirkt sich die Log-Normalverteilung aber auf die Existenz von analytischen Lösungen für die Preise von Optionen auf Anleihen aus. Daher sind numerische Verfahren für die Bewertung erforderlich. Im Abschnitt 5.3 wird im Detail darauf eingegangen.

5.2.2 Das Modell von Hull und White

Basierend auf dem Modell von Vasicek (1977) und Cox, Ingersoll und Ross (1985) entwickelten Hull und White (1990) ein Einfaktormodell für die Short-Rate, das durch folgende stochastische Differentialgleichung beschrieben werden kann:

$$dr = (\theta(t) - \alpha r)\, dt + \sigma dz, \tag{5.9}$$

beziehungsweise durch

$$dr = \alpha \left(\frac{\theta(t)}{\alpha} - r \right) dt + \sigma dz. \tag{5.10}$$

Wobei α und σ Konstanten sind. Das Hull-White Modell kann als das Modell von Ho und Lee (1986) mit einer Mean Reversion der Rate α oder als Vasicek Modell mit zeitabhängigem Reversionsniveau bezeichnet werden.

Unter Mittelwerttendenz (*mean reversion*) versteht man ein Phänomen, dass Zinssätze im Laufe der Zeit auf ein langfristiges Durchschnittszinsniveau (*Reversionsniveau, reversion level*) zurücktendieren (vgl. Hull, 2000, S. 539). Ist die Short-Rate r hoch, führt die Mean Reversion zu einer negativen Driftrate, bei niedrigem r bewirkt sie eine positive Driftrate.

Der zeitabhängige Driftparameter $\theta(t)$ erlaubt es die beobachtete Zinsstruktur zu modellieren und ergibt sich mit

$$\theta(t) = \frac{\delta f(0,t)}{\delta t} + \alpha f(0,t) + \frac{\sigma^2}{2\alpha}(1 - e^{-2\alpha t}). \tag{5.11}$$

Die analytische Formel für Anleihenpreise ist im Hull-White Modell durch

$$P(T,u) = A(T,u)e^{-B(T,u),r(T)} \tag{5.12}$$

gegeben.

Mit

$$B(T,u) = \frac{1 - e^{-\alpha(u-T)}}{\alpha}$$

und

$$lnA(T,u) = ln\frac{P(t,u)}{P(t,T)} - B(T,u)\frac{\delta lnP(t,T)}{\delta T} - \frac{1}{4\alpha^3}\left(e^{-\alpha(u-t)} - e^{-\alpha(T-t)}\right)^2\left(e^{2\alpha(T-t)-1}\right).$$

5.3 Modellieren der Short-Rate mittels Binomialbäumen

Im Gegensatz zu anderen Modellen besitzt das BDT Modell keine analytische Lösung für einfache Zinsderivate, wie für Optionen auf Bonds oder Swaptions. Für die Bewertung von Derivaten muss daher ein numerisches Verfahren angewendet werden. In diesem Fall kommen

Binomialbäume zur Modellierung des Short-Rate Modells zur Anwendung. Wie der vorherige Abschnitt stützen sich auch die Ausführungen dieses Abschnitts auf (Clewlow und Strickland, 1998, S. 234 ff).

An dieser Stelle sollen kurz die wichtigsten Grundlagen erläutert werden. In weiterer Folge wird dann im Detail auf die Anpassung des Modells an die Zins- und Volatilitätstruktur eingegangen.

Abbildung 5.2 zeigt einen Baum mit drei Schritten. Die risikoneutrale Wahrscheinlichkeit einer Auf- beziehungsweise Abwärtsbewegung wird mit 0.5 angenommen. Auch wird die ursprüngliche Zins - und Volatilitätsstruktur als gegeben vorausgesetzt. Die Short-Rate ist zu Beginn mit r gegeben. Als nächster Schritt werden die Short-Rates r_u und r_d so gewählt, dass die ursprüngliche Zins- und Volatilitätsstruktur wiedergegeben wird. Um das zu gewährleisten, muss eine Nullkupon -Anleihe, die zum Zeitpunkt 2 fällig wird, bewertet werden, sodass der Wert in allen 3 Knoten eine Geldeinheit beträgt. y_k bezeichnet die Rendite (*yield*) einer Nullkupon-Anleihe P_k mit der Fälligkeit zu $k\Delta t$ $(k = 1, \ldots, N)$ und σ_k die Volatilität von y_k. Es gilt:

$$P_k = e^{y_k(k\Delta t)}. \tag{5.13}$$

Um die ursprüngliche Volatilität nachzubilden, muss gelten:

$$y_k^u = y_k^d e^{2\sigma_k\sqrt{\Delta t}}, \tag{5.14}$$

und somit

$$\sigma_k = \frac{1}{2\sqrt{\Delta t}} ln\left(\frac{y_k^u}{y_k^d}\right). \tag{5.15}$$

y_k^k und y_k^d werden im Baum durch eine Anleihe mit der Fälligkeit $k+1$ berechnet. Um r_u und r_d zu bestimmen, kommen numerische Verfahren, wie die Newton-Raphson Methode zur Anwendung. Nach der Berechnung von r_u und r_d können r_{uu},r_{ud} und r_{dd} durch die Anpassung der Zins- und Volatilitätsstruktur einer dreiperiodigen Anleihe bestimmt werden. Ein analoges Vorgehen ist für die restlichen Schritte im Baum notwendig.

Eine effiziente Konstruktion solcher Binomialbäume kann mittels der Methode der Vorwärtsinduktion (*forward induction*), die von Jamshidian (1991) entwickelt wurde, erreicht werden. Jamshidian (1991) zeigt, dass die Short-Rate r zum Zeitpunkt t im BDT Modell bestimmt ist durch

$$r(t) = U(t)e^{\sigma(t)z(t)}. \tag{5.16}$$

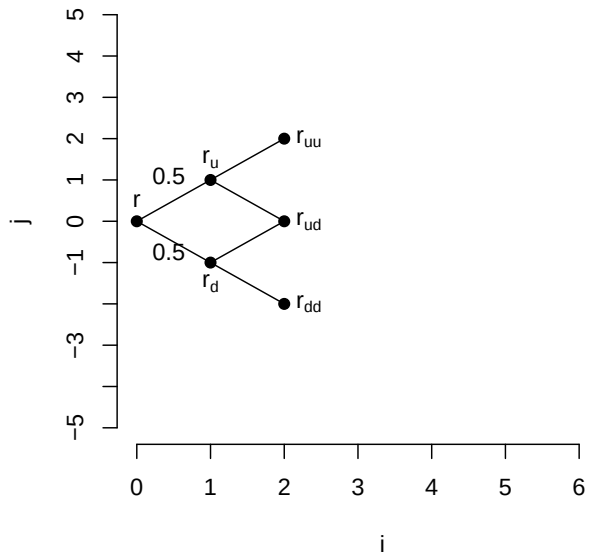

Abbildung 5.2: Binomialbaum (vgl. Clewlow und Strickland, 1998, S.234)

Wobei $U(t)$ der Median der (log-normal)-Verteilung von r zum Zeitpunkt t, $\sigma(t)$ die Volatilität der Short-Rate zum Zeitpunkt t und $z(t)$ eine Brownsche Bewegung zum Zeitpunkt t ist. Um die Zins- und Volatilitätsstruktur nachzubilden, muss $U(t)$ und $\sigma(t)$ in jedem Zeitpunkt ermittelt werden. Wird die Volatilität als konstant angenommen, dies entspricht nur einer Anpassung an die Zinsstruktur, ergibt sich die Short-Rate mit

$$r(t) = U(t)e^{\sigma z(t)}. \tag{5.17}$$

j aus Abbildung 5.2 folgt einer zentrierten Binomialverteilung mit Mittelwert 0 und Varianz N (N steht für die Anzahl der Zeitintervalle). Damit ist $j\sqrt{\Delta t}$ binomialverteilt mit Mittelwert 0 und Varianz t. Geht $\Delta t \to 0$, konvergiert der Prozess gegen den Wiener Prozess $z(t)$. Gleichung 5.16 kann diskretisiert werden, die Short-Rate in einem Punkt (i, j) im Binomialbaum (vgl. Abbildung 5.2) ist dann durch

$$r_{i,j} = U(i)e^{\sigma(i)j\sqrt{\Delta t}} \tag{5.18}$$

gegeben. Um die Short-Rate $r_{i,j}$ für alle Knoten im Baum zu bestimmen, muss daher $U(i)$ und $\sigma(i)$ für jeden Knoten bestimmt werden.

Berechnung der zeitabhängigen Funktionen $U(i)$ und $\sigma(i)$

Für die Bestimmung der Funktionen kommen Arrow-Debreu Zustandspreise zur Anwendung. $Q_{i,j}$ wird als Wert eines Wertpapiers zum Zeitpunkt 0 definiert. Das Wertpapier zahlt eine Geldeinheit bei Erreichen des Knotens (i, j), ansonsten 0. $Q_{0,0}$ wird mit 1 festgelegt. Die Zustandspreise können als diskontierte Wahrscheinlichkeiten interpretiert werden und sind gleichzeitig die Bausteine für alle Wertpapiere.

Der Preis einer Nullkupon-Anleihe mit einer Fälligkeit zu $(i + 1)\Delta t$ kann mit Hilfe dieser Zustandspreise und einperiodiger Diskontfaktoren ausgedrückt werden als

$$P(i+1) = \sum_j Q_{i,j} d_{i,j}. \tag{5.19}$$

Unter Annahme diskreter Verzinsung ergibt sich

$$d_{i,j} = \frac{1}{1 + r_{i,j}\Delta t}, \tag{5.20}$$

bei stetiger Verzinsung gilt

$$d_{i,j} = e^{-r_{i,j}\Delta t}. \tag{5.21}$$

Die Zustandspreise im Knoten (i, j) ergeben sich durch die Methode der Vorwärtsinduktion abhängig von den Werten zum Zeitpunkt $i - 1$ gemäß der folgenden Gleichung (unter der Annahme diskreter Verzinsung):

$$Q_{i,j} = \frac{1}{2} Q_{i-1,j-1} d_{i-1,j-1} + \frac{1}{2} Q_{i-1,j+1} d_{i-1,j+1} \qquad j = -i+1, \ldots, i-1 \tag{5.22}$$

$$= \frac{1}{2} Q_{i-1,j-1} \frac{1}{1 + r_{i-1,j-1}\Delta t} + \frac{1}{2} Q_{i-1,j+1} \frac{1}{1 + r_{i-1,j+1}\Delta t}$$

Die Gleichung 5.22 ist gültig für alle Zeitpunkte i, ausgenommen sind die Extrempunkte $(i, j = i), (i, j = -i)$. Für die gilt folgende Beziehung

$$Q_{i,i} = \frac{1}{2} Q_{i-1,i-1} d_{i-1,i-1}, \tag{5.23}$$

$$Q_{i,-i} = \frac{1}{2} Q_{i-1,-i+1} d_{i-1,-i+1}. \tag{5.24}$$

5.3.1 Anpassung des Black-Derman-Toy Modells an die Zinsstruktur

Die Anpassung des BDT Modells erfolgt nur an die Zinsstruktur, wenn die Short-Rate Volatilität als konstant angenommen wird. Der Mean Reversion Parameter der Gleichung 5.8 $\left(\frac{\sigma'(t)}{\sigma(t)}\right)$ ist daher Null und die stochastische Differentialgleichung 5.8 ändert sich auf

$$d\ lnr = \theta(t)dt + \sigma dz. \tag{5.25}$$

Die Diskretisierung ergibt für die Short-Rate $r_{i,j}$:

$$r_{i,j} = U(i)e^{\sigma j\sqrt{\Delta t}}. \tag{5.26}$$

Aus Gleichung 5.19 in Kombination mit 5.26 ergeben sich die Preise von Nullkupon-Anleihen mit

$$P(i+1) = \sum_j Q_{i,j}\frac{1}{1+U(i)e^{\sigma j\sqrt{\Delta t}}\Delta t}. \tag{5.27}$$

Diese Gleichung kann allerdings nicht explizit gelöst werden, daher muss $U(i)$ über eine numerische Methode (z.B. Newton-Raphson) ermittelt werden. Mittels $U(i)$ kann dann die Short-Rate $r_{i,j}$ berechnet werden.

Zusammenfassend wird die Berechnung der Short-Rates schrittweise angeführt. Wobei $i > 0$ und $U(i-1), Q_{i-1,j}, d_{i-1,j}$ für alle j zum Zeitpunkt $i-1$ bereits errechnet wurden.

Schritt 1:

Die Werte des Startknotens $(0,0)$ sind definitionsgemäß gegeben durch:

$$\begin{aligned} U(0) = 0 = r_{0,0} &= y(1) \\ Q_{0,0} &= 1 \\ d_{0,0} &= \frac{1}{1+r_{0,0}\Delta t}. \end{aligned}$$

Schritt 2:

Berechne $Q_{i,j}$:

$$\begin{aligned} Q_{i,i} &= \frac{1}{2}Q_{i-1,i-1}d_{i-1,i-1} \\ Q_{i,j} &= \frac{1}{2}Q_{i-1,j-1}d_{i-1,j-1} + \frac{1}{2}Q_{i-1,j+1}d_{i-1,j+1} \qquad j = -i+1 \ldots, i-1 \\ Q_{i,-i} &= \frac{1}{2}Q_{i-1,-i+1}d_{i-1,-i+1} \end{aligned}$$

Schritt 3:

Berechne $U(i)$ aus

$$P(i+1) = \sum_j Q_{i,j} \frac{1}{1 + U(i)e^{(\sigma j\sqrt{\Delta t})}\Delta t}$$

mit einem numerischen Lösungsverfahren.

Schritt 4:

Berechne die Short-Rate mit $U(i)$ und aktualisiere für alle Knoten zum Zeitpunkt i die Diskontfaktoren.

$$\begin{aligned} r_{i,j} &= U(i)e^{\sigma j\sqrt{\Delta t}} \\ d_{i,j} &= \frac{1}{1 + r_{i,j}\Delta t} \end{aligned}$$

5.3.2 Anpassung des Black-Derman-Toy Modells an die Zins- und Volatilitätsstruktur

Wie bereits mehrfach erwähnt, kann das BDT Modell auch auf die Zins - und Volatilitätsstruktur angepasst werden. Die Volatilität $\sigma(t)$ ist nun zeitabhängig. Die Short-Rate für einen Knoten (i, j) ergibt sich dann mit

$$r_{i,j} = U(i)e^{(\sigma(i)j\sqrt{\Delta t})}. \tag{5.28}$$

$P^u(i)$ und $P^d(i)$ bezeichnen nun die Diskontfunktionen für eine Aufwärts- beziehungsweise Abwärtsbewegung im Binomialbaum. Die dazugehörige Rendite wird mit $y^u(i)$ beziehungsweise $y^d(i)$ bezeichnet. Die Spezifizierung der Spot-Rate und der Volatilität zum Zeitpunkt $i = 0$ ist gleichbedeutend mit der Spezifizierung der Diskontfunktionen $P^u(i)$ und $P^d(i)$ für

alle $i \geq 1$. $P^u(i)$ und $P^d(i)$ müssen daher auch konsistent mit den bekannten Werten von $y(i)$ und $\sigma_y(i)$ sein. Der erste Schritt bei der Baumkonstruktion ist somit die Bestimmung von $P^u(i)$ und $P^d(i)$ für alle $i \geq 2$.

Der Zusammenhang zwischen $P(i)$, $P^u(i)$ und $P^d(i)$ ergibt sich über

$$P(i) = \frac{1}{1 + r_{0,0}\Delta t}\left(0.5P^u(i) + 0.5P^d(i)\right) \qquad i = 2, \ldots, N. \tag{5.29}$$

Die Volatilität kann nun über $P(i)$ und $P^u(i)$ ausgedrückt werden (vgl. Gleichung 5.15):

$$\sigma_y(i) = \frac{1}{2\sqrt{\Delta t}} ln\left(\frac{P(i)^u}{P(i)^d}\right). \tag{5.30}$$

Die Lösung der Gleichungen 5.29 und 5.30 ergibt

$$P^d(i) = P^u(i)^{\exp(2\sigma_y(i)\sqrt{\Delta t})} \tag{5.31}$$

und $P^u(i)$ ist die Lösung von

$$P^u(i) + P^u(i)^{\exp(2\sigma_y(i)\sqrt{\Delta t})} = 2P(i)(1 + r_{0,0}\Delta t). \tag{5.32}$$

Wiederum wird die Methode der Vorwärtsinduktion verwendet um die zeitabhängigen Funktionen zu bestimmen. Die Zustandspreise ergeben sich nun in Abhängigkeit von den Aufwärts- beziehungsweise Abwärtsknoten. $Q^u_{i,j}$ bezeichnet dabei den Wert eines Wertpapiers ausgehend vom Knoten u (vgl. Abbildung 5.2) , das eine Geldeinheit ausbezahlt falls der Knoten (i, j) erreicht wird und 0 andernfalls. Selbiges gilt für $Q^d_{i,j}$, aber ausgehend vom Knoten d.

$$Q^u_{i,j} = \frac{1}{2}Q^u_{i-1,j-1}d_{i-1,j-1} + \frac{1}{2}Q^u_{i-1,j+1}d_{i-1,j+1} \tag{5.33}$$

$$Q^d_{i,j} = \frac{1}{2}Q^d_{i-1,j-1}d_{i-1,j-1} + \frac{1}{2}Q^d_{i-1,j+1}d_{i-1,j+1} \tag{5.34}$$

Für die Berechnung von $U(i)$ und $\sigma(i)$ müssen nun die folgenden zwei Gleichungen numerisch gelöst werden:

$$P^u(i+1) = \sum_j Q^u_{i,j}d_{i,j}, \tag{5.35}$$

$$P^d(i+1) = \sum_j Q^d_{i,j}d_{i,j}. \tag{5.36}$$

Zusammenfassend wird die Berechnung der Short-Rates schrittweise angeführt. Wobei $i > 0$ und $U(i-1), \sigma(i-1), Q^u_{i-1,j}, Q^d_{i-1,j}, d_{i-1,j}$ für alle j zum Zeitpunkt $i-1$ bereits errechnet wurden.

Schritt 1:

Die Werte des Startknotens $(0,0)$ sind definitionsgemäß durch

$$\begin{aligned} U(0) = 0 = r_{0,0} = y(1), \\ Q^u_{1,1} = 1, \\ Q^d_{1,-1} = 1, \\ \sigma(0) = \sigma_y(1). \\ d_{0,0} = \frac{1}{1 + r_{0,0}\Delta t} \end{aligned}$$

gegeben.

Schritt 2:

Berechne $P^u(i)$ und $P^d(i)$ für $i = 2$ bis N.

$$P^d(i) = P^u(i)^{\exp(-2\sigma_y(i)\sqrt{\Delta t})}.$$

$P^u(i)$ ergibt sich als Lösung der Gleichung

$$P^u(i) + P^u(i)^{\exp(-2\sigma_y(i)\sqrt{\Delta t})} = 2P(i)(1 + r_{0,0}\Delta t)$$

Schritt 3:

Berechnung von $Q^u_{i,j}$ und $Q^d_{i,j}$.

$$\begin{aligned} Q^u_{i,j} &= \frac{1}{2} Q^u_{i-1,j-1} d_{i-1,j-1} + \frac{1}{2} Q^u_{i-1,j+1} d_{i-1,j+1} \\ Q^d_{i,j} &= \frac{1}{2} Q^d_{i-1,j-1} d_{i-1,j-1} + \frac{1}{2} Q^d_{i-1,j+1} d_{i-1,j+1} \end{aligned}$$

Schritt 4:

Berechnung von $U(i)$ und $\sigma(i)$ mit Hilfe von $P^u(i+1)$ und $P^d(i+1)$.

$$\begin{aligned} P^u(i+1) &= \sum_j Q^u_{i,j} \frac{1}{1+U(i)e^{\sigma(i)j\sqrt{\Delta t}}\Delta t} \\ P^d(i+1) &= \sum_j Q^d_{i,j} \frac{1}{1+U(i)e^{\sigma(i)j\sqrt{\Delta t}}\Delta t} \end{aligned}$$

Schritt 5:

Über die berechneten Werte von $U(i)$ und $\sigma(i)$ kann die Short-Rate und der einperiodige Diskontfaktor für alle j zum Zeitpunkt i berechnet werden:

$$\begin{aligned} r_{i,j} &= U(i)e^{\sigma(i)j\sqrt{\Delta t}}, \\ d_{i,j} &= \frac{1}{1+r_{i,j}\Delta t}. \end{aligned}$$

5.3.3 Bewertung von Zinsderivaten mittels Binomialbäumen

Für die Bewertung von Zinsderivaten ist nun der Binomialbaum mit den ermittelten Short-Rates erforderlich. Aufbauend auf diesen Binomialbaum soll gezeigt werden, wie einfache Zinsderivate mit der Methode der Rückwärtsinduktion bewertet werden können. Wiederum kommen Binomialbäume für die Bewertung zur Anwendung. Die Struktur dieser Bäume entspricht der bisher in diesem Abschnitt verwendeten.

Bewertung von Optionen auf Nullkupon-Anleihen

Es soll eine Option auf eine Nullkupon-Anleihe bewertet werden. Die Laufzeit der Option beträgt dabei T, die der Anleihe s ($T \leq s$). Die Zeitschritte bis zur Fälligkeit der Option beziehungsweise der Anleihe werden mit N_s ($s = N_s\Delta t$) beziehungsweise N_T ($T = N_T\Delta t$) bezeichnet. Für die Bewertung muss der Short-Rate Baum bis zum Zeitpunkt N_s berechnet worden sein. Der Wert der Nullkupon-Anleihe mit einer Fälligkeit s im Knoten (i,j) wird mit $Ps_{i,j}$ bezeichnet.

Als erster Schritt bei der Bewertung wird der Wert der Nullkupon-Anleihe für jeden Knoten (i,j) im Baum berechnet. $Ps_{N_s,j}$ ist definitionsgemäß 1. Ausgehend von diesen Werten werden mittels Rückwertsinduktion die übrigen Werte über die Beziehung

$$Ps_{i,j} = \frac{1}{2} d_{i,j} \left(Ps_{i+1,j+1} + Ps_{i+1,j-1} \right), \tag{5.37}$$

die für alle Knoten j zum Zeitpunkt i gilt, berechnet. $d_{i,j}$ ergibt sich dabei aus dem zuvor berechneten Short-Rate Baum.

Der nächste Schritt besteht aus der Berechnung des Payoffs der Option zum Zeitpunkt der Fälligkeit. Je nach Auszahlungsfunktion ergibt sich etwa der Wert einer Call-Option mit Strike-Preis X für alle Knoten j zum Zeitpunkt i mit

$$C_{N_T,j} = max\{0, Ps_{N_T,j} - X\}. \tag{5.38}$$

Für eine Put-Option gilt:

$$P_{N_T,j} = max\{0, X - Ps_{N_T,j}\}. \tag{5.39}$$

Ausgehend von den Optionswerten der Knoten zum Zeitpunkt N_T kann nun der Wert einer europäischen Option für alle übrigen Knoten bis zum Ausgangsknoten $(0, 0)$ mittels

$$C_{i,j} = \frac{1}{2} d_{i,j} \left(C_{i+1,j+1} + C_{i+1,j-1} \right) \tag{5.40}$$

berechnet werden. Allerdings kann die Berechnung durch die Tatsache, dass die Zustandspreise den diskontierten Wahrscheinlichkeiten entsprechen, vereinfacht werden. Der Wert ergibt sich im Falle einer Call-Option mit

$$C_{0,0} = \sum_j Q_{N_T,j} max\{0, Ps_{N_T,j} - X\}. \tag{5.41}$$

Für eine Put-Option ist die Auszahlungsfunktion entsprechend anzupassen.

Für die Bewertung einer amerikanischen Option wird der Wert der Option in jedem Knoten als das Maximum aus dem intrinsischen Wert und dem diskontierten Erwartungswert berechnet. Für eine amerikanische Call-Option ergibt sich der Wert für alle Knoten j zum Zeitpunkt i mit

$$C_{i,j} = max \left\{ Ps_{i,j} - X,\ \frac{1}{2} d_{i,j} \left(C_{i+1,j+1} + C_{i+1,j-1} \right) \right\}. \tag{5.42}$$

Der Wert der amerikanischen Option zum Zeitpunkt 0 ist dann durch den Anfangsknoten $(0, 0)$ gegeben.

Bewertung von europäischen Swaptions

Unter Swaptions versteht man Optionen, bei denen das Underlying Zinsswaps sind. Die Swap-Rate ist dabei der feste Zinssatz, der durch einen variablen Zinssatz, z.B. LIBOR-basierten

(*London Interbank Offered Rate*), ausgetauscht werden kann. Ein Zinsswap kann als Austausch einer festverzinslichen Anleihe (z.B. Kuponanleihe) mit einer Anleihe, deren Zahlungen auf variablen Zinssätzen basieren, gesehen werden. Zu Beginn der Laufzeit des Swaps stimmt der Wert der Anleihe mit variabler Verzinsung mit dem Nominalwert überein. Eine europäische Swaption kann daher als Option, eine Kuponanleihe gegen den Nominalwert des Swaps zu tauschen, betrachtet werden. Der Wert der Swaption ist damit der Wert einer Option auf eine Kuponanleihe mit einem Strike-Preis, der dem Nominalwert des Swaps entspricht. Für die weiteren Ausführungen wird der Nominalwert des Swaps auf eins normiert.

Unter einer Payer-Swaption versteht man eine Option, die den Inhaber berechtigt feste gegen variable Zahlungen zu tauschen, also eine Put-Option auf eine Anleihe mit fester Verzinsung. Eine Receiver-Swaption ist dann eine Option, bei der der Inhaber berechtigt ist, variable gegen fixe Zahlungen zu tauschen, also eine Call-Option auf eine Anleihe mit fester Verzinsung. Der Strike-Preis entspricht dabei dem Nominalwert des Swaps. Die Kuponzahlungen der Anleihe betragen, sofern halbjährliche Kupons fällig sind, die Hälfte der Swap-Rate.

Für die Bewertung der Swaption wird nun der Short-Rate Binomialbaum gebildet. Basierend auf diesen Baum wird der Wert der Kuponanleihe $B_{i,j}$ für jeden Knoten berechnet. Die Laufzeit der Option beträgt dabei T, die der Anleihe s ($T \leq s$). Die Zeitschritte bis zur Fälligkeit der Option beziehungsweise der Anleihe werden mit N_s ($s = N_s\Delta t$) beziehungsweise N_T ($T = N_T\Delta t$) bezeichnet. Zum Zeitpunkt der Fälligkeit der Anleihe N_s ergibt sich der Wert mit

$$B_{N_s,j} = 1 + \frac{\text{Kupon}}{2}. \tag{5.43}$$

Mittels Rückwertsinduktion können die Werte in den übrigen Knoten des Baums über

$$B_{i,j} = \frac{1}{2}\left(B_{i+1,j+1} + B_{i+1,j+1}\right) \tag{5.44}$$

berechnet werden.

Mit Hilfe der Zustandspreise $Q_{N_T,j}$ kann nun der Wert einer europäischen Swaption berechnet werden. Für eine Payer-Swaption ergibt sich der Wert mit

$$\sum_j Q_{N_T,j} max\{0, 1 - B_{N_T,j}\}, \tag{5.45}$$

für eine Receiver-Swaption mit

$$\sum_j Q_{N_T,j} max\{0, B_{N_T,j} - 1\}. \tag{5.46}$$

5.4 Modellieren der Short-Rate mittels Trinomialbäumen

Im Abschnitt 5.3 wurden Binomialbäume zur Modellierung der Short-Rate verwendet. Im Gegensatz zu Binomialbäumen bieten Trinomialbäume einen zusätzlichen Freiheitsgrad. Es wird damit auch die Modellierung von Modellen mit Mean Reversion, wie es das Hull-White Modell ist, möglich. Die Ausführungen stützen sich, wenn nicht explizit angegeben, wieder auf (Clewlow und Strickland, 1998, S.255 ff).

Für die Trinomialbäume wird der Zeitschritt wieder mit Δt und der Schritt im Zinssatz mit Δr festgelegt. Die Anpassung der Änderung der Short-Rate an den richtigen Mittelwert und an die Standardabweichung erfolgt dabei über die Wahrscheinlichkeiten. In Abbildung 5.3 ist ein Trinomialbaum abgebildet. Ein Knoten (i, j) im Baum wird durch $t = i\Delta t$ und $r_{i,j} = r_{0,0} + j\Delta r$ charakterisiert. Um die Konvergenz und Stabilität der Trinomialbaummethode zu gewährleisten, muss ein passendes Verhältnis zwischen Δr und Δt gewählt werden. Dieses wird mit

$$\Delta r = \sigma\sqrt{3\Delta t} \tag{5.47}$$

festgelegt.

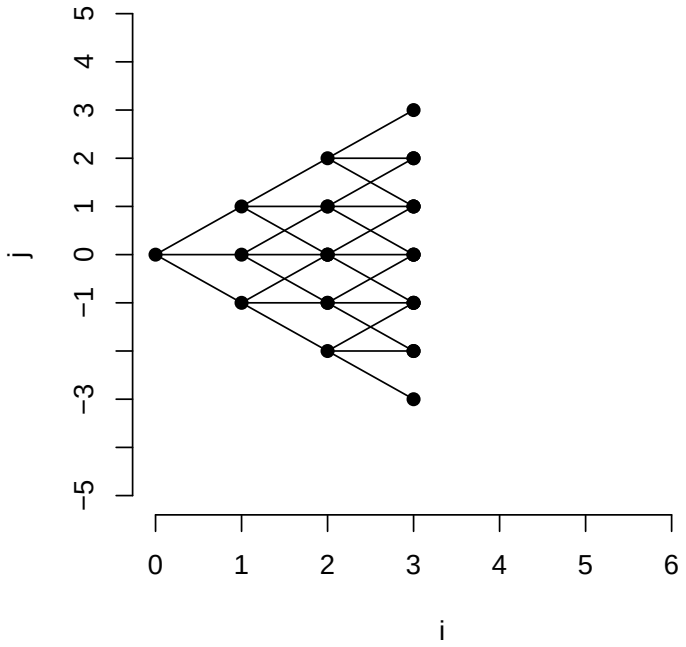

Abbildung 5.3: Trinomialbaum

In einem Trinomialbaum können die Verzweigungen etwas allgemeiner gestaltet sein als in Abbildung 5.3 dargestellt. In Abbildung 5.4, 5.5 und 5.6 sind drei alternative Verzweigungsmechanismen skizziert.

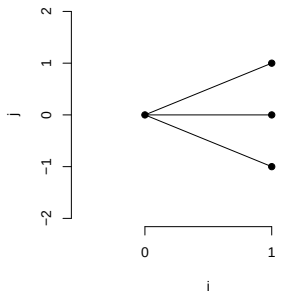

Abbildung 5.4: normal

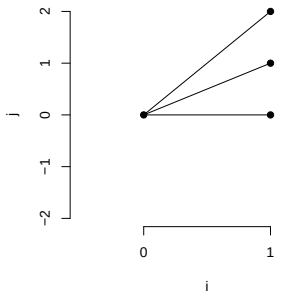

Abbildung 5.5: aufwärts

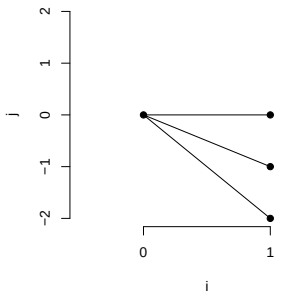

Abbildung 5.6: abwärts

Speziell die Verzweigungsmethoden, die in Abbildung 5.5 und 5.6 dargestellt sind, erweisen sich für die Modellierung der Mean Reversion als nützlich. Der normale Verzweigungsmechanismus (Abbildung 5.4) ermöglicht die Veränderung der Short-Rate im Zeitintervall Δt um $\pm\Delta r$ beziehungsweise ein Gleichbleiben. Der Aufwärtsverzweigungsmechanismus (Abbildung 5.5) führt im Falle einer niedrigen Short-Rate zu einer Veränderung um $+2\Delta r$ oder $+\Delta r$ beziehungsweise zu einem Gleichbleiben. Ein Abwärtsverzweigungsmechanismus (Abbildung 5.6) kommt bei einer hohen Short-Rate zur Anwendung und ermöglicht eine Reduktion um $-2\Delta r$ oder $-\Delta r$ beziehungsweise ein Gleichbleiben. In weiterer Folge werden diese drei Verzweigungsmechanismen für die Erhöhung der Flexibilität des Trinomialbaums zur Anwendung kommen.

Die Short-Rate wird in diesem Abschnitt mit Hilfe von Trinomialbäumen und dem Modell von Hull-White modelliert. Obwohl beim Hull-White Modell teilweise analytische Formeln für die Bewertung von Zinsderivaten existieren (vgl. Abschnitt 5.2.2), muss bei komplizierteren, etwa pfadabhängigen, Derivaten auf numerische Bewertungsverfahren zurückgegriffen werden.

5.4.1 Anpassung des Hull-White Modells an die Zinsstruktur

Hull und White (1994) haben ein robustes zweistufiges Verfahren zur Konstruktion von Trinomialbäumen vorgeschlagen, welches in diesem Abschnitt für das Einfaktor Hull-White Modell zur Anwendung kommt. Der Trinomialbaum wird dabei zur Approximation des Prozesses verwendet, der durch folgende stochastische Differentialgleichung gegeben ist:

$$dr = (\theta(t) - \alpha r)\, dt + \sigma dz. \tag{5.48}$$

Als erster Schritt wird unter der Verwendung von konstanten Zeitschritten und Zinssatzschritten ein Trinomialbaum für die Short-Rate, die dem vereinfachten Prozess ($\theta(t) = 0$ und $r_{0,0} = 0$)

$$dr = -\alpha r dt + \sigma dz \tag{5.49}$$

folgt, konstruiert. Der Prozess ist symmetrisch um $r = 0$ und $dr(t) = (r(t + \Delta t) - r(t)) \sim N(-\alpha r(t)\Delta t, \sigma^2 \Delta t)$ (unter der Annahme, dass Terme höherer Ordnung als Δt nicht berücksichtigt werden). Abbildung 5.3 zeigt solch einen symmetrischen Baum nach drei Zeitschritten.

Um nun den Trinomialbaum bilden zu können, müssen die Verzweigungen in jedem Knoten bestimmt werden. Die Art der Verzweigung ist abhängig vom Vorzeichen und Größe von α und von j. Mit der Art der Verzweigung wird auch die Wahrscheinlichkeit bestimmt, die wiederum maßgebend für die Anpassung an den Erwartungswert und die Varianz der Veränderung der Short-Rate des vereinfachten Prozesses ist. Normalerweise ist der normale Verzweigungsmechanismus (Abbildung 5.4) passend, aber speziell wenn der Mean-Reversion Parameter $\alpha \neq 0$, muss auf die anderen Verzweigungsmechanismen zurück gegriffen werden.

j_{max} wird als jener Wert von j definiert, bei dem die Art der Verzweigung von normal auf aufwärts gewechselt wird (d.h. von Abbildung 5.4 zu 5.5). Bei j_{min} wird von normal auf abwärts (d.h. von Abbildung 5.4 zu 5.6) gewechselt. Hull und White (1994) zeigen, dass wenn

$$j_{max} = \left\lceil \frac{0.1835}{\alpha \Delta t} \right\rceil \tag{5.50}$$

und

$$j_{min} = -\left\lfloor \frac{0.1835}{\alpha \Delta t} \right\rfloor, \tag{5.51}$$

d.h.

$$j_{min} = -j_{max},$$

die Wahrscheinlichkeiten einer Auf-, Abwärts- und horizontalen Verzweigung immer positiv sind. Über die Beziehungen, dass sich die Wahrscheinlichkeiten in einem Knoten auf eins summieren und der Mittelwert und die Varianz der Änderung der Short-Rate nachgebildet werden soll, können je nach Verzweigungsart die Wahrscheinlichkeiten berechnet werden. Die Wahrscheinlichkeiten für einen Knoten (i, j) der Verzweigungsart gemäß Abbildung 5.4 sind durch

$$\begin{aligned} p_u &= \frac{1}{6} + \frac{1}{2}(\alpha^2 j^2 \Delta t^2 - \alpha j \Delta t) \\ p_m &= \frac{2}{3} - \alpha^2 j^2 \Delta t^2 \\ p_d &= \frac{1}{6} + \frac{1}{2}(\alpha^2 j^2 \Delta t^2 + \alpha j \Delta t) \end{aligned}$$

gegeben.

Für die Verzweigungsart gemäß Abbildung 5.5 ergeben sich die Wahrscheinlichkeiten mit

$$\begin{aligned} p_u &= \frac{1}{6} + \frac{1}{2}(\alpha^2 j^2 \Delta t^2 + \alpha j \Delta t) \\ p_m &= -\frac{1}{3} - \alpha^2 j^2 \Delta t^2 - 2\alpha j \Delta t \\ p_d &= \frac{7}{6} + \frac{1}{2}(\alpha^2 j^2 \Delta t^2 + 3\alpha j \Delta t), \end{aligned}$$

und für die Verzweigungsart der Abbildung 5.6 mit

$$\begin{aligned} p_u &= \frac{7}{6} + \frac{1}{2}(\alpha^2 j^2 \Delta t^2 - 3\alpha j \Delta t) \\ p_m &= -\frac{1}{3} - \alpha^2 j^2 \Delta t^2 + 2\alpha j \Delta t \\ p_d &= \frac{1}{6} + \frac{1}{2}(\alpha^2 j^2 \Delta t^2 - \alpha j \Delta t). \end{aligned}$$

Zu bemerken ist, dass die Wahrscheinlichkeiten nur von j abhängen, da Δt und α konstant sind.

Der im ersten Schritt konstruierte Baum ist auf keinen Fall konsistent mit der beobachteten Zinsstruktur. Deshalb wird ein zeitabhängiger Driftparameter a_i definiert, der die Knoten des Baumes so verschiebt, dass die beobachtete Zinsstruktur repliziert wird. Der so konstruierte Baum repräsentiert den Prozess, der durch Gleichung 5.48 definiert ist. Die Werte der Short-Rate r in einem Knoten (i, j) ergeben sich als die korrespondierenden Werte vom Baum des vereinfachten Prozesses plus dem Driftparameter a_i. Der Parameter a_i hat dabei die Form

$$a_i = \frac{ln \sum_j Q_{i,j} e^{-j\Delta r \Delta t} - lnP(i+1)}{\Delta t} \tag{5.52}$$

und wird so gewählt, dass a_i den korrekten Wert einer in $(i+1)\Delta t$ fälligen Nullkupon-Anleihe liefert. Für die Berechnung von a_i müssen alle Zustandspreise $Q_{i,j}$ bis zum Zeitpunkt i berechnet worden sein.

$Q_{i,j}$ bezeichnet wiederum den Wert eines Wertpapiers, das bei Erreichen des Knotens (i, j) eine Geldeinheit und sonst 0 ausbezahlt. Die Berechnung des Driftparameter a_i und der Zustandspreise $Q_{i,j}$ erfolgt mit der Methode der Vorwärtsinduktion.

Die Berechnung der Zustandspreise $Q_{i,j}$ für einen Knoten basiert dabei auf den Zustandspreisen der Vorgängerknoten j', die sich je nach Verzweigungsprozess unterscheiden:

$$Q_{i,j} = \sum_{j'} Q_{i-1,j'} p_{j',j} d_{i-1,j'}, \tag{5.53}$$

wobei der Diskontfaktor für einen Knoten (i, j) mit $d_{i,j} = e^{-r_{i,j}\Delta t}$ gegeben ist.

Zusammenfassend soll nochmals schrittweise erläutert werden wie der Short-Rate Trinomialbaum berechnet wird.

Schritt 1:

Berechnen der Short-Rate für alle Knoten (i, j) im Baum gemäß dem vereinfachten Prozess: $r_{i,j} = j\Delta r$, wobei $r_{0,0} = 0$.

Schritt 2:

Bestimmen der Verzweigungsstruktur des Baumes und der dazugehörigen Wahrscheinlichkeiten in Abhängigkeit von j_{max}.

Schritt 3:

Bestimmen der Zustandspreise in Abhängigkeit vom Verzweigungsprozess.

$$Q_{i,j} = \sum_{j'} Q_{i-1,j'} p_{j',j} d_{i-1,j'}.$$

Schritt 4:

Berechnen des Driftparamters über

$$a_i = \frac{ln \sum_j Q_{i,j} e^{-j\Delta r \Delta t} - lnP(i+1)}{\Delta t}.$$

Schritt 5:

Berechnen der Short-Rate und der Diskontfaktoren unter Berücksichtigung der um a_i angepassten Short-Rate.

5.4.2 Bewertung von Zinsderivaten mittels Trinomialbäumen

Analog zu Punkt 5.3.3 soll eine Option auf eine Nullkupon-Anleihe bewertet werden. Die Laufzeit der Option beträgt dabei T, die der Anleihe s ($T \leq s$). Die Zeitschritte bis zur Fälligkeit der Option beziehungsweise der Anleihe werden wieder mit N_s ($s = N_s \Delta t$) beziehungsweise N_T ($T = N_T \Delta t$) bezeichnet. Für die Bewertung muss der Short-Rate Baum bis zum Zeitpunkt N_s berechnet worden sein. Der Wert der Nullkupon-Anleihe mit einer Fälligkeit s im Knoten (i, j) wird mit $Ps_{i,j}$ bezeichnet.

Als erster Schritt bei der Bewertung wird der Wert der Nullkupon-Anleihe für jeden Knoten (i, j) im Baum berechnet. $Ps_{N_s,j}$ ist definitionsgemäß 1. Ausgehend von diesen Werten werden mittels Rückwertsinduktion die übrigen Werte über die Beziehung

$$Ps_{i,j} = d_{i,j} \left(Ps_{i+1,u}\ p_{u,i,j} + Ps_{i+1,m}\ p_{m,i,j} + Ps_{i+1,d}\ p_{d,i,j} \right), \tag{5.54}$$

die für alle Knoten j zum Zeitpunkt i gilt, berechnet. $d_{i,j}$ ergibt sich dabei aus dem zuvor berechneten Short-Rate Baum. Betrachtet vom Knoten (i, j) werden die Werte von Ps in den drei Nachfolgeknoten mit den Wahrscheinlichkeiten, die der Verzweigungsmethode entsprechen, multipliziert und anschließend aufsummiert und abdiskontiert.

Der nächste Schritt besteht aus der Berechnung des Payoffs der Option zum Zeitpunkt der Fälligkeit. Je nach Auszahlungsfunktion ergibt sich etwa der Wert einer Call-Option mit Strike-Preis X für alle Knoten j zum Zeitpunkt i mit

$$C_{N_T,j} = max\{0, Ps_{N_T,j} - X\}. \tag{5.55}$$

Für eine Put-Option gilt:

$$P_{N_T,j} = max\{0, X - Ps_{N_T,j}\}. \tag{5.56}$$

Ausgehend von den Optionswerten der Knoten zum Zeitpunkt N_T kann nun der Wert einer europäischen Option für alle übrigen Knoten bis zum Ausgangsknoten $(0, 0)$ mittels

$$C_{i,j} = d_{i,j} \left(C_{i+1,u}\ p_{u,i,j} + C_{i+1,m}\ p_{m,i,j} + C_{i+1,d}\ p_{d,i,j} \right) \tag{5.57}$$

berechnet werden. Allerdings kann die Berechnung durch die Tatsache, dass die Zustandspreise den diskontierten Wahrscheinlichkeiten entsprechen, vereinfacht werden. Der Wert ergibt sich im Falle einer Call-Option mit

$$C_{0,0} = \sum_j Q_{N_T,j} max\{0, Ps_{N_T,j} - X\}. \tag{5.58}$$

Für eine Put-Option ist die Auszahlungsfunktion entsprechend anzupassen.

Für die Bewertung einer amerikanischen Option wird der Wert der Option in jedem Knoten als das Maximum aus dem intrinsischen Werte und dem diskontierten Erwartungswert berechnet. Für eine amerikanische Call-Option ergibt sich der Wert für alle Knoten j zum Zeitpunkt i mit

$$C_{i,j} = max\left\{Ps_{i,j} - X,\ d_{i,j}\left(C_{i+1,u}\ p_{u,i,j} + C_{i+1,m}\ p_{m,i,j} + C_{i+1,d}\ p_{d,i,j}\right)\right\}. \tag{5.59}$$

Der Wert der amerikanischen Option zum Zeitpunkt 0 ist dann durch den Anfangsknoten $(0, 0)$ gegeben.

5.5 Bewertung von Zinsderivaten mit dem R Package shortrate

Für die in den Abschnitten 5.3 und 5.4 vorgestellten numerischen Verfahren zur Berechnung der Short-Rate und zur Bewertung von Zinsderivaten wurden entsprechende Funktionen in `R` implementiert. Die Funktionen wurden kompakt zu dem Package `shortrate` zusammengefasst. Der Quellcode des gesamten Packages wurde im Rahmen dieser Diplomarbeit in Zusammenarbeit mit Mag.Robert Ferstl entwickelt und ist im Anhang B zu finden.

In diesem Abschnitt soll das Package mit seinen Funktionen vorgestellt und die Handhabung erläutert werden. Abgeschlossen wird dieser Abschnitt mit einer Präsentation von Bewertungen von Zinsderivaten, die mit dem Package `shortrate` durchgeführt werden. Die Beispiele wurden dabei dem Buch von Clewlow und Strickland (1998) entnommen.

5.5.1 Informationen über das Package shortrate

Für die Implementierung der Funktionen wurde wie beim Package `termstrc` ein objektorientierter Ansatz gewählt. Bevor jedoch im Detail auf Methoden des Packages eingegangen wird, sollen die Funktionen des Packages vorgestellt und erläutert werden.

Für die Berechnung der Short-Rate stehen die Modelle von Black-Derman-Toy, kalibriert auf die Zinsstruktur beziehungsweise auf die Zins- und Volatilitätsstruktur und das Modell von Hull-White, kalibriert auf die Zinsstruktur, zur Verfügung. Tabelle 5.1 gibt einen Überblick über die im Package verfügbaren Funktionen.

Als S3 Klassen wurden dabei `bdt_y`,`bdt_yv`,`hw`,`pd_prices`,`b_prices`,`EbonOpt`,`AbondOpt`, `binomialtree`,`trinomialtree` und `Eswaption` definiert. Für diese Klassen stehen eigene `plot`, `print` und `summary` Methoden zur Verfügung.

Funktion	Erklärung
`bdt_y(R,sigR)`	BDT Modell auf die Zinsstruktur kalibriert
`bdt_yv(R,sigR`	BDT Modell auf die Zins -und Volatilitätsstruktur kalibriert
`hw(R,alpha,sigma)`	HW Modell auf die Zinsstruktur kalibriert
`pd_prices.bdt_y(bdt_y)`	Bewertung von Nullkupon-Anleihen basierend auf dem BDT Modell
`pd_prices.hw(hw)`	Bewertung von Nullkupon-Anleihen basierend auf dem HW Modell
`b_prices.bdt_y(bdt_y)`	Bewertung von Kupon-Anleihen basierend auf dem BDT Modell
`EbondOpt.bdt_y(bdt_y)`	Bewertung von europ. Bond-Optionen basierend auf dem BDT Modell
`EbondOpt.hw(hw)`	Bewertung von europ. Bond-Optionen basierend auf dem HW Modell
`AbondOpt.bdt_y(bdt_y)`	Bewertung von amerik. Bond-Optionen basierend auf dem BDT Modell
`AbondOpt.hw(hw)`	Bewertung von amerik. Bond-Optionen basierend auf dem HW Modell
`Eswaptiont.bdt_y(bdt_y)`	Bewertung von europ. Swaptions basierend auf dem BDT Modell

Tabelle 5.1: Funktionen des Packages shortrate

Stellvertretend für die Funktionalität des Packages sollen die Funktion `bdt_yv` und die dazugehörigen Methoden anhand eines Beispiels vorgestellt werden. Das Beispiel wurde aus dem Buch von Clewlow entnommen (vgl. Clewlow und Strickland, 1998, S. 243 f).

Es soll der Short-Rate Baum für eine flache Zinsstruktur von 5% mit Hilfe des Black-Derman-Toy Modells für vier Jahre bestimmt werden. Die Volatilität nimmt schrittweise von 10% auf 7% ab.

Der Funktionsaufruf sieht in `R` wie folgt aus:

```
#load package shortrate
library(shortrate)
#term structure
R <- rep(0.05,5)
#volatility structure
sigR <- c(0.1,0.09,0.08,0.07)

myres <- bdt_yv(R,sigR)
```

Über die `summary` Methode lassen sich die kalibrierte Volatilität und Konvergenzinformationen der Optimierungsfunktion ausgeben:

```
summary(myres)

------------------------------------------------
Black-Derman-Toy Model
fitted to Yield and Volatility Data:
------------------------------------------------
```

```
Convergence info from nlm():
-----------------------------------------------------
  objective               convergence message                 iterations
1 "1.01154162823747e-09" "0"         "X-convergence (3)"     "19"
2 "0.448518292407742"    "1"         "false convergence (8)" "19"
3 "0.213023395878015"    "1"         "false convergence (8)" "42"

U ... :
-----------------------------------------------------
         0          1          2          3
0.05000000 0.04979976 0.05014759 0.05073057

sig ... calibrated volatility parameter:
-----------------------------------------------------
         0          1          2          3
0.10000000 0.09220753 0.07998385 0.06977951

Pu ... price of bond in up movement:
-----------------------------------------------------
        2         3         4
0.9482178 0.8999871 0.8549788

Pd ... price of bond in down movement:
-----------------------------------------------------
        2         3         4
0.9565596 0.9141190 0.8726595
```

Die berechneten Bäume lassen sich sowohl über die `print` als auch `plot` Methode ausgeben. An dieser Stelle soll nur die grafische Repräsentation der Bäume präsentiert werden. Die Plots können über

```
#plot short rate tree
plot(myres,1)
#plot discount factor tree
plot(myres,2)
#plot state price trees
plot(myres,3,dy=-0.3,dx=-0.05)
plot(myres,4,dy=0.5,dx=-0.1)
```

generiert werden.

In Abbildung 5.7 ist der Baum mit den Short-Rates und Diskontfaktoren dargestellt. Abbildung 5.8 enthält die Bäume der Zustandspreise. Ein Vergleich mit den Ergebnissen aus

Clewlow und Strickland (1998) zeigt eine gute Übereinstimmung. Komplett identische Ergebnisse konnten allerdings nicht erreicht werden, da die numerische Optimierung sehr sensitiv auf Startwerte ist und eine Konvergenz der Optimierung nicht zwingend erreicht werden kann (vgl. London, 2004, S. 480).

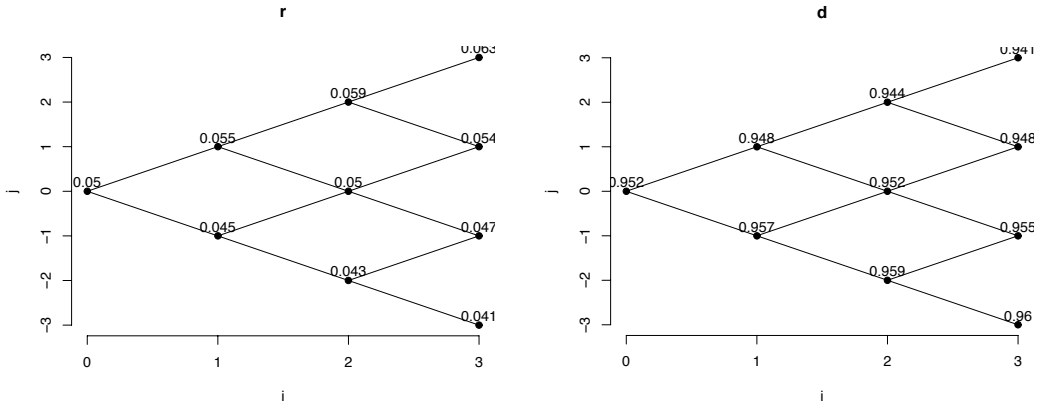

Abbildung 5.7: Short-Rate und Diskontfaktoren Binomialbaum

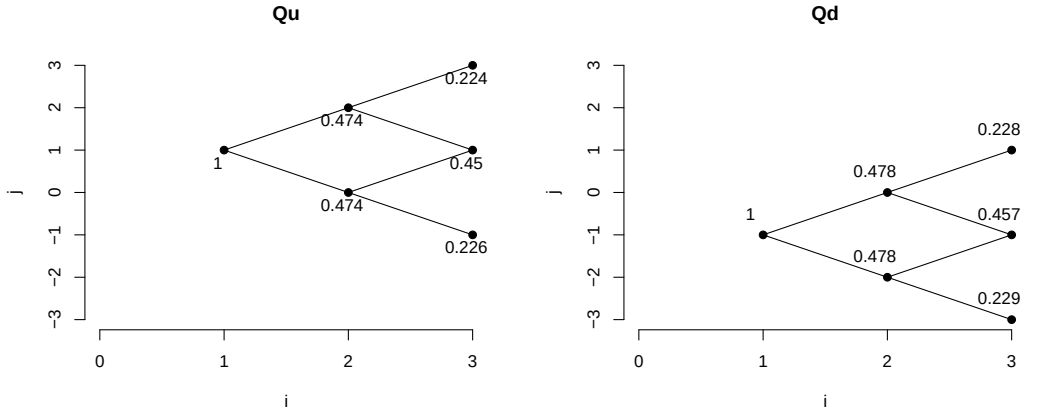

Abbildung 5.8: Binomialbaum der Zustandspreise

5.5.2 Bewertung von amerikanischen Optionen auf Anleihen

In diesem Abschnitt sollen zwei weitere Beispiele aus Clewlow und Strickland (1998) mit Funktionen aus dem Package `shortrate` gelöst werden. Für die Bestimmung der Short-Rate kommen dabei die Modelle von Black-Derman-Toy und Hull-White, kalibriert jeweils auf die Zinsstruktur, zum Einsatz.

Bewertung von amerikanischen Optionen auf Anleihen unter Verwendung des BDT Modells

Es soll eine amerikanische Put-Option auf Nullkupon-Anleihen bewertet werden. Die für die Berechnung notwendigen Diskontfaktoren beziehungsweise Zustandspreise werden über das BDT Modell, welches auf die Zinsstruktur kalibriert wurde, bezogen. Die Laufzeit der Anleihe soll 10 Jahre betragen, die Zinsstruktur wird als flach mit einer jährlichen Rendite von 5% angenommen. Die Volatilität beträgt konstant 10%. Die Put Option hat eine Laufzeit von 6 Jahren und einen Ausübungspreis von 0.80 (Das Beispiel ist in Clewlow und Strickland (1998) auf Seite 248 zu finden).

Als erster Schritt wird das Short-Rate Modell berechnet. Dieses liefert die für die weitere Berechnung notwendigen Bäume der Diskontfaktoren und Zustandspreise. Die Bäume sind in Abbildung 5.9 dargestellt.

```
#term structure
R <- rep(0.05,10)
#constant volatility
sigR <- 0.1
myres <- bdt_y(rep,sigR)
#plot discount factors and state price tree
plot.binomialtree(myres$d)
plot.binomialtree(myres$Q)
```

Da die `plot` Methode nur für die Short-Rate und Diskontfaktoren Bäume festgelegt wurde, muss für das Plotten der Zustandspreisbäume auf die zugrunde liegende Funktion `plot.binomialtree` zurück gegriffen werden.

Der nächste Schritt besteht in der Berechnung der Preise der Nullkupon-Anleihe. Diese Berechnung wird eigentlich implizit bei der Berechnung des Optionspreises durchgeführt. Aus Gründen der Nachvollziehbarkeit und um eine Vergleichbarkeit der Ergebnisse mit den Beispiel aus Clewlow und Strickland (1998) zu ermöglichen, ist der entsprechende Baum in Abbildung 5.10 dargestellt. Der Plot wird über den Befehl

```
mypd_prices <- pd_prices.bdt_y(myres)
plot(mypd_prices,dy=0.5)
```

generiert.

Mit Hilfe der Nullkupon-Anleihen Preise für jeden Knoten des Baums kann nun die amerikanische Put-Option bewertet werden. Die notwendigen Funktionsaufrufe sind

```
#time to maturity of the option
To=6
#strike price
X=0.8
myAbondOpt<-AbondOpt.bdt_y(myres,"put",To,X)
#plot american option price tree
plot(myAbondOpt,dy=0.3)
```

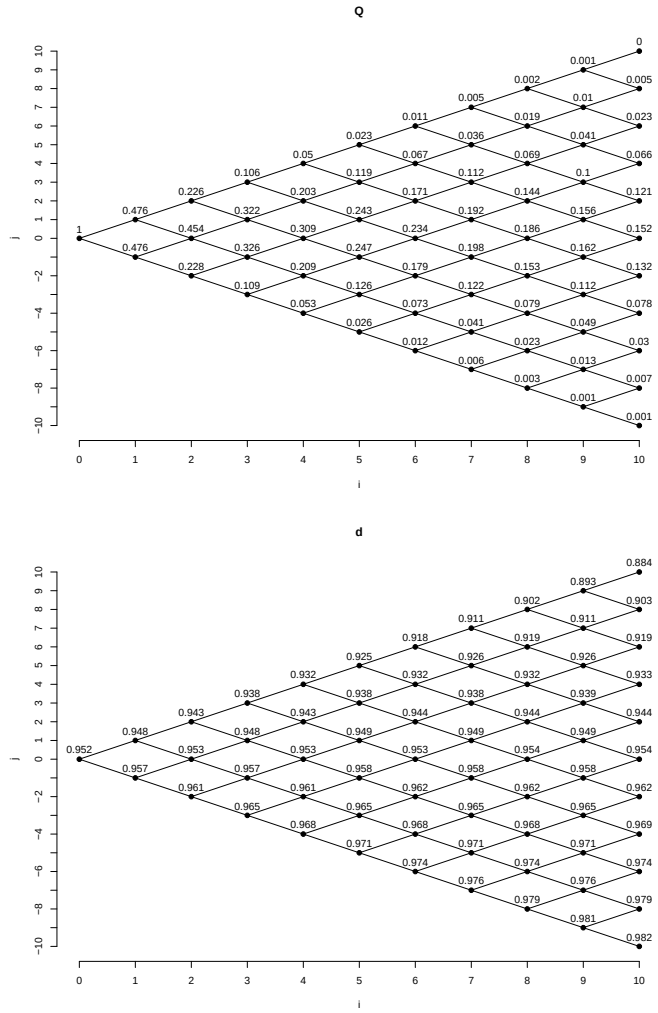

Abbildung 5.9: Binomialbaum der Zustandspreise und Diskontfaktoren

Wie Abbildung 5.11 zeigt, ergibt sich der Optionspreis mit 0.185 und entspricht damit ziemlich gut dem Ergebnis von Clewlow und Strickland (1998) mit 0.1861. Die Unterschied lässt sich wieder auf die verwendeten Optimierungsfunktionen und deren Konvergenz zurückführen.

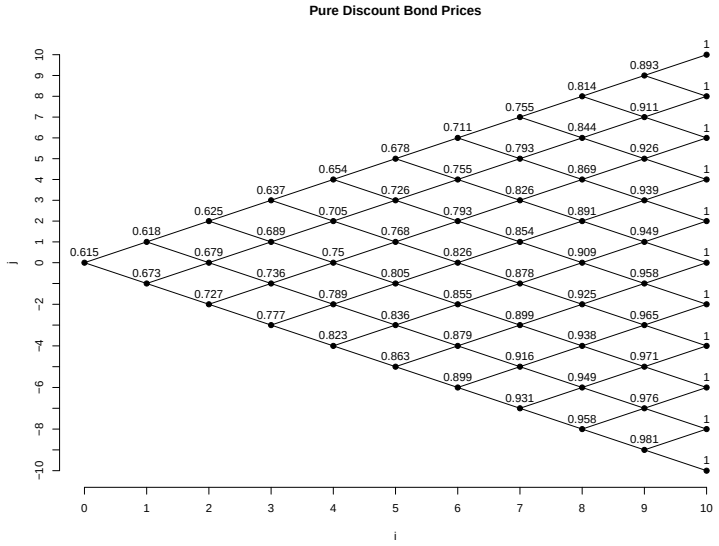

Abbildung 5.10: Baum der Nullkupon-Anleihen Preise

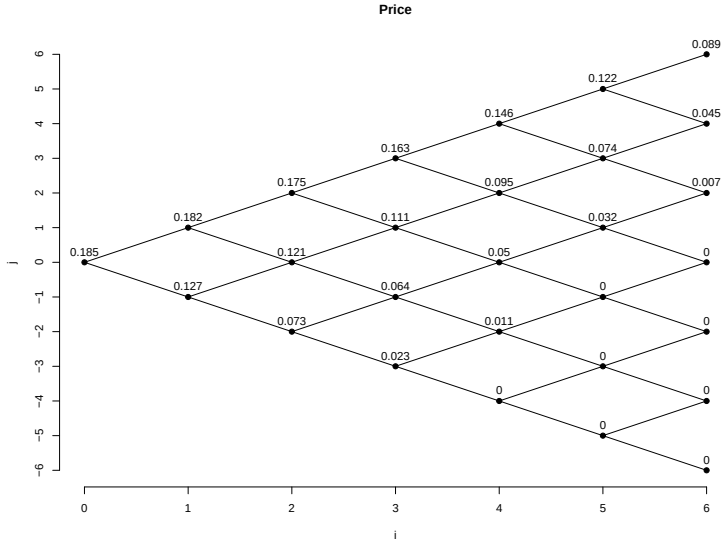

Abbildung 5.11: Binomialbaum mit Werten der amerikanischen Option

Bewertung von amerikanischen Optionen auf Anleihen unter Verwendung des HW Modells

In diesem Beispiel soll wieder eine amerikanische Option bewertet werden. Allerdings werden die Zustandspreise und die Diskontfaktoren über das HW-Modell, das auf die Zinsstruktur kalibriert wurde, berechnet. Dementsprechend wird die Bewertung der Option mit Hilfe von Trinomialbäumen erfolgen.

Die Laufzeit der Anleihe soll 5 Jahre betragen, die Zinsstruktur wird als steigend angenommen (von 5% auf 7%). Die Volatilität beträgt konstant 10% und der Parameter α des HW Modells wird mit 0.1 angenommen. Die Option hat eine Laufzeit von 3 Jahren und einen Ausübungspreis von 0.85 (Das Beispiel ist in Clewlow und Strickland (1998) auf Seite 287 zu finden).

Die Berechnungschritte werden bei diesem Beispiel zusammengefasst und nur der Baum mit den errechneten Optionswerten wird präsentiert. Die notwendigen Funktionsaufrufe sehen in R wie folgt aus:

```
#load package shortrate
library(shortrate)
#term structure
R<-c(0.05,0.0575,0.0625,0.0675,0.07,0.0725)

##short rate parameters
alpha <- 0.1
#constant volatility
sigma <- 0.01

#calculate short rate trees
myres <- hw(R,alpha,sigma)

#maturity of the option
To <- 3
#strike price
X<- 0.85
type="put"

myAbondOpt <- AbondOpt.hw(myhw,type,To,X)
#plot price tree
plot(myAbondOpt,col_u="red",col_n="green",col_d="blue")
```

Der Baum mit den Ergebnissen der Berechnungen ist in Abbildung 5.12 dargestellt. Der Preis für eine amerikanische Put Option beträgt damit 0.415 und entspricht dem Ergebnis von Clewlow und Strickland (1998).

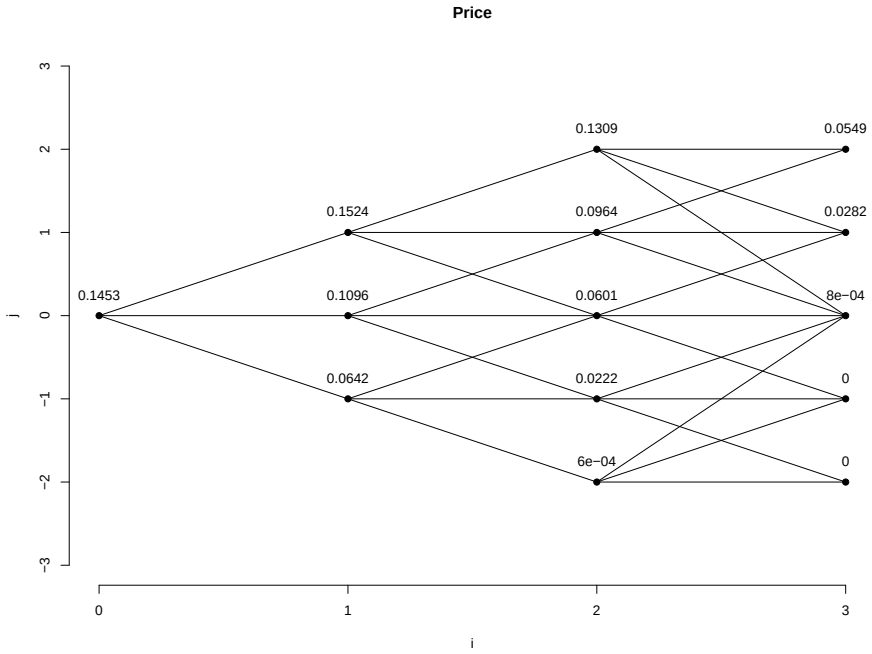

Abbildung 5.12: Trinomialbaum mit Werten der amerikanischen Option

5.5.3 Bewertung von europäischen Swaptions

Die Bewertung von europäischen Swaptions mit Hilfe der Funktionen des Packages `shortrate` wird anhand eines Beispiels von Clewlow und Strickland (1998) gezeigt. Das Beispiel ist auf Seite 251 zu finden. Es soll eine europäische Payer-Swaption mit einer Laufzeit von zwei Jahren bewertet werden. Das Underlying ist dabei ein dreijähriger Zinsswap. Die Zinsstruktur wird als flach mit einer jährlichen Rendite von 5% angenommen. Die Volalitiltät der Short-Rate soll konstant 10% betragen. Die Swap-Rate des Zinsswaps beträgt 4.5%.

Als erster Schritt werden die Bäume für die Diskontfaktoren und die Zustandspreise berechnet. Zur Anwendung kommt in diesem Beispiel wieder das ‚auf die Zinsstruktur kalibrierte, Black-Derman-Toy Modell.

Die Funktionsaufrufe sehen dabei wie folgt aus

```
#term structure
R <- rep (0.05,7)
#constant volatility
sigR <- 0.1

myres <- bdt_y(R,sigR)

#time to maturity  of the option
To <- 2
```

```
#time to maturity of the swap
Ts <- 3
#swap rate = coupon of the bond
Sr <- 4.5/100
#principal of the swap =1
```

In Abbildung 5.13 ist der Zustandspreis- und Diskontfaktorenbinomialbaum dargestellt. Auf Basis dieser Bäume errechnet sich über den Funktionsaufruf

```
#calculate bdt short rate model
myres <- bdt_y(R,sigR)
#plot short-rate and discount factor tree
plot.binomialtree(myres$Q,dy=0.3,main="Q")
plot.binomialtree(myres$d,dy=0.34,main="d")
#calculate bond prices
myb_prices <- b_prices.bdt_y(myres,Sr)
#plot bond price tree
plot(myb_prices,dy=0.3)
```

der Anleihenpreisbaum, der in Abbildung 5.14 dargestellt ist.

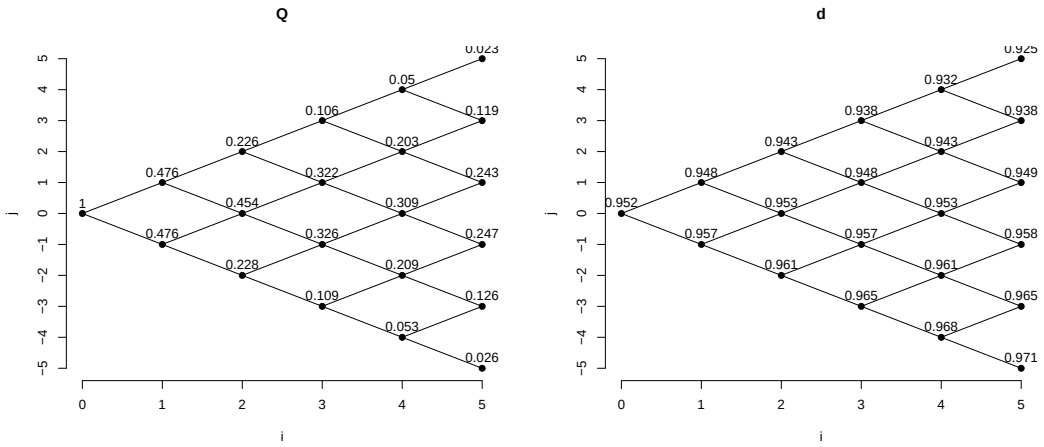

Abbildung 5.13: Binomialbaum der Zustandspreise und Diskontfaktoren

Der Optionspreis für die europäische Payer-Swaption errechnet sich mit Hilfe des Funktionsaufrufs

```
myEswaption <- Eswaption.bdt_y(myres,Sr,"payer",To,X=1)
summary(myEswaption)
```

Der Aufruf der `summary` Methode für das Objekt `myEswaption` gibt den Preis der Option, der dem errechneten Wert von Clewlow und Strickland (1998) entspricht.

```
 ----------------------------------------------------
European Swaption Price:
----------------------------------------------------
[1] 0.0324388

cr ... Swap rate:
----------------------------------------------------
[1] 0.045

Type:
----------------------------------------------------
[1] "payer"

Time to maturity:
----------------------------------------------------
[1] 2

Strike Price = swap principal:
----------------------------------------------------
[1] 1
```

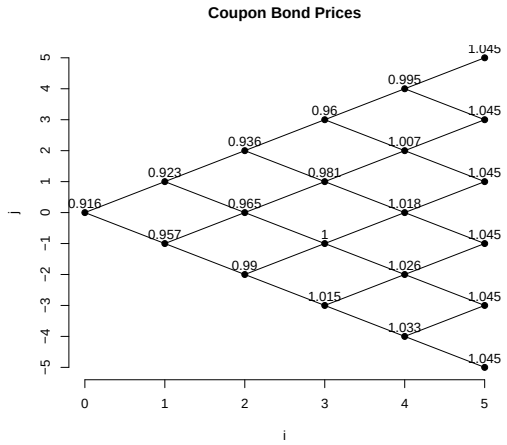

Abbildung 5.14: Binomialbaum der Anleihenpreise

Schlussbetrachtung

Im Teil I der vorliegenden Arbeit wurde ein ausführlicher Überblick über die relevanten `R` Packages für die Lösung von Problemstellungen des Financial Engineerings präsentiert. Im Detail wurde dabei auf die umfangreiche Erweiterung `Rmetrics` eingegangen. `Rmetrics` stellt innerhalb der `R` Packagelandschaft sicherlich einen einzigartigen Ansatz dar. In vielen Bereichen gibt es jedoch große Überschneidungen mit anderen Packages. In Kapitel 1 wurden diese aufgezeigt und Alternativpackages präsentiert. Die Stärke von `Rmetrics` liegt eher in der Möglichkeit mit Hilfe der Funktionen Fragestellungen der Lehre zu lösen. Funktionen für die Lösung von praktischen Fragestellungen wurde kaum berücksichtigt. `Rmetrics` bietet eine gute Möglichkeit einfache Finanzmodelle kennen zu lernen. Im Falle von komplexeren Problemstellungen scheint aber die Verwendung selbst implementierter Funktionen beziehungsweise Packages sinnvoll, im speziellen da die Kompatibilität von `Rmetrics` mit den Funktionen des `base` Packages teilweise nicht gewährleistet ist.

Im Teil II wurde der mangelnden Möglichkeit praktische Fragestellungen mit `Rmetrics` zu lösen Rechnung getragen. So wurden relevante Problemstellungen mit `R` gelöst. Für die Zinsstruktur- und Credit-Spread-Schätzung in Kapitel 3 wurde ein eigenes Package entwickelt. Das Package `termstrc` ist mittlerweile als Beta-Version unter `http://r-forge.wu-wien.ac.at/projects/termstrc/` mit einer vollständigen Dokumentation und zwei Datensätzen frei verfügbar. In Zukunft scheint eine Veröffentlichung als offizielles `R` Package auf `CRAN` realistisch und dies wird auch angestrebt.

Die Problemstellungen in Kapitel 4 wurden teilweise mit `Rmetrics` Funktionen gelöst. Allerdings mussten für die Lösung einiger Fragestellungen eigene Funktionen entwickelt werden. Wiederum wurde das Package `termstrc` verwendet um ein Teilproblem zu lösen. Dies zeigt sehr deutlich die guten Verwendungsmöglichkeiten des Packages.

Die Bewertung von Zinsderivaten im Kapitel 5 wurde mit Funktionen aus dem Package `shortrate` basierend auf Short-Rate Modellen durchgeführt. Das selbst entwickelte Package `shortrate` vereinigt Funktionen, die in der Form noch nicht in `R` existieren, auch gibt es keine entsprechenden Veröffentlichungen auf `CRAN`.

In dieser Arbeit wurde gezeigt wie `R` und Erweiterungs-Packages eingesetzt werden können

um gängige Finanzmodelle zu implementieren. Die im Rahmen dieser Arbeit entstandenen Packages wurden alle objektorientiert, mit den `R S3` Klassen, programmiert und können als Ergänzung in der `R` Packagelandschaft gesehen werden. Speziell, weil auf den Gebieten der Zinsstrukturschätzung und der Short-Rate Modelle noch keine bekannten, auf `CRAN` veröffentlichten, Packages existieren.

Zusätzlich wurde auch versucht durch die Entwicklung eigener Packages die Flexibilität und das Potential von `R` zu demonstrieren.

Es bleibt zu wünschen, dass durch diese Arbeit ein kleiner Beitrag in der Bekanntheitssteigerung von `R` geleistet wurde und `R` in Zukunft zunehmend für die Lösung von Aufgabenstellungen im Bereich des Financial Engineerings verwendet wird.

Danksagung

Ich möchte

Univ.-Prof. Dr. Gregor Dorfleitner für die umfangreiche Betreuung und wertvolle Tipps,

Mag. Robert Ferstl für die sehr gute Zusammenarbeit beim gemeinsamen Erstellen der Packages `termstrc` und `shortrate` und für zahlreiche Tipps

und all jenen, die mich während des Verfassens dieser Arbeit unterstützt haben,

danken.

Anhang A

Sourcecode - Zinsstruktur und Credit-Spread Schätzung

```
#######################################################################
#                          Spot Rates Nelson/Siegel                   #
#######################################################################

nelson_siegel <-
  function(beta, m, curve="spotrate") {

  switch(curve,
  spotrate = beta[1] + beta[2] * ((1-exp(-m/beta[4]))/(m/beta[4]))
             + beta[3]*(((1-exp(-m/beta[4]))/(m/beta[4]))
             -exp(-m/beta[4])),
  forwardrate = beta[1]+beta[2]*exp(-m/beta[4])+beta[3]*(m/beta[4])
                *exp(-m/beta[4]),
  creditspread = beta[1] + beta[2] * ((1-exp(-m/beta[4]))/(m/beta[4]))
             + beta[3]*exp(-m/beta[4])
 )
}

#######################################################################
#                          Spot Rates Svensson                        #
#######################################################################

svensson <-
  function(beta, m) {
  beta[1] + beta[2] * ((1 - exp(-m/beta[4]))/(m/beta[4])) +
  beta[3] * (((1 - exp(-m/beta[4]))/(m/beta[4])) - exp(-m/beta[4])) +
  beta[5] * (((1 - exp(-m/beta[6]))/(m/beta[6])) - exp(-m/beta[6]))}

#######################################################################
#                          loss function                              #
#######################################################################
```

```
loss_function <- function(p,phat,omega,weights) {
  if (weights=="none") omega <- rep(1,length(p))
  sum(omega*((p-phat)^2))}

#######################################################################
#                  Term structure estimation                          #
#######################################################################

# OPTIONS: - countries: vector with countries,
#                        i.e. c("AUSTRIA","GERMANY")

#                        first country is the reference country for

#                        credit spread estimation
#          - bonddata: dataset where countries are included
#          - maturity_spectrum: include only bonds within that range
#                               "all" ... use all available bonds
#          - method: "Nelson/Siegel", "Svensson"
#          - fit: "prices", "yields"
#          - weights: "none", "duration"
#          - startparam: matrix with startparameters; one row for one
#                        country; number of columns is 4 for the nelson/siegel
#                        and 6 for the svensson method
#          - control: a list with control parameters for the
#                        nlminb optimisation function

termstrc_estim <-
  function(countries,
           bonddata,
           maturity_spectrum="all",
           method="Nelson/Siegel",
           fit = "prices",
           weights="none",
           startparam,
           control=list(eval.max=1000)

           ) {

  # select given countries from bonddata
  bonddata <- bonddata[countries]

  if (length(maturity_spectrum=1)){bonddata <- bonddata}
  else{bonddata <- maturity_range(bonddata,maturity_spectrum[1],


  # number of countries
  n_countries <- length(bonddata)

  # create cashflows matrix
  cf <- lapply(bonddata,create_cashflows_matrix)

  # create maturities matrix
  m <- lapply(bonddata,create_maturities_matrix)
```

```
 # calculate dirty prices
 p <- mapply(function(i) bonddata[[i]]$PRICE +
              bonddata[[i]]$ACCRUED,1:n_countries)

 names(p) <- names(bonddata)


 cf_p <- mapply(function(i) create_cashflows_matrix(bonddata[[i]],
              include_price=TRUE),1:n_countries)
 names(cf_p) <- names(bonddata)

 m_p <- mapply(function(i) create_maturities_matrix(bonddata[[i]],
              include_price=TRUE),1:n_countries)
 names(m_p) <- names(bonddata)

 # calculate bond yields
 yields <- mapply(function(i) bond_yields(cf_p[[i]],m_p[[i]]),
          1:n_countries)
 names(yields) <- names(bonddata)

 #calculate duration
 duration <- mapply(function(i) duration(cf_p[[i]],m_p[[i]],
                         yields[[i]][,2]),1:n_countries)
 names(duration) <- names(bonddata)

 # SINGLE-CURVE ESTIMATION
 # objective function type

 obj_fct_prices <- function(b) {
    loss_function(p[[i]],
       bond_prices(method,b,m[[i]],cf[[i]])$bond_prices,
       duration[[i]][,3],weights)}

 obj_fct_yields <- function(b) {
   loss_function(yields[[i]][,"Yield"],bond_yields(rbind(
   -bond_prices(method,b,m[[i]],cf[[i]])$bond_prices,cf[[i]]),
       m_p[[i]]))}

  obj_fct <- switch(fit,
              "prices" = obj_fct_prices,
              "yields" = obj_fct_yields)

# bounds for single curve estimation
lower_bounds<-switch(method,
                     "Nelson/Siegel" = c(0, -Inf, -Inf, 0),
                     "Svensson" = c(0, -Inf, -Inf, 0, -Inf, 0))

upper_bounds<-switch(method,
                     "Nelson/Siegel" = rep(Inf, 4),
                     "Svensson" = rep(Inf, 6))

# single curve estimation
```

```
 #calculate optimal parameter
 opt_result <- list()
 for (i in 1:n_countries){
 opt_result[[i]] <- nlminb(startparam[i,],obj_fct,
        lower = lower_bounds, upper = upper_bounds,control=control)
 }
 names(opt_result) <- names(bonddata)

 #estimated prices

 estimated_prices <- mapply(function(i) bond_prices(method,
    opt_result[[i]]$par,m[[i]],cf[[i]])$bond_prices,1:n_countries)
 names(estimated_prices) <- names(bonddata)

 #calculate spotrates according to chosen approach

 spotrates <- mapply(function(i) srates(method,opt_result[[i]]$par,
                        yields[[i]][,1]),1:n_countries)

 names(spotrates)<-names(bonddata)
 #return list of results
 result <- list(maturity_spectrum=maturity_spectrum,method=method,
                        fit=fit,weights=weights,n_countries=n_countries,
                        cashflows=cf,maturities=m,dirty_prices=p,
                estimated_prices=estimated_prices,yields=yields,
                opt_result=opt_result,spotrates=spotrates)

 class(result) <- "termstrc_singlecurve"
 result
 }


##########################################################################
#             print-method for termstrc_singlecurve                      #
##########################################################################


print.termstrc_singlecurve <-
  function(x,...) {
  cat("-------------------------------------------------\n")
  cat("Parameters for single-curve estimation:\n")
  cat("\n")
  cat("Method:",method,"\n")
  cat("Fitted:",fit,"\n")
  cat("Weights:",weights,"\n")
  cat("\n")
  cat("-------------------------------------------------\n")
  cat("\n")
  parameters <- mapply(function(i) x$opt_result[[i]]$par,1:length(x$opt_result))
  colnames(parameters) <- names(x$opt_result)
  n_par <- as.character(nrow(parameters))
  rownames(parameters) <- switch(n_par,
          "4"=c("beta_0","beta_1","beta_2","tau_1"),
```

```
          "6"=c("beta_0","beta_1","beta_2","tau_1","beta_3","tau_2"))
  print.default(parameters)
  cat("\n")
  }



#######################################################################
#            plot-method for termstrc_singlecurve                     #
#######################################################################

plot.termstrc_singlecurve <-
  function(x,maturity_min=1,maturity_max=15,spread_curves=TRUE,...) {
    # calculate theoretical yield curves
     yield_curves <- switch(x$method,
              "Nelson/Siegel" = mapply(function(i)
                nelson_siegel(x$opt_result[[i]]$par,
                seq(maturity_min,maturity_max,0.01)),
                1:x$n_countries),

              "Svensson" = mapply(function(i) svensson(x$opt_result[[i]]$par,
                seq(maturity_min,maturity_max,0.01)),1:x$n_countries))

    # plot each yield curve seperately
    par(mfrow=c(2,2))
    for (i in 1:x$n_countries) {
      plot(seq(maturity_min,maturity_max,0.01),yield_curves[,i],
      type="l",
      ylim=c(0,0.05),
      xlab="Maturities",
      ylab="Yields",
      lwd=2,
      col="steelblue")
      title(names(x$opt_result)[i])
      grid()
      points(x$yields[[i]],col="red")
      par(ask=TRUE)
    }


    par(mfrow=c(1,2))

    ## plot all yield curves together
    matplot(seq(maturity_min,maturity_max,0.01),yield_curves,type="l",
                col=1:x$n_countries,lty=1,lwd=2,

    xlab="Maturities",
    ylab="Yields",
    xlim=c(maturity_min,maturity_max))
    title("Yield curves")
    legend("bottomright",legend=names(x$opt_result),
    col=1:x$n_countries,lty=1,lwd=2)
    grid()
```

```
    ## calculate spread curves
    if (spread_curves==TRUE)
    yield_curve_ref <- yield_curves[,1]
    n <- x$n_countries-1              #number of spread curves
    spread_curves <- yield_curves[,2:(n+1)]-
    matrix(rep(yield_curve_ref,n),ncol=n)

    ### plot spread curves
    matplot(seq(maturity_min,maturity_max,0.01),
    spread_curves,type="l",col=2:(n+1),lty=1,lwd=2,
    xlab="Maturities",ylab="Spreads",
    xlim=c(maturity_min,maturity_max))
    title("Spread curves")
    legend("bottomright",legend=names(x$opt_result)[2:(n+1)],
    col=2:(n+1),lty=1,lwd=2)
    grid()
}



#######################################################################
#            summary-method for termstrc_singlecurve                  #
#######################################################################

summary.termstrc_singlecurve <-
    function(x) {
    RMSE_p <- mapply(function(i) rmse(x$dirty_prices[[i]],
    x$estimated_prices[[i]]),1:x$n_countries)
    AABSE_p <- mapply(function(i) aabse(x$dirty_prices[[i]],
    x$estimated_prices[[i]]),1:x$n_countries)
    RMSE_y <- mapply(function(i) rmse(x$yields[[i]][,2],
    x$spotrates[[i]]),1:x$n_countries)
    AABSE_y <- mapply(function(i) aabse(x$yields[[i]][,2],
    x$spotrates[[i]]),1:x$n_countries)

    gof <- rbind(RMSE_p,AABSE_p,RMSE_y,AABSE_y)
    colnames(gof) <- names(x$dirty_prices)
    rownames(gof) <- c("RMSE-Prices","AABSE-Prices",
    "RMSE-Yields","AABSE-Yields")
    cat("-------------------------------------------------\n")
    cat("Goodness of fit tests:\n")
    cat("-------------------------------------------------\n")
    cat("\n")
    print.default(gof)
    cat("\n")
    cat("\n")
    convergence <- as.matrix(mapply(function(i) x$opt_result[[i]]$message,
    1:length(x$opt_result)))
    colnames(convergence) <- "Convergence info"
    rownames(convergence) <- names(x$dirty_prices)
    print.default(convergence)
}
```

```
#######################################################################
#                    Bond pricing function                            #
#######################################################################

bond_prices <-
  function(method="Nelson/Siegel", beta, m, cf,multicurve=FALSE) {



  # calculate spot rates for single-curve estimation

  spot_rates <-
    switch(method,
           "Nelson/Siegel" = nelson_siegel(beta,m),
           "Svensson"=svensson(beta,m)
           )
   # replace NAs by zeroes
   spot_rates[is.nan(spot_rates)] <- 0


  # calculate discount factors
  discount_factors <- exp(-m*spot_rates)

  # calculate bond prices
  bond_prices <- apply(cf*discount_factors, 2, sum)

  # return spot rates, discount factors and bond prices
  return (list(spot_rates=spot_rates,
               discount_factors=discount_factors,
               bond_prices=bond_prices))
}

#######################################################################
#            cashflows matrix for a given country                     #
#######################################################################

create_cashflows_matrix <- function(country,include_price=F) {

  n_of_cf <- summary(as.factor(country$CASHFLOWS$ISIN))
  n_of_bonds <- length(n_of_cf)
  max_cf <- max(n_of_cf)
  pos_cf <- c(0,cumsum(n_of_cf))

  CASHFLOWMATRIX <-
    mapply(function(i) c(country$CASHFLOWS$CF[(pos_cf[i]+1):pos_cf[i+1]],
                         rep(0,max_cf-n_of_cf[i])),
           1:n_of_bonds)

  if (include_price == T) {CASHFLOWMATRIX <- rbind(-(country[["PRICE"]]
      +country[["ACCRUED"]]),CASHFLOWMATRIX)}
```

```
  colnames(CASHFLOWMATRIX) <- country$ISIN
  CASHFLOWMATRIX
}

#####################################################################
#              maturities matrix for a given country                #
#####################################################################

create_maturities_matrix <-
  function(country,include_price=F) {

  n_of_cf <- summary(as.factor(country$CASHFLOWS$ISIN))
  n_of_bonds <- length(n_of_cf)
  max_cf <- max(n_of_cf)
  pos_cf <- c(0,cumsum(n_of_cf))
  year_diff <- as.numeric(difftime(as.Date(country$CASHFLOW$DATE),
                           as.Date(country$TODAY),units="days"))/365

  MATURITYMATRIX <-
     mapply(function(i) c(year_diff[(pos_cf[i]+1):pos_cf[i+1]],
                     rep(0,max_cf-n_of_cf[i])),
            1:n_of_bonds)

  if (include_price == T) {MATURITYMATRIX <- rbind(rep(0,n_of_bonds),
                           MATURITYMATRIX)}
  colnames(MATURITYMATRIX) <- country$ISIN
  MATURITYMATRIX
}

#####################################################################
#              bond yields for a given country                      #
#####################################################################

bond_yields <- function(cashflows, m,tol=1e-10) {

  # convert input data to matrices, if necessary
  if (!is.matrix(cashflows))
    cashflows <- as.matrix(cashflows)
  if (!is.matrix(m))
    m < -as.matrix(m)

  # create empty bond yields matrix in appropriate size
  bondyields<-matrix(0, nrow=ncol(cashflows), ncol=2)

  # put maximum of m of every bond into first column of bond yields matrix
  bondyields[,1] <- apply(m, 2, max)

  # traverse list of bonds
  for (i in seq(ncol(cashflows))) {

    # calculate bond price with yield
    yield_function<-function(y) {
       t(cashflows[,i])%*%exp(-m[,i]*y)
```

```
    }

    # calculate roots and
    # put result into second column of bond yields matrix
    bondyields[i,2]<-uniroot(
    yield_function, c(0, 1), tol = tol,maxiter=3000)$root

  }

  # return calculated bond yields matrix
  rownames(bondyields) <- colnames(cashflows)
  colnames(bondyields) <- c("Maturity","Yield")
  bondyields
}

#######################################################################
#                        maturity range                               #
#######################################################################

maturity_range <-
  function(bonddata,lower,upper) {

m <- lapply(bonddata,create_maturities_matrix)
colmax <- function(m) apply(m,2,max)
m_max <- lapply(m,colmax)

bonds_in_range <- function(country) names(country[which(
                 country>lower & country<upper)])

# list with ISINs of bonds in range

isins_range <- lapply(m_max,bonds_in_range)

index_set <- which(unlist(lapply(isins_range,length))>0)

# list with positions of bonds in isins_range

isins_range_pos <- list()

for (i in 1:length(bonddata)) {
        isins_range_pos[[i]] <- which(bonddata[[i]]$ISIN %in%
            isins_range[[i]])
    }
names(isins_range_pos) <- names(bonddata)

# first part of bonddata for filtering

if(length(bonddata[[1]])>8) N=8 else N=6

print(N)

first <- function(lst) lst[c(1:N)]
filtered <- lapply(bonddata,first)
```

```
bonddata_range <- list()
for (i in 1:length(bonddata)) {
bonddata_range[[i]] <- as.list(as.data.frame(filtered[[i]])
                       [isins_range_pos[[i]],])
# convert to character
bonddata_range[[i]][["ISIN"]] <-
as.character(bonddata_range[[i]][["ISIN"]])
bonddata_range[[i]][["MATURITYDATE"]] <-
as.character(bonddata_range[[i]][["MATURITYDATE"]])
bonddata_range[[i]][["STARTDATE"]] <-
as.character(bonddata_range[[i]][["STARTDATE"]])
}
names(bonddata_range) <- names(bonddata)

# list with positions of cashflows in isins_range

isins_range_pos <- list()
for (i in 1:length(bonddata)) {
isins_range_pos[[i]] <- which(bonddata[[i]][["CASHFLOWS"]]
                        [["ISIN"]]%in%isins_range[[i]])
}
names(isins_range_pos) <- names(bonddata)

for (i in 1:length(bonddata)) {
CASHFLOWS <- as.list(as.data.frame(bonddata[[i]]
             [["CASHFLOWS"]])[isins_range_pos[[i]],])
CASHFLOWS$ISIN <- as.character(CASHFLOWS$ISIN)
CASHFLOWS$DATE <- as.character(CASHFLOWS$DATE)
bonddata_range[[i]][["CASHFLOWS"]] <- list()
bonddata_range[[i]][["CASHFLOWS"]] <- CASHFLOWS
}
names(bonddata_range) <- names(bonddata)
bonddata_range

# add TODAY from bonddata

for (i in 1:length(bonddata)) {
bonddata_range[[i]][["TODAY"]] <- bonddata[[i]][["TODAY"]]
}

# delete countries where no bonds are available

bonddata_range <- bonddata_range[index_set]
bonddata_range
}

#####################################################################
#                       Root mean squared error                     #
#####################################################################

rmse <-
function (actual,estimated) {
```

```
                        e <- actual - estimated
                        rmse <- sqrt(1/length(e)*sum((e-mean(e))^2))
      rmse
      }

#####################################################################
#                  Average absolute error                           #
#####################################################################

aabse <-
function (actual,estimated){
                        e <- actual - estimated
      aabse <- 1/length(e)*sum(abs(e-mean(e)))
                }

#####################################################################
#                  Duration                                         #
#####################################################################

# calculates duration, modified duration and weights for optimization

# cf ... cashflows (dirty price included)
# m ... maturities (zero included)
# y ... bond yields
duration <-
function (cf,m,y) {
       y <- matrix(rep(y,nrow(m)),ncol=ncol(m),byrow=T)
       d <- apply(cf*m*exp(-y*m),2,sum)/-cf[1,]
       md <- d/(1+y[1,])
       omega <- (1/md)*sum(1/md)
       cbind(d,md,omega)
    }

# example

cf = matrix(c(-103, 5, 5, 105,-102,3,3,103),ncol=2)
m = matrix(c(0,1,2,3,0,0.5,1.5,2),ncol=2)
y = bond_yields(cf,m)
y = matrix(y[,2],ncol=2)

duration(cf,m,y)

#####################################################################
#                  Spotrate calculation                             #
#####################################################################
#calculate spotrate according to chosen approach,
#optimal parametervector and maturity vector.



 srates <- function(method,beta,m){

  func<- switch(method,
```

```
                                "Nelson/Siegel" = nelson_siegel(beta,m),
                                "Svensson" = svensson(beta,m))
func
        }
```

Anhang B

Sourcecode - Short Rate Modelle

B.1 Sourcecode - Black-Derman-Toy Modell

```
############################################################
# Black-Derman-Toy Model                                   #
# Fitted to Yield Curve Data                               #
############################################################


bdt_y <- function(R,sigR){

# initialise parameters
N <- length(R)-1
T <- N
dt <-  N/T   # currently only dt=1 is available

# pure discount bond prices
P <- (1+R*dt)^(-seq(0,N,dt))
names(P) <- as.character(seq(0,N))

# initalise first node

J <- (N-1)*2+1
Q <- matrix(rep(NA,J*N),J,N)


rownames(Q) <- as.character(seq(-N+1,N-1,dt))
colnames(Q) <- as.character(seq(0,N-1))
r <- Q
d <- Q

i <- "0"
Q[i,i] <- 1
U <- rep(NA,N)
names(U) <- as.character(seq(0,N-1))
U[i] <- R[1]
r[i,i] <- R[1]
```

```
d[i,i] <- 1/(1+r[i,i]*dt)

opt <- rep(NA,N-1)
names(opt) <- as.character(seq(1,N-1))

# evolve tree for short rate
for(i in 1:(N-1)){

  # update security prices at i

  Q[as.character(-i),as.character(i)] <-
    0.5*Q[as.character(-i+1),as.character(i-1)]*
    d[as.character(-i+1),as.character(i-1)]
  Q[as.character(i),as.character(i)] <-
    0.5*Q[as.character(i-1),as.character(i-1)]*
    d[as.character(i-1),as.character(i-1)]

  if(i>1){     # because at i = 1 only extreme cases
  for(j in seq(-i+2,i-2,2)){
   Q[as.character(j),as.character(i)] <-
      0.5*Q[as.character(j-1),as.character(i-1)]*
      d[as.character(j-1),as.character(i-1)] +
      0.5*Q[as.character(j+1),as.character(i-1)]*
      d[as.character(j+1),as.character(i-1)]
     }
  }

  #   use numerical search to solve for U[i]

  f <- function (u) {z <- 0
                     for(j in seq(-i,i,2)){
                     z <- z +
                     Q[as.character(j),as.character(i)]*
                     (1/(1+(u*exp(sigR*j*sqrt(dt))))*dt)
                      }
                      z-P[as.character(i+1)]
                      }
  optimres <- uniroot(f, c(0, 1))
  opt[i] <- optimres$estim.prec
  U[as.character(i)] <- optimres$root

  #   set r[.] and d[.]
  for(j in seq(-i,i,2)){
   r[as.character(j),as.character(i)] <-
      U[as.character(i)]*exp(sigR*(-j)*sqrt(dt))
   d[as.character(j),as.character(i)] <-
      1/(1+r[as.character(j),as.character(i)]*dt)
  }
}

 rownames(Q) <- as.character(seq(N-1,-N+1,-dt))
 rownames(r) <- rownames(Q)
 rownames(d) <- rownames(Q)
```

```
 solution <- list(r=r,d=d,Q=Q,P=P,U=U,opt=opt,N=N)
 class(solution) <- c("bdt_y","binomialtree")
 solution
}

#calculate pure discount bond prices
pd_prices.bdt_y<- function(x,...){

 Ps<-x$d*0.5


 #initialize Ps for backward induction
 Ps[,as.character(x$N-1)]<- rep(1,nrow(x$d)) + x$d[,as.character(x$N-1)]*0


 for ( i in (x$N-2):0){


        for(j in c(i:-i)){

                if (is.na(Ps[as.character(j),as.character(i)])==FALSE) {

                Ps[as.character(j),as.character(i)] <- Ps[as.character(j),
                as.character(i)]*(Ps[as.character(j+1),as.character(i+1)] +
                Ps[as.character(j-1),as.character(i+1)] )

                }
        }

 }
 label <- "bdt_y"
 solution <- list (Ps,label)
 names(solution) <- c("Ps","label")
 class(solution) <- c("pd_prices","binomialtree")
 solution
}

#calculate price of an european option on a pure discount bond
EbondOpt.bdt_y<-function(y,type="put",To,X) {


        price = switch(type,

                        "call"=sum(y$Q[,as.character(To)]*
                        apply(cbind(( pd_prices.bdt(y)$Ps[,
                                as.character(To)]-X),0),1,max),na.rm=TRUE),

                        "put"=sum(y$Q[,as.character(To)]*
                        apply(cbind((X - pd_prices.bdt(y)$Ps[,
                                as.character (To)]),0),1,max),na.rm=TRUE))

 solution <- list(price,type,To,X)
```

```
 class(solution) <- c("EbondOpt")
 names(solution) <- c("Price","type","To","X")
 return(solution)
}

#calculate price of an american option on a pure discount bond
AbondOpt.bdt_y <- function(y,type="put",To,X) {

 #build empty binomial tree for option values
 price<-matrix(NA,nrow=nrow(pd_prices.bdt(y)$Ps), ncol=(To+1))
 rownames(price)<- rownames(pd_prices.bdt(y)$Ps)
 colnames(price)<-colnames(pd_prices.bdt(y)$Ps)[1:(To+1)]


 #define payoff functions
 payoff_call <- function(K,S){apply(cbind(0,S-K),1,max)}
 payoff_put  <- function(K,S){apply(cbind(0,K-S),1,max)}

 payoff<-switch(type,
                                                "call"=payoff_call,
                                                "put"= payoff_put)

 #calculate option value for To:
 #value put = max(X-Ps)
 #value call= max(Ps-X)
 price[,as.character(To)] <- payoff(X,pd_prices.bdt(y)$Ps[,as.character(To)])




#backward induction for american option
for ( i in (To-1):0){

        for(j in c(i:-i)){

                #only perfom calculation for nodes which are not NA
                if(is.na(pd_prices.bdt(y)$Ps[as.character(j),
                as.character(i)])==FALSE) {

                #optionvalue put p[i,j] = max(p[ij], X - Ps[ij])
                #= max(d[ij]*0.5(p[i+1,j+1] + p[i+1,j-1]) , X - Ps[ij] )

                price[as.character(j),as.character(i)] <- max(
                y$d[as.character(j),as.character(i)]*0.5*(
                price[as.character(j+1),as.character(i+1)]+
                price[as.character(j-1),as.character(i+1)]),
                payoff(X,pd_prices.bdt(y)$Ps[as.character(j),
                as.character(i)]))

                }
```

```
        }


 }
 label <- "bdt_y"
  solution <- list(price,type,To,X,label )
 class(solution) <- c("AbondOpt","binomialtree" )
 names(solution) <- c("Price","type","To","X","label")
 return(solution)

}


#calculate bond prices
#x=bdt_y object
#cr=coupon of the bond
b_prices.bdt_y <- function(x,cr){

        B <- matrix(NA,nrow=nrow(x$Q),ncol=ncol(x$Q))
        colnames(B) <- colnames(x$Q)
        rownames(B) <- rownames(x$Q)

        #calculate price for maturity date of the bond
        B[,x$N]<- x$Q[,x$N]*0 + (1+ Sr)

 for(i in (x$N-2):0) {


                for ( j in seq(i,-i,by=-2)){

                B[as.character(j),as.character(i)] <-   x$d[as.character(j),
                as.character(i)]*0.5*(B[as.character(j+1),as.character(i+1)]
                + B[as.character(j-1),as.character(i+1)] +cr)


                        }


        }

 label <- "bdt_y"
 solution <- list (B,label,cr)
 names(solution) <- c("B","label","couponrate")
 class(solution) <- c("b_prices","binomialtree")
 solution

}



#Valuation of an European swaption
Eswaption.bdt_y <- function(x,cr,type="payer",To,X){
```

```
        price = switch(type,

                        "receiver"=sum(x$Q[,as.character(To)]*
                        apply(cbind(( b_prices.bdt_y(x,cr)$B[,
                                as.character(To)]-X),0),1,max),na.rm=TRUE),

                        "payer"=sum(x$Q[,as.character(To)]*
                        apply(cbind((X - b_prices.bdt_y(x,cr)$B[,
                                as.character (To)]),0),1,max),na.rm=TRUE))

 solution <- list(price,cr,type,To,X)
 names(solution) <- c("price","couponrate","type","To","X")
 class(solution) <- "Eswaption"
 return(solution)

}

############################################################
# Black-Derman-Toy Model                                   #
# Fitted to Yield and Volatility Data                      #
############################################################


bdt_yv <- function(R,sigR) {

N <- length(R)-1

# initialise parameters

T <- N

# precompute constants
dt <-  N/T
sdt <- sqrt(dt)

# yield and volatility data
names(R) <- as.character(seq(0,N))
names(sigR) <- as.character(seq(1,N))

# pure discount bond prices
P <- (1+R*dt)^(-seq(0,N,dt))
names(P) <- as.character(seq(0,N))

# initalise first node

J <- (N-1)*2+1
r <- matrix(rep(NA,J*N),J,N)
rownames(r) <- as.character(seq(N-1,-N+1,-dt))
colnames(r) <- as.character(seq(0,N-1))
d <- r
Qu <- r
Qd <- r
U <- rep(NA,N)
```

```
names(U) <- as.character(seq(0,N-1))
sig <- U
i ="0"
U[i] <- R["1"]
sig[i] <- sigR["1"]
r[i,i] <- R["1"]
d[i,i] <- 1/(1+r[i,i]*dt)
Qu["1","1"] <- 1
Qd["-1","1"] <- 1

Pu <- rep(NA,N-1)
names(Pu) <- as.character(2:N)
Pd <- Pu

opt <- matrix(NA,nrow=N-1,ncol=4)
rownames(opt) <- as.character(seq(1,N-1))
colnames(opt)<- c("objective","convergence","message","iterations")

# compute Pu[.] and Pd[.]

for(i in 2:N){

  f <- function(Pu){
        Pu + Pu^exp(-2*sigR[as.character(i)]*sdt)-
        2*P[as.character(i)]*(1+r["0","0"]*dt)
  }
  Pu[as.character(i)] <- uniroot(f, c(0, 1))$root
  Pd[as.character(i)] <- Pu[as.character(i)]^exp(-2*sigR[as.character(i)]*sdt)
}

# ... and now the fun begins ...

# evolve tree for the short rate

for(i in 1:(N-1)){

   # update pure security prices at timestep i
   #
   # sum(c(...),na.rm=TRUE) takes care of NAs
   # the if conditions ensure that the indexing stays inside the matrices

   if(i>1){
            for(j in seq(-i+2,i,2)){
               Qu[as.character(j),as.character(i)] <- sum(c(
                 if(i<=N&&(j-1)>=(-N+1)){
                   0.5*(Qu[as.character(j-1),as.character(i-1)]*
                   d[as.character(j-1),as.character(i-1)])
                 },
                 if(i<=N&&(j+1)<=(N-1)){
                   0.5*(Qu[as.character(j+1),as.character(i-1)]*
                   d[as.character(j+1),as.character(i-1)])
                 }
                 ),na.rm=TRUE)
```

```
              }

            for(j in seq(i-2,-i,-2)){
               Qd[as.character(j),as.character(i)] <- sum(c(
                 if(i<=N&&(j-1)>=(-N+1)){
                   0.5*(Qd[as.character(j-1),as.character(i-1)]*
                   d[as.character(j-1),as.character(i-1)])
                 },
                 if(i<=N&&(j+1)<=(N-1)){
                   0.5*(Qd[as.character(j+1),as.character(i-1)]*
                   d[as.character(j+1),as.character(i-1)])
                 }
               ),na.rm=TRUE)
            }
}

 # solve simultaneously for U[i] and sig[i]


 f_Pu <- function(u,sigma) {z <- 0        # function for Pu[i+1]
                 for(j in seq(-i+2,,2)){
                 z <- z +
                 Qu[as.character(j),as.character(i)]*
                 (1/(1+(u*exp(sigma*j*sdt)*dt))) # ok
                  }
                  as.numeric(z-Pu[as.character(i+1)])
                  }

 f_Pd <- function(u,sigma) {z <- 0       # function for Pd[i+1]
                 for(j in seq(i-2,-i,-2)){
                 z <- z +
                 Qd[as.character(j),as.character(i)]*
                 (1/(1+(u*exp(sigma*j*sdt)*dt)))
                  }
                  as.numeric(z-Pd[as.character(i+1)])
                  }

 f_error <- function(x) {
              u <- x[1]
              sigma <- x[2]
              abs(f_Pu(u,sigma))+abs(f_Pd(u,sigma))
             }

# startpar <- c(U[as.character(i-1)],sig[as.character(i-1)])
startpar <- c(R[as.character(i+1)],sigR[as.character(i+1)])


optimres <- nlminb(startpar,f_error,lower=c(0.01,0.01),upper=c(1,1))
#print(optimres)
opt[i,1] <- optimres$objective
opt[i,2] <- optimres$convergence
opt[i,3] <- optimres$message
opt[i,4] <- optimres$iterations
```

```
  U[as.character(i)] <- optimres$par[1]
  sig[as.character(i)] <- optimres$par[2]

   # set r[.] and d[.]
   for(j in seq(-i,i,2)){
      r[as.character(j),as.character(i)] <-
        U[as.character(i)]*exp(sig[as.character(i)]*j*sdt)
      d[as.character(j),as.character(i)] <-
        1/(1+r[as.character(j),as.character(i)]*dt)
   }

}

 solution <- list(r=r,d=d,Qu=Qu,Qd=Qd,Pu=Pu,Pd=Pd,U=U,sig=sig,opt=opt)
 class(solution) <- c("bdt_yv","binomialtree")
 solution
}
```

B.2 Sourcecode - Hull-White Modell

```
############################################################
# Hull and White one factor short rate model               #
# Fitted to Yield Curve Data                               #
############################################################


hw <- function(R,alpha,sigma){


#time horizon
N<- length(R)+1

#time steps
T<- N


##calculate constants
#time step-size
delta_t<- N/T

#state step-size
delta_r<- sigma*sqrt(3*delta_t)

#M
M<- -alpha*delta_t

#criteria for branch switching
jmax<- ceiling(-0.1853/M)
jmin<- -jmax

#pure discount bond prices
P <- exp(-R[2:N-1]*(seq(1,N-1,delta_t)))


##create trees (in a matrix representation)
#pure securtiy price
Q<-matrix(rep(NA,(2*(min((2*N+1),jmax))+1))*(N-1),
        nrow=2*(min((2*N+1),jmax))+1,ncol=N-1)

rownames(Q) <- as.character(seq(min((2*N+1),jmax),
        -min((2*N+1),jmax),-delta_t))
colnames(Q) <- as.character(seq(0,N-2))

#dicount factor
d<-Q

#branching type
#n for normal
#u for upward branching
```

```
#d for downward branching
b<-Q

#up propability
p_u<-Q

#middle propability
p_m <- Q
#down probpability
p_d<- Q
#r
r<-Q
#x
x<-Q



##initialise start node
i <- "0"
Q[i,i] <- 1
r[i,i] <- R[1]
d[i,i] <- exp(-R[1]*1)
x[i,i] <- 0

##create propability trees
for( i in 0:(N-2)) {

        for(j in seq( min(i,jmax),-min(i,jmax),by=-delta_t)){


                if(j==-jmax) { # upward branching

                        p_u[as.character(j),as.character(i)] =
                        1/6 + (j^2*M^2 - j*M)/2

                        p_m[as.character(j),as.character(i)] =
                        -1/3 - j^2*M^2+2*j*M

                        p_d[as.character(j),as.character(i)]=
                        1 - (1/6 + (j^2*M^2 - j*M)/2  -1/3 - j^2*M^2+2*j*M)

                        b[as.character(j),as.character(i)]="u"


                        }


                        if(j == jmax) {         #down branching
                        p_u[as.character(j),as.character(i)] =
                        7/6 + (j^2*M^2 + 3*j*M)/2
```

```
                        p_m[as.character(j),as.character(i)] =
                        1 - ( 7/6 + (j^2*M^2 + 3*j*M)/2 +
                        1/6+ (j^2*M^2 + j*M)/2)

                        p_d[as.character(j),as.character(i)]=
                        1/6+ (j^2*M^2 + j*M)/2

                        b[as.character(j),as.character(i)]="d"

                                }

                if( j> jmin && j < jmax){ # normal branching

                        p_u[as.character(j),as.character(i)] =
                        1/6 + (j^2*M^2+j*M)/2

                        p_m[as.character(j),as.character(i)] =
                        2/3 - j^2*M^2

                        p_d[as.character(j),as.character(i)]=
                        1-(1/6 + (j^2*M^2+j*M)/2 + 2/3 - j^2*M^2)

                        b[as.character(j),as.character(i)]="n"

                        }


        }

}

##build x tree (i.e. tree for the simplified process)

for(i in 0:(N-2)) {

        x[as.character(min(i,jmax):-min(i,jmax)),as.character(i)] <-
        x["0","0"]+ (min(i,jmax):-min(i,jmax))*delta_r

        }




##calculate Q,r,d tree
for ( i in 1:(N-2)){

        #create temporary matrix for the calculation of Q
         prob<-matrix(rep(0,(min(i,jmax)*2+1)*3),nrow=min(i,jmax)*2+1,ncol=3)
         rownames(prob) <- as.character(seq(min(i,jmax),-min(i,jmax),-delta_t))


        #calculate Q
```

```
for(j in seq(min(jmax,i-1),-min(jmax,i-1),by=-delta_t)){

        #handle normal branching
        if ( b[as.character(j),as.character(i-1)]=="n"){

        prob[as.character(j+1),1]<- prob[as.character(j+1),1] +
        p_u[as.character(j),as.character(i-1)]*Q[as.character(j),
        as.character(i-1)]*d[as.character(j),as.character(i-1)]

        prob[as.character(j),2]<- prob[as.character(j),2] +
        p_m[as.character(j),as.character(i-1)]*Q[as.character(j),
        as.character(i-1)]*d[as.character(j),as.character(i-1)]

        prob[as.character(j-1),3]<- prob[as.character(j-1),3] +
        p_d[as.character(j),as.character(i-1)]*Q[as.character(j),
        as.character(i-1)]*d[as.character(j),as.character(i-1)]


        }

        #handele upwards branching
        if (b[as.character(j),as.character(i-1)]=="u"){

        prob[as.character(j+2),1]<- prob[as.character(j+2),1] +
        p_u[as.character(j),as.character(i-1)]*Q[as.character(j),
        as.character(i-1)]*d[as.character(j),as.character(i-1)]

        prob[as.character(j+1),2]<- prob[as.character(j+1),2] +
        p_m[as.character(j),as.character(i-1)]*Q[as.character(j),
        as.character(i-1)]*d[as.character(j),as.character(i-1)]

        prob[as.character(j),3]<- prob[as.character(j),3] +
        p_d[as.character(j),as.character(i-1)]*Q[as.character(j),
        as.character(i-1)]*d[as.character(j),as.character(i-1)]


                }

        #handel downwards branching
        if (b[as.character(j),as.character(i-1)]=="d"){

        prob[as.character(j),1]<- prob[as.character(j),1] +
        p_u[as.character(j),as.character(i-1)]*Q[as.character(j),
        as.character(i-1)]*d[as.character(j),as.character(i-1)]

        prob[as.character(j-1),2]<- prob[as.character(j-1),2] +
        p_m[as.character(j),as.character(i-1)]*Q[as.character(j),
        as.character(i-1)]*d[as.character(j),as.character(i-1)]

        prob[as.character(j-2),3]<- prob[as.character(j-2),3] +
        p_d[as.character(j),as.character(i-1)]*Q[as.character(j),
        as.character(i-1)]*d[as.character(j),as.character(i-1)]
```

```
                        }


                }


        Q[as.character(min(i,jmax):-min(i,jmax)),as.character(i)] <-
        apply(prob,1,sum)

        a <- (log(sum(Q[as.character(min(i,jmax):-min(i,jmax)),
        as.character(i)]*exp(-(min(i,jmax):-min(i,jmax))*
        delta_t*delta_r)))) - log(P[i+1])/delta_t

        r[as.character(min(i,jmax):-min(i,jmax)),as.character(i)] <-
        x[as.character(min(i,jmax):-min(i,jmax)),as.character(i)] +a

        d[as.character(min(i,jmax):-min(i,jmax)),as.character(i)] <-
        exp(- r[as.character(min(i,jmax):-min(i,jmax)),
        as.character(i)]*delta_t)

        }

solution<- list(N=N,delta_t=delta_t,delta_r=delta_r,jmax=jmax,
        Q=Q,r=r,d=d,p_u=p_u,p_m=p_m,p_d=p_d,b=b)
class(solution) <- c("hw","trinomialtree")
solution
}

#calculate pure discount bond prices for a hw object
pd_prices.hw<- function(x,...){



##matrix for pure discount bond prices
Ps<- matrix(NA,ncol=ncol(x$Q),nrow=nrow(x$Q))
rownames(Ps)<-rownames(x$Q)
colnames(Ps)<-colnames(x$Q)

#initialize Ps for backward induction
Ps[,as.character(x$N-2)]<- rep(1,min(x$N-1,x$jmax)*2+1)

##backward induction for pure discount bond prices
for ( i in (x$N-3):0){


        for(j in seq(min(x$jmax,i),-min(x$jmax,i),by=-x$delta_t)){

                #handle normal branching
                if ( x$b[as.character(j),as.character(i)]=="n"){

                Ps[as.character(j),as.character(i)]<- x$d[as.character(j),
                as.character(i)]*sum((Ps[as.character((j+1):(j-1)),
```

```
                as.character(i+1)]*c(x$p_u[as.character(j),as.character(i)],
                x$p_m[as.character(j),as.character(i)],x$p_d[as.character(j),
                as.character(i)])))


                        }



                #handle upward branching
                if ( x$b[as.character(j),as.character(i)]=="u"){

                Ps[as.character(j),as.character(i)]<- x$d[as.character(j),
                as.character(i)]*sum((Ps[as.character((j+2):(j))),
                as.character(i+1)]*c(x$p_u[as.character(j),as.character(i)],
                x$p_m[as.character(j),as.character(i)],x$p_d[as.character(j),
                as.character(i)])))

                        }


                #handle downward branching
                if ( x$b[as.character(j),as.character(i)]=="d"){

                Ps[as.character(j),as.character(i)]<- x$d[as.character(j),
                as.character(i)]*sum((Ps[as.character((j):(j-2))),
                as.character(i+1)]*c(x$p_u[as.character(j),as.character(i)],
                x$p_m[as.character(j),as.character(i)],x$p_d[as.character(j),
                as.character(i)])))


                        }

        }


}
label <- "hw"
solution <- list (Ps,x$b,x$Q,x$jmax,label)
names(solution) <- c("Ps","b","Q","jmax","label")
class(solution) <- c("pd_prices","trinomialtree")
solution
}


#valuation of an european option on a pure discount bond
EbondOpt.hw<-function(y,type="put",To,X) {



        payoff = switch(type,
                        "call"=apply(cbind(0,pd_prices.hw(y)$Ps[
                        as.character(min(To,y$jmax):-min(To,y$jmax)),
```

```
                        as.character(To-1)]-X),1,max),

                        "put"=apply(cbind(0,X-pd_prices.hw(y)$Ps[
                        as.character(min(To,y$jmax):-min(To,y$jmax)),
                        as.character(To-1)]),1,max))

price<- sum((y$Q[as.character(min(To,y$jmax):-min(To,y$jmax)),
                as.character(To-1)])*payoff)

label <- "hw"
solution <- list(price,type,To,X,label)
class(solution) <- "EbondOpt"
names(solution) <- c("Price","type","To","X","label")
return(solution)
 }


#valuation of an american option on a pure discount bond
AbondOpt.hw <- function(y,type="put",To,X) {


#build trinomial tree for american option
price<-matrix(NA,nrow=nrow(pd_prices.hw(y)$Ps), ncol=(To+1))
rownames(price)<- rownames(pd_prices.hw(y)$Ps)
colnames(price)<-colnames(pd_prices.hw(y)$Ps)[1:(To+1)]

payoff_call <- function(K,S){apply(cbind(0,S-K),1,max)}
payoff_put  <- function(K,S){apply(cbind(0,K-S),1,max)}

  payoff<-switch(type,
                                                "call"=payoff_call,
                                                "put"= payoff_put)


 #calculate option value for To:
 #value put = max(X-Ps)
 #value call= max(Ps-X)
 price[,as.character(To)] <- payoff(X,pd_prices.hw(y)$Ps[,as.character(To)])



#backward induction for american option
for ( i in (To-1):0){


        for(j in seq(min(y$jmax,i),-min(y$jmax,i),by=-y$delta_t)){

                #handle normal branching
                if ( y$b[as.character(j),as.character(i)]=="n"){


                price[as.character(j),as.character(i)] <-
                max(payoff(X,pd_prices.hw(y)$Ps[as.character(j),
```

```
                as.character(i)]),y$d[as.character(j),as.character(i)]*sum(
                payoff(X,pd_prices.hw(y)$Ps[as.character((j+1):(j-1)),
                as.character(i+1)])*c(y$p_u[as.character(j),as.character(i)],
                y$p_m[as.character(j),as.character(i)],y$p_d[as.character(j),
                as.character(i)])))

                        }

                #handle upward branching
                if ( y$b[as.character(j),as.character(i)]=="u"){

                price[as.character(j),as.character(i)] <-
                max(payoff(X,pd_prices.hw(y)$Ps[as.character(j),
                as.character(i)]),y$d[as.character(j),as.character(i)]*sum(
                payoff(X,pd_prices.hw(y)$Ps[as.character((j):(j+2)),
                as.character(i+1)])*c(y$p_u[as.character(j),as.character(i)],
                y$p_m[as.character(j),as.character(i)],y$p_d[as.character(j),
                as.character(i)])))

                        }

                #handle downward branching
                if ( y$b[as.character(j),as.character(i)]=="d"){

                price[as.character(j),as.character(i)] <-
                max(payoff(X,pd_prices.hw(y)$Ps[as.character(j),
                as.character(i)]),y$d[as.character(j),as.character(i)]*sum(
                payoff(X,pd_prices.hw(y)$Ps[as.character((j):(j-2)),
                as.character(i+1)])*c(y$p_u[as.character(j),as.character(i)],
                y$p_m[as.character(j),as.character(i)],y$p_d[as.character(j),
                as.character(i)])))



                        }

                }


        }

label <- "hw"
solution <- list(price,y$b,type,To,X,label)
names(solution) <- c("Price", "b","type","To","X","label" )
class(solution) <- c("AbondOpt","hw","trinomialtree")
solution

}
```

B.3 Sourcecode - S3 Methoden für das Black-Derman-Toy und Hull-White Modell

```
############################################################
# S3 Plot Methods
#
############################################################

#trinomialtree plot for trees in a matrix representation
plot.trinomialtree <- function(M,b,dx=0,dy=0.25,digits=3,col_u="black",
        col_n="black",col_d="black",...){


x <- seq(min(as.numeric(colnames(M))),max(as.numeric(colnames(M))),by=0.5)
y <- (-max(c(as.numeric(colnames(M)))): max(c(as.numeric(colnames(M)))))
plot.default(x,y,  type = "n",axes=FALSE, xlab="i", ylab="j",...)

axis(2,y,y)

x_tickname <- seq(min(as.numeric(colnames(M))),max(as.numeric(colnames(M))))
axis(1,x_tickname,x_tickname)

 for (i in c(as.numeric(colnames(M)))){

        for(j in c(as.numeric(rownames(M)))){

                if(is.na(M[as.character(j),as.character(i)])==FALSE) {

                points(i,j,pch=19)
                text(i+dx,j+dy,round(M[as.character(j),
                        as.character(i)],digits))

                        if(b[as.character(j),as.character(i)]=="n" &&
                        (i <  max(c(as.numeric(colnames(M)))))) {

                                lines(c(i,i+1),c(j,j+1),col=col_u)
                                lines(c(i,i+1),c(j,j),col=col_n)
                                lines(c(i,i+1),c(j,j-1),col=col_d)
                                }


                        if(b[as.character(j),as.character(i)]=="d" &&
                        (i <  max(c(as.numeric(colnames(M)))))) {

                                lines(c(i,i+1),c(j,j),col=col_n)
                                lines(c(i,i+1),c(j,j-1),col=col_d)
                                lines(c(i,i+1),c(j,j-2),col=col_d)
                                }

                        if(b[as.character(j),as.character(i)]=="u" &&
                        (i <  max(c(as.numeric(colnames(M)))))) {
```

```
                                    lines(c(i,i+1),c(j,j+2),col=col_u)
                                    lines(c(i,i+1),c(j,j+1),col=col_u)
                                    lines(c(i,i+1),c(j,j),col=col_n)
                                    }
                }


        }
 }
}

#binomialtree plot for trees in a matrix representation
plot.binomialtree <- function(M,dx=0,dy=0.2,digits=3,
                                        col_u="black",col_d="black",...){


x <- seq(min(as.numeric(colnames(M))),
                        max(as.numeric(colnames(M))),by=0.5)
y <- (-max(c(as.numeric(colnames(M))))): max(c(as.numeric(colnames(M)))))

plot.default(x,y,  type = "n",axes=FALSE, xlab="i", ylab="j",...)

axis(2,y,y)

x_tickname <- seq(min(as.numeric(colnames(M))),max(as.numeric(colnames(M))))
axis(1,x_tickname,x_tickname)

 for (i in c(as.numeric(colnames(M)))){

        for(j in c(as.numeric(rownames(M)))){

                if(is.na(M[as.character(j),as.character(i)])==FALSE) {

                points(i,j,pch=19)
                text(i+dx,j+dy,round(M[as.character(j),
                        as.character(i)],digits))


                        if(i <  max(c(as.numeric(colnames(M)))) ) {
                                lines(c(i,i+1),c(j,j+1),col=col_u)
                                lines(c(i,i+1),c(j,j-1),col=col_d)
                                }

                }
        }
  }

}

#use these numbers for the desired tree plots
#5 for Q (state prices)
#6 for r (short rate)
```

```
#7 for d (discoutn factors)
#8 for p_u (upward branching probabilities)
#9 for p_m (normal branching probabilites)
#10for p_d (downwad branching probabilites)

plot.hw <- function(x,trees,dx=0,dy=0.25,digits=3,
                        col_u="black",col_n="black",col_d="black",...) {

        par(mfrow=c(length(trees),1))

                for (p in c(trees)){

                plot.trinomialtree(M=x[[p]],b=x$b,dx=dx,dy=dy,digits=digits,
                col_u=col_u,col_n=col_n,col_d=col_d,main=names(x[p]))

                }

}


#use these numbers for the desired tree plots
#1 for r (short rate)
#2 for d (discount factors)
#3 for Q (state prices)


plot.bdt_y <- function(x,dx=0,dy=0.2,digits=3,
                                col_u="black",col_d="black",...) {

        par(mfrow=c(2,1))

                for (p in c(1,2)){

                plot.binomialtree(M=x[[p]],dx=dx,dy=dy,digits=digits,
                col_u=col_u,col_d=col_d,main=names(x[p]),...)

                }

}


#use these numbers for the desired tree plots
#1 for r (short rate)
#2 for d (discount factors)
#3 for Qu (state prices)
#4 for Qd (state prices)


plot.bdt_yv <- function(x,trees,dx=0,dy=0.2,digits=3,
                        col_u="black",col_d="black",...) {

        par(mfrow=c(length(trees),1))
```

```
                for (p in c(trees)){

                plot.binomialtree(M=x[[p]],dx=dx,dy=dy,digits=digits,
                col_u=col_u,col_d=col_d,main=names(x[p]))

                }

}

plot.pd_prices <- function(x,dx=0,dy=0.25,digits=3,
                        col_u="black",col_n="black",col_d="black",...) {

                                if ( x$label=="hw"){
                                plot.trinomialtree(x$Ps,b=x$b,dx=dx,dy=dy,
                                digits=digits, col_u=col_u,col_n=col_n,
                                col_d=col_d,
                                main="Pure Discount Bond Prices",...)}


                                if ( x$label=="bdt_y"){
                                plot.binomialtree(M=x$Ps,dx=dx,dy=dy,
                                digits=digits, col_u=col_u,col_d=col_d,
                                main="Pure Discount Bond Prices",...)
                                }



}

plot.b_prices <- function(x,dx=0,dy=0.25,digits=3,
                        col_u="black",col_n="black",col_d="black",...){


                                plot.binomialtree(M=x$B,dx=dx,dy=dy,
                                digits=digits, col_u=col_u,col_d=col_d,
                                main="Coupon Bond Prices",...)

}



plot.AbondOpt <- function(x,dx=0,dy=0.25,digits=3,
                        col_u="black",col_n="black",col_d="black",...){

                                if ( x$label=="hw"){
                                plot.trinomialtree(x$Price,x$b,dx=dx,
                                dy=dy,digits=digits,
                                col_u=col_u,col_n=col_n,col_d=col_d,
                                main="Price",...)}

                                if ( x$label=="bdt_y"){
                                plot.binomialtree(M=x$Price,dx=dx,dy=dy,
```

```
                                    digits=digits,
                                    col_u=col_u,col_d=col_d,main="Price",...)

                                    }
}



############################################################
# S3 Print Methods
#
############################################################


print.hw <-
  function(x,...) {
    cat("-------------------------------------------------\n")
    cat("       Hull-White Model fitted to Yield Data:     \n")
    cat("-------------------------------------------------\n")
    cat("\n")
    cat("r ... shortrate:\n")
    cat("-------------------------------------------------\n")
    print.default(x$r)
    cat("\n")
    cat("d ... discount function:\n")
    cat("-------------------------------------------------\n")
    print.default(x$d)
    cat("\n")
    cat("Q ... state prices:\n")
    cat("-------------------------------------------------\n")
    print.default(x$Q)
    cat("\n")
  }

print.bdt_y <-
  function(x,...) {
    cat("-------------------------------------------------\n")
    cat("Black-Derman-Toy Model fitted to Yield Data:\n")
    cat("-------------------------------------------------\n")
    cat("\n")
    cat("r ... shortrate:\n")
    cat("-------------------------------------------------\n")
    print.default(x$r)
    cat("\n")
    cat("d ... discount function:\n")
    cat("-------------------------------------------------\n")
    print.default(x$d)
    cat("\n")
    cat("Q ... state prices:\n")
    cat("-------------------------------------------------\n")
    print.default(x$Q)
    cat("\n")
  }
```

```
print.bdt_yv <-
  function(x,...) {
    cat("-------------------------------------------------\n")
    cat("Black-Derman-Toy Model\n")
    cat("fitted to Yield and Volatility Data:\n")
    cat("-------------------------------------------------\n")
    cat("\n")
    cat("r ... shortrate:\n")
    cat("-------------------------------------------------\n")
    print.default(x$r)
    cat("\n")
    cat("d ... discount function:\n")
    cat("-------------------------------------------------\n")
    print.default(x$d)
    cat("\n")
    cat("Qu ... state securities (Arrow-Debreu)\n")
    cat("       prices for an up movement:\n")
    cat("-------------------------------------------------\n")
    print.default(x$Qu)
    cat("\n")
    cat("Qd ... state securities (Arrow-Debreu)\n")
    cat("       prices for a down movement:\n")
    cat("-------------------------------------------------\n")
    print.default(x$Qd)
    cat("\n")
  }

print.pd_prices <-
  function(x,...) {


        if (x$label=="hw" ){

    cat("-------------------------------------------------\n")
    cat("       Hull-White Model fitted to Yield Data:      \n")
    cat("-------------------------------------------------\n")
    cat("\n")
    cat("Ps ... Pure Discount Bond Prices:\n")
    cat("-------------------------------------------------\n")
    print.default(x$Ps)
    cat("\n")
   }

        if ( x$label == "bdt_y"){

    cat("-------------------------------------------------\n")
     cat("Black-Derman-Toy Model fitted to Yield Data:\n")
    cat("-------------------------------------------------\n")
    cat("\n")
    cat("Ps ... Pure Discount Bond Prices:\n")
    cat("-------------------------------------------------\n")
    print.default(x$Ps)
```

```
    cat("\n")
   }



}

print.b_prices <-
  function(x,...) {

    cat("-------------------------------------------------\n")
    cat("B ... Coupon Bond Prices:      \n")
    cat("-------------------------------------------------\n")
    print.default(x$B)
    cat("\n")
    cat("cr ... Coupon Rate:\n")
    cat("-------------------------------------------------\n")
    print.default(x$couponrate)

   }


print.EbondOpt <-
  function(x,...) {

    cat("-------------------------------------------------\n")
    cat("       European Bond Option:      \n")
    cat("-------------------------------------------------\n")
    cat("\n")
    cat("P...Price:\n")
    cat("-------------------------------------------------\n")
    print.default(x$Price)
    cat("\n")
   }

print.AbondOpt <-

          function(x,...) {
    cat("-------------------------------------------------\n")
    cat("       American Bond Option:      \n")
    cat("-------------------------------------------------\n")
    cat("\n")
    cat("P...Price:\n")
    cat("-------------------------------------------------\n")
    print.default(x$Price[as.character(0),as.character(0)])
    cat("\n")
    cat("-------------------------------------------------\n")
    cat("Price-Tree:\n")
    cat("-------------------------------------------------\n")
    print.default(x$Price)
    cat("\n")
   }
```

```
print.Eswaption <-
  function(x,...) {

    cat("-----------------------------------------------------\n")
    cat("European Swaption Price:      \n")
    cat("-----------------------------------------------------\n")
    print.default(x$price)
    cat("\n")
    cat("cr ... Swap rate:\n")
    cat("-----------------------------------------------------\n")
    print.default(x$couponrate)
    cat("\n")
    cat("Type:\n")
    cat("-----------------------------------------------------\n")
    print.default(x$type)
    cat("\n")
    cat("Time to maturity:\n")
    cat("-----------------------------------------------------\n")
    print.default(x$To)
    cat("\n")
    cat("Strike Price = swap principal:\n")
    cat("-----------------------------------------------------\n")
    print.default(x$X)
   }


############################################################
# S3 Summary Methods
#
############################################################


summary.hw <-
  function(x,...) {
    cat("-----------------------------------------------------\n")
    cat("       Hull-White Model fitted to Yield Data:      \n")
    cat("-----------------------------------------------------\n")
    cat("\n")
    cat("N ... Time steps :\n")
    print.default(x$N)
    cat("-----------------------------------------------------\n")
    cat("\n")
    cat("delta.t ... Time steps size :\n")
    print.default(x$delta_t)
    cat("-----------------------------------------------------\n")
        cat("\n")
    cat("R...observed term structure:\n")
    print.default(R)
    cat("-----------------------------------------------------\n")
        cat("\n")
    cat("sigma...volatility:\n")
    print.default(sigma)
    cat("-----------------------------------------------------\n")
```

```
        cat("\n")
    cat("alpha...drift paramter:\n")
    print.default(alpha)
        cat("-------------------------------------------------\n")
        cat("\n")
    cat("jmax ...branching parameter:\n")
    print.default(x$jmax)
    cat("-------------------------------------------------\n")
    cat("\n")
    cat("b-tree ...branching typ for each node :\n")
    cat("n...normal|u...upwards|d...downwards :\n")
    cat("-------------------------------------------------\n")
    print.default(x$b)

  }


summary.bdt_y <-
  function(x,...) {
    cat("-------------------------------------------------\n")
    cat("Black-Derman-Toy Model fitted to Yield Data:\n")
    cat("-------------------------------------------------\n")
    cat("\n")
    cat("Estimation precision from uniroot():\n")
    cat("-------------------------------------------------\n")
    print.default(x$opt)
    cat("\n")
    cat("U ... :\n")
    cat("-------------------------------------------------\n")
    print.default(x$U)
    cat("\n")
    cat("P ... :\n")
    cat("-------------------------------------------------\n")
    print.default(x$P)
  }

summary.bdt_yv <-
  function(x,...) {
    cat("-------------------------------------------------\n")
    cat("Black-Derman-Toy Model\n")
    cat("fitted to Yield and Volatility Data:\n")
    cat("-------------------------------------------------\n")
    cat("\n")
    cat("Convergence info from nlm():\n")
    cat("-------------------------------------------------\n")
    print.default(x$opt)
    cat("\n")
    cat("U ... :\n")
    cat("-------------------------------------------------\n")
    print.default(x$U)
    cat("\n")
    cat("sig ... calibrated volatility parameter:\n")
    cat("-------------------------------------------------\n")
```

```
    print.default(x$sig)
    cat("\n")
    cat("Pu ... price of bond in up movement:\n")
    cat("-------------------------------------------------\n")
    print.default(x$Pu)
    cat("\n")
    cat("Pd ... price of bond in down movement:\n")
    cat("-------------------------------------------------\n")
    print.default(x$Pd)
  }

summary.pd_prices <-
  function(x,...) {

        if(x$label=="hw") {

    cat("-------------------------------------------------\n")
    cat("       Hull-White Model fitted to Yield Data:      \n")
    cat("-------------------------------------------------\n")
    cat("\n")
    cat("Ps for this branching process\n")
    print.default(x$b)
    cat("\n")
   }

   if(x$label=="bdt_y") {
   print(x)
   }

}

summary.b_prices <-
  function(x,...) { print(x,...)}

summary.EbondOpt <-
  function(x,...) {
    cat("-------------------------------------------------\n")
    cat("       European Bond Option:      \n")
    cat("-------------------------------------------------\n")
    cat("\n")
    cat("Type:\n")
    cat("-------------------------------------------------\n")
    print.default(x$type)
    cat("\n")
    cat("-------------------------------------------------\n")
    cat("X...Strike Price\n")
    print.default(x$X)
    cat("\n")
    cat("-------------------------------------------------\n")
    cat("To...Time to maturity\n")
    print.default(x$To)
    cat("\n")
    cat("-------------------------------------------------\n")
```

```
    cat("P...Price\n")
    print.default(x$Price)
    cat("\n")
   }


summary.AbondOpt <-
  function(x,...) {
    cat("-----------------------------------------------------\n")
    cat("       American Bond Option:      \n")
    cat("-----------------------------------------------------\n")
    cat("\n")
    cat("Type:\n")
    cat("-----------------------------------------------------\n")
    print.default(x$type)
    cat("\n")
    cat("-----------------------------------------------------\n")
    cat("X...Strike Price\n")
    print.default(x$X)
    cat("\n")
    cat("-----------------------------------------------------\n")
    cat("To...Time to maturity\n")
    print.default(x$To)
    cat("\n")
    cat("-----------------------------------------------------\n")
    cat("P...Price\n")
    print.default(x$Price[as.character(0),as.character(0)])
    cat("\n")
   }

summary.Eswaption <-
  function(x,...) { print(x)}
```

Anhang C

Sourcecode - Optionsbewertung

```
############################################################
#  functions for the simulation of jump diffusion models #
############################################################

#load required package fOptions
library(fOptions)

#generate poission random numbers
N<-function(pathlength,mcsteps,lambda){

        pois=matrix(rpois(mcsteps*pathlength,lambda),ncol=mcsteps)
        pois
        }

#generate normal distributed numbers subject to N
M <- function(a,b,N,eps2) {

        path = a*N + b

        }

#jump diffusion path
jumppath <- function (eps,M,mu,sigma,deltat) {

        path=(mu-sigma*sigma/2 )*deltat + sigma*sqrt(deltat)*eps + M
        path
                }



#payoff-function asian arithmetic call or put
aAsian<-function(path,Type,S,X,Time,r){
        St<-mean(S*exp(cumsum(path)))

        if (Type =="c") payoff = exp(-r*Time)*max(St-X,0)
        if (Type == "p") payoff = exp(-r*Time)*max(0,X-St)
```

```
        payoff
        }


# main mc simulation function for mertons jump diffusion model
MCOptionJD <- function (optioncharac=list(Type,S,X,Time,sigma,r,mu),
                        mcsettings=list(deltat,pathlength,mcsteps,
                        mcloops,antithetic,standardization,
                        trace,scrambling,seed),
                        pathgen,
                        payofffunc,
                        jumpsettings=list(a,b,lambda)) {


                #initialize ci and sim_payoff matrix
                ci<-matrix(0,ncol=2,nrow=mcsettings$mcloops)
                sim_price<-matrix(0,ncol=1,nrow=mcsettings$mcloops)

                #perform simulation as often as specified in mcsettings$mcloops
                for (j in 1:mcsettings$mcloops) {


                #generate jump diffusion path
                path<- pathgen(rnorm.pseudo(mcsettings$pathlength,
                                        mcsettings$mcsteps),
                        M(jumpsettings$a,jumpsettings$b,N(
                        mcsettings$pathlength,
                        mcsettings$mcsteps,jumpsettings$lambda)),
                        mu=optioncharac$mu,optioncharac$sigma,
                        mcsettings$deltat)


                #calculate payoff
                it<- mapply(function(i) payofffunc(path[,i],optioncharac$Type,
                                optioncharac$S,optioncharac$X,
                                optioncharac$Time,optioncharac$r),
                                1:mcsettings$mcsteps)

                #save result
                ci[j,]<-c(mean(it)-1.96*sd(it)/sqrt(mcsettings$mcsteps),
                        mean(it)+1.96*sd(it)/sqrt(mcsettings$mcsteps))
                sim_price[j,]<-mean(it)

                                                        }
                #return result
                return=list(ci=ci,sim_price=sim_price)

 }
```

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Black, F., Derman, E. und Toy, W. (1990): A One-factor Model of Interest Rates and its Applications to Treasury Bond Options. Financial Analysts' Journal, 46, 33–39

Black, Fischer und Scholes, Myron (1973): The Pricing of Options and Corporate Liabilities. Journal of Political Economy, 81, 637–654

Bliss, Robert R. (1996): Testing term structure estimation methods. Federal Reserve Bank of Atlanta (96-12). – Technischer Bericht

Bolder, David und Streliski, David (1999): Yield Curve Modelling at the Bank of Canada. Bank of Canada (84). – Technical Reports

Clewlow, Les und Strickland, Chris (1998): Implementing Derivative Models. John Wiley & Sons, Wiley Series in Financial Engineering

Cox, J.A., Ingersoll, J.E. und Ross, S.A. (1985): A Theory of the Term Structure of Interest Rates. Econometrica,, Nr. 53, 385–467

Cox, J.A., Ross, A. und Rubinstein, M. (October 1979): Option Pricing: A Simplified Approach. Journal of Financial Economics 7

Diebold, Francis X. und Li, Canlin (2003): Forecasting the Term Structure of Government Bond Yields. National Bureau of Economic Research, Inc (10048). – Technischer Bericht

Ferenczi, Izabella (2005): Globale Optimierung unter Nebenbedingungen mit dünnen Gittern. Diplomarbeit, Technische Universität München - Zentrum Mathematik, ⟨URL: `http://www.mathematik.uni-wuerzburg.de/~ferenczi/tagungen.html`⟩

Frankel, J.A und Lown, C.S. (1994): An Indicator of Future Inflation Extracted from the Steepness of the Interest Rate Yield Curve along its Entire Length. Quarterly Journal of Economics,, Nr. 109, 517–530

Geyer, Alois und Mader, Richard (1999): Estimation of the term structure of interest rates - A parametric approach. Österreichische Nationalbank (37). – Working Paper

Glassermann, Paul (2005): Monte Carlo Methods in Financial Engineering. Springer Verlag - Berlin Heidelberg New York

Gray, David M. (1990): Usage Summary for Selected Optimization Routines. Bell Laboratories (153). – Computing Science Technical Report ⟨URL: `http://netlib.bell-labs.com/cm/cs/cstr/153.pdf`⟩

Halton, J.H. (1960): On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik, 2, 84–90

Halton, J.H. und Smith, G.B (1964): Algorithm 247: Radical-inverse quasi-random point sequence. Communicatons of the ACM 7, Nr. 701-702

Haug, E. G. (1997): The complete Guide to Option Pricing Formulas. McGraw Hill

Ho, T.S.Y und Lee, S-B (1986): Term Structure Movements and Pricing Interest Rate Contingent Claims. Journal of Finance, 41, 1011–1029

Hornik, Kurt (2006): The R FAQ. 2006 ⟨URL: `http://CRAN.R-project.org/doc/FAQ/R-FAQ.html`⟩ – Zugriff am 4-01-2007

Hull, John und White, Alan (1990): Pricing interest-rate derivative securities. The Review of Financial Studies, 3, Nr. 4, 573–592

Hull, John und White, Alan (1994): Numerical Procedures for Implementing Term Structure Models I:Single-factor Models. The Journal of Derivatives,, Nr. Fall, 7–16

Hull, John C. (2000): Options, Futures & Other Derivatives. 5. Auflage. Prentice-Hall, National Bureau of Economic Research, Inc, 4871

Ihaka, Robert und Gentleman, Robert (1996): R: A Language for Data Analysis and Graphics. Journal of Computational and Graphical Statistics, 5, Nr. 3, 299–314

Jamshidian, F. (1991): Forward Induction and Construction of Yield Curve Diffusion Models. Journal of Fixed Income, 1, 62–74

Jankowitsch, Rainer und Pichler, Stefan (2004): Parsimonious Estimation of Credit Spreads. The Journal of Fixed Income, 14, Nr. 3, 49–63

Ligges, Uwe (2005): Programmieren mit R. Springer Berlin Heidelberg New York

London, Justin (2004): Modeling Derivatives in C++. John Wiley&Sons. Ltd

Merton, Robert C. (1973): Theory of Rational Option Pricing. Bell Journal of Economics and Management Science, 4, 141–183

Merton, Robert C. (1976): Option pricing when underlying stock returns are discontinuous. Journal of Financial Economics, 3, 125:144

Nelson, Charles R. und Siegel, Andrew F. (1987): Parsimonious Modeling of Yield Curves. The Journal of Business, 60, Nr. 4, 473–489

Novak, E. und Ritter, K.; C.A. Floudas, P.M. Paradalos (Hrsg.) (1996): Global Optimization Using Hyperbolic Cross Points. Kluwer Academic Publishers, 19–33

R Development Core Team (2006): The R language definition. R Foundation for Statistical Computing, 2006 ⟨URL: `http://www.R-project.org`⟩

Sandmann, Klaus (1999): Einführung in die Stochastik von Finanzmärkten. Springer Verlag - Berlin Heidelberg New York

Sobol', I.H. (1967): On the distribution of points in a cube and the approximation evaluation of integrals. USSR Journal of Computational Mathematics and Mathematical Physics, 7, 784–802

Svensson, Lars E.O. (1994): Estimating and Interpreting Forward Interest Rates: Sweden 1992 -1994. National Bureau of Economic Research, Inc (4871). – Technical Reports

Vasicek, O.A. (1977): An Equilibrium Characterization of the Term Structure . Journal of Financial Economics, 5, 177–188

Wilmott, Paul (2001): Paul Wilmott Indroduces Quantitative Finance. John Wiley&Sons. Ltd

Wilmott, Paul (2006): Paul Wilmott on Quantitative Finance. 2. Auflage. John Wiley&Sons. Ltd

Wuertz, Diethelm (2006a): fBasics: Rmetrics - Marketes and Basic Statistics. 2006 ⟨URL: `http://www.rmetrics.org`⟩

Wuertz, Diethelm (2006b): fCalendar: Rmetrics - Chronological Objects. 2006 ⟨URL: `http://www.rmetrics.org`⟩

Wuertz, Diethelm (2006c): fEcofin: Ecofin - Selected Economic and Financial Data Sets. 2006 ⟨URL: `http://www.rmetrics.org`⟩

Wuertz, Diethelm (2006d): fExtremes: Rmetrics - Extreme Financial Market Data. 2006 ⟨URL: `http://www.rmetrics.org`⟩

Wuertz, Diethelm (2006e): fMultivar: Rmetrics - Multivariate Market Analysis. 2006 ⟨URL: `http://www.rmetrics.org`⟩

Wuertz, Diethelm (2006f): fOptions: Rmetrics - Option Valuation. 2006 ⟨URL: `http://www.rmetrics.org`⟩

Wuertz, Diethelm (2006g): fPortfolio: Rmetrics - Portfolio Selection and Optimization. 2006 ⟨URL: `http://www.rmetrics.org`⟩

Wuertz, Diethelm (2006h): fSeries: Rmetrics - The Dynamical Process Behind Markets. 2006 ⟨URL: `http://www.rmetrics.org`⟩

Wuertz, Diethelm (2006i): Rmetrics. 2006 ⟨URL: `http://www.rmetrics.org`⟩ – Zugriff am 4-01-2007