Zürcher Hochschule für Angewandte Wissenschaften
ZHAW

School of Management and Law
Abteilung Banking, Finance, Insurance

Bachelor of Science in Business Administration
Studienrichtung Banking and Finance

Bewertung von amerikanischen Optionen unter Berücksichtigung diskreter Dividenden

Bachelorarbeit

vorgelegt von:

Thomas Frei

Klasse W.BA.BO.13HS.TZBFc

Matrikelnummer: S00714550

eingereicht bei:

Dr. Norbert Hilber

Zürich, 24. Mai 2017

Management Summary

«Bewertung von amerikanischen Optionen unter Berücksichtigung diskreter Dividenden» - Bachelor-Thesis eingereicht von Thomas Frei

Optionen nehmen an den heutigen Finanzmärkten eine wichtige Stellung ein. Die Ermittlung ihrer fairen Preise ist darum ein bedeutendes Thema in der Finanzmathematik. Unter der Annahme, dass der Basiswert keine Dividenden ausschüttet, lässt sich eine europäische Option im Black-Scholes-Modell einfach analytisch bewerten. Aufgrund ihres umfassenden Ausübungsrechts ist der Wert einer amerikanischen Option von einer optimalen Ausübungsstrategie abhängig. Aus diesem Grund lassen sich amerikanische Kontrakte nicht analytisch bewerten. Es muss auf numerische Methoden ausgewichen werden. Diese erlauben dafür die Berücksichtigung von Dividendenzahlungen.

Das Ziel dieser Arbeit besteht darin, das Bewertungsproblem für amerikanische Optionen aufzuzeigen und unter Anwendung der Finiten-Differenzen-Methode numerisch zu lösen. Es wird ein Programm zur Bestimmung des fairen Preises amerikanischer Optionen unter Berücksichtigung diskreter Dividendenzahlungen ausgearbeitet.

In einem ersten Schritt wird das Konzept der Optionsbewertung mittels finiter Differenzen für den einfachen Fall von europäischen Optionen beschrieben. Anschliessend wird eine Kernroutine für dieses Bewertungsproblem implementiert. Hiernach wird auf die wesentliche Herausforderung bei der Bewertung von Optionen amerikanischen Stils eingegangen. Die gewonnenen Einsichten werden genutzt, um die Kernroutine für die Bewertung amerikanischer Optionen zu adaptieren. Das Problem der freien Randwerte wird dabei mit Hilfe des Projected Successive Over-Relaxation-Verfahren bewältigt. Im Anschluss werden die Auswirkungen diskreter Dividendenzahlungen erörtert und schliesslich ein Algorithmus implementiert, welcher das Bewertungsproblem für amerikanische Optionen löst und auch mit Dividendenzahlungen umgehen kann.

Die Routinen wurden nach jedem Entwicklungsschritt anhand von Beispielen aus der Fachliteratur getestet. Die Genauigkeit der erzeugten Resultate legt die Vermutung nahe, dass die Programme zuverlässig arbeiten.

Die Implementierung ist ohne grosse Mühe zu bewerkstelligen. Die Routinen zeichnen sich durch Stabilität aus und nehmen keine lange Rechenzeit in Anspruch. Auf einfache Weise lassen sich mit den Verfahren aufschlussreiche Eindrücke über das Verhalten amerikanischer Optionen bei Dividendenzahlungen gewinnen. Die Routinen können optimiert werden, indem die Annahme homogener Randbedingungen fallen gelassen wird und stattdessen zutreffendere inhomogene Randbedingungen implementiert werden. Weiter wäre es möglich Konvergenz zu beschleunigen, indem anstelle des Gauss-Seidel-Verfahrens mit einer tatsächlichen Überrelaxation gearbeitet würde.

Inhaltsverzeichnis

I Abbildungs- und Tabellenverzeichnis

II Notationsverzeichnis

S/s	Preis des Basiswerts
K	Ausübungspreis / Strike-Preis
T	(Rest)Laufzeit einer Option
t	Laufende Zeit
V	Fairer Wert einer Option
V_C, V_P	Fairer Preis einer Call- oder Put-Option (unabhängig des Stils)
V_C^{Eur}, V_P^{Eur}	Fairer Preis einer europäischen Call- oder Put-Option
V_C^{Am}, V_P^{Am}	Fairer Preis einer amerikanischen Call- oder Put-Option
$(S_T - K)^+$	$\max\{S_T - K, 0\}$ Auszahlungsfunktion einer Call-Option
$(K - S_T)^+$	$\max\{K - S_T, 0\}$ Auszahlungsfunktion einer Put-Option
$g(s)$	Auszahlungsfunktion
r	Risikoloser stetiger Zinssatz
σ	Volatilität des Basiswerts
q	Stetige Dividendenrendite
$\delta_h^+(f)$	Vorwärtsdifferenzenquotient
$\delta_h^-(f)$	Rückwärtsdifferenzenquotient
$\delta_h\ (f)$	Zentraler Differenzenquotient
G	Diskretisierungsintervall in der Variable s
$s_{links}, s_{rechts,}$	Linker bzw. rechter Randpunkt von G
N	Anzahl der inneren Diskretisierungspunkte in s
M	Anzahl der zeitlichen Diskretisierungspunkte vor Laufzeit-ende

w	Näherungswert für den fairen Preis einer Option
h	Maschenweit der Diskretisierung in s
$\mathcal{G}_s$	Diskretisierungsgitter für den Basiswert s
k	Maschenweite der Diskretisierung in der Zeit
$\mathcal{G}_t$	Diskretisierungsgitter für die Zeit
θ	Parameter des Theta-Verfahrens
A, B, C	Tridiagonalmatrix A und Hilfsmatrizen
ω	Relaxationsparameter
$S_f(t)$	Freier Randwert / Optimaler Ausübungspunkt einer amerikanischen Option
D	Betrag einer diskreten Dividendenzahlung des Basiswerts
t_D	Zeitpunkt einer Dividendenzahlung / ex-Datum
t_D^-	Zeitpunkt unmittelbar vor einer Dividendenzahlung
t_D^+	Zeitpunkt unmittelbar nach einer Dividendenzahlung

1 Einleitung

1.1 Problemstellung

Heutzutage ist der Derivativhandel ein wichtiger Teil des Finanzmarktes. Optionen werden von Investoren zum Zweck der Absicherung, Spekulation oder Arbitrage eingesetzt. Optionen sind auch Bestandteile vieler strukturierter Produkte, die in den vergangenen Jahren grosse Bedeutung gewonnen haben. Die Ermittlung der fairen Preise von Optionen ist darum ein wesentliches Thema, mit dem sich die Finanzmathematik beschäftigt. Zur Bewertung von Optionen europäischen Stils kommt verbreitet das Modell von Black, Scholes (Black & Scholes, 1973) und Merton (Merton, 1973) zur Anwendung. Grösser und bedeutender ist jedoch der Markt von Optionen amerikanischen Stils. (Hull, 2012, S. 7) Das vorzeitige Ausübungsrecht von amerikanischen Optionen verursacht jedoch Schwierigkeiten bei ihrer Bewertung, weil der Wert der Option von einer optimalen Ausübungsstrategie abhängt. Aufgrund dieses freien Randwertproblems ist eine analytische Bewertung amerikanischer Optionen im Black-Scholes-Modell nicht möglich. Im Weiteren berücksichtigt das Modell keine Dividendenzahlungen des Basiswerts, während in der Praxis Dividenden eine grosse Rolle spielen. Die Bewertung von Optionen, deren Basiswerte während der Laufzeit diskrete Dividenden ausschütten, beinhaltet somit eine weitere Komplikation. Es wurden verschiedene numerische Verfahren entwickelt, welche trotzdem eine näherungsweise Optionsbewertung erlauben. In der vorliegenden Arbeit wird mit der Finiten-Differenzen-Methode eines dieser numerischen Verfahren untersucht und auf die Bewertung von europäischen sowie amerikanischen Optionen unter Berücksichtigung von Dividendenausschüttungen des Basiswerts angewendet.

1.2 Zielsetzung

Ziel der Arbeit ist es, in einem ersten Schritt das oben skizzierte Bewertungsproblem für amerikanische Optionen auf einen Basiswert mit und ohne diskrete Dividenden zu beschreiben. Anschliessend soll erörtert werden, wie die Finite-Differenzen-Methode auf die Aufgabe angewendet werden kann. Das numerische Verfahren soll alsdann in MATLAB® implementiert und die Korrektheit der damit erzeugten Ergebnisse anhand von Beispielen aus

der Fachliteratur getestet werden. Auf eine Analyse des Konvergenzverhaltens wird jedoch verzichtet.

1.3 Methodik und Aufbau der Arbeit

Anhand einschlägiger Fachliteratur werden die theoretischen Grundlagen erarbeitet. In einem ersten Schritt wird dann das Konzept der Optionsbewertung unter Anwendung der Finiten-Differenzen-Methode beschrieben. Die gewonnenen Erkenntnisse werden anschliessend angewendet, um europäische Optionen zu bewerten. Hiernach wird mit dem Successive Over-Relaxation-Verfahren ein Algorithmus zur numerischen Lösung linearer Gleichungssysteme vorgestellt, bevor auf die wesentliche Herausforderung der Bewertung von amerikanischen Optionen eingegangen wird. Das Problem der freien Randwerte wird erörtert und mit dem Projected Successive Over-Relaxation-Verfahren ein möglicher Lösungsansatz vorgestellt. Im Anschluss an die Bewertung amerikanischer Optionen ohne Berücksichtigung von Dividendenzahlungen werden die Auswirkungen diskreten Dividendenzahlungen diskutiert. Schliesslich werden europäische und amerikanische Optionen mit Berücksichtigung diskreter Dividendenzahlungen bewertet.

2 Finanzoptionen

2.1 Die verschiedenen Arten von Optionen

Der Käufer einer **Option** erwirbt das einmalige Recht, nicht aber die Verpflichtung, einen Basiswert S zu einem im Voraus festgelegten Ausübungspreis K am oder bis zum Verfallstag zu kaufen beziehungsweise (an den Stillhalter der Option) zu verkaufen.

Eine Option, die ein Kaufrecht beinhaltet, wird **Call-Option** (deutsch: Kaufoption) genannt. Gewährt die Option ein Verkaufsrecht, handelt es sich um eine **Put-Option** (Verkaufsoption).

Bei einer Option **europäischen Stils** ist der Ausübungszeitraum auf einen einzigen zukünftigen Zeitpunkt am Laufzeitende der Option beschränkt.

Bei Optionen **amerikanischen Stils** hingegen erstreckt sich der Ausübungszeitraum über die ganze Laufzeit T der Option.

Die Finanzindustrie hat noch weitere exotische Optionsstile hervorgebracht, wie zum Beispiel bermudische oder asiatische Optionen. Im Folgenden werden jedoch nur europäische und amerikanische Optionen auf Aktien behandelt.

2.2 Innerer Wert und Zeitwert von Optionen

Ein Optionshalter wird von seinem Ausübungsrecht nur dann Gebrauch machen, wenn er dadurch einen Gewinn erzielen kann. Da er keiner Ausübungspflicht untersteht, wird er andernfalls von einer Ausübung absehen. Ein Gewinn erzielt der Halter einer Call-Option, wenn der Basiswert S zum Ausübungszeitpunkt über dem Ausübungspreis K liegt. Das Ausüben einer Put-Option führt zu einem Gewinn, wenn bei der Ausübung der Basiswert S den Strikepreis K unterschreitet.
Der Wert der Option zum Fälligkeitszeitpunkt T kann somit durch die Auszahlungsfunktion $V_C(S_T, T)$ beziehungsweise $V_P(S_T, T)$ angegeben werden:

$$V_C(S_T, T) = \max\{S_T - K, 0\} = (S_T - K)^+ \qquad \textbf{2.1}$$

$$V_P(S_T, T) = \max\{K - S_T, 0\} = (K - S_T)^+ \qquad \textbf{2.2}$$

Abbildung 1 zeigt die Auszahlungsfunktionen von Call- beziehungsweise Put-Optionen.

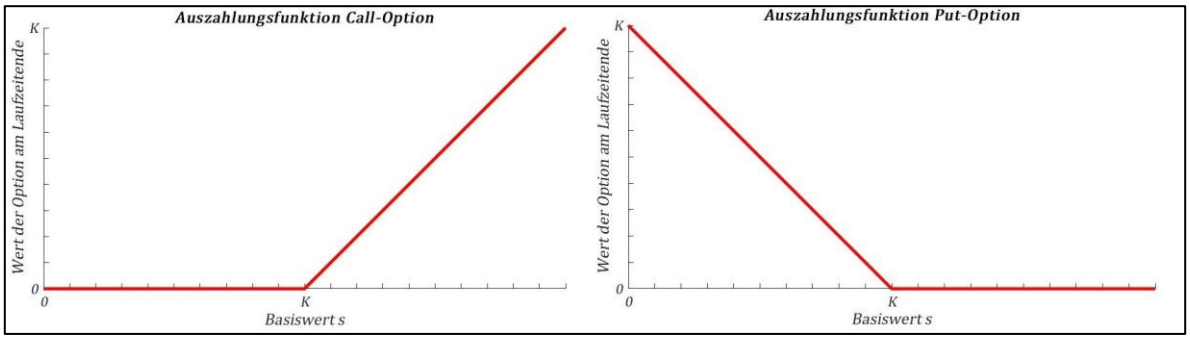

Abbildung 1 Auszahlungsfunktionen

Der Wert, den die Auszahlungsfunktion für einen aktuellen Kurswert des Basistitels annimmt, ist der innere (oder intrinsische) Wert der Option. Solange die Option ihr Laufzeitende noch nicht erreicht hat, weicht dieser vom tatsächlichen Optionspreis ab. Diese Abweichung ist der Zeitwert der Option.

2.3 Untere und obere Preisschranken europäischer Optionen

Unter Annahme, dass keine Dividenden gezahlt werden, lassen sich für europäische Kontrakte einfache Ober- und Untergrenzen des Optionswerts herleiten. Die Überlegungen erfolgen auf dem Paradigma der Arbitragefreiheit. Sie basieren also auf der Grundannahme, dass Marktpreisanpassungen Möglichkeiten zum Erzielen von sofortigen, risikolosen Gewinnen über längere Zeit verhindern. (Merton, 1973)

Wertuntergrenze einer europäischen Call-Option:

Es lege ein Investor ein Portefeuille A an, welches aus einem risikolosen Zero-Bond mit Kapitalrückzahlung von K bei Laufzeitende und einer Call-Option besteht. Der Kaufpreis der Option beträgt V_C^{Eur} und für den Zero-Bond ist Ke^{-rT} aufzuwenden. Am Ende der Laufzeit von Option und Bond kann der Investor die Aktie ohne weitere Aufwendungen erwerben.

Das Portefeuille A hat am Ende den gleichen Wert wie die Aktie. Zum Zeitpunkt der Investition muss also das Portefeuille A ebenso mindestens dem Wert S_t der Direktanlage in die Aktie entsprechen, damit sich keine Arbitragemöglichkeit ergibt: $V_C^{Eur} + Ke^{-r(T-t)} \geq S_t$ Es ergibt sich daraus die Untergrenze: $V_C^{Eur} \geq S_t - Ke^{-r(T-t)}$.

Wertobergrenze einer europäischen Call-Option:

Eine Kaufoption kann zu keinem Zeitpunkt mehr wert sein als der Basiswert S_t. Ansonsten kann durch Kauf des Basiswerts und gleichzeitigem Verkauf der Call-Option ein Arbitragegewinn erzielt werden.

Wertuntergrenze einer europäischen Put-Option:

Ein Investor sichere seine Aktie mit dem gleichzeitigen Kauf einer Put-Option ab. Wenn die Option endet, hat dieses Portefeuille B einen Wert von $max\{S_T, K\}$. Denn falls $S_T < K$, wird die Option ausgeübt und die Aktie zu K verkauft. Gilt $S_T > K$, ist die Option wertlos, aber der Aktienwert übertrifft K. Damit die Prämisse der Arbitragefreiheit eingehalten ist, muss das Portefeuille B zum Investitionszeitpunkt mindestens einem risikolosen Zero-Bond mit Kapitalrückzahlung von K bei Laufzeitende entsprechen. Es folgt $V_P^{Eur} + S_t \geq Ke^{-r(T-t)}$ beziehungsweise $V_P^{Eur} \geq Ke^{-r(T-t)} - S_t$.

Wertobergrenze einer europäischen Put-Option:

Die Put-Option erreicht bei $S = 0$ ihr Wertmaximum. Und zwar entspricht dieses dem Barwert des Ausübungspreises: $V_P^{Eur}(0,t) = Ke^{-r(T-t)}$. Dies ist somit auch die globale Wertobergrenze der Put-Option: $V_P^{Eur} \leq Ke^{-r(T-t)}$

Tabelle 1 fasst die obigen Überlegungen zusammen.

	Call-Option	Put-Option
Untergrenze	$V_C^{Eur} \geq S_t - Ke^{-r(T-t)}$	$V_P^{Eur} \geq Ke^{-r(T-t)} - S_t$
Obergrenze	S_t	$V_P^{Eur} \leq Ke^{-r(T-t)}$

Tabelle 1 Untere und obere Preisschranken europäischer Optionen

3 Das Bewertungsmodell nach Black und Scholes

Der Preis einer Option hängt von den Merkmalen des Kontrakts, dem Basiswert und den weiteren Marktbedingungen ab. 1973 gelang es Fischer Black und Myron Scholes (Black & Scholes, 1973) sowie Robert Merton (Merton, 1973) mit ihren Publikationen den bisher wichtigsten Meilenstein auf dem Gebiet der Optionsbewertung zu legen. Ihr finanzmathematisches Modell und die daraus entwickelte Bewertungsformel erlaubten es erstmals, den fairen Preis einer europäischen Finanzoption zu bestimmen. Das Modell bildet die Grundlage für viele weiterführende finanzmathematische Anwendungen. Es gilt heutzutage als Referenzmodell für die Bewertung von Finanzderivaten. Die Theorie wurde auf der Annahme der Arbitragefreiheit entwickelt. Prämisse dieser Annahme ist unter anderem ein friktionsloser und effizienter Markt. Weiter geht das Modell davon aus, dass die logarithmischen Aktienrenditen normalverteilt sind und Aktienkurse einer Zufallsbewegung folgen. Alles in Allem wurden folgende Modellannahmen getroffen (Hull, 2000, S. 299-325):

- Arbitrage ist nicht möglich.
- Es fallen keine Gebühren, Steuern oder sonstige Transaktionskosten an.
- Der Markt ist informationseffizient: Jede Partei hat sofortigen Zugang zu allen wesentlichen Informationen.
- Kein einzelner Marktteilnehmer kann den Marktpreis beeinflussen.
- Risikolose Anlagen und Kredite sind für alle Marktteilnehmer jederzeit uneingeschränkt und in beliebiger Höhe verfügbar.
- Es gibt einen stetigen risikofreien Zinssatz $r > 0$, der für alle Kreditnehmer und Laufzeiten gleich ist und sich nicht ändert.
- Alle Finanzinstrumente sind zu jeder Zeit in beliebiger Menge verfügbar und können in beliebig kleine Einheiten aufgeteilt und gehandelt werden.
- Der Leerverkauf von Finanzinstrumenten ist uneingeschränkt möglich.
- Die Volatilität σ des Basiswert ist konstant.
- Der Basiswert zahlt während der Laufzeit der Option keine Dividenden.
- Die Aktienkurse folgen einer geometrischen brownschen Bewegung.

Gewichtige Kritikpunkte des ursprünglichen Modells betreffen die Annahme der konstanten Volatilität und die Modellierung der Aktienkurse.

Während das Modell von einer konstanten Volatilität σ ausgeht, zeigen ex-post-Berechnungen der impliziten Volatilität ein anderes Bild. Die implizite Volatilität wird aus tatsächlichen Marktpreisen der Optionen unter Anwendung des Modells errechnet. Es zeigt sich, dass Optionen auf denselben Basiswert bei verschiedenen Ausübungspreisen zu unterschiedlichen Volatilitäten führen, was der Modellannahme widerspricht. (Rubinstein, 1994)

Der vereinfachten Abbildung der Aktienkursentwicklung widersprechen empirische Untersuchungen der tatsächlichen Renditeverteilung. Es ist allgemein bekannt, dass die Normalverteilung aufgrund ihrer Wölbung die Extremwerte zu wenig gewichtet. Dies führt dazu, dass die Wahrscheinlichkeit für das Auftreten extremer Renditewerte unterschätzt wird.

Erweiterungen des ursprünglichen Modells erlauben es, einige Annahmen zu lockern und die genannten Defekte zu entschärfen. So lässt sich der einfache Fall von stetigen deterministischen Dividendenzahlungen mit dem Modell auch abbilden. An Stelle einer konstanten Volatilität kann auf ein stochastisches Volatilitätsmodell zurückgegriffen werden, was im Constant Elasticity of Variance-Model (Cox & Ross, 1976) oder im Local Volatility-Model (Derman & Kani, 1994) gemacht wird.

Weiter lässt sich das Modell mittels computergestützter numerischer Verfahren auf andere Problemstellungen anwenden. Obwohl das Modell nur die Bewertung von europäischen Kontrakten erlaubt, kann die darauf basierende Finite-Differenzen-Methode verwendet werden, um Optionen amerikanischen Stils zu bewerten.

4 Optionsbewertung mit der Finiten-Differenzen-Methode

Für die Bewertung einer europäischen Option hält das Black-Scholes-Modell die folgende lineare parabolische Differentialgleichung zweiter Ordnung bereit:

$$\begin{cases} \partial_t V + \frac{1}{2}\sigma^2 s^2 \partial_{ss} V + (r - q)s\partial_s V - rV = 0 & in\ \mathbb{R}^+ \times [0, T[\\ V(s, T) = g(s) & in\ \mathbb{R}^+ \end{cases} \quad \textbf{4.1}$$

In der Black-Scholes-Gleichung 4.1 ist V eine Funktion der Laufzeit t des Kontraktes. Der Anschaulichkeit halber wird im weiteren Verlauf die Funktion v als Funktion in der ***Rest-***laufzeit betrachtet (Mit $t = 0$ im Verfallzeitpunkt und $t = T$ im Bewertungszeitpunkt). Aus der Kettenregel ergibt sich somit die neue Gleichung:

$$\begin{cases} \partial_t v - \frac{1}{2}\sigma^2 s^2 \partial_{ss} v - (r - q)s\partial_s v + rv = 0 & in\ \mathbb{R}^+ \times]0, T] \\ v(s, 0) = g(s) & in\ \mathbb{R}^+ \end{cases}$$

Zur weiteren Vereinfachung werden die Koeffizienten substituiert durch:

$$\begin{aligned} a(s) &:= -\frac{1}{2}\sigma^2 s^2 \\ b(s) &:= -(r - q)s \\ c(s) &:= r \end{aligned}$$

Damit verkürzt sich die Gleichung zu:

$$\begin{cases} \partial_t v - a(s)\partial_{ss} v - b(s)\partial_s v + c(s)v = 0 & in\ \mathbb{R}^+ \times]0, T] \\ v(s, 0) = g(s) & in\ \mathbb{R}^+ \end{cases} \quad \textbf{4.2}$$

Die Gleichung 4.2 setzt sich aus drei partiellen Ableitungen der Funktion $v(s, t)$ zusammen, welche jeweils durch einen Differenzenquotienten approximiert werden können. Dazu ersetzt man die erste Ableitung nach s, die durch den infinitesimalen Differentialquotienten $\partial s V$ gegeben ist, durch einen finiten Differenzenquotienten.

$$\partial_s V(s_0, t) \coloneqq \lim_{\Delta s \to 0} \frac{V(s_0 + \Delta s, t) - V(s_0, t)}{\Delta s}$$

Je nach dem in welche Richtung vom Punkt x_0 aus differenziert wird, unterscheidet man zwischen dem Vorwärtsdifferenzenquotient

$$\delta_h^+(f) \coloneqq \frac{f(x+h) - f(x)}{h}$$

und dem Rückwärtsdifferenzenquotient

$$\delta_h^-(f) \coloneqq \frac{f(x) - f(x-h)}{h}$$

Für die Quotientenbildung werden dabei jeweils die zwei Punkte $x_0$und $\mathrm{x}_0 + \mathrm{h}$ beziehungsweise x_0 und $x_0 - h$ verwendet. Eine genauere Annäherung an die Ableitung an der Stelle x_0 liefert nach dem Satz von Taylor der zentrale Differenzenquotient als Kombination der obigen Näherungen. Dabei werden zur Bestimmung des Funktionswerts an der Stelle s_0 die zwei Punkte $x_0 - h$ und $x_0 + \mathrm{h}$ verwendet.

$$\delta_h(f) \coloneqq \frac{f(x+h) - f(x-h)}{2h}$$

Abbildung 2 stellt die drei Differenzenquotienten schematisch dar und verdeutlicht die genauere Annäherung der Sekantensteigung von $\delta_h(f)$ an die Tangentensteigung im Punkt x_0.

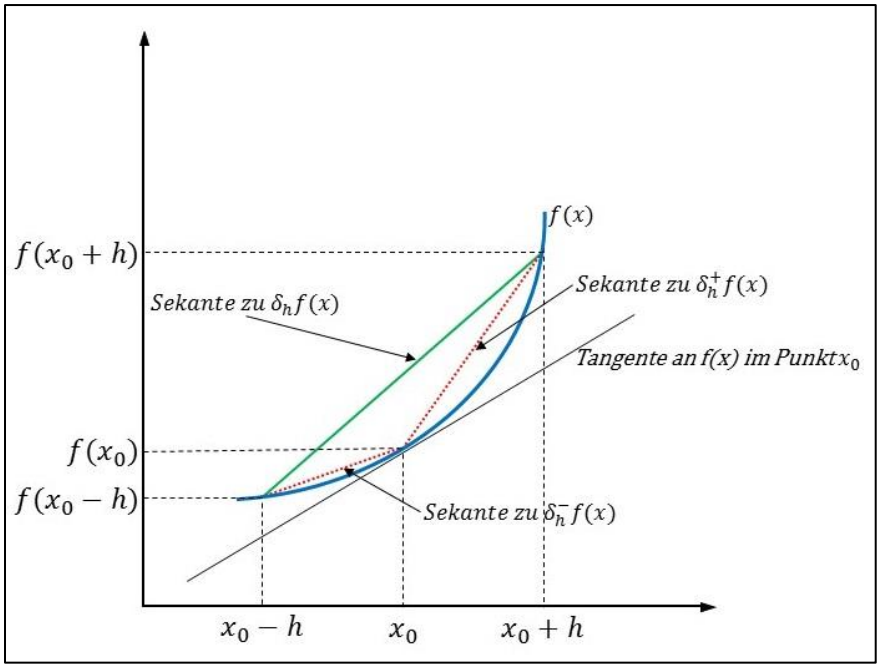

Abbildung 2 Skizze der Differenzenquotienten in Anlehnung an (Wilmott et al., 2009, S. 137)

Auch die zweite Ableitung einer Funktion lässt sich mit dem zentralen Differenzenquotienten annähern:

$$\delta_h^2 f(x) := \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}$$

Damit eine lineare Differentialgleichung unter Verwendung von finiten Differenzen diskretisiert werden kann, muss sie lokalisiert werden. Dazu wird die Definitionsmenge von der Menge der positiven reellen Zahlen $\mathbb{R}^+$auf ein frei wählbares Intervall G eingeschränkt. Für die Randpunkte, die das Intervall begrenzen, müssen die Funktionswerte bekannt sein oder festgelegt werden. Diese beiden Funktionswerte bilden die Randbedingungen. Anschliessend wird das Intervall Gin eine endliche Teilmenge von Gitterpunkten unterteilt.
Gleichung 4.2 wird in der Variable s lokalisiert, indem die Definitionsmenge auf das Intervall $G = [s_{links}, s_{rechts}]$ beschränkt wird. Es gibt verschiedene Verfahren um die Funktionswerte an den Randpunkten $v(s_{links,}t)$ und $v(s_{rechts,}t)$ zu bestimmen. Der Einfachheit halber werden in der vorliegenden Arbeit jedoch homogene Randbedingungen verwendet, das heisst es wird angenommen, dass die Funktionswerte an den Randpunkten s_{links} und s_{rechts} null sind.

Auch wenn diese Annahme nicht zutrifft, kann der Fehler der Approximation auf einem kleinen Teilintervall $\tilde{G} \subset G$ beliebig klein gehalten werden. Gesucht ist nun eine Funktion $w(s,t)$, welche die gewählten Randbedingungen erfüllt und v im Intervall G annähert. Zusammen mit der Auszahlungsfunktion, welche im Zeitpunkt $t = 0$ für alle s bekannt ist, gelten dann folgende Bedingungen:

$$\begin{cases} \partial_t w + a(s)\partial_{SS} w + b(s)\partial_S w + c(s)w = 0 & in \quad G \times]0,T] \\ w(s_{links}, t) = 0 & in \quad]0,T] \\ w(s_{rechts}, t) = 0 & in \quad]0,T] \\ w(s,0) = g(s) & in \quad]0,T] \end{cases}$$

Nun wird das Intervall G in eine endliche Teilmenge von $N+2$ Gitterpunkten unterteilt. Dazu wird ein äquidistantes Gitter $\mathcal{g}_s$ eingeführt. $\mathcal{g}_s$ besteht aus den beiden Randpunkten s_{links} und s_{rechts}.sowie N Punkten, die innerhalb des Intervalls G liegen. In einem äquidistanten Gitter ist der Abstand der Punkte, die Maschenweite h, gegeben.

$$h = \frac{s_{rechts} - s_{links}}{N+1}$$

Indem man die partiellen Ableitungen nach s in der Gleichung 4.2 durch die zentralen Differenzenquotienten ersetzt, erhält man für alle inneren Gitterpunkte s_i für $i = 1, \dots, N$ die Näherung:

$$\partial_t w(s_i,t) + a(s_i)\frac{w(s_{i-1},t) - 2w(s_i,t) + w(s_{i+1},t)}{h^2} + b(s_i)\frac{-w(s_{i-1},t) + w(s_{i+1},t)}{2h} + c(s_i)w(s_i,t) \approx 0$$

Durch Ersetzten aller Funktionen $w(s_i,t)$ durch Näherungsfunktionen $w_i(t)$ und nach Umformung schreibt sich die obige Gleichung als:

$$w_i'(t) + \left(\frac{a(s_i)}{h^2} - \frac{b(s_i)}{2h}\right)w_{i-1}(t) + \left(-\frac{2a(s_i)}{h^2} + c(s_i)\right)w_i(t) + \left(\frac{a(s_i)}{h^2} + \frac{b(s_i)}{2h}\right)w_{i+1}(t) = 0 \qquad \textbf{4.3}$$

Der Übersichtlichkeit halber werden die bekannten Koeffizienten substituiert durch:

$$\alpha_i \coloneqq \frac{a(s_i)}{h^2} - \frac{b(s_i)}{2h} \quad \beta_i \coloneqq -\frac{2a(s_i)}{h^2} + c(s_i) \quad \gamma_i \coloneqq \frac{a(s_i)}{h^2} + \frac{b(s_i)}{2h}$$

Die Gleichung 4.3 schreibt sich nun verkürzt:

$$w_i' + \alpha_i w_{i-1}(t) + \beta_i w_i(t) + \gamma_i w_{i+1}(t) = 0$$

Mit den Randbedingungen $w_0(t) = 0$ und $w_{N+1}(t) = 0$ lässt sich somit für alle inneren Gitterpunkte mit $i = 1, .., N$ das folgende Gleichungssystem aufstellen:

$$\begin{cases} w_1'(t) + \beta_1 w_1(t) + \gamma_1 w_2(t) = 0 \\ w_2'(t) + \alpha_2 w_1(t) + \beta_2 w_2(t) + \gamma_2 w_3(t) = 0 \\ \quad \dots = 0 \\ w_{N-1}'(t) + \alpha_{N-1} w_{N-2}(t) + \beta_{N-1} w_{N-1}(t) + \gamma_{N-1} w_N(t) = 0 \\ w_N'(t) + \alpha_N w_{N-1}(t) + \beta_N w_N(t) = 0 \end{cases} \quad \textbf{4.4}$$

Dieses System aus N Gleichungen und N Unbekannten kann in Matrixschreibeweise geschrieben werden:

$$w'(t) + Aw(t) = 0 \quad \textbf{4.5}$$

Dabei ist A eine NxN-Tridiagonalmatrix bestehend aus den Teilmatrizen M_a, M_b und M_c

$$A \coloneqq M_a + M_b + M_c \coloneqq \begin{pmatrix} \beta_1 & \gamma_1 & & & & & \\ \alpha_2 & \beta_2 & \gamma_2 & & & & \\ & \alpha_3 & \beta_3 & \gamma_3 & & & \\ & & & \ddots & & & \\ & & & \alpha_{N-2} & \beta_{N-2} & \gamma_{N-2} & \\ & & & & \alpha_{N-1} & \beta_{N-1} & \gamma_{N-1} \\ & & & & & \alpha_N & \beta_N \end{pmatrix} \quad \textbf{4.6}$$

$$M_a := \frac{1}{h^2}\begin{pmatrix} -2a & a & & & & & & \\ a & -2a & a & & & & & \\ & a & -2a & a & & & & \\ & & a & & & & & \\ & & & a & -2a & a & & \\ & & & & a & -2a & a & \\ & & & & & a & -2a \end{pmatrix}$$

$$M_b := \frac{1}{2h}\begin{pmatrix} 0 & b & & & & & & \\ -b & 0 & b & & & & & \\ & -b & 0 & b & & & & \\ & & & \ddots & & & & \\ & & & -b & 0 & b & & \\ & & & & -b & 0 & b & \\ & & & & & b & 0 \end{pmatrix}$$

$$M_c := \begin{pmatrix} c & 0 & & & & & & \\ 0 & c & 0 & & & & & \\ & 0 & c & 0 & & & & \\ & & & \ddots & & & & \\ & & & 0 & c & 0 & & \\ & & & & 0 & c & 0 & \\ & & & & & 0 & c \end{pmatrix}$$

Der Wert der Option bei Laufzeitende ist durch die bekannte Auszahlungsfunktion g von s (siehe Gleichungen 2.1 und 2.2) gegeben: $w(0) = w(s, 0) = g(s)$.

$$g = \begin{pmatrix} g(s_1) \\ g(s_2) \\ \vdots \\ g(s_3) \\ g(s_4) \end{pmatrix}$$

In Matrixschreibweise ist das Gleichungssystem mit Anfangsbedingung beschrieben durch:

$$\begin{cases} w'(t) + Aw(t) = 0 \\ w(0) = g \end{cases} \qquad \textbf{4.7}$$

Als nächstes ist die Diskretisierung in der der Zeit vorzunehmen, um ein volldiskretes Verfahren zu erhalten. Die Restlaufzeit beschränkt das Zeitintervall auf $[T, 0]$. Dieses wird in $M + 1$ Zeitpunkte unterteilt, so dass ein äquidistantes Gitter $\mathcal{g}_t$ entsteht. Die Maschenweite des Gitters beträgt k.

$$k = \frac{T}{M}$$

Indem die erste Ableitung nach der Zeit in der Gleichung 4.5 an jedem Gitterpunkt t_j für $j = 1, \dots, M + 1$durch den Vorwärtsdifferenzenquotient δ_k^+ ersetzt wird, ergibt sich:

$$\frac{w(t_j + k) - w(t_j)}{k} + Aw(t_j) \approx 0 \qquad \textbf{4.8}$$

Jede Funktion $w(t_j)$ wird durch ihre entsprechende Näherungsfunktion $w_j$ersetzt:

$$\frac{w_{j+1} - w_j}{k} + Aw_j = 0$$

Aus dieser Beziehung lässt sich die folgende Rechenvorschrift ableiten, mit der w_{j+1} explizit aus dem vorhergehenden Wert w_j errechnet werden kann.

$$w_{j+1} = (1 - kA)w_j$$

Es handelt sich um das **explizite Euler-Verfahren**. Bei Verwendung des Rückwärtsdifferenzenquotienten in der Gleichung 4.3 ergibt sich:

$$\frac{w(t_j) - w(t_j - k)}{k} + Aw(t_j) \approx 0$$

Oder mit w_j als Näherungsfunktionen für $w(t_j)$:

$$\frac{w_{j+1} - w_j}{k} + Aw_{j+1} = 0 \qquad \textbf{4.9}$$

Durch Umstellen der Gleichung 4.9 wird offensichtlich, dass hier der Wert w_{j+1} nicht nur von w_j, sondern auch von w_{j+1} selbst abhängig ist. Denn der Wert von w_{j+1} errechnet sich aus dem Wert w_j und einem Wertzuwachs Δw, welcher hier $k(-A)w_{j+1}$ entspricht.

$$w_{j+1} = w_j + k(-A)w_{j+1}$$

Dies ist das **implizite Euler-Verfahren**. Das implizite Euler-Verfahren hat den Vorteil, dass es unbedingte Stabilität aufweist, während das explizite Euler-Verfahren nur stabil ist, wenn bestimmte Restriktionen bezüglich Zeitschrittgrösse eingehalten werden. Das explizite Verfahren ist andererseits einfacher in Computeranwendungen zu implementieren. Die beiden Rechenvorschriften lassen sich im Theta-Verfahren mittels Linearkombination miteinander verbinden. Dies gelingt durch Einführung eines Parameters $\theta \, \epsilon [0,1]$.

$$w_{j+1} = w_j + k((1-\theta)(-A)w_j + \theta(-A)w_{j+1}) \qquad \textbf{4.10}$$

Wobei die Wahl von $\theta = 0$ zum (voll) impliziten und $\theta = 1$ zum (voll) expliziten Euler-Verfahren führen. Das Theta-Verfahren nach Crank-Nicholson mit $\theta = 0.5$ ist wie das voll implizite Verfahren unbedingt stabil, liefert aber genauere Ergebnisse als dieses und hat darum einen besonderen Stellenwert. (Wilmott et al., 2009, S. 159).

Eine Umformung der Gleichung 4.10 mit Hilfe der Einheitsmatrix $\mathbb{I}$ führt zu:

$$(\mathbb{I} + k\theta A)w_{j+1} = (\mathbb{I} - k(1-\theta)A)w_j \qquad \textbf{4.11}$$

Nach Einführung der Hilfsmatrizen B und C, lässt sich das Gleichungssystem in eine einfachere Form bringen:

$$w_{j+1} = B^{-1}Cw_j \qquad \textbf{4.12}$$

Dabei werden die Hilfsmatrizen folgendermassen definiert:

$$B \coloneqq \mathbb{I} + k\theta A \; und \; C \coloneqq \mathbb{I} - k(1-\theta)A \qquad \textbf{4.13}$$

Unter Hinzunahme der Randbedingungen und unter Verwendung der Anfangsbedingung $w(s,0) = g(s)$ kann mittels obiger Gleichung 4.12 eine Routine zur Bewertung von europäischen Optionen implementiert werden.

5 Bewertung europäischer Optionen ohne Berücksichtigung diskreter Dividenden

5.1 Einleitung

Die im vorherigen Kapitel erarbeitete Methode wird verwendet, um einen Algorithmus zur Bewertung von europäischen Optionen zu entwickeln. Die Funktion `Eur()` soll einen Vektor mit $N+2$ Basiswerten und den dazugehörigen Optionswerten im Zeitpunkt $t = 0$ ausgeben. Aufgrund der oben erwähnten Vorteile des Crank-Nicholson-Verfahrens soll dieses, sprich $\theta = 0.5$, gewählt werden. Im Weiteren soll die Diskretisierung ebenfalls auf einem äquidistanten Gitter mit homogenen Randbedingungen erfolgen. Die Koeffizienten der Black-Scholes-Differentialgleichung werden in der Routine bereits hinterlegt. Ebenso die Auszahlungsfunktionen sowohl für Kauf- als auch für Verkaufsoptionen. Die Wahl zwischen Call und Put erfolgt durch Übergabe der Variable *CallorPut*. Weitere Eingabeargumente der Funktion sind die Modellparameter (K, T, σ, r und q) und die Diskretisierungsparameter (s_{links}, s_{rechts}, N und M).

5.2 Implementierung des Kernalgorithmus in MATLAB®

Mittels der übergebenen Parameter werden die Vektoren s und w erzeugt, die das Diskretisierungsgitter aufspannen. Anschlissend wird die tridiagonale Matrix A (siehe Gleichung 4.6) definiert. Mit Hilfe der Einheitsmatrix werden die Hilfsmatrizen B und C geschaffen (siehe Definition 4.13). In einer Rückwärtsiteration vom Laufzeitende aus wird dann der aktuelle

Wert der Option für aktuelle Kurse des Basiswerts berechnet. Dazu wird das Gleichungssystem 4.12 für jeden Zeitpunkt vor dem Laufzeitende gelöst und die Ergebnisse im Vektor *w* abgespeichert. Die ermittelten Werte werden dabei in jedem Iterationsschritt überschrieben. Schliesslich sind die Vektoren s und w noch mit den Randwerten zu ergänzen.

MATLAB®-Code der Funktion `Eur()`

```
function [s,w] = Eur(CallorPut,K,T,sigma,r,q,sl,sr,N,M)
theta = 0.5;
a = @(s)-0.5*sigma^2*s.^2;
b = @(s)-(r-q)*s;
c = @(s)r*s.^0;
if CallorPut == 'c'
   g = @(s)max(0,s-K);
elseif CallorPut == 'p'
   g = @(s)max(0,K-s);
end
h = (sr-sl)/(N+1);
s = (sl:h:sr)';
k = T/M;
M0 = spdiags(c(s(2:N+1)),0,N,N);
M1 = 1/(2*h)*spdiags([-b(s(3:N+2)),zeros(N,1),b(s(1:N))],-1:1,N,N);
M2 = 1/h^2*spdiags([a(s(3:N+2)),-2*a(s(2:N+1)),a(s(1:N))],-1:1,N,N);
A = M2+M1+M0; I = speye(N);
s = s(2:N+1);
B = I + k*theta*A; C = I - (1-theta)*k*A;
w = g(s);
for j = 1:M
    w = B\(C*w);
end
s=[sl;s;sr]; w=[0;w;0];
```

Der Code `Eur()` ist im *Anhang 1* mit Kommentaren versehen.

5.3 Numerische Beispiele

Beispiel 1

Gegeben sind die Modellparameter $K = 40$, $T = 0.5$, $\sigma = 0.2$ und $r = 0.1$. Der Wert der europäischen Put-Option soll für einen aktuellen Kurs des Basiswerts $s = 42$ berechnet werden. (Hull, 200, S. 316) Aus dem Vektor *w*, welchen `Eur('p',40,0.5,0.2,0.1,0,0,160,1023,800)` ausgibt, wird der Wert für $s = 42$ mittels einer MATLAB®-Funktion interpoliert.

Die Routine liefert $w(42{,}0.5) = 0.8084$ während die Berechnung mit der Black-Scholes-Formel in Excel einen Wert von 0.8086 ergibt.

Beispiel 2

Mit der Funktion `Eur()` soll der Wert einer europäischen Put-Option mit $K = 10$, $T = 1$, $\sigma = 0.6$, $r = 0.25$ und $q = 0.2$ dargestellt werden (Seydel, 2006, S. 163). Mit dem Aufruf der Funktion `[s,w]=Eur('p',10,1,0.6,0.25,0.2,0,40,1023,800)` werden die Vektoren erzeugt und in *Abbildung 3* graphisch dargestellt.

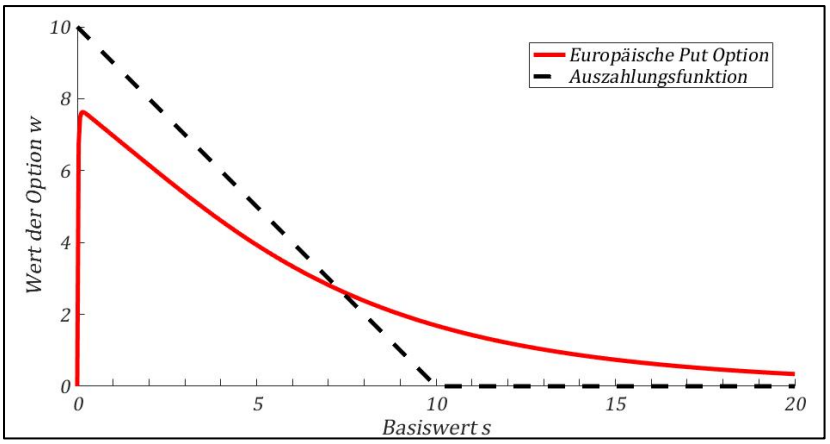

Abbildung 3 Approximierter Preis der Put-Option aus Beispiel 1

Aufgrund der homogenen Randbedingungen gibt die Näherung um die Stelle $s = 0$ keine sinnvollen Werte.

Beispiel 3

Es soll die Oberfläche einer Put-Option mit $K = 10$, $T = 1$, $\sigma = 0.3$, $r = 0.06$ und $q = 0$ dargestellt werden (Seydel, 2006, S. 163).

Aufbauend auf der Funktion `Eur()` wird die Routine `Eur3D()` implementiert. In der neuen Routine wird zusätzlich eine $N \times (M + 1)$-Matrix Z erzeugt, in der spaltenweise der Vektor w jedes Iterationsschrittes gespeichert wird. Der ganze Code `Eur3D()` kann im *Anhang 2* eingesehen werden. Mit `Eur3D('p',10,1,0.3,0.06,0,0,40,511,200)` wurde die *Abbildung 4* erstellt.

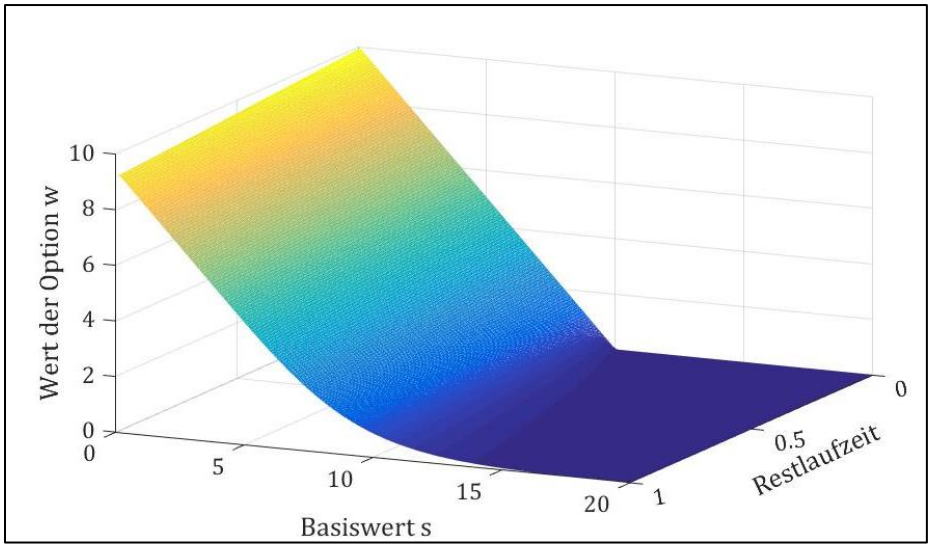

Abbildung 4 Oberfläche der Put-Option aus Beispiel 2

6 Splittingverfahren

6.1 Herleitung des Successive Over-Relaxation-Verfahrens

In der MATLAB®-Routine `Eur()` wird der Vektor w durch die Inversion der Matrix A gefunden. Dank ihrer speziellen Struktur ist die Matrix A nur dünnbesetzt. Denn als $N \times N$-Tridiagonalmatrix weisen nur ihre Hauptdiagonale und deren Nachbardiagonalen von null abweichende Werte auf. Es sind also von N^2-Einträgen lediglich $3N - 2$ von null verschieden. Dünnbesetzte Matrizen haben den Vorteil, dass Null-Elemente nicht abspeichert werden müssen. Die Inverse von A teilt diese nützliche Eigenschaft der Ursprungsmartix nicht und benötigt für die Speicherung somit bedeutend mehr Speicherplatz. In der MATLAB®-Routine `Eur()` werden die Matrix A beziehungsweise ihre Teilmatrizen *M0*, *M1* und *M2* als Sparse-Matrizen erstellt. Diese effiziente Speicherung spart Kapazität.

Es gibt Algorithmen, die lineare Gleichungssysteme $Ax = b$ für tridiagonale Matrizen sehr effizient lösen. (Wilmott et al., 2009, S. 147) Für computergestützte Verfahren sind vor allem iterative Methoden interessant, welche Konvergenzeigenschaften ausnützen. Im Unterschied zu direkten Verfahren nähert man sich dabei ausgehend von einer Startnäherung schrittweise dem exakten Wert an und bricht die Wiederholung ab, falls die Genauigkeit hoch genug ist.

Diese Methoden bieten sich bei der Verwendung von finiten Differenzen an, da das Gleichungssystem für viele dicht aufeinander folgende Zeitpunkte berechnet werden muss und so der Wert des vorherigen Zeitpunktes als zuverlässiger Schätzer verwendet werden kann. Im Folgenden wird das Successive Over-Relaxation-Verfahren (SOR-Verfahren) vorgestellt, welches sich für den Einsatz bei Optionsberechnungen bestens eignet, da es nach einer kleinen Adaption auch für die Bewertung von amerikanischen Optionen gebraucht werden kann.

Das SOR-Verfahren ist eine Verfeinerung des Gauss-Seidel-Verfahrens, welches wiederum auf dem Jacobi-Verfahren beruht.

Das Problem $Ax = b$ mit $A \epsilon \mathbb{R}^{N,N}$ und $b \epsilon \mathbb{R}^{N}$, bestehend aus einer bekannten Matrix A, dem bekannten Vektor b und dem gesuchten Vektor x, lässt sich als Gleichungssystem ausschreiben:

$$\begin{array}{lllll} A_{1,1}x_1 & + A_{1,2}x_2 & + \ldots & + A_{1,N}x_N & = b_1 \\ A_{2,1}x_1 & + A_{2,2}x_2 & + \ldots & + A_{2,N}x_N & = b_2 \\ & \ldots & & & = \ldots \\ A_{N,1}x_1 & + A_{N,2}x_2 & + \ldots & + A_{N,N}x_N & = b_N \end{array}$$

Durch Umstellen erhält man das Gleichungssystem:

$$\begin{array}{ll} x_1 = & \frac{1}{A_{1,1}}\left(b_1 - \left(A_{1,2}x_2 + \cdots + A_{1,N}x_N\right)\right) \\ x_2 = & \frac{1}{A_{2,2}}\left(b_2 - \left(A_{2,1}x_1 + \cdots + A_{2,N}x_N\right)\right) \\ \ldots & \ldots \\ x_N = & \frac{1}{A_{N,N}}\left(b_N - \left(A_{N,1}x_1 + \cdots + A_{N,N}x_{N-1}\right)\right) \end{array} \qquad \textbf{6.1}$$

Jedes Vektorelement von x ist nun durch die anderen Elemente von x sowie durch die bekannten Elemente von A und b bestimmt. Da alle Elemente von x unbekannt sind, wird eine Anfangsschätzung x^0 für x benötigt. Ausgehend von diesem Vektor kann anschliessend jedes Vektorelement von x berechnet und der so errechnete Vektor als neuer Schätzvektor verwendet werden. Gleichungssystem 6.1 lautet für den i-ten Iterationsschritt:

$$
\begin{aligned}
x_1^{i+1} = &\quad \frac{1}{A_{1,1}}\left(b_1 - \left(A_{1,2}x_2^i + \cdots + A_{1,N}x_N^i\right)\right)\\
x_2^{i+1} = &\quad \frac{1}{A_{2,2}}\left(b_2 - \left(A_{2,1}x_1^i + \cdots + A_{2,N}x_N^i\right)\right)\\
\cdots &\quad \cdots\\
x_N^{i+1} = &\quad \frac{1}{A_{N,N}}\left(b_N - \left(A_{N,1}x_1^i + \cdots + A_{N,N}x_{N-1}^i\right)\right)
\end{aligned}
$$

Die hochgestellte Zahl in der obigen Schreibweise indexiert den Iterationsschritt.

Daraus lässt sich die rekursive Folgevorschrift des **Jacobi-Verfahrens** ableiten:

$$x_n^{i+1} = \frac{1}{A_{n,n}}(b_n - \sum_{m=1}^{n-1} A_{n,m}x_m^i - \sum_{m=n+1}^{N} A_{n,m}x_m^i)$$

Es fällt auf, dass bei dieser Methode der Schätzungsvektor unverändert bleibt bis die letzte Gleichung sequentiell gelöst wurde. Bei der Gauss-Seidel-Methode wird eine Verfeinerung des oben beschriebenen Verfahren dadurch erreicht, dass der Schätzungsvektor $\tilde{x}$ laufend, also schon innerhalb eines Iterationsschrittes, angepasst wird. Im Iterationsschritt $i+1$ verändert sich der Schätzungsvektor $\tilde{x}$ schematisch wie folgt:

$$\tilde{x} = \begin{pmatrix} x_1^i \\ x_2^i \\ x_3^i \\ \vdots \\ x_{N-1}^i \\ x_N^i \end{pmatrix} \rightarrow x_1^{i+1} \rightarrow \tilde{x} = \begin{pmatrix} \boldsymbol{x_1^{i+1}} \\ x_2^i \\ x_3^i \\ \vdots \\ x_{N-1}^i \\ x_N^i \end{pmatrix} \rightarrow x_2^{i+1} \rightarrow \tilde{x} = \begin{pmatrix} \boldsymbol{x_1^{i+1}} \\ \boldsymbol{x_2^{i+1}} \\ x_3^i \\ \vdots \\ x_{N-1}^i \\ x_N^i \end{pmatrix} \rightarrow \boldsymbol{x_3^{i+1}} \rightarrow \cdots \rightarrow \tilde{x} = \begin{pmatrix} \boldsymbol{x_1^{i+1}} \\ \boldsymbol{x_2^{i+1}} \\ \boldsymbol{x_3^{i+1}} \\ \vdots \\ \boldsymbol{x_{N-1}^{i+1}} \\ x_N^i \end{pmatrix} \rightarrow x_N^{i+1}$$

Die Vorschrift zur Berechnung des n-ten Elements von x lautet für das **Gauss-Seidel-Verfahren** somit:

$$x_n^{i+1} = \frac{1}{A_{n,n}}(b_n - \sum_{m=1}^{n-1} A_{n,m}x_m^{i+1} - \sum_{m=n+1}^{N} A_{n,m}x_m^i)$$

Dank dem laufenden Anpassen des Schätzvektors konvergiert das Gauss-Seidel-Verfahren schneller als das Jacobi-Verfahren.

Weist eine Vorschrift monotone Konvergenz auf, wird in jedem Iterationsschritt der Wert eines Elements um ein Inkrement Δx in die Richtung der exakten Lösung korrigiert.

$$x_n^{i+1} = x_n^i + \Delta x$$

Die monotone Konvergenz kann nochmals beschleunigt werden, indem man den Korrekturterm Δx immer etwas weiter extrapoliert, als der Algorithmus vorgibt. Dies gelingt durch Einführung eines Relaxationsparameters ω.

$$x_n^{i+1} = x_n^i + \boldsymbol{\omega}\Delta x$$

Indem man nun zur Berechnung von Δx beziehungsweise $\overline{x_n^{i+1}}$ auf den Gauss Seidel Algorithmus zurückgreift erhält man:

$$x_n^{i+1} = x_n^i + \omega\left(\overline{x_n^{i+1}} - x_n^i\right) \equiv$$
$$x_n^{i+1} = (1-\omega)x_n^i + \omega\overline{x_n^{i+1}}$$

Auf diese Weise wird die neuere und bessere Information $\overline{x_n^{i+1}}$ stärker gewichten, als der alte Wert x_n^i. Im **SOR-Verfahren** lautet die Iterationsvorschrift entsprechend:

$$x_n^{i+1} = (1-\omega)x_n^i + \frac{\omega}{A_{n,n}}\left(b_n - \sum_{m=1}^{n-1} A_{n,m}x_m^{i+1} - \sum_{m=n+1}^{N} A_{n,m}x_m^i\right) \qquad \textbf{6.2}$$

Um eine beschleunigte Konvergenz zu erhalten ist $\omega > 1$ zu wählen. Für $\omega \in (1,2)$ konvergiert das SOR-Verfahren gegen die exakte Lösung des Gleichungssystems und benötigt bei optimal gewähltem ω weniger Iterationsschritte, um die gleiche Genauigkeit zu erzielen wie das Gauss-Seidel-Verfahren. (Günther & Jüngel, 2010, S. 208) Ob ein optimaler Wert

für ω existiert, hängt von den Eigenschaften der Matrix ab. Gewöhnlich wird dieser heuristisch ermittelt. (Wilmott et al., 2009, S. 153).

6.2 Implementation des SOR-Verfahrens

Die Rechenvorschrift 6.2 wird verwendet, um eine MATLAB®-Routine zu entwickeln, die ein Gleichungssystem $Ax = b$ nach dem unbekannten Vektor x auflöst. Der Funktion sor() müssen die Matrix A, der Vektor b sowie der Schätzvektor x^0 übergeben werden.
Die Iteration wird abgebrochen sobald eine willkürlich gewählte Fehlerschranke (es wird 10^{-12} angewandt) unterschritten wird. Die exakte Lösung der Aufgabenstellung ist unbekannt. Es ist darum nicht möglich, die Abweichung zum exakten Wert zu bestimmen. Die Genauigkeit der Lösung wird anhand der Änderung gegenüber des vorherigen Iterationsschrittes gemessen. Solange die grösste Änderung eines Vektorelements die Fehlerschranke überschreitet, wird die Operation wiederholt.
Die Wahl des optimalen Relaxationsparameters ist nicht trivial. Darum wird im weiteren Verlauf auf eine Überrelaxation verzichtet und das Gauss-Seidel-Verfahren ($\omega = 1$) angewendet.

MATLAB®-Code der Funktion SOR()

```
function x = SOR(A,x0,b)
Omega = 1;
schranke = 10^-12;
x = x0;
test = x0;
fehler = 10;
xneu = zeros(length(A));
while fehler > schranke
    for i = 1:length(A)
    xneu(i) = x(i) - (Omega/A(i,i))*(A(i,:) * x - b(i));
    x(i) = xneu(i);
    end
fehler = max(abs(x-test));
test = x;
end
```

Der ganze Code mit Kommentaren ist im *Anhang 3* zu finden.

Zum Test, ob die obige Funktion zuverlässig arbeitet soll *Beispiel 1* nachgerechnet werden. Diesmal mit wird in der Iterationsschlaufe der Routine `Eur()` zum Lösen des Gleichungssystems `SOR(B,w,C*w)` verwendet, statt `w = B\(C*w)`. Das Ergebnis nach Interpolation beträgt wiederum 0.8084. Die Funktion `SOR()` scheint zuverlässig zu arbeiten

7 Optionen amerikanischen Stils

7.1 Vorzeitige Ausübung

Eine amerikanische Option kann während der ganzen Laufzeit ausgeübt werden. Verglichen mit einer europäischen Option bietet sie damit ein umfangreicheres Ausübungsrecht. Folglich muss der Wert einer amerikanischen Option mindestens dem Wert einer gleichen europäischen Option entsprechen. Es muss also $V^{Am} \geq V^{Eur}$ gelten. Die allfällig auftretende Differenz zwischen zwei gleichen Kontrakten verschiedenen Stils ist die sogenannte Ausübungsprämie.

Der Preis einer europäischen Put-Option erreicht sein Maximum bei $S = 0$, und zwar entspricht dieses dem Barwert des Strikepreises: $V_P^{Eur}(0,t) = Ke^{-r(T-t)}$, welcher für $r > 0$ unter dem intrinsischen Wert K an derselben Stelle liegt (siehe dazu Kapital 2.3). Dies gilt für europäische Puts mit oder ohne Berücksichtigung von Dividenden.

Amerikanische Optionen hingegen können den intrinsischen Wert nie unterschreiten. Für sie gilt $V_P^{am}(S,t) \geq (K - S_t)^+$ beziehungsweise $V_C^{am}(S,t) \geq (S_t - K)^+$. Andernfalls wäre ein sofortiger risikoloser Gewinn möglich, indem man die Option kauft und sofort ausübt. Wenn gleichzeitig der zugehörige Basiswert gekauft würde, bliebe bei einer Put-Option ein Gewinn von $K - V_P^{Am} - S_t > 0$. Bei einer Call-Option wäre bei gleichzeitigem Verkauf des Basiswerts ein Gewinn von $S_t - V_C^{Am} - K > 0$ zu erzielen. Solche Gewinne widersprechen dem Paradigma der Arbitragefreiheit.

Es zeigt sich, dass eine amerikanische Call-Option auf einen dividendenlosen Basiswert den gleichen Wert hat wie die entsprechende europäische Option. Die Wertuntergrenze einer europäischen Verkaufsoption ist $V_C^{Eur} \geq S_t - Ke^{-r(T-t)}$ (siehe dazu *Tabelle 1*, *Kapitel 2.3*). Für $r > 0$ ist der der Barwert des Ausübungspreises während der Laufzeit kleiner als *K*. Somit gilt auch $V_C^{Eur} > (S - K)^+$ bis zum Laufzeitende, was bedeutet, dass die Option ihren intrinsischen Wert immer übersteigt. Die zusätzliche Restriktion für amerikanische Call-Optionen $V_C^{am}(S,t) \geq (S_t - K)^+$ ist in diesem Fall bereits für europäische Optionen erfüllt. Der Stil spielt dann keine Rolle und die Optionen sind gleichwertig $V_C^{Am} = V_C^{Eur}$.

Dividendenzahlungen belohnen das Halten einer Aktie. Es besteht der Anreiz, eine amerikanische Kaufoption frühzeitig auszuüben, um in den Genuss der Dividende zu kommen. Darum kann beim Vorliegen von diskreten Dividendenzahlungen ein amerikanischer Call nicht mit einem europäischen Kontrakt gleichgesetzt werden. Dies gilt ebenso, wenn eine stetige Dividendenrendite angenommen wird.

7.2 Freies Randwertproblem

Die Black-Scholes-Gleichung gilt für europäische Optionen. Amerikanische Optionen lassen sich damit grundsätzlich nicht analytisch bewerten (Mit Ausnahme des obigen Spezialfalls eines Calls ohne Dividendenzahlungen). Wenn es optimal wäre eine amerikanische Option über die ganze Laufzeit zu halten, müsste sie den gleichen Wert haben wie eine europäische Option. Da dies nicht der Fall ist, muss es freien Randwert $S_f(t)$ geben, an dem es optimal ist, die Option auszuüben. Dies lässt sich am Beispiel eines Puts leicht zeigen. Es ist klar, dass $S_f(t)$ zwischen 0 und K liegen muss, wo die Option im Geld ist. Wie oben beschrieben gilt $V_P^{am}(S,t) \geq (K - S_t)^+$. Anhand eines Portefeuilles π bestehend aus einem Put und dem entsprechenden Basiswert wird ersichtlich, dass die Option ausgeübt wird, wenn $V_P^{am}(S,t) = K - S$. Dann beläuft sich der Wert von π auf $(K - S) + S = K$. Statt das Portefuille zu halten, wird dann vorzugsweise die Option ausgeübt und der Betrag K zum risikolosen Zinssatz r reinvestiert. Wenn $V_P^{am}(S,t)$ den intrinsischen Wert übersteigt, ist der Wert des Portefeuilles grösser als K und somit ist es vorteilhaft, die Option nicht auszuüben.

$$S < S_f(t) \rightarrow V_P^{Am} = K - S \rightarrow \textit{vorzeitige Ausübung lohnt sich}$$

$$S > S_f(t) \rightarrow V_P^{Am} > (K - S)^+ \rightarrow die\ Option\ wird\ gehalten$$

Abbildung 5 stellt den obigen Sachverhalt graphisch dar.

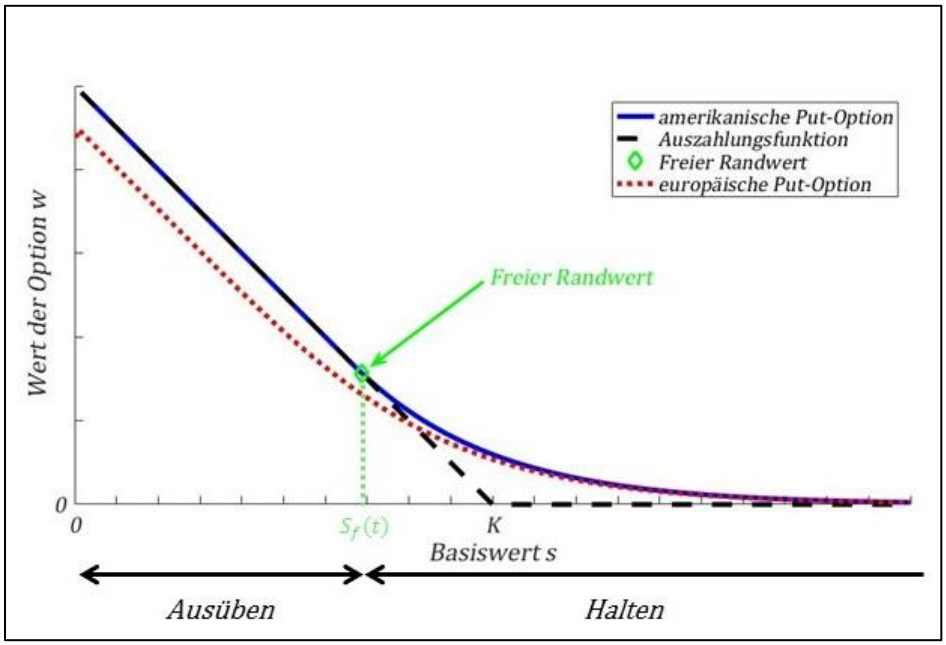

Abbildung 5 Freier Randwert in Anlehnung an (Seydel, 2006, S. 141)

Der Optionswert ist für $S < S_f(t)$ durch die Auszahlungsfunktion bestimmt. Für $S > S_f(t)$ wird die Option nicht vorzeitig ausgeübt und sie verhält sich folglich wie eine europäische Option. Auf diesem Bereich erfüllt sie die Black-Scholes-Gleichung. Der freie Randwert $S_f(t)$ ist der Punkt, an dem die Kurve des Optionswerts in die Auszahlungsfunktion übergeht. Er ist a priori unbekannt. Die Problemstellung gehört somit zur Problemklasse des freien Randwerts.

7.3 Hindernisproblem

Die Betrachtung eines klassischen Hindernisproblems verdeutlicht das vorliegende Problem des freien Randes. Man nimmt ein glattes Objekt an, dessen Höhe durch die Funktion $f(x)$ gegeben sei. Man stelle sich nun ein elastisches Band vor, welches an den Endpunkten $A = -1$ und $B = +1$ befestigt ist. Unter das Gummiband wird ein Objekt geschoben, so dass das

Band mit minimaler Länge über das Objekt gespannt wird. Die Ablenkung des Bands vom ursprünglichen Verkauf ist durch die Funktion $u(x)$ gegeben. Damit es nur eine Kontaktregion gibt, wird verlangt, dass $f(\pm 1) < 0$ und $f(x) > 0$ für $-1 < x < 1$. Zudem soll es sich um ein konkaves Objekt handeln, so dass die zweite Ableitung negativ ist ($f'' < 0$). Es ist klar, dass das Gummiband an jeder Stelle entweder oberhalb des Objekts oder genau auf diesem zu liegen kommt. An den Punkten $x = P$ und $x = Q$ berührt das Gummiband das Objekt tangential. Diese beiden unbekannten Punkte sind die sogenannten freien Randwerte. Sie markieren Anfang und Ende der Kontaktregion. *Abbildung 6* veranschaulicht die Anordnung des Hindernisproblems.

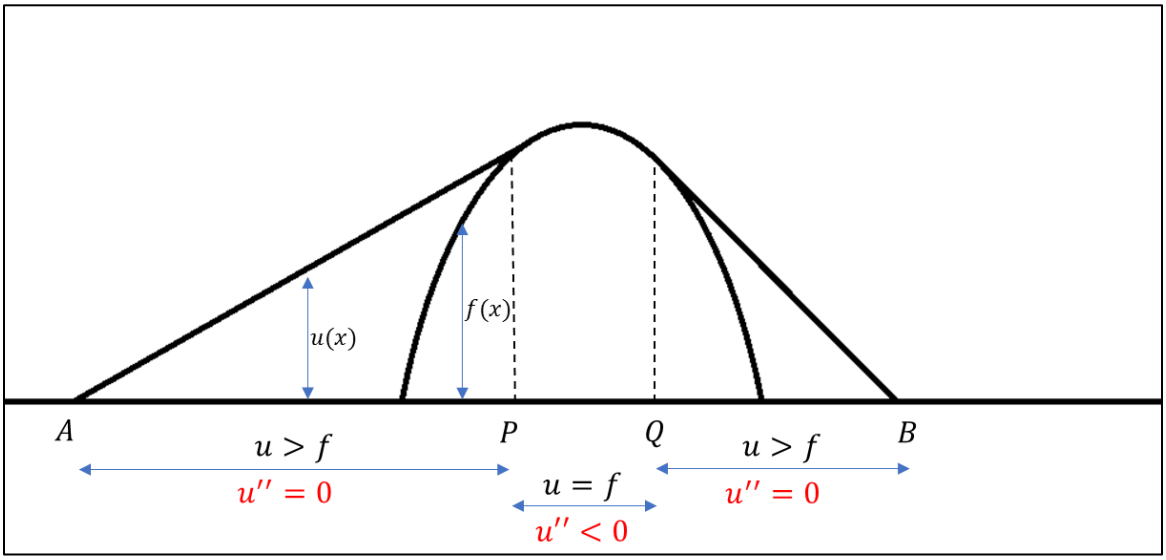

Abbildung 6 Skizze des Hindernisproblems in Anlehnung an (Wilmott et al., 2009, S. 109)

Man sieht nun, dass für die zwei äusseren Intervalle $A < x < P$ und $Q < x < B$ folgende Bedingungen zutreffen:

1. $u'' = 0$, da das Gummiband gerade verlaufen muss.
2. Das Gummiband verläuft auf diesem Abschnitt oberhalb des Objekts, also $u > f$

Anders ausgedrückt gilt: Wenn $u > f$ dann $u'' = 0$.

Für das Kontaktintervall $P < x < Q$ treffen folgende Bedingungen zu:

1. Das Gummiband liegt direkt auf dem Objekt, das heisst $u = f$.

2. Zudem ist bekannt, dass $u'' = f'' = 0$, da das Gummiband den konkaven Verlauf des Objekts annimmt.

Somit gilt: Wenn $u = f$, dann $u'' < 0$

Aus der Anordnung ergeben sich die Funktionswerte $u(-1) = 0$ und $u(1) = 0$ für die Befestigungspunkte A und B. Ferner wird für die Funktion f und erste Ableitung f' Stetigkeit vorausgesetzt.

Diese Beobachtungen lassen sich nun in der Form eines linearen Komplementaritätsproblems beschreiben:

$$\begin{aligned} u''(u-f) &= 0 \\ -u'' &\geq 0 \\ u-f &> 0 \end{aligned} \qquad \textbf{7.1}$$

In dieser Formulierung sind die a priori unbekannten Punkte $x = P$ und $x = Q$ nicht explizit enthalten. Durch Lösung der obigen Vorschrift sind die freien Randwerte jedoch implizit gegeben. Die Randpunkte liegen da, wo die Funktion $u - f$ von nicht-null ($u - f > 0$) zu null ($u - f = 0$) wechselt.

In Analogie zum Hindernisproblem kann das freie Randwertproblem von amerikanischen Optionen ebenfalls als lineares Komplementaritätsproblem formuliert werden.

7.4 Lineares komplementäres Problem

Das freie Randwertproblem amerikanischer Optionen in der Form des Komplementaritätsproblem 7.1 lautet:

$$\begin{cases} \left(\partial_t V + \frac{1}{2}\sigma^2 s^2 \partial_{ss} V + (r-q)s\partial_s V - rV\right)(V-g) = 0 & in \quad \mathbb{R}^+ \\ \partial_t V + \frac{1}{2}\sigma^2 s^2 \partial_{ss} V + (r-q)s\partial_s V - rV \leq 0 & in \quad \mathbb{R}^+ \\ V(s,t) \geq g(s) & in \quad \mathbb{R}^+ \\ V(s,T) = g(s) & in \quad \mathbb{R}^+ \end{cases} \qquad \textbf{7.2}$$

Neben den drei Bedingungen des Komplementaritätsproblems ist weiterhin die Auszahlungsfunktion bekannt. Auf den Beweis, dass die Analogie zutrifft, soll hier verzichtet werden.

Im *Kapitel 4* wurde das Bewertungsproblem europäischer Optionen mittels finiter Differenzen in die Matrix-Schreibweise 4.12 überführt. Das System 7.2 kann nun mit denselben Matrizen und Vektoren abgebildet werden.

$$\begin{cases} (Bw_{j+1} - Cw_j)(w_{j+1} - g) = 0 \\ Bw_{j+1} \geq Cw_j \\ w_{j+1} - g \geq 0 \end{cases} \qquad \textbf{7.3}$$

Man erhält so das Ungleichungssystem 7.3. Dieses lässt sich numerisch mittels einer Adaption des bereits bekannten SOR-Verfahrens (siehe *Kapitel 6*), dem Projected Successive Over-Relaxation-Verfahren, lösen.

7.5 Projected Successive Over-Relaxation-Verfahren

Das Projected SOR-Verfahren (pSOR-Verfahren) löst auch ein Gleichungssystem der Form $Ax = b$ iterativ, jedoch unter Berücksichtigung einer zusätzlichen Nebenbedingung $b \geq y$. Nach Creyer ist das Verfahren in der Lage, das diskrete lineare Komplementaritätsproblem zu lösen. Für eine symmetrisch, positiv definierte Matrix A und 1< w <1 konvergiert die Methode gegen die eindeutig definierte Lösung des Problems. (Creyer, 1971). Es lässt sich zeigen, dass das pSOR-Verfahren zum Zwecke der Optionsbewertung genutzt werden kann (Günther & Jüngel, 2010, S. 210-213).

Die Bedingungen 7.3 lassen sich zusammengefasst auch schreiben als:

$$\begin{gathered} \min\{Bw_{j+1} - Cw_j, w_j - g\} = 0 \equiv \\ \min\{w_{j+1} - B^{-1}Cw_j, w_{j+1} - g\} = 0 \equiv \\ v = \max\{B^{-1}Cw_j, g\} \end{gathered} \qquad \textbf{7.4}$$

Der SOR-Algorithmus wurde im *Kapitel 6* benutzt, um das Gleichungssystem $Bw_{j+1} - Cw_j = 0$ (Gleichung 4.12) zu lösen. Anstatt aber wie in dieser Vorschrift das neu berechnete Element in den Schätzvektor zu übernehmen, wird beim pSOR-Verfahren zusätzlich das Einhalten der Bedingung $v = \max\{B^{-1}Cw_j, g\}$ verlangt. Dies führt dazu, dass entweder $w = g$ oder $w > g$ und somit der Nebenbedingung $w_{j+1} - g \geq 0$ Rechnung getragen wird.
Im impliziten Euler-Verfahren hängen die Werte von w derart zusammen, dass die Gewährleistung der Bedingung gleichzeitig mit der Berechnung der Iteration stattzufinden hat, um konsistente Ergebnisse zu erzielen.

Der Code der Funktion pSOR() ist im *Anhang 4* hinterlegt.

8 Bewertung amerikanischer Optionen ohne Berücksichtigung diskreter Dividenden

8.1 Einleitung

Der im *Kapitel 5* vorgestellte Algorithmus `Eur()` wird nun unter Zuhilfenahme des pSOR-Verfahrens zur Funktion `Ami()` für die Bewertung von amerikanischen Optionen abgeändert. Wie bei den europäischen Optionen, soll das Programm einen Vektor mit $N+2$ Basiswerten und den dazugehörigen Optionswerte im Zeitpunkt $t = 0$ ausgeben. Es wird bei der Diskretisierung das gleiche Vorgehen angewendet und es werden dieselben Eingabeargumente verwendet wie in *Kapitel 5*. Die Routine `pSOR()` aus *Kapitel 7.5* wird unveränderten übernommen (mit $\omega = 1$ und einer Fehlerschranke von 10^{-12}).

8.2 Implementierung in MATLAB®

Für die Preisberechnung muss nun das Ungleichungssystem 7.2 gelöst werden. Entsprechend muss in Abweichung zu Routine `Eur()` in der Iterationsschlaufe Funktion `pSOR()` mit Übergabe der Argumente aufgerufen werden. Die Wertzuweisung `w = B\C*w` ist an dieser Stelle durch `w = pSOR(B,w,C*w,g(s))` zu ersetzen. Dabei wird w aus dem letzten

Iterationsschritt als Schätzvektor verwendet. Der Code für die Funktion `Ami()` mit Kommentaren ist im *Anhang 5* zu finden.

8.3 Numerische Beispiele

Beispiel 4

D.Y. Tangman et al. entwickelten einen neuen Bewertungsalgorithmus (Optimal Compact Algorithm, OCA) und verglichen diesen mit anderen Bewertungsmethoden (Tangman et al., 2008). Als Massstab für die Genauigkeit nahmen sie die monoton konvergierende Binomialmethode nach Leisen und Reimer (Leisen & Reimer, 1996). Zu Vergleichszwecken berechnen sie unter anderem eine amerikanische Put-Option mit den Parametern $K = 10$, $T = 3,\ \sigma = 0.3\ r = 0.10,\ q\ =\ 0.05$ (Tangman et al., 2008, S. 27).

Das Beispiel wird mit `Ami()` nachgerechnet. Dazu werden die Diskretisierungsparameter $N = 511$ und $M = 400$ sowie $s_{links} = 0$ und $s_{rechts} = 4K$ gewählt. Aus den Ergebnisvektoren werden mittels Interpolation die Resultate herausgelesen. *Tabelle 2* zeigt einen Vergleich der Resultate.

Basiswert s	**Tangman et al**	**Leisen & Reimer**	**`americanput()`**
80	23.0780	23.0777	23.0774
90	7.7252	17.7250	17.7247
100	13.7203	13.7203	13.7199
110	10.6881	10.6881	10.6877
120	8.3720	8.3721	8.3716

Tabelle 2 Resultate Beispiel 4: Berechnung des fairen Preises einer amerikanischen Put-Option

Beispiel 5

Beispiel 2 soll auf eine amerikanische Put-Option übertragen und das Ergebnis mit dem bereits berechneten europäischen Kontrakt verglichen werden. *Abbildung 7* zeigt die Ergebnisse von `Eur()` und `Ami()` für die Vorgaben aus *Beispiel 2*.

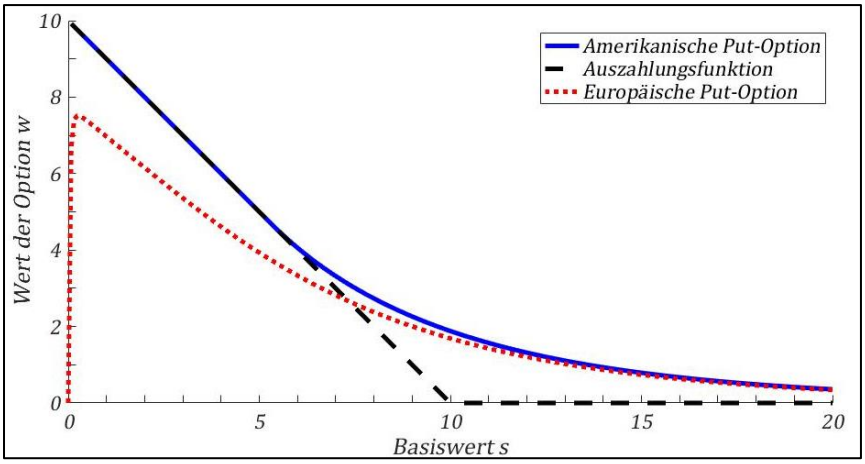

Abbildung 7 Gegenüberstellung amerikanische und europäische Put-Option Beispiel 5 (vgl. Beispiel 2)

Beispiel 6

Bei einem risikolosen stetigen Zinssatz $r = 0.06$ soll die Oberfläche der amerikanischen Put-Funktion mit der optimalen Ausübungslinie dargestellt werden. Die Optionsparameter sind $K = 10, T = 1$ und $\sigma = 0.3$ (Seydel, 2006, S. 7).

Dazu wird die Routine `Ami()` zu `Ami3D()` abgeändert, indem eine Matrix Z verwendet wird, um die Wertvektoren spaltenweise abzuspeichern. In der Iterationsschlaufe wird zudem für jedes *w* der freie Randwert herausgelesen und abgespeichert. Der ganze Algorithmus `Ami3D ()` mit Kommentaren ist im *Anhang 6* zu finden. In *Abbildung 8* wird das Ergebnis für $N = 1023$ und $M = 800$ (ohne Randpunkte) dargestellt.

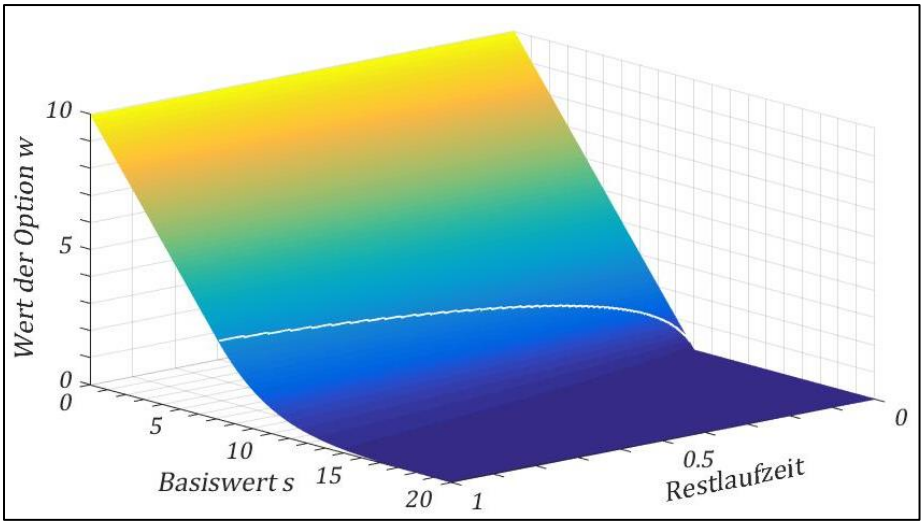

Abbildung 8 Oberfläche einer amerikanischen Put-Option mit vorzeitiger Ausübungslinie Beispiel 6

Beispiel 7

In einem Paper über diskrete Dividenden bei amerikanischen Optionen bestimmt Gunter Meyer die Ausübungslinie verschiedener Optionen über die Zeit (Meyer, 2001). Als Vergleich zu den Puts mit diskreten Dividenden wird folgende Option mit stetiger Dividende genommen: $K = 3,\ T = 0.5,\ s = 0.4,\ r = 0.08$ und $q = 0.0552$ (q wurde so gewählt, dass die Option den gleichen Wert in $t = T$ aufweist, wie die zu vergleichende Option mit diskreter Dividende. Siehe dazu *Beispiele 11 und 12*). Es soll die vorzeitige Ausübungslinie gezeichnet werden.

Die Routine `Ami()` wird so ergänzt, dass während der Iteration aus jedem erzeugten Vektor w der freie Randwert herausgelesen und abgespeichert wird. Die Funktion `AmiFreier-Randwert()` gibt diesen Vektor dann zusammen mit dem dazugehörigen Zeitvektor aus, so dass die Bewegung des freien Randwerts in *Abbildung 9* dargestellt werden kann. *Anhang 7* beinhaltet den Code zur Funktion `AmiFreierRandwert()` mit Kommentaren.

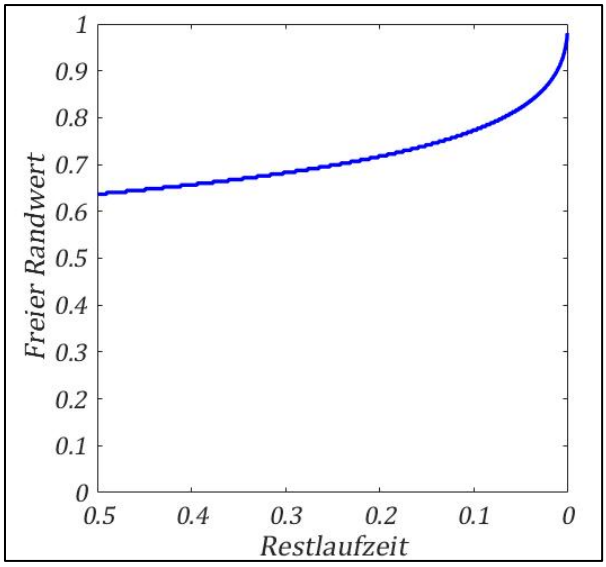

Abbildung 9 Optimale Ausübungslinie für eine Put-Option Beispiel 7

9 Diskrete Dividenden

9.1 Eigenschaften Diskreter Dividenden

Im ursprünglichen Black-Scholes-Modell blieben Gewinnausschüttungen des Basiswerts gänzlich unberücksichtigt. Später wurden stetige Dividendenrenditen in das Modell aufgenommen. Die Annahme, dass Dividenden stetig bezahlt werden, ist jedoch eine starke Vereinfachung der realen Verhältnisse.

Die Mitgliedschaftsrechte für Aktionäre in der Schweiz beinhalten das Recht auf einen verhältnismässigen Anteil am Bilanzgewinn, soweit dieser nach dem Gesetz oder den Statuten zur Verteilung unter den Aktionären bestimmt ist. (Art. 660 Schweizerisches Obligationenrecht) Damit eine Gesellschaft überhaupt Dividenden auszahlen kann, muss sie, nach Abzug allfälliger vorrangigerer Verluste, einen Bilanzgewinn ausweisen. Anschliessend müssen die gesetzlichen und statutarischen Zuweisungen an die Reserven erfolgen (Art. 674 Schweizerisches Obligationenrecht). Und schliesslich stimmt die Generalversammlung über den definitiven Ausschüttungsbetrag ab.

Die Dividende hängt somit grundsätzlich vom erwarteten Gewinn des Unternehmens ab. Wenn Anleger Aktien nach dem Discounted Earnings-Ansatz bewerten, hängt auch der Aktienkurs vom erwarteten Gewinn ab. Aktienkurs und Dividendenhöhe sind also nicht voneinander unabhängig.
Der genaue Betrag der Dividendenzahlung ist erst kurz vor Auszahlung bekannt. Da börsenkotierte Unternehmungen aber auch Zwischenresultate publizieren und meistens eine gleichbleibende Dividendenpolitik betreiben, kann der Betrag im Vorfeld abgeschätzt werden.

Im Folgenden werden darum einige Annahmen getroffen:

- Der Dividendenbetrag D ist im Voraus bekannt.
- Die Dividende ist eine diskrete Zahlung und das Datum der Dividendenzahlung t_D (ex-Datum) ist bekannt.
- Die Dividende ist von *s* unabhängig.

Die Dividende wird als absoluter Betrag in der Nominalwährung der Aktie angegeben und bezieht sich immer auf eine Aktie. Diese wird in der Regel bar und nur einmal im Jahr ausbezahlt. In anderen Ländern können auch mehrmalige Auszahlungen pro Jahr Usanz sein wie zum Beispiel in den USA, wo quartalsweise Dividenden Standard sind.

9.2 Auswirkung einer Dividendenzahlung auf den Aktienkurs und den Optionswert

Beim Kauf einer Aktie kurz von der Dividendenzahlung t_D^- erhält der Käufer zusammen mit der Aktie das Recht auf die anstehende Auszahlung. Unter Annahme der Arbitragefreiheit muss dieser Vorteil gegenüber dem Kauf zum kurz darauffolgenden Zeitpunkt nach der Dividendenzahlung t_D^+ durch einen Kurssprung nach unten korrigiert werden.

$$S(t_D^-) = S(t_D^+) - D$$

Im Gegensatz zum Aktionär hat der Halter einer Option kein Anrecht auf Dividendenzahlung. Der Kurssprung betrifft ihn nicht direkt. Zumal die Kursänderung für den Optionshalter

absehbar ist, wenn Zeitpunkt und Höhe der Dividendenzahlung bekannt sind. Folglich sollte sich der Werte der Option durch die Auszahlung nicht verändern und über den Zeitpunkt der Dividendenzahlung t_D stetig sein. Trotz Diskontinuität in der unabhängigen Variable s muss also die zeitliche Stetigkeit gewährleistet sein. Es lässt sich nachfolgende Stetigkeitsbedingung aufstellen:

$$V(s, t_D^-) = V(s - D, t_D^+) \qquad \textbf{9.2}$$

Kurz vor einer Dividendenzahlung ist das Ausüben einer amerikanischen Put-Option eventuell nicht mehr attraktiv, da mit dem Verkauf der Aktie auch das Recht auf die anstehende Dividendenzahlung verkauft wird. Der Wert $S_f(t)$ verschiebt sich gegen $s = 0$. Wenn der Dividendenbetrag gross genug ist, löst sich die Funktion des Optionswerts sogar ganz von der Auszahlungsfunktion. Dann übersteigt der faire Preis der Option für alle Kurse von s den intrinsischen Optionswert. Nach der Dividendenzahlung verschiebt sich die Kurve um den Dividendenbetrag zurück nach links. *Abbildung 10* zeigt beispielhaft die Bewegung der Kurve. Die Verschiebung verläuft horizontal. Der Wert der Option ändert sich im Zeitpunkt t_D darum nicht.

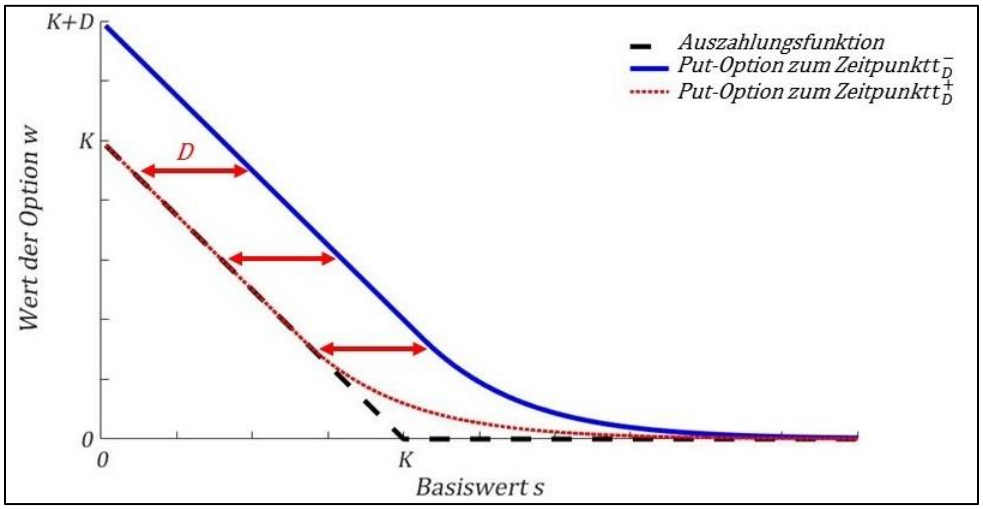

Abbildung 10 Auswirkungen einer Dividendenzahlung auf eine amerikanische Put-Option in Anlehnung an (Wilmott, 2000, S. 130)

10 Bewertung von Optionen mit diskreten Dividenden

10.1 Einleitung

Die Programme `Eur()` und `Ami()` sollen so weiterentwickelt werden, dass anstelle von stetigen Dividenden diskrete Dividendenzahlungen berücksichtigt werden können. Da während der Laufzeit einer Option mehrere Dividendenzahlungen stattfinden können, sollen die Routinen mit beliebig vielen Dividendenzahlungen umgehen können. Die Eingabe erfolgt mittels Übergabe eines Zeilenvektors $D = [D_1\ D_2\ ...\ D_n]$ für die Dividendenhöhen und eines zweiten Zeilenvektors $t_D = [t_{D1}\ t_{D2}\ ...\ t_{Dn}]$ für die entsprechenden Auszahlungszeitpunkte. Um die Eingabe zu erleichtern, werden die Auszahlungszeitpunkte t_D dabei vom Bewertungszeitpunkt aus in Richtung der Kalenderzeit bemessen. Die anderen Argumente sind aus den Vorgängerprogrammen bekannt.

10.2 Implementierung in MATLAB®

Die Berechnung wird nach der Finiten-Differenzen-Methode wieder vom Laufzeitende aus mittels Rückwärtsiteration durchgeführt. Die ganze Laufzeit wird durch die Auszahlungszeitpunkte in verschiedene Zeitintervalle unterteilt (siehe *Abbildung 11*).

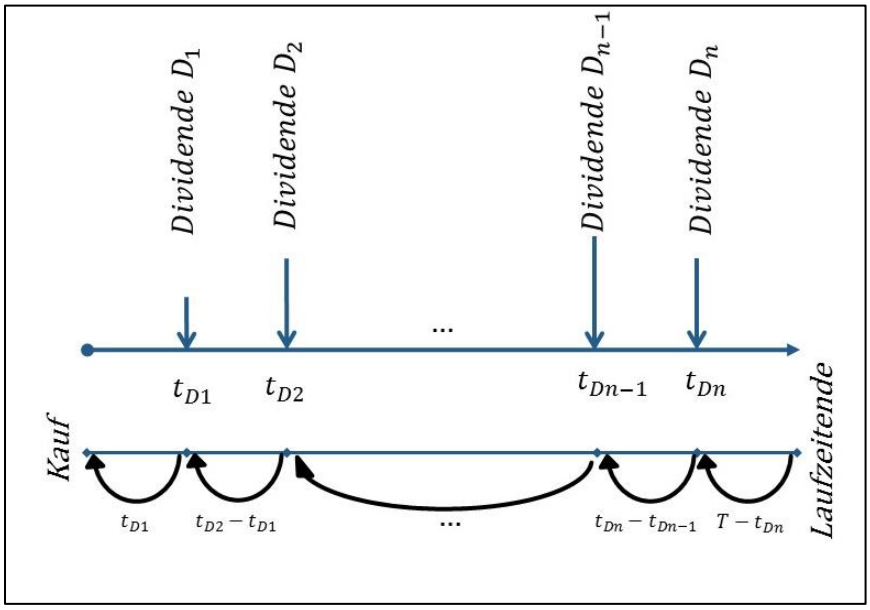

Abbildung 11 Rückwärtsiteration bei mehreren Dividendenzahlungen

Die Auszahlungsfunktion ist der Startvektor für die Rückwärtsberechnung vom Laufzeitende bis zur letzten Dividendenzahlung D_n. Die Iteration nähert sich so vom Laufzeitende ausgehend dem Punkt t_{Dn} an, wo der Kurssprung erfolgt. In t_{Dn} muss die Stetigkeitsbedingung 9.2 erfüllt sein, um dem Kurssprung wie oben beschrieben Rechnung zu tragen. Alle Optionswerte werden in t_{Dn} auf der s-Achse um D_n verschoben. Dabei ist zu beachten, dass w nur Optionswerte zu den Punkten des Diskretisierungsgitter $\mathcal{g}_s \times \mathcal{g}_t$ enthält. Nur wenn D_n. einem Vielfachen der Maschenweite k entspricht, ist der Punkt $s - D_n$ ein Element von $\mathcal{g}_s$. Darum sind die neuen Werte $w(s - D, t_{Dn})$ mittels Interpolation aus dem Vektor w herauszulesen. Der neue Vektor w nach der Verschiebung ist der Ausgangspunkt für die Rückwärtsiteration im nächsten Zeitintervall. Die Verschiebung der Optionswerte nach rechts kann bei Call-Optionen dazu führen, dass w den inneren Wert der Option unterschreitet. Da dies bei amerikanischen Optionen ausgeschlossen ist (siehe *Kapitel 7.1*), muss in der Routine für amerikanische Optionen `AmiDiv()` zusätzlich die Bedingung $w(s, t_{Dn}^-) = \max\{w(s - D, t_{Dn}^+), g(s)\}$ durchgesetzt werden.

MATLAB®-Code der Funktion `AmiDiv()`

```
function [s,w] = AmiDiv(CallorPut,K,T,sigma,r,D,td,sl,sr,N,M)
theta = 0.5;
a = @(s)-0.5*sigma^2*s.^2;
b = @(s)-(r)*s;
c = @(s)r*s.^0;
if CallorPut == 'c'
    g = @(s)max(0,s-K);
elseif CallorPut == 'p'
    g = @(s)max(0,K-s);
end
h = (sr-sl)/(N+1);
s = (sl:h:sr)';
zeitspanne = flip([td,T]-[0,td]);
k = zeitspanne/M;
dividende = flip([0,D]);
M0 = spdiags(c(s(2:N+1)),0,N,N);
M1 = 1/(2*h)*spdiags([-b(s(3:N+2)),zeros(N,1),b(s(1:N))],-1:1,N,N);
M2 = 1/h^2*spdiags([a(s(3:N+2)),-2*a(s(2:N+1)),a(s(1:N))],-1:1,N,N);
A = M2+M1+M0; I = speye(N);
s = s(2:N+1);
w = g(s);
for z=1:length(td)+1
    B = I + k(z)*theta*A; C = I - (1-theta)*k(z)*A;
    for j = 1:M
        w = pSOR(B,w,C*w,g(s));
    end
    w = max(interp1(s,w,s-dividende(z),'pchip'),g(s));
```

```
end
s = [sl;s;sr];w = [0;w;0];
```

Im *Anhang 9* findet sich der obige Code mit Kommentaren. *Anhang 8* zeigt den ganzen Code der Funktion `EurDiv()` mit Kommentaren.

10.3 Numerische Beispiele für europäische Optionen

Beispiel 8

Eine europäischen Call-Option mit $X = 100,\ T = 1,\ \sigma = 0.3$ zahle in der Mitte der Laufzeit $t_D = 0.5$ eine einmalige Dividende von $D = 7$. Der stetige risikolose Zinssatz r betrage 6%. Es soll der faire Preis der Option bei einem Kurs des Basistitels $s = 100$ berechnet werden (Haug et al., 2003, S. 21)

Die Autoren Haug et al. berechnen mit ihrer Methode einen Wert von 3.4383. `EurDiv('c',100,1,0.3,0.06,7,0.5,0,400,511,400)` liefert nach Interpolation das Ergebnis: 3.4376.

Beispiel 9

Es werden europäische Call-Option mit $X = 100$ und $\sigma = 0.25$ mit verschiedenen Laufzeiten T betrachtet. Der Basiswert der Optionen zahlt jeweils in der Mitte jedes Jahres eine Dividende von 4. Der stetige risikolose Zinssatz r betrage 6%. (Haug et al., 2003, S. 27)

Die Berechnung erfolgte mit den Diskretisierungsparametern $s_{links} = 0, s_{rechts} = 400, N = 1023$ und $M = 500$. *Tabelle 3* stellt die mit `EurDiv()` berechneten und interpolierten Resultate den von den Autoren errechneten Werte gegenüber.

Laufzeit T	Zeitpunkte der Dividendenzahlung td	Wert für S=100 nach Haug et al.	Wert für S =100 mit `EurDiv ()`
1	0.5	10.6606	10.6603
2	0.5 / 1.5	15.1989	15.1732
3	0.5 / 1.5 / 2.5	18.5984	18.2005

Tabelle 3 Vergleich der Resultate aus Beispiel 9

10.4 Numerische Beispiele für amerikanische Optionen

Beispiel 10

Gesucht ist der faire Preis einer amerikanischen Put-Option mit $K = 1,\ T = 0.5, \sigma = 0.4$ deren Basistitel eine einmalige Dividende von $D = 0.02$ zum Zeitpunkt $t_d = 0.3$ bezahlt. Der stetige risikolose Zinssatz r betrage 8%. Der Autor des Beispiels wählt die Randpunkte $s_{links} = 0$ und $s_{rechts} = 3$ (Meyer, 2001, S. 12 ff.)

Es werden dieselben Randpunkte sowie $N = 511$ und $M = 400$ für die Berechnung verwendet. Die mit `AmiDiv()` erzeugten interpolierten Ergebnisse werden in *Tabelle 4* mit den Resultaten des Autors verglichen.

Kurs Basiswert s	Optionswert nach (Meyer, 2001)	Optionswert mit `AmiDiv()`
0.8	0.2228	0.2229
1.0	0.1046	0.1046
1.2	0.0430	0.0430

Tabelle 4 Resultate aus Beispiel 10

Beispiel 11

Es soll die vorzeitige Ausübungslinie einer Option auf einen Basiswert mit einmaliger Dividendenzahlung abgebildet werden. Dazu wird die Option aus *Beispiel 10* widerverwendet.

Damit die Ausübungslinie gezeichnet werden kann, wird die Option `AmiDiv()` analog zu `AmiFreierRandwer()` abgeändert. Mit der so angefertigten Funktion `AmiDivFreierRandwert()` konnte *Abbildung 12* erzeugt werden. Die Diskretisierung erfolgte mit $s_{links} = 0$, $s_{rechts} = 3$, N=1023 und M=800. Der Code der Funktion `AmiDivFreierRandwert()` wird in *Anhang 10* zur Verfügung gestellt

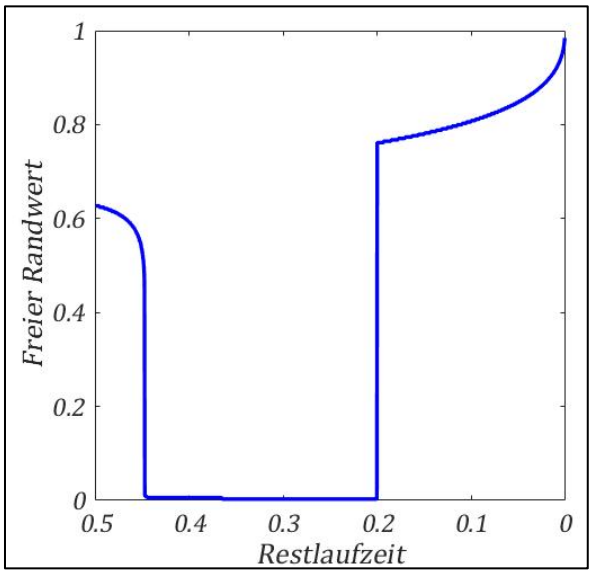

Abbildung 12 Optimale Ausübungslinie der Put-Option aus Beispiel 11

Beispiel 12

Darstellung der vorzeitigen Ausübungslinie einer amerikanischen Put-Option mit zwei im Voraus bekannten Dividendenzahlungen. $K = 1,\ T = 0.5,\ \sigma = 0.4,\ r = 0.12\ D_1 = 0.015, t_{D1} = 0.15, D_2 = \ 0.02, t_{D2} = 0.4, N = 1023, M = 800$. (Meyer, 2001, S. 158).

Mit `AmiDivFreierRandwert('p',1,0.5,0.4,0.12,[0.015 0.02],[0.15 0.4],0,3,1023,800)` wurde *Abbildung 13* erzeugt.

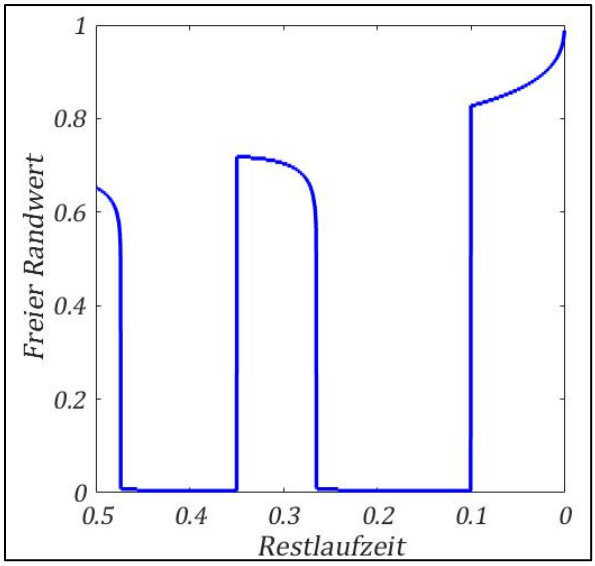

Abbildung 13 Optimale Ausübungslinie der Put-Option aus Beispiel 12

11 Fazit

Die benutzten Methoden scheinen zur Anwendung auf die Bewertung amerikanischer Optionen gut geeignet zu sein. Die Ergebnisse der Berechnungen konnten anhand der Fachliteratur geprüft werden. Die Genauigkeit der erzeugten Resultate legt die Vermutung nahe, dass die entwickelten Routinen zuverlässig arbeiten. Eine grösser Abweichung war bei der Berechnung der europäischen Option mit drei Dividendenzahlungen zu verzeichnen (*Beispiel 9*). Da sich die Fehler bei solchen Berechnungen kumulieren, war das Auftreten grösserer Differenzen zu erwarten. Um die Genauigkeit der Anwendungen abschliessend beurteilen zu können, wäre eine eingehendere Validierung notwendig. Ebenso müssten die Konvergenzeigenschaften geprüft werden. Anhand von Berechnungen mit empirischen Daten könnte die Praxistauglichkeit der Routinen geprüft werden.

Die Implementierung ist ohne grosse Mühe zu bewerkstelligen. Die Routinen zeichnen sich durch Stabilität aus und nehmen keine lange Rechenzeit in Anspruch. Auf einfache Weise lassen sich mit den Verfahren gute Eindrücke über das Verhalten amerikanischer Optionen bei Dividendenzahlungen gewinnen. Mit der Darstellung der optimalen Ausübungslinie lassen sich die Eigenheiten amerikanischer Optionen aufschlussreich veranschaulichen.

Die Routinen könnten noch verbessert werden, indem die Annahme homogener Randbedingungen fallen gelassen wird und stattdessen zutreffendere inhomogene Randbedingungen

implementiert werden. So könnten zum Beispiel Dirichlet- oder Neumannrandbedingungen angewandt werden. Weiter wäre es möglich, Konvergenz zu beschleunigen, indem anstelle des Gauss-Seidel-Verfahrens mit einer tatsächlichen Überrelaxation gearbeitet würde.
Zudem können die gewonnenen Erkenntnisse auf weitere Problemstellungen übertragen werden. So sollte sich die Finite-Differenzen-Methode auch zur Bewertung pfadabhängiger Derivate (z. B. Barrier-Optionen) eignen.

12 Literaturverzeichnis

Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3), S. 637-659.

Cox, J. C., & Ross, S. A. (1976). The valuation of options for alternative stochastic processes. *Journal of financial economics*, 3(1-2), S. 145-166.

Cryer, & C. W. (1971). The solution of a quadratic programming problem using systematic overrelaxation. *SIAM Journal on Control*, 9(3), S. 385-392.

Derman, & E., Kani I. (1994). Riding on a Smile. *RISK*, 7(2), S. 32-39.

Günther, M., & Jüngel, A. (2010). *Finanzderivate mit MATLAB*. 2. Auflage. Wiesbaden: Vieweg + Teubner Verlag.

Haug, E. G., Haug, J., & Lewis, A. (2003). Back to basics: a new approach to the discrete dividend problem. *Wilmott magazine*, 9(4), S. 37-47.

Hull, J. C. (2012). *Options, Futures, And Other Derivatives*. 8. Auflage (Global Edition). Essex: Pearson.

Leisen, D., & Reimer, M. (1996). Binomial Models for Option Valutation - Examining and Improving Convergence. *Applied Mathematical Finance*, 3(4), S. 319-346.

Merton, R. C. (1973). Theory of Rational Option Pricing. *The Bell Journal of Economics and Management Science*, 4(1), S. 141-183

Meyer, G. H. (2001). Numerical investigation of early exercise in American puts with discrete dividends. *Strategic Management Journal*, 2(4), S. 379-393.

Meyer, G.H., (2001). On pricing American and Asian options with PDE methods. *Acta Math. Univ. Comenianae*, 70(1), S. 153-165.

Rubinstein, M. (1994). Implied Binomial Trees. *Journal of Finance*, 49 (3), S. 771-818

Seydel, R. U. (2006). *Tools for Computational Finance*. 3. Auflage. Berlin / Heidelberg: Springer Verlag.

Tangman, D. Y., Gopaul, A., & Bhuruth, M. (2008). A fast high-order finite difference algorithm for pricing American options. *Journal of computational and applied mathematics*, 222(1), S. 17-29.

Wilmott, P. (2000). *On quantitative Finance*. West Sussex: John Wiley & Sons Ltd.

Wilmott, P., Howison, S., & Dewynne, J. (2009). *The mathematics of financial derivatives: A student introduction*. New York: Cambridge University Press.

Anhang 1: MATLAB®-Code Eur()

```
% Berechnet den Wert einer europaeischen Option zum Kaufzeitpunkt t=T fuer
% N+2 Kurswerte s des Basiswerts (ohne Beruecksichtigung diskreter Dividenden).
% Gibt Optionswerte im Vektor w und Gitterpunkte von s im Vektor s aus.
% Die Berechnung erfolgt mittels Crank-Nicholson-Verfahren auf einem
% aequidistanten Gitter mit N+2 Punkten in s und M+1 Zeitpunkten sowie
% homogenen Randbedingungen.
%
% Argumente: CallorPut ('c' oder 'p'), Ausuebungspreis K, Restlaufzeit T,
% Volatilitaet sigma, stetiger risikoloser Zinssatz r, stetige Dividende q,
% linker Rand fuer s
% sl, rechter Rand für s sr, N, M
function [s,w] = Eur(CallorPut,K,T,sigma,r,q,sl,sr,N,M)
% Crank-Nicholson-Verfahren
theta = 0.5;
% Koeffizienten der Black-Scholes-Gleichung
a = @(s)-0.5*sigma^2*s.^2;
b = @(s)-(r-q)*s;
c = @(s)r*s.^0;
% Auszahlungsfunktion festlegen
if CallorPut == 'c'
g = @(s)max(0,s-K);
elseif CallorPut == 'p'
g = @(s)max(0,K-s);
end
% Aequidistantes Gitter definieren
h = (sr-sl)/(N+1);
s = (sl:h:sr)';
k = T/M;
% Matrizen A und I definieren
M0 = spdiags(c(s(2:N+1)),0,N,N);
M1 = 1/(2*h)*spdiags([-b(s(3:N+2)),zeros(N,1),b(s(1:N))],-1:1,N,N);
M2 = 1/h^2*spdiags([a(s(3:N+2)),-2*a(s(2:N+1)),a(s(1:N))],-1:1,N,N);
A = M2+M1+M0; I = speye(N);
% Vektor s auf innere Gitterpunkte beschraenken
s = s(2:N+1);
% Hilfsmatrizen B und C definieren
B = I + k*theta*A; C = I - (1-theta)*k*A;
% Startvektor w definieren (Auszahlungsfunktion)
w = g(s);
% Loop ueber Zeitpunkte (der Vektor wj wird laufend ueberschrieben)
for j = 1:M
% Loesung des Gleichungssystems
w = B\(C*w);
end
% Randpunkte dazunehmen
s=[sl;s;sr]; w=[0;w;0];
```

Anhang 2: MATLAB®-Code Eur3D()

```
% Berechnet und zeichnet den Wert einer europaeischen Option in
% Abhaengigkeit der Kurswerte s und der Zeit (ohne Beruecksichtigung diskreter
% Dividenden). Die Berechnung erfolgt mittels Crank-Nicholson-Verfahren
% auf einem aequidistanten Gitter mit N+2 Punkten in s und M+1 Zeitpunkten
% sowie homogenen Randbedingungen. In der Darstellung werden die Randpunkte
% weggelassen.
%
% Argumente: CallorPut ('c' oder 'p') Ausuebungspreis K, Restlaufzeit T,
% Volatilitaet sigma, stetiger risikoloser Zinssatz r, stetige Dividende q,
% linker Rand für s sl, rechter Rand für s sr, N, M
function [s,zeit,Z] = Eur3D(CallorPut,K,T,sigma,r,q,sl,sr,N,M)
% Crank-Nicholson-Verfahren
theta = 0.5;
% Koeffizienten der Black-Scholes-Gleichung
a = @(s)-0.5*sigma^2*s.^2;
b = @(s)-(r-q)*s;
c = @(s)r*s.^0;
% Auszahlungsfunktion festlegen
if CallorPut == 'c'
g = @(s)max(0,s-K);
elseif CallorPut == 'p'
g = @(s)max(0,K-s);
end
% Aequidistantes Gitter definieren
h = (sr-sl)/(N+1);
s = (sl:h:sr)';
k = T/M;
% Matrizen A und I definieren
M0 = spdiags(c(s(2:N+1)),0,N,N);
M1 = 1/(2*h)*spdiags([-b(s(3:N+2)),zeros(N,1),b(s(1:N))],-1:1,N,N);
M2 = 1/h^2*spdiags([a(s(3:N+2)),-2*a(s(2:N+1)),a(s(1:N))],-1:1,N,N);
A = M2+M1+M0; I = speye(N);
% Vektor s auf innere Gitterpunkte beschraenken
s = s(2:N+1);
% Hilfsmatrizen B und C definieren
B = I + k*theta*A; C = I - (1-theta)*k*A;
% Startvektor w definieren (Auszahlungsfunktion)
w = g(s);
% Matrix fuer die Speicherung aller M+1 Vektoren w
Z = zeros(N,M+1);
% Abspeicherung der Auszahlungswerte zum Zeitpunkt t=0
Z(:,1) = g(s);
% Loop ueber Zeitpunkte (der Vektor wj wird laufend ueberschrieben)
for j = 1:M
% Loesung des Gleichungssystems
w = B\(C*w);
% Speicherung des aktuellen Vektors w
Z(:,j+1) = w;
```

```
end
% Zeitvektor erstellen
zeit = [0:k:T];
% Oberflaeche zeichnen
mesh(s,zeit,Z')
```

Anhang 3: MATLAB®-Code SOR()

```
% Loest das Gleichungssystem Ax=b mit Hilfe des Startvektors x0
% iterativ im SOR-Verfahren.
%
% Argumente: NxN-Matrix A, Start(spalten)vektor x0 der Laenge N,
% Ergebnisvektor b der Laenge N
function x = SOR(A,x0,b)
% Relaxationsparameter Omega (0 < Omega < 2). Omega = 1 fuer Gauss-Seidel-Verfahren
Omega = 1;
% Frei waehlbare Fehlerschranke
schranke = 10^-12;
% Schaetzvektor
x = x0;
% Vektor fuer Fehlermessung
test = x0;
% Pruefwert 'fehler' zu Beginn groesser als Fehlerschranke
fehler = 10;
% Vektor fuer Zwischenresultat aus Rechenvorschrift
xneu = zeros(length(A));
% Loop solange Pruefwert > Fehlerschranke
while fehler > schranke
for i = 1:length(A)
% Rechenvorschrift fuer SOR-Verfahren
xneu(i) = x(i) - (Omega/A(i,i))*(A(i,:) * x - b(i));
% Update des Schaetzvektors an Stelle i
x(i) = xneu(i);
end
% Groesste Aenderung im Vektor x gegenüber dem vorherigen Iterationsschritt finden
fehler = max(abs(x-test));
% Abspeichern des aktuellen Vektors fuer Fehlermessung im naechsten Iterationsschritt
test = x;
end
```

Anhang 4: MATLAB®-Code pSOR()

```
% Loest das Gleichungssystem Ax=b unter der Nebenbedingung x>=y
% mit Hilfe des Startvektors x0 iterativ im Projected SOR-Verfahren.
%
% Argumente: NxN-Matrix A, Start(spaleten)vektor x0 der Laenge N,
% Ergebnisvektor b der Laenge N
function x = pSOR(A,x0,b,y)
% Relaxationsparameter Omega (0 < Omega < 2). Omega = 1 fuer Gauss-Seidel-Verfahren
Omega = 1;
% Frei waehlbare Fehlerschranke
schranke = 10^-12;
% Schaetzvektor
x = x0;
% Vektor fuer Fehlermessung
test = x0;
% Pruefwert 'fehler' zu Beginn groesser als Fehlerschranke
fehler = 10;
% Vektor fuer Zwischenresultat aus Rechenvorschrift
xneu(1:length(A))=0;
% Loop solange Pruefwert > Fehlerschranke
while fehler > schranke
for i = 1:length(A)
% Rechenvorschrift fuer SOR-Verfahren
xneu(i) = x(i) - (Omega/A(i,i))*(A(i,:) * x - b(i));
% Update des Schaetzvektors an Stelle i mit Projektion auf y
x(i) = max(xneu(i),y(i));
end
% Groesste Aenderung im Vektor x gegenüber dem vorherigen Iterationsschritt
% finden
fehler = max(abs(x-test));
% Abspeichern des aktuellen Vektors fuer Fehlermessung im naechsten
% Iterationsschritt
test = x;
end
```

Anhang 5: MATLAB®-Code Ami()

```
% Berechnet den Wert einer amerikanischen Option zum Kaufzeitpunkt t=T fuer
% N+2 Kurswerte s des Basiswerts (ohne Beruecksichtigung diskreter Dividenden).
% Gibt Optionswerte im Vektor w und Gitterpunkte von s im Vektor s aus.
% Die Berechnung erfolgt mittels Crank-Nicholson-Verfahren auf einem
% aequidistanten Gitter mit N+2 Punkten in s und M+1 Zeitpunkten sowie
% homogenen Randbedingungen.
% Greift zur Verwendung des Gauss-Seidel-Verfahrens auf pSOR() zurueck.
%
% Argumente: CallorPut ('c' oder 'p'), Ausuebungspreis K, Restlaufzeit T,
% Volatilitaet sigma, stetiger risikoloser Zinssatz r, stetige Dividende q,
% linker Rand fuer s sl, rechter Rand für s sr, N, M
function [s,w] = Ami(CallorPut,K,T,sigma,r,q,sl,sr,N,M)
% Crank-Nicholson-Verfahren
theta = 0.5;
% Koeffizienten der Black-Scholes-Gleichung
a = @(s)-0.5*sigma^2*s.^2;
b = @(s)-(r-q)*s;
c = @(s)r*s.^0;
% Auszahlungsfunktion festlegen
if CallorPut == 'c'
g = @(s)max(0,s-K);
elseif CallorPut == 'p'
g = @(s)max(0,K-s);
end
% Aequidistantes Gitter definieren
h = (sr-sl)/(N+1);
s = (sl:h:sr)';
k = T/M;
% Matrizen A und I definieren
M0 = spdiags(c(s(2:N+1)),0,N,N);
M1 = 1/(2*h)*spdiags([-b(s(3:N+2)),zeros(N,1),b(s(1:N))],-1:1,N,N);
M2 = 1/h^2*spdiags([a(s(3:N+2)),-2*a(s(2:N+1)),a(s(1:N))],-1:1,N,N);
A = M2+M1+M0; I = speye(N);
% Vektor s auf innere Gitterpunkte beschraenken
s = s(2:N+1);
% Hilfsmatrizen B und C definieren
B = I + k*theta*A; C = I - (1-theta)*k*A;
% Startvektor w definieren (Auszahlungsfunktion)
w = g(s);
% Loop ueber Zeitpunkte (der Vektor wj wird laufend ueberschrieben)
for j = 1:M
% Loesung des Ungleichungssystems mittels pSOR()
w = pSOR(B,w,C*w,g(s));
end
% Randpunkte dazunehmen
s = [sl;s;sr];w = [0;w;0];
```

Anhang 6: MATLAB®-Code Ami3D()

```
% Berechnet und zeichnet den Wert einer amerikanischen Option in
% Abhaengigkeit der Kurswerte s und der Zeit (ohne Beruecksichtigung diskreter
% Dividenden). Die Berechnung erfolgt mittels Crank-Nicholson-Verfahren
% auf einem aequidistanten Gitter mit N+2 Punkten in s und M+1 Zeitpunkten
% und homogenen Randbedingungen. Greift zur Verwendung des Gauss-Seidel-Verfahrens
% auf pSOR() zurueck. Die Randpunkte werden weggelassen.
%
% Argumente: CallorPut ('c' oder 'p'), Ausuebungspreis K, Restlaufzeit T,
% Volatilitaet sigma, stetiger risikoser Zinssatz r, stetige Dividende q,
% linker Rand für s sl, rechter Rand für s sr, N, M
function [s,zeit,Z,sFreierRand,wFreierRand] = Ami3D(CallorPut,K,T,sigma,r,q,sl,sr,N,M)
% Crank-Nicholson-Verfahren
theta = 0.5;
% Koeffizienten der Black-Scholes Gleichung
a = @(s)-0.5*sigma^2*s.^2;
b = @(s)-(r-q)*s;
c = @(s)r*s.^0;
% Auszahlungsfunktion festlegen
if CallorPut == 'c'
g = @(s)max(0,s-K);
elseif CallorPut == 'p'
g = @(s)max(0,K-s);
end
% Aequidistantes Gitter definieren
h = (sr-sl)/(N+1);
s = (sl:h:sr)';
k = T/M;
% Matrizen A und I definieren
M0 = spdiags(c(s(2:N+1)),0,N,N);
M1 = 1/(2*h)*spdiags([-b(s(3:N+2)),zeros(N,1),b(s(1:N))],-1:1,N,N);
M2 = 1/h^2*spdiags([a(s(3:N+2)),-2*a(s(2:N+1)),a(s(1:N))],-1:1,N,N);
A = M2+M1+M0; I = speye(N);
% Vektor s auf innere Gitterpunkte beschraenken
s = s(2:N+1);
% Hilfsmatrizen B und C definieren
B = I + k*theta*A; C = I - (1-theta)*k*A;
% Startvektor w definieren (Auszahlungsfunktion)
w = g(s);
% Matrix fuer die Speicherung aller M+1 Vektoren w bereitstellen
Z = zeros(N,M+1);
% Abspeicherung der Auszahlungswerte zum Zeitpunkt t=0
Z(:,1) = g(s);
% Vektoren für freien Randwert bereitstellen
sFreierRand = zeros(M+1);
wFreierRand = zeros(M+1);
% Loop ueber Zeitpunkte (der Vektor wj wird laufend ueberschrieben)
for j = 1:M
% Loesung des Ungleichungssystems mittels pSOR()
```

```
w = pSOR(B,w,C*w,g(s));
% Speicherung des aktuellen Vektors w
Z(:,j+1) = w;
% Punkt der vorzeitigen Ausuebung finden und speichern
idx = find(w > g(s),1);
sFreierRand(j+1) = s(idx);
wFreierRand(j+1) = w(idx);
end
% Zeitvektor erstellen
zeit = [0:k:T];
% Oberflaeche zeichnen
mesh(s,zeit,Z')
hold on
plot3(sFreierRand(2:M+1),zeit(2:M+1),wFreierRand(2:M+1),'r-')
```

Anhang 7: MATLAB®-Code AmiFreierRandwert()

```
% Stellt den freien Randwert im zeitlichen Verlauf dar
% (ohne Beruecksichtigung diskreter Dividenden). Die Berechnung erfolgt
% mittels Crank-Nicholson-Verfahren auf einem aequidistanten Gitter mit
% N+2 Punkten in s und M+1 Zeitpunkten und homogenen Randbedingungen. Die
% Funktion wendet das Gauss-Seidel-Verfahren an, indem sie auf pSOR() zurueckgreift.
%
% Argumente:CallorPut ('c' oder 'p') Ausuebungspreis K, Restlaufzeit T,
% Volatilitaet sigma, Stetiger risikoser Zinssatz r, stetige Dividende q,
% linker Rand für s sl, rechter Rand für s sr, N, M
function [zeit,sFreierRand] = AmiFreierRandwert(CallorPut,K,T,sigma,r,q,sl,sr,N,M)
% Crank-Nicholson-Verfahren
theta = 0.5;
% Koeffizienten der Black-Scholes-Gleichung
a = @(s)-0.5*sigma^2*s.^2;
b = @(s)-(r-q)*s;
c = @(s)r*s.^0;
% Auszahlungsfunktion festlegen
if CallorPut == 'c'
g = @(s)max(0,s-K);
elseif CallorPut == 'p'
g = @(s)max(0,K-s);
end
% Aequidistantes Gitter definieren
h = (sr-sl)/(N+1);
s = (sl:h:sr)';
k = T/M;
% Matrizen A und I definieren
M0 = spdiags(c(s(2:N+1)),0,N,N);
M1 = 1/(2*h)*spdiags([-b(s(3:N+2)),zeros(N,1),b(s(1:N))],-1:1,N,N);
M2 = 1/h^2*spdiags([a(s(3:N+2)),-2*a(s(2:N+1)),a(s(1:N))],-1:1,N,N);
A = M2+M1+M0; I = speye(N);
% Vektor s auf innere Gitterpunkte beschraenken
s = s(2:N+1);
% Hilfsmatrizen B und C definieren
B = I + k*theta*A; C = I - (1-theta)*k*A;
% Startvektor w definieren (Auszahlungsfunktion)
w = g(s);
sFreierRand = zeros(M);
zeit = zeros(M);
% Loop ueber Zeitpunkte (der Vektor wj wird laufend ueberschrieben)
for j = 1:M
% Loesung des Ungleichungssystems mittels pSOR()
w = pSOR(B,w,C*w,g(s));
% Freien Randwert finden
idx = find(w > g(s),1);
sFreierRand(j)=s(idx);
% Zeitpunkt speichern
zeit(j) = j*k;
```

```
end
% Grafik erstellen
plot(zeit,sFreierRand,'b-')
```

Anhang 8: MATLAB®-Code EurDiv()

```
% Berechnet den Wert einer europaeischen Option zum Kaufzeitpunkt t=T fuer
% N+2 Kurswerte s des Basiswerts mit Beruecksichtigung diskreter Dividenden.
% Gibt Optionswerte im Vektor w und Gitterpunkte von s im Vektor s aus.
% Die Berechnung erfolgt mittels Crank-Nicholson-Verfahren auf einem
% aequidistanten Gitter mit N+2 Punkten in s und M+1 Zeitpunkten sowie
% homogenen Randbedingungen.
%
% Argumente: CallorPut ('c' oder 'p'), Ausuebungspreis K, Restlaufzeit T,
% Volatilitaet sigma, stetiger risikoloser Zinssatz r, Zeilenvektor D
% mit Dividenenbetraegen, Zeilenvektor td mit Dividendenzeitpunkten, linker Rand
% fuer s sl, rechter Rand für s sr, N, M
function [s,w] = EurDiv(CallorPut,K,T,sigma,r,D,td,sl,sr,N,M)
% Crank-Nicholson-Verfahren
theta = 0.5;
% Koeffizienten der Black-Scholes-Gleichung
a = @(s)-0.5*sigma^2*s.^2;
b = @(s)-(r)*s;
c = @(s)r*s.^0;
% Auszahlungsfunktion festlegen
if CallorPut == 'c'
g = @(s)max(0,s-K);
elseif CallorPut == 'p'
g = @(s)max(0,K-s);
end
% Aequidistantes Gitter definieren
h = (sr-sl)/(N+1);
s = (sl:h:sr)';
zeitspanne = flip([td,T]-[0,td]);
k = zeitspanne/M;
dividende = flip([0,D]);
% Matrizen A und I definieren
M0 = spdiags(c(s(2:N+1)),0,N,N);
M1 = 1/(2*h)*spdiags([-b(s(3:N+2)),zeros(N,1),b(s(1:N))],-1:1,N,N);
M2 = 1/h^2*spdiags([a(s(3:N+2)),-2*a(s(2:N+1)),a(s(1:N))],-1:1,N,N);
A = M2+M1+M0; I = speye(N);
s = s(2:N+1);
% Startvektor w definieren (Auszahlungsfunktion)
w = g(s);
% Loop ueber Zeitabschnitte
for z=1:length(zeitspanne)
% Hilfsmatrizen definieren
B = I + k(z)*theta*A; C = I - (1-theta)*k(z)*A;
% Loop ueber Zeitpunkte (der Vektor wj wird laufend ueberschrieben)
for j = 1:M
w = B\(C*w);
end
% Kurssprung
w = interp1(s,w,s-dividende(z),'pchip');
```

```
end
% Randpunkte dazunehmen
s = [sl;s;sr];w = [0;w;0];
```

Anhang 9: MATLAB®-Code AmiDiv()

```
% Berechnet den Wert einer amerikanischen Option zum Kaufzeitpunkt t=T fuer
% N+2 Kurswerte s des Basiswerts mit Beruecksichtigung diskreter Dividenden.
% Gibt Optionswerte im Vektor w und Gitterpunkte von s im Vektor s aus.
% Die Berechnung erfolgt mittels Crank-Nicholson-Verfahren auf einem
% aequidistanten Gitter mit N+2 Punkten in s und M+1 Zeitpunkten sowie
% homogenen Randbedingungen. Greift zur Verwendung des Gauss-Seidel-Verfahrens
% auf die Funktion pSOR() zurueck.
%
% Argumente: CallorPut ('c' oder 'p')Ausuebungspreis K, Restlaufzeit T,
% Volatilitaet sigma, stetiger risikoloser Zinssatz r, Zeilenvektor D
% mit Dividenenbetraegen, Zeilenvektor td mit Dividendenzeitpunkten, linker
% Rand fuer s sl, rechter Rand für s sr, N, M
function [s,w] = AmiDiv(CallorPut,K,T,sigma,r,D,td,sl,sr,N,M)
% Crank-Nicholson-Verfahren
theta = 0.5;
% Koeffizienten der Black-Scholes-Gleichung
a = @(s)-0.5*sigma^2*s.^2;
b = @(s)-(r)*s;
c = @(s)r*s.^0;
% Auszahlungsfunktion festlegen
if CallorPut == 'c'
g = @(s)max(0,s-K);
elseif CallorPut == 'p'
g = @(s)max(0,K-s);
end
% Aequidistantes Gitter definieren
h = (sr-sl)/(N+1);
s = (sl:h:sr)';
zeitspanne = flip([td,T]-[0,td]);
k = zeitspanne/M;
dividende = flip([0,D]);
% Matrizen A und I definieren
M0 = spdiags(c(s(2:N+1)),0,N,N);
M1 = 1/(2*h)*spdiags([-b(s(3:N+2)),zeros(N,1),b(s(1:N))],-1:1,N,N);
M2 = 1/h^2*spdiags([a(s(3:N+2)),-2*a(s(2:N+1)),a(s(1:N))],-1:1,N,N);
A = M2+M1+M0; I = speye(N);
% Vektor s auf innere Gitterpunkte beschraenken
s = s(2:N+1);
% Startvektor w definieren (Auszahlungsfunktion)
w = g(s);
% Loop ueber Zeitabschnitte
for z=1:length(td)+1
% Hilfsmatrizen definieren
B = I + k(z)*theta*A; C = I - (1-theta)*k(z)*A;
% Loop ueber Zeitpunkte (der Vektor wj wird laufend ueberschrieben)
for j = 1:M
% Loesung des Ungleichungssystems mittels pSOR()
w = pSOR(B,w,C*w,g(s));
```

```
end
% Kurssprung
w = max(interp1(s,w,s-dividende(z),'pchip'),g(s));
end
% Randpunkte dazunehmen
s = [sl;s;sr];w = [0;w;0];
```

Anhang 10: MATLAB®-Code AmiDivFreierRandwert()

```
% Stellt den freien Randwert im zeitlichen Verlauf dar mit Beruecksichtigung
% diskreter Dividenden. Die Berechnung erfolgt mittels Crank-Nicholson-Verfahren
% auf einem aequidistanten Gitter mit N+2 Punkten in s und M+1 Zeitpunkten
% und homogenen Randbedingungen. Die Funktion wendet das Gauss-Seidel-Verfahren an,
% indem sie auf pSOR() zurueckgreift.
%
% Argumente: CallorPut ('c' oder 'p'), Ausuebungspreis K, Restlaufzeit T,
% Volatilitaet sigma, stetiger risikoloser Zinssatz r, Zeilenvektor D
% mit Dividenenbetraegen, Zeilenvektor td mit Dividendenzeitpunkten, linker Rand
% fuer s sl, rechter Rand für s sr, N, M
function [tFreierRand,sFreierRand] = AmiDivFreierRandwert(CallorPut,K,T,sigma,r,D,td,sl,sr,N,M)
% Crank-Nicholson-Verfahren
theta = 0.5;
% Koeffizienten der Black-Scholes-Gleichung
a = @(s)-0.5*sigma^2*s.^2;
b = @(s)-(r)*s;
c = @(s)r*s.^0;
% Auszahlungsfunktion festlegen
if CallorPut == 'c'
g = @(s)max(0,s-K);
elseif CallorPut == 'p'
g = @(s)max(0,K-s);
end
% Aequidistantes Gitter definieren
h = (sr-sl)/(N+1);
s = (sl:h:sr)';
zeitspanne = flip([td,T]-[0,td]);
k = zeitspanne/M;
anzahlabschnitte = length(td)+1;
zeitpunkt = 0;
dividende = flip([0,D]);
% Matrizen A und I definieren
M0 = spdiags(c(s(2:N+1)),0,N,N);
M1 = 1/(2*h)*spdiags([-b(s(3:N+2)),zeros(N,1),b(s(1:N))],-1:1,N,N);
M2 = 1/h^2*spdiags([a(s(3:N+2)),-2*a(s(2:N+1)),a(s(1:N))],-1:1,N,N);
A = M2+M1+M0; I = speye(N);
% Vektor s auf innere Gitterpunkte beschraenken
s = s(2:N+1);
% Startvektor w definieren (Auszahlungsfunktion)
w = g(s);
% Vektoren zur Speicherung der freien Randwerte erzeugen
sFreierRand = zeros(anzahlabschnitte*M);
tFreierRand = zeros(anzahlabschnitte*M);
% Loop ueber die Zeitabschnitte
for z=1:anzahlabschnitte
% Hilfsmatrizen definieren
B = I + k(z)*theta*A; C = I - (1-theta)*k(z)*A;
% Loop ueber Zeitpunkte (der Vektor wj wird laufend ueberschrieben)
```

```
for j = 1:M
% Loesung des Ungleichungssystems mittels pSOR()
w = pSOR(B,w,C*w,g(s));
% Freien Randwert finden und speichern
idx = find(w > g(s),1);
sFreierRand((z-1)*M+j)=s(idx);
tFreierRand((z-1)*M+j)=zeitpunkt+k(z)*j;
end
zeitpunkt = zeitpunkt + zeitspanne(z);
% Kurssprung
w = max(interp1(s,w,s-dividende(z),'pchip'),g(s));
end
% Grafik erstellen
plot(tFreierRand,sFreierRand,'b-')
```