UNIVERSITÄT
BAYREUTH

Die Monte-Carlo-Methode mit Verfahren höherer Ordnung

Bachelorarbeit

von

Johanna Weigand

Fakultät für Mathematik, Physik und Informatik
Mathematisches Institut

Datum: 03. August 2012

Aufgabenstellung und Betreuung:
Prof. Dr. L. Grüne

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Diese Arbeit beschäftigt sich mit der Monte-Carlo-Simulation zur Berechnung des Erwartungswertes einer Zufallsvariablen, welche die Lösung einer stochastischen Differentialgleichung ist. Dabei wird die Lösung der stochastischen Differentialgleichung mit Verfahren unterschiedlicher Ordnung approximiert. In diesem Kapitel wird zuerst als Motivation die Optionsbewertung als ein Anwendungsbeispiel aus der Finanzmathematik gegeben. Dazu wird erklärt, was Optionen sind und ein Beispiel gegeben, das die Problemstellung deutlich macht. Daraufhin wird die Problemstellung dargestellt und der Grundalgorithmus der Optionsbewertung aufgestellt. Am Schluss wird ein Überblick über die folgenden Kapitel gegeben. Dieses Kapitel orientiert sich an den Arbeiten von L. Grüne [2], K. Schäfer [6] und M. Günther und A. Jüngel [3].

1.1 Motivation: Optionen

Ein Optionsgeschäft ist ein bedingtes Termingeschäft und zählt zu den derivativen Finanzinstrumenten. Eine europäische Option ist ein Vertrag, der dem Optionsinhaber das Recht (aber nicht die Pflicht) gibt, einen Basiswert S (z.B. Aktien) zu einem genau bestimmten Zeitpunkt (Ausübungszeitpunkt T) zu einem vorher festgelegten Preis (Ausübungspreis K) zu kaufen (Call-Option) oder zu verkaufen (Put-Option).

Ist der Ausübungszeitpunkt (oder auch Fälligkeitszeitpunkt) erreicht, so kann der Optionsinhaber die Option ausüben, d.h. den Basiswert kaufen (verkaufen), oder die Option verfallen lassen, wenn dies für ihn günstiger ist. Der Verkäufer (Stillhalter) der Option hat die Pflicht, den Basiswert gegebenenfalls zu kaufen oder zu verkaufen, wenn der Optionsinhaber dies fordert. In der Praxis wird der Basiswert häufig nicht gekauft (oder verkauft), sondern der tatsächliche Kauf (Verkauf) wird durch Differenzzahlungen substituiert (Barausgleich). Der Käufer der Option zahlt für das Recht eine Prämie, den sogenannten Optionspreis.

Kann eine Option nur zu dem Fälligkeitszeitpunkt T ausgeübt werden, so nennt man die Option europäisch. Kann sie zu jedem beliebigen Zeitpunkt $t \in [0, T]$ ausgeübt werden, so spricht man von einer amerikanischen Option. In dieser Arbeit werden nur europäische Optionen behandelt, da die von uns verwendeten Algorithmen nicht auf amerikanische Optionen anwendbar sind.

Man kann Optionen nutzen, um sich gegen Risiken (z.B. Preisänderungsrisiken) abzusichern (Hedging) oder man kann Risiken übernehmen (spekulieren, Trading).

Doch wie spekuliert man mit einer Option und wie sichert man sich mit Optionen gegen Risiken ab? Hierzu wird zuerst eine Formel für den Optionswert zum Laufzeitende T benötigt.

Definition 1.1 Der Wert einer Call-Option V_C zum Zeitpunkt T ($S(T)$ ist der Kurs des Basiswertes zum Laufzeitende), berechnet sich wie folgt:

$$V_C(T, S(T)) = \max\{S(T) - K, 0\} =: (S(T) - K)^+ \tag{1.1}$$

Liegt der Kurs bei $S > K$, so rentiert es sich die Option auszuüben. Man macht damit einen „Gewinn“ von $S - K$, da man den Basiswert für K vom Stillhalter kauft und ihn sofort wieder für S am Markt verkaufen könnte. Der Wert entspricht also der Differenz zwischen dem Kurswert am Kassamarkt (Börse) und dem Ausübungspreis. Liegt der Kurs S unter K, also $S < K$, so lässt man die Option verfallen, da man den Basiswert am Markt billiger kaufen kann. Somit ist der Optionswert im Zeitpunkt T gleich Null.

Der Wert einer Put-Option zum Zeitpunkt T ergibt sich analog zu:

$$V_P(T, S(T)) = \max\{K - S(T), 0\} =: (K - S(T))^+ \tag{1.2}$$

⊡

Der Käufer einer Call-Option setzt somit auf steigende Kurse, da er bei einem steigenden Kurs den Basiswert billig über die Option beziehen kann und dann gleich wieder teuer zum Marktpreis verkaufen könnte. Der Verkäufer hofft auf fallende Kurse, da er dann nicht liefern muss und die Optionsprämie als Gewinn behält. Bei einem Put verhält es sich genau entgegengesetzt.

Beispiel 1.2 Ein Pralinenhersteller fürchtet einen massiven Preisanstieg am Markt für Kakaobohnen. Er möchte einen Teil seines Bedarfs gegen dieses Preisrisiko absichern und kauft deshalb eine Call-Option (Kaufoption) auf Kakaobohnen. Der Ausübungszeitpunkt T ist der 1. Oktober diesen Jahres und der Ausübungspreis ist $K = 10.000$ Euro für 5 Tonnen Kakaobohnen (=Basiswert). Die Prämie für diese Option beträgt 200 Euro.

Daraus ergeben sich folgende Gewinn- und Verlustprofile für den Pralinenhersteller (Optionsinhaber) und die Bank (Stillhaber, Emittent der Option):

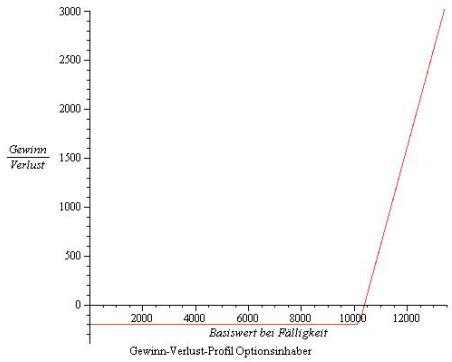

Abbildung 1.1: Gewinn-Verlust-Profil des Optionskäufers

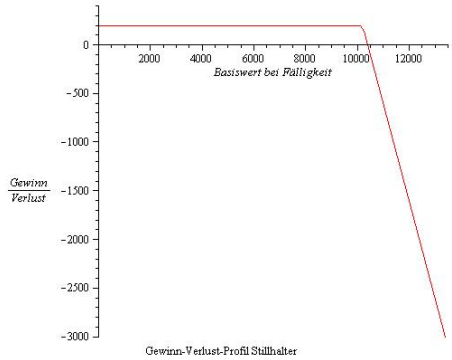

Abbildung 1.2: Gewinn-Verlust-Profil des Optionsverkäufers

Liegt der Kurs für 5 Tonnen Kakaobohnen am 1. Oktober bei 8.000 Euro, dann wird der Pralinenhersteller die Option verfallen lassen, da er für die Bohnen am regulären Markt 2.000 Euro weniger bezahlen muss. Der Wert der Option ist somit 0 Euro und er hat einen Verlust in Höhe der Optionsprämie, da er die Option gekauft hat, aber nicht ausübt.
Liegt der Kurs aber bei 10.100 Euro, so wird er die Option ausüben, da er am Markt 100 Euro mehr bezahlen müsste. Allerdings hat er bereits eine Prämie von 200 Euro gezahlt, sodass er immer noch einen Verlust in Höhe von 100 Euro hat.
Liegt der Kurs nun bei 13.000 Euro, so wird er die Option weiterhin ausüben. Er macht dabei einen Gewinn von $13.000 - 10.000 - 200 = 2.800$ Euro.
Die Begriffe Gewinn und Verlust beziehen sich immer auf die Alternative, keine Option zu kaufen und die 5 Tonnen direkt am Markt zu beziehen. Der Gewinn ist also der Betrag, den er aufgrund der Option weniger zahlen muss.

⊡

1.2 Problemstellung: Bewertung von Optionen

Im letzten Abschnitt haben wir Optionen kennen gelernt und erfahren, dass der Optionskäufer eine Prämie an den Optionsverkäufer zahlen muss. Wie man an Beispiel 1.2 sehr gut sieht, übernimmt der Ausgeber der Option (meistens eine Bank oder Versicherung) einen Teil des Risikos des Optionskäufers. In dem Beispiel macht der Pralinenhersteller bei einem Kurs von $S = 13.000$ Euro einen Gewinn von 2.800 Euro gegenüber der Börse. Denn ab einem Kurs von $S = 10.000$ ist er gegen das Risiko der Preissteigerung

abgesichert, dieses Risiko trägt nun der Optionsverkäufer. Für diese Risikoübernahme bekommt der Stillhalter die Optionsprämie. Doch wie hoch muss die Optionsprämie sein, damit sie dieses Risiko widerspiegelt?

Das heißt wir brauchen den Wert der Option zu dem Zeitpunkt $t_0 = 0$. Dieser Optionswert, meist noch mit einem Gebührenaufschlag versehen, wird fairer Preis genannt. Den fairen Preis kann man zum Beispiel mit dem tatsächlichen Optionspreis an der Börse vergleichen (da Optionen dort wie Aktien gehandelt werden und sich der Preis somit aus Angebot und Nachfrage ergibt), um festzustellen, ob die Optionen überbewertet sind.

Es sind einige Grundannahmen über den Finanzmarkt nötig, mit denen wir uns aber nicht weiter beschäftigen. Nur eine ist für das Verständnis des folgenden Algorithmus wichtig:

- Zinsen werden kontinuierlich mit dem jährlichen Zinssatz r berechnet: für eine Anlage von x Euro für t Jahre ergibt sich der Zins zu $e^{rt}x$.

Im Folgenden werden diese Bezeichnungen verwendet:

$S(t)$	:	Basiswert zur Zeit t mit $S(t) \geq 0 \; \forall \; t \in [0, T]$
$V(t, S)$	:	Preis der Option zur Zeit t abhängig vom aktuellen Basiswert S
$K > 0$	:	vereinbarter Ausübungspreis
$T > 0$	:	vereinbarte Laufzeit

1.2.1 Grundalgorithmus der Optionsbewertung

In diesem Abschnitt wird der Grundalgorithmus der Optionsbewertung vorgestellt. Dieser ermöglicht uns, den Optionswert zu jedem beliebigen Zeitpunkt $t \in [0, T]$ zu berechnen. In den folgenden Kapiteln werden wir diesen Algorithmus modifizieren und spezifizieren.

Für den Grundalgorithmus verwenden wir das Prinzip der risikoneutralen Bewertung für europäische Optionen, das 1973 durch Black, Scholes und Merton eingeführt wurde.

Sie haben gezeigt, dass der Wert einer Option $V(t, S(t))$ zum Zeitpunkt $t < T$ als

$$V(t, S(t)) = e^{r(t-T)} \mathbb{E}[V(T, S(T))]$$

berechnet werden kann. Das heißt, sobald wir $\mathbb{E}[V(T, S(T))]$ kennen, können wir $V(t, S(t))$ für jedes beliebige $t \in [0, T]$ berechnen. Für ausführliche Erläuterungen siehe [2, Kapitel 2].

Algorithmus 1.3 (Grundalgorithmus der Optionsbewertung)

1. Bestimme eine Formel für den Optionswert $V(T, S(T))$ zum Laufzeitende (in Abhängigkeit vom Kurs $S = S(T)$ am Laufzeitende)
2. Bestimme ausgehend vom Basiswert $S(t)$ zur Zeit $t < T$ mit Hilfe eines stochastischen Kursmodells die Zufallsvariable $S(T) = S(T, \omega)$
3. Berechne den Optionswert als den abgezinsten Erwartungswert

$$V(t, S(t)) = e^{r(t-T)}\mathbb{E}[V(T, S(T))]$$

⊡

Für Schritt 1 werden die Formeln aus der Definition 1.1 benutzt. Im weiteren werden wir jedoch nur noch den Erwartungswert des Kurses berechnen, sodass dieser Schritt entfällt.

Schritt 3 hängt von dem stochastischen Modell ab, das für Schritt 2 verwendet wurde. In Schritt 2 werden wir eine stochastische Differentialgleichung mit Hilfe zweier numerischer Verfahren approximieren, um die Zufallsvariable $S(T)$ zu berechnen. Für Schritt 3 wird dann eine Monte-Carlo-Simulation verwendet.

Algorithmus 1.4 (Grundalgorithmus zur Berechnung des Erwartungswertes des Kurses)

1. Bestimme ausgehend vom Basiswert S(t) zur Zeit $t < T$ mit Hilfe eines stochastischen Kursmodells die Zufallsvariable $S(T) = S(T, \omega)$
2. Berechne den Erwartungswert $\mathbb{E}[S(T)]$ des Kurses

⊡

Die Vereinfachung des Grundalgorithmus hat keinen Einfluss auf die Wahl der Methoden. Die im Weiteren behandelten Algorithmen können problemlos auf beide Grundalgorithmen angewendet werden. Der Einfachheit halber werden wir die Vorgehensweise auf den Algorithmus 1.4 anpassen. Der Vorteil ist, dass für den Erwartungswert des Kurses die exakte Lösung bekannt ist und damit bei der numerischen Simulation der Fehler exakt gemessen werden kann.

1.3 Aufbau

In dieser Arbeit werden Methoden und Algorithmen vorgestellt, mit deren Hilfe man den Erwartungswert einer Zufallsvariable, die die Lösung einer stochastischen Differentialgleichung ist, berechnen kann. Die Frage, die mit dieser Arbeit beantwortet werden soll, ist, ob sich der Aufwand für Verfahren höherer Ordnung zur Lösung stochastischer Differentialgleichungen lohnt, oder ob der Fehler der Monte-Carlo-Simulation zur Berechnung des Erwartungswertes zu groß ist.

In Kapitel 2 wird zunächst der Wiener-Prozess vorgestellt, mit dessen Hilfe im Folgenden stochastische Itô-Differentialgleichungen definiert werden. Dann wird die geometrische Brownsche Bewegung als Beispiel für eine einfache stochastische Differentialgleichung definiert. Diese Differentialgleichung hat den großen Vorteil, dass man die exakte Lösung und den Erwartungswert kennt. Deshalb werden wir im Weiteren die geometrische Brownsche Bewegung als Kursmodell für Schritt 1 des Grundalgorithmus 1.4 verwenden.

Das dritte Kapitel behandelt die numerische Lösung von stochastischen Differentialgleichungen. Bevor Algorithmen angegeben werden können, müssen zwei Konvergenzbegriffe definiert werden. Dann werden drei Möglichkeit für die Approximation des Wiener-Prozesses behandelt, die wir für die Lösung der stochastischen Differentialgleichungen benötigen. Am Ende des Kapitels werden zwei ableitungsfreie Verfahren unterschiedlicher Ordnung vorgestellt. Diese zwei Verfahren werden dann auf die geometrische Brownsche Bewegung angepasst. Damit können wir Schritt 1 des Grundalgorithmus 1.4 lösen.

Für Schritt 2 des Grundalgorithmus 1.4 verwenden wir eine Monte-Carlo-Simulation. Diese wird im vierten Kapitel behandelt. Zuerst wird die Grundidee skizziert und der stochastische Hintergrund dazu behandelt. Nachdem der Grundalgorithmus angepasst wurde, beschäftigen wir uns mit einer Analyse der Konvergenzgeschwindigkeit und Konfidenzintervallen. Zum Abschluss lernen wir noch eine Möglichkeit der Varianzreduktion kennen, wodurch wir die Konvergenzgeschwindigkeit erhöhen können.

Damit haben wir alles Nötige, um uns in Kapitel 5 mit der Fehleranalyse beschäftigen zu können. Zuerst wird der Fehler der Monte-Carlo-Simulation zur Bestimmung des Erwartungswertes in Abhängigkeit vom gewählten Verfahren theoretisch abgeschätzt. Dann werden Ergebnisse aus der Implementierung diskutiert. Anschließend werden die Ergebnisse aus Theorie und Praxis miteinander verglichen.

Kapitel 2

Stochastische Differentialgleichungen

Um den Kursverlauf in Schritt 1 des Grundalgorithmus 1.4 zu modellieren, werden wir stochastische Differentialgleichungen verwenden. Dazu wird in diesem Kapitel zuerst eine kurze Einführung zu dem Thema Wiener-Prozess gegeben. Mit Hilfe dieses stochastischen Prozesses werden dann stochastische Differentialgleichungen nach Itô definiert. Am Ende des Kapitels wird die geometrische Brownsche Bewegung als eine spezielle Differentialgleichung vorgestellt, die wir im Weiteren für die Modellierung des Kurses verwenden werden. Für Definitionen und Sätze aus der Stochastik verweise ich auf die Standardliteratur (z.B. [1] und [5]). Dieses Kapitel basiert auf den Arbeiten von L. Grüne [2], M. Günther und A. Jüngel [3] und T. Höllbacher [4].

2.1 Der Wiener-Prozess

Zur Erinnerung: Eine Zufallsvariable heißt Gauß- oder normalverteilt mit Erwartungswert μ und Varianz σ^2 (Notation: $X \sim \mathcal{N}(\mu, \sigma^2)$), falls sie die Dichtefunktion

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

besitzt.

Definition 2.1 Ein (stetiger) stochastischer Prozess X_t, $t \in [0, \infty)$, ist eine Familie von Zufallsvariablen $X : \Omega \times [0, \infty) \to \mathbb{R}$, wobei $t \longmapsto X(\omega, t)$ stetig ist für alle $\omega \in \Omega$. Wir schreiben $X_t = X(t) = X(., t)$, d.h., X_t ist eine Zufallsvariable.

⊡

Für jedes feste $\omega \in \Omega$ ist die Realisierung $X(.,\omega) : \mathbb{R} \to \mathbb{R}$ eine „normale“ reelle Funktion. Solch eine Realisierung nennt man bei stochastischen Prozessen auch Pfad.

Im Weiteren benötigen wir einen bestimmten stochastischen Prozess, den sogenannten Wiener-Prozess (auch Brownsche Bewegung genannt).

Satz 2.2 Es gibt einen stetigen stochastischen Prozess W_t (Wiener-Prozess) mit den Eigenschaften

1. $W_0 = 0$ (P-) fast sicher, d.h. $P(\omega \in \Omega : W_0(\omega) = 0) = 1$,
2. $\forall\ 0 \leq s \leq t$ gilt: $W_t - W_s$ ist $\mathcal{N}(0, t-s)$-verteilt,
3. $\forall\ 0 \leq r \leq u \leq s \leq t$ gilt: $W_t - W_s$ und $W_u - W_r$ sind unabhängig und
4. W_t ist $\mathcal{N}(0,t)$-verteilt.

⊡

Besonders wichtig ist hierbei der Punkt 2, da wir in den folgenden Algorithmen gerade mit solchen Inkrementen rechnen werden.

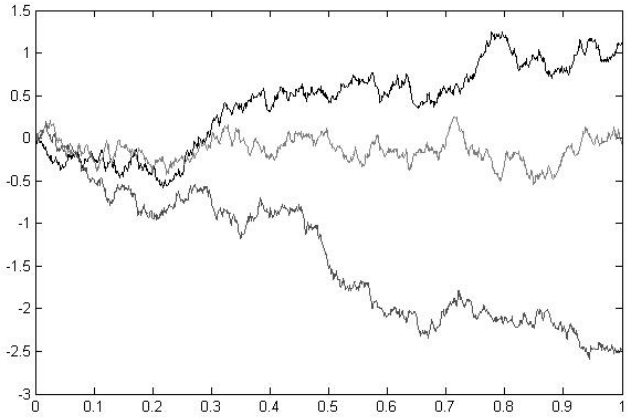

Abbildung 2.1: Verschiedene Pfade des Wiener-Prozesses

2.2 Die stochastische Differentialgleichung nach Itô

Wie man in der Abbildung 2.1 gut erkennen kann, ist der Wiener-Prozess für sich nicht ausreichend, um einen Kursverlauf zu modellieren. Zum einen muss sichergestellt werden, dass $S(t)$ nicht negativ wird und zum anderen möchte man bestimmte Parameter vorgeben können, die den Kursverlauf

beeinflussen. Dies führt uns zu der Definition einer stochastischen Differentialgleichung. Hierbei kommt der Zufall durch den Wiener-Prozess in die Differentialgleichung.

Definition 2.3 Eine stochastische Differentialgleichung im Sinne von Itô ist gegeben durch

$$dX(t) = a(t, X(t))dt + b(t, X(t))dW_t \tag{2.1}$$

wobei $X(t)$ ein stochastischer Prozess, W_t der Wiener-Prozess und a und b geeignete (d.h. hinreichend reguläre) Funktionen seien. Die Gleichung (2.1) ist die symbolische Schreibweise für die Integralgleichung

$$X(t) = X(t_0) + \int_{t_0}^{t} a(\tau, X(\tau))d\tau + \int_{t_0}^{t} b(\tau, X(\tau))dW_\tau. \tag{2.2}$$

Erfüllt ein stochastischer Prozess die Gleichung (2.2), so heißt er Itô-Prozess.

⊡

Wäre jeder Pfad des Wiener-Prozesses differenzierbar, so wäre die Schreibweise

$$\int_{t_0}^{t} dW_\tau$$

eine Kurzschreibweise für

$$\int_{t_0}^{t} \frac{d}{d\tau} W(\tau)d\tau.$$

Da die Pfade normalerweise nicht differenzierbar sind, muss man sich überlegen, was mit $\int_{t_0}^{t} dW_\tau$ genau gemeint ist. Damit hat sich Kiyosi Itô beschäftigt. Ausführliche Informationen zu diesem Thema befinden sich unter anderem in [2, Kapitel 4] und in [5, Kapitel 3].

2.3 Die geometrische Brownsche Bewegung

Die geometrische Brownsche Bewegung ist eine einfache, aber sehr verbreitete stochastische Differentialgleichung zur Modellierung von Kursverläufen. Der große Vorteil dieses Modells ist, dass die exakte Lösung, der Erwartungswert und die Varianz bekannt sind. Im Verlauf dieser Arbeit wird die geometrische Brownsche Bewegung als Kursmodell für Schritt 1 des Grundalgorithmus 1.4 verwendet.

Wir schreiben jetzt wieder $S(t)$ anstatt $X(t)$ für den Kursverlauf. Zum Modellieren stehen uns die zwei Parameter μ und σ zur Verfügung. Der Parameter μ steht für die Rendite und der Parameter σ für die Volatilität.

Das Modell ist gegeben durch:

$$dS(t) = \mu S(t)dt + \sigma S(t)dW_t \tag{2.3}$$

mit der exakten Lösung

$$S(t; S_0) = S_0 \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W(t)\right), \tag{2.4}$$

dem Erwartungswert

$$\mathbb{E}[S(t, S_0)] = S_0 e^{\mu t} \tag{2.5}$$

und der Varianz

$$Var[S(t)] = S_0^2 e^{2\mu t}(e^{\sigma^2 t} - 1). \tag{2.6}$$

Um die geometrische Brownsche Bewegung in der risikoneutralen Optionsbewertung nach Black, Scholes und Merton verwenden zu können, muss gelten: $\mu = r$. Der Parameter σ kann aus den historischen Daten oder aus den Marktdaten geschätzt werden.

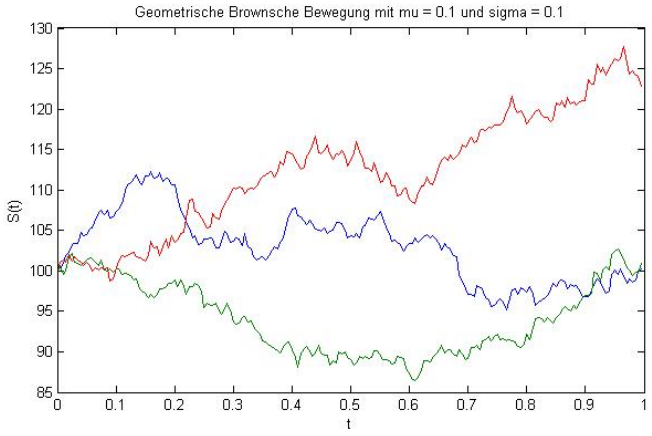

Abbildung 2.2: Mögliche Kursverläufe auf Basis der geometrischen Brownschen Bewegung mit $\mu = 0.1$ und $\sigma = 0.1$, approximiert mit dem Euler-Maruyama-Verfahren aus Kapitel 3.3 mit M=200.

Kapitel 3

Numerische Lösung stochastischer Differentialgleichungen

In diesem Kapitel beschäftigen wir uns mit numerischen Lösungen für stochastische Itô-Differentialgleichungen. Bei komplexen Modellen können wir die stochastische Differentialgleichung nicht mehr exakt lösen, doch durch die zwei in diesem Kapitel behandelten Verfahren haben wir trotzdem eine Möglichkeit, $S(T)$ für Schritt 1 des Grundalgorithmus 1.4 zu berechnen. Zuerst müssen wir uns jedoch mit zwei Konvergenzbegriffen vertraut machen (Abschnitt 3.1). Dann lernen wir drei Approximationen für den Wiener-Prozess kennen (Abschnitt 3.2) und schließlich zwei Verfahren zur Approximation der stochastischen Differentialgleichung in den Abschnitten 3.3 und 3.4. Dieses Kapitel orientiert sich an den Arbeiten von P. Kloeden und E. Platen [5] und L. Grüne [2].

3.1 Konvergenzbegriffe

Die Frage, die sich uns nun stellt, ist: Was ist in unserem Kontext überhaupt eine numerische Approximation? Da wir mit Zufallsvariablen arbeiten, ist nicht jede Approximation eines Pfades gut, doch für unsere Anwendung reicht es, wenn die Approximationen im Mittel hinreichend gut sind.

Wir betrachten einen stochastischen Prozess X auf dem Intervall $[0, T]$. Dieser soll durch eine Folge numerischer Approximationen $\tilde{X}_j$ auf Zeitgittern der Form $\mathcal{T}_j = \{0, \bar{h}_j, 2\bar{h}_j, ..., M_j\bar{h}_j\}$ mit konstanten Schrittweiten $\bar{h}_j = T/M_j$ approximiert werden mit $\lim_{j\to\infty} M_j = \infty$. Wir betrachten die Approximation nur zum Zeitpunkt T und schreiben dazu kurz: $X(T) = X(T, \omega)$ und $\tilde{X}_j(T) = \tilde{X}_j(T, \omega)$.

Definition 3.1
(i) Die Folge $\tilde{X}_j$ von stochastischen Prozessen heißt starke Approximation für X zur Zeit T bzgl. einer Funktion $g : \mathbb{R}^n \to \mathbb{R}^m$, wenn die Bedingung

$$\lim_{j\to\infty} \mathbb{E}\left\|g(X(T)) - g(\tilde{X}_j(T))\right\| = 0$$

gilt. Sie heißt starke Approximation mit Ordnung $\gamma > 0$, falls für alle $j \geq j_0$ zusätzlich die Abschätzung

$$\mathbb{E}\left[\left\|g(X(T)) - g(\tilde{X}_j(T))\right\|\right] \leq Ch_j^\gamma$$

für ein $C > 0$ gilt.
(ii) Die Folge $\tilde{X}_j$ von stochastischen Prozessen heißt schwache Approximation für X zur Zeit T bzgl. einer Funktion $g : \mathbb{R}^n \to \mathbb{R}^m$, wenn die Bedingung

$$\lim_{j\to\infty} \left\|\mathbb{E}[g(X(T))] - \mathbb{E}[g(\tilde{X}_j(T))]\right\| = 0$$

gilt. Sie heißt schwache Approximation mit Ordnung $\beta > 0$, falls für alle $j \geq j_0$ zusätzlich die Abschätzung

$$\left\|\mathbb{E}[g(X(T))] - \mathbb{E}[g(\tilde{X}_j(T))]\right\| \leq Ch_j^\beta$$

für ein $C > 0$ gilt.

⊡

Dabei ist der Begriff der starken Approximation stärker als der Begriff der schwachen Approximation. Das heißt, ist $\tilde{X}$ eine starke Approximation der Ordnung γ, so ist $\tilde{X}$ auch eine schwache Approximation der Ordnung $\beta \geq \gamma$. Für einen Beweis siehe [2, Kapitel 7].

3.2 Approximation des Wiener-Prozesses

Wie wir in Kapitel 2 gesehen haben, benötigen wir einen Wiener-Prozess für die Definition der stochastischen Differentialgleichung. Folglich wird auch in der Lösung der Differentialgleichungen ein Wiener-Prozess benötigt. Hier werden wir einen Algorithmus für eine starke Approximation und zwei Algorithmen für eine schwache Approximation des Wiener-Prozesses kennen lernen. Diese werden wir später für unsere zwei Verfahren zur Lösung von stochastischen Differentialgleichungen verwenden.

Um später den Erwartungswert berechnen zu können, benötigen wir N Lösungen für $S(T)$ und somit auch N Gitterfunktionen des Wiener-Prozesses. M ist die Anzahl an Gitterpunkten pro Gitterfunktion, die Schrittweite ist also $h = T/M$.

Algorithmus 3.2 (Starke Approximation des Wiener-Prozesses durch Gitterfunktion)
Gegeben: Schrittweite h, Gitter $\mathcal{T} = \{t_0, ..., t_M\}$ mit $t_j = jh$.
Gesucht: N approximierende Pfade $\tilde{W}(t_j, \omega_1), ..., \tilde{W}(t_j, \omega_N)$ des Wiener-Prozesses auf $\mathcal{T}$.

(1) Für i = 1,...,N:
  (2a) Erzeuge unabhängige $\mathcal{N}(0, h)$-verteilte Zufallszahlen $\Delta W_j(\omega_i)$ für $j = 0, ..., M-1$.
  (2b) Erzeuge Gitterfunktionen $\tilde{W}(t_j, \omega_i)$ mittels der Rekursion

$$\tilde{W}(t_0, \omega_i) = 0$$

$$\tilde{W}(t_{j+1}, \omega_i) = \tilde{W}(t_j, \omega_i) + \Delta W_j(\omega_i), j = 0, ..., M-1$$

(3) Ende der i-Schleife

⊡

Für diesen Algorithmus beziehen wir uns auf Satz 2.2. In Punkt 2 haben wir dort gesagt, dass für die Inkremente ΔW des Wiener Prozesses gilt: $\Delta W \sim \mathcal{N}(0, h)$.

Für die erste schwache Approximation benötigen wir zweipunktverteilte Zufallsvariablen mit

$$P\left(\Delta W_j(\omega_i) = \pm\sqrt{h}\right) = \frac{1}{2} \tag{3.1}$$

Dieser Wiener-Prozess wird später für das Verfahren erster Ordnung (Algorithmus 3.4) verwendet.

Algorithmus 3.3 (Schwache Approximation des Wiener-Prozesses durch Gitterfunktion)
Gegeben: Schrittweite h, Gitter $\mathcal{T} = \{t_0, ..., t_M\}$ mit $t_j = jh$.
Gesucht: N schwach approximierende Pfade $\tilde{W}(t_j, \omega_1), ..., \tilde{W}(t_j, \omega_N)$ des Wiener-Prozesses.

(1) Für i = 1,...,N:
  (2a) Erzeuge unabhängige zweipunktverteilte Zufallszahlen $\Delta W_j(\omega_i)$, für die die Gleichung (3.1) gilt für $j = 0, ..., M-1$.
  (2b) Erzeuge Gitterfunktionen $\tilde{W}(t_j, \omega_i)$ mittels der Rekursion

$$\tilde{W}(t_0, \omega_i) = 0$$

$$\tilde{W}(t_{j+1}, \omega_i) = \tilde{W}(t_j, \omega_i) + \Delta W_j(\omega_i), j = 0, ..., M-1$$

(3) Ende der i-Schleife

⊡

Für die zweite schwache Approximation benötigen wir dreipunktverteilte Zufallsvariablen mit

$$P\left(\Delta W_j(\omega_i) = \pm\sqrt{3h}\right) = \frac{1}{6} \quad \text{und} \quad P\left(\Delta W_j(\omega_i) = 0\right) = \frac{2}{3}. \tag{3.2}$$

Diese Approximation des Wiener-Prozesses wird für das Verfahren zweiter Ordnung (Absatz 4) verwendet. Der Algorithmus ist der gleiche wie für zweipunktverteilte Zufallsvariablen, nur in Schritt (2a) werden die Zufallszahlen nach der Gleichung (3.2) erzeugt.

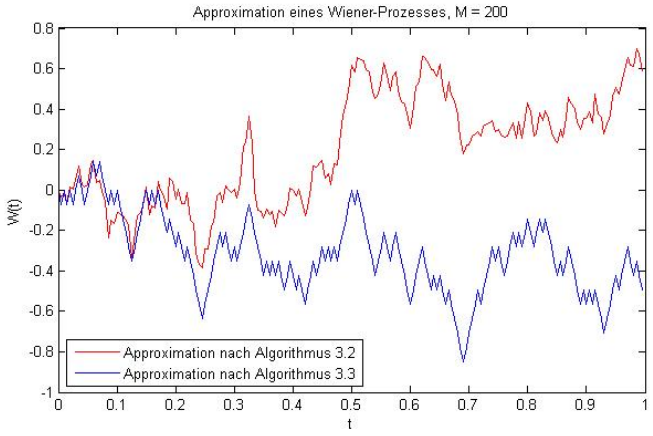

Abbildung 3.1: Vergleich der Algorithmen 3.2 und 3.3, randn('state',2^{17}), M=200

Damit sind nun alle Vorbereitungen abgeschlossen und wir können uns den Verfahren zur Lösung stochastischer Differentialgleichungen zuwenden.

3.3 Verfahren 1. Ordnung

Eines der einfachsten Verfahren für Itô-Differentialgleichungen ist das sogenannte Euler-Maruyama-Verfahren, das zu der Gruppe der stochastischen Einschritt-Verfahren gehört. Verwendet man eine schwache Approximation des Wiener-Prozesses für dieses Verfahren, so besitzt es die schwache Approximationsordnung $\beta = 1$ bezüglich C_P^4-Funktionen g. Einen Beweis dafür findet man in [5, Kap. 14.1].

Der Raum C_P^4 ist die Menge aller Funktionen, die 4-mal stetig differenzierbar sind und deren partielle Ableitungen bis zum Grad 4 in der Norm durch ein Polynom P beschränkt sind.

Wir betrachten nun die skalare Itô-Differentialgleichung

$$dX(t) = a(t, X(t)) + b(t, X(t))dW_t$$

mit dem Anfangswert $X(t_0) = X_0$.

Für ein Gitter $\mathcal{T} = \{t_0, ..., t_M\}$ mit $t_M = T$ ist die Euler-Approximation gegeben durch

$$\tilde{X}(t_{j+1}, \omega_i) = \tilde{X}(t_j, \omega_i) + a(t_j, \tilde{X}(t_j, \omega_i))(t_{j+1} - t_j) + b(t_j, \tilde{X}(t_j, \omega_i))\Delta W_j(\omega_i)$$

mit

$$\Delta W_j(\omega_i) = W(t_{j+1}, \omega_i) - W(t_j, \omega_i)$$

Wählt man $b \equiv 0$, so erhält man das Euler-Verfahren für gewöhnliche Differentialgleichungen.

Wir wollen nun den Algorithmus speziell für die geometrische Brownsche Bewegung aufschreiben. Zur Erinnerung, sie ist gegeben durch (2.3)

$$dS(t) = \mu S(t)dt + \sigma S(t)dW_t.$$

Somit ist

$$a(t, X(t)) = \mu X(t)$$

und

$$b(t, X(t)) = \sigma X(t).$$

Im Folgenden wollen wir äquidistante Schrittweiten betrachten, also $t_{j+1} - t_j = h$. Wie im vorherigen Teilkapitel gilt auch hier, dass wir für die Berechnung des Erwartungswertes des Kurses N Lösungen für $S(T)$ benötigen.

Algorithmus 3.4 (Lösung der geometrischen Brownschen Bewegung mit dem Euler-Maruyama-Verfahren)
Gegeben: Schrittweite h, Gitter $\mathcal{T} = \{t_0, ..., t_M\}$ mit $t_j = jh$
Gesucht: N approximative Pfade $\tilde{X}(., \omega_1), ..., \tilde{X}(., \omega_N)$ für (2.3) auf $\mathcal{T}$ mit $X(t_0, \omega_i) = X_0$

(0) Erzeuge N approximative Pfade $\tilde{W}(., \omega_i)$ für $i = 1, ..., N$ auf $\mathcal{T}$ für den Wiener-Prozess W.
(1) Für $i = 1, ..., N$:
  (2) Erzeuge Gitterfunktionen $\tilde{X}(t_j, \omega_i)$ mittels der Rekursion

$$\tilde{X}(t_0, \omega_i) = X_0$$

$$\tilde{X}(t_{j+1}, \omega_i) = \tilde{X}(t_j, \omega_i) + h\mu\tilde{X}(t_j, \omega_i) + \sigma\tilde{X}(t_j, \omega_i)\Delta W_j(\omega_i)$$

  für $j = 0, ..., M - 1$
(3) Ende der i-Schleife

⊡

Für die Approximation des Wiener-Prozesses können wir die starke Approximation nach Algorithmus 3.2 oder die schwache Approximation mit zweipunktverteilten Zufallsvariablen verwenden.

3.4 Verfahren 2. Ordnung

Als Verfahren höherer Ordnung wählen wir ein ableitungsfreies Schema. Verwendet man eine schwache Approximation für den Wiener-Prozess, so ist das Verfahren eine schwache Approximation der Ordnung $\beta = 2$. Wählt man $b \equiv 0$, so erhält man das Heun-Verfahren für gewöhnliche Differentialgleichungen, deshalb werden wir das Verfahren im Weiteren Heun-Verfahren nennen.

Als Kurzschreibweise für $\tilde{X}(t_j, \omega_i)$ wird $\tilde{X}_j$ verwendet und für $\Delta W_j(\omega_i)$ schreiben wir $\Delta\tilde{W}$.

$$\begin{aligned}\tilde{X}_{j+1} = \tilde{X}_j &+ \frac{1}{2}(a(\bar{Y}_j) + a(\tilde{X}_j))h \\ &+ \frac{1}{4}(b(\bar{Y}_j^+) + b(\bar{Y}_j^-) + 2b(\tilde{X}_j))\Delta\tilde{W} \\ &+ \frac{1}{4}(b(\bar{Y}_j^+) - b(\bar{Y}_j^-))((\Delta\tilde{W})^2 - h)h^{-1/2} \qquad (3.3)\end{aligned}$$

mit

$$\bar{Y}_j = \tilde{X}_j + ha(\tilde{X}_j) + b(\tilde{X}_j)\Delta\tilde{W}$$

und

$$\bar{Y}_j^{\pm} = \tilde{X}_j + ha(\tilde{X}_j) \pm b(\tilde{X}_j)\sqrt{h}$$

Wollen wir diesen Algorithmus für die geometrische Brownsche Bewegung verwenden, so setzen wir $a(t, X(t))$ und $b(t, X(t))$ wie im vorigen Abschnitt.

Für die $\Delta\tilde{W}$ kann man entweder die starke Approximation nach Algorithmus 3.2 verwenden oder eine schwache Approximation mit dreipunktverteilten Zufallsvariablen. Die in Kapitel 3.2 genannte schwache Approximation mit zweipunktverteilten Zufallsvariablen ist nicht zulässig.

Wie man bereits hier sieht, werden die Verfahren mit steigender Ordnung komplexer und umfangreicher. Damit steigt der Aufwand erheblich, sowohl in der Entwicklung der Verfahren als auch bei der Rechenzeit. Da man für die Berechnung des Erwartungswertes des Kurses viele solcher Schritte benötigt $(N \cdot M)$, macht sich dieser Mehraufwand stark bemerkbar. Dies werden wir in Kapitel 5 genauer untersuchen.

Kapitel 4

Die Monte-Carlo-Simulation

In diesem Kapitel wird eine Einführung in die Monte-Carlo-Simulation zur Berechnung des Erwartungswertes gegeben. Der Vorteil der Monte-Carlo-Methode ist, dass wir sie auch für komplexere Kursmodelle als die geometrische Brownsche Bewegung verwenden können, bei denen es keine expliziten Formeln für die Lösung der stochastischen Differentialgleichung gibt. Das heißt, wir können in Schritt 1 des Grundalgorithmus 1.4 anstatt der exakten eine numerische Lösung der stochastischen Differentialgleichung verwenden und dann mit der Monte-Carlo-Simulation in Schritt 2 den Erwartungswert von $S(T)$ (bzw. den Optionswert $\mathbb{E}[V(T, S(T))]$) berechnen. Dieses Kapitel basiert auf den Arbeiten von L. Grüne [2], T. Höllbacher [4] und H. Georgii [1].

4.1 Grundidee

Die Grundidee der Monte-Carlo-Methode ist, dass wir mit Hilfe des Computers Zufallszahlen erzeugen und damit den Erwartungswert einer Zufallsvariable schätzen. Dass diese Herangehensweise überhaupt funktioniert, zeigt folgender Satz.

Satz 3.1 (Zentraler Grenzwertsatz) Sei $(X_i)_{i\geq 1}$ eine Folge von unabhängigen, identisch verteilten Zufallsvariablen mit Erwartungswert $\mathbb{E}[X_i] = \mu$ und Varianz $Var[X_i] = \sigma^2$. Dann gilt:

$$S_n = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \frac{X_i - \mu}{\sigma} = \frac{\sum_{i=1}^{n} X_i - n\mu}{\sqrt{n}\sigma} \longrightarrow \mathcal{N}(0,1) \tag{4.1}$$

⊡

Für einen Beweis siehe [1, Kapitel 5.3]

Daraus folgt, dass für hinreichend große n $Y_n := \sum_{i=1}^{n} X_i$ approximativ normalverteilt ist mit Erwartungswert $\mathbb{E}[Y_n] = n\mu$ und Varianz $Var[Y_n] = n\sigma^2$, d.h. $Y_n \sim \mathcal{N}(n\mu, n\sigma^2)$.

Bemerkenswert an dieser Stelle ist, dass keine Voraussetzungen an die Verteilung der X_i gemacht worden ist, sie müssen also nicht normalverteilt sein.

Wenn wir nun den Erwartungswert μ einer Zufallsvariable X schätzen wollen, so erzeugen wir Stichproben $X_i \sim X$. Für große N gilt dann mit hoher Wahrscheinlichkeit (siehe 3.1):

$$\frac{1}{N}\sum_{i=1}^{N} X_i \approx \mu = \mathbb{E}[X]$$

Genauer wird das in Abschnitt 4.2 untersucht. Dies ist die Grundidee der Monte-Carlo-Simulation. Der Erwartungswert einer Zufallsvariablen wird approximiert, indem man N Zufallszahlen erzeugt und darüber den Mittelwert bildet. Dabei gilt, dass mit steigendem N auch die Genauigkeit der Approximation steigt.

Mit dieser Erkenntnis können wir nun Schritt 3 des Grundalgorithmus 1.3 anpassen:

3. Simuliere N Kurswerte $S(T, \omega_1), ..., S(T, \omega_N)$ und approximiere den Optionswert als den abgezinsten Mittelwert

$$V(t, S(t)) = e^{r(t-T)}\frac{1}{N}\sum_{i=1}^{N} V(T, S(T, \omega_i)).$$

Da wir im folgenden nicht den Optionswert, sondern den Erwartungswert von $S(T)$ berechnen wollen, modifizieren wir Schritt 2 des Grundalgorithmus 1.4 wie folgt:

2. Simuliere N Kurswerte $S(T, \omega_1), ..., S(T, \omega_N)$ und approximiere den Erwartungswert als den Mittelwert

$$\mathbb{E}[S(T)] = \frac{1}{N}\sum_{i=1}^{N} S(T, \omega_i).$$

Damit ist unser Grundalgorithmus 1.4 vollständig. Bevor wir uns im nächsten Kapitel mit einer Fehlerabschätzung des Grundalgorithmus 1.4 beschäftigen, brauchen wir jedoch erst einmal eine Aussage über die Genauigkeit der von uns verwendeten Monte-Carlo-Simulation.

4.2 Konvergenzgeschwindigkeit

Definition 4.2 Gegeben sei $p \in [0,1]$, $N \in \mathbb{N}$. Ein Intervall der Form $I = [\mu - \epsilon, \mu + \epsilon]$ heißt p-Konfidenzintervall der Monte-Carlo-Methode, wenn

$$P\left(\frac{1}{N}\sum_{i=1}^{N} X_i \in I\right) = p$$

ist.

⊡

Man kann also ein Intervall angeben, in dem der Erwartungswert mit z.B. der Wahrscheinlichkeit 95% liegt. Für die Fehlerabschätzung in Kapitel 5 benötigen wir nun eine genauere Aussage über die Form des p-Konfidenzintervalls. Diese liefert uns der folgende Satz:

Satz 4.3 Gegeben sei $p \in (0,1)$. Dann existiert $k > 0$ und eine Folge $p_N \longrightarrow p$, sodass die p_N-Konfidenzintervalle I_N von der Form

$$I_N = \left[\mu - \frac{k\sigma}{\sqrt{N}}, \mu + \frac{k\sigma}{\sqrt{N}}\right]$$

sind mit $\mu = \mathbb{E}[X_i]$ und $\sigma^2 = Var[X_i]$.

Beweis: Wähle $x \in \mathbb{R}$ so, dass für die Verteilungsfunktion der Standard-Normalverteilung

$$F(x) - F(-x) = p$$

gilt. Auf Grund der Konvergenz

$$\lim_{N\to\infty} P\left(\frac{Y_N - N\mu}{\sigma\sqrt{N}} \leq x\right) = F(x)$$

existiert für jedes $x \in \mathbb{R}$ eine Folge $\delta_N(x) \longrightarrow 0$ mit

$$\left|P\left(\frac{Y_N - N\mu}{\sigma\sqrt{N}} \leq x\right) - F(x)\right| = \delta_N(x).$$

Damit erhalten wir

$$\begin{aligned} P\left(\left|\frac{Y_N - N\mu}{\sigma\sqrt{N}}\right| \leq x\right) &= F(x) - F(-x) + \delta_N(x) - \delta_N(-x) \\ &= p + \delta_N(x) - \delta_N(-x) =: p_N. \end{aligned}$$

Sicherlich gilt mit dieser Definition $p_N \longrightarrow p$. Zudem gilt

$$\left|\frac{Y_N - N\mu}{\sigma\sqrt{N}}\right| \leq x \Leftrightarrow \frac{Y_N}{N} = \frac{1}{N}\sum_{i=1}^{N} X_i \in \left[\mu - \frac{x\sigma}{\sqrt{N}}, \mu + \frac{x\sigma}{\sqrt{N}}\right].$$

Die Behauptung folgt also mit $k = x$.

$\boxdot$

Bei festem p ist das Intervall also von $\sqrt{N}$ abhängig, was auf eine langsame Konvergenz hinweist.

4.3 Varianzreduktion mit antithetische Zufallszahlen

Das p-Konfidenzintervall hängt nicht nur von $\sqrt{N}$ ab, sondern der Fehler ist auch proportional zu der Varianz $Var[S(t)] = \sigma^2$. Dies legt nahe, dass eine Varianzreduktion die Konvergenzgeschwindigkeit deutlich verbessern kann. In diesem Abschnitt beschäftigen wir uns mit einer bestimmten Form der Varianzreduktion, den antithetischen Zufallsvariablen.

Bisher haben wir in der Monte-Carlo-Simulation die Zufallsvariable $S(T) = f(Z)$ verwendet, wobei Z standardnormalverteilt ist. Denn wir verwenden einen Wiener-Prozess für die Lösung der Differentialgleichung, und wie bereits bekannt ist, ist $W_T \sim \mathcal{N}(0, T)$ und lässt sich somit schreiben als $\sqrt{T}Z$ mit $Z \sim \mathcal{N}(0, 1)$.

Würden wir zum Beispiel mit der exakten Lösung (2.4) rechnen, so wäre

$$f(Z) = S_0 \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right)T + \sigma\sqrt{T}Z\right).$$

Das Ziel ist es nun, eine Funktion $\hat{f}$ zu finden, für die $\mathbb{E}[f(Z)] = \mathbb{E}[\hat{f}(Z)]$ und $Var[\hat{f}(Z)] < Var[f(Z)]$ gilt. Eine Möglichkeit ist die antithetische Zufallsvariable zu Z, die gerade durch $-Z$ gegeben ist. Unser neues $\hat{f}$ ist dann gegeben durch

$$\hat{f}(Z) = \frac{f(Z) + f(-Z)}{2}.$$

In der Monte-Carlo-Simulation wird somit

$$\frac{1}{N}\sum_{i=1}^{N} f(Z(\omega_i)) \qquad \text{durch} \qquad \frac{1}{N}\sum_{i=1}^{N} \frac{f(Z(\omega_i)) + f(-Z(\omega_i))}{2}$$

ersetzt.

Wird die geometrische Brownsche Bewegung als Kursmodell verwendet, so kann man beweisen, dass der Erwartungswert gleich bleibt und die Varianz sich mindestens halbiert. Für einen Beweis siehe [2, Kapitel 5].

Da wir nicht mit der exakten Lösung rechnen, sondern mit numerischen Approximationen, stellt sich nun die Frage nach der Umsetzung im Algorithmus. Wir erzeugen zuerst unsere Pfade für den Wiener-Prozess. Diese verwenden wir einmal wie gewohnt, allerdings erzeugen wir jetzt mit jedem Pfad zwei Lösungen der stochastischen Differentialgleichung. Denn wir verwenden den Pfad ein zweites Mal, indem wir jedes ΔW negieren und dann mit diesen $-\Delta W$ eine zweite Lösung berechnen. Somit erhalten wir die doppelte Anzahl an Lösungen. Damit wir die Ergebnisse trotzdem vergleichen können, wird in dem Algorithmus mit Varianzreduktion nur mit $\frac{N}{2}$ gerechnet. Somit ergeben sich bei dem normalen und bei dem modifizierten Algorithmus genau N Lösungen der geometrischen Brownschen Bewegung.

Als Beispiel wird hier das Euler-Maruyama-Verfahren aufgeführt; das Heun-Verfahren wird analog verändert.

Algorithmus 4.4 (Lösung der geometrischen Brownschen Bewegung mit dem Euler-Maruyama-Verfahren und Varianzreduktion)
Gegeben: Schrittweite h, Gitter $\mathcal{T} = \{t_0, ..., t_M\}$ mit $t_j = jh$
Gesucht: N approximative Pfade $\tilde{X}(., \omega_1), ..., \tilde{X}(., \omega_N)$ für (4.1) auf $\mathcal{T}$ mit $X(t_0, \omega_i) = X_0$
(0) Erzeuge $\frac{N}{2}$ approximative Pfade $\tilde{W}(., \omega_i)$ für $i = 1, ..., \frac{N}{2}$ auf $\mathcal{T}$ für den Wiener Prozess W.
(1) Für $i = 1, ..., \frac{N}{2}$:
(2a) Erzeuge Gitterfunktionen $\tilde{X}(t_j, \omega_i)$ mittels der Rekursion

$$\tilde{X}(t_0, \omega_i) = X_0$$

$$\tilde{X}(t_{j+1}, \omega_i) = \tilde{X}(t_j, \omega_i)(1 + h\mu + \sigma \Delta W_j(\omega_i))$$

für $j = 0, ..., M - 1$
(2b) Erzeuge Gitterfunktionen $\tilde{X}(t_j, \omega_i)$ mittels der Rekursion

$$\tilde{X}(t_0, \omega_i) = X_0$$

$$\tilde{X}(t_{j+1}, \omega_i) = \tilde{X}(t_j, \omega_i)(1 + h\mu - \sigma \Delta W_j(\omega_i))$$

für $j = 0, ..., M - 1$
(3) Ende der i-Schleife

⊡

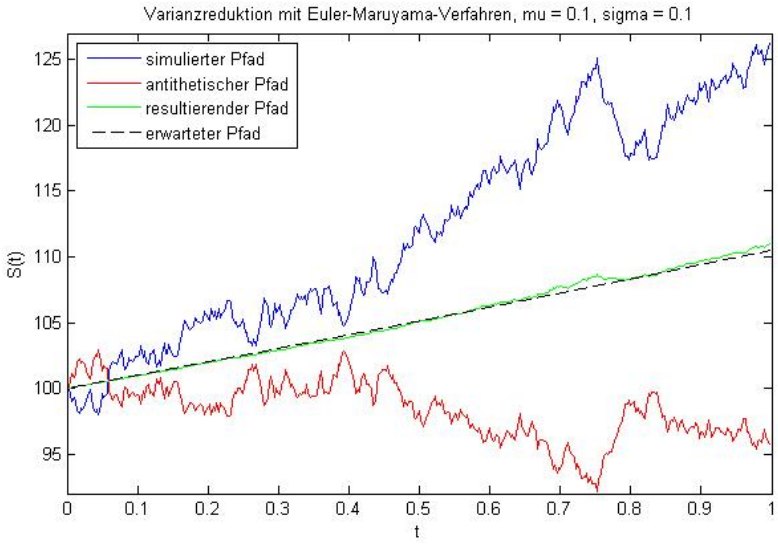

Abbildung 4.1: Simulierter Pfad mit antithetischem Pfad, resultierendem Pfad ($\hat{f}(Z)$) und erwartetem Pfad (nach (2.5)), $\mu = 0.1$ und $\sigma = 0.1$, approximiert mit Algorithmus 4.4, N=1, M=400, $S_0 = 100$

Kapitel 5

Fehleranalyse

Im ersten Kapitel haben wir den Grundalgorithmus 1.4 definiert. Im zweiten Kapitel haben wir uns mit stochastischen Differentialgleichungen beschäftigt und die geometrische Brownsche Bewegung als Kursmodell für Schritt eins des Grundalgorithmus gewählt. Um Schritt eins implementieren zu können, sind im dritten Kapitel zwei numerische Lösungsverfahren eingeführt worden. Diese numerischen Lösungen verwenden wir dann in einer Monte-Carlo-Simulation (Kapitel vier). Dies ermöglicht eine Approximation des Erwartungswertes für Schritt zwei des Grundalgorithmus.

Im Folgenden wollen wir nun analytisch und durch Versuche am Computer herausfinden, ob sich der Mehraufwand für das Verfahren zweiter Ordnung lohnt, oder ob der Fehler der Monte-Carlo-Simulation bereits so groß ist, dass er alle Genauigkeitsbemühungen durch Verfahren höherer Ordnung zunichte macht. Die theoretische Analyse werden wir im nächsten Absatz behandeln und danach werden wir mit Hilfe von Matlab einige Tests durchführen.

Zur Erinnerung skizzieren wir den Algorithmus, den wir nun untersuchen, nochmals.

1. Berechne N Werte für $S(T)$ mit Hilfe der Verfahren aus den Abschnitten 3.3 und 3.4.

2. Berechne den Erwartungswert als

$$\mathbb{E}[S(T)] = \frac{1}{N} \sum_{i=1}^{N} S(T).$$

5.1 Theoretische Fehlerabschätzung

Die Frage ist nun, welche Fehler sich ergeben, wenn wir für die Monte-Carlo-Simulation die numerischen Lösungen verwenden und wie wir N anpassen

müssen, damit sich bei steigender Ordnung auch ein kleineres Konfidenzintervall ergibt.

Aus der Definition 4.2 und dem Satz 4.3 folgt:

Mit Wahrscheinlichkeit p_N gilt:

$$\frac{1}{N}\sum_{i=1}^{N}\tilde{S}_i(T) \in [\mathbb{E}[\tilde{S}_i(T)] - \frac{k\sigma}{\sqrt{N}}, \mathbb{E}[\tilde{S}_i(T)] + \frac{k\sigma}{\sqrt{N}}] \tag{5.1}$$

mit $\tilde{S}_i(T) = \tilde{S}_j(T, \omega_i)$ ist eine numerische Approximation von $S(T)$ mit Schrittweite $h_j = T/M_j$ und $\sigma^2 = Var[\tilde{S}_i(T)]$.

Sei nun $\tilde{S}_i(T)$ eine schwache Approximation der Ordnung β (nach Definition 3.1).

$$\Longrightarrow \|\mathbb{E}[S(T)] - \mathbb{E}[\tilde{S}_i(T)]\| \leq Ch_j^{\beta} \tag{5.2}$$

Satz 5.1 Es gelte (5.1) und (5.2). Dann gilt mit mindestens der Wahrscheinlichkeit p_N:

$$\frac{1}{N}\sum_{i=1}^{N}\tilde{S}_i(T) \in \left[\mathbb{E}[S(T)] - \frac{k\sigma}{\sqrt{N}} - Ch_j^{\beta}, \mathbb{E}[S(T)] + \frac{k\sigma}{\sqrt{N}} + Ch_j^{\beta}\right] \tag{5.3}$$

Beweis: Aus (5.2) folgt:
Fall 1: $\mathbb{E}[S(T)] \geq \mathbb{E}[\tilde{S}_i(T)]$
$\Rightarrow \mathbb{E}[S(T)] - \mathbb{E}[\tilde{S}_i(T)] \leq Ch_j^{\beta}$ und daraus folgt:

1. $\mathbb{E}[\tilde{S}_i(T)] \geq \mathbb{E}[S(T)] - Ch_j^{\beta}$
2. $\mathbb{E}[\tilde{S}_i(T)] \leq \mathbb{E}[S(T)] + Ch_j^{\beta}$

Fall 2: $\mathbb{E}[S(T)] \leq \mathbb{E}[\tilde{S}_i(T)]$
$\Rightarrow \mathbb{E}[\tilde{S}_i(T)] - \mathbb{E}[S(T)] \leq Ch_j^{\beta}$ und auch daraus folgt:

1. $\mathbb{E}[\tilde{S}_i(T)] \geq \mathbb{E}[S(T)] - Ch_j^{\beta}$
2. $\mathbb{E}[\tilde{S}_i(T)] \leq \mathbb{E}[S(T)] + Ch_j^{\beta}$

Aus (5.1) folgt dann mit 1. und 2.

$$\left[\mathbb{E}[\tilde{S}_i(T)] - \frac{k\sigma}{\sqrt{N}}, \mathbb{E}[\tilde{S}_i(T)] + \frac{k\sigma}{\sqrt{N}}\right] \subseteq \left[\mathbb{E}[S(T)] - \frac{k\sigma}{\sqrt{N}} - Ch_j^{\beta}, \mathbb{E}[S(T)] + \frac{k\sigma}{\sqrt{N}} + Ch_j^{\beta}\right]$$

und daraus folgt die Behauptung.

$\boxdot$

Damit haben wir zwei Einflussgrößen auf die Länge des Konfidenzintervalls:
i) $\frac{k\sigma}{\sqrt{N}}$, beeinflussbar durch die Größe N aus der Monte-Carlo-Simulation und
ii) Ch_j^β, beeinflussbar durch die Wahl des Approximationverfahrens und durch die Wahl der Schrittweite h.

Betrachtet man das Intervall aus (5.3), so kommt man auf folgende Überlegung: Um den Aufwand zu minimieren, sollte man N so wählen, dass $\frac{k\sigma}{\sqrt{N}} \approx Ch_j^\beta$. Ist N kleiner, so lohnt sich der Aufwand für das Verfahren höherer Ordnung nicht; ist N größer, so wird das Konfidenzintervall trotzdem nicht signifikant kleiner (z.B.: $Ch_j^\beta = 10$ und $\frac{k\sigma}{\sqrt{N}} = 10^{-3}$). (Um zu kennzeichnen, zu welchem Verfahren das jeweilige C und N gehört, wird ab hier C_β und N_β verwendet)
$\Rightarrow \frac{k\sigma}{\sqrt{N_\beta}} = Konst * C_\beta h_j^\beta$

$$\Rightarrow N_\beta = \underbrace{\left(\frac{k\sigma}{Konst \cdot C_\beta}\right)^2}_{\tilde{C}_\beta} \frac{1}{h_j^{2\beta}}$$

Hieraus kann man gut erkennen, dass bei steigender Ordnung des Verfahrens das optimale N exponentiell steigt.

Wenn man zum Beispiel annimmt, dass $h = \frac{1}{1000}$ klein genug ist für $\beta = 1$, $\beta = 2$ und $\beta = 4$, dann gilt:

$$\begin{aligned} N_1 &= \tilde{C}_1 \frac{1}{h^2}, \\ N_2 &= \tilde{C}_2 \frac{1}{h^4} = \frac{\tilde{C}_2}{\tilde{C}_1} N_1 \frac{1}{h^2} = \frac{\tilde{C}_2}{\tilde{C}_1} N_1 10^6, \\ N_4 &= \tilde{C}_4 \frac{1}{h^8} = \frac{\tilde{C}_4}{\tilde{C}_1} N_1 \frac{1}{h^6} = \frac{\tilde{C}_4}{\tilde{C}_1} N_1 10^{18} \end{aligned} \quad (5.4)$$

5.2 Numerische Simulation

In diesem Abschnitt wird die Implementierung der Algorithmen betrachtet mit dem Ziel, die Fehlerentwicklung bei der Berechnung des Erwartungswertes zu analysieren. Wir vergleichen das Verfahren erster Ordnung mit dem Verfahren zweiter Ordnung und wollen sehen, ob sich der Mehraufwand für das Verfahren zweiter Ordnung lohnt oder ob der Fehler der Monte-Carlo-Simulation zu stark dominiert (vergleiche dazu das Intervall aus Satz 5.1). Dazu werden wir sowohl die Fehler zur exakten Lösung betrachten (dem Erwartungswert von S), als auch die Rechenzeit.

Die Verfahren wurden alle in Matlab implementiert und auch alle Abbildungen wurden mit Matlab erstellt. Alle Laufzeitanalysen wurden mit Matlab R2011b erzeugt. Das dazu verwendete System ist Windows Vista 64 Bit mit Intel(R) Core(TM)2 Duo CPU P8400 @ 2.26Ghz und 4GB RAM.

Für die Berechnung von $S(T)$ wird das Euler-Maruyama-Verfahren aus Kapitel 3.3 (angewendet auf die geometrische Brownsche Bewegung) als Verfahren erster Ordnung verwendet. Als Verfahren zweiter Ordnung wird das Heun-Verfahren aus Kapitel 3.4 verwendet mit $a(t, S(t)) = \mu S(t)$ und $b(t, S(t)) = \sigma S(t)$. Die so berechneten Werte für $S(T)$ werden dann in die Monte-Carlo-Simulation zur Berechnung des Erwartungswertes eingesetzt.

Der Erwartungswert wird mit unterschiedlichen Schrittweiten h ($h = T/M$) für die Verfahren und mit unterschiedlichen N für die Monte-Carlo-Simulation berechnet. Diese so berechnete Approximation des Erwartungswertes wird mit $\mathbb{E}_{MCS}$ bezeichnet. Diese Berechnungen werden jeweils P-mal durchgeführt. Der prozentuale Fehler wird dann berechnet durch

$$err_{Verf} = \frac{\frac{1}{P}\sum_{i=1}^{P} \mathbb{E}_{MCS} - \mathbb{E}[S]}{\mathbb{E}[S]},$$

wobei der exakte Erwartungswert $\mathbb{E}[S]$ durch die Formel (2.5) gegeben ist:

$$\mathbb{E}[S] = \mathbb{E}[S(T, S_0)] = S_0 e^{\mu T}.$$

Dabei entspricht das P-malige Berechnen des Erwartungswertes mit N Pfaden einem einmaligen Berechnen des Erwartungswertes mit $P \cdot N$ Pfaden. Die Angaben für den Parameter N in den Grafiken und Tabellen beziehen sich immer auf das Modell mit P-maliger Simulationen des Erwartungswertes.

Der Wiener-Prozess wird nicht gesondert erzeugt, sondern direkt in die zwei Verfahren eingebaut. Der Vorteil dabei ist, dass er nicht gespeichert und mit übergeben werden muss. Damit man trotzdem immer die gleichen Zufallszahlen für beide Verfahren verwendet, gibt es in Matlab den Befehl randn('state',0). Obwohl die Ordnung für beide Verfahren für eine Verwendung der schwachen Approximation angegeben ist, wird eine starke Approximation für den Wiener-Prozess verwendet. Dies liegt an der Tatsache, dass das Verfahren zweiter Ordnung einen dreipunktverteilten Wiener-Prozess benötigt und das Euler-Maruyama-Verfahren nur einen zweipunktverteilten Wiener-Prozess. Damit würde aber die Vergleichbarkeit verloren gehen. Somit werden $\mathcal{N}(0, h)$-verteilte Zufallszahlen für $\Delta\tilde{W}$ verwendet.

Die festen Parameter sind $S_0 = 100$, $T = 1$, $P = 20$ und $\mu = 0,1$. Die Volatilität σ nimmt die Werte 0.01, 0.1 und 0.3 an. Die Werte für h (M) und N variieren. Die Zeit wird in Sekunden angegeben. Der Matlab-Code kann im Anhang eingesehen werden.

5.2.1 Variation der Schrittweite h

Als erstes betrachten wir was passiert, wenn wir die Schrittweite h verkleinern. Die ersten drei Testreihen wurden in Matlab mit randn('state',0) erzeugt.

1. Testreihe

Tabelle 5.1: $N = 100$, $\mu = 0.1$, $\sigma = 0.01$				
	t_{Euler}	err_{Euler}	t_{Heun}	err_{Heun}
$h = 10^{-2}$	0.04	$3.033172 \cdot 10^{-4}$	0.08	$3.538853 \cdot 10^{-4}$
$h = 10^{-3}$	0.22	$6.033344 \cdot 10^{-6}$	0.57	$1.098323 \cdot 10^{-5}$
$h = 10^{-4}$	2.00	$2.844494 \cdot 10^{-5}$	5.51	$2.793562 \cdot 10^{-5}$
$h = 10^{-5}$	19.74	$2.226374 \cdot 10^{-4}$	55.05	$2.225859 \cdot 10^{-4}$

2. Testreihe

Tabelle 5.2: $N = 100$, $\mu = 0.1$, $\sigma = 0.1$				
	t_{Euler}	err_{Euler}	t_{Heun}	err_{Heun}
$h = 10^{-2}$	0.02	$3.654814 \cdot 10^{-3}$	0.06	$3.732581 \cdot 10^{-3}$
$h = 10^{-3}$	0.21	$8.689660 \cdot 10^{-5}$	0.56	$8.731406 \cdot 10^{-5}$
$h = 10^{-4}$	1.97	$1.805147 \cdot 10^{-4}$	5.52	$1.790630 \cdot 10^{-4}$
$h = 10^{-5}$	19.72	$2.074612 \cdot 10^{-3}$	55.07	$2.074378 \cdot 10^{-3}$

3. Testreihe

Tabelle 5.2: $N = 100$, $\mu = 0.1$, $\sigma = 0.3$				
	t_{Euler}	err_{Euler}	t_{Heun}	err_{Heun}
$h = 10^{-2}$	0.03	$1.208921 \cdot 10^{-2}$	0.06	$1.234818 \cdot 10^{-2}$
$h = 10^{-3}$	0.20	$1.515900 \cdot 10^{-3}$	0.56	$1.564036 \cdot 10^{-3}$
$h = 10^{-4}$	1.99	$1.705273 \cdot 10^{-4}$	5.53	$1.796036 \cdot 10^{-4}$
$h = 10^{-5}$	19.87	$5.756512 \cdot 10^{-3}$	55.03	$5.754382 \cdot 10^{-3}$

Betrachten wir zuerst die Ergebnisse der Tabellen. Es fällt auf, dass das Verfahren zweiter Ordnung erst ab einer sehr kleinen Schrittweite minimal besser ist als das Verfahren erster Ordnung. Die Differenzen in den Fehlern sind marginal und sinken mit sinkender Schrittweite. Die Verfahren nähern sich somit aneinander an.

Da die Simulationen 20-mal durchgeführt werden, ist N effektiv gleich $20 \cdot 100 = 2000$.

Die Grafiken unterstützen diese Aussage. Es ist durchgehend nur der Fehler des Heun-Verfahrens zu sehen, da er den Fehler des Euler-Verfahrens überdeckt (Abb. 5.1 und 5.2). Bei Abbildung 5.1 ist $N = 100$ und somit haben wir effektiv $N = 2000$, was sehr wenig ist. In Abbildung 5.2 ist $N = 1000$

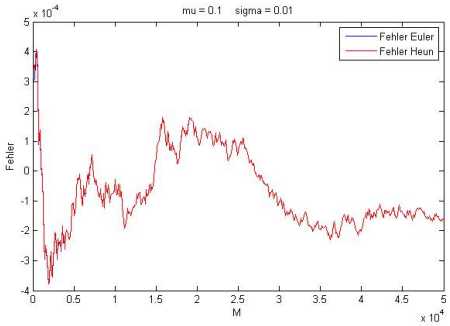

Abbildung 5.1: Variation von M, $\sigma = 0.01$, $N = 100$, randn('state',0)

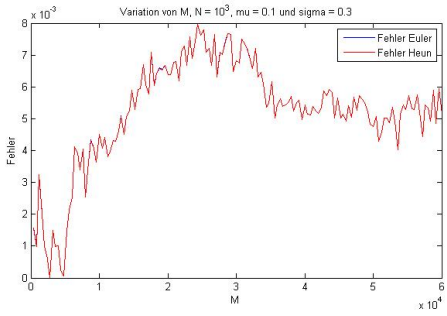

Abbildung 5.2: Variation von M, $\sigma = 0.3$, $N = 10^3$, randn('state',2^8)

und somit effektiv bei 20.000, doch auch hier erhalten wir die gleichen Ergebnisse. Dies legt nahe, dass der Fehler der Monte-Carlo-Simulation sehr groß ist im Vergleich zu den Verfahrensfehlern und diese dominiert.

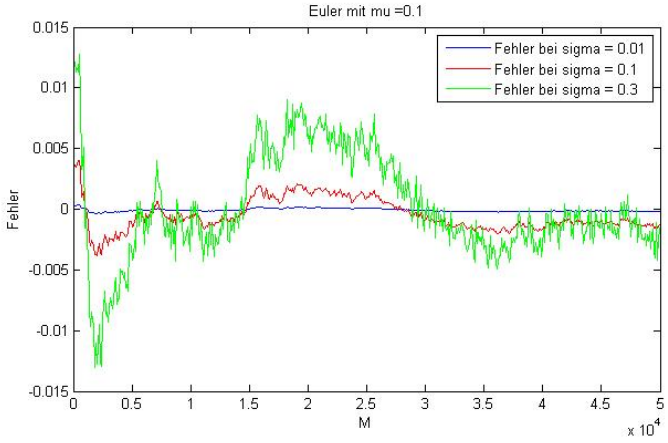

Abbildung 5.3: Variation von M, Vergleich verschiedener σ, randn('state',0)

Mit steigendem σ (σ ist ein Parameter der geometrische Brownsche Bewegung: $dS(t) = \mu S(t)dt + \sigma S(t)dW_t$) steigt sowohl die Volatilität des Fehlers als auch der Fehler selbst (Abbildung 5.3).

Die Formel (2.6) gibt uns die Varianz der exakten Lösung der geometrischen Brownschen Bewegung:

$$Var[S(T)] = S_0^2 e^{2\mu T}(e^{\sigma^2 T} - 1).$$

Bei kleinen Schrittweiten h sind die Varianz der exakten Lösung und die Varianz der Approximation ungefähr gleich. Somit beeinflusst der Parameter σ die Varianz unserer Approximation. Diese Varianz geht nun wieder in die

Länge des Konfidenzintervalls ein (siehe Satz 5.1):

$$\frac{1}{N}\sum_{i=1}^{N}\tilde{S}_i(T) \in \left[\mathbb{E}[S(T)] - \frac{k\cdot\sqrt{Var[\tilde{S}_i(T)]}}{\sqrt{N}} - Ch_j^{\beta}, \mathbb{E}[S(T)] + \frac{k\cdot\sqrt{Var[\tilde{S}_i(T)]}}{\sqrt{N}} + Ch_j^{\beta}\right]$$

Mit wachsendem σ vergrößert sich also das Konfidenzintervall und die Varianz des approximierten Kurses. Der approximierte Erwartungswert schwankt also in einem größeren Intervall.

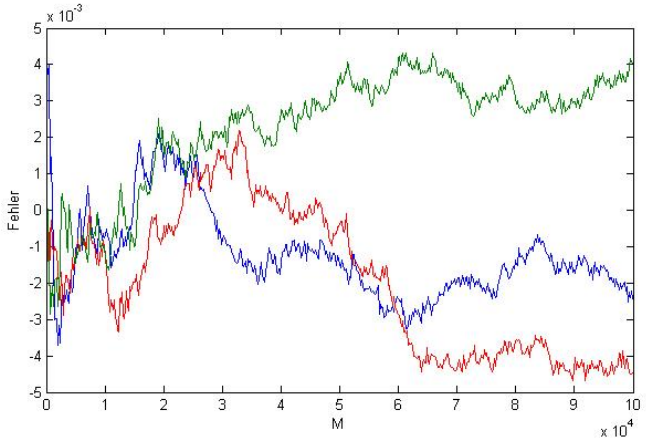

Abbildung 5.4: Euler-Verfahren mit $\mu = 0.1$ und $\sigma = 0.1$, 3 verschiedene Startwerte für den Zufallszahlengenerator $(0, 2^8, 2^{32})$

Startet man den Zufallszahlengenerator in Matlab mit verschiedenen Zahlen, so erhält man unterschiedliche Fehlerverteilungen (Abbildung 5.4). Die Verfahren reagieren sehr sensitiv auf die Zufallszahlen. Bei vielen Startwerten wird das Verfahren mit sinkender Schrittweite sogar immer schlechter.

Die Rechenzeit steigt bei beiden Verfahren linear. Das Euler-Maruyama-Verfahren benötigt ungefähr 36% der Zeit des Heun-Verfahrens. Die Fehler der Verfahren sind sich jedoch sehr ähnlich, sodass sich der Aufwand für das Verfahren zweiter Ordnung nicht rentiert. Hier sollte stets das Euler-Verfahren verwendet werden und die Schrittweite relativ groß gewählt werden.

5.2.2 Variation von N

Als nächstes wollen wir untersuchen was passiert, wenn wir die Schrittweite fest wählen und die Anzahl der Realisierungen (N) vergrößern.

4. Testreihe

Tabelle 1.2: $h = 10^{-2}$, $\mu = 0.1$, $\sigma = 0.01$				
	t_{Euler}	err_{Euler}	t_{Heun}	err_{Heun}
$N = 10^2$	0.02	$3.033172 \cdot 10^{-4}$	0.06	$3.538853 \cdot 10^{-4}$
$N = 10^3$	0.22	$4.506992 \cdot 10^{-5}$	0.57	$4.833985 \cdot 10^{-6}$
$N = 10^4$	2.07	$5.272936 \cdot 10^{-5}$	5.59	$2.773707 \cdot 10^{-6}$
$N = 10^5$	20.67	$5.702576 \cdot 10^{-5}$	59.13	$7.082698 \cdot 10^{-6}$

5. Testreihe

Tabelle 2.2: $h = 10^{-2}$, $\mu = 0.1$, $\sigma = 0.1$				
	t_{Euler}	err_{Euler}	t_{Heun}	err_{Heun}
$N = 10^2$	0.03	$3.654814 \cdot 10^{-3}$	0.06	$3.732581 \cdot 10^{-3}$
$N = 10^3$	0.21	$5.999193 \cdot 10^{-5}$	0.57	$1.044761 \cdot 10^{-4}$
$N = 10^4$	2.08	$6.718468 \cdot 10^{-5}$	5.61	$1.626060 \cdot 10^{-5}$
$N = 10^5$	20.73	$1.174676 \cdot 10^{-4}$	56.12	$6.745030 \cdot 10^{-5}$

6. Testreihe

Tabelle 2.2: $h = 10^{-2}$, $\mu = 0.1$, $\sigma = 0.3$				
	t_{Euler}	err_{Euler}	t_{Heun}	err_{Heun}
$N = 10^2$	0.03	$1.208921 \cdot 10^{-2}$	0.07	$1.234818 \cdot 10^{-2}$
$N = 10^3$	0.22	$6.128122 \cdot 10^{-4}$	0.57	$6.031953 \cdot 10^{-4}$
$N = 10^4$	2.07	$6.385794 \cdot 10^{-5}$	5.61	$4.557452 \cdot 10^{-6}$
$N = 10^5$	20.65	$2.294244 \cdot 10^{-4}$	56.08	$1.791492 \cdot 10^{-4}$

Wie verändert sich nun der prozentuale Fehler, wenn wir unterschiedlich viele Lösungen der geometrischen Brownschen Bewegung für die Monte-Carlo-Simulation verwenden? Auch hier können wir beobachten, dass wie bei der Variation der Schrittweite, bei kleinen N das Euler-Verfahren minimal genauer ist als das Heun-Verfahren. Im Gegensatz zur Variation der Schrittweite kann man hier jedoch bei größeren N deutliche Unterschiede zwischen den Verfahren erkennen.

Betrachtet man hier die Grafiken, so sieht man, dass bei kleinem σ das Heun-Verfahren besser ist als das Euler-Verfahren (Abbildung 5.5). Auf Abbildung 5.6 sieht man, dass das Heun-Verfahren zeitweise schlechter ist als das Euler-Verfahren ($N = 2 \cdot 10^4$ bis $N = 3 \cdot 10^4$). Bei Abbildung 5.7 ist das Euler-Verfahren sogar dauerhaft besser als das Heun-Verfahren.

Startet man den Zufallszahlengenerator an verschiedenen Stellen, so bekommt man andere Fehler für die gleichen N. Dies sehen wir in den Abbildungen 5.6, 5.7 und 5.8. Es schwankt nicht nur der Fehler, sondern abhängig von den Startwerten des Zufallszahlengenerators auch, welches Verfahren das bessere ist (vor allem bei großen σ). Bei kleinen σ (z.B. $\sigma = 0.01$) ist das Heun-Verfahren eindeutig besser. Bei größeren σ kann man kein besseres

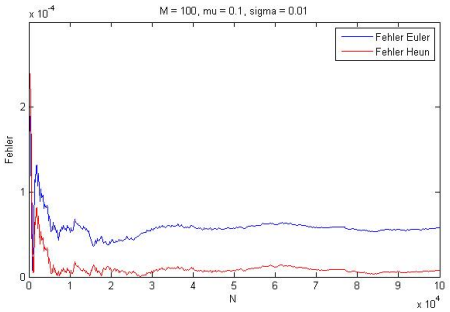

Abbildung 5.5: Variation von N, $\sigma = 0.01$, randn('state',0)

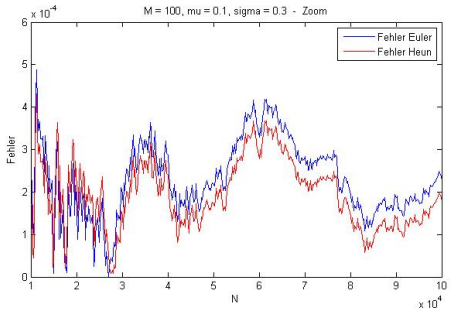

Abbildung 5.6: Variation von N, $\sigma = 0.3$, randn('state',0), Zoom

Verfahren bestimmen. Da die Fehler relativ zum Gesamtfehler gesehen immer näher beieinander liegen und mal das eine, mal das andere Verfahren besser ist, sollte man auf Grund des geringeren Rechenzeitaufwandes das Euler-Verfahren verwenden ($\sigma = 0.1$ und $\sigma = 0.3$).

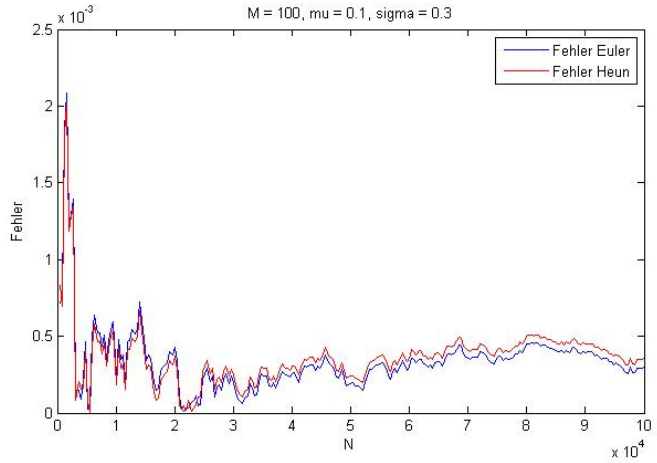

Abbildung 5.7: Variation von N, $\sigma = 0.3$, randn('state',2^{32})

Betrachtet man die Abbildungen 5.5 bis 5.8, so erkennt man, dass ab einem bestimmten N die Genauigkeit nicht mehr signifikant steigt. In Abbildung 5.5 ist dies ab $N = 0.6 \cdot 10^4 \cdot 20 = 1.2 \cdot 10^5$ der Fall, bei Abbildung 5.8 ändert sich ab $N = 1.7 \cdot 10^4 \cdot 20 = 3.4 \cdot 10^5$ nicht mehr viel und bei $\sigma = 0.3$ schwankt der Fehler auch ab $N = 0.6 \cdot 10^4 \cdot 20 = 1.2 \cdot 10^5$ immer in dem gleichen Intervall. Es ist im allgemeinen $N = 1 \cdot 10^4 \cdot 20 = 2 \cdot 10^5$ ausreichend. Ab diesem Wert schwankt der Fehler immer im gleichen Intervall.

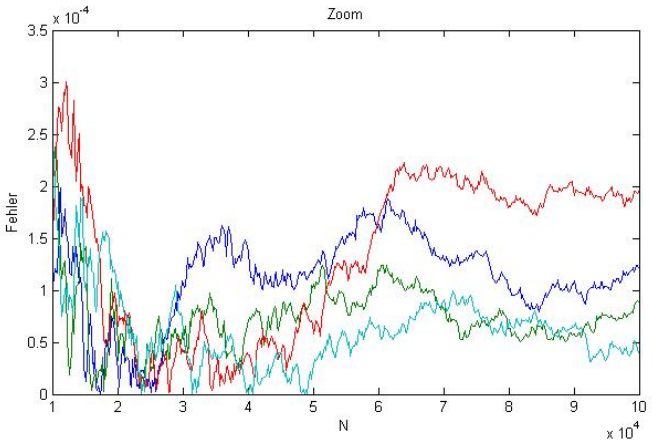

Abbildung 5.8: Euler-Verfahren mit $\mu = 0.1$ und $\sigma = 0.1$, 4 verschiedene Startwerte für den Zufallszahlengenerator $(0, 2^8, 3 \cdot 2^{22}, 2^{32})$

Da bei wenigen Startwerten für den Zufallszahlengenerator der Fehler nach diesem N noch etwas geringer wird, kann man bis zu $N = 3 \cdot 10^4 \cdot 20 = 6 \cdot 10^5$ gehen, danach rentiert sich der zusätzliche Rechenaufwand nicht mehr.

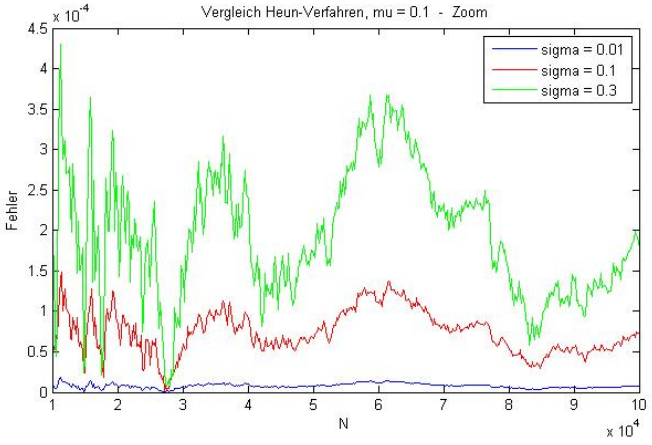

Abbildung 5.9: Vergleich Heun-Verfahren, verschiedene σ, randn('state',0)

Wie bei der Variation der Schrittweite steigt auch hier mit steigendem σ sowohl die Volatilität des Fehlers als auch der Fehler selbst. Die Begründung ist hier die gleiche wie im vorherigen Abschnitt. Dies gilt sowohl für das Heun-Verfahren (Abbildung 5.9) als auch für das Euler-Verfahren.

Die Rechenzeit steigt bei beiden Verfahren linear, das Euler-Verfahren benötigt ungefähr 37% der Rechenzeit des Heun-Verfahrens (Abbildung 5.10).

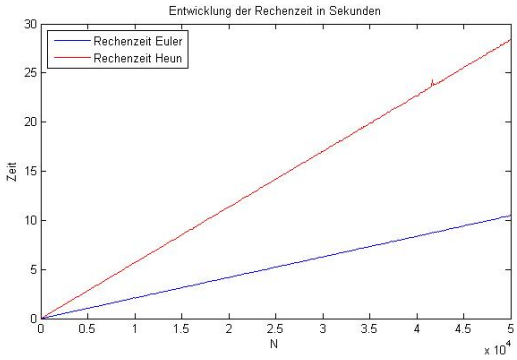

Abbildung 5.10: Rechenzeit der beiden Verfahren

Um beurteilen zu können, welches Verfahren besser ist, müssen wir für beide Verfahren die gleiche Rechenzeit erhalten. Dies erreichen wir zum einen, indem wir für das Euler-Verfahren N verdreifachen. Dann benötigt das Euler-Verfahren 110.5% der Rechenzeit des Heun-Verfahrens. Für kleine σ (0.01) ist das Heun-Verfahren weiterhin besser (Abb. 5.11). Für größere σ (ab 0.1) ist das Euler-Verfahren zu bevorzugen. Zum einen wird der Fehler für kleine N schneller klein, zum anderen ist bei großen N wieder nicht eindeutig, welches Verfahren besser ist. Allerdings ist das Euler-Verfahren öfters besser als das Heun-Verfahren (Abb. 5.12 und 5.13). Hier sollte für das Euler-Verfahren $N = 3 \cdot 1 \cdot 10^4 \cdot 20 = 6 \cdot 10^5$ gewählt werden.

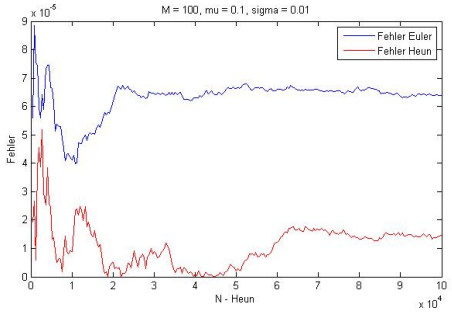

Abbildung 5.11: $N_{Euler} = 3 \cdot N_{Heun}$, $\sigma = 0.01$, randn('state',2^8)

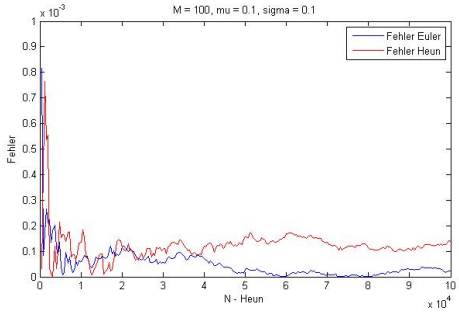

Abbildung 5.12: $N_{Euler} = 3 \cdot N_{Heun}$, $\sigma = 0.1$, randn('state',2^{32})

Wenn man nun für das Heun-Verfahren weiterhin $M = 100$ verwendet und für das Euler-Verfahren $M = 300$ ($h = \frac{T}{M}$), so benötigt das Euler-Verfahren 106% der Rechenzeit des Heun-Verfahrens. Für kleine σ ist das Heun-Verfahren immer noch besser (Abb. 5.14). Allerdings ist das Euler-Verfahren hier besser als bei einer Verdreifachung von N. Für größere σ ist das Heun-Verfahren im relevanten Bereich von N besser (Abb. 5.15). Dies bedeutet, dass die Verdreifachung von N für das Euler-Verfahren besser ist,

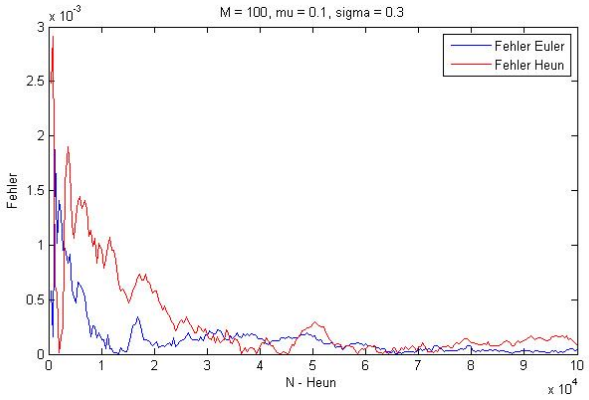

Abbildung 5.13: $N_{Euler} = 3 \cdot N_{Heun}$, $\sigma = 0.3$, randn('state',2^{16})

da dort bei größeren σ das Euler-Verfahren besser war und wir das Heun-Verfahren nicht verändert haben. Also sollte man für größere σ weiterhin das Euler-Verfahren mit großem N wählen anstatt das Heun-Verfahren.

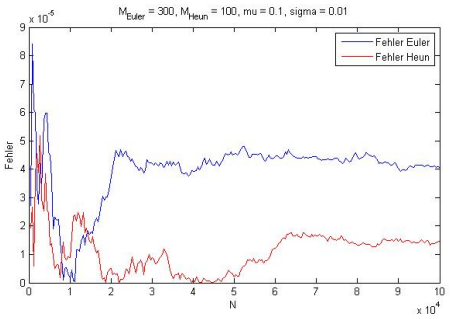

Abbildung 5.14: $M_{Euler} = 3 \cdot M_{Heun}$, $\sigma = 0.01$, randn('state',2^8)

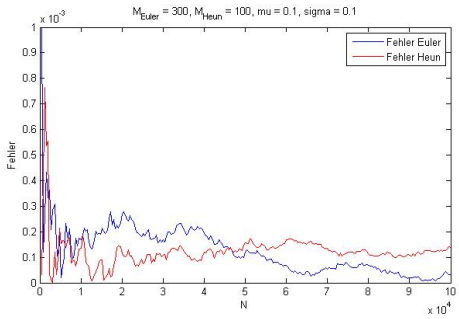

Abbildung 5.15: $M_{Euler} = 3 \cdot M_{Heun}$, $\sigma = 0.1$, randn('state',2^{32})

Vergleichen wir nun die Variation von N mit verschiedenen Schrittweiten h ($h = T/M$), so ergibt eine kleinere Schrittweite nicht unbedingt einen kleineren Fehler. In Abbildung 5.16 gilt sogar: je größer die Schrittweite desto genauer das Ergebnis. Für das Heun-Verfahren konvergieren alle Fehler gegen ungefähr den gleichen Wert. Diese Ergebnisse rechtfertigen auf keinen Fall eine Verdreifachung der Rechenzeit. Bei der Variation von N sollte somit eine relativ große Schrittweite gewählt werden.

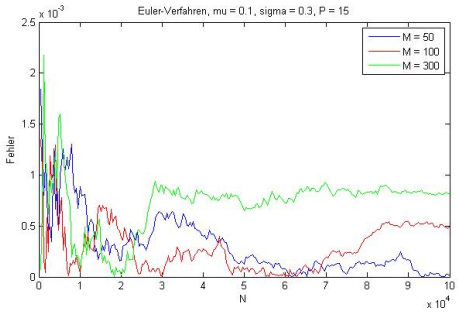

Abbildung 5.16: Euler-Verfahren mit verschiedenen Schrittweiten h, $\sigma = 0.3$, P=15, randn('state',2^8)

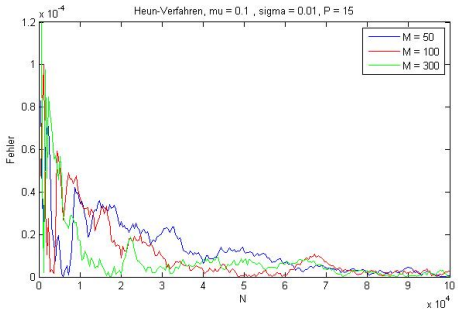

Abbildung 5.17: Heun-Verfahren mit verschiedenen Schrittweiten h, $\sigma = 0.01$, P=15, randn('state',2^{16})

5.2.3 Vergleich der Variation von h und von N

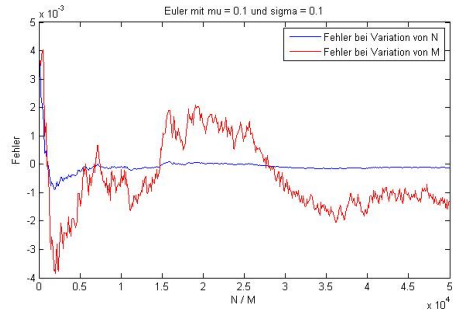

Abbildung 5.18: Vergleich mit Euler, $\sigma = 0.1$, randn('state',0)

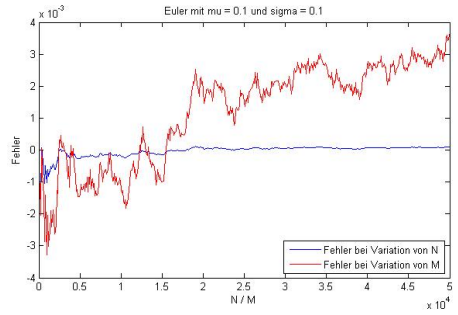

Abbildung 5.19: Vergleich mit Euler, $\sigma = 0.1$, randn('state',2^{32})

Vergleicht man die zwei Variationsmöglichkeiten, so sieht man, dass die Variation von N bei weitem überlegen ist (Abb. 5.18 und 5.19). Zum einen ist die Volatilität des Fehlers geringer, zum anderen ist auch der Fehler selbst geringer. Bei dem Start der Zufallszahlen mit randn('state',0) ist der prozentuale Fehler um einen Faktor von bis zu 100 kleiner. Bei randn('state',2^{32}) steigt der Fehler bei einer Variation der Schrittweite sogar immer stärker an.

Vergleicht man das Ergebnis von $M = 100$ mit dem Ergebnis von $M = 300$, so verdreifacht sich zwar die Rechenzeit, das Ergebnis wird aber nicht unbedingt besser (Abb. 5.20). Teilweise erreicht man mit der gröberen Schrittweite sogar ein genaueres Ergebnis (vergleiche auch vorheriger Abschnitt, Abb. 5.16 und 5.17). Man sollte also eine relativ grobe Schrittweite wählen und stattdessen N vergrößern.

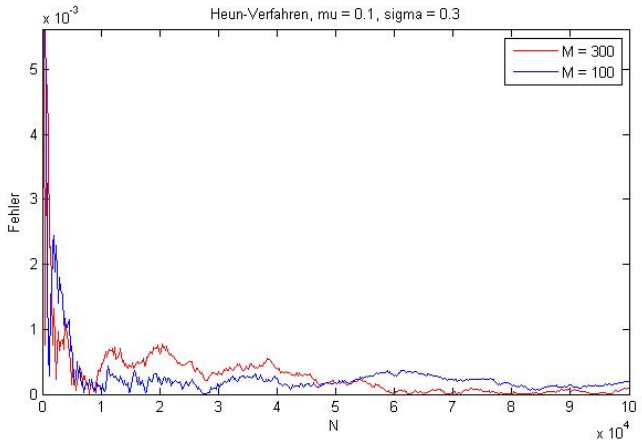

Abbildung 5.20: Heun-Verfahren mit verschiedenen Schrittweiten h, $\sigma = 0.3$, randn('state',0)

Da wir im vorherigen Abschnitt gesehen haben, dass ab einem gewissen N der Fehler nicht mehr gravierend kleiner wird und somit nur die Rechenzeit steigt, sollte man N nicht beliebig steigern. In den durchgeführten Analysen war ein N von $1 \cdot 10^4 \cdot 20 = 2 \cdot 10^5$ meistens ausreichend. Allerdings kann man aufgrund der anhaltenden Schwankungen im Fehler vor allem bei großen σ bei jedem beliebigen N einen relativ großen Fehler erhalten.

Bei einem relativ kleinen N und einer immer kleiner werdenden Schrittweite schwankt der Fehler in einem sehr großen Intervall. Bei einer relativ groben Schrittweite und einem immer größer werdenden N wird der Fehler geringer. Vergleicht man dieses Ergebnis mit dem Intervall aus (5.3), so erkennt man, dass der Fehler der Monte-Carlo-Simulation sehr groß ist im Vergleich zu den Verfahrensfehlern.

5.2.4 Varianzreduktion

In diesem Abschnitt betrachten wir, welche Auswirkungen eine Varianzreduktion mit antithetischen Zufallszahlen nach Kapitel 4.3 hat. In Abschnitt 5.2.1 haben wir nochmals gesehen, welchen Einfluss die Varianz des Kurses auf die Länge des Konfidenzintervalls hat. Für die Varianzreduktion halbieren wir N, damit die Vergleichbarkeit mit den vorherigen Abschnitten erhalten bleibt (vergleiche Algorithmus 4.4).

In den Abbildungen 5.21 bis 5.24 wird der Fehler bei einem Verfahren ohne Varianzreduktion mit dem Fehler des gleichen Verfahrens mit Varianzreduktion verglichen. Es ist deutlich zu erkennen, dass die Ausschläge nach oben und nach unten bis zu einem Faktor von 10 geringer werden. Durch die antithetischen Zufallszahlen stabilisieren sich die Fehler. Beide Verfahren werden

meistens genauer, jedoch ist in manchen Fällen das Verfahren mit Varianzreduktion kurzzeitig schlechter. Dies liegt an den großen Schwankungen der Verfahren ohne antithetische Zufallsvariablen.

Das Euler-Verfahren mit antithetischen Zufallszahlen braucht ungefähr 68% der Rechenzeit des Euler-Verfahrens ohne Varianzreduktion, das Heun-Verfahren braucht 92%. Das heißt, mit einem geringeren Zeitaufwand kann ein besseres Ergebnis erzielt werden. Es sollte also immer eine Varianzreduktion verwendet werden.

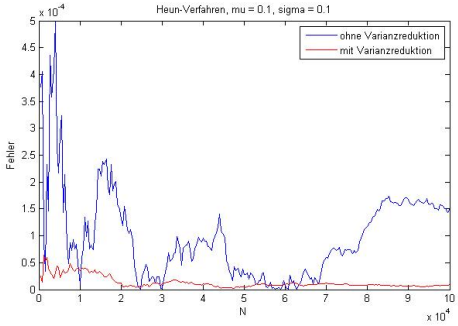

Abbildung 5.21: Vergleich des Heun-Verfahrens mit und ohne Varianzreduktion, $\sigma = 0.1$, randn('state',2^8)

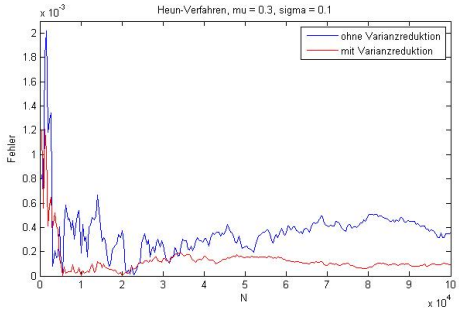

Abbildung 5.22: Vergleich des Heun-Verfahrens mit und ohne Varianzreduktion, $\sigma = 0.3$, randn('state',2^{32})

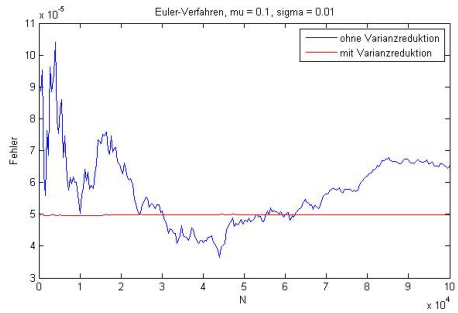

Abbildung 5.23: Vergleich des Euler-Verfahrens mit und ohne Varianzreduktion, $\sigma = 0.01$, randn('state',2^8)

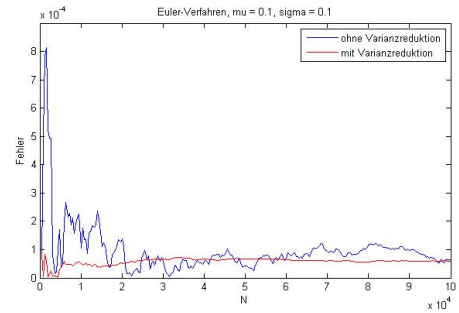

Abbildung 5.24: Vergleich des Euler-Verfahrens mit und ohne Varianzreduktion, $\sigma = 0.1$, randn('state',2^{32})

Die Abbildung 5.25 zeigt das Euler- und das Heun-Verfahren mit Varianzreduktion bei einem σ von 0.01. Unabhängig von den Startzahlen für den Zufallszahlengenerator in Matlab sind die Fehlerverteilungen identisch, der Fehler des Euler-Verfahrens liegt immer bei $5 \cdot 10^{-5}$. Hier ist der stabilisierende Effekt der antithetischen Zufallszahlen am besten zu sehen. Bei kleinen σ sollte ausnahmslos das Heun-Verfahren verwendet werden. Außerdem ist $N = 0.2 \cdot 10^4 \cdot 20 = 4 \cdot 10^4$ auf Grund der hohen Stabilität ausreichend.

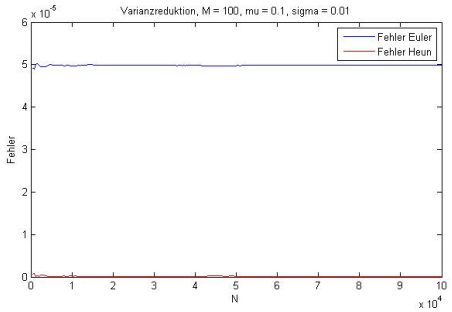

Abbildung 5.25: Varianzreduktion, Vergleich Euler- und Heun-Verfahren, $\sigma = 0.01$, randn('state',2^{16})

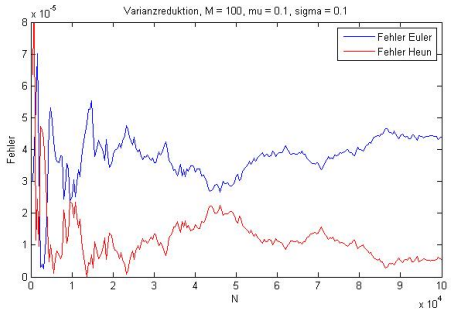

Abbildung 5.26: Varianzreduktion, Vergleich Euler- und Heun-Verfahren, $\sigma = 0.1$, randn('state',2^{16})

Abbildung 5.26 zeigt den Vergleich zwischen dem Verfahren erster Ordnung und dem Verfahren zweiter Ordnung bei $\sigma = 0.1$. Am Anfang ist, unabhängig vom Startwert, das Euler-Verfahren kurzzeitig besser. Spätestens ab $N = 2 \cdot 10^4 \cdot 20 = 4 \cdot 10^5$, meistens aber ab $N = 0.5 \cdot 10^4 \cdot 20 = 1 \cdot 10^5$ ist das Heun-Verfahren besser, auch wenn man für das Euler-Verfahren N verdreifacht. Im Gegensatz zu den Verfahren ohne Varianzreduktion, bei denen bei $\sigma = 0.1$ bereits das Euler-Verfahren verwendet werden sollte, ist hier immer noch das Heun-Verfahren überlegen. Meistens ist $N = 0.6 \cdot 10^4 \cdot 20 = 1.2 \cdot 10^5$ ausreichend, aber in seltenen Fällen ist $N = 2 \cdot 10^4 \cdot 20 = 4 \cdot 10^5$ für das Heun-Verfahren nötig.

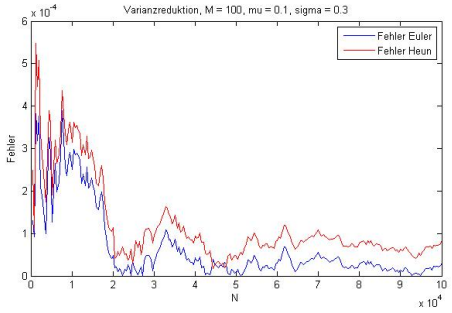

Abbildung 5.27: Varianzreduktion, Vergleich Euler- und Heun-Verfahren, $\sigma = 0.3$, randn('state',2^8)

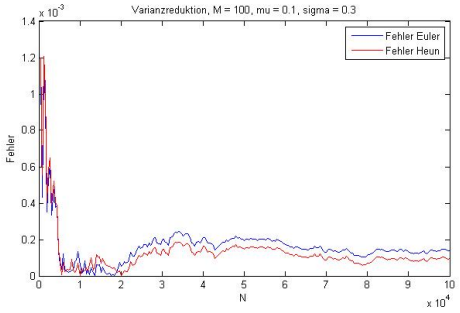

Abbildung 5.28: Varianzreduktion, Vergleich Euler- und Heun-Verfahren, $\sigma = 0.3$, randn('state',2^{32})

Bei $\sigma = 0.3$ ist in den meisten Fällen das Euler-Verfahren besser, nur bei wenigen Startwerten schwanken die Fehler umeinander. Ein Sonderfall ergibt sich bei randn('state',2^{32}), denn hier ist das Heun-Verfahren besser. Ohne antithetische Zufallszahlen war das bei diesem Startwert genau umge-

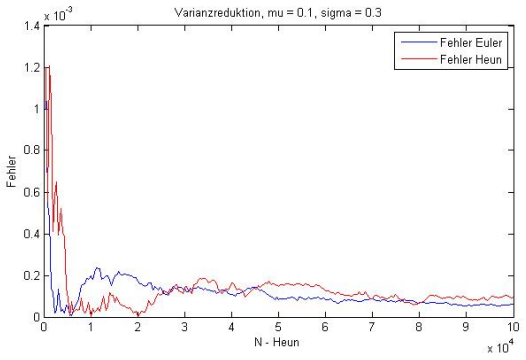

Abbildung 5.29: Varianzreduktion mit $N_{Euler} = 3 \cdot N_{Heun}$, Vergleich Euler- und Heun-Verfahren, $\sigma = 0.3$, randn('state',2^{32})

kehrt. Gleicht man die Rechenzeit des Euler-Verfahrens an die Rechenzeit des Heun-Verfahrens an, indem man $N_{Euler} = 3 \cdot N_{Heun}$ wählt (Abb. 5.29), so ist das Euler-Verfahren immer gleich gut oder besser. Insbesondere auch bei randn('state',2^{32}). Es sollte also bei hohen σ das Euler-Verfahren gewählt werden, mit N zwischen $3 \cdot 0.5 \cdot 10^4 \cdot 20 = 3 \cdot 10^5$ und $3 \cdot 3 \cdot 10^4 \cdot 20 = 1.8 \cdot 10^6$.

Da die Verfahrensfehler ab einem bestimmten N relativ stabil werden, kann man das optimale N leicht bestimmen, auch wenn man die analytische Lösung nicht kennt. Man berechnet den Erwartungswert mit unterschiedlichen N und sobald der Erwartungswert immer im gleichen Intervall schwankt, kann man abbrechen.

5.2.5 Fazit

Die zentrale Frage dieser Arbeit ist, ob sich Verfahren höherer Ordnung in der Monte-Carlo-Simulation rentieren. Dies wurde theoretisch und numerisch untersucht.

Aus der theoretischen Analyse bekamen wir ein Konfidenzintervall für unsere Lösung. Auf die Länge des Intervalls konnte man durch die Schrittweite h, das Verfahren und die Anzahl an Lösungen der stochastischen Differentialgleichung N für die Monte-Carlo-Simulation Einfluss nehmen. Die Analyse ergab, dass es bei festem Verfahren und fester Schrittweite ein optimales N gibt.

$$\left[\mathbb{E}[S(T)] - \frac{k \cdot \sqrt{Var[\tilde{S}_i(T)]}}{\sqrt{N}} - Ch_j^{\beta}, \mathbb{E}[S(T)] + \frac{k \cdot \sqrt{Var[\tilde{S}_i(T)]}}{\sqrt{N}} + Ch_j^{\beta} \right]$$

In der numerischen Analyse haben wir zuerst untersucht was passiert, wenn man die Schrittweite verkleinert. Das Ergebnis war eindeutig: Die beiden Verfahren hatten fast den gleichen Fehler und der Fehler schwankte enorm.

Somit sollte man die Schrittweite groß lassen, der Mehraufwand für kleinere Schrittweiten lohnt sich nicht.

Vergrößert man N, so ist das Heun-Verfahren für kleine σ besser. Bei größeren σ (bereits ab $\sigma = 0.1$) sollte man das Euler-Verfahren wählen. Insbesondere wenn man die Rechenzeit der beiden Verfahren angleicht, indem man $N_{Euler} = 3 \cdot N_{Heun}$ wählt, ist das Euler-Verfahren gleich gut oder besser. Bei dem Heun-Verfahren reicht meistens $N = 2 \cdot 10^5$ aus, für das Euler-Verfahren sollte man $N = 6 \cdot 10^5$ verwenden.

Gleicht man die Rechenzeit der beiden Verfahren an, indem man bei für das Euler-Verfahren dreimal so viele Gitterpunkte verwendet, so ist das Heun-Verfahren immer gleich gut oder besser. Allerdings ist das Ergebnis schlechter als bei der Verdreifachung von N und sollte deshalb nicht verwendet werden.

Mit einer Verdreifachung der Anzahl der Gitterpunkten ist keine deutliche Verbesserung des Ergebnisses bei der Variation von N zu erreichen. Somit sollte die Schrittweite - unabhängig vom Verfahren - sehr grob gewählt werde.

In allen Testreihen schwankt der Fehler ab einem bestimmten Wert für N immer im gleichen Intervall. Dieses Ergebnis stimmt mit der theoretischen Analyse überein, in der ein optimales N unterstellt wurde.

Vergleicht man die Variation der Schrittweite h mit der Variation von N, so ist die Variation von N bei gleicher Rechenzeit um den Faktor 100 besser. Ein weiterer Grund, um die Schrittweite groß zu lassen und stattdessen N zu vergrößern. Dies deutet darauf hin, dass der Monte-Carlo-Fehler im Vergleich zu den Verfahrensfehlern sehr groß ist.

Insgesamt hängen die Fehler vor allem bei großen σ und relativ kleinen N stark von den gewählten Zufallszahlen ab.

Die Volatilität σ beeinflusst die Varianz und somit auch die Länge des Konfidenzintervalls. Mit steigendem σ steigt die Volatilität und die Größe des Fehlers. Mit einer Varianzreduktion für die Varianz des Kurses erreichten wir eine Genauigkeitssteigerung bis zu dem Faktor 10. Die Verfahren werden nicht nur genauer, sonder auch die Schwankungen der Fehler werden geringer. Außerdem haben die Verfahren mit antithetischen Zufallszahlen eine geringere Rechenzeit. Deshalb sollte - unabhängig von der Wahl des Verfahrens - immer eine Varianzreduktion verwendet werden.

Bei einer Verwendung von antithetischen Zufallszahlen sollte bis zu $\sigma = 0.1$ das Heun-Verfahren verwendet werden. Erst ab sehr hohen σ ist das Euler-Verfahren besser als das Heun-Verfahren. Bei einer Varianzreduktion rentiert sich das Heun-Verfahren also länger als wenn wir keine Varianzreduktion verwenden. Auch werden die benötigten N kleiner. Aufgrund der Stabilität

ist bei $\sigma = 0.01$ nur ein N von $4 \cdot 10^4$ nötig, bei $\sigma = 0.1$ reicht $N = 1.2 \cdot 10^5$. Für das Euler-Verfahren bei $\sigma = 0.3$ braucht man öfters ein N größer als $3 \cdot 10^5$, was jedoch immer noch einen geringeren Rechenaufwand als ohne Varianzreduktion bedeutet.

Kapitel 6

Zusammenfassung

Ein Gebiet der Finanzmathematik ist die Bestimmung von Optionswerten. Dabei wird der Kursverlauf des Basiswertes mit einer stochastischen Differentialgleichung modelliert. Den Wert dieses Kurses zum Zeitpunkt T verwendet man dann, um den Erwartungswert des Optionswertes zum Zeitpunkt T zu berechnen. Wenn man keine analytische Formel für den zu berechnenden Erwartungswert hat, so kann man ihn mit einer Monte-Carlo-Simulation approximieren.

In dieser Arbeit haben wir die geometrische Brownsche Bewegung verwendet, um den Kursverlauf des Basiswertes zu modellieren. Anschließend haben wir verschiedene numerische Lösungsverfahren verwendet, um diese stochastische Differentialgleichung zu lösen. Als Verfahren erster Ordnung haben wir das Euler-Maruyama-Verfahren definiert, als Verfahren zweiter Ordnung das Heun-Verfahren. Anstatt den Erwartungswert des Optionswertes in T zu berechnen wurde in dieser Arbeit mit Hilfe einer Monte-Carlo-Simulation der Erwartungswert des Kurses zum Zeitpunkt T berechnet.

Die zu beantwortende Frage war, ob sich Verfahren höherer Ordnung zum Lösen stochastischer Differentialgleichungen rentieren, wenn man diese Lösungen in eine Monte-Carlo-Simulation zur Berechnung des Erwartungswertes einsetzt.

Die numerischen Versuche ergaben, dass bei größeren N und konstanter Schrittweite bei kleinen σ (σ = Volatilität, Parameter der geometrischen Brownschen Bewegung) das Heun-Verfahren deutlich überlegen ist. Wird der Parameter σ größer, so steigt die Volatilität des Fehlers. Das Euler-Verfahren ist im relevanten Bereich für N meist besser als das Heun-Verfahren, deshalb sollte bei höheren σ das Euler-Verfahren verwendet werden.

Bei einer Varianzreduktion durch antithetische Zufallszahlen sinkt der Fehler bis zu dem Faktor 10 und wird wesentlich stabiler. Außerdem ist der Zeitaufwand geringer. Somit sollte man - unabhängig vom Verfahren - immer eine Varianzreduktion verwenden. Das Heun-Verfahren ist bei antithetischen

Zufallszahlen viel länger besser als das Euler-Verfahren. Erst ab einem sehr großen σ ist das Euler-Verfahren besser.

Der Versuch, die Genauigkeit des Ergebnisses bei festem N durch eine kleinere Schrittweite für die Berechnung der $S(T)$ zu verbessern, schlägt leider fehl. Der prozentuale Fehler bei einer Vergrößerung von N für die Monte-Carlo-Simulation ist um den Faktor 100 geringer als der Fehler bei einer Verkleinerung der Schrittweite - bei gleicher Rechenzeit.

Leider kann man den Erwartungswert nicht beliebig genau berechnen. Ab einem N von ungefähr $2 \cdot 10^5$ für das Heun-Verfahren und $6 \cdot 10^5$ für das Euler-Verfahren ist keine gravierende Verbesserung zu erreichen.

Zusammenfassend lässt sich also sagen, dass man bei kleinen σ auf jeden Fall das Heun-Verfahren verwenden sollte. Sobald σ größer wird, ist das Euler-Verfahren zu bevorzugen. Es sollte stets eine Varianzreduktion verwendet werden. Die Schrittweite h sollte groß gewählt werden und N sollte nicht zu groß gewählt werden. Um das optimale N herauszufinden, sollte man längere Proberechnungen starten und das N wählen, ab dem sich der Fehler nicht mehr stark verändert. Alles in allem ist der prozentuale Fehler dennoch sehr gering, sodass die Monte-Carlo-Simulation eine gute Methode ist, um einen Erwartungswert zu approximieren.

In einer weiterführenden Arbeit wäre es noch interessant, die Konstanten aus der Fehlerabschätzung aus Abschnitt 5.1 zu berechnen. Außerdem könnte man Verfahren mit einer höherer Ordnung als zwei testen.

Anhang A

Matlab-Code

[1] main

[2] Euler-Verfahren

[3] Heun-Verfahren

[4] Monte-Carlo-Simulation

[5] Code zum Erzeugen der Bilder

[6] Euler-Verfahren mit Varianzreduktion

[7] Heun-Verfahren mit Varianzreduktion

[8] main mit Varianzreduktion

[9] main mit $N_{Euler} = 3 \cdot N_{Heun}$

[10] main mit $M_{Euler} = 300$ und $M_{Heun} = 100$

[1] Main

```
S0 = 100;
T = 1;
t0 = 0;
P = 20; % Anzahl an Monte-Carlo-Simulationen

mu = 0.01;
sigma = 0.01;
M = 100; % Anzahl der Gitterpunkte
N = 30000  % Anzahl der Pfade in der Monte-Carlo-Simulation

h = (T-t0)/M;

format long;

randn('state',0)
tic;
for i=1:P
      euler1 = euler(M,h,S0,N,sigma,mu);
      mcseuler(i) = mcs(N,euler1);
end
Zeiteuler = toc

randn('state', 0)
tic;
for i=1:P
      heun1 = heun(M,h,S0,N,sigma,mu);
      mcsheun(i) = mcs(N,heun1);
end
Zeitheun = toc

ST = S0*exp(mu*T);

sum1=0;
sum2=0;
for i=1:P
    sum1 = sum1 + mcseuler(i);
    sum2 = sum2 + mcsheun(i);
end

fehlermcseuler = abs(1/P*sum1 - ST)/ST
fehlermcsheun = abs(1/P*sum2 - ST)/ST
```

[2] Euler-Verfahren

```
% Verfahren schwacher 1. Ordnung
% Euler-Maruyama

function S = euler(M,h,S0,N,sigma,mu)

for j=1:N
    S(j)=S0;
    for i=1:M
        S(j)=S(j)+h*mu*S(j)+randn*sqrt(h)*sigma*S(j);
    end
end
```

[3] Heun-Verfahren

```
% ableitungsfreies Verfahren schwacher 2. Ordnung, analog Heun

function S = heun(M,h,S0,N,sigma,mu)

for j=1:N
    S(j) = S0;
    for i=1:M
        WP = randn*sqrt(h);  % Delta W
        Y1=S(j)+h*mu*S(j)+sigma*S(j)*WP;
        Y2=S(j)+h*mu*S(j)+sigma*S(j)*sqrt(h);
        Y3=S(j)+h*mu*S(j)-sigma*S(j)*sqrt(h);
        S(j)=S(j)+0.5*h*mu*(Y1+S(j))+0.25*sigma*(Y2+Y3+2*S(j))*WP
        +0.25/sqrt(h)*sigma*(Y2-Y3)*(WP^2-h);
    end
end
```

[4] Monte-Carlo-Simulation

```
% Approximation des Erwartungswertes

function V = mcs(N,S)

sum = 0;
for i=1:N
    sum = sum + S(i);
end
V=1/N*sum;
```

[5] Code zum Erzeugen der Bilder

```
% Variation von M, die Variation von N erfolgt analog
T = 1;
t0 = 0;
P = 20;
S0 = 100;

M = 200; % Anzahl der Gitterpunkte
N = 100;  % Anzahl der Pfade in der Monte-Carlo-Simulation
mu = 0.1;
sigma = 0.01;

for j=1:500

        h = (T-t0)/M;

        randn('state',2^32)
        tic;
        for i=1:P
            euler1 = euler(M,h,S0,N,sigma,mu);
            mcseuler(i) = mcs(N,euler1);
        end
        Zeiteuler(j) = toc;

        randn('state',2^32)
        tic;
        for i=1:P
            heun1 = heun(M,h,S0,N,sigma,mu);
            mcsheun(i) = mcs(N,heun1);
        end
        Zeitheun(j) = toc;

        ST = S0*exp(mu*T);
        sum1 = 0;
        sum2 = 0;
        for i=1:P
           sum1 = sum1 + mcseuler(i);
           sum2 = sum2 + mcsheun(i);
        end
        fehlermcseuler(j) = (1/P*sum1 - ST)/ST;
        fehlermcsheun(j) = (1/P*sum2 - ST)/ST;

        y(j) = M;
        M = M + 200;
end

figure(1)
plot(y,fehlermcseuler,'b',y,fehlermcsheun,'r')
```

```
figure(2)
plot(y,abs(fehlermcseuler),'b',y,abs(fehlermcsheun),'r')

figure(3)
plot(y,Zeiteuler,'b',y,Zeitheun,'r')
```

[6] Euler-Verfahren mit Varianzreduktion

```
% Verfahren schwacher 1. Ordnung mit Varianzreduktion
% Euler-Maruyama

function S = eulervr(M,h,S0,N,sigma,mu)

for j=1:2:N
    S(j)=S0;
    S(j+1)=S0;
    for i=1:M
        WP=randn*sqrt(h);
        S(j) = S(j)  +h*mu*S(j)  +WP*sigma*S(j);
        S(j+1)=S(j+1)+h*mu*S(j+1)-WP*sigma*S(j+1);
    end
end
```

[7] Heun-Verfahren mit Varianzreduktion

```
function S = heunvr(M,h,S0,N,sigma,mu)

for j=1:2:N
    S(j) = S0;
    S(j+1) = S0;
    for i=1:M
        WP = randn*sqrt(h);
        Y1=S(j)+h*mu*S(j)+sigma*S(j)*WP;
        Y2=S(j)+h*mu*S(j)+sigma*S(j)*sqrt(h);
        Y3=S(j)+h*mu*S(j)-sigma*S(j)*sqrt(h);
        S(j)=S(j)+0.5*h*mu*(Y1+S(j))+0.25*sigma*(Y2+Y3+2*S(j))
        *WP+0.25/sqrt(h)*sigma*(Y2-Y3)*(WP^2-h);

        Y1=S(j+1)+h*mu*S(j+1)-sigma*S(j+1)*WP;
        Y2=S(j+1)+h*mu*S(j+1)+sigma*S(j+1)*sqrt(h);
        Y3=S(j+1)+h*mu*S(j+1)-sigma*S(j+1)*sqrt(h);
        S(j+1)=S(j+1)+0.5*h*mu*(Y1+S(j+1))-0.25*sigma*(Y2+Y3+2*S(j+1))
        *WP+0.25/sqrt(h)*sigma*(Y2-Y3)*(WP^2-h);
    end
end
```

[8] main mit Varianzreduktion

```
S0 = 100;
T = 1;
t0 = 0;
P = 20; % Anzahl an Monte-Carlo-Simulationen

mu = 0.01;
sigma = 0.01;
M = 100; % Anzahl der Gitterpunkte
N = 30000  % Anzahl der Pfade in der Monte-Carlo-Simulation

h = (T-t0)/M;

format long;

randn('state',0)
tic;
for i=1:P
      euler1 = eulervr(M,h,S0,N,sigma,mu);
      mcseuler(i) = mcs(N,euler1);
end
Zeiteuler = toc

randn('state', 0)
tic;
for i=1:P
      heun1 = heunvr(M,h,S0,N,sigma,mu);
      mcsheun(i) = mcs(N,heun1);
end
Zeitheun = toc

ST = S0*exp(mu*T);

sum1=0;
sum2=0;
for i=1:P
    sum1 = sum1 + mcseuler(i);
    sum2 = sum2 + mcsheun(i);
end

fehlermcseuler = abs(1/P*sum1 - ST)/ST
fehlermcsheun = abs(1/P*sum2 - ST)/ST
```

[9] main mit $N_{Euler} = 3 \cdot N_{Heun}$

```
S0 = 100;
T = 1;
t0 = 0;
P = 20; % Anzahl an Monte-Carlo-Simulationen

mu = 0.01;
sigma = 0.01;
M = 100; % Anzahl der Gitterpunkte
N = 1000  % Anzahl der Pfade in der Monte-Carlo-Simulation

h = (T-t0)/M;

format long;

randn('state',0)
tic;
for i=1:P
      euler1 = euler(M,h,S0,3*N,sigma,mu);
      mcseuler(i) = mcs(3*N,euler1);
end
Zeiteuler = toc

randn('state', 0)
tic;
for i=1:P
      heun1 = heun(M,h,S0,N,sigma,mu);
      mcsheun(i) = mcs(N,heun1);
end
Zeitheun = toc

ST = S0*exp(mu*T);

sum1=0;
sum2=0;
for i=1:P
    sum1 = sum1 + mcseuler(i);
    sum2 = sum2 + mcsheun(i);
end

fehlermcseuler = abs(1/P*sum1 - ST)/ST
fehlermcsheun = abs(1/P*sum2 - ST)/ST
```

[10] main mit $M_{Euler} = 300$ und $M_{Heun} = 100$

```
S0 = 100;
T = 1;
t0 = 0;
P = 20; % Anzahl an Monte-Carlo-Simulationen

mu = 0.01;
sigma = 0.01;
M1 = 300;  % Anzahl der Gitterpunkte fuer das Euler-Verfahren
M2 = 100;  % Anzahl der Gitterpunkte fuer das Heun-Verfahren
N = 30000  % Anzahl der Pfade in der Monte-Carlo-Simulation

format long;

h = (T-t0)/M1;
randn('state',0)
tic;
for i=1:P
      euler1 = euler(M1,h,S0,N,sigma,mu);
      mcseuler(i) = mcs(N,euler1);
end
Zeiteuler = toc

h = (T-t0)/M2;
randn('state', 0)
tic;
for i=1:P
      heun1 = heun(M2,h,S0,N,sigma,mu);
      mcsheun(i) = mcs(N,heun1);
end
Zeitheun = toc

ST = S0*exp(mu*T);

sum1=0;
sum2=0;
for i=1:P
    sum1 = sum1 + mcseuler(i);
    sum2 = sum2 + mcsheun(i);
end

fehlermcseuler = abs(1/P*sum1 - ST)/ST
fehlermcsheun = abs(1/P*sum2 - ST)/ST
```

Literaturverzeichnis

[1] H. Georgii, Stochastik, de Gruyter, 2002. (4., überarbeitete und erweiterte Auflage 2009).

[2] L. Grüne, Numerische Methoden der Finanzmathematik, Vorlesungsskript, 2. Auflage, 2010.

[3] M. Günther & A. Jüngel, Finanzderivate mit MATLAB®, Vieweg Verlag, 2003.

[4] T. Höllbacher, Hedging mit Monte Carlo Algorithmen, Diplomarbeit, 2011.

[5] P. E. Kloeden & E. Platen, Numerical Solution of Stochastic Differential Equations, Springer–Verlag, 1992. (3rd revised and updated printing, 1999).

[6] K. Schäfer, Finanzmanagement, Vorlesungsskript, 2011.

Erklärung

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

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

Bayreuth, den 03. August 2012

_____

Johanna Weigand