Numerische Methoden II

Tim Hoffmann

23. Januar 2007

1 Numerische Beispiele

Numerische Methoden zur Approximation von Derivatpreisen:

- Trinomische Gittermethode
- Implizite Finite Differenzen
- Explizite Finite Differenzen
- Crank-Nicolson Methode

Ziel:
Überprüfung auf Genauigkeit und Verhalten der vier Methoden anhand zweier Beispiele mit bekannten exakten Preisen.

Derivat:
Europäische Call-Option: $P(T,11)$ mit Ausübung in $T = 1$ und Strike $K_1 = 0.627268$ und $K_2 = 0.55$ bei $r(T) = 0.05$
Exakte Preise für Call-Option im Zeitpunkt $t = 0$ von 0.031338 in Beispiel #1 und 0.082620 in Beispiel #2.

Approximative Preisermittlung:
Vasicek Modell:

$$dr(t) = -\alpha(r(t) - \mu)dt + \sigma d\tilde{W}(t)$$

mit folgenden Parametern:

$$\mu = 0.05$$
$$\alpha = 0.1$$
$$\sigma = 0.02$$

und

$$r(0) = 0.05$$

Transformation von $r(t)$:

$$X(t) = \left(r(t) - \mu - e^{-\alpha t}(r(0) - \mu)\right)\Big/\sigma$$

so dass $X(0) = 0$ und $dX(t) = -\alpha X(t) + d\tilde{W}(t)$
Unterteilung des Zeitintervalls [0,1] in M Schritte der Länge $\Delta t = 1/M$
Restriktion der Variable x: $x \in [x_0, x_N]$

1.1 Beispiel #1

Vergleich der Methoden:

1.) Erwarteter Preis zu Anzahl Zeitschritte bei festem k

Explizite Differenzen und Trinomische Gittermethode weisen nahezu identischen Verlauf auf. Daher im Weiteren Vernachlässigung der Trinomischen Methode.

Das Crank-Nicolson Verfahren Konvergiert mit Rate M^{-2}, wogegen die anderen Verfahren nur mit Rate M^{-1} gegen den exakten Wert streben. Die lässt noch nicht den Schluss zu, dass das CN Verfahren das Genaueste ist.

2.) Erwarteter Preis bzw. Preisfehler zu Gittergröße bei festem k

Bei der Implementierung der Verfahren ist die Größe des Gitters von größerer Bedeutung als die Anzahl der Zeitschritte.

CN Verfahren scheint weiterhin das Beste zu sein, da sowohl eine höhere Konvergenz als auch ein geringerer Fehler im Bezug auf die Gittergröße im Vergleich zu den Differenzenmethoden erreicht wird.

Jedoch benötigt das CN Verfahren die aufwendige Lösung eines Tridiagonalen Gleichungssystems, so dass bei gleicher Gittergröße Preise mit der Methode der expliziten Differenzen schneller berechnet werden können.

3.) Preisfehler zu Gittergröße bei variierendem k

Implizite Differenzen: kein Zusammenhang zwischen Genauigkeit und *k*

Explizite Differenzen & CN Verfahren: signifikante Änderungen bei variierendem *k*

1.2 Beispiel #2

Im Gegensatz zu Beispiel #1 fällt der Strike-Wert nicht immer auf einen Knoten. Dies hat zur Folge, dass wie im Beispiel der expliziten finiten Differenzen der erwartete Preis um den exakten Wert oszilliert.

Implizite Differenzen & CN Verfahren: geringe Oszillation, schnelle Konvergenz

Crank-Nicolson: hier: $k = 1$ liefert schlechtestes Ergebnis (*vgl. Beispiel #1*)

2 Simulationen

Motivation:

Ausgangsproblem: Berechnung von Derivatpreisen ohne analytische Preisformel

$$V(t) = \mathrm{E}_Q\left[\exp\left(-\int_t^T r(u)du \right) V(T) \middle| F_t \right]$$

Bekannte Numerische Methoden werden signifikant aufwendiger bei Hinzufügen von weiteren Dimensionen, d.h. bei Verwendung von Mehrfaktor-Modellen.

Alternative:
Monte Carlo Simulation:
Simuliere die Zufallsvariable

$$X(t) = \exp\left(-\int_t^T r(u)du \right) V(T)$$

zur Berechnung von $\mathrm{E}_Q[X]$.

2.1 Basis Monte Carlo Modell

Ausgangspunkt:
Ein-Faktor-Modell für *r(t)* mit SDE:

$$dr(t) = a(t, r(t))dt + b(t, r(t))d\tilde{W}(t)$$

Ziel:
Berechnung des Derivatpreises zum Zeitpunkt 0 mit Fälligkeit *T*, wobei *V(T)* bekannt ist.

Algorithmus:
Schritt 1:

Diskretisierung des Zeitintervalls [0,T] in n Teile der Länge $\Delta t = T/n$
Definiere: $t_j = j\Delta t$

Schritt 2:

Für Simulation *i* (*i* = 1, …., *N*) simuliere *n* iid Standard Normalverteilte Zufallsvariablen: $\varepsilon_j^{(i)} \sim N(0,1)$ für *j* = 1, …., *N*

Schritt 3:

Rekursive Berechnung von *r(t)* für jeden Pfad *i*:
Sei $r^{(i)}(0) = r(0)$
Für *j* = 1, …., *n*:

$$r^{(i)}(t_j) = r^{(i)}(t_{j-1}) + a(t_{j-1}, r^{(i)}(t_{j-1}))\Delta t + b(t_{j-1}, r^{(i)}(t_{j-1}))\sqrt{\Delta t}\varepsilon_j^{(i)}$$

Schritt 4:

Berechne Simulation *i*: $V^{(i)}(T) \equiv V(T, r^{(i)}(t_n))$

Schritt 5:

Berechnung des simulierten Wertes von *X*:

$$X_i = \exp\left(-\sum_{j=0}^{n-1} r^{(i)}(t_j)\Delta t \right) V^{(i)}(T)$$

Schritt 6:

Berechnung des Erwartungswertes:

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$$

Dies ist der erwartete Derivatpreis *V(0)*.

Fehlerquellen:

- Simulationsfehler: Anzahl der Simulationen endlich
- Diskretisierung des Zeitintervalls [0,*T*]:
  - Approximation des Integrals durch Summe
  - Rekursive Berechnung von $r^{(i)}(t_j)$ berücksichtigt nicht, dass im Allgemeinen simulierte Verteilung von der tatsächlichen abweicht.

Simulationsfehler:

Varianz: $S^2 = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})^2$

Standardabweichnung des Schätzers $\bar{X}$:

$$SE(\bar{X}) = \frac{S}{\sqrt{N}}$$

Falls S nicht klein ist relativ zu $\bar{X}$ muss, um eine gegebene Genauigkeit zu erreichen, *N* sehr groß sein.

Ziel:

Reduzierung der Standardabweichnung

→ *Latin Hypercube Sampling* und *Quasi Monte Carlo Methode*

2.2 Stratified und Latin Hypercube Sampling

Stratified Sampling:

Basis Monte Carlo Simulation ist äquivalent zur Simulation von uniformverteilten Zufallsvariablen:

Allgemein: Seien $U_i \sim U[0,1]$ uniformverteilt, dann gilt:

$$X_i = F_X^{-1}(U_i)$$

und $F_X(x)$ ist die Gesamtverteilung von *X*.

Ersetze simulierte Werte $U_1,....,U_N$ durch deterministische Werte, beispielsweise folgendermaßen:

$$U_i = \frac{i-0.5}{N} \text{ für } i = 1, \ldots, N$$

Für große *N* konvergiert erwarteter Wert mit Rate $1/N$ gegen den exakten Wert.

→ signifikante Verbesserung zur Basis Methode

Nachteil: Sehr aufwendig im mehrdimensionalen Fall

Latin Hypercube Sampling:

Sehr ähnlich zur Basis Methode, jedoch unterschiedlich bei der Simulation der $\varepsilon_j^{(i)}$.

Modifikation des 2. Schrittes:

Schritt 2a:

Für jedes $j = 1, \ldots, d$ sei $P_j = (P_j^{(1)}, P_j^{(2)}, \ldots, P_j^{(N)})$ eine zufällige Permutation von (1, 2, …., N), wobei P_j unabhängig.

Schritt 2b:

Für i = 1, …., N und j = 1, …., d seien $\xi_j^{(i)}$ iid uniformverteilte Zufallsvariablen auf [0,1]. Sei

$$U_j^{(i)} = \frac{P_j^{(i)} - 1 + \xi_j^{(i)}}{N}$$

D.h.: jedes Intervall $[0,1/N], [1/N, 2/N], \ldots, [(N-1)/N, 1]$ enthält somit exakt ein $U_j^{(i)}$

Schritt 2c:

Setze $\varepsilon_j^{(i)} = \Phi^{-1}(U_j^{(i)})$

Vorteil:

Gleichmäßige Verteilung der Zufallsvariablen im Gitter

Nachteil:

Beim Hinzufügen einer weiteren Simulation muss nicht nur diese zusätzliche sondern alle $N + 1$ Simulationen neu berechnet werden.

2.3 Quasi Monte Carlo Methode

Zwei Varianten zur Modifikation des 2. Schrittes:

2.3.1 Halton Sequences

Schritt 2a:

Seien $p_1, \ldots, p_n$ die ersten n Primzahlen

Schritt 2b:

Für jedes $j = 1, \ldots, n$, $i = 1, 2, \ldots, N$ transformiere die Zahl i zur Basis p, d.h.:

Finde die eindeutige Menge $a_j(i,l)$ mit $0 \le a_j(i,l) < p$ für $l = 0, 1, \ldots$, so dass

$$i = \sum_{l=0}^{\infty} a_j(i,l)\, p_j^l = \sum_{l=0}^{m(i,p_j)} a_j(i,l)\, p_j^l ,$$

wobei $m(i, p_j)$ der größte Index der Ziffer ungleich Null zur Basis p_j ist.

Schritt 2c:

Rücktransformation in Dezimalsystem:

$$U_j^{(i)} = \sum_{l=0}^{m(i,p_j)} a_j(i,l)\, p_j^{-l-1}$$

Schritt 2d:

Setze $\varepsilon_j^{(i)} = \Phi^{-1}(U_j^{(i)})$

In diesem Schritt wird mit Hilfe der inversen Funktion der Gesamverteilungsfunktion ein quasi-zufälliges Muster der Standard Normalverteilung generiert.

2.3.2 Faure Sequences

Schritt 2a:

Sei p die kleinste Primzahl mit $p \geq n$.

Schritt 2b:

Sei $m = \min\{k \in \square : k > \log(N)/\log(p)\} - 1$

D.h.: $m+1$ ist der größte benötigte Index zur Darstellung jeder Zahl von 1 bis N zur Basis p

Schritt 2c:

Definiere Matrix $C = [c_{kl}]_{k,l=0}^{m}$ wie folgt:

$c_{kl} = 0$ für $k > l$

$c_{0l} = 1$ für $l = 0,....m$

$c_{kk} = 1$ für $k = 1,....m$

$c_{kl} = c_{k,l-1} + c_{k-1,l-1}$ für $k = 1,....,m-1$ und $l = k+1,....,m$

Schritt 2d:

Für j = 1und für jedes i = 1, 2, …., N schreibe i analog zu Halton zur Basis p.

Definiere Vektor:

$$a_j(i,k) = \left(a_j(i,0),....,a_j(i,m)\right)'$$

Schritt 2e:

Für jedes j = 2, …., n und k = 0, ….., m sei

$$a_j(i,k) = \sum_{l=0}^{\infty} c_{kl} a_{j-1}(i,l) \bmod p$$

Vektornotation:

$$a_j(i) = C a_{j-1}(i) \bmod p$$

Dieser Schritt verhindert das Duplizieren von Zahlen.

Schritt 2f:

Rücktransformation in Dezimalsystem:

$$U_j^{(i)} = \sum_{l=0}^{m(i,p_j)} a_j(i,l)\, p_j^{-l-1}$$

Schritt 2g:

Setze $\varepsilon_j^{(i)} = \Phi^{-1}(U_j^{(i)})$

Vorteil:

- Hinzufügen weiterer Simulationen problemlos
- Generierte Werte sind deterministisch ohne Einfluss von Zufälligkeit
- Gleichmäßige Verteilung der Zufallsvariablen im Gitter