G 59071

RISIKO MANAGER

23 . 2014

[KREDITRISIKO] [MARKTRISIKO] [LIQUIDITÄTSRISIKO] [OPRISK] [ERM] [REGULIERUNG]

FIRM
Frankfurter Institut für Risikomanagement und Regulierung

Inhalt

WWW.RISIKO-MANAGER.COM

Optionspreisberechnung via Fast Fourier Transform (Teil 4)

Divide et impera

Der vorangegangene Beitrag (siehe RISIKO MANAGER 20/2014) hat die Bewertung europäischer Call-Optionen mittels Fourierinversion angesprochen. Diese numerische Berechnungsmethode führt den Optionspreis für gegebenen Strike und fixe Maturität auf die Auswertung eines uneigentlichen Integrals zurück. Der Integrand besteht dabei im Wesentlichen aus der charakteristischen Funktion des logarithmierten Aktienkurses, welche für Lévy-Modelle durch die Cumulant-Funktion gegeben ist. In diesem Teil unserer Risikomanagementserie beschäftigen wir uns mit der geschickten Auswertung des uneigentlichen Integrals unter Einsatz der Fast-Fourier-Transformation (FFT).

Die Fast-Fourier-Transformation stellt eine der bedeutendsten Innovationen im Bereich des wissenschaftlichen Rechnens dar, deren Anfänge bis zu Carl Friedrich Gauß zu Beginn des 19. Jahrhunderts zurückreichen [für eine Übersicht zur Entwicklungshistorie siehe Heideman et al. 1985]. Unser Ziel ist die effiziente Berechnung von Koeffizienten β_j (vgl. ▶ **Gleichung 01**) für komplexe Werte $f_k \in \mathbb{C}, k=0,\dots,N-1$.

▶ **Gleichung 01**

$$\beta_j = \frac{1}{N}\sum_{k=0}^{N-1} f_k e^{-2\pi i jk/N}, \quad j=0,\dots,N-1,$$

Koeffizienten dieser Art treten beispielsweise bei der Interpolation äquidistanter Stützpunkte $(x_k, f_k), x_k = 2\pi k/N, k=0,\dots,N-1$, durch ein trigonometrisches Polynom $p(x)=\sum_{j=0}^{N-1}\beta_j \exp(ijx)$ auf (das heißt, die β_j sollen so gewählt werden, dass $p(x_k)=f_k$ für alle $k=0,\dots,N-1$ gilt). Die

Fortsetzung auf Seite 7

Fortsetzung von Seite 1

Abbildung der Funktionswerte f_k auf die Koeffizienten β_j wird Diskrete Fouriertransformation (DFT) genannt und findet neben der in diesem Kapitel skizzierten Verwendungsmöglichkeit beispielsweise Anwendung in der digitalen Signalverarbeitung, der Bildbearbeitung oder der Lösung partieller Differentialgleichungen [für eine umfangreiche Übersicht, siehe Bracewell 2000]). Wie wir später in diesem Kapitel sehen werden, lässt sich auch die Berechnung von Optionspreisen für eine Menge $k_j j=0,\ldots,N-1$, von Strikes unter bestimmten Voraussetzungen in Summen wie in ▶ **Gleichung 01** überführen.

Die nächsten Abschnitte wenden sich den folgenden drei Fragen zu:

(a) Was ist das Problem bei der Auswertung von ▶ **Gleichung 01**?

(b) Wie kann dieses Problem behoben bzw. entschärft werden?

(c) Welche Relevanz hat das für die Berechnung von Optionspreisen?

Was ist das Problem bei der Auswertung von ▶ Gleichung 01?

Der Schwachpunkt der Berechnungsvorschrift in ▶ **Gleichung 01** ist numerischer Natur und liegt in der potenziell langen Auswertungszeit, die zur Ermittlung der Koeffizienten β_j für große Werte von N notwendig ist. Stünde man vor der Aufgabe, ▶ **Gleichung 01** zu implementieren, sähe ein erster, „naiver" Algorithmus vermutlich wie in ▶ **Infobox 01** aus (sämtliche in diesem Kapitel vorkommenden Programmzeilen sind für das Programm MATLAB konzipiert, jedoch leicht für andere Programmiersprachen/-software adaptierbar).

Für jedes β_j müssen N (komplexe) Multiplikationen $f_k e^{-2\pi ijk/N}, k=0,\ldots,N-1$, durchgeführt und die Ergebnisse in $N-1$ (komplexen) Additionen zusammengerechnet werden. Da N Koeffizienten $\beta_j, j=0,\ldots,N-1$, berechnet werden müssen, ergibt sich insgesamt ein quadratischer Aufwand, welchen wir mit $\Theta(N^2)$ bezeichnen. Anders ausgedrückt: Bei einer Verdopplung der Anzahl der Koeffizienten vervierfacht sich die Anzahl der Rechenoperationen, was in vielen Anwendungsfällen zu ungenügenden Rechenzeiten führt und die Eignung der DFT ohne weitere Modifikationen in Frage stellt. ▶ **Abb. 01** visualisiert die Problematik.

▶ **Info-Box 01**

Algorithmus (1): Direkte Berechnung von ▶ Gleichung 01

```
function beta = DFT_slow( f )

N=length(f);
beta=zeros(N,1);

for j=0:N-1
   sum=0;
   for k=0:N-1
      sum=sum+f(k+1)*exp(-2*pi*1i*j*k/N);
   end
   beta(j+1)=sum/N;
end

end
```

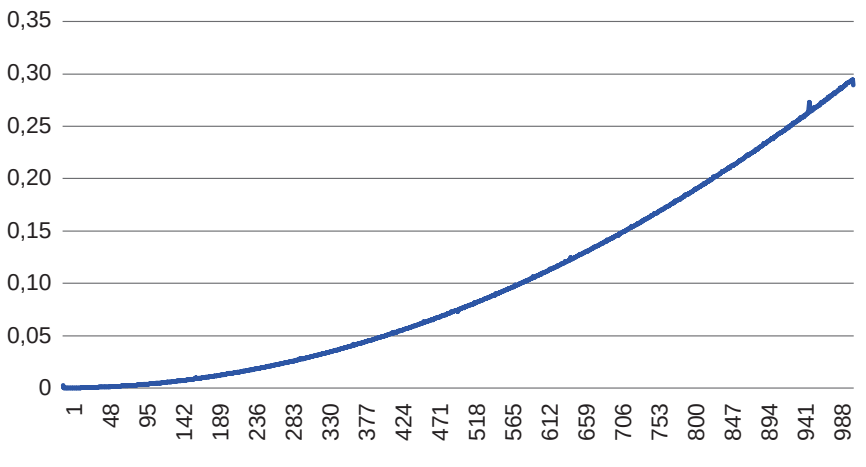
Der Graph repräsentiert für $N=1,\ldots,1024$ jeweils die benötigte Zeitspanne zur Ermittlung der β_j bei „naiver" Implementierung von ▶ **Gleichung 01** mittels Algorithmus 1 (vgl. ▶ **Infobox 01**). Die f_k, welche als Input in die Funktion eingehen, wurden dabei jeweils vor Aufruf des Algorithmus zufällig im Intervall [0,1000] gezogen. Es bestätigt sich die erwartete quadratische Zunahme der Berechnungszeit für steigendes N.

Wie kann das beschriebene Problem behoben bzw. entschärft werden?

Zur Minderung der Laufzeitproblematik kommt die eingangs angesprochene FFT zum Einsatz, welche ein mächtiges Werkzeug zur Berechnung der DFT in ▶ **Gleichung 01** darstellt. Das zu Grunde liegende Paradigma lautet „divide et impera". Die Formulierung „Divide et impera" wird teilweise Niccolò Machiavelli (1469–1527) zugeschrieben, der in seinem 1532 erschienenen Buch „Der Fürst" den Fürsten Medici erklärt, wie sie ihre Herrschaft ausüben sollten. Im Kern geht es darum, dass man ein Volk oder eine Gruppierung in Untergruppen aufspalten sollte, um sie leichter zu beherrschen. In unserer Anwendung heißt das: Durch geschickte, rekursive Zerlegung der Summen in ▶ **Gleichung 01** kann das originäre Problem in viele kleinere Teilprobleme gesplittet werden, deren Berechnungsaufwand

▶ **Abb. 01**

Berechnungsdauer in Sekunden (y-Achse) zur Auswertung der β_j in ▶ Gleichung 01 in Abhängigkeit von N (x-Achse) bei Aufruf von Algorithmus (1) auf einem Standard-PC



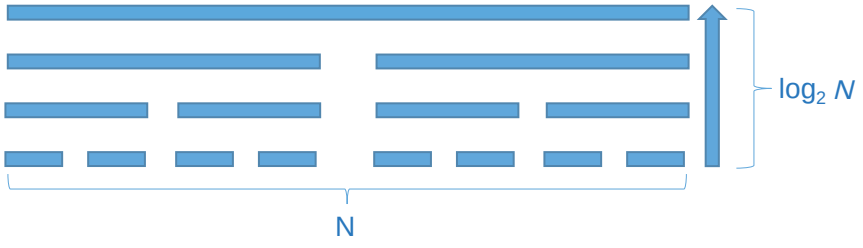
jeweils deutlich geringer ist und auch nach der Aggregation zum gewünschten Gesamtergebnis in einer deutlich verkürzten Laufzeit resultiert. Wie wir am Ende dieses Abschnitts sehen werden, führt die FFT zu einer Laufzeit der Größenordnung $\Theta(N \log_2(N))$ und damit einer signifikanten Beschleunigung der initialen $\Theta(N^2)$-Rechendauer. Die grundlegende Idee ist in ► **Abb. 02** für $N=2^3=8$ dargestellt. Das ursprüngliche Problem wird in zwei Teilprobleme „halber Komplexität" geteilt, welche wiederum auf analoge Art und Weise weiter zerlegt werden. Auf unterster Ebene (siehe ► **Abb. 02**) ergeben sich somit N sehr leicht zu lösende Teilprobleme geringer Komplexität. Die Lösung dieser Probleme muss schließlich geeignet aggregiert werden, um die Lösung des übergeordneten Problems zu erhalten. Da die Aggregation (wie wir sehen werden) einen Aufwand der Größenordnung $\Theta(N)$ in Anspruch nimmt und es insgesamt $\Theta(\log_2(N))$ Aggregationen gibt, entsteht die genannte Laufzeit in der Größenordnung $\Theta(N \log_2(N))$.

Wegen der hohen Relevanz haben sich viele Publikationen der Entwicklung unterschiedlicher FFT-Algorithmen gewidmet. Eines der frühesten und bekanntesten Verfahren stammt von Cooley/Tuckey [vgl. Cooley/Tuckey 1965]. In Anlehnung an Freund/Hoppe [Freund/Hoppe 2007, S. 81ff] stellen wir in diesem Artikel ein weiteres, leicht verständliches Verfahren von Gentleman/Sande [vgl. Gentleman/Sande 1966] vor. Eine genaue Herleitung findet sich in ► **Infobox 04**. Der daraus resultierende Algorithmus zur FFT für $N=2^n$ ist in ► **Infobox 02** widergegeben.

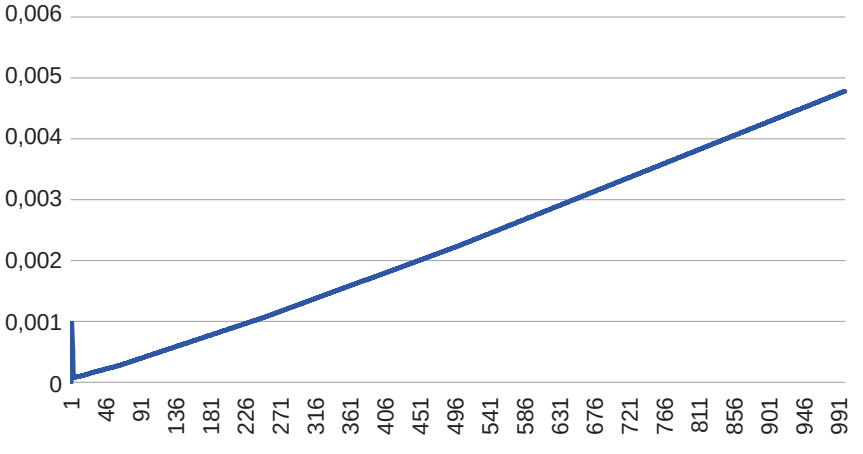
Um einen Vergleich mit ► **Abb. 01** durchführen zu können, rufen wir Algorithmus 2 (siehe ► **Infobox 02**) für $N=2^n$, $n=0,\ldots,10$, auf. Die so erhaltenen Laufzeitpaare extrapolieren wir linear für die restlichen $N=1,\ldots,1024$, um eine grobe Vorstellung von der Rechenzeitentwicklung zu erhalten. ► **Abb. 03** zeigt das Ergebnis.

Abgesehen von einem „Ausreißer" für $N=1$, der dem erstmaligen Aufruf des Algorithmus geschuldet ist, zeigt sich ein nahezu linearer Verlauf für die Laufzeitentwicklung. Diese Beobachtung steht im Einklang mit der erwarteten Rechenzeit der Größenordnung $\Theta(N \log_2 N)$. Für zunehmendes N weist $\log_2 N$ eine abnehmende Steigung auf (im Grenzwert für unendlich großes N ist diese schließlich Null), weshalb optisch der Eindruck eines nahezu

Zerlegungsvorschrift bei der FFT ► **Abb. 02**



Berechnungsdauer in Sekunden (y-Achse) zur Auswertung der β_j in ► Gleichung 01 in Abhängigkeit von N (x-Achse) bei Aufruf von Algorithmus (2) auf einem Standard-PC ► **Abb. 03**



► **Info-Box 02**

Algorithmus (2): FFT zur Auswertung von ► Gleichung 01

```
function beta = DFT_fast( f )

N=length(f);
n=log2(N);

f_temp1=f;

for m=n:-1:1
   M=2^(m-1);
   R=2^(n-m);
   f_temp2=reshape(f_temp1,2*R,M);
   for r=0:R-1
      for k=0:M-1
         f_temp2(r+1,k+1)=f_temp1(r+1,k+1)+f_temp1(r+1,k+M+1);
         f_temp2(r+R+1,k+1)=(f_temp1(r+1,k+1)-f_temp1(r+1,k+M+1))*exp(-
         2*pi*1i*k/2^m);
      end
   end
   f_temp1=f_temp2;
end

beta=f_temp1/N;
end
```

Adjustiertes Laufzeitenverhältnis der beiden vorgestellten Algorithmen für $N=2^n, n=1,...,16$ ▶ Tab. 01

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Verhältnis	0,21	0,39	0,42	0,39	0,42	0,47	0,50	0,54	0,57	0,59	0,60	0,60	0,60	0,61	0,61	0,60

linearen Anstiegs der Berechnungsdauer entsteht. Vergleicht man die Laufzeit der Algorithmen für $N=1024$, stellt man fest, dass der Zeitspanne von ca. 0,3 Sekunden bei Algorithmus 1 (vgl. ▶ **Infobox 01**) eine Dauer von ca. 0,005 Sekunden bei Algorithmus 2 (vgl. ▶ **Infobox 02**) gegenübersteht. Die naive DFT-Implementierung beansprucht in diesem Fall daher die 60-fache Rechenzeit der FFT-Implementierung.

Für eine weitere empirische Überprüfung der Laufzeiten der Algorithmen vergleichen wir deren Verhältnis für variierende Werte von N. Falls Algorithmus 1 (vgl. ▶ **Infobox 01**) wie behauptet $\Theta(N^2)$ und Algorithmus (2) $\Theta(N \log_2 N)$ Rechenoperationen benötigt, so sollte das Verhältnis „(N-mal Laufzeit Algorithmus (1))/(2^n-mal Laufzeit Algorithmus (2))" in etwa konstant sein. ▶ **Tab. 01** zeigt das Verhältnis für die Werte $n=1,...,16$ auf.

Der Vermutung entsprechend zeigt sich für mittlere und große Werte von n ein nahezu konstantes Verhältnis von ca. 0,6. Die Abweichungen für kleinere Werte von n lassen sich durch externe Effekte wie beispielsweise initiale Speicherbelegung und Belegung der Ergebnisvariablen erklären, welche in diesen Fällen relativ gesehen stärker ins Gewicht fallen und das Verhältnis verzerren.

Welche Relevanz hat die FFT für die Berechnung von Optionspreisen?

Der zurückliegende Teil unserer Artikelserie hat gezeigt, dass der Preis $C_T(k)$ einer Europäischen Call-Option mit Laufzeit T und Strike $\exp(k)$ (k wird daher im Folgenden als log-Strike bezeichnet) unter bestimmten technischen Voraussetzungen angenähert werden kann mittels ▶ **Gleichung 02** für bestimmte $\alpha > 0$, wobei Re(.) den Realteil des in Klammern stehenden Ausdrucks bezeichnet und $\psi_T(v)$ neben v und α von der Fouriertransformation des logarithmierten Aktienkurses zum Zeitpunkt T im zu Grunde liegenden Modell für den Aktienpreisprozess abhängt.

Darüber hinaus illustrieren wir in Anlehnung an Carr/Madan [vgl. Carr/Madan 1999], dass für ein äquidistantes Log-Strike-Gitter k_j, j=0,...,N-1, bei Approximation des Inversionsintegrals durch eine geeignete Summe die Preise $C_T(k_j)$ in die Form von ▶ **Gleichung 01** gebracht und folglich durch Anwendung der FFT für $N=2^n$ extrem schnell ausgewertet werden können.

Unterteilt man das Intervall [0,A] in $N-1$ Intervalle $[v_l, v_{l+1}]$, $l=0,...,N\text{-}2, 0=v_0<...<v_{N-1}=A$, der Länge Δv (d.h. $A=(N-1)\Delta v$) und wendet die Trapezregel an [für Details zu Hintergrund und Fehlerabschätzungen, siehe Freund/Hoppe 2007, S. 166f], folgt ▶ **Gleichung 03**.

Für eine äquidistante Menge von log-Strikes $k_j=-b+\Delta k\, j, j=0,...,N-1$, mit einem $b\in\mathbb{R}$ und einer Schrittweite $\Delta k>0$, sodass $\Delta v\Delta k=2\pi/N$ gilt, ergibt sich ▶ **Gleichung 04** und in der Folge [neue Gleichung S. 10].

Abgesehen von den Korrekturtermen $\psi_T(0)+e^{-iv_{N-1}k_j}\psi_T(v_{N-1})$ entspricht die Auswertung der Call-Preise $C_T(k_j), j=0,...,N-1$, damit in struktureller Hinsicht genau der Berechnung der $\beta_j, j=0,...,N-1$, in ▶ **Gleichung 01**. Die frei wählbare Variable b, die in der Definition der k_j vorkommt, dient der Verschiebung des Bereichs, in dem die Call-Preise ausgewertet werden sollen. Eine symmetrische Spanne $[-(N-1)\Delta k/2, (N-1)\Delta k/2]$ der k_j erhält man beispielsweise durch die Wahl $b=(N-1)\Delta k/2$. Zusammengefasst sieht eine mögliche Implementierung des FFT-Algorithmus zur simultanen Auswertung von Call-Preisen für eine Menge von Strikes $\exp(k_j), j=0,...,N-1$, wie in ▶ **Infobox 03** widergegeben aus (man beachte, dass die

▶ **Gleichung 02**

$$C_T(k) \approx \frac{\exp(-\alpha k)}{\pi} Re\left(\int_0^A e^{-ivk}\psi_T(v)dv\right),$$

▶ **Gleichung 03**

$$C_T(k) \approx \frac{\exp(-\alpha k)}{\pi}\Delta v\ \mathrm{Re}\left(\sum_{l=0}^{N-1} e^{-iv_l k}\psi_T(v_l) - \frac{1}{2}\left(\psi_T(0)+e^{-iv_{N-1}k}\psi_T(v_{N-1})\right)\right).$$

▶ **Info-Box 03**

Algorithmus (3): FFT zur Auswertung von ▶ Gleichung 03

```
% 1) Lege Inputparameter fest
% – Fixiere T, Parameter des Aktienpreisprozesses und α (:=alpha)
% – „Erstelle" die charakteristische Funktion ψ_T (:=psi)
% – Fixiere N=2^n für eine natürliche Zahl n
% – Wähle Approximations „Feinheit" Δv (:= Delta_v) und log-Strike
%   Schrittweite Δk (:=Delta_k), so dass Δv·Δk=2π/N gilt (hier besteht ein %
    Tradeoff)
% – Bestimme auszuwertenden Strike-Bereich durch Fixierung von b

% 2) Initialisierung der FFT-Parameter
k=-b+Delta_k*(0:N-1)';

f=zeros(1,N);
for l=0:N-1
    f(l+1)=exp(1i*l*b*Delta_v)*psi(v(l+1));
end

% 3) Berechnung der Call-Preise unter Zuhilfenahme von Algorithmus (2)
C=N*DFT_fast (f);
C=C-1/2*(psi(0)+exp(-1i*(N-1)*Delta_v*k')*psi((N-1)*Delta_v));
C=exp(-alpha*k)/pi*Delta_v.*real(C);
```

k_j zwar äquidistant gewählt werden, es sich hierbei aber um logarithmierte Strikes handelt und die Strikes im eigentlichen Sinne durch $K_j = \exp(k_j)$ gegeben sind). Der Vektor C gibt dabei die Werte der Call-Optionen für die Strikes $\exp(k_j)$ zurück.

Zum Abschluss dieses Kapitels illustrieren wir das vorgestellte Verfahren anhand konkreter Beispiele. Zur Überprüfung der Funktionalität der FFT beginnen wir mit dem Black-Scholes (BS)-Modell, in welchem die Call-Preise europäischer Optionen bekannt sind und als Vergleichsmaßstab dienen können. Ausgehend von einem Aktienkursprozess $\{S_t\}_{t\geq 0}$ der in ▶ **Gleichung 05** widergegebenen Form

▶ **Gleichung 05**

$$S_t = S_0 \exp\left(\left(r - \frac{\sigma^2}{2}\right)t + \sigma W_t\right),$$

unter dem risikoneutralen Wahrscheinlichkeitsmaß mit dem risikolosen Zinssatz r, der Volatilität σ und der Brownschen Bewegung $\{W_t\}_{t\geq 0}$ ist die charakteristische Funktion des logarithmierten Aktienkurses zum Zeitpunkt T gegeben durch ▶ **Gleichung 06**.

Für die Anwendung der FFT gilt es, die Größen $N=2^n$ (die Anzahl der Strikes, für die Call-Preise berechnet werden), Δv (die Feinheit der Diskretisierung, welche für die Berechnung der Summe in ▶ **Gleichung 04** herangezogen wird) und b (Verschiebungskonstante für den Bereich, in welchem die N Call-Preise berechnet werden) festzulegen. Nach Fixierung dieser Werte liefert die FFT eine simultane Auswertung von Black-Scholes-Call-Preisen für die log-Strikes $-b, -b+\Delta k, \ldots, -b+\Delta k\,(N-1)$ (und damit für die Strikes $\exp(-b), \ldots, \exp(-b+\Delta k\,(N-1))$, wobei Δk durch die Forderung $\Delta v \Delta k = 2\pi/N$ bestimmt ist. Hat man beispielsweise den risikolosen Zinssatz $r=0{,}02$ und eine Aktie mit den Werten $S_0=66, \sigma=0{,}15$ gegeben und interessiert sich für die Bewertung europäischer Call-Optionen im Bereich [60,72], würde man sinnigerweise b so wählen, dass $\exp(-b)=60$ gilt, um eine möglichst feine Auswertung der Call-Preise im ausgewählten Bereich zu erhalten. Für die exemplarische Wahl von $N=1024$ und $\Delta v=0{,}25$ erhalten wir acht Werte im Strikebereich [60,72], welche in ▶ **Tab. 02** den bekannten Call-Preisen im BS-Modell gegenübergestellt sind.

▶ **Gleichung 04**

$$\sum_{l=0}^{N-1} e^{-iv_l k_j}\psi_T(v_l) = \sum_{l=0}^{N-1} e^{-il\Delta v(-b+\Delta k\, j)}\psi_T(v_l) = \sum_{l=0}^{N-1} \underbrace{e^{-ilj\Delta v\Delta k}}_{=e^{-2\pi ijl/N}} \underbrace{e^{ilb\Delta v}\psi_T(v_l)}_{:=f_l}$$

▶ **neue Gleichung**

$$C_T(k_j) \approx \frac{\exp(-\alpha k_j)}{\pi}\Delta v\, \mathrm{Re}\left(\sum_{l=0}^{N-1} e^{-\frac{2\pi ilj}{N}} f_l - \frac{1}{2}\left(\psi_T(0) + e^{-iv_{N-1}k_j}\psi_T(v_{N-1})\right)\right).$$

▶ **Gleichung 06**

$$\phi_T(u) = \exp\left(iu\left(\ln S_0 + \left(r - \frac{\sigma^2}{2}\right)T\right) - \frac{\sigma^2}{2}u^2 T\right), \quad u \in \mathbb{R}.$$

▶ **Tab. 02**

Vergleich der analytisch berechneten Call-Preise im BS-Modell mit der FFT.

Strike	FFT	Analytisch
60	6,496974984	6,496974983
61,49	5,205083047	5,205083046
63,02	4,004769052	4,004769051
64,58	2,939001074	2,939001073
66,19	2,043388116	2,043388115
67,83	1,337266900	1,337266899
69,52	0,818846034	0,818846033
71,24	0,466622376	0,466622376

▶ **Gleichung 07**

$$\phi_T(u) = \exp\left(iu\left(log(S_0) + \left(r + \frac{1}{\nu}\right)T\right)\right) * \left(1 - i\theta\nu + \frac{\sigma^2 u^2 \nu}{2}\right)^{-\frac{T}{\nu}}, \quad u \in \mathbb{R},$$

▶ **Tab. 03**

Festlegungen zur Spezifikation des VG-Modells

T	S_0	r	θ	w	ν	α	N	Δv	Δk
1/3	95	0,02	-0,1	0,21	2	1,5	$4096 = 2^{12}$	0,25	$2\pi/(N\Delta v)$

Es zeigt sich, dass die getroffene Wahl von Δv eine sehr genaue Auswertung erlaubt und die Preise der FFT in diesem Beispiel eine extrem hohe Genauigkeit der Größenordnung 10^{-8} aufweisen.

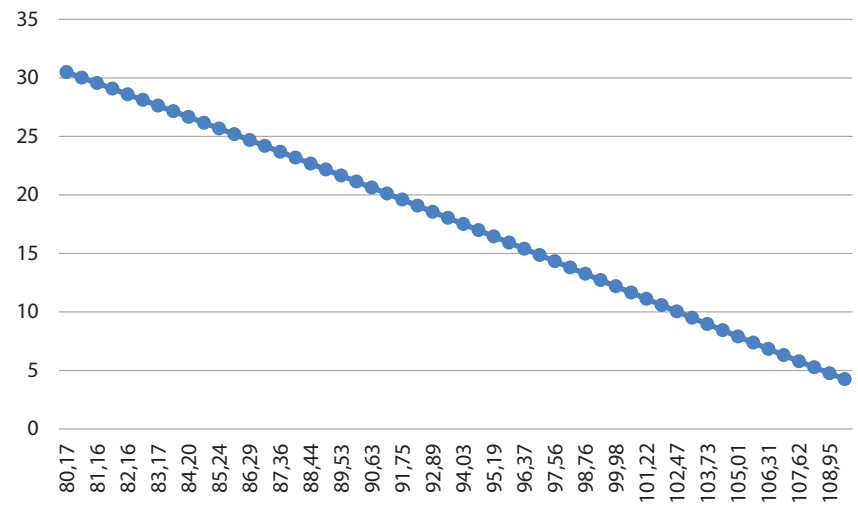
Ein weiteres Beispiel ist das Variance Gamma (VG)-Modell aus Carr et al. [vgl. Carr et al. 1998] mit der charakteristischen Funktion des logarithmierten Aktienkurses zum Zeitpunkt T unter dem risikoneutralen Maß gemäß ▶ **Gleichung 07** mit dem Spotpreis S_0, dem risikolosen Zinssatz r und den Modellparametern $\theta \in \mathbb{R}$, $\sigma > 0, \nu > 0$. Die in ▶ **Tab. 03** zusammengefassten Festlegungen haben wir zur Spezifikation des VG-Modells und zur Durchführung von Algorithmus 3 (vgl. ▶ **Info-Box 03**) getroffen.

Ferner wählen wir $b=(N-1)\Delta k/2$, sodass $k_j, j=0,\ldots,N-1$, äquidistante Werte in etwa im Intervall [–12,12] annimmt. Ungeachtet der daraus resultierenden großen Strike-Spanne für $\exp(k_j)$ erhält man beispielsweise im Intervall [80,110], welches aufgrund des Spotpreises $S_0=95$ für At-the-money-Optionen interessant ist, 52 Call-

Preise. Zwar ist zu beachten, dass sich die Rasterung $\exp(k_{j+1}) - \exp(k_j)$ aufgrund des äquidistanten Log-Strike-Gitters für zunehmendes j vergröbert. Jedoch reicht diese für die 52 Call-Preise im Beispiel von 0,49 (für Strikes bei 80) bis 0,67 (für Strikes bei 110) und ermöglicht damit eine insgesamt recht feine Auswertung der Optionspreise. Ein weiterer potenzieller Kritikpunkt ist die Tatsache, dass durch die FFT 4096 Call-Optionen ausgewertet werden, von denen möglicherweise nur 52 aus praktischer Sicht relevant sind, weshalb man direkt Gleichung (2) für die k von Interesse berechnen könnte. Dem ist entgegen zu halten, dass die FFT zum Einen trotz Evaluation aller N Preise mit einem Aufwand der Größenordnung $\Theta(N\log_2(N)) = \Theta(4096 * 12)$ einen Effizienzvorteil gegenüber der mit der naiven Auswertung der 52 Optionen verbundenen Anzahl $\Theta(4096 * 52)$ an Rechenoperationen besitzt. Zum Anderen kann durch Variation der Parameter b und Δk die Feinheit der zu berechnenden Werte im anwendungsrelevanten Bereich gesteuert werden. Eine grafische Übersicht der mit FFT berechneten 52 Call-Preise ist in ► **Abb.** 04 gegeben. Die Auswertungszeit zur initialen Belegung und Durchführung von Algorithmus 3 (vgl. ► **Info-Box** 03) beträgt rund eine Viertelsekunde. □

► Abb. 04

Resultierende Call-Optionspreise (y-Achse) im Strike-Intervall [80,110] (x-Achse) bei Auswertung von ► Gleichung 01 mittels Algorithmus 2.



Zusammenfassung und Ausblick

Dieses Kapitel hat sich mit der Fast-Fourier-Transformation auseinandergesetzt, einem mächtigen numerischen Verfahren zur beschleunigten Auswertung diskreter Fouriertransformationen. Da die FFT neben verschiedensten Anwendungsgebieten auch bei der simultanen Berechnung von Call-Preisen für eine Vielzahl von Strikes zur Anwendung kommen kann, ist das Verständnis um ihre Notwendigkeit, ihre Funktionsweise und ihre Implementierung von großer Bedeutung. Diese drei Aspekten haben wir jeweils aufgegriffen und anhand von konkreten Algorithmen erläutert.

Quellenverzeichnis sowie weiterführende Literaturhinweise:

***Bracewell, R. N. (2000)**: The Fourier Transform and its Applications. McGraw-Hill Higher Education.*

***Carr, P./Chang, E. C./Madan, D. B. (1998)**: The Variance Gamma Process and Option Pricing, in: European Finance Review 2: 79–105.*

***Carr, P./Madan, D. B. (1999)**: Option valuation using the fast Fourier transform, in: The Journal of Computational Finance 2(4): 61–73.*

***Cooley, J. W./Tukey, J. W. (1965)**: An Algorithm for the Machine Calculation of complex Fourier Series. Mathematics of Computation 19: 297–301.*

***Freund, R. W./Hoppe, H. W. (2000)**: Stoer/Bulirsch: Numerische Mathematik 1., 10. Auflage, Springer.*

***Gentleman, W.M./Sande, G. (1966)**: Fast Fourier Transforms – For Fun and Profit, in: Proceedings AFIPS Fall Joint Computer Conference 29: 503-578. Spartan Books.*

***Heideman, M.T./Johnson, D. H./Burrus, C. S. (1985). Gauss and the History of the Fast Fourier Transform. Archive for History of Exact Sciences 34(3)**:265-277, 3. Auflage, Springer.*

Autor

Steffen Schenk ist wissenschaftlicher Mitarbeiter am Lehrstuhl für Finanzmathematik der Technischen Universität München. Er ist Autor einiger Fachpublikationen in den Gebieten Finanz- und Versicherungsmathematik, Risikomanagement und Stochastik.

Wir bedanken uns herzlich bei KMPG AG für das Sponsoring des KPMG Center of Excellence in Risk Management an der Technischen Universität München, über welches die Stelle von Steffen Schenk finanziert wird.

▶ Info-Box 04

Aufgrund des „Teile und herrsche“-Konstruktionsprinzips erzielt die FFT den größten Effizienzgewinn für Zweierpotenzen von N. Es gelte daher $N=2^n, n\in\mathrm{N}$. Wenn wir die Hilfsgröße $\in_n := \exp(-2\pi i/2^n)$ definieren, kann Gleichung (1) umgeformt werden zu

$$N\beta_j = \sum_{k=0}^{N-1} f_k e^{-2\pi i jk/N} = \sum_{k=0}^{N-1} f_k \left(e^{-2\pi i/2^n}\right)^{jk} = \sum_{k=0}^{N-1} f_k \epsilon_n^{jk}, \qquad j = 0, \ldots, N-1.$$

Betrachtet man gerade und ungerade j getrennt, erhält man für $M := N/2$ und $h = 0, \ldots, M$

$$N\beta_{2h} = \sum_{k=0}^{N-1} f_k \epsilon_n^{2hk} = \sum_{k=0}^{M-1} \left(f_k \epsilon_n^{2hk} + f_{k+M} \underbrace{\epsilon_n^{2h(k+M)}}_{=\epsilon_n^{2hk}\epsilon_n^{2hM}} \right) = \sum_{k=0}^{M-1} (f_k + f_{k+M}\epsilon_n^{2hM})\,\epsilon_n^{2hk}.$$

Da $\epsilon_n^{2hM} = 1$ und $\epsilon_n^{2hk} = \epsilon_{n-1}^{hk}$ gelten, ergibt sich

$$N\beta_{2h} = \sum_{k=0}^{M-1} (f_k + f_{k+M})\,\epsilon_{n-1}^{hk}.$$

Für ungerade j erhält man analog

$$N\beta_{2h+1} = \sum_{k=0}^{M-1} \left(f_k \epsilon_n^{(2h+1)k} + f_{k+M}\epsilon_n^{(2h+1)(k+M)} \right) = \sum_{k=0}^{M-1} \epsilon_n^k (f_k + f_{k+M}\epsilon_n^{2hM}\epsilon_n^M)\epsilon_n^{2hk}.$$

Wendet man neben $\epsilon_n^M = -1$ wiederum $\epsilon_n^{2hM} = 1$ und $\epsilon_n^{2hk} = \epsilon_{n-1}^{hk}$ an, so folgt

$$N\beta_{2h+1} = \sum_{k=0}^{M-1} \epsilon_n^k (f_k - f_{k+M})\,\epsilon_n^{2hk}.$$

Definiert man $\tilde{\beta}_h := \beta_{2h}$, $\tilde{\tilde{\beta}}_h := \beta_{2h+1}$ sowie $\tilde{f}_k := f_k + f_{k+M}$, $\tilde{\tilde{f}}_k := \epsilon_n^k (f_k - f_{k+M})$, erhält man zusammengefasst folgende Repräsentation für die β_j:

$$N\tilde{\beta}_h = \sum_{k=0}^{M-1} \tilde{f}_k\, \epsilon_{n-1}^{hk}, \qquad h = 0, \ldots, M,$$

$$N\tilde{\tilde{\beta}}_h = \sum_{k=0}^{M-1} \tilde{\tilde{f}}_k\, \epsilon_{n-1}^{hk}, \qquad h = 0, \ldots, M.$$

Diese Darstellung bietet zwei zentrale Erkenntnisse: Zum Einen benötigt jedes β_j statt der ursprünglich N Multiplikationen und $N-1$ Additionen nurmehr $M=N/2$ Multiplikationen und $M-1$ Additionen. Zum Anderen besitzen die neuen Koeffizienten eine zu Gleichung (1) identische Struktur und können daher wiederum in der beschriebenen Art und Weise in jeweils zwei Summen der Länge $M/2$ zerlegt werden. Zwar gehen die Zerlegungen jeweils mit der zusätzlichen Berechnung der Werte $\tilde{f}_k$ und $\tilde{\tilde{f}}_k$ einher und bedingen daher weitere Multiplikationen und Additionen. Jedoch zeigt der folgende Abschnitt, dass in Summe eine beträchtliche Reduktion des Aufwands von $\Theta(N^2)$ zu $\Theta(N\log_2(\mathrm{N})) = \Theta(Nn)$ erzielt wird.

Für $m \le n$ definieren wir die Größen $M^{(m-1)} := 2^{m-1}$ und $R^{(m-1)} := 2^{n-m}$ und die $2R^{(m-1)} \times M^{(m-1)}$-Matrix $f_{l,k}^{(m-1)}$, $l = 0, \ldots, 2R^{(m-1)}-1$, $k = 0, \ldots, M^{(m-1)}-1$. Führt man die Zerlegung der Summen konsequent fort, ergibt sich für die Initialisierung $f_{0,k}^{(n)} := f_k$, $k = 0, \ldots, \mathrm{N}-1$, und $m = n, n-1 \ldots, 1$ die Rekursion (siehe [Freund, Hoppe (2007), S. 82])

$$f_{r,k}^{(m-1)} = f_{r,k}^{(m)} + f_{r,k+M^{(m-1)}}^{(m)},$$

$$f_{r+R^{(m-1)},k}^{(m-1)} = \left(f_{r,k}^{(m)} - f_{r,k+M^{(m-1)}}^{(m)} \right) \epsilon_m^k,$$

mit $\mathrm{r} = 0, \ldots, \mathrm{R}^{(m-1)}-1$ und $k = 0, \ldots, M^{(m-1)}-1$. Die finalen Werte für die β_j erhält man schließlich als

$$\beta_j = \frac{1}{N} f_{j,0}^{(0)}, \qquad \mathrm{j} = 0, \ldots, N-1.$$

Zu beobachten ist, dass für $m = n, n-1 \ldots, 1$ in jedem Schleifendurchlauf zusätzliche Operationen zur Berechnung der $f_{*,*}^{(m-1)}$ aus den gegebenen Werten $f_{*,*}^{(m)}$ anfallen. Bei genauerer Betrachtung handelt es sich bei der Auswertung der $f_{r,k}^{(m-1)}$, $\mathrm{r} = 0, \ldots, \mathrm{R}^{(m-1)}-1$, $k = 0, \ldots, M^{(m-1)}-1$, um $R^{(m-1)}M^{(m-1)}$ Additionen, bei der Auswertung aller $f_{r+R^{(m-1)},k}^{(m-1)}$ um weitere $R^{(m-1)}M^{(m-1)}$ Additionen (bzw. Subtraktionen) und $R^{(m-1)}M^{(m-1)}$ Multiplikationen. In Gänze fallen damit für fixes m Rechenoperationen in der Größenordnung $\Theta(2R^{(m-1)}M^{(m-1)}) = \Theta(2^n) = \Theta(N)$ an. Da die Zerlegung für alle $m = n, n-1, \ldots, 1$ durchgeführt wird, beträgt der Gesamtaufwand für die FFT wie eingangs behauptet $\Theta(Nn) = \Theta(N\log_2 N)$. Die Implementierung der FFT gestaltet sich einfach, wenn man insgesamt zwei Arrays verwendet, die zur Speicherung der $f_{*,*}^{(m)}$-Werte einerseits und $f_{*,*}^{(m-1)}$-Werte andererseits herangezogen werden. Speicherschonendere Implementierungen sind möglich, liegen jedoch nicht im Fokus des vorliegenden Artikels.