Modellierung von Optionspreisen mit PROC FCMP [1]

Stephan Meyer
Institut für Informationswirtschaft und Marketing (IISM)
Karlsruher Institut für Technologie
Englerstr. 14
D-76131 Karlsruhe
Stephan.meyer@kit.edu

Felix Fritz
Institut für Informationswirtschaft und Marketing (IISM)
Karlsruher Institut für Technologie
Englerstr. 14
D-76131 Karlsruhe
Felix.Fritz2@kit.edu

Christof Weinhardt
Institut für Informationswirtschaft und Marketing (IISM)
Karlsruher Institut für Technologie
Englerstr. 14
D-76131 Karlsruhe
weinhardt@kit.edu

Zusammenfassung

Finanzmärkte produzieren heutzutage jede Millisekunde unvorstellbare Mengen an Daten wie Kursbeobachtungen (Quotes), Nachrichten, et cetera. Die Verarbeitung dieser Daten benötigt enorme Kapazitäten und deshalb hoch performante Algorithmen für effiziente Analysen. In diesem Artikel zeigen wir, wie SAS eingesetzt werden kann, um Methoden zur Optionspreismodellierung mithilfe von PROC FCMP zu implementieren. Wir nutzen eine ad-hoc Erweiterung des gängigen Black-Scholes Optionspreismodell zur Bewertung beliebiger Derivate. Wir erhalten den dafür notwendigen Voltilitätsparameter implizit aus gehandelten EUREX Optionen. Dabei verdeutlichen wir, welche SAS Funktionen nötig sind und welche Probleme auftreten können.

Schlüsselwörter: PROC FCMP, Finanzmarktdaten, Optionspreisbestimmung, Black-Scholes Modell, Implizite Volatilität

1 Einführung

Die Modellierung von Optionspreisen erfordert die Auswahl eines mathematischen Modells und die anschließende Verarbeitung aller benötigten Eingabeparameter. Aufgrund der Priorisierung der Performance gilt in praktischen Belangen das Black-Scholes Optionspreismodell als der Standard in der Finanzwirtschaft. Komplexere Modelle für die Bestimmung des Optionspreises erfordern Laufzeiten, die einen Einsatz außerhalb des theoretischen Bereiches problematisch machen.
Für das Black-Scholes Modell wird als Input unter anderem ein Volatilitätsparamter des zugrundeliegenden Instruments (z.B. Indizes, Aktien, Commodities) benötigt. Wir nut-

[1] Wir danken der Börse Stuttgart für die finanzielle Unterstützung.

zen eine Erweiterung des Black-Scholes Modells – das sogenannte "Practioners Black-Scholes Modell" (PBS) – bei welchem der Volatilitätsparameter über eine implizite Volatilitätsfunktion bestimmt wird. Diese Funktion wird fortlaufend anhand von, an der EUREX[2] gehandelten, Optionsscheinen neu kalibriert.

2 Umsetzung in SAS

Im folgenden Kapitel zeigen wir ausschnittsweise wie das Black-Scholes Modell zur Bestimmung von Optionspreisen in SAS erfolgreich implementiert werden kann und welche Probleme dabei auftauchen können. Zuerst gehen wir kurz auf die benötigte Prozedur *PROC FCMP* ein, um im Anschluss detaillierter den eigentlichen Programmcode zu beschreiben.

2.1 PROC FCMP

Die „Function Compiler Procedure" (*FCMP*) ermöglicht es eigene Funktionen zu erstellen und abzulegen, um innerhalb von anderen SAS Methoden darauf zuzugreifen. So ist es beispielsweise möglich in einem Data Step Variablen an eine *FCMP* Funktion zu übergeben und das Ergebnis in einer neuen Variablen zu speichern, oder aber bereits existierende Variablen zu verändern. Dabei ist zu bemerken, dass die Namen, der an die FCMP Funktion übergebenen Variablen, nicht zwingend mit denen bei der Definition der Funktion verwendeten Variablennamen übereinstimmen müssen, jedoch die Reihenfolge der Argumente.

FCMP bietet bereits eine umfangreiche Bibliothek an nutzbaren Funktionen. Zu diesen gehört auch die *SOLVE* Funktion, die eine gegebene Funktion nach einem Argument auflösen kann. Die Syntax sieht dabei wie folgt aus (vgl. [2]):

```
variable = solve( "function_name", options_array, expected_value,
                  arg1, arg2, … , argN );
```

Hierbei spezifiziert *options_array* die Einstellungen des Iterationsverfahrens: den Startwert, ein absolutes Fehlerkriterium, ein relatives Fehlerkriterium, die maximale Anzahl an Iterationen, sowie einen Eintrag für den (zurückgegebenen) „solve"-Status. Es sei vorab gesagt, dass der iterative Lösungsprozess der *SOLVE* Funktion es nahelegt in Spezialfällen andere Lösungen (z.B. ein Sekantenverfahren) selbst zu implementieren, um eventuell bessere Ergebnisse zu erzielen.

Außerdem bietet *PROC FCMP* die Möglichkeit Makros innerhalb eines Data Steps auszuführen indem man innerhalb einer Funktion oder Subroutine ein Makro über den Ausdruck

```
rc0=run_macro('macroname', arg1, arg2 ,… );
```

[2] Die European Exchange (EUREX) ist eine der führenden Terminbörsen für Finanzderivate.

aufruft. Hierbei müssen die Argumente *FCMP* Variablen sein, welche dem aufgerufenen Makro als Makrovariablen mit demselben Namen übergeben werden. Die Variable *rc0* ist eine Kontrollvariable, die den Wert 1 oder 0 annimmt, wenn das Makro korrekt ausgeführt wurde oder nicht. Ein ausführlicherer Einblick in die Nutzung von *PROC FCMP* ist beispielsweise in [3] gegeben.

2.2 Schätzung der impliziten Volatilität

Um möglichst effizient die implizite Volatilität eines beliebigen Instruments zu bestimmen, implementieren wir alle erforderlichen Funktionen mithilfe von *PROC FCMP,* um diese direkt aus einem Data Step heraus benutzen zu können. Die Schätzung der impliziten Volatilität aus EUREX Optionen erfordert die Auflösung der Black-Scholes Formel nach der (impliziten) Volatilität. Dabei gilt zu beachten, dass die Preise, der an der EUREX gehandelten Optionen, keine Modellpreise darstellen, sondern Preise die u.a. Erwartungen der Marktteilnehmer widerspiegeln und somit in der Regel von Modellpreisen abweichen. Insbesondere ist üblicherweise der sogenannte „Volatility-Smile“ zu beobachten, d.h. die impliziten Volatilitäten sind nicht für alle Strike-Preise gleich, wie es das Black-Scholes Modell implizieren würde, sondern sie sind parabelförmig (im Strike-Preis).

SAS bietet bereits vorgefertigte Funktionen für die klassischen Black-Scholes Formeln für Calls (*blkshclprc*) und Puts (*blkshptprc*). Wir nutzen die integrierte *SOLVE* Funktion, um die Formel nach dem gewünschten Parameter aufzulösen. In der SAS Dokumentation ([1]) ist der klassische Fall der Berechnung der impliziten Volatilität als Beispiel[3] formuliert. Aufgrund der Vorgehensweise der *SOLVE* Funktion ist der Algorithmus nur bedingt anwendbar, da der erfolgreiche Abschluss der Routine stark von den gewählten Eingabeparametern abhängt und zum anderen nicht testet, ob diese Parameter überhaupt zu einer gültigen Lösung führen können. Im Falle von Call-Optionen gilt zum Beispiel, dass je größer der Preis des Basiswertes relativ zum Strike-Preis ist, desto sensitiver ist die Bestimmung der impliziten Volatilität gegenüber Abweichungen der realen Daten zum Black-Scholes-Modell. In unseren Berechnungen hat die Implementierung des Sekantenverfahrens als Lösungsalgorithmus zu besseren Ergebnissen geführt, da hier Randwerte angegeben werden können.
Der folgende Auszug aus dem Code zeigt die Implementierung von Funktionen zur Preisbestimmung der impliziten Volatilität von Call- und Put-Optionen nach dem Black-Scholes Modell ohne Dividenden (wie zum Beispiel bei Optionen auf den DAX). Die Funktionen *blkschC* und *blkschP* berechnen den Optionspreis unter Berücksichtigung aller Eingabeparameter: Strike-Preis (*strike*), Restlaufzeit (*t2m*), Preis des Basiswertes (*uly*), risikoloser Zins (*r*) und die Volatilität (*volty*).

```
* Implizite Black-Scholes Volatilität;
proc fcmp outlib= sasuser.funcs.bs;
```

[3] Das Beispiel der Dokumentation kann aufgrund des in der Vergangenheit liegenden Verfallstages nicht funktionieren.

```
*Call;
 function blkschC(strike, t2m, uly, r, volty);
          v=abs(volty);
       return(blkshclprc(strike, t2m, uly, r, v));
    endsub;

* Put;
 function blkschP(strike, t2m, uly, r, volty);
      v=abs(volty);
      return(blkshptprc(strike, t2m, uly, r, v));
    endsub;

* Löse Call-Gleichung nach Volatilität auf;
function impliedvolaC(opt_price, strike, t2m, uly, r);
     array opts[5] initial abconv relconv maxiter solvstatus
                 ( 0.2  .0001 1.0e-6 1000 );
     x=solve( "blkschC",opts,opt_price,strike,t2m,uly,r,.);
     return(x);
endsub;

* Löse Put-Gleichung nach Volatilität auf;
function impliedvolaP(opt_price, strike, t2m, uly, r);
     array opts[5] initial abconv relconv maxiter solvstatus
                 ( 0.2  .0001 1.0e-6 1000 );
     x=solve( "blkschP", opts, opt_price, strike, t2m, uly, r, .);
     return(x);
     endsub;
quit;
```

Alle Funktionen werden in der Bibliothek *sasuser.funcs* abgespeichert. Die beiden Funktionen *impliedvolaC(...)* und *impliedvolaP(...)* berechnen die implizite Volatilität aus den vorhandenen Optionspreisen. Die Funktionen müssen generell neu definiert werden, da die bereits in SAS enthaltenen Funktionen (*blkshclprc* bzw. *blkshptprc*) nicht direkt an die *SOLVE* Funktion übergeben werden kann. Zusätzlich vermeiden wir durch die Verwendung des Betrags der Volatilität (*v*), dass die *SOLVE* Funktion negative Werte in die Black-Scholes Formeln einsetzt, was insgesamt zu weniger fehlerhaften Iterationsvorgängen führt.
Alternativ haben wir ein Sekantenverfahren implementiert, welches sowohl das letztere Problem vermeidet, als auch im Durchschnitt bessere Ergebnisse zur Auflösung der Gleichung erreicht als die integrierte *SOLVE* Funktion. Hierbei bezeichnen *start1* und *start2* die Grenzen des Startintervalls, welche gleichzeitig die Extremwerte für die implizite Volatilität darstellen.

```
proc fcmp outlib=sasuser.funcs.sekanten;

function impliedvolaC_sekanten(uly,strike,t2m,r,opt_price,
     start1,start2,maxerror,maxiter);

     implvola2=start2;
     implvola1=start1;
     d1=blkshclprc(strike,t2m,uly,r,implvola1)- opt_price;
     d2=blkshclprc(strike,t2m,uly,r,implvola2)- opt_price;
     i=1;
     relerror=maxerror+1;

     do while ( (i LE maxiter) and (relerror > maxerror) );

          implvola= implvola2- (implvola2- implvola1)/( d2 -
               d1 ) * d2;
          if implvola GE start2 then implvola=start2;
          if implvola LE start1 then implvola=start1;
          fct_value = blkshclprc(strike,t2m,uly,r,implvola)
                    -  opt_price;

          relerror=abs(fct_value);
          implvola1=implvola2;
          implvola2=implvola;
          d1=d2;
          d2=blkshclprc(strike, t2m, uly,r,implvola2)-
                         opt_price;
          i=i+1;
     end;

     return(implvola);
     endsub;
```

Verwendet man alternativ die Funktion *blkshptprc* erhält man ein äquivalentes Verfahren für Put-Optionen.
Beide Verfahren liefern nun implizite Volatilitäten mit verschiedenen Strike-Preisen und verschiedenen Restlaufzeiten. Um den oben genannten Modellabweichungen gerecht zu werden, gibt es verschiedene Möglichkeiten, um die empirischen impliziten Volatilitäten zeitlich und/oder in den Strike-Preisen zu aggregieren.

Prinzipiell bieten sich zwei grundlegende Ansätze an – ein globaler Ansatz, bei dem der Volatilitätsparameter aus historischen Daten[4] einmalig bestimmt wird oder ein dynamischer Ansatz, bei welchem die (impliziten) Volatilitäten zu jeder Beobachtung neu berechnet werden, d.h. das Modell wird fortlaufend neu kalibriert. Dabei wird zu jedem Zeitpunkt eine parametrische Funktion anhand beobachteter Optionspreise geschätzt. Dieser Ansatz ist als „Practitioners Black-Scholes" Modell[5] (PBS) bekannt und wird ausführlich von [4] diskutiert.

Wir implementieren das PBS Modell in SAS folgendermaßen: Ausgehend von den oben bestimmten impliziten Volatilitäten, nutzen wir, für die Bewertung eines beliebigen Derivats, diejenigen (gehandelten) Optionen, bzw. deren impliziten Volatilitäten, die die nächst längere bzw. kürzere Restlaufzeit besitzen. Anhand dieser schätzen wir je eine Funktion (zweiter Ordnung) des Basispreises und interpolieren linear zwischen beiden in Abhängigkeit der Restlaufzeit des zu bewertenden Derivats.

Das folgende Makro *getvola* geht von einer Tabelle *options* aus, welche die bereits berechneten impliziten Volatilitäten sowie die jeweiligen Optionsmerkmale enthält und führt für einen beliebigen Zeitpunkt (*datetime*) und für eine beliebige Restlaufzeit (*maturity*) die beschriebene Prozedur durch. Die Parameterschätzungen für die drei Funktionskoeffizienten werden in einer Tabelle *dummy3* abgespeichert.

```
%macro getvola;
%let date=%sysfunc(dequote(&datetime));
%let maturity=%sysfunc(dequote(&maturity));

*Finde Optionen zum jeweiligen Zeitpunkt mit ähnlicher Restlaufzeit;
proc sql;
     create table dummy as
select * , t2m-&maturity as timediff,
               strike*strike as strikesq
               from options
                    where datetime =&date
                         order by timediff;

create table dummy2 as
     select * from dummy
          where (timediff in (select min(timediff) from
                         dummy where timediff GE 0)
               or timediff in (select max(timediff) from   dummy
                              where timediff<0) );
```

[4] Üblicherweise verwendet man die vergangene (realisierte) Volatilität des Basiswertes oder Durchschnitte von impliziten Volatilitäten aus beobachteten Optionspreisen.

[5] Alternative Bezeichnungen (je nach Autor) sind ‚ad-hoc Black-Scholes' oder ‚Implied Volatility Function Model'.

```
quit;

* Quadratische Regression;
proc reg data=dummy2 outest=dummy3 noprint;
          model implvola = strike  strikesq;
               by timediff;
run;
%mend getvola;
```

Wir verdeutlichen den Makroaufruf via *PROC FCMP* innerhalb eines Data Steps anhand der Bewertung von sogenannten exotischen Optionen[6] – genauer gesagt „Down-and-In Puts“. Diese besitzen als zusätzlichen Parameter ein Barrierelevel (*H*), welches mindestens erreicht werden muss, damit der „Down-and-In-Put“ eine Auszahlung gewährt. Die zugehörige Bewertungsformel (nach Black-Scholes) ist nicht vorimplementiert und wird deshalb im ersten Teil des *PROC FCMP* Aufrufs definiert (*daiput*). Die zu bewertenden Derivate inklusive ihrer Merkmale (Strike-Preis, Underlying, usw.) sollen sich in der Tabelle *pricing* befinden.

```
proc fcmp outlib=sasuser.funcs.vola;

* Bezeichnungen: Strike-Preis(K), Underlying-Preis (S), Restlaufzeit
(t2m), Barriere (H), Risikoloser Zins (r), Volatilität(volty);

function daiput(S,K,H,t2m,r,volty);

     z =(r +( volty**2 /2))/ volty**2;
     y =log(H**2/(S*K))/( volty *sqrt(t2m))+ z* volty *sqrt(t2m);
     x1= log(S/H)/( volty *sqrt(t2m))+ z* volty *sqrt(t2m);
     y1= log(H/S)/( volty *sqrt(t2m))+ z* volty *sqrt(t2m);

     diput= -S*CDF("Normal",-x1)
           + K* exp(-r*t2m)*CDF("Normal",-x1+ volty*sqrt(t2m))
           + S* H/S)*(2*z)*( CDF("Normal",y)-CDF("Normal",y1) ) -
          K*exp(-r*t2m)*(H/S)**(2*z - 2)*
               ( CDF("Normal",y- volty *sqrt(t2m))
                    - CDF("Normal",y1- volty *sqrt(t2m)));
     return(diput);
     endsub;

subroutine obtainVolaParameter(datetime,strike,maturity,implvola);
     outargs implvola;
```

[6] Für einen detaillierten Überblick zu (exotischen) Optionen und deren Bewertungsformeln, verweisen wir auf [5].

```
     rc0=run_macro('getvola', datetime, maturity);
     array t[2] /nosymbols;
     array a[2] /nosymbols;
     array b1[2] /nosymbols;
     array b2[2] /nosymbols;
     rc1=read_array("work.dummy3",t,"timediff");
     rc2=read_array("work.dummy3",a,"Intercept");
     rc3=read_array("work.dummy3",b1,"Strike_price");
     rc4=read_array("work.dummy3",b2,"strikesq");
     s1=a[1]+ b1[1]*strike +b2[1] * strike**2;
     s2=a[2]+ b1[2]*strike +b2[2] * strike**2;
     implVola=(abs(t[2])*s1+abs(t[1])*s2)/(abs(t[1])+abs(t[2]));

     endsub;
quit;

 * Variable implvola muss initialisiert sein;
data pricing;
     set pricing;
     CALL obtainVolaParameter(datetime,strike,maturity,implvola);

     Price= daiput(underlying,strike,barrier,maturity,r,implvola);
run;
```

Abschließend sei erwähnt, dass die benötigte Rechenzeit massiv reduziert werden kann, wenn die Optionsdaten zur Bestimmung der impliziten Volatilität in einzelne, zum Beispiel nach Datum gruppierte Tabellen aufgeteilt werden. Zudem kann man die Funktion der impliziten Volatilität in Abhängigkeit der Restlaufzeit definieren und deren Schätzung über alle Restlaufzeiten durchführen. Dadurch wird der Zeitaufwand für die Erstellung der Tabelle *dummy* eingespart. Allerdings zeigt unsere empirische Analyse [1], dass erstere Methode für Barriere-Optionen leicht bessere Ergebnisse liefert.

3 Fazit

Zusammenfassend lässt sich festhalten, dass SAS im Gegensatz zu Programmsuiten wie Matlab oder R den großen Vorteil mit sich bringt, extrem große Datenmengen verarbeiten zu können. Die Preisbestimmung von Optionen direkt in SAS zu integrieren ist daher eine sehr performante Möglichkeit schnell qualitativ hochwertige Ergebnisse zu erzielen. Speziell wenn die Verarbeitung und Bewertung von Intraday-Daten (z.B. Derivate auf Basis von minütlichen Beobachtungen) über einen längeren Zeitraum erforderlich ist, ist die vorgestellte Methode über *SAS BASE* und *PROC FCMP* eine effiziente Umsetzung des Black-Scholes Modell.

Literatur

[1] Fritz, Felix, Stephan Meyer und Christof Weinhardt: Pricing of Bank-issued Investment Products – Premium Shifts and Investor Wealth. Working paper, 2012.

[2] SAS Institute Inc., Cary, NC, USA: The FCMP Procedure. SAS Publishing, 2003. http://support.sas.com/documentation/onlinedoc/base/91/fcmp.pdf [14.03.2013]

[3] Secosky, Jason: User-Written DATA Step Functions. Proceedings of the NESUG, 2007. http://www.nesug.org/Proceedings/nesug07/bb/bb14.pdf [14.03.2013]

[4] Berkowitz J.: On Justification for the ad hoc Black-Scholes Method of Option, Studies in Nonlinear Dynamics & Econometrics, 14(1), 2009

[5] Hull, J. C.: Options, Futures and Other Derivatives, 6th edition, Prentice Hall, 2005