CHALMERS | GÖTEBORGS UNIVERSITET

# Numerisk prissättning av exotiska optioner

## En undersökning av asiatiska, barriär- och lookback-optioner med Monte Carlo- och Crank-Nicolson-metoden

*Examensarbete för kandidatexamen i matematik vid Göteborgs universitet*
*Kandidatarbete inom civilingenjörsutbildningen vid Chalmers*

Kasper Bågmark
Emil Carlsson
Victor Ebberstein
Nadja Grochevaia
Carl Söderpalm

Institutionen för matematiska vetenskaper
Chalmers tekniska högskola
Göteborgs universitet
Göteborg 2017

# Numerisk prissättning av exotiska optioner

En undersökning av asiatiska, barriär- och lookback-optioner med Monte Carlo- och Crank-Nicolson-metoden

*Examensarbete för kandidatexamen i tillämpad matematik inom matematikprogrammet vid Göteborgs universitet*

Kasper Bågmark

*Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk matematik vid Chalmers*

Emil Carlsson Victor Ebberstein
Nadja Grochevaia Carl Söderpalm

Handledare: Simone Calogero
Examinator: Marina Axelson-Fisk Maria Roginskaya

Institutionen för matematiska vetenskaper
Chalmers tekniska högskola
Göteborgs universitet
Göteborg 2017

Institutionen för matematiska vetenskaper
Göteborg 2017

# Populärvetenskaplig presentation

I dagens utvecklade samhälle anses optioner vara en relativt ny investeringsform i jämförelse med mer traditionella alternativ såsom aktier. Men till skillnad från den allmänna uppfattningen har optioner i själva verket använts under en längre tid än vad många tror. En option är ett kontrakt som är direkt knutet till en underliggande tillgång, vilket kan bestå av till exempel en aktie eller en valuta, med mera. Detta kontrakt ger ägaren rätten, men inte skyldigheten, att köpa eller sälja den underliggande tillgången till ett förutbestämt pris.

Det tidigaste fyndet av vad som efterliknar dagens optioner finns beskrivet i boken *Politik* av Aristoteles från 400-talet f.Kr. I denna bok hittar man berättelsen om filosofen Thales av Miletus och hur han skapade en förmögenhet genom att köpa upp rättigheter att bruka olivpressar precis innan skördetid, med syfte att sälja av dessa till ett högre pris när det väl nalkades skörd. Även om termen option inte hade myntats vid den här tiden, hade Thales skapat den första köpoptionen.

En annan anmärkningsvärd händelse var den så kallade Tulpanmanin i Nederländerna i mitten på 1600-talet. Vid den här tiden fick tulpanen ett ordentligt uppsving i popularitet bland så väl lokalbefolkning som i övriga delar av Europa, vilket i sin tur gav upphov till en omåttlig prisuppgång på tulpanlökar. Tulpangrossisterna började då teckna köpoptioner som gav dem rätten att i framtiden köpa tulpanlökar till ett lägre fast pris. I takt med de skenande priserna började en andrahandsmarknad för tulpankontrakt att ta form, vilket gjorde det möjligt för allmänheten att spekulera i tulpanmarknaden. Detta kulminerade till slut i en djup nationell lågkonjunktur i Nederländerna, där en betydande andel av befolkningen försattes i personlig konkurs.

Optionskontrakt vid den här tiden saknade den legitimitet de har idag, då optionsmarknaden mer eller mindre var oreglerad. Genom åren har optioner ständigt mötts av en enorm skepticism och deras rykte har minst sagt varit bristfälligt. Denna inställning till optioner kom att ändras när *Chicago Board of Options Exchange* (CBOE) inrättades på 1970-talet. För första gången fanns det ett regelverk för standardiserade optioner samt en rättvis marknad där dessa kunde handlas. Dock fanns det fortfarande en del ovisshet kring optionernas värde och vad som kunde anses vara god valuta för pengar.

Till följd av denna ovisshet har ett flertal matematiska modeller utvecklats för prissättning av optioner, där binomialmodellen och Black-Scholes-modellen var bland de första. Binomialmodellen utgår från premissen att priset för den underliggande tillgången endast kan röra sig upp eller ner från en tidpunkt till en annan. Black-Scholes-modellen är en mer generell version av binomialmodellen och behandlar oändligt små avstånd mellan tidpunkter. Den skapades med ändamålet att beräkna teoretiska priser för standardiserade optioner med kända lösendatum och har lagt grunden för modern optionsteori. Detta har banat väg för vidare forskning och prissättning av mer komplexa optioner, så kallade exotiska optioner. Dessa är oftast mer komplicerade kontrakt med speciella egenskaper eller lösenvärdesfunktioner som är anpassade för att möta speciella behov på marknaden. Till exempel kan de vara beroende av den underliggande tillgångens värde i fler tidspunkter än endast den sista.

Black-Scholes-modellen gav upphov till en ekvation som tar hänsyn till samtliga specifika variabler som används vid prissättning av optioner. Denna ekvation löses lämpligen med hjälp av en dator och en algoritm som successivt beräknar optionens pris. En sådan algoritm kan till exempel vara Crank-Nicolson-metoden, vilken kategoriseras som en så kallad finit differensmetod. Denna metod har visat sig fungera väl för dessa ändamål och andra problem med enkla randvillkor. Med randvillkor menas speciella värden som ekvationen måste uppfylla vid ändarna av dess variablers intervall. Andra möjliga metoder att använda är så kallade finita elementmetoder. Dessa är lite mer sofistikerade och lämpar sig egentligen bättre för mer komplicerade problem. Ytterligare en prissättningsmetod, vilken används dagligen vid prissättningen av dessa optioner, är Monte Carlo-metoden. Denna har dock visat sig behöva korrigeras för att komma till bukt med ett fel som uppstår vid standardimplementeringen.

Optioner har blivit mycket populära spekulationsverktyg. Prestandakraven för en noggrann prissättning är vanligtvis mycket hög om optionens struktur är väldigt komplex. Många olika metoder har tillämpats för att angripa detta problem och vidare forskning inom området är nödvändig. Den här presentationen har gett en kort inblick i vad optioner är och hur användbar matematiken kan vara för att prissätta dessa rättvist. Optioner har många fördelar, tillsammans med flertalet användningsområden, vare sig det handlar om olivpressar, tulpaner eller aktier.

**Sammanfattning**

Denna rapport syftar till att studera asiatiska, lookback- och barräroptioner av europeiska slag i tidsintervallet $[0, T]$, där $T$ är lösendagen. Målsättningen är att undersöka numeriska metoder som tillämpas vid prissättning givet Black-Scholes-modellen. Detta görs i både `C++` och MATLAB med avsikt att jämföra precision och hastighet mellan de två programspråken. Dessutom undersöks diverse känslighetsmått, så kallade Greeks.

De numeriska metoder som studeras för prissättning av samtliga optioner är Monte Carlo-metoden och Crank-Nicolson-metoden, en finit differensmetod. Vid implementering av Monte Carlo-metoden upprepas ett obundet slumpmässigt urval $N$ gånger för att generera ett approximativt pris. Detta genom indelning av intervallet $[0, T]$ i $n$ delintervall och simulering av en diskret utveckling för värdet $S(t)$ av den underliggande tillgången. Crank-Nicolson-metoden tillämpas genom invertering av tidsvariabeln i Black-Scholes partiella differentialekvation, där rumsderivatorna är centrerade, medan tidsderivatan uppskattas både framåt och bakåt. Implementering av denna medför en indelning av både tidsintervallet $[0, T]$ och rumsintervallet $[0, x_{max}]$ i $n$ respektive $m$ delintervall.

Undersökningen av utvalda Greeks visar på att asiatiska optioner är mindre känsliga för volatilitet än vad lookback- och barriäroptioner är. Dessutom konstateras att Crank-Nicolson-metoden är överlägsen Monte Carlo-metoden för prissättning av samtliga studerade optioner. Detta beror bland annat på ett konvergensfel som uppkommer för Monte Carlo-metoden då lookback- och barriäroptionerna prissätts, vilket gör metoden oerhört tidskrävande. Slutligen fastslås att `C++` är språket att föredra för en snabb och relativt noggrann approximation.

**Abstract**

This paper examines Asian, lookback and barrier options of European style on the time interval $[0, T]$, where $T$ is the time of maturity. The purpose is to investigate numerical methods to compute their price within the Black-Scholes model. This is carried out in both C++ and MATLAB with the objective of comparing the computational performance of the two programming languages. Moreover, various sensitivity quantities, the so called Greeks, are investigated.

The numerical pricing methods selected are the Monte Carlo method and the Crank-Nicolson finite difference method. Implementation of the Monte Carlo method consists of iterating an unbounded random sample $N$ times in order to generate an approximate price. This is achieved through partitioning the interval $[0, T]$ into $n$ subintervals and simulating a discrete path for the value $S(t)$ of the underlying asset. The Crank-Nicolson method is applied through inverting the time variable of the Black-Scholes partial differential equation, where the space derivatives are centered and the time derivatives are estimated in a forward-backward manner. Implementation of the method implies partitioning both the time interval $[0, T]$ and the space interval $[0, x_{max}]$ into $n$ and $m$ subintervals, respectively.

Examination of the selected Greeks show that Asian options are less sensitive to volatility than lookback and barrier options are. Furthermore, it is concluded that the Crank-Nicolson method is superior to the Monte Carlo method for the pricing of all of the examined options. One of the reasons for this is a convergence problem that arises for the lookback and barrier options, which causes the Monte Carlo method to be very time-consuming. Lastly, C++ is shown to be the language of choice for a fast and relatively accurate approximation.

# Innehåll

# Förord

Denna rapport har producerats för Matematiska Vetenskaper, Chalmers Tekniska Högskola, under våren 2017 som en slutprodukt av ett kandidatprojekt. Det har innefattat fyra studenter från Civilingenjörsprogrammet Teknisk Matematik på Chalmers och en student från Matematikprogrammet på Göteborgs Universitet.

Under arbetets gång har vi haft stor hjälp och skulle vilja tacka alla inblandade. Ett extra tack går till vår handledare Simone Calogero. Han har varit utomordentligt engagerade och följt projektet varje steg på vägen. Det var dessutom han som introducerade de flesta av oss för optionsteorin, vilket för flera av oss ledde till valet av detta projekt. Tack för din insats och ditt stora engagemang!

Nedan kommer huvudförfattaren av respektive avsnitt i rapporten att presenteras. Alla projektets medlemmar har korrekturläst, reviderat och kommit med idéer på samtliga avsnitt, vilket innebär att rapporten bör ses som en gruppinsats snarare än en sammanställning av individuella bidrag. Underavsnitt presenteras inte då författaren av huvudsavsnittet anses vara ansvarig även för dessa.

Populärvetenskaplig presentation - Nadja Grochevaia
Sammanfattning/Abstract - Nadja Grochevaia
Förord - Victor Eberstein
1. Inledning - Victor Eberstein & Carl Söderpalm
2. Bakgrund - Victor Eberstein & Carl Söderpalm
3. Asiatiska optioner - Victor Eberstein & Carl Söderpalm
4. Lookback-optioner - Emil Carlsson & Nadja Grochevaia
5. Barriäroptioner - Kasper Bågmark
6. Slutsats - Kasper Bågmark & Emil Carlsson

A.1 Carl Söderpalm
A.2 Carl Söderpalm
A.3 Victor Eberstein
A.4 Carl Söderpalm
A.5 Victor Eberstein
B.1 Emil Carlsson & Nadja Grochevaia
B.2 Victor Eberstein & Nadja Grochevaia
B.3 Kasper Bågmark & Emil Carlsson
C - Hela gruppen

Utöver denna sammanställning har en loggbok förts under arbetets gång för såväl gruppen som individen. Individens loggbok innehåller den tid som lagts ned på projektet och vad den lagts på. Gruppens loggbok sammanfattar hur våren fortlöpt och var gruppen har befunnit sig i arbetet vid olika tidpunkter.

Projektet har emellanåt varit en utmaning, men har till största del fungerat väl med stor samarbetsvilja hos samtliga medlemmar. Mycket arbete och bra sammanhållning har fört gruppen framåt även under de jobbiga och krävande perioderna. I det hela är vi tacksamma för projektet, som har varit lärorikt på mer än ett akademiskt plan.

# 1 Inledning

Finansiell matematik är ett av de snabbast växande fälten inom tillämpad matematik. Ett av de mest centrala resultaten är Black-Scholes matematiska modell av en marknad med finansiella derivat, vilken publicerades 1973. För detta tilldelades Myron Scholes och Robert C. Merton Sveriges Riksbanks pris i ekonomisk vetenskap till Alfred Nobels minne 1997. Den tredje upphovsmakaren, Fisher Black, hade dessvärre gått bort vid tillfället. Trots att deras arbete framförallt behandlade de simplaste optionstyperna har det lagt grunden för modern optionsteori. Publikationen bidrog till att handlandet av optioner ökade explosionsartat. Denna nya teori, tillsammans med ett ökat allmänintresse, banade väg för fortsatt forskning och utökad förståelse av mer komplexa optioner.

Black-Scholes-modellen gav upphov till en partiell differentialekvation för en options prisfunktion. Därav följde två naturliga tillvägagångssätt för att erhålla optionspriser: numerisk integration och numeriska lösningsmetoder av partiella differentialekvationer. Det senare använde sig Eduardo Schwartz av, 1977, när han var den förste att applicera finita differensmetoder för optionsprissättning [1]. Phelim P. Boyle introducerade senare samma år Monte Carlo-metoden som ett tredje sätt att prissätta optioner. I hans artikel presenterades endast resultat för europeiska standardoptioner, men han konstaterade även att metoden var applicerbar för andra typer [2]. Tillsammans med binomialmodellen, som publicerades 1979 av Cox, Ross och Rubenstein, tillhör dessa de vanligast förekommande prissättningsmodellerna.

I denna rapport har vi valt att studera de exotiska optionerna: asiatiska, lookback- och barriäroptioner av europeiska slag. Vi kommer studera prissättningen av dessa med olika numeriska metoder samt skillnader i användningsområde. Numeriska metoder behövs vid prissättning av optioner när en exakt prisformel inte finns tillgänglig eller är oanvändbar av någon anledning. Vi har valt att titta på två numeriska metoder: Monte Carlo-metoden och den finita differensmetoden Crank-Nicolson-metoden. Dessa kommer implementeras i två olika programspråk, MATLAB och `C++`, där vi kommer göra jämförelser mellan språken vad gäller precision och hastighet. Crank-Nicolson-metoden är naturlig att beskriva och implementera med hjälp av matriser. För implementeringen i `C++` har vi därför valt att använda oss av biblioteket *Armadillo*, version 7.800.2. Biblioteket är utformat för att på ett smidigt sätt kunna använda sig av matrisoperationer [3].

Rapporten är uppdelad i ett bakgrundsavsnitt, ett avsnitt för vardera option och en slutsats. I bakgrunden tar vi först upp nödvändig teori och notation. Vi introducerar sedan de numeriska metoderna och utvärderar deras tidskomplexiteter. Slutligen diskuterar vi olika *Greeks* för optioner och deras definitioner. Optionsavsnitten inleder vi med korta bakgrunder till när och varför optionen introducerades samt olika definitioner och användningsområden. Därefter diskuterar vi den optionsspecifika implementeringen av de numeriska metoderna. Vi presenterar sedan resultat utifrån dessa metoder och diskuterar diverse företeelser, varpå vi drar ett antal slutsatser. Rapportens slutsats är en sammanfattning av alla delslutsatser i optionsavsnitten. Vi diskuterar valet av programspråk och numerisk metod för olika ändamål, men även intressanta ämnen för framtida studier.

För att kunna jämföra tidsåtgången, dels för implementeringarna av de olika metoderna och dels programspråken sinsemellan, har samtliga beräkningar genomförts på samma dator. Specifikationer för datorn:

| Hårdvara | Mjukvara |
|---|---|
| MacBook Pro 2014 | OSX El Capitan 10.11.4 |
| 2.4 GHz Intel Core i5 | Matlab R2016b |
| 8 GB, 1600 MHz, DDR3 | C++98 |

Dessutom finns all MATLAB-kod, som använts för att generera resultaten som presenteras i kommande avsnitt, tillgänglig i Appendix C.

# 2 Bakgrund

Vi kommer i detta avsnitt repetera nödvändig teori samt presentera notation som återkommer gång på gång i rapporten. Läsaren förväntas vara bekant med teorin sedan tidigare, varför vi inte behandlar denna i detalj. Vi kommer även introducera de numeriska metoderna mer utförligt och beskriva olika variansreducerande verktyg för Monte Carlo-metoden.

## 2.1 Allmänt om optioner

Antag att vi har en underliggande tillgång med ett initialt värde $S(0)$ och att värdet vid tiden $t > 0$ ges av

$$S(t) = S(0)e^{(r-\frac{1}{2}\sigma^2)t+\sigma W(t)}. \tag{2.1}$$

Här är $r$ den riskfria räntan, $\sigma$ tillgångens volatilitet och $W(t)$ en Wiener process i det risk-neutrala sannolikhetsmåttet. Både $r$ och $\sigma$ antas konstanta i tiden. Då har en option, som endast beror på värdet av denna underliggande tillgång på lösendagen (*time of maturity*) $T$, ett lösenvärde (*pay-off*) enligt

$$Y = g(S(T)),$$

där $g$ kallas lösenvärdesfunktionen. Till exempel har den europeiska köpoptionen (*call option*) lösenvärdet $(S(T) - K)_+$, medan säljoptionen (*put option*) har $(K - S(T))_+$. Här fungerar $(\,\cdot\,)_+$ som ett kortare sätt att beteckna $\max\{\,\cdot\,,\,0\}$ och $K$ är optionens lösenpris (*strike price*).

Hädanefter i rapporten kommer vi alltid befinna oss i det risk-neutrala sannolikhetsmåttet, om inget annat anges. Black-Scholes-priset för optionen med lösenvärdet $Y$ är då

$$\Pi_Y(t) = e^{-r(T-t)}\mathbb{E}[Y \mid \mathcal{F}_S(t)], \quad t \in [0, T]. \tag{2.2}$$

Om vi låter $\Pi_Y(t) = v(t, S(t))$ har vi Black-Scholes partiella differentialekvation (PDE) för prisfunktionen $v(t, x)$:

$$\begin{cases} \partial_t v + \frac{1}{2}\sigma^2 x^2 \partial_x^2 v + rx\partial_x v - rv = 0, \quad 0 < t \leq T,\ x > 0 \\ v(T, x) = g(x). \end{cases} \tag{2.3}$$

Med denna kan vi, för vissa typer av optioner, härleda en exakt lösningsformel för $\Pi_Y(t)$. Till exempel har vi formeln för priset av en europeisk köpoption med lösenvärdet $EK = (S(T) - K)_+$ enligt

$$\Pi_{EK}(t) = S(t)\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2), \tag{2.4}$$

där

$$d_2 = \frac{\log\left(\frac{S(t)}{K}\right) + (r - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \quad d_1 = d_2 + \sigma\sqrt{T-t}$$

och $\Phi(\,\cdot\,)$ är den kumulativa fördelningsfunktionen för en standardiserad normalfördelning.

Även prisformeln för den europeiska säljoptionen kan härledas från Black-Scholes PDE. Givet identiska parametrar leder denna och (2.4) till sälj-köp-pariteten (*put-call-parity*) för europeiska optioner:

$$\Pi_{EK}(t) - \Pi_{ES}(t) = S(t) - Ke^{-r(T-t)}.$$

Pariteten implicerar i praktiken att värdet av en sälj- eller köpoption direkt kan härledas utifrån sin motpart. Om relationen inte skulle uppfyllas innebär det att en arbitrage möjlighet existerar.

Standardoptioner (*vanilla options*) är exempel på optioner som endast beror på $S(T)$. I denna rapport studerar vi tre typer av exotiska optioner (*exotic options*) vars lösenvärden beror på alla tillgångens värden under tidsintervallet $[0, T]$. För en exotisk options prisfunktion finns sällan en användbar exakt formel och i vissa fall finns det ingen alls, till exempel när volatiliteten är en stokastisk process. Därför behöver numeriska metoder appliceras för att beräkna priset.

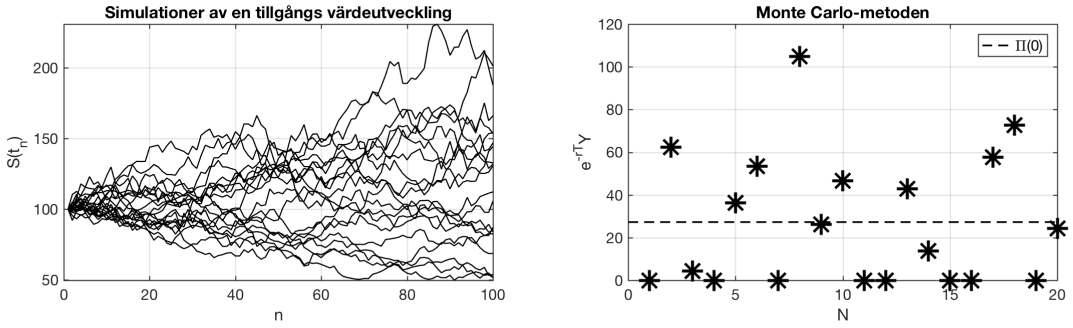
## 2.2 Numeriska metoder för prissättning av optioner

För att prissätta optioner kommer i huvudsak två metoder, av olika karaktär, att undersökas och användas. Dessa är Crank-Nicolson- och Monte Carlo-metoden. Här presenteras det som gäller genomgående för samtliga optioner, medan mer specifika egenskaper presenteras i vardera optionsavsnitt. I denna rapport studeras bara konstanta volatiliteter, men de numeriska metoderna fungerar även om man låter $\sigma$ vara en stokastisk process. Alltså kan samma implementeringar som presenteras i Appendix C användas med en slumpgenerering av $\sigma$.

### Monte Carlo-metoden

Den första metoden att användas är Monte Carlo-metoden. Metoden bygger på ett upprepat obundet slumpmässigt urval för att erhålla numeriska resultat.

För prissättning av optioner används metoden genom att upprepade gånger simulera en möjlig värdeutveckling hos den underliggande tillgången $S(t)$. Detta görs genom att dela in intervallet $[0, T]$ i $n$ stycken delintervall och simulera ett framtida värde genom (2.1) och beräkna lösenvärdet för det simulerade värdet. Ett approximativt pris beräknas genom (2.2), där väntevärdet uppskattas med det aritmetiska medelvärdet av $N$ simulerade lösenvärden. I figur 1 presenteras en visualisering av metoden.




Figur 1: Till vänster har vi värdeutvecklingar och till höger lösenvärdet (med skalfaktor) för europeiska standard köpoptioner givet dessa utvecklingar samt priset enligt Monte Carlo-metoden. Parametrar: $S(0) = 100$, $\sigma = 0.5$, $r = 0.02$, $K = 100$, $T = 0.5$, $n = 100$, $N = 20$.

Utifrån samtliga simulerade värden kan ett fel, i form av standardavvikelse, beräknas och ett konfidensintervall för priset bestämmas. Felet bör inte vara större än att det i verkligheten motsvarar ett fåtal cent. I ett försök att åstadkomma detta kommer två variansreducerande tekniker att användas; antitetiskt variat och kontrollvariat. Antitetiskt variat används genomgående, då det är kopplat till simuleringen av $S(t)$ och inte optionsspecifikt. Kontrollvariat är dock mer kopplat till optionstyp och i de fall det används specificeras det i respektive avsnitt. En mer detaljerad beskrivning av metoden och de variansreducerande teknikerna finns i Appendix B.1.

### Finita differensmetoden; Crank-Nicolson

Crank-Nicolson-metoden, vilken är den andra metoden som kommer att användas, är en finit differensmetod för att lösa Black-Scholes partiella differentialekvation (2.3) med tillhörande rand- och slutvillkor. Detta görs genom att invertera ekvationen i tiden och låta rumsderivatorna vara centrerade medan tidsderivatan dels uppskattas framåt och dels bakåt. Metoden är alltså ett mellanting av Eulers framåt- och bakåtmetod och lösningen $v(x, t+\Delta t)$ bestäms enligt:

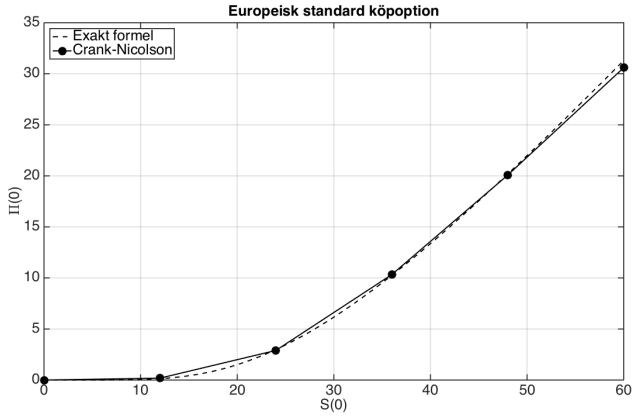
$$v(x, t + \Delta t) = \frac{1}{2} v_b(x, t + \Delta t) + \frac{1}{2} v_f(x, t + \Delta t)$$

där $v_b(x, t+\Delta t)$ och $v_f(x, t+\Delta t)$ beräknas enligt Eulers bakåt- och framåtmetod respektive.

Tidsintervallet, $[0, T]$, delas in i $n$ delintervall enligt $t_0 = 0, \ldots, t_n = T$ med steglängd $\Delta t = \frac{T}{n}$. Rumsintervallet, $x \in [0, x_{max}]$, delas på liknande sätt in i $m$ delintervall: $x_0 = 0, \ldots, x_m = x_{max}$ med steglängd $\Delta x = \frac{x_{max}}{m}$. Om nu $v_i$ betecknar vektorn innehållandes elementen $\{v(t_i, x_j)\}_{j=0}^{m}$ leder Crank-Nicolson-metoden till ekvationssystemet

$$Av_{i+1} = Bv_i \tag{2.5}$$

där $A$ och $B$ är tridiagonala matriser av storlek $m+1 \times m+1$. Dessutom är matriselementen endast beroende av rumsvariabeln, $x_j$. Hur dessa ser ut i mer detalj, tillsammans med en mer detaljerad beskrivning av metoden, presenteras i Appendix B.2. Denna metod illustreras i figur 2.




Figur 2: Crank-Nicolson-metoden med $m = 5$ tillsammans med den exakta formeln över olika startvärden $S(0)$. Parametrar: $\sigma = 0.5$, $r = 0.02$, $K = 30$, $T = 1$, $n = 500$.

#### Tidskomplexitet

Metodernas asymptotiska tidskomplexiteter är olika. Att jämföra dessa fungerar som en första analys av hur de presterar tidsmässigt och varför den ena kan komma att föredras över den andra. En enkel analys av Monte Carlo-metoden, med antagandet att generering av slumptal sker i konstant tid, ger en tidskomplexitet enligt $\mathcal{O}(N \times n)$. Detta antagande är rimligt, men i praktiken är denna tid inte försumbar, utan innebär en väsentlig mängd processortid. Tidskomplexiteten för Crank-Nicolson-metoden är $\mathcal{O}(m+n)$. Beräkningen av matriselementen i $A$ sker i $\mathcal{O}(m)$, likaså inverseringen av denna tridiagonala $m+1 \times m+1$-matris [4]. Vidare är den tidsoberoende och görs därför bara en gång. Slutligen beräknas priset i $n+1$ stycken tidssteg. Alltså kommer komplexiteten bero på den dominanta storheten av $m$ och $n$.

## 2.3 Greeks

*Greeks* mäter hur parameterkänsligt priset för en option är. De är partiella derivator av prisfunktionen $v(t, x)$ med avseende på parametrarna $x,\ \sigma,\ t$ och $r$.

Delta och gamma är derivator med avseende på $x$ enligt

$$\Delta := \partial_x v \quad \text{och} \quad \Gamma := \partial_x^2 v.$$

$\Delta$ anses vanligtvis vara den viktigaste av Greeks:en, eftersom tillgångens pris oftast löper större risk för stora förändringar än de andra parametrarna. Vidare bestämmer $\Delta$ antalet andelar (*shares*) $h_S(t)$ av den underliggande tillgången i en portfölj med hedgingstrategi enligt $h_S(t) = \Delta(t, S(t))$. I övrigt har vi vega, theta och rho enligt

$$\nu := \partial_\sigma v, \quad \Theta := \partial_t v \quad \text{och} \quad \rho := \partial_r v.$$

Vi kommer framförallt studera $\Delta$ och $\nu$, då dessa är av störst intresse när man jämför optioner. De skillnader man kan observera är ofta anledningen till varför en viss typ av exotisk option används.

# 3 Asiatiska optioner

## 3.1 Introduktion

Asiatiska optioner introducerades i slutet av 1970-talet som en ny typ av option passande handel med exempelvis råolja. De fick sitt namn 1987 av Mark Standish och David Spaughton när dessa var på affärsresa i Tokyo.

En asiatisk option har en lösenvärdesfunktion som beror på medelvärdet av den underliggande tillgångens pris över tidsintervallet $t \in [0, T]$. Givet en medelvärdesfunktion $M(0,t)$ har vi följande lösenvärden vid lösendagen:

$$\begin{aligned} &\text{Köp:} \quad AK = (M(0,T) - K)_+ , \\ &\text{Sälj:} \quad AS = (K - M(0,T))_+ . \end{aligned} \tag{3.1}$$

Det aritmetiska medelvärdet är det vanligaste, men även det geometriska används. Båda två förekommer i kontinuerliga och diskreta versioner, vilka presenteras i tabell 1. Vidare ger alltid det geometriska medelvärdet upphov till ett lägre (eller lika stort) pris för en köpoption än vad det aritmetiska gör. För säljoptioner är det tvärtom. I Appendix A.1 härleds båda fallen.

Tabell 1: Medelvärdesfunktioner.

| | Aritmetiskt | Geometriskt |
|---|---|---|
| Kontinuerlig | $\frac{1}{T}\int_0^T S(u)du$ | $\exp\left(\frac{1}{T}\int_0^T \ln S(u)du\right)$ |
| Diskret | $\frac{1}{N+1}\sum_{i=0}^N S(t_i)$ | $\left(\prod_{i=0}^N S(t_i)\right)^{\frac{1}{N+1}}$ |

Lösenvärdena för asiatiska optioner är inte lika känsliga för plötsliga prisfluktuationer som de för europeiska standardoptioner. För underliggande tillgångar med hög volatilitet, eller som potentiellt kan prismanipuleras på ett oförutsägbart sätt, innebär alltså användningen av asiatiska optioner en avsevärd riskreduktion [5].

### Sälj-köp-paritet

Asiatiska optioner har, likt de europeiska standardoptionerna, en sälj-köp-paritet. Pariteten för optioner med det aritmetiska medelvärdet i kontinuerlig tid ges av

$$\Pi_{AK}^{(A)}(t) - \Pi_{AS}^{(A)}(t) = e^{-r(T-t)}\left[\frac{1}{T}\int_0^t S(u)du + \frac{e^{r(T-t)}-1}{rT}S(t) - K\right], \quad t \in [0,T]$$

och härleds tillsammans med sin diskreta motsvarighet i Appendix A.2. Det fall som är intressant för prissättningen av optioner i nutid är dock då $t = 0$. Sälj-köp-pariteten ovan tar då formen

$$\Pi_{AK}^{(A)}(0) - \Pi_{AS}^{(A)}(0) = e^{-r(T)}\left[\frac{e^{r(T)}-1}{rT}S(0) - K\right].$$

Det går även att härleda en sälj-köp-paritet för asiatiska optioner med geometriska medelvärdesfunktioner [6]. Vi har till exempel, för det kontinuerliga fallet, pariteten enligt

$$\begin{aligned} &\Pi_{AK}^{(G)}(t) - \Pi_{AS}^{(G)}(t) = \\ &= e^{-r(T-t)}\left[G(0,t)^{\frac{t}{T}}S(t)^{\frac{T-t}{T}}\exp\left((T-t)\left\{\frac{\sigma^2}{6}\left(\frac{T-t}{T}\right)^2 + \frac{r-\sigma^2/2}{2}\frac{T-t}{T}\right\} - K\right)\right], \end{aligned}$$

med $G(0,t) = \exp\left(\frac{1}{t}\int_0^t \ln S(u)du\right)$.

## 3.2 Numerisk implementering

De numeriska metoderna har valts att uteslutet implementeras för den asiatiska optionen med det aritmetiska medelvärdet. Detta dels för att den handlas i högre utsträckning, men framförallt eftersom dess exakta prissättningsformel kräver upp till sju timmar av beräkningstid och således inte är användbar i praktiken [7].

### Monte Carlo-metoden

Implementeringen av Monte Carlo-metoden för asiatiska optioner är relativt rättfram oavsett om det är geometriskt eller aritmetiskt medelvärde som är av intresse. Värdet av den underliggande tillgången finns tillgängligt för varje tidssteg, vilket innebär att dess diskreta medelvärde kan bestämmas. Därefter kan även lösenvärdet och på så sätt priset beräknas.

Eftersom fokus ligger på optionen med det aritmetiska medelvärdet kan optionen med det diskreta geometriska medelvärdet användas som kontrollvariat. Detta eftersom den har en exakt och relativt enkel prissättningformel

$$\Pi_{AK}^{(G)}(0) = e^{-rT}\left(e^{d_1}S(0)\Phi(d_2) - K\Phi(d_2 - \sigma\sqrt{\frac{T}{3}})\right) \tag{3.2}$$

med parametrarna $d_1$ och $d_2$ enligt

$$d_1 = \frac{1}{2}(r - \frac{\sigma^2}{6})T, \quad d_2 = \frac{\log\frac{S(0)}{K} + \frac{1}{2}(r + \frac{\sigma^2}{6})T}{\sigma\sqrt{\frac{T}{3}}}.$$

En härledning av denna återfinns i Appendix A.3.

I de kommande resultatavsnitten har detta kontrollvariat genomgående använts, då det minskar standardavvikelsen kraftigt. Att så är fallet kommer att visas och styrkas i resultatdelen.

### Crank-Nicolson-metoden

Implementeringen av Crank-Nicolson-metoden för asiatiska optioner blir, till skillnad från Monte Carlo-implementeringen, något besvärligare. Grundprincipen är densamma som tidigare beskrivits, och systemet (2.5) är det som behöver lösas, men matriselementen i de båda matriserna skiljer sig från standardfallet. Framförallt eftersom Black-Scholes ekvation (2.3) här beskriver en $1+2$-dimensionell PDE.

Vi låter $Y(t) = \int_0^t S(u)du$ och dimensionsreducerar ekvationen till en $1+1$-dimensionell PDE genom $\Pi_A(t) = v(t, S(t), Y(t))$ och $v(t,x,y) = xg(t,z)$, där

$$z = \frac{1}{rT}(1 - e^{-r(T-t)}) + \frac{e^{-r(T-t)}}{T}\frac{y}{x} - e^{-r(T-t)}\frac{K}{x}.$$

Vi får då en annorlunda ekvation och därmed annorlunda matriselement. Den ekvation som $g$ satisfierar är

$$\begin{aligned} &\partial_t g + \frac{1}{2}(\gamma(t) - z)^2\partial_{zz}g = 0, \quad t \in [0,T],\ z \in \mathbb{R} \\ &\text{där } \gamma(t) = \frac{1 - e^{-r(T-t)}}{rT}. \end{aligned} \tag{3.3}$$

Gränsvärdena för $g$ lyder: $\lim_{z\to-\infty} g(t,z) = \lim_{z\to\infty}(g(t,z) - z) = 0$, vilka härleds tillsammans med ekvationen av Shreve [8]. En fullständig härledning av ekvationen genom variabelsubstitution återfinns i Appendix A.4.

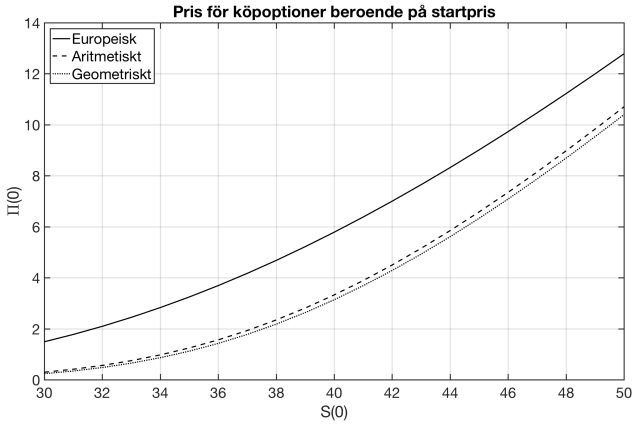
### Asymptotisk tidskomplexitet

Lösningen av ekvation (3.3) med hjälp av Crank-Nicolsons metod leder till att matriselementen är tidsberoende och måste uppdateras efter varje tidsiteration, vilket gör metoden

mer tidskrävande än för andra optioner. En enkel tidskomplexitetsanalys leder till att Crank-Nicolson-metoden för asiatiska optioner, till skillnad från övriga, har komplexiteten $\mathcal{O}(m \times n)$. Monte Carlo-metoden, å andra sidan, har som vanligt komplexiteten $\mathcal{O}(N \times n)$. Hur varje matriselement ser ut går att läsa om i Appendix B.2.

## 3.3 Resultat och diskussion

Exekveringen av de numeriska metoderna leder till diverse intressanta observationer angående hur dessa presterar. Därutöver görs olika jämförelser med teorin för optionen. Till exempel kan vi testa om priset för den geometriska asiatiska köpoptionen faktiskt är lägre än den aritmetiska. I figur 3 ser vi att fallet är sådant, men även att priset för båda asiatiska optionerna är lägre än det för den europeiska standardoptionen.



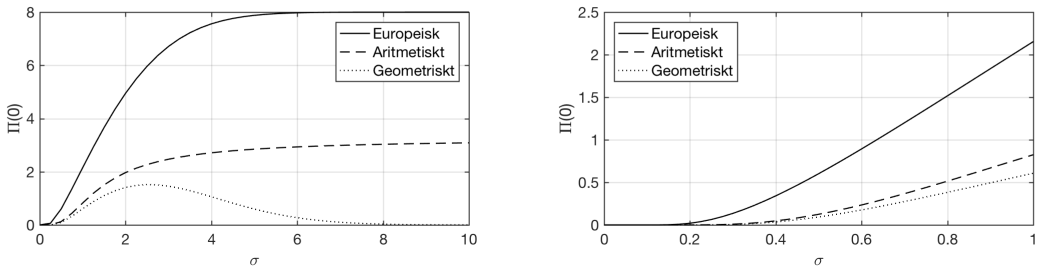

Figur 3: Priset för den aritmetiska är större eller lika med priset för den geometriska, oberoende av $S(0)$. Övriga värden: $r = 0.02$, $\sigma = 0.5$, $K = 40$, $T = 0.5$. Den europeiska optionen och den geometriska asiatiska har simulerats med sina exakta formler, medan den aritmetiska asiatiska har simulerats med Crank-Nicolson-metoden.

I avsnittet kommer vi upprepade gånger jämföra resultaten med exakta värden tillgängliga i Andrew Lyasoffs artikel från 2016 [7]. Lyasoff härleder en exakt lösningsformel, vilken han sedan använder för att beräkna diverse priser givet olika parametrar. Artikelns beräkningar gjordes med hjälp av numerisk integration.

### Greeks

Hög volatilitet är, som tidigare nämnt, en anledning till att använda en asiatisk option. Det är därför naturligt att studera $\nu$ för att rättfärdiga detta. Vi kan till exempel jämföra de europeiska och asiatiska optionspriserna som funktioner av $\sigma$. Då följer alltså $\nu$ som kurvornas lutningar.

I figur 4 ser vi att det europeiska köpoptionspriset konvergerar mot $S(0)$ för höga $\sigma$, vilket inses om vi låter $\sigma \to \infty$ i Black-Scholes-priset för en europeisk köpoption (2.4). Om vi gör samma sak i prisformeln för den geometriska asiatiska köpoptionen (3.2) ser vi att dess pris konvergerar mot 0, vilket stämmer överens med figuren. Även priset för den aritmetiska asiatiska köpoptionen konvergerar. Denna konvergenslinje beror på $S(0)$, $K$ och $T$, och närmar sig $S(0)$ för stora $T$.

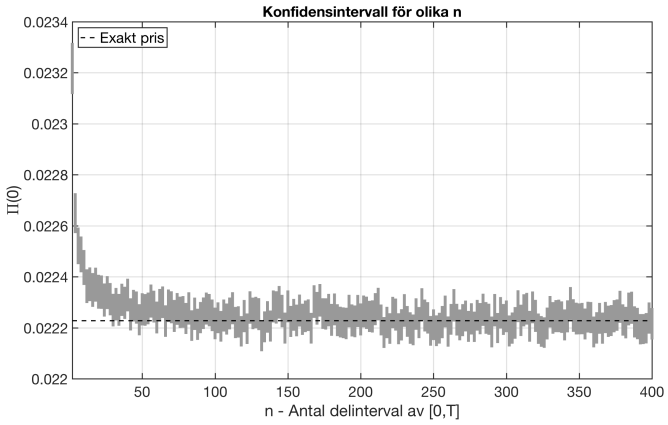



Figur 4: Till vänster har vi graferna för väldigt höga $\sigma$ och till höger ett förtydligande för mer realistiska volatiliteter. Övriga värden: $S(0) = 8$, $r = 0.02$, $K = 12$, $T = 1$. Den europeiska optionen och den geometriska asiatiska har simulerats med sina exakta formler, medan den aritmetiska asiatiska har simulerats med Crank-Nicolson-metoden.

Från figuren kan vi även konstatera att $\nu_E \geq \nu_A$ för realistiska $\sigma$ och ju högre volatilitet desto större potentiell förlust för köparen av en europeisk option jämfört med köparen av en asiatisk. Alltså innebär användningen av asiatiska optioner en riskreduktion när den underliggande tillgången har en hög volatilitet.

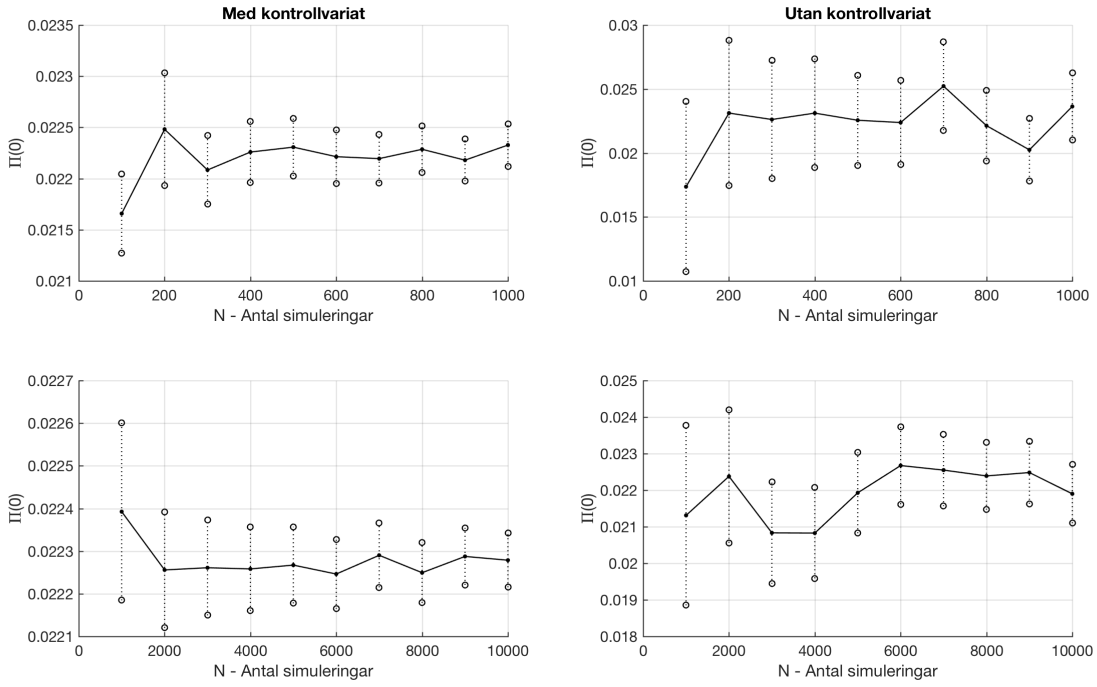
### Monte Carlo-metodens användbarhet

För att direkt kunna avgöra om Monte Carlo-metoden går att tillämpa för asiatiska optioner testas hur priset konvergerar för ökande $n$. I figur 5 syns det att konfidensintervallen verkar täcka det verkliga priset för $n > 100$ med de valda parametrarna, och i fortsättningen kommer därmed $n = 126$ att användas för $T = 1/2$.




Figur 5: Det 95-procentiga konfidensintervallet för olika val av $n$ erhållet med Monte Carlo-metoden. Parametrar: $S(0) = 1$, $r = 1/40$, $\sigma = 1/3$, $K = 11/10$, $T = 1/2$, $N = 10000$. Lyasoffs exakta pris: 0.02222765943.

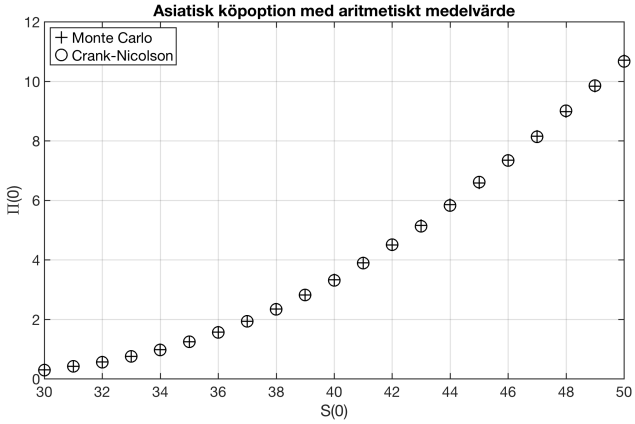
Det exakta priset för en geometrisk asiatisk option fungerar utmärkt som kontrollvariat vid Monte Carlo-simulering av priset för en aritmetisk asiatisk option. Kemna och Vorst visade detta redan i slutet av 80-talet och presenterade sina jämförelser i tabellformat [5]. Vi reproducerar därför liknande resultat med andra parametrar och presenterar dessa i mer talande grafer. I figur 6 syns det tydligt att användandet av ett kontrollvariat har stor inverkan på konfidensintervallen för priset.




Figur 6: Priset av en asiatisk köpoption beräknat med Monte Carlo-metoden för olika antal simuleringar ($N$) tillsammans med ett konfidensintervall på 95%. I de två figurerna till vänster användes kontrollvariat, men inte i de till höger. Notera storleksordningen på axlarna! Parametrar: $S(0) = 1$, $r = 1/40$, $\sigma = 1/3$, $K = 11/10$, $T = 1/2$. Lyasoffs exakta pris: 0.02222765943.

### Jämförelser metoder och programspråk emellan

Detta avsnitt påbörjas med att presentera figur 7 för att visa att de två implementerade metoderna ger lika resultat vid prissättning av optionen.




Figur 7: Priset av en asiatisk köpoption, som funktion av $S(0)$, bestämt med två skilda metoder. Övriga värden: $r = 0.02$, $\sigma = 0.5$, $K = 40$, $T = 1/2$.

Det har tidigare reflekterats kring att Crank-Nicolson-metoden och Monte Carlo-metoden har liknande asymptotiska tidskomplexiteter för asiatiska optioner. Även om detta är fallet så är inte den faktiska tidsåtgången särskilt lik metoderna emellan. I tabell 2 och 3 presenteras relativa fel som implementeringarna producerat, tiden som krävdes samt standardavvikelser

för Monte Carlo-metoden. Som tidigare har parametrarna $S(0) = 1$, $r = 1/40$, $\sigma = 1/3$, $K = 11/10$ och $T = 1/2$ använts, vilka gav det exakta priset 0.02222765943. Det relativa felet är skillnaden mellan det exakta och det simulerade priset uttryckt i procent av det senare, så att det enkelt kan jämföras med standardavvikelsen.

Tabell 2: Standardavvikelser och relativa fel som Monte Carlo-algoritmen producerat för olika $N$ uttryckt i procent av $\Pi(0)$ samt tiden det tog för MATLAB respektive `C++` att utföra beräkningarna.

| $N$ | Std (%) | Relativt fel (%) | MATLAB ($s$) | `C++` ($s$) |
|---|---|---|---|---|
| 1 000 | 0.476881 | 0.469412 | 0.020869 | 0.004439 |
| 10 000 | 0.140131 | 0.195894 | 0.139913 | 0.040559 |
| 100 000 | 0.044839 | 0.073961 | 1.648927 | 0.380492 |
| 1 000 000 | 0.014167 | 0.063732 | 76.156223 | 3.755683 |

Tabell 3: Jämförelse av olika antal delinterval av rummet ($m$) vid användning av Crank-Nicolson-metoden för en asiatisk köpoption.[1] Återigen presenteras det relativa felet uttryckt i procent av $\Pi(0)$ och tiden beräkningen tog för respektive programspråk.

| $(m, n)$ | Relativt fel (%) | MATLAB ($s$) | `C++` ($s$) |
|---|---|---|---|
| $(30, 126)$ | 0.405130 | 0.020166 | 0.001150 |
| $(100, 126)$ | 0.007771 | 0.041482 | 0.033144 |
| $(500, 126)$ | 0.002619 | 0.327330 | 0.467065 |
| $(2000, 126)$ | 0.001536 | 6.015739 | 20.898224 |

Vad gäller metoderna är det tydligt att Crank-Nicolson är att föredra. Den är överlägsen Monte Carlo-metoden vad gäller både noggrannhet och tidsåtgång. En ytterligare observation är att Monte Carlo-metodens konfidensintervall på 95% inte innehåller det exakta priset för $N = 1\ 000\ 000$. Detta beror med största sannolikhet på att $n$ inte är tillräckligt stort för att priset ska ha hunnit konvergera exakt. Tillsammans med att konfidensintervallet är väldigt litet ger detta upphov till att priset hamnar utanför.

Valet av programspråk är även det enkelt. `C++` slår MATLAB för alla storlekar på $N$ i Monte Carlo-metoden och för snabba och relativt bra approximationer med Crank-Nicolson-metoden. Eftersöks en mer exakt approximation bör MATLAB användas. Detta följer från den simpla anledningen att MATLAB är optimerat för matrisberäkningar och därför presterar bättre än `C++` för stora matriser.

## 3.4 Slutsats

Det går, utifrån presenterade resultat, att dra ett antal slutsatser. Till att börja med kan vi bekräfta att den asiatiska optionen inte är lika känslig för plötsliga prisvariationer som den europeiska standardoptionen. Den är därför med rätta en riskreducerande option för underliggande tillgångar med hög volatilitet.

Dessutom har det visats att Monte Carlo-metoden är stabil för att simulera priset av en asiatisk option. Vidare kan konfidensintervallen minskas drastiskt med hjälp av Kemna och Vorsts kontrollvariat. Metoden slår dock ändå inte Crank-Nicolson-metoden, vilken presterar mycket bättre både i noggrannhet och tidsåtgång.

Slutligen konstateras att `C++` är programspråket att föredra för en snabb och relativt noggrann prissättning. Eftersöks ett ännu mer noggrant resultat bör MATLAB användas.

[1] $m$ ökas i Crank-Nicolsons-metod eftersom detta, efter experimenterande med parametrar, visat sig effektivt för att förbättra resultaten. Priset konvergerar när uppdelningen av tidsintervallet blivit tillräcklig noggrann och en ytterligare förfining ger inte ett märkbart bättre resultat i förhållande till tidsåtgången.

# 4 Lookback-optioner

En lookback-option är en exotisk option vars lösenvärdesfunktion beror på det lägsta eller högsta värde den underliggande tillgången antagit fram till lösendagen. Detta gör det möjligt för investerare att analysera historiska optionspriser för diverse underliggande tillgångar och därefter fatta beslut baserat på tillgångens optimala värde oavsett tidpunkt [9].

## 4.1 Introduktion

Att behålla eller sälja en tillgång är det eviga dilemmat för en investerare. Sälj för tidigt och gå miste om vinst eller för sent och gå med förlust. Lookback-optioner är en lösning på detta dilemma. De ger investeraren möjlighet att köpa eller sälja den underliggande tillgången till det mest fördelaktiga priset denna har antagit under intervallet $[0, T]$. Detta innebär att prissättning av lookback-optioner kräver stor träffsäkerhet för att inte leda till stora förluster för utställaren.

Det finns två olika typer av lookback-optioner; de med fast lösenpris och de med rörligt lösenpris. Nedanstående lösenvärdesfunktioner i tabell 4 beskriver hur lösenvärdet beräknas för samtliga optioner [10].

Tabell 4: Lookback-optioner.

| | Köpoption | Säljoption |
|---|---|---|
| Fast lösenpris | $LK_{fast} = (\max_{0 \le t \le T} S(t) - K)_+$ | $LS_{fast} = (K - \min_{0 \le t \le T} S(t))_+$ |
| Rörligt lösenpris | $LK_{rörlig} = (S(T) - \min_{0 \le t \le T} S(t))_+$ | $LS_{rörlig} = (\max_{0 \le t \le T} S(t) - S(T))_+$ |

## 4.2 Numerisk implementering

Under detta arbete kommer priset på lookback-optioner med rörligt lösenpris att implementeras med både Monte Carlo- och Crank-Nicolson-metoden. De med fast lösenpris kommer dock endast att implementeras med den förstnämnde. Det finns exakta formler för samtliga typer i tabell 4, vilka kommer användas för jämförelser av simulerade priser. För de med fast lösenpris används Conze-Viswanathan-modellen [11] och för de med rörligt lösenpris används en exakt formel beskriven i Appendix A.5.

### Monte Carlo-metoden

Implementeringen av Monte Carlo-metoden för lookback-optionerna är relativt enkel. En simulerad värdeutveckling av den underliggande tillgången ger ett maximalt, minimalt och slutvärde, vilka används för att bestämma simuleringens lösenvärde.

Lookback-optionen beror starkt på volatiliteten, vilket visas i kommande avsnitt. Detta motiverar valet att använda ett kontrollvariat. Det valda kontrollvariatet för lookback-optionen med fast lösenpris är en standard europeisk köp eller sälj-option med samma lösenpris. För rörligt lösenpris används den underliggande tillgångens startvärde, $S(0)$, som lösenpris för kontrollvariatet. Implementeringen av kontrollvariatet, tillsammans med ytterligare möjliga varianter, finns att läsa om i mer detalj i Appendix B.1.

Ett problem som kan uppstå vid implementeringen av Monte Carlo-metoden är att precisionen minskar vid diskretiseringen av en kontinuerlig lookback-option. Maximum och minimum kommer alltid vara större respektive mindre för det kontinuerliga fallet än för det diskreta. Metoden undervärderar respektive övervärderar därför optionens värde. Detta finns mer utförligt att läsa om i Appendix B.3.

### Crank-Nicolson-metoden

Med utgångspunkt i Black-Scholes-ekvationen bestäms aktiepriset samt dess största värde. Låt $Y(t)$ vara det maximala värdet för den underliggande tillgången enligt

$$Y(t) = \max_{0\leq\lambda\leq t} S(\lambda) = S(0)e^{M(t)}, \quad 0 \leq t \leq T,$$
$$\text{där } M(t) = \max_{0\leq\lambda\leq t} \left( (r - \frac{\sigma^2}{2})t + \sigma W(t) \right),$$

och $\Pi_{LS}(t) = v(t, S(t), Y(t))$. Funktionen $v(t, x, y)$ uppfyller då PDE:n

$$\partial_t v + rx\partial_x v + \frac{1}{2}\sigma^2 x^2 \partial_{xx} v = rv, \quad 0 \leq t < T, \quad 0 \leq x \leq y \tag{4.1}$$

För att lösa det här problemet med Crank-Nicolson-metoden behöver vi dimensionsreducera ovanstående ekvation från en 1+2 till en 1+1 dimensionell PDE. Vi utför substitutionen $v(t, x, y) = yu(t, z)$, där $z = x/y$. Vi erhåller då

$$\begin{cases} \partial_t u + rz\partial_z u + \frac{1}{2}\sigma^2 z^2 \partial_{zz} u = ru, & 0 \leq t < T, \quad 0 < z < 1, \\ u(t, 0) = e^{-r(T-t)}, \quad u(t, 1) = u_z(t, 1), & 0 \leq t \leq T, \\ u(T, z) = 1 - z, & 0 \leq z \leq 1. \end{cases} \tag{4.2}$$

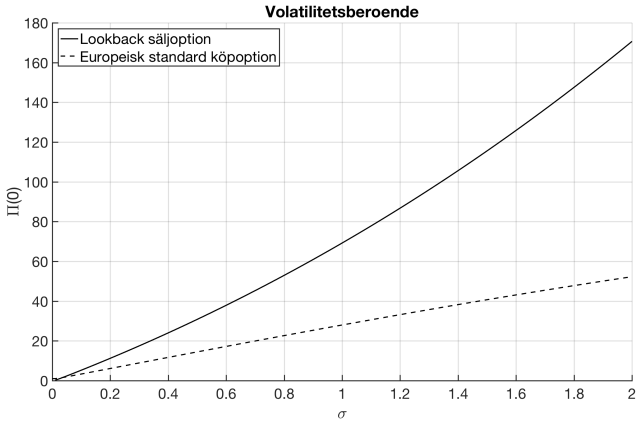
Det är på denna PDE Crank-Nicolson-metoden appliceras. En utförligare beskrivning finns i Appendix B.2.

## 4.3 Resultat och diskussion

### Greeks

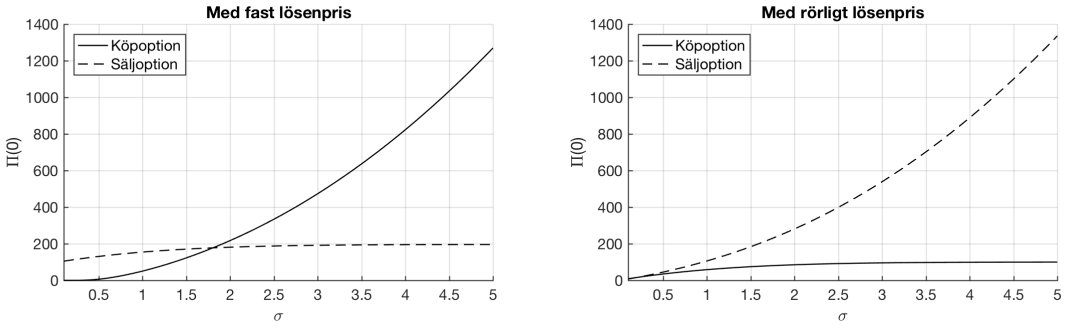
Vid en osäker marknad med hög volatilitet kan effekterna av ett felaktigt sälj- eller köpbeslut bli väldigt stora. Därför är det av intresse att undersöka $\nu$ för lookback-optioner.

I figur 8 ser vi att lookback-optionens pris, givet ett rörligt lösenpris, påverkas i större grad av en ökad volatilitet än vad en europeisk standardoption gör. Vi ser att $\nu_L > \nu_E$, vilket följer från att lookback-optionen innebär större potentiella vinster för stora $\sigma$.




Figur 8: Optionspriser som funktioner av $\sigma$ beräknat med exakta formler. Övriga värden: $S(0) = 100$ $r = 0.02$, $T = 1/2$.

Vidare kan vi till vänster i figur 9 se hur priset för en lookback-säljoption med fast lösenpris konvergerar mot lösenpriset då $\sigma$ ökar. Detta för att den underliggande tillgångens värde svänger mer för stora $\sigma$ och termen $\min_{0\leq t\leq T} S(t)$ går då mot 0. Noterbart är att korresponderande köpoption inte konvergerar, eftersom termen $\max_{0\leq t\leq T} S(t)$ inte har någon övre begränsning.

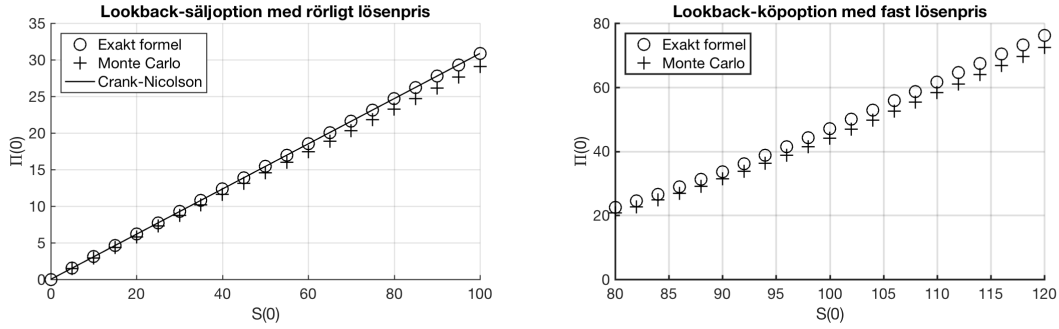



Figur 9: Till vänster: Priset av lookback-optioner med fast lösenfunktion som funktion av $\sigma$. Övriga parametrar: $S(0) = 100$, $r = 0.02$, $K = 200$, $T = 1$. Till höger: Priset av lookback-optioner med rörligt lösenfunktion som funktion av $\sigma$. Övriga parametrar: $S(0) = 100$, $r = 0.02$, $T = 1$.

Liknande observationer kan göras till höger i figur 9 som illustrerar hur lookback-optioner med rörligt lösenvärde beter sig. Här är det köpoptionen som konvergerar mot den underliggande tillgångens startvärde, vilket enkelt inses från lösenvärdesfunktionen i tabell 4. En mer intuitiv förklaring är att $\sigma \to \infty$ gör det mer värt att köpa den underliggande tillgången direkt istället för en köpoption med rörligt lösenvärde.

### Prestationen hos implementeringarna

Vi börjar detta avsnittet med att, till vänster i figur 10, illustera hur implementeringen av Monte Carlo- och Crank-Nicolson-metoden för lookback-optioner med rörligt lösenpris förhåller sig till den exakta formeln för olika $S(0)$. Till höger ser vi hur implementeringen av Monte Carlo-metoden för lookback-optionen med fast lösenpris förhåller sig till det exakta priset.




Figur 10: Till vänster: Lookback-säljoptionens värde med rörligt $K$ med avseende på startvärde $S(0)$. Använda parametrar: $r = 0.02$, $\sigma = 0.5$, $T = 0.5$, $n = 252$, $N = 10000$. Till höger: Lookback köpoptionens värde med fast $K$ med avseende på startvärde $S(0)$. Använda parametrar: $r = 0.02$, $\sigma = 0.5$, $K = 100$, $T = 1$, $n = 252$, $N = 100000$.

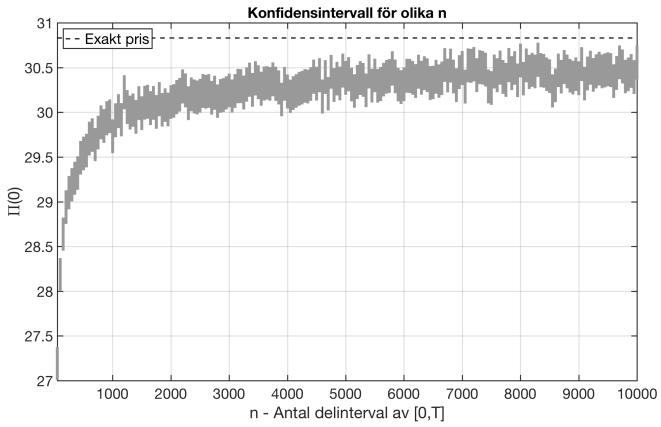
Det är tydligt att Monte Carlo-implementeringen avviker från det exakta värdet när $S(0)$ ökar, medan Crank-Nicolson-implementeringen följer den exakta formeln. Den raka lutningen i figuren förklaras av att lösningen av den reducerade PDE:n (4.2) ger att priset för en lookback-option med rörligt lösenpris bestäms som en procentsats av startvärdet.

Vidare noteras att priset från Monte Carlo-metoden alltid tycks vara något lägre än det exakta priset, oavsett om lösenpriset är rörligt eller fast. Detta förstärks ytterligare i tabell 5, där vi kan se optionspriser för olika antal simuleringar. Oavsett antal simuleringar är det relativa felet i princip oförändrat samtidigt som standardavvikelsen minskar.

Tabell 5: Lookback-säljoption med rörligt lösenpris för olika $N$ i Monte Carlo-metoden. Övriga parametrar $S(0) = 100$, $r = 0.02$, $\sigma = 0.5$, $T = 1/2$, $n = 126$ . Exakta priset är 30.8306.

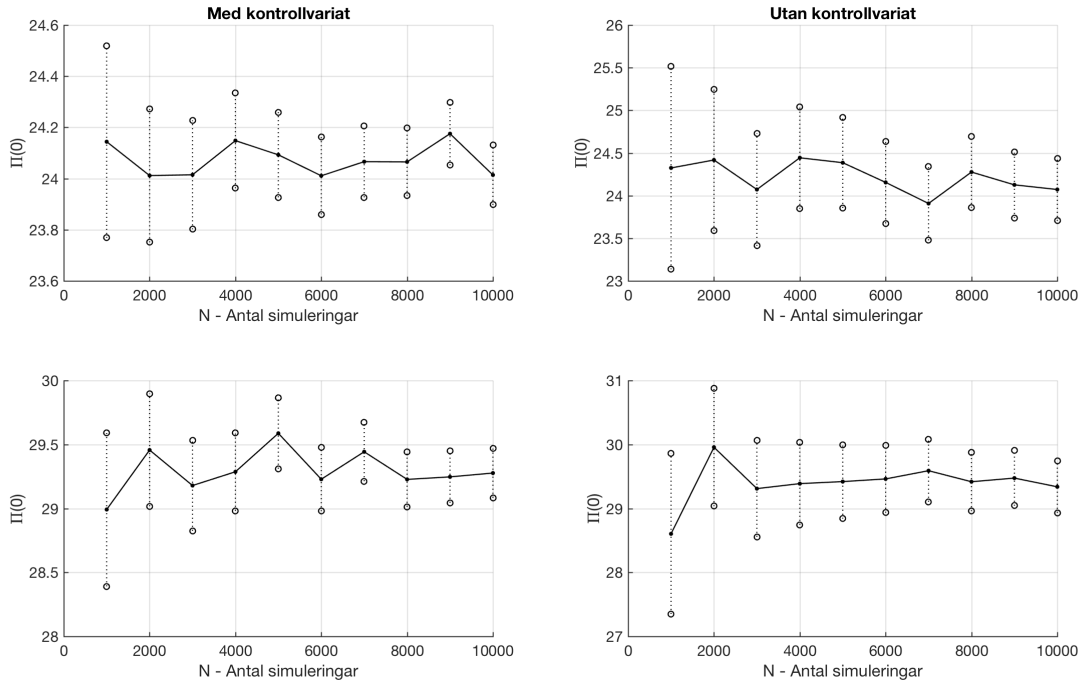
| N | Monte Carlo-pris | Std (%) | Relativt fel (%) |
|---|---|---|---|
| 10 000 | 28.4118 | 0.3339 | 8.5132 |
| 100 000 | 28.3977 | 0.1053 | 8.5672 |
| 1 000 000 | 28.4125 | 0.0333 | 8.5108 |

I figur 11 ser vi hur Monte Carlo-metoden presterar när man ökar antalet diskreta tidspunkter, $n + 1$, i Monte Carlo-simuleringarna av optioner med rörligt lösenpris. Metoden tycks kräva väldigt stora $n$ för att närma sig det exakta värdet, vilket i praktiken är ett stort problem då det är enormt tidkrävande. Detta följer från problematiken kring diskretiseringen, vilken nämndes i tidigare avsnitt.




Figur 11: Det 95-procentiga konfidensintervallet för olika val av $n$ erhållet med Monte Carlo-metoden. Parametrar: $S(0) = 100$, $r = 0.02$, $\sigma = 0.5$, $T = 1/2$, $N = 10000$.

Till följd av detta kan vi konstatera att vår implementering av Monte Carlo-metoden inte är tillräckligt bra. Om vi dock antar att vi kan lösa detta diskretiseringsproblem visar figur 12 den positiva effekten av att använda ett kontrollvariat för både lookback-optionen med rörligt och med fast lösenpris. Vi ser tydligt att implementeringen med kontrollvariat genererar ett mindre konfidensintervall än den utan, oavsett antalet iterationer $N$. Liknande effekter fås av att introducera ett kontrollvariat för lookback-optionen med fast lösenpris.




Figur 12: Priset av en lookback-köpoption $\Pi(0)$ med rörligt $K$ (överst) samt med fast $K$ (nederst) beräknat med Monte Carlo-metoden för olika antal simuleringar ($N$) tillsammans med ett konfidensintervall på 95%. Övriga parametrar är $S(0) = 100$, $r = 0.02$, $\sigma = 0.5$, $K = 100$, $T = 1$, $n = 252$.

### Programspråken

Då Monte Carlo-metoden tenderar att avvika från exakta värden väljer vi att förkasta denna till förmån för Crank-Nicolson-metoden när det gäller lookback-optioner med rörligt $K$. I tabell 6 presenteras hur Crank-Nicolson-metoden presterar både precisions- och tidsmässigt. Vi ser att det relativa felet är mycket mindre än vad Monte Carlo-metoden gav, men framförallt att det konvergerar mot 0 vid en förfinad tids- och rumsindelning. Dessutom konstateras att `C++` presterar bättre än MATLAB tidsmässigt för en approximativ prissättning, medan det omvända gäller för väldigt exakta priser. Det relativa felet är definierat på liknande sätt som för asiatiska optioner.

Tabell 6: Jämförelse av olika $m$ vid användning av Crank-Nicolson-metoden för en lookback-köpoption med rörligt lösenpris. Parametrar: $S(0) = 50$, $r = 0.02$, $\sigma = 0.5$, $T = 0.5$. Det exakta priset är 15.4153.

| $(m, n)$ | Relativt fel (%) | MATLAB ($s$) | `C++` ($s$) |
|---|---|---|---|
| $(100, 126)$ | 1.877736 | 0.004057 | 0.002006 |
| $(500, 126)$ | 0.192702 | 0.042035 | 0.051078 |
| $(1000, 126)$ | 0.019591 | 0.217531 | 0.216000 |
| $(5000, 500)$ | 0.016020 | 17.450126 | 19.130068 |

För lookback med fast lösenpris har vi ingen PDE-lösning att tillgå utan får förlita oss på Monte Carlo-metoden. Det är då viktigt att använda sig av ett kontrollvariat, vilket vi illusterade i figur 12, och att ha tillräckligt stora $n$ och $N$. I tabell 7 kan vi se hur de olika programspråken presterade tidsmässigt för Monte Carlo-implementeringen för olika val av $N$. Det är tydligt att `C++` är att föredra eftersom det är snabbare än MATLAB för stora värden

på $N$, då det är nödvändigt för att åstadkomma ett litet konfidensintervall. Värt att notera är också att MATLAB börjar få problem med $N \geq 1\ 000\ 000$ medan `C++` klarar de stora $N$ som krävs för att få väldigt precis resultat.

Tabell 7: Standardavvikelser och relativa fel som Monte Carlo-algoritmen producerat för olika $N$ uttryckt i procent av $\Pi(0)$ samt tiden det tog för MATLAB respektive `C++` att utföra beräkningarna för lookback-köpoption med fast lösenpris. Övriga parametrar: $S(0) = 50$, $r = 0.02$, $\sigma = 0.5$, $K = 55$, $T = 1$. Exakt pris: 19.1201.

| $N$ | Std (%) | Relativt fel (%) | MATLAB ($s$) | `C++` ($s$) |
|---|---|---|---|---|
| 1 000 | 1.429442 | 6.164043 | 0.034061 | 0.026267 |
| 10 000 | 0.443192 | 7.674175 | 0.214506 | 0.250483 |
| 100 000 | 0.142145 | 7.339910 | 3.187360 | 2.492441 |
| 1 000 000 | 0.044951 | 7.243420 | 330.156259 | 24.984999 |

### 4.4 Slutsats

För att sammanfatta vårt resultat kan vi fastställa att en lookback-option, oavsett lösentyp, beror starkt på volatiliteten. Vi kan vidare slå fast att Monte Carlo-implementeringen för båda typerna av lookback-optioner tenderar att avvika från den exakta formeln, vilket är speciellt tydligt för optionerna med rörligt lösenpris. Resultat visar även att Crank-Nicolson-metoden är att föredra när det gäller prissättning av lookback-optioner med rörligt lösenpris, då den ger ett mer precist resultat än vad Monte Carlo-metoden gör utan att kräva nämnvärt mer tid.

Anledningen till att Monte Carlo-implementeringen avviker från den exakta formeln beror med största sannolikhet på diskretiseringen. Vi ser tydligt att Monte Carlo-metoden undervärderar optioners värde när de beror på maximum och på samma sätt övervärderar optioner som beror på minimum. Det är alltså inte självklart att den diskreta lookback-optionen kommer konvergera mot den kontinuerliga versionen när vi ökar antalet Monte Carlo-simuleringar. En djupare diskussion kring detta presenteras i Appendix B.3.

Vidare noterar vi att valet av programspråk är `C++` vid implementeringen av både Monte Carlo- och Crank-Nicolson-metoden, men av olika anledningar. För Monte Carlo-metoden handlar det om att minska konfidensintervallet och köra många simuleringar, vilket `C++` visade sig prestera bäst för. Vid användning av Crank-Nicolson-metoden fås istället ett litet relativt fel med få simuleringar, och även här visade det sig att `C++` presterade bättre än vad MATLAB gjorde.

## 5 Barriäroptioner

### 5.1 Introduktion

Barriäroptionen (*barrier option*) kom till användning under sent 60-tal och hänvisades till som en specialoption (*special option*) rent initialt av Gerard L. Snyder 1969 [12]. Först fyra år senare, 1973, presenterades en analytisk formel för beräkningen av den här nya optionen av Robert C. Merton [13].

Optionen, för vilken Merton härledde en formel, fungerade väldigt likartat den europeiska standardoptionen. Den väsentliga skillnaden här är att optionerna är villkorade med en barriärsnivå. Barriäroptionen är aktiv fram till eller från och med att barriärnivån är nådd för $S(t)$ fram till lösendagen, beroende på om det är en Ut-option (*Knock-Out option*) eller en In-option (*Knock-In option*). De här två varianterna i kombination med utgångslägen $S(0)$, antingen över eller under barriärens värde $B$, bildar fyra olika optioner.

#### Optionsvarianterna

De fyra olika optionerna definieras som följer:

- Upp-Ut-optionen: $S(0) < B$ och optionen blir utslagen om $\exists t \in [0,T] : S(t) \geq B$.
- Upp-In-optionen: $S(0) < B$ och optionen aktiveras om $\exists t \in [0,T] : S(t) \geq B$.
- Ner-Ut-optionen: $S(0) > B$ och optionen blir utslagen om $\exists t \in [0,T] : S(t) \leq B$.
- Ner-In-optionen: $S(0) > B$ och optionen aktiveras om $\exists t \in [0,T] : S(t) \leq B$.

Precis som för de europeiska standardoptionerna beräknas lösenvärdena, förutsatt att optionen i fråga är aktiv, enligt $(S(T) - K)_+$ och $(K - S(T))_+$ för köp- respektive säljoptionerna. För enkelhetens skull fokuserar vi i det här avsnittet på Upp-optionerna, då Ner-optionerna kan behandlas analogt.

#### Användningsområden

Förhållandet mellan barriären $B$ och lösenpriset $K$ är av högsta relevans för hur optionen är avsedd att användas. Exempelvis finns det fall då valet av barriär i relation till lösenpris orsakar att optionens värde aldrig kommer överstiga 0. Det här gäller bland annat för Upp-Ut-optionen då barriären är satt under lösenpriset.

Användningsområden i mer konkreta fall för barriäroptioner är oftast i hedgingsammanhang. En ägare till en viss tillgång skulle som exempel kunna vara orolig för en temporär nedgång i värde och då kunna ha användning av att förslagsvis köpa en Ner-In-säljoption.

#### In-Ut-Pariteten

För att dra en parallell till den asiatiska optionen, där en sälj-köp-paritet gäller i en arbitragefri marknad, kan man här tala om en så kallad In-Ut-Paritet. Utifrån tidigare nämnda lösenvärdesfunktioner ser vi att en kombination av en Ut- och en In-option, med samma lösenpris $K$ och barriärnivå $B$, perfekt avspeglar den europeiska standardoptionen. Detta eftersom In-optionen blir aktiv i samma stund som Ut-optionen blir utslagen. In-Ut-Pariteten kan alltså beskrivas enligt:

$$\begin{aligned} \Pi_{Ut-K}(t) + \Pi_{In-K}(t) &= \Pi_{EK}(t) \\ \Pi_{Ut-S}(t) + \Pi_{In-S}(t) &= \Pi_{ES}(t) \end{aligned} \tag{5.1}$$

### 5.2 Numerisk implementering

Den implementering som görs för den här optionen är, precis som för tidigare nämnda optioner: Monte Carlo-metoden och Crank-Nicolson-metoden.

#### Monte Carlo

Monte Carlo-simuleringarna görs på samma sätt som för den europeiska standardoptionen med enda skillnaden att barriären tas i beaktning. Detta genom att sätta lösenvärdet till noll ifall optionen inte blivit aktiverad eller om den har blivit utslagen. Till skillnad från den asiatiska och lookback-optionen används här inget kontrollvariat, utan endast ett antitetisk variat. Metoden och variansreduceringen tas upp i mer detalj i Appendix B.1.

Implementeringen av Monte Carlo-metoden kan orsaka felavvikelser för prissättningen av en del optioner. Detta kan bero på den diskretisering som görs av den kontinuerliga barriäroptionen. Risken finns då att metoden undervärderar eller övervärderar optionens värde. Detta finns mer utförligt att läsa om i Appendix B.3.

### Crank-Nicolson

Den finita differensmetoden Crank-Nicolson-metoden som tidigare tagits upp i bakgrunden (2.2) används även för denna option. Det som görs annorlunda är att randvillkoren här tvingar exempelvis Upp-Ut-optionens pris att sättas till noll för startvärden ovanför barriären. Metoden baserar sin lösning på Black-Scholes-ekvationen (2.3), där $v(t, S(t))$ är priset för optionen vid tidpunkt $t$. Gränsvärdena som gäller för den här ekvationen med en Upp-Ut-option härleds av Shreve [8] och är:

$$\begin{cases} v(t,0) = 0, & \forall t \in [0,T] \\ v(t,B) = 0, & \forall t \in [0,T) \\ v(T,x) = (x-K)_+, & \forall x \in [0,B). \end{cases} \tag{5.2}$$

Utifrån dessa kan de två matriserna i Crank-Nicolson-metoden härledas, vilket gör att optionspriset kan simuleras med givna parametrar. Från det här beräknade priset kan även priset för Upp-In-optionen beräknas med hjälp av In-Ut-Pariteten (5.1).

### Exakt formel

Metoderna som nämnts ovan är i själva verket inte alltid nödvändiga. Detta eftersom det finns en exakt formel för att beräkna värdet för Upp-Ut-optionen som finns beskriven i Appendix A.5. Dock är detta under andra antaganden inte nödvändigtvis fallet.

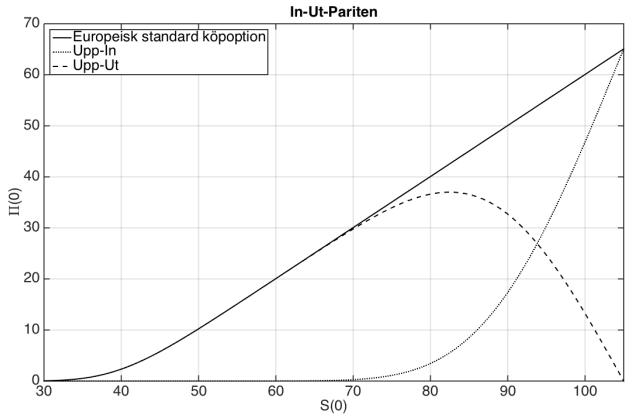
## 5.3 Resultat och diskussion

Det här resultatavsnittet kommer att påbörjas med en presentation av optionens direkta beroende av ett par olika Greeks. Därefter är det av intresse att visa hur In-Ut-Pariteten förhåller sig till diverse parametrar. Utöver detta är det av intresse att jämföra precision och relativa fel hos de två numeriska metoderna. Slutligen kommer det undersökas vilket programspråk som är att föredra för de olika metoderna.

För att säkerställa figurerna och tabellernas precision används den exakta formeln fortsättningsvis om inte annat anges. Genom det här avsnittet kommer den riskfria räntan hållas fix till $r = 0.02$.

### In-Ut-Pariteten

Från figur 13 ses att Upp-Ut-optionens pris går mot 0 och att optionen alltid kommer att vara utslagen då $S(0) \geq 105$. Upp-In-optionens pris går här mot den europeiska standardoptionens pris när vi låter $S(0)$ gå mot barriärens värde på 105. Optionen blir då konstant aktiverad och på så sätt definierad precis som den europeiska standardoptionen.

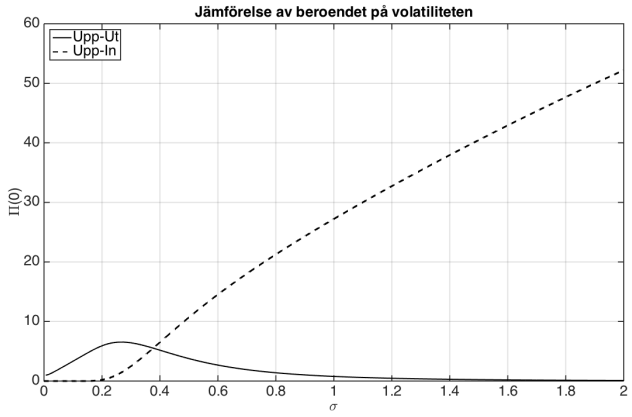



Figur 13: In-Ut-Pariteten illustrerad över priset på optionerna med avseende på startpriset $S(0)$. Parametrar som används här är i övrigt: $\sigma = 0.5$, $K = 40$, $B = 105$ och $T = 1$.

### Greeks

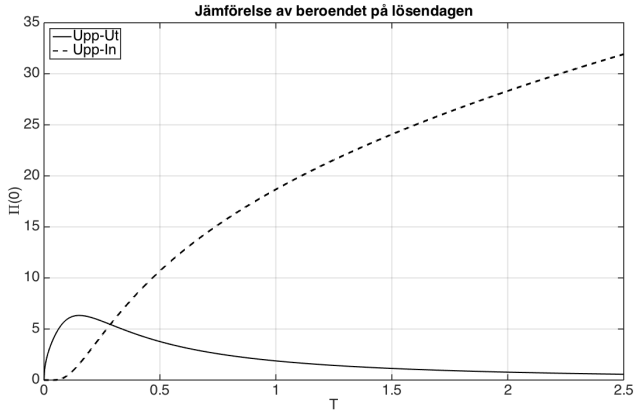
När barriäroptionens beroende av variablerna undersöks kan det urskiljas att både volatiliteten och lösendagen är av högsta relevans för huruvida In- eller Ut-optionen är den som kommer vara aktiv. Denna anledningen leder till valet att nedan illustrera hur priset för optionerna beräknas för stigande volatilitet och lösendag.

I figur 14 urskiljs tydligt hur Upp-Ut-optionens beräknade värde konvergerar mot 0 när volatiliteten stiger. Upp-In-optionen konvergerar däremot mot den europeiska köpoptionen på grund av In-Ut-Pariteten som tidigare tagits upp.




Figur 14: Optionspriset beräknat med avseende på volatilitetsvärden mellan 0 och 2. Övriga parametrar här är: $S(0) = 100$, $K = 100$, $B = 150$ och $T = 1$.

Precis som för den volatilitetsberoende figuren ser vi i figur 15 att ett senare lösendatum ger en högre sannolikhet för att Upp-Ut-optionen ska bli utslagen. Likaså kommer In-optionen att konvergera mot den europeiska standardoptionen.

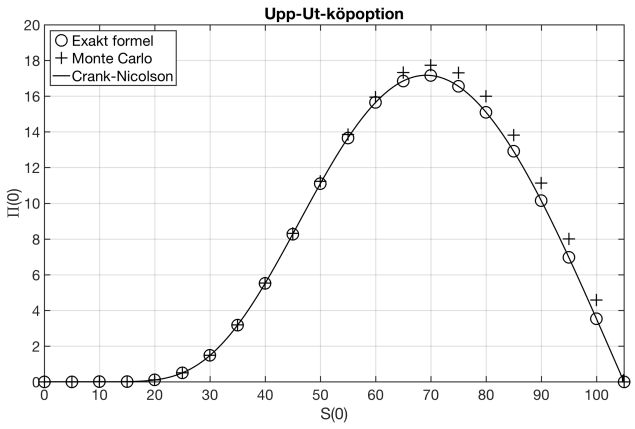



Figur 15: Optionspriset mot lösendagen mellan 0 och 2.5. Övriga parametrar här är: $S(0) = 100$, $\sigma = 0.5$, $K = 100$ och $B = 150$.

### Numeriska simuleringar och programspråk

Den här delen av resulatsavsnittet kommer behandla hur våra implementeringar av de numeriska metoderna förhåller sig till varandra tids- och precisionsmässigt.

Det som tydligast går att urskilja direkt från figur 16 är hur Monte Carlo-metoden successivt tappar precision i samband med att värdet på $S(0)$ närmare sig barriären. Crank-Nicolson-metoden håller sig däremot mer intakt till den exakta lösningsformeln.



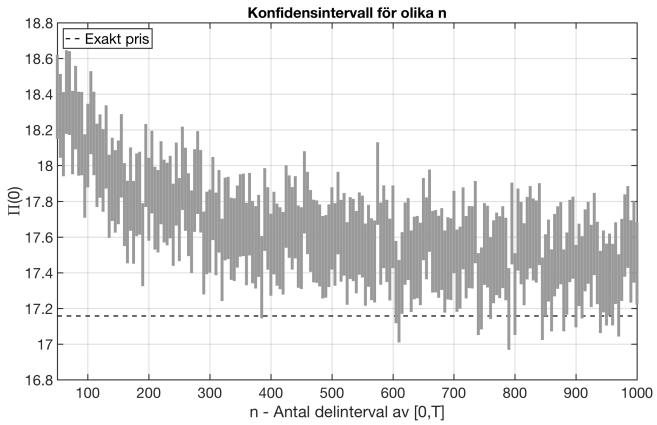

Figur 16: Upp-Ut-köpoptionens värde, med avseende på startpriset $S(0)$, för tillgången vid tid $t_0 = 0$. Parametrar som används här är i övrigt: $\sigma = 0.5$, $K = 40$, $B = 105$ och $T = 0.5$.

För att jämföra programsspråkens effektivitet för de olika metoderna har upprepade iterationer för olika precisionsnivåer genomförts. I tabell 8 presenteras resultaten för Monte Carlo-metoden. Vid dessa simuleringar används följande värden: $S(0) = 70$, $\sigma = 0.5$, $K = 40$, $B = 105$ och $T = 0.5$.

Tabell 8: Här presenteras de standardavvikelser som Monte Carlo-algoritmen producerat för olika $N$ uttryckt i procent av $\Pi(0)$ samt ett relativt fel. Dessutom presenteras tiden det tog för MATLAB respektive `C++` för simuleringarna.

| $N$ | Std (%) | Relativt fel (%) | MATLAB ($s$) | `C++` ($s$) |
|---|---|---|---|---|
| 1 000 | 2.018876 | 6.943140 | 0.020898 | 0.004006 |
| 10 000 | 0.648449 | 6.096618 | 0.130358 | 0.035010 |
| 100 000 | 0.208405 | 4.699859 | 1.506163 | 0.330011 |
| 1 000 000 | 0.065977 | 4.673220 | 66.228936 | 3.334956 |

Vid Monte Carlo-simuleringarna urskiljs tydligt att `C++` är att föredra vid försök att minimera standardavvikelsen, då beräkningstiden kan bli avsevärt mindre för stora $N$. Det går även notera att det relativa felet verkar konvergera mot ett betydande fel. Vid dessa simuleringar används en parameter $n = 126$ som representerar antalet delintervall för Monte Carlo-iterationerna. För att undersöka konvergensen närmare låter vi $N = 10\ 000$ vara fixt och låter $n$ gå mot 1000 i figur 17.




Figur 17: Optionspriset med konfidensintervall på 95% mot antalet delintervall av [0, T] upp till 1000 med Monte Carlo-metoden. Övriga parametrar här är: $S(0) = 70$, $\sigma = 0.5$, $K = 40$, $B = 105$ och $T = 0.5$.

Från denna figur kan det urskiljas att det som krävs för att få Monte Carlo-metoden att konvergera mot det exakta priset är väldigt höga $n$ och $N$. Detta gör metoden väldigt långsam och opraktisk i det här avseendet. Resultatet visar på bristerna hos den här implementeringen av Monte Carlo-metoden och förstärker intresset för en mer avancerad modell, med användning av så kallade Brownska broar (*Brownian bridges*), som beskrivs kortfattat i Appendix B.3.

I tabell 9 presenteras de värden som erhölls med Crank-Nicolson-metoden. Det går enkelt observera att metoden snabbt konvergerar mot det exakta priset. Simuleringarna visar även att tidsskillnaden mellan programspråken, för en precision med dessa val av antal delintervall $(m, n)$, är nästan obefintlig.

Tabell 9: Jämförelse av olika antal rumsdelintervall $m$ och tidsdelintervall $n$ vid användning av Crank-Nicolson-metoden för en Upp-Ut-köpoption.

| $(m, n)$ | Relativt fel (%) | MATLAB ($s$) | C++ ($s$) |
|---|---|---|---|
| $(15, 15)$ | 0.424446 | 0.000540 | 0.000086 |
| $(210, 210)$ | 0.000810 | 0.016837 | 0.011236 |
| $(1050, 1050)$ | 0.000032 | 1.441715 | 1.320856 |

## 5.4 Slutsats

Som sammanfattning av barriäroptionen går det konstatera att det här instrumentet är starkt beroende av volatiliteten och lösendagen. Bland annat urskiljs det att hög volatilitet, eller stort $T$, får Upp-In-optionen att aktiveras med hög sannolikhet och då bli prissatt som den europeiska standardoptionen. Från dessa resultat går det i hög grad specificera parametrarna hos optionen för anpassning till det behov som den underliggande tillgången skapar för såväl optionsutställaren som optionsägaren. Resultatet som presenterats för de numeriska metoderna visar att Crank-Nicolson-metoden ger en betydligt högre precision, och kortare beräkningstider, än vad Monte Carlo-metoden gör.

Det går även säga att Monte Carlo-metoden är för ineffektiv för att få ut värden för optionen med tillräckligt små relativa fel på rimliga beräkningstider. Metoden behöver en mer avancerad uppbyggnad, förslagsvis med Brownska broar, för att klara av prissättningar för den här optionen. Sammanfattningsvis går det även säga att C++ är programspråket att föredra för de båda numeriska metoderna.

# 6 Slutsats

Syftet med denna rapport var att redogöra för hur Monte Carlo- och Crank-Nicolson-metoden fungerar och presterar för asiatiska, lookback- och barriäroptioner. Det var även av intresse att jämföra de olika optionernas beroende på olika parametrar. Metoderna skulle även implementeras i de två programmeringsspråken MATLAB och `C++`.

Från undersökningen av utvalda Greeks går det att urskilja separata användningsområden för de olika optionerna. Vi kan dra slutsatsen att en asiatisk option är att föredra när den underliggande tillgången har en hög volatilitet och riskminimering är efterfrågat. Vidare kan vi slå fast att de andra optionerna är mer känsliga för volatiliteten och implicerar en större risk för såväl optionsutställare som för optionsägare. I fallet för en barriäroption med stigande volatilitet konvergerar In-optionerna mot den europeiska standardoptionen och förlorar då indirekt sitt syfte, medan Ut-optionen konvergerar mot 0. Det här är även fallet för barriäroptionen då tiden till lösendagen förlängs. För lookback-optioner kan vi konstatera att dessa beror starkt på volatiliteten, vilket speglas i hur priset för dessa optioner förändras med en ökande volatilitet.

När de numeriska metoderna jämfördes för de olika optionerna framgick det att Crank-Nicolson-metoden är att föredra i samtliga fall. Detta eftersom den ger ett mindre relativt fel, tillsammans med en kortare beräkningstid, i jämförelse med Monte Carlo-metoden. Vid implementering av Monte Carlo-metoden konstaterades ett betydande fel hos både lookback- och barriäroptionen. Detta fel beror med största sannolikhet på diskretiseringen av de kontinuerliga fallen, där eventuella extremvärden mellan tidspunkterna inte tas i beaktning. För att motverka detta hade en Monte Carlo-metod som även modellerat en Brownsk bro (*Brownian bridge*) mellan tidspunkterna varit att föredra. Den här metoden hade då genererat ett mycket mer sannolikt värde för varje tidsintervall och på så sätt gjort metoden mer exakt och förhindrat diskretiseringsfelet. Diskretiseringen får inte lika stora konsekvenser för ett medelvärde, varför metoden blir stabil vid simulering av den asiatiska optionens värde.

Vidare har vi, för Monte Carlo-implementeringen, observerat en betydande effekt av kontrollvariat. Framförallt för asiatiska optioner, där användandet av ett kontrollvariat genererar väldigt små konfidensintervall. Trots detta är Crank-Nicolson-metoden att föredra även för den asiatiska optionen.

Vid jämförelse av programspråken konstaterades att `C++` var att föredra i de flesta avseenden. Detta berodde på att beräkningstiden kunde förkortas avsevärt vid stora beräkningar. Ett undantagsfall kan exempelvis vara när hög noggrannhet efterfrågas för Crank-Nicolson-metoden, detta eftersom MATLAB är bättre på att hantera stora matrisoperationer. Den här extrema sortens noggrannhet behövs däremot väldigt sällan och av denna anledning blir `C++` det allmänt bästa valet utifrån de här numeriska metoderna.

Värt att nämna är att koden vi tagit fram för optionerna även är applicerbar för fallet då volatiliteten är en stokastisk process. För sådana fall går det inte att ta fram en exakt formel, utan numeriska approximationer är ett måste. Här har således de undersökta metoderna större relevans.

Avslutningsvis konstaterar vi att ett intressant område för framtida studier är att implementera den metod som nämnts kring användning av Brownska broar för att korrigera Monte Carlo-metoden. För lookback-optioner kan även ett multipelt kontrollvariat vara av intresse för att minimera standardavvikelsen. När det kommer till att lösa de partiella differentialekvationerna numeriskt kan även finita elementmetoden vara intressant att implementera och jämföra med Crank-Nicolson-metoden.

# Referenser


[1] Eduardo S. Schwartz. The valuation of warrants: Implementing a new approach. *Journal of Financial Economics*, 4(1):79–93, January 1977.

[2] Phelim P. Boyle. Options: A Monte Carlo approach. *Journal of Financial Economics*, 4(3):323–338, May 1977.

[3] Conrad Sanderson and Ryan Curtin. Armadillo: a template-based c++ library for linear algebra. *The Journal of Open Source Software*, 1(2), jun 2016.

[4] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996.

[5] A. G. Z. Kemna and A. C. F. Vorst. A pricing method for options based on average asset values. *Journal of Banking & Finance*, 14(1):113–129, March 1990.

[6] Y.K. Kwok. *Mathematical Models of Financial Derivatives*. Springer Finance. Springer Berlin Heidelberg, 2008.

[7] Andrew Lyasoff. Another look at the integral of exponential Brownian motion and the pricing of Asian options. *Finance and Stochastics*, 20(4):1061–1096, October 2016.

[8] S.E. Shreve. *Stochastic Calculus for Finance II: Continuous-Time Models*. Number v. 11 in Springer Finance Textbooks. Springer, 2004.

[9] F. de Weert. *Exotic Options Trading*. The Wiley Finance Series. Wiley, 2011.

[10] D.J. Duffy. *Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach*. The Wiley Finance Series. Wiley, 2006.

[11] Antoine Conze and Viswanathan. Path Dependent Options: The Case of Lookback Options. *Journal of Finance*, 46(5):1893–1907, December 1991.

[12] Gerard L. Snyder. Alternative form of options. *Financial Analysts Journal*, 25(5):93–99, 1969.

[13] Robert C. Merton. Theory of Rational Option Pricing. *Bell Journal of Economics*, 4(1):141–183, Spring 1973.

# A Bevis och härledningar

## A.1 Asiatiska optioners pris vid geometriskt kontra aritmetiskt medelvärde

I detta avsnitt visar vi att det geometriska medelvärdet ger upphov till ett lägre pris för en asiatisk köpoption (och ett högre pris för en säljoption) än vad det aritmetiska medelvärdet gör. Notera att priset för en asiatisk option vid $t = 0$ är

$$\text{Köp:} \quad \Pi_{AK}(0) = e^{-rT}\mathbb{E}\Big[\big[M(0,T,S(t)) - K\big]_+\Big],$$
$$\text{Sälj:} \quad \Pi_{AS}(0) = e^{-rT}\mathbb{E}\Big[\big[K - M(0,T,S(t))\big]_+\Big],$$

för $t \in [0,T]$ och någon funktion $M$ som ger ett visst typ av medelvärde av $S(t)$ på detta interval. Vi har följande definitioner för de olika medelvärdena

$$\text{Aritmetiskt:} \quad A(0,T,S(t)) = \frac{1}{N+1}\sum_{i=0}^{N} S(t_i),$$
$$\text{Geometriskt:} \quad G(0,T,S(t)) = \left(\prod_{i=0}^{N} S(t_i)\right)^{\frac{1}{N+1}}.$$

och börjar med en nödvändig sats.

**Sats 1.** *Det geometriska medelvärdet av en värdemängd $\{m_i \geq 0\}_{i=0}^{N}$ är alltid mindre eller lika med mängdens aritmetiska medelvärde.*

***Bevis.*** Vi kommer bevisa satsen genom induktion. Om vi låter

$$\alpha = \frac{m_0 + m_1 + \cdots + m_N}{N+1}$$

vill vi alltså visa $\alpha^{N+1} \geq m_0 m_1 \cdots m_N$.

**Basfall:** För $N = 0$ har vi en likhet och olikheten ovan är således sann.

**Hypotes:** Antag att olikheten håller för alla icke-negativa heltal 0 till $N$.

**Induktionssteg:** Antag att vi har värdemängden $\{m_i \geq 0\}_{i=0}^{N+1}$ med ett aritmetiskt medelvärde $\alpha$ som uppfyller $(N+2)\alpha = m_0 + m_1 + \cdots + m_{N+1}$.

Om $m_i = \alpha \ \forall i = 0, 1, ..., N+1$ har vi en likhet mellan de olika medelvärdena och satsen är bevisad. Annars kan vi hitta ett värde i mängden som är mindre än $\alpha$ och ett som är större, säg $m_N > \alpha$ och $m_{N+1} < \alpha$. Då har vi

$$(m_N - \alpha)(\alpha - m_{N+1}) > 0. \tag{A.1}$$

Betrakta nu mängden $m_0, m_1, ..., m_{N-1}, \tilde{m}$ med

$$\tilde{m} := m_N + m_{N+1} - \alpha \geq m_N - \alpha > 0.$$

Eftersom

$$(N+1)\alpha = m_0 + \cdots + m_{N-1} + m_N + m_{N+1} - \alpha = m_0 + \cdots + m_{N-1} + \tilde{m}$$

är $\alpha$ medelvärdet för $m_0, m_1, ..., m_{N-1}, \tilde{m}$ och enligt hypotesen har vi

$$\alpha^{N+2} = \alpha^{N+1}\alpha \geq m_0 m_1 \cdots m_{N-1}\tilde{m}\alpha. \tag{A.2}$$

Enligt (A.1) har vi

$$(m_N + m_{N+1} - \alpha)\alpha - m_N m_{N+1} = (m_N - \alpha)(\alpha - m_{N+1}) > 0,$$

alltså

$$\tilde{m}\alpha > m_N m_{N+1} \tag{A.3}$$

och $\alpha > 0$. Detta innebär att om något av värdena $m_0, m_1, ..., m_{N-1}$ är lika med 0 har vi en strikt olikhet i (A.2). Om fallet inte är sådant har vi i och med (A.3) en övre gräns för högerledet i (A.2), vilket innebär att olikheten

$$\alpha^{N+2} \geq m_0 m_1 \cdots m_{N+1}$$

håller för båda fallen och satsen är således bevisad.

■

Vi är nu redo att bevisa satsen vi är intresserade av!

**Sats 2.** *Det geometriska medelvärdet ger upphov till ett lägre pris för en asiatisk köpoption (och ett högre pris för en säljoption) än vad det aritmetiska medelvärdet gör.*

***Bevis.*** Vi har följande implikation för två stokastiska variabler $X$ och $Y$:

$$X \leq Y \implies \mathbb{E}[X] \leq \mathbb{E}[Y]. \tag{A.4}$$

Om vi låter $X = e^{-rT}[G-K]_+$ och $Y = e^{-rT}[A-K]_+$ har vi alltså visat satsen, så länge $X \leq Y$ stämmer. Det gör det enligt följande ekvivalenta uttryck, där Sats 1 och det faktum att maximumfunktionen inte stör olikheter har använts:

$$\begin{aligned} G \leq A &\iff G - K \leq A - K \iff [G-K]_+ \leq [A-K]_+ \iff \\ &\iff e^{-rT}[G-K]_+ \leq e^{-rT}[A-K]_+. \end{aligned}$$

$$\therefore \quad e^{-rT}\mathbb{E}\Big[\big[G(0,T,S(t)) - K\big]_+\Big] \leq e^{-rT}\mathbb{E}\Big[\big[A(0,T,S(t)) - K\big]_+\Big].$$

Beviset för parentesen gällande säljoptioner följer från samma typ av resonemang som ovan, vilket inses lätt genom att multiplicera den första olikheten med $-1$.

■

## A.2 Aritmetiska asiatiska optioners sälj-köp-paritet

Nedan härleds sälj-köp-pariteten för asiatiska optioner med aritmetiska medelvärdesfunktioner, både kontinuerliga och diskreta, presenterade i Tabell 1.

**Sats 3.** *Antag att vi som vanligt har en finansiell tillgång med ett initialt pris $S(0)$ och priset $S(t)$ vid tiden $t > 0$ enligt*

$$S(t) = S(0)e^{(r-\frac{1}{2}\sigma^2)t+\sigma W(t)}. \tag{A.5}$$

*Då gäller den kontinuerliga sälj-köp-pariteten*

$$\Pi_{AK}(t) - \Pi_{AS}(t) = e^{-r(T-t)}\left(\frac{1}{T}\int_0^t S(u)du + \frac{e^{r(T-t)}-1}{rT}S(t) - K\right)$$

*och den diskreta*

$$\Pi_{AK}(t) - \Pi_{AS}(t) = e^{-r(T-t)}\left(\frac{1}{N+1}\sum_{i=0}^{n} S(t_i) + \frac{S(t)}{N+1}\frac{1-e^{r(T+1)}}{1-e^r} - K\right).$$

*I den senare är $N$ antalet delinterval på $[0,T]$, $t_i$ olika tider med $t_0 = 0$, $t_n = t$ och $t_N = T$ samt $t_i = i\frac{T}{N}$ (intervallet likformigt uppdelat).*

***Bevis.*** Vi har följande definitioner för de asiatiska optionernas lösenvärde vid lösendagen:

$$\Pi_{AK}(T) = \big[M(0,T) - K\big]_+ \quad \text{och} \quad \Pi_{AS}(T) = \big[K - M(0,T)\big]_+. \tag{A.6}$$

Notera att

$$\Pi_{AK}(T) - \Pi_{AS}(T) = \big[M(0,T) - K\big]_+ - \big[K - M(0,T)\big]_+ = M(0,T) - K$$

är ett simplare sätt att skriva lösenvärdet för portföljen ovan. Vi har priset vid $t = 0$ för ett derivat med lösenvärde $Y = g(S(T))$ enligt $\Pi_Y(0) = e^{-rT}\mathbb{E}[g(S(T))]$. Då följer att

$$\Pi_{AK}(0) - \Pi_{AS}(0) = e^{-rT}\mathbb{E}[M(0,T) - K].$$

Givet att vi befinner oss i en tidpunkt $t \in [0,T]$ har vi informationen $\mathcal{F}_S(t)$ om tillgångens värdeutveckling över $[0,t]$. Då har vi

$$\Pi_{AK}(t) - \Pi_{AS}(t) = e^{-r(T-t)}\mathbb{E}[M(0,T) - K \mid \mathcal{F}_S(t)]. \tag{A.7}$$

Med (A.7) kan vi ta oss an de enskilda fallen och vi börjar med det kontinuerliga. Vi har

$$\begin{aligned}
\Pi_{AK}(t) - \Pi_{AS}(t) &= e^{-r(T-t)}\mathbb{E}\left[\frac{1}{T}\int_0^T S(u)du - K \mid \mathcal{F}_S(t)\right] \\
&= e^{-r(T-t)}\mathbb{E}\left[\frac{1}{T}\int_0^t S(u)du + \frac{1}{T}\int_t^T S(u)du - K \mid \mathcal{F}_S(t)\right] \\
&= e^{-r(T-t)}\left(\frac{1}{T}\int_0^t S(u)du + \frac{1}{T}\int_t^T \mathbb{E}\big[S(u)\big]du - K\right).
\end{aligned}$$

Vi skriver om $S(u)$, $u \in [t,T]$, med hjälp av (A.5) enligt

$$S(u) = S(t)e^{(r-\frac{1}{2}\sigma^2)(u-t)+\sigma\sqrt{u-t}\frac{W(u-t)}{\sqrt{u-t}}}$$

och noterar att $W(u-t)/\sqrt{u-t} \sim N(0,1)$. Låt oss kalla denna stokastiska process för $X$, alltså har vi $S(u) = h(X)$. Då följer, enligt

$$\mathbb{E}[h(X)] = \int_{\mathbb{R}} h(x)f_X(x)dx \quad \text{och} \quad f_X(x) = \phi(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2},$$

att

$$\begin{aligned}\mathbb{E}[S(u)] &= \int_{\mathbb{R}} S(t)e^{(r-\frac{1}{2}\sigma^2)(u-t)+\sigma\sqrt{u-t}x-\frac{1}{2}x^2}\frac{1}{\sqrt{2\pi}}dx \\ &= S(t)e^{r(u-t)}\int_{\mathbb{R}} e^{-\frac{1}{2}(x-\sigma\sqrt{u-t})^2}\frac{1}{\sqrt{2\pi}}dx = S(t)e^{r(u-t)},\end{aligned} \tag{A.8}$$

där den sista likheten inses genom $\int_{\mathbb{R}}\phi(z)dz = 1$.

Vi kan nu avsluta den första delen av beviset genom

$$\begin{aligned}\Pi_{AK}(t) - \Pi_{AS}(t) &= e^{-r(T-t)}\left(\frac{1}{T}\int_0^t S(u)du + \frac{1}{T}\int_t^T S(t)e^{r(u-t)}du - K\right) \\ &= e^{-r(T-t)}\left(\frac{1}{T}\int_0^t S(u)du + \frac{e^{r(T-t)}-1}{rT}S(t) - K\right).\end{aligned}$$

Det diskreta fallet följer från ett nästan identiskt resonemang, men vi behöver ett uttryck för $\mathbb{E}[S(t_i)]$, $i = n, ..., N$, likt det för $\mathbb{E}[S(u)]$ ovan. Om vi låter

$$S(t_i) = S(t_n)e^{(r-\frac{1}{2}\sigma^2)(t_i)+\sigma\sqrt{t_i}\frac{W(t_i)}{\sqrt{t_i}}}$$

har vi

$$\mathbb{E}[S(t_i)] = S(t_n)e^{rt_i},$$

vilket inses genom förenklingarna i (A.8). Vi kan nu avsluta beviset:

$$\begin{aligned}\Pi_{AK}(t) - \Pi_{AS}(t) &= e^{-r(T-t)}\mathbb{E}\left[\frac{1}{N+1}\sum_{i=0}^{N} S(t_i) - K \mid S(t_0), \cdots, S(t_n)\right] \\ &= e^{-r(T-t)}\left(\frac{1}{N+1}\left(\sum_{i=0}^{n} S(t_i) + \sum_{i=n+1}^{N}\mathbb{E}\big[S(t_i)\big]\right) - K\right) \\ &= e^{-r(T-t)}\left(\frac{1}{N+1}\left(\sum_{i=0}^{n} S(t_i) + \sum_{i=n+1}^{N} S(t_n)e^{rt_i}\right) - K\right) \\ &= e^{-r(T-t)}\left(\frac{1}{N+1}\sum_{i=0}^{n} S(t_i) + \frac{S(t)}{N+1}\frac{1-e^{r(T+1)}}{1-e^r} - K\right).\end{aligned}$$

■

## A.3 Härledning av priset för geometriska asiatiska optioner

Det här kommer vara en härledning av det analytiska priset för en asiatisk option grundad på det geometriska medelvärdet, kallad GA-option. Endast beviset för köpoptioner kommer behandlas då säljoptioner följer analogt.

Hämtade formler och satser kommer från: Paul Glasserman - *Monte Carlo Methods in Financial Engineering* [14].

### Teori

För att kunna härleda den analytiska formeln behöver vi först ta upp ett par kända satser. Antagandet om värdeutvecklingen hos $S(t)$, och formeln för priset av en europeisk standard köpoption, upprepas nedan som första sats av bekvämlighetskäl. De återfinns dessutom i sin helhet i bakgrunden som formel (2.1) och (2.3).

**Sats 4.** *Betrakta $S(t)$, för vilken värdet beskrivs enligt:*

$$S(t) = S(0)e^{(r-\frac{\sigma^2}{2})t+\sigma W(t)}. \tag{A.9}$$

*Priset för en europeisk standard köpoption på ovan underliggande tillgång med lösenpris $K$ och lösendag $T$ ges av $\Pi_{EK}(0) = \mathbf{E}\left[e^{-rT}(S(T)-K)_+\right]$, vilket bestäms till:*

$$\Pi_{EK}(0) = S(0)\Phi(d) - e^{-rT}K\Phi(d-\sigma\sqrt{T}), \quad d = \frac{\log\frac{S(0)}{K} + (r+\frac{\sigma^2}{2})T}{\sigma\sqrt{T}}. \tag{A.10}$$

**Sats 5.** *(Formel 2.23 från Glasserman):*
*En linjär transformation av en normalfördelad vektor är också normalfördelad:*

$$X \sim N(\mu, \Sigma) \implies AX \sim N(A\mu, A\Sigma A^T). \tag{A.11}$$

*$\mu$ är en vector av längd $d$, $\Sigma$ kovariansmatrisen av storlek $d \times d$ och $A$ en $k \times d$-matris som beskriver den linjära transformationen.*

**Sats 6.** *(Formel 3.6 från Glasserman):*
*Låt $\Sigma$ vara kovariansmatrisen för $(W(t_1), \ldots, W(t_n))$, då har vi att*

$$\Sigma_{i,j} = \min(t_i, t_j). \tag{A.12}$$

### Härledning av pris för GA köpoptioner

Lösenvärdet hos den sökta optionen vid lösendagen är

$$Y(T) = \left(\left(\prod_{i=1}^{n} S(t_i)\right)^{1/n} - K\right)_+,$$

medan priset för optionen bestäms av

$$\Pi_{AK}(0) = e^{-rT}\mathbf{E}\left[\left(\left(\prod_{i=1}^{n} S(t_i)\right)^{1/n} - K\right)_+\right].$$

Det diskreta geometriska medelvärdet av $S(t)$ är $\left(\prod_{i=1}^{n} S(t_i)\right)^{1/n}$ och om vi beskriver $S(t)$ som i ekvation (A.9) kan vi uttrycka detta geometriska medelvärde som:

$$\left(\prod_{i=1}^{n} S(t_i)\right)^{1/n} = S(0)\exp\left((r-\frac{\sigma^2}{2})\frac{1}{n}\sum_{i=1}^{n} t_i + \frac{\sigma}{n}\sum_{i=1}^{n} W(t_i)\right). \tag{A.13}$$

Nu kan vi med hjälp av ekvationerna (A.11) och (A.12) betrakta $\sum_{i=1}^{n} W(t_i)$ som en linjär transformation av vektorn $X = [W(t_1), \ldots, W(t_n)]$ genom att låta $A = [1, \ldots, 1]$ vara en $1 \times n$-vektor och kovariansmatrisen av storlek $n \times n$ beskrivas av:

$$C = \begin{bmatrix} t_1 & t_1 & t_1 & \ldots & t_1 \\ t_1 & t_2 & t_2 & \ldots & t_2 \\ t_1 & t_2 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & t_{n-1} & t_{n-1} \\ t_1 & t_2 & \ldots & t_{n-1} & t_n \end{bmatrix}.$$

Detta ger att

$$\sum_{i=1}^{n} W(t_i) = AX \sim N\left(0, \sum_{i=1}^{n} (2i-1) t_{n+1-i}\right). \tag{A.14}$$

Låt nu

$$\tilde{T} = \frac{1}{n}\sum_{i=1}^{n} t_i, \quad \tilde{\sigma}^2 = \frac{\sigma^2}{n^2 \tilde{T}} \sum_{i=1}^{n} (2i-1) t_{n+1-i}, \quad \tilde{r} = r - \frac{1}{2}(\sigma^2 - \tilde{\sigma}^2)$$

. Om vi använder detta kan vi skriva om ekvation (A.13) ovan som

$$S(\tilde{T}) = \left(\prod_{i=1}^{n} S(t_i)\right)^{1/n} = S(0) \exp\left((\tilde{r} - \frac{\tilde{\sigma}^2}{2})\tilde{T} + \tilde{\sigma} W(\tilde{T})\right). \tag{A.15}$$

Vi ser att detta beskriver en underliggande tillgång som i formel (A.9) med parametrarna: $\tilde{T}$, $\tilde{\sigma}^2$, $\tilde{r}$. Dessutom ser vi att det nu går att uttrycka optionens pris som

$$\Pi_{AC}(0) = e^{-rT} \mathbf{E}\left[\left(S(\tilde{T}) - K\right)_+\right]. \tag{A.16}$$

Detta väntevärde kan vi beräkna med hjälp av formel (A.10) för prissättningen av standard europeiska köpoptioner. Gör vi detta fås slutligen priset som

$$\begin{aligned} \Pi_{AK}(0) &= e^{-rT} \mathbf{E}\left[\left(\left(\prod_{i=1}^{n} S(t_i)\right)^{1/n} - K\right)_+\right] = \\ &= e^{-rT}\left(e^{\tilde{r}\tilde{T}} S(0) \Phi(\tilde{d}) - K \Phi(\tilde{d} - \tilde{\sigma}\sqrt{\tilde{T}})\right), \end{aligned}$$

där $\tilde{d}$ beskrivs enligt

$$\tilde{d} = \frac{\log \frac{S(0)}{K} + (\tilde{r} + \frac{\tilde{\sigma}^2}{2})\tilde{T}}{\tilde{\sigma}\sqrt{\tilde{T}}}.$$

Om vi nu bestämmer oss för att låta intervallet $[0, T]$ vara likformigt uppdelat, så att $t_i = \frac{(i-1)T}{n-1}$, kan vi bestämma:

$$\tilde{T} = T/2, \tag{A.17}$$

$$\tilde{\sigma}^2 = \frac{\sigma^2}{3}(2 - \frac{1}{n^2}), \tag{A.18}$$

$$\tilde{r} = r - \frac{\sigma^2}{6}(1 + \frac{1}{n^2}). \tag{A.19}$$

Det följer att med dessa värden insatta och antagandet att $n$ är stort, så att termerna av ordning $n^{-2}$ kan försummas, fås formeln som används i Kemna och Vorst artikel för att variansreducera priset av en asiatisk option grundad på det aritmetiska medelvärdet [15]. Det vill säga

$$\Pi_{AK}(0) = e^{-rT}\left(e^{d_1}S(0)\Phi(d_2) - K\Phi(d_2 - \sigma\sqrt{\frac{T}{3}})\right), \tag{A.20}$$

med parametrarna $d_1$ och $d_2$ enligt

$$\begin{aligned} d_1 &= \frac{1}{2}(r - \frac{\sigma^2}{6})T \\ d_2 &= \frac{\log\frac{S(0)}{K} + \frac{1}{2}(r + \frac{\sigma^2}{6})T}{\sigma\sqrt{\frac{T}{3}}}. \end{aligned}$$

## A.4 Att dimensionsreducera Black-Scholes ekvation för en asiatisk option

Vi bevisar i detta avsnitt att man, via en variabelsubstitution, kan dimensionsreducera Black-Scholes ekvation för en asiatisk option,

$$\partial_t c + rx\partial_x c + x\partial_y c + \frac{1}{2}\sigma^2 x^2 \partial_x^2 c - rc = 0, \quad 0 \le t < T,\ x > 0,\ y > 0, \tag{A.21}$$

vilken uppfylls av prisfunktionen $c(t,x,y)$ med randvillkoret $c(T,x,y) = (y/T - K)_+$.

**Sats 7.** *Den 1+2-dimensionella PDE:n (A.21) kan reduceras till en 1+1-dimensionell PDE. Låt*

$$c(t,x,y) = xg(t,z), \quad z = \frac{1}{rT}(1 - e^{-r(T-t)}) + \frac{e^{-r(T-t)}}{T}\frac{y}{x} - e^{-r(T-t)}\frac{K}{x}.$$

*Då uppfyller $g$*

$$\partial_t g + \frac{1}{2}(\gamma(t) - z)^2 \partial_z^2 g = 0, \quad t \in [0,T],\ z \in \mathbb{R},$$

*där $\gamma(t) = \frac{1-e^{-r(T-t)}}{rT}$.*

***Bevis.*** Vi använder oss av relationen mellan $c$ och $g$ ovan för att utveckla varje term i (A.21) för sig. Till följd av kedje- och produktregeln har vi

$$\begin{aligned}
\partial_t c &= x\partial_t g = x\left(\frac{\partial g}{\partial t}\frac{\partial t}{\partial t} + \frac{\partial g}{\partial z}\frac{\partial z}{\partial t}\right) = x(\partial_t g + \partial_z g \partial_t z),\\
rx\partial_x c &= rx\partial_x(xg) = rx(g + x\partial_x g) = rxg + rx^2\partial_z g\partial_x z,\\
x\partial_y c &= x^2\partial_y g = x^2\partial_z g\partial_y z,\\
\frac{1}{2}\sigma^2 x^2\partial_x^2 c &= ... = \frac{1}{2}\sigma^2 x^2(2\partial_z g\partial_x z + x\partial_z^2 g(\partial_x z)^2 + x\partial_z g\partial_x^2 z),\\
-rc &= -rxg.
\end{aligned} \tag{A.22}$$

Vi tar nu fram nödvändiga uttryck för olika partiella derivator av $z$ för att kunna slutföra beviset med en förenkling. Vi får

$$\begin{aligned}
\partial_x z &= -\frac{e^{-r(t-t)}}{T}\frac{y}{x^2} + e^{-r(t-t)}\frac{K}{x^2},\\
\partial_x^2 z &= 2\frac{e^{-r(t-t)}}{T}\frac{y}{x^3} - 2e^{-r(t-t)}\frac{K}{x^3},\\
\partial_y z &= \frac{e^{-r(t-t)}}{T}\frac{1}{x},\\
\partial_t z &= -\frac{e^{-r(t-t)}}{rT} + \frac{re^{-r(t-t)}}{T}\frac{y}{x} - re^{-r(t-t)}\frac{K}{x}.
\end{aligned} \tag{A.23}$$

Vi noterar att alla utvecklade termer i vänsterledet av (A.21) innehåller ett $x > 0$ vilket vi förkortar bort. Vidare innehåller den andra termen en delterm som tar ut den sista termen $(-rg)$. Vi får då

$$\partial_t g + \partial_z g\partial_t z + \partial_z g\left(rx\partial_x z + x\partial_y z + \sigma^2 x\partial_x z + \frac{1}{2}\sigma^2 x^2\partial_x^2 z\right) + \frac{1}{2}\sigma^2 x^2\partial_z^2 g(\partial_x z)^2 = 0.$$

Vid insättning av de partiella derivatorna (A.23) i parentesen ovan får vi att denna är lika med $-\partial_t z$ och vi observerar att $(x\partial_x z)^2 = (\gamma(t) - z)^2$. Detta ger oss slutligen

$$\partial_t g + \frac{1}{2}(\gamma(t) - z)^2\partial_z^2 g = 0$$

och beviset är klart.

■

## A.5 Exakta lösningsformler

Här kommer de exakta lösningsformlerna för de undersökta optionerna att presenteras utan härledning, men den finns beskriven av Shreve [16]. De exakta lösningsformlerna används i figurer för att visa beteenden hos de olika optionerna och för att jämföra de lösningar som våra numeriska approximationer givit.

### Formler

Uttrycken för priset hos de optioner som presenteras nedan är besvärliga. För att de ska bli mer överskådliga är det lämpligt att införa följande funktioner:

$$\delta_+(\tau, s) = \frac{\log(s) + (r + \sigma^2/2)\tau}{\sigma\sqrt{\tau}},$$
$$\delta_-(\tau, s) = \frac{\log(s) + (r - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}.$$

### Barrär Upp-Ut-köpoption

Priset för en barriär Upp-Ut-Köpoption beskrivs när $t = 0$ enligt:

$$\begin{aligned}\Pi_{Upp-och-Ut-K}(0) =& S(0)\left[\Phi(\delta_+(T, \frac{S(0)}{K})) - \Phi(\delta_+(T, \frac{S(0)}{B}))\right] \\ &- e^{-r*T}K\left[\Phi(\delta_-(T, \frac{S(0)}{K})) - \Phi(\delta_-(T, \frac{S(0)}{B}))\right] \\ &- B(\frac{S(0)}{B})^{-\frac{2r}{\sigma^2}}\left[\Phi(\delta_+(T, \frac{B^2}{KS(0)})) - \Phi(\delta_+(T, \frac{B}{S(0)}))\right] \\ &+ e^{-rT}(\frac{S(0)}{B})^{-\frac{2r}{\sigma^2}+1}\left[\Phi(\delta_+(T, \frac{B^2}{KS(0)})) - \Phi(\delta_+(T, \frac{B}{S(0)}))\right].\end{aligned}$$

### Lookback-säljoption med rörligt lösenvärde

Priset för en lookback-säljoption med rörligt lösenvärde beskrivs när $t = 0$ enligt:

$$\begin{aligned}\Pi_{LP:F}(0) =& S(0)(1 + \frac{\sigma^2}{2r})\Phi(\delta_+(T, 1)) + e^{-rT}\Phi(-\delta_-(T, 1)) \\ &- \frac{\sigma^2}{2r}e^{-rT}\Phi(-\delta_-(T, 1)) - 1.\end{aligned}$$

# Referenser Appendix A

[14] Paul Glasserman. *Monte Carlo methods in financial engineering*. Applications of mathematics ; 53. Springer, New York, NY ; Berlin ; Heidelberg [u.a.], 2004.

[15] A. G. Z. Kemna and A. C. F. Vorst. A pricing method for options based on average asset values. *Journal of Banking & Finance*, 14(1):113–129, March 1990.

[16] S.E. Shreve. *Stochastic Calculus for Finance II: Continuous-Time Models*. Number v. 11 in Springer Finance Textbooks. Springer, 2004.

# B Utökad diskussion

## B.1 Monte Carlo-metoden

Monte Carlo metoden för prissättning av finansiella derivat presenterades för första gången i slutet av 70-talet av Phelim P. Boyle och har sedan dess varit ett värdefullt beräkningsverktyg för prissättningsproblem inom finansvärlden. Metoden innebär en processimulering som genererar avkastningar för underliggande värdepapper och under antagandet av riskneutralitet härleder optionsvärdet. Metoden inkluderar även två olika reduktionstekniker för varians som förbättrar dess effektivitet och resultatens noggrannhet. Detta gör Monte Carlo till en viktig metod för finansvärlden då företagens trovärdighet baseras på värdering av dess skulder [17].

Metoden beräknar volymen av en mängd där volymen kan tolkas som en sannolikhet. Man genererar slumpmässigt de möjliga utfall och väljer ut en del av dem som faller inom en given mängd som en uppskattning av dess volym. *De stora talens lagen* ser till att uppskattningen konvergerar mot ett korrekt svar vid ökande antal körningar.

Det är klokt att börja diskussionen i form av värdering av en finit integral av en godtycklig funktion $f : [0, 1] \to \mathbb{R}$, enligt

$$\int_0^1 f(x)dx \tag{B.1}$$

Integralen kan approximeras med hjälp av en Riemann summa

$$\int_0^1 f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(\frac{i}{n})$$

Anta sedan att $U$ är en likformigt fördelad slumpvariabel över $[0, 1]$. Då kommer $f(U)$ också vara slumpmässigt fördelad, med det förväntade värdet, som vi kan sätta till $I$ enligt följande

$$\mathbf{E}[f(U)] = \int_0^1 f(x)g(x)dx = I \tag{B.2}$$

där $g(x)$ är en täthetsfunktion för $U$ som är lika med 1 i området $[0, 1] \in \mathbb{R}$.

För att få en uppskattning av $I$, ett antal slumpmässiga $n$-punkter jämt fördelade över $[0, 1]$ är utvalda, för varje punkt $U_i$ beräknas $f(U_i)$ och därifrån fås det uppskattade medelvärdet $\widehat{I}$ och följande integraluppskattning

$$\int_0^1 f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(U_i) = \widehat{I} \tag{B.3}$$

där integralen B.1 är nu beräknad med hjälp av Monte Carlo metoden [20].

Monte Carlo metoden genererar noggranna approximationer, nästintill precisa värden för komplicerade funktioner som saknar enkla numeriska lösningar. *De stora talens lagen* säkerställer att summan $\widehat{I}$ i ekvationen B.3 konvergerar mot det förväntade värdet i B.2 för integralen för stora $n$ enligt

$$\lim_{n\to\infty}(\frac{1}{n}\sum_{i=1}^{n} f(U_i)) = I$$

Under antagandet om att $f$ är en kvadratisk integrerbar funktion, dvs

$$\int_{-\infty}^{\infty} |f(x)|^2 \, dx < \infty$$

kan vi sätta variansen $\sigma^2$ av funktionen $f$ till

$$\sigma_f^2 = \int_0^1 (f(x) - I)^2 dx$$

där $\sigma_f$ representerar standardavvikelsen av den undersökta funktionen, vilket resulterar i att felet för Monte Carlo beräkning blir normalfördelad enligt följande

$$\widehat{I} - I \in N\left(0, \frac{\sigma_f^2}{n}\right)$$

där medelvärdet är lika med 0 och variansen för tillräckligt stora $n$ kan approximeras till $\sigma_f^2/n$. Detta kan även påvisas av *den centrala gränsvärdessatsen* och ett ökande $n$ ger bättre resultat för approximationen. Då $I$ är en okänt funktion blir även $\sigma_f$ okänt, men med hjälp av standardavvikelsens definition för uppskattningen av $I$, kan följande uttryck för testvariansen tas fram

$$\hat{\sigma}_f^2 = \frac{1}{n-1}\sum_{i=1}^{n}(f(x_i) - \widehat{I}_n)^2 \tag{B.4}$$

Konfidensintervallet minimeras med ökat $n$, då standardavvikelsen av $\widehat{I}$ har formen $\sigma_f/\sqrt{n}$, vilket kännetecknar Monte Carlo metoden. Det finns ett antal olika tekniker för att åstadkomma minskningen av felet på uppskattningen, vilka ska presenteras härnäst.

**Antitetisk Variat**

Metoden reducerar variansen genom att undersöka den negativa korrelationen mellan de två uppskattningar och fungerar på så sätt att om de stokastiska slumpvariablerna definieras enligt $X \sim U(0,1)$ och $Y = 1 - X \sim U(0,1)$, där $U$ är en likformigt fördelad slumpvariabel inom området $[0,1]$ då kan variansen skrivas om enligt

$$\mathbf{Var}[X+Y] = \mathbf{Var}[X] + \mathbf{Var}[Y] + 2\mathbf{Cov}[X,Y]$$

Låt sedan $X_i$ ha samma fördelning som $X$. Då definieras *antitetisk variat*, $AV$ av följande

$$AV = \frac{1}{2n}(\sum_{i=1}^{n} X_i + \sum_{i=1}^{n} Y_i) = \frac{1}{n}\sum_{i=1}^{n}(\frac{X_i + Y_i}{2}) \tag{B.5}$$

där $AV$ är medelvärdet av $n$ antal oberoende observationer

$$(\frac{X_1+Y_1}{2}), (\frac{X_2+Y_2}{2}), ..., (\frac{X_n+Y_n}{2})$$

Inom prissättningen av finansiella derivat kan vi betrakta det vanliga neutrala optionsvärde med lösenpriset $K$ och aktiekursen $S$, som ges av $n$- replikationer som ( i det här fallet en köpoption som betecknas med $C$):

$$\widehat{C} = \frac{1}{n}\sum_{i=1}^{n} C_i \equiv \frac{1}{n}\sum_{i=1}^{n} e^{-rT} max\left\{0, S(T) - K\right\}$$

där slutliga aktiepriset under risk-neutralitet kan genereras av

$$S(T) = S(0)e^{(r-(1/2)\sigma^2)T + \sigma\sqrt{T}Z_i},\ i = 1, ..., n, \tag{B.6}$$

där $S(0)$ är dagens aktiepris, $r$ är den riskfria räntan, $\sigma$ är volatiliteten, $T$ är lösentiden och $Z_i$ är oberoende normalfördelade stickprov.

Antitetisk variat bygger på observationer att om $Z_i$ är normalfördelad, så är även $-Z_i$ det. Därför byts $Z_i$ i ekvationen B.6 för aktiepriset till $-Z_i$. På liknande sätt, blir varje

$$\widetilde{C_i} = e^{-rT} max\left\{0, \widetilde{S}(T) - K\right\}$$

till neutral estimator för optionspriset, därför fås

$$\widehat{C}_{AV} = \frac{1}{n}\sum_{i=1}^{n}\frac{C_i + \widetilde{C_i}}{2} \tag{B.7}$$

Anledningen till varför antitetisk variat är att föredra vid variansredusering, är att dess så kallade antitetiska par $(Z_i, -Z_i)$ är mer likformigt fördelade än de $2n$ oberoende stickprov i ekvationen (B.5). Detta resulterar i att medelvärdet över $(Z_i, -Z_i)$ är alltid lika med 0, medans medelvärdet över oberoende urval är alltid skilt från 0. Variansen för $C_i$ och $\widetilde{C_i}$ blir då

$$\mathbf{Var}\left[\frac{C_i + \widetilde{C_i}}{2}\right] = \frac{1}{2}(\mathbf{Var}[C_i] + \mathbf{Cov}[C_i, \widetilde{C_i}])$$

Köpoptionens värde bestämt med hjälp av antitetisk variat, $\widehat{C}_{AV}$ använder sig av dubbelt så många replikationer än $\widehat{C}$ så man måste ta hänsyn till skillnader i beräkningskraven. Om

$$\mathbf{Cov}[C_i, \widetilde{C_i}] < 0 \Rightarrow \mathbf{Var}[\frac{C_i + \widetilde{C}_i}{2}] < \frac{1}{2}\mathbf{Var}[C_i]$$

kommer detta att ge en bättre uppskattning av variansen än en oberoende uppskattning och därmed är variansen reducerad, vilket även är ett villkor för antitetisk variat. [21]

**kontrollvariat**

I Monte Carlo simulationerna vill vi estimera $\Pi = E[X(t)]$ där $X$ är en stokastisk variabel. Den huvudsakliga idéen med ett kontrollvariat, *control variate*, är nu att anta att vi har en annan stokastisk variabel $Y$ där $E[Y] = \mu_Y$ är känt. Vi kan då konstruera ett annat oberoende estimat för $\Pi$ igenom att istället simulera följande i en Monte Carlo metod[18]:

$$X + c(Y - \mu_Y) \tag{B.8}$$

Det inses lätt att (B.8) kan användas för att uppskatta $\Pi$ då vi har att

$$\mathbb{E}[X + c(Y - \mu_Y)] = \mathbb{E}[X] + c(\mathbb{E}[Y] - \mu_Y) = \Pi$$

Vidare vill vi välja c så att variansen för (B.8) minimeras.

**Lemma 1.** $c^* = -\dfrac{Cov(X,Y)}{Var(Y)}$ *minimerar variansen till kontrollvariatet B.8.*

***Bevis.*** Vi har att

$$Var(X + c(Y - \mu_Y)) = Var(X + cY) = Var(X) + c^2 Var(Y) + 2cCov(X,Y)$$

$$=> \frac{\partial}{\partial c}(Var(X + c(Y - \mu_Y)) = 2(cVar(Y) + Cov(X,Y))$$

Vi har då att

$$c = \frac{-Cov(X,Y)}{Var(Y)}$$

med

$$\frac{\partial^2}{\partial c^2}(Var(X + c(Y - \mu_Y))) = Var(Y) > 0$$

■

Variansen för (B.8) blir därav

$$Var(X + c(Y - \mu_Y)) = Var(X) - \frac{Cov(X,Y)^2}{Var(Y)} \tag{B.9}$$

I praktiken kommer X och Y bestämmas genom medelvärdet av Monte Carlo-simulationerna och vi erhåller då kontrollvariat

$$\bar{X} + c^*(\bar{Y} - \mu_Y) \tag{B.10}$$

Variansen för (B.10) erhålls då igenom

$$Var(\frac{1}{N}\sum_{i=1}^{N} X_i + c^*(\frac{1}{N}\sum_{i=1}^{N} Y_i - \mu_Y)) = \frac{1}{(N)^2}\sum_{i=1}^{N} Var(X_i + c^*(Y_i - \mu_Y)) =$$
$$= \frac{1}{N}(Var(X) - \frac{Cov(X,Y)^2}{Var(Y)})$$

Termen $Cov(X,Y)$ samt $Var(Y)$ kan i de flesta fallen inte beräknas analytiskt utan måste estimeras på något sätt. Ett sätt att göra detta på är att använda stickprovs kovariansen för ens Monte Carlo simuleringar

$$Cov(X,Y)^* = \frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y})$$
$$Var(Y)^* = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \bar{Y})^2$$

**Utvidgning till multipla kontrollvariat**

kontrollvariatet (B.8) kan utvidgas till att innehålla n stycken kontrollvariat på följande sätt

$$X + A^T(Y - \mu_Y) \tag{B.11}$$

Där Y är en vektor med n stycken kontrollvariat, $\mu_Y$ är en vektor bestående av korresponderande analytiska väntevärden för elementen i Y och A är en viktningsvektor [19]. Vi väljer att sätta $Z = Y - \mu_Y$ för lättare notation.

**Sats 8.** *Om matrisen $\Sigma_Z$ är positivt definit minimeras variansen för (B.11) av*

$$A = -(\Sigma_Z)^{-1}\Sigma_{X,Z} \tag{B.12}$$

*där Z är vektorn $Z = Y - \mu_y$, $\Sigma_Z$ är kovariansmatrisen för Z och $\Sigma_{X,Z}$ är en vektor vars element defineras som $\Sigma_{X,Z_i} = Cov(X, Z_i)$. För detta specifika A gäller även att*

$$Var(X + A(Y - \mu_y)) = Var(X)(1 - R), R = \frac{\Sigma_{X,Z}^T \Sigma Z^{-1}\Sigma_{X,Z}}{Var(X)} \tag{B.13}$$

***Bevis.***

$$g(A) = Var(X + A^T(Y - \mu_Y)) = Var(X) + 2Cov(X, A^T(Y - \mu_Y)) + E[A^T(Y - \mu_Y)]$$

Vi deriverar nu med avseende på $a_i$

$$\frac{\partial g}{\partial a_i} = 2(Cov(X, (Y_i - \mu_{Y_i})) + \sum_{j=1}^{n} a_j E[(Y_i - \mu_{Y_i})(Y_j - \mu_j)]))$$

Vi sätter nu $Z_i = Y - \mu_Y$ och söker $\frac{\partial g}{\partial a_i} = 0$, för $i = 1...n$, vilket ger oss följande matrisekvation

$$\Sigma_{X,Z} + A^T\Sigma_Z = 0$$
$$=> A = (\Sigma_Z)^{-1}\Sigma_{X,Z}$$

För att försäkra oss om att detta är ett minimum noterar vi att Hessianen av $g(A)$ är $H_{g(A)} = \Sigma_Z$, vilket vi antagit är positivt definit.

■

**Multipla kontrollvariat för lookback**

Det multipla kontrollvariatet (B.11) lämpar sig bra för lookback-optioner. Motiveringen för att använda detta på Lookback med fast lösenpris är att dessa kan modelleras som maximum av en mängd europeiska köp eller säljoptioner, alla med samma lösenpris men med olika lösendagar fördelade över intervallet $[0, T]$ där $T$ är lösendagen för lookback optionen. Detta kontrollvariat är även användbart i fallet med rörligt lösenpris. Vi behöver nu försäkra som om att kovariansmatrisen är positivt definit.

En kovariansmatris, $\Sigma_Z$, är alltid positivt semidefinit. Antag nu att matrisen är singulär och icke-inverterbar, det vill säga $\exists a : a^T \Sigma_Z a = 0$ där minst ett element i $a$ är nollskilt. Notera att

$$Var(aZ) = \sum_{i=1}^{n}(a_i Z_i + \sum_{j=1}^{n} a_i a_j Cov(Z_i, Z_j)) = a^T \Sigma_Z a$$

Vilket ger att $Var(a(Y - \mu_Y)) = Var(aZ) = 0$ för något a, $aY$ är då konstant lika med $a\mu_Y$. Detta är inte fallet för den valda kontrollvariat vektorn bestående av europeiska standard optioner då denna beror på en underliggande tillgång som modelleras som en brownsk rörelse. $\Sigma_Z$ är alltså positivt definit för vårt val av $Y$.

Vi kommer inte använda oss av (B.11) i denna rapport. Detta på grund utav valet av implementation för den algoritm som simulerar den underliggande tillgångens väg gör att det mutlipla kontrollvariatet endast fungerar för tillgångar vars startvärde är så kallat *in the money* eller *close to the money*, det vill säga startvärdet är förhållandevis nära eller större än lösenpriset. Om nu lösenvärdet är mycket större än startvärdet riskerar matrisen $\Sigma_Z$ att bli illa konditionerad. En lösning på detta kan vara att implementera fördelningen av kontrollvariaten på ett annat sätt, ej likformigt, detta har inte gjorts i rapporten utan kan vara ett ämne för vidare studier.

## B.2 Tillämpning av Crank-Nicolson-metoden på Black-Scholes ekvation

En finit differensmetod (FDM) är en numeriska metod för att lösa en partiella differentialekvation (PDE). Detta görs genom att approximera PDE:n med differensekvationer, där man låter finita differenser approximerar derivatorna. Crank-Nicolson-metoden, en FDM som är numeriskt stabil och implicit i tiden, utvecklades av John Crank och Phyllis Nicolson och publicerades 1947. Dess användning för prissättning av optioner blev aktuell först när Black-Scholes-modellen introducerades 1973.

### Approximation av derivator

Crank-Nicolson-metoden är ett mellanting av Eulers framåt- och bakåtmetod, vilket leder till en högre noggrannhet. Tidsderivatan approximeras både framåt och bakåt i tiden, medan rumsderivator hålls centrerade. Crank-Nicolson-metoden beskrivs enligt

$$v(x, t + \Delta t) = \frac{1}{2} v_b(x, t + \Delta t) + \frac{1}{2} v_f(x, t + \Delta t), \tag{B.14}$$

där $v_b(x, t+\Delta t)$ och $v_f(x, t+\Delta t)$ beräknas enligt Eulers bakåt- och framåtmetod respektive.

Den centrerade approximationen av rumsderivator är densamma för båda Eulers metoder. Första ordningens rumsderivator approximeras enligt

$$\partial_x v \approx \frac{v(t, x + \Delta x) - v(t, x - \Delta x)}{2\Delta x}, \tag{B.15}$$

och andra ordningens enligt

$$\partial_x^2 v \approx \frac{v(t, x + \Delta x) - 2v(t, x) + v(t, x - \Delta x)}{\Delta x^2}. \tag{B.16}$$

Det är för tidsderivatan metoderna skiljer sig åt. I Eulers framåtmetod approximeras tidsderivatan enligt

$$\partial_t v \approx \frac{v(t + \Delta t, x) - v(t, x)}{\Delta t}, \tag{B.17}$$

medan den i Eulers bakåtmetod approximeras enligt

$$\partial_t v \approx \frac{v(t, x) - v(t - \Delta t, x)}{\Delta t}. \tag{B.18}$$

Crank-Nicolson-metoden kommer här att appliceras för en $1 + 1$-dimensionell PDE. Det vill säga en rumsvariabel $x$ och en tidsvariabel $t$, gällande i ett område $(0,\ T) \times (0,\ X)$, med tillhörande slut- och randvillkor. Vi kan då diskretisera tidsintervallet i $n + 1$ tidspunkter enligt

$$\{t_i\}_{i=0}^{n} \quad \text{där } t_i = i\frac{T}{n}. \tag{B.19}$$

På liknande sätt diskretiserar vi rumsintervallet

$$\{x_j\}_{j=0}^{m} \quad \text{där } x_j = j\frac{X}{m}. \tag{B.20}$$

Då fås följaktligen att $\Delta t = t_{i+1} - t_i$ och $\Delta x = x_{j+1} - x_j$.

### Applicering på Black-Scholes ekvation

Den PDE som här undersöks är Black-Scholes ekvation, vilken finns beskriven tidigare i rapporten, men som upprepas här av bekvämlighetsskäl. Funktionen $u(x, t)$ representerar prissättningsfunktionen för eftersökt option och uppfyller Black-Scholes ekvation:

$$\begin{cases} \partial_t u + rx\partial_x u + \frac{1}{2}\sigma^2 x^2 \partial_x^2 u = ru & t \in (0, T),\ x \in (0, X) \\ u(T, x) = g(x) & x \in (0, X), \end{cases} \tag{B.21}$$

där $g(x)$ är lösenvärdesfunktionen hos optionen. Priset för optionen bestäms enligt $\Pi_Y(t) = u(t, S(t))$, där $Y = g(S(T))$ är lösenvärdet.

Genom att omdefiniera prissättningsfunktionen som $v(t,x) = u(T-t,x)$, och på så sätt invertera tiden i (B.21), fås istället motsvarande PDE med begynnelsevillkor:

$$\begin{cases} -\partial_t v + rx\partial_x v + \frac{1}{2}\sigma^2 x^2 \partial_x^2 v = rv & t \in (0,T),\ x \in (0,X) \\ v(0,x) = g(x) & x \in (0,X). \end{cases} \tag{B.22}$$

Med uppdelningen av tids- och rumsintervallen enligt (B.19) och (B.20) kan vi i de diskreta punkterna $(t_i,\ x_j)$ benämna lösningen $v_{i,j} = v(t_i, x_j)$. $v_{0,j} = g(x_j)$ fås från begynnelsevillkoret. Eulers framåtmetod ger för PDE:n (B.22), med approximationerna för rumsderivatan (B.15)-(B.16) och för tidsderivatan (B.17), följande:

$$v_{i+1,j} = v_{i,j}(1 - r\Delta t) + \frac{1}{2} d\left[ rx_j(v_{i,j+1} - v_{i,j-1})\Delta x + \sigma^2 x_j^2 (v_{i,j+1} - 2v_{i,j} + v_{i,j-1}) \right]. \tag{B.23}$$

Eulers bakåtmetod ger för motsvarande, med tidsderivatan enligt (B.18), istället:

$$\begin{aligned} v_{i+1,j} = v_{i,j} + \frac{1}{2} d\big[ & rx_j(v_{i+1,j+1} - v_{i+1,j-1})\Delta x + \\ & + \sigma^2 x_j^2 (v_{i+1,j+1} - 2v_{i+1,j} + v_{i+1,j-1}) \big] - r\Delta t v_{i+1,j}. \end{aligned} \tag{B.24}$$

Här har $d = \frac{\Delta t}{\Delta x^2}$ introducerats för enkelhetens skull.

Med dessa resultat insatta i (B.14) fås efter algebraiska omskrivningar att

$$\begin{aligned} & v_{i+1,j}(1 + \frac{1}{2} r\Delta t) - \frac{1}{4} d\left[ rx_j(v_{i+1,j+1} - v_{i+1,j-1})\Delta x + \sigma^2 x_j^2 (v_{i+1,j+1} - 2v_{i+1,j} + v_{i+1,j-1}) \right] = \\ & = v_{i,j}(1 - \frac{1}{2} r\Delta t) + \frac{1}{4} d\left[ rx_j(v_{i,j+1} - v_{i,j-1})\Delta x + \sigma^2 x_j^2 (v_{i,j+1} - 2v_{i,j} + v_{i,j-1}) \right], \end{aligned}$$

vilket enklare beskrivs på matrisform

$$Av_{i+1} = Bv_i, \quad \text{där } v_i = \begin{bmatrix} v_{i,0} \\ v_{i,1} \\ \vdots \\ v_{i,m} \end{bmatrix}. \tag{B.25}$$

$A$ och $B$ är här $(m+1) \times (m+1)$ tridiagonala matriser. $A$ har elementen

$$\begin{cases} A_{k,k} = 1 + \frac{1}{2} r\Delta t + \frac{1}{2} d\sigma^2 x_k^2 \\ A_{k,k+1} = -\frac{1}{4} drx_k \Delta x - \frac{1}{4} d\sigma^2 x_k^2 \qquad k = 1,\ \ldots\ , m-1 \\ A_{k,k-1} = \frac{1}{4} drx_k \Delta x - \frac{1}{4} d\sigma^2 x_k^2. \end{cases} \tag{B.26}$$

$B$'s element är snarlika, men skiljer sig något. De är

$$\begin{cases} B_{k,k} = 1 - \frac{1}{2} r\Delta t - \frac{1}{2} d\sigma^2 x_k^2 \\ B_{k,k+1} = \frac{1}{4} drx_k \Delta x + \frac{1}{4} d\sigma^2 x_k^2 \qquad k = 1,\ \ldots\ , m-1 \\ B_{k,k-1} = -\frac{1}{4} drx_k \Delta x + \frac{1}{4} d\sigma^2 x_k^2. \end{cases} \tag{B.27}$$

Den första och sista raden i dessa matriser kan specificeras för att uppfylla eventuella randvillkor. Alltså då $x = 0$ och $x = X$. Detta kommer vara tydligt när appliceringar för de undersökta optionerna presenteras. Vi ser dessutom att systemet, tillsammans med randvillkoren, kommer gå att lösa eftersom vektorn $v_0$ är känd från begynnelsevillkoret.

### Applicering för europeiska optioner

För de europeiska standardoptionerna har vi en option vars prissättningsfunktion $v(t,x)$ uppfyller (B.22). Randvillkoren för köpoptionen är:

$$\begin{cases} v(t,0) = 0 \\ v(t,x) = x - Ke^{-rt}, \quad \text{då } x \to \infty, \end{cases}$$

och begynnelsevärdet: $v(0,x) = g(x) = (x-K)_+$. För säljoptionen blir de istället:

$$\begin{cases} v(t,0) = Ke^{-rt} \\ v(t,x) = 0, \quad \text{då } x \to \infty, \end{cases}$$

med begynnelsevärdet: $v(0,x) = g(x) = (K-x)_+$.

Vid implementering av metoden är det viktigt att låta $X$ vara tillräckligt stort för att representera $\infty$ för valda parametrar. För att senare upprätthålla dessa randvillkor för varje tidssteg måste rad 0 och rad $m$ i både matris $A$ och $B$ specifieras. Här låter vi

$$A_{0,0} = B_{0,0} = 1 \quad A_{m,m} = B_{m,m} = 1,$$

och uppdaterar randvillkoret för $v(t,0)$ efter varje tidsiteration.

### Applicering för barriäroptioner

För Barriär Upp-Ut-köpoption är appliceringen väldigt lik den europeiska köpoptionen. Skillnaden blir i randvillkoren, där på grund av barriären, optionen blir utslagen. Ekvationen betraktas därmed endast då $x \in (0, B)$. Randvillkoren blir således:

$$\begin{cases} v(t,0) = 0 \\ v(t,B) = 0. \end{cases}$$

För att upprätthålla dessa randvillkor för varje tidssteg måste även här rad 0 och rad $m$ i både matris $A$ och $B$ specifieras. Här låter vi

$$A_{0,0} = B_{0,0} = 1 \quad A_{m,m} = B_{m,m} = 1,$$

Här behöver det dock inte korrigeras för randvillkoren, då dessa automatiskt uppfylls i varje tidsiteration.

### Applicering för lookback-optioner

För lookback-optionerna fungerar inte samma tankesätt som använts för de europeiska och barriär-optionerna. Detta eftersom prissättningsfunktionen här beror på ytterligare en variabel. PDE:n som behöver lösas är alltså 1+2-dimensionell. Med $Y(t) = \max_{0 \le u \le t} S(u)$ visar Shreve att PDE:n kan dimensionsreduceras genom $\Pi(t) = v(t, S(t), Y(t))$ och $v(t,x,y) = yg(t,z)$, där $z = \frac{x}{y}$ [22]. Den $1+1$-dimensionella PDE:n som erhålls lyder:

$$\begin{cases} \partial_t g + rz\partial_z g + \frac{1}{2}\sigma^2 z^2 \partial_z^2 g = rg & t \in (0,T),\ z \in (0,1) \\ g(T,z) = 1 - z & z \in (0,1). \end{cases}$$

Randvillkoren i detta fall är:

$$\begin{aligned} u(t,0) &= e^{-r(T-t)} \\ u(t,1) &= \partial_t u(t,1). \end{aligned}$$

På liknande sätt som för appliceringen hos europeiska standardoptioner kan matriselementen i (B.26) och (B.27) härledas, men med rumsvariabeln $z$ istället. Det som behöver korrigeras för är randvillkoren, vilket innebär rad 0 och $m$ i de båda matriserna. Här får vi:

$$A_{0,0} = e^{-r\Delta t} \quad B_{0,0} = 1 \quad A_{m,m-1} = -\frac{1}{1-\Delta z},$$

och ingen korrigering behövs för att upprätthålla randvillkoren i varje tidsiteration.

**Applicering för asiatiska optioner**

För de asiatiska optionerna, likt lookback-optioner, krävs ett större arbete jämfört med de tidigare optionerna. Detta återigen eftersom PDE:n som behöver lösas är $1+2$-dimensionell. Om vi här låter $Y(t) = \int_0^t S(u)du$ och dimensionsreducerar till en $1+1$-dimensionell PDE genom $\Pi_A(t) = v(t, S(t), Y(t))$ och $v(t,x,y) = xg(t,z)$, där

$$z = \frac{1}{rT}(1 - e^{-r(T-t)}) + \frac{e^{-r(T-t)}}{T}\frac{y}{x} - e^{-r(T-t)}\frac{K}{x}, \tag{B.28}$$

fås en annorlunda ekvation och därmed annorlunda matriselement. Den ekvation $g$ då satisfierar lyder:

$$\begin{cases} \partial_t g + \frac{\sigma^2}{2}(\gamma(t) - z)^2 \partial_{zz} g = 0, \quad t \in [0,T],\ z \in \mathbb{R} \\ g(T,z) = (z)_+, \end{cases} \\ \text{där } \gamma(t) = \frac{1 - e^{-r(T-t)}}{rT}. \tag{B.29}$$

Denna ekvation härleds i Appendix A.4 genom variabelsubstitutionen (B.28).

Randvillkor för denna ekvation kan specificeras enligt: $\lim_{z\to-\infty} g(t,z) = \lim_{z\to\infty}(g(t,z) - z) = 0$. Dessa härleds tillsammans med ekvationen av Shreve. [22]

Här genomgås samma steg som för Black-Scholes ekvation. Invertera tiden i (B.29) och applicera sedan Eulers framåt- och bakåtmetod på PDE:n som erhålls. Ur detta erhålls från Eulers framåtmetod

$$g_{i+1,j} = g_{i,j} + \frac{\sigma^2}{2}\frac{\Delta t}{\Delta z^2}(\gamma(t_i) - z_j)^2(g_{i,j+1} - 2g_{i,j} + g_{i,j-1}), \tag{B.30}$$

och från Eulers bakåtmetod

$$g_{i+1,j} = g_{i,j} + \frac{\sigma^2}{2}\frac{\Delta t}{\Delta z^2}(\gamma(t_i) - z_j)^2(g_{i+1,j+1} - 2g_{i+1,j} + g_{i+1,j-1}). \tag{B.31}$$

Om vi använder (B.30) och (B.31) i (B.14) och sedan följer motsvarande resonemang som för appliceringen på Black-Scholes ekvation fås systemet

$$Ag_{i+1} = Bg_i, \quad \text{där } g_i = \begin{bmatrix} g_{i,0} \\ g_{i,1} \\ \vdots \\ g_{i,m} \end{bmatrix}.$$

Likt då är även dessa matriser tridiagonala. $A$'s element är

$$\begin{cases} A_{k,k} = 1 + \frac{\sigma^2}{2}\frac{\Delta t}{\Delta z^2}(\gamma(t_{i+1}) - z_j)^2 \\ A_{k,k+1} = -\frac{\sigma^2}{4}\frac{\Delta t}{\Delta z^2}(\gamma(t_{i+1}) - z_j)^2 \qquad k = 1,\ \ldots\ , m-1 \\ A_{k,k-1} = -\frac{\sigma^2}{4}\frac{\Delta t}{\Delta z^2}(\gamma(t_{i+1}) - z_j)^2, \end{cases} \tag{B.32}$$

medan $B$'s bestäms enligt

$$\begin{cases} B_{k,k} = 1 - \frac{\sigma^2}{2}\frac{\Delta t}{\Delta z^2}(\gamma(t_i) - z_j)^2 \\ B_{k,k+1} = \frac{\sigma^2}{4}\frac{\Delta t}{\Delta z^2}(\gamma(t_i) - z_j)^2 \qquad k = 1,\ \ldots\ , m-1 \\ B_{k,k-1} = \frac{\sigma^2}{4}\frac{\Delta t}{\Delta z^2}(\gamma(t_i) - z_j)^2. \end{cases} \tag{B.33}$$

Här är det återigen viktigt att låta ändarna på intervallet för $z \in (z_{min}, z_{max})$ vara tillräckligt stora för att $z_{min}$ och $z_{max}$ i sammanhanget ska kunna representera $\pm\infty$. För att uppfylla randvillkoren sätts dessutom följande:

$$A_{0,0} = A_{m,m} = 1 \quad B_{0,0} = B_{m,m} = 1.$$

Inte heller här behöver randvillkoren korrigeras, eftersom de uppfylls automatiskt.

## B.3 Monte Carlo-metodens feldiskussion

I rapporten har ett återkommande fel upptäckts för implementationen av Monte Carlo gällande barriär- och lookback-optioner. Denna appendixdel täcker en utförligare diskussion av felet och som inte är central för rapporten men kan vara av intresse för läsaren.

### Implementeringen

Grundtanken med vår Monte Carlo-implementation är att Monte Carlo-approximationen konvergerar mot det verkliga värdet när vi låter antalet simuleringar gå mot oändligheten

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} M_i(n,t) = \Pi(t) \tag{B.34}$$

där $M_i(n,t)$ är den i:th Monte Carlo-simuleringen med $n$ antal tidspunkter och $\Pi(t)$ är det verkliga värdet. Gränsvärdet (B.34) gäller för asiatiska optioner men för barriär- och lookback-optioner tycks detta inte alltid gälla i respektive resultatsdel. Detta grundar sig i att dessa optioner är starkt beroende av extremvärden.

När vi approximerar en Brownsk rörelse diskret går vi miste av precision då vi alltid har ett ändligt antal tidspunkter. För ett interval mellan två tidspunkter $(t_i, t_i + 1]$ är sannolikheten att den Brownska rörelsen antagit ett lokalt extremvärde i ändpunkterna 0. Monte Carlo-metoden kommer därför aldrig kunna få med det faktiska extremvärdet. Värdet som metoden konvergerar mot blir alltså konsekvent mindre eller större än det verkliga värdet beroende på optionens lösenvärdesfunktion.

$$max(S(t_i), S(t_{i+1})) < max_{t_i<t\leq t_{i+1}} S(t) \tag{B.35}$$

Det inses då att skillnaden mellan Monte Carlo-extremvärdet och det faktiska extremvärdet är omväntproptionell mot antalet tidspunkter, n. Kravet på implementeringen av Monte Carlo-metoden blir då betydligt mer krävande. Både antalet simuleringar $N$ och antalet tidspunkter $n$ måste gå mot oändligheten för att det diskreta lösenvärdet för optionen skall konvergera mot det faktiska kontinuerliga värdet(B.36). Detta är icke önskvärt då tidskomplexiteten för algoritmen är $O(Nn)$.

$$\lim_{N\to\infty} \lim_{n\to\infty} \frac{1}{N} \sum_{i=1}^{N} M_i(n,t) = \Pi(t) \tag{B.36}$$

Dessa teoretiska observationer har bekräftats i resultatsdelarna för lookback- och barriäroptionen. Framförallt har vi sett att de relativa felen hos båda optionerna är för stora för att användas om beräkningstiderna ska begränsas rimligt. Beräkningskapaciteten konstateras bli alldeles för begränsande då den här implementeringen ska beräkna ett tillräckligt exakt pris. Vanligtvis talas det om ett fel på ett par cent vid den här typen av approximativ prissättning.

### Korrigerande tillägg

Den implementering som har gjorts har alltså stora brister utifrån ovannämnda observationer som både påvisats i resultatdelarna och diskuterats teoretiskt. Ett förslag på tillvägagångsätt för att lösa detta problem presenteras här nedan med en Brownsk bro, där vi hämtat definitionerna från Steven E .Shreves bok *Stochastic Calculus for finance II* [22].

**Definition 1.** *Antag att $W(t)$ är en Brownsk rörelse. För en fix tidpunkt $T$ definierar vi en Brownsk bro, $X(t)$, från 0 till 0 på tidsintervallet $[0,T]$ som*

$$X^{0\to 0}(t) = W(t) - \frac{t}{T} W(T). \tag{B.37}$$

Vidare använder vi definition 1 för att definera följande

**Definition 2.** *Låt $a, b \in \mathbb{R}$ vi definierar då en Brownsk bro från a till b på intervallet $[0, T]$ på följande sätt*

$$X^{a\to b} = a + \frac{(b-a)t}{T} + X^{0\to 0}(t). \tag{B.38}$$

*Där $X^{0\to 0}(t)$ är en Brownsk bro från 0 till 0 på intervallet $[0, T]$*

Samplingen av den underliggande aktien $S(t)$ ger diskreta punkter $S(t_i)$ med Monte Carlo-metoden. Den Brownska rörelsen som metoden simulerat reduceras till Brownska broar mellan de diskreta tidspunkterna $t_i$. Vidare introducerar vi följande stokastiska variabel

$$M_i^{a\to b} = max_{t_i<t\leq t_{i+1}} X^{a\to b}(t) \tag{B.39}$$

Stående på Shreves axlar noterar vi att täthetsfunktionen för (B.39) kan uttryckas på följande sätt [22]

$$f_{M_i^{a\to b}}(x) = \frac{2(2x-a-b)}{T} e^{-\frac{2(x-a)(x-b)}{T}}, x > max(a, b) \tag{B.40}$$

Den kumulativa sannolikhetsfunktion kan då enkelt tas fram till att bli

$$F_{M_i^{a\to b}}(y) = 1 - e^{-\frac{2(a-y)(b-y)}{T}} \tag{B.41}$$

Istället för att använda maximum från den diskreta mängden $\{S(t_i)\}_{i=0}^{n+1}$ kan nu maximum från $\{M_i^{S(t_i)\to S(t_{i+1})}\}_{i=0}^{n}$ användas. En mängd $\{M_i^{S(t_i)\to S(t_{i+1})}\}_{i=0}^{n}$ kan till exempel generas i en Monte Carlo-implementation med hjälp av en invers-transformmetod på (B.41).

Vi har att $max(\{M_i^{S(t_i)\to S(t_{i+1})}\}_{i=0}^{n}) > max(\{S(t_i)\}_{i=0}^{n+1})$. Fördelen med detta är att en implementation skulle modellera det faktiska extremvärdet med en mycket högre precision. För resultat och vidare läsning se Basileios Papatheodorous rapport [23].

Notera att för den stokastiska variabeln

$$m_i^{a\to b} = min_{t_i<t\leq t_{i+1}} X^{a\to b}(t) \tag{B.42}$$

har vi istället att täthetsfunktionen ser ut på följande sätt

$$f_{m_i^{a\to b}}(x) = \frac{2(a+b-2x)}{T} e^{-\frac{2(a-x)(b-x)}{T}}, x < min(a, b). \tag{B.43}$$

En implementation baserat på ovanstående hade varit av intresse om mer tid funnits till projektet. Detta kan vara ett ämne för vidare studier.

# Referenser Appendix B


[17] Phelim P. Boyle. Options: A Monte Carlo approach. *Journal of Financial Economics*, 4(3):323–338, May 1977.

[18] S.M. Ross. *Simulation*. Academic Press, 2013.

[19] Peter E. Glynn and Roberto Szechtman. Some new perspectives on the method of control variate. *Monte Carlo and Quasi-Monte Carlo Methods 2000*.

[20] Paul Glasserman. *Monte Carlo methods in financial engineering*. Applications of mathematics ; 53. Springer, New York, NY ; Berlin ; Heidelberg [u.a.], 2004.

[21] Phelim Boyle, Mark Broadie, and Paul Glasserman. Monte Carlo methods for security pricing. *Journal of Economic Dynamics and Control*, 21(8-9):1267–1321, June 1997.

[22] S.E. Shreve. *Stochastic Calculus for Finance II: Continuous-Time Models*. Number v. 11 in Springer Finance Textbooks. Springer, 2004.

[23] B. Papatheodorou. *Enhanced Monte Carlo Methods for Pricing and Hedging Exotic Options*. University of Oxford, 2005.

[24] W.F. Ames. *Numerical methods for partial differential equations*. Computer science and applied mathematics. Academic Press, 1977.

# C Programkod - Matlab

Här presenteras delar av den kod som har producerats under projektets gång för att prissätta de undersökta optionerna. Endast koden i MATLAB kommer presenteras, eftersom motsvarande i `C++` är snarlik och enkelt kan reproduceras utifrån denna kod.

## C.1 Monte Carlo-metoden

**Underliggande tillgångens värdeutveckling**

```
% StockPath.m
%
% Function for simulating stock paths. Returns a matrix, where the columns
% consist of simulated stock paths. Yields twice the number of requested
% simulations, since Antithetic variates are used.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of subintervals with respect to time.
% - N > 0, Integer. Number of simulated paths.

function Path = StockPath(S0,sig,r,T,n,N)
    h=T/n;
    w=randn(N,n);
    W=[-w;w];
    q=ones(2*N,n);
    Path=S0*exp((r-sig^2/2)*h.*cumsum(q')+sig*sqrt(h)*cumsum(W'));
    Path = [S0*ones(1,2*N); Path];
end
```

**Europeisk standard köpoption**

```
% MonteCarlo_EC.m
%
% Computes the price of an European vanilla call option using Monte Carlo
% simulations. Returns a vector consisting of the price as its first
% element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_EC(S0,sig,r,K,T,n,N)
    stockPath=StockPath(S0,sig,r,T,n,N);
    payOff=max(0,stockPath(end,:)-K);
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

**Europeisk standard säljoption**

```
% MonteCarlo_EP.m
%
% Computes the price of an European vanilla put option using Monte Carlo
% simulations. Returns a vector consisting of the price as its first
% element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_EP(S0,sig,r,K,T,n,N)
    stockPath=StockPath(S0,sig,r,T,n,N);
    payOff=max(0,K-stockPath(end,:));
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

**Asiatisk köpoption - Aritmetiskt medelvärde; Utan kontrollvariat**

```
% MonteCarlo_AC_crude.m
%
% Computes the price of an Asian call option using Monte Carlo
% simulations. Returns a vector consisting of the price as its first
% element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_AC_crude(S0,sig,r,K,T,n,N)
    stockPath=StockPath(S0,sig,r,T,n,N);
    payOff=max(0,mean(stockPath)-K);
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

**Asiatisk köpoption - Aritmetiskt medelvärde; Med kontrollvariat**

```
% MonteCarlo_AC.m
%
% Computes the price of an Asian call option using Monte Carlo
% simulations. Here the price of the geometric Asian call is used as a
% control variate. Returns a vector consisting of the price as its first
% element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
```

```
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_AC(S0,sig,r,K,T,n,N)
    % Deterministic value for control variate
    detValueCV = ClosedFormula_AC_geo(S0,sig,r,K,T);

    stockPath=StockPath(S0,sig,r,T,n,N);
    payOff=max(0,mean(stockPath)-K);
    payOffCV = max(geomean(stockPath)-K,0); %Pay-off control variate
    price = exp(-r*T)*(mean(payOff)-mean(payOffCV) + detValueCV);
    deviation = std(payOff-payOffCV+detValueCV)/sqrt(2*N);
end
```

**Asiatisk säljoption - Aritmetiskt medelvärde; Utan kontrollvariat**

```
% MonteCarlo_AP_crude.m
%
% Computes the price of an Asian put option using Monte Carlo
% simulations. Returns a vector consisting of the price as its first
% element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_AP_crude(S0,sig,r,K,T,n,N)
    stockPath = StockPath(S0,sig,r,T,n,N);
    payOff = max(0,K-mean(stockPath));
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

**Asiatisk säljoption - Aritmetiskt medelvärde; Med kontrollvariat**

```
% MonteCarlo_AP.m
%
% Computes the price of an Asian put option using Monte Carlo
% simulations. Here the price of the geometric Asian put is used as a
% control variate. Returns a vector consisting of the price as its first
% element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.
```

```
function [price, deviation] = MonteCarlo_AP(S0,sig,r,K,T,n,N)
    % Deterministic value for control variate
    detValueCV = ClosedFormula_AP_geo(S0,sig,r,K,T);

    stockPath=StockPath(S0,sig,r,T,n,N);
    payOff=max(0,K-mean(stockPath));
    payOffCV = max(K-geomean(stockPath),0); %Pay-off control variate
    price = exp(-r*T)*(mean(payOff-payOffCV) + detValueCV);
    deviation = std(payOff-payOffCV + detValueCV)/sqrt(2*N);
end
```

**Lookback-köpoption - Fast lösenpris; Utan kontrollvariat**

```
% MonteCarlo_LC_fixed_crude.m
%
% Computes the price of an Lookback call option with fixed strike using
% Monte Carlo simulations. Returns a vector consisting of the price as its
% first element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_LC_fixed_crude(S0,sig,r,K,T,n,N)
    stockPath = StockPath(S0,sig,r,T,n,N);
    payOff = max(0,max(stockPath)-K);
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

**Lookback-köpoption - Fast lösenpris; Med kontrollvariat**

```
% MonteCarlo_LC_fixed.m
%
% Computes the price of an Lookback call option with fixed strike using
% Monte Carlo simulations. Here the price of the European vanilla call is
% used as a control variate. Returns a vector consisting of the price as
% its first element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_LC_fixed(S0,sig,r,K,T,n,N)
    % Deterministic value for control variate
    detValueCV = blsprice(S0,K,r,T,sig);
```

```
    stockPath = StockPath(S0,sig,r,T,n,N);
    payOffCV = max(stockPath(end,:)-K,0);     % Pay-off control variate
    payOff = max(0,max(stockPath)-K);

    coVar = cov(payOffCV,payOff);
    payOff = payOff-(coVar(1,2)/var(payOffCV))*(payOffCV-detValueCV);
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

**Lookback-säljoption - Fast lösenpris; Utan kontrollvariat**

```
% MonteCarlo_LP_fixed_crude.m
%
% Computes the price of an Lookback put option with fixed strike using
% Monte Carlo simulations. Returns a vector consisting of the price as its
% first element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_LP_fixed_crude(S0,sig,r,K,T,n,N)
    stockPath = StockPath(S0,sig,r,T,n,N);
    payOff = max(0,K-min(stockPath));
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

**Lookback-säljoption - Fast lösenpris; Med kontrollvariat**

```
% MonteCarlo_LP_fixed.m
%
% Computes the price of an Lookback put option with fixed strike using
% Monte Carlo simulations. Here the price of the European vanilla put is
% used as a control variate. Returns a vector consisting of the price as
% its first element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_LP_fixed(S0,sig,r,K,T,n,N)
    % Deterministic value for control variate
    [~, detValueCV] = blsprice(S0,K,r,T,sig);

    stockPath = StockPath(S0,sig,r,T,n,N);
    payOffCV = max(K-stockPath(end,:),0);     % Pay-off control variate
```

```
    payOff = max(0,K-min(stockPath));

    coVar = cov(payOffCV,payOff);
    payOff = payOff-(coVar(1,2)/var(payOffCV))*(payOffCV-detValueCV);
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

**Lookback-köpoption - Rörligt lösenpris; Utan kontrollvariat**

```
% MonteCarlo_LC_floating_crude.m
%
% Computes the price of an Lookback call option with floating strike using
% Monte Carlo simulations. Returns a vector consisting of the price as its
% first element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_LC_floating_crude(S0,sig,r,T,n,N)
    stockPath = StockPath(S0,sig,r,T,n,N);
    payOff = max(0,stockPath(n+1,:)-min(stockPath));
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

**Lookback-köpoption - Rörligt lösenpris; Med kontrollvariat**

```
% MonteCarlo_LC_floating.m
%
% Computes the price of an Lookback call option with floating strike using
% Monte Carlo simulations. Here the price of the European vanilla call,
% with K=S0, is used as a control variate. Returns a vector consisting of
% the price as its first element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_LC_floating(S0,sig,r,T,n,N)
    % Deterministic value for control variate
    detValueCV = blsprice(S0,S0,r,T,sig);

    stockPath = StockPath(S0,sig,r,T,n,N);
    payOffCV = max(stockPath(end,:)-S0,0);      % Pay-off control variate
    payOff = max(0,stockPath(end,:)-min(stockPath));
```

```
    coVar = cov(payOffCV,payOff);
    payOff = payOff-(coVar(1,2)/var(payOffCV))*(payOffCV-detValueCV);
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

**Lookback-säljoption - Rörligt lösenpris; Utan kontrollvariat**

```
% MonteCarlo_LP_floating_crude.m
%
% Computes the price of an Lookback put option with floating strike using
% Monte Carlo simulations. Returns a vector consisting of the price as its
% first element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_LP_floating_crude(S0,sig,r,T,n,N)
    stockPath = StockPath(S0,sig,r,T,n,N);
    payOff = max(0,max(stockPath)-stockPath(end,:));
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

**Lookback-säljoption - Rörligt lösenpris; Med kontrollvariat**

```
% MonteCarlo_LP_floating.m
%
% Computes the price of an Lookback put option with floating strike using
% Monte Carlo simulations. Here the price of the European vanilla put, with
% K=S0, is used as a control variate. Returns a vector consisting of the
% price as its first element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.

function [price, deviation] = MonteCarlo_LP_floating(S0,sig,r,T,n,N)
    % Deterministic value for control variate
    [~, detValueCV] = blsprice(S0,S0,r,T,sig);
    stockPath = StockPath(S0,sig,r,T,n,N);
    payOffCV = max(S0-stockPath(end,:),0);     % Pay-off control variate
    payOff = max(0,max(stockPath)-stockPath(end,:));

    coVar = cov(payOffCV,payOff);
    payOff = payOff-(coVar(1,2)/var(payOffCV))*(payOffCV-detValueCV);
    price = exp(-r*T)*mean(payOff);
```

```
    deviation = std(payOff)/sqrt(2*N);
end
```

**Barriär Upp-Ut-köpoption**

```
% MonteCarlo_BUOC.m
%
% Computes the price of an Barrier Up-and-Out call option using Monte Carlo
% simulations. Returns a vector consisting of the price as its first
% element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.
% - B > S0. Barrier level.


function [price, deviation] = MonteCarlo_BUOC(S0,sig,r,K,T,n,N,B)
    stockPath=StockPath(S0,sig,r,T,n,N);
    for i=1:2*N
        if any(stockPath(:,i) >= B)
            stockPath(n+1,i)=0;
        end
    end
    payOff=max(0,stockPath(n+1,:)-K);
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

**Barriär Upp-In-köpoption**

```
% MonteCarlo_BUIC.m
%
% Computes the price of an Barrier Up-and-In call option using Monte Carlo
% simulations. Returns a vector consisting of the price as its first
% element, and the standard deviation as its second.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - N > 0, Integer. Number of simulated paths.
% - B > S0. Barrier level.

function [price, deviation] = MonteCarlo_BUIC(S0,sig,r,K,T,n,N,B)
    stockPath = StockPath(S0,sig,r,T,n,N);
    for i=1:2*N
        if all(stockPath(:,i) < B)
            stockPath(n+1,i)=0;
        end
```

```
    end
    payOff = max(0,stockPath(n+1,:)-K);
    price = exp(-r*T)*mean(payOff);
    deviation = std(payOff)/sqrt(2*N);
end
```

## C.2 Crank-Nicolson-metoden

**Europeisk standard köpoption**

```
% CrankNicolson_EC.m
%
% Computes the price of an European call option using the finite difference
% method: Crank-Nicolson. Returns a vector consisting of four elements.
% The first one is the requested price. The second one is the solution
% matrix. Each row represent a time step, where t=T is the first row and
% t=0 the last. The third and fourth elements are the partitions of the
% space and time respectively.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - m > 0, Integer. Number of partitions of the stock price.
% - X >> S0. Quantity sufficiently large to represent S='inf'.
%
% For price of exact input S0: X mod m = 0.

function [price,sol,space,time] = CrankNicolson_EC(S0,sig,r,K,T,n,m,X)
    % Define deltas (steps) and initiate solution matrix.
    priceIndex = round((m/X)*S0+1);
    dt=T/n;
    dx=X/m;
    d=dt/dx^2;
    sol=zeros(n+1,m+1);
    time = 0:dt:T;
    space = 0:dx:X;

    A = zeros(m+1,m+1);
    B = zeros(m+1,m+1);

    % To satisfy u_(i+1,1)=u_(i,1),
    % u_(i+1,end) ~ u(i, end) (updated in solution-loop exact)
    A(1,1) = 1;
    A(m+1,m+1) = 1;
    B(1,1) = 1;
    B(m+1,m+1) = 1;

    % Boundary Conditions, solution
    sol(1,:) = max(space-K,0);  % Pay-off
    sol(:,1)=0; % C(t,0) = 0, S(t)=0
    sol(:,m+1)=X-K*exp(-r*time);     % C(t,'inf'), S(t) ~ 'inf'

    % Standard elements in matrices, row (2) to (end-1)
    alpha = @(x) (1/4)*d*sig^2*space(x)^2;
```

```
    beta = @(x) (1/4)*d*r*space(x)*dx;

    for k=2:m
        A(k,k-1)= beta(k)-alpha(k);
        A(k,k)= 1+r*dt/2+2*alpha(k);
        A(k,k+1)= -beta(k)-alpha(k);

        B(k,k-1)= -beta(k)+alpha(k);
        B(k,k)= 1-r*dt/2-2*alpha(k);
        B(k,k+1)= beta(k)+alpha(k);
    end
    invA = inv(A);
    transB = transpose(B);
    transinvA = transpose(invA);
    for i=2:n+1
        sol(i,:) = sol(i-1,:)*transB*transinvA;
        sol(i,m+1)=X-K*exp(-r*time(i)); % Updating, exact B.C.
    end
    price = sol(end,priceIndex);
end
```

**Europeisk standard säljoption**

```
% CrankNicolson_EP.m
%
% Computes the price of an European put option using the finite difference
% method: Crank-Nicolson. Returns a vector consisting of four elements.
% The first one is the requested price. The second one is the solution
% matrix. Each row represent a time step, where t=T is the first row and
% t=0 the last. The third and fourth elements are the partitions of the
% space and time respectively.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - m > 0, Integer. Number of partitions of the stock price.
% - X >> S0. Quantity sufficiently large to represent S='inf'.
%
% For price of exact input S0: X mod m = 0.

function [price,sol,space,time] = CrankNicolson_EP(S0,sig,r,K,T,n,m,X)
    % Define deltas (steps) and initiate solution matrix.
    priceIndex = round((m/X)*S0+1);
    dt=T/n;
    dx=X/m;
    d=dt/dx^2;
    sol=zeros(n+1,m+1);
    time = 0:dt:T;
    space = 0:dx:X;

    A = zeros(m+1,m+1);
    B = zeros(m+1,m+1);

    % To satisfy u_(i+1,1) ~ u_(i,1) (updated in solution-loop exact),
```

```
    % u_(i+1,end)=u(i, end)
    A(1,1) = 1;
    A(m+1,m+1) = 1;
    B(1,1) = 1;
    B(m+1,m+1) = 1;

    % Boundary Conditions, solution
    sol(1,:) = max(K-space,0);  % Pay-off
    sol(:,1)=K*exp(-r*time); % P(t,0), S(t) = 0
    sol(:,m+1)=0;    %P(t,'inf') = 0, S(t) ~ 'inf'

    % Standard elements in matrices, row (2) to (end-1)
    alpha = @(x) (1/4)*d*sig^2*space(x)^2;
    beta = @(x) (1/4)*d*r*space(x)*dx;

    for k=2:m
        A(k,k-1)= beta(k)-alpha(k);
        A(k,k)= 1+r*dt/2+2*alpha(k);
        A(k,k+1)= -beta(k)-alpha(k);

        B(k,k-1)= -beta(k)+alpha(k);
        B(k,k)= 1-r*dt/2-2*alpha(k);
        B(k,k+1)= beta(k)+alpha(k);
    end
    invA = inv(A);
    transB = transpose(B);
    transinvA = transpose(invA);
    for i=2:n+1
        sol(i,:) = sol(i-1,:)*transB*transinvA;
        sol(i,1)=K*exp(-r*time(i)); % Updating, exact B.C.
    end
    price = sol(end, priceIndex);
end
```

**Asiatisk köpoption - Aritmetiskt medelvärde**

```
% CrankNicolson_AC.m
%
% Computes the price of an Asian call option using the finite difference
% method: Crank-Nicolson. Returns a vector consisting of four elements.
% The first one is the requested price. The second one is the solution
% matrix. Each row represent a time step, where t=T is the first row and
% t=0 the last. The third and fourth elements are the partitions of the
% space and time respectively.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - m > 0, Integer. Number of partitions of the stock price.

function [price,sol,space,time] = CrankNicolson_AC(S0,sig,r,K,T,n,m)
    % Define deltas (steps) and initiate solution matrix.
    %Zmax=1;
    Z0 = (1-exp(-r*T))/(r*T) - K*exp(-r*T)/S0;
```

```
    dt=T/n;
    dz=2*(abs(Z0)+1)/m;
    %dz=2*Zmax/m;
    d = dt/dz^2;
    %Z0 = (1-exp(-r*T))/(r*T) - K*exp(-r*T)/S0;
    sol=zeros(n+1,m+1);
    time = 0:dt:T;
    space = -(abs(Z0)+1):dz:(abs(Z0)+1);
    [~,Z0index] = min(abs(space-Z0));
    space = space-space(Z0index)+Z0; %Adjust vector so space(Z0index) = Z0
    A = zeros(m+1,m+1);
    B = zeros(m+1,m+1);

    % To satisfy u_(i+1,1)=u_(i,1),
    % u_(i+1,end) ~ u(i, end) (updated in solution-loop exact)
    A(1,1) = 1;
    A(m+1,m+1) = 1;
    B(1,1) = 1;
    B(m+1,m+1) = 1;

    % Boundary Conditions, solution
    sol(1,:) = max(space,0);   % Pay-off
    sol(:,1)=0; % C(t,0) = 0, S(t)=0
    sol(:,m+1)=space(end);     % C(t,'inf'), S(t) ~ 'inf'

    % Standard elements in matrices, row (2) to (end-1)
    Q = (1-exp(-r*time))/(r*T);
    alpha = @(h,j) d*sig^2*(Q(h)-space(j))^2/4;

    for i=2:n+1
        for k=2:m
            A(k,k-1)= -alpha(i,k);
            A(k,k)= 1+2*alpha(i,k);
            A(k,k+1)=-alpha(i,k);

            B(k,k-1)= alpha(i-1,k);
            B(k,k)=1-2*alpha(i-1,k);
            B(k,k+1)= alpha(i-1,k);
        end
        sol(i,:)=A\(B*sol(i-1,:)');
    end
    price = S0*sol(end,Z0index);
end
```

**Asiatisk säljoption - Aritmetiskt medelvärde**

```
% CrankNicolson_AP.m
%
% Computes the price of an Asian put option using the finite difference
% method: Crank-Nicolson. Returns a vector consisting of four elements.
% The first one is the requested price. The second one is the solution
% matrix. Each row represent a time step, where t=T is the first row and
% t=0 the last. The third and fourth elements are the partitions of the
% space and time respectively.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
```

```
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - m > 0, Integer. Number of partitions of the stock price.

function [price,sol,space,time] = CrankNicolson_AP(S0,sig,r,K,T,n,m)
    % Define deltas (steps) and initiate solution matrix.
    %Zmax=1;
    Z0 = -(1-exp(-r*T))/(r*T) + K*exp(-r*T)/S0;
    dt=T/n;
    dz=2*(abs(Z0)+1)/m;
    %dz=2*Zmax/m;
    d = dt/dz^2;
    %Z0 = (1-exp(-r*T))/(r*T) - K*exp(-r*T)/S0;
    sol=zeros(n+1,m+1);
    time = 0:dt:T;
    space = -(abs(Z0)+1):dz:(abs(Z0)+1);
    [~,Z0index] = min(abs(space-Z0));
    space = space-space(Z0index)+Z0; %Adjust vector so space(Z0index) = Z0
    A = zeros(m+1,m+1);
    B = zeros(m+1,m+1);

    % To satisfy u_(i+1,1)=u_(i,1),
    % u_(i+1,end) ~ u(i, end) (updated in solution-loop exact)
    A(1,1) = 1;
    A(m+1,m+1) = 1;
    B(1,1) = 1;
    B(m+1,m+1) = 1;

    % Boundary Conditions, solution
    sol(1,:) = max(space,0);   % Pay-off
    sol(:,1)=0; % C(t,0) = 0, S(t)=0
    sol(:,m+1)=space(end);     % C(t,'inf'), S(t) ~ 'inf'

    % Standard elements in matrices, row (2) to (end-1)
    Q = (1-exp(-r*time))/(r*T);
    alpha = @(h,j) d*sig^2*(Q(h)-space(j))^2/4;

    for i=2:n+1
        for k=2:m
            A(k,k-1)= -alpha(i,k);
            A(k,k)= 1+2*alpha(i,k);
            A(k,k+1)=-alpha(i,k);

            B(k,k-1)= alpha(i-1,k);
            B(k,k)=1-2*alpha(i-1,k);
            B(k,k+1)= alpha(i-1,k);
        end
        sol(i,:)=A\(B*sol(i-1,:)');
    end
    price = S0*sol(end,Z0index);
end
```

**Lookback-säljoption - Rörligt lösenpris**

```
% CrankNicolson_LP_floating.m
%
```

```
% Computes the price of an Lookback put option with floating strike using
% the finite difference method: Crank-Nicolson. Returns a vector consisting
% of four elements. The first one is the requested price. The second one is
% the solution matrix. Each row represent a time step, where t=T is the
% first row and t=0 the last. The third and fourth elements are the
% partitions of the space and time respectively.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - m > 0, Integer. Number of partitions of the stock price.

function [price,sol,space,time] = CrankNicolson_LP_floating(S0,sig,r,T,n,m)
    % Define deltas (steps) and initiate solution matrix.
    dt=T/n;
    dz=1/m;
    d=dt/dz^2;
    sol=zeros(n+1,m+1);
    time = 0:dt:T;
    space = 0:dz:1;

    A = zeros(m+1,m+1);
    B = zeros(m+1,m+1);

    % To satisfy u_(i+1,1)=exp(-r*dt)*u_(i,1),
    % u_(i+1,end)=u(i+1, end)_dz
    A(1,1) = exp(-r*dt);
    A(m+1,m+1) = 1;
    B(1,1)= 1;
    B(m+1,m+1) = 0;
    A(m+1,m) = -1/(1-dz);

    %BC
    sol(1,:) = 1-space; % u(0,z)
    sol(:,1)=exp(-r*time); % u(t,0)

    % Standard elements in matrices, row (2) to (end-1)
    alpha = @(x) (1/4)*d*sig^2*space(x)^2;
    beta = @(x) (1/4)*d*r*space(x)*dz;

    for k=2:m
        A(k,k-1)= beta(k)-alpha(k);
        A(k,k)= 1+r*dt/2+2*alpha(k);
        A(k,k+1)= -beta(k)-alpha(k);

        B(k,k-1)= -beta(k)+alpha(k);
        B(k,k)= 1-r*dt/2-2*alpha(k);
        B(k,k+1)= beta(k)+alpha(k);
    end
    invA = inv(A);
    transB = transpose(B);
    transinvA = transpose(invA);
    for i=2:n+1
        sol(i,:) = sol(i-1,:)*transB*transinvA;
```

```
    end
    price = S0*sol(end,end);
end
```

**Barriär Upp-Ut-köpoption**

```
% CrankNicolson_BUOC.m
%
% Computes the price of an Barrier Up-and-Out call option using the finite
% difference method: Crank-Nicolson. Returns a vector consisting of four
% elements. The first one is the requested price. The second one is the
% solution matrix. Each row represent a time step, where t=T is the first
% row and t=0 the last. The third and fourth elements are the partitions of
% the space and time respectively.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - m > 0, Integer. Number of partitions of the stock price.
% - B > S0. Barrier level.
%
% For price of exact input S0: B mod m = 0.

function [price,sol,space,time] = CrankNicolson_BUOC(S0,sig,r,K,T,n,m,B)
    % Define deltas (steps) and initiate solution matrix.
    priceIndex = round((m/B)*S0+1)
    dt=T/n;
    dx=B/m;
    d=dt/dx^2;
    sol=zeros(n+1,m+1);
    time = 0:dt:T;
    space = 0:dx:B;

    A = zeros(m+1,m+1);
    B = zeros(m+1,m+1);

    % To satisfy u_(i+1,1)=u_(i,1),
    % u_(i+1,end) = u(i, end)
    A(1,1) = 1;
    A(m+1,m+1) = 1;
    B(1,1) = 1;
    B(m+1,m+1) = 1;

    % Boundary Conditions, solution
    sol(1,:) = max(space-K,0);  % Pay-off
    sol(:,1)=0; % C(t,0) = 0, S(t)=0
    sol(:,m+1)=0;     % C(t,Barrier), S(t) = Barrier

    % Standard elements in matrices, row (2) to (end-1)
    alpha = @(x) (1/4)*d*sig^2*space(x)^2;
    beta = @(x) (1/4)*d*r*space(x)*dx;

    for k=2:m
        A(k,k-1)= beta(k)-alpha(k);
```

```
        A(k,k)= 1+r*dt/2+2*alpha(k);
        A(k,k+1)= -beta(k)-alpha(k);

        B(k,k-1)= -beta(k)+alpha(k);
        B(k,k)= 1-r*dt/2-2*alpha(k);
        B(k,k+1)= beta(k)+alpha(k);
    end
    invA = inv(A);
    transB = transpose(B);
    transinvA = transpose(invA);
    for i=2:n+1
        sol(i,:) = sol(i-1,:)*transB*transinvA;
    end
    price = sol(end,priceIndex);
end
```

**Barriär Upp-In-köpoption**

```
% CrankNicolson_BUIC.m
%
% Computes the price of an Barrier Up-and-Out call option using the finite
% difference method: Crank-Nicolson. Returns a vector consisting of four
% elements. The first one is the requested price. The second one is the
% solution matrix. Each row represent a time step, where t=T is the first
% row and t=0 the last. The third and fourth elements are the partitions of
% the space and time respectively.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - n > 0, Integer. Number of partitions of the time interval [0 T].
% - m > 0, Integer. Number of partitions of the stock price.
% - B > S0. Barrier level.
%
% For price of exact input S0: B mod m = 0.

function [price,sol,space,time] = CrankNicolson_BUIC(S0,sig,r,K,T,n,m,B)
    dx=B/m;
    priceIndex = round((m/B)*S0+1);
    m_hat = floor(Bar/dx);
    [~, ~, tmpsol] = BSCrankUpOutCall(T,X,n,m_hat,r,sig,K,Bar);
    [time, space, sol] = BSCrankEUCall(T,X,n,m,r,sig,K);
    sol(:,1:m_hat+1) = sol(:,1:m_hat+1)-tmpsol;
    price = sol(end, priceIndex);
end
```

## C.3 Exakta formler

**Asiatisk köpoption - Geometriskt medelvärde**

```
% ClosedFormula_AC_geo.m
%
% Function that returns the exact price of an geometric Asian call option.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.


function price = ClosedFormula_AC_geo(S0,sig,r,K,T)
    dStar = (T/2)*(r-sig.^2/6);
    d = (log(S0/K) + (T/2)*(r+sig.^2/6))/(sig*sqrt(T/3));
    price = exp(dStar)*S0*normcdf(d)-K*normcdf(d-sig*sqrt(T/3));
end
```

**Asiatisk säljoption - Geometriskt medelvärde**

```
% ClosedFormula_AC_geo.m
%
% Function that returns the exact price of an geometric Asian call option.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.


function price = ClosedFormula_AP_geo(S0,sig,r,K,T)
    dStar = (T/2)*(r-sig.^2/6);
    d = (log(S0/K) + (T/2)*(r+sig.^2/6))/(sig*sqrt(T/3));
    price = -exp(dStar)*S0*normcdf(-d)+K*normcdf(-d+sig*sqrt(T/3));
end
```

**Lookback-säljoption - Rörligt lösenpris**

```
% ClosedFormula_LP_gloating.m
%
% Function that returns the exact price of an Lookback put option with
% floating strike.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - T >= 0. Time to maturity.


function price = ClosedFormula_LP_floating(S0,sig,r,T)
    dP = @(tau,s) (log(s)+(r+sig^2/2)*tau)/(sig*sqrt(tau)); %deltaPlus
```

```
    dM = @(tau,s) (log(s)+(r-sig^2/2)*tau)/(sig*sqrt(tau)); %deltaMinus

    price = S0*((1+sig^2/(2*r))*normcdf(dP(T,1))...
            + exp(-r*T)*normcdf(-dM(T,1))...
            - (sig^2/(2*r))*exp(-r*T)*1^(1-2*r/sig^2)*normcdf(-dM(T,1))-1);
end
```

**Barriär Upp-Ut-köpoption**

```
% ClosedFormula_BUOC.m
%
% Function that returns the exact price of an Barrier Up-and-Out call
% option.
%
% Variables:
% - S0 >= 0. Initial stock price.
% - sig >= 0. Volatility.
% - r >= 0. Risk-free interest rate.
% - K >= 0. Strike price.
% - T >= 0. Time to maturity.
% - B > S0. Barrier level.


function price = ClosedFormula_BUOC(S0,sig,r,K,T,B)
    dP = @(tau,s) (log(s)+(r+sig^2/2)*tau)/(sig*sqrt(tau)); %deltaPlus
    dM = @(tau,s) (log(s)+(r-sig^2/2)*tau)/(sig*sqrt(tau)); %deltaMinus

    price = S0*(normcdf(dP(T,S0/K))-normcdf(dP(T,S0/B)))...
    - exp(-r*T)*K*(normcdf(dM(T,S0/K))-normcdf(dM(T,S0/B)))...
    - B*(S0/B)^(-2*r/sig^2)...
    * (normcdf(dP(T,B^2/(K*S0)))-normcdf(dP(T,B/S0)))...
    + exp(-r*T)*K*(S0/B)^(-2*r/sig^2+1)...
    * (normcdf(dM(T,B^2/(K*S0)))-normcdf(dM(T,B/S0)));
end
```