Enhancement of the Bessel computation in the COS method.
(Nederlandse titel: Verbetering van de Bessel berekening in de COS methode.)

Verslag ten behoeve van het
Delft Institute for Applied Mathematics
als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE

doort

Toni Budimir

Delft, Nederland
June 2011

Copyright © 2011 door Toni Budimir. Alle rechten voorbehouden.
“Enhancement of the Bessel computation in the COS method.”
(Nederlandse titel: “Verbetering van de Bessel berekening in de COS methode.”)

Toni Budimir

Technische Universiteit Delft

Begeleider
Prof.dr.ir. C.W. Oosterlee

Overige commissieleden
Prof.dr.ir. C. Vuik
Dr. H.M. Schuttelaars
Dr. G.F. Ridderbos

June, 2011
Delft
1 Introduction

In the financial world it is important to have fast numerical methods to determine the prices of options. These methods need to have a low computational time but also have to be accurate. Methods that are slow are not helpful in the financial world since the parameters of financial products are constantly changing over time.

The research is based on previous findings from literature mostly written by F.Fang and C.W.Oosterlee. They developed an algorithm to determine the prices of options using the COS method. Their algorithm for determining option values was already a lot faster than other methods. However, there was still a bottleneck in the computation which was the Bessel function. In my research the focus will be on replacing this function by a proper approximation.

Much of the information provided by F.Fang has been included in the appendix since this information is vital for the understanding of the algorithm for pricing Bermudan options under the Heston model. We will not further discuss this information since it is too complex to discuss in the main article. In the normal sections we will discuss the parts of the algorithm that are key to our problem.

In section 2 we will discuss the basic concept of options and how these financial products are priced. There are different kinds of options that will be discussed but in the rest of the research we will focus on the Bermudan options. The techniques used can also be transformed for pricing European and American options but this is not further investigated.

In section 3 we will look at the COS-method which is a Fourier cosine series expansion method used for determining the option value. Usually we do not know the behaviour of the underlying asset of the option but we do know the characteristic function of this behaviour from which we can reconstruct the probability density function. We will then discuss the use of this COS method on a more accurate model for option pricing. This model is called the Heston model and also takes into account the volatility which is the variance of the underlying asset.

In section 4 we will again look at the COS method used on the Heston model. However, here we look at a special form of options, called the Bermudan options. These models have a special feature that they can be exercised at different times during the lifespan of the option.

In section 5 we discuss the Bessel function and the approximations that are used to replace this function. We will show two different approximations that can be used in different regions of the complex plane. The regions and accuracy will be shown in different tables.

In section 6 we will discuss the numerical results that are found by using the original code provided by F.Fang and the new code which uses the Bessel approximation. We will see that there is indeed a significant reduction in computational time without any loss in accuracy.

In the last section we will discuss the conclusions that can be made by looking at the data found by replacing the Bessel function. We will also make some recommendations for further research.

The main goal of this research is speeding up the algorithm for computation of Bermudan option prices through replacing the Bessel function by an approximation.
1.1 Commonly used symbols

In this paper we will use a lot of symbols so that it can be hard to keep track of all the meanings of these symbols. Therefore, we provide a list of all these symbols and their meaning.

- $S(t)$: asset price at time $t$.
- $K$: exercise price of an option.
- $T$: expiry date.
- $t_m$: exercise point for an Bermudan option.
- $V(S,T), v(x,t_0)$: value of an option.
- $C(S,T)$: value of a call option.
- $P(S,T)$: value of a put option.
- $\Delta t$: time step, $\Delta t = t_m - t_{m-1}$.
- $r$: interest rate.
- $\mu$: average rate of growth of an asset.
- $\sigma$: volatility of the asset.
- $f(y|x)$: probability density function of an asset.
- $\phi(\omega)$: characteristic function.
- $A_k$: Fourier cosine coefficients.
- $F_k, V_k$: approximation of the Fourier cosine coefficients.
- $[a,b]$: truncation region.
- $\sum'$: summation sign that indicates that the first term is multiplied by one-half.
- $q$: dividend rate.
- $N$: number of COS terms.
- $\sigma_t$: variance of the asset price process.
- $x_t$: log-asset price variable.
- $\lambda$: speed of mean reversion.
- $\bar{v}$: the mean level of variance.
- $\eta$: volatility of the volatility.
- $W_t$: Brownian motion.
- $\rho$: correlation factor.
- $q$: variable that indicates how close we are to meeting the Feller condition.
- $\zeta$: variable for the non-central chi-square distribution.
- $p_v(v_t|v_s)$: probability density function of the variance of the asset price given a previous variance.
- $g(x_m,t_m)$: payoff function for a Bermudan option.
- $c(x_m,\sigma_m,t_m)$: continuation function for a Bermudan option.
- $I_q(z)$: modified Bessel function of the first kind with order $q$.
- $\varsigma_j$: quadrature grid point.
- $J$: number of quadrature points in the variance dimension.
• **TOL**: tolerance level for the error.

• **M**: number of exercise points for a Bermudan option.

• **x_m**: early-exercise point.

• **M_{k,j}**: matrix used for calculating Fourier cosine coefficients.

• **M_c**: Hankel matrix.

• **M_s**: Toeplitz matrix.

• **[a_v, b_v]**: truncation range for the log-variance density domain.

• **p_{ln(v)}(\sigma_t|\sigma_s)**: probability density of the log-variance of the asset price given a previous log-variance.

• **p_{x,ln(v)}(x_t, \sigma_t|x_s, \sigma_s)**: probability density function of the joint distribution of the log-stock price and the log-variance.

• **\phi(\omega; x_s, \sigma_t, \sigma_t)**: conditional characteristic function for the Heston model.

• **\Phi(v; v_t, v_s)**: characteristic function for the Heston model.

• **w_j**: weight of the quadrature points.

• **\beta_n**: vector used in the calculation of the Bermudan option prices.
2 Introduction to options and pricing

An option is a contract that gives the owner of the option the opportunity to buy or sell an asset at a prescribed price at a certain time in the future. The owner of the option may buy or sell an asset but is not obligated to do so. Therefore, an option has a value because either the owner exercises his right and will make a profit or he does not and will lose nothing. The profit that the owner makes is the difference between the asset’s value and the exercise price. One might imagine an owner of the option buying an option at a price that is below the asset’s price and immediately selling it to make an instant profit.

There are two kinds of options that will be discussed in this paper. First of all, we have a call option that gives the owner of the option the right to buy an asset for a prescribed amount called the exercise price at a prescribed time in the future known as the expiry date. It is clear that an owner of a call option will want the asset’s price to rise rapidly so he can buy the asset for a lower price and make a profit that is equal to the difference between the asset’s price and the exercise price.

Another option that will be discussed is the so called put option. A put option gives the owner of the option the opportunity to sell an asset for a price that is agreed upon by both the owner and writer of the option. Again we have that this prescribed amount is called the exercise price and that the time at which this option has to be exercised is called the expiry date of the option. In this case we have that the owner will make the most profit when the value of the underlying asset drops because then the owner can sell the option for a price that is higher than the actual price of the underlying asset. The formula for the profit of a put option is equal to the formula of the call option, namely the difference between the exercise price and the underlying asset’s price.

2.1 European, American and Bermudan options

An European option is the simplest option in terms of pricing. The owner of this kind of option has the right to exercise the option at the time of maturity. This time of maturity is the only moment where the owner of the option can make a decision whether to exercise the option or not. The value of the option is equal to the difference between the exercise price that is included in the contract and the asset price at the time of maturity. We call the exercise price \( K \) and the asset price at expiry \( S(T) \). Since the owner of the option is not obliged to exercise the option we have that the value of the call option at time of expiry is given by this formula:

\[
V(S, T) = \max(S(T) - K, 0)
\]  

The value of a put option is given by:

\[
V(S, T) = \max(K - S(T), 0)
\]

An American option is a more complex financial product since it gives the owner the option to exercise the option at any time till the time of maturity. Since the owner of the option has more choices this financial product will usually be more expensive than the European option. The exercise price is still given by \( K \) and we now have the value of the asset at time \( t \) given by \( S(t) \). The value of the option is given by this formula for a call option:

\[
V(S, t) = \max(S(t) - K, 0)
\]

For a put option we get a similar equation that gives us the following value:

\[
V(S, t) = \max(K - S(t), 0)
\]

Bermudan options are special options that can be exercised at multiple times during the duration of the option. The times at which the option can be exercised are pre-specified. Between these dates the option
is priced like a normal European option. We start with time $t_0$ and then have times: $t_1, t_2, ..., t_M$ with $\Delta t = t_m - t_{m-1}, t_0 < t_1 < ... t_M = T$. The pricing formula is then the following:

$$
\begin{align*}
\begin{cases}
  c(x, t_{m-1}) &= e^{-r\Delta t} \int_{\mathbb{R}} v(y, t_m) f(y|x) \, dy, \\
  v(x, t_{m-1}) &= \max(g(x, t_{m-1}), c(x, t_{m-1})),
\end{cases}
\end{align*}
$$

followed by

$$
v(x, t_0) = e^{-r\Delta t} \int_{\mathbb{R}} v(y, t_1) f(y|x) \, dy. \tag{5}
$$

We define $x$ and $y$ as the logarithm over the asset price and the strike price with: $x := \ln(S(t_{m-1}/K))$ and $y := \ln(S(t_m)/K)$. The option value is here stated by $v(x, t)$, the continuation value is called $c(x, t)$ and the payoff at time $t$ is $g(x, t)$.

### 2.2 Basic model for asset prices

We will start with our first model that looks at the behaviour of asset prices. We have to make two basic assumptions:

- Markets respond immediately to any new information about an asset’s price.
- The past history of the asset’s behaviour is fully shown in the present price which does not hold any other information.

We can now conclude that the model is only affected by new information that is added to the model. So we have to deal with a Markov process. A Markov process means that the future price only depends on the current price and not on past prices. We will look at the relative change of the asset price which is denoted by $\frac{dS}{S}$. We will look at this change at a time called $t$ when the asset price is $S$. We now take a small time-step called $dt$ and assume that the asset price changes to $S + dS$. The change in $\frac{dS}{S}$ can be divided into two parts.

The first part is concerned with the predictable return that would be earned if the money would be put in a bank account with zero risk. This part gives the contribution:

$$
\mu dt \tag{6}
$$

In this part of the change of $\frac{dS}{S}$ we have that $\mu$ is the average rate of growth of an asset price which is also called the drift. Here we will assume that $\mu$ is a constant.

The second part of the change $\frac{dS}{S}$ is the random change in the asset price by external effects. This part is denoted as following:

$$
\sigma dX \tag{7}
$$

In this contribution we have that $\sigma$ is the volatility of the asset. The volatility is the standard deviation of the returns. The part $dX$ is a sample from a normal distribution.

If we put these two parts together we get the following formula for the change in asset price:

$$
\frac{dS}{S} = \mu dt + \sigma dX \tag{8}
$$

We call a formula like this a stochastic differential equation. The term $dX$ is called a Wiener process that describes the randomness that an asset’s price holds. The Wiener process has the following properties:

- The variable $dX$ is random and drawn from the normal distribution.
- The mean of $dX$ is zero.
- The variance of $dX$ is $dt$.  

9
We can write $dX$ as $\sqrt{\phi} dt$. We take that $\phi$ is the standardised normal distribution which has mean zero and unit variance. We take $\sqrt{\phi}$ because any other choice of scaling would lead to a trivial or meaningless problem.

### 2.3 Model for option prices

We will now look at the price modeling for options. We first have to define a couple of parameters:

- The value of an option is denoted by $V(S, t)$. The value is a function of the current value of the underlying asset, $S$ and the time $t$.
- The volatility of the underlying asset is $\sigma$.
- The exercise price is $K$.
- The expiry is $T$.
- The interest rate $r$.

We can look at the value of the option at the time of expiry $t = T$. We can say that if $S > K$ it makes sense to exercise a call option. If $S < K$ it is clear that the owner of the option will not exercise the option so the option’s value is 0. We now call the value of a call option $C(S, t)$ which is a function of the underlying asset price, $S$ and the time $t$. We now have our first boundary condition, namely:

$$C(S, T) = \max(S - K, 0) \quad (9)$$

On the other hand if we look at a put option we will have a different boundary condition. If $S > K$, the owner of the option will not exercise the option because this would mean that he sell the underlying asset for a price that is lower than the asset’s price. If we look at the case that $S < K$, it makes sense to exercise the option. We call the value of a put option $P(S, t)$ which is a function of the underlying asset price, $S$ and the time $t$. This leaves us with the following boundary condition:

$$P(S, T) = \max(K - S, 0) \quad (10)$$

We will now list some assumptions that are necessary for the analysis of the value of an option:

- The asset’s price follows the log normal random walk, $\frac{dS}{S} = \mu dt + \sigma dX$
- The interest rate $r$ and the volatility $\sigma$ are functions of the time which are known over the entire lifespan of the option.
- There are no transaction costs associated with hedging a portfolio. Hedging means that we buy a set of call and put options that minimise our loss if there is a change in the underlying asset price.
- The underlying asset pays no dividends during the life of the option. Dividend is a portion of the profit that the corporation makes that is payed to the shareholders.
- All risk-free portfolios must earn the same return.
- We can trade continuously in the underlying asset which in real life is not the case because of trading hours of stock exchanges.
- We can sell or buy any number of the underlying asset so we do not have to take into account that the amount of an option bought or sold should be an integer.

We can use Ito’s lemma on the asset’s price to come to a value for an option. Ito’s lemma relates a small change in the function of a random variable to a small change of the random variable. We saw in the previous section that $dX = \sqrt{\phi} dt$ so we can now state the following:

$$dX^2 \rightarrow dt \text{ as } dt \rightarrow 0 \quad (11)$$
We take the function $V(S,t)$ that is a smooth function of $S$ and $t$ and we forget for the moment that $S$ is stochastic. We can write the Taylor expansion for $V(S,t)$ which is:

$$dV = \frac{\partial V}{\partial S} dS + \frac{\partial V}{\partial t} dt + \frac{1}{2} \frac{\partial^2 V}{\partial S^2} dS^2 + \ldots$$

(12)

We look at the term $dS^2$ which is:

$$dS^2 = (\sigma S dX + \mu S dt)^2 = \sigma^2 S^2 dX^2 + 2\sigma S dX dt + \mu^2 S^2 dt^2$$

(13)

Since we have that $dX^2 \to dt$ as $dt \to 0$:

$$dS^2 \to \sigma^2 S^2 dt$$

(14)

We substitute this in the formula for $dV$ and then get:

$$dV = \frac{\partial V}{\partial S} (\sigma S dX + \mu S dt) + \frac{\partial V}{\partial t} dt + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} dt = \sigma S \frac{\partial V}{\partial S} dX + \left( \mu S \frac{\partial V}{\partial S} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + \frac{\partial V}{\partial t} \right) dt$$

(15)

We call this equation the Black-Scholes equation. Now suppose we construct a portfolio of one option and an amount of $\Delta$ of the underlying asset. We call this portfolio $\Pi$ and its value is: $\Pi = V - \Delta S$. We can take a small time-step and then get that:

$$d\Pi = dV - \Delta dS$$

(16)

We insert the Black-Scholes formula and the formula for the underlying asset price into this formula and then get:

$$d\Pi = \sigma S \left( \frac{\partial V}{\partial S} - \Delta \right) dX + \left( \mu S \frac{\partial V}{\partial S} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + \frac{\partial V}{\partial t} - \mu \Delta S \right) dt$$

(17)

We can eliminate the random part of this equation by choosing $\Delta = \frac{\partial V}{\partial S}$. This results in a portfolio with the following formula:

$$d\Pi = \left( \frac{\partial V}{\partial t} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} \right) dt$$

(18)

We can use the assumptions we made earlier. There is no difference between the return on the portfolio and investing the money in a risk-free bank. Therefore, the change in $d\Pi$ should be equal to the amount that is returned by putting it in a bank account. We now have that $d\Pi = r\Pi dt$. This leaves us with the following:

$$r\Pi dt = \left( \frac{\partial V}{\partial t} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} \right) dt$$

(19)

Using the fact that $\Pi = V - \frac{\partial V}{\partial S} S$ gives us:

$$\frac{\partial V}{\partial t} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS \frac{\partial V}{\partial S} - rV = 0$$

(20)

This is called the Black-Scholes partial differential equation. An option has to satisfy this partial differential equation under the assumptions that we made earlier.

We can also look at the Black Scholes partial differential equation for American options which does not satisfy the earlier proposed assumptions. An American call option has the boundary condition: $C(S,T) \geq \max(S-K,0)$. We do not have the arbitrage argument for American options that we did have for the European options. This means that we now can have a difference between the return of investing in a bank account or buying options. This yields that the equality in the latter Black-Scholes partial differential equation is now an inequality:

$$\frac{\partial V}{\partial t} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS \frac{\partial V}{\partial S} - rV \leq 0$$

(21)
2.4 Feynman-Kac equation

We are going to link our parabolic differential equation, the Black-Scholes equation, with the stochastic process that is found in option pricing. The Feynman-Kac formula gives us a way to solve these problems. Consider the following PDE:

\[
\frac{\partial f}{\partial t} + \mu(x,t) \frac{\partial f}{\partial x} + \frac{1}{2}\sigma^2(x,t) \frac{\partial^2 f}{\partial x^2} = V(x,t)f
\]  

(22)

Defined for all real \(x\) and \(t\) on the interval \([0,T]\) with:

\[
f(x,T) = \psi(x)
\]  

(23)

In this equation \(\mu, \sigma, \psi\) and \(V\) are known functions. \(T\) is a parameter and \(f\) is an unknown function.

We apply the Feynman-Kac formula which gives us that:

\[
f(x,t) = \mathbb{E}\left[ e^{-\int_t^T V(X_{\tau}) d\tau} \psi(X_T) | X_t = x \right]
\]  

(24)

where \(X\) is an Ito process derived from:

\[
dX = \mu(X,t)dt + \sigma(X,t)dW
\]  

(25)

We have that \(W\) is a Wiener process and \(X(0) = x\).

Now we can use this formula on the Black-Scholes equation:

\[
\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2S^2 \frac{\partial^2 V}{\partial S^2} + rS \frac{\partial V}{\partial S} - rV = 0
\]  

(26)

We see that there is a great resemblance with the PDE we considered earlier. Here we have that:

\[
\mu(S,t) = rS
\]  

(27)

\[
\sigma(S,t) = \frac{1}{2}\sigma^2S
\]  

(28)

\[
V(S,t) = r
\]  

(29)

We can derive the function that links the PDE to the stochastic processes and we will call this function \(v(S,t)\):

\[
v(S,t) = \mathbb{E}[e^{-\int_t^T rX_{\tau} d\tau} \psi(X_T) | X_t = x]
\]  

(30)

\[
v(S,t) = e^{-r(T-t)} \mathbb{E}[\psi(X_T) | X_t = x]
\]  

(31)

We will now change the variables to the same notation as in article 1.

\[
v(x,t_0) = e^{-r\Delta t} \mathbb{E}[v(y,T) | x]
\]  

(32)

Here we have that \(\Delta t = T - t\). We are going to calculate the expectation value of \(v(y,T)\) by calculating the value of the integral of \(V(y,T)\) multiplied by its probability density function \(f(y|x)\) which leaves us with the following equation:

\[
v(x,t_0) = e^{-r\Delta t} \int_{\mathbb{R}} v(y,T) f(y|x) dy
\]  

(33)

We are going to solve this formula but this can not be done analytically so we have to use numerical approximations. We will consider a method in the next section called the COS-method.
3 The COS-method; A Fourier cosine series expansion method

As said earlier, we are going to solve the equation for $v(t_0)$ numerically. There is now a problem since we do not know the function $f(y|x)$. However, we do know the characteristic function of the probability function for the asset price. We do know the function $v(y,T)$ which is the payoff function given in equation (9) for a call option and in equation (10) for a put option. The density function, $f(x)$, and the characteristic function, $\phi(\omega)$, form a Fourier pair:

$$\phi(\omega) = \int_{\mathbb{R}} e^{ix\omega} f(x) dx$$  \hspace{1cm} (34)

$$f(x) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-ix\omega} \phi(\omega) d\omega$$  \hspace{1cm} (35)

We have to solve the inverse integral in the latter equation because we do know what $\phi(\omega)$ is but not what $f(x)$ is. We do this by using cosine expansions. This work is heavily based on the PhD thesis of F.Fang, [1]:

3.1 Density recovery

The main idea of recovering the function $f(x)$ by its characteristic function is that we reconstruct the whole integral from its Fourier cosine series expansion. For a function supported on $[0, \pi]$, the cosine expansion is:

$$f(\theta) = \sum_{k=0}^{\infty} A_k \cos(k\theta), \quad \text{with} \quad A_k = \frac{2}{\pi} \int_{0}^{\pi} f(\theta) \cos(k\theta) d\theta$$  \hspace{1cm} (36)

Here the summation sign $\sum'$ means that the first term of the summation is weighted by one-half. If we have a function on another interval than $[0, \pi]$ say $[a, b] \subset \mathbb{R}$, we have to change the parameters:

$$\theta = \frac{x-a}{b-a} \pi, \quad \text{and} \quad x = \frac{b-a}{\pi} \theta + a$$  \hspace{1cm} (37)

We then get that:

$$f(x) = \sum_{k=0}^{\infty} A_k \cos \left( \frac{k\pi}{b-a} x \right), \quad \text{with} \quad A_k = \frac{2}{b-a} \int_{a}^{b} f(x) \cos \left( \frac{k\pi}{b-a} x \right) dx$$  \hspace{1cm} (38)

We have truncated the interval but this will not be a problem if we take a big enough $[a, b]$ because the existence of a Fourier transform gives us that the value of the integrand decays to zero at $\pm \infty$. Suppose we have chosen a good $[a, b]$, then we have that:

$$\phi_1(\omega) = \int_{a}^{b} e^{ix\omega} f(x) dx \approx \int_{\mathbb{R}} e^{ix\omega} f(x) dx = \phi(\omega)$$  \hspace{1cm} (39)

We can put this approximation in the function of $A_k$:

$$A_k = \frac{2}{b-a} \Re \left\{ \phi_1 \left( \frac{k\pi}{b-a} \right) \cdot e^{-i \frac{k\pi}{b-a}} \right\}$$  \hspace{1cm} (40)

We take the real value because this is the cosine written in exponentials. We can use the approximation made earlier were $\phi_1(\omega) \approx \phi(\omega)$. We call this $F_k$ instead of $A_k$ and $A_k \approx F_k$:

$$F_k = \frac{2}{b-a} \Re \left\{ \phi \left( \frac{k\pi}{b-a} \right) \cdot e^{-i \frac{k\pi}{b-a}} \right\}$$  \hspace{1cm} (41)

We then replace $A_k$ by $F_k$ in the cosine series expansion of $f(x)$:

$$f_1(x) = \sum_{k=0}^{\infty} F_k \cos \left( \frac{k\pi}{b-a} x \right)$$  \hspace{1cm} (42)
We truncate the series summation and get:

\[ f_2(x) = \sum_{k=0}^{N-1} F_k \cos \left( k\pi \frac{x-a}{b-a} \right) \] (43)

We have an equation that can be solved numerically. The process described above is called the COS method and is very suitable for deriving the density function of an options payoff function.

### 3.2 COS-formula

We are going to use this method to derive the equation for European options. We start with the following formula:

\[ v(x,t_0) = e^{-r\Delta t} \int_{\mathbb{R}} v(y,T) f(y|x) dy \] (44)

Since the density decays rapidly to zero at \( \pm \infty \) we can truncate the region with \([a,b] \subset \mathbb{R}\) without losing a lot of accuracy. This gives us our first approximation:

\[ v_1(x,t_0) = e^{-r\Delta t} \int_{a}^{b} v(y,T) f(y|x) dy \] (45)

As said before, the density function is usually not known but the characteristics function is. We replace the density function by its cosine expansion in \( y \):

\[ f(y|x) = \sum_{k=0}^{\infty} A_k(x) \cos \left( k\pi \frac{y-a}{b-a} \right) \] (46)

with

\[ A_k = \frac{2}{b-a} \int_{a}^{b} f(y|x) \cos \left( k\pi \frac{y-a}{b-a} \right) dy \] (47)

These two functions give us the next approximation:

\[ v_1(x,t_0) = e^{-r\Delta t} \sum_{k=0}^{\infty} A_k(x) \int_{a}^{b} v(y,T) \cos \left( k\pi \frac{y-a}{b-a} \right) dy \] (48)

We can interchange summation and integration:

\[ v_1(x,t_0) = e^{-r\Delta t} \sum_{k=0}^{\infty} A_k(x) \int_{a}^{b} v(y,T) \cos \left( k\pi \frac{y-a}{b-a} \right) dy \] (49)

We introduce a new definition:

\[ V_k := \frac{2}{b-a} \int_{a}^{b} v(y,T) \cos \left( k\pi \frac{y-a}{b-a} \right) dy \] (50)

We substitute this definition in the approximation:

\[ v_1(x,t_0) = \frac{1}{2} (b-a) e^{-r\Delta t} \sum_{k=0}^{\infty} A_k(x) V_k \] (51)

The \( V_k \) are the Fourier cosine coefficients of \( v(y,T) \). These coefficients decay rapidly to zero so we can truncate the summation without losing significant accuracy:

\[ v_2(x,t_0) = \frac{1}{2} (b-a) e^{-r\Delta t} \sum_{k=0}^{N-1} A_k(x) V_k \] (52)
Just like in the previous section we can replace $A_k(x)$ by $F_k(x)$ which gives us:

$$v_3(x,t_0) = e^{-r\Delta t} \sum_{k=0}^{N-1} \text{Re} \left\{ \phi \left( \frac{k\pi}{b-a} : x \right) e^{-ik\frac{\pi}{\Delta t}} \right\} V_k$$

(53)

This is the COS formula for the underlying processes while pricing European Options. $V_k$ can be solved analytically for European options which makes this method to a very fast and accurate method.

The next step is to decide what $V_k$ should be in the COS-method. These $V_k$’s can be found analytically for several contracts. We will use the result stated below.

**Result** The cosine series coefficients, $\chi_k$, of $g(y) = e^y$ on $[c,d] \subset [a,b]$,

$$\chi_k(c,d) := \int_c^d e^y \cos \left( k\pi \frac{y-a}{b-a} \right) dy$$

(54)

and the cosine series coefficients, $\psi_k$, of $g(y) = 1$ on $[c,d] \subset [a,b]$,

$$\psi_k(c,d) := \int_c^d \cos \left( k\pi \frac{y-a}{b-a} \right) dy$$

(55)

are known analytically

Basic calculus shows us that:

$$\chi_k(c,d) = \frac{1}{1 + \left( \frac{k\pi}{b-a} \right)^2} \left[ \cos \left( k\pi \frac{d-a}{b-a} \right) e^d - \cos \left( k\pi \frac{c-a}{b-a} \right) e^c + \frac{k\pi}{b-a} \sin \left( k\pi \frac{d-a}{b-a} \right) e^d - \frac{k\pi}{b-a} \sin \left( k\pi \frac{c-a}{b-a} \right) e^c \right]$$

(56)

and

$$\psi_k(c,d) = \begin{cases} \frac{\sin \left( k\pi \frac{d-a}{b-a} \right)}{d-c} - \frac{\sin \left( k\pi \frac{c-a}{b-a} \right)}{k\pi} & \text{if } k \neq 0; \\ \frac{b-a}{k\pi} & \text{if } k = 0 \end{cases}$$

For the rest of the research we will define the asset’s price as the log-asset’s price with:

$$x := \ln \left( \frac{S_0}{K} \right) \quad \text{and} \quad y := \ln \left( \frac{S_T}{K} \right)$$

(57)

where $S_0$ is the asset’s price at time $t_0$, $S_T$ is the asset’s price at expire time and $K$ is the exercise price. The payoff function for an European option with the log-stock price is equal to:

$$v(y,t) \equiv \max(\alpha \cdot K(e^y - 1), 0) \quad \text{with} \quad \alpha = \begin{cases} 1 & \text{for a call;} \\ -1 & \text{for a put.} \end{cases}$$

(58)

Then we get the following expression for $V_k$ of an European call option.

$$V_k^{\text{call}} = \frac{2}{b-a} \int_0^b K(e^y - 1) \cos \left( k\pi \frac{y-a}{b-a} \right) dy = \frac{2}{b-a} K(\chi_k(0,b) - \psi_k(0,b))$$

(59)

We have the an expression for the characteristic function $\phi$:

$$\phi(\omega) = \exp(\omega(r - \frac{1}{2}\sigma^2)\Delta t - \frac{1}{2}\omega^2\sigma^2\Delta t)$$

(60)

This characteristic function can be used for the Geometric Brownian Motion (GBM) which will be shown in an example. We first need to find an expression for the truncation range $[a,b]$ which we find in [1]:

$$[a,b] := [c_1 - L\sqrt{c_2 + \sqrt{c_4}}, c_1 + L\sqrt{c_2 + \sqrt{c_4}}] \quad \text{with} \quad L = 10$$

(61)
with \( c_i \) the cumulants of the stochastic process:

\[
\begin{align*}
    c_1 &= \mu T \\
    c_2 &= \sigma^2 T \\
    c_4 &= 0
\end{align*}
\] (62-64)

We will see that the error convergence is exponential and therefore very fast when we use the COS method.

Now we will discuss the first example of this paper. The set of parameters is performed under the GBM model with a short time to maturity. Under GBM model we know the density function \( f(y|x) \) for European options exact. We can therefore calculate the value of these options analytically. We will now show the error convergence for the COS method applied to European options under the GBM model. The parameters are:

\[
S_0 = 100, r = 0.1, q = 0, T = 0.1, \sigma = 0.25
\] (65)

We will examine three different strike prices, \( K = 80, 100 \) and \( 120 \).

<table>
<thead>
<tr>
<th>S \ N</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>8.43 \times 10^{-3}</td>
<td>7.64 \times 10^{-8}</td>
<td>1.76 \times 10^{-12}</td>
<td>1.76 \times 10^{-12}</td>
<td>1.76 \times 10^{-12}</td>
</tr>
<tr>
<td>100</td>
<td>1.40 \times 10^{-3}</td>
<td>8.88 \times 10^{-8}</td>
<td>1.91 \times 10^{-14}</td>
<td>1.91 \times 10^{-14}</td>
<td>1.91 \times 10^{-14}</td>
</tr>
<tr>
<td>120</td>
<td>2.59 \times 10^{-3}</td>
<td>8.89 \times 10^{-8}</td>
<td>4.65 \times 10^{-14}</td>
<td>4.65 \times 10^{-14}</td>
<td>4.65 \times 10^{-14}</td>
</tr>
</tbody>
</table>

3.3 COS formula for the Heston Model

If we do not know the volatility of the underlying asset, which is usually the case in the financial world, we can also model this volatility by using the Heston model. Volatility is the square root of the variance of the underlying asset that we look at. This model gives us two differential equations:

\[
\begin{align*}
    dx_t &= (\mu - \frac{1}{2} \nu_t)dt + \sqrt{v_t}dW_{1t} \\
    dv_t &= \lambda(\bar{v} - v_t)dt + \eta \sqrt{v_t}dW_{2t}
\end{align*}
\] (66-67)

where \( x_t \) denotes the log-asset’s price variable and \( v_t \) the variance of the asset’s price process. The parameters have the following definitions: \( \lambda \) is the speed of mean reversion, \( \bar{v} \) is the mean level of variance and \( \eta \) is the volatility of the volatility. The mean reversion means that the asset’s price will always return to the average price of the asset. So when the asset’s price is very high or very low this parameter indicates how fast the asset price will return to the mean. The mean level of variance gives us the expected value of the variance in the asset price. The volatility of the volatility is the parameter that indicates the variance of the variance of the underlying asset price. The Brownian motions \( W_{1t} \) and \( W_{2t} \) are correlated with correlation factor \( \rho \).

The characteristic function of the log-asset’s price is:

\[
\phi_{hes}(\omega, v_0) = \exp(i\omega \mu \Delta t + \frac{v_0}{\eta} \left( \frac{1 - e^{-\rho \Delta t}}{1 - e^{-\rho \Delta t}} \right) (\lambda \rho, \omega - D)). \]

\[
\exp\left( \frac{\lambda \omega}{\eta} \left( \Delta t (\lambda \rho, \omega - D) - 2 \log(\frac{1 - G e^{-\rho \Delta t}}{1 - G}) \right) \right)
\] (68)

with

\[
D = \sqrt{(\lambda - i\rho \omega)^2 + (\omega^2 + i\omega \eta)^2} \quad \text{and} \quad G = \frac{\lambda \rho, \omega - D}{\lambda \rho, \omega + D}
\] (69)
We consider two different maturities: \( T = 1 \) and \( T = 10 \). From the work of F. Fang we get that a proper truncation range is:

\[
[a, b] := [c_1 - 12\sqrt{|c_2|}, c_1 + 12\sqrt{|c_2|}]
\]

Under the Heston model we have the following cumulants:

\[
c_1 = \mu T + (1 - e^{\lambda T}) \frac{e^{\nu_0} - e^{\bar{\nu}T}}{\lambda} - \frac{1}{2} \bar{\nu} T
\]

\[
c_2 = \frac{1}{8\lambda^3} (\eta T \lambda e^{-\lambda T} (v_0 - \bar{v})(8\lambda \rho - 4\eta) + \lambda \rho \eta (1 - e^{-\lambda T}) (16\bar{v} - 8v_0) + 2\bar{v} \lambda T (-4\lambda \rho \eta + \eta^2 + 4\lambda^2) + \\
\eta^2 ((\bar{v} - 2v_0)e^{-2\lambda T} + \bar{v}(6e^{-\lambda T} - 7) + 2v_0) + 8\lambda^2 (v_0 - \bar{v})(1 - e^{-\lambda T}))
\]

We will now look at an example with the Heston model using these parameters:

\[
S_0 = 100, K = 100, r = 0, q = 0, \lambda = 0, \eta = 0.5751, \bar{v} = 0.0398, v_0 = 0.0175, \rho = -0.5711
\]

Again we will see that the error convergence is exponential and the method is therefore very fast and accurate.

| Table 2: Error convergence for both tests using the COS method under the Heston model for European options. |
|-----------------|--------|--------|--------|--------|--------|--------|--------|
| S               | N      | 32     | 64     | 96     | 128    | 160    | 192    |
| T=1             | abs. error | 3.74 $10^{-12}$ | 4.05 $10^{-14}$ | 1.18 $10^{-16}$ | 2.33 $10^{-17}$ | 2.81 $10^{-17}$ | 2.70 $10^{-17}$ |
| T=10            | abs. error | 7.44 $10^{-12}$ | 2.98 $10^{-14}$ | 3.67 $10^{-16}$ | 8.80 $10^{-18}$ | 7.95 $10^{-10}$ | 4.50 $10^{-10}$ |

We see that for a longer time span the error convergence is slower for a small number of points. If we have \( N = 128 \) both test have a high error convergence. This are much fewer points in comparison to other pricing techniques like the Carr-Madan method that needs 2000 points for an error of order \( O(10^{-3}) \) when \( T = 10 \).
4 Heston model for the Bermudan options

The Heston stochastic volatility model can be expressed as the logarithm of the stock price and the variance. We find this model by applying Ito’s lemma. This model consists of the following equations:

\[ dx_t = \left( \mu - \frac{1}{2} \sigma_t \right) dt + \rho \sqrt{\sigma_t} dW_{1,t} + \sqrt{1 - \rho^2} \sqrt{\sigma_t} dW_{2,t} \]  
\[ dv_t = \lambda (\bar{v} - v_t) dt + \eta \sqrt{v_t} dW_{1,t} \]  

Again we have the three parameters that are non-negative: \( \lambda, \bar{v} \) and \( \eta \). These parameters represent respectively the speed of mean reversion, the mean level of variance and the volatility of the volatility. The Brownian processes \( W_{1,t} \) and \( W_{2,t} \) are independent. Parameter \( \rho \) is a correlation factor.

The Feller condition, \( 2\lambda \bar{v} \geq \eta^2 \), ensures that \( v_t \) does not become negative. From literature we get a variable \( q \) that indicates whether we meet the Feller condition and a variable \( \zeta \) that is used for the probability density function.

\[ q := \frac{2\lambda \bar{v}}{\eta^2} - 1, \quad \text{and} \quad \zeta := \frac{2\lambda}{(1 - e^{-\lambda(t-s)})\eta^2} \]  

We now have that \( 2\zeta v_t \sim \chi^2(q, 2q v_t e^{-\lambda(t-s)}) \) for \( 0 < s < t \). This is the non-central chi-square distribution. We can then derive the probability density function of \( v_t \) given \( v_s \):

\[ p_v(v_t|v_s) = \zeta e^{-\zeta (v_t e^{-\lambda(t-s)} + v_s)} \left( \frac{v_t}{v_s e^{-\lambda(t-s)}} \right)^{\frac{q}{2}} I_q \left( 2\zeta e^{-\frac{1}{2} \lambda(t-s)} \sqrt{v_s v_t} \right) \]  

Here \( I_q \) is the modified Bessel function of the first kind with order \( q \). In the rest of this paper this function will be very important since it is main cause for the high computational time for Bermudan options. The Feller condition is often not met in practice so in most cases we will have that \( q < 0 \). This causes a problem for the cumulative distribution of the variance. In the left tail, the variance density grows very fast which leads to significant errors in the computation of the option prices.

We will now look at the COS method for Bermudan options under the Heston model. First we will show a couple of formulas that are of great importance in determining the pricing method. The value of a Bermudan option contains of two values: the continuation value and the payoff value. We have \( M \) moments where we can decide whether to keep the option or use the right to exercise the option. This gives us the following formula for the value of the Bermudan option:

\[ v(x_m, \sigma_m, t_m) = \begin{cases} 
  g(x_m, t_m) & \text{for } m = M; \\
  \max[c(x_m, \sigma_m, t_m), g(x_m, t_m)] & \text{for } m = 1, 2, \ldots, M - 1; \\
  c(x_m, \sigma_m, t_m) & \text{for } m = 0;
\end{cases} \]  

Here we have that \( g(x_m, t_m) \) is the payoff function. At time \( t_M \) we are at the end of the lifespan of the option so the value of the option is now equal to the value of the payoff. The function \( c(x_m, \sigma_m, t_m) \) gives us the continuation value of the option. At the starting point of the option the value of the option is equal to the continuation value of the option since there is no point in exercising the option at the same moment as purchasing it.

In the appendix there is a complete explanation of how the COS method works for Bermudan options under the Heston model. Since this method is too complicated to derive here we will just discuss the parts that are of importance to us and summarise the steps that are made in the algorithm.
In the algorithm we find one appearance of the Bessel function that is of great interest to us. This Bessel function is of the following form:

\[ I_q \left( e^{\frac{1}{2}(\sigma_{m+1} + \sigma_m)} \cdot 2\kappa(q)e^{-\frac{1}{2}\gamma(q)\Delta t} \right) \]  

(78)

with:

\[ q = \omega \left( \frac{\lambda}{\eta} - \frac{1}{2} \right) + \frac{1}{2} i \omega^2 (1 - \rho^2) \quad \text{and} \quad \gamma(q) = \sqrt{\lambda^2 - 2i\eta q} \quad \text{and} \quad \kappa(q) = \frac{2\gamma(q)}{\eta^2 (1 - e^{-\gamma(q)\Delta t})} \]  

(79)

We will be focusing on this function and the replacement of this function by one of the approximations described in the next section.

5 Bessel function

In our algorithm we use the modified Bessel function of the first kind. This function is the solution to the following partial differential equation:

\[ z^2 \frac{d^2 y}{dz^2} + z \frac{dy}{dz} - (z^2 + q^2) y = 0 \]  

(80)

In this formula \( q \) is a real constant. The solutions are of the general form:

\[ I_q(z) = \left( \frac{z}{2} \right)^q \sum_{k=0}^{\infty} \frac{\left( \frac{z}{2} \right)^k}{k! \Gamma(q + k + 1)} \]  

(81)

Here \( \Gamma(q+k+1) \) denotes the normal Gamma function. The Bessel function has a high computational time in Matlab which leads to a long loop time in our algorithm, especially for negative values of \( q \). Therefore, we want to use approximations for the Bessel function that are found in [5] by N.Temme.

5.1 First approximation

When the absolute value of \( z \) is small we can approximate the Bessel function by truncating the summation from an infinity number of terms to a finite number of terms. This approximation only works in the first quadrant of the complex plane. We can however transform values from the fourth quadrant \((x > 0, y < 0)\) to the first quadrant by the following equation:

\[ I_q(z) = e^{-\text{abs}(\text{Re}(z))} I_q(z) \]  

(82)

In the special case that \( q = -\frac{1}{2} \) we can use the following approximation:

\[ I_q = \sqrt{\frac{2\pi}{\pi}} \cdot \frac{\cosh(z)}{z} \]  

(83)

For \( q \) different we have a range of applicability for \( z \) smaller than 2. Below we have described the algorithm that is used for the approximation.

Initialisation

- Calculate \( w := z^2/4 \).
- Determine \( t := w/(q + 1) \).
- Assign some variables: \( s := 1 + t, k := 1 \) and \( e := 1 \).
Loop while error is greater than tolerance

- Calculate $t := \frac{t_{k+1}}{(k+1)^{q+k+1}}$
- Assign: $s := s + t$.
- Determine the error: $\text{abs} \left( \frac{t}{1 + \text{abs}(s)} \right)$
- For the next loop we assign: $k := k + 1$.

Finalisation

- The value of the Bessel approximation is: $I := \left( \frac{1}{2} \right)^q \frac{s}{\Gamma(q+1)}$

5.2 Second approximation

We can also use an asymptotic expansion of the modified Bessel function of the first kind for values of $z$ in the first quadrant. This approximation is given by the formula below:

$$I_q(z) \sim \frac{e^z}{(2\pi z)^{1/2}} \left( \sum_{k=0}^{\infty} (-1)^k \frac{a_k(q)}{z^k} + i e^{\pi i} e^{-2z} \sum_{k=0}^{\infty} \frac{a_k(q)}{z^k} \right)$$

We can neglect the second part when $e^{-2z}$ is small enough. For this we use the following recurrence relation:

$$a_{k+1}(q) = q^2 - \left( k + \frac{1}{2} \right)^2 a_k(q)$$

Here we use that $a_0(q) = 1$ and $k = 0, 1, 2, \ldots$. This gives us the following algorithm that is accurate when $|z| \geq 18$. This algorithm can also be used for values in the fourth quadrant by transforming the values:

$$I_q(z) = e^{-\text{abs}(\text{Re}(z))} I_q(z)$$

Again we will show the steps that we make in the algorithm:
Initialisation

- Assign some dummy variables that we will be used in the loop: \( t := 1, s := -1, \text{ser}1 := 1, \text{ser}2 := 1, q2 := q^2, k := 0, e := 1 \) and \( kmax := 2 + \text{trunc}(\text{abs}(2 \times z)) \). Here we have that trunc means that we round the value to the closest integer.

Loop for error greater than tolerance and \( k \) is smaller than \( kmax \).

- Calculate \( t := t \times \left( \frac{q^2-(k+0.5)^2}{2qz(k+1)} \right) \).
- Determine \( \text{ser}1 := \text{ser}1 + s \times t \) and \( \text{ser}2 = \text{ser}2 + t \).
- Calculate \( e := \text{abs} \left( \frac{1}{\text{ser}1} \right) + \text{abs} \left( \frac{1}{\text{ser}2} \right) \).
- For the next loop we assign: \( s := -s \) and \( k := k + 1 \).

Finalisation

- We determine the value of this Bessel approximation: \( I := e^z \times \frac{\text{ser}1 + z \times \text{exp}(-2qz + q \times z) \times \text{ser}2}{\sqrt{2qz}} \)

We now have to look at the regions at which we can use the approximations without losing accuracy. For the first series expansion we see that if \( q = \pm 1 \) we have to stay close to the origin for \( y \) and can take any value for \( x \). If we have that \( q = -\frac{1}{2} \), we can always use the approximation in a very wide range. In any other case for \( q \) we have to check these conditions for our approximation: \( y \) has to be from the interval \([-2, 2]\).

We now look at the second approximation to check the range for a proper interval. For every value of \( q \) we can apply the second approximation in all quadrants of the complex plane except for the third \((x < 0, y < 0)\). Furthermore we have to assign an tolerance level to the approximations. This tolerance level is taken very large since there is a very rapid decay in the values of the terms. We take the tolerance level therefore \(10^{-10}\).

<table>
<thead>
<tr>
<th>x</th>
<th>(-20)</th>
<th>(-15)</th>
<th>(-10)</th>
<th>(-5)</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(+01)</td>
<td>(+00)</td>
<td>(+01)</td>
<td>(+17)</td>
<td>+02</td>
<td>+01</td>
<td>+00</td>
<td>+01</td>
<td>+01</td>
</tr>
<tr>
<td>2</td>
<td>(+02)</td>
<td>(+00)</td>
<td>(+01)</td>
<td>(+17)</td>
<td>+02</td>
<td>+01</td>
<td>+00</td>
<td>+01</td>
<td>+01</td>
</tr>
<tr>
<td>3</td>
<td>(+03)</td>
<td>(+00)</td>
<td>(+01)</td>
<td>(+17)</td>
<td>+02</td>
<td>+01</td>
<td>+00</td>
<td>+01</td>
<td>+01</td>
</tr>
<tr>
<td>4</td>
<td>(+04)</td>
<td>(+00)</td>
<td>(+01)</td>
<td>(+17)</td>
<td>+02</td>
<td>+01</td>
<td>+00</td>
<td>+01</td>
<td>+01</td>
</tr>
<tr>
<td>5</td>
<td>(+05)</td>
<td>(+00)</td>
<td>(+01)</td>
<td>(+17)</td>
<td>+02</td>
<td>+01</td>
<td>+00</td>
<td>+01</td>
<td>+01</td>
</tr>
<tr>
<td>6</td>
<td>(+06)</td>
<td>(+00)</td>
<td>(+01)</td>
<td>(+17)</td>
<td>+02</td>
<td>+01</td>
<td>+00</td>
<td>+01</td>
<td>+01</td>
</tr>
<tr>
<td>7</td>
<td>(+07)</td>
<td>(+00)</td>
<td>(+01)</td>
<td>(+17)</td>
<td>+02</td>
<td>+01</td>
<td>+00</td>
<td>+01</td>
<td>+01</td>
</tr>
<tr>
<td>8</td>
<td>(+08)</td>
<td>(+00)</td>
<td>(+01)</td>
<td>(+17)</td>
<td>+02</td>
<td>+01</td>
<td>+00</td>
<td>+01</td>
<td>+01</td>
</tr>
</tbody>
</table>

Table 3: Convergence in \( x \) and \( y \) for Inauxpower(first approximation) with \( q = -1 \).

<table>
<thead>
<tr>
<th>x</th>
<th>(-3)</th>
<th>(-2)</th>
<th>(-1)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(-01)</td>
<td>(-17)</td>
<td>(-16)</td>
<td>(-18)</td>
<td>+16</td>
<td>+17</td>
<td>+01</td>
</tr>
<tr>
<td>2</td>
<td>(-01)</td>
<td>(-17)</td>
<td>(-16)</td>
<td>(-18)</td>
<td>+16</td>
<td>+17</td>
<td>+01</td>
</tr>
<tr>
<td>3</td>
<td>(-01)</td>
<td>(-17)</td>
<td>(-16)</td>
<td>(-18)</td>
<td>+16</td>
<td>+17</td>
<td>+01</td>
</tr>
</tbody>
</table>

Table 4: Convergence in \( x \) and \( y \) for Inauxpower(first approximation) with \( q = -1 \).
Table 5: Convergence in x and y for Inuzasymp(second approximation) with $q = +0.6$.

<table>
<thead>
<tr>
<th>y</th>
<th>-20</th>
<th>-15</th>
<th>-10</th>
<th>-5</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>-20</td>
<td>$10^{+01}$</td>
<td>$10^{+00}$</td>
<td>$10^{-04}$</td>
<td>$10^{-02}$</td>
<td>$10^{-15}$</td>
<td>$10^{-02}$</td>
<td>$10^{-01}$</td>
<td>$10^{+00}$</td>
<td>$10^{+01}$</td>
</tr>
<tr>
<td>-15</td>
<td>$10^{+02}$</td>
<td>$10^{+00}$</td>
<td>$10^{-01}$</td>
<td>$10^{-02}$</td>
<td>$10^{-15}$</td>
<td>$10^{-02}$</td>
<td>$10^{-01}$</td>
<td>$10^{+00}$</td>
<td>$10^{+02}$</td>
</tr>
<tr>
<td>-10</td>
<td>$10^{+03}$</td>
<td>$10^{+00}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-15}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+03}$</td>
</tr>
<tr>
<td>-5</td>
<td>$10^{+04}$</td>
<td>$10^{+00}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-15}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+04}$</td>
</tr>
<tr>
<td>0</td>
<td>$10^{+05}$</td>
<td>$10^{+00}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-15}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+06}$</td>
</tr>
<tr>
<td>5</td>
<td>$10^{+00}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-15}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+04}$</td>
</tr>
<tr>
<td>10</td>
<td>$10^{+00}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-15}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+03}$</td>
</tr>
<tr>
<td>15</td>
<td>$10^{+00}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-15}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+02}$</td>
</tr>
<tr>
<td>20</td>
<td>$10^{+00}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-15}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+01}$</td>
</tr>
</tbody>
</table>

Table 6: Convergence in x and y for Inuzasymp(second approximation) with $q = -1$.

<table>
<thead>
<tr>
<th>y</th>
<th>-20</th>
<th>-15</th>
<th>-10</th>
<th>-5</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>-20</td>
<td>$10^{+01}$</td>
<td>$10^{+00}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-18}$</td>
</tr>
<tr>
<td>-15</td>
<td>$10^{+00}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-14}$</td>
<td>$10^{-16}$</td>
<td>$10^{-16}$</td>
<td>$10^{-16}$</td>
<td>$10^{-16}$</td>
<td>$10^{-16}$</td>
</tr>
<tr>
<td>-10</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-10}$</td>
<td>$10^{-12}$</td>
<td>$10^{-12}$</td>
<td>$10^{-12}$</td>
<td>$10^{-12}$</td>
<td>$10^{-16}$</td>
</tr>
<tr>
<td>-5</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-05}$</td>
<td>$10^{-08}$</td>
<td>$10^{-12}$</td>
<td>$10^{-12}$</td>
<td>$10^{-12}$</td>
<td>$10^{-16}$</td>
</tr>
<tr>
<td>0</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-01}$</td>
<td>$10^{-05}$</td>
<td>$10^{-08}$</td>
<td>$10^{-12}$</td>
<td>$10^{-12}$</td>
<td>$10^{-16}$</td>
</tr>
<tr>
<td>5</td>
<td>$10^{-06}$</td>
<td>$10^{-05}$</td>
<td>$10^{-05}$</td>
<td>$10^{-05}$</td>
<td>$10^{-05}$</td>
<td>$10^{-08}$</td>
<td>$10^{-12}$</td>
<td>$10^{-12}$</td>
<td>$10^{-16}$</td>
</tr>
<tr>
<td>10</td>
<td>$10^{-10}$</td>
<td>$10^{-10}$</td>
<td>$10^{-10}$</td>
<td>$10^{-10}$</td>
<td>$10^{-10}$</td>
<td>$10^{-12}$</td>
<td>$10^{-14}$</td>
<td>$10^{-14}$</td>
<td>$10^{-18}$</td>
</tr>
<tr>
<td>15</td>
<td>$10^{-14}$</td>
<td>$10^{-14}$</td>
<td>$10^{-14}$</td>
<td>$10^{-14}$</td>
<td>$10^{-14}$</td>
<td>$10^{-16}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
</tr>
<tr>
<td>20</td>
<td>$10^{-18}$</td>
<td>$10^{-18}$</td>
<td>$10^{-18}$</td>
<td>$10^{-18}$</td>
<td>$10^{-18}$</td>
<td>$10^{-18}$</td>
<td>$10^{-18}$</td>
<td>$10^{-18}$</td>
<td>$10^{-18}$</td>
</tr>
</tbody>
</table>

Table 7: Convergence in x and y for Inuzasymp(second approximation) with $q = -1$.

<table>
<thead>
<tr>
<th>y</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>$10^{+01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+04}$</td>
<td>$10^{+04}$</td>
<td>$10^{-05}$</td>
<td>$10^{-05}$</td>
</tr>
<tr>
<td>-2</td>
<td>$10^{+01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+04}$</td>
<td>$10^{+03}$</td>
<td>$10^{-04}$</td>
<td>$10^{-05}$</td>
</tr>
<tr>
<td>-1</td>
<td>$10^{+01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+04}$</td>
<td>$10^{+03}$</td>
<td>$10^{-04}$</td>
<td>$10^{-05}$</td>
</tr>
<tr>
<td>0</td>
<td>$10^{+01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+01}$</td>
<td>$10^{+04}$</td>
<td>$10^{+03}$</td>
<td>$10^{-04}$</td>
<td>$10^{-05}$</td>
</tr>
<tr>
<td>1</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{+04}$</td>
<td>$10^{+03}$</td>
<td>$10^{-04}$</td>
<td>$10^{-05}$</td>
</tr>
<tr>
<td>2</td>
<td>$10^{+03}$</td>
<td>$10^{+03}$</td>
<td>$10^{+03}$</td>
<td>$10^{+04}$</td>
<td>$10^{+03}$</td>
<td>$10^{-04}$</td>
<td>$10^{-05}$</td>
</tr>
<tr>
<td>3</td>
<td>$10^{+03}$</td>
<td>$10^{+03}$</td>
<td>$10^{+03}$</td>
<td>$10^{+04}$</td>
<td>$10^{+03}$</td>
<td>$10^{-04}$</td>
<td>$10^{-05}$</td>
</tr>
</tbody>
</table>

Table 8: Convergence in x and y for Inuzasymp(second approximation) with $q = +0.6$.

<table>
<thead>
<tr>
<th>y</th>
<th>-20</th>
<th>-15</th>
<th>-10</th>
<th>-5</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>-20</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{-15}$</td>
<td>$10^{-15}$</td>
<td>$10^{-18}$</td>
<td>$10^{-17}$</td>
<td>$10^{-18}$</td>
</tr>
<tr>
<td>-15</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{-15}$</td>
<td>$10^{-15}$</td>
<td>$10^{-18}$</td>
<td>$10^{-17}$</td>
<td>$10^{-18}$</td>
</tr>
<tr>
<td>-10</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{-15}$</td>
<td>$10^{-15}$</td>
<td>$10^{-18}$</td>
<td>$10^{-17}$</td>
<td>$10^{-18}$</td>
</tr>
<tr>
<td>-5</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{+02}$</td>
<td>$10^{-15}$</td>
<td>$10^{-15}$</td>
<td>$10^{-18}$</td>
<td>$10^{-17}$</td>
<td>$10^{-18}$</td>
</tr>
<tr>
<td>0</td>
<td>$10^{+06}$</td>
<td>$10^{+06}$</td>
<td>$10^{+06}$</td>
<td>$10^{+06}$</td>
<td>$10^{-11}$</td>
<td>$10^{+09}$</td>
<td>$10^{-12}$</td>
<td>$10^{-12}$</td>
<td>$10^{-15}$</td>
</tr>
<tr>
<td>5</td>
<td>$10^{-10}$</td>
<td>$10^{-10}$</td>
<td>$10^{-10}$</td>
<td>$10^{-10}$</td>
<td>$10^{-11}$</td>
<td>$10^{+09}$</td>
<td>$10^{-12}$</td>
<td>$10^{-12}$</td>
<td>$10^{-15}$</td>
</tr>
<tr>
<td>10</td>
<td>$10^{-15}$</td>
<td>$10^{-15}$</td>
<td>$10^{-15}$</td>
<td>$10^{-15}$</td>
<td>$10^{-15}$</td>
<td>$10^{-15}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
</tr>
<tr>
<td>15</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
</tr>
</tbody>
</table>
6 Numerical results

We will now look at the error convergence and computation time of the COS-method in the case we use Bermudan options. We will compare the results of the normal COS-method with the COS-method that uses the Bessel approximation. From the literature, we get three test that we use for the comparison. These test have the following parameters:

- Test 1: \( \eta = 0.5, \lambda = 5, \bar{\nu} = 0.04, T = 1. \)
- Test 2: \( \eta = 0.5, \lambda = 0.5, \bar{\nu} = 0.04, T = 1. \)
- Test 3: \( \eta = 1, \lambda = 0.5, \bar{\nu} = 0.04, T = 10. \)

The first test satisfies the Feller condition so \( q > 0. \) The other two tests do not satisfy this condition and we will see that the computational time is then much larger. All three test have parameters for a put option that also suffices the following parameters for all three tests:

\[
\rho = -0.9, \nu_0 = 0.04, S_0 = 100, K = 100, r = 0
\] 

Furthermore, there will not be any dividend payments taken into account.

First we will look at test 1 which satisfies the Feller condition. In all the tests we will consider a Bermudan option that has 12 points where we can decide to exercise or keep the option. The number of COS terms is also kept constant in the first test. We will change the tolerance level of the error and the size of \( J \) which is the number of quadrature points in the variance dimension. In the first table we see the results when we calculate the value of this option using the COS method with the regular Bessel function. The table under this first table shows the results when we use the COS method with the Bessel approximation.

### Table 9: Convergence in \( J \) for Test No.1 \( (q = 0.6) \) with \( N = 2^7, M = 12 \) and the European option reference value is 7.5789038982.

<table>
<thead>
<tr>
<th>( J = 2^d )</th>
<th>Fourier cosine expansion plus Gauss-Legendre Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d )</td>
<td>time(sec)</td>
</tr>
<tr>
<td>4</td>
<td>0.38</td>
</tr>
<tr>
<td>5</td>
<td>0.48</td>
</tr>
<tr>
<td>6</td>
<td>1.72</td>
</tr>
<tr>
<td>7</td>
<td>7.06</td>
</tr>
</tbody>
</table>

### Table 10: Convergence in \( J \) for Test No.1 \( (q = 0.6) \) with \( N = 2^7, M = 12 \) and the European option reference value is 7.5789038982.

<table>
<thead>
<tr>
<th>( J = 2^d )</th>
<th>Fourier cosine expansion with Bessel approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d )</td>
<td>time(sec)</td>
</tr>
<tr>
<td>4</td>
<td>0.12</td>
</tr>
<tr>
<td>5</td>
<td>0.39</td>
</tr>
<tr>
<td>6</td>
<td>1.55</td>
</tr>
<tr>
<td>7</td>
<td>6.53</td>
</tr>
</tbody>
</table>

From the tables above we can see that for a test that suffices the Feller condition we only have a small improvement in the computational time and the error convergence is the same. Nevertheless, the old COS method should be changed to the one that uses the Bessel approximation since it is faster.

We are more interested in determining whether or not the Bessel approximation works properly when the Feller condition does not hold because in this case the normal Bessel function has a large computational time. We will have to increase the number of COS terms to get a good result. We will leave the tolerance
level the same for all test. We will increase J and look at the convergence of \( q \) to \(-1\). The first table shows the error and computational time for the second and third test without the approximation of the Bessel function.

Table 11: Convergence in \( J \) as \( q \to -1 \); Fourier cosine expansion plus Gauss-Legendre rule, \( N = 2^8, M = 12, TOL = 10^{-7} \), European reference values are 6.2710582179 (Test No. 2) and 13.0842710701 (Test No. 3).

<table>
<thead>
<tr>
<th>((J = 2^4))</th>
<th>Test No. 2 ((q = -0.84))</th>
<th>Test No. 3 ((q = -0.96))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d)</td>
<td>time(secs)</td>
<td>time(secs)</td>
</tr>
<tr>
<td></td>
<td>total Loop error</td>
<td>total Init. Loop error</td>
</tr>
<tr>
<td>6</td>
<td>3.61 3.27 0.34 5.63</td>
<td>3.42 3.11 0.31 -22.7</td>
</tr>
<tr>
<td>7</td>
<td>14.1 13.43 0.65 6.89 (10^{-3})</td>
<td>12.9 12.29 0.61 -8.51 (10^{-2})</td>
</tr>
<tr>
<td>8</td>
<td>62.3 56.78 5.49 -2.12 (10^{-5})</td>
<td>57.3 52.18 5.10 -1.60 (10^{-3})</td>
</tr>
</tbody>
</table>

In the table below we look at the error convergence and the computational time of the second and third test with the difference that now we apply the two Bessel approximations in our code. All other variables are taken the same and the test is done on the same computer as the tests above.

Table 12: Convergence in \( J \) as \( q \to -1 \); Fourier cosine expansion with Bessel approximation, \( N = 2^8, M = 12, TOL = 10^{-7} \), European reference values are 6.2710582179 (Test No. 2) and 13.0842710701 (Test No. 3).

<table>
<thead>
<tr>
<th>((J = 2^4))</th>
<th>Test No. 2 ((q = -0.84))</th>
<th>Test No. 3 ((q = -0.96))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d)</td>
<td>time(secs)</td>
<td>time(secs)</td>
</tr>
<tr>
<td></td>
<td>total Loop error</td>
<td>total Init. Loop error</td>
</tr>
<tr>
<td>6</td>
<td>2.1 1.80 0.28 5.63</td>
<td>2.20 1.91 0.29 -22.7</td>
</tr>
<tr>
<td>7</td>
<td>7.6 6.97 0.58 6.89 (10^{-3})</td>
<td>7.98 7.42 0.56 -9.06 (10^{-2})</td>
</tr>
<tr>
<td>8</td>
<td>36.4 31.43 5.00 -2.12 (10^{-5})</td>
<td>38.5 33.4 5.09 -8.71 (10^{-3})</td>
</tr>
</tbody>
</table>

We see that when the Feller condition is not met we definitely want to use the Bessel approximation. The computational time is reduced with more than 40% for the second test and for the third test the time is reduced by more than 30%. The error convergence is just as good for the Bessel approximation as for the Bessel function.

In the figures below we show the input values for the Bessel function for the different test. Most of the input is near the origin. These values fall in the range of applicability of the first approximation. Since all values are in the fourth quadrant of the complex plane we will have to transform them to the first quadrant as explained in the section about the Bessel function. The second approximation is also used but this approximation only handles about 1% of all the input for the Bessel function because most values have an absolute value smaller than 18.
Figure 1: Input for the Bessel approximation (test 1)
Figure 2: Input for the Bessel approximation (test 2)
Figure 3: Input for the Bessel approximation (test 3)
7 Conclusion and recommendations

In this research we have examined the COS method which is a Fourier cosine series expansion method. This numerical method can be used for pricing options for which we know the characteristic function but not the function itself. This is usually the case for describing the behaviour of options and their underlying assets. The COS method was used for determining the prices of options under the Heston model that also models the volatility of the underlying asset.

We were especially interested in the Bessel function which is found in the COS method for Bermudan options. The main goal was to accelerate the computation of this function by using proper approximations. This goal has been met with an improvement of 40\% for one of our test cases. In another test case we improved the computational time by about 30\%. In fact the order of the error convergence did not change at all so the approximations used are highly accurate.

The research could be further extended by not only looking at the improvement of the COS method when pricing Bermudan options but also look at other sorts of options like American and Asian options. The pricing of American options is more complex so the replacement of the Bessel function might show significant improvement in computational time.

Another aspect that could be examined is the pricing of options under the Heston-Hull-White model which is a more elaborate model in comparison to the Heston model. In this model we have that the interest rate is also stochastic which describes the financial world even better. It might very well be possible that the COS method is also applicable to this model and that the approximation of the Bessel function could speed up the computation.
References


A COS-method for Bermudan options

Since the density function decays very rapidly to zero for \( y \to \pm \infty \), we can truncate the integration range to \([a, b] \subset \mathbb{R}\) without losing accuracy. We predefine a tolerance level \( TOL \) for which we have that:

\[
\int_{\mathbb{R} \setminus [a, b]} f(y|x) dy < TOL 
\]  

We can now use the normal COS method for European options for finding the pricing for keeping our option at a point \( t_{m-1} \):

\[
\hat{v}(x, t_{m-1}) = e^{-r\Delta t} \sum_{k=0}^{N-1} \text{Re} \left\{ \phi(\frac{k\pi}{b-a}) e^{-ik\pi \frac{x}{b-a}} \right\} V_k(t_m) 
\]

If we have an exponential Levy process, this formula can be simplified to:

\[
\hat{v}(x, t_{m-1}) = e^{-r\Delta t} \sum_{k=0}^{N-1} \text{Re} \left\{ \phi_{\text{levy}}(\frac{k\pi}{b-a}) e^{-ik\pi \frac{x}{b-a}} \right\} V_k(t_m) 
\]

with \( \phi_{\text{levy}}(\omega) = \phi_{\text{levy}}(\omega; 0) \). We can also approximate \( v(x, t_0) \) by:

\[
\hat{v}(x, t_0) = e^{-r\Delta t} \sum_{k=0}^{N-1} \text{Re} \left\{ \phi_{\text{levy}}(\frac{k\pi}{b-a}) e^{ik\pi \frac{x}{b-a}} \right\} V_k(t_1) 
\]

For the pricing of Bermudan options it is important to compute the \( V_k(t_1) \) which are the cosine coefficients of the option value at time \( t_1 \). By inserting these coefficients into the formula for \( \hat{v}(x, t_0) \), we get the option value. The normal definition of \( V_k(t_m) \) is:

\[
V_k(t_m) := \frac{2}{b-a} \int_a^b v(y, t_m) \cos \left( \frac{k\pi y}{b-a} - \alpha \right) dy
\]

We can split this integral in two parts where \( x_m^* \) is the early exercise point at time \( t_m \). We define this point as the point where the continuation value equals the payoff, so \( c(x_m^*, t_m) = g(x_m^*, t_m) \). Since we look at the log asset’s price we get the following formula for \( g(x_m^*, t_m) \):

\[
g(x_m^*, t_m) = \max(\alpha K(e^x - 1), 0), \quad \alpha = \begin{cases} 
1 & \text{for a call} \\
-1 & \text{for a put}
\end{cases}
\]

The formula for \( V_k(t_m) \) can then be split into two intervals. One interval is: \([a, x_m^*]\) and the other interval is: \([x_m^*, b]\). On the first interval we have the payoff function and on the other interval the continuation value:

\[
V_k(t_m) = \begin{cases} 
C_k(a, x_m^*, t_m) + G_k(x_m^*, b), & \text{for a call} \\
G_k(a, x_m^*) + C_k(x_m^*, b, t_m), & \text{for a put}
\end{cases}
\]

for \( m = M - 1, M - 2, ..., 1 \) and:

\[
V_k(t_M) = \begin{cases} 
G_k(0, b), & \text{for a call} \\
G_k(a, 0), & \text{for a put}
\end{cases}
\]

We use the same formula for \( G_k \) and \( C_K \) as we used for the \( V_k \)'s for the European options.

\[
G_k(x_1, x_2) := \frac{2}{b-a} \int_{x_1}^{x_2} g(x, t_m) \cos \left( \frac{k\pi x}{b-a} - \alpha \right) dx
\]

\[
C_k(x_1, x_2, t_m) := \frac{2}{b-a} \int_{x_1}^{x_2} c(x, t_m) \cos \left( \frac{k\pi x}{b-a} - \alpha \right) dx
\]
We can determine $G_k(x_1, x_2)$ analytically since we know the formula for $g(x, t_m)$:

$$G_k(x_1, x_2) = \frac{2}{b-a} \alpha K[\chi_k(x_1, x_2) - \psi_k(x_1, x_2)] \quad (96)$$

where $\alpha, \chi_k$ and $\psi_k$ are the same as used earlier.

We are going to derive the formulas for $V_j(t_m)$ the Fourier cosine coefficients of the option value, with $j = 0, 1, ..., N-1$ and $m = 1, 2, ..., M$. At time $t_M$ this formula is exact and for time $t_{M-1}$ we can use the approximation: $\hat{c}(x, t_{M-1})$ which we insert in the formula for $C_k(x_1, x_2, t_m)$. This gives us as a result:

$$\hat{C}_k(x_1, x_2, t_{M-1}) = e^{-r\Delta t} \text{Re} \left\{ \sum_{j=0}^{N-1} \phi_{levy} \left( \frac{j\pi}{b-a} \right) V_j(t_M) \cdot M_{k,j}(x_1, x_2) \right\} \quad (97)$$

with the coefficients $M_{k,j}(x_1, x_2)$:

$$M_{k,j}(x_1, x_2) := \frac{2}{b-a} \int_{x_1}^{x_2} e^{ij\pi \frac{x-a}{b-a}} \cos \left( k\pi \frac{x-a}{b-a} \right) dx. \quad (98)$$

We have an exact expression for $V_j(t_M)$. However, we have to determine the continuation value at $M$ different point so we have to approximate the value $V_j(t_M)$ for all points: $M - 2, M - 3, \ldots, 1$. We then get the following approximation:

$$\hat{C}_k(x_1, x_2, t_m) = e^{-r\Delta t} \text{Re} \left\{ \sum_{j=0}^{N-1} \phi_{levy} \left( \frac{j\pi}{b-a} \right) \hat{V}_j(t_{m+1}) \cdot M_{k,j}(x_1, x_2) \right\} \quad (99)$$

We can determine the approximation $\hat{V}_k(t_m)$ with the approximation of $\hat{C}_k(x_1, x_2, t_m)$. This can be done for multiple coefficients at the same time so we can write this as a vector. So the Fourier cosine coefficients are:

$$\hat{V}_k(t_m) = \begin{cases} \hat{C}_k(a, x_m^*, t_m) + G_k(x_m^*, b), & \text{for a call} \\ G_k(a, x_m^*) + \hat{C}_k(x_m^*, b, t_m), & \text{for a put} \end{cases}$$

The bold letters are vectors. We now have an algorithm for constructing the price of a Bermudan option although this is not the fastest way of constructing the Fourier cosine coefficients.

**Initialisation** For $k = 0 \ldots N-1$:

- $V_k(t_M) = G_k(0, b)$ for call options and $V_k(t_M) = G_k(a, 0)$ for put options

**Loop** For $m=M-1 \ldots 1$:

- Determine an early-exercise point $x_m^*$
- Determine $\hat{V}_k(t_m)$

**Final step** Insert $\hat{V}_k(t_1)$ into the equation for $\hat{v}(x, t_0)$

### A.1 Algorithm for finding the Fourier cosine coefficients

In the previous section we have shown how the COS-method works for Bermudan options. We will now show that the computation of the Fourier cosine coefficients can be done through a fast algorithm. The important part is that the matrix $M_{k,j}$ is of a special form that will be explained below.

As we have seen in the previous section:

$$M_{k,j}(x_1, x_2) := \frac{2}{b-a} \int_{x_1}^{x_2} e^{ij\pi \frac{x-a}{b-a}} \cos \left( k\pi \frac{x-a}{b-a} \right) dx. \quad (100)$$
We can write the cosine as a sum of two exponentials and then get that:

\[ M_{k,j}(x_1, x_2) := \frac{2}{b-a} \int_{x_1}^{x_2} e^{ij\pi \frac{x-a}{b-a}} \left( \frac{1}{2} (e^{i\pi \frac{x-a}{b-a}} + e^{-i\pi \frac{x-a}{b-a}}) \right) dx. \]  

Furthermore, we can multiply the exponentials:

\[ M_{k,j}(x_1, x_2) := \frac{1}{b-a} \int_{x_1}^{x_2} e^{i(j+k)\pi \frac{x-a}{b-a}} + e^{i(j-k)\pi \frac{x-a}{b-a}} dx. \]  

We can split this expression in two integrals which are simple to calculate. This gives us:

\[ M_{k,j}(x_1, x_2) := \frac{1}{b-a} \left[ \left( \frac{1}{j+k} e^{i(j+k)\pi \frac{x-a}{b-a}} \right)_{x_1}^{x_2} + \left( \frac{1}{j-k} e^{i(j-k)\pi \frac{x-a}{b-a}} \right)_{x_1}^{x_2} \right] \]

The first part between the square brackets will be the coefficients of the matrix \( M_{k,j}^1 \) and the second part between the brackets will be defined as the matrix \( M_{k,j}^2 \). For the first matrix we have to separately calculate the coefficients for \( k = j = 0 \). The second matrix has different coefficients when \( k = j \). This gives us the following definition for the two different matrices.

\[ M_{k,j}^1 = \begin{cases} \frac{(x_2-x_1)\pi i}{b-a} & k = j = 0, \\ \frac{1}{j+k} \left( e^{i(j+k)\pi \frac{x_2-x_1}{b-a}} - e^{i(j+k)\pi \frac{x_1-x_2}{b-a}} \right) & \text{otherwise}. \end{cases} \]  

The second matrix has these coefficients:

\[ M_{k,j}^2 = \begin{cases} \frac{(x_2-x_1)\pi i}{b-a} & k = j, \\ \frac{1}{j-k} \left( e^{i(j-k)\pi \frac{x_2-x_1}{b-a}} - e^{i(j-k)\pi \frac{x_1-x_2}{b-a}} \right) & \text{otherwise}. \end{cases} \]  

We can substitute this definition into the formula for \( \hat{C}_k(x_1, x_2, t_m) \). The new formula becomes:

\[ \hat{C}_k(x_1, x_2, t_m) = e^{t\Delta t} \pi \text{Im} \left\{ (M_c + M_s)u \right\} \]  

In this equation \( \text{Im}\{\cdot\} \) denotes the imaginary part of the input. The input argument is an matrix-vector multiplication where \( u \) is a vector:

\[ u := \{ u_j \}_{j=0}^{N-1}, \quad u_j := \phi \left( \frac{j\pi}{b-a} \right) \hat{V}_j(t_{m+1}), \quad u_0 = \frac{1}{2} \phi(0)V_0(t_{m+1}) \]  

The matrices that we have split \( M_{k,j} \) into are of a special form. On this special form of matrices we can use the fast Fourier transform algorithm to enhance the computational time of the Fourier cosine coefficients. The first matrix \( M_c \) is called a Hankel matrix and is of the standard form below:

\[ M_c = \begin{bmatrix} m_0 & m_1 & m_2 & \cdots & m_{N-1} \\ m_1 & m_2 & \cdots & \cdots & m_N \\ \vdots & \ddots & \ddots & \cdots & \vdots \\ m_{N-2} & m_{N-3} & \cdots & \cdots & m_{2N-3} \\ m_{N-1} & m_N & \cdots & m_{2N-3} & m_{2N-2} \end{bmatrix} \]
The other matrix $M_s$ is called a Toeplitz matrix which is of the form below:

$$M_s = \begin{bmatrix}
m_0 & m_1 & \ldots & m_{N-2} & m_{N-1} \\
m_{-1} & m_0 & m_1 & \ldots & m_{N-2} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
m_{2-N} & m_{1-N} & \ldots & m_0 & m_1 \\
m_{1-N} & m_{2-N} & \ldots & m_{-1} & m_0
\end{bmatrix}$$

Both these matrices have the following coefficients:

$$m_j := \begin{cases} 
\frac{(x_2-x_1)\pi i}{b-a} & j = 0, \\
\frac{1}{7} \left( e^{ij \frac{(x_2-a)x}{b-a}} - e^{ij \frac{(x_1-a)x}{b-a}} \right) & \text{otherwise.}
\end{cases} \quad (109)$$

The computation of the matrix vector multiplication of the Toeplitz matrix can be transformed to a circular convolution of the two vectors $m_s$ and $u_s$. These vectors have length $2N$ and the first $N$ elements are equal to the matrix vector product $M_s u$. The vectors are of the form stated below:

$$m_s = [m_0, m_{-1}, m_{-2}, \ldots, m_{1-N}, 0, m_{N-1}, m_{N-2}, \ldots, m_1]^T \quad (110)$$

$$u_s = [u_0, u_1, \ldots, u_{N-1}, 0, \ldots, 0]^T \quad (111)$$

The matrix vector multiplication is equal to the first $N$ elements of the circular convolution $m_s \circledast u_s$. For the circular convolution of the Hankel matrix we need to compute two different vectors which are equal to the matrix vector multiplication $M_c u$. These vectors are both of length $2N$ and the product of $M_c u$ is equal to the first $N$ terms in reversed order. The vectors are shown below:

$$m_c = [m_{2N-1}, m_{2N-2}, \ldots, m_1, m_0]^T \quad (112)$$

$$u_c = [0, 0, \ldots, 0, u_0, u_1, \ldots, u_{N-2}, u_{N-1}]^T \quad (113)$$

A circular convolution is equal to the inverse discrete Fourier transform of the product of the forward discrete Fourier transform which are both very fast functions.

$$x \circledast y = D^{-1} \{ D(x) \cdot D(y) \} \quad (114)$$

We now have a fast algorithm for the computation of $\hat{C}(x_1, x_2, t_m)$.

1. Determine $m_j(x_1, x_2)$ for $j = 0, 1, \ldots, N - 1$.
2. Compute the vectors $m_s(x_1, x_2)$ and $m_c(x_1, x_2)$ with the property that $m_{-j} = -m_j$.
3. Construct the vector $u(t_m)$.
4. Construct the vector $u_s$ by adding $N$ zeros under the vector $u(t_m)$.
5. Calculate $M_s u$ which are the first $N$ elements of $D^{-1} \{ D(m_s) \cdot D(u_s) \}$
6. Compute $M_c u$ which are the first $N$ elements of $D^{-1} \{ D(m_c) \cdot \text{sgn} \cdot D(u_s) \}$ in reversed order.
7. Finally compute $\hat{C}(x_1, x_2, t_m) = e^{-r\Delta t} \text{Im} \{ M_s u + M_c u \}$

## B Heston Model for Bermudan options

### B.1 Tolerance level

We will discuss the truncation range for the log-variance density. In previous sections we have given a standard equation for the truncation but this will not work for the log-variance process since this is
dependent of the center of the density function and the decay of the left and right tail. We will now work with Newton’s Method to determine proper boundaries for the log-variance process given a tolerance level, TOL. The Newton method will stop when \( p_{ln(v), x}(x|\sigma_t; T) < TOL \) for an \( x \in \mathbb{R} \setminus [a_v, b_v] \). Here the interval \([a_v, b_v]\) is the interval that we will use for the log-variance density domain. As usual for the Newton method we will need to give a proper initial guess. We estimate that the center of the truncation range should be the logarithm of the mean value of the variance:

\[
\ln(\mathbb{E}(v_t)) = \ln(v_0) e^{-\lambda T} + \hat{\beta}(1 - e^{-\lambda T})
\]

As noticed earlier the left tail decays much slower than the right tail so the left side of the truncation range should be greater. There is also a relation between the speed of decay and the value of \( q \) so as an initial guess we use that:

\[
[a_v^0, b_v^0] = \left[ \ln(\mathbb{E}(v_t)) - \frac{5}{1 + q}, \ln(\mathbb{E}(v_t)) + \frac{2}{1 + q} \right]
\]

### B.2 Log-variance process

There is a problem with the probability density when we do not meet the Feller condition. A solution for this problem is to transform the variance to a log-variance process. The probability density of the log variance then becomes:

\[
p_{\ln(v)}(\sigma_t|\sigma_s) = \zeta e^{-\zeta (e^{\sigma_s} e^{-\lambda(t-s)} + e^{\sigma_t})} \left( \frac{e^{\sigma_s}}{e^{\sigma_s} + e^{-\lambda(t-s)}} \right)^{\frac{q}{2}} e^{\sigma_t} I_q \left( 2\zeta e^{-\frac{q}{2} \lambda(t-s)} \sqrt{e^{\sigma_s} e^{\sigma_t}} \right)
\]

In the formula above we have that \( \sigma_s := \ln(v_s) \) and \( \sigma_t := \ln(v_t) \). The function denotes the probability density of the log-variance at time \( t \) given the log-variance at time \( s \). This log-variance probability density function makes the left tail of the regular variance process smooth so that the values are smaller and therefore the errors will become smaller.

We will now combine the log-variance density and the log-price that we used at the start of this section. The Heston model suggests that variance at time \( t \) is independent of the log-stock price at time \( s \). This gives us an expression for the joint distribution of the log-stock price and the variance process:

\[
p_{x, v}(x_t, v_t|x_s, v_s) = p_{x, v}(x_t|v_t, x_s, v_s) \cdot p_v(v_t|v_s)
\]

We will now transform this joint distribution to the log-variance process.

\[
p_{x, \ln(v)}(x_t, \sigma_t|x_s, \sigma_s) = p_{x, \ln(v)}(x_t|\sigma_t, x_s, \sigma_s) \cdot p_{\ln(v)}(\sigma_t|\sigma_s)
\]

Here \( p_{x, \ln(v)} \) is the probability function of the log-stock price at a future time \( t \), given the log-variance and the log-stock price at time \( s \). We have already given the function \( p_{\ln(v)}(\sigma_t|\sigma_s) \). The formula for \( p_{x, \ln(v)}(x_t|\sigma_t, x_s, \sigma_s) \) is not of a closed-form but we can derive its conditional characteristic function:

\[
\Phi(\omega; x, \sigma_t, \sigma_s, \sigma_s) := \mathbb{E}_s[\exp(i \omega x_t|\sigma_t)]
\]

\[
\exp \left( i \omega \left[ x_s + \mu(t-s) + \frac{\xi}{2} (e^{\sigma_t} - e^{\sigma_s} - \lambda \hat{\beta}(t-s)) \right] \right)
\]

\[
\Phi \left( \omega \left( \frac{\lambda \omega}{\eta} - \frac{1}{2} \right) + \frac{1}{2} \lambda \omega^2 (1 - \mu^2); e^{\sigma_t}, e^{\sigma_s} \right)
\]

In this equation we have the characteristic function for the Heston model which is defined as:
\[
\Phi(v; u, v) := \mathbb{E} [\exp (iv \int_0^t v_r d\tau | u, v)]
\]

\[
\begin{align*}
&\quad \frac{t_n}{\sqrt{\pi \eta_n} 4 \gamma (v) e^{-\frac{1}{2} \gamma (v) (t-s)}} \gamma (v) e^{-\frac{1}{2} \gamma (v) (t-s)} \eta_n (1 - e^{-\gamma (v) (t-s)}) \lambda (1 - e^{-\gamma (v) (t-s)}) \\
&= \exp \left( \frac{\nu_n + \nu_m}{\eta_n} \left[ \frac{\lambda (1 - e^{-\gamma (v) (t-s)})}{1 - e^{-\gamma (v) (t-s)}} - \frac{\gamma (v) (1 + e^{-\gamma (v) (t-s)})}{1 - e^{-\gamma (v) (t-s)}} \right] \right) \\
&\quad \gamma (v) := \sqrt{\lambda^2 - 2i \eta^2 v}
\end{align*}
\]

### B.3 COS method for Bermudan options

To simplify the notation we will change the name of the variables \( x_{m-1} \) and \( \sigma_{m-1} \) to \( x_m \) and \( \sigma_m \). The continuation value can be written as a double integral:

\[
c(x_m, \sigma_m, t_m) = e^{-r \Delta t} \int_{\mathbb{R}} \int_{\mathbb{R}} v(x_{m+1}, \sigma_{m+1}, l_{m+1}) p_{x, \ln(v)}(x_{m+1}, \sigma_{m+1}, x_m, \sigma_m) d\sigma_{m+1} dx_{m+1}
\]

From the formula for \( p_{x, \ln(v)} \) we can rewrite this integral to the following form:

\[
c(x_m, \sigma_m, t_m) = e^{-r \Delta t} \int_{\mathbb{R}} \int_{\mathbb{R}} v(x_{m+1}, \sigma_{m+1}, l_{m+1}) p_{x, \ln(v)}(x_{m+1}, \sigma_{m+1}, x_m, \sigma_m) dx_{m+1} \int_{\mathbb{R}} p_{\ln(v)}(\sigma_{m+1}, \sigma_m) d\sigma_{m+1}
\]

We can solve this integral numerically which we will do to get a good method for pricing Bermudan options. The inner integral is the same formula for pricing European options between time points: \( t_m \) and \( t_{m+1} \). The COS-method will be used to approximate the unknown function \( p_{x, \ln(v)} \). In the following part we will show how we recover this function. First we will define a truncation range that will be done in same way as for the normal Heston model shown in a previous section so:

\[
[a, b] := [c_1 - 12 \sqrt{|c_2|}, c_1 + 12 \sqrt{|c_2|}]
\]

We will now write the unknown function as his Fourier cosine expansion:

\[
p_{x, \ln(v)}(x_{m+1} | \sigma_{m+1}, x_m, \sigma_m) = \sum_{n=0}^{\infty} P_n(\sigma_{m+1}, x_m, \sigma_m) \cos \left( \frac{n \pi x_{m+1} - a}{b - a} \right)
\]

Here we have that \( P_n \) denote the Fourier cosine coefficients which are defined as followed:

\[
P_n(\sigma_{m+1}, x_m, \sigma_m) := \frac{2}{b - a} \int_a^b p_{x, \ln(v)}(x_{m+1} | \sigma_{m+1}, x_m, \sigma_m) \cos \left( \frac{n \pi x_{m+1} - a}{b - a} \right) dx_{m+1}
\]

We will now write these Fourier cosine coefficients in terms of the characteristic function as is usual to the COS-method. This gives us the following formula for \( P_n \):

\[
P_n(\sigma_{m+1}, x_m, \sigma_m) \approx \frac{2}{b - a} \Re \left\{ \phi \left( \frac{n \pi}{b - a}; x_m, \sigma_{m+1}, \sigma_m \right) e^{-i \pi \frac{x_{m+1}}{b-a}} \right\}
\]

Since we will calculate these coefficients numerically we will have to truncate to summation to \( N \) terms. This will only cause a very small error since the terms converge to zero very rapidly. We now have a semi-analytic formula for the function \( p_{x, \ln(v)} \):

\[
p_{x, \ln(v)}(x_{m+1} | \sigma_{m+1}, x_m, \sigma_m) \approx \sum_{n=0}^{\infty} \frac{2}{b - a} \Re \left\{ \phi \left( \frac{n \pi}{b - a}; 0, \sigma_{m+1}, \sigma_m \right) e^{-i \pi \frac{x_{m+1}}{b-a}} \right\} \cos \left( \frac{n \pi x_{m+1} - a}{b - a} \right)
\]
Here we have used the fact that \( \phi(\omega; x_m, \sigma_{m+1}, \sigma_m) = e^{i\omega x_m} \phi(\omega; 0, \sigma_{m+1}, \sigma_m) \). This means that the terms \( x_m \) can be separated from the \( \sigma \) terms as an exponential term.

We will now truncate the region of integration over which we integrate for \( c(x_m, \sigma_m, t_m) \). We have explained that for the log-variance we will determine a truncation range \( [a, b] \) using the Newton method. For the log-stock price we have give a standard interval that can be used namely \([a, b]\). The approximation of the continuation value now becomes:

\[
c_1(x_m, \sigma_m, t_m) := e^{-r\Delta t} \int_a^b \left[ \int_a^b v(x_{m+1}, \sigma_{m+1}, t_{m+1}) p_{x|\ln(v)}(x_{m+1}|\sigma_{m+1}, x_m) dx_{m+1} \right] p_{\ln(v)}(\sigma_{m+1}|\sigma_m) d\sigma_{m+1}
\]

We now have to discretise the outer integral over \( \sigma_{m+1} \) by interpolation-based quadrature rules which is appropriate since \( p_{\ln(v)} \) is known analytically. The method we will apply is an \( J \)-point quadrature integration rule called the Gauss-Legendre quadrature. We can apply this method since we have no singularities in our function and we can add weights to the quadrature. The new approximation becomes:

\[
c_2(x_m, \sigma_m, t_m) := e^{-r\Delta t} \sum_{j=0}^{J-1} a_j p_{\ln(v)}(\sigma_j|\sigma_m) \cdot \left[ v(x_{m+1}, \sigma_j, t_{m+1}) p_{x|\ln(v)}(x_{m+1}|\sigma_j, x_m) dx_{m+1} \right]
\]

Since we also do not know the exact formula for \( p_{x|\ln(v)} \) we will replace this function by its Fourier cosine expansion.

\[
c_3(x_m, \sigma_m, t_m) := e^{-r\Delta t} \sum_{j=0}^{J-1} a_j \sum_{n=0}^{N-1} V_{n,j}(t_{m+1}) \Re \left\{ \tilde{\varphi} \left( \frac{n\pi}{b-a}, \sigma_j, \sigma_m \right) e^{i\pi \frac{x_{m+1}}{x_m}} \right\}
\]

with:

\[
V_{n,j}(t_{m+1}) := \frac{2}{b-a} \int_a^b v(x_{m+1}, \sigma_j, t_{m+1}) \cos \left( \frac{n\pi x_{m+1}}{b-a} \right) dx_{m+1}
\]

and:

\[
\tilde{\varphi}(\omega, \sigma_{m+1}, \sigma_m) := p_{\ln(v)}(\sigma_{m+1}, \sigma_m) \cdot \varphi(\omega; 0, e^{\sigma_{m+1}}, e^{\sigma_m})
\]

This new function \( \tilde{\varphi} \) is the characteristic function for the Heston model. We now only have one Bessel function in this function so we have reduced the computational time. This Bessel function is of the following form:

\[
I_{\nu} \left( e^{\frac{1}{2}(\sigma_{m+1} + \sigma_m)} \cdot 2\kappa(\nu)e^{-\frac{1}{2}\gamma(\nu)\Delta t} \right)
\]

with:

\[
\nu = \omega \left( \frac{\lambda \rho}{\eta} - \frac{1}{2} \right) + \frac{1}{2} i \omega^2 (1 - \rho^2) \quad \text{and} \quad \kappa(\nu) = \frac{2\gamma(\nu)}{\eta^2 (1 - e^{-\gamma(\nu)\Delta t})}
\]

Finally we interchange the summation in \( c_3(x_m, \sigma_m, t_m) \) so that we have a discrete formula for the continuation value:

\[
c_3(x_m, \sigma_m, t_m) = e^{-r\Delta t} \Re \left\{ \sum_{n=0}^{N-1} \beta_n(\sigma_m, t_m) e^{i\pi \frac{x_{m+1}}{x_m}} \right\}
\]

with:

\[
\beta_n(\sigma_m, t_m) := \sum_{j=0}^{J-1} a_j V_{n,j}(t_{m+1}) \tilde{\varphi} \left( \frac{n\pi}{b-a}, \sigma_j, \sigma_m \right)
\]

The series coefficients of \( c_3(x_m, \sigma_m, t_m) \) are only dependent of the log-variance and not the log-stock price. We work with quadrature points on a log-variance grid for which we always use the same grid so we can change \( \sigma_m \) as a variable by the quadrature points \( \varphi_i \). This gives us the following approximation for the continuation value:

\[
c_3(x_m, \varphi_i, t_m) = e^{-r\Delta t} \Re \left\{ \sum_{n=0}^{N-1} \beta_n(\varphi_i, t_m) e^{i\pi \frac{x_{m+1}}{x_m}} \right\}
\]
with:

\[ \beta_n(\varsigma, t_m) := \sum_{j=0}^{J-1} w_j V_{n,j}(t_{m+1}) \tilde{\phi} \left( \frac{n\pi}{b-a}, \varsigma_j, \varsigma_p \right) \]  

The main advantage of changing the summation in \( c_3(x_m, \varsigma_p, t_m) \) is that we can calculate the value independent of the value of \( x_m \) so we can determine the early-exercise points very rapidly by determining when

\[ c_3(x_m, \varsigma_p, t_m) - g(x_m) = 0 \quad \text{for} \quad j = 0, 1, \ldots, J - 1 \]  

This value of \( x^*(\sigma_m, t_m) \) which are the early exercise points can be found by the following formulas:

- In point \( t_M \):
  \[ v(x_M, \sigma_M, t_M) = g(x_M) \] 
- In points \( t_m \) with \( m = 1, 2, \ldots, M - 1 \):
  \[ \hat{v}(x_t, \sigma_t, t_m) = \begin{cases} g(x_m) & \text{for} \; x \in [a, x^*(\sigma_m, t_m)] \\ c_3(x_m, \sigma_m, t_m) & \text{for} \; [x^*(\sigma_m, t_m), b] \end{cases} \]  

for a put option, and:

\[ \hat{v}(x_t, \sigma_t, t_m) = \begin{cases} c_3(x_m, \sigma_m, t_m) & \text{for} \; x \in [a, x^*(\sigma_m, t_m)] \\ g(x_m) & \text{for} \; [x^*(\sigma_m, t_m), b] \end{cases} \]  

for a call option.

- In point \( t_0 \):
  \[ \hat{v}(x_0, \sigma_0, t_0) = c_3(x_0, \sigma_0, t_0) \] 

Through this algorithm we can determine the value of the option at times \( t_0 \) and \( t_M \) but we can speed up the process by using a technique called backward recursion to recover the cosine series coefficients.

### B.4 Backward recursion

Using backward recursion we can reconstruct all \( \hat{v}(x_t, \sigma_t, t_m) \) for all time point \( m = 1, 2, \ldots, M - 1 \) by using the option value approximation at time \( t_M \). We have an analytic solution for the Fourier coefficients at time \( t_M \) because this value is independent of time so:

\[ V_{n,j}(t_M) = \begin{cases} G_n(0, b) & \text{for call options} \\ G_n(a, 0) & \text{for put options} \end{cases} \]  

Here \( G_n \) are the functions build by the cosine coefficients of the payoff function:

\[ g(y) = \max (\alpha K (e^y - 1), 0) \begin{cases} 1 & \text{for call options} \\ -1 & \text{for put options} \end{cases} \]  

and:

\[ G_n(l, u) := \frac{2}{b-a} \int_{l}^{u} g(y) \cos \left( \frac{n\pi y}{b-a} \right) dy, \]  

The analytic solution of this equation is:

\[ G_n(l, u) = \begin{cases} \frac{2}{b-a} \alpha K [\chi_k(l^*, u^*) - \psi_k(l^*, u^*]) & \text{for call options} \\ 1 & \text{for put options} \end{cases} \]  

with for \( \alpha = 1: \; l^* = \max(l, 0) \) and \( u^* = \max(u, 0) \). In case we have that \( \alpha = -1: \; l^* = \min(l, 0) \) and \( u^* = \min(u, 0) \). We still have the same definitions for \( \chi_k \) and \( \psi_k \).
Next we will calculate the value of the option at time \( t_{M-1} \). Since we know the value of \( V(t_M) \) we can calculate the values of \( \beta_n(s_p, t_{M-1}) \) for \( p = 0, 1, \ldots, J - 1 \). Inserting these values in the formula for \( c_3(x_m, s_p, t_m) \) we find the value of \( c_3(x_{M-1}, s_p, t_{M-1}) \). Again we use Newton’s method to find an early-exercise point for which \( c_3(x_{M-1}, s_p, t_{M-1}) - g(y) = 0 \). where \( y \equiv x^*(s_p, t_{M-1}) \). This gives us the early-exercise point and the definition of \( \hat{\varphi}(x_{M-1}, s_p, t_{M-1}) \). Therefore, we can split the integral of \( V_{n,j}(t_{m+1}) \) in two parts:

\[
\hat{V}_{k,p}(t_{M-1}) = \begin{cases} 
\hat{C}_{k,p}(x^*(s_p, t_{M-1}), b, t_{M-1}) + G_k(a, x^*(s_p, t_{M-1})) & \text{for call options} \\
\hat{C}_{k,p}(x^*(a, s_p, t_{M-1}), t_{M-1}) + G_k(x^*(s_p, t_{M-1}), b) & \text{for put options} 
\end{cases} 
\tag{148}
\]

Here we use the approximations \( \hat{V} \) and \( \hat{C} \) where \( \hat{C} \) represent the cosine coefficients of the continuation value:

\[
\hat{C}_{k,p}(l, u, t_{M-1}) := \frac{2}{b - a} \int_l^u c_3(y, s_p, t_{M-1}) \cos \left( k\pi \frac{y - a}{b - a} \right) dy 
\tag{149}
\]

Again we will replace \( c_3 \) in this equation by the COS approximation, then change summation and integration. This gives us a new formula for \( \hat{C}_{k,p} \):

\[
\hat{C}_{k,p}(l, u, t_{M-1}) = e^{-r\Delta t} \Re \left\{ \sum_{n=0}^{N-1} M_{k,n}(l, u) \hat{\beta}_n(s_p, t_{M-1}) \right\} 
\tag{150}
\]

with

\[
M_{k,n}(l, u) := \int_l^u \exp \left( i\pi y \frac{a}{b - a} \right) \cos \left( k\pi \frac{y - a}{b - a} \right) dy 
\tag{151}
\]

We can find an solution for this problem analytically. The next step is to transform this equation into an matrix vector product. We have a matrix \( M(l, u) \) which has the values \( M_{k,n} \) as elements and the matrix \( B(t_{M-1}) \) that contains the column vectors: \( \hat{\beta}_0(t_{M-1}), \hat{\beta}_1(t_{M-1}), \ldots, \hat{\beta}_{J-1}(t_{M-1}) \). This gives us the following equation:

\[
\hat{C}(l, u, t_{M-1}) = e^{-r\Delta t} \Re \{ M(l, u) B'(t_{M-1}) \} 
\tag{152}
\]

In the vector \( B(t_{M-1}) \) the column vectors are computed by:

\[
\beta_p(t_{M-1}) = [V(t_M) \cdot \hat{\varphi}(s_p)] \mathbf{w} 
\tag{153}
\]

In this equation we have the \( \mathbf{w} \) is the vector that contains all the quadrature weights and the matrix \( \hat{\varphi}(s_p) \) contains as elements: \( \hat{\varphi}(\frac{m\pi}{b-a}, s_p, s_p) \).

In the section before we have shown that this matrix \( M(l, u) \) can be split in a Hankel and a Toeplitz matrix so we can apply the fast Fourier transform algorithm. By applying this algorithm we can find the values of \( \hat{V}(t_m) \) for \( m = M - 1, M - 2, \ldots, 1 \) with the following formulas:

\[
\begin{align*}
\hat{V}(t_m) & := \begin{cases} 
\hat{C}(x^*(s_p, t_m), b, t_m) + G(a, x^*(s_p, t_m)) & \text{for a put} \\
\hat{C}(a, x^*(s_p, t_m), t_m) + G(x^*(s_p, t_m), b) & \text{for a call} 
\end{cases} \\
\hat{\beta}_j(t_{m-1}) & := [\hat{V}(t_m) \cdot \hat{\varphi}(s_j)] \mathbf{w} \\
\hat{B}(t_{m-1}) & := [\hat{\beta}_0(t_{m-1}), \hat{\beta}_1(t_{m-1}), \ldots, \hat{\beta}_{J-1}(t_{m-1})] \\
\hat{C}(l, u, t_{m-1}) & := e^{-r\Delta t} \Re \{ M(l, u) \hat{B}'(t_{m-1}) \}
\end{align*}
\tag{154}
\]
B.5 Algorithm

The algorithm used to describe the pricing of Bermudan options under the Heston model is given below.

Initialisation

- Find the truncation range for the log-variance $a_v$ and $b_v$ by applying Newton’s method.
- Next we calculate the value of $V(t_M)$ with the analytic formula that is given.
- Construct the matrix $\tilde{\varphi}(\varsigma_j)$ for $j = 0, 1, \ldots, J - 1$.

Loop In this loop we recover $\hat{V}(t_m)$ for $m = M - 1, M - 2, \ldots, 1$

- Determine the early-exercise point by applying Newton’s method.
- Calculate the first row and column of the matrices $M_s$ and $M_c$.
- We calculate the following vectors: $\hat{\beta}_j(t_m) = \left[ \hat{V}(t_m) \cdot \tilde{\varphi}(\varsigma_j) \right] w$ for $j = 0, 1, \ldots, J - 1$.
- Then we multiply the first element of $\hat{\beta}_j(t_m)$ by one-half.
- We compute the column vectors of $\hat{C}_{t_m}$ by the equation: $\exp{-r\Delta t} \text{Re} \left\{ M\hat{\beta}_j(t_{m-1}) \right\}$ using the fast Fourier transform algorithm.
- We can now calculate the values of $\hat{V}(t_m)$.

Finalisation

- Finally we calculate $\hat{v}(x, \varsigma_j, t_0)$. A spline interpolation can be used to determine $\hat{v}(x, \sigma_0, t_0)$.
C Matlab-code

In this appendix we will show the Matlab-codes that are used in our research.

C.1 Code for European options

```matlab
% This is the code I have written myself for pricing European options under
% the Heston model.
% S0 = current stock price
% K = Strike price
% T = time to maturity
% q: dividend rate
% r: risk-free interest rate
% N = number of COS terms
% sigma = volatility of the asset

clear all;
close all;

N=input('N = ');
K = input('K = ');
tic;

sigma = 0.25;
r = 0.1;
q = 0;
T = 0.1;
S0 = 100;

c1 = ((r - q) - sigma^2)*T;
c2 = sigma^2*T;

L = 10;
a = c1 - L * sqrt(c2);
b = c1 + L * sqrt(c2);
c = 0;
d = b;

x = log(S0/K);

d1 = (x + (r + 0.5 * sigma^2)*T)/(sigma * sqrt(T));
d2 = (x + (r - 0.5 * sigma^2)*T)/(sigma * sqrt(T));

Nd1 = normcdf(d1);
Nd2 = normcdf(d2);

v = S0 * Nd1 - K * exp(-r*T) * Nd2;

% for d = 4:1:8
% N = 2^d;
% chi = zeros(N,1);
% psi = zeros(N,1);
% V = zeros(N,1);
% G = zeros(N,1);
% for k = 1:N
%   o(k) = ((k-1)*pi)/(b-a);
```

\[ \chi(k) = \frac{1}{1+o(k)^2} \times \left( \cos(o(k) \cdot (d-a)) \times \exp(d) - \cos(o(k) \cdot (c-a)) \times \exp(c) + \ldots \right) \]
\[ \psi(k) = (\sin(o(k) \cdot (d-a)) - \sin(o(k) \cdot (c-a))) / o(k) \]
\[ \psi(1) = d - c; \]
\[ V(k) = \frac{2}{b-a} \times K \times (\chi(k) - \psi(k)); \]
\[ \phi(k) = \exp(1i \times o(k) \times (r - 0.5 \times \sigma^2) \times T - 0.5 \times o(k)^2 \times \sigma^2 \times T) \times \exp(1i \ldots \times o(k) \times (x-a)); \]
\[ G(k) = \text{real}(\phi(k)) \times V(k); \]
\[ \text{end} \]

C.2 Original code for Bermudan options

```matlab
function [C TotalTime InitTime LoopTime err] = BarrierHestonCosGaussianoud(S0, CP, K, T, q, r, Params, N, watchtime, J, L, TOL)
    tic
    i = complex(0,1);
    mu=r-q;
    dt=T/watchtime;
    if max(size(S0))==1
        x0=log(S0/K);
    else
        x0=0;
    end
    % truncation range of the density
    lambda = Params(1); eta = Params(2); vmean = Params(3); v0=Params(4); rho=Params(5); mean=mu+T*(1-exp(-lambda*T))*vmean-v0)/(2*lambda)-vmean*T/2;
    std=sqrt(vmean*(1+eta)*T);
    a = mean +x0- L*std;
    b = mean +x0+ L*std;
    end
```

% This function returns the price of a discrete double
% barrier option, with the lower barrier being "low_bar" and the upper barrier
% being "up_bar", and the number of monitoring dates being "watchtime"
% As the name suggests, discretization of the variance dimension is by
% Legendre-Gauss Quadrature rule.
% Inputs:
% S0: current stock price
% CP=1 call; CP=-1 put
% K: strike price
% T: time to maturity
% q: dividend rate
% r: risk-free interest rate
% Params: = [lambda eta vmean v0 rho];
% N: number of COS terms
% watchtime: number of monitoring dates
% J: number of quadrature points in variance dimension
% L: coefficient for truncation range in log-stock dimension
% TOL: truncation-error tolerance in variance dimension
% Fang Fang, Delft, April 2010.

% Initialization
mu=r-q;
dx=T/watchtime;
if max(size(S0))==1
    x0=log(S0/K);
else
    x0=0;
end
% truncation range of the density
lambda = Params(1); eta = Params(2); vmean = Params(3); v0=Params(4); rho=Params(5);
mean=mu+T*(1-exp(-lambda*T))*vmean-v0)/(2*lambda)-vmean*T/2;
std=sqrt(vmean*(1+eta)*T);
a = mean +x0- L*std;
b = mean +x0+ L*std;
% COS *grid*: frequencies of the cosine basis functions
n = (0:N−1).';% N by 1
co =pi/(b−a);% N by 1
omega = n*co;% N by 1

% vol−grid
v0=Params(4);eta=Params(2);
logv0=log(v0);
temp=exp(−lambda*T);
logvolmean= log(v0*temp + vmean*(1−temp));
q=2*vmean+lambda/eta^2−1;
logvolstd=sqrt(1/(q+1));

%initial guess
logVolRange=[logvolmean−5*logvolstd, logvolmean+2*logvolstd];

%Newton's method to find the variance range
c=2*lambda/((1−exp(−lambda*T))*eta^2);
u=c*v0*temp;
for jd=1:2
    y=logVolRange(jd);
    for id=1:50
        v=c*exp(y);
        temp0=sqrt(u.*v);
        plogv=c*exp(−u−v).*(v./u).^((1/2)*q).*temp0.*exp(y);
        if abs(plogv)<TOL
            break;
        end
        temp2 = besseli(q+1,2*temp0);
        dpdv=(v.*temp1−q*temp1−temp2.*v−temp1).*exp(−u−v+y).*(v./u).^((q/2)+c);
        y=y−plogv/dpdv;
    end
    logVolRange(jd)=y;
end

%define log−variance grid using Legendre−Gauss Quadrature rule
[logVolGrid, VolWeight]=lgwt(J,logVolRange(1),logVolRange(2));
VolGrid= exp(logVolGrid);
VolWeight=flipdim(VolWeight,1);

%phi:note that this function is defined between two adjacent time points.
phi=zeros(J, J, N); % vT v0 omega
c=2*lambda/((1−exp(−lambda*dt))*eta^2);
alpha=omega*(lambda*rho/eta−0.5)+0.5*i*omega.^2*(1−rho^2);
gamma=abs(lambdqrt(2+2*eta^2+i*alpha));
value1=(1/eta^2)*((−gamma.*(1+exp(−gamma*dt))./(1−exp(−gamma*dt))));
value2=4*gamma.*exp(−1/2*gamma*dt)./(eta^2*(1−exp(−gamma*dt)));
value3=c*(gamma.*exp(−1/2*(gamma−lambda)*dt)).*(1−exp(−lambda*dt).
<=...)/(lambda*(1−exp(−gamma*dt)));
for id2=1:J
    v0temp=VolGrid(id2);
    for id=1:J
        vTtemp=VolGrid(id);
        x=sqrt(v0temp*vTtemp)*value2;
        ccf=exp((v0temp+vTtemp)*value1 + abs(real(x))+ (v0temp−vTtemp)
            +lambda*eta^2+0.5*q*log(vTtemp+exp(lambdqrt(dt))/v0temp));
        ll=besseli(q,x,1);
        ccf=ccf.*value3.*ll.*exp(i*omega*(mu*dt+rho/eta*(vTtemp−v0temp−lambdqrt*vmean*dt))));
        phi(id2, id, :)=shiftdim(ccf*VolWeight(id),−2);
    end
end

low_bar=a;
up_bar=b;

%%%%%%%%%%%%%%%% Vkd(T) -----------------------------------------------
if CP==1
    c=max(0,low_bar); d=max(b,0);
```matlab
else
    c=min(low_bar,0); d=min(up_bar,0);
end
expd=exp(d); expc=exp(c);
expdma=exp(i*omega*(d-a)); expcma=exp(i*omega*(c-a));
cosdma=real(expdma); sindma=imag(expdma);
coscma=real(expcma); sincma=imag(expcma);
Xcoeff=1./(1+(omega).ˆ2);% N by 1
X = (cosdma*expd−coscma*expc + omega.* (sindma*expd
−sincma*expc) )...∗Xcoeff;% N by 1
Ycoeff=1./(co*n(2:end));
Y = (sindma − sincma);
Y(2:end) = Y(2:end)∗Ycoeff;
Y(1) = d−c;
Vkq = CP*K*(X−Y)*2/(b−a);% N by 1
Vkq = repmat(Vkq, [1 J]);% N by N

%============ Vj(tm) =============
eexprt=exp(−r*dt);
df=exprt/pi;
coeff=ones(2*N,1);coeff(2:2:end)=−1;
coeff=repmat(coeff, [1 J]);
buf = repmat([0;1./(1:(N−1))'], [1 J]);% N by 1
buf2= repmat(1./(N:2*N−2)', [1 J]);

%================Prepare Ms and Mc ==========
c=max(low_bar, a);
d=min(up_bar,b);
expdma =repmat(exp(i*omega*(d-a)),[1 J]);
expcma = repmat(exp(i*omega*(c-a)),[1 J]);
dma=(d-a)*co*ones(1, J);%N by 1
cma=(c-a)*co*ones(1, J);
mvec = (expdma − expcma).*buf;%N by J
mvec(1,:) = (d−c)*co*i;
mderive = −conj(mvec);% N by J
mvec = flipdim(mvec,1);% N by J
fftsm = fft([mderive; zeros(1,J);mvec(1:end−1,:)])
mdervem=( flipdim(merv,1);mvect);% N by N−1
mdervem=[mdervem; exp(i*(2*N−1)*dma) − exp(i*(2*N−1)*cma)/(2*N−1)];%N by J
fftm = fft((flipdim(merv,1);mvect));% J by N

%Main loop over the watching dates
for tim=watchtime−1:2
    for id=1:N
        u(id,:) =Vkq(id,:)*phi(:,:, id).';
    end
    us = zeros(2*N, J);
    us(1:N,:) = u;
    fftus = fft(us);
    Msu = ifft(fftus.* fftsm);% 2*N−1 by N
    Msu = Msu(1:N,:);% N by J

    %For Mc * u
    Mceu = ifft((fftsm.*coeff).*fftm);% 2*N by J
    Mcu = flipdim(Mceu(1:N,:),1);% N by J
    % Put together
    Vkq=df*imag(Mcu+Msu);
end
end
```

% Final Pricing Formula

for id=1:N
    u(id,:) = Vk(id,:) * phi(:,:, id).';
end

u(1,:) = 0.5 * u(1,:);

if max(size(S0))==1
    Ctemp = exprt * real(sum(u.*repmat(exp(i*omega*(x0−a)), [1 J]),1));
    C = interp1(logVolGrid, Ctemp', logv0, 'spline');
else
    C = zeros(size(S0));
    for id=1:length(S0)
        Ctemp = exprt * real(sum(u.*repmat(exp(i*omega*(log(S0(id)/K−a)), [1 J]),1)));
        C(id) = interp1(logVolGrid, Ctemp', logv0, 'spline');
    end
end

LoopTime = toc;

q=0;

% reference value
ref = Euro_Cos(20, S0, CP, K, T, q, r, 'Heston', Params, 1024);

err = C−ref;

TotalTime = LoopTime + InitTime;

return;

function [x,w]=lgwt(N,a,b)

% lgwt.m
%
% This script is for computing definite integrals using Legendre-Gauss
% Quadrature. Computes the Legendre-Gauss nodes and weights on an interval
% [a,b] with truncation order N
%
% Suppose you have a continuous function f(x) which is defined on [a,b]
% which you can evaluate at any x in [a,b]. Simply evaluate it at all of
% the values contained in the x vector to obtain a vector f. Then compute
% the definite integral using sum(f.*w);

% Written by Greg von Winckel - 02/25/2004
N=N−1;
N1=N+1; N2=N+2;
xu=linspace(−1,1,N1)';

% Initial guess
y=cos((2*(0:N)'+1)*pi/(2*N+2))+(0.27/N1)*sin(pi*xu*N/N2);

% Legendre-Gauss Vandermonde Matrix
L=zeros(N1,N2);

% Derivative of LGVM
Lp=zeros(N1,N2);

% Compute the zeros of the N+1 Legendre Polynomial
% using the recursion relation and the Newton-Raphson method
y0=10;
y=(2*(0:N)+1)*pi/(2*N+2)+(0.27/N1)*sin(pi*xu*N/N2);

% Iterate until new points are uniformly within epsilon of old points
while max(abs(y−y0))>eps
    L(:,1)=1;
Lp(:,1)=0;
L(:,2)=y;
Lp(:,2)=1;
for k=2:N1
L(:,k+1)=( (2*k-1)*y.*L(:,k)-(k-1)*L(:,k-1) )/k;
end
Lp=(N2)*( L(:,N1)-y.*L(:,N2) )./(1-y.^2);
y0=y;
y=y0-L(:,N2)./Lp;

% Linear map from [-1,1] to [a,b]
x=(a*(1-y)+b*(1+y))/2;
% Compute the weights
w=(b-a)./(1-y.^2).*Lp.^2*(N2/N1).^2;

% For European option pricing and for many strikes
% Here K is a vector
function C = Euro_Cos(L, S0, CP, K, T, q, r, Model, Params, M)
%tic;
%change K vetor to a column vector if it is not
N=length(K);
sizeK = size(K);
if sizeK(1)==1
K=K';
end

%Initialise timestep
dt=T;mu=r-q;
qstar=L*gridvol_model(dt,Model,Params);
x0=log(S0./K);% vector
b=qstar;
a=-qstar;
c=meanvar_model(mu, dt, Model, Params);
mean=c(1);
var=c(2);
std=sqrt(var);
a=mean-L*std;
b=mean+L*std;
a=-100;
b=20;

%——— For the coefficients of density ———
j = [0:M-1];% 1 by M
co =pi/(b-a);
omega = j*co;
cf = cf_model(omega, mu, dt, Model, Params);% 1 by M
cf(1)=0.5*cf(1);
%——— For the coefficients of Payoff ———
if CP==1
% c=0;d=b;
dma = pi;
cma= (-a)*co;
expd=exp(d);
X = (cos(j*dma)*expd - cos(j*cma) + j*co.* ( - sin(j*cma) ) ) ./((j*co).^2);
Y = ( - sin(j*(2:end)*cma)) ./(co*j(2:end));
function cf = cf.model(u, mu, tau, Model, Params)

if strcmpi(Model, 'BS') == 1;
    sigma = Params(1);
    i = complex(0,1);
    cf = exp(i * u * ((mu - 1/2 * sigma^2) * tau) - 1/2 * sigma^2 * u.^2 * tau);
end

if strcmpi(Model, 'CGMY') == 1;
    sigma = Params(1);
    C = Params(2);
    G = Params(3);
    M = Params(4);
    Y = Params(5);
    gamma_neg_Y = gamma(-Y);
    M_Y = M^Y;
    G_Y = G^Y;
    phi_CGMY = C * gamma_neg_Y * ((M-i+u).^Y - M_Y + (G+ i+u).*Y - G_Y);
    mu_tilde = -sigma^2/2 - C + gamma_neg_Y * ((M-1).^Y - M_Y + (G+1).*Y - G_Y);
    cf = i*(mu + mu_tilde*u - sigma^2/2 - u.^2 + phi_CGMY);
    cf = exp(tau*cf);
end

if nargin > 1
    dcf=zeros(size(cf));
end
return;
end

if strcmpi(Model, 'Heston');
    lambda = Params(1); eta = Params(2); vmean = Params(3); v0 = Params(4); rho = ...
    Params(5);
    i = complex(0,1);
    numtau = i*u+mu+tau;
    D = sqrt((lambda-rho*eta*i+u).^2 + (u.^2+1+i+u+eta^2));
end
alpha=(\lambda & \rho*\eta*i*u-D);
G=alpha./(\lambda & \rho*\eta*i*u+D);
beta=1-exp(-D*tau);
gamma=(1-G.*exp(-D*tau));
etasq=eta^2;

cf=exp(i*umutau+v0/etasz+(beta./gamma)*alpha+lambda*vmean/etasq... 
(tau+alpha-2*log(gamma.)/(1-G)));

C.3 Enhanced Code for Bermudan options

% As the name suggests, discretization of the variance dimension is by
% Legendre-Gauss Quadrature rule.

%Inputs:
% S0: current stock price
% CP=1 call; CP=-1 put
% K: strike price
% T: time to maturity
% q: dividend rate
% r: risk-free interest rate
% Params: =\lambda eta vmean v0 rho;
% N: number of COS terms
% watchtime: number of monitoring dates
% J: number of quadrature points in variance dimension
% L: coefficient for truncation range in log-stock dimension
% TOL: truncation error tolerance in variance dimension

function [C TotalTime InitTime LoopTime err] = BarrierHestonCosGaussiannieuw(S0, CP, ...
K, T, q, r,...
Params,N, watchtime,J,L,TOL)
tic
i=complex(0,1);

%Initialization
mu=r-q;
dt=T/watchtime;
if max(size(S0))==1
x0=log(S0/K);
else
x0=0;
end

% truncation range of the density
lambda = Params(1); eta = Params(2); vmean = Params(3); v0=Params(4); rho=Params(5);
mean=mu*T+(1-exp(-lambda*T))*(vmean-v0)/(2*lambda)-vmean*T/2;
std=sqrt(vmean*(1+eta)*T);
a = mean +x0-L*std;
b = mean +x0+ L*std;

% COS "grid": frequencies of the cosine basis functions
n = (0:N-1)'; N by 1
co =pi/(b-a);
omega = n*co N by 1

% vol-grid
v0=Params(4);eta=Params(2);
logv0=log(v0);
temp=exp(-lambda*T);
logvolmean= log(v0*temp + vmean*(1-temp));
\( q = 2 \cdot v_{\text{mean}} \cdot \lambda / \eta^2 - 1; \)

\( \logV0\text{std} = \sqrt{1 / (q+1)}; \)

**Initial guess**

\( \logV0\text{Range} = [\logV0\text{mean} - 5 \cdot \logV0\text{std}, \logV0\text{mean} + 2 \cdot \logV0\text{std}]; \)

**Newton's method to find the variance range**

\( c = 2 \cdot \lambda / [(1 - \exp(-\lambda \cdot T)) \cdot \eta^2]; \)

\( u = c \cdot v0 \cdot \text{temp}; \)

\( \gamma_{q1} = \gamma(q+1); \)

\( \gamma_{q2} = \gamma(q+2); \)

**for** \( jd = 1:2 \)

\( y = \logV0\text{Range}(jd); \)

**for** \( id = 1:50 \)

\( v = c \cdot \text{exp}(y); \)

\( \text{temp} = \sqrt{u \cdot v}; \)

\( \text{plogv} = c \cdot \text{exp}(-u - v). \cdot (v / u) \cdot \text{temp} \cdot \text{exp}(y); \)

**if** \( \text{abs}(\text{plogv}) < \text{TOL} \)

**break;**

**end**

\( \text{temp}2 = \text{Inuzpower}(q+1, \gamma_{q2}, 2 \cdot \text{temp0}, 10^{-16}); \)

\( \text{dpdv} = -(v \cdot \text{temp1} - q \cdot \text{temp1} - \text{temp2} \cdot v - \text{temp1}) \cdot \text{exp}(-u - v + y) \cdot ((v / u) \cdot (q/2)) \cdot c; \)

\( y = y - \text{plogv}/\text{dpdv}; \)

**end**

\( \logV0\text{Range}(jd) = y; \)

**end**

%define log−variance grid using Legendre−Gauss Quadrature rule

\( [\logV0\text{Grid}, \text{VolWeight}] = \text{lgt}(J, \logV0\text{Range}(1), \logV0\text{Range}(2)); \)

\( \text{VolWeight} = \text{flipdim}(\text{VolWeight}, 1); \)

\( \text{VolGrid} = \text{exp}(\logV0\text{Grid}); \)

\( \text{VolWeight} = \text{VolWeight} \cdot \text{VolGrid}; \)

\%phi:note that this function is defined between two adjacent time points.

\( \phi = \text{zeros}(J, J, N); \) % \( \text{vT v0 omega} \)

\( c = 2 \cdot \lambda / ([1 - \exp(-\lambda \cdot dt)] \cdot \eta^2); \)

\( \alpha = \omega \cdot (\lambda \cdot \rho / \eta - 0.5) + 0.5 \cdot i \cdot \omega^2 \cdot (1 - \rho^2); \)

\( \gamma_{1} = \sqrt{\lambda^2 - 2 \cdot \eta^2 \cdot i \cdot \alpha}; \)

\( \text{value1} = (1 / \eta^2) \cdot \left( -\gamma_{1} \cdot (1 + \exp(-\gamma_{1} \cdot dt)) / (1 - \exp(-\gamma_{1} \cdot dt)) \right); \)

\( \text{value2} = 4 \cdot \gamma_{1} \cdot \exp(-1/2 \cdot \gamma_{1} \cdot dt) / (\eta^2 \cdot (1 - \exp(-\gamma_{1} \cdot dt))); \)

\( \text{value3} = c \cdot (\gamma_{1} \cdot \exp(-1/2 \cdot (\gamma_{1} - \lambda) \cdot dt) \cdot ... \)

\( \cdot (1 - \exp(-\lambda \cdot dt))) / (\lambda \cdot (1 - \exp(-\gamma_{1} \cdot dt))); \)

\( \text{value4} = \text{value3} \cdot \exp(i \cdot \omega \cdot \mu \cdot dt); \)

**for** \( id2 = 1:J \)

\( v0\text{temp} = \text{VolGrid}(id2); \)

**for** \( idz = 1:1:J \)

\( v\text{temp} = \text{VolWeight}(idz); \)

\( x = \sqrt{v0\text{temp} \cdot v\text{temp}} \cdot \text{value2}; \)

\( \text{ccf} = \exp((v0\text{temp} + v\text{temp}) \cdot \text{value1} + \text{abs}(\text{real}(x))) + (v0\text{temp} - v\text{temp}) \cdot ... \)

\( \lambda \cdot (1 - \exp(-\lambda \cdot dt)); \)

\( \maxx = \max(x); \)

**if** \( \maxx < 1 \)

\( I2 = \text{Inuzpower}(q, \gamma_{q1}, x, 10^{-10}); \)

\( I1 = \exp(-\text{abs}(\text{real}(x))) \cdot I2; \)

**else** \( \maxx \geq 18 \)

\( I1 = \exp(-\text{abs}(\text{real}(x))) \cdot \text{Inuasymp}(q, x, 10^{-10}); \)

**end**

% For the interval in between we use the regular Bessel function
else
  I1 = besseli(q, x, 1);
end
ccf = ccf.*I1.*value4.*exp(i*omega/eta*(vTtemp-v0temp-lambda*vmean*dt));
ccftemp = ccf*vWtemp;
ccf=ccf.*I1.*value4.*exp(i*omega*rho/eta*(vTtemp−v0temp−lambda*vmean*dt));
ccftemp = ccf*vWtemp;
phi(id2, idz, :)=shiftdim(ccftemp, -2);
end
end

% low
bar = log(LowBarrier/K);
% up
bar = log(UpBarrier/K);
low
bar=a;
up
bar=b;

%========= Vkq(T) ================================================
if CP==1
c=max(0, low
bar); d=max(b, 0);
else
c=min(low
bar, 0); d=min(up
bar, 0);
end
expd=exp(d); expc=exp(c);
expdma=exp(i*omega*(d−a)); expcma=exp(i*omega*(c−a));
cosdma=real(expdma); sindma=imag(expdma);
coscma=real(expcma); sincma=imag(expcma);
Xcoeff=1./(1+(omega).ˆ2);% N by 1
X = (cosdma*expd−coscma*expc + omega.* (sindma*expd − sincma*expc ))...% N by 1
.∗Xcoeff;% N by 1
Ycoeff=1./(co*n(2:end));
Y = (sindma − sincma);
Y(2:end) = Y(2:end).*Ycoeff;
Y(1) = d−c;
Vkq = CP*K*(X−Y)*2/(b−a);% N by 1
Vkq = repmat(Vkq, [1 J]);% N by N

%============ Vj(tm) =============
exprt=exp(−r*dt);
df=exprt/pi;
coeff=ones(2*N,1); coeff(2:2:end)=−1;
coeff=repmat(coeff, [1 J]);
buf = repmat([0;1./(1:(N−1))'], [1 J]);
buf2 = repmat(1./(N:2*N−2)', [1 J]);

%================Prepare Ms and Mc =========
c=max(low
bar, a);
d=min(up
bar, b);
expdma = repmat(exp(i*omega*(d−a)),[1 J]);
expcma = repmat(exp(i*omega*(c−a)),[1 J]);
dma=(d−a)*co*ones(1, J);%N by 1
cma=(c−a)*co*ones(1, J);
mvec = (expdma − expcma).*buf;%N by J
mvec(l,:)=(d−c).*co;i;
mdervative = −conj(mvec);% N by J
mvec = flipdim(mvec,1);% N by J
fftms = fft([mdervative; zeros(1,J);mvec(1:end−1,:)]);
mdervative = ( repmat(exp(i*(N−1)*dма), [N 1]).*expdma ...)
  − repmat(exp(i*(N−1)*cма), [N 1]).*expcma).*[ones(1, J); buf2];%N by J,w + v0
mdervative=mdervative(2:end, :);% N−1 by N
mdervative=[mdervative; exp(i*(2*N−1)*dма) − exp(i*(2*N−1)*cма)/(2*N−1)];%N by J
fftmc = fft([flipdim(mdervative,1);mvec]);%w + v0
u=zeros(N,J);
InitTime=toc;
tic;
tic;
% Main loop over the watching dates
for tim=watchtime:-1:2
    for id=1:N
        u(id,:) = Vkq(id,:)*phi(:, :, id).';
    end
    u(1,:) = 0.5*u(1,:);
    us = zeros(2*N, J);
    us(1:N,:) = u;
    fftus = fft(us);
    Msu = ifft(fftus.*fftms);  % 2*N-1 by N
    Msu = Msu(1:N,:);  % N by J

    % For Mc * u
    Mcu = ifft((fftus.*coeff).*fftmc);  % 2*N by J
    Mcu = flipdim(Mcu(1:N,:),1);  % N by J

    % Put together
    Vkq = df*imag(Mcu+Msu);
end  %for loop over time lattice

%====== Final Pricing Formula ===========
for id=1:N
    u(id,:) = Vkq(id,:)*phi(:, :, id).';
end
    u(1,:) = 0.5*u(1,:);

if max(size(S0))==1
    Ctemp = exprt*real(sum(u.*repmat(exp(i*omega*(x0-a)), [1 J]),1));
    C = interpl(logVolGrid, Ctemp', logv0, 'spline');
else
    C = zeros(size(S0));
    for id=1:length(S0)
        Ctemp = exprt*real(sum(u.*repmat(exp(i*omega*(log(S0(id)/K)-a)), [1 J]),1));
        C(id) = interpl(logVolGrid, Ctemp', logv0, 'spline');
    end
end

LoopTime = toc;
q=0;
%Reference value
ref = Euro.Cos(20, S0, CP, K, T, q, r, 'Heston', Params, 1024);
err = C-ref;
TotalTime = LoopTime + InitTime;
return;

% This function returns the Bessel approximation according to the first
% Bessel approximation.
function s = Inuzpower(nu, gammag, z, eps)
if nu == -1
    s = Inuzpower(1, z, eps);
elseif nu == -1/2
    s = sqrt(2*z/pi) * cosh(z)/z;
else
    w = rdivide(times(z, z), 4);
    t = rdivide(w, (nu+1));
    s = 1 + t;
    k = 1;
    e = 1;
    while e > eps
        s = s + rdivide(k, e);
        e = e + 1;
        k = rdivide(k, e);
    end
end
\[ t = \text{rdivide}(\text{times}(t,w), ((k+1) \times (nu + k + 1))); \]
\[ s = s + t; \]
\[ \text{if } s < 0 \]
\[ s = -s; \]
\[ \text{end} \]
\[ f = \text{rdivide}(t, (1+s)); \]
\[ \text{if } f < 0 \]
\[ e = -f; \]
\[ \text{else} \]
\[ e = f; \]
\[ \text{end} \]
\[ k = k + 1; \]
\[ \text{end} \]
\[ x = z/2; \]
\[ y = \text{power}(x, nu); \]
\[ s = \text{times}(y, (\text{rdivide}(s, \text{gammaq}))); \]
\[ \text{end} \]

% This function returns the Bessel function according to the second
% approximation.

function s = Inuzasymp(nu, z, eps)

format long;

% Initialize variables
\[ t = \text{ones}(\text{length}(z),1); \]
\[ \text{ser1} = \text{ones}(\text{length}(z),1); \]
\[ \text{ser2} = \text{ones}(\text{length}(z),1); \]
\[ s = \text{ones}(\text{length}(z),1) \times -1; \]
\[ \text{nu2} = \text{ones}(\text{length}(z),1) \times \text{nu}^2; \]
\[ k = \text{zeros}(\text{length}(z),1); \]
\[ \text{e} = 1; \]
\[ \text{kmax} = (2 + \text{fix}(\text{abs}(z \times 2))); \]

\[ \text{while } (\text{e} > \text{eps}) \& (\text{kmax} > k) \]
\[ t = t \times (\text{nu2} - (k + 0.5) \times 2) / (2 \times z \times (k + 1)); \]
\[ \text{ser1} = \text{ser1} + s \times t; \]
\[ \text{ser2} = \text{ser2} + t; \]
\[ e = \text{abs}(t \times \text{ser1}) + \text{abs}(t \times \text{ser2}); \]
\[ s = -s; \]
\[ k = k + 1; \]
\[ \text{end} \]
\[ \text{e} = \text{exp}(z); \]
\[ s = e \times (\text{ser1} + \text{li} \times \text{exp}(z - 2 \times \text{nu} \times \text{pi} \times \text{li}) \times \text{ser2}) / \text{sqrt}(2 \times z \times \text{pi}); \]
\[ \text{return}; \]
\[ \text{end} \]