Pricing Turbo Warrants

Jonatan Eriksson*,
Dept. of Mathematics, Uppsala University, jonatan@math.uu.se,

Jonas Persson*,
Dept. of Information Technology, Uppsala University, jonasp@it.uu.se.

* Supported by FMB, the Swedish Graduate School in Mathematics and Computing.

March 23, 2006

Abstract

We numerically price the financial contracts named turbo warrant that were released early in 2005. They have been studied mathematically in [3] where explicit pricing formulas for the Geometric Brownian motion were derived. For more general underlying stochastic processes we have no analytical formulas and numerical methods are necessary. In this work two different methods are compared, stochastic pricing using a Monte Carlo method and a deterministic PDE approach using finite differences. The methods are evaluated in terms of numerical efficiency, computation time and accuracy. In the numerical experiments the geometric Brownian motion has been used as underlying stochastic process. Our results show that for low accuracy the methods are almost equal in efficiency but for higher accuracy the finite difference method is much more efficient.

Keywords: Turbo warrant, Finite difference method, Monte Carlo method, Option pricing

AMS subject classification: 65M06, 65C05, 91B24, 91B28

1 Introduction

The name turbo warrant first appeared in Germany at the end of 2001 as the name of a warrant having a knock-out level at the strike price. This type of instrument is however nothing else than a knock-out barrier option. A more interesting situation appears when the barrier is strictly in the money and a rebate is paid if the barrier is hit. Such contracts were introduced at the end of February 2005 by Société Generale (SG). For turbo call warrants (also called

long turbos) the rebate is the difference between the lowest recorded stock-price during a three hour period after the barrier is hit and the strike price, and for turbo put warrants (also called short turbos) the rebate is calculated as the difference between the strike price and the largest recorded stock-price during a three-hour period after the first hitting time. Other financial institutes offer similar products with the same name and different rebates.

Closed form expressions for the price has been derived in [3] where the underlying process considered is geometric Brownian motion (GBM). If the underlying process is considered to be something else than GBM we have no analytical expression for the price and finding the price numerically is necessary. We have found no other references pricing these new contracts. Regular barrier options though have been studied and discussed in many papers and books including [2, 4, 9, 10, 11].

In this paper we compare the accuracy and efficiency of two methods for numerically price turbo warrants. A Monte Carlo (MC) method is used in the stochastic setting. Formulating the problem in a PDE setting allows for the use of standard numerical methods such as a finite difference (FD) or finite element method. Here we consider only the FD-method. Studying the case with GBM allows for comparison of the numerical results with exact expressions which considerably simplifies the evaluation of the methods.

Section 2 gives an introduction to turbo warrants and how they can be priced using stochastic methods. In Section 3 we derive pricing equations for turbo warrants. In Section 4 we present the FD method used to price the warrants and in Section 5 we describe how we can implement a MC algorithm to simulate the expected value determining the price. In the final sections 6 and 7 we present our numerical experiments and the results obtained and give some concluding remarks.

2 Turbo warrant theory

Throughout the paper S_t will denote the stock-price process at time t. We assume that the process follows an Itô-diffusion which is absorbed at zero and which satisfies a stochastic differential equation (SDE) of the form $dS_t = rS_t dt + S_t \sigma(S_t, t) dW_t$. Here r denotes the risk-free rate of interest, σ is the local volatility surface and W is a standard Brownian motion. For a background to Itô-calculus and stochastic differential equations we refer the reader to [1] and [6]. We will use solutions to SDE to model turbo warrant prices. The turbo warrants that we consider come in two types, turbo-puts and turbo-calls, and the pay-offs are defined in the following way.

- A turbo call warrant pays $(S_T - K)^+$ at maturity T if a prescribed barrier $b \geq K$ has not been hit by S_t at any time $t < T$. If S_t hits the barrier the contract is immediately terminated and a new contract starts. The new contract is a call option on the minimum process of S_t, with the same strike-price, maturing δ time-units later.

- A turbo put warrant pays $(K - S_T)^+$ at maturity T if a prescribed barrier $b \leq K$ has not been hit by S_t at any time $t < T$. If S_t hits the barrier the contract is immediately terminated and a new contract starts. The new contract is a put option on the maximum process of S_t, with the same strike-price, maturing δ time-units later.

Other banks offer turbo warrants with different rebates. E.g. the difference between the the strike price and the mean value of the stock process at closing time during a few days after the barrier is hit.

To price the warrant we consider the instrument as a sum of two instruments. A knock-out warrant maturing at time T and a knock-in warrant maturing δ time-units after knock-in. Let us first consider the turbo call warrant. By risk-neutral valuation the value of the knock-out part at time t, given that $S_t = s$ and that the warrant is still alive, is given by

$$F(s,t) = \mathbf{E}_{s,t} e^{-r(T-t)} (S_T - K)^+ \mathbf{1}_{\{\tau_b > T\}},$$

where $\tau_b = \inf\{t \geq 0 : S_t \leq b\}$ is the stopping time at which the stock-price hits the barrier for the first time. Here $\mathbf{E}_{s,t}$ denotes the expected value given that $S(t) = s$ and $\mathbf{1}_{\{a>b\}}$ is the indicator function, which is 1 if $a > b$ and 0 otherwise. The value at time t of the rebate, given that $S_t = s$, and that the warrant is still alive, is given by the following expression

$$G(s,t) = \mathbf{E}_{s,t} e^{-r(\tau_b + \delta - t)} (m_{\tau_b + \delta} - K)^+ \mathbf{1}_{\{\tau_b \leq T\}},$$

where $m_t = \min_{0 \leq u \leq t} S_u$ is the running minimum of the process S_t. The price for the turbo call warrant given that it is still alive and that $S_t = s$ is then given by $V(s,t) = F(s,t) + G(s,t)$. In the put-case everything is similar with $(S_T - K)^+$ and $(m_{\tau_b+\delta} - K)^+$ replaced by $(K - S_T)^+$ and $(K - M_{\tau_b+\delta})^+$ where $M_t = \max_{0 \leq u \leq t} S_u$ is the running maximum of the process S_t. To find the price numerically one can either use MC methods and evaluate the above expectations directly, as we do in Section 5, or one can use e.g. FD methods and solve the pricing equations satisfied by F and G. This is explained in more detail in Section 4. In the special case of σ being constant the price is explicitly computed using the strong Markov property and time-homogeneity of the underlying process S_t. Explicit calculations can be found in [3]. To be able to measure the accuracy of our numerical methods and to compare with the exact value, we do all our numerical experiments using constant volatility. This is no loss of generality since our numerical methods, both MC and FD, are easily adapted to more general stochastic processes.

3 Pricing turbo warrants with differential equations

To price a turbo warrant using PDE-methods one must solve several equations. The price is composed of two main parts. The knock-out part, with value at

time t given by

$$F(s,t) = \mathbf{E}_{s,t} e^{-r(T-t)} \phi(S_T) \mathbf{1}_{\{\tau_b > T\}},$$

and the rebate part, with the value

$$G(s,t) = \mathbf{E}_{s,t} e^{-r(\tau_b+\delta-t)} \phi(Y_{\tau_b+\delta}) \mathbf{1}_{\{\tau_b \leq T\}}.$$

Here $\phi(s) = (s-K)^+$ and $Y_t = \min_{0\leq u\leq t} S_t$ if it is a call warrant, and $\phi(s) = (K-s)^+$ and $Y_t = \max_{0\leq u\leq t} S_t$ if it is a put warrant. Thus for call warrants, the contract is alive on $s > b$, and for put warrants it is alive on $s < b$. Let us begin with the call warrant. All warrant prices satisfy different versions of the Black-Scholes equation and therefore we denote by $\mathcal{M}$ the Black-Scholes operator, i.e.

$$\mathcal{M}\psi = \frac{1}{2} s^2 \sigma^2(s,t) \psi_{ss} + rs\psi_s - r\psi + \psi_t, \tag{1}$$

where ψ_s denotes differentiation with respect to the spatial variable s corresponding to the stock-price and ψ_t denotes differentiation with respect to time.

3.1 A PDE for turbo warrants of call-type

In this case we need to solve one Black-Scholes equation for the knock-out part, one for the knock-in part *before* it has knocked in and one for the knock-in part *after* it is knocked in. The first equation is straight forward to derive, it is just the ordinary Black-Scholes equation with an additional boundary-condition $F(b,t) = 0$ for all $t \in [0,T]$. The knock-in parts are not so straight forward to derive. Here we describe in short how one can reason:

Assume that at the present time t the underlying asset has not yet hit the barrier. Notice first that by iterating the conditional expectation we can rewrite the expression $\mathbf{E}_{s,t} e^{-r(\tau_b+\delta-\tau_b)} (m_{\tau_b+\delta} - K)^+ \mathbf{1}_{\{\tau_b \leq T\}}$ as

$$\mathbf{E}_{s,t} e^{-r(\tau_b - t)} \left(\mathbf{E}_{b,\tau_b} e^{-r\delta} (m_{\tau_b+\delta} - K)^+ \right) \mathbf{1}_{\{\tau_b \leq T\}}.$$

Thus to compute the price we need to know the value of the rebate at the barrier, i.e. at the time the rebate is activated. This is given by the expectation $g(b,\tau_b) = \mathbf{E}_{b,\tau_b} e^{-r\delta} (m_{\tau_b+\delta} - K)^+$. Given the function g, the value of the rebate *before* it is activated is then given by $G(s,t) = \mathbf{E}_{s,t} e^{-r(\tau_b - t)} g(b,\tau_b) \mathbf{1}_{\{\tau_b \leq T\}}$. By the assumption that at time t the warrant is still alive, it follows by application of the Itô-formula (see e.g. [1]) that $G(s,t)$ is given by the solution to Black-Scholes equation with boundary value $g(b,t)$ at $s = b$ and final value 0. The equation for the price is given in Proposition 1 below.

Proposition 1 *Assume that at time t the warrant is still alive (i.e. the barrier is not yet hit). Then the value at time t of the rebate, given that $S_t = s$, is given by $G(s,t)$ where G is the solution to the following Black-Scholes equation:*

$$\begin{cases} \mathcal{M}G(s,t) = 0 \text{ on } (b,\infty) \times [0,T), \\ G(b,t) = g(b,t) \text{ for } t \in [0,T], \\ G(s,T) = 0 \text{ for } s > b. \end{cases} \tag{2}$$

What remains is to determine the function g. We do as follows: Let $u \in [0,T]$ and denote by $\Omega_u = \{(s,m) \in (0,\infty)\times(0,\infty) : s > m, K \leq m < b\} \times [u, u+\delta)$. Let $H^u(s,m,t)$ be the solution to the following equation:

$$\begin{cases} \mathcal{M}H^u(s,m,t) = 0 \text{ on } \Omega_u \\ H^u(s,K,t) = 0 \text{ for } t \in [u,u+\delta), \\ H^u(s,m,u+\delta) = (m-K)^+ \text{ for } s \geq m, K \leq m \leq b, \\ \frac{\partial H^u}{\partial m}(m,m,t) = 0 \text{ for } t \in [u,u+\delta), K \leq m < b. \end{cases} \tag{3}$$

The following result holds.

Proposition 2 *The boundary values $g(b,t)$ are given by $g(b,t) = H^t(b,b,t)$ where $H^u(s,m,t)$ solves Equation (3).*

The intuitive explanation for the expression for the boundary value is that when the underlying hits the barrier for the first time, that is at time τ_b, the rebate which itself is a contract in terms of the minimum-process starts. The value function solves a Black-Scholes equation but only from τ_b and until the contract matures, that is δ time-units into the future. We only need to know what the value is at τ_b, hence the random variable $H^{\tau_b}(b,b,\tau_b)$.

Proof First, notice that the minimum process has zero quadratic variation. That is any $(dm_t)^2$ in the Itô-formula vanishes. Second, notice that m_t is constant except on a set of times which has Lebesgue-measure zero. The differential dm_t is only non-zero when $m_t = S_t$, i.e. when S_t realizes the current minima. By applying the Itô formula to $H^u(S_t,m_t,t)$ and integrating from $t=u$ to $t = u+\delta$ we obtain

$$\begin{aligned} & e^{-r(u+\delta)}H^u(S_{u+\delta},m_{u+\delta},u+\delta) = e^{-ru}H^u(S_u,m_u,u) + \\ + & \int_u^{u+\delta} e^{-rt}\mathcal{M}H^u(S_t,m_t,t)dt + \int_u^{u+\delta} e^{-rt}\frac{\partial H^u(S_t,m_t,t)}{\partial m}dm_t + \\ + & \int_u^{u+\delta} e^{-rt}\frac{\partial H^u(S_t,m_t,t)}{\partial s}S_t\sigma(S_t,t)dW_t. \end{aligned}$$

Taking expectation of both sides, using that H solves equation (3) and conditioning on $S_u = s, m_u = m$, yields

$$\begin{aligned} e^{-ru}H^u(s,m,t) &= \mathbf{E}_{s,m,u}e^{-r(u+\delta)}H^u(S_{u+\delta},m_{u+\delta},u+\delta) \\ &= \mathbf{E}_{s,m,u}e^{-r(u+\delta)}(m_{u+\delta}-K)^+. \end{aligned}$$

Hence the stochastic representation formula

$$H^u(s,m,u) = \mathbf{E}_{s,m,u}e^{-r\delta}(m_{u+\delta}-K)^+$$

for H is obtained. It follows that $H^{\tau_b}(b,b,\tau_b) = \mathbf{E}_{b,\tau_b}e^{-r\delta}(m_{\tau_b+\delta}-K)^+ = g(b,\tau_b)$ and hence that the boundary values are given by $g(b,t) = H^t(b,b,t)$.

3.2 A PDE for turbo warrants of put-type

Like in the call case we need to solve several Black-Scholes differential equations. One equation for the knock-out part, which is just the ordinary Black-Scholes equation on $(0,b)\times[0,T)$ with the additional boundary condition zero at b corresponding to knock-out. When it comes to the knock-in part we need to first determine the boundary values $g(b,t)$ which are given by a PDE similar to Equation (3), and then solve for the value of the knock-in part before knock in which amounts to solve an equation similar to (2).

The rebate can be rewritten as

$$\mathbf{E}_{s,t}e^{-r(\tau_b-t)}\left(\mathbf{E}_{b,\tau_b}e^{-r\delta}(K-M_{\tau_b+\delta})^+\right)\mathbf{1}_{\{\tau_b\le T\}}.$$

Let $g(b,\tau_b)=\mathbf{E}_{b,\tau_b}e^{-r\delta}(K-M_{\tau_b+\delta})^+$. Then $G(s,t)=\mathbf{E}_{s,t}e^{-r(\tau_b-t)}g(b,\tau_b)\mathbf{1}_{\{\tau_b\le T\}}$ gives the value of the rebate before activation. Since in the put case the warrant is alive on $(0,b)$ the equation satisfied by G is the Black-Scholes equation on the domain $(0,b)\times[0,T)$ with final value zero and boundary value $g(b,t)$ at $s=b$ and zero at $s=0$. More precisely

Proposition 3 *Assume that at time t the warrant is still alive (i.e. the barrier is not yet hit). Then the time t-value of the rebate, given that $S_t=s$, is given by $G(s,t)$ where G is the solution to the following Black-Scholes equation:*

$$\begin{cases} \mathcal{M}G(s,t)=0 \text{ on } (0,b)\times[0,T),\\ G(b,t)=g(b,t) \text{ for } t\in[0,T],\\ G(0,t)=0 \text{ for } t\in[0,T],\\ G(s,T)=0 \text{ for } 0<s<b. \end{cases} \tag{4}$$

The boundary values $g(s,t)$ can be found by solving a partial differential equation. Take $u\in[0,T]$ and let $\Omega_u=\{(s,M)\in(0,\infty)\times(0,\infty): s\le M, b<M\le K\}\times[u,u+\delta)$ and let $H^u(s,M,t)$ be the solution to the following equation:

$$\begin{cases} \mathcal{M}H^u(s,M,t)=0 \text{ on } \Omega_u\\ H^u(s,K,t)=0 \text{ for } t\in[u,u+\delta),\\ H^u(s,M,u+\delta)=(K-M)^+ \text{ for } s\le M, b<M\le K,\\ \frac{\partial H^u}{\partial M}(M,M,t)=0 \text{ for } t\in[u,u+\delta), b<M\le K. \end{cases} \tag{5}$$

The following proposition holds.

Proposition 4 *The boundary values $g(b,t)$ are given by $g(b,t)=H^t(b,b,t)$ where $H^u(s,m,t)$ solves Equation (5).*

To prove this proposition one argues like in the call case and uses the Itô-formula.

4 The Finite Difference method

The price of the turbo warrant is computed numerically by solving several Black-Scholes PDEs (see Sec. 3.1 and 3.2). We make a semi-discretization with finite

differences in space and then use an ODE solver in time. This is sometimes referred to as the Method of lines. For references on finite differences in the context of option pricing see e.g. [7] , [8], [9] and [10]. For the time-discretization we use the implicit Backward differentiation formula of order 2 (denoted BDF2) [5]. The FD method has been implmented in Matlab.

4.1 The space discretization

The derivatives in the PDE are all in s and t, see Equation (1), and are approximated on the space-grid using centered second order finite differences. In the m-direction we also discretize space, but since the dependency on m is only at the boundary $s = m$ we can in practise divide the space into m-levels and solve a one-dimensional PDE on each level. The result is a semi-discrete problem

$$\frac{\partial H(t)}{\partial t} = AH(t) + f(t)$$

where A is the finite difference matrix and $f(t)$ comes from boundary conditions. Note though that A and $f(t)$ are different on each m-level and that H really depends on s, m and t but we get a clearer presentation by suppressing the variables not needed. The boundary conditions and the way we have implemented them will be described in more detail later. The matrix A has a tridiagonal structure and is thus very sparse.

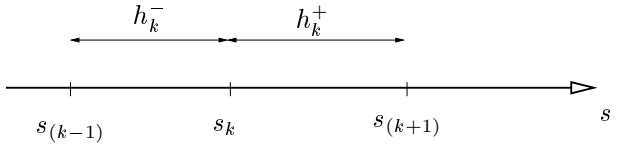

Figure 1: The s-axis. Here s_k, $k = 1 \dots N$ denotes the k:th node of the space discretization of s.

In Figure 1 we show the grid spacing for a non equidistant grid. The finite differences on such a grid will have different weights for different grid-points. These weights are calculated according to equation (7) and (9) below.

$$\frac{\partial H(s_k)}{\partial s} = H_s(s_k) \approx a_{s_k} H(s_{(k+1)}) + b_{s_k} H(s_k) + c_{s_k} H(s_{(k-1)}), \tag{6}$$

where

$$a_{s_k} = \frac{h_k^-}{h_k^+(h_k^- + h_k^+)}, \quad b_{s_k} = \frac{h_k^+ - h_k^-}{h_k^- h_k^+}, \quad c_{s_k} = -\frac{h_k^+}{h_k^-(h_k^- + h_k^+)}, \tag{7}$$

and the corresponding coefficients for the second derivatives are

$$\frac{\partial^2 H(s_k)}{\partial s^2} = H_{s^2}(s_k) \approx a_{s_k s_k} H(s_{(k+1)}) + b_{s_k s_k} H(s_k) + + c_{s_k s_k} H(s_{(k-1)}), \tag{8}$$

where

$$a_{s_k s_k} = \frac{2}{h_k^+(h_k^- + h_k^+)}, \quad b_{s_k s_k} = -\frac{2}{h_k^- h_k^+}, \quad c_{s_k s_k} = \frac{2}{h_k^-(h_k^- + h_k^+)}. \tag{9}$$

If the volatility is level dependent but not time dependent these coefficients can be computed once for each PDE problem. Using Matlabs sparse format only non-zero element will be stored.

4.2 Time discretization

The time derivative is approximated using the implicit multi-step method BDF2 (see e.g. [5] for more details). The BDF2 algorithm using evenly spaced grid-points in time is

$$\frac{\partial H(t)}{\partial t} \approx (\frac{3}{2}H^{n+1} - 2H^n + \frac{1}{2}H^{n-1})/\Delta t.$$

Since BDF2 is a multi-step method we have used the one-step first order accurate method Euler backwards for the first time-step.

The implicit nature of BDF2 requires the solution of a tri-diagonal linear system of equations for each time-step. An explicit method would of course not require the solution of linear systems but on the other hand explicit methods have problem dependent limitations in the size of the time-step that can be used. BDF2 is unconditionally stable which allows us to use large time-steps without loosing stability. Small time-steps should of course be used for more accurate solutions.

If we denote the vector of unknowns at time-level n and m-level m by u_m^n the resulting system of equations on m-level m will be

$$(\frac{3}{2}I - \Delta t A)u_m^{n+1} = 2u_m^n - \frac{1}{2}u_m^{n-1} + f_m^{n+1}\Delta t$$

which can be simplified to

$$Bu_m^{n+1} = c$$

since the solutions u_m^n and u_m^{n-1} are considered to be known when solving for time-level $n+1$. Here $B = \frac{3}{2}I - \Delta t A$ and $c = 2u_m^n - \frac{1}{2}u_m^{n-1} + f_m^{n+1}\Delta t$. The linear system of equations can be solved using any appropriate direct or iterative solver. Since we use Matlab we have used the built-in \-operator.

4.3 The computational domain and boundary conditions

4.3.1 Turbo warrant call

The area where the PDE must be solved is described in Section 3 (here we study the call option only). When solving the problem numerically one must truncate this area since we cannot solve numerically for all $s \geq m$ and all $m \geq K$ (for the put option this is not needed). The computational domain is

therefore defined as a truncation of the original domain and numerical boundary conditions must be imposed. The computational domain is depicted in Figure (2). In our computations we have set s_{max} to 8 times s_{min} where s_{min} for each m-level is on the line $s = m$. At the far-field boundary (i.e. at s_{max})

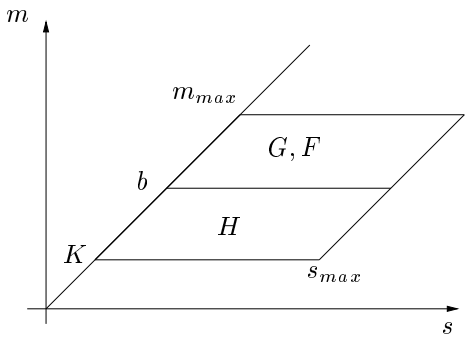

Figure 2: Computational domain for the turbo warrant call.

we use a numerical linearity boundary condition stating that $\partial H^2/\partial^2 s = 0$. This boundary condition is described in [9] where it is mentioned as one of two boundary conditions often used in financial problems when the price can be regarded as linear in s for large s-values. In this case, final data is typically a constant function and for most parameter values the linearity assumption works very well as a boundary condition at the far field boundary. If the time to maturity is very long we have noticed that this boundary condition can influence the error in the numerical solution. However, increasing s_{max} somewhat has proven to remedy this problem.

At the boundary $s = m$ we have to discretize the boundary condition in Equation (3), i.e. that $\partial H/\partial m = 0$ should hold there. Note that the variable m does not appear in the PDE, only in the terminal condition and in the boundary condition at the left boundary. One should also note that numerically this is the only connection between the PDEs in the m-direction. This allows us to solve the PDE at one m-level at the time starting from $m = K$ where the solution is known and work upwards.

We use extrapolation from previous m-levels to approximate the boundary condition across $s = m$. At $m = K + \Delta m$ we set the boundary value to zero. This approximates the derivative being zero. At the next m-level we again want to approximate the derivative at the boundary by extrapolating from the previous m-level. However, in general the value of s where we need the function value on the previous m-level might not be in the grid. If this is the case we interpolate in the s-direction to find the value needed for the extrapolation. For all the following steps we use a second order approximation using values from the two previous m-levels. The idea is described in Figure 3 where α and $\hat{\alpha}$ are the s-values needed in the extrapolations.

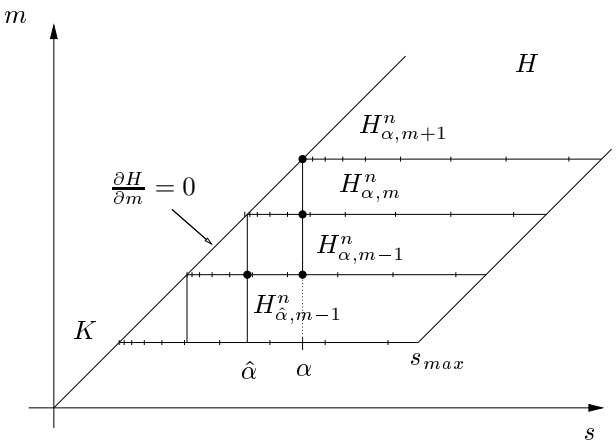

Figure 3: The boundary condition at the next m-level is extrapolated from the previous m-levels where values between grid-points are interpolated.

The second order approximation used is:

$$H^n_{\alpha,m+1} = \frac{4}{3}H^n_{\alpha,m-1} - \frac{1}{3}H^n_{\alpha,m}.$$

Note that normally when approximating a derivative like this a factor Δm would be involved but that disappears since we approximate a derivative equal to zero. Notice that the final condition in the interior of the domain and the numerically computed Dirichlet boundary condition at the $s = m$ boundary will in most cases not coincide. However, this is not introduced only in the numerical approximations, also the continuous problem has a discontinuitu of this type at maturity.

In Figure 4 we show a close-up of the result of a numerical computation of the three different parts making up the price of a turbo warrant call. The sum of the three parts then add to the warrant value. Note that this is only the values at $t = 0$ and where $s = m$ and that when solving the PDE problem we compute values for many different s-values at each m-level.

4.3.2 Turbo put warrant

For the turbo put the computational domain is different from the turbo call. In section 3.2 the mathematical domain is described. When pricing the call warrant we had to truncate the computational domain, this is not necessary here since there are natural restrictions on s and M, where M denotes the maximum process.

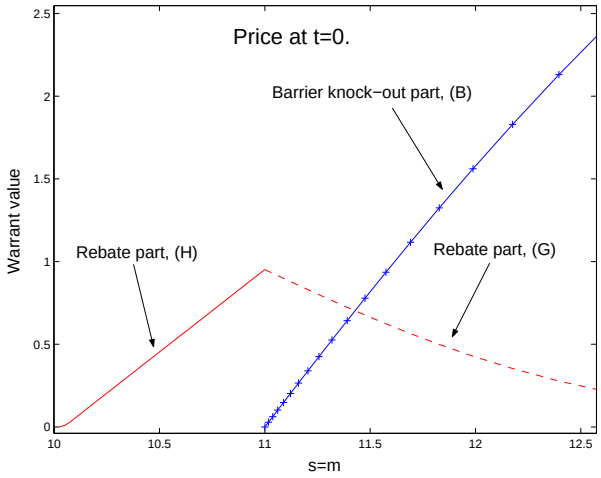

Figure 4: Close-up of the price at the barrier for a turbo call warrant. The price is the sum of the three parts.

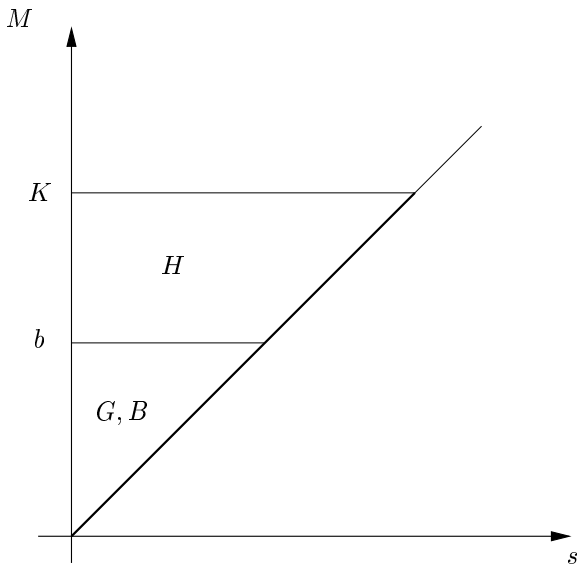

Figure 5: Computational domain for the turbo put warrant.

At the boundary $s = M$ we have the same boundary condition as in the call-case and numerically this means that we can use the same approximation in a similar way. At the $s = 0$ boundary we have used the Dirichlet boundary condition $H(0, M, t) = G(0, M, t) = 0$. As in the call case we first solve the

H-problem, here from $m = K$ down to $m = b$, and then the G and barrier problem.

4.4 Using a non-equidistant grid in space

The efficiency of finite difference methods can often be improved substantially by placing grid-points adaptively. This has been investigated in e.g. [7] and [8], where space-time adaptive FD-methods for multi dimensional European options are developed. These methods though, were developed with several dimensions in mind and the goal was to use few grid-points in each dimension to efficiently use a small number of grid-points. However, all adaptive methods come with an extra cost and to use an adaptive method in this case (to cleverly place the grid-points in space) would probably not reduce the total computational time needed to solve the problem.

Instead we choose a middle way with a simple improvement on the standard equidistant finite difference grid. The grid-points have been placed between $s = m$ and s_{max} (note that s_{max} is different on each m-level) using a logarithmic distribution so that we automatically get more grid-points close to the boundary $s = m$. This is because we know that the solution changes the most over space and time close to the boundary.

5 Pricing turbo warrants with Monte Carlo methods

Monte Carlo methods for option pricing is a straight forward and rather intuitive method. It is used to price many different contracts, the simplest perhaps the path-independent European option. Pricing the path-dependent turbo warrant, though, is a bit more complicated. Just as the FD method we have implemented the MC method in Matlab.

The value at time t of the turbo call warrant, given that $S_t = s$, is

$$V(s,t) = \mathbf{E}_{s,t} e^{-r(T-t)} \phi(S_T) \mathbf{1}_{\{\tau_b > T\}} + \mathbf{E}_{s,t} e^{-r(\tau_b + \delta - t)} \phi(Y_{\tau_b + \delta}) \mathbf{1}_{\{\tau_b \leq T\}}.$$

Both of these expectations depend on the whole path of the underlying asset and to simulate the price by using Monte Carlo simulations it is in general necessary to simulate the whole trajectory and at each step in time check whether the barrier is crossed or not.

Let us, like in Section 3, denote by $F(s,t)$ the first of the above expectations and by $G(s,t)$ the second expectation. To approximate

$$F(s,t) = \mathbf{E}_{s,t} e^{-r(T-t)} \phi(S_T) \mathbf{1}_{\{\tau_b > T\}}$$

we simulate N trajectories starting at s at time t and keep only the n trajectories which did not cross the barrier at any step in time. Of all simulated S_T-values

let $\mathcal{Z}$ be the subset of paths which have not crossed the barrier. Then the approximate value of F is given by

$$F_{appr}(s,t) = e^{-r(T-t)}\frac{1}{N}\sum_{S\in\mathcal{Z}}\phi(S).$$

The second term, i.e. $G(s,t) = \mathbf{E}_{s,t}e^{-r(\tau_b+\delta-t)}\phi(Y_{\tau_b+\delta})\mathbf{1}_{\{\tau_b\leq T\}}$, is a bit more complicated to simulate. Here we need to simulate both the first hitting time τ_b and the path of the minimum or maximum process. The functions ϕ and Y are defined in Section 3. For each simulated path of the process S_t we keep track of the current minima or maxima and if the barrier has been crossed or not. If the barrier is crossed at some time before maturity T we save that time and keep simulating the process during the rebate period δ. To obtain good accuracy in this method it is crucial that the number of time-steps in the interval $[\tau_b, \tau_b+\delta]$ is sufficiently large, as well as the number of time-steps on the whole time interval $[0,T]$. As a technical note we should mention that since we have used geometric Brownian motion we use the continuity correction factor due to Broadie and Glasserman presented in [2].

When implementing this method it is not necessary to save all simulated trajectories but only the current minima (maxima) and S_t. If a simulated path crosses the barrier we keep simulating that path for another δ time-units. Between τ_b and $\tau_b+\delta$ we record the minimum (maximum) of the process. For each simulated path we then calculate an approximate value for $e^{-r(\tau_b+\delta-t)}\phi(Y_{\tau_b+\delta})\mathbf{1}_{\{\tau_b\leq T\}}$ and by averaging over all simulated paths the approximate value of $G(s,t)$ is calculated in the usual way. The number of simulated trajectories will decrease as paths hit the barrier.

We have used the same number of time-steps on the shorter time interval $[\tau_b, \tau_b+\delta]$ as on the longer interval $[0,\ T]$. This works well but is probably not an optimal choice. How to choose the relation between the time-steps could certainly be investigated more thoroughly.

We choose a tolerance level $\epsilon > 0$ for the error (including both statistical and temporal error) and allow for $\epsilon/2$ to come from the statistical part. Then we simulate N paths and calculate the approximate value for the price and the sampled standard variance

$$s_N^2 = \frac{1}{N-1}\left(\sum_{S\in\mathcal{Z}}(V_a-\phi(S))^2 + \sum_{S\notin\mathcal{Z}}(V_a-\phi(m))^2\right),$$

where V_a is the current approximation of the price and m is the minimum value at $\tau_b+\delta$. The standard error is then calculated as $s_N/\sqrt{N}$. Note that the left sum includes all trajectories not hitting the barrier and the right sum includes all trajectories that hit the barrier.

We then check if

$$1.96 s_N/\sqrt{N} \leq \epsilon/2 \quad \text{(implying that } \mathbb{P}(|V_a - V| \leq \epsilon/2) > 0.95).$$

If the inequality is not satisfied we do the same computations for another N paths and update the price and the standard error and check again. We keep

doing this until the inequality is satisfied.

To estimate the error due to the time discretization we calculate the price for two different time steps, one being four times the other, to obtain $V_a(\Delta t)$ and $V_a(\Delta t/4)$. By assuming that the temporal discretization error τ does not depend on the number of trajectories used we obtain $V_a(\Delta t) = V_a + \epsilon_0 + \tau$ and $V_a(\Delta t/4) = V_a + \epsilon_1 + \tau/2$ where ϵ_0 and ϵ_1 are the errors due to the number of trajectories and we use that the time discretization is of order 1/2. For a large number of trajectories these errors are, according to the central limit theorem, approximately normally distributed with mean zero and sample variance s_N^2/N (assuming that they are equally distributed). The difference $\Delta V_a = V_a(\Delta t) - V_a(\Delta t/4)$ can be observed from the simulations and the difference $\epsilon_1 - \epsilon_0$ is asymptotically normally distributed with sample variance $2s_N^2/N$. By subtraction we obtain

$$\tau/2 = \epsilon_1 - \epsilon_0 + \Delta V_a \in \mathcal{N}(\Delta V_a, 2s_N^2/N), \text{ (asymptotically)}.$$

Now we can calculate $\Delta V_a + 1.96\sqrt{2}s_N/\sqrt{N}$ and take smaller time-steps until this quantity is smaller than ϵ in modulus. In this way we know with 95% certainty that the total absolute error is smaller than ϵ.

6 Numerical experiments

The numerical efficiency of the MC method and the FD method has been studied and compared. One should bear in mind when comparing these methods that they differ significantly in one sense. The FD method computes, and has to compute, the price of the warrant for several possible values of the underlying asset. This is not necessary to do using the MC method, where only the price of the warrant for the current stock price of today has to be computed.

This of course goes both ways, if we actually need several prices for different values of the underlying asset, then the FD method will have an advantage. If, however, only one price is needed then the MC method has an advantage. Normally one needs several prices for hedging reasons but how and in what ways these contracts are hedged lies outside the scope of this article.

We have implemented both methods also for the turbo warrant put and verified that they produce correct results. In the numerical experiments though, we have studied the turbo warrant call only. This is because we believe that studying also the put would give little extra information about the performance of the two methods.

The parameters used in all tests have been the following. The risk-free rate of return r was set to 1.5% and the volatility was fixed to 25%. The time to maturity was 0.2 years and the rebate period $\delta = 4 * 10^{-4}$ years (i.e. about 3.5 hours). The strike-price was $K = 10$, the barrier at $b = 11$ and the stock-price at $t = 0$ was $S_0 = 12$.

All computations were performed on an Intel Xeon with 2×3.2GHz processors and 3Gb memory. Only one processor has been used though, we have not performed parallel computations.

6.1 Numerical efficiency

For the MC method we have used the method described in Section 5 to terminate the computations when we have reached a certain predefined level of accuracy ϵ. This will give us a confidence interval for the absolute error in the price. The computational time needed to achieve this error level is then compared to the time needed by the FD method to get an absolute error of ϵ for the same value of the underlying asset. In Figure 6 we see that the FD method is up to 50 to 60 times faster than the MC method when it comes to achieve a certain accuracy. However, it is also obvious that for lower accuracy the two methods are almost equal in efficiency. Figure 6 also show reference slopes of what can be expected of the FD and the MC method respectively. For the FD method we have used second order discretizations in space and time and the work scales linearly so we should expect the overall error to be of order 2. In the MC method we have used an Euler scheme in time which is of order $1/2$ in the strong sense. The statistical error is also of order $1/2$ and also in this case the work scales linearly so that the computational time can be expected to scale as $1/2$ compared to the error.

For both methods there are of course a number of parameters important for the numerical result. In the FD method the most important parameters are Δt, Δs and Δm. Of these it has turned out that very few Δt are needed to get a good result. For the MC method the number of simulations N and the number of time-steps are of greatest interest. In Figure 7 we see the absolute error at the point $S_0 = 12$ for both methods. And this time the absolute error is plotted for the MC method and not the statistical error plotted in Figure 6. To the left we see the errors using the FD method with different combinations of space-steps and m-steps. The number of time-steps was fixed at 25. For each line we have 40 to 200 m-steps and a fixed number of space-steps which is then varied from line to line. For the MC method we have, for each line, 1 to 9 million realizations in the middle and 10 to 100 million realizations to the right. Along the lines we then vary the number of time-steps from 10 to 320.

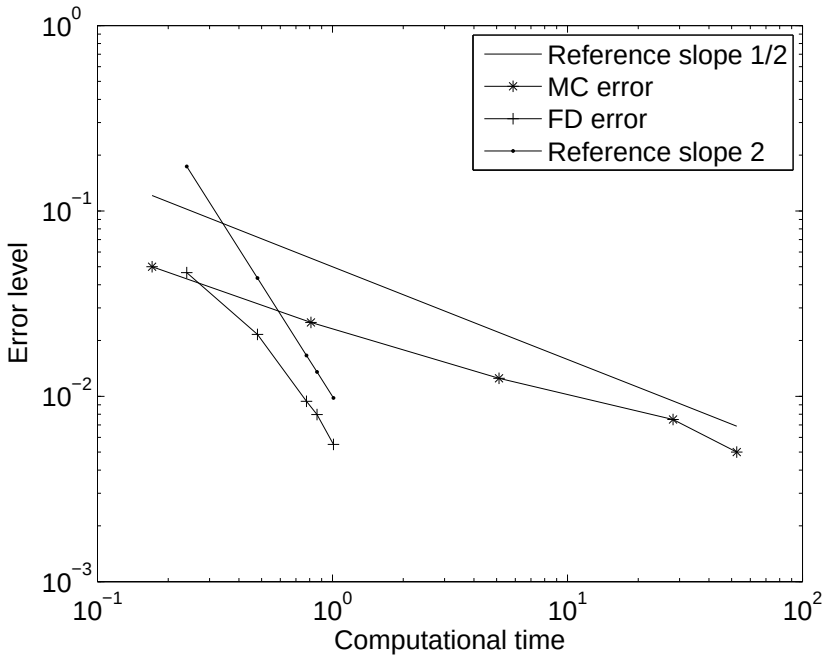

Figure 6: Efficiency comparison plotted in loglog-plot, Monte Carlo method and Finite Difference method.

7 Conclusions

In this paper we price numerically the turbo warrants offered by the French bank Société Generale. The turbo warrant is a type of barrier option paying a rebate when hitting the barrier. For both call and put turbo warrants, we have implemented both a Monte Carlo and a finite difference method. Here we have only studied the turbo call though, since studying the put also would give little extra information to our analysis.

Our results show that the finite difference method is up to 60 times faster than the Monte Carlo method depending on required accuracy. The Monte Carlo method is typically more efficient for lower accuracy. However, the Monte Carlo method is easier to implement since it is directly applicable to the stochastic formulation of the problem. The finite difference solver requires the problem to be formulated in the somewhat more involved PDE setting.

Worth noticing here, as always when comparing these methods, is that the Monte Carlo method calculates the warrant price for one specific value of the underlying stock while the finite difference method calculates the price for several prices of the underlying stock. This can be important when e.g. hedging these contracts.

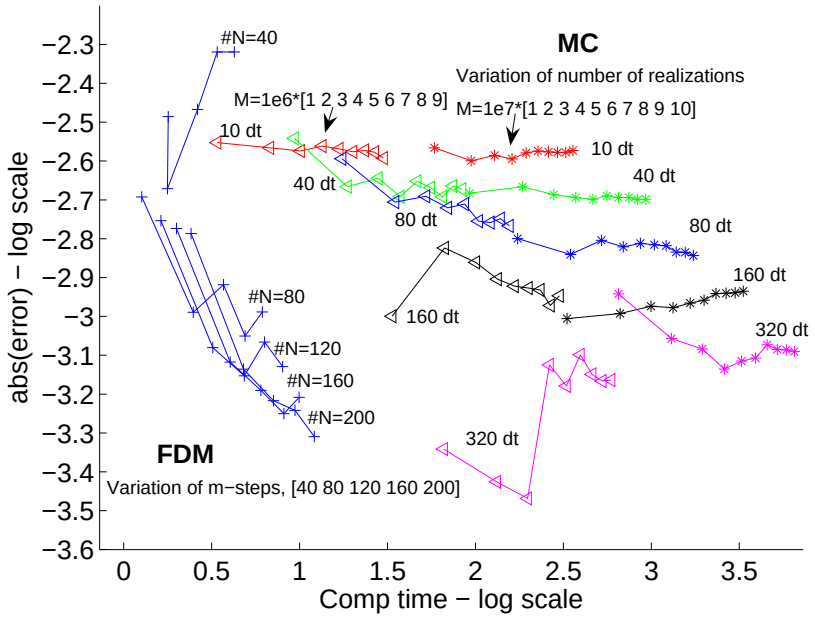

Figure 7: Absolute error for the MC and the FD method. FDM: N is number of grid-points in space, denoted by '+'. MC: M =1 to 9 million realizations denoted by '$\triangleleft$' and M =10 to 50 millions by '$*$'.

References

[1] T. Björk. *Arbitrage Theory in Continuous Time*. Oxford University Press, New-York, 1998.

[2] M. Broadie, P. Glasserman, and S. Kou. A Continuity Correction for Discrete Barrier Options. *Math. Finance*, 7:325–348, 1997.

[3] J. Eriksson. Explicit Pricing Formulas for Turbo Warrants. Accepted in RISK magazine.

[4] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer Verlag, 2004.

[5] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations, Nonstiff problems, 2nd ed.* Springer-Verlag, Berlin, 1993.

[6] B. Øksendal. *Stochastic Differential Equations - An Introduction with Applications*. Springer Verlag, 2003.

[7] J. Persson and L. von Sydow. Pricing European multi-asset options using a space-time adaptive FD-method. Technical Report 2003-

059, Dept. of Information Technology, Uppsala University, Available at http://www.it.uu.se/research/reports/, 2003. Accepted in Computing and Visualization in Science.

[8] J. Persson, L. von Sydow, P. Lötstedt, and J. Tysk. Space-Time Adaptive Finite Difference Method for European Multi-Asset Options. Technical Report 2004-055, Dept. of Information Technology, Uppsala University, Available at http://www.it.uu.se/research/reports/, 2004.

[9] D. Tavella and C. Randall. *Pricing Financial Instruments - The Finite Difference Method*. John Wiley & Sonc, Inc., 2000.

[10] P. Wilmott, J. Dewynne, and S. Howison. *Option Pricing: Mathematical models and computation*. Oxford Financial Press, 1993.

[11] R. Zwan, K.R. Vetzal, and P.A. Forsyth. PDE methods for pricing barrier options. *Journal of Economic Dynamics and Control*, 24:1563–1590, 2000.