Module 5: Modelling Interest Rate Derivatives
Pricing of a Bermudan Swaption using Monte-Carlo simulation

November 5, 2018
Abstract

In the following assignment, a Bermudan Swaption pricer based on the One-Factor Hull-White Short Rate Model is implemented using Monte Carlo simulation. For this purpose, the Hull-White model is calibrated to market data and subsequently an algorithm, which has been discussed in the lectures, is implemented to determine the optimal exercise strategy and thus the price of the Bermudan Swaption. The method proposed by Longstaff & Schwartz [LS01] is implemented in Matlab to derive a regression-based lower bound on the price of the Bermudan Swaption.

Contents

1 Preliminaries 1

2 Hull-White Model Overview and Model Calibration 3
   2.1 Calibration to ATM Swaptions .......................... 3
   2.2 Simulation of the short-rate ................................. 5

3 Theory and Implementation of Regression-Based Monte Carlo Methods 7
   3.1 Dynamic Programming Formulation ............................. 8
   3.2 Longstaff-Schwartz Monte Carlo method .............................. 8

4 Numerical Results 10

A Appendix: Market Data 13

B Appendix: Matlab Code 15
   B.1 Calibration algorithm ........................................... 15
   B.2 Generation of transformed short-rate paths ......................... 17
   B.3 Longstaff-Schwartz algorithm ..................................... 19
   B.4 Auxiliary functions .............................................. 21

1 Preliminaries

A swaption is a vanilla interest-rate derivative which represents an option on a fixed-for-floating swap. In case of a Payer Swaption it represents the right of the holder to enter a swap with a given maturity with a fixed, previously agreed rate \( k \), at a future time. If the holder has only the right to exercise the option at its maturity, we refer to it as a so-called European Swaption. In case the holder has the right to exercise it at some pre-defined fixing dates, it is referred to as a Bermudan Swaption. For a given Swaption that expires at time \( T_0 \) the payoff (at \( T_0 \)) is then defined as the maximum value of comparing the value of the underlying swap and zero:

\[
V_{\text{swaption}}(T_0) = \max(V_{\text{swap}}(T_0), 0)
\]

(1)

The value \( V_{\text{swap}} \) of the swap at time \( T_0 \) is then given as (see for example [BM07])

\[
V_{\text{swap}}(T_0) = \sum_{n=0}^{N-1} \tau_n P(T_0, T_{n+1})(L_n(T_0) - k),
\]

(2)
where

$$L_n(T_0) = L_n(T_0, T_n, T_{n+1}) = \frac{P(T_0, T_n) - P(T_0, T_{n+1})}{\tau_n P(T_0, T_{n+1})}, \quad (3)$$

is defined as the simple forward interest rate defined at time $T_0$ for the period from $T_n$ to $T_{n+1}$, $k$ represents the agreed fixed rate of the swap, $\tau_n$ the day count fraction between two payment dates and $P(T_0, T_{n+1})$ is the discount factor of the future payment ($L_n(T_0) - k$) until time $T_0$.

Let $\tau_i := T_i - T_{i-1}$ for $k = 1, \ldots, n$, to denote the time between two payments. Define $A(t; T_0, T_N) = \sum_{i=0}^{N-1} \tau_i P(t, T_{i+1})$ as the Annuity factor at time $t$ from $T_0$ to $T_N$. $P(t, T_i)$ is the price of a zero coupon bond at time $t$ maturing at time $T_i$.

The swap rate $R^*(T_0; T_n, T_N)$ is then defined as

$$R^*(T_0; T_n, T_N) = \frac{P(T_0, T_n) - P(T_0, T_N)}{A(T_0; T_n, T_N)}, \quad (4)$$

Substituting, equation (3) into (2) and using the relation (4), the value of a spot swap at time $T_0$ can be expressed as

$$V_{\text{swap}}(T_0; T_0, T_N, k) = A(T_0; T_0, T_N)(R^*(T_0; T_0, T_N) - k). \quad (5)$$

The value of a swaption at expiry $T_0$ can then be expressed using the swap value as payoff

$$V_{\text{swaption}}(T_0; T_0, T_N, k) = \max(A(T_0; T_0, T_N)(R^*(T_0; T_0, T_N) - k), 0), \quad (6)$$

and for times $t < T_0$ under the risk-neutral measure $\mathbb{Q}$ as

$$V_{\text{swaption}}(t; T_0, T_N, k) = \mathbb{E}_t \left[ e^{-\int_t^{T_0} \rho(s)ds} A(T_0; T_0, T_N) \max \left( (R^*(T_0; T_0, T_N) - k), 0 \right) \right]. \quad (7)$$

By use of the Radon-Nikodym derivative

$$\frac{d\mathbb{Q}^A}{d\mathbb{Q}} \bigg|_t = \frac{A(T_0)/A(t)}{\beta(T_0)/\beta(t)} \quad (8)$$

to switch from the risk-neutral measure under which the discounted price of an asset is a martingale to the so-called swap or annuity measure and using $\beta(t) = e^{\int_t^T \rho(s)ds}$, the value of the swaption can be expressed as

$$V_{\text{swaption}}(t; T_0, T_N, k) = \mathbb{E}_t \left[ e^{-\int_t^{T_0} \rho(s)ds} A(T_0) \max \left( (R^*(T_0; T_0, T_N) - k), 0 \right) \right]$$

$$= \beta(t) \mathbb{E}_t \left[ \beta(T_0)^{-1} A(T_0) \max \left( (R^*(T_0; T_0, T_N) - k), 0 \right) \right]$$

$$= \beta(t) \mathbb{E}_t \left[ \beta(T_0)^{-1} A(T_0) \frac{\beta(T_0)}{A(T_0)} \frac{\beta(T_0)/\beta(t)}{A(t)/A(T_0)} \max \left( (R^*(T_0; T_0, T_N) - k), 0 \right) \right]$$

$$= \beta(t) \mathbb{E}_t \left[ \max \left( (R^*(T_0; T_0, T_N) - k), 0 \right) \right], \quad (9)$$

where $A(t) = A(t; T_0, T_N)$ and $A(T_0) = A(T_0; T_0, T_N)$ was used to simplify notation.

After having defined the payoff of an European Swaption, the characteristics of a Bermudan Swaption are briefly defined (see [BM07]).

**Definition 1.1 (Bermudan Payer Swaption).** A Bermudan Payer Swaption is defined by three distinct dates $T_K < T_H < T_N$, where the holder of the option is allowed to enter at pre-defined dates between $T_K$ and $T_H$ (the swaption expiry date) into a swap at time $T_L$ with first reset at time $T_L$, expiry at time $T_N$ and fixed rate $k$.

The swap start date and length therefore strongly depend on the chosen exercise time $T_L$. For completeness it is noted, that Bermudan Swaptions can also include a no-call period, meaning that the swaption can only be called after a pre-defined date.
2 Hull-White Model Overview and Model Calibration

The Hull-White interest rate model is a so-called short-rate term structure model as it directly models the short rate process $dr(t)$. The model was developed as an extension to the Vasicek model in the seminal paper by Hull and White [HW90] and is therefore referred to as the "extended Vasicek model" for the reason that the respective parameters of the Vasicek model are made time-dependent. In this assignment, the version of the Hull-White interest rate model is used where the level of mean-reversion is a function of time. The dynamics of the short rate under the risk-neutral measure $Q$ evolves according to the following SDE, see [HW94]:

$$dr(t) = \left(\theta(t) - \alpha r(t)\right)dt + \sigma dW_t, \quad (10)$$

where $\theta(t)$ determines the time-dependent level of mean-reversion, $\alpha$ the speed of mean-reversion, $\sigma$ the volatility of the interest rate process and $W$ represents a $Q$-Brownian motion.

The solution to (10) can be written as (see for example [Gla03])

$$r(t) = r(u)e^{-\alpha(t-u)} + \int_u^t \theta(s)e^{-\alpha(t-s)}ds + \sigma \int_u^t e^{-\alpha(t-s)}dW_s, \quad (11)$$

One of the features of the Hull-White model that was often often criticized is that the model allows negative interest rates. However, due to the recent financial crisis and the current negative interest rate environment, the model has grown in importance again. This can be seen as the short rate has a normal distribution with mean and variance. Given the solution (11), the conditional distribution of the Hull-White model can be computed, with mean and variance given as

$$\mathbb{E}[r(t)|F_u] = r(u)e^{-\alpha(t-u)} + \int_u^t \theta(s)e^{-\alpha(t-s)}ds$$

$$\text{Var}[r(t)|F_u] = \frac{\sigma^2}{2\alpha}(1 - e^{-2\alpha(t-u)}), \quad (12)$$

where $\mathcal{F}_u$ denotes the filtration of the process up to time $u$. It is therefore Gaussian with $\mathcal{N} \sim (r(u)e^{-\alpha(t-u)} + \int_u^t \theta(s)e^{-\alpha(t-s)}ds, \frac{\sigma^2}{2\alpha}(1 - e^{-2\alpha(t-u)}))$.

The path of $r$ for the times $t = t_0, t_1, ..., t_{n-1}, t_n$ can therefore be simulated exactly by realizing that $r$ can be represented as a standard normal Gaussian, shifted by the expectation and multiplied by the variance.

$$r(t_{i+1}) = r(t_i)e^{-\alpha(t_{i+1}-t_i)} + \int_{t_i}^{t_{i+1}} \theta(s)e^{-\alpha(t_{i+1}-s)}ds + \text{Var}[r(t_{i+1})|\mathcal{F}_{t_i}] Z_i, \quad (14)$$

where $Z_1, ..., Z_n$ represent i.i.d. standard normal variables generated using a random number generator. One can therefore refrain from discretizing the SDE using an approximation scheme such as the Euler scheme and directly simulate the path given by the formula (14).

2.1 Calibration to ATM Swaptions

To use the Hull-White effectively to price Bermudan Swaptions, the model has to be calibrated to observed market prices of liquidly traded instruments. The calibration procedure can use either vanilla swaptions or caps/floors, see [AA01]. For the assignment, at-the-money vanilla swaption
prices are used to calibrate the model. The parameters that have to be calibrated for the Hull-White model are $\alpha$ and $\sigma$. The time-dependant mean-reversion parameter $\theta(t)$ is then given as a function of $\alpha$ and $\sigma$ and the initial instantaneous forward rate that can be observed in the market for maturity $T$ by the relation

$$f^*(0, T) = -\frac{\partial \ln P^*(0, T)}{\partial T},$$

with $P^*(0, T)$ as the current market discount factor for maturity $T$. [BM07] show that $\theta(t)$ can then be computed as

$$\theta(t) = \frac{\partial f^*(0, t)}{\partial T} + \alpha f^*(0, t) + \frac{\sigma^2}{2\alpha}(1 - e^{-2\alpha}),$$

where $\frac{\partial f^*(0, t)}{\partial T}$ denotes the partial derivative of $f^*$ with respect to the maturity argument, and $\alpha$ and $\sigma$ as calibrated from observed market prices. The derivative can be computed using finite differences using the observed forward rates in the market.

Since the emergence of negative interest rates in the early 2010's, the until then predominately used Black formula to calculate prices of European Swaptions has become ineffective as it does not allow negative interest rates. Under the Black76 model the forward rate is lognormally distributed with a constant volatility $\sigma_{B76}$:

$$dF_{B76} = \sigma_{B76} F dW_t$$

The price of a call option under the Black76 model can then be computed as, see [Bla76]:

$$c_{B76} = e^{-r(T-t)}[F_t \Phi(d^+) - K \Phi(d^-)]$$

with $d^+$ and $d^-$ are given as

$$d^+ = \ln\left(\frac{F_t}{K}\right) + \frac{(\sigma_{B76}^2/2)(T-t)}{\sigma_{B76}\sqrt{T-t}}, \quad d^- = d^+ - \sigma_{B76}\sqrt{T-t}$$

where $\sigma_{B76}$ represents the volatility, $K$ the strike of the option and $\Phi(x)$ the CDF of the normal distribution.

In case the forward rate turns negative, as seen in the current interest rate environments in Switzerland and Europe, the logarithm of $\ln(F_t/K)$ is not defined and therefore the formula (17) cannot be used. It is therefore now common market practice to quote implied volatilities not only based on the Black76 model but also on the so-called Normal model, also known as Bachelier model, which assumes the forward rate to be normally distributed according to the following SDE:

$$dF_t = \sigma_N dW_t,$$

where $\sigma_N$ is the volatility under the Normal model which is different from the the Black76 volatility $\sigma_{B76}$ seen above.

Similarly to the closed-form price of a call option under the Black76 model, there exists also a analytical solution under the Normal model (see [DDCB09] for proof):

$$c_N = e^{-r(T-t)}[(F_t - K)\Phi(d) + \sigma_N \sqrt{T-t} \phi(d)]$$

with $d$ given as

$$d = \frac{F_t - K}{\sigma_N \sqrt{T-t}},$$

4
where $\Phi(x)$ is again the CDF of a normal distribution and $\phi(x)$ denotes the PDF of a normal distribution respectively.

The task of calibration is then defined as finding the parameters $\alpha$ and $\sigma$ of the Hull-White model such that the model-implied price $P_{i,Hull-White}$, i.e. the price of swaption $i$ under the Hull-White model, matches the market price $P_{i,market}$, implied by observed normal volatilities in the market, across all swaptions $i = 1, ..., N$ used for calibration. A common proposed measure for the fit of calibration can be found in Clewlow and Strickland [CS03], who suggest to minimize the sum of squared proportional differences between $P_{i,Hull-White}$ and $P_{i,market}$ in relation to the market price, i.e.

$$\min \left[ \sum_{i=1}^{N} \left( \frac{P_{i,Hull-White} - P_{i,market}}{P_{i,market}} \right)^2 \right]$$  \hspace{1cm} (21)

The required formula to derive the market price of the swaption given an implied normal volatility is based on the Normal model and given as

$$P_{i,market}(K, S(0), T_0, T_N, \sigma_N) = A(0; T_0, T_N)c_N(K, S(0), T_0, \sigma_N),$$ \hspace{1cm} (22)

where $c_N$ is the call price of an option under the Bachelier model defined in equation (19), with $K = S(0)$ because the quoted volatilities are valid for at-the-money swaptions, $T_0$ is the expiry of the swaption, $T_N$ is the maturity of the underlying swap and $\sigma_N$ is the market implied volatility in basis points.

To proceed with the calibration procedure one needs also to calculate the price of a swaption under the Hull-White model. Re-writing the value of a swap in equation (5) as

$$V_{swap}(T_0; T_0, T_N, k) = 1 - P(T_0, T_N) - kA(T_0; T_0, T_N)$$
$$= 1 - P(T_0, T_N) - k \sum_{i=0}^{N-1} \tau_i P(T_0, T_{i+1}),$$ \hspace{1cm} (23)

the payoff of a swaption can be written as an option on a sum of discount bonds

$$V_{swaption}(T_0; T_0, T_N; k) = \max(1 - P(T_0, T_N) - k \sum_{i=0}^{N-1} \tau_i P(T_0, T_{i+1}), 0).$$ \hspace{1cm} (24)

Jamshidian [Jam89] has shown, what is called as Jamshidian’s trick, that this can be written as the sum of options on discount bonds (see [Jam89] for details). Using the decomposition by Jamshidian, the price $P_{i,Hull-White}(K, T_0, T_N)$ is calculated and the required parameters $\alpha$ and $\sigma$ are determined.

### 2.2 Simulation of the short-rate

To simulate the paths of the Hull-White interest rate model (10) the model parameters $\alpha$ and $\sigma$ are calibrated first and subsequently the stochastic differential equation is simulated from the distribution of the solution to the Hull-White model.

It is often more convenient not to simulate the Hull-White process as defined under equation (10) but under a transformation of the process, see [AP10]. The new process is defined as:

$$x(t) = r(t) - f(0, t),$$ \hspace{1cm} (25)

where the new process $x(t)$ is defined as the difference between the short rate $r(t)$ and the instantaneous forward rate $f(0, t)$.
From the equation above it is clear, that the new process \( x(t) \) starts at \( x(0) = 0 \) since, \( f(0, 0) = r(0) \). [AP10] show that the transformed process follows the SDE

\[
\frac{dx(t)}{dt} = (y(t) - \alpha x(t)) \, dt + \sigma_r \, dW_t, \quad (26)
\]

where

\[
y(t) = \frac{\sigma_r^2}{2\alpha} (1 - e^{-2\alpha t}) \quad (27)
\]

and \( \sigma_r \) is the volatility of the short-rate process defined under (10) with \( \sigma_r \) and \( \alpha \) being constant.

The process \( dx(t) \) therefore also follows a Hull-White process where \( \theta(t) \) has been replaced by a deterministic term \( y(t) \) that is independent of the instantaneous forward rate observed in the market. Furthermore, under the new process the price of a zero coupon bond can be expressed analytically by the following expression:

\[
P(t, T) = \frac{P^*(0, T)}{P^*(0, t)} \exp \left( -x(t)G(t, T) - \frac{y(t)}{2}G(t, T)^2 \right) \quad (28)
\]

\[
G(t, T) = \frac{1}{\alpha} \left( 1 - e^{-\alpha(T-t)} \right). \quad (29)
\]

**Proof**, see [AP10]

\( P^*(0, T) \) and \( P^*(0, t) \) are derived from the initially observed term structure and \( y(t) \) is defined by equation (27). The solution for (26) can be derived similar to the regular Hull-White process. Apply Ito’s Lemma to the function \( f(x, t) = x(t)e^{\alpha t} \) to obtain:

\[
d(x(t)e^{\alpha t}) = e^{\alpha t}dx(t) + \alpha x(t)e^{\alpha t}dt = e^{\alpha t}((y(t) - \alpha x(t)) \, dt + \sigma_r \, dW_t) + \alpha x(t)e^{\alpha t}dt
\]

Integrating the expression with \( u < t \) yields:

\[
\int_u^t d(x(t)e^{\alpha t}) = \int_u^t e^{\alpha u}y(s)ds + \int_u^t e^{\alpha s}\sigma_r \, dW_s
\]

\[
x(t)e^{\alpha t} = e^{\alpha u}x(0) + \int_u^t e^{\alpha s}y(s)ds + \int_u^t e^{\alpha s}\sigma_r \, dW_s
\]

The general solution can then be expressed as:

\[
x(t) = e^{-\alpha(t-u)}x(u) + \int_u^t e^{-\alpha(t-s)}y(s)ds + \int_u^t e^{-\alpha(t-s)}\sigma_r \, dW_s \quad (30)
\]

The solution (30) of the transformed Hull-White short-rate process is also Gaussian with conditional mean and variance similar to the one derived in Section 2. The conditional expectation of an increment from time \( t_{i+1} \) to \( t_i \) can be derived as follows:

\[
\mathbb{E}[x(t_{i+1})|x(t_i)] = e^{-\alpha(\Delta t)}x(t_i) + \int_{t_i}^{t_{i+1}} e^{-\alpha(t_{i+1}-s)}y(s)ds + 0
\]

\[
= e^{-\alpha(\Delta t)}x(t_i) + \frac{\sigma_r^2}{2\alpha} \int_{t_i}^{t_{i+1}} e^{-\alpha(t_{i+1}-s)}(1 - e^{-2\alpha s}) \, ds
\]

\[
= e^{-\alpha(\Delta t)}x(t_i) + \frac{\sigma_r^2}{2\alpha^2}(1 + e^{-2\alpha t_{i+1}} - e^{-\alpha(\Delta t)} - e^{-\alpha(t_{i+1}+t_i)}), \quad (31)
\]

where \( y(s) \) was defined in (27) and \( \Delta t = t_{i+1} - t_i \).
The conditional variance $\text{Var}[x(t_{i+1})|x(t_i)]$ can be computed similarly as
\[
\text{Var}[x(t_{i+1})|x(t_i)] = \text{Var}\left(\int_{t_i}^{t_{i+1}} e^{-\alpha(t_{i+1}-s)} \sigma_r dW_s\right)
\]
\[
= \int_{t_i}^{t_{i+1}} (e^{-\alpha(t_{i+1}-s)} \sigma_r)^2 ds
\]
\[
= \sigma_r^2 \int_{t_i}^{t_{i+1}} (e^{-2\alpha(t_{i+1}-s)}) ds
\]
\[
= \frac{\sigma_r^2}{2\alpha} \left(1 - e^{-2\alpha \Delta t}\right). \tag{32}
\]

The process $dx(t)$ can therefore be simulated in a bias free manner by discretizing $x(t)$ for the times $t = t_0, t_1, \ldots, t_{n-1}$ and applying
\[
x(t_{i+1}) = e^{-\alpha\Delta t} x(t_i) + \frac{\sigma_r^2}{2\alpha^2} (1 + e^{-2\alpha t_{i+1}} - e^{-\alpha\Delta t} - e^{-\alpha(t_{i+1}+t_i)}) + \frac{\sigma_r}{\sqrt{2\alpha}} \sqrt{1 - e^{-2\alpha \Delta t}} Z_i. \tag{33}
\]

Assume the European Swaption has a payoff as specified in equation (6). Under the transformed process (25), the price of any financial instrument can be written as (in the $Q$ measure)
\[
V(0) = \mathbb{E}\left[e^{-\int_0^T r(s) ds} G(T)\right]
\]
\[
= \mathbb{E}\left[e^{-\int_0^T x(s) + f(0,s) ds} G(T)\right]
\]
\[
= P^*(0,T) \mathbb{E}\left[e^{-\int_0^T x(s) ds} G(T)\right], \tag{34}
\]
with $G(T)$ being the non-negative payoff of the option at time $T$ and where the fact was used, that $P^*(0,T) = e^{-\int_0^T I(0,s) ds}$ and $P^*(0,T)$ is observable in the market today and can therefore be pulled out of the unconditional expectation. The task of pricing any financial instrument then consists of simulating the evolution of $x(t)$ until expiry $T$ of the option using the relation (33) derived above across a number of paths $j = 1, \ldots, N$. Subsequently, the price can be computed by averaging the payoff discounted by the transformed discounting factor $e^{-\int_0^T x(s) ds}$ multiplied by the currently observed zero coupon bond price with the same maturity.

Define $I(0,T) = -\int_0^T x(s) ds$ and the transformed discount factor as $D(0,T) = e^{I(0,T)}$, an approximation scheme is used so that
\[
I(t_{i+1}) = I(t_i) - x(t_i) \Delta t. \tag{35}
\]

3 Theory and Implementation of Regression-Based Monte Carlo Methods

The Monte Carlo method is usually well-suited for pricing of high dimensionality problems or problems that depend on the whole path of the short-rate. In contrast, for problems with early exercise the Monte Carlo method is usually not the method of first choice as we travel forward in time and therefore would need expensive sub-simulations at each potential exercise date. Nevertheless, a considerable amount of research has been devoted to Monte Carlo method to price American or Bermudan options with early exercise features.
3.1 Dynamic Programming Formulation

Every American Option pricing problem can be defined by a dynamic programming formulation. Assume that the American option can only be exercised at discrete time intervals $t_1 < t_2 < \ldots < t_m$, where $t_m$ represents the last exercise opportunity and can be defined as $t_m = T$, i.e. we refer to this as a Bermudan option. To simplify notation, the payoff $G(t_i)$ at time $t_i$ will be written as $G_i$ in the following section. Denote by $X_t$ the state of the underlying process at time $t$. Then the dynamic programming principle can be defined as, see [Gla03]

$$V_m(x) = G_m(x)$$

$$V_{i-1}(x) = \max\{G_{i-1}(x), \mathbb{E}[D_{i-1,i}(X_i)V_i(X_i)|X_{i-1} = x]\}, \quad i = 1, \ldots, m.$$  

where $D_{i-1,i}$ represents a general discount factor based on the underlying process and $V_i(X_i)$ the value of the option at time $t_i$, given the current state $X_i = x$. The principle then states, that we choose at each exercise opportunity between the payoff at the current state $G_{i-1}(x)$ and the continuation value by not exercising the option. The continuation value can therefore be written as

$$C_i(x) = \mathbb{E}[D_{i-1,i}(X_i)V_i(X_i)|X_{i-1} = x].$$

3.2 Longstaff-Schwartz Monte Carlo method

The Longstaff-Schwartz Monte Carlo method, sometimes also called the Least Squares Monte Carlo method, was developed by Longstaff and Schwartz [LS01] to value American options using the Monte Carlo method based on the previously stated dynamic programming principle by the use of regression methods. Similar methods using regression based Monte Carlo have been proposed by Carrière [Car96], and Tsitsiklis and Van Roy [TVR01]. To compute the continuation value $C_i(x)$ defined in (38), the algorithm performs a regression of the value of the option $V_{i+1}(X_{i+1})$ on the current state of the underlying process $x$. The continuation value can then be expressed by

$$\mathbb{E}[D_{i-1,i}(X_i)V_i(X_i)|X_{i-1} = x] = \sum_{l=0}^{k} \hat{\beta}_l \psi_l(x),$$

where $\psi_l(x)$, $l = 1, \ldots, k$ represents a set of basis function that describe the relation between continuation value and the state of the current underlying process, and $\hat{\beta}_l$ are the estimated parameters at time $t_i$ for basis function $\psi_l$.

Denote the current state of the underlying process at time $t_i$ for path $j$ by $X_{i,j}$ for $j = 1, \ldots, N$. Longstaff and Schwartz then suggest that only nodes for which the payoff is positive should be included in the regression performed in (39), i.e. $G_i(X_{i,j}) > 0$. Traversing backwards from maturity over the pre-defined exercise dates, the regression coefficients at each time $t_i$ can then be obtained by regressing the discounted payoffs

$$y_{i,j} = e^{-\int_{t_{i,j}}^{t_i} x_j(s)ds} V_j(X_{i,j})$$

on the values of the underlying stochastic process at time $X_{i,j} = x_{i,j}$ for path $j$ by least squares regression method. The coefficients at time $t_i$ are then obtained as

$$\left(\hat{\beta}_0, \ldots, \hat{\beta}_k\right)^T = \left(\psi(X_i)\psi(X_i)^T\right)^{-1}\psi(X_i)(y_{t_i+1,1}, \ldots, y_{t_i+1,N})^T,$$
with $\psi(x) = (\psi_1(x), \ldots, \psi_k(x))^\top$.

To perform the regression-based Monte Carlo approach described above, one clearly has to choose a suitable set of basis functions. The task of choosing basis functions is by far trivial as if one chooses too few basis functions, the continuation value cannot be properly estimated from the performed regression. On the other hand, if one chooses too many of such explanatory variables, there is a danger of overfitting which not only leads to more numerical work but also reduces the quality of the estimate. Ultimately, the chosen basis functions serve as a basis to approximate the information we have up to time $T$. [AP10] argue that for a Bermudan Swaption, the main information is captured in the overall level of interest rates on each exercise date and in the slope of the yield curve. As such, they suggest to measure the overall level of interest rates by using the swap rate or the value of a swap which starts on the exercise date and matures with the Bermudan Swaption itself. A second basis function should then be chosen to measure the slope of the yield curve. This can be achieved by either using a forward starting swap, which starts on the next exercise date, or better the spot Libor rate from the exercise date until the next period. For the purpose of pricing, in this assignment the swap rate to measure the overall level of interest rates and the spot Libor rate to measure the slope of the curve will be used. In Section 4 the chosen basis functions are presented again.
4 Numerical Results

In the next section, the results of the implemented Bermudan Swaption pricer are presented. The valuation date and all market data are retrieved per October, 11th 2018. The swaption that will be priced is a 10x15 Bermudan Swaption, where the option expires after 10 years and the underlying swap matures after 15 years. The option will have annual exercise dates until the expiry of the option in 10 years, starting with the first reset date in year 1. Depending on the chosen exercise date of the Bermudan Swaption, the length of the swap which the holder of the option receives after exercising, varies between 14 years (if exercised at the first exercise date) and 5 years (if exercised at expiry of the swaption).\footnote{For simplicity in the valuation of the Bermudan Swaption, it is assumed that the fixing date always falls on the beginning of the interest rate period, i.e. the fixing offset is equal to zero, and the payment date always falls on the end of the interest rate period, i.e. pay offset is equal to zero. Further, no date adjustments will be performed.}

For calibration purposes, the relevant co-terminal European Swaptions are used, i.e. all swaptions that have same underlying maturity of the swap and option. For a 10x15 Bermudan Swaption the respective European Swaptions and their respective normal volatilities are presented below in Table 1:\footnote{Missing values are linearly interpolated.}

<table>
<thead>
<tr>
<th>Swaption</th>
<th>Volatility (bps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x14</td>
<td>54.41</td>
</tr>
<tr>
<td>2x13</td>
<td>55.43</td>
</tr>
<tr>
<td>3x12</td>
<td>55.69</td>
</tr>
<tr>
<td>4x11</td>
<td>62.70</td>
</tr>
<tr>
<td>5x10</td>
<td>69.71</td>
</tr>
<tr>
<td>6x9</td>
<td>70.91</td>
</tr>
<tr>
<td>7x8</td>
<td>71.77</td>
</tr>
<tr>
<td>8x7</td>
<td>72.33</td>
</tr>
<tr>
<td>9x6</td>
<td>72.57</td>
</tr>
<tr>
<td>10x5</td>
<td>72.51</td>
</tr>
</tbody>
</table>

Table 1: Overview of European Swaptions and normal volatilities used for calibration

Applying the above mentioned calibration process to the volatility data of the European Swaptions quoted in the market, gives the following estimates for $\alpha$ and $\sigma$:

$$\hat{\alpha} = 0.0563$$
$$\hat{\sigma}_r = 0.029.$$

These parameters are subsequently used to price the 10x15 Bermudan Swaption with a final maturity of the swap in 15 years. For the purpose of pricing, 10'000 paths are generated for the process $x(t)$ using the simulation procedure defined in equation (33). Similarly, the process $I(t)$ defined in equation (35) is simulated and the transformed discount factor $D(0,T) = e^{I(0,T)}$ is calculated. In Figure 1, the generated paths for $x(t)$ until maturity of the Swaption as well as the transformed discount factors are plotted for the first 1’000 paths. These paths serve as a basis to evaluate the payoffs of the Bermudan Swaption at each exercise date.
Having simulated the paths for the process $x(t)$, the price of an European Swaption can be computed using equation (34). Under the Hull-White model the price of a Swaption is solely determined by the yield curve simulated up to time $T$, i.e. the expiry of the option. With the help of the bond reconstitution formula derived in equation (29), the swap rate at time $T$ can be computed and as a result the price of the European Swaption is obtained. The European Swaption will be exercised if the underlying swap has a positive value.

To derive the value of the Bermudan Swaption, the simulated period of 10 years, is divided into 10 intervals, each corresponding to one exercise date of the Swaption. For the purpose of pricing, different basis functions have been considered and combined. As discussed in the previous section, the swap rate and the spot Libor rate are used as regressors to explain the continuation value. In addition, the squared swap and spot Libor rate as well as the quotient of swap rate and spot Libor rate have been considered. Table 2 displays the computed prices of the Bermudan Swaption based on a notional of 100EUR and based on 10’000 paths for different sets of basis functions. $S$ represents the swap rate of a swap starting at the exercise date and maturing with the Bermudan Swaption and $L$ represents the spot Libor rate from the exercise date until the next exercise date$^3$:

$^3$All regressions contain a constant which is not explicitly listed under the basis functions
Table 2: Estimated values for a Bermudan Payer swaption implied by the Hull-White one factor model. The swaption allows the holder to enter into a swap with a maturity of 15 years, paying a fixed coupon of 1% and receiving a floating six-month rate. The fixed side is paid annually, the floating side is paid semi-annually. Values are expressed per 100EUR notional amount.

<table>
<thead>
<tr>
<th>Basis Functions</th>
<th>Bermudan Swaption price</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S, L$</td>
<td>2.104</td>
</tr>
<tr>
<td>$S, L, S^2, L^2$</td>
<td>2.230</td>
</tr>
<tr>
<td>$S, L, S^2, L^2, S \times L$</td>
<td>2.150</td>
</tr>
</tbody>
</table>

The price of the Bermudan Payer Swaption was calculated as being around 2.20 EUR. The results show that including more basis functions in the regression for the continuation value, does not have substantial effects on the price of the Bermudan payer swaption. Increasing the strike shows, that the value of the Bermudan payer swaption decreases as expected, as the holder of the option will have to pay the fixed rate when he enters the swap. An increase in the strike therefore reduces the value of the swaption.
A Appendix: Market Data

Figure 2: Euro Area Yield Curve for AAA rated companies - Data retrieved from ECB website per Oct. 11th, 2018 - [ecb].
Figure 3: Normal volatilities for ATM Swaptions quoted in basis points (bps) - Data retrieved from Bloomberg per Oct. 11th, 2018.
B Appendix: Matlab Code

B.1 Calibration algorithm

% Method called to calibrate the parameters 'alpha' and 'sigma_r' to the
% selected European Swaptions (calibInstruments)
function [alpha, sigma_r] = calibration(alpha0, sigma_r0, intCurve, calibInstruments)

% Load market data used for calibration
[timeToExpiryVol, timeToMaturityVol, timeToExpiryStrike, timeToMaturityStrike, volValues, volStrikes] = loadMarketData();

% Set initial guesses for 'alpha' and 'sigma_r'
v = [alpha0, sigma_r0];

% Create Handle to function
fun = @(v)minSum(v, intCurve, calibInstruments, timeToExpiryVol, timeToMaturityVol, timeToExpiryStrike, timeToMaturityStrike, volValues, volStrikes);

% Minimize the sum of squared relative differences
[res, fval] = fminsearch(fun, v);

alpha = res(1);
sigma_r = res(2);

end

% Function that performs the optimization
function res = minSum(v, intCurve, calibInstruments, timeToExpiryVol, timeToMaturityVol, timeToExpiryStrike, timeToMaturityStrike, volValues, volStrikes)

% Initialize parameters and matrices
alpha = v(1);
sigma_r = v(2);

sizeCalibs = size(calibInstruments);
HWSwaptionValues = zeros(sizeCalibs(1),1);
NormalSwaptionValues = zeros(sizeCalibs(1),1);
x0 = 0.01;
Computes the value of the Swaption under the Hull–White model using Jamshidian’s decomposition

% Computes the value of the Swaption under the Hull–White model
% using Jamshidian’s decomposition
for i=1:sizeCalibs(1)
    timeToExpiry = calibInstruments(i,1);
    timeToMaturity = calibInstruments(i,2);
    strike = interp2(timeToMaturityStrike, timeToExpiryStrike, volStrikes, timeToMaturity, timeToExpiry, 'linear');

    % Search for value of x, for which the swap at time T0 is equal to zero
    fun = @(x) getRate(x, timeToExpiry, timeToMaturity, alpha, sigma_r, intCurve, strike);
    x_star = fzero(fun, x0);

    % Compute swaption price using Jamshidian’s decomposition using formula from Andersen and Piterbarg (2010)
    valueSwaptionHW = 0;
    for j=1:timeToMaturity
        kBonds = xBondPrice(timeToExpiry, timeToExpiry+j, x_star, alpha, sigma_r, intCurve);
        valueSwaptionHW = valueSwaptionHW + strike * optionBondPrice(alpha, sigma_r, timeToExpiry, timeToExpiry+j, kBonds, intCurve);
    end
    HWSwaptionValues(i) = valueSwaptionHW + kBonds * optionBondPrice(alpha, sigma_r, timeToExpiry, timeToExpiry+timeToMaturity, kBonds, intCurve);
end

% Compute the market prices under the Normal model, using the market volatilities as input
for i=1:sizeCalibs(1)
    timeToExpiry = calibInstruments(i,1);
    timeToMaturity = calibInstruments(i,2);
    sigma_market = interp2(timeToMaturityVol, timeToExpiryVol, volValues, timeToMaturity, timeToExpiry, 'linear');
    NormalSwaptionValues(i) = NormalBlackATMSwaptionFormula(sigma_market, timeToExpiry, timeToMaturity, intCurve);
end
% Compute sum of squared relative differences
diffPrices = ((HWSwaptionValues - NormalSwaptionValues)./NormalSwaptionValues).^2;
res = sum(diffPrices,1);

% Function to compute the par swap rate needed for Jamshidian’s
do
decomposition
function y = getRate(x, timeToExpiry, timeToMaturity, alpha, sigma_r, intCurve, strike)
    y = xBondPrice(timeToExpiry, timeToExpiry+timeToMaturity, x, alpha, sigma_r, intCurve);
    for i=1:timeToMaturity
        y = y + strike*xBondPrice(timeToExpiry, timeToExpiry+i, x, alpha, sigma_r, intCurve);
    end
    y = y - 1;
end

% Function to compute the option price on a coupon bond following
formula
% from Damiano and Brigo (2007)
function res = optionBondPrice(alpha, sigma_r, timeToExpiry, timeToMaturity, strikeX, intCurve)
bond1 = marketZeroBondPrice(intCurve, (timeToExpiry+timeToMaturity));
bond2 = marketZeroBondPrice(intCurve, timeToExpiry);
sigma_p = sigma_r/alpha * (1-exp(-alpha*timeToMaturity))*((1-exp(-2*alpha*(timeToExpiry)))/(2*alpha))^-0.5;
h = 1/sigma_p*log(bond1/(bond2*strikeX))+sigma_p/2;
res = bond1*N(h)-strikeX*bond2*N(h-sigma_p);
end

B.2 Generation of transformed short-rate paths

% Generates transformed HullWhite process paths.
% The process is transformed by setting: x(t) = r(t) - f(0,t)
function [shortRate, discountFactor] = GenerateShortRate(alpha, sigma_r,
timeToExpiry, paths)

% set seed
randn('state', 100);

% gets deltaT: I am using time expressed in days, so I divide by 250
deltaT = 1 / 250;

nbrsteps = timeToExpiry/deltaT;

shortRate = zeros(nbrsteps, paths);
discountFactor = zeros(nbrsteps, paths);
discountFactor(1,:) = 1;

% generates uncorrelated random numbers ...
RandNumbers = randn(nbrsteps, paths);

% generates paths
for i = 1:paths

% x(0) = 0
x_ti = 0;
I_ti = 0;

for j = 2:nbrsteps

% Compute current time and time of period before
  til = j*deltaT;
  ti = (j-1)*deltaT;

  alpha_int = exp(-alpha*deltaT);
  time_term = sigma_r^2/(2*alpha^2)*(1+exp(-2*alpha*til)-exp(-alpha*deltaT)-exp(-alpha*(til+ti)));
  var_xt = sigma_r^2/(2*alpha)*(1-alpha_int^2);

  x_expectation = alpha_int*x_ti + time_term;
  x_variance = sqrt(var_xt);

  % Bias free simulation of x_til
  x_til = x_expectation + x_variance*RandNumbers(j,i);

  % Approximate discretization of I_til
  I_til = I_ti - x_til*deltaT;
\begin{align*}
\text{x}_{t_i} & = \text{x}_{t_{i-1}}; \\
\text{I}_{t_i} & = \text{I}_{t_{i-1}}; \\
\text{shortRate}(j,i) & = \text{x}_{t_i}; \\
\text{discountFactor}(j,i) & = \exp(\text{I}_{t_i}); \\
\end{align*}

\begin{figure}
\centering
\begin{minipage}{\textwidth}
\begin{verbatim}
  f = figure; 
  hold on; 
  subplot(2,1,1); 
  plot(shortRate); 
  ylabel('x') 
  xlabel('time (in days)') 

  subplot(2,1,2); 
  plot(discountFactor); 
  ylabel('Discount Factor')
  xlabel('time (in days)')
\end{verbatim}
\end{minipage}
\end{figure}

\subsection*{B.3 Longstaff-Schwartz algorithm}

\begin{verbatim}
% prepares workspace
format long; 
clear all; 
clc; 

% Define Swaption parameters
timeToExpiry = 10; 
swapMaturity = 15; 
k = 0.01; %Strike 
deltaT = 1/250; 
numberExercises = 10; 
tau = 1; 

% Define simulation parameters
paths = 10000; 
alpha = 0.0563;
\end{verbatim}
\[ \sigma_r = 0.029; \]

\%Generate paths of transformed process \( x \)

\[
[\text{shortRateX}, \text{discountFactorX}] = \text{GenerateShortRate}(\alpha, \sigma_r, \\
\text{timeToExpiry}, \text{paths});
\]

res = 0;

timeBetweenExerciseOpportunities = timeToExpiry/deltaT / numberExercises;

maxdays = timeToExpiry/deltaT;

payoffs = zeros(numberExercises, paths);

swapRate = zeros(numberExercises, paths);

liborRate = zeros(numberExercises, paths);

discountFactorX_t = zeros(numberExercises, paths);

exerciseIndices = zeros(numberExercises, paths);

for i = maxdays:-timeBetweenExerciseOpportunities:

    for j = 1:paths

        discountFactorX_t(i*deltaT, j) = discountFactorX(i,j) - \\
        discountFactorX(i-1/deltaT+1,j) + 1;

        swapRate(i*deltaT, j) = calcSwapRate(shortRateX(i,j), i*deltaT, \\
        swapMaturity, tau, alpha, sigma_r, intCurve);

        liborRate(i*deltaT, j) = calcLiborRate(shortRateX(i,j), i*deltaT \\
        , swapMaturity, tau, alpha, sigma_r, intCurve);

        payoffs(i*deltaT, j) = max(swapRate(i*deltaT, j) - k, 0);

    end

% For final payoff time, no regression will be performed
if i == maxdays

    continue

end

% CashFlows on which the regression will be performed

cashFlows = payoffs(i*deltaT+1,:).*discountFactorX_t(i*deltaT+1,:);

itmIndices = find(payoffs(i*deltaT,:));

swapRates = swapRate(i*deltaT,:).';

liborRates = liborRate(i*deltaT,:).';

% Initialize matrix for \( y \) values
x = ones(size(itmIndices,:) ,1) ,6);

x(:,1:6) = [ones(size(itmIndices,:), 1), 1), swapRates(itmIndices),
            liborRates(itmIndices), ... swapRates(itmIndices).^2, liborRates(itmIndices).^2, swapRates(itmIndices).*liborRates(itmIndices)];

% Only consider the cashflows that are in-the-money
y = cashFlows(itmIndices,:);

% Calculate regression coefficients
coefficients = x \ y;

% Calculate continuation and exercise values
continuationValue = x * coefficients;
exerciseValue = max(swapRates(itmIndices,1) - k,[]) ,2);

exerciseNow = (exerciseValue > continuationValue) .* exerciseValue;
exerciseIndices = find(exerciseNow);
continuationIndices = find(~exerciseNow);

replaceIndices = itmIndices(exerciseIndices);
zeroIndices = itmIndices(continuationIndices);

payoffs(i*deltaT,replaceIndices) = exerciseNow(exerciseIndices);
%Set payoffs for paths which are exercised for times after to 0
payoffs((i*deltaT+1):end,replaceIndices) = 0;
payoffs(i*deltaT,zeroIndices) = 0;
end

% Calculate swaption payoff
exercisePath = find(payoffs);
swaptionValue = marketZeroBondPrice(intCurve , timeToExpiry) * sum(payoffs(exercisePath) .* discountFactorX(exercisePath/deltaT))/paths*100;

B.4 Auxiliary functions

function cum = N(x)
cum = 0.5*(1+erf(x/sqrt(2)))
end

% Calculates the par swap rate of a swap with starting time t and
maturity
% T using bond reconstitution formula for process dx(t)

function swaprate = calcSwapRate(shortRate, t, T, tau, alpha, sigma_r, intCurve)
    denom = 0;
    for j = 1:(T - t)
        denom = denom + tau * xBondPrice(t, t+j, shortRate, alpha, sigma_r, intCurve);
    end
    swaprate = (1 - xBondPrice(t, T, shortRate, alpha, sigma_r, intCurve)) / denom;
end

% Computes the spot Libor rate at time t until the next period T = t +
% deltaT using bond reconstitution formula for process dx(t)
function liborrate = calcLiborRate(shortRate, t, T, tau, alpha, sigma_r, intCurve)
    liborrate = (1/xBondPrice(t, T, shortRate, alpha, sigma_r, intCurve) - 1)/tau;
end

% Computes the bond price under the process x(t)
function bondPrice = xBondPrice(t, T, xt, alpha, sigma_r, intCurve)
    bondPriceT = marketZeroBondPrice(intCurve, T);
    bondPricet = marketZeroBondPrice(intCurve, t);
    gtT = 1/alpha * (1 - exp(-alpha*(T-t)));
    yt = sigma_r^2/(2*alpha)*(1-exp(-2*alpha*t));
    bondPrice = bondPriceT/bondPricet * exp(-xt*gtT - 0.5*yt*gtT^2);
end

% Computes zero bond price from current yield curve observed in the
% market
function bondPrice = marketZeroBondPrice(intCurve, timeToMaturity)

% Requires a 2xm-Matrix for the current observed term structure
intCurveMaturities = intCurve(:,1);
intCurveRates = intCurve(:,2);
if (timeToMaturity < intCurveMaturities(1))
    interpolatedRate = intCurveRates(1);
else if (timeToMaturity > intCurveMaturities(end))
    interpolatedRate = intCurveRates(end);
else
    interpolatedRate = interp1(intCurveMaturities, intCurveRates, timeToMaturity, 'linear');
end

bondPrice = exp(-interpolatedRate/100*timeToMaturity);
References


