JAMSHIDIAN SWAPTION FORMULA FINE TUNED

PETER CASPERS

First Version March 28, 2013 - This Version May 4, 2013

ABSTRACT. The Jamshidian swaption formula a.k.a. the Jamshidian trick reduces the pricing of an european swaption to the pricing of a series of zerbond options. This works in a one factor interest rate model in which zerobond prices are monotonic in the state variable. We review the method and write it down taking into account the start delay of the underlying swap which is usually ignored in the literature.

1. INTRODUCTION

Consider an (physical settled) european swaption with expiry t_e giving the holder the right to enter into a swap starting at $t_s \geq t_e$. Usually $t_s - t_e$ is positive, e.g. 2 business days. We want to price this swaption in an interest rate model driven by one factor, e.g. the Hull White 1 factor model. The method is however not restricted to this particular model. It rather works for every model in which zerobond prices satisfy

$$P(t,t',T,x(t)) \leq P(t,t',T,x'(t)) \text{ whenever } x'(t) \geq x(t) \tag{1.1}$$

where we write x for the driving state variable and $P(t,t',T)$ for the $t-$ price of a forward zerobond starting at t' and maturing at T . We shall assume such a model for the rest of this paper. We note that 1.1 holds for every affine one factor model with monotonically increasing $A(t,\cdot)$ function, since $P(t,T,x) = \exp(-A(t,T)x + B(t,T))$ here and hence

$$P(t,t',T,x(t)) = e^{(A(t,t')-A(t,T))x(t)+B(t,T)-B(t,t')} \tag{1.2}$$

In particular for the Hull White model it is immediately verified, that $A(t,\cdot)$ is increasing.

In section 2 we review the method allowing explicitly for a start delay of the underlying swap, which is usually ignored in the literature. Section 3 gives numerical examples using the open source library QuantLib [1].

2. START DELAY

With notations as above at time t_e the value of a payer swaption is

$$P(t_e,t_e,t_s,x(t_e))\left(1-\sum_{i=1}^{n} c_i P(t_e,t_s,t_i,x(t_e))\right)^+ \tag{2.1}$$

Date: May 4, 2013.

where the c_i represent the fixed coupons with c_n including a final capital payment. The t_s-forward price of the float leg observed at t_e is 1. Here we assume a monocurve setup as well as some simplifications on the estimation of the float coupons.

The first step in Jamshidians method is to find a value x^* for which

$$1 - \sum_i c_i P(t_e, t_s, t_i, x^*) = 0 \tag{2.2}$$

which is usually done by a one dimensional numerical zero search. Now set

$$K_i := P(t_e, t_s, t_i, x^*) \tag{2.3}$$

which allows to rewrite the swaption payoff 2.1 as

$$P(t_e, t_e, t_s, x(t_e)) \left(\sum_i c_i K_i - \sum_i c_i P(t_e, t_s, t_i, x(t_e)) \right)^+ \tag{2.4}$$

Since we are working under the general assumption of $P(t, T, \cdot)$ being monotonically decreasing this expression is non negative if and only if $x \geq x^*$. This is also true for each summand $K_i - c_i P(t_e, t_s, t_i, x)$ for the same reason and therefore the payoff can be written

$$P(t_e, t_e, t_s, x(t_e)) \sum_i c_i \left(K_i - P(t_e, t_s, t_i, x(t_e)) \right)^+ \tag{2.5}$$

This shows that the swaption payoff at time t_e is the same as the sum of zerobond option payoffs at time t_e with strikes as defined in 2.3 and call and put reversed in comparision to the swaption.

The zerobond option we are looking at here is in more detail an option with expiry t_e, on which the t_s-forward zerobond price is marked against the strike K_i and the difference, if positive, is paid out on t_s.

What remains to do is to extend the usual zerobond option pricing formula by a pricing formula allowing for an expiry date different from the start date of the underlying bond. This is however easy to do. For example we can refer to [2] and note that Proposition 4.5.1 can be generalized to our situation by choosing the t_s-forward measure for pricing, replacing $P(T, T^*)$ by $P(t_e, t_s, T)$ in our notation, and setting the upper integral bound of v to t_e. More formal:

Proposition 1. *(slight generalization of Propostion 4.5.1 in* [2]*) Consider a European call option with expiry t_e and paying at t_s the amount*

$$(P(t_e, t_s, T) - K)^+ \tag{2.6}$$

In a Gaussian HJM model we have

$$V(t) = P(t, T)\Phi(d_+) - P(t, t_s)K\Phi(d_-) \tag{2.7}$$

with

$$d_{\pm} = \frac{\ln(P(t,T))/(KP(t,t_s)) \pm v/2}{\sqrt{v}} \tag{2.8}$$

$$v = \int_t^{t_e} |\sigma_P(u,T) - \sigma_P(u,t_s)|2du \tag{2.9}$$

where $\sigma_P(\cdot,t)$ is the volatility of the zero coupon bond process $P(\cdot,t)$.

Proof. In the t_s - forward measure Q^{t_s} we have

$$V(t) = P(t,t_s)E_t^{t_s}\left((P(t_s,t_s,T) - K)^+\right) \tag{2.10}$$

Since $P(\cdot,t_s,T)$ is a Q^{t_s} - martingale with dynamics

$$dP(t,t_s,T)/P(t,t_s,T) = -(\sigma_P(t,t_s) - \sigma_P(t,T))dW(t) \tag{2.11}$$

we can apply the usual Black formula to get $V(t)$. □

3. Numerical Examples

We give two numerical examples in this section. The computations have been carried out using the QuantLib library [1]. The pricing date is 27-03-2013 and the swaption is a standard payer swaption on a 10y swap versus Euribor 6m with expiry 27-03-2018 and value date 29-03-2018. The interest rate curve is assumed to be flat forward at 3% (continously compounded rate). We are using a Hull White model with 1% reversion and 1% model volatility. Table 1 shows pricing results with different engines: The original Jamshidian engine ignores start delays and assumes the start date to be equal to the expiry date. The new Jamshidian engine takes into account the start delay as described in section 2. The last two engines are numerical engines respecting the start delay too. While the new Jamshidian engine and the two numerical engines agree up to 0.06 bp, the original engine deviates by 0.24bp.

Table 1. Pricing example with 2 open days start delay

Pricing Engine	NPV
Original Jamshidian Engine	0.062566446
New Jamshidian Engine	0.062590737
Tree Swaption Engine	0.062596428
Proprietary Integral Engine	0.062588027

The picture becomes even clearer when doing the same pricing with an (artificial) settlement delay of 42 open days instead of 2, leaving all other parameters as before. Table 2 shows the results for this case, now the original engine being off by over 7bp while the other three engines still agree up to 0.08bp.

References

[1] QuantLib A free/open-source library for quantitative finance, http://www.quantlib.org
[2] Piterbarg, Andersen: Interest Rate Modeling, Volume I, Atlantic Financial Press, 2010

E-mail address, Peter Caspers: pcaspers1973@gmail.com

TABLE 2. Pricing example with 42 open days start delay

Pricing Engine	NPV
Original Jamshidian Engine	0.061439389
New Jamshidian Engine	0.062174228
Tree Swaption Engine	0.062182501
Proprietary Integral Engine	0.062168930