INTEREST RATES MODELS

Homework assignment #4

Andrew Lesniewski
Baruch College
New York

March 13, 2017

Problems

The purpose of the first four problems is to implement the standard SABR model. **Note**: In all option models below assume the discount factors to be 1.

1. Implement the normal and lognormal option models (in case you have not done it yet...).

2. Use Excel's Solver (or write your own Newton-Raphson type routine in Python) in order to convert the lognormal volatilities to normal volatilities.

3. Implement the SABR implied normal volatility function. This function should take as arguments the model parameters (α, β, ρ), the relevant market parameters (σ, F) and the contract terms (option maturity and strike). Remember to use the result of the calculation in Problem 3 of Homework Assignment #3 for strikes near the money.

4. Using the enclosed data sheet and Excel's Solver utility (or any other optimization routine, if Excel is below you), calibrate the parameters of the SABR model. Fix the beta parameter to be $\beta = 0.5$. Keep on your mind, Solver is a generic tool and may sometimes be finicky. I suggest starting the searches with the following initial guess: $\sigma_0 = 0.1$, $\alpha = 0.3$, $\rho = 0.0$.

The purpose of the next three problems is to compare the accuracy of the SABR formula for implied volatility. Specifically, we will run a large number of Monte Carlo simulations of the exact SABR model, and compare the results of pricing a European option to the value obtained by applying the asymptotic SABR formula. As usual, we denote the SABR parameters by $\sigma, \alpha, \beta, \rho$.

5. Use the Python function `numpy.random.normal` to generate uncorrelated Gaussian random numbers. For the purposes of this assignment, you will need to generate a sequence of two dimensional Gaussian random numbers (x, y), $(x, y \sim N(0, 1))$ whose correlation coefficient is $\mathsf{Corr}(x, y) = \rho$. Explain how to do it, and verify your algorithm.

6. Write code for generating MC paths for the SABR model. I suggest the following approach. Instead of using the naive Euler scheme for both equations, integrate first the equation for σ, $\sigma(t) = \sigma_0\, e^{\alpha Z(t) - \alpha^2 t/2}$. Then discretize:

$$F_{i+1} = \max\left(F_i + \sigma_i F_i^{\beta}\, \sqrt{\delta t}\, x_i,\ 0\right),$$
$$\sigma_{i+1} = \sigma_i e^{\alpha\sqrt{\delta t}\, y_i - \alpha^2\, \delta t/2},$$

   where δt is the time step, and (x_i, y_i), $i = 1, \ldots, numSteps$, are the Gaussian random vectors described in Problem 5. Keep on your mind that the "long jump" method does not work for SABR, and you have to sample at intermediate time steps in order to assure accuracy of the procedure. In fact, in order to assure that the correlation coefficient between dW and dZ is respected by the Monte Carlo simulation, δt ought to be fairly small. Note the presence of $\max(\ \cdot\ , 0)$ in the algorithm above, which implements the Dirichlet boundary condition.

7. Use (at least) 5000 MC paths to value each of the options in Table 1. Note: A call or put option refers here to the call on the rate or put on the rate, respectively (and not on the market). Compare the results to those obtained from Black's formula (lognormal or normal), with the appropriate SABR implied volatility used for the volatility argument. Remember, we assume that all discount factors are equal to 1.

This assignment is due on March 27

	T	F_0	K	σ_0	α	β	ρ
call	0.25	0.005	0.01	0.0072	1.1	0.2	0.8
call	0.25	0.05	0.06	0.0224	0.8	0.5	-0.2
call	0.5	0.01	0.02	0.0400	0.8	0.5	0.7
call	0.5	0.05	0.06	0.0182	0.7	0.2	-0.3
call	1	0.015	0.02	0.0232	0.6	0.2	0.4
call	1	0.04	0.05	0.0500	0.5	0.5	-0.3
call	2	0.02	0.03	0.0707	0.5	0.5	0.3
call	2	0.06	0.07	0.0176	0.5	0.2	-0.1
call	5	0.05	0.06	0.0146	0.4	0.2	0.2
call	5	0.05	0.07	0.0447	0.3	0.5	-0.5
put	0.25	0.005	0.002	0.0354	0.9	0.5	0.8
put	0.25	0.04	0.02	0.0095	0.9	0.2	-0.0
put	0.5	0.01	0.005	0.0400	0.7	0.5	0.4
put	0.5	0.03	0.02	0.0577	0.6	0.5	-0.2
put	1	0.03	0.01	0.0202	0.6	0.2	0.5
put	1	0.05	0.04	0.0182	0.6	0.2	-0.5
put	2	0.04	0.025	0.0500	0.5	0.5	0.2
put	2	0.05	0.035	0.0447	0.5	0.5	-0.1
put	5	0.05	0.035	0.0146	0.4	0.2	0.1
put	5	0.06	0.04	0.0176	0.4	0.2	0.0

Table 1