Diffusion Model
Final Project: Calibration of the Heston Model

Tommasini Nicola

February 3, 2021

Abstract

We want to calibrate the Heston model using different optimization techniques. The data was taken from Bloomberg and they are 35 prices (5 maturities and 7 strikes) of the Facebook (FB) call option. Three algorithms of calibration are proposed: Genetic Algorithm (GA), Levenberg-Marquardt (LM), and Surrogate optimization (SO).
The project has been developed in MATLAB R2019.

1 Heston Model

The Heston Model is defined by the following stochastic processes:

$$dS_t = rS_t dt + \sqrt{v_t} S_t dW^{(s)} \tag{1}$$

$$dv_t = \kappa(\theta - v_t)dt + \sigma\sqrt{v_t} dW_t^{(v)} \tag{2}$$

$$dW^{(s)} dW_t^{(v)} = \rho \tag{3}$$

where the volatility follows the so called CIR process (Cox–Ingersoll–Ross). The two-process follows different but correlated Brownian motion, with a constant correlation $\rho \in [-1, 1]$.
For the CIR process, we need to add a condition in order to obtain a strictly positive process, the Feller condition. It's expressed by the following relation:

$$2\kappa\theta \geq \sigma^2 \tag{4}$$

Except correlation, all the others parameter involved are strictly positive and they are:

- v_0: is the initial variance
- θ: is the long run mean level of v_t
- κ: is the mean reversion rate
- σ: is the volatility of the variance (also known as vol of vol)

1.1 Pricing

Heston derived a closed form solution for the price of an European call option on asset with stochastic volatility.
From the Garman's partial differential equation we have:

$$\frac{\partial C}{\partial t} + \frac{S^2 v}{2}\frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC + [\kappa(\theta - v) - \lambda v]\frac{\partial C}{\partial v} + \frac{\sigma^2 v}{\partial v^2} + \rho\sigma SV\frac{\partial^2 C}{\partial S \partial v} = 0 \tag{5}$$

where λ denotes the market price of volatility.
Equipped with the previous equation and applying an analogy with the B&S pricing formula one can obtain:

$$C(S_t, v_t, t, T) = S_t P_1 - Ke^{-r(T-t)}P2 \tag{6}$$

where P_1 and P_2 are defined by the the inverse Fourier transformation as follow:

$$P_j(x, v_t, T, K) = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re(\frac{e^{-i\phi ln(K)} f_j(x, v_t, T, \phi)}{i\phi})d\phi, \quad j = 1, 2 \tag{7}$$

and where:

$$x = ln(S_t) \tag{8}$$

$$f_j(x, v_t, T, \phi) = e^{C(T-t,\phi)+D(T-t,\phi)v_t+i\phi x} \tag{9}$$

$$C(T-t,\phi) = r\phi i\tau + \frac{a}{\sigma^2}[(b_j - \rho\sigma\phi + d)\tau - 2ln(\frac{1-ge^{d\tau}}{1-g})] \tag{10}$$

$$D(T-t,\phi) = \frac{b_j - \rho\sigma\phi i + d}{\sigma^2}(\frac{1-e^{dr}}{1-ge^{d\tau}}) \tag{11}$$

$$g = \frac{b_j - \rho\sigma\phi i + d}{b_j - \rho\sigma\phi i - d} \tag{12}$$

$$d = \sqrt{(\rho\sigma\phi - b_j)^2 - \sigma^2(2u_j\phi i - \phi^2)} \tag{13}$$

for $j = 1, 2$ and defining:

$$u_1 = \frac{1}{2}, \quad u_2 = -\frac{1}{2}, a = \kappa\theta, \quad b_1 = \kappa + \lambda - \rho\sigma, \quad b_2 = \kappa + \lambda \tag{14}$$

This is the classic method for deriving the characteristic function (Heston 1993). Other methods have been developed such as the Carr-Madan approach (1999) and the little Heston trap. The latter is proposed as it's the one that MATLAB used in the pricing algorithm.
The little Heston trap is simply defined as follow:

$$C(T-t,\phi) = r\phi i\tau + \frac{a}{\sigma^2}[(b_j - \rho\sigma\phi + d)\tau - 2ln(\frac{1-\varepsilon_j e^{-d\tau}}{1-\varepsilon_j})] \tag{15}$$

$$D(T-t,\phi) = \frac{b_j - \rho\sigma\phi i + d}{\sigma^2}(\frac{1-e^{-d\tau}}{1-\varepsilon e^{-d\tau}}) \tag{16}$$

$$\varepsilon = \frac{b_j - \rho\sigma\phi i - d}{b_j - \rho\sigma\phi i + d} \tag{17}$$

The importance to have a close form of the Heston model relies on the fact that it speed up the calibration process of vanilla option. The characteristic function allow us to use the Fast Fourier Transform (FFT) for the option pricing. Other methods like numerical integration and Monte Carlo simulation are not treated in this project.

2 Calibration

The model calibration consists to find out a set of parameter that allow our model to replicate the market behavior. To achieve this result there are several function that can be consider as objective function such as the average relative percentage error (ARPE),the relative percentage error (RPE),root mean square error (RMSE) and the mean square error (MSE).
For the GA and SO the MSE has been used, instead for the LM the RMSE. They are defined as follow:

$$RMSE = \sqrt{[\text{Market Price} - C_{Heston}(r, S_0, T, K, v_0, \theta, \kappa, \sigma, \rho)]^2} \tag{18}$$

$$MSE = \sum_{i=1}^{N} \frac{[\text{Market Price} - C_{Heston}(r, S_0, T, K, v_0, \theta, \kappa, \sigma, \rho)]^2}{N} \tag{19}$$

where N is the number of prices and the parameters that we want to calibrate are $v_0, \theta, \kappa, \sigma, \rho$.
To solve the calibration problem multiple techniques and approach are proposed:

- Genetic Algorithm (GA) + *fmincon*: is an heuristic random-based classical evolutionary algorithm for the research of a global minima.
- Levenberg–Marquardt algorithm (LM): uses a nonlinear, least-squares-fit algorithm. This algorithm combines the steepest descent and Gauss-Newton methods. If the model (in this case our objective function) is well-behaved then the local minima won't be a problem.
- Surrogate optimization (SO) + *fmincon* : utilizes a surrogate model to approximate the expensive to evaluate objective function, thus greatly improved optimization efficiency.

where *fmincon* is a build in function in MATLAB used to find a minimum of constrained nonlinear multivariable function. The key idea is to use algorithm such as GA and SO to search the general location of the global minimum and then using *fmincon* to find out the exact position (Hybrid approach).

2.1 Initial data and parameters

For this project, we have retrieved the price of the Facebook call option from Bloomberg, shown in table 1, for 5 different maturities (from 19/03/2021 to 17/09/2021) and for 7 different strikes (from 265 to 295 with a step size of 5).
In table 2 there are the initial arbitrary parameters that have been used the compute the first price approach.
The other assumptions are the no presence of dividends ($q = 0$) and the risk-free rate equal to 0.

	Strike						
Maturity	265	270	275	280	285	290	295
19/03/'21	25.05	22.14	19.37	16.87	14.80	12.77	11.02
16/04/'21	28.10	25.30	22.62	20.27	18.00	15.92	14.05
21/05/'21	32.5	29.75	27.40	25.00	22.60	20.70	18.70
18/06/'21	34.75	32.02	29.40	27.20	24.85	22.85	20.90
17/09/'21	41.50	38.97	36.62	34.22	32.00	29.92	27.87

Table 1: Price FB

Initial Parameters							
r	q	S_0	v_0	θ	κ	σ	ρ
0	0	277	0.04	0.05	1	0.3	-0.5

Table 2: Initial arbitrary parameter and assumptions (for r and q)

In figure 1 there is the implied volatility of B&S model obtained from the market price (using the MATLAB function *blsimpv*). In figure 2 there are plotted both the market and the Heston one price. As we will see later, after calibration, the surfaces will have to match.

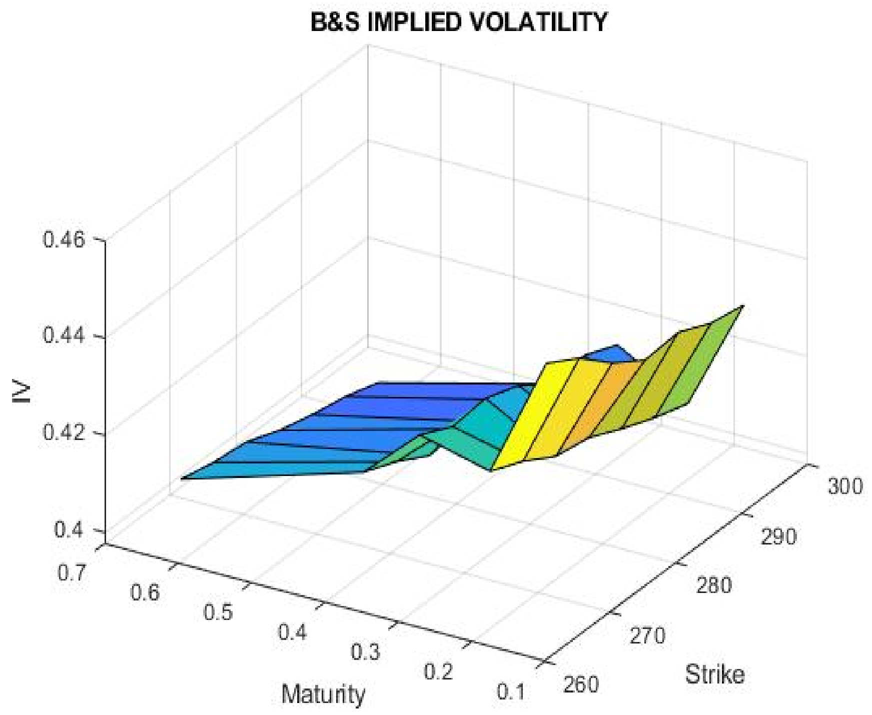

Figure 1: Implied Volatility

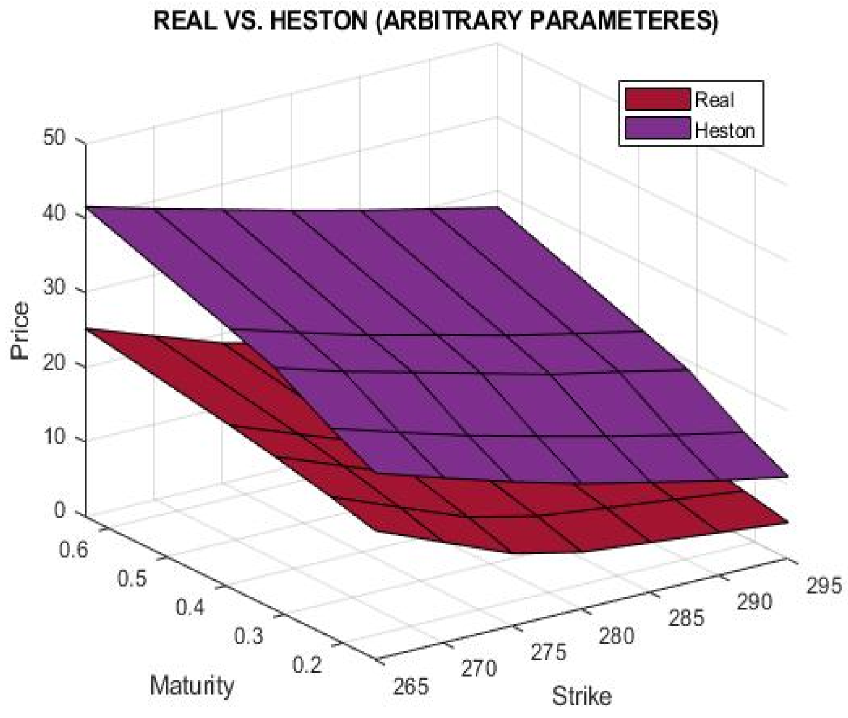

Figure 2: Real vs. Heston price (arbitrary parameters)

3 Results

3.1 Genetic Algorithm

The first technique that has been used is the GA + $fmincon$. For this algorithm we have impose:

- MaxGenerations: maximum number of iterations before the algorithm halts equal to 120
- PopulationSize: size of the population equal to 100.

In addition, shown in table 3, we add lower and upper bound for the parameters:

GA - Bounds					
	v_0	θ	κ	σ	ρ
lb	0.01	0.01	0.01	0.01	-1
ub	10	2	10	10	1

Table 3: Lower (lb) and upper (ub) bounds for GA

The table 4 shows the squared errors of the implicit volatilities, which are then represented in the figure 5 by a histogram. The magnitude of the error is of the right order, to consider the success of the calibration. In figure 4 real prices have been compared with calibrated ones, then in figure 3 shows the implied volatility obtained with the calibrated parameters and as we can see it does not show anomalous behavior.

GA - IV ERROR							
Date	265	270	275	280	285	290	295
19/03/'21	5.09E-06	3.59E-05	2.39E-04	1.01E-04	5.71E-07	6.74E-05	1.31E-04
16/04/'21	1.91E-04	5.81E-05	1.55E-07	5.27E-07	7.64E-05	1.86E-04	2.37E-04
21/05/'21	1.18E-09	1.53E-05	1.79E-04	1.60E-04	3.35E-05	4.23E-05	2.59E-05
18/06/'21	2.68E-05	3.08E-06	3.47E-06	1.86E-05	9.79E-08	1.39E-07	4.17E-08
17/09/'21	3.09E-05	6.85E-06	2.76E-06	1.25E-06	7.48E-08	1.39E-07	3.47E-07

Table 4: GA - squared error of implied volatility

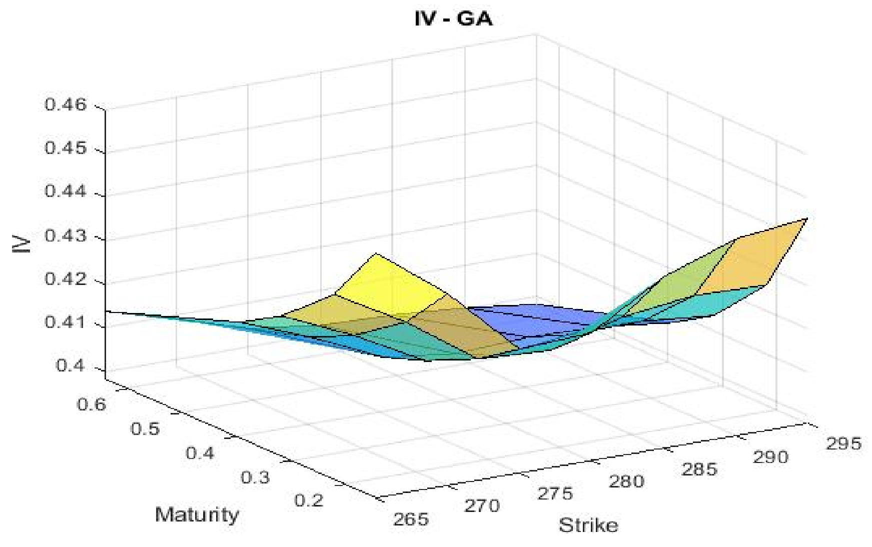

Figure 3: Implied Volatility - Calibration with GA algorithm

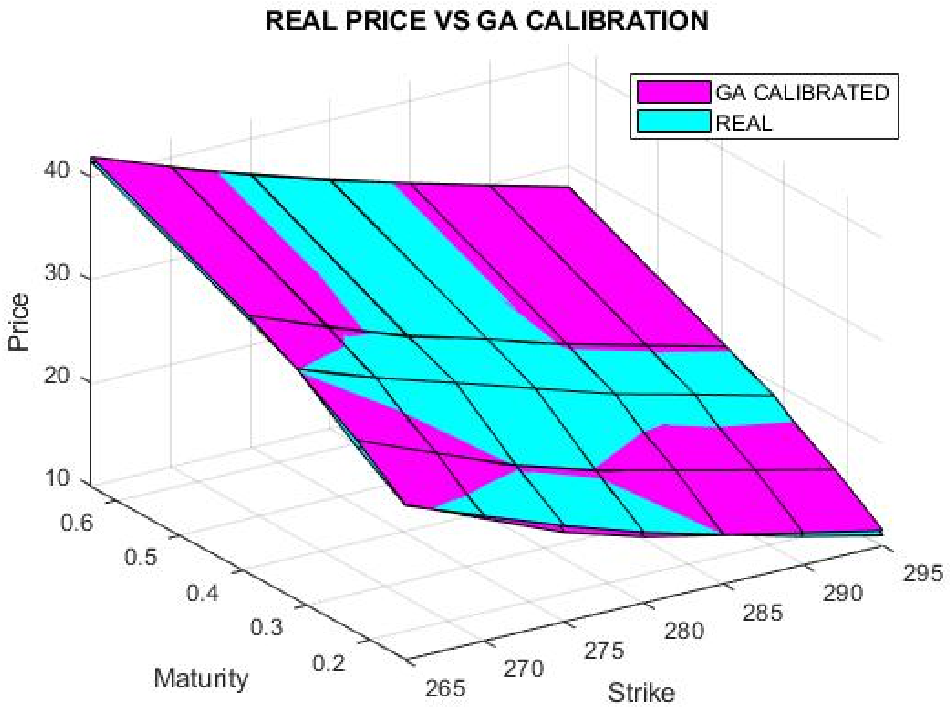

Figure 4: Comparison between Real and GA calibrated prices

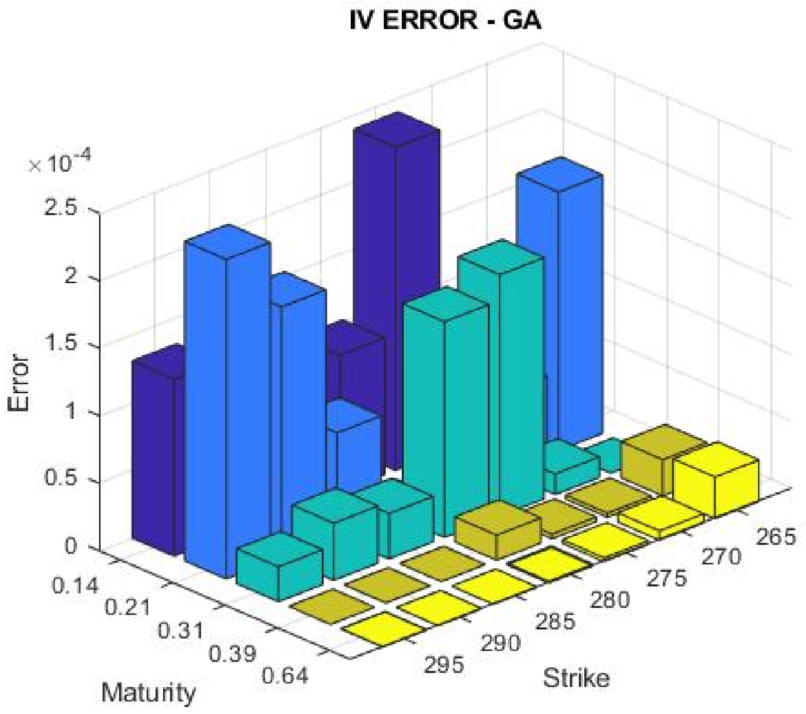

Figure 5: Implied volatility GA Error

3.2 Levenberg-Marquard

The implementation of this technique was faster since we have used the same bounds of the GA and we just set the *MaxFunctionEvaluation*= 1500.
The initial guess are the initial parameter in table 2.
The results acquire (table 5) are consistent with those previously obtained. The orders of magnitude match and we find again in this case, bigger errors for short maturities.
As before, in figure 6 and 5 are shown the implied volatility and the error.

LM - IV ERROR							
Date	265	270	275	280	285	290	295
19/03/'21	5.09E-06	3.59E-05	2.39E-04	1.01E-04	5.71E-07	6.74E-05	1.31E-04
16/04/'21	1.91E-04	5.81E-05	1.55E-07	5.27E-07	7.64E-05	1.86E-04	2.37E-04
21/05/'21	1.18E-09	1.53E-05	1.79E-04	1.60E-04	3.35E-05	4.23E-05	2.59E-05
18/06/'21	2.68E-05	3.08E-06	3.47E-06	1.86E-05	9.79E-08	1.39E-07	4.17E-08
17/09/'21	3.09E-05	6.85E-06	2.76E-06	1.25E-06	7.48E-08	1.39E-07	3.47E-07

Table 5: LM - squared error of implied volatility

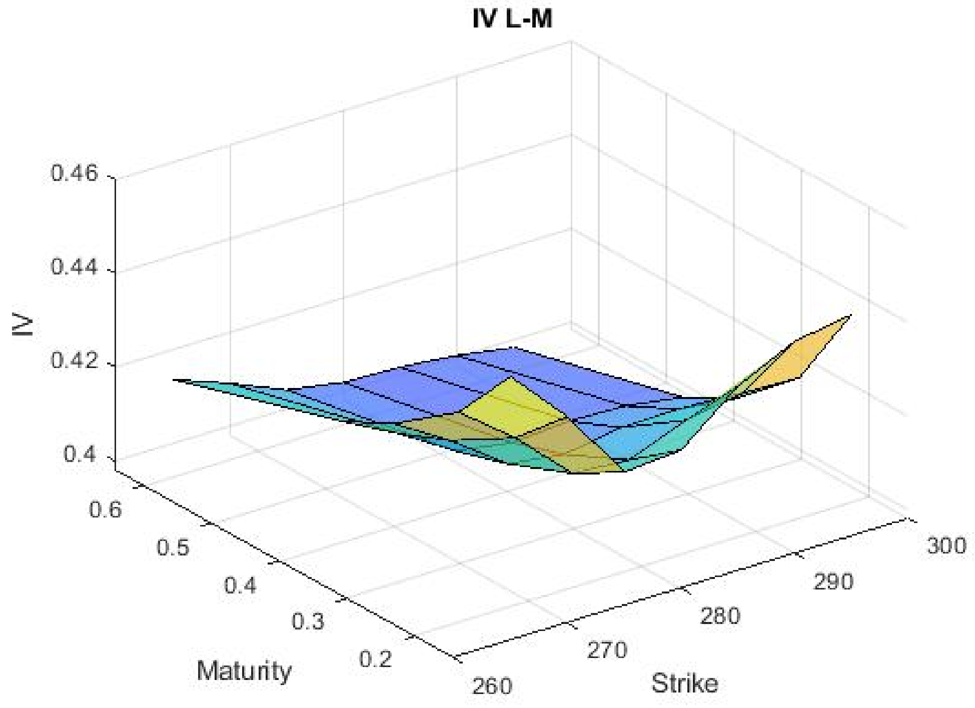

Figure 6: Implied Volatility - Calibration with L-M algorithm

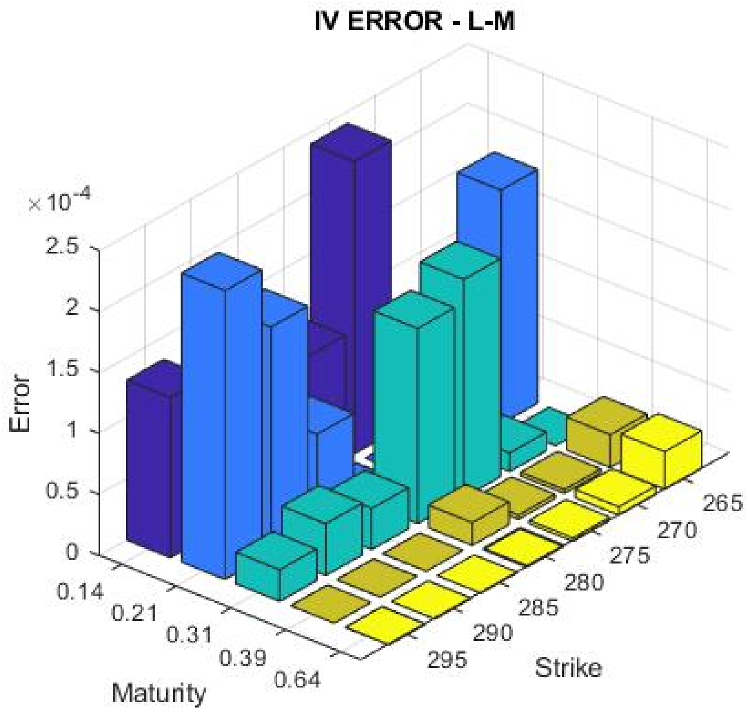

Figure 7: Implied volatility L-M Error

3.3 Surrogative Optimizer

The last method applied, again in the hybrid version with the help of the fmincon, was the Surrogative Optimizer. Also in this case the set up was quick, in fact in addition to the classic bounds we did not specify other parameters. The results obtained (shown in the following table (6) and figures (8 - 9)) are perfectly in line with the previous ones, demonstrating once again a correct calibration.

SO - IV ERROR							
Date	265	270	275	280	285	290	295
19/03/'21	5.09E-06	3.59E-05	2.39E-04	1.01E-04	5.71E-07	6.74E-05	1.31E-04
16/04/'21	1.91E-04	5.81E-05	1.55E-07	5.27E-07	7.64E-05	1.86E-04	2.37E-04
21/05/'21	1.18E-09	1.53E-05	1.79E-04	1.60E-04	3.35E-05	4.23E-05	2.59E-05
18/06/'21	2.68E-05	3.08E-06	3.47E-06	1.86E-05	9.79E-08	1.39E-07	4.17E-08
17/09/'21	3.09E-05	6.85E-06	2.76E-06	1.25E-06	7.48E-08	1.39E-07	3.47E-07

Table 6: LM - squared error of implied volatility

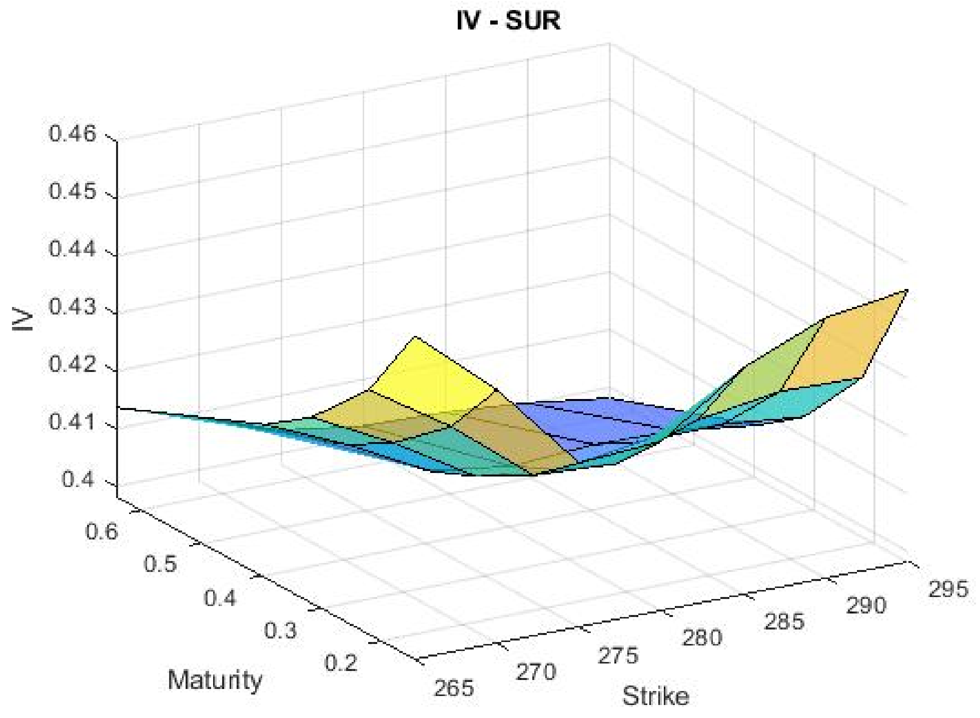

Figure 8: Implied Volatility - Calibration with SO algorithm

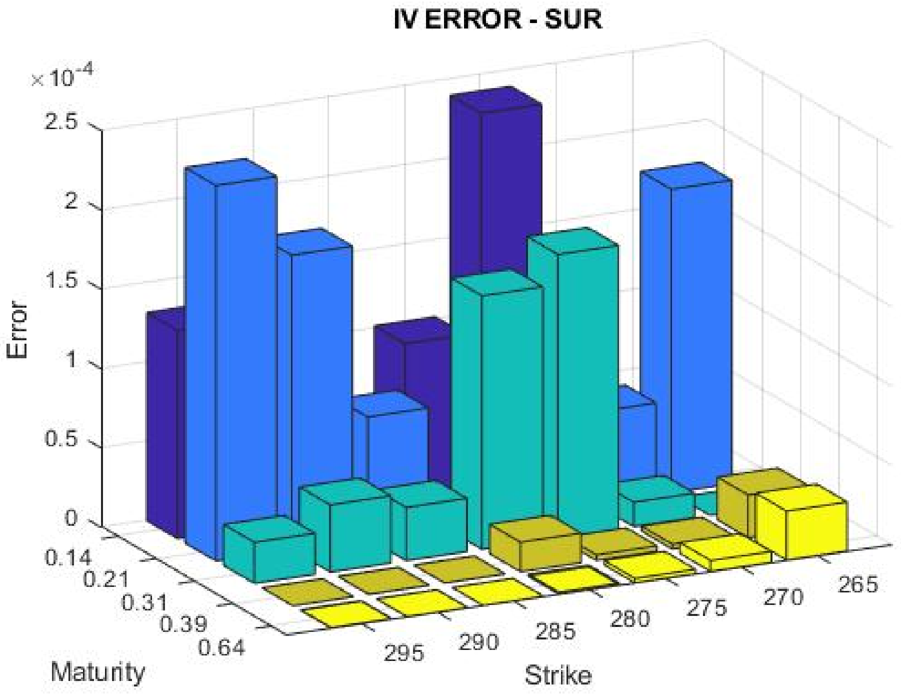

Figure 9: Implied volatility SO Error

3.4 Calibrated Parameters

The parameters obtained from the calibration using the various methods are shown in table 7 and as we can see the values are almost the same (they change from the 5th decimal digit).
Notice how the correlation is practically -1 (inversely correlated), but it had no effect on the calibration process.

Method	Calibrated Parameters				
	V_0	θ	κ	σ	ρ
GA	0.18514	0.16917	3.63999	0.57594	-0.99951
LM	0.18513	0.16917	3.63956	0.57562	-0.99999
SUR	0.18514	0.16917	3.63991	0.57587	-0.99961

Table 7: Calibrated parameters

Regarding the Feller condition, it was not added during the objective function minimization process but verified afterwards. In fact, by not imposing it a priori, we avoid adding another constrain to our algorithm, thus finding all possible solutions. The posteriori verification, if satisfied, will not require further precautions.
The Feller condition is satisfied for all methods:

$$2\kappa\theta \geq \sigma^2 \tag{20}$$

$$1.23 \geq 0.33 \tag{21}$$

4 Conclusion

From the above results, we can see how all three algorithms (both hybrid and non-hybrid) lead to the same results. The squared error of the volatility differences is of the order of magnitude hoped for this kind of problem. Also, the calibration of the parameters led to good results (practically the same results for each method).
It can be noted in table 8 that the MSE is also identical in all cases, however of a satisfactory result.
From table 9 we can see the runtime of each algorithm. While LM and SO + fmincon operate in the order of the minute (4 min for SO and 31 seconds for LM), the Genetic + fmincon turns out to be extremely expensive in terms of time (77 min).
A possible solution to overcome this problem could be a better calibration of the algorithm's hyperparameters and the use of its parallelized version, to optimize computational costs (with the help of the GPU). The other two algorithms implemented are by nature less expensive and do not require special attention.
Other improvements that can be made could be the use of the Carr-Madan version for the calculation of the characteristic function.

Method	MSE
GA	5.369E-05
LM	5.368E-05
SUR	5.369E-05

Table 8: MSE Comparison

Method	Runtime (s)
GA	4669
LM	31
SUR	257

Table 9: Runtime Comparison

5 Code

Matlab code without plots. In this code the output vectors from the various algorithms have been saved (to save time especially for the GA). There are also errors on prices and the automatic export of errors to excel (both price and volatility)

```
clear;
%IMPORT DATA
global Rate AssetPrice Maturity DividendYield OptSpec strike z real settle maturity

data = readtable('real_price_fb.xlsx');
M1 = data{:,1};
M2 = data{:,2};
M3 = data{:,3};
M4 = data{:,4};
M5 = data{:,5};
strike =(265:5:295);
real = transpose(data{:,:});
% figure(20)
% hold on;
% plot(strike, M1);
% plot(strike, M2);
% plot(strike, M3);
% plot(strike, M4);
% plot(strike, M5);
% title(' REAL CALL PRICE ');
% hold off
%IMPORT PARAMETER
AssetPrice = 277;
Rate = 0.0;
DividendYield = 0.00;
OptSpec = 'call';
V0 = 0.04;
ThetaV = 0.05;
Kappa = 1;
SigmaV = 0.3;
RhoSV = -0.5;

formatIn = 'dd/mm/yyyy';
settle = datenum( '26/01/2021' , formatIn );
mat1 = datenum('19/03/2021', formatIn );
mat2 = datenum('16/04/2021', formatIn );
mat3 = datenum('21/05/2021', formatIn );
mat4 = datenum('18/06/2021', formatIn );
mat5 = datenum('17/09/2021', formatIn );
maturity = [mat1,mat2,mat3,mat4,mat5];
Maturity = (maturity - settle)/365;
z=[V0,ThetaV,Kappa,SigmaV,RhoSV];
% implied vol
vol_r = zeros(5,7);
call_heston = zeros(5,7);
for i=1:7
for j=1:5
vol_r(j,i) = blsimpv(AssetPrice, strike(i),0.0,Maturity(j), real(j,i));
call_heston(j,i)=
    optByHestonFFT(Rate,AssetPrice,settle,maturity(j),OptSpec,strike(i),V0,ThetaV,Kappa,SigmaV,RhoSV);
end
end
Error = abs(real - call_heston);
t = table(real,call_heston,Error);
writetable(t,'error.xlsx');

%Genetic Algo
% tic
% rng default
% numberOfVariables = 5;
% lb = [ 0.01,0.01,0.01,0.01,-1];
% ub = [10,2,10,10,1];
% objfun =@(x)HestonObjFunction(x);
% opts = optimoptions(@ga,'PlotFcn',{@gaplotbestf,@gaplotstopping},'MaxGenerations',120);
% opts.PopulationSize = 100;
% [p_ga,Fval_ga, exitFlag_ga, Output_ga]= ga(objfun,
    numberOfVariables,[],[],[],[],lb,ub,[],opts);
% result = fmincon(objfun, p_ga, [],[],[],[],lb, ub,[]);
% toc
result =
    [0.185143210498775,0.169177294865401,3.63999916718472,0.575941612534588,-0.999516133374009];
price_cali = zeros(5,7);
vol_imp_cal = zeros(5,7);
for i = 1:7
for j=1:5
price_cali(j,i)=
    optByHestonFFT(Rate,AssetPrice,settle,maturity(j),OptSpec,strike(i),result(1),result(2),result(3),result
vol_imp_cal(j,i) = blsimpv(AssetPrice, strike(i),0.0,Maturity(j), price_cali(j,i));
end
end
ga_iv_error = abs(vol_r - vol_imp_cal).^2;
ga_error_price = abs(real - price_cali);
t_1 = table((ga_error_price));
t_2 = table((ga_iv_error));
writetable(t_1,'ga_price_error.xlsx');
writetable(t_2,'ga_iv_error.xlsx');
ga = abs(real - price_cali).^2;
MSE_ga = sum(ga_iv_error(:))/numel(vol_r);


%LEVENBERG-MANQUARD
tic
lb = [ 0.01,0.01,0.01,0.01,-1];
ub = [10,2,10,10,1];
x0=[0.04, 0.05,1,0.3,-0.5];
objfun =@(x)HestonObjFunction_1(x);
options =
    optimoptions(@lsqnonlin,'Algorithm','levenberg-marquardt','MaxFunctionEvaluation',1500);
x = lsqnonlin(objfun,x0,lb,ub,options);
toc
% x =
    [0.185139575307730,0.169171845735356,3.63956551167195,0.575622871107371,-0.999999999246454];
f1 = 2*x(3)*x(2);
f2 = x(4).^2;

price_lm = zeros(5,7);
vol_imp_lm = zeros(5,7);
for i = 1:7
for j=1:5
price_lm(j,i)=
    optByHestonFFT(Rate,AssetPrice,settle,maturity(j),OptSpec,strike(i),x(1),x(2),x(3),x(4),x(5));
vol_imp_lm(j,i) = blsimpv(AssetPrice, strike(i),0.0,Maturity(j), price_lm(j,i));
end
end
lm_iv_error = abs(vol_r - vol_imp_lm).^2;
lm_price_error = abs(real - price_lm);
t_3 = table((lm_price_error));
t_4 = table((lm_iv_error));
writetable(t_3,'lm_price_error.xlsx');
writetable(t_4,'lm_iv_error.xlsx');
lm = abs(real - price_lm).^2;
MSE_lm = sum(lm_iv_error(:))/numel(vol_r);
```

```
%Surrogate
rng default % for reproducibility
tic
lb = [ 0.01,0.01,0.01,0.01,-1];
ub = [10,2,10,10,1];

objfun =@(x)HestonObjFunction(x); % objective
opts = optimoptions('surrogateopt','PlotFcn',[]);
[xsur_1,fsur,flgsur,osur] = surrogateopt(objfun,lb,ub,opts);
xsur = fmincon(objfun, xsur_1, [],[],[],[],lb, ub,[]);
toc
% xsur =
    [0.185142479517688,0.169176216603031,3.63991484143612,0.575878029877144,-0.999612802682312];
price_sur = zeros(5,7);
vol_imp_sur = zeros(5,7);
for i = 1:7
for j=1:5
price_sur(j,i)=
    optByHestonFFT(Rate,AssetPrice,settle,maturity(j),OptSpec,strike(i),xsur(1),xsur(2),xsur(3),xsur(4),xsur
vol_imp_sur(j,i) = blsimpv(AssetPrice, strike(i),0.0,Maturity(j), price_sur(j,i));
end
end
sur_iv_error = abs(vol_r - vol_imp_sur).^2;
sur_price_error = abs(real - price_sur);
t_5 = table((sur_price_error));
t_6 = table((sur_iv_error));
writetable(t_5,'sur_price_error.xlsx');
writetable(t_6,'sur_iv_error.xlsx');
sur = abs(real - price_sur).^2;
MSE_sur = sum(sur_iv_error(:))/numel(vol_r);
function MSE = HestonObjFunction (p)
global Rate AssetPrice maturity OptSpec strike real settle
Price_tmp = zeros(5,7);
for i = 1:7
for j=1:5
Price_tmp(j,i) =
    optByHestonFFT(Rate,AssetPrice,settle,maturity(j),OptSpec,strike(i),p(1),p(2),p(3),p(4),p(5));
end
end
er = abs(real - Price_tmp).^2;
MSE = sum(er(:))/numel(real);
end
%For LM
function MSE = HestonObjFunction_1 (p)
global Rate AssetPrice maturity OptSpec strike real settle
Price_tmp = zeros(5,7);
for i = 1:7
for j=1:5
Price_tmp(j,i) =
    optByHestonFFT(Rate,AssetPrice,settle,maturity(j),OptSpec,strike(i),p(1),p(2),p(3),p(4),p(5));
end
end

%use this one for sqlnonlin
MSE = abs(real - Price_tmp);

end
```

6 Bibliography

[1] Heston, S. L. “A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options.” The Review of Financial Studies. Vol 6. No. 2. 1993.

[2] Albrecher, H., Mayer, P., Schoutens, W., and Tistaert, J. ”The Little Heston Trap.” Working Paper, Linz and Graz University of Technology, K.U. Leuven, ING Financial Markets, 2006.

[3] Chourdakis, K. “Option Pricing Using Fractional FFT.” Journal of Computational Finance. 2005.

[4] Goldberg, David E., Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley, 1989.

[5] Wang, Y., and C. A. Shoemaker. A General Stochastic Algorithm Framework for Minimizing Expensive Black Box Objective Functions Based on Surrogate Models and Sensitivity Analysis. arXiv:1410.6271v1 (2014). Available at https://arxiv.org/pdf/1410.6271.