Research on Heston Model with C#

WANG BOLONG –CMF
S1816623
S1816623@ed.ac.uk

Abstract — Heston model is common use in recent quantitative financial model. Therefore, in this experiment, we explore the different methods of pricing option, including Heston Model formula, Monte-Carlo Method. We also exercise the calibration in the practice. Finally, we explore other fields of quantitative finance based on Heston Model experimentally.

Keywords: Heston Model, Formula, Monte-Carlo Method

1. Introduction

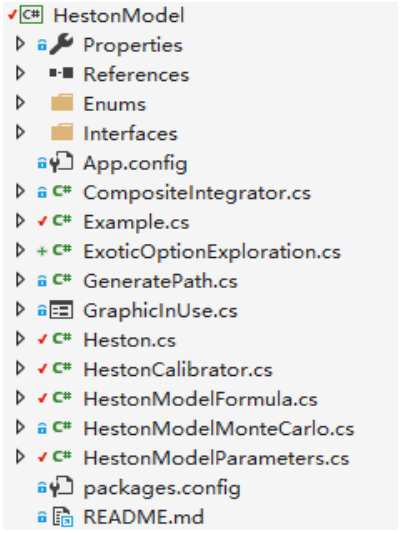
The classical Black-Scholes model assumes that the volatility is constant for the lifetime of the option. This assumption is contradicted by empirical evidence. In this situation, market need another model to price the option and Heston Model is necessary to research.

In this experiment, we establish the Heston model Formula, and design the Monte-Carlo method. We make it sure to work by practice experimentally. We also design the calibrator for calibration. After that, we explore it effect on exotic option, including Asian option, Lookback Option, Shout Option and Rainbow Option. All the experiments are documented and coded.

2. Model Design and Implement

2.1 Design (Task 2.1)

In this task, we design 10 classes file to implement the code HestonModelFormula.cs is use to implement the formula, HestonModelMonteCarlo.cs have MonteCarlo method and in this class, we can also calculate European option, Asian option, and Lookback option.HestonModelParameter.cs is used to initialize the interface in Heston.Interface file.



HestonCalibrator.cs is use to implement calibration. We also need a class to generate the price path, as GeneratePath.cs. ExoticOptionExploration.cs has the code to implement rainbow option and shout option. We write a Example. Class which is used to debug and implement task. As for CompositeIntegrator.cs in used to numerical integrate.

2.2 Heston Formula (Task 2.2)

In this part, we use HestonModelFormula.cs to achieve the task2.2, with parameters as following:

r=0.025	Kappa=1.5768	Rho=-0.5711	S = 100
V=0.175	Theta*=0.0398	Sigma=0.5751	

Then we implement the HestonModelFormula.cs to get the data as following table:

Strike K	Option exercise T	callPrice(0,S,v)
100	1	7.27
100	2	11.72
100	3	15.45

100	4	18.73
1001	15	42.99

The example code is shown as `TestHestonModelFormula` method in the Example.cs

2.3 Heston Call/Put with Monte Carlo (Task 2.3)

In this part, we use HestonModelMonteCarlo.cs to achieve the task2.2, with parameters as following

r=0.1	Kappa=2	Rho=0.5	S = 100
V=0.04	Theta*=0.06	Sigma=0.4	

Then we implement the HestonModelMonteCarlo.cs as well as GeneratePath.cs to get the data as following table:

Strike K	Option exercise T	Price(0,S,v)
100	1	13.8
100	2	21.9
100	3	29.5
100	4	36.9
1001	15	77.7

HestonModelMonteCarlo.cs is used to implement Monte-Carlo method and GeneratePath.cs is used to create a path for Price to simulate random walk.

The example code is shown as the method of `TestHestonModelMonteCarlo` in the Example.cs

2.4 Checking Heston Formula and Monte Carlo (Task 2.4)

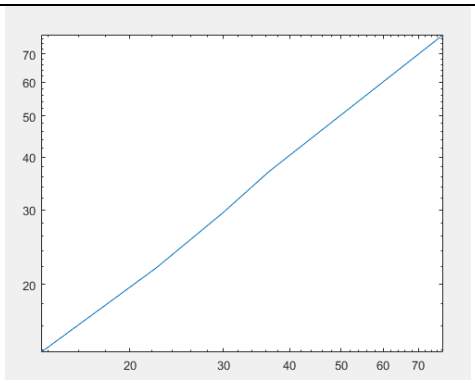
Implement two method at the same time.

2.4.1 Experiment1

We use the parameters as following:

r=0.1	Kappa=2	Rho=0.5	S = 100
V=0.04	Theta*=0.06	Sigma=0.4	

The Log-log scale plot:



The price we calculated is :

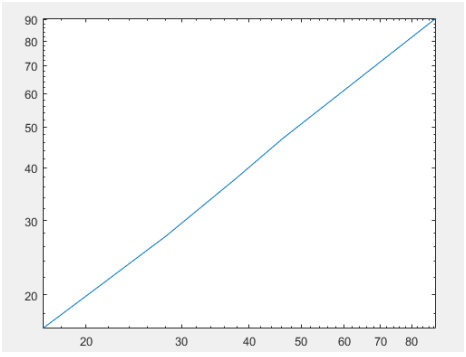
HestonModelFormula	HestonModelMonteCarlo
13.6108	13.8532
22.4081	21.8810
29.9214	29.5096
36.5487	36.9219
77.8725	77.6782

2.4.2 Experiment2

We use the parameters as following:

r=0.15	Kappa=2	Rho=0.56	S = 100
V=0.04	Theta*=0.06	Sigma=0.3	

The Log-log scale plot:



The price we calculated is :

HestonModelFormula	HestonModelMonteCarlo
16.62	16.6
28.12	27.6
37.87	37.8
46.26	47.1
88.69	90.7

Apparently, prices match between different methods

The reason is that we could assume the price calculated by formula is accurate, and in statistical perspective, with the number of sample increasing, the average converges to its real value. So if we use more samples in Monte-Carlo method, the result will be more accurate.

2.5 Calibration (Task 2.5)

SetGuessParameters

r=0.025	Kappa=1.5768	Rho=-0.5711	S = 100
V=0.175	Theta*=0.0398	Sigma=0.5751	

AdjustedParameters

r=0.025	Kappa=1.5787	Rho=-0.5703	S = 100
V=0.09	Theta*=0.1140	Sigma=0.5713	

Termination type: 1
Num iterations 5
Calibration outcome: FinishedOK
error: 0.15988

2.6 Checking Calibration (Task 2.6)

2.6.1
SetGuessParameters

r=0.1	Kappa=2	Rho=0.5	S = 100
V=0.04	Theta*=-0.06	Sigma=0.4	

Calibration outcome: `FailedOtherReason`
Termination type: -8
Num iterations 1
Failed reason:internal integrity control detected infinite or NAN values in function/gradient. Abnormal termination signalled.

2.6.2
SetGuessParameters

r=0.1	Kappa=2	Rho=0.5	S = 100
V=0.00	Theta*=0.06	Sigma=0.4	

AdjustedParameters

r=0.1	Kappa=2.0	Rho=0.51	S = 100
V=0.047	Theta*=0.02	Sigma=0.39	

Termination type: 1
Num iterations 5
Calibration outcome: FinishedOK
error: 20.8525594963294

2.7 Pricing Asian Arithmetic Option (Task 2.7)

We use the following parameters:

r=0.1	Kappa=2	Rho=0.5	S = 100
V=0.04	Theta*=0.06	Sigma=0.4	

And finally we got the MC call price

K	T	T1…TM	MC Price
100	1	0.75,1	11.9
100	2	0.25,0.50,0.75,1,1.25,1.5,1.75	11.5
100	3	1.0,2.0,3.0	19.7

2.8 Pricing Lookback Option (Task 2.8)

We use the following parameters:

r=0.1	Kappa=2	Rho=0.5	S = 100
V=0.04	Theta*=0.06	Sigma=0.4	

And finally we got the MC price

Option exercise T	MC Price
1	19.2
3	37.2
5	50.6
7	58.8
9	66.8

3. Exploration

3.1 Pricing of other exotic derivatives

.3.1.1 Pricing Shout Options
(The attached document is in the “Reference” file)

The shouting payoff is given by max(S(t) – S(T), 0), which can be expressed as max(S(T) – K, S(t) – K, 0) – (S(T) – K).;

We use the following parameter

r=0.1	Kappa=2	Rho=0.5	S = 100
V=0.04	Theta*=0.06	Sigma=0.4	

And then we get the price

Strike K	Option exercise T	Price(0,S,v)
100	1	14.5
100	2	19.5
100	3	22.5
100	4	24.0
1001	15	26.6

3.1.2 Pricing Rainbow Option

Rainbow options are usually calls or puts on the best or worst of n underlying assets. Thus, the payoffs at expiry for rainbow European options are:

Call on max:

Payoff = max(max(S1,S2,⋯,Sn)-K,0)

Put on max:

Payoff = max(K-max(S1,S2,⋯,Sn),0)

We use the following parameters:

r=0.1	Kappa=2	Rho=0.5	S = 100
V=0.04	Theta*=0.06	Sigma=0.4	

And then we get MC Price in the table.

K	T	S1,S2⋯SM	MC Price
100	1	100,100,100	30.1
100	2	100,100,100	48.5
100	3	100,100,100	63.0
100	4	100,100,100	78.1
100	15	100,100,100	163.3

The method implement successfully.

4. Reference

[1] OOPA course website

[2] Siska, D.A note on the Heston model

[3] C# for financial Market,

Appendix

1. HestonModel

Enums

Interfaces

CompositeIntegrator

Example

GeneratePath

Heston

HestonCalibrator

HestonModelFormula

HestonModelMonteCarlo

HestonModelParameters

2. HestonCmdLine

Program