GPU 平行運算與財務工程實作班

Heston 模型應用於結構商品之開發設計

昀騰金融科技

技術長

董夢雲 博士

dongmy@ms5.hinet.net

Part I Heston 模型與結構商品設計開發(15hrs)

一、Heston 模型介紹	案例一
二、蒙地卡羅模擬法	案例二
三、CPU 多線程的實作	案例三
四、結構商品的實例	案例四
五、結構商品的程式實作	案例五

Part II GPU 架構下的結構商品開發(15hrs)

六、GPU 與 CUDA 介紹	案例六
七、C#與 CUDA 的整合開發	案例七
八、CUDA 的變量與記憶體管理	案例八
九、CUDA 下的模擬與 cuRand 程式庫	案例九
十、GPU 版的結構商品模擬	案例十

昀騰金融科技股份有限公司

技術長
金融博士、證券分析師

董夢雲 Andy Dong

ID:50917111
Line/WeChat:andydong3137
E:andydong1209@gmail.com
https://github.com/andydong1209
M:(T)0988-065-751 (C)1508-919-2872
10647 台北市大安區辛亥路一段 50 號 4 樓

現職：國立台灣大學財務金融研究所兼任教授級專家
國立台灣科技大學財務金融研究所兼任助理教授
台灣金融研訓院 2021 年菁英講座

經歷：中國信託商業銀行交易室研發科主管
凱基證券風險管理部主管兼亞洲區風險管理主管
中華開發金控、工業銀行風險管理處處長
永豐金控、商業銀行風險管理處處長
永豐商業銀行結構商品開發部副總經理

學歷：國立台灣大學電機工程學系學士
國立中央大學財務管理學研究所博士
專業：證券暨投資分析人員合格(1996)

專長：風險管理理論與實務，資本配置與額度規劃、資產負債管理實務
外匯與利率結構商品評價實務，股權與債權及衍生商品評價實務
GPU 平行運算與結構商品系統開發，CUDA、OpenCL
CPU 平行運算與 ALM 系統開發，C#/C++/C、.Net Framework、SQL
人工智慧(Deep Learning)交易策略開發，Python、Keras、TensorFlow

Part I Heston 模型與結構商品設計開發

主題一 Heston 模型介紹

一、古典資產模型

二、市場匯率行為

三、Heston 模型與解析解

四、避險參數

五、實作案例一

一、古典資產模型

(一)Black-Scholes 對資產行為的假設

- Black-Scholes 模型之下股票價格變化的程序
  - 金融資產價格的假設是它遵行著所謂的擴散程序(diffusion process)

$$dS/S = \mu \cdot dt + \sigma \cdot dZ$$

  - ✓ $\frac{dS}{S} = \frac{S_{t+dt} - S_t}{S_t}$ ＝金融資產的報酬率，
  - ✓ dt ＝單位時間，
  - ✓ μ ＝單位時間內預期金融資產的報酬率，
  - ✓ σ ＝單位時間內預期金融資產的標準差。

- Z＝一隨機變數，為平均數為零，變異數為 t 之常態分配，$Z \sim \Phi(0,t)$ 。
  - Z 稱之為韋恩程序。
  - dZ ＝單位時間內，Z 的變動量，為一期望值為零，變異數為 dt 之常態分配，$dZ \sim \Phi(0,dt)$ 。

(二)解析解

◆ 以 Plain Vanilla 之歐式外幣選擇權買、賣權為例，定價公式如下

$$C = Se^{-yT}N(d_1) - Ke^{-rT}N(d_2) \quad \text{......(1.1)}$$

$$P = Ke^{-rT}N(-d_2) - Se^{-yT}N(-d_1) \quad \text{......(1.2)}$$

$$d_1 = \frac{\ln(S/K) + (r - y + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = \frac{\ln(S/K) + (r - y - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}$$

- N(x)表標準常態累積機率密度函數(CDF)在 x 的值。
- S = 即期匯率，K = 執行匯率，r = 本國貨幣資金成本，
- y = 外國貨幣持有收益，T = 到期日的時間，σ = 匯率之波動性。

二、市場匯率行為

(一)外匯市場報價資訊

- 外匯選擇權市場的流動性很高，即使長天期的契約亦是如此，下面資訊可由市場取得。
  - At-The-Money，ATM，的波動性，
  - 25∆Call 與 Put 的 Risk Reversal，RR，
  - 25∆Wings 的 Vega-Weighted Butterfly，VWB。
- 由上面資訊，我們可推導出三個基本的隱含波動性，
  - 使用這三個波動性，我們可建構出整個 Smile。
- 市場資訊可分別如下取得，
  - Currency Volatility Quote: Bloomberg: XOPT
  - 美元 LIBOR: RT: LIBOR01
  - NDF Swap Point: RT: TRADNDF

◆ Currency Volatility Quote: Bloomberg: XOPT

XOPT

<HELP> for explanation. P167c Curncy **OVDV**

Enter 1<GO> to Save

Currency Volatility Surface

Save | Send | Download | Options | 3D Graph | * Bloomberg (BGN) USDCNY

Currencies: USD-CNY Date: 5/ 7/08

USD Calls/Puts Deltas Format: 1 RR/BF

Calendar: 3 Weekends Side: 1 Bid/Ask

EXP	ATM(50D)		25D RR		25D BF		10D RR		10D BF	
	Bid	Ask	Bid	Ask	Bid	Ask	Bid	Ask	Bid	Ask
1W	2.050	4.155	-2.170	0.545	-0.930	1.175	-4.140	1.120	-0.625	1.475
2W	2.360	3.980	-1.845	0.210	-0.645	0.965	-3.475	0.430	-0.255	1.355
3W	2.570	3.970	-1.715	0.055	-0.525	0.870	-3.200	0.125	-0.100	1.295
1M	3.245	3.745	-1.150	-0.520	-0.070	0.425	-2.130	-0.985	0.365	0.865
2M	3.480	3.980	-1.215	-0.590	-0.050	0.445	-2.260	-1.115	0.440	0.940
3M	3.785	4.135	-1.160	-0.725	0.040	0.390	-2.135	-1.335	0.550	0.900
4M	4.060	4.470	-1.295	-0.785	0.015	0.420	-2.320	-1.395	0.525	0.935
6M	4.555	4.980	-1.465	-0.930	0.005	0.430	-2.455	-1.485	0.515	0.940
9M	4.940	5.320	-1.510	-1.035	0.055	0.435	-2.580	-1.720	0.595	0.970
1Y	5.420	5.720	-1.440	-1.060	0.110	0.410	-2.610	-1.930	0.665	0.965
18M	5.790	6.255	-1.580	-1.000	0.045	0.505	-2.810	-1.755	0.685	1.150
2Y	6.760	7.260	-1.770	-1.140	0.015	0.515	-3.025	-1.885	0.790	1.290
5Y	7.870	9.620	-2.825	-0.625	-0.565	1.180	-4.905	-0.885	0.430	2.175

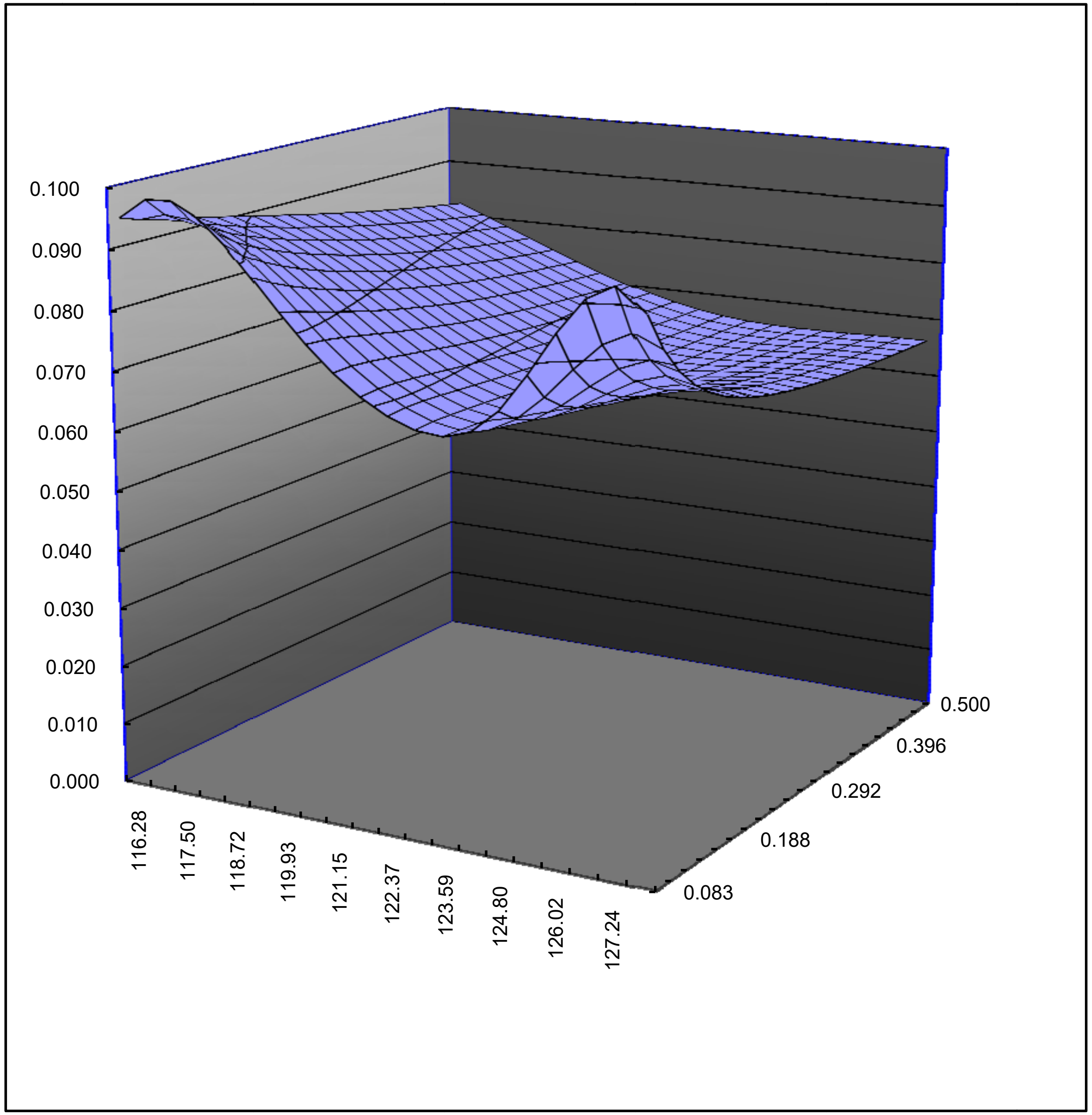
5.57

*Default RR = USD Call - USD Put

(二)Surface(USDJPY, 2007/7/11)

◆ 將不同時點的 Smile Curve 畫在同一立體圖上，形成一個曲面。

	0.083	0.104	0.125	0.146	0.167	0.188	0.208	0.229	0.250	0.271	0.292	0.313	0.333	0.354	0.375	0.396	0.417	0.438	0.458	0.479	0.500
116.28	0.095	0.094	0.094	0.093	0.092	0.091	0.091	0.090	0.089	0.089	0.088	0.087	0.087	0.086	0.085	0.085	0.084	0.084	0.083	0.083	0.082
116.89	0.099	0.096	0.094	0.092	0.091	0.090	0.089	0.089	0.088	0.087	0.086	0.085	0.085	0.084	0.083	0.083	0.082	0.082	0.081	0.081	0.080
117.50	0.099	0.095	0.093	0.091	0.090	0.089	0.088	0.087	0.086	0.085	0.084	0.084	0.083	0.082	0.082	0.081	0.081	0.080	0.080	0.079	0.079
118.11	0.097	0.093	0.091	0.089	0.088	0.087	0.086	0.085	0.084	0.083	0.082	0.082	0.081	0.080	0.080	0.079	0.079	0.078	0.078	0.077	0.077
118.72	0.093	0.090	0.088	0.087	0.085	0.084	0.083	0.082	0.082	0.081	0.080	0.079	0.079	0.078	0.078	0.077	0.077	0.076	0.076	0.076	0.075
119.32	0.088	0.086	0.085	0.084	0.083	0.082	0.081	0.080	0.079	0.078	0.078	0.077	0.077	0.076	0.076	0.075	0.075	0.074	0.074	0.074	0.073
119.93	0.083	0.082	0.081	0.080	0.079	0.079	0.078	0.077	0.076	0.076	0.075	0.075	0.074	0.074	0.074	0.073	0.073	0.073	0.072	0.072	0.072
120.54	0.078	0.078	0.078	0.077	0.076	0.076	0.075	0.074	0.074	0.073	0.073	0.073	0.072	0.072	0.072	0.071	0.071	0.071	0.071	0.070	0.070
121.15	0.074	0.075	0.074	0.074	0.073	0.073	0.072	0.072	0.072	0.071	0.071	0.071	0.070	0.070	0.070	0.070	0.070	0.069	0.069	0.069	0.069
121.76	0.071	0.071	0.071	0.071	0.071	0.070	0.070	0.070	0.070	0.069	0.069	0.069	0.069	0.069	0.068	0.068	0.068	0.068	0.068	0.068	0.068
122.37	0.069	0.069	0.069	0.069	0.069	0.068	0.068	0.068	0.068	0.068	0.068	0.068	0.067	0.067	0.067	0.067	0.067	0.067	0.067	0.067	0.067
122.98	0.067	0.067	0.067	0.067	0.067	0.067	0.067	0.067	0.067	0.067	0.067	0.067	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066
123.59	0.067	0.067	0.067	0.067	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066
124.20	0.068	0.067	0.067	0.067	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066
124.80	0.072	0.070	0.068	0.068	0.067	0.067	0.067	0.067	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066
125.41	0.078	0.074	0.071	0.070	0.069	0.068	0.068	0.067	0.067	0.067	0.067	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066
126.02	0.085	0.078	0.075	0.073	0.071	0.070	0.069	0.069	0.068	0.068	0.067	0.067	0.067	0.067	0.066	0.066	0.066	0.066	0.066	0.066	0.066
126.63	0.091	0.083	0.078	0.075	0.073	0.072	0.071	0.070	0.069	0.068	0.068	0.068	0.067	0.067	0.067	0.066	0.066	0.066	0.066	0.066	0.066
127.24	0.093	0.085	0.080	0.077	0.074	0.072	0.071	0.070	0.069	0.069	0.068	0.068	0.067	0.067	0.067	0.067	0.066	0.066	0.066	0.066	0.066
127.85	0.089	0.083	0.079	0.076	0.073	0.072	0.071	0.070	0.069	0.068	0.068	0.067	0.067	0.067	0.067	0.066	0.066	0.066	0.066	0.066	0.066



0.100
0.090
0.080
0.070
0.060
0.050
0.040
0.030
0.020
0.010
0.000
116.28
117.50
118.72
119.93
121.15
122.37
123.59
124.80
126.02
127.24
0.500
0.396
0.292
0.188
0.083

三、Heston 模型與解析解

(一)資產價格行為

◆ Steven Heston(1993)提出下面模型，

$$dS_t = \mu S_t dt + \sqrt{V_t} S_t dW_t^1 \quad \text{(3.1)}$$

$$dV_t = \kappa(\theta - V_t)dt + \sigma\sqrt{V_t} dW_t^2 \quad \text{(3.2)}$$

$$dW_t^1 dW_t^2 = \rho \cdot dt \quad \text{(3.3)}$$

- 其中 $\{S_t\}_{t\geq 0}$ 表價格過程，$\{V_t\}_{t\geq 0}$ 表波動性過程。
- 以 P 測度表示此真實世界下的機率測量。
- $\{W_t^1\}_{t\geq 0}$ 與 $\{W_t^2\}_{t\geq 0}$ 表真實世界中兩相關的布朗運動過程，相關係數為 ρ。
- $\{V_t\}_{t\geq 0}$ 為一平方根均數回覆過程，長期平均為 θ，回覆速率為 κ，σ 稱之為波動性之波動性。
- μ、ρ、θ、κ、σ 均為常數。

◆ 在 Q 測度下，(3.1)、(3.2)、(3.3)式成為，

$$dS_t = rS_t dt + \sqrt{V_t} S_t dZ_t^1 \quad \text{.......(3.4)}$$

$$dV_t = \kappa^* (\theta^* - V_t) dt + \sigma \sqrt{V_t} dZ_t^2 \quad \text{.......(3.5)}$$

$$dZ_t^1 dZ_t^2 = \rho \cdot dt \quad \text{.......(3.6)}$$

- 其中，$\kappa^* = \kappa + \lambda$，$\theta^* = \dfrac{\kappa\theta}{\kappa + \lambda}$。
- 由於我們所在意的為評價問題，因此所處理的測度為 Q 測度。
  - ✓ 後面的市場校準也是求得 Q 測度下的參數。
  - ✓ 參數 λ_t 的數值並不是重要的，因為已經吸收在 κ*與 θ*中，沒有明白的出現在(3.4)、(3.5)、(3.6)。
- 使用非線性最適化方法，校準出五個模型參數，V_0、κ*、θ*、ρ、σ。
  - ✓ QunatLib、Intel MKL、IMSL、Centerspace NMath 程式庫皆有內建最適化模組。
  - ✓ Nelder-Mead 與 Levenberg-Marquardt 演算法是較為被採用的方法。
  - ✓ 此部分因只要執行一次，CPU 端程式執行即可。

(二)Vanilla Call 解析解

◆ 封閉解公式

➢ 對不發放股利的歐式買權，Heston 模型的封閉解為，

$$C(S_t,V_t,t,T)=S_tP_1-Ke^{-r(T-t)}P_2 \quad \text{......(3.7)}$$

$$P_j(x_t,V_t,T,K)=\frac{1}{2}+\frac{1}{\pi}\int_0^{\infty}\mathrm{Re}\left(\frac{e^{i\phi\ln(K)}f_j(x_t,V_t,T,\phi)}{i\phi}\right)d\phi \quad \text{......(3.8)}$$

$x_t=\ln(S_t)$，$\tau=T-t$，

$$f_j(x_t,V_t,\tau,\phi)=\exp\{C(\tau,\phi)+D(\tau,\phi)V_t+i\phi x_t\} \quad \text{......(3.9)}$$

$$C(\tau,\phi)=r\phi i\tau+\frac{a}{\sigma^2}\left[(b_j-\rho\sigma\phi i+d)\tau-2\ln\left(\frac{1-ge^{d\tau}}{1-g}\right)\right] \quad \text{......(3.10)}$$

$$D(\tau,\phi)=\frac{b_j-\rho\sigma\phi i}{\sigma^2}\left(\frac{1-e^{d\tau}}{1-ge^{d\tau}}\right) \quad \text{......(3.11)}$$

$$g=\frac{b_j-\rho\sigma\phi i+d}{b_j-\rho\sigma\phi i-d} \quad \text{...(3.12)}$$

$$d=\sqrt{(\rho\sigma\phi i-b_j)-\sigma^2(2u_j\phi i-\phi^2)} \quad \text{..(3.13)}$$

$j=1,2$

✓ 其中

$u_1=\frac{1}{2}$，$u_2=-\frac{1}{2}$

$a=k^*\theta^*$，$b_1=k^*-\rho\sigma$，$b_2=k^*$

(三)複數運算

- 前面(3.8)~(3.13)式中，涉及複數的運算，下面簡單摘要其規則。

$z = x + iy$ ，$i = \sqrt{-1}$ ，$\mathrm{Re}(z) = x$ ，$\mathrm{Im}(z) = y$ 。

$z = (x, y)$

$z_1 = x_1 + iy_1 = (x_1, y_1)$ ，$z_2 = x_2 + iy_2 = (x_2, y_2)$

- 四則運算

$$z_1 + z_2 = (x_1 + x_2) + i(y_1 + y_2) = (x_1 + x_2, y_1 + y_2)$$

$$z_1 - z_2 = (x_1 - x_2) + i(y_1 - y_2) = (x_1 - x_2, y_1 - y_2)$$

$$z_1 \times z_2 = (x_1 x_2 - y_1 y_2) + i(x_1 y_2 + x_2 y_1) = (x_1 x_2 - y_1 y_2, x_1 y_2 + x_2 y_1)$$

$$z_1 / z_2 = \frac{(x_1 + iy_1)}{(x_2 + iy_2)} \times \frac{(x_2 - iy_2)}{(x_2 - iy_2)} = \frac{(x_1 x_2 + y_1 y_2)}{x_2^2 + y_2^2} - i \frac{(x_2 y_1 - x_1 y_2)}{x_2^2 + y_2^2}$$

◆ 極座標、冪次與根

$$z = x + iy = r(\cos\theta + i\sin\theta) \text{ , } r = \sqrt{x^2 + y^2} \text{ , } \theta = \arctan\frac{y}{x} = \arg z \text{ ,}$$

$$x = r\cos\theta \text{ , } y = r\sin\theta \text{ ,}$$

$$\bar{z} = x - iy \text{ , } |z| = \sqrt{z\bar{z}} = r$$

$$z^n = r^n(\cos n\theta + i\sin n\theta)$$

$$\sqrt[n]{z} = \sqrt[n]{r}\left(\cos\left(\frac{\theta + 2k\pi}{n}\right) + i\sin\left(\frac{\theta + 2k\pi}{n}\right)\right) \text{ , } k = 0,1,\ldots,n-1$$

◆ 指數函數、尤拉公式與對數函數

$$z = x + iy = r(\cos\theta + i\sin\theta) \text{ , } r = \sqrt{x^2 + y^2} \text{ , } \theta = \arctan\frac{y}{x} = \arg z \text{ ,}$$

$$\exp(z) = \exp(x + iy) = \exp(x)\cdot\exp(iy) = \exp(x)\cdot(\cos y + i\sin y)$$

$$\exp(i\theta) = \cos\theta + i\sin\theta$$

$$\ln(z) = \ln(x + iy) = \ln(r(\cos\theta + i\sin\theta)) = \ln(r) + i\theta$$

(四)數值積分 Gauss-Laguerre 求值法

- (3.8)式的計算涉及半無限區間的積分，可使用 Gauss-Laguerre 法計算，以加速計算效率，
  - 令積分運算式如下式，

$$G=\int_0^\infty f(x)dx$$

  - 令 n 點 Gauss-Laguerre 求值公式為

$$G=\int_0^\infty f(x)dx=\sum_{i=0}^{n-1}\lambda_i f(x_i) \quad \text{......(3.14)}$$

  - 其中 x_i 為下面 n 階 Laguerre 多項式的 n 個零點，λ_i 為求積係數。

$$L_n(x)=e^x\frac{d^n}{dx^n}(x^n e^{-x})\text{ ，}0\le x\le +\infty \quad \text{......(3.15)}$$

- 當 n=5，5 階 Gauss-Laguerre 求積公式的結點為，

$x_0 = 0.26355990$，$x_1 = 1.41340290$，$x_2 = 3.59642600$，$x_3 = 7.08580990$，$x_4 = 12.64080000$。

- 相對應的求積係數為，

$\lambda_0 = 0.6790941054$，$\lambda_1 = 1.638487956$，$\lambda_2 = 2.769426772$，$\lambda_3 = 4.315944000$，$\lambda_4 = 7.104896230$。

(五)特徵函數

◆ (3.8)積分式中 Integrand 對 Phi 的作圖。

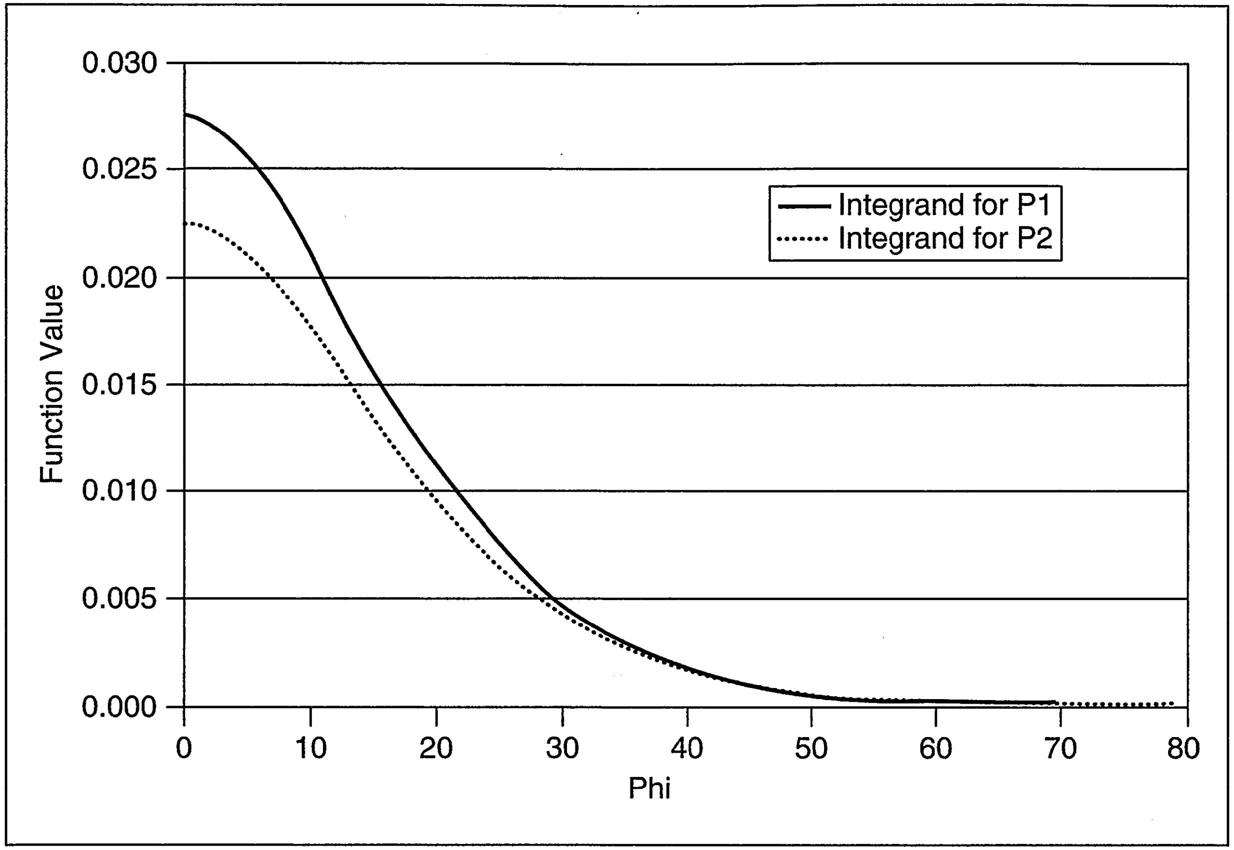


FIGURE 5.4 Convergence of Functions Used in Integration

◆ 在不同相關係數下(ρ=-0.5， ρ=+0.5)，(3.7)式 Call 價格與 Black-Scholes 計算之 Call 價格的差距，H_C-BS_C。

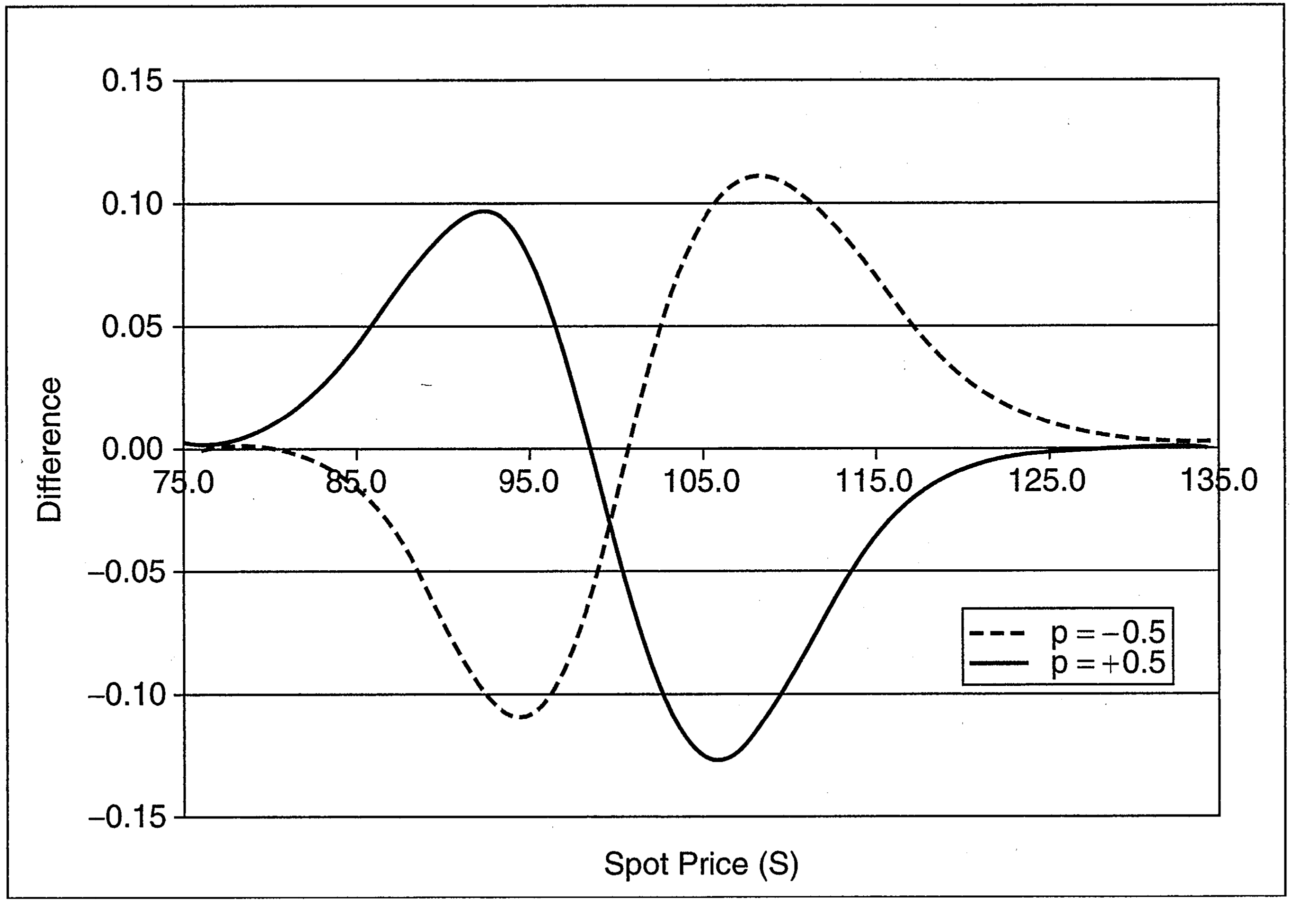


FIGURE 5.8 Plots of Call Price Differences with Varying Correlation

四、避險參數

(一)Delta 與 Gamma

◆ 使用 Center Difference 的方法，以減少誤差。

$$\Delta = \frac{\partial C}{\partial S} = \frac{C(S+h)-C(S-h)}{2h} \qquad \text{(4.1)}$$

$$\Gamma = \frac{\partial^2 C}{\partial S^2} \approx \frac{C(S+h)-2C(S)+C(S-h)}{h^2} \qquad \text{(4.2)}$$

- 使用同一組亂數可使估計誤差較小。
- C(S, σ , r, t, h)，C(S-h)，C(S+h)，三個值。

(二)Vega、Theta 與 Rho

◆ 類似差分，

$$Vega = \frac{\partial C}{\partial \sigma} = \frac{C(\sigma + h) - C(\sigma)}{h} \quad \text{……(4.3)}$$

$$Theta = \frac{\partial C}{\partial t} = \frac{C(t - h) - C(t)}{h} \quad \text{……(4.4)}$$

$$delta = \frac{\partial C}{\partial r} \approx \frac{C(r + h) - C(r)}{h} \quad \text{……(4.5)}$$

- Theta 日數減少。
- C(S, σ, r, t, h)，C(σ +h)，C(t-h)，C(r+h)，四個值。
- 全部六個值，便足夠了。

五、實作案例一

(一)R 語言實作

◆ 使用 R 語言內建的函數與功能，來撰寫 Heston 模型的解析解相對容易，

➢ 主程式

```
setwd("D:\\FEMC\\RCode")
source("HestonPrice.R")
source("HestonProb.R")
# Option features
S = 100;             # Spot price
K = 100;             # Strike price
tau = 0.5;           # Maturity
r = 0.03;            # Risk free rate
q = 0.00;            # Dividend yield
kappa = 5;           # Heston parameter : reversion speed
sigma = 0.5;         # Heston parameter : volatility of variance
rho = -0.8;          # Heston parameter : correlation
theta = 0.05;        # Heston parameter : reversion level
v0 = 0.05;           # Heston parameter : initial variance
lambda = 0;          # Heston parameter : risk preference
                     # Expression for the characteristic function
Trap = 0;            # 0 = Original Heston formulation
                     # 1 = Albrecher et. al. formulation
```

```
# Integration range
Lphi = 0.000001;   # Lower limit
Uphi = 50;         # Upper limit
num = 100;         # subdivision num

# Obtain the Heston put and call
HPut = HestonPrice('P', kappa, theta, lambda, rho, sigma, tau, K, S, r, q, v0,
   Trap, Lphi, Uphi, num);
HCall = HestonPrice('C', kappa, theta, lambda, rho, sigma, tau, K, S, r, q, v0,
   Trap, Lphi, Uphi, num);

# Output the result
print(HPut);
print(HCall);
```

➢ 副程式

```
# Heston (1993) price of a European option.
# Uses the original formulation by Heston
# Heston parameters:
#    kappa  = volatility mean reversion speed parameter
#    theta  = volatility mean reversion level parameter
#    lambda = risk parameter
#    rho    = correlation between two Brownian motions
#    sigma  = volatility of variance
#    v0     = initial variance
# Option features.
#    PutCall = 'C'all or 'P'ut
#    K = strike price
#    S = spot price
#    r = risk free rate
#    q = dividend yield
#    T = maturity
# Integration features
#    L = lower limit
#    U = upper limit
#    num = integration increment

HestonPrice = function(PutCall, kappa, theta, lambda, rho, sigma, T, K, S, r, q, v0, trap, Lphi,
   Uphi, num)
{
   # The integrals
   I1 = HestonProb(Lphi, Uphi, num, kappa, theta, lambda, rho, sigma,
```

```
        T, K, S, r, q, v0, 1, trap);
    I2 = HestonProb(Lphi, Uphi, num, kappa, theta, lambda, rho, sigma,
        T, K, S, r, q, v0, 2, trap);
    # The probabilities P1 and P2
    P1 = 1/2 + I1/pi;
    P2 = 1/2 + I2/pi;
    # The call price
    HestonC = S*exp(-q*T)*P1 - K*exp(-r*T)*P2;
    # Output the option price
    if (PutCall == 'C')
    {
        y = HestonC;
    }
    else
    {
        # The put price by put-call parity
        HestonP = HestonC - S*exp(-q*T) + K*exp(-r*T);
        y = HestonP;
    }
    return(y)
}
```

```
# Returns the risk neutral probabilities P1 and P2.
# integrand = integrand of Probability
# phi = integration variable
# Integration features
#    Lphi = lower limit
#    Uphi = upper limit
# Pnum = 1 or 2 (for the probabilities)
# Heston parameters:
#    kappa  = volatility mean reversion speed parameter
#    theta  = volatility mean reversion level parameter
#    lambda = risk parameter
#    rho    = correlation between two Brownian motions
#    sigma  = volatility of variance
#    v0     = initial variance
# Option features.
#    PutCall = 'C'all or 'P'ut
#    K = strike price
#    S = spot price
#    r = risk free rate
#    q = dividend yield
#    Trap = 1 "Little Trap" formulation
#           0  Original Heston formulation

HestonProb = function(Lphi, Uphi, num, kappa, theta, lambda, rho,
   sigma, tau, K, S, r, q,  v0, Pnum, Trap)
{
   x = log(S);
   a = kappa * theta;
```

```
if (Pnum == 1)
{
   u = 0.5;
   b = kappa + lambda - rho * sigma;
}
else
{
   u = -0.5;
   b = kappa + lambda;
}

integrand = function(phi)
{
   Zi = complex(0, 1);

   d = sqrt((rho*sigma*phi*Zi - b)^2 - sigma^2*(2*u*phi*Zi - phi^2));
   g = (b - rho*sigma*phi*Zi + d) / (b - rho*sigma*phi*Zi - d);

   if (Trap==1)    # "Little Heston Trap" formulation
   {
      c = 1/g;
      D = (b - rho*sigma*Zi*phi - d)/sigma^2*((1-exp(-d*tau))
         /(1-c*exp(-d*tau)));
      G = (1 - c*exp(-d*tau))/(1-c);
      C = (r-q)*Zi*phi*tau + a/sigma^2*((b - rho*sigma*Zi*phi - d)*tau
         - 2*log(G));
   }
   else
```

```
      {
         if (Trap==0)  # Original Heston formulation.
         {
            G = (1 - g*exp(d*tau))/(1-g);
            C = (r-q)*Zi*phi*tau + a/sigma^2*((b - rho*sigma*Zi*phi + d)*tau
               - 2*log(G));
            D = (b - rho*sigma*Zi*phi + d)/sigma^2*((1-exp(d*tau))
               /(1-g*exp(d*tau)));
         }
      }

      # The characteristic function.
      f = exp(C + D*v0 + Zi*phi*x);

      # Return the real part of the integrand.
      integ = Re(exp(-Zi*phi*log(K))*f/Zi/phi);

      return(integ);
   }

   Total = integrate(f=integrand,lower=Lphi, upper=Uphi, subdivisions=num);
   # Get value of the integrate function
   ans = Total$value;

   return(ans);
}
```

(二)C#語言實作

◆ Most Simple Version

➢ \VS2015Prj\HestonPrice_GaussLaguerre*.*

```
// Heston parameters
public struct HParam
{
    public double kappa;       // Mean reversion speed
    public double theta;       // Mean reversion level
    public double sigma;       // Volatility of variance
    public double v0;          // Initial variance
    public double rho;         // Correlation
    public double lambda;      // Risk parameter
}
// Settings for the option price calculation
public struct OpSet
{
    public double S;           // Spot price
    public double K;           // Strke price
    public double T;           // Maturity
    public double r;           // Risk free rate
    public double q;           // Dividend
    public string PutCall;     // "P"ut or "C"all
    public int trap;           // 1="Little Trap" characteristic function; 2=Original Heston c.f.
```

```
}

class HestonPriceGaussLaguerre
{
    static void Main(string[] args)
    {
        // 32-point Gauss-Laguerre Abscissas and weights
        double[] x = new Double[32];
        double[] w = new Double[32];
        using(TextReader reader = File.OpenText("../../GaussLaguerre32.txt"))
        {
            for(int k=0;k<=31;k++)
            {
                string text = reader.ReadLine();
                string[] bits = text.Split(' ');
                x[k] = double.Parse(bits[0]);
                w[k] = double.Parse(bits[1]);
            }
        }
        // Heston parameters
        HParam param = new HParam();
```

```
        param.kappa = 1.5;          param.theta = 0.04;         param.sigma = 0.3;
        param.v0 = 0.05412;         param.rho = -0.9;           param.lambda = 0.0;
        // Option settings
        OpSet settings = new OpSet();
        settings.S = 101.52;        settings.K = 100.0;         settings.T = 0.15;
        settings.r = 0.02;          settings.q = 0.0;           settings.PutCall = "C";
        settings.trap = 1;
        // The Heston price
        HestonPrice HP = new HestonPrice();
        double Price = HP.HestonPriceGaussLaguerre(param,settings,x,w);
        Console.WriteLine("Heston price using 32-point Gauss Laguerre");
        Console.WriteLine("------------------------------------------ ");
        Console.WriteLine("Option Flavor =  {0,0:F5}",settings.PutCall);
        Console.WriteLine("Strike Price  =  {0,0:0}" ,settings.K);
        Console.WriteLine("Maturity      =  {0,0:F2}",settings.T);
        Console.WriteLine("Price         =  {0,0:F4}",Price);
        Console.WriteLine("------------------------------------------ ");
        Console.WriteLine(" ");
    }
}
```

```
class HestonPrice
{
    // Heston Integrand
    public double HestonProb(double phi,HParam param,OpSet settings,int Pnum)
    {
        Complex i  = new Complex(0.0,1.0);                // Imaginary unit
        double S = settings.S;
        double K = settings.K;
        double T = settings.T;
        double r = settings.r;
        double q = settings.q;
        double kappa = param.kappa;
        double theta = param.theta;
        double sigma = param.sigma;
        double v0 = param.v0;
        double rho = param.rho;
        double lambda = param.lambda;
        double x = Math.Log(S);
        double a = kappa*theta;
        int Trap = settings.trap;
        Complex b,u,d,g,c,D,G,C,f,integrand = new Complex();

        // Parameters "u" and "b" are different for P1 and P2
        if(Pnum==1)
        {
            u = 0.5;
```

```
        b = kappa + lambda - rho*sigma;
    }
    else
    {
        u = -0.5;
        b = kappa + lambda;
    }
    d = Complex.Sqrt(Complex.Pow(rho*sigma*i*phi - b,2.0) - sigma*sigma*(2.0*u*i*phi - phi*phi));
    g = (b - rho*sigma*i*phi + d) / (b - rho*sigma*i*phi - d);
    if(Trap==1)
    {
        // "Little Heston Trap" formulation
        c = 1.0/g;
        D = (b - rho*sigma*i*phi - d)/sigma/sigma*((1.0-Complex.Exp(-d*T))/(1.0-c*Complex.Exp(-d*T)));
        G = (1.0 - c*Complex.Exp(-d*T))/(1-c);
        C = (r-q)*i*phi*T + a/sigma/sigma*((b - rho*sigma*i*phi - d)*T - 2.0*Complex.Log(G));
    }
    else
    {
        // Original Heston formulation.
        G = (1.0 - g*Complex.Exp(d*T))/(1.0-g);
        C = (r-q)*i*phi*T + a/sigma/sigma*((b - rho*sigma*i*phi + d)*T - 2.0*Complex.Log(G));
        D = (b - rho*sigma*i*phi + d)/sigma/sigma*((1.0-Complex.Exp(d*T))/(1.0-g*Complex.Exp(d*T)));
    }
```

```
    // The characteristic function.
    f = Complex.Exp(C + D*v0 + i*phi*x);

    // The integrand.
    integrand = Complex.Exp(-i*phi*Math.Log(K))*f/i/phi;

    // Return the real part of the integrand.
    return integrand.Real;
}


// Heston Price by Gauss-Laguerre Integration
public double HestonPriceGaussLaguerre(HParam param,OpSet settings,double[] x,double[] w)
{
    double[] int1 = new Double[32];
    double[] int2 = new Double[32];
    // Numerical integration
    for(int j=0;j<=31;j++)
    {
        int1[j] = w[j] * HestonProb(x[j],param,settings,1);
        int2[j] = w[j] * HestonProb(x[j],param,settings,2);
    }

    // Define P1 and P2
    double pi = Math.PI;
    double P1 = 0.5 + 1.0/pi*int1.Sum();
```

```
        double P2 = 0.5 + 1.0/pi*int2.Sum();

        // The call price
        double S = settings.S;
        double K = settings.K;
        double T = settings.T;
        double r = settings.r;
        double q = settings.q;
        string PutCall = settings.PutCall;
        double HestonC = S*Math.Exp(-q*T)*P1 - K*Math.Exp(-r*T)*P2;

        // The put price by put-call parity
        double HestonP = HestonC - S*Math.Exp(-q*T) + K*Math.Exp(-r*T);

        // Output the option price
        if(PutCall == "C")
           return HestonC;
        else
           return HestonP;
    }
}
```

◆ Consolidated Heston Model

➢ \VS2015Prj\Analytic*.*

```
OpSet opSet = new OpSet();
HParam hParam = new HParam();

opSet.PutCall = "C";
opSet.S = Convert.ToDouble(textBox2.Text);
opSet.K = Convert.ToDouble(textBox3.Text);
opSet.T = Convert.ToDouble(textBox4.Text);
opSet.r = Convert.ToDouble(textBox5.Text);
opSet.q = Convert.ToDouble(textBox6.Text);

hParam.kappa = Convert.ToDouble(textBox7.Text);
hParam.theta = Convert.ToDouble(textBox8.Text);
hParam.sigma = Convert.ToDouble(textBox9.Text);
hParam.v0 = Convert.ToDouble(textBox10.Text);
hParam.rho = Convert.ToDouble(textBox11.Text);
hParam.lambda = Convert.ToDouble(textBox12.Text);

Stopwatch SW = new Stopwatch();
SW.Start();
//T01_GaussLaguerre.GaussLaguerre();
double C0 = T01_GaussLaguerre.GaussLaguerreConsolidated(opSet, hParam);
textBox13.Text = C0.ToString();
```

```
SW.Stop();
textBox21.Text = SW.ElapsedMilliseconds.ToString();

double dS = 0.005 * opSet.S;
opSet.S = opSet.S + dS;
double Cplus = T01_GaussLaguerre.GaussLaguerreConsolidated(opSet, hParam);
opSet.S = opSet.S - dS;
double Cminus = T01_GaussLaguerre.GaussLaguerreConsolidated(opSet, hParam);
double CDelta = (Cplus - Cminus) / (2 * dS);
textBox15.Text = CDelta.ToString();
```

```
public static double GaussLaguerreConsolidated(OpSet opSet, HParam hParam)
{
    // 32-point Gauss-Laguerre Abscissas and weights
    double[] x = new Double[32];
    double[] w = new Double[32];
    using (TextReader reader = File.OpenText("../../GaussLaguerre32.txt"))
    {
        for (int k = 0; k <= 31; k++)
        {
            string text = reader.ReadLine();
            string[] bits = text.Split(' ');
            x[k] = double.Parse(bits[0]);
            w[k] = double.Parse(bits[1]);
        }
    }
    HParam param = new HParam();
    param.kappa = hParam.kappa;          // Heston Parameter: Mean reversion speed
    param.theta = hParam.theta;          // Heston Parameter: Mean reversion level
    param.sigma = hParam.sigma;          // Heston Parameter: Volatility of Variance
    param.v0 = hParam.v0;                // Heston Parameter: Current Variance
    param.rho = hParam.rho;              // Heston Parameter: Correlation
    param.lambda = 0.0;                  // Heston Parameter: Risk parameter

    OpSet settings = new OpSet();
    settings.S = opSet.S;                // Spot Price
    settings.K = opSet.K;                // Strike Price
```

```
    settings.T = opSet.T;                  // Maturity in Years
    settings.r = opSet.r;                  // Interest Rate
    settings.q = opSet.q;                  // Dividend yield
    settings.PutCall = opSet.PutCall;      // "P"ut or "C"all
    settings.trap = 1;                     // 1="Little Trap" characteristic function

    // The Heston price
    HestonPriceConsolidated HPC = new HestonPriceConsolidated();
    double Price = HPC.HestonPriceConsol(param, settings, x, w);
    return Price;
}
```

```
// Heston Price by Gauss-Laguerre Integration
public double HestonPriceConsol(HParam param, OpSet settings, double[] x, double[] w)
{
    double[] int1 = new Double[32];
    // Numerical integration
    for (int j = 0; j <= 31; j++)
    {
        int1[j] = w[j] * HestonProbConsol(x[j], param, settings);
    }

    // Define P1 and P2
    double pi = Math.PI;
    double I = int1.Sum();

    // The call price
    double S = settings.S;
    double K = settings.K;
    double r = settings.r;
    double q = settings.q;
    double T = settings.T;
    string PutCall = settings.PutCall;
    double HestonC = 0.5 * S * Math.Exp(-q * T) - 0.5 * K * Math.Exp(-r * T) + I / pi;

    // The put price by put-call parity
    double HestonP = HestonC - S * Math.Exp(-q * T) + K * Math.Exp(-r * T);
```

```
    // Output the option price
    if (PutCall == "C")
        return HestonC;
    else
        return HestonP;
}


// Heston Integrand
public double HestonProbConsol(double phi, HParam param, OpSet settings)
{
    Complex i = new Complex(0.0, 1.0);                  // Imaginary unit
    double S = settings.S;
    double K = settings.K;
    double T = settings.T;
    double r = settings.r;
    double q = settings.q;
    double kappa = param.kappa;
    double theta = param.theta;
    double sigma = param.sigma;
    double v0 = param.v0;
    double rho = param.rho;
    double lambda = param.lambda;
    double x = Math.Log(S);
    double a = kappa * theta;
    int Trap = settings.trap;
```

```
Complex b1, u1, d1, g1, c1, D1, G1, C1, f1, b2, u2, d2, g2, c2, D2, G2, C2, f2, integrand = new Complex();

// The first characteristic function
u1 = 0.5;
b1 = kappa + lambda - rho * sigma;
d1 = Complex.Sqrt(Complex.Pow(rho*sigma*i*phi-b1, 2) - sigma*sigma*(2.0*u1*i*phi-phi*phi));
g1 = (b1 - rho * sigma * i * phi + d1) / (b1 - rho * sigma * i * phi - d1);
if (Trap == 1)
{
    // "Little Heston Trap" formulation
    c1 = 1.0 / g1;
    D1 = (b1 - rho * sigma * i * phi - d1) / sigma / sigma
        * ((1.0 - Complex.Exp(-d1 * T)) / (1.0 - c1 * Complex.Exp(-d1 * T)));
    G1 = (1.0 - c1 * Complex.Exp(-d1 * T)) / (1.0 - c1);
    C1 = (r - q) * i * phi * T + a / sigma / sigma
        * ((b1 - rho * sigma * i * phi - d1) * T - 2.0 * Complex.Log(G1));
}
else
{
    // Original Heston formulation.
    G1 = (1.0 - g1 * Complex.Exp(d1 * T)) / (1.0 - g1);
    C1 = (r - q) * i * phi * T + a / sigma / sigma
        * ((b1 - rho * sigma * i * phi + d1) * T - 2.0 * Complex.Log(G1));
    D1 = (b1 - rho * sigma * i * phi + d1) / sigma / sigma
        * ((1.0 - Complex.Exp(d1 * T)) / (1.0 - g1 * Complex.Exp(d1 * T)));
}
```

```
f1 = Complex.Exp(C1 + D1 * v0 + i * phi * x);

// The second characteristic function
u2 = -0.5;
b2 = kappa + lambda;
d2 = Complex.Sqrt(Complex.Pow(rho * sigma * i * phi - b2, 2)
    - sigma * sigma * (2.0 * u2 * i * phi - phi * phi));
g2 = (b2 - rho * sigma * i * phi + d2) / (b2 - rho * sigma * i * phi - d2);
if (Trap == 1)
{
    // "Little Heston Trap" formulation
    c2 = 1.0 / g2;
    D2 = (b2 - rho * sigma * i * phi - d2) / sigma / sigma
        * ((1.0 - Complex.Exp(-d2 * T)) / (1.0 - c2 * Complex.Exp(-d2 * T)));
    G2 = (1.0 - c2 * Complex.Exp(-d2 * T)) / (1.0 - c2);
    C2 = (r - q) * i * phi * T + a / sigma / sigma
        * ((b2 - rho * sigma * i * phi - d2) * T - 2.0 * Complex.Log(G2));
}
else
{
    // Original Heston formulation.
    G2 = (1.0 - g2 * Complex.Exp(d2 * T)) / (1.0 - g2);
    C2 = (r - q) * i * phi * T + a / sigma / sigma
        * ((b2 - rho * sigma * i * phi + d2) * T - 2.0 * Complex.Log(G2));
    D2 = (b2 - rho * sigma * i * phi + d2) / sigma / sigma
        * ((1.0 - Complex.Exp(d2 * T)) / (1.0 - g2 * Complex.Exp(d2 * T)));
```

```
        }
        f2 = Complex.Exp(C2 + D2 * v0 + i * phi * x);

        // The integrand.
        integrand = Complex.Exp(-i * phi * Complex.Log(K)) / i / phi
            * (S * Complex.Exp(-q * T) * f1 - K * Complex.Exp(-r * T) * f2);

        // Return the real part of the integrand.
        return integrand.Real;
    }
```