

# 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](mailto: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

# Part I Heston 模型與結構商品

## 設計開發

# 主題四 結構商品的實例

一、市場校準

二、目標贖回型遠期契約(Target Redemption Forward)契約規格

三、觸及失效型遠期契約(Knock-Out Forward)契約規格

四、實作案例四

# 一、市場校準

## (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。
- $25\Delta$  Call 與 Put 的 Risk Reversal，RR。
- $25\Delta$  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.  
Enter 1<GO> to Save

P167c Curncy**OVDV**

**Currency Volatility Surface**

Save	Send	Download	Options	3D Graph	* Bloomberg (BGN) USDCNY					
Currencies: USD-CNY				Date: 5/ 7/08		Format: 1 RR/BF				
USD Calls/Puts Deltas				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

\*Default

RR = USD Call - USD Put

Australia 61 2 9777 8600 Brazil 5511 3048 4500 Europe 44 20 7330 7500 Germany 49 69 9204 1210 Hong Kong 852 2977 6000  
Japan 81 3 3201 8900 Singapore 65 6212 1000 U.S. 1 212 318 2000 Copyright 2008 Bloomberg Finance L.P.  
H169-403-0 07-May-2008 15:11:59

◆ 美元 LIBOR: RT: LIBOR01

REUTERS BBA LIBOR RATES						LIBOR01
BRITISH BANKERS ASSOCIATION INTEREST SETTLEMENT RATES						Alternative to <3750>
[06/05/08] RATES AT 11:00 LONDON TIME 06/05/2008						Disclaimer <LIBORDISC>
						BBA Guide <BBAMENU>
	USD	GBP	CAD	EUR	JPY	EUR 365
0/N	<u>2.36500</u>	5.10000	3.02667	4.07750	SN 0.55625	4.13413
1WK	<u>2.58500</u>	5.11938	3.09667	4.26250	0.59875	4.32170
2WK	<u>2.63000</u>	5.24750	3.12500	4.30188	0.62000	4.36163
1MO	<u>2.67375</u>	5.45000	3.21000	4.38625	0.67375	4.44717
2MO	<u>2.72375</u>	5.67750	3.33000	4.68000	0.81656	4.74500
3MO	<u>2.75750</u>	5.80563	3.38833	4.85563	0.92125	4.92307
4MO	2.79125	5.80563	3.40000	4.86688	0.94406	4.93448
5MO	2.83625	5.80500	3.40000	4.87313	0.96406	4.94081
6MO	<u>2.87625</u>	5.80625	3.40500	4.88188	0.98375	4.94968
7MO	2.89875	5.80625	3.43333	4.89063	1.00625	4.95856
8MO	2.91813	5.80563	3.43667	4.90375	1.02625	4.97186
9MO	<u>2.94000</u>	5.80438	3.45500	4.91563	1.04500	4.98390
10MO	2.96375	5.80250	3.49333	4.92813	1.06438	4.99658
11MO	2.99000	5.80188	3.51500	4.94375	1.08313	5.01241
12MO	<u>3.01500</u>	5.80188	3.55000	4.95750	1.10375	5.02635

<0#LIBORSUPERRICS> RICs for above <0#LIBORRICS> Contributor RICs

◆ NDF Swap Point: RT: TRADNDF

15:18 07MAY08

Tradition Asia Ltd  
ASIAN NON-DELIVERABLE FORWARDS

SP01881

TRADNDF

	FWD TWD USD/TWD	FWD CNY USD/CNY	OUT PHP USD/PHP	OUT INR USD/INR	OUT KRW USD/KRW
SP	30.495	<del>6.9858</del>	42.39/42.40	41.16/41.17	***** /*****
<del>1W</del>	<del>-0.000/+0.000</del>	6.9800 /6.9850	42.42/42.45	41.19/41.22	***** /*****
<del>1M</del>	<del>-0.060/-0.030</del>	6.9700 /6.9750	42.54/42.59	41.32/41.37	***** /*****
<del>2M</del>	<del>-0.140/-0.110</del>	6.9370 /6.9420	42.66/42.71	41.39/41.44	***** /*****
<del>3M</del>	<del>-0.245/-0.215</del>	6.9000 /6.9050	42.79/42.84	41.43/41.48	***** /*****
<del>6M</del>	<del>-0.455/-0.415</del>	<u>6.7670</u> /6.7750	43.11/43.21	41.62/41.72	***** /*****
<del>9M</del>	<del>-0.680/-0.630</del>	6.6150 /6.6250	43.44/43.54	41.74/41.84	***** /*****
<del>1Y</del>	<del>-0.850/-0.800</del>	<u>6.4680</u> /6.4730	43.72/43.82	41.89/41.99	***** /*****

CONTACT : DANNY / PAULINE TEL: 852-2521-2303

HKG DEALING : TRND  
SGP DEALING : TRSA



## ◆ UBS Volatility Quote Page ◦

http://ibol01.ubb.ubs.com - USDJPY VolCast from UBS Investment Bank - Microsoft Internet Explorer

Ccy pair	USDJPY	Connect Status:	<input type="checkbox"/>	<input checked="" type="checkbox"/> Vol Grid	<a href="#">HiLoView</a>
JPY per USD	121.76	Bid / Ask	<input type="radio"/>	<input checked="" type="checkbox"/> RiskReversals	<a href="#">VolHistory</a>
Update date	11-Jul-2007	Mid price	<input checked="" type="radio"/>	<input checked="" type="checkbox"/> Strangles	<a href="#">ProbView</a>
Update time	02:05:08	Select pair (200 loaded)	<input type="button" value="v"/>	<input checked="" type="checkbox"/> ButterFlies	<a href="#">VolView</a>
Cut time	TKY				
Face	USD				

VOL	OH	1W	2W	3W	1M	6W	2M	3M	4M	6M	9M	1Y	2Y
Prem Ccy	12-Jul-2007	18-Jul-2007	25-Jul-2007	01-Aug-2007	09-Aug-2007	22-Aug-2007	11-Sep-2007	11-Oct-2007	09-Nov-2007	10-Jan-2008	10-Apr-2008	10-Jul-2008	09-Jul-2009
USD	Thu:1	Wed:7	Wed:14	Wed:21	Thu:29	Wed:42	Tue:62	Thu:92	Fri:121	Thu:183	Thu:274	Thu:365	Thu:729
10D USD C	11.69	7.42	7.20	6.90	6.65	6.48	6.41	6.49	6.51	6.53	6.51	6.57	6.49
15D USD C	11.78	7.48	7.23	6.92	6.67	6.50	6.43	6.48	6.50	6.52	6.52	6.56	6.39
20D USD C	11.97	7.63	7.35	7.04	6.79	6.63	6.57	6.62	6.59	6.56	6.53	6.54	6.38
25D USD C	12.12	7.75	7.45	7.12	6.87	6.72	6.65	6.67	6.65	6.62	6.60	6.60	6.46
35D USD C	12.50	8.06	7.71	7.38	7.13	6.97	6.91	6.92	6.89	6.87	6.85	6.85	6.81
ATM	13.00	8.50	8.10	7.75	7.50	7.35	7.30	7.30	7.28	7.25	7.25	7.25	7.15
35D USD P	13.65	9.08	8.63	8.27	8.02	7.88	7.83	7.84	7.83	7.81	7.82	7.83	7.59
25D USD P	14.37	9.75	9.25	8.87	8.62	8.49	8.45	8.47	8.49	8.48	8.50	8.52	8.36
20D USD P	14.78	10.12	9.60	9.22	8.98	8.85	8.82	8.86	8.88	8.87	8.91	8.95	8.89
15D USD P	15.27	10.58	10.02	9.63	9.38	9.25	9.22	9.27	9.34	9.39	9.47	9.55	9.54
10D USD P	15.81	11.08	10.50	10.10	9.85	9.72	9.69	9.76	9.95	10.13	10.27	10.41	10.63
Risk Reversals													
10D R/R	4.20P	3.63P	3.28P	3.19P	3.20P	3.23P	3.28P	3.27P	3.43P	3.60P	3.75P	3.84P	4.13P
15D R/R	3.45P	3.09P	2.79P	2.71P	2.71P	2.75P	2.79P	2.79P	2.84P	2.86P	2.94P	2.98P	3.15P
20D R/R	2.83P	2.50P	2.25P	2.19P	2.18P	2.22P	2.26P	2.25P	2.29P	2.31P	2.37P	2.40P	2.51P
25D R/R	2.24P	1.99P	1.80P	1.74P	1.75P	1.78P	1.80P	1.80P	1.84P	1.85P	1.90P	1.92P	1.90P
35D R/R	1.15P	1.02P	0.91P	0.89P	0.89P	0.90P	0.92P	0.91P	0.94P	0.95P	0.96P	0.98P	0.79P
Strangles													
10D Strg	13.75	9.25	8.85	8.50	8.25	8.10	8.05	8.12	8.23	8.33	8.39	8.49	8.56
15D Strg	13.52	9.03	8.63	8.28	8.03	7.88	7.83	7.88	7.92	7.96	7.99	8.06	7.97
20D Strg	13.37	8.87	8.48	8.13	7.88	7.74	7.69	7.74	7.74	7.71	7.72	7.75	7.64
25D Strg	13.25	8.75	8.35	8.00	7.75	7.60	7.55	7.57	7.57	7.55	7.55	7.56	7.41
35D Strg	13.08	8.57	8.17	7.82	7.57	7.43	7.37	7.38	7.36	7.34	7.34	7.34	7.20
Butterflies													
10D Fly	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.82	0.95	1.08	1.14	1.24	1.41
15D Fly	0.52	0.53	0.53	0.53	0.53	0.52	0.53	0.58	0.64	0.71	0.74	0.81	0.82
20D Fly	0.37	0.37	0.38	0.38	0.38	0.39	0.39	0.44	0.46	0.46	0.47	0.50	0.49
25D Fly	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.27	0.29	0.30	0.30	0.31	0.26
35D Fly	0.08	0.07	0.07	0.07	0.07	0.07	0.07	0.08	0.09	0.09	0.09	0.09	0.05

◆ 在外匯市場中，所謂 ATM 波動性，是指使 Straddle 策略為 0 Δ 的執行價格時的波動性。

- Straddle 策略為 Call(K,T)+Put(K,T)。
- 此 Straddle 策略因為 0 Δ，因此無需 Delta Hedge。
- 由於  $\Delta_C = -\Delta_P$ ，因此有下面關係，

$$e^{-r_f T} \Phi \left( \frac{\ln \left( \frac{S_0}{K_{ATM}} \right) + \left( r_d - r_f + \frac{\sigma_{ATM}^2}{2} \right) T}{\sigma_{ATM} \sqrt{T}} \right) = e^{-r_f T} \Phi \left( - \frac{\ln \left( \frac{S_0}{K_{ATM}} \right) + \left( r_d - r_f + \frac{\sigma_{ATM}^2}{2} \right) T}{\sigma_{ATM} \sqrt{T}} \right) \dots\dots\dots (1.1)$$

✓  $\sigma_{ATM}$  為 ATM 的波動性， $K_{ATM}$  為 ATM 的執行價格。

✓  $\Phi$  為常態分配的累積機率密度函數。

- 由(1.1)可得，

$$K_{ATM} = S_0 e^{\left( r_d - r_f + \frac{1}{2} \sigma_{ATM}^2 \right) T} \dots\dots\dots (1.2)$$

◆ RR 為同時買入一個 Call 與賣出一個 Put，兩者有對稱的  $\Delta$ 。

➤ RR 通常以  $\sigma_{25\Delta c}$  與  $\sigma_{25\Delta p}$  的差額報價。因此，我們有下面關係，

$$\sigma_{RR} = \sigma_{25\Delta c} - \sigma_{25\Delta p} \dots\dots\dots(1.3)$$

◆ VWB 為賣出一個 ATM 的 Straddle，同時買入一個  $25\Delta$  的 Strangle。

➤ 為達到 Vega-weighted，前者的數量必需小於後者的數量。

✓ 因為 Straddle 的 Vega 大於 Strangle 的 Vega。

➤ VWB 的波動性關係可表示為，

$$\sigma_{VWB} = \frac{\sigma_{25\Delta c} + \sigma_{25\Delta p}}{2} - \sigma_{ATM} \dots\dots\dots(1.4)$$

◆ 由(1.3)與(1.4)式，可求得  $\sigma_{25\Delta c}$  與  $\sigma_{25\Delta p}$  這兩個隱含波動性如下，

$$\sigma_{25\Delta c} = \sigma_{ATM} + \sigma_{VWB} + \frac{1}{2}\sigma_{RR} \dots\dots\dots(1.5)$$

$$\sigma_{25\Delta p} = \sigma_{ATM} + \sigma_{VWB} - \frac{1}{2}\sigma_{RR} \dots\dots\dots(1.6)$$

◆ 利用(1.5)與(1.6)式，可求得  $K_{25\Delta c}$  如下式，

$$e^{-r_f T} \Phi \left( \frac{\ln \left( \frac{S_0}{K_{25\Delta c}} \right) + \left( r_d - r_f + \frac{\sigma_{25\Delta c}^2}{2} \right) T}{\sigma_{25\Delta c} \sqrt{T}} \right) = 0.25$$

$$K_{25\Delta c} = S_0 e^{\alpha \sigma_{25\Delta c} \sqrt{T} + \left( r_d - r_f + \frac{1}{2} \sigma_{25\Delta c}^2 \right) T} \dots\dots\dots(1.7)$$

$$\alpha = \Phi^{-1} \left( \frac{1}{4} e^{r_f T} \right)$$

◆  $K_{25\Delta p}$  如下式，

$$-e^{-r_f T} \Phi \left( -\frac{\ln \left( \frac{S_0}{K_{25\Delta p}} \right) + \left( r_d - r_f + \frac{\sigma_{25\Delta p}^2}{2} \right) T}{\sigma_{25\Delta p} \sqrt{T}} \right) = -0.25$$

$$K_{25\Delta p} = S_0 e^{-\alpha \sigma_{25\Delta p} \sqrt{T} + \left( r_d - r_f + \frac{1}{2} \sigma_{25\Delta p}^2 \right) T} \dots\dots\dots (1.8)$$

$$\alpha = \Phi^{-1} \left( \frac{1}{4} e^{r_f T} \right)$$

➤ 通常我們有  $\alpha > 0$  且  $K_{25\Delta p} < K_{ATM} < K_{25\Delta c}$ 。

## (二) Smile Effect

### ◆ 波動性 v.s. 執行價格(EURUSD, 2005/7/1)

EURUSD data as of 1 July 2005

$$T = 3m (= 94/365y)$$

$$S_0 = 1.205$$

$$\sigma_{\text{ATM}} = 9.05\%$$

$$\sigma_{\text{RR}} = -0.50\%$$

$$\sigma_{\text{VWB}} = 0.13\%$$

$\Rightarrow$

$$\sigma_{50\Delta c} = 8.93\%$$

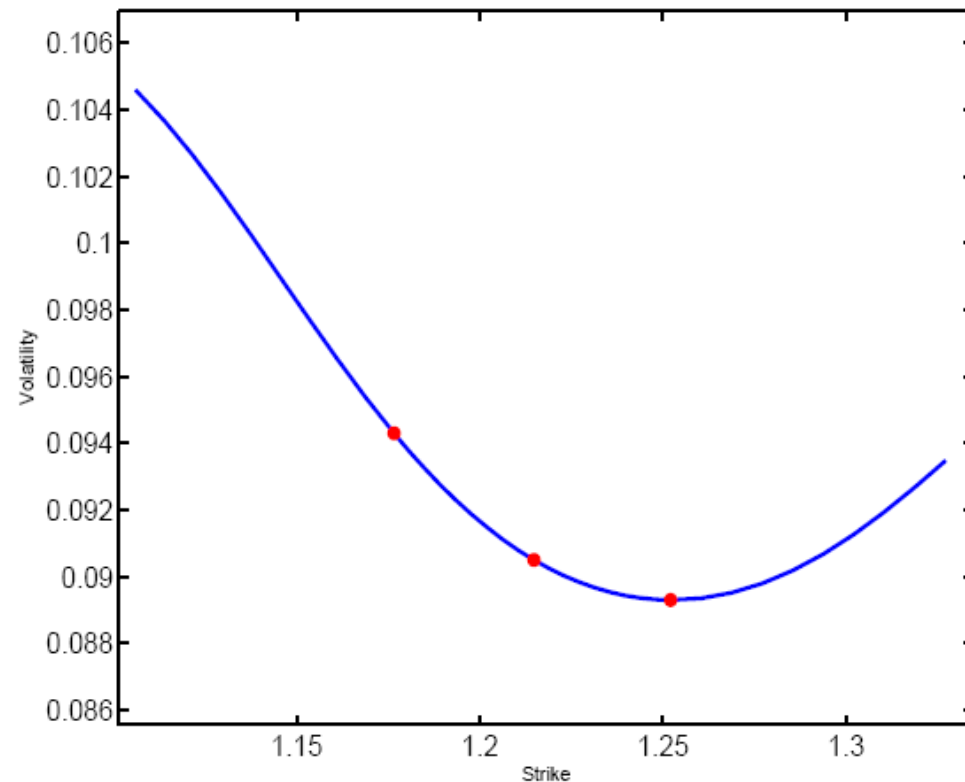
$$\sigma_{25\Delta c} = 9.05\%$$

$$\sigma_{25\Delta p} = 9.43\%$$

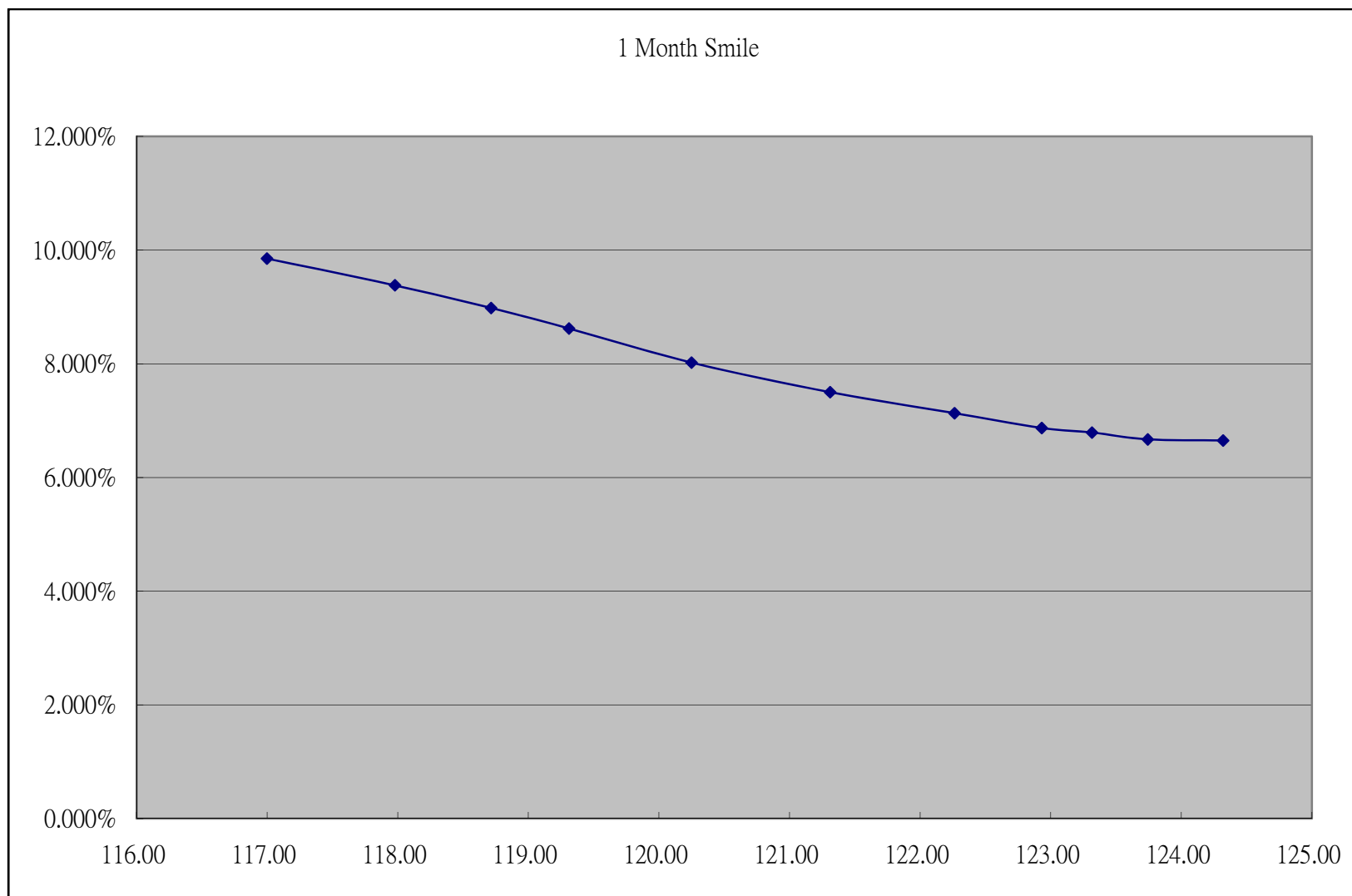
$$K_{\text{ATM}} = 1.2148$$

$$K_{25\Delta c} = 1.1767$$

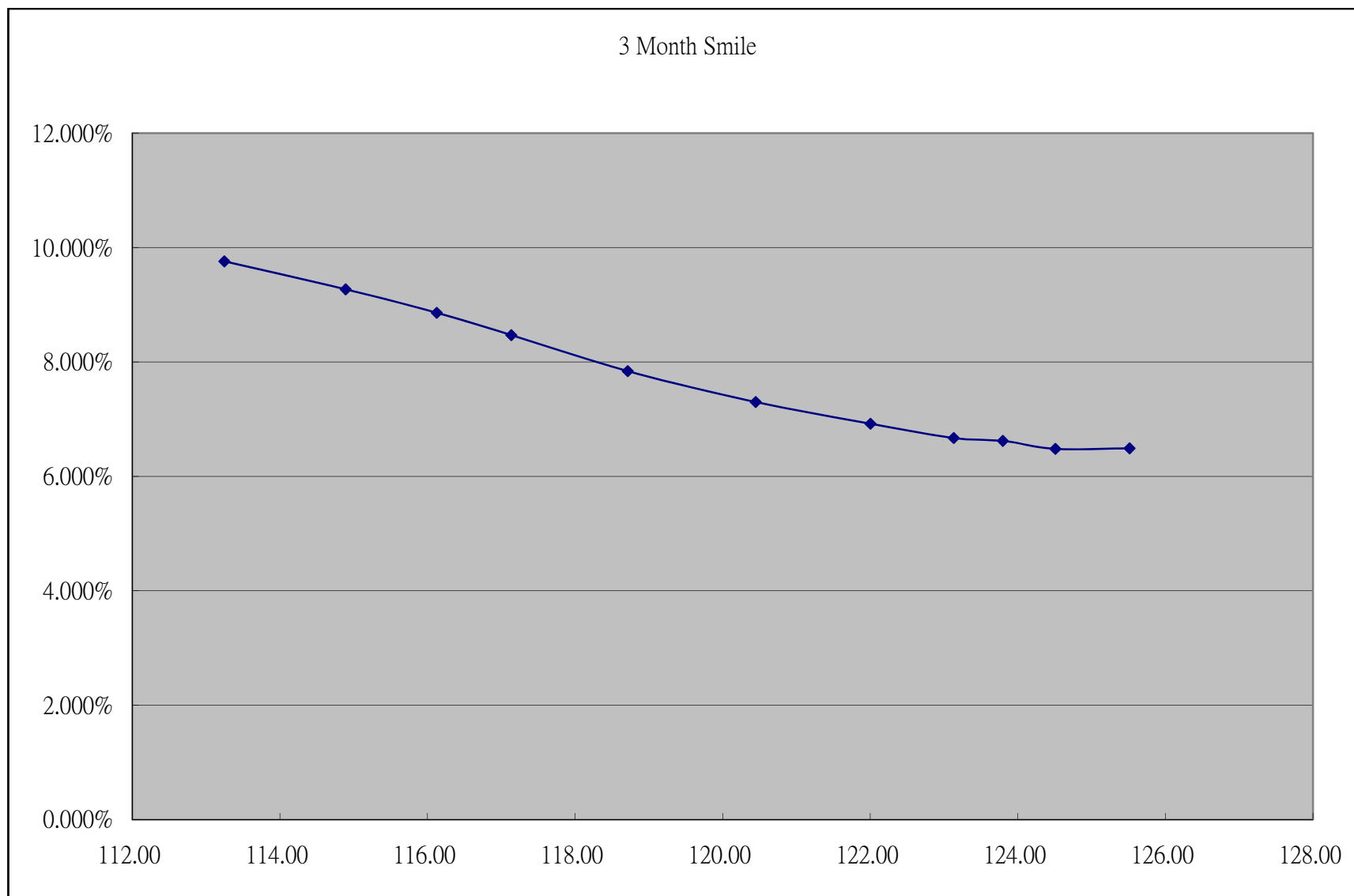
$$K_{25\Delta p} = 1.2521$$



◆ 波動性 v.s.執行價格(USDJPY, 2007/7/11)

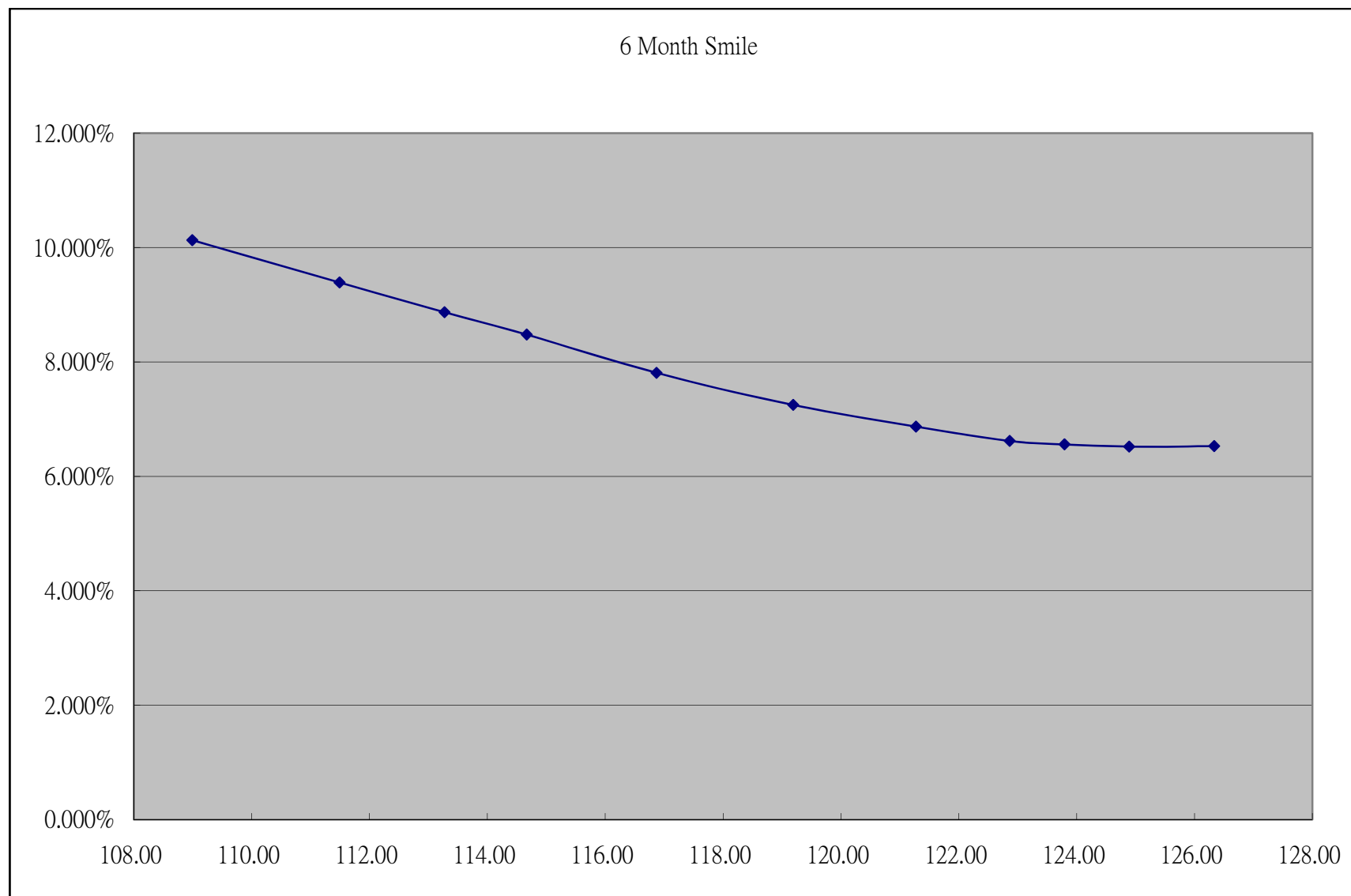


◆ 波動性 v.s.執行價格(USDJPY, 2007/7/11)



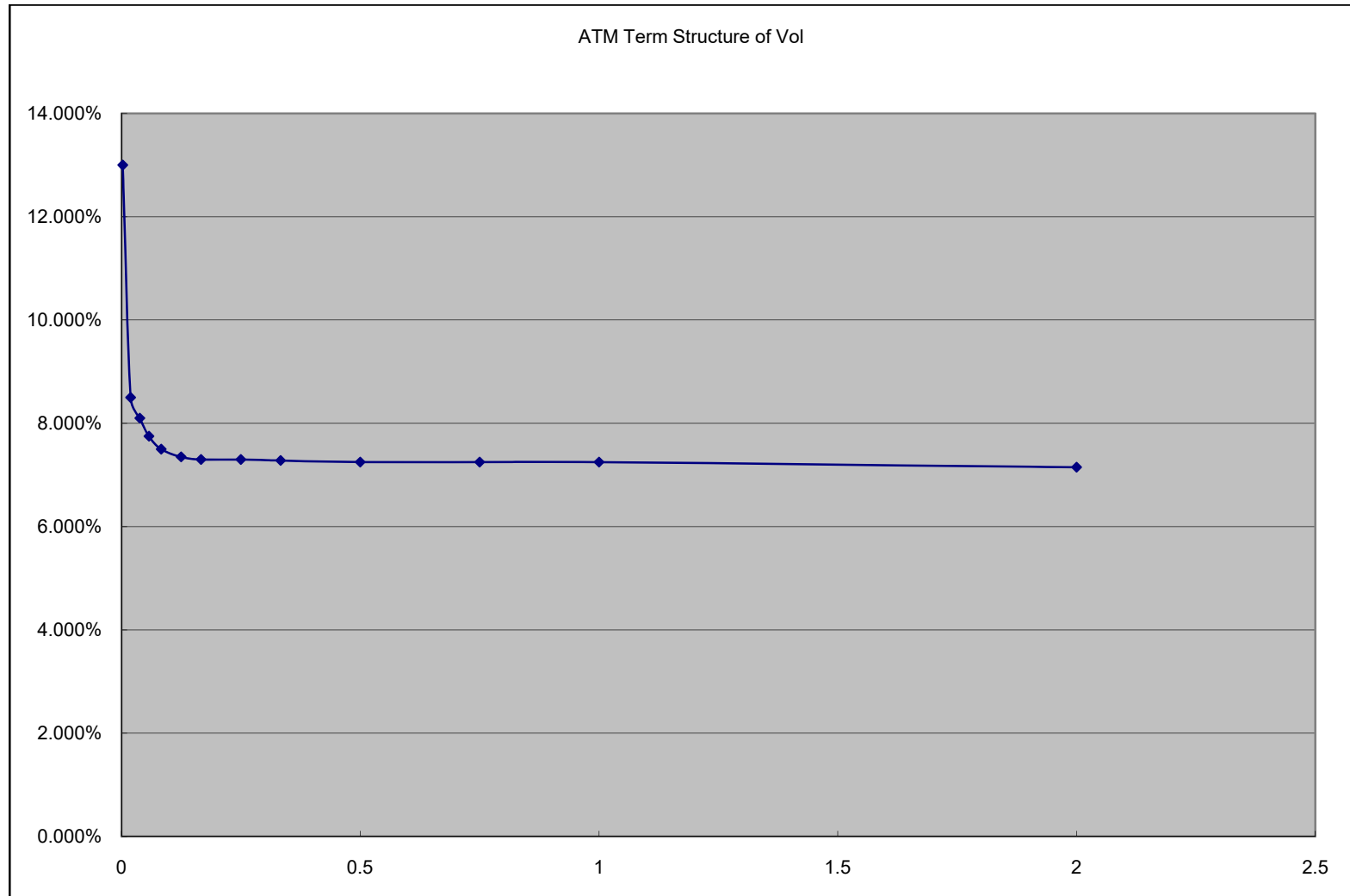


◆ 波動性 v.s. 執行價格(USDJPY, 2007/7/11)



### (三)Term Structure

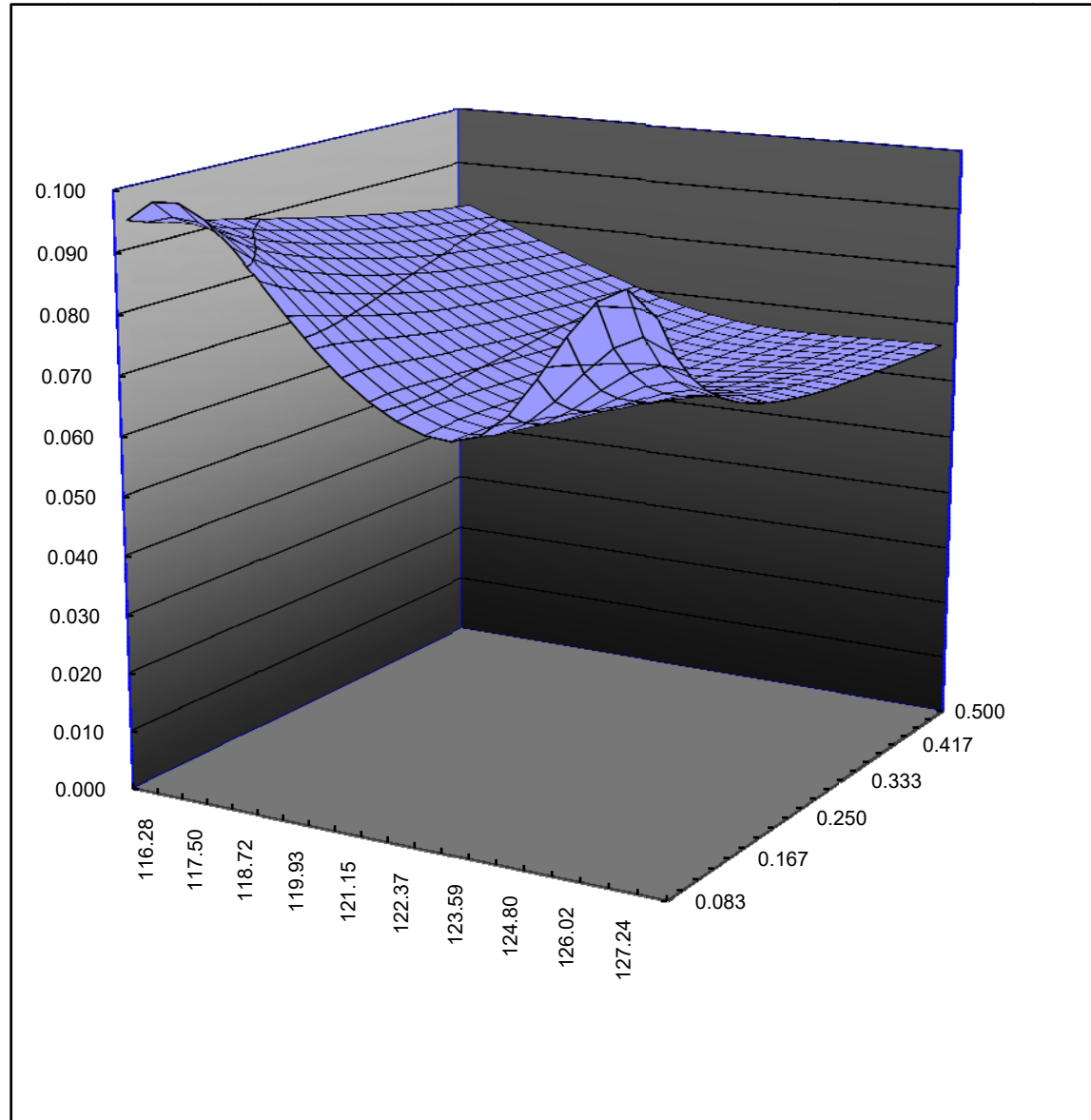
#### ◆ ATM Term Structure of Vol(USDJPY, 2007/7/11)



## (四)Surface

◆ 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



## (五)市場資料校準

◆ (1.3.4)、(1.3.5)、(1.3.6)式中隨機過程中的參數，必需使用市場參數估計之。

- 由於市場上乃以 Black-Scholes 模型來報價，因此我們須先以 BS 模型計算選擇權的權利金，

$$BSC(S_t, K, T-t, \sigma_M, r_t, y_t) = BSC(\sigma_M)$$

✓  $\sigma_M(K)$  為市場上的波動性報價，為執行價格的函數。

- 根據(1.3.7)式與(1.3.4)、(1.3.5)、(1.3.6)式，可知 Heston 模型的選擇權權利金可表示為，

$$HC(S_t, K, T-t, V_t, r_t, y_t, \kappa^*, \theta^*, \sigma, \rho) = HC(V_t, \kappa^*, \theta^*, \sigma, \rho)$$

設定下面目標函購，假設市場上有 n 個選擇權報價，以隨機過程中的參數為控制變數。

$$\min_{V_t, \kappa^*, \theta^*, \sigma, \rho} \left( \sum_{i=1}^n \left( BSC_i(\sigma_M) - HC_i(V_t, \kappa^*, \theta^*, \sigma, \rho) \right)^2 \right) \dots\dots\dots (1.10)$$

- 利用非線性最適化演算法，如 Powell 法，求得控制變數之最佳解。
- 可使用模擬退火法(Simulated Annealing)，避免局部最佳解。

◆ 使用 2007/7/11 USD/JPY 市場資訊，

- 1M、2M、3M、6M 四個時點。
- 10D Call、25D Call、ATM、25D Put、10D Put 五個 Strikes。
- 求得數值如下，

$$V_t = 0.0061126543, \theta^* = 0.0072726465, \sigma = 0.2639879042,$$

$$\kappa^* = 2.0675040055, \rho = -0.5363162751。$$

- 誤差值為 0.00787682644。

◆ 注意一：在外匯市場中，交易員的報價是報出隱含波動率，外匯選擇權的執行價格和 Delta 與市場的隱含波動率報價有關，

- 其中關於 Delta 的定義，在特定的情況(e.g. Currency Pair)下，還必須進行權利金調整(Premium Adjustment)，
- 不同的 Delta 定義會得到不同的執行價，這是市場上的慣例，
  - ✓ 也是市場的參與者所必須要知道的。
  - ✓ Spot Delta 與 Forward Delta。

◆ Delta Convention :

- 關於 Delta 是否要加入權利金調整項，一個簡單的通則，
  - ✓ 若是權利金的計價幣別(Premium Currency)為外國貨幣時，Delta 必須考慮權利金的調整項。

◆ 舉例而言，考慮一個 USD/ZAR 的 Call，Premium Currency 為 USD，

- 假設  $S=11.8303$ ， $K=14.3169$ ， $T=1$ ， $r_d=6.521\%$ ， $r_f=0.345\%$ ， $\sigma=19.15\%$ ，帶入 BS 公式，可得到  $C=0.358731$ ，此時調整前的 Delta 為 0.280674。
- 此例中，USD/ZAR 的匯率是以 ZAR 來衡量，但是 Premium Currency 是 USD。依照上述的通則，Delta 必須進行權利金的調整。
  - ✓ 買權的權利金為 0.358731 "ZAR"，但交易雙方實際收付的貨幣為 USD。
  - ✓ 對於買方而言，必須支付相當於 0.358731 "ZAR" 的 "USD" 給賣方。以交易當時的匯率  $S=11.8303$  還換算，相當於 0.030323 的 USD。
  - ✓ 換句話說，買方必須借入 0.030323 的 USD 來支付權利金，即相當於買方建立了一個 0.030323 單位的 USD "負" 部位。
- 考慮調整前 Delta 的涵義，購入此買權相當於持有 0.280674 單位的 USD "正" 部位。買方在購入此買權後，USD 的淨部位應該為  $0.280674 - 0.030323$ ，約略等於 0.25。
  - ✓ 買方的調整後 Delta 應約為 0.25。此時的執行價格  $X=14.3169$ ，即為 25D Call 的執行價格。



◆ 注意二：假設市場上可觀察到  $n$  筆選擇權的權利金報價(e.g.  $n=25$ )，並以 Heston 隨機過程中的 5 個參數  $(V_t, \kappa^*, \theta^*, \sigma, \rho)$  為控制變數。

➤ 就單獨一筆選擇權而言，Heston 模型的誤差定義可以有兩種，Vol 誤差與 Premium 誤差：

$$\text{Case(1)} \quad \text{Error}_i = \sigma_{M_i} - \sigma_{IV\_HC_i},$$

✓ 對第  $i$  筆選擇權來說，誤差  $\text{Error}_i$  在 Case(1)中定義為“市場報價  $\sigma_{M_i}$  與利用 Heston 模型算出的理論價依 BS 公式反推回的  $IV(\sigma_{IV\_HC_i})$  間的差距”。

$$\text{Case(2)} \quad \text{Error}_i = \frac{(BSC_i(\sigma_{M_i}) - HC_i(V_t, \kappa^*, \theta^*, \sigma, \rho))}{BSC_i(\sigma_{M_i})}。$$

✓ 對第  $i$  筆選擇權來說，誤差  $\text{Error}_i$  在 Case(2)中定義為“市場報價權利金與利用 Heston 模型算出的理論價間的(百分)差距”。

## 二、目標贖回型遠期契約(TRF)契約規格

### (一)商品定義

◆ 現行的目標贖回型遠期契約(TRF)的標的資產為一個參考指標匯率，

➤ 如同一般型的遠期契約一樣，交易雙方在到期日有義務依一個事先約定的執行價格進行交割與結算，

✓ 但和一般型遠期契約不同的是，TRF 具有多重的期數與比價日，

◆ 除了目標獲利贖回(Target Profit Redemption)的條款以外，TRF 也會包括許多特別的交易條件。

➤ Knock In：生效價位。標的資產的即期價位若於指定比價時點觸及此 Knock In 價位，則交易契約由無效狀態進入生效狀態，交易雙方才開始負有交割義務。

✓ 指定比價時點通常又可區分為美式 AKI 或歐式 EKI。

➤ Knock Out：出局價位。標的資產的即期價位若於指定比價時點觸及此 Knock Out 價位，則交易契約失效，交易雙方不再負有交割義務。

✓ 指定比價時點通常又可區分為美式 AKO 或離散式 DKO。

- **Target Profit**：目標獲利贖回。TRF 結構中，具有多重的比價日期，例如每月比價一次，每次比價結果如果投資人是處於獲利，則累加每次獲利金額(點數、百分比)。一旦累加達到目標獲利時，則交易自動提前結束；若未達到目標獲利時，則交易仍將繼續比價，直到最後一期到期日為止。
  - ✓ 這種 Target Profit 形式有時也稱為 Target Accumulator。
- **Target Loss**：目標損失贖回。Target Profit 的附屬條款，為避免投資人損失超過承受能力，在 Target Profit 之外，又規定 Target Loss，每次比價結果如果投資人是處於損失，則累加每次損失金額(點數、百分比)。
  - ✓ 當投資人損失累計達到目標損失時(不計入獲利)，則交易自動提前結束。
- **Gearing Ratio**：槓桿倍數。和 KOF 相似，TRF 結構中，投資人通常同時具有買入選擇權(付權利金)與賣出選擇權(收權利金)的策略，以收取的權利金去融通付出的權利金。為讓收付權利金兩者相當，除了調整成分選擇權的履約價位之外，還可以透過調整成分選擇權名日本金的方法，達到零成本權利金的效果。
  - ✓ 賣出選擇權的名日本金與買入選擇權名日本金的比例，稱為槓桿倍數，實務上通常是 2 倍。
  - ✓ 投資人使用槓桿倍數之優點，是可以得到一個比較具有優勢的履約價位，其缺點是投資人比價損失時的金額，名日本金是以比價獲利時的 2 倍計算。



## (二)商品介紹

### ◆ Target Forward

- 可分為看多型與看空型，期初發行人與投資人約定名目本金、履約價、槓桿倍數及目標出場(Target-Out Event)點數等相關交易條件，於約定之日期表進行比價，例如：每月比價一次。
- (1) 看多型：
  - ✓ 若即期比價匯率高於或等於履約價，投資人於相對應交割日以履約價買入名目本金。
  - ✓ 若即期比價匯率低於履約價，投資人於相對應交割日以履約價買入槓桿倍數的名目本金。
  - ✓ 於到期日前，若投資人累積獲利達到出場目標，則此交易提前終止。
- (2) 看空型：
  - ✓ 若即期比價匯率低於或等於履約價，投資人於相對應交割日以履約價賣出名目本金。
  - ✓ 若即期比價匯率高於履約價，投資人於相對應交割日以履約價賣出槓桿倍數的名目本金。
  - ✓ 於到期日前，若投資人累積獲利達到出場目標，則此交易提前終止。

## ◆ Target Forward with EKI

- 可分為看多型與看空型，發行人期初與投資人除了約定名目本金、履約價、槓桿倍數及目標出場點數等相關交易條件之外，另約定歐式觸發生效價 EKI，於約定之日期表進行比價。
- (1) 看多型：
  - ✓ 若即期比價匯率高於或等於履約價，投資人於相對應交割日以履約價買入名目本金。
  - ✓ 若即期比價匯率低於履約價且高於或等於 EKI，雙方無交割。
  - ✓ 若即期比價匯率低於 EKI，投資人於相對應交割日以履約價買入槓桿倍數的名目本金。
  - ✓ 於到期日前，若投資人累積獲利達到出場目標，則此交易提前終止。
- (2) 看空型：
  - ✓ 若即期比價匯率低於或等於履約價，投資人於相對應交割日以履約價賣出名目本金。
  - ✓ 若即期比價匯率高於履約價且低於或等於 EKI，雙方無交割。
  - ✓ 若即期比價匯率高於 EKI，投資人於相對應交割日以履約價賣出槓桿倍數的名目本金。
  - ✓ 於到期日前，若投資人累積獲利達到出場目標，則此交易提前終止。

## ◆ Target Forward with EKI and Loss Limit

- 可分為看多型與看空型，發行人期初除與投資人約定名目本金、履約價、歐式觸發生效價 EKI、槓桿倍數、獲利出場點數等相關交易條件之外，更加入損失出場點數，於約定之日期表進行比價。
- (1) 看多型：
  - ✓ 若即期比價匯率高於或等於履約價，投資人於相對應交割日以履約價買入名目本金。
  - ✓ 若即期比價匯率低於履約價且高於或等於 EKI，雙方無交割。
  - ✓ 若即期比價匯率低於 EKI，投資人於相對應交割日以履約價買入槓桿倍數的名目本金。
  - ✓ 於到期日前，若投資人累積獲利達到獲利出場目標，則此交易提前終止。
  - ✓ 於到期日前，若投資人累積損失達到損失出場目標，則此交易提前終止。
- (2) 看空型：
  - ✓ 若即期比價匯率低於或等於履約價，投資人於相對應交割日以履約價賣出名目本金。
  - ✓ 若即期比價匯率高於履約價且低於或等於 EKI，雙方無交割。
  - ✓ 若即期比價匯率高於 EKI，投資人於相對應交割日以履約價賣出槓桿倍數的名目本金。
  - ✓ 於到期日前，若投資人累積獲利達到獲利出場目標，則此交易提前終止。
  - ✓ 於到期日前，若投資人累積虧損達到損失出場目標，則此交易提前終止。

## ◆ Bonus Target Forward

- 可分為看多型與看空型，發行人期初除與投資人約定名目本金、履約價、槓桿倍數、目標出場次數等相關交易條件，亦約定紅利金額，於約定之日期表進行比價。
- (1) 看多型：
  - ✓ 若即期比價匯率高於或等於履約價，投資人於相對應交割日收取紅利金額。
  - ✓ 若即期比價匯率低於履約價，投資人於相對應交割日以履約價買入槓桿倍數的名目本金。
  - ✓ 於每一比價日，若即期比價匯率高於或等於履約價累積次數達到目標出場次數，投資人於相對應交割日收取紅利金額且本交易將提前中止。
- (2) 看空型：
  - ✓ 若即期比價匯率低於或等於履約價，投資人於相對應交割日收取紅利金額。
  - ✓ 若即期比價匯率高於履約價，投資人於相對應交割日以履約價賣出槓桿倍數的名目本金。
  - ✓ 於每一比價日，若即期比價匯率低於或等於履約價累積次數達到目標出場次數，投資人於相對應交割日收取紅利金額且本交易將提前中止。



## ◆ Bonus Target Forward with EKI

- 可分為看多型與看空型，發行人期初除了與投資人約定名目本金、履約價、槓桿倍數、紅利金額及目標出場次數等相關交易條件外，亦加入歐式觸發生效價 EKI，於約定之日期表進行比價。
- (1) 看多型：
  - ✓ 若即期比價匯率高於或等於履約價，投資人於相對應交割日收取紅利金額。
  - ✓ 若即期比價匯率低於履約價且高於或等於 EKI，雙方無交割。
  - ✓ 若即期比價匯率低於 EKI，投資人於相對應交割日以履約價買入槓桿倍數的名目本金。
  - ✓ 於每一比價日，若即期比價匯率高於或等於履約價累積次數達到目標出場次數，投資人於相對應交割日收取紅利金額且本交易將提前中止。
- (2) 看空型：
  - ✓ 若即期比價匯率低於或等於履約價，投資人於相對應交割日收取紅利金額。
  - ✓ 若即期比價匯率高於履約價且低於或等於 EKI，雙方無交割。
  - ✓ 若即期比價匯率高於 EKI，投資人於相對應交割日以履約價賣出槓桿倍數的名目本金。
  - ✓ 於每一比價日，若即期比價匯率低於或等於履約價累積次數達到目標出場次數，投資人於相對應交割日收取紅利金額且本交易將提前中止。

## ◆ Bonus Target Forward with EKI and Loss Limit

- 和 Bonus Target Forward with EKI 相似，但多了損失出場點數的條款，也就是說，於到期日前，若投資人累積損失達到損失出場目標，則此交易提前終止。

## ◆ Pivot Target Forward

- 發行人期初與投資人約定名目本金、槓桿倍數及目標出場點數等相關交易條件，同時並約定高履約價(High Strike Rate)、低履約價(Low Strike Rate)以及樞紐價(Pivot Rate)，依照約定之日期表進行比價，通常是每月比價一次。
  - ✓ 若即期比價匯率高於高履約價，投資人於相對應交割日以高履約價賣出槓桿倍數的名目本金。
  - ✓ 若即期比價匯率高於樞紐價且低於或等於高履約價，投資人於相對應交割日以高履約價賣出名目本金。
  - ✓ 若即期比價匯率高於或等於低履約價且低於樞紐價，投資人於相對應交割日以低履約價買入名目本金。
  - ✓ 若即期比價匯率低於低履約價，投資人於相對應交割日以低履約價買入槓桿倍數的名目本金。
  - ✓ 於到期日前，若投資人累積獲利達到出場目標，則此交易提前終止。

## ◆ Pivot Target Forward with EKI

➤ 發行人期初除與投資人約定名目本金、高履約價、低履約價、樞紐價、槓桿倍數及目標出場點數等相關交易條件外，亦約定高觸發生效價(High European Knock-In Barrier, HEKI ) 及低觸發生效價(Low European Knock-In Barrier, LEKI )，於約定之日期表進行比價。

- ✓ 若即期比價匯率高於 HEKI，投資人於相對應交割日以高履約價賣出槓桿倍數的名目本金。
- ✓ 若即期比價匯率高於高履約價且低於或等於 HEKI，雙方無交割。
- ✓ 若即期比價匯率高於樞紐價且低於或等於高履約價，投資人於相對應交割日以高履約價賣出名目本金。
- ✓ 若即期比價匯率高於或等於低履約價且低於樞紐價，投資人於相對應交割日以低履約價買入名目本金。
- ✓ 若即期比價匯率低於低履約價且高於或等於 LEKI，雙方無交割。
- ✓ 若即期比價匯率低於 LEKI，投資人於相對應交割日以低履約價買入槓桿倍數的名目本金。
- ✓ 於到期日前，若投資人累積獲利達到出場目標，則此交易提前終止。

## ◆ Pivot Target Forward with EKI and Loss Limit

- 和 Pivot Target Forward with EKI 相似，但多了損失出場點數的條款，也就是說，於到期日前，若投資人累積損失達到損失出場目標，則此交易提前終止。

## ◆ Pivot Bonus Target Forward

- 發行人期初與投資人約定名目本金、高履約價、低履約價、目標出場次數及紅利金額等相關交易條件，於約定之日期表進行比價。
  - ✓ 若即期比價匯率高於高履約價，投資人於相對應交割日以高履約價賣出槓桿倍數的名目本金。
  - ✓ 若即期比價匯率高於或等於低履約價且低於或等於高履約價，投資人於相對應交割日收取紅利金額。
  - ✓ 若即期比價匯率低於低履約價，投資人於相對應交割日以低履約價買入槓桿倍數的名目本金。
  - ✓ 於每一比價日，若即期比價匯率高於或等於低履約價且低於或等於高履約價之累積次數達到目標出場次數，投資人於相對應交割日收取紅利金額且本交易將提前中止。

## ◆ Pivot Bonus Target Forward with EKI

- 發行人期初除了與投資人約定名目本金、高履約價、低履約價、目標出場次數及紅利金額等相關交易條件之外，另約定高觸發生效價 HEKI 及低觸發生效價 LEKI，於約定之日期表進行比價。
  - ✓ 若即期比價匯率高於 HEKI，投資人於相對應交割日以高履約價賣出槓桿倍數的名目本金。
  - ✓ 若即期比價匯率高於高履約價且低於 HEKI，雙方無交割。
  - ✓ 若即期比價匯率高於或等於低履約價且低於或等於高履約價，投資人於相對應交割日收取紅利金額。
  - ✓ 若即期比價匯率低於低履約價且高於 LEKI，雙方無交割。
  - ✓ 若即期比價匯率低於 LEKI，投資人於相對應交割日以低履約價買入槓桿倍數的名目本金。
  - ✓ 於每一比價日，若即期比價匯率高於或等於低履約價且低於或等於高履約價之累積次數達到目標出場次數，投資人於相對應交割日收取紅利金額且本交易將提前中止。

## ◆ Pivot Bonus Target Forward with EKI and Loss Limit

- 和 Pivot Bonus Target Forward with EKI 相似，但多了損失出場點數的條款，也就是說，於到期日前，若投資人累積損失達到損失出場目標，則此交易提前終止。



### (三)商品範例

◆ 由第二部分的商品介紹可知，TRF 有著多樣的形式與交易條件，以下我們將針對報酬函數最為複雜的 Pivot Target Forward with EKI，以一個實例說明於下：

- 比價天期與頻率(Expiries)：一年期交易，每個月比價一次，共 12 次比價
- 幣別(CCY)：USD/JPY
- 即期參考價(Spot Ref)：101.50
- 名目本金(Notional Amount)：每次比價 USD \$100,000 (比價獲利時採用)；USD 200,000 (比價損失時採用)
- 槓桿倍數(Gearing Ratio)：2
- 比價匯率(Fixing Rate)頁面：東京時間下午三點；路透社頁面：TKFE
- 高歐式觸發生效價(High EKI Barrier)：109.00
- 高履約價(High Strike)：107.00
- 樞紐價(Pivot)：101.00
- 低履約價(Low Strike)：95.00

■ 低歐式觸發生效價(Low EKI Barrier)：93.00

■ 出場目標(Target Profit)：800 pips Full (外匯點數 800 點，即每 1 美元累積獲利達 8 日圓)

■ 價內事件(ITM Event)：若高履約價(107.00)≥比價匯率≥低履約價(95.00)，分為以下兩種情形：

(1)若比價匯率<樞紐價(101.00)，投資人買 USD 100,000 相對於 JPY@ 95.00， $ITM = \text{Max}(0, \text{比價匯率} - 95.00)$

(2)若比價匯率≥樞紐價(101.00)，投資人賣 USD 100,000 相對於 JPY@ 107.00， $ITM = \text{Max}(0, 107.00 - \text{比價匯率})$

■ 目標出場事件(Target-Out Event)：每當價內事件發生時，計算獲利並加總累積獲利點數，當每 1 美元累積獲利高於或等於 800 點時，本交易提前出場，交易雙方將不再有任何義務

■ 比價情境：每次比價時（若尚未發生目標出場事件，即交易尚未提前終止）

若比價匯率> 109.00	投資人需賣 USD \$200,000 相對於 JPY@107.00
若 $101.00 \leq \text{比價匯率} \leq 107.00$	投資人需賣 USD \$100,000 相對於 JPY@107.00，並計算獲利點數
若 $95.00 \leq \text{比價匯率} < 101.00$	投資人需買 USD \$100,000 相對於 JPY@95.00，並計算獲利點數
若比價匯率< 93.00	投資人需買 USD \$200,000 相對於 JPY@95.00
其餘情況交易雙方無交割	

### 三、觸及失效型遠期契約(KO Forward)契約規格

#### (一)商品定義

◆ 現行的觸及失效型遠期契約(KOF)的標的資產為一個參考指標匯率，

- 交易雙方在到期日有義務依一個事先約定的履約價(Strike)進行交割與結算，
- 但和一般型遠期契約不同的是，KOF 通常具有多重的期數與比價日，除了觸及失效的條款以外，KOF 也會包括許多特別的交易條件。
  - ✓ Knock In：觸及生效。標的資產的即期價位若於指定比價時點觸及此 Knock In 價位，則交易契約由無效狀態進入生效狀態，交易雙方才開始負有交割義務。依其指定比價時點不同，通常又可區分為美式(American Knock In, AKI)或歐式(European Knock In, EKI)。Knock In 條款的設定很彈性，可以在不同的期別有不同的 AKI 或 EKI；如同其名所言，AKI 表示在相應的期別中，只要即期價位一旦觸及此 AKI 價位，則契約立時生效，相對地，EKI 表示在相應的期別中，若在指定的比價時點即期價位觸及此 EKI 價位，契約才生效。
  - ✓ Knock Out：觸及失效。標的資產的即期價位若於指定比價時點觸及此 Knock Out 價位，則交易契約失效，交易雙方不再負有交割義務。依其指定比價時點不同，通常又可區分為美式(American Knock Out, AKO)或離散式(Discrete Knock Out, DKO)。Knock Out 條款的設定通常是針對整個契約，而非指單期，因此 AKO 表示在契約簽立生效後，一旦即期價位觸及此 AKO 價位，則剩餘契約皆失效，而 DKO 則針對整個契約中，指定的比價

時點做觀察，若即期價位於指定的觀察時點觸及此 DKO 價位，則剩餘契約失效。在此，使用 Discrete 表示在整個"契約"中的指定觀察點是多次的，有別於 European 是用來形容在一個特定"期別"之中，表示在這一期中只有一個觀察點。

- ✓ Gearing Ratio：槓桿倍數。KOF 中，投資人通常同時具有買入選擇權(付權利金)與賣出選擇權(收權利金)的部位，而以收取的權利金去融通付出的權利金。為讓收付兩者相當，除了調整成分選擇權的履約價位之外，還可以調整成分選擇權的名目本金，來達到零成本權利金的效果。賣出選擇權的名目本金與買入選擇權名目本金的比例，即稱為槓桿倍數，實務上通常是 2 倍。投資人使用槓桿倍數的優點，是可以得到一個比較具有優勢的履約價，其缺點是投資人比價損失時的金額，名目本金是以比價獲利時的 2 倍計算。
- ✓ Reset Provision：可重設條款。指 Strike、Knock In、Knock Out 等價位，會因為即期價位觸及特定條件而發生改變，產生重新設定的現象。通常重設的結果可讓投資人轉換為較為有利的狀態。
- ✓ Call Provision：可提前贖回條款。此條款使發行人可以主動在約定的日期(Call date)行使權利，提前結束交易。此條款通常用來保護發行人，發行人將以某種型式付出權利金。
- ✓ Put Provision：可提前賣回條款。此條款使投資人可以主動在約定的日期(Put date)行使權利，提前結束交易。此條款通常用來保護投資人，投資人將以某種型式付出權利金。

## (二)商品介紹

### ◆ Knock-Out Forward

- 發行人期初與投資人約定名目本金、履約價、槓桿倍數及觸發失效價(Discrete Knock-Out Barrier, DKO<sup>1</sup>)等相關交易條件，於約定之日期表(Fixing Schedule)進行比價，例如：每月比價一次。
- 看多型：
  - ✓ 若即期比價匯率高於或等於履約價，投資人於相對應交割日以履約價買入名目本金。
  - ✓ 若即期比價匯率低於履約價，投資人於相對應交割日以履約價買入槓桿倍數的名目本金。
  - ✓ 於到期日前，每一比價日，比價匯率高於或等於 DKO，投資人以履約價買入名目本金，且本交易提前出場，交易雙方將不再有任何義務。
- 看空型：
  - ✓ 若即期比價匯率低於或等於履約價，投資人於相對應交割日以履約價賣出名目本金。
  - ✓ 若即期比價匯率高於履約價，投資人於相對應交割日以履約價賣出槓桿倍數的名目本金。
  - ✓ 於到期日前，每一比價日，比價匯率低於或等於 DKO，投資人以履約價賣出名目本金，且本交易提前出場，交易雙方將不再有任何義務。

---

<sup>1</sup> 若觸及失效條款為連續比價，則觸發失效價應記為 AKO，但在不影響閱讀性及造成混淆的假設下，文中皆以 DKO 表示觸發失效價。

## ◆ Knock-Out Forward with EKI

- 發行人期初除與投資人約定名目本金、履約價、槓桿倍數、觸發失效價等相關交易條件外，另約定歐式觸發生效價(European Knock-In Barrier, EKI)，於約定之日期表進行比價，亦分為看多型與看空型兩種。
- 看多型：
  - ✓ 若即期比價匯率高於或等於履約價，投資人於相對應交割日以履約價買入名目本金。
  - ✓ 若即期比價匯率低於履約價且高於或等於 EKI 雙方無交割。
  - ✓ 若即期比價匯率低於 EKI，投資人於相對應交割日以履約價買入槓桿倍數的名目本金。
  - ✓ 於到期日前，每一比價日，比價匯率高於或等於 DKO，投資人買入名目本金，且本交易提前出場，交易雙方將不再有任何義務。
- 看空型：
  - ✓ 若即期比價匯率低於或等於履約價，投資人於相對應交割日以履約價賣出名目本金。
  - ✓ 若即期比價匯率高於履約價且低於或等於 EKI 雙方無交割。
  - ✓ 若即期比價匯率高於 EKI，投資人於相對應交割日以履約價賣出槓桿倍數的名目本金。
  - ✓ 於到期日前，每一比價日，比價匯率低於或等於 DKO，投資人賣出名目本金，且本交易提前出場，交易雙方將不再有任何義務。

## ◆ Bonus Knock-Out Forward

➤ 發行人期初除與投資人約定名目本金、履約價、槓桿倍數、觸發失效價等相關交易條件外，另約定紅利金額(Bonus)，於約定之日期表進行比價，分為看多型與看空型兩種。

➤ 看多型：

- ✓ 若即期比價匯率低於履約價，投資人於相對應交割日以履約價買入槓桿倍數的名目本金。
- ✓ 若即期比價匯率高於或等於履約價，投資人於相對應交割日以履約價買入名目本金或收取紅利金額。
- ✓ 於到期日前，每一比價日，比價匯率高於或等於 DKO，(投資人收取紅利金額<sup>2</sup>，)本交易提前出場，交易雙方將不再有任何義務。

➤ 看空型：

- ✓ 若即期比價匯率高於履約價，投資人於相對應交割日以履約價賣出槓桿倍數的名目本金。
- ✓ 若即期比價匯率低於或等於履約價，投資人於相對應交割日以履約價賣出名目本金或收取紅利金額。
- ✓ 於到期日前，每一比價日，比價匯率低於或等於 DKO，(投資人收取紅利金額<sup>3</sup>，)本交易提前出場，交易雙方將不再有任何義務。

---

<sup>2</sup> 視不同交易契約條款而定。

<sup>3</sup> 視不同交易契約條款而定。



## ◆ Bonus Knock-Out Forward with EKI

- 發行人期初與投資人約定名目本金、履約價、槓桿倍數、紅利金額、歐式觸發生效價及觸發失效價等相關交易條件，於約定之日期表進行比價。
- 看多型：
  - ✓ 若即期比價匯率低於履約價且高於或等於 EKI 雙方無交割。
  - ✓ 若即期比價匯率低於 EKI，投資人於相對應交割日以履約價買入槓桿倍數的名目本金。
  - ✓ 若即期比價匯率高於或等於履約價，投資人於相對應交割日以履約價買入名目本金或收取紅利金額。
  - ✓ 於到期日前，每一比價日，比價匯率高於或等於 DKO，(投資人收取紅利金額<sup>4</sup>，)本交易提前出場，交易雙方將不再有任何義務。
- 看空型：
  - ✓ 若即期比價匯率高於履約價且低於或等於 EKI 雙方無交割。
  - ✓ 若即期比價匯率高於 EKI，投資人於相對應交割日以履約價賣出槓桿倍數的名目本金。
  - ✓ 若即期比價匯率低於或等於履約價，投資人於相對應交割日以履約價賣出名目本金或收取紅利金額。
  - ✓ 於到期日前，每一比價日，比價匯率低於或等於 DKO，(投資人收取紅利金額<sup>5</sup>，)本交易提前出場，交易雙方將不再有任何義務。

---

<sup>4</sup> 視不同交易契約條款而定。

<sup>5</sup> 視不同交易契約條款而定。



### (三)商品範例

◆ 由上述可知，KOF 有許多種形式與交易條件，可以配合投資人的需要做交易條件的調整，  
以下我們將針對交易條件最為複雜的 Bonus Knock-Out Forward with EKI，以一個實例說明  
於下：

- 交易類型：看多型
- 比價天期與頻率(Expiries)：二年期交易，每個月比價一次，共 24 次比價
- 幣別(Direction and CCY)：Buy USD/JPY
- 即期參考價(Spot Ref)：95.57
- 名目本金(Notional Amount)：每次比價 USD \$150,000 (比價獲利時採用)；USD \$300,000 (比價損失時採用)
- 槓桿倍數(Leverage Ratio)：2
- 紅利金額(Bonus)：USD \$1,500
- 比價匯率(Fixing Rate)頁面：東京時間下午三點；路透社頁面：TKFE
- 履約價(Strike)：94.00

- 歐式觸發生效價(EKI)：88.00，每期皆相同
- 觸發失效價(AKO)：103.50
- 觸發失效事件(AKO Event)：自交易開始起任何時間，若美元兌日圓即期匯率高於或等於 103.50，則視為觸發失效事件發生，投資人收取紅利金額 USD \$1,500，且本交易提前出場，交易雙方將不再有任何義務。
- 比價情境：每次比價時（若尚未發生觸發失效事件，即交易尚未提前終止）

若比價匯率 $\geq$ 94.00	投資人需買 USD 150,000 相對於 JPY@94.00
若 $88.00 \leq$ 比價匯率 $< 94.00$	交易雙方無交割
若比價匯率 $< 88.00$	投資人需買 USD 300,000 相對於 JPY@94.00

## 四、實作案例四

### 市場校準的幾種可行方式

- 1.使用 Excel Stand Alone Spread Sheet 。
- 2.使用 R 最適化套件。
- 3.使用 C#控制 Excel Spread Sheet 。
- 4.使用 C#搭配 R.Net 。
- 5.使用 C#搭配 QuantLib C++版本。
- 6.使用 C#搭配 QuantLib C#版本。

# (一)參數校準：使用 Excel Stand Alone Spread Sheet

## 方法一：Excel Spread Sheet

Microsoft Excel - heston93_Calib20131114JPVOL.xls																
輸入需要解答的問題																
B5	1															
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
1	X(1)	X(2)	X(3)	X(4)	X(5)	X(6)	Y( )	C(1)	C(2)	C(3)	C(4)	C(5)				
2								0.080000	0.080000	1.000000	2.000000	0.000000	guess value	Heston Calib	Mon	
3								0.084664	0.083361	2.902726	4.684232	-0.116401	calibrated result			
4	Vanilla vol	Phi (1 for call, -1 for put)	Spot	Strike	TTM in years	rd	rf	BS Value	Initial Variance	LongRun Variance	Vol Of Variance	Mean Reversion	Correlation	Market Price Of Vol Risk	Heston Vanilla	BS implies vol
5	23.20%	1	100.0000	96.0000	0.5000	2.00%	0.00%	9.156	0.084664	0.083361	2.902726	4.684232	-0.116401	0	9.173	0.233
6	22.83%	1	100.0000	98.0000	0.5000	2.00%	0.00%	7.937	0.084664	0.083361	2.902726	4.684232	-0.116401	0	7.909	0.227
7	22.51%	-1	100.0000	100.0000	0.5000	2.00%	0.00%	5.827	0.084664	0.083361	2.902726	4.684232	-0.116401	0	5.796	0.224
8	22.24%	-1	100.0000	102.0000	0.5000	2.00%	0.00%	6.803	0.084664	0.083361	2.902726	4.684232	-0.116401	0	6.797	0.222
9	22.02%	1	100.0000	104.0000	0.5000	2.00%	0.00%	4.926	0.084664	0.083361	2.902726	4.684232	-0.116401	0	4.974	0.222
10	24.26%	1	100.0000	96.0000	1.0000	2.00%	0.00%	12.613	0.084664	0.083361	2.902726	4.684232	-0.116401	0	12.594	0.242
11	23.53%	1	100.0000	98.0000	1.0000	2.00%	0.00%	11.284	0.084664	0.083361	2.902726	4.684232	-0.116401	0	11.329	0.236
12	23.00%	-1	100.0000	100.0000	1.0000	2.00%	0.00%	8.109	0.084664	0.083361	2.902726	4.684232	-0.116401	0	8.074	0.229
13	23.08%	1	100.0000	102.0000	1.0000	2.00%	0.00%	9.197	0.084664	0.083361	2.902726	4.684232	-0.116401	0	9.234	0.232
14	23.71%	1	100.0000	104.0000	1.0000	2.00%	0.00%	8.588	0.084664	0.083361	2.902726	4.684232	-0.116401	0	8.530	0.236
15																
16																
17																
18																
19																
20																
21																
22																
23																
24																
25																
26																
27																
28																
29																
30																
31																
32																
33																
34																
35																
36																
37																
38																
39																
40																
41																
42																
43																
44																
45																
46																
47																
48																
49																
50																
51																
52																
53																
54																
55																
56																
57																
58																
59																
60																
61																
62																
63																
64																
65																
66																
67																
68																
69																
70																
71																
72																
73																
74																
75																
76																
77																
78																
79																
80																
81																
82																
83																
84																
85																
86																
87																
88																
89																
90																
91																
92																
93																
94																
95																
96																
97																
98																
99																
100																

在按btn(Heston Calib)之前，需先mark選取 RANGE(B5:H14)的範圍，代表你所選取的Option樣本

◆ 使用方式：見 Spread Sheet。

## (二)參數校準：使用 R 最適化套件。

◆ R 語言內建了非線性最適化的運算功能，這在模型參數的校準上非常有用。

- R 提供了 `nlm()`與 `optim()`這兩個函數，讓我們進行未受限的最適化。
- 使用者可以選擇使用 BFGS、共軛梯度、退火模擬等方法，進行計算。

### ◆ 簡單最適化案例

- 我們先介紹 `nlm()`的使用，`nlm()`需要傳入一個目標函數，起始向量，以及目標函數中的參數值。令目標函數如下，

$$g(x_1, x_2, x_3) = \sin(x_1) - \sin(x_2 - A) + x_3^2 + B$$

- 其中， $(x_1, x_2, x_3)$  為搜尋的變數值， $A$ 、 $B$  為參數。我們希望在給定  $A$ 、 $B$  參數值下，找到使目標函數值最小化的變數值。

$$\min_{(x_1, x_2, x_3)} g(x_1, x_2, x_3 \mid A, B)$$

```
> g<- function(x, A, B) {  
+ out <- sin(x[1]) - sin(x[2]-A) + x[3]^2 + B  
+ return(out) }  
> results <- nlm(g, c(1,2,3), A=4, B=2)  
> results  
$minimum  
[1] 6.497025e-13  
$estimate  
[1] -1.570797e+00 -7.123895e-01 -4.990548e-07  
$gradient  
[1] 5.300922e-08 -1.465494e-08 1.776357e-09  
$code  
[1] 1  
$iterations  
[1] 12
```

程式列表 4.2

- 上面程式先建立目標函數  $g$ ，呼叫  $nlm()$  時，需傳入目標函數  $g$ ，猜測的初始值向量  $c(1, 2, 3)$ ，以及給定的  $A$ 、 $B$  參數值。
- 極小值為  $6.497025 \times 10^{-13}$ ，估計的最適解為  $(-1.570797, -0.7123895, -4.990548 \times 10^{-7})$ 。



## ◆ 受限最適化與懲罰函數法

➤ 對於受限函数的最適化，一個簡單的做法就是在未受限的目標函数，加上懲罰函数。

✓ 所謂的懲罰函数就是針對超出限制範圍的變數，給予目標函数偏離極大的數值。

✓ 舉例而言，如果上例中我們希望  $x_2 \geq 0$ ，我們可以設定目標函数為，

$$f(x_1, x_2, x_3) = g(x_1, x_2, x_3) + 1000000 \times (-x_2) \text{ , if } x_2 < 0$$

$$f(x_1, x_2, x_3) = g(x_1, x_2, x_3) \text{ , if } x_2 \geq 0$$

$$\min_{(x_1, x_2, x_3)} f(x_1, x_2, x_3 \mid A, B)$$

➤ 如果  $x_2 < 0$ ，則懲罰函数將產生一個很大的正數，此很大的正數將不利於目標函数的極小化。

```
> h <- function(x, A, B) {  
+ if(x[2]>=0) res <- g(x, A, B) else res <- g(x, A, B)-1000000*x[2]  
+ return(res) }  
> results <- nlm(h, c(1,2,3), A=4, B=2)
```

```
> results
$minimum
[1] 0.3501565
$estimate
[1] -1.106889e+00 7.153626e-06 3.557386e-02
$gradient
[1] 0.44744613 0.65364941 0.071114872
$iterations
[1] 26
> g(c(1, 1, 1), A=4, B=2)
[1] 3.982591
```

程式列表 4.3

- 上面程式中，我們要求第二個變數的解要大於 0，的最適估計值為 0.3501565。估計的最適解為  $(-1.106889, 7.153626 \times 10^{-6}, 3.557386 \times 10^{-2})$ 。

### (三)參數校準：使用 C#控制 Excel spread sheet

◆ Read Excel File in C# , <https://coderwall.com/p/app3ya/read-excel-file-in-c> 。

- .NET 4+ allows C# to read and manipulate Microsoft Excel files, for computers that have Excel installed (if you do not have Excel installed, see NPOI).
- First, add the reference to Microsoft Excel XX.X Object Library, located in the COM tab of the Reference Manager. I have given this the using alias of Excel.

```
using Excel = Microsoft.Office.Interop.Excel;    //Microsoft Excel 14 object in references-> COM tab
```

- Next, you'll need to create references for each COM object that is accessed. Each reference must be kept to effectively exit the application on completion.

```
//Create COM Objects. Create a COM object for everything that is referenced
Excel.Application xlApp = new Excel.Application();
Excel.Workbook xlWorkbook = xlApp.Workbooks.Open(@"sandbox_test.xlsx");
Excel._Worksheet xlWorksheet = xlWorkbook.Sheets[1];
Excel.Range xlRange = xlWorksheet.UsedRange;
```

- Then you can read from the sheet, keeping in mind that indexing in Excel is not 0 based. This just reads the cells and prints them back just as they were in the file.

```
//iterate over the rows and columns and print to the console as it appears in the file
//excel is not zero based!!
for (int i = 1; i <= rowCount; i++)
{
    for (int j = 1; j <= colCount; j++)
    {
        //new line
        if (j == 1)
            Console.WriteLine("\r\n");

        //write the value to the console
        if (xlRange.Cells[i, j] != null && xlRange.Cells[i, j].Value2 != null)
            Console.Write(xlRange.Cells[i, j].Value2.ToString() + "\t");

    }
}
```

- Lastly, the references to the unmanaged memory must be released. If this is not properly done, then there will be lingering processes that will hold the file access writes to your Excel workbook.

```
//cleanup
GC.Collect();
GC.WaitForPendingFinalizers();

//rule of thumb for releasing com objects:
// never use two dots, all COM objects must be referenced and released individually
// ex: [something].[something].[something] is bad

//release com objects to fully kill excel process from running in the background
Marshal.ReleaseComObject(xlRange);
Marshal.ReleaseComObject(xlWorksheet);

//close and release
xlWorkbook.Close();
Marshal.ReleaseComObject(xlWorkbook);

//quit and release
xlApp.Quit();
Marshal.ReleaseComObject(xlApp);
```

## ◆ Full Code:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using Excel = Microsoft.Office.Interop.Excel;           //microsoft Excel 14 object in references-> COM tab

namespace Sandbox
{
    public class Read_From_Excel
    {
        public static void getExcelFile()
        {
            //Create COM Objects. Create a COM object for everything that is referenced
            Excel.Application xlApp = new Excel.Application();
            Excel.Workbook xlWorkbook = xlApp.Workbooks.Open(@"C:\Users\E56626\Desktop\Teddy\VS2012\
                Sandbox\sandbox_test - Copy - Copy.xlsx");
            Excel._Worksheet xlWorksheet = xlWorkbook.Sheets[1];
            Excel.Range xlRange = xlWorksheet.UsedRange;
        }
    }
}
```

```
int rowCount = xlRange.Rows.Count;
int colCount = xlRange.Columns.Count;

//iterate over the rows and columns and print to the console as it appears in the file
//excel is not zero based!!
for (int i = 1; i <= rowCount; i++)
{
    for (int j = 1; j <= colCount; j++)
    {
        //new line
        if (j == 1)
            Console.Write("\r\n");

        //write the value to the console
        if (xlRange.Cells[i, j] != null && xlRange.Cells[i, j].Value2 != null)
            Console.Write(xlRange.Cells[i, j].Value2.ToString() + "\t");
    }
}

//cleanup
GC.Collect();
GC.WaitForPendingFinalizers();

//rule of thumb for releasing com objects:
// never use two dots, all COM objects must be referenced and released individually
// ex: [something].[something].[something] is bad
```

```
//release com objects to fully kill excel process from running in the background
Marshal.ReleaseComObject(xlRange);
Marshal.ReleaseComObject(xlWorksheet);

//close and release
xlWorkbook.Close();
Marshal.ReleaseComObject(xlWorkbook);

//quit and release
xlApp.Quit();
Marshal.ReleaseComObject(xlApp);
}
}
}
```



◆ C#呼叫 Excel VBA 的 Code , <http://www.cnblogs.com/heekui/archive/2008/03/30/1129355.html> 。

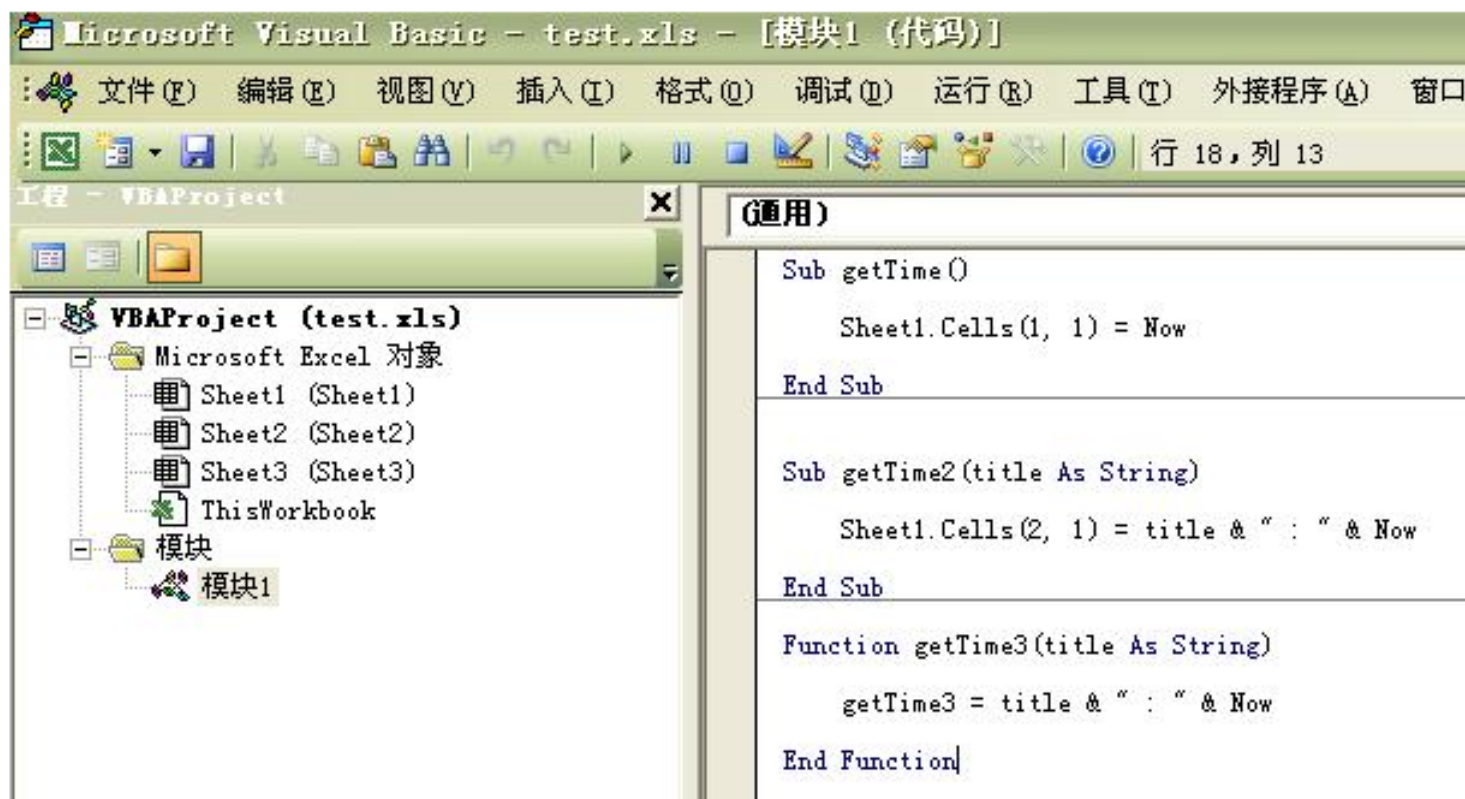
➤ 有時需要在 C#中執行 Excel VBA Macro , 甚至還需要在執行 VBA 之後 , 獲取返回值 , 再進行相應的處理。

➤ 幫助類僅提供一個方法 : RunExcelMacro 。

✓ 參數說明 :

string	excelFilePath	Excel 文件路徑
string	macroName	Macro 名稱
object[]	parameters	Macro 參數組
out object	rtnValue	Macro 返回值
bool	isShowExcel	執行時是否顯示 Excel

- 補充說明：VBA Macro 需如下圖寫在模組中，才能被方法識別。寫在 ThisWorkbook 中不能被識別。



◆ 執行 Excel VBA Macro 幫助類，注釋已詳細。核心部分就是通過反射方式，呼叫 Excel VBA Macro，

➤ oBook.Save()很重要，否則即使執行了 VBA Macro，也不會保存 Excel 更改後的內容：

```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4 using Excel = Microsoft.Office.Interop.Excel;
5 using Microsoft.Office.Core;
6 using System.IO;
7
8 namespace DoVBAMacro
9 {
10     /// <summary>
11     /// 執行 Excel VBA 宏說明類
12     /// </summary>
13     public class ExcelMacroHelper
14     {
15         /// <summary>
16         /// 執行 Excel 中的宏
17         /// </summary>
18         /// <param name="excelFilePath">Excel 檔路徑</param>
19         /// <param name="macroName">宏名稱</param>
20         /// <param name="parameters">巨集引數組</param>
```

```
21    /// <param name="rtnValue">宏返回值</param>
22    /// <param name="isShowExcel">執行時是否顯示 Excel</param>
23    public void RunExcelMacro(
24        string excelFilePath,
25        string macroName,
26        object[] parameters,
27        out object rtnValue,
28        bool isShowExcel
29    )
30    {
31        try
32        {
33            #region 檢查入參
34
35            // 檢查檔是否存在
36            if (!File.Exists(excelFilePath))
37            {
38                throw new System.Exception(excelFilePath + " 文件不存在");
39            }
40
41            // 檢查是否輸入宏名稱
42            if (string.IsNullOrEmpty(macroName))
43            {
44                throw new System.Exception("請輸入宏的名稱");
45            }
46
```

```
47         #endregion
48
49         #region 調用宏處理
50
51         // 準備打開 Excel 文件時的缺省參數對象
52         object oMissing = System.Reflection.Missing.Value;
53
54         // 根據參數組是否為空，準備參數組物件
55         object[] paraObjects;
56
57         if (parameters == null)
58         {
59             paraObjects = new object[] { macroName };
60         }
61         else
62         {
63             // 巨集引數組長度
64             int paraLength = parameters.Length;
65
66             paraObjects = new object[paraLength + 1];
67
68             paraObjects[0] = macroName;
69             for (int i = 0; i < paraLength; i++)
70             {
71                 paraObjects[i + 1] = parameters[i];
72             }
73         }
74     }
75 }
```

```
73     }
74
75     // 創建 Excel 對象示例
76     Excel.ApplicationClass oExcel = new Excel.ApplicationClass();
77
78     // 判斷是否要求執行時 Excel 可見
79     if (isShowExcel)
80     {
81         // 使創建的物件可見
82         oExcel.Visible = true;
83     }
84
85     // 創建 Workbooks 對象
86     Excel.Workbooks oBooks = oExcel.Workbooks;
87
88     // 創建 Workbook 對象
89     Excel._Workbook oBook = null;
90
91     // 打開指定的 Excel 文件
92     oBook = oBooks.Open(
93         excelFilePath,
94         oMissing,
95         oMissing,
96         oMissing,
97         oMissing,
98         oMissing,
```

```
99         oMissing,
100         oMissing,
101         oMissing,
102         oMissing,
103         oMissing,
104         oMissing,
105         oMissing,
106         oMissing,
107         oMissing
108     );
109
110     // 執行 Excel 中的宏
111     rtnValue = this.RunMacro(oExcel, paraObjects);
112
113     // 保存更改
114     oBook.Save();
115
116     // 退出 Workbook
117     oBook.Close(false, oMissing, oMissing);
118
119     #endregion
120
121     #region 釋放對象
122
123     // 釋放 Workbook 對象
124     System.Runtime.InteropServices.Marshal.ReleaseComObject(oBook);
```

```
125         oBook = null;
126
127         // 釋放 Workbooks 對象
128         System.Runtime.InteropServices.Marshal.ReleaseComObject(oBooks);
129         oBooks = null;
130
131         // 關閉 Excel
132         oExcel.Quit();
133
134         // 釋放 Excel 對象
135         System.Runtime.InteropServices.Marshal.ReleaseComObject(oExcel);
136         oExcel = null;
137
138         // 調用垃圾回收
139         GC.Collect();
140
141         #endregion
142     }
143     catch (Exception ex)
144     {
145         throw ex;
146     }
147 }
148
149 /// <summary>
150 /// 執行宏
```



```

151     /// </summary>
152     /// <param name="oApp">Excel 對象</param>
153     /// <param name="oRunArgs">參數（第一個參數為指定宏名稱，後面為指定宏的參數值）</param>
154     /// <returns>宏返回值</returns>
155     private object RunMacro(object oApp, object[] oRunArgs)
156     {
157         try
158         {
159             // 聲明一個返回物件
160             object objRtn;
161
162             // 反射方式執行巨集
163             objRtn = oApp.GetType().InvokeMember(
164                                     "Run",
165                                     System.Reflection.BindingFlags.Default |
166                                     System.Reflection.BindingFlags.InvokeMethod,
167                                     null,
168                                     oApp,
169                                     oRunArgs
170                                 );
171
172             // 返回值
173             return objRtn;
174
175         }
176         catch (Exception ex)

```

```
177         {
178             // 如果有底層異常，拋出底層異常
179             if (ex.InnerException.Message.ToString().Length > 0)
180             {
181                 throw ex.InnerException;
182             }
183             else
184             {
185                 throw ex;
186             }
187         }
188     }
189 }
190 }
191
```

## ◆ 示例三個 VBA Macro 的方法：

```
1 Sub getTime()  
2  
3     Sheet1.Cells(1, 1) = Now  
4  
5 End Sub  
6  
7  
8 Sub getTime2(title As String)  
9  
10    Sheet1.Cells(2, 1) = title & " : " & Now  
11  
12 End Sub  
13  
14 Function getTime3(title As String) As String  
15  
16    getTime3 = title & " : " & Now  
17  
18 End Function  
19
```

## ◆ 對應的三個使用方法

- 1.不帶參數的巨集調用
- 2.帶參數的巨集調用
- 3.有返回值的巨集調用

```
1     private void btnExe_Click(object sender, EventArgs e)
2     {
3         try
4         {
5             // 返回對象
6             object objRtn = new object();
7
8             // 獲得一個 ExcelMacroHelper 物件
9             ExcelMacroHelper excelMacroHelper = new ExcelMacroHelper();
10
11            // 執行指定 Excel 中的巨集，執行時顯示 Excel
12            excelMacroHelper.RunExcelMacro(
13
14                @"E:\csharp_study\DoVBAMacro\test.xls",
15                "getTime2",
16                new Object[] { "現在時刻" },
17                out objRtn,
18                true
19            );
```

```
19
20 // 執行指定 Excel 中的巨集，執行時不顯示 Excel
21 excelMacroHelper.RunExcelMacro(
22     @"E:\csharp_study\DoVBAMacro\test.xls",
23     "getTime2",
24     new Object[] { "現在時刻" },
25     out objRtn,
26     false
27 );
28
29 // 執行指定 Excel 中的巨集，執行時顯示 Excel，有返回值
30 excelMacroHelper.RunExcelMacro(
31     @"E:\csharp_study\DoVBAMacro\test.xls",
32     "getTime3",
33     new Object[] { "現在時刻" },
34     out objRtn,
35     true
36 );
37
38 MessageBox.Show((string)objRtn);
39
40 }
41 catch(System.Exception ex)
42 {
43     MessageBox.Show(ex.Message);
```

```
44         }  
45     }
```

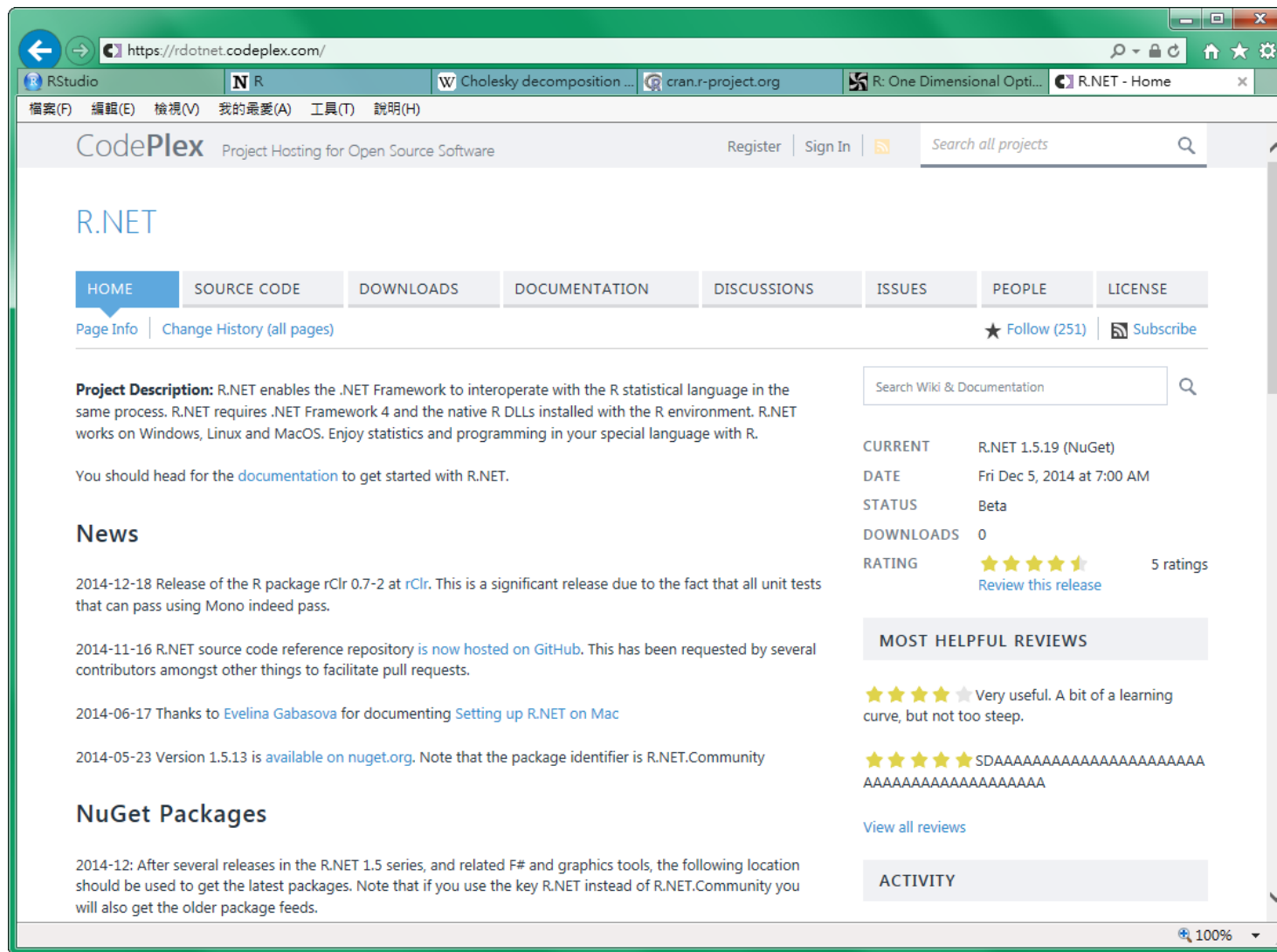
➤ 幫助類需添加引用：Microsoft Excel 11.0 Object Library

#### (四)參數校準：使用 C#搭配 R.Net。

◆ RdotNet 是一個使用 C#開發的中介引擎，它可以讓我們直接在.NET 的程式之中與電腦中已安裝的 R 程式互相溝通。

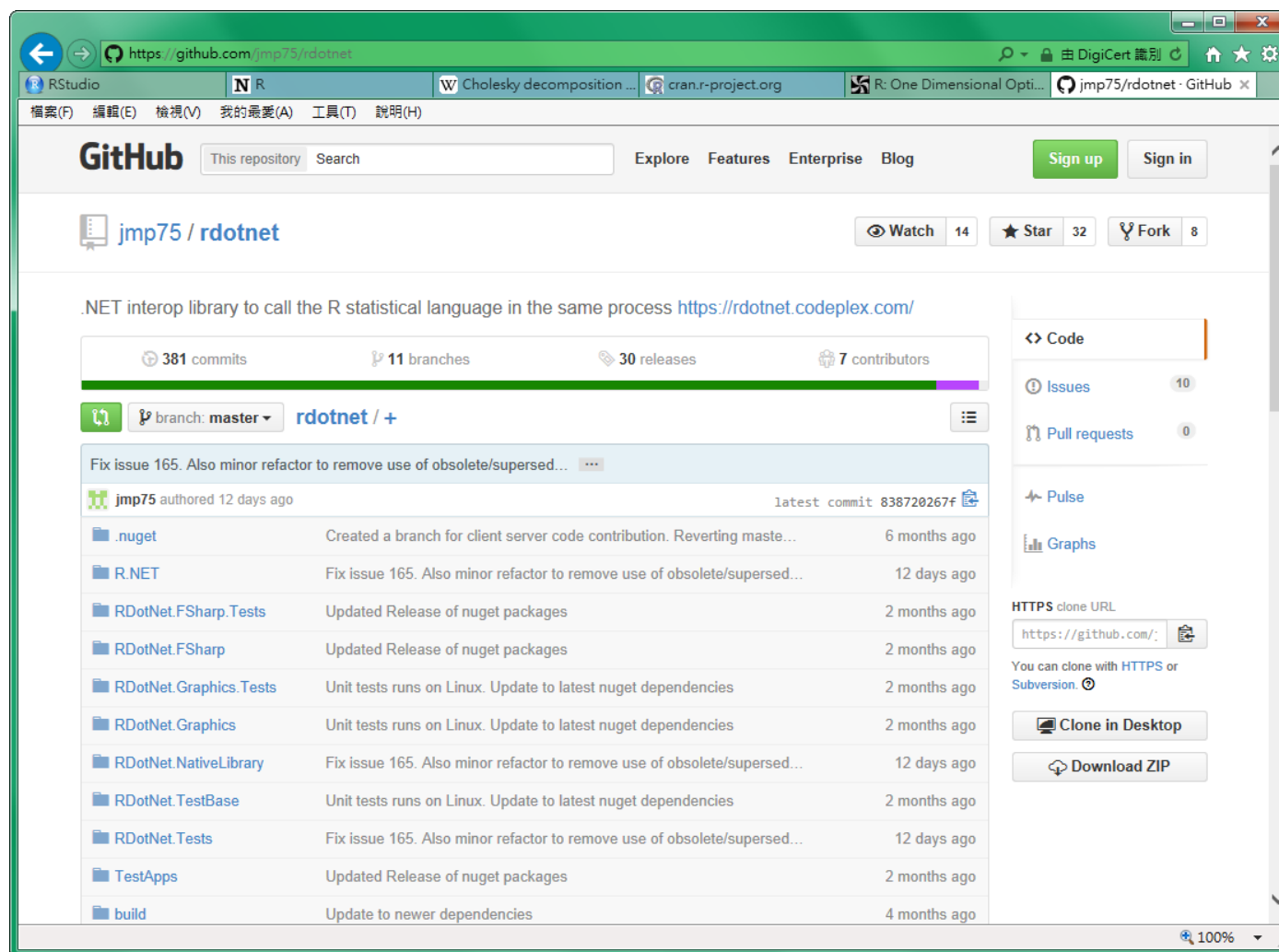
- 亦即，我們可以在 C#的程式之中，呼叫 R 程式幫我們進行運算，然後直接把結果取出，在進行 C# 下一步的運算。
- 如此，可以補足 C#程式庫的不足，我們也不需要自行撰寫複雜的矩陣與最適化運算程式。

◆ RdotNet 在 CodePlex 的網址為，<https://rdotnet.codeplex.com/>，網站如下圖。

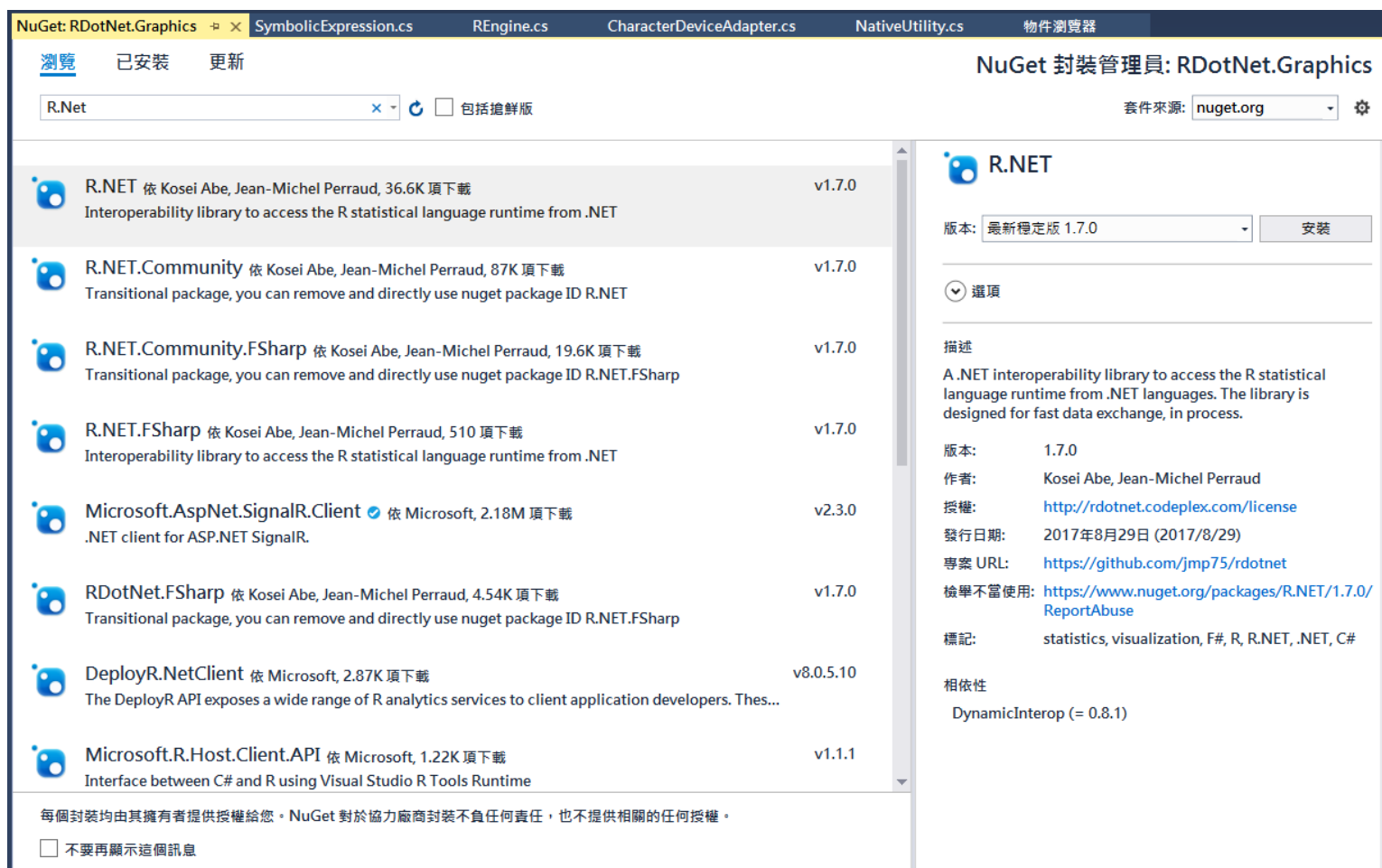




- ◆ 最新版的 RdotNet 為 1.7.0，讀者可以在 <https://github.com/jmp75/rdotnet> 取得最新版的 Source Code，自行編譯。



- ◆ 程式庫需要使用 Visual Studio 的 Package 管理工具 NuGet 來下載，在搜尋項目輸入 R.Net 便可找到相關專案。

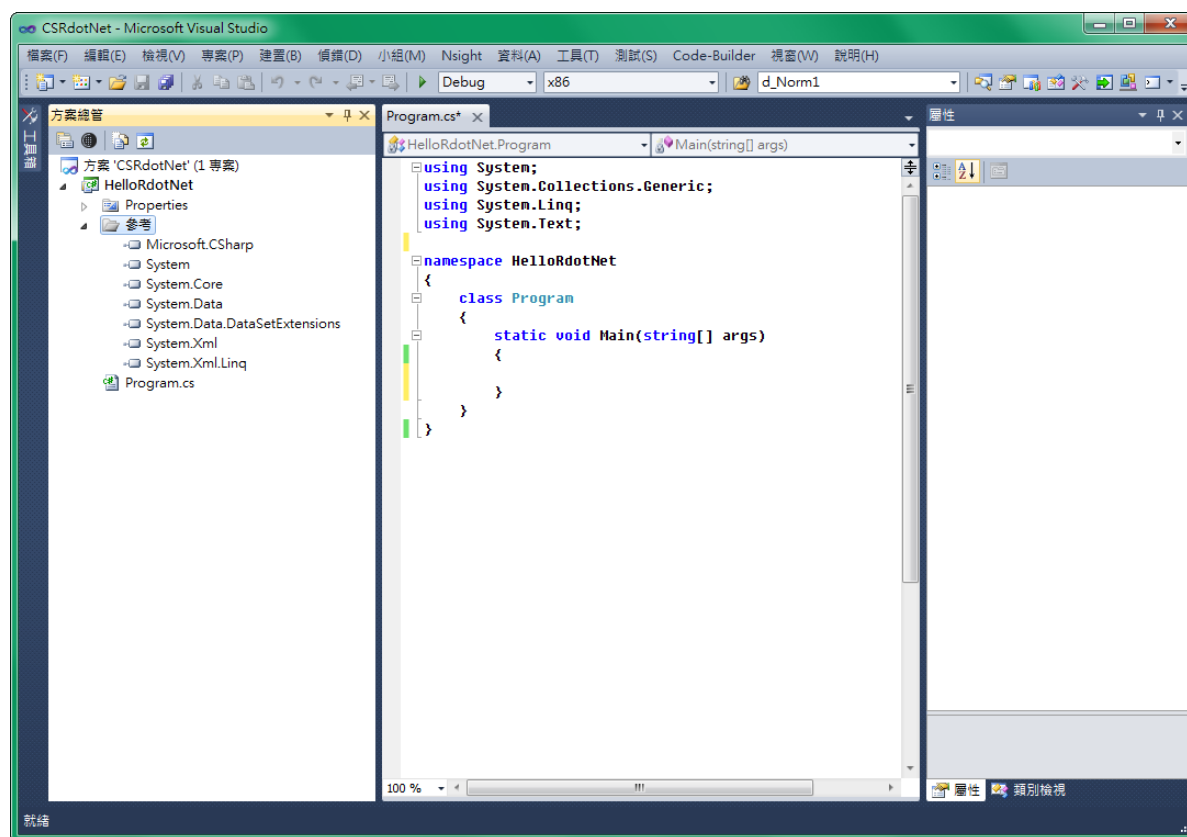


◆ 原來的網站為 <https://archive.codeplex.com/?p=rdotnet>，舊版程式庫已經被移除了。

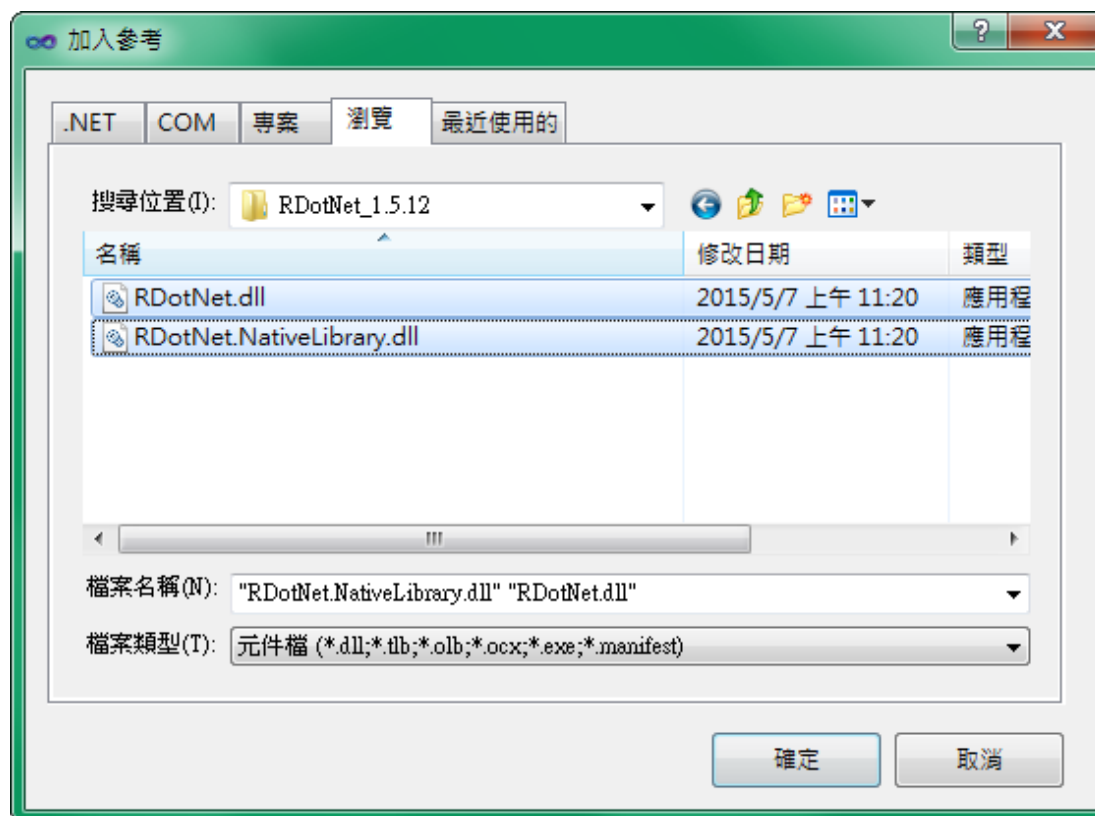
➤ 本書的範例是使用 RdotNet 1.5.12 版，搭配 Visual Studio 2010 以及 R 3.2.0 i386 版本。

## ◆ 一、Hello 專案

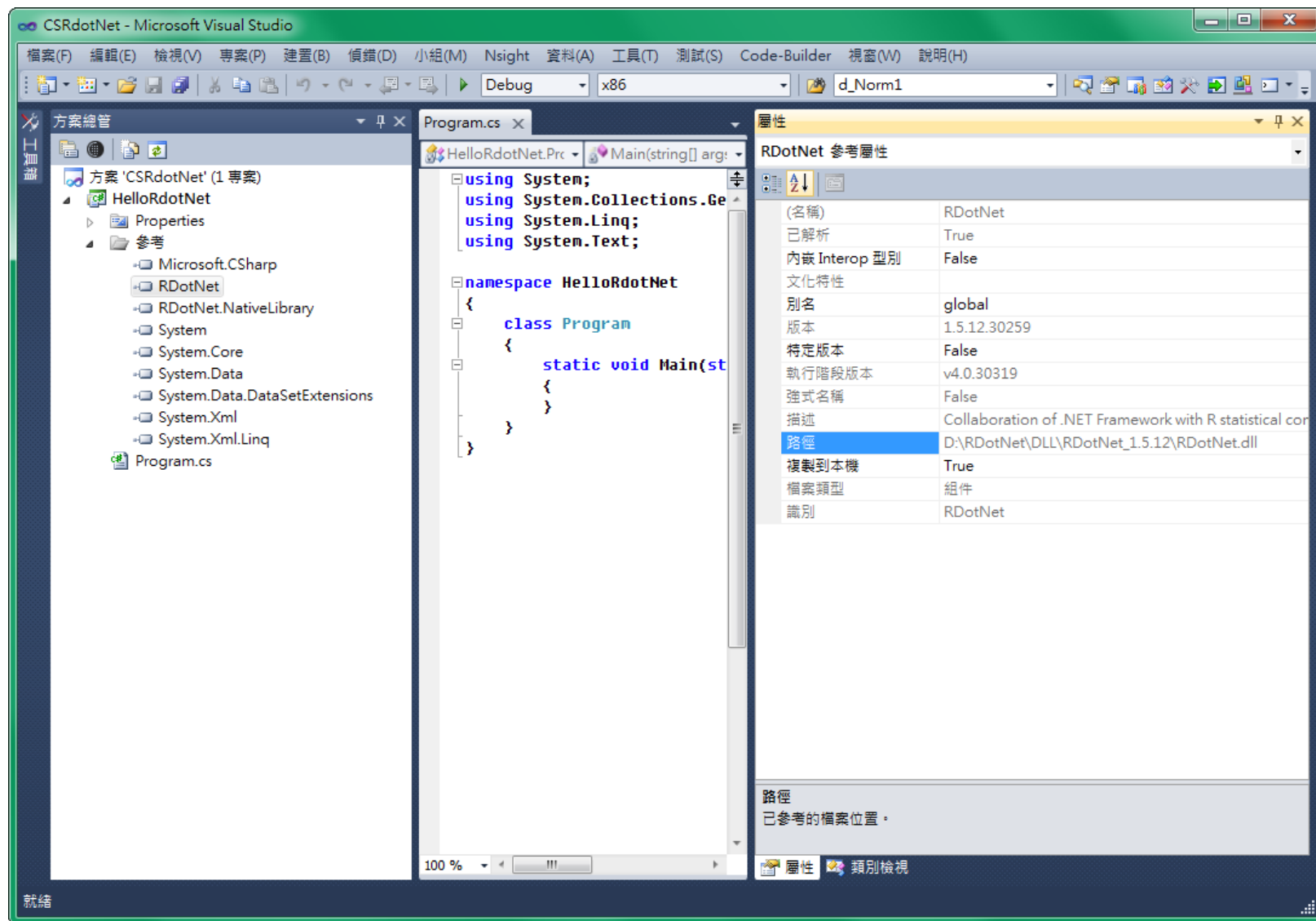
➤ 首先，建立一個主控台的新專案，如下圖。



- 點選右方方案總管內專案的參考項目，按滑鼠右鍵，選取加入參考。出現下方視窗，在瀏覽頁中，找到下載的 RdotNet 的 DLL 程式庫，將兩者都選取。



- 此時，參考中看到我們加入的兩個 DLL。



➤ 輸入下面程式碼，編譯專案。

```
#001 using System;
#002 using System.Collections.Generic;
#003 using System.Linq;
#004 using System.Text;
#005
#006 using RDotNet;
#007 using RDotNet.NativeLibrary;
#008
#009 namespace HelloRdotNet
#010 {
#011     class Program
#012     {
#013         static void Main(string[] args)
#014         {
#015             REngine engine = REngine.GetInstance();
#016             // A somewhat contrived but customary Hello World:
#017             CharacterVector charVec1 = engine.CreateCharacterVector(
#018                 new[] { "Hello, R world!, .NET speaking" });
#019             engine.SetSymbol("greetings", charVec1);
#020
#021             // print out in the console
#022             engine.Evaluate("str(greetings)");
#023             string[] a = engine.Evaluate("'Hi there .NET, from the R engine'")
#024                 .AsCharacter().ToArray();
```

```

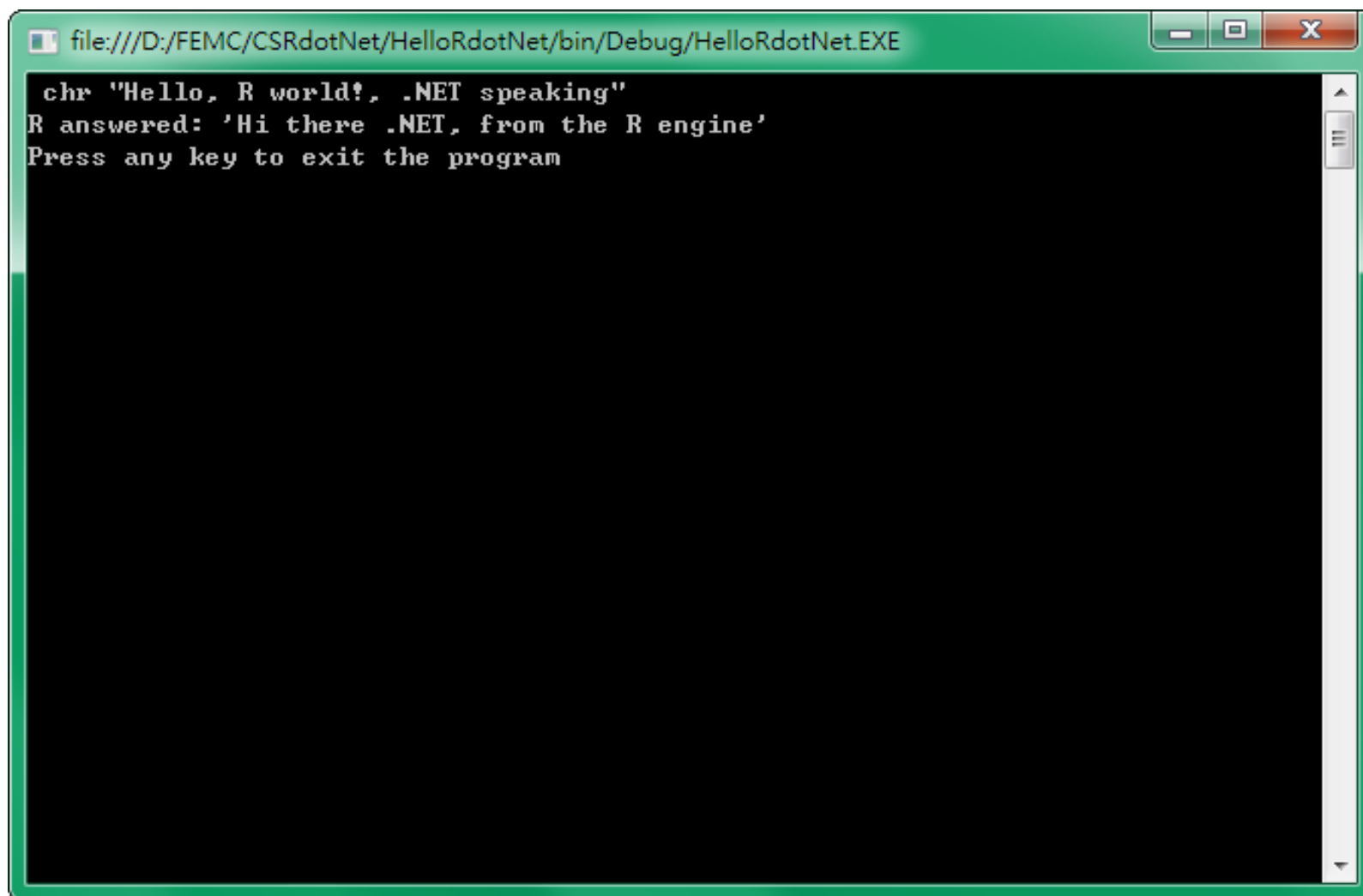
#025         Console.WriteLine("R answered: '{0}'", a[0]);
#026         Console.WriteLine("Press any key to exit the program");
#027         Console.ReadKey();
#028
#029         engine.Dispose();
#030     }
#031 }
#032 }

```

程式列表4.1

- ✓ #006、#007 先加入這兩個 DLL 的引用。#015 產生中介引擎。#017、#018 產生一個字元向量變數 charVec1，此字元向量是 R 對應的資料結構，用來儲存字元之用。#019 則將此字元向量變數命名為 greetings。greetings 是 R 中此字元向量的名稱，charVec1 則是 C#中此字元向量的名稱。
- ✓ #022 透過中介引擎，要求 R 執行 str(greetings)此指令，這相當於在 R 的主控制台提示符號 '>' 後，直接打入指令 str(greetings)。這就會在我們前面執行 C#執行檔後，輸出畫面的第一行文字，chr "Hello, R world!, .NET speaking"。
- ✓ #023 行要求 R 執行 'Hi there .NET, from the R engine'，並將輸出結果以字元方式取得，再轉成陣列儲存於變數 a 中。#025 將 a 的內容輸出，成為畫面的第二行文字，R answered: 'Hi there .NET, from the R engine'。

◆ 執行程式，輸出畫面如下。



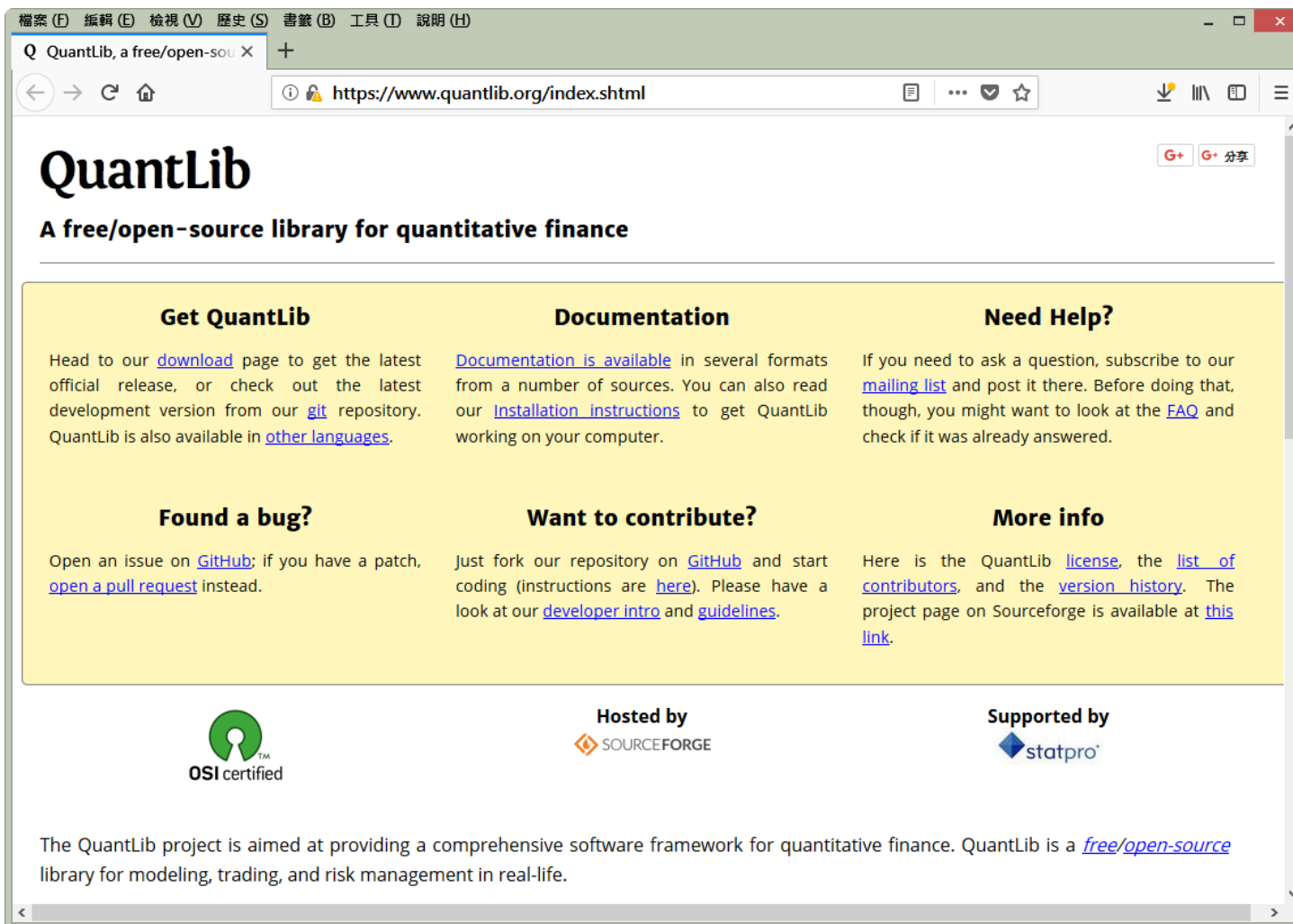
A screenshot of a Windows command prompt window. The title bar is green and contains the text "file:///D:/FEMC/CSRdotNet/HelloRdotNet/bin/Debug/HelloRdotNet.EXE" along with standard minimize, maximize, and close buttons. The command prompt area has a black background with white text. The text displayed is: "chr 'Hello, R world!, .NET speaking'", "R answered: 'Hi there .NET, from the R engine'", and "Press any key to exit the program". A vertical scrollbar is visible on the right side of the command prompt area.

```
file:///D:/FEMC/CSRdotNet/HelloRdotNet/bin/Debug/HelloRdotNet.EXE  
chr 'Hello, R world!, .NET speaking'  
R answered: 'Hi there .NET, from the R engine'  
Press any key to exit the program
```

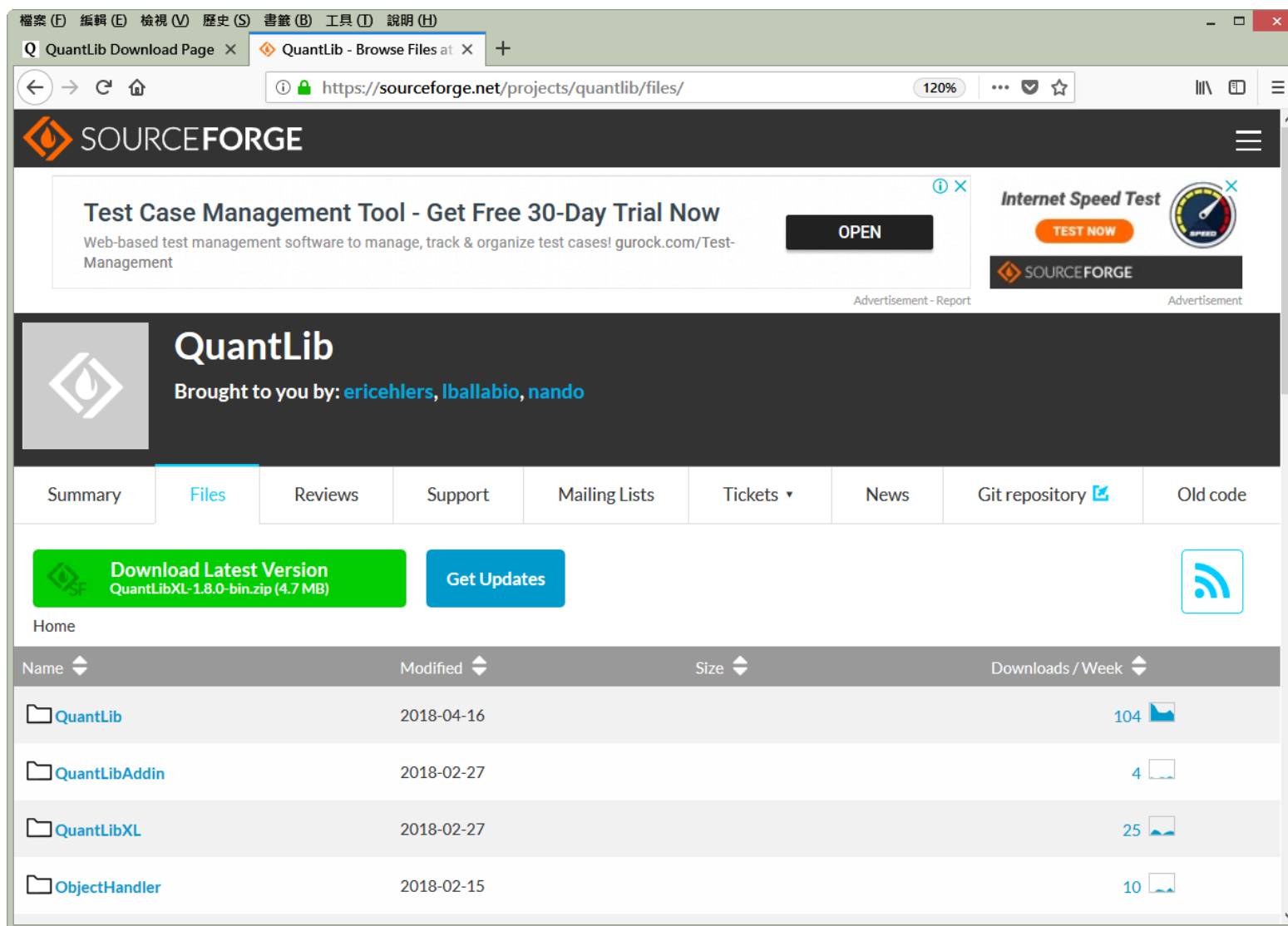


## (五)參數校準：使用 C#搭配 QuantLib C++版本。

### ◆ QuantLib 網址。



## ◆ SourceForge 下載網址



The screenshot shows the SourceForge website for the QuantLib project. The browser window has two tabs: 'QuantLib Download Page' and 'QuantLib - Browse Files at'. The address bar shows the URL 'https://sourceforge.net/projects/quantlib/files/'. The page features the SourceForge logo, a banner for a 'Test Case Management Tool', and an 'Internet Speed Test' advertisement. Below these, the QuantLib logo and the text 'Brought to you by: ericehlers, lballabio, nando' are displayed. A navigation bar includes links for Summary, Files (selected), Reviews, Support, Mailing Lists, Tickets, News, Git repository, and Old code. A green button 'Download Latest Version' (QuantLibXL-1.8.0-bin.zip (4.7 MB)) and a blue button 'Get Updates' are prominent. A table lists the available files:

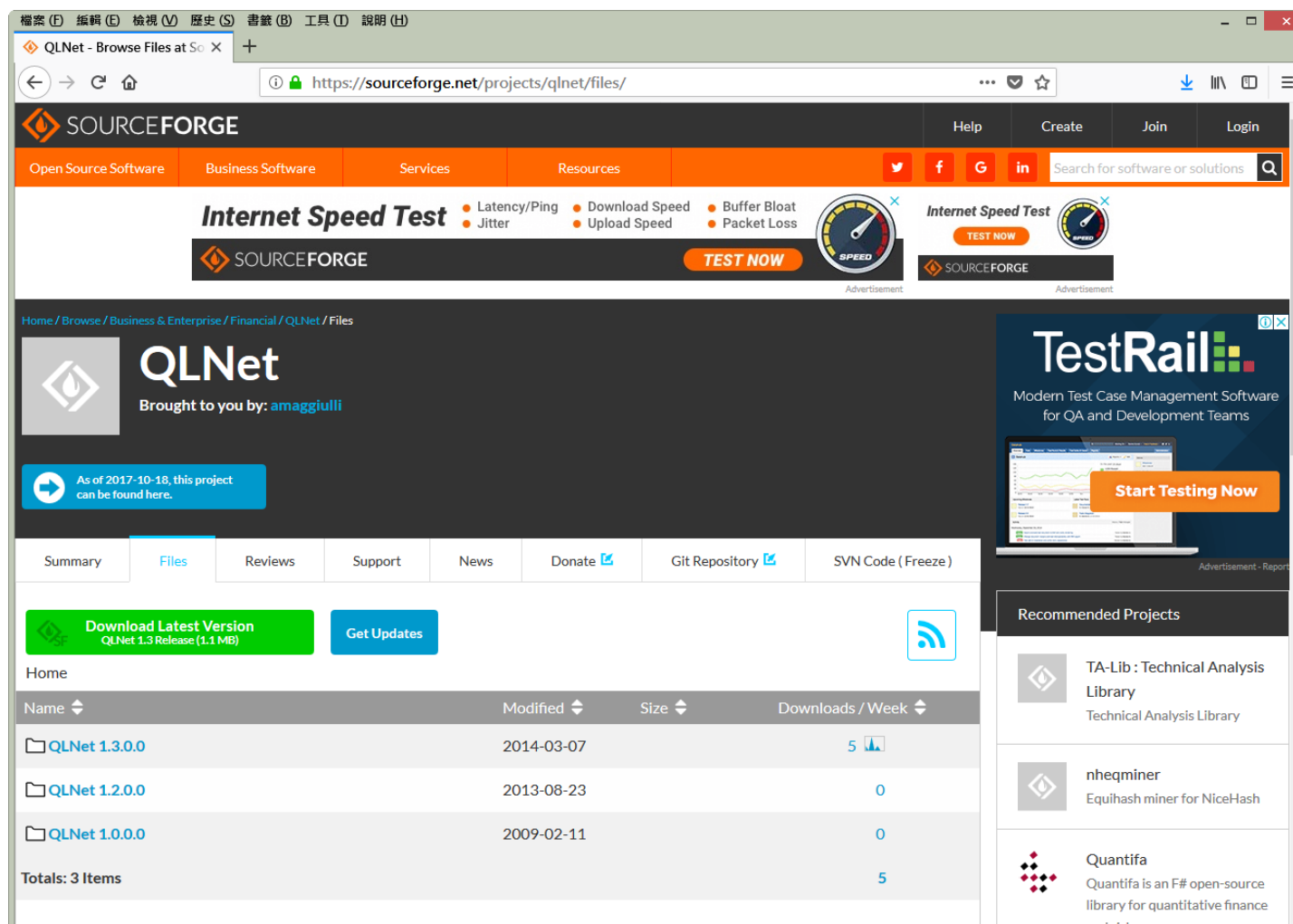
Name	Modified	Size	Downloads / Week
QuantLib	2018-04-16		104
QuantLibAddin	2018-02-27		4
QuantLibXL	2018-02-27		25
ObjectHandler	2018-02-15		10

◆ C#呼叫 C/C++動態程式庫(DLL)：請自行網上參考。

- "Using Intel Math Kernel Library and Intel Integrated Performance Primitives in the Microsoft.NET Framework", Intel Comp. Technical Paper。
- 精通.NET 互操作：P/Invoke, C++ Interop 和 COM Interop，黃際洲、崔曉源，人民郵電出版社，2009 五月。

## (六)參數校準：使用 C#搭配 QuantLib C#版本。

### ◆ 以 C#改寫 QuantLib C++程式碼



## ◆ 講師之前專案採用的方式。

- 最高的執行效率，比 C++ 版執行還快。
- 最好的擴充修改彈性。
- 100% 原生碼相容。
- 最佳的學習過程。
- 最花時間的方法。

## (七)參數校準：範例結果

◆ 評價日 2014/1/24，USD/JPY 看漲目標贖回型。

交易日：2014 年 1 月 24 日

即期匯率參考價：103.44

槓桿比率：2.00 倍

執行價匯率：99.90

觸及生效匯率：95.00

契約期限：一年

比價頻率：每月(12 次)

內在價值目標獲利：1 美元對應 4 日圓

## ◆ Heston 校準參數

Heston.Variance = 0.00770547621786487;

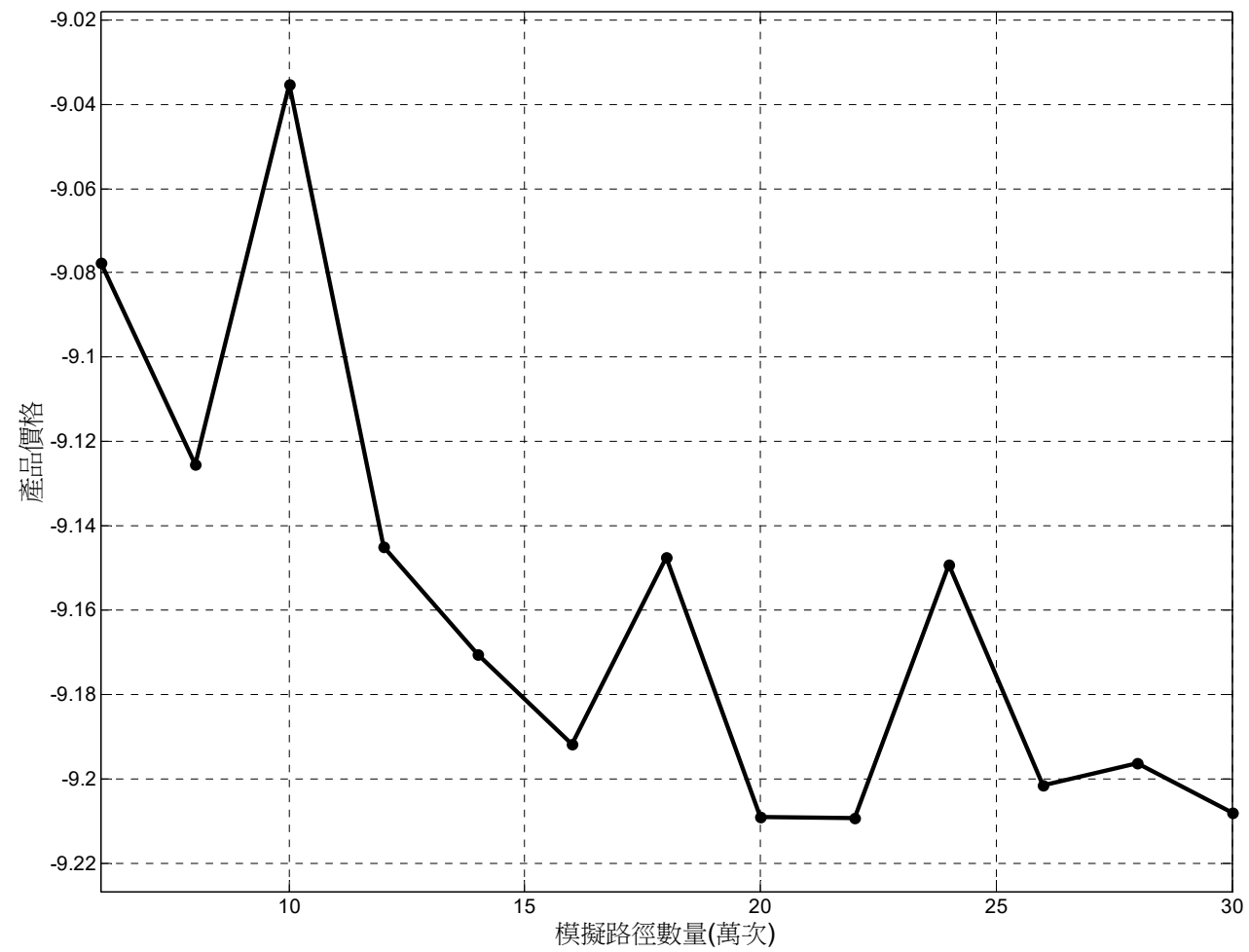
Heston.Kappa = 2.20366282736578;

Heston.Theta = 0.0164951784035976;

Heston.Sigma = 0.33220849746904;

Heston.Rho = -0.277814270110106;

## ◆ 價格收斂





## ◆ 其他參數

SamplePaths	Price	Seconds	Delta	Gamma	Vega	Theta
60000	-9.077558332	42.293	5.512406732	-0.409836909	-3.091002011	0.104117647
80000	-9.125341309	54.679	5.469385748	-0.412778325	-3.148566466	0.104117647
100000	-9.035331286	68.775	5.464588027	-0.359614174	-3.099353443	0.104117647
120000	-9.145061381	89.329	5.501519029	-0.363008071	-3.186643590	0.104117647
140000	-9.170509147	99.258	5.509867670	-0.352625073	-3.199435558	0.104117647
160000	-9.191752471	108.840	5.527067663	-0.339508071	-3.186106195	0.104117647
180000	-9.147401155	126.073	5.499924418	-0.354709865	-3.199056769	0.104117647
200000	-9.209144330	141.437	5.463393767	-0.321905441	-3.211709554	0.104117647
220000	-9.209310352	151.634	5.474486647	-0.298742500	-3.235235585	0.104117647
240000	-9.149269306	156.852	5.485129905	-0.301698861	-3.235176424	0.104117647
260000	-9.201571516	169.135	5.475643380	-0.298182804	-3.238932679	0.104117647
280000	-9.196342961	182.356	5.468440896	-0.298306537	-3.239918091	0.104117647
300000	-9.208031596	194.172	5.485004414	-0.284366803	-3.267691432	0.104117647