結構商品開發與高速運算應用

Structured Products Development with HPC Application

昀騰金融科技

技術長

董夢雲 博士

dongmy@ms5.hinet.net

Part II CMS 連結每日區間計息可贖回債券

CMS Linked Daily Range Accrual Callable Note

大　　綱

一、CMS 連結每日區間計息可贖回債券

◆ 產品範例

產品名稱	10 年期美元 CMS 連結每日區間計息 100%保本型商品
產品代號	CDRNXXXX
賣方	XXXX 股份有限公司
連結標的	紐約時間上午十一時 USD CMS 10 年參考比價利率(依 Reuters 之"ICESWAP3"頁面)
商品交易日	2020 年 10 月 17 日
繳款期限	2020 年 10 月 19 日下午 3:30 之前
商品成立日	2020 年 11 月 01 日
商品到期日	2030 年 11 月 01 日
保本率	100 %
結構化利息(1.80%)	[1.80]% × D/N 每年，下限 0.00%，每季期末支付 N 表每一計息區間總營業日數 D 表每一計息區間 CMS 10 年落於上、下限利率間的營業日數
配息公式	契約本金 × 結構化利息 × 計息基準
計息基準	30/360
契約本金	美金壹佰萬元整（USD$1,000,000）
發行價	100%
期末贖回金額	100%契約本金
利率下限	0.00%
利率上限	第 1 年：3.00%，第 2 年：3.25%，第 3 年：3.75% 第 4 - 6 年：4.00%，第 7 - 10 年：4.25%
發行人贖回	適用
發行人贖回日期	由 2021 年開始每一利息支付日可以贖回
發行人贖回通知日	贖回日前 5 個營業日

◆ 發行人動機

- 鎖定長期資金來源成本
- 利率期限結構如果平坦，可以贖回債券
- 沒有凍結期，發行後便可贖回

◆ 投資人動機

- 高收益
- 預期利率期限結構呈現平坦的形式
- 最低收益為 0.00%的利息收入

◆ 另一孿生產品，CMS Linked Spread Callable Note

- Payoff = Max[CMS20Y – CMS5Y, 0.0]
- 預期利率期限結構呈現陡峭的變化

二、LMM 多因子遠期利率模型

- 早期金融市場的變化型式簡單
  - 利率期限結構大都呈現平行上升、下降形式
  - 單因子即期利率模型可以達成此要求

- 現代金融市場的變化型式複雜
  - 利率期限結構可能陡峭變動(短期下降、長期上升)
  - 利率期限結構也可能平坦變動(短期上升、長期下降)
  - 最近美元的利率倒掛也是以前難以設想的

◆ Basel III 的 IRRBB 六種情境(下表)

Scope of application: Mandatory for all banks within the scope of application set out in Section III.
Content: Quantitative information.
Frequency: Annual, as at the bank's financial year-end.
Format: Fixed.
Accompanying narrative: Commentary on the significance of the reported values and an explanation of any material changes since the previous reporting period.

In reporting currency	ΔEVE		ΔNII	
Period	**T**	**T–1**	**T**	**T–1**
Parallel up				
Parallel down				
Steepener				
Flattener				
Short rate up				
Short rate down				
Maximum				
Period	**T**		**T–1**	
Tier 1 capital				

Table 4. Revised interest rate shocks $\Delta\tilde{R}_{shocktype,c}$

	ARS	AUD	BRL	CAD	CHF	CNY	EUR	GBP	HKD	IDR	INR
Parallel	2,018	310	692	204	110	224	180	225	177	880	431
Short	2,858	440	980	290	155	317	255	319	251	1,246	611
Long	1,345	207	461	136	73	149	120	150	118	586	288

	JPY	KRW	MXN	RUB	SAR	SEK	SGD	TRY	USD	ZAR
Parallel	53	283	452	521	216	198	138	896	197	520
Short	75	401	641	738	306	280	196	1,270	279	737
Long	35	188	301	347	144	132	92	597	131	347

- 新台幣
  - ✓ Parallel：110 b.p.
  - ✓ Short：115 b.p.
  - ✓ Long：100 b.p.

◆ 即期利率模型的發展遇到瓶頸

- 單因子不足以呈現出複雜的利率期限結構變化
- 雙因子模型、三因子模型應運而生
- 現代財務中的執性定價理論完備
  - ✓ 即期利率模型的資產定價往往存在套利機會
  - ✓ 違背金融市場的無套利原則

◆ 多因子遠期利率模型在理論上的發展取得優勢地位

- HJM 是第一個連續的無套利多因子遠期利率模型
  - ✓ 實務操作比較困難
- LMM(或稱 LFM, Lognormal Forward Model)是一個離散的無套利多因子遠期利率模型
  - ✓ 配合實務上的離散遠期利率報價，貼近現實
  - ✓ LFM 成為市場上的定價主流模型

(一)遠期利率的行為

- 在 LFM 中，20 年期的期限中，以 Quarterly Forward Rate 為利率因子，共有 80 個
  - 0~3M(0)、3M~6M(1)、6M~9M(2)、9M~1Y(3)、…、19Y6M~19Y9M(78)、19Y9M~20Y(79)
    - ✓ 10 年契約期間，牽涉 CMS 10 年指標利率
    - ✓ 走 39 步，1~39，第 0 步為目前
  - 每個遠期利率因子有其漂移項(Drift Term)與隨機項(Stochastic Term)
    - ✓ 各遠期利率的漂移項大小，取決於其隨機項數值
    - ✓ 各遠期利率的隨機項大小不同
    - ✓ 各期漂移項之間因無套利限制，有一定的關聯性
  - 彼此具有相關性，非獨立因子
    - ✓ 遠期利率因子的共變異數為時變量
    - ✓ 需要對相關性矩陣做 Cholesky 分解，產生相關性隨機變數
    - ✓ 80 X 80 矩陣處理的運算量非常大

- 在 T_0 時點，我們有下面各個遠期利率，N=79，F_0 已經比價
  - $F_1(T_0,T_1)$、$F_2(T_1,T_2)$、$F_3(T_2,T_3)$、$F_4(T_3,T_4)$、……、$F_{N-1}(T_{N-2},T_{N-1})$、$F_N(T_{N-1},T_N)$。

	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
T_0	-	F1(t0)	F2(t0)	F3(t0)	F4(t0)	F5(t0)	F6(t0)	F7(t0)	F8(t0)	F9(t0)	F10(t0)

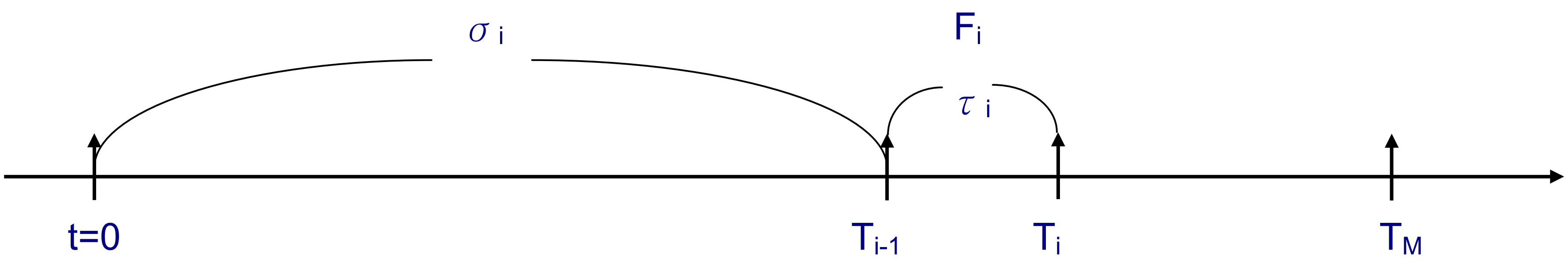
- 在 T_1 時點，我們有下面各個遠期利率
  - $F_2(T_1,T_2)$、$F_3(T_2,T_3)$、$F_4(T_3,T_4)$、……、$F_{N-1}(T_{N-2},T_{N-1})$、$F_N(T_{N-1},T_N)$。

	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
T_0	-	F1(T0)	F2(T0)	F3(T0)	F4(T0)	F5(T0)	F6(T0)	F7(T0)	F8(T0)	F9(T0)	F10(T0)
T_1		-	F2(T1)	F3(T1)	F4(T1)	F5(T1)	F6(T1)	F7(T1)	F8(T1)	F9(T1)	F10(T1)

◆ 在 T_2 時點，我們有下面各個遠期利率

- $F_3(T_2,T_3)$、$F_4(T_3,T_4)$、……、$F_{N-1}(T_{N-2},T_{N-1})$、$F_N(T_{N-1},T_N)$。

	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
T_0	-	F1(T0)	F2(T0)	F3(T0)	F4(T0)	F5(T0)	F6(T0)	F7(T0)	F8(T0)	F9(T0)	F10(T0)
T_1		-	F2(T1)	F3(T1)	F4(T1)	F5(T1)	F6(T1)	F7(T1)	F8(T1)	F9(T1)	F10(T1)
T_2			-	F3(T2)	F4(T2)	F5(T2)	F6(T2)	F7(T2)	F8(T2)	F9(T2)	F10(T2)



◆ 從 T_0 到 T_1，$F_2(T_0)$到 $F_2(T_1)$，$F_3(T_0)$到 $F_3(T_1)$，...，是以何種擴散過程前進?

- 可以都是幾何布朗運動，如下式所示麼？

$$\begin{bmatrix} \frac{dF_2}{F_2} \\ \frac{dF_3}{F_3} \\ \dots \\ \frac{dF_N}{F_N} \end{bmatrix} = \begin{bmatrix} \mu_2 \\ \mu_3 \\ \dots \\ \mu_N \end{bmatrix} \cdot dt + \begin{bmatrix} \sigma_{22} & \sigma_{23} & \dots & \sigma_{2N} \\ \sigma_{32} & \sigma_{33} & \dots & \sigma_{3N} \\ \dots & \dots & \dots & \dots \\ \sigma_{N2} & \sigma_{N3} & \dots & \sigma_{NN} \end{bmatrix} \cdot \begin{bmatrix} dZ_2 \\ dZ_3 \\ \dots \\ dZ_N \end{bmatrix}$$

- 有可能所有的漂移項都是零麼？..............................Q1

$\mu_2 = \mu_3 = \dots = \mu_N = 0$ 。

- 有可能所有主對角線外的波動性都是零麼？...................Q2

$\sigma_{i,j} = 0, \forall i \neq j$ 。

- 對 Q1，No

- 令 $F_k(t) = F(t;T_{k-1},T_k)$，$F_k(t) = L(T_{k-1},T_k)$，$Q^k$ 為以 T_k 到期之零息債券 $P(.,T_k)$為 Numeraire 的遠期（調整）機率測度。

  - 由前一節的說明，可知 $F_k(t)$在 Q^k 下為鞅性的。

    $$dF_k(t) = \boldsymbol{\sigma}_k(t)F_k(t)d\mathbf{Z}^k(t) \text{，} k = 1\ldots M \text{，} t \le T_{k-1}$$

    - ✓ 以 $F_1(t)$為例的表達如下式，

    $$dF_1(t) = [\sigma_1(t), 0, \quad 0, \quad \ldots, \quad 0] \cdot F_1(t) \cdot \begin{bmatrix} dZ_1^1(t) \\ dZ_2^2(t) \\ dZ_3^3(t) \\ \ldots \\ dZ_M^M(t) \end{bmatrix}$$

◆ $F_k(t)$的矩陣表示式如下，

$$dF_k(t) = [0, \quad 0, \quad \sigma_k(t), \quad 0, \quad 0] \cdot F_k(t) \cdot \begin{bmatrix} dZ_1^k(t) \\ \dots \\ dZ_k^k(t) \\ \dots \\ dZ_M^k(t) \end{bmatrix} \qquad \text{.......(2.1)}$$

➢ $\mathbf{Z}^k$ 為 Q^k 測度下 M 維的行向量布朗運動。$\boldsymbol{\sigma}_k$ 為 Q^k 測度下 M 維的列向量。

$$d\mathbf{Z}^k(t) d\mathbf{Z}^k(t)' = \boldsymbol{\rho} \cdot dt \text{ ，} \boldsymbol{\rho} = \left[\rho_{i,j}\right] \text{，} i, j = 1 \ldots M \qquad \text{.......(2.2)}$$

$$\boldsymbol{\sigma}_j(t) = [0, \quad 0, \quad \ldots, \quad \sigma_j(t), \quad \ldots, \quad 0, \quad 0]$$

◆ 在 Black 76 模型下，我們假設 GBM 的形式

$$dF(t) = \sigma(t) F(t) dZ(t)$$

$$\frac{dF_t}{F_t} = \sigma dZ_t$$

- 對 Q2，No

- 上式可以純量公式表示如下，
  - 同上，類似 Black 76 的形式

$$dF_k(t) = \sigma_k(t)F_k(t)dZ_k^k(t) \quad \text{......} (2.3)$$

  - ✓ Z_k^k 為 $\mathbf{Z}^k$ 中的第 k 個成份。

- 藉由 Ito's 公式，(2.3)式可寫為，

$$d\ln F_k(t) = -\frac{\sigma_k(t)^2}{2}dt + \sigma_k(t)dZ_k(t)$$

  - 上式的解可求得為，

$$\ln F_k(T) = \ln F_k(0) - \int_0^T \frac{\sigma_k(t)^2}{2}dt + \int_0^T \sigma_k(t)dZ_k(t) \quad \text{......} (2.4)$$

◆ 在對數常態假設下，$F_k(t)$在遠期（調整）機率測度 Q^i 下的動態，分別為

- $i<k,\quad t\le T_i$ ：有正的漂移項，

$$dF_k(t)=\sigma_k(t)F_k(t)\sum_{j=i+1}^{k}\frac{\rho_{k,j}\tau_j\sigma_j(t)F_j(t)}{1+\tau_jF_j(t)}dt+\sigma_k(t)F_k(t)dZ_k(t) \qquad (2.5)$$

- $i=k,\quad t\le T_i$ ：無漂移項，

$$dF_k(t)=\sigma_k(t)F_k(t)dZ_k(t)$$

- $i>k,\quad t\le T_i$ ：有負的漂移項，

$$dF_k(t)=-\sigma_k(t)F_k(t)\sum_{j=k+1}^{i}\frac{\rho_{k,j}\tau_j\sigma_j(t)F_j(t)}{1+\tau_jF_j(t)}dt+\sigma_k(t)F_k(t)dZ_k(t)$$

◆ 一種類比的觀點。

- 定速前進的汽車，V_O，車上有醉漢隨機前後走動，Z_O。
  - ✓ 另一汽車上有觀察者，車速 Vs。
- $V_O > V_S$，ΔT 時間後，醉漢運動位移如下，

  $(V_O - V_S)\Delta\mathrm{T} + Z_O$
- $V_O = V_S$，醉漢運動行為如下，

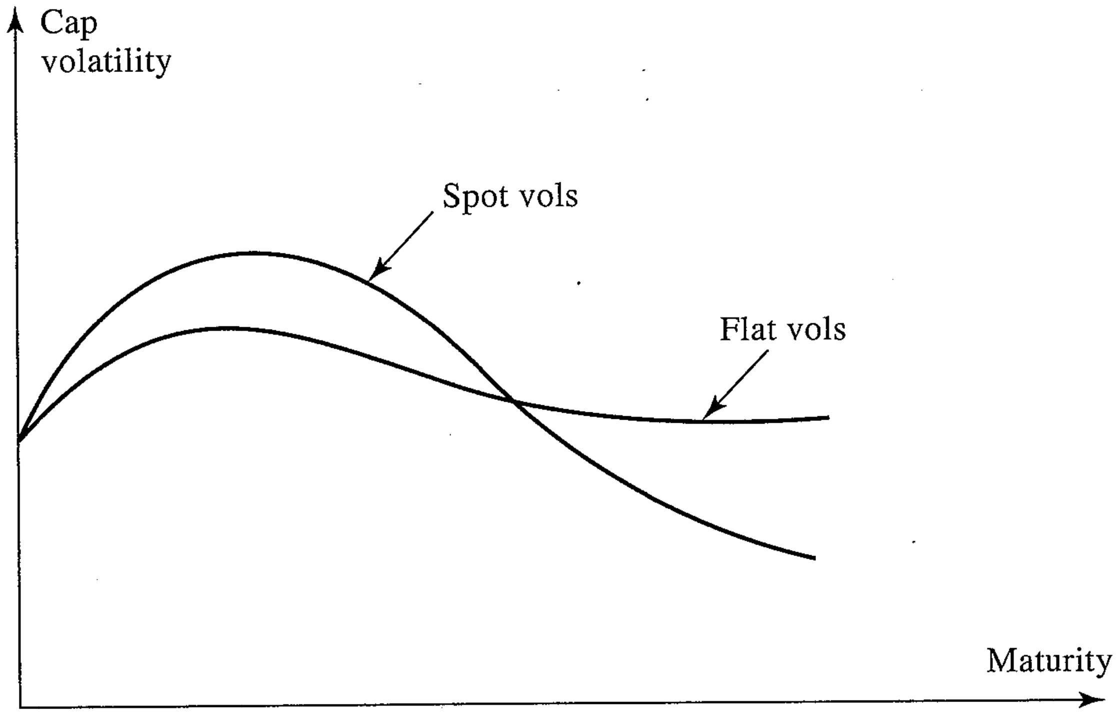
  Z_O
- $V_O < V_S$，ΔT 時間後，醉漢運動位移如下，

  $-(V_O - V_S)\Delta\mathrm{T} + Z_O$

(二)遠期利率的波動性期限結構

- 下圖為典型的 Volatility Term Structure 的關係圖。
  - Flat Vol 可由市場 Caps/Floors 報價取得。
  - Spot Vol 可以使用 Bootstrap 方式求得 Caplet/Fllorlet 的 Vol 值。

Figure 26.3 The volatility hump.



◆ 特定的函數關係可以用來描繪 Spot Volatility Curve。

➢ $p(s)=(a+bs)e^{-cs}+d$ 為常用的函數關係，s 為遠期利率的起始時間間距。

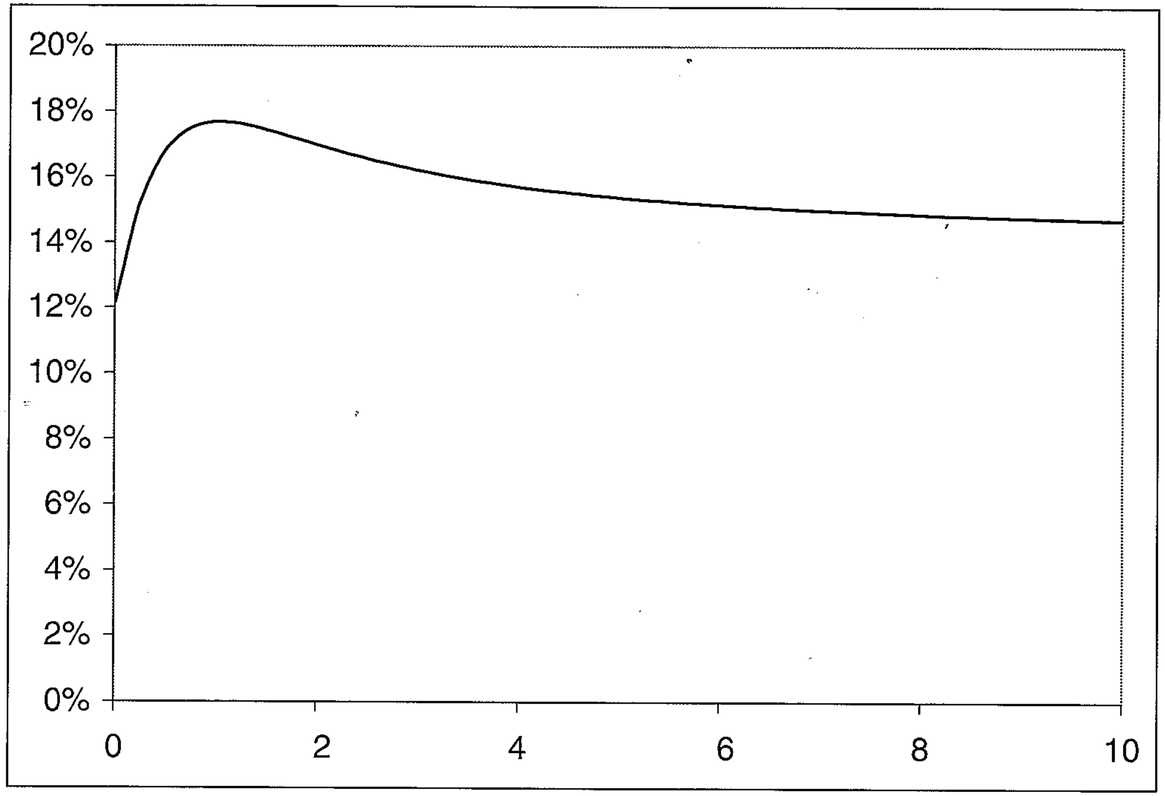


Fig. 14.1. The term structure of implied volatilities of caplets implied by the functional form (14.21), with $a=-0.02$, $b=0.3$, $c=2$, and $d=0.14$.

◆ 令 $F(t,T_{i-1},T_i)=F_i$ 之波動性函數為 $\sigma_i(t)$ ，則我們有下面關係，

$$\sigma^{caplet}(t,T_{i-1},T_i)^2=\frac{1}{\tau_{t,T_{i-1}}}\int_t^{T_{i-1}}\sigma_i^2(t)dt$$

◆ 令 F_j 為由 T_{j-1} 到 T_j 之遠期利率，其波動性 $\sigma_j(t)$ 有下面的型式，

$$p(t_{j-1}-t)=p(s)=(a+b\cdot s)e^{-c\cdot s}+d \quad \text{.......................................(2.6)}$$

➢ p 函數與 j 無關，遠期利率波動性曲線為 Time-homogeneous。

◆ σ_j 為由 Caplet 報出之 T_{j-1} 到 T_j 之遠期利率的波動性，利用此市場資訊，找出適當的 a, b, c, d 參數，

➢ σ_j 與波動性函數的關係為，

$$\sigma_j^2(T_{j-1}-0)=\int_0^{T_{j-1}}\sigma(u)^2\cdot du$$

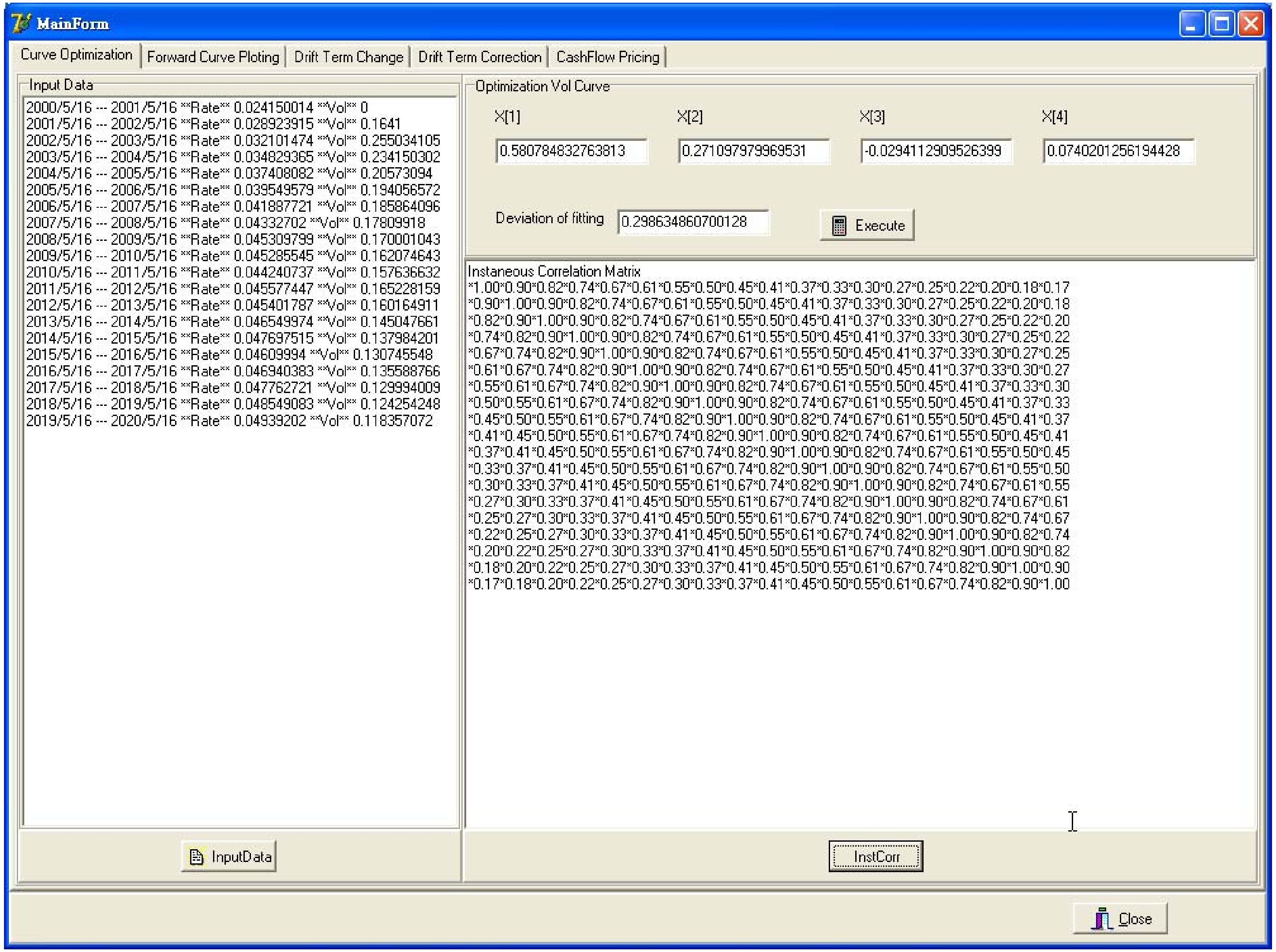
◆ 設定如下的目標函數，

$$\min\sum_j\left|\sigma_j^2 T_{j-1}-\int_0^{T_{j-1}} p(T_{j-1}-u)^2\,du\right| \quad\text{..........}(2.7)$$

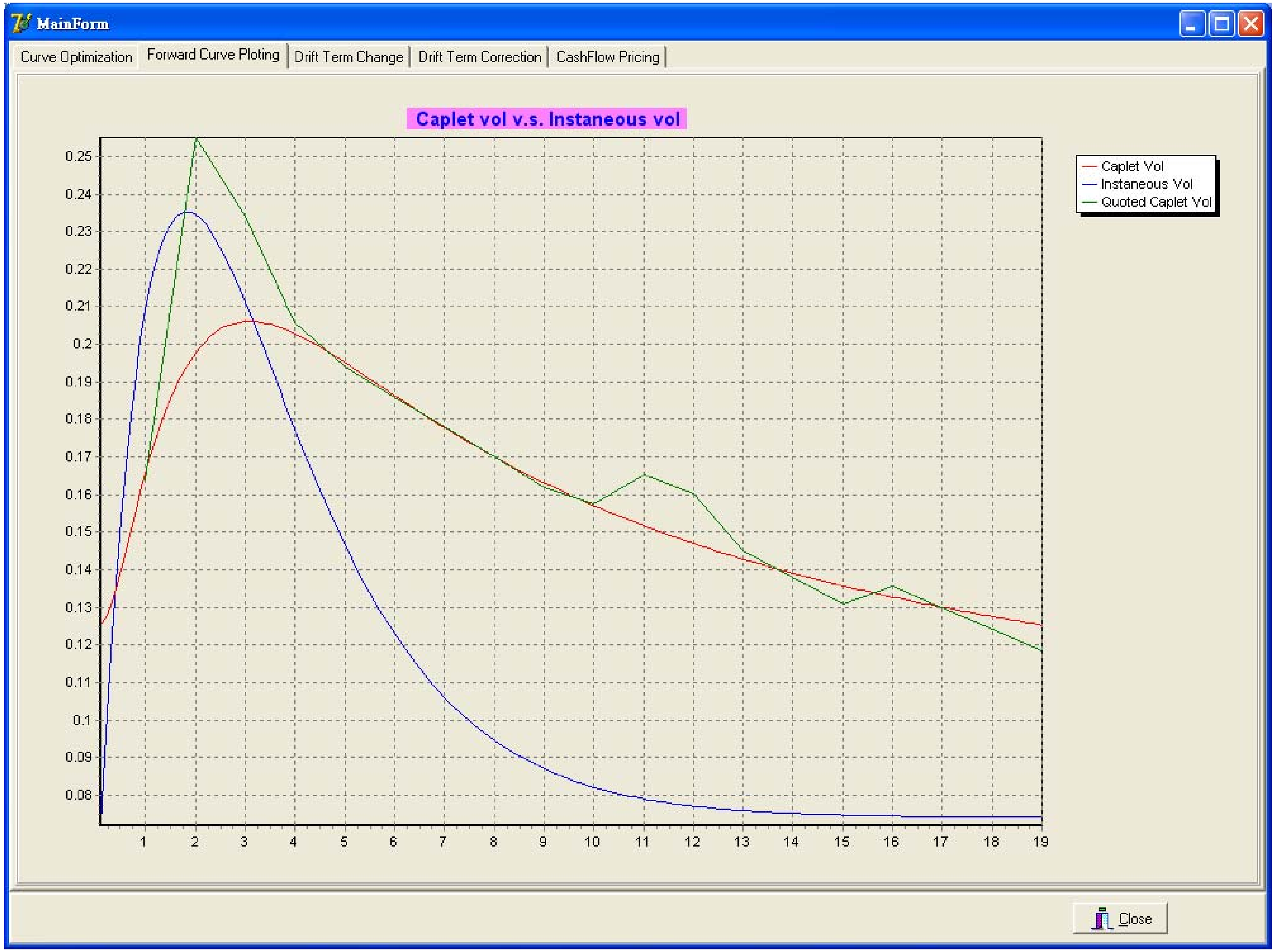
- 利用非線性最適化方式，使用 Powell 演算法求得參數。
- 任何遠期利率 $F(t,T_{i-1},T_i)$ 的波動性，可如下求得，

$$\sigma_i^2(T_{i-1}-t)=\int_t^{T_{i-1}}\sigma(u)^2\cdot du$$

◆ Powell 法最適化之實作



◆ (2.6)式最適化曲線之作圖



- 實務上處理時，增加一個自由度 k_j，提高波動性的擬合度。

$$\min \sum_j \left| \sigma_j^2 T_{j-1} - k_j \int_0^{T_{j-1}} p(T_{j-1}-u)^2\, du \right| \quad \text{(2.8)}$$

  - QuantLib 的範例也是如此處理。

(三)相關性結構

- 如果一特定時點的現金流量與兩個以上的遠期利率有關，則他們之間的相關性需加以考慮。
  - 模擬時需同時產生這些具相關性的遠期利率。
    - ✓ (2.1.6)式中的隨機項需反應出相關性。

$$dZ_i dZ_j = \rho_{i,j} dt$$

- 相關係數可以考慮由歷史資料求得之歷史相關性來替代。
  - 實際估計歷史資料，往往產生不理想的性質。
    - ✓ 部份相關係數為負的。
  - 也可使用 Pivot Matrices 方法。
    - ✓ 利用相關係數矩陣的少數項點，配合外加的規範性質，導出合理的相關係數矩陣。
  - 考慮使用主成份分析法，求取相關係數矩陣的主要性質，避開上面的異常性質。
    - ✓ 可使用一些減秩的公式，如 Rebonato 的三角函數。

◆ 遠期利率歷史相相關性。

1.00	0.82	0.69	0.65	0.58	0.47	0.29	0.23	0.43	0.47	0.33	0.43	0.29	0.23	0.26	0.21	0.23	0.29	0.25
0.82	1.00	0.80	0.73	0.68	0.55	0.45	0.40	0.53	0.57	0.42	0.45	0.48	0.34	0.35	0.32	0.32	0.31	0.32
0.69	0.80	1.00	0.76	0.72	0.63	0.47	0.56	0.67	0.61	0.48	0.52	0.48	0.54	0.46	0.42	0.45	0.42	0.35
0.65	0.73	0.76	1.00	0.78	0.67	0.58	0.56	0.68	0.70	0.56	0.59	0.58	0.50	0.50	0.48	0.49	0.44	0.35
0.58	0.68	0.72	0.78	1.00	0.84	0.66	0.67	0.71	0.73	0.70	0.67	0.64	0.59	0.58	0.65	0.65	0.53	0.42
0.47	0.55	0.63	0.67	0.84	1.00	0.77	0.68	0.73	0.69	0.77	0.69	0.66	0.63	0.61	0.68	0.70	0.57	0.45
0.29	0.45	0.47	0.58	0.66	0.77	1.00	0.72	0.71	0.65	0.65	0.62	0.71	0.62	0.63	0.66	0.64	0.52	0.38
0.23	0.40	0.56	0.56	0.67	0.68	0.72	1.00	0.73	0.66	0.64	0.56	0.61	0.72	0.59	0.64	0.64	0.49	0.46
0.43	0.53	0.67	0.68	0.71	0.73	0.71	0.73	1.00	0.75	0.59	0.66	0.69	0.69	0.69	0.63	0.64	0.52	0.40
0.47	0.57	0.61	0.70	0.73	0.69	0.65	0.66	0.75	1.00	0.63	0.68	0.70	0.63	0.64	0.65	0.62	0.52	0.40
0.33	0.42	0.48	0.56	0.70	0.77	0.65	0.64	0.59	0.63	1.00	0.83	0.72	0.64	0.58	0.68	0.73	0.57	0.45
0.43	0.45	0.52	0.59	0.67	0.69	0.62	0.56	0.66	0.68	0.83	1.00	0.82	0.69	0.67	0.70	0.69	0.65	0.43
0.29	0.48	0.48	0.58	0.64	0.66	0.71	0.61	0.69	0.70	0.72	0.82	1.00	0.79	0.78	0.79	0.72	0.59	0.42
0.23	0.34	0.54	0.50	0.59	0.63	0.62	0.72	0.69	0.63	0.64	0.69	0.79	1.00	0.82	0.83	0.79	0.60	0.45
0.26	0.35	0.46	0.50	0.58	0.61	0.63	0.59	0.69	0.64	0.58	0.67	0.78	0.82	1.00	0.90	0.80	0.50	0.22
0.21	0.32	0.42	0.48	0.65	0.68	0.66	0.64	0.63	0.65	0.68	0.70	0.79	0.83	0.90	1.00	0.94	0.71	0.46
0.23	0.32	0.45	0.49	0.65	0.70	0.64	0.64	0.64	0.62	0.73	0.69	0.72	0.79	0.80	0.94	1.00	0.82	0.66
0.29	0.31	0.42	0.44	0.53	0.57	0.52	0.49	0.52	0.52	0.57	0.65	0.59	0.60	0.50	0.71	0.82	1.00	0.84
0.25	0.32	0.35	0.35	0.42	0.45	0.38	0.46	0.40	0.40	0.45	0.43	0.42	0.45	0.22	0.46	0.66	0.84	1.00

Table 6.2. Historically estimated instantaneous correlation matrix for forward LIBOR rates, February 1, 2002.

◆ 相關係數也可以使用市場 Swaption 的 Vol 報價資料求得。

➢ 標的物為由 T_α 到 T_β 之 Swap 的 Swaption，其 Black76 波動性的報價與遠期利率的關係可近似如下(Rebonato's Formula)，

$$v_{\alpha,\beta}^2(T_\alpha) = \sum_{i,j=\alpha+1}^{\beta} \frac{w_i(0)w_j(0)F_i(0)F_j(0)\rho_{i,j}}{S_{\alpha,\beta}(0)^2} \int_0^{T_\alpha} \sigma_i(t)\sigma_j(t)dt \quad \text{......(2.9)}$$

$$w_i(t) = \frac{\tau_i P(t,T_i)}{\sum_{k=\alpha+1}^{\beta} \tau_k P(t,T_k)}$$

➢ 我們可以假設相關係數為 Time Homogeneous 的型式，

$$\rho_{i,j} = \rho(T_i - T_j) = \exp(-\beta \,|\, T_i - T_j \,|) \quad \text{......(2.10)}$$

✓ 類似(2.7)式，給定(2.6)式與(2.10)式的函數結構，極小化(2.7)式誤差平方和，求取控制變數 a, b, c, d 與 β。

◆ 市場上 ATM Swaption Vol 報價

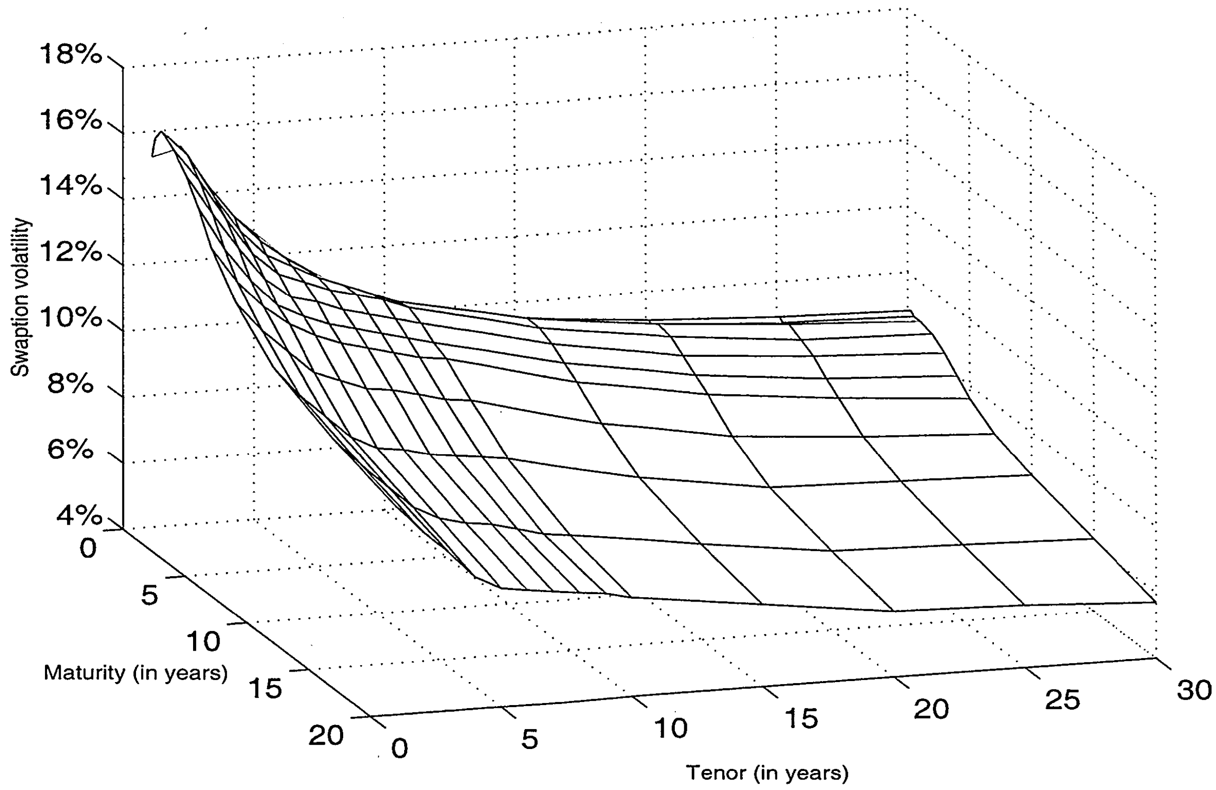


Fig. 1.4. At-the-money Euro swaption volatility surface on February 8, 2001, at 5 p.m.

- (2.10)式的結構是最簡單的型式，其他可能型式有，
  - 兩參數型式，市場上的主流形式

$$\rho_{i,j}=\rho_{\infty}+(1-\rho_{\infty})\exp(-\beta\,|T_i-T_j|) \quad \text{.........(2.11)}$$

  - 三參數型式，

$$\rho_{i,j}=\rho_{\infty}+(1-\rho_{\infty})\exp[-|T_i-T_j|(\beta-\alpha(\max(T_i-T_j)-1))]$$

主題三、多因子模擬的處理

(一)模型市場參數校正

- 使用 Swaption Vol 來進行模型參數的市場校正。
  - (2.8)式，a、b、c、d 四個參數，k_j，j=1..79，79 個參數
  - (2.11)式，alpha、beta 兩個參數
  - 一共 85 個參數要校正

- (2.9)式可以求得 Swaption Vol 的近式公式
  - 以模型求的之 Swaption Vol，與市場報價之 Swaption Vol，形成誤差及小化之目標函數

$$\min_{a,b,c,d,k_j,alpha,beta} \sum \left| \sigma_{\alpha,\beta}^2 T_\alpha - v_{\alpha,\beta}^2 T_\alpha \right| \quad \text{.........(3.1)}$$

- 使用 Levenberg-Marquardt 演算法，找出 85 個參數最適值
  - 需要很長時間，約 6 分鐘跑出來

- 可以多跑一個 Factor，86 個參數
  - 考慮到節假日的調整，多一期比較保險

(二)多因子模擬

- 每條路徑有 80 個利率在漂移
  - 隨時間前進，有些遠期利率 Fixing，越來越少要模擬
  - 一共走 10 年，配合期初，共走 39 步
  - 第一步有 80X80 相關性矩陣要分解
    - 第二步為 79X79 相關性矩陣要分解
    - 越來越少，最後 40X40 相關性矩陣要分解

- RV(t)的路徑如下圖，RV(T_0)為市場已知的資料，
  - ✓ 由 RV(T_0)→RV(T_1)，RV(T_1)→RV(T_2)，…，RV(T_6)→RV(T_7)。
  - ✓ $F_1(T_0)$，$F_2(T_1)$，$F_3(T_2)$，…，$F_7(T_6)$，$F_8(T_7)$，$F_9(T_8)$，$F_{10}(T_9)$，(紅色利率)為到期之利率。

	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
RV(T_0)	-	F1(T0)	F2(T0)	F3(T0)	F4(T0)	F5(T0)	F6(T0)	F7(T0)	F8(T0)	F9(T0)	F10(T0)
RV(T_1)		-	F2(T1)	F3(T1)	F4(T1)	F5(T1)	F6(T1)	F7(T1)	F8(T1)	F9(T1)	F10(T1)
RV(T_2)			-	F3(T2)	F4(T2)	F5(T2)	F6(T2)	F7(T2)	F8(T2)	F9(T2)	F10(T2)
RV(T_3)				-	F4(T3)	F5(T3)	F6(T3)	F7(T3)	F8(T3)	F9(T3)	F10(T3)
RV(T_4)					-	F5(T4)	F6(T4)	F7(T4)	F8(T4)	F9(T4)	F10(T4)
RV(T_5)						-	F6(T5)	F7(T5)	F8(T5)	F9(T5)	F10(T5)
RV(T_6)							-	F7(T6)	F8(T6)	F9(T6)	F10(T6)
RV(T_7)								-	F8(T7)	F9(T7)	F10(T7)
RV(T_8)									-	F9(T8)	F10(T8)
RV(T_9)										-	F10(T9)
RV(T_{10})											-

- 依照模擬之遠期利率，計算 CMS 利率
  - 只有計算 40 個季節點的 CMS 利率
- 節點間的每日 CMS 利率，可以採用線性內插補足
  - 依照契約條件，計算每期 Coupon
  - 折現期初價值，便可求的 Non-Callable 債券 MTM
- 可贖回債券可用 LSMC 方式，決定是否提前贖後
  - 由期末往前判斷
  - CMS 利率、Discount Function 可為自變數
  - 類似 ZCN 的方式進行

(三)多因子平行模擬

- 單線程模擬速度太慢
  - 以 2 萬條路徑為例，本人桌機需 20 分鐘跑完
    - ✓ Intel i5 8 cores CPU、16 G RAM，Windows 10

- 採取多線程平行運算
  - .Net Framework Task Parallel Library
  - 使用 8 核平行運算
  - 約 2 分鐘跑完，10X 性能提升

- 配合平行運算，改變程式的演算邏輯
  - 矩陣分解每條路徑都相同
    - 可以期初先做好，傳入各線程直接使用
    - 此法在單線程也可使用

(四)GPU 平行運算的模擬加速實作

- 大量模擬路徑可以使用 GPU 進一步加速
  - 使用 GPU 的多核來進行模擬運算
  - 模型市場校正與 LSMC 的部分可以在 CPU 端執行
  - 相關性矩陣的處理可以先進行一次，結果直接傳入 GPU

GPU 程式的撰寫邏輯

- 由 CPU 起動程式
  - 進行 IO，設定計算所需相關變數
- 起動 GPU，傳入相關變數
  - GPU 執行多核平行運算，產生結果
- 將結果由 GPU 中取出，傳回 CPU
  - CPU 內進一步加工
- CPU 輸出
  - 顯示給使用者

◆ 一個簡單的範例，SAXPY，y[i] = a*x[i]+y[i]。

➢ CPU Loop

```
void saxpy(int n, float a, float *x, float *y)
{
    for(int i=0; i<n; i++)
    {
        y[i] = a * x[i] + y[i];
    }
}
```

➢ GPU Code

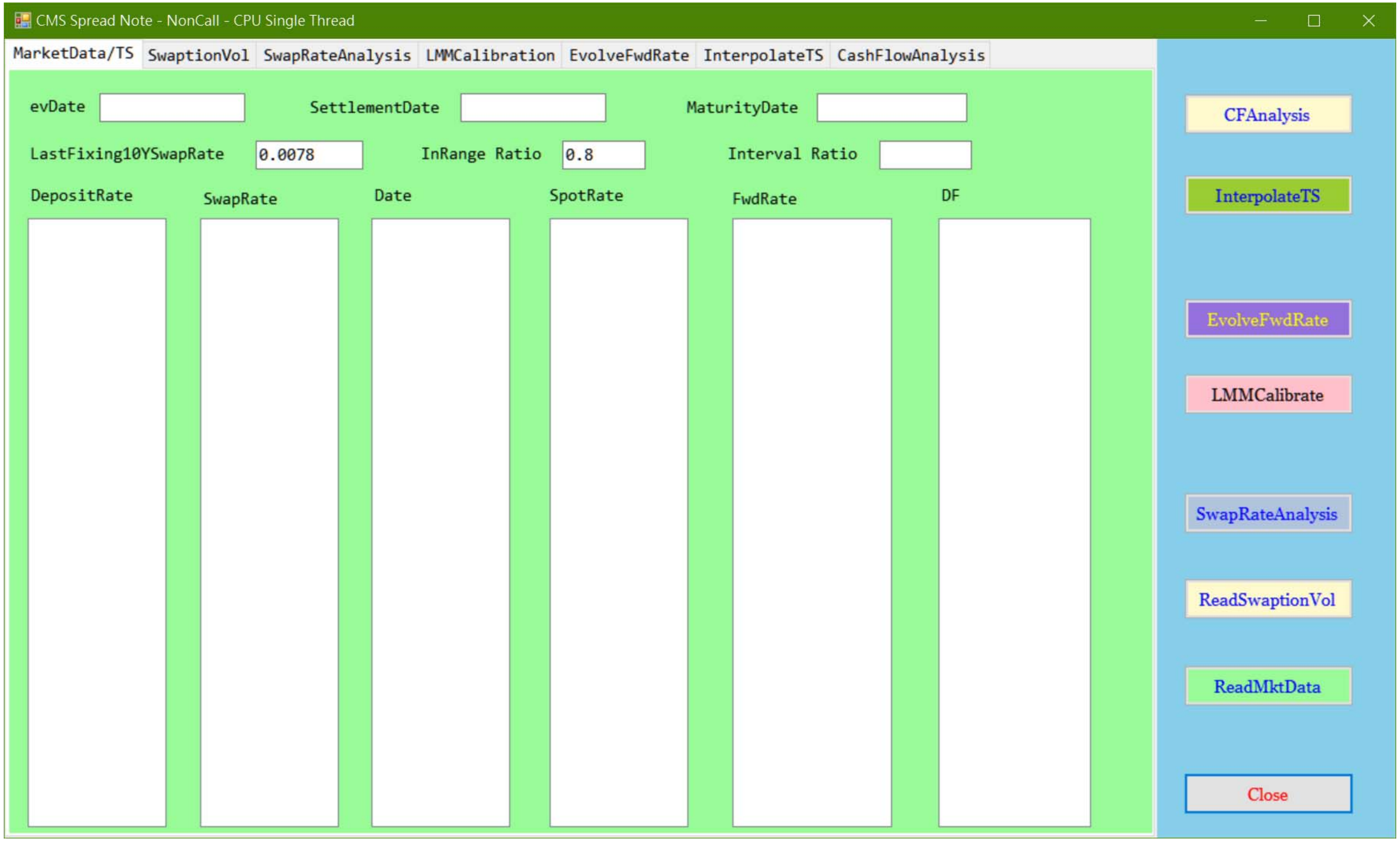
```
__global__ void saxpy(int n, float a, float *x, float *y)
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < n) y[i] = a*x[i] + y[i];
}
```

- 在 GPU 中，所有的核分為 n 個 Block，每個 Block 分為 m 個 Thread
  - 如果一個核是 $Block_i$，$Thread_j$，則其在 GPU 的總索引為

    $Total\_index = Block_i * n + Thread_j$
  - 1024 個核，分為 8 個 Blocks，每個 Block 有 128 個 Threads
    - ✓ (Blocki, Threadj) = (0, 0)，Total_index = 0 * 128 + 0 = 0(First Core)
    - ✓ (Blocki, Threadj) = (1, 0)，Total_index = 1 * 128 + 0 = 128
    - ✓ (Blocki, Threadj) = (3, 22)，Total_index = 3 * 128 + 22 = 406
    - ✓ (Blocki, Threadj) = (7, 127)，Total_index = 7 * 128 + 127 = 1023(Last Core)

主題四、CPU 平行運算的模擬加速實作

◆ 範例程式



◆ 評價日期：2020/10/15

➢ Settlement Date：2020/10/19，以 2020/10/15 市場資料評價。

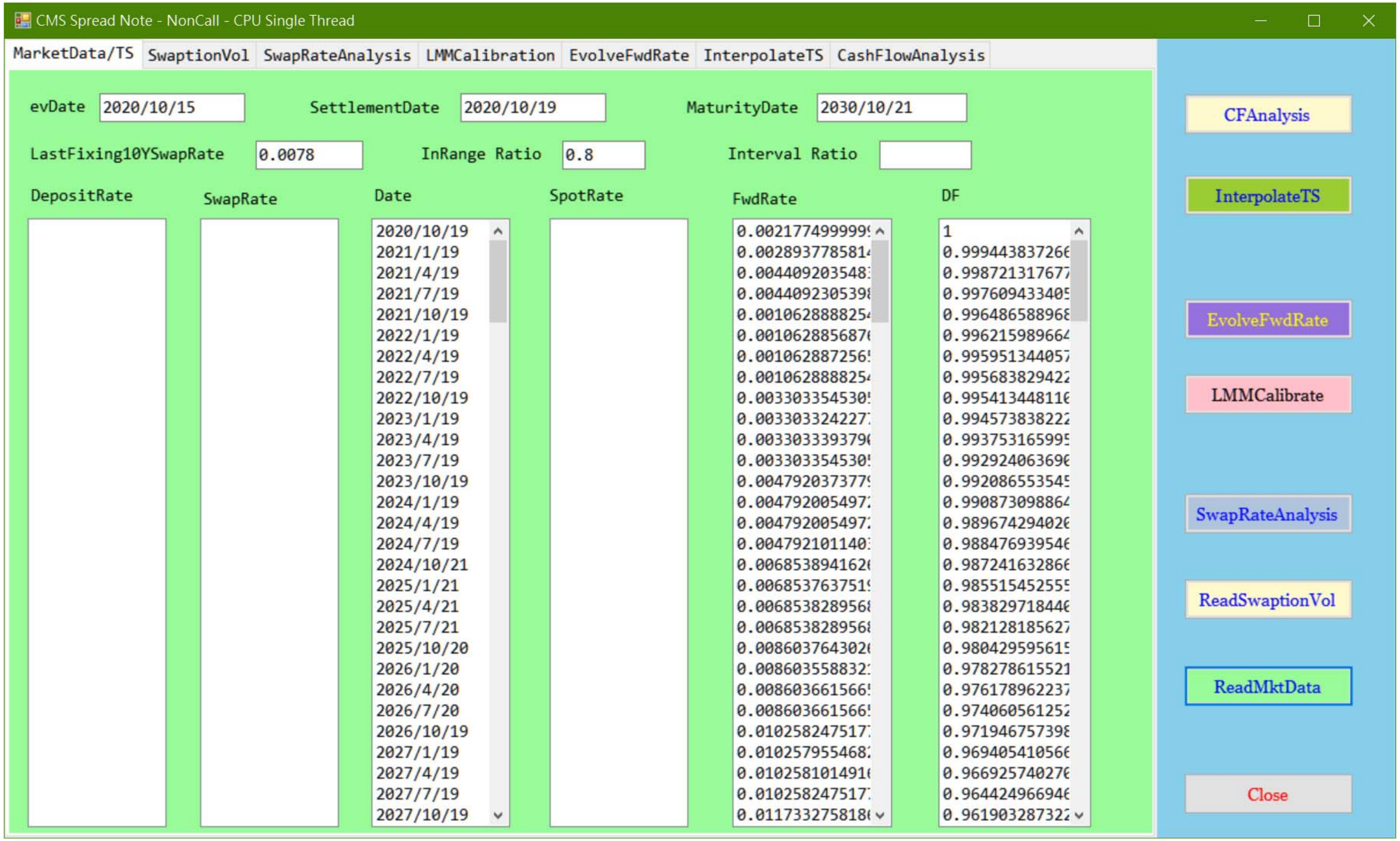
```
// deposits 2020/10/15, 3
public Datum[] depositData = new Datum[]
{
    new Datum { n = 3, units = TimeUnit.Months, rate = 0.0021775 },
    new Datum { n = 6, units = TimeUnit.Months, rate = 0.0025325 },
    new Datum { n = 12, units = TimeUnit.Months, rate = 0.0034775 }
};

// swaps 2020/10/15, 17
public Datum[] swapData = new Datum[]
{
    new Datum { n =  2, units = TimeUnit.Years, rate = 0.002301 },
    new Datum { n =  3, units = TimeUnit.Years, rate = 0.00265 },
    new Datum { n =  4, units = TimeUnit.Years, rate = 0.0032055 },
    new Datum { n =  5, units = TimeUnit.Years, rate = 0.003946 },
    new Datum { n =  6, units = TimeUnit.Years, rate = 0.00473 },
    new Datum { n =  7, units = TimeUnit.Years, rate = 0.005525 },
    new Datum { n =  8, units = TimeUnit.Years, rate = 0.006305 },
    new Datum { n =  9, units = TimeUnit.Years, rate = 0.007037 },
    new Datum { n =  10, units = TimeUnit.Years, rate = 0.007701 },
    new Datum { n =  11, units = TimeUnit.Years, rate = 0.008285 },
    new Datum { n =  12, units = TimeUnit.Years, rate = 0.0088075 },
    new Datum { n =  15, units = TimeUnit.Years, rate = 0.0099405 },
    new Datum { n =  20, units = TimeUnit.Years, rate = 0.01108966 },
    new Datum { n =  25, units = TimeUnit.Years, rate = 0.011555 },
    new Datum { n =  30, units = TimeUnit.Years, rate = 0.011764 },
    new Datum { n =  40, units = TimeUnit.Years, rate = 0.0114154 },
    new Datum { n =  50, units = TimeUnit.Years, rate = 0.0106654 }
};
```

◆ 契約條件，

- 為比對 Bloomberg 上、下限固定
  - ✓ 上限固定 0.00%
  - ✓ 下限固定 4.25%
- Coupon Rate
  - ✓ Non-Call Coupon Rate 1.65%
  - ✓ Callable Coupon Rate 1.80%

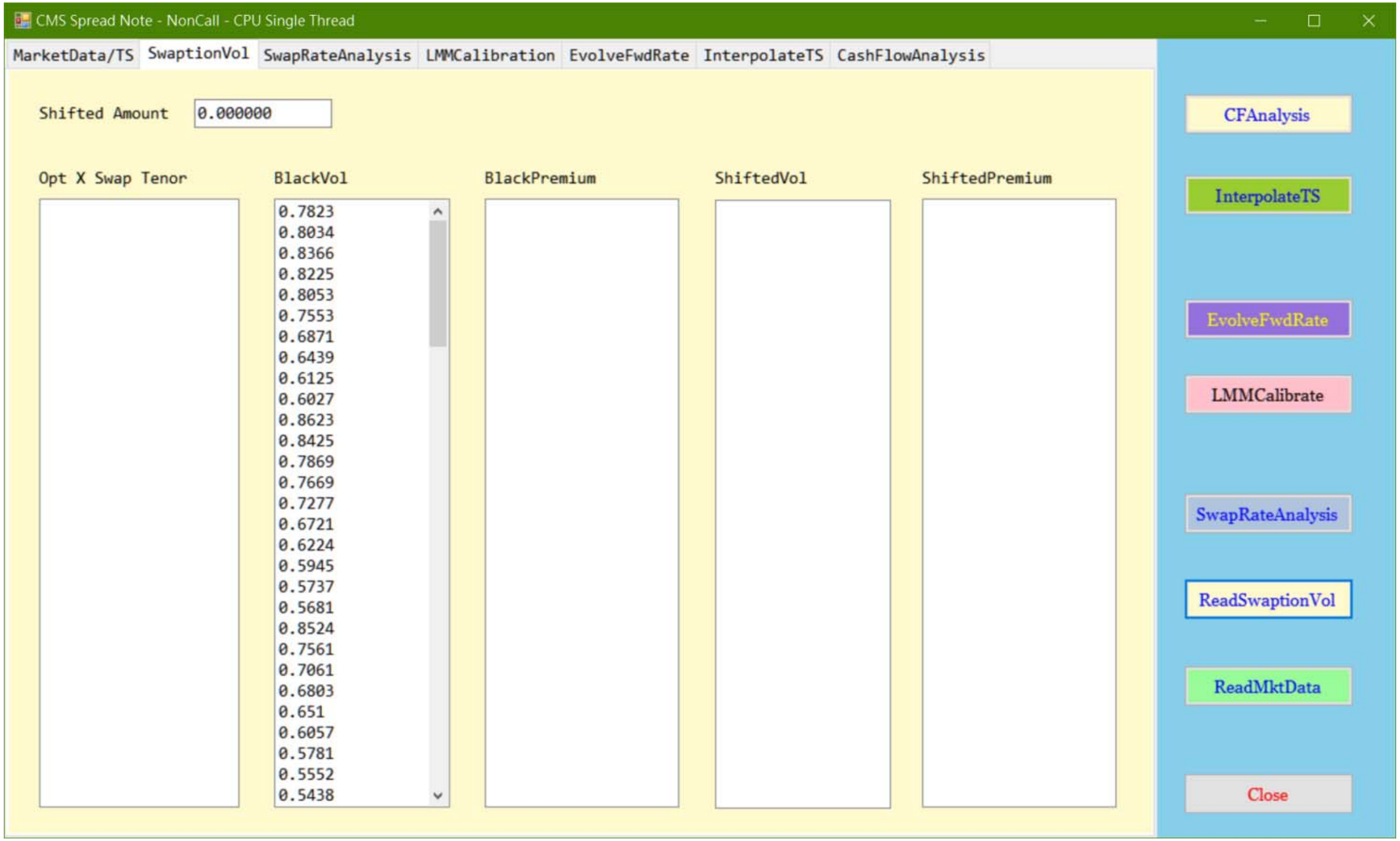
◆ Read Market Data



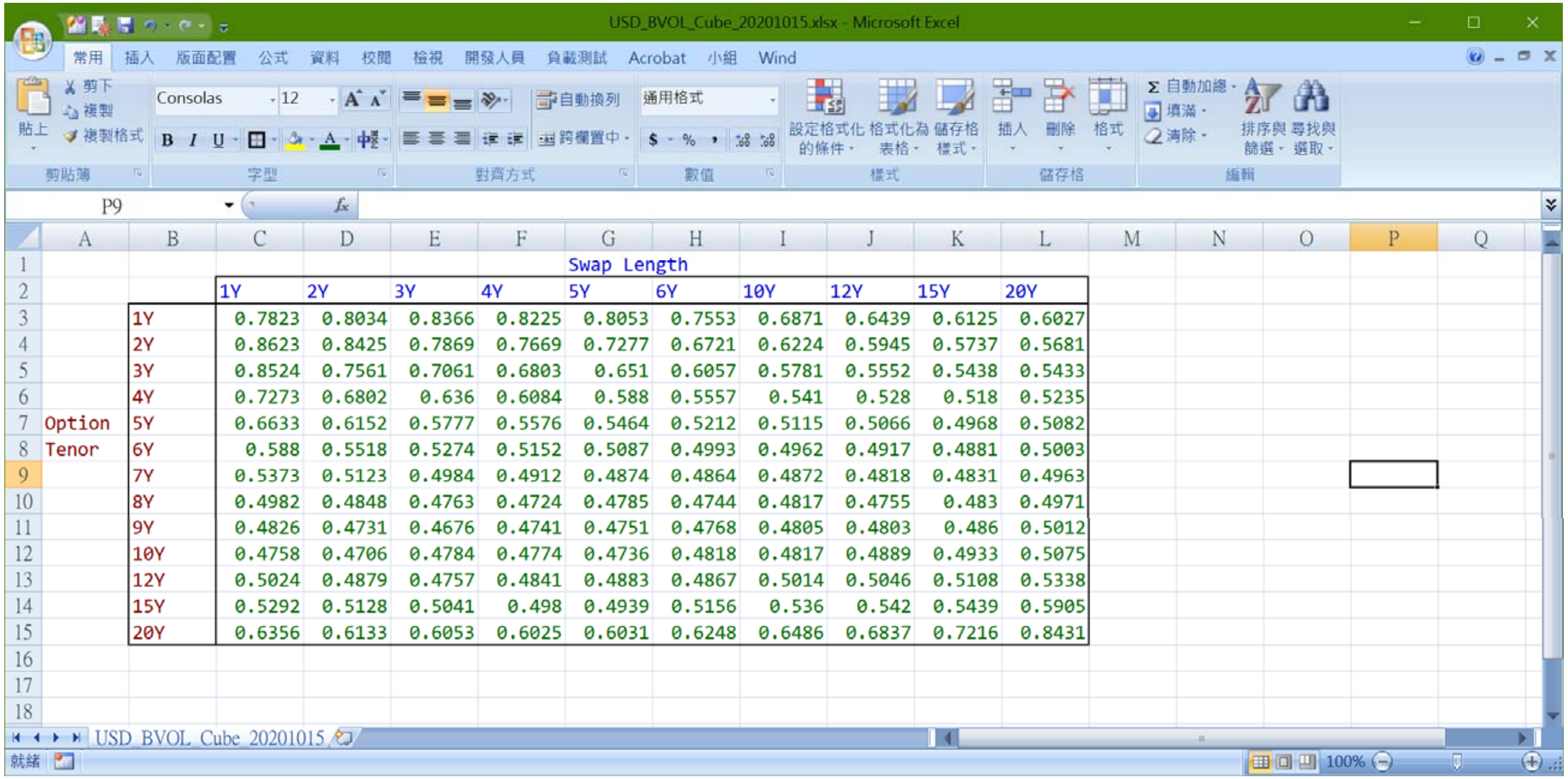
◆ 產生 200 個遠期利率

```
0.00217749999999997 => 0M * 3M
0.0028937785814902  => 3M * 6M
0.00440920354831275 => 6M * 9M
0.00440923053989172
0.00106288882546075
0.00106288568760426
0.00106288725653145
0.00106288882545988
0.00330335453054107
0.00330332422772184
0.00330333937908456
0.00330335453054107
0.00479203737798277
0.00479200549721522
0.0047920054972161
0.00479210114036468
0.00685389416264682
0.00685376375190572
0.00685382895686264
0.00685382895686352
…
…
…
```

◆ ReadSwaptionVol 功能

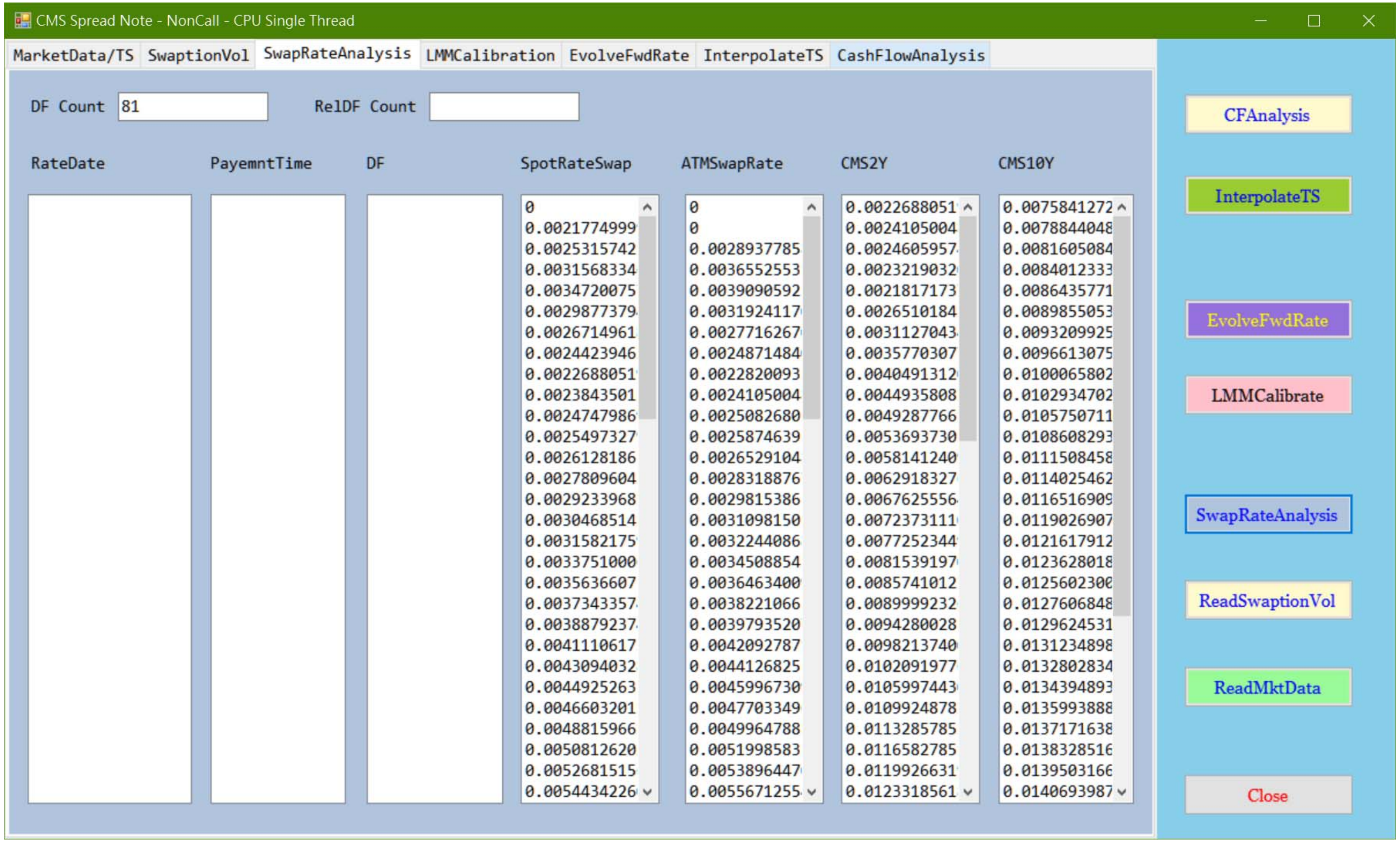


➢ 讀入資料



		Swap Length									
		1Y	2Y	3Y	4Y	5Y	6Y	10Y	12Y	15Y	20Y
	1Y	0.7823	0.8034	0.8366	0.8225	0.8053	0.7553	0.6871	0.6439	0.6125	0.6027
	2Y	0.8623	0.8425	0.7869	0.7669	0.7277	0.6721	0.6224	0.5945	0.5737	0.5681
	3Y	0.8524	0.7561	0.7061	0.6803	0.651	0.6057	0.5781	0.5552	0.5438	0.5433
	4Y	0.7273	0.6802	0.636	0.6084	0.588	0.5557	0.541	0.528	0.518	0.5235
Option	5Y	0.6633	0.6152	0.5777	0.5576	0.5464	0.5212	0.5115	0.5066	0.4968	0.5082
Tenor	6Y	0.588	0.5518	0.5274	0.5152	0.5087	0.4993	0.4962	0.4917	0.4881	0.5003
	7Y	0.5373	0.5123	0.4984	0.4912	0.4874	0.4864	0.4872	0.4818	0.4831	0.4963
	8Y	0.4982	0.4848	0.4763	0.4724	0.4785	0.4744	0.4817	0.4755	0.483	0.4971
	9Y	0.4826	0.4731	0.4676	0.4741	0.4751	0.4768	0.4805	0.4803	0.486	0.5012
	10Y	0.4758	0.4706	0.4784	0.4774	0.4736	0.4818	0.4817	0.4889	0.4933	0.5075
	12Y	0.5024	0.4879	0.4757	0.4841	0.4883	0.4867	0.5014	0.5046	0.5108	0.5338
	15Y	0.5292	0.5128	0.5041	0.498	0.4939	0.5156	0.536	0.542	0.5439	0.5905
	20Y	0.6356	0.6133	0.6053	0.6025	0.6031	0.6248	0.6486	0.6837	0.7216	0.8431

◆ SwapRateAnalysis 功能



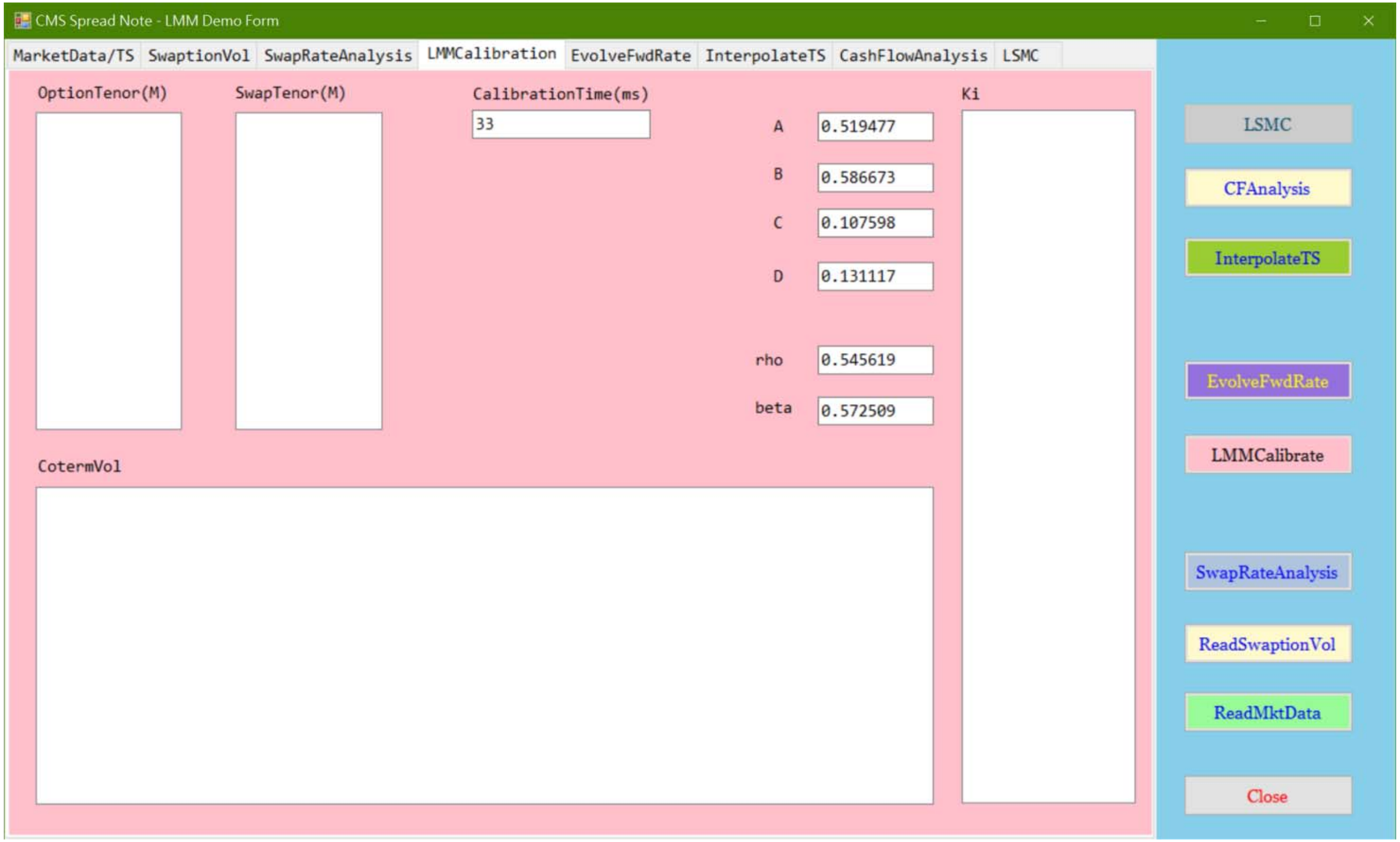
- 根據 Swaps 簡化公式，計算 2Y 與 10Y 的 Swaps 利率。
  - ✓ SpotRateSwap 為 0*1Q、0*2Q、0*3Q、0*4Q、…，的即期 Swaps。
  - ✓ ATMSwapRate 為 1Q*2Q、1Q*3Q、1Q*4Q、…，的 ATM Swaps Rates，都是由 1Q(3M)後開始的 Forward Start Swap Rate。

- 簡化公式，

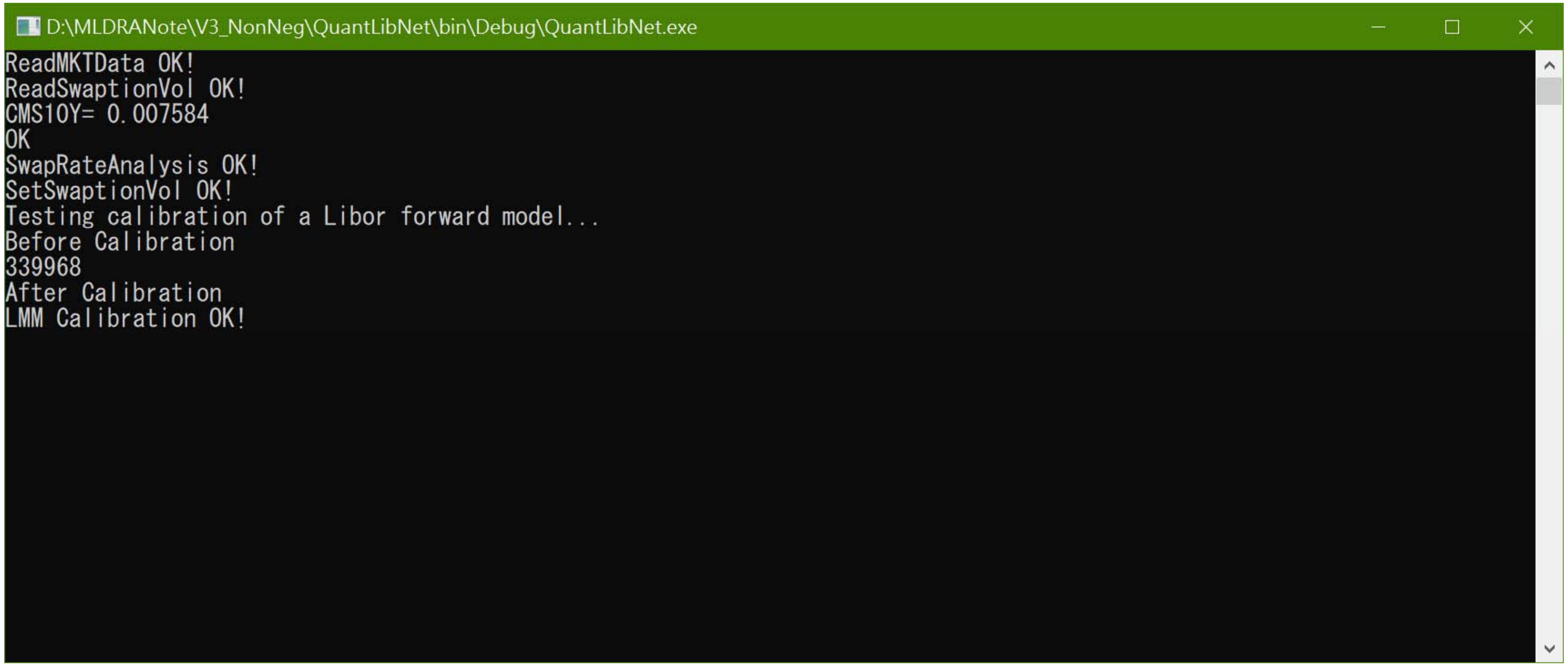
$$\mathrm{S}_{\mathrm{t}}^{10\mathrm{Y}} = \frac{DF_t - DF_{t+40}}{\sum_{i=1}^{t+40} DF_i \times \tau_i - \sum_{i=1}^{t} DF_i \times \tau_i}$$

  - ✓ Quarterly Payment，$\tau_0 = 0$。

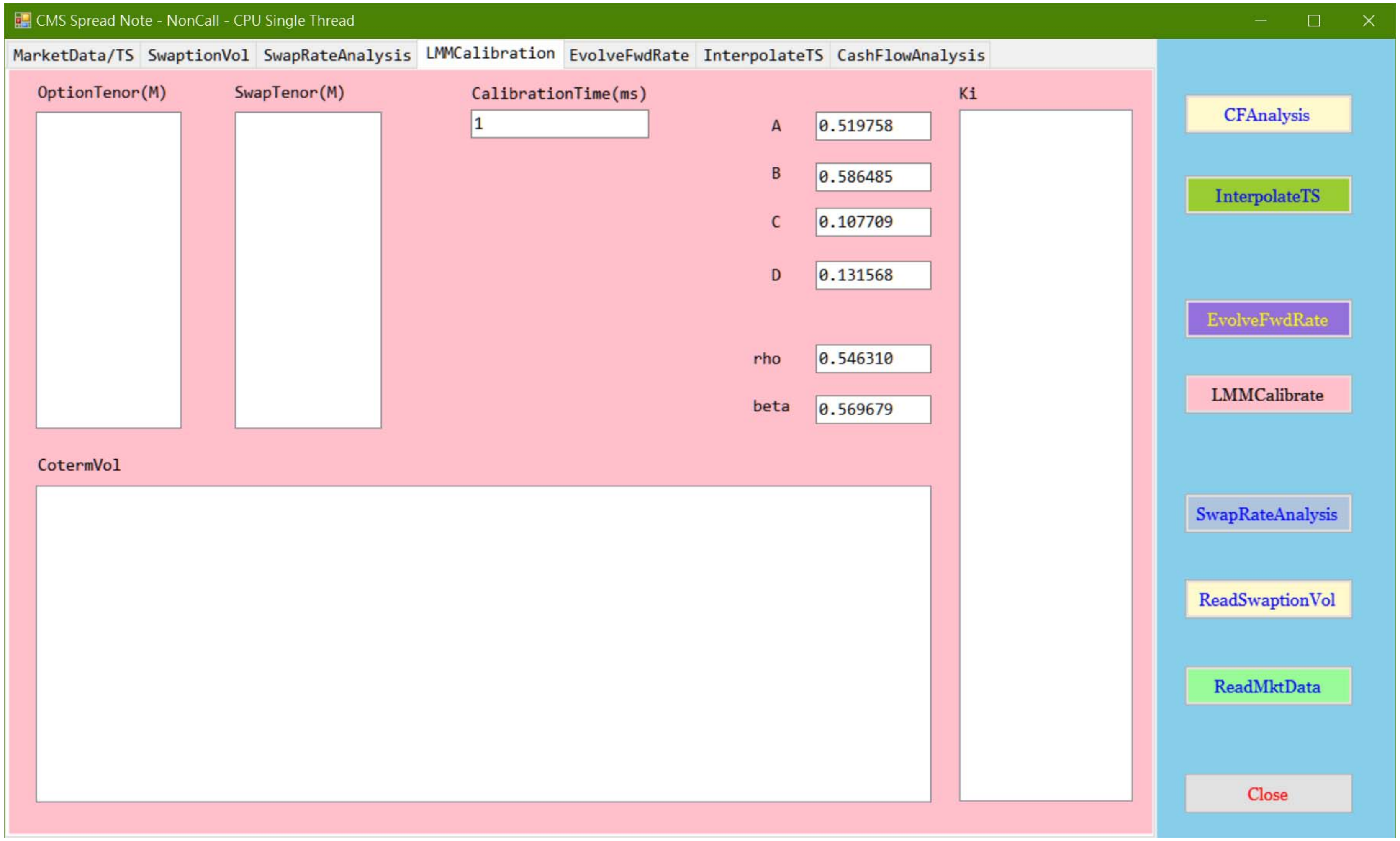
◆ LMMCalibrate 功能，執行結果，80Q 版本



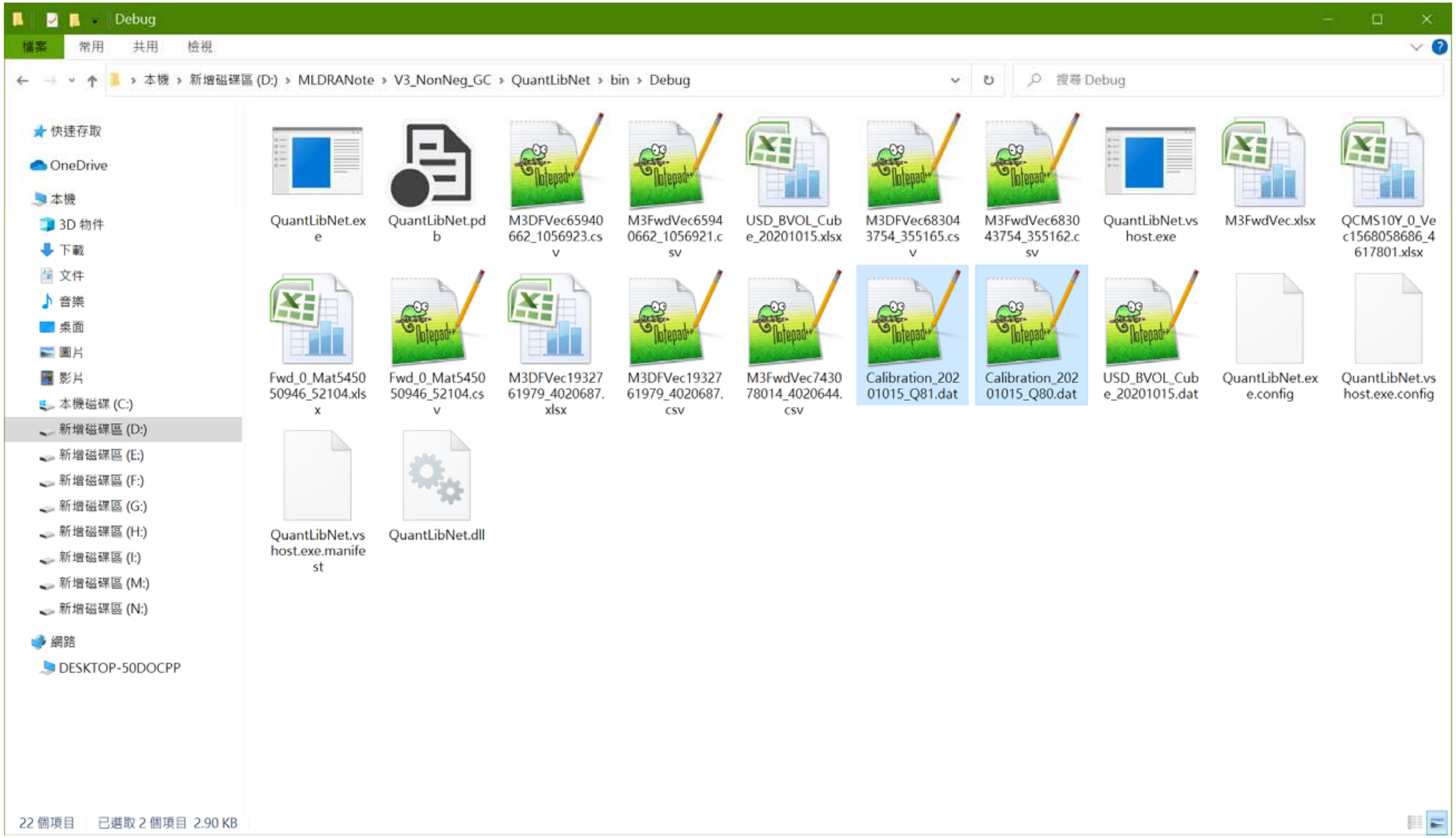
➢ 使用程式執行校正，需要很長時間，340 秒，6 分鐘。



◆ 執行結果，81Q 版本，自檔案讀入已校正的市場參數。



➢ 已經校正出兩個係數檔，80Q 與 81Q。



- 有兩個輸出版本，80Q 與 81Q。
  - 兩者很接近，a, b, c, d 前三個 digits 相同。
  - 80Q，Calibration_20201015_Q80.dat

```
0.5194773100209    => a
0.586672937731144  => b
0.10759785949273   => c
0.131116630648598  => d
1                   => Q1
1                   => Q2
1                   => Q3
1                   => Q4
1.12584146090359
1.10374406375037
1.09482565853241
1.09680044636936
1.15140824419052
1.13763534220798
1.1351790733104
1.14306328950591
…
…
…
4.22679990675235
```

```
3.87180213907463
1.12886500252014    => Q76
1.10385351582301    => Q77
0.22840152153812    => Q78
0.993618783102851   => Q79
1.00621441908949    => Q80
0.54561852495569    => alpha, Long Term Corr.
0.572509000121121   => beta, Beta.
```

➢ 81Q，Calibration_20201015_Q81.dat

```
0.519758391294533    => a
0.586484656602705    => b
0.107709173477166    => c
0.131567790462024    => d
1                     => Q1
1                     => Q2
1                     => Q3
1                     => Q4
1.12772229819777
1.10530700484041
1.09626055226323
1.09826473228478
1.15369686779565
1.13972292638023
1.1372318481531
1.14523282143237
…
…
…
4.27154350997034
3.91142231927671
1.13094926755325      => Q76
1.10639455658716      => Q77
0.217803954232345     => Q78
```

```
0.993510771333513     => Q79
1.00631944281195      => Q80
1                     => Q81
0.546309658288132    => alpha, Long Term Corr.
0.569679245936634    => beta, Beta.
```

◆ EvolveFwdRate 功能，期初模擬，evDate = 2020/10/15，使用 81Q 來模擬。

➢ 81 個 Fwd Rates，走 41 步。

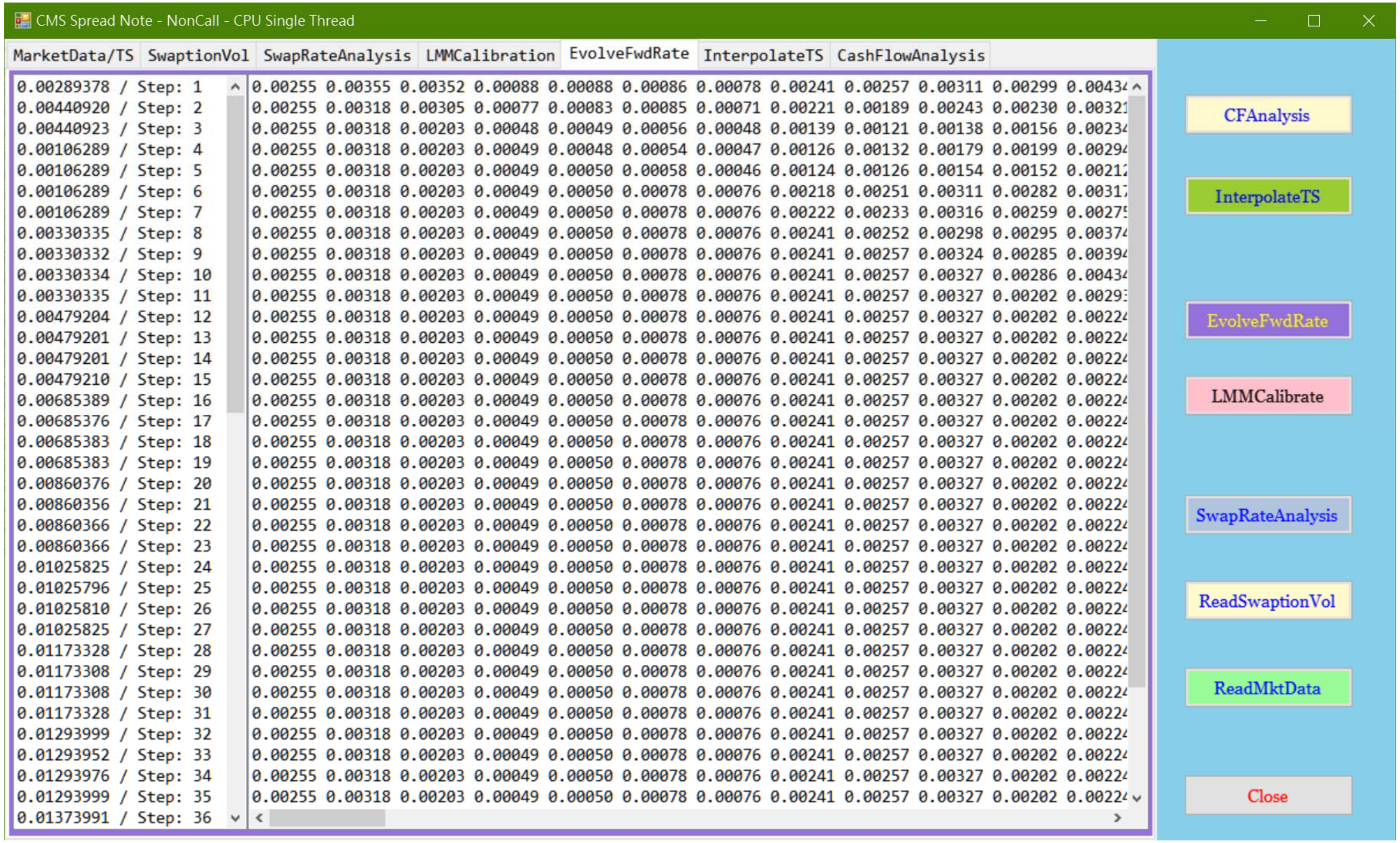
✓ 由 3*6M 開始漂移，0*3M 已經到期。使用 MT，seed=1234。

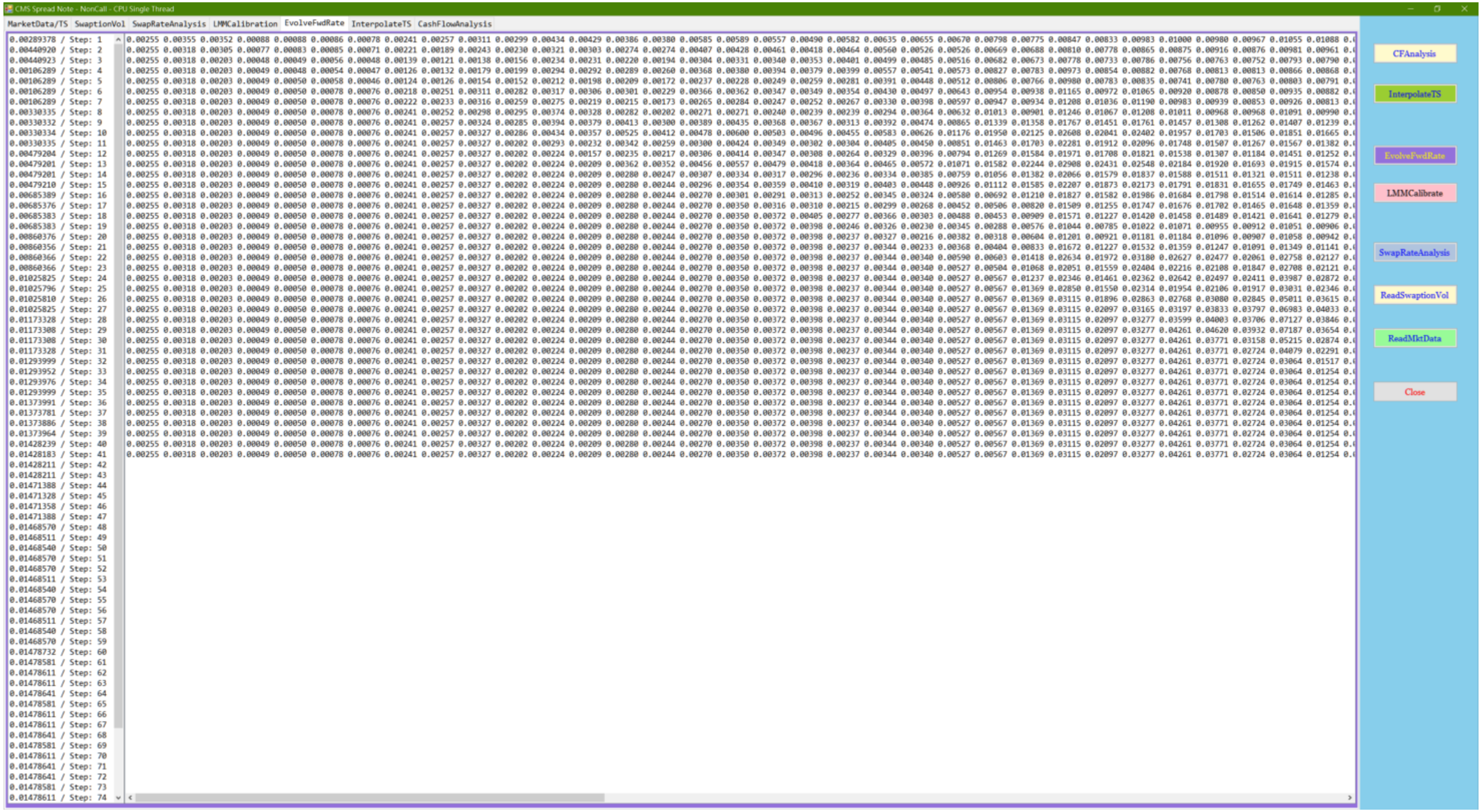
➢ 相關日期

```
IssueDate = new Date(19, 10, 2020); // Note issue date, must be a business date
Maturity = new Date(21, 10, 2030); // Note Maturity date, must be a business date
Today = new Date(15, 10, 2020);
today = Today; // 2020/10/15
mtmDate = evDate; // 2020/10/15
Settings.setEvaluationDate(mtmDate);
calendar = aCal; //new TARGET();
settlement = calendar.advance(mtmDate, settlementDays, TimeUnit.Days);  // 2020/10/19

// 2040/10/19 + 6M = 2041/4/19 (82 Q)
endDate = IssueDate + new Period(246, TimeUnit.Months);
// 20*4 + 2 => 82Q => 83 Dates
```

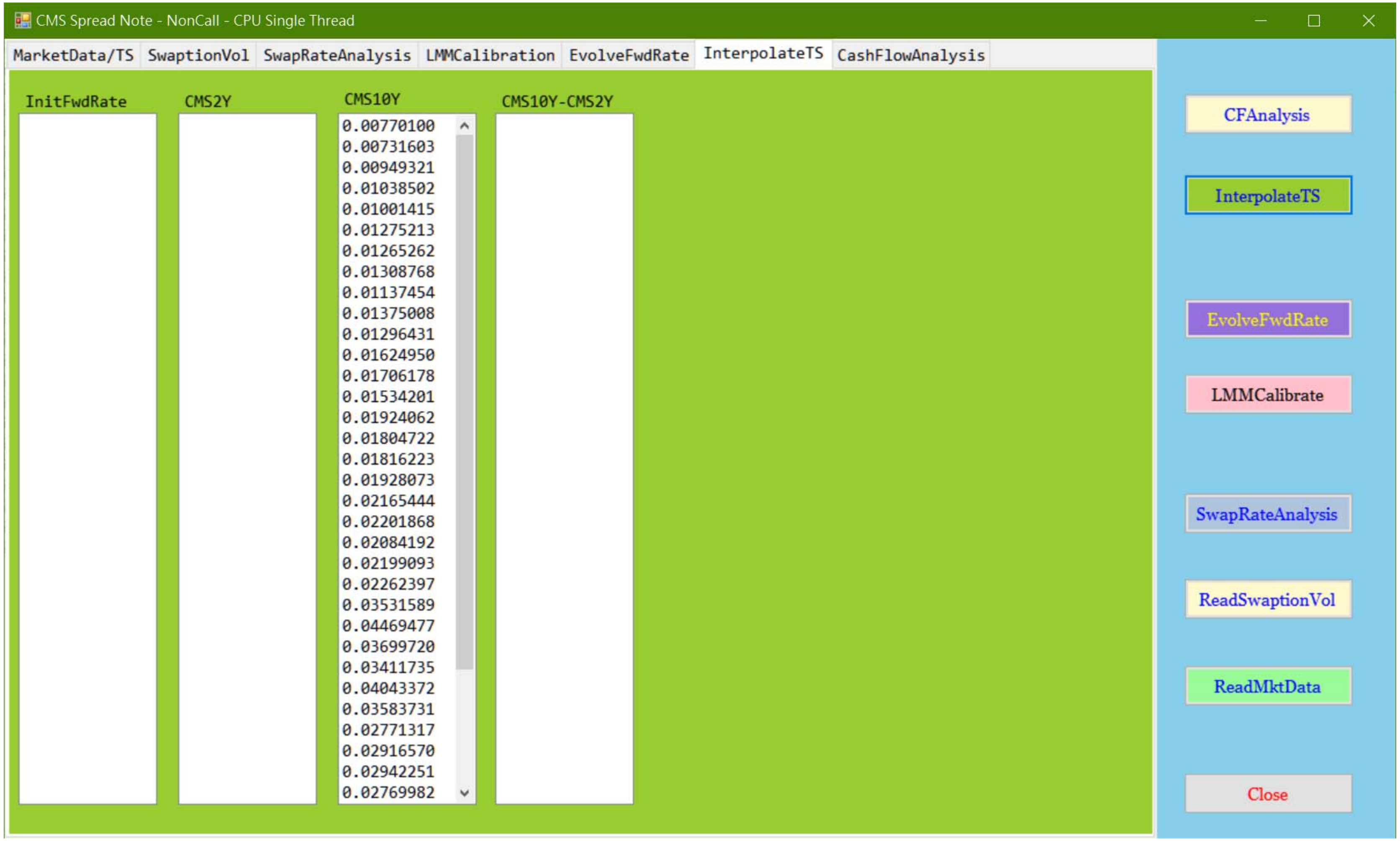
➢ Drift Rate Table



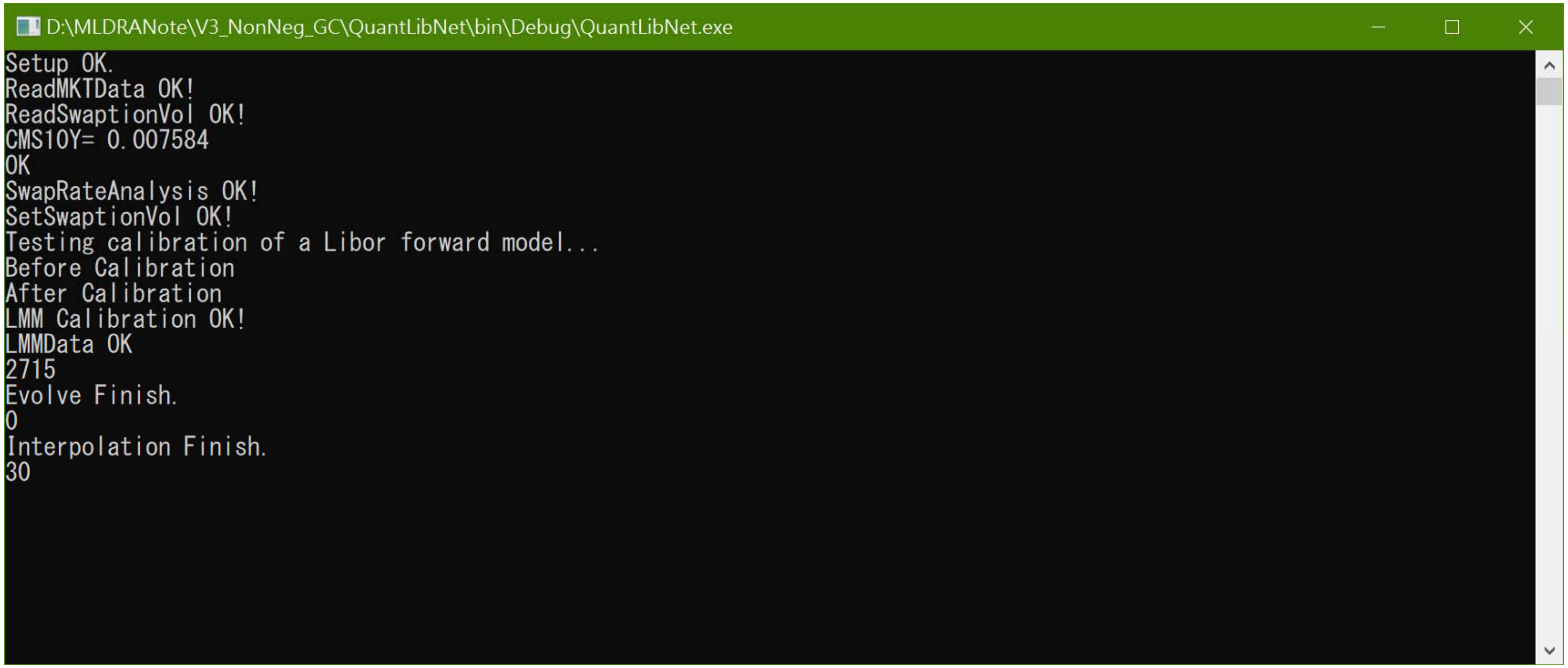

CMS Spread Note - NonCall - CPU Single Thread
MarketData/TS
SwaptionVol
SwapRateAnalysis
LMMCalibration
EvolveFwdRate
InterpolateTS
CashFlowAnalysis
CFAnalysis
InterpolateTS
EvolveFwdRate
LMMCalibrate
SwapRateAnalysis
ReadSwaptionVol
ReadMktData
Close

◆ InterpolateTS 功能，使用節點模擬的遠期利率，計算 CMS10Y。

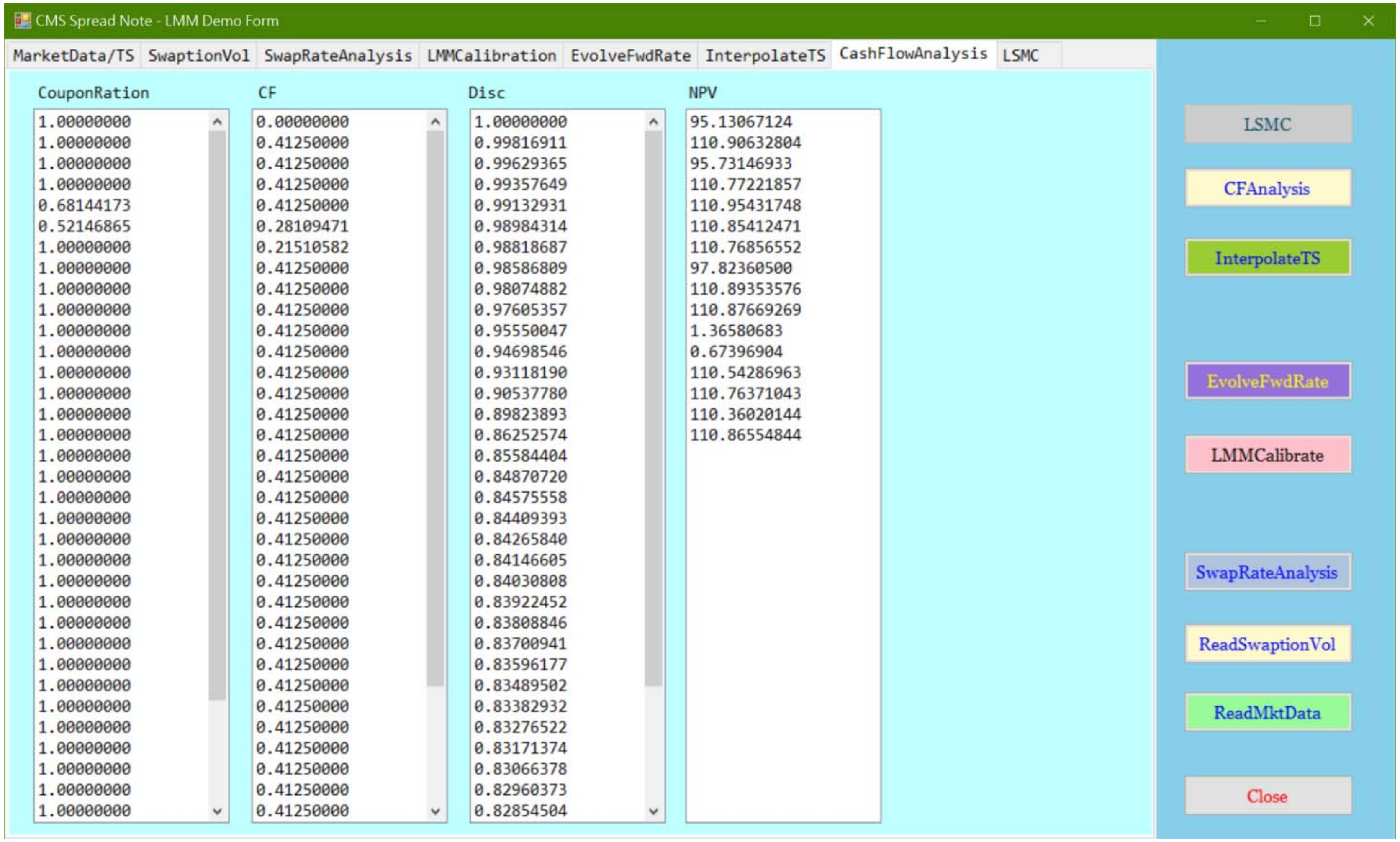
➢ 期初 CMS10Y = 0.007701，有對準市場資料。

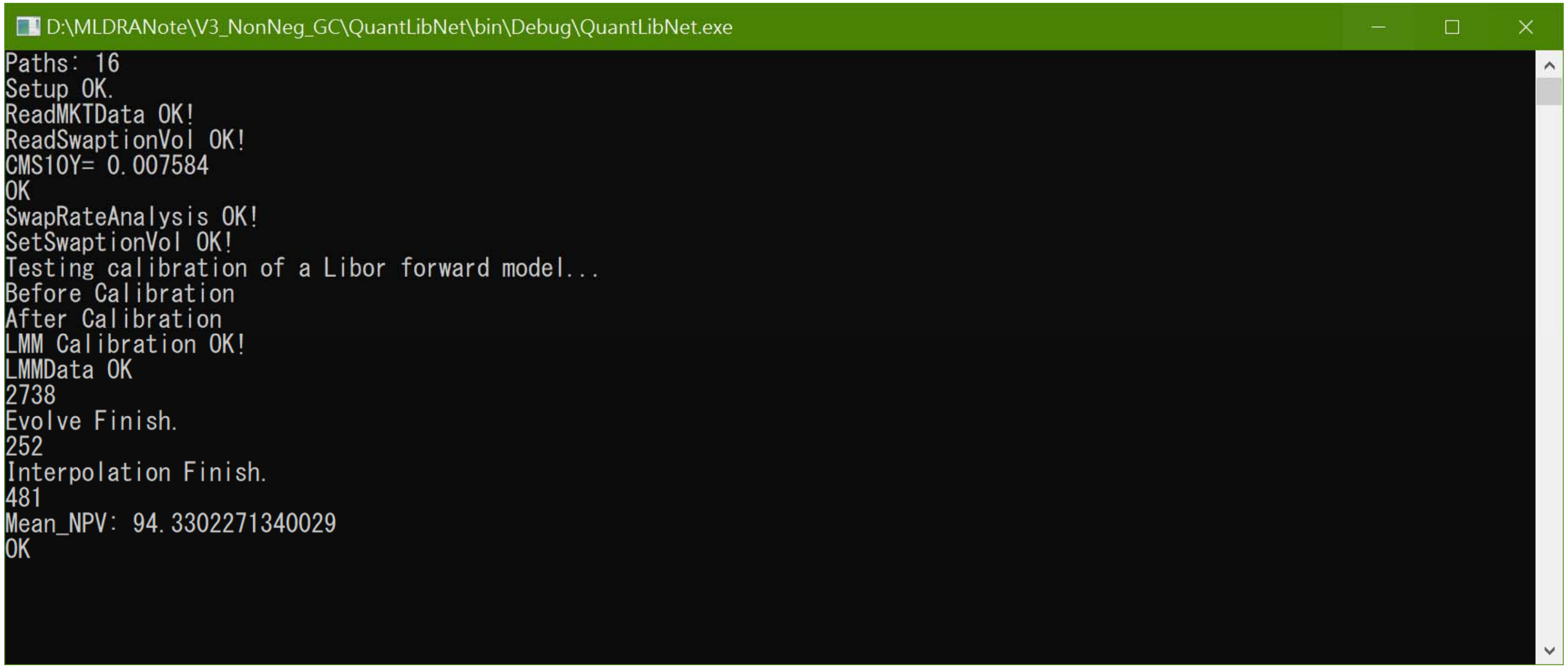


➢ 一條 Path 計算節點的 CMS10Y 只要 0.030 秒。

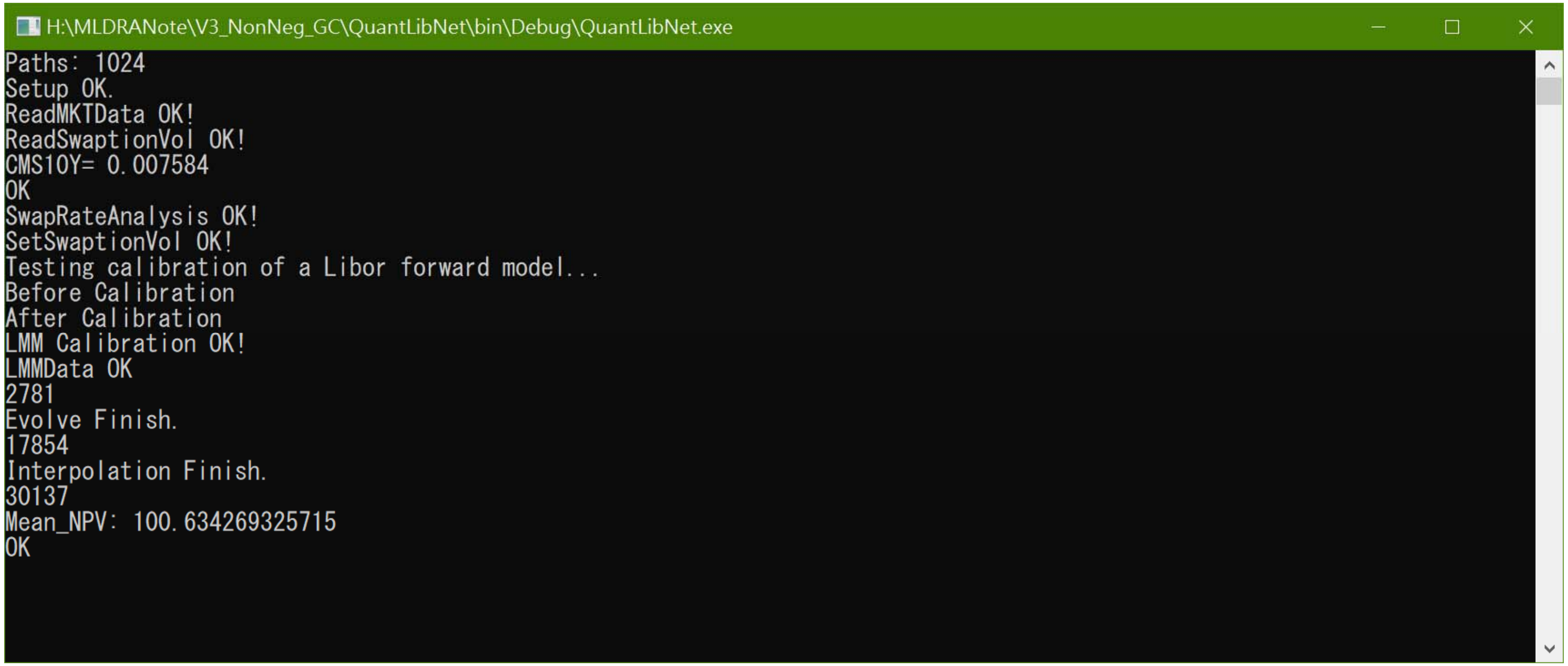


◆ CFAnalysis 功能，16 條，Non-Call，Coupon Rate 1.65%

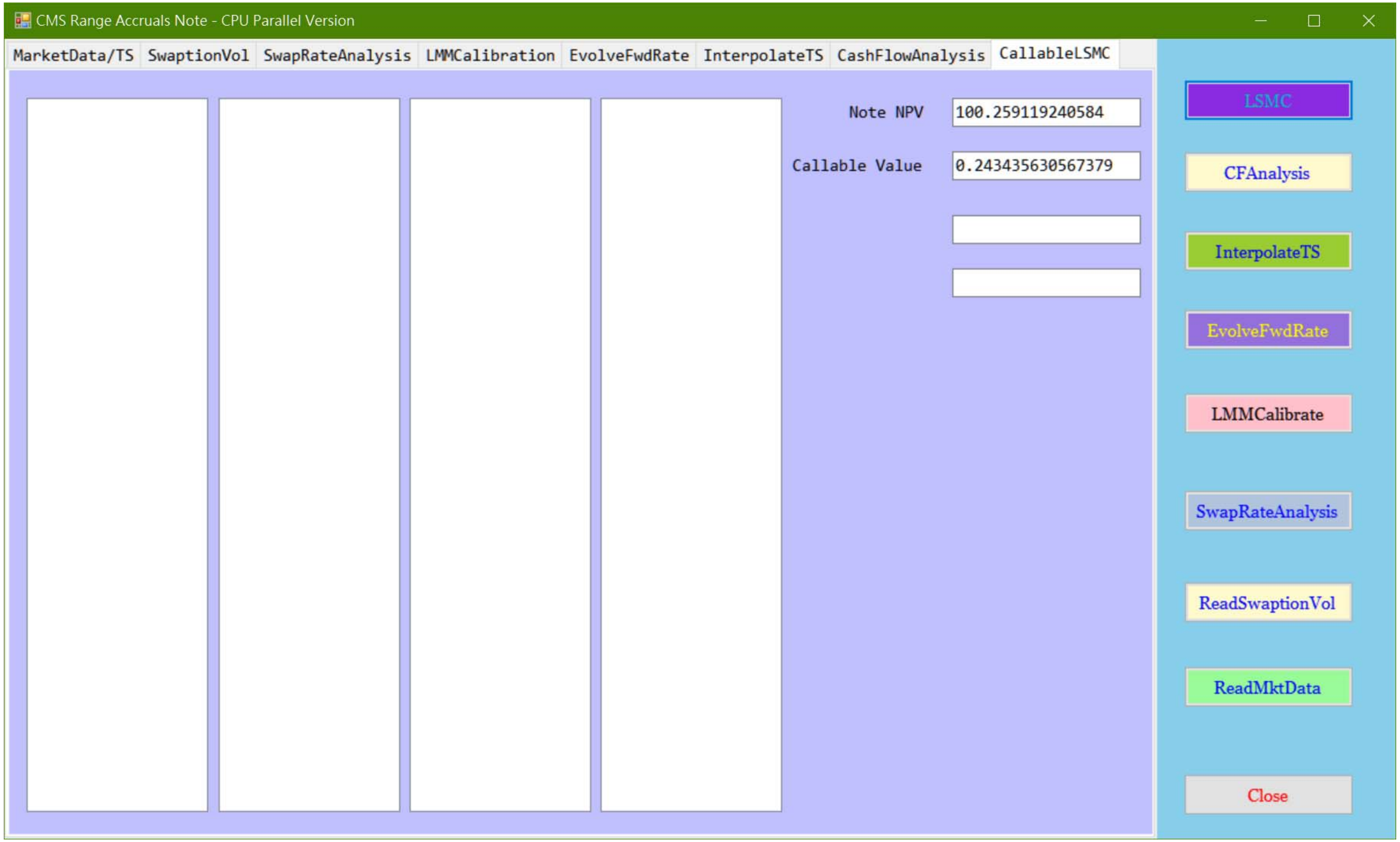


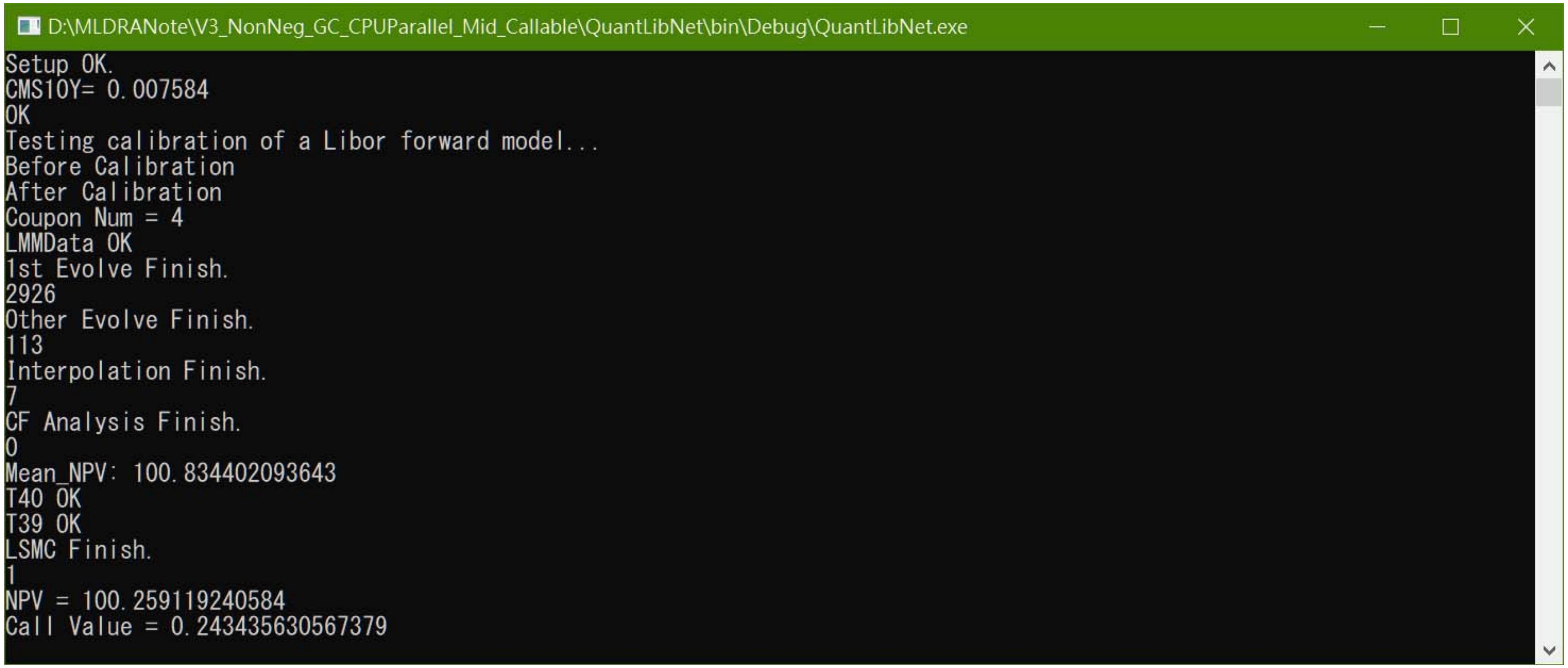

D:\MLDRANote\V3_NonNeg_GC\QuantLibNet\bin\Debug\QuantLibNet.exe
Paths: 16
Setup OK.
ReadMKTData OK!
ReadSwaptionVol OK!
CMS10Y= 0.007584
OK
SwapRateAnalysis OK!
SetSwaptionVol OK!
Testing calibration of a Libor forward model...
Before Calibration
After Calibration
LMM Calibration OK!
LMMData OK
2738
Evolve Finish.
252
Interpolation Finish.
481
Mean_NPV: 94.3302271340029
OK

◆ 1024 條，CouponRate 1.65%



◆ LSMC 功能，期初評價 LSMC，CouponRate = 1.80%，evDate = 2020/10/15




D:\MLDRANote\V3_NonNeg_GC_CPUParallel_Mid_Callable\QuantLibNet\bin\Debug\QuantLibNet.exe
Setup OK.
CMS10Y= 0.007584
OK
Testing calibration of a Libor forward model...
Before Calibration
After Calibration
Coupon Num = 4
LMMData OK
1st Evolve Finish.
2926
Other Evolve Finish.
113
Interpolation Finish.
7
CF Analysis Finish.
0
Mean_NPV: 100.834402093643
T40 OK
T39 OK
LSMC Finish.
1
NPV = 100.259119240584
Call Value = 0.243435630567379

五、GPU OpenCL 高速運算模擬加速實作

- 實務上 2 萬條的模擬路徑為一般的要求
  - 以 CPU 單線程模擬 2 萬條路徑，約需時 20 分鐘計算出 MTM，1X
- 以 CPU 多線程模擬 2 萬條路徑，約需時 2 分鐘計算出 MTM，10X
  - 如果只有少數部位，30 檔以下，可以在 1 小時內處理完成
- SBM 下需要計算 Delta、Curvature、Vega 等敏感性
  - 有 10 個利率風險因子、5 個波動性風險因子要評估
  - 使用差分方式計算
    - ✓ Delta 要計算 10 個值
    - ✓ Curvature Up、Curvature Down 各要計算 10 個值
    - ✓ Vega 要計算 5 個值

◆ 使用 GPU 成為可能的必要方法

- 需要大量的記憶體空間
  - ✓ 大量的隨機亂數要模擬，還有相關性矩陣
- 以 GPU 多線程模擬 2 萬條路徑，約需時 20 秒鐘計算出 MTM，60X
  - ✓ 我的桌機使用 Gefore GTX 1660，6 GB RAM，4224 Cores
  - ✓ 20 年期模擬(80 Factors)
- 12 GB 的 RAM 是必備裝置，24GB 更好
  - ✓ 30 年期模擬(120 Factors)

◆ 作為 Issuer 需要動態管理風險

- 每日 MTM、Greeks 計算
- 每月風險資本申報
- 高速運算勢所難免