Politecnico di Milano

Local Volatility Model

Progetto per l'A.A. 2018/2019

Autori:
Laura Locatelli
Pietro Manzoni
Matteo Paggiaro
Luca Parafioriti

14 Gennaio 2019

1 Ottimizzazione del codice

Per prima cosa, procediamo alla stesura della funzione richiesta. Notiamo che all'interno del codice `example_calibration_ecorp.m` fornitoci a lezione è già presente una funzione `solve_dupire.m`. In realtà la struttura complessiva è più articolata e prevede una chiamata a cascata di più funzioni annidate:

`example_calibration_ecorp.m` → `calibrator.m` → `model_volatility.m` → `solve_dupire.m`

Decidiamo così di ritoccare le ultime due, allegate al progetto con i nomi `model_volatility_mod.m` e `solve_dupire_mod.m`. Nello specifico, le maggiori modifiche apportate riguardano proprio quest'ultima: la funzione originale `solve_dupire.m` riceve in ingresso un tempo di expiry T_1 (oltre a molti altri dati e parametri) e risolve numericamente l'equazione di Dupire nella regione $[k_{min}, k_{max}] \times [0, T_1]$. Tuttavia, durante il processo di calibrazione del modello, si rende necessario calcolare la soluzione per diverse expiries $\{T_1, T_2 \ldots, T_N\}$: qui il codice mostra tutta la sua inefficienza, andando a risolvere ogni volta l'equazione nell'intera regione $[k_{min}, k_{max}] \times [0, T_k]$, quando invece la soluzione è già nota in $[k_{min}, k_{max}] \times [0, T_{k-1}]$ dall'iterazione precedente.

Invece, per quanto riguarda la funzione `model_volatility_mod.m`, le modifiche sono puramente tecniche (e non concettuali) per via del fatto che l'output di `solve_dupire_mod.m` è ora in forma matriciale e non più vettoriale.

Completa infine il trittico la funzione `calibrator_mod.m`, che si differenzia da `calibrator.m` per il semplice fatto di chiamare `model_volatility_mod.m` al posto di `model_volatility.m`.

Da ultimo, ci teniamo a sottolineare due piccole cose:

- ▷ poichè tipicamente le expiries non sono equamente distribuite, la griglia che viene creata da `solve_dupire_mod.m` non è più equispaziata. Infatti il passo di discretizzazione temporale nell'intervallo $[T_{k-1}, T_k]$ è tanto più raffinato quanto più le due scadenze sono vicine nel tempo. Tuttavia riteniamo che ciò non sia assolutamente un problema in termini di affidabilità o cattiva approssimazione della soluzione.
- ▷ il codice originale `solve_dupire.m` mantiene in ogni caso la propria utilità nei casi in cui sia necessario risolvere l'equazione di Dupire per un'unica data di expiry, come si osserverà al punto successivo.

2 ECORP

2.1 Calibrazione dei dati

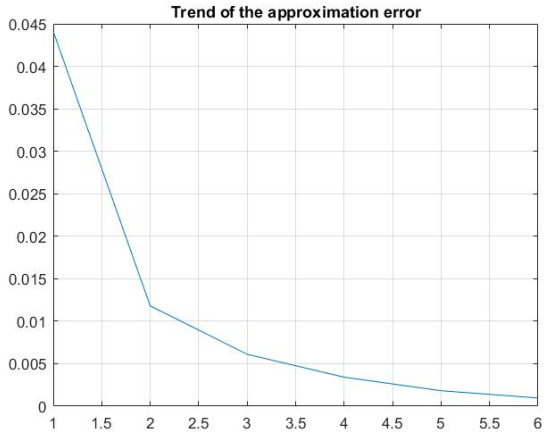
Consideriamo adesso i dati di mercato estratti dal dataset `ECORP`.
All'interno del file `point2.m`, utilizzando il codice ottimizzato `calibrator_mod.m` sviluppato nell'esercizio precedente, calibriamo il modello con il comando:

```
[V] = calibrator_mod(T,Knorm,MktVol,tol,MaxIter, N,M,Kmin, Kmax, 'cn');
```

La funzione restituisce la matrice `V`, legata alla volatilità $\eta(t,x)$ del modello attraverso la relazione:

$$V_{ij} = \eta(T_i, K_{ij}) \tag{1}$$

Il grafico dell'errore di approssimazione mostra come la calibrazione sia di buona qualità: all'aumentare del numero di iterazioni la volatilità spiegata dal modello si avvicina sempre di più alla volatilità del mercato.



2.2 Pricing di opzioni Call

Si vuole adesso calcolare il prezzo di due Call Option (EU) con Maturity $T = 0.5$ anni e Strike $K_1 = 0.9S(0)$ e $K_2 = 1.1S(0)$.
Ottenuta la volatilità $\eta(t,x)$ del modello con la calibrazione precedente, è possibile calcolare i prezzi delle call attraverso la risoluzione numerica dell'**equazione di Dupire**:

$$\frac{\partial}{\partial T}c_0(T,k) = \frac{1}{2}\eta^2(T,k)k^2\frac{\partial^2}{\partial k^2}c_0(T,k) \tag{2}$$

L'equazione di Dupire esprime un legame, in termini di equazione differenziale alle derivate parziali, tra $\eta(t,x)$ e i prezzi delle call normalizzate $c_0(T,k)$. Per risolvere numericamente l'equazione facciamo uso della function `solve_dupire.m`.
Attraverso la relazione

$$C_0(T,K) = D_0(T)F_0(T)c_0(T,k)$$

si possono poi riscalare i prezzi calcolati, ottenendo:

$$C_0(T, K_1) = 286.6985 \qquad \sigma^{BS}(T, K_1) = 11.93\%$$
$$C_0(T, K_2) = 4.3049 \qquad \sigma^{BS}(T, K_2) = 8.31$$

Inoltre, per completezza, decidiamo di implementare una simulazione con il metodo di Monte Carlo in modo da ottenere un'approssimazione del prezzo delle due call options. Per quanto questo approccio stocastico porti ogni volta ad un risultato numerico diverso, lanciando qualche volta il codice si può apprezzare come i valori ottenuti tendano ad essere ragionevolmente vicini ai prezzi calcolati.

2.3 Simulazione di Monte Carlo

Forward Starting Options Una forward starting option con date T_1 e $T_2 > T_1$ è un opzione con payoff dato da:

$$\Phi(T_1, T_2) = (S(T_2) - kS(T_1))^+ \tag{3}$$

con $k > 0$. Il prezzo dell'opzione all'istante T_1 è dato da:

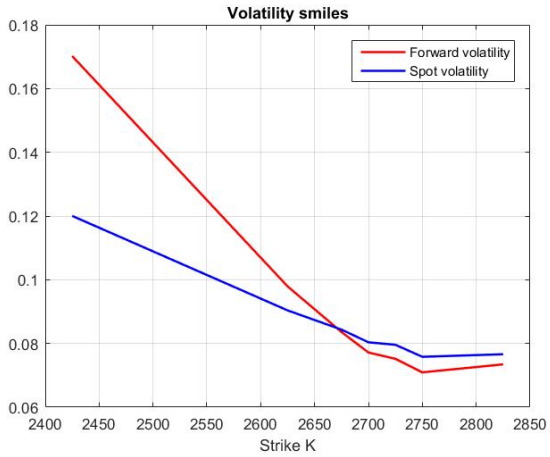
$$C_{T_1}(T_1, T_2, k) = D_{T_1}(T_2)\mathbb{E}[\Phi(T_1, T_2)|\mathcal{F}_{T1}] \tag{4}$$

Pricing Come già fatto nel punto precedente, eseguiamo una simulazione di Monte Carlo per stimare i prezzi di due forward options aventi $T_1 = 2$, $T_2 = 2.5$ e $k_1 = 0.9$, $k_2 = 1.1$, ottenendo i seguenti risultati.

$$C_{T_1}(T_1, T_2, k_1) = 298.12 \qquad \sigma^{BS}_{fwd}(T_1, T_2, k_1) = 16.53\%$$
$$C_{T_1}(T_1, T_2, k_2) = 32.77 \qquad \sigma^{BS}_{fwd}(T_1, T_2, k_2) = 15.15\%$$

2.4 Volatility smile

Utilizzando la calibrazione del modello fatta nel punto 2.1, otteniamo le due curve *spot* e *forward* della volatilità:



Il grafico evidenzia una caratteristica tipica del *Local Volatility Model*, ossia il fatto che la curva della volatilità *forward* abbia una pendenza maggiore di quella *spot*.

2.5 La caduta dei giganti

A questo punto della trattazione, supponiamo di scoprire che in realtà la forward starting option di cui ci siamo appena occupati esiste veramente. Con trepidazione andiamo a controllare la sua quotazione e ci accorgiamo che è del tutto differente dal prezzo che abbiamo calcolato. Dovevamo e/o potevamo aspettarcelo?

In questo breve paragrafo daremo spazio ad alcune riflessioni riguardanti l'affidabilità e la verosimiglianza del *Local Volatility Model*.

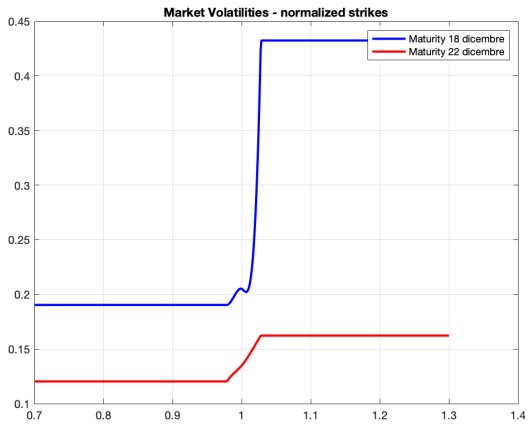
Per cominciare, ragioniamo sull'origine stessa del modello. Esso nasce come una modifica del modello Black and Scholes, nel tentativo di ovviare, probabilmente, a quella che è la più grande e pretenziosa ipotesi: l'introduzione di un parametro di volatilità costante. Ciò che era stata la fortuna di Black and Scholes -grazie al fatto di poter offrire delle formule chiuse per il pricing- ne aveva al contempo decretato l'inutilità pratica, dal momento che gli andamenti del mercato erano tutto fuorchè aderenti a ciò che era previsto da tale modello. Pertanto l'introduzione di una *volatility surface* sembra a primo impatto essere decisamente un buon compromesso. D'altra parte un'analisi critica suggerisce che anche il *Local Volatility Model* abbia i propri limiti strutturali.

Più nel dettaglio, ci siamo soffermati su tre aspetti che a nostro parere rappresentano i punti deboli del modello:

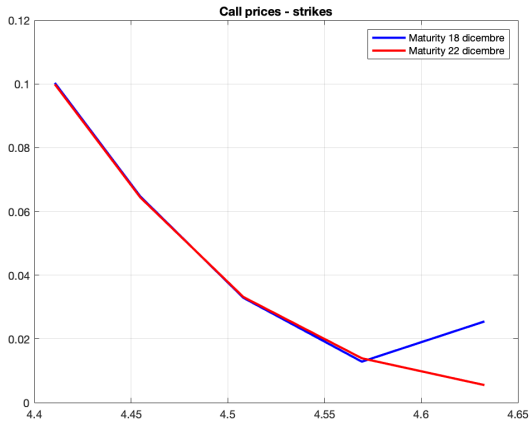
- ▷ come spesso capita in queste situazioni, si cerca di estrapolare informazioni partendo da un set finito di dati. Nel nostro caso, abbiamo fatto largo uso di interpolazione (lineare e con spline) per risalire ai dati mancanti e, per quanto inevitabile, ciò è tipicamente fonte di errore.
- ▷ la necessità matematica di ricorrere al caso. Allo stesso modo infatti, anche l'utilizzo del metodo Monte Carlo ha luci e ombre. Si tratta sicuramente di un algoritmo semplice e facilmente applicabile ai nostri problemi, tuttavia l'aleatorietà del metodo non garantisce per nulla accuratezza nei risultati.
- ▷ infine, uno dei punti cruciali di questo modello (come di moltissimi altri), è il tentativo di spiegare completamente i prezzi dei titoli tramite la loro volatilità. Nonostante infatti l'intuizione e l'evidenza empirica suggeriscano che ci sia grande interdipendenza tra le due cose, non ci si può aspettare di avere una completa correlazione.

3 FAIL

Consideriamo adesso i dati presenti nelle tabelle `FAIL`. All'interno del file `point3.m` utilizziamo, come in precedenza, il comando per la calibrazione del modello.
L'algoritmo non produce nessun risultato e restituisce un messaggio di errore. Analizzando i grafici delle volatilità di mercato in funzione delle strikes normalizzate, si osserva che nel caso della call con maturity 18 Dicembre stranamente non si ottiene una funzione convessa.



L'anomalia per quest'ultima call si ripropone nell'analisi del prezzo delle Call in relazione alle Strikes, ove il suo grafico non presenta andamento decrescente (come invece avviene per la call con maturity 22 dicembre).



Quest'ultima osservazione mette in luce un'opportunità di **arbitraggio**, che illustreremo brevemente.

Costruzione dell'arbitraggio Consideriamo i valori delle stikes $k_1 = 4.5693 < k_2 = 4.6327$, con i relativi prezzi della call con maturity 18 dicembre: $C(k_1) = 0.0129 < C(k_2) = 0.0255$.

- $t = 0$: adottiamo una *long position* nella call con stike k_1 e una *short position* nella call con strike k_2, investendo la differenza $C(k_2) - C(k_1)$ in banca.
  In questo modo abbiamo un flusso netto di denaro pari a
  $$-C(k_1) + C(k_2) - (C(k_2) - C(k_1)) = 0$$
- $t = T$: si registra un flusso di denaro pari a
  $$(S(T)–k_1)^+ - (S(T)–k_2)^+ + (C(k_2) - C(k_1))e^{rT} > 0$$
  poichè $k_1 < k_2$ e $C(k_1) < C(k_2)$.

Abbiamo quindi costruito un arbitraggio, tramite una strategia che senza alcun rischio comporta un portafoglio il cui valore iniziale è nullo, mentre alla maturity è sicuramente positivo.
Il dataset FAIL non rispetta quindi l'ipotesi di non-arbitraggio del modello teorico adottato; questo spiega il conseguente fallimento dell'algoritmo numerico.

4 EURUSD

Analizziamo adesso i dati di mercato dei tassi di cambio presenti nel database `EURUSD`.

4.1 Calibrazione del modello

In generale le opzioni sui tassi di cambio non sono caratterizzate dagli Strike Price ma dal Δ, ovvero dalla sensibilità di Call Option EU scritte sui tassi rispetto ad una variazione di prezzo dei tassi stessi, definita nel modello Black & Scholes come:

$$\Delta = \frac{\partial C_t(T)}{\partial X_t} = \mathcal{N}(d_1) \tag{5}$$

Esiste un legame biunivoco tra Δ e gli strike K. Il parametro d_1 dipende infatti dallo strike K, come mostrato nella seguente formula:

$$d_1(K) = \frac{\log \frac{F_0(T)}{K} + \frac{\sigma(T,\Delta)}{2}}{\sigma(T,\Delta)\sqrt{T}} \tag{6}$$

È possibile quindi calcolare il valore degli strike $\{K_{ij}\}$ in funzione di $\{T_i\}$ e $\{\Delta_j\}$ risolvendo numericamente l'equazione

$$\mathcal{N}(d_1(K)) = \Delta$$

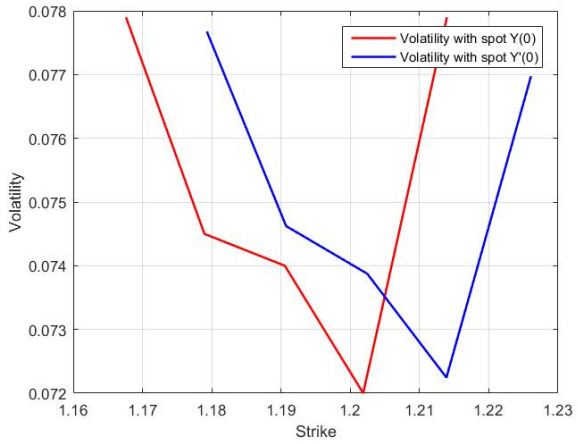
Calcolati i valori degli strike prices procediamo poi con la calibrazione del modello, utilizzando ancora una volta la function `calibrator_mod.m`.

4.2 Modifica dello spot price

Siccome i prezzi *spot* dell'azione e del FX cambiano in continuazione durante un normale giorno di attività finanziaria, viene da chiedersi come cambia la *volatility surface* del modello al variare del prezzo *spot*. Supponiamo per esempio che avvenga il seguente *shift*:

$$Y'(0) = 1.01 \times Y(0)$$

Ricalcolando i prezzi aggiornati $F_0'(T)$ dei *forward* e stimando K con la stessa procedura di prima, abbiamo tutti i parametri che servono per calibrare il modello. Otteniamo le seguenti curve di volatilità, calcolate rispettivamente con prezzi iniziali $Y(0)$ e $Y'(0)$.



Dal grafico si osserva che il modello si comporta come *sticky-delta*. Si vede infatti come la nuova *volatility surface* ottenuta con $Y'(0)$ sia una traslazione della prima. La curva della volatilità rimane quindi immutata rispetto al Δ delle opzioni, mentre varia rispetto al loro strike.