Pricing of Electricity Derivatives

Matteo Gardini*

1 Luglio 2019

Sommario

In questo documento presentiamo un modello per la simulazioni dei prezzi spot dell'energia elettrica che può essere utilizzato per il pricing di derivati. Questo modello è un'estensione di quanto presentato in [9]. Vengono illustrate tecniche di calibrazione dei parametri, di smoothing della curva forward e di simulazioni Monte Carlo, necessarie per il pricing di opzioni esotiche.

1 Pricing di Derivati sull'Energia Elettrica

Dall'inizio della deregolarizzazione dei mercati elettrici le pricipali compagni elettriche hanno dovuto fronteggiare rischi ed incertezze sull'andamento dei prezzi sconosciuti per altri tipi di commodity. I mercati elettrici esibiscono, difatti, caratteristiche peculiari tra le quali un'alta volatilità, la presenza di frequenti salti ed un andamento di tipo stagionali con prezzi più elevati di inverno e d'estate e più contenuti in autunno e primavera. Una delle principali strategie per mitigare il rischio di fluttuazione dei prezzi è ricorrere all'utilizzo di strumenti derivati complessi. Tuttavia, per utilizzare in maniera sicura e consapevole questi strumenti è necessario avere un'ottima conoscenza delle proprietà statistiche dei mercati dell'energia elettrica ed è fondamentale ricavare un modello di pricing atto a valutare il valore dei contratti derivati.

Nel corso dell'utimo decennio sono stati proposti diversi modelli utilizzabili a tal fine. La maggior parte di essi sono in grado di catturare dinamiche mean reverting, spikes di prezzo e stagionalità. Per ulteriori riferimenti gli articoli principali sono [9], [3] e [4].

Il documento seguente è sviluppato come segue: nella Sezione 2 è presentato il modello. Nella Sezione 3 sono illustrate le metodologie di calibrazione e fitting dei parametri del modello mentre, nella 4, è presentato un metodo molto semplice per lo smoothing della curva forward basato su spline di secondo grado. Nella Sezione 6 sono elencati gli algoritmi di simulazione necessari per eseguire simulazioni Monte Carlo usando il modello sviluppato. Infine, nella sezione 7 sono riportati alcuni risultati numerici che riguardano sia le prestazioni del modello sia il suo utilizzo per il pricing di derivati.

*PhD Report: Maggio 2019, ERG Spa, Genova. per commenti e suggerimenti scrivere a gardini@dima.unige.it

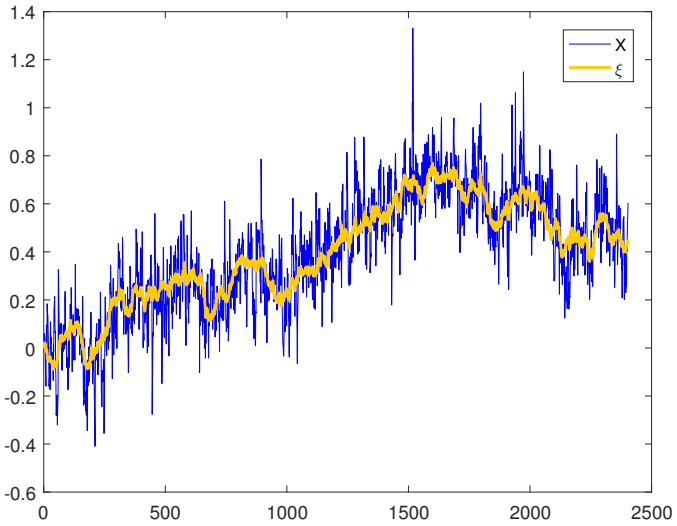

Figura 1: Traiettoria del processo X_t. Da qui si nota che il processo risultante è il processo χ_t che ha un comportamento mean-reverting attorno al processo ξ_t

2 Il modello

Per lo sviluppo del nostro modello proponiamo quanto proposto inizialmente in [6] e ripreso in [9].
Consideriamo le seguenti dinamiche:

$$\begin{aligned} d\chi_t &= -k\chi_t dt + \sigma_\chi dW_\chi + dJ_t \\ d\xi_t &= \sigma_\xi dW_\xi \end{aligned} \tag{1}$$

con $\mathbb{E}\left[dW_\chi dW_\xi\right] = 0$ e dove J_t è la parte di salto ipotizzando i salti distribuiti come una normal $\mathcal{N}\left(\mu_J, \sigma_J^2\right)$ e con intensità λ. Questo tipo di modello è un modello a due fattori. La dinamica in χ tenta di modellizzare la componente di breve termine, molto più volatile, mean-reverting e con la presenza di salti mentre la dinamica in ξ vuole rappresentare la dinamica di lungo periodo meno volatile e non mean-reverting. L'assunzione di correlazione nulla nella componente browniana è suggerita in [9] ma non assunta in [6]. I processi χ e ξ vengono sommati e viene definito il nuovo processo X_t:

$$X_t = \xi_t + \chi_t$$

Il processo X_t risulta essere un processo stocastico con salti che ha un comportamento mean-reverting sulla dinamica di lungo periodo. Una possibile traiettoria è mostrata in Figura 1.

Come è noto la dinamica dei prezzi dell'enegia elettrica esibisce anche una certa stagionalità che presenta prezzi più alti in inverno ed estate e più bassi in inverno. Oltre alla componente di macrostaginalità è presente una componente di micro-stagionalità legata alla variazione di domanda di energia nel corso

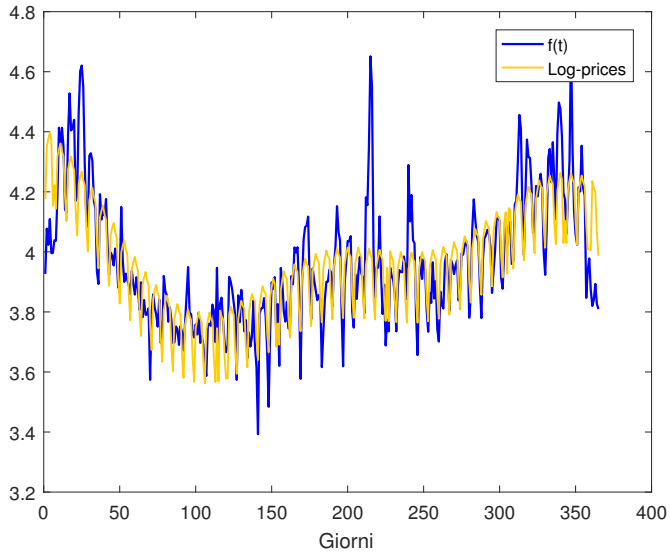

Figura 2: Possibile forma della componete stagionale $f(t)$ per il prezzo PUN dell'anno 2017.

della settimana. Storicamente si osserva che i prezzi dell'energia elettrica sono più bassi durante i weekend e nelle festività. Per questo motivo noi considereremo una microstagionalità all'interno della singola settimana. E' possibile considerare una stagionalità stocastica ma, ai nostri fini, noi considereremo una stagionalità deterministica indicata con $f(t)$ che comprende sia gli effetti di macro che di micro-stagionalità. Considereremo, quindi, come suggerito in [9]:

$$f(t) = s_1 \sin(2\pi t) + s_2 \cos(2\pi t) + s_3 \sin(4\pi t) + s_4 \cos(4\pi t) + s_5 t + \sum_{i \in N_d} w_i \mathbb{1}(t) \tag{2}$$

dove $N_d \in \{lun, mar, mer, gio, ven, sab, dom\}$ e dove t è il tempo espresso in frazione d'anno. Una possibile forma della componente deterministica $f(t)$ è mostrata in Figura 2.

Il prezzo spot, indicato con $S(t)$ viene quindi modellizzato, seguendo l'approccio proposto in [10], come:

$$S_t = \exp\{f(t) + X_t\} \tag{3}$$

2.1 Significato dei parametri

Consideriamo il modello appena definito in (1). Ognuno dei parametri ha un chiaro significato economico. In particolare:

- k: è la velocità di rientro del processo verso la media: maggiore è e maggiore è forza con cui il processo viene richiamato verso la media.
- σ_χ: è la volatilità del processo di breve periodo. Maggiore essa è e maggiore è volatile la traiettoria.

- J_t: è la component di salto i cui salti sono distribuiti come una normale $\mathcal{N}\left(\mu_J, \sigma_J^2\right)$. I salti avvengono con intensità λ. Questa è la parte responsabile delle presenza di spikes nelle simulazioni di prezzi.

- σ_ξ: è la volatilità del processo di lungo periodo. Maggiore essa è e maggiore è volatile la traiettoria e maggiormente si apre il fascio di simulazioni al passare del tempo.

Notiamo, che il significato di questi parametri è molto chiaro e che quindi ognuno di essi può essere arbitrariamente imposto per consentire, all'occorrenza, analisi di tipo what-if.

3 Calibrazione

Una volta definito un modello, la parte veramente complessa è la sua calibrazione. La parte di calibrazione è spessa densa di insidie per numerosi motivi. In primis perchè bisogna trovare una metodologia, ogni volta differente, atta a stimare i parametri del modello. Nel corso degli anni sono stati proposti diversi algoritmi tra cui il metodo di Massima Verosimigliana (MLE), il metodo dei Momenti Generalizzato (GMM), minimi quadrati non lineare etc. Usando questi metodi ci si scontra principalmente con due problemi.
Il primo è che sono metodi numerici e quindi, operando in aritmetica finita, vengono introdotti necessariamente errori. Questi errori, all'interno di una procedura di stima, geneneralmente basata su algortimi di minimizzazione, può avere effetti devastanti. Ricordiamo che, se il problema di minimizzazione non è lineare possono esistere più minimi locali e quindi si potrebbe avere la convergenza verso un set di parametri che non è quello ottimo. Questo problema può essere superato tramite algoritmi di tipo evolutivo che sono però meno precisi e molto più lenti.
Il secondo problema con cui ci si scontra è la disponibilità dei dati. Spesso le serie storiche presentano outliers, buchi o composte di pochi dati. Outliers e buchi possono comportare errori nella stima dei parametri mentre la presenza di pochi dati porta ad un problema di bontà della stima.
Questo è il motivo per cui è spesso inutile definire un modello con numerosissimi parametri che poi non si è in grado di stimare. Spesso è meglio accontentarsi di un modello semplificato che, pur non catturando tutti i fattori di mercato, consente una buona stima dei parametri.

3.1 Stima dei parametri del processo X_t

Consideriamo la serie storica di prezzi powe spot giornalieri dell'anno 2017 indicata con P_t. Calcoliamone il logaritmo $p_t = \log\left(P_t\right)$ ottenendo una serie come quella mostrata in Figura 3.

Prima di eseguire la stima dei parametri di X_t dobbiamo rimuovere dalla serie p_t la stagionalità. Dobbiamo quindi stimare la funzione di stagionalità $f\left(t\right)$ definita in (2). Per far questo stimiamo i parametri s_1,s_2, s_3, s_4, s_5, w_i, tramite regressione lineare della funzione $f\left(t\right)$ contro i prezzi p_t. Otteniamo poi la serie destragionalizzata dei log-price, indicata con x_t, come:

$$x_t = p_t - f\left(t\right) \tag{4}$$

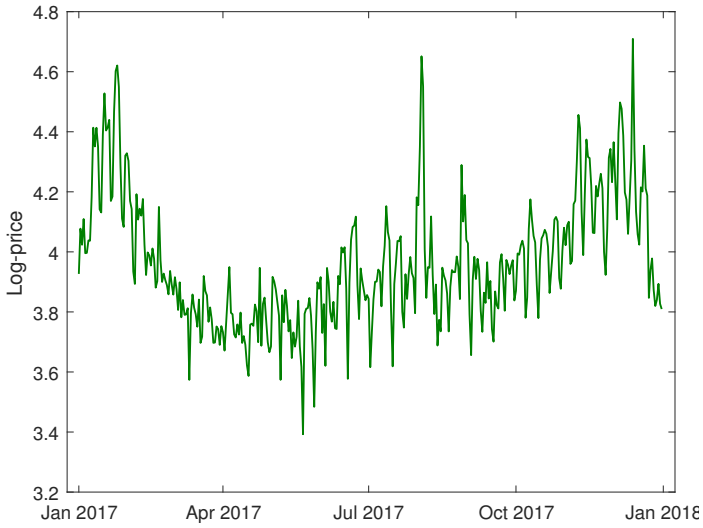

Figura 3: Log-price dei prezzi del power spot Italia per l'anno 2017. Si osserva che è presente una forte componente di stagionalità.

che abbiamo rappresentato in Figura 4 dove si osserva che la componente di stagionalità è stata rimossa.

Una volta rimossa la stagionalità è possibile quindi stimare i parametri del processo X_t di cui abbiamo una realizzazione, indicata con x_t. Osserviamo che, avendo supposto $\rho_{\chi}\xi = 0$ ci consente di stimare separatamente la componente di short-term χ e long-term ξ. In particolare stimeremo la componente di short-term χ dai dati spot e la componente di long-term ξ dalle opzioni quotate a mercato su EEX corrispondenti al prodotto forward corrispondete al periodo spot di simulazione. Per la stima di ξ un'alternativa è basarsi sulla serie storica delle quotazioni del prodotto forward corrispondente al periodo spot di cui si intender avere la simulazione. Ad esempio, se oggi fosse il 17 giugno 2019 e volessimo simulare lo spot di Settembre 2019 per la componente di lungo periodo potrei usare o le opzioni scritte sui forward di settembre o le quotazioni del prodotto forward Settembre 2019. Se abbiamo a disposizione un mercato di derivati sul prodotto forward mese settembre 2019 per ottenere un pricing consistente con questo mercato dovremo stimare la componente di lungo termine ξ a partire dalle quotazioni delle opzioni sui forward. Siccome vogliamo ottenere pricing consistenti con il mercato EEX scegliamo, in questo documento, questa seconda strada. La calibrazione sulle quotazioni storiche è ancora più semplice e ci si può riferire a [2].

Per la calibrazione della parte di lungo termine χ osserviamo che l'unico parametro da stimare è la volatilità σ_{χ}. Date le quotazioni delle opzioni Europee sul lungo periodo da esse è possibile estrarre la volatilità implicita invernendo la formula di Black ed ottenere così il parametro σ_{χ}. Numerosi software di calcolo presentano algoritmi già implementati per estrarre la volatilità implicita del modello di Black (in *MATLAB* è presenta la funzione `blkimplvol`) ma è possibile implementare un semplice metodo di Newton adatto allo scopo come

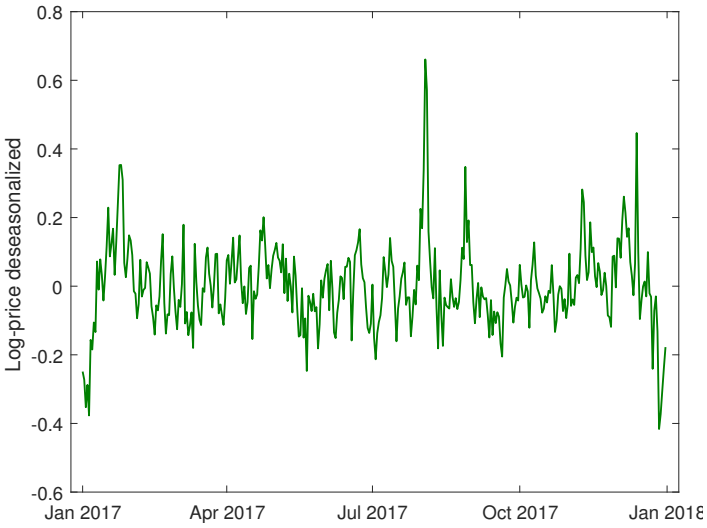

Figura 4: Log-price dei prezzi del power spot Italia per l'anno 2017 destagionalizzati.

quello mostrato nel Listato 1

Algorithm 1 Black Implied Volatility Estimation

1: Sia σ_{old} un valore iniziale per la volatilità e $C^{mkt}(t, F_t)$ il valore di mercato in corrispondenza della quotazione ad oggi del forward F_t
2: Sia $\nu = \frac{\partial C(t,F_T)}{\partial \sigma}$.
3: Fissata una tolleranza *toll* calcolare $e = C(t, F_T) - C^{mkt}(t, F_t)$
4: **while** $e \geq toll$ **do**
5: Calcola $\sigma_{new} = \sigma_{old} - \frac{C(t,F_T) - C^{mkt}(t,F_t)}{\nu}$
6: Aggiorna $\sigma_{old} = \sigma_{new}$
7: **end while**
8: **return** σ_{new}

Calibriamo ora la parte di breve termine ξ_t. Per stimare il set di parametri $\theta = (k, \sigma_\xi, \mu_J, \sigma_J, \lambda)$ ci serviamo della tecnica chiamata *Maximum Likelihood Estimator* (MLE). Questa tecnica consiste, data una certo segnale storico, di stimare quale sia il set di parametri che massimizzano la probabilità di aver osservato quella serie storica. Siccome le quotazioni delle spot sono giornaliere poniamo $\Delta t = 1/365$. Poniamo inoltre $\phi = 1 - \Delta t$. E' possibile dimostrare che la funzione di densità di X_t dato X_{t-1} è dato da:

$$f(X_t|X_{t-1}) = (\lambda \Delta t) N_1(X_t|X_{t-1}) + (1 - \lambda \Delta t) N_2(X_t|X_{t-1})$$

dove

$$N_1\left(X_t|X_{t-1}\right) = \left(2\pi\left(\sigma^2+\sigma_J^2\right)\right)^{-1/2}\exp\left(-\frac{\left(X_t-\phi X_{t-1}-\mu_J\right)^2}{2\left(\sigma^2+\sigma_J^2\right)}\right)$$

$$N_2\left(X_t|X_{t-1}\right) = \left(2\pi\left(\sigma^2\right)\right)^{-1/2}\exp\left(-\frac{\left(X_t-\phi X_{t-1}\right)^2}{2\sigma^2}\right)$$

Il parametro θ può essere stimato minimizzando la seguente funzione obiettivo

$$\min_{\Theta} -\sum_{t=1}^{T}\log\left(f\left(X_t|X_{t-1}\right)\right) \tag{5}$$

soggetta ai seguenti vincoli:

$$\phi < 1, \quad \sigma^2 > 0, \quad \sigma_J > 0, \quad 0 \leq \lambda\Delta t \leq 1.$$

All'atto pratico, è sufficiente sostituire alla variabile aleatoria generica X_t la serie delle sue realizzazioni x_t e minimizzare rispetto a θ la quantità mostrata in (5). Questa quantità non può essere minimizzata analiticamente ma è necessario un algoritmo di ottimizzazione. Nelle nostre analisi, come suggerito in [8], utilizziamo la funzione *MATLAB*, `mle`.
Terminata la procedura di calibrazione siamo in possesso del seguente set di parametri $\theta = (\sigma_\xi, \sigma_\chi, k, \mu_J, \sigma_J, \lambda)$ e possiamo quindi passare alla scrittura di un algoritmo che ci permetta di simulare il processo mostrato in (1).

Prima di proseguire è bene fare alcune considerazioni sul processo di calibrazione.
Da un punto di vista teorico tutti gli algoritmi presentati convergono al set di parametri "vero" ovvero si dice che sono stimatori non distorti dei parametri. Tuttavia la realtà spesso si discosta notevolmente dalla teoria. Sono spesso presenti dati mancanti, outliers che possono destabilizzare la procedura di calibrazione. Ecco perchè è necessario sempre dare uno sguardo attento ai dati su cui viene effettuata la calibrazione per valutare se è il caso di eseguire un loro preprocessamento. Talvolta può capitare che i dati su cui effettuare la calibrazione siano pochi. In questo caso il risultato potrebbe essere poco affidabile. Altra cosa da tenere in considerazione è, come già accennato, che gli algoritmi di minimizzazione spesso tendono a stagnare nei minimi locali. Ecco perchè è sempre meglio eseguire ottimizzazioni chiamate multi-start in cui il punto iniziale di partenza dell'algoritmo viene variata e l'ottimizzazione viene lanciata quindi a partire da più punti iniziali: alla fine viene scelto il valore di parametri a cui corrisponde il valore minimo della funzione obiettivo. Una possibile alternativa è l'utilizzo di metodi di minimizzazione non deterministici che non presentano il problema della stagnazione in minimi locali ma che possono essere estramamente onerosi.
Da ultimo un'occhio esperto di mercato a volte conta più di tutte le procedure di matematiche possibili per la stima di un parametro.

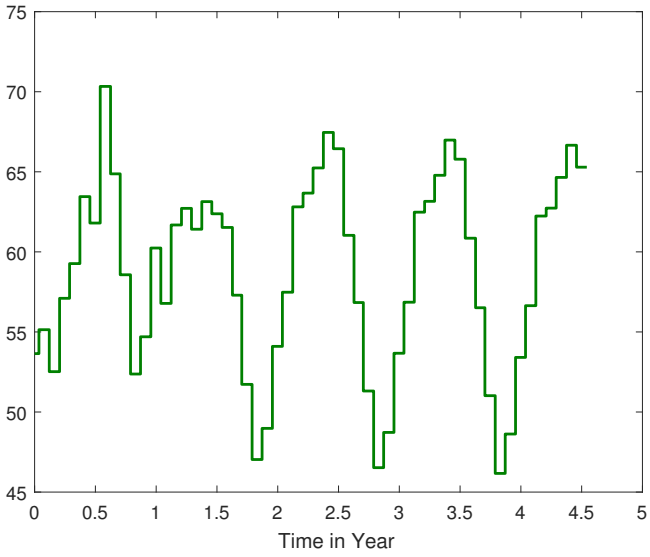

Figura 5: Curva forward dei prodotti mensili.

4 Smoothing della curva Forward

Per lo smoothing della curva forward prendiamo spunto da quanto presentato in [1] e in [7] utilizzando un approccio meno raffinato ma di semplice implementazione basato sull'utilizzo di spline di secondo grado. L'idea centrale per ottenere simulazioni coerenti con il mercato è assumere la curva Forward come migliore indicativa di quello che sarà il prezzo spot di mercato e far si che le simulazioni convergano ad essa. L'algoritmo seguente può essere applicato per interpolare qualsiasi curva forward, ma noi lo applicheremo ai prodotti power forward Italia.

Ogni giorno, a mercato, si osservano i prodotti forward cone diversa granularità. Supponiamo di avere una serie di n prodotti forward v_i $i = 1, \ldots, n$ non sovrapposti, ognuno con una certo time to maturity T_i $i = 1, \ldots, n$. Supponiamo, ad esempio, di avere quotazioni mensili. La situazione è mostrata in Figura 5.

L'obiettivo è rendere questa curva più liscia rispettando i vincoli sulle aree. Mese per mese, l'area sottesa alla curva che otterremo dovrà essere pari all'area sottesa che della curva forward flat originale. Qualunque software numerico consente di ottenere un'inteprolante polinomiale di una funzione ma il vincolo sulle aree complica leggermente le cose.

Consideriamo un'interpolante di tipo quadratico su ogni intervallo $[T_i, T_{i+1}]$ indicata con $f_i(t) : [T_i, T_{i+1}] \to \mathbb{R}$. Sia $f(t)$ l'interpolante risultante finale che verifica $f(t) = f_i(t) \quad t \in [T_i, T_{i+1}]$. Poniamo

$$f_i(t) = g_i\left(\frac{t - T_i}{T_{i+1} - T_i}\right)$$

con

$$g_i(s) = a_i s^2 + b_i s + c_i \quad s \in [0, 1].$$

ed osserviamo che

$$
\begin{aligned}
&g_i'(s) = 2a_i s + b_i \quad s \in [0,1] \\
&f_i(T_i) = g_i(0) = c_i \\
&f_i(T_{i+1}) = g_i(1) = a_i + b_i + c_i \\
&f_i'(T_i) = \frac{1}{T_{i+1} - T_i} g_i'(0) = \frac{b_i}{T_{i+1} - T_i} \\
&f_i'(T_i) = \frac{1}{T_{i+1} - T_i} g_i'(1) = \frac{2a_i + b_i}{T_{i+1} - T_i}
\end{aligned}
$$

e poniamo $\Delta T_i = T_i - T_{i-1}$

Ora imponiamo alcuni vincoli sulla funzione $f(t)$ di modo che rispetti alcune proprietà desiderate. Cerchiamo in particolare una funzione continua con derivata continua nei punti T_i. Inoltre vogliamo richiedere il vincolo sul rispetto delle aree sottese.
Poniamoci nel punto T_i. I vincoli di continuità nei punti T_i sono:

$$f_i(T_i) = f_{i+1}(T_i)$$

ovvero

$$a_i + b_i + c_i = c_{i+1}, \tag{6}$$

mentre quelli sulla continuità della derivata prima sono:

$$f_i'(T_i) = f_{i+1}'(T_i)$$

ovvero

$$\frac{2a_{i+1} + b_i}{\Delta T_i} = \frac{b_{i+1}}{\Delta T_{i+1}}, \tag{7}$$

Consideriamo, da ultimo, il vincolo intergrale su ciascun intervallo $[T_i, T_{i+1}]$

$$v_i = \int_{T_i}^{T_{i+1}} \frac{1}{T_{i+1} - T_i} f_i(t)\, t = \int_0^1 g_i(s)\, ds = \frac{a_i}{3} + \frac{b_i}{2} + c_i \tag{8}$$

dove si è usato il cambio di variabili $s = \frac{t - T_i}{T_{i+1} - T_i}$. In definitiva abbiamo:

$$
\begin{aligned}
a_i + b_i + c_i &= c_{i+1} \quad (n-1) \\
\frac{2a_{i+1} + b_i}{\Delta T_i} &= \frac{b_{i+1}}{\Delta T_{i+1}}, \quad (n-1) \\
v_i &= \frac{a_i}{3} + \frac{b_i}{2} + c_i \quad (n)
\end{aligned} \tag{9}
$$

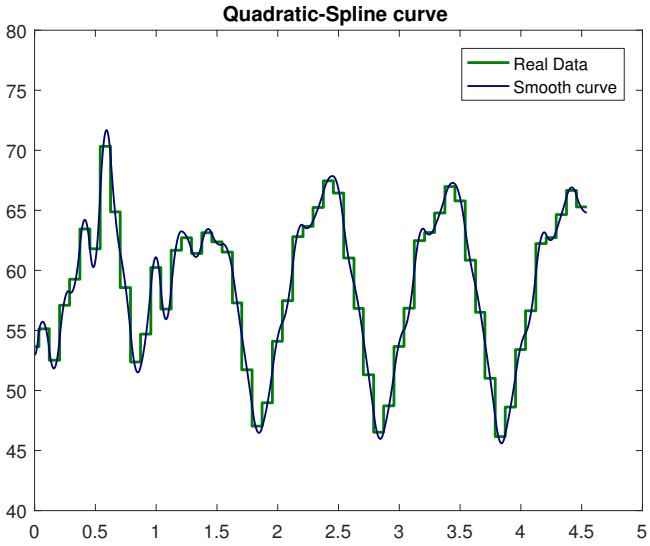

Figura 6: Curva forward interpolata con splines di secondo grado.

dove n indica il numero di equazioni. Abbiamo quindi $3n$ parametri da stimare e $3n-2$ equazioni e quindi dovremo aggiungere delle equazioni al cotorno. Osservando che:

$$c_i = v_i - \frac{a_i}{3} - \frac{b_i}{2} \tag{10}$$

$$a_i = \frac{1}{2}\left(\frac{\Delta T_i}{\Delta T_{i+1}} b_{i+1} - b_i\right) \tag{11}$$

ed inserendole in 9 otteniamo, dopo una breve rielaborazione dei termini, $n-2$ equazioni del tipo:

$$b_i + 2\left(\frac{\Delta T_i}{\Delta T_{i+1}} + 1\right) b_{i+1} + \frac{\Delta T_{i+1}}{\Delta T_{i+2}} b_{i+2} = 6\left(v_{i+1} - v_i\right) \tag{12}$$

Per avere un sistema di n equazioni in n incognite devo aggiungere due condizioni al contorno. Scegliamo due condizioni sulle derivate e quindi imponiamo che:

$$b_i = d_1 \Delta T_1$$
$$b_{n+1} = d_2 \Delta T_{n+1}$$

con d_1 e d_2 sono i valori dati per le derivate in T_1 e T_{n+1}. Risolvendo il sistema lineare in (12) si ottengono i b_i da cui è possibile ricavare gli a_i e, quindi, i c_i. Il sistema in (12) può essere risolto in $MATLAB$ tramite l'operatore \. Il risultato di questa procedura è mostrato in Figura 6.

Si potrebbe essere spinti a pensare che aumentando il grado delle spline si ottenga una migliore interpolazione. Come si osserva in Figura 7, in questo caso, passando ad un spline di terzo grado si ottengono delle oscillazioni molto più marcate nell'interpolante. Ecco perchè si è preferito utilizzare spline di secondo grado.

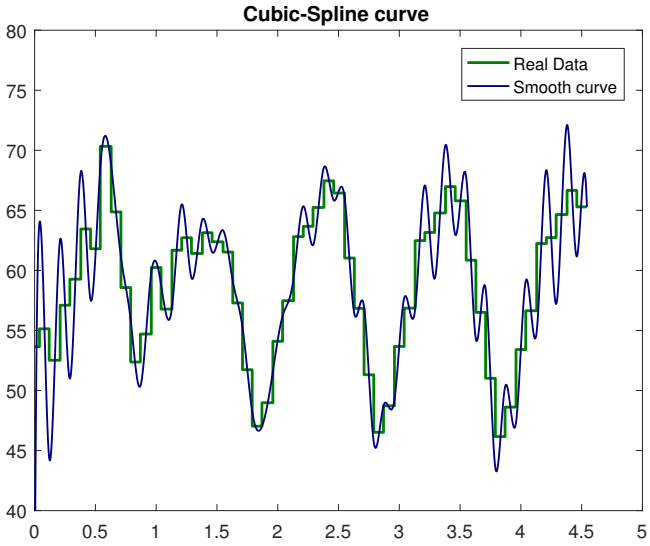

Figura 7: Curva forward interpolata con splines di terzo grado.

Osserviamo che, in realtà, non è veritiero supporre che siano quotati i singoli prodotti mensili con una maturity di quattro anni. Quello che si osserva oggi è che, su EEX, sono quotati solo i sei mesi successi alla data odierna e poi, via via, i prodotti si fanno meno granulari. Generalmente per con un'orizzone di quattro anni è quotato solo il Calendar. Per passare alla forma mensile ci sono due strade. O si cerca di dare una forma al prodotto calendar di modo che la media sull'anno non cambi oppure, in maniera più raffinata, è possibile contemplare una versione più elaborata dell'algoritmo utilizzato per lo smoothing della curva forward incorporando nella curva forward anche una stagionalità derivante dallo storico spot. Questo algoritmo sarà eventualmente oggetto di una futura ricerca. Ai nostri fini aziendali, per il momento, possiamo accontentarci di un'approccio del primo tipo dato che risulta essere più intuitivo.

4.1 Aggiunta della microstagionalità

Osserviamo che non è ancora stata aggiunta alla curva forward la microstagionalità. Per fare questo procediamo come segue. Nella sezione 2 tramite regressione abbiamo ricavato i parametri presenti in (2) che poi abbiamo usato per destagionalizzare la serie storica di prezzi spot giornalieri. I coefficienti w_i sono i coefficienti della microstagionalità e sono proporzionali a quanto "pesa" ogni giorno della settimana in termini di prezzo. I coeffienti w_i per l'anno 2017 sono riportati in Tabella 1.

In maniera molto semplice definiamo un indice di forma $\boldsymbol{l} = [l_1, l_2, \ldots, l_7,]$ in cui

$$l_i = \frac{w_i}{\sum_{j=1}^{7} w_j}.$$

A questo punto posso applicare l'indice così definito alla curva forward smothed ed ottenere una curva forward che contiene una microstagionalità ponendo

Parametro	Valore Stimato
w_1	3.9031
w_2	4.0851
w_3	4.1125
w_4	4.1376
w_5	4.1175
w_6	4.1061
w_7	3.9776

Tabella 1: Valori dei parametri della microstagionalità calibrati sull'anno 2017. w_1 è domenica, w_2 lunedì etc. Notare come i parametri relativi alla domenica ed al sabato siano più bassi dei parametri relativi ai giorni feriali considerati.

$$\boldsymbol{L} = \underbrace{[\boldsymbol{l}, \ldots, \boldsymbol{l}]}_{N_w}$$

dove N_w è il numero di settimane. Otteniamo la curve forward $F(t)$ con micro stagionalità ponendo:

$$F(t) = f(t) \cdot \boldsymbol{L}.$$

dove $\cdot$ denota il prodotto scalare.

5 Risk-Neutral Measure and Price of Risk

Le simulazioni del prezzi spot che simularemo dovranno essere consistenti con la curva forward. Se eseguiamo la calibrazione dei parametri e simuliamo la componente stocastica X_t e modelliziamo il sottostante spot come $S_t = \exp\{F(t) + X_t\}$ quello che otteniamo è che la media delle simulazioni spot non converge alla curva forward $F(t)$ (Figura 8). Questo perchè non è stato calibrato il market price of risk, che indicheremo con γ_t.

Per calcolare il premio per il rischio procediamo prendendo spunto da [9] e [8].
Dati i prezzi forward, indicati con $F(t,T)$ è possibile estrarre il market-price of risk, variabile nel tempo, γ_t tramite le seguenti relazioni.

$$\begin{aligned}
F(t,T) &= \mathbb{E}^{\mathbb{Q}}[S_T|\mathcal{F}_t] \\
&= \mathbb{E}^{\mathbb{Q}}\left[e^{F(t)+\xi(T)+\chi(T)}|\mathcal{F}_t\right] \\
&= \mathbb{E}^{\mathbb{Q}}\left[e^{F(t)+\xi(T)+\int_t^T \gamma_s ds+\sigma_\chi W_T^\chi}|\mathcal{F}_t\right] \\
&= \mathbb{E}^{\mathbb{Q}}\left[e^{\int_t^T \gamma_s ds} e^{F(t)+\xi(T)+\sigma_\chi W_T^\chi}|\mathcal{F}_t\right] \\
&= e^{\int_t^T \gamma_s ds}\mathbb{E}^{\mathbb{P}}\left[e^{F(t)+\xi(T)+\sigma_\chi W_T^\chi}|\mathcal{F}_t\right]
\end{aligned}$$

Discretizzando questa equazione otteniamo:

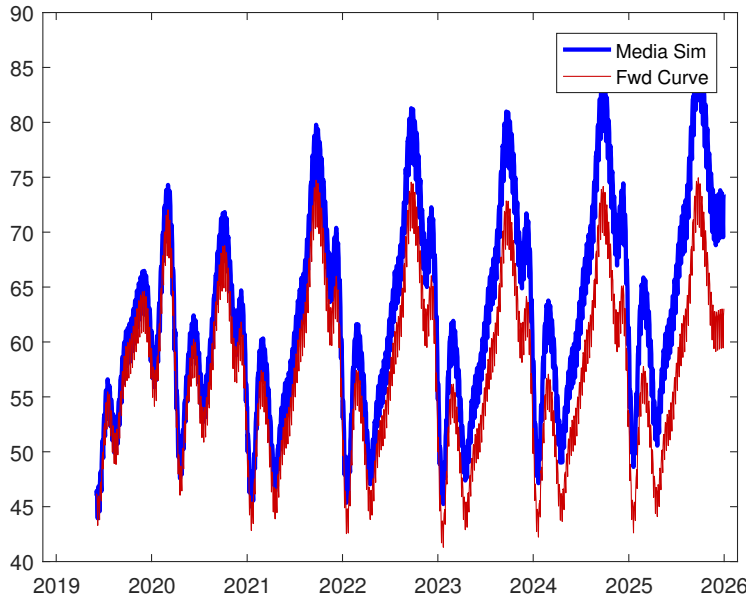

Figura 8: Non convergenza alla curva Forward delle simulazioni in assenza di aggiunta del market price of risk λ.

$$\gamma_{T-1} = \frac{\log\left(\frac{F(t,T)}{\mathbb{E}^{\mathbb{P}}\left[e^{F(t)+\xi(T)+\sigma_X W_T^X}|\mathcal{F}_t\right]}\right)}{\Delta t} - \sum_{s=t}^{T-2}\gamma_s \tag{13}$$

e quindi possiamo ottenere la funzione deterministica γ_t applicando in maniera iterativa l'Equazione (13). Grazie al premio per il rischio γ_t il prezzo spot può essere trasformato nella misura neutrale al rischio e ottenere la consistenza delle simulazioni con la curva forward. Osserviamo, inoltre, che l'Equazione (13) si traduce nel risolvere un sistema lineare. Infatti

$$\begin{aligned} F(t,T) &= e^{\int_t^T \gamma_s ds}\mathbb{E}\left[S_T|\mathcal{F}_t\right] \\ \int_t^T \gamma_s ds &= \log\left(F(t,T)/\mathbb{E}\left[S_T|\mathcal{F}_t\right]\right) \end{aligned}$$

e indicando con $t = t_0, t_1, \cdots, t_n = T$ abbiamo che:

$$\begin{aligned} &\gamma_1\Delta t = \log\left(F(t,t_1)/\mathbb{E}\left[S_{t_1}|\mathcal{F}_t\right]\right) \\ &\gamma_1\Delta t + \gamma_2\Delta t = \log\left(F(t,t_2)/\mathbb{E}\left[S_{t_2}|\mathcal{F}_t\right]\right) \\ &\gamma_1\Delta t + \gamma_2\Delta t + \gamma_3\Delta t = \log\left(F(t,t_3)/\mathbb{E}\left[S_{t_3}|\mathcal{F}_t\right]\right) \\ &\cdots \\ &\gamma_1\Delta t + \gamma_2\Delta t + \cdots + \gamma_n\Delta t = \log\left(F(t,T)/\mathbb{E}\left[S_T|\mathcal{F}_t\right]\right) \end{aligned} \tag{14}$$

e quindi indicato con $\boldsymbol{b} = [\log(F(t,t_1)/\mathbb{E}[S_{t_1}|\mathcal{F}_t]), \ldots, \log(F(t,t_n)/\mathbb{E}[S_{t_n}|\mathcal{F}_t])]/\Delta t$, con

$$A = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 1 & 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \ldots & 1 \end{bmatrix}$$

e con

$$\boldsymbol{\gamma} = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_n. \end{bmatrix}$$

Abbiamo allora da risolvere il seguente sistema lineare triangolare inferiore:

$$A\boldsymbol{\gamma} = \boldsymbol{b}$$

cosa che può essere fatta in maniera molto semplice con un qualsiasi software numerico, tra cui *MATLAB*, tramite l'istruzione `gamma= A\b`
Una volta calcolato γ_t possiamo ottenere simulazioni risk-neutral utilizzando al seguente dinamica per la componente long-term:

$$d\xi_t = \gamma_t dt + \sigma_\xi dW_\xi.$$

Aggiungendo questa componente di drift γ_t alla dinamica di long-term otteniamo la convergenza delle simualzioni in media alla curva forward di mercato, come si nota in Figura 9.

6 Algoritmi Numerici di Simulazione

In questa sezione presentiamo gli algoritmi necessari per la simulazione del processo:

$$\begin{aligned} d\chi_t &= -k\chi_t dt + \sigma_\chi dW_\chi + dJ_t \\ d\xi_t &= \gamma_t dt + \sigma_\xi dW_\xi \end{aligned} \tag{15}$$

Discreizziamo l'equazione (15) ottenendo

$$\begin{aligned} \Delta\chi_{t+1} &= -k\chi_t \Delta t + \sigma_\chi \Delta W_\chi + \Delta J_t \\ \Delta\xi_t &= \gamma_t \Delta t + \sigma_\xi \Delta W_\xi \end{aligned}$$

dove $\Delta W_k \sim \mathcal{N}(0,t)$ mentre $\log \Delta J_t \sim \mathcal{N}(\mu_J, \sigma_J^2)$ se c è presente un salto nell'intervallo di tempo Δt. Qualunque software numerico contiene le funzionalità per generare numeri casuali distribuiti normalmente e quindi la simulazione delle variabili W_k è immediata. Ci serve un algoritmo per simulare gli istanti di salto nell'intervallo $[0,T]$. Utilizziamo l'algoritmo presente nel Listato 2, proposto in [10].

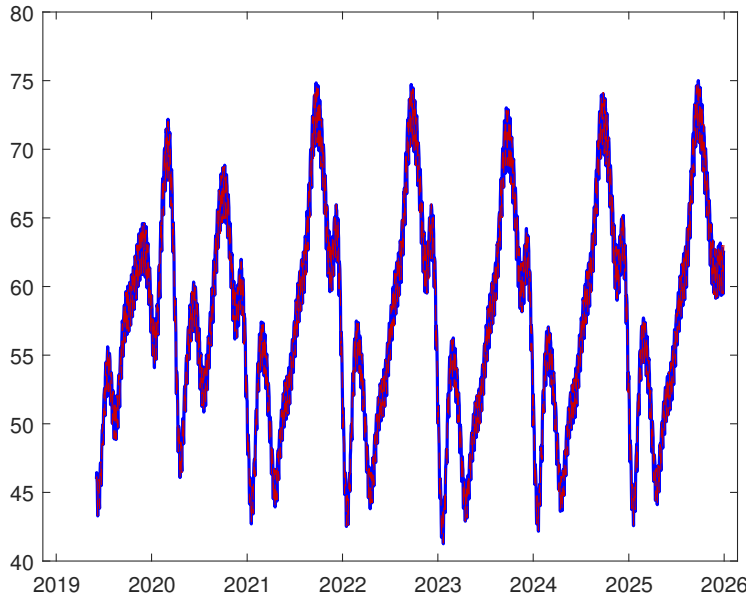

Figura 9: Corvergenza della media delle simulazioni alla curva forward di mercato con l'aggiunta del market price of risk γ_t.

Algorithm 2 Jump Times Simulation

1: Simula una variabile aleatoria N distribuita come una Poisson di parametro λT. N è il numero di salti in $[0, T]$
2: Simula N variabili aleatorie indipendenti U_i distribuite uniformemente su $[0, T]$
3: **return** U_i come istanti di salto del processo.

Algorithm 3 Dynamics Simulation

1: Data una griglia temporale discreta $0 = t_1, \ldots, t_m = T$
2: Simula m variabili aleatorie indipendenti $\nu_i \sim \mathcal{N}(0, 1)$ e m variabili aleatorie indipendenti $\eta_i \sim \mathcal{N}(0, 1)$.
3: Porre $\xi_1 = 0$ e $\chi_1 = 0$
4: Simulare $\chi_{i+1} = \chi_i - k\chi_i \Delta t + \sigma_\chi \sqrt{\Delta t}\nu_i + \Delta J_{t_i}$
5: Se l'istante di tempo t_i è un'istante di salto $\Delta J_{t_i} \sim \mathcal{N}\left(\mu_J, \sigma_J^2\right)$ altrimenti $\Delta J_{t_i} = 0$.
6: Simulare $\xi_{t+1} = \xi_t \gamma_t \Delta t + \sigma_\xi \sqrt{\Delta t}\eta_i$

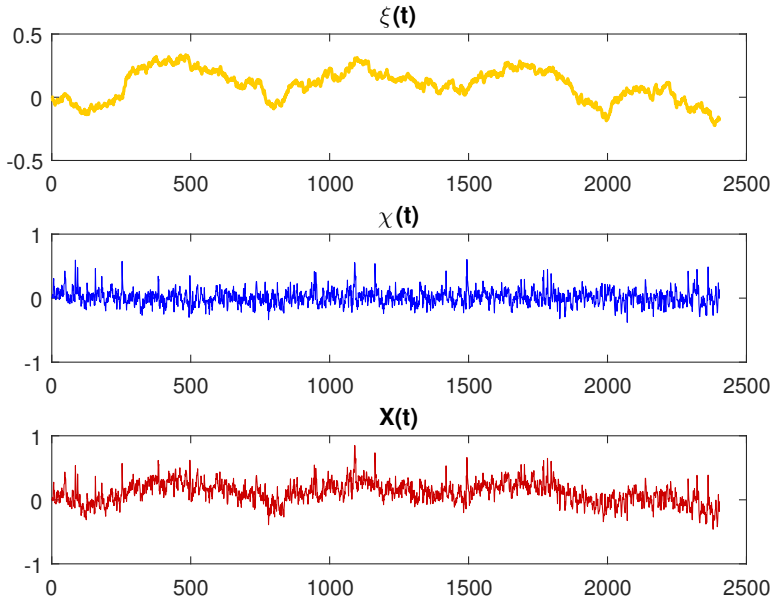

Figura 10: Possibili traiettorie dei processi $\chi(t)$, $\xi(t)$ e $X(t)$

Una volta simulati gli istanti di salto T_i è possibile simulare la dinamica di (15) tramite il seguente l'Algortimo 3.
Il processo $X(t) = \chi(t) + \xi(t)$ può essere quindi approssimato come $X_i = \xi_i + \chi_i$.
Una possibile realizzazione è mostrata in Figura 10.

7 Risutlati Numerici

In questa sezione presentiamo alcuni risultati numerici. In una prima parte verifichiamo la procedura di calibrazione e le proprietà delle simulazioni, mentre nalla seconda parte della sezione mostriamo come utilizzare il modello per il pricing di opzioni sullo Spot del Power.

7.1 Calibrazione e Proprietà del modello

Consideriamo come dati power spot Italia i prezzi medi giornalieri dal 1 Gennaio 2017 al 31 Dicembre 2017. Come data di Settlement del mercato Forward prendiamo le quotazioni di EEX al giorno 17 Giugno 2019.

7.1.1 Calibrazione della Componente Short-Term $\chi(t)$

Seguendo quanto proposto nella sezione 3 otteniamo per i parametri $\theta = (k, \sigma_\chi, \mu_J, \sigma_J, \lambda)$ i valori riportati in Tabella 2. Questi parametri, come accennato, sono stati ricavati a partire della serie storica dei prezzi medi giornalieri spot Italia nell'orizzone temporale 1 Gennaio 2017 - 31 Dicembre 2017.

Parametro	Valore Stimato
k	129.1107
σ_χ	1.4670
μ_J	0.0620
σ_J	0.1739
λ	22.6792

Tabella 2: Parameti calibrati della componente $\chi(t)$ tramite Maximul Likelihood Estimator.

7.1.2 Calibrazione della Componente Long-Term $\xi(t)$

Il parametro della componete long-term $\xi(t)$, ovvero la volatilità di lungo periodo indicata con σ_ξ viene calibrato a partire dalle volatilità implicite del prodotto corrispondente al prodotto spot che si vuole simulare.
Quindi, volendo simulare il power spot del mese di Settembre 2019, un approccio possibile sarebbe quello di calibrare il paramtro σ_ξ come la volatilità implicita dalle quotazioni delle Call/Put a mercato del prodotto FDBM 2019.09.

Per ottenere la volatilità implicita di un'opzione forward nel modello di Black76 si può usare la routine di *MATLAB* `blkimplvol` o implementare l'Algoritmo 1. Le volatilità implicite per opzioni Call e Put su diversi prodotti Forward Power Italia sono mostrati in Figura 17. Per il calcolo delle volatilità implicite si è fatto uso della funzione *MATLAB* `blkimplvol`.

7.2 Alcune proprietà del Modello

In questa sezione presentiamo alcune proprietà del modello.
Iniziamo con il confrontare le traiettorie generate dal modello con quelle reali. In Figura 18 osserviamo che le traiettorie generate dal modello sono compatibili con quelle osservate nella realtà.
Analizziamo ora la distribuzione dei log-rendimenti. In Figura 19 osserviamo che anche le distribuzioni dei log-rendimenti reali e simulati sono abbastanza simili, a conferma della bontà del modello.
Osserviamo in Figura 20 che le traiettorie generate dal modello non sono in fase. E' possibile che una traiettoria ammetta un prezzo più basso rispetto ad un'altra in una fase dell'anno e poi si verifichi un'inversione di prezzo. Le simulazioni generate non sono quindi delle semplici traslazioni o dilatazioni della curva forward originale: il modello permette di generare traiettorie di prezzo più complesse e variegate.
Da ultimo, in Figura 21, controlliamo la convergenza delle simulazioni alla curva forward di mercato a cui aggiungiamo il 95 ed il 5 percentile. Osserviamo che la convergenza delle simulazioni alla curva forward è garantito e quindi il modello è consistente con la curva forward di mercato. Ricordiamo che l'apertura del fascio di simulazioni sul lungo periodo è determinato dal parametro σ_ξ. In questo caso si è scelto di utilizzare $\sigma_\xi = 0.30$ il valore della volatilità implicita della Call sul prodotto FDBM 2019.09 in corrispondenza dello strike $K = 57$. Tale scelta non ha un motivo specifico e, a seconda del prodotto che si deve prezzare, verrà usata la $\sigma_\xi = 0.30$ più appropriata.
Notiamo che tutti i parametri del modello hanno un chiaro significato: modi-

ficando i parametri calibrati a mercato è possibile "manipolare" le simulazioni eseguendo tutti gli stress-test necessari del modello.

7.3 Pricing

In questa sezione analizziamo come si comporta il modello di pricing su una serie di opzioni Call Asiatiche con diverse strike con sottostante S_t il prezzo spot PUN del mese di settembre 2019 ognuna con payoff:

$$\Phi = (A - K)^+$$

dove

$$A = \frac{1}{N} \sum_{i=1}^{N} S_i$$

con N numero di giorni del mese.
Confrontiamo poi il prezzo di tale opzione con i prezzi di analoghe opzioni Europee sui prodotti forward corrispondenti, ovvero FDBM 2019.09 per diversi valori dello strike, con $K = [52, 52.5, \cdots, 68, 68.5]$. I risultati sono riportati in Figura 22.

Osserviamo che i premi delle Call Asiatiche sulla media dello Spot e i premi delle rispettive opzioni Call Europee scritte sui forward dello stesso periodo sono abbastanza simili. Questa differenza è spiegabile dal fatto che abbiamo garantito la convergenza delle simulazioni dello spot alla forward corrispondente: facendo così stiamo assumento che il forward sia la migliore previsione dello spot e quindi avere un'opzione Call Europea sul forward (che è assunto essere pari alla media dello spot del mese considerato) o una Call Asiatica sullo Spot è, sostanzialmente, la stessa cosa. Questa cosa non deve sorprendere: l'assunzione che lo spot sia più volatile del forward, e che quindi un contratto sullo spot sia più rischioso che uno sul forward, è vera solo se si ragiona a livello di quotazioni giornaliere. Nel caso in cui si ragioni con medie mese e nel caso in cui si richieda la consistenza con la curva di mercato non ha senso aspettarsi grandi scostamenti la l'opzione Europea forward e l'opzione Asiatica sulla media dato che il sottostante è il medesimo. Diverso è se si va a considerare un periodo all'interno del mese di settembre. Prezziamo, ad esempio, una Call Asiatica scritta sui giorni 25-26 settembre 2017 (martedì-mercoledì) e confroniamo, in Figura 23, il premio ottenuto con quello della rispettiva Call Europea scritta sul forward del mese corrispondente.

Il prezzo più alto rispetto alla corrispondente opzione sul forward è dovuto al fatto che il prezzo medio del 25-26 settembre è più alto del prezzo medio del mese di settembre (Figura 24)

Il modello consente di prezzare in maniera coerente anche sotto-periodi del mese. Consideriamo il pricing di tre Call Asiatiche con strike $K = 57$: una sulla prima metà del mese di settembre, una sulla seconda metà ed una sull'intero mese. I risultati sono riportati in Tabella 3.

Osserviamo che il prezzo della Call Asiatica scritta sull'intero mese non coincide con la media dei premi delle Call Asiatiche scritte sui sottoperiodi. Questo è ragionevole perchè il payoff non è lineare. Può capitare, infatti, che la Call Asiatica sul primo sottoperiodo venga esercitata, la corrispondente sul secondo periodo non venga esercita e la Call Asiatica su tutto il periodo venga esercitata

Periodo	Premio
$01/09/2019 - 15/09/2019$	2.9585
$16/09/2019 - 30/09/2019$	4.3697
$01/09/2019 - 30/09/2019$	3.4721

Tabella 3: Premi in $[EUR/MWh]$ di tre Call Asiatiche scritte sul tre periodi differenti del mese di settembre 2019.

come possiamo osservare in Figura 25. non coincidendo i payoff non è pensabile che i premi coincidano.

8 Conclusioni e sviluppi futuri

In questo documento abbiamo presentato numerosi argomenti. Abbiamo definito un modello a due fattori che permetta di modellare il prezzo spot dell'energia elettrica: esso è costituito da una componente di lungo periodo di tipo browniana ed una componente di breve di tipo mean-reverting. Abbiamo moi mostrato come calibrare il modello, assumendo correlazione nulla tra la componente di breve e lungo periodo: la componente di lungo periodo è stata calibrata usando le volatilità implicite dei prodotti quotati mentre la componente di breve è stata calibrata utilizzando la serie storica del prezzo power spot tramite MLE. Per poter garantire consistenza con il mercato abbiamo richiesto che la media delle simulazioni convegesse alla curva forward di mercato. Inizialmente abbiamo preso le quotazione forward ed abbiamo costruito una versione "smooth" della curva forward per mezzo di splines quadratiche, vi abbiamo aggiunto una microstagionalità e, infine, abbiamo stimato il valore del premio per il rischio, garantendo la consistenza delle simulazioni spot e della curva forward.
Abbiamo mostrato gli algoritmi numerici necessari per le simulazioni ed abbiamo analizzato alcune proprietà del modello. Da ultimo abbiamo utilizzato il modello per effettuare il pricing di contratti sul prezzo power spot.

Abbiamo costruito un modello che permette di eseguire simulazioni giornaliere: per poter eseguire simulazioni orarie un approccio immediato può essere il seguente: prendere le simulazioni giornaliere e montarci sopra una profilatura oraria la cui media coincida con la media del prezzo giornaliero simulato. La simulazione diretta dei prezzi giornalieri non è ancora stata testata e risulta essere, a prima vista, più complessa vista la dinamica più "estrema".

Il modello derivato può essere facilmente adattato al qualsiasi prodotto power e non che presenti gli stessi fatti stilizzati di mercato della commodity PUN. Inoltre, il modello può essere utilizzato anche per il calcolo del Profit at Risk (PaR) di certo asset.
Un'altro punto forza del modello, già sottolineato precedentemente, è la totale trasparenza dei parametri calibrati. Ogni parametro ha una sua chiara interpretazione e quindi il modello può essere stressato variando a piacere il valore dei parametri. Questo consente di effetturare analisi di tipo what-if in maniera semplice.

Il modello sviluppato, al momento, riesce a trattare solo il caso singola commodity. La trasposizione di un modello del genere in ambito multivariato è allo studio e non trova numerosi riscontri in letteratura anche se molto recentemente sono stati fatti i primi studi [5]. La complessità dell'estensione del modello ad un framework multi variato è dovuta alla complessità di modellizzare la dipendenza estremamente complessa di un sistema, quello dell'energy spot appunto. Ancora una volta, la parte più complessa non è la costruzione del modello ma la stima dei parametri. Se stimare una matrice di covarianza per variabili normali è una questione delicata, definire cosa sia la dipendenza degli istanti di salto è sicuramente una questione interessante e per nulla banale.

Un'ulteriore estensione di questo modello potrebbe contemplare il "basis-risk". L'introduzione della modellizzazione del basis-risk porterebbe ad un'aumendo del prezzo delle opzioni scritte sullo spot.

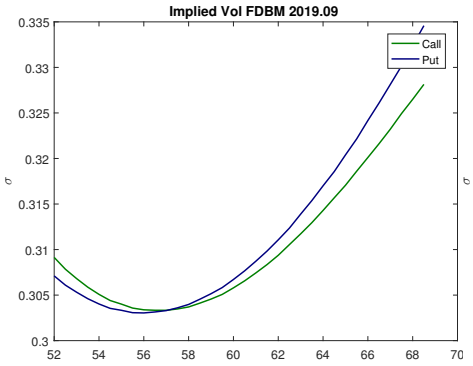

Figura 11: FDBM 2019.09

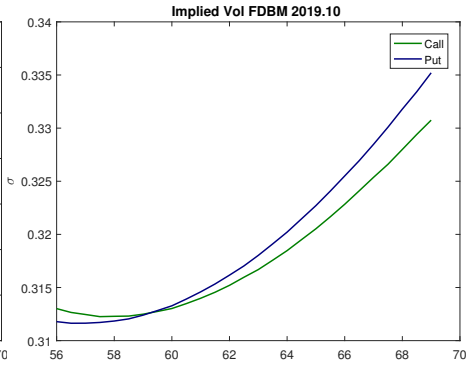

Figura 12: FDBM 2019.10

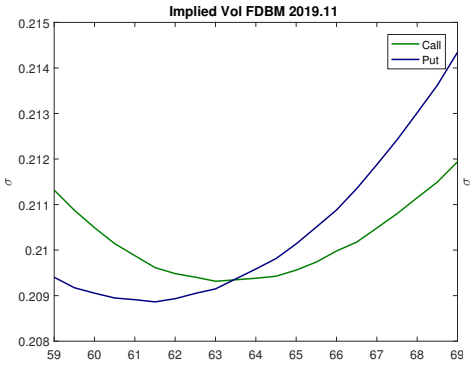

Figura 13: FDBM 2019.11

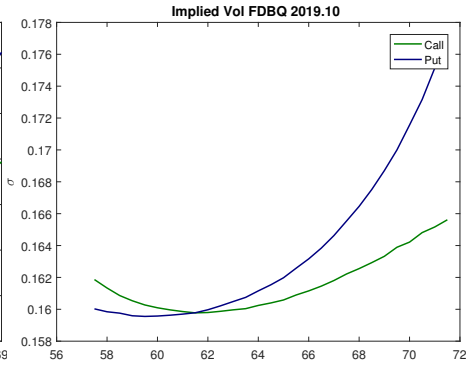

Figura 14: FDBW 2019.04

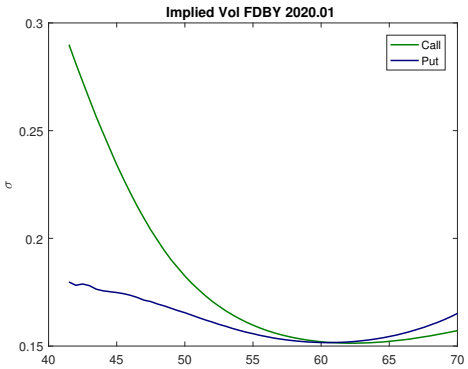

Figura 15: FDBY 2020

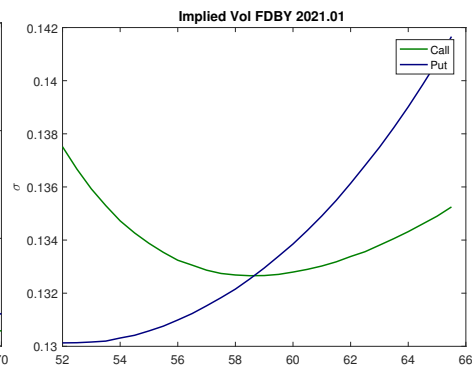

Figura 16: FDBY 2021

Figura 17: Smile di Volatilità per Call e Put per diversi prodotti Power Forward Italia alla Settlement Date 17 Giugno 2019.

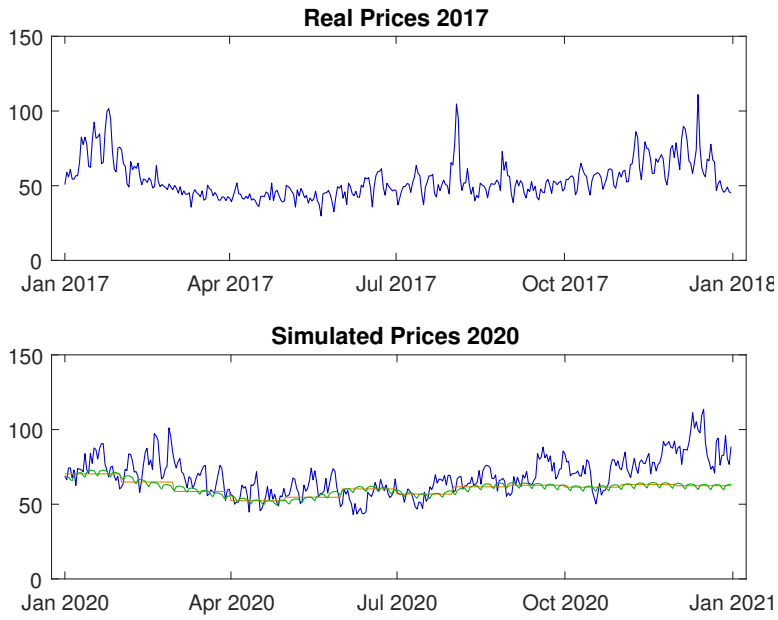

Figura 18: Traiettoria reale dei prezzi spot del 2017 confrontata con la traiettoria simulata dei prezzi.

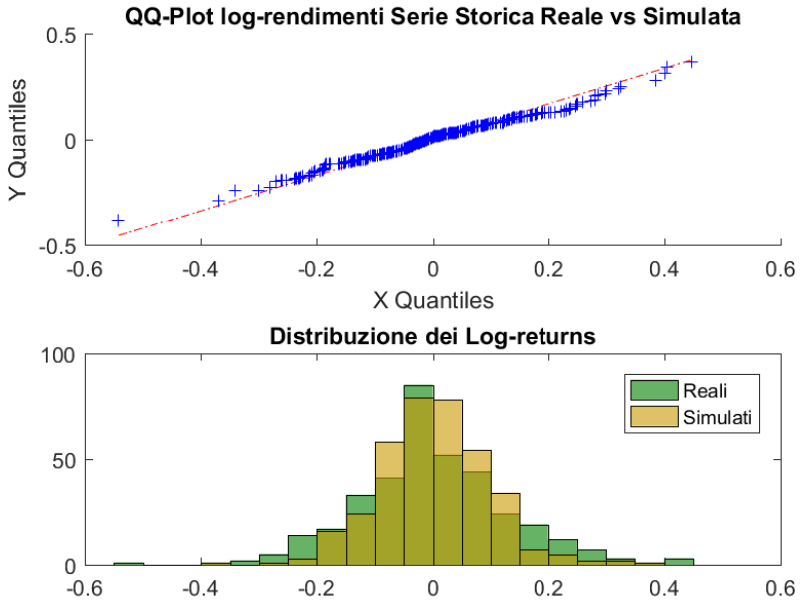

Figura 19: Distribuzione dei log-rendimenti dei prezzi. In alto il QQ-plot mentre in basso l'istogramma dei log rendimenti delle due serie (reale e simulata).

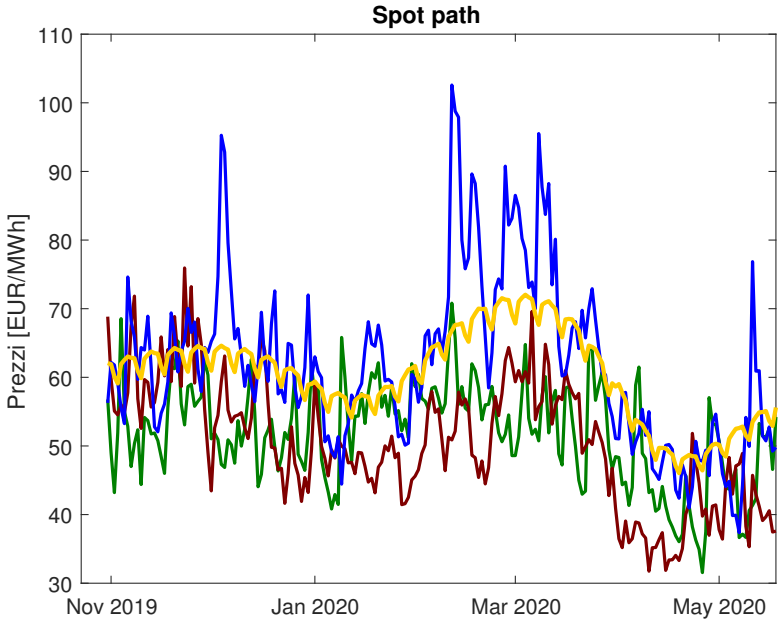

Figura 20: Alcune traiettorie possibili generate dal modello proposto. Si nota come la "fase" non sia la stessa per tutte le simulazioni.

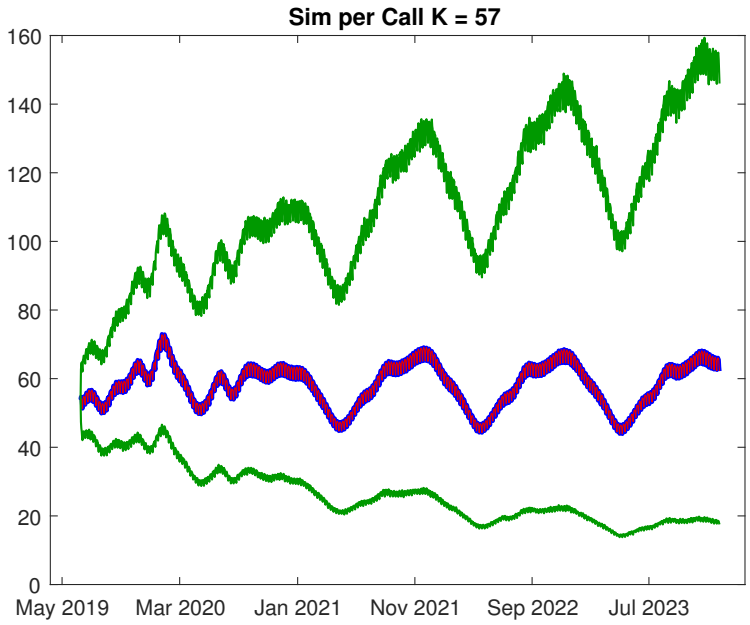

Figura 21: Convergenza delle simulazioni alla curva forward e andamento del 5 e del 95 percentile.

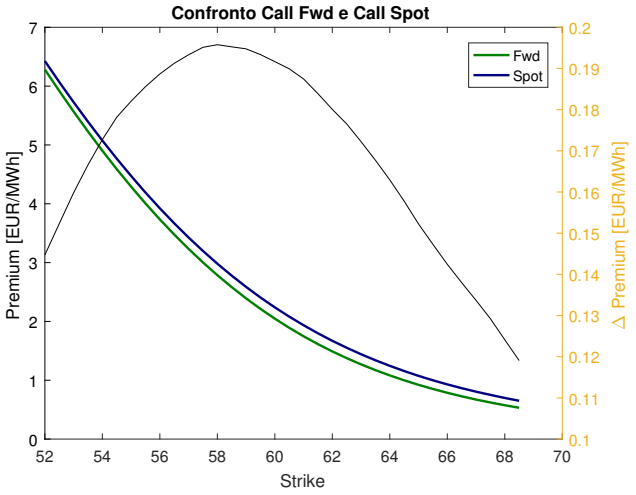

Figura 22: Premio di una Call Asiatica sullo Spot confrontata con il premio di una Call Europea sul sul forward del mese corrispondente FDBM 2019.09.

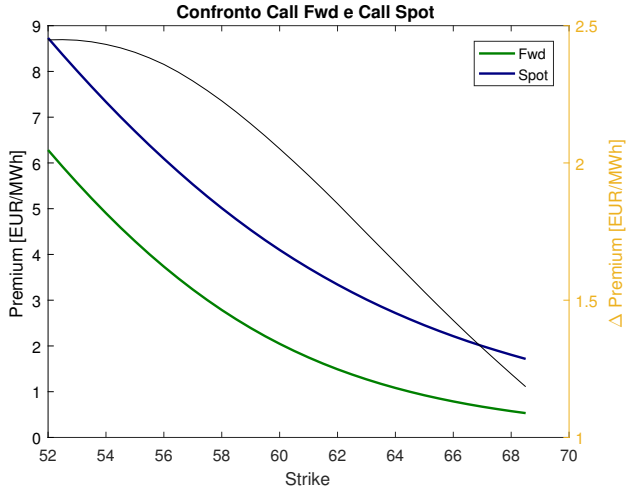

Figura 23: Premio di una Call Asiatica sullo Spot dei giorni 25-26 settembre 2019 confrontata con il premio di una Call Europea sul sul forward del mese corrispondente FDBM 2019.09.

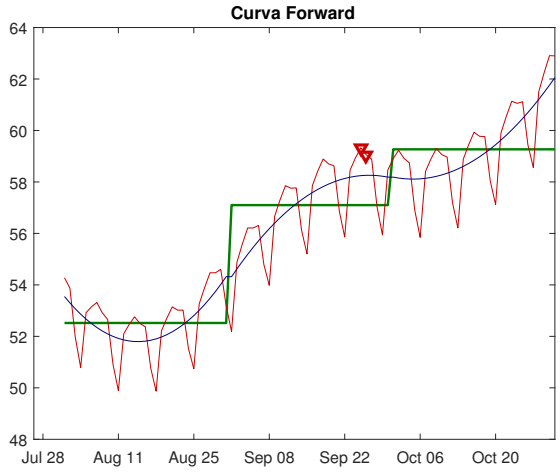

Figura 24: Curva Forward dei mesi di agosto, settembre, ottobre 2019. Evidenziati i giorni del 25-26 settembre 2019.

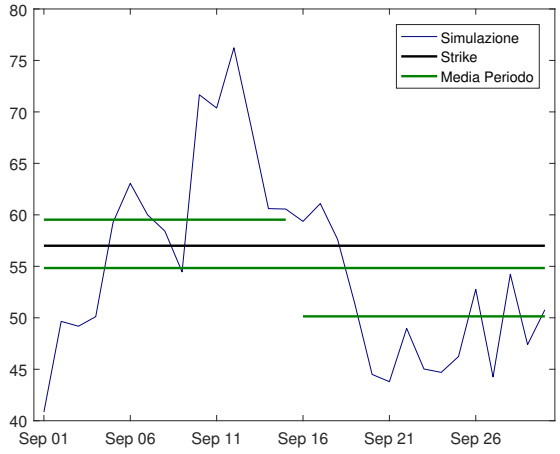

Figura 25: Esercizio delle Call Asiatiche. Laddove la media (linea verde) è superiore allo strike K (linea nera) la Call viene esercitata. Risulta che la Call sull'intero periodo e sul secondo periodo non viene esercitata, mentre viene esercitata la Call sul primo periodo.

9 Codici Matlab

In questa sezione alleghiamo, per completezza i codici *MATLAB* utilizzati. Tutti questi codici sono a livello prototipale e non ingegnerizzati a dovere. Si avverte l'utente di eseguire accurati controlli e verifiche prima di usare questi codici scopo industriale. L'autore si solleva da ogni tipo di responsabilità derivante dall'utilizzo da parte di terzi dei codici allegati.

9.1 Codice main per la costruzione delle Curva Forward Smooth

Il seguente codice consente la creazione di una curva forward smooth utulizzando splines di secondo grado come interpolante.

```
% Setta le variabili
clc
clear variables
close all

%% Abilita per test-sample
% Quadratic-SPLINE INTERPOLATION
% T = [0 1 2 3 4];
% dT = T(2:end)-T(1:end-1);
% F = [30 60 40 25];

% Abilita per real data
ds = datastore('CurvaFwdINPUT.xlsx');
ds.Range = 'D2:L57';
ds.Sheets = 'ForwardCurve';
dati = read(ds);
FWD = [dati.Forward,dati.T];

% T = [FWD(:,2)',FWD(end,2)+31/365];
T = [0,FWD(:,2)'];
dT = T(2:end)-T(1:end-1);
F = FWD(:,1)';

% Crea le Spines di Secondo Grado
n = length(F);

v = (F(2:end) - F(1:end-1))';
c = 2.*(dT(1:end-1)./dT(2:end)+1);
d = ones(n-1,1);
u = dT(1:end-1)./dT(2:end);

B = spdiags([d c' u'],-1:1,n-1,n-1);

b = B\(6.*v);

b = [0;b];

a = 0.5.*((dT(1:end-1)./dT(2:end))'.*b(2:end) - b(1:end-1));

a = [a; - 0.5*b(end)];

c = F' - a./3 - b./2;

%%
```

```
fp = [];
tp = [];
fwd_flat = [];


nDay = nan(n,1);
lunFwdDie = nan(n,1);

for j = 1:n
    Tini = T(j);
    Tfin = T(j+1);

    nDay(j) = round((Tfin-Tini)*365);
    t = linspace(Tini,Tfin,nDay(j));

    f = @(x) a(j).*((t-Tini)./(Tfin-Tini)).^2 +...
        b(j).*((t-Tini)./(Tfin-Tini)) + ...
        c(j);

    fwd_flat = [fwd_flat,F(j).*ones(1,nDay(j))];
    lunFwdDie(j) = numel(f(t));

    fp = [fp f(t)];
    tp = [tp t];
end

tp = tp';
tpday = (1:numel(fp))';
calendario_fwd = datetime(2019,06,17)+ days(tpday);

calExcel = m2xdate(calendario_fwd);
tp = tp';

%% Plot della forward
figure();
plot(calendario_fwd,fp);

idx = calendario_fwd ≥ datetime(2019,08,01) & ...
    calendario_fwd ≤ datetime(2019,09,30);

AVG = mean(fp(idx));
hold on
plot(calendario_fwd(idx),fp(idx),'--r');
plot(calendario_fwd(idx),fwd_flat(idx),'Color',[0 0.5 0]);

%%
if(false)
    haic = figure();
    plot(tp,fwd_flat,'r','LineWidth',1.5,'Color',[0 0.5 0]);
    hold on
    plot(tp,fp,'b','LineWidth',1,'Color',[0 0 0.5]);
    legend('Real Data','Smooth curve');
    title('Quadratic-Spline curve');
    ylim([40 80]);
end


smothed = fp';
flatted = fwd_flat';
L = [calExcel ,flatted smothed];

% salva il risultati del pricing
```

```
save('ForwardPerPricing.mat','calendario_fwd','fwd_flat','fp');
```

9.2 Codice mail per l'esecuzione delle simulazioni Power Spot

Il seguente codice contente di eseguire le simulazioni del prezzo powe spot consistente con la curve forward i mercato.

```
%% Prepara il WorkSpace
close all
clear variables
clc

% For reproducibility
seed = 4;

%% Ottieni i dati di Input
fileDati = 'data.xlsx';
ds = datastore(fileDati);
ds.Range = 'A1:B366';

dati = read(ds);
dateAnno = dati.Date;
prezziSpot = dati.PUN;

logPrices = log(prezziSpot);

% Correlaizone tra componente di breve e di lungo
rho = 0;

% Crea la matrice di destagionalizzazione
[C,macroC,microC] = CreateDeseasonalizationMatrix(dateAnno);

% destagionalizza la serie storica
seasonalityParams = C\logPrices;

% Ricava la componente stocastica
X = logPrices - C*seasonalityParams;


%% Fitting dei parametri della parte stocastica

% Intervallo temporale
dt = 1/365;

% Prepara lo stimatore di massima verosimiglianza
Pt = X(2:end);
Pt_1 = X(1:end-1);

mrjpdf = @(Pt, a, phi, mu_J, sigmaSq, sigmaSq_J, lambda) ...
    lambda.*exp((-(Pt-a-phi.*Pt_1-mu_J).^2)./ ...
    (2.*(sigmaSq+sigmaSq_J))).* ...
        (1/sqrt(2.*pi.*(sigmaSq+sigmaSq_J))) + ...
    (1-lambda).*exp((-(Pt-a-phi.*Pt_1).^2)/(2.*sigmaSq)).* ...
    (1/sqrt(2.*pi.*sigmaSq));

% Constraints:
% phi < 1 (k > 0)
% sigmaSq > 0
```

```
% sigmaSq_J > 0
% 0 ≤ lambda ≤ 1

lb = [-Inf -Inf -Inf 0 0 0];
ub = [Inf 1 Inf Inf Inf 1];

% Initial values
x0 = [0 0 0 var(X) var(X) 0.5];

% Solve maximum likelihood
params = ...
    mle(Pt,'pdf',mrjpdf,'start',x0,'lowerbound',lb,'upperbound',ub,...
    'optimfun','fmincon');


% Obtain calibrated parameters
% alpha = params(1)/dt;
kappa = (1-params(2))/dt;
mu_J = params(3);
sigmaS = sqrt(params(4)/dt);
sigma_J = sqrt(params(5));
lambda = params(6)/dt;


%% Crea la parte deterministica giornaliera da montare sulla ...
    nuova curva fwd
load('ForwardPerPricing.mat');

% Prendi i parametri relativi ai giorni della settimana
weeklyParams = seasonalityParams(6:12);

% Crea l'indice per lo shape
idxShape = weeklyParams./mean(weeklyParams);
howManyRep = ceil(numel(calendario_fwd)/numel(idxShape))+5;
idxShape = repmat(idxShape,howManyRep,1);

% Ora crea lo shape per la curva forward
firstDay = weekday(calendario_fwd(1));
idxShape = idxShape(firstDay:firstDay + numel(calendario_fwd)-1)';

% Ora monta lo shape sulla curva Forward Smooth
fwdCurveIta = fp.*idxShape;


%% Ora simula i prezzi Spot
T = numel(fwdCurveIta)./365;
nDates = numel(fwdCurveIta);
nSim = 5000;

sigmaL = 0.21; % Volatility di lungo periodo
rng(seed);
XSim = ...
    SimulaLogPrices(sigmaL,sigmaS,lambda,kappa,mu_J,sigma_J,...
    nSim,nDates,T,rho);


% Crea le simulazioni NON RISK NEUTRAL
spotItaSimRealProbability = repmat(fwdCurveIta,nSim,1).*exp(XSim);

% Calcola il premio per il Rischio
MPR1_UN = ...
    CalcolaMartetPriceOfRisk...
```

```
111      (nDates,spotItaSimRealProbability,...
112      fwdCurveIta,dt);
113
114
115  % Setta il seme generatore simulazioni
116  rng(seed);
117  % Esegui le simulazioni Risk-Neutral
118  [XSimRN,S,L] = ...
119     SimulaLogPrices(sigmaL,sigmaS,lambda,kappa,mu_J,sigma_J,...
120     nSim,nDates,T,rho,MPR1_UN);
121
122  % Crea le simulazioni Spot
123  spotItaSimRiskNeutral = repmat(fwdCurveIta,nSim,1).*exp(XSimRN);
```

9.3 Funzioni Ausiliarie

In questa sezione sono riportarte le funzioni ausiliarie agli script **`main.m`** e necessarie al loro funzionamento.

9.3.1 Destagionalizzazione della serie di prezzi Spot

Questa funzione prende in input la serie storica dei log-price e ne calcola la componente macro e micro stagionale, resituendola.

```
1   function [C,macroC,microC] = CreateDeseasonalizationMatrix(dateAnno)
2   % Ottieni l'anno
3   Anno = unique(year(dateAnno));
4
5   % holidays
6   vacanze = [datetime(Anno,01,01),...
7           datetime(Anno,01,06),...
8           datetime(Anno,04,25),...
9           datetime(Anno,05,01),...
10          datetime(Anno,06,02),...
11          datetime(Anno,08,15),...
12          datetime(Anno,11,01),...
13          datetime(Anno,12,08),...
14          datetime(Anno,12,25),...
15          datetime(Anno,12,26),...
16      ];
17
18  % Numero di giorni.
19  nDay = numel(dateAnno);
20
21  % Crea la matrice di microstagonalita
22  microC = zeros(nDay,7);
23
24  for i = 1:nDay
25      % Se e' un giorno di business ma non e' sabato o domenica ...
            trattalo come domenica
26      currentDate = dateAnno(i);
27      if(ismember(currentDate,vacanze) && ¬...
            (weekday(currentDate)==1 || ...
28              weekday(currentDate)==7))
29          microC(i,1) = 1;
30      else
31          microC(i,weekday(currentDate)) = 1;
32      end
```

```
33 end
34
35 % Ora crea la matrice di Macrostagionalita
36 seasonMatrix = @(t) [sin(2.*pi.*t) cos(2.*pi.*t) sin(4.*pi.*t) ...
37     cos(4.*pi.*t) t];
38
39 ttime = linspace(0,1,nDay)';
40
41 % Matrici di destagionalizzazione
42 macroC = seasonMatrix(ttime);
43
44 % Matrix C
45 C = [macroC microC];
46
47 end
```

9.3.2 Simulazione dei log-prices $X_t = \chi_t + \xi_t$

Questa funzione contente la simulazione dei log-prices, ovvero della componente $X_t = \chi_t + \xi_t$.

```
1  function [X,...
2      S,L] = ...
3      SimulaLogPrices(sigmaL,sigmaS,lambda,k,mu_J,sigma_J,...
4      Nsim,Ndates,T,rho,MpR1)
5  % Numero di argomenti minore di dieci
6  if(nargin<11)
7      disp('Real Probability');
8      MpR1=zeros(Ndates,1);
9  end
10
11 % Calcola il dt
12 dt = T/Ndates;
13
14 % Vector of times
15 t = linspace(0,T,Ndates);
16
17 S = nan(Nsim,Ndates);
18 L = nan(Nsim,Ndates);
19
20 % Inizializza
21 S(:,1) = 0;
22 L(:,1) = 0;
23
24 % Crea le variabili correlate
25 X1 = randn(Nsim*Ndates,1);
26 X2 = randn(Nsim*Ndates,1);
27 X3 = rho*X1 + sqrt(1-rho.^2)*X2;
28
29 WL = reshape(X1,Nsim,Ndates);
30 WS = reshape(X3,Nsim,Ndates);
31
32 for i = 1:Nsim
33
34     % Simula gli istanti di salto
35     [isJumpTime,Njumps] = SimulateJumpTimes(lambda,T,t);
36
37     Jumps = mu_J + sigma_J.*randn(Njumps,1);
38
39     if(mod(i,2500)==0)
```

```
40          fprintf('Sim: %d di %d\n',i,Nsim);
41      end
42
43      cc = 1;
44
45      for j = 2:Ndates
46
47          L(i,j) = L(i,j-1) - MpR1(j-1).*dt + ...
                sigmaL.*sqrt(dt).*WL(i,j);
48          S(i,j) = S(i,j-1) + k(1).*(0 - S(i,j-1)).*dt + ...
                sigmaS.*sqrt(dt).*WS(i,j);
49
50          % Se e' un salto
51          if(isJumpTime(j))
52              S(i,j) = S(i,j) + Jumps(cc);
53              cc = cc + 1;
54          end
55      end
56
57  end
58
59  % Restituisci il processo complessivo (Short + Long)
60  X = S + L;
61
62  end
```

9.3.3 Calcolo del Premio per il Rischio γ_t

Questa funzione calcola il premio per il rischio γ_t necessario per la creazione di simulazioni consistenti con la curva forward.

```
1   function [X,...
2       S,L] = ...
3       SimulaLogPrices(sigmaL,sigmaS,lambda,k,mu_J,sigma_J,...
4       Nsim,Ndates,T,rho,MpR1)
5   % Numero di argomenti minore di dieci
6   if(nargin<11)
7       disp('Real Probability');
8       MpR1=zeros(Ndates,1);
9   end
10
11  % Calcola il dt
12  dt = T/Ndates;
13
14  % Vector of times
15  t = linspace(0,T,Ndates);
16
17  S = nan(Nsim,Ndates);
18  L = nan(Nsim,Ndates);
19
20  % Inizializza
21  S(:,1) = 0;
22  L(:,1) = 0;
23
24  % Crea le variabili correlate
25  X1 = randn(Nsim*Ndates,1);
26  X2 = randn(Nsim*Ndates,1);
27  X3 = rho*X1 + sqrt(1-rho.^2)*X2;
28
29  WL = reshape(X1,Nsim,Ndates);
```

```
30  WS = reshape(X3,Nsim,Ndates);
31
32  for i = 1:Nsim
33
34      % Simula gli istanti di salto
35      [isJumpTime,Njumps] = SimulateJumpTimes(lambda,T,t);
36
37      Jumps = mu_J + sigma_J.*randn(Njumps,1);
38
39      if(mod(i,2500)==0)
40          fprintf('Sim: %d di %d\n',i,Nsim);
41      end
42
43      cc = 1;
44
45      for j = 2:Ndates
46
47          L(i,j) = L(i,j-1) - MpR1(j-1).*dt + ...
                sigmaL.*sqrt(dt).*WL(i,j);
48          S(i,j) = S(i,j-1) + k(1).*(0 - S(i,j-1)).*dt + ...
                sigmaS.*sqrt(dt).*WS(i,j);
49
50          % Se e' un salto
51          if(isJumpTime(j))
52              S(i,j) = S(i,j) + Jumps(cc);
53              cc = cc + 1;
54          end
55      end
56
57  end
58
59  % Restituisci il processo complessivo (Short + Long)
60  X = S + L;
61
62  end
```

9.3.4 Simulazione degli istanti di salto di J_t

Questa funzione permette di generare i tempi di salto del processo jump J_t

```
1   function [isJumpTimes,Nt] = SimulateJumpTimes(lambda,T,t)
2   % Vettore che mi dice se e' un tempo di salto
3   isJumpTimes = false(size(t));
4
5   % genera il numero di salti
6   Nt = poissrnd(lambda*T);
7
8   % Continuos Jump times
9   CJT = T.*rand(Nt);
10
11  for i = 1:Nt
12      [¬,idxMIN] = min(abs(t-CJT(i)));
13      isJumpTimes(idxMIN) = true;
14  end
15
16  end
```

Riferimenti bibliografici

[1] F.E Benth, J.S. Benth, and S. Koekebakker. *Stochastic Modelling of Eletricity and Related Markets*. Workd Scientific, 2008.

[2] D. Brigo, A. Dalessandro, M. Neugebauer, and F. Triki. A stochastic processes toolkit for risk management, Novembrer 2007.

[3] A. Cartea and M.G. Figueroa. Pricing in electricity markets: a mean reverting jump diffusion model with seasonalit. 2005.

[4] A. Cartea and P. Villaplana. Spot price modeling and the valuation of electricity forward contracts: The role of demand and capacity. *Journal of Banking and Finance*, (12):2502–2519, 2008.

[5] N. Cufaro Petroni and P. Sabino. Pricing exchange options with correlated jump-diffusion processes. *Quantitative Finance*, 2017.

[6] S. Eduardo and J. Smoth. Short-term variations and long-term dynamics in commodity prices. *Management Science*, (7):893–911, 2000.

[7] T. Kluge. *Pricing Swing Options and other Electricity Derivatives*. PhD thesis, University of Oxford, 2006.

[8] The Mathworks. `https://it.mathworks.com/help/fininst/examples/simulating-electricity-prices-with-mean-reversion-and-jump-diffusion.html` (2005/06/12).

[9] J. Saifert and M. Uhrig-Homburg. Modelling jumps in electricity prices: Theory and empirical evidence. 2006.

[10] P. Tankov and R. Cont. *Financial Modeling with Jump Processes*. Chapman and Hall, 2004.