Università degli studi di Parma
Francesco Zanni

Modelli Matematici per la Finanza

Indagine sul potenziale speedup quadratico per la valutazione di opzioni tramite metodo Monte Carlo usando algoritmi di computazione quantistica

Obiettivo

L'obiettivo di questo lavoro è quello di condurre una prima indagine riguardo algoritmi quantistici per la valutazione di opzioni finanziarie europee.
L'aspettativa è quella di ottenere uno speedup quadratico rispetto alla controparte classica, migliorandone quindi la convergenza. Verrà osservato un algoritmo presente all'interno del pacchetto qiskit messo a disposizione da IBM e implementato un algoritmo unario, entrambi i quali saranno applicati a sistemi quantistici simulati in assenza di rumore.

Sommario

1. Introduzione alla meccanica quantistica

Vista la necessità di utilizzare sistemi quantistici si introducono brevemente i concetti di sistema quantistico e misura in meccanica quantistica.
Un sistema quantistico è un sistema fisico identificato da una funzione d'onda, spesso rappresentata con la notazione $|\psi\rangle$, un elemento di questo tipo è chiamato "ket".
La funzione d'onda rappresenta lo stato del sistema e ne determina di conseguenza le proprietà.
L'equazione che governa l'evoluzione di uno stato quantistico caratterizzato da una particella è l'equazione di Schrödinger, che rispetto alla base della posizione spaziale assume questa forma:

$$\left(-\frac{\hbar^2}{2m}\Delta + V(\vec{x},t)\right)\psi(\vec{x},t) = i\hbar\frac{\partial\psi(\vec{x},t)}{\partial t} \quad (1)$$

Nell'equazione compaiono:

- La costante di Planck ridotta $\hbar$;
- La massa m della particella
- Il potenziale V al quale il sistema è soggetto;
- La funzione d'onda $\psi(\vec{x},t)$, che in questo caso è rappresentata sulla base della posizione e di conseguenza è funzione dello spazio $\vec{x}$ e del tempo t;
- L'operatore laplaciano $\Delta = \sum_i \frac{\partial^2}{\partial^2 x_i}$
- L'unità immaginaria i: la funzione d'onda è infatti a valori complessi

Per potenziali V indipendenti dal tempo l'equazione è a variabili separabili, di conseguenza l'unica soluzione possibile è che entrambi i membri siano costanti. Ponendo costante il primo membro si ottiene l'equazione di Schrödinger indipendente dal tempo:

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + V\psi = E\psi$$

A questo punto, definendo l'hamiltoniana del sistema ovvero l'energia, e ricavando l'operatore corrispondente attraverso la sostituzione canonica: $p \to \frac{\hbar}{i}\frac{\partial^2}{\partial x^2}$, è immediato osservare che l'equazione precedente è un'equazione agli autovalori per l'operatore Hamiltoniana:

$$\widehat{H}\psi = E\psi$$

Questo viene poi generalizzato ad hamiltoniane più complesse e a più corpi.
La funzione d'onda è un vettore appartenente ad uno spazio di Hilbert, ovvero uno spazio vettoriale complesso dotato di un prodotto scalare e di una metrica indotta da esso.
È possibile esplicitare la funzione d'onda rispetto a basi differenti, in precedenza è stata esplicitata rispetto alla posizione $\vec{x}$, ma è possibile farlo anche per il momento $\vec{p}$, l'energia o altre grandezze osservabili.
L'interpretazione della meccanica quantistica in termini di operatori postula che gli unici stati ammissibili per il sistema, quindi le uniche funzioni d'onda, siano autovettori dell'hamiltoniana e le uniche energie possibili gli autovalori.
Siccome gli autovettori identificano uno stato del sistema è comune chiamare gli stati così definiti "autostati" dell'hamiltoniana.
È importante osservare che la linearità dell'equazione di Shrödinger fa sì che anche una combinazione lineare di soluzioni sia a sua volta soluzione, rendendo possibile l'esistenza di funzioni d'onda che sono combinazioni di autovettori, esse rappresentano stati caratterizzati dalla sovrapposizione di autostati differenti. Per esempio, supponendo che un'hamiltoniana abbia spettro discreto con n autovalori differenti e identificando con $|\psi_n\rangle$ i relativi autovettori, lo stato generale del sistema è dato da:

$$|\Psi\rangle = \sum_n \alpha_n |\psi_n\rangle \quad (2)$$

È allora fondamentale definire in che modo uno stato è legato a quantità misurabili (chiamate osservabili) di un sistema fisico e in particolare cosa si intende con *misura* di un'osservabile: ogni osservabile fisica di un sistema, così come l'energia citata precedentemente, è rappresentata in meccanica quantistica da un operatore autoaggiunto ed i possibili valori che una misura può fornire sono gli autovalori dell'operatore corrispondente.
Un operatore autoaggiunto A, che agisce sullo spazio vettoriale della funzione d'onda, è un operatore tale che, una volta definito il suo operatore hermitiano aggiunto A^*, rispetti l'identità $A^* = A$. Questa proprietà è sufficiente per garantire che tutti gli autovalori di un operatore autoaggiunto sono reali. Si garantisce di conseguenza che, nonostante la funzione d'onda appartenga ad uno spazio vettoriale complesso, i risultati di una misura, ovvero gli autovalori dell'operatore autoaggiunto associato, siano sempre reali.
Per esempio, come già visto, l'operatore associato all'osservabile "energia del sistema" è l'hamiltoniana $\widehat{H}$, gli autovalori di questo operatore rappresentano tutti e i soli valori che può assumere l'energia del sistema descritto da quell'hamiltoniana.
In presenza di uno stato sovrapposizione, nel momento della misura si dice che lo stato "collassa" nell'autostato corrispondente all'autovalore misurato, con una probabilità che è proporzionale al quadrato del coefficiente di quello stato. Riprendendo lo stato definito in (1) e supponendo che esso sia *normalizzato*, cioè che $\sum_n |\alpha_n| = 1$, allora la misura di un'osservabile può fornire ognuno degli n autovalori, ciascuno con probabilità $|\alpha_n|^2$.

2. Introduzione alla computazione quantistica

In computazione quantistica l'unità di informazione, che svolge la controparte del bit classico, è chiamata qubit, ovvero quantum bit.
Un qubit è un sistema quantistico che per qualche osservabile presenta esattamente due autostati ortogonali tra loro, identificati con $|0\rangle$ e $|1\rangle$: questa rappresentazione è chiamata "ket" e verrà descritta con più precisione successivamente. L'idea fondamentale è che avendo esattamente due stati una misura eseguita sul sistema debba restituire necessariamente uno dei due autovalori del relativo autostato associato all'operatore che rappresenta l'osservabile misurata.
È utile inizialmente visualizzare il qubit come una particella a spin ½ ed utilizzare come operatore la matrice di Pauli $S_z = \frac{\hbar}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, che è l'operatore associato all'osservabile "momento angolare lungo l'asse z".
Gli autovettori della matrice sono $\frac{\hbar}{2}\binom{1}{0}$ e $\frac{\hbar}{2}\binom{0}{1}$, che rappresentano rispettivamente gli stati con momento angolare lungo z uguale a $\frac{\hbar}{2}$ e $-\frac{\hbar}{2}$.
Una misura di questa osservabile condotta sul sistema fornirà di conseguenza uno di questi due valori, che per essere interpretati come bit di informazione vengono mappati in 0 e 1.

Come verrà mostrato c'è una corrispondenza tra la notazione "ket" e quella vettoriale.

Questo concetto può essere generalizzato ad altri tipi di sistemi, con il solo vincolo che lo spettro dell'operatore associato agli autostati utilizzati per definire il qubit abbia dimensione 2.

Se il sistema si trova in uno dei due autostati allora codifica un bit di informazione classico ed eseguendo una misura, questa fornirà con certezza l'autovalore relativo all'autostato in cui si trova il sistema.
Il modo in cui i qubit si differenziano dai bit classici è che essi possono trovarsi in qualsiasi sovrapposizione dei due stati:

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$$

In questo caso il risultato di una eventuale misura può fornire ciascuno dei due valori, rispettivamente con probabilità $|a_0^2|$ e $|a_1^2|$(assumendo che lo stato sia normalizzato). Questa caratteristica permette di svolgere operazioni che non sono possibili con computer classici, per esempio eseguire calcoli in parallelo: preparando il sistema in uno stato iniziale che è sovrapposizione di più stati è possibile eseguire in una sola operazione un calcolo su tutta la base di autostati in modo parallelo.

Si prepara per esempio una sovrapposizione di stati, e si applica ad essa una trasformazione, lo stato finale può essere visto come una sovrapposizione di tutti i risultati possibili.
Per quanto detto sulla misura però, misurando lo stato del sistema esso collasserà in un autostato. Per rendere utile quindi questo calcolo in parallelo è necessario trovare soluzioni alla semplice misura, che permettano di avere informazioni su tutto lo stato finale.

2.1 Rappresentazione

Come accennato precedenza le due notazioni più comuni sono:

- **Notazione Braket:**

Lo stato di un qubit, come mostrato fino a questo punto, è rappresentato da una sovrapposizione dei due possibili autostati: $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ .Per rappresentare gli operatori che agiscono sugli stati si fa uso dello spazio duale: chiamato $\mathcal{H}$ lo spazio di Hilbert al quale appartiene lo stato $|\psi\rangle$, si definisce lo spazio duale $\mathcal{H}^*$ come lo spazio delle applicazioni lineari che vanno da $\mathcal{H}$ ai numeri complessi, i vettori di questo stato vengono chiamati "bra" e sono identificati con $\langle\psi|$. Attraverso questa notazione si può esprimere il prodotto scalare tra i due vettori $|\psi\rangle$ e $|\phi\rangle$ con $\langle\psi|\phi\rangle$, definendo in questo modo l'azione dell'applicazione lineare rappresentata da ciascun bra.

Per rappresentare un operatore quindi si accosta un ket ad un bra, in modo tale da ottenere un'operazione che porta lo stato di partenza in un altro stato appartenente allo stesso spazio:

$$(|\psi\rangle\langle\phi|)\ |\gamma\rangle = |\psi\rangle(\langle\phi|\gamma\rangle)$$

L'operatore in questo caso porta $|\gamma\rangle$ nello stato $|\psi\rangle$ moltiplicato per un coefficiente dato dal prodotto scalare fra $|\phi\rangle$ e $|\gamma\rangle$. Ovviamente anche combinazioni lineari di operatori (ket-bra) sono a loro volta operatori.

Siccome le basi che verranno utilizzate in seguito sono tutte ortonormali, per il resto della trattazione si avrà che, dati due vettori di base $|\psi\rangle$ e $|\phi\rangle$: $\langle\phi|\psi\rangle = \delta_{\psi,\phi}$.

In particolare, si mostrano 4 operatori usati frequentemente in computazione quantistica: l'operatore identità e gli operatori di Pauli, legati a proprietà delle particelle a spin ½.

$$\begin{aligned} \hat{I} &= |0\rangle\langle 0| + |1\rangle\langle 1| \\ \hat{\sigma}_x &= |0\rangle\langle 1| + |1\rangle\langle 0| \\ \hat{\sigma}_y &= i(|1\rangle\langle 0| - |0\rangle\langle 1|) \\ \hat{\sigma}_z &= |0\rangle\langle 0| - |1\rangle\langle 1| \end{aligned} \qquad (3)$$

Questi 4 operatori formano inoltre una base dello spazio degli operatori agenti su un qubit.

Passando alla rappresentazione di stati a più qubit formalmente la scrittura corretta prevede l'utilizzo del prodotto tensoriale, definito, per il prodotto tra due vettori di stato di un qubit nel seguente modo:

$$|a\rangle = a_0|0\rangle + a_1|1\rangle, \quad |b\rangle = b_0|0\rangle + b_1|1\rangle \quad \Rightarrow$$

$$|a\rangle \otimes |b\rangle = |ab\rangle = a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_0|10\rangle + a_1b_1|11\rangle$$

Il risultato di questo prodotto è un ket che appartiene ad uno spazio di Hilbert la cui dimensione è data dal prodotto delle dimensioni degli spazi di partenza. Nel nostro caso, siccome gli spazi di partenza hanno entrambi dimensione 2, il prodotto tensoriale ha dimensione 4 e si mostra che una base è data dalle 4 combinazioni degli stati di base dei vettori di partenza.

Questa definizione è generale e si applica analogamente a vettori di dimensioni maggiori.
Per comodità di scrittura gli stati a più qubit vengono accorpati all'interno di un unico ket, sottintendendo la seguente uguaglianza:

$$|\psi_1\rangle \otimes |\psi_2\rangle \otimes \dots |\psi_n\rangle =: |\psi_1\psi_2 \dots \psi_n\rangle$$

- **Notazione vettoriale:**

In questa notazione lo stato e gli operatori sono rappresentati rispettivamente da vettori colonna, che rappresentano le componenti lungo i possibili autostati di base e matrici (si mostrano come esempio la matrice identità e le tre matrici di Pauli, corrispondenti agli operatori mostrati in (3)

$$\psi = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \qquad \hat{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \hat{\sigma}_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \hat{\sigma}_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \hat{\sigma}_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad (4)$$

Mettendo a confronto le due notazioni si sottolineano le seguenti relazioni:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \Rightarrow \quad |\psi\rangle = \psi_0|0\rangle + \psi_1|1\rangle = \psi_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \psi_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix}$$

Per quanto riguarda gli stati a più qubit vale la stessa notazione, ma aumenta la dimensione dei vettori e delle matrici, in generale, dati n qubit lo stato del sistema è un vettore di 2^n componenti e gli operatori sono rappresentati da matrici di dimensione $2^n x 2^n$. Anche in questo caso si utilizza in prodotto tensoriale, e rimane valido che i vettori o operatori ottenuti abbiano come dimensione il prodotto delle dimensioni dei vettori di partenza, per esempio:

$$a = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix},\ b = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \qquad \Rightarrow\ a \otimes b = \begin{pmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{pmatrix}$$

In questo caso le componenti del vettore si riferiscono ai coefficienti di ciascuno dei 4 autostati possibili

A seconda del contesto una notazione è preferibile rispetto all'altra, in particolare per esplicitare lo stato di un sistema si usa spesso la notazione braket rendendo facile omettere gli elementi di base per i quali il coefficiente è nullo, mentre per rappresentare operatori è di più immediata visualizzazione la rappresentazione matriciale.

Si fa ora un esempio di stato ed operatori in entrambe le notazioni per un sistema a 2 qubit:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$CNOT = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10| = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$CNOT(|\psi\rangle) = \frac{1}{\sqrt{2}}(|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10|)(|00\rangle + |10\rangle)$$
$$= \frac{1}{\sqrt{2}}(\langle 00|00\rangle|00\rangle + \langle 10|10\rangle|11\rangle) = \frac{1}{\sqrt{2}}\ (|00\rangle + |11\rangle)$$

$$CNOT(\psi) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

2.2 Gates quantistici

La computazione quantistica, esattamente come quella classica si basa sull'applicazione di porte logiche, chiamate gates.
Come i gates classici, i gates quantistici sono operazioni che, partendo da uno stato iniziale, lo trasformano in uno stato finale.
Ogni gate quantistico è definito attraverso un operatore unitario e ogni operatore unitario della dimensione corretta può essere utilizzato per definire un gate quantistico.
Un operatore unitario è un operatore tale che, definendo con $U^\dagger$ il suo operatore hermitiano aggiunto, $U^\dagger U = UU^\dagger = \mathbb{1}$, questa proprietà è necessaria per garantire che lo stato ottenuto dall'applicazione di un gate quantistico a partire da uno stato normalizzato sia a sua volta normalizzato, garantendo quindi la conservazione della probabilità.

Fra i gates quantistici a 1 qubit più comuni si osservano:

- Gates di Pauli (X, Y, Z): applicano la trasformazione unitaria rappresentata dalla relativa matrice di Pauli (mostrate sopra)
- Gate Hadamard (H): rappresentato dalla matrice $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. Particolarità di questo gate è che applicandolo ad un qubit nello stato $|0\rangle$ si ottiene lo stato finale $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, cioè una sovrapposizione uniforme dei due stati possibili. Di conseguenza, partendo dallo stato iniziale $|0^n\rangle$ ed applicando un gate Hadamard ad ogni qubit si ottiene lo stato che presenta la sovrapposizione di ogni stato possibile con distribuzione di probabilità uniforme: $\hat{H}^n|0^n\rangle = \frac{1}{\sqrt{2}^n}\sum_i |i\rangle$

Fra i gates a più qubit invece i più utilizzati sono:

- Gate di tipo "controlled": l'operazione prevista viene eseguita sul secondo qubit solo se il primo, chiamato qubit di controllo si trova nello stato $|1\rangle$, mentre sul primo qubit non si applica nessuna trasformazione. Per esempio, un gate comune è il CNOT, ovvero il controlled-not: si applica il gate logico NOT al secondo qubit solo se il primo si trova nello stato $|1\rangle$.
- Swap gate: gate che scambia lo stato di due qubit. Usato spesso su sistemi reali, dove i qubit non sono "fully connected", ovvero dove non tutti i qubit possono comunicare con tutti gli altri, si mostra in figura un esempio di architettura di un sistema reale, In questo caso è necessario applicare una serie di swap gate fino ad "avvicinare" i qubit che si vogliono mettere in comunicazione attraverso gates a più qubit. Nell'esempio della figura, volendo per esempio applicare un gate ai qubit 4 e 6, che non comunicano direttamente tra loro è necessario procedere prima con uno swap fra 4 e 5, applicare il gate desiderato a 5 e 6 ed infine riportare l'ampiezza desiderata sul qubit 4 attraverso un ultimo swap ai qubit 4 e 5.

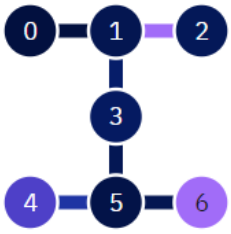


Figura 1: Architettura del sistema IBM_oslo

Una considerazione importante riguardante i gates a più qubit è che, essendo operatori unitari essi soddisfano due importanti caratteristiche: il numero di output del gate deve essere uguale al numero di input, e deve essere conservato il prodotto scalare fra due stati differenti prima e dopo l'applicazione del gate, per esempio un gate che a due stati differenti associa lo stesso output non può esistere in quanto violerebbe la seconda condizione.
Questa richiesta limita le operazioni che possono essere rappresentate attraverso gates quantistici, per esempio operazioni logiche alla base della computazione classica come gate NOT, OR o AND non possono avere una controparte quantistica, in quanto presentano un solo bit di output a partire da due bit in input, per questo in computazione quantistica sono particolarmente diffusi i gate di tipo controlled citati in precedenza.

2.3 Circuiti Quantistici

Il modo più comune di visualizzare un algoritmo di computazione quantistica è attraverso i circuiti quantistici: si rappresenta ogni qubit con una linea orizzontale, sulla quale si inseriscono in ordine di applicazione da sinistra a destra tutti i gates utilizzati, identificati da una sigla fino ad arrivare, solitamente, alla misura di uno o più qubit.

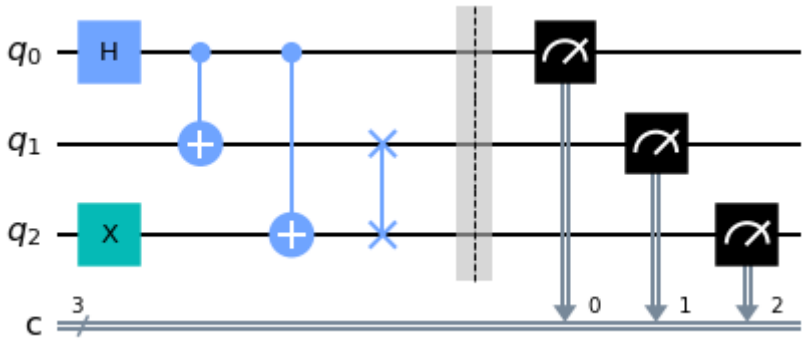


Figura 2: Esempio di circuito quantistico

I gates che riguardano più bit sono rappresentati da una linea verticale che li connette, in particolare i gates rappresentati nell'immagine sono:

- CNOT: come tutti i gates di tipo controlled si rappresenta il controllo sul qubit identificato dal pallino pieno e l'operatore identificato dal simbolo sull'altro qubit, in questo caso, si identifica col simbolo "+" in quanto si può interpretare il gate logico NOT come la somma modulo 2 di un'unità.
- Swap: viene identificato da una croce su ciascuno dei due qubit

Il separatore verticale successivo allo swap-gate è stato inserito solamente come separazione visiva tra il circuito vero e proprio e la misura, che avviene come ultima sezione del circuito. In questo caso, ma non necessariamente, si misurano tutti i 3 qubit e il risultato della misura viene scritto su un registro classico c, ovvero 3 bit di memoria classica.

3. Introduzione al modello Black-Scholes

3.1 Opzioni

Un' opzione finanziaria europea di tipo call è un contratto che dà il diritto (ma non l'obbligo) a chi lo detiene (holder) di acquistare un determinato asset, a un determinato prezzo chiamato strike price (K) a una data futura specificata, chiamata scadenza o maturity e indicata con T.

Analogamente un'opzione europea di tipo put offre il diritto di vendere l'asset sotto le stesse condizioni (strike price e scadenza).Siccome il contratto offre il diritto ma non il dovere di esercitare l'opzione il detentore deciderà di esercitare il diritto solo nel caso in cui questo porti un guadagno, è possibile quindi rappresentare matematicamente il profitto ottenuto alla data di scadenza(payoff) del contratto nel seguente modo:
Call Europea: $V(T) = max(S(T) - K, 0)$
Put Europea: $V(T) = max(K - S(T), 0)$

Dove si identifica con $V(T)$ il valore dell'opzione al momento della scadenza, con K lo strike price e con S(t) il prezzo dell'asset, siccome la call è europea può essere esercitata

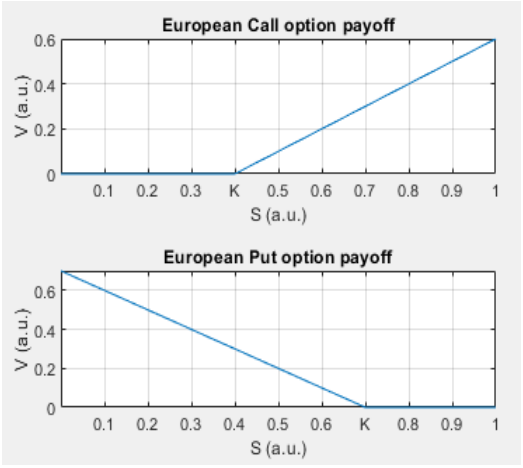


Figura 3: Funzione payoff per i due tipi di opzione europea

soltanto alla data di scadenza il valore utilizzato per calcolare il payoff è S(T)

Una proprietà importante è chiamata put-call parity, essa riguarda due opzioni, una di tipo call e una di tipo put, con parametri identici e sfrutta il fatto che a parità di parametri la differenza tra il payoff di una put e quello di una call non è altro che K-S:

$$S + V_p^{eu} - V_c^{eu} = Ke^{-r(T-t)} \quad (5)$$

Il fattore esponenziale è necessario per attualizzare all'istante t generico lo strike price, si fa infatti uso di r, ovvero del tasso di interesse "risk free", che in questa trattazione è considerato costante durante la vita del contratto.

3.2 Modello Black-Scholes

L'obiettivo primario è quello di dare un giusto valore all'opzione, ovvero valutare $V(t)$, per fare ciò il modello capostipite che viene utilizzato è il modello di Black-Scholes:
Esso si basa su alcune ipotesi:

1. Il mercato non presenta opportunità di arbitraggio, ovvero non esistono portafogli in grado di avere una garanzia di guadagno entro una scadenza
2. Il mercato è efficiente, anche detto "frictionless", ovvero
    a. Non ci sono costi di transazione
    b. I tassi di interesse per prestare e prendere in prestito denaro sono identici
    c. Tutti gli individui hanno accesso completo ed uguale a tutte le informazioni disponibili
    d. Tutti i titoli sono disponibili in ogni momento ed in qualsiasi importo
    e. Il trading individuale non ha influenze sul prezzo di un asset
3. Tasso di interesse r(risk-free rate) e volatilità σ degli asset sono considerati costanti nell'arco di vita dell'opzione considerata
4. L'opzione è di tipo europeo (ovvero esercitabile soltanto alla scadenza)
5. Il prezzo dell'asset sottostante segue un moto Browniano geometrico $W(t)$ definito nel seguente modo:
    a. $W(0) = 0$
    b. $W(t) \approx N(0, t)$
    c. Incrementi su intervalli di tempo disgiunti sono indipendenti
    d. $W(t)$ dipende in modo continuo dal tempo t
    e. Il moto soddisfa la seguente equazione differenziale stocastica (ipotesi che lo rende geometrico):

$$dS_t = rS_t dt + \sigma S_t dW_t$$

Seguendo queste ipotesi e applicando il lemma di Ito, il quale permette di rimuovere l'aleatorietà e considerare come variabili indipendenti il tempo t e il prezzo dell'asset S (che altrimenti sarebbero legati siccome $S = S(t)$) si ricava l'equazione differenziale di Black-Scholes:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \quad (6)$$

Con: $t \in [0,T], \quad S \in [0,+\infty), \quad V(S,T) = \max(S(T) - K, 0)$
L'equazione è di natura backward in quanto dipende dalla condizione finale data dal prezzo dell'asset alla scadenza T, ovvero dal payoff.

3.3 Soluzione

Per questo modello, che rappresenta il caso più semplice, è nota la soluzione analitica, scritta in termini di distribuzioni normali cumulate $\mathcal{F}(x)$:

$$\boldsymbol{V(S,t)} = S\,\mathcal{F}(d_1) - Ke^{-r(T-t)}\mathcal{F}(d_2)$$

Con:

$$\mathcal{F}(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt$$

$$d_1 = \left[\log\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right] \frac{1}{\sigma(T-t)}$$

$$d_2 = d_1 - \sigma(T-t)$$

Tuttavia, passando a contratti differenti e più complicati, come per esempio opzioni barriera basket, oppure rilassando le ipotesi non sempre è nota la soluzione analitica ed è necessario quindi ricorrere ad approssimazioni.

I metodi per approssimare il valore di opzioni vengono spesso testati sul caso più semplice, in modo da essere in grado di confrontare con una soluzione analitica, per poi venire applicati ai casi più complessi.

3.4 Metodo Monte Carlo classico

Uno dei modi per approssimare il valore di un'opzione è quello di ricorrere al metodo Monte Carlo, sfruttando quindi l'aleatorietà nell'evoluzione del prezzo dell'asset sottostante.

L'obbiettivo del metodo è quello di ottenere un valore dell'opzione inteso come valore di aspettazione del payoff.

L'ipotesi prevede che il sottostante segua un moto Browniano geometrico, ovvero che esso sia descritto dalla equazione differenziale stocastica mostrata in precedenza:

$$dS_t = rS_t dt + \sigma S_t dW_t$$

L'equazione mostra un termine di drift, che dipende dal risk free rate e induce un andamento positivo per il valore del sottostante, e uno diffusivo, che invece dipende dalla dinamica stocastica e quindi dalla volatilità e ne determina di conseguenza le fluttuazioni.
Questa equazione può essere integrata, ottenendo una soluzione che mantiene una componente

stocastica, rappresentata da $z \sim N(0,1)$:

$$S(t) = S_0 e^{(r-\frac{\sigma^2}{2})t} e^{\sigma\sqrt{t}z} \tag{8}$$

Il metodo Monte Carlo consiste nell'estrarre un numero elevato di valori di z dalla distribuzione di probabilità, ottenendo quindi una distribuzione di valori di $S(T)$ ed utilizzarli per calcolare i relativi payoff.

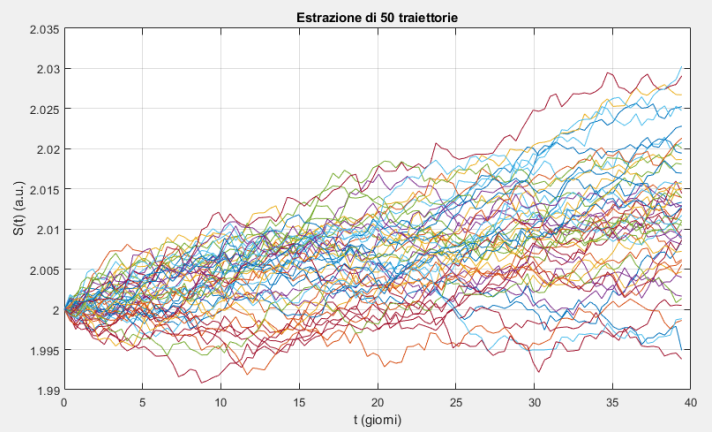


Figura 4: Esempio di estrazione di traiettorie generate casualmente

Si ottiene uno stimatore attraverso la media su n estrazioni:

$$\hat{V}_N = \frac{1}{N}\sum_{i=1}^{N} V_i$$

Questo stimatore presenta come valore di aspettazione il valore cercato:
$E\left[\hat{V}_N\right] = V = E[\exp(-\mathrm{rT})\ \max(S(T) - K, 0)]$ e vi converge all'aumentare di N.

Si osserva infatti che per il teorema del limite centrale al crescere di N il valore ottenuto di $\hat{V}_N$ tende ad essere estratto da una distribuzione normale di media $E\left[\hat{V}_N\right] = V$ e varianza $Var[V] = \frac{\sigma_V^2}{N}$, dove con σ_V si identifica la varianza della distribuzione dei valori singoli V_i.

Questa proprietà garantisce che la differenza tra lo stimatore e il valore esatto, ovvero l'errore associato a questo metodo sia di ordine pari alla radice della varianza, ovvero:

$$\hat{V}_N - V \sim\ O\left(\frac{1}{\sqrt{N}}\right)$$

È possibile quindi affermare che il metodo Monte Carlo ha ordine di convergenza $1/\sqrt{N}$.

In modo analogo è possibile utilizzare la disuguaglianza di Chebyshev per ricavare il numero di estrazioni necessarie al fine di mantenere costante l'errore commesso. A partire dalla varianza della distribuzione σ_V^2 la disuguaglianza fornisce la probabilità che la misura ottenuta dalla media di N estrazioni disti più di ε dal valore esatto:

$$P\left[\left|\hat{V} - V_{ex}\right| \geq \varepsilon\right] \leq \frac{\sigma_V^2}{\mathrm{N}\varepsilon^2}$$

Di conseguenza, volendo mantenere costante questa probabilità, riducendo però l'errore commesso ε è necessario eseguire un numero di estrazioni di ordine $N = \mathcal{O}\left(\frac{\sigma_V^2}{\varepsilon^2}\right)$, ritrovando la stessa relazione tra errore commesso e numero di estrazioni osservata in precedenza.

Questo ordine di convergenza è particolarmente sconveniente rispetto ad altre approssimazioni, come le differenze finite, soprattutto in basse dimensioni.

L'obbiettivo dell'implementazione degli algoritmi quantistici è quello di migliorare questo ordine di convergenza: si indagherà in particolare l'implementabilità di un algoritmo chiamato "$Amplitude\ Estimation$", il quale presenta un ordine di convergenza $1/N$, trattandosi quindi di un possibile speedup quadratico nella convergenza del metodo.

4. Implementazione

Si mostra ora il funzionamento generale di un algoritmo quantistico che implementa un metodo Monte Carlo.

Inizialmente si ha a disposizione un registro di qubit, il cui stato può essere identificato dal prodotto degli stati dei singoli qubit, per comodità è comune identificarlo con un unico ket, lo stato iniziale vede sempre tutti i qubit nello stato $|0\rangle$:

$$|\psi_0\rangle = |0\rangle \otimes |0\rangle \otimes \dots \otimes |0\rangle \coloneqq |0\rangle|0\rangle \dots |0\rangle \coloneqq |0 \dots 0\rangle$$

A questo punto viene applicata la serie di gates prevista dal circuito, svolgendo di fatto un algoritmo $\mathcal{A}$. Questo algoritmo porterà il sistema, in generale, in una sovrapposizione di stati, identificata da

$$|\psi_1\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$$

Dove α_i rappresenta il coefficiente dello stato i, il numero di diversi autostati possibili è 2^n, si può quindi interpretare i come il numero naturale codificato dalla stringa di bit del registro in quello stato.

Dalla misura di tutto il registro il sistema collassa in uno degli stati possibili, con probabilità $|\alpha_i|^2$

Esempio

Si mostra un esempio di circuito che presenta gates a 1 e 2 qubit. Lo stato iniziale per convenzione è sempre $|0^n\rangle$, inizialmente si applicano separatamente i due gate a 1 qubit:

1. Sul primo qubit si applica un Pauli X:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad , oppure \qquad (|0\rangle\langle 1| + |1\rangle\langle 0|)|0\rangle = |1\rangle$$

2. Sul secondo qubit si applica un gate Hadamard (H)

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix}=\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix},$$

$$oppure\,\frac{1}{\sqrt{2}}(|0\rangle\langle 0|+|0\rangle\langle 1|+|1\rangle\langle 0|-|1\rangle\langle 1|)|0\rangle=\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$$

3. Lo stato a 2 qubit risultante è il seguente (per la notazione braket è utile inserire a pedice l'indice relativo a quale qubit si trova nello stato indicato):

$$\begin{pmatrix}0\\ 1\end{pmatrix}\otimes\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}=\begin{pmatrix}0\\ 0\\ \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}\quad ,oppure\quad |1\rangle_1\otimes\frac{1}{\sqrt{2}}(|0\rangle_2+|1\rangle_2)=\frac{1}{\sqrt{2}}(|10\rangle+|11\rangle)$$

Si sottolinea che le componenti del vettore di dimensione 4 rappresentano rispettivamente i coefficienti dei 4 possibili stati $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$.
In questa situazione la notazione "ket" risulta più intuitiva dal punto di vista della visualizzazione dello stato, mentre la notazione matriciale sia più efficace nella visualizzazione di operatori.

4. Si applica infine il gate SWAP:

$$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}0\\ 0\\ \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}=\begin{pmatrix}0\\ \frac{1}{\sqrt{2}}\\ 0\\ \frac{1}{\sqrt{2}}\end{pmatrix}$$

Analogamente

$$(|00\rangle\langle 00|+|01\rangle\langle 10|+|10\rangle\langle 01|+|11\rangle\langle 11|)(\tfrac{1}{\sqrt{2}}(|10\rangle+|11\rangle))=\tfrac{1}{\sqrt{2}}(|01\rangle+|11\rangle)$$

Una misura sullo stato così costruito darà come possibili risultati le due stringhe 01 e 10 con uguale probabilità pari a 1/2.

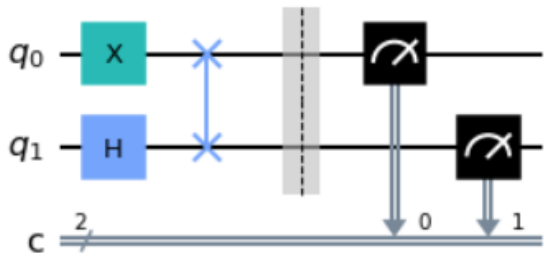


Figura 5: Rappresentazione del circuito descritto

4.1 Esempio di algoritmo Monte Carlo

Partendo da una distribuzione di probabilità nota, si mostra un primo algoritmo che ha lo stesso scaling del corrispettivo classico e di conseguenza non porta benefici dal punto di vista della complessità computazionale; tuttavia, è utile per mostrare il funzionamento di algoritmi quantistici.

Come visto in precedenza, a partire dallo stato iniziale $\left|0^{\otimes n}\right\rangle = |0 \dots 0\rangle$ si applica una serie di operatori che porta il sistema nello stato

$$|\psi_1\rangle = \mathcal{A}\left|0^{\otimes n}\right\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$$

L'operatore $\mathcal{A}$ riassume l'insieme di gates applicati al registro di qubit, e siccome agisce sul vettore di stato degli n qubit può essere rappresentato come una matrice $2^n \times 2^n$.
Si considera poi una funzione $v(x)$: $\{0,1\}^n \to \mathbb{R}$, che mappa ogni stringa binaria a un valore reale.
L'obiettivo è, dato lo stato finale, di ottenere il valore di aspettazione

$$\mathbb{E}[v(\mathcal{A}\left|0^{\otimes n}\right\rangle)] = \sum_{i=0}^{2^n-1} |\alpha_i|^2 v(i)$$

Ovviamente se a questo punto si misura il registro di qubit lo stato collassa in uno dei valori possibili, non è quindi efficace procedere in questo modo. Per questo si agisce con una rotazione su un qubit di appoggio (che viene messo in evidenza all'interno di un ket posto in seguito allo stato originale). Dato un autostato generico $|x\rangle$ si assume di poter definire un operatore di rotazione sul qubit di appoggio agisce in questo modo:

$$\mathcal{R}|x\rangle|0\rangle = |x\rangle(\sqrt{1 - v(x)}\,|0\rangle + \sqrt{v(x)}|1\rangle)$$

Applicando quindi la rotazione allo stato $|\psi_1\rangle|0\rangle$, dove $|\psi_1\rangle$ è la sovrapposizione creata in precedenza, l'operatore agirà, come appena definito, su ogni elemento della sovrapposizione, portando ad uno stato finale $|\chi\rangle$:

$$|\chi\rangle = \mathcal{R}\left(\sum_{i=0}^{2^n-1} \alpha_i |i\rangle\,|0\rangle\right) = \sum_{i=0}^{2^{n-1}} \alpha_i \mathcal{R}\,|i\rangle|0\rangle = \sum_{i=0}^{2^n-1} \{\alpha_i |i\rangle(\sqrt{1 - v(i)}\,|0\rangle + \sqrt{v(i)}|1\rangle)\} \;= \mathcal{F}\left|0^{\otimes n}\right\rangle|0\rangle$$

Con $\mathcal{F}$ riassume l'insieme delle operazioni eseguite in un unico operatore unitario, ovvero l'applicazione dell'operatore $\mathcal{A}$ sui primi n qubit e la successiva rotazione, sul qubit di appoggio.
L'operatore $\mathcal{F}$ ha di conseguenza dimensione $2^{n+1} \times 2^{n+1}$.
Ciò che si ottiene ora è che misurando il qubit di appoggio, la probabilità di ottenere lo stato $|1\rangle$ è esattamente il valore di aspettazione cercato.
In seguito alla misura, infatti il qubit di appoggio collassa nello stato misurato, il cui coefficiente è dato dal valore di aspettazione che si intende misurare, per mettere in evidenza le probabilità associate alle misure sul qubit di appoggio è possibile riscrivere lo stato $|\chi\rangle$ in questo modo:

$$|\chi\rangle = \left(\sum_{i=0}^{2^{n-1}} \alpha_i \sqrt{1 - v(i)}|i\rangle\right)|0\rangle + \left(\sum_{i=0}^{2^{n-1}} \alpha_i \sqrt{v(i)}|i\rangle\right)|1\rangle$$

La misura non fa quindi collassare lo stato dei primi n qubit, ma soltanto del qubit di appoggio, questa è la caratteristica che rende utile questo tipo di approccio.
Concludendo, la probabilità μ di ottenere 1 dalla misura del qubit di appoggio è esattamente:

$$\mu = \langle\chi|(I_{2^n} \otimes |1\rangle\langle 1|)|\chi\rangle = \sum_{i=0}^{2^n-1} |\alpha_i|^2 v(i) = \mathbb{E}[v(\mathcal{A})]$$

Ciò che è possibile fare è ripetere n volte la misura e ottenere così una stima della probabilità cercata.

Procedendo in questo modo lo scaling è lo stesso del Monte Carlo classico, e di conseguenza non c'è vantaggio nell'utilizzo di un computer quantistico, se non che l'aleatorietà non è simulata attraverso algoritmi classici ma intrinseca nella fisica dei qubit.

Per avere un vantaggio bisogna quindi trovare un algoritmo per stimare il valore di aspettazione che si cerca in un modo differente, questo viene fatto attraverso un algoritmo chiamato Amplitude Estimation.

4.2 Ampitude distribution per circuiti unari

Il problema iniziale da risolvere però sta nel primo passaggio: ottenere la corretta distribuzione di probabilità come coefficienti degli autostati appartenenti alla base unaria. L'algoritmo che svolge questo processo è chiamato Amplitude distributor.

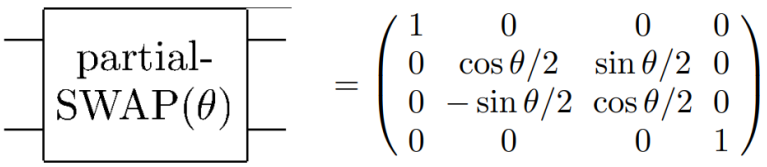
Viene mostrato un esempio (ripreso da [5]) che opera sulla base unaria, ovvero sul sottospazio degli stati che possiedono un solo qubit nello stato $|1\rangle$, questo limita molto gli stati utilizzabili, passando da 2^n a n, ma permette di fare una correzione degli errori migliore, come verrà mostrato ed è di conseguenza meglio applicabile ai sistemi attualmente disponibili, che sono particolarmente rumorosi. Si richiede inoltre meno connettività tra i qubit, minore profondità del circuito rendendo ancora una volta questo approccio più applicabile a sistemi fisici. In aggiunta gli algoritmi di amplitude distribution per circuiti binari sono spesso non risolvibili analiticamente e vengono di conseguenza implementati attraverso reti neurali ([5]), mentre quello mostrato per la base unaria consente di ottenere analiticamente gli operatori che portano il sistema nello stato cercato.

Il circuito presenta i seguenti step:

1. Lo stato iniziale, come negli altri casi è $|0^n\rangle$;
2. Inizialmente si definisce una funzione biunivoca che mappa i valori dell'asset alla base unaria, in modo tale da poter associare ad uno stato della base unaria un valore dell'asset. In questo momento si tratta solo di definire la relazione, che verrà utilizzata in seguito per associare lo stato ottenuto dalla misura ad un valore dell'asset, questo corrisponde alla definizione della funzione v vista nell'esempio precedente.
3. Come prima operazione sul circuito si applica un gate Pauli X al qubit centrale, portandolo da $|0\rangle$ a $|1\rangle$ e ottenendo così un elemento della base unaria (nel caso di

numero pari di qubit, dove non è ben definito un qubit centrale, si sceglie arbitrariamente a quale dei due qubit centrale

4. Utilizzando gate P-swap (ovvero Partial swap o Parametrized swap) fra un qubit e gli adiacenti si costruiscono i corretti coefficienti, partendo dal qubit centrale e procedendo verso gli estremi.
   I gate P-swap applicano una rotazione parziale da un qubit all'altro, nel seguente modo:



$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta/2 & \sin\theta/2 & 0 \\ 0 & -\sin\theta/2 & \cos\theta/2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figura 6: Gate partial-SWAP

Questa operazione è unitaria e lo stato rimane una sovrapposizione di elementi della base unaria, ciò che si ottiene è il trasferimento di parte dell'ampiezza (coefficiente) da un qubit all'adiacente.
Procedendo in questo modo si avrà che l'ampiezza di ogni stato unario dipende da tanti angoli quanto dista dal qubit centrale.
Dal calcolo dello stato ottenuto applicando $n-1$ p-swap si osserva che il set di equazioni che si ottiene per ricavare gli angoli in funzione delle ampiezze desiderate è facilmente risolvibile, si mostra la soluzione unica per il caso a 8 qubit:

$$\theta_7 = \tan^{-1}\left(-\frac{\alpha_0}{\alpha_1}\right), \quad \theta_6 = \tan^{-1}\left(-\frac{\alpha_7}{\alpha_6}\right)$$

$$\theta_5 = \tan^{-1}\left(-\frac{\alpha_1}{\alpha_1}\cos(\theta_7)\right), \quad \theta_4 = \tan^{-1}\left(-\frac{\alpha_6}{\alpha_5}\cos(\theta_6)\right), \quad \theta_3 = \tan^{-1}\left(-\frac{\alpha_2}{\alpha_3}\cos(\theta_5)\right)$$

$$\theta_2 = \tan^{-1}\left(-\frac{\alpha_5}{\alpha_4}\cos(\theta_4)\right), \quad \theta_1 = \tan^{-1}\left(-\frac{\alpha_3}{\alpha_4}\frac{\cos(\theta_2)}{\cos(\theta_3)}\right)$$

Gli angoli verranno quindi definiti in ordine opposto rispetto a quello di applicazione: a partire dagli angoli relativi agli operatori che agiscono sui qubit agli estremi del registro ci si muove poi verso l'interno fino ad arrivare al primo che agisce sul qubit centrale.
Il circuito appena descritto è mostrato in figura e rappresenta la parte centrale dell'algoritmo unario che verrà utilizzato per la valutazione dell'opzione europea.

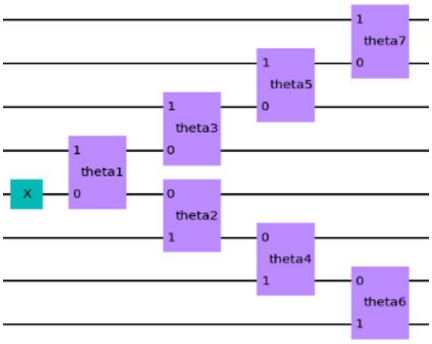


Figura 7: Circuito di amplitude distribution per la base unaria

Con questa definizione degli angoli lo stato finale che si ottiene è quello cercato, ovvero

$$|\psi\rangle = \sum_{i=0}^{2^{n-1}} \alpha_i \, |i\rangle$$

Al termine di questo processo si applica la seconda parte del circuito, che consente di trasferire sul qubit di appoggio la probabilità cercata, ovvero il valore atteso del payoff.
Si utilizzano dei gates due qubit, in particolare delle rotazioni controllate, a coppie di qubit che prevedono quello di appoggio e ciascuno dei qubit che codificano un prezzo maggiore dello strike price, ovvero i valori del sottostante per i quali il payoff non è nullo.

Queste rotazioni traferiscono sul qubit di appoggio un'ampiezza proporzionale alla differenza tra il prezzo che il qubit codifica e lo strike price:

$$\phi_i = 2 \, arcsin\left(\sqrt{\frac{S_i - K}{S_{max} - K}}\right)$$

Si introduce il denominatore come normalizzazione. In seguito a questa operazione lo stato del sistema è il seguente:

$$|\psi\rangle = \sum_{S_i \leq K} \alpha_i |i\rangle \, |0\rangle + \sum_{S_i > K} \alpha_i \left(\cos\left(\frac{\phi_i}{2}\right)\right) |i\rangle |0\rangle + \sum_{S_i > K} \alpha_i \sqrt{\frac{S_i - K}{S_{max} - K}} \; |i\rangle |1\rangle$$

Si mostrano separatamente i ket che corrispondono ai qubit che codificano la discretizzazione del sottostante seguiti dal qubit di appoggio, in modo da evidenziare che la probabilità di trovare il qubit di appoggio nello stato $|1\rangle$ sia esattamente:

$$P(|1\rangle) = \sum_{S_i > K} \alpha_i \frac{S_i - K}{S_{max} - K}$$

Dalla quale si ricava direttamente il valore atteso del payoff:

$$V = P(|1\rangle) \, (S_{max} - K)$$

L'utilità della computazione quantistica entra in gioco per la possibilità di stimare questa probabilità in modo più efficiente rispetto al calcolo della frequenza su numerose misure.

Per fare ciò si ricorre all'utilizzo di algoritmi di Amplitude Estimation.

4.3Amplitude Estimation

L'algoritmo di amplitude Estimation ha l'obettivo di stimare l'ampiezza di un determinato stato quantistico.
Si definisce inizialmente un operatore A che a partire da uno stato iniziale con un numero arbitrario di qubit, lo porta in:

$$A|0\rangle = \sqrt{1-a}|\psi_0\rangle|0\rangle + \sqrt{a}|\psi_1\rangle|1\rangle \quad (9)$$

Dove $|\psi_i\rangle$ rappresentano due stati non necessariamente ortogonali dei primi n-1 qubit, mentre il secondo ket si riferisce allo stato dell'ultimo qubit, che viene chiamato qubit di appoggio.

Si cerca la probabilità di misurare 1 sul qubit di appoggio, ovvero la probabilità misurando l'osservabile di riferimento sul qubit di appoggio si ottengano gli autovalori relativi all'autostato indicato con $|1\rangle$. Il processo di Amplitude Estimation si basa sull'algoritmo di Grover ed utilizza due operatori: lo stesso operatore A e un operatore Q definito nel seguente modo:

$$Q = -AS_0A^{\dagger}S_{\psi_0}$$

La definizione fa uso dell'operatore A, il suo inverso $A^{\dagger}$ e due operatori di riflessione: S_0, che cambia segno al coefficiente dello stato $|0\rangle_n|0\rangle$, e l'operatore S_{ψ_0}, che cambia segno al coefficiente dello stato $|\psi_0\rangle$:

$$S_0 = I - 2|0\rangle_{n+1}\langle 0|_{n+1}$$

$$S_{\psi_0} = I - 2|\psi_0\rangle|0\rangle\langle 0|\langle\psi_0|$$

Questi operatori vengono chiamati di riflessione in quanto hanno la particolarità di agire soltanto tramite un cambio di segno di certe componenti dello stato.
Per quanto riguarda il secondo operatore si mostrano il ket ed il bra che agiscono sul qubit di appoggio separatamente dal resto dei qubit, l'operazione che viene effettuata è la seguente: a partire da uno stato generico $|\phi\rangle|\lambda\rangle$, dove ϕ indica lo stato dei primi n qubit e λ quello del qubit di appoggio si applica l'operatore (ricordando l'interpretazione dei termini $\langle bra|ket\rangle$ come prodotto scalare):

$$S_{\psi_0}|\phi\rangle|\lambda\rangle = (I - 2|\psi_0\rangle|0\rangle\langle 0|\langle\psi_0|)|\phi\rangle|\lambda\rangle = |\phi\rangle|\lambda\rangle - 2(\langle\psi_0|\phi\rangle\langle 0|\lambda\rangle)|\psi_0\rangle|0\rangle$$

Il primo operatore risulta un caso particolare del secondo, che presenta come stato $|\psi_0\rangle$ lo stato iniziale $|0^{\otimes n}\rangle$.

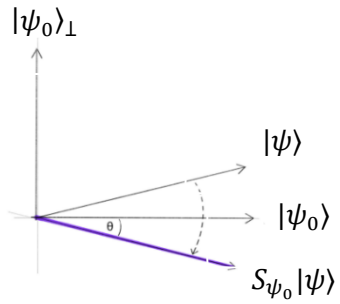


Figura 8: Rappresentazione grafica della riflessione rispetto allo stato $|\psi_0\rangle$

L' algoritmo che verrà implementato applicherà combinazioni di questi operatori, in particolare potenze dell'operatore di Grover Q, in modo iterativo fino al raggiungimento di una accuratezza desiderata.

4.4 Correzione degli errori

Un forte punto a favore dell'utilizzo della base unaria, oltre alla implementazione più semplice, è quello di rendere possibile una correzione degli errori semplice ma efficace. I sistemi fisici a disposizione attualmente sono caratterizzati da frequenti errori di bit-flip o phase-flip: ovvero errori in cui lo stato di un qubit, oppure la sua fase vengono invertiti. Questi errori possono essere limitati considerando soltanto un sottospazio dello spazio di Hilbert a disposizione, come nel nostro caso di circuito unario. In seguito alla misura è infatti facile verificare se lo stato misurato appartiene oppure no alla base unaria, misurando un numero di qubit nello stato $|1\rangle$ diverso da 1 si è certi di essere in presenza di un errore di bit-flip, in cui uno dei qubit ha cambiato il proprio stato. È possibile quindi rigettare tale misura in quanto affetta da errore.

Questo, tuttavia, non riguarda le simulazioni svolte in quanto caratterizzate da assenza di rumore, ma questa correzione può diventare utile applicando questo tipo di algoritmi a sistemi reali.

5. IBM Quantum Lab

L'implementazione degli algoritmi è stata svolta attraverso IBM Quantum, che mette a disposizione un ambiente nel quale è possibile programmare circuiti quantistici ed eseguirli su simulatori e sistemi quantistici reali.

Il linguaggio utilizzato per simulare i sistemi è python e in particolare si fa utilizzo del pacchetto Qiskit.
In questo ambiente è possibile definire i circuiti quantistici in termini dei loro gates, avendo a disposizione comandi già pronti per i gate più comuni e anche la disponibilità di definire gates personalizzati attraverso la rappresentazione matriciale del relativo operatore.

Il pacchetto mette a disposizione backend diverse, ovvero funzioni che forniscono informazioni differenti riguardo l'applicazione di un determinato circuito
Le backend che sono state utlizzate sono le seguenti:

1. statevector_simulator: consente di simulare il circuito e restituisce lo stato finale di ogni qubit;
2. unitary_simulator: restituisce una matrice $2^n \times 2^n$ che rappresenta l'operatore unitario equivalente all'intero circuito;
3. qasm_simulator: consente di simulare la misura di uno o più qubit e ripetere l'operazione un numero arbitrario di volte, è ciò che fornisce i risultati delle simulazioni, che una volta mediati permettono di calcolare il valore cercato;

Questo per quanto riguarda soltanto sistemi quantistici simulati, ma il portale offre anche la disponibilità di eseguire circuiti e misure anche su sistemi quantistici reali. Quelli accessibili in modo gratuito però presentano un numero limitato di qubit (massimo 5) con una connettività limitata: ogni qubit è collegato al massimo ad altri 2 qubit.

5.1 Monte Carlo classico

Si mostrano brevemente i risultati ottenuti attraverso un algoritmo Monte Carlo classico, in modo da poterli confrontare con i successivi risultati:

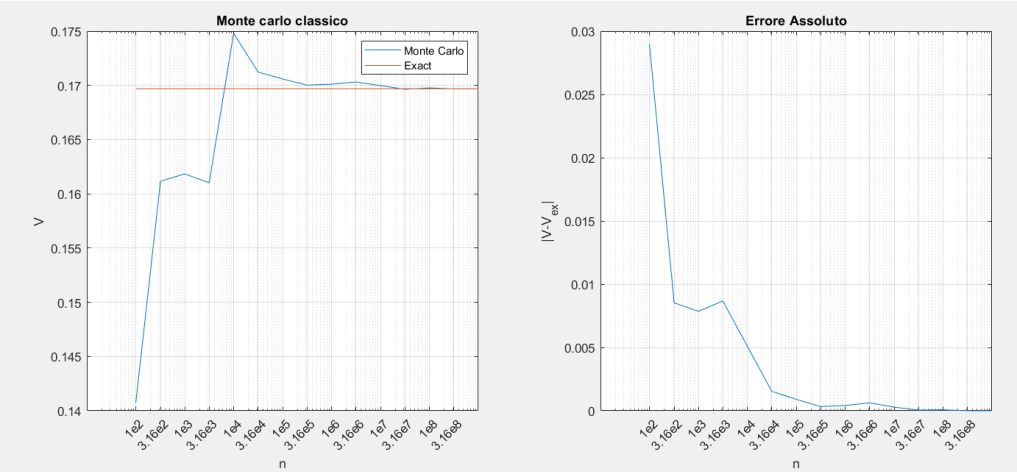


Figura 9: Risultati numerici per un algoritmo Monte Carlo classico

Attraverso l'equazione differenziale stocastica si estrae il prezzo dell'asset alla scadenza n volte, se ne calcola il payoff e si esegue la media su tutti i valori ottenuti.

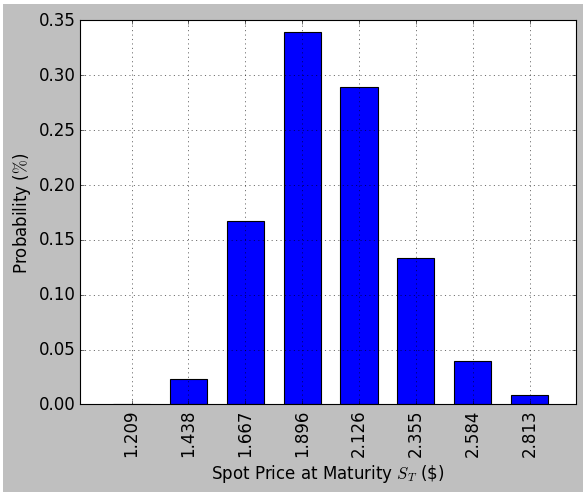
5.2 Circuito unario

Per implementare il circuito unario si procede come descritto in precedenza, il codice è riportato in appendice, si mostrano i passaggi fondamentali:

```
#parametri finanziari
S0 = 2
sigma = 0.4
r = 0.05
T = 40/365
K=1.896


#parametri della distribuzione
mu = (r - 0.5 * sigma**2) * T + log(S0)
sigmad = sigma * sqrt(T)
mean = exp(mu + sigmad**2 / 2)
variance = (exp(sigmad**2) - 1) * exp(2 * mu + sigmad**2)
stddev = sqrt(variance)


low = maximum(0, mean - 3 * stddev)
high = mean + 3 * stddev

#generazione della distribuzione
Distr = LogNormalDistribution(
    3, mu=mu, sigma=sigmad**2, bounds=(low, high)
)
```

Inizialmente si definiscono i parametri finanziari del contratto: valore iniziale, tasso di interesse, durata del contratto, volatilità e strike price. Da questi si ottengono i parametri della distribuzione, che viene generata attraverso la funzione $LogNormalDistribution$, i valori che vengono restituiti dalla funzione, all' interno di Distr.probabilities, verranno utilizzati per definire le ampiezze degli stati unari del sistema.

Si è scelto di codificare valori dell'asset all' interno di un range centrato nella media della distribuzione e di larghezza pari a 3 deviazioni standard, in modo tale da non trascurare una porzione troppo grande della distribuzione ma allo stesso tempo non occupare stati della base unaria codificando valori la cui probabilità è talmente ridotta da risultare in conteggi nulli.



La distribuzione che si ottiene è la seguente, il grafico mostra le probabilità in funzione dei valori discretizzati del sottostante, che sono in numero uguale al numero di bit utilizzato.

```
a0=sqrt(Distr.probabilities[0])
a1=sqrt(Distr.probabilities[1])
a2=sqrt(Distr.probabilities[2])
a3=sqrt(Distr.probabilities[3])
a4=sqrt(Distr.probabilities[4])
a5=sqrt(Distr.probabilities[5])
a6=sqrt(Distr.probabilities[6])
a7=sqrt(Distr.probabilities[7])

theta7=arctan(a0/a1);
theta6=arctan(a7/a6);
theta5=arctan(a1/(a2*cos(theta7)));
theta4=arctan(a6/(a5*cos(theta6)));
theta3=arctan(a2/(a3*cos(theta5)));
theta2=arctan(a5/(a4*cos(theta4)));
theta1=arctan((a3*cos(theta2))/(a4*cos(theta3)));
```

Vengono definite le ampiezze degli stati attraverso la radice delle probabilità, e sulla base di queste si definiscono gli angoli dei p-swap che verranno applicati al circuito.

```
p_swap1=qi.Operator([[1, 0, 0, 0],
                     [0, cos(theta1), sin(theta1), 0],
                     [0, -sin(theta1), cos(theta1), 0],
                     [0, 0, 0, 1]])
```

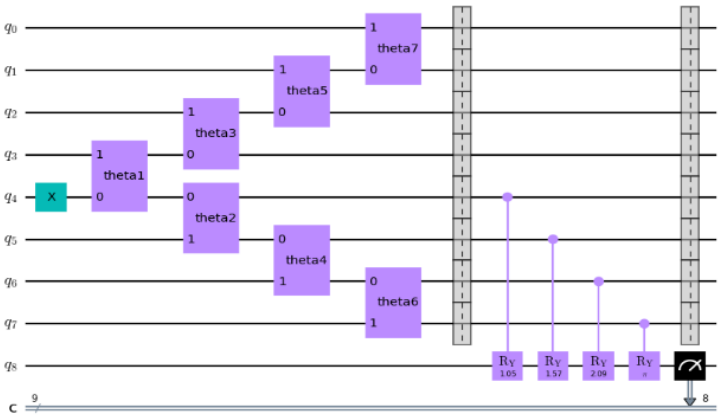
Il gate p-swap non è presente fra quelli predefiniti dal pacchetto qiskit, di conseguenza è necessario definirlo in forma matriciale in funzione di ogni angolo θ_i

```
meas = QuantumCircuit(9,9)
meas.x(4)

meas.unitary(p_swap1, [4,3],label='theta1')
meas.unitary(p_swap2, [4,5],label='theta2')
meas.unitary(p_swap3, [3,2],label='theta3')
meas.unitary(p_swap4, [5,6],label='theta4')
meas.unitary(p_swap5, [2,1],label='theta5')
meas.unitary(p_swap6, [6,7],label='theta6')
meas.unitary(p_swap7, [1,0],label='theta7')


meas.barrier(range(8))

meas.cry(2*arcsin(sqrt((Distr.values[4]-K)/(high-K))),4,8)
meas.cry(2*arcsin(sqrt((Distr.values[5]-K)/(high-K))),5,8)
meas.cry(2*arcsin(sqrt((Distr.values[6]-K)/(high-K))),6,8)
meas.cry(2*arcsin(sqrt((Distr.values[7]-K)/(high-K))),7,8)
```

Vengono applicati nel corretto ordine i gate, a partire dal Pauli-X sul qubit 4, i 7 p-swap e successivamente le rotazioni controllate (“cry”) sui qubit che codificano un prezzo maggiore dello strike price.

Viene inserita anche una barriera fra i p-swap e le rotazioni per separare visivamente le due parti del circuito.

Il circuito che si ottiene è il seguente



Si utilizza il quibit 8 come qubit di appoggio, la cui ampiezza al termine del circuito rappresenta la radice del valore di aspettazione del payoff normalizzato, dal quale è immediato calcolare:

$$V = P(|1\rangle) \cdot (S_{max} - K)$$

```
backend_sim = Aer.get_backend('qasm_simulator')

nshots=100
job_sim = backend_sim.run(transpile(qc, backend_sim), shots=nshots)


result_sim = job_sim.result()
counts = result_sim.get_counts(qc)

from qiskit.visualization import plot_histogram

plot_histogram(counts)
```

Attraverso "$qasm\_simulator$" si sceglie infine quante volte ripetere la misura.

I risultati vengono restituiti all'interno di un vettore chiamato $counts$

In prima istanza si calcola il payoff misurando i primi 8 qubit: per definire la misura è sufficiente il comando "meas.measure(range(8),range(8))", specificando due vettori, il primo dei quali indica su quali qubit eseguire la misura, il secondo indica in quale posizione del registro classico c inserire la misura di ciascun qubit.
In questo caso si usa range(8) che restituisce un vettore con i primi 8 interi a partire da 0, la misura è condotta quindi sui qubit da 0 a 7. Stessa cosa vale per il secondo vettore, si inseriscono quindi i risultati della misura nello stesso ordine all'interno del registro classico, per fare in modo che al primo bit classico del registro c corrisponda la misura del primo qubit e così via.

Inizialmente, quindi, la misura viene eseguita senza applicare le rotazioni controllate che presenta la seconda parte del circuito. Il risultato della misura è un vettore della base unaria, al quale è associato un valore dell'asset. Si procede ripetendo la misura un numero elevato di volte tenendo traccia della frequenza di ogni risultato.
Le frequenze, al crescere del numero di misure approssimano sempre meglio la probabilità di misurare ogni stato, di conseguenza la sua ampiezza e quindi la distribuzione calcolata inizialmente.
I risultati ottenuti sono i seguenti:

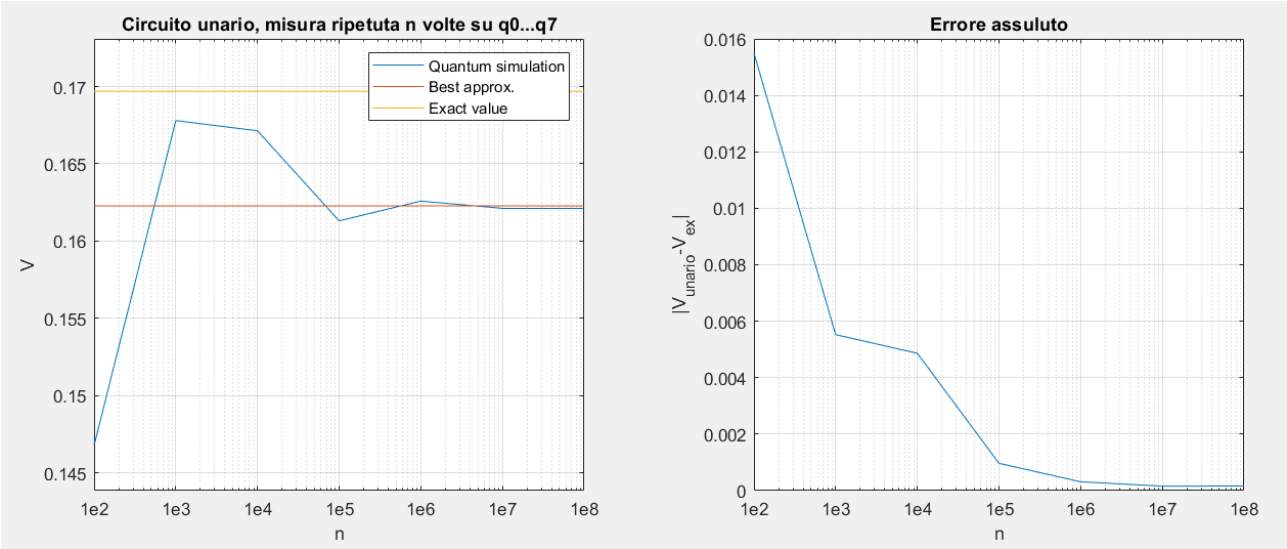


Figura 10: valutazione dell'opzione Call eseguendo la misura sui qubit da 0 a 7 del circuito unario

Vengono indicati i due valori "*Exact value*" e "*Best approx*": il primo rappresenta appunto il vaore esatto ottenuto risolvendo l'equazione di Black-Scholes, mentre il secondo è la migliore approssimazione possibile considerando la discretizzazione sui qubit a disposizione. È quindi ragionevole che il risultato si avvicini all'approssimazione rispetto che al valore esatto, per questo motivo nel grafico di destra si mostra come errore il valore assoluto della differenza tra il valore ottenuto tramite la misura ed il valore "*Best approx*".

Questo risultato tuttavia è da intendere soltanto come una conferma della corretta definizione del circuito, in particolare della parte di amplitude distribution: i primi 8 qubit vengono infatti preparati "caricando" la distribuzione corretta, ottenuta dalla soluzione dell'equazione differenziale stocastica. La convergenza del metodo è quindi conseguenza del fatto che aumentando il numero di misure (shots) della simulazione la frequenza misurata approssima sempre meglio la probabilità esatta determinata dalla preparazione del circuito, in modo del tutto analogo ad un Monte Carlo classico.

Si procede poi applicando la seconda parte del circuito, ovvero le rotazioni controllate: dopo l'applicazione di questi 4 gates si arriva allo stato cercato per il qubit di appoggio, che per come è stato implementato il circuito è il qubit 8.

Lo stato del qubit di appoggio è il seguente $|\psi\rangle = \sqrt{1-a}|0\rangle + \sqrt{a}|1\rangle$, dove per come è stato definito il circuito $a = \frac{V_{call}}{S_{max}-K}$, come detto in precedenza quindi misurando la probabilità di ottenere 1 dalla misura del qubit si può ricavare in modo diretto il valore del'opzione.

È possibile, come nella misura precedente, procedere misurando direttamente il qubit 8 e ripetere questa misura più volte. Il risultato che si ottiene è teoricamente del tutto analogo al precedente, i quanto non viene applicato nessun algoritmo di Amplitude Estimation e di conseguenza non ci si aspettano differenze rispetto al Monte Carlo classico:

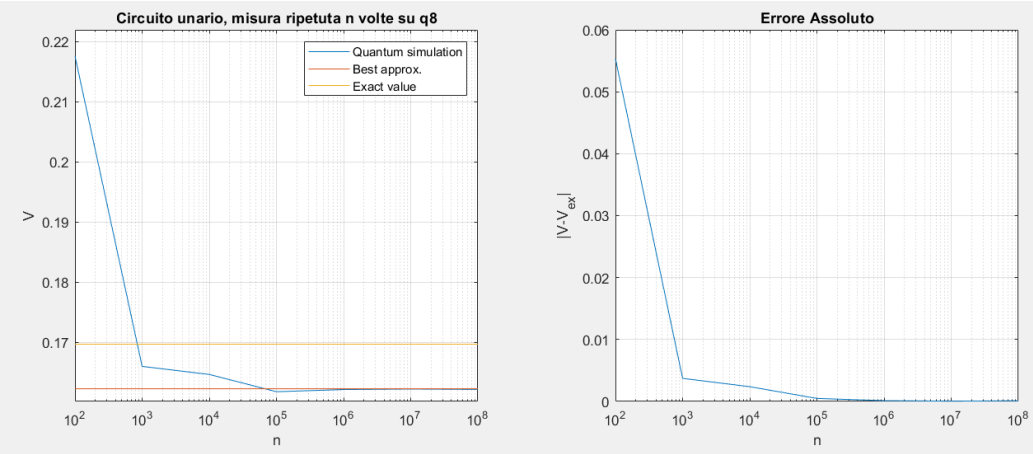


Figura 11:valutazione dell'opzione Call eseguendo la misura sul qubit q8 al termine del circuito unario

A questo punto l'obiettivo è quello di applicare l'algoritmo di Amplitude Estimation allo stato finale del qubit 8. Il pacchetto Qiskit comprende una funzione built-in chiamata "$IterativeAmplitudeEstimation$", che può essere lanciata conoscendo gli operatori A e Q introdotti nel paragrafo dedicato all'amplitude estimation.

Tuttavia la definizione dei due operatori richiesti non è possibile se non essendo a conoscenza dell'ampiezza dello stato, i due operatori infatti nella documentazione del pacchetto qiskit vengono definiti nel seguente modo:

$$A = Ry(2\arcsin(a)) \qquad Q = Ry(4\arcsin(a))$$

Sono quindi rotazioni rispetto all'asse y di un angolo ottenuto direttamente dall'ampiezza a che si vuole stimare. Di conseguenza l'applicazione di questo algoritmo non risulta utile alla stima dell'ampiezza cercata, che si suppone ignota.

Come ultima conferma è possibile verificare che all'aumentare del numero di qubit il valore ottenuto tramite la misura sul qubit finale approssima via via meglio il valore esatto, è stato quindi modificato il codice in modo da renderlo adattivo rispetto al numero di qubit:

```
mu = (r - 0.5 * sigma**2) * T + log(S0)
sigmad = sigma * sqrt(T)
mean = exp(mu + sigmad**2 / 2)
variance = (exp(sigmad**2) - 1) * exp(2 * mu + sigmad**2)
stddev = sqrt(variance)
n=3 #log 2 del numero di qubit

low = maximum(0, mean - 3 * stddev)
high = mean + 3 * stddev
```

Viene definita la variabile n come logaritmo in base 2 del numero di qubit, in modo tale da raddoppiare il numero di qubit incrementando n

Vengono implementati cicli per definire le ampiezze α_i, gli angoli θ_i dei gate p-swap ed infine i p-swap stessi.

```
a=zeros(2**n)
thetas = zeros(2**n-1)
#Definiizone delle ampiezze sulla base della distribuzione cercata
for i in range(len(a)):
    a[i]=sqrt(Distr.probabilities[i])

#Definizione degli angoli agli estremi
thetas[-1]=arctan(a[0]/a[1])
thetas[-2]=arctan(a[-1]/a[-2])

#Definizione degli angoli intermedi
for i in range(int(len(a)/2)-2):
    thetas[-3-i*2]=arctan(a[i+1]/(a[i+2]*cos(thetas[2**n-2*(i+1)])))
    thetas[-4-i*2]=arctan(a[-i-2]/(a[-i-3]*cos(thetas[2**n-1-2*(i+1)])))

#definizione dell'angolo centrale (theta1)
thetas[0]=arctan((a[2**(n-1)-1]*cos(thetas[1]))/(a[2**(n-1)]*cos(thetas[2])))

#Definizione dei gates p-swap
for i in range(len(a)-1):
    string = "p_swap"+str(i+1)+"=qi.Operator([[1, 0, 0, 0],[0, cos(thetas["+str(i)+"]), \
    sin(thetas["+str(i)+"]), 0],[0, -sin(thetas["+str(i)+"]), cos(thetas["+str(i)+"]), 0],[0, 0, 0, 1]])"
    exec(string)
```

I gates così definiti vengono applicati al circuito.

```
#Applicazione dei gates p-swap al circuito
string = "meas.unitary(p_swap1, [2**(n-1),2**(n-1)-1],label='theta1')"
exec(string)
for i in range(2**(n-1)-1):
    string1 = "meas.unitary(p_swap"+str(3+2*i)+", ["+str(2**(n-1)-1-i)+","+str(2**(n-1)-2-i)+"]\
    ,label='theta"+str(3+2*i)+"')"
    string2 = "meas.unitary(p_swap"+str(2+2*i)+", ["+str(2**(n-1)+i)+","+str(2**(n-1)+i+1)+"],\
    label='theta"+str(2+2*i)+"')"
    exec(string1)
    exec(string2)
```

Successivamente si ricerca l'indice del primo qubit che codifica un prezzo maggiore dello strike price, che sarà il primo a partire dal quale verranno applicate le rotazioni controllate

```
meas.barrier(range(2**n))
#Ricerca del qubit che codifica il primo prezzo maggiore dello strike price
ik=0;
for i in range(len(a)):
    if Distr.values[i]>K:
        ik=i+1
        break
#Applicazione delle rotazioni controllate (CRY)
for i in range(ik,len(a)):
    string = "meas.cry(2*arcsin(sqrt((Distr.values["+str(i)+"]-K)/(high-K))),"+str(i)+","+str(2**n)+")"
    exec(string)
```

Infine si implementa la misura del qubit di appoggio: il qubit di appoggio è l'ultimo, ovvero quello identificato dall'indice 2^n.

```
meas.barrier(range(2**n))
circ.add_register(meas.cregs[0])
#misura del qubit di appoggio
meas.measure(2**n,0)
```

La seconda parte del circuito, nella quale si definisce il numero di misure e si calcola il valore atteso del payoff a partire dalla probabilità di misurare il qubit di appoggio nello stato $|1\rangle$ non è stata modificata rispetto al codice mostrato precedentemente.

Si mostrano due esempi del circuito ottenuto al variare del numero di qubit:

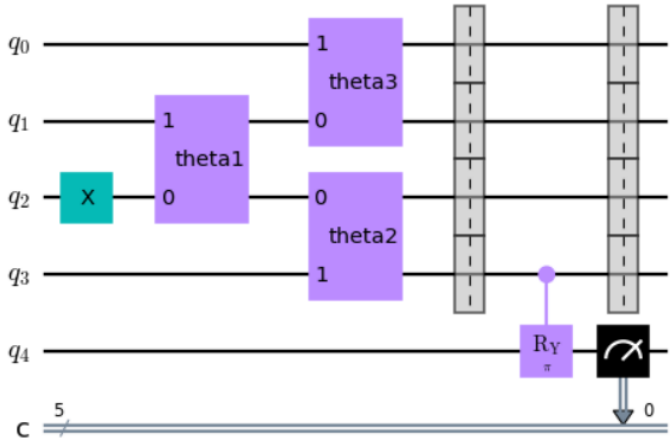


Figura 12: Circuito unario per 4 qubit

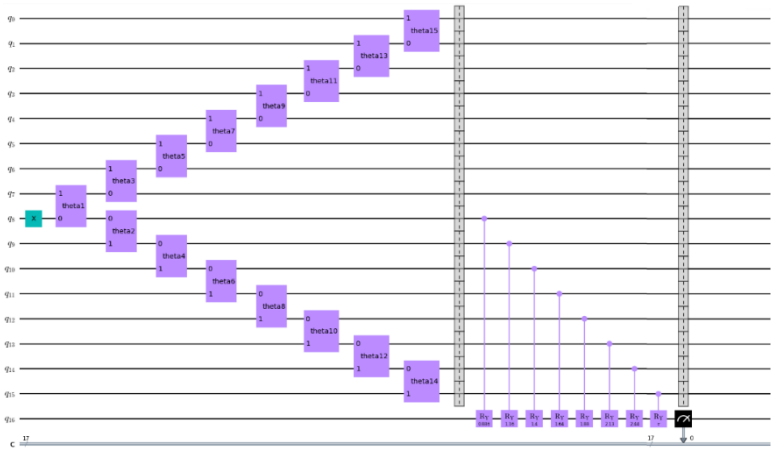


Figura 13: Circuito unario per 16 qubit

Si mostrano i risultati ottenuti:

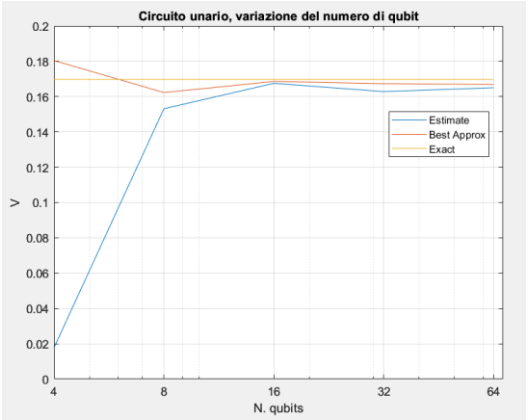


Figura 14: Valutazione dell'opzione Call tramite algoritmo unario al variare del numero di qubit (numero di misure $nshots = 10^4$)

Si osserva come, ragionevolmente, sia il valore best approx che la stima ottenuta tramite le misure convergano verso il valore esatto. Tuttavia il costo computazionale, sia in termini di tempo di esecuzione che di memoria occupata cresce esponenzialmente insieme al numero di quibit, 8 qubit rappresenta un buon compromesso per avere una discretizzazione sufficiente e allo stesso tempo essere in grado di aumentare il numero massimo di misure effettuate in tempi contenuti.

5.3 IBM - European Call Option Pricing

Si mostrano ora i risultati ottenuti con il programma "European Call Option Pricing" già presente all'interno del pacchetto fornito da IBM: esso risolve la valutazione di un' opzione europea di tipo call con gli stessi parametri finanziari utilizzati anche nel circuito unario.
Il numero di qubit utilizzati di default dal programma è 3, motivo per il quale il circuito unario descritto in precedenza è stato realizzato con 8 qubit, in modo tale che in entrambi i casi la discretizzazione del valore del sottostante fosse realizzata dagli stessi 8 valori. Si utilizzano infatti tutti i 2^n stati in questo caso, a contrario del circuito unario dove gli stati utilizzati sono soltanto n.

Il circuito prepara, come nel caso precedente un qubit di appoggio nello stato

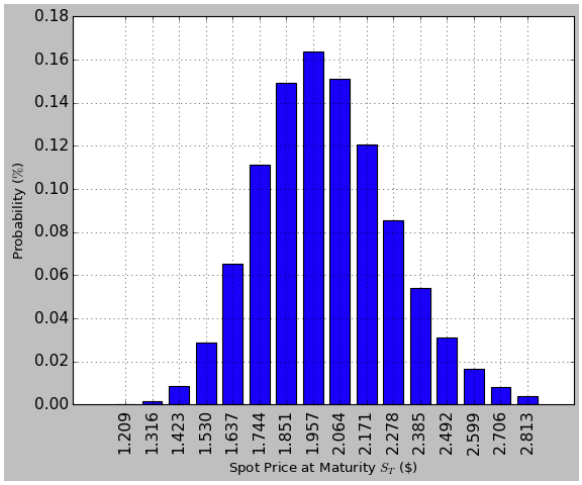
$$|\Psi\rangle = \sqrt{1-a}|\psi_0\rangle + \sqrt{a}|\psi_1\rangle$$

Al quale applica in seguito la funzione *"$Iterative\ AmplitudeEstimation$"*, per fornire il valore stimato ed il relativo intervallo di confidenza del 95%.

Siccome si tratta di una prima indagine è stato seguito un approccio "black box" non si entra quindi nel dettaglio dell'esatto circuito implementato. Si mostrano i passaggi fondamentali del codice:

```
# number of qubits to represent the uncertainty
num_uncertainty_qubits = 3

# parameters for considered random distribution
S = 2.0  # initial spot price
vol = 0.4  # volatility of 40%
r = 0.05  # annual interest rate of 4%
T = 40 / 365  # 40 days to maturity

# resulting parameters for log-normal distribution
mu = (r - 0.5 * vol**2) * T + np.log(S)
sigma = vol * np.sqrt(T)
mean = np.exp(mu + sigma**2 / 2)
variance = (np.exp(sigma**2) - 1) * np.exp(2 * mu + sigma**2)
stddev = np.sqrt(variance)

# lowest and highest value considered for the spot price; in between, an equidistant discretization is considered.
low = np.maximum(0, mean - 3 * stddev)
high = mean + 3 * stddev

# construct A operator for QAE for the payoff function by
# composing the uncertainty model and the objective
uncertainty_model = LogNormalDistribution(
    num_uncertainty_qubits, mu=mu, sigma=sigma**2, bounds=(low, high)
)
```

Analogamente al circuito unario si definiscono i parametri finanziari e si genera la distribuzione, in questo caso sulla base del numero di qubit scelto nel primo comando.



Si mostra il caso con 4 qubit, la discretizzazione, a contrario del caso precedente prevede 2^n possibili valori per l'asset.

```
qi = QuantumInstance(Aer.get_backend("aer_simulator"), shots=100)
problem = EstimationProblem(
    state_preparation=european_call,
    objective_qubits=[3],
    post_processing=european_call_objective.post_processing,
)
# construct amplitude estimation
ae = IterativeAmplitudeEstimation(epsilon, alpha=alpha, quantum_instance=qi)
```

Si definisce il tipo di problema (Amplitude Estimation) e il numero di misure da effettuare

A questo punto il circuito viene eseguito tramite il comando "$result = ae.estimate(problem)$", che restituisce il valore approssimato e il relativo intervallo di confidenza.

Si mostrano in seguito i risultati ottenuti al variare del numero di misure, mantenendo 3 qubit in modo da avere la stessa discretizzazione che si ottiene per 8 qubit nel circuito unario:

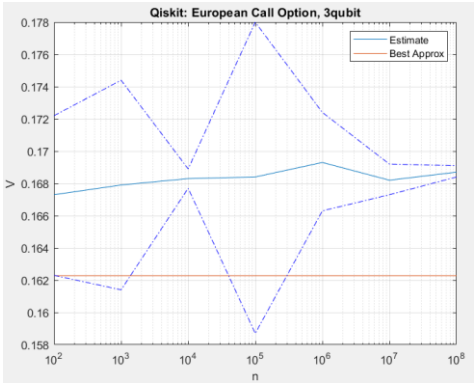


Figura 15: Risultati ottenuti tramite il programma "European Call Option Pricing", mantenedo 3 qubit e variando il numero di misure(n)

Si osservano diversi dettagli:

- Non è presente una convergenza al valore cercato
- Forte inconsistenza nell'intervallo di confidenza: per i primi passi segue un andamento casuale per poi iniziare a stringersi attorno al valore medio calcolato
- In generale si osserva una sottostima dell'intervallo di confidenza che di conseguenza spesso non si sovrappone al valore cercate ($Best\ Approx$)

Si è passati in seguito ad aumentare il numero di qubit utilizzati, con l'intenzione di migliorare la discretizzazione e di conseguenza la stima:

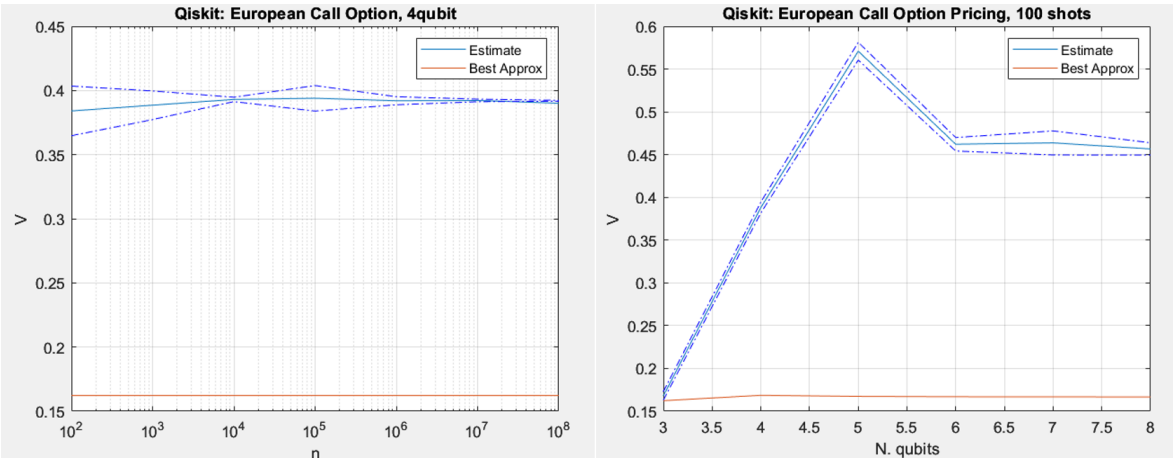


Figura 16: Risultati ottenuti tramite il programma "European Call Option Pricing", per 3 qubit al variare del numero di misure (sinistra) e per 100 misure al variare del numero di qubit (destra)

Il risultato che si ottiene al variare dei qubit sembra stabilizzarsi su un risultato differente da quello atteso. Si sottolinea che la migliore stima ($best\ approx$) varia al variare del numero di qubit: la discretizzazzione prevede via via sempre più punti, di conseguenza migliora la stima che può essere ottenuta, tuttavia questa variazione non è apprezzabile a livello grafico, poiché su una scala molto inferiore rispetto all'errore commesso dall'algoritmo quantistico.

Anche in questo caso infine l'intervallo di confidenza è fortemente sottostimato.

5.4 IBM – Funzione "european_call_pricing"

Si è proceduto poi al test della funzione "european_call_pricing", anch'essa presente nel pacchetto Qiskit finance.
Questa funzione prendendo in input la distribuzione generata inizialmente e lo strike price genera il circuito di amplitude distribution, al quale si applica in seguito la Estimation nello stesso modo del caso precedente:

```
from qiskit_finance.applications.estimation import EuropeanCallPricing

european_call_pricing = EuropeanCallPricing(
    num_state_qubits=num_uncertainty_qubits,
    strike_price=strike_price,
    rescaling_factor=c_approx,
    bounds=(low, high),
    uncertainty_model=uncertainty_model,
)

qi = QuantumInstance(Aer.get_backend("aer_simulator"), shots=100)
problem = european_call_pricing.to_estimation_problem()
# construct amplitude estimation
ae = IterativeAmplitudeEstimation(epsilon, alpha=alpha, quantum_instance=qi)
result = ae.estimate(problem)

conf_int = np.array(result.confidence_interval_processed)
```

Si procede come nel caso precedente, quindi analizzando come prima cosa la stima al variare del numero delle misure:

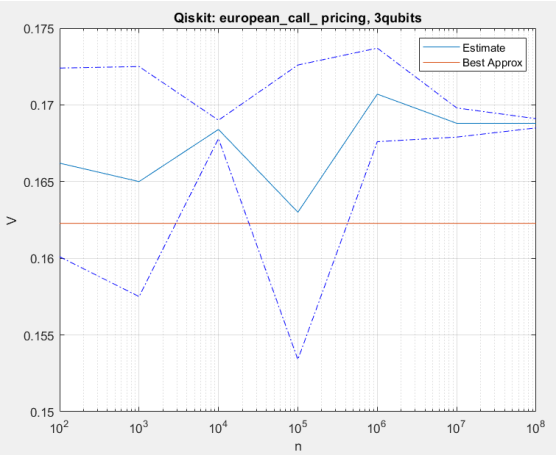


Figura 17: Risultati ottenuti tramite la funzione "european_call_pricing", mantenendo 3 qubit al variare del numero di misure (n)

Come nel caso precedente non si osserva alcuna tendenza ad approssimare progressivamente meglio il risultato esatto.

In seguito, come nel caso precedente si mostrano i risultati ottenuti al variare del numero di qubits:

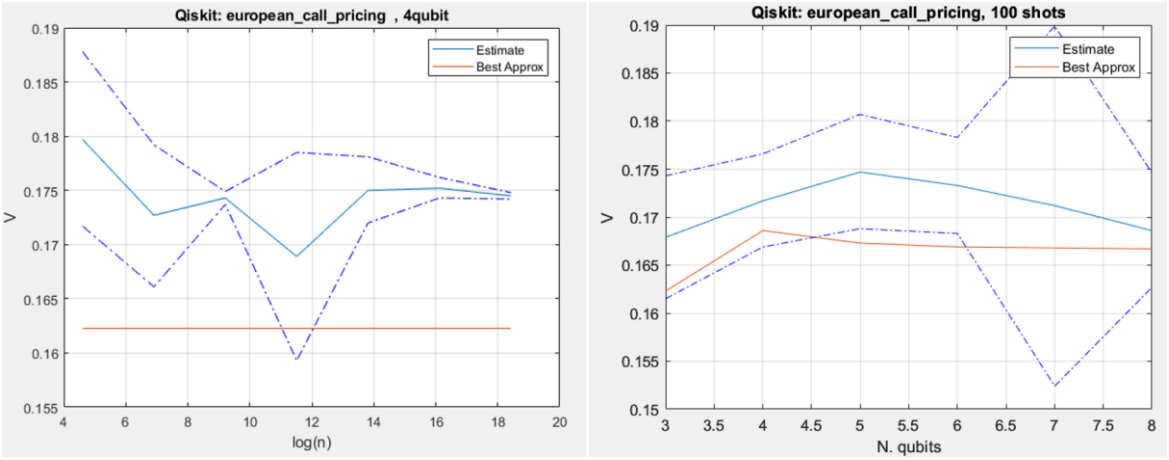


Figura 18:Risultati ottenuti tramite il programma "European Call Option Pricing", per 3 qubit al variare del numero di misure (sinistra) e per 100 misure al variare del numero di qubit (destra)

A contrario del caso precedente, utilizzando questa funzione la stima sembra migliorare all'aumentare del numero di qubit e il valore esatto ricade spesso all'interno dell'intervallo di confidenza.

Si conduce quindi un'indagine finale fissando a 8 il numero di qubit e valutando l'andamento al crescere del numero di misure:

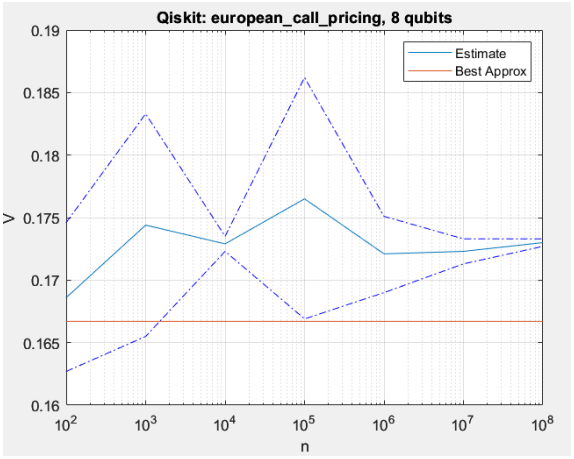


Figura 19:Risultati ottenuti tramite la funzione "european_call_pricing", mantenendo 8 qubit al variare del numero di misure (n)

Non si osserva, come per numeri di qubit inferiori, una tendenza a migliorare l'approssimazione al crescere del numero di misure e in aggiunta il risultato è del tutto paragonabile a quello ottenuto con soltanto 4 qubit. Non è quindi confermato un guadagno in accuratezza aumentando il numero di qubit.

5.5 Verifica della put-call parity

In questa sezione si confrontano i valori ottenuti in precedenza attraverso lo script European Call Option Pricing con quelli ottenuti dall' analogo European Put Option Pricing, nel quale l'unica differenza consiste nella modifica del payoff.

Si indaga sulla consistenza dei risultati rispetto alla put-call parity, mostrata in (5)

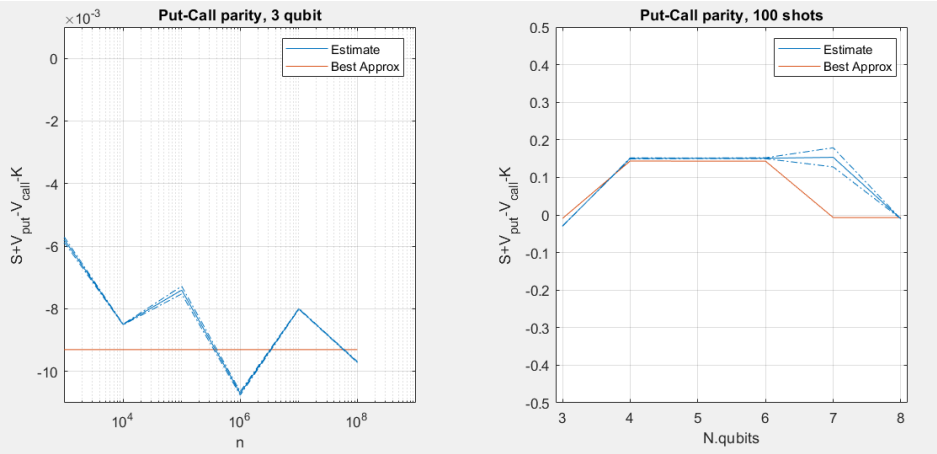


Figura 20: Verifica della put-call parity per 3 qubit al variare del numero di misure (sinistra) e per 100 misure al variare del numero di qubit (destra)

Anche in questo caso si nota come l'intervallo di confidenza per la maggior parte delle misure non si sovrapponga al valore $best\ approx$, che ha lo stesso significato dei casi precedenti. Tuttavia si osserva che al variare del numero di qubit il valore ottenuto rimane paragonabile alla migliore stima e ne segue l' andamento, al contrario di ciò che succedeva considerando solamente il valore dell'opizone call, il quale sembrava divergere dal valore $best\ approx$.

Sotto questo aspetto si può concludere che questi algoritmi siano consistenti dal punto di vista della parity.

5.6 Modifica dei parametri finanziari

Si conclude con un' ultima indagine modificando i parametri finanziari, siccome per tutti i casi precedenti sono stati utilizzati quelli preimpostati dal programma "$European\ call\ option\ pricing$".
Si pongono $S_0 = 11, K = 10, \sigma = 0.3, r = 0.06, T = 1$ e si ripetono le misure mostrate in precedenza:

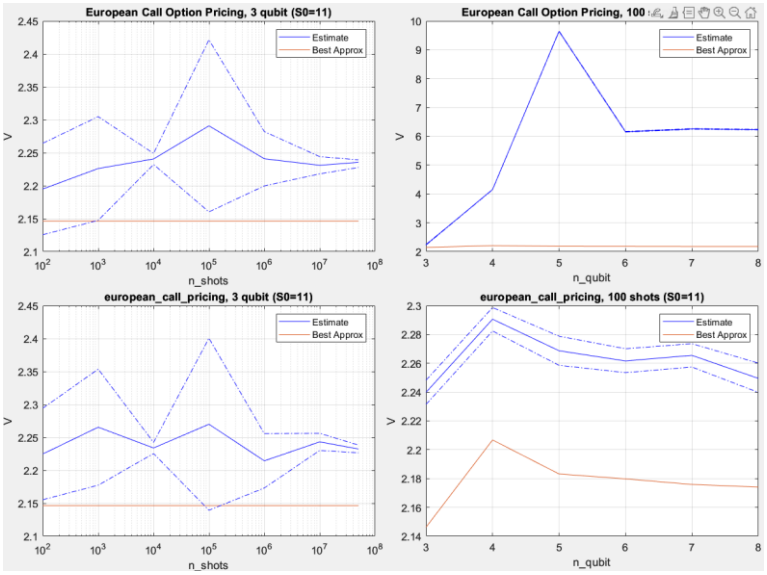


Figura 21: Valutazione di un'opzione con parametri modificati, si utilizza il programma "European Call Option Pricing" in alto, si utilizza la funzione "european_call_pricing" in basso. A sinistra si mostra l'andamento per 3 qubit al variare del numero di misure, a destra l'andamento al variare del numero di qubit eseguendo 100 misure.

Si osservano gli stessi andamenti anche con parametri finanziari differenti, in particolare la funzione $european\_call\_pricing$ sembra rimanere più vicina al valore esatto, nonostante in questo caso esso non si sovrapponga con l'intervallo di confidenza.
Per quanto riguarda il programma $European\ call\ option\ pricing$ si continua ad osservare una forte sottostima dell'intervallo di confidenza.

6. Conclusione

Per quanto riguarda l'algoritmo unario si è mostrata più immediata implementabilità e si riescono a condurre test ma l'algoritmo di Amplitude Estimation non risulta utile alla soluzione del problema.

Riguardo il test dei programmi già presenti in qiskit è stato tenuto un approccio "black box", utilizzandoli così come vengono messi a disposizione. I risultati non sono soddisfacenti e fanno supporre la presenza di dettagli o condizioni ignorati con questo tipo di approccio.

Il programma "Europrean Call Option Pricing" fornisce risultati significativamente distanti rispetto ai valori esatti, in particolare divergenti rispetto al valore esatto. La stima migliora utilizzando la funzione "european_call_pricing", anche se resta inconsistente. Non è quindi possibile effettuare un confronto tra un metodo Monte Carlo classico e queste alternative di natura quantistica.

In generale non si osserva un miglioramento nella stima all'aumentare dei qubit utilizzati, contrariamente a quanto si osserva in [6] e [7] e contrariamente alle aspettative.

Un'importante anomalia osservata è un pattern all'interno della distribuzione dell'intervallo di confidenza: sembra infatti essere presente una correlazione diversa da quella attesa tra l'ampiezza dell' intervallo di confidenza ed il numero di misure effettuate. Ci si apetta infatti che l'intervallo di confidenza si restringa attorno al valore medio aumentando il numero di misure, ciò che si osserva però in tutti i risultati ottenuti è un intervallo particolarmente stretto in corrispondenza del risultato ottenuto tramite 1000 misure, che poi si allarga per il valore successivo prima di entrare in un trend decrescente.

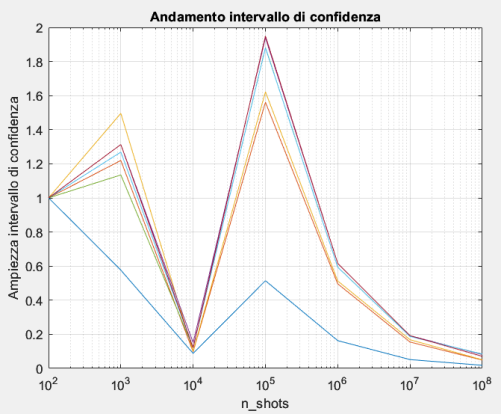


Figura 22: Andamento anomalo dell'intervallo di confidenza, si inseriscono i risultati sia per il programma "European Call Option Pricing" che con la funzione "european_call_pricing" relativi a 3 qubit al variare del numero di misure. Si normalizza a 1 l'intervallo di confidenza del primo risultato, che si riferisce a 100 misure.

Per mettere in luce questa anomalia si mostra il grafico che presenta l'ampiezza dell'intervallo di confidenza di ciascuna delle simulazioni mostrate in precedenza in funzione del numero di shots, ovvero di misure. Si normalizza a 1 l'ampiezza iniziale (riferita a 100 misure) e si nota come l' andamento sia lo stesso per quasi tutte le simulazioni effettuate con una piccola eccezione relativa alla simulazione a 4 qubit per $European\ Call\ Option$.
Questo andamento è anomalo e non risulta previsto dagli algoritmi utilizzati, si ritiene di conseguenza che sia indice di un bug oppure di dettagli non considerati, anche se impossibile da confermare in quanto non è presente una documentazione riguardo la definizione dell'intervallo di confidenza.

Risulta quindi necessario andare più a fondo all'interno degli algoritmi e funzioni built-in del pacchetto qiskit per poter manipolare i programmi o le funzioni e poterli ottimizzare al fine di ottenere risultati più consistenti.

In ogni caso l’approccio quantistico, sembra avere un forte potenziale dal punto di vista teorico in termini di miglioramento del metodo Monte Carlo, portando ad un impatto anche nell’ambito della finanza. Per ora, tuttavia, i sistemi fisici disponibili al pubblico non rendono possibile ottenere risultati paragonabili ad alternative classiche; in particolare il numero di qubit limitato gioca un forte ruolo nel limitare la precisione delle stime che si possono ottenere.

Appendice

1- **European Call Option:**
Notebook Python European Call Option Pricing (Qiskit)
2- **Amplitude Estimation:**
Notebook Python per Amplitude Estimation (Qiskit)
3- **Circuito unario:**
Notebook Pyton per il circuito unario (GitHub personale)

Bibliografia

[1]. David J. Griffiths - Introduzione alla Meccanica Quantistica – 1995
[2]. P.Wilmott, J. Dewynne and S. Howison, 'Option Pricing', Oxford Financial Pres0073, 1993
[3]. R. Seydel, Tools for Computational Finance, Springer, 2009
[4]. P. Rebentrost, Quantum computational finance: Monte Carlo pricing of financial derivatives, arXiv:1805.00109, 2018
[5]. Sergi Ramos-Calderer, Quantum unary approach to option pricing, 2019 arXiv:1912.01618
[6]. N. Stamatopoulos, Daniel J. Egger, Option Pricing using Quantum Computers, arXiv:1905.02666, 2020
[7]. Stefan Woerner, Daniel J. Egger, Quantum risk analysis, arXiv:1806.06893, 2019
[8]. Michael A. Nielsen, Isaac L. Chuang, Quantum computation and quantum information, Cambridge, 2010