

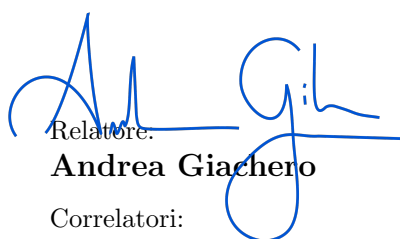
Università degli Studi di Milano-Bicocca

DIPARTIMENTO DI FISICA "GIUSEPPE OCCHIALINI"

Corso di Laurea Triennale in Fisica



Algoritmi quantistici per la risoluzione di equazioni differenziali


Relatore:
Andrea Giachero
Correlatori:
Roberto Moretti
Danilo Labranca

Candidato:
Federico Shin'ichi Finardi
Matricola 813878

Anno Accademico 2022-2023

Sommario

La presente tesi si colloca nell'ambito del *Quantum Computing*, una branca delle tecnologie quantistiche in forte ascesa negli ultimi anni, che si fonda sulla possibilità di manipolare coerentemente stati quantistici per risolvere problemi.

L'unità fondamentale dell'informazione quantistica è il *qubit*, ossia un sistema quantistico a due livelli, $|0\rangle$ ed $|1\rangle$, che costituiscono la base computazionale. Lo stato generico di un qubit è esprimibile come una sovrapposizione quantistica dei due autostati, ed è possibile implementare operazioni su di esso tramite *quantum gates*, ovvero l'estensione quantistica delle porte logiche tradizionali. Una sequenza di quantum gates applicata ad un sistema multi-qubit, prende il nome di *circuito quantistico*. La corretta progettazione di un circuito quantistico è dunque un aspetto cruciale nello sviluppo di algoritmi quantistici.

In questo lavoro di tesi viene utilizzato il pacchetto *Qiskit*, sviluppato dall'azienda IBM, il quale permette sia di simulare il comportamento di circuiti quantistici su hardware classico, che di programmare processori quantistici reali in cloud.

Nella presente tesi si propone di sviluppare un algoritmo quantistico per la risoluzione di *equazioni differenziali ordinarie* (EDO), in particolare quella dell'oscillatore armonico smorzato. A tale scopo, vengono presi in considerazione dei circuiti quantistici variazionali, ossia circuiti che dipendono da un insieme di parametri da ottimizzare. In questo contesto, un circuito variazionale è associato all'equazione incognita dipendente dal tempo $f(t)$, mentre le sue derivate vengono calcolate applicando la *Parameter-shift Rule*. Quest'ultimo costituisce un metodo esatto per il calcolo della derivata, ossia non si basa sul calcolo di differenze finite, come la maggior parte degli algoritmi classici.

In particolare, la tesi descrive la strategia utilizzata per approssimare la soluzione dell'equazione differenziale tramite l'ottimizzazione dei parametri del circuito variazionale in input. Quest'ultima è stata condotta in modo da minimizzare un'opportuna funzione di costo, che quantifica la bontà della soluzione fornita. A questo fine sono stati utilizzati due algoritmi di ottimizzazione, chiamati *COBYLA* ed *Adam*.

Il procedimento sperimentale adottato è consistito nell'esaminare diversi circuiti quantistici variazionali, in modo da analizzare le prestazioni dell'algoritmo sviluppato, considerando diverse topologie del circuito, variando il numero di qubit, il numero dei gradi di libertà da ottimizzare, il numero di quantum gates e la quantità di entanglement.

Le analisi effettuate sono state valutate per sistemi con diverso numero di qubit fino ad un massimo di 6. Dai risultati ottenuti, abbiamo potuto osservare come,

all'aumentare del numero di qubit e del numero di gradi di libertà del circuito, la funzione di costo (l'errore quadratico medio rispetto alla soluzione esatta) tenda a migliorare di un fattore di circa 3 passando da 2 a 6 qubit. Un altro risultato di interesse si è ottenuto aumentando il numero di gate dipendenti dal tempo all'interno del circuito, piuttosto che aumentando il numero di qubit. Ciò mostra che un sistema di pochi qubit, aventi molti gate dipendenti da t , può fornire risultati confrontabili con configurazioni a più qubit, se questi hanno un numero minore di gate dipendenti dal tempo. Questo risultato suggerisce la possibilità di avere alta efficienza computazionale nella risoluzione di una EDO, anche senza la necessità di ricorrere a sistemi multi qubit.

Il lavoro svolto apre la strada allo studio di equazioni differenziali più complesse e a testare ulteriori strategie di ottimizzazione, come l'utilizzo di circuiti variazionali più elaborati e costruiti su misura per le specifiche architetture quantistiche oggi esistenti.

Indice

1	Computer Quantistici	5
1.1	Quantum Computing	5
1.2	Stati a qubit singolo e multipli	6
1.2.1	Qubit	6
1.2.2	Misura e osservabile fisica	7
1.2.3	Sfera di Bloch	8
1.2.4	Sistemi a più qubit	9
1.3	Quantum Gates	10
1.3.1	Pauli gates	10
1.3.2	Hadamard gate	11
1.3.3	Porte rotazionali	12
1.3.4	Controlled NOT gate	12
1.4	Hardware di un qubit	14
1.5	IBM e Qiskit	16
2	Risoluzione di EDO attraverso circuiti quantistici	18
2.1	Risoluzione classica di equazioni differenziali	18
2.1.1	Metodo Runge Kutta	19
2.1.2	Oscillatore Armonico Semplice	20
2.1.3	Oscillatore Armonico Smorzato	21
2.2	Ottimizzazione di circuiti quantistici variazionali	23
2.2.1	COBYLA	23
2.2.2	Adam	25
2.2.3	Funzione di costo	27
2.3	Qiskit Gradient e Parameter-shift Rule	29
2.3.1	Parameter-shift Rule	29
3	Implementazione dell'algoritmo e analisi dei risultati	33
3.1	Descrizione dell'algoritmo realizzato	33
3.1.1	Circuito quantistico variazionale (Ansatz)	33
3.1.2	Osservabile Hermitiana e valore di aspettazione	34
3.1.3	Gradiente dello stato quantistico	36
3.1.4	Ottimizzazione dei parametri quantistici	37
3.1.5	Confronto con il metodo classico di Runge-Kutta	40
3.2	Sperimentazione e risultati	40

3.2.1	Circuito quantistico 1: $RY(t)$	42
3.2.2	Circuito quantistico 2: $RY(t) + CNOT$	46
3.2.3	Circuito quantistico 3: $RY(t) + CRX$	49
3.2.4	Circuito quantistico 4: $RY(t) + CRX(t)$	51
3.2.5	Confronto dei risultati fra i circuiti	54
3.2.6	Influenza dei gate dipendenti dal tempo	56
3.3	Confronto tra gli algoritmi COBYLA e Adam	58
3.4	Test su hardware quantistico	60
4	Conclusioni	65

Capitolo 1

Computer Quantistici

1.1 Quantum Computing

Quando si parla di computer quantistici ci si riferisce a quei dispositivi in grado di prendere in ingresso dei dati ed elaborarli sfruttando fenomeni fisici che sono descrivibili dalla meccanica quantistica.

Il quantum computing mira a sfruttare particolari proprietà della meccanica quantistica, come la sovrapposizione, l'entanglement e l'interferenza, con lo scopo di ottenere un vantaggio quantistico rispetto ai sistemi di calcolo tradizionali. Tuttavia, è bene chiarire che il ramo della computazione quantistica non vuole essere un sostituto dei processi classici attualmente in uso, ma piuttosto un complemento che offre nuove opportunità e soluzioni per risolvere specifici problemi complessi in modo più efficiente.

Si parla spesso di supremazia quantistica o vantaggio quantistico. Secondo la definizione di John Preskill [1], ci si riferisce alla supremazia quantistica quando un computer quantistico può risolvere problemi irrisolvibili per i computer classici, indipendentemente dalla loro rilevanza pratica. Un esempio famoso è stato l'annuncio da parte di Google nel 2019 [2], in cui affermavano di aver raggiunto la supremazia quantistica risolvendo un particolare problema in pochi minuti. Secondo le loro stime, persino il computer classico più potente avrebbe impiegato 10^4 anni per risolverlo. Tuttavia, dal momento che si trattava di un problema dimostrativo con un interesse pratico molto limitato e specifico, l'obiettivo della supremazia quantistica rimane ancora oggi un argomento di dibattito aperto all'interno delle comunità scientifiche. Con il concetto di vantaggio quantistico, ci si riferisce invece alla capacità di un computer quantistico nel risolvere problemi specifici, che siano utili nel mondo scientifico o industriale, molto più velocemente di qualunque algoritmo classico conosciuto su hardware classico. Recentemente, IBM ha annunciato di aver raggiunto un vantaggio quantistico [3]. Tuttavia, tali risultati sono stati successivamente smentiti da altri gruppi di ricerca.

In linea generale, i computer quantistici sono sistemi capaci di eseguire specifici algoritmi in maniera esponenzialmente più rapida rispetto ai computer classici. Ciò però non implica necessariamente che siano più efficienti in tutti i tipi di calcolo.

Nonostante i rapidi progressi in questo campo, ad oggi non disponiamo ancora di computer quantistici che possano eseguire lunghe serie di operazioni senza commettere errori, a differenza dei computer classici.

1.2 Stati a qubit singolo e multipli

1.2.1 Qubit

L'oggetto matematico che rappresenta l'unità fondamentale del quantum computing è il bit quantistico o qubit. A differenza del bit classico, che può assumere solo due possibili valori, 0 o 1, il qubit può essere descritto sia da singoli stati corrispondenti ai valori classici (0 e 1), sia da uno stato più generale espresso come una combinazione lineare di questi. In sostanza, lo stato di un qubit viene definito come una sovrapposizione degli stati di base $|0\rangle$ e $|1\rangle$ (in notazione di Dirac). Sfruttando le leggi della meccanica quantistica, è possibile dunque definire uno oggetto matematico di base in grado di accedere ad una quantità maggiore di informazioni rispetto all'analogo classico, consentendo l'elaborazione in parallelo di più dati codificati nella sovrapposizione dei suoi stati interni. Tuttavia è importante tener presente che, a causa del collasso dello stato a seguito di un processo di misura, solo parte di queste informazioni contenute nel sistema saranno a noi accessibili.

Cerchiamo quindi ora di capire meglio come si struttura matematicamente lo stato di un qubit e cosa si intende con il concetto di misura.

La sovrapposizione dello stato di un qubit può essere espressa nella seguente forma:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \quad (1.1)$$

dove $|0\rangle$ e $|1\rangle$ costituiscono una base ortonormale di vettori complessi nello spazio di Hilbert di dimensione 2:

$$\mathcal{B} = \left\{ |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\} \quad (1.2)$$

questa base prende il nome di *base computazionale*. Mentre α e β sono coefficienti complessi che devono soddisfare la condizione di normalizzazione per uno stato quantistico, ossia:

$$|\alpha|^2 + |\beta|^2 = 1 \quad (1.3)$$

In cui ciascun modulo quadro esprime la probabilità che, effettuando una misura sullo stato $|\psi\rangle$, il sistema collassi sul corrispettivo autostato della base computazionale. Ma cosa significa effettuare una misura?

1.2.2 Misura e osservabile fisica

Con il termine di *misura*, in meccanica quantistica si descrive un'interazione tra il sistema che viene misurato (descritto dal vettore di $|\psi\rangle$) e l'apparato di misura (descritto da un operatore Hermitiano che vedremo a breve). Tale interazione causa inevitabilmente una perturbazione e quindi una modifica del sistema, da cui lo stato quantistico è portato a ridursi (collassare) in un singolo stato definito. Questo è l'unico modo per conoscere il valore di un qubit. È importante però precisare che l'azione di misura su un sistema costituisce un processo non deterministico e irreversibile, in quanto il risultato ottenuto dal collasso non può essere previsto con certezza e implica la perdita di tutte le informazioni che il sistema (o nel nostro caso il qubit) conteneva prima della misura stessa.

L'informazione che si voglia misurare relativamente ad uno stato è strettamente dipendente dall'apparato utilizzato e dall'operatore che lo descrive.

Ogni osservabile fisica è descritta da un operatore Hermitiano i cui risultati a seguito di una misura, corrispondono ai possibili autovalori associati agli autostati dell'osservabile. Per conoscere tale risultato, è necessario prima riscrivere lo stato del sistema attraverso la stessa base sul quale è definito l'operatore Hermitiano. Facciamo un esempio. Supponiamo di voler effettuare una misura dello spin di una particella attraverso l'operatore \hat{S}_z , in cui però lo stato di tale particella è definito in una base differente da quella dell'operatore, come nella seguente forma:

$$|\psi\rangle = a |+\rangle + b |-\rangle \quad (1.4)$$

$$\text{con } |+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \text{ e } |-\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

L'operatore di spin lungo l'asse z, descritto dalla matrice di Pauli σ_z può esser espresso attraverso la seguente definizione di operatore:

$$\begin{aligned} \hat{O} &= \sum_i \lambda_i |v_i\rangle \langle v_i| \\ \Rightarrow \hat{S}_z &= \frac{\hbar}{2} (|\uparrow\rangle \langle \uparrow| - |\downarrow\rangle \langle \downarrow|) = \frac{\hbar}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \frac{\hbar}{2} \sigma_z \end{aligned} \quad (1.5)$$

in cui λ_i e $|v_i\rangle$ sono l'autovalore e l'autostato i -esimi del generico operatore \hat{O} . Per avere una più corretta e semplice interpretazione dei risultati ottenuti attraverso una misura, è opportuno a questo punto riscrivere lo stato $|\psi\rangle$ come una combinazione lineare degli autostati della base dell'operatore di spin \hat{S}_z :

$$\chi_s = \left\{ |\uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; |\downarrow\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\} \quad (1.6)$$

$$\begin{aligned} |+\rangle &= \frac{1}{\sqrt{2}} (|\uparrow\rangle + |\downarrow\rangle) \\ |-\rangle &= \frac{1}{\sqrt{2}} (|\uparrow\rangle - |\downarrow\rangle) \end{aligned} \quad (1.7)$$

da cui otteniamo:

$$|\psi\rangle = \frac{a+b}{\sqrt{2}} |\uparrow\rangle + \frac{a-b}{\sqrt{2}} |\downarrow\rangle \quad (1.8)$$

In questo modo lo stato della particella risulta scritto in una forma utile ad ottenere delle informazioni complete relative ai possibili stati sul quale il sistema può collassare a seguito di una misura di spin.

Lo stesso vale per i qubit. Per ottenere la corretta misurazione del valore di uno stato di un qubit dopo una serie di operazioni quantistiche, è necessario conoscere la base computazionale su cui lo stato è definito. Altrimenti, si rischia di ottenere informazioni errate o fuorvianti.

1.2.3 Sfera di Bloch

Tornando all'analisi dello stato di un qubit e alla sua condizione di normalizzazione (1.3), possiamo pensare ad una possibile scrittura dello stato in questo modo:

$$|\psi\rangle = e^{i\gamma} \left[\cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\varphi} \sin\left(\frac{\theta}{2}\right) |1\rangle \right] \quad (1.9)$$

dove il fattore di fase $e^{i\gamma}$ può essere trascurato in quanto non contribuisce in alcun modo alla probabilità di ottenere una certa misura per un qualunque calcolo fisico. La dipendenza dai parametri θ e φ ci permette di ricorrere ad un espediente grafico utile a visualizzare tutte le possibili configurazioni di stato che un qubit può assumere. Tale rappresentazione prende il nome di *Sfera di Bloch* e descrive lo stato del qubit come un vettore (chiamato vettore di Bloch) in grado di mappare tutti i punti sulla superficie di una sfera unitaria al variare dei due parametri.

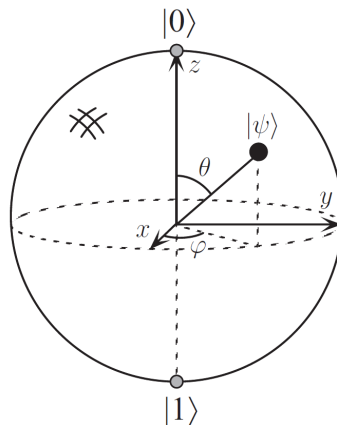


Figura 1.1: Rappresentazione della Sfera di Bloch [4]

La sfera di Bloch è uno strumento utile a visualizzare gli stati, le trasformazioni di un singolo qubit e facilitare l'analisi e la progettazione di algoritmi quantistici.

1.2.4 Sistemi a più qubit

Cerchiamo ora di capire come estendere i ragionamenti visti finora relativi a un singolo qubit, a un sistema a più qubit. Quando si studia un singolo qubit, lo spazio di Hilbert in cui esso vive ha dimensione 2, poiché gli unici stati accessibili sono $|0\rangle$ e $|1\rangle$. Nel momento in cui si considera un sistema a più qubit, il numero delle possibili combinazioni degli stati aumenta esponenzialmente, in quanto combinando gli spazi di Hilbert di ciascun qubit tra di essi, lo spazio vettoriale complessivo si estende (mediante il prodotto tensoriale). Per semplificare, prendiamo ad esempio un sistema composto da due soli qubit. Siano H_1 e H_2 gli spazi di Hilbert rispettivi a ciascun qubit. Attraverso il prodotto tensoriale tra questi, indicato come $H_1 \otimes H_2$, verrà generato uno spazio vettoriale composto da tutte le possibili combinazioni lineari dei vettori di base dei due sottospazi. Quindi, lo spazio di Hilbert totale del sistema a due qubit sarà:

$$H = H_1 \otimes H_2 = \{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\} \quad (1.10)$$

che possiamo riscrivere in maniera più compatta come:

$$H = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\} \quad (1.11)$$

In generale, se abbiamo n qubit, il prodotto tensoriale degli spazi di Hilbert dei singoli qubit genererà uno spazio vettoriale di dimensione pari a 2^n , poiché ogni qubit può assumere uno dei due stati possibili ($|0\rangle$ o $|1\rangle$). in questo modo, a differenza dei computer classici, la quantità di informazione che può essere codificata in un sistema quantistico, può essere molto elevata anche per un numero relativamente piccolo di qubit (per esempio, $n=60$).

Consideriamo ora due stati $|\varphi_1\rangle$ e $|\varphi_2\rangle$, presi rispettivamente uno dal sottospazio H_1 e l'altro dal sottospazio H_2 . In tal caso, lo stato complessivo appartenente allo spazio di Hilbert totale H , potrà essere riscritto come:

$$|\psi\rangle = |\varphi_1\rangle \otimes |\varphi_2\rangle \quad (1.12)$$

Uno stato a più qubit riscrivibile in questo modo, prende il nome di *Stato separabile*. Tuttavia, è importante tenere presente che non sempre è possibile rappresentare lo stato di un sistema a più qubit come prodotto tensoriale degli stati a singolo qubit. Facciamo un esempio:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle) \quad (1.13)$$

In questo caso, non esiste un modo per effettuare una riduzione algebrica che consenta di distinguere i due possibili stati a singolo qubit come nella forma (1.12). Uno stato di questo tipo prende il nome di *Stato entangled* e la sua peculiarità è che le proprietà di un singolo qubit non possono essere descritte in modo indipendente dagli altri qubit

correlati. Ciò significa che, misurare lo stato di un qubit influenzerà istantaneamente lo stato degli altri qubit nel sistema. Questa caratteristica degli stati entangled è uno degli elementi distintivi del quantum computing, che consente la realizzazione di algoritmi quantistici potenzialmente più efficienti dei sistemi classici nell'ambito della risoluzione di problemi complessi e dell'elaborazione delle informazioni.

1.3 Quantum Gates

Analogamente alla situazione classica, in cui vengono definiti degli strumenti matematici utili a svolgere diverse tipologie di operazioni logiche su dei bit, anche nel caso del quantum computing è possibile definire degli elementi che consentano di operare sui qubit. Questi elementi prendono il nome di *quantum gates* e consistono in porte logiche quantistiche descritte da matrici unitarie ($UU^\dagger = I$) di dimensione $2^N \times 2^N$, dove N rappresenta il numero di qubit del sistema.

Una particolarità delle porte quantistiche (conseguenza dell'unitarietà), è il cosiddetto *reversible computing*, ovverossia la caratteristica di essere invertibili. Nella computazione classica, le operazioni logiche sono reversibili solo in modo limitato, il che significa che alcune informazioni vengono perse durante il calcolo e di conseguenza, la computazione stessa non può essere eseguita in maniera inversa senza perdita di informazioni. Ciò porta a considerare la computazione classica generalmente irreversibile. Al contrario, nella computazione quantistica, le porte logiche ammettono operazioni inverse, consentendo di operare in maniera completamente reversibile e di poter ripercorrere i calcoli retroattivamente, senza perdita di informazioni.

1.3.1 Pauli gates

Consideriamo i gate a singolo qubit (ossia quelli agenti su un solo qubit). Le matrici unitarie corrispondenti hanno dimensione 2×2 e le più comuni che possiamo prendere come esempio, sono le matrici di Pauli σ_x , σ_y e σ_z :

$$\begin{aligned}\sigma_x \equiv X &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = |0\rangle \langle 1| + |1\rangle \langle 0| \\ \sigma_y \equiv Y &= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = -i |0\rangle \langle 1| + i |1\rangle \langle 0| \\ \sigma_z \equiv Z &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = |0\rangle \langle 0| + |1\rangle \langle 1|\end{aligned}\tag{1.14}$$

Applicando singolarmente queste porte agli stati della base computazionale otteniamo le seguenti trasformazioni:

	X - Gate	Y - Gate	Z - Gate
Stato iniziale	Stato finale		
$ 0\rangle$	$ 1\rangle$	$i 1\rangle$	$ 0\rangle$
$ 1\rangle$	$ 0\rangle$	$-i 0\rangle$	$- 1\rangle$

Tabella 1.1: Risultati relativi alle applicazioni delle matrici di Pauli sugli autostati della base computazionale

Possiamo osservare che l'applicazione della porta **X** o **Y** comporta un cambio di autostato del qubit. Questo avviene perché le matrici che rappresentano queste porte, operano su una base differente rispetto alla base computazionale utilizzata. Inoltre, la porta **X** è anche conosciuta come porta **NOT**, in quanto il risultato ottenuto è analogo all'operazione di negazione per una porta classica.

Per quanto riguarda invece l'applicazione della porta **Z**, poiché l'operazione avviene sulla stessa base computazionale, lo stato finale del qubit viene mantenuto invariato (a meno di una fase). In altri termini, se lo stato iniziale del qubit è $|0\rangle$ o $|1\rangle$, l'applicazione della porta **Z** non altera lo stato, rimanendo rispettivamente $|0\rangle$ o $|1\rangle$. Tuttavia, se lo stato del qubit è una sovrapposizione di $|0\rangle$ e $|1\rangle$, l'applicazione della porta **Z** influenzerà le fasi relative degli stati combinati, lasciando però invariata la probabilità di trovare il qubit in uno dei due autostati durante una misura.

1.3.2 Hadamard gate

L'Hadamard Gate (o **H Gate**), è una porta logica quantistica a singolo qubit, la cui particolarità è quella di produrre una sovrapposizione di stato quantistica, in cui il qubit (inizialmente definito per semplicità mediante uno stato della base computazionale) avrà una probabilità del 50% di essere misurato nello stato $|0\rangle$ e una probabilità del 50% di essere misurato nello stato $|1\rangle$. In altri termini, l'Hadamard Gate crea una sovrapposizione di probabilità tra gli stati di base, consentendo una maggiore versatilità e flessibilità nella manipolazione delle informazioni quantistiche.

La rappresentazione matriciale di questa porta può essere espressa come:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}} (|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|) \quad (1.15)$$

A seguito della sua applicazione sugli autostati della base computazionale, si ottiene:

	H - Gate
Stato iniziale	Stato finale
$ 0\rangle$	$\frac{1}{\sqrt{2}}(0\rangle + 1\rangle) = +\rangle$
$ 1\rangle$	$\frac{1}{\sqrt{2}}(0\rangle - 1\rangle) = -\rangle$

Tabella 1.2: Risultati relativi all'applicazione della matrice di Hadamard sugli autostati della base computazionale

1.3.3 Porte rotazionali

Altri gate a singolo qubit che torneranno molto utili per lo studio di questa tesi, sono le porte rotazionali R_X , R_Y e R_Z , le quali vengono definite come matrici unitarie generate mediante gli operatori di Pauli. La relazione che le descrive è la seguente:

$$R_A(\theta) = I \cos(\theta/2) - iA \sin(\theta/2) \quad \text{con } A = X, Y, Z \quad (1.16)$$

da cui possiamo esprimere le loro rispettive rappresentazioni matriciali:

$$\begin{aligned}
R_X(\theta) &= e^{-iX\theta/2} = \begin{pmatrix} \cos(\frac{\theta}{2}) & -i \sin(\frac{\theta}{2}) \\ -i \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix} \\
R_Y(\theta) &= e^{-iY\theta/2} = \begin{pmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix} \\
R_Z(\theta) &= e^{-iZ\theta/2} = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}
\end{aligned} \quad (1.17)$$

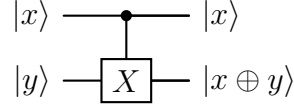
In sostanza, attraverso il parametro angolare θ , ciascuna di queste matrici descrive una rotazione arbitraria di un qubit attorno ad uno specifico asse della sfera di Bloch. La combinazione di queste porte logiche parametriche all'interno di un circuito quantistico, permette di svolgere una sequenza ottimale di operazioni attraverso la ricerca dei migliori parametri. Come vedremo più avanti, tale ricerca può essere effettuata mediante particolari algoritmi di ottimizzazione sia di tipo classico, che quantistico.

1.3.4 Controlled NOT gate

Come accennato in precedenza, è possibile definire un gate quantistico anche con dimensioni maggiori di 2 in relazione al numero di qubit sul quale questo viene applicato. Un esempio molto importante è la porta CNOT (o Controlled NOT Gate). Si tratta di una porta che agisce su due qubit (dunque descritta da una matrice

4×4), uno di controllo ed uno di target. In sostanza, a seconda del valore del primo, si avrà o meno, un cambio di stato del secondo.

La sua rappresentazione grafica, si può osservare nel seguente schema, con l'applicazione di una porta NOT al qubit di target, la quale viene collegata al qubit di controllo in modo da poter essere azionata o meno a seconda dello stato di quest'ultimo:



Circuito 1.1: Rappresentazione a blocchi del Controlled-NOT Gate

dove il simbolo \oplus indica l'operazione aritmetica modulo 2 sulla somma, e corrisponde a scrivere $(x + y) \bmod 2$.

Dalle seguenti tabelle si può osservare un'analogia tra il comportamento quantistico del target e il risultato classico per una porta XOR:

CNOT - Gate		a XOR b		
$ \text{control}\rangle \otimes \text{target}\rangle$	Stato finale	bit a	bit b	bit finale
$ 00\rangle$	$ 00\rangle$	0	0	0
$ 01\rangle$	$ 01\rangle$	0	1	1
$ 10\rangle$	$ 11\rangle$	1	0	1
$ 11\rangle$	$ 10\rangle$	1	1	0

(a) CNOT Gate

(b) XOR Gate

Tabella 1.3: Nella tabella (a), sono riportati i risultati relativi all'applicazione della porta CNOT su due qubit. Si può notare come il qubit di target cambi di stato solamente quando il qubit di controllo è allo stato $|1\rangle$. Nella tabella (b), sono riportati i risultati relativi all'applicazione della porta XOR su due bit classici.

la cui la forma matriciale può essere espressa nel seguente modo:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (1.18)$$

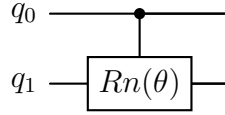
L'importanza di questo gate risiede soprattutto nel fatto che la sua matrice non derivi da una fattorizzazione di sottomatrici a singolo qubit (come invece abbiamo visto essere possibile per gli stati separabili, secondo la relazione (1.12)). Per tale ragione, la sua applicazione su una generica coppia di qubit, permette di dare luogo a uno stato complessivo di tipo entangled.

Osservazioni:

- I gates a singolo qubit finora presentati, sono casi particolari della quantum gate U_3 , la quale rappresenta una rotazione arbitraria del gruppo $SU(2)$:

$$U_3(\theta, \phi, \lambda) = \begin{pmatrix} \cos(\theta) & -e^{i\lambda} \sin(\theta) \\ e^{i\phi} \sin(\theta) & e^{i(\phi+\lambda)} \cos(\theta) \end{pmatrix} \quad (1.19)$$

- In analogia alla CNOT, che implementa la porta X su un qubit target condizionato da un qubit di controllo, è possibile implementare una porta rotazionale R_n attorno all'asse x, y o z, sul qubit di target, gestita anch'essa da un qubit di controllo:



Circuito 1.2: Rappresentazione a blocchi di una **Controlled- R_n gate** con $n=x,y,z$.

- La composizione di diverse porte quantistiche, permette la realizzazione dei cosiddetti circuiti quantistici, i quali costituiscono il fondamento del quantum computing. In altri termini, possiamo considerare i circuiti quantistici come una sequenza di porte logiche in grado di svolgere particolari operazioni su stati quantistici, in modo da poter eseguire calcoli di diversa complessità e sfruttare le proprietà quantistiche della sovrapposizione e dell'entanglement.

1.4 Hardware di un qubit

La realizzazione hardware di un qubit dipende dalla tecnologia utilizzata per implementarlo. Tra le diverse tipologie una delle più comuni, sulla quale porteremo l'attenzione in questa sezione, è quella dei *qubit superconduttori*. In un apparato di questo tipo, un qubit viene realizzato utilizzando un circuito elettronico superconduttore che opera a temperature estremamente basse (circa 20 mK). Quando un materiale superconduttore viene raffreddato al di sotto della sua temperatura di transizione critica, entra in gioco un particolare fenomeno della meccanica quantistica, in cui gli elettroni sono in grado di attraversare i materiali senza incontrare resistenza, permettendo in questo modo di creare una corrente elettrica continua che fluisca senza dissipazione di energia. La ragione di tale fenomeno è dovuta al fatto che gli elettroni nei materiali superconduttori, accoppiandosi tra di loro, formano degli stati legati noti come *coppie di Cooper*¹. Grazie alla loro natura bosonica, dovuta agli spin

¹Questo accoppiamento è mediato dalle interazioni tra gli elettroni e la rete cristallina del materiale.

opposti degli elettroni legati, le coppie di Cooper sono in grado di occupare tutte lo stesso stato di singola particella, formando così lo stato condensato di Bose-Einstein. La funzione d'onda complessiva, descrive il comportamento collettivo di tutte le coppie di Cooper nel superconduttore e permette loro di manifestare le proprietà uniche della superconduttività.

Cerchiamo quindi di capire come sfruttare quanto analizzato, per la realizzazione fisica di un qubit e come di conseguenza poter distinguere i suoi stati. Immaginiamo di costruire un circuito LC puramente reattivo (si veda la Figura 1.2 (a)) in cui la conduzione di corrente elettrica sia idealmente possibile sotto forma di coppie di Cooper. A livello quantistico, l'energia del sistema può essere descritta mediante l'hamiltoniana dell'oscillatore armonico quantistico, considerando la carica e il flusso di carica come se fossero rispettivamente posizione e momento di una particella [5]. Tuttavia, la realizzazione di un circuito di questo genere, comporta una spaziatura equidistante tra i suoi livelli di energia di una quantità $\hbar\omega$ (caratteristica dell'oscillatore armonico), impedendoci così la possibilità di avere un sistema a due livelli distinguibili che descrivano gli stati di un qubit (si veda Figura 1.2 (b)). Per ovviare a ciò, possiamo introdurre nel circuito un'induttanza non lineare (in sostituzione a quella lineare, L) che produca degli effetti anarmonici al sistema, rendendo distinguibili le frequenze di transizione dei livelli energetici e avere in altri termini, una rottura della degenerazione sulla spaziatura tra i livelli (si veda Figura 1.2 (c) e (d)).

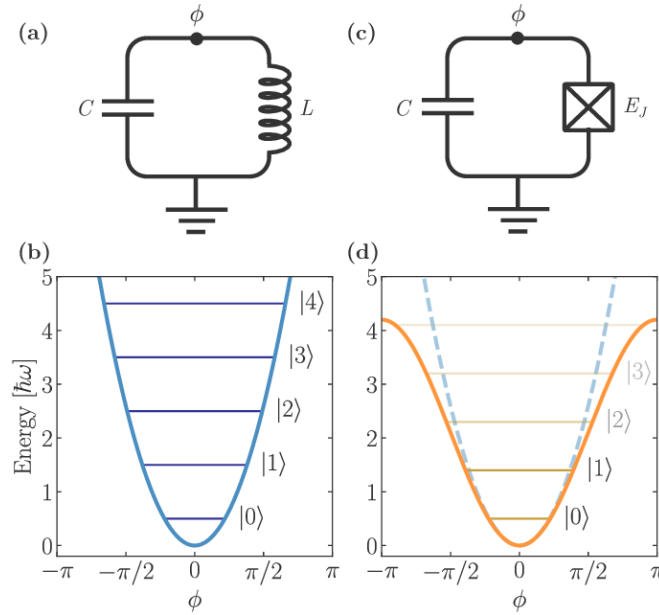


Figura 1.2: Rappresentazione dei circuiti L-C e JJ-C con i loro rispettivi potenziali, armonico per il primo e anarmonico per il secondo.

Tale componente non lineare prende il nome di *giunzione Josephson*, e consiste in un elemento circuitale non-dissipativo che ci permette di modificare la forma funzionale del potenziale armonico, in modo da poter realizzare un sistema a due livelli distinti sul quale associare i due stati del qubit: $|0\rangle$ per lo stato fondamentale e $|1\rangle$ per il primo stato eccitato (con le relative frequenze di transizione $\omega^{0 \rightarrow 1}$ e

$\omega^{1 \rightarrow 2}$). In questa configurazione, la dinamica del sistema è influenzata principalmente dall'energia di Josephson (E_J) e dall'energia associata alla capacità totale del circuito (E_C). Pertanto, il rapporto tra queste due energie, E_J/E_C , diventa un parametro importante per determinare il comportamento del sistema e le sue caratteristiche dominanti. In particolare, gli studi dei qubit a superconduttore si concentrano principalmente nel regime $E_J \gg E_C$ in quanto, rispetto alla condizione $E_J \leq E_C$, si osserva una notevole riduzione del rumore associato alla variazione di carica sulla giunzione e una maggiore stabilità del potenziale hamiltoniano.

Tra i qubit a superconduttore, ne classifichiamo alcuni in particolare che lavorano a regime $E_J/E_C \sim 100$, e che prendono il nome di *qubit Transmon*. Questi tipi di circuiti, vengono realizzati cercando di ridurre l'energia di carica E_C attraverso l'utilizzo di condensatori con capacità elevate ($E_C = e^2/(2C)$).

Un altro fattore molto importante da tenere in considerazione dei qubit Transmon (come per tutti i qubit quantistici d'altronde), è il fatto di esse soggetti a un fenomeno noto come decoerenza, che può portare ad errori nella computazione quantistica. Con il termine decoerenza ci si riferisce alla perdita di stabilità degli stati quantistici nel sistema a causa delle interazioni con l'ambiente circostante. Queste interazioni possono causare la transizione del qubit da uno stato quantistico desiderato ad uno stato indesiderato, introducendo così errori sulle misure. Una delle principali cause di questo fenomeno, è data dal rumore elettronico che può provenire da fluttuazioni termiche o da effetti quantistici indesiderati nelle componenti del circuito.

1.5 IBM e Qiskit

Per programmare e realizzare la struttura logica di un sistema quantistico esistono diversi framework informatici (insieme di strumenti, librerie e moduli software utilizzati per sviluppare, testare e implementare applicazioni quantistiche). Uno dei framework più popolari per il quantum computing è Qiskit, sviluppato dalla IBM Quantum.

Qiskit è un framework di sviluppo open source basato sul linguaggio di programmazione Python, il quale fornisce agli sviluppatori uno strumento per creare applicazioni quantistiche che possono inoltre essere testate sugli hardware quantistici realizzati e messi a disposizione dell'utente da parte di IBM stessa, attraverso una piattaforma cloud chiamata IBM Quantum Experience [6]. Attraverso questa piattaforma, è possibile accedere in maniera del tutto libera ad una serie di dispositivi quantistici fisici di IBM, dei quali si possono conoscere le relative informazioni, tra cui in particolare, il numero di qubit, l'errore di misura, i tempi di decoerenza o l'accoppiamento tra i diversi qubit.

La piattaforma IBM Quantum Experience offre anche diversi tipi di simulatori quantistici, ognuno dei quali presenta diverse funzionalità e prestazioni. Ad esempio, il simulatore "`statevector simulator`" simula il comportamento di un circuito quantistico basato sulla rappresentazione vettoriale dello stato quantistico, tenendo traccia delle ampiezze di probabilità che gli stati possono avere durante tutto il processamento dell'algoritmo. Mentre il simulatore "`qasm simulator`", simula il comportamento di un circuito quantistico reale basandosi sul numero di misure per

ottenere una distribuzione di probabilità dei risultati, senza però conoscere gli stati vettoriali che può assumere il sistema.

Come vedremo in seguito, per lo studio di questa tesi non verrà fatto uso di alcun simulatore quantistico, in quanto le operazioni svolte consistono in composizioni di matrici numeriche e calcoli algebrici che sfruttano direttamente le risorse locali dell'hardware sul quale viene eseguito il codice. Nonostante ciò, sarà comunque possibile osservare il comportamento di un dispositivo fisico reale e metterlo a confronto con i risultati teorici, mediante l'utilizzo dei parametri quantistici ottimali ricavati attraverso gli algoritmi di ottimizzazione classici del codice stesso.

Capitolo 2

Risoluzione di EDO attraverso circuiti quantistici

In questo capitolo, illustreremo gli strumenti necessari nella la parte pratica di questa tesi, per cercare di risolvere le equazioni differenziali ordinarie, attraverso l'utilizzo di operazioni quantistiche, affiancate da algoritmi di ottimizzazione classici. In particolare, nella prima sezione forniremo una breve panoramica sulla teoria delle equazioni differenziali ordinarie e su come vengono risolte classicamente. Nella seconda sezione, illustreremo l'utilizzo di circuiti quantistici variazionali e gli algoritmi di minimizzazione classici che permetteranno di adattare il circuito approssimandolo alla soluzione desiderata. Infine, nella terza sezione, ci concentreremo su un metodo utile per rappresentare il gradiente di un sistema quantistico, con lo scopo di descrivere i termini differenziali di una funzione, attraverso sistemi di porte quantistiche.

2.1 Risoluzione classica di equazioni differenziali

Un'equazione differenziale ordinaria (EDO) è un tipo di equazione matematica che coinvolge una funzione incognita di una variabile indipendente e le sue derivate rispetto a quella variabile.

Questi tipi di equazioni costituiscono un tassello molto importante all'interno di diverse discipline scientifiche, in quanto rappresentano strumenti matematici utili a descrivere e modellare fenomeni che cambiano nel tempo. Attraverso l'uso di equazioni differenziali, è possibile analizzare e comprendere il comportamento di sistemi dinamici, effettuare previsioni a partire da condizioni iniziali e progettare soluzioni ottimali per problemi complessi.

Tuttavia risolvere un'equazione differenziale può essere un compito impegnativo e talvolta complesso. Mentre alcune equazioni differenziali possono essere risolte in modo analitico, ovvero ottenendo una soluzione espressa in termini di funzioni elementari come polinomi, esponenziali o trigonometriche, molte equazioni differenziali non hanno soluzione analitica nota e richiedono quindi l'uso di metodi approssimati

o numerici. È molto importante tenere presente che non esiste un metodo universale per risolvere tutte le equazioni differenziali. La scelta del metodo dipende dalla natura dell'equazione, dalle condizioni del problema e dagli obiettivi dell'analisi. Una delle famiglie di modelli di approssimazione più utilizzate nella risoluzione di equazioni differenziali, che vedremo nella sezione successiva, è quella relativa ai metodi di Runge-Kutta.

2.1.1 Metodo Runge Kutta

I metodi di Runge-Kutta [7] sono una famiglia di metodi iterativi basati su un approccio numerico classico utilizzato per risolvere approssimativamente le equazioni differenziali ordinarie con problemi a valori iniziali.

La tecnica più comune, al quale faremo riferimento, è il metodo di Runge-Kutta di quarto ordine (RK4). L'idea di base è quella di calcolare l'approssimazione del gradiente in quattro punti diversi all'interno di un intervallo $[t, t + h]$, in cui h rappresenta la larghezza dei sottointervalli (o passi) temporali su cui è stato suddiviso l'intervallo di interesse.

Considerando un'equazione differenziale ordinaria con condizioni iniziali:

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases} \quad (2.1)$$

L'approccio del metodo RK4, consiste nel valutare il termine a destra dell'equazione, sfruttando quattro termini a coefficienti pesati:

$$y_{t+1} = y_t + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h \quad (2.2)$$

$$\begin{aligned} k_1 &= f(t, y) \\ k_2 &= f\left(t + \frac{h}{2}, y + \frac{k_1 h}{2}\right) \\ k_3 &= f\left(t + \frac{h}{2}, y + \frac{k_2 h}{2}\right) \\ k_4 &= f(t + h, y + k_3 h) \end{aligned} \quad (2.3)$$

Il primo step, consiste nel ricavare sequenzialmente questi coefficienti, utilizzando le condizioni iniziali del problema, per poi trovare il corrispondente valore di y_{t+1} mediante l'eq.(2.2). A questo punto, in modo iterativo, si possono ripetere questi step con i nuovi valori ottenuti per i passi successivi.

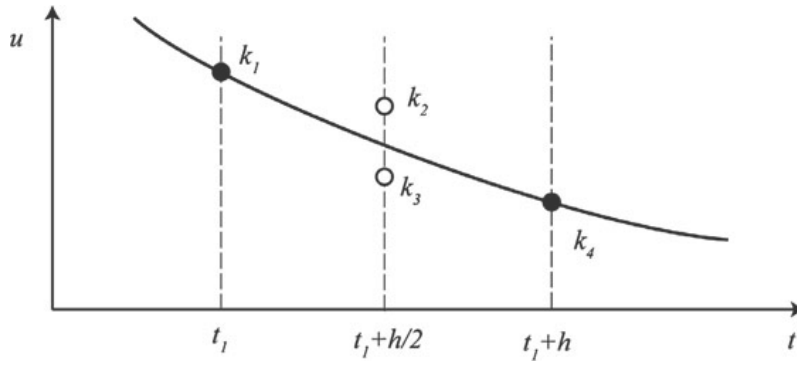


Figura 2.1: Rappresentazione grafica dei punti all'interno di un sottointervallo in cui vengono calcolati i coefficienti k [8]

In sostanza, invece di calcolare la derivata solo in un singolo punto, il metodo RK4 considera anche informazioni aggiuntive su come la funzione cambia lungo il passo considerato. Ciò significa che, tale approccio tiene conto, non solo della pendenza della funzione in un punto specifico bensì, considera anche le informazioni sulla pendenza nei punti intermedi. In questo modo, si migliora la precisione dell'approssimazione rispetto ad altri metodi numerici più semplici, come per esempio il metodo di Eulero [9].

2.1.2 Oscillatore Armonico Semplice

L'oscillatore armonico semplice è un sistema fisico ideale in cui si ha il moto periodico di un oggetto sottoposto ad una perturbazione iniziale che causa uno scostamento dall'equilibrio.

L'equazione differenziale che ne descrive il moto è scritta nella seguente forma:

$$\frac{d^2x}{dt^2} + \omega_0^2 x = 0 \quad (2.4)$$

dove ω_0 è la pulsazione o frequenza angolare del sistema.

La soluzione di questa equazione può essere facilmente ottenuta attraverso molteplici metodi analitici [10]. Una possibile forma con cui posso rappresentarla è la seguente:

$$x(t) = A \cos(\omega_0 t) + iB \sin(\omega_0 t) \quad (2.5)$$

dove A e B rappresentano delle costanti reali definibili imponendo le condizioni iniziali del problema. La soluzione "fisica" corrispondente è data dalla sola parte reale $A \cos(\omega_0 t)$:

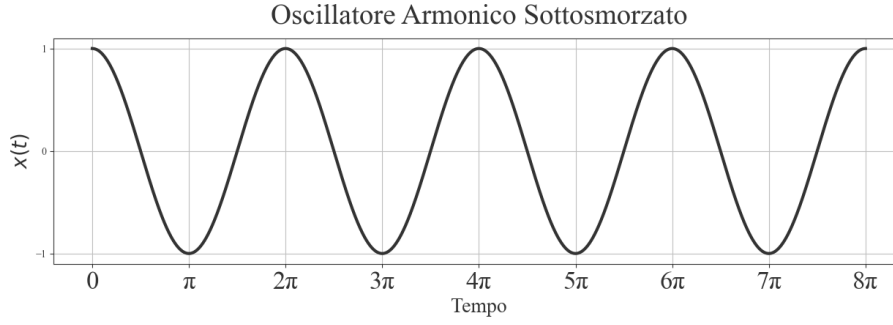


Figura 2.2: Soluzione dell'oscillatore armonico semplice con $\omega_0 = 1$, $x(0) = 1$ e $x'(0) = 0$.

2.1.3 Oscillatore Armonico Smorzato

L'oscillatore armonico smorzato rappresenta un'estensione del modello precedentemente accennato, con la differenza che il sistema in gioco è soggetto ad una forza esterna di smorzamento (come per esempio l'attrito) la cui presenza comporta un'attenuazione dell'ampiezza di oscillazione nel tempo, portando il sistema ad una condizione di equilibrio stabile.

L'equazione differenziale che ne descrive il moto può essere scritta nella seguente forma:

$$\frac{d^2x}{dt^2} + \gamma \frac{dx}{dt} + \omega_0^2 x = 0 \quad (2.6)$$

dove γ rappresenta il coefficiente di smorzamento.

Anche in questo caso ci ritroviamo ad avere un'equazione differenziale facilmente risolvibile analiticamente. Un modo per farlo è attraverso l'equazione caratteristica ricavabile dalla soluzione di prova $x = e^{\lambda t}$:

$$\lambda^2 e^{\lambda t} + \gamma \lambda e^{\lambda t} + \omega_0^2 e^{\lambda t} = 0 \quad \Rightarrow \quad \lambda^2 + \gamma \lambda + \omega_0^2 = 0 \quad (2.7)$$

Dall'equazione di secondo grado appena ottenuta, possiamo ricavare con molta semplicità le possibili soluzioni del problema:

$$\lambda_{\pm} = -\frac{\gamma}{2} \pm \sqrt{\frac{\gamma^2}{4} - \omega_0^2} \quad (2.8)$$

In base al valore dei coefficienti γ e ω_0 , il radicando distingue il problema in tre possibili regimi di smorzamento:

Sottosmorzato [$\gamma < 2\omega_0$]	$x(t) = e^{-\frac{\gamma}{2}t} \cdot [A \cos(\omega t + \varphi)]$
Criticamente smorzato [$\gamma = 2\omega_0$]	$x(t) = (C + Dt)e^{-\frac{\gamma}{2}t}$
Sovrasmorzato [$\gamma > 2\omega_0$]	$x(t) = e^{-\frac{\gamma}{2}t} \cdot [Fe^{\omega t} + Ge^{-\omega t}]$

Tabella 2.1: Leggi orarie dell'oscillatore armonico smorzato, distinti sui tre regimi.

con $\omega = \sqrt{\left|\omega_0^2 - \frac{\gamma^2}{4}\right|}$.

Nel caso di regime sottosmorzato, possiamo notare che il moto descritto mantiene la sua caratteristica oscillatoria con una pulsazione ω (come osservato nell'oscillatore armonico semplice), tuttavia la presenza anche del fattore esponenziale inverso, restituisce un andamento oscillatorio sempre più attenuato all'aumentare del tempo.

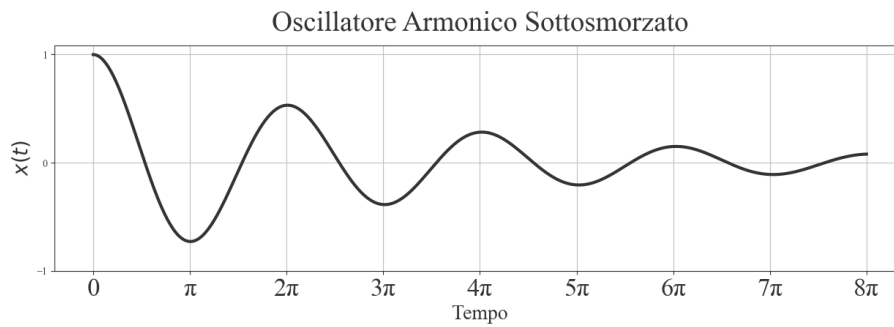


Figura 2.3: Soluzione dell'oscillatore armonico sottosmorzato con $\omega_0 = 1$, $\gamma = 0.2$, $x(0) = 1$ e $x'(0) = 0$.

Nel caso di regime criticamente smorzato, osserviamo un principio di oscillazione solamente iniziale, seguito da una decrescita esponenziale.

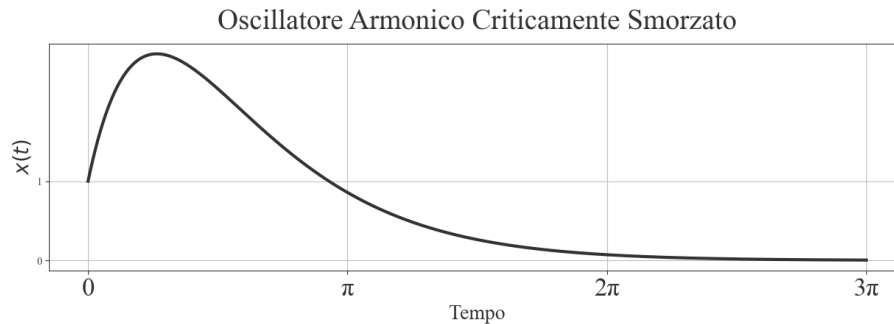


Figura 2.4: Soluzione dell'oscillatore armonico criticamente smorzato con $\omega_0 = 1$, $\gamma = 2$, $x(0) = 1$ e $x'(0) = 5$.

Infine, nel regime di sovrasmorzamento, la curva iniziale tende a restringersi e ad abbassarsi, facendo prevalere una forma esponenziale che, all'aumentare di γ , sarà portata ad attenuarsi sempre di più.

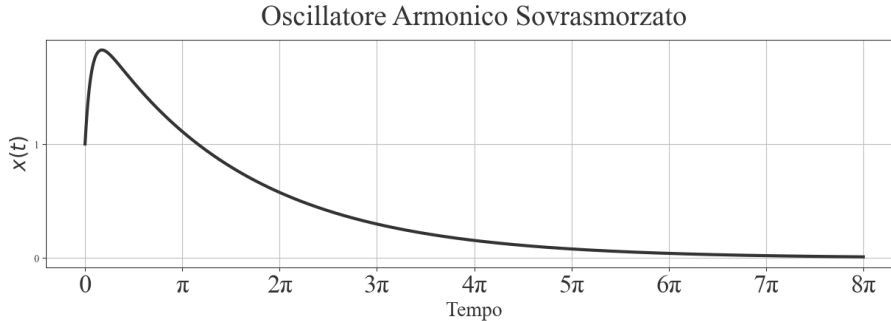


Figura 2.5: Soluzione dell'oscillatore armonico sovrasmorzato con $\omega_0 = 1$, $\gamma = 5$, $x(0) = 1$ e $x'(0) = 5$.

2.2 Ottimizzazione di circuiti quantistici variazionali

Un circuito quantistico variazionale è un tipo di circuito quantistico utilizzabile per algoritmi di ottimizzazione ed è costituito da una serie di porte quantistiche che dipendono da un set di parametri, anche chiamati "pesi" del circuito. Questi gradi di libertà permettono al circuito di adattarsi ed esprimere una varietà di stati quantistici differenti. La finalità di un circuito di questo tipo, è quella di trovare la combinazione ottimale di tali parametri in modo da minimizzare una funzione opportunamente definita, chiamata "funzione di costo", che quantifica l'accuratezza di una soluzione rispetto ad un obiettivo fissato. Per fare ciò, può essere utilizzato un algoritmo di ottimizzazione di tipo classico. Lo studio di tale algoritmo acquista notevole rilevanza in questo lavoro di tesi, in quanto la loro efficacia determina l'accuratezza nella soluzione del nostro problema.

Ad oggi, sono stati progettati molteplici algoritmi di ottimizzazione e in genere, la scelta di quale utilizzare, dipende principalmente dal contesto di studio o dal tipo di risposta che questi possono fornire a seconda del modello considerato. In particolare, Qiskit offre la possibilità di accedere ad una numerosa lista di algoritmi di ottimizzazione [11] e tra questi nello specifico, ne abbiamo selezionati due che hanno fornito un riscontro utile per i nostri studi: L'algoritmo COBYLA e l'algoritmo ADAM. A seguire proveremo a darne qualche accenno.

2.2.1 COBYLA

L'algoritmo COBYLA, acronimo di Constrained Optimization BY Linear Approximation, è un metodo di ottimizzazione numerica per problemi vincolati, che sfrutta la teoria della *programmazione lineare* [12]. Nello specifico, la programmazione lineare (PL) è un metodo per la ricerca del miglior risultato in un modello matematico,

rappresentabile mediante una relazione lineare (detta funzione obiettivo). Più formalmente, la PL è una tecnica di ottimizzazione in cui la funzione da massimizzare (o minimizzare), è lineare e i vincoli sono descritti da equazioni e/o disequazioni lineari:

$$\text{(Funzione obiettivo)} \quad f(x_1, \dots, x_n) = c_1 x_1 + \dots + c_n x_n = c^\top x \quad (2.9)$$

$$\text{(Vincoli)} \quad \left. \begin{array}{ccc} a_{11}x_1 + \dots + a_{1n}x_n \leq b_1 \\ \vdots \\ a_{m1}x_1 + \dots + a_{mn}x_n \leq b_m \end{array} \right\} \equiv Ax \leq b \quad (2.10)$$

o in maniera più compatta:

$$\begin{array}{l} \max \\ \text{o min} \end{array} \{c^\top x \mid x \in \mathbb{R}^n \wedge Ax \leq b \wedge x \geq 0\} \quad (2.11)$$

In queste relazioni, x rappresenta il vettore delle variabili da determinare e l'insieme delle soluzioni che soddisfano simultaneamente (2.9) e (2.10), viene chiamato *regione ammissibile* (o feasible region). L'idea di base, consiste nel cercare all'interno di tale regione, un vettore ottimale \bar{x} tale per cui $c^\top \bar{x} \geq c^\top x, \forall x_i \geq 0$ (o col segno \leq in caso di minimizzazione). A livello grafico, possiamo visualizzare i vincoli come gli oggetti necessari per delimitare le frontiere di un politopo convesso¹ n-dimensionale, il quale rappresenta la regione ammissibile (compresa la frontiera) su cui deve essere ottimizzata la funzione obiettivo. Esistono diversi teoremi legati alla formulazione del *problema primale standard* [13], molto utili a semplificare il metodo di ricerca dei valori ottimali associati al problema considerato.

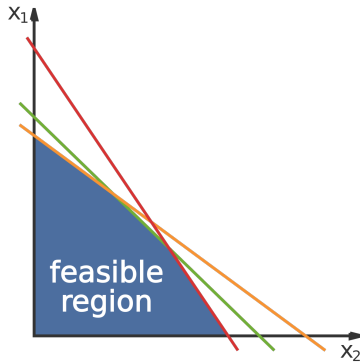


Figura 2.6: Rappresentazione della regione ammissibile, ottenuta mediante tre vincoli, i quali definiscono i bordi del politopo bidimensionale.

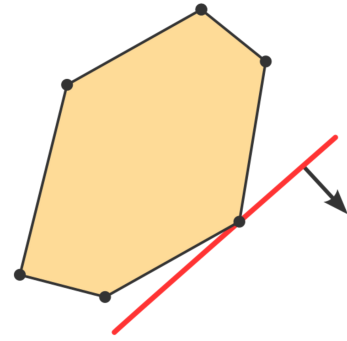


Figura 2.7: Rappresentazione della regione ammissibile di un sistema a sei vincoli in due dimensioni. Il set di variabili ottimali corrisponde alle coordinate di uno dei vertici del politopo ed è il punto in cui la funzione obiettivo (retta rossa), interseca la regione ammissibile.

¹Un politopo è la generalizzazione del poliedro tridimensionale a uno spazio euclideo reale di dimensione generica.

Grazie a queste considerazioni, è stato ideato un metodo numerico per la risoluzione dei problemi di programmazione lineare, chiamato *algoritmo del simplesso* [12]. Sostanzialmente, questo algoritmo consiste in una ricerca della soluzione ottimale partendo da uno dei vertici del politopo, per poi spostarsi lungo la frontiera in direzione di altri vertici su cui la funzione obiettivo abbia valori non decrescenti (o decrescenti in caso di minimizzazione). Tale strategia di ricerca, si basa sul fatto che, nel caso di un problema di programmazione lineare, la soluzione ottimale si trova sempre su uno dei vertici del politopo [12, 13]. Tuttavia, il metodo del simplesso presenta alcune limitazioni quando si tratta di problemi più complessi con funzioni obiettivo o vincoli non lineari. In questi casi, l'approccio puramente lineare del simplesso potrebbe non essere sufficiente per modellare accuratamente le relazioni tra le variabili e ottenere una soluzione ottimale.

L'algoritmo COBYLA svolge un ruolo fondamentale come alternativa valida per l'ottimizzazione dei problemi non lineari. Si tratta di un metodo che non richiede il calcolo esplicito delle derivate della funzione obiettivo o dei vincoli, bensì si basa sull'approssimazione del problema originale utilizzando la programmazione lineare. Ad ogni iterazione, COBYLA risolve un problema di tipo PL approssimato, con il fine di fornire una soluzione che si avvicini il più possibile al problema non lineare originale. In questo modo, ad ogni iterazione, la soluzione ottenuta può essere valutata attraverso la funzione obiettivo e i vincoli del problema non lineare, così da stabilire se questa, possa essere un buon candidato per guidare ulteriormente la ricerca verso una convergenza ottimale. L'idea chiave è che, anche se il problema lineare approssimato non rappresenti in modo esatto il problema originale, esso può comunque fornire indicazioni utili sulla direzione da seguire per trovare la soluzione ottimale [14].

2.2.2 Adam

Per capire come funzionino l'algoritmo Adam (Adaptive moment estimation) è bene introdurre i due principali metodi dal quale esso deriva: il Gradient Descent (discesa del gradiente) [15] e lo Stochastic Gradient Descent (discesa stocastica del gradiente) [16].

Il primo (GD), è un algoritmo di ottimizzazione iterativo ampiamente utilizzato per la ricerca del minimo di una funzione obiettivo. Esso si basa sull'idea di calcolare il gradiente di tale funzione rispetto ai parametri del modello e aggiornare iterativamente questi parametri, in direzione opposta al gradiente stesso, in modo da spostarsi verso un punto di minimo. Questo processo viene ripetuto fino al raggiungimento di una condizione di convergenza desiderata, in cui l'errore tra i risultati previsti e quelli effettivi, sia il più piccolo possibile. Tuttavia, ciò può risultare computazionalmente costoso quando si lavora con un set di dati molto grande, poiché l'aggiornamento dei parametri avviene solamente dopo aver campionato l'intero dataset considerato e dunque, il tempo di calcolo richiesto potrebbe diventare eccessivo anche per una singola iterazione.

A tal proposito, si può pensare in alternativa di scomporre il set di dati in sottoinsiemi più piccoli e campionarli stocasticamente in modo da stimare il valore del gradiente e aggiornarlo più rapidamente ad ogni singola iterazione. Questa tecnica è cono-

sciuta come Stochastic Gradient Descent (SGD) e si basa sull'idea che la stima del gradiente, ottenuta da un sottoinsieme rappresentativo dei dati, sia sufficientemente approssimata rispetto al gradiente calcolato sull'intero dataset.

Tuttavia, è importante notare che, l'utilizzo della SGD può comportare un aumento delle fluttuazioni durante il processo di aggiornamento dei pesi del modello. Questo, è dovuto al fatto che il gradiente viene stimato utilizzando solo un sottoinsieme dei dati, il che introduce una certa variabilità nella direzione del gradiente calcolato e di conseguenza, tali fluttuazioni possono influire sulla stabilità e sulla convergenza dell'algoritmo.

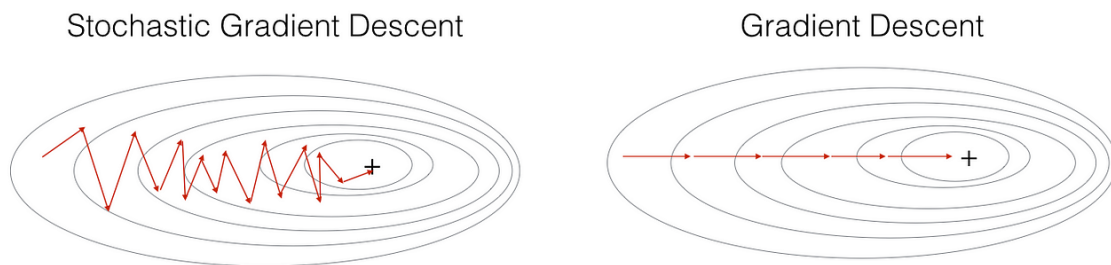


Figura 2.8: Confronto tra i due metodi, SGD e DG, illustrando in particolare il comportamento oscillante del primo. [17]

Un possibile metodo che aiuta a ridurre il rumore e ad accelerare la convergenza, consiste nell'aggiunta del *momento* (SGD con momento), ossia un oggetto in grado di modificare il peso dei parametri in relazione alla media ponderata dei gradienti ottenuti nelle iterazioni precedenti. In pratica, l'effetto del momento è quello di tenere traccia delle direzioni in cui i gradienti hanno puntato, in modo tale da influenzare gli aggiornamenti dei pesi correnti e accelerare di conseguenza il processo di convergenza verso un punto ottimale.

Infine, un altro concetto fondamentale da introdurre, utilizzato da entrambe le tecniche appena descritte, è il *learning rate* (tasso di apprendimento), ossia un parametro di regolazione che determina quanto i parametri del modello vengono modificati ad ogni iterazione dell'algoritmo di ottimizzazione.

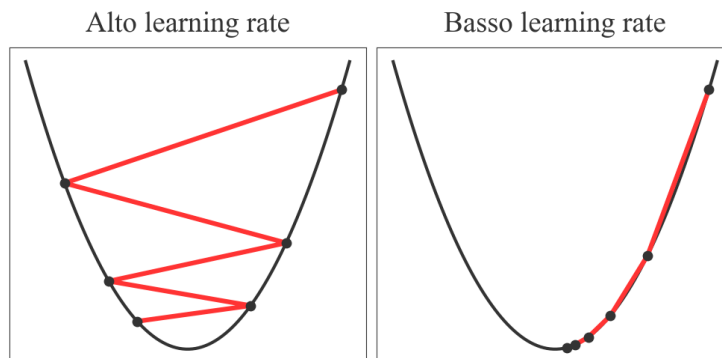


Figura 2.9: Confronto del comportamento di ricerca del minimo a due diversi valori di learning rate.

Un learning rate elevato implica passi più grandi, che possono portare ad un apprendimento rapido ma anche a oscillazioni e salti eccessivi nella ricerca del punto ottimale. D'altra parte, un learning rate troppo basso può rallentare l'apprendimento, portando a una convergenza lenta o a rimanere bloccati in minimi locali.

Nelle tecniche GD e SGD, la scelta del learning rate avviene prima dell'esecuzione dell'algoritmo. Per tale ragione, trovare un valore adeguato, che permetta di ottenere i migliori risultati, può richiedere diversi test con differenti learning rate.

Esistono anche tecniche di adattamento del learning rate, le quali cercano di gestire dinamicamente il tasso di apprendimento durante il processo di ottimizzazione. Un esempio, è proprio l'algoritmo Adam, il quale adatta il learning rate per ciascun parametro in base ai momenti del gradiente stimati, consentendo così, una convergenza più stabile ed efficiente.

Adam rappresenta un'estensione delle caratteristiche della SGD con momento, poiché tiene conto della media mobile del primo e del secondo momento del gradiente. Più precisamente, il primo momento corrisponde ad una media mobile dei gradienti calcolati durante l'ottimizzazione, rappresentando la direzione media degli aggiornamenti, mentre il secondo momento corrisponde ad una media mobile dei gradienti al quadrato, rappresentando la varianza degli aggiornamenti. Queste caratteristiche consentono ad Adam di gestire efficacemente funzioni obiettivo non stazionarie e gradienti rumorosi. Inoltre, tale algoritmo richiede limitate risorse hardware ed il suo funzionamento è invariante rispetto a dilatazioni del gradiente [18].

2.2.3 Funzione di costo

Per cercare di risolvere un'equazione differenziale, è necessario valutare un metodo per quantificare quanto bene la funzione di prova, ottenuta dal circuito quantistico variazionale (come verrà spiegato nella sezione 3.1), si adatti a rappresentare la soluzione del problema considerato. Per tale scopo si definisce una *funzione di costo* (analoga alla funzione obiettivo, menzionata nella sezione 2.2.1), ossia una misura della discrepanza tra i risultati previsti da un certo modello (nel nostro caso dal circuito quantistico variazionale con i relativi parametri in ingresso) e i risultati effettivi. L'obiettivo è quello di minimizzare questa discrepanza con lo scopo di migliorare l'accuratezza del modello studiato.

In merito al nostro problema, la funzione di costo verrà valutata in un certo intervallo di tempo e può essere descritta mediante due contributi: Uno relativo al differenziale ($\mathcal{L}^{(\text{diff})}$), e l'altro relativo alle sue condizioni iniziali ($\mathcal{L}^{(\text{init})}$):

$$\mathcal{L}[\partial_t^2 f, \partial_t f, f, t] = \mathcal{L}^{(\text{diff})}[\partial_t^2 f, \partial_t f, f, t] + \mathcal{L}^{(\text{init})}[f, t] \quad (2.12)$$

Il contributo differenziale, assume la seguente forma:

$$\mathcal{L}^{(\text{diff})}[\partial_t^2 f, \partial_t f, f, t] = \frac{1}{M} \sum_{i=1}^M L(F[\partial_t^2 f(t_i), \partial_t f(t_i), f(t_i), t_i], 0) \quad (2.13)$$

in cui M rappresenta il numero di punti dell'intervallo preso in considerazione, la

funzione F corrisponde all'equazione differenziale scritta nella forma $F = a \cdot \partial_t^2 f(t) + b \cdot \partial_t f(t) + c \cdot f(t) + d \cdot t + e = 0$, mentre la funzione $L(a, b)$, descrive come viene misurata la distanza tra i due argomenti a e b (che vedremo a breve). La funzione $f(t)$ sarà approssimata da un circuito quantistico secondo la metodologia illustrata nella sezione 3.1. Di conseguenza, tanto più quest'ultima si avvicinerà ad essere la soluzione dell'equazione differenziale, tanto più $\mathcal{L}^{(\text{diff})}$ si avvicinerà a zero.

Il secondo contributo tiene invece conto sia della distanza tra il valore della funzione al tempo iniziale t_0 e il valore noto $u_0 = f(0)$ corrispondente, sia della distanza tra il valore della sua derivata valutata in t_0 e il valore noto v_0 corrispondente:

$$\mathcal{L}^{(\text{init})}[f, t] = \eta \cdot [L(f(t_0), u_0) + L(\partial_t f(t_0), v_0)] \quad (2.14)$$

dove abbiamo introdotto anche η , che denomineremo coefficiente di ancoraggio, che controlla il peso del termine associato alle condizioni iniziali, nella procedura di ottimizzazione. In particolare, un valore maggiore di uno ($\eta > 1$), favorisce l'imposizione delle condizioni iniziali, in quanto avranno un peso significativo sulla funzione di costo complessiva.

I tipi di distanze metriche L sui quali ci siamo soffermati per la costruzione della funzione di costo, sono stati, *l'errore quadratico medio* (Mean Squared Error, MSE) e *l'errore medio assoluto* (Mean Absolutated Error, MAE).

La funzione di costo con la MSE, viene calcolata prendendo la media dei quadrati delle differenze tra i valori previsti e i valori effettivi:

$$L(a, b) = (a - b)^2 \quad (2.15)$$

In particolare, il peso con cui viene valutato l'errore, risulta maggiore per grandi distanze e meno significativo per distanze più piccole.

Per quanto riguarda invece la metrica MAE, la funzione di costo viene calcolata mediante la media delle differenze assolute tra i valori previsti e i valori effettivi:

$$L(a, b) = |a - b| \quad (2.16)$$

In questo caso, la convergenza potrebbe risultare più lenta. Tuttavia, una volta vicini alla soluzione ottimale, si può raggiungere una precisione maggiore rispetto alla MSE.

Poiché la scelta della metrica determina come l'algoritmo di ottimizzazione valuta la distanza tra i vettori e, influenzando sulla convergenza, è opportuno effettuare valutazioni appropriate basate sul contesto e sugli obiettivi del problema, oppure seguendo un approccio sperimentale [19].

2.3 Qiskit Gradient e Parameter-shift Rule

Il Gradient Framework [20] di Qiskit è una libreria progettata per calcolare gradienti utilizzando circuiti quantistici variazionali. La classe corrispondente, utilizzabile in Python, prende il nome di `Gradient()` e supporta diversi metodi per il calcolo del gradiente.

Il ruolo della classe Gradient consiste nella costruzione di un insieme di circuiti quantistici che vengano utilizzati per approssimare il valore del gradiente associato al circuito di input, considerando i relativi parametri sul quale si desidera effettuare la derivazione.

Per costruire un circuito quantistico in grado di descrivere un'equazione differenziale desiderata, abbiamo sfruttato il metodo della Parameter-shift Rule, il quale ci ha permesso di tradurre ciascun termine differenziale in una forma adatta per l'elaborazione quantistica e per la realizzazione della funzione di costo (illustrata nella sezione precedente), necessaria per l'ottimizzazione dei parametri quantistici, e di conseguenza, per la ricerca della soluzione ottimale dell'equazione differenziale considerata.

Nella prossima sezione proviamo quindi ad esaminare alcuni aspetti teorici del metodo appena accennato.

2.3.1 Parameter-shift Rule

La Parameter-shift Rule è una tecnica utilizzata per la misurazione del gradiente di circuiti quantistici parametrizzati. Il vantaggio di questo approccio è dato dal fatto di poter esprimere la derivata della funzione quantistica $f(t)$ come una combinazione lineare di due funzioni quantistiche corrispondenti alla funzione di partenza.

Consideriamo per i nostri studi una funzione $f(t, \theta)$ con t , parametro dinamico rappresentate il tempo, e θ , set di parametri opportunamente ottimizzati secondo i metodi descritti nella sezione 2.2. Tale funzione potrà essere definita come il valore di aspettazione di un'osservabile rispetto allo stato del sistema modificato dal circuito quantistico (omettiamo i parametri θ , in quanto non sono necessari per spiegare la tecnica della Parameter-shift Rule):

$$f(t) = \langle \psi | \hat{A} | \psi \rangle = \langle 0 | U_G^\dagger(t) \hat{A} U_G(t) | 0 \rangle \quad (2.17)$$

dove \hat{A} è l'osservabile, mentre il gate parametrizzato² $U_G(t)$ consiste in una matrice unitaria di singolo qubit della forma:

$$U_G(t) = e^{-iGt} \quad (2.18)$$

In cui G è il generatore Hermitiano associato, il quale viene definito con al più due

²Nei seguenti passaggi considereremo per semplicità un circuito quantistico composto da un solo gate, tuttavia si può dimostrare facilmente che la trattazione a più gate, risulta del tutto analoga a questa (con qualche conto in più di derivazione).

autovalori, $\pm r$ (il motivo sarà chiaro a breve).

Andando a derivare $f(t)$ rispetto al tempo, otteniamo la seguente espressione:

$$\partial_t f(t) = \langle \psi | \partial_t U_G^\dagger(t) \hat{A} U_G(t) | \psi \rangle + \langle \psi | U_G^\dagger(t) \hat{A} \partial_t U_G(t) | \psi \rangle \quad (2.19)$$

Dove il primo termine risulta essere proprio l'hermitiano coniugato (*h.c.*) del secondo. Sapendo inoltre che $\partial_t U_G(t) = -iG e^{-iGt} = -iG \cdot U_G(t)$, possiamo riscrivere per comodità $|\psi'\rangle := U_G(t) |\psi\rangle$, in modo da lavorare con un'espressione più compatta:

$$\partial_t f(t) = \langle \psi' | \hat{A}(-iG) | \psi' \rangle + h.c. \quad (2.20)$$

Per una qualunque coppia \hat{B} e \hat{C} di operatori, abbiamo che:

$$\begin{aligned} \langle \psi | \hat{B}^\dagger \hat{A} \hat{C} | \psi \rangle + h.c. = & \frac{1}{2} \left[\langle \psi | (\hat{B} + \hat{C})^\dagger \hat{A} (\hat{B} + \hat{C}) | \psi \rangle + \right. \\ & \left. - \langle \psi | (\hat{B} - \hat{C})^\dagger \hat{A} (\hat{B} - \hat{C}) | \psi \rangle \right] \end{aligned} \quad (2.21)$$

Considerando allora $\hat{B} = \mathbb{1}$ e $\hat{C} = -iG/r$, possiamo riscrivere l'eq.(2.20) come:

$$\begin{aligned} \partial_t f(t) = & \frac{r}{2} \left[\langle \psi' | (\mathbb{1} - iGr^{-1})^\dagger \hat{A} (\mathbb{1} - iGr^{-1}) | \psi' \rangle + \right. \\ & \left. - \langle \psi' | (\mathbb{1} + iGr^{-1})^\dagger \hat{A} (\mathbb{1} + iGr^{-1}) | \psi' \rangle \right] \end{aligned} \quad (2.22)$$

Teorema. *Se il generatore Hermitiano G dell'operatore unitario $U_G(t) = e^{-iGt}$ ha al più due autovalori $\pm r$, allora vale la seguente identità:*

$$U_G \left(\pm \frac{\pi}{4r} \right) = \frac{1}{\sqrt{2}} (\mathbb{1} \mp iGr^{-1}) \quad (2.23)$$

Dimostrazione. Attraverso gli sviluppi in serie di Taylor, possiamo riscrivere l'operatore unitario come:

$$\begin{aligned} U_G(t) = e^{-iGt} &= \sum_{k=0}^{\infty} \frac{(-it)^k \cdot G^k}{k!} = \\ &= \sum_{k=0}^{\infty} \frac{(-it)^{2k} \cdot G^{2k}}{(2k)!} + \sum_{k=0}^{\infty} \frac{(-it)^{2k+1} \cdot G^{2k+1}}{(2k+1)!} \end{aligned} \quad (2.24)$$

Poiché il generatore G è stato definito con uno spettro $\{\pm r\}$, vale la relazione $G^2 = r^2 \cdot \mathbb{1}$ e di conseguenza abbiamo che, $G^{2k} = r^{2k} \cdot \mathbb{1}$:

$$\begin{aligned} U_G(t) &= \sum_{k=0}^{\infty} \frac{(-1)^k (t \cdot r)^{2k}}{(2k)!} \cdot \mathbb{1} - \sum_{k=0}^{\infty} \frac{(-1)^k (t \cdot r)^{2k+1}}{(2k+1)!} \cdot iGr^{-1} = \\ &= \mathbb{1} \cdot \cos(t \cdot r) - \frac{iG}{r} \cdot \sin(t \cdot r) \end{aligned} \quad (2.25)$$

Possiamo a questo punto osservare che, per $t = \pm \frac{\pi}{4r}$, si ottiene la relazione (2.23). \square

Sfruttando tale identità all'interno dell'eq.(2.22), otteniamo:

$$\begin{aligned} \partial_t f(t) = r \Big[& \langle \psi' | U_G^\dagger \left(\frac{\pi}{4r} \right) \hat{A} U_G \left(\frac{\pi}{4r} \right) | \psi' \rangle + \\ & - \langle \psi' | U_G^\dagger \left(-\frac{\pi}{4r} \right) \hat{A} U_G \left(-\frac{\pi}{4r} \right) | \psi' \rangle \Big] \end{aligned} \quad (2.26)$$

e tenendo presente che $U_G(a)U_G(b) = U_G(a+b)$, possiamo riscrivere $|\psi'\rangle$ come in origine:

$$\begin{aligned} \partial_t f(t) = r \Big[& \langle \psi | U_G^\dagger \left(t + \frac{\pi}{4r} \right) \hat{A} U_G \left(t + \frac{\pi}{4r} \right) | \psi \rangle + \\ & - \langle \psi | U_G^\dagger \left(t - \frac{\pi}{4r} \right) \hat{A} U_G \left(t - \frac{\pi}{4r} \right) | \psi \rangle \Big] \end{aligned} \quad (2.27)$$

Da cui ritroviamo la forma di f per ciascun valore di aspettazione, ma rispettivamente con uno sfasamento di $\pm \pi/4r$:

$$\partial_t f(t) = r \left[f \left(t + \frac{\pi}{4r} \right) - f \left(t - \frac{\pi}{4r} \right) \right] \quad (2.28)$$

Il valore della costante r dipende dalla parametrizzazione. Per esempio, Se stiamo considerando le porte rotazionali 1.3.3, avremo $r = \frac{1}{2}$ (si veda la forma esponenziale delle porte rotazionali R_n (1.17)) [21, 22].

La trattazione appena discussa, è stata valutata considerando le porte quantistiche a singolo qubit. Tuttavia, è possibile comportarsi in maniera analoga anche per i gate a più qubit, ricorrendo alla decomposizione³ per poter trattare degli oggetti più semplici (si vedano i circuiti 2.2, 2.3, 2.4).

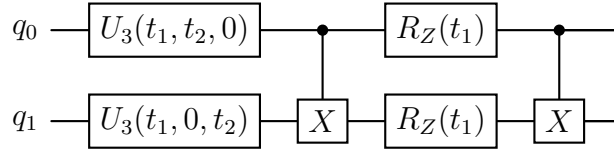
Osservazione: Sia dato un circuito quantistico variazionale ad n qubit. Definiamo N_P , come il numero di porte quantistiche a singolo qubit presenti all'interno del circuito e p_i , come il numero di parametri dal quale dipende l' i -esimo gate. Se le porte parametriche sono tutte a singolo qubit, si può osservare, mediante passaggi analoghi a quelli visti finora, che il numero di circuiti quantistici necessari per descrivere il gradiente attraverso la Parameter-shift Rule⁴, corrisponde a $N_C = \sum_{i=1}^{N_P} 2p_i$, ciascuno con N_P porte al suo interno.

Supponiamo per esempio, di aver un circuito quantistico a 2 qubit costruito nel

³La decomposizione di un gate, è il processo di scomporre un operatore quantistico complesso in una sequenza di operatori elementari, in modo da semplificare la sua implementazione e controllo su un sistema quantistico.

⁴Ogni valore di aspettazione che compone la relazione del gradiente, corrisponde ad un circuito quantistico.

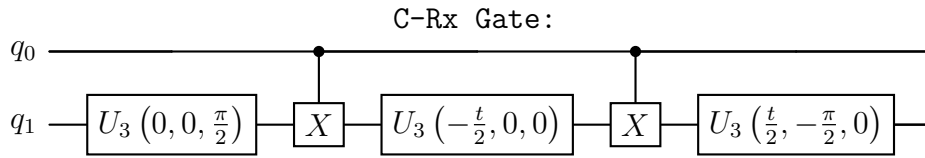
seguinte modo:



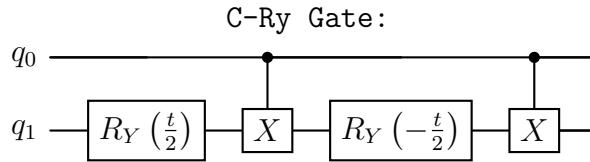
Circuito 2.1: Rappresentazione a blocchi di un generico circuito quantistico variazionale a 2 qubit

Il numero di gate utilizzati, è $N_P = 6$, ciascuno con un numero differente di parametri. Svolgendo i conti, si ottiene che il numero di circuiti necessari per descrivere il gradiente del circuito di partenza, corrisponde a $N_C = 12$.

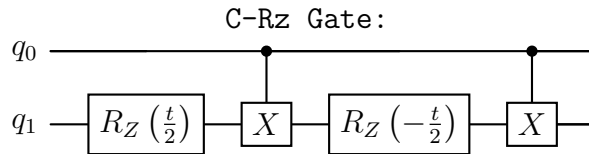
Introducendo inoltre le porte parametriche a 2 qubit, **Controlled-Rn Gate**, si osserva un maggiore incremento di N_C , il quale si può spiegare attraverso la decomposizione di tali porte in una serie di gate più semplici:



Circuito 2.2: Decomposizione del Controlled-Rx Gate svolta da Qiskit



Circuito 2.3: Decomposizione del Controlled-Ry Gate svolta da Qiskit



Circuito 2.4: Decomposizione del Controlled-Rz Gate svolta da Qiskit

È importante sottolineare ciò, in quanto l'aumento del numero di porte o parametri, comporta un incremento della complessità computazionale per descrivere il gradiente, e di conseguenza un processamento, in termini di tempo, sempre più dispendioso. Tale complessità cresce ulteriormente nel momento in cui viene considerato anche il gradiente di secondo ordine (ricavabile attraverso l'applicazione della Parameter-shift Rule sul gradiente precedentemente costruito).

Capitolo 3

Implementazione dell'algoritmo e analisi dei risultati

3.1 Descrizione dell'algoritmo realizzato

3.1.1 Circuito quantistico variazionale (Ansatz)

La scelta di configurazione dei gate da utilizzare in un circuito quantistico, che sia capace di descrivere il comportamento di una funzione desiderata, non è una procedura semplice, poiché ad oggi non esistono criteri deterministici che ci permettano di fare distinzioni accurate su quale circuito possa essere il migliore. Tuttavia, è comunque possibile fare alcune considerazioni iniziali da cui partire.

Il primo aspetto fondamentale sul quale prestare attenzione è il numero di qubit. Attualmente, nel campo di ricerca sul quantum computing, si tende a sviluppare modelli scalabili, che possano comportarsi in maniera efficiente, soprattutto per un basso numero di qubit. Ciò, è dovuto al fatto che le risorse fisiche disponibili attualmente, sono ancora molto limitate. Inoltre, come abbiamo visto nelle sezioni precedenti, l'approccio che adotteremo per la ricerca dei migliori parametri quantistici, consiste nell'utilizzo di algoritmi di ottimizzazione classica. Pertanto, la dipendenza dalle risorse locali disponibili sul calcolatore classico, rappresenta una notevole limitazione sulla possibilità di elaborare sistemi quantistici di diversa complessità, sia in termini del numero di qubit implementabili, che in termini di numero e tipo di gate utilizzabili per costruire un circuito quantistico.

Un'ulteriore considerazione che si riallaccia alla complessità del circuito, riguarda il numero di gate parametrici nel sistema. Come prima cosa però, è necessario fare una distinzione fra i tipi di parametri in gioco: Il parametro t , che descriveremo attraverso il metodo `Parametr("t")` di Qiskit, viene utilizzato per introdurre una dipendenza temporale nel circuito quantistico. La sua utilità è proprio quella di applicare delle variazioni al circuito in funzione del tempo, in modo da confrontare i suoi risultati con quelli attesi da una funzione reale $f(t)$ presa in considerazione.

L'insieme dei parametri θ , definiti invece dal metodo `ParameterVector("theta", n_par)`, vengono utilizzati per descrivere e “modellare” l'output del circuito. Attra-

verso gli algoritmi di ottimizzazione, questi parametri verranno modificati affinché, il circuito quantistico associato, possa approssimare nel modo più accurato possibile la soluzione numerica reale.

Entrambe le tipologie di parametri appena descritte, svolgono un ruolo significativo nella complessità computazionale del sistema. In particolare, il parametro temporale ha un impatto determinante sulla quantità di circuiti da calcolare, generati tramite la Parameter-shift Rule (si veda la sezione 2.3.1). Pertanto, le operazioni svolte sui circuiti quantistici, richiederanno un maggior numero di risorse computazionali, all'aumentare del numero di gate dipendenti dal tempo. Per quanto riguarda invece i parametri θ , essi influiscono direttamente sul costo computazionale degli algoritmi di ottimizzazione, poiché il tempo di elaborazione di tali algoritmi dipende principalmente dal numero di parametri da ottimizzare ricevuti in ingresso.

Lo stato descritto dal circuito quantistico applicato su un sistema di qubit, può essere espresso nella seguente forma:

$$|f_{t,\theta}\rangle = |\psi(t, \theta)\rangle = U(t, \theta) |0\rangle \quad (3.1)$$

La realizzazione di un circuito mediante Qiskit, avviene come nel seguente esempio:

```

1 ansatz = QuantumCircuit(2)
2 ansatz.ry(t*theta[0], 0)
3 ansatz.ry(t*theta[1], 1)
4 ansatz.rx(t*theta[2], 0)
5 ansatz.rx(t*theta[3], 1)
```

Codice 3.1: Rappresentazione di un generico circuito quantistico variazionale a 2 qubit.

In questo esempio sono state utilizzate delle porte rotazionali, i cui argomenti rappresentano rispettivamente, l'angolo di rotazione e il qubit sul quale applicare il gate.

3.1.2 Osservabile Hermitiana e valore di aspettazione

Una volta costruito il circuito quantistico, è necessario definire un'osservabile Hermitiana con il quale calcolare i valori di aspettazione dello stato $|f_{t,\theta}\rangle$, all'interno di intervallo temporale considerato. Anche in questo caso, la scelta del tipo di osservabile, non ha alcun criterio ben definito, se non quello di rispettare l'hermitianità (in quanto si sta lavorando su quantità reali). Alla riga 1 del seguente segmento di codice, si può leggere un esempio di osservabile Hermitiana rappresentata con Qiskit.

```

1 observable = Z^Z
2
3 # -----
4
5 op = ~StateFn(observable) @ CircuitStateFn(primitive = ansatz)

```

Codice 3.2: Rappresentazione di una generica osservabile Hermitiana a 2 qubit (riga 1).
Composizione tra osservabile e stato del circuito (riga 2).

mentre alla riga 5 è rappresentata l'applicazione dell'osservabile sullo stato $|\psi(t, \theta)\rangle$. In particolare, `StateFn` e `CircuitStateFn` rappresentano delle funzioni di Qiskit che convertono rispettivamente l'osservabile e il circuito quantistico in una forma compatibile, affinché questi si possano combinare fra loro. Il simbolo di tilde (\sim), applicato all'osservabile quantistica, permette di effettuare una misura sullo stato del circuito quantistico considerato. Mentre il simbolo di chiocciola ($@$), permette di effettuare il prodotto tensoriale tra l'osservabile e lo stato del circuito.

Con questi strumenti, è possibile dunque definire il valore di aspettazione nel seguente modo:

$$f(\theta, t) = \langle \psi(t, \theta) | \hat{O} | \psi(t, \theta) \rangle \quad (3.2)$$

Supponiamo di voler calcolare per ogni punto di in un certo intervallo di tempo $[0, t_0]$, il valore di aspettazione di un'osservabile su un circuito quantistico, di cui i parametri θ sono assegnati. I risultati ottenuti descriveranno una curva di valori reali il cui comportamento è influenzato dalla scelta dei parametri quantistici assegnati al circuito.

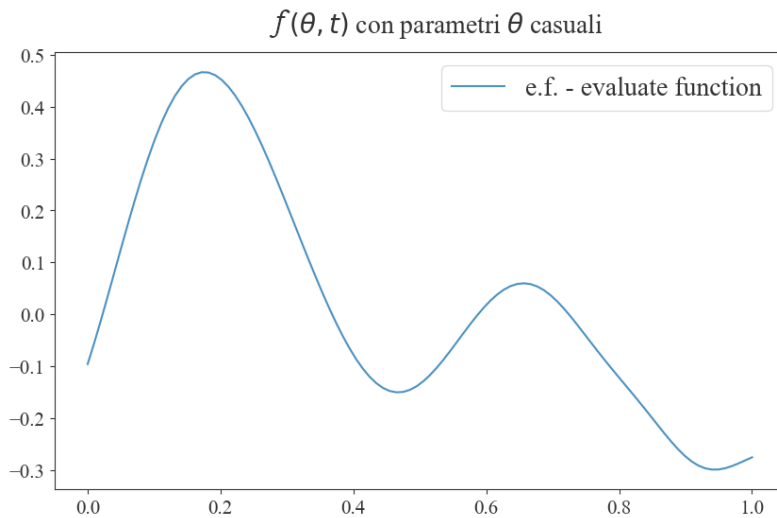


Figura 3.1: Andamento dei valori reali di un circuito quantistico variazionale con parametri fissati e al variare del tempo.

Ricavando gli opportuni parametri attraverso metodi di minimizzazione della funzione di costo, come spiegato nella sezione 2.2, è possibile approssimare tale curva alla soluzione reale di una EDO presa in considerazione.

Il segmento di codice realizzato per elaborare un valore di aspettazione, è descritto mediante la seguente funzione:

```

1 def evaluate_function(time: Parameter, weights: ParameterVector, operator:
  ↳ Union[ComposedOp, SummedOp]) -> np.float64:
2     """Returning the expectation value of a circuit"""
3     expval = operator.assign_parameters({t:time, theta: weights}).eval()
4     if np.abs(expval.imag) > 1e-5:
5         print('Expectation value has imaginary part. Check Hermitianity')
6     return expval.real

```

Codice 3.3: Funzione `evaluate_function`, utilizzata per il calcolo del valore di aspettazione.

La quale riceve come argomenti, il parametro temporale (`time`), il vettore dei parametri θ (`weight`) e l'operatore descritto dall'applicazione dell'osservabile sullo stato quantistico del sistema (`operator`), ossia $\hat{O}|\psi(t, \theta)\rangle$.

Grazie ai metodi, `assign_parameters()` ed `eval()` dell'oggetto `operator`, è possibile rispettivamente assegnare i valori ai parametri del circuito e calcolarne successivamente il valore di aspettazione.

Inoltre, è stata inclusa una condizione per verificare la presenza di termini complessi nel valore di aspettazione. Nel caso in cui si stia utilizzando un'osservabile non Hermitiana, se la parte immaginaria fosse inferiore a 10^{-5} , il codice restituirebbe la parte reale del valore di aspettazione, senza segnalare¹. Questa verifica è stata adottata per evitare possibili errori di floating point²

Questa flessibilità permette di considerare anche operatori non Hermitiani, a patto che la parte immaginaria sia sufficientemente piccola da essere trascurabile o poco rilevante per le misurazioni effettuate.

3.1.3 Gradiente dello stato quantistico

Costruendo il circuito quantistico di partenza, mediante le considerazioni fatte nella precedente sezione (3.1.1), è stato realizzato un oggetto che possa rappresentare l'approssimazione di una ipotetica soluzione reale $f(t)$ di una EDO. Tuttavia, per poter rappresentare l'intera equazione differenziale attraverso un sistema quantistico, è necessario costruire anche i circuiti che rappresentino i termini differenziali di

¹In caso contrario, restituirebbe comunque la parte reale, ma avvisando della presenza di una parte immaginaria scartata.

²Rappresenta l'errore in virgola mobile che potrebbe presentarsi sulla parte reale per descrivere i risultati, nel momento in cui vado ad escludere la parte immaginaria.

tale equazione. Come menzionato nella sezione 2.3, per costruire questi circuiti quantistici, è stata utilizzata la classe `Gradient()` di Qiskit, sfruttando il metodo della Parameter-shift Rule:

```

1 # creating first and second derivative generators from op with the parameter
  ↪ shift rule.
2 first_derivative = Gradient(grad_method = 'param_shift').convert(operator = op,
  ↪ params = t)
3 second_derivative = Gradient(grad_method = 'param_shift').convert(operator =
  ↪ first_derivative, params = t)

```

Codice 3.4: Costruzione dei sistemi quantistici che descrivono la derivata prima e la derivata seconda di un circuito quantistico, attraverso il metodo della Parameter-shift Rule.

dove il metodo `convert()`, viene utilizzato per convertire il circuito di partenza (contenuto nella variabile `op`), in una rappresentazione adatta per calcolare la derivata rispetto al tempo. Successivamente, è possibile ottenere la derivata seconda, semplicemente applicando nuovamente lo stesso metodo, ma con il sistema derivato in input al metodo `convert()`.

3.1.4 Ottimizzazione dei parametri quantistici

Una volta definiti tutti gli oggetti quantistici necessari per studiare un'equazione differenziale ordinaria (quindi $|f_{t,\theta}\rangle$, \hat{O} , $\partial_t |f_{t,\theta}\rangle$ e $\partial_t^2 |f_{t,\theta}\rangle$), diventa essenziale costruire una funzione di costo che valuti quanto bene i circuiti realizzati descrivano la funzione reale in base ai parametri assegnati.

```

1 # training vector weights to approximate the target function.
2
3 pt_nb = 15 # reference points in a time grid
4 tgrid = np.arange(pt_nb) / pt_nb * 2*np.pi # t from 0 to 2pi
5
6 # cost function definition (where x is an array containing the quantum
  ↪ parameters plus an additional parameter)
7 def cost_fun(x):
8     f = np.zeros(pt_nb)
9     df = np.zeros(pt_nb)
10    ddf = np.zeros(pt_nb)
11    cost_diff = 0
12
13    for i, t in enumerate(tgrid):
14        f[i] = evaluate_function(t, x[:-1], op) + x[-1]
15        df[i] = evaluate_function(t, x[:-1], first_derivative)
16        ddf[i] = evaluate_function(t, x[:-1], second_derivative)

```

```

17     cost_diff = cost_diff + np.abs(ddf[i] + a*df[i] + b*f[i])
18     f0 = evaluate_function(t0, x[:-1], op) + x[-1] # introducing initial
    ↪ condition  $f(0) = y0$ 
19     df0 = evaluate_function(t0, x[:-1], first_derivative) # introducing initial
    ↪ condition  $f'(0) = u0$ 
20
21     cost_val = cost_diff + 10*pt_nb * (f0 - u0)**2 + 10*pt_nb * (df0 - du0)**2
22     return cost_val

```

Codice 3.5: Definizione della funzione di costo associata ad una generica EDO di secondo ordine

Nel segmento di codice appena illustrato, viene rappresentata la definizione della funzione di costo descritta nella sezione 2.2.3.

Nelle prime due righe, è stato inizializzato un intervallo di tempo suddiviso da un numero equi-spaziato di punti, pari a `pt_nb` (Nel nostro caso, sono stati scelti solo 15 punti in modo da ridurre i tempi computazionali dell'algoritmo).

Per la funzione di costo sono stati definiti gli array, `f`, `df` e `ddf`, i quali conterranno i risultati della funzione `evaluate_function`, considerata in ogni punto dell'intervallo di tempo `tgrid` e con un set di parametri da ottimizzare assegnato in input alla funzione di costo. Attraverso la combinazione di questi risultati, viene studiata la condizione di annullamento della EDO, in modo da ricavarne la soluzione approssimata.

La variabile `cost_diff`, corrisponde al termine di costo differenziale $\mathcal{L}^{(\text{diff})}$ definito dall'equazione (2.13). Mentre le variabili `f0` e `df0`, compongono il contributo dovuto alle condizioni iniziali $\mathcal{L}^{(\text{init})}$ definito dall'equazione (2.14). Combinando infine questi risultati, si ottiene così la funzione di costo `cost_val` associata ai parametri θ (`x[:-1]`), che descrive l'accuratezza raggiunta dall'equazione differenziale approssimata.

A questo punto è possibile sfruttare uno degli algoritmi di ottimizzazione, messi a disposizione da Qiskit, con lo scopo di minimizzare la funzione di costo appena definita. Come descritto nella sezione 2.2, gli algoritmi di cui faremo uso nel corso delle sperimentazioni, saranno COBYLA e ADAM.

```

1  # defining the optimization approach.
2
3  optimizer = COBYLA(maxiter = 1000, tol = 1e-4, rhobeg = 0.1)
4
5  # ----- or -----
6
7  optimizer = ADAM(maxiter = 1000, tol = 1e-4, lr = 0.2)

```

Codice 3.6: Definizione degli algoritmi di ottimizzazione utilizzati.

Descriviamo brevemente gli argomenti forniti in ingresso ai due algoritmi:

- **maxiter**: Numero massimo di iterazioni nel processo di ottimizzazione³;
- **tol**: Tolleranza di convergenza dell'algoritmo, ovvero la differenza massima accettabile tra due punti successivi durante l'ottimizzazione;
- **rhobeg**: Parametro iniziale per il passo di ricerca dell'algoritmo COBYLA. Rappresenta la dimensione approssimativa iniziale della regione di ricerca;
- **lr**: Tasso di apprendimento iniziale (learning rate. si veda la sezione 2.2.2) dell'algoritmo ADAM.

Per dare il via alla ricerca dei migliori parametri, viene sfruttato il metodo `minimize()`, il quale riceverà in input, una funzione di costo da minimizzare e dei parametri iniziali con cui far partire il processo di minimizzazione su tale funzione.

```
1 # theta entries are initialized randomly. An additional weight helps the circuit  
  ↪ to learn initial conditions.  
2 initial_point = 0.1*np.random.rand(theta._size + 1)  
3  
4 # running the optimization  
5 result = optimizer.minimize(fun = cost_fun, x0 = initial_point)  
6  
7 # final results  
8 cost_function = result.fun  
9 best_params = result.x
```

Codice 3.7: Definizione dei parametri iniziali, riga 2. Avvio della ricerca del minimo della funzione di costo, riga 5. Restituzione risultati, riga 8 e 9.

In particolare nel nostro caso, i valori iniziali rappresentano delle quantità numeriche scelte randomicamente in un intervallo tra 0 e 0.1, che verranno assegnate ai parametri quantistici θ e ad un parametro aggiuntivo utilizzato al fine di migliorare la ricerca sulle condizioni iniziali del problema.

Una volta concluso l'intero processo di ottimizzazione, il metodo `minimize()` restituisce la funzione di costo finale e i migliori parametri associati, con cui possiamo rappresentare la soluzione approssimata dell'equazione differenziale ordinaria considerata.

³Per COBYLA, ogni iterazione corrisponde all'aggiornamento di un singolo parametro da ottimizzare. Per Adam, ogni iterazione corrisponde al completo aggiornamento di tutti i parametri da ottimizzare

3.1.5 Confronto con il metodo classico di Runge-Kutta

Utilizzando il metodo classico di Runge-Kutta (RK45) per la risoluzione di equazioni differenziali ordinarie, descritto nella sezione 2.1.1, si vuole infine mettere a confronto una soluzione classica con i risultati forniti dal sistema quantistico associato ai parametri ottimizzati.

```
1 def DH0(t,y,gamma,om_0):
2     return [y[1], - gamma * y[1] - om_0**2 * y[0]]
3
4 tdata = np.linspace(0, 2*np.pi, 100)
5 tspan = [tdata[0], tdata[-1]]
6 ic = [u0, du0]
7 quantum_sol = np.zeros(tdata.shape[0])
8
9 rk45_sol = solve_ivp(fun = DH0, tspan = tspan, y0 = ic, method = 'RK45', t_eval
    ↪ = tdata, args = (gamma, om_0))
10
11 for i, x in enumerate(tdata):
12     quantum_sol[i] = evaluate_function(x, best_params[:-1], op) +
    ↪ best_params[-1]
13
14 plt.plot(tdata, quantum_sol, label = 'Quantum solution')
15 plt.plot(tdata, rk45_sol.y[0], label = 'RK45 solution')
16 plt.title('Comparison between RK45 and Quantum solutions')
17 plt.legend()
18 plt.show()
```

Codice 3.8: Costruzione della soluzione, mediante il metodo RK45 e successiva comparazione con la soluzione quantistica.

Per ricavare la soluzione classica del metodo di Runge-Kutta, è stata utilizzata la classe `solve_ivp`, del pacchetto *SciPy*. In particolare, l'equazione che è stata fornita come argomento a questa classe, si riferisce all'equazione differenziale di un oscillatore armonico smorzato, in quanto sarà oggetto di studio per le analisi affrontate nella sezione seguente.

3.2 Sperimentazione e risultati

In questa sezione verrà studiato il comportamento dell'algoritmo precedentemente discusso, al fine di risolvere l'equazione differenziale di un oscillatore armonico smorzato.

Affinché si possano comprendere e distinguere al meglio le ragioni che causano delle

particolari modifiche alla risposta dell'algoritmo realizzato, è necessario che il set di elementi manipolabili di tale problema, come i parametri dell'oscillatore armonico, i circuiti quantistici, le osservabili, ecc., non sia troppo elevato. Per tale ragione, le sperimentazioni discusse in questa sezione verranno affrontate facendo variare un numero più ristretto di tali elementi, mentre i restanti verranno definiti e mantenuti invariati.

Riportiamo di seguito le relative restrizioni adottate:

Scelta dei parametri e delle condizioni iniziali dell'oscillatore armonico smorzato:

Frequenza angolare	$\omega_0 = 1$
Coefficiente di smorzamento	$\gamma \in [0, 4)$
Condizioni iniziali	$x(0) = x_0 = 0.8$
	$dx(0) = u_0 = 0$

Tabella 3.1: coefficienti e condizioni iniziali della EDO di un oscillatore armonico smorzato. γ è stato l'unico parametro che abbiamo fatto variare.

Scelta dell'osservabile Hermitiana:

L'osservabile Hermitiana è stata definita mediante un operatore di Pauli Z per ciascun qubit del sistema, e un fattore moltiplicativo determinato da uno dei parametri θ da ottimizzare.

```

1 observable = theta[-1]*Z
2 for i in np.arange(n_q-1):
3     observable = observable ^ Z

```

Codice 3.9: Definizione di un'osservabile Hermitiana con i soli operatori di Pauli. n_q rappresenta il numero di qubit considerati nel circuito.

Scelta delle metriche nella funzione di costo:

Dal seguente segmento di codice, si può osservare che, la metrica associata alla funzione di costo differenziale (riga 3) corrisponde all'errore medio assoluto (2.16). Mentre la metrica associata alle condizioni iniziali (riga 6), è descritta dall'errore quadratico medio (2.15).

È importante sottolineare che la scelta delle metriche è stata effettuata in modo arbitrario, basandoci esclusivamente su quelle definite nella sezione 2.2.3. Probabilmente, attraverso ulteriori studi, sarebbe stato possibile individuare metriche più appropriate.

```

1 # [...]
2     # MSE (Mean Squared Error) metric
3     cost_diff = cost_diff + np.abs(ddf[i] + a*df[i] + b*f[i] + c)
4 # [...]
5     # MAE (Mean Absolutated Error) metric
6     cost_val = cost_diff + 10*pt_nb*(f0 - u0)**2 + 10*pt_nb*(df0 - du0)**2
7 # [...]

```

Codice 3.10: Rappresentazione delle metriche utilizzate nella definizione della funzione di costo.

Scelta dell’algoritmo di ottimizzazione:

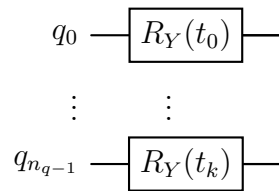
Come vedremo nella sezione 3.3, rispetto all’algoritmo COBYLA, Adam risulta essere un’alternativa più efficace per la ricerca del minimo della funzione di costo. Tuttavia, in termini di complessità computazionale e utilizzo di risorse, Adam si è dimostrato meno efficiente, pertanto, la scelta dell’algoritmo COBYLA è risultata più appropriata agli studi inerenti a questa sezione.

Scelta del numero di qubit:

Essendo necessario tenere presente delle limitazioni associate alla complessità computazionale, la costruzione e l’analisi dei circuiti quantistici variazionali, sono stati affrontati per sistemi con un numero di qubit non più grande di 6.

3.2.1 Circuito quantistico 1: $R_Y(t)$

Il circuito quantistico più semplice analizzato, consiste in un insieme di gate rotazionali $R_Y(t \cdot \theta_i + \theta_j)$, ognuno dei quali applicato a ciascun qubit del sistema.



Circuito 3.1: Circuito quantistico 1 con n_q numero di qubit e $t_k = t \cdot \theta_i + \theta_j$.

Considerando come parametri iniziali, dei valori casuali compresi tra 0 ed 0.1 (come descritto nella riga 2 del codice 3.7), si è partiti a studiare la configurazione di un oscillatore armonico semplice, ossia quando il coefficiente di smorzamento γ è nullo. Dai seguenti grafici si può osservare come la minimizzazione della funzione di costo abbia restituito dei buoni parametri per la descrizione della funzione reale.

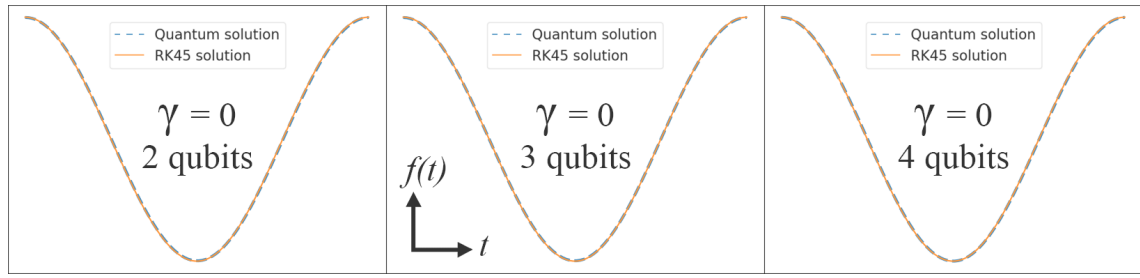


Figura 3.2: Comparazione fra le soluzioni approssimate dal metodo RK45 e le soluzioni quantistiche, per γ per le configurazioni a 2, 3 e 4 qubit.

In particolare, in questi grafici (figura 3.2) sono state messe a confronto le soluzioni dell'oscillatore armonico, ottenute mediante il metodo classico di Runge-Kutta (RK45), con le funzioni quantistiche ricavate dai circuiti quantistici ottimizzati a diversi numeri di qubit (queste ultime, rappresentate in linea tratteggiata, sono nascoste dalle curve del metodo classico).

Per farsi un'idea numerica sulla bontà dei dati ottenuti in relazione alle soluzioni ricavate dal modello di Runge-Kutta, può essere considerata la somma degli scarti quadratici (RSS - Residual Sum of Squares) tra le due funzioni, ottenendo dunque:

Somma degli scarti quadratici (RSS)			
	2 qubit	3 qubit	4 qubit
$\gamma = 0$	1.657×10^{-3}	0.264×10^{-3}	1.020×10^{-3}

Tabella 3.2: Valori degli RSS per γ nullo.

Di seguito sono stati riportati invece i risultati delle relative funzioni di costo ottenute dall'ottimizzazione:

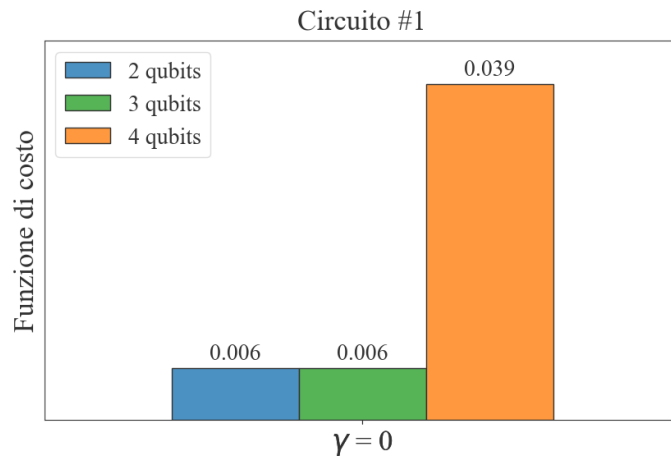


Figura 3.3: Istogramma circuito 1 delle funzioni di costo a 2, 3 e 4 qubit, per $\gamma = 0$.

Nonostante la funzione di costo restituisca una buona approssimazione della funzione reale, per tutte e tre le configurazioni, non è stato tuttavia possibile determinare se l'aumento del numero di qubit, abbia potuto migliorare le prestazioni di ricerca. Anzi, è possibile che un aumento di questo tipo, il quale comporta un incremento dei parametri quantistici da ottimizzare, determini un leggero abbassamento delle prestazioni dell'algoritmo COBYLA in prossimità del minimo.

	Funzione di costo		
	2 qubit	3 qubit	4 qubit
$\gamma = 0$	0.584×10^{-2}	0.597×10^{-2}	3.905×10^{-2}

Tabella 3.3: Valori delle funzioni di costo per γ nullo.

Rispetto alla somma degli scarti quadratici (RSS), la funzione di costo costituisce già un buon indicatore su cosa aspettarci nel confronto tra i risultati sperimentali e la soluzione esatta. Tuttavia è bene precisare, che tale quantità è da interpretare come uno scarto quadratico medio sulla ricerca dei parametri ottimali e non ha alcuna relazione diretta con il modello di approssimazione classico comparato.

Successivamente è stata eseguita un'analisi della funzione di costo in un intervallo di smorzamento $0 < \gamma < 4$, con un passo di 0.1. In particolare, per la scelta dei parametri quantistici iniziali di ciascuno smorzamento, sono stati considerati i migliori parametri del il circuito quantistico precedentemente studiato (per $\gamma = 0$) e, aumentando gradualmente il livello di smorzamento, sono stati poi ricavati parametri ottimali dei regimi successivi, utilizzando come punto di partenza i migliori parametri ottenuti al γ del passo precedente.

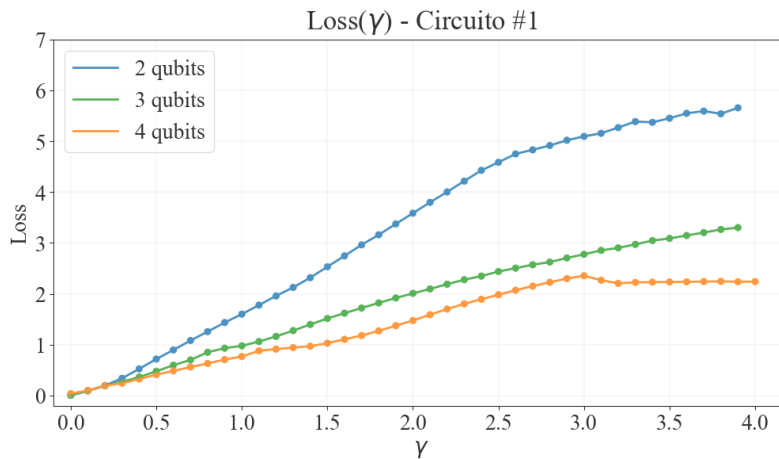


Figura 3.4: Studio della funzione di costo al variare di γ per 2, 3 e 4 qubit.

All'aumentare di γ , si verifica un evidente peggioramento della funzione di costo. Ciò è dovuto al fatto che i parametri iniziali da ottimizzare, sono stati scelti prendendo

come riferimento quelli ottimali per il caso dell'oscillatore armonico semplice. Per tale ragione, il miglior adattamento trovato per una funzione sinusoidale, potrebbe non rappresentare un buon punto di partenza per regimi di smorzamento in cui il termine oscillante possa risultare trascurabile. Nonostante ciò, in questo caso, è comunque possibile notare un miglioramento legato al numero di qubit. In particolare, si osserva che all'aumentare del numero di questi, la funzione di costo tende ad avere un incremento minore, al crescere di gamma, rispetto ai circuiti con un numero di qubit inferiore.

Presi come campioni due differenti regimi di smorzamento ($\gamma = 1.5$ e $\gamma = 3.5$), possiamo valutare numericamente come variano le relative funzioni di costo:

	Funzione di costo		
	2 qubit	3 qubit	4 qubit
$\gamma = 1.5$	2.532	1.515	1.029
$\gamma = 3.5$	5.451	3.089	2.232

Tabella 3.4: Valori delle funzioni di costo per $\gamma = 1.5$ e 3.5 .

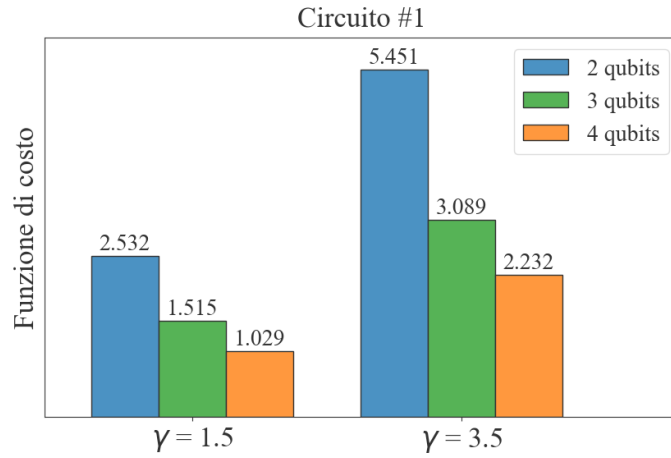


Figura 3.5: Istogramma circuito 1 delle funzioni di costo a 2, 3 e 4 qubit, per $\gamma = 1.5$ e 3.5 .

Possiamo di conseguenza osservare dai grafici in figura 3.6 come, anche la soluzione quantistica messa a confronto con quella ottenuta dal modello classico, subisca un peggioramento al crescere di γ :

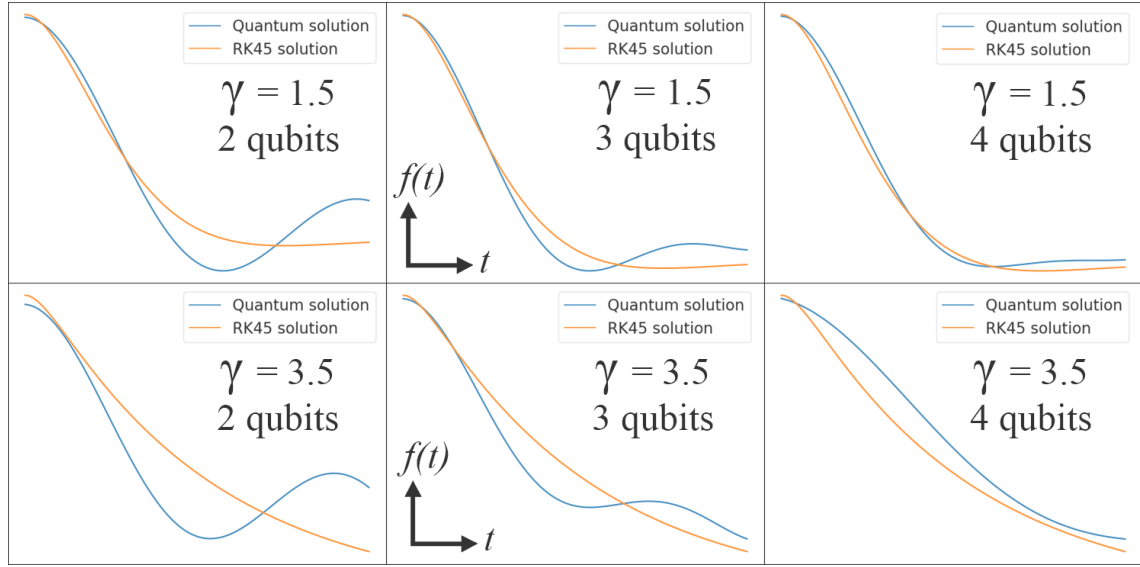


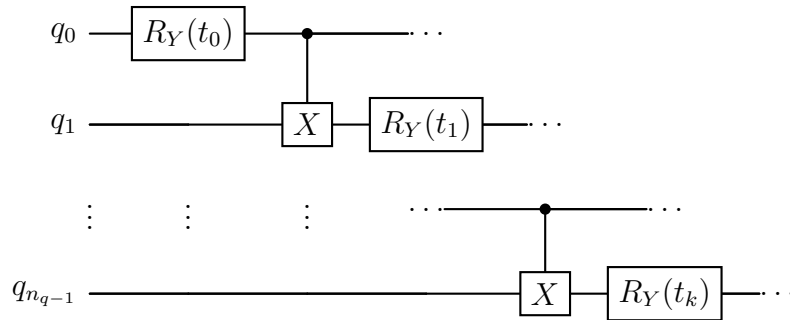
Figura 3.6: Grafici di comparazione fra le soluzioni quantistiche e quelle classiche, per $\gamma = 1.5$ e 3.5 .

	Somma degli scarti quadratici (RSS)		
	2 qubit	3 qubit	4 qubit
$\gamma = 1.5$	7.232×10^{-1}	2.319×10^{-1}	0.614×10^{-1}
$\gamma = 3.5$	14.257×10^{-1}	3.498×10^{-1}	3.364×10^{-1}

Tabella 3.5: Valori degli RSS per $\gamma = 1.5$ e 3.5 .

3.2.2 Circuito quantistico 2: $R_Y(t) + \text{CNOT}$

Nel secondo circuito sperimentato sono state aggiunte delle porte **CNOT** tra i gate già presenti nel circuito precedente, con lo scopo di analizzare il tipo di risposta del sistema in presenza di entanglement.



Circuito 3.2: Circuito quantistico 2 con n_q numero di qubit e $t_k = t \cdot \theta_i + \theta_j$.

I risultati ottenuti mostrano anche in questo caso una buona approssimazione della funzione reale per $\gamma = 0$, senza alcun apparente vantaggio aumentando il numero di qubit del sistema.

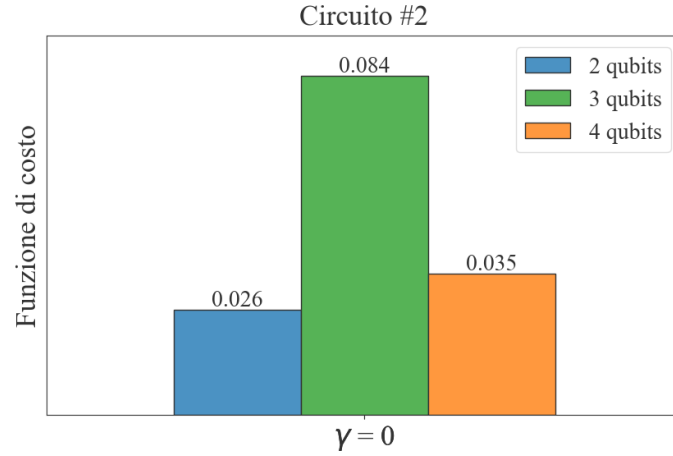


Figura 3.7: Istogramma circuito 2 delle funzioni di costo a 2, 3 e 4 qubit, per γ nullo.

A differenza dei circuiti a 3 e 4 qubit, è interessante osservare nell'istogramma seguente come l'errore sul circuito a 2 qubit sia rimasto pressoché invariato, addirittura diminuito, passando da $\gamma = 1.5$ a $\gamma = 3.5$.

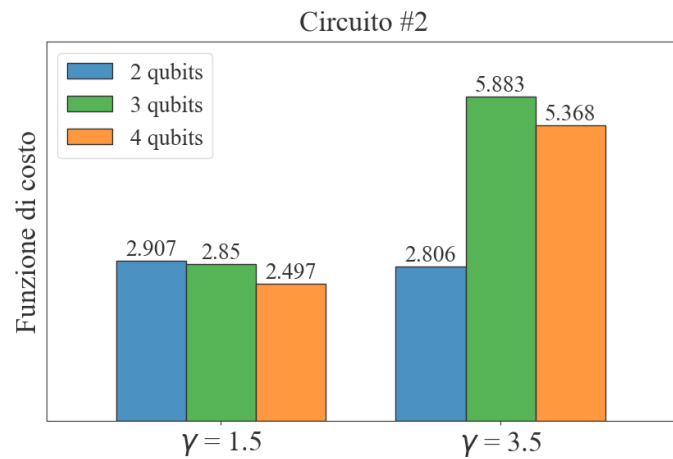


Figura 3.8: Istogramma circuito 2 delle funzioni di costo a 2, 3 e 4 qubit, per $\gamma = 1.5$ e 3.5 .

	Funzione di costo		
	2 qubit	3 qubit	4 qubit
$\gamma = 0$	2.578×10^{-2}	8.392×10^{-2}	3.453×10^{-2}
$\gamma = 1.5$	2.907	2.850	2.497
$\gamma = 3.5$	2.806	5.883	5.368

Tabella 3.6: Valori delle funzioni di costo a 2, 3 e 4 qubit, per $\gamma = 0, 1.5$ e 3.5 .

La ragione di tale comportamento, può essere spiegata mediante il grafico in figura 3.9 della funzione di costo al variare di γ .

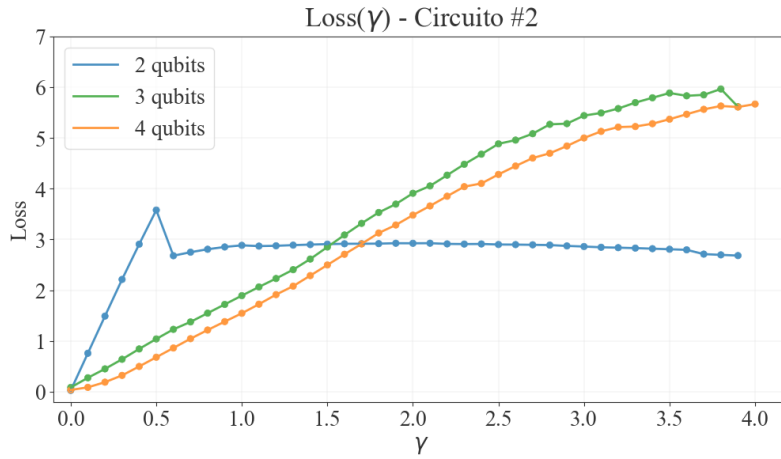


Figura 3.9: Studio della funzione di costo al variare di γ per 2, 3 e 4 qubit.

Sostanzialmente i parametri iniziali, assegnati nella configurazione di $\gamma = 0$, hanno causato un fenomeno di divergenza molto rapido per la funzione di costo associata al sistema a 2 qubit. Come conseguenza a ciò, si può notare un salto drastico della funzione di costo nella regione tra $\gamma = 0.5$ e $\gamma = 0.6$.

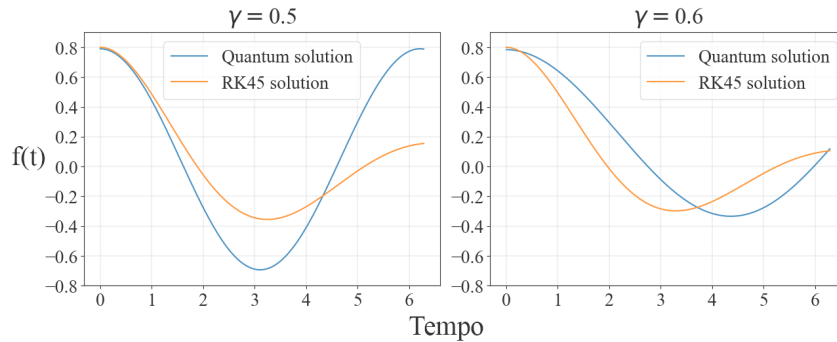
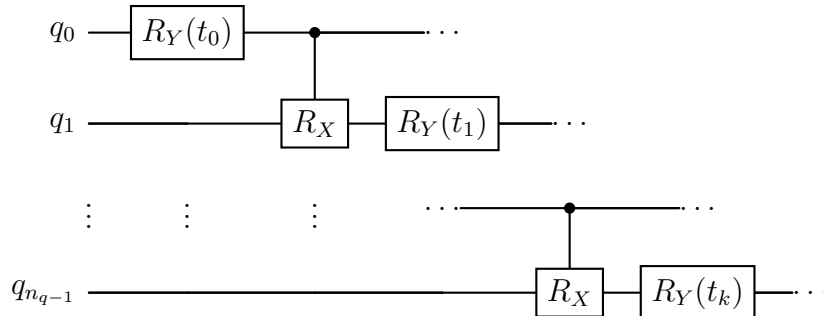


Figura 3.10: Rappresentazione della soluzione nella regione di smorzamento in cui è avvenuto il salto della funzione di costo.

L'utilizzo di un certo tipo di parametri iniziali può influenzare notevolmente il processo di minimizzazione nelle ricerche successive. Per questo motivo, anche se nelle prime iterazioni COBYLA è rimasto "intrappolato" in un minimo locale corrispondente al set dei buoni parametri per $\gamma = 0$, al crescere dello smorzamento, l'algoritmo COBYLA è stato in grado di uscire dalla regione di tale minimo fornendo una soluzione migliore e parametri più adatti per le interazioni successive. Questo potrebbe dunque spiegare il salto osservato nel grafico, portandoci inoltre ad osservare il successivo andamento accennato in figura 3.9: Per i valori successivi di γ , la funzione di costo è rimasta circa costante (con una lieve decrescita), risultando in questo modo, migliore dei circuiti a 3 e 4 qubit che, diversamente, hanno mantenuto per tutta l'elaborazione successiva, un andamento divergente influenzato dei parametri iniziali. In tutte e tre le configurazioni del circuito, l'aumento del valore di γ , comporta una ricerca del valore ottimale in regioni via via sempre più lontane dal minimo assoluto. Pertanto, se i parametri ottimizzati da COBYLA, come appena accennato, rimanessero intrappolati in una regione di minimo locale, la modifica questi, comporterebbe variazioni poco significative rispetto ai valori di riferimento iniziali, senza dunque portare miglioramenti nell'ottimizzazione. Questo è ciò che è successo per i circuiti a 3 e 4 qubit. Al contrario, se l'algoritmo COBYLA per qualche ragione riuscisse ad allontanarsi dalla regione di minimo locale, i parametri di ricerca potrebbero cambiare in modo significativo e diventare potenzialmente dei migliori candidati rispetto a quelli iniziali, come è successo per il circuito a 2 qubit. Ciò potrebbe essere attribuito al fatto che tale circuito, offre una maggiore flessibilità e una dimensione dello spazio dei parametri più limitata rispetto ai circuiti con un maggior numero di qubit. Di conseguenza, l'algoritmo COBYLA potrebbe esplorare un numero maggiore di regioni diverse nello spazio dei parametri e trovare soluzioni più adatte al problema considerato.

3.2.3 Circuito quantistico 3: $RY(t) + CRX$

Nel terzo circuito studiato, sono state sostituite le porte **CNOT**, con i gate **Controlled-RX** dipendenti dai soli parametri θ ($\rightarrow CR_X(\theta_i)$).



Circuito 3.3: Circuito quantistico 3 con n_q numero di qubit e $t_k = t \cdot \theta_i + \theta_j$. In questo caso le porte CR_X dipendono solo dai parametri θ .

I risultati ottenuti per questa configurazione, non mostrano sostanziali differenze

rispetto al primo circuito. Di seguito sono riportati i risultati grafici e numerici.

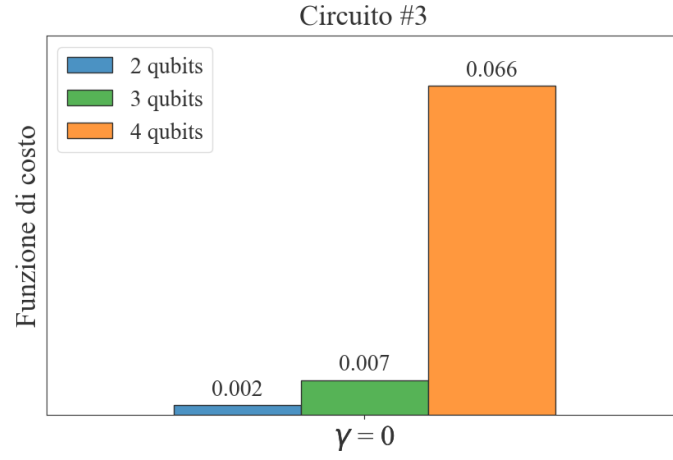


Figura 3.11: Istogramma delle funzioni di costo a 2, 3 e 4 qubit, per γ nullo.

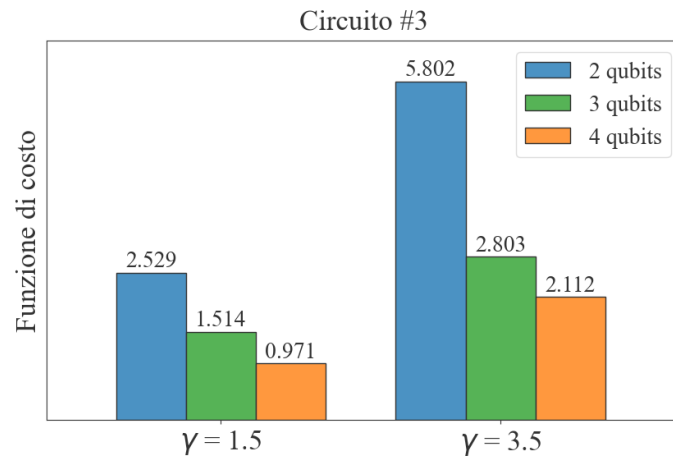


Figura 3.12: Istogramma delle funzioni di costo a 2, 3 e 4 qubit, per $\gamma = 1.5$ e 3.5 .

	Funzione di costo		
	2 qubit	3 qubit	4 qubit
$\gamma = 0$	0.209×10^{-2}	0.656×10^{-2}	6.634×10^{-2}
$\gamma = 1.5$	2.529	1.514	0.971
$\gamma = 3.5$	5.802	2.803	2.112

Tabella 3.7: Valori delle funzioni di costo a 2, 3 e 4 qubit, per $\gamma = 0, 1.5$ e 3.5 .

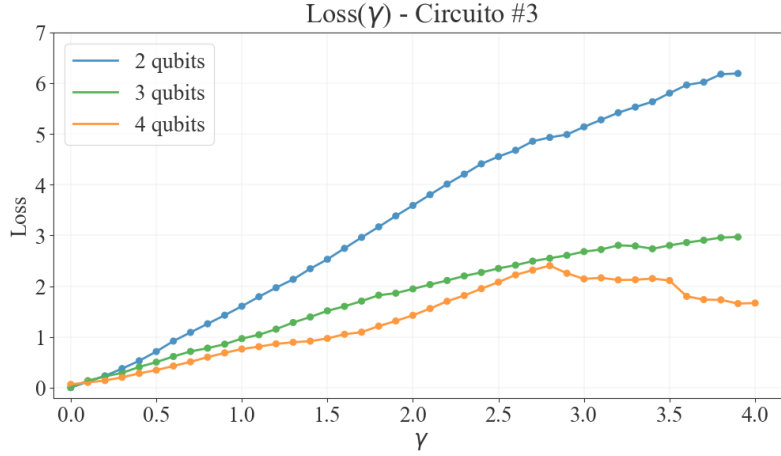
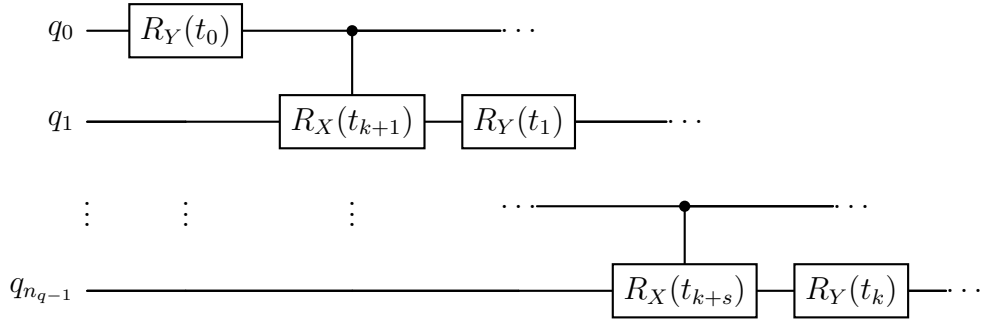


Figura 3.13: Studio della funzione di costo al variare di γ , per 2, 3 e 4 qubit.

3.2.4 Circuito quantistico 4: $R_Y(t) + CRX(t)$

Nel quarto circuito è stata considerata la stessa configurazione di gate del circuito appena discusso, con l'aggiunta però di una dipendenza dal parametro temporale nelle porte $CR_X(t \cdot \theta)$:



Circuito 3.4: Circuito quantistico 4 con n_q numero di qubit e $t_k = t \cdot \theta_i + \theta_j$ per i gate R_Y e $t_k = t \cdot \theta_i$ per i gate CR_X .

Come spiegato nella sezione 2.3.1, relativa al metodo della Parameter-shift Rule, l'aumento del numero di parametri temporali presenti all'interno di un circuito quantistico, comporta un aumento significativo di gate utilizzati per descrivere i gradienti. Pertanto, per ragioni di complessità computazionale in termini di tempo di elaborazione, l'analisi sul comportamento della funzione di costo al variare di γ , è stata eseguita solamente per il caso a 3 qubit.

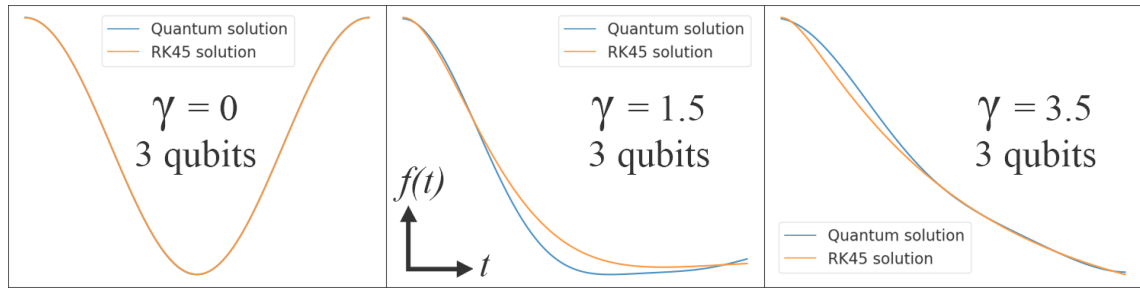


Figura 3.14: Comparazione fra soluzioni quantistiche e soluzioni classiche del modello di Runge-Kutta, per $\gamma = 0, 1.5$ e 3.5 .

3 qubit

	RSS	Funzione di costo
$\gamma = 0$	0.023×10^{-2}	0.025×10^{-1}
$\gamma = 1.5$	14.733×10^{-2}	10.786×10^{-1}
$\gamma = 3.5$	3.585×10^{-2}	14.190×10^{-1}

Tabella 3.8: Valori numerici degli RSS e delle funzioni di costo a 3 qubit, per $\gamma = 0, 1.5$ e 3.5 .

Dai valori ottenuti della la funzione di costo, si può osservare un miglioramento nella ricerca dei parametri ottimali, soprattutto per quanto riguarda i circuiti con coefficiente di smorzamento più alto.

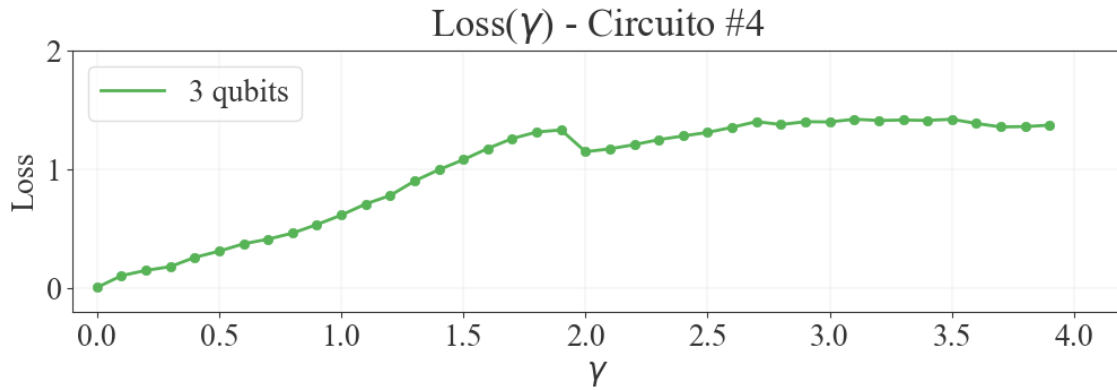


Figura 3.15: Studio della funzione di costo al variare di γ , per 3 qubit.

Un'ulteriore verifica affrontata con il presente circuito, è consistita nel fissare il coefficiente di smorzamento ad un generico valore di γ (esempio 1.5), e far variare il numero di qubit in un range leggermente più ampio, ossia da 2 a 6 qubit. I risultati ottenuti sono di significativo interesse, in quanto si è potuto osservare un evidente miglioramento dovuto all'aumento del numero di qubit.

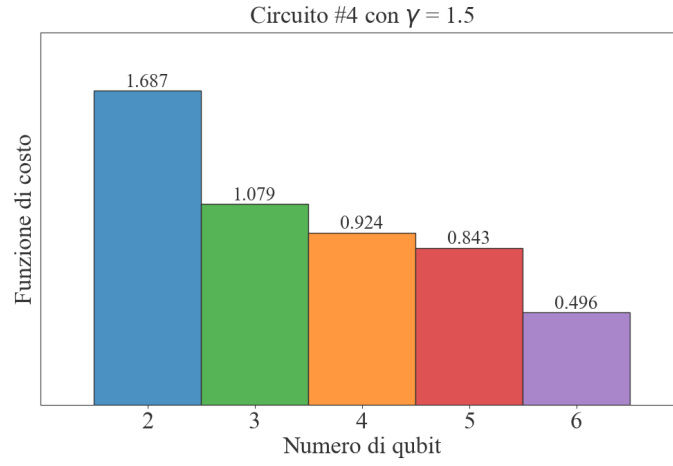


Figura 3.16: Istogramma della funzione di costo per $\gamma = 1.5$ al variare del numero di qubit nel circuito 4.

$\gamma = 1.5$

	RSS	Funzione di costo
2 qubit	5.526×10^{-1}	1.687
3 qubit	1.473×10^{-1}	1.079
4 qubit	1.087×10^{-1}	0.924
5 qubit	0.646×10^{-1}	0.843
6 qubit	0.049×10^{-1}	0.496

Tabella 3.9: Valori numerici degli RSS e delle funzioni di costo per $\gamma = 1.5$ al variare del numero di qubit nel circuito 4.

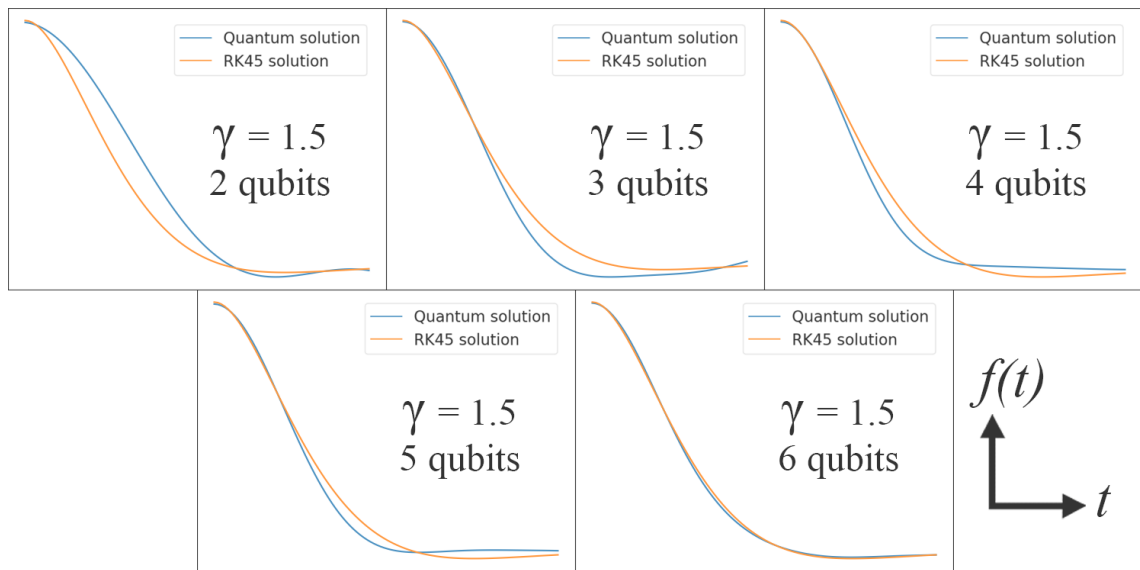


Figura 3.17: Comparazione fra le soluzioni quantistiche e le soluzioni del modello classico di Runge-Kutta al variare del numero di qubit.

Come si può osservare dai grafici in figura 3.17 e dalla colonna delle RSS, l'approssimazione quantistica della soluzione rispetto al modello classico tende ad avere un miglior accordo nel caso a 6 qubit.

3.2.5 Confronto dei risultati fra i circuiti

In questa sezione, sono stati messi in relazione i risultati ottenuti dalle quattro tipologie di circuito studiate finora, al fine di comprendere ulteriori aspetti utili per la costruzione di un buon algoritmo quantistico per le approssimazioni delle EDO.

Nel caso dei circuiti a due qubit (figura 3.18), come discusso nella sezione 3.2.2, si è potuta osservare una rapida crescita iniziale della funzione di costo per quanto riguarda il circuito con le CNOT gate, susseguita da una ridefinizione dei parametri quantistici da parte di COBYLA, portando un miglioramento relativo nelle condizioni di smorzamento successive. Tuttavia, nonostante tale circuito presenti dei migliori risultati (da $\gamma = 1.7$ in poi) rispetto ai circuiti 1 e 2, non si può affermare che si tratti della miglior configurazione circuitale, in quanto si osserva dagli altri grafici una divergenza significativa della funzione di costo al crescere di γ , rispetto agli altri circuiti studiati. Pertanto si è portati inoltre a supporre che l'aggiunta di gate non parametrici all'interno di un circuito quantistico variazionale, non porti migliorie ai risultati, poiché non modulabili a seconda dell'EDO studiata.

Un ulteriore aspetto, osservato nei tre grafici seguenti, riguarda l'andamento molto simile delle funzioni di costo dei circuiti 1 e 3. La ragione di tale comportamento potrebbe essere attribuita a una struttura circuitale simile in termini di complessità. I gate CRX, avendo una dipendenza solo dai parametri quantistici θ e non dal parametro temporale, non hanno contribuito ad aumentare il numero di gate necessari per descrivere i gradienti. È quindi possibile che l'algoritmo di ottimizzazione abbia cercato di assegnare i migliori parametri ai termini di entanglement, al fine di attenuare il comportamento divergente riscontrato con le porte CNOT (le quali non potevano essere modulate, poiché non parametriche) e favorire così le trasformazioni dovute ai gate rotazionali $R_Y(t)$. Ad ogni modo, l'aggiunta di tali porte, non ha costituito un peggioramento sulla ricerca del minimo, ma anzi, si può osservare che, per valori di γ alti, si abbia un leggero miglioramento, dovuto probabilmente alla presenza di più gradi di libertà nel sistema.

Infine, dai vari grafici sottostanti, ma più in particolare dalla figura 3.19, si nota un miglioramento nella ricerca del minimo della funzione di costo con il circuito 4, ovvero con l'aggiunta del parametro temporale nei gate CRX. Questo comportamento può essere intuitivamente spiegato dalla presenza nel sistema di un maggior numero di gate dipendenti dal tempo, i quali permettono ai parametri θ , di adattarsi meglio a ciascun tempo dell'intervallo considerato.

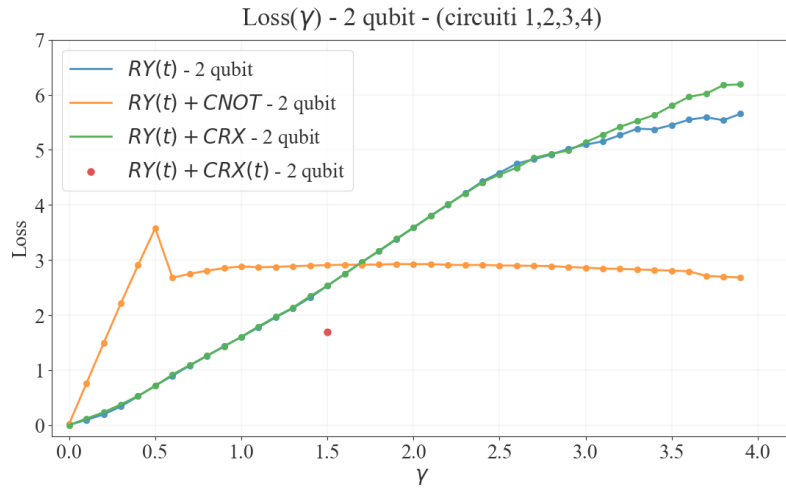


Figura 3.18: Confronto delle funzioni di costo per i quattro circuiti studiati a 2 qubit.

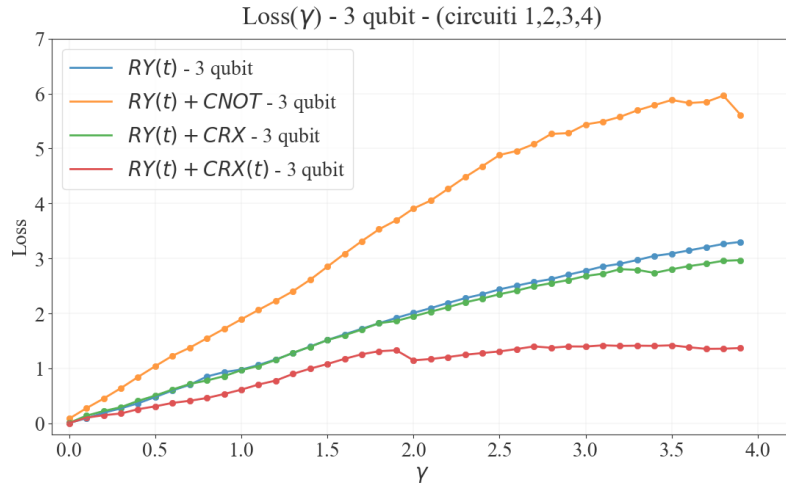


Figura 3.19: Confronto delle funzioni di costo per i quattro circuiti studiati a 3 qubit.

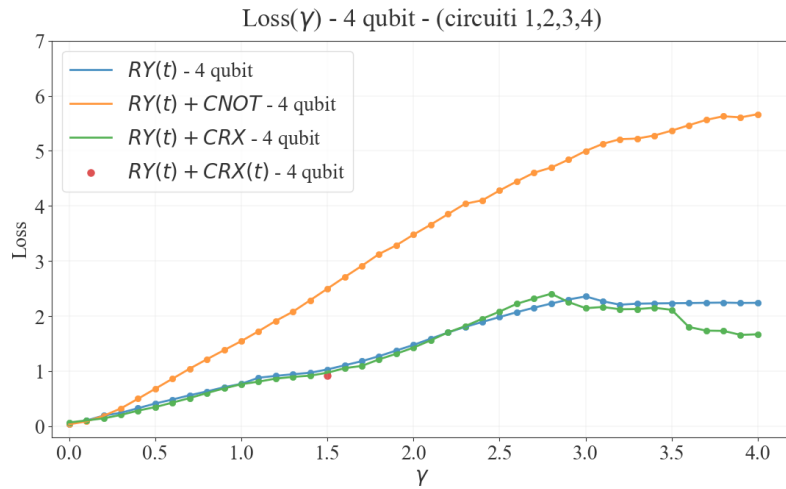


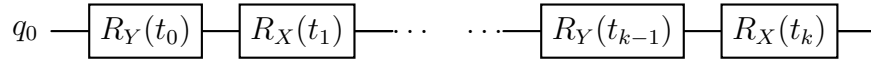
Figura 3.20: Confronto delle funzioni di costo per i quattro circuiti studiati a 4 qubit.

Nella seguente sezione, verrà studiato un circuito quantistico ad un solo qubit, con lo scopo di mostrare come l'aumento del numero di gate dipendenti dal tempo nel circuito, possa effettivamente migliorare l'ottimizzazione e, di conseguenza, aumentare l'accuratezza della soluzione approssimata di un'Equazione Differenziale Ordinaria.

3.2.6 Influenza dei gate dipendenti dal tempo

Nella presente sezione illustreremo gli effetti riscontrati con l'aumento del numero di gate dipendenti dal tempo, presenti in un circuito quantistico a singolo qubit, utilizzato per risolvere l'EDO di un oscillatore armonico smorzato.

Il circuito quantistico a singolo qubit preso in considerazione, consiste in una sequenza di porte rotazionali R_X ed R_Y dipendenti dal tempo. Tali gate verranno ripetuti per un certo numero di volte, con lo scopo di studiare come il parametro temporale influenzi il processo di ottimizzazione per la risoluzione di un'EDO.



Circuito 3.5: Circuito quantistico a 1 qubit, con $t_k = t \cdot \theta_i + \theta_j$.

Il circuito è stato rappresentato con Qiskit nel seguente modo:

```

1 np = 4 # number of pair iterations
2 t = Parameter("t")
3 theta = ParameterVector("theta", np*4)
4
5 ansatz = QuantumCircuit(1)
6 for k in np.arange(0, np*4, 4):
7     ansatz.ry(t*theta[k] + theta[k+1], 0)
8     ansatz.rx(t*theta[k+2] + theta[k+3], 0)

```

Codice 3.11: Rappresentazione in Qiskit di un circuito quantistico ad 1 qubit, in cui viene iterata l'applicazione della coppia di gate R_X - R_Y per un certo numero di volte.

dove np rappresenta il numero di coppie di gate R_Y - R_X applicati al qubit.

Per le analisi, sono stati scelti rispettivamente i circuiti con $np = 2, 3$ e 4 , e sono stati utilizzati per la risoluzione dell'EDO di un oscillatore armonico smorzato con $\gamma = 1.5$.

Riportando i risultati nel seguente istogramma, è possibile notare come l'ottimizzazione dei parametri θ , sia stata più efficace utilizzando il circuito con il maggior numero di gate dipendenti dal tempo (colonna C), rispetto a quelli considerati.

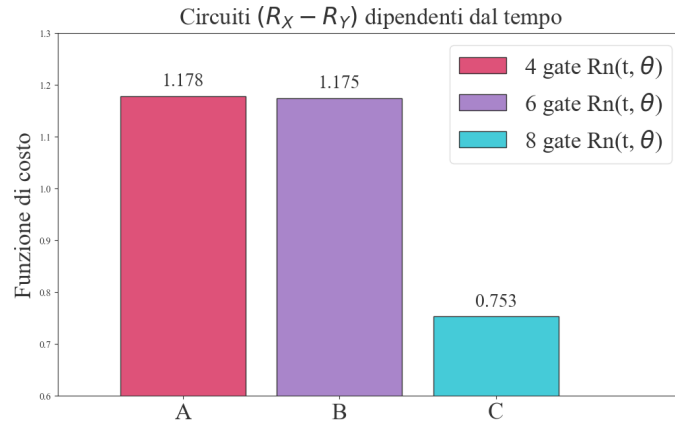


Figura 3.21: Istogramma delle funzioni di costo al variare del numero di gate dipendenti dal tempo all'interno del circuito quantistico.

$\gamma = 1.5$

	Funzione di costo	RSS
A	1.178	0.222
B	1.175	0.230
C	0.753	0.052

Tabella 3.10: Valori numerici delle funzioni di costo e degli RSS relativi ai 3 circuiti quantistici studiati.

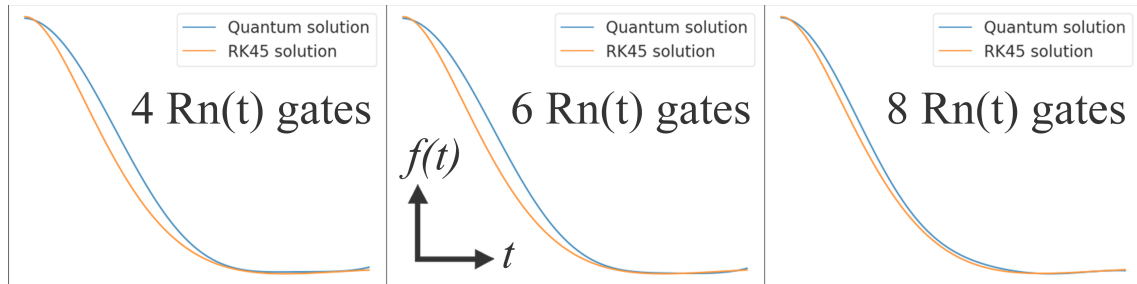


Figura 3.22: Comparazione fra le soluzioni quantistiche e le soluzioni del modello classico, mostrando come il migliore accordo si sia trovato con il maggior numero di gate dipendenti dal tempo.

Per fare un'ulteriore verifica, è stato rieseguito il circuito con i migliori risultati (circuito con 8 gate rotazionali) rimuovendo la dipendenza temporale dagli ultimi due gate:

```
1 ansatz.ry(t*theta[0] + theta[1], 0)
2 ansatz.rx(t*theta[2] + theta[3], 0)
3 ansatz.ry(t*theta[4] + theta[5], 0)
4 ansatz.rx(t*theta[6] + theta[7], 0)
5 ansatz.ry(t*theta[8] + theta[9], 0)
6 ansatz.rx(t*theta[10] + theta[11], 0)
7 ansatz.ry(theta[12], 0)
8 ansatz.rx(theta[13], 0)
```

Codice 3.12: Rappresentazione in Qiskit del circuito C, ma senza le dipendenze temporali per gli ultimi due gate.

Risultato della funzione di costo: 1.207

Come ci si aspettava, il riscontro ottenuto, non ha portato alcun miglioramento nella ricerca dei parametri ottimali, rimanendo invece in linea ai risultati dei circuiti più piccoli.

3.3 Confronto tra gli agloritmi COBYLA e Adam

L'utilizzo dell'algoritmo di ottimizzazione COBYLA, per gli studi affrontati in questa tesi è stato principalmente dettato dalla necessità di produrre risultati in tempi ragionevoli. Tra gli algoritmi di ottimizzazione forniti da Qiskit, COBYLA si è dimostrato essere il più efficiente in termini di elaborazione computazionale, pur mantenendo una discreta efficacia nella ricerca dei migliori parametri quantistici, per descrivere la soluzione dell'equazione differenziale di un oscillatore armonico smorzato.

Per ottenere risultati più accurati e una soluzione più precisa dell'EDO, l'utilizzo dell'algoritmo Adam rappresenta invece una migliore alternativa in termini di efficacia, in quanto rappresenta uno strumento di grande versatilità nella ricerca del minimo, grazie al suo learning rate adattivo durante il processo di ottimizzazione, che permette una convergenza più veloce verso la soluzione ottimale. Tuttavia, nonostante i suoi vantaggi, questo algoritmo si è rivelato su Qiskit, poco efficiente computazionalmente.

In questa sezione si vogliono mostrare dei semplici risultati di prova, effettuati per confrontare i due algoritmi in questione nella risoluzione di un'equazione differenziale ordinaria.

Oltre a dipendere dall'algoritmo di ottimizzazione utilizzato, la complessità computazionale cresce anche a causa delle operazioni svolte sui circuiti quantistici del sistema, in particolare su quelli che descrivono i gradienti. Pertanto, si è voluto per semplicità, ridurre il problema ad un'equazione differenziale del primo ordine del tipo:

$$\frac{dx}{dt} + x = 0 \quad (3.3)$$

Per la scelta del circuito quantistico si è optato nel considerare una sequenza di gate dipendenti dal tempo, applicati su un solo qubit:

```

1 ansatz.ry(t*theta[0] + theta[1], 0)
2 ansatz.rx(t*theta[2] + theta[3], 0)
3 ansatz.ry(t*theta[4] + theta[5], 0)
4 ansatz.rx(t*theta[6] + theta[7], 0)

```

Codice 3.13: Rappresentazione in Qiskit del circuito quantistico utilizzato per il confronto dei due algoritmi di ottimizzazione.

mentre per quanto riguarda l'osservabile, è stata mantenuta quella utilizzata nelle sezioni precedenti (si veda il codice 3.9).

Infine, i parametri iniziali utilizzati per entrambi gli algoritmi sono stati i seguenti:

```

1 initial_params = [0.21716281, 0.29543572, 0.17863761, 0.18992607, 0.17780186,
  ↪ 0.25914271, 0.16410744, 0.02885816, 0.23024582, 0.09417163]

```

Codice 3.14: Valori numerici dei parametri iniziali utilizzati per il confronto.

Da le analisi effettuate si sono quindi ottenuti i risultati riportati nel seguente istogramma:

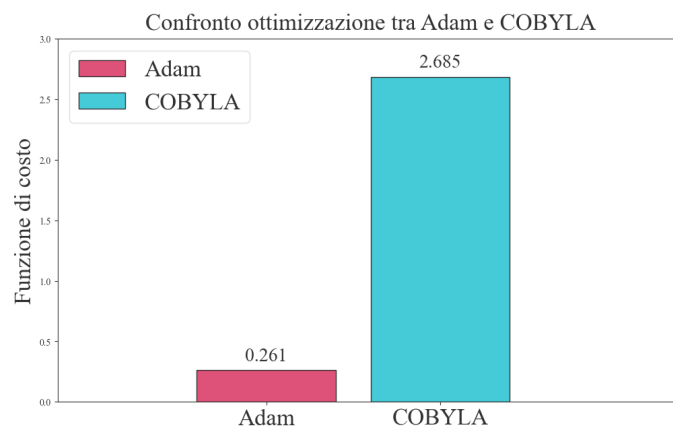


Figura 3.23: Istogramma delle funzioni di costo relative agli algoritmi Adam e COBYLA associate a circuito quantistico studiato per risolvere l'EDO 3.3.

Osservando inoltre il confronto delle soluzioni fra modello quantistico e modello classico, si ha un'ulteriore conferma che l'algoritmo di ottimizzazione Adam, si comporta molto meglio nella ricerca della soluzione ottimale di un'equazione differenziale, rispetto all'algoritmo COBYLA.

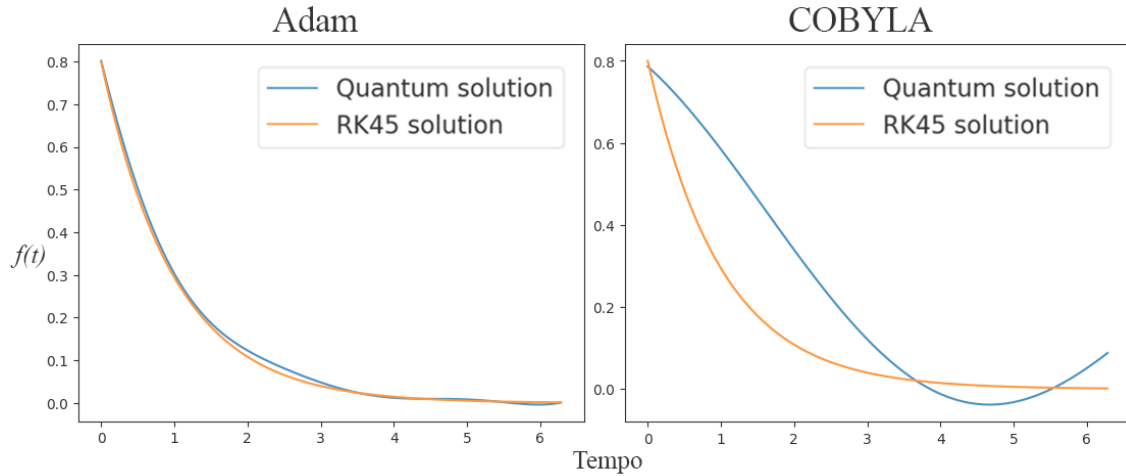


Figura 3.24: Confronto delle soluzioni dei due algoritmi con la soluzione del modello classico di Runge-Kutta.

	RSS	Funzione di costo
Adam	0.868×10^{-2}	0.261
COBYLA	1.828	2.685

Tabella 3.11: Valori numerici degli RSS e delle funzioni di costo associati ai risultati ottenuti dai due differenti algoritmi di ottimizzazione.

3.4 Test su hardware quantistico

Come accennato nella sezione 1.5, attraverso la piattaforma cloud di IBM, si ha la possibilità di accedere ad una selezione di dispositivi hardware con i quali è possibile testare il funzionamento dei propri algoritmi quantistici. Nella presente sezione, si vuole discutere l'implementazione di un circuito quantistico tra quelli ottimizzati in precedenza, su un dispositivo reale in modo da analizzarne i risultati e confrontarli con la soluzione dell'EDO ottenuta per via simulativa.

È importante evidenziare che il lavoro svolto in questa tesi non è incentrato sull'ottimizzazione dell'algoritmo in funzione di un particolare hardware quantistico. Di conseguenza, nel corso delle sperimentazioni precedentemente discusse, non sono stati approfonditi e tenuti in considerazione alcuni concetti necessari per un'integrazione ottimale, che avrebbero consentito di massimizzare la performance di esecuzione dei nostri algoritmi per un dato processore quantistico.

Per lo svolgimento di questa prova su dispositivo reale, è stato preso in considerazione il circuito quantistico a 6 qubit studiato nella sezione 3.2.4, con i relativi parametri ottenuti mediante l'ottimizzazione.

```

1 theta = ParameterVector("theta", 17)
2 t = Parameter("t")
3
4 psi = QuantumCircuit(6)
5
6 psi.ry(theta[0]*t + theta[1], 0)
7 psi.crx(theta[12]*t, 0, 1)
8 psi.ry(theta[2]*t + theta[3], 1)
9 psi.crx(theta[13]*t, 1, 2)
10 psi.ry(theta[4]*t + theta[5], 2)
11 psi.crx(theta[14]*t, 2, 3)
12 psi.ry(theta[6]*t + theta[7], 3)
13 psi.crx(theta[15]*t, 3, 4)
14 psi.ry(theta[8]*t + theta[9], 4)
15 psi.crx(theta[16]*t, 4, 5)
16 psi.ry(theta[10]*t + theta[11], 5)
17
18 best_param = [0.29904865, 0.18139089, 0.17993054, 0.197327, 0.58606573,
    ↪ -0.26180599, 0.28410509, 0.11913297, 0.22506643, 0.09970065, 0.23561782,
    ↪ 0.07704035, 0.55268409, 0.2573991, 0.47801607, 0.31228026, 0.1790344,
    ↪ 0.8801246, -0.01067926]

```

Codice 3.15: Circuito quantistico a 6 qubit precedentemente ottimizzato per risolvere l'EDO di un oscillatore armonico smorzato con $\gamma = 1.5$.

Per potersi interfacciare con un dispositivo quantistico reale di IBM, è stato necessario istanziare l'apposita classe `IBMPProvider`, che permette di selezionare il dispositivo con cui eseguire l'algoritmo.

```

1 provider = IBMPProvider(instance='ibm-q/open/main')
2 backend = provider.get_backend("ibm_perth",
    ↪ instance='ibm-q-research-2/uni-milano-bicoc-1/main')

```

Codice 3.16: Connessione al dispositivo quantistico reale.

L'oggetto `IBMPProvider` consiste in una classe di Qiskit responsabile della connessione con l'infrastruttura di calcolo quantistico di IBM e attraverso di essa viene definito

un backend, ossia il dispositivo quantistico specifico con il quale si desidera interagire. Nel nostro caso è stato scelto il dispositivo a 7 qubit `ibm_perth`.

```
1 q_instance = QuantumInstance(backend, shots=2000)
2 sampler = CircuitSampler(q_instance)
```

Codice 3.17: Definizione dell'oggetto con cui scambiare le informazioni tra algoritmo e dispositivo quantistico reale.

L'oggetto definito come `sampler`, si occupa invece di gestire aspetti come l'invio del circuito al backend, l'esecuzione delle porte quantistiche e la raccolta dei risultati delle misurazioni. Attraverso questi strumenti di base è stato possibile effettuare una comunicazione con il dispositivo fisico, in modo da poter eseguire le operazioni precedentemente assegnate alla funzione `evaluate_function` (3.3), per costruire la soluzione approssimata dell'EDO di un oscillatore armonico smorzato (con $\gamma = 1.5$).

Confrontando infine i risultati ottenuti dal dispositivo reale (curva blu) con quelli ricavati dalla simulazione (curva arancione), si può osservare una discordanza significativa che può essere spiegata attraverso alcuni concetti fondamentali legati all'architettura hardware considerata e alla modalità di implementazione di un algoritmo quantistico su tale sistema fisico.

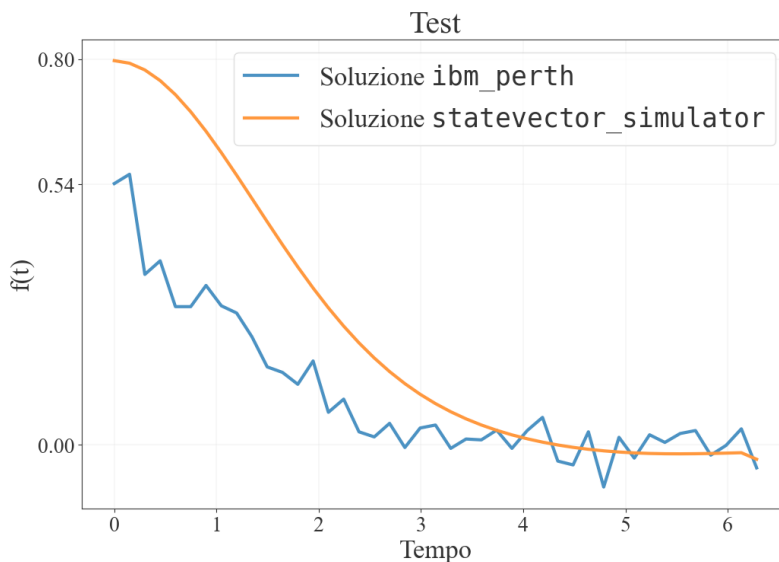


Figura 3.25: Confronto tra la soluzione ottenuta dal dispositivo quantistico e la soluzione ottenuta dalla simulazione.

Mapa di connettività dei qubit (o qubit layout):

Con mappa di connettività dei qubit, ci si riferisce alla disposizione fisica dei qubit e come questi siano organizzati e collegati tra loro all'interno del sistema quantistico

reale. La configurazione dei qubit può variare a seconda dell'architettura utilizzata e determina quali qubit possono essere entangled o interagire tra loro in modo diretto. Ogni processore quantistico ha una specifica configurazione di qubit fisici e potrebbe presentare limitazioni nella creazione di entanglement tra determinate coppie di qubit. Pertanto, considerare attentamente il layout dei qubit è fondamentale nella progettazione di algoritmi quantistici, in quanto l'esecuzione dei circuiti può essere influenzata negativamente se vengono eseguite operazioni di entanglement su qubit non fisicamente connessi tra loro nell'hardware reale. Nel nostro caso specifico, la mappa di connettività relativa all'architettura del dispositivo `ibm_perth` è rappresentata dal seguente schema:

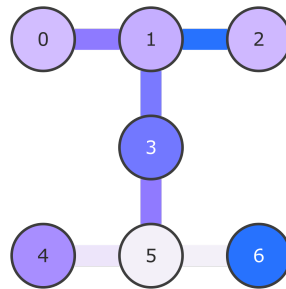


Figura 3.26: Mappa di connettività dei qubit del backend `ibm_perth`

Attraverso questa mappa è possibile osservare i tipi di connessioni tra i qubit e comprendere quale siano le coppie con cui è possibile instaurare l'entanglement. In relazione a ciò, si può subito notare come i gate `CRX`, utilizzati nel circuito descritto nel codice 3.15, non siano stati configurati in modo opportuno per adattarsi al meglio all'architettura del processore considerato.

Gate nativi (o basis gates):

I gate nativi si riferiscono ad un insieme limitato di quantum gates fondamentali che un dispositivo reale può supportare fisicamente, senza la necessità di ricorrere a decomposizioni o approssimazioni. Questi gate sono considerati il set di base del processore quantistico e sono direttamente implementabili sull'hardware del dispositivo. Attraverso l'utilizzo diretto di questa base, per la realizzazione di un circuito quantistico, è possibile ottenere risultati più accurati e affidabili nelle simulazioni e nell'esecuzione dei circuiti quantistici.

In particolare, i gate nativi del dispositivo `ibm_perth` sono rispettivamente, `CX`, `ID`, `RZ`, `SX` e `X`. Come si può invece notare dal codice 3.15, il circuito quantistico da noi considerato è stato realizzato mediante porte `RY` e `CRX`. Per tale ragione, l'oggetto `sampler`, precedentemente definito, ha dovuto ricorrere a processi di transpiling per rimanipolare il circuito in modo tale che potesse essere eseguito dall'hardware.

Transpiling:

Come appena accennato, con il termine transpiling, ci si riferisce al processo di ridefinizione e adattamento di un circuito quantistico sorgente, al fine di renderlo

Capitolo 4

Conclusioni

In questo lavoro di tesi si è voluta esplorare la progettazione di un algoritmo quantistico per la risoluzione di equazioni differenziali ordinarie, combinando tecniche di derivazione esatte di tipo quantistico con strumenti di ottimizzazione classica per la ricerca della soluzione ottimale.

Attraverso lo studio dell'EDO di un oscillatore armonico smorzato, sono emersi diversi risultati significativi. In particolare, è stato osservato che, l'aumento del numero di qubit nel sistema, porta a soluzioni più accurate, grazie al maggior numero di gradi di libertà a disposizione e all'aumento della dimensione dello spazio di Hilbert del sistema quantistico.

Un'altra variabile di fondamentale importanza nel migliorare la performance del modello è il numero di quantum gates dipendenti dal tempo. In particolare, l'aumento del numero di tali gate ha consentito di migliorare significativamente l'accuratezza nella ricerca del minimo della funzione di costo, anche con l'utilizzo di un singolo qubit.

Particolare attenzione è stata posta nel confronto tra due algoritmi di ottimizzazione dei parametri del circuito variazionale, ossia COBYLA e Adam, come implementati da Qiskit. COBYLA si è dimostrato più efficiente computazionalmente, a discapito di fornire soluzioni spesso non ottimali rispetto a quelle di Adam. In particolare, dalle sperimentazioni condotte, COBYLA ha presentato una certa sensibilità alla scelta dei parametri iniziali. Un'inizializzazione inadeguata di tali parametri può quindi compromettere la capacità dell'algoritmo nel convergere verso il minimo globale. Tale caratteristica può rappresentare un ostacolo significativo nella risoluzione di equazioni differenziali ordinarie generiche, in quanto potrebbe essere necessario effettuare diversi tentativi di calibrazione dei parametri iniziali in base all'equazione differenziale specifica, rendendo in questo modo il processo di ricerca del minimo, più laborioso e complesso.

È importante sottolineare come la performance del modello implementato in questa tesi dipenda fortemente sia dalla tipologia di circuito quantistico utilizzato che dall'efficacia degli algoritmi di ottimizzazione, nonché dalle risorse computazionali a disposizione, che pongono un limite sulla dimensione dei circuiti e sul numero di parametri. Pertanto, la difficoltà riscontrata nel raggiungere risultati ottimali in questa tesi non esclude la possibilità che l'implementazione di circuiti quantistici

diversi da quelli analizzati possa portare a una migliore adattabilità e a risultati più accurati, aprendo così ulteriori prospettive di ricerca futura.

Un ultimo aspetto riscontrato riguarda i risultati ottenuti con il test su dispositivo reale. In particolare, si osserva che i simulatori quantistici mostrano una buona compatibilità con i metodi classici. Tuttavia, ciò non risulta per i backend quantistici reali a causa del rumore presente. Per migliorare tali risultati, si potrebbero implementare circuiti quantistici ottimizzati per il backend specifico utilizzato o provare anche ad usare metodi di *error correction* [23] ed *error mitigation* (seppur siano ad oggi tecniche ancora emergenti che richiederebbero ulteriori studi per funzionare al meglio).

Il metodo sviluppato nel presente lavoro di tesi per la risoluzione di equazioni differenziali ordinarie non rappresenta ancora un'alternativa valida ai modelli classici attualmente in uso. Tuttavia, le sperimentazioni condotte e i risultati ottenuti costituiscono una base di partenza per meglio comprendere il problema dell'approccio quantistico, raffinare il modello e studiare implementazioni più efficienti.

Sarebbe possibile ad esempio estendere lo studio ad equazioni differenziali più complesse. Le sperimentazioni condotte nella tesi si sono focalizzate su un'EDO specifica, ossia quella di un oscillatore armonico smorzato. Esplorare altre tipologie di equazioni differenziali potrebbe fornire una visione più completa delle capacità e delle limitazioni dell'approccio quantistico.

Infine, un'altra direzione di ricerca interessante, riguarda la realizzazione di circuiti quantistici variazionali su misura per le architetture quantistiche oggi esistenti. Attualmente, i computer quantistici disponibili sono soggetti a limitazioni in termini di numero di qubit, connettività e rumore. Sviluppare circuiti quantistici adattati a queste specifiche limitazioni consentirebbe di rappresentare in maniera accurata, le soluzioni delle EDO attraverso dispositivi quantistici reali. Per le prospettive future ci si aspetta uno sviluppo di qubit meno rumorosi e sempre più numerosi all'interno dei dispositivi reali, insieme a tecniche di mitigazione e correzione degli errori sempre più efficienti. In queste condizioni gli algoritmi quantistici avranno il potenziale per superare quelli classici, anche nella risoluzione di equazioni differenziali.

Bibliografia

- [1] John Preskill. *Why I Called It ‘Quantum Supremacy’*. 2019. URL: <https://www.quantamagazine.org/john-preskill-explains-quantum-supremacy-20191002/>.
- [2] F. Arute et al. «Quantum supremacy using a programmable superconducting processor». In: *Nature* 574 (2019). URL: <https://doi.org/10.1038/s41586-019-1666-5>.
- [3] Y. Kim, A. Eddins e S. et al. Anand. «Evidence for the utility of quantum computing before fault tolerance». In: *Nature* (2023). URL: <https://doi.org/10.1038/s41586-023-06096-3>.
- [4] Michael A. Nielsen e Isaac L. Chuang. *Quantum Computation and Quantum Information*. 2022.
- [5] P. Krantz et al. «A quantum engineer’s guide to superconducting qubits». In: *Applied Physics Reviews* 6 (2019). URL: <https://doi.org/10.1063/1.5089550>.
- [6] *IBM Quantum Experience*. URL: <https://quantum-computing.ibm.com/>.
- [7] W.H. Press et al. *Numerical Recipes in C: The Art of Scientific Computing, Second edition*. Cambridge University Press, 1992. Cap. 16. ISBN: 0-521-75033-4.
- [8] Jiyuan Tu, Kiao Inthavong e Goodarz Ahmadi. «Numerical Methods and Its Solution». In: set. 2013, pp. 167–231. ISBN: 978-94-007-4487-5. DOI: 10.1007/978-94-007-4488-2_7.
- [9] *Simple Euler method and its modifications*. URL: https://tlakoba.w3.uvm.edu/math337/notes_1.pdf.
- [10] *How to solve harmonic oscillator differential equation*. URL: <https://math.stackexchange.com/questions/1575291/how-to-solve-harmonic-oscillator-differential-equation-dfrac2xdt2-d>.
- [11] *qiskit.algorithms.optimizers*. URL: <https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.html#module-qiskit.algorithms.optimizers>.
- [12] *Linear Programming*. URL: https://handwiki.org/wiki/Linear_programming.
- [13] Ping-Qi PAN. *Linear Programming Computation*. URL: <https://doi.org/10.1007/978-981-19-0147-8>.
- [14] *COBYLA*. URL: <https://handwiki.org/wiki/COBYLA>.
- [15] Jeff M. Phillips. «Gradient Descent». In: *Mathematical Foundations for Data Analysis*. Cham: Springer International Publishing, 2021, pp. 125–142. URL: https://doi.org/10.1007/978-3-030-62341-8_6.

- [16] Abodunrin AbdulGafar Adigun e Chika Yinka-Banjo. «Comparing Stochastic Gradient Descent and Mini-batch Gradient Descent Algorithms in Loan Risk Assessment». In: *Informatics and Intelligent Applications*. URL: https://doi.org/10.1007/978-3-030-95630-1_20.
- [17] *Complete Guide to Adam Optimization*. URL: <https://towardsdatascience.com/complete-guide-to-adam-optimization-1e5f29532c3d>.
- [18] Diederik P. Kingma e Jimmy Ba. «Adam: A Method for Stochastic Optimization». In: (2017).
- [19] Oleksandr Kyriienko, Annie E. Paine e Vincent E. Elfving. «Solving nonlinear differential equations with differentiable quantum circuits». In: *Physical Review A* 103.5 (2021). URL: <https://doi.org/10.1103/physreva.103.052416>.
- [20] *Qiskit Gradient Framework*. URL: https://qiskit.org/documentation/tutorials/operators/02_gradients_framework.html.
- [21] Gavin E. Crooks. «Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition». In: (2019). eprint: 1905.13311. URL: <https://arxiv.org/abs/1905.13311>.
- [22] *Parameter-shift Rule*. URL: https://pennylane.ai/qml/glossary/parameter_shift/.
- [23] Google Quantum AI. «Suppressing quantum errors by scaling a surface code logical qubit». In: *Nature* (2023). URL: <https://doi.org/10.1038/s41586-022-05434-1>.