Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Università degli studi di Padova

Dipartimento di Ingegneria dell'informazione - 'DEI'

Tesi Triennale in Ingegneria Informatica

Progettazione e sviluppo di un sistema di sincronia per la distribuzione dell'entanglement

Relatore
Giuseppe Vallone

Correlatore
Francesco Santagiustina
Alberto De Toni

Laureando
Federico Muscarà

Matricola
2001594

Anno Accademico
2022-2023

Ringraziamenti

Ringrazio il Relatore, Prof. Vallone, per avermi concesso la possibilità di inserire il mio progetto di tesi in un contesto sperimentale, permettendomi così di ampliare i miei orizzonti accademici. Un sentito ringraziamento ai Dottori De Toni e Santagiustina, che hanno saputo guidarmi durante questi mesi, supervisionando la stesura di questo progetto, rispondendo prontamente a tutti i miei quesiti.

Abstract

L'entanglement è una proprietà di alcuni sistemi fisici che presentano correlazioni non-locali. Oltre al suo ruolo in fisica fondamentale, negli ultimi decenni è diventato una risorsa chiave nelle nuove tecnologie di comunicazione e computazione quantistica. La generazione e distribuzione di fotoni entangled è infatti alla base del futuro internet quantistico, permettendo sia il trasferimento di stati quantistici tramite teletrasporto quantistico che l'implementazione di numerosi protocolli di crittografia invulnerabili alle minacce computazionali moderne. Questo progetto di tesi si inserisce nel contesto di un esperimento di distribuzione dell'entanglement e mira a realizzare un sistema di sincronia tra la sorgente di fotoni entangled e i ricevitori. In primo luogo è presentata un'interfaccia interattiva implementata per agevolare l'utilizzo dell'analizzatore di spettro Signal Hound BB60C, che ha dimostrato di essere uno strumento prezioso per la gestione e l'analisi di segnali elettrici, contribuendo a confermare la stabilità di un sistema di sincronia in frequenza implementato tramite trasmissione ottica di impulsi. La seconda parte del progetto presenta invece le fasi di realizzazione di un software di sincronia temporale tramite correlazione incrociata, realizzato attraverso script Python, atto a compensare il ritardo tra sequenze di tempi di rilevamento di fotoni provenienti da coppie entangled, ma rilevate con ricevitori localizzati in ambienti distanti. Sincronizzando le due sequenze temporali è possibile individuare i rilevamenti associati a coppie entangled e procedere nell'analisi dei dati necessaria ai vari protocolli.

Indice

1 Prefazione

Il presente progetto di tesi si colloca all'interno di un ampio contesto: l'obiettivo principale è realizzare un sistema di distribuzione dell'entanglement su lunghe distanze tramite fibra ottica. All'interno di questa tesi sono presentati due software realizzati a supporto della realizzazione di questo apparato sperimentale, e in particolare del suo sistema di sincronia. Uno fornisce un'interfaccia grafica e di controllo per un analizzatore di spettro mentre l'altro consente di stimare con precisione il ritardo tra sequenze di rilevamenti di fotoni provenienti da coppie entangled, distribuite a ricevitori situati in ambienti distanti.

Al fine di comprendere appieno il contesto in cui si inserisce questo progetto, il capitolo 2 consente di acquisire familiarità con i concetti fondamentali della Meccanica e dell'Ottica Quantistica, definendo inoltre la casistica sperimentale affrontata in concreto, parte integrante del progetto europeo *Quantera SECRET* (SECuRe quantum communication based on Energy-Time/time-bin entanglement).

Dopo aver introdotto nel capitolo 3 alcuni strumenti matematici di analisi di Fourier necessari utilizzati nel seguito per l'analisi dei dati sperimentali, il capitolo 4 illustra le fasi di progettazione dell'interfaccia grafica che garantisce un semplice e rapido utilizzo dell'analizzatore di spettro *Signal Hound BB60C*. Quest'ultima, di cui è fornito il codice nella sezione A di questo documento, si è rivelata fondamentale per dimostrare la corretta trasmissione di un segnale di sincronia su lunghe distanze.

Infine, il processo di creazione di un sistema di stima del ritardo temporale tra sequenze di rilevamenti di fotoni provenienti da coppie entangled ma registrati da ricevitori distanti è esposto nel capitolo 5. Il software realizzato, sfruttando la correlazione temporale dei fotoni entangled, permette di allineare temporalmente le sequenze di rilevamenti con una precisione superiore al centinaio di picosecondi, al fine di poter svolgere successive le analisi dei dati necessarie all'implementazione dei vari protocolli.

2

Entanglement quantistico: introduzione e requisiti di sincronia per la distribuzione di fotoni entangled

Il software sviluppato e descritto in questo progetto di tesi svolge un ruolo di supporto per esperimenti di comunicazione quantistica, in particolare per quanto riguarda la distribuzione dell'entanglement. L'entanglement è un fenomeno che lega strettamente in un legame di sovrapposizione particelle quantistiche, anche se separate da distanze considerevoli. Tuttavia, sfruttare l'entanglement richiede infrastrutture complesse e strumenti adeguati. In particolare il software qui descritto è stato progettato per fornire un ambiente interattivo e intuitivo per agevolare l'implementazione dell'esperimento di distribuzione di informazione legata all'entanglement illustrato alla sezione 2.3.1. È importante sottolineare che il caso sperimentale in questione fa parte del progetto europeo *Quantera SECRET* (SECuRe quantum communication based on Energy-Time/time-bin entanglement). Il team di ricerca coinvolto comprende il Prof. Vallone, il Prof. Villoresi e i dottori Agnesi, Santagiustina e Vijayadharan.
Al fine di comprendere a pieno il contesto in cui questo progetto si inserisce, è necessario prima approfondire i concetti base di Meccanica e Ottica Quantistica.

2.1 Breve introduzione alla Meccanica Quantistica e all'entanglement

Il capitolo introdurrà molto brevemente alcune nozioni di base legate a Meccanica e Ottica Quantistica per lettori aventi background diversi ma desiderosi di leggere questo progetto di tesi. Poiché un'esposizione completa e rigorosa di tali argomenti non è conforme allo scopo di questa

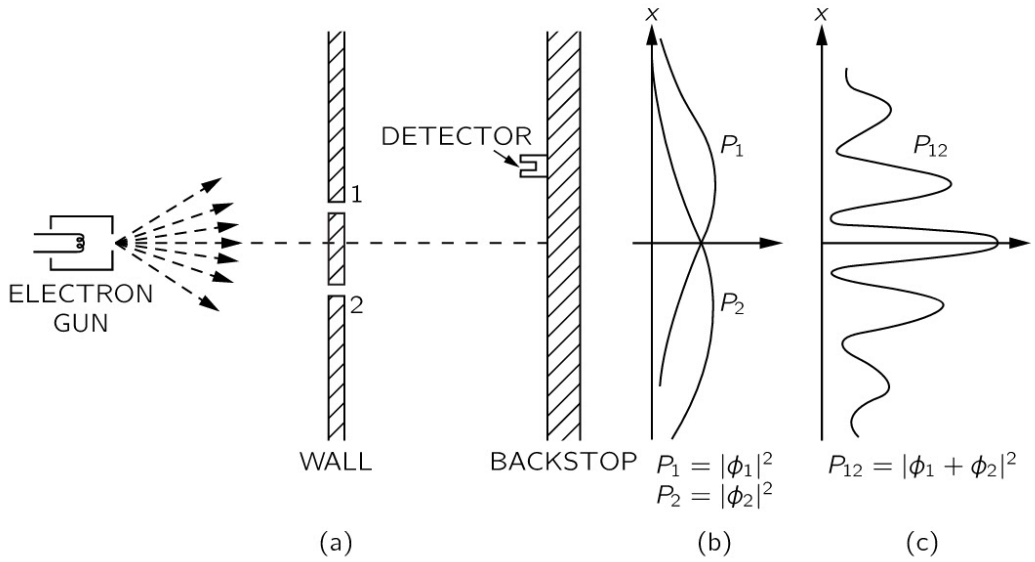


Figura 2.1: Esperimento di interferenza a doppia fenditura con elettroni. Distribuzioni di probabilità P_1 e P_2 di arrivi degli elettroni sullo schermo dopo aver chiuso rispettivamente le fenditure 2 e 1. Distribuzione di probabilità osservata P_{12} quando entrambe le fenditure sono aperte. *Crediti: Figura da [1]*

Appendice, si rimanda il lettore interessato ad alcuni riferimenti bibliografici. Un corso base di Meccanica Quantistica è fornito nel riferimento [1], contenente l'analisi del *Double slit experiment* (Esperimento della doppia fenditura). Un'introduzione all'Informazione Quantistica e Calcolo Quantistico è riscontrabile nel riferimento [2]. Per quanto riguarda Ottica Quantistica suggeriamo le opere [3, 4, 5, 6, 7]. Tali riferimenti bibliografici, sono stati ampiamente utilizzati come fonti per delineare l'esposizione che segue, che non pretende di introdurre alcuna novità a quanto già assodato.

2.1.1 Principi della meccanica quantistica

L'Ottica Quantistica si basa sui principi della Meccanica Quantistica, una teoria che, sebbene finora non conciliata con successo con la relatività generale, ha già dimostrato il suo valore per le previsioni estremamente accurate che consente di ottenere, sulle quali molte delle nostre tecnologie trovano le proprie fondamenta.

L'esperimento della doppia fenditura di Young (Young's double-slit experiment) ha dimostrato la dualità onda-corpuscolo della luce, nonché la natura fondamentalmente probabilistica dei fenomeni della Meccanica Quantistica. In particolare, secondo Feynman questo esperimento "ha in sé il cuore della Meccanica Quantistica". Si veda la figura 2.1: la raffigurazione illustra una sorgente che invia elettroni attraverso due fenditure (di opportune dimensioni) verso uno schermo che ne consente la rilevazione. Osserviamo le distribuzioni P1, P2 e P12 quando, rispettivamente, solo la fessura 1 è aperta, solo la fessura 2 è aperta, entrambe le fenditure sono aperte. P12 risulta essere differente dalla somma di P1 e P2, vale a dire: gli elettroni provenienti dalle

fenditure 1 e 2 sembrano interferire. Ciò che risulta sorprendente è che rileviamo pattern di interferenza anche nel momento in cui vengono inviati gli elettroni uno dopo l'altro attraverso le fenditure! In qualche modo dunque una particella interferisce con sé stessa. Tale fenomeno è stato spiegato attraverso la nozione di sovrapposizione (*superposition*).

Principio 2.1.1. ***Principio di sovrapposizione***
Ad ogni sistema fisico è associato uno spazio di Hilbert $\mathscr{H}$. Lo stato del sistema è definito in qualsiasi momento dal vettore normalizzato $|\psi(t)\rangle \in \mathscr{H}$. [*]

Questo principio implica che ogni combinazione lineare $|\psi\rangle = \sum_i C_i |\psi_i\rangle$ dei vettori di stato $|\psi_i\rangle$, con C_i complessi e $\sum_i |C_i|^2 = 1$, è uno stato vettoriale valido, che rappresenta un possibile sistema fisico. Nel nostro caso: $|\phi_{12}\rangle = \frac{1}{\sqrt{2}}|\phi_1\rangle + \frac{1}{\sqrt{2}}|\phi_2\rangle$, ovvero lo stato dell'elettrone risulta, durante la sua propagazione, dalla somma degli stati di ipotetici elettroni provenienti da ciascuna fenditura.

Postulato 2.1.2. ***Postulato di misura (o regola di Born)***

1. *Un operatore lineare autoaggiunto $\widehat{A}$ su $\mathscr{H}$ è associato a qualsiasi variabile dinamica misurabile A : $\widehat{A}$ si dice osservabile associato ad A. Per $\mathscr{H}$ di dimensione finita con base ortonormale, $\widehat{A}$ è una matrice hermitiana.*

2. *Sia $|\psi\rangle$ lo stato del sistema quando una misura di A è compiuta. Qualunque sia $|\psi\rangle$, gli unici risultati possibili della misura sono gli autovalori (reali) a_α dell'osservabile $\widehat{A}$.*

3. *Sia $\widehat{P_\alpha}$ il proiettore sul sottospazio associato all'autovalore a_α. La probabilità di trovare il valore a_α quando si misura A è :*

$$Prob(a_\alpha) = ||\psi_\alpha||^2 \ \ dove \ \ |\psi_\alpha\rangle = \widehat{P_\alpha}|\psi\rangle$$

4. *Subito dopo una misura di A che fornisce il valore a_α, il nuovo stato del sistema $|\psi'\rangle$ è*

$$|\psi'\rangle = \frac{\psi_\alpha}{||\psi_\alpha||} \, .$$

[*]Usiamo la notazione bra-ket di Dirac. L'azione di un funzionale lineare su un vettore in $\mathscr{H}$ è indicata come un prodotto interno $\langle\phi||\psi\rangle$ costituito da una parte destra (il ket) $|\psi\rangle$, e una sinistra (il bra) $\langle\phi|$, i quali sono vettori di $\mathscr{H}$ e del suo spazio duale, secondo il teorema di rappresentazione di Riesz. Un operatore lineare (rappresentato da una matrice quadrata di dimensioni finite) applicato a un ket si scrive $\widehat{B}|\phi\rangle$ e restituisce un ket. Il prodotto esterno $|\phi\rangle\langle\psi|$ definisce un operatore lineare su $\mathscr{H}$.

Pertanto, lo stato vettoriale descrive la distribuzione di probabilità delle variabili misurabili del sistema (come posizione, quantità di moto, energia e rotazione). Per questo abbiamo bisogno della normalizzazione, poiché le probabilità devono fornire somma pari a uno. Inoltre, secondo l'interpretazione di Copenaghen della Meccanica Quantistica, quando misuriamo uno stato esso collassa in uno degli autostati dell'osservabile. Quindi, se proviamo a misurare da quale fenditura è passato l'elettrone, lo troveremo in una sola posizione e non interferirà più con sé tesso. E' molto interessante notare che, se si possiede una conoscenza parziale della fenditura attraversata dalla particella, si otterrà una parziale riduzione dell'interferenza (decoerenza), proporzionale all'informazione ottenuta. Quindi la misurazione (che è un tipo di interazione che implica il trasferimento di informazioni) modifica sempre il sistema, a meno che non sia già un autostato dell'osservabile misurato. Ma cosa si può dire riguardo l'evoluzione "naturale" di uno stato quantistico?

Principio 2.1.3. ***Evoluzione temporale (o equazione di Schrödinger)***
Sia $|\psi(t)\rangle$ lo stato di un sistema all'istante di tempo t. Fintantoché il sistema non viene misurato la sua evoluzione segue:

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \widehat{H}|\psi(t)\rangle$$

dove $\widehat{H}$ è l'energia osservabile, anche definita Hamiltoniana del sistema.

L'esperimento della doppia fenditura ci ha permesso di introdurre i principi della meccanica quantistica e intravedere le profonde relazioni tra sovrapposizione, misurazione e informazione. Ci concentreremo ora sulla quantità unitaria dell'informazione quantistica: il qubit.

2.1.2 Il qubit in polarizzazione

Per implementare i protocolli di Informazione Quantistica abbiamo bisogno di un portatore di informazione. Qualsiasi particella può contenere informazione ma, per ragioni pratiche, questo portatore sarà sempre la luce. La luce viene emessa e interagisce in unità di energia quantizzata chiamata fotone. Un modo semplice per codificare informazioni quantistiche in un fotone è sfruttare la sua polarizzazione. La polarizzazione è una nozione dell'elettromagnetismo classico che rappresenta l'orientamento del campo elettrico e magnetico nel piano trasversale alla direzione di propagazione. In ambito quantistico, bisogna considerare che se un fotone passa attraverso una lamina birifrangente (la cui proprietà è quella di separare le componenti ortogonali della luce) il fotone non può dividersi: lo troveremo dunque con una sola polarizzazione (o quella ortogonale ad essa), quando misurata, con probabilità dipendenti dalla precedente polarizzazione (se ripetiamo l'esperimento con numerosi fotoni si tende al classico beam splitting, secondo il principio di corrispondenza). Dopo aver superato la lastra birifrangente e prima di misurarla il fotone sarà (generalmente) in uno stato di sovrapposizione.

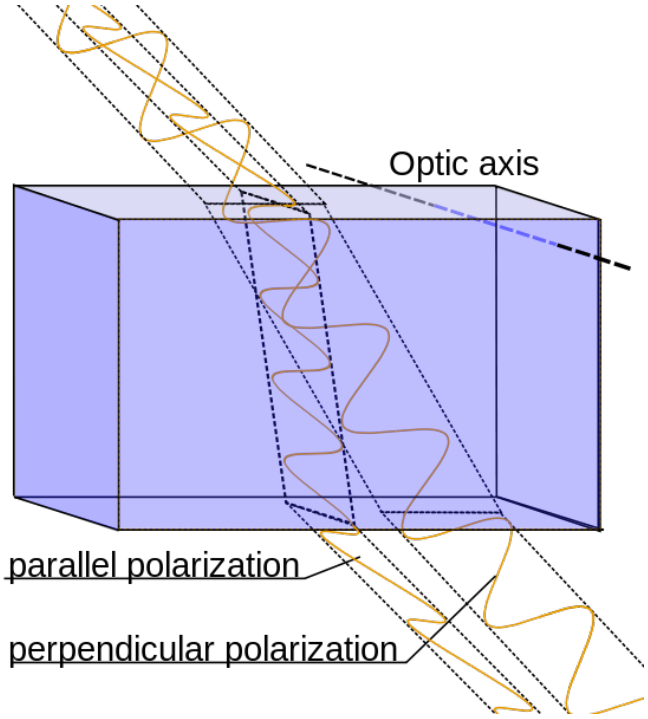


Figura 2.2: Cristallo birifrangente che separa le due polarizzazioni ortogonali del fascio incidente.

Siano $\vec{x}$ e $\vec{y}$ una base ortonormale del piano perpendicolare alla direzione di propagazione $\vec{z}$: la polarizzazione lineare può essere descritta come

$$\vec{E} = E\cos(\omega t + \phi)(\vec{x}\cos\theta + \vec{y}\sin\theta) \tag{2.1}$$

e la polarizzazione circolare come

$$\vec{E} = E(\vec{x}\cos(\omega t + \phi) + \vec{y}\sin(\omega t + \phi)) \tag{2.2}$$

dove ω è la frequenza angolare, θ l'angolo di polarizzazione e ϕ la fase ottica che potrebbe essere impostata a zero modificando l'origine del tempo. Questo porterà alla formulazione della Meccanica Quantistica due basi dello spazio di Hilbert, una lineare: $\{|H\rangle, |V\rangle\}$ (polarizzazione orizzontale e verticale) e una circolare: $\{|R\rangle, |L\rangle\}$ (polarizzazione circolare destra e sinistra). In base al principio di sovrapposizione, qualsiasi stato di polarizzazione può essere descritto come la combinazione lineare complessa dei vettori unitari ortogonali:

$$|\psi\rangle = c_H|H\rangle + c_V|V\rangle = c_R|R\rangle + c_L|L\rangle \tag{2.3}$$

in particolare, valgono le seguenti relazioni:

$$|R\rangle = \frac{1}{\sqrt{2}}(|V\rangle + i|H\rangle) \;;\; |L\rangle = \frac{1}{\sqrt{2}}(|V\rangle - i|H\rangle) \tag{2.4}$$

$$|V\rangle = \frac{1}{\sqrt{2}}(|R\rangle + |L\rangle) \quad ; \quad |H\rangle = \frac{1}{\sqrt{2}}(-i|R\rangle + i|L\rangle) \tag{2.5}$$

e perfettamente in accordo con la rappresentazione classica in cui è possibile visualizzare la polarizzazione circolare come somma di due onde polarizzate ortogonalmente con uno sfasamento di $\frac{\pi}{4}$ qui rappresentato dal coefficiente $i = e^{i\frac{\pi}{4}}$, che può essere realizzato in pratica utilizzando una piastra a quarto d'onda.
Siccome in ogni base data usiamo due numeri complessi per descrivere la polarizzazione di un fotone, si può pensare che siano necessari quattro parametri reali. In realtà, a causa della condizione di normalizzazione e del fatto che solo lo sfasamento relativo tra le due polarizzazioni sovrapposte è rilevante, possiamo parametrizzare lo stato come:

$$|\psi_{pol-qubit}\rangle = \cos\frac{\theta}{2}|H\rangle + e^{i\varphi}\sin\frac{\theta}{2}|V\rangle \tag{2.6}$$

dove $0 \leq \theta \leq \pi$ e $0 \leq \varphi < 2\pi$ possono essere reinterpretate come le coordinate polari di un punto su una sfera che rappresenta tutte le possibili polarizzazioni, denominata sfera di Poincaré. Ora è evidente che sono necessarie solo due coordinate reali per descrivere uno stato di uno spazio di Hilbert, con apparentemente due dimensioni complesse o quattro reali, perché gli stati sono effettivamente rappresentati in modo univoco da raggi nello spazio proiettivo di Hilbert (poiché il ridimensionamento non ha importanza nel momento in cui dobbiamo normalizzare il vettore di stato quando si torna alle probabilità), e fino a uno sfasamento (poiché la fase è importante solo quando i vettori di stato vengono sommati). È possibile realizzare un'astrazione del sistema fisico reale e considerare un generico bit quantistico (o *qubit*) nello spazio di stati generato dalle basi $\{|0\rangle, |1\rangle\}$. Se un bit classico è uno scalare con due possibili valori (che rappresentano due possibili stati), un qubit è definito da un punto sulla sfera di Bloch (che rappresenta la sovrapposizione di due stati!)

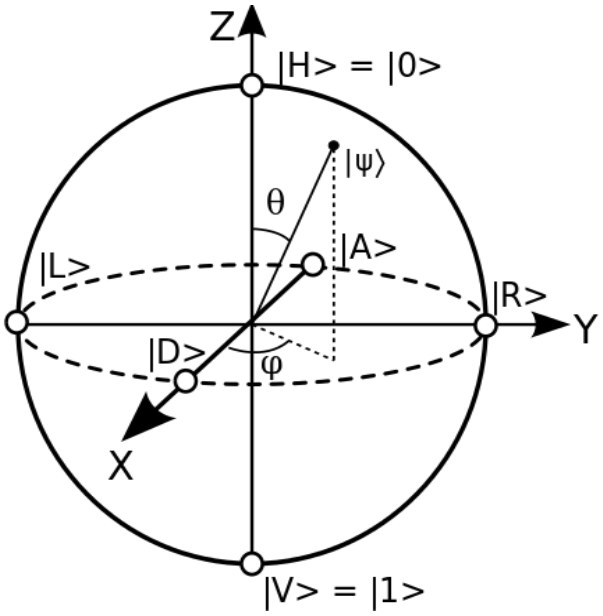


Figura 2.3: La sfera di Poincaré-Bloch ci permette di visualizzare i diversi stati quantistici di polarizzazione ($|D\rangle$ e $|A\rangle$ rappresentano la polarizzazione diagonale e anti-diagonale), o in generale un qubit. Possiamo vedere θ come la "proporzione" di $|0\rangle$ e $|1\rangle$ nella sovrapposizione mentre φ determina il loro sfasamento relativo.

Immaginiamo ora di provare a misurare la polarizzazione di un fotone nello stato

$$|\psi_{pol-qubit}\rangle = \cos\frac{\theta}{2}|H\rangle + e^{i\varphi}\sin\frac{\theta}{2}|V\rangle$$

nella base verticale/orizzontale. Rileveremo una sua polarizzazione orizzontale con probabilità $(\cos\frac{\theta}{2})^2$ e verticale con probabilità $(\sin\frac{\theta}{2})^2$. Notiamo come queste probabilità non dipendano da φ, dunque non possediamo informazione alcuna sulle "proporzioni" della polarizzazione precedente in base circolare o diagonale, e lo stato collassato può ora essere scritto come sovrapposizioni ugualmente pesate di $|R\rangle$ e $|L\rangle$ oppure di $|D\rangle$ e $|A\rangle$, rispettivamente nella nella base circolare e diagonale. Formalmente parlando, ogni vettore di una base ha proiezioni di uguale lunghezza su tutti i vettori dell'altra base. In questi casi, si dice che le basi sono *mutuamente imparziali*.

2.1.3 Entanglement e non-località

Finora abbiamo visto come una particella (con una sola coppia binaria osservabile, nel caso del qubit) viene descritta nel formalismo della Meccanica Quantistica. Introdurremo ora i sistemi quantistici di più particelle. Questo ci permetterà anche di introdurre il concetto di sistemi entangled e di mostrare come essi possano produrre correlazioni non locali.

Sistemi compositi

Supponiamo di voler rappresentare un sistema di due particelle con stati $\phi_A \in \mathscr{H}_A$ e $\phi_B \in \mathscr{H}_B$, entrambi qubit. Ciascuna particella necessita di uno spazio di Hilbert a due dimensioni per essere rappresentata, quindi possiamo aspettarci che il nostro sistema globale si trovi in uno spazio di Hilbert $\mathscr{H}_{AB}$ tale per cui $dim(\mathscr{H}_{AB}) = dim(\mathscr{H}_A) \times dim(\mathscr{H}_B) = 4$.

Postulato 2.1.4. *Lo spazio di Hilbert di un sistema composito è il prodotto tensore dello spazio di Hilbert degli spazi degli stati associati ai sistemi componenti.*

Nel nostro caso $\mathscr{H}_{AB} = \mathscr{H}_A \otimes \mathscr{H}_B$. Supponiamo che $\{|0_A\rangle, |1_A\rangle\}$ sia una base ortonormale di $\mathscr{H}_A$ e $\{|0_B\rangle, |1_B\rangle\}$ una di $\mathscr{H}_B$. Una base di $\mathscr{H}_A \otimes \mathscr{H}_B$ è data da :

$$|X_{00}\rangle = |0_A\rangle \otimes |0_B\rangle, |X_{01}\rangle = |0_A\rangle \otimes |1_B\rangle, |X_{10}\rangle = |1_A\rangle \otimes |0_B\rangle, |X_{11}\rangle = |1_A\rangle \otimes |1_B\rangle \quad (2.7)$$

Ora, se gli stati componenti $|\phi_A\rangle$ e $|\phi_B\rangle$ possono essere riscritti come:

$$|\phi_A\rangle = \lambda_A |0_A\rangle + \mu_A |1_A\rangle \text{ and } |\phi_B\rangle = \lambda_B |0_B\rangle + \mu_B |1_B\rangle$$

il sistema composito risultante deve essere:

$$|\phi_A\rangle \otimes |\phi_B\rangle = \lambda_A\lambda_B |X_{00}\rangle + \lambda_A\mu_B |X_{01}\rangle + \mu_A\lambda_B |X_{10}\rangle + \mu_A\mu_B |X_{11}\rangle \ . \quad (2.8)$$

Questa combinazione lineare si estende su tutto $\mathscr{H}_A \otimes \mathscr{H}_B$? Un generico sistema $|\psi\rangle \in \mathscr{H}_A \otimes \mathscr{H}_B$ può essere riscritto come:

$$|\psi\rangle = \alpha_{00} |X_{00}\rangle + \alpha_{01} |X_{01}\rangle + \alpha_{10} |X_{10}\rangle + \alpha_{11} |X_{11}\rangle \quad \text{and,} \quad (2.9)$$

$$|\psi\rangle = |\phi_A\rangle \otimes |\phi_B\rangle \quad \Leftrightarrow \quad \alpha_{00}\alpha_{11} = \alpha_{01}\alpha_{10} \, . \quad (2.10)$$

Pertanto, solo un sottoinsieme di $\mathscr{H}_A \otimes \mathscr{H}_B$ può essere espresso come $|\phi_A\rangle \otimes |\phi_B\rangle$, gli stati corrispondenti sono chiamati *stati separabili*, tutti gli altri sono chiamati *stati entangled*.

Stati entangled

Si consideri un esempio di stato entangled:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0_A\rangle \otimes |0_B\rangle + |1_A\rangle \otimes |1_B\rangle) \, , \quad (2.11)$$

prima della misurazione, non possiamo dire quali saranno gli stati collassati dei qubit, ma quando misuriamo uno dei qubit (in qualsiasi base), l'altro collasserà immediatamente nello stesso

stato finale. Questo, nonostante la distanza arbitraria che separa le due particelle. Si noti che poiché il risultato della prima misurazione è casuale, questo fenomeno non ci consente di trasmettere informazioni più velocemente della velocità della luce. Consideriamo la base Hadamard, composta da $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ e $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Si ha anche

$$\begin{aligned}\frac{1}{\sqrt{2}}(|+_A\rangle \otimes |+_B\rangle + |-_A\rangle \otimes |-_B\rangle) &= \frac{1}{\sqrt{2}}\Bigg(\frac{1}{2}(|0_A\rangle + |1_A\rangle) \otimes (|0_B\rangle + |1_B\rangle) + \\ &\qquad \frac{1}{2}(|0_A\rangle - |1_A\rangle) \otimes (|0_B\rangle - |1_B\rangle)\Bigg) \\ &= \frac{1}{2\sqrt{2}}\Bigg(|0_A\rangle \otimes |0_B\rangle + |0_A\rangle \otimes |1_B\rangle + |1_A\rangle \otimes |0_B\rangle + \\ &\qquad |1_A\rangle \otimes |1_B\rangle + |0_A\rangle \otimes |0_B\rangle - |0_A\rangle \otimes |1_B\rangle - \\ &\qquad |1_A\rangle \otimes |0_B\rangle + |1_A\rangle \otimes |1_B\rangle\Bigg) \\ &= \frac{1}{2\sqrt{2}}(2|0_A\rangle \otimes |0_B\rangle + 2|1_A\rangle \otimes |1_B\rangle) = |\Phi^+\rangle\end{aligned} \tag{2.12}$$

quindi misurando un sistema nella base Hadamard proietteremo lo stato dell'altro sistema su quello corrispondente al nostro risultato. In realtà lo stesso vale per qualsiasi base di qubit, dati infatti due generici stati ortogonali $|\psi^{\theta,\varphi}\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$ and $|\psi^{\pi-\theta,\pi-\varphi}\rangle = \cos\frac{\pi-\theta}{2}|0\rangle + e^{i(\pi-\varphi)}\sin\frac{\pi+\theta}{2}|1\rangle$ si può mostrare che

$$\frac{1}{\sqrt{2}}\left(|\psi_A^{\theta,\varphi}\rangle \otimes |\psi_B^{\theta,\varphi}\rangle + |\psi_A^{\pi-\theta,\pi-\varphi}\rangle \otimes |\psi_B^{\pi-\theta,\pi-\varphi}\rangle\right) = |\Phi^+\rangle \tag{2.13}$$

quindi potremmo dire che la scelta della base di misura di un sottosistema determina il collasso dell'altro sottosistema in uno degli stati che compongono quella base, il che mostra tutta la peculiarità degli stati entangled.

Il paradosso EPR e la non-località

All'alba della Meccanica Quantististica, l'entanglement causò importanti dibattiti. Questi vertevano principalmente sulla possibilità che il risultato apparentemente casuale delle misure sui sistemi quantistici potesse venire spiegata tramite l'ausilio di proprietà predeterminate ma nascoste (le cosiddette variabili nascoste). Albert Einstein, Boris Podolsky e Nathan Rosen, nel loro famoso articolo del 1935: *'Can Quantum-Mechanical Description of Physical Reality Be Considered Complete'?* dimostrarono come l'esistenza di sistemi entangled come quello descritto nel paragrafo precedente portasse a delle predizioni incompatibili con i principi di *località* e *realismo* [8], concludendo che la teoria quantistica dovesse essere considerata incompleta. Ne seguì un dibattito con Niels Bohr [9], ma fu soltanto nel 1964, quando John Stewart Bell de-

rivò dei limiti alle correlazione tra misure effettuate su sistemi fisici distanti sotto le ipotesi di località e realismo che la questione fu portata sul piano sperimentale [10]. Questo genere di limiti vennero chiamati *disugualgianze di Bell*, e la più utilizzata tra queste è la disuguaglianza di Clauser-Horne-Shimony-Holt (CHSH) [11], che si applica a sistemi bipartiti, ovvero divisi in due sottosistemi separati che sono misurati da due osservatori, tradizionalmente chiamati Alice e Bob. Ciascuno di essi sceglie casualmente una tra due misure da effettuare: $x, y \in \{0, 1\}$ saranno rispettivamente la scelta della misura di Alice e Bob. Gli esiti delle misure di Alice e Bob, a e b sono variabili dicotomiche, ovvero $a, b \in \{-1, +1\}$. La correlazione tra le variabili a e b, condizionata alla scelta di misure (x, y), è definita come

$$\langle a_x b_y \rangle := \sum_{a,b} abp(a, b \mid x, y).$$

La disuguaglianza CHSH pone un limite a una particolare combinazione di queste correlazioni, chiamata parametro S, definito come

$$S = \langle a_0 b_0 \rangle + \langle a_0 b_1 \rangle + \langle a_1 b_0 \rangle - \langle a_1 b_1 \rangle .$$

Vale infatti per tutte le teorie a variabili nascoste locali: $|S| \leq 2$

Con lo sviluppo tecnologico nei decenni seguenti fu possibile testare sperimentalmente l'esistenza di sistemi fisici (gli stati entangled) con i quali fosse possibile violare le disuguaglianze di Bell, dimostrando che la natura non poteva essere descritta da teorie a variabili nascoste locali. I primi esperimenti suggerivano già tale violazione, ciò nonostante le disuguaglianze di Bell valgono solo sotto alcune assunzioni molto rigide, e furono notate delle falle (i cosiddetti *loopholes*) nell'implementazione pratica dei test di Bell [12]. Queste falle forzano l'adozione di ipotesi addizionali a quelle di località e realismo per garantire la validità del test di Bell. Senza tali ipotesi addizionali i risultati sperimentali potrebbero infatti essere spiegati attraverso un modello a variabili nascoste locali. Recentemente sono stati fatti grandi progressi nel colmare simultaneamente tutte le principali lacune negli esperimenti [13, 14, 15, 16], confermando l'incompatibilità tra dati sperimentali e teorie a variabili nascoste locali. Oltre all'interesse puramente scientifico dei test di Bell, questi sono ugualmente alla base di protocolli di comunicazione quantistica ultra sicuri: infatti una violazione delle disuguaglianze di Bell implica, indipendentemente dai dispositivi utilizzati per la misura, l'esistenza di entanglement tra i sottosistemi misurati, il che può essere sfruttato per la realizzazione di protocolli *device-independent* [17, 18, 19], in cui non è necessario caratterizzare i sistemi utilizzati (questi potrebbero in teoria anche essere sotto il controllo di un avversario) pur mantenendo la sicurezza del protocollo.

2.2 Test di Bell in *time-bin* senza falla della post-selezione

2.2.1 Generazione di fotoni entangled in *time-bin*

Tra le varie proprietà (o gradi di libertà) dei fotoni che possono essere "entangled" vi è il loro tempo di generazione. Consideriamo un fotone con lunghezza d'onda di 775nm che sia in sovrapposizione di posizione tra due posizioni x_0 e x_1, con $x_1 - x_0 = 3$ m, scriviamo quindi il suo stato come

$$\frac{1}{\sqrt{2}}(|x_0\rangle + e^{i\phi}|x_1\rangle)\,. \tag{2.14}$$

Supponiamo inoltre che il fotone si propaghi in spazio libero verso un cristallo non-lineare, in cui possa avvenire il fenomeno di conversione parametrica spontanea (*spontaneous parametric downconversion*, abbreviato come SPDC, in inglese). Questo fenomeno trasforma un fotone a una data frequenza in due fotoni di frequenza (e quindi energia) dimezzata (o più in generale la cui somma dà la frequenza originale), tramite l'interazione tra il fotone di partenza e gli atomi costituenti il cristallo, che presenta un'elevata suscettività non-lineare del secondo ordine. Dato che il fotone di partenza è in sovrapposizione spaziale, vi sono due possibili momenti (o *time-bin*) in cui la coppia di fotoni a 1550 nm può venire generata tramite SPDC: t_0 e t_1, tali che $t_1 - t_0 = \frac{x_1 - x_0}{c} \approx 10$ ns (dove c è la velocità della luce). Tale processo non fa collassare la sovrapposizione e mantiene la relazione di fase tra le due componenti, si ottiene quindi lo stato entangled:

$$|\phi_{AB}\rangle = \frac{1}{\sqrt{2}}(|0_A\rangle \otimes |0_B\rangle + e^{i\phi}|1_A\rangle \otimes |1_B\rangle)\,, \tag{2.15}$$

in cui i qubit $|0\rangle$ e $|1\rangle$ sono associati ai due tempi di generazione t_0 e t_1 mentre A e B indicano i due fotoni (che possono differire negli altri gradi di libertà, quali polarizzazione o modo spaziale). Il processo di SPDC è estremamente improbabile, per produrre coppie di fotoni entangled si sfrutta dunque generalmente un impulso di pompa contenente milioni di fotoni e per produrre la sovrapposizione spaziale (o equivalentemente temporale, nel senso di un fotone che può arrivare prima o dopo al cristallo) viene generalmente diviso un impulso iniziale in due copie tramite uno specchio semi-riflettente (chiamato *beam-splitter*) e si fa percorrere ad una copia un percorso più lungo dell'altro prima di ricombinarli (usando un secondo *beam-splitter*), in modo che si crei un ritardo tra i due. Questo dispositivo viene chiamato interferometro sbilanciato di Mach-Zehnder.

2.2.2 Test di Bell con fotoni entangled in time-bin

Supponiamo che la fase dello stato in Eq. 2.15 sia regolata a $\phi = 0$, ottenendo di fatto lo stato $|\Phi^+\rangle$ introdotto in Eq. 2.11. Per ottenere una violazione massima delle disuguaglianze di Bell

con questo stato, le due misure da effettuare sono quelle degli osservabili $\widehat{O}_{a|x}$ e $\widehat{O}_{b|y}$ definiti da

$$\widehat{O}_{a|0} := \widehat{\sigma}_2, \quad \widehat{O}_{a|1} := \widehat{\sigma}_1, \quad \widehat{O}_{b|0} := \frac{1}{\sqrt{2}}(\widehat{\sigma}_2 + \widehat{\sigma}_1), \quad \widehat{O}_{b|1} := \frac{1}{\sqrt{2}}(\widehat{\sigma}_2 - \widehat{\sigma}_1)$$

in cui vengono usate due delle tre matrici di Pauli

$$\widehat{\sigma}_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \widehat{\sigma}_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \widehat{\sigma}_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.16}$$

Infatti secondo la teoria quantistica le correlazioni tra le misure sono date da

$$\langle a_x b_y \rangle = \left\langle \Phi^+ \middle| \widehat{O}_{a|x} \otimes \widehat{O}_{b|y} \middle| \Phi^+ \right\rangle$$

che risulta in

$$\langle a_0 b_0 \rangle = \langle a_0 b_1 \rangle = \langle a_1 b_0 \rangle = \frac{1}{\sqrt{2}}, \quad \langle a_1 b_1 \rangle = -\frac{1}{\sqrt{2}}$$

e infine nel parametro di Bell $S = \langle a_0 b_0 \rangle + \langle a_0 b_1 \rangle + \langle a_1 b_0 \rangle - \langle a_1 b_1 \rangle = 2\sqrt{2} \approx 2.81$[†].

Questi osservabili corrispondono a delle proiezioni nelle seguenti basi

$$\begin{aligned} \mathrm{A}_0 &= \left\{ \frac{1}{\sqrt{2}}(|0_A\rangle + i|1_A\rangle), \frac{1}{\sqrt{2}}(|0_A\rangle - i|1_A\rangle) \right\}, \\ \mathrm{A}_1 &= \left\{ \frac{1}{\sqrt{2}}(|0_A\rangle + |1_A\rangle), \frac{1}{\sqrt{2}}(|0_A\rangle - |1_A\rangle) \right\}, \\ \mathrm{B}_0 &= \left\{ \frac{1}{\sqrt{2}}\left(|0_B\rangle + e^{i\frac{\pi}{4}}|1_B\rangle\right), \frac{1}{\sqrt{2}}\left(|0_A\rangle + e^{i\frac{5\pi}{4}}|1_A\rangle\right) \right\}, \\ \mathrm{B}_1 &= \left\{ \frac{1}{\sqrt{2}}\left(|0_B\rangle + e^{i\frac{7\pi}{4}}|1_B\rangle\right), \frac{1}{\sqrt{2}}\left(|0_A\rangle + e^{i\frac{3\pi}{4}}|1_A\rangle\right) \right\} \end{aligned} \tag{2.17}$$

Per implementare tali misure è necessario ricombinare in un interferometro sbilanciato i pacchetti d'onda nei due *time-bin* (t_0 e t_1) introducendo a seconda della misura desiderata una differenza di fase particolare tra le due componenti $|0\rangle$ e $|1\rangle$ (che chiameremo φ_A per Alice e φ_B per Bob). Due possibili schemi per effettuare tale misura sono riportati in Fig. 2.4. In entrambi viene inizialmente generato uno stato entangled in *time-bin* producendo innanzitutto una sovrapposizione di due impulsi con un ritardo in un interferometro sbilanciato, che viene poi usato per pompare un cristallo non-lineare e produrre una coppia di fotoni entangled tramite SPDC. I due fotoni sono in seguito divisi e inviati uno ad Alice e l'altro a Bob. Sulla sinistra è presentato lo schema che fù utilizzato per primo per la misura di stati entangled in *time-bin*. Questo presenta un interferometro con un sbilanciamento identico a quello dell'interferometro di pompa, e un controllore di fase in uno dei rami per implementare le varie misure. Con questo schema però soltanto metà delle volte si avrà una proiezione nelle basi desiderate, in quanto

[†]Il valore di $2\sqrt{2}$ è il più alto ottenibile secondo la fisica quantistica, e viene chiamato limite di Tsirelson [20].

nulla garantisce che i pacchetti d'onda prendano i percorsi idonei a compensare il ritardo tra i due. Infatti può succedere che un fotone generato dal primo impulso passi per il cammino più corto o che quello generato dal secondo impulso percorra il cammino più lungo. In questi casi le misure non presenteranno le correlazioni necessarie a violare la disuguaglianza di Bell. Storicamente, questi casi furono inizialmente ingenuamente scartati, calcolando il parametro S solo sui casi favorevoli. Venne poi dimostrato come questa post-selezione introducesse una falla nelle disuguaglianze di Bell, che venne dunque chiamata *post-selection loophole* [21]. Risultò quindi necessario sviluppare un sistema di misura che non rendesse necessario scartare parte dei dati. Questo fu realizzato per la prima volta nel 2018 dal gruppo Quantum Future, ottenendo la prima violazione genuina delle disuguaglianze di Bell in *time-bin* [22]. A tal fine fu utilizzato un secondo interferometro di Mach-Zehnder, questa volta bilanciato, che permette di sfruttare l'interferenza tra i due cammini per indirizzare i pacchetti d'onda nel ramo corretto dell'interferometro sbilanciato di misura. Questo schema, chiamato "*time-bin* attivo" è presentato sulla parte destra della Figura 2.4.

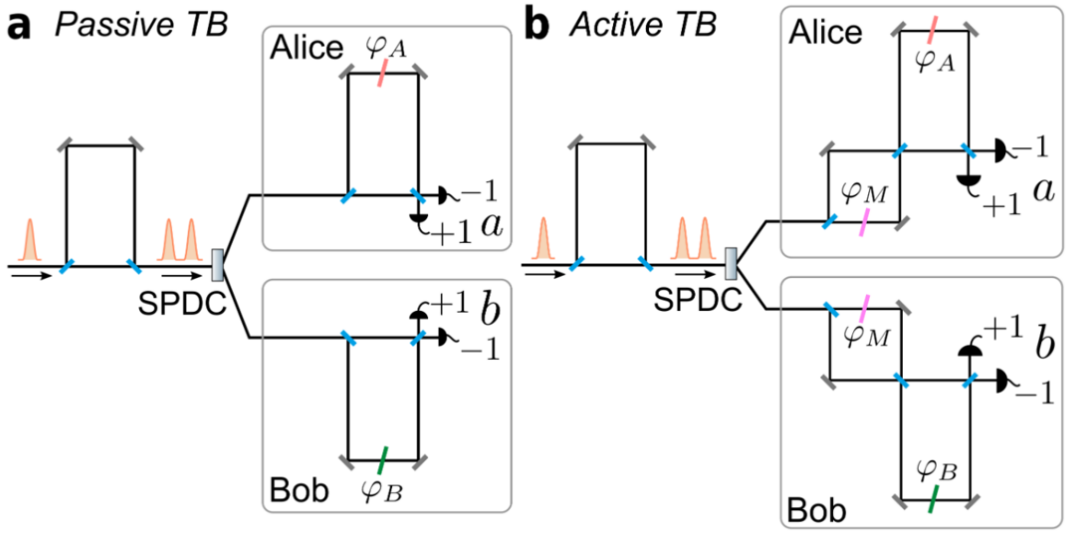


Figura 2.4: Schema da Ref. [22] che espone la differenza tra sistemi passivi e attivi di misura di stati entangled in *time-bin*.

2.3 Requisiti di sincronia per la distribuzione di *time-bin* entanglement su lunghe distanze

Le attività di questa tesi si collocano nella preparazione di un nuovo esperimento da parte del gruppo Quantum Future nel quale una versione aggiornata dello schema di distribuzione di entanglement in *time-bin* attivo verrà implementato su una lunga distanza. Effettuare l'esperimento su lunga distanza ha una duplice utilità. Innanzitutto, la lunga distanza può essere sfruttata per limitare la possibilità di comunicazione tra i due apparati di misura, infatti la relatività

impone che nessuna informazione possa essere comunicata ad una velocità maggiore di quella della luce. Questo implica che se due ricevitori sono piazzati ad una distanza d qualsiasi comunicazione tra i due non può avvenire in un tempo minore di $\frac{d}{c}$. Variando la misura effettuata casualmente e con un rate superiore a $\frac{c}{d}$ si è quindi sicuri che al momento della misura nessuna informazione sulla misura implementata possa aver raggiunto l'altra stazione e influenzato l'esito della misura. In questo modo si garantiscono le condizioni necessarie a un test di Bell senza falle di località (*locality loophole*). Il secondo vantaggio è di tipo tecnologico e consiste nella dimostrazione che la distribuzione dell'entanglement possa essere effettuata su lunghe distanze. L'entanglement costituisce infatti una risorsa per l'implementazione di numerosi protocolli, dalla distribuzione quantistica di chiavi al teletrasporto quantistico.

La distribuzione di entanglement su lunghe distanze presenta tuttavia numerose sfide: la trasmissione può degradare la qualità dell'entanglement, introduce delle perdite e pone il problema della sincronia. Come vedremo è infatti necessario implementare un sistema di sincronia capace di allineare i due ricevitori temporalmente e in frequenza al laser di pompa. Questo verrà fatto sfruttando la stessa rete in fibra ottica utilizzata per la trasmissione dei fotoni entangled. Verrà ora descritto brevemente l'esperimento prospettato e i vari requisiti per quanto riguarda il sistema di sincronizzazione.

2.3.1 Setup elettro-ottico

Il setup elettro-ottico che sarà utilizzato nell'esperimento è illustrato in Fig. 2.5. Il laser di pompa è un laser titanio-zaffiro impulsato tramite *mode-locking* che emette impulsi luminosi a 775 nm di alcuni picosecondi a un rate di 76 MHz. Questi impulsi vengono iniettati in fibra ottica attraversano un interferometro di Mach-Zehnder sbilanciato di 5 ns per preparare la sovrapposizione temporale. La generazione dei fotoni entangled a 1550 nm avviene tramite tramite SPDC in un cristallo di potassio-titanil-fosfato (abbreviato in KTP), che produce due fotoni con polarizzazioni ortogonali, e viene quindi classificato come *tipo II*. I fotoni possono quindi essere divisi in funzione della loro polarizzazione tramite l'utilizzo di un *polarization beam-splitter* e sono indirizzati a due ricevitori identici (Alice e Bob), uno dei quali sarà piazzato a lunga distanza (10 km massimo) dalla sorgente. I ricevitori sono costituiti da uno *switch* ottico in grado di indirizzare i pachetti d'onda nel cammino corretto di un interferometro di Mach-Zhender sbilanciato al fine di compensare il loro ritardo e ottenere le proiezioni nelle basi desiderate. I ricevitori sono realizzati in fibra tranne per un linea di ritardo implementata in spazio libero che ha una duplice funzione: permette innanzitutto di rendere il ritardo nel ricevitore uguale a quello dell'interferometro di pompa con un precisione di alcune decine di micrometri (il che aumenta la qualità dell'interferenza), e permette inoltre di controllare la differenza di fase al fine di implementare le misure desiderate. I fotoni sono in seguito rilevati da dei rilevatori a singolo fotone di tipo *superconducting nanowire* (SNSPD). In questi rilevatori, mantenuti tramite cryogenia a una temperatura di 0.8 K, è sufficiente l'arrivo di un singolo fotone perché la sua

energia riscaldi un nano-cavo e interrompa il suo stato di superconduttività, il che risulta in una variazione di corrente che indica l'avvenuto rilevamento.

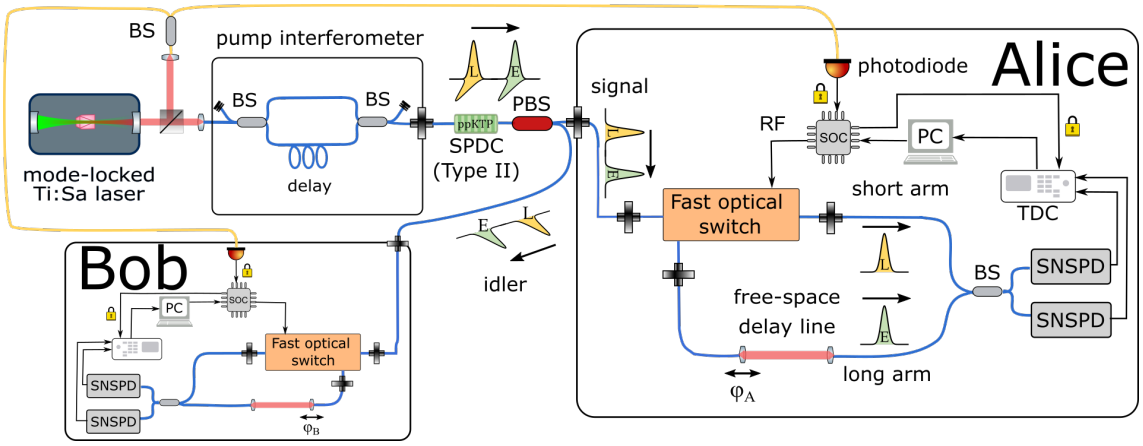


Figura 2.5: Configurazione elettro-ottica utilizzata per la generazione di stati time-bin entangled e loro misurazione attraverso un test di Bell senza "falle di post-selezione" (post-selection loophole). Le fibre che mantengono la polarizzazione sono disegnate in blu mentre le fibre monomodali sono disegnate in giallo. E: pacchetto d'onda iniziale, L: pacchetto d'onda tardiva, BS: Beam Splitter, PBS: BS polarizzante, RF: Radio Frequency electrical signal (segnale elettrico a radiofrequenza), SOC: System On Chip ("sistema su chip"), SNSPD: Superconducting Nanowire Single-Photon Detector (rilevatore di fotone singolo a nanofilo superconduttore), SPDC: Spontaneous Parametric Down-Conversion crystal (cristallo di conversione parametrica spontanea, TDC: Time-to-Digital Converter (convertitore "tempo-digitale").

2.3.2 Distribuzione della frequenza

Il segnale elettrico di rilevamento viene trasmesso ad un *time-to-digital converter* che registra con precisione di alcuni picosecondi il tempo di rilevamento del fotone, producendo i cosiddetti *time-tags*. Questo permetterà in seguito di associare rilevamenti corrispondenti al medesimo inpulso di pompa, e ottenere informazioni sul cammino percorso dai fotoni, in modo da valutare le prestazioni dello *switch* ottico, il cui design innovativo deve ancora fare oggetto di pubblicazione. A tal fine è indispensabile che il *clock* del time-tagger sia sincronizzato alla frequenza con la quale il laser emette gli impulsi, che è di 76 MHz, tuttavia presenta leggere fluttuazioni dovute a vibrazioni e cambi di temperatura che modificano la lunghezza della cavità del laser. Si rende quindi necessario effettuare il *locking* del *time-tagger* agli impulsi del laser, che vengono in parte ridiretti a una fibra ausiliaria collegata a un fotodiodo all'interno dei ricevitori. Il *time-tagger* utilizzato (quTools quTAG) però accetta come *clock* soltanto segnali a 10 MHz, ragione per cui il segnale del fotodiodo a 76 MHz viene prima trasformato in un segnale a 10 MHz con l'ausilio di una scheda elettronica Field-Programmable Gate Array (FPGA) Xilinx Zynq®-7000 integrata in una *evaluation board* (Avnet ZedBoard). Questa scheda non è soltanto responsabile della produzione di questo segnale di clock ma anche di un segnale usato per controllare lo *switch* ottico, che necessita anch'esso di essere sincronizzato con gli impulsi di pompa al fine di modulare correttamente i vari pacchetti d'onda costituenti i fotoni entangled.

Nel Capitolo 4 verrà presentato lo sviluppo di un'interfaccia per un analizzatore di spettro RF che sarà utilizzato per controllare la qualità del segnale ricevuto dal fotodiodo dopo la trasmissione su una bobina di 10 km di fibra ITU-T G.652. Questa fibra è infatti pensata per la trasmissione di luce con lunghezza d'onda tra i 1310 nm e 1625 nm ma per semplicità si prospetta di utilizzarla per la trasmissione degli impulsi del laser di pompa a 775 nm. Si temono quindi possibili alterazioni del segnale, che potrebbero impedire l'utilizzo del segnale del fotodiodo per il *locking* della scheda FPGA.

2.3.3 Stima del ritardo temporale

Un altro problema riscontrato nella distribuzione a lunga distanza di fotoni entangled è quello, partendo dalle due sequenze di tempi di rilevamento prodotte dai due TDC, di determinare quali rilevamenti corrispondono a fotoni appartenenti alla stessa coppia entangled. In altri termini va determinato l'*offset* temporale tra le due sequenze di *time-tags*, dovuto ai differenti tempi di trasmissione ma anche dall'assenza di un riferimento temporale assoluto, in quanto i TDC utilizzano il loro momento di accensione come origine dell'asse dei tempi, e avendo due TDC in postazioni lontane il ritardo nell'accensione dei TDC potrebbe contarsi addirittura in secondi. Risulta possibile sfruttare la rete internet per inviare un commando di acquisizione sincronizzata dei *time-tags* ma anche in questo caso vi sarà un ritardo di un centinaio di millisecondi tra le due catture. Fortunatamente ci viene in aiuto la randomicità del processo di SPDC. Infatti, soltanto circa un impulso su cento produrrà una coppia di fotoni entangled. Sarà quindi possibile correlare le sequenze di tempi di rilevamento ai due ricevitori per determinare il ritardo tra i due. L'obbiettivo del Capitolo 5 sarà infatti lo sviluppo di uno script capace di trovare tale ritardo (con un a precisione inferiore a quella del periodo del laser, ovvero circa 13.15 ns) a partire dalle due sequenze di *time-tags*. Questa operazione è complicata dal fatto che la maggior parte dei fotoni viene persa lungo il cammino e solo un fotone su cento circa viene rilevato. I rilevamenti corrispondenti a coppie entangled rischiano quindi di confondersi con un fondo di coincidenze accidentali ed è necessario effettuare la correlazione con un alta precisione temporale. Questo comporta però un innalzamento del carico computazionale necessario al calcolo della correlazione e sarà quindi necessario l'utilizzo di algoritmi efficienti e una scelta ponderata dei vari parametri.

3

Trasformate di Fourier: breve presentazione dei concetti fondamentali di interesse

Le Trasformate di Fourier rappresentano uno degli strumenti più versatili nell'analisi dei segnali e dei dati. Esse trovano applicazione in diversi ambiti, dall'elaborazione delle immagini all'automatica, dall'analisi in frequenza di sistemi fisici dinamici alla teoria dei segnali. Questo capitolo ne esplorerà unicamente i concetti chiave, fornendo i fondamenti teorici basilari per garantire una maggiore familiarità con gli argomenti che verranno presentati nelle sezioni seguenti. In particolare, si presterà maggiore attenzione sulla DFT (Discrete Fourier Transform) e sull'algoritmo FFT (Fast Fourier Transform) [23] [24], direttamente impiegati per la realizzazione del sistema di sincronia di interesse nel presente progetto di tesi.

3.0.1 Trasformata di Fourier: definizioni chiave nel caso continuo

Concettualmente, la trasformata di Fourier è una trasformazione matematica che converte un segnale o una funzione nel dominio del tempo in un insieme di componenti sinusoidali (armoniche) nel dominio della frequenza. Ciò consente di scomporre un segnale generico e successivamente utilizzare l'anti-trasformata di Fourier (la formula inversa) per ricostruirlo. Gli insiemi di valori in funzione della frequenza (continui o discreti) sono chiamati spettri di ampiezza e di fase.
Si consideri una funzione (o un qualsiasi segnale continuo nel tempo) *f(t)*, tale per cui l'integrale (improprio)

$$\int_{-\infty}^{\infty} |f(t)|dt = \int_{\mathbb{R}} |f(t)|dt \tag{3.1}$$

sia convergente (si dice anche f(t) $\in L^1(\mathbb{R})$). Dal punto di vista matematico, per tale f segue la definizione 3.0.1.

Definizione 3.0.1. *Si definisce Trasformata di Fourier (FT) della funzione continua f(t) il risultato $F(\omega)$ determinato dall'integrazione*

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-i\omega t} dt \tag{3.2}$$

La Trasformata di Fourier Inversa (IFT) permette invece di ritornare al dominio del tempo da quello delle frequenze.

Definizione 3.0.2. *Sia $F(\omega) : \mathbb{R} \to \mathbb{C}$ una funzione trasformabile secondo Fourier. La funzione f(t) è anti-trasformata di Fourier di $F(\omega)$ se vale*

$$f(t) = \int_{-\infty}^{\infty} F(\omega) \cdot e^{i\omega t} d\omega \tag{3.3}$$

Nel contesto del presente progetto di tesi scenderemo maggiormente nel dettaglio per quanto riguarda la trasformata di Fourier discreta, alla base dell'algoritmo FFT adoperato nella sezione 5.3.1.

3.0.2 Trasformata di Fourier Discreta (DFT)

Di fondamentale importanza per gli argomenti che verranno esposti nei capitoli successivi è la "versione" discreta della Trasformata di Fourier presentata precedentemente. Analogamente a quanto esposto per la FT nel caso continuo, anche la DFT è una trasformata matematica che converte un segnale, questa volta discreto, dal dominio del tempo al dominio delle frequenze, consentendo di analizzare le componenti di frequenza presenti nel segnale stesso.
Come verrà presentato alla sezione 3.1, nella pratica, la DFT viene calcolata utilizzando algoritmi efficienti come l'Algoritmo della Trasformata Veloce di Fourier (FFT), che riduce il numero di operazioni richieste, passando da una complessità computazionale di $O(n^2)$ a $O(nlog(n))$ (con n numero di elementi della successione discreta su cui l'algoritmo è applicato), rendendo fattibile il calcolo della DFT anche per sequenze di grandi dimensioni.

Matematicamente, la Trasformata discreta di Fourier è una trasformazione che può essere riscritta in termini di moltiplicazione di un vettore di dati per una matrice di coefficienti complessi.
Si consideri una funzione f definita su N punti discreti (figura 3.1), denominati $x_0, x_1 \ldots x_{N-1}$, tali da poter essere inseriti in un vettore denominato $x[j]$.
E' possibile immaginare graficamente f come una curva campionata in corrispondenza della successione dei punti delle ascisse introdotti.

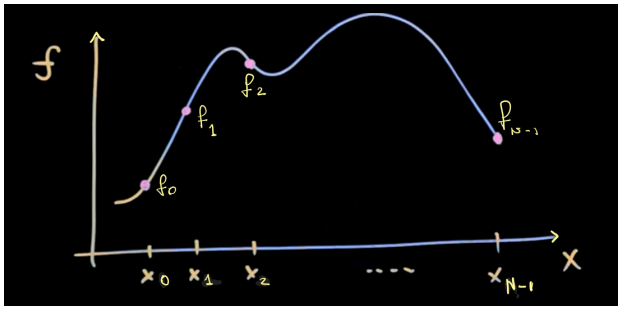


Figura 3.1: Esempio di funzione f con punti f_j di campionamento e x_j su asse delle ascisse

I valori che assume la funzione per ciascun componente di $x[j]$ sono identificati come f_0, f_1 ... f_{N-1}, i quali formano il vettore di dati $f[j]$.
Da quest'ultimo è possibile estrarre un vettore $\hat{f}[k]$ di N coefficienti complessi applicando la Trasformata di Fourier Discreta così definita:

Definizione 3.0.3 (Trasformata di Fourier Discreta (DFT))**.**

$$\hat{f}_k = \sum_{j=0}^{N-1} f_j \cdot e^{-i2\pi j \frac{k}{N}} \tag{3.4}$$

Ciascuno dei coefficienti ricavati dall'espressione 3.4 definisce una componente di frequenza che costituisce il vettore $f[j]$. Ad esempio, $\hat{f}_0$ comunica quanto di quella bassa frequenza ad esso associata sia presente nei dati di partenza. Allo stesso modo, $\hat{f}_1$ sarà indice del quantitativo della frequenza immediatamente superiore nel vettore $f[j]$. Discorso analogo vale per gli $\hat{f}_k$ successivi, fino a $\hat{f}_{N-1}$, rappresentante la massima frequenza possibile di oscillazione tra gli N elementi considerati.
In altri termini:

$$f_0, f_1, \dots f_{N-1} \xrightarrow{DFT} \hat{f}_0, \hat{f}_1, \dots \hat{f}_{N-1} \tag{3.5}$$

A partire dai coefficienti di Fourier posso ricavare nuovamente il dataset discreto di partenza riapplicando (come nel caso della FT) l'anti-trasformata della DFT:

Definizione 3.0.4 (Anti-trasformata di Fourier discreta (IDFT))**.**

$$f_k = \frac{1}{N} \sum_{j=0}^{N-1} \hat{f}_j \cdot e^{i2\pi j \frac{k}{N}} \tag{3.6}$$

Si noti ora come nella formula 3.4 l'esponente dell'espressione $e^{\frac{-i2\pi}{N}}$ venga moltiplicato per gli indici interi j e k. Tale espressione definisce una frequenza fondamentale ed è possibile ridenominarla come ω_N, anche detta radice dell'unità primitiva N-esima (la cui N-esima potenza è pari a 1).
Questa frequenza fondamentale è importante per determinare una matrice che rappresenta la trasformazione in oggetto, che chiamo *matrice di Fourier discreta*, indicata come Ω_N. Ciò significa che è possibile derivare una matrice, espressa in funzione di ω_N, che, se moltiplicata per

il vettore $f[j]$, mi permette di ottenere come risultato il vettore complesso $\hat{f}[k]$.
Per determinare i termini di Ω_N consideriamo nuovamente la formula 3.4 e, più in particolare, l'espressione che moltiplica le componenti di $f[j]$, ovvero $e^{\frac{-i2\pi jk}{N}}$. Per $k = 0$ otteniamo la prima riga, per $k = 1$ la seconda e così via per i valori dell'indice intero successivi. Con $k = 0$, indipendentemente dai valori di j per i quali calcoliamo la sommatoria, l'unità $e^{\frac{2i\pi k}{N}}$ varrà sempre 1. Dunque la prima riga di Ω_N conterrà solo 1. Per $k = 1$ la seconda riga della *Matrice di Fourier discreta* apparirà diversa. Per $j = 0$ si ottiene nuovamente 1, per $j = 1$ si ottiene $e^{\frac{2i\pi jk}{N}} = e^{\frac{2i\pi}{N}} = \omega_N$. Allo stesso modo è immediato osservare che per $j = 2$ si otterrà ${\omega_N}^2$, fino ad ottenere ${\omega_N}^{N-1}$ per $j = N - 1$. Proseguendo per i successivi valori di k, si ottiene

$$\Omega_N = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & {\omega_N}^2 & \dots & {\omega_N}^{N-1} \\ 1 & {\omega_N}^2 & {\omega_N}^4 & \dots & {\omega_N}^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & {\omega_N}^{N-1} & {\omega_N}^{2(N-1)} & \dots & {\omega_N}^{(N-1)^2} \end{bmatrix} \tag{3.7}$$

Vale dunque l'equazione 3.8

$$\begin{bmatrix} \hat{f}_0 \\ \hat{f}_1 \\ \hat{f}_2 \\ \vdots \\ \hat{f}_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & {\omega_N}^2 & \dots & {\omega_N}^{N-1} \\ 1 & {\omega_N}^2 & {\omega_N}^4 & \dots & {\omega_N}^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & {\omega_N}^{N-1} & {\omega_N}^{2(N-1)} & \dots & {\omega_N}^{(N-1)^2} \end{bmatrix} \times \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{N-1} \end{bmatrix} \tag{3.8}$$

La *matrice di Foruier discreta* fornisce una rappresentazione compatta e strutturata delle frequenze presenti nel segnale e ha dimensioni $N \times N$. Tuttavia non è così rapido e semplice calcolarla completamente per poter applicare la trasformata di Fourier ad una sequenza di dati. Ω_N costituisce infatti solo la base degli algoritmi FFT, che consentono di calcolare la trasformata di Fourier in modo ancora più veloce rispetto all'approccio matriciale diretto, come sarà presentato nella sezione 3.1.

3.0.3 Teorema della convoluzione

Il Teorema della Convoluzione è un risultato importante nell'ambito delle trasformate di Fourier, che stabilisce una relazione tra la convoluzione nel dominio del tempo e la moltiplicazione nel dominio della frequenza.
Supponiamo di avere due segnali nel dominio del tempo, x(t) e h(t), le cui trasformate di Fourier sono rispettivamente X(f) e H(f). La convoluzione dei due segnali nel dominio del tempo

è definita come:

$$(x * h)(t) = \int_{-\infty}^{\infty} x(\tau) \cdot h(t-\tau)\, d\tau \tag{3.9}$$

Teorema 3.0.1 (Teorema della convoluzione, caso continuo)**.** *Il teorema della convoluzione afferma che la trasformata di Fourier della convoluzione di due segnali nel dominio del tempo è uguale al prodotto delle trasformate di Fourier di questi segnali nel dominio della frequenza. Matematicamente si ha che*

$$\mathscr{F}\{x * h\}(f) = X(f) \cdot H(f) \tag{3.10}$$

In modo analogo tale proprietà è valida per la convoluzione discreta di due successioni **x** e **y**, definita come:

$$x * y = \sum_{k=-\infty}^{\infty} x[k] \cdot y[n-k] \tag{3.11}$$

Si ha infatti che

$$DFT[x * y] = DFT[x] \times DFT[y] \tag{3.12}$$

Un'importante applicazione di questo teorema è presentata alla sezione 5.3.1.

3.1 Fast Fourier transform (FFT)

L'algoritmo FFT è uno strumento di fondamentale importanza per il calcolo della Trasformata di Fourier Discreta su un set di dati. Come derivato dalla sezione 3.0.2, ad un vettore di coefficienti è possibile applicare la Trasformata discreta di Fourier attraverso un'operazione di moltiplicazione per la *matrice di Fourier discreta* Ω_N. Tuttavia, per scomporre il vettore in componenti di frequenza, questo approccio può risultare poco rapido ed efficiente dal punto di vista computazionale. Il calcolo della Trasformata di Foruier discreta attraverso Ω_N ha infatti complessità dell'ordine $O(N^2)$, dal momento che il vettore di partenza di N coefficienti $f[j]$ dovrà essere moltiplicato per tutte le N righe della matrice per ottenere le componenti di $\hat{f}[k]$. L'algoritmo FFT consente di ridurre la complessità computazionale dell'operazione dall'ordine $O(N)$ all'ordine $O(Nlog(N))$ (quasi lineare in N).
Di seguito è presentato, brevemente e concettualmente, l'algoritmo FFT di Cooley-Tukey, basato sul paradigma *divide et impera* che, in questo caso, sfrutta ricorsivamente la proprietà di simmetria e la struttura a "farfalla" delle matrici di Fourier discrete per ridurre notevolmente il numero di operazioni necessarie per calcolare la DFT.

Supponiamo che N sia nel nostro caso una potenza di 2, ad esempio poniamo $N = 2^{10} = 1024$. Con l'approccio classico avremmo

$$\hat{f}[k] = \Omega_{1024} \times f[j]. \tag{3.13}$$

Risulta possibile scomporre l'equazione 3.13 in modo differente, riorganizzando il vettore colonna $f[j]$ posizionando nelle sue prime posizioni tutti i propri componenti di indice pari, lasciando i restanti componenti di indice dispari nelle posizioni successive. Omettendo la dimostrazione matematica che porta al risultato riportato di seguito, si ottiene una nuova equazione:

$$\hat{f}[k] = \begin{bmatrix} I_{512} & -D_{512} \\ I_{512} & -D_{512} \end{bmatrix} \times \begin{bmatrix} \Omega_{512} & 0 \\ 0 & \Omega_{512} \end{bmatrix} \times \begin{bmatrix} f_{odd} \\ f_{even} \end{bmatrix} \tag{3.14}$$

Dove la matrice D_{512} è una matrice diagonale del tipo

$$D_{512} = \begin{bmatrix} \Omega^0 & 0 & 0 & \dots & 0 \\ 0 & \Omega^1 & 0 & \dots & 0 \\ 0 & 0 & \Omega^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \Omega^{511} \end{bmatrix}$$

Nell'equazione 3.14 è importante sottolineare che la scomposizione di Ω_{1024} nelle due matrici presentate velocizza notevolmente i calcoli. La prima matrice è infatti composta da quattro blocchi diagonali, che consentono di rendere maggiormente efficienti le operazioni di moltiplicazione. La seconda risulta nettamente meno "densa" rispetto Ω_{1024}, considerando due blocchi da 512 elementi e una forte presenza di zeri, dimezzando il costo computazionale che si otteneva con moltiplicazioni per la *matrice di Fourier discreta* stessa.
Risulta ora possibile iterare nuovamente la scomposizione 3.14 per le matrici Ω_{512} ripetendo la stessa procedura di organizzazione delle componenti di f_{odd} e f_{even}, riportando, per ciascuna sottosequenza, dapprima i termini di indice pari, dopodiché quelli di indice dispari. L'algoritmo si ripete ricorsivamente fino al raggiungimento di Ω_2. Sfruttando la simmetria di Ω_N e la ridondanza dei dati, è possibile, attraverso queste scomposizioni, dimezzare passo dopo passo la complessità computazionale delle operazioni da eseguire.
Anche nel caso in cui non si disponga di un set di dati di partenza con N potenza di 2, sarebbe comunque conveniente aggiungere tanti zeri (operazione di *padding*) quanti sono necessari per arrivare ad avere un numero di termini discreti pari alla più vicina potenza di 2 raggiungibile e applicare l'algoritmo FFT. Esistono ad ogni modo versioni dell'FFT valide per qualsiasi valore di N, anche per N primo.

4

Analizzatore di Spettro e Interfaccia Python per il modello *BB60C*

Gli analizzatori di spettro sono dispositivi elettronici critici nell'elaborazione dei segnali e nelle comunicazioni. Essi consentono di visualizzare e analizzare le componenti di frequenza di un segnale, fornendo una panoramica della sua distribuzione spettrale. In particolare un analizzatore di spettro digitale combina un ricevitore supereterodina, un convertitore analogico-digitale e la Fast Fourier Transform (FFT) per calcolare lo spettrogramma di un segnale in ingresso. La tecnica di conversione supereterodina, comunemente utilizzata nei ricevitori radio, permette di traslare in frequenza un segnale prima di analizzarlo. Questo permette di analizzare in sequenza diverse porzioni dello spettro utilizzando un filtro passa-banda fisso per filtrare una sezione variabile dello spettro originale. Con questa scansione è quindi possibile analizzare larghe porzioni di spettro andando ben oltre la larghezza di banda istantanea del dispositivo.

La visualizzazione dello spettro consente di identificare interferenze, rumore e altre caratteristiche che possono influire sulla qualità dei segnali trasmessi o ricevuti. Questi strumenti sono particolarmente utili nella ricerca e nello sviluppo di sistemi di comunicazione wireless, radar, dispositivi elettromagnetici e elettronici.

Figura 4.1: Analizzatore di spettro Signal Hound, modello BB60C

Nelle sezioni seguenti sarà innanzitutto presentato l'analizzatore di spettro *Signal Hound BB60C* (figura 4.1), utilizzato nel contesto sperimentale presentato alla sezione 2.3.1. Più in par-

ticolare verranno esposte le specifiche chiave del dispositivo, consentendo una visione panoramica dell'ampiezza delle funzionalità offerte dal modello. Sarà in seguito esaminato l'approccio software adottato per implementare un'interfaccia grafica di facile utilizzo per adoperare l'analizzatore di spettro di interesse, fornendo inoltre il codice python finalizzato alla progettazione di tale ambiente interattivo e intuitivo. Verrà infine delineato l'utilizzo pratico in laboratorio della piattaforma sviluppata.

4.1 Analizzatore di spettro *Signal Hound BB60C*

Come anticipato nelle precedenti righe, nel contesto dell'esperimento trattato nella presente tesi, è stato adoperato un analizzatore di spettro ad alta performance prodotto dalla *Signal Hound*, un'azienda specializzata nello sviluppo di strumenti di misura e analisi dei segnali RF (Radio Frequenza). Si tratta in particolare del modello BB60C.
Di seguito sono presentate le principali (e fondamentali) caratteristiche del dispositivo*:

- Range di frequenza da 9 kHz a 6 GHz: il dispositivo consente di rilevare e analizzare frequenze nell'intervallo riportato.
- Velocità di Scansione fino a 24 GHz/sec.
- Ampio Range Dinamico da -158 dBm a +10 dBm: si tratta dell' ampio intervallo di potenza di un segnale che l'analizzatore di spettro è in grado di misurare.
- Larghezza di Banda di Risoluzione disponibile da 10 Hz a 10 MHz. La larghezza di banda di risoluzione indica la separazione minima necessaria tra due componenti spettrali per essere discriminate e visualizzate separatamente. Può essere impostata a piacere entro l'intervallo specificato.
- Larghezza di Banda Istantanea di 27 MHz: il dispositivo può visualizzare e analizzare un'ampia banda di frequenze contemporaneamente.

4.2 Progettazione software per analizzatore di spettro *Signal Hound BB60C*

Questa sezione ha lo scopo di presentare il software che soddisfa l'esigenza di fornire un ambiente interattivo e intuitivo per l'analisi dello spettro dei segnali utilizzando l'analizzatore di spettro

*Per ulteriori informazioni dettagliate riguardo alle specifiche complete dell'analizzatore di spettro BB60C, si consiglia di fare riferimento al sito web ufficiale del produttore riportato al seguente link.

Signal Hound, serie BB60C. I file prodotti si inseriscono in un progetto preesistente realizzato da docenti e ricercatori del gruppo di ricerca Quantum Future. Tale progetto, denominato *Lab Research Software*, è una piattaforma software, realizzata con linguaggio di programmazione Python 3, contenente risorse per l'utilizzo di dispositivi di laboratorio (tra i quali misuratori di potenza, oscilloscopi e time-tagger), il cui utilizzo è semplificato attraverso interfacce utente appositamente programmate. Più in particolare, per ciascun dispositivo considerato, tali risorse si articolano nelle classi *Device*, *Manager* e *Widget*.
Le classi Device e Manager consentono di performare tutti calcoli in backend per la corretta elaborazione delle funzionalità offerte dal dispositivo in questione, ciascuna delle quali potrà poi essere resa visibile all'utente attraverso l'interfaccia grafica fornita dalle classi Widget.

Perfettamente in linea con la filosofia del progetto *Lab Research Software*, anche per l'analizzatore di spettro BB60C sono state realizzate classi analoghe, presentate discorsivamente nelle sottosezioni seguenti (il codice effettivamente prodotto è stato inserito per intero in Appendice A). A tale scopo, queste fanno riferimento alle Application Programming Interface (API) per interagire con l'analizzatore di spettro Signal Hound BB60C (reperibili cliccando sul seguente indirizzo: Signal Hound sdk), prelevando le risorse necessarie dal modulo *devices.bb_api*.

4.2.1 Classe Manager per *SignalHound BB60C*

Questa classe (si veda la sezione A.1.1) fornisce un'implementazione per la gestione dell'analizzatore di spettro Signal Hound BB60C. Offre la funzionalità principale di analisi in frequenza del segnale, modulando automaticamente e in maniera intelligente i parametri di acquisizione delle misure, trattati a breve.
Sono presenti funzioni di connessione e configurazione del dispositivo, al quale è così consentito l'accesso.
Il manager viene inizializzato con valori di default assegnati ai parametri per compiere l'analisi dello spettro del segnale in ingresso. Tali parametri sono:

- Frequenza centrale (sulla quale sarà centrata la scansione e visualizzazione delle frequenze componenti del segnale, poste lungo l'asse x), in Hz.

- Span (intervallo compreso tra frequenza iniziale e frequenza finale di acquisizione), in Hz.

- Livello di potenza di riferimento (sul quale sarà centrata la visualizzazione della potenza del segnale, posta lungo l'asse y) in dBm.

- Guadagno e/o attenuazione: parametri che consentono di amplificare e/o attenuare l'intensità del segnale in ingresso, modulando dunque la sensibilità della misurazione.

- RBW (Resolution BandWidth): rappresenta la modalità di analisi utilizzata per separare i segnali nelle diverse frequenze. Una RBW più "piccola" aumenta la risoluzione dell'analizzatore, richiedendo di conseguenza più tempo per acquisire i dati.
- VBW (Video BandWidth): la banda passante del filtro applicato ai dati dello spettro dopo l'acquisizione. Incrementando tale parametro viene ridotto il rumore e l'incertezza nelle misurazioni, riducendo a sua volta però la capacità di separare segnali vicini.
- Tempo, per acquisizione, per catturare tutti i dati dello spettro del segnale.
- RBW shape : determina la funzione di finestra utilizzata per l'analisi dello spettro. Di default è selezionata la funzione di finestra "flat top" o "Finestra massimamente piatta".
- Rejection: indica il livello di rifiuto di rumore.
- Detector: tipo di rivelatore utilizzato (nel codice viene impostato il detector di default, optando per una restituzione di valori minimi e massimi di ampiezza per ciascun bin di frequenza).
- Scale: indica la scala di visualizzazione dei dati dello spettro.
- Processing units: unità di elaborazione dei dati dello spettro. Di default è assegnato un valore che emula un tradizionale analizzatore di spettro a scala logaritmica.

Ciascuno di questi parametri può essere manipolato direttamente dall'utilizzatore del programma. Qualora vengano assegnati valori "errati" e/o non coerenti tra loro a parametri differenti, interverrà la funzione *set_parameter*, provvedendo a reimpostare in maniera consistente i valori stessi. Ciascun cambiamento viene comunicato alla classe Widget, trattata a breve, per un corretto aggiornamento dell'interfaccia utente.
L'acquisizione dello spettro del segnale è compiuta attraverso la funzione *get_sweep*, invocata attraverso un *LoopThread* ogni 300ms, ottenendo così una lettura dei dati del segnale pressoché real time. *LoopThread* è una classe sviluppata nella piattaforma *Lab research software*, finalizzata a semplificare l'utilizzo delle classi *QThread* e *Qtimer* del modulo Python *PyQt6.QtCore*.
Infine, la classe manager fornisce funzioni per il salvataggio su file (binario o di testo) dei dati delle analisi eseguite, rispettivamente *before_saving* e *saving_function*.

4.2.2 Classe Widget per *SignalHound BB60C*

La classe Widget (sezione A.1.2), come ampiamente anticipato, fornisce un'interfaccia grafica che rende visibile l'analisi dello spettro dei segnali di ingresso.
È implementata per essere interattiva, attraverso risposte immediate agli input da tastiera. L'interazione "real time" è garantita grazie alla classe *myQLabelQLineEdit*, presente nel progetto

Lab Research Software, utilizzata per creare i campi di input per i parametri dello spettro nel widget.
L' interfaccia è inserita in una *SuperWindow*, classe che, ancora una volta, è stata sviluppata e resa disponibile nella piattaforma software del laboratorio. Essa è fondamentale per una corretta visualizzazione dei risultati e fornisce un "handler" indispensabile alle classi Widget.

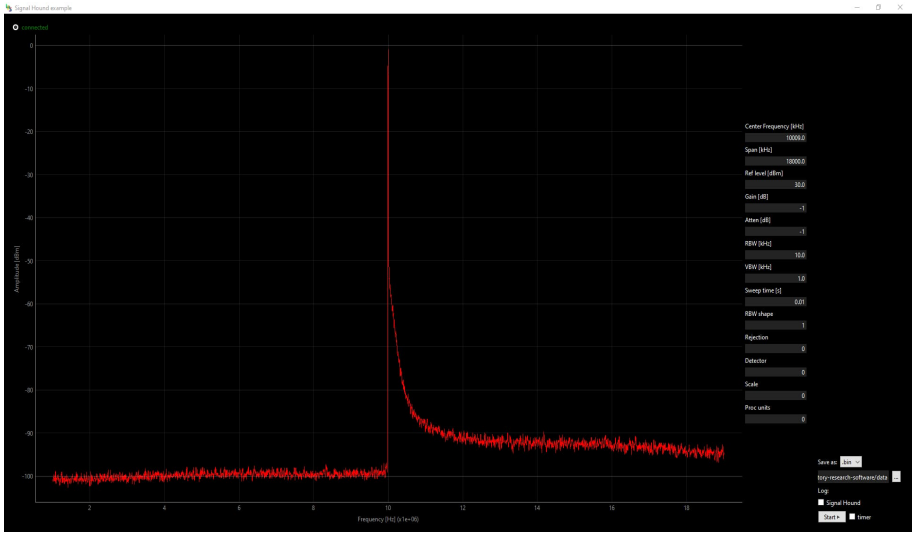


Figura 4.2: Interfaccia interattiva per l'analisi dello spettro dei segnali attraverso l'analizzatore di spettro Signal Hound BB60C, fornita attraverso la classe Widget (widget_signal_hound_bb60c.py, sezione 5.1.2)

La classe consente la connessione all'analizzatore di spettro attraverso il pulsante "Connect" (dalla classe *QRadioButton*).
Dopo aver stabilito la connessione con il dispositivo, il widget avvia la scansione dello spettro e visualizza i dati risultanti in un grafico di tipo *PlotWidget* di *pyqtgraph*. I dati dello spettro vengono visualizzati in scala logaritmica, accompagnati da una finestra in cui sono presenti tutti i parametri che regolano l'acquisizione corrente. Una modifica di tali parametri direttamente dalla finestra di visualizzazione comporta la chiamata del metodo *set_parameter* della classe Manager precedentemente affrontato.
Infine, il codice presentato nella sezione A.1.3 fornisce un esempio di utilizzo della classe Widget (e Manager) per l'analizzatore di spettro Signal Hound BB60C.

4.2.3 Applicazione e misure

Dopo aver delineato le specifiche del dispositivo Signal Hound BB60C e aver esaminato attentamente la piattaforma software creata per garantirne un utilizzo efficiente e flessibile, questa sezione si concentra sull'applicazione pratica del software sviluppato nel contesto dell'esperimento descritto nella sezione 2.3.1. Una delle motivazioni fondamentali che hanno guidato

questa fase dell'esperimento è stata la necessità di esaminare le sfide legate alla trasmissione di impulsi ottici a 775nm su lunghi tratti di fibra ITU-T G.652, pensata per luce tra i 1310nm e 1625nm. Essendo l'obiettivo quello di utilizzare gli stessi impulsi generati dal laser che pompa il cristallo non-lineare per la generazione dei fotoni entangled per la distribuzione della frequenza, era importante verificare che il canale ottico non introducesse distorsioni negli impulsi tali da rendere impossibile questa sincronizzazione. In particolare, si temevano possibili effetti di "spatial mode dispersion" dato che la fibra in questione diviene multimodo a 775nm. Questi potrebbero infatti causare una duplicazione degli impulsi e dunque della frequenza del segnale ricevuto. Un analisi spettrale del segnale generato da un fotodiodo ricevente tali impulsi è stata quindi effettuata sfruttando il software qui presentato in modo da comparare lo spettro ottenuto piazzando il fotodiodo direttamente in uscita al laser di pompa a quello ottenuto inserendo una bobina di 10 chilometri di fibra ITU-T G.652.

L'utilizzo del modello BB60C è stato cruciale per analizzare lo spettro del segnale elettrico generato dal fotodiodo, verificando che in entrambi i casi fosse ricevuto correttamente e che potesse essere utilizzato per la sincronizzazione del resto dell'elettronica. L'uso dello spool ci ha permesso quindi di simulare condizioni reali in cui il segnale ottico di sincronia debba essere trasmesso tra postazioni distanti al fine di permettere la distribuzione dell'entanglement tra di esse.
La prima cattura dati da parte dello spectrum analyzer, senza che il segnale ottico percorresse i 10km di fibra, ha una durata di circa 30 minuti, riportata attraverso la funzione *saving_function* della classe Manager (sezione 4.2.1) su file di testo aventi, per ciascuna riga, la seguente struttura:

```
[[lista di frequenze in Hertz], [lista di potenze in dBm]]
```

Raccogliendo tali informazioni è stato possibile riportare su più grafici lo spettro del segnale acquisito in input. Disponendo di un laser di pompa con una frequenza fondamentale di 76 MHz, quando il segnale ottico emesso viene scomposto in frequenza, ci aspettiamo naturalmente di osservare uno spettro con un picco ben definito attorno ai 76 MHz stessi. Il focus di analisi è ricaduto su quest'ultimo e sull'armonica di secondo ordine associata alla frequenza di 152MHz. Concentrandoci sia sul picco principale che sull'armonica successiva, possiamo infatti verificare se il segnale sia ricevuto correttamente e valutare eventuali distorsioni o interferenze che potrebbero influire sulle nostre misure nel corso del tempo.
Di seguito è presentato lo spettro del segnale elettrico nel primo caso di analisi.

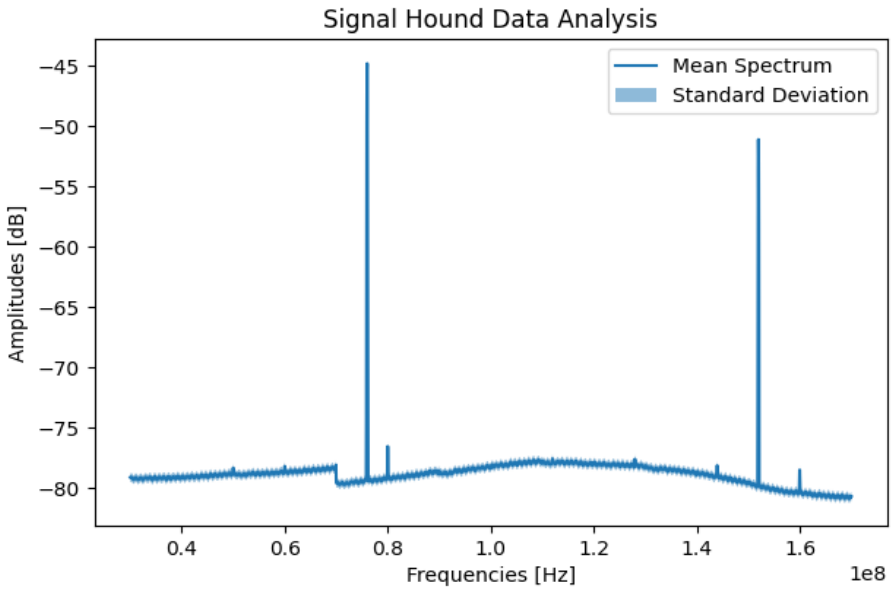


Figura 4.3: Porzione di spettro del segnale elettrico nel caso in cui non è inserita la fibra di 10km.

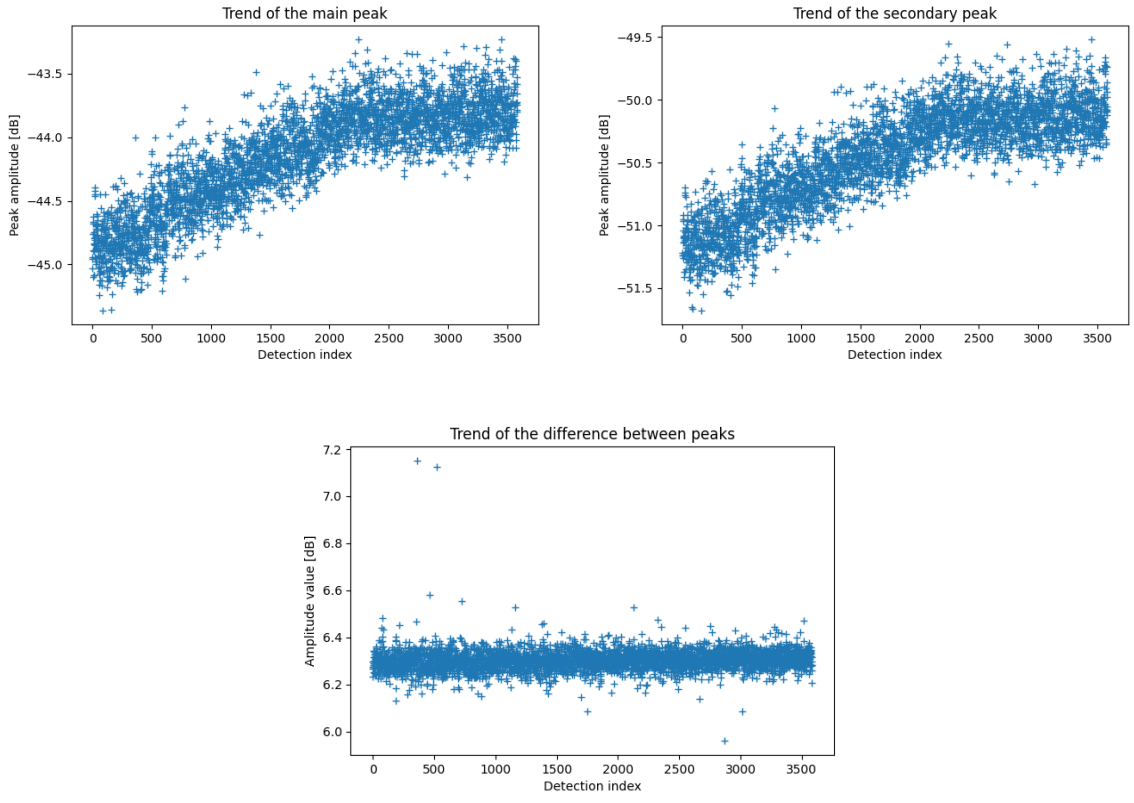


Figura 4.4: Andamento della potenza del picco principale, dell'armonica secondaria e della distanza (sempre in termini di potenza) tra le due componenti del segnale nel corso del tempo.

In particolare, la figura 4.3 rappresenta la media delle molteplici acquisizioni dello spettro del segnale, ripetute ogni 300 millisecondi dal *loop_thread* della classe Manager, per una dura-

ta complessiva di circa 3 minuti e 30 secondi. Per avere una visione completa dell'evoluzione del segnale nel corso del tempo, sono state condotte ulteriori analisi per i restanti 26 minuti e 30 secondi di cattura dei dati. Durante questo periodo, è stato registrato regolarmente il comportamento del picco principale e della sua armonica a 152 MHz, osservando come variano nel tempo. Dalla figura 4.4 si osserva un andamento pressoché identico per le componenti di interesse del segnale durante tutta la durata dell'acquisizione. Ciò comporta una differenza tra i due picchi costante (salvo rarissime eccezioni), stabilizzata attorno ai 6.3 dBm.

Anche per la seconda casistica sperimentale, durante la quale parte del segnale ottico ha percorso 10km di fibra prima di convergere sul fotodiodo, lo spettro del segnale elettrico è stato misurato per un totale di 30 minuti attraverso le classi Manager e Widget implementate . Il risultato ottenuto non si differenzia troppo da quello riportato in precedenza. La figura 4.5 presenta la media delle acquisizioni dello spettro del segnale ripetute per un minuto e quaranta cinque secondi.

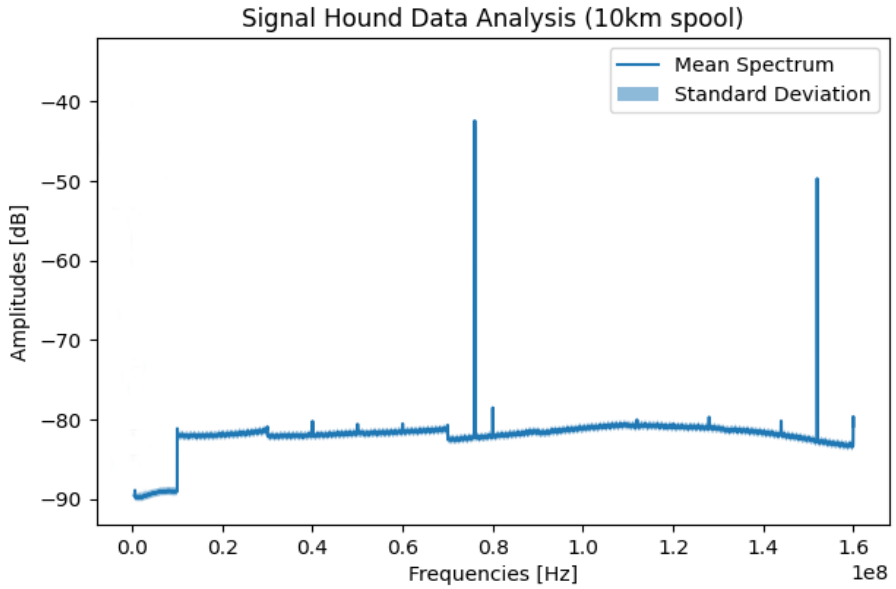


Figura 4.5: Porzione di spettro del segnale elettrico ottenuto convertendo la parte degli impulsi ottici del laser di pompa che hanno percorso 10km di fibra.

Si distinguono ancora chiaramente il picco centrale e l'armonica di secondo ordine, posizionati attorno ai 76MHz e 152MHz. Per analizzare l'intero tempo di cattura dello spettro sono stati realizzati i grafici in figura 4.6.

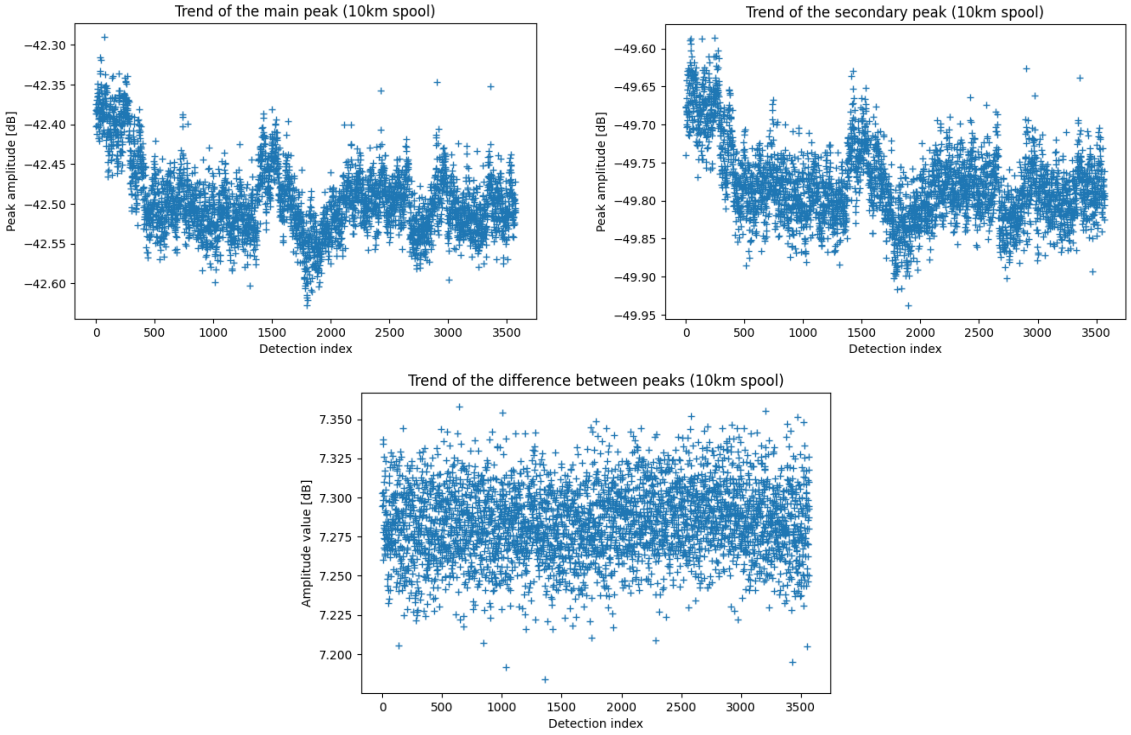


Figura 4.6: Andamento della potenza del picco principale, dell'armonica secondaria e della distanza (sempre in termini di potenza) tra le due componenti del segnale nel corso del tempo (caso in cui parte del segnale ottico ha attraversato la fibra da 10km).

Anche per questa seconda casistica sperimentale si evidenzia un andamento del picco principale molto simile a quello della seconda armonica, anche se si riscontrano leggere oscillazioni che in precedenza non si erano manifestate. La differenza tra le due componenti risulta ancora stabile, variando in un intervallo di circa 0.2dBm (tra i 7.2 dBm e 7.4dBm).

In conclusione, i risultati ottenuti dall'analisi degli spettri delle frequenze nel corso del tempo dimostrano prima di tutto la stabilità del segnale durante l'intero periodo di acquisizione dei dati. La mancanza di interferenze o distorsioni rilevanti evidenzia quindi la robustezza del sistema sperimentale implementato.
In particolare, la differenza tra il picco principale e la sua armonica secondaria, sia nel caso senza fibra ottica sia con un fibra ottica di 10 km, ha fornito risultati promettenti, mantenendosi ampia sino al termine delle catture. Era lecito aspettarsi tuttavia che tale differenza fosse più contenuta nel caso in cui si è inserita la bobina di fibra ottica, a causa di possibili distorsioni che il segnale ottico poteva riscontrare durante il suo (lungo) percorso. L'effetto più significativo riscontrato è stato invece quello di una forte attenuazione ottica (30dB osssia 3dB/km) che non permetteva un utilizzo diretto del segnale del fotodiodo per lockare la scheda *Zedboard*. È stato quindi necessario utilizzare un amplificatore e un comparatore per ristabilire un segnale sufficientemente intenso e pulito per essere usato come clock.

In sintesi, l'interfaccia interattiva implementata per agevolare l'utilizzo del Signal Hound

BB60C ha dimostrato di essere uno strumento prezioso per la gestione e l'analisi dei dati sperimentali, contribuendo a confermare la validità del setup di laboratorio destinato alla distribuzione dell'entanglement su lunghe distanze, esperimento che getta le basi per ulteriori indagini e sperimentazioni nel campo della comunicazione quantistica a lunghe distanze.

5

Sistema *Delay finder*

In relazione all'esperimento delineato nella sezione 2.3.1 della presente tesi, il capitolo seguente presenta lo sviluppo di un sistema di stima del ritardo temporale tra sequenze di rilevamenti di fotoni provenienti da coppie entangled. Più precisamente, date sequenze di tempi di rilevamento (*timetags*) di fotoni prodotte da *time taggers* distinti (e distanti), l'obiettivo è quello di determinare la differenza tra i tempi assegnati da ciascun *time taggers* alla rilevazione di due fotoni generati a coppie e inviati alle due stazioni di misura ai fini dell'esperimento. A tal fine verrà sfruttata la correlazione tra i tempi di rilevamento nelle due stazioni di misura, infatti per ogni impulso luminoso del laser di pompa che attraversa il cristallo non-lineare verrà generata una coppia di fotoni entangled solo con una bassa probabilità (tipicamente inferiore al 1%). Sarà quindi possibile "riallineare" i due assi temporali identificando il ritardo che massimizza la correlazione tra rilevamenti alle due stazioni. Purtroppo però non tutti i rilevamenti corrispondono a fotoni provenienti da coppie entangled e le perdite di canale fanno si che la maggior parte delle generazioni di coppie entangled culminino con la segnalazione di avvenuta ricezione di un fotone da parte di uno solo dei due canali o da nessuno dei due. Ciò influenza negativamente la correlazione trai i *timetags* dei due ricevitori e rappresenta un ostacolo da tenere in considerazione nella progettazione del sistema di riallineamento dei tempi di arrivo dei fotoni.
Nelle sezione seguente sono trattati brevemente i passi fondamentali che costituiranno la base per l'accurata analisi dei tempi di arrivo dei fotoni entangled su ricevitori differenti. Per evidenziare le correlazioni temporali tra coppie di fotoni entangled è stata necessaria la creazione di script in linguaggio python, alcuni dei quali verranno presentati nelle sezioni successive.

5.1 Analisi fondamentali

I primi test per lo sviluppo del software di correzione del ritardo sono stati effettuati su un dataset di prova prodotto utilizzando un singolo time-tagger con due canali. Il dataset consiste in una cattura di circa 18 secondi, nella quale i tempi di arrivo dei fotoni per canale sono espressi in picosecondi. I canali, identificati come "2" e "4", registrano rispettivamente 306960 e 274861 rilevamenti.

5.1.1 Cross-correlazione: riallineamento delle sequenze dei tempi di arrivo dei fotoni

Le prime operazioni effettuate al fine di analizzare le sequenze di tempi di rilevamento nel dataset di prova, consistono nel riportare tali dati ad essere rappresentati sotto forma di sequenze binarie. Questo permette in un secondo momento di effettuare operazioni di cross-correlazione tra segnali discreti nel tempo, come verrà presentato a breve. Per poter compiere questa operazione di conversione dei dati in sequenze binarie è stato scelto arbitrariamente (solo per questa fase "iniziale") un periodo di campionamento (*sampling period*). Ciascun istante temporale rilevato (che indicheremo anche col termine di *tag* o *detection* per alludere alla segnalazione del ricevitore *time tagger*), inserito nell'array *tags_A* se rilevato dal canale di ricezione "2", viceversa inserito nell'array *tags_B* (si veda codice sottostante), è stato diviso per questo intervallo di campionamento, al fine di conoscere dopo quanti sampling periods la rilevazione temporale sia avvenuta.

```
tags_A = tags[channels == 2]
tags_B = tags[channels == 4]
sampling_period = int(1e7) # in picosecondi

quotients_A = np.array(tags_A // sampling_period, dtype='int64')
quotients_B = np.array(tags_B // sampling_period, dtype='int64')
```

A tal punto, inserendo in un array binario un "uno" per ciascuno di questi indici interi, lasciando uno "zero" in tutte le altre posizioni, si ottengono le sequenze che evidenziano in quali periodi di campionamento ricadono gli istanti di tempo che rivelano l'arrivo dei fotoni.

```
bins_A = int(tags_A[-1] // sampling_period) +1
bins_B = int(tags_B[-1] // sampling_period) +1
detections_A = np.zeros(bins_A, dtype='uint16')
detections_B = np.zeros(bins_B, dtype='uint16')
detections_A[quotients_A] = 1
detections_B[quotients_B] = 1
```

Si può ora procedere con l'operazione di cross-correlazione tra le sequenze binarie così determinate.

Intuitivamente la cross-correlazione (correlazione incrociata) è una misura di similitudine tra segnali nel dominio del tempo. Nel nostro caso tali segnali sono discreti (sequenze di ”zero” e ”uno”) e risulta un'operazione di fondamentale importanza per ”riallinearli”, ovvero per determinare lo spostamento di uno rispetto l'altro affinché la somiglianza tra essi sia massimizzata. Fornendone una definizione più formale, adottando quella maggiormente illustrata nei volumi di teoria dei segnali, la cross-correlazione tra due segnali discreti **a** e **v** è definita come segue:

$$c_k = \sum_n a_{n+k} \times \bar{v_n}$$

dove $\bar{v_n}$ indica il complesso coniugato dell'elemento v_n. La formula indica che ogni elemento di c_k è dato dalla somma dei prodotti punto a punto tra le sequenze (nel nostro caso binarie) **a** e **v**, per ogni shift compiuto. Tale definizione è riportata anche nella documentazione della funzione *numpy.correlate*, utilizzata nello script python per il calcolo della correlazione incrociata tra le sequenze binarie. La modalità adottata per tale operazione è *valid*: l'output contiene solo le cross correlazioni che possono essere calcolate finché la sequenza più corta è completamente contenuta nella più lunga, senza aggiungere zeri ai bordi dei segnali.
Nel nostro modello, la somiglianza tra le sequenze binarie è massimizzata nel momento in cui si trova il massimo tra i prodotti scalari calcolati per ogni shift applicato. Il numero di shift k per il quale è determinato tale massimo corrisponde allo spostamento da applicare ad una delle due sequenze per il loro corretto riallineamento (in sampling period). Nel caso della funzione *numpy.correlate* adoperata, è la sequenza binaria passata come secondo parametro che verrà fatta traslare sulla prima, da sinistra verso destra, calcolando, per ogni spostamento, il prodotto scalare come di sopra riportato.

```
padding_param = 100
padded_detections_B = np.pad(detections_B, (padding_param, padding_param))
conv_result = np.correlate(padded_detections_B, detections_A)
```

Prima dell'operazione di correlazione è stato applicato un *padding* a sinistra e a destra dell'array binario *detections_B*. Aggiungendo zeri ai bordi delle sequenze di input prima della cross correlazione (nel nostro caso cento ”zeri” a inizio e fine di *detections_B*), è possibile controllare la quantità di shift per i quali sarà calcolato il prodotto scalare tra le due sequenze. Infatti se le sequenze corrispondono ad acquisizioni più lunghe del delay massimo che ci si aspetta tra i due *time-tagger* calcolare il prodotto scalare tra le due sequenze soltanto per il sottoinsieme di shift che si reputa plausibile può portare ad un'importante riduzione dei tempi di esecuzione.

In figura 5.1 è raffigurato il grafico che illustra il picco di coincidenze trovato eseguendo l'operazione di cross-correlazione tra i due array binari di esempio. Più precisamente il massimo numero di coincidenze è stato trovato dopo 91 spostamenti di *detections_A* su *detections_B*, pari a 91 finestre di campionamento. Considerando il padding di 100 zeri precedentemente applicato alla sequenza *B*, il risultato trovato ci indica che l'array *B* stesso era in origine ”in anticipo”

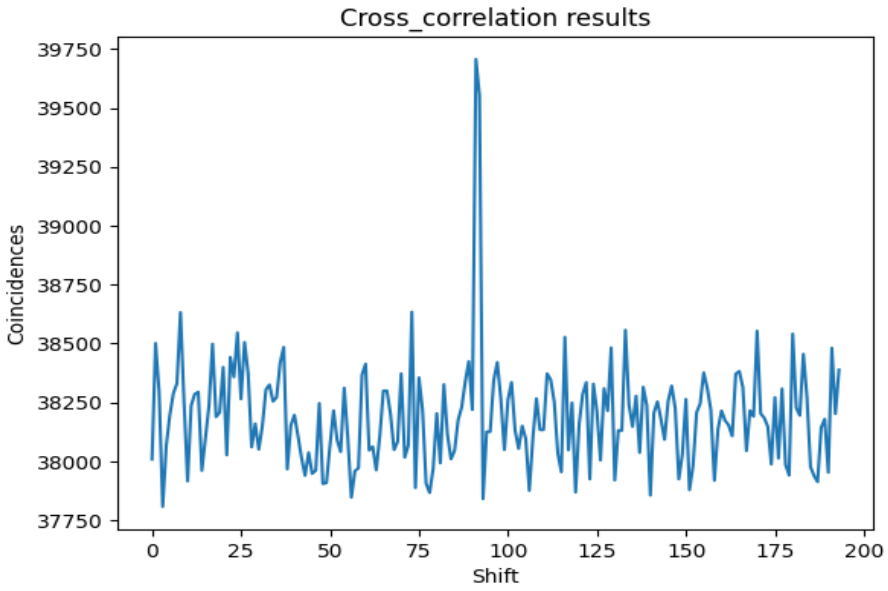


Figura 5.1: Andamento dei valori dei prodotti scalari eseguiti tra le sequenze binarie per ogni shift compiuto durante l'operazione di cross-correlazione.

di 9 periodi di campionamento rispetto l'array A (in altri termini sono stati necessari 9 intervalli di campionamento da anteporre a B per poterlo riallineare con la sequenza A).
L'operazione di cross-correlazione ci ha quindi permesso di trovare lo sfasamento temporale da applicare alle due sequenze per trovare il massimo numero di finestre di campionamento in cui sono stati registrati rilevamenti di tempi di arrivo dei fotoni da entrambi i canali di ricezione. L'intervallo temporale è però approssimativo, essendo un multiplo intero della finestra di campionamento scelta. Inoltre, è certo che non tutti gli "uno" che abbiano posizioni combacianti nelle sequenze binarie e che allo stesso tempo partecipino al massimo prodotto scalare tra esse siano associati a detections di fotoni relativi a stesse coppie entangled. Ciò è dovuto agli "errori sperimentali" menzionati in precedenza, come rivelazioni di singoli fotoni non correlati o falsi rilevamenti. Questa fase di analisi dunque costituisce la base per un'elaborazione dei dati più accurata e precisa, presentata nella sezione 5.1.3.

5.1.2 SNR: ACCURATEZZA E PRECISIONE DELLA CROSS-CORRELAZIONE

Per valutare la forma e la precisione di un'operazione di cross correlazione è stata implementata la funzione *SNR_corr* come segue:

```
def SNR_corr(corr):
  SNR = 0
  max_corr = max(corr)
  argmax_cor = np.argmax(corr)
  mean_corr = (np.sum(corr[:argmax_cor-1]) + np.sum(corr[argmax_cor+2:]))/(
    len(corr)-3)
```

```
  std_corr = np.std(np.concatenate((corr[:argmax_cor-1],corr[argmax_cor+2:]))
    )
  if std_corr > 0:
    SNR = abs((max_corr-mean_corr))/std_corr
  return SNR
```

Il termine SNR richiama il concetto di ”rapporto segnale rumore” (*signal to noise ratio*), in quanto la funzione cerca di esprimere la relazione tra la potenza massima del segnale (picco della cross-correlazione) e la potenza del rumore (fluttuazioni nel resto della cross-correlazione). Più in particolare, la misura implementata dalla funzione è denominata ”Z-score”, un procedimento matematico che permette di calcolare la distanza del picco massimo dalla media dei valori in termini di deviazioni standard. La media aritmetica e la deviazione standard degli elementi della correlazione incrociata sono state calcolate escludendo il picco della correlazione, nonché il valore ad esso immediatamente precedente e successivo.
Nel caso in cui la deviazione standard così calcolata risulti zero, anche il valore dell'SNR restituito in output sarà zero. Una deviazione standard pari a zero equivale infatti ad avere una sequenza di elementi tutti uguali. Ciò avviene per i dati della cross-correlazione quando si interviene con un periodo di campionamento troppo elevato nel processo di generazione delle sequenze binarie di interesse, esposto nella sottosezione precedente. Almeno un tempo di arrivo dei fotoni ”ricade” dunque in ciascuna finestra di campionamento, portando il corrispondente elemento della sequenza binaria ad essere identificato col valore ”uno”. Da una cross-correlazione di sequenze contenenti solo valori ”uno” non si ottiene alcun picco di interesse, ma unicamente rumore: l'SNR in questo caso deve essere zero.
In conclusione, la funzione restituisce un alto valore dell'SNR nel momento in cui il picco sia ben distinto dal rumore, mentre un valore basso può indicare che il picco sia poco distinguibile o non significativo.

5.1.3 Istogramma: ritardo tra detections relative a coincidenze

Una volta determinato l'intervallo temporale che consente il riallineamento delle sequenze binarie per massimizzarne il prodotto scalare, è possibile proseguire con l'elaborazione dei dati sin qui ottenuti, con l'obiettivo di conoscere più precisamente le differenze temporali che intercorrono tra tempi di arrivo di fotoni relativi a stesse coppie entangled sui diversi canali di ricezione.

```
if delay_in_sampling_periods>0:
  coincidences = np.array(detections_A[0:min(bins_A, bins_B)][
    delay_in_sampling_periods:] * (detections_B[0:min(bins_A, bins_B)][0:-
    delay_in_sampling_periods]))
else:
```

```
coincidences = np.array(detections_B[0:min(bins_A, bins_B)][-
  delay_in_sampling_periods:] * (detections_A[0:min(bins_A, bins_B)][0:
  delay_in_sampling_periods]))
```

Nella porzione di codice di sopra riportata, *delay_in_sampling_periods* corrisponde nel nostro esempio proprio al valore 9, numero di periodi di campionamento, determinato in precedenza, che indica di quanto la sequenza *B* (originaria, senza padding aggiunto) fosse in anticipo rispetto ad *A*. E' un numero positivo, dunque viene eseguito il prodotto scalare tra le sequenze binarie di interesse *detections_A* e *detections_B* con precisi accorgimenti: dapprima gli array vengono riportati ad avere stessa lunghezza, dopodiché sono considerati gli elementi della sequenza *A* solo a partire da quello di indice pari a *delay_in_sampling_periods* e, allo stesso modo, si considerano gli elementi della sequenza binaria *B* con esclusione di un numero di elementi finali pari a valore stesso di tale variabile. Queste operazioni permettono di applicare il prodotto scalare tra le sequenze binarie nella posizione che assumevano in corrispondenza dello shift relativo al numero massimo di coincidenze raggiunto. In altri termini, viene effettuato il prodotto punto a punto tra gli array dopo aver traslato correttamente *A* dell'intervallo temporale corretto, applicando così il riallineamento voluto.
Il caso avente *delay_in_sampling_periods* negativo, identifica la situazione in cui sia la sequenza *A* ad essere in anticipo rispetto alla *B* (e che dunque sia stato necessaria una traslazione di *A* di un quantitativo di sampling periods superiore al parametro di padding per il riallineamento delle sequenze). Prima di poter eseguire il prodotto punto a punto tra gli array è necessario applicare gli *slicing* "inversi", indicati dopo la condizione *else* del codice.
L'array *coincidences* così ottenuto è binario. Di forte interesse sono gli indici delle proprie celle in corrispondenza delle quali si trovi un "uno". Tali indici rappresentano fattori interi che individuano in quale finestra temporale sia avvenuta almeno una rilevazione dei ricevitori di **entrambi** i canali.

```
coincidences_indexes = np.where(coincidences == 1)[0]
```

Risulta ora possibile risalire direttamente ai valori degli istanti temporali di arrivo dei fotoni, collezionati negli array *tags_A* e *tags_B*, relativi proprio a tali indici, per poterne successivamente calcolare le differenze:

```
tag_of_coincA = np.array([bin_tag_dictA.get(tag_index + max(0,
    delay_in_sampling_periods)) for tag_index in coincidences_indexes])
tag_of_coincB = np.array([bin_tag_dictB.get(tag_index - min(0,
    delay_in_sampling_periods)) for tag_index in coincidences_indexes])
```

dove *bin_tag_dictA* rappresenta il dizionario che associa ad ogni indice numerico (chiave del dizionario) il tag di *tags_A* in posizione individuata da tale indice (valore del dizionario). Analogo ragionamento vale per *bin_tag_dictB*.
Risulta ora possibile calcolare le differenze temporali tra istanti di rilevamento dei fotoni relativi alle coincidenze trovate a seguito del riallineamento delle sequenze binarie.

```
diff_of_coinc_tags = tag_of_coincA - tag_of_coincB
```

L'operazione che conclude tale fase di analisi prevede la creazione di un istogramma in grado di illustrare con maggiore precisione rispetto all'operazione di cross-correlazione il ritardo tra tempi di arrivo dei fotoni correlati. Il grafico è riportato in figura 5.2.

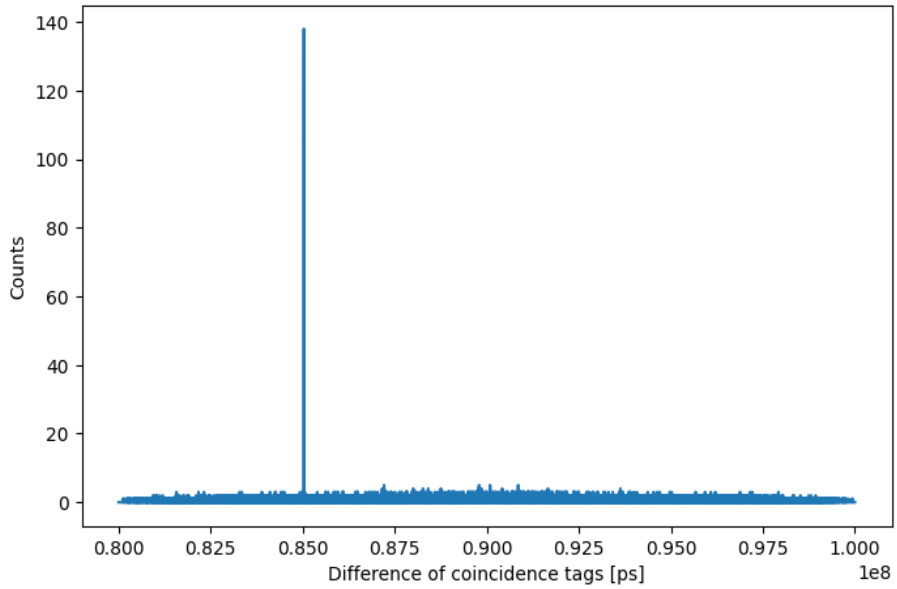


Figura 5.2: Istogramma che illustra le differenze tra tempi di arrivo di fotoni relativi alle coincidenze registrate a seguito del riallineamento degli array binari nell'esempio di prova.

L'istogramma è centrato orizzontalmente sull'istante temporale che determinava il ritardo tra le sequenze binarie di prova, calcolato nella fase di analisi precedente con le operazioni di cross correlazione: 9 sampling periods, con sampling period di 10μs (in figura è espresso in picosecondi). La regione di ricerca (span) è di due periodi di campionamento. L'ampiezza degli intervalli dell'istogramma finalizzati al conteggio delle differenze degli istanti temporali è pari a 81ps (tale valore rappresenta l'unità di misura utilizzata unicamente nel dataset di prova per collezionare i rilevamenti temporali dei fotoni nei canali di ricezione).
Il grafico dell'esempio individua un picco di quasi 140 conteggi di differenze di tempi di arrivo, ben distinto dal "rumore" di fondo. L'analisi fornita è dunque in grado di separare la differenza tra gli istanti di arrivo dei fotoni "realmente" correlati da quelle relative a segnalazioni "errate" (rilevamenti di singoli fotoni, falsi rilevamenti) che, dunque, non sono da considerare. Risulta inoltre evidente la maggiore precisione apportata nel calcolo di tale differenza temporale: circa 85μs contro i 90μs calcolati in precedenza.

5.1.4 Risultati finali: zoom dell'istogramma e fit gaussiano

Nonostante l'istogramma abbia già incrementato l'accuratezza nell'elaborazione dei dati di prova presentati, è possibile raffinare ulteriormente la ricerca del ritardo tra i time tags di fotoni

correlati.
Aumentando lo zoom nella finestra di ricerca dell'istogramma è possibile tracciare un fit gaussiano nella regione del grafico prossima al picco trovato, come mostrato in figura 5.3.

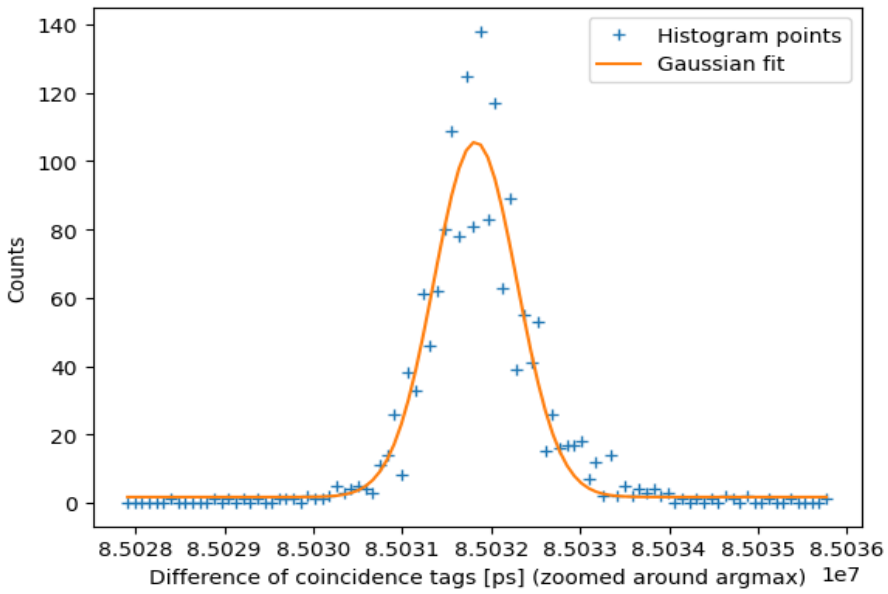


Figura 5.3: Fit con curva gaussiana su finestra ridotta dell'istogramma tracciato, fondamentale per un'analisi ancor più raffinata dell'andamento delle differenze dei tempi di arrivo dei fotoni sui canali di ricezione.

L'ingrandimento è compiuto su una finestra di ricerca di circa 100 intervalli (*bins*) disposti lungo l'asse x centrati sul valore della media della curva gaussiana. Quest'ultima fase di analisi restituisce il valore finale del ritardo trovato tra i tempi di arrivo di fotoni correlati con il proprio intervallo di confidenza: $85031823 \pm 17 ps$, rispetto al valore di $85031882 ps$ determinato con l'istogramma senza fit gaussiano.

5.2 Funzione *Delay Finder*

Tutte le fasi di analisi presentate, finalizzate a determinare il ritardo che intercorre tra tempi di arrivo di fotoni relativi a coppie correlate, sono state incorporate in un'unica funzione python, denominata *delay_finder*. In precedenza però, alcuni dei parametri fondamentali per l'elaborazione dei tempi di arrivo dei fotoni (come il periodo di campionamento o l'ampiezza degli intervalli distribuiti lungo l'asse dei tempi dell'istogramma) erano stati determinati arbitrariamente, per rendere apprezzabile l'analisi del dataset di prova. L'obiettivo è ora quello di consentire alla funzione di interpretare in maniera flessibile i dataset forniti come input, adattando di conseguenza i parametri di analisi, per ottenere un risultato finale accurato. L'ultima versione di *delay_finder* è presentata alla sezione A.2.1.

5.2.1 Parametri di input

La dichiarazione della funzione è la seguente:

```
def delay_finder(tags1, tags2, delay_max, bin_size=None, peak_width=1e2)
```

- *tags1* e *tags2* (*numpy array con dtype=int64*) rappresentano gli array dei tempi di arrivo dei fotoni nei due canali di ricezione. Contengono elementi espressi in picosecondi.
- *delay_max* (*float*) è il parametro che individua il massimo ritardo che si desidera considerare nell'analisi delle coincidenze tra i segnali. La funzione non eseguirà dunque ricerche di intervalli temporali che siano più grandi di tale valore per determinare il risultato finale. Come sarà presentato nelle sezioni seguenti, è di fondamentale importanza per il calcolo di ulteriori dati fondamentali per l'elaborazione degli array di input.
- *bin_size* (*float*) rappresenta la dimensione di ciascun intervallo posto lungo l'asse dei tempi dell'istogramma, grazie alla quale vengono conteggiate le differenze temporali tra le coincidenze. Se non specificato, la funzione determinerà automaticamente una dimensione adeguata, come mostrato in seguito.
- *peak_width* (*float*) è il parametro che indica una stima della larghezza approssimativa del picco dell'istogramma tracciato nella fase finale di analisi (espressa in picosecondi). Tale larghezza dipende principalmente dalla durata degli impulsi ottici e dal jitter dei rilevatori. Può essere specificato manualmente, in caso contrario assume il valore assegnato di default, trattato più a fondo in seguito.

5.2.2 Periodo di campionamento e complessità della correlazione incrociata

Il periodo di campionamento è il parametro cruciale per l'analisi dei dati temporali forniti in input. Come ampiamente visto nelle sezioni dedicate all'analisi del dataset di esempio, il suo valore è importante perché definisce la risoluzione temporale attraverso la quale vengono analizzati i dati e determinato il risultato finale. Qui di seguito sono presentate le considerazioni che definiscono la modalità di calcolo di tale parametro affinché risulti consistente con i dati forniti in ingresso, indipendentemente da quanto possano variare.
L'operazione che richiede maggior costo computazionale è certamente la cross-correlazione tra le sequenze binarie di interesse. Il trade-off principale consiste nel fornire un sampling period che consenta di effettuare un'analisi precisa e accurata dei dati di input ma che allo stesso tempo non complichi eccessivamente i calcoli della correlazione incrociata. Ad esempio, un periodo

di campionamento ”piccolo” giova fortemente alla risoluzione delle finestre temporali entro le quali vengono registrate le segnalazioni dei tempi di arrivo dei fotoni (finestre temporali più piccole rileveranno meglio singole segnalazioni, raffinandone la scansione). Tuttavia, incrementa a sua volta il numero di slot temporali registrati e dunque di elementi delle sequenze binarie di interesse per l'operazione di cross-correlazione, aumentandone il tempo di esecuzione. Al contrario, un sampling period ”grande” determina intervalli temporali in cui ricadono troppi tags con una conseguente perdita di precsione di analisi nonostante consenta di operare su un minor numero di elementi binari successivamente generati. In figura 5.4, ottenuta attraverso test sperimentali, è illustrata la proporzionalità quadratica inversa del tempo di esecuzione della correlazione incrociata al variare del sampling period. Come ci si aspetta, per periodi di campionamento inferiori risultano tempi di esecuzione più alti.

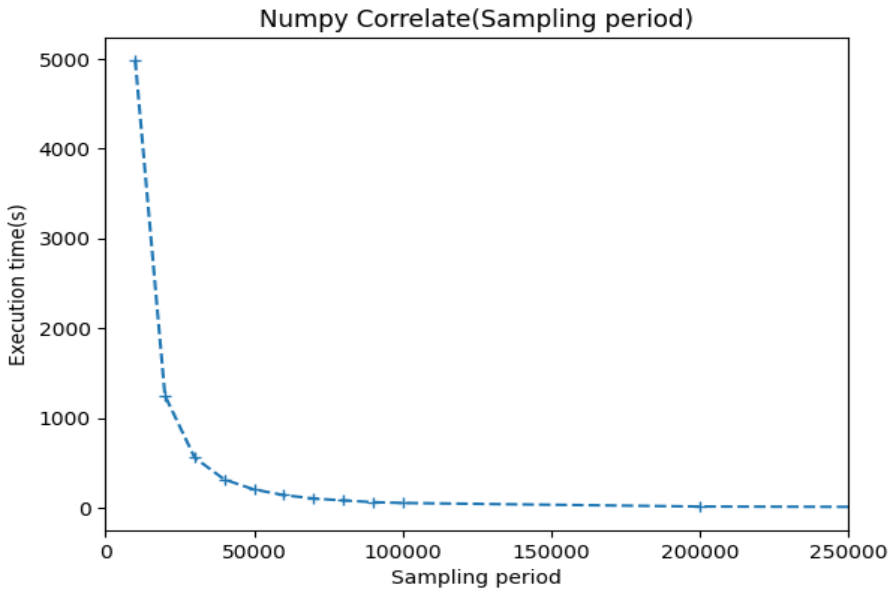


Figura 5.4: Andamento quadratico del tempo di esecuzione dell'operazione di cross-correlazione tra sequenze di detections binarie rispetto al periodo di campionamento utilizzato.

La durata di esecuzione dell'operazione di cross-correlazione dipende anche dal tempo di acquisizione impiegato. Ad acquisizioni di dati più durature corrispondono più detections registrate dai time taggers.
Linearmente cresce il numero di prodotti tra gli argomenti binari delle sequenze generate, mentre gli shift compiuti tra ogni prodotto scalare completo, a parità di sampling period e lunghezza di padding, restano inalterati.
Infine, il tempo necessario affinché la correlazione incrociata tra le sequenze binarie sia completata dipende linearmente dalla lunghezza di padding (figura 5.5) e dal ritardo massimo tra i due canali che desideriamo poter compensare: più ampie sono tali misure e più shift (e quindi prodotti scalari tra sequenze) verranno effettuati durante l'operazione di correlazione.

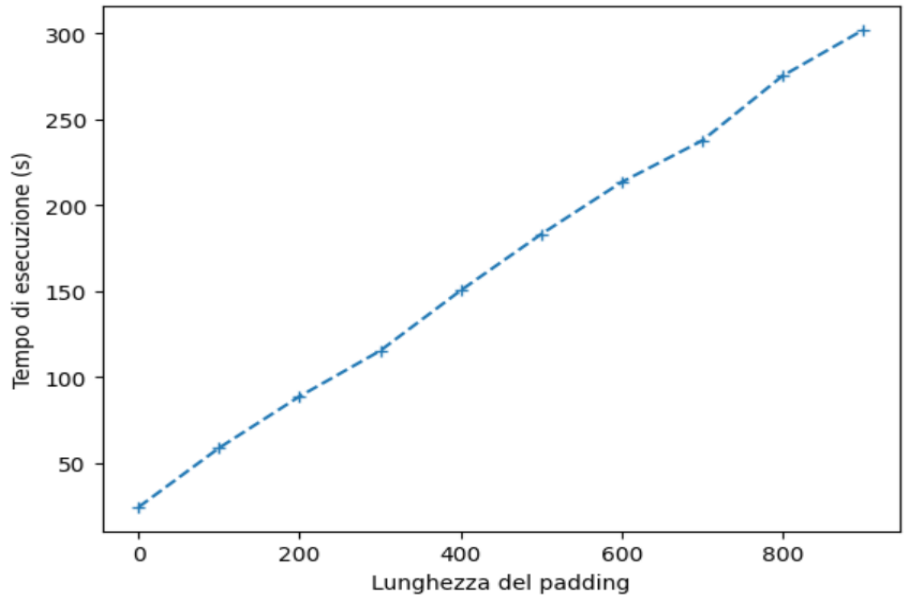


Figura 5.5: Tempo di esecuzione dell'operazione di cross-correlazione tra sequenze di detections binarie lineare in funzione della lunghezza di padding applicata ad una di esse.

Per formalizzare matematicamente il tempo di esecuzione della cross-correlazione tra sequenze binarie, esprimiamo la lunghezza di padding L come il rapporto tra il delay massimo (D_{max}) che si vuole calcolare (terzo parametro di input della funzione *delay_finder*) e il periodo di campionamento (Ts), che vorremmo determinare.

$$L = \frac{D_{\max}}{T_s} \tag{5.1}$$

Supponendo per semplicità che gli array binari di detections abbiano stessa lunghezza prima del padding, L rappresenta il numero di shift (numero di finestre temporali che costituiranno appunto il ritardo da determinare) per i quali verrà calcolato il prodotto scalare tra i due segnali. Tale numero sarà approssimativamente $2L+1 \sim L$. Per ogni prodotto scalare, sarà necessario eseguire un numero di prodotti pari a T_c/T_s (rappresenta il numero totale di campioni temporali nel dataset), dove T_c indica il tempo di acquisizione.
In tal modo, la complessità temporale complessiva dell'algoritmo di correlazione è esprimibile come

$$T_{\text{exe}} = \frac{T_c \times L}{T_s} = \frac{T_c \times D_{\max}}{T_s^2} \tag{5.2}$$

in accordo con le riflessioni precedentemente delineate.

Risulta quindi evidente che spingersi alla ricerca di vasti ritardi tra tags (aumentando il parametro di analisi *delay_max*) porta ad un incremento dei tempi di attesa per il completamento delle operazioni di correlazione incrociata. A questo scopo, è stata determinata una costante

(che denomineremo c) che consente di mantenere il sampling period consistente con l'aumento del parametro di delay, per il mantenimento dei tempi di esecuzione dell'algoritmo costanti. Sperimentalmente si è determinato che, con un tempo di acquisizione T_c di circa 18 secondi, *delay_max* impostato a 9 ms e con un sampling period di 10μs, il tempo di esecuzione dell'operazione di cross correlazione, applicata alle sequenze binarie generate con questo set di parametri, si aggira attorno ai 2 secondi. Si può dunque procedere a determinare la costante c desiderata, passando prima per la costante k "nascosta", determinante per considerare ulteriori parametri omessi che condizionano le tempistiche di esecuzione dell'algoritmo:

$$T_{\text{exe}} = \frac{k \times T_c \times D_{max}}{T_s^2} \tag{5.3}$$

da cui risulta, con le dovute sostituzioni numeriche, $k \sim 10^3$, esprimendo tutte le grandezze temporali in picosecondi.
Proseguendo e invertendo la formula ottengo

$$T_s = \sqrt{\frac{k \times T_c \times D_{\text{max}}}{T_{\text{exe}}}} = \sqrt{\frac{10^3 \times T_c \times D_{\text{max}}}{2 \times 10^{12}}} \approx \sqrt{T_c \times D_{\text{max}}} \times 2 \times 10^{-5} \tag{5.4}$$

L'espressione in Eq. 5.4 fornisce, in conclusione, la modalità di calcolo del periodo di campionamento, determinando inoltre la costante c ricercata (pari al valore approssimato di 2×10^{-5}).

5.2.3 Limite superiore al periodo di campionamento

```
SAMPLING_MAX = 3e6
min_acq_time = min(tags1[-1], tags2[-1])
time_const = 2e-5
period_of_sampling = int(max(int(peak_width), min((np.sqrt(min_acq_time*
    delay_max)*time_const), SAMPLING_MAX)))
```

Le righe di codice di sopra illustrate riportano la modalità di calcolo della lunghezza di padding (in accordo con l'equazione 4.1) e la procedura attraverso la quale il sampling period viene determinato nella funzione *delay_finder*. Come si può osservare, è rispettata l'espressione in Eq. 5.4, ma viene introdotto con la costante *SAMPLING_MAX* un limite superiore ai valori che può assumere il periodo di campionamento. Come anticipato nelle sezioni precedenti, una finestra di campionamento troppo alta diminuisce la precisione delle operazioni di analisi dei dati in input, portando *delay_finder* stessa a fallire nel determinare il ritardo tra i tempi di arrivo dei fotoni.
Il valore di *SAMPLING_MAX* è stato determinato sperimentalmente. L'operazione di cross-correlazione è stata ripetuta sulle sequenze binarie generate a partire dagli array di tags del dataset,

usando di volta in volta sampling period diversi. Il valore dell'SNR così ottenuto per ciascuna esecuzione è riportato in Fig. 5.6.

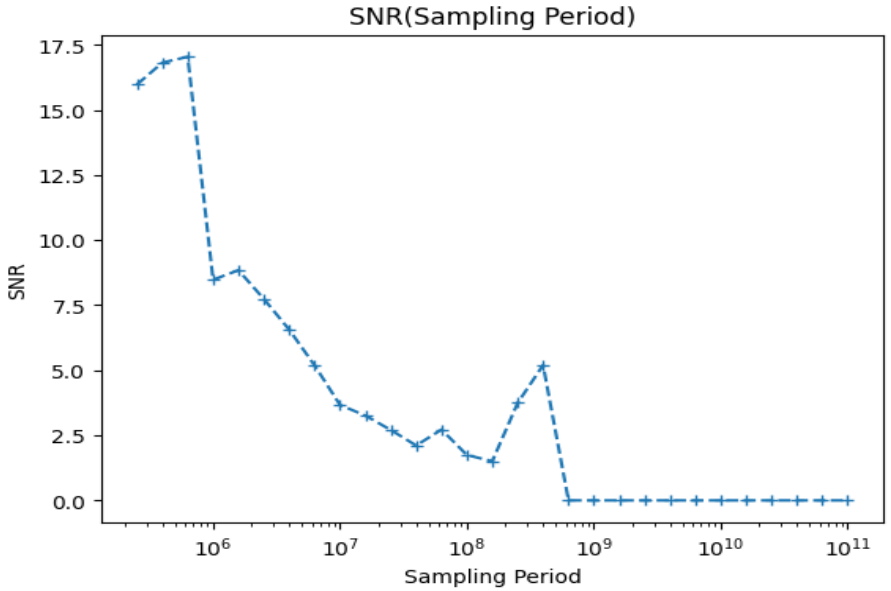


Figura 5.6: Andamento dell'SNR in relazione alle variazioni di sampling period nel calcolo della correlazione incrociata.

Dal grafico è ben evidente la decrescita di precisione della correlazione con l'incremento del sampling period. Il picco nella correlazione si confonde infine con le fluttuazioni casuali per sampling period che raggiungono (e superano) i millisecondi. L'SNR resta apprezzabile per valori che sull'asse dei tempi precedono le decine di micro secondi. Come limite superiore al sampling period è stato dunque scelto il valore di 3 micro secondi, in corrispondenza del quale l'SNR si mantiene abbondantemente sopra il valore di 5. Tale scelta è giustificata anche da analisi sperimentali riportate alla sezione 5.3.2 (successi e fallimenti di *delay_finder*).

In conclusione, le righe di codice di *delay_finder* precedentemente riportate valutano se il periodo di campionamento calcolato con l'espressione in Eq. 5.4 sia inferiore al massimo valore che possa assumere. Successivamente, verificano che il periodo di campionamento sia almeno uguale alla larghezza del picco principale dell'istogramma in Fig. 5.3, in modo da mantenere un singolo picco nella correlazione.

5.2.4 Parametri per l'istogramma

Il periodo di campionamento non è l'unico parametro ad essere determinato dalla funzione a seconda del dataset fornito in input. Un procedimento analogo è infatti riservato al parametro *bin_size* nel caso in cui non venga specificato direttamente dall'utente (come visto alla sezione 5.2.1).
Impostando un valore appropriato per *bin_size*, è possibile regolare la risoluzione dell'istogramma finale. Un *bin_size* più piccolo crea più intervalli con risoluzione fine, mentre un un valore

più grande produce tali intervalli in numero minore, rendendo la risoluzione più grossolana. Se non viene fornito un valore specifico per il parametro, la funzione imposta automaticamente un valore basato sul periodo di campionamento, come segue:

```
if bin_size is None:
    bin_size = period_of_sampling // 10000
```

Il fattore 10^4 è stato scelto per mantenere una dettagliata rappresentazione dei dati nell'istogramma anche per il valore massimo che il sampling period può assumere (in tal caso vengono generati intervalli di ampiezza 300 picosecondi l'uno). Nel contesto dell'esperimento di interesse di questa tesi, difficilmente sono stati adoperati periodi di campionamento inferiori a 10^5, dunque il fattore utilizzato consente di ottenere intervalli temporali che raggiungono la dimensione minima di decine di picosecondi.

I restanti parametri trattati nella presente sezione sono *peak_width* e *zoomed_span*. Il primo, presentato alla sezione 4.2.1, ha come valore di deafult 100 picosecondi, intervallo temporale adeguato per il contesto dell'esperimento trattato alla sezione 2.3.1 per individuare il picco di conteggi di coincidenze nell'istogramma. Il secondo parametro viene determinato come è illustrato nella seguente porzione di codice di *delay_finder*.

```
if peak_width//bin_size>1:
    zoomed_span = int(6*peak_width//bin_size) #espresso in bins
```

Il parametro *zoomed_span* rappresenta la dimensione della finestra di ricerca "ingrandita" attorno al picco principale dell'istogramma, espressa in termini di numero di intervalli temporali (disposti sull'asse dei tempi del grafico). È utilizzato durante il processo di fitting gaussiano per ottenere una stima più accurata dei dati rappresentati graficamente. Tale processo è eseguito solo nel momento in cui la larghezza stimata del picco è maggiore dell'ampiezza degli intervalli dell'istogramma e si rischia così che il picco sia "scisso" in molteplici intervalli contigui, perdendo precisione e accuratezza nella rappresentazione del risultato finale. Come mostrato nelle righe di codice precedenti, se questa casistica è verificata, la finestra di ricerca ingrandita rappresentata da *zoomed_span* sarà di 6 volte la dimensione (espressa in intervalli temporali) della regione di asse dei tempi dedicata al picco dell'istogramma (risultati empirici che hanno dimostrato che 6 fosse un valore efficace per ottenere un fitting accurato del picco).

5.2.5 Limiti di prestazione

Per lo sviluppo della funzione *delay_finder* sono stati adottati accorgimenti atti a mantenerne tempi di esecuzione apprezzabili, senza gravare sull'accuratezza del risultato trasmesso in output. Nella sezione 5.2.2 sono state infatti delineate le considerazioni finalizzate a determinare la modalità di calcolo del periodo di campionamento affinché non complicasse eccessivamente

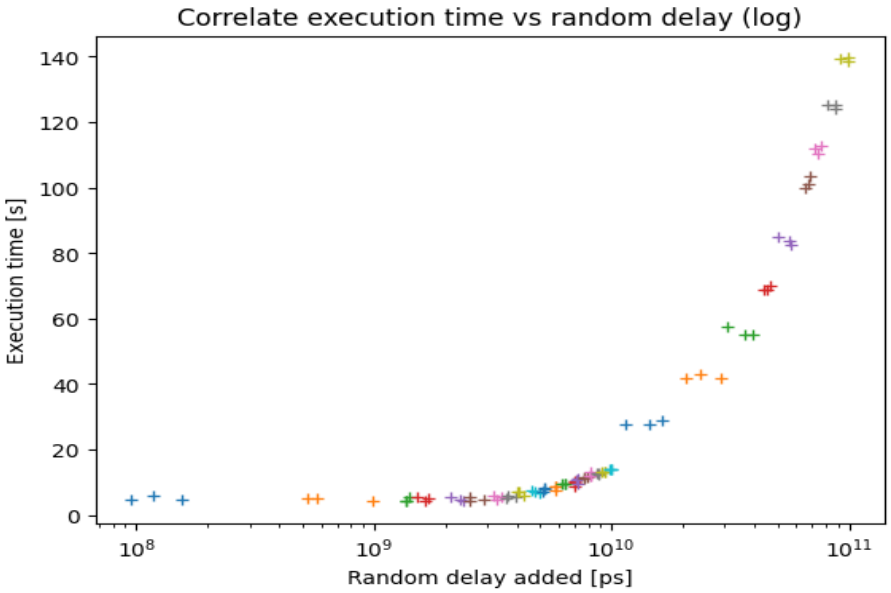


Figura 5.7: Andamento del tempo di esecuzione della funzione *delay_finder* in dipendenza dai ritardi introdotti tra i tags del dataset fornito come input.

i calcoli della correlazione incrociata. Tuttavia, si sono presentate casistiche sperimentali nelle quali la durata di esecuzione della funzione è risultata poco pratica.

La figura 5.7 ne costituisce un esempio. Nel grafico sono tracciati i tempi di esecuzione della funzione *delay_finder* in corrispondenza di ritardi introdotti artificialmente tra i tempi di arrivo dei fotoni registrati. Più in particolare, sono stati considerati 3 ritardi casuali per ciascun parametro *delay_max* col quale la funzione è stata testata, riportati sull'asse delle ascisse [*]
Come si può notare, la durata della funzione di analisi si mantiene pressoché costante per ritardi introdotti inferiori ai 10ms. L'incremento dei tempi di esecuzione a partire dai valori temporali successivi è notevole (sottolineiamo la scala logaritmica del grafico), portando la durata di *delay_finder* oltre i due minuti. Questo fenomeno è del tutto plausibile e trova una spiegazione logica nelle politiche di calcolo dei parametri fondamentali alle quali sono dedicate le sezioni precedenti.
L'espressione in Eq. 5.4 evidenziava il rapporto dinamico tra il periodo di campionamento e il valore della variabile *delay_max*: l'incremento di quest'ultima tende a innalzare il sampling period stesso. Questa relazione, tuttavia, presenta un andamento interessante: fino a quando il periodo di campionamento non raggiunge il valore corrispondente alla costante *SAMPLING_MAX* precedentemente determinata, vi è un aumento proporzionale tra *delay_max* ed esso. Dopodiché, il periodo di campionamento cessa di aumentare (mantenendosi pari a 3ms per nostra scelta) e ciò avviene al superamento di un certo valore critico di *delay_max*. La relazione tra i due parametri gioca un ruolo cruciale nelle prestazioni dell'algoritmo di cross-correlazione. Poi-

[*]Per questa prova sperimentale (e per quelle presentate a seguire) è stato sfruttato un dataset consistente in una cattura di circa 7 secondi, con canali che registrano 90153 e 130255 tempi di arrivo rispettivi.

ché il periodo di campionamento determina la risoluzione temporale con cui vengono eseguiti i calcoli, la sua evoluzione in risposta a variazioni di *delay_max* ha un impatto diretto sulla durata complessiva dell'esecuzione della cross-correlazione. Pertanto, con l'aumento di *delay_max* e la conseguente possibile saturazione del periodo di campionamento, la durata di esecuzione dell'algoritmo di cross-correlazione inevitabilmente aumenta.
Il punto di saturazione del periodo di campionamento è ben evidente nella figura 5.8. Ciò avviene per il valore critico di *delay_max* calcolabile invertendo l'espressione in Eq. 5.4 come segue:

$$D_{\text{sat}} = \frac{\left(\frac{T_s}{2\times10^{-5}}\right)^2}{T_c} \approx \frac{\left(\frac{3\times10^6}{2\times10^{-5}}\right)^2}{7\times10^{12}} \approx 3.2\times10^9 \tag{5.5}$$

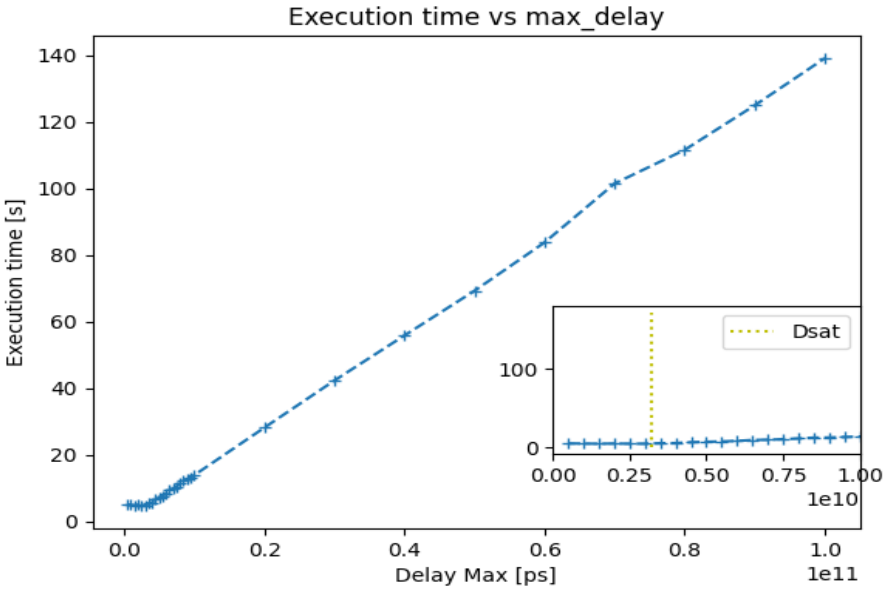


Figura 5.8: Andamento del tempo di esecuzione di *delay_finder* al variare del parametro *delay_max*. Nella parte destra della figura è presente un ingrandimento della regione iniziale della curva.

La curva di sopra riportata evidenzia bene l'incremento dei tempi di esecuzione di *delay_finder* in corrispondenza dei valori successivi al ritardo "di saturazione" *D_sat* calcolato.
Le tecniche utilizzate per risolvere i limiti relativi ai tempi di esecuzione di *delay_finder* saranno trattate nella sezione successiva.

5.3 Funzione *Delay Finder fft*

5.3.1 fft correlate

Per per risolvere i limiti relativi alle prestazioni di *delay_finder* si è optato per l'utilizzo di algoritmi basati su trasformate di Fourier (algoritmi "FFT"), in grado di eseguire (e velocizzare) le

operazioni di correlazione (si veda il capitolo 3 della presente tesi per una migliore comprensione degli argomenti seguenti). A questo scopo è stata infatti implementata *delay_finder_fft*. Nel complesso, per quanto riguarda il codice, si appresta ad essere una versione pressoché identica a quella vista per *delay_finder*. Tuttavia, l'algoritmo di correlazione tra sequenze binarie adoperato è profondamente differente.

```
def fft_correlate(detections1, detections2):
  spec1 = np.fft.fft(detections1)
  spec2 = np.fft.fft(detections2)
  cross_corr_result = np.fft.ifft(spec1*np.conjugate(spec2))
  #circular shift
  cross_corr_result = np.concatenate((cross_corr_result[len(cross_corr_result
    )//2+1*(len(cross_corr_result)%2):],cross_corr_result[0:len(
    cross_corr_result)//2+1*(len(cross_corr_result)%2)]))
  #Computing the real part of the cross_corr_result array, rounded to the
    nearest integer number
  cross_corr_result = np.array(np.round(np.real(cross_corr_result)),dtype='
    uint16')
  return cross_corr_result
```

Dapprima l'algoritmo di correlazione calcola la Trasformata di Fourier Veloce (FFT) di entrambi gli insiemi di dati, rappresentati da *detections1* e *detections2*. Successivamente, i due spettri calcolati vengono moltiplicati nel dominio delle frequenze, il che corrisponde, per teorema della convoluzione (sezione 3.0.3) a eseguire la convoluzione tra i segnali discreti nel dominio del tempo.
Dopo aver ottenuto la correlazione di Fourier, si esegue un'operazione di spostamento circolare: l'array di correlazione viene diviso a metà e le sue parti vengono scambiate. Questo permette di posizionare il picco di correlazione al centro dell'array per delay nulli.
Sfruttando la Trasformata di Fourier Veloce, *fft_correlate* è in grado di eseguire la correlazione *full* tra due insiemi di dati attraverso esecuzioni estremamente più veloci rispetto a quanto può garantire *numpy.correlate*. Tale modalità consente di eseguire il prodotto punto a punto tra le sequenze binarie di interesse in ogni punto di sovrapposizione, senza che necessariamente una delle due resti completamente sovrapposta all'altra per ogni shift compiuto (come invece accade per la modalità *valid*, di default per *numpy.correlate*).
In figura 5.9 è illustrato l'andamento del tempo di esecuzione dell'algoritmo *fft_correlate*. La curva ha andamento sperimentale che si avvicina a quello asintotico $nlog(n)$ (con n numero di elementi degli array binari sui quali è calcolata la correlazione con FFT).

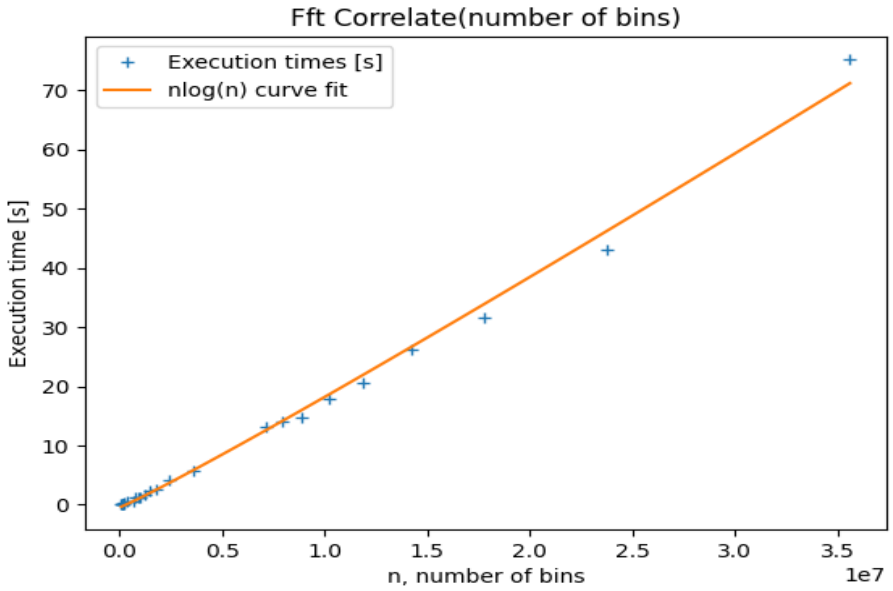


Figura 5.9: Andamento del tempo di esecuzione di *fft_correlate* in dipendenza dal numero di elementi degli array binari su cui l'algoritmo è applicato. Fit aggiuntivo con curva nlog(n).

Come anticipato, le tempistiche di computazione diretta della correlazione operata con *numpy.correlate* sono largamente superiori rispetto a quelle calcolate sfruttando la FFT in *fft_correlate*.

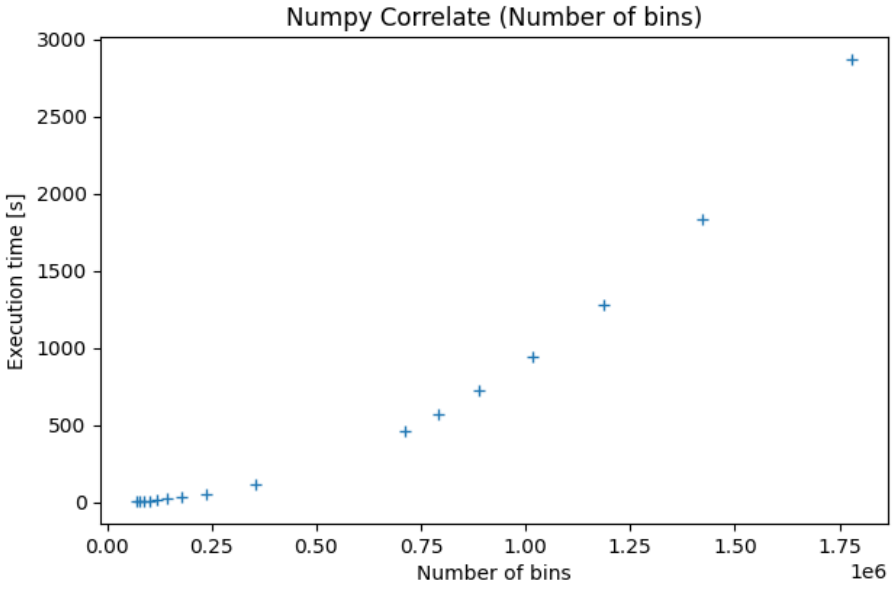


Figura 5.10: Andamento del tempo di esecuzione di *numpy.correlate* in dipendenza dal numero di elementi degli array binari su cui l'algoritmo è applicato.

Dalla figura 5.10 si notano infatti tempi di esecuzione eccessivamente elevati per quantità di elementi degli array binari nell'ordine del milione. *numpy.correlate* impiega circa 30 minuti per compiere un'operazione di correlazione incrociata tra array binari contenenti 1 milione e mezzo

di elementi, arrivando addirittura a sfiorare i 50 minuti per sequenze di un 1 milione e 750 mila componenti. L'algoritmo *fft_correlate* è invece pressoché immediato per queste moli di dati.

5.3.2 Periodo di campionamento e invocazione di Delay Finder

Le prestazioni di *fft_correlate* appena presentate sono essenziali per esporre la procedura attraverso la quale il periodo di campionamento è determinato nella funzione completa. La complessità dell'algoritmo fft è $O(nlog(n))$, dove n è il numero di elementi delle due sequenze binarie soggette all'operazione di cross-correlazione full. Il sampling period per la funzione *delay_finder_fft* è determinato attraverso una modalità che consente di mantenere costante proprio questo numero di elementi al variare della lunghezza dei due array di detections originari, contenenti i tempi di arrivo dei fotoni sui due canali di ricezione.

```
def delay_finder_fft(tags1, tags2, delay_max=100e9, peak_width=1e2):
    last_tag = max(tags1[-1],tags2[-1])
    MAX_BINS = 1e7
    SAMPLING_MAX = 3e6
    period_of_sampling = last_tag//MAX_BINS
```

Come delineato dalle righe di codice di sopra riportate, il numero di elementi delle sequenze binarie da mantenere costante è di 10^7, determinato a fronte dei risultati sperimentali forniti nella sezione precedente, in figura 5.9. In corrispondenza di questo valore, il tempo di esecuzione dell'algoritmo di correlazione si mantiene attorno ai 20 secondi.
Tuttavia, adottando questa modalità, il sampling period può assumere valori superiori alla costante *SAMPLING_MAX* determinata in precedenza. Per capire se ciò possa costituire un problema, sono stati tracciati grafici in grado di visualizzare l'accuratezza dei risultati forniti in output da *delay_finder_fft*, posta a confronto con quella di *delay_finder*. Quest'ultimi sono reperibili alla sezione A.2.3 e A.2.4, riassunti nella figura 5.11 riportata di seguito.

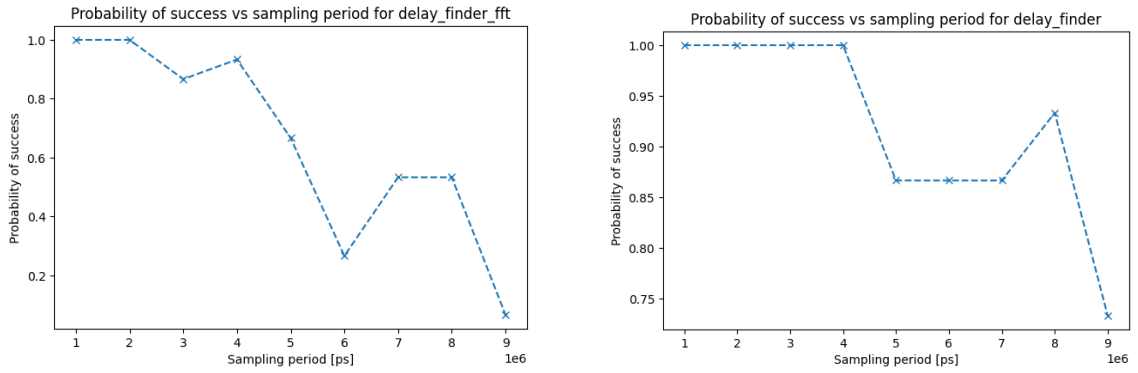


Figura 5.11: Probabilità di successo (restituzione risultato corretto) di *delay_finder_fft* e *delay_finder*

Le curve di sopra illustrate rappresentano, in termini probabilistici, la precisione di *delay_finder_fft* e *delay_finder* rispettivamente. Più in particolare, ogni punto di spezzamento dei grafici corri-

sponde all'esecuzione della rispettiva funzione per un totale di 15 volte, adoperando il periodo di campionamento riportato sull'asse x. Ciascuna di esse è stata eseguita dopo aver introdotto sperimentalmente un ritardo differente tra i tempi di arrivo dei fotoni acquisiti dai ricevitori.
Nella finestra di sinistra è immediato visualizzare un forte decremento della probabilità di successo di *delay_finder_fft* per periodi di campionamento maggiori del limite superiore. Con un SNR sempre più basso la funzione fallisce più di quella che non utilizza le FFT, fenomeno riconducibile all'operazione di cross correlazione effettuata in modalità full, da cui ne consegue quindi una correlazione più lunga e una maggiore probabilità che un picco di rumore superi casualmente quello reale.

Risulta dunque evidente l'impossibilità di eseguire *delay_finder_fft* con sampling period che superino *SAMPLING_MAX*.
La soluzione a tale problema è delineata nelle righe di codice seguenti:

```
if period_of_sampling>=SAMPLING_MAX:
    return delay_finder(tags1, tags2, delay_max, peak_width)
```

La scelta è quella di delegare l'analisi del dataset di input a *delay_finder* (con il parametro *delay_max* impostato di default a 100ms). Come si evince infatti dalla figura 5.11, la precisione della funzione priva di fft si mantiene elevata per quasi tutta la finestra di visualizzazione (subisce un crollo in corrispondenza del periodo di campionamento 9×10^6 che, che ad ogni modo, risulta essere troppo più grande del proprio limite superiore e, dunque, non verrebbe mai selezionato per l'analisi in oggetto).

5.4 Funzioni a confronto: conclusioni

Con l'esposizione dettagliata delle metodologie di calcolo dei parametri fondamentali delle funzioni *delay_finder* e *delay_finder_fft* si è fornita una panoramica completa delle tecniche utilizzate per determinare con precisione il ritardo tra le rivelazioni di fotoni correlati su diversi canali di ricezione, sia tramite l'utilizzo della cross-correlazione classica con la funzione *numpy.correlate*, sia attraverso l'approccio basato sulla trasformata di Fourier (FFT) con la funzione *fft_correlate*.
Questa sezione si pone l'obiettivo di ottenere una comprensione chiara dei punti di forza e delle limitazioni delle funzioni, consentendo così di fare un bilancio completo dell'efficacia di entrambe le soluzioni proposte.
Come visto nella sezione 5.3.1, l'algoritmo di correlazione incrociata che sfrutta le trasformate di Fourier veloci raggiunge risultati nettamente superiori a quelli mostrati dall'algoritmo di correlazione classico per la modalità "full" dell'operazione. *fft_correlate* termina infatti in pochi secondi la propria esecuzione per quantità di elementi di array binari che richiedevano decine di minuti utilizzando l'algoritmo di *delay_finder*.

D'altro canto però, un tempo di acquisizione dei dati dell'esperimento sufficientemente ampio può portare a generare lunghi array di detections. Lunghe catture dati possono risultare utili nel momento in cui i fotoni rivelati siano in numero limitato a causa della natura probabilistica dei processi quantistici e, in altre parole, non è garantito che un numero elevato di fotoni venga rivelato in modo coerente. Risulta possibile così accumulare più dati statistici e catturare più fotoni entangled rilevanti per l'esperimento. Tuttavia, come anticipato nella sezione precedente, array estesi contenenti i tempi di arrivo dei fotoni sui due canali di ricezione possono generare, per *delay_finder_fft*, un periodo di campionamento che oltrepassa il proprio limite superiore, con conseguente perdita di precisione e delegazione delle operazioni a *delay_finder*. Ad ogni modo, nel contesto dell'esperimento delineato alla sezione 2.3.1, è molto difficile che tale limite di *delay_finder_fft* si presenti. Ricordando che il periodo di campionamento viene determinato come $T_{cattura}/10^7$, servirebbe una durata di cattura dei tempi di arrivo dei fotoni di 30 secondi per raggiungere un periodo di campionamento di 3×10^6 picosecondi. Un intervallo di tempo decisamente molto ampio nel nostro caso.
La funzione che adopera le trasformate di Fourier veloci appare dunque quella più vantaggiosa, ricordando inoltre che, come visto nella sezione 5.2.5, *delay_finder* ha dimostrato di avere limiti relativi ai propri tempi di esecuzione per ritardi tra istanti di arrivo dei fotoni superiori ai 10ms. Di seguito, in figura 5.11, è riportato nuovamente il grafico che illustra tali limiti e, accanto ad esso, sono raffigurate misure di prestazioni temporali di *delay_finder_fft* per gli stessi ritardi introdotti.
Come ampiamente discusso dalla sezione 5.3.2, l'implementazione di *delay_finder_fft* prevede che il periodo di campionamento sia calcolato in modo tale da mantenere sempre costante il numero di elementi di array binari soggetti alla cross-correlazione "full", unico parametro che determina un tempo di esecuzione di tale operazione di circa 20 secondi (figura 5.9).

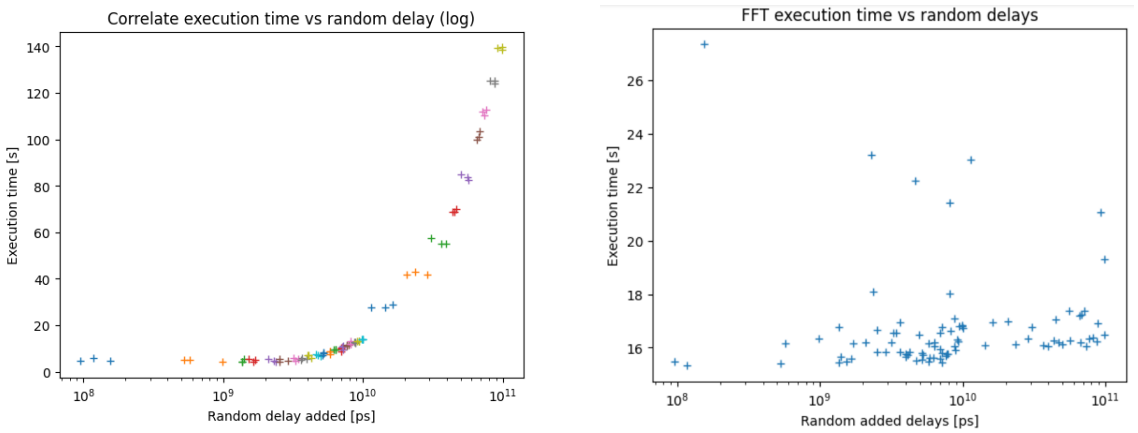


Figura 5.12: Tempi di esecuzione di *delay_finder* e *delay_finder_fft* in relazione ai ritardi sperimentalmente introdotti nel dataset di partenza.

E' ciò che appare nella finestra di destra di figura 5.12: per i *delay* temporali rispetto ai quali i tempi di esecuzione di *delay_finder* superano anche i 2 minuti, la durata di *delay_finder_fft* si

mantiene saldamente sui 20 secondi (anche se la maggior parte dei punti del grafico ricade per di più al di sotto dei 18 secondi).

In conclusione, la funzione "fft" permette di determinare velocemente ritardi tra detections correlate ben oltre i millisecondi, conservando accuratezza e precisione di calcolo per tempi di acquisizione dei dati sperimentali che non oltrepassino i 30 secondi. Appare come suo unico svantaggio l'esecuzione della correlazione sempre in modalità full, anche se si conosce a priori che il delay da determinare sia molto più contenuto della durata delle catture. Inoltre, è importante notare che in situazioni in cui il ritardo tra i tempi di arrivo dei fotoni si sospetta sia basso, inferiore al millisecondo, potrebbe essere più vantaggioso utilizzare la versione *delay_finder*. Questa scelta è motivata dal fatto che, in questi casi, *delay_finder* richiederà meno di 20 secondi di esecuzione (come è visibile in figura 5.8), evitando l'overhead aggiuntivo introdotto dalla politica di calcolo del periodo di campionamento di *delay_finder_fft*.

5.5 Applicazione

Dopo aver esaminato accuratamente le componenti teoriche e tecniche del sistema di sincronia atto a riallineare gli istanti temporali di arrivo di fotoni entangled su più ricevitori, è stato possibile analizzare le prestazioni degli algoritmi che implementano il modello realizzato. In particolar modo, nelle sezioni precedenti, grande attenzione è stata riservata alle funzioni *delay_finder* e *delay_finder_fft* per analizzarne infine i punti di forza e debolezza.
Nella sezione seguente è presentato l'utilizzo concreto del sistema implementato, adottato su dati acquisiti sperimentalmente con il setup esposto nella sezione 2.3.1.
I rilevamenti degli istanti di arrivo temporali sono effettuati dai due ricevitori A e B, che registrano rispettivamente 2'149'581 e 782'688 detections.

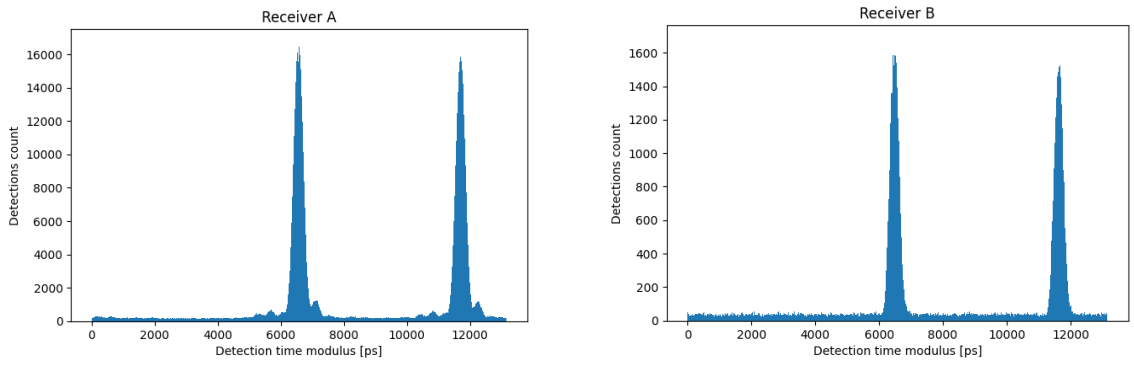


Figura 5.13: Istogrammi per visualizzare la distribuzione temporale delle detections dei ricevitori A e B rispetto al periodo di impulsata del laser utilizzato.

Una prima operazione impiegata nell'analisi corrente è visualizzare la distribuzione ciclica degli istanti temporali rispetto al periodo del laser di riferimento. Questo può essere utile per

individuare eventuali pattern o picchi nella distribuzione dei dati temporali raccolti dai ricevitori A e B.
Tracciando gli istogrammi che rappresentano la distribuzione dei tempi di arrivo dei fotoni rispetto al periodo di impulsata del laser di $\frac{1}{76\times10^6}s$, si ottiene il risultato di figura 5.13.
Le detections sono state fatte partire dal riferimento di 0 secondi per ciascun ricevitore, traslate poi di un intervallo temporale per centrare i due time-bin all'interno del periodo del laser (l'intervallo di tempo corrisponde proprio a metà del periodo del laser stesso). Per ottenere gli andamenti rappresentati in figura è stata inoltre applicata l'operazione di modulo (*np.mod()*) rispetto al periodo del laser calcolato in precedenza per ciascuna detections corrispondente ad un rilevamento di A o B. In altre parole, questo passaggio determina in quale porzione del periodo cade ogni istante temporale.
In secondo luogo, l'intervallo di cattura di interesse per l'analisi temporale dei dati acquisiti è stata ridotto a circa 5 secondi. Con il parametro *delay_max* impostato a 300 millisecondi è stata invocata *delay_finder* sugli array di detections corrispondenti ai rilevamenti dei ricevitori A e B.
Il risultato ottenuto evidenzia un ritardo temporale di circa 100 millisecondi dell'array B rispetto all'array A. Più precisamente l'output restituito è di 109'302'460'200 picosecondi, (calcolato con un periodo di campionamento pari a *SAMPLING_MAX*). Tale output è confermato anche dall'invocazione di *delay_finder_ftt* (per la quale il ritardo finale stabilito è di 109'302'249'964 picosecondi).
Risulta ora possibile riallineare le due sequenze di detections, applicando il ritardo trovato dalle funzioni implementate alla sequenza B. Come step conclusivo di analisi, in linea con quanto presentato alla sezione 5.1.3, sono state identificati i tags corrispondenti allo stesso impulso laser.

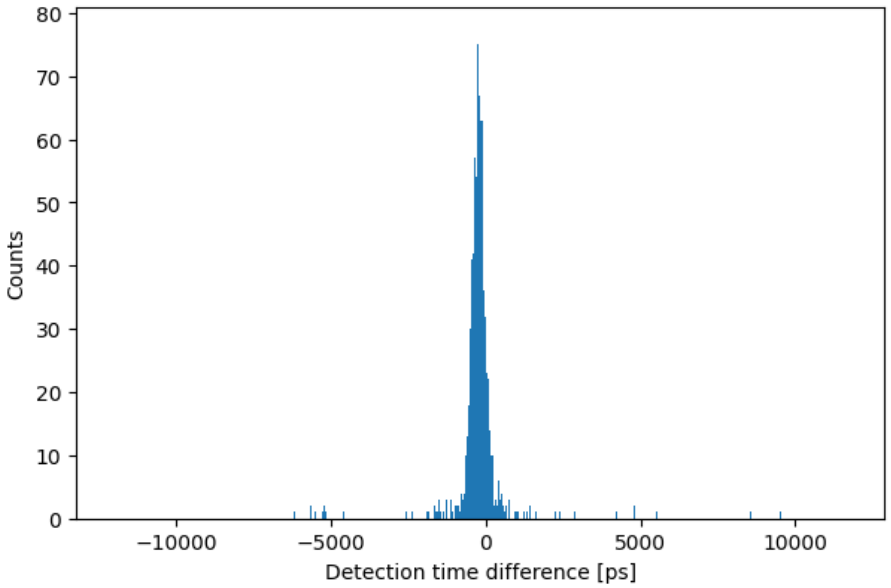


Figura 5.14: Istogramma relativo a differenze temporali tra istanti di arrivo di fotoni riconducibili a stesse coppie entangled dopo la correzione del ritardo.

Tracciando l'istogramma che illustra le differenze tra gli istanti di rilevamento di fotoni corrispondenti a coincidenze determinate a seguito del riallineamento delle sequenze binarie, si ottiene il risultato di figura 5.14
Si evince un picco ben definito pressapoco centrato sul valore zero dell'asse delle ascisse, con un conteggio massimo di quasi 80 coincidenze utilizzando bin di 2 ps). Verso -5000 ps e +5000 ps sono in visibili detections concentrate che derivano dai casi in cui sono state generate due coppie di fotoni entangled dall'impulso di pompa: una coppia dal primo impulso e una seconda dal secondo impulso (a 5 ns di distanza), ed è stata fatta la differenza tra i tempi di arrivo di fotoni appartenenti a coppie diverse ma generate nello stesso periodo del laser.

6

Considerazioni Finali

Il progetto di tesi delineato si è collocato nel contesto sperimentale di supportare la realizzazione di un sistema di sincronia per la distribuzione dell'entanglement quantistico. A sostegno di misure necessarie alla verifica della qualità e stabilità del segnale di *locking* in frequenza ottenuto tramite trasmissione di impulsi ottici, è stata sviluppata un'interfaccia grafica che consentisse di acquisire e visualizzare in maniera rapida ed efficiente i dati acquisiti dall'analizzatore di spettro in dotazione. In seguito è stato sviluppato uno script per la stima accurata del ritardo temporale tra sequenze di rilevamenti di fotoni entangled distribuiti tra sistemi di comunicazione separati. Nello sviluppo di questo software si è cercato di ottimizzare le prestazioni utilizzando algoritmi efficienti e si è valutato l'impatto dei vari parametri nelle prestazioni e nella qualità della stima. In conclusione, si ribadiscono i contributi principali di questa ricerca:

- Sviluppo di un'interfaccia grafica per uno strumento. Tale interfaccia consente il controllo dell'analizzatore di spettro *Signal Hound BB60C*, così come la visualizzazione e il salvataggio dei periodogrammi da esso prodotti. Con tale interfaccia sono state condotte misure di laboratorio per testare l'efficacia del sistema di distribuzione della frequenza. I risultati sperimentali ottenuti al capitolo 4 hanno confermato la possibilità di trasmettere come segnale di sincronia direttamente la luce usata per la generazione delle coppie entangled anche se la fibre ottica utilizzata non è stata pensata per trasmissioni a quella lunghezza d'onda (775 nm).

- Sviluppo di un sistema di stima del ritardo. Abbiamo sviluppato un sistema di sincronia temporale basato su algoritmi di correlazione incrociata e trasformate di Fourier veloci. Questo sistema è stato progettato per riallineare gli istanti temporali di arrivo di fotoni entangled su più ricevitori, consentendo di dimostrare l'avvenuta trasmissione e ricezione di fotoni entangled in ambienti separati.

- Applicazione Pratica. È stata dimostrata l'applicabilità del nostro sistema su dati acquisiti sperimentalmente con un setup specifico. Questo ha incluso l'analisi delle distribuzioni temporali delle rilevazioni dei fotoni e la correzione dei ritardi tra le sequenze di rilevamenti, determinando, al termine del capitolo 5, un risultato soddisfacente che ha potuto risolvere la problematica di stima del ritardo e riconoscimento di coppie entangled distribuite.

Il software di sincronia sviluppato sarà a breve utilizzato nei primi esperimenti di distribuzione a lunga distanze di *time-bin entanglement* senza falla della post-selezione, ma potrà tornare utile anche in esperimenti successivi di distribuzione di entanglement. L'interfaccia grafica dell'analizzatore di spettro trova invece un'utilità ancora più generica dato che potrà essere utilizzata ogniqualvolta si renda necessario prendere misure con l'analizzatore di spettro.

A

Codice sviluppato e grafici sperimentali

A.1 BB06C Spectrum Analyzer

A.1.1 manager_signal_hound_bb60c.py

```
1
2 """
3     Manager (and Device code) for Signal Hound Spectrum Analyzer bb60 series
      device
4
5     @author: Federico Muscara'
6 """
7
8 # NOTE: The BB60C does not require the use of bbSelfCal for device
      calibration. Instead, for the BB60C, if
9 # the device deviates in temperature, simply call bbInitiate again which will
       re-calibrate the device at its
10 # current operating temperature.
11
12 import PyQt6
13 import sys
14 import os.path
15 sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), os.
      path.pardir)))
16 from managers.manager import Manager
17 from devices.bb_api import *
```

```
from PyQt6 import QtCore
# NOTE: utils.simple_threads is a module wrote to simplify the use of threads in PyQt6.
from utils.simple_threads import LoopThread, SaveThread
import numpy as np

class ManagerSignalHoundBB60C(Manager):

invalid_parameter = QtCore.pyqtSignal(str)

sweep_completed = QtCore.pyqtSignal(tuple)

"""
    Initializes the ManagerSignalHoundBB60C instance by defining parameters to compute get_sweep method
    in the loop thread.
"""
def __init__(self, **kwargs):
    super().__init__()

    self.signal_hound = None
    self.is_connected = False

    #Default initialization for attributes
    self.center_freq = 1.0e6 #Center frequency in hertz.
    self.span = 1.0e6 #Span in hertz (Range from start to stop frequency).
    self.ref_level = 30.0 #Reference level in dBm.
    self.gain = BB_AUTO_GAIN #When BB_AUTO_GAIN is selected, the API uses the reference level provided in bbConfigureLevel() to choose the best gain setting for an input signal with amplitude equal to the reference level provided.
    self.atten = BB_AUTO_ATTEN #Is -1
    self.rbw = 10.0e3 #Resolution bandwidth in hertz.
    self.vbw = 1e3 #Video bandwidth in hertz.
    self.sweep_time = 10.0e-3 #Sweep time in seconds.
    self.rbw_shape = BB_RBW_SHAPE_FLATTOP #. This choice determines the window function used and the bandwidth cutoff of the RBW filter
    self.rejection = BB_NO_SPUR_REJECT #Rejection can be used to optimize certain aspects of the signal.
    self.detector = BB_MIN_AND_MAX #The detector determines the type of signal processing performed on the data.
```

```
    self.scale =  BB_LOG_SCALE #Sweeps will be returned as dBm values,
    self.proc_units = BB_LOG #To emulate a traditional spectrum analyzer,
    select BB_LOG.

    self.__start = None
    self.__bin_size = None
    self.__trace_len = None

    self.freqs = None
    self.trace_max = None

    for key,val in kwargs.items():
        setattr(self, key, val)

    #Initializing the loop thread that is going to repeat every 300 ms.
    self.command_handler = LoopThread(300)
    #Adding a callback function to the loop thread.
    self.command_handler.add_loop_command_to_queue(self.get_sweep)

"""
    Function called after the device is configurated for a simple sweep to
    plot results

    Args:
        self

    Returns:
        tuple: A tuple containing the frequency values and the maximum
    amplitude values of the sweep.
"""
def get_sweep(self) -> tuple:
    # Get sweep
    self.trace_max = bb_fetch_trace_32f(self.signal_hound["handle"], self.
    __trace_len)["trace_max"]

    # Plot
    self.freqs = [self.__start + i * self.__bin_size for i in range(self.
    __trace_len)]
    self.sweep_completed.emit((self.freqs, self.trace_max))
    return self.freqs, self.trace_max

```

```
87
88  """
89      Set all the parameters to the device.
90
91      This function sets all the parameters defined in the class attributes to
        the device.
92
93      Returns:
94          None
95  """
96  def set_all_parameters(self):
97      attributes = ['center_freq', 'span', 'ref_level', 'gain', 'atten', 'rbw',
         'vbw', 'sweep_time', 'rbw_shape', 'rejection', 'detector', 'scale', '
        proc_units']
98      for attr in attributes:
99          self.set_parameter(attr, getattr(self, attr))
100
101 """
102     Updates the device parameters with a new given value inserted directly
        from the widget.
103     The new value is set on the subsequent loop thread call.
104
105     bbConfigureAcquisition() : Configures the detector and linear/log scaling
106     bbConfigureCenterSpan() : Configures the frequency range
107     bbConfigureLevel() : Configures reference level and internal attenuators
108     bbConfigureGain() : Configures internal amplifiers
109     bbConfigureSweepCoupling() : Configures RBW/VBW/Window function/sweep
        time
110     bbConfigureProcUnits() : Configures VBW processing
111
112     Args:
113         parameter_name (str): The name of the parameter to set.
114         val (float): The value of the parameter.
115
116     Returns:
117         None
118 """
119 def set_parameter(self, parameter_name: str, val: float) -> None:
120
121     def wrapper():
122         setattr(self, parameter_name, val)
```

```
123
124         # Resetting parameters when one is inconsistent with an other.
125         if(self.center_freq - (self.span)/2 < float(BB60_MIN_FREQ.value)) or
    (self.center_freq + (self.span)/2 > float(BB60_MAX_FREQ.value)):
126             if parameter_name == "center_freq":
127                 self.span = min(abs(self.center_freq - float(BB60_MIN_FREQ.
    value)), abs(self.center_freq - float(BB60_MAX_FREQ.value)))*2
128                 if(self.rbw > self.span):
129                     self.rbw = self.span
130                     self.invalid_parameter.emit("rbw")
131                 self.invalid_parameter.emit("span")
132
133             elif parameter_name == "span":
134                 if(self.center_freq - (self.span)/2 < float(BB60_MIN_FREQ.
    value)) and (self.center_freq + (self.span)/2 > float(BB60_MAX_SPAN.value)
    ):
135                     self.span = min(abs(self.center_freq - float(
    BB60_MIN_FREQ.value)), abs(self.center_freq - float(BB60_MAX_FREQ.value)))
    *2
136                     self.invalid_parameter.emit("span")
137                 else:
138                     self.center_freq = [float(BB60_MIN_FREQ.value) + self.
    span/2, float(BB60_MAX_FREQ.value) - self.span/2][np.argmax([self.
    center_freq - self.span/2 < float(BB_MIN_FREQ.value), self.center_freq +
    self.span/2 > float(BB60_MAX_FREQ.value)])]
139                     self.invalid_parameter.emit("center_freq")
140         if not self.is_connected:
141             return
142
143         if parameter_name == "span":
144             if(self.rbw > self.span):
145                 self.rbw = self.span
146                 self.invalid_parameter.emit("rbw")
147
148         if parameter_name == "rbw":
149             if(self.rbw > self.span):
150                 self.span = self.rbw
151                 self.invalid_parameter.emit("span")
152
153         bb_configure_center_span(self.signal_hound["handle"], self.
    center_freq, self.span)
```

```
154         bb_configure_ref_level(self.signal_hound["handle"], self.ref_level)
155         bb_configure_gain_atten(self.signal_hound["handle"], c_double(self.
    gain), c_double(self.atten))
156         bb_configure_sweep_coupling(self.signal_hound["handle"], self.rbw,
    self.vbw, self.sweep_time, c_double(self.rbw_shape), c_double(self.
    rejection))
157         bb_configure_acquisition(self.signal_hound["handle"], c_double(self.
    detector), c_double(self.scale))
158         bb_configure_proc_units(self.signal_hound["handle"], c_double(self.
    proc_units))
159 
160         #Once the device is configurated, it will be initialized using the
    BB_SWEEPING flag.
161         # If the device deviates in temperature, bbInitiate is called again
    to re-calibrate
162         # the device at its current operating temperature.
163         bb_initiate(self.signal_hound["handle"], BB_SWEEPING, 0)
164         query = bb_query_trace_info(self.signal_hound["handle"])
165         self.__trace_len = query["trace_len"]
166         self.__bin_size = query["bin_size"]
167         self.__start = query["start"]
168 
169     self.command_handler.add_command_to_queue(wrapper)
170 
171 """
172     Function called before saving the sweep data.
173 
174     Args:
175         file_path (str): The path of the file to be saved.
176         format (SaveThread.Format): The format of the file.
177 
178     Returns:
179         None
180 """
181 def before_saving(self, file_path: str, format: SaveThread.Format):
182     if format == SaveThread.Format.BIN:
183         with open(file_path, 'ab') as file:
184             file.write('Logging format v1\n'.encode())
185             file.write(('[Frequencies [Hz], Amplitudes [dBm]]\n').encode())
186     else:
187         with open(file_path, 'a') as file:
```

```
            file.write('Logging format v1\n')
            file.write('[Frequencies [Hz], Amplitude [dBm]]\n')

"""
    Function for saving the sweep data to a file.

    Args:
        file_path (str): The path of the file to be saved.
        format (SaveThread.Format): The format of the file.

    Returns:
        None
"""
def saving_function(self, file_path: str, format: SaveThread.Format) -> None:
    if format == SaveThread.Format.BIN:
        with open(file_path, 'ab') as file:
            file.write(str([self.freqs, list(self.trace_max)]).encode())
            file.write("\n".encode())
    else:
        with open(file_path, 'a') as file:
            file.write(str([self.freqs, list(self.trace_max)]))
            file.write("\n")

#------------------------------DEVICE-------------------------------------

"""
   Connects the ManagerSignalHoundBB60C to the device.

   Args:
    self The ManagerSignalHoundBB60C class instance pointer.
"""
def connect(self):
    if self.is_connected:
        return
    try:
        self.signal_hound = bb_open_device()
    except:
        self.signal_hound = None
        return
    self.is_connected = True

```

```
"""
    Disconnects the ManagerSignalHoundBB60C from the device.

    Args:
        self The ManagerSignalHoundBB60C class instance pointer.
"""
def disconnect(self):
    if not self.is_connected:
        return
    bb_close_device(self.signal_hound["handle"])
    self.is_connected = False

"""
    Starts the loopThread

    Args:
        self The ManagerSignalHoundBB60C class instance pointer.
"""
def start(self):
    if not self.is_connected:
        self.connect()
        pass
    self.command_handler.start_loop()

"""
    Stops the loopThread

    Args:
        self The ManagerSignalHoundBB60C class instance pointer.
"""
def stop(self):
    if self.command_handler.is_running():
        self.command_handler.stop_loop()
    self.disconnect()

#------------------------------EXAMPLE-----------------------------------

"""
    Example of use of the ManagerSignalHoundBB60C class.
"""

```

```
270 if __name__ == "__main__":
271     from PyQt6.QtCore import QCoreApplication
272     app = QCoreApplication([])
273     manager = ManagerSignalHoundBB60C()
274     manager.sweep_completed.connect(lambda x: print(x))
275     manager.start()
276     #manager.set_parameters()
277     #tuplex, tupley = manager.get_sweep()
278     #plt.plot(tuplex, tupley)
279     app.exec()
```

A.1.2 widget_signal_hound_bb60c.py

```
1
2 '''
3     Signal Hound Spectrum Analyzer bb60 series controller widget
4
5     @author: Federico Muscara'
6 '''
7
8 import PyQt6  # leave! otherwise pyqtgraph guesses PyQt5 is the backend
9 from PyQt6 import QtCore
10 import sys
11 import os.path
12 sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), os.
      path.pardir)))
13
14 from PyQt6.QtWidgets import QWidget
15 from devices.bb_api import *
16 import pyqtgraph as pg
17 from PyQt6.QtWidgets import QRadioButton, QHBoxLayout, QVBoxLayout,
      QApplication
18 from managers.manager_signal_hound_bb60c import ManagerSignalHoundBB60C
19 from PyQt6.QtWidgets import QWidget, QVBoxLayout
20 import logging
21 from utils.my_qlabel_qlineedit import myQLabelQLineEdit
22
23
24 class WidgetSignalHoundBB60C(QWidget):
25
26     """
```

```
        Signal Hound Spectrum Analyzer bb60 series controller widget

        Args:
            **kwargs: Additional arguments to initialize the
    ManagerSignalHoundBB60C.

        Returns:
            None
    """
    def __init__(self, **kwargs):
        super().__init__()

        self.manager = ManagerSignalHoundBB60C(**kwargs)
        self.manager.sweep_completed.connect(self._on_newplotdata)

        self._init_gui()

        self.manager.invalid_parameter.connect(self._on_invalid_parameter)

    """
        Initialize the GUI for the widget.
    """
    def _init_gui(self):

        self.parameters_QLineEdits = [
            myQLabelQLineEdit('Center Frequency [kHz]', input_type=float, min
    =0, max=BB_MAX_FREQ.value * 1e-3, fixed_width=130),
            myQLabelQLineEdit('Span [kHz]', input_type=float, min=20, max=
    BB_MAX_SPAN.value * 1e-3, fixed_width=130),
            myQLabelQLineEdit('Ref level [dBm]', input_type=float, min=-130,
    max=130, fixed_width=130),
            myQLabelQLineEdit('Gain [dB]', input_type=float, min=BB_AUTO_GAIN
    , max=BB_MAX_GAIN, fixed_width=130),
            myQLabelQLineEdit('Atten [dB]', input_type=float, min=
    BB_AUTO_ATTEN, max=BB_MAX_ATTEN, fixed_width=130),
            myQLabelQLineEdit('RBW [kHz]', input_type=float, min=BB_MIN_RBW.
    value * 1e-3, max=BB_MAX_RBW.value * 1e-3, fixed_width=130),
            myQLabelQLineEdit('VBW [kHz]', input_type=float, min=0, max=
    BB_MAX_RBW.value * 1e-3, fixed_width=130),
            myQLabelQLineEdit('Sweep time [s]', input_type=float, min=
    BB_MIN_SWEEP_TIME.value, max=BB_MAX_SWEEP_TIME.value, fixed_width=130),
```

```
59          myQLabelQLineEdit('RBW shape', input_type=float, min=
    BB_RBW_SHAPE_NUTTALL, max=BB_RBW_SHAPE_CISPR, fixed_width=130),
60          myQLabelQLineEdit('Rejection', input_type=float, min=
    BB_NO_SPUR_REJECT, max=BB_SPUR_REJECT, fixed_width=130),
61          myQLabelQLineEdit('Detector', input_type=float, min=
    BB_MIN_AND_MAX, max=BB_AVERAGE, fixed_width=130),
62          myQLabelQLineEdit('Scale', input_type=float, min=BB_LOG_SCALE,
    max=BB_LIN_FULL_SCALE, fixed_width=130),
63          myQLabelQLineEdit('Proc units', input_type=float, min=BB_LOG, max
    =BB_SAMPLE, fixed_width=130)
64      ]
65
66      self._plot = pg.PlotWidget(self, labels={'bottom': 'Frequency [Hz]',
    'left': 'Amplitude [dBm]'})
67      self._plot.setMinimumWidth(800)
68      self._curve = self._plot.plot()
69
70      self._plot.showGrid(x=True, y=True)
71
72      self._connection_QRadioButton = QRadioButton('connect')
73      self._connection_QRadioButton.setAutoExclusive(False)
74      self._connection_QRadioButton.clicked.connect(self.
    _on_connection_QRadioButton_clicked)
75      self._connection_QRadioButton.setStyleSheet("QRadioButton:unchecked{
    margin:0; color: red;} QRadioButton:checked{ margin:0; color: green;}")
76
77      self.__Vlayouts = [QVBoxLayout(), QVBoxLayout()]
78      self.__Vlayouts[1].setAlignment(QtCore.Qt.AlignmentFlag.AlignTop)
79      self.__Vlayouts[1].setAlignment(QtCore.Qt.AlignmentFlag.AlignRight)
    # TODO Allineamento box di destra
80      self.__Hlayout = QHBoxLayout()
81
82      self.setLayout(self.__Hlayout)
83      self.__Hlayout.addLayout(self.__Vlayouts[0])
84      self.__Hlayout.addLayout(self.__Vlayouts[1])
85      self.__Vlayouts[0].addWidget(self._connection_QRadioButton)
86      self.__Vlayouts[0].addWidget(self._plot)
87
88      # Dictionary of sweep parameters and their corresponding
    set_parameters functions
89      self.parameters_set_functions = {
```

```
            'Center Frequency [kHz]': lambda val: self.manager.set_parameter('center_freq', val * 1e3),
            'Span [kHz]': lambda val: self.manager.set_parameter('span', val * 1e3),
            'Ref level [dBm]': lambda val: self.manager.set_parameter('ref_level', val),
            'Gain [dB]': lambda val: self.manager.set_parameter('gain', val),
            'Atten [dB]': lambda val: self.manager.set_parameter('atten', val),
            'RBW [kHz]': lambda val: self.manager.set_parameter('rbw', val * 1e3),
            'VBW [kHz]': lambda val: self.manager.set_parameter('vbw', val * 1e3),
            'Sweep time [s]': lambda val: self.manager.set_parameter('sweep_time', val),
            'RBW shape': lambda val: self.manager.set_parameter('rbw_shape', val),
            'Rejection': lambda val: self.manager.set_parameter('rejection', val),
            'Detector': lambda val: self.manager.set_parameter('detector', val),
            'Scale': lambda val: self.manager.set_parameter('scale', val),
            'Proc units': lambda val: self.manager.set_parameter('proc_units', val)
        }

        # Dictionary of sweep parameters and their corresponding attribute names
        self.names_dict = {
            'Center Frequency [kHz]': 'center_freq',
            'Span [kHz]': 'span',
            'Ref level [dBm]': 'ref_level',
            'Gain [dB]': 'gain',
            'Atten [dB]': 'atten',
            'RBW [kHz]': 'rbw',
            'VBW [kHz]': 'vbw',
            'Sweep time [s]': 'sweep_time',
            'RBW shape': 'rbw_shape',
            'Rejection': 'rejection',
            'Detector': 'detector',
            'Scale': 'scale',
```

```
            'Proc units': 'proc_units'
        }

        # To represent "frequency parameters" in kHz
        for lineedit in self.parameters_QLineEdits:
            if 'kHz' in lineedit._title_QLabel.text():
                lineedit.set_value(1e-3 * getattr(self.manager, self.
names_dict[lineedit._title_QLabel.text()]))
            else:
                lineedit.set_value(getattr(self.manager, self.names_dict[
lineedit._title_QLabel.text()]))
            lineedit.value_changed.emit(lineedit.get_value())
            self.__Vlayouts[1].addWidget(lineedit)

            # Connect the value_changed signal to the corresponding
set_parameter function
            if lineedit._title_QLabel.text() in list(self.
parameters_set_functions.keys()):
                lineedit.value_changed.connect(self.parameters_set_functions[
lineedit._title_QLabel.text()])

    """
        Callback function when an invalid parameter is encountered.

        Args:
            parameter (str): The name of the invalid parameter.

        Returns:
            None
    """
    def _on_invalid_parameter(self, parameter: str):
        for lineedit in self.parameters_QLineEdits:
            if parameter == self.names_dict[lineedit._title_QLabel.text()]:
                lineedit.set_value(1e-3 * getattr(self.manager, parameter))

    """
        Callback function when new plot data is available.

        Args:
            data (tuple): A tuple containing the spectrum data (spectrum_x,
spectrum_y).
```

```
154
155         Returns:
156             None
157     """
158     def _on_newplotdata(self, data: tuple) -> None:
159         spectrum_x, spectrum_y = data
160         self._curve.setData(spectrum_x, spectrum_y, pen='r', name='Spectrum',
     _callSync='off')
161
162     """
163         Callback function when the connection QRadioButton is clicked.
164     """
165     def _on_connection_QRadioButton_clicked(self) -> None:
166         if self._connection_QRadioButton.isChecked():
167             self._connection_QRadioButton.setText('connecting')
168             QApplication.processEvents()
169             self.manager.connect()
170             if self.manager.is_connected:
171                 self._connection_QRadioButton.setText('connected')
172                 self.manager.start()
173                 self.parameters_QLineEdits[0].value_changed.emit(self.
    parameters_QLineEdits[0].get_value())
174             else:
175                 logging.warning('No connection established')
176                 self._connection_QRadioButton.setText('connect')
177                 self._connection_QRadioButton.setChecked(False)
178         else:
179             self.manager.stop()
180             self._connection_QRadioButton.setText('connect')
181
182 # -----------------EXAMPLE-----------------
183
184 """
185     Example of use of the WidgetSignalHoundBB60C class.
186
187 """
188 if __name__ == '__main__':
189     from utils.super_window import SuperWindow
190
191     window = SuperWindow(skin='black', title='Widget Signal Hound BB60C')
192
```

```
193     widget = WidgetSignalHoundBB60C()
194 
195     # To solve USB timeout error:
196     # widget.manager.connect()
197     # widget.manager.disconnect()
198 
199     window.add_vertically(widget)
200 
201     window.show()
```

A.1.3 app_signal_hound_bb60c.py

```
1 
2 """
3     Example of app development for Signal Hound BB60C series spectrum
      analyzer.
4 
5     @author: Federico Muscara'
6 """
7 import PyQt6  # leave! otherwise pyqtgraph guesses PyQt5 is the backend
8 from PyQt6.QtWidgets import *
9 from src.utils.super_window import SuperWindow
10 from src.widgets.widget_signal_hound_bb60c import WidgetSignalHoundBB60C
11 from src.widgets.widget_export import WidgetExport
12 
13 window = SuperWindow(skin='black', title='Signal Hound example', min_size
      =(800, 0))
14 
15 signal_hound = WidgetSignalHoundBB60C()
16 export = WidgetExport(log_time_s=20)
17 
18 export.add_saving_function(signal_hound.manager.saving_function, start_func=
      signal_hound.manager.before_saving, label='Signal Hound')
19 
20 window.add_vertically(signal_hound)
21 window.add_horizontally(export)
22 
23 window.show()
```

A.2 Funzioni delay_finder

A.2.1 delay_finder

```
def delay_finder(tags1, tags2, delay_max, bin_size=None, peak_width=1e2) ->
    tuple:
  delay_max = float(delay_max)
  SAMPLING_MAX = 3e6 #3micros
  min_acq_time = min(tags1[-1], tags2[-1])
  time_const = 2e-5
  period_of_sampling = int(max(int(peak_width), min((np.sqrt(min_acq_time*
    delay_max)*time_const), SAMPLING_MAX)))

  if bin_size is None:
    bin_size = period_of_sampling // 10000
  PADDING_PARAM = int(delay_max // period_of_sampling +1)
  #Bins creation
  bins1 = int(tags1[-1] // period_of_sampling) +1
  bins2 = int(tags2[-1] // period_of_sampling) +1
  #Quotients
  quotients1 = np.array(tags1 // period_of_sampling, dtype='int64')
  quotients2 = np.array(tags2 // period_of_sampling, dtype='int64')
  #Detections
  detections1 = np.zeros(bins1, dtype='uint16')
  detections2 = np.zeros(bins2, dtype='uint16')
  detections1[quotients1] = 1
  detections2[quotients2] = 1

  #bin:tag dicts
  bin_tag_dict1 = dict(zip(quotients1, tags1))
  bin_tag_dict2 = dict(zip(quotients2, tags2))
  #Cross-Correlation
  padded_detections2 = np.pad(detections2, (PADDING_PARAM, PADDING_PARAM))
  cross_corr_result = np.correlate(padded_detections2, detections1)

  SNR = SNR_corr(cross_corr_result)
  if SNR == 0:
      return (period_of_sampling, None, None, cross_corr_result, SNR)

  cross_corr_argmax = np.argmax(cross_corr_result)

```

```
delay_in_sampling_period = PADDING_PARAM - cross_corr_argmax

if delay_in_sampling_period>0:
  coincidences = np.array(detections1[0:min(bins1, bins2)][
  delay_in_sampling_period:] * (detections2[0:min(bins1, bins2)][0:-
  delay_in_sampling_period]))
elif delay_in_sampling_period<0:
  coincidences = np.array(detections2[0:min(bins1, bins2)][-
  delay_in_sampling_period:] * (detections1[0:min(bins1, bins2)][0:
  delay_in_sampling_period]))
else:
  coincidences = np.array(detections2[0:min(bins1, bins2)] * detections1[0:
  min(bins1, bins2)])

coincidences_indexes = np.where(coincidences == 1)[0]
tag_of_coinc1 = np.array([bin_tag_dict1.get(tag_index + max(0,
  delay_in_sampling_period)) for tag_index in coincidences_indexes])
tag_of_coinc2 = np.array([bin_tag_dict2.get(tag_index - min(0,
  delay_in_sampling_period)) for tag_index in coincidences_indexes])
diff_of_coinc_tags = tag_of_coinc1 - tag_of_coinc2

#Histogram
delay = (delay_in_sampling_period)*period_of_sampling
span = 2*period_of_sampling
bins = np.arange(delay-span//2, delay+span//2, bin_size)

hist = np.histogram(diff_of_coinc_tags, bins=bins)
argmax_bin = np.argmax(hist[0])
argmax_hist_value = bins[argmax_bin]

#Gaussian fit
if peak_width//bin_size>1:
  zoomed_span = int(6*peak_width//bin_size) #espresso in bins
  zoomed_bins = hist[1][argmax_bin-zoomed_span//2 : argmax_bin+zoomed_span
  //2]
  zoomed_hist = hist[0][argmax_bin-zoomed_span//2 : argmax_bin+zoomed_span
  //2]
  my_guess = gaussian_guess(zoomed_bins, zoomed_hist)
  popt, pcov = curve_fit(gaussian, zoomed_bins, zoomed_hist, p0=my_guess)
  argmax_gauss_value = popt[1]
else:
```

```
    argmax_gauss_value = None

  ret_tuple = (period_of_sampling, argmax_hist_value, argmax_gauss_value,
    cross_corr_result, SNR)
  return ret_tuple
```

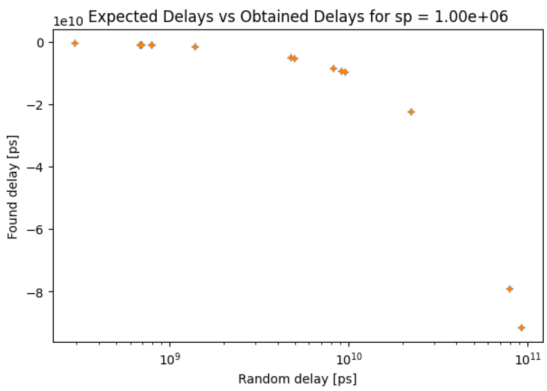
A.2.2 DELAY_FINDER_FFT

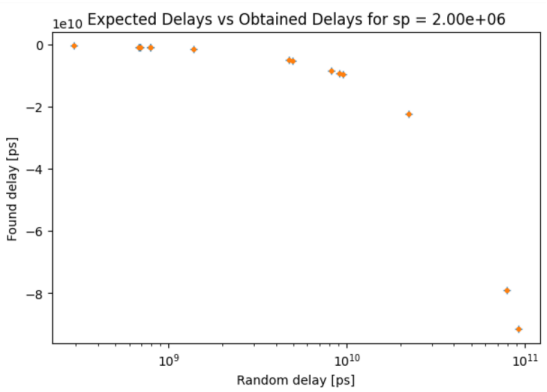
```
def delay_finder_fft(tags1, tags2, delay_max=100e9, peak_width=1e2) -> tuple:
    last_tag = max(tags1[-1],tags2[-1])
    MAX_BINS = 1e7
    SAMPLING_MAX = 3e6
    period_of_sampling = last_tag//MAX_BINS
    if period_of_sampling>=SAMPLING_MAX:
      return delay_finder(tags1, tags2, delay_max, peak_width)

    #Bins creation
    bins1 = int(last_tag // period_of_sampling) +1
    bins2 = int(last_tag // period_of_sampling) +1
    #Quotients
    quotients1 = np.array(tags1 // period_of_sampling, dtype='int64')
    quotients2 = np.array(tags2 // period_of_sampling, dtype='int64')
    #Detections
    detections1 = np.zeros(bins1, dtype='uint16')
    detections2 = np.zeros(bins2, dtype='uint16')
    detections1[quotients1] = 1
    detections2[quotients2] = 1

    #bin:tag dicts
    bin_tag_dict1 = dict(zip(quotients1, tags1))
    bin_tag_dict2 = dict(zip(quotients2, tags2))

    #Cross-Correlation
    cross_corr_result = fft_correlate(detections1, detections2)

    SNR = SNR_corr(cross_corr_result)
    if SNR == 0:
      return (period_of_sampling, None, None, cross_corr_result, SNR)

    cross_corr_argmax = np.argmax(cross_corr_result) - (len(cross_corr_result
    )//2)
```

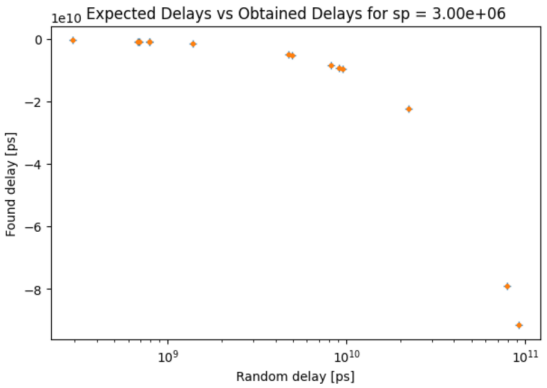
```
33
34      delay_in_sampling_period = - cross_corr_argmax
35
36      if delay_in_sampling_period>0:
37          coincidences = np.array(detections1[0:min(bins1, bins2)][
        delay_in_sampling_period:] * (detections2[0:min(bins1, bins2)][0:-
        delay_in_sampling_period]))
38      elif delay_in_sampling_period<0:
39          coincidences = np.array(detections2[0:min(bins1, bins2)][-
        delay_in_sampling_period:] * (detections1[0:min(bins1, bins2)][0:
        delay_in_sampling_period]))
40      else:
41          coincidences = np.array(detections2[0:min(bins1, bins2)] *
        detections1[0:min(bins1, bins2)])
42
43      coincidences_indexes = np.where(coincidences == 1)[0]
44      tag_of_coinc1 = np.array([bin_tag_dict1.get(tag_index + max(0,
        delay_in_sampling_period)) for tag_index in coincidences_indexes])
45      tag_of_coinc2 = np.array([bin_tag_dict2.get(tag_index - min(0,
        delay_in_sampling_period)) for tag_index in coincidences_indexes])
46      diff_of_coinc_tags = tag_of_coinc1 - tag_of_coinc2
47
48      #Histogram
49      delay = (delay_in_sampling_period)*period_of_sampling
50      span = 2*period_of_sampling
51      bin_size = span//20000
52      bins = np.arange(delay-span//2, delay+span//2, bin_size)
53
54      hist = np.histogram(diff_of_coinc_tags, bins=bins)
55      argmax_bin = np.argmax(hist[0])
56      argmax_hist_value = (bins[argmax_bin] + bins[argmax_bin+1])/2
57
58      #Gaussian fit
59      if peak_width//bin_size>1:
60          zoomed_span = int(6*peak_width//bin_size) #espresso in bins
61          zoomed_bins = hist[1][argmax_bin-zoomed_span//2 : argmax_bin+
        zoomed_span//2]
62          zoomed_hist = hist[0][argmax_bin-zoomed_span//2 : argmax_bin+
        zoomed_span//2]
63          my_guess = gaussian_guess(zoomed_bins, zoomed_hist)
64          popt, pcov = curve_fit(gaussian, zoomed_bins, zoomed_hist, p0=
```

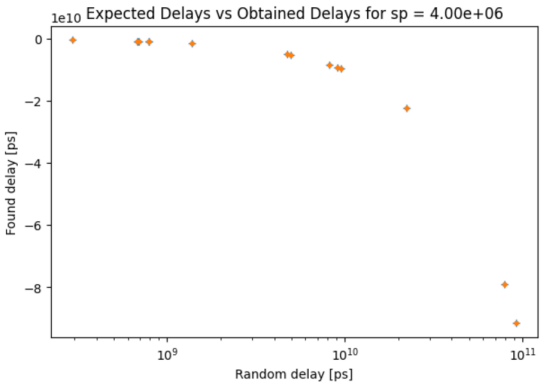
```
    my_guess)
65          argmax_gauss_value = popt[1]
66      else:
67          argmax_gauss_value = None
68
69      ret_tuple = (period_of_sampling, argmax_hist_value, argmax_gauss_value,
    cross_corr_result, SNR)
70      return ret_tuple
```

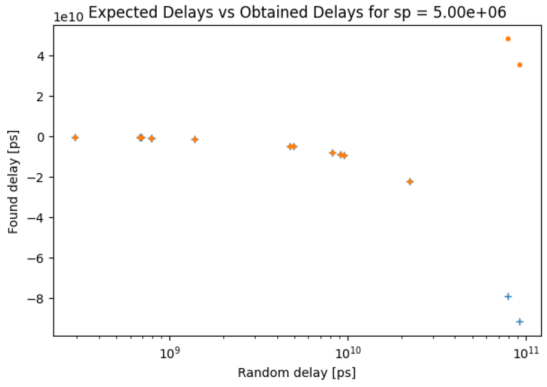
A.2.3 Precisione di *delay_finder*

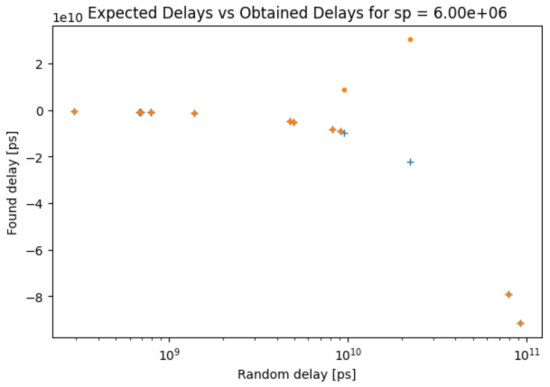












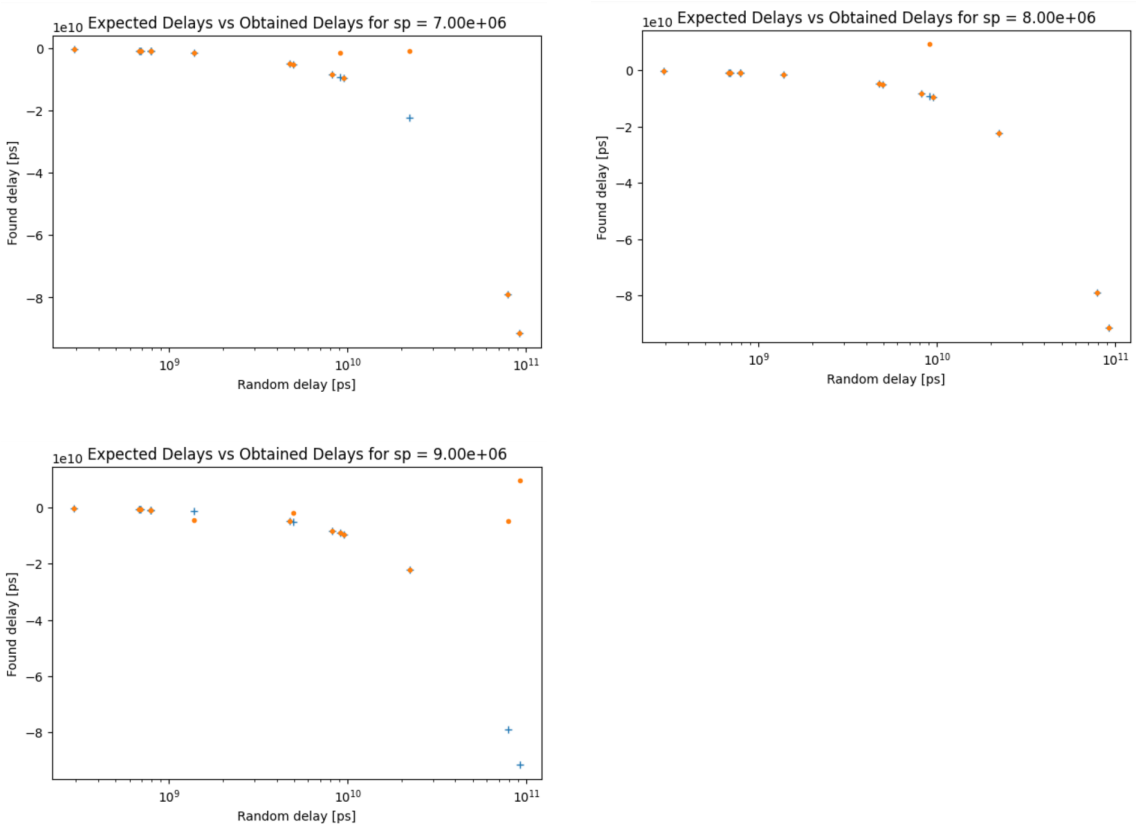
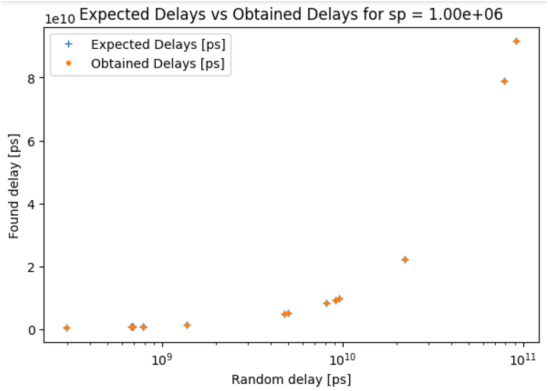
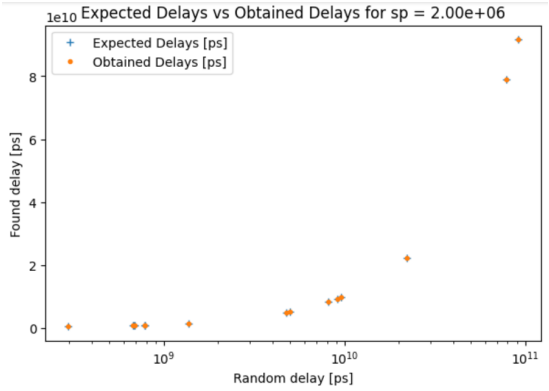
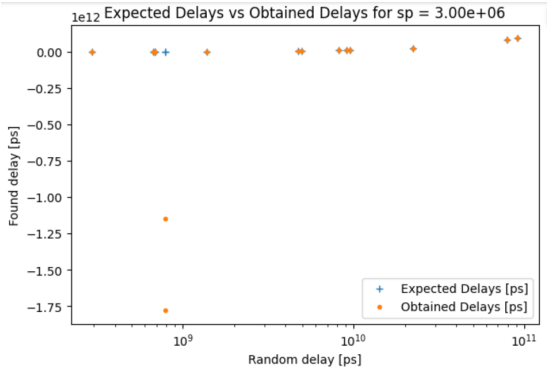


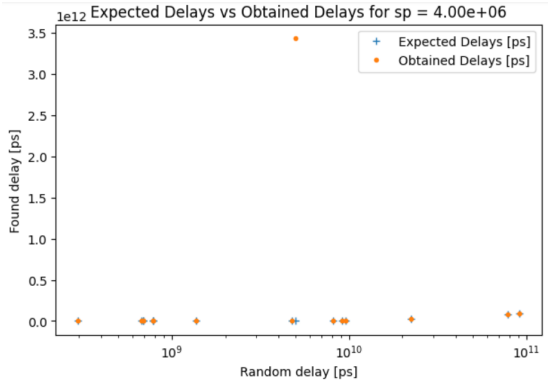
Figura A.2: Grafici che illustrano la precisione del risultato fornito dalla funzione *delay_finder_fft* (*obtained delay*, in giallo) rispetto a quello ”realmente” corretto (*expected delay*, in blu). Ciascun riquadro si diversifica da quelli adiacenti per il periodo di campionamento utilizzato dalla funzione.

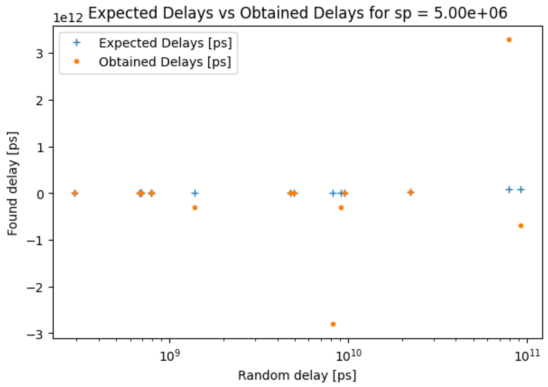
A.2.4 Precisione di *delay_finder_fft*

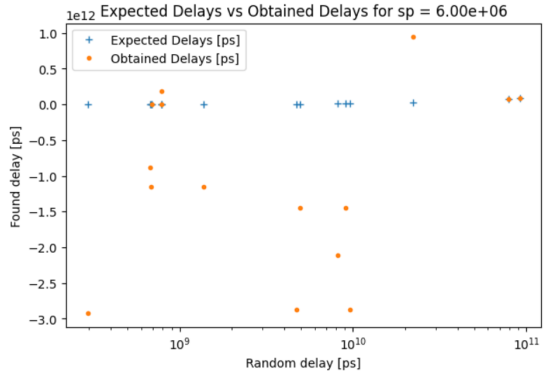












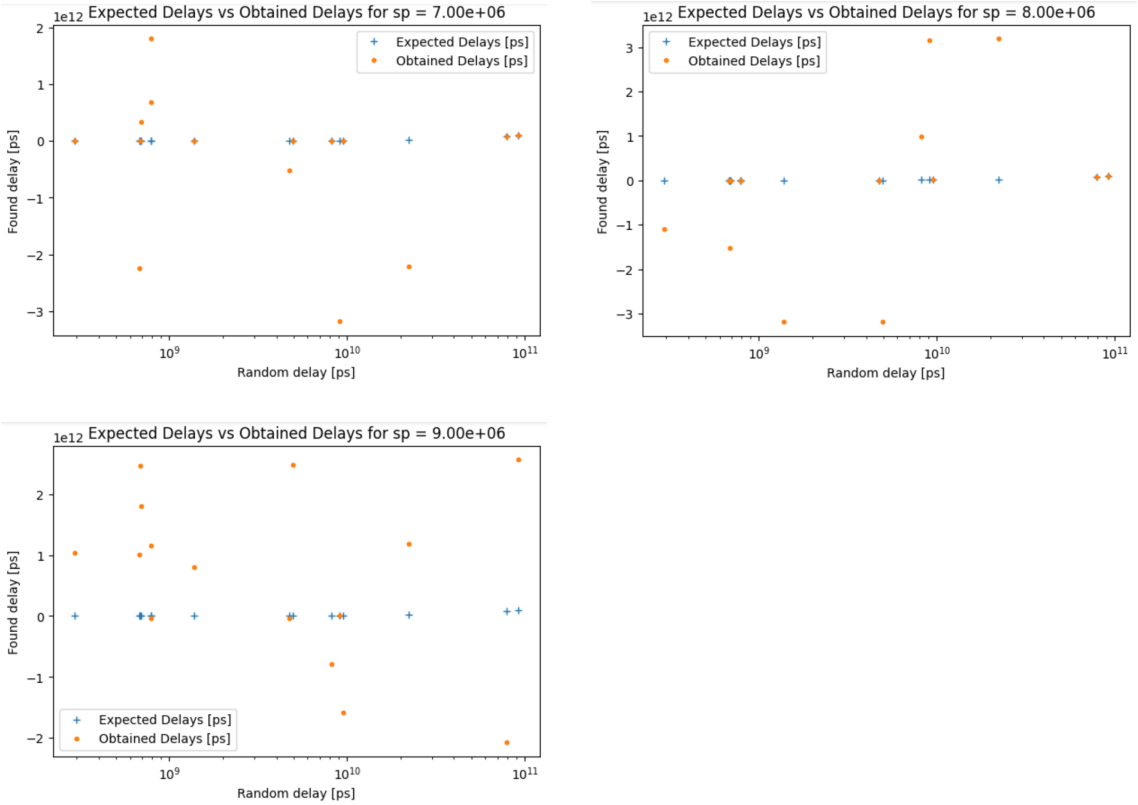


Figura A.4: Grafici che illustrano la precisione del risultato fornito dalla funzione *delay_finder_fft* (*obtained delay*, in giallo) rispetto a quello ”realmente” corretto (*expected delay*, in blu). Ciascun riquadro si diversifica da quelli adiacenti per il periodo di campionamento utilizzato dalla funzione.

Bibliografia

[1] R. L. Richard Feynman, Matthew Sands, *The Feynman Lectures on Physics, Vol. I: The New Millennium Edition: Mainly Mechanics, Radiation, and Heat*. Basic Books, 2011.

[2] M. L. Bellac, *A Short Introduction to Quantum Information and Quantum Computation*. Cambridge University Press, 2006.

[3] L. Mandel and E. Wolf, *Optical Coherence and Quantum Optics*. Cambridge University Press, Sep. 1995. [Online]. Available: https://doi.org/10.1017/cbo9781139644105

[4] R. Loudon, *The Quantum Theory of Light*. OUP Oxford, 2000. [Online]. Available: https://books.google.it/books?id=AEkfajgqldoC

[5] C. Gerry and P. Knight, *Introductory Quantum Optics*. Cambridge University Press, Oct. 2004. [Online]. Available: https://doi.org/10.1017/cbo9780511791239

[6] J. Garrison and R. Chiao, *Quantum Optics*. Oxford University Press, 06 2008. [Online]. Available: https://doi.org/10.1093/acprof:oso/9780198508861.001.0001

[7] D. Bouwmeester, A. K. Ekert, and A. Zeilinger, *The Physics of Quantum Information: Quantum Cryptography, Quantum Teleportation, Quantum Computation*. Springer Science & Business Media, Mar. 2013, google-Books-ID: Qu7wCAAAQBAJ.

[8] A. Einstein, B. Podolsky, and N. Rosen, "Can quantum-mechanical description of physical reality be considered complete?" *Phys. Rev.*, vol. 47, pp. 777–780, 5 1935. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.47.777

[9] N. Bohr, "Can quantum-mechanical description of physical reality be considered complete?" *Phys. Rev.*, vol. 48, pp. 696–702, 10 1935. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.48.696

[10] J. S. Bell, "On the Einstein Podolsky Rosen paradox," *Physics Physique Fizika*, vol. 1, pp. 195–200, 11 1964. [Online]. Available: https://link.aps.org/doi/10.1103/PhysicsPhysiqueFizika.1.195

[11] J. F. Clauser, M. A. Horne, A. Shimony, and R. A. Holt, "Proposed experiment to test local hidden-variable theories," *Phys. Rev. Lett.*, vol. 23, pp. 880–884, 10 1969. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.23.880

[12] J.-Å. Larsson, "Loopholes in Bell inequality tests of local realism," *J. Phys. A: Math. Theor.*, vol. 47, p. 424003, 10 2014. [Online]. Available: https://iopscience.iop.org/article/10.1088/1751-8113/47/42/424003

[13] B. Hensen, H. Bernien, A. E. Dréau, A. Reiserer, N. Kalb, M. S. Blok, J. Ruitenberg, R. F. L. Vermeulen, R. N. Schouten, C. Abellán, W. Amaya, V. Pruneri, M. W. Mitchell, M. Markham, D. J. Twitchen, D. Elkouss, S. Wehner, T. H. Taminiau, and R. Hanson, "Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres," *Nature*, vol. 526, no. 7575, pp. 682–686, 10 2015. [Online]. Available: https://doi.org/10.1038/nature15759

[14] M. Giustina, M. A. M. Versteegh, S. Wengerowsky, J. Handsteiner, A. Hochrainer, K. Phelan, F. Steinlechner, J. Kofler, J.-A. Larsson, C. Abellán, W. Amaya, V. Pruneri, M. W. Mitchell, J. Beyer, T. Gerrits, A. E. Lita, L. K. Shalm, S. W. Nam, T. Scheidl, R. Ursin, B. Wittmann, and A. Zeilinger, "Significant-loophole-free test of Bell's theorem with entangled photons," *Phys. Rev. Lett.*, vol. 115, p. 250401, 12 2015. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.115.250401

[15] L. K. Shalm, E. Meyer-Scott, B. G. Christensen, P. Bierhorst, M. A. Wayne, M. J. Stevens, T. Gerrits, S. Glancy, D. R. Hamel, M. S. Allman, K. J. Coakley, S. D. Dyer, C. Hodge, A. E. Lita, V. B. Verma, C. Lambrocco, E. Tortorici, A. L. Migdall, Y. Zhang, D. R. Kumor, W. H. Farr, F. Marsili, M. D. Shaw, J. A. Stern, C. Abellán, W. Amaya, V. Pruneri, T. Jennewein, M. W. Mitchell, P. G. Kwiat, J. C. Bienfang, R. P. Mirin, E. Knill, and S. W. Nam, "Strong loophole-free test of local realism," *Phys. Rev. Lett.*, vol. 115, p. 250402, 12 2015. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.115.250402

[16] W. Rosenfeld, D. Burchardt, R. Garthoff, K. Redeker, N. Ortegel, M. Rau, and H. Weinfurter, "Event-ready Bell test using entangled atoms simultaneously closing detection and locality loopholes," *Phys. Rev. Lett.*, vol. 119, p. 010402, 7 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.119.010402

[17] D. P. Nadlinger, P. Drmota, B. C. Nichol, G. Araneda, D. Main, R. Srinivas, D. M. Lucas, C. J. Ballance, K. Ivanov, E. Y.-Z. Tan, P. Sekatski, R. L. Urbanke, R. Renner, N. Sangouard, and J.-D. Bancal, "Experimental quantum key distribution certified by Bell's theorem," *Nature*, vol. 607, no. 7920, pp. 682–686, Jul. 2022. [Online]. Available: https://doi.org/10.1038/s41586-022-04941-5

[18] W. Zhang, T. van Leent, K. Redeker, R. Garthoff, R. Schwonnek, F. Fertig, S. Eppelt, W. Rosenfeld, V. Scarani, C. C.-W. Lim, and H. Weinfurter, "A device-independent quantum key distribution system for distant users," *Nature*, vol. 607, no. 7920, pp. 687–691, Jul. 2022. [Online]. Available: https://doi.org/10.1038/s41586-022-04891-y

[19] W.-Z. Liu, Y.-Z. Zhang, Y.-Z. Zhen, M.-H. Li, Y. Liu, J. Fan, F. Xu, Q. Zhang, and J.-W. Pan, "Toward a photonic demonstration of device-independent quantum key distribution," *Phys. Rev. Lett.*, vol. 129, p. 050502, 7 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.129.050502

[20] B. S. Cirel'son, "Quantum generalizations of bell's inequality," *Letters in Mathematical Physics*, vol. 4, no. 2, pp. 93–100, 1980.

[21] J. Jogenfors and J.-Å. Larsson, "Energy-time entanglement, elements of reality, and local realism," *J. Phys. A: Math. Theor.*, vol. 47, no. 42, p. 424032, Oct. 2014. [Online]. Available: https://iopscience.iop.org/article/10.1088/1751-8113/47/42/424032

[22] F. Vedovato, C. Agnesi, M. Tomasin, M. Avesani, J.-Å. Larsson, G. Vallone, and P. Villoresi, "Postselection-loophole-free Bell violation with genuine time-bin entanglement," *Phys. Rev. Lett.*, vol. 121, no. 19, p. 190401, Nov. 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.121.190401

[23] A. Vretblad, *Fourier Analisys and its application*. Springer, 2003.

[24] R. M. Gray and J. W. Goodman, *Fourier Transforms: An Introduction for Engineers*. McGraw-Hill, 1986.