Università Politecnica delle Marche

Wireless Communications and Navigation Systems

Stima e correzione dell'IQ imbalance

Matteo Orlandini e Jacopo Pagliuca

Prof. Ennio Gambi
Prof. Adelmo de Santis
Dott. Gianluca Ciattaglia

24 febbraio 2021

Indice

1 Introduzione

Lo scopo di questo progetto è quello di stimare e correggere l'IQ imbalance in una modulazione QPSK usando Simulink e Matlab. La phase-shift keying (PSK) è una modulazione digitale che trasmette i dati cambiando la fase della portante. La quadrature phase-shift keying (QPSK) utilizza quattro punti equidistanti attorno a un cerchio sul diagramma della costellazione. Con quattro fasi, la QPSK può codificare due bit per simbolo, come mostrato nel diagramma 1, e può usare la codifica Gray per ridurre al minimo il bit error rate(BER).

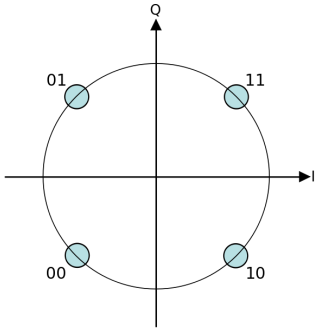


Figura 1: Diagramma della costellazione QPSK con codifica Gray. Ogni simbolo adiacente differisce solo di un bit. [9]

Il modello di un modulatore QPSK è mostrato in figura 2. Inizialmente i bit a rate R bit/s vengono codificati NRZ, successivamente divisi in due flussi a rate $R/2$ bit/s e filtrati con un filtro modellatore dell'impulso, come ad esempio un filtro a coseno rialzato. I due flussi, in fase (I) e in quadratura (Q), sono modulati a frequenza f_c rispettivamente da un $\cos(2\pi f_c t)$ e da un $\sin(2\pi f_c t)$. Il segnale QPSK $s(t)$ è definito come

$$s(t) = A\cos\left(2\pi f_c t + (2n+1)\frac{\pi}{4}\right) \qquad n = 0, 1, 2, 3. \tag{1}$$

Questo segnale può essere pensato come un segnale complesso e, dunque, scomposto in una parte in fase e una parte in quadratura $s(t) = s_I(t) + i s_Q(t)$. Dove $s_I(t)$ e $s_Q(t)$ sono definite nel seguente modo:

$$s_I(t) = \pm A\cos\left(2\pi f_c t + \frac{\pi}{4}\right) \tag{2}$$

$$s_Q(t) = \pm A\sin\left(2\pi f_c t + \frac{\pi}{4}\right). \tag{3}$$

Come mostrato dalle equazioni (2) e (3), in un modulatore I/Q ideale, l'ampiezza dei segnali in fase e quadratura è la stessa. Una differenza fra

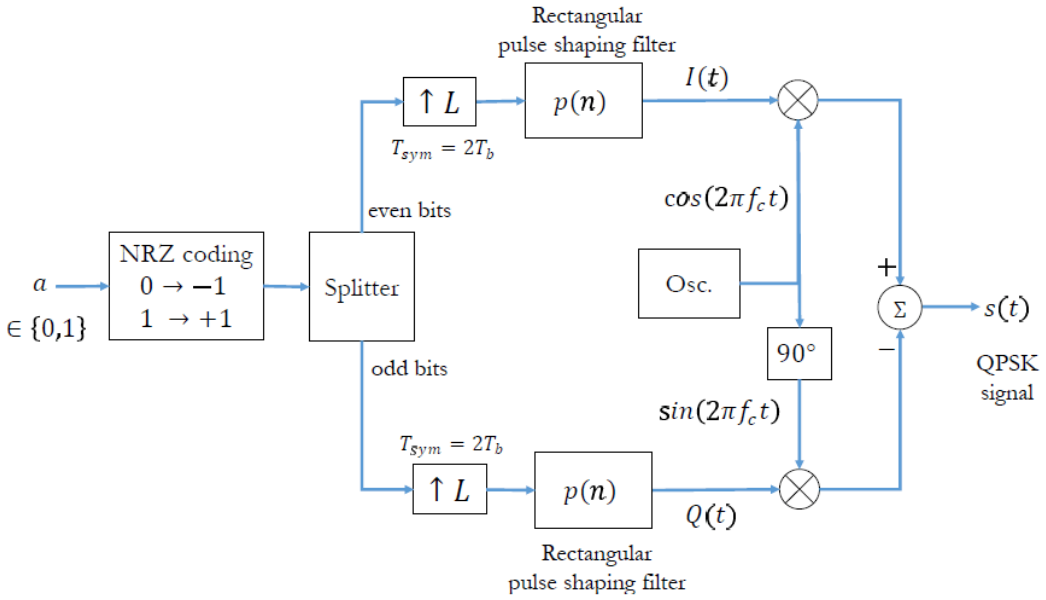


Figura 2: Modello del modulatore QPSK. [7]

l'ampiezza dei segnali I e Q è detta *IQ imbalance*. Applicandola volontariamente da trasmettitore, un valore positivo di imbalance indica che l'ampiezza del segnale in fase è maggiore di quello in quadratura. In presenza di uno sbilanciamento G in decibel, i segnali (2) e (3) diventano:

$$s_I(t) = g_I \cdot \mathrm{Re}\{s(t)\} \tag{4}$$

$$s_Q(t) = g_Q \cdot \mathrm{Im}\{s(t)\}\,. \tag{5}$$

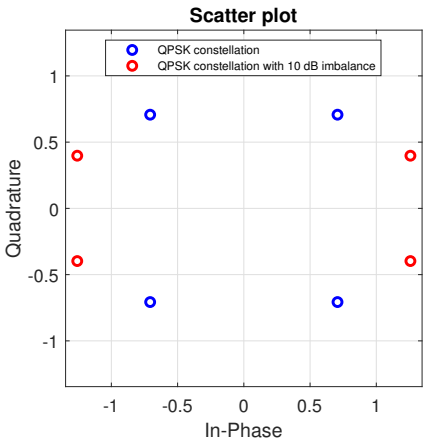
dove $g_I = 10^{0.5\frac{G}{20}}$ e $g_Q = 10^{-0.5\frac{G}{20}}$. Il risultato di un imbalance di ampiezza di 10 dB è mostrato in figura 3a.

Le due portanti possono anche differire nella fase, in questo caso si parla di *errore di quadratura*. Normalmente la differenza di fase fra la portante I e Q è di 90°. Deviazioni di questo valore sono espresse come errore di quadratura. In un trasmettitore ideale esso è pari a 0°, un valore positivo indica invece una differenza di fase maggiore di 90°. Il risultato di un imbalance di fase di 10° è mostrato in figura 3b.

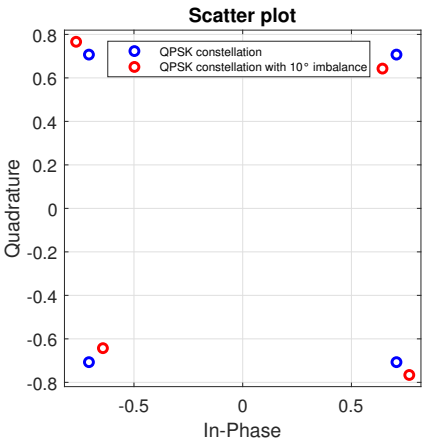
In generale, si può presentare sia un imbalance di ampiezza sia di fase come mentre illustrato in figura 3c. Si possono descrivere i due segnali in fase e in quadratura nel seguente modo:

$$s_I(t) = g_I \cdot \mathrm{Re}\{s(t)\} \cdot e^{-i0.5I_p(\pi/180)} \tag{6}$$

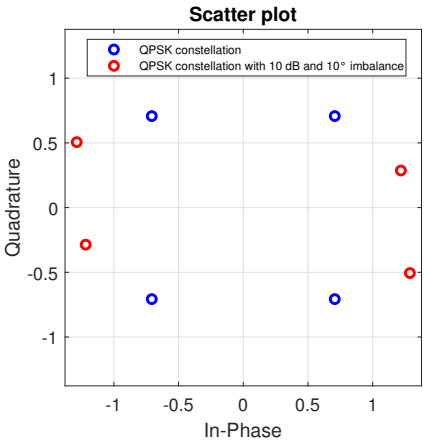
$$s_Q(t) = g_Q \cdot \mathrm{Im}\{s(t)\} \cdot e^{i0.5I_p(\pi/180)}. \tag{7}$$



(a) Imbalance di ampiezza di 10 dB



(b) Imbalance di fase di 10°



(c) Imbalance di ampiezza di 10 dB e di fase di 10°

Figura 3: Diversi tipi di imbalance

dove $g_I = 10^{0.5\frac{G}{20}}$ e $g_Q = 10^{-0.5\frac{G}{20}}$ e I_p rappresenta l'imbalance di fase.

Nella fase iniziale del progetto sono stati acquisiti tramite USRP diversi segnali modulati QPSK, con e senza IQ imbalance, prodotti con il programma WinIQSIM. Questo software è un ambiente di laboratorio con tutti gli strumenti necessari per l'analisi e la valutazione dei sistemi di trasmissione digitale. È possibile simulare tutti i sistemi coinvolti nel processo di modulazione e ricezione dei segnali. Il programma consente di definire blocchi di origini dati, configurarli e persino simulare la distorsione o l'interferenza del segnale. La USRP è stata usata con Simulink per salvare i campioni acquisiti e processarli successivamente. I segnali sono stati salvati come 801 frame contenenti 12500 campioni ciascuno, con una frequenza di campionamento di 200 kHz.

Il capitolo 2 mostra il ricevitore QPSK utilizzato in Simulink e i risultati ottenuti. Nel ricevitore occorre inserire dei blocchi che possano effettuare il recupero del sincronismo di simbolo e di portante. Il modello Simulink mostra lo spettro del segnale prima e dopo il filtro a coseno rialzato in ricezione e la costellazione dei campioni prima e dopo il sincronismo di simbolo e di portante.

Il capitolo 3 descrive gli algoritmi usati in Matlab per stimare e corregge l'IQ imbalance. Nello script è stato applicato un imbalance del 30% al segnale acquisito senza imbalance allo scopo confrontarlo con quello modificato in trasmissione. Successivamente, si è provato a correggerlo e a ritornare al segnale originale, ottenendo inoltre una stima del suo sbilanciamento. Per quanto riguarda i segnali acquisiti con imbalance, il processo di recupero dei valori di sbilanciamento è complicato dall'ambiguità di fase al ricevitore. Per questo è stato scritto un algoritmo dedicato solo a questi segnali.

2 Modello Simulink

I segnali usati per questo progetto sono stati acquisiti tramite una USRP B210 che acquisisce i campioni di una modulazione QPSK con portante a 3 GHz con una frequenza di campionamento di 200 kHz. I dati sono stati acquisiti usando Simulink, esportati sul workspace di Matlab e salvati per essere infine usati dal ricevitore QPSK.

Il modello Simulink per visualizzare i segnali salvati è stato preso dall'esempio "QPSK Transmitter and Receiver in Simulink" [8] ed è illustrato in figura 4. Il blocco "simout1" prende dal workspace i file precedentemente salvati come 801 frame composti da 12500 campioni. Nel campo "sample time" occorre inserire l'intervallo di tempo necessario per acquisire un frame da 12500 campioni, cioè il prodotto tra il tempo di campionamento e i campioni di un frame. Il sample time è dunque uguale a `(1/200e3)*12500`, notando che la frequenza di campionamento è di 200 kHz.

Il buffer viene utilizzato per convertire i campioni scalari in un frame di output a una velocità inferiore. In questo caso, il frame di output ha una velocità pari a quella di ingresso in quanto nel campo "Output buffer size" è stato inserito il valore 12500.

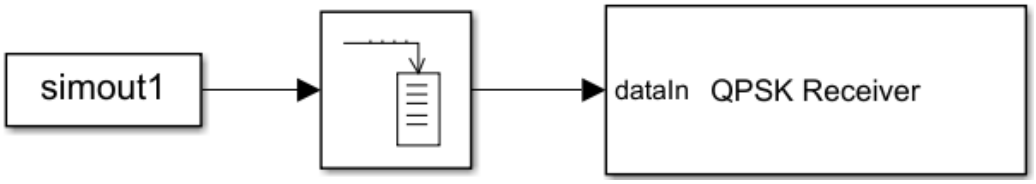


Figura 4: Modello Simulink

Il blocco "QPSK Receiver" è mostrato in 5. Il modello del ricevitore QPSK è composto principalmente da:

- un filtro a coseno rialzato con un fattore di rolloff di 0.9
- un compensatore della frequenza che stima un offset della frequenza del segnale ricevuto e lo corregge
- un sincronizzatore di simbolo che ricampiona il segnale in ingresso secondo una temporizzazione in modo che le decisioni sui simboli vengano prese negli istanti di campionamento ottimali
- un sincronizzatore di portante che compensa l'offset di frequenza e di fase

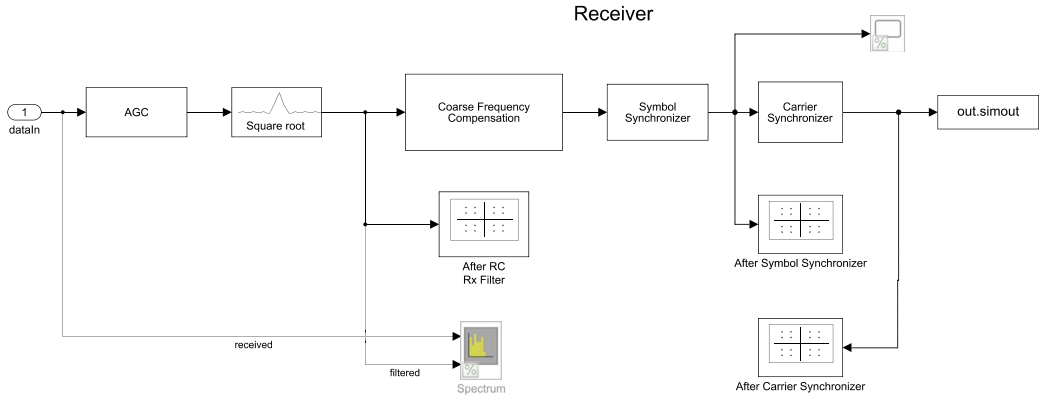


Figura 5: Modello ricevitore

Di seguito sono descritti più in dettaglio i vari blocchi che compongono il ricevitore. Il filtro a coseno rialzato ha un fattore di rolloff di 0.9, una larghezza del filtro di 8 simboli e 2 campioni per simbolo in input.

Il blocco AGC (Automatic Gain Controller) regola in modo adattivo il proprio guadagno per ottenere un livello di segnale costante in uscita, garantendo che i guadagni equivalenti dei rilevatori di errore di fase e di temporizzazione si mantengano costanti nel tempo. Questo blocco è necessario perché l'ampiezza del segnale ricevuto influisce sulla precisione della portante e del sincronismo di simbolo. La potenza in uscita dall'AGC è stata impostata pari a 2 W.

Il sottosistema Coarse Frequency Compensation è mostrato in figura 6. Il Coarse Frequency Compensation corregge il segnale di ingresso con una stima approssimativa dell'offset di frequenza che viene stimato mediando l'output dell'algoritmo, basato sulla correlazione, del blocco Coarse Frequency Compensator. La compensazione viene eseguita dal blocco Phase / Frequency Offset. Di solito c'è un offset di frequenza residua anche dopo la compensazione grossolana della frequenza, che provocherebbe una lenta rotazione

della costellazione. Il blocco Carrier Synchronizer compensa questa frequenza residua. Il Coarse Frequency Compensator è configurato con un offset di frequenza massimo di 12 500 kHz.

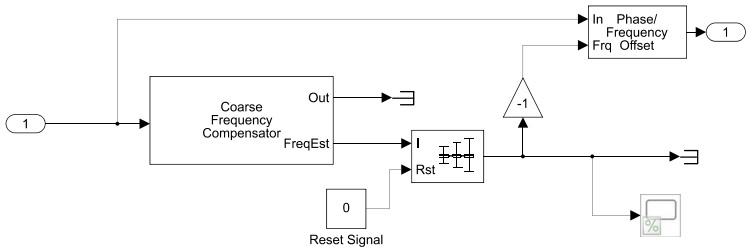


Figura 6: Coarse Frequency Compensation

Il blocco Symbol Synchronizer si occupa di campionare correttamente il simbolo nel tempo ottimo di campionamento. Il Symbol Synchronizer implementa una PLL per correggere l'errore di temporizzazione nel segnale ricevuto. L'errore viene stimato utilizzando l'algoritmo di Gardner che non dipende dalla rotazione della costellazione. Il Symbol Synchronizer lavora con 8 campioni per simbolo. I parametri Damping factor, Normalized loop bandwidth e Detector gain hanno i valori predefiniti presenti nell'esempio e sono impostati a 1 (smorzamento critico), 0.01 e 5.4 rispettivamente, in modo che la PLL si agganci rapidamente al tempo corretto introducendo un piccolo jitter temporale.

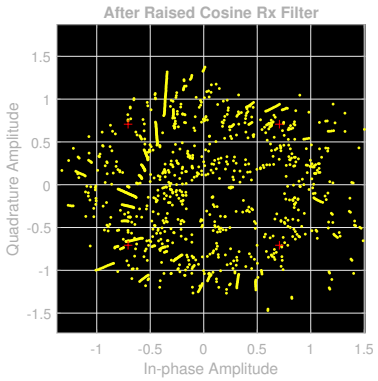
Il Carrier Synchronizer è necessario per sistemare l'offset di frequenza residua e dell'offset di fase nel segnale di ingresso. Questo blocco implementa un anello ad aggancio di fase (PLL). Il Carrier Synchronizer lavora con 8 campioni per simbolo. I parametri Damping factor e Normalized loop bandwidth

del blocco sono impostati rispettivamente a 1 (smorzamento critico) e 0.01, in modo che la PLL si agganci rapidamente alla fase desiderata introducendo un leggero rumore di fase.

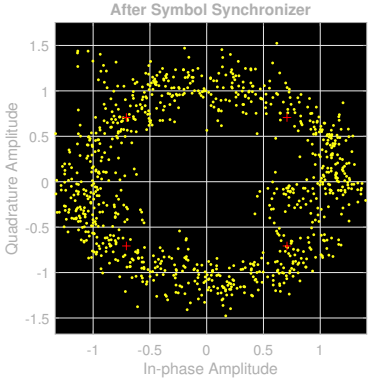
Di seguito sono illustrati i risultati che si ottengono dal modello Simulink. Le figure 7a, 7b e 7c mostrano le costellazioni dopo che il segnale senza imbalance è stato processato da ogni blocco Simulink, mentre le figure 8a, 8b e 8c rappresentano le costellazioni dopo che il segnale con imbalance di ampiezza al 30% è stato processato da ogni blocco Simulink.

Da una prima analisi visiva, non è facile verificare la presenza di imbalance sui simboli direttamente in trasmissione osservando le figure 7a e 8a. La prima figura, a causa della rotazione dei punti della costellazione QPSK, dovrebbe rappresentare una circonferenza, mentre la seconda figura dovrebbe rappresentare un'ellisse a causa della rotazione di una costellazione con imbalance. Si può osservare che la presenza di imbalance non è verificabile da una rapida analisi visiva dei simboli acquisiti. Occorre usare gli altri blocchi Simulink per discriminare l'imbalance.

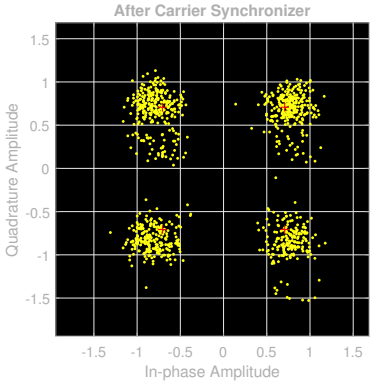
Da un confronto tra i diagrammi della costellazione QPSK della figura 7c e il diagramma 8c, si può notare che i punti della costellazione del segnale con imbalance non sono centrati nei punti della costellazione QPSK. Dunque, si può concludere che nella seconda figura è presente un imbalance di ampiezza.



(a) Segnale senza imbalance dopo il filtro a coseno rialzato

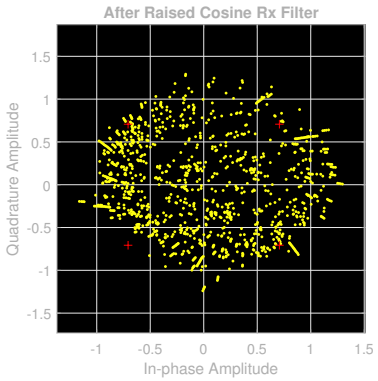


(b) Segnale senza imbalance dopo il symbol synchronizer

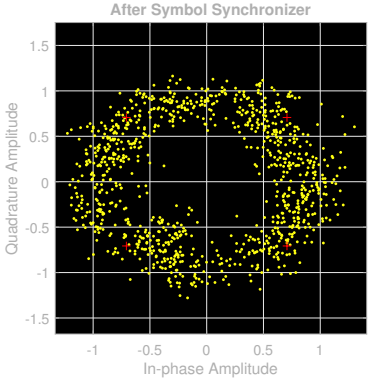


(c) Segnale senza imbalance dopo il carrier synchronizer

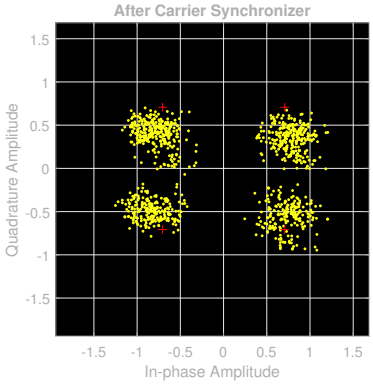
Figura 7: Segnale senza imbalance in Simulink



(a) Segnale con 30% di imbalance dopo il filtro a coseno rialzato



(b) Segnale con 30% di imbalance dopo il symbol synchronizer



(c) Segnale con 30% di imbalance dopo il carrier synchronizer

Figura 8: Segnale con 30% imbalance in Simulink

3 Script Matlab

Per stimare e correggere l'imbalance dei segnali acquisiti con il modello Simulink precedentemente descritto è stato scritto uno script Matlab. Nel programma vengono inizialmente caricati nel workspace i dati delle acquisizioni, salvati precedentemente con estensione `.mat` come mostrato nel codice 1. Questi file contengono due diversi tipi di dato per ognuno dei segnali ricevuti dalla USRP, uno in forma di matrice di dimensione 801×12500 e uno in forma di timeseries. La timeseries rappresenta l'evoluzione temporale di un processo dinamico ed è utilizzata per modellare dei dati che vengono campionati su intervalli di tempo discreti. Questo tipo di dato è necessario per simulare i segnali in Simulink. I segnali usati nel progetto sono elencati di seguito:

- signal_1: segnale senza imbalance
- signal_2: segnale senza imbalance con potenza maggiore rispetto signal_1
- signal_IQ10: segnale con 10% di imbalance in trasmissione
- signal_IQ20: segnale con 20% di imbalance in trasmissione
- signal_IQ30: segnale con 30% di imbalance in trasmissione.

```
signal1 = load('Signal_1.mat');
signal2 = load('Signal_2.mat');
signal_IQ10 = load('Signal_IQ10.mat');
signal_IQ20 = load('Signal_IQ20.mat');
signal_IQ30 = load('Signal_IQ30.mat');
```

Codice 1: Codice per caricare i segnali nel workspace

La prima parte dello script ha l'obbiettivo di modificare uno dei due segnali originali al quale non era stato imposto un IQ imbalance in trasmissione applicando un certo imbalance, il segnale usato nell'esempio è `signal1`. La funzione `apply_IQ_imbal` consente di applicare un imbalance di ampiezza e fase ad un segnale qualsiasi. Nel nostro caso siamo interessati solo a quello di ampiezza, inseriamo quindi come parametro di ingresso alla funzione il valore in dB desiderato: $20 \cdot \log_{10}(1.3)$ per un imbalance del 30%. A questo punto la funzione aumenta la parte reale e riduce quella immaginaria del valore inserito, per ogni campione.

```
ampImb = 20*log10(1.3); % 30% imbalance
phImb = 0; % 0 degrees
Fs = 2e05; % sampling frequency
[imbalanced_signal,imbalanced_signal_timeseries] = ...
    apply_IQ_imbal(signal1.simout, ampImb, phImb, Fs);
```

Codice 2: Codice per applicare IQ imbalance al segnale originale

Il risultato della funzione `apply_IQ_imbal` è mostrato nelle figure 9 e 10, in cui sono illustrati gli ultimi 1000 campioni rispettivamente della parte reale e immaginaria del segnale sbilanciato. Si può vedere che la parte reale del segnale con imbalance è stata aumentata rispetto al segnale originale, mentre la parte immaginaria è stata attenuata.

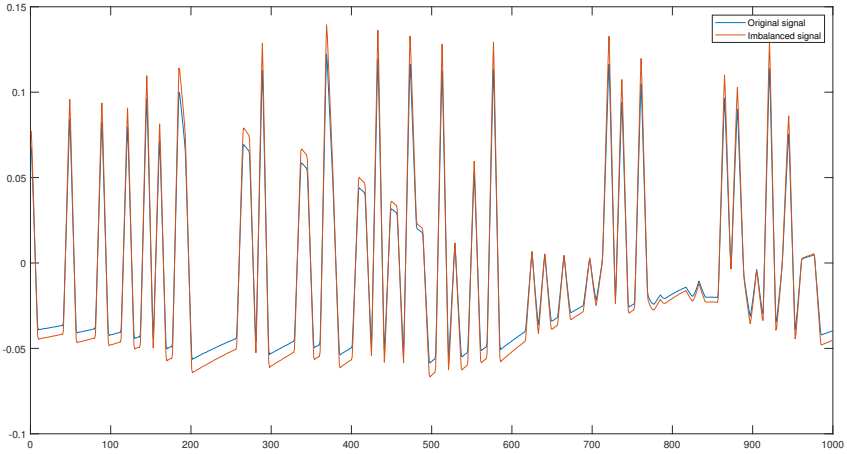


Figura 9: Confronto della parte reale del segnale con imbalance e del segnale originale

La funzione `apply_IQ_imbal` è descritta nel codice 3 in cui si può vedere che in ingresso prende una matrice chiamata `signal`, un imbalance di ampiezza `A` e di fase `P` e la frequenza di campionamento `Fs`. La funzione Matlab `iqimbal` applica un IQ imbalance ad un segnale di ingresso che può essere vettore o matrice. La frequenza di campionamento `Fs` è necessaria per ricostruire la timeseries secondo dei valori temporali corretti. La funzione `apply_IQ_imbal` restituisce il segnale con imbalance come matrice `signal_imb` e come timeseries `signal_imb_timeseries`, quest'ultima è utile per simulare il segnale in Simulink.

Si possono osservare i risultati applicando il segnale `signal_imb_timeseries` in ingresso al modello Simulink nelle figure 11a, 11b e 11c. In questo caso

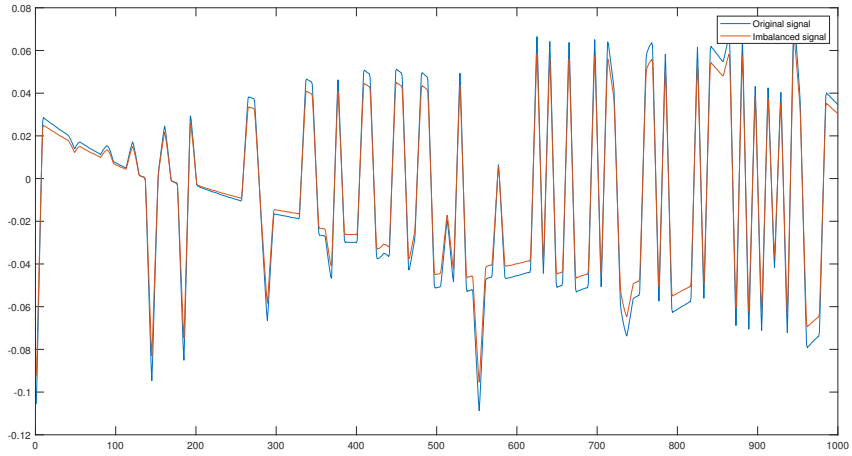


Figura 10: Confronto della parte immaginaria del segnale con imbalance e del segnale originale

la figura 11a si avvicina ad un elisse, ma Simulink non mostra la costellazione con l'imbalance appropriato in figura 11c. Simulink, infatti, non riesce a recuperare l'imbalance se viene applicato direttamente sul segnale ricevuto.

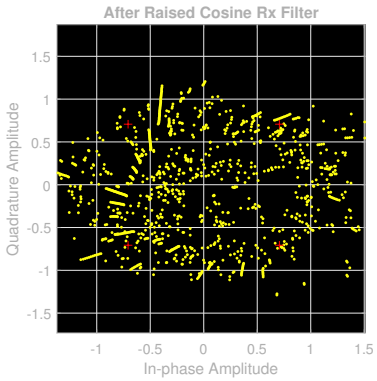
```
function [signal_imb,signal_imb_timeseries] = ...
    apply_IQ_imbal(signal, A, P, Fs)
    signal_imb = iqimbal(signal, A, P);
    L = size(signal);
    i = 0: 1 : L(1)-1;
    signal_imb_timeseries = timeseries(signal_imb, L(2)/Fs*i');
end
```

Codice 3: Funzione apply_IQ_imbal

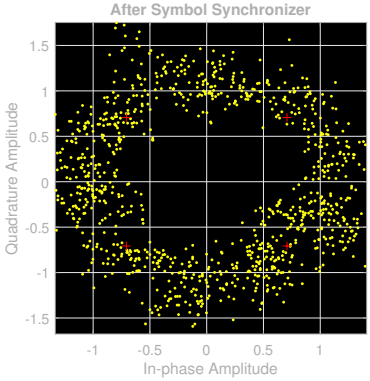
La fase successiva consiste del correggere il segnale precedentemente modificato come mostrato di seguito. La funzione `imbalance_correction` corregge un segnale generico preso come vettore, per questo si usa `reshape(imbalanced_signal.', 1, [])`, in cui può essere presente un imbalance sia di ampiezza che di fase.

```
corrected_signal = imbalance_correction(...
    reshape(imbalanced_signal.', 1, []), ampImb, phImb);
```

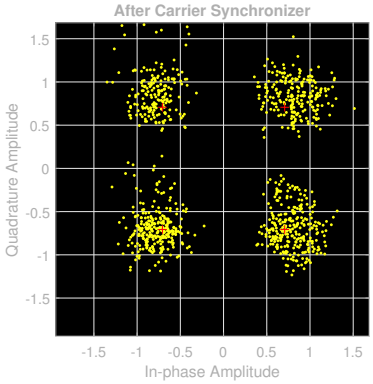
Codice 4: Codice per correggere l'IQ imbalance



(a) `signal_imb_timeseries` dopo il filtro a coseno rialzato



(b) `signal_imb_timeseries` dopo il symbol synchronizer



(c) `signal_imb_timeseries` dopo il carrier synchronizer

Figura 11: Segnale `signal_imb_timeseries` in Simulink

La funzione `imbalance_correction` prende in ingresso un vettore `signal`, un'imbalance di ampiezza `A` e di fase `P` ed è descritta nel codice 5.

Per capire il funzionamento di `iqimbal2coef` si definiscono due matrici S e X di dimensione $2 \times N$, dove N è il numero totale di campioni, che rappresentano rispettivamente il segnale ideale e di quello sbilanciato. La prima riga di entrambe le matrici contiene la parte reale del segnale, mentre la seconda riga quella immaginaria. La matrice X si ottiene come $X = K \cdot S$, dove K è una matrice 2×2 i cui valori sono determinati dall'imbalance di ampiezza A, espresso in decibel, e di fase P, espresso in gradi. La matrice K ha la forma

$$K = \begin{bmatrix} I_{gain}\cos(\theta_i) & Q_{gain}\cos(\theta_q) \\ I_{gain}\sin\theta_i & Q_{gain}\sin(\theta_q) \end{bmatrix}$$

dove

$$\begin{cases} I_{gain} = 10^{0.5\frac{A}{20}} \\ Q_{gain} = 10^{-0.5\frac{A}{20}} \\ \theta_i = -\dfrac{P}{2} \cdot \dfrac{\pi}{180} \\ \theta_q = \dfrac{\pi}{2} + \dfrac{P}{2} \cdot \dfrac{\pi}{180} \end{cases}$$

La matrice $Y = R \cdot X$ di dimensione $2 \times N$ è definita come la matrice di output del compensatore dell'imbalance. Per rimuovere completamente l'imbalance, R deve essere l'inversa della matrice K, quindi $R = K^{-1}$.

Da questa breve teoria, si capisce perché occorre lavorare con una matrice `X` di dimensione $2 \times N$, dove le due righe rappresentano rispettivamente la parte reale e immaginaria del segnale. Successivamente viene calcolato il valore `C` che rappresenta il coefficiente di compensazione calcolato dalla funzione `iqimbal2coef`. Dalla teoria [4], la matrice R che rimuove completamente l'imbalance è espressa come:

$$R = \begin{bmatrix} 1 + \mathrm{Re}\{C\} & \mathrm{Im}\{C\} \\ \mathrm{Im}\{C\} & 1 - \mathrm{Re}\{C\} \end{bmatrix}$$

Il segnale originale può essere recuperato tramite il prodotto tra il segnale con imbalance, `X`, e la matrice di correzione, `R`.

```
function [corrected_signal] = imbalance_correction(signal, A, P)
    X = [real(signal); imag(signal)];
    % iqimbal2coef converts an I/Q amplitude and phase imbalance
    % to its equivalent compensator coefficient
    C = iqimbal2coef(A,P);
    R = [1+real(C) imag(C); imag(C) 1-real(C)];
    corrected_signal = R*X;
```

```
end
```

Codice 5: Funzione imbalance_correction

Il risultato dell'output della funzione `imbalance_correction` è mostrato nei grafici 12 e 13 che illustrano il confronto tra gli ultimi 1000 campioni rispettivamente della parte reale e immaginaria del segnale corretto e di quello originale.

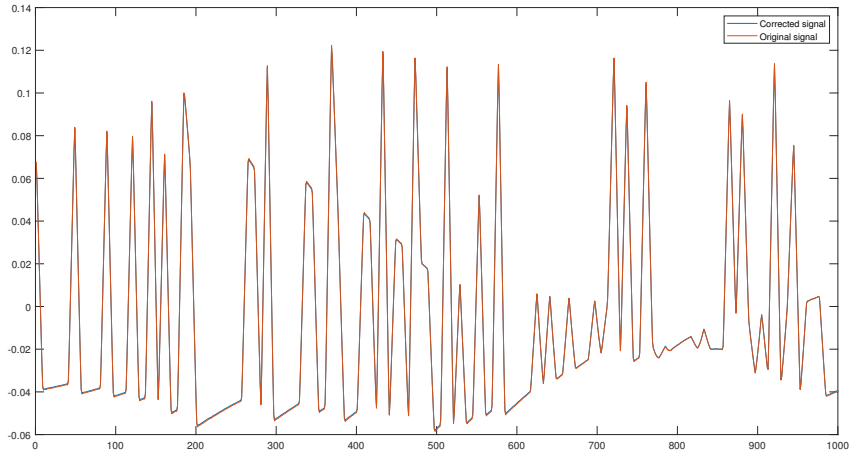


Figura 12: Confronto della parte reale del segnale corretto e del segnale originale

Successivamente, la funzione `imbalance_estimation` calcola la stima dell'imbalance di ampiezza e fase per un vettore che rappresenta un segnale, chiamato `imbalanced_signal`, a cui è stato applicato imbalance. Poiché il segnale in ingresso deve essere un vettore riga, si modifica opportunamente la matrice `imbalanced_signal` usando `reshape(imbalanced_signal.', [], 1)`.

```
[ampImbEst phImbEst] = imbalance_estimation(...
    reshape(imbalanced_signal', 1, []));
% Compare the estimated imbalance values with the specified ones.
[ampImbEst phImbEst; ampImb phImb]
```

Codice 6: Codice per stimare l'IQ imbalance

La funzione `imbalance_estimation` descritta nel codice 7 utilizza `comm.IQImbalanceCompensator` per creare l'oggetto `hIQComp` che ha lo scopo di compensare l'imbalance tra la componente in fase e in quadratura dell'input. Il segnale deve essere poi normalizzato, dividendolo per la sua deviazione

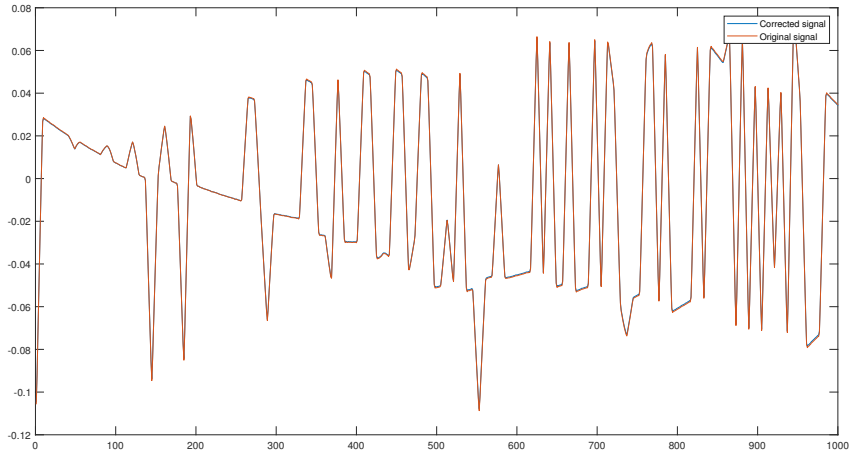


Figura 13: Confronto della parte immaginaria del segnale corretto e del segnale originale

standard. Successivamente, la funzione `step` stima l'I/Q imbalance del segnale di input, `normalized_signal`, e ritorna il segnale compensato `compSig` e i coefficienti di compensazione stimati `coef`. Infine, `iqcoef2imbal` prende l'ultimo coefficiente di compensazione e lo converte in due valori `ampImbEst` e `phImbEst`, i quali rappresentano rispettivamente la stima dell'imbalance di ampiezza e di fase.

```
function [ampImbEst phImbEst] = imbalance_estimation(signal)
    hIQComp = comm.IQImbalanceCompensator('CoefficientOutputPort',true);
    % Normalize the power of the signal
    normalized_signal = signal./std(signal);
    % step processes the input data, normalized_signal,
    % to produce the output for System object, hIQComp.
    [compSig,coef] = step(hIQComp,normalized_signal);
    % iqcoef2imbal Computes the amplitude imbalance and
    % phase imbalance that a given compensator
    % coefficient will correct.
    [ampImbEst,phImbEst] = iqcoef2imbal(coef(end));
    release(hIQComp);
end
```

Codice 7: Funzione imbalance_estimation

L'output di `imbalance_estimation` è molto simile ai valori attesi, infatti per il segnale modificato dalla funzione vista in precedenza `apply_IQ_imbal`, viene stimato un imbalance di ampiezza di 2.4623 dB, mentre quello atteso è

$20 \cdot \log_{10}(1.3) = 2.2789\,\text{dB}$ e un imbalance di fase di $0.262\,07°$, mentre quello atteso è di $0°$. La funzione `imbalance_estimation` non fornisce, invece, risultati soddisfacenti per i segnali ai quali è stato applicato l'imbalance prima della trasmissione, cioè per `signal_IQ10`, `signal_IQ20` e `signal_IQ30`.

La causa è probabilmente l'errore di frequenza di ripristino della portante QPSK che causa la rotazione della costellazione di simboli ricevuta rispetto alla costellazione attesa. Questo porta la nuvola di campioni ad essere indistinguibile da una nuvola senza imbalance come si può vedere dalla figura 8a.

Per i segnali a cui è stato applicato un IQ imbalance in trasmissione è stato quindi utilizzato un approccio diverso. In seguito al processamento su Simulink, dopo il blocco carrier synchronizer, è possibile osservare che le nuvole di campioni dei segnali con imbalance si addensano in quattro punti che rappresentano i vertici di un rettangolo, come mostrato in figura 8c. I segnali con imbalance dopo il carrier synchronizer sono stati esportati sul workspace e salvati in file di estensione `.mat`.

La parte dello script che calcola l'imbalance su questi segnali è mostrata nel codice 8. La funzione `imbalance_algorithm_estimation` prende in ingresso il segnale dopo il carrier synchronizer come un vettore riga e restituisce il valore `ampImbEst` che contiene la stima dell'imbalance di ampiezza.

```
signal_IQ30_after_carrier_synch = ...
    load('Signal_IQ30_after_carrier_synch.mat');
[ampImbEst] = imbalance_algorithm_estimation(...
    reshape((signal_IQ30_after_carrier_synch.simout)', 1,[]))
```

Codice 8: Codice per stimare l'IQ imbalance dei segnali dopo il carrier synchronizer

Per spiegare la funzione `imbalance_algorithm_estimation` occorre pensare ai punti della costellazione ideale QPSK come i vertici di un quadrato e ai punti della costellazione ideale sbilanciata come i vertici di un rettangolo. Nel nostro caso, poiché l'imbalance è del 30%, il rettangolo è posto in orizzontale, cioè con la base di lunghezza maggiore dell'altezza. Dalla simulazione di Simulink, si può vedere che il rettangolo, però, ruota nel tempo assumendo una posizione verticale o orizzontale. Questo è probabilmente dovuto all'errore di frequenza di ripristino della portante, come precedentemente notato.

L'idea è quindi quella di raggruppare i campioni in intervalli di lunghezza $N = 1000$, nel quale è visibile una forma coerente della costellazione e stimare la lunghezza dei lati calcolando la deviazione standard della parte reale e della parte immaginaria dei punti. Questo perché la parte reale rappresenta

idealmente metà lunghezza della base del rettangolo e la parte immaginaria metà lunghezza dell'altezza.

Inizialmente nella funzione `imbalance_algorithm_estimation` vengono eliminati i primi zeri con la funzione `delete_first_zeroes`. Successivamente, viene preso il centesimo intervallo di lunghezza $N = 1000$ come primo intervallo, si calcola dunque la radice quadrata del rapporto della deviazione standard della parte reale e della parte immaginaria di ogni intervallo di lunghezza N. Questo valore calcolato rappresenta il rapporto fra i lati del rettangolo. Per tenere conto della posizione verticale che può assumere il "rettangolo" durante la simulazione di Simulink, il rapporto viene invertito quando risulta minore di uno.

La presenza della radice quadrata nei calcoli è spiegata di seguito. Consideriamo un quadrato di lato A, i cui vertici rappresentano I punti di una costellazione QPSK ideale. L'effetto di un imbalance di ampiezza di valore K è quello di produrre una costellazione rettangolare, di base $A' = AK$ e di altezza $B' = A/K$. Il rapporto fra i lati sarà quindi $A'/B' = K^2$. Per ottenere il coefficiente K di imbalance, quindi, è necessario considerare la radice quadrata del rapporto fra i lati.

```
function [imbal] = imbalance_algorithm_estimation(signal)
    signal = delete_first_zeroes(signal);
    signal_real = real(signal);
    signal_imag = imag(signal);
    imbal = 0;
    N = 1000;
    for i = 100:(length(signal)/N)-1
        f_real = signal_real(i*N+1:i*N+N);
        f_imag = signal_imag(i*N+1:i*N+N);
        r = sqrt(std(f_real)./std(f_imag));
        if (r < 1)
            r = 1/r;
        end
        imbal = imbal + r;
    end
    imbal = imbal/(i-100);
end
```

Codice 9: Funzione imbalance_algorithm_estimation

4 Conclusioni

Questo progetto ha mostrato, tramite lo script Matlab, come sia possibile applicare e correggere l'IQ imbalance e come stimare l'imbalance da un segnale sconosciuto. Questa stima è migliore per un segnale sbilanciato da software rispetto a un segnale con imbalance acquisito da un trasmettitore.

Questo errore è dovuto al recupero della portante in ricezione e all'ambiguità di fase. Quest'ultima deriva dal fatto che il sincronizzatore non conosce il vero orientamento del segnale trasmesso. Per una data modulazione simmetrica ci possono essere una serie di orientamenti convergenti, che dipendono dall'ordine di modulazione. Ad esempio, la PAM avrà due possibili orientamenti, la QPSK e la QAM ne avranno quattro, mentre una M-PSK con avrà M possibili orientamenti [2].

Un sistema di recupero della portante è un circuito utilizzato per stimare e compensare le differenze di frequenza e di fase tra l'onda portante di un segnale ricevuto e l'oscillatore locale del ricevitore ai fini della demodulazione coerente. In un sistema di comunicazione ideale, gli oscillatori del segnale portante del trasmettitore e del ricevitore sarebbero perfettamente adattati in frequenza e fase, consentendo in tal modo una demodulazione perfetta e coerente del segnale modulato in banda base.

Tuttavia, trasmettitori e ricevitori raramente condividono lo stesso oscillatore portante. I sistemi di ricezione delle comunicazioni sono generalmente indipendenti dai sistemi di trasmissione e contengono i propri oscillatori con offset e instabilità di frequenza e fase. Gli errori che si presentano a causa di un errore di fase e di frequenza sono mostrate rispettivamente nelle figure 14a e 14b. Tutte queste variazioni di frequenza e di fase devono essere stimate utilizzando le informazioni nel segnale ricevuto per riprodurre o recuperare il segnale portante sul ricevitore e consentire una demodulazione coerente.

Per mostrare matematicamente l'effetto di un errore di fase e frequenza si possono prendere i segnali in fase e in quadratura del trasmettitore QPSK. Il segnale in fase modulato 2ASK a frequenza f_c può essere scritto come:

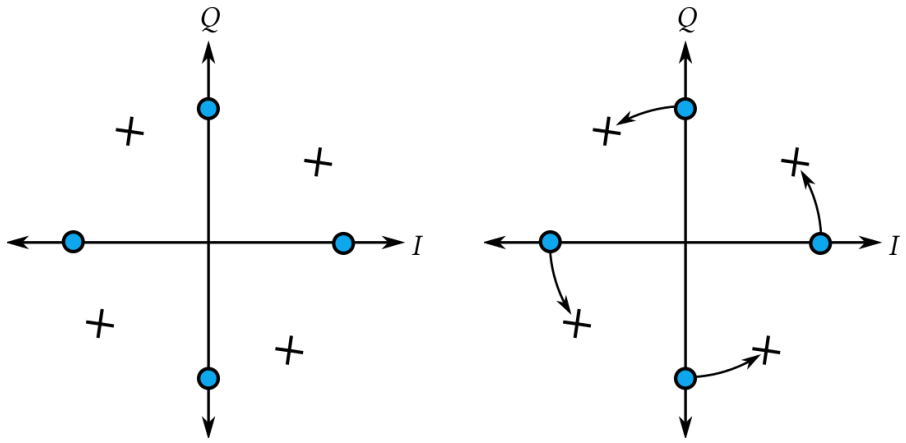
$$s(t) = \pm A \cos(2\pi f_c t + \phi_c). \tag{8}$$

Per effettuare una demodulazione coerente occorre moltiplicare $s(t)$ per la stessa portante a frequenza f_c e con fase ϕ_c. In ricezione $s(t)$ viene moltiplicato per un segnale a frequenza $f = f_c$ con fase $\phi = \phi_c$ come di seguito:

$$y(t) = s(t) \cdot \cos(2\pi f t + \phi). \tag{9}$$

Applicando le formule di Werner, l'equazione (9) diventa:

$$y(t) = \pm \frac{A}{2} \left[1 + \cos\left(4\pi f_c t + 2\phi_c\right)\right]. \tag{10}$$



(a) Errore di recupero della fase della portante QPSK che causa un offset fisso della costellazione [1]

(b) Errore di recupero della frequenza della portante QPSK che causa la rotazione della costellazione [1]

Figura 14: Diversi tipi di errore nel recupero della portante

Si può ottiene il segnale originale con un filtro passa basso che ha frequenza di taglio minore di $2f_c$.

Come già detto, non è possibile avere in ricezione una portante uguale a quella in trasmissione. In ricezione $s(t)$ viene moltiplicato per un segnale a frequenza $f \neq f_c$ con fase $\phi \neq \phi_c$. Usando le formule di Werner, nel caso di una portante in ricezione diversa da quella in trasmissione, l'equazione (9) diventa:

$$\begin{aligned} y(t) = \pm A\cos(2\pi f_c t + \phi_c) \cdot \cos(2\pi f t + \phi) = \\ \pm\frac{A}{2}\left\{\cos\left[2\pi(f_c - f)t + \phi_c - \phi\right] + \cos\left[2\pi(f_c + f)t + \phi_c + \phi\right]\right\}. \end{aligned} \tag{11}$$

Ponendo $f = f_c + \Delta_f$ e $\phi = \phi_c + \Delta_p$ si può riscrivere l'equazione (11) come:

$$y(t) = \pm\frac{A}{2}\left\{\cos\left(2\pi\Delta_f t + \Delta_p\right) + \cos\left[2\pi(2f_c + \Delta_f)t + 2\phi_c + \Delta_p\right]\right\} \tag{12}$$

avendo usato il fatto che $\cos\left(-2\pi\Delta_f t - \Delta_p\right) = \cos\left(2\pi\Delta_f t + \Delta_p\right)$. Dall'equazione (12) si può osservare che anche eliminando il termine a frequenza $2f_c$, il segnale $y(t)$ è tempo variante. L'errore di frequenza che fa ruotare la costellazione è rappresentato da Δ_f, mentre l'errore di fase che provoca un offset sulla costellazione è rappresentato da Δ_p.

Riferimenti bibliografici

[1] *Carrier recovery*. Wikipedia. URL: https://en.wikipedia.org/wiki/Carrier_recovery.

[2] Travis F. Collins, Robin Getz, Pu Di e Alexander M. Wyglinski. *Software-Defined Radio for Engineers*. Artech House, 2018. ISBN: 978-1-63081-457-1.

[3] *comm.IQImbalanceCompensator*. Matlab. URL: https://it.mathworks.com/help/comm/ref/comm.iqimbalancecompensator-system-object.html.

[4] *iqcoef2imbal*. Matlab. URL: https://www.mathworks.com/help/comm/ref/iqcoef2imbal.html.

[5] *iqimbal*. Matlab. URL: https://it.mathworks.com/help/comm/ref/iqimbal.html.

[6] *iqimbal2coef*. Matlab. URL: https://it.mathworks.com/help/comm/ref/iqimbal2coef.html.

[7] *QPSK modulation and demodulation*. GaussianWaves. URL: https://www.gaussianwaves.com/2010/10/qpsk-modulation-and-demodulation-2/.

[8] *QPSK Transmitter and Receiver in Simulink*. Matlab. URL: https://it.mathworks.com/help/comm/ug/qpsk-transmitter-and-receiver-in-simulink.html.

[9] *Quadrature phase-shift keying (QPSK)*. Wikipedia. URL: https://en.wikipedia.org/wiki/Phase-shift_keying#Quadrature_phase-shift_keying_(QPSK).