ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA MAGISTRALE IN
Ingegneria Elettronica e Telecomunicazioni per lo Sviluppo Sostenibile

TITOLO DELLA TESI

Sviluppo di un codice Python per l'assistenza al progetto di PLL a rapporto intero o frazionario

Tesi in
Elettronica per l'elaborazione analogica del segnale LM

Relatore
Prof. Sergio Callegari

Presentata da
Raffaele Nucci

Anno accademico 2016 - 2017

Parole chiave

PLL

Modulatore Σ-Δ

Scientific Python

Assistenza al progetto

Sintetizzatori di frequenza a rapporto frazionario

Sommario

Introduzione

In questi ultimi anni la richiesta e l'utilizzo di sistemi wireless sono diventati di fondamentale importanza e questo ha determinato la necessità di impiego di circuiti elettronici ad elevato livello di integrazione e a basso consumo di potenza.

Un circuito elettronico ampiamente utilizzato in questo ambito è l'anello ad aggancio di fase o *Phase Locked Loop* (PLL), il quale ha il compito di generare un segnale periodico la cui fase è in relazione costante con la fase di un segnale di riferimento. Il PLL trova largo impiego nei generatori di clock per sistemi a microprocessore, nei circuiti di ricostruzione del clock, detti anche *clock recovery*, in grado di estrarre il clock da un segnale modulato e aperiodico, nelle telecomunicazioni può essere utilizzato come demodulatore FM (Frequency Modulation) oppure nella sintesi di frequenza. Quest'ultimo è l'impiego maggiormente approfondito in questa tesi.

Nella sintesi di frequenza, l'architettura del PLL viene impiegata per pilotare un oscillatore controllato in tensione o VCO (Voltage Controlled Oscillator) che è in grado di produrre un segnale ad elevata frequenza, ma che di per sé avrebbe una scarsa precisione, con un segnale di riferimento preciso, avente una frequenza più bassa rispetto a quelle normalmente utilizzate in ambito delle telecomunicazioni. Questa sintonizzazione si ottiene con un anello di retroazione che consente di confrontare la fase del segnale di riferimento con la fase del segnale di uscita del VCO a valle di un divisore di frequenza, il quale definisce il rapporto fra la frequenza del segnale di uscita e la frequenza del segnale di riferimento. L'aggancio di fase è infatti strumentale alla sintonizzazione frequenziale, essendoci un legame integro-differenziale fra la fase e la frequenza del segnale di uscita.

Il divisore di frequenza per sua natura può mettere a disposizione solo rapporti interi fra la frequenza di uscita e quella di ingresso, per cui il PLL nella sua forma più semplice, detta appunto a rapporto intero, riesce ad ottenere segnali in uscita vincolati ad avere frequenze multiple di quella di riferimento, non è quindi facile ottenere una regolazione fine della frequenza sintetizzata. Infatti, a tale scopo, sarebbe necessario un segnale di riferimento a bassa frequenza insieme ad un rapporto di moltiplicazione intero elevato. Un tale set up, benché in linea di principio praticabile, dal punto di vista pratico evidenzia delle

problematicità significative, non ultime quelle legate al rumore di fase, che risulterebbe esaltato.

Per questo motivo, sono stati introdotti PLL caratterizzati da un'architettura più sofisticata, in grado di offrire un rapporto di frequenza frazionario. Essi consentono di avere una buona risoluzione in frequenza con cui sintonizzare l'uscita e al tempo stesso permettono di usare riferimenti di frequenza più elevata.

Il vantaggio di avere una regolazione fine della frequenza, oltre che ad una elevata velocità di commutazione delle frequenze sintetizzate e buone performance in termini di rumore di fase in uscita, determina però anche una maggiore complessità del progetto, in quanto per avere un rapporto frazionario è necessario incorporare nell'architettura del divisore un modulatore Σ-Δ (sigma- delta).

Il ruolo del modulatore Σ-Δ è di interfacciarsi con il divisore di frequenza, definendone il rapporto di divisione ad ogni ciclo. In particolare, il modulatore Σ-Δ è in grado di produrre una sequenza opportuna che consente di commutare rapidamente fra diversi rapporti interi, emulando così un rapporto frazionario. Questo approccio introduce un'ulteriore componente di rumore denominata rumore di quantizzazione.

Per consentire l'analisi delle performance dinamiche e del rumore in uscita del sintetizzatore di frequenze che incorpora il modulatore Σ-Δ, è possibile sfruttare opportuni modelli matematici. Attraverso la simulazione del modello, è possibile verificare la stabilità del sistema valutando direttamente la risposta. È chiaro quindi che ci si trova di fronte alla necessità di sviluppare modelli e sistemi di simulazione più articolati di quelli che sarebbero sufficienti a una semplice valutazione mediante il calcolo del margine di fase e di ampiezza dell'anello.

In alternativa alla simulazione "nel dominio del tempo" dell'intero modello non lineare del PLL, che richiede sempre tempi molto lunghi (anche estremamente lunghi in funzione del livello di dettaglio con cui si intende valutare il comportamento del sistema) e che comunque offre sempre risultati molto specifici e quindi difficilmente generalizzabili in linee guida di progetto, è possibile far riferimento a modelli approssimati definiti "nel dominio delle frequenze". L'idea è quella di identificare tutte le possibili sorgenti di rumore che possono contribuire al rumore di fase all'uscita del PLL e di definire in forma approssimata delle funzioni che pesino il ruolo dei diversi contributi alle diverse frequenze all'uscita del sistema.

Tipiche sorgenti di rumore da considerare sono quelle relative alla rumorosità del comparatore di fase, del VCO, ecc. Ovviamente, nel caso di PLL a rapporto frazionario va preso in considerazione anche il rumore di quantizzazione introdotto dal modulatore Σ-Δ.

Un approccio in frequenza non può prescindere da operazioni di linearizzazione ed è quindi necessariamente meno accurato di una simulazione time domain del modello non lineare di origine. Tuttavia, consente di ottenere risultati in tempi infinitamente più brevi, e in una forma non specifica, che si presta ad essere tradotta in linee guida di progetto.

Ad esempio, una volta stabilito in che modo il rumore di quantizzazione del modulatore Σ-Δ si propaga all'uscita del PLL, questa informazione può venire impiegata per progettare il modulatore Σ-Δ in modo che il suo rumore di quantizzazione non diventi la componente di rumore dominante all'interno del sistema.

In altri termini, l'approccio in frequenza offre strumenti che si prestano ad essere incorporati in soluzioni di assistenza software al progetto o electronic design assistance (EDA).

Lo scopo di questa tesi è proprio quello di utilizzare uno dei principali approcci per la modellazione nel dominio delle frequenze delle componenti di rumore di PLL a rapporto intero e frazionario, codificandolo in uno strumento di assistenza al progetto. Il framework che si è scelto di utilizzare è quello sviluppato da Perrott al Massachusetts Institute of Technology di Cambridge tra il 2002 e il 2010 [1] [2].

In particolare, l'obiettivo prefissato era di sviluppare un tool open source, in grado di replicare ed estendere le funzionalità di un codice attualmente disponibile solo in forma proprietaria e chiusa, benché gratuita [3]. Le ragioni sulla base delle quali si è deciso per un approccio a codice aperto sono fondamentalmente tre. La prima è di flessibilità. Porzioni di un codice aperto possono venire arricchite via via di nuove funzionalità o venire incorporate in codici più articolati. La seconda ha carattere didattico: un codice aperto può venire studiato e diventare strumento di formazione. La terza è di riproducibiltà della ricerca. Quando nell'ambito di una ricerca viene usato un codice aperto, i risultati sono più facilmente riproducibili e verificabili da altri esperti.

Per il tool si è scelto di sviluppare codice nel linguaggio di programmazione Python [4] [5], impiegando alcune popolari librerie di estensione per il calcolo numerico [6] [7] [8] e scientifico [9] [10].

Il ruolo del software di assistenza al progetto è quello di acquisire le specifiche del sintetizzatore di frequenza in termini di parametri caratterizzanti il comportamento ad anello chiuso (quali la funzione di trasferimento ad anello chiuso richiesta al sistema) e di derivare da essi le specifiche dei filtri da incorporare nel PLL, oltre che una caratterizzazione del comportamento atteso. Quest'ultima comprende tra l'altro la risposta del sintetizzatore ad una sollecitazione a gradino (cioè alla richiesta di passare a un'altra frequenza di uscita) e una caratterizzazione completa della rumorosità di fase in uscita.

Le specifiche comprendevano inoltre la capacità di compensare in fase di progetto la presenza di comportamenti reattivi parassiti in alcuni elementi del PLL (caratterizzati come poli e zeri parassiti della funzione ad anello aperto del modulatore), e di valutare la sensibilità del modulatore (in termini di comportamento atteso), alla variazione dei parametri di modello associati ad alcuni componenti.

La Figura 1 illustra graficamente l'operatività del software in termini di input richiesti, passi di elaborazione, output forniti.

Rispetto ai codici proprietari già disponibili le differenze fondamentali sono principalmente due. La prima è un'estensione delle funzionalità (p.e. la capacità di gestire modelli di ordine più alto o funzioni di trasferimento di forma arbitraria). La seconda è la presenza di interfacce di tipo "programmatico" ovvero la possibilità di usare il codice come libreria all'interno di altri codici EDA più articolati.

Quest'ultimo aspetto è cruciale con riferimento al contesto in cui il lavoro di tesi si è configurato. Il lavoro è stato infatti svolto all'interno di un gruppo di ricerca dell'università di Bologna, impegnato in una linea di lavoro relativa all'ottimizzazione delle caratteristiche di noise shaping dei modulatori Σ-Δ. Nello specifico, il gruppo si occupa dell'uso di tecniche formali di ottimizzazione convessa nella determinazione dei parametri dei filtri impiegati all'interno dei modulatori Σ-Δ per far sì che il rumore di quantizzazione prodotto dai modulatori abbia determinate caratteristiche [11].

Il gruppo di ricerca ha già sviluppato una serie di strumenti EDA per il progetto di modulatori Σ-Δ raccolti nel pacchetto software open source PyDSM [12] [13], basato sul Python scientifico [14]. L'interesse del gruppo di ricerca nei PLL è quindi ovvio, trattandosi di un campo applicativo fondamentale per i modulatori Σ-Δ.

Benché il codice prodotto nell'ambito del lavoro di tesi sia estremamente generale e assolutamente utile anche al progetto di PLL a rapporto intero, il gruppo di ricerca focalizza evidentemente la sua attenzione sulle funzionalità relative ai modulatori a rapporto frazionario. L'aspettativa è che in futuro, il codice prodotto con il lavoro di tesi possa venire integrato all'interno di PyDSM supportando il progetto di modulatori Σ-Δ "ottimi" relativamente alla loro incorporazione in soluzioni di sintesi di frequenza.

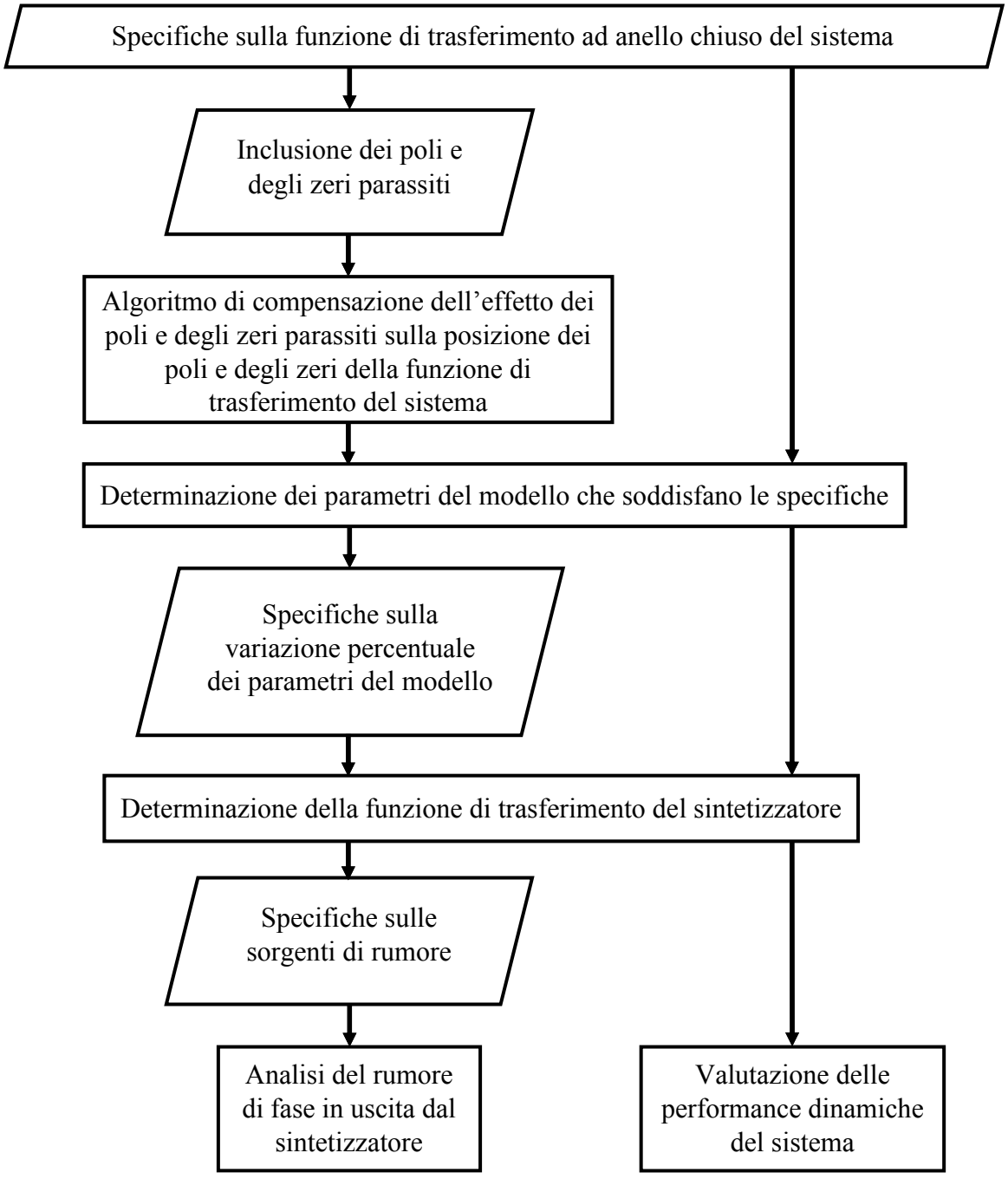

Figura 1: diagramma di flusso del progetto

Capitolo 1

BACKGROUND

In questo capitolo, verranno illustrate brevemente le conoscenze pregresse necessarie per comprendere la struttura e il funzionamento dei singoli dispositivi che costituiscono un sintetizzatore di frequenza a rapporto frazionario *N* con modulatore Σ-Δ e i modelli nel dominio del tempo e nel dominio della frequenza di ciascun dispositivo, al fine di ottenere un modello complessivo del sintetizzatore, rappresentabile come funzione di trasferimento del sistema. Inoltre, verrà brevemente descritta la disposizione dei poli e degli zeri della funzione di trasferimento per le diverse forme implementate in questo strumento di assistenza al progetto di PLL.

Capitolo 1.1: Sintetizzatore di frequenza a rapporto frazionario *N* con modulatore Σ-Δ

Il sintetizzatore di frequenza a rapporto frazionario *N* con modulatore Σ-Δ è realizzato mediante un sistema in retroazione costituito da un PLL. In particolare, è in grado di sintonizzare un semplice VCO a basso costo ad una frequenza pari a *N* volte la frequenza del segnale di riferimento, con *N* frazionario imposto dall'azione combinata del modulatore Σ-Δ e del divisore di frequenza. Essendoci un legame integro-differenziale fra la frequenza e la fase del segnale di uscita del sintetizzatore, la sintonizzazione è ottenuta grazie all'azione di aggancio di fase del PLL.

Il sintetizzatore è costituito da un PFD (*phase-frequency detector*) ovvero un rilevatore della frequenza o della fase del segnale in ingresso, una pompa di carica (*charge pump*), un filtro in anello aperto (*Loop Filter*) caratterizzato da una funzione di trasferimento *H(s)*, un VCO, un divisore di frequenza (*Frequency Divider*) e un modulatore Σ-Δ.

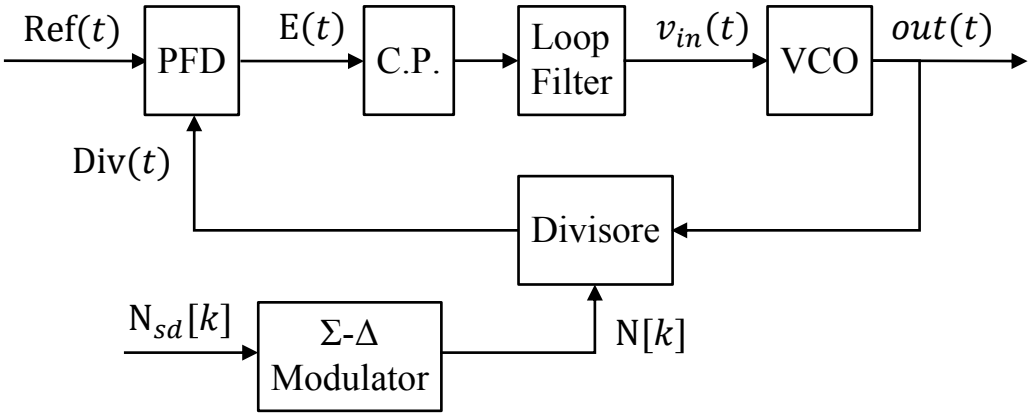

Figura 2: Diagramma a blocchi di un sintetizzatore delta-sigma frazionario-N

Phase Frequency Detector

Il PFD è circuito che fornisce in uscita un segnale E(t) non nullo se i due segnali di ingresso Ref(t) e Div(t) sono sfasati fra di loro. L'architettura di questo componente può essere sia analogica che digitale, ma ci si soffermerà su due architetture digitali aventi un approccio basato sull'XOR o sul TRISTATE.

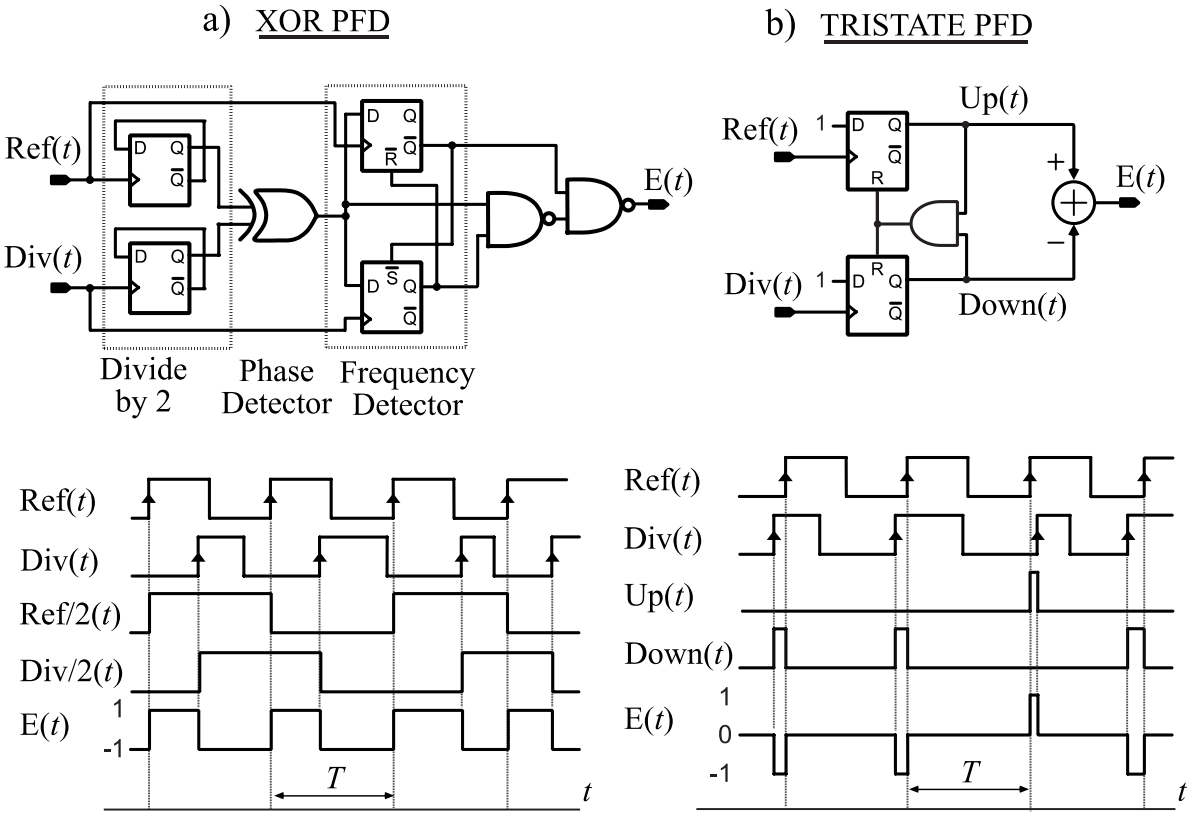

Figura 3 : Architetture del Phase Frequency Detector (PFD) considerate:

a) basata su approccio XOR

b) basata su approccio a tre stati. Fonte [3]

Come rappresentato in Figura 3, l'architettura TRISTATE consente di avere in uscita un segnale E(t) a 3 livelli, costituito da impulsi di breve durata. Questa architettura determina vantaggi in termini di consumo di energia, in quanto la pompa di carica in cascata al PFD erogherà potenza per tempi minori rispetto al tempo di ciclo T, corrispondente al periodo del segnale di riferimento Ref(t), e diminuirà di conseguenza il rumore introdotto dalla pompa di carica. Un altro vantaggio è il contenimento del rumore dovuto alle componenti spurie legate al segnale di riferimento Ref(t) [3]. In controparte qualsiasi mismatch strutturale dei due flip-flop D determina una differente caratteristica delle uscite Up(t) e Down(t) e quindi un degrado della linearità della funzione che lega l'errore di fase in ingresso con la tensione che rappresenta l'errore di fase all'uscita del filtro. Inoltre, la porta AND che confronta i segnali Up(t) e Down(t) deve essere molto veloce per evitare di produrre problemi di *dead zone* (letteralmente "zona morta") legati al ritardo introdotto dalla porta sul reset dei due flip-flop D. La dead zone è un intervallo di possibili differenze di fase fra i segnali di ingresso del PFD che non viene rilevato dalla pompa di carica, di conseguenza si crea un offset fra il segnale di riferimento e il segnale di uscita del PLL.

L'architettura XOR è priva di problemi di dead zone e offre un'ottima caratteristica lineare dell'errore di fase in uscita E(t), in questo caso a due livelli, ma a discapito del maggior consumo di energia della pompa di carica e di conseguenza maggior rumore introdotto dalla stessa, oltre che un aumento delle componenti spurie del rumore legate al segnale di riferimento Ref(t) [3].

La scelta fra le due architetture è da valutare in base all'impiego del sintetizzatore di frequenze. Se si realizza un sintetizzatore di frequenze a rapporto intero conviene scegliere l'architettura TRISTATE, perché non è richiesta un'ottima linearità ed è necessario ridurre le componenti di rumore introdotte dalla pompa di carica e dal segnale di riferimento, essendo la frequenza del segnale di riferimento f_{Ref} bassa. Se invece si realizza un sintetizzatore di frequenza a rapporto frazionario N con modulatore Σ-Δ, conviene scegliere l'architettura XOR, perché è necessaria una buona linearità per migliorare il modo in cui viene processato il rumore di quantizzazione, ma anche perché le componenti di rumore introdotte dal riferimento e dalla pompa di carica sono proporzionali al periodo $T=1/f_{\mathrm{Ref}}$ del segnale di riferimento Ref(t), il quale ha ad una frequenza f_{Ref} potenzialmente più alta rispetto al PLL a rapporto intero, grazie al rapporto frazionario N.

Pompa di carica

La pompa di carica (charge pump) ha il compito di creare una tensione legata all'errore di fase E(t). Questa presenta un'architettura in cui l'uscita è una corrente che carica un nodo di tipo capacitivo, in particolare la corrente in uscita dalla pompa è idealmente un'onda quadra di ampiezza pari al guadagno I_{cp}, ovvero una versione scalata di un fattore I_{cp} del segnale errore E(t). L'ingresso del Loop Filter sarà di conseguenza una corrente alternata a due livelli $+I_{cp}$ e $-I_{cp}$ nel caso di PFD con architettura XOR, oppure una corrente alternata a tre livelli $+I_{cp}$, 0 e $-I_{cp}$ nel caso di PFD TRISTATE. Le architetture della pompa di carica sono generalmente affette da problematiche di dead zone, infatti le non idealità dei componenti attivi che la costituiscono (esempio interruttori MOS) determinano una zona di indeterminazione dell'uscita della pompa di carica nel caso in cui le fasi del segnale di riferimento Ref(t) e di quello retroazionato Div(t) sono abbastanza vicine. Questa mancanza di correzione del segnale di errore del sistema in retroazione determina errore di fase in uscita.

Loop Filter

Il Loop Filter è un filtro passa basso che controlla la dinamica della retroazione. Da questo dipende l'ordine della funzione di trasferimento del sintetizzatore di frequenza e la posizione dei poli e degli zeri.

VCO

L'oscillatore controllato in tensione VCO presenta una capacità variabile con la tensione di controllo, da questa dipende la frequenza di oscillazione di riposo, detta frequenza di *free running*, la quale corrisponderà alla frequenza del segnale di uscita f_{out}. Se si considera come ingresso la tensione di controllo e come uscita la fase $\varphi_{out}(t)$, il VCO si comporta come un integratore.

$$\varphi_{out}(t) = 2\pi K_V \int_{-\infty}^{t} v_{in}(t')\,dt' \qquad (1)$$

con K_V costante caratteristica del VCO.

Divisore di frequenza

Il modo in cui viene controllato il divisore di frequenza è differente a seconda della natura del rapporto N fra la f_{out} e la f_{Ref}. Considerando solo divisori programmabili, ovvero divisori in cui N è variabile mediante un segnale di controllo, nel caso di rapporto N intero, il divisore di frequenza per N è una macchina a stati finiti che genera un solo fronte di salita ogni N fronti in ingresso con N che rimane stabile nell'uso del divisore. In questo modo la f_{out} può essere solo un multiplo della f_{ref}. Nel caso di N frazionario il divisore è costituito da un contatore programmabile pilotato da un modulatore Σ-Δ che determina una sequenza apparentemente regolare che consente di commutare rapidamente fra diversi rapporti interi, questo consente di spostare il rumore di quantizzazione a frequenze più elevate e quindi facilmente filtrabile.

Capitolo 1.2: modello del PLL nel dominio del tempo

Si vuole determinare un modello nel dominio del tempo del sintetizzatore di frequenza a rapporto frazionario N con modulatore Σ-Δ, che consideri anche gli effetti della variazione di N.

Si suddivide il problema determinando un modello per ogni componente del sintetizzatore, come ampiamente dimostrato in [2].

Phase Frequency Detector

Nel caso di PFD con architettura TRISTATE, l'uscita E(t) è rappresentata da un treno di impulsi la cui larghezza è funzione del valore assoluto della differenza di fase fra Ref(t) e Div(t) e segno uguale a quello della differenza. Associando alla fase di Ref(t) e alla fase di Div(t) le due sequenze discrete $\varphi_{\mathrm{Ref}}[k]$ e $\varphi_{\mathrm{Div}}[k]$, il riferimento a fase costante $\varphi_{\mathrm{Ref}}[k]$ è nullo e gli istanti in cui si hanno i fronti di salita se l'impulso è positivo e i fronti di discesa se l'impulso è negativo, avvengono ogni $T=1/f_{\mathrm{Ref}}$. Definita la larghezza del singolo impulso $\Delta t[k]$, questa sarà direttamente proporzionale alla differenza di fase secondo la formula:

$$\Delta t[k] = \frac{T}{2\pi}(\varphi_{\mathrm{Ref}}[k] - \varphi_{\mathrm{Div}}[k]) \qquad (2)$$

Supponendo che la larghezza degli impulsi sia sufficientemente breve rispetto alla costante di tempo dominante del Loop Filter, la convoluzione della risposta impulsiva del PFD con la risposta impulsiva del Loop Filter $h(t)$ è approssimabile alla convoluzione di una successione

di delta di Dirac rappresentativi della risposta impulsiva del PFD con la stessa $h(t)$. L'approssimazione è valida nel momento in cui la frequenza di riferimento è sufficientemente maggiore della banda del Loop Filter, quindi tanto più nel caso dei sintetizzatori Σ-Δ. Il segnale $E(t)$ è di conseguenza rappresentabile come una successione di delta di Dirac aventi ampiezza pari all'area dei corrispondenti impulsi.

$$\mathrm{E}(t) = \sum_{k=-\infty}^{+\infty} \Delta t[k]\delta(t - kT) \qquad (3)$$

Il caso di architettura XOR del PFD può essere ricondotto al modello descritto per l'architettura TRISTATE. Il segnale $E(t)$ può essere scomposto in due componenti: un'onda quadra $E_{spur}(t)$ indipendente dalla differenza di fase fra $Ref(t)$ e $Div(t)$ e un treno di impulsi a 3 livelli come nel caso precedente. In questo caso la larghezza degli impulsi è data da:

$$\Delta t[k] = \frac{T}{2\pi}(\varphi_{\mathrm{Ref}}[k] - \varphi_{\mathrm{Div}}[k] - \pi) \qquad (4)$$

Con considerazioni analoghe al PFD TRISTATE si ha:

$$\mathrm{E}(t) = \sum_{k=-\infty}^{+\infty} 2\Delta t[k]\delta(t - kT) + \mathrm{E_{spur}(t)} \qquad (5)$$

Quindi non considerando l'offset pari a π e ignorando la componente fornita dall'onda quadra $E_{spur}(t)$, il modello è identico a quello TRISTATE eccetto un guadagno pari a 2.

VCO

Il modello del VCO è dato dall'equazione che esprime la variazione di fase dell'uscita del VCO espressa in (1), mentre il valore complessivo della fase dell'uscita del VCO è:

$$\varphi_{VCO}(t) = 2\pi f_{nom} t + \varphi_{out}(t) \qquad (6)$$

dove f_{nom} è la frequenza nominale che si vuole sintetizzare, differente dalla frequenza di uscita del sintetizzatore f_{out} per effetto dei rumori.

Pompa di carica

La pompa di carica ha come modello un semplice fattore di scala I_{cp}.

Loop Filter

Il Loop Filter invece è caratterizzato dalla risposta impulsiva $h(t)$.

Divisore di frequenza

Il modello del divisore proposto da Perrott in [2] è basato sull'osservazione che i fronti di salita di $E_{spur}(t)$ avvengono ogni $2\pi N[k]$ radianti, con $N[k]$ valore assunto dal rapporto N al k-esimo ciclo, quindi:

$$\varphi_{VCO}(t[k] + \Delta t[k]) - \varphi_{VCO}(t[k-1] + \Delta t[k-1]) = 2\pi N[k-1] \qquad (7)$$

Considerando la (6) e che $t[k] - t[k\text{-}1] = T$ e definendo il rapporto nominale desiderato come il rapporto fra la frequenza nominale che si vuole sintetizzare e la frequenza del segnale di riferimento $N_{nom} = f_{nom} / f_{ref} = f_{nom} T$, si ottiene:

$$\begin{aligned} 2\pi f_{nom}(\Delta t[k] - \Delta t[k-1]) \\ = 2\pi(N[k-1] - N_{nom}) - \varphi_{out}(t[k] + \Delta t[k]) \\ - \varphi_{out}(t[k-1] + \Delta t[k-1]) \end{aligned} \qquad (8)$$

Sommando tutti i campioni del tempo positivi fino a k, assumendo condizioni iniziali nulle e definendo $n[k]$ come la differenza fra il valore del rapporto N al k-esimo ciclo e il valore nominale desiderato $n[k] = N[k] - N_{nom}$, si ottiene la larghezza dell'impulso di Div(t):

$$\Delta t[k] = \left(\frac{T}{2\pi}\right)\left(\frac{1}{N_{nom}}\right)\left(2\pi \sum_{m=1}^{k} n[m-1] - \varphi_{out}[k]\right) \qquad (9)$$

Sostituendo il termine $\Delta t[k]$ che compare in (2) e considerando che $\varphi_{Ref}[k]$ è nullo, si ottiene il modello del divisore:

$$\varphi_{Div}[k] = \frac{1}{N_{nom}}\left(-2\pi \sum_{m=1}^{k} n[m-1] + \varphi_{out}[k]\right) \qquad (10)$$

Definiti α il fattore del PDF pari ad 1 per architettura TRISTATE e pari a 2 per architettura XOR e $\varphi_n[k]$ l'uscita dell'accumulatore contenuto nel blocco divisore, si ottiene il modello nel dominio del tempo del sintetizzatore rappresentato in Figura 4:

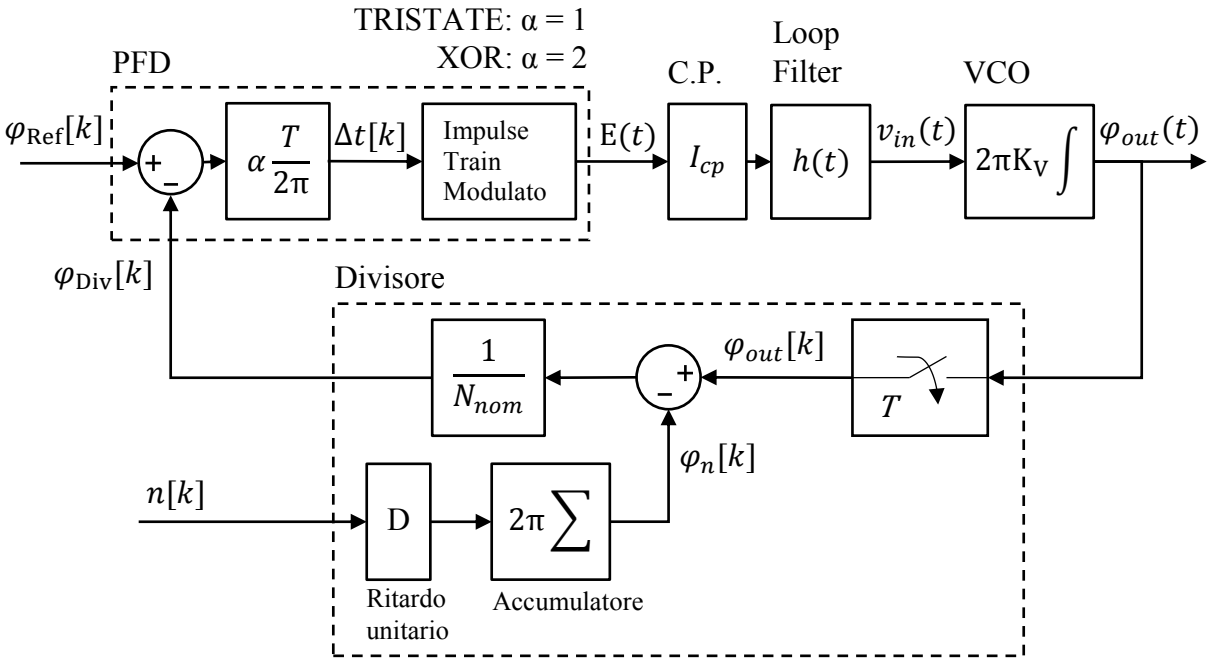

Figura 4: schema a blocchi modello nel dominio del tempo del sintetizzatore di frequenza delta-sigma frazionario-N

Osservando il modello proposto in [2], l'uscita del divisore $\varphi_{\mathrm{Div}}[k]$ è data da due contributi, il primo è la versione campionata della variazione di fase in uscita $\varphi_{out}(t)$ divisa per N_{nom}, il secondo è l'integrazione della variazione numerica di $n[k]$. Dunque, il divisore ha un ingresso continuo nel dominio del tempo e un'uscita discreta nel dominio del tempo. Il segnale E(t) in uscita dal PFD è un segnale discreto nei valori rappresentato come una successione di impulsi con larghezza proporzionale al segnale errore del sistema in retroazione. La pompa di carica e il Loop Filter, convertendo il segnale discreto nei valori in un segnale continuo nel dominio del tempo $v_{in}(t)$, effettuano di fatto una conversione digitale/analogica D/A.

Capitolo 1.3: modello del PLL nel dominio delle frequenze

Il modello del sintetizzatore di frequenze nel dominio delle frequenze è basato su un'approssimazione di pseudocontinuità, avvalorata dalla presenza del filtro passa basso nel ramo di feed-forward dell'anello (Loop Filter).

Come determinato in (3), il segnale errore E(t) è esprimibile come una successione di delta di Dirac aventi ampiezza pari all'area dei corrispondenti impulsi, quindi è trattabile come un segnale campionato nel dominio del tempo.

In generale, dato un segnale $x(t)$ campionato nel tempo con periodo T e convertito in una successione di delta di Dirac è esprimibile come:

$$\hat{x}(t) = \sum_{k=-\infty}^{+\infty} x(kT)\delta(t - kT) \qquad (11)$$

Applicando la trasformata di Fourier tempo discreta al segnale $\hat{x}(t)$, si ottiene:

$$\hat{X}(f) = \frac{1}{T}\sum_{k=-\infty}^{+\infty} X\left(f - \frac{k}{T}\right) \qquad (12)$$

Quindi, la trasformata di Fourier di un segnale campionato è equivalente alla ripetizione periodica della trasformata di Fourier del segnale non campionato, scalata di un fattore $1/T$.

L'approssimazione di pseudo-continuità assume che il contenuto armonico di $x(t)$ sia confinato all'interno della banda $[-1/(2T), 1/(2T)]$, in modo che non si manifesti il fenomeno dell'*aliasing,* ovvero sovrapposizione fra le repliche della trasformata di $x(t)$ nel dominio delle frequenze. Un'altra supposizione dell'approssimazione di pseudocontinuità è che all'interno del sistema ci sia un filtro passa basso in grado di isolare una singola replica della trasformata di $x(t)$ in banda base. Questo implica che la banda del filtro passa basso debba essere inferiore a $1/(2T)$. Nel sintetizzatore, questo compito è svolto dal Loop Filter.

Il modello del sintetizzatore nel dominio delle frequenze, a questo punto, è semplicemente ottenibile applicando la trasformata di Fourier per i blocchi tempo continui, la trasformata Z per i blocchi tempo discreti e l'approssimazione di pseudocontinuità per i blocchi che effettuano il campionamento nel dominio del tempo.

Nel modello nel dominio delle frequenze, i segnali sono indicati utilizzando una notazione che riporta una dipendenza dal tempo. Infatti, questi segnali sono stocastici e con ciò si vuole

rimarcare che non sono modellabili attraverso la loro trasformata di Fourier, bensì attraverso la loro densità spettrale di potenza.

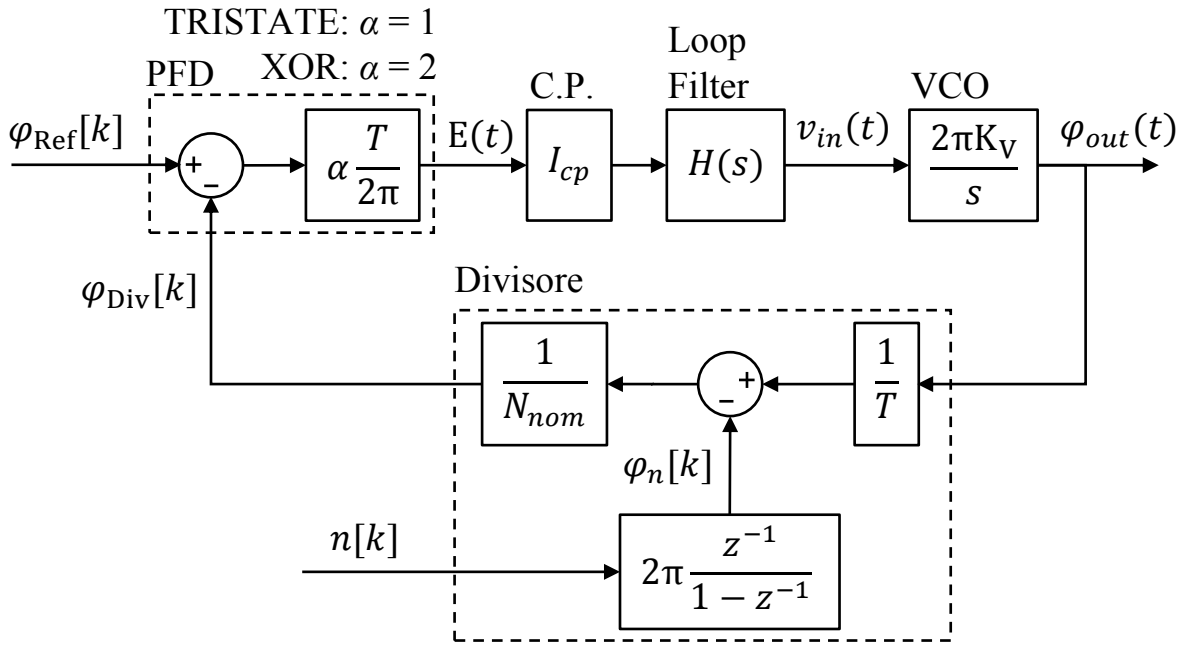

Figura 5: schema a blocchi modello nel dominio delle frequenze del sintetizzatore di frequenza delta-sigma frazionario-N

Capitolo 1.4: filtri Butterworth, Bessel, Chebyshev ed ellittici

Il comportamento complessivo del PLL, ovvero il modo in cui la fase di uscita reagisce alle variazioni della fase d'ingresso, è definito mediante delle specifiche di progetto.

Le specifiche di progetto sono poste mediante una funzione di trasferimento ed è ragionevole che siano definite attraverso delle forme standard, quali sono i filtri di Butterworth, di Bessel, di Chebyshev ed ellittici, caratterizzati da differenti disposizioni dei poli e degli zeri della stessa funzione di trasferimento.

Una generica funzione di trasferimento di un sistema può essere espressa nella forma fattorizzata:

$$TF(s) = \frac{\rho \prod_i (s + z_{ri}) \prod_i (s^2 + 2\zeta_i \omega_{zi} s + {\omega_{zi}}^2)}{s^g \prod_i (s + p_{ri}) \prod_i \left(s^2 + 2\xi_i \omega_{pi} s + {\omega_{pi}}^2\right)} \qquad (13)$$

con: ρ costante di trasferimento, g molteplicità del polo nullo o tipo, $-z_{ri} \neq 0$ zeri reali, $-p_{ri} \neq 0$ poli reali, ω_{zi} pulsazioni naturali delle coppie di zeri complessi coniugati, ω_{pi} pulsazioni naturali delle coppie di poli complessi coniugati, ζ_i smorzamenti delle coppie di zeri complessi coniugati, ξ_i smorzamenti delle coppie di poli complessi coniugati [15].

Un filtro ideale, caratterizzato da guadagno unitario in banda passante e attenuazione infinita in banda attenuata, non è fisicamente realizzabile. Esistono però diverse approssimazioni con modelli analogici standard che definiscono la forma del filtro, in grado di soddisfare le specifiche di progetto del filtro definite in Figura 6.

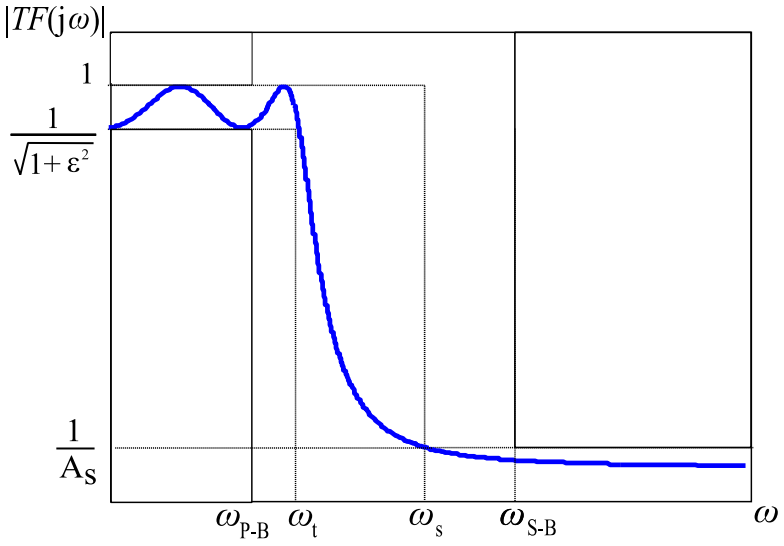

Figura 6: Maschera delle specifiche di progetto di un filtro prototipo passa basso

dove:

- la banda [0 , $\omega_{\text{P-B}}$] è la banda passante;
- la banda [$\omega_{\text{S-B}}$, $+\infty$] è la banda attenuata;
- la banda [$\omega_{\text{P-B}}$, $\omega_{\text{S-B}}$] è la banda di transizione;
- ω_t è la reale pulsazione di banda passante;
- ω_s è la reale pulsazione di banda attenuata;
- $1/\sqrt{1+\varepsilon^2}$ è la specifica sul *ripple* (oscillazione) massimo in banda passante;
- $1/A_S$ è la specifica sull'attenuazione minima in banda attenuata.

Il ripple massimo in banda passante δ_1 è legato alla sua specifica dalla relazione:

$$1-\delta_1 = \frac{1}{\sqrt{1+\varepsilon^2}} \qquad (14)$$

mentre la minima attenuazione in banda attenuata δ_2 è legata alla sua specifica dalla relazione:

$$\delta_2 = \frac{1}{A_s} \qquad (15)$$

In questo progetto ci si sofferma su cinque modelli analogici standard che approssimano la risposta del filtro *TF*(*s*): Butterworth, Bessel, Chebyshev di tipo I, Chebyshev di tipo II e ellittici.

Filtro di Butterworth solo poli

Il filtro di Butterworth con solo poli presenta una funzione di trasferimento del tipo:

$$TF(s) = \frac{1}{B^{(m)}\left(\frac{s}{\omega_0}\right)} \qquad |TF(\mathrm{j}\omega)|^2 = \frac{1}{1+\omega^{2m}} \qquad (16)$$

con $B^{(m)}(s/\omega_0)$ polinomio di Butterworth, ω_0 pulsazione naturale comune a tutti i poli, ω pulsazione, m ordine del filtro.

La funzione di trasferimento fa parte della famiglia di funzioni monotone in banda attenuata. I filtri di Butterworth sono massimamente piatti in banda passante, non hanno fase lineare e richiedono un elevato ordine m per garantire una buona ripidità della regione di transizione tra banda passante e banda attenuata.

I poli p_i sono distribuiti su una circonferenza di raggio pari alla pulsazione naturale comune a tutti i poli. Quindi:

$$p_i = \omega_p e^{j\theta_i} \qquad \theta_i = \frac{\pi(2m_1+1)}{2m} \qquad \theta_i = \frac{m_1\pi}{m} \qquad (17)$$

m pari, m_1=0,1, … , $2m$-1 m dispari, m_1=0,1, … , $2m$

Considerando ω_p unitario, si ottiene il prototipo passa basso, caratterizzato da un diagramma dei poli e degli zeri nel piano s e una risposta al gradino unitario riportati sotto:

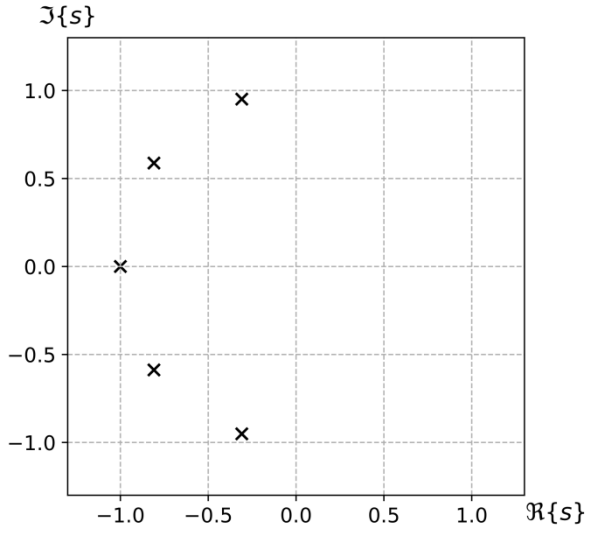

Figura 7: Diagramma dei poli e degli zeri nel piano s del prototipo Butterworth di 5°ordine

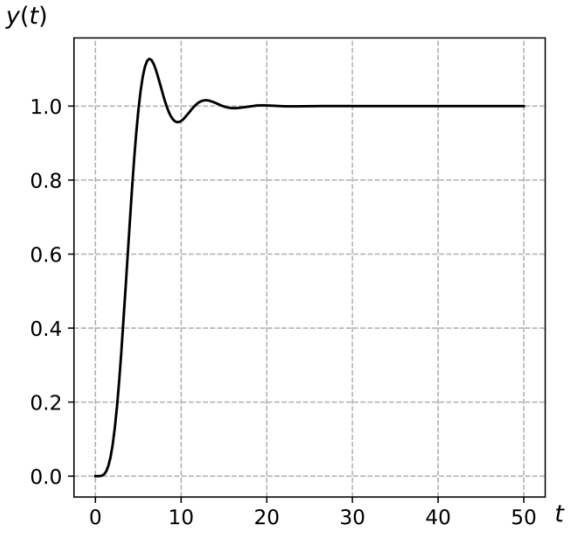

Figura 8: Risposta al gradino unitario del prototipo Butterworth di 5°ordine

Filtro di Bessel solo poli

Il filtro di Bessel con solo poli presenta una funzione di trasferimento del tipo:

$$TF(s) = \frac{\theta_m(0)}{\theta_m\left(\frac{s}{\omega_0}\right)} \qquad (18)$$

$$\theta_m(s) = \sum_{m_1=0}^{m} \alpha_{m1} s^{m_1} \qquad \alpha_{m1} = \frac{(2m - m_1)!}{2^{m-m1} m_1! \, (m - m_1)!} \qquad m_1 = 0, 1, \dots, m \qquad (19)$$

con $\theta_m(s)$ polinomio di Bessel, m ordine del filtro, ω pulsazione, ω_0 frequenza di taglio del filtro definita in modo asintotico e non la classica frequenza di taglio a -3dB.

I filtri di Bessel hanno la massima linearità nella risposta in fase nella banda passante.

I poli p_i del filtro di Bessel sono quindi le radici del polinomio di Bessel $\theta_m(s/\omega_0)$.

Considerando ω_p unitario, si ottiene il prototipo passa basso, caratterizzato da un diagramma dei poli e degli zeri nel piano s e una risposta al gradino unitario riportati sotto:

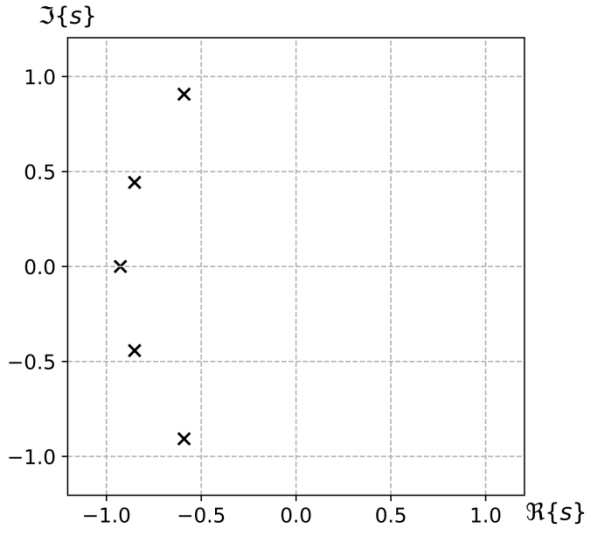

Figura 9: Diagramma dei poli e degli zeri nel piano s del prototipo Bessel di 5°ordine

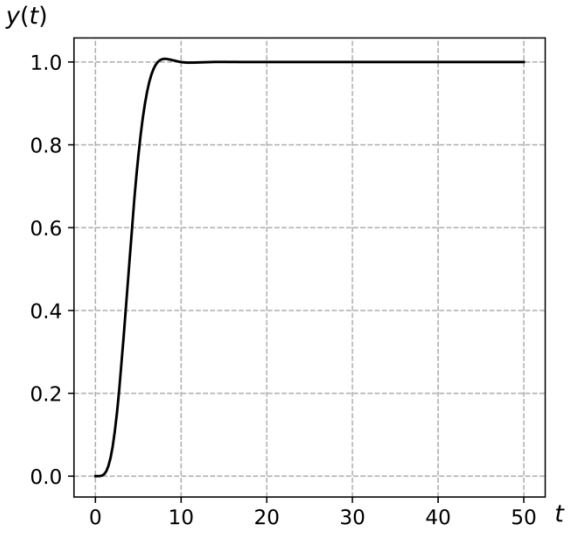

Figura 10: Risposta al gradino unitario del prototipo Bessel di 5°ordine

Filtro di Chebyshev di tipo I solo poli

Il filtro Chebyshev di tipo I con solo poli presenta una funzione di trasferimento del tipo:

$$|TF(\mathrm{j}\omega)|^2 = \frac{1}{1+\varepsilon^2 \mathrm{T}_m^2\left(\frac{\omega}{\omega_0}\right)} \qquad \mathrm{T}_m(\omega) = \cosh(m\cosh^{-1}(\omega)) \qquad (20)$$

con $\mathrm{T}_m(\omega)$ polinomio di Chebyshev, m ordine del filtro, ω pulsazione, ω_0 frequenza di taglio del filtro definita in modo asintotico e non la classica frequenza di taglio a -3dB, ε costante legata all'ampiezza del ripple in banda passante.

La funzione di trasferimento fa parte della famiglia delle funzioni monotone in banda attenuata, ma a differenza dei filtri Butterworth presenta un ripple in banda passante. I filtri Chebyshev di tipo I hanno una risposta in fase ben lontana dalla linearità, ma hanno una rapidità di transizione tra banda passante e banda attenuata maggiore rispetto ai filtri Butterworth e questo consente la riduzione dell'ordine del filtro.

I poli p_i giacciono su un'ellissi, i cui parametri dipendono da m, ε e ω_0:

$$C = \frac{\omega_0}{2}\left\{\left[\sqrt{1+\varepsilon^{-2}}+\varepsilon^{-1}\right]^{\frac{1}{m}} + \left[\sqrt{1+\varepsilon^{-2}}+\varepsilon^{-1}\right]^{-\frac{1}{m}}\right\}$$

$$D = \frac{\omega_0}{2}\left\{\left[\sqrt{1+\varepsilon^{-2}}+\varepsilon^{-1}\right]^{\frac{1}{m}} - \left[\sqrt{1+\varepsilon^{-2}}+\varepsilon^{-1}\right]^{-\frac{1}{m}}\right\}$$

$$p_i = -C\sin\left[\left(\frac{2m_1-1}{2m}\right)\pi\right] + \mathrm{j}D\cos\left[\left(\frac{2m_1-1}{2m}\right)\pi\right] \qquad m_1 = 1, 2, \ldots, m \qquad (21)$$

Considerando ω_p unitario, si ottiene il prototipo passa basso, caratterizzato da un diagramma dei poli e degli zeri nel piano s e una risposta al gradino unitario riportati sotto, validi per un determinato valore di attenuazione minima in banda passante:

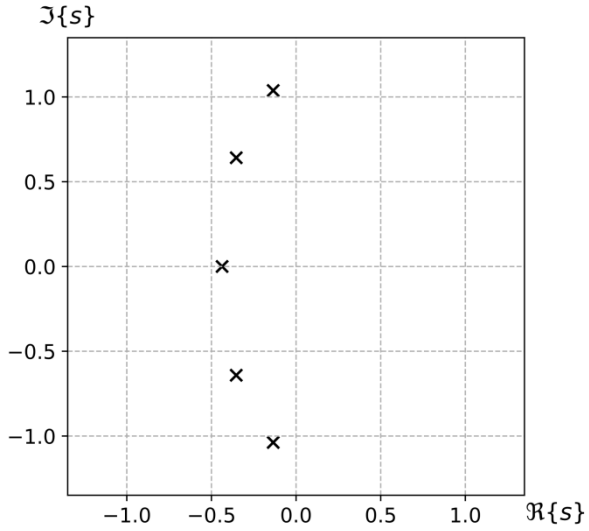

Figura 11: Diagramma dei poli e degli zeri nel piano s del prototipo Chebyshev I del 5°ordine con ripple massimo in banda passante di 0,25dB

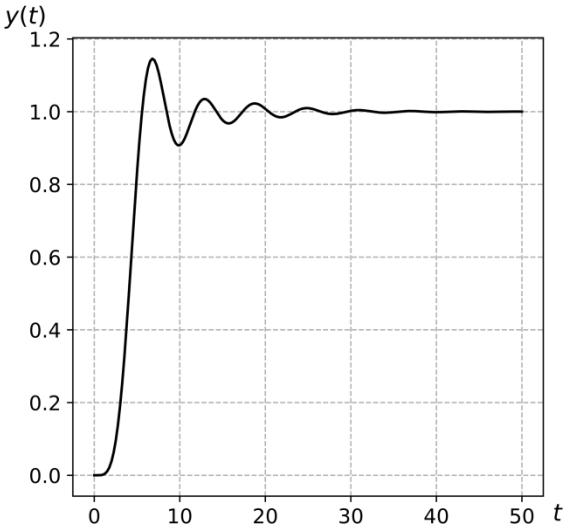

Figura 12: Risposta al gradino unitario del prototipo Chebyshev I del 5°ordine con ripple massimo in banda passante di 0,25dB

Filtro di Chebyshev di tipo II

Il filtro Chebyshev di tipo II o inverso presenta una funzione di trasferimento del tipo:

$$|TF(\mathrm{j}\omega)|^2 = \frac{1}{1 + \frac{A_s^2 - 1}{\mathrm{T}_m^2\left(\frac{\omega_{S-B}}{\omega}\right)}} \qquad \mathrm{T}_m(\omega) = \cosh(m \cosh^{-1}(\omega)) \qquad (22)$$

con $\mathrm{T}_m(\omega)$ polinomio di Chebyshev, m ordine del filtro, ω pulsazione, $\omega_{\text{S-B}}$ pulsazione che specifica l'inizio della *stop band* (banda attenuata), A_s costante legata all'ampiezza dell'attenuazione minima in banda attenuata.

La funzione di trasferimento, avendo degli zeri, non è monotona in banda attenuata, non presenta ripple in banda passante mentre presenta ripple in banda attenuata. I filtri Chebyshev di tipo II hanno una risposta in fase ben lontana dalla linearità, ma hanno una rapidità di transizione tra banda passante e banda attenuata maggiore rispetto ai filtri Butterworth e questo consente la riduzione dell'ordine del filtro.

I poli p_i giacciono su un'ellissi e hanno valore pari all'inverso dei poli indicati in (21) per i filtri Chebyshev di tipo I.

In questo caso però ci sono anche le coppie di zeri complessi coniugati a parte reale nulla:

$$z_{\mathrm{i}} = \mathrm{j}\frac{\omega_0}{\cos\left(\frac{\pi}{2}\frac{2m_1 - 1}{m}\right)} \qquad m_1 = 1, 2, \ldots, m \qquad (23)$$

con ω_0 pulsazione di taglio del filtro definita in maniera asintotica considerando solo i poli.

Considerando ω_p unitario, si ottiene il prototipo passa basso, caratterizzato da un diagramma dei poli e degli zeri nel piano *s* e una risposta al gradino unitario riportati sotto, validi per un determinato valore di attenuazione minima in banda attenuata:

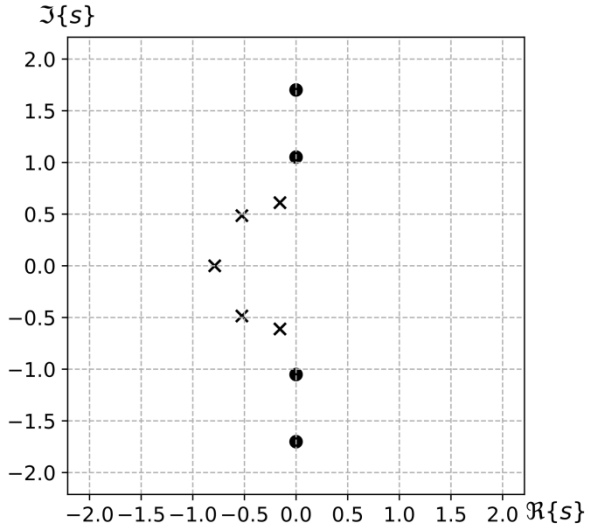

Figura 13: Diagramma dei poli e degli zeri nel piano s del prototipo Chebyshev II del 5°ordine con attenuazione minima in banda attenuata di 40dB

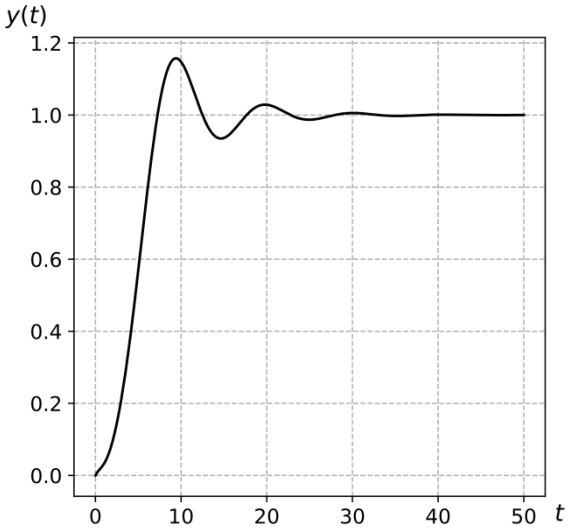

Figura 14: Risposta al gradino unitario del prototipo Chebyshev II del 5°ordine con attenuazione minima in banda attenuata di 40dB

Filtro ellittico

Il filtro ellittico o di Cauer presenta una funzione di trasferimento del tipo:

$$|TF(\mathrm{j}\omega)|^2 = \frac{1}{1 + \varepsilon^2 \mathrm{R}_m^2\left(\upsilon, \frac{\omega}{\omega_0}\right)} \qquad (24)$$

con $R_m(\upsilon, \omega/\omega_0)$ funzione razionale di Chebyshev, ω pulsazione, ω_0 frequenza di taglio del filtro definita in modo asintotico e non la classica frequenza di taglio a -3dB, ε costante legata all'ampiezza del ripple in banda passante e υ fattore di selettività.

La funzione di trasferimento consente di avere un ripple equivalente in banda passante e in banda attenuata, ma ha una risposta in fase estremamente non lineare.

I poli p_i giacciono su un'ellissi, mentre gli zeri sono coppie di zeri complessi coniugati a parte reale nulla. La posizione dei poli e degli zeri è determinata grazie a funzioni jacobiane ellittiche.

Considerando ω_p unitario, si ottiene il prototipo passa basso, caratterizzato da un diagramma dei poli e degli zeri nel piano *s* e una risposta al gradino unitario riportati sotto, validi per un determinato valore di ripple massimo in banda passante e di attenuazione minima in banda attenuata:

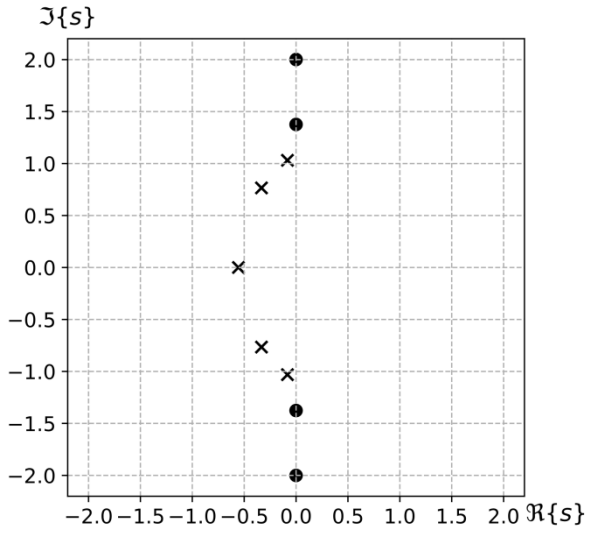

Figura 15: Diagramma dei poli e degli zeri nel piano s del prototipo ellittico del 5°ordine con ripple massimo in banda passante di 0,25dB e attenuazione minima in banda attenuata di 40dB

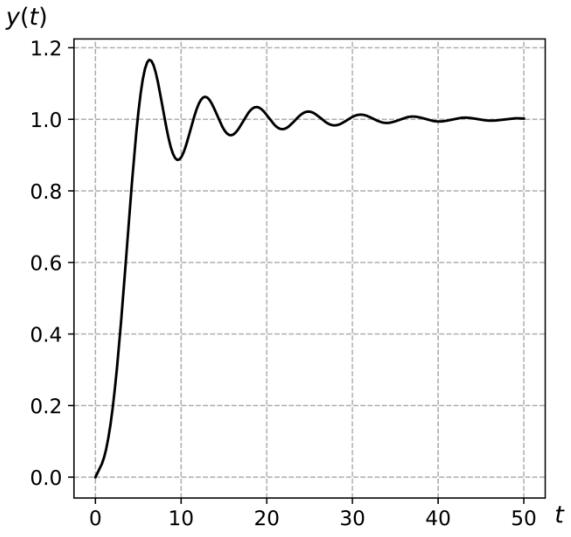

Figura 16: Risposta al gradino unitario del prototipo ellittico del 5°ordine con ripple massimo in banda passante di 0,25dB e attenuazione minima in banda attenuata di 40dB

Il grafico riportato in Figura 17 confronta il modulo delle funzioni di trasferimento (espresso in decibel) dei prototipi passa basso aventi ω_p unitario per le forme suddette. Questo è valido solo per una determinata specifica sul ripple massimo in banda passante e sull'attenuazione minima in banda passante:

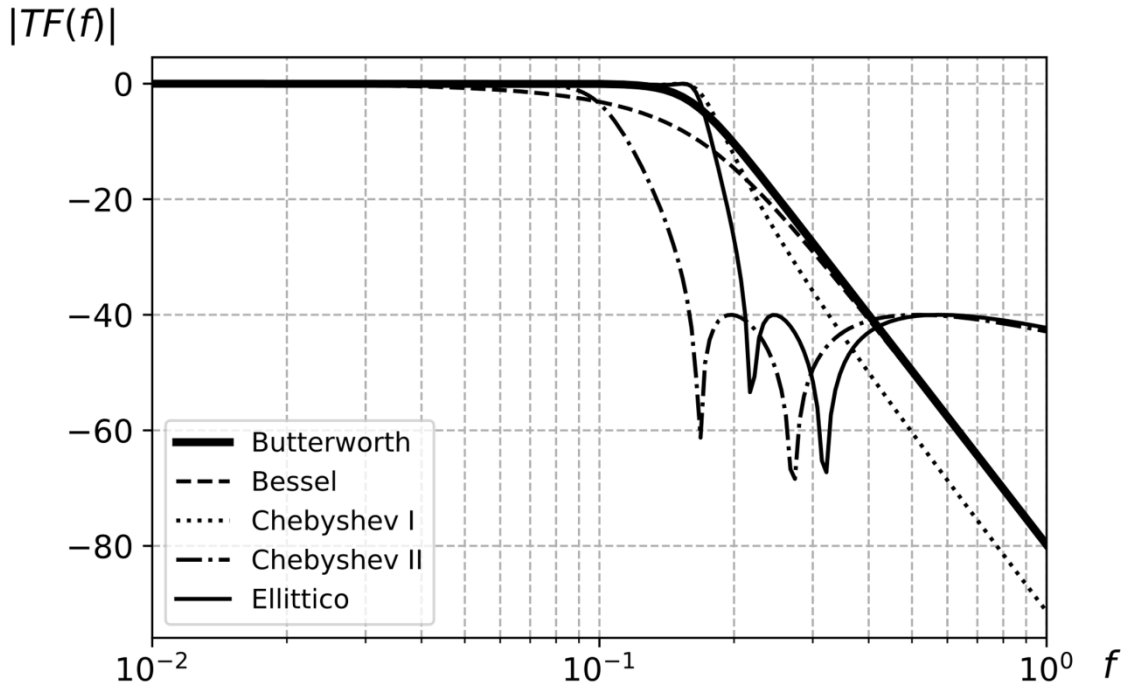

Figura 17: Modulo della funzione di trasferimento dei prototipi passa basso del 5°ordine di Butterworth, di Bessel, di Chebyshev I con ripple massimo in banda passante di 0,25dB, di Chebyshev II con attenuazione minima in banda attenuata di 40dB ed ellittico con ripple massimo in banda passante di 0,25dB e attenuazione minima in banda attenuata di 40dB.

Da notare che la specifica sull'attenuazione minima in banda attenuata influenza molto la posizione della frequenza di taglio del filtro.

Capitolo 2

PARAMETRIZZAZIONE DEL MODELLO DEL PLL

Per consentire l'analisi delle performance dinamiche e del rumore in uscita del sintetizzatore di frequenze che incorpora il modulatore Σ-Δ, è possibile sfruttare un modello matematico gestibile via software. Questo approccio, basato sulla simulazione del modello nel dominio delle frequenze, consente di verificare la stabilità del sistema valutando direttamente la risposta del sistema ed è quindi differente dal classico calcolo del margine di fase e del margine di ampiezza [16].

Il modello consente infatti di progettare il PLL impostando le caratteristiche dinamiche ad anello chiuso, anziché dedurre il comportamento dell'anello chiuso dall'analisi ad anello aperto.

La chiave di questo approccio è quella di progettare prima la dinamica ad anello chiuso secondo specifiche desiderate, e quindi calcolare la risposta ad anello aperto necessaria per realizzare il desiderato comportamento ad anello chiuso

Il modello matematico del PLL è definito mediante una serie di parametri che rappresentano le variabili di controllo del sistema.

Il PLL è un sistema in retroazione rappresentabile mediante lo schema a blocchi sottostante:

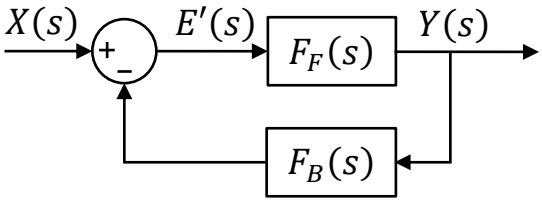

Figura 18: Generico sistema retroazionato

dove:

- $X(s)$ è la trasformata di Laplace della fase del segnale in ingresso al sistema;
- $Y(s)$ la trasformata di Laplace della fase del segnale in uscita dal sistema;
- $E'(s)$ la trasformata di Laplace del segnale errore;

- $F_F(s)$ la funzione di trasferimento del blocco diretto (*feed-forward*);
- $F_B(s)$ la funzione di trasferimento del blocco in retroazione (*feedback*).

Il guadagno ad anello aperto è:

$$A(s) = F_F(s)F_B(s) \tag{25}$$

In particolare, nel PLL, il blocco $F_F(s)$ contiene il prodotto delle funzioni di trasferimento del PFD, della pompa di carica, del Loop Filter e del VCO, mentre $F_B(s)$ non ha effetti dinamici, poiché costituito dal solo divisore di frequenza.

La funzione di trasferimento del sistema retroazionato si ottiene dal sistema:

$$\begin{cases} E'(s) = X(s) - F_B(s)Y(s) \\ Y(s) = E'(s)F_F(s) \end{cases} \tag{26}$$

da cui:

$$Y(s) = \frac{F_F(s)}{1 + F_F(s)F_B(s)}X(s) \tag{27}$$

Quindi la funzione di trasferimento del sistema retroazionato è:

$$TF(s) = \frac{Y(s)}{X(s)} = \frac{F_F(s)}{1 + F_F(s)F_B(s)} = \frac{F_F(s)}{1 + A(s)} \tag{28}$$

Nel caso del PLL, se si considera come ingresso la fase del segnale di riferimento, nella variabile s=j2πf, f non è intesa come frequenza del segnale in uscita del PLL, ma la frequenza alla quale la fase del riferimento è cambiata. In altri termini è la frequenza di *offset* o deviazione rispetto alla frequenza di uscita nominale del PLL.

Capitolo 2.1: Funzione di trasferimento descrittiva del PLL

Al fine di ottenere un modello del PLL semplice e del quale è possibile studiare la stabilità del sistema senza determinare il margine di fase e il margine di ampiezza, Perrott in [2] propone di utilizzare una funzione di trasferimento $G(s)$ differente da quella indicata in (28):

$$G(s) = \frac{A(s)}{1 + A(s)} \tag{29}$$

e quindi legata a $TF(s)$ indicata in (28) secondo la relazione:

$$G(s) = TF(s)F_B(s) \tag{30}$$

Essendo il blocco $F_B(s)$ una costante moltiplicativa senza dinamica, $G(s)$ ha la stessa risposta in frequenza di $TF(s)$, di conseguenza la scelta di $G(s)$ come modello è consistente.

La funzione di trasferimento $TF(s)$, e di conseguenza anche $G(s)$, deve essere necessariamente di tipo passa basso. La banda di $G(s)$ indica infatti l'abilità del PLL di tracciare le variazioni di frequenza e di fase e la rapidità con la quale si ottiene l'aggancio alla frequenza del segnale di riferimento in ingresso.

È da notare che:

$$\begin{cases} G(s) \to 1 & \text{quando} \quad A(s) \to +\infty \\ G(s) \to 0 & \text{quando} \quad A(s) \to 0 \end{cases} \tag{31}$$

Questo implica che alle basse frequenze, $A(s)$ deve essere molto elevato, mentre ad alte frequenze deve ridursi. Per questo motivo $A(s)$ deve avere almeno un polo. Grazie alla retroazione, la banda di $G(s)$ è significativamente più grande rispetto a quella di $A(s)$. La frequenza di taglio di $G(s)$ è approssimativamente la frequenza alla quale $A(s)$ ha guadagno unitario e per frequenze maggiori della frequenza di taglio $G(s) \approx A(s)$.

Capitolo 2.2: Definizione delle specifiche del modello del PLL

Il modello del PLL proposto da Perrott in [2] è definito secondo alcune specifiche che influenzano le prestazioni dinamiche del sistema in retroazione $G(s)$.

Il processo di design del PLL è scomponibile in due passaggi:

1) Design della $G(s)$ per ottenere le caratteristiche desiderate del sistema retroazionato;

2) Design della $A(s)$ che determina la $G(s)$ desiderata.

Le specifiche sono:

- ***Order*** o equivalentemente $\boldsymbol{m}$, definito come indicatore del *roll-off* di $G(s)$ e non come numero di variabili di stato indipendenti del sistema. Il roll-off indica di quanto cala il modulo della funzione di trasferimento ogni decade di frequenza e ha come unità di misura il dB/dec. Nel modello proposto da Perrott, $G(s)$ è ottenuta a partire da un filtro prototipo, al quale vengono poi aggiunti poli e zeri al fine di ottenere un Loop Filter $H(s)$ in grado di fornire la $G(s)$ desiderata stabile anche in presenza di non idealità. Nella definizione di m nel modello di Perrott, questo coincide che il numero di poli (non parassiti) della funzione prototipo, quindi se la funzione prototipo è una funzione di trasferimento con solo poli e ha un roll-off pari a -20dB/dec, allora $m = 1$, se ha un roll-off di -40dB/dec allora $m = 2$. Nel caso in cui la funzione prototipo sia di tipo Chebyshev II o ellittica, m è legato al roll-off che si avrebbe considerando solo i poli, ovvero al roll-off della caratterista di trasferimento valutata fra la frequenza di taglio e la minima frequenza naturale degli zeri della caratteristica di Chebyshev II o ellittica. Questa scelta è legata al fatto che per un designer di filtri, il roll-off è una caratteristica più interessante rispetto al numero di variabili di stato indipendenti.

- $\boldsymbol{f_0}$, definita come la banda asintotica della funzione prototipo di $G(s)$ o frequenza di taglio o di *cut-off*. La banda asintotica corrisponde alla frequenza alla quale la caratteristica asintotica del roll-off della risposta frequenziale del sistema attraversa l'asintoto della funzione di trasferimento nella regione di banda passante. La scelta di utilizzare la banda asintotica invece della classica frequenza di taglio a -3dB consente di rendere il modello molto più veloce nei calcoli, in quanto facilmente determinabile dai poli della funzione prototipo di $G(s)$:

$$f_0 = \frac{1}{2\pi}\left(\prod_{i=0}^{m-1} p_i\right)^{\frac{1}{m}} \quad (32)$$

con p_i poli reali o complessi (nel caso saranno a coppie di poli complessi coniugati)

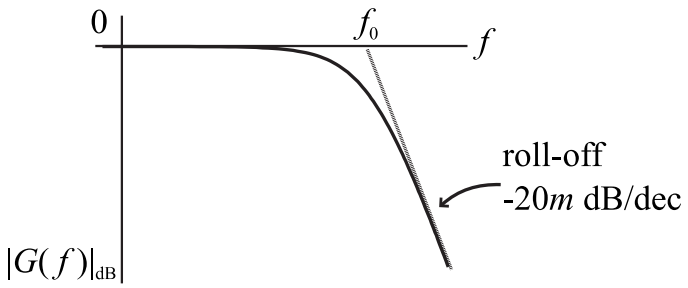

Figura 19: Definizione di frequenza di taglio e di ordine di G(s)

- ***shape*** o tipo di filtro, è il tipo di filtro utilizzato come prototipo di $G(s)$ e indica la forma della funzione di trasferimento e la corrispondente risposta al gradino date le specifiche di ordine m e frequenza di taglio f_0. Può essere Butterworth, Bessel, Chebyshev I, Chebyshev II o ellittico.

- ***PLL type,*** è il numero di puri integratori, ovvero la molteplicità del polo nullo, della funzione di trasferimento ad anello aperto *A(s)*. *PLL type,* indicato con notazione romana, è necessariamente almeno pari a I, dato che l'uscita del VCO è in frequenza, mentre la funzione di trasferimento è definita in termini di fase (integrale della frequenza). Generalmente *PLL type* è I o II. Per ottenere un secondo integratore basta inserire un condensatore in uscita dalla pompa di carica sulla quale viene integrata la corrente. Per comodità l'integratore aggiuntivo viene considerato interno alla funzione di trasferimento del Loop Filter, per questo in alcuni testi *PLL type* viene denominato *typeLF* [3].

  I PLL di tipo I, a differenza dei PLL di tipo II, non presentano picchi indesiderati nella $G(s)$ e hanno un breve *settling time* (tempo necessario per la risposta al gradino unitario di raggiungere e di rimanere attorno al valore unitario entro un certo intervallo).

  I PLL di tipo II vengono utilizzati quando è necessario che la tensione in ingresso del VCO raggiunga un qualsiasi valore in continua, forzando simultaneamente l'errore di fase in ingresso al Loop Filter ad avere un valore continuo nullo. Infatti, in un semplice blocco di guadagno, la componente continua dell'uscita dipende direttamente

dal livello continuo dell'ingresso, mentre con un blocco guadagno-integratore la componente continua dell'uscita può essere arbitrariamente scelta in base alla durata del gradino in ingresso. Un'ulteriore vantaggio è quello di adattare il livello della componente continua della tensione di ingresso al VCO per sfruttare l'intero intervallo di uscita del VCO e questo lo rende preferibile rispetto al tipo I per i designer di PLL. Infatti, il Loop Filter di tipo I non ha generalmente un guadagno sufficientemente elevato per considerare tutta la portata dell'uscita del VCO coincidente con la tensione di alimentazione, di conseguenza è necessario utilizzare convertitore digitale analogico D/A aggiuntivo che si occupa del tuning grossolano della tensione di controllo del VCO, mentre il Loop Filter si occupa del tuning fine. Il guadagno del Loop Filter dipende dalla banda asintotica richiesta e da K_V del VCO, in particolare al diminuire della banda è richiesto un guadagno del Loop Filter minore e all'aumentare di K_V si richiede un guadagno del Loop Filter minore.

- $\boldsymbol{f_z}$ è la frequenza associata allo zero aggiuntivo della funzione di trasferimento ad anello aperto $A(s)$. Se il PLL è di tipo II, $A(s)$ deve contenere per motivi di stabilità anche uno zero. Infatti, i due integratori forniscono uno sfasamento di 180° ancor prima che il guadagno di $A(s)$ raggiunga l'unità, quindi non ci sarebbe in questo caso margine di fase per garantire la stabilità di $G(s)$. Lo zero aggiuntivo è introdotto per fornire un adeguato margine di fase e quindi un effetto di compensazione del polo introdotto dall'integratore. Affinché lo zero abbia l'effetto desiderato, è necessario che la sua frequenza f_z sia inferiore alla frequenza per la quale il guadagno di $A(s)$ è unitario. La necessità di inserire zeri aggiuntivi può nascere anche per PLL di tipo I per via dei poli di $A(s)$ che potrebbero non garantire un adeguato margine di fase. In pratica si aggiungono degli zeri extra ad $A(s)$ per migliorare la caratteristica di roll-off di $G(s)$.

- $\boldsymbol{f_z/f_0}$ è la posizione relativa dello zero aggiuntivo rispetto alla banda asintotica di $G(s)$ ed è un buon parametro di design della $G(s)$ per controllare l'effetto di compensazione ottenuto grazie allo zero aggiuntivo. Evidentemente, per le ragioni suddette, il rapporto f_z/f_0 deve essere minore di 1, ma dal punto di vista pratico f_z/f_0 deve essere minore di 1/2. I valori tipici ricadono nell'intervallo [1/10, 1/3]. Secondo Perrott [3] per applicazioni non a larga banda del PLL si usa come valore tipico 1/8, mentre per applicazioni a banda larga è consigliato un valore inferiore, ad esempio 1/9. La presenza di uno zero aggiuntivo determina un picco nella funzione di trasferimento

$G(s)$, la cui posizione dipende da f_z/f_0. All'aumentare di f_z/f_0, l'ampiezza del picco aumenta, ma diminuisce lo settling time.

Capitolo 2.3: Parametrizzazione del modello $G(s)$ del PLL

La funzione di trasferimento $G(s)$ è definita a partire da una funzione di trasferimento prototipo (Butterworth, Bessel, Chebyshev I e II, ellittico) che soddisfa le specifiche di progetto in termini di banda, roll-off, ripple massimo in banda passante, attenuazione minima in banda attenuata. A questo prototipo possono essere aggiunti poli e zeri nel caso fossero necessari per garantire la stabilità. Questo avviene obbligatoriamente nel caso di PLL di tipo II in cui il polo è introdotto dall'integratore e lo zero diventa indispensabile, opzionalmente per PLL di tipo I. Nella pratica, con prototipi aventi solo poli come Butterworth, Bessel e Chebyshev I, non è quasi mai richiesto un ordine m maggiore di 3, perché un roll-off maggiore di -60dB/dec generalmente non è necessario e perché avere un Loop Filter di elevato ordine comporta un aumento dei costi. Con prototipi Chebyshev II ed ellittici invece, essendo la specifica sul roll-off valida fra la frequenza di taglio e la frequenza minima degli zeri del prototipo, ordini m maggiori di 3 possono essere richiesti, infatti il roll-off oltre la massima frequenza degli zeri del prototipo è nullo in per ordini pari e -20dB/dec per ordini dispari.

Basandosi sullo schema a blocchi del modello nel dominio delle frequenze del sintetizzatore di frequenza a rapporto frazionario N con modulatore Σ-Δ di Figura 5, è possibile ricondurre il ramo diretto che include il PFD, la pompa di carica (C.P.), il Loop Filter e il VCO come in Figura 20:

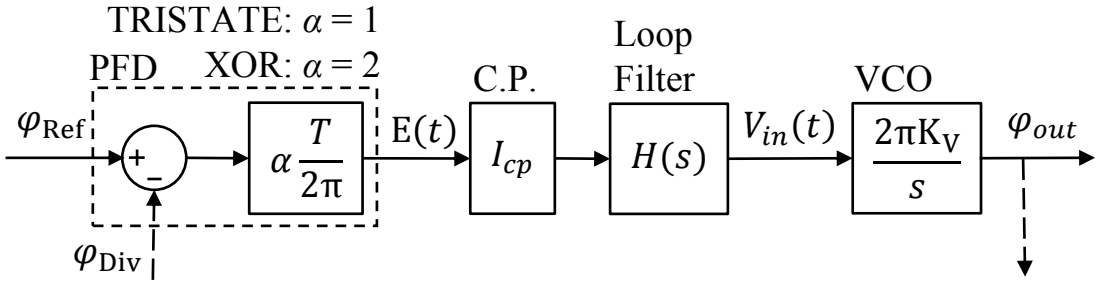

Figura 20: ramo diretto del modello nel dominio delle frequenze del PLL

e il ramo di retroazione che include soltanto il divisore di frequenza, il cui modello è un semplice fattore di guadagno, come in Figura 21:

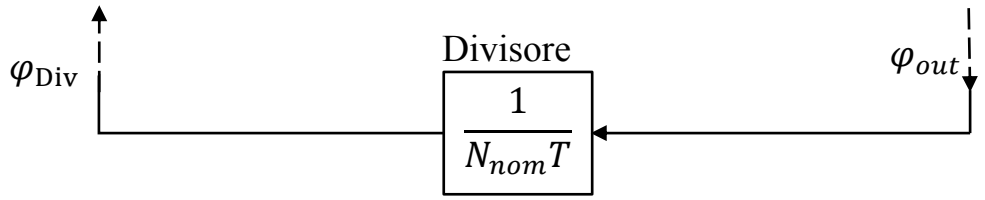

Figura 21: ramo di retroazione del modello nel dominio delle frequenze del PLL

Da qui si nota che il Loop Filter ha una funzione di trasferimento $H(s)$ che deve avere gli stessi poli e gli stessi zeri di $A(s)$, eccetto il polo nullo introdotto dal VCO.

Osservando la Figura 20 e la Figura 21, si ha:

$$F_F(s) = \frac{\alpha T I_{cp} H(s) \mathrm{K_V}}{s} \quad (33)$$

$$F_B(s) = \frac{1}{N_{nom}T} \quad (34)$$

Il guadagno ad anello aperto di ottiene applicando la (25):

$$A(s) = F_F(s)F_B(s) = \frac{\alpha I_{cp} H(s) \mathrm{K_V}}{N_{nom}} \quad (35)$$

Al fine di semplificare i conti, è conveniente considerare $A(s)$ come:

$$A(s) = \frac{N_A(s)}{D_A(s)} \quad (36)$$

dove $N_A(s)$ e $D_A(s)$ sono i polinomi in s, con $D_A(s)$ che contiene g poli nell'origine, con g pari al tipo di PLL.

Dalla (29), sostituendo la (36), si ottiene facilmente:

$$G(s) = \frac{N_A(s)}{D_A(s) + N_A(s)} \quad (37)$$

È evidente che:

- $G(s)$ deve avere gli stessi zeri di $A(s)$ e quindi di $H(s)$;
- se l'ordine di $G(s)$ è m, allora anche l'ordine di $D_A(s)$ e quindi di $A(s)$ è pari a m:

- poiché g è almeno 1, $A(s) \to \infty$ per $s \to 0$ (ovvero in DC). Questo implica che $G(s) \to 1$ per $s \to \infty$

Perrott in [3] considera solo prototipi di $G(s)$ in cui non ci sono zeri, ovvero non considera i prototipi di Chebyshev II ed ellittici. Questo limite del modello di Perrott è superabile modificando il modello e considerando anche gli zeri di $G(s)$, come verrà illustrato successivamente.

Capitolo 2.4: Caso $G(s)$ solo poli

In questo caso rientrano soltanto i prototipi di Butterworth, Bessel e Chebyshev I.

Conviene suddividere il problema in ulteriori due casi, ovvero PLL di tipo I e PLL di tipo II

PLL di tipo I

Perrott in [1] crea due modelli, uno per la $A(s)$, o analogamente per la $H(s)$, e uno per la $G(s)$. Il modello della $H(s)$ è definito dai parametri del Loop Filter, in particolare avrà un numero di parametri pari all'ordine di $H(s)$ + 1, in quanto il guadagno di $H(s)$ non è unitario. Discorso analogo per $A(s)$ in cui il numero dei parametri è pari all'ordine di $A(s)$, non essendo necessario un parametro per il polo nell'origine introdotto dal VCO. In questo caso $G(s)$ può essere considerata uguale al filtro prototipo, perché non ci sono poli e zeri aggiuntivi. Considerando il caso di prototipo di ordine m, $G(s)$ avendo solo poli, può essere modellata con m parametri, infatti il guadagno di $G(s)$ è preassegnato: $G(0) = 1$.

m	$H(s)$	$A(s)$	$G(s)$
1	K_{LP}	$\frac{K}{s}$	$\frac{1}{1+\frac{s}{\omega_{c0}}}$
2	$\frac{K_{LP}}{1+\frac{s}{\omega_P}}$	$\frac{K}{s\left(1+\frac{s}{\omega_P}\right)}$	$\frac{1}{1+\frac{s}{\omega_{c0}Q}+\frac{s^2}{\omega_{c0}^2}}$
3	$\frac{K_{LP}}{1+\frac{s}{\omega_P Q_P}+\frac{s^2}{\omega_P^2}}$	$\frac{K}{s\left(1+\frac{s}{\omega_P Q_P}+\frac{s^2}{\omega_P^2}\right)}$	$\frac{1}{\left(1+\frac{s}{\omega_{c1}}\right)\left(1+\frac{s}{\omega_{c0}Q}+\frac{s^2}{\omega_{c0}^2}\right)}$

Tabella 1: modelli delle funzioni di trasferimento H(s), A(s) e G(s) per PLL di tipo I, con G(s) avente solo poli

dove:

- m è l'ordine del prototipo
- K_{LP}, ω_P, Q_P sono i parametri del modello di $H(s)$, ovvero rispettivamente il guadagno, la pulsazione naturale del polo reale o complesso coniugato e il fattore di merito della coppia di poli complessi coniugati;
- K, ω_P, Q_P sono i parametri del modello di $A(s)$, dove il guadagno K è proporzionale a K_{LP} e ω_P, Q_P sono i medesimi di $H(s)$ per costruzione;
- ω_{c0}, ω_{c1}, Q sono i parametri del filtro prototipo $G(s)$, ovvero rispettivamente le pulsazioni naturali dei poli reali o complesso coniugato e il fattore di merito della coppia di poli complessi coniugati.

Il legame fra i guadagni di $H(s)$ e di $A(s)$ è:

$$K_{LP} = K\left(\frac{N_{nom}}{\mathrm{K_V} I_{cp}\alpha}\right) \tag{38}$$

A partire dal modello di $G(s)$ di ordine 3, è possibile determinare il modello di $G(s)$ di ordine 2 ponendo $\omega_{c1} \to \infty$ e quello di ordine 1 ponendo $\omega_{c0} \to \infty$ (in questo caso, nella notazione di Perrott, la pulsazione del polo reale si indica comunque con ω_{c0}).

Per generalità si considera il caso di ordine 3. La $A(s)$ può essere suddivisa in $N_A(s)$ e $D_A(s)$, come indicato in (36).

$$N_A(s) = 1 \tag{39}$$

$$D_A(s) = \left(\frac{s}{\omega_K}\right)\left(1 + \frac{s}{\omega_P Q_P} + \frac{s^2}{\omega_P^2}\right) \tag{40}$$

Dato che non ci sono zeri è infatti possibile utilizzare una costante scalare che replica il ruolo di ω_K, quindi:

$$K = \omega_K \tag{41}$$

Scrivendo la $G(s)$ in forma non fattorizzata si ha:

$$G(s) = \frac{1}{1 + \left(\frac{1}{\omega_{c0}Q} + \frac{1}{\omega_{c1}}\right)s + \left(\frac{1}{\omega_{c0}^2} + \frac{1}{\omega_{c1}\omega_{c0}Q}\right)s^2 + \left(\frac{1}{\omega_{c1}\omega_{c0}^2}\right)s^3} \tag{42}$$

Sostituendo (39) e (40) in (37) si ottiene:

$$G(s) = \frac{1}{1 + \left(\frac{1}{\omega_K}\right)s + \left(\frac{1}{\omega_K\omega_P Q_P}\right)s^2 + \left(\frac{1}{\omega_K\omega_P^2}\right)s^3} \tag{43}$$

Uguagliando i coefficienti uno ad uno delle ultime due espressioni, si ottengono i legami fra i parametri del modello di $A(s)$ e i parametri del modello di $G(s)$:

$$\begin{cases} \omega_K = \dfrac{\omega_{c1}\omega_{c0}Q}{\omega_{c0}Q + \omega_{c1}} \\ \omega_P = \omega_{c0}\sqrt{\dfrac{\omega_{c1}}{\omega_K}} = \omega_{c0}\sqrt{\dfrac{\omega_{c0}Q + \omega_{c1}}{\omega_{c0}Q}} \\ Q_P = \dfrac{\omega_{c1}\omega_{c0}^2 Q}{\omega_{c1}Q + \omega_{c0}}\dfrac{1}{\omega_K\omega_P} = \dfrac{\omega_P Q}{\omega_{c1}Q + \omega_{c0}} = \dfrac{\sqrt{\omega_{c0}Q(\omega_{c0}Q + \omega_{c1})}}{\omega_{c1}Q + \omega_{c0}} \end{cases} \tag{44}$$

Risulta interessante trovare una generalizzazione per i coefficienti di $D_A(s)$ indicati in (40).

Nel caso di $G(s)$ di ordine 3 si ha:

$$\begin{cases} \dfrac{1}{\omega_K} = \dfrac{1}{\omega_{c0}Q} + \dfrac{1}{\omega_{c1}} \\ \dfrac{1}{\omega_P Q_P} = \dfrac{\omega_{c1}Q + \omega_{c0}}{\omega_{c0}(\omega_{c0}Q + \omega_{c1})} \\ \dfrac{1}{\omega_P^2} = \dfrac{Q}{(\omega_{c0}Q + \omega_{c1})\omega_{c0}} \end{cases} \tag{45}$$

Dalla quale si ricavano i coefficienti di $D_A(s)$ per $G(s)$ di ordine 2 ponendo $\omega_{c1} \to \infty$:

$$\begin{cases} \dfrac{1}{\omega_K} = \dfrac{1}{\omega_{c0}Q} \\ \dfrac{1}{\omega_P Q_P} = \dfrac{Q}{\omega_{c0}} \\ \dfrac{1}{\omega_P^2} = 0 \end{cases} \tag{46}$$

e di ordine 1 ponendo $\omega_{c0} \to \infty$ e utilizzando la notazione di Perrott per la pulsazione del polo reale ω_{c0}:

$$\begin{cases} \dfrac{1}{\omega_K} = \dfrac{1}{\omega_{c0}} \\ \dfrac{1}{\omega_P Q_P} = 0 \\ \dfrac{1}{\omega_P^2} = 0 \end{cases} \qquad (47)$$

Si è così ottenuta la configurazione dei parametri della $A(s)$ che determina la $G(s)$ secondo le specifiche.

Nella tabella riassuntiva sottostante sono riportate le configurazioni dei parametri di $A(s)$ che determinano le $G(s)$ desiderate di ordine 1, 2 e 3.

m	K	ω_P	Q_P
1	ω_{c0}	N/A	N/A
2	$\omega_{c0}Q$	$\dfrac{\omega_{c0}}{Q}$	N/A
3	$\dfrac{\omega_{c1}\omega_{c0}Q}{\omega_{c0}Q+\omega_{c1}}$	$\omega_{c0}\sqrt{\dfrac{\omega_{c0}Q+\omega_{c1}}{\omega_{c0}Q}}$	$\dfrac{\sqrt{\omega_{c0}Q(\omega_{c0}Q+\omega_{c1})}}{\omega_{c1}Q+\omega_{c0}}$

Tabella 2: configurazione dei parametri del Loop Filter che determina la G(s) con solo poli desiderata, per ordini 1, 2, 3 per PLL di tipo I

PLL di tipo II

In questo caso $A(s)$ deve avere uno zero alla pulsazione angolare $\omega_z = 2\pi f_z$, per garantire un appropriato margine di fase. I modelli di $H(s)$, $A(s)$ e $G(s)$ diventano:

m	$H(s)$	$A(s)$	$G(s)$
1	$\frac{K_{LP}\left(1+\frac{s}{\omega_z}\right)}{s}$	$\frac{K\left(1+\frac{s}{\omega_z}\right)}{s^2}$	$\frac{\left(1+\frac{s}{\omega_z}\right)}{\left(1+\frac{s}{\omega_{cp}}\right)\left(1+\frac{s}{\omega_{c0}}\right)}$
2	$\frac{K_{LP}\left(1+\frac{s}{\omega_z}\right)}{s\left(1+\frac{s}{\omega_P}\right)}$	$\frac{K\left(1+\frac{s}{\omega_z}\right)}{s^2\left(1+\frac{s}{\omega_P}\right)}$	$\frac{\left(1+\frac{s}{\omega_z}\right)}{\left(1+\frac{s}{\omega_{cp}}\right)\left(1+\frac{s}{\omega_{c0}Q}+\frac{s^2}{\omega_{c0}^2}\right)}$
3	$\frac{K_{LP}\left(1+\frac{s}{\omega_z}\right)}{s\left(1+\frac{s}{\omega_P Q_P}+\frac{s^2}{\omega_P^2}\right)}$	$\frac{K\left(1+\frac{s}{\omega_z}\right)}{s^2\left(1+\frac{s}{\omega_P Q_P}+\frac{s^2}{\omega_P^2}\right)}$	$\frac{\left(1+\frac{s}{\omega_z}\right)}{\left(1+\frac{s}{\omega_{cp}}\right)\left(1+\frac{s}{\omega_{c1}}\right)\left(1+\frac{s}{\omega_{c0}Q}+\frac{s^2}{\omega_{c0}^2}\right)}$

Tabella 3: modelli delle funzioni di trasferimento H(s), A(s) e G(s) per PLL di tipo II, con G(s) avente solo poli

dove ω_{cp} è la pulsazione naturale del polo aggiunto per costruzione di $G(s)$, generato dallo stadio di integrazione aggiuntivo nella $H(s)$. Gli altri parametri sono i medesimi descritti nel caso precedente. Il fatto che lo stadio integratore produca un polo aggiuntivo $p = -\omega_{cp}$ su $G(s)$ è molto conveniente, infatti se ci fosse solo lo zero extra il roll-off di $G(s)$ verrebbe alterato.

Per generalità si considera il caso di ordine 3. La $A(s)$ può essere suddivisa in $N_A(s)$ e $D_A(s)$, come indicato in (36).

$$N_A(s) = \left(1+\frac{s}{\omega_z}\right) \tag{48}$$

$$D_A(s) = \left(\frac{s^2}{\omega_K^2}\right)\left(1+\frac{s}{\omega_P Q_P}+\frac{s^2}{\omega_P^2}\right) \tag{49}$$

Scrivendo la $G(s)$ in forma non fattorizzata si ha:

$$G(s) = \frac{1}{1 + \left(\frac{1}{\omega_{c0}Q} + \frac{1}{\omega_{c1}} + \frac{1}{\omega_{c1}}\right)s + \left(\frac{1}{\omega_{c0}^2} + \frac{1}{\omega_{c1}\omega_{c0}Q} + \frac{1}{\omega_{cp}\omega_{c0}Q} + \frac{1}{\omega_{cp}\omega_{c1}}\right)s^2 + \left(\frac{1}{\omega_{cp}\omega_{c0}^2} + \frac{1}{\omega_{c1}\omega_{c0}^2} + \frac{1}{\omega_{cp}\omega_{c1}\omega_{c0}Q}\right)s^3 + \left(\frac{1}{\omega_{cp}\omega_{c1}\omega_{c0}^2}\right)s^4} \quad (50)$$

Sostituendo (48) e (49) in (37) si ottiene:

$$G(s) = \frac{1 + \frac{s}{\omega_z}}{1 + \left(\frac{1}{\omega_z}\right)s + \left(\frac{1}{\omega_K^2}\right)s^2 + \left(\frac{1}{Q_P\omega_K\omega_P^2}\right)s^3 + \left(\frac{1}{\omega_K^2\omega_P^2}\right)s^4} \quad (51)$$

Uguagliando i coefficienti uno ad uno delle ultime due espressioni, si ottengono i legami fra i parametri del modello di $A(s)$ e i parametri del modello di $G(s)$:

$$\begin{cases} \omega_{cp} = \dfrac{\omega_z\omega_{c1}\omega_{c0}Q}{\omega_{c1}\omega_{c0}Q - \omega_z\omega_{c0}Q - \omega_z\omega_{c1}} \\ \omega_K = \dfrac{\omega_{c0}\sqrt{\omega_{cp}\omega_{c1}Q}}{\sqrt{\omega_{c0}^2Q + \omega_{c1}\omega_{c0} + \omega_{cp}\omega_{c0} + \omega_{cp}\omega_{c1}Q}} \\ \omega_P = \omega_{c0}\sqrt{\dfrac{\omega_{cp}\omega_{c1}}{\omega_K}} \\ Q_P = \dfrac{\omega_{cp}\omega_{c1}\omega_{c0}^2Q}{\omega_K^2\omega_P(\omega_{cp}Q + \omega_{c1}Q + \omega_{c0})} = \dfrac{\omega_{cp}Q}{\omega_{cp}Q + \omega_{c1}Q + \omega_{c0}} \end{cases} \quad (52)$$

Risulta interessante trovare una generalizzazione per i coefficienti di $D_A(s)$ indicati in (47).

Nel caso di $G(s)$ di ordine 3 si ha:

$$\begin{cases} \dfrac{1}{\omega_K^2} = \dfrac{\omega_{c1}\omega_{c0}Q(\omega_{c1}-\omega_z) - \omega_z\omega_{c1}^2 + Q^2(\omega_{c0}^2(\omega_{c1}-\omega_z) + \omega_z\omega_{c1}^2)}{\omega_z\omega_{c1}^2\omega_{c0}^2Q^2} \\ \dfrac{1}{\omega_PQ_P} = \dfrac{\omega_{c1}^2\omega_{c0}Q^2 - \omega_z\omega_{c1}\omega_{c0} + Q(\omega_{c0}^2(\omega_{c1}-\omega_z) + \omega_z\omega_{c1}^2)}{\omega_{c0}(\omega_{c1}\omega_{c0}Q(\omega_{c1}-\omega_z) - \omega_z\omega_{c1}^2 + Q^2(\omega_{c0}^2(\omega_{c1}-\omega_z) + \omega_z\omega_{c1}^2))} \\ \dfrac{1}{\omega_P^2} = \dfrac{Q(\omega_{c0}Q(\omega_{c1}-\omega_z) - \omega_z\omega_{c1})}{\omega_{c0}(\omega_{c1}\omega_{c0}Q(\omega_{c1}-\omega_z) - \omega_z\omega_{c1}^2 + Q^2(\omega_{c0}^2(\omega_{c1}-\omega_z) + \omega_z\omega_{c1}^2))} \end{cases} \quad (53)$$

Dalla quale si ricavano i coefficienti di $D_A(s)$ per $G(s)$ di ordine 2 ponendo $\omega_{c1} \to \infty$:

$$\begin{cases} \dfrac{1}{\omega_K^2} = \dfrac{\omega_{c0}Q - \omega_z + \omega_z Q^2}{\omega_z \omega_{c0}^2 Q^2} \\ \dfrac{1}{\omega_P Q_P} = \dfrac{Q(\omega_{c0}Q - \omega_z)}{\omega_{c0}(\omega_{c0}Q - \omega_z + \omega_z Q^2)} \\ \dfrac{1}{\omega_P^2} = 0 \end{cases} \quad (54)$$

e di ordine 1, ponendo $\omega_{c0} \to \infty$ e utilizzando la notazione di Perrott per la pulsazione del polo reale ω_{c0}:

$$\begin{cases} \dfrac{1}{\omega_K} = \dfrac{\omega_{c0} - \omega_z}{\omega_z \omega_{c0}^2} \\ \dfrac{1}{\omega_P Q_P} = 0 \\ \dfrac{1}{\omega_P^2} = 0 \end{cases} \quad (55)$$

Si è così ottenuta la configurazione dei parametri della $A(s)$ che determina la $G(s)$ secondo le specifiche.

Nella tabella riassuntive sottostanti sono riportate le configurazioni dei parametri di $A(s)$ per le $G(s)$ di ordine 1, 2 e 3.

m	ω_{cp}	K	ω_P	Q_P
1	$\dfrac{\omega_z}{1 - \dfrac{\omega_z}{\omega_{c0}}}$	$\omega_{cp}\omega_{c0}$	N/A	N/A
2	$\dfrac{\omega_z}{1 - \dfrac{\omega_z}{\omega_{c0}Q}}$	$\dfrac{\omega_{c0}Q}{\dfrac{1}{\omega_{cp}} + \dfrac{Q}{\omega_{c0}}}$	$\omega_{c0}\left(\dfrac{\omega_{cp}}{\omega_{c0}} + \dfrac{1}{Q}\right)$	N/A
3	$\dfrac{\omega_z}{1 - \dfrac{\omega_z}{\omega_{c1}} - \dfrac{\omega_z}{\omega_{c0}Q}}$	$\dfrac{\omega_{c0}Q}{\dfrac{1}{\omega_{cp}} + \dfrac{Q}{\omega_{c0}} + Q\left(\dfrac{1}{\omega_{c0}} + \dfrac{\omega_{c0}}{\omega_{cp}\omega_{c1}}\right)}$	$\omega_{c0}\sqrt{\dfrac{\omega_{cp}\omega_{c1}}{K}}$	$\dfrac{\omega_P Q}{\omega_{c0} + Q(\omega_{cp} + \omega_{c1})}$

Tabella 4: configurazione dei parametri del Loop Filter che determina la G(s) con solo poli desiderata, per ordini 1, 2, 3 per PLL di tipo II

Capitolo 2.5: Caso *G*(*s*) con zeri

In questo caso rientrano i prototipi di Chebyshev II ed ellittici. Perrott in [1] non considera l'effetto degli zeri dei prototipi, i quali modificano i parametri del modello di *A*(*s*) e di conseguenza il *G*(*s*) non è quello desiderato. Affinché la *G*(*s*) risulti di tipo Chebyshev II o ellittica, è necessario adottare un modello del loop filter *H*(*s*) e quindi di *A*(*s*) differente. In particolare, gli zeri di *H*(*s*) e di *A*(*s*) (non considerando quelli extra aggiunti per la stabilità e quelli parassiti) devono coincidere con gli zeri del prototipo di *G*(*s*). Per questo motivo verrà utilizzata la medesima notazione in tutti e tre i modelli.

Il metodo enunciato prima nel caso di prototipi con zeri non è valido, in quanto i coefficienti del polinomio $D_A(s)$ dipenderanno anche dagli zeri e questa dipendenza non è la medesima per ogni ordine, per cui non è possibile determinare, come nel caso precedente, i parametri del Loop Filter a partire dalla formulazione del terzo ordine.

Con il metodo alternativo proposto di seguito questo problema non sussiste. In particolare, questo metodo è valido anche per *G*(*s*) aventi solo poli.

Conviene anche in questo caso suddividere il problema in ulteriori due casi, ovvero PLL di tipo I e PLL di tipo II

PLL di tipo I

I modelli di *H*(*s*), *A*(*s*) e *G*(*s*) sono quelli indicati in tabella:

m	$H(s)$	$A(s)$	$G(s)$
1	K_{LP}	$\frac{K}{s}$	$\frac{1}{1+\frac{s}{\omega_{c0}}}$
2	$\frac{K_{LP}\left(1+\frac{s^2}{\omega_{z0}^2}\right)}{1+\frac{s}{\omega_P}}$	$\frac{K\left(1+\frac{s^2}{\omega_{z0}^2}\right)}{s\left(1+\frac{s}{\omega_P}\right)}$	$\frac{1+\frac{s^2}{\omega_{z0}^2}}{1+\frac{s}{\omega_{c0}Q}+\frac{s^2}{\omega_{c0}^2}}$
3	$\frac{K_{LP}\left(1+\frac{s^2}{\omega_{z0}^2}\right)}{1+\frac{s}{\omega_P Q_P}+\frac{s^2}{\omega_P^2}}$	$\frac{K\left(1+\frac{s^2}{\omega_{z0}^2}\right)}{s\left(1+\frac{s}{\omega_P Q_P}+\frac{s^2}{\omega_P^2}\right)}$	$\frac{1+\frac{s^2}{\omega_{z0}^2}}{\left(1+\frac{s}{\omega_{c1}}\right)\left(1+\frac{s}{\omega_{c0}Q}+\frac{s^2}{\omega_{c0}^2}\right)}$

Tabella 5: modelli delle funzioni di trasferimento H(s), A(s) e G(s) per PLL di tipo I, con G(s) avente anche zeri

dove ω_{z0} è la pulsazione naturale della coppia di zeri complessi coniugati a parte reale nulla del prototipo di Chebyshev II e ellittico. Come anticipato prima, gli zeri di $H(s)$ devono coincidere con il prototipo, per cui il parametro ω_{z0} è univocamente determinato dal prototipo.

Considerando:

$$G(s) = \frac{N_G(s)}{D_G(s)} \tag{56}$$

dove $N_G(s)$ e $D_G(s)$ devono essere normalizzati rispetto al coefficiente di s^0, in modo che sia unitario.

Si parametrizza il denominatore di $A(s)$ con:

$$D_A(s) = \frac{s}{\omega_K} X_A(s) \tag{57}$$

dove $X_A(s)$ deve essere normalizzato in modo che il coefficiente di s^0 sia unitario.

Sostituendo la (57) e la (56) in (37) si ha:

$$\frac{N_G(s)}{D_G(s)} = \frac{N_A(s)}{\frac{s}{\omega_K} X_A(s) + N_A(s)} \tag{58}$$

Da cui si ricava:

$$\begin{cases} N_A(s) = N_G(s) \\ X_A(s) = \dfrac{\omega_K}{s} (D_G(s) - N_G(s)) \end{cases} \tag{59}$$

dove grazie alla normalizzazione, il coefficiente di s^0 di $D_G(s)$ - $N_G(s)$ è nullo. Questo consente la semplificazione del termine s al denominatore, così 1 / ω_K sarà uguale al coefficiente di s^1 di $D_G(s)$ -$N_G(s)$.

In pratica si sta imponendo che:

$$\begin{cases} N_A(s) = N_G(s) \\ D_A(s) = D_G(s) - N_G(s) \end{cases} \tag{60}$$

Indicando con d_{Gi} e n_{Gi} i coefficienti di s^i rispettivamente di $N_G(s)$ e $D_G(s)$ normalizzati, si ha:

$$\omega_K = \frac{1}{d_{G_1} - n_{G_1}} \tag{61}$$

e la correttezza del prototipo iniziale $G(s)$ può essere verificata controllando l'invarianza:

$$d_{G_0} - n_{G_0} = 0 \tag{62}$$

Applicando questo metodo alternativo, è possibile determinare la configurazione dei parametri del Loop Filter che determina la $G(s)$ desiderata per i differenti ordini.

Il caso del 1° ordine coincide con il caso senza zeri, essendo i modelli di $H(s)$, $A(s)$ e $G(s)$ coincidenti.

Nel caso del 2° ordine si ha:

$$\begin{cases} N_G(s) = 1 + \dfrac{s^2}{\omega_{z0}^2} \\ D_G(s) = 1 + \left(\dfrac{1}{\omega_{c0}Q}\right)s + \left(\dfrac{1}{\omega_{c0}^2}\right)s^2 \end{cases} \tag{63}$$

Dalla (61) si ricava:

$$\omega_K = \omega_{c0}Q \tag{64}$$

Dalla (60) si ricava:

$$\begin{cases} N_A(s) = 1 + \dfrac{s^2}{\omega_{z0}^2} \\ D_A(s) = \left(\dfrac{1}{\omega_{c0}Q}\right)s + \left(\dfrac{1}{\omega_{c0}^2} - \dfrac{1}{\omega_{z0}^2}\right)s^2 \end{cases} \tag{65}$$

Per cui:

$$X_A(s) = \omega_K\left(\frac{1}{\omega_{c0}^2} - \frac{1}{\omega_{z0}^2}\right)s + 1 \tag{66}$$

Il parametro K coincide per costruzione con ω_K, mentre il parametro restante del modello di $H(s)$, ovvero ω_P, si determina mediante la radice del polinomio $X_A(s)$. In questo caso è immediato uguagliare uno ad uno i coefficienti dei polinomi $X_A(s)$ e $D_H(s)$:

$$\frac{1}{\omega_P} = \omega_K \left(\frac{1}{\omega_{c0}^2} - \frac{1}{\omega_{z0}^2} \right) = \frac{Q}{\omega_{c0}} - \frac{\omega_{c0} Q}{\omega_{z0}^2} \tag{67}$$

Nel caso del 3° ordine, si ha:

$$\begin{cases} N_G(s) = 1 + \dfrac{s^2}{\omega_{z0}^2} \\ D_G(s) = 1 + \left(\dfrac{1}{\omega_{c0} Q} + \dfrac{1}{\omega_{c1}} \right) s + \left(\dfrac{1}{\omega_{c0}^2} + \dfrac{1}{\omega_{c1} \omega_{c0} Q} \right) s^2 + \left(\dfrac{1}{\omega_{c1} \omega_{c0}^2} \right) s^3 \end{cases} \tag{68}$$

Dalla (61) si ricava:

$$\omega_K = \left(\frac{1}{\omega_{c0} Q} + \frac{1}{\omega_{c1}} \right)^{-1} \tag{69}$$

Dalla (60):

$$\begin{cases} N_A(s) = 1 + \dfrac{s^2}{\omega_{z0}^2} \\ D_A(s) = \left(\dfrac{1}{\omega_{c0} Q} + \dfrac{1}{\omega_{c1}} \right) s + \left(\dfrac{1}{\omega_{c0}^2} + \dfrac{1}{\omega_{c1} \omega_{c0} Q} - \dfrac{1}{\omega_{z0}^2} \right) s^2 + \left(\dfrac{1}{\omega_{c1} \omega_{c0}^2} \right) s^3 \end{cases} \tag{70}$$

Per cui:

$$X_A(s) = 1 + \omega_K \left(\frac{1}{\omega_{c0}^2} + \frac{1}{\omega_{c1} \omega_{c0} Q} - \frac{1}{\omega_{z0}^2} \right) s + \left(\frac{\omega_K}{\omega_{c1} \omega_{c0}^2} \right) s^2 \tag{71}$$

Il parametro K coincide per costruzione con ω_K.

I parametri restanti del modello di $H(s)$, ovvero ω_P e Q_P, si determinano mediante le radici del polinomio $X_A(s)$. In questo caso è immediato uguagliare uno ad uno i coefficienti dei polinomi $X_A(s)$ e $D_H(s)$:

$$\begin{cases} \dfrac{1}{\omega_P^2} = \dfrac{\omega_K}{\omega_{c1} \omega_{c0}^2} = \dfrac{1}{\dfrac{\omega_{c1} \omega_{c0}}{Q} + \omega_{c0}^2} \\ \dfrac{1}{\omega_P Q_P} = \dfrac{\omega_K}{\omega_{c0}^2} + \dfrac{\omega_K}{\omega_{c1} \omega_{c0} Q} - \dfrac{\omega_K}{\omega_{z0}^2} = \dfrac{1}{\dfrac{\omega_{c0}}{Q} + \dfrac{\omega_{c0}^2}{\omega_{c1}}} + \dfrac{1}{\omega_{c1} + \omega_{c0} Q} - \dfrac{1}{\dfrac{\omega_{z0}^2}{\omega_{c0} Q} + \dfrac{\omega_{z0}^2}{\omega_{c1}}} \end{cases} \tag{72}$$

Nella tabella riassuntiva sottostante sono riportate le configurazioni dei parametri di $A(s)$ per le $G(s)$ di ordine 1, 2 e 3.

m	K	ω_P	Q_P
1	ω_{c0}	N/A	N/A
2	$\omega_{c0}Q$	$\frac{\omega_{c0}\omega_{z0}^2}{Q(\omega_{z0}^2-\omega_{c0}^2)}$	N/A
3	$\frac{\omega_{c1}\omega_{c0}Q}{\omega_{c0}Q+\omega_{c1}}$	$\omega_{c0}\sqrt{\frac{\omega_{c0}Q+\omega_{c1}}{\omega_{c0}Q}}$	$\frac{\omega_P}{\frac{1}{\frac{\omega_{c0}}{Q}+\frac{\omega_{c0}^2}{\omega_{c1}}}+\frac{1}{\omega_{c1}+\omega_{c0}Q}-\frac{1}{\frac{\omega_{z0}^2}{\omega_{c0}Q}+\frac{\omega_{z0}^2}{\omega_{c1}}}}$

Tabella 6: configurazione dei parametri del Loop Filter che determina la G(s) avente anche zeri desiderata, per ordini 1, 2, 3 per PLL di tipo I

A riprova della validità del metodo anche per filtri con solo poli, con la $G(s)$ espressa in (42) si ha:

$$\begin{cases} N_G(s) = 1 \\ D_G(s) = 1 + \left(\frac{1}{\omega_{c0}Q}+\frac{1}{\omega_{c1}}\right)s + \left(\frac{1}{\omega_{c0}^2}+\frac{1}{\omega_{c1}\omega_{c0}Q}\right)s^2 + \left(\frac{1}{\omega_{c1}\omega_{c0}^2}\right)s^3 \end{cases} \quad (73)$$

per cui:

$$\begin{cases} N_A(s) = 1 \\ D_A(s) = \left(\frac{1}{\omega_{c0}Q}+\frac{1}{\omega_{c1}}\right)s + \left(\frac{1}{\omega_{c0}^2}+\frac{1}{\omega_{c1}\omega_{c0}Q}\right)s^2 + \left(\frac{1}{\omega_{c1}\omega_{c0}^2}\right)s^3 = \\ \quad = \frac{s}{\omega_K}\left(1+\omega_K\left(\frac{1}{\omega_{c0}^2}+\frac{1}{\omega_{c1}\omega_{c0}Q}\right)s+\left(\frac{\omega_K}{\omega_{c1}\omega_{c0}^2}\right)s^2\right) \end{cases} \quad (74)$$

Dall'equazioni (61) si ottiene:

$$\omega_K = \left(\frac{1}{\omega_{c0}Q}+\frac{1}{\omega_{c1}}\right)^{-1} \quad (75)$$

Uguagliando la seconda equazione del sistema (74) con la (40) si perviene alle medesime relazioni espresse in (44).

PLL di tipo II

I modelli di $H(s)$, $A(s)$ e $G(s)$ sono quelli indicati in tabella:

m	$H(s)$	$A(s)$	$G(s)$
1	$\frac{K_{LP}\left(1+\frac{s}{\omega_z}\right)}{s}$	$\frac{K\left(1+\frac{s}{\omega_z}\right)}{s^2}$	$\frac{\left(1+\frac{s}{\omega_z}\right)}{\left(1+\frac{s}{\omega_{cp}}\right)\left(1+\frac{s}{\omega_{c0}}\right)}$
2	$\frac{K_{LP}\left(1+\frac{s}{\omega_z}\right)\left(1+\frac{s^2}{\omega_{z0}^2}\right)}{s\left(1+\frac{s}{\omega_P}\right)}$	$\frac{K\left(1+\frac{s}{\omega_z}\right)\left(1+\frac{s^2}{\omega_{z0}^2}\right)}{s^2\left(1+\frac{s}{\omega_P}\right)}$	$\frac{\left(1+\frac{s}{\omega_z}\right)\left(1+\frac{s^2}{\omega_{z0}^2}\right)}{\left(1+\frac{s}{\omega_{cp}}\right)\left(1+\frac{s}{\omega_{c0}Q}+\frac{s^2}{\omega_{c0}^2}\right)}$
3	$\frac{K_{LP}\left(1+\frac{s}{\omega_z}\right)\left(1+\frac{s^2}{\omega_{z0}^2}\right)}{s\left(1+\frac{s}{\omega_P Q_P}+\frac{s^2}{\omega_P^2}\right)}$	$\frac{K\left(1+\frac{s}{\omega_z}\right)\left(1+\frac{s^2}{\omega_{z0}^2}\right)}{s^2\left(1+\frac{s}{\omega_P Q_P}+\frac{s^2}{\omega_P^2}\right)}$	$\frac{\left(1+\frac{s}{\omega_z}\right)\left(1+\frac{s^2}{\omega_{z0}^2}\right)}{\left(1+\frac{s}{\omega_{cp}}\right)\left(1+\frac{s}{\omega_{c1}}\right)\left(1+\frac{s}{\omega_{c0}Q}+\frac{s^2}{\omega_{c0}^2}\right)}$

Tabella 7: modelli delle funzioni di trasferimento H(s), A(s) e G(s) per PLL di tipo II, con G(s) avente anche zeri

dove ω_{z0} è la pulsazione naturale della coppia di zeri complessi coniugati a parte reale nulla del prototipo di Chebyshev II e ellittico. Come anticipato prima, gli zeri di $H(s)$ devono coincidere con il prototipo, per cui il parametro ω_{z0} è univocamente determinato dal prototipo.

Considerando $G(s)$ come in (56), si parametrizza il denominatore di $A(s)$ con:

$$D_A(s) = \frac{s^2}{\omega_K^2} X_A(s) \qquad (76)$$

dove $X_A(s)$ deve essere normalizzato in modo che il coefficiente di s^0 sia 1.

Sostituendo la (76) e la (56) in (37) si ha:

$$\frac{N_G(s)}{D_G(s)} = \frac{N_A(s)}{\frac{s^2}{\omega_K^2} X_A(s) + N_A(s)} \qquad (77)$$

Dalla (77) è abbastanza evidente che se $N_A(s)$ non ha termini in s^1, nemmeno $D_G(s)$ ne avrà.

Poiché in questo caso:

$$N_A(s) = N_G(s) = \left(1 + \frac{s}{\omega_z}\right)(\dots) \qquad (78)$$

allora:

$$\begin{cases} N_A(s) = N_G(s) \\ X_A(s) = \dfrac{\omega_K^2}{s^2}\left(D_G(s) - N_G(s)\right) \end{cases} \qquad (79)$$

Dalla seconda equazione del sistema è evidente che sono necessarie due cancellazioni nell'espressione di $X_A(s)$ affinché il termine s^2 sia raccoglibile a denominatore, ovvero i termini in s^1 e s^0 di $D_G(s)$ e $N_G(s)$ devono annullarsi facendo la differenza. La cancellazione dei termini in s^0 è automatica essendo entrambi i polinomi normalizzati e quindi aventi coefficienti unitari in s^0. Discorso differente per la cancellazione dei termini in s^1, per la quale è necessario aggiungere un grado di libertà in più a $G(s)$ dato dal polo aggiuntivo $p = -\omega_{cp}$. Con questo grado di libertà, assunto preassegnato lo zero aggiuntivo $z = -\omega_z$, è possibile determinare il valore di ω_{cp} che consente la cancellazione del termine s^1 di $D_G(s) - N_G(s)$.

Considerando il polo e lo zero aggiunti si ha:

$$G(s) = \frac{N_G(s)}{D_G(s)} = \frac{1 + \frac{s}{\omega_z}}{1 + \frac{s}{\omega_{cp}}} \frac{N_{\hat{G}}(s)}{D_{\hat{G}}(s)} = \frac{1 + \frac{s}{\omega_z}}{1 + \frac{s}{\omega_{cp}}} \hat{G}(s) \qquad (80)$$

dove $\hat{G}(s)$ è la funzione di trasferimento del prototipo.

L'equazione (78) diventa:

$$\begin{cases} N_G(s) = \left(1 + \dfrac{s}{\omega_z}\right) N_{\hat{G}}(s) \\ X_A(s) = \dfrac{\omega_K^2}{s^2}\left(\left(1 + \dfrac{s}{\omega_{cp}}\right) D_{\hat{G}}(s) - \left(1 + \dfrac{s}{\omega_z}\right) N_{\hat{G}}(s)\right) \end{cases} \qquad (81)$$

Dalla seconda equazione del sistema è evidente che ω_{cp} dipende non solo da ω_z, ma anche dai coefficienti in s^1 di $N_{\hat{G}}(s)$ e $D_{\hat{G}}(s)$. Nei protitipi Chebyshev II ed ellittici la dipendenza da $N_{\hat{G}}(s)$ non c'è essendoci solo coppie di zeri complessi coniugati a parte reale nulla, ma la trattazione ha validità generale anche nel caso in cui la parte reale non fosse nulla. La costante

ω_K^2 è invece definita come il valore che normalizza $X_A(s)$ in modo che il coefficiente di s^0 sia unitario.

Indicando con $n_{\hat{G}_i}$ e $d_{\hat{G}_i}$ i coefficienti di s^i di $\mathrm{N}_{\hat{\mathrm{G}}}(\mathrm{s})$ e $D_{\hat{\mathrm{G}}}(\mathrm{s})$, si ottiene:

$$\omega_{cp} = \frac{d_{\hat{G}_0}}{n_{\hat{G}_1} + \frac{n_{\hat{G}_0}}{\omega_z} - d_{\hat{G}_1}} \tag{82}$$

Di conseguenza, dalla (37), $A(s)$ può essere espresso come:

$$\begin{cases} N_A(s) = \left(1 + \frac{s}{\omega_z}\right) N_{\hat{G}}(s) \\ D_A(s) = \left(1 + \frac{s}{\omega_{cp}}\right) D_{\hat{G}}(s) - \left(1 + \frac{s}{\omega_z}\right) N_{\hat{G}}(s) \end{cases} \tag{83}$$

Mentre il valore che normalizza $X_A(s)$ è:

$$\omega_K^2 = \frac{1}{d_{A_2}} \tag{84}$$

con d_{A_2} che indica il coefficiente di s^2 di $D_A(s)$.

La correttezza del prototipo iniziale $G(s)$ può essere verificata controllando l'invarianza:

$$d_{A_0} = 0 \tag{85}$$

I parametri relativi al denominatore della funzione di trasferimento del Loop Filter $H(s)$, si ottengono mediante le radici del polinomio $X_A(s)$, mentre il guadagno K è:

$$K = \omega_K^2 \tag{86}$$

Applicando questo metodo alternativo, è possibile determinare la configurazione dei parametri del Loop Filter che determina la $G(s)$ desiderata per i differenti ordini.

Il caso del 1° ordine coincide con il caso senza zeri, essendo i modelli di $H(s)$, $A(s)$ e $G(s)$ coincidenti.

Nel caso del 2° ordine il prototipo di $G(s)$ è esprimibile come:

$$\begin{cases} N_{\hat{G}}(s) = 1 + \left(\frac{1}{\omega_{z0}^2}\right)s^2 \\ D_{\hat{G}}(s) = 1 + \left(\frac{1}{\omega_{c0}Q}\right)s + \left(\frac{1}{\omega_{c0}^2}\right)s^2 \end{cases} \quad (87)$$

Utilizzando la (80) si ha:

$$\begin{cases} N_G(s) = 1 + \left(\frac{1}{\omega_z}\right)s + \left(\frac{1}{\omega_{z0}^2}\right)s^2 + \left(\frac{1}{\omega_{z0}^2\omega_z}\right)s^3 \\ D_G(s) = 1 + \left(\frac{1}{\omega_{cp}\omega_{c0}Q} + \frac{1}{\omega_{cp}}\right)s + \left(\frac{1}{\omega_{c0}^2} + \frac{1}{\omega_{cp}\omega_{c0}Q}\right)s^2 + \left(\frac{1}{\omega_{cp}\omega_{c0}^2}\right)s^3 \end{cases} \quad (88)$$

Applicando la (82) si ottiene il la pulsazione naturale del polo aggiuntivo:

$$\omega_{cp} = \frac{1}{\frac{1}{\omega_z} - \frac{1}{\omega_{c0}Q}} \quad (89)$$

quindi dalla (83):

$$\begin{cases} N_A(s) = 1 + \left(\frac{1}{\omega_z}\right)s + \left(\frac{1}{\omega_{z0}^2}\right)s^2 + \left(\frac{1}{\omega_{z0}^2\omega_z}\right)s^3 \\ D_A(s) = \left(\frac{1}{\omega_{cp}\omega_{c0}Q} + \frac{1}{\omega_{cp}} - \frac{1}{\omega_z}\right)s + \left(\frac{1}{\omega_{c0}^2} + \frac{1}{\omega_{cp}\omega_{c0}Q} - \frac{1}{\omega_{z0}^2}\right)s^2 + \\ \qquad + \left(\frac{1}{\omega_{cp}\omega_{c0}^2} - \frac{1}{\omega_{z0}^2\omega_z}\right)s^3 \end{cases} \quad (90)$$

Applicando (84) si ottiene la costante coincidente con il guadagno del Loop Filter K:

$$\omega_K^2 = \frac{1}{d_{A_2}} = \left(\frac{1}{\omega_{c0}^2} + \frac{1}{\omega_{cp}\omega_{c0}Q} - \frac{1}{\omega_{z0}^2}\right)^{-1} \quad (91)$$

Dalla (81):

$$X_A(s) = 1 + \omega_K^2\left(\frac{1}{\omega_{cp}\omega_{c0}^2} - \frac{1}{\omega_{z0}^2\omega_z}\right)s \quad (92)$$

Il parametro restante del modello di $H(s)$, ovvero ω_P, si determina mediante la radice del polinomio $X_A(s)$. In questo caso è immediato uguagliare uno ad uno i coefficienti dei polinomi $X_A(s)$ e $D_H(s)$:

$$\frac{1}{\omega_P} = \frac{1}{\omega_{cp} + \frac{\omega_{c0}}{Q} + \omega_z - \frac{\omega_{cp}\omega_{c0}^2}{\omega_{z0}^2} - \frac{\omega_{z0}^2\omega_z}{\omega_{c0}^2} - \frac{\omega_{z0}^2\omega_z}{\omega_{cp}\omega_{c0}Q}} \tag{93}$$

Nel caso del 3° ordine, il prototipo di $G(s)$ è esprimibile come:

$$\begin{cases} N_{\hat{G}}(s) = 1 + \left(\frac{1}{\omega_{z0}^2}\right)s^2 \\ D_{\hat{G}}(s) = 1 + \left(\frac{1}{\omega_{c0}Q} + \frac{1}{\omega_{c1}}\right)s + \left(\frac{1}{\omega_{c0}^2} + \frac{1}{\omega_{c1}\omega_{c0}Q}\right)s^2 + \left(\frac{1}{\omega_{c1}\omega_{c0}^2}\right)s^3 \end{cases} \tag{94}$$

Utilizzando la (80) si ha:

$$\begin{cases} N_G(s) = 1 + \left(\frac{1}{\omega_z}\right)s + \left(\frac{1}{\omega_{z0}^2}\right)s^2 + \left(\frac{1}{\omega_{z0}^2\omega_z}\right)s^3 \\ D_G(s) = 1 + \left(\frac{1}{\omega_{c0}Q} + \frac{1}{\omega_{c1}} + \frac{1}{\omega_{cp}}\right)s + \\ \left(\frac{1}{\omega_{c0}^2} + \frac{1}{\omega_{c1}\omega_{c0}Q} + \frac{1}{\omega_{cp}\omega_{c0}Q} + \frac{1}{\omega_{cp}\omega_{c1}}\right)s^2 + \\ + \left(\frac{1}{\omega_{c1}\omega_{c0}^2} + \frac{1}{\omega_{cp}\omega_{c0}^2} + \frac{1}{\omega_{cp}\omega_{c1}\omega_{c0}Q}\right)s^3 + \left(\frac{1}{\omega_{cp}\omega_{c1}\omega_{c0}^2}\right)s^4 \end{cases} \tag{95}$$

Applicando la (82) si ottiene il la pulsazione naturale del polo aggiuntivo:

$$\omega_{cp} = \frac{1}{\frac{1}{\omega_z} - \frac{1}{\omega_{c0}Q} - \frac{1}{\omega_{c1}}} \tag{96}$$

quindi dalla (83):

$$\begin{cases} N_A(s) = 1 + \left(\frac{1}{\omega_z}\right)s + \left(\frac{1}{\omega_{z0}^2}\right)s^2 + \left(\frac{1}{\omega_{z0}^2\omega_z}\right)s^3 \\ D_A(s) = \left(\frac{1}{\omega_{c0}Q} + \frac{1}{\omega_{c1}} + \frac{1}{\omega_{cp}} - \frac{1}{\omega_z}\right)s + \\ + \left(\frac{1}{\omega_{c0}^2} + \frac{1}{\omega_{c1}\omega_{c0}Q} + \frac{1}{\omega_{cp}\omega_{c0}Q} + \frac{1}{\omega_{cp}\omega_{c1}} - \frac{1}{\omega_{z0}^2}\right)s^2 + \\ + \left(\frac{1}{\omega_{c1}\omega_{c0}^2} + \frac{1}{\omega_{cp}\omega_{c0}^2} + \frac{1}{\omega_{cp}\omega_{c1}\omega_{c0}Q} - \frac{1}{\omega_{z0}^2\omega_z}\right)s^3 + \left(\frac{1}{\omega_{cp}\omega_{c1}\omega_{c0}^2}\right)s^4 \end{cases} \tag{97}$$

Applicando (84) si ottiene la costante coincidente con il guadagno del Loop Filter K:

$$\omega_K^2 = \left(\frac{1}{\omega_{c0}^2} + \frac{1}{\omega_{c1}\omega_{c0}Q} + \frac{1}{\omega_{cp}\omega_{c0}Q} + \frac{1}{\omega_{cp}\omega_{c1}} - \frac{1}{\omega_{z0}^2}\right)^{-1} \tag{98}$$

Dalla (81):

$$X_A(s) = \left(\frac{\omega_K^2}{\omega_{cp}\omega_{c1}\omega_{c0}^2}\right)s^2 + \omega_K^2\left(\frac{1}{\omega_{c1}\omega_{c0}^2} + \frac{1}{\omega_{cp}\omega_{c0}^2} + \frac{1}{\omega_{cp}\omega_{c1}\omega_{c0}Q} - \frac{1}{\omega_{z0}^2\omega_z}\right)s + 1 \quad (99)$$

I parametri restanti del modello di $H(s)$, ovvero ω_P e Q_P, si determinano mediante le radici del polinomio $X_A(s)$. In questo caso è immediato uguagliare uno ad uno i coefficienti dei polinomi $X_A(s)$ e $D_H(s)$:

$$\begin{cases} \dfrac{1}{\omega_P^2} = \dfrac{\omega_K^2}{\omega_{cp}\omega_{c1}\omega_{c0}^2} = \dfrac{1}{\omega_{cp}\omega_{c1} + \dfrac{\omega_{cp}\omega_{c0}}{Q} + \dfrac{\omega_{c1}\omega_{c0}}{Q} + \omega_{c0}^2 - \dfrac{\omega_{cp}\omega_{c1}\omega_{c0}^2}{\omega_{z0}^2}} \\ \dfrac{1}{\omega_P Q_P} = \omega_K^2\left(\dfrac{1}{\omega_{c1}\omega_{c0}^2} + \dfrac{1}{\omega_{cp}\omega_{c0}^2} + \dfrac{1}{\omega_{cp}\omega_{c1}\omega_{c0}Q} - \dfrac{1}{\omega_{z0}^2\omega_z}\right) \\ \quad = \dfrac{\omega_{cp}\omega_{c1}\omega_{c0}^2}{\omega_P^2}\left(\dfrac{1}{\omega_{c1}\omega_{c0}^2} + \dfrac{1}{\omega_{cp}\omega_{c0}^2} + \dfrac{1}{\omega_{cp}\omega_{c1}\omega_{c0}Q} - \dfrac{1}{\omega_{z0}^2\omega_z}\right) \\ \quad = \dfrac{1}{\omega_P^2}\left(\omega_{cp} + \omega_{c1} + \dfrac{\omega_{c0}}{Q} - \dfrac{\omega_{cp}\omega_{c1}\omega_{c0}^2}{\omega_{z0}^2\omega_z}\right) \end{cases} \quad (100)$$

Nella tabella riassuntiva sottostante sono riportate le configurazioni dei parametri di $A(s)$ per le $G(s)$ di ordine 1, 2 e 3.

m	ω_{cp}	K	ω_P	Q_P
1	$\dfrac{\omega_z}{1-\dfrac{\omega_z}{\omega_{c0}}}$	$\omega_{cp}\omega_{c0}$	N/A	N/A
2	$\dfrac{\omega_z}{1-\dfrac{\omega_z}{\omega_{c0}Q}}$	$\dfrac{\omega_{c0}Q}{\dfrac{1}{\omega_{cp}}+\dfrac{Q}{\omega_{c0}}-\dfrac{\omega_{c0}Q}{\omega_{z0}^2}}$	$\dfrac{1}{K\left(\dfrac{1}{\omega_{cp}\omega_{c0}^2}-\dfrac{1}{\omega_{z0}^2\omega_z}\right)}$	N/A
3	$\dfrac{\omega_z}{1-\dfrac{\omega_z}{\omega_{c1}}-\dfrac{\omega_z}{\omega_{c0}Q}}$	$\dfrac{\omega_{c0}Q}{\dfrac{1}{\omega_{cp}}+\dfrac{1}{\omega_{c1}}+Q\left(\dfrac{1}{\omega_{c0}}+\dfrac{\omega_{c0}}{\omega_{cp}\omega_{c1}}-\dfrac{\omega_{c0}}{\omega_{z0}^2}\right)}$	$\omega_{c0}\sqrt{\dfrac{\omega_{cp}\omega_{c1}}{K}}$	Q_{P_3}
$Q_{P\_3} = \dfrac{\omega_P Q}{\omega_{c0} + Q\left(\omega_{cp} + \omega_{c1} - \dfrac{\omega_{cp}\omega_{c1}\omega_{c0}^2}{\omega_{z0}^2\omega_z}\right)}$				

Tabella 8: configurazione dei parametri del Loop Filter che determina la G(s) avente anche zeri desiderata, per ordini 1, 2, 3 per PLL di tipo II

Capitolo 2.6: Generalizzazione del metodo per ogni ordine

Come anticipato precedentemente, il contributo originale rispetto al software a codice chiuso prodotto da Perrott, utilizzato come riferimento in grado di gestire al massimo filtri del terzo ordine privi di zeri quali Butterworth, Bessel e Chebyshev I, è quello di gestire anche comportamenti di ordine superiore sia per i filtri privi di zeri, sia per filtri aventi zeri quali Chebyshev II ed ellittici.

In realtà, il metodo alternativo ha potenzialità maggiori, in quanto qualunque caratterizzazione della $G(s)$ può essere gestita, ma in questo contesto verrà riportata una generalizzazione valida per qualsiasi ordine per le cinque forme sopra citate.

Nella pratica, non ha molto senso cercare di progettare dinamiche così articolate da richiedere un ordine così elevato, ma in alcuni casi può rendersi necessario andare a replicare risultati prodotti da altri, in cui si ha una caratterizzazione della funzione di trasferimento $G(s)$ che non corrisponde ad un modello standard implementato nel software di riferimento.

La generalizzazione per qualsiasi ordine m del PLL tiene conto della disposizione e del numero di poli e di zeri delle differenti funzioni di trasferimento.

I parametri del modello sono definiti assumendo che gli eventuali zeri del prototipo di $G(s)$ siano coppie di zeri complessi coniugati a parte reale nulla, come nei casi considerati, ma la parametrizzazione è facilmente estendibile anche nel caso di caratterizzazioni non standard che presentano zeri reali o coppie di zeri complessi coniugati a parte reale non nulla.

Nella definizione dei parametri del modello si è già imposto che gli zeri non aggiuntivi di $H(s)$ e di $A(s)$ coincidano con gli zeri del prototipo di $G(s)$, per cui sono indicati nello stesso modo.

Nel caso di PLL di tipo I, i modelli di *H*(*s*), *A*(*s*) e *G*(*s*) generalizzati per ogni ordine di *G*(*s*) sono:

$H(s)$	$\frac{K_{LP}\prod_{i=0}^{l_{zccH}-1}\left(1+\frac{s^2}{\omega_{z0i}^2}\right)}{\prod_{i=0}^{l_{prH}-1}\left(1+\frac{s}{\omega_{Pi}}\right)\prod_{i=l_{prH}}^{l_H}\left(1+\frac{s}{\omega_{Pi}Q_{Pi}}+\frac{s^2}{\omega_{Pi}^2}\right)}$
$A(s)$	$\frac{K\prod_{i=0}^{l_{zccA}-1}\left(1+\frac{s^2}{\omega_{z0i}^2}\right)}{s\prod_{i=0}^{l_{prA}-1}\left(1+\frac{s}{\omega_{Pi}}\right)\prod_{i=l_{prA}}^{l_A}\left(1+\frac{s}{\omega_{Pi}Q_{Pi}}+\frac{s^2}{\omega_{Pi}^2}\right)}$
$G(s)$	$\frac{\prod_{i=0}^{l_{zccG}-1}\left(1+\frac{s^2}{\omega_{z0i}^2}\right)}{\prod_{i=0}^{l_{prG}-1}\left(1+\frac{s}{\omega_{ci}}\right)\prod_{i=l_{prG}}^{l_G}\left(1+\frac{s}{\omega_{ci}Q_i}+\frac{s^2}{\omega_{ci}^2}\right)}$

Tabella 9: modelli generalizzati per ogni ordine delle funzioni di trasferimento H(s), A(s) e G(s) per PLL di tipo I, con G(s) avente anche zeri

Nel caso di PLL di tipo II di *H*(*s*), *A*(*s*) e *G*(*s*) i modelli generalizzati per ogni ordine di *G*(*s*) sono:

$H(s)$	$\frac{K_{LP}\left(1+\frac{s}{\omega_z}\right)\prod_{i=0}^{l_{zccH}-1}\left(1+\frac{s^2}{\omega_{z0i}^2}\right)}{\prod_{i=0}^{l_{prH}-1}\left(1+\frac{s}{\omega_{Pi}}\right)\prod_{i=l_{prH}}^{l_H}\left(1+\frac{s}{\omega_{Pi}Q_{Pi}}+\frac{s^2}{\omega_{Pi}^2}\right)}$
$A(s)$	$\frac{K\left(1+\frac{s}{\omega_z}\right)\prod_{i=0}^{l_{zccA}-1}\left(1+\frac{s^2}{\omega_{z0i}^2}\right)}{s^2\prod_{i=0}^{l_{prA}-1}\left(1+\frac{s}{\omega_{Pi}}\right)\prod_{i=l_{prA}}^{l_A}\left(1+\frac{s}{\omega_{Pi}Q_{Pi}}+\frac{s^2}{\omega_{Pi}^2}\right)}$
$G(s)$	$\frac{\left(1+\frac{s}{\omega_z}\right)\prod_{i=0}^{l_{zccG}-1}\left(1+\frac{s^2}{\omega_{z0i}^2}\right)}{\left(1+\frac{s}{\omega_{cp}}\right)\prod_{i=0}^{l_{prG}-1}\left(1+\frac{s}{\omega_{ci}}\right)\prod_{i=l_{prG}}^{l_G}\left(1+\frac{s}{\omega_{ci}Q_i}+\frac{s^2}{\omega_{ci}^2}\right)}$

Tabella 10: modelli generalizzati per ogni ordine delle funzioni di trasferimento H(s), A(s) e G(s) per PLL di tipo II, con G(s) avente anche zeri

Dove:

- K_{LP} è il guadagno di $H(s)$, legato alla variabile di controllo K dalla relazione (38);
- ω_{Pi} è una variabile di controllo del sistema che rappresenta la i-esima pulsazione naturale dei poli reali o delle coppie di poli complessi coniugati di $H(s)$ e di $A(s)$;
- Q_{Pi} è una variabile di controllo del sistema che rappresenta l'i-esimo fattore di merito delle coppie di poli complessi coniugati di $H(s)$ e di $A(s)$;
- ω_{z0i} è una variabile di controllo del sistema che rappresenta l'i-esima pulsazione naturale degli zeri complessi coniugati di $H(s)$ e di $A(s)$, che necessariamente coincide con quella ottenuta dal prototipo di $G(s)$;
- K è una variabile di controllo del sistema che rappresenta il guadagno di $A(s)$;
- ω_{ci} è la i-esima pulsazione naturale dei poli reali o delle coppie di poli complessi coniugati del prototipo di $G(s)$;
- Q_i è l'i-esimo fattore di merito delle coppie di poli complessi coniugati del prototipo di $G(s)$;
- ω_{cp} è la pulsazione naturale del polo aggiuntivo introdotto dall'ulteriore stadio di integrazione presente nei PLL di tipo II;
- ω_{ci} è la pulsazione naturale dello zero aggiunto per garantire un adeguato margine di fase nei PLL di tipo II;
- gli indici delle produttorie dipendono dall'ordine e dalla forma del prototipo di $G(s)$ secondo le tabelle riportate di seguito, dove m indica l'ordine del prototipo di $G(s)$

m	l_{prH}, l_{prA}	l_H, l_A	l_{prG}	l_G
$pari$	0	$\frac{m}{2}-1$	1	$\frac{m}{2}$
$dispari$	1	$\frac{m}{2}$	0	$\frac{m}{2}-1$

Tabella 11: estremi delle produttorie al denominatore dei modelli di H(s), A(s) e G(s).

shape	$l_{zccH,A}, l_{zccG}$
Butterworth, Bessel, Chebyshev I	0
Chebyshev II, Ellittico	$\left\lfloor\frac{m}{2}\right\rfloor$

Tabella 12: estremi delle produttorie al numeratore dei modelli di H(s), A(s) e G(s).

La determinazione delle variabili di controllo del sistema, ovvero dei parametri del modello della funzione di trasferimento ad anello aperto $A(s)$ è difficilmente ottenibile a mano, per questo risulta comodo sfruttare le potenzialità di calcolo numerico di software quali Python o Matlab.

In particolare, il software, date le specifiche della $G(s)$, dovrà determinare le variabili di controllo mediante la procedura descritta nel metodo alternativo proposto nel Capitolo 2.5.

Nel caso di PLL di tipo I, il guadagno K di $A(s)$ si determina imponendo $K = \omega_K$ definita in (61). Gli altri parametri si ottengono determinando le radici dei polinomi $N_A(s)$ e $X_A(s)$.

In maniera analoga, nel caso di PLL di tipo II, il guadagno K di $A(s)$ si determina imponendo $K = \omega_K{}^2$ definita nella (84) e ω_{cp} tramite la (82). Gli altri parametri si ottengono determinando le radici dei polinomi $N_A(s)$ e $X_A(s)$.

Capitolo 3

ANALISI DEL RUMORE

Un sintetizzatore di frequenza ideale genera in uscita $out(t)$ una perfetta sinusoide alla frequenza f_{out}. Nella realtà i componenti che lo costituiscono introducono rumore, con l'effetto di spargere la potenza del segnale in uscita anche alle frequenze adiacenti alla f_{out}.

Il modello nel dominio delle frequenze del PLL, definito mediante la funzione di trasferimento descrittiva $G(s)$, consente di determinare in modo diretto le performance del sistema da punto di vista del rumore. In particolare, è possibile considerare sorgenti di rumore inserite in qualunque punto dell'anello che modellano il comportamento rumoroso dei componenti e valutare come il rumore prodotto nei vari punti della catena si riflette sull'uscita del sistema. Si ricorda che $s = j2\pi f$, dove f non è intesa come frequenza del segnale in uscita del PLL, ma la frequenza alla quale la fase del riferimento è cambiata. In altri termini è la frequenza di *offset* f_{off} o deviazione rispetto alla frequenza di uscita nominale del PLL.

La presenza di rumore nei componenti determina rumore di fase in uscita dal PLL, che può essere espresso nelle due versioni *single sideband* (singola banda adiacente alla portante f_{out}) e *double sideband* (doppia banda adiacente alla portante), ma l'IEEE ha adottato come standard la versione single sideband, indicata con $L(f)$ ed espressa in dBc/Hz. Il rumore di fase $L(f)$ è la potenza del segnale in uscita valutata all'interno di una banda di 1Hz centrata ad una frequenza pari a f_{off} distante dalla portante f_{out}, rapportata alla potenza totale del segnale portante. Questo dipenderà dalla sorgente di rumore e dalla corrispondente funzione di trasferimento.

Un'altra grandezza rappresentativa del rumore di fase è il *jitter*. Il jitter consente di apprezzare l'incertezza temporale di un evento, in particolare l'incertezza temporale con cui si collocano i fronti o il passaggio per lo zero di un segnale che dovrebbe essere periodico.

Se si considera il segnale in uscita dal sintetizzatore di frequenza come:

$$out(t) = out'\big(t + jit(t)\big) \tag{101}$$

dove $jit(t)$ è realizzazione di un processo stocastico avente valore medio nullo rappresentativa dell'effetto del rumore e la funzione $out'(\,)$ è periodica con periodo pari a $1/f_{out}$.

La funzione $jit(t)$ è quindi il jitter, ovvero l'effetto del rumore nel dominio del tempo.

Il legame con la fase del segnale in uscita è:

$$\varphi_{out}(t) = 2\pi f_{out}\, jit(t) \tag{102}$$

quindi:

$$out(t) = out'\left(t + \frac{\varphi_{out}(t)}{2\pi f_{out}}\right) \tag{103}$$

Lo scarto quadratico medio del jitter, più brevemente indicato con jit_{rms}, si ottiene a partire dal rumore di fase single banded $L(f)$, tramite la relazione:

$$jit_{rms} = \frac{\sqrt{2\int_{f_{off\_max}}^{f_{off\_max}} L_{out}(f)\,\mathrm{d}f}}{2\pi f_{out}} \tag{104}$$

dove [f_{off_min} , f_{off_max}] è l'intervallo di integrazione e si è tenuto conto del legame fra lo spettro di potenza del rumore di fase $S_{\varphi out}(f)$ e $L_{out}(f)$:

$$L_{out}(f) = \frac{1}{2} S_{\varphi_{out}}(f) \tag{105}$$

L'approccio basato sull'analisi nel dominio delle frequenze/fase di un modello lineare come $G(s)$ è più semplice rispetto all'analisi nel dominio del tempo, perché è possibile analizzare separatamente le diverse sorgenti di rumore.

Le sorgenti di rumore del sintetizzatore di frequenza sono indicate in Figura 22.

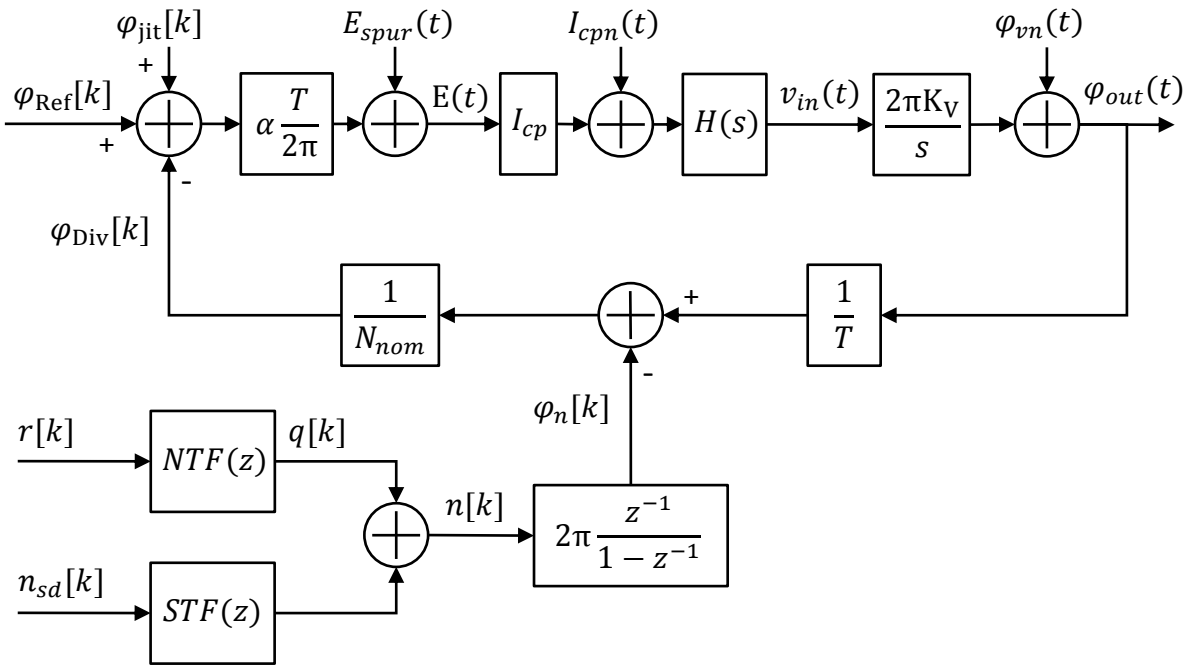

Figura 22: Sorgenti di rumore del PLL

dove:

- $\varphi_{jit}[k]$ è la componente di jitter introdotta dal segnale di riferimento e dal divisore, dovuta alle transizioni non istantanee dei segnali Ref[k] e Div[k]. È caratterizzata da uno spettro di potenza bianco come riportato in Figura 23:

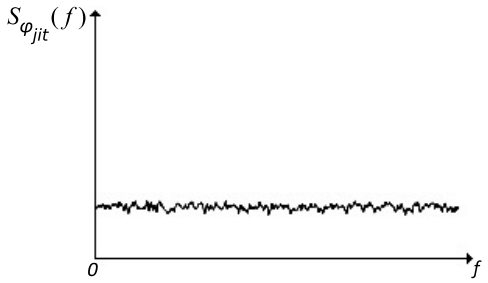

Figura 23: Spettro di potenza del rumore introdotto dal segnale di riferimento e dal divisore

- $E_{spur}(t)$ è una componente spuria periodica alla frequenza parti ad $1/T$. É causata dall'utilizzo del PFD con architettura XOR o con architettura TRISTATE quando il duty cycle dell'uscita E(t) non è nullo. Il segnale $E_{spur}(t)$, essendo periodico, avrà uno spettro di potenza come riportato in Figura 24:

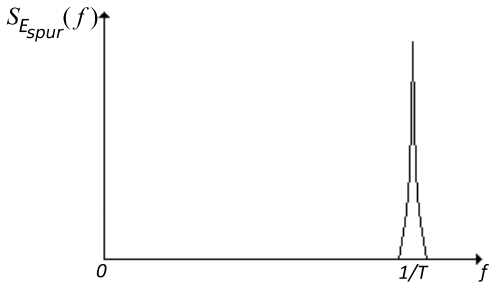

Figura 24: Spettro di potenza del rumore introdotto dal PFD

- $I_{cpn}(t)$ è il segnale rumore introdotto dai transistor che compongono la pompa di carica. È caratterizzato da uno spettro di potenza come quello riportato in Figura 25:

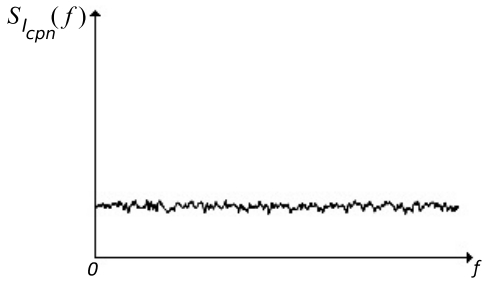

Figura 25: Spettro di potenza del rumore introdotto dalla pompa di carica

- $\varphi_{vn}(t)$ è il rumore introdotto dal VCO e dal Loop Filter avente spettro di potenza come rappresentato in Figura 26:

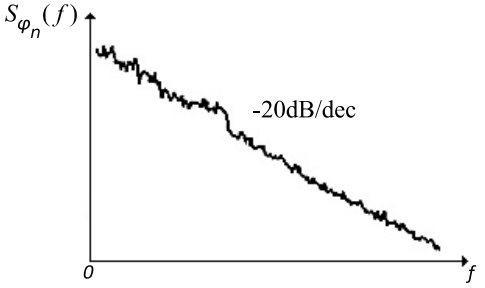

Figura 26: Spettro di potenza del rumore introdotto dal Loop Filter e dal VCO

- $r[k]$ è il rumore di quantizzazione avente uno spettro di potenza bianco uniformemente distribuito nella banda $F = [0,1]$ con $F=fT$ e con ampiezza pari a 1/12.

- $q[k]$ è il rumore di quantizzazione in uscita dal modulatore Σ-Δ.

- $\varphi_n[k]$ è un segnale astratto interno al divisore che rappresenta l'uscita dall'accumulatore digitale, utile per il calcolo del rumore di fase introdotto dall'azione combinata del modulatore Σ-Δ e del divisore di frequenza.

Capitolo 3.1: Tecnica di noise shaping del rumore di quantizzazione mediante la scelta dell'architettura del modulatore Σ-Δ

Il modulatore Σ-Δ presenta un'architettura puramente digitale la cui uscita è commutata velocemente fra pochi valori in modo che il valore medio della sequenza corrisponda al segnale d'ingresso avente alta risoluzione e con energia confinata alle basse frequenze. Grazie a questo, il rumore di quantizzazione viene spostato a frequenze elevate (tecnica di noise shaping), per cui lo spettro del rumore di quantizzazione che si propaga dall'uscita dal modulatore Σ-Δ avrà un andamento come quello riportato in Figura 27:

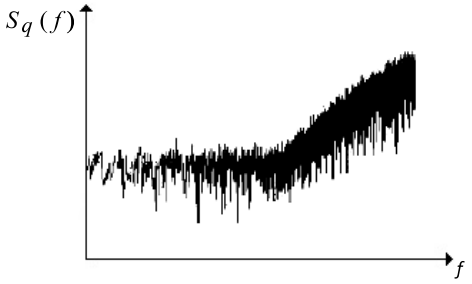

Figura 27: Esempio di spettro di potenza del rumore quantizzazione in uscita dal modulatore Σ-Δ

Il modello del modulatore Σ-Δ si basa sull'ipotesi che il rumore di quantizzazione sia indipendente dall'ingresso e che presenti uno spettro di potenza bianco. Risulta conveniente parametrizzare il modulatore Σ-Δ attraverso due funzioni di trasferimento, ovvero la *signal transfer function STF*(*z*) (funzione di trasferimento riferita al segnale d'ingresso) e la *noise transfer function NTF*(*z*) (funzione di trasferimento riferita al rumore).

Per cui lo spettro del rumore di quantizzazione in uscita dal modulatore Σ-Δ è determinato direttamente dalla *NTF*(*z*) del modulatore stesso.

Questo aspetto del rumore di quantizzazione è particolarmente interessante, perché a differenza di quello che accade per tutte le altre sorgenti di rumore, si hanno maggiori possibilità di controllo. Ovvero il contributo del rumore di quantizzazione sul rumore di fase in uscita dal sintetizzatore dipende sia dal modo in cui il rumore di quantizzazione in uscita dal modulatore Σ-Δ si propaga verso l'uscita, sia dall'architettura stessa modulatore Σ-Δ mediante la sua *NTF*(*z*).

In particolare, se il rumore di quantizzazione in uscita dal modulatore Σ-Δ subisse un azione di filtraggio di tipo passa basso nel modo in cui questo viene propagato in uscita dal sintetizzatore, con l'azione combinata di noise shaping del modulatore stesso è possibile ridurre l'effetto del rumore di quantizzazione sul rumore di fase in uscita.

Per cui, facendo in modo che il modulatore Σ-Δ converta il segnale ad alta risoluzione con una frequenza molto maggiore rispetto alla frequenza di taglio dell'azione filtrante di tipo passa basso, quest'ultima estrarrà solo la componente a bassa frequenza del segnale, definito da $n_{sd}[k]$, che può avere una risoluzione arbitraria. Questo consente al sintetizzatore di essere controllato con alta risoluzione indipendentemente dalla frequenza del segnale di riferimento, per cui è possibile avere alta risoluzione della frequenza in uscita anche con alta frequenza del segnale di riferimento.

Questo aspetto apre la possibilità di coprogettare il modulatore Σ-Δ e il suo quantizzatore in modo che il rumore di quantizzazione in uscita dallo stesso sia il migliore possibile in relazione a come il PLL lo propaga in uscita.

A seguire sono riportate alcune possibili architetture del modulatore Σ-Δ che è possibile incorporare nei sintetizzatori di frequenza a rapporto frazionario *N*.

Modulatore Σ-Δ standard a singolo stadio del primo ordine

L'architettura del modulatore Σ-Δ standard a singolo stadio del primo ordine ha una funzione di trasferimento descritta dal diagramma a blocchi sottostante:

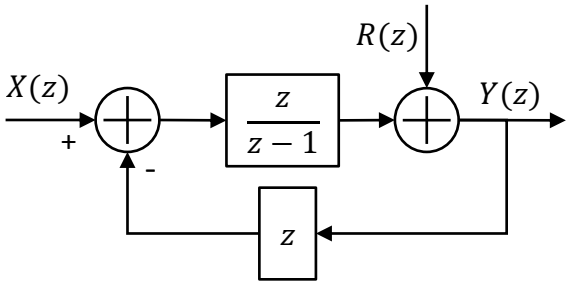

Figura 28: Diagramma del modello di un modulatore Σ-Δ standard

dove $X(z)$ è la trasformata Z dell'ingresso del modulatore, $Y(z)$ è la trasformata Z dell'uscita del modulatore e $R(z)$ è la trasformata Z del rumore di quantizzazione.

La funzione di trasferimento è descritta da:

$$Y(z) = X(z) + (1 - z^{-1})R(z) \qquad (106)$$

Scomponendola nelle due funzioni di trasferimento del segnale e del rumore, si ha:

$$STF(z) = \frac{Y(z)}{X(z)} = 1 \qquad (107)$$

$$NTF(z) = \frac{Y(z)}{R(z)} = 1 - z^{-1} \qquad (108)$$

La $NTF(z)$ si comporta come una derivata discreta, ovvero la funzione di trasferimento di un filtro passa alto.

Per ordini $m_{\Sigma\Delta}$ maggiori di uno la (106) diventa:

$$Y(z) = X(z) + (1 - z^{-1})^{m_{\Sigma\Delta}} R(z) \qquad (109)$$

Il problema di questa architettura è che per ordini maggiori o uguali a 3 il sistema diventa instabile.

Modulatore Σ-Δ Multi Stage Noise Shaping (MASH)

Una possibile alternativa al modulatore Σ-Δ standard è l'architettura *Multi Stage Noise Shaping* MASH, costituita dalla cascata di modulatori Σ-Δ di basso ordine. Con questa architettura si ottiene un modulatore Σ-Δ con ordine elevato senza problemi di stabilità, a differenza dei modulatori realizzati con un singolo stadio.

Nella Figura 29 è riportato un esempio di architettura MASH del secondo ordine:

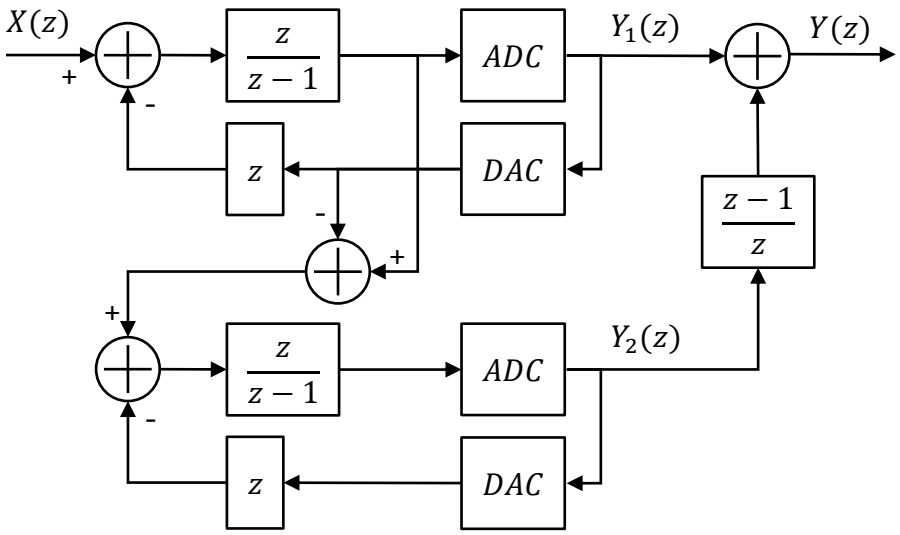

Figura 29: Architettura di un modulatore Σ-Δ MASH del secondo ordine

dove ADC è un *Analog-to-Digital Converter*, ovvero un convertitore analogico digitale che introduce rumore di quantizzazione $R(z)$, mentre DAC è un *Digital-to-Analog Coverter*, ovvero un convertitore digitale analogico.

La funzione di trasferimento del primo stadio è:

$$Y_1(z) = X(z) + (1 + z^{-1})R_1(z) \quad (110)$$

La funzione di trasferimento del secondo stadio è:

$$Y_2(z) = -R_1(z) + (1 + z^{-1})R_2(z) \quad (111)$$

Quella complessiva è:

$$Y(z) = Y_1(z) + (1 + z^{-1})Y_2(z) \quad (112)$$

Sostituendo le (110) e (111) e supponendo $R_1(z) = R_2(z) = R(z)$:

$$Y(z) = X(z) + (1 + z^{-1})^2 R(z) \quad (113)$$

Iterando l'architettura si ottiene un modulatore Σ-Δ con architettura di tipo MASH di ordine $m_{\Sigma\Delta}$:

$$Y(z) = X(z) + (1 - z^{-1})^{m_{\Sigma\Delta}} R(z) \quad (114)$$

Per cui in questo caso si ha:

$$STF(z) = \frac{Y(z)}{X(z)} = 1 \quad (115)$$

$$NTF(z) = \frac{Y(z)}{R(z)} = (1 - z^{-1})^{m_{\Sigma\Delta}} \quad (116)$$

Le funzioni di trasferimento $STF(z)$ e $NTF(z)$ sono le medesime del modulatore Σ-Δ standard di ordine $m_{\Sigma\Delta}$, ma in questo caso la struttura è stabile. L'uscita $Y(t)$ è un segnale digitale a $m_{\Sigma\Delta}$ bit ovvero $2^{m\Sigma\Delta}$ livelli a differenza del modulatore Σ-Δ standard in cui l'uscita aveva un solo bit. Affinché l'architettura funzioni è necessario che gli stadi abbiano le medesime caratteristiche.

Modulatore Σ-Δ Error Feedback Structured (EFS)

Il modulatore Σ-Δ *Error Feedback Structured* (EFS), mediante un meccanismo di retroazione, cerca di annullare gli effetti del rumore di quantizzazione, riportando sul ramo di retroazione solo l'errore di quantizzazione e non il segnale di uscita.

Il modulatore Σ-Δ EFS ha un'architettura rappresentata dal diagramma a blocchi sottostante:

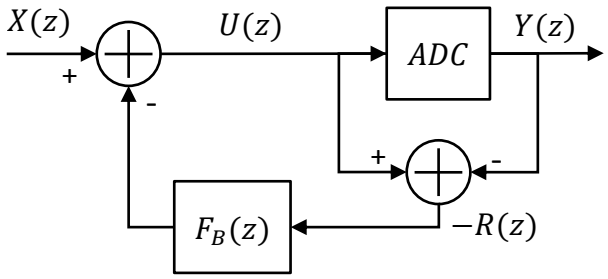

Figura 30: Diagramma architettura di un modulatore Σ-Δ Error Feedback Structured

Il cui modello lineare è:

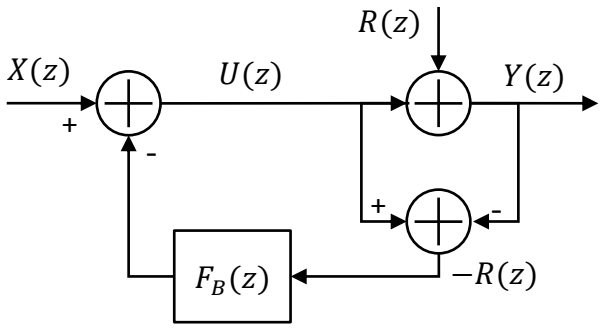

Figura 31: Modello del modulatore Σ-Δ Error Feedback Structured

dove $R(z)$ è la trasformata Z del rumore di quantizzazione introdotto dal convertitore analogico digitale (per esempio un semplice quantizzatore).

Dal modello di Figura 31, si ottiene:

$$\begin{cases} Y(z) = U(z) + R(z) \\ U(z) = X(z) - F_B(z)R(z) \end{cases} \quad (117)$$

Per cui:

$$Y(z) = X(z) + \big(1 - F_B(z)\big)R(z) \quad (118)$$

Scomponendo le due funzioni di trasferimento del segnale e del rumore:

$$STF(z) = \frac{Y(z)}{X(z)} = 1 \quad (119)$$

$$NTF(z) = \frac{Y(z)}{R(z)} = 1 - F_B(z) \quad (120)$$

Se si impone che la *NTF*(*z*) abbia una risposta di tipo *Finite Impulse Response* FIR, ovvero esprimibile come:

$$NTF(z) = \sum_{k=0}^{M-1} b_k z^{-k} \quad (121)$$

dove *M*-1 è l'ordine del filtro digitale, *M* è il numero dei coefficienti del filtro FIR, allora il sistema è stabile, dato che in un filtro FIR ci sono *M*-1 poli nulli.

I coefficienti del filtro FIR sono indicati mediante una sequenza [b_0, b_1, …, b_{M-1}] e b_0 deve essere unitario, in modo che il polinomio sia monico.

Tramite la (120) e la (121) è possibile determinare la $F_B(z)$ che determina una *NTF*(*z*) di tipo FIR:

$$F_B(z) = \sum_{k=1}^{M-1} -b_k z^{-k} \quad (122)$$

Capitolo 3.2: Determinazione delle funzioni di trasferimento relative alle singole sorgenti di rumore

Per determinare l'effetto delle singole sorgenti di rumore sul rumore di fase in uscita dal sintetizzatore di frequenze, è necessario determinare ciascuna funzione di trasferimento.

Considerando come sorgente il rumore $\varphi_{jit}[k]$ introdotto dal segnale di riferimento e dall'uscita del divisore e annullando tutti gli ingressi e le altre sorgenti di rumore, si ottiene:

$$\mathcal{L}\{\varphi_{out}(t)\} = (\alpha T) I_{cp} H(s) \left(\frac{\mathrm{K_V}}{s}\right) \left(\mathcal{L}\{\varphi_{jit}[k]\} - \left(\frac{1}{N_{nom}}\right)\left(\frac{1}{T}\right)\mathcal{L}\{\varphi_{out}(t)\}\right) \quad (123)$$

Sfruttando la parametrizzazione espressa in (35) diventa:

$$\mathcal{L}\{\varphi_{out}(t)\} = T N_{nom} A(s) \mathcal{L}\{\varphi_{jit}[k]\} - A(s)\mathcal{L}\{\varphi_{out}(t)\} \quad (124)$$

In base alla (29) si ottiene la funzione di trasferimento:

$$G_{jit}(s) = \frac{\mathcal{L}\{\varphi_{out}(t)\}}{\mathcal{L}\{\varphi_{jit}[k]\}} = \frac{T N_{nom} A(s)}{1 + A(s)} = T N_{nom} G(s) \quad (125)$$

Considerando come sorgente il rumore $E_{spur}(t)$ introdotto dal PFD e annullando tutti gli ingressi e le altre sorgenti di rumore, si ottiene:

$$\mathcal{L}\{\varphi_{out}(t)\} = I_{cp} H(s) \left(\frac{2\pi \mathrm{K_V}}{s}\right) \left(\mathcal{L}\{E_{spur}(t)\} - \left(\frac{\alpha}{2\pi}\right)\left(\frac{1}{N_{nom}}\right)\mathcal{L}\{\varphi_{out}(t)\}\right) \quad (126)$$

Sfruttando la parametrizzazione espressa in (35) diventa:

$$\mathcal{L}\{\varphi_{out}(t)\} = \left(\frac{2\pi}{\alpha}\right) N_{nom} A(s) \mathcal{L}\{E_{spur}(t)\} - A(s)\mathcal{L}\{\varphi_{out}(t)\} \quad (127)$$

In base alla (29) si ottiene la funzione di trasferimento:

$$G_{spur}(s) = \frac{\mathcal{L}\{\varphi_{out}(t)\}}{\mathcal{L}\{E_{spur}(t)\}} = \left(\frac{2\pi}{\alpha}\right) N_{nom} \frac{A(s)}{1 + A(s)} = \left(\frac{2\pi}{\alpha}\right) N_{nom} G(s) \quad (128)$$

Considerando come sorgente il rumore $I_{cpn}(t)$ introdotto dalla pompa di carica e annullando tutti gli ingressi e le altre sorgenti di rumore, si ottiene:

$$\mathcal{L}\{\varphi_{out}(t)\} = H(s) \left(\frac{2\pi \mathrm{K_V}}{s}\right) \left(\mathcal{L}\{I_{cpn}(t)\} - I_{cp} \left(\frac{\alpha}{2\pi}\right)\left(\frac{1}{N_{nom}}\right)\mathcal{L}\{\varphi_{out}(t)\}\right) \quad (129)$$

Sfruttando la parametrizzazione espressa in (35) diventa:

$$\mathcal{L}\{\varphi_{out}(t)\} = \left(\frac{2\pi}{\alpha}\right)\left(\frac{1}{I_{cp}}\right) N_{nom} A(s) \mathcal{L}\{I_{cpn}(t)\} - A(s)\mathcal{L}\{\varphi_{out}(t)\} \quad (130)$$

In base alla (29) si ottiene la funzione di trasferimento:

$$G_{cpn}(s) = \frac{\mathcal{L}\{\varphi_{out}(t)\}}{\mathcal{L}\{I_{cpn}(t)\}} = \left(\frac{2\pi}{\alpha}\right)\left(\frac{1}{I_{cp}}\right) N_{nom} \frac{A(s)}{1 + A(s)} = \left(\frac{2\pi}{\alpha}\right)\left(\frac{1}{I_{cp}}\right) N_{nom} G(s) \quad (131)$$

Considerando come sorgente il rumore $\varphi_{vn}(t)$ introdotto dal Loop Filter e dal VCO e annullando tutti gli ingressi e le altre sorgenti di rumore, si ottiene:

$$\mathcal{L}\{\varphi_{out}(t)\} = \mathcal{L}\{\varphi_{vn}(t)\} - \left(\frac{2\pi K_V}{s}\right) H(s) I_{cp} \left(\frac{\alpha}{2\pi}\right)\left(\frac{1}{N_{nom}}\right) \mathcal{L}\{\varphi_{out}(t)\} \quad (132)$$

Sfruttando la parametrizzazione espressa in (35) diventa:

$$\mathcal{L}\{\varphi_{out}(t)\} = \mathcal{L}\{\varphi_{vn}(t)\} - A(s)\mathcal{L}\{\varphi_{out}(t)\} \quad (133)$$

In base alla (29) si ottiene la funzione di trasferimento:

$$G_{vn}(s) = \frac{\mathcal{L}\{\varphi_{out}(t)\}}{\mathcal{L}\{\varphi_{vn}(t)\}} = \frac{1}{1 - A(s)} = 1 - G(s) \quad (134)$$

Risulta più semplice considerare il rumore introdotto dal Loop Filter e dal VCO riportato all'ingresso del VCO indicato con $\varphi_{vn}'(t)$. In questo caso si ottiene direttamente dalla (134):

$$G_{vn}'(s) = \frac{\mathcal{L}\{\varphi_{out}(t)\}}{\mathcal{L}\{\varphi_{vn}'(t)\}} = \left(\frac{2\pi K_V}{s}\right) G_{vn}(s) = \left(\frac{2\pi K_V}{s}\right)(1 - G(s)) \quad (135)$$

Considerando come sorgente il rumore astratto $\varphi_n[k]$ introdotto dall'azione combinata del modulatore Σ-Δ e del divisore di frequenza, si ottiene:

$$\mathcal{L}\{\varphi_{out}(t)\} = -(\alpha T) I_{cp} H(s) \left(\frac{K_V}{s}\right)\left(\left(\frac{1}{N_{nom}}\right)\left(\frac{1}{T}\right)\mathcal{L}\{\varphi_{out}(t)\} - \left(\frac{1}{N_{nom}}\right)\mathcal{L}\{\varphi_n[k]\}\right) \quad (136)$$

Sfruttando la parametrizzazione espressa in (35) diventa:

$$\mathcal{L}\{\varphi_{out}(t)\} = -A(s)\mathcal{L}\{\varphi_{out}(t)\} + TA(s)\mathcal{L}\{\varphi_n[k]\} \quad (137)$$

In base alla (29) si ottiene la funzione di trasferimento:

$$G_n(s) = \frac{\mathcal{L}\{\varphi_{out}(t)\}}{\mathcal{L}\{\varphi_n[k]\}} = \frac{TA(s)}{1 + A(s)} = TG(s) \qquad (138)$$

Noto:

$$z = e^{sT} \qquad (139)$$

la funzione di trasferimento dell'accumulatore digitale dovuto dalla natura integrante del divisore è:

$$G_{acc}(s) = \frac{\mathcal{L}\{\varphi_n[k]\}}{\mathcal{L}\{n[k]\}} = 2\pi \frac{e^{-sT}}{1 - e^{-sT}} \qquad (140)$$

Riassumendo le funzioni di trasferimento relative alle singole sorgenti di rumore con la Figura 32:

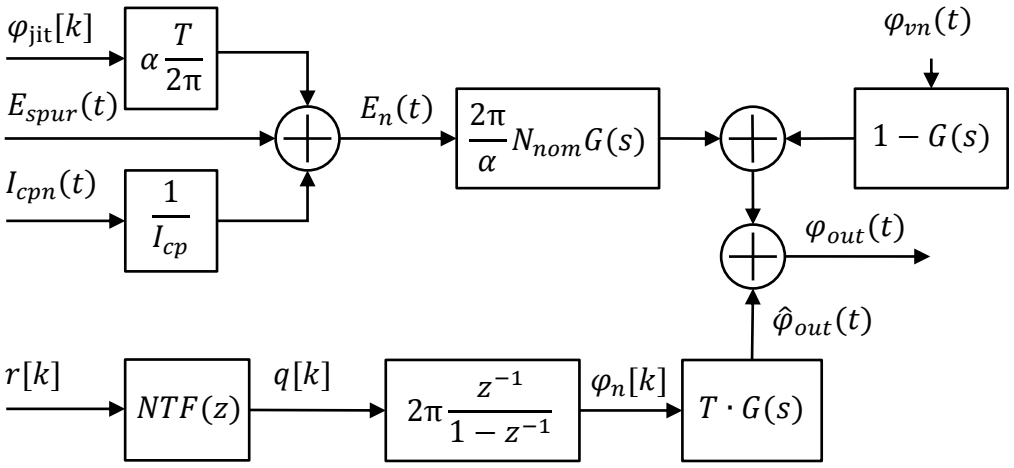

Figura 32: Riassunto funzioni di trasferimento relative alle singole sorgenti di rumore del sintetizzatore di frequenza

si nota che il rumore introdotto dal divisore e dal segnale di riferimento, dal PFD e dalla pompa di carica subisce un filtraggio di tipo passa basso tramite la dinamica del PLL descritta da $G(s)$. Risulta comodo definire il segnale $E_n(t)$ che raggruppa i tre contributi di rumore riportati all'uscita del PFD, motivo per il quale $E_n(t)$ prende il nome di *detector noise.*

Il rumore introdotto dal VCO, ovvero *VCO noise*, invece è soggetto ad un filtraggio di tipo passa alto 1 - $G(s)$.

Per questi motivi, se la banda del PLL f_0 è molto bassa, il rumore VCO noise rappresenterà la componente dominante in un'ampia gamma di frequenze maggiori della banda del PLL, mentre se f_0 è alta, il VCO noise sarà attenuato in un range di frequenze maggiore, a discapito del detector noise che non è attenuato in banda.

Il modulatore Σ-Δ, attraverso la *NTF*(*z*), modella il rumore di quantizzazione.

L'accumulatore digitale presente all'interno del divisore, introdotto nel modello per considerare la natura integrante del divisore, riduce di uno l'ordine della *NTF*(*z*).

Il contributo del rumore di quantizzazione sul rumore di fase in uscita è filtrato dalla dinamica del PLL mediante *G*(*s*), per cui le componenti ad alta frequenza introdotte dal modulatore Σ-Δ vengono attenuate.

Capitolo 3.3: Determinazione dei contributi delle singole sorgenti di rumore sul rumore di fase in uscita

Nel calcolo del rumore di fase dovuto alle singole sorgenti di rumore è necessario considerare che nell'architettura del sintetizzatore sono presenti segnali tempo continui e tempo discreti.

Dato un generico segnale *x* e il corrispondente segnale *y* ottenuto filtrando *x*:

- se *x*(*t*) è un segnale tempo continuo e il filtro è tempo continuo con funzione di trasferimento *TF*(*f*), l'uscita *y*(*t*) ha uno spettro di potenza dato da:

$$S_y(f) = |TF(f)|^2 S_x(f) \tag{141}$$

- se *x*[*k*] è un segnale tempo discreto e il filtro è tempo discreto con funzione di trasferimento $TF(e^{j2\pi fT})$, l'uscita *y*[*k*] ha uno spettro di potenza dato da:

$$S_y\left(e^{j2\pi fT}\right) = \left|TF\left(e^{j2\pi fT}\right)\right|^2 S_x\left(e^{j2\pi fT}\right) \tag{142}$$

- se *x*[*k*] è un segnale tempo discreto e il filtro è tempo continuo con funzione di trasferimento *TF*(*f*), assumendo *x* come un treno di impulsi modulati di periodo *T*, l'uscita *y*(*t*) ha uno spettro di potenza dato da:

$$S_y(f) = \frac{1}{T}|TF(f)|^2 S_x\left(e^{j2\pi fT}\right) \tag{143}$$

Lo spettro di potenza del rumore di fase in uscita è dato dalla somma dei contributi delle singole sorgenti di rumore:

$$S_{\varphi_{out}}(f) = S_{\varphi_{ref\_div}}(f) + S_{\varphi_{PFD}}(f) + S_{\varphi_{cp}}(f) + S_{\varphi_{VCO}}(f) + S_{\varphi_q}(f) \qquad (144)$$

Risulta conveniente raggruppare i primi tre contributi dello spettro di potenza del rumore di fase, considerando un'unica sorgente di rumore riferita all'uscita del PFD, ovvero il detector noise:

$$S_{\varphi_{out}}(f) = S_{\varphi_{dn}}(f) + S_{\varphi_{VCO}}(f) + S_{\varphi_q}(f) \qquad (145)$$

Il contributo del detector noise sullo spettro di potenza del rumore di fase in uscita, considerando $En(t)$ tempo continuo in quanto risultato di una modulazione a treno di impulsi, si ottiene osservando la Figura 32 e la relazione (141):

$$S_{\varphi_{dn}}(f) = \left(\frac{2\pi}{\alpha} N_{nom}\right)^2 |G(\mathrm{j}2\pi f)|^2 S_{E_n}(f) \qquad (146)$$

dove $S_{En}(f)$ è lo spettro di potenza del detector noise.

Il contributo del VCO noise sullo spettro di potenza del rumore di fase di uscita, considerando per convenienza lo spettro di potenza della sorgente di rumore riportata all'ingresso del VCO, si ottiene dalle relazioni espresse in (141), essendo $\varphi_{vn}'(t)$ tempo continuo, e in (135):

$$S_{\varphi_{VCO}}(f) = \left|\frac{2\pi \mathrm{K_V}}{s}\right|^2 |1 - G(\mathrm{j}2\pi f)|^2 S_{\varphi_{n'}}(f) \qquad (147)$$

L'impatto del rumore di quantizzazione sullo spettro di potenza de rumore di fase in uscita dal sintetizzatore dipende dall'architettura scelta per il modulatore Σ-Δ.

In generale se il modulatore Σ-Δ ha una generica funzione di trasferimento del rumore $NTF(z)$, il contributo del rumore di quantizzazione si ottiene dalle equazioni (138) e (140) oppure osservando per comodità la Figura 32. Considerando che $\varphi_n[k]$ è un segnale tempo discreto che viene filtrato con $G(s)$ tempo continua, è necessario utilizzare la (143). Siccome $r[k]$ è un segnale tempo discreto che viene filtrato da $NTF(z)$ tempo discreta bisogna considerare la (142). Si ottiene:

$$S_{\varphi_q}(f) = \frac{1}{T} |G_n(\mathrm{j}2\pi f)|^2 |G_{acc}(f)|^2 \left|NTF\left(e^{\mathrm{j}2\pi f T}\right)\right|^2 S_r(f)$$

$$S_{\varphi_q}(f) = \frac{1}{T}|T \cdot G(\mathrm{j}2\pi f)|^2 \left|2\pi \frac{e^{-\mathrm{j}2\pi fT}}{1 - e^{-\mathrm{j}2\pi fT}}\right|^2 \left|NTF\left(e^{\mathrm{j}2\pi fT}\right)\right|^2 S_r(f) \quad (148)$$

dove $S_r(f)$ è pari ad 1/12 se si considera il rumore di quantizzazione bianco e indipendente dall'ingresso del modulatore Σ-Δ.

Nel caso si utilizzi un modulatore Σ-Δ con architettura MASH, avente una $NTF(z)$ di ordine $m_{\Sigma\Delta}$ indicata in (116), il contributo dello spettro di potenza del rumore di fase dato dal rumore di quantizzazione espresso in (148) diventa:

$$S_{\varphi_q}(f) = \frac{1}{T}|T \cdot G(\mathrm{j}2\pi f)|^2 \left|2\pi \frac{e^{-\mathrm{j}2\pi fT}}{1 - e^{-\mathrm{j}2\pi fT}}\right|^2 \left|\left(1 - e^{-\mathrm{j}2\pi fT}\right)^{m_{\Sigma\Delta}}\right|^2 S_r(f)$$

$$S_{\varphi_q}(f) = \frac{1}{T}|T \cdot G(\mathrm{j}2\pi f)|^2 \left|2\pi e^{-\mathrm{j}2\pi fT}\left(1 - e^{-\mathrm{j}2\pi fT}\right)^{m_{\Sigma\Delta}-1}\right|^2 S_r(f)$$

$$S_{\varphi_q}(f) = \frac{1}{T}|T \cdot G(\mathrm{j}2\pi f)|^2 \left|2\pi e^{-\mathrm{j}2\pi fT}\left(e^{-\mathrm{j}\pi fT}\right)^{m_{\Sigma\Delta}-1}\left(e^{\mathrm{j}\pi fT} - e^{-\mathrm{j}\pi fT}\right)^{m_{\Sigma\Delta}-1}\right|^2 S_r(f)$$

$$S_{\varphi_q}(f) = \frac{1}{T}|T \cdot G(\mathrm{j}2\pi f)|^2 \left((2\pi)^2 (2\sin(\pi fT))^{2(m_{\Sigma\Delta}-1)}\right) S_r(f) \quad (149)$$

Si nota quindi che l'ordine della funzione di trasferimento del rumore di quantizzazione è ridotto di uno per via dell'azione integrante del divisore. Per frequenze molto inferiori a $1/T$, il rumore modellato dalla $NTF(z)$ ha un roll-off pari a $-20(m_{\Sigma\Delta} - 1)$ dB/decade. Pertanto, se l'ordine di $G(f)$ è pari all'ordine del modulatore Σ-Δ, il contributo del rumore di quantizzazione sull'errore di fase presenta un roll-off pari a -20dB/decade fuori dalla banda del PLL, ovvero la stessa del VCO noise.

Capitolo 4

CALCOLO DEI PARAMETRI DEL MODELLO DEL PLL E ANALISI DEL RUMORE DI FASE MEDIANTE L'ASSISTENTE AL DESIGN DI PLL

Le procedure di calcolo dei parametri del modello del PLL e dell'analisi del rumore di fase descritti nei capitoli precedenti possono essere sviluppate in un software di calcolo numerico.

Il gruppo di lavoro di Perrott ha messo gratuitamente a disposizione un software [17] utilizzabile solo attraverso un'interfaccia grafica, il cui codice è chiuso. Mediante l'interfaccia grafica di Perrott è possibile progettare PLL solo di ordine al massimo pari a 3, con caratteristiche della $G(s)$ implementate correttamente solo di tipo Butterworth, Bessel o Chebyshev I. Il modo in cui viene gestito il prototipo di tipo Chebyshev II ha delle inesattezze che non sono trascurabili, infatti Perrott, non considerando l'effetto degli zeri del prototipo di Chebyshev II sul valore dei parametri del modello del PLL e di conseguenza anche sul rumore di fase, di fatto non progetta il PLL con caratteristiche dinamiche dettate dal prototipo di Chebyshev II. Inoltre, per stessa ammissione di Perrott in [3], il prototipo ellittico non è stato implementato per problemi di calcolo dei poli e degli zeri del prototipo stesso nell'ambiente di programmazione utilizzato dallo stesso, ovvero Matlab.

Essendo il codice chiuso e avendo a disposizione solo un'interfaccia grafica, non è stato possibile visionare il codice stesso per creare un software di assistenza semplicemente codificandolo nell'ambiente di programmazione scelto. Inoltre, non sono stati semplicemente convertiti gli algoritmi descritti da Perrott in [1] e [2], in quanto semplici formule valide per ordini non superiori a 3.

Prima di estendere le potenzialità offerte dall'interfaccia grafica resa disponibile da Perrott, è stato quindi necessario arrivare a replicare i risultati ottenibili dal codice chiuso di Perrott nei casi in cui questo funziona correttamente.

L’ambiente di programmazione scelto per implementare il tool di assistenza al design di PLL è Python.

Una delle ragioni fondamentali per la quale si è scelto Python riguarda la preesistenza di un toolbox per il progetto di modulatori Σ-Δ raccolti nel pacchetto software open source PyDSM [12] [13], basato sul Python scientifico, svolto all'interno di un gruppo di ricerca dell'università di Bologna, impegnato in una linea di lavoro relativa all'ottimizzazione delle caratteristiche di noise shaping dei modulatori Σ-Δ.

Python, a differenza di Matlab, è un sistema aperto e general purpose. Queste caratteristiche rendono più facile produrre codice multipiattaforma, fruibile in ambienti in cui i vincoli di costo sono significativi (p.e., quello didattico). Inoltre, la scelta di codificare in forma aperta un tool di assistenza al progetto di PLL, andando a riciclare anche alcune funzionalità già esistenti in programmi gratuiti si giustifica in quanto un codice aperto si presta ad essere impiegato non solo come strumento a sé stante, ma anche come piattaforma su cui sperimentare nuove idee e possibilità e come libreria in grado mettere funzionalità a disposizione di progetti software più articolati.

Per questo motivo è possibile usare il codice prodotto come libreria all'interno di altri codici EDA più articolati ed è un contributo notevole rispetto al software reso disponibile da Perrott.

Matlab, a differenza di Python che è completamente gratuito, è abbastanza costoso e questo lo rende appetibile solo a chi ha sufficienti fondi per comprare una licenza. Inoltre, gli algoritmi di Matlab sono proprietari, per cui non è possibile vedere il codice della maggior parte degli algoritmi che vengono utilizzati ed è quindi necessario fidarsi che questi siano stati implementati correttamente, oltre al fatto che è difficile e in certe circostanze impossibile estendere le funzionalità di Matlab [18].

Python metta a disposizione un ambiente scientifico molto flessibile, con funzionalità che sono analoghe a quelle di un ambiente di calcolo numerico come Matlab, attraverso un pacchetto chiamato Numpy [6] [7] [8]. Inoltre, mediante la libreria Matplotlib [19] [20], è possibile creare grafici aventi caratteristiche analoghe a quelle di Matlab. Grazie al pacchetto nominato Scipy [9] [10] è possibile sfruttare algoritmi scientifici già codificati in maniera estremamente ottimizzata.

Il linguaggio di programmazione general purpose di Python presenta una moltitudine di tipi di dato che lo rendono ottimale per l’organizzazione di dati. Ad esempio, contiene liste, set,

dizionari. I dizionari sono simili ad array, ma l'accesso al singolo elemento non è necessariamente un indice intero, bensì una chiave che può essere un qualsiasi tipo di dato. Questo la rende ottimale per la creazione di tabelle di lookup. Anche Matlab supporta i dizionari, ma le chiavi devono essere dello stesso tipo di dato [21].

Una delle caratteristiche del linguaggio Python è quella di essere estremamente conciso e di consentire una prototipazione molto rapida. I codici scritti in Python tendono ad essere più compatti e leggibili dei corrispondenti codici Matlab. Ciò è ascrivibile sia alla maggiore modernità del linguaggio, sia alla sua superiore espressività.

Il codice Python prodotto ha quindi il compito di calcolare i parametri del modello della funzione di trasferimento ad anello aperto $A(s)$ a partire dalle specifiche sulle caratteristiche dinamiche richieste per la funzione di trasferimento ad anello chiuso e di valutare in maniera grafica la posizione dei poli e degli zeri della funzione di trasferimento ad anello chiuso $G(s)$ ottenuta, dalla quale estrapolare le informazioni sulla stabilità del sistema. Al tempo stesso deve essere possibile valutare graficamente la risposta del sintetizzatore ad una sollecitazione a gradino, che esprime come il PLL si comporta alla richiesta di passare a un'altra frequenza di uscita, e il modulo della funzione di trasferimento.

Un ulteriore compito del codice è quello di valutare l'effetto delle differenti sorgenti di rumore sul rumore di fase in uscita. Le sorgenti di rumore sono espresse mediante specifiche scelte appositamente coincidenti con quelle utilizzate da Perrott in modo che i software siano confrontabili e al tempo stesso sia facilitato il designer di PLL che già utilizza il software di Perrott.

Capitolo 4.1.: Calcolo dei parametri del modello del PLL

La procedura seguita nel codice Python sviluppato per il calcolo dei parametri del modello di $A(s)$ è la seguente:

1) Date le specifiche riguardanti l'ordine del PLL m, la forma desiderata del modello di $G(s)$, il ripple massimo in banda passante rp e l'attenuazione minima in banda attenuata rs, si determinano i poli p_{ma} e gli zeri z_{ma} del prototipo analogico avente pulsazione angolare di taglio unitaria.

$$rp = -20\log\left(\frac{1}{\sqrt{1+\varepsilon^2}}\right) \quad (150)$$

$$rs = -20\log\left(\frac{1}{A_s}\right) \quad (151)$$

2) Si determina la pulsazione critica W_m della maschera del filtro da utilizzare nelle routine di design dei filtri analogici implementate da Python, in modo che la frequenza di taglio del filtro prodotto coincida con quella definita dalle specifiche f_0.

$$W_m = \frac{2\pi f_0}{\sqrt[m]{\prod_{i=0}^{m-1} -p_{\mathrm{ma}_i}}} \quad (152)$$

dove p_{ma_i} è l'i-esimo polo del prototipo analogico a pulsazione di taglio unitaria.

3) Si determina la funzione di trasferimento del prototipo di $G(s)$ ovvero della $G(s)$ desiderata mediante le routine di design dei filtri analogici di Python.

4) Si determinano i polinomi $N_G(s)$ e $D_G(s)$ normalizzati rispetto al coefficiente di s^0.

   NOTA: nel caso di filtri Chebyshev II ed ellittici, essendo $G(0) \neq 1$, prima della normalizzazione è necessario moltiplicare il numeratore per un fattore dipendente dalla specifica del ripple minimo in banda passante, in modo che $G(0)$ sia unitario e la procedura successiva continui a determinare in modo corretto i parametri del Loop Filter e di $A(s)$.

5) Se il PLL è di tipo I si utilizza il sistema (60) per determinare $N_A(s)$ e $D_A(s)$ ovvero la forma "ba" della funzione di trasferimento $A(s)$ e si calcola ω_K con la (61). La forma "ba" di una funzione di trasferimento è espressa come una sequenza di due array contenenti rispettivamente i coefficienti dei polinomi $N_A(s)$ e $D_A(s)$.

6) Se il PLL è di tipo II, a partire dalla specifica sulla posizione dello zero aggiuntivo rispetto alla frequenza di taglio asintotica f_0 espressa dal rapporto f_z/f_0, si ha $\omega_z = 2\pi f_z$, si calcola ω_{cp} tramite la (82), si utilizza il sistema (83) per determinare $N_A(s)$ e $D_A(s)$ e si calcola ω_K^2 con la (84).

7) Si determina la variabile di controllo K pari a:

$$K = \omega_K^g \tag{153}$$

con g pari al tipo di PLL.

8) Si determinano le restanti variabili di controllo comuni ai modelli di $A(s)$ e di $H(s)$, a partire dai poli e dagli zeri ottenuti trasformando la funzione di trasferimento A(s) dalla forma "ba" alla forma "zpk" sfruttando le funzioni di elaborazione dei segnali di Python. La forma "zpk" è una sequenza di 3 elementi, il primo è l'array degli zeri, il secondo è l'array dei poli, il terzo è il guadagno.

Per determinare le variabili di controllo relative al numeratore $N_A(s)$, essendo costituito da sole coppie di zeri complessi coniugati, si individuano le singole coppie di zeri complessi coniugati (z_{cci} , z_{cci}*) e si utilizza la seguente formula:

$$\omega_{z0i} = -\sqrt{z_{cc_i} {z_{cc_i}}^*} \tag{154}$$

Per determinare le variabili di controllo relative al numeratore $D_A(s)$, si individuano i poli reali p_{ri} e le coppie di poli complessi coniugati (p_{cci} , p_{cci}*) e si utilizzano le seguenti formule:

$$\omega_{Pi} = -p_{r_i} \tag{155}$$

$$\omega_{Pi} = -\sqrt{p_{cc_i} {p_{cc_i}}^*} \tag{156}$$

$$Q_{Pi} = -\frac{\sqrt{p_{cc_i} {p_{cc_i}}^*}}{p_{cc_i} + {p_{cc_i}}^*} \tag{157}$$

9) Si verifica se con le variabili di controllo determinate si ottiene la posizione dei poli e degli zeri di $G(s)$ richiesta dalle specifiche di progetto, eventualmente determinando il diagramma dei poli e degli zeri, il grafico della funzione di trasferimento e della risposta al gradino unitario.

Capitolo 4.2.: Analisi del rumore di fase in uscita dal PLL

Utilizzando Python è possibile procedere all'analisi del rumore di fase in uscita dal sintetizzatore di frequenza. Per fare questo è necessario definire delle specifiche di progetto utili per determinare il rumore di fase in uscita $L(f)$ e il corrispondente rms jitter.

Le specifiche di progetto riguardanti l'analisi del rumore sono:

- La frequenza del segnale di riferimento f_{ref}.
- La frequenza del segnale di uscita che si vuole sintetizzare f_{out}.
- La specifica sul detector noise espressa in dBc/Hz

$$L_{dn}(f) = \left(\frac{2\pi}{\alpha} N_{nom}\right)^2 \left(\frac{1}{2} S_{E_n}(f)\right) \quad (158)$$

  dove la dipendenza dalla frequenza non sussiste essendo $S_{En}(f)$ uno spettro di potenza bianco (nell'analisi si trascura l'effetto delle componenti spurie introdotte dal PFD), per cui la specifica verrà in seguito indicata con L_{dn}.

- la specifica sul VCO noise espressa in dBc/Hz

$$L_{VCO}(f) = \left(\frac{\mathrm{K_V}}{f_{off\_VCO}}\right)^2 \left(\frac{1}{2} S_{\varphi_n'}(f)\right) \quad (159)$$

  dove la dipendenza dalla frequenza non sussiste essendo $S_{\varphi n}'(f)$ uno spettro di potenza bianco (rumore riportato all'ingresso del VCO), per cui la specifica verrà in seguito indicata con L_{VCO}.

  È necessario fornire anche il valore della frequenza di offset f_{off_VCO} alla quale la specifica suddetta è stata determinata.

- L'ordine $m_{\Sigma\Delta}$ del modulatore Σ-Δ, nel caso questo abbia un'architettura MASH
- La funzione di trasferimento del rumore $NTF(z)$ del generico modulatore Σ-Δ. Questa è definita tramite due array che contengono i coefficienti dei polinomi numeratore e denominatore della $NTF(z)$

Il programma consente di analizzare separatamente le sorgenti di rumore e di valutare anche l'effetto complessivo, determinando il grafico del rumore di fase in uscita.

Supponendo nulli il rumore di quantizzazione e il VCO noise, il rumore di fase in uscita dal sintetizzatore dovuto al detector noise si ottiene considerando la (146) e la (158):

$$L_{out\_dn}(f) = L_{dn}|G(\mathrm{j}2\pi f)|^2 \qquad (160)$$

Nota: L_{dn} e $L_{out\_dn}(f)$ indicati in (160) sono lineari e non in decibel di potenza.

Supponendo nulli il rumore di quantizzazione e il detector noise, il rumore di fase in uscita dal sintetizzatore dovuto al VCO noise si ottiene considerando la (147) e la (159):

$$L_{out\_VCO}(f) = L_{VCO}\left(\frac{f_{off\_VCO}}{f}\right)^2 |1 - G(\mathrm{j}2\pi f)|^2 \qquad (161)$$

Nota: L_{VCO} e $L_{out\_VCO}(f)$ indicati in (161) sono lineari e non in decibel di potenza.

Supponendo nullo il detector noise e il VCO noise, se l'architettura del modulatore Σ-Δ è di tipo MASH con ordine $m_{\Sigma\Delta}$, il rumore di fase in uscita dal sintetizzatore dovuto al rumore di quantizzazione bianco indipendente dal segnale di ingresso del modulatore, si ottiene considerando la (149):

$$L_{out\_q}(f) = \frac{1}{12}\cdot\frac{1}{f_{ref}}|G(\mathrm{j}2\pi f)|^2\left((2\pi)^2\left(2\sin\left(\pi\frac{f}{f_{ref}}\right)\right)^{2(m_{\Sigma\Delta}-1)}\right) \qquad (162)$$

Nel caso di modulatore Σ-Δ generico caratterizzato dalla funzione di trasferimento del rumore $NTF(z)$, considerando la (148) si ottiene:

$$L_{out\_q}(f) = \frac{1}{12}\cdot\frac{1}{f_{ref}}|G(\mathrm{j}2\pi f)|^2\left|2\pi\frac{e^{-\mathrm{j}2\pi\frac{f}{f_{ref}}}}{1-e^{-\mathrm{j}2\pi\frac{f}{f_{ref}}}}\right|^2\left|NTF\left(e^{\mathrm{j}2\pi\frac{f}{f_{ref}}}\right)\right|^2 \qquad (163)$$

Il rumore di fase totale in uscita è la somma dei contributi:

$$L_{out}(f) = L_{out\_dn}(f) + L_{out\_VCO}(f) + L_{out\_q}(f) \qquad (164)$$

Il valore di rms jitter si ottiene direttamente dalla (104), dove l'intervallo di valutazione $[f_{off\_min}, f_{off\_max}]$ è settabile direttamente dall'utente tramite l'interfaccia grafica, altrimenti assumono per default valori rispettivamente pari a un $0.1\,f_0$ e $100\,f_0$.

Il programma implementato in Python consente anche di considerare l'effetto del *flicker noise* o rumore rosa introdotto dalla pompa di carica ed eventualmente anche dal buffer del segnale

di riferimento. Il flicker noise presenta una densità spettrale di potenza con andamento iperbolico 1/*f*.

Per considerare l'effetto del rumore rosa è necessario inserire un'ulteriore specifica che identifica la *corner frequency* f_{corn_dn} ovvero la frequenza alla quale la densità spettrale di potenza del flicker noise coincide numericamente alla densità spettrale del rumore bianco introdotto dal PFD (detector noise).

Il flicker noise ha una pendenza pari a -10dB/decade e sommandolo al detector noise definito precedentemente determina un contributo sull'errore di fase in uscita dal sintetizzatore pari a:

$$L_{out\_dn}(f) = L_{dn}\left(1 + \frac{f_{corn\_dn}}{f}\right)|G(\mathrm{j}2\pi f)|^2 \qquad (165)$$

dove anche in questo caso L_{dn} e $L_{out\_dn}(f)$ sono indicati in lineare e non in decibel di potenza.

In alcuni casi il flicker noise non ha una pendenza pari a -10dB/decade, per questo motivo il programma in Python consente di cambiare questa specifica di progetto indicando esplicitamente la pendenza assunta dal flicker noise $slope_{fn\_dn}$ e la (165) diventa:

$$L_{out\_dn}(f) = L_{dn}\left(1 + \left(\frac{f_{corn\_dn}}{f}\right)^{\frac{slope_{fn\_dn}}{-10}}\right)|G(\mathrm{j}2\pi f)|^2 \qquad (166)$$

Allo stesso modo è possibile aggiungere rumore $1/f^3$ al VCO noise specificando la corner frequency f_{corn_VCO}.

Il rumore $1/f^3$ aggiunto al VCO ha una pendenza pari a -30dB/decade e sommandolo al VCO noise definito precedentemente determina un contributo sull'errore di fase in uscita dal sintetizzatore pari a:

$$L_{out\_VCO}(f) = L_{VCO}\left(1 + \frac{f_{corn\_VCO}}{f}\right)\left(\frac{f_{off\_VCO}}{f}\right)^2|1 - G(\mathrm{j}2\pi f)|^2 \qquad (167)$$

Nel caso questo presenti una pendenza differente da -30dB/decade, si aggiunge una specifica di progetto in cui viene indicata la pendenza assunta $slope_{fn\_VCO}$ e la (167) diventa:

$$L_{out\_VCO}(f) = L_{VCO}\left(1 + \left(\frac{f_{corn\_VCO}}{f}\right)^{\frac{slope_{fn\_VCO}}{-10}-2}\right)\left(\frac{f_{off\_VCO}}{f}\right)^2|1 - G(\mathrm{j}2\pi f)|^2 \qquad (168)$$

Capitolo 5

TECNICA DI COMPENSAZIONE DEGLI EFFETTI REATTIVI PARASSITI DELL'ANELLO APERTO

Il metodo di calcolo dei parametri della funzione di trasferimento ad anello aperto $A(s)$ del sintetizzatore di frequenze, tali da ottenere una funzione di trasferimento ad anello chiuso $G(s)$ desiderata, illustrato nel Capitolo 2, assume che la $A(s)$ possa essere realizzata priva di poli e di zeri parassiti. Tuttavia, questa ipotesi non è verificata, in quanto i componenti del PLL, come il VCO e lo stesso Loop Filter, introducono per loro natura degli effetti reattivi parassiti, ovvero dei poli e degli zeri parassiti. Il valore degli zeri e dei poli parassiti non è controllabile dal designer dal PLL, perché nascono intrinsecamente e indipendentemente dall'architettura scelta. Per questo motivo, l'unica accortezza che il designer può adottare è quella di considerarli noti tramite simulazioni o misure sui componenti del PLL, di valutarne gli effetti e modificare opportunamente le variabili di controllo a disposizione, in modo che il sistema abbia il comportamento desiderato anche in presenza di parassiti.

Nel caso di PLL di tipo I, si definiscono i poli dominanti della funzione di trasferimento $G(s)$ di ordine m, gli m poli aventi pulsazione naturale minore. Vale a dire che i poli dominanti, nel caso non ci fossero parassiti, coincidono con i poli della $G(s)$ desiderata definita dai diversi prototipi.

Nel caso di PLL di tipo II, i poli dominanti di $G(s)$ di ordine m sono gli m poli aventi pulsazione naturale minore escludendo il polo a pulsazione naturale minima, coincidente con l'extra-polo aggiuntivo $p = -\omega_{cp}$. In tal modo, i poli dominanti coincidano con i poli della $G(s)$ desiderata nel caso in cui non ci fossero i parassiti.

I poli e gli zeri parassiti di $A(s)$, per essere tali, devono avere una pulsazione naturale sufficientemente alta rispetto ai poli dominanti, più precisamente devono ricadere al di fuori della banda di $G(s)$.

Questa assunzione consente di effettuare un'importante semplificazione, in quanto per frequenze sufficientemente maggiori della banda f_0, come conseguenza della (31), la $A(s)$ e la G(s) sono praticamente uguali. Di conseguenza, la posizione dei poli e degli zeri parassiti

nel piano S si può considerare immutata, ovvero $A(s)$ e $G(s)$ hanno i medesimi poli e zeri parassiti. Questa semplificazione perde di validità nel momento in cui i parassiti non si comportano da tali, ovvero quando hanno pulsazioni prossime alla banda di $G(s)$.

I poli e gli zeri parassiti hanno l'effetto di spostare nello spazio S la posizione dei poli dominanti, questo determina un peggioramento delle performance dinamiche del sistema, che non avrà più la $G(s)$ nella forma desiderata.

Il modo in cui la posizione dei poli dominanti nello spazio S cambia non è lineare e si può valutare dalla (29). In Figura 33 è rappresentato il differente effetto di un polo parassita sulla posizione del polo reale e della coppia di poli complessa coniugata per $G(s)$ del terzo ordine:

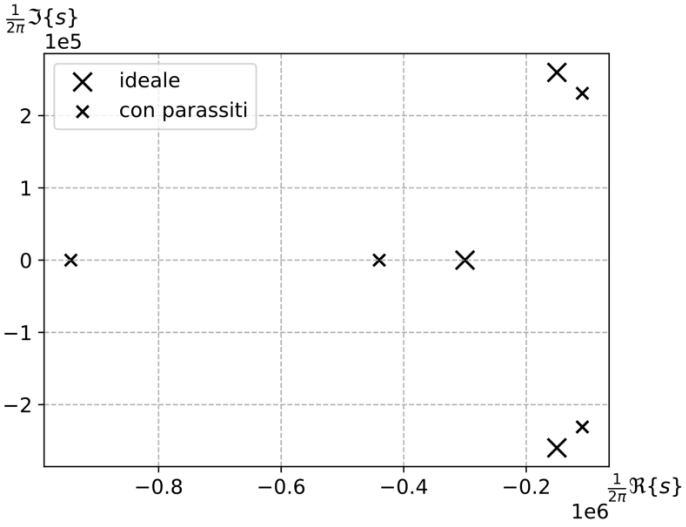

Figura 33: Effetto di un polo parassita sulla posizione dei poli dominanti della G(s)

L'idea di base è quella di creare un algoritmo in grado di effettuare una compensazione degli effetti dei poli e degli zeri parassiti, in modo che la posizione dei poli dominanti della $G(s)$ sul piano S, coincida con la posizione dei poli della $G(s)$ desiderata.

Fino ad ora non si è parlato degli zeri della $G(s)$ desiderata in quanto questi non vengono perturbati dalla presenza dei parassiti, infatti dalla (37) si nota che $G(s)$ e $A(s)$ hanno gli stessi zeri. La presenza di poli e zeri aggiuntivi nella funzione di trasferimento ad anello aperto $A(s)$ non varia la pulsazione naturale degli zeri, quindi anche in questo caso i parametri degli zeri complessi coniugati di $A(s)$ coincidono con gli zeri complessi coniugati della $G(s)$ desiderata nelle forme Chebyshev2 e Ellittici.

Il fatto che i poli dominanti si spostino per effetto dei parassiti potrebbe far pensare che la procedura di calcolo dei parametri di $A(s)$ sia del tutto inutile. Tuttavia, nell'ipotesi in cui la

pulsazione dei parassiti sia sufficientemente maggiore della banda di $G(s)$, i poli dominanti non si spostano di molto rispetto alla posizione ideale, per questo motivo, è possibile sfruttare il metodo di calcolo dei parametri per determinare una configurazione iniziale dei parametri di $A(s)$ utile all'algoritmo di compensazione dei parassiti.

L'algoritmo di compensazione dei parassiti è quindi un algoritmo di ottimizzazione non lineare che cerca la configurazione dei parametri che determina una posizione dei poli dominanti coincidente a quella desiderata, anche in presenza di parassiti. Questo algoritmo di ottimizzazione utilizza come configurazione di parametri di partenza, quella ottenuta dal metodo di calcolo dei parametri nel caso senza parassiti.

Perrott in [2] propone un algoritmo di ottimizzazione non lineare valido per $G(s)$ al massimo del terzo ordine.

L'algoritmo proposto da Perrott, nel caso di $G(s)$ del terzo ordine senza zeri, è descrivibile con i seguenti passi:

1. Scompone i poli della $G(s)$ desiderata in un vettore contente il polo reale, la parte reale della coppia di poli complessa coniugata e la parte immaginaria positiva della coppia di poli complessa coniugata.

$$\overline{\boldsymbol{P_{des}}} = \begin{pmatrix} \Re\{p_{cc\_des}\} \\ \Im\{p_{cc\_des}\} \\ p_{r\_des} \end{pmatrix} \quad (169)$$

2. Determina la configurazione dei parametri della $A(s)$, in questo caso K, ω_P e Q_P, mediante le formule espresse in Tabella 2 per PLL di tipo I o in Tabella 4 per PLL di tipo II, che verrà utilizzata come punto di partenza dell'algoritmo.

$$\overline{\boldsymbol{K_\iota}} = \begin{pmatrix} K_i & \omega_{Pi} & Q_{P_i} \end{pmatrix} \quad (170)$$

Ciclicamente:

3. Perturba i parametri di $A(s)$, al primo ciclo di una quantità piccola $\overline{\boldsymbol{\Delta K_1}}$, nei cicli successivi in base al vettore $\overline{\boldsymbol{\Delta K}}$ determinato nel punto 7.

$$\overline{\boldsymbol{\Delta K_1}} = \begin{pmatrix} \Delta K_1 & \Delta\omega_{P1} & \Delta Q_{P_1} \end{pmatrix} \quad (171)$$

$$\overline{\boldsymbol{K_{\iota+1}}} = \overline{\boldsymbol{K_\iota}} + \overline{\boldsymbol{\Delta K_\iota}} \quad (172)$$

4. Utilizzando i parametri perturbati, determina la $A(s)$ senza parassiti, aggiunge i parassiti alla $A(s)$ e calcola la $G(s)$ tramite la (29).

5. Seleziona i poli dominanti della $G(s)$ e li scompone in un vettore come al passo 1.

$$\overline{\boldsymbol{P_a}} = \begin{pmatrix} \Re\{p_{cc\_a}\} \\ \Im\{p_{cc\_a}\} \\ p_{r\_a} \end{pmatrix} \quad (173)$$

6. Crea una matrice che rappresenta la linearizzazione valida localmente delle funzioni che modellano la differenza fra la posizione dei poli di $G(s)$ desiderata e la posizione dei poli dominanti attuale, scomposti entrambi come al punto 1, in funzione dei parametri di $A(s)$.

$$\overline{\boldsymbol{\Delta P}} = \overline{\boldsymbol{P_{des}}} - \overline{\boldsymbol{P_a}} = \begin{pmatrix} \Re\{\Delta p_{cc}\} \\ \Im\{\Delta p_{cc}\} \\ \Delta p_r \end{pmatrix} \quad (174)$$

$$\overline{\overline{\boldsymbol{C}}} = \overline{\boldsymbol{\Delta P}}\ \overline{\boldsymbol{\Delta K_\iota}} = \begin{pmatrix} \frac{\Re\{\Delta p_{cc}\}}{\Delta K_i} & \frac{\Re\{\Delta p_{cc}\}}{\Delta K_i} & \frac{\Re\{\Delta p_{cc}\}}{\Delta K_i} \\ \frac{\Im\{\Delta p_{cc}\}}{\Delta \omega_{P_i}} & \frac{\Im\{\Delta p_{cc}\}}{\Delta \omega_{P_i}} & \frac{\Im\{\Delta p_{cc}\}}{\Delta \omega_{P_i}} \\ \frac{\Delta p_r}{\Delta Q_{P_i}} & \frac{\Delta p_r}{\Delta Q_{P_i}} & \frac{\Delta p_r}{\Delta Q_{P_i}} \end{pmatrix} \quad (175)$$

7. Calcola la perturbazione dei parametri da applicare al passo successivo tramite la matrice e il vettore differenza di posizione dei poli desiderati e di posizione dei poli dominanti attuale

$$\overline{\boldsymbol{\Delta K_{\iota+1}}} = \overline{\overline{\boldsymbol{C}}}^{-1}\overline{\boldsymbol{\Delta P}} \quad (176)$$

8. Se la differenza posizione poli desiderati e posizione poli dominanti attuale è inferiore ad una soglia, i parametri calcolati compensano l'effetto dei parassiti

Questo algoritmo può avere problemi di convergenza nel caso i parassiti abbiano una pulsazione inferiore a quattro volte o nei casi migliori a due volte la banda di $G(s)$, a seconda del tipo di PLL e della forma.

Un altro limite di questo approccio è la difficoltà di estendere questa procedura per $G(s)$ di ordine maggiore di tre. Inoltre, creare un algoritmo di ottimizzazione ad hoc per un caso specifico non consente una formulazione generale del problema.

Per questi motivi è conveniente sfruttare algoritmi di ottimizzazione già presenti in letteratura e incorporati in linguaggi di programmazione come Python.

Capitolo 5.1: Implementazione dell'algoritmo di compensazione dei parassiti con il metodo di Nelder-Mead

Un possibile algoritmo di ottimizzazione adatto alla compensazione dei parassiti è quello che sfrutta il metodo di Nelder-Mead. Il metodo di Nelder-Mead, invece di utilizzare le derivate di una funzione, si basa sul concetto del simplesso, ovvero un politopo di m dimensioni con il minor numero di vertici. Ad esempio, il politopo di dimensione 0 è il punto, di dimensione 1 è il segmento, di dimensione 2 il triangolo, di dimensione 3 il tetraedro e così via. Quindi il simplesso m-dimensionale ha m+1 vertici. Il metodo consente di determinare il punto di minimo di una funzione obiettivo. Affinché il metodo funzioni, è necessario che la funzione obiettivo non presenti delle discontinuità nell'intorno del punto di valutazione della funzione, altrimenti non converge. Il metodo determina in modo iterativo le posizioni di test degli argomenti della funzione obiettivo, estrapolando passo per passo il comportamento della funzione obiettivo all'interno del dominio definito dai vertici del simplesso. Ad ogni passo cambia un singolo vertice, ovvero quello in cui la funzione obiettivo è maggiore, sostituendolo con il centroide dei restanti m vertici. Se nel nuovo punto la funzione obiettivo è minore, la direzione di ricerca del minimo è quella definita dal nuovo punto rispetto a quello precedente, altrimenti si cerca nella direzione del vertice nel quale la funzione obiettivo è minore.

Questo metodo suggerisce quindi di definire una funzione obiettivo la cui minimizzazione determina la compensazione degli effetti dei poli e degli zeri parassiti.

Una possibile funzione obiettivo è quella definita dalla distanza euclidea fra i poli della $G(s)$ desiderata e i poli dominanti della $G(s)$ soggetta ai parassiti introdotti nella funzione di trasferimento ad anello aperto $A(s)$.

L'algoritmo di minimo che adotta il metodo di Nelder-Mead implementato nella letteratura di Python però non è in grado di lavorare con vettori complessi, ma solo con vettori scalari. Risulta quindi necessario definire una scomposizione dei poli dominanti della $G(s)$ e dei poli della $G(s)$ desiderata in un vettore contenente nei primi elementi i poli reali, poi a due a due la parte reale e la parte immaginaria delle coppie di poli complessi coniugati, considerando solo

il polo a parte immaginaria positiva. In tal modo, la lunghezza del vettore coincide con il numero dei poli senza perdita di informazione, infatti il polo a parte immaginaria negativa della singola coppia di poli complessi coniugati è direttamente legato all'altro da un'operazione di semplice coniugazione.

Per cui la funzione obiettivo sarà definita dalla norma L2 della differenza dei due vettori rappresentativi dei poli della $G(s)$ desiderata e dei poli dominanti $G(s)$ soggetti ai parassiti e calcolati iterativamente dall'algoritmo di minimo.

Utilizzando questo algoritmo è possibile quindi ottenere i medesimi risultati dell'algoritmo di Perrott senza creare un algoritmo di ottimizzazione ad hoc ed estendibile a $G(s)$ di qualsiasi ordine, specificando come parametro la tolleranza richiesta sul valore della norma L2 sopra citata.

La funzione da minimizzare implementata in Python è caratterizzata dai seguenti argomenti:

- Un vettore contenente la configurazione iniziale delle variabili di controllo del sistema, ovvero K, ω_{Pi} e Q_{Pi} definiti dalla Tabella 9 per PLL di tipo I e dalla Tabella 10 per PLL di tipo II, determinati nel caso ideale senza parassiti

- Un vettore contente ω_{z0i} definiti nelle medesime tabelle citate al punto precedente, che in questo caso sono costatanti non subendo l'effetto dei parassiti

- La pulsazione ω_z nel caso di PLL di tipo II e l'indicazione sul tipo di PLL

- L'indicazione sulla forma desiderata della $G(s)$ e sul ripple massimo in banda passante, utili per determinare la funzione $G(s)$ con l'accortezza della normalizzazione della $G(0)$ in caso di filtri di ordine pari aventi specifiche sul ripple in banda passante

- Due liste contenenti le informazioni sui poli e sugli zeri parassiti, in particolare è possibile esprimere queste informazioni in termini di frequenza dei poli e degli zeri parassiti reali e tuple di due elementi corrispondenti alla frequenza dei poli e degli zeri complessi coniugati parassiti e dal relativo fattore di merito

La funzione obiettivo implementata può essere riassunta nei seguenti passi:

1. Si determina la funzione di trasferimento ad anello aperto $A(s)$ a partire dalla configurazione iniziale delle variabili di controllo del modello utilizzando di fatto il metodo espresso dalla Tabella 9 per PLL di tipo I e dalla Tabella 10 per PLL di tipo II, questa configurazione varia ad ogni ciclo di valutazione della funzione obiettivo.

2. Si aggiungono i poli e gli zeri parassiti, in particolare conviene avere la $A(s)$ nella forma "ba" in modo da aggiungere i singoli poi e i singoli zeri mediante operazioni di somma e di prodotto vettoriale dei coefficienti dei polinomi.

3. Si determina la $G(s)$ a partire dalla $A(s)$ a cui sono stati aggiunti i parassiti

4. Si selezionano solo i poli dominanti della $G(s)$ a seconda del tipo di PLL e si costruisce il vettore che indica la posizione dei poli dominanti ottenuto con la configurazione delle variabili di controllo al generico passo di valutazione della funzione obiettivo

5. Si determina la norma L2 della differenza fra i due vettori rappresentativi della posizione dei poli della $G(s)$ desiderata e della posizione dei poli dominanti corrente.

L'algoritmo di compensazione termina nel momento in cui la valutazione della funzione obiettivo, per la configurazione delle variabili di controllo determinate, risulta inferiore ad una certa soglia, definita da un parametro dell'algoritmo di minimo che implementa il metodo di Nelder-Mead.

Il programma in Python per l'assistenza al progetto di PLL consente anche di effettuare una valutazione grafica dell'effetto di compensazione dei poli e degli zeri parassiti, confrontando nel diagramma dei poli e degli zeri i poli dominanti ideali, quelli senza algoritmo di compensazione e quelli ottenuti con l'algoritmo di compensazione. Questo confronto è possibile anche nel grafico della risposta al gradino unitario e del modulo della funzione di trasferimento.

Capitolo 6

EFFETTO DELLA VARIAZIONE DEI PARAMETRI DEL MODELLO

Una volta determinati i parametri della funzione di trasferimento ad anello aperto $A(s)$, eventualmente considerando anche la presenza di poli e zeri parassiti, è necessario determinare la topologia di circuito del Loop Filter che implementa una funzione di trasferimento $H(s)$ caratterizzata dal modello ottenuto. Tuttavia, nell'implementazione hardware del PLL, ci sono fenomeni intrinsechi che determinano una variazione dei parametri del modello della $A(s)$. Questi fenomeni riguardano l'impossibilità di ottenere mediante una struttura hardware la configurazione dei parametri ottenuta con infinita precisione, la dipendenza dalla temperata e anche il fatto che eventuali mismatch strutturali dei dispositivi che costituiscono il Loop Filter determinano un comportamento differente considerando diversi chip.

La variazione dei parametri del modello di $A(s)$ determina di conseguenza una variazione di $G(s)$ secondo la (29), per cui la $G(s)$ non avrà più le caratteristiche dinamiche desiderate.

L'approccio standard prevede di valutare il margine di fase della funzione di trasferimento ad anello aperto per ogni variazione dei parametri, per verificare se il sistema $G(s)$ rimane stabile. Questo approccio però non fornisce alcuna informazione sulla variazione delle caratteristiche dinamiche della $G(s)$, ma solo sulla stabilità del sistema.

Adottando invece il modello della $G(s)$ e della $A(s)$, è possibile verificare direttamente mediante la simulazione in Python il diagramma dei poli e degli zeri, la risposta del sistema al gradino unitario e il modulo della funzione di trasferimento, senza andare a calcolare necessariamente il margine di fase.

Il diagramma dei poli e degli zeri fornisce direttamente informazioni sulla stabilità del sistema, infatti se per ogni variazione dei parametri di $A(s)$ i poli restano sul semipiano sinistro del piano S, allora il sistema è stabile.

Anche dalla risposta al gradino unitario si vede se il sistema è stabile ovvero se il sistema converge ad 1. Inoltre, è possibile verificare come varia il settling time e le eventuali sovraelongazioni della risposta.

Il programma in Python per l'assistenza al design di PLL, consente di valutare l'effetto di ogni singolo parametro di $A(s)$, definendo un intervallo di valutazione che rappresenta la variazione percentuale del parametro rispetto al valore determinato dai metodi di calcolo dei parametri. È necessario anche specificare l'incremento percentuale all'interno dell'intervallo.

Questa parte di programma può essere anche sfruttata per valutare come cambiano le caratteristiche dinamiche della $G(s)$ al variare dei parametri del modello.

Considerando a titolo di esempio la $G(s)$ di un PLL di tipo II, con f_z/f_0 pari ad 1/8, f_0=300KHz, ripple massimo in banda passante $r_p = \delta_1$ = 2dB, attenuazione minima in banda attenuata $r_s = \delta_2$ =40dB, verranno riportati qui sotto gli effetti della variazione dei parametri della $A(s)$ con una variazione percentuale del ±15%. Le figure sono riportate nel dominio delle frequenze [Hz] e non quello delle pulsazioni naturali, in quanto più comode per un designer di PLL.

Analizzando il diagramma dei poli e degli zeri, sarà evidente che la posizione della coppia di zeri complessi coniugati sarà indipendente dalla variazione dei parametri K, ω_P e Q_P essendo univocamente definita dalla $G(s)$ desiderata. Inoltre, anche la posizione dello zero sarà indipendente da K, ω_P, Q_P e ω_{z0} essendo determinata dal parametro fz/f_0 costante e pari a 1/8.

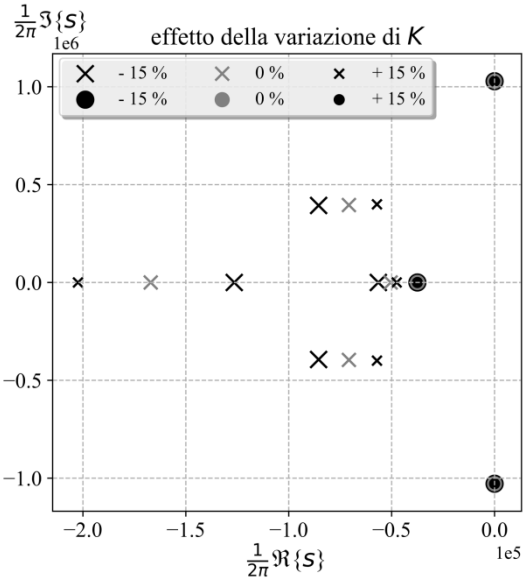

Figura 34: effetto della variazione di K sul diagramma dei poli e degli zeri di G(f)

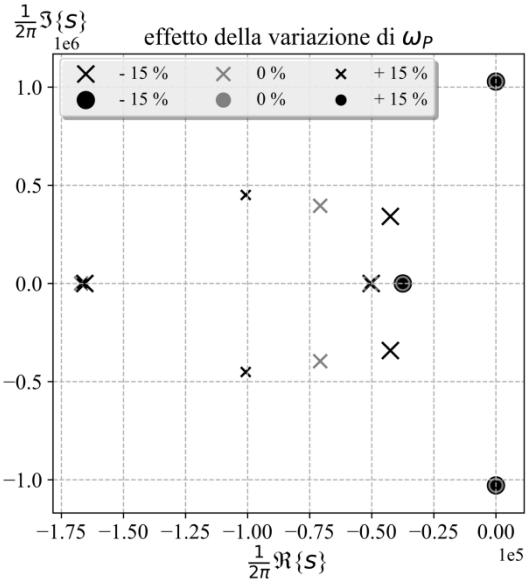

Figura 35: effetto della variazione di ω_P sul diagramma dei poli e degli zeri di G(f)

Osservando la Figura 34 si nota che la posizione del polo reale $p_r = -\omega_{cp}$ varia poco con il guadagno K, mentre si nota una dipendenza non lineare della parte reale dei poli della $G(s)$ desiderata. Invece la parte immaginaria dei poli complessi coniugati della $G(s)$ desiderata non è alterata dalla variazione di K.

Dalla Figura 35 si osserva che i poli reali della $G(s)$ sono praticamente indipendenti dalla ω_P, mentre si ha un legame non lineare della posizione della coppia di poli complessi coniugati, la cui pulsazione naturale aumenta all'aumentare di ω_P.

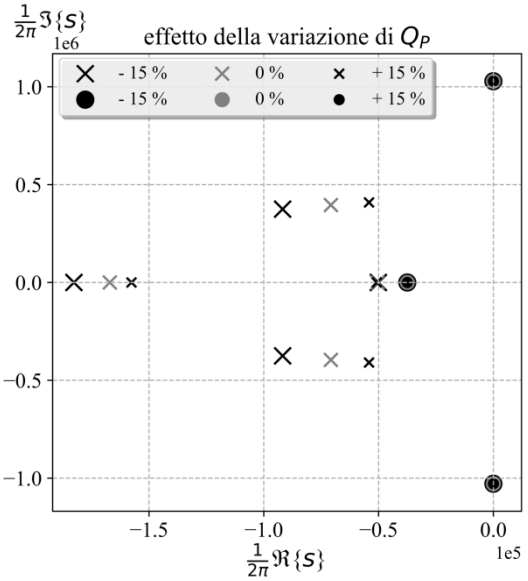

Figura 36: effetto della variazione di Q_P sul diagramma dei poli e degli zeri di G(f)

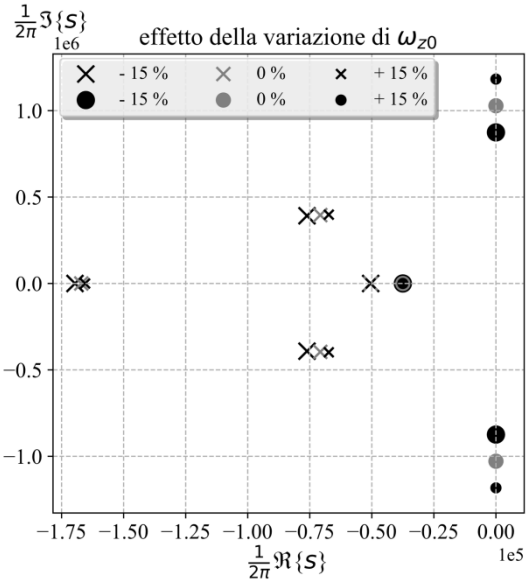

Figura 37: effetto della variazione di ω_{z0} sul diagramma dei poli e degli zeri di G(f)

Come evidenziato in Figura 36, la posizione del polo reale $p_r = -\omega_{cp}$ non dipende Q_P. L'effetto della variazione del fattore di merito Q_P sulla parte immaginaria della coppia di poli complessi coniugati è trascurabile, mentre si ha una dipendenza non lineare della parte reale dei poli della $G(s)$ desiderata, in particolare la pulsazione naturale del polo diminuisce all'aumentare di Q_P.

Dalla Figura 37 si nota come la variazione della pulsazione naturale della coppia di zeri complessi coniugati determina una piccola variazione della posizione dei poli della $G(s)$ desiderata, mentre non altera il polo reale $p_r = -\omega_{cp}$. Questo implica che l'ipotesi di Perrott di non considerare le coppie di zeri complessi coniugati, rappresenta comunque una buona approssimazione del sistema $G(s)$, il quale però non potrà avere le caratteristiche dinamiche richieste senza modificare il modello della $G(s)$, ovvero considerando nel modello anche gli zeri della $G(s)$ desiderata nei casi Chebyshev II ed Ellittici.

Analizzando invece la risposta al gradino unitario del sistema, nel caso specifico si ottengono i risultati riportati nei grafici sottostanti:

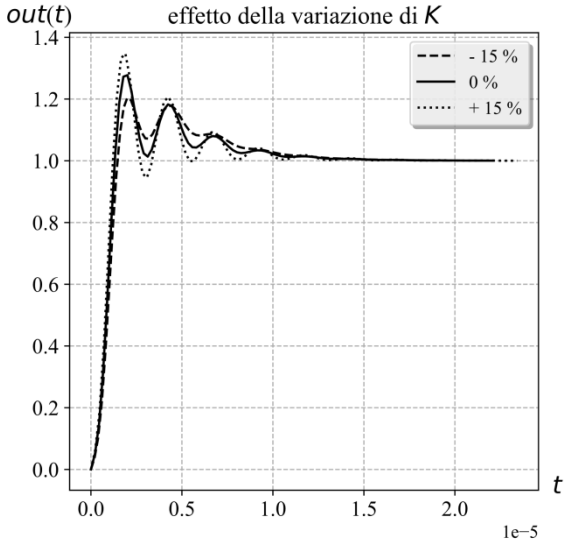

Figura 38: effetto della variazione di K sulla risposta al gradino unitario di G(f)

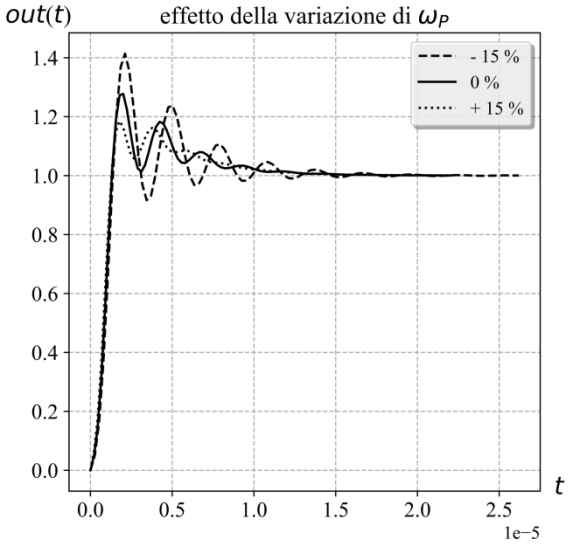

Figura 39: effetto della variazione di ω_P sulla risposta al gradino unitario di G(f)

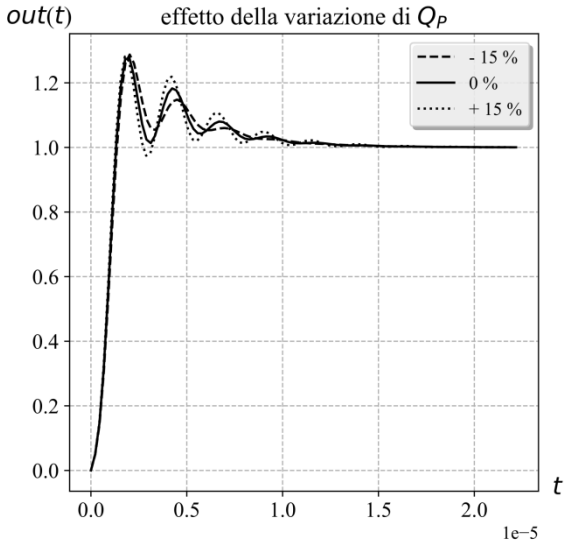

Figura 40: effetto della variazione di Q_P sulla risposta al gradino unitario di G(f)

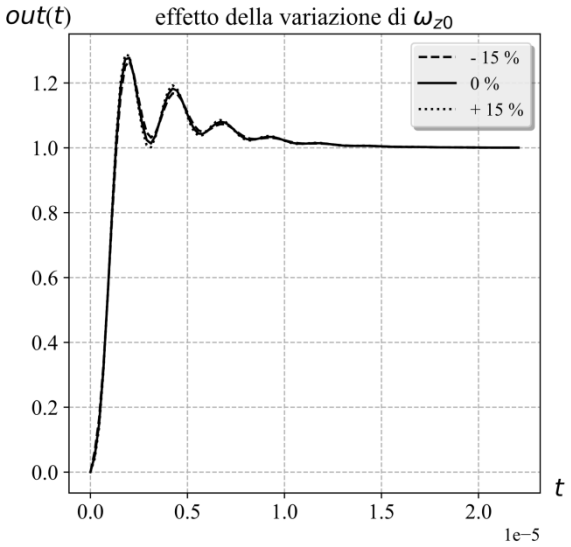

Figura 41: effetto della variazione di ω_{z0} sulla risposta al gradino unitario di G(f)

Come si osserva dalla Figura 38, al variare di *K,* cambia l'ampiezza delle sovraelongazioni della risposta al gradino, ma non cambiano gli istanti temporali ai quali si manifestano i picchi. In particolare, all'aumenta di *K*, aumenta l'ampiezza della sovraelongazioni.

Dalla Figura 39 si nota che la variazione del parametro ω_P determina una differente ampiezza delle sovraelongazioni della risposta al gradino, ma anche differenti istanti temporali in cui si hanno i picchi. All'aumenta di ω_P si ha una riduzione dell'ampiezza della sovraelongazione e i picchi avvengono in istanti di tempo inferiori, cambiando di conseguenza il settling time.

Valutando la Figura 40, si osserva che la velocità con la quale la sovraelongazione della risposta al gradino unitario si annulla dipende dal fattore di merito Q_P in modo non è lineare, in particolare al diminuire di Q_P aumenta la velocità di annullamento della sovraelongazione, ovvero diminuisce il settling time.

Come si osserva dalla Figura 41, la variazione della pulsazione naturale della coppia di zeri complessi coniugati della $G(s)$ desiderata, determina una variazione della risposta al gradino pressoché trascurabile, motivo per cui il modello proposto da Perrott risulta una buona approssimazione per l'analisi della stabilità della $G(s)$, ma non per la determinazione corretta della $G(s)$ desiderata che include zeri per filtri Ellittici e di Chebyshev II.

Valutando invece gli effetti della variazione dei parametri della $A(s)$ sul modulo della funzione di trasferimento, nel caso specifico, si ottengono gli andamenti riportati in Figura 42, Figura 43, Figura 44 e Figura 45.

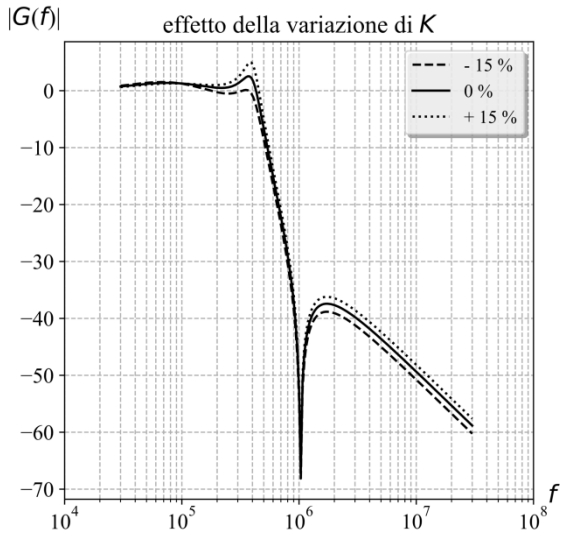

Figura 42: effetto della variazione di K sul modulo della funzione di trasferimento G(f)

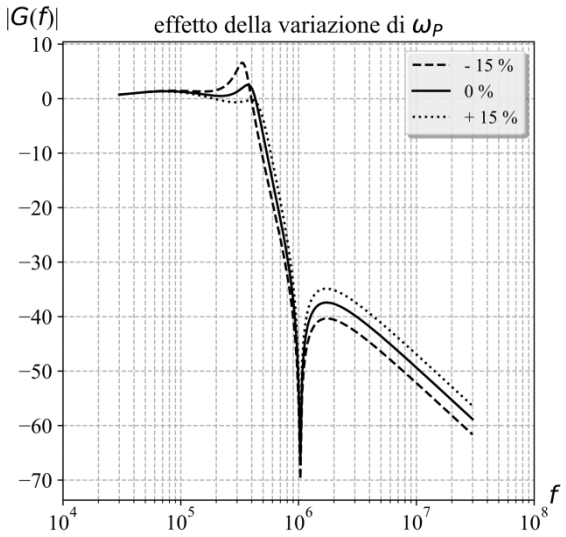

Figura 43: effetto della variazione di ω_P sul modulo della funzione di trasferimento G(f)

Come si osserva dalla Figura 42, la variazione del parametro K determina picchi differenti del modulo della funzione di trasferimento sia in banda passante che in banda attenuta, in particolare all'aumentare di K aumenta l'ampiezza del picco, il quale avviene sempre alla stessa frequenza.

Osservando la Figura 43 invece si nota che al variare del parametro ω_P cambia sia l'ampiezza dei picchi in banda passante e dei picchi in banda attenuata, sia la frequenza alla quale avvengono. All'aumentare di ω_P diminuisce l'ampiezza del picco in banda passante e aumenta la frequenza alla quale avviene, mentre il picco in banda attenuta aumenta all'aumentare di ω_P.

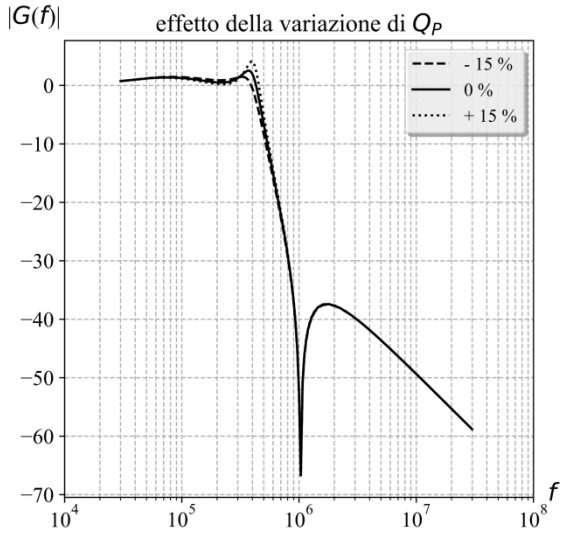

Figura 44: effetto della variazione di Q_P sul modulo della funzione di trasferimento G(s)

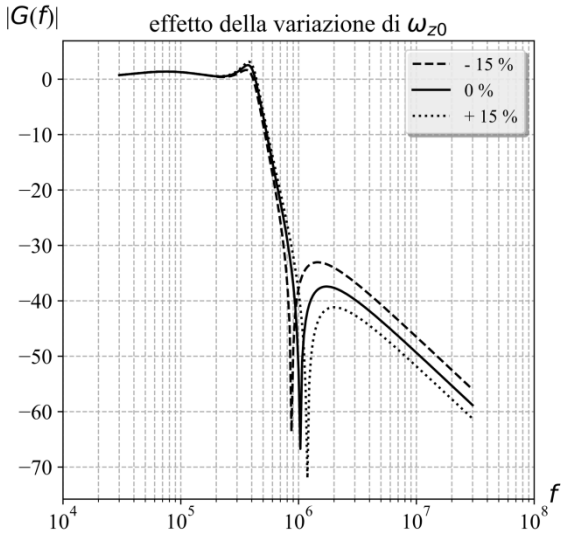

Figura 45: effetto della variazione di ω_{z0} sul modulo della funzione di trasferimento G(s)

Dalla Figura 44 si deduce che la variazione di Q_P non ha effetti sul modulo della funzione di trasferimento in banda attenuata, mentre in banda passante all'aumentare della Q_P aumenta l'ampiezza del picco e aumenta la frequenza alla quale questo avviene.

Come si nota dalla Figura 45, la variazione della pulsazione naturale della coppia di zeri complessi coniugati, non determina particolari effetti in banda passante sul modulo della funzione di trasferimento, mentre in banda attenuta l'ampiezza del picco diminuisce all'aumenta di ω_{z0}, mentre la frequenza alla quale avviene il picco diminuisce. Il modello di Perrott non tiene conto dell'effetto in banda attenuata, il quale non è trascurabile nella valutazione delle performance del sistema. In pratica il modello di Perrott non considerando gli zeri della $G(s)$ desidera, sovrastima l'attenuazione in banda attenuata.

Risulta interessante sfruttare l'algoritmo di valutazione dell'effetto della variazione dei parametri di $A(s)$ per valutare l'influenza della scelta del rapporto fz/f_0 sulla risposta dinamica del sistema, in particolare può essere valutato direttamente dal programma il modo in cui variano il diagramma dei poli e degli zeri, la risposta la gradino e il modulo della funzione di trasferimento rispetto alla f_z/f_0 nominale.

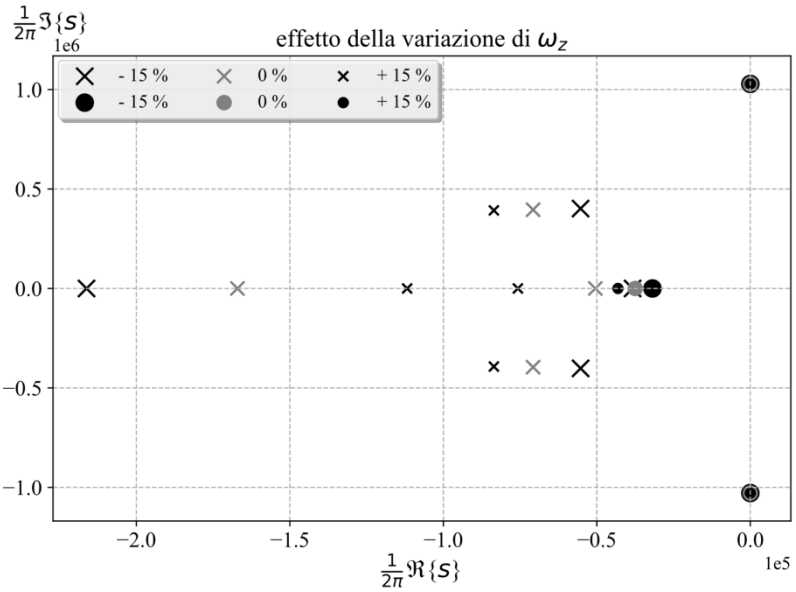

Figura 46: effetto della variazione di ω_z sul diagramma dei poli e degli zeri di G(f)

Osservando la Figura 46, la posizione della coppia degli zeri complessi coniugati rimane immutata al variare di ω_z, la posizione del polo reale $p_r = -\omega_{cp}$ ha una forte dipendenza dalla ω_z. come è normale che sia dovendo effettuare una compensazione. Inoltre, si nota che all'aumentare di ω_z diminuisce in modo non lineare la pulsazione del polo reale della $G(s)$ desiderata, mentre la parte reale della coppia di poli complessi coniugati aumenta. La parte immaginaria rimane pressoché immutata.

Dalla Figura 47 si nota che la velocità con cui la sovraelongazione si annulla è praticamente indipendente dalla ω_z mentre l'ampiezza dei picchi diminuisce all'aumentare della ω_z.

Per quanto riguarda il modulo della funzione di trasferimento, dalla Figura 48 si osserva che l'ampiezza dei picchi in banda passante in banda attenuta dipendono dalla ω_z e in particolare, all'aumentare di ω_z diminuisce l'ampiezza sia dei picchi in banda passante sia dei picchi in banda attenuata.

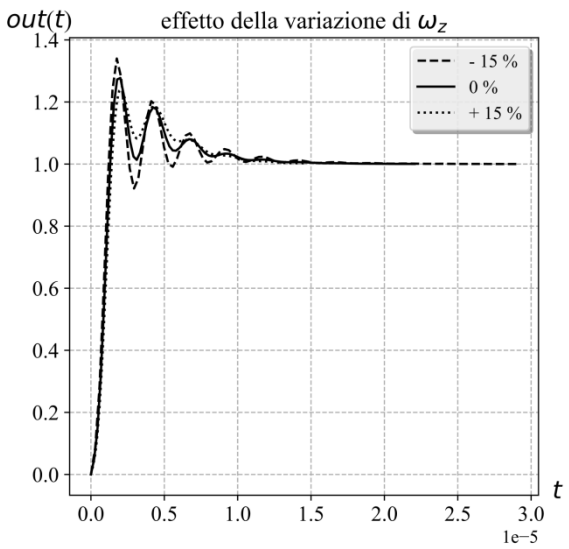

Figura 47: effetto della variazione di ω_z sulla risposta al gradino unitario di G(f)

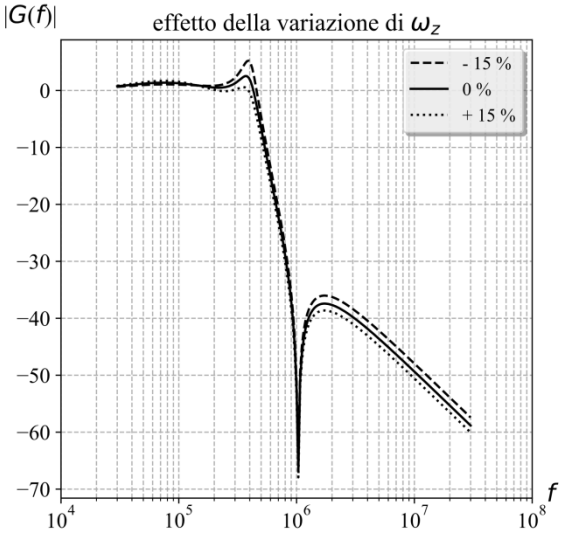

Figura 48: effetto della variazione di ω_z sul modulo della funzione di trasferimento G(f)

Capitolo 7

ESEMPIO DI UTILIZZO DELL'INTERFACCIA GRAFICA DELL'ASSISTENTE AL PROGETTO DI PLL

Per evitare che il designer di PLL modifichi di volta in volta svariati script di Python che:

- svolgono le funzioni di calcolo delle variabili di controllo del PLL tali da ottenere la $G(s)$ desiderata, anche in presenza di poli e zeri parassiti nella $A(s)$;
- verificano gli effetti delle variazioni dei parametri di $A(s)$;
- analizzano il rumore di fase in uscita dal PLL;
- o una combinazione delle funzioni precedenti,

si è prodotta un'interfaccia grafica scritta in PyQt5 [22].

PyQt5 è un modulo Python che consente di accedere alle funzionalità di Qt5, che a sua volta è un framework per lo sviluppo di software applicativo sviluppato in C++ e disponibile per numerose piattaforme (Windows, Linux, Mac, Ios, Android, ecc.). L'uso di PyQt5 consente di realizzare interfacce grafiche GUI (graphical user interface), organizzate su un layout suddiviso in box verticali, box orizzontali o griglie. All'interno dei box o delle griglie è possibile inserire dei componenti grafici chiamati *Widget* che possono essere ad esempio semplici scritte *QLabel*, caselle di testo *QLineEdit*, pulsanti *QPushButton* e aree di grafico *FigureCanvas*.

La scelta di impiegare Python e PyQt5 fa sì che il tool di assistenza al progetto di PLL sviluppato nella tesi sia utilizzabile virtualmente su qualunque sistema di uso comune, dai PC ai tablet.

L'interfaccia grafica consente di richiedere direttamente le specifiche sul progetto del PLL al designer e di visualizzare direttamente in un'unica schermata:

- i valori delle variabili di controllo, ovvero i parametri del modello di $A(s)$, tali da ottenere le caratteristiche dinamiche della funzione di trasferimento ad anello chiuso $G(s)$ richieste;

- un grafico selezionato fra:
    - il diagramma dei poli e degli zeri della $G(s)$;
    - la risposta della $G(s)$ alla sollecitazione a gradino;
    - il modulo della funzione di trasferimento $G(s)$;
    - il rumore di fase in uscita.
- il valore quadratico medio del jitter.

Si fornirà ora un esempio d'uso dell'interfaccia grafica.

Le specifiche sulla funzione di trasferimento ad anello chiuso $G(s)$ desiderata sono riassunte in Tabella 13.

m	indicatore del *roll-off* di $G(s)$
$\boldsymbol{f_0}$	banda asintotica di $G(s)$ [Hz]
shape	forma del prototipo che può essere Butterworth, Bessel, Chebyshev I, Chebyshev II o ellittico
rp	ripple massimo in banda passante per $G(s)$ di tipo Chebyshev I ed ellittico [dB]
rs	attenuazione minima in banda attenuata per $G(s)$ di tipo Chebyshev II ed ellittico [dB]
PLL type	numero di puri integratori (1 o 2) della funzione di trasferimento ad anello aperto *A(s)*
$\boldsymbol{f_z/f_0}$	posizione relativa dello zero aggiuntivo per PLL di tipo I rispetto alla banda asintotica di $G(s)$

Tabella 13: riassunto delle specifiche sulla dinamica della funzione di trasferimento ad anello chiuso G(s)

Le specifiche necessarie per valutare il rumore di fase sono riassunte in Tabella 14:

$\boldsymbol{f_{ref}}$	frequenza del segnale di riferimento [Hz]

f_{out}	frequenza nominale del segnale di uscita [Hz]
L_{dn}	detector noise [dBc/Hz]
f_{corn_dn}	corner frequency [Hz] per la specifica del flicker noise
$slope_{fn\_dn}$	pendenza del flicker noise [dB/decade]
L_{VCO}	VCO noise [dBc/Hz]
f_{corn_VCO}	corner frequency [Hz] per la specifica del flicker noise
$slope_{fn\_VCO}$	pendenza del flicker noise [dB/decade]
$m_{\Sigma\Delta}$	ordine del modulatore Σ-Δ nel caso questo abbia un'architettura MASH
$NTF(z)$	funzione di trasferimento del rumore del generico modulatore Σ-Δ
f_{off_min}	frequenza minima dell'intervallo di integrazione di $L(f)$ per il calcolo del valore quadratico medio del jitter
f_{off_max}	frequenza massima dell'intervallo di integrazione di $L(f)$ per il calcolo del valore quadratico medio del jitter

Tabella 14: riassunto delle specifiche necessarie per un'analisi completa del rumore di fase

Qualora si volessero considerare effetti reattivi parassiti introdotti nella funzione di trasferimento ad anello aperto $A(s)$, le specifiche sono riassunte in Tabella 15:

f_{Pi}	frequenze dei poli parassiti reali o complessi coniugati
Q_{Pi}	fattori di qualità delle coppie di poli parassiti complessi coniugati
f_{Zi}	frequenze degli zeri parassiti reali o complessi coniugati
Q_{Zi}	fattori di qualità delle coppie di zeri parassiti complessi coniugati

Tabella 15: riassunto delle specifiche sui poli e sugli zeri parassiti della funzione di trasferimento ad anello aperto A(s)

Capitolo 7.1: Caso di studio PLL di tipo I Chebyshev II

Per inserire le specifiche sull'ordine m della $G(s)$, sulla banda f_0, sulla forma della $G(s)$ desiderata, sulle specifiche sul ripple massimo in banda passante e sull'attenuazione minima in banda attenuta nel caso di filtri Chebyshev o Ellittici, sul tipo di PLL e nel caso fosse di tipo II il valore del rapporto f_z/f_0, l'interfaccia grafica è organizzata come in Figura 49, dove le caselle di testo si disattivano a seconda dei pulsanti premuti per evitare errori di input delle specifiche. Qualora il designer inserisse specifiche errate, come ad esempio ordini <1, banda negativa, specifiche sul ripple e sull'attenuazioni non in dB, valori non ammissibili per il rapporto f_z/f_0, l'interfaccia apre una finestra di errore in cui viene specificato il motivo dello stesso.

Supponendo si voglia progettare un PLL di tipo I caratterizzato da una $G(f)$ del 4° ordine, nella sezione "Dynamic Parameters" dell'interfaccia è necessario inserire le specifiche come rappresentato in Figura 49:

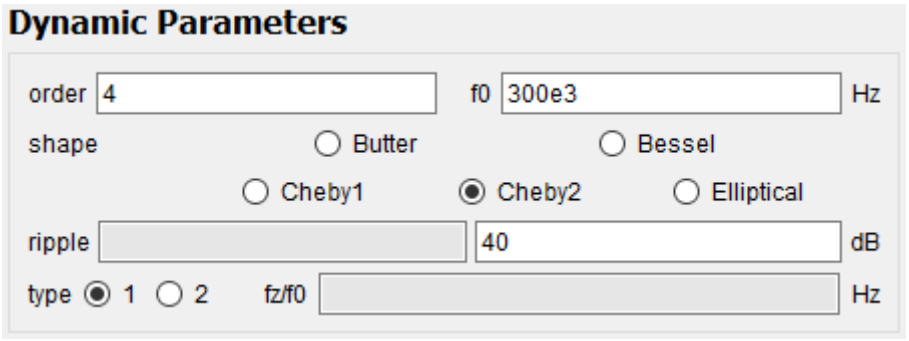

Figura 49: Input delle specifiche sulla dinamica della G(f)

Dopodiché è necessario scegliere cosa si vuole visualizzare fra il diagramma dei poli e degli zeri "Pole/Zero Diagram", la risposta al gradino unitario "StepResponse", il modulo della funzione di trasferimento "Transfer Function" e il rumore di fase in uscita dal PLL "Noise Plot". In particolare, nella sezione "Resulting Plot", oltre a scegliere quale grafico visualizzare, è possibile indicare a proprio piacimento gli estremi delle ascisse Xmin e Xmax e delle ordinate Ymin e Ymax del grafico. Inoltre, nel caso la simulazione non abbia la risoluzione desiderata o si voglia concentrare la simulazione per un intervallo di valutazione specifico, è possibile modificare gli estremi di valutazione mediante le caselle di testo "Start", "Stop" e indicare il numero di punti "N" di valutazione all'interno dell'intervallo. Attraverso le caselle denominate *QCheckBox* "lin" e "log" è possibile scegliere anche il tipo di incremento lineare o logaritmico dei punti di valutazione, mentre se entrambi sono deselezionati vale l'opzione di default del programma, che per il grafico del rumore di fase e

del modulo della funzione di trasferimento è Start = $f_0/10$, Stop = $100f_0$, N=200 con incremento logaritmico.

La parte dell'interfaccia grafica che si occupa delle opzioni sul grafico da visualizzare si presenta come in Figura 50:

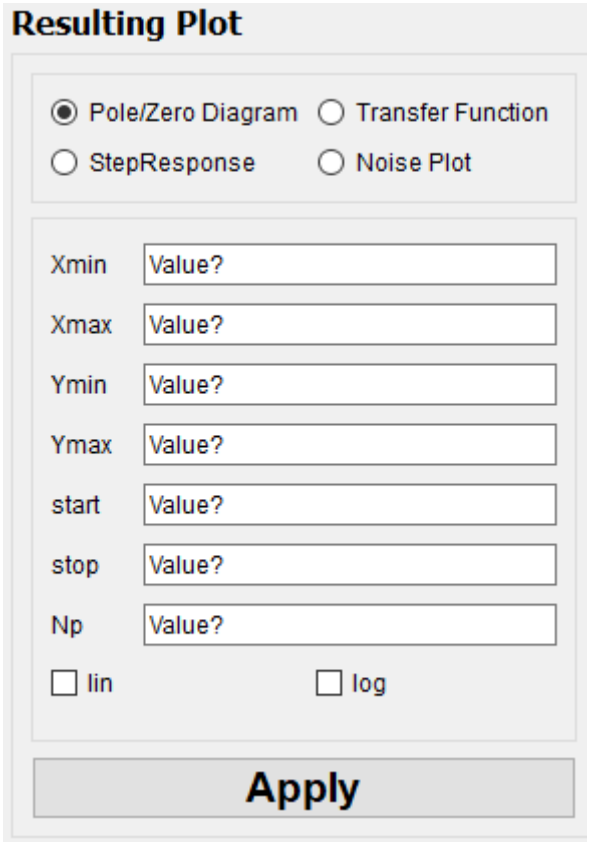

Figura 50: Scelta del grafico da visualizzare e delle opzioni di simulazione e degli estremi degli assi

Calcolo dei parametri della funzione di trasferimento ad anello aperto $A(f)$

A questo punto è possibile determinare il valore delle variabili di controllo nel caso in esame.

I valori dei parametri della $A(f)$ vengono riportati nella sezione "Resulting Open Loop Parameters" dell'interfaccia grafica. A differenza della trattazione fatta nei capitoli precedenti, per il designer conviene considerare come parametri le frequenze dei poli e degli zeri, invece delle pulsazioni naturali, perché più semplici da interpretare. Per cui i parametri di $A(f)$ sono K, f_P, Q_P, f_{z0} e f_z. dove evidentemente $f_P = \omega_P/2\pi, f_z = \omega_z/2\pi, f_{z0} = \omega_{z0}/2\pi$.

Nel caso del parametro f_P indicato nella sezione "Resulting Open Loop Parameters" questo in realtà è una lista di frequenze naturali dei corrispondenti poli che possono essere reali o

complessi coniugati. La distinzione fra poli reali e poli complessi coniugati si determina per confronto con Q_P, anch'esso rappresentato da una lista della stessa lunghezza di f_P. Gli elementi aventi lo stesso indice delle due sequenze fanno riferimento allo stesso polo, quindi se l'i-esimo elemento di Q_P è indicato con "None", allora f_{Pi} è la frequenza naturale di un polo reale, altrimenti è la frequenza naturale di una coppia di poli complessi coniugati con fattore di merito pari a Q_{Pi}. Questa notazione è stata introdotta per consentire il progetto di $G(f)$ anche di ordine maggiore a 3.

Nel caso del parametro f_{z0} essendo gli zeri della $G(f)$ desiderata nel caso Chebyshev II ed Ellittico a parte reale nulla, la i-esima coppia di zeri complessi coniugati è identificata dalla sola frequenza naturale f_{z0i}.

Una volta scelte le specifiche iniziali sulla dinamica, selezionando il pulsante "Apply" nella sezione "Resulting Plot", si ottengono i parametri della $A(f)$, che nel caso in esame sono quelli rappresentati in Figura 51:

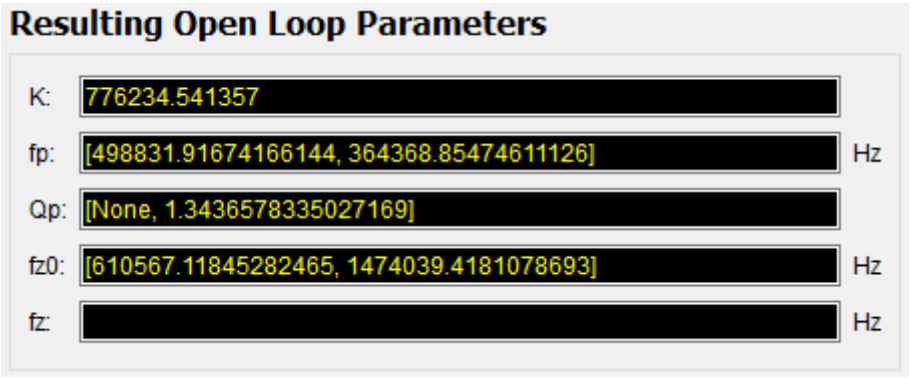

Figura 51: Valore delle variabili di controllo di A(s) che determinano la G(s) desiderata

Come si nota dalla Figura 51, il primo elemento della lista f_P è un la frequenza del polo reale di $A(f)$ essendo il corrispondente elemento della lista Q_P pari a None, mentre il secondo elemento è la frequenza della coppia di poli complessi coniugati.

Il diagramma dei poli e degli zeri di $G(f)$ ottenuto con questa configurazione dei parametri viene riportato in funzione della frequenza e non della pulsazione per comodità di lettura del designer. Da questo è immediato verificare la stabilità del sistema e in questo caso essendo tutti i poli a parte reale negativa il sistema è stabile.

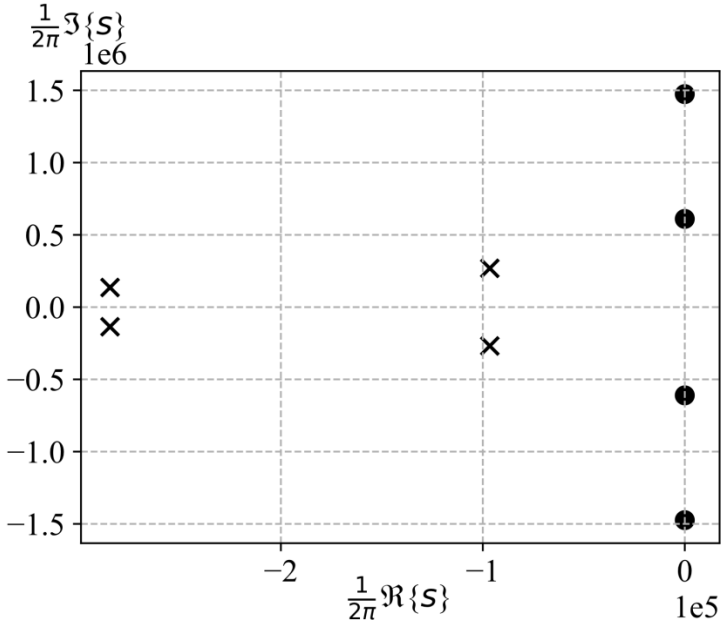

Figura 52: Diagramma dei poli e degli zeri di G(f) del 4°ordine di tipo I, Chebyshev II con f_0 300KHz e attenuazione minima 40dB

Dalla risposta al gradino unitario del sistema, si verifica direttamente se il sistema converge ovvero se è stabile e si può determinare graficamente il settling time.

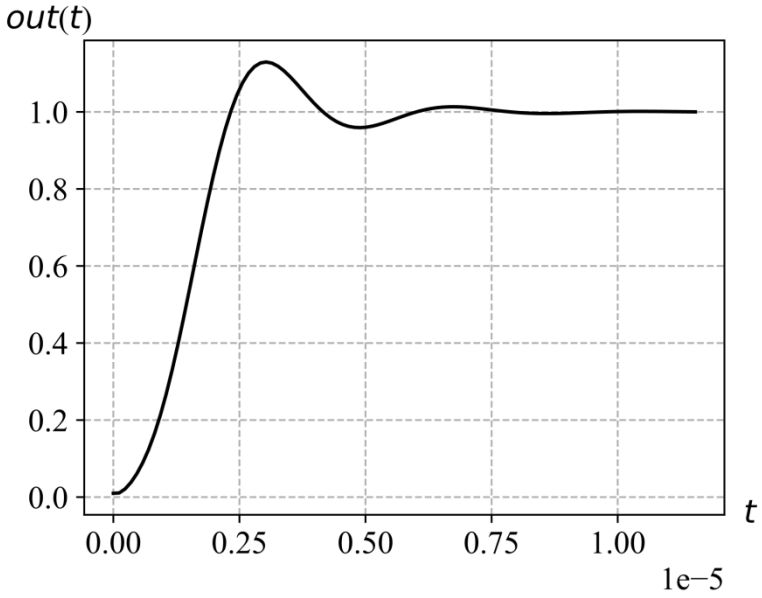

Figura 53: Risposta al gradino unitario di G(f) del 4°ordine di tipo I, Chebyshev II con f_0 300KHz e attenuazione minima 40dB

Dal grafico del modulo della funzione di trasferimento è possibile verificare se le specifiche di progetto sono rispettate. Nel caso di studio si nota l'assenza di ripple in banda passante essendo il filtro di Chebyshev II, l'assenza di picchi indesiderati in quanto PLL di tipo I e che la specifica sull'attenuazione minima di 40dB per ogni frequenza della banda attenuata è

rispettata. Si osserva anche che a regime l'attenuazione è costante essendo il numero di poli uguale al numero degli zeri.

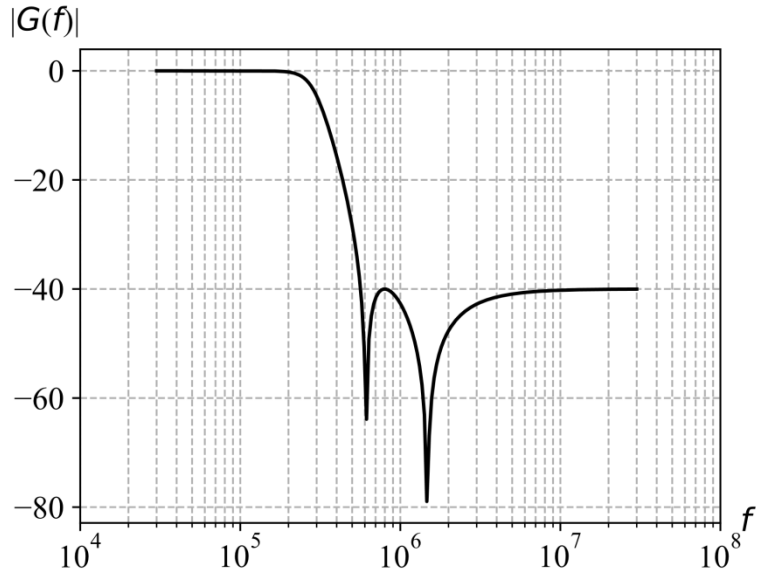

Figura 54: Modulo della funzione di trasferimento G(f) del 4°ordine di tipo I, Chebyshev II con f_0 300KHz e attenuazione minima 40dB

Valutazione dell'impatto della variazione dei parametri della funzione di trasferimento ad anello aperto *A*(*f*)

Per verificare l'impatto della variazione dei parametri di $A(f)$, è necessario attivare nell'interfaccia grafica il bottone ON a fianco della variabile della quale si vuole verificare l'effetto della sua variazione rispetto al valore ottenuto in precedenza.

Se si vuole verificare la variazione in un range di percentuali compreso fra [-15%, +15%] con uno *step* (passo) del 15% del guadagno K o della frequenza dello zero extra f_z, è necessario inserire nella casella di testo corrispondente una dicitura coincidente con quella di Matlab "start: step: stop", ovvero -0.15:0.15:0.15.

Discorso differente per i parametri f_P, Q_P e f_{z0}, questi infatti sono delle sequenze come indicato precedentemente, quindi in questo caso è necessario indicare di quale parametro si vogliono verificare gli effetti della variazione mediante un indice. Questo indice si ricava per confronto dalla indicazione dei parametri di $A(f)$ nella sezione "Resulting Open Loop Parameters" dell'interfaccia grafica. Se nella sequenza c'è un solo elemento, si può continuare ad utilizzare la notazione "start: step: stop" di Matlab, altrimenti la notazione scelta è: "([start,

step, stop], indice)" se si vuole variare un solo parametro, oppure "([start, step, stop], indice1), ([start, step, stop], indice2), …" se si vogliono variare più parametri. Gli indici partono da 0.

Nel caso in esame, se volessimo valutare l'effetto della variazione della frequenza del polo reale di *A*(*f*), in un range di variazioni percentuali rispetto al caso ideale compreso fra [-15%, +15%] con uno *step* (passo) del 15%, è necessario attivare il pulsante ON riferito a "fp alter" nella sezione "Alter Open Loop Parameters" dell'interfaccia grafica ed inserire la dicitura espressa in Figura 55, dove l'indice è 0 essendo reale solo il primo polo di *A*(*f*) della sequenza "fp".

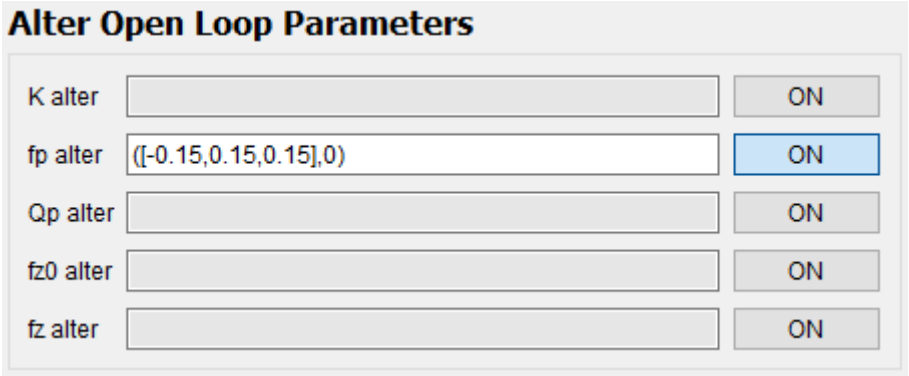

Figura 55: Scelta della variazione dei parametri di A(s)

Il risultato ottenuto è rappresentato dal diagramma dei poli e degli zeri, dalla risposta al gradino e dal modulo della funzione di trasferimento riportati sotto.

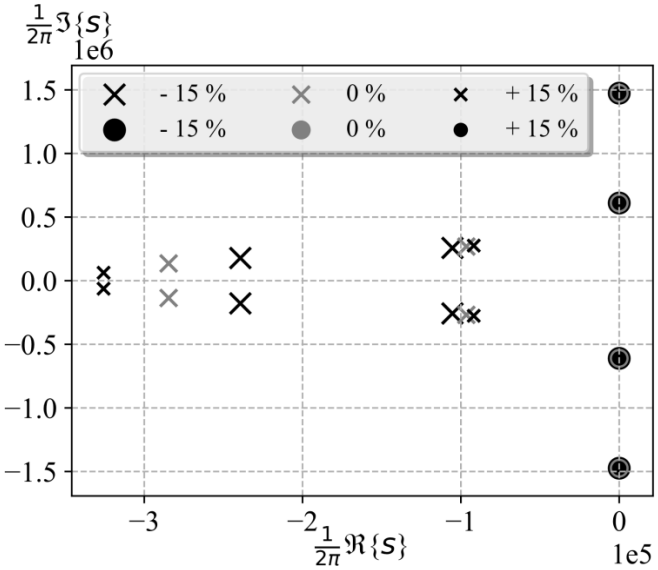

Figura 56: Effetto della variazione del polo reale di A(f) sulla posizione dei poli e degli zeri di G(f)

Si nota che la variazione della pulsazione del polo reale di $A(f)$ non ha effetti sulla posizione degli zeri, mentre l'effetto sulla posizione dei poli non è lineare.

Analizzando invece la risposta al gradino unitario rappresentata in Figura 57, si deduce che all'aumentare della frequenza del polo reale, il picco della sovraelongazione della risposta si riduce e al contempo si riduce anche il settling time.

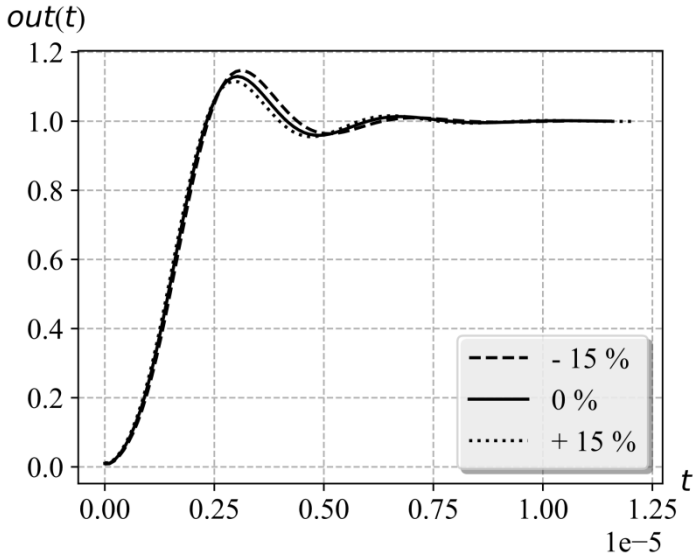

Figura 57: Effetto della variazione del polo reale di A(f) sulla risposta al gradino unitario di G(f)

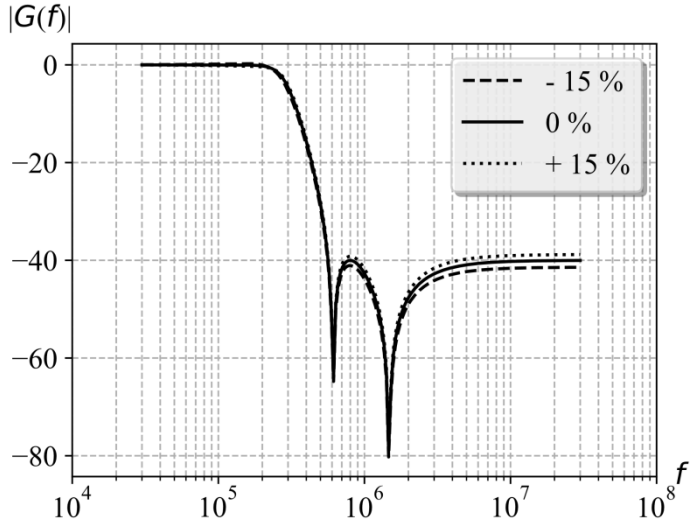

Figura 58: Effetto della variazione del polo reale di A(f) sul modulo della funzione di trasferimento G(f)

Infine, osservando l'effetto sul modulo della funzione di trasferimento in Figura 58, si verifica che in banda passante ci sono conseguenze pressoché trascurabili, mentre in banda attenuata è evidente che per frequenze del polo reale superiori a quella determinata in precedenza, la specifica di attenuazione minima di -40dB non è rispettata. Per cui il designer deve tenere in

considerazione che la valutazione esatta dei parametri consente di soddisfare le specifiche in modo stretto.

Supponendo invece di voler valutare gli effetti della variazione del guadagno *K* in un intervallo [-15%, +15%] con step del 15%, nel diagramma dei poli si nota una dipendenza non lineare della posizione dei poli rispetto al guadagno *K* e una indipendenza della posizione degli zeri, essendo questi i medesi di *A*(*f*) per costruzione del modello.

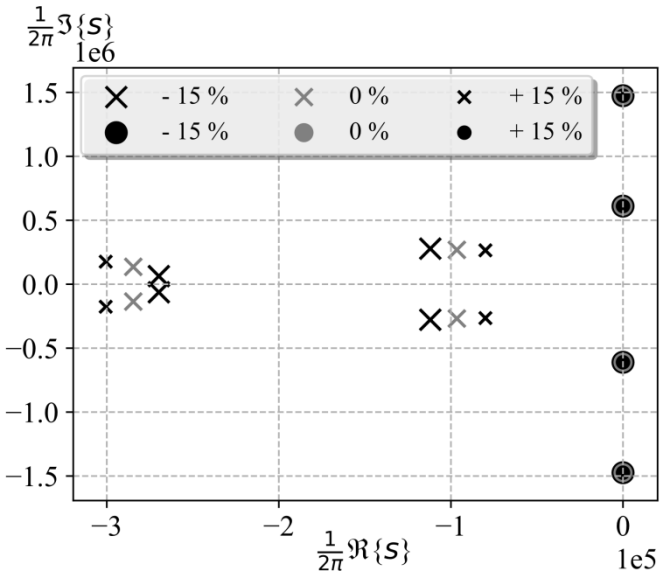

Figura 59: Effetto della variazione del guadagno K di A(f) sulla posizione dei poli e degli zeri di G(f)

Osservando invece la risposta al gradino unitario, si evince che all'aumentare del guadagno aumenta l'ampiezza del picco della sovraelongazione e il settling time.

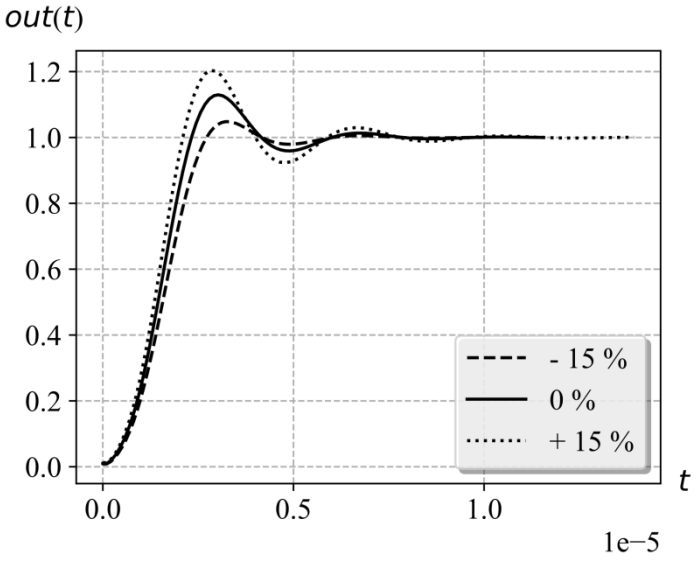

Figura 60: Effetto della variazione del guadagno K di A(f) sulla risposta al gradino unitario di G(f)

Per quanto riguarda l'effetto della variazione del guadagno *K* sul modulo della funzione di trasferimento, si osserva che questo determina un notevole cambiamento del comportamento in banda passante. Infatti, si ha un picco indesiderato per PLL di tipo I con *K* maggiori di quello nominale e un'attenuazione tutt'altro che trascurabile in banda per frequenze prossime alla frequenza di taglio. L'effetto in banda attenuata invece è il medesimo del caso precedente, ovvero per guadagni superiori a quello nominale la specifica di -40dB di attenuazione minima non è rispettata.

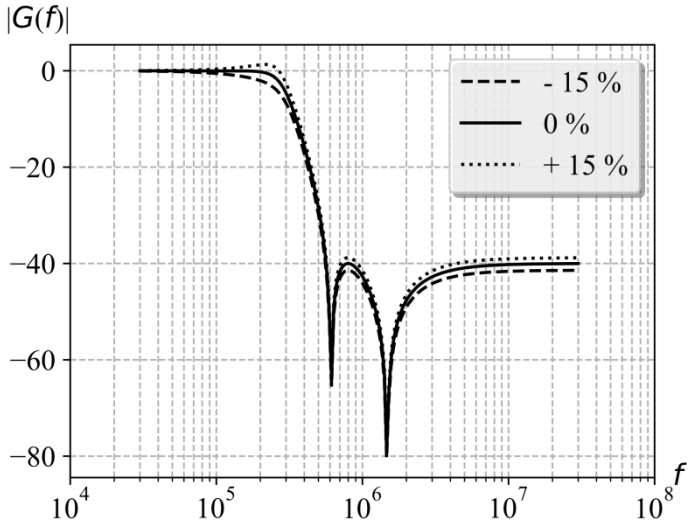

Figura 61: Effetto della variazione del guadagno K di A(f) sul modulo della funzione di trasferimento G(f)

Inclusione di poli e zeri parassiti nella funzione di trasferimento ad anello aperto *A*(*f*) e ricalcolo delle variabili di controllo con l'algoritmo di compensazione

L'interfaccia grafica consente anche di inserire i poli e gli zeri parassiti, in modo che l'algoritmo determini i parametri della funzione di trasferimento ad anello aperto *A*(*f*) che consentono di compensare la presenza di parassiti, facendo in modo che i poli dominanti della funzione di trasferimento ad anello chiuso *G*(*f*) abbiano la posizione desiderata definita delle specifiche dinamiche di progetto.

I poli e gli zeri parassiti possono essere reali o complessi coniugati. Nella sezione dell'interfaccia grafica mostrata in Figura 62, nella casella di testo "paris pole", attivando il relativo pulsante ON, è possibile inserire la frequenza di un polo reale o di un polo complesso coniugato. Nel caso il polo parassita fosse reale il pulsante ON relativo a "Q pole" deve essere disattivo, invece se è complesso coniugato è necessario inserire il valore del fattore di merito

in “Q pole”. Per consentire l’inserimento di più poli parassiti, è possibile inserire delle sequenze di frequenze e di fattori di merito. In questo caso si inserisce la sequenza (o un solo elemento) di frequenze dei poli reali nella casella di testo “real paris pole”, e due sequenze della stessa lunghezza in “paris pole” e “Q pole” dove l’i-esimo elemento della sequenza è relativo alla i-esima coppia di poli complessi coniugati. Per cui la dicitura da inserire è del tipo “[$f_{P0}, f_{P1}, f_{P2}, \ldots$]” in “paris pole” e “[$Q_{P0}, Q_{P1}, Q_{P3}, \ldots$]” in “Q pole”. Discorso analogo per gli zeri parassiti.

Supponendo una coppia di poli complessi coniugati parassiti aventi frequenza naturale di 1MHz e fattore di merito 0.707, è necessario inserire i dati nell’interfaccia grafica come nella Figura 62.

Figura 62: Scelta dei poli parassiti della funzione di trasferimento ad anello aperto A(f)

L’algoritmo determina i seguenti valori delle variabili di controllo, tali da compensare la presenza della coppia di poli complessi coniugati parassiti:

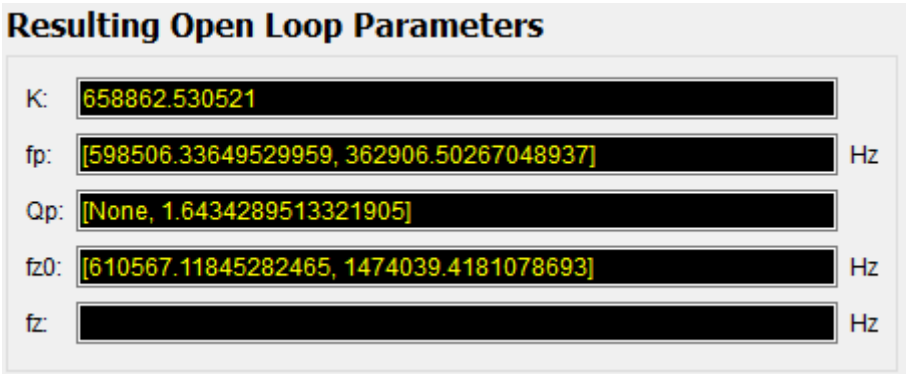

Figura 63: Parametri di A(f) che compensano i parassiti nel caso in esame

Selezionano il diagramma dei poli e degli zeri, si riesce a visualizzare il confronto fra il caso in cui non ci sono parassiti, il caso in cui ci sono parassiti ma non si adotta alcun algoritmo di compensazione e il caso in cui ci sono parassiti e si adotta l’algoritmo di compensazione.

Inoltre, si nota che, adottando l'algoritmo di compensazione, i poli dominanti tornano ad avere la posizione della *G*(*f*) desiderata e che l'ipotesi per la quale i parassiti si spostano poco durante la singola iterazione dell'algoritmo è verificata. Per gli zeri vale sempre il discorso che essendo coincidenti con il modello di *A*(*f*) non subiscono l'effetto dei parassiti.

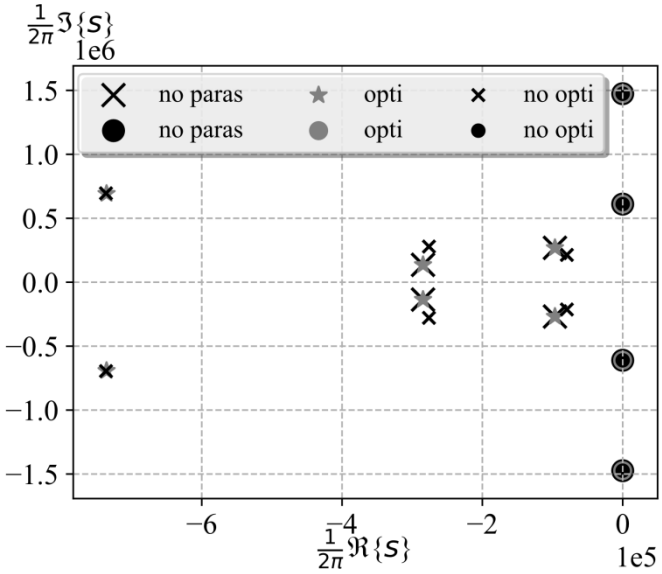

Figura 64: Effetto dei parassiti di A(f) e dell'algoritmo di compensazione sulla posizione dei poli e degli zeri di G(f)

Osservando invece la risposta al gradino unitario si nota come i parassiti determinano una modifica del valore di picco e del settling time non trascurabile. Inoltre, si evidenzia il notevole beneficio della compensazione dei parassiti, grazie al quale il valore di picco della sovraelongazione è lo stesso di quello desiderato e il settling time è quasi il medesimo.

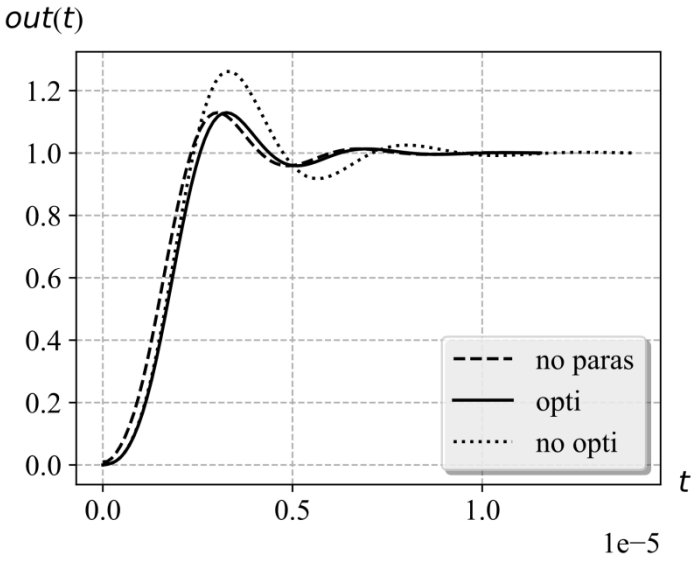

Figura 65 Effetto dei parassiti di A(f) e dell'algoritmo di compensazione sulla risposta al gradino unitario di G(f)

Per quanto riguarda il modulo della funzione di trasferimento, grazie alla compensazione il comportamento in banda passante è quello desiderato e non c'è il picco indesiderato per i PLL di tipo I. In banda attenuata, il comportamento si discosta da quello desiderato in modo inevitabile, per via dell'effetto dei parassiti, senza però invalidare le specifiche di progetto del filtro di attenuazione minima in banda attenuata che vengono abbondantemente soddisfatte.

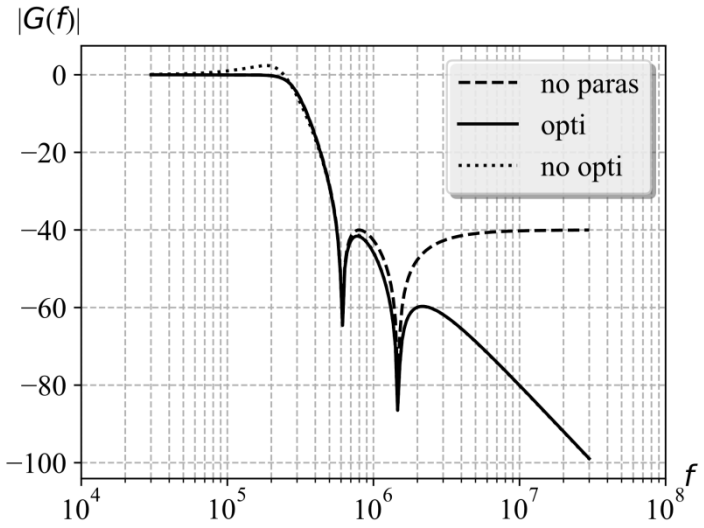

Figura 66: Effetto dei parassiti di A(f) e dell'algoritmo di compensazione sul modulo della funzione di trasferimento G(f)

Analisi del rumore di fase

Il programma dotato di interfaccia grafica per l'assistenza al progetto di PLL consente anche di valutare il rumore di fase in uscita dal PLL. Per fare questo è necessario che il designer inserisca delle specifiche di progetto nella sezione "Noise Parameters" dell'interfaccia come mostrato nella Figura 67. Le specifiche necessarie per valutare il rumore di fase sono:

- la frequenza del segnale di riferimento f_{ref} da inserire nella casella di testo "ref freq".
- la frequenza del segnale di uscita f_{out} da inserire nella casella di testo "out freq".
- il detector noise L_{dn} espresso in dBc/Hz nella casella di testo "det". Nel caso si voglia considerare anche il flicker noise, nella casella di testo è necessario inserire anche la corner frequency f_{corn_dn} mediante una sequenza [L_{dn}, f_{corn_dn}]. Qualora si volesse considerare una pendenza del flicker noise differente da -10dB/decade, è necessario aggiungere alla sequenza anche $slope_{fn\_dn}$, ovvero [L_{dn}, f_{corn_dn}, $slope_{fn\_dn}$].

- il VCO noise L_{VCO} espresso in dBc/Hz nella casella di testo "VCO" e la frequenza di offset alla quale il VCO noise è stato calcolato f_{off_VCO} nella casella di testo "freq off". Nel caso si voglia considerare anche il flicker noise, nella casella di testo "VCO" è necessario inserire anche la corner frequency f_{corn_VCO} mediante una sequenza [L_{VCO}, f_{corn_VCO}]. Qualora si volesse considerare una pendenza del flicker noise differente da -30dB/decade, è necessario aggiungere alla sequenza anche $slope_{fn\_VCO}$, ovvero [L_{VCO}, f_{corn_VCO}, $slope_{fn\_VCO}$].

- L'ordine $m_{\Sigma\Delta}$ del modulatore Σ-Δ, nel caso questo abbia un'architettura MASH, selezionando direttamente l'ordine nei pulsanti numerati denominati con "S-D"

- La funzione di trasferimento del rumore del generico modulatore sigma delta nella forma "ba". Se la $NTF(z)$ è di tipo FIR, ci saranno solo i coefficienti del numeratore e all'interno della casella di testo "S-D" si inserisce una sequenza [b_0, b_1, b_2, …], altrimenti è necessario aggiungere anche la sequenza dei coefficienti del denominatore, per cui la dicitura diventa "[b_0, b_1, b_2, …], [a_0, a_1, a_2, …]"

Il programma consente di analizzare separatamente le varie sorgenti di rumore, abilitando il corrispondente pulsante ON.

Per effettuare l'analisi del rumore è necessario che nella sezione "Resulting Plot" sia selezionato il pulsante "Noise Plot".

Analizzando il sistema come svolto da Perrott in [3] con una f_{ref} = 20MHz e una f_{out} = 1.84GHz, se le specifiche sulle sorgenti di rumore sono L_{dn} = -76dBc/Hz, L_{VCO} = -140dBc/Hz, f_{off_VCO} = 5MHz e $m_{\Sigma\Delta}$ = 3, la sezione "Noise Parameters" dell'interfaccia grafica si presenta come in Figura 67:

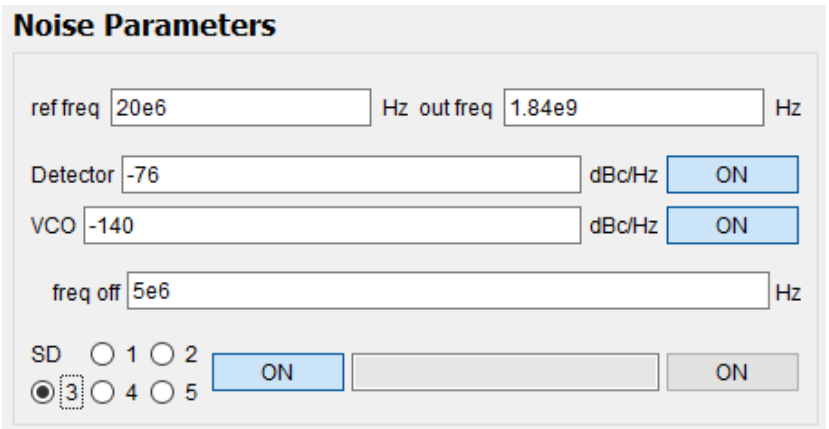

Figura 67: Specifiche sulle sorgenti di rumore

Il programma a questo punto determina direttamente il grafico del rumore di fase in uscita, nel range di default di frequenze compreso fra $f_0/10$ e $100f_0$. Se invece si specificano delle coordinate Xmin, Xmax, Ymin e Ymax il grafico viene visualizzato solo all'interno di questi intervalli. Xmin e Xmax hanno una duplice funzione nel "Noise Plot", infatti questi specificano anche la f_{off_min} e la f_{off_max} del calcolo del rms jitter. Se questi non sono specificati, il programma per default impone f_{off_min}=10Hz e f_{off_max} = 100MHz.

Facendo un'analisi con f_{off} che vanno da $f_0/10$ a $1000f_0$, scegliendo come tali start e stop, Np = 1000 e tipo di incremento logaritmo e selezionando come area di grafico Xmin = $f_0/10$, Xmax = $1000f_0$, Ymin = -250dBc/Hz, Ymax = -50dBc/Hz si ottiene la Figura 68 rappresentante il rumore di fase:

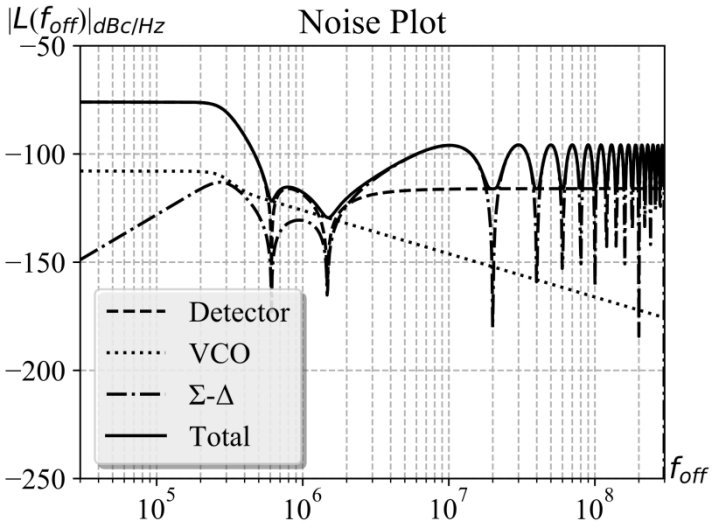

Figura 68: Rumore di fase in uscita del PLL dovuto alle differenti sorgenti di rumore

Dalla Figura 68 si evince che la scelta di utilizzare un filtro Chebyshev di ordine pari, tale per cui a regime l'attenuazione è costante, non è ottimale al fine della riduzione degli effetti del rumore di quantizzazione, il quale diviene la componente dominante quando interviene la coppia di zeri complessi coniugati a frequenza maggiore.

L'interfaccia grafica rende disponibile il risultato sul jitter nella sezione "Resulting Jitter" e nel caso in esame assume il valore:

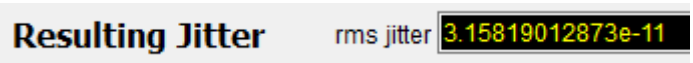

Figura 69: valore quadratico medio del jitter in uscita

Provando a considerare un filtro Chebyshev II di ordine dispari, ad esempio del terzo ordine, il rumore di fase dovrebbe ridursi, in quanto l'attenuazione di $G(f)$ in banda attenuata a regime è di -20dB/dec e non costante. Nel caso in esame il rumore di quantizzazione rimarrebbe comunque la componente dominante dopo i 2 MHz, come mostrato nella Figura 70:

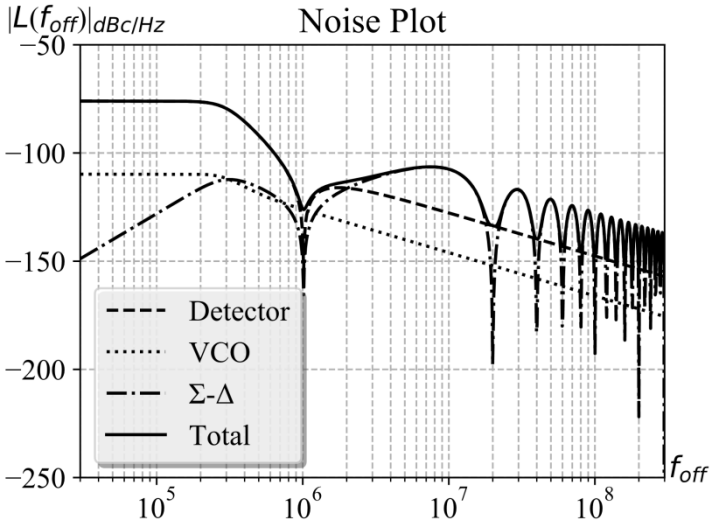

Figura 70: Rumore di fase in uscita del PLL dovuto alle differenti sorgenti di rumore

In questo caso, nonostante l'ordine del filtro sia minore, il jitter in uscita è inferiore al caso del quarto ordine e assume il valore di 12,128ps.

Gli effetti delle coppie di zeri complessi coniugati dei filtri Chebyshev II sono di conseguenza non trascurabili e per fare in modo che il rumore di quantizzazione non sia il contributo dominante sul rumore di fase, conviene scegliere un prototipo privo di zeri, o al massimo un PLL di tipo II come sarà illustrato nel caso successivo.

Capitolo 7.2: Caso di studio PLL di tipo II Butterworth

Considerando un PLL di tipo II privo di zeri (ad esclusione dello zero extra), caratterizzato da un modello $G(f)$ del terzo ordine di Butterworth, banda asintotica di 300KHz come nel caso precedente e rapporto f_z/f_0 pari ad 1/8, si vogliono determinare i parametri della funzione di trasferimento ad anello aperto $A(f)$, considerare la presenza di parassiti, valutare gli effetti della variazione dei parametri e soprattutto confrontare l'analisi del rumore con il caso precedente, includendo eventualmente anche gli effetti del flicker noise.

Calcolo dei parametri della funzione di trasferimento ad anello aperto $A(f)$

Una volta inserite le specifiche nell'interfaccia come illustrato nel caso precedente, l'algoritmo determina i parametri della funzione di trasferimento ad anello aperto $A(f)$ nel caso privo di poli e zeri parassiti. I valori ottenuti delle variabili di controllo sono indicati in Figura 71:

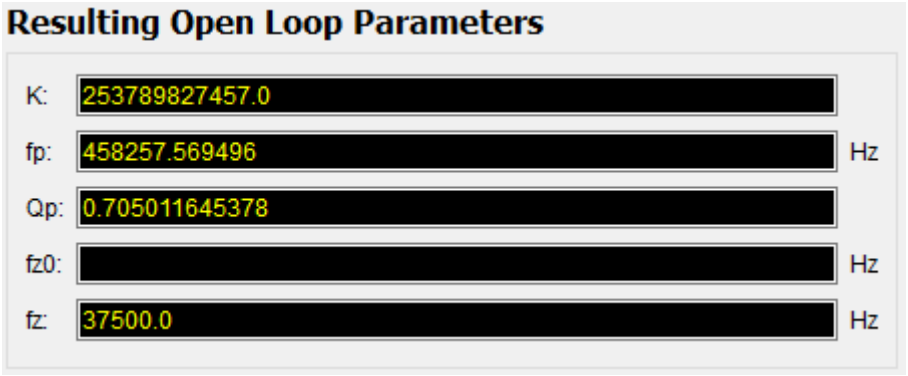

Figura 71: Valore delle variabili di controllo di A(f) che determinano la G(f) desiderata

Il diagramma dei poli e degli zeri mostra la presenza del polo aggiunto dallo stadio di integrazione aggiuntivo del PLL di tipo II e dello zero extra utile per la stabilità del sistema. Da questo si determina il valore di f_{cp} utile per valutare il valore di picco indesiderato del modulo della funzione di trasferimento in banda passante. Nel caso in esame la $f_{cp} = \omega_{cp}/2\pi =$ 50KHz.

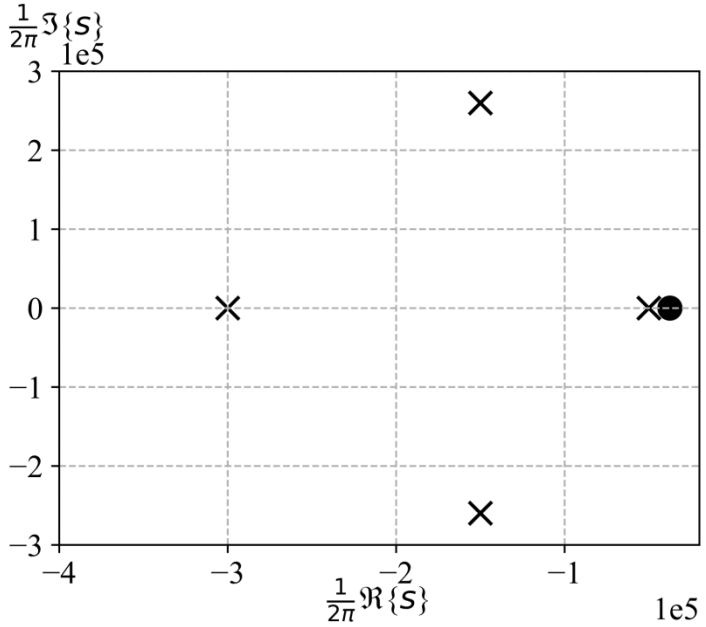

Figura 72: Diagramma dei poli e degli zeri di G(f) del 3°ordine di tipo II, Butterworth con f_0 300KHz e f_z/f_0=1/8

La risposta al gradino unitario presenta un picco della sovraelongazione più elevato rispetto all'analogo PLL di tipo I e anche un settling time più duraturo. Si rammenda che per ridurre il settling time nei PLL di tipo II è necessario scegliere rapporti f_z/f_0 più alti, ma questo va a discapito dell'ampiezza del picco della sovraelongazione della risposta che diventa sempre più alto.

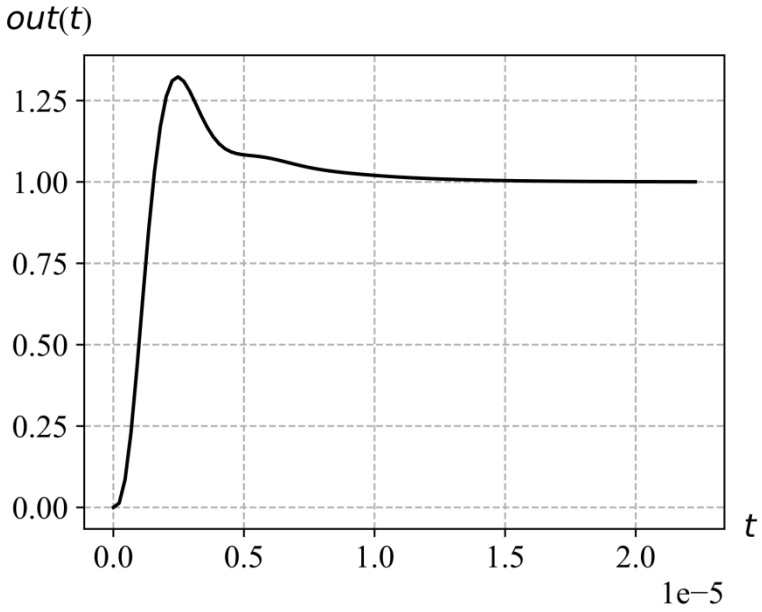

Figura 73: Risposta al gradino unitario di G(f) del 3°ordine di tipo II, Butterworth con f_0 300KHz e f_z/f_0=1/8

Risulta interessante osservare l'effetto del dello zero aggiuntivo e del polo aggiunto dallo stadio di integrazione sul comportamento in banda passante della $G(f)$.

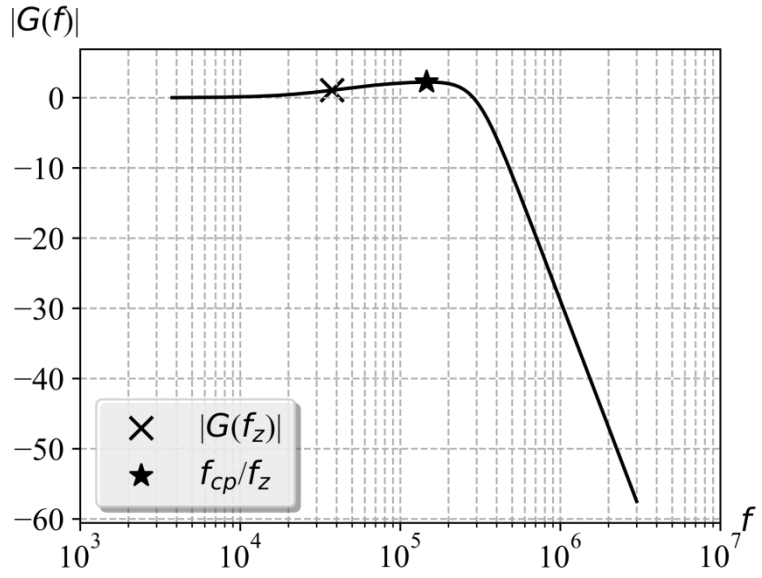

Figura 74: Modulo della funzione di trasferimento G(f) del 3°ordine di tipo II, Butterworth con f_0 300KHz e f_z/f_0=1/8

Come si osserva dalla Figura 74, la presenza dello zero determina un picco indesiderato in banda passante, ma inevitabile ai fini della stabilità del sistema. L'ampiezza del picco espressa in dB è circa f_{cp}/f_z, quindi nel caso in esame è circa 2.5dB.

Valutazione dell'impatto della variazione dei parametri della funzione di trasferimento ad anello aperto *A*(*f*)

Si vuole ora valutare a titolo di esempio l'effetto della variazione della frequenza della coppia di poli complessi coniugati rispetto al valore nominale determinato precedentemente, in un intervallo [-20%, +20%] con uno step del 20%. Si ricorda che si suppone che le variabili di controllo siano indipendenti tra di loro, ovvero che sia possibile variare il singolo parametro della *A*(*f*), mantenendo costanti gli altri.

Osservando il diagramma dei poli e degli zeri in Figura 75, l'effetto della variazione della frequenza della coppia di poli complessi coniugati determina uno spostamento non lineare della posizione dei poli dominanti della *G*(*f*), mentre ha effetto pressoché trascurabile sulla posizione del polo determinato dallo stadio di integrazione aggiuntivo.

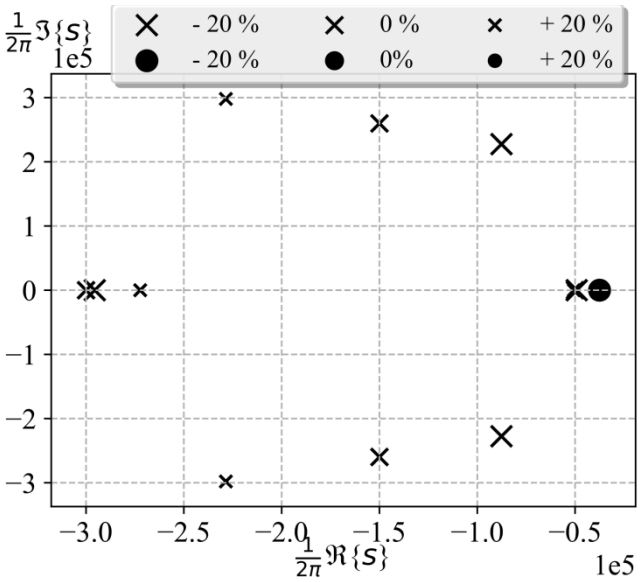

Figura 75: Effetto della variazione della frequenza della coppia di poli complessi coniugati di A(f) sulla posizione dei poli e degli zeri di G(f)

Valutando le differenti risposte al gradino unitario dovute alle differenti frequenze della coppia di poli complessi coniugati di *A*(*f*), si nota che all'aumentare di queste si riduce il picco della sovraelongazione e di conseguenza anche il settling time.

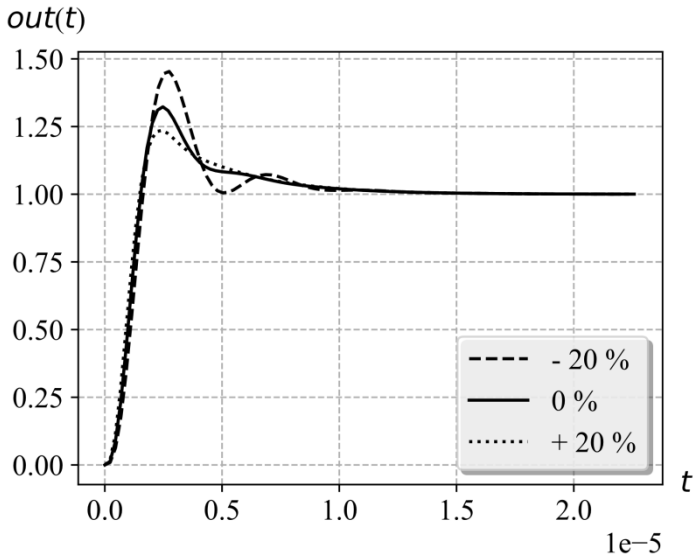

Figura 76: Effetto della variazione della frequenza della coppia di poli complessi coniugati di A(f) sulla risposta al gradino unitario di G(f)

Per quanto riguarda il modulo della funzione di trasferimento, per valori inferiori rispetto a quelli nominali della variabile di controllo alterata il comportamento in banda passante peggiora, perché si risalta il picco indesiderato, mentre in banda attenuata si ha un'attenuazione maggiore con la stessa pendenza. Per valori superiori, invece, il

comportamento in banda passante migliora a discapito della minore attenuazione a parità di frequenza rispetto al caso nominale.

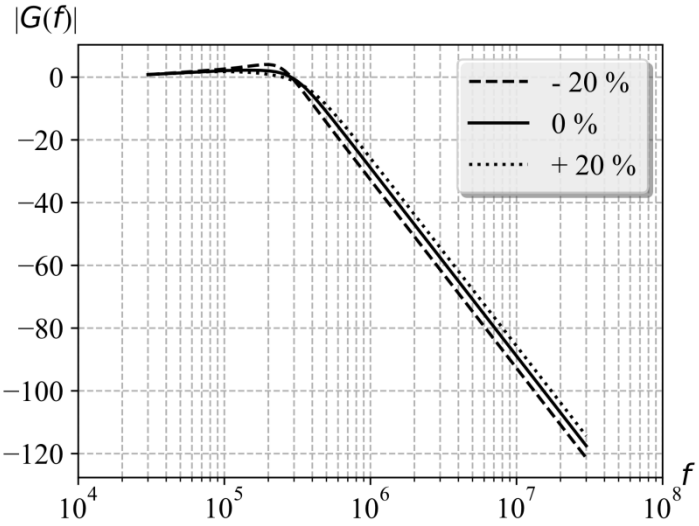

Figura 77: Effetto della variazione della frequenza della coppia di poli complessi coniugati di A(f) sul modulo della funzione di trasferimento G(f)

Inclusione di poli e zeri parassiti nella funzione di trasferimento ad anello aperto *A(f)* e ricalcolo delle variabili di controllo con l'algoritmo di compensazione

Si vuole ora valutare l'efficacia dell'algoritmo di compensazione dei poli e degli zeri parassiti, considerando uno zero parassita reale alla frequenza di 8MHz, un polo parassita reale alla frequenza di 1.5MHz, una coppia di poli parassiti complessi coniugati con frequenza 3,5MHz e fattore di merito pari a 3,5.

Nell'interfaccia grafica l'input dei parassiti, supposti noti da simulazioni dei componenti che costituiscono il PLL, è la seguente:

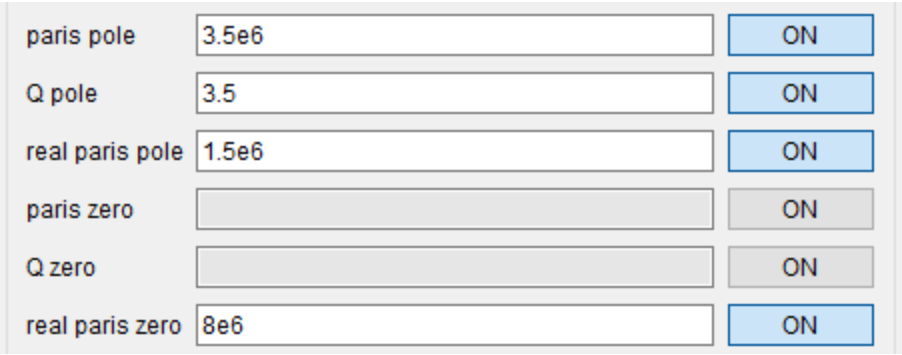

Figura 78: Scelta dei poli e degli zeri parassiti della funzione di trasferimento ad anello aperto A(f)

L’algoritmo determina i seguenti valori delle variabili di controllo tali da compensare la presenza dei parassiti:

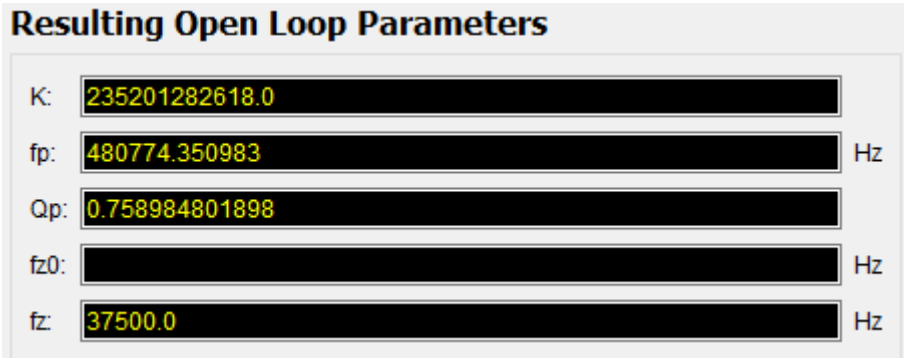

Figura 79: Parametri di A(f) che compensano l’effetto dii parassiti nel caso in esame

Si nota quindi che, in questo caso, per compensare l’effetto dei parassiti, è necessario aumentare il guadagno, la frequenza e il fattore di merito della coppia di poli complessi coniugati di $A(f)$.

Il diagramma dei poli e degli zeri risultante di Figura 80 dimostra la validità dell’ipotesi dell’algoritmo di compensazione per la quale i parassiti non subiscono spostamenti apprezzabili della loro posizione nello spazio S, avendo questi una frequenza sufficientemente maggiore della banda della $G(f)$.

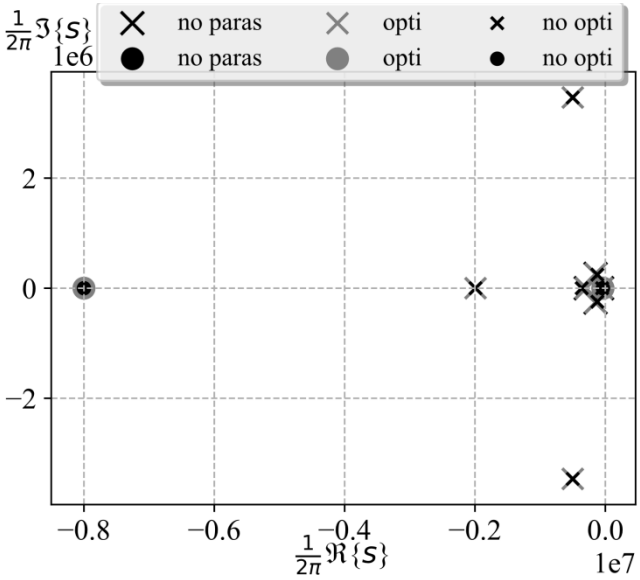

Figura 80: Effetto dei parassiti di A(f) e dell’algoritmo di compensazione sulla posizione dei poli e degli zeri di G(f)

Per verificare l’effettivo funzionamento dell’algoritmo di compensazione, è possibile fare una selezione dell’area di grafico impostando opportunamente i valori di Xmin, Xmax, Ymin e Ymax che nel caso riportato in Figura 81 sono rispettivamente -0.4e6, -0.02e6, -0.3e6, 0.3e6.

Si nota che i poli dominanti, senza algoritmo di compensazione, vengono spostati nel piano per effetto dei parassiti, mentre adottando l'ottimizzazione delle variabili di controllo, i poli dominanti tornano ad avere la posizione desiderata dettata dalle specifiche sulla dinamica del sistema.

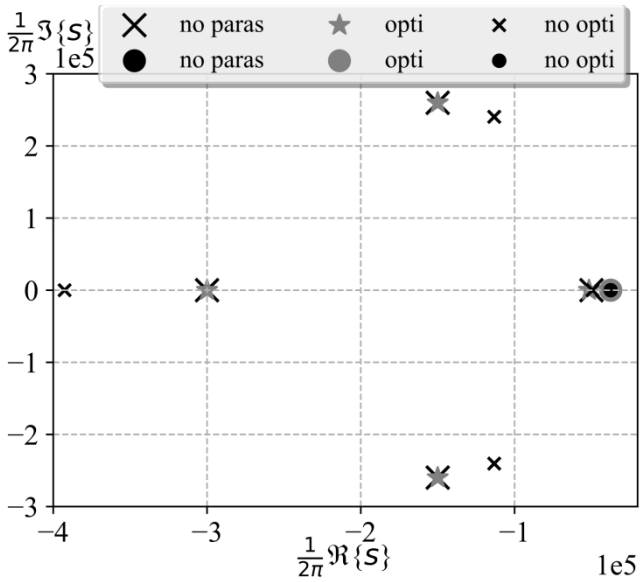

Figura 81: Effetto dei parassiti di A(f) e dell'algoritmo di compensazione sulla posizione dei poli dominanti di G(f)

Osservando invece la risposta al gradino unitario, risulta ancora più evidente l'effetto di compensazione, con caratteristica prossima al caso privo di parassiti, ma avente picco della sovraelongazione e settling time leggermente superiori.

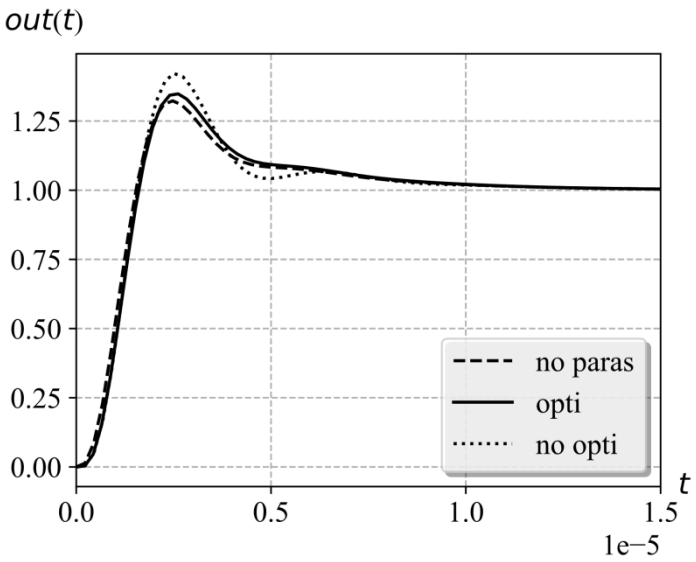

Figura 82 Effetto dei parassiti di A(f) e dell'algoritmo di compensazione sulla risposta al gradino unitario di G(f)

Osservando il modulo della funzione di trasferimento risultante, si nota un comportamento atteso in banda passante grazie alla compensazione che riduce il picco indesiderato dovuto sia dal fatto che il PLL è di tipo II sia dai parassiti, ma si manifesta anche un picco in banda attenuata dovuto alla coppia di poli parassiti complessi coniugati. A regime la pendenza non sarà più di -60dB/dec, bensì di -100dB/dec dovuto alla presenza di due poli parassiti in più rispetto agli zeri parassiti.

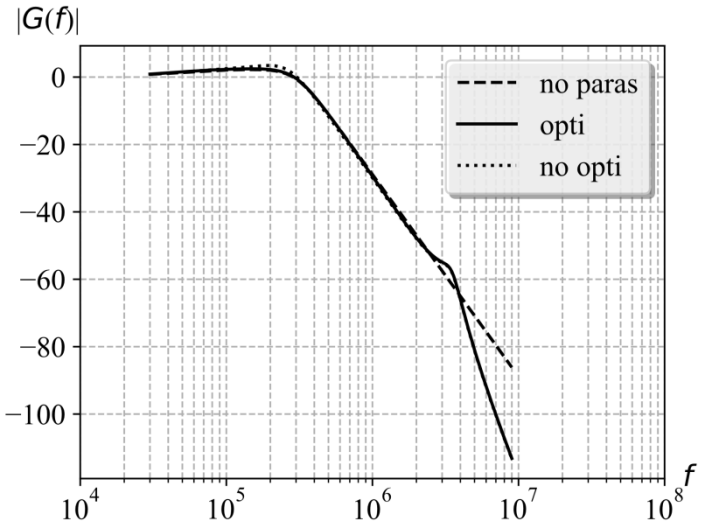

Figura 83: Effetto dei parassiti di A(f) e dell'algoritmo di compensazione sul modulo della funzione di trasferimento G(f)

Analisi dettagliata del rumore di fase

Si ipotizza di analizzare il rumore di fase in uscita dal PLL come svolto da Perrott in [3] con una f_{ref} = 20MHz e una f_{out} = 1.84GHz. Se le specifiche sulle sorgenti di rumore sono L_{dn} = -76dBc/Hz, L_{VCO} = -140dBc/Hz, f_{off_VCO} = 5MHz e $m_{\Sigma\Delta}$ = 3, facendo un'analisi con f_{off} che vanno da 10KHz a 100MHz scegliendo come tali start e stop, Np 1000 e tipo di incremento logaritmo, selezionando come area di grafico Xmin = 10KHz, Xmax = 100MHz, Ymin = -160dBc/Hz e Ymax= -60dBc/Hz, si ottiene il grafico del rumore di fase di Figura 84.

Osservandolo, si nota che il contributo del rumore di quantizzazione sul rumore di fase in uscita del PLL diventa il contributo dominante nel range di frequenze compreso fra 2MHz e 10MHz e che a differenza dei PLL di tipo I c'è il picco indesiderato anche nel rumore di fase.

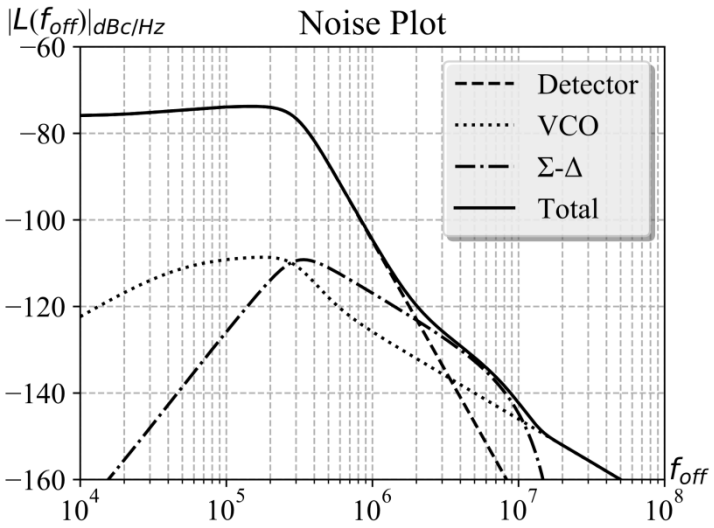

Figura 84: Rumore di fase in uscita del PLL dovuto alle differenti sorgenti di rumore

In questo caso il valore quadratico medio del jitter in uscita è di circa 14.2ps, che è maggiore rispetto al caso Chebyshev II del terzo ordine, che, considerando gli stessi estremi di integrazione, è pari a 12.8ps. Questo solo grazie alla caratteristica più selettiva.

In particolare, si nota che a confronto del caso Chebyshev II (sia di quarto ordine che di terzo ordine), il rumore di fase a frequenze superiori a circa 1.3MHz è nettamente inferiore. Si riporta il confronto fra il rumore di fase in uscita nei differenti casi:

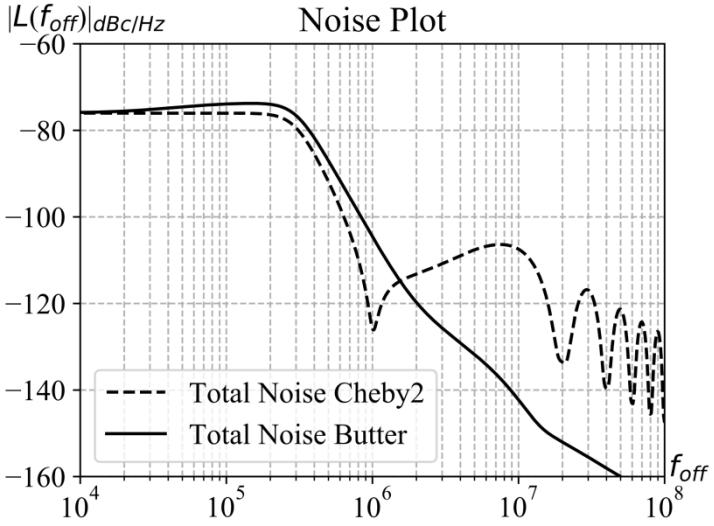

Figura 85: Rumore di fase in uscita del PLL dovuto alle differenti sorgenti di rumore

Inclusione del flicker noise come componente del detector noise

Si vuole ora analizzare l'effetto del flicker noise introdotto dalla pompa di carica ed eventualmente anche dal buffer del segnale di riferimento. Questo è identificato dalla corner frequency e si analizza il caso in cui L_{dn} = -90 dBc/Hz, f_{corn_dn} = 1KHz, L_{VCO} = -140dBc/Hz, f_{off_VCO} = 5MHz e $m_{\Sigma\Delta}$ = 3, con f_{off_min} = 10Hz e f_{off_max} = 100MHz per la valutazione del valore quadratico medio del jitter.

Il rumore di fase risultante è rappresentato in Figura 86:

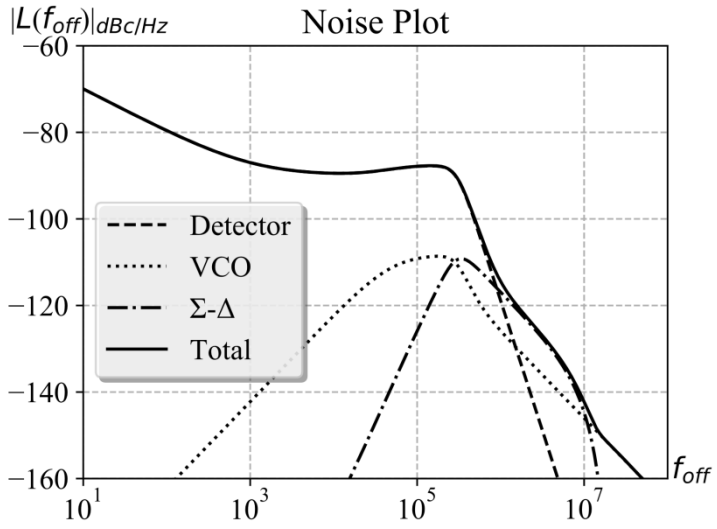

Figura 86: Effetto del flicker noise classico di -10dB/decade sul rumore di fase in uscita del PLL dovuto alle differenti sorgenti di rumore

Si osserva che a basse frequenze il contributo del flicker noise sul rumore di fase è tutt'altro che trascurabile. Nel caso in esame essendo L_{dn} inferiore rispetto al caso precedente, il rms jitter è nettamente inferiore e pari a 3,357 ps. Confrontandolo però nelle stesse condizioni di L_{dn} = -90dBc senza flicker noise in cui rms jitter è pari a 3,325ps, si nota l'effettivo aumento del jitter dovuto al flicker noise.

Se invece si considerasse una pendenza del flicker noise differente dalla classica -10dB/decade, si inserisce anche il parametro $slope_{fn\_dn}$ e supponendolo pari a -15, nelle stesse condizioni precedenti di L_{dn} = -90 dBc/Hz, f_{corn_dn} = 1KHz, L_{VCO} = -140dBc/Hz, f_{off_VCO} = 5MHz e $m_{\Sigma\Delta}$ = 3, valutando il jitter con f_{off_min} = 10Hz e f_{off_max} = 100MHz, si ottiene il rumore di fase in uscita di Figura 87.

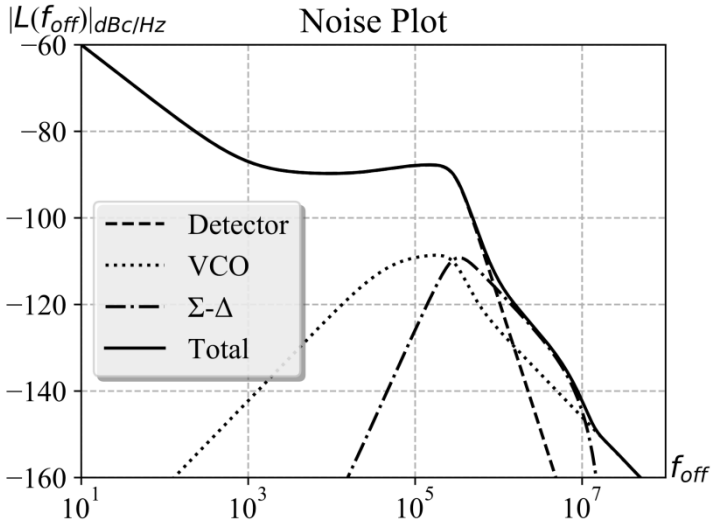

Figura 87: Effetto del flicker noise di -15dB/decade sul rumore di fase in uscita del PLL dovuto alle differenti sorgenti di rumore

Si ha quindi un peggioramento del rumore di fase alle basse frequenze di offset e anche un peggioramento del rms jitter rispetto al caso del flicker noise classico, che in questo caso vale 3.38ps.

Inclusione del flicker noise come componente del VCO noise

Discorso analogo per il flicker noise introdotto dal VCO riportato all'ingresso del VCO.

Considerando L_{VCO} = -140 dBc/Hz calcolato alla f_{off_VCO} = 5MHz e f_{corn_VCO} = 1KHz, L_{dn} = -90 dBc/Hz e $m_{\Sigma\Delta}$ = 3, valutando il rms jitter fra le frequenze di offset f_{off_min} = 10Hz e f_{off_max} = 100MHz si ottiene il grafico di Figura 88.

Si osserva una differente pendenza della curva del contributo del VCO noise sul rumore di fase a partire dalla corner frequency di 1KHz.

In questo caso, non essendo a basse frequenze il VCO noise la componente dominante sul rumore di fase, in quanto soggetto ad un filtraggio di tipo passa alto ad opera del PLL, la variazione del rms jitter rispetto al caso senza flicker noise è minima, ovvero 3,3262ps a confronto di 3,3254ps.

:

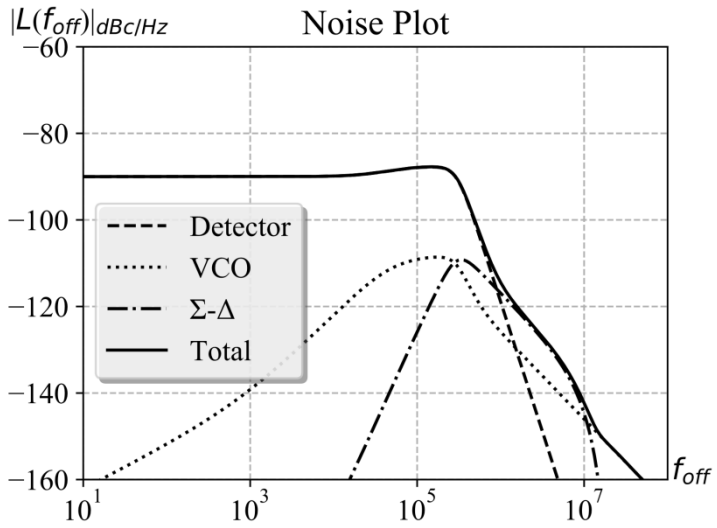

Figura 88: Effetto del flicker noise classico di -30dB/decade introdotto dal VCO noise, sul rumore di fase in uscita del PLL dovuto alle differenti sorgenti di rumore

Considerando invece una differente pendenza del flicker noise introdotto da VCO, ovvero L_{VCO} = -140 dBc/Hz calcolato alla f_{off_VCO} = 5MHz, f_{corn_VCO} = 1KHz e $slope_{fn\_dn}$ = -35, L_{dn} = -90 dBc/Hz e $m_{\Sigma\Delta}$ = 3 e valutando il rms jitter fra le frequenze di offset f_{off_min} = 10Hz e f_{off_max} = 100MHz si ottiene:

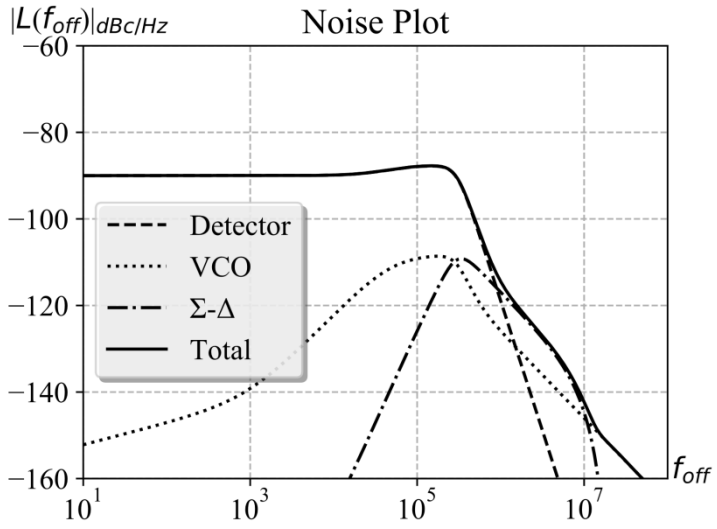

Figura 89: Effetto del flicker noise di -35dB/decade introdotto dal VCO noise, sul rumore di fase in uscita del PLL dovuto alle differenti sorgenti di rumore

Anche in questo caso, non essendo il VCO noise la componente dominante sul rumore di fase a basse frequenze, in quanto soggetto ad un filtraggio di tipo passa alto ad opera del PLL, la variazione del rms jitter rispetto al caso senza flicker noise è minima, ovvero 3,3255ps a confronto di 3,3254ps

Per visualizzare la caratteristica tipica del flicker noise introdotto dal solo VCO, è necessario togliere l'influenza del PLL, essendo il VCO noise soggetto ad un filtraggio di tipo passa alto dal PLL. Per far questo basta imporre una f_0 molto bassa (nell'esempio 1Hz) e si ottiene:

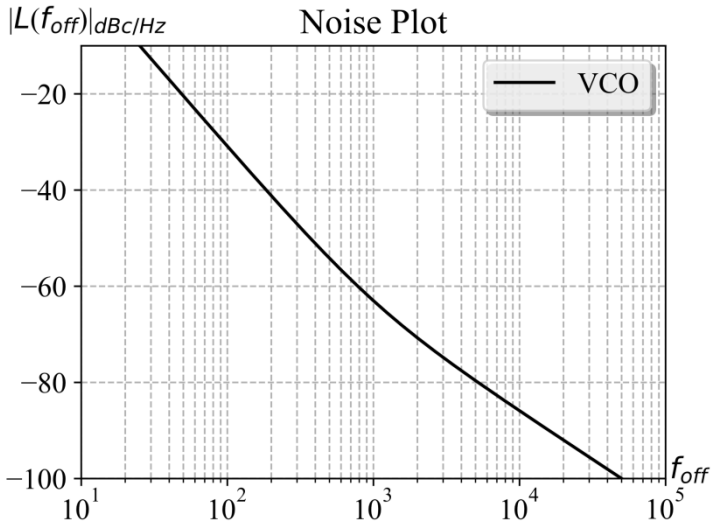

Figura 90: Componente del VCO noise riportata all'ingresso del VCO soggetta ad un flicker noise di -35dB/decade

Valutazione dell'effetto dell'inclusione di poli e zeri parassiti nella funzione di trasferimento ad anello aperto *A*(*f*) sul rumore di fase

Si vuole ora valutare il rumore di fase in uscita nel caso in cui ci siano i parassiti esaminati in precedenza e dai quali sono già stati determinati i parametri della funzione di trasferimento in anello aperto tali da compensarli.

Quindi, nell'ipotesi in cui ci sia uno zero parassita reale alla frequenza di 8MHz, un polo parassita reale alla frequenza di 2MHz, una coppia di poli parassiti complessi coniugati con frequenza 3,5MHz e fattore di merito pari a 3,5, supponendo L_{dn} = -76 dBc/Hz, L_{VCO} = -140 dBc/Hz calcolato alla f_{off_VCO} = 5MHz e f_{corn_VCO} = 1KHz, valutando il rms jitter fra le frequenze di offset f_{off_min} = 10Hz e f_{off_max} = 100MHz si ottiene la Figura 91.

Il questo caso i parassiti hanno un effetto benevolo sul rumore di quantizzazione, in quanto quest'ultimo ha un contributo dominante solo nel range di frequenze compreso fra i 2MHz e i 5MHz. In particolare, si nota il picco dovuto alla coppia di poli parassiti complessi coniugati

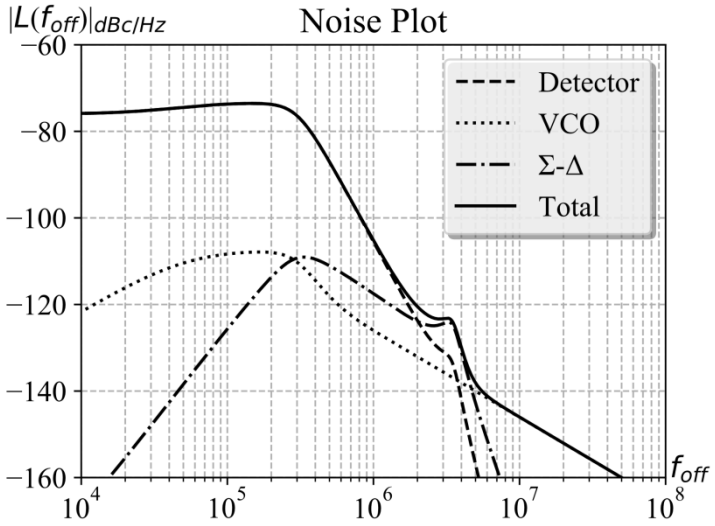

Figura 91: Rumore di fase in uscita del PLL dovuto alle differenti sorgenti di rumore, con A(f) soggetta a poli e zeri parassiti

Il rms jitter è pari a 14.4916ps, ovvero superiore rispetto al caso senza parassiti in cui era 14.2064ps, dovuto al picco introdotto dalla coppia di poli parassiti complessi coniugati.

Questo suggerisce la possibilità di non avere il rumore di quantizzazione come contributo dominante sul rumore di fase in uscita. Infatti, nel caso ci fosse solo un solo polo parassita ad una frequenza sufficientemente distante dalla banda del PLL, è possibile che il contributo del rumore di quantizzazione sul rumore di fase in uscita non sia mai dominante.

Infatti, nel caso di un singolo polo parassita alla frequenza pari a 1MHz, si ottiene la Figura 92 e il valore efficace del jitter è di 3.356ps.

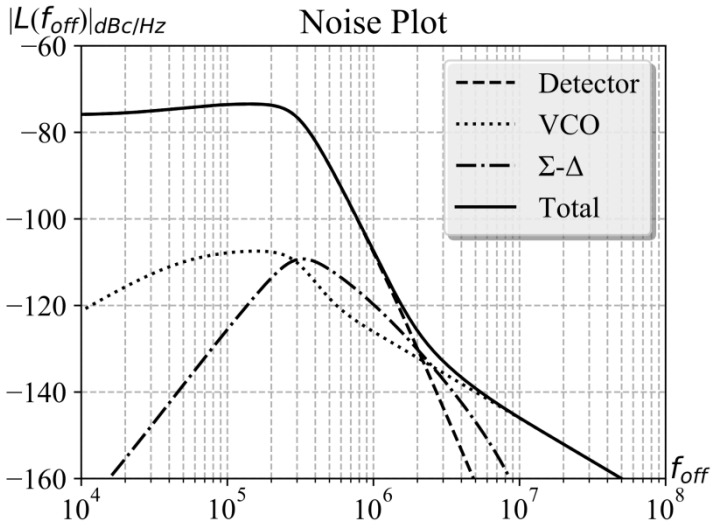

Figura 92: Rumore di fase in uscita del PLL dovuto alle differenti sorgenti di rumore, con A(f) soggetta a un polo parassita tale per cui il rumore di quantizzazione non è mai l'effetto dominante

Valutazione dell'impatto della variazione dei parametri della funzione di trasferimento ad anello aperto *A*(*f*) sul rumore di fase

L'interfaccia grafica consente anche di verificare l'effetto della variazione dei parametri della funzione di trasferimento ad anello aperto sul rumore di fase e di esplicitare nella sezione "Resulting Jitter" il valore massimo e il valore minimo del valore quadratico medio del jitter delle valutazioni effettuate con le differenti configurazioni di parametri.

A partire dall'ultimo caso in esame, ovvero considerando un singolo polo reale parassita alla frequenza di 1MHz e supponendo L_{dn} = -76 dBc/Hz, L_{VCO} = -140 dBc/Hz calcolato alla f_{off_VCO} = 5MHz e f_{corn_VCO} = 1KHz, valutando il rms jitter fra le frequenze di offset f_{off_min} = 10Hz e f_{off_max} = 100MHz, nel caso si volesse valutare una variazione percentuale nel range [-20%, +20%] rispetto al valore nominale della frequenza della coppia di poli complessi coniugati di *A*(*f*), si ottiene:

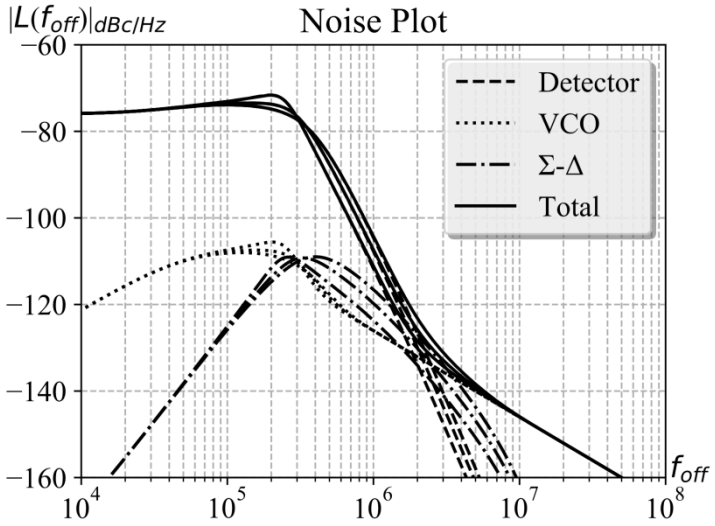

Figura 93: Effetto della variazione del 20% della frequenza della coppia di poli complessi coniugati di A(f) sul rumore di fase in uscita del PLL dovuto alle differenti sorgenti di rumore, con A(f) soggetta a un polo parassita tale per cui il rumore di quantizzazione non è mai l'effetto dominante

Come si osserva, in ogni caso il rumore di quantizzazione non rappresenta la componente dominante sul rumore di fase, il quale ha un picco indesiderato ancora più elevato per frequenze inferiori a quella determinata dall'algoritmo di compensazione del polo parassita.

Il programma determina anche il massimo e il minimo fra le differenti configurazioni dei parametri valutate del rms jitter e il risultato è visualizzato nell'interfaccia grafica come in Figura 94:

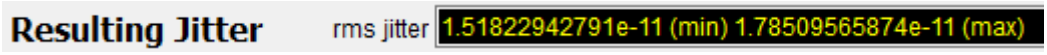

Figura 94: valore minimo e massimo del rms jitter in uscita

Valutazione dell'effetto del rumore di quantizzazione introdotto da un modulatore Σ-Δ caratterizzato da una generica *NTF*(*z*) sul rumore di fase

L'interfaccia grafica consente anche di determinare il rumore di fase in uscita, fornendo una specifica *NTF*(*z*) del modulatore Σ-Δ.

Sopponendo di voler inserire una *NTF*(*z*) di tipo FIR come ad esempio $NTF(z)=1-3z^{-1}+3z^{-2}-3z^{-3}$, considerando le restanti sorgenti di rumore come nel caso precedente, compreso il polo parassita alla frequenza di 1MHz, si ottiene il medesimo risultato dell'architettura MASH di ordine 3.

Nel caso di *NTF*(*z*) espressa nella forma "ba" come ad esempio:

$$NTF(z) = \frac{1 - 3z^{-1} + 3z^{-2} - z^{-3}}{1 + 0.5z^{-1}} \qquad (177)$$

si ottiene:

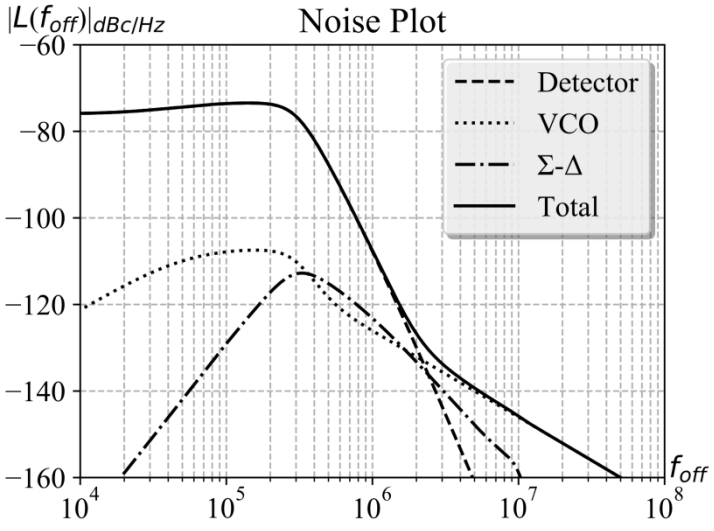

Figura 95: Rumore di fase in uscita del PLL dovuto alle differenti sorgenti di rumore, con A(f) soggetta a un polo parassita tale per cui il rumore di quantizzazione non è mai l'effetto dominante

Capitolo 8

UTILIZZO PROGRAMMATICO DELLE FUNZIONI PYTHON DELL'ASSISTENTE AL PROGETTO DI PLL

A differenza del software messo a disposizione gratuitamente da Perrott sotto forma di sola interfaccia grafica, il cui codice è chiuso e proprietario, il software realizzato in Python per l'assistenza al progetto di PLL consente non solo di essere impiegato da un operatore tramite l'interfaccia grafica, ma anche di essere utilizzato in maniera programmatica, ovvero di essere richiamato come una libreria di funzioni da altri programmi.

Per cui è possibile incorporare il software in progetti più ampi, come ad esempio PyDSM ed essere sfruttato per progettare il modulatore Σ-Δ e il suo quantizzatore in modo che il rumore di quantizzazione in uscita dallo stesso sia il migliore possibile in relazione a come il PLL lo propaga in uscita.

Capitolo 8.1: Signature delle funzioni Python principali messe a disposizione dall'assistente al progetto di PLL

Le signature delle funzioni principali messe a disposizione dall'assistente al progetto di PLL programmato in Python sono le seguenti:

- G = **ideal_G** (N, f0, ftype, rp=None, rs=None, output='ba')

  Questa funzione consente di determinare la funzione di trasferimento ad anello chiuso *G*(*f*) desiderata a partire dalle specifiche sull'ordine, sulla banda asintotica, sul tipo di prototipo, sul ripple massimo in banda passante e sull'attenuazione minima in banda attenuata.

  Gli input della funzione sono:

  - N, numero intero che indica la specifica sul roll-off ovvero l'ordine della *G*(*f*) desiderata;

- f0, numero reale che indica frequenza di taglio asintotica [Hz] della *G*(*f*) desiderata;
- ftype, stringa che indica la forma del prototipo e può essere 'butter', 'bessel', cheby1', 'cheby2' o 'ellip';
- rp, numero reale che esprime il ripple massimo in banda passante [dB]. È un parametro opzionale perché deve essere None per prototipi Butterworth, Bessel e Chebyshev II;
- rs, numero reale che esprime l'attenuazione minima in banda attenuata [dB]. È un parametro opzionale perché deve essere None per prototipi Butterworth, Bessel e Chebyshev I;
- output, è una stringa opzionale che consente di scegliere fra la forma 'ba' e la forma 'zpk' come uscita della funzione di trasferimento della *G*(*f*) desiderata.

L'output della funzione è:

- una tupla di due array contenenti i coefficienti dei polinomi del numeratore e del denominatore della *G*(*f*) desiderata se output = 'ba';
- una tupla di due array e un numero reale. Gli array contengono i rispettivamente gli zeri e i poli della *G*(*f*) desiderata, mentre il numero reale è il guadagno se output = 'zpk'.

- A, p = **G2A_typeI** (G, N, ftype, rp, output='ba')

  Questa funzione consente di determinare la funzione di trasferimento ad anello aperto *A*(*f*) nel caso di PLL di tipo I a partire dalle specifiche sull'ordine, sul tipo di prototipo, sul ripple massimo in banda passante e sulla funzione di trasferimento della *G*(*f*) desiderata.

  Gli input della funzione sono:

  - G, tupla contenente la forma 'ba' o la forma 'zpk' della funzione di trasferimento ad anello chiuso *G*(*f*) desiderata;
  - N, ftype, rp e output sono gli stessi definiti per la funzione ideal_G( ).

Gli output della funzione sono:

- A, tupla contenente la forma ‘ba’ o la forma ‘zpk’ della funzione di trasferimento ad anello aperto $A(f)$;
- p, tupla di parametri utili per il calcolo delle variabili di controllo. In questo caso contiene un solo numero reale wk corrispondente alla ω_K indicata nei capitoli precedenti, ovvero al guadagno della $A(f)$.

- A, p = **G2A_typeII** (G, fz, N, ftype, rp, output='ba')

  Questa funzione consente di determinare la funzione di trasferimento ad anello aperto $A(f)$ nel caso di PLL di tipo II a partire dalle specifiche sulla frequenza dello zero aggiuntivo, sull’ordine, sul tipo di prototipo, sul ripple massimo in banda passante e sulla funzione di trasferimento della $G(f)$ desiderata.

  Gli input della funzione sono:

  - G, tupla contenente la forma ‘ba’ o la forma ‘zpk’ della funzione di trasferimento ad anello chiuso $G(f)$ desiderata;
  - fz, numero reale che indica la frequenza [Hz] dello zero aggiuntivo per PLL di tipo II;
  - N, ftype, rp e output sono gli stessi definiti per la funzione ideal_G( ).

  Gli output della funzione sono:

  - A, tupla contenente la forma ‘ba’ o la forma ‘zpk’ della funzione di trasferimento ad anello aperto $A(f)$;
  - p, tupla di parametri utili per il calcolo delle variabili di controllo. In questo caso contiene due numeri reali, il primo è wk corrispondente alla ω_K che è la radice quadrata del guadagno di $A(f)$, il secondo è wz corrispondente alla ω_z ovvero la pulsazione naturale dello zero aggiuntivo, il terzo è wcp corrispondente alla ω_{cp} ovvero la pulsazione naturale del polo introdotto dallo stadio di integrazione aggiuntivo per PLL di tipo II.

- G = **A2G** (A, N, ftype, rp, output='ba')

  Questa funzione consente di determinare la funzione di trasferimento in catena chiusa *G*(*f*) a partire dalla funzione di trasferimento in catena aperta *A*(*f*) e alle specifiche sull'ordine, sul tipo di prototipo, sul ripple massimo in banda passante.

  Gli input della funzione sono:

  - A, tupla contenente la forma 'ba' o la forma 'zpk' della funzione di trasferimento ad anello aperto *A*(*f*);
  - N, ftype, rp e output sono gli stessi definiti per la funzione ideal_G( ).

  L'output della funzione è:

  - G, tupla contenente la forma 'ba' o la forma 'zpk' della funzione di trasferimento ad anello chiuso *G*(*f*) determinata a partire dalla generica *A*(*f*).

- H = **A2H** (A, typeLF, K1=1, output='ba')

  Questa funzione consente di determinare la funzione di trasferimento del Loop Filter *H*(*f*) a partire dalla funzione di trasferimento in catena aperta *A*(*f*) e dalle specifiche sul tipo di PLL, sulla topologia del PFD, sul valore nominale del rapporto fra la frequenza di uscita del PLL e la frequenza di riferimento, sul guadagno della pompa di carica e sul guadagno del VCO.

  Gli input della funzione sono:

  - A, tupla contenente la forma 'ba' o la forma 'zpk' della funzione di trasferimento ad anello aperto *A*(*f*);
  - typeLF, numero intero pari a 1 o 2 che indica il tipo di PLL;
  - K1, numero reale che dipende dai parametri del PLL secondo la formula:

$$\mathrm{K1} = \frac{N_{nom}}{\mathrm{K_V} I_{cp} \alpha}$$

  - output, stringa che consente di scegliere la forma 'zpk' o 'ba' per l'uscita della funzione.

L'output della funzione è:

- H, tupla contenente la forma 'ba' o la forma 'zpk' della funzione del Loop Filter $H(f)$ determinata a partire dalla generica $A(f)$.

- A = **H2A** (A, K1=1, output='ba')

  Questa funzione consente di determinare la funzione di trasferimento in catena aperta $A(f)$ a partire dalla funzione di trasferimento del Loop Filter $H(f)$ e dalle specifiche sul tipo di PLL, sulla topologia del PFD, sul valore nominale del rapporto fra la frequenza di uscita del PLL e la frequenza di riferimento, sul guadagno della pompa di carica e sul guadagno del VCO. Gli input e gli output sono analoghi alla funzione A2H( ).

- p = **characterize_G** (G, N, typeLF, ftype, f0, fz_f0, rp, K1=1)

  Questa funzione consente di determinare i parametri caratteristici dei modelli delle funzioni di trasferimento $H(f)$ e $G(f)$ a partire dalle specifiche sulla funzione di trasferimento ad anello chiuso desiderata.

  Gli input della funzione sono:

  - G, tupla contenente la forma 'ba' o la forma 'zpk' della funzione di trasferimento ad anello chiuso desiderata;
  - typeLF, numero intero pari a 1 o 2 che indica il tipo di PLL;
  - fz_f0, numero reale che indica la posizione relativa dello zero aggiuntivo rispetto alla frequenza di taglio asintotica per PLL di tipo II, corrispondente alla specifica $fz/f0$ indicata nei capitoli precedenti;
  - ftype, f0, rp, K1 e output sono gli stessi definiti per le funzioni ideal_G( ) e A2H( ).

  L'output della funzione è una tupla p contenente in ordine:

  - H, tupla contenente la forma 'ba' o la forma 'zpk' della funzione del Loop Filter $H(f)$ determinata a partire dalla generica $A(f)$;

- K, numero reale che rappresenta il guadagno del Loop Filter $H(f)$, indicato nei capitoli precedenti con K_{LP};
- param contiene i parametri del denominatore del modello del Loop Filter e può essere:
  - None se N=1;
  - un numero reale coincidente con la frequenza naturale del polo reale se N=2;
  - una semplice tupla di due numeri reali corrispondenti alla frequenza naturale della coppia di poli complessi coniugati e al relativo fattore di merito se N=3;
  - un array di tuple di due elementi coincidenti con la frequenza naturale e il fattore di merito di coppie di poli complessi coniugati o numeri reali corrispondenti alla frequenza naturale dei poli reali nel caso di N>3.
- paramz contiene i parametri del numeratore del modello del Loop Filter e può essere:
  - None se N=1 o per i prototipi Butterworth, Bessel e Chebyshev I;
  - un numero reale coincidente con la frequenza naturale della coppia di zeri complessi coniugati a parte reale nulla se N=2 o N=3 per prototipi di Chebyshev II ed ellittici;
  - un array di numeri reali corrispondenti alle frequenze naturali delle coppie di zeri complessi coniugati a parte reale nulla per N>3 per prototipi di Chebyshev II ed ellittici.
- fz, numero reale che indica la frequenza dello zero aggiuntivo per PLL di tipo II, nel caso di PLL di tipo I è None;
- wk, numero reale di supporto per il calcolo del guadagno del Loop Filter;
- wcp numero reale corrispondente alla ω_{cp} ovvero la pulsazione naturale del polo introdotto dallo stadio di integrazione aggiuntivo per PLL di tipo II.

- H = **MODELtoH** (N, typeLF, K, param, paramz, fz, K1=1, output='ba')

  Questa funzione determina la funzione di trasferimento del Loop Filter nella forma ‘ba’ o ‘zpk’ a partire dai parametri della funzione di trasferimento stessa a seconda del tipo di PLL.

  Gli input della funzione sono coincidenti con le notazioni indicate per la funzione characterize_G( ).

  L’output della funzione è una tupla H corrispondente alla forma ‘ba’ o ‘zpk’ della funzione di trasferimento del Loop Filter $H(s)$.

- H = **MODELtoG** (N, typeLF, ftype, rp, K, param, paramz, fz, K1=1, output='zpk')

  Questa funzione determina direttamente la funzione di trasferimento in catena chiusa $G(s)$ nella forma ‘ba’ o ‘zpk’ a partire dai parametri della funzione di trasferimento stessa a seconda del tipo di PLL e delle specifiche sul ripple in banda passante per filtri Chebyshev I o ellittici di ordine pari.

  Gli input della funzione sono coincidenti con le notazioni indicate per le funzioni characterize_G( ) e A2G( ).

  L’output della funzione è una tupla G corrispondente alla forma ‘ba’ o ‘zpk’ della funzione di trasferimento in catena chiusa $G(s)$.

- H = **addparis** (H, parispole, pariszero)

  Questa funzione consente di aggiungere gli effetti reattivi parassiti o più in generale poli e zeri di cui è nota la frequenza e il fattore di merito in caso di coppie di poli o zeri complessi coniugati ad una generica funzione di trasferimento.

  Gli input della funzione sono:

  - H, tupla contenente la forma ‘ba’ o la forma ‘zpk’ della generica funzione di trasferimento;
  - parispole è una lista che rappresenta i poli parassiti. Gli elementi della lista possono essere:

        - numero reali coincidente con la frequenza naturale dei poli parassiti reali;
        - tuple di due numeri reali corrispondenti alla frequenza naturala e al fattore di merito di coppie di poli parassiti complessi coniugati.
    - pariszero è una lista che rappresenta gli zeri parassiti. Gli elementi della lista possono essere:
        - numero reali coincidente con la frequenza naturale degli zeri parassiti reali;
        - tuple di due numeri reali corrispondenti alla frequenza naturala e al fattore di merito di coppie di zeri parassiti complessi coniugati.

  L'output della funzione è una tupla H corrispondente alla forma 'ba' o 'zpk' della funzione di trasferimento a cui sono stati aggiunti poli e zeri.

- p = **parisitic** (f0, N, ftype, typeLF, fz_f0, rp, rs, parispole = [], pariszero = [])

  Questa funzione consente di determinare i parametri caratteristici del modello delle funzioni di trasferimento del Loop Filter $H(s)$ tali da compensare la presenza di effetti reattivi parassiti assunti noti e introdotti nella funzione di trasferimento ad anello aperto. La compensazione è ottenuta mediante un algoritmo di ottimizzazione di Nelder-Mead e fa in modo che la posizione dei poli dominanti della $G(f)$ sia la stessa della $G(f)$ desiderata.

  Gli input della funzione sono:

    - f0, N, ftype, typeLF, fz_f0, rp e rs coincidenti con le notazioni indicate per le funzioni ideal_G( ) e characterize_G( );
    - parispole e pariszero coincidenti con le notazioni espresse per la funzione addparis( ).

  L'output della funzione è una tupla p contenente in ordine K, param, paramz e fz indicati per la funzione characterize_G( ).

- fig, ax = **PoleZeroDiagram** (G, Xmin=None, Xmax=None, Ymin=None, Ymax=None, legend=None, fig=None, ax=None, dim=100, col='b')

  Questa funzione consente di visualizzare su un grafico il diagramma dei poli e degli zeri nel piano s/2π di una generica funzione di trasferimento. I parametri opzionali consentono di stabilire i limiti del grafico, la dimensione e il colore dei marker, la label da utilizzare come leggenda e la possibilità di plottare il diagramma su grafici preesistenti.

  Gli input della funzione sono:

  - G, tupla contenente la forma 'ba' o la forma 'zpk' della generica funzione di trasferimento;
  - Xmin, Xmax, Ymin, Ymax, numeri reali che indicano i limiti delle ascisse e delle ordinate del grafico
  - legend, stringa utilizzata come leggenda, se None viene utilizzata quella di default
  - fig e ax, oggetti grafici sui quali si vuole aggiungere il diagramma dei poli e degli zeri, se None la funzione crea un nuovo grafico
  - dim, numero intero che esprime la dimensione del marker
  - col, stringa che indica il colore del marker

  Gli output della funzione sono:

  - fig e ax, oggetti grafici che contengono il diagramma dei poli e degli zeri della generica funzione di trasferimento.

- fig, ax = **StepResponse** (G, Xmin=None, Xmax=None, Ymin=None, Ymax=None, start=None, stop=None, Np=None, space=None, legend=None, fig=None, ax=None)

  Questa funzione consente di visualizzare su un grafico la risposta alla sollecitazione a gradino di una generica funzione di trasferimento. I parametri opzionali consentono di stabilire i limiti del grafico, la base dei tempi di valutazione, la label da utilizzare come leggenda e la possibilità di plottare il diagramma su grafici preesistenti.

Gli input della funzione sono:

- G, tupla contenente la forma ‘ba’ o la forma ‘zpk’ della generica funzione di trasferimento;
- Xmin, Xmax, Ymin, Ymax, numeri reali che indicano i limiti delle ascisse e delle ordinate del grafico;
- start e stop, numeri reali che indicano l’inizio e la fine del tempo di valutazione, se None valgono i valori di default;
- Np numero di campioni temporali di valutazione della risposta;
- space, stringa che indica il tipo di incremento temporale, ovvero ‘lin’ se lineare, ‘log’ se logaritmico, None per il tipo di incremento di default;
- legend, stringa utilizzata come leggenda, se None viene utilizzata quella di default;
- fig e ax, oggetti grafici sui quali si vuole aggiungere il diagramma dei poli e degli zeri, se None la funzione crea un nuovo grafico.

Gli output della funzione sono:

- fig e ax, oggetti grafici che contengono la risposta alla sollecitazione a gradino di una generica funzione di trasferimento.

- fig, ax = **TransferFunction** (G, f0, Xmin=None, Xmax=None, Ymin=None, Ymax=None, start=None, stop=None, Np=None, space=None, legend=None, fig=None, ax=None)

  Questa funzione consente di visualizzare su un grafico il modulo di una generica funzione di trasferimento. I parametri opzionali consentono di stabilire i limiti del grafico, la base delle frequenze di valutazione, la label da utilizzare come leggenda e la possibilità di plottare il diagramma su grafici preesistenti.

  Gli input e gli output sono analoghi a quelli espressi per la funzione StepResponse( ), con la differenza che la base è nel dominio delle frequenze e l’input f0 della frequenza di taglio asintotica utilizzata per definire i limiti dell’intervallo di valutazione delle frequenze di default.

- rms, fig, ax = **NoisePlot** (G, f0, fref, fout, det=None, vco=None, foff=None, sd=None, ntf=None, Xmin=None, Xmax=None, Ymin=None, Ymax=None, start=None, stop=None, Np=None, space=None, legend=True, fig=None, ax=None)

  Questa funzione consente di visualizzare su un grafico il rumore di fase in uscita dal PLL e di calcolare il valore quadratico medio del jitter. La funzione permette anche di valutare l'effetto singolo o la somma degli effetti delle varie sorgenti di rumore. I parametri opzionali consentono di stabilire i limiti del grafico, la base delle frequenze di offset di valutazione, un metodo per disattivare le label da utilizzare come leggenda nel caso di più chiamate della funzione stessa e la possibilità di plottare il diagramma su grafici preesistenti.

  Gli input della funzione sono:

  - G, tupla contenente la forma 'ba' o la forma 'zpk' della generica funzione di trasferimento;
  - f0, numero reale che indica la frequenza di taglio asintotica f_0 [Hz] di $G(s)$;
  - fref, numero reale che rappresenta la frequenza del segnale di riferimento f_{ref} [Hz];
  - fout, numero reale che rappresenta la frequenza di uscita desiderata f_{out} [Hz];
  - det, specifica del detection noise. Può essere:
    - un singolo numero reale che indica la densità spettrale di potenza L_{det} [dBc/Hz];
    - un array di due o tre elementi, dove det[0] coincide con L_{det} [dBc/Hz]e det[1] con la corner frequency del detection noise f_{corn_dn} [Hz] e det[2] con la specifica della pendenza del flicker noise $slope_{fn\_dn}$ [dBc].
  - vco, specifica del VCO noise. Può essere:
    - un singolo numero reale che indica la densità spettrale di potenza L_{VCO} [dBc/Hz];

        - un array di due o tre elementi, dove vco[0] coincide con L_{det} [dBc/Hz]e vco[1] con la corner frequency del VCO noise f_{corn_VCO} [Hz] e vco[2] con la specifica della pendenza del flicker noise $slope_{fn\_VCO}$ [dBc].

    - foff, numero reale che rappresenta la frequenza di offset f_{off_VCO} [Hz] alla quale la specifica L_{VCO} è stata determinata;

    - sd, numero intero positivo che indica l'ordine $m_{\Sigma\Delta}$ del modulatore Σ-Δ con architettura MASH;

    - ntf, tupla contenente la forma "ba" della funzione di trasferimento del rumore $NTF(z)$ di un generico modulatore Σ-Δ;

    - Xmin, Xmax, Ymin, Ymax, start, stop, step, Np, space, fig e ax sono analoghi a quelli espressi per la funzione StepResponse( ), con la differenza che Xmin e Xmax vengono utilizzati anche come estremi di integrazione per il calcolo del valore quadratico medio del jitter.

    - legend, variabile opzionale booleana che se True abilita le label di default, altrimenti le disattiva. Questa variabile è utile per evitare di riproporre ogni volta le medesime label quando la funzione è richiamata all'interno di un ciclo;

  Gli output della funzione sono:

    - fig e ax, oggetti grafici che contengono il rumore di fase in uscita dal PLL;

    - rms, valore quadratico medio del jitter.

- fig, ax, rms = **plotG** (G, f0, ResPlot, fref=None, fout=None, det=None, vco=None, foff=None, sd=None, ntf=None, Xmin=None, Xmax=None, Ymin=None, Ymax=None, start=None, stop=None, Np=None, space=None, legend=None, fig=None, ax=None, dim=100, col='b')

  Questa funzione riassuntiva richiama le funzioni NoisePlot( ), PoleZeroDiagram( ), StepResponse( ), TransferFunction( ) a seconda del grafico desiderato.

Gli input e gli output hanno le caratteristiche espresse per le funzioni suddette, con la differenza di ResPlot che è una stringa utile per indicare quale grafico rappresentare e può essere 'PoleZeroDiagram', 'StepResponse', 'TransferFunction' o 'NoisePlot'.

- fig, ax, rmss = **variationpar** (N, f0, typeLF, ftype, rp, K, param, paramz, fz, ResPlot, K1=1, Kalter=numpy.array([1]), fpalter= numpy.array([1]), Qpalter= numpy.array([1]), fz0alter= numpy.array([1]), fzalter= numpy.array([1]), fref=None, fout=None, det=None, vco=None, foff=None, sd=None, ntf=None, Xmin=None, Xmax=None, Ymin=None, Ymax=None, start=None, stop=None, Np=None, space=None, parispole=[], pariszero=[], fig=None)

  Questa funzione consente di visualizzare su un grafico gli effetti dalla variazione dei parametri della funzione di trasferimento del Loop Filter $H(f)$ ed equivalentemente della $A(f)$ sulla posizione dei poli e degli zeri della funzione di trasferimento ad anello chiuso $G(f)$, sulla risposta al gradino, sul modulo della funzione di trasferimento e sul rumore di fase in uscita, anche in presenza di effetti reattivi parassiti. La funzione permette di specificare l'array di variazione percentuale per ogni parametro del Loop Filter.

  Gli input della funzione sono:

  - N, f0, typeLF, ftype, rp, K, param, paramz, fz, K1 sono coincidenti con le notazioni indicate per le funzioni ideal_G( ) e characterize_G( );
  - Resplot, stringa che indica quale grafico rappresentare. Può essere 'PoleZeroDiagram', 'StepResponse', 'TransferFunction' o 'NoisePlot';
  - Kalter, fzalter sono array che contengono l'indicazione delle variazioni percentuali da valutare rispettivamente per il guadagno del Loop Filter e per la frequenza dello zero aggiuntivo per PLL di tipo II. Se ad esempio l'elemento dell'array è 1 la variazione percentuale è dello 0%, se 0.8 la variazione percentuale è del -20%;
  - fpalter, Qpalter e fz0alter sono liste di tuple di due elementi che contiene l'indicazione delle variazioni percentuali da valutare rispettivamente della frequenza dei poli reali o di coppie di poli complessi coniugati, dei fattori di merito delle coppie di poli complessi coniugati e della frequenza delle coppie

di zeri complessi coniugati a parte reale nulla del Loop Filter. Il primo elemento della tupla è l'array che contiene i valori corrispondenti alle variazioni percentuali, il secondo elemento è l'indice del polo o dello zero del quale si vuole valutare l'effetto della variazione.

- fref, fout, det, vco, foff, sd, ntf sono coincidenti con le notazioni indicate per la funzione NoisePlot( );
- Xmin, Xmax, Ymin, Ymax, start, stop, Np, space e fig sono coincidenti con le notazioni indicate per la funzione plotG( );
- parispole e pariszero sono coincidenti con le notazioni indicate per la funzione parisitic( ).

Gli output della funzione sono:

- fig e ax, oggetti grafici che contengono il grafico selezionato da ResPlot;
- rmss, lista contenente le valutazioni del valore quadratici medi del jitter per ogni differente configurazione dei parametri fp, Qp, fz, fz0 e K.

Capitolo 8.2: Esempi di utilizzo programmatico delle funzioni Python dell'assistente al progetto di PLL

Calcolo dei parametri della funzione di trasferimento ad anello aperto A(f)

Per calcolare i parametri della funzione di trasferimento ad anello aperto e valutare graficamente la posizione dei poli e degli zeri della funzione di trasferimento ad anello chiuso $G(f)$, la risposta al gradino unitario e il modulo della $G(f)$, come ottenuto mediante l'interfaccia grafica in Figura 72, Figura 73 e Figura 74, è possibile richiamare le funzioni Python descritte nel Capitolo 8.1.

In particolare:

1. Si determina la funzione di trasferimento ad anello chiuso desiderata tramite la funzione ideal_G( ) con N=3, f0=300e3, ftype='butter', rp=None, rs=None.

2. Si calcolano i parametri della funzione di trasferimento ad anello aperto mediante la funzione characterize_G( ) con G coincidente con l'output del passo 1, N=3, typeLF=2, ftype='butter', f0=300e3, fz_f0=0.125, rp=None.

3. Si determina la funzione di trasferimento ad anello chiuso ottenuta con la configurazione dei parametri di $A(s)$ mediante la funzione MODELtoG( ) con N=3, typeLF=2, ftype='butter', rp=None e K, param, paramz e fz coincidenti con l'output al passo 2.

4. Si istanzia una figura vuota delle dimensioni desiderate tramite la funzione figure( ) della libreria Matplotlib.

5. Si richiama la funzione plotG( ) con G coincidente con l'output al passo 3 e ResPlot pari a 'PoleZeroDiagram' per ottenere la Figura 72, 'StepResponse' per la Figura 73, 'PoleZeroDiagram' per la Figura 74.

Inclusione di poli e zeri parassiti nella funzione di trasferimento ad anello aperto A(f) e ricalcolo delle variabili di controllo con l'algoritmo di compensazione

Per valutare l'efficacia dell'algoritmo di compensazione dei poli e degli zeri parassiti, confrontando il caso privo di parassiti, il caso con parassiti senza adottare l'algoritmo di compensazione e il caso in cui l'algoritmo viene adottato, ottenendo la Figura 80, la Figura 81, la Figura 82 e la Figura 83, è possibile sfruttare le funzioni Python implementate nel seguente modo:

1. Si determina la funzione di trasferimento ad anello chiuso nel caso privo di parassiti richiamando in successione le funzioni ideal_G( ), characterize_G( ) e MODELtoG( ) con N=3, f0=300e3, ftype='butter', rp=None, rs=None, typeLF=2, fz_f0=0.125.

2. Si determina la funzione di trasferimento ad anello chiuso nel caso con effetti reattivi parassiti senza adottare l'algoritmo di compensazione, utilizzando i parametri determinati dalla funzione characterize_G( ) del punto 1 e richiamando in successione MODELtoH( ), addparis( ), H2A( ), A2G( ) con parispole = [1.5e6, (3.5e6, 3.5)] e pariszero = [8e6].

3. Si determina la funzione di trasferimento ad anello chiuso nel caso in cui l'algoritmo di compensazione viene adottato, richiamando in successione le funzioni ideal_G( ), parisitic( ), MODELtoH( ), addparis( ), H2A( ), A2G( ).

4. Si istanzia una figura vuota delle dimensioni desiderate tramite la funzione figure( ) della libreria Matplotlib.

5. Si richiama tre volte la funzione plotG( ), con G ottenuta nei tre casi determinati nei punti 1, 2 e 3, legend differente per distinguere i tre grafici e ResPlot pari a 'PoleZeroDiagram' per ottenere la Figura 80, 'StepResponse' per la Figura 82, 'PoleZeroDiagram' per la Figura 83. Per la Figura 81 è necessario cambiare i limiti del grafico con i parametri opzionali Xmin Xmax Ymin e Ymax rispettivamente pari a -0.4e6, -0.02e6, -0.3e6, 0.3e6.

Analisi dettagliata del rumore di fase e inclusione del flicker noise

Per valutare il rumore di fase in uscita dal PLL dovuto alle differenti sorgenti di rumore e ottenere i grafici rappresentati in Figura 84, Figura 86, Figura 87, Figura 88 e Figura 89, è sufficiente richiamare le funzioni nel seguente ordine:

1. Si determina la funzione di trasferimento ad anello chiuso desiderata tramite la funzione ideal_G( ) con N=3, f0=300e3, ftype='butter', rp=None, rs=40.

2. Si calcolano i parametri della funzione di trasferimento ad anello aperto mediante la funzione characterize_G( ) con G coincidente con l'output del passo 1, N=3, typeLF=2, ftype='butter', f0=300e3, fz_f0=0.125, rp=None.

3. Si determina la funzione di trasferimento ad anello chiuso ottenuta con la configurazione dei parametri di $A(s)$ mediante la funzione MODELtoG( ) con N=3, typeLF=2, ftype='butter', rp=None e K, param, paramz e fz coincidenti con l'output al passo 2.

4. Si istanzia una figura vuota delle dimensioni desiderate tramite la funzione figure( ) della libreria Matplotlib.

5. Si richiama la funzione plotG( ) con G coincidente con l'output al passo 3, ResPlot pari a 'NoisePlot', fref=20e6, fout=1.84e9, ntf = None e

a. det = -76, vco = -140, foff = 5e6, sd = 3, start =Xmin = 10e3, stop = Xmax = 100e6, Ymin = -160 e Ymax = -60 per ottenere la Figura 84;

b. det[0] = -90, det[1] = 1e3, vco = -140, foff= 5e6, sd = 3, start = Xmin = 10, stop = Xmax = 100e6, Ymin = -160 e Ymax = -60, Np = 1000, space='log' per ottenere la Figura 86;

c. stessi parametri del punto 5.b con det[2] = -15 per ottenere la Figura 87;

d. det = -90, vco[0] = -140, vco[1]=1e3, foff= 5e6, sd = 3, start = Xmin = 10, stop = Xmax = 100e6, Ymin = -160 e Ymax = -60, Np = 1000, space='log' per ottenere la Figura 88;

e. stessi parametri del punto 5.d con vco[2] = -35 per ottenere la Figura 89.

Valutazione dell'effetto dell'inclusione di poli e zeri parassiti nella funzione di trasferimento ad anello aperto A(f) sul rumore di fase

Per valutare il rumore di fase in uscita nel caso in cui ci siano effetti reattivi parassiti nella funzione di trasferimento ad anello aperto $A(f)$ e ottenere i grafici rappresentati in Figura 91, Figura 92 e Figura 95, si possono sfruttare le funzioni Python descritte nel capitolo 8.1 nel seguente ordine:

1. Si calcolano i parametri della funzione di trasferimento ad anello aperto mediante la funzione parisitic( ) con f0=300e3, N=3, ftype='butter', typeLF=2, fz_f0=0.125, rp = None, rs = None e

   a. parispole = [2e6, (3.5e6,3.5)], pariszero =[8e6] per la Figura 91;

   b. parispole = [1e6], pariszero = [] per la Figura 92 e Figura 95.

2. Si determina la funzione di trasferimento del Loop Filter $H(s)$ mediante la funzione MODELtoH( ) con K, param, paramz e fz determinati dalla funzione parisitic( ) al passo 1.

3. Si aggiungono i parassiti alla funzione di trasferimento $H(s)$ mediante la funzione addparis( ) con H determinato al passo 2 da MODELtoH( ) e parispole e pariszero indicati al punto 1.a. e 1.b.

4. Si determinano la funzione di trasferimento ad anello aperto mediante la funzione H2A( ) e quella ad anello chiuso tramite la funzione A2G( ).

5. Si istanzia una figura vuota delle dimensioni desiderate tramite la funzione figure( ) della libreria Matplotlib.

6. Si richiama la funzione plotG( ) con G coincidente con l'output al passo 4, ResPlot pari a 'NoisePlot', fref=20e6, fout=1.84e9, det= -76, vco= -140, foff=5e6, start= Xmin=10, stop=Xmax =100e6, Ymin= -160 e Ymax= -60, Np=1000, space='log' e

    a. sd=3, ntf=None per ottenere la Figura 91 e Figura 92;

    b. sd=None e ntf=(numpy.array([1, -3, 3, -1]), numpy.array([1])) per ottenere la Figura 95.

Valutazione dell'impatto della variazione dei parametri della funzione di trasferimento ad anello aperto A(f) sul rumore di fase e sulle caratteristiche dinamiche ad anello chiuso

Per valutare l'effetto della variazione dei parametri della funzione di trasferimento ad anello aperto $A(f)$ sulle caratteristiche dinamiche ad anello chiuso e sul rumore di fase in uscita dal PLL e ottenere ad esempio i grafici di Figura 75, Figura 76, Figura 77 e Figura 93, si possono sfruttare le funzioni Python implementate nel seguente modo:

1. Se non ci sono poli e zeri parassiti come per la Figura 75, la Figura 76 e la Figura 77, si determina la funzione di trasferimento ad anello chiuso desiderata tramite la funzione ideal_G( ) con N=3, f0=300e3, ftype='butter', rp=None, rs=None e si calcolano i parametri della funzione di trasferimento ad anello aperto mediante la funzione characterize_G( ) con G coincidente con l'output del passo 1, N=3, typeLF=2, ftype='butter', f0=300e3, fz_f0=0.125, rp=None. Altrimenti se ci sono poli o zeri parassiti come per la Figura 93 si calcolano direttamente i parametri con la funzione parisitic( ) con f0=300e3, N=3, ftype='butter', typeLF=2, fz_f0=0.125, rp = None, rs = None, parispole = [1e3], pariszero = [].

2. Si istanzia una figura vuota delle dimensioni desiderate tramite la funzione figure( ) della libreria Matplotlib.

3. Si richiama la funzione variationpar( ) con i parametri K, param, paramz e fz determinati al punto 1, Kalter = numpy.array([1]), fpalter = [(numpy.array([0.8, 1, 1.2]), 0)], fz0alter = [(numpy.array([1]), 0)], Qpalter = [(numpy.array([1]), 0)], fzalter = numpy.array([1]) e

    a. ResPlot = ‘PoleZeroDiagram’ per la Figura 75;

    b. ResPlot = ‘StepResponse’ per la Figura 76;

    c. ResPlot = ‘TransferFunction’ per la Figura 77;

    d. ResPlot = ‘NoisePlot, fref=20e6, fout=1.84e9, det = -76, vco = -140, foff = 5e6, start = Xmin = 10, stop = Xmax = 100e6, Ymin = -160 e Ymax = -60, Np = 1000, space=’log’, sd = 3, ntf = None, parispole =[1e3], pariszero = [] per la Figura 93.

Conclusioni

In questo lavoro di tesi, si è affrontato il problema di realizzare un software di assistenza al progetto di sintetizzatori di frequenza basati su PLL a rapporto intero o frazionario, attraverso l'uso di un ambiente di programmazione basato sul linguaggio Python e alcune sue estensioni per applicazioni numeriche e scientifiche.

Punti di partenza per il lavoro di tesi sono stati:

i. una serie di articoli scientifici in buona parte prodotti dal gruppo di lavoro del Prof. M. H. Perrott tra il 2002 e il 2010 al Massachusetts Institute of Technology, Cambridge, USA;

ii. un applicativo di assistenza al progetto di PLL a "codice chiuso", operabile esclusivamente attraverso un'interfaccia grafica e su un unico sistema operativo, distribuito gratuitamente dallo stesso gruppo di ricerca, come parte di CppSim, un sofisticato simulatore "a livello di sistema".

Obiettivo di minimo del lavoro era quello di replicare le funzionalità del PLL Design Assistant sopra citato, creando un codice a sorgente aperto e multipiattaforma.

Si voleva inoltre svincolare le funzionalità di assistenza al progetto dalla necessità di un controllo attraverso un'interfaccia grafica, strutturando il software in due strati ben definiti:

1. uno inferiore, in cui tutte le funzionalità sono messe a disposizione per via "programmatica" attraverso un'Application Programming Interface ben definita;

2. uno superiore dedicato unicamente a riproporre tutte le funzionalità in una forma più adatta a un operatore umano.

La motivazione del lavoro era quella di mettere a disposizione un codice di assistenza al progetto di PLL che fosse "aperto" e adatto quindi a essere impiegato come base per testare nuovi approcci o per introdurre nuove forme di assistenza al progettista. Inoltre, si voleva avere un'interfaccia programmatica, in modo che il codice potesse venire impiegato non solo come strumento a sé stante, ma anche come libreria, incorporabile in progetti più ampi. A tale riguardo, è opportuno ricordare che il gruppo di ricerca all'interno del quale è stato svolto il

lavoro di tesi è impegnato nello sviluppo di un *tool* per il progetto di modulatori Σ-Δ, detto PyDSM.

I PLL a rapporto frazionario, incorporano generalmente al loro interno un modulatore di questo tipo e un tema di ricerca assai interessante che si è concretizzato ultimamente è quello della progettazione congiunta del PLL e del modulatore Σ-Δ posto al suo interno. Uno sviluppo ovvio del codice prodotto con il lavoro di tesi è quindi proprio la sua incorporazione in PyDSM, affinché possa fornire specifiche per tale progettazione congiunta. La scelta di Python come linguaggio di sviluppo è stata proprio motivata dall'esigenza di favorire l'integrazione con PyDSM che è basato su un mix di Python e C.

Nell'ambito del lavoro di tesi, tutti gli obiettivi di minimo sono stati raggiunti. A tale proposito è necessario sottolineare che non si è trattato solo di compiere una "riscrittura" in un diverso linguaggio e stile di programmazione di un prodotto esistente, visto che di questo non esistevano i sorgenti. Né si è trattato semplicemente di codificare algoritmi già pienamente formalizzati in articoli scientifici, visto che la formalizzazione era a volte vaga e che si è voluta mantenere la compatibilità con un "prototipo" che a volte deviava nei comportamenti da quanto descritto negli articoli teorici. Nella pratica, è stato necessario acquisire una competenza approfondita dei metodi elaborati e descritti dal gruppo del Prof. Perrott ed, episodicamente, affiancare a tale competenza un certo grado di *reverse engineering* del campione di software disponibile.

Si è poi provveduto ad andare oltre agli obiettivi di minimo, sperimentando alcune estensioni alle funzionalità del codice di base. Ad esempio, attraverso una rivisitazione dei modelli matematici è stato possibile ampliare le funzionalità del codice, così da gestire in maniera corretta PLL con funzioni di trasferimento comprendenti zeri o di ordine elevato. Infatti, il codice a sorgente chiuso usato come benchmark e gli articoli scientifici formalizzano metodi validi solo fino al 3° ordine. Inoltre, il codice di benchmark non gestisce correttamente forme di Cauer o di Chebyshev di tipo II che incorporano zeri.

Queste estensioni sono significative. Ad esempio, dalle analisi svolte, si osserva che scegliere caratteristiche dinamiche conformi all'approssimazione di Chebyshev di tipo II o di Cauer congiuntamente ad architetture PLL di tipo frazionario offre una minore attenuazione del rumore di quantizzazione introdotto dal modulatore Σ-Δ, ma al tempo stesso una maggiore attenuazione della componente dominante del rumore a basse frequenze di offset rispetto frequenza del segnale portante, determinando quindi un minore jitter in uscita.

Inoltre, dalle analisi svolte risulta anche che, nel caso si volesse minimizzare il rumore di fase a frequenze oltre la banda del sistema, è preferibile fare in modo che il sistema non abbia zeri dominanti, ovvero scegliere prototipi Butterworth, Bessel o Chebyshev I. Con questa scelta è infatti possibile fare in modo che il rumore di quantizzazione, il quale generalmente è la componente di rumore maggioritaria nei PLL a rapporto frazionario, abbia un effetto minore, o comunque inferiore in bande ristette, rispetto ai contributi sul rumore di fase delle altre sorgenti di rumore del sistema.

Non sarebbe stato possibile arrivare a una valutazione quantitativa di tali aspetti, senza un codice in grado di gestire correttamente anche PLL con funzioni di trasferimento comprendenti zeri e potenzialmente di ordine elevato.

Tra i problemi che rimangono aperti, vi è quello di completare l'integrazione del codice sviluppato nella tesi con PyDSM. Quest'ultimo tool, sviluppato all'Università di Bologna e al Politecnico di Milano, utilizza tecniche formali di ottimizzazione convessa (e in particolare di programmazione semidefinita) per progettare modulatori Σ-Δ le cui caratteristiche di noise shaping corrispondano a diversi criteri di ottimalità. L'integrazione dovrà rivolgersi al progetto di PLL a rapporto frazionario che impiegano modulatori Σ-Δ al loro interno. In particolare, il codice di supporto al progetto di PLL dovrà essere in grado di fornire una descrizione del PLL che possa essere usata come criterio di ottimalità all'interno di PyDSM, affinché gli strumenti contenuti in quest'ultima suite possano definire il modulatore Σ-Δ più adatto all'applicazione nel PLL. Inoltre, dovrà consentire di acquisire l'output di PyDSM per valutare l'effettivo comportamento del modulatore Σ-Δ all'interno del PLL.

Bibliografia

[1] C. Y. Lau and M. H. Perrott, "Fractional-N Frequency Synthesizer Design at the Transfer Function Level Using a Direct Closed Loop Realization Algorithm," *DAC,* pp. 526-531, 2003.

[2] M. H. Perrott, M. D. Trott and C. G. Sodini, "A Modeling Approach for Σ-Δ Fractional-N Frequency Synthesizer Allowing Straightforward Noise Analysis," *IEEE JOURNAL OF SOLID-STATE CIRCUITS,* vol. 37, no. 8, pp. 1028-1038, 2002.

[3] M. H. Perrott, "PLL Design Using the PLL Design Assistant Program," pp. 1-33, 2005.

[4] G. v. Rossum, "Python tutorial, Technical Report CS-R9526," Centrum voor Wiskunde en Informatica (CWI), Amsterdam, 1995.

[5] "Python Language Reference, version 3.5," Python Software Foundation, [Online]. Available: http://www.python.org.

[6] "numpy website," [Online]. Available: http://www.numpy.org.

[7] P. F. Dubois, K. Hinsen and J. Hugunin, "Numerical Python," *Computers in Physics,* vol. X, no. 3, 1996.

[8] T. E. Oliphant, "Guide to NumPy," Brigham Young University, March 2006. [Online].

[9] "scipy website," [Online]. Available: http://www.scipy.org.

[10] T. O. P. P. a. o. Eric Jones, "SciPy: Open source scientific tools for," 2001. [Online]. Available: http://www.scipy.org/.

[11] F. B. A. B. Sergio Callegari, "Optimal Quantization Noise Management in Wideband Fractional-N PLLs," 2015.

[12] "PyDSM 0.13.0.1 documentation," [Online]. Available: http://pythonhosted.org/pydsm/.

[13] "Python Based Delta-Sigma modulator design tools," [Online]. Available: https://pypi.python.org/pypi/pydsm.

[14] S. Callegari and F. Bizzarri, "Teaching ΔΣ modulators with PyDSM and scientific Python," *2015 IEEE International Symposium on Circuits and Systems (ISCAS),* pp. 1802-1805, 2015.

[15] A. Antoniou, Digital Filters: Analysis, Design, and Signal Processing Applications, Grow Hill, 2017.

[16] M. H. Perrott, Short course on Phase Locked Loops, San Diego, 2009.

[17] "cppsim website," [Online]. Available: http://www.cppsim.com.

[18] "pyzo website," [Online]. Available: http://www.pyzo.org/python_vs_matlab.html.

[19] "matplotlib website," [Online]. Available: https://matplotlib.org/.

[20] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering Vol. 9, N. 90,* 2007.

[21] D. Phillip and M. Feldman, "phillipmfeldman.org," 26 August 2017. [Online]. Available: http://phillipmfeldman.org/Python/Advantages_of_Python_Over_Matlab.html.

[22] "riverbank computing website," Riverbank Computing Limited, [Online]. Available: https://www.riverbankcomputing.com/software/pyqt/intro.