UNIVERSITA POLITECNICA DELLE MARCHE

FACOLTA DI INGEGNERIA

Corso di laurea triennale in Ingegneria Elettronica

Studio di un simulatore circuitale basato su Wave Digital Filters (WDF) e implementazione di semplici circuiti analogici.

Study of a circuit simulator based on Wave Digital Filters (WDF) and implementation of simple analog circuits.

Tesi di Laurea di:
CRISTIANO BRUNI

Relatore: Chiar.mo
Prof. **STEFANO SQUARTINI**

Correlatore:
Ing. **LEONARDO GABRIELLI**

Anno Accademico 2023/2024

ABSTRACT

La presente tesi si propone di studiare e testare il funzionamento del metodo Wave Digital Filter (WDF), impiegato per la risoluzione in tempo reale e la simulazione di circuiti elettrici e implementato nel framework “pywdf” in Python, disponibile su GitHub e sviluppato da “Gustav Anthon”, “Xavier Lizarraga-Seijas” e “Frederic Font”, attraverso la simulazione digitale di semplici circuiti analogici.

I circuiti saranno rappresentati inizialmente mediante uno schema elettrico, successivamente con un grafo ad albero utilizzato nel metodo WDF per digitalizzare il circuito tramite software, e infine verranno illustrati i segnali in ingresso e in uscita dai circuiti esaminati attraverso grafici, al fine di valutare le prestazioni del framework in termini di precisione e tempi di esecuzione.

Le prestazioni del framework verranno confrontate con quelle ottenute utilizzando simulatori circuitali come SPICE, noto per la sua precisione e qualità, ma caratterizzato da un elevato costo computazionale e quindi, da lunghi tempi di risoluzione dei circuiti.

Indice

Capitolo 1

Introduzione

Negli ultimi decenni, la simulazione dei circuiti elettrici è diventata una componente fondamentale nel processo di progettazione elettronica.

L'evoluzione dei metodi di simulazione e l'avvento di strumenti software sempre più avanzati hanno reso possibile l'analisi dettagliata del comportamento di circuiti complessi, riducendo i tempi di sviluppo e i costi associati alla prototipazione fisica.

Tra i vari metodi di simulazione, i filtri digitali a onde (Wave Digital Filters, WDF) rappresentano un approccio innovativo e particolarmente efficace per la modellazione di circuiti non lineari e con componenti distribuiti, Il metodo WDF si basa su una rappresentazione dei segnali come onde anziché tensioni e correnti, offrendo vantaggi significativi in termini di stabilità numerica, efficienza computazionale e modularità.

Questi aspetti lo rendono particolarmente adatto per la simulazione in tempo reale, trovando applicazioni nei sistemi audio, negli strumenti musicali digitali e nei circuiti elettronici complessi.

Il crescente interesse per i WDF ha portato allo sviluppo di diverse librerie software dedicate, tra cui pywdf, un progetto open source che implementa il metodo WDF in Python.

Python, grazie alla sua semplicità d'uso, all'ampia disponibilità di librerie scientifiche e alla sua natura flessibile, si presta come un linguaggio ideale per la simulazione e l'analisi di circuiti, consentendo lo sviluppo di strumenti personalizzati e facilmente integrabili in diversi contesti applicativi.

L'obiettivo di questa tesi è esplorare l'utilizzo del metodo WDF per la simulazione di circuiti elettrici implementato da pywdf in Python, analizzando le sue potenzialità, i suoi limiti e le sue applicazioni pratiche.

1.1 Stato dell'arte

La simulazione dei circuiti elettrici ha subito una rapida evoluzione grazie allo sviluppo di algoritmi numerici avanzati e all'incremento della potenza di calcolo disponibile.

I metodi tradizionali di simulazione, come l'analisi nodale modificata (Modified Nodal Analysis, MNA), sono stati ampiamente utilizzati per decenni e rimangono ancora oggi tra i più diffusi; Tuttavia, questi metodi possono manifestare significative limitazioni in presenza di componenti non lineari o di circuiti che richiedono simulazioni in tempo reale, come avviene nel campo dell'audio digitale e degli effetti per strumenti musicali.

In questo contesto, il metodo dei filtri d'onda digitali (WDF) è emerso come una soluzione particolarmente promettente.

Proposto inizialmente negli anni '70 per l'implementazione digitale di filtri analogici, il metodo WDF è stato successivamente esteso alla simulazione di circuiti elettrici complessi.

A differenza dei metodi tradizionali, i WDF rappresentano i segnali sotto forma di onde, suddividendo il circuito in elementi modulari che possono essere facilmente interconnessi, consentendo una gestione efficiente dei componenti anche non lineari e mantenendo al contempo una stabilità numerica che si rivela cruciale nelle simulazioni dinamiche.

Negli ultimi anni, l'interesse per i WDF è cresciuto significativamente, soprattutto grazie alle loro applicazioni in ambito audio in quanto permettono di modellare amplificatori, effetti sonori e strumenti musicali con un livello di dettaglio e realismo difficilmente raggiungibile con altri metodi.

La modularità del WDF consente di costruire simulazioni complesse a partire da componenti di base, facilitando la sperimentazione e l'ottimizzazione di nuove configurazioni circuitali, portando così allo sviluppo di un "Antiderivative Antialiasing (ADAA)" allo scopo di ridurre l'effetto aliasing, senza sovra campionare i segnali, per dispositivi non lineari [1], circuiti oscillatori con fattore di alta qualità per la progettazione di componenti analogici a segnale misto (AMS) [2].

Il software pywdf, disponibile su GitHub, rappresenta una delle implementazioni più accessibili del metodo WDF, scritto in python, pywdf sfrutta le potenzialità del linguaggio per offrire un ambiente di simulazione flessibile e facilmente estendibile.

L’integrazione con le librerie scientifiche di Python, come numpy e matplotlib, amplia ulteriormente le possibilità di analisi e ottimizzazione, rendendo pywdf uno strumento potente per la ricerca e lo sviluppo di nuove tecniche di simulazione.

La tesi si propone di analizzare le caratteristiche del metodo WDF, esplorare le funzionalità di pywdf e valutarne l’efficacia in diverse applicazioni pratiche, verranno discussi i principali vantaggi del metodo, i limiti incontrati durante l’implementazione e le potenziali aree di miglioramento, con l’obiettivo di contribuire allo sviluppo e alla diffusione di questa tecnologia innovativa.

Capitolo 2

Teoria dei Wave Digital Filter

2.1 Introduzione ai filtri digitali

I filtri digitali sono strumenti fondamentali nell'elaborazione del segnale, utilizzati per modificare o analizzare segnali digitali attraverso algoritmi matematici.

Questi filtri sono onnipresenti in una vasta gamma di applicazioni, tra cui l'elaborazione audio, le telecomunicazioni e l'elaborazione delle immagini.

La principale differenza rispetto ai filtri analogici risiede nella capacità dei filtri digitali di eseguire operazioni complesse con una precisione e una stabilità elevata, indipendentemente dalle variazioni ambientali come temperatura o invecchiamento dei componenti, inoltre permettono la realizzazione di funzioni di trasferimento a volte impossibili per i filtri analogici, risultando nonostante tutte queste qualità, più economici.

È possibile utilizzare questo tipo di filtro anche su segnali analogici, che devono essere dapprima campionati e poi convertiti in digitale; Una volta eseguito il filtraggio, si genera nuovamente il segnale analogico tramite una semplice interpolazione del segnale digitale in uscita.

Questa interpolazione, o conversione, viene eseguita da circuiti chiamati convertitori digitali-analogici (DAC, Digital to Analog Converter).

I filtri digitali si dividono in due categorie principali, Finite Impulse Response (FIR) e Infinite Impulse Response (IIR) [3], i FIR sono noti per la loro stabilità incondizionata, mentre gli IIR possono realizzare risposte più complesse utilizzando meno risorse computazionali, ma con il potenziale rischio di instabilità.

2.2 Fondamenti dei Wave Digital Filter

Il metodo dei Wave Digitl Filter (WDF) è stato sviluppato da Alfred Fettweis agli inizi degli anni 70 come modo per progettare speciali filtri digitali, ma con la stessa struttura di alcuni prototipi analogici.

I filtri digitali progettati secondo i principi dei Wave Digital Filter e sulla base di questi filtri di riferimento, utilizzano variabili d'onda come variabili di segnale e godono di eccellenti caratteristiche di tolleranza e insensibilità a troncamenti e arrotondamenti della risposta in frequenza [4].

Queste proprietà vengono dal fatto che i Wave Digital filter sono derivati dalla discretizzazione locale degli elementi del circuito e delle loro connessioni, un processo che, se implementato correttamente, garantisce che le proprietà energetiche dell'analogo circuito di riferimento vengano replicate.

Grazie a queste qualità, i WDF sono stati utilizzati inizialmente per la realizzazione di sistemi meccanici concentrati acustici, (ovvero un sistema in cui le proprietà fisiche come massa, rigidezza, forza, possono essere trattate come se fossero concentrate in punti specifici) in quanto tali sono descritti con equazioni differenziali nel tempo e non nello spazio.

In questo modo sono state trasformate le variabili di forza e velocità, come per la tensione e la corrente elettrica, in variabili d'onda ed utilizzate sempre nel campo audio, ma con applicazioni meccaniche, non attraverso dei circuiti, ma grazie a delle guide d'onda [5].

Un esempio di componenti realizzati con questo sistema sono il martello del pianoforte e l'ancia per strumenti a fiato.

L'utilizzo del metodo WDF, fino ai primi del 2000, è stato limitato ai componenti elettrici lineari e collegamenti in serie e parallelo, ciò a causa della complessità dell'implementazione di componenti non lineari.

Recentemente si è destato un notevole interesse per la modellazione di circuiti audio non lineari e si stanno espandendo nel metodo WDF.

2. 3 Trasformazione dei dispositivi, dalle leggi di Kirchhoff al dominio WDF

Come citato precedentemente, il metodo dei Wave Digital Filter si basa sulla trasformazione delle variabili elettriche di tensione e corrente, in qualcosa di completamente diverso, le variabili d'onda.

Lo scopo di questa trasformazione è di riuscire a semplificare le equazioni circuitali dei componenti elettrici, in particolare quelle differenziali dei componenti reattivi, e simularli digitalmente mantenendo la passività dei sistemi originali e prevenendo l'instabilità.

Le equazioni delle variabili d'onda vengono definite nella teoria di Scattering [4], da cui, attraverso l'utilizzo combinato con le leggi di Kirchhoff, otteniamo le equazioni delle onde a_0 incidenti e b_0 riflesse, come combinazione lineare di tensione v_0 e corrente i_0, mentre il parametro R_0, chiamato resistenza della porta, è un parametro arbitrario che verrà scelto allo scopo di garantire la stabilità del sistema:

$$\mathbf{a_0} = \mathbf{v_0} + \mathbf{R_0 i_0}$$

$$\mathbf{b_0} = \mathbf{v_0} - \mathbf{R_0 i_0} \tag{2.1}$$

Nonostante in questa tesi non le sfrutteremo, per la realizzazione del metodo WDF possono essere utilizzate onde di corrente o di potenza, ma noi introdurremo solo quelle di tensione, in quanto utilizzate dal simulatore circuitale che esamineremo nei capitoli successivi.

Considerando la (2.1) scriviamo le equazioni in forma matriciale ed otteniamo:

$$\begin{bmatrix} \mathbf{a_0} \\ \mathbf{b_0} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{R_0} \\ \mathbf{1} & \mathbf{-R_0} \end{bmatrix} \begin{bmatrix} \mathbf{v_0} \\ \mathbf{i_0} \end{bmatrix} \tag{2.2}$$

Dalla (2.2), considerando che una delle proprietà delle variabili d'onda è che sono invertibili, allora deve essere che $R_0 \neq 0$.

Definiamo tensione e corrente in funzione delle onde da (2.2):

$$\mathbf{v_0} = \frac{\mathbf{1}}{\mathbf{2}}\mathbf{a_0} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{b_0}$$

$$\mathbf{i_0} = \frac{\mathbf{1}}{\mathbf{2R_0}}\mathbf{a_0} - \frac{\mathbf{1}}{\mathbf{2R_0}}\mathbf{b_0} \qquad (2.3)$$

Una volta eliminata la dipendenza reciproca tra tensione e corrente grazie a (2.3), possiamo iniziare a definire alcuni dispositivi elettrici per far capire come avviene la trasformazione dei componenti.

Prendiamo un resistore, il suo comportamento viene definito dalla legge di Ohm, che mette in relazione la tensione della porta v_0 con la corrente i_0 attraverso la resistenza elettrica R del componente:

$$\mathbf{v_0} = \mathbf{R}\mathbf{i_0} \qquad (2.4)$$

Allo scopo di ottenere la trasformazione del dispositivo nel metodo WDF, andiamo a sostituire le variabili della (2.4) con la (2.3) ottenendo:

$$\frac{\mathbf{1}}{\mathbf{2}}\mathbf{a_0} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{b_0} = \frac{\mathbf{R}}{\mathbf{2R_0}}\mathbf{a_0} - \frac{\mathbf{R}}{\mathbf{2R_0}}\mathbf{b_0} \qquad (2.5)$$

Risolvendo per l'onda riflessa b_0 si ottiene l'equazione nel dominio d'onda a tempo continuo non adattata:

$$\mathbf{b_0} = \frac{\mathbf{R}-\mathbf{R_0}}{\mathbf{R}+\mathbf{R_0}}\mathbf{a_0} \qquad (2.6)$$

Dato che nella (2.4) non ci sono derivate, allora l'equazione risulta valida anche nel tempo discreto.
Allo scopo di ridurre le perdite e garantire la stabilità, risulta opportuno adattare la porta del dispositivo, per farlo andremo ad utilizzare una $R_0 = R$, in questo modo avremo che $b_0 = 0$ [5].

Il comportamento del resistore nel dominio WDF, con porta adattata, permette di eliminare le riflessioni dell'onda, in questo modo il dispositivo assorbe l'intera onda incidente senza rifletterla.

Un altro dispositivo che andremo ad esaminare è il generatore ideale di tensione.
Un generatore ideale di tensione produce una tensione di porta e(t):

$$\mathbf{v_0(t) = e(t)} \tag{2.7}$$

Sostituiamo con la (2.3) e otteniamo:

$$\mathbf{\frac{1}{2}a_0(t) + \frac{1}{2}b_0(t) = e(t)} \tag{2.8}$$

Risolvendo per trovare l'onda riflessa b_0 (t), si ottiene l'equazione del dominio d'onda in tempo continuo, non adattata:

$$\mathbf{b_0(t) = 2e(t) - a_0(t)} \tag{2.9}$$

La versione a tempo discreto è ottenuta sostituendo t con n nella (2.9) in quanto non presenta equazioni differenziali:

$$\mathbf{b_0(n) = 2e(n) - a_0(n)} \tag{2.10}$$

Possiamo notare che il generatore ideale di tensione, a differenza del resistore, ha una equazione d'onda che non dipende dalla resistenza della porta, pertanto non può essere adattato.

Con questi esempi, sono stati trattati solo dispositivi WDF senza memoria (algebrici), ciò significa che le equazioni nel dominio di Kirchhoff che collegano tensione, corrente e ingresso, sono istantanee, ossia non dipendono dal passato ma solo dal presente.

Questo si traduce, nel dominio delle onde, in una relazione lineare e istantanea tra l'onda incidente a, l'onda riflessa b, e un eventuale ingresso.

Adesso, verranno introdotti, tramite un esempio, gli elementi reattivi come il condensatore e l'induttore.

Gli elementi reattivi hanno, nelle loro equazioni costitutive, una derivata nel tempo continuo e per lavorare con essi nel tempo discreto e questa derivata deve essere discretizzata necessariamente per la simulazione numerica.

Il metodo più comune utilizzato dai WDF per approssimare le derivate temporali è la regola del trapezio, un metodo numerico per integrare funzioni, corrisponde all'applicare la Trasformata Bilineare, che mappa conformemente dal piano s (tempo continuo) nel piano di Laplace, al piano z (tempo discreto).

Analizziamo la trasformata del condensatore, la sua equazione costitutiva, che prevede l'utilizzo di una derivata, è:

$$\mathbf{i_0(t)} = \mathbf{C} \cdot \frac{\mathbf{dv(t)}}{\mathbf{dt}} \tag{2.11}$$

Prima di procedendo esattamente come nel caso del resistore, andiamo ad applicare la trasformata di Laplace per trasformare la sua equazione differenziale in una equazione algebrica, così da renderla più facile da elaborare:

$$\mathbf{I(s) = C\,(s\,V(s) - v(0))} \tag{2.12}$$

Dove v(0) rappresenta la tensione dipendente dalle condizioni iniziali, che nel nostro caso sono nulle, quindi la (2.12) diventa:

$$\mathbf{I(s) = C\,s\,V(s)} \tag{2.12}$$

Una volta semplificata la equazione differenziale possiamo proseguire sostituendo le variabili di tensione e corrente con la (2.3), ottenendo così:

$$\frac{\mathbf{1}}{\mathbf{2R_0}}\,\mathbf{A_0(s)} - \frac{\mathbf{1}}{\mathbf{2R_0}}\,\mathbf{B_0(s)} = \frac{\boldsymbol{C\,s}}{\mathbf{2}}\,\mathbf{A_0(s)} + \frac{\boldsymbol{C\,s}}{\mathbf{2}}\mathbf{B(s)} \tag{2.13}$$

Ora necessiteremo della funzione di trasferimento allo scopo di eseguire la trasformata bilineare della equazione, per trovarla basta cercare il rapporto tra l'onda incidente e l'onda riflessa come di seguito:

$$\mathbf{H_0(s)} = \frac{\mathbf{B_0(s)}}{\mathbf{A_0(s)}} = \frac{\mathbf{1 - R_0\,Cs}}{\mathbf{1 + R_0\,Cs}} \tag{2.24}$$

Per trasformare la funzione di trasferimento $H_0(s)$ in una forma discreta $H_0(z^{-1})$, è necessaria la Trasformata Bilineare, che realizza un mapping dal piano s (tempo continuo) al piano z (tempo discreto) preservando certe proprietà del sistema, come la stabilità e la frequenza di risposta.
Quindi dalla (2.24) otteniamo:

$$\mathbf{H_0(z^{-1}) = \frac{(T-2R_0C)+(T+2R_0C)z^{-1}}{(T+2R_0C)+(T-2R_0C)Z^{-1}} \ : \ s = f_{BLT}(z^{-1}) = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}} \qquad (2.25)$$

Ora andiamo ad anti trasformare dal dominio z e risolvere l'equazione (2.25) per ottenere l'onda riflessa dal condensatore con porta non adattata:

$$\mathbf{b_0[n] = -\frac{T-2R_0C}{T+2R_0c} \cdot b_0[n-1] + \frac{T-2R_0C}{T+2R_0C} \cdot a_0[n] + a_0[n-1]} \qquad (2.26)$$

Nella (2.26) notiamo una dipendenza dagli istanti precedenti dello stato n del condensatore, esattamente come dovrebbe essere.

La (2.26) può essere adattata impostando la $R_0 = \frac{T}{2C}$, in questo modo il contributo dell'onda incidente attuale e dell'onda riflessa precedentemente, viene annullato e rimane solo il contributo dell'onda in ingresso all'istante precedente:

$$\mathbf{b_0[n] = a_0[n-1]} \qquad (2.27)$$

La discretizzazione dei componenti reattivi può avvenire, non solo attraverso la trasformata bilineare, ma anche con la trasformata bilineare deformata, la trasformata α e la trasformata di Mobius.

Le prime due sono le più utilizzate, in quanto preservano ordine e stabilità, inoltre la trasformata bilineare risulta accurata fino al secondo ordine, mentre la deformata risulta più utile quando una risposta in frequenza ha una caratteristica importante, in quanto una specifica frequenza di interesse nel dominio continuo può essere mappata esattamente nel dominio discreto, utile per picchi o notch che devono essere preservati [5].

Un po' più complessi dei dispositivi appena esaminato, sono i dispositivi due-porte, ma sono solamente algebrici, quindi non abbiamo problemi di discretizzazione.

Ciò significa che basterà applicare le leggi di Kirchhoff alle matrici, per poi risolvere utilizzando i parametri di Scattering per ogni due-porte.

Legge di Kirchhoff applicata alle matrici:

$$\begin{bmatrix} \mathbf{x_{00}} & \mathbf{x_{01}} \\ \mathbf{x_{10}} & \mathbf{x_{11}} \end{bmatrix} \begin{bmatrix} \mathbf{v_0} \\ \mathbf{v_1} \end{bmatrix} + \begin{bmatrix} \mathbf{y_{00}} & \mathbf{y_{01}} \\ \mathbf{y_{10}} & \mathbf{y_{11}} \end{bmatrix} \begin{bmatrix} \mathbf{i_0} \\ \mathbf{i_1} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{2.28}$$

Oppure in forma vettoriale:

$$\mathbf{Xv + Yi = 0} \tag{2.29}$$

Esattamente come nel caso dei dispositivi non lineari, andiamo a sostituire con le equazioni della (2.3) ed otteniamo:

$$\mathbf{X}\left(\frac{\mathbf{1}}{\mathbf{2}}\mathbf{a} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{b}\right) + \mathbf{Y}\left(\frac{\mathbf{1}}{\mathbf{2R}}\mathbf{a} - \frac{\mathbf{1}}{\mathbf{2R}}\mathbf{b}\right) = \mathbf{0} \tag{2.30}$$

Risolvo per ottenere l'onda b non adattata:

$$\mathbf{b} = -\frac{\mathbf{RX+Y}}{\mathbf{RX+Y}}\mathbf{a} = \mathbf{Sa} \tag{2.31}$$

Dove il parametro S rappresenta la matrice di scattering i cui elementi descrivono la capacità di riflettere e trasmettere le onde dei componenti elettrici (o più in generale dei materiali).

Dopo questa introduzione degli elementi due-porte andremo a vedere la realizzazione di un adattatore serie come esempio.

Un adattatore serie è un dispositivo in grado di collegare due componenti seguendo le leggi elettriche relative alla serie tra elementi circuitali, cioè:

$$\mathbf{v_0} = -\mathbf{v_1} \tag{2.32}$$

$$\mathbf{i_0} = -\mathbf{i_1}$$

Utilizzando la definizione dei parametri d'onda (2.3) nelle leggi di Kirchhoff per gli elementi in serie, otteniamo:

$$\frac{\mathbf{a_0}}{\mathbf{2}} + \frac{\mathbf{b_0}}{\mathbf{2}} = -\frac{\mathbf{a_1}}{\mathbf{2}} - \frac{\mathbf{b_1}}{\mathbf{2}} \tag{2.33}$$

$$\frac{\mathbf{a_0}}{\mathbf{2R_0}} - \frac{\mathbf{b_0}}{\mathbf{2R_0}} = \frac{\mathbf{a_1}}{\mathbf{2R_1}} - \frac{\mathbf{b_1}}{\mathbf{2R_1}} \tag{2.34}$$

Risolviamo la (2.34) per trovare b_1 e la (2.33) per trovare b_0, ottenendo la matrice delle onde riflesse $[b_0 \quad b_1]^T$ non adattata:

$$\begin{bmatrix} \mathbf{b_0} \\ \mathbf{b_1} \end{bmatrix} = \begin{bmatrix} -\frac{\mathbf{R_0 - R_1}}{\mathbf{R_0 + R_1}} & -\frac{\mathbf{2R_0}}{\mathbf{R_0 + R_1}} \\ -\frac{\mathbf{2R_1}}{\mathbf{R_0 + R_1}} & \frac{\mathbf{R_0 - R_1}}{\mathbf{R_0 + R_1}} \end{bmatrix} \begin{bmatrix} \mathbf{a_0} \\ \mathbf{a_1} \end{bmatrix} \tag{2.35}$$

Per adattare la porta 0 dell'adattatore serie (la porta d'ingresso), dobbiamo fare in modo che il coefficiente di riflessione alla prima porta $s_{11} = -\frac{R_0 - R_1}{R_0 + R_1}$ e della seconda porta $s_{22} = -\frac{R_0 - R_1}{R_0 + R_1}$ siano nulli, ciò avviene per $R_0 = R_1$ e avremo l'equazione adattata nel dominio delle onde:

$$\begin{bmatrix} \mathbf{b_0} \\ \mathbf{b_1} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{-1} \\ \mathbf{-1} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{a_0} \\ \mathbf{a_1} \end{bmatrix} \tag{2.36}$$

Quanto applicato in questo esempio, è applicabile anche per la realizzazione di adattatori parallelo, trasformatori, circolatori e giratori due porte, ma anche per la realizzazione degli adattatori N-porte [5].

In molte strutture di filtri, le connessioni tra elementi possono essere descritte facilmente tramite configurazioni in serie o parallelo, tuttavia non tutti gli adattatori multi-porta possono essere ridotti a queste forme semplici.

Esiste una classe molto ampia di adattatori multi-porta, chiamati adattatori di tipo R [5], che richiedono trattamenti più sofisticati.

Gli adattatori tipo R sono una classe di adattatori multi-porta che possono includere giunzioni pure (giunzioni multi-porta semplici che non possono essere ridotte a configurazioni più basilari) e giunzioni con elementi lineari multi-porta assorbiti (giunzioni che integrano elementi lineari più complessi all'interno della loro struttura).

Per gestire queste connessioni, è necessario derivare le matrici di scattering che descrivono come le onde si comportano all'interno della giunzione.

Queste matrici sono fondamentali per calcolare il comportamento dinamico del sistema e garantire che le connessioni siano correttamente adattate.

I componenti reattivi, per quanto più complessi rispetto ai dispositivi regolati da equazioni algebriche, vengono comunque reinterpretati dai WDF in maniera efficiente, il problema sorge quando andiamo ad utilizzare dispositivi non lineari.

Questa è una classe di componenti complessi che è stata rigorosamente studiata nei WDF, in quanto particolarmente utilizzati nella realizzazione di filtri a scopo musicale.

In un circuito contenente elementi non lineari, le equazioni che descrivono il comportamento del sistema non possono essere espresse semplicemente con relazioni lineari di tensione e correnti, per esempio la relazione corrente-tensione di un diodo segue leggi esponenziali o la relazione di un transistor segue leggi quadratiche.

Un metodo comunemente usato per risolvere equazioni non lineari nei simulatori circuitali è l'iterazione di Newton-Raphson che linearizza le equazioni non lineari in ogni punto di iterazione per trovare una soluzione approssimata, si calcolano successivamente approssimazioni della soluzione fino a raggiungere una convergenza accettabile, oppure, i simulatori come SPICE, sono principalmente basati su simulazioni nel dominio del tempo, dove si risolvono direttamente le equazioni differenziali nel tempo discreto, utilizzando metodi numerici come l'integrazione implicita per mantenere la stabilità, specialmente per i circuiti rigidi o con non linearità complesse.

Grazie a questi metodi di risoluzione possiamo ottenere una simulazione precisa di MOSFET e BJT, ma a scapito del costo computazionale, che può comportare elevati tempi di risoluzione dei circuiti e della stabilità numerica.

Nei Wave Digital Filters, il trattamento delle non linearità è diverso perché si basa su una rappresentazione delle variabili del circuito tramite onde, piuttosto che direttamente su tensione e corrente, concentrandosi su scattering e adattamento di impedenza, inoltre risulta comodo l'utilizzo degli adattatori di tipo R per gestire le non linearità multiple nel circuito [6].

L'utilizzo delle onde ci concede di avere una maggiore stabilità numerica rispetto all'utilizzo degli altri metodi, ma comportando una maggiore complessità di elaborazione dei modelli.

2.4 Rappresentazione circuitale WDF

Nei WDF i circuiti elettrici vengono rappresentati attraverso una struttura ad albero, in quanto risulta più efficiente rispetto ai normali grafi per il metodo modulare che sfrutta questa tecnica di simulazione.

In questo metodo, i circuiti elettrici vengono rappresentati come una struttura gerarchica ad albero [7] in cui ogni nodo corrisponde ad una giunzione tra più componenti, come quelle in serie o parallelo, e ogni foglia rappresenta un singolo componente del circuito, come resistenze, induttori, condensatori.

Gli elementi non lineari, come i diodi, sono generalmente posizionati al nodo radice per gestire efficacemente il comportamento non lineare, consentendo un calcolo condizionale delle onde riflesse a seconda dello stato del componente (polarizzato o non polarizzato).

La rappresentazione ad albero dei WDF ci consente di scomporre un circuito complesso in elementi più semplici, che possono essere trattati indipendentemente e quindi di diminuire il costo computazionale, in quanto non necessiteremo di risolvere grandi equazioni simultaneamente.

Adesso proseguiamo con un esempio di circuito [8]:

Per costruire una struttura ad albero WDF a partire da un circuito analogico tradizionale, è necessario seguire alcuni passaggi fondamentali:

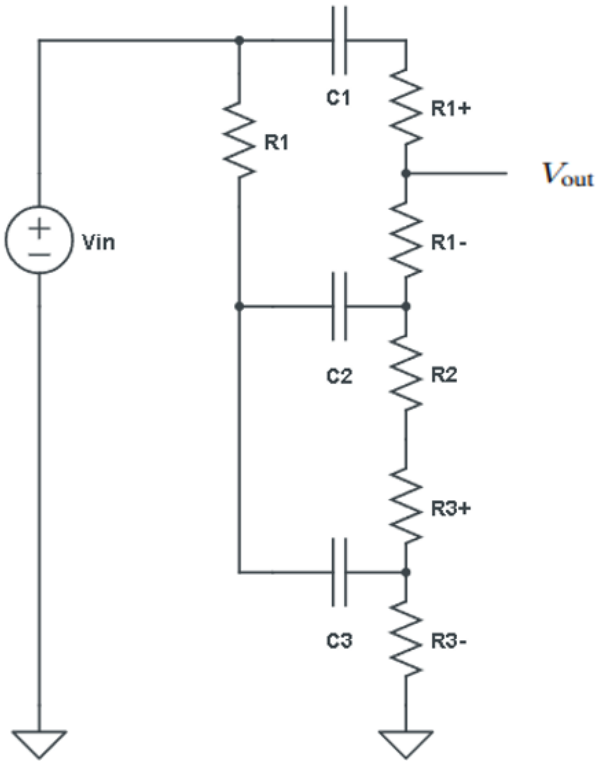

Figura 2.1: Bassman tone stack circuit

Il circuito viene raffigurato dapprima tramite un grafo, per individuare rami e nodi:

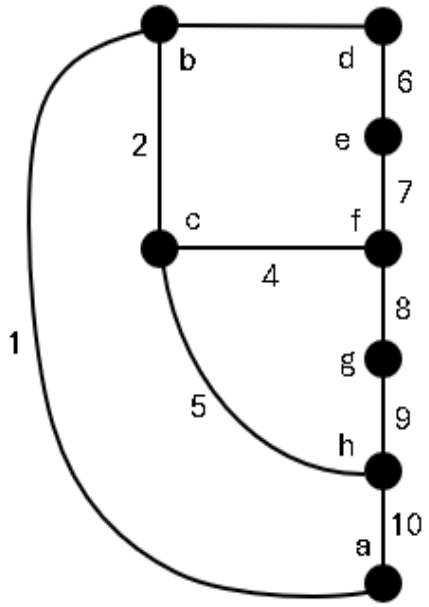

Figura 2.2: grafo del circuito di Bassman

Successivamente viene scomposto nei suoi elementi fondamentali, quali resistenze, condensatori, induttori e diodi, viene identificato il tipo di connessione tra i componenti (serie o parallelo o adattatori tipo R) evidenziando i nodi e rami che li collegano:

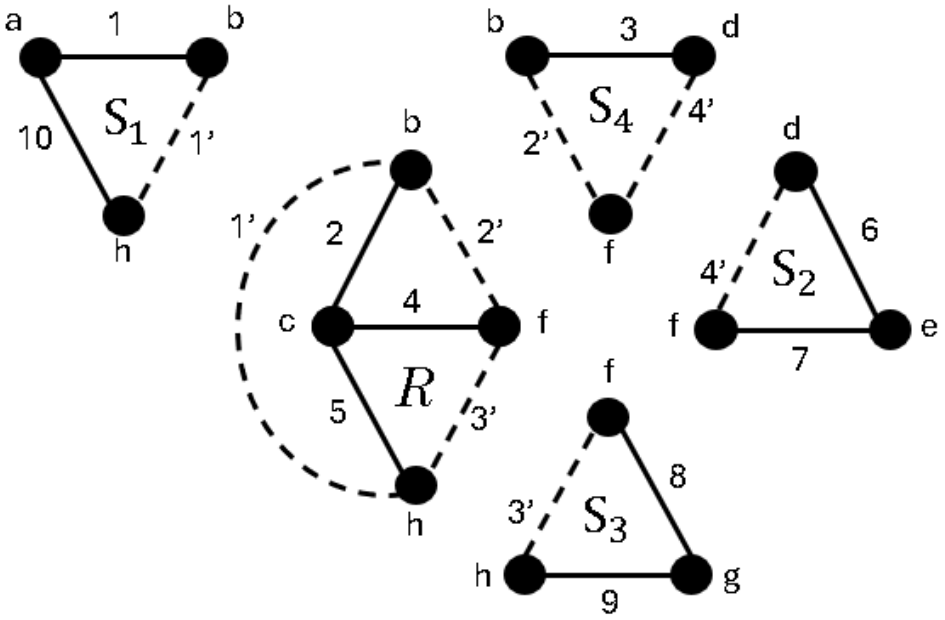

Figura 2.3: separazione dei componenti

Individuati foglie (componenti singoli) e adattatori (serie, parallelo, tipo R) potremo realizzare il nostro albero:

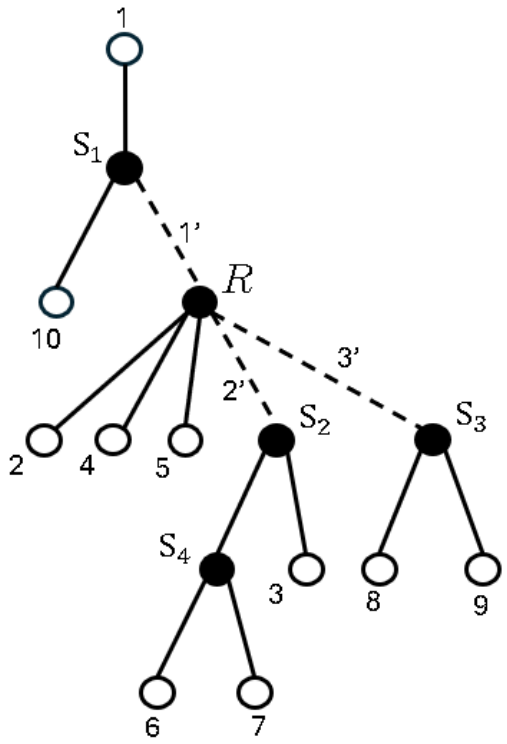

Figura 2.4: albero SPQR

Infine, ogni componente viene rappresentato con il proprio simbolo circuitale in una raffigurazione a blocchi:

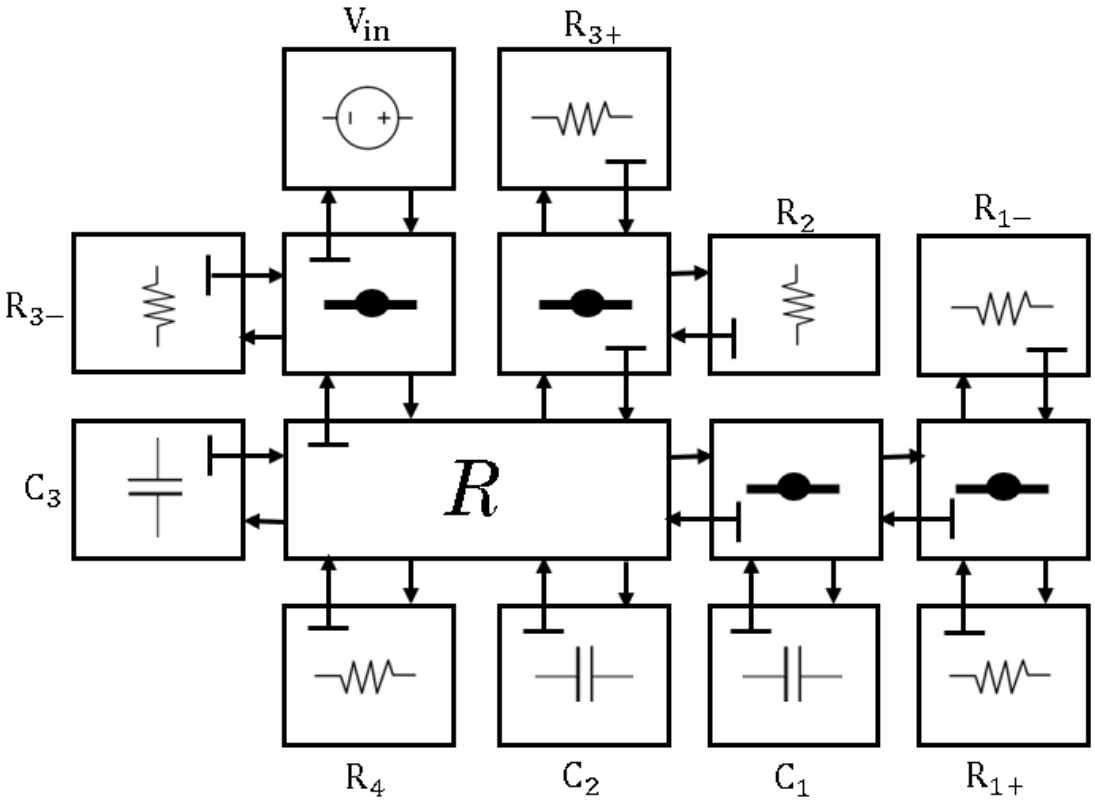

Figura 2.5: struttura adattatore WDF corrispondente al circuito

Come già detto, questo modo di rappresentazione del ciruito dei WDF, aiuta ad avere un costo computazionale minore ed è di facile comprensione del circuito, in quanto descrive esattamente il modo in cui viene descritto il circuito anche dal punto di vista di codice.

2.5 Proprietà e vantaggi dei WDF

Il metodo dei Wave Digital Filter è riuscito a trovare impiego grazie alla trasformazione delle variabili circuitali di tensione e corrente, in variabili d'onda, semplificando

l’implementazione di componenti reattivi grazie alla trasformazione di equazioni differenziali in algebriche.

Inoltre, il metodo di rappresentazione del circuito ad albero, garantisce una modularità del metodo e quindi una ulteriore semplificazione dei dati da processare da un compilatore.

Questi fattori vanno a comportare una riduzione del costo computazionale per la risoluzione dei circuiti, il che rende i WDF ottimi per una risoluzione in tempo reale, soprattutto se paragonato con altri metodi.

In aggiunta abbiamo una notevole stabilità numerica data dall’utilizzo delle equazioni algebriche piuttosto che differenziali e grazie all’adattamento dei dispositivi.

Esattamente come negli altri metodi di simulazione, i componenti non lineari risultano difficili da implementare e di alto costo computazionale nella risoluzione dei circuiti.

I WDF, per essere utilizzati, necessitano di una completa conoscenza del metodo, della trasformazione dei componenti circuitali e come già detto, i circuiti non lineari o a tempo variabile possono essere difficili da modellare accuratamente.

Capitolo 3

Strumenti e tecnologie utilizzate

3.1 Introduzione a Python per la simulazione di WDF

Python è un linguaggio di programmazione di alto livello, orientato a oggetti e adatto, tra gli altri usi, a sviluppare applicazioni.

Spesso è tra i primi linguaggi di programmazione ad essere studiato tra i neofiti, in quanto simile a uno pseudo-codice, semplice e versatile, inoltre dispone di un ricco assortimento di tipi, funzioni di base, librerie standard e sintassi avanzate quali *slicing* e *list comprehension*.

È un linguaggio interpretato, il che significa che il codice Python può essere eseguito direttamente senza la necessità di una compilazione, questo lo rende estremamente flessibile e adatto per una vasta gamma di applicazioni, tra cui lo sviluppo web, Data science (con librerie come NumPy, Pandas, e Matplotlib), Machine learning, Automazione e scripting, Sviluppo di software

Python è un linguaggio molto popolare, in quanto è di facile apprendimento grazie a una sintassi chiara e leggibile, inoltre possiede un ampio ecosistema, in quanto la community di Python ha creato un'enorme quantità di librerie e strumenti per ogni campo applicativo, dispone di un cross-platform (funziona su Windows, macOS e Linux) e possiede ancora una comunità attiva, quindi tutorial e forum disponibili per imparare e risolvere problemi.

Python è disponibile sul sito ufficiale https://www.python.org per delucidazioni, installazione ed aggiornamenti.

I linguaggi di programmazione necessitano di un ambiente di sviluppo integrato (IDE), per scrivere codici, compilarli e gestire progetti; In questa tesi utilizzeremo PyCharm,

un IDE sviluppato da JetBrains, in grado di fornire strumenti di debugging avanzati, una facile gestione di progetti, supporti per lo sviluppo di framework web e un sistema di completamento del codice intelligente che permette di migliorare la produttività.

È disponibile al sito [10] insieme a un video introduttivo ed approfondimenti.

3.2 Presentazione di pywdf: Panoramica e installazione

pywdf è una libreria di Python open-source sviluppata da Gustav Anthon, Xavier Lizarraga-Seijas e Frederic Font, basata sul lavoro in C++ di Jatin Chowdhury [11], disponibile su github [12] e utilizzabile solo su Linux ed installabile tramite

“pip install git+https://github.com/gusanthon/pywdf”.

Pywdf è utile per creare, modellare e simulare circuiti con il metodo dei Wave Digital Filter grazie a vari tools di analisi e alla quantità di componenti elettrici implementati nella libreria.

Il framework contiene al suo interno tre file principali, wdf.py per la creazione di elementi circuitali, circuit.py in cui troviamo i tools di analisi e rtype.py che consente l’implementazione degli adattatori di tipo R.

Tutti i componenti elettrici contenuti in wdf.py sono figli della classe “baseWDF”, che fornisce le caratteristiche base dei componenti, come la possibilità di connettersi a elementi precedenti sulla gerarchia della struttura ad albero e la possibilità di avere onde in ingresso.

Altri componenti sono figli indiretti, come il diodo e il generatore ideale di tensione, in quanto devono essere inseriti necessariamente come nodo radice dell’albero e discendono da “rootWDF” che forza questo utilizzo impedendo che vengano inseriti in altre posizioni gerarchiche, quindi anche volendo non è possibile utilizzare la rappresentazione circuitale ordinaria.

Nel file wdf.py troviamo cortocircuiti, circuiti aperti, interruttori, resistenze, capacità, induttanze, invertitore di polarità, adattatore serie, adattatore parallelo, generatore ideale di tensione, generatore resistivo di tensione, diodo e diodo pair.

Non notiamo generatori di corrente in quanto gli sviluppatori hanno applicato, per la trasformazione dei componenti nel dominio delle onde, solamente delle onde di tensione.

Nel file rtype.py possiamo generare gli adattatori di tipo R a n-porte grazie alla classe "RtypeAdaptor", figlia anch'essa di baseWDF e una figlia della classe citata per la generazione di adattatori tipo R n-porte non lineari.

Nel codice all'interno di circuit.py invece, troviamo i tools di analisi e l'intero processo di elaborazione dell'onda in ingresso del circuito, che consiste nel processare ogni singolo campione del segnale.

Per la generazione del circuito, una volta inizializzati tutti i componenti necessari, basterà impostare una sorgente di tensione, il componente destinato ad essere il nodo radice e il componente da cui prendiamo l'onda da esaminare, cioè l'uscita.

I tools implementati da circuit.py prevedono il processamento del segnale, il calcolo della risposta all'impulso, l'analisi dello spettro in frequenza, analisi transitoria in AC.

Oltre ai tre file che costituiscono il cuore della libreria, vi sono anche degli esempi di circuiti, allo scopo di far capire il funzionamento di pywdf e osservare il metodo di visualizzazione dei grafici offerto.

3.3 Altri strumenti e librerie usate

Nello sviluppo dei circuiti con il metodo WDF utilizzato da pywdf, è stato trovato opportuno l'utilizzo congiunto di altre librerie di Python, scaricate sempre da github, allo scopo di rendere più agevole il lavoro svolto.

La prima utilizzata è "Matplotlib", è una libreria di visualizzazione dei dati che consente di creare grafici e plot 2D e 3D di vario tipo [13], come grafici a linee, barre, scatter plot, istogrammi e molto altro, consente di esportare i grafici in vari formati ed è compatibile con altre librerie, inoltre è particolarmente apprezzata per la sua flessibilità e semplicità nell'uso.

La seconda è “NumPy”, una libreria base per l'elaborazione numerica, offre un supporto per array multidimensionali e funzioni per l’algebra lineare.

Usata nel codice di programmazione per la generazione di onde in ingresso al circuito.

CAPITOLO 4

Simulazione dei Wave Digital Filters

In questa tesi andremo a realizzare due circuiti, un filtro RC passa-basso passivo, eseguendo grafici per diverse frequenze di taglio e un raddrizzatore a singola semionda.

Parleremo del funzionamento di tali circuiti, illustreremo gli schemi circuitali, la loro rappresentazione ad albero, eseguendo il plot della onda entrante nel circuito, dell'onda uscente e la risposta in frequenza.

4.1Implementazione di un filtro passa-basso

Un filtro passa-basso passivo RC è un circuito molto semplice e comune, utilizzato per attenuare le alte frequenze di un segnale e lasciar passare principalmente quelle al di sotto di una data frequenza di soglia (taglio).

In elettronica può essere costituito da circuiti di diverso tipo, può essere di tipo attivo se presenta elementi attivi quali amplificatori operazionali, o passivo se presenta componenti come resistori e condensatori.

In base alla pendenza di taglio della risposta in frequenza, possiamo distinguere filtri di ordine diverso, primo ordine (20 dB per decade), di secondo ordine (40 dB per decade), di terzo ordine (60 dB per decade) e così via

Il circuito in esame è composto da due elementi, una resistenza (R) e un condensatore (C), che insieme determinano la frequenza di taglio del filtro, collegati in serie tra loro e quindi, nel nostro circuito, rappresentabili da un adattatore serie nella costruzione della struttura ad albero.

Iniziamo rappresentando il circuito del filtro passa-basso RC:

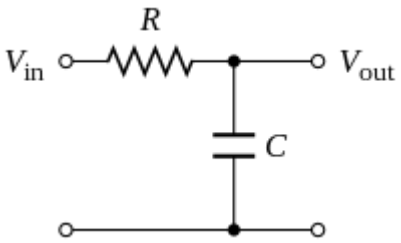

Figura 4.1: circuito filtro passa-basso

Come detto, la resistenza R e il condensatore C sono in serie tra loro e quindi verranno rappresentati tramite un adattatore serie:

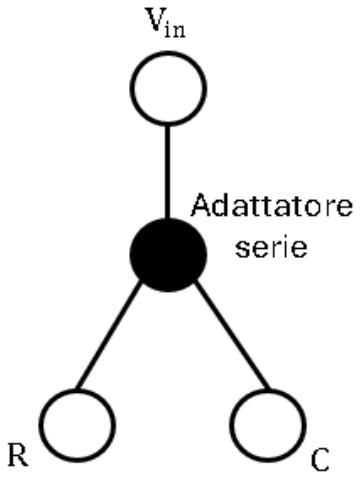

Figura 4.2: albero SPQR filtro passa-basso

Ora raffigureremo il circuito nella struttura convenzionale per i WDF:

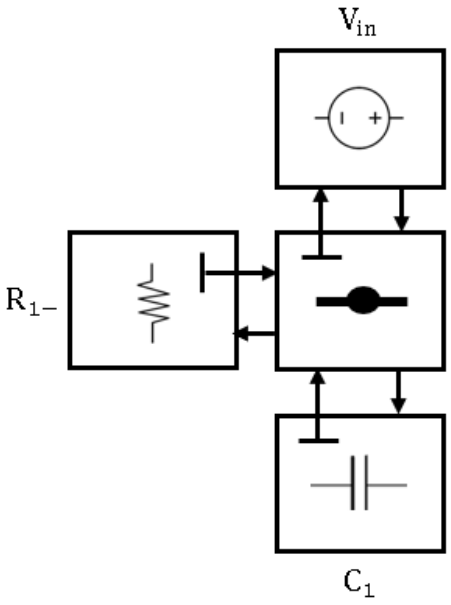

Figura 4.3: rappresentazione WDF fitro passa-basso

Dopo aver raffigurato il filtro passa-basso con la sua rappresentazione ad albero, svolgeremo la simulazione del circuito e paragoneremo i risutati ottenuti da pywdf con quelli simulati con LTspice.

4. 2 Analisi delle prestazioni del filtro simulato e confronto con LTspice

La simulazione del filtro è stata svolta per diverse frequenze, a partire dalla frequenza di Nyquist (metà della frequenza di campionamento) fino ad arrivare ad una frequenza di taglio prossima allo zero per osservare le distorsioni del segnale e il comportamento del filtro.

Abbiamo scelto di filtrare una onda quadra di ampiezza unitaria, che riceveremo in ingresso al generatore ideale di tensione, in quanto, dopo il filtraggio, tende ad assumere un comportamento sinusoidale.

Frequenza di campionamento $f_c = 5000$ Hz, Frequenza di taglio $f_T = 2500$ Hz:

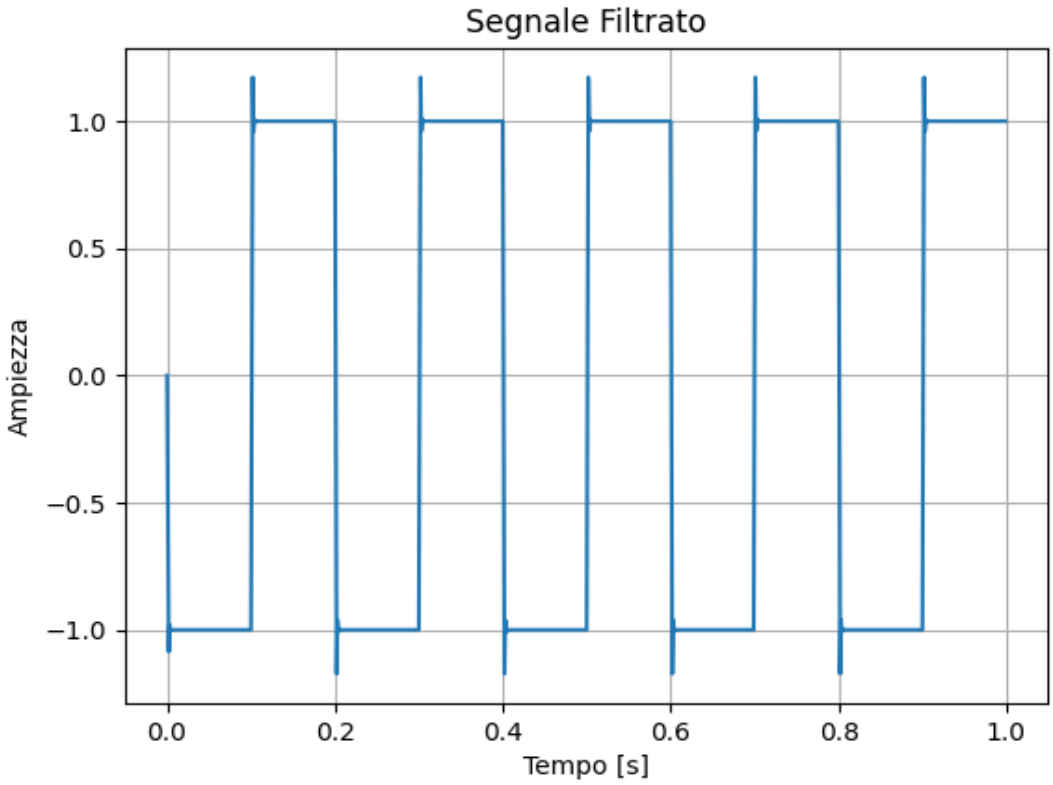

Figura 4.4

La prima cosa che possiamo notare è l'inversione dell'onda quadra, oltre alla distorsione che discuteremo successivamente.

L'inversione dell'onda è dovuta alla natura dell'adattatore serie che abbiamo descritto a pagina 17, come si può evincere dalla equazione che lega l'onda riflessa b e l'onda incidente a adattata, la matrice di scattering presenta coefficienti sulla anti-diagonale di segno negativo (2.36).

Tali coefficienti rappresentano la trasmissione dell'onda tra i due dispositivi in serie, pertanto portano all'inversione dell'onda.

Questo porta all'inserimento di un componente che normalmente non sarebbe necessario alla realizzazione del circuito, l'invertitore di polarità, comportando un aumento, seppur leggero, del costo computazionale e ad una non immediatezza della comprensione di un semplice circuito.

Una volta inserito l'invertitore di segnale nel codice, l'onda riflessa dal condensatore non risulterà più invertita e potremo continuare ad analizzare il nostro filtro:

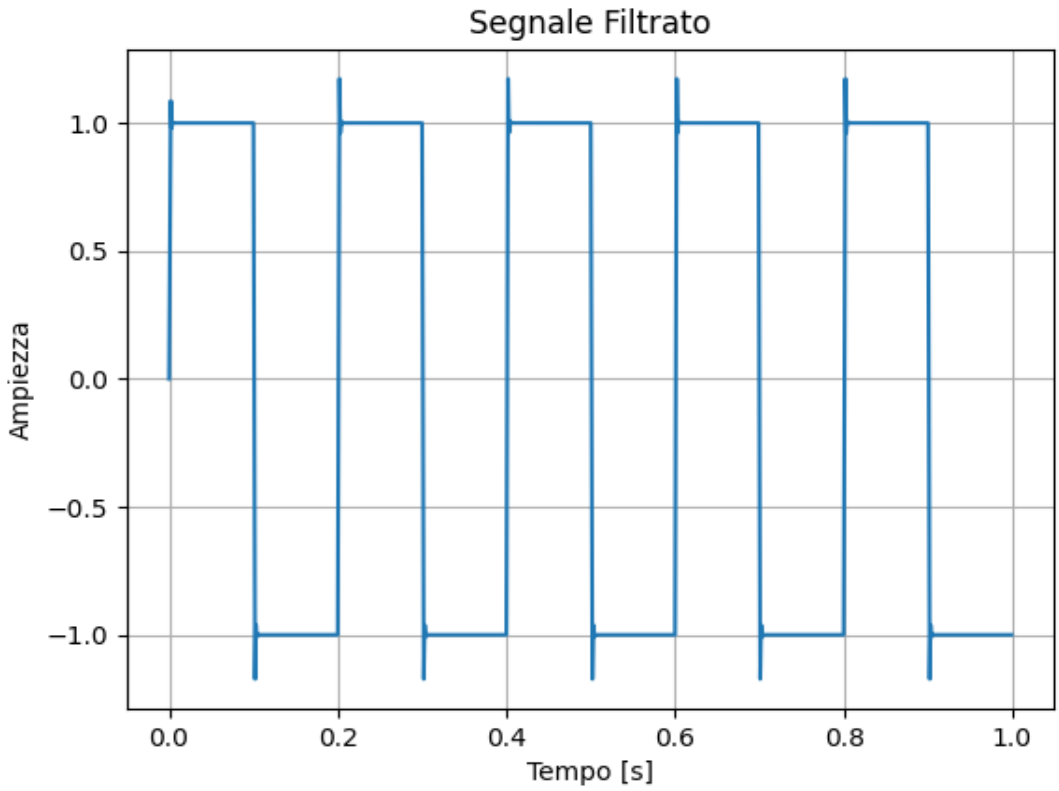

Figura 4.5

Mentre la risposta in frequenza del filtro sarà la seguente:

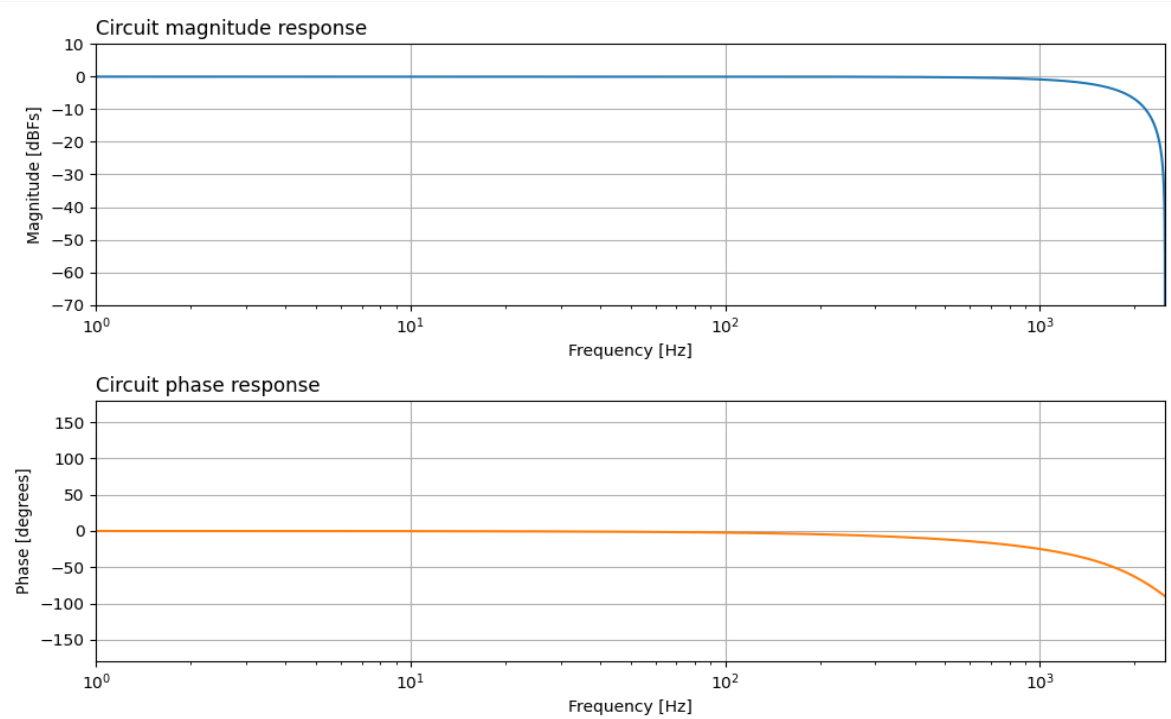

Figura 4.6

Possiamo notare che l'onda in uscita dal filtro (4.5), risulta avere dei picchi netti, laddove dovrebbe incominciare ad arrotondarsi a causa del filtraggio.

Questo avviene perché la frequenza di taglio è ancora troppo alta in confronto alla frequenza di campionamento e il simulatore circuitale in esame non riesce ad avere una attenuazione delle alte frequenze tanto da arrotondare i picchi, causando una distorsione ben visibile.

Per quanto riguarda la risposta in modulo del filtro, rimane costante a zero fino ad arrivare in prossimità della frequenza di taglio facendo passare le frequenze sottosoglia, poi abbiamo un rapido decadimento alla frequenza di taglio.

Questo è un comportamento tipico di un filtro passa-basso.

Per la risposta in fase notiamo una decadenza avvicinandoci alla frequenza di taglio, mentre per le basse frequenze è prossima a 0^o.

Al crescere della frequenza, la fase inizia a scendere fino a raggiungere un angolo massimo negativo di -90^o, indicativo del ritardo di fase introdotto dal filtro.

Confrontiamo con i risultati ottenuti da LTspice a parità di circuito:

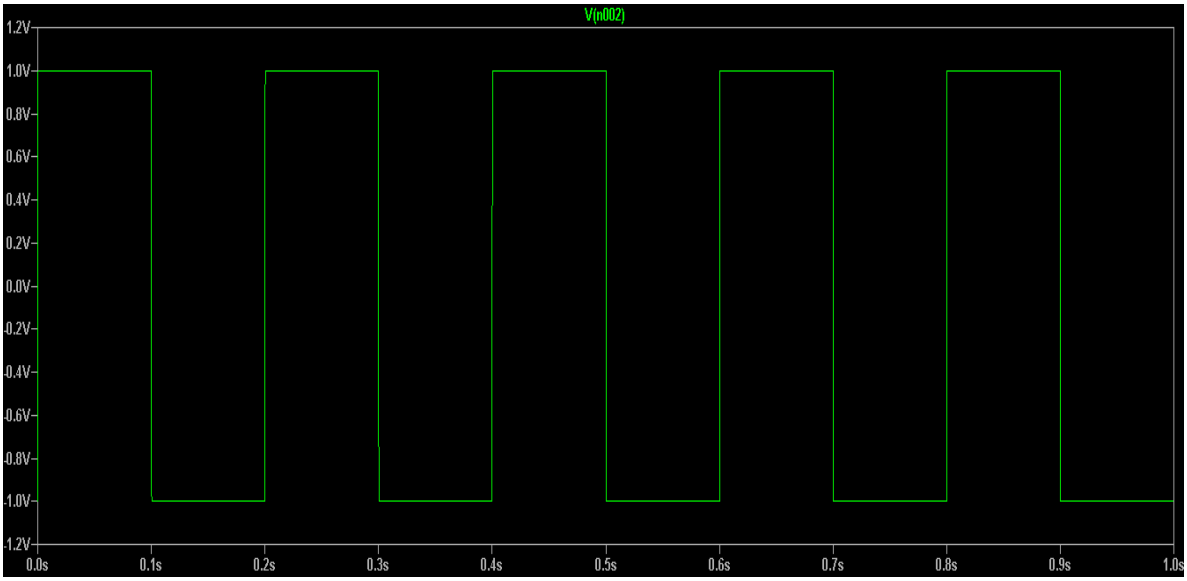

Figura 4.7

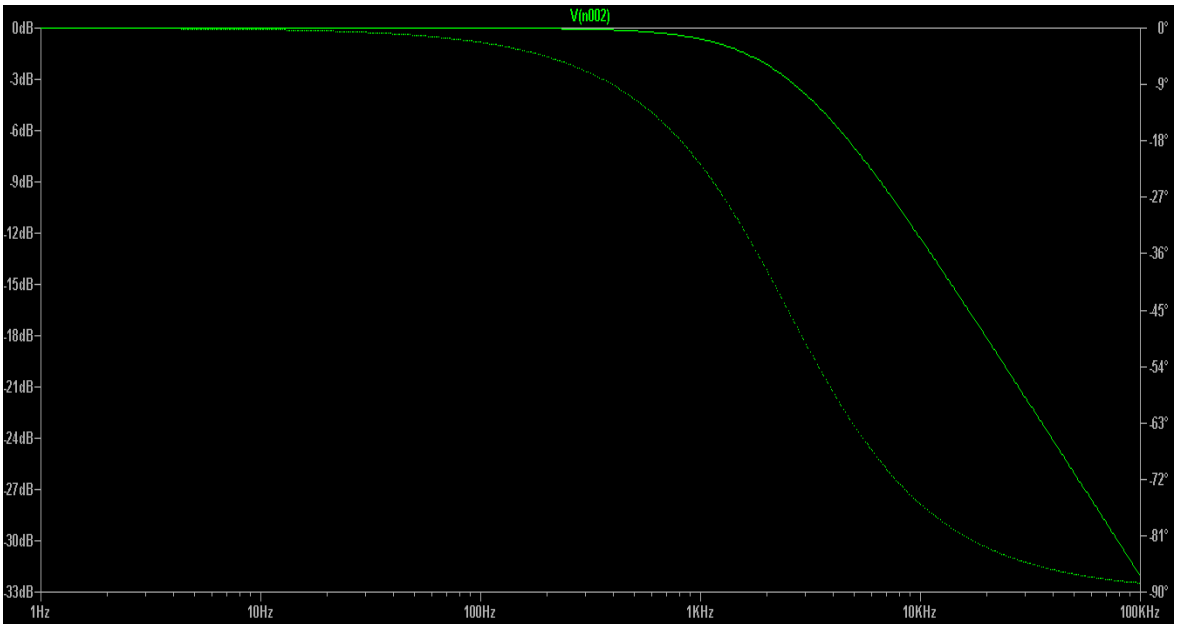

Figura 4.8

Dalla simulazione con LTspice (4.7), a differenza della simulazione con pywdf, abbiamo un'onda di uscita dal circuito che inizia ad arrotondarsi sui fronti di salita e discesa, le alte frequenze non sono ancora abbastanza attenuate da riuscire a dare un effetto perfettamente visibile al segnale, ma non presenta distorsioni.

La risposta in modulo (linea continua) e fase (linea tratteggiata) presentano le medesime prestazioni; Tuttavia, la rappresentazione con LTspice risulta più pratica.

Per $f_c = 5000$ Hz e $f_T = 500$ Hz, dieci volte inferiore rispetto alla f_c :

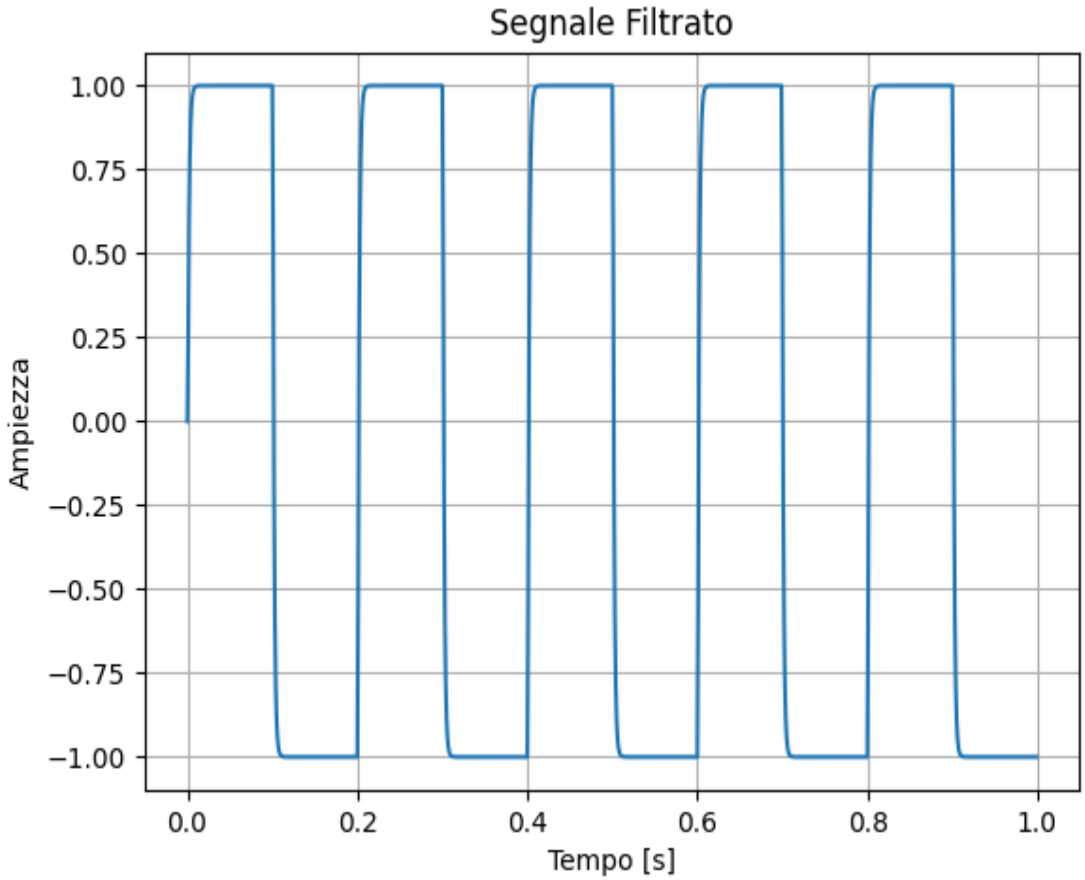

Figura 4.9

Inseriamo anche la risposta in modulo e fase del filtro:

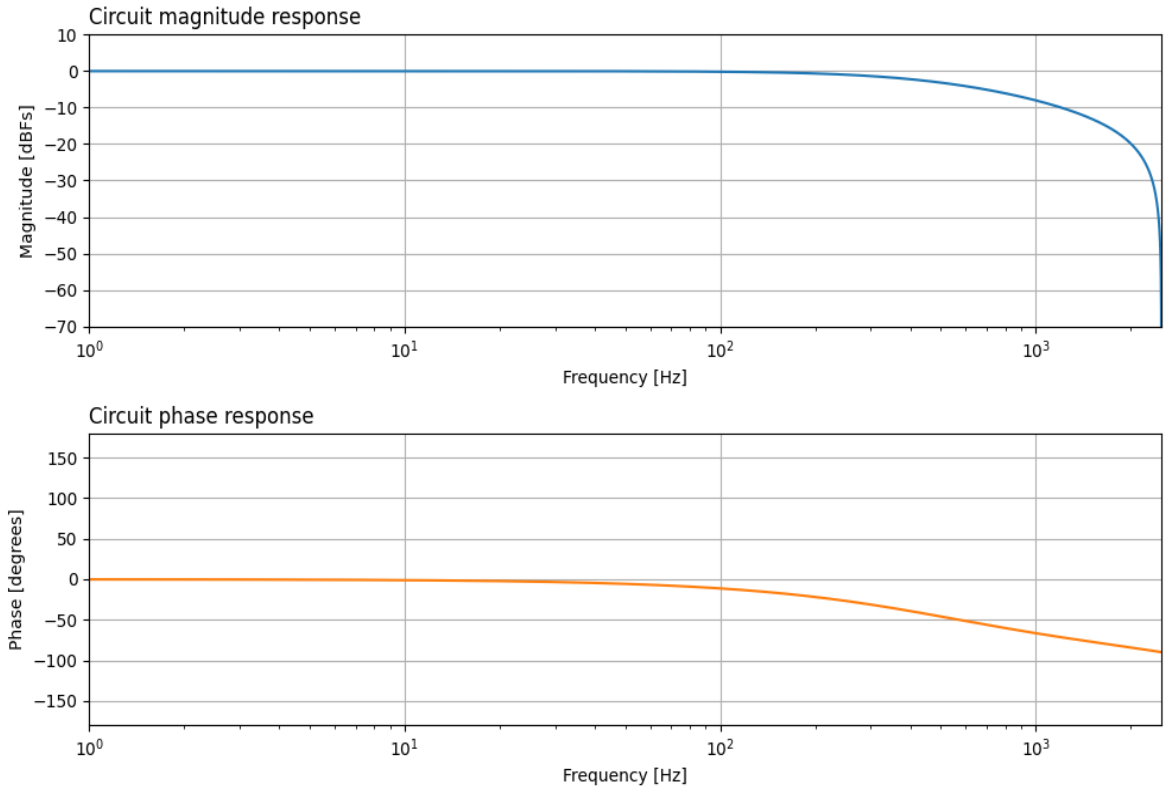

Figura 4.10

Utilizzando una frequenza di taglio dieci volte inferiore rispetto alla frequenza di campionamento, riusciamo a vedere un arrotondamento dei picchi del segnale (4.9), quindi un filtraggio delle frequenze superiori alla soglia, anche se la forma d'onda quadra è quasi completamente mantenuta.

Il comportamento del filtro rimane comunque tipico.

Nella risposta in modulo notiamo un comportamento analogo all'impostazione precedente, ma la decadenza avviene in prossimità della nuova frequenza, cioè $f_T = 500$ Hz, il comportamento del filtro rimane quello aspettato.

Invece nella risposta in frequenza riusciamo a notare più facilmente il decadimento, per frequenze superiori alla soglia, che inizia in prossimità di $f_T = 500$ Hz e si arresta a -90^o.

Anche in questo caso il comportamento del filtro è esattamente quello atteso.

Confrontiamo con i risultati ottenuti con LTspice:

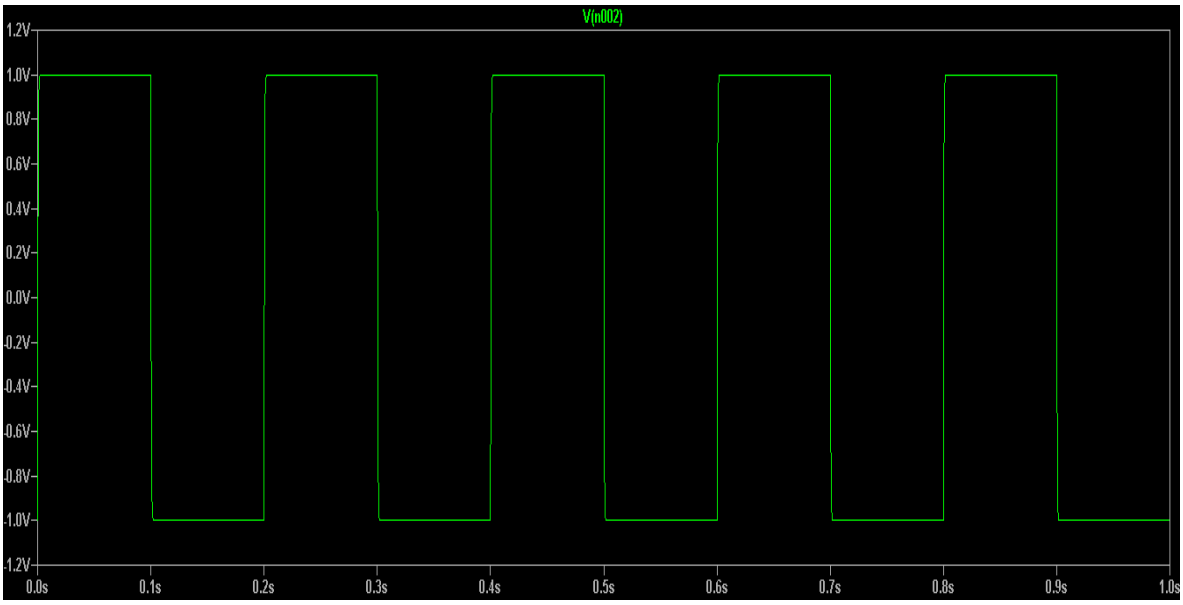

Figura 4.11

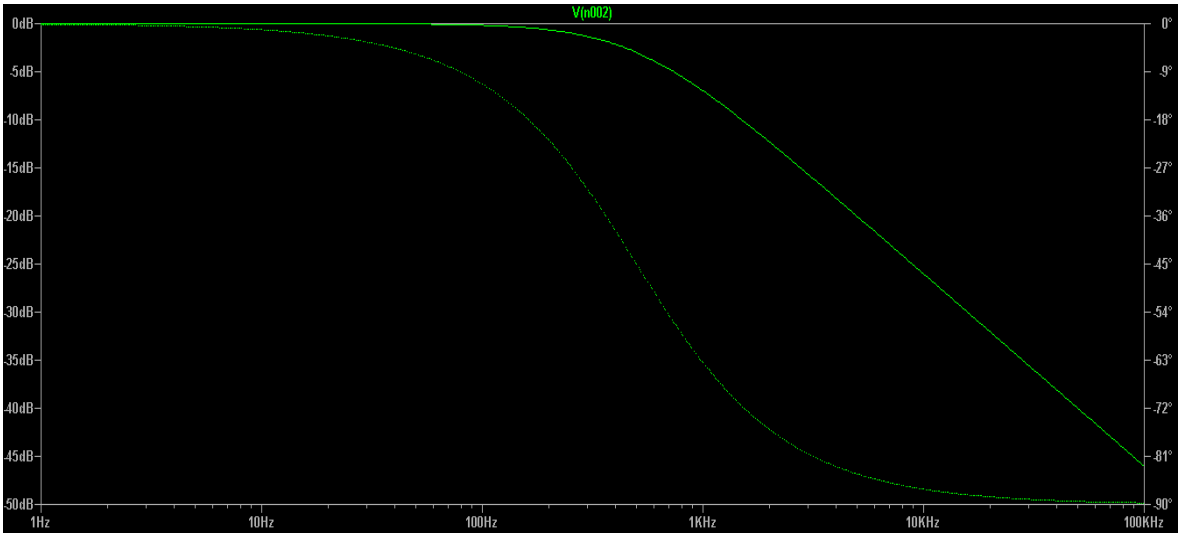

Figura 4.12

La differenza tra (4.9) e (4.11) potrebbe essere data solo dalla qualità dell'immagine, comunque da entrambe le simulazioni possiamo notare la piccola attenuazione delle alte frequenze.

Confrontando invece la (4.10) con la (4.12), anche in questo caso notiamo la stessa risposta, come detto precedentemente, il grafico presentato da LTspice sulla risposta del filtro, presenta in modo più dettagliato il contenuto rispetto alla simulazione in Python.

Ora osserviamo il comportamento del filtro utilizzando una frequenza di taglio cento volte inferiore rispetto alla frequenza di campionamento, quindi filtrando maggiormente le alte frequenze:

Per $f_c = 5000$ Hz e $f_T = 50$ Hz, cento volte inferiore rispetto alla f_c:

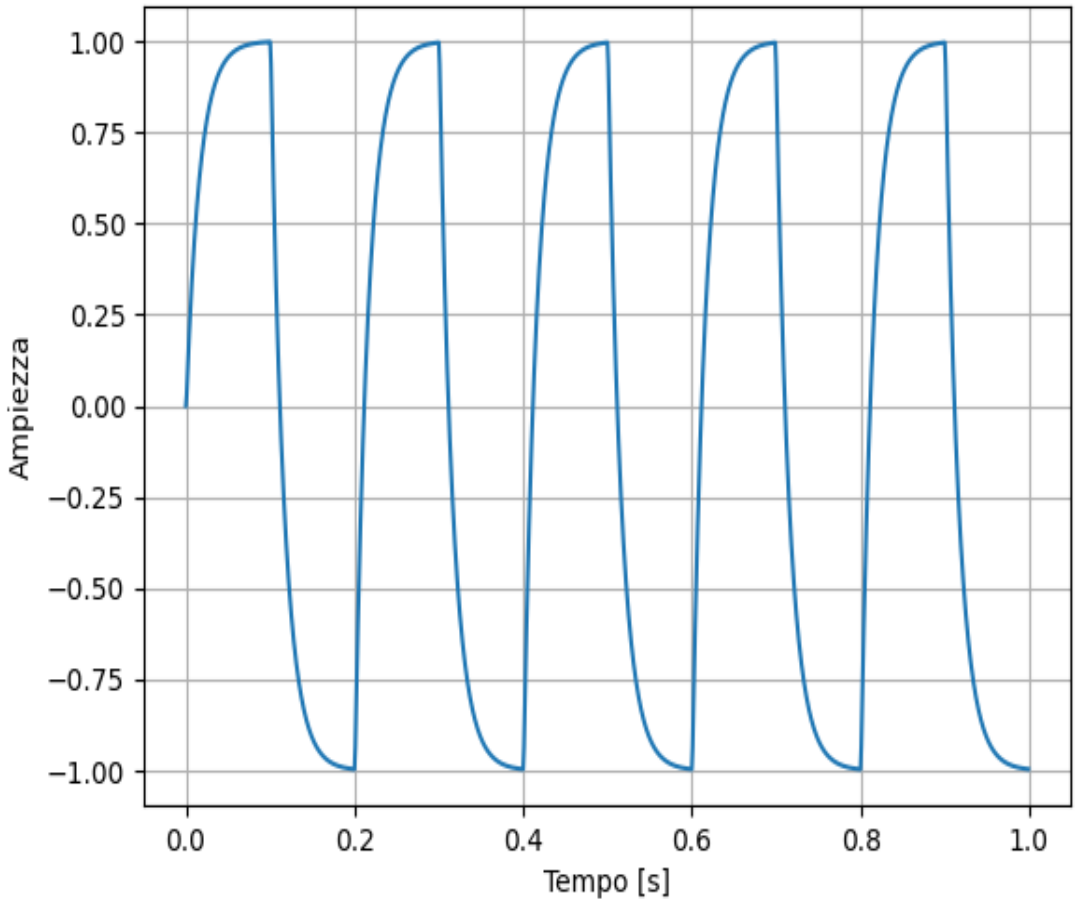

Figura 4.13

Visualizziamo anche la risposta in modulo e fase del filtro:

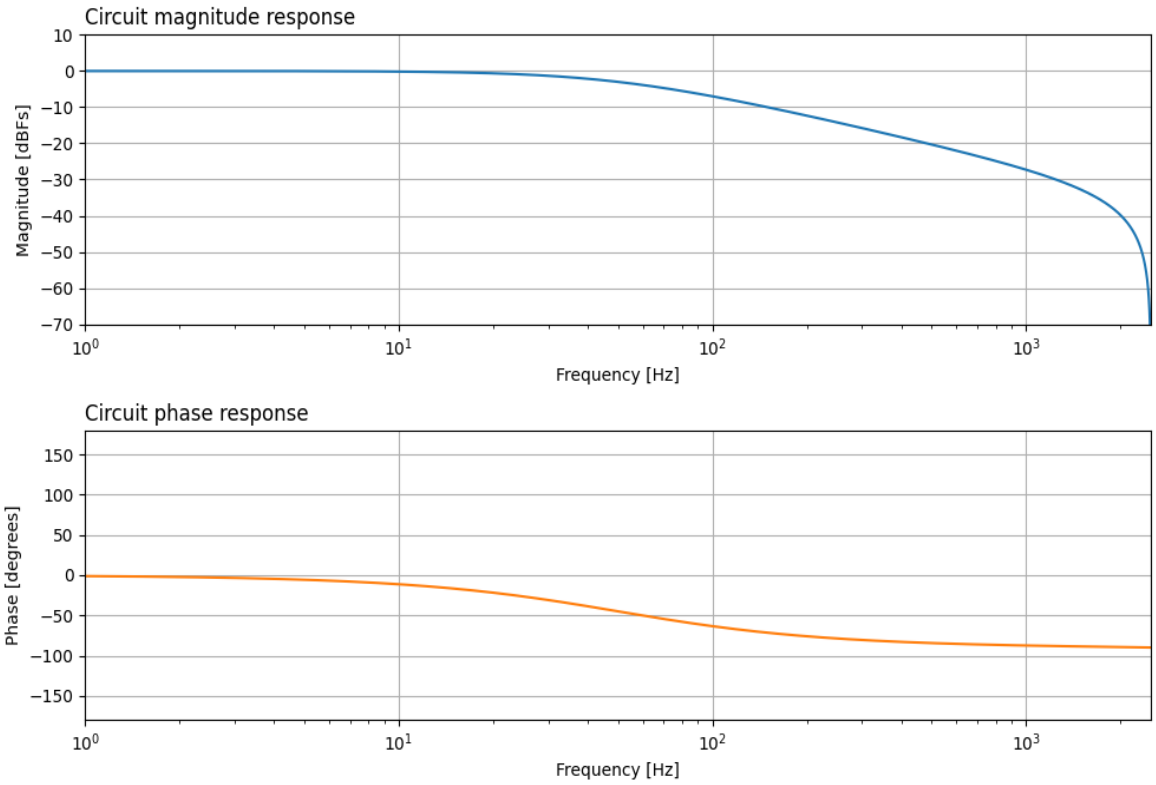

Figura 4.14

Dal grafico (4.13) notiamo che, utilizzando una frequenza di taglio tanto bassa rispetto alla frequenza di campionamento, abbiamo un importante arrotondamento dei picchi, quindi un addolcimento dei fronti d'onda, inoltre avremo una transizione dal picco positivo al picco negativo più graduale e quindi un allargamento della banda di transizione.

Nella risposta del modulo riusciamo a notare, in modo più marcato rispetto agli altri grafici, la pendenza di – 20 dB per decade introdotta dal condensatore, e avremo una forte attenuazione delle componenti in frequenza notevolmente superiori alla frequenza di taglio.

La risposta della fase del filtro introduce uno sfasamento crescente man mano che ci avviciniamo e superiamo la frequenza di taglio, mentre per frequenze inferiori lo sfasamento è quasi nullo.

La componente fondamentale dell'onda quadra subirà un piccolo sfasamento (meno significativo rispetto alle alte frequenze), ma le armoniche superiori verranno sfasate sempre di più, portando a un'onda in uscita che non ha le caratteristiche nette e immediate dell'onda quadra originale.

Visualizziamo il filtro realizzato con LTspice:

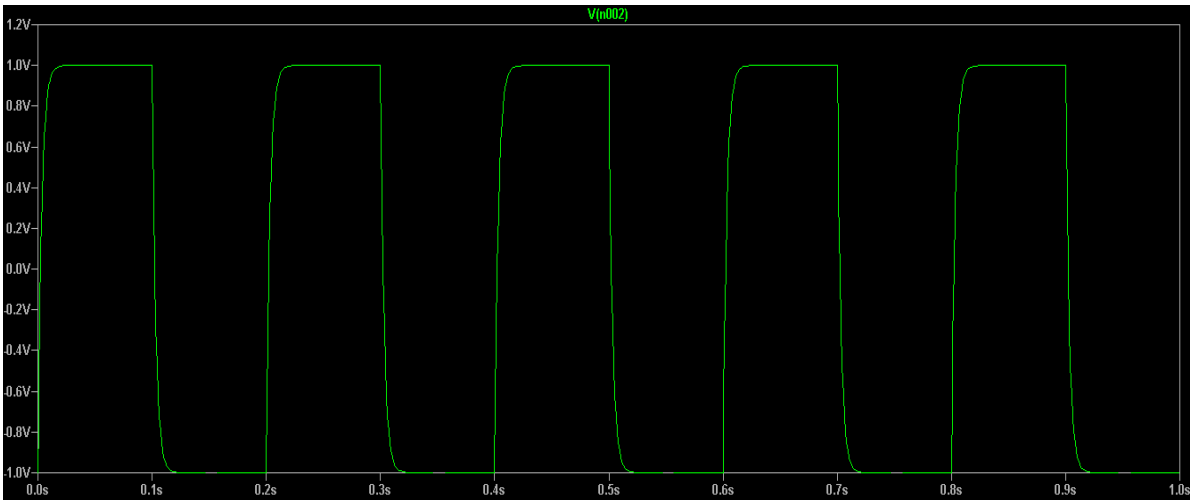

Figura 4.15

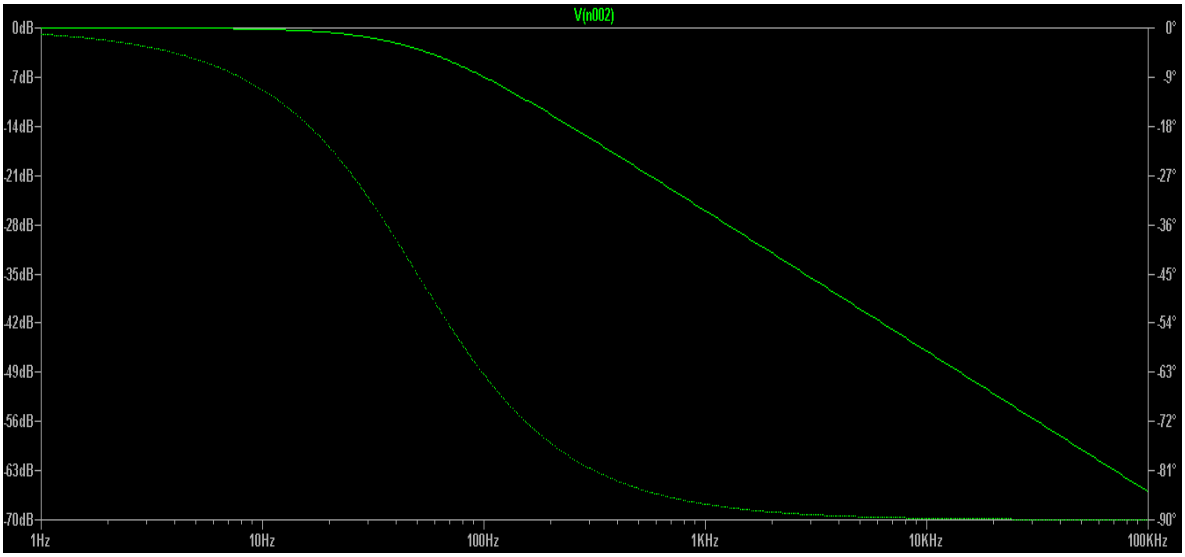

Figura 4.16

Il segnale che vediamo in figura (4.13), simulato da pywdf, presenta una maggior attenuazione delle alte frequenze rispetto alla simulazione con LTspice, e quindi una maggior prestazione del filtro.

Dopo aver mostrato il comportamento del filtro per frequenze accettabili per la preservazione delle armoniche fondamentali del segnale, si è deciso dimostrare il comportamento del filtro e del simulatore, per frequenze di taglio che potrebbero portare a distorsioni:

Per $f_c = 5000$ Hz e $f_T = 1$ Hz :

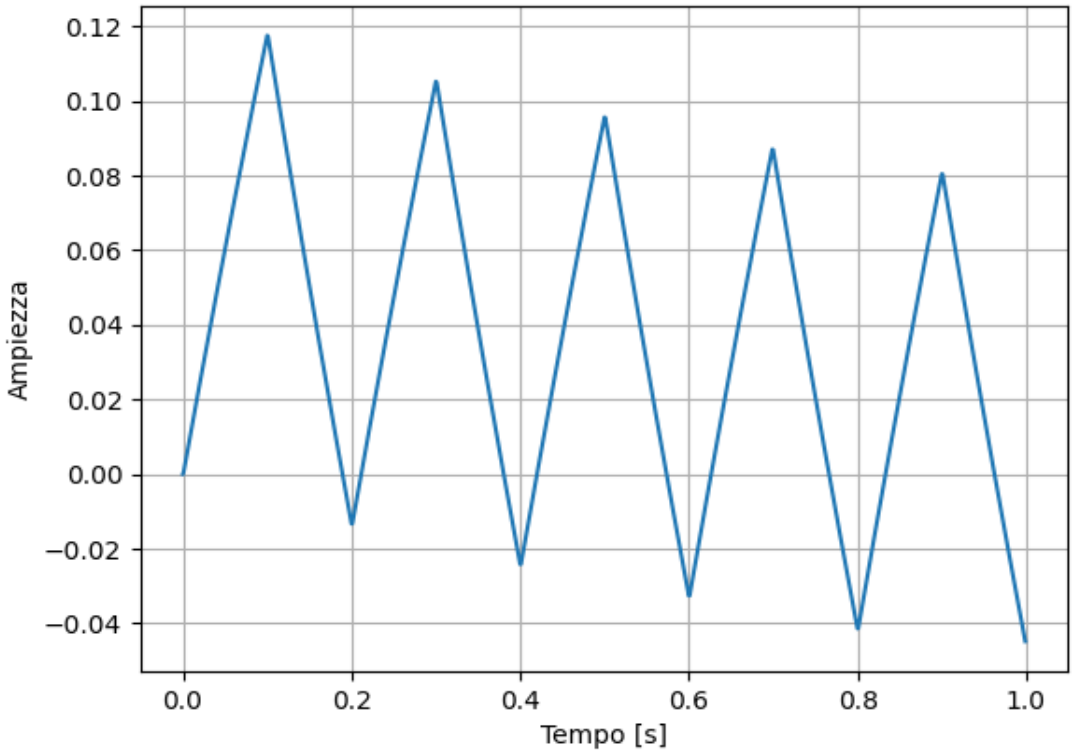

Figura 4.17

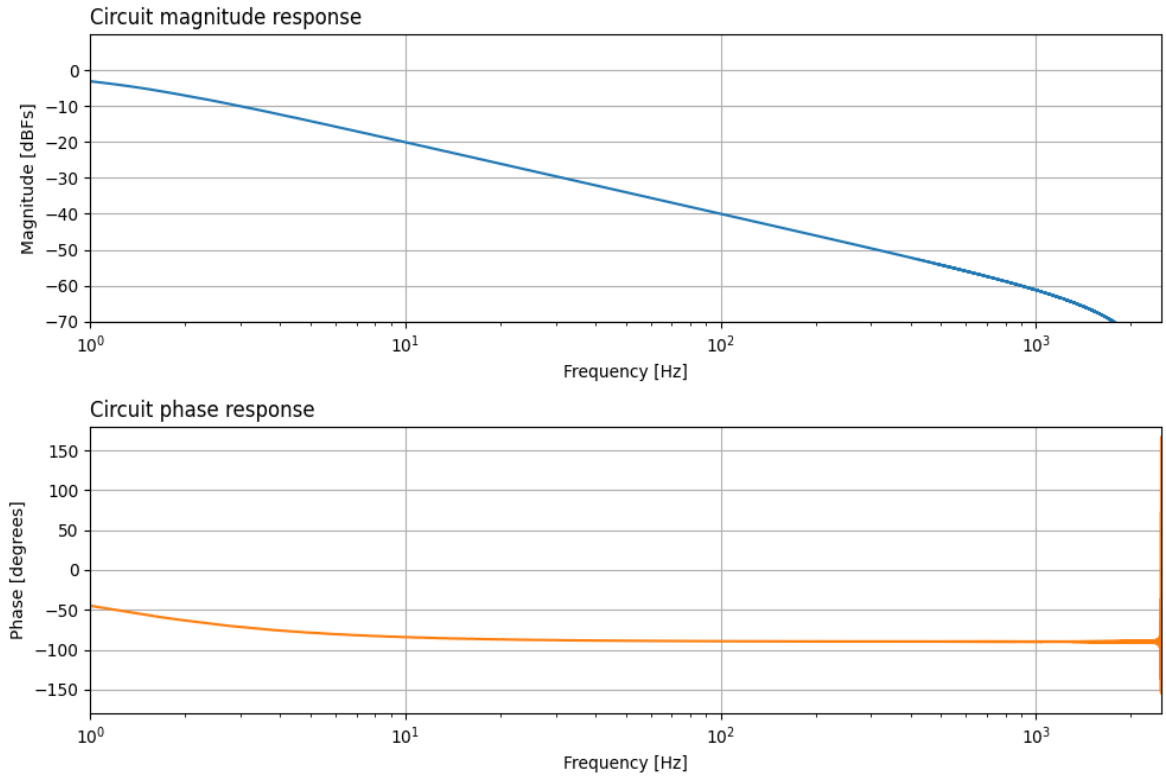

Figura 4.18

Dalla risposta in modulo in (4.10) possiamo notare una pendenza di -20 dB per decade costante, ciò comporterà una uscita del filtro con un'ampiezza vicino allo zero, infatti non raggiunge 0.12 V come picco massimo.

Notiamo invece, nella risposta in fase, che per frequenze strettamente maggiori della frequenza di taglio, un notevole sfasamento che potrebbe non essere facilmente individuabile nel segnale in uscita.

Quindi in questo caso il nostro simulatore non è riuscito a rappresentare correttamente il segnale.

Verifichiamo l'andamento dell'onda e della risposta del filtro con LTspice:

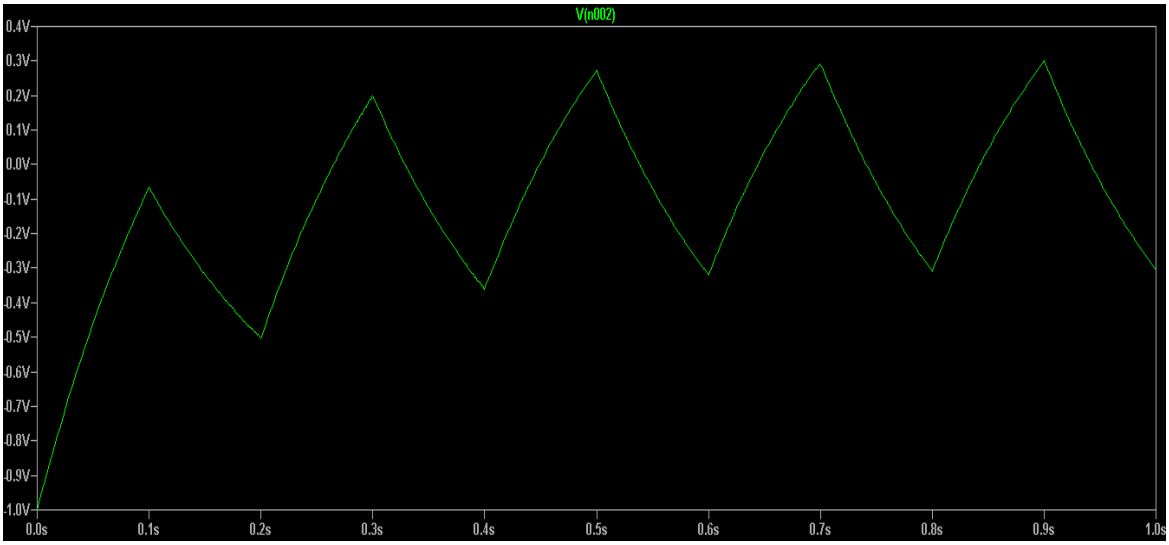

Figura 4.19

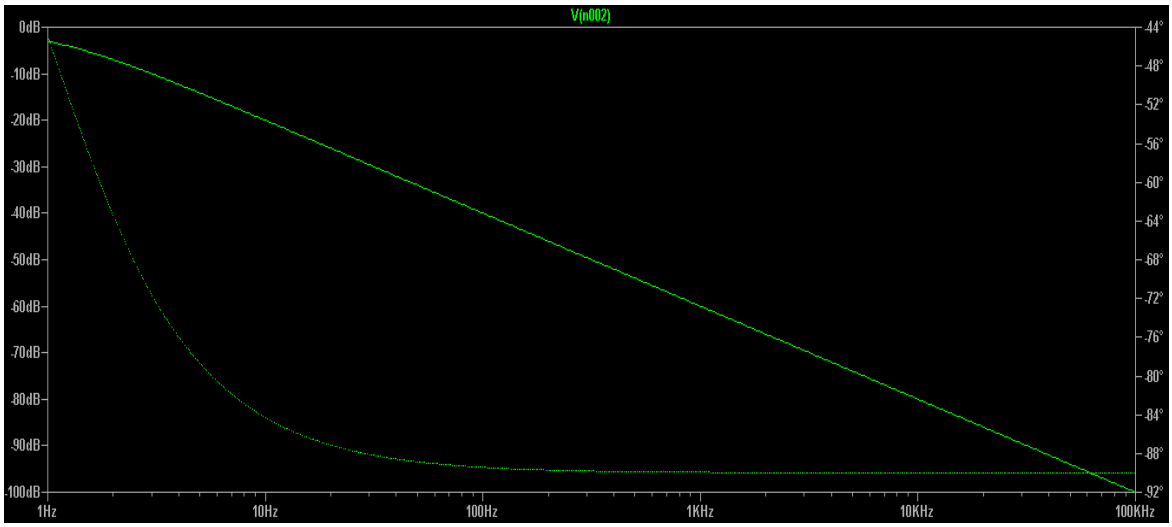

Figura 4.20

Notiamo che le due onde (4.17) e (4.18) presentano ampiezza diverse e un comportamento simile.

L'uscita non è quella che potremo aspettarci da un filtro passa-basso, in quanto dovremmo avere un segnale, si di ampiezza molto piccola, ma con una tendenza ad una onda sinusoidale, quindi con transizioni molto lente e curve, la simulazione con LTspice, rispetto a pywdf presenta solo una transizione più lenta, ma comunque si allontana dalla risposta ideale del filtro.

L'ultimo test verrà effettuato per $f_c = 5000$ Hz e $f_T = 0.001$ Hz:

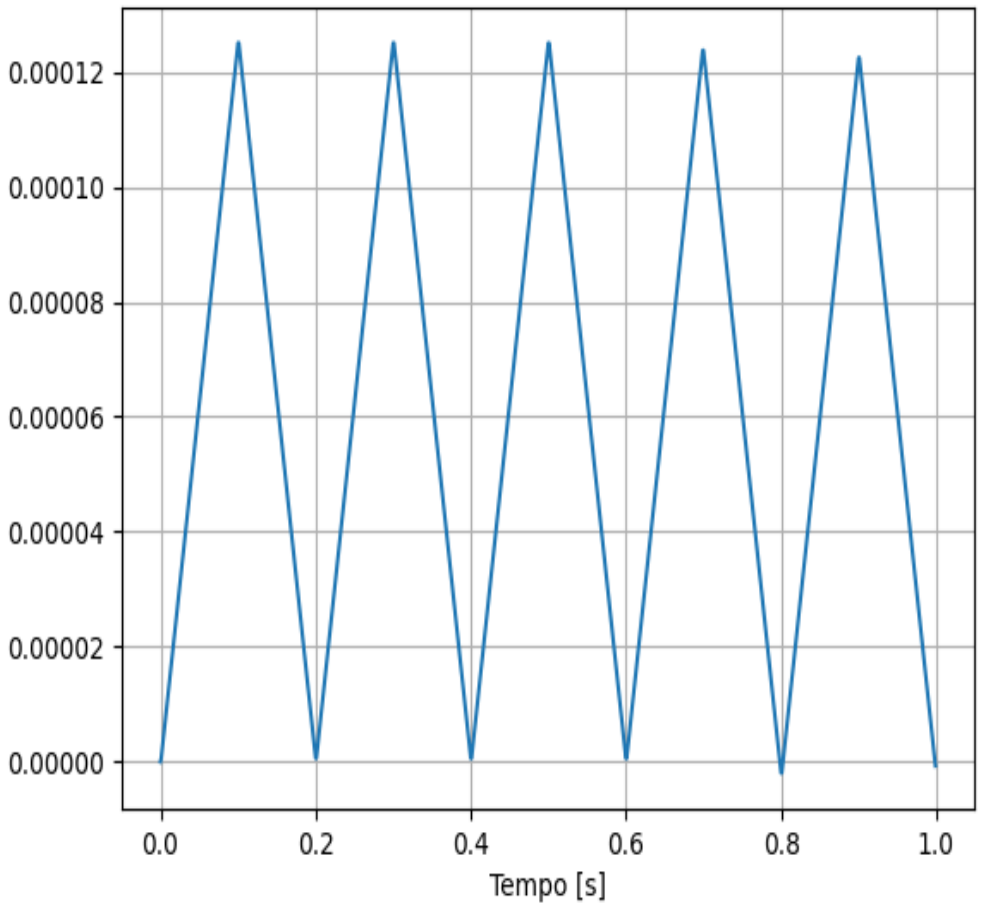

Figura 4.21

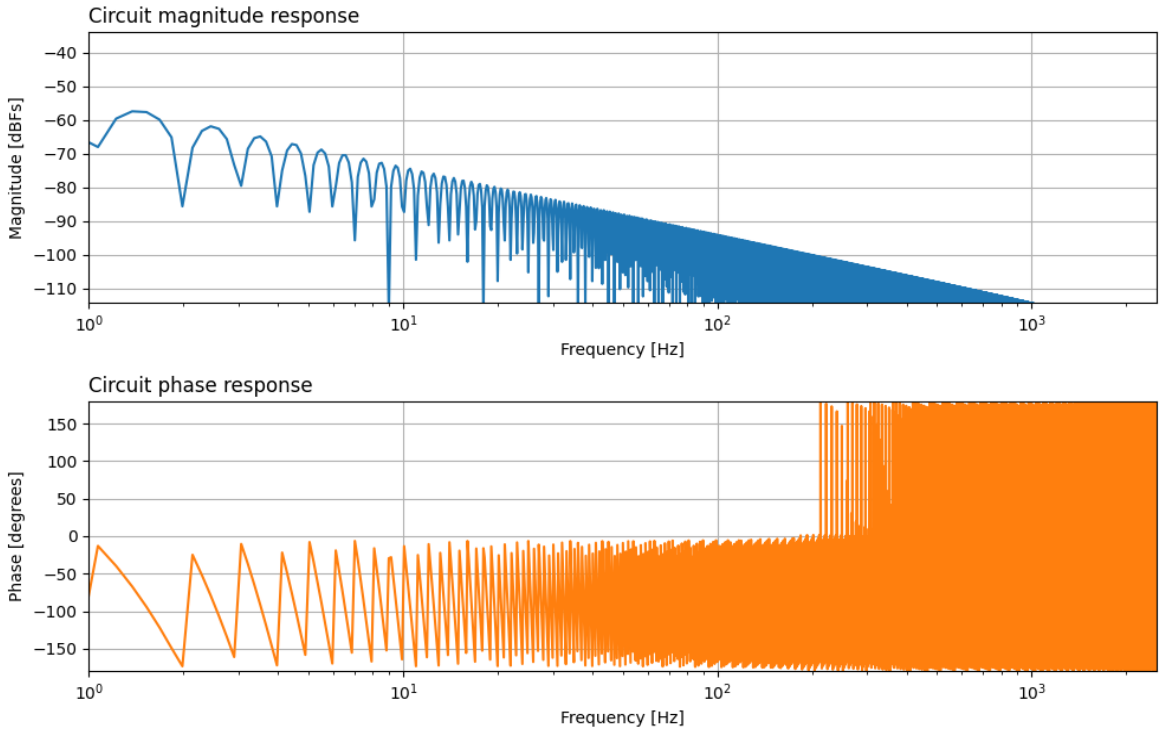

Figura 4.22

Per una frequenza di taglio così bassa rispetto alla frequenza di campionamento, notiamo una completa distorsione della risposta del filtro e pertanto una distorsione del segnale in uscita.

Valutiamo i risultati di LTspice:

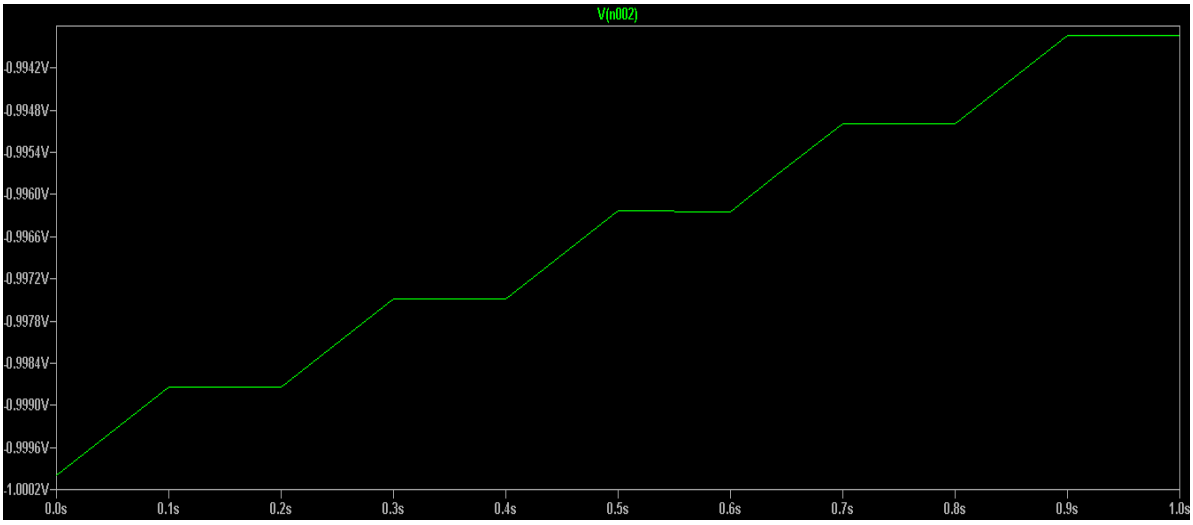

Figura 4.23

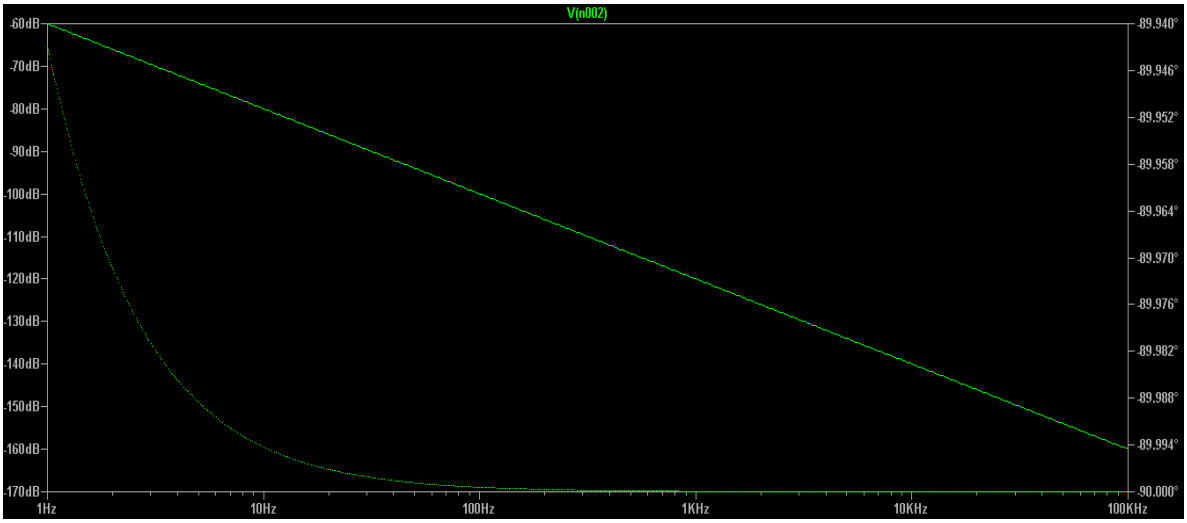

Figura 4.24

Notiamo che la risposta del filtro, sia in modulo che fase da LTspice, non risulta distorta; Potremo avere delle prestazioni maggiori della simulazione in WDF di pywdf, ma osservando (4.21) e (4.23) notiamo in entrambi i casi dei segnai distorti.

Potremo dire che la differenza nei comportamenti dei due simulatori sta nel fatto che LTspice simula un circuito analogico, quindi lavora ad una frequenza di campionamento infinita, a differenza di pywdf in cui deve essere impostata e quindi limitata.

Dopo aver confrontato le risposte del filtro, andiamo a definire il numero di operazioni matematiche svolte da pywdf per la realizzazione dei componenti, allo scopo di rendere disponibile un confronto tra altri simulatori.

Nel circuito vengono utilizzati un generatore ideale di tensione, un resistore, un capacitore, un adattatore serie e un invertitore di polarità.

Nel codice sorgente, essendo tutti questi componenti figli della stessa classe, opereranno tutti una somma e una divisione in aggiunta alle operazioni matematiche che generano tali dispositivi.

Una resistenza, in questo modo, comporterà 1 somma e 2 divisioni, un condensatore, 1 somma, 2 moltiplicazioni e 3 divisioni, il generatore ideale 2 somme, 1 moltiplicazioni e 1 divisione, l'invertitore di polarità 2 somme e 2 divisioni, infine l'adattatore serie con 8 somme, 1 moltiplicazione e 3 divisioni, per un totale del circuito di 14 somme, 4 moltiplicazioni e 11 divisioni.

Comunemente somme e sottrazioni vengono accorpate insieme, in quanto hanno lo stesso costo computazionale, infatti si è scelto di operare in questo modo.

Componenti	Somme	Moltiplicazioni	Divisioni
Resistenza	1	0	2
Conduttanza	1	2	3
Generatore ideale	2	1	1
Invertitore di polarità	2	0	2
Adattatore serie	8	1	3
Totale	14	4	11

Tabella 1: operazioni matematiche del filtro passa-basso

4. 3 Implementazione di un raddrizzatore a singola semionda

Un raddrizzatore a singola semionda è un circuito che prevede l'utilizzo di un diodo per eliminare parte di un segnale; Per un diodo polarizzato direttamente, passerà solo la semionda positiva del segnale, al contrario per un diodo in inversa, passerà solo la semionda negativa.

Il circuito in esame prevede un generatore di tensione sinusoidale, un diodo e in serie una resistenza ed un condensatore in parallelo tra loro, ma andiamo a testare, innanzitutto, il comportamento del diodo in pywdf con un semplice circuito:

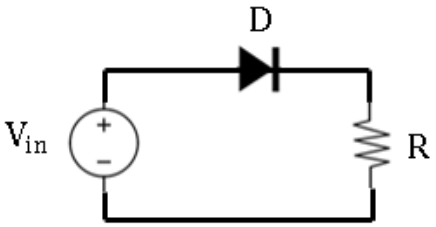

Figura 4.25: circuito di prova del diodo

Da cui la struttura ad albero:

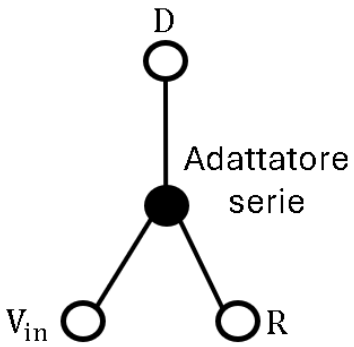

Figura 4.26: Struttura ad albero del diodo

Eseguiamo i plot dell'onda in uscita, sia da pywdf che da LTspice avendo come onda in ingresso, un segnale sinusoidale:

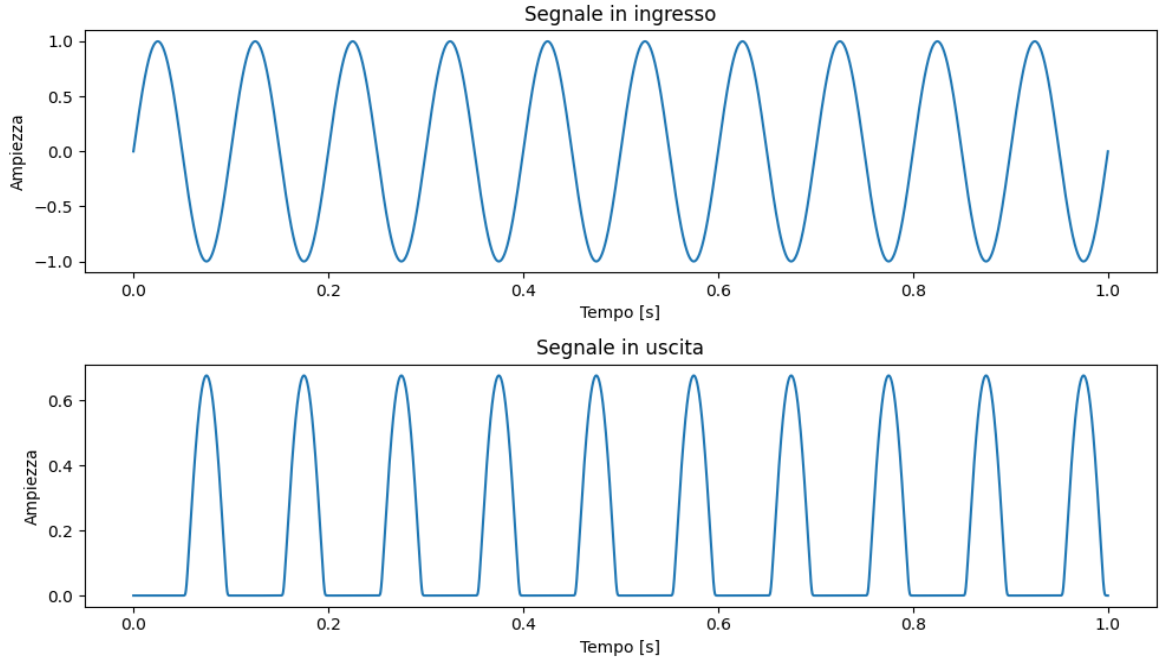

Figura 4.27

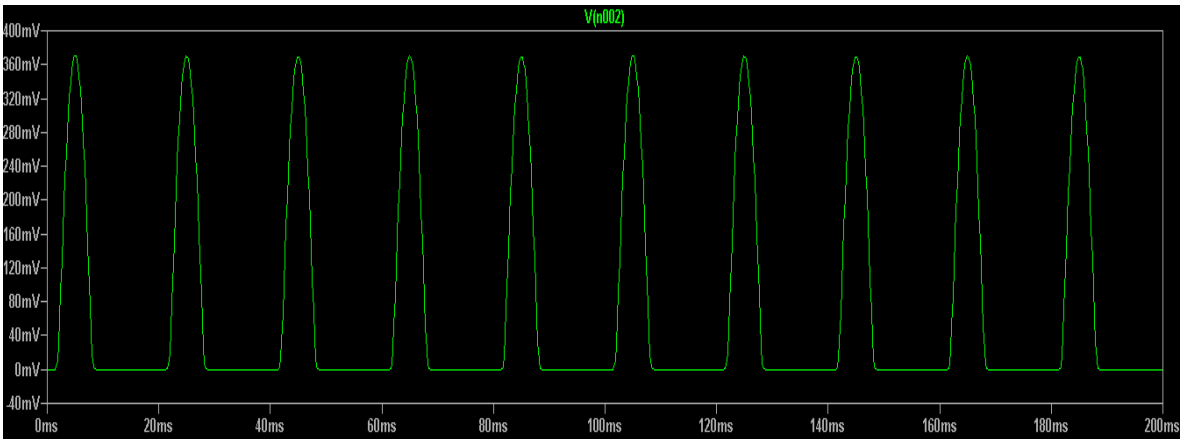

Figura 4.28

Si nota immediatamente una differenza importante tra i due grafici; Il plot realizzato da pywdf (4.27) sembra mostrare un diodo polarizzato inversamente, in quanto la semionda positiva della sinusoide viene annullata e passa solo la semionda negativa, che viene invertita, esattamente come successo per il filtro passa-basso, dall'adattatore serie.

Questo ci porta a adottare delle contromisure per sistemare il problema, cioè l'inserimento dell'invertitore di polarità, ma non per invertire l'onda in uscita dall'adattatore, ma per invertire il segnale d'ingresso uscente dal generatore di tensione:

```
I = PolarityInverter(Vs) # invertitore di polarità(elemento da invertire)
S = SeriesAdaptor(I, R)  #adattatore serie (elemento1, elemento2)
D = Diode(S, 2.52e-9)  #diodo(prossimo elemento, Vt)
```

Dopo aver inizializzato i componenti circuitali, strutturando in questa maniera il circuito, otteniamo:

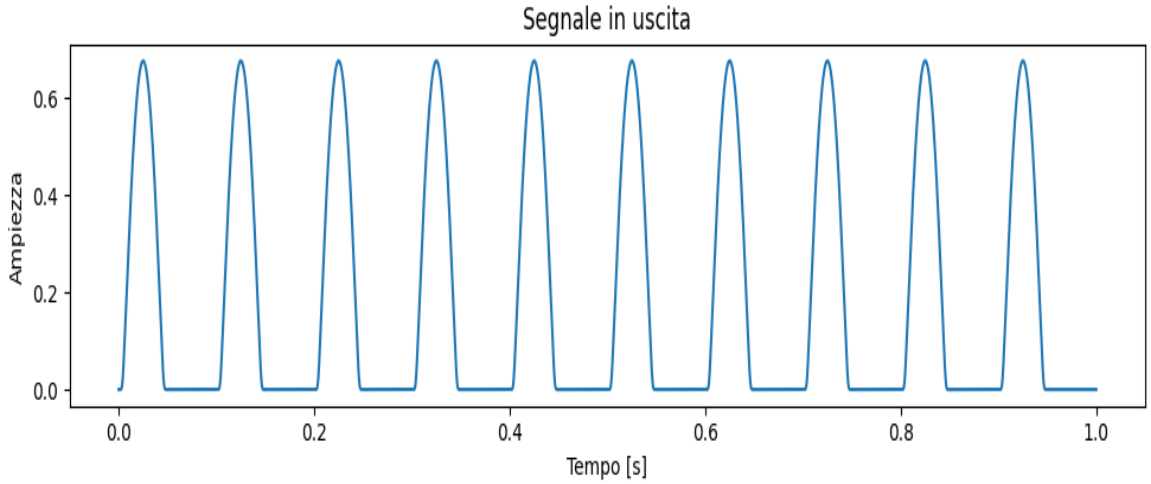

Figura 4.29

Qui possiamo notare le stesse performance tra LTspice (4.28) e pywdf (4.29), al meno dell'ampiezza.

Adesso andiamo a rappresentare il circuito raddrizzatore a singola semionda e confrontarlo:

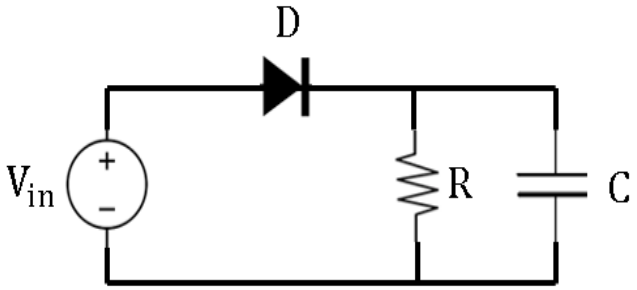

Figura 4.30: raddrizzatore a singola semionda

La sua rappresentazione ad albero sarà:

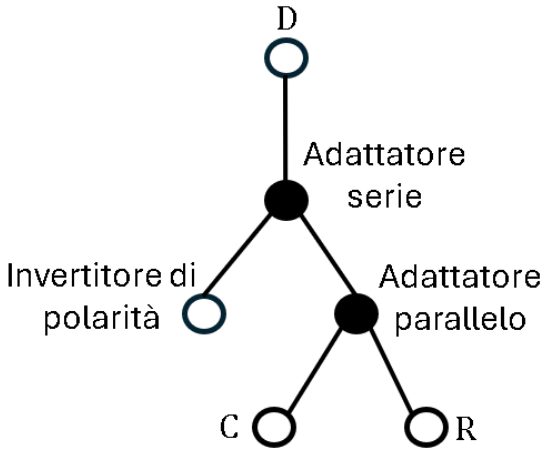

Figura 4.31: albero circuito raddrizzatore a singola semionda

La cui struttura di rappresentazione in WDF sarà:

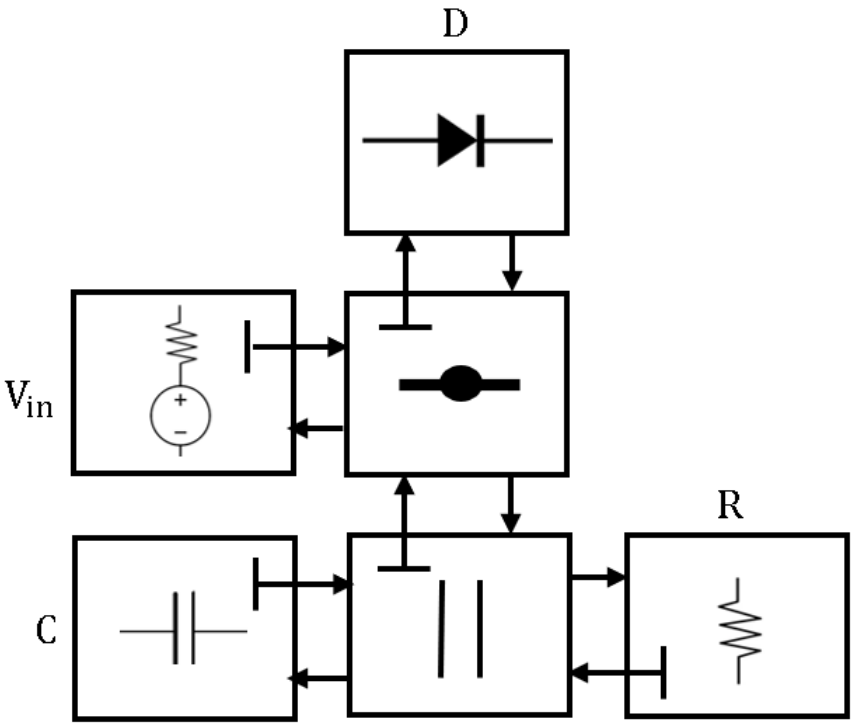

Figura 4.32: struttura adattatori WDF

Come spiegato nel capitolo 2.6 il componente non lineare deve essere messo necessariamente come nodo radice, susseguiranno l'adattatore serie tra il generatore invertito (I) e l'adattatore parallelo che racchiuderà il condensatore e il resistore, componente la cui uscita viene osservata per studiare il comportamento del circuito.

Ora vediamo i risultati dei simulatori, cominciando da pywdf e poi LTspice:

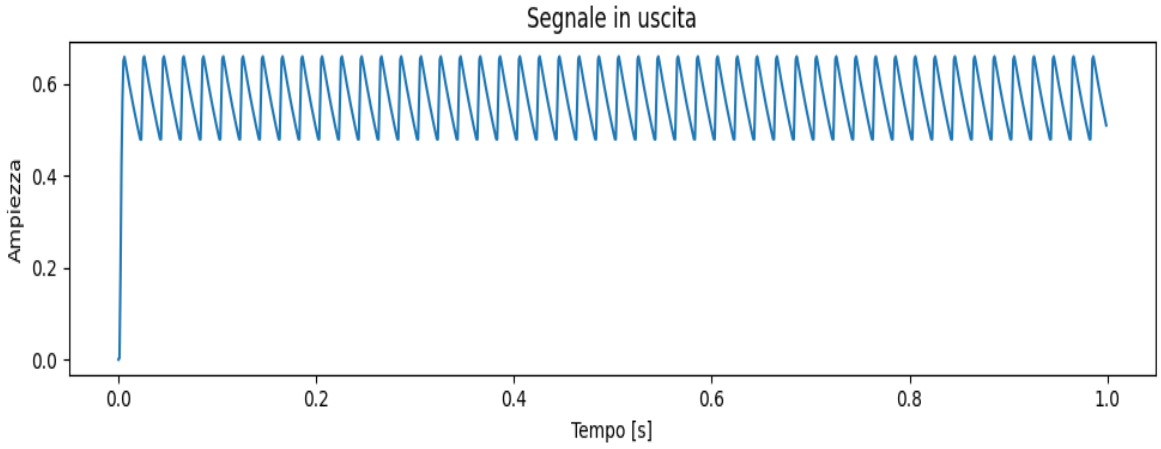

Figura 4.33

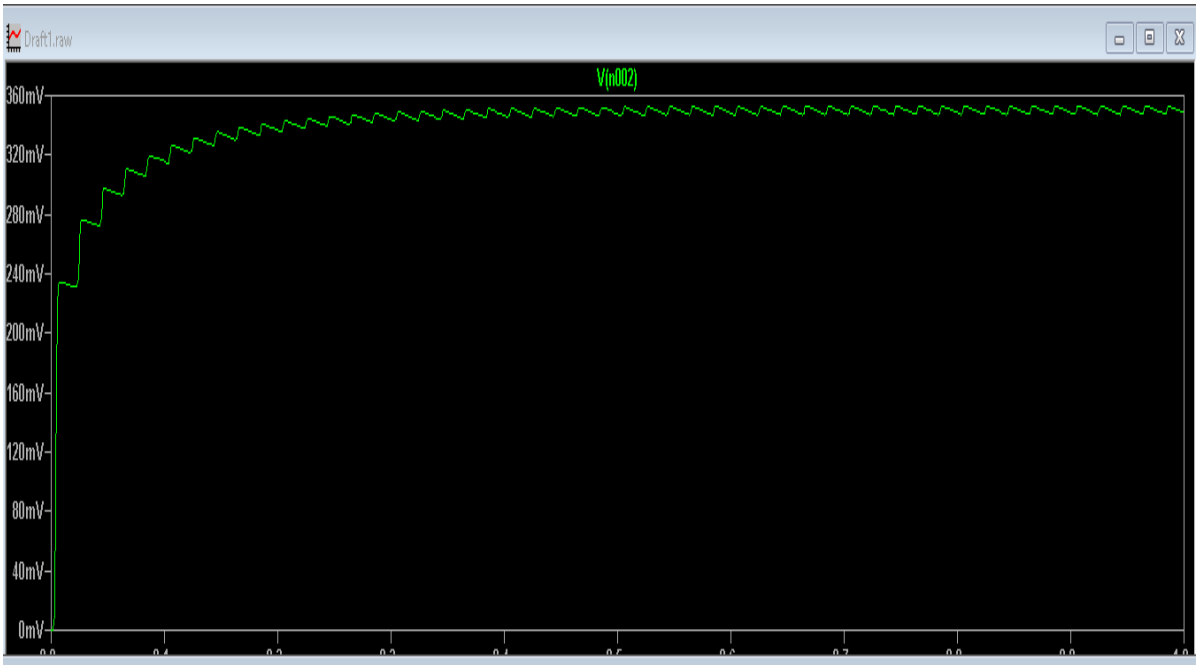

Figura 4.34

I due simulatori, a parità di circuito, presentano delle differenze importanti;

Il simulatore sotto esame presenta una forma d'onda a dente di sega, ciò suggerisce che il modello potrebbe non rappresentare accuratamente il comportamento del circuito con il condensatore, in quanto non sta smorzando in maniera efficiente le variazioni;

La forma d'onda presentata da LTspice invece, risulta quasi quadrata, il che rende il plot più vicino, almeno rispetto a pywdf, al comportamento ideale, anche se presenta comunque un effetto ripple.

Inoltre, le ampiezze sono differenti e il tempo di salita al picco, ciò è dovuto alla diversa implementazione del diodo, in quanto LTspice ci fornisce una scelta di diodi reali, tra cui il 1N4148 utilizzato nella simulazione; pywdf invece, consente solo di impostare la corrente di saturazione inversa (I_s = 2.52 nA come nel diodo 1N4148) e la tensione termica (parametro non indicato nei diodi a disposizione in LTspice)

Definito il circuito raddrizzatore e paragonato con LTspice, vediamo il numero di operazioni matematiche svolto da pywdf per la realizzazione del circuito.

Sommate alle operazioni matematiche per la realizzazione dei componenti, aggiungiamo 1 somma e 1 divisione, in quanto figlie della classe madre "baseWDF", come citato per il filtro passa-basso.

Nel circuito vengono utilizzati un generatore resistivo di tensione con 1 somma e 2 divisioni, l'invertitore di polarità con 2 somme e 2 divisioni, un resistore con 1 somma e 2 divisioni, un capacitore con 1 somma, 2 moltiplicazioni e 3 divisioni, un adattatore parallelo con 6 somme, 1 moltiplicazione e 3 divisioni, un adattatore serie con 8 somme, 1 moltiplicazione e 3 divisioni, infine abbiamo il diodo con 5 somme, 8 moltiplicazioni, 2 divisioni, un logaritmo e in base ai parametri per l'approssimazione, aggiungiamo 3 somme, 1 divisione e un esponenziale, oppure 6 somme, 3 moltiplicazioni, 1 divisione e un esponenziale, o per ultima configurazione 4 somme, 1 divisione, un logaritmo e un esponenziale.

Componente	Somma	Moltiplicazione	Divisione	Logaritmo	Esponenziale
Generatore reale	1	0	2	0	0
Invertitore di polarità	2	0	2	0	0
Resistore	1	0	2	0	0
Condensatore	1	2	3	0	0
Adattatore Parallelo	6	1	3	0	0
Adattatore serie	8	1	3	0	0
Totale componenti	19	4	15	0	0

Tabella 2: operazioni matematiche circuito raddrizzatore senza diodo

Per il diodo creiamo una tabella a parte, allo scopo di rappresentare tutti i casi previsti dal codice che variano il numero di operazioni matematiche in base alla impostazione del componente:

Diodo	Somme	Moltiplicazioni	Divisioni	Logaritmi	esponenziali
Configurazione 1	8	8	3	1	1
Configurazione 2	11	11	3	1	1
Configurazione 3	9	8	3	2	1
Totale 1	27	12	18	1	1
Totale 2	30	15	18	1	1
Totale 3	28	12	18	2	1

Tabella 3: operazioni matematiche del diodo e totale circuito raddrizzatore

Totale 1, Totale 2, Totale 3 sono ottenuti sommando Totale componenti nella tabella precedente, con i tre casi derivanti dalla configurazione del diodo, così da poter distinguere i diversi costi computazionali.

Capitolo 5

Conclusioni

Con questo lavoro abbiamo simulato dei semplici circuiti analogici in Python attraverso la libreria pywdf con il metodo dei Wave Digital Filter e attraverso LTspice, per poi confrontarli e discutere di prestazioni e differenze che possiamo trovare tra i due simulatori circuitali.

Dopo queste prove, possiamo dire che la libreria di pywdf può ancora essere migliorata, risolvendo il problema del diodo normalmente invertito e impostare l'adattatore serie in modo che non inverta il segnale, in quanto, per semplici circuiti, basta utilizzare l'invertitore per risolvere il problema, ma per circuiti complessi può risultare problematico dover inserire tanti invertitori e non commettere errori.

Il codice presenta descrizioni delle funzionalità dei tools nel file circuit.py, ma non presenta alcuna spiegazione dei componenti in wdf.py, non esplicita l'inversione dell'onda data dall'adattatore serie e non parla dell'inversione del diodo.

Gli adattatori tipo R, che sono indispensabili per la simulazione di circuiti più complessi, come in figura (2.1) e per la realizzazione di componenti non ancora implementati (come amplificatori operazionali), necessitano del supporto di altre librerie, come al link "https://github.com/jatinchowdhury18/R-Solver" per l'utilizzo di adattatori tipo R.

Risulta utile la possibilità di avere a disposizione la possibilità di settare a piacere i componenti, ma potrebbero far comodo avere parametri aggiuntivi per impostare componenti come il diodo allo scopo di realizzare un circuito reale.

Un grande vantaggio rimane sicuramente l'ambiente di lavoro, flessibile e ricco di risorse, offerto da Python che insieme alla sua grande community può portare al miglioramento della libreria e delle performance delle simulazioni anche dal punto di vista grafico.

Inoltre, a differenza di LTspice, necessita di un'ottima conoscenza del metodo WDF per implementare i circuiti, questo rende difficili le prime simulazioni.

In sintesi, si ritiene pywdf una libreria ancora in corso d'opera, che però merita di essere tenuta d'occhio, in quanto il metodo WDF risulta comunque importante per la

risoluzione in tempo reale di circuiti complessi e l'ambiente di lavoro, quale è Python, può consentire un rapido sviluppo del framework.

Bibliografia

[1] Albertini D., ANTIDERIVATIVE ANTIALIASING IN NONLINEAR WAVE DIGITAL FILTERS, con Albertini D., Bernardini A., Sarti A.,in *"Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020), Vienna, Austria"* 2021.

[2] Alshaya A., (2024), "FPGA Crystal Oscillator Circuit Emulation Based on Wave Digital Filter", con Alshaya A., Pamarti S., Papavassiliou C., in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 32, no. 01, pp. 103-115, 2024. doi: 10.1109/TVLSI.2023.3305597.

[3] Pal R., (2017), "Comparison of the design of FIR and IIR filters for a given specification and removal of phase distortion from IIR filters", in 2017 International Conference on Advances in Computing, Communication and Control (ICAC3), Mumbai, India, 2017, pp. 1-3, doi: 10.1109/ICAC3.2017.8318772.

[4] Fettweis, A., Wave digital filters: Theory and practice, in "*Proceedings of the IEEE", 74*, 270-327, 1986.

[5] Werner K. J., Virtual analog modelling of audio circuitry using Wave Digital Filters, in *"Standford Digital Repository "*, 2016.

[6] id, Resolving Wave Digital Filters with Multiple/Multiport Nonlinearities, con Abel J. S., Nangia V., Smith J. O. III, Werner K. J., in *"Center for Computer Research in Music and Acoustics"*, 2015.

[7] De Sanctis G., Sarti A., (2010), "Virtual Analog Modeling in the Wave-Digital Domain," in *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 715-727, doi: 10.1109/TASL.2009.2033637

[8] Werner K. J., Wave Digital Filter Adaptors for Arbitrary Topologies and Multiport Linear Elements, con Abel J. S, Smith J. O. III, Werner K. J., in *"Center for Computer Research in Music and Acoustics"*, 2015.

[9] Choma J., Scattering Parameters: Concept, Theory, and Applications, in *"University of Southern California"*, 2009.

[10] Galimov I., (2022), "Start Studying Machine Learning with PyCharm", JETBRAINS Blog, (Ultimo accesso 14 Ottobre).

[11] Chowdhury J., (2022), "chowdsp_wdf: An Advanced C++ Library for Wave Digital Circuit Modelling", github, (Ultimo accesso 14 Ottobre).

[12] Anthon G., PYWDF: an Open-Source Library for Prototyping and Simulating Wave Digital Filter Circuit in Python, con Anthon G., Font F., Lizarraga-Seijas X., in *"Proceedings of the 26th International Conference on Digital Audio Effects (DAFx23), Copenhagen, Denmark"*, 2023

[13] Hunter J. D., Matplotlib: A 2D Graphics Environment, in *"Computing in Science & Engineering"*, vol. 9, no. 3, pp. 90-95, 2007.