UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

cdLT in Ingegneria Informatica

Corso di Laboratorio di Automatica – A.A. 2018/19

# Progetti d'esame su regolatori PID, MEX files e filtri digitali

**Giuseppe Longo** - matricola 175983

1. Tarare un PID nel caso in cui la curva di processo sia generata dalla seguente f.d.t.

$$G(s) = \frac{0{,}65}{(150s+1)^3(250s+1)^3}$$

2. Scrivere un MEX file che, assegnato un vettore numerico, restituisce la media aritmetica e la deviazione standard. Prevedere la possibilità che, nel caso in cui la funzione venga chiamata con un unico argomento, venga restituita soltanto la media aritmetica.

3. Costruire un filtro digitale passa-basso per un segnale dato.

**1 - TARARE UN PID NEL CASO IN CUI LA CURVA DI PROCESSO SIA GENERATA DALLA SEGUENTE F.D.T.**

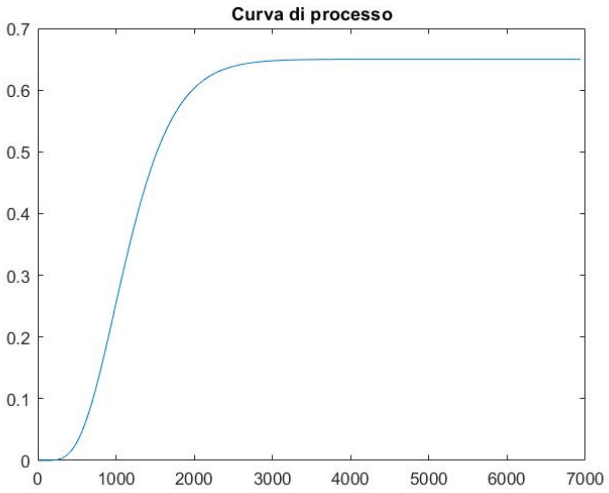
$$\boldsymbol{G(s) = \frac{0,65}{(150s+1)^3(250s+1)^3}}$$

**Introduzione e analisi della curva di processo**

Viene richiesta la taratura di un controllore $C(s)$ standard di tipo $PID$ da porre in un sistema dinamico in retroazione. Esso, in base alle varie implementazioni (che vedremo) andrà a far asservire, a regime, l'uscita $Y(s)$ al riferimento $R(s)$.

Il metodo *Zieger-Nichols (a ciclo aperto)* diviene un qualcosa di estremamente funzionale in questo contesto: se la curva di processo della risposta al gradino del sistema è di natura *sigmoidale*, sarà possibile approssimare la f.d.t. a un particolare *sistema del primo ordine* in cui sono evidenziate delle specifiche variabili. A questo punto, tramite la determinazione di quest'ultime attraverso alcuni metodi (*per ispezione grafica, delle aree, dei minimi quadrati*), sarà possibile ottenere i parametri di ogni tipo di regolatore PID (*P, I, D e le loro combinazioni).*

La traccia proposta riporta la $G(s)$ del sistema dinamico che vogliamo regolare. Quindi la curva di processo può essere elaborata campionando la risposta al gradino unitario ($T_c = 0.1\ sec$).




La curva ottenuta è certamente singmoidale: parte a tangente nulla e non presenta sovraelongazione. Quindi esisterà certamente un sistema del 1° ordine, la cui risposta al gradino avrà un andamento approssimabile a quello visto per il sistema proposto:

$$G_{appr}(s) = k\frac{e^{-s\tau}}{1+sT}$$

dove $k, \tau, T$ sono i parametri da definire attraverso i metodi di approssimazione. Come già detto, a partire da essi e tramite la tabella di *Zieger-Nichols,* sarà possibile ottenere immediatamente i valori di $k_p, T_I, T_D$ essenziali per la creazione dei PID.

### 1. Metodo per ispezione grafica

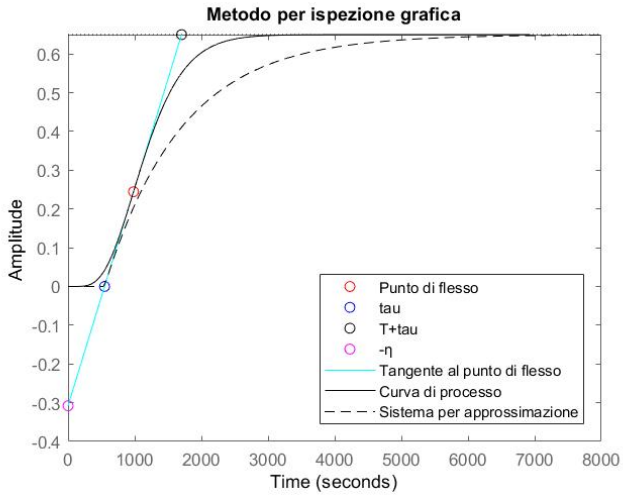
Il primo metodo a cui possiamo ricorrere è quello per "ispezione grafica". Prendiamo perciò in analisi la $G_{appr}(s)$ appena definita e andiamo ad osservarne le caratteristiche. Tramite il teorema del valore finale, possiamo scrivere

$$\lim_{t\to+inf} y_{grad}(t) = \lim_{s\to 0} s\, Y_{grad}(s) = \lim_{s\to 0} G(s) = G(0) = k\frac{e^{-s\tau}}{1+sT} = k$$

La costante $k$ coincide con il guadagno in continua del sistema e pertanto vale $k = y_{inf}$. Quindi in riferimento ai dati campionati, il valore finale al quale andrà ad asservire il gradino, sarà uguale al valore della risposta in corrispondenza dell'ultimo istante di tempo campionato (quando la risposta è già a regime).

Il tempo di ritardo τ, che in base al suo valore può ritardare o anticipare il tempo di salita, graficamente indica l'intercetta tra l'asse dei tempi e la retta tangente al punto di flesso della curva di processo.

La costante di tempo $T$, invece, sarà responsabile dell'intercetta col valore di regime sulla retta tangente al punto di flesso, nel valore temporale di $\tau + T$. Considerazioni di tipo geometrico ci permettono di valutare $T$ come $T = \frac{k}{\eta}\tau$, dove $\eta$ è intesa come il valore assoluto dell'intercetta della tangente al punto di flesso con l'asse delle ordinate.




$$G_{isp-grafica}(s) = 0{,}65\frac{e^{-547{,}3853s}}{1+1155{,}5383s}$$

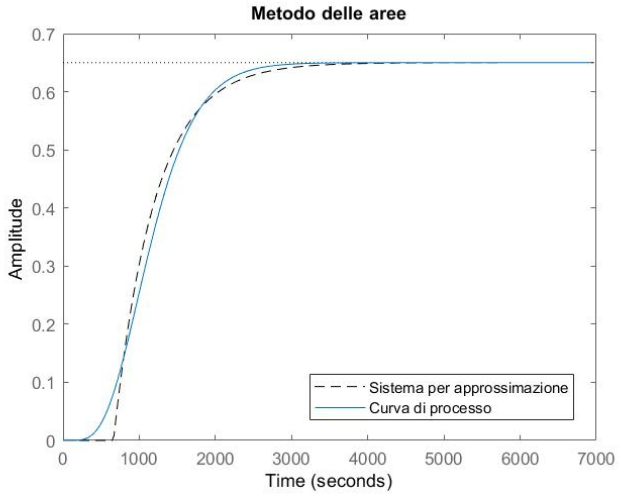
### 2. Metodo delle aree

Un altro procedimento che permette di ottenere un sistema del primo ordine che possa approssimare la curva di processo è il cosiddetto "*metodo delle aree*". Vale sempre che $k$ sia il valore finale della risposta ($k = y_{inf}$), la quale può essere graficata attraverso la funzione

$$y(t) = \begin{cases} 0 & t \le \tau \\ k\left(1 - e^{\frac{t-\tau}{T}}\right) & t > \tau \end{cases}$$

Lo scopo è sempre quello di rendere il grafico della risposta sopra citata più simile possibile a quello della curva di processo. Perciò a questo scopo, individuiamo *due aree* associate a $y(t)$: la prima coincide con quella compresa tra $t_{in} = 0$ e $t_{fin} \to +inf$; la seconda è invece l'area sottesa alla curva che va dall'istante iniziale $t_{in} = 0$ a $t_{fin} = \tau + T$, ovvero all'intercetta (sulla tangente) del valore di regime.

Lo sviluppo della dimostrazione porterà a ottenere che $T = \frac{A_2\, e}{k}$, mentre $\tau = \frac{A_1}{k} - T$.




$$G_{aree}(s) = 0{,}65\frac{e^{-662{,}0038s}}{1 + 538{,}0462s}$$

### 3. Metodo dei minimi quadrati

Il terzo e ultimo metodo "dei minimi quadrati" parte da quanto definito precedentemente ma si diversifica sull'approccio risolutivo. Difatti abbiamo sempre quella risposta $y(t)$ sperimentale ma che adesso viene vista sotto un'altra lente. Lo scopo è sempre quello di ottenere i valori di $k, \tau, T$ facendo in modo che le due curve (risposta di $y(t)$ e curva di processo) abbino un andamento simile. Il tutto allora può essere tradotto dicendo che lo *scarto quadratico medio* tra la curva di processo e la risposta $y(t)$ sia il più piccolo possibile.

Possedendo i dati campionati e i rispettivi tempi in cui essi sono stati prelevati, costruiamo un terzo vettore contenente i valori della funzione $y(t)$ in corrispondenza degli stessi tempi di campionamento.

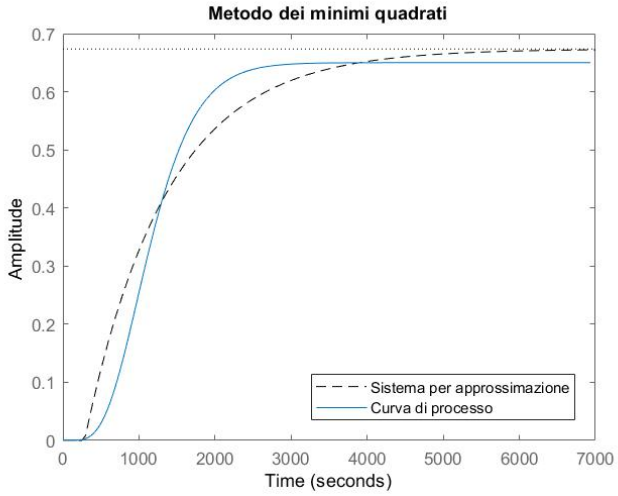
$$\left(k_c T_c, \boldsymbol{y}_{k_c}(k_c \boldsymbol{T_c})\right) \rightarrow \left(k_c T_c, \boldsymbol{y}(k_c \boldsymbol{T_c})\right)$$

Generiamo infine un *vettore degli scarti quadratici* $\varepsilon_k = y_{k_c}(k_c T_c) - y(k_c T_c)$. A questo punto, lo scopo è minimizzare l'indice di scarto quadratico al $k_c - esimo$ tempo, agendo sulla terna $k, \tau, T$ e dalla quale $y(k_c T_c)$ dipende. Vedendo il tutto dalla prospettiva che ci interessa

$$\frac{1}{N}\sum_{i=0}^{N-1} \varepsilon_k^2 \simeq 0 \rightarrow \underset{\tau, T, k}{min}$$

e quindi la terna sopra citata è detta *stimatore dei minimi quadrati*.

La risoluzione avviene in maniera algoritmica mediante l'utilizzo di una funzione *Matlab*.




$$G_{min-quad} = 0{,}67\frac{e^{-284{,}2406s}}{1 + 1077{,}0123s}$$

Si nota immediatamente come questo metodo non risulti particolarmente efficace per il problema proposto: la $y(t)$ presenta una tangente al flesso più piatta e questo, a regime, determina un netto discostamento dal valore finale della curva di processo (a differenza di quanto visto con metodo grafico e delle aree).

**Parametri di Ziegler-Nichols (a ciclo aperto)**

Una volta stimato il sistema del 1° ordine e quindi la terna $k, \tau, T$, agiamo facendo quanto preannunciato nell'introduzione (e che dà un senso a quanto fatto finora). Difatti, tramite i parametri della $G_{appr}(s)$ ottenuti, possiamo ricavare quelli necessari a tarare i vari tipi di regolatori standard PID. Vedremo a titolo di esempio, la realizzazione di P, PI e PID (anche PI-D come particolarizzazione di quest'ultimo). Il tutto è riassunto nella semplice e chiara relazione tabellare sottostante.

Per quanto riguarda l'esercizio proposto, scegliamo di utilizzare per il resto della simulazione, i parametri dedotti tramite il metodo delle aree, che permettono di ottenere un sistema del 1° ordine chiaramente più prossimo alla curva di processo rispetto a quanto ottenuto con gli altri.

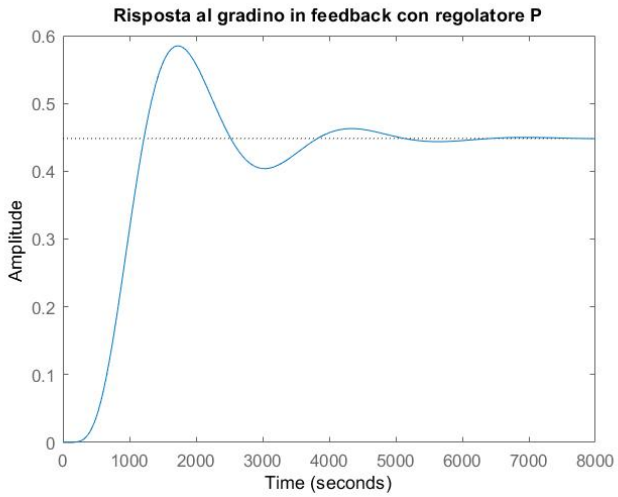
| REGOLATORE | $k_p$ | $T_I$ | $T_D$ |
|---|---|---|---|
| **P** | $\frac{T}{k\tau}$ | - | - |
| **PI** | $0{,}9\ \frac{T}{k\tau}$ | $3{,}3\ \tau$ | - |
| **PID** | $1{,}2\ \frac{T}{k\tau}$ | $2\ \tau$ | $0{,}5\ \tau$ |

| REGOLATORE | $k_p$ | $T_I$ | $T_D$ |
|---|---|---|---|
| **P** | 1,25 | - | - |
| **PI** | 1,12 | 2184,6 | - |
| **PID** | 1,5 | 1324 | 331 |

$$G_{aree}(s) \rightarrow \begin{cases} k = 0{,}65 \\ \tau = 662 \\ T = 538{,}04 \end{cases}$$

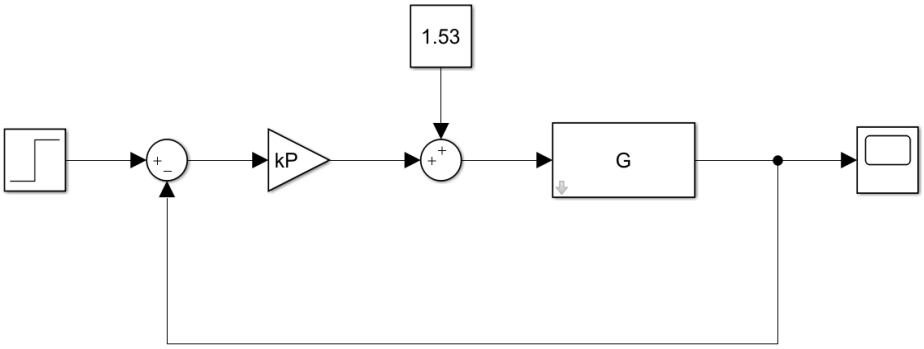
## 1. Regolatore P

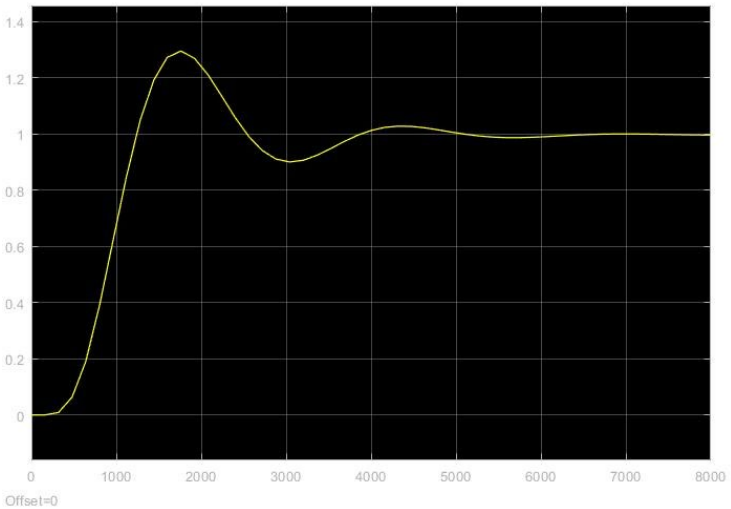
L'algoritmo di controllo più semplice è il *regolatore proporzionale* P: $C_P(s) = k_p$




Il gradino passato come riferimento è unitario, quindi è evidente l'errore di inseguimento presente. La sovraelongazione, invece, è certamente dovuta a poli complessi e coniugati che si vengono a creare a ciclo chiuso e quindi a causa della retroazione (la $G(s)$ ha poli reali e $k_p$ è costante).

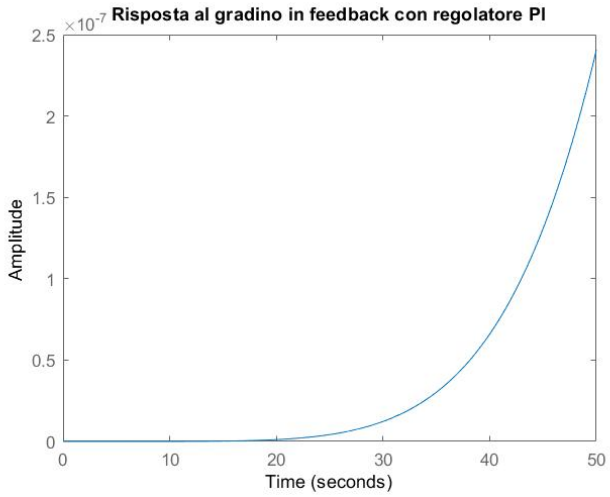
Se non possiamo attenuare l'andamento pseudo-oscillatorio, qualcosa si può fare per quanto riguarda l'errore di inseguimento. La procedura è definita nel gergo *reset manuale* e a transitorio esaurito, rende l'inseguimento nullo. Tecnicamente la soluzione prevede l'inserimento del *segnale di Bias* $u_b$ come disturbo di carico che, nell'ipotesi in cui il guadagno statico abbia valore finito, vale $u_b = \frac{1}{G(0)} = \frac{1}{0{,}65} = 1{,}53$.



## 2. Regolatore PI

Un regolatore di questo tipo unisce all'azione proporzionale quella *integrativa.* A questo punto il controllore assume una struttura interna più complessa, definita come $C_{PI}(s) = k_p \frac{1+sT_I}{s\,T_I} = k_p(1 + \frac{1}{s\,T_I})$.



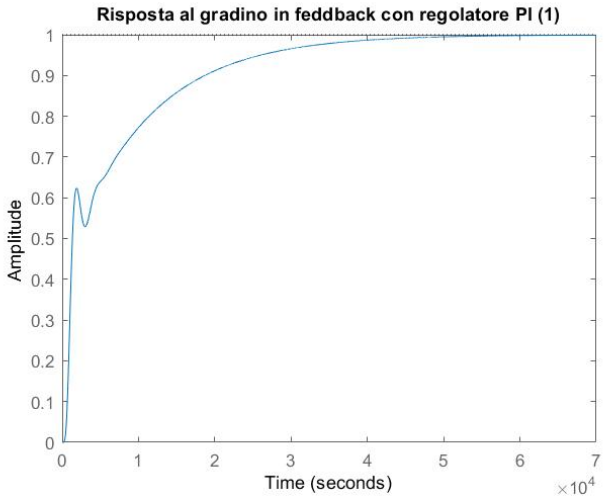

Il grafico che otteniamo non è certamente quanto poteva essere previsto: i parametri di Ziegler-Nichols ottenuti non bastano a mantenere stabile il sistema.

Il problema sta tutto dietro allo zero $z = -\frac{1}{T_I}$ detto anche *stabilizzatore.* La sua azione è cruciale, in quanto incide sulla velocità di integrazione e se scelto piccolo porta a un accumulatore "morbido". Allo stesso tempo, la necessità di far collassare presto la spezzata a un valore costante, ci porta a dover spingere lo zero in AF e quindi fargli assumere un valore tendenzialmente più grande Quindi la scelta di $T_I$ deve essere fatta cercando un compromesso in base a ciò che si vuole ottenere.

Nel caso che stiamo analizzando, certamente non vi è alternativa: lo zero deve essere tirato più in BF cosicché il sistema retroazionato possa risultare BIBO-stabile. Inoltre, più portiamo all'origine degli assi il valore dello zero, più corta sarà la banda passante (il regolatore si comporta come un filtro) e maggiori saranno tempo di salita e tempo di assestamento.

Definisco una costante moltiplicativa $a$ alla quale farò assumere sperimentalmente dei valori progressivi con approssimazione decimale di due cifre. Essa andrà ad amplificare $T_I$ e lo zero sopra identificato assumerà il valore di $z = -\frac{1}{a\,T_I}$. Lo scopo è ottenere il minimo valore di $a$ tale per cui lo zero possa spostarsi del minimo valor possibile per avere una risposta stabile del sistema (e poche conseguenze in termini di tempo di assestamento).

Il tutto può essere calcolato con un semplice script Matlab che ci porta a considerare $a = 2{,}18$.



## 3. Regolatore PID

Il PID unisce l'azione di tutte le componenti regolative, ovvero proporzionale, integrale e derivativa. Quest'ultima infatti, è detta anche *predittiva* in quanto approssima il futuro controllo grazie all'andamento della derivata, permettendo quindi prestazioni migliori. Nel dominio di Laplace avremo

$$C_{PID-IDEALE}(S) = k_p\left(1 + \frac{1}{s\,T_I} + s\,T_D\right) = k_p\left(\frac{s^2 T_I T_D + sT_I + 1}{sT_I}\right)$$

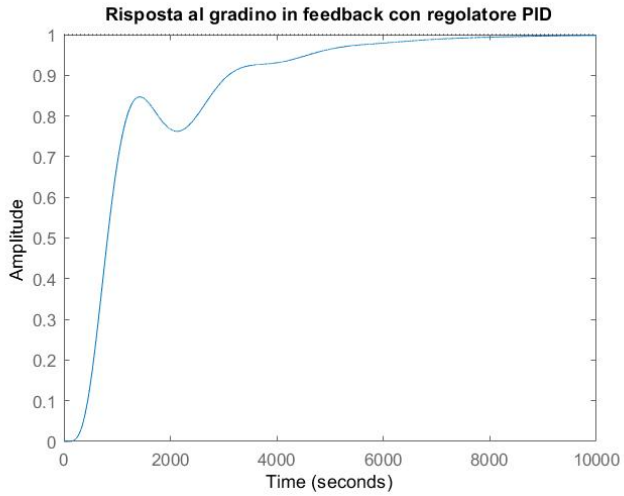
È evidente il perché parliamo di PID *ideale*: la presenza di due zeri e un solo polo rende il PID così definito non fisicamente realizzabile. Il problema deriva difatti dalla componente che porta a conseguenze di idealità per tutto il regolatore.

Per andare incontro a questa situazione, si parte dal fatto che l'azione derivativa, agendo sull'ingresso, porta con sé un rumore di misura derivante dalla catena inversa la quale in alta frequenza, può determinare conseguenze drastiche per gli attuatoti. Per questo motivo, nelle implementazioni reali, si sceglie di aggiungere in cascata un *filtro passa-basso* che inoltre rende il PID fisicamente realizzabile.

L'unico effetto indesiderato si riscontra sul filtraggio al quale l'azione derivativa è sottoposta (vengono attenuate le pulsazioni sulle quali D lavora).

$$C_{PID-REALE}(S) = k_p\left(1 + \frac{1}{s\,T_I} + \frac{s\,T_D}{1 + \frac{sT_D}{N}}\right)$$

Il parametro $N$ è inteso "variabile", nel senso di poter assumere un valore (di solito tra 8 e 30) affinché la banda passante venga resa più o meno ampia e quindi la stessa azione del filtro (in relazione al rumore di misura presente).




### Regolatore PI-D

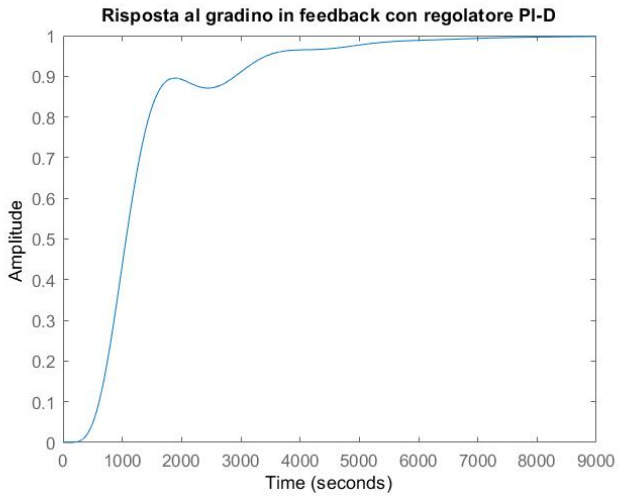
L'azione derivativa introdotta col PID agisce, per definizione stessa, sull'errore di inseguimento e la stessa cosa vale sia per P che per I (l'ingresso al regolatore è proprio $e(t)$).

Consideriamo di avere un segnale di riferimento costante a tratti e che può assumere un valore diverso per i vari istanti di tempo processati. Il salto da un valore all'altro non ha alcun effetto sulla componente integrativa e tantomeno su quella proporzionale. Discorso a parte va fatto per l'azione derivativa: i bruschi

cambiamenti di segnale comportano a rapidi spostamenti degli attuatori, che possono così bruciarsi e quindi guastare l'intero sistema di controllo.

Si parla quindi di *calci derivativi* e si presentano con bruschi variazioni di segnale in intervalli di tempo molto piccoli $\left(\frac{de}{dt} \to inf\right)$. Ad esempio, il PID sopra descritto ne presenta uno abbastanza prominente.

Quindi l'approccio risolutivo al problema è introdotto con i PI-D. La strategia prevede di portare l'azione derivativa sulla catena inversa, diminuendo l'impatto della variazione di segnale che viene prima processato dalle componenti P e I. Questa scelta può comportare comunque alcuni peggioramenti nella risposta, poiché la predizione non viene svolta in relazione all'errore di inseguimento ma su $y(t)$.




Nel caso corrente, i vantaggi dalla implementazione PI-D rispetto al PID sono più che evidenti: la spezzata diventa meno "aggressiva" e migliora il tempo di assestamento, mentre quello di salita rimane pressoché invariato.

**Regolatore PI-D digitale**

Tutte le varie implementazioni dei regolatori standard viste fino ad ora, fanno riferimento a un sistema in retroazione analizzato a tempo continuo. Quindi viene naturale chiedersi come realmente sia l'approccio risolutivo nella realtà e quindi a tempo discreto.

Partiamo dalla definizione di PI-D a tempo continuo, che ha la seguente forma.

$$u(t) = k_p e(t) + \frac{k_p}{T_I}\int_0^t e(\lambda)d\lambda - k_p T_D \frac{dy}{dt}$$

Ovviamente, prima di avviarci alla sua discretizzazione, occorre definire un *tempo di campionamento.* Per calcolarlo, si parte dal sistema PI-D analogico e si ottengono tempo di assestamento, tempo di salita e banda passante (in relazione sempre alla risposta al gradino). Una delle metodologie prevede

$$T_c \leq \frac{t_s}{40} \to T_c \leq 129{,}02\ sec$$

Decidiamo allora di scegliere $T_c = 50\ sec$.

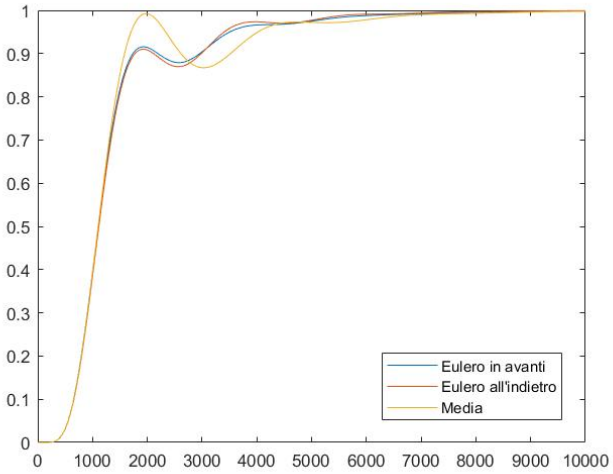
A questo punto possiamo riscrivere il segnale di controllo del PI-D $u(t)$ nel dominio discreto. È possibile dimostrare che questo sia uguale a un'equazione alle differenze del tipo

$$u_{k+1} = k_p e_{k+1} + \frac{k_p}{T_I} T_c(a_1 e_k + a_2 e_{k+1}) + I_k + b_1 D_k + b_2(y_{k+1} - y_k)$$

Dato per certo di avere la terna $k_p, T_I, T_D$ ottenuta con il paradigma Ziegler-Nichols, gli unici e nuovi parametri da dover calcolare sono $a_1, a_2, b_1, b_2$. Essi ovviamente cambiano in relazione al tipo di discretizzazione adottata, che può essere *Eulero in avanti* (il valore corrente è dato come rapporto incrementale, quindi il sistema è ritardato di un istante per essere realizzabile) o *Eulero all'indietro* (il valore corrente è dato come "rapporto decrementale" tra il valore a $kT_c$ e quello a $(k-1)T_c$) o addirittura come media tra i due (metodo *Tustin*).

| DISCRETIZZAZIONE | $\boldsymbol{a_1}$ | $\boldsymbol{a_2}$ | $\boldsymbol{b_1}$ | $\boldsymbol{b_2}$ |
|---|---|---|---|---|
| **Eulero in avanti** | 0 | 1 | $1 - \frac{N\,T_c}{T_D}$ | $-N\,k_p$ |
| **Eulero all'indietro** | 1 | 0 | $\frac{T_D}{N\,T_c + T_D}$ | $-\frac{N\,k_p\,T_D}{N\,T_c + T_D}$ |
| **Tustin (media)** | 0,5 | 0,5 | $\frac{T_D - 2NT_c}{T_D + 2NT_c}$ | $-\frac{2\,N\,k_p\,T_D}{T_D + 2NT_c}$ |

| DISCRETIZZAZIONE | $\boldsymbol{a_1}$ | $\boldsymbol{a_2}$ | $\boldsymbol{b_1}$ | $\boldsymbol{b_2}$ |
|---|---|---|---|---|
| **Eulero in avanti** | 0 | 1 | $-0{,}51$ | $-15$ |
| **Eulero all'indietro** | 1 | 0 | $0{,}99$ | $-14{,}95$ |
| **Tustin (media)** | 0,5 | 0,5 | $0{,}98$ | $-29{,}82$ |



**Parametri di Ziegler-Nichols (a ciclo chiuso)**

A inizio trattazione è stato specificato come l'approccio adottato per la definizione dei PID, faceva appello al fatto che la curva di processo fosse di natura sigmoidale. Quindi la stima del sistema del primo ordine è stata attuata tramite un'analisi *a catena aperta*.

Ma nel caso in cui la curva di processo non fosse stata propriamente sigmoidale, l'unica metodologia possibile per la definizione del PID si sarebbe basata su un'implementazione in retroazione negativa del sistema. Anche qui si ha a disposizione un'immediata tabella di Ziegler-Nichols in cui $k_p, T_I, T_D$ sono tarati in base a specifici valori ottenuti in funzione del *margine di guadagno* e dell'*ultimate frequency*.

Nella realtà questo paradigma non è mai utilizzato, in quanto genericamente porta a delle conseguenze importanti su tutto il sistema di controllo, col rischio che venga bruciato.

Per i motivi appena spiegati, si decide di non effettuarne la trattazione.

```c
/*
 * 2 - DATO UN VETTORE NUMERICO, RESTITUIRE LA MEDIA ARITMETICA E LA DEVIAZIONE
 STANDARD.
 * NEL CASO DI IN CUI LA FUNZIONE VENGA CHIAMATA CON UN UNICO ARGOMENTO,
 RESTITUIRE SOLAMENTE LA MEDIA ARITMETICA.
 *
 */

#include "mex.h"
#include "math.h"

/* Macro che assegna i puntatori dei risultati */
#define MEDIA_ARIT_RIS plhs[0]
#define DEVIAZ_STD_RIS plhs[1]

/* Funzione che implementa la media aritmetica */
static void media_arit(double *risultato, double *vettore, int num_righe, int
num_colonne) {
    int i,length;
    if (num_righe==1)
        length = num_colonne;
    else
        length = num_righe;
    for (i=0; i<length; i++) {
        *(risultato) = *(risultato) + *(vettore+i);
    }
    *(risultato) = *(risultato)/length;
    return;
}//media aritmetica

/* Funzione che implementa la deviazione standard */
static void deviaz_std(double *risultato, double *media_arit, double *vettore,
int num_righe, int num_colonne) {
    double temp = 0.0;
    int i,length;
    if (num_righe==1)
        length = num_colonne;
    else
        length = num_righe;
    for (i=0; i<length; i++) {
        temp = temp + pow((*(vettore+i)-*(media_arit)),2);
    }
    *(risultato) = sqrt(temp/length);
    return;
}//deviazione standard

/* Funzione di interfacciamento con l'ambiente Matlab */
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]) {
    // Puntatori locali agli argomenti della funzione e al risultato
    double *media_arit_ris, *deviaz_std_ris;
    double *vettore;
    // Controllo su numero argomenti in uscita e in entrata
    if(nrhs != 1)
```

```
        mexErrMsgTxt("E' richiesto un vettore numerico come argomento.");
    if(nlhs > 2)
        mexErrMsgTxt("Lo script ritorna due soli risultati.");
    // Verifica sull'argomento in input, che deve essere un vettore
    mwSize num_righe = mxGetM(prhs[0]);
    mwSize num_colonne = mxGetN(prhs[0]);
    if (num_righe>1 && num_colonne>1)
        mexErrMsgTxt("L'argomento in input non può essere una matrice.");
    // Allocazione dei puntatori in uscita definiti nella macro iniziale
    MEDIA_ARIT_RIS = mxCreateDoubleMatrix(1,1,mxREAL);
    DEVIAZ_STD_RIS = mxCreateDoubleMatrix(1,1,mxREAL);
    // Le variabili locali in uscita vengono fatte puntare alle locazioni di
    memoria precedentemente definite
    media_arit_ris = mxGetPr(MEDIA_ARIT_RIS);
    deviaz_std_ris = mxGetPr(DEVIAZ_STD_RIS);
    // La variabile locale in entrata viene fatta puntare alla locazione di
    memoria in input
    vettore = mxGetPr(prhs[0]);
    // Esecuzione delle funzioni interne di media aritmetica e deviazione
    standard
    media_arit(media_arit_ris,vettore,num_righe,num_colonne);
    deviaz_std(deviaz_std_ris,media_arit_ris,vettore,num_righe,num_colonne);
    return;
}//mexFunction
```

## 3 - COSTRUIRE UN FILTRO DIGITALE PASSA-BASSO PER UN SEGNALE DATO (*SEGNALE_16*)

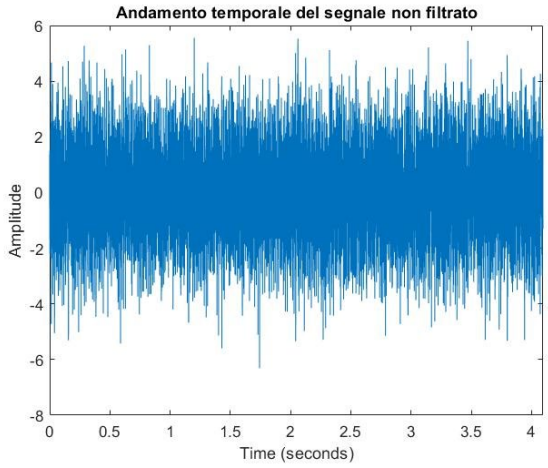
### Introduzione e analisi iniziale del segnale non filtrato

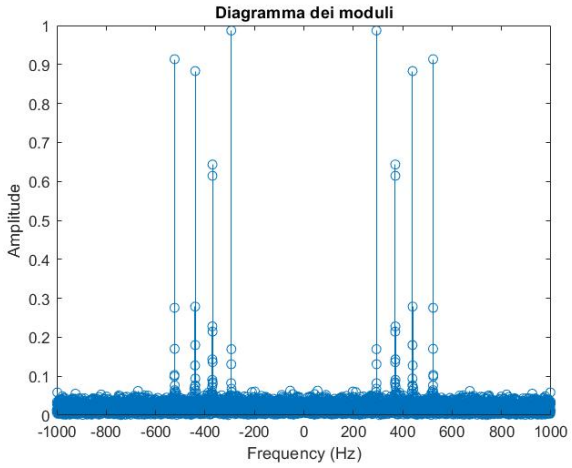
Viene richiesta la creazione di uno specifico sistema *LTI-TD* che processi un segnale passato in ingresso e ne porti in uscita lo stesso filtrato di tutte le frequenze superiori a una certa banda passante $\omega_{BW}$. Il componente così descritto prende il nome di *filtro passa-basso,* la cui implementazione avverrà sulla base di diverse tipologie.

Innanzitutto, analizziamo il segnale prima che venga elaborato. Il file fornito è del tipo $.mat$ e questo implica che siam di fronte a un segnale già campionato da Matlab e tradotto in un formato binario proprietario. Esso porta con sé due valori: $Fc$, o meglio $f_c$, che indica la *frequenza di campionamento* $f_c = 2000\ Hz = 2\ kHz$ e il vettore $x$ che contiene i valori del segnale analogico originario per ogni istante di campionamento.

Per coerenza col *teorema fondamentale del campionamento*, le uniche frequenze che potremo analizzare sono quelle poste nell'intervallo $[-1\ kHz, 1\ kHz)$, evitando così problemi di *aliasing.* Invece, dal vettore $x$, possiamo estrarre la lunghezza dello stesso e così avere il numero di campioni che effettivamente si hanno a disposizione.

È chiaro che per una corretta applicabilità della $DFT$ e così ottenere il diagramma delle ampiezze, la sequenza della trasformata deve essere finita e quindi basarsi su $N$ istanti in numero più vicino possibile a quello dei campioni $x$. Si effettua perciò un $padding$ al vettore dei dati tramite un insieme di zeri, affinché $N$ sia un multiplo di 2 e si abbia una complessità computazionale minima.




Analizzando il diagramma dei moduli, notiamo, in BF, la presenza di *4 toni predominanti* rispetto agli altri, ovvero di ampiezza nettamente distinguibile. Per questo motivo, il filtro passa-basso che vogliamo costruire, andrà a processare il segnale preservando le pulsazioni di frequenza minore e cercando di annullare quelle oltre la banda passante. Di conseguenza, in questo caso, essa sarà posta a $\omega_{BW} = 600\ Hz$.

Abbiamo già specificato che la creazione del filtro verrà attuata sulla base di diverse metodologie, ognuna con caratteristiche costruttive e operative diverse, e quindi con conseguenze diverse sul segnale filtrato:

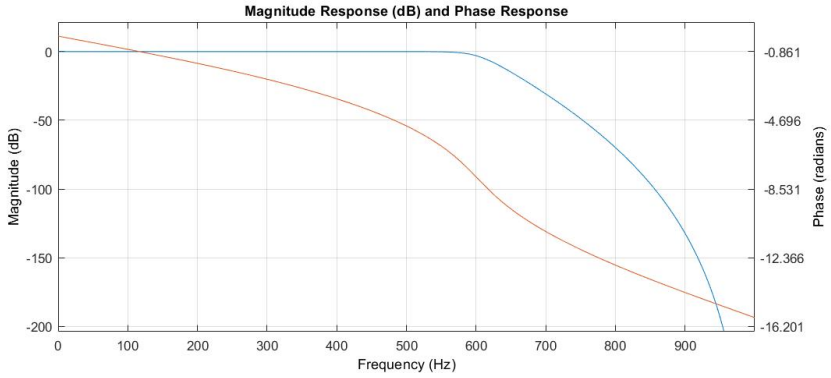
- *Filtro Butterworth*
- *Filtro Chebyshev*
- *Filtro FIR*

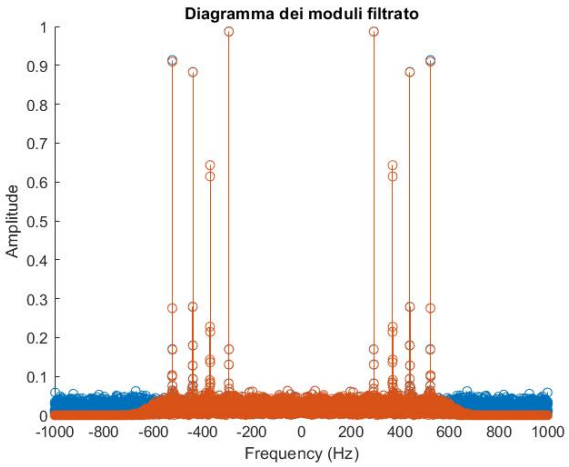
## 1. Filtro Butterworth

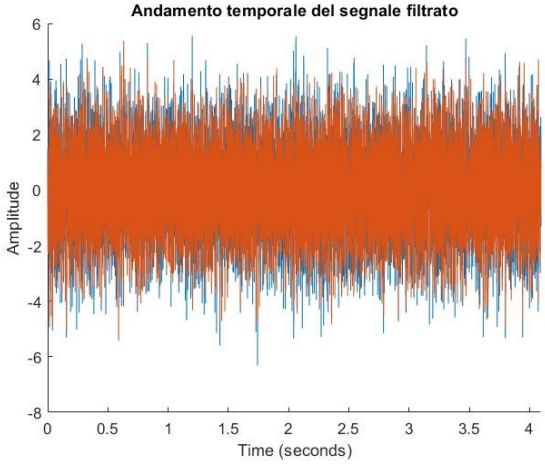
I filtri *IIR* di tipo *Butterworth* sono un sistema *LTI BIBO – stabile* caratterizzati da una funzione di trasferimento $H(s)$ con modulo nullo fino alla pulsazione di banda passante. A quel punto, la spezzata del modulo inizierà ad abbassarsi con pendenza $-20n\frac{dB}{dec}$, dove $n$ indica il grado del filtro: più aumenta n, minore sarà la media frequenza e quindi più immediato "l'abbattimento" delle pulsazioni dalla banda stop. Per contro, analizzando il filtro in termini discreti (*trasformata-Z di H(s)*), aumenterà il numero di registri e quindi la sua complessità strutturale.

$$|H(j\omega)|^2 = \frac{1}{1 + \left(\frac{\omega}{\omega_{BW}}\right)^{2n}}$$

Quindi, a partire dalla $\omega_{BW}$, tutte le armoniche verranno attenuate o anche completamente annullate.



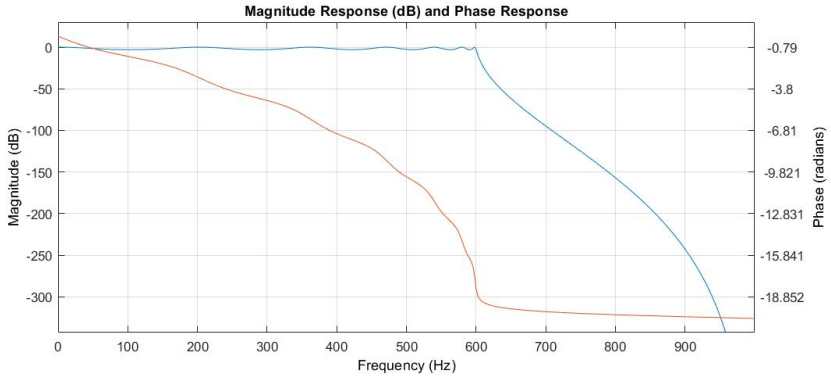
## 2. Filtro Chebyshev

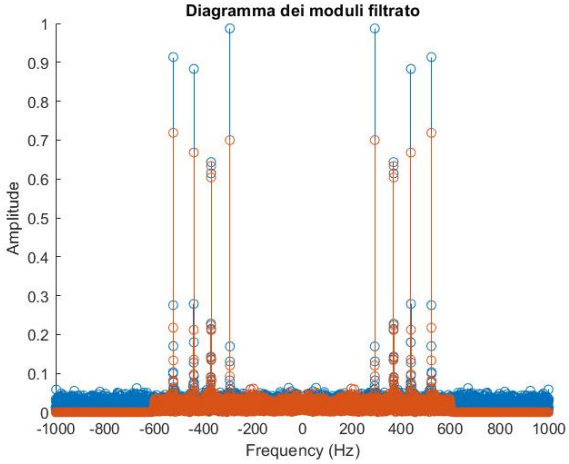
Il filtro *Chebyshev* nasce dalla riscrittura dei poli del Butterworth. Se il numero di registri deriva dai poli e zeri presenti nel dominio discreto, attuando una manipolazione matematica è possibile ottenere una f.d.t. meno "meno costosa" rispetto al filtro precedente

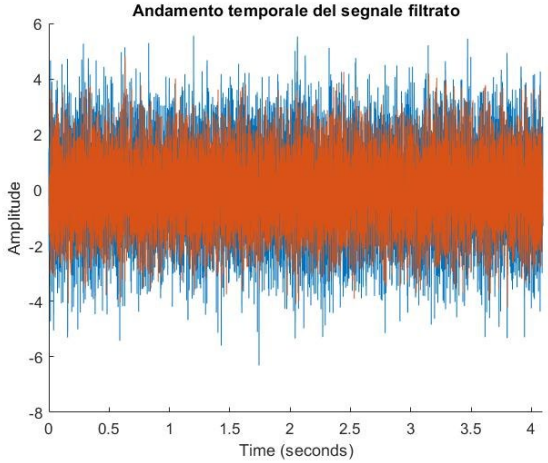
$$|H(j\omega)|^2 = \frac{1}{1 + \varepsilon\, C_n\left(\frac{\omega}{\omega_{BW}}\right)^2}$$

dove $Cn(.)$ è un polinomio di $n - esimo\ grado$ ottenuto per sostituzioni successive. Interessante è notare la presenza del parametro $\varepsilon$: esso descrive l'ampiezza della fascia in cui può oscillare la funzione modulo, provocando il fenomeno di $ripple$.

Nello specifico, con il filtro di Chebyshev, otteniamo un'azione più "aggressiva" sull'attenuazione dei moduli in banda stop, ma il prezzo da pagare è la presenza di oscillazioni in corrispondenza delle pulsazioni di banda passante: in tale intervallo le armoniche godono di guadagni diversi.




È immediatamente evidente dall'analisi in frequenza il fenomeno di ripple, così come una pendenza più pronunciata della spezzata a partire dalla banda passante. Questo comporta, ed è chiaro sul diagramma dei moduli, una più veloce attenuazione dei moduli in banda stop e anche tra le frequenze non predominanti in banda passante, cosa che assolutamente non succedeva col filtro precedente.

## 3. Filtro FIR

Nei filtri precedentemente descritti è sempre stato rispettato l'andamento desiderato sui moduli. Cosa opposta invece è successa per la fase: dai diagrammi è evidente che sia Chebyshev che Butterworth comportino dei netti rifasamenti sul segnale filtrato. Quindi affinché ciò non avvenga, deve succedere che la fase abbia un andamento lineare, del tipo $k\omega$. In questo modo tutte le armoniche verranno traslate della stessa quantità costante *k,* cosicché il filtro causi solo fenomeni di ritardo e non alteri le pulsazioni delle armoniche originali. In termini analitici, il tutto si traduce scrivendo

$$\varphi(\omega) = k\omega \rightarrow \frac{d\phi}{d\omega} = k$$

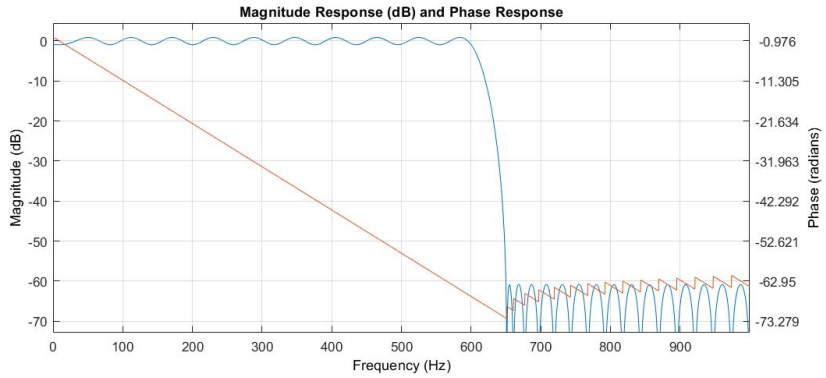
Quest'ultima derivata prende il nome di *group delay*.

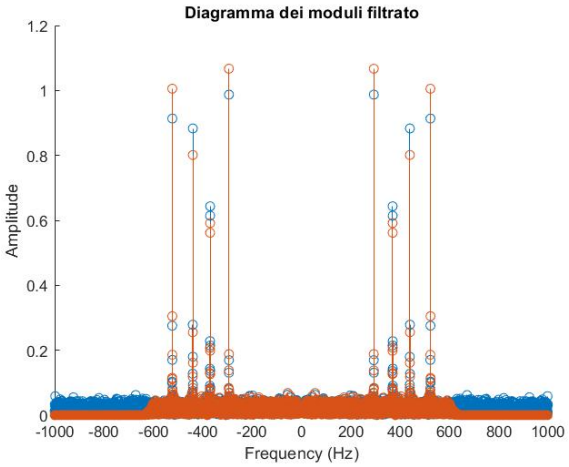
Ma un group delay costante insieme al modulo nullo nella banda passante, rispecchiano delle specifiche di costruzione di un *passa-basso ideale*, poiché non esiste alcun tipo di filtro analogico o digitale in grado di soddisfare entrambe le condizioni.

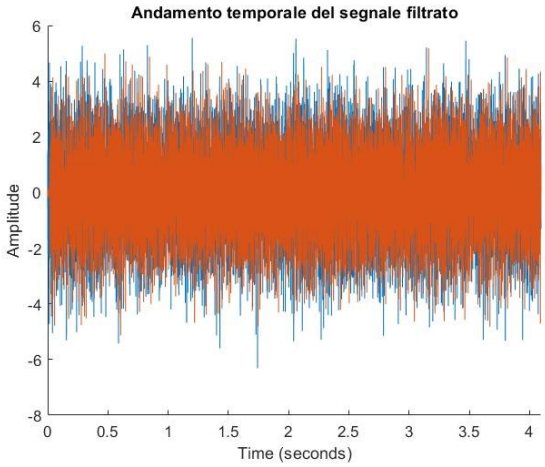
Un compromesso però può essere raggiunto attraverso i *filtri FIR*. A differenza degli IIR, essi sono caratterizzati da una risposta all'impulso finita: nel caso in cui la sequenza della risposta all'impulso del FIR sia pari o dispari, la fase avrà un andamento lineare.

$$H(z) = \sum_{k=0}^{n-1} b_k z^{-k}$$

A questo punto la domanda sorge spontanea: *perché utilizzare filtri IIR (Butterworth e Chebyshev) se i FIR offrono caratteristiche vicine all'ideale?* La risposta è alquanto semplice e prende voce da costi di implementazione dei FIR rispetto agli IIR. Difatti, uno stesso filtro passa-basso può essere ottenuto con molti meno registri se costruito, ad esempio, di tipologia Butterworth rispetto a quelli necessari per la costruzione con metodologia *FIR*.



Anche in questo caso si presenta il fenomeno del ripple in corrispondenza della banda passante, che addirittura amplifica la maggior parte dei toni ed è più accentuato sui moduli predominanti. Ancora più particolare è il fenomeno che si riscontra a partire dalla banda stop, chiamato *fenomeno di Gibbs*. Esso fa riferimento all'andamento a gobbe della spezzata ed è generato dalla presenza di zeri in alta frequenza.

**Spettrogramma**

Un altro strumento per ispezionare come agisca l'operazione di filtraggio potrebbe essere lo *spettrogramma*. L'asse delle ascisse coincide con quello dei tempi e copre l'intero intervallo di campionamento, mentre quello delle ordinate rappresenta le frequenze e in particolare tutte quelle contenute in $\left[0, \frac{f_c}{2}\right)$, espresse in $kHz$. Un quadratino chiaro indica la presenza dell'armonica avente quella frequenza nell'istante di tempo considerato; una porzione più scura ne indica l'assenza.