Università degli Studi di Padova

FACOLTÀ DI INGEGNERIA

Corso di Laurea in Ingegneria Meccanica e Meccatronica (curriculum meccatronico)

Tesi di laurea triennale

Studio e simulazione di una tecnica innovativa di condizionamento delle forze controelettromotrici non sinusoidali

Laureando:
Davide Mutterle

Relatore:
Ch.mo Prof. Mauro Zigliotto

Anno Accademico 2011–2012

Indice

1 Introduzione

Gli algoritmi della Discrete Fourier Transform (DFT), e le versioni collegate della Fast Fourier Transform (FFT), sono normalmente utilizzati per il calcolo di ampiezza e fase dell'intero spettro di un segnale. La conoscenza dello spettro potrebbe risultare di interesse per alcune applicazioni in tempo reale, come alcuni azionamenti elettrici basati sul tecniche di controllo avanzate. Generalmente, tuttavia, non è necessario conoscere l'intero spettro del segnale, bensì è sufficiente la conoscenza di una singola armonica, o di un esiguo numero di armoniche. I classici algoritmi di DFT e FFT non sono efficienti, dal punto di vista computazionale, per la misura di un singolo tono in frequenza, senza contare il notevole carico computazionale richiesto per il calcolo completo dello spettro del segnale.

La teoria dell'elaborazione numerica dei segnali, tuttavia, offre un'alternativa qualora sia necessario misurare solo un'armonica e sia richiesta una bassa complessità computazionale. L'algoritmo è noto con il nome di *algoritmo di Goertzel*, il cui dettaglio è riportato nel par. 2.2, dopo un rapido richiamo alla definizione della DFT ed alle sue proprietà dato nel par. 2.1.

1.1 Motivazioni dell'uso di Goertzel

L'algoritmo di Goertzel è molto vantaggioso rispetto alla FFT principalmente per la bassa complessità computazionale nel calcolo di una singola armonica. Si rimanda al par. 2.3 per una trattazione dettagliata.

Un altro vantaggio dell'algoritmo di Goertzel è la sua natura incrementale, che consente il calcolo "distribuito" nel tempo dell'armonica di interesse durante l'esecuzione dell'algoritmo. Nella FFT invece è richiesta l'intera finestra di acquisizione del segnale prima di procedere con i calcoli.
Nel caso dell'analisi in tempo reale delle armoniche in azionamenti elettrici, con la FFT è necessario completare tutti i calcoli in un solo periodo di campionamento. Questo rappresenta un notevole sforzo computazionale. Maggiori dettagli nel par. 2.3.

1.2 Obiettivi della tesi

- Spiegare il funzionamento dell'algoritmo di Goertzel (par. 2)
- Simulare un'implementazione completa in Simulink (par. 3)
- Simulare l'algoritmo in alcuni casi di interesse (par. 4)
- Applicare l'algoritmo alle f.e.m. non sinusoidali (par. 5)

2 Descrizione dell'algoritmo di Goertzel

2.1 Principi di base della DFT

La DFT è una trasformazione lineare a tempo discreto nel dominio della frequenza, ottenuta da un segnale discreto nel dominio del tempo. La sua definizione, insieme alla Inverse Discrete Fourier Transform (IDFT), è la seguente [1, pp. 85-86]:

$$\begin{aligned} X(kF) &= \sum_{r=0}^{N-1} T_c\, x(rT_c) e^{-j2\pi kr/N}, \qquad 0 \le k,\, r \le N-1 \\ x(rT_c) &= \sum_{k=0}^{N-1} F\, X(kF) e^{j2\pi kr/N}, \qquad 0 \le k,\, r \le N-1 \end{aligned} \tag{1}$$

Nella (1), $x(r)$ e $X(k)$ sono rispettivamente il segnale campionato e la sua trasformazione nel dominio della frequenza, T è il periodo di campionamento, F è la frequenza di campionamento, N è il numero totale di campioni. Il valore assoluto di $X(kF)$ rappresenta l'ampiezza del tono alla frequenza kF. E' usuale utilizzare una versione normalizzata della DFT, imponendo $T_c = 1$ e $F = 1/N$. Si ottiene[1]:

$$\begin{aligned} X(k) &= \sum_{r=0}^{N-1} x(r) e^{-j2\pi kr/N}, \qquad 0 \le k,\, r \le N-1 \\ x(r) &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi kr/N}, \qquad 0 \le k,\, r \le N-1 \end{aligned} \tag{2}$$

E' inoltre comune definire:

$$W_N = e^{-j2\pi/N} \tag{3}$$

che porta a:

$$\begin{aligned} X(k) &= \sum_{r=0}^{N-1} x(r) W_N^{kr} \\ x(r) &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kr} \end{aligned} \tag{4}$$

Ci sono alcune importanti considerazioni da fare sulla natura della DFT, valide sia per la (1) che per la (2). In [1, p. 158], che riporta la teoria dei

[1] Dunque l'ampiezza dell'armonica reale si ottiene a partire da quella normalizzata moltiplicandola per T_c.

segnali per sistemi continui e sistemi discreti in un orizzonte più ampio di una teoria unificata, è dimostrato che il segnale nel dominio del tempo $x(r)$ ed il segnale nel dominio della frequenza $X(k)$ sono legati dalle seguenti relazioni:

1. $x(r)$ periodo di campionamento: T_c;
2. $x(r)$ numero totale di campioni: N;
3. $x(r)$ finestra temporale di acquisizione: $T_w = NT_c$;
4. $X(k)$ risoluzione in frequenza $F = \dfrac{1}{T_w}$;
5. $X(k)$ numero totale di campioni: N;
6. $X(k)$ range di frequenza: $F_w = NF = \dfrac{1}{T_c}$.

In altre parole, il segnale $X(k)$ è composto da N campioni di frequenza, con una risoluzione di F che è inversamente proporzionale alla finestra temporale di acquisizione di $x(r)$. L'intervallo di frequenze di $X(k)$ è inversamente proporzionale al periodo di campionamento T_c. E' da evidenziare dunque che una elevata risoluzione in frequenza della DFT viene raggiunta con una lunga finestra temporale di acquisizione T_w, mentre un intervallo più ampio di frequenze è ottenuto con un piccolo periodo di campionamento T_c.

Ci sono altre due importanti caratteristiche della DFT. La prima, da [1, p. 158], è che la periodicità della DFT è pari a F_w, uguale a:

$$X(-kF) = X((N-k)F) \tag{5}$$

La seconda ricorda la simmetria Hermitiana della trasformata di Fourier quando il segnale $x(r)$ è reale [1, pp. 175]:

$$X(kF) = X^*(-kF) \tag{6}$$

dove $X^*(k)$ è il segnale complesso coniugato di $X(k)$. Combinando la (5) e la (6), segue che:

$$X(kF) = X^*((N-k)F) \tag{7}$$

La (7) mostra che le frequenze misurabili sono collocate nell'intervallo $[0,\, F_w/2]$, mentre le frequenze nell'intervallo $[F_w/2,\, F_w]$ possiedono simmetria Hermitiana rispetto al primo intervallo. Ciò significa che, nella pratica, le misure di frequenza possono essere ottenute da zero fino ad una frequenza pari a $F_w/2 = 1/(2T_c)$, che è il valore noto come frequenza di Nyquist.

2.2 Teoria dell'algoritmo di Goertzel

Le basi dell'algoritmo di Goertzel si ottengono direttamente manipolando la (2). Si consideri inizialmente la seguente identità, relativa alla periodicità della DFT di $e^{j2\pi}$:

$$W_N^{-kN} = e^{(j2\pi/N)Nk} = e^{j2\pi k} = 1 \tag{8}$$

Utilizzando la (8) nella prima delle (4), segue:

$$X(k) = W_N^{-kN} \sum_{r=0}^{N-1} x(r) W_N^{kr} = \sum_{r=0}^{N-1} x(r) W_N^{-k(N-r)} \tag{9}$$

Si può ora introdurre la seguente sequenza, basata sulla (9):

$$y(n) = \sum_{r=0}^{N-1} x(r) W_N^{-k(n-r)} \tag{10}$$

La sequenza (10) è pari a $X(k)$ quando $n = N$:

$$X(k) = y(n)\big|_{n=N} \tag{11}$$

La sequenza (10) rappresenta una convoluzione[2] di lunghezza finita tra $x(n)$ e W_N^{-kn}. Essa può essere anche considerata come l'output di un sistema lineare, la cui *sequenza ponderatrice* è pari a W_N^{-kn}, come riportato in Figura 1.

Nel caso continuo la risposta di un sistema dinamico lineare tempo-invariante, a partire dalle condizioni di quiete, è data dall'integrale di convoluzione della risposta all'impulso g(t) con il segnale di ingresso x(t) e tale relazione in termini di trasformate di Laplace si riduce al prodotto algebrico delle rispettive trasformate.

Nel caso tempo-discreto, il valore (o campione) della sequenza di uscita all'istante k-esimo sia esprimibile con la sommatoria di convoluzione tra la "sequenza ponderatrice" d_k e la sequenza di ingresso e_k e che tale relazione è ancora riconducibile a un prodotto algebrico in termini di trasformata Z.

E' ora possibile utilizzare la trasformata Zeta per passare al dominio delle frequenze. Per i segnali esponenziali, la trasformata Zeta diventa:

$$Z\left[a^n\right] = \frac{1}{1 - az^{-1}} \tag{12}$$

[2]Si ricorda che la sequenza di uscita di un sistema lineare invariante a tempo discreto è la somma di convoluzione fra la sequenza di ingresso e la risposta impulsiva del sistema stesso.

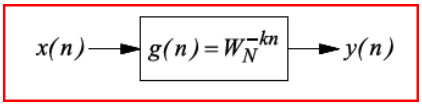

Figura 1: Sistema lineare a tempo discreto con risposta impulsiva $g(n)$ pari a W_N^{-kn}.

Pertanto, la trasformata Zeta della risposta impulsiva W_N^{-kn} è pari a:

$$H(z) = \frac{1}{1 - W_N^{-k} z^{-1}} \tag{13}$$

Moltiplicando il numeratore ed il denominatore di (13) per $1 - W_N^k z^{-1}$, si ottiene:

$$H(z) = \frac{1 - W_N^k z^{-1}}{\left(1 - W_N^{-k} z^{-1}\right)\left(1 - W_N^k z^{-1}\right)} = \frac{1 - W_N^k z^{-1}}{1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}} \tag{14}$$

L'espressione (14) rappresenta la funzione di trasferimento tra la sequenza $y(n)$ e l'ingresso $x(n)$, che restituisce il tono armonico di interesse per $n = N$, come riportato in (11).

2.3 Implementazione reale

L'espressione (14) può essere convenientemente separata in un filtro IIR ed un filtro FIR[3], come riportato nella (15).

$$H(z) = \underbrace{\frac{1}{1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}}}_{\text{IIR filter}} \cdot \underbrace{\left(1 - W_N^k z^{-1}\right)}_{\text{FIR filter}} \tag{15}$$

Si consideri dapprima il filtro IIR. Denominando $s(n)$ l'uscita del filtro IIR e $S(z)$ la sua trasformata Zeta, segue:

$$X(z) = \left(1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}\right) S(z) \tag{16}$$

[3]E' bene ricordare che un filtro IIR (Infinite Impulse Response) ha un'uscita che dipende dai precedenti valori *dell'uscita* stessa, mentre un filtro FIR (Finite Impulse Response) ha un'uscita che dipende solo da valori attuali e precedenti *dell'ingresso*.

Nel dominio del tempo, si ha:

$$s(n) = x(n) + 2\cos\left(\frac{2\pi k}{N}\right) s(n-1) - s(n-2) \tag{17}$$

Dalla (15) si ha subito:

$$s(n)\left[1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}\right] = x(n)$$

$$s(n) - 2\cos\left(\frac{2\pi k}{N}\right) s(n-1) + s(n-2) = x(n)$$

da cui la (17).

Dato che l'ingresso $x(n)$ è campionato durante il controllo in tempo reale, anche la sequenza $s(n)$ può essere calcolata in tempo reale. Detta $y(n)$ l'uscita del filtro FIR, e siccome è di interesse solo il valore di $y(n)$ per $n = N$, segue che le operazioni del filtro FIR possono essere eseguite solo quando $n = N$, ovviamente salvando $s(N)$ e $s(N-1)$. Si ottiene:

$$X(k) = y(n)\big|_{n=N} = s(N) - W_N^k s(N-1) \tag{18}$$

L'espressione (18) è nel dominio complesso, dato che W_N^k è un numero complesso. Dalla (3), si ricavano le equazioni dei due elementi che compongono un numero complesso, ossia la parte reale e la parte immaginaria. Applicando la formula di Eulero alla (3) si ottiene:

$$W_N = \cos\left(\frac{2\pi}{N}\right) - j\sin\left(\frac{2\pi}{N}\right) \tag{19}$$

Sostituendo nella (18) si ottengono la parte reale e quella immaginaria:

$$\begin{aligned} \Re\left[X(k)\right] &= s(N) - \cos\left(\frac{2\pi k}{N}\right) s(N-1) \\ \Im\left[X(k)\right] &= \sin\left(\frac{2\pi k}{N}\right) s(N-1) \end{aligned} \tag{20}$$

Spesso è più comodo rappresentare un numero complesso in modulo e fase, ricavabili direttamente dalle (20)[4].

$$\begin{aligned} |X(k)| &= \frac{\sqrt{(\Re\left[X(k)\right])^2 + (\Im\left[X(k)\right])^2}}{(0.5 * N)} \\ \angle X(k) &= atan2(\Im\left[X(k)\right], \Re\left[X(k)\right]) \end{aligned} \tag{21}$$

[4]La divisione per il fattore $\frac{N}{2}$ nel calcolo del modulo sarà chiara osservando l'esempio del par. 2.4. Nella (26) è mostrato come il modulo di un coseno di ampiezza unitaria sia pari, applicando Goertzel, a $\frac{N}{2}$. Perciò è necessario dividere il modulo calcolato con Goertzel per tale valore, così da ottenere l'ampiezza reale dell'armonica di interesse.

Lo schema a blocchi completo dell'algoritmo di Goertzel è riportato in Figura 2.

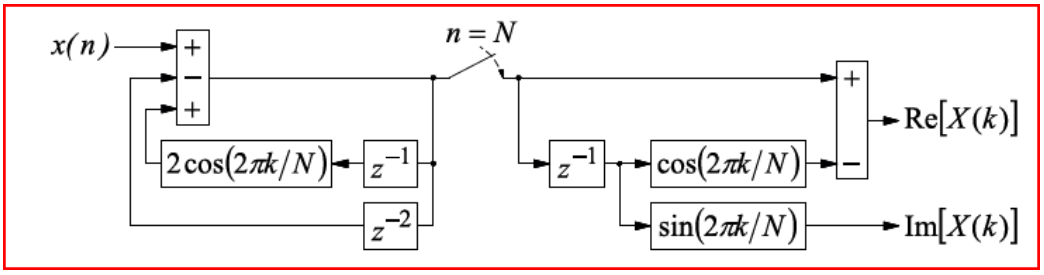

Figura 2: Schema a blocchi dell'algoritmo di Goertzel.

La complessità computazionale dell'algoritmo di Goertzel è facilmente calcolabile considerando il numero di moltiplicazioni ed addizioni per ogni iterazione della procedura. Quando $x(n)$ è un segnale reale (come nel nostro caso), la (17) mostra che ogni $s(n)$ è ottenuto mediante una moltiplicazione (se il termine $2\cos(2\pi k/N)$ è salvato come singolo coefficiente) e due addizioni. Essendo interessati a N iterazioni della sequenza, il valore di $s(N)$ è ottenuto con N moltiplicazioni e $2N$ addizioni. Il calcolo di $\Re[X(k)]$ e $\Im[X(k)]$ richiede due moltiplicazioni ed un'addizione. Lo sforzo computazionale totale dell'algoritmo di Goertzel per segnali reali è, dunque, $N+2$ moltiplicazioni e $2N+1$ addizioni. Questo è un risparmio se comparato al metodo diretto della (4), che richiede $2N$ moltiplicazioni e $2N$ addizioni per calcolare la parte reale ad immaginaria di $X(k)$, da un segnale reale $x(k)$. E' importante sottolineare, tuttavia, che sia l'algoritmo di Goertzel sia la DFT diretta hanno una complessità computazionale proporzionale a N^2 *quando è richiesto l'intero spettro di frequenze*. In questi casi, gli algoritmi FFT, che hanno una complessità computazionale proporzionale a $N\log_2 N$, risultano più vantaggiosi. In generale, si può sostenere [2, p. 322] che l'uso dell'algoritmo di Goertzel può essere vantaggioso in termini computazionali se il numero di armoniche di interesse è inferiore a $\log_2 N$[5].

Esiste inoltre un vantaggio nascosto nell'utilizzo dell'algoritmo di Goertzel rispetto al metodo diretto della (4). In quest'ultima, l'intera finestra di acquisizione del segnale è richiesta prima di poter calcolare $X(k)$. Ciò significa che il valore di $X(k)$ è calcolato una volta sola all'istante temporale $n=N$, mentre non vengono eseguite operazioni per gli istanti precedenti $1 \leq n \leq N-1$. Nel caso dell'analisi in tempo reale delle armoniche

[5]Si ricordi che, fissato il periodo di campionamento T_c, la risoluzione in frequenza è $F = 1/NT_c$ e quindi essa è tanto migliore quanto maggiore è N.

in azionamenti elettrici, ciò significa che il microprocessore che esegue la procedura di controllo dovrebbe essere sufficientemente veloce da eseguire $2N$ moltiplicazioni e $2N$ addizioni in un singolo periodo di campionamento, quando $n = N$. Questo sforzo computazionale rappresenta un collo di bottiglia. Al contrario, la definizione ricorsiva dell'algoritmo di Goertzel permette la suddivisione delle $N + 2$ moltiplicazioni e delle $2N + 1$ addizioni nell'intera finestra temporale di acquisizione. Per ogni istante di campionamento viene solo aggiornata la sequenza $s(n)$, con uno sforzo computazionale di una moltiplicazione e due addizioni. Quando $n = N$ vengono eseguite due moltiplicazioni ed un'addizione in più, per un totale di tre moltiplicazioni e tre addizioni nel caso peggiore. In questo senso, l'algoritmo di Goertzel può essere efficacemente utilizzato in sistemi real-time senza cambiamenti significativi nel tempo di esecuzione delle procedure di controllo. Inoltre, il valore di N può essere incrementato a piacimento senza alcun effetto sullo sforzo computazionale di ogni periodo di campionamento, raggiungendo potenzialmente risoluzioni in frequenza molto alte (la risoluzione in frequenza è infatti inversamente proporzionale alla lunghezza della finestra temporale di acquisizione del segnale).

2.4 Esempio di calcolo dell'algoritmo di Goertzel

Si supponga di applicare l'algoritmo di Goertzel alla sequenza di ingresso

$$x[n] = \cos(2\pi f_0 n T_c + \varphi) \qquad n = 0, \ldots, N-1 \qquad \varphi \in [0, 2\pi] \tag{22}$$

dove la frequenza f_0 è scelta pari ad un multiplo della frequenza F_c/N (risoluzione frequenziale della DFT), i.e. $f_0 = k_0 F_c/N$ con $k_0 \in \{0, 1, \ldots, N-1\}$.
L'algoritmo consiste nel filtrare la sequenza x con un filtro avente risposta impulsiva

$$h[n] \triangleq W_N^{-kn} \delta_{-1}[n] \qquad n \in \mathbb{Z} \tag{23}$$

dove $\delta_{-1}[n]$ è la sequenza gradino unitario (i.e. $\delta_{-1}[n] = 1$ per $n \geq 0$ e $\delta_{-1}[n] = 0$ per $n < 0$), e $k \in \{0, 1, \ldots, N-1\}$ è l'indice del tono della DFT che si desidera rilevare. Come mostrato in (11), il campione N–esimo dell'uscita y del filtro è pari al tono k-esimo della DFT X della sequenza di ingresso x, ovvero $y[N] = X[k]$.
In particolare, per la sequenza (22) e un generico $k \in \{0, 1, \ldots, N-1\}$ in (23) vale che

$$y[N] = X[k] = \frac{N}{2} (\cos\varphi + j \sin\varphi)\, \delta[k - k_0] + \frac{N}{2} (\cos\varphi - j \sin\varphi)\, \delta[k + k_0] \tag{24}$$

dove $\delta[n]$ è la funzione "delta" di Kronecker (i.e. $\delta[0] = 1$ e $\delta[n] = 0$ per $n \neq 0$). Pertanto, scegliendo $k = k_0$ si ha che

$$y[N] = X[k_0] = \frac{N}{2}\left(\cos\varphi + j\sin\varphi\right) \tag{25}$$

e quindi

$$|y[N]| = |X[k_0]| = \frac{N}{2} \tag{26}$$

$$\angle y[N] = \angle X[k_0] = \operatorname{atan2}\left(\sin\varphi, \cos\varphi\right) = \varphi \tag{27}$$

Nota: la (24) è stata ottenuta espandendo preliminarmente l'espressione (22) con la formula di addizione del coseno

$$\begin{aligned} x[n] &= \cos\varphi\cos(2\pi f_0 n T_c) - \sin\varphi\sin(2\pi f_0 n T_c) = \\ &= \cos\varphi\cos\left(\frac{2\pi}{N}k_0 n\right) - \sin\varphi\sin\left(\frac{2\pi}{N}k_0 n\right) \end{aligned} \tag{28}$$

ed utilizzando poi le seguenti trasformate notevoli

$$\cos\left(\frac{2\pi}{N}k_0 n\right) \quad \xrightarrow{\text{DFT}} \quad \frac{N}{2}\delta[k+k_0] + \frac{N}{2}\delta[k-k_0] \tag{29}$$

$$\sin\left(\frac{2\pi}{N}k_0 n\right) \quad \xrightarrow{\text{DFT}} \quad j\,\frac{N}{2}\delta[k+k_0] - j\,\frac{N}{2}\delta[k-k_0] \tag{30}$$

dove $k \in \{0,\, 1,\, \ldots,\, N-1\}$. Dalla (28) si ottiene

$$\begin{aligned} X[k] = \cos\varphi\left(\frac{N}{2}\delta[k+k_0] + \frac{N}{2}\delta[k-k_0]\right) - \\ - \sin\varphi\left(j\,\frac{N}{2}\delta[k+k_0] - j\,\frac{N}{2}\delta[k-k_0]\right) \end{aligned} \tag{31}$$

da cui, raccogliendo $\frac{N}{2}\delta[k+k_0]$ e $\frac{N}{2}\delta[k-k_0]$, si ricava la (24).

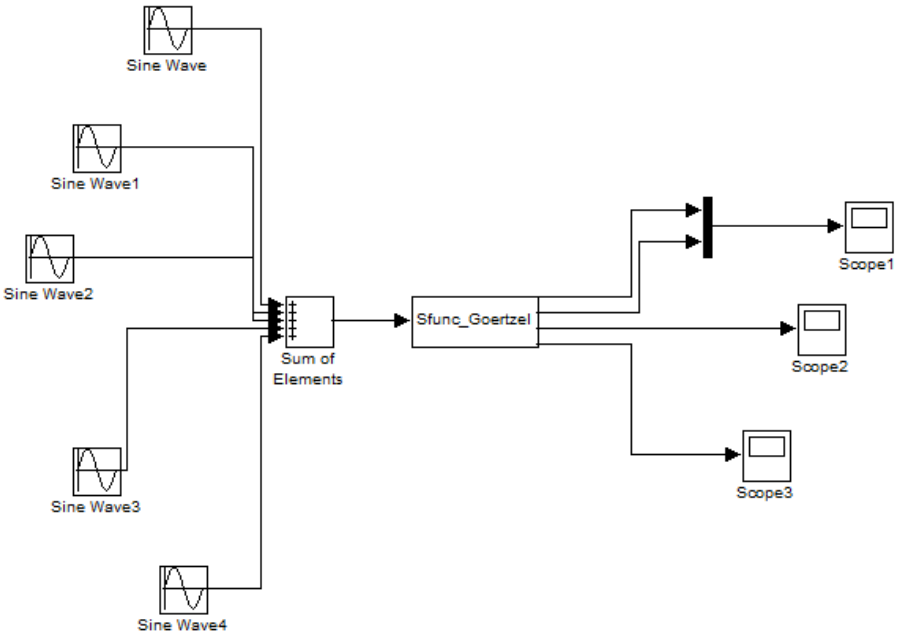

Figura 3: Modello di Simulink dell'algoritmo di Goertzel

3 Implementazione in Simulink

Per implementare l'algoritmo di Goertzel in Simulink si faccia riferimento al modello a blocchi di Figura 3.

A sinistra la serie di blocchi 'SinWave' rappresentano le armoniche che sommate tra loro attraverso il blocco sommatore danno origine al segnale del quale bisogna conoscere l'armonica fondamentale applicando l'algoritmo di Goertzel.
Questo segnale è mandato in ingresso al blocco 'Level-2 M-file S-function' che è il cuore del modello. Questo blocco svolge una funzione scritta in codice MATLAB denominata S-function, la quale sarà spiegata dettagliatamente in seguito. E' proprio la S-function che applica l'algoritmo di Goertzel al segnale di ingresso.
In uscita ci sono tre oscilloscopi che servono a visualizzare i risultati della simulazione. Il primo visualizza insieme l'ingresso e l'uscita (grazie al multiplexer), il secondo l'ingresso e il terzo l'uscita[6].

3.1 Struttura della S-function

La S-function comincia con la funzione $setup$, che inizializza il blocco 'Level-2 M-file S-function' determinando alcune caratteristiche:

- Numero di porte in ingresso e uscita
- Tipo e complessità dei dati in ingresso e uscita
- Numero di parametri
- Periodo di campionamento (continuo, discreto o variabile)
- Funzioni usate nella S-function

Di seguito il listato della funzione $setup$, da cui si possono ricavare tutte le informazioni elencate:

```
function Sfunc_Goertzel(block)

    setup(block);
```

[6]La scelta apparentemente ridondante di visualizzare ingresso e uscita sia separatamente che sullo stesso oscilloscopio serve a mostrare i risultati delle simulazioni nel par.4. In particolare è utile rappresentare ingresso e uscita su uno stesso grafico per evidenziare il ritardo del segnale di output, pari alla finestra di acquisizione.

```
function setup(block)

    % Register number of input and output ports
    block.NumInputPorts = 1;
    block.NumOutputPorts = 4;

    % Setup port properties to be dynamically inherited
    block.SetPreCompInpPortInfoToDynamic;
    block.SetPreCompOutPortInfoToDynamic;

    % Input port properties
    block.InputPort(1).DatatypeID = 0; % double
    block.InputPort(1).Complexity = 'Real';

    % Output port properties
    block.OutputPort(1).DatatypeID = 0; % double
    block.OutputPort(1).Complexity = 'Real';
    block.OutputPort(2).DatatypeID = 0; % double
    block.OutputPort(2).Complexity = 'Real';
    block.OutputPort(3).DatatypeID = 0; % double
    block.OutputPort(3).Complexity = 'Real';
    block.OutputPort(4).DatatypeID = 0; % double
    block.OutputPort(4).Complexity = 'Real';

    % Register parameters
    block.NumDialogPrms = 3;

    % Register sample times
    % [0 offset] : Continuous sample time
    % [positive_num offset] : Discrete sample time
    % [-1, 0] : Port-based sample time
    % [-2, 0] : Variable sample time
    block.SampleTimes = [block.DialogPrm(1).Data 0];
    %% ————————————————————————
    %% Options
    %% ————————————————————————-
    % Specify if Accelerator should use TLC or call back
    % into M-file
    block.SetAccelRunOnTLC(false);

    % Register methods
    block.RegBlockMethod('CheckParameters',
    @CheckPrms);
    block.RegBlockMethod('SetInputPortSamplingMode',
    @SetInpPortFrameData);
    block.RegBlockMethod('PostPropagationSetup',
```

@DoPostPropSetup);
block.RegBlockMethod('InitializeConditions',
@InitConditions);
block.RegBlockMethod('Outputs', @Outputs);

All'inizio della funzione è stabilito che il blocco debba avere una porta di ingresso e quattro di uscita.
I comandi $SetPreCompInpPortInfoToDynamic$ e $SetPreCompOutPortInfoToDynamic$ stabiliscono che le porte di ingresso e uscita ereditano le loro proprietà (dimensione, tipo di dati, complessità, campionamento) dal modello.
Le porte di ingresso e uscita sono impostate in modo che i dati che le attraversano siano numeri reali di tipo double.
I tre parametri (T_C, N, freq_goertzel) sono indicati direttamente nelle opzioni del blocco di Simulink, pertanto non è necessario che siano scritti anche nella S-function.
Il campionamento è discreto con periodo pari a T_C
(block.DialogPrm(1).Data) e 0 offset.
Il comando $SetAccelRunOnTLC(false)$ specifica di non usare il Target Language Compiler (che consente di generare un codice in linguaggio C a partire dal modello di Simulink).
Le funzioni dichiarate col comando $RegBlockMethod$ sono definite nelle successive parti di codice.

```
function CheckPrms(block)
% This function reads the parameters from the Workspace.
% If parameters are not found, it returns an error message

    if (isempty(block.DialogPrm(1).Data))
        error('Missing Tc.');
    end
    if (isempty(block.DialogPrm(2).Data))
        error('Missing N.');
    end
    if (isempty(block.DialogPrm(3).Data))
        error('Missing freq_goertzel.');
    end

%endfunction

function SetInpPortFrameData(block, idx, fd)
% This function sets the sampling mode
```

```
        block.InputPort(idx).SamplingMode = fd;
        block.OutputPort(1).SamplingMode = fd;
        block.OutputPort(2).SamplingMode = fd;
        block.OutputPort(3).SamplingMode = fd;
        block.OutputPort(4).SamplingMode = fd;

    %endfunction

    function DoPostPropSetup(block)
    % This function declairs the variables

        % Number of variables
        block.NumDworks = 8;
        % Counter (from 0 to N-1)
        block.Dwork(1).Name = 'cont';
        block.Dwork(1).Dimensions = 1;
        block.Dwork(1).DatatypeID = 0;
        block.Dwork(1).Complexity = 'Real';
        block.Dwork(1).UsedAsDiscState = true;
        % Variabili intermedie per il calcolo di Goertzel
        block.Dwork(2).Name = 's_k';
        block.Dwork(2).Dimensions = 1;
        block.Dwork(2).DatatypeID = 0;
        block.Dwork(2).Complexity = 'Real';
        block.Dwork(2).UsedAsDiscState = true;
        block.Dwork(3).Name = 's_km1';
        block.Dwork(3).Dimensions = 1;
        block.Dwork(3).DatatypeID = 0;
        block.Dwork(3).Complexity = 'Real';
        block.Dwork(3).UsedAsDiscState = true;
        block.Dwork(4).Name = 's_km2';
        block.Dwork(4).Dimensions = 1;
        block.Dwork(4).DatatypeID = 0;
        block.Dwork(4).Complexity = 'Real';
        block.Dwork(4).UsedAsDiscState = true;
        % Real and imaginary parts of the output
        block.Dwork(5).Name = 'outp_real_km1';
        block.Dwork(5).Dimensions = 1;
        block.Dwork(5).DatatypeID = 0;
        block.Dwork(5).Complexity = 'Real';
        block.Dwork(5).UsedAsDiscState = true;
        block.Dwork(6).Name = 'outp_imag_km1';
```

```
        block.Dwork(6).Dimensions = 1;
        block.Dwork(6).DatatypeID = 0;
        block.Dwork(6).Complexity = 'Real';
        block.Dwork(6).UsedAsDiscState = true;
        % Output of the system
        block.Dwork(7).Name = 'outp';
        block.Dwork(7).Dimensions = 1;
        block.Dwork(7).DatatypeID = 0;
        block.Dwork(7).Complexity = 'Real';
        block.Dwork(7).UsedAsDiscState = true;
        % Time of the output
        block.Dwork(8).Name = 'time';
        block.Dwork(8).Dimensions = 1;
        block.Dwork(8).DatatypeID = 0;
        block.Dwork(8).Complexity = 'Real';
        block.Dwork(8).UsedAsDiscState = true;

    %endfunction

    function InitConditions(block)
    % This function sets the initial conditions of the variables and
    % output ports

        block.Dwork(1).Data = 0;
        block.Dwork(2).Data = 0;
        block.Dwork(3).Data = 0;
        block.Dwork(4).Data = 0;
        block.Dwork(5).Data = 0;
        block.Dwork(6).Data = 0;
        block.Dwork(7).Data = 0;
        block.Dwork(8).Data = 0;
        block.OutputPort(1).Data = 0;
        block.OutputPort(2).Data = 0;
        block.OutputPort(3).Data = 0;
        block.OutputPort(4).Data = 0;

    % endfunction
```

La funzione $CheckPrms$ controlla i campi riservati ai tre parametri. Se non trova i dati, restituisce un messaggio di errore.

La funzione $SetInpPortFrameData$ stabilisce il tipo di campionamento per le porte di ingresso e uscita. In questo caso le porte di ingresso e

uscita accettano segnali $frame - based$. Un frame è un pacchetto di campioni singoli. Con i frame si possono processare contemporaneamente più campioni e ridurre le comunicazioni tra i blocchi, diminuendo così il tempo della simulazione.

La funzione $DoPostPropSetup$ dichiara le variabili ($Dwork$) necessarie per lo svolgimento dell'algoritmo. La variabile $cont$ serve a contare i campioni del segnale di ingresso. Le variabili s_k, s_km1, s_km2, $outp\_real\_km1$ e $outp\_imag\_km1$ sono rispettivamente le variabili $s(n)$, $s(n-1)$, $s(n-2)$, $\Re[X(k)]$ e $\Im[X(k)]$ del paragrafo 2.3. Nella variabile $outp$ è salvato il risultato in uscita della S-function che è rappresentato sugli oscilloscopi. La variabile $time$ è incrementata di 1 ad ogni campione preso in esame e serve a collocare nel tempo ciascun campione dell'output:

$$t = time * T_C \tag{32}$$

dove t è il tempo in millisecondi.

La funzione $InitConditions$ inizializza a 0 tutte le variabili e le porte d'uscita.

Il cuore della S-function è la funzione $Outputs$ che calcola il risultato dell'algoritmo di Goertzel.

```
function Outputs(block)
% Questa funzione calcola l'algoritmo e mostra i risultati

% Updating the variables
cont = block.Dwork(1).Data;
s_k = block.Dwork(2).Data;
s_km1 = block.Dwork(3).Data;
s_km2 = block.Dwork(4).Data;
outp_real_km1 = block.Dwork(5).Data;
outp_imag_km1 = block.Dwork(6).Data;
outp = block.Dwork(7).Data;
time = block.Dwork(8).Data;
Tc = block.DialogPrm(1).Data;
N = block.DialogPrm(2).Data;
freq_goertzel = block.DialogPrm(3).Data;
inp = block.InputPort(1).Data;

% Real and imaginary coefficients
realW = cos(2*pi*freq_goertzel*Tc);
imagW = sin(2*pi*freq_goertzel*Tc);

if (cont == N-1)
```

```
        % Real and imaginary part calculation
        outp_real_km1 = s_k-realW*s_km1;
        outp_imag_km1 = imagW*s_km1;
        % Setting to zero s_km1, s_km2 and cont for next
        detection
        s_km1 = 0;
        s_km2 = 0;
        cont = 0;

    else

        % Save s_km1 and s_km2
        s_km2 = s_km1;
        s_km1 = s_k;

    end;

    % Increment of the counter
    cont = cont+1;
    % Update of s_k
    s_k = inp + 2*realW*s_km1-s_km2;

    % Update of the output
    time = time+1;
    outp= (sqrt(outp_real_km1*outp_real_km1+
    outp_imag_km1*outp_imag_km1)/(0.5*N))
    *cos(2*pi*freq_goertzel*time*Tc+
    atan2(outp_imag_km1,outp_real_km1));

    % Visualize the output
    block.OutputPort(1).Data = inp;
    block.OutputPort(2).Data = outp;
    block.OutputPort(3).Data = inp;
    block.OutputPort(4).Data = outp;

    % Update of the variables
    block.Dwork(1).Data = cont;
    block.Dwork(2).Data = s_k;
    block.Dwork(3).Data = s_km1;
    block.Dwork(4).Data = s_km2;
    block.Dwork(5).Data = outp_real_km1;
    block.Dwork(6).Data = outp_imag_km1;
    block.Dwork(7).Data = outp;
    block.Dwork(8).Data = time;
```

% endfunction

All'inizio della funzione le variabili $Dwork$, i parametri e la variabile inp vengono aggiornati con i valori dei rispettivi campi. Nei campi $block.Dwork(n).Data$ possono essere salvate le condizioni iniziali (funzione $InitConditions$) oppure i valori intermedi delle variabili aggiornati all'interno della funzione. Nei campi $block.DialogPrm(n).Data$ ci sono i valori dei parametri (T_C, N, $freq\_goertzel$) letti dalla funzione $CheckPrms$. Il campo $block.InputPort(1).Data$ è progressivamente aggiornato coi campioni di input che vengono passati uno ad uno in ingresso alla funzione.

Le successive righe di codice forniscono le istruzioni per il calcolo dell'algoritmo di Goertzel. I coefficienti $realW$ e $imagW$ sono la parte reale e immaginaria di W_N^k, come si vede dalla (19). Per i primi $N-1$ campioni l'algoritmo aggiorna le variabili s_km1 e s_km2 rispettivamente con i valori precedenti di s_k e s_km1, la variabile s_k come nella (17), e incrementa il contatore di 1.

Quando l'algoritmo arriva all' N-esimo campione ($cont = N-1$), esegue le parti di codice comprese all'interno dell'istruzione if. Goertzel aggiorna la parte reale e immaginaria dell'uscita come nella (20) e azzera le variabili s_km1, s_km2 e $cont$.

L'output è prodotto in tempo reale per ogni campione temporale. Viene incrementata la variabile temporale e poi calcolato l'output come segue:

$$outp = |X(k)| * cos(2 * \pi * freq\_goertzel * time * T_C + \angle X(k)) \qquad (33)$$

dove $|X(k)|$ e $\angle X(k)$ sono calcolati come nella (21).
Poi vengono specificati i segnali da visualizzare sugli oscilloscopi (si veda il par. 3.

Infine tutti i $block.Dwork(n).Data$ sono aggiornati coi nuovi valori assegnati alle variabili all'interno della funzione.

4 Risultati delle simulazioni

4.1 Semplice sinusoide "nascosta" in un segnale periodico

In questo paragrafo viene testato il modello descritto nel par. 3. Prendendo in esame la Figura 3, bisogna scegliere i valori dei parametri richiesti dai vari blocchi di Simulink. Le cinque armoniche sommate in ingresso abbiano le seguenti equazioni:

$$\begin{aligned} SinWave &= 5 * \sin(2 * \pi * freq\_goertzel * t) \\ SinWave1 &= 0.5 * \sin(2 * \pi * freq\_goertzel * 2 * t) \\ SinWave2 &= 2 * \sin(2 * \pi * freq\_goertzel * 3 * t) \\ SinWave3 &= 1 * \sin(2 * \pi * freq\_goertzel * 4 * t) \\ SinWave4 &= 0.3 * \sin(2 * \pi * freq\_goertzel * 5 * t) \end{aligned} \tag{34}$$

La S-function di Goertzel richiede, oltre ad un segnale d'ingresso, anche 3 parametri fondamentali, descritti nel par. 3. I valori di tali parametri scelti in questa simulazione sono i seguenti:

$$\begin{aligned} T_C &= 0.1ms \\ N &= 10000 \\ freq\_goertzel &= 20Hz \end{aligned} \tag{35}$$

che ricordiamo essere rispettivamente il periodo di campionamento, il numero di campioni di una finestra di acquisizione e la frequenza di Goertzel.

L'algoritmo dovrebbe dare in uscita la sola componente fondamentale ($SineWave$) a 20 Hz, con un ritardo pari alla durata dell'esecuzione dell'algoritmo T_G, facilmente calcolabile:

$$T_G = N * T_C = 1s \tag{36}$$

Il risultato della simulazione è rappresentato in Figura 4. L'uscita (in viola) è nulla fino a T_G e poi diventa una sinusoide di ampiezza pari a 5 e periodo di 50 ms (e frequenza = 1/50 ms = 20 Hz). La sinusoide è pari a quella fondamentale, quindi il risultato è quello atteso.

La fase è calcolata rispetto a un coseno senza sfasamento (si veda l'esempio del par. 2.4 con $\varphi = 0$). In questo caso la fase dovrebbe essere uguale a $-\frac{\pi}{2} = -1.5708$, che corrisponde a un seno senza sfasamento. Come si vede in Figura 5, la fase è circa uguale a $-\frac{\pi}{2}$ (a meno di un piccolo ritardo), quindi il risultato è quello atteso.

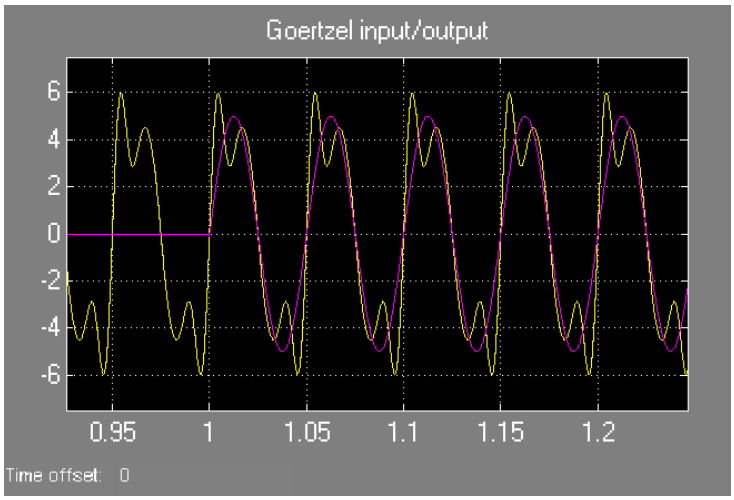

Figura 4: Risultato della simulazione

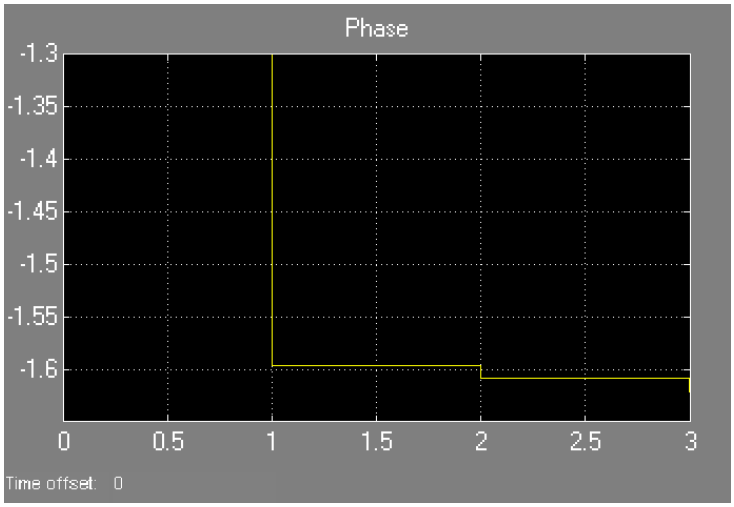

Figura 5: Fase dell'armonica fondamentale

4.2 Modifica della S-function per analizzare anche i "transitori di convergenza"

Se l'algoritmo aggiorna l'uscita più volte prima di acquisire tutti gli N campioni dell'ingresso, Goertzel si avvicina progressivamente alla sinusoide corretta?

Per rispondere a questo quesito bisogna modificare leggermente il codice della S-function. All'interno della funzione $DoPostPropSetup$ si deve aggiungere la seguente variabile contatore, reale di tipo double:

```
block.Dwork(9).Name = 'cont_outp';
block.Dwork(9).Dimensions = 1;
block.Dwork(9).DatatypeID = 0;
block.Dwork(9).Complexity = 'Real';
block.Dwork(9).UsedAsDiscState = true;
```

La variabile $UsedAsDiscState = true$ vuol dire che i valori assunti dal contatore sono quelli degli stati discreti del blocco.

Questo contatore serve a fare in modo che la S-function calcoli la parte reale e immaginaria dell'uscita con maggiore frequenza (ogni $N/10$ conteggi) rispetto ai normali N conteggi.

La funzione Outputs è modificata come segue:

```
% Real and imaginary coefficients
realW = cos(2*pi*freq_goertzel*Tc);
imagW = sin(2*pi*freq_goertzel*Tc);

% Update of the real and imaginary part every N/10 samples,
% before the acquisition time
if (cont_outp == N/10-1)

    % Real and imaginary part calculation
    outp_real_km1 = s_k-realW*s_km1;
    outp_imag_km1 = imagW*s_km1;
    % Setting cont_outp to zero
    cont_outp = 0;

end;

if (cont == N-1)

    % Setting to zero s_km1, s_km2 and cont for next
    detection
    s_km1 = 0;
```

```
        s_km2 = 0;
        cont = 0;

    else

        % Save s_km1 and s_km2
        s_km2 = s_km1;
        s_km1 = s_k;

    end;

    % Increment of the counters
    cont = cont+1;
    cont_outp = cont_outp+1;
    % Update of s_k
    s_k = inp + 2*realW*s_km1-s_km2;
```

Dopo un tempo di 0.1 secondi ($T_G/10$), l'algoritmo manderà in output una risposta ottenuta esaminando solo 1000 campioni anzichè 10000. Quello che si vuole verificare è se al termine dell'esecuzione ($T_G = 1s$), l'uscita si sia progressivamente avvicinata alla sinusoide fondamentale.

L'uscita che si vede sull'oscilloscopio è rappresentata in Figura 6.

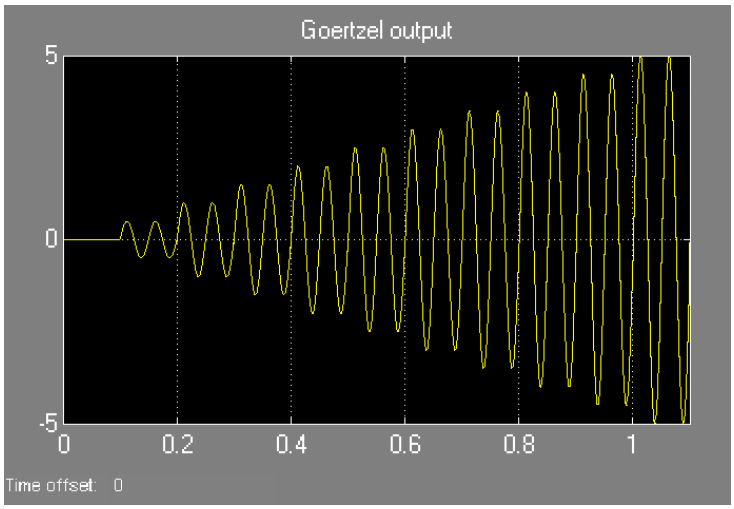

Figura 6: Uscita della S-function modificata

Come si vede, avvicinandosi al termine dell'algoritmo, l'uscita si avvicina progressivamente all'armonica attesa.

Si può quindi rispondere affermativamente alla domanda iniziale.

4.3 Risultati con armoniche variabili

In presenza di disturbi durante l'esecuzione dell'algoritmo, può capitare che questi modifichino l'ampiezza di una o più armoniche secondarie dell'ingresso in modo significativo. L'algoritmo di Goertzel è in grado di fornire in uscita l'armonica fondamentale alla frequenza di Goertzel, senza farsi influenzare dalla variazione del segnale di ingresso?

Nell' esempio che segue, si fa variare la seconda armonica (alla frequenza di 40 Hz) tra le ampiezze seguenti: 0.1, 0.5, 0.7, 1. I blocchi di Simulink usati sono mostrati in Figura 7.

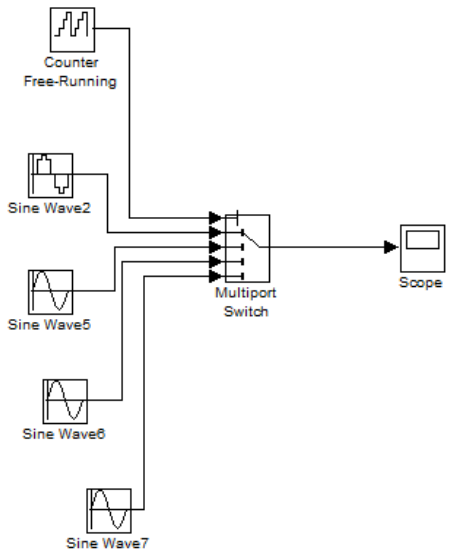

Figura 7: Modello di Simulink per variare la seconda armonica

Il blocco $CounterFree - Running$ controlla il $MultiportSwitch$ mandando in uscita alternativamente ciascuno dei 4 ingressi.

In Figura 8 è riportato l'andamento nel tempo della seconda armonica. Ogni 50 millisecondi essa cambia ampiezza.

La seconda armonica è sommata alle altre e inviata in ingresso al sistema.

L'armonica fondamentale è sempre la stessa (ampiezza=5, freq=20Hz).

Il segnale di ingresso dell'algoritmo di Goertzel è riportato in Figura 10. Si vede chiaramente che l'ingresso non è periodico con periodo di 50 millisecondi.
L'uscita è riportata in Figura 11.

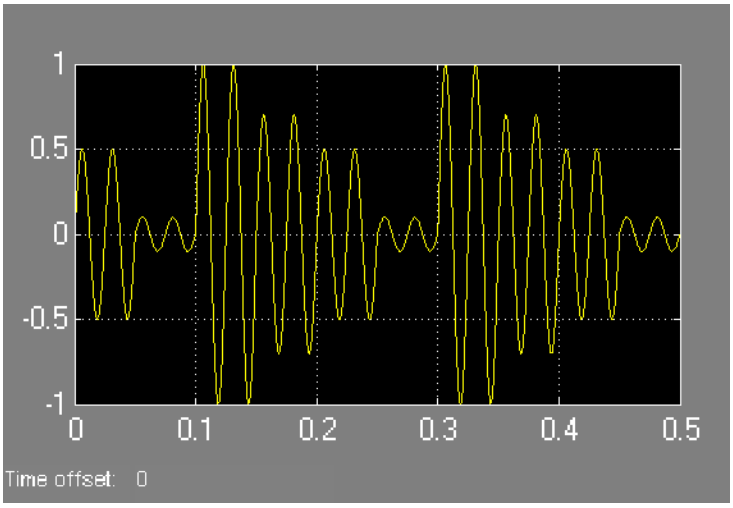

Figura 8: Andamento della seconda armonica

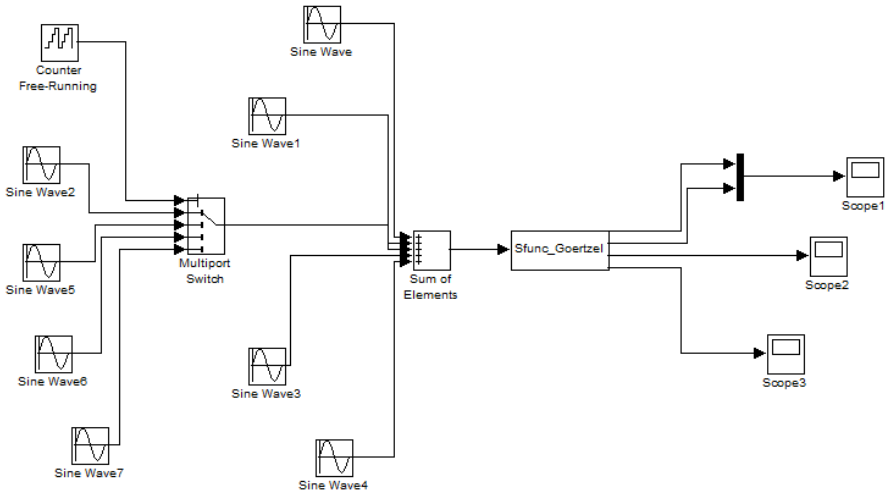

Figura 9: Modello di Simulink con seconda armonica variabile

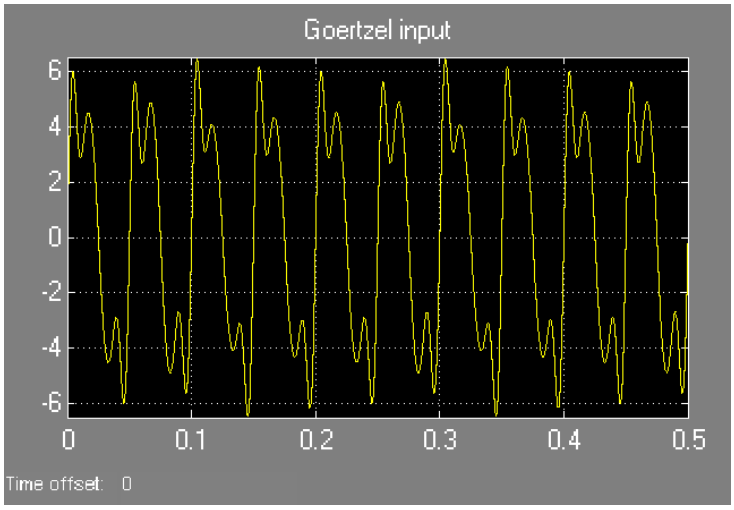

Figura 10: Ingresso della S-function

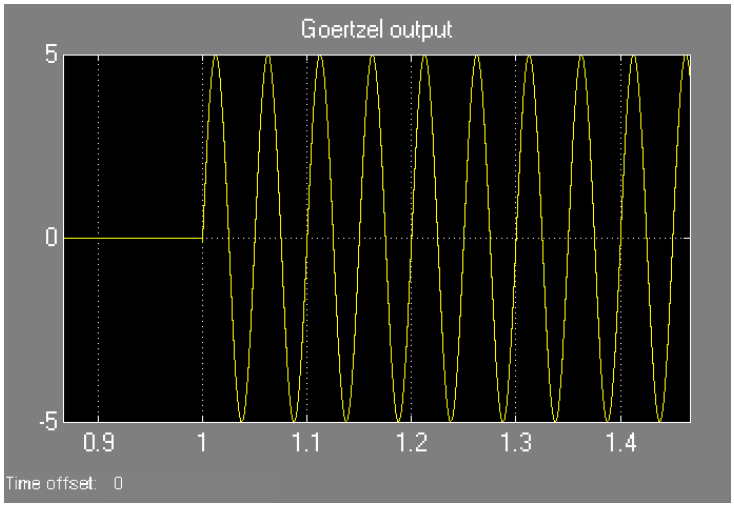

Figura 11: Uscita della S-function

Al termine dell'esecuzione dell'algoritmo ($T_G = 1s$), il sistema è in grado di fornire in uscita l'armonica fondamentale. L'ampiezza è pari a 5, che è quella attesa.

In conclusione, si può affermare che l'algoritmo di Goertzel funziona anche se durante l'esecuzione un'armonica secondaria varia in ampiezza.

5 Applicazione a f.e.m non sinusoidali

5.1 Simulazione dell'algoritmo con le f.e.m. del motore PM-LA-01

Ora si prova a far funzionare Goertzel con dei dati reali. In ingresso all'algoritmo si mandano i dati relativi alla f.e.m. del motore PM-LA-01. In Figura 12 è riportato l'andamento della f.e.m. nel tempo.

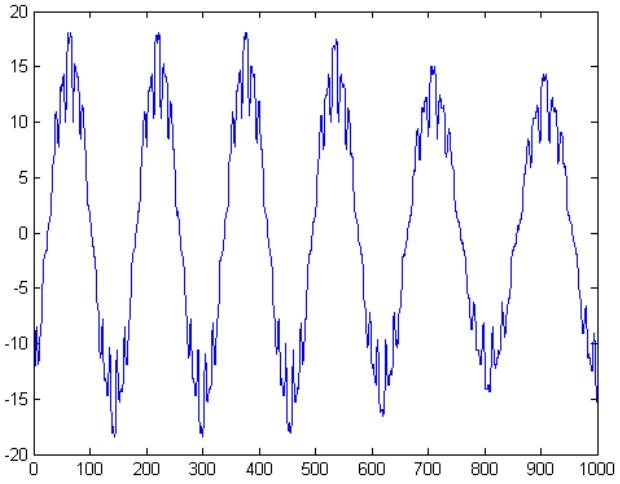

Figura 12: F.e.m. del motore

Si vede che nel tempo la f.e.m. subisce un'attenuazione. Per l'analisi con Goertzel si considera solo il primo periodo e lo si ripete nel tempo in modo da lavorare con un segnale periodico in ingresso, come richiesto dall'algoritmo (Figura 13).

Il modello di Simulink è sempre lo stesso (Figura 14), con la differenza che in ingresso (blocco $FromWorkspace$) ci sono i dati relativi alla f.e.m. del motore caricati nello Workspace di MATLAB. La variabile $Voltage$ rappresenta una matrice $1000x2$, dove la prima colonna è l'asse dei tempi e

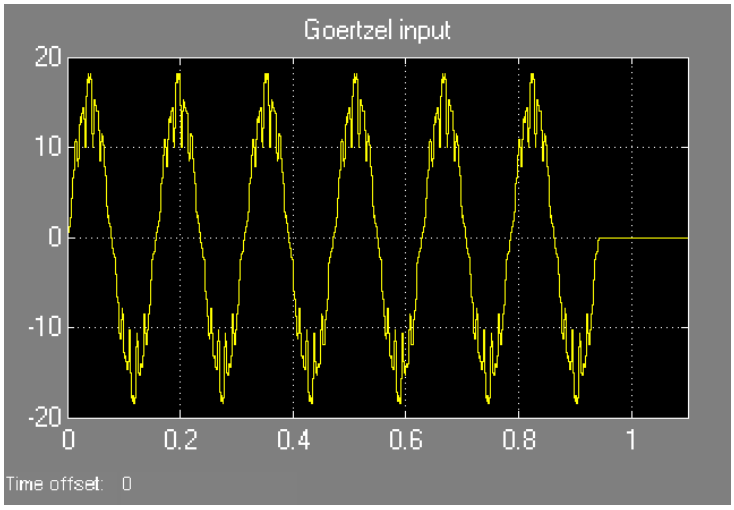

Figura 13: Segnale periodico d'ingresso

nella seconda ci sono i valori di f.e.m. per ogni istante di campionamento (Figura 13).

Dai dati della f.e.m. si risale facilmente ai tre parametri richiesti dalla S-function. Il periodo di campionamento T_C è la distanza temporale tra un campione e l'altro, la $frequenza\ di\ Goertzel$ si ricava misurando il periodo del segnale, il numero di campioni N è dato da:

$$N = \frac{T}{T_C} \tag{37}$$

dove T è il periodo della f.e.m. I valori dei 3 parametri sono i seguenti:

$$\begin{aligned} T_C &= 1ms \\ N &= 157 \\ freq\_goertzel &= 6.37Hz \end{aligned} \tag{38}$$

L'output della simulazione è riportato in Figura 15. L'ampiezza della fondamentale è circa 15.55 Volt.

Una volta ottenuta la f.e.m. concatenata si può calcolare il flusso concatenato magnetico. Si ricorda che il flusso concatenato Λ_{mg} si calcola nel seguente modo:

$$\Lambda_{mg} = \frac{V_{pk-pk}}{2\sqrt{3}} \frac{1}{\omega_{me}} \tag{39}$$

La tensione picco-picco è il doppio dell'ampiezza:

$$V_{pk-pk} = 15.55 * 2 = 31.1V \tag{40}$$

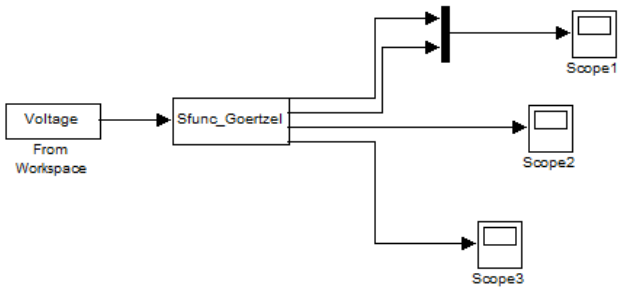

Figura 14: Modello di Simulink per l'analisi della f.e.m. del motore PM-LA-01

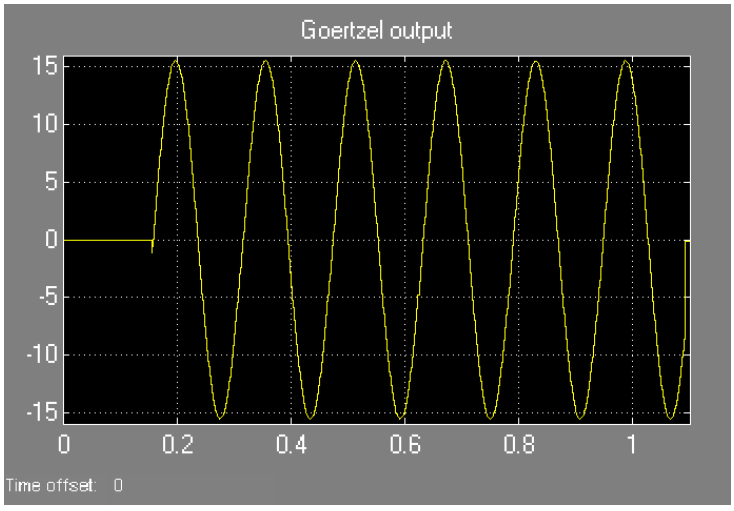

Figura 15: Armonica fondamentale della f.e.m. del motore PM-LA-01

Conoscendo il periodo T della f.e.m. ($T = 1/freq\_goertzel = 157ms$) si calcola facilmente la velocità meccanico-elettrica:

$$\omega_{me} = \frac{2\pi}{T} = 40.02\frac{rad}{s} \tag{41}$$

Sostituendo nella (39) si ottiene:

$$\Lambda_{mg} = 0.2243\frac{Vs}{rad} \tag{42}$$

Si può verificare l'esattezza di questo calcolo sostituendo nella (39) i valori nominali del motore:

$$\omega_N = 3000rpm = 314.159\frac{rad}{s} \tag{43}$$

$$V_N = 190V_{RMS} \tag{44}$$

$$\Lambda_{mg} = \frac{V_N\sqrt{2}}{\sqrt{3}}\frac{1}{2\omega_N} = 0.2469\frac{Vs}{rad} \tag{45}$$

dove $2\omega_N = \omega_{me}$ poichè il motore ha 2 coppie polari.

5.2 Applicazioni industriali dell'algoritmo di Goertzel

Per concludere la tesi, vengono portati due esempi di possibili applicazioni industriali dell'algoritmo di Goertzel. Per un approfondimento dei temi trattati in questo paragrafo si rimanda alle voci [3] e [4] nei riferimenti bibliografici.

Il primo esempio riguarda l'esecuzione di test non distruttivi sui materiali. In [3, p.2] è spiegato che le prove non distruttive rappresentano il complesso di esami, prove e rilievi condotti impiegando metodi che non alterano il materiale e non richiedono la distruzione o l'asportazione di campioni dalla struttura in esame.
Riferendosi in particolare alle prove non distruttive con la tecnica delle correnti indotte, sfruttando il sensore di campo magnetico fluxset, si dice che [3, p.5] la messa in opera dello strumento è stata resa possibile anche dal contemporaneo sviluppo ed ottimizzazione degli algoritmi di elaborazione dei segnali acquisiti.[...] Il software di gestione ed elaborazione necessita anche di caratteristiche quali la ridotta occupazione di memoria (installazione on-board) e la velocità di calcolo (elaborazione in real time); queste specifiche progettuali, sono state ottemperate effettuando analisi in frequenza, sfruttando l'algoritmo di Goertzel.

Un'altro esempio di applicazione di Goertzel è quello della modulazione DTMF (Dual-Tone Multi-Frequency). In [4, p.47] si legge che questo tipo di modulazione è usata dagli apparecchi telefonici per trasmettere alla centrale il numero telefonico composto dall'utente. Il modo più diretto per effettuare una demodulazione di un segnale DTMF è quello di ricostruire lo spettro del segnale ricevuto mediante la DFT o gli algoritmi a complessità ridotta della FFT e di Goertzel [4, p.48]. La demodulazione di un segnale DTMF richiede ad esempio di individuare solo 8 possibili frequenze [4, p.51], quindi l'algoritmo di Goertzel risulta più efficiente rispetto alla FFT dato il basso numero di frequenze di interesse, come spiegato nel par. 2.3.

Riferimenti bibliografici

[1] G. Cariolaro,(2001), *La teoria unificata dei segnali - Nuova edizione*, UTET Libreria

[2] A. V. Oppenheim and R. W. Schafer, (1996), *Elaborazione numerica dei segnali*, Franco Angeli

[3] M. Laracca, Tesi di Dottorato di Ricerca in Ingegneria Elettrica e dell'Informazione, *Metodi e strumenti di misura per l'esecuzione di test non distruttivi su materiali conduttori*, Università degli Studi di Cassino, (2005)

[4] G. Baccarani, Esercitazioni, 2007, URL:http://didattica.arces.unibo.it/mod/resource/view.php?id=370