Esercizi sulla trasformata di Fourier (FFT)

francesco.fuso@unipi.it

(Dated: version 2 - FF, 5 aprile 2018)

Questa nota ha lo scopo di introdurre l'argomento dell'*analisi di Fourier* condotta numericamente (*Discrete Fourier Transform* - DFT) attraverso il metodo della *Fast Fourier Transform (FFT)* applicato a dati acquisiti con Arduino durante le esercitazioni pratiche. Dal punto di vista concettuale, si tratta di un argomento potentissimo e di importanza ubiqua in tantissimi settori della ricerca scientifica. Il nostro approccio al problema è minimale e ci accontentiamo di impiegare una versione dell'algoritmo FFT implementato in Python per trattare dati di origine sperimentale. La nota, dopo una breve introduzione, fornisce suggerimenti pratici e istruzioni per lo svolgimento dell'esercizio.

I. ANALISI DI FOURIER

In termini molto generali e volutamente privi dei contenuti concettuali e matematici che avete trovato, o troverete, in altri corsi, l'analisi di Fourier può essere considerata un meraviglioso strumento per esaminare sistemi fisici, in particolare quelli che si comportano in modo "lineare", in cui, per esempio, i segnali in "uscita" e "ingresso" sono legati tra loro da una relazione causale che può essere modellata da una funzione (semplice e possibilmente analitica) di trasferimento.

Questa funzione, generalmente complessa, descrive il comportamento del sistema in un dato dominio, che è molto spesso quello delle frequenze, o frequenze angolari, come abbiamo più volte avuto modo di incontrare nel nostro corso. La *trasformata* di Fourier, che è alla base del metodo di analisi che vogliamo trattare, consente di passare al dominio "coniugato", che è quello dei tempi. Naturalmente esistono altri domini coniugati tra loro, cioè tali che il prodotto tra le variabili (coordinate) dei due domini sia adimensionale; tuttavia in questa nota, in accordo con le nostre esigenze e possibilità, ci restringiamo a esaminare la trasformazione dal dominio dei tempi a quello delle frequenze. L'operazione inversa, che può anche essere eseguita, è chiamata *anti-trasformata*: nei nostri esempi essa permette di passare dal dominio delle frequenze a quello dei tempi.

Per essere estremamente grossolani e animati da senso pratico, possiamo vedere la trasformata di Fourier come un'estensione del concetto di *serie* di Fourier che già abbiamo incontrato e utilizzato nel nostro corso. Abbiamo in particolare già mostrato come una generica funzione *periodica* di frequenza angolare ω possa essere espressa come somma di (tante) funzioni seno e coseno, dette armoniche, ognuna di frequenza angolare $k\omega$, con k intero. I pesi della somma li abbiamo definiti *coefficienti* dello sviluppo in serie di Fourier, e, all'epoca, abbiamo anche stabilito una regola per la loro determinazione.

La trasformata di Fourier rappresenta, in pratica, l'applicazione dello stesso concetto a funzioni *non periodiche* ma dipendenti dal tempo secondo un andamento "qualsiasi". Detta $g(t)$ una tale funzione, si ha che la sua trasformata di Fourier è, a parte eventuali coefficienti di normalizzazione,

$$\mathscr{F}\{g(t)\} = \tilde{g}(f) = \int_{-\infty}^{\infty} g(t)\exp(j2\pi ft)dt \ , \qquad (1)$$

dove $j = \sqrt{-1}$ è l'unità immaginaria e con $f = \omega/(2\pi)$ indichiamo una frequenza. La $\tilde{g}(f)$, calcolata per una data f (ovvero per l'intervallo infinitesimo $f, f+df$, secondo quanto discuteremo nel seguito), rappresenta in sostanza il valore dei coefficienti (complessi) dell'espansione di Fourier della $g(t)$.

L'anti-trasformata può essere definita come

$$\mathscr{F}^{-1}\{\tilde{g}(f)\} = g(t) = \int_{-\infty}^{\infty} \tilde{g}(f)\exp(-j2\pi ft)df \ , \qquad (2)$$

sempre a parte eventuali coefficienti di normalizzazione.

La bellezza dell'analisi di Fourier può essere facilmente compresa ricordando quanto abbiamo fatto, per esercizio, con la serie di Fourier. All'epoca abbiamo osservato come il comportamento di un sistema (un filtro passa-basso o passa-alto, o una combinazione dei due) per un segnale periodico non sinusoidale $g_{in}(t)$ (un'onda quadra o triangolare) potesse essere ben riprodotto utilizzando la funzione di trasferimento $T(f)$ determinata per segnali sinusoidali, e ricostruendo il segnale di uscita $g_{out}(t)$ come sovrapposizione di tante componenti che rappresentavano l'applicazione della funzione di trasferimento alle componenti del segnale in ingresso. Questa procedura può essere generalizzata, affermando che

$$g_{out}(t) = \mathscr{F}^{-1}\{\tilde{g}_{out}(f)\} \qquad (3)$$

$$\tilde{g}_{out}(f) = T(f)\tilde{g}_{in}(f) \qquad (4)$$

$$\tilde{g}_{in}(f) = \mathscr{F}\{g_{in}(t)\} \ , \qquad (5)$$

cioè, in sostanza, che il comportamento del sistema nel dominio delle frequenze è descritto da una semplice operazione di prodotto tra funzione di trasferimento e trasformata di Fourier del segnale in ingresso, e che il segnale di uscita può essere ricostruito anti-trasformando tale prodotto (tutto questo, a parte eventuali coefficienti di normalizzazione ha, come sapete, il nome di "prodotto di convoluzione").

In questa nota, e almeno per questo anno accademico, non applicheremo l'analisi di Fourier a questa interessantissima casistica, limitandoci solo a eseguire trasformate

di Fourier $\tilde{g}(f)$ di segnali $g(t)$ dipendenti dal tempo e acquisiti sperimentalmente con Arduino in qualche esperienza. Dunque in questa nota ci proponiamo di costruire lo spettro dei coefficienti di Fourier di alcuni segnali acquisiti in laboratorio: questo è anche il contenuto dell'esercizio obbligatorio proposto.

Anche questa semplice procedura può risultare estremamente utile. Pensate ad esempio ad un segnale periodico, a una certa frequenza, che è sovrapposto a un rumore, periodico a una o più frequenze, o addirittura aperiodico. Lo spettro del segnale consente di individuare le sue componenti a diverse frequenze permettendo, ad esempio, di determinare l'ampiezza dell'oscillazione del segnale periodico anche se questa è praticamente dello stesso ordine, o addirittura minore, rispetto all'ampiezza del rumore. Inoltre il passaggio dal dominio dei tempi a quello delle frequenze può permettere di ricavare la curva di risposta (spettro) di un oscillatore partendo da misure in funzione del tempo, che talvolta sono le uniche a poter essere eseguite. La possibilità di anti-trasformare consente anche l'operazione inversa, cioè costruire l'andamento temporale di un'oscillazione smorzata partendo dai dati spettrali.

Prima di procedere con le istruzioni pratiche, osserviamo che la trasformata di Fourier, Eq. 1, è per sua definizione, complessa. Nelle nostre esperienze pratiche il segnale è generalmente una d.d.p. acquisita da Arduino in funzione del tempo, dunque si tratta di una grandezza reale. Il *modulo* della trasformata di Fourier fornisce già praticamente tutte le informazioni rilevanti sullo spettro delle ampiezze: ad esempio, eventuali picchi nello spettro della $\tilde{g}(f)$ a determinati valori delle f indicano delle periodicità nella $g(t)$ a quelle frequenze.

Talvolta, per esempio nello studio degli oscillatori smorzati, può essere rilevante calcolare il *modulo quadro* $|\tilde{g}(f)|^2$, a cui si dà spesso il nome di *spettro di potenza*. Questa denominazione tiene conto del fatto che la potenza è proporzionale al modulo quadro dell'ampiezza: dunque tale spettro fornisce informazioni sulle componenti spettrali della potenza che interessa il sistema sotto analisi, cioè, nel nostro caso, gli eventuali circuiti realizzati.

Per quanto riguarda le dimensioni e le unità di misura delle grandezze ottenute tramite FFT, esse possono essere ricavate dalla definizione di Eq. 1. Dal punto di vista dimensionale, notate che moltiplicare la $g(t)$ per un tempo, come in Eq. 1, equivale a dividere per una frequenza. Di conseguenza, e in accordo con quanto sopra affermato sul fatto che $\tilde{g}(f)$ rappresenta i coefficienti di Fourier nell'intervallo $f, f + df$, la $\tilde{g}(f)$ può essere considerata come una *densità spettrale di ampiezza*; analogamente la $|\tilde{g}(f)|^2$ una *densità spettrale di potenza*. Nel caso continuo della matematica, l'intervallo di frequenza può essere reso piccolo a piacere, fino a diventare un infinitesimo df. Nella realtà fisica della trasformata discreta, l'intervallo di frequenza Δf è finito e la sua larghezza è determinata dalle caratteristiche del campionamento della $g(t)$, secondo quanto illustreremo nel seguito. Per fare un esempio, supponiamo di avere una funzione perfettamente sinusoidale (monocromatica) di frequenza f_0 determinata. Secondo la definizione di Eq. 1, il suo spettro sarebbe rappresentato da una funzione delta centrata in $f = f_0$. Tuttavia, se la funzione è discreta, cioè campionata su intervalli temporali finiti, la FFT produce un picco di altezza finita e larghezza non nulla, corrispondente proprio alla risoluzione in frequenza Δf.

Il valore numerico di tale picco dipende dalla sua larghezza, cioè da Δf, oltre che dall'ampiezza della funzione sinusoidale ed esistono diverse possibilità di normalizzazione che in questa nota non intendiamo descrivere. Anche in considerazione del fatto che generalmente i nostri segnali campionati sono misurati in unità arbitrarie (i digit di Arduino), per non appesantire troppo l'esercizio va benissimo se vi limitate a esprimere la $\tilde{g}(f)$ ottenuta per FFT in *unità arbitrarie*.

A. DFT e FFT

Dal punto di vista matematico, l'espressione di Eq. 1 è ricchissima di conseguenze e implicazioni, sia nella fisica classica che in quella quantistica (e relativistica). Tuttavia, come al solito, la fisica non è la patria degli integrali, e neanche delle derivate, e il calcolo esplicito dell'integrale che compare nella definizione può essere affrontato solo per casi "modello", quelli che interessano la soluzione di semplici, o semplicissimi, esercizi da libro di testo.

Esistono dei validi strumenti numerici che permettono di eseguire il calcolo in maniera *discreta*, cioè partendo da una $g(t)$ definita per punti, che dunque non è una funzione, ma rappresenta, per esempio, una misura campionata nel tempo (sarebbe più corretto indicarla come $g(t_i)$, ma evitiamo di farlo per non appesantire la presentazione); il risultato della trasformazione è anche uno spettro discreto (che dovrebbe essere indicato con $\tilde{g}(f_j)$), generalmente complesso. Questi strumenti, uno dei quali è appunto la FFT, sono presenti in tutti i software di analisi dati, compreso, ovviamente (e con diverse implementazioni che qui non discuteremo), Python. Se siete interessati, potete sicuramente trovare in rete, o altrove, informazioni su come funzionano gli algoritmi per la DFT e verificare come l'aggettivo fast della FFT sia giustificato da un metodo di calcolo molto efficiente in termini di risorse.

Prima di procedere, diamo qualche chiarimento sull'implementazione FFT in Python a cui facciamo esplicito riferimento. Il problema principale da risolvere riguarda la *scala* di f, cioè quanto valgono il minimo e il massimo valore delle f da usare come asse orizzontale per lo spettro. Infatti, se l'Eq. 1 non dà alcuna indicazione sulla presenza di limiti in f, il calcolo numerico (discreto) impone delle regole precise.

Senza entrare nei dettagli, le regole principali sono le seguenti:

1. la FFT funziona al meglio, cioè senza la necessità di applicare artifici particolari, se l'array di partenza,

quello che rappresenta la $g(t)$ (in forma discreta, per punti), è composto da 2^n punti, con n intero. Vedete che, spesso, questo requisito è automaticamente soddisfatto quando i dati di partenza sono quelli acquisiti con una delle varie combinazioni di sketch e script che abbiamo impiegato con Arduino nel corso dell'anno.

2. Nell'implementazione da noi usata, l'array trasformato è fatto da $2^{n-1}+1$ punti. Supponendo array di partenza costituiti da $2^{11} = 2048$ punti: la nostra FFT è rappresentata da array reali di 1025 punti (la metà più uno dei punti di partenza). Il punto "in più", quello che rende dispari questo valore, si riferisce a $|\tilde{g}(f=0)|$ e, per i nostri scopi, ha scarsa importanza. Esso infatti rappresenta il valore medio del segnale, calcolato nella durata complessiva dell'acquisizione e influenzato, ad esempio, da eventuali offset, bias, etc.. Osservate che, per come è definita la trasformata di Fourier, anche un piccolo offset, o bias, presente nel segnale può condurre a un valore molto alto del coefficiente spettrale per $f = 0$. Quindi non stupitevi se i vostri spettri presenteranno un grosso picco per la componente continua. Ancora, per migliorare la leggibilità dello spettro spesso può essere opportuno rappresentarli con l'asse verticale logaritmico, scelta che in genere permette di visualizzare nello stesso grafico dati che coprono un vasto intervallo di valori.

3. Lo spettro di Fourier ottenuto parte sempre da $f = 0$ e si estende fino a $f_{max} = 1/(2\Delta t_{eff})$, dove, per noi, Δt_{eff} rappresenta l'intervallo temporale effettivo tra due campionamenti successivi.

4. L'affermazione precedente accende un campanello di allarme: usando Arduino, l'intervallo Δt_{eff} è normalmente definito con un'accuratezza piuttosto limitata a causa dei (noti) limiti nella definizione e nella misura dei tempi intrinseci nel funzionamento del microcontroller. Nella sua implementazione ordinaria, l'algoritmo FFT richiederebbe dati equispaziati temporalmente, per cui l'uso di Arduino è inevitabilmente accompagnato da una incertezza, difficilmente valutabile, nella costruzione dello spettro.

5. La risoluzione in frequenza dello spettro ottenuto, cioè la distanza Δf (in unità di f) tra due punti consecutivi dello spettro, è ovviamente pari a $f_{max}/(2^{n-1})$. Poiché l'intervallo complessivo di durata dell'acquisizione è $\Delta T \simeq 2^n \Delta t_{eff}$, dove il simbolo $\simeq$ può essere sostituito per i nostri scopi dal simbolo $=$ (vero se, in pratica, trascuriamo le incertezze nella misura dei tempi di cui sopra), si ha anche $\Delta f = 1/\Delta T$. Nella pratica conviene servirsi proprio della misura di ΔT per stabilire la scala in frequenza dei nostri spettri, assumendo implicitamente che il campionamento sia equispaziato temporalmente.

6. Sulla base di quanto appena affermato, è evidente che la risoluzione in frequenza migliora all'aumentare della durata temporale del record di dati. Visto che nelle nostre acquisizioni operiamo tipicamente con un intervallo nominale di campionamento relativamente breve (decine o centinaia di μs, in genere), costruire un record di durata sufficiente per garantire una ragionevole risoluzione in frequenza implica acquisizioni di record relativamente lunghi, cioè costituiti da un numero relativamente grande di dati. Come sappiamo, questo può essere facilmente realizzato (con segnali ripetitivi) usando opportune strategie di sincronizzazione per Arduino. Naturalmente anche in questo caso non ha senso esagerare: pretendere una risoluzione in frequenza molto elevata cozza con l'incertezza nella determinazione dei tempi tipica di Arduino. Inoltre, in particolare nei casi in cui ci si aspetta la presenza di picchi relativamente "larghi" nello spettro, per esempio per gli oscillatori RLC fortemente smorzati, può essere sufficiente anche esaminare acquisizioni con un ridotto numero di punti, fino a soli 256 punti.

7. In generale, non esiste un criterio che permetta di aggiustare a priori il rate di campionamento, ovvero l'intervallo Δt_{nom}, allo scopo di ottenere la "migliore" FFT. L'applicazione di un famoso teorema, detto *di Nyquist*, suggerisce come nel caso di funzioni periodiche, che sono quelle di nostro interesse in questo semplice esercizio, bastino tre punti per ciclo per ottenere informazioni rilevanti sulle periodicità. Naturalmente il consiglio è quello di campionare almeno una decina di punti per ogni ciclo. Notate che forti sovra-campionamenti, cioè avere un gran numero di punti per ogni ciclo, unito alla lunghezza relativamente piccola dei record acquisiti, può condurre ad avere spettri poco risolti in frequenza, Viceversa, forti sotto-campionamenti, in cui solo pochi punti sono acquisiti per ogni ciclo, può condurre ad artefatti di diverso tipo, il più frequente dei quali consiste nell'ottenere picchi spettrali "frastagliati".

8. Per un utente esperto sono a disposizione diverse strategie finalizzate a controllare numericamente, entro certi limiti, l'esito della FFT. La più impostante riguarda l'impostazione della cosiddetta "finestra". La funzione finestra rappresenta in pratica il "peso" che viene attribuito ai vari punti che costituiscono la $g(t)$ nell'operazione numerica di trasformata. Nella configurazione più semplice, quella consigliata per lo svolgimento di questo esercizio, la finestra è piatta ("rettangolare"), cioè tutti i punti vengono considerati con lo stesso peso. È probabile che in futuro vi troverete a usare finestre di altro genere, che possono essere utili per ridurre la comparsa di eventuali artefatti nello spettro.

Dal punto di vista pratico, poiché le grandezze campionate sono, nel nostro caso, *reali* e il nostro interesse è quello di determinare il *modulo* della trasformata di Fourier, il comando di Python che possiamo usare è `abs(numpy.fft.rfft(V))`, dove V rappresenta l'array di dati della nostra misura (il segnale) e `abs` indica l'operazione di calcolo del modulo dell'array complesso prodotto dalla FFT. Questo comando genera un nuovo array, che possiamo graficare in funzione di un array, da costruire a parte, che riporta i valori discreti della frequenza f determinati come sopra specificato. Il grafico ottenuto rappresenta lo spettro $\tilde{V}$ del segnale.

Se, invece, foste interessati a costruire lo spettro in potenza, allora il comando da impiegare sarebbe `(abs(numpy.fft.rfft(V)))**2`, che, di nuovo, produrrebbe un array da graficare in funzione dell'array f costruito come appena illustrato.

II. L'ESERCIZIO

All'esercizio si può rispondere con qualche grafico e qualche riga di commento (da caricare in formato pdf, possibilmente in unico file, sulla pagina di e-learning del corso). L'esericizio richiede di calcolare tramite FFT gli spettri di qualche acquisizione realizzata con Arduino nel corso delle esperienze pratiche. Per i motivi prima esposti, si consiglia di impiegare acquisizioni con record lunghi, per esempio quelle create dalla combinazione di sketch e script `synclong2016`, `harmlong` e `transosc`. Normalmente queste combinazioni creano record composti da 2048 coppie di dati, tempo (in μs) e valore digitalizzato (in unità arbitrarie, ovvero, nel nostro linguaggio, digit): la trasformata di Fourier di questi dati dà luogo ad un array composto da 1025 punti, compreso lo zero. Tuttavia in alcuni casi, per esempio per lo studio degli oscillatori RLC, può essere sufficiente esaminare record da soli 256 punti.

L'esercizio può essere compiuto su qualsiasi file in vostro possesso che rappresenti l'andamento temporale di un segnale, con caratteristiche preferibilmente note e riferibili a precise situazioni fisiche di interesse nel corso (da specificare nei commenti).

Qualche esempio di dati che potrebbero essere impiegati è elencato qui nel seguito.

A. Forme d'onda

Nelle prime esperienze in cui abbiamo usato Arduino ci siamo posti il problema di verificare la ricostruzione dei segnali periodici prodotti dal generatore di forme d'onda. Noi già conosciamo il metodo, quello della serie di Fourier, adatto per determinare i coefficienti dell'espansione per forme d'onda "semplici". Come ricordate, in un precedente esercizio abbiamo proprio usato la serie di Fourier per ricostruire numericamente delle forme d'onda periodiche. Qui la finalità dell'esercizio è diversa: si parte da dati "reali", necessariamente fisici, cioè dotati di "imperfezioni" dovute al generatore o al processo di campionamento/digitalizzazione, e di questi dati si calcola la FFT.

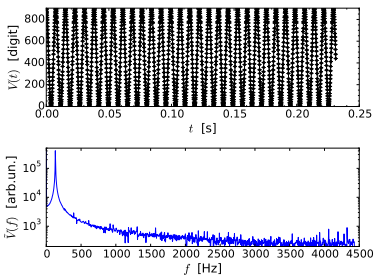

Figura 1. Segnale $V(t)$ e spettro FFT, $\tilde{V}(f)$, corrispondente per una forma d'onda sinusoidale prodotta dal generatore di forme d'onda e acquisita in record lunghi da 2048 punti. Il segnale è rappresentato con punti, dotati delle barre di errore convenzionali, raccordati da una linea continua, che serve solo da guida per gli occhi. Lo spettro è rappresentato in unità arbitrarie e in scala logaritmica: per lo spettro non è riportata alcuna barra di errore, che sarebbe complicato (e poco opportuno) determinare, e il grafico è realizzato impiegando una linea continua.

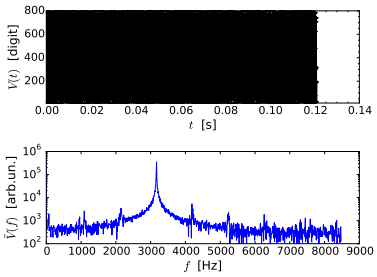

Figura 2. Analogo di Fig. 1 per un'altra forma d'onda sinusoidale, evidentemente acquisita in modo da registrare un grande numero di cicli (l'andamento della $V(t)$ nel pannello superiore risulta completamente non intelligibile). Notate l'aumento nella risoluzione in frequenza dello spettro.

Per intenderci, se avete un record che rappresenta la digitalizzazione di un'onda quadra o triangolare, la FFT dovrebbe mostrare un picco principale alla frequenza dell'onda stessa (a parte il contributo della frequenza zero, che corrisponde alle componenti continue di offset o bias, e che, per i nostri scopi, non è rilevante), più altri picchi minori che corrispondono alle diverse armoniche. La frequenza di questi picchi dovrebbe essere nel corretto rapporto con la frequenza fondamentale e anche la loro ampiezza dovrebbe soddisfare le aspettative (ripensate all'espansione in serie di seni e coseni per queste forme d'onda: vedrete che le armoniche presenti dovrebbero essere solo le dispari e che i coefficienti dovrebbero seguire un andamento specifico per la forma d'onda). Aveste anche un'acquisizione di "pinna di squalo" con un numero

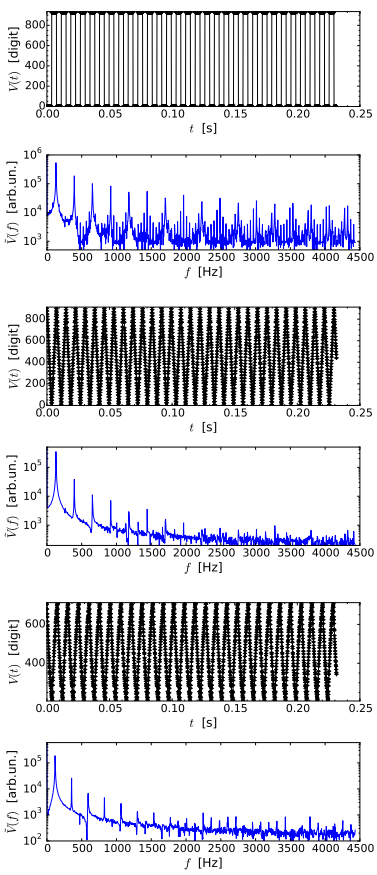

Figura 3. Analogo di Fig. 1 per una forma d'onda quadra, una triangolare e una a "pinna di squalo" acquisita nell'esperienza pratica con il circuito RC.

sufficiente di punti, potrebbe anche essere interessante studiarne la trasformata di Fourier.

Invece l'acquisizione di una forma d'onda sinusoidale dovrebbe mostrare solo un picco spettrale, alla frequenza dell'onda stessa. In linea di principio, questo picco dovrebbe apparire "strettissimo" (in matematica una "delta"), a testimonianza del carattere monocromatico della forma d'onda sinusoidale.

I condizionali che ho volutamente impiegato stanno a ricordare che la realtà sperimentale è, appunto, una realtà: infatti potrebbero essere presenti dei rumori sovrapposti al segnale (tipica è la presenza di rumori attorno alla frequenza di rete o suoi multipli), oppure potrebbero diventare rilevanti eventuali effetti legati al campionamento (fluttuazioni nell'intervallo di campionamento, errori sistematici nella digitalizzazione da parte di Arduino, etc.). Infine, poiché le forme d'onda sono anch'esse "reali", cioè prodotte da un dispositivo reale quale il generatore di forme d'onda, esse potrebbero presentare differenze rispetto alle aspettative, per esempio la presenza di armoniche inattese e, soprattutto, una larghezza finita dei picchi spettrali (la monocromaticità prevista dalla matematica non è mai possibile nella realtà).

Le Figs. 1-3 presentano alcuni esempi, che sono riportati qui senza commento (nel vostro esercizio mi aspetto qualche commento in più, almeno sui parametri di acquisizione, che devono essere specificati, e sulla frequenza della forma d'onda acquisita). Il segnale campionato è denominato $V(t)$ e rappresentato in unità arbitrarie di digitalizzazione (digit).

B. Oscillatore smorzato

Un valido esercizio per apprezzare la bellezza dell'analisi di Fourier è costituito dalla trasformazione del segnale acquisito con l'oscillatore armonico smorzato RLC. Nel nostro corso analizziamo il comportamento di questo circuito prima nel dominio dei tempi, ottenendo dei segnali rappresentativi di un'oscillazione smorzata, e poi nel dominio delle frequenze, ricostruendo la funzione di trasferimento del circuito e analizzando il suo andamento al variare della frequenza.

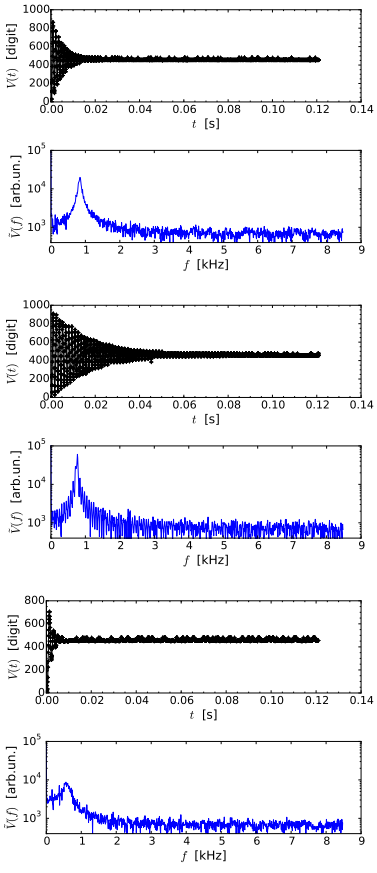

Figura 4. Analogo di Fig. 1 per segnali acquisiti nelle esperienze con l'oscillatore smorzato RLC operato in varie condizioni.

La trasformata di Fourier permette di passare da un dominio all'altro. Dunque, almeno tralasciando le "imperfezioni" dovute al campionamento, il modulo dello

spettro FFT del segnale acquisito in determinate condizioni deve riprodurre l'andamento del guadagno, o attenuazione, nelle stesse condizioni. Potete facilmente verificarlo, osservando in particolare come lo spettro ottenuto sia piccato attorno alla frequenza di risonanza e come la sua larghezza dipenda dal fattore di qualità, ovvero dallo smorzamento dell'oscillatore: la larghezza del "picco di risonanza" nello spettro è tanto maggiore quanto più smorzato è l'oscillatore.

Anche in questo caso si riportano alcuni esempi, volutamente senza commento (ma nei vostri esercizi mi aspetto commenti e qualche indicazione sui parametri sperimentali), nelle Figs. 4, 5. Notate che in alcuni casi sono state impiegate delle acquisizioni costituite da soli 256 punti, con ovvie conseguenze sulla risoluzione in frequenza dello spettro.

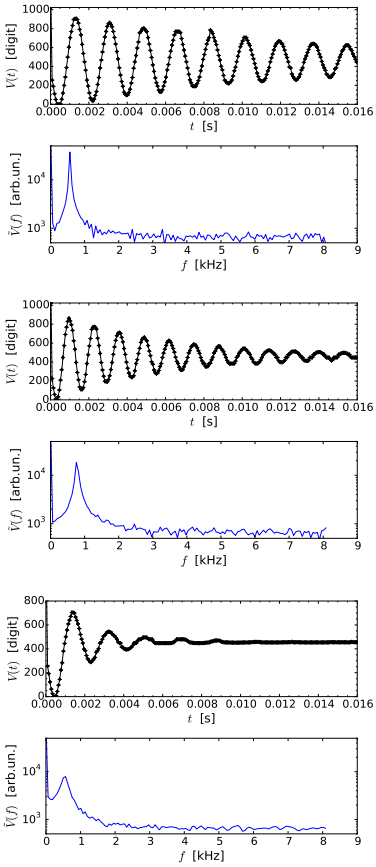

Figura 5. Analogo di Fig. 4: gli spettri sono qui costruiti a partire da acquisizioni composte da soli 256 punti, con ovvia perdita di risoluzione in frequenza negli spettri.

C. Oscillatore a reazione (con transistor)

Infine, un'altra interessante occasione di impiego della FFT è con l'oscillatore a reazione basato su transistor BJT a emettitore comune e rete di sfasamento per ottenere un feedback positivo. In queste condizioni la d.d.p. oscillante che si ottiene in uscita è attesa avere una forma "strana", che assomiglia solo lontanamente a una sinusoide a causa dei meccanismi di operazione dell'oscillatore. Di tutto questo è fatto cenno in un'altra nota.

In questa sede ci limitiamo a presentare in Fig. 6 alcuni esempi acquisiti in diverse condizioni di operazione. Anche qui rinunciamo a specificare commenti (che però dovrebbero essere presenti, assieme alla descrizione dei parametri sperimentali, nel vostro esercizio).

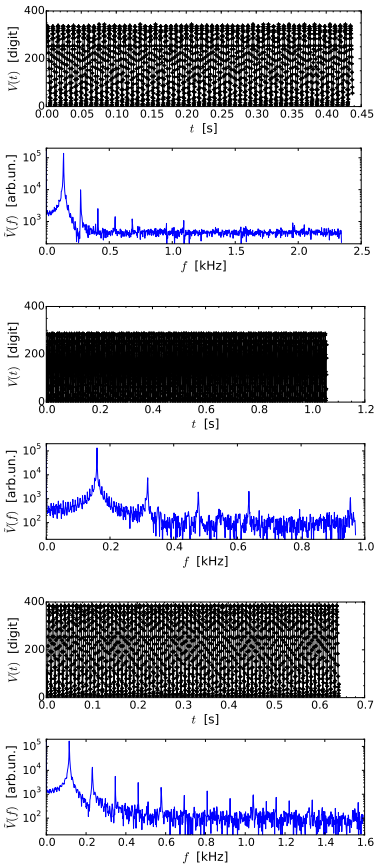

Figura 6. Analogo di Fig. 1 per segnali acquisiti in oscillatori a reazione (con transistor) operati in diverse condizioni.