

SETTIMANA MATEMATICA

Laboratorio “Matematica, Musica, e Computer”

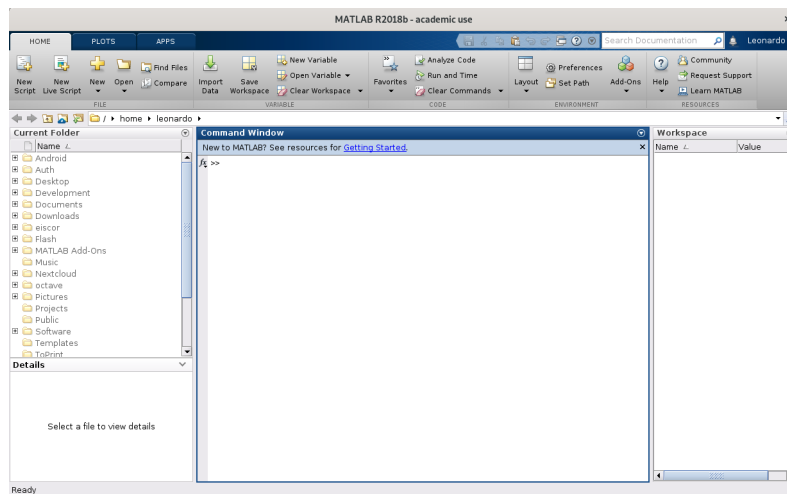
Parte 1: Costruire un sintetizzatore

20 – 22 aprile 2022

1 Primo incontro con MATLAB

In questo laboratorio ci occuperemo di produrre ed analizzare musica tramite il computer, e per questo avremo bisogno di realizzare dei programmi. Per facilitarci il compito, utilizzeremo un linguaggio pensato apposta per matematici, che facilita di molto alcune operazioni. Questo linguaggio si chiama MATLAB¹.

Sui vostri computer potete aprire MATLAB premendo CTRL+ALT+T per aprire un terminale e poi scrivendo `matlab` seguito da invio, o in alternativa premendo ALT+F2 e di nuovo scrivendo `matlab` e premendo invio. Vi si aprirà una finestra simile a questa:



Nella parte centrale potete inserire comandi per effettuare semplici operazioni aritmetiche. Ad esempio:

```
>> 2 + 3
```

```
ans =
```

¹MATLAB è anche il nome del software, a pagamento e piuttosto costoso, che permette di eseguire programmi scritti in questo linguaggio. Tuttavia, esiste un software gratuito che possiamo utilizzare in alternativa di nome GNU/Octave.

Potremmo voler assegnare l'output ad una variabile, ovvero dargli un nome che ci possa tornare utile in seguito². Supponiamo di volerla chiamare `x`:

```
>> x = 2 + 3

x =

5
```

Se non vogliamo vedere il risultato stampato a schermo, possiamo aggiungere un `;` alla fine per sopprimerlo. La seguente istruzione è equivalente:

```
>> x = 2 + 3;
>>
```

Quando dovremo rappresentare i suoni come sampling a tempi discreti di una funzione $s(t)$, avremo bisogno di *vettori*, ovvero di sequenze (finite) di numeri reali. Questi possono essere definiti in MATLAB utilizzando i simboli `[ e ]`. Ad esempio, per generare il vettore che contiene gli interi da 1 a 10 potremmo usare la seguente sintassi:

```
>> x = [ 1 2 3 4 5 6 7 8 9 10 ];
```

A volte per maggior chiarezza è preferibile separare le varie entrate di un vettore con delle virgole, ottenendo la seguente istruzione con l'analogo risultato:

```
>> x = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ];
```

I simboli `[ e ]` si possono anche usare per concatenare vettori. Ad esempio, provate ad eseguire i seguenti comandi, cosa succede?

```
>> v1 = [ 1, 2, 3 ];
>> v2 = [ 4, 5, 6 ];
>> v = [ v1, v2 ]
```

Per accedere agli elementi di un vettore possiamo usare le parentesi tonde:

```
>> v(4)

ans =

4
```

Notiamo che MATLAB comincia a contare gli elementi dei vettori da 1, invece che da 0 come fanno molti linguaggi di programmazione.

²In MATLAB le variabili non vanno dichiarate prima di utilizzarle.

1.1 Funzioni

MATLAB mette a disposizione alcune *funzioni*, che probabilmente saranno familiari a chi di voi abbia visto un po' di programmazione. Dal punto di vista astratto, le funzioni dei linguaggi di programmazione sono molto simili alle funzioni che siamo abituati a considerare come matematici: prendono come input uno o più valori (o, talvolta, nessuno), e restituiscono (uno o più) output. Come vedremo, la sintassi è in effetti molto simile a quella che utilizziamo in matematica.

Noi perlopiù useremo funzioni come semplici utenti, alcune fornite da MATLAB, ed alcune scritte per questo laboratorio. Facciamo un esempio: esiste la funzione `length` che, come il nome suggerisce, permette di determinare la lunghezza di un vettore. Proviamo a dare il seguente comando:

```
>> length(v)
```

```
ans =
```

```
6
```

Volendo, possiamo anche assegnare il risultato ad una variabile, con il comando `n = length(v);`. In generale, una funzione in MATLAB di nome `funzione` si trova definita all'interno di un file `funzione.m`. La lista degli argomenti (o parametri) che questa funzione prende in input va passata fra parentesi tonde.

Se siete interessati, potete provare a curiosare nei file `.m` che saranno messi a disposizione per questo laboratorio, per vedere le operazioni matematiche che si nascondono all'interno.

Prima di procedere, osserviamo un altro tipo di istruzione che ci potrà essere molto utile. Per creare un vettore che contenga una progressione aritmetica del tipo

$$x_0, x_0 + h, x_0 + 2h, \dots, x_0 + kh = x_k,$$

per qualche k intero e $h \in \mathbb{R}$, possiamo scrivere

```
>> x = x_0 : h : x_k;
```

Esercizio 1.1. *Si provi a creare, utilizzando la sintassi sopra, il vettore di tutti i numeri dispari fra 1 e 31 (estremi compresi).*

2 Visualizzare le frequenze di vari strumenti

Scarichiamo le funzioni che ci serviranno per il laboratorio da <https://leonardo.robol.it/teaching/matematica-e-musica-2022/>. Possiamo estrarre il file `.zip` e salvare i file in una cartella. Dopodiché, spostiamoci con il file browser di MATLAB (che trovate nella parte sinistra della finestra) in quella cartella.

Ricordiamoci ora che ogni suono periodico che “suona” una nota con frequenza ω , si può scrivere in serie di Fourier come

$$s(t) = \frac{a_0}{2} + \sum_{i=1}^{\infty} a_i \cos(2\pi i \omega t) + b_i \sin(2\pi i \omega t).$$

Determinare i coefficienti a_i, b_i richiede il calcolo di alcuni integrali, ma è un'operazione che può essere efficientemente approssimata da un computer. Inoltre, dal punto di vista dell'acustica possiamo immaginare che la serie sia composta solo da seni (o solo da coseni): sebbene la funzione risultante sia diversa, il nostro orecchio non è in grado di percepirlo.

Nei file che avete scaricato troverete una funzione `analizza_suono` che permette, dato un file audio, di ricavare i coefficienti di Fourier di una piccola parte iniziale. Dato che il nostro orecchio non distingue seni e coseni, possiamo immaginare che i coefficienti ottenuti siano tutti relativi ai seni (per semplicità la funzione riporterà solo questi).

Esercizio 2.1. Si ascolti il file `flute_A4.wav` (basta farci doppio clic sopra usando il programma di gestione dei file, che trovate nella barra delle applicazioni in alto a sinistra), che contiene una breve registrazione di un flauto, che suona un La a 440 Hz. Si utilizzi la funzione `analizza_suono` per visualizzare la forma d'onda e i principali coefficienti di Fourier. La sintassi da utilizzare è `analizza_suono('nomefile.wav');`

La funzione identifica il valore dei primi coefficienti non nulli se si indica anche una frequenza di base da cui cercarli. Ad esempio, `c = analizza_suono('nomefile.wav', 440);` restituisce il vettore di coefficienti `c` per il flauto.

Il diapason è uno dispositivo solitamente utilizzato per accordare gli strumenti a corda, ed ha la particolarità di produrre un suono molto simile alla funzione $\sin(2\pi\omega t)$, con $\omega = 440$ Hz. Proviamo a riprodurre il suono con MATLAB. Per fare questo, avremo bisogno di generare una versione discreta della funzione $\sin(2\pi \cdot 440 \cdot t)$, valutandola per alcuni tempi distanziati di una distanza Δt , ad esempio di $\frac{1}{44100}$ secondi.

Esercizio 2.2. Realizzare un suono di un secondo di un diapason, seguendo i seguenti passi:

- Generare un vettore t che contenga gli elementi:

$$t = [0 \quad \Delta t \quad 2\Delta t \quad \dots \quad 1], \quad \Delta t := \frac{1}{44100}.$$

Questo sarà possibile definendo `dt = 1/44100` e poi utilizzando la sintassi `0 : dt : 1`.

- Valutiamo la funzione $\sin(2\pi \cdot 44100 \cdot t)$ su tutti gli elementi di questo vettore, con il comando

```
>> s = sin(2*pi*440*t);
```

dove t è il vettore definito in precedenza. MATLAB permette di applicare una funzione ad un vettore, e la valuta in tutti i suoi elementi. La costante π è inoltre disponibile nella variabile `pi`.

- Suonare il diapason con il comando `sound(s, 44100);`

I diapason non sono strumenti molto interessanti, ma abbiamo già identificato i coefficienti di Fourier del flauto nell'esercizio 2.1. Possiamo provare a ricostruirlo a partire da quei coefficienti. In effetti, avendo memorizzato i coefficienti $c_1, c_2, \dots, c_8$, possiamo provare a suonare un La creando il suono

$$s(t) = c_1 \sin(2\pi \cdot 440 \cdot t) + c_2 \sin(2\pi \cdot 880 \cdot t) + \dots + c_8 \sin(2\pi \cdot 440 \cdot 8 \cdot t).$$

Esercizio 2.3. Utilizzando i coefficienti determinati precedentemente, simulare una nota La eseguita dal flauto, in maniera analoga a quanto fatto con il diapason. Potete usare solo i primi coefficienti, e non inserirli tutti, e vedere come il suono migliora.

Esercizio 2.4. Si provi a simulare un organo con lo stesso sistema utilizzato per il flauto, partendo dal file `great-organ_A3.wav`. Come il nome suggerisce, in questa registrazione l'organo produce una nota con frequenza fondamentale 220 Hz, un'ottava sotto a quella del flauto. Si provi anche con gli archi registrati in `strings_A3.wav` (molto più difficili da riprodurre fedelmente).

3 Suoniamo Fra Martino

Per semplificare la sistematica ripetizione delle operazioni fatte nell'esercizio 2.3, potete provare a utilizzare la funzione `crea_suono`. Questa prende come input un vettore con i coefficienti della serie di Fourier, la nota da suonare, e la durata del suono. Come output, restituisce il vettore `s` da poter dare in input alla funzione `sound`.

Ad esempio, potremmo risolvere l'esercizio 3 con i seguenti comandi:

```
>> c = [ c1, c2, c3, c4, c5, c6, c7, c8 ];  
>> s = crea_suono(c, 'La5', 1);  
>> sound(s, 44100);
```

La notazione “La4” indica che la nota da eseguire è un La, sull’ottava del La centrale (ovvero quello a 440 Hz). Su uno spartito si tratta della seguente nota:



Vogliamo ora fare un altro passo avanti, e scrivere un piccolo programma che ci permetta di riprodurre dei brani musicali da noi sintetizzati. A questo scopo, possiamo creare un nuovo file `musica.m` (il file si può chiamare come vogliamo, l'importante è che l'estensione sia `.m`), e inseriamo al suo interno una lista di comandi che vogliamo eseguire in sequenza.

Ad esempio, per riprodurre le note Do, Re, Mi potremmo realizzare il seguente file:

```
c = [ c1, ..., c8 ]; % Coefficienti per il flauto  
  
s = []; % Cominciamo con un vettore vuoto  
  
% ... e aggiungiamo vari pezzi che suonano tutte le note, in sequenza.  
s = [ s, crea_suono(c, 'Do5', 1) ];  
s = [ s, crea_suono(c, 'Re5', 1) ];  
s = [ s, crea_suono(c, 'Mi5', 1) ];  
  
sound(s, 44100);
```

Possiamo poi eseguire il file dando il comando `musica`.

Attenzione: Il comando `sound` manda il suono in esecuzione e ritorna subito, per cui non è possibile dare una sequenza di comandi come:

```
>> sound(crea_suono(...), 44100);  
>> sound(crea_suono(...), 44100);  
>> sound(crea_suono(...), 44100);
```

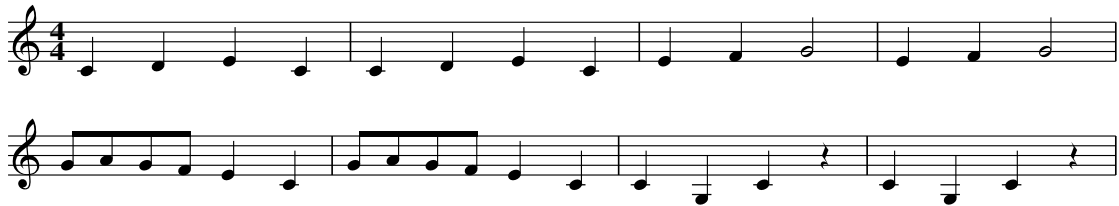
Questi verrebbero eseguiti quasi in contemporanea, generando una (probabilmente orribile) sovrapposizione di suoni. È necessario invece assemblare completamente il vettore `s` allungandolo con i vari pezzi ottenuti dalla funzione `crea_suono`, ed eseguire `sound` solo alla fine.

Esercizio 3.1. Si scriva un file `fra_martino.m` che esegua il canone Fra Martino, il cui spartito è riportato qui sotto. Nel caso si scelga come strumento il flauto, conviene partire dal do sopra il do centrale, dove il suono è un pochino più realistico.

	c_1	c_2	c_3	c_4	c_5	c_6	c_8	c_{16}
A	d	d	d	m	m	d		
E	d		d			f		
I	m	d				d		m
O	m	f	m	d				
U	f	m	d					

Tabella 1: Intensità dei coefficienti di Fourier all'interno delle vocali. Il simbolo f sta per “forte”, m per “medio”, e d per “debole”.

Fra Martino



Si provi poi a far suonare la stessa melodia all'organo approssimato in precedenza, ma ora trasponendola in giù di un'ottava, partendo dal Do centrale (Do4), in cui è plausibile che la nostra approssimazione dell'organo sia un po' più credibile.

4 Suonare le vocali

Nel 1857 il fisico Hermann Ludwig Ferdinand von Helmholtz si propose di dimostrare che, tramite una opportuna sovrapposizione di armoniche (ovvero di seni e coseni di serie di Fourier) era possibile replicare il tono delle vocali.

Immaginò un esperimento dove un cantante lirico produce una vocale dopo l'altra vicino ad una pianoforte, e qualcuno misura le vibrazioni trasmesse alle varie corde per distinguere le varie componenti del suono³.

In effetti, Helmholtz realizzò un esperimento molto simile a quello descritto, sostituendo il pianoforte con delle opportune sfere risonanti, e determinò qualitativamente l'intensità dei coefficienti della varie armoniche all'interno delle vocali. I suoi risultati si possono riassumere nella seguente Tabella 1 (si veda [1]).

Helmholtz si spinse anche oltre, e realizzò un dispositivo che permettesse di simulare le vocali combinando diapason con frequenze diverse, tutte multiple della stessa frequenza base, e la cui intensità di suono fosse regolabile. In sostanza, aveva inventato una specie di serie di Fourier meccanica!

Esercizio 4.1. *Si provi a replicare l'esperimento di Hemholtz, ma lavorando al computer invece che costruendo uno strumento con dei diapason. Si provi a costruire i coefficienti di Fourier delle vocali, e farle “suonare”. L'intensità esatta di “forte”, “medio”, o “debole” non era specificata Hemholtz, quindi ci sarà bisogno di fare qualche prova!*

³Una corda tesa ha una sua frequenza caratteristica, e se sottoposta ad una vibrazione con la stessa frequenza comincerà a vibrare essa stessa, un fenomeno noto come *risonanza*.

5 Terzo suono di Tartini

Nel 1714 il violinista Giuseppe Tartini si accorse di un fenomeno curioso. Se vengono suonate due note insieme creando un intervallo di quinta (ad esempio, Do e Sol), allora verrà prodotta una terza nota (ora nota come *terzo suono di Tartini*).

In questo esempio, la nuova nota corrisponderà ad un altro Do all’ottava inferiore. Da cosa è causato questo fenomeno? Una spiegazione si può dare cercando di capire come il nostro orecchio identifica l’altezza di una nota; in pratica, l’orecchio umano “riconosce” i coefficienti di Fourier all’interno di un suono, ed identifica come altezza della nota la frequenza di quello più basso, di cui tutti gli altri sono un multiplo.

Ad esempio, se consideriamo il suono:

$$s(t) = \sin(2\pi\omega t) + \sin(4\pi\omega t) + \sin(6\pi\omega t), \quad \omega \approx 261.63 \text{ Hz},$$

il nostro orecchio lo riconoscerà come un Do centrale (“Do4” nella nostra notazione). Cosa succede se modifichiamo il suono levando la frequenza fondamentale, ovvero considerando la seguente funzione?

$$\hat{s}(t) = \sin(4\pi\omega t) + \sin(6\pi\omega t), \quad \omega \approx 523 \text{ Hz}.$$

Ora la frequenza più bassa è 2ω , invece che ω , ovvero il Do all’ottava sopra. L’altra armonica presente, però, ha frequenza 3ω , che non è un multiplo di 2ω . In questo caso, il nostro orecchio sarà portato ad “immaginare” una frequenza che in realtà non esiste, e che sia il massimo comun divisore fra le frequenze udite. Dato che 2 e 3 sono primi fra loro, questo significa che udiremo ω , ancora una volta la frequenza del do centrale.

Esercizio 5.1. *Si provi a verificare sperimentalmente il fenomeno del suono di Tartini, creando artificialmente un suono senza la fondamentale, e confrontandolo con un suono semplice. Un consiglio: per semplificare la creazione di $s(t)$ come sopra, possiamo utilizzare come serie di Fourier i coefficienti $[1 \ 1 \ 1]$, che sovrapporranno automaticamente le frequenze $\omega, 2\omega, 3\omega$, e utilizzare poi come vettore $[0 \ 1 \ 1]$ per levare la fondamentale.*

Riferimenti bibliografici

- [1] Mark R Petersen. Musical analysis and synthesis in matlab. *College Mathematics Journal*, pages 396–401, 2004.