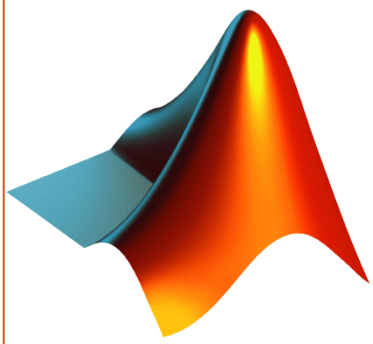
Università Politecnica delle Marche
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione



Analisi ed elaborazione dei dati su MATLAB

Nicola Fioranelli

Corso di Elettrotecnica
Prof. Simone Fiori
A.A. 2014-2015

SOMMARIO

PREMESSA

Matlab è una suite completa e molto potente per il calcolo, la visualizzazione e la programmazione. Sviluppato dall'americana Mathworks, il suo nome deriva dalla crasi di due termini MATrix LABoratory: ovvero laboratorio di matrici, letteralmente. L'applicativo, infatti, era stato originariamente concepito per l'algebra lineare e per poter lavorare con grande agilità sulle matrici, semplificando quindi tutto il lavoro di scrittura del codice (anche piuttosto complesso a volte) che si sarebbe dovuto fare con altri linguaggi di alto livello, tipo C++ o Pascal.

Tutto il software si basa su matrici ed array (ovvero vettori, che altro non sono che matrici 1xn o mx1); la gran parte delle operazioni viene quindi svolta su di esse.

Essendo tuttavia un programma di calcolo evoluto è stato concepito per essere utilizzato in due modi diversi: sia in modalità 'interattiva' ovvero inserendo comandi ed ottenendo risultati, sia tramite script e funzioni. Uno script (.m) o M-file è una sequenza di istruzioni che vengono eseguite da Matlab passo passo, quasi come un eseguibile: ciò consente di automatizzare una serie di calcoli, caricare dati da poter utilizzare anche in successive elaborazioni; una funzione, invece, è un particolare script che prende dati in ingresso, li elabora e li restituisce alla funziona stessa o allo script chiamante.

Insomma Matlab è un programma complesso per i più svariati utilizzi nell'algebra lineare e nella matematica ma anche in ambienti universitari legati all'ingegneria e di cui, in questo progetto, analizzeremo solo in parte gli aspetti correlati all'analisi e all'elaborazione dei dati.

IL PROGETTO

Questo progetto intende mettere in luce le capacità di acquisizione, elaborazione e visualizzazione dei dati di Matlab. Lo scopo è di migliorare un'immagine "sgranata" ottenuta graficando una matrice di valori; di questi valori analizzeremo quindi la distribuzione statistica prima e dopo le operazioni di elaborazione: esse consistono fondamentalmente nell'interpolazione, un metodo fondamentale per aggiungere informazioni in loro mancanza.

In seguito verificheremo anche le potenzialità di questo strumento nel far fronte alla mancanza di dati.

Utilizzeremo un altro metodo per interpolare che fa uso di operazioni statistiche e ne analizzeremo le differenze con interp1 e in termini di risultati ottenuti.

Otterremo infine l'angolo di rotazione da cui ricavare nuovamente la figura di partenza, ma questa volta, definitivamente "libera" dai dati iniziali.

Un'ultima breve parte la dedicheremo all'efficienza degli algoritmi, in particolare soffermandoci sull'analisi computazionale dei comandi ricorsivi 'for' e della diversa efficienza nelle operazioni sui vettori.

CARICAMENTO DEI DATI E PRIME OPERAZIONI

In Matlab è possibile caricare file e dati da diversi supporti: il nostro è un file binario .mat che caricheremo attraverso il comando `load` nel programma. Il database di informazioni viene caricato su una variabile che per default è `'x'` e che è una matrice di due colonne e numero variabile di righe in base a quale file si sceglie di caricare.

Per poter rappresentare i dati è necessario disporre di due vettori che possiamo ottenere usando alcune proprietà base delle matrici; essendo `'x'` una matrice verticale per comodità sarebbe opportuno ottenere due vettori in orizzontale.

```
X=x(:,1)';
Y=x(:,2)';
```

Con questi due comandi abbiamo cancellato ora la prima ora la seconda colonna al fine di ottenere due vettori separati per le ascisse e per le ordinate, da ultimo abbiamo fatto la trasposizione ottenendo quindi il vettore in orizzontale.

Il passo successivo è quello di graficare le coppie di punti (x, y): ciò è possibile grazie alla funzione `plot` che ci consente di visualizzare in una nuova finestra il grafico dei punti. Il comando viene scritto nel modo seguente: `plot(X,Y,LineSpec)` in cui x e y sono generalmente due vettori che rappresentano l'ascissa e l'ordinata del punto; infine con LineSpec andiamo a

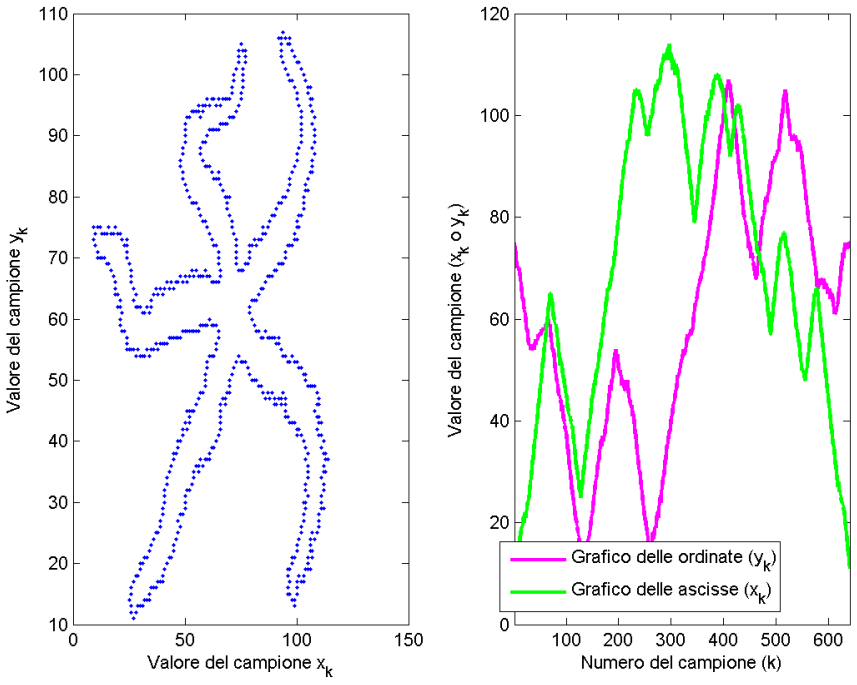


FIGURA 1 – A sinistra il grafico del database, a destra i valori delle ascisse e delle ordinate.

definire lo stile del grafico: il colore, il tipo di linea, la grandezza. Matlab è un software professionale, infatti plot consente di inserire sulla stessa figura diversi grafici andando semplicemente ad aggiungere le altre variabili di seguito.

Il primo grafico che otteniamo riporta sulla sinistra la rappresentazione delle coppie di valori del database e sulla destra il grafico delle ascisse e delle ordinate separatamente. E' possibile ottenere due grafici all'interno di una stessa finestra usando il comando `subplot`. (Figura 1)

ANALISI STATISTICA DEI DATI

Utilizzando il comando `hist(X, n)` è possibile ottenere la distribuzione statistica di un certo dato X ripartita in n intervalli uguali fra loro (in gergo 'bin'). Sfruttando la funzione è possibile vedere che le ascisse sono concentrate intorno ai valori del 60 e del 100 aree del grafico che corrispondo ad una maggiore densità di punti; analogo vale per le ordinate la cui maggior densità è possibile trovarla intorno ai valori del 70. L'analisi mostra una distribuzione non Gaussiana per le ascisse e del tutto asimmetrica, mentre per le ordinate (seppur in modo imperfetto) è possibile intravedere la presenza della campana di Gauss.

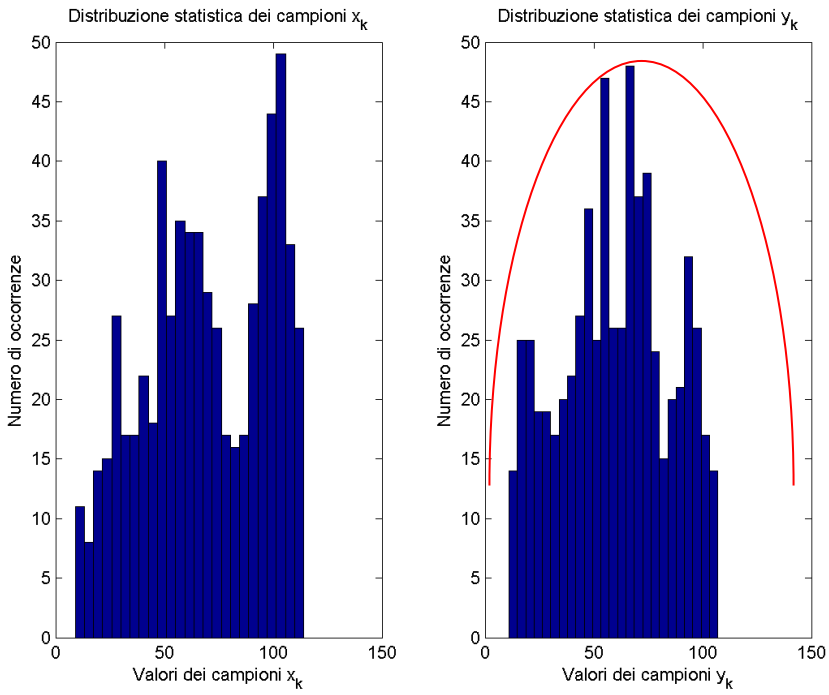


FIGURA 2 - A sinistra distribuzione statistica delle ascisse, a destra delle ordinate.

(Figura 2)

DECIMAZIONE O INTERPOLAZIONE?

Lo scopo del progetto, come anticipato all'inizio, è quello di realizzare un grafico della figura meno 'sgranato' ovvero avere più punti che consentano di ottenere un contorno più definito. L'interpolazione è un metodo per individuare nuovi punti del piano cartesiano a partire da un insieme finito di dati, come nel nostro caso; tuttavia un grafico meno 'sgranato' è possibile ottenerlo anche attraverso uno stratagemma molto astuto che però aggira l'ostacolo senza superarlo: la decimazione!

`px2 = X(1:4:n); py2 = Y(1:4:n);` questi due comandi ci consentono di prendere un valore ogni 4 sia per le x che per le y andando ad ottenere un grafico con meno punti che se collegati fra loro danno effettivamente una forma più lineare.

LA COORDINATA CURVILINEA E LA FUNZIONE INTERP1

L'interpolazione per poter funzionare ha bisogno di una successione di valori strettamente monotona, cosa che al momento non abbiamo dato che i vettori X ed Y sono coordinate di punti non necessariamente in ordine.

Per poter interpolare in modo funzionale abbiamo bisogno di introdurre una nuova variabile che ci tornerà utile nei prossimi calcoli: la coordinata curvilinea. Fin'ora abbiamo avuto nei vettori X ed Y solo una serie di coordinate discrete e non legate fra loro; la coordinata curvilinea ci consentirà di passare dall'insieme dei naturali (che rappresentava l'indice delle coppie di punti (x_k,y_k)) all'insieme dei reali: infatti essa è la curva che collega tutti punti ovvero una distanza.

Troviamo la coordinata curvilinea nel modo seguente:

```
for k=2:1:n;
    st(1,k)=st(1,k-1)+sqrt( (X(1,k)-X(1,k-1))^2 +
(Y(1,k)-Y(1,k-1))^2 );
end
```

in cui `st` è il vettore della coordinata curvilinea che abbiamo inizializzato prima a 0.

La funzione interp1 ha la seguente sintassi:
`xi=interp1(st,X,si,'Method');`

Di fatto vado a considerare le X in funzione della coordinata curvilinea, `si` è un vettore di punti generici e `xi` è il vettore delle ascisse interpolate (cioè una serie di nuovi punti che andranno ad aumentare la precisione del grafico), infine Method è il metodo da utilizzare nell'interpolazione:
`'linear'` – è l'interpolazione lineare che tra due punti crea un segmento rettilineo;
`'spline'` – è un tipo di interpolazione che fa uso delle curve ed è naturalmente più precisa di linear anche se consuma più memoria.

Una volta interpolate le x e le y ottengo nuove coppie di punti che vanno a migliorare il grafico.
(Figura 3)

Come già anticipato sopra, nel codice abbiamo inserito di nuovo la funzione `hist` ma questa volta con i dati interpolati: il grafico è molto più equalizzato infatti i 'bin' in cui vi erano pochi punti sono stati riempiti con altri punti (ovviamente presi con uguale distanza) e gli altri più pieni si sono riconfermati tali.

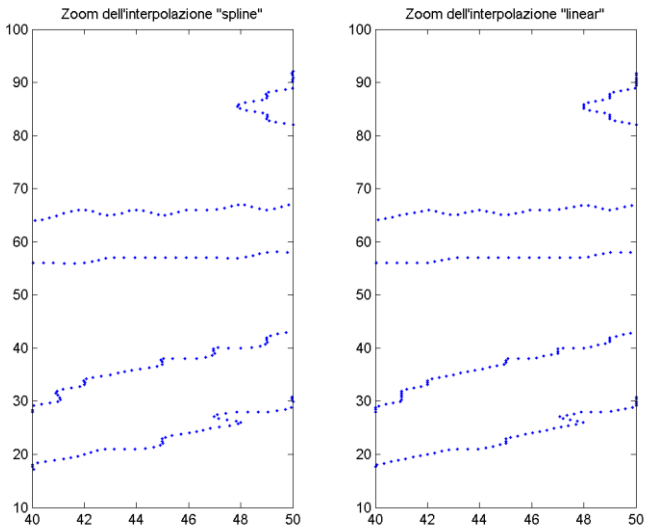


FIGURA 3 - Dettaglio delle interpolazioni: a sinistra i contorni sono più "morbidi", a destra 'linear' da una ricostruzione più spigolosa.

RICOSTRUZIONE DI DATI MANCANTI

La funzione `interp1` è molto complessa e potente in realtà; fin'ora l'utilizzo che ne abbiamo fatto è stato quasi elementare: in questo breve paragrafo illustreremo uno dei moltissimi campi di utilizzo di Matlab.

Spesso capita che durante la raccolta dei dati o nel loro caricamento ci possano essere dei problemi che compromettono la validità di una parte di essi: `interp1` è una funzione che ci permette di ricostruire, se pur non nel modo originario, quella perdita di dati causata da vari fattori.

Un esempio è stato implementato nel nostro codice. Eliminando parte del vettore delle x e delle y si ottiene sostanzialmente un grafico con un 'buco' di informazioni: la funzione riesce a colmarlo tracciando un'approssimazione del grafico. Salta subito all'occhio una considerazione: tanto più esteso sarà il danno e tanto meno sarà possibile ricostruirlo con efficacia. (Figura 4)

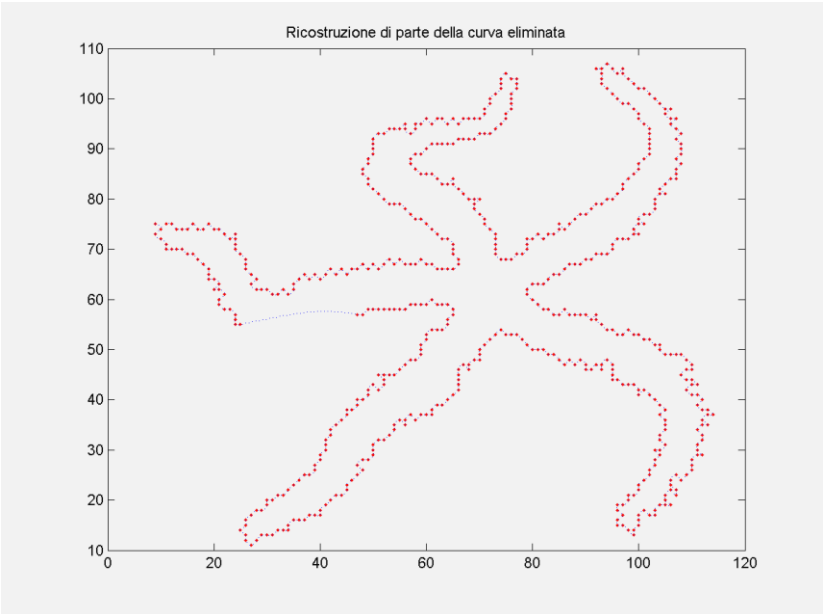


FIGURA 4 - Ricostruzione di parte della curva eliminata

ANGOLO DI ROTAZIONE

Fin'ora abbiamo cercato dei modi per poter graficare e migliorare la rappresentazione dei dati: lo scopo di quest'ultima operazione è cercare di "allontanarci" dalla limitatezza dei dati discreti in origine al fine di ottenere nuove informazioni da utilizzare più agilmente.

Lo strumento che utilizzeremo è la matematica, in particolare la trigonometria, grazie alla nozione delle coordinate polari. Un sistema di coordinate polari è un sistema di coordinate bidimensionale nel quale ogni punto del piano è identificato da un angolo e da una distanza da un punto fisso detto polo. Questo modo di lavorare ci permette di mantenere invariata la struttura di base del grafico anche in seguito a ingrandimenti o riduzioni (è molto utile nel riconoscimento facciale, dove potrebbe capitare talvolta di avere il viso più vicino altre più lontano).

```
R = 500; si = linspace(0,1,R);
yi=interp1(st,Y,si,'spline');
xi=interp1(st,X,si,'spline');

v = diff(xi);
u = diff(yi);

phi = unwrap([0 angle(v + 1i*u)]);
```

Queste linee di codice riassumono tutto il procedimento di passaggio dalla coordinata cartesiana a quella polare. Per prima cosa abbiamo scelto di operare con i valori delle x e delle y interpolati (si noti il campionamento molto più grande, nel nostro caso 500); la formula trigonometrica è $\arctan\left(\frac{y}{x}\right)$, tuttavia essendo le nostre x e y dei vettori abbiamo prima fatto la loro differenza (ovvero rispettivamente x_k - x_{k-1} e y_k – y_{k-1}) e in seguito l'arcotangente.

Si noti che l'angolo di rotazione `phi` è stato ottenuto con un procedimento leggermente diverso per un migliore risultato: `angle` è una funzione che restituisce gli angoli di fase, in radianti, per ciascun elemento del vettore passatogli sottoforma di numero complesso in cui v, rappresenta la parte reale, ed u, la parte immaginaria. Per completezza `unwrap` è una funzione che corregge gli angoli di fase aggiungendo multipli di $\pm 2\pi$ nei salti che si possono creare tra elementi consecutivi.

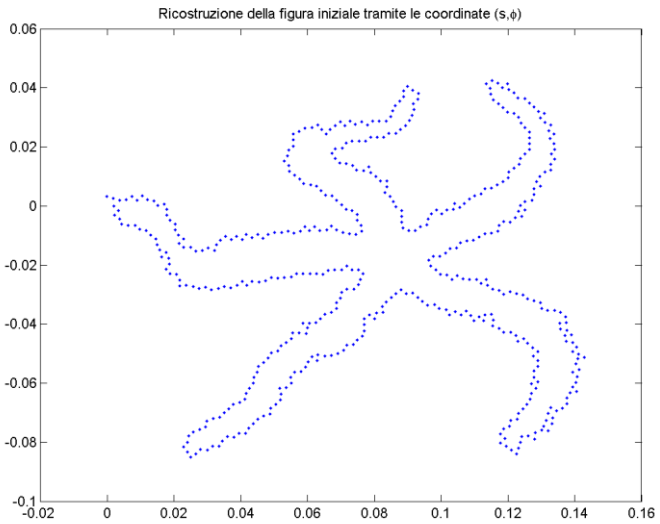


FIGURA 5 - Ricostruzione della figura iniziale

La parte finale del progetto è stata ricostruire tramite le formule inverse l'immagine originaria.

```
del = 1/R;
xc=zeros(1,R); yc=zeros(1,R);

xc = cumsum (del*cos(phi));
yc = cumsum (del*sin(phi));
```

Qui abbiamo semplicemente calcolato l'ampiezza degli intervalli tra i valori della coordinata curvilinea (tutti uguali), creato due nuovi vettori delle ascisse e delle ordinate in cui abbiamo poi inserito i valori grazie alle formule del seno e del coseno. (Figura 5)

STATMOD E L'INTERPOLAZIONE STATISTICA

Una alternativa alla ricostruzione con `interp1` è quella realizzata dal prof. Fiori: `statmod`. `Statmod` basa il suo funzionamento sul modello della regressione statistica implementato grazie ad una rete neurale; in altre parole anziché valutare le variabili come accoppiate al fine di cercare un modello che le possa descrivere, la funzione valuta la densità probabilistica delle ascisse e delle ordinate separatamente.

Abbiamo voluto mostrare pregi e, forse, qualche limitazione di questo metodo in `'programma_statmod.m'` in cui tutte le interpolazioni vengono fatte con `statmod`.

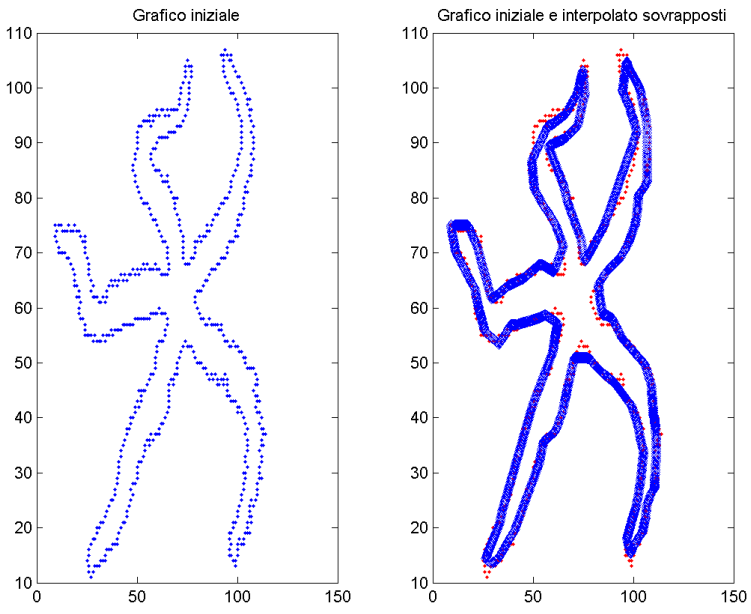


FIGURA 6 - A sinistra la figura iniziale, a destra il grafico interpolato con 'statmod'

Ciò che subito salta all'occhio è la difficoltà nella ricostruzione della figura nel caso della presenza di buchi e ciò, essendo `statmod` basata sull'analisi tramite la distribuzione statistica, è da ricercare nel fatto che in quel punto essa è più scarsa rispetto alla restante parte della figura, provocando un diradamento dei punti anzichè una ricostruzione come per `interp1`.

Un altro limite è che mentre nell'interpolazione delle ascisse e delle ordinate singolarmente i risultati sono apprezzabili, non si può dire la stessa cosa nel caso dell'angolo di rotazione.

Un ultimo aspetto che vorrei sottolineare riguarda invece il miglioramento dell'immagine, a mio parere più riuscito che con `interp1`.

ANALISI COMPUTAZIONE DEI COMANDI RICORSIVI

In quest'ultima sezione mi piacerebbe dare un cenno di analisi computazionale relativamente ai comandi ricorsivi. Di solito in un primo momento (sarà forse l'influenza di C++) si è tentati di programmare Matlab usando i cicli `for`; in realtà bisogna ricordare che l'applicativo è stato sviluppato per poter lavorare nella massima efficienza con vettori e matrici.

In generale quindi, usare un ciclo `for` è sempre meno efficiente che operare direttamente (qualora sia possibile) tra matrici e vettori. I due programmi `'programma_eff.m'` e `'programma_noneff.m'` sono copie del progetto, senza le scelte dell'utente ma con caricamenti e operazioni già impostate, che adottano il primo le operazioni con i vettori, il secondo i cicli `for`.

I risultati, seppur non clamorosi, dimostrano quanto detto: il programma con i cicli impiega per la sua esecuzione `6.8850 s` mentre quello efficiente `6.5570 s`. Ho verificato i dati inserendo in testa al programma il comando `t0=clock;` e in coda `time=etime(clock, t0);` che restituisce il tempo trascorso.

Tuttavia per piccole elaborazioni come le nostre la differenza non è molta (il `for` per il vettore caricato in questo programma cicla solo 641 volte, non molte per notare un grande risparmio di tempo), infatti usando i comandi `tic` e `toc` agli estremi dell'ultimo `for` e nella corrispondente operazione con i vettori ho ottenuto `0.000146 s` e `0.000135 s` per il programma efficiente.

CONCLUSIONI

Realizzare questo progetto come completamento dell'esame di elettrotecnica è stato un fattore di crescita notevole: Matlab è un programma complesso e molto grande per essere compreso con un solo mese di lavoro ma sicuramente è stato interessante poterne assaporare alcuni tratti relativi all'elaborazione dei dati.

Realizzare un progetto è decisamente fonte di sviluppo del pensiero e spero, seppur in minima parte, di aver elaborato alcune considerazioni abbastanza interessanti (nonostante sia alle prime armi) mosso dalla curiosità di capire il funzionamento di alcune operazioni.

Un ultimo aspetto molto interessante è stata l'analisi computazionale di alcuni algoritmi, già accennata in parte nel corso di Fondamenti di Informatica e che tra breve riprenderemo in Algoritmi e Strutture dati; l'auspicio è quello di aver dato una chiara differenza tra i due metodi seppur i dati non siano di grande aiuto dato il piccolo scarto tra un programma e l'altro, come dicevo, forse dettato dalle poche iterazioni da eseguire.

Una stella marina o un cavalluccio, a vostra scelta: poi Matlab pensa a fare i conti!

Nicola Fioranelli
Ancona, 04 marzo 2015

```
% Progetto MATLAB per il corso di Elettrotecnica (a.a. 2014/2015)
% Studente partecipante: Nicola Fioranelli
% Creato il 04.02.2015
% Presso Dipartimento di Ingegneria dell'Informazione
% Università Politecnica delle Marche
% -------------------------------------------------------------
% Cancello tutte le variabili, chiudo tutte le figure esistenti e cancello
% lo schermo
clear all; close all; clc

% Carico il file .mat con il comando 'load'
K = menu('Scegli un contorno','Stella marina','Cavalluccio marino');
switch K
    case 1, load StarFish-data.mat;
    case 2, load SeaHorse-data.mat;
end

% Dato che esso viene inizializzato da Matlab come matrice
% divido le colonne delle matrice in due vettori in riga
% X e Y sono due matrici trasposte dell'originale x
X = x(:,1)';
Y = x(:,2)';

% Utilizzo il comando length per calcolare il numero 'n' di campioni
n = length(X);

% Utilizzo il comando 'plot' che mi consente di graficare
% il database iniziale.
figure(1); subplot(1,2,1); plot (X,Y, 'b.')
           xlabel('Valore del campione x_k')
           ylabel('Valore del campione y_k')
           subplot(1,2,2); plot (1:n,Y,'m',1:n,X,'g','LineWidth',2)
           xlim ([1 n]);
           legend('Grafico delle ordinate (y_k)','Grafico delle ascisse
(x_k)','Location','Best')
           xlabel('Numero del campione (k)')
           ylabel('Valore del campione (x_k o y_k)')

% I due grafici successivi sono le distribuzioni statistiche dei valori della x e della
y.
figure(2); subplot(1,2,1); hist (X,25);title('Distribuzione statistica dei campioni x_k')
           ylabel('Numero di occorrenze'); xlabel('Valori dei campioni x_k')
           subplot(1,2,2); hist (Y,25);title('Distribuzione statistica dei campioni y_k')
           ylabel('Numero di occorrenze'); xlabel('Valori dei campioni y_k')

% Resampling con "decimazione"
px2 = X(1:4:n); py2 = Y(1:4:n);
figure(3); plot(X,Y,'b.',px2,py2,'r.'); title ('Interpolazione con "decimazione"')

% Creo il vettore per la coordinata curvilinea di lunghezza n.
st=zeros(1,n);

% Uso le operazioni tra i vettori per ottenere la coordinata curvilinea.
st=[0 cumsum(sqrt(diff(X).^2 + diff(Y).^2))];
```

```
% Normalizzazione secondo l'ultimo valore.
st=st/st(n);

% Creo il vettore dei punti semplici per l'interpolazione.
R = 2000; si = linspace(0,1,R);

% Interpolo la x e la y con il metodo spline.
yi=interp1(st,Y,si,'spline');
xi=interp1(st,X,si,'spline');

% Nella figura si può notare il maggior numero di punti presenti che ho
% definito nel vettore dei punti semplici con passo pari a 0.1.
figure(4); subplot(2,2,1); plot(X,Y,'.');title('Grafico iniziale')
           subplot(2,2,2); plot(xi,yi,'.');title('Grafico interpolato con il metodo
"spline"')
           subplot(2,2,[3 4]); plot(X,Y,'r.',xi,yi,'bd');title('Grafico iniziale e
interpolato sovrapposti')

% La distribuzione statistica con 25 bin da un grafico molto più
% equalizzato in quanto ho aggiunto punti laddove mancavano.
figure(5); hist(st,25); title('Distribuzione statistica dei valori s_k')
ylabel('Numero di occorrenze'); xlabel('Valori dei campioni s_k')

% Ho fatto l'interpolazione anche con il metodo 'linear' per notare la
% differenza.
yl=interp1(st,Y,si,'linear');
xl=interp1(st,X,si,'linear');

figure(6); subplot(2,2,1); plot(X,Y,'.');title('Grafico iniziale')
           subplot(2,2,2); plot(xl,yl,'.');title('Grafico interpolato con il metodo
"linear"')
           subplot(2,2,[3 4]); plot(X,Y,'r.',xl,yl,'bd');title('Grafico iniziale e
interpolato sovrapposti')

% In figura 7 ho creato uno zoom sulle figure interpolate per notare il
% diverso metodo di interazione.
% La lineare crea una sequenza di punti più lineare appunto mentre la
% spline crea una serie di punti 'a curva'.
figure(7); subplot(1,2,1); plot(xi,yi,'.'); xlim([40 50]);title('Zoom
dell''interpolazione "spline"')
           subplot(1,2,2); plot(xl,yl,'.'); xlim([40 50]);title('Zoom
dell''interpolazione "linear"')


% ------------------------------------------------------------------------
% Esempio di ricostruzione attraverso l'interpolazione

prompt = {['Inserisci il valore minimo (<' num2str(n) ') da cui iniziare a cancellare:'],
          ['Inserisci il valore massimo (<' num2str(n) ') fino a cui cancellare:']};
dlg_title = 'Simulazione dati mancanti'; num_lines = 1; def = {'30','50'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
emin=str2double(answer(1)); emax=str2double(answer(2));

xe = X;
ye = Y;
```

```
xe(emin:emax)=[];
ye(emin:emax)=[];

cc=[0 cumsum(sqrt(diff(xe).^2 + diff(ye).^2))];
cc=cc/cc(end);

R = 2000; cci=linspace(0,1,R);
yi=interp1(cc,ye,cci,'spline');
xi=interp1(cc,xe,cci,'spline');

figure(8); plot(xe,ye,'.r',xi,yi,':b'); title('Ricostruzione di parte della curva
eliminata')

% ------------------------------------------------------------------------
% Resampling e calcolo dell'angolo di rotazione

% Resampling della x e della y con il metodo spline per calcolare l'angolo di
% rotazione.
R = 500; si = linspace(0,1,R);
yi=interp1(st,Y,si,'spline');
xi=interp1(st,X,si,'spline');

% Creo il vettore v (differenza delle ascisse) con i dati già interpolati.
v = diff(xi);

% Creo il vettore u (differenza delle ordinate) con i dati già interpolati.
u = diff(yi);

% Calcolo l'angolo di rotazione grazie ad angle passandogli i dati
% sottoforma di numeri complessi e miglioro l'immagine con unwrap.
phi = unwrap([0 angle(v + 1i*u)]);

figure(9); subplot(2,1,1);plot(si,phi);xlabel('Coordinata s');ylabel('Angolo di rotazione
\phi')
           subplot(2,1,2);hist(phi,25);xlabel('Angolo di rotazione \phi');ylabel('Numero
di occorrenze')
                                       title('Distribuzione statistica dei valori
\phi_k')

% ------------------------------------------------------------------------

% Calcolo l'ampiezza dell'intervallo tra i valori di s in [0, 1]
del = 1/R; % Gli intervalli sono tutti uguali.

xc=zeros(1,R); yc=zeros(1,R);

% Utilizzando le formule inverse ricostruisco l'immagine a partire
% dalla coordinata curvilinea.
xc = cumsum (del*cos(phi));
yc = cumsum (del*sin(phi));

figure(10);plot(xc,yc,'.');title('Ricostruzione della figura iniziale tramite le
coordinate (s,\phi)')
```