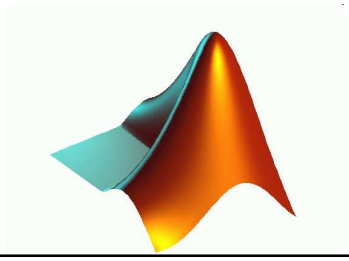
INTRODUZIONE A MATLAB

E' un sistema interattivo destinato alla gestione di matrici.

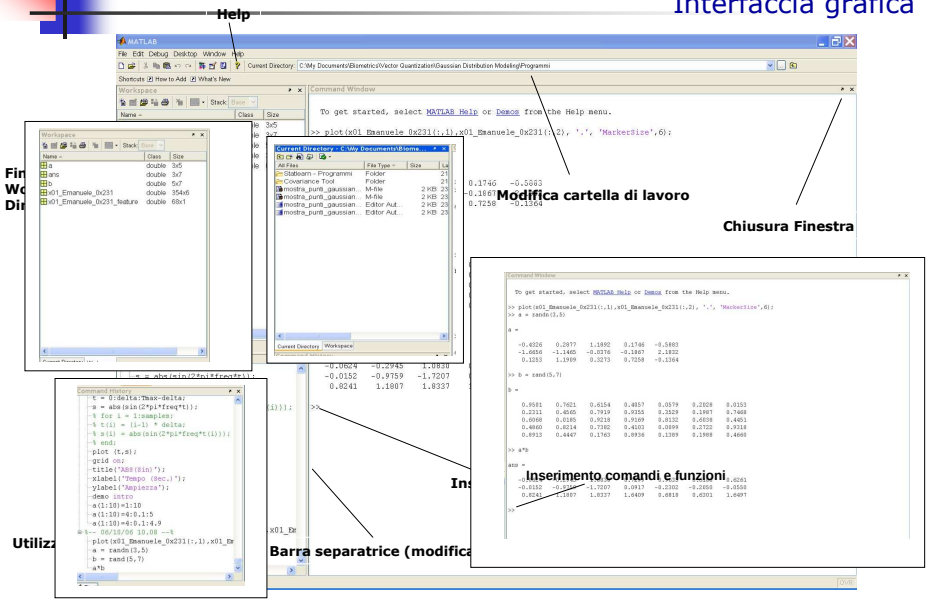
L'unità di base è una matrice.

Matlab è un linguaggio di programmazione ad alte prestazioni per la gestione di calcoli, simulazioni e visualizzazioni.



Schermata Matlab

Interfaccia grafica



La linea di comando

- Quando si lancia il programma Matlab compare un prompt come in DOS : >> che rappresenta la linea di comando di Matlab.

- Accetta dichiarazioni di variabili, espressioni e chiamate a tutte le funzioni disponibili nel programma. Tutte le funzioni di MATLAB non sono altro che files di testo, simili a quelli che l'utente può generare con un text editor, e vengono eseguite semplicemente digitandone il nome sulla linea di comando. MATLAB permette inoltre di richiamare le ultime righe di comandi inseriti usando le frecce in alto e in basso.

Assegnazione di variabili costanti

```
>> a = 1.54
```

→ a è il nome della costante, 1.54 è il valore.

```
>> a = 1.54;
```

→ ";" non visualizza la risposta sullo schermo

```
>> 5
ans = 5
```

→ "ans" è il nome della variabile di default

Di default Matlab lavora in doppia precisione. Ogni numero memorizzato in doppia precisione occupa 8 bites.

Operazioni aritmetiche

+	addizione
–	sottrazione
/	divisione
*	moltiplicazione
^	potenza

$$x = \frac{3+5^3-2/3}{4(5+2^4)}$$

ATTENZIONE: l'intero calcolo va scritto in riga. E' necessario un uso adeguato delle parentesi () per le precedenze aritmetiche.

```
>> x = (3 + 5^3 - 2/3)/(4*(5 + 2^4))
x = 1.5159
```

Variabili predefinite

pi	π
i , j	unità immaginaria
NaN	Not a Number (Inf *0)
eps	2.2204e-16 precisione di macchina

Comandi di uso generale

- who: elenco delle variabili definite in memoria;
- whos: informazioni su tutte le variabili in memoria;
- clear: cancella tutte le variabili in memoria o una in particolare se specificata (clear nome_variabile);
- clc: Ripulisco la Command Window;
- save: salva tutte le variabili in memoria sul file specificato, in vari formati;
- load: richiama in memoria le variabili salvate sul file specificato;
- quit: Per uscire dall'Ambiente;
- help: Matlab presenta un help in linea con le informazioni riguardanti la sintassi di tutte le funzioni disponibili e di tutte le istruzioni.
  Per accedere a queste informazioni basta digitare al prompt:
  >>help nome_funzione (o <nome_istruzione>)
  E vengono visualizzate tutte le informazioni riguardanti la funzione o istruzione richiesta.

Format

visualizzazione dei numeri sul display:

```
>> format type
```

valori di type	risultato	pi
short	Virgola fissa scalata con 5 cifre	`3.1416`
short e	Forma exp con 5 cifre di mantissa	`3.1416e+000`
short g	Rappresentazione migliore con 5 cifre	`3.1416`
long	Virgola fissa scalata con 15 cifre	`3.14159265358979`
long e	Forma exp con 15 cifre di mantissa	`3.141592653589793e+000`
long g	Rappresentazione migliore con 15 cifre	`3.14159265358979`

Vettori e Matrici

- Matlab è orientato alla gestione di matrici; infatti in Matlab ogni variabile è una matrice. Gli scalari non sono altro che matrici 1x1. che appresenta la linea di comando di Matlab.

- L'inserimento di un vettore o matrice in generale avviene utilizzando le parantesi quadre, inserendo gli elementi per righe e separando gli elementi di una stessa riga con le virgole o spazi e le diverse righe con punti e virgola.

Assegnazione di matrici e arrays

Modi equivalenti di generare un vettore riga:

```
v = [1 5 8 12]
v = [1,5,8,12]

v = [1:8]
```

Generazione di un vettore colonna:

```
v = [1;5;8;12]
v = [1 5 8 12]'
```

Generazione di una matrice di dimensione (2 x 3):

```
>> m = [1 6 2; 3 9 1]

m =

     1     6     2
     3     9     1
```

Operazioni sulle matrici

Accedere agli elementi: `m(i,j)` (`v(i)` per accedere al vettore)

Estrarre una riga della matrice: riga=m(#,:)
es: `riga=m(2,:)` ***estrae tutta la riga 2 della matrice m***

Estrarre colonne della matrice: col=m(:,#)
es: `col=m(:,1)` ***estrae tutta la colonna 1 della matrice m***

La sottomatrice avente elementi a_{mn} con $m_1 \le m \le m_2$ e $n_1 \le n \le n_2$, è indirizzata come $A(m_1:m_2, n_1:n_2)$;
es: ***A(1:2,2:3) dà [2,3;5,6]***

Operazioni sulle matrici

Gli elementi di vettori e matrici possono essere espressioni MATLAB. Per esempio inserendo il vettore

```
>> x=[ -1.31 sqrt(3) (1+2+3)*4/5 ]
```

Matlab memorizzerà il vettore

x=-1.3100 1.7321 4.8000

Scrivendo

```
>> x(5)=abs(x(1))
```

Si ottiene

x =-1.3100 1.7321 4.8000 0 1.3100

Ovvero la dimensione di x è automaticamente aumentata per inserire il nuovo componente.

Gli intervalli (1/2)

Il Matlab permette di definire intervalli numerici in modo semplice ed automatico. Esistono per tale scopo specifici operatori e funzioni.

L'operatore "**:**"
Consente la generazione di intervalli equi-spaziati
Sintassi: valore iniziale: incremento: valore finale
N.B. l'incremento di default è pari a 1

```
X=1:5 => X=(1 2 3 4 5)
X=0:2:10 => X=(0 2 4 6 8 10)
X=0:3:10 => X=(0 3 6 9)
X=0:1.5:9 => x=(0 1.5 3.0 4.5 6.0 7.5 9.0)
X=0:-1:-5 => X=(0 –1 –2 –3 –4 -5)
```

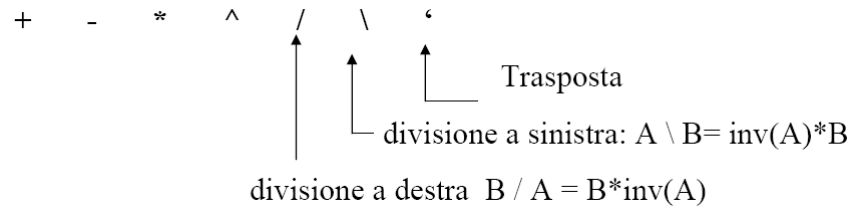
Gli intervalli (2/2)

L'operatore "linspace"
La funzione linspace crea un intervallo numerico prefissando il numero di punti piuttosto che l'incremento.
Sintassi: linspace(valoreInziale, valoreFinale, numeroPunti)
N.B: Il numero di punti di default è 100

Esempio:

```
>> s = linspace(1,10,6)

s =

    1.0000    2.8000    4.6000    6.4000    8.2000   10.0000
```

Operatori applicabili a matrici



Ricorda: L'ordine dei fattori è importante quando si fanno operazioni sulle matrici: **A*B ≠ B*A**

Altre funzioni operanti su matrici:

inv (inversa di una matrice)

det (determinante di una matrice)

size (dimensioni di una matrice)

rank (rango di una matrice)

Operatori applicabili a matrici

Esistono operatori particolari che permettono di effettuare operazioni su vettori ***elemento per elemento*** senza ricorrere ai cicli

(**.***, **./**, **.^**)

Se x e y sono due vettori per moltiplicare elemento per elemento i due vettori basta fare: z = x .* y;

ES: x=[1 2 3] y=[4 5 6]

>> z=x.*y darà come risultato z=[4 10 18]
>> z=x./y darà come risultato z=[0.25 0.4 0.5]
>> z=x.\y darà come risultato z=[4.0 2.5 2.0]

Operatori applicabili a matrici

eig (autovalori di una matrice) opera su matrici quadrate nel modo seguente:

y=eig(A)
produce un vettore y contenente gli autovalori della matrice A.

[U,D] = eig(A)
produce una matrice U avente per colonne gli autovettori della matrice A e una matrice diagonale D avente sulla diagonale gli autovalori di A.

Altre funzioni operanti su vettori (riga o colonna):

max, min, median, sort, sum, prod, length

Funzioni utili lavorando con le matrici

- **`eye(n)`** matrice identità n x n
- **`zeros(m,n)`** matrice di 0 m x n
- **`ones(m,n)`** matrice di 1 m x n
- **`rand(m,n)`** matrice casuale di valori tra 0 e 1 mxn
- **`diag(X)`** se X è un vettore con n elementi, crea una matrice quadrata diagonale di dimensione n x n con gli elementi di X sulla diagonale. Se X è una matrice quadrata n x n produce un vettore di n elementi pari a quelli sulla diagonale di X

Unire 2 matrici:
dopo aver creato 2 vettori o matrici I e Y

```
A=[I,Y]
A=[I;Y]
```

Funzioni matematiche intrinseche

`sqrt(x)`	radice quadrata
`round(x)`	arrotondamento
`fix(x)`	parte intera del numero
`sign(x)`	segno del numero (vale 1, 0 o -1)

`cos(x), sin(x), tan(x)`
`cosh(x), sinh(x), tanh(x)`
`acos(x), asin(x), atan(x)`
`exp(x), log(x), log10(x)`

Per z complesso:

real(z)	parte reale
imag(z)	parte complessa
conj(z)	complesso coniugato

Stringhe

In matlab una stringa è un vettore di caratteri:

- **`s = 'oste';`**
- **`s(1) => o`**
- **`s(1)='a'; => s='aste'`**
- **`s=['c',s] => s='caste'`**

Operazioni sulle stringhe:

`str2mat` : trasforma un sequenza di stringhe in una matrice quadrata
`int2str` : trasforma un intero in una stringa
`str2num` : trasforma una stringa in un numero

→**`disp('testo')`**

Esercizio

Creare una matrice in cui la prima riga sia composta dai numeri da 1 a 10, la seconda riga composta dai numeri da 11 a 20 e la e la terza dai numeri da 21 a 30. Modificare la seconda riga in modo da annullarne gli elementi.

```
>> m=[1:10;11:20;21:30]

m =

     1     2     3     4     5     6     7     8     9    10
    11    12    13    14    15    16    17    18    19    20
    21    22    23    24    25    26    27    28    29    30

>> m(2,:)=0

m =

     1     2     3     4     5     6     7     8     9    10
     0     0     0     0     0     0     0     0     0     0
    21    22    23    24    25    26    27    28    29    30
```

Finestre grafiche

MATLAB ha anche la possibilità di lavorare con delle finestre grafiche sulle quali si possono fare disegni bidimensionali o tridimensionali. Una finestra grafica viene aperta con il comando **figure** (in ambiente Windows anche con File-New-Figure). Si esegua:

```
>> figure
>> figure(n)
```

Il comando `plot`

L' istruzione per fare disegni bidimensionali è

`plot(x,y)`

dove x = (x1, ..., xn), y = (y1, ..., yn) sono vettori della stessa dimensione n
L'effetto è che sulla finestra grafica corrente (se non ci sono finestre grafiche ne viene aperta una) viene introdotto un sistema di coordinate cartesiane bidimensionale e si congiunge (x1, y1) con (x2, y2), (x2, y2) con (x3, y3), ..., (xn-1, yn-1) con (xn, yn) con una linea continua di colore blu.

Si provi

```
plot([0 2 3 4 6],[0 1 2 4 3])
```

Modalità di rappresentazione

Le modalità di rappresentazione dei punti possono essere cambiate dando un terzo argomento *stringa s* a plot che individua la modalità di rappresentazione.
L'istruzione plot nella sua forma completa è

`plot(x,y,s)`

Per vedere tutte le possibili modalità di rappresentazione si veda l'help in linea su plot.
Esempio:

```
>>plot([0 2 3 4 6],[0 1 2 4 3],'--')
>>plot([0 2 3 4 6],[0 1 2 4 3],'r+')
>>plot([0 2 3 4 6],[0 1 2 4 3],'ko')
```

Comandi utili

– grid: sovrappone al grafico un grigliato
– title: aggiunge un titolo del disegno
– xlabel: aggiunge una legenda per l'asse x
– ylabel: aggiunge una legenda per l'asse y
– axis: riscala gli assi del grafico
– clf : cancella il grafico corrente

```
>> xlabel('string')
>> title('string')
>> axis([xmin xmax ymin ymax]), axis square
```

Plot di funzioni reali

L'istruzione plot può fare più disegni sulla finestra grafica. Basta dare più coppie (x, y) (o terne(x, y, s)) come argomento a plot.

Ad esempio se si vuole tracciare in [0, 2] il grafico del seno e del coseno si possono usare le istruzioni

```
>>h=0.01;
>>x=0:h:2*pi;
>>y=sin(x);
>>z=cos(x);
>>plot(x,y,x,z)
```

Si noti come, automaticamente, MATLAB disegni i due grafici con colori diversi.

Gestione della finestra grafica

L'istruzione plot, prima di disegnare sulla finestra grafica corrente, cancella ogni disegno preesistente. Per evitare questo si usa il comando **hold on**. Dopo la sua esecuzione, sulla finestra grafica corrente viene mantenuto ogni disegno preesistente.
Volendo tracciare in [0, 2] il grafico del seno e del coseno si possono usare allora anche le istruzioni:

```
>>h=0.01;
>>x=0:h:2*pi;
>>y=sin(x);
>>plot(x,y);
>>hold on
>>z=cos(x);
>>plot(x,z,'r')
```

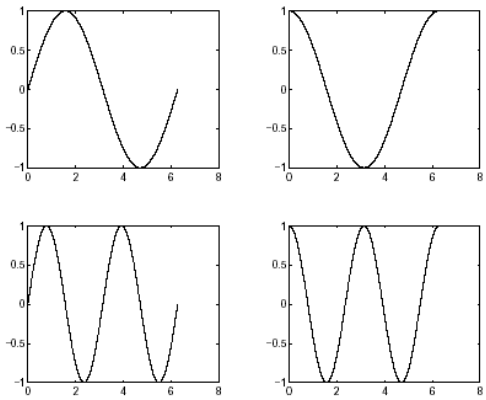
Per tornare alla situazione in cui vengono cancellati i disegni preesistenti si usa hold off.

Gestione della finestra grafica

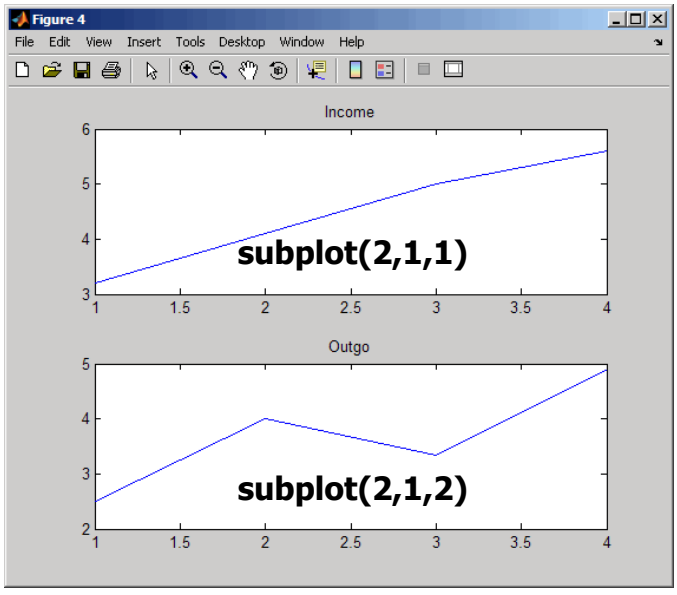
E' possibile spezzare la finestra grafica corrente in una matrice [mxn] di sottofinestre grafiche. Il comando subplot(m,n,k) divide la finestra corrente (ne crea una se non esistono finestre grafiche) in una matrice [mxn] e fa diventare la k-esima, contata seguendo le righe, la finestra corrente.

```
>>x=0:h:2*pi;
>>subplot(2,2,1)
>>plot(x,sin(x))
>>subplot(2,2,2)
>>plot(x,cos(x))
>>subplot(2,2,3)
>>plot(x,sin(2*x))
>>subplot(2,2,4)
>>plot(x,cos(2*x))
```

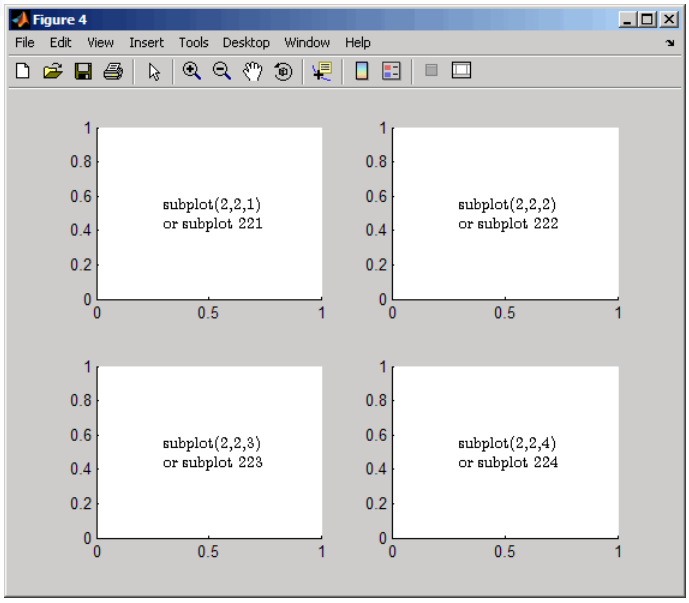
Gestione della finestra grafica



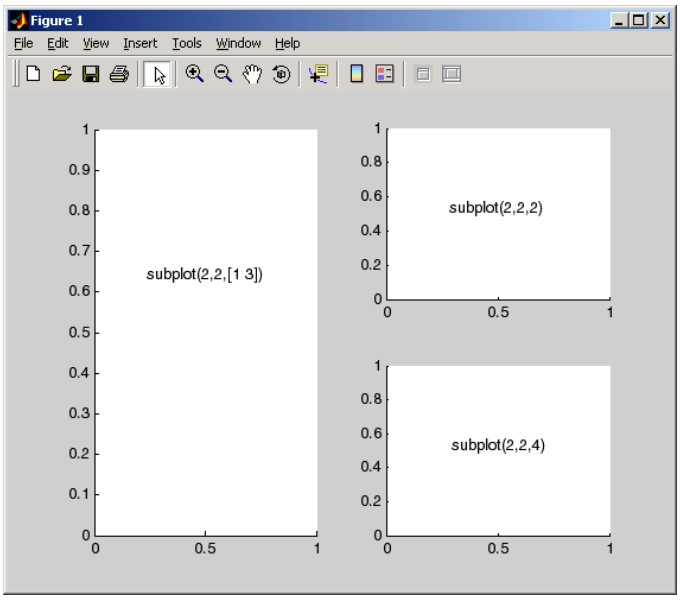
Gestione della finestra grafica



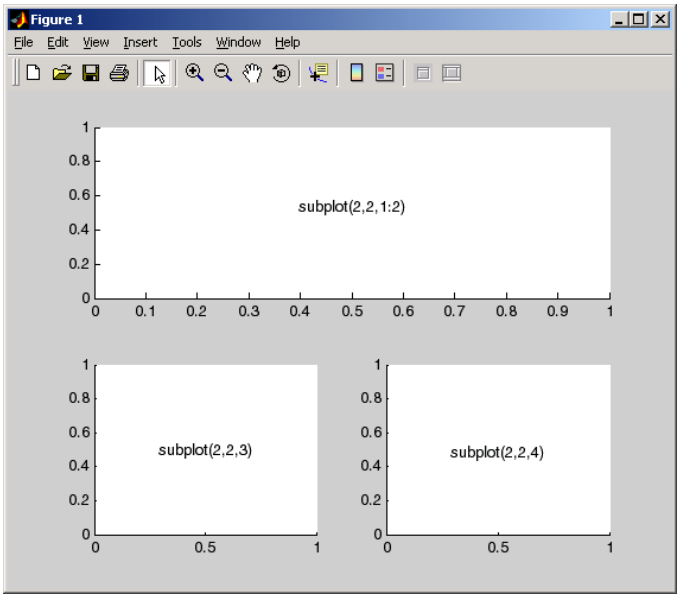
Gestione della finestra grafica



Gestione della finestra grafica



Gestione della finestra grafica



La programmazione

In Matlab si possono realizzare degli M-file, ovvero file di testo contenenti sequenze di comando e strutture di controllo che vengono interpretate dal Matlab.

I file, prodotti mediante un editor di testo devono essere salvati in un file con estensione .m, in una directory contenuta nel path.

Macro

Gli M-file di tipo macro operano sulle variabili contenute in memoria, e non esistono variabili locali. Contengono una serie di comandi che vengono automaticamente eseguiti quando si esegue la macro.
Per eseguire un M-file basta digitarne il nome (senza l'estensione) dalla riga di comando.

Per creare una macro:

1. aprire un file nuovo (emacs oppure File → New)
2. scrivere il codice della macro
3. salvare il file **`filename.m`**
4. eseguire da linea di comando >> filename + **`INVIO`**

Micro esempio di macro...

```
% ESEMPIO DI MACRO: calcola la matrice trasposta di
% una matrice A e ne visualizza l'output a schermo
(usare anche il comando disp).


Atrasp = A';
disp('la trasposta della matrice è')
Atrasp
```

Function

Matlab permette di definire funzioni utente.
Le funzioni vanno scritte in modo identico agli M-file, tranne pe
l'intestazione che è del tipo:

```
function [variabili uscita]=nomefunzione(variabili ingresso
```

```
funzione [out1, out2, ...] = nomefunzione (IN1, IN2, ...)
```

N.B.: Le funzioni vanno salvate in un file avente lo stesso nome della funzione.
Tutte le variabili sono locali alla funzione, per cui dopo la sua esecuzione non restano in memoria.

La function viene chiamata da linea di comando con:
nomefunction(valore_variabile_ingresso)

Esempio di function

```
% ESEMPIO DI FUNCTION: trasformare la macro di prima in
% una function.
function [Atrasp] = trasposta(A)
Atrasp = A';
```

oppure

```
function trasposta(A)
Atrasp = A';
disp('la trasposta è')
Atrasp
```

```
Atrasp = A';
disp('la trasposta della matrice è')
Atrasp
```

Istruzione `if, else, elseif`

`if` valuta un'espressione logica ed esegue una serie di istruzioni a seconda del valore dell'espressione logica.

```
if espressione logica
   istruzioni
elseif espressione logica
   istruzioni
else
   istruzioni
end
```

Operatori di relazione:

>	maggiore
<	minore
>=	maggiore o uguale
<=	minore o uguale
==	uguale
~=	diverso

Esempio istruzione if (1)

Aprire un file nuovo, salvarlo con nome dispar.m
La function prende in input un numero e controlla se esso è pari o dispari.

```
function dispar(x)
if rem(x,2) == 0
    disp('Il numero è pari')
else
    disp ('Il numero è dispari')
end
```

Esempio istruzione if (2)

```
% function per calcolare l'inversa di una matrice 2x2

function [B] = inversa(A)

if (A(1,1)*A(2,2))-(A(1,2)*A(2,1)) == 0
    disp('la matrice non è invertibile')
else
    detA = (A(1,1)*A(2,2))-(A(1,2)*A(2,1));
end

B = 1/detA*[A(2,2), -A(1,2);A(1,1), - A(2,1)];
```

Ciclo for

Il ciclo **for** esegue un numero di istruzioni per un numero fissato di volte.

```
for indice = inizio:incremento:fine
   istruzioni
end
```

Esempio: somma degli elementi di un vettore

```
somma = 0;
v = [3 6 7 0 3];
for j=1:1:length(v)
     somma = somma + v(j);
end
```

Esempio di ciclo for

```
% Calcolare il valore medio di un vettore.

x = [1 2 3 4 5 6 7];

somma = 0;
for j = 1:1:length(x)
    somma = somma + x(j);
end

media = somma/length(x)
```

Ciclo `while`

Il ciclo **`while`** esegue un numero di istruzioni finché l'espressione di controllo rimane vera.

```
while espressione di controllo
   istruzioni
end
```

Esercizio: dividere un numero per 2 finché il risultato non sia inferiore a 0.005. Contare il numero delle operazioni di divisione effettuate.

```
a = 2390;     % dividendo
b = 2;        % divisore
count = 0;
while (a/b > 0.005)
     c = a/b;
     a = c;
     count = count + 1;
end
count
```

Istruzione switch

switch valuta un'espressione ed esegue un unico caso (**case**) possibile di istruzioni in base al valore di tale espressione.

```
switch espressione_switch
    case espressione_case
      istruzioni,..., istruzioni
    case {espr1, espr2, espr3,...}
      istruzioni,..., istruzioni
...
    otherwise
      istruzioni,..., istruzioni
 end
```

Esempio di istruzione switch

```
% Creare una macro che chiede all'utente di inserire un
% valore di eccentricità e fare visualizzare il tipo
% di conica.
```

Codice per inserire dati da linea di comando:
`nomevariabile = input('testo da visualizzare')`

```
eccentricita = input('Inserire un valore di eccentricità  ')

switch (eccentricita)
    case eccentricita == 1
        disp('La conica è una parabola!')
    case eccentricita == 0
        disp('La conica è una circonferenza!')
    case eccentricita > 1
        disp('La conica è un iperbole!')
    otherwise
        disp('La conica è un ellisse!')
end
```

Interpolazione polinomiale

Dato un vettore x contenente i valori della variabile indipendente e un
vettore y con i rispettivi valori da interpolare la funzione
polyfit
fornisce un polinomio p(x) di grado **n** che interpola le coppie (xi,yi) con il metodo dei minimi quadrati.

$$p(x) = p_1 x^n + p_2 x^{n-1} + \ldots + p_n x + p_{n+1}$$

Sintassi:

```
var_p = polyfit(x, y, n)
```

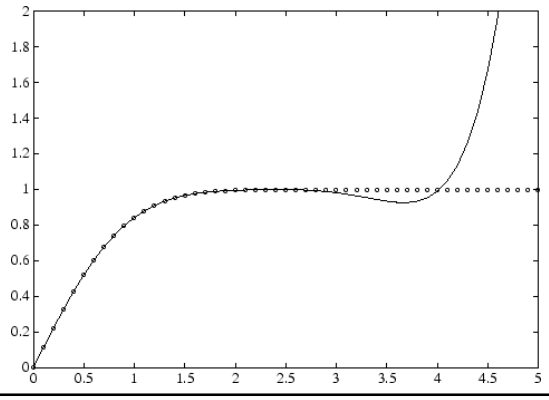
Interpolazione Esempio

```
% erf (x) è due volte l'integrante della
  distribuzione Gaussiana

x = (0: 0.1: 2.5) ';
y = erf(x);
p = polyfit (x, y, 6)

p =

0,0084 -0,0983 0,4217 -0,7435 0,1471 1,1064 0,0004
```

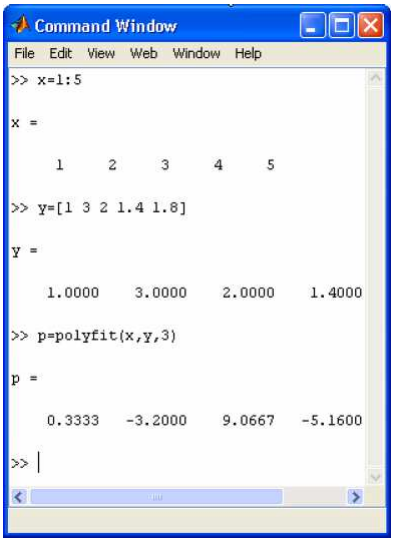
$$0.0084x^6 - 0.0983x^5 + 0.4217x^4 - 0.7435x^3 + 0.1471x^2 + 1.1064x + 0.0004$$

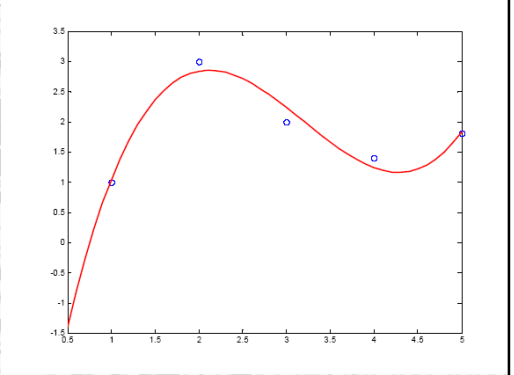
Interpolazione Esempio

```
% Proviamo a plottare le due funzioni

f = polyval (p, x);

plot (x, y, 'O', x, f ,'-')

axsis ([0 5 0 2])
```



Interpolazione Esempio 2





Metodi di interpolazione

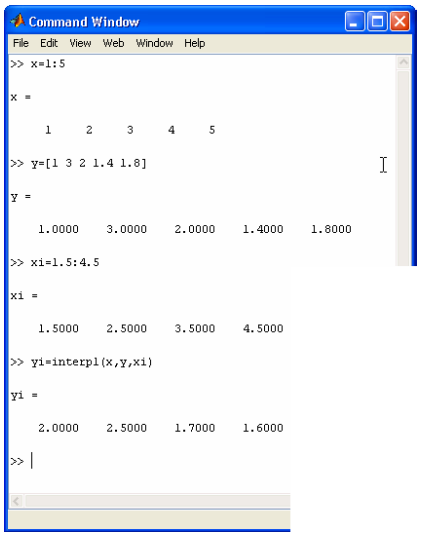
Sono assegnati i vettori x e y contenenti rispettivamente i valori della variabile indipeindente e di quella dipendente di una funzione y=f(x).
Dato un vettore xi, la funzione interp1 fornisce un vettore yi costituito dai valori associati alle ascisse in xi. I valori yi sono ottenuti mediante interpolazione.

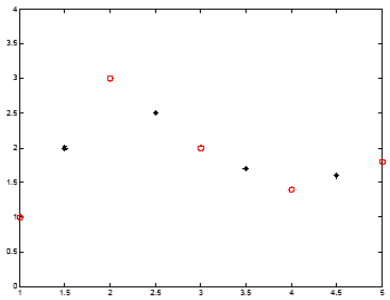
Sintassi:
yi=interp1(x, y, xi, [metodo])

Tipi di metodo: 'linear' (default), 'spline', 'cubic'

Osservazione:
x e xi devono essere monotoni crescenti

Interpolazione Esempio 3

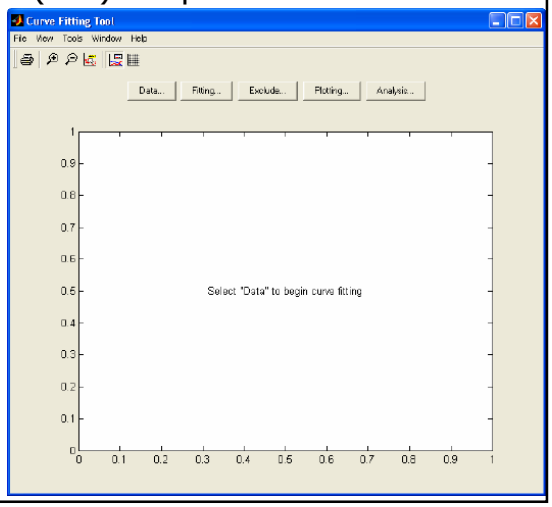




Il Curve Fitting Tool

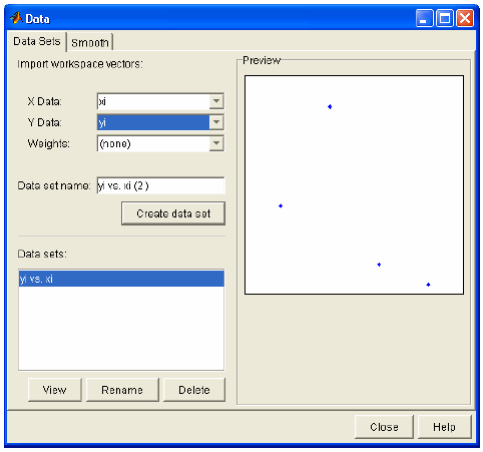
E' una raccolta di funzioni specifiche per l'interpolazione dei dati. Presenta un'interfaccia grafica (GUI) che permette un facile impiego
di tali funzioni.

Si avvia dal Workspace con la funzione cftool.



Il Curve Fitting Tool

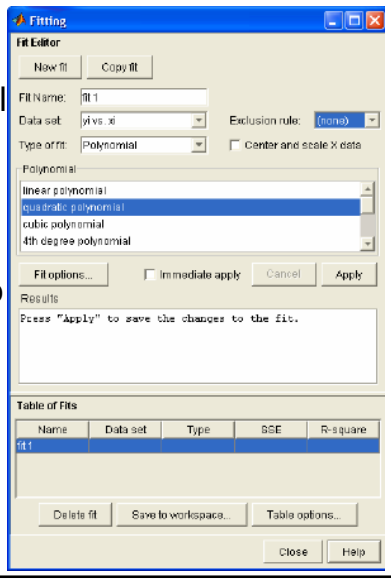
Importazione dati.
Dal tasto Data si accede alla sezione di importazione dati. Questi devono essere già presenti nel Workspace.



Il Curve Fitting Tool

Dal tasto Fitting si accede alla sezione omonima.
Si va sul tasto New Fit Si può definire il tipo di modello da adoperare per il fitting.

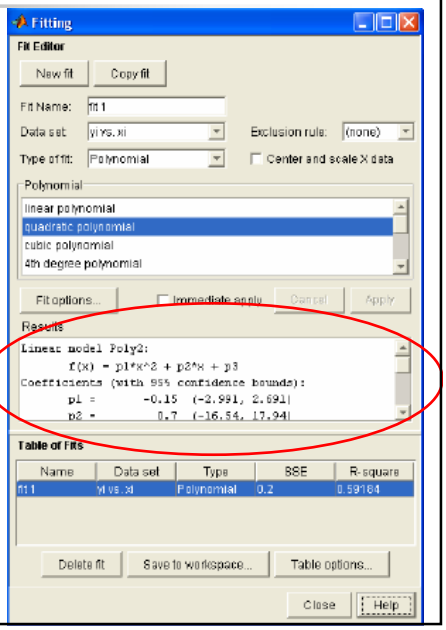
Premendo Apply darà i risultati del tipo di fitting scelto



Il Curve Fitting Tool

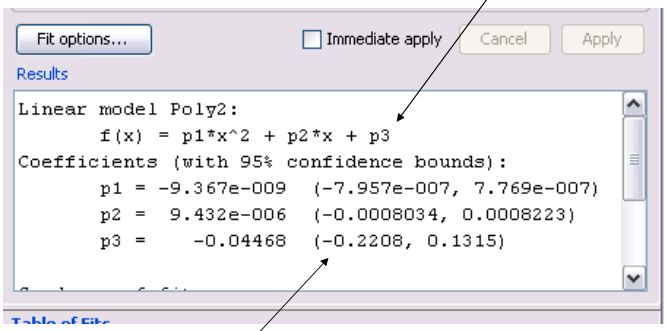
Il tool fornisce i risultati in termini Sia dei coefficienti del modello che in forma grafica.

Parametri del modello



Results: Parametri del modello

Nella finestra di *Results* troviamo nella prima parte l'espressione analitica del modello della polinominale scelta

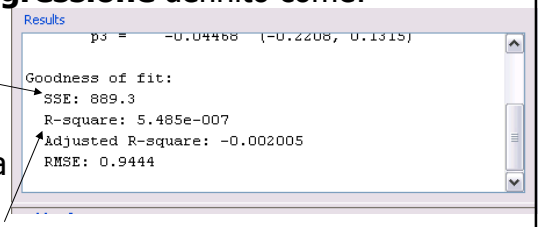


Elencati i valori dei coefficienti della polinominale ed il loro relativo intervallo di fiducia in cui può variare il valore

Results: Parametri del modello

SSE (Sum of Squared Errors) un altro indice della **capacità predittiva della retta di regressione** definito come:

$$SSE = \sum_{i=1}^{N} (\varepsilon_i)^2$$



(***R-square***) il coefficiente di determinazione il rapporto tra la devianza dovuta alla regressione e la devianza totale

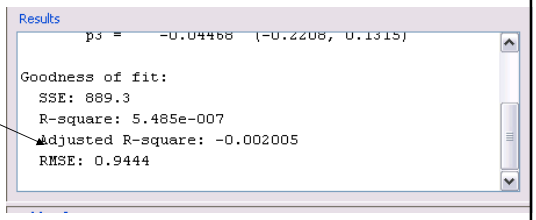
$$R^2 = \frac{\text{Deviazione della regressione}}{\text{Deviazione totale}} = \frac{\sum_{i=1}^{N} (\hat{y} - \bar{y})^2}{\sum_{i=n}^{N} (y - \bar{y})^2}$$

È la proporzione di variazione totale della variabile dipendente ***Y*** che è spiegata dalla variabile indipendente ***X***

Results: Parametri del modello

Il coefficiente di determinazione aggiustato (***adjusted R-square***):

$$R^2_{adj} = R^2 - \frac{p \cdot \left(1 - R^2\right)}{N - p - 1}$$



- ***N*** è il numero di coppie di dati od individui misurati
- ***p*** è il numero di variabili
  (nel caso della regressione lineare semplice **p = 1**).

$$R^2_{adj} < R^2$$

Richiami dell'analisi di Fourier

La serie di Fourier decompone un segnale in una serie infinta di seni e coseni a frequenze diverse.
Assumiamo di avere un segnale di durata 1s, quindi 0<t<1, tale segnale si può rappresentare come una serie infinita:

$$f(t) = a_0 + \sum_{n=1}^{\infty} \left(a_n \sin(2\pi nt) + b_n \cos(2\pi nt)\right)$$

dove f(t) è il segnale nel dominio del tempo mentre a_n e b_n sono i coefficienti incogniti della serie.
L'intero n, ha l'unità di misura degli Hz, e indica la frequenza dell'onda.

Richiami dell'analisi di Fourier

La serie di Fourier:

- è in grado di ricostruire un segnale (a meno di un epsilon).
- è formalmente periodica, ossia l'inizio del ciclo è uguale alla fine (f(t=0)=f(t=1)).

Al fine di determinare i coefficienti della serie si richiamano alcune proprietà:

1 $\int_0^1 \sin(2\pi nt)\sin(2\pi mt)\,dt = 0;$ n e m sono interi con $n \neq m$

2 $\int_0^1 \sin(2\pi nt)\sin(2\pi nt)\,dt = 1/2;$ n è un intero

3 $\int_0^1 \cos(2\pi nt)\cos(2\pi mt)\,dt = 0;$ n e m sono interi con $n \neq m$

4 $\int_0^1 \cos(2\pi nt)\cos(2\pi nt)\,dt = 1/2;$ n è un intero

5 $\int_0^1 \cos(2\pi nt)\sin(2\pi mt)\,dt = 0;$ n e m sono interi

Richiami dell'analisi di Fourier

Prendiamo in considerazione la serie di fourier

$$f(t) = a_0 + \sum_{n=1}^{\infty}\left(a_n \sin(2\pi nt) + b_n \cos(2\pi nt)\right)$$

e moltiplichiamo ambo i membri per $\sin(2\pi mt)$
Si ottiene

$$f(t)\sin(2\pi mt) = \left(a_0 + \sum_{n=1}^{\infty}\left(a_n \sin(2\pi nt) + b_n \cos(2\pi nt)\right)\right)\sin(2\pi mt)$$

Integriamo da 0 a 1

$$\int_0^1 f(t)\sin(2\pi mt)dt = \int_0^1\left(a_0 + \sum_{n=1}^{\infty}\left(a_n \sin(2\pi nt) + b_n \cos(2\pi nt)\right)\right)\sin(2\pi mt)\,dt$$

Richiami dell'analisi di Fourier

$$\int_0^1 \left(a_0 + \sum_{n=1}^{\infty} \left(a_n \sin(2\pi nt) + b_n \cos(2\pi nt) \right) \right) \sin(2\pi mt)\, dt =$$

$$= \int_0^1 a_0 \sin(2\pi mt)\, dt + \int_0^1 \sum_{n=1}^{\infty} a_n \sin(2\pi nt) \sin(2\pi mt)\, dt + \int_0^1 \sum_{n=1}^{\infty} b_n \cos(2\pi nt) \sin(2\pi mt)\, dt$$

0 $\qquad$ $\frac{a_m}{2}$ $\qquad$ 0

Banalmente si ricava che

$$\int_0^1 f(t) \sin(2\pi mt) dt = \frac{a_m}{2}$$

$$\int_0^1 f(t) \cos(2\pi mt) dt = \frac{b_m}{2}$$

$$\int_0^1 f(t) dt = a_0$$

che sono proprio i coefficienti che cercavamo.

Richiami dell'analisi di Fourier (esempio)

Prendiamo ad esempio:

$$f(t) = \sin(2\pi t)$$

Calcoliamo i coefficienti; per cui sostituiamo nelle formule il segnale in esempio

$$\int_0^1 \sin(2\pi t) \sin(2\pi nt) dt = \frac{a_n}{2}$$

Per la proprietà 2 tutti gli a_n sono nulli tranne

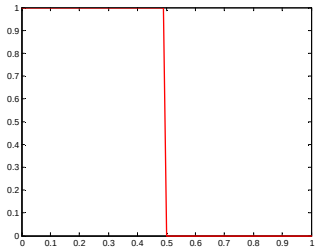
$a_1 = 1$ $\qquad$ 2 $\qquad$ $\int_0^1 \sin(2\pi nt) \sin(2\pi nt)\, dt = 1/2$; $\quad n$ è un intero

Per la proprietà 5 tutti i b_n sono nulli.

$b_n = 1$ $\qquad$ 5 $\qquad$ $\int_0^1 \cos(2\pi nt) \sin(2\pi mt)\, dt = 0$; $\quad n$ e m sono interi

Richiami dell'analisi di Fourier (esempio)

Prendiamo ad esempio un'onda quadra:



$$\begin{cases} f(t)=1 & 0<t<\frac{1}{2} \\ f(t)=0 & \frac{1}{2}<t<1 \end{cases}$$

Calcoliamo i coefficienti a_n:

$$2\int_0^1 f(t)\sin(2\pi nt)dt = a_n \Rightarrow a_n = 2\int_0^{1/2} 1\cdot\sin(2\pi nt)dt = \left[\frac{-2\cos(2\pi nt)}{2\pi n}\right]_0^{1/2} = \frac{1-\cos(\pi nt)}{\pi n}$$

Per n dispari si ottiene:

$$a_n = \frac{2}{\pi n}$$

Per n pari si ottiene:

$$a_n = 0$$

Richiami dell'analisi di Fourier (esempio)

Calcoliamo i coefficienti b_n:

$$2\int_0^1 f(t)\cos(2\pi nt)dt = b_n \Rightarrow b_n = 2\int_0^{1/2} 1\cdot\cos(2\pi nt)dt = \left[\frac{-2\sin(2\pi nt)}{2\pi n}\right]_0^{1/2} = \frac{\sin(\pi nt)}{\pi n}$$

Per qualsiasi n si ottiene:

$$b_n = 0$$

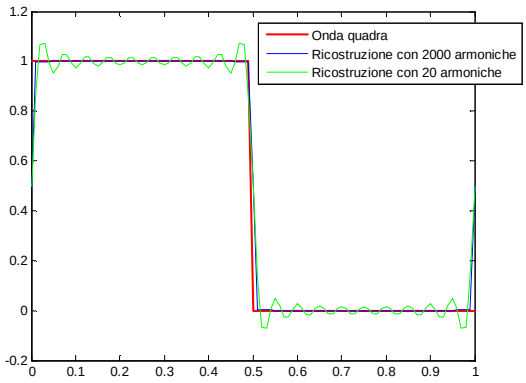
Calcoliamo a_0:

$$\int_0^1 f(t)dt = a_0 \Rightarrow a_0 = \int_0^{1/2} 1dt = \frac{1}{2}$$

Richiami dell'analisi di Fourier (esempio)

Vediamo in Matlab:

```
N = 2000;
x = [0:0.01:1];
f = ones(1,101)*1/2;
for i = 1:2:N
a = 2/(pi*i);
f = f+ a*sin(2*pi*i*x);
end
plot(x,f)
```



Trasformata discreta di Fourier

La trasformata discreta di Fourier (DFT,*Discrete Fourier Tranform*), equivale alla serie di Fourier per segnali a tempo discreto.

Il successo di questo strumento è in buona parte dovuto all'esistenza di un algoritmo, noto come FFT (*Fast Fourier Transform, i.e. trasformata veloce di Fourier*), che permette di calcolare la DFT in maniera veloce.

Per questo motivo la FFT viene utilizzata sostanzialmente in tutti i campi applicativi che fanno uso dell'elaborazione numerica dei segnali.

Trasformata discreta di Fourier

Si consideri un segnale $v(k)$ a tempo discreto e periodico di periodo N, cioè tale che $v(k + N) = v(k)$. È possibile pensare a $v(k)$ come a dei campioni ottenuti da un segnale tempo continuo $v_a(t)$ campionato ad istanti $t = kT_c$, cioè

$$v(k) = v_a(kT_c).$$

Il segnale $v(k)$, si può scrivere come sovrapposizione finita di "fasori" a tempo discreto

$$v(k) = \sum_{h=0}^{N-1} V_h e^{j\frac{2\pi}{N}hk}$$

dove i coefficienti di Fourier V_h si possono ottenere dalla relazione:

$$V_h = \frac{1}{N}\sum_{k=0}^{N-1} v(k) e^{-j\frac{2\pi}{N}hk}$$

Il segnale $V(h) = V_h$ prende il nome di *Trasformata Discreta di Fourier* del segnale $v(k)$.

Trasformata discreta di Fourier

L'equazione

$$V_h = \frac{1}{N}\sum_{k=0}^{N-1} v(k) e^{-j\frac{2\pi}{N}hk}$$

permette di ottenere i coefficienti della trasformata a partire dal segnale.

Quindi un segnale periodico a tempo discreto si può rappresentare come sovrapposizione di un numero *finito N*, pari al numero di campioni in un periodo, di esponenziali complessi.

Trasformata discreta di Fourier

Si ricordi che, per i segnali periodici a tempo continuo, sono necessari un numero infinito (seppur numerabile) di esponenziali complessi (la serie di Fourier). Questo si può in un certo senso intuire notando che, fissata la durata (in secondi) di un periodo T del segnale $v_a(t)$, e pensando $v(k) = v_a(kT_c)$, sono necessari $N = T/T_c$ campioni per periodo. Quando T_c tende a zero, N tende ad infinito mostrando che, riducendo il passo di campionamento T_c, sono necessari un numero crescente di "fasori" per rappresentare il segnale.

La funzione FFT

fft(x) :calcola la DFT del vettore x utilizzando l'algoritmo fft.

La sintassi:

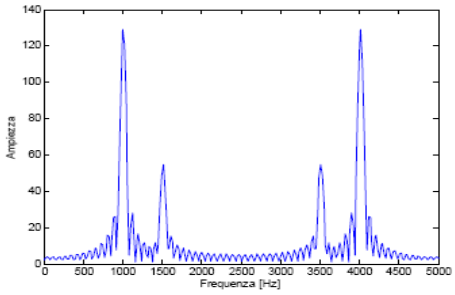
fft(x,N)

calcola la FFT su N punti, aggiungendo zeri se x ha meno di N punti, e troncando se ne ha di più.

FFT esempio

•Il seguente script calcola e visualizza la FFT di un segnale composto da due sinusoidi rispettivamente a 1000 e 1500 Hz.

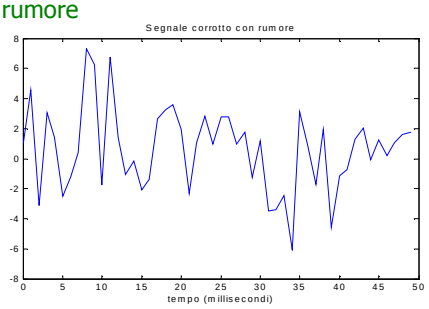
```
t=0:(0.2e-3):63*0.2e-3; % vettore dei tempi
s1=4*cos(2*pi*1000*t+0.2*pi); % generazione del segnale x dato
s2=1.5*cos(2*pi*1500*t); % dalla somma di due sinusoidi
x=s1+s2;
X=abs(fft(x,256)); % calcolo dell'FFT su 256 punti
f=linspace(0,1/0.2e-3,256); % vettore delle frequenze
plot(f,X);
xlabel('Frequenza [Hz]');
ylabel('Ampiezza');
```



FFT esempio

Il seguente script calcola e visualizza la FFT di un segnale composto da due sinusoidi a cui è aggiunto del rumore

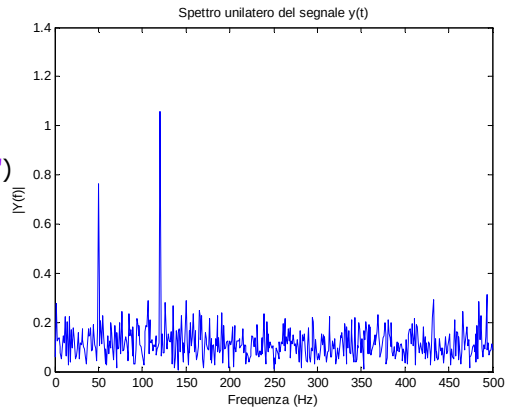
```
Fs = 1000;                    % Frequenza di campionamento
T = 1/Fs;                     % Tempo di campionamento
L = 1000;                     % Lunghezza del segnale
t = (0:L-1)*T;                % Vettore tempo
% Somma di una sinusoide a 50 Hz e una a 120 Hz
x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
y = x + 2*randn(size(t));  % Sinusoide più rumore
plot(Fs*t(1:50),y(1:50))
title('Segnale corrotto con rumore')
xlabel('tempo (millisecondi)')
```



FFT esempio

È difficile identificare le componenti in frequenza semplicemente guardando il segnale. Convertendo il segnale nel dominio della frequenza usando la FFT:

```
NFFT = 2^nextpow2(L); % Successiva potenza di 2 dalla lunghezza del segnale
Y = fft(y,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1);

% Plottiamo lo spettro unilatero
% del segnale y(t).
plot(f,2*abs(Y(1:NFFT/2+1)))
title('Spettro unilatero del segnale y(t)')
xlabel('Frequenza (Hz)')
ylabel('|Y(f)|')
```



Approfondimenti Trasformata discreta di Fourier

Il calcolo della trasformata discreta di Fourier di N moltiplicazioni ed N somme (complesse) per ogni valore V_h, $h = 0; 1; :::; N - 1$.

$$V_h = \frac{1}{N}\sum_{k=0}^{N-1} v(k)\, e^{-j\frac{2\pi}{N}hk}$$

Poiché si debbono calcolare N valori, in totale sono richieste N^2 somme ed N^2 moltiplicazioni. Questo si può abbreviare dicendo che il calcolo della DFT ha complessità di calcolo $\boldsymbol{O(N^2)}$ per gli algoritmi ottimizzati la complessità di calcolo è $\boldsymbol{O(N \log_2 N)}$.

Ecco l'importanza di avere un vettore di lunghezza proporzionale a 2^n

a titolo esemplificativo, che per $N = 1024$, $N^2 = 1048576$ mentre $N \log_2 N = 10240$; quindi,per $N = 1000$, il numero di operazioni necessarie viene ridotto di un ordine 100.

Approfondimenti Trasformata discreta di Fourier

Una descrizione semplice di queste equazioni è che i numeri complessi V_h rappresentano l'ampiezza e la fase di diverse componenti sinusoidali del "segnale" in ingresso $v(k)$.

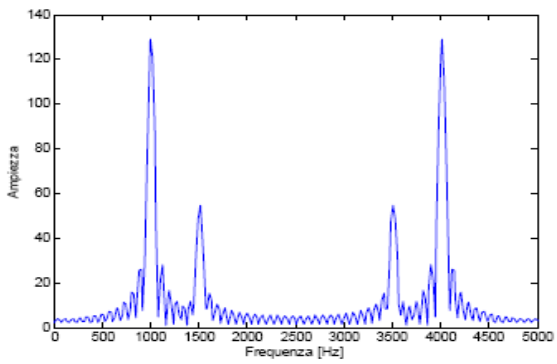
$$V_h = \frac{1}{N}\sum_{k=0}^{N-1} v(k)\, e^{-j\frac{2\pi}{N}hk}$$

Esprimendo gli V_h in forma polare, possiamo ottenere immediatamente l'ampiezza delle sinusoidi A_h e la fase φ_h da modulo e argomento di V_h, rispettivamente

$$A_h = |V_h| = \sqrt{\mathrm{Re}(V_h)^2 + \mathrm{Im}(V_h)^2},$$
$$\varphi_h = \arg(V_h) = \mathrm{atan}\,2\big(\mathrm{Im}(V_h), \mathrm{Re}(V_h)\big).$$

Approfondimenti Trasformata discreta di Fourier

I coefficienti della FFT godono della proprietà di *simmetria coniugata* in quanto il segnale originale è reale, quindi il modulo è una funzione pari e la fase è dispari, inoltre essendo segnali tempo discreto, sono periodici in 2*PI (fft() ti fa vedere infatti solo da 0 a 2*PI in pulsazione).



Approfondimenti Trasformata discreta di Fourier

per individuare a quale frequenza però dobbiamo normalizzare l'asse delle frequenze. La fft() fornisce i campioni nell'intervallo [0; 2*PI], tale intervallo deve corrispondere a [0: Fs] dove Fs è la frequenza di campionamento (infatti per il teorema di Nyquist campionando a Fs puoi ricostruire senza aliasing solo segnali limitati in banda a Fs/2. quindi devi crearti un "asse delle frequenze" in questi modi:

$$f = Fs * \text{linspace}\left(0,1,2\text{^}\,\text{nextpow2}(L)\right);$$

$$f = Fs * \text{linspace}\left(0,1,\text{Length}(Y)\right);$$

o velocemente:

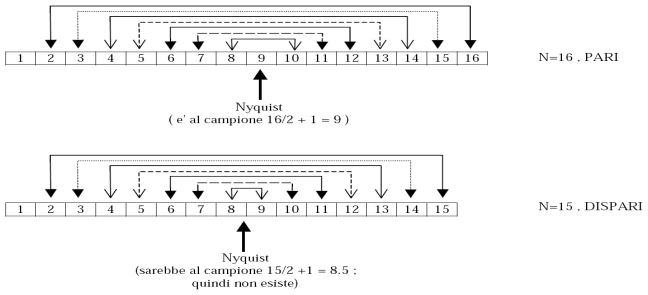
$$f = Fs/2 * \text{linspace}\left(0,1,NFFT/2+1\right);$$

Approfondimenti Trasformata discreta di Fourier

Gli schemi mettono in evidenza, tramite frecce, i campioni che godono della simmetria complesso coniugata quando si trasforma una sequenza reale.

Sono da notare infine due cose:

- il campione 1 non è simmetrico con nessuno
- Nyquist esiste solo se il vettore è di lunghezza pari (quando esiste non deve soddisfare alcuna simmetria).



Approfondimenti Trasformata discreta di Fourier

Possiamo fare delle osservazioni:

1. gli indici dei vettori in Matlab incominciano SEMPRE da 1 e mai da 0;
2. la componente continua è quindi il campione di posto 1 (visto che quello di posto 0 non esiste);
3. il campione al Nyquist occupa la posizione (N/2 + 1) dove N è ovviamente la lunghezza del vettore. Nel nostro esempio il Nyquist è quindi il campione numero 9;
4. attenzione alle simmetrie! Visto che il segnale nel tempo p(n) era reale, la sua trasformata deve avere modulo con simmetria pari e fase con simmetria dispari;

Trasformata inversa di Fourier

IFFT(X) è la trasformata inversa di Fourier di X.

La sintassi:

IFFT(X)

IFFT(X) restituisce la trasformata inversa di Fourier del vettore X. Se X è una matrice, il comando restituisce la trasformata inversa di ogni colonna della matrice.
IFFT controlla X, al fine di verificare se i vettori in X lungo la dimensione in considerazione hanno simmetria coniugata. Se si, il calcolo è più veloce e l'uscita è reale.

IFFT(X,N)

IFFT(X,N) è la trasformata inversa su N-punti.

Esempio trasformata inversa di Fourier

FFT ad N frequenze

```
Fs = 1000;                  % Frequenza di campionamento
T = 1/Fs;                   % Tempo di campionamento
L = 1000;                   % Lunghezza del segnale
t = (0:L-1)*T;              % Vettore tempo
x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);  % Somma di una sinusoide a 50 Hz e una a 120 Hz
y = x + 2*randn(size(t));  % Sinusoide più rumore
NFFT = 2^nextpow2(L);   % Successiva potenza di 2 dalla lunghezza del segnale
Y = fft(y,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1); % trovo lo spettro in potenza
% preparo il vettore di cui calcolare la IDFT, lasciando elementi non nulli solo in corrispondenza
delle frequenze stimate
vettore = ones(1,length(Y));
vettore(N+1:end-N)=0;
vettore(NFFT/2+1) = 1;
vettore=Y.*vettore;
segnaleRicostruito = real(ifft(vettore,NFFT))*L;  % ricostruzione del segnale
```

Esempio trasformata inversa di Fourier

FFT con soglia

```
% uso tanti punti FFT quanti sono i campioni del segnale per avere
% il segnale ricostruito di lunghezza pari al segnale iniziale
Y = fft(y,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1); % trovo lo spettro in potenza
% calcolo le ampiezze ed il loro valore massimo
ampiezze = abs(Y);
[maxAmpiezza indice] = max(ampiezze);
% metto a zero tutte le ampiezze delle sinusoidi che non interessano
Y(find(ampiezze(:) < maxAmpiezza*soglia)) = 0;
% ricostruzione del segnale
segnaleRicostruito = real(ifft(Y,NFFT))*L;
```

La Trasformata Wavelet

$$f(x) = \sum_k c_{j_0,k}\varphi_{j_0,k} + \sum_{j=j_0}^{j_l}\sum_k d_{j,k}\psi_{j,k}$$

La Trasformata di Fourier

$$f(x) = x_0 + \sum_n a_n \cos(nx) + \sum_n b_n \sin(nx)$$

La Trasformata Wavelet

In generale è rappresentata da una particolare coppia di funzioni (φ, ψ) formanti un Sistema Ortonormale Completo (CONS) (basi del relativo spazio funzionale)

Aventi le seguenti proprietà:

- Norma unitaria
- Ortogonalità
- Componibilità lineare

Analogie: trasformata di Fourier e Wavelet

Fourier $f(x) = x_0 + \sum_n a_n \cos(nx) + \sum_n b_n \sin(nx)$

Coefficienti

Wavelet $f(x) = \sum_k c_{j_0,k} \varphi_{j_0,k} + \sum_{j=j_0}^{j_l} \sum_k d_{j,k} \psi_{j,k}$

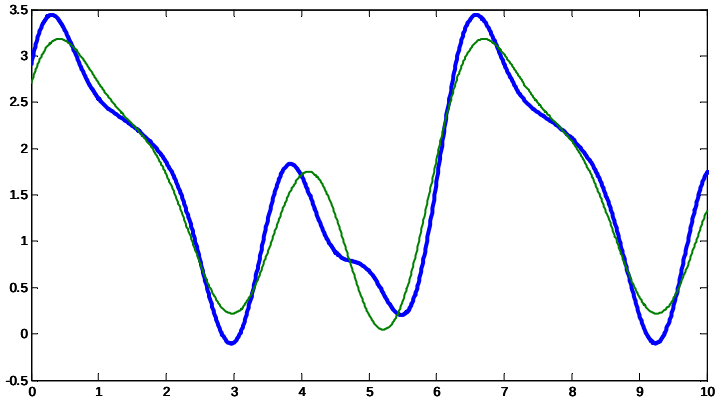
La trasformata di Fourier

$$f(x) = x_0 + \sum_n a_n \cos(nx) + \sum_n b_n \sin(nx)$$

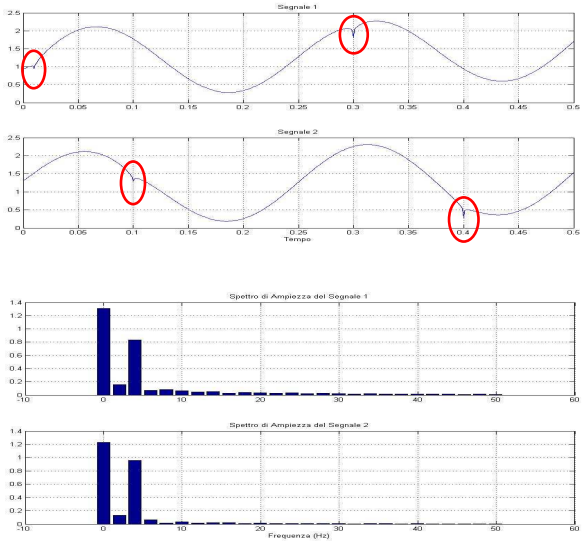
Possiamo ricostruire un segnale con buona approssimazione utilizzando un numero finito di armoniche

La trasformata di Fourier

$$y(x) = 1{,}5 + \sin(x + 45°) + 0{,}8\cos(2x) + 0{,}5\cos(3x)$$



Limiti della trasformata di Fourier



Caratteristica delle Wavelet

Base funzionale

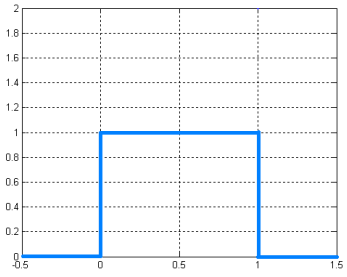
(CONS)

φ — Funzione di approssimazione

ψ — Funzione di dettaglio

Funzione di approssimazione (Scaling)

$$\varphi(x) = \begin{cases} +1 & 0 \le x < 1 \\ 0 & \textit{altrove} \end{cases}$$

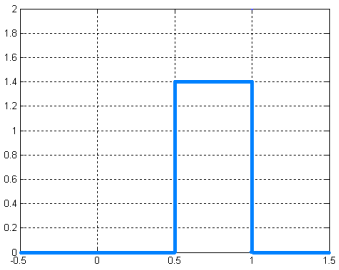


Funzione di approssimazione

Può essere scalata e traslata come:

$$\varphi_{j,k}(x) = 2^{\frac{j}{2}} \cdot \varphi\left(2^j x - k\right)$$

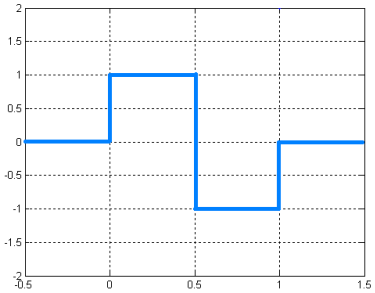
$$\varphi_{1,0}(x)$$



Funzione di dettaglio

Wavelet di Haar

$$\psi(x) = \begin{cases} +1 & 0 \le x < \frac{1}{2} \\ -1 & \frac{1}{2} \le x < 1 \\ 0 & altrove \end{cases}$$

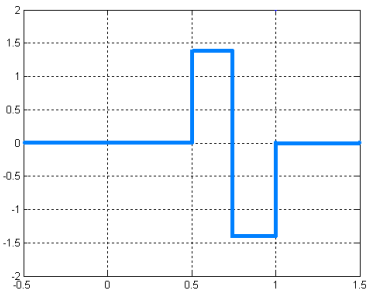


Funzione di dettaglio

Può essere scalata e traslata come:

$$\psi_{j,k}(x) = 2^{\frac{j}{2}} \cdot \psi\left(2^j x - k\right)$$

$\psi_{1,0}(x)$

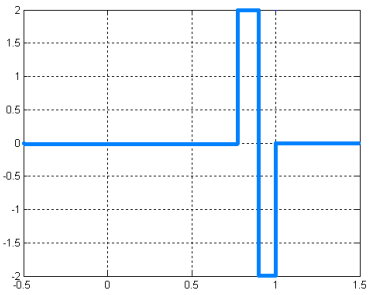


Funzione di dettaglio

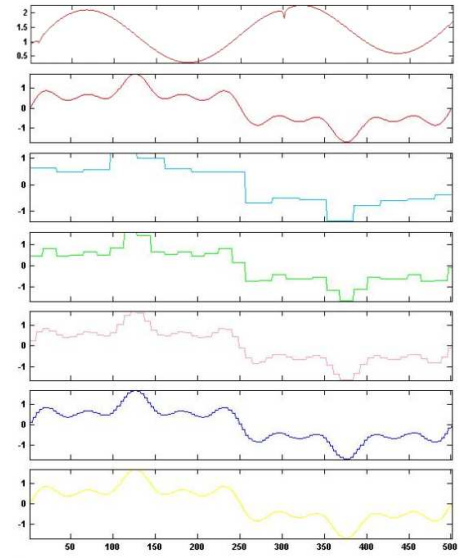
Può essere traslata e scalata come:

$$\psi_{j,k}(x) = 2^{\frac{j}{2}} \cdot \psi\left(2^j x - k\right)$$

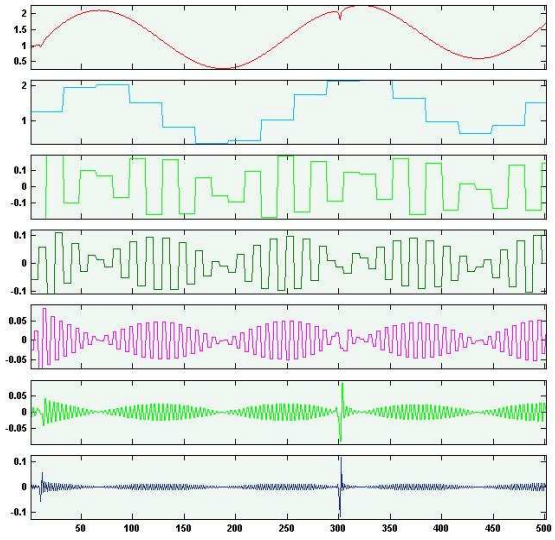
$\psi_{2,0}(x)$



Funzione di approssimazione

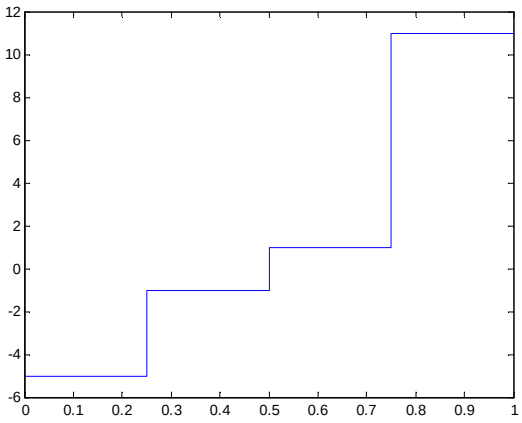


Funzione di dettaglio



Esempi Wavelet

Prendiamo una f(x)=[-5, -1, 1, 11] prima scomposizione



Esempi Wavelet

Prima scomposizione:

$$f(x) = c_1\varphi_{00} + c_2\psi_{00} + c_3\psi_{10} + c_4\psi_{11}$$

$$B = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix}$$

$$INV(B) * f(x)' = 1.5 \quad -4.5 \quad -2 \quad -5$$

Esempi Wavelet

Seconda scomposizione:

$$f(x) = c_1\varphi_{10} + c_2\varphi_{11} + c_3\psi_{10} + c_4\psi_{11}$$

$$V = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

$$INV(V) * f(x)' = -3 \quad 6 \quad -2 \quad -5$$

Esempi Wavelet

Terza scomposizione:

$$f(x) = c_1\varphi_{20} + c_2\varphi_{21} + c_3\varphi_{22} + c_4\varphi_{23}$$

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$INV(A) * f(x)' = -5 \quad -1 \quad 1 \quad 11$$

Esempi Wavelet

Considerazioni:

$$-5(\varphi_{20}) \quad -1(\varphi_{21}) \quad +1(\varphi_{22}) \quad +11(\varphi_{23})$$
$$-3(\varphi_{10}) \quad +6(\varphi_{11}) \quad -2(\psi_{10}) \quad -5(\psi_{11})$$
$$1{,}5(\varphi_{00}) \quad -4{,}5(\psi_{00}) \quad -2(\psi_{10}) \quad -5(\psi_{11})$$

$$\frac{-5-1}{2} = -3$$

Le tre soluzioni dei sistemi di equazioni visti in precedenza notiamo alcuni particolari

Esempi Wavelet

Considerazioni:

$$-5(\varphi_{20}) \quad -1(\varphi_{21}) \quad +1(\varphi_{22}) \quad +11(\varphi_{23})$$
$$-3(\varphi_{10}) \quad +6(\varphi_{11}) \quad -2(\psi_{10}) \quad -5(\psi_{11})$$
$$1{,}5(\varphi_{00}) \quad -4{,}5(\psi_{00}) \quad -2(\psi_{10}) \quad -5(\psi_{11})$$

$$\frac{1+11}{2} = 6$$

Le tre soluzioni dei sistemi di equazioni visti in precedenza notiamo alcuni particolari

Esempi Wavelet

Considerazioni:

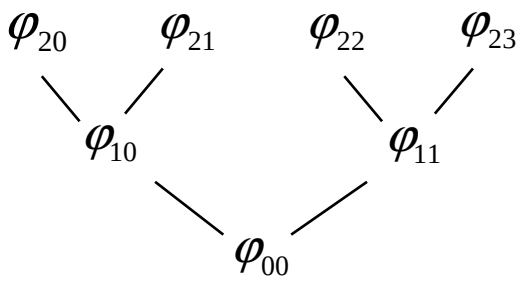
$$-5(\varphi_{20}) \quad -1(\varphi_{21}) \quad +1(\varphi_{22}) \quad +11(\varphi_{23})$$
$$-3(\varphi_{10}) \quad +6(\varphi_{11}) \quad -2(\psi_{10}) \quad -5(\psi_{11})$$
$$1{,}5(\varphi_{00}) \quad -4{,}5(\psi_{00}) \quad -2(\psi_{10}) \quad -5(\psi_{11})$$

$$\frac{-3+6}{2} = 1{,}5$$

Le tre soluzioni dei sistemi di equazioni visti in precedenza notiamo alcuni particolari

Esempi Wavelet

Schema piramidale



Esempi Wavelet

Coefficienti di dettaglio

$$
\begin{array}{llll}
-5(\varphi_{20}) & -1(\varphi_{21}) & +1(\varphi_{22}) & +11(\varphi_{23}) \\
-3(\varphi_{10}) & +6(\varphi_{11}) & -2(\psi_{10}) & -5(\psi_{11}) \\
1{,}5(\varphi_{00}) & -4{,}5(\psi_{00}) & -2(\psi_{10}) & -5(\psi_{11})
\end{array}
$$

$$-5-(-3)=-2$$

Esempi Wavelet

Coefficienti di dettaglio

$$
\begin{array}{llll}
-5(\varphi_{20}) & -1(\varphi_{21}) & +1(\varphi_{22}) & +11(\varphi_{23}) \\
-3(\varphi_{10}) & +6(\varphi_{11}) & -2(\psi_{10}) & -5(\psi_{11}) \\
1{,}5(\varphi_{00}) & -4{,}5(\psi_{00}) & -2(\psi_{10}) & -5(\psi_{11})
\end{array}
$$

$$1-6=-5$$

Esempi Wavelet

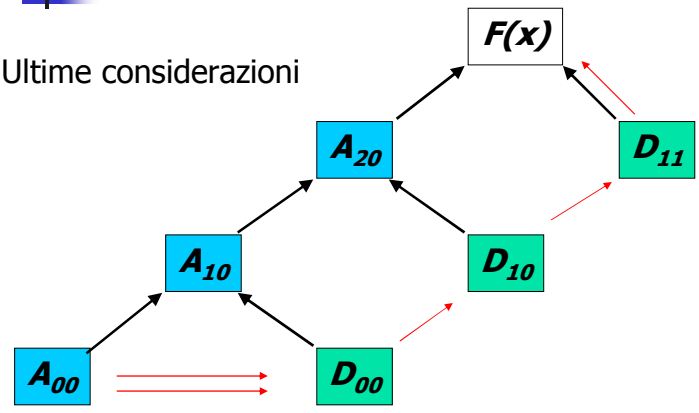
Coefficienti di dettaglio

$$-5(\varphi_{20}) \quad -1(\varphi_{21}) \quad +1(\varphi_{22}) \quad +11(\varphi_{23})$$
$$-3(\varphi_{10}) \quad +6(\varphi_{11}) \quad -2(\psi_{10}) \quad -5(\psi_{11})$$
$$1{,}5(\varphi_{00}) \quad -4{,}5(\psi_{00}) \quad -2(\psi_{10}) \quad -5(\psi_{11})$$

$$-3-1{,}5=-4{,}5$$

Esempi Wavelet

Ultime considerazioni



$$f(x)=A_{00}+D_{00}+D_{10}+D_{11}$$

Trasformata Wavelet

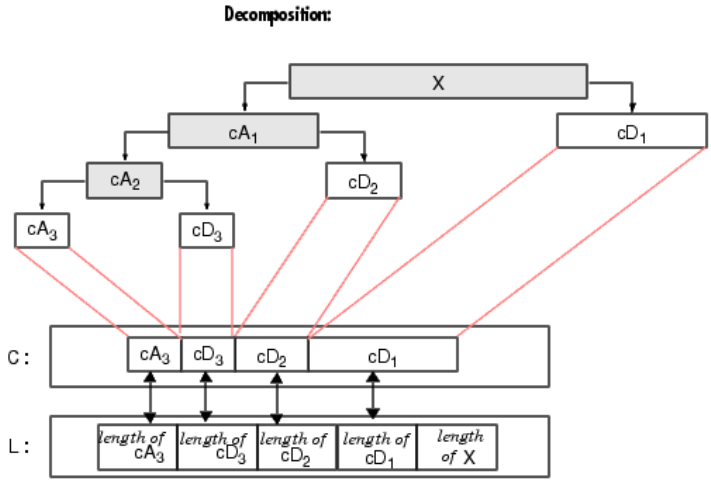
wavedec :calcola la TW del vettore x.

La sintassi:

[C,L] = wavedec(X,N,'wname')

calcola la TW del segnale X al livello N, usando il filtro wavelet scelto attraverso "wname". N deve essere un intero positivo. In uscita si hanno 2 vettori: C contiene i coefficienti wavelet dei livelli di scomposizione richiesti mentre L contiene le lunghezze dei relativi coefficienti per ogni livello di scomposizione.

Trasformata Wavelet



Trasformata Wavelet

wrcoef : ricostruisce un singolo livello della TW monodimensionale.

La sintassi:

X = wrcoef('type',C,L,'wname',N)

"type": può essere 'a' per indicare approssimazione oppure 'd' per indicare dettaglio
"C, L": sono i vettori calcolati con il comando wavedec
"wname": è il filtro wavelt utilizzato (in wavedec)
"N": è il livello di cui si richiede la ricostruzione

Trasformata Wavelet

waverec : opera la TW inversa

La sintassi:

X = waverec(C,L,'wname')

"C, L": sono i vettori calcolati con il comando wavedec
"wname": è il filtro wavelet utilizzato (in wavedec)

In output si ottiene "X" ossia il segnale ricostruito nel dominio del tempo

Trasformata Wavelet

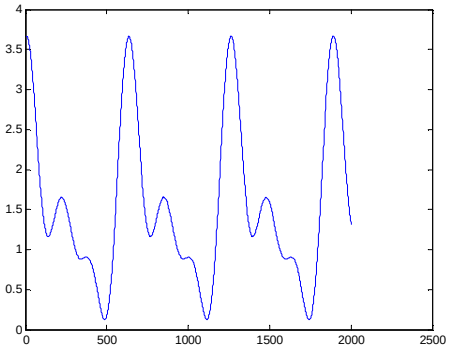
Generiamo il seguente segnale in Matlab:

$$y(x) = 1{,}5 + \sin(x + 45°) + 0{,}8\cos(2x) + 0{,}5\cos(3x)$$

v
plot(y)

Osserviamo quanto vale il segnale nel punto di ascissa 1000:

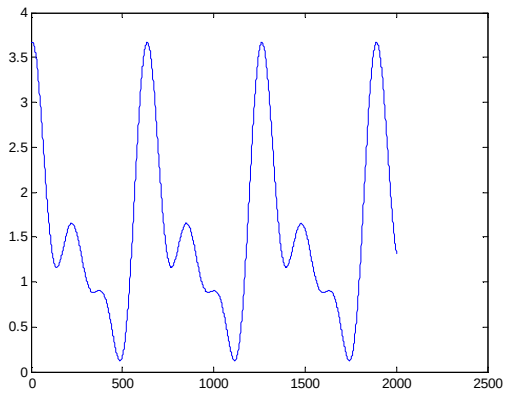
```
>> y(1000)

ans =

    0.9034
```



Trasformata Wavelet

Variamo l'ampiezza di tale punto:

>> y(1000)=0.905;



È evidente che l'alterazione non risulta evidente!!

Trasformata Wavelet

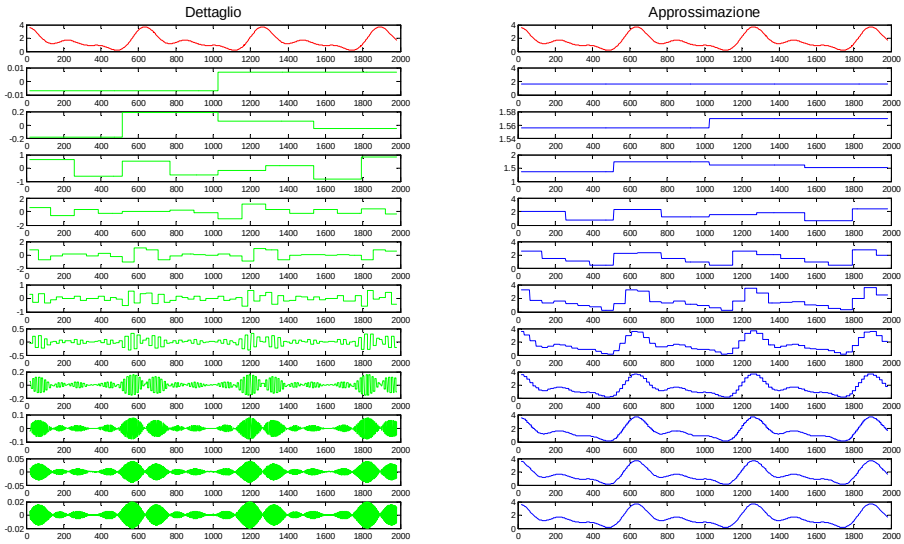
Applichiamo la trasformata wavelet; per far questo prepariamo un file.m in Matlab

```
s=y;
w='haar';

n=round(log2(length(s)));

S1=figure;
set(S1, 'Units', 'normalized', 'Position', [0 0 1 0.9]);
plot(s), title ('Segnale originale', 'FontSize', 16)

[c,l]=wavedec(s,n,w);

for i=1:n
    D(i,:)=wrcoef('d',c,l,w,i);
    A(i,:)=wrcoef('a',c,l,w,i);
end
```

Trasformata Wavelet

```
tt=1+20:length(y)-20;

S2=figure;
set(S2, 'Units', 'normalized', 'Position', [0 0 1 0.9]);
hold on
subplot(n+1,2,1);
plot(tt,s(tt),'r')
title ('Dettaglio','FontSize', 16);

for i=2:2:2*n
    subplot(n+1,2,i+1);
    plot(tt,D(n-(i/2)+1,tt),'g');
end

subplot(n+1,2,2);plot(tt,s(tt),'r')
title ('Approssimazione','FontSize', 16);

for i=3:2:2*n+1
    subplot(n+1,2,i+1);
    plot(tt,A(n-((i-1)/2)+1,tt),'b');
end
```

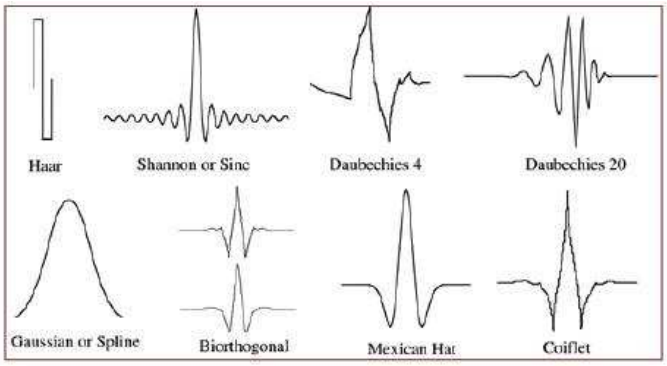
Trasformata Wavelet



Trasformata Wavelet

È evidente che non si riesce a rilevare nulla. Ciò è dovuto al fatto che abbiamo utilizzato il filtro wavelt di Haar, puramente didattico.

Esistono a riguardo numerosi filtri wavelet che bisogna scegliere in maniera accurata ai fini dell'analisi



Trasformata Wavelet

Proviamo quindi a sostituire nel programmino Matlab al posto del filtro wavelet

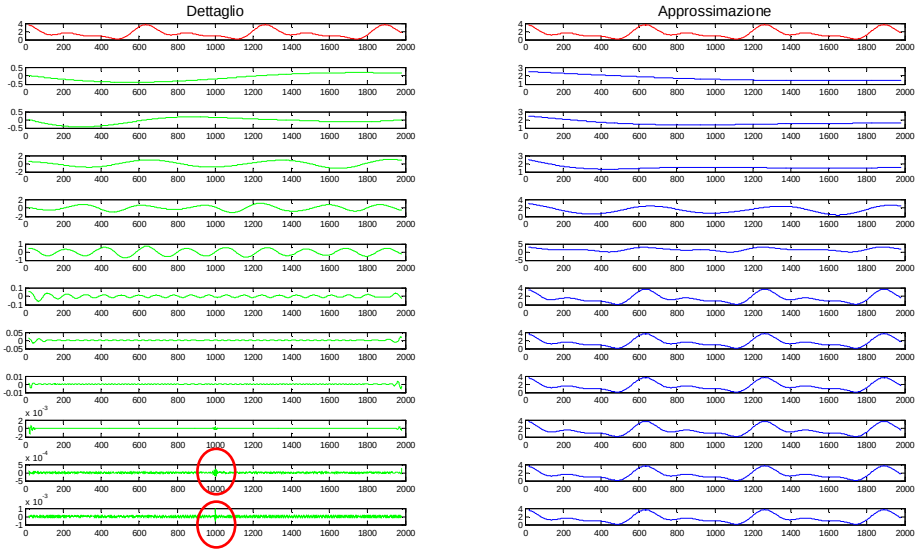
w='haar';

il filtro wavelet

w='db13';

Osserviamo come cambiano i risultati.

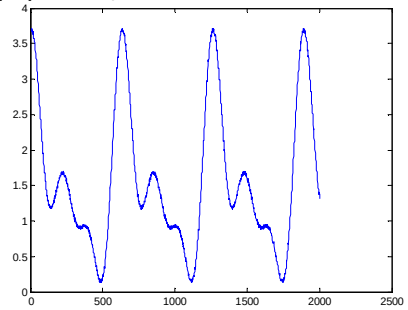
Trasformata Wavelet



Trasformata Wavelet

Complichiamo l'esercizio: aggiungiamo del rumore al segnale. A seguito della generazione del segnale aggiungere:

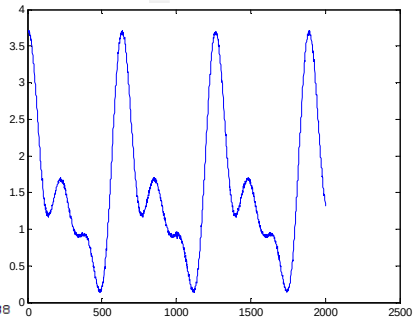
```
noise=0.05*rand(1,length(y));
y=y+noise;
```



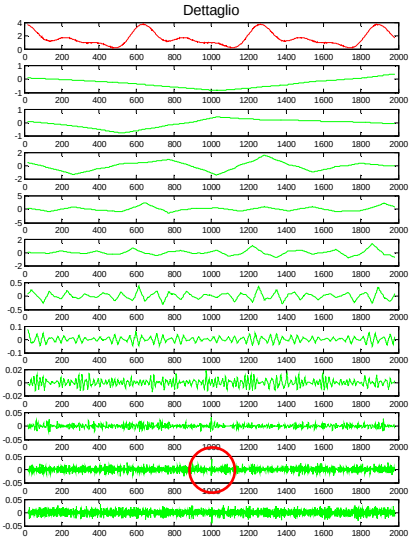
```
>> y(1000:1005)

ans =

    0.9436    0.9291    0.9436    0.9520    0.9082    0.9288
```

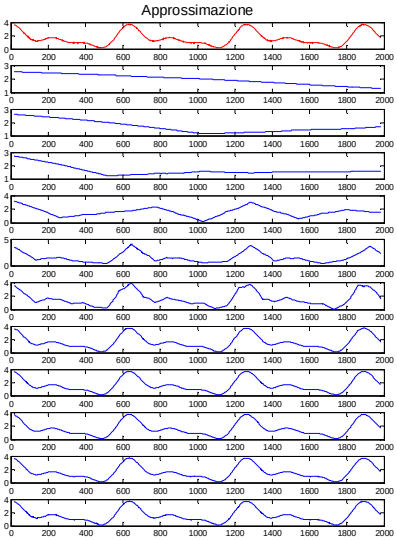
Spike

```
>> y(1000)=0.93;
y(1001)=0.97;
y(1002)=0.97;
y(1003)=0.97;
y(1004)=0.93;
y(1005)=0.87;
```

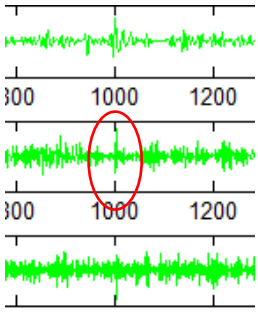


Trasformata Wavelet





Trasformata Wavelet



Lo spike è stato identificato con una ‘db3’!!

Molto importante per questo tipo di studio è la densità di campionamento!!

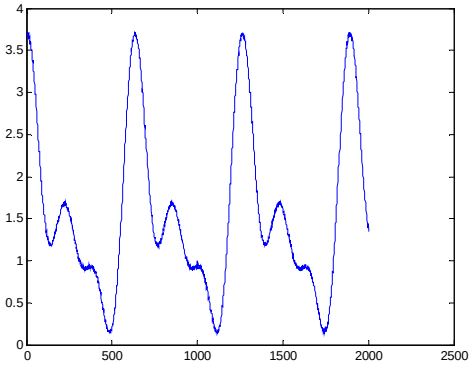
Infatti se avessimo alterato un solo punto avremmo perso la dinamica dello spike.

Il prossimo esempio chiarifica il concetto di alta densità di campionamento.

Trasformata Wavelet

Generiamo uno spike alterando un solo punto:

```
>> y(1000)=1;
```



Applichiamo a tale segnale la stessa analisi proposta precedentemente.

Trasformata Wavelet

È evidente che abbiamo perso la dinamica dello spike!!