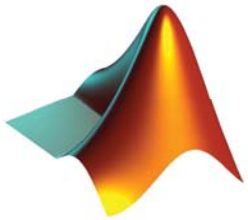
L'ambiente Matlab
per le applicazioni industriali
(Parte 1 – Panoramica)



Un pò di storia

- **Acronimo di Matrix Laboratory**
- **Creato alla fine degli anni '70 alla New Mexico University da Chris Molen (ed altri)**
- **Lo scopo iniziale era fornire un ausilio ai corsi di algebra lineare e di calcolo numerico per studenti senza conoscenze di programmazione**
- **Integra gradatamente librerie già disponibili (es. Linpack)**
- **Nel 1984 inizia la commercializzazione da parte di Mathworks**
- **Evolve successivamente come suite completa di supporto alla ricerca scientifica ed allo sviluppo di applicazioni ad elevato contenuto scientifico**

Un pò di storia

- **Oggi ha diffusione universale ed è maggiormente orientato alle applicazioni industriali mediante la realizzazione di parecchi toolbox specifici**
- **Esistono diverse soluzioni di acquisto parziale (a nessuno serve tutto…) che consentono di accedere alle sole parti utili**
- **Le prossima versione (in beta testing) sarà definitivamente object-oriented**
- **L'impiego didattico rimane attuale ma molto spesso è limitato al solo prodotto base (il motore di calcolo) usato in modalità interpretata**

Perché Matlab?

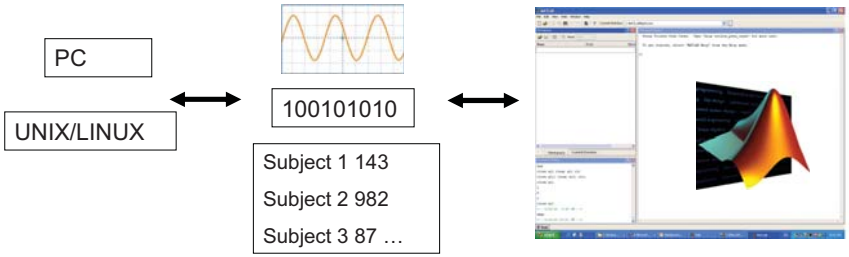
Acquisizione dati

- MATLAB fornisce primitive per acquisire ed analizzare dati da qualsiasi sorgente digitale

Perché Matlab?

Importazione dati

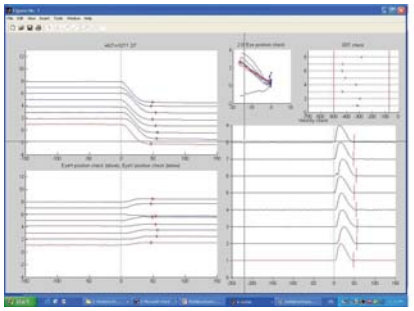
- Si possono importare in MATLAB dati in qualunque formato e da qualunque piattaforma



Perché Matlab?

Strumenti di analisi

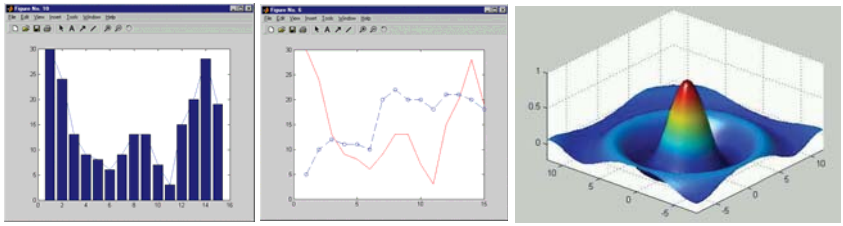
- Esistono librerie per compiere qualunque tipo di analisi sui dati
- E' possibile costruire qualunque tipo di rappresentazione grafica



Perchè Matlab?

Grafica N-dimensionale

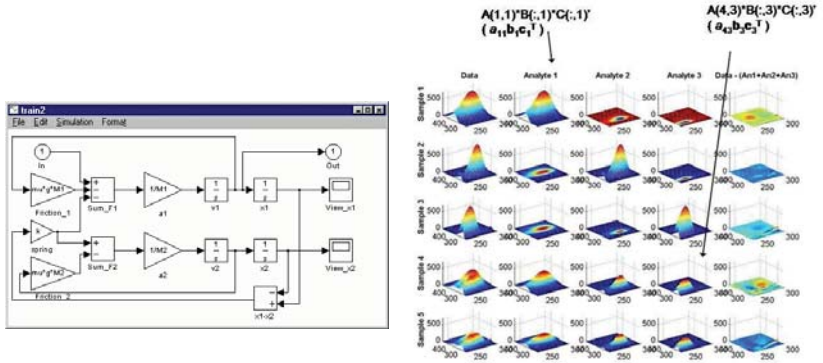
- Sono disponibili tutte le opzioni di grafica da 2 a 4 dimensioni
- Possibile controllo totale della formattazione e di qualunque elemento visivo



Perchè Matlab?

Modellazione dinamica

- E' possibile costruire modelli dell'interazione di sistemi dinamici complessi e verificarli su dati sperimentali



Perchè Matlab?

- **Disponibilità di sw**
  - **Matlab contiene un numero enorme di funzioni pronte per l'uso, organizzate in toolbox tematici e perfettamente commentate**
  - **Il sito di Mathworks è ricchissimo di esempi ed applicazioni pronte per l'uso**
  - **Sono disponibili in rete risorse di pubblico dominio che estendono ulteriormente questo insieme a specifici problemi applicativi**
  - **Le probabilità di trovarsi il lavoro già fatto sono parecchio elevate…**

Alcuni fatti …

- **Matlab serve solo ai matematici**
  - FALSO! anzi, è vero il contrario. Serve poco (o nulla) ai matematici, è invece pensato e sviluppato per la progettazione di sistemi, quindi per l'ingegneria
- **Matlab è lento**
  - FALSO! è lento solo se usato in modalità interpretata. Il compilatore genera eseguibili di discreta efficienza, specie se si usano bene le funzioni già ottimizzate.
- **Matlab costa troppo**
  - FALSO! Sì, il costo iniziale è oggi il vero limite alla diffusione di Matlab nell'industria. Solo le grandi aziende usano Matlab originale, le altre…si arrangiano. Ma rispetto ai costi umani di sviluppo sw, Matlab costa poco.

Uso di Matlab

Valutare espressioni utilizzando variabili

- **Le espressioni immesse sono interpretate e valutate immediatamente dal sistema**
- **Le variabili sono identificativi usati per memorizzare valori**
- **Le variabili calcolate possono essere richiamate in seguito**
- **Le variabili non vanno dichiarate (dichiarazione = assegnazione)**

Variabile* = *Espressione

Oppure

Espressione

N.B. i nomi delle variabili sono case-sentitive

Uso di Matlab

Lavorare con le matrici

- **Matlab opera normalmente con un unico oggetto di base, una matrice rettangolare di numeri (array) indicizzata con (…,…)**
- **Una matrice è una struttura indicizzata da 2 valori: riga e colonna**
- **Il numero di righe o colonne è arbitrario**
- **Gli indici partono sempre da 1**

Uno scalare è un singolo numero rappresentato da matlab come una matrice 1x1.

Un vettore è un array mono-dimensionale di numeri rappresentato come matrice n x 1 (vettore colonna) o 1 x n (vettore riga)

Si usano "," e ";" per separare gli elementi nella assegnazione

Operazioni con matrici

Una matrice vuota può essere creata con [ ]

L'operatore [ ] può essere usato per cancellare righe o colonne

Le funzioni size e ndims ritornano il numero di elementi e di dimensioni di una variabile matriciale

La struttura 2D è generalizzabile ad un numero arbitrario di dimensioni (matrici N-dimensionali)

L'operatore ':' consente di estrarre sottomatrici mediante la definizione di intervalli di righe o colonne

Operazioni con matrici

Tutte le normali operazioni di calcolo sono utilizzabili sulle matrici combinate con scalari

- **+ Addizione**
- **- Sottrazione**
- *** Moltiplicazione**
- **/ Divisione**
- **^ Elevamento a potenza**

Le operazioni con scalari sono sempre intese come applicate ai singoli elementi della matrice

Operazioni con matrici

Le operazioni tra matrici assumono particolari proprietà

Addizione e Sottrazione

Le dimensioni delle matrici devono essere uguali

Moltiplicazione

Ci sono due possibili implementazioni

*prodotto algebrico (righe x colonne) – operatore **

*prodotto elemento x elemento – operatore .**

I vincoli sono diversi nei due casi

Operazioni con matrici

Operazioni specifiche per matrici

La trasposta si forma scambiando tra loro le righe con le colonne –operatore '

inv	**inversa di una matrice**
det	**determinante di una matrice**
trace	**traccia di una matrice**
rank	**rango di una matrice**
zeros	**matrice nulla**
ones	**matrice unaria**
diag	**matrice diagonale**
eye	**matrice identità**

Esistono numerosissime altre operazioni specifiche per le matrici

Gli scripts di Matlab

Raggruppano una serie di comandi senza eseguirli (modalità differita)

•Matlab può eseguire sequenze di comandi contenuti in un file
•I file che contengono comandi Matlab devono avere estensione '*.m'
•Gli M-files si possono scrivere e salvare con l'apposito editor
•Gli M-files sono eseguibili al command prompt come un comando
•Gli M-files possono chiamare altri M-files

N.B. Per essere eseguibile, un M-file deve avere il suo path settato nella configurazione di Matlab

Gli scripts di Matlab

Vantaggi degli M-files

- **Sviluppo facilitato dall'editor**
- **Possibili modifiche a valle dell'esecuzione**
- **Leggibilità/Portabilità – si possono aggiungere commenti tramite il simbolo '%' per facilitare la comprensione**
- **Salvare M-files è più efficiente che salvare il workspace**

Controllo di Flusso

- **Il controllo di flusso consente a Matlab di superare la semplice funzionalità di puro calcolo**
- **Con il controllo di flusso Matlab può essere usato come un linguaggio di programmazione ad alto livello orientato alla elaborazione di matrici**
- **Il controllo di flusso è implementato tramite statement condizionali e cicli**

Statement condizionali

If, Else, and Elseif

- **Uno statement if valuta una espressione logica ed esegue un gruppo di comandi se questa è vera**
- **La list dei comandi condizionati termina con uno statement end**
- **Se l'espressione logica è falsa, tutti I comandi condizionati sono saltati**
- **L'esecuzione dello script riprende dopo lo statement end**

```
if espressione_logica
    comandi
end
```

Statement condizionali

If, Else, and Elseif

- Lo statement **else** forza l'esecuzione dei comandi successivi se l'espressione logica originale è falsa
- Solo una delle due liste viene eseguita

```
if espressione_logica
    comandi 1
else
    comandi 2
end
```

Statement condizionali

If, Else, and Elseif

- Lo statement **elseif** nidifica una nuova struttura if dopo un else
- Sono una delle liste presenti viene eseguita

```
if espressione_logica_1
    comandi 1
elseif espressione_logica_2
    comandi 2
elseif espressione_logica_3
   comandi 3
end
```

Cicli

Ciclo For

Nel ciclo **for** la lista di comandi viene eseguita un numero fissato di volte.

```
for index = inizio:incremento:fine
    comandi
end
```

Se '*incremento*' non è definito, il default è 1

Ciclo While

Nel ciclo **while** la lista di comandi viene eseguita finchè la condizione rimane vera

```
while espressione_logica
    comandi
end
```

Funzioni

- Costituiscono dei blocchi elementari di programmazione
- Consentono al codice di essere generico e riutilizzabile
- Dato un insieme di inputs, eseguono una serie di comandi e ritornano un output
- In Matlab, ogni funzione è un M-file
- E' prassi nominare il file come la funzione, cioè il file *funcname.m* contiene la funzione definita da:

```
function outargs = funcname(inargs)
```

- **return** termina il calcolo e ritorna al chiamante (opzionale alla fine del file)

Aggiungere un Help

- **Inserire sempre alcune righe di commento tra la dichiarazione della funzione e la prima riga di codice**
- **Così si abilita automaticamente al funzione help**
- **Le righe di commento sono scandite da lookfor che trova tutte le occorrenza di un termine**

```
function [y] = cube(x)
% Calcola il cubo di x
y = x*x*x;
```

```
>> help cube
Calcola il cubo di x
```

Importare dati da file

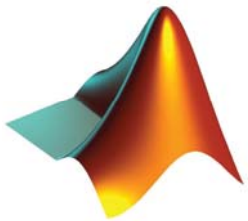
- **Il comando load importa dati da un generico file ASCII in una variabile Matlab (matrice)**

  Nome_variabile = load('filename')

- **Ci sono restrizioni sulla struttura del file, per cui funziona solo su file strutturati in modo regolare (es. tabelle)**
- **Molto potente ma di uso limitato**
- **Esiste il comando simmetrico save**

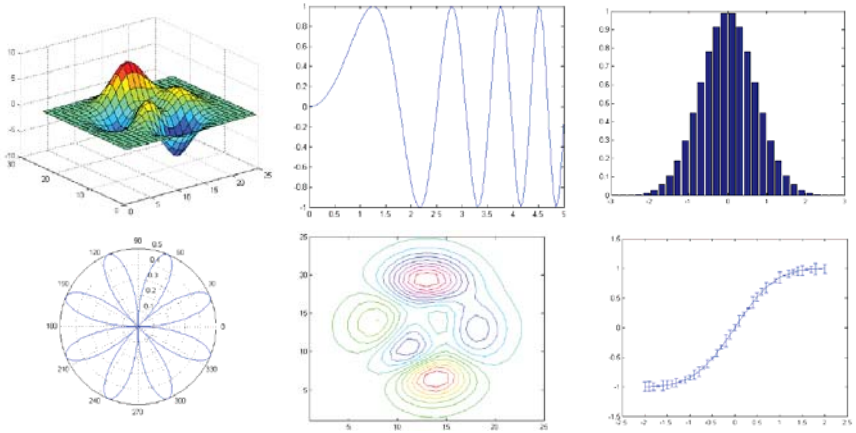
Spesso si ricorre ad una gestione manuale della importazione dei dati, che la rende una delle principali cause di errore nello sviluppo di codice Matlab

L'ambiente Matlab
per le applicazioni industriali
(Parte 2 – Grafica 2D e 3D)



Cosa si può fare

- **Matlab ha un ottimo motore per la generazione di grafici, in grado di produrre qualuqnue tipo di rappresentazione dei dati**



Generazione dei dati

- **Il motore grafico di Matlab non usa le funzioni ma solo array di numeri. Date quindi le funzioni**
  - **a=t^2**
  - **b=sin(2*pi*t)**
  - **c=exp(-10*t)**
  - **d=cos(4*pi*t)**
  - **e=2*t^3-4*t^2+t**
- **queste vanno calcolate in uno specifico intervallo campionato**
- **L'intervallo di definisce di norma con la sintassi inizio:passo:fine, oppure con linspace(inizio,fine,campioni)**

```
t=0:0.01:10;  %assegna il vettore delle ascisse
y=t.^2; %calcola il vettore delle ordinate
        % ma solo nell'intervallo specificato
```

La funzione plot()

- **La più semplice funzione di grafica è plot()**
- **Cosa succede scrivendo plot(y)?**
  - Matlab genera automaticamente una figura, disegna i punti corrispondenti ai dati y e li connette con linee
  - L'asse x non è corretto (Matlab usa gli indici come default)
- **plot(x,y) risulta simile ma ora l'asse x è corretto**
- **plot(x1,y1,s1,x2,y2,s2, …) plottano più funzioni con un solo comando**
- **Se x è una matrice, plot(x) disegna le colonne come tracce separate**

La funzione plot()

- **Se si plottano in sequenza a e b si vede solo b**
- **Matlab sostituisce ogni plot con il successivo, in assenza di altre istruzioni**
- **Per sovrapporre i due plot nella stessa figura bloccata si usa il comando hold on (hold off disabilita il blocco)**
- **Per avere I plot in figure diverse si usa il comando figure**

```
plot(t,a)
plot(t,b)
```

```
% Disegna a e b in un plot
plot(t,a);
hold on;
plot(t,b);
```

```
% Disegna due plot
plot(t,a);
figure;
plot(t,b);
```

Proprietà delle linee

- **Senza specifiche, tutti i plot sono fatti nel colore di default … blu**
- **Tutti gli attributi grafici sono modificabili selezionando le opzioni ammesse dal comando plot**

```
% linea rossa
Plot(t,a,'r');
Hold on;
% linea nera
Plot(t,b,'k');
% linea a punti verdi
Plot(t,c,'g.');
% linea a croci cyan
Plot(t,d,'cx')
% linea tratteggiata %
a cerchi magenta
Plot(t,e,'--om')
```

Line Style Specifiers

Specifier	Line Style
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line

Color Specifiers

Specifier	Color
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White

Marker Specifiers

Specifier	Marker Type
+	Plus sign
o	Circle
*	Asterisk
.	Point
x	Cross
'square' or s	Square
'diamond' or d	Diamond
^	Upward-pointing triangle
v	Downward-pointing triangle

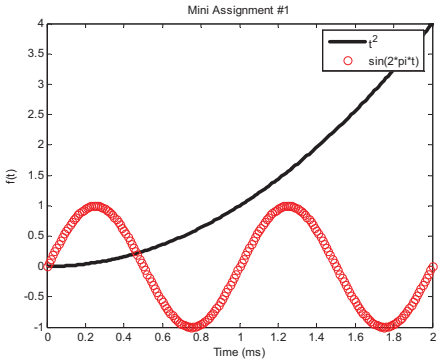
Etichette, Titolo e Legenda

- **Per aggiungere etichette agli assi x e y, si usano l comandi xlabel e ylabel**
- **Per aggiungere un titolo si usa il comando title**
- **Per aggiungere una legenda si usa il comando legend**

```
plot(t,a,t,b,'r',t,c,'--om');  %genera tutti i plot in un colpo
title('Random Plots')
xlabel('t(ms)');
ylabel('f(t)')
legend('Funzione 1','Funzione 2','Funzione 3');
```

Esempio

- **Disegnare a come linea nera spessa**
- **Disegnare b come una serie di cerchi rossi**
- **Etichettare gli assi, aggiungere titolo e legenda**

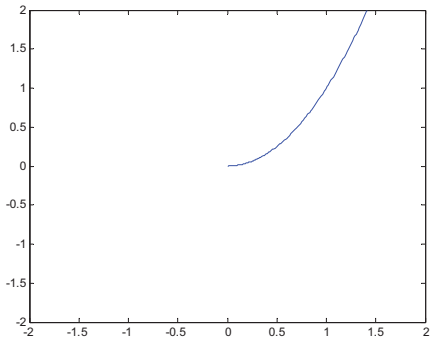



```
figure
plot(t,a,'k','LineWidth',3);
hold on;
plot(t,b,'ro')
xlabel('Time (ms)');
ylabel('f(t)');
legend('t^2','sin(2*pi*t)');
title('Mini Assignment #1')
```

Il comando axis

- **Il comando axis modifica l'intervallo visualizzato sul grafico ed aggiunge ulteriori controlli**
  - Axis([xmin xmax ymin ymax])
  - Axis equal;
  - Axis square;

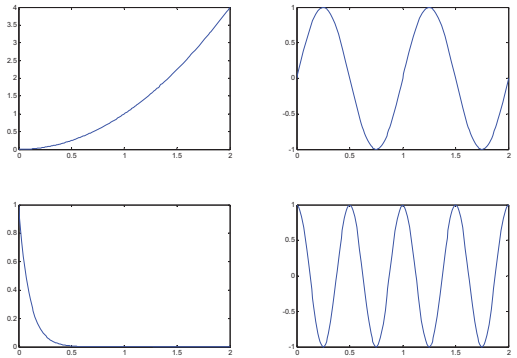
```
Figure;
Plot(t,a,'r');
Axis([-2 2 -2 2]);
```



La funzione subplot()

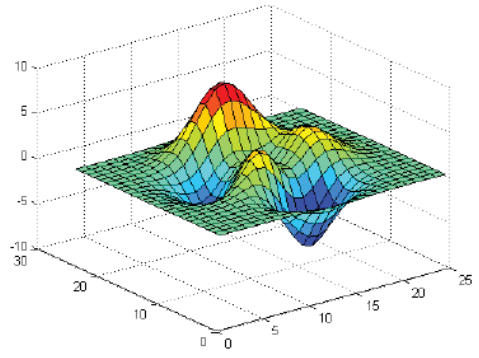
- **Con subplot è possibile realizzare una struttura matriciale di grafici omogenei, che rende più agevole la comparazione**
- **Ogni subplot identifica un particolare grafico su cui agiscono le plot successive**

```
figure;
subplot(2,2,1)
plot(t,a);
subplot(2,2,2)
plot(t,b);
subplot(2,2,3)
plot(t,c);
subplot(2,2,4)
plot(t,d);
```



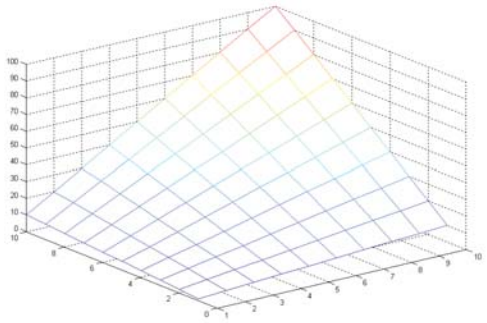
Grafici 3D

- **Matlab fornisce una vastissima scelta di opzioni per la grafica di dati in 3D**
- **Le funzioni di base utilizzate sono mesh, surf, contour pensate per visualizzare grafici del tipo Z=f(X,Y)**



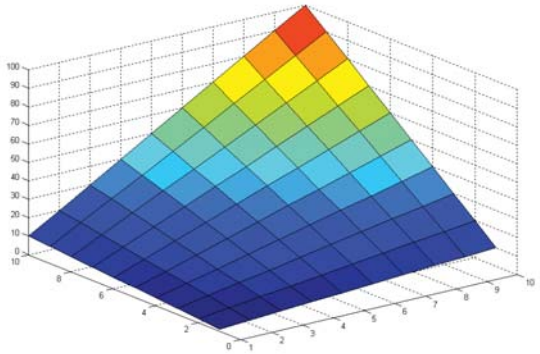
La funzione mesh

- **La funzione mesh connette una serie di punti discreti con una struttura reticolare (griglia o mesh)**
  - mesh(x,y,z) dove X(i) and Y(j) sono le posizioni dei nodi della griglia e Z(i,j) è il valore assunto in ogni nodo
  - mesh(Z) assume che X e Y siano 1..N e 1..M



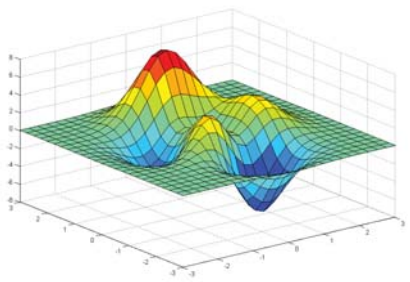
La funzione surf

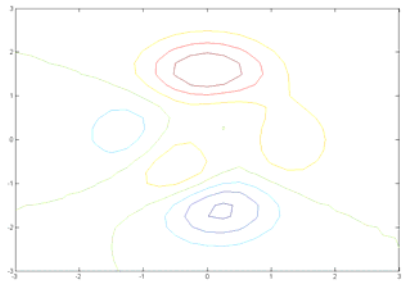
- **E' concettualmente identica alla mesh, con l'unica differenza che la griglia è riempita con sfumature di colore**



La funzione contour

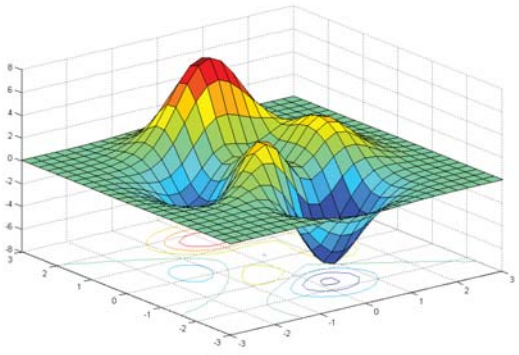
- **Proietta i punti di uguale altezza in 3D (curve di livello) su un piano 2D sottostante**
- **Per il resto è analoga a surf o mesh – contour(x,y,z)**





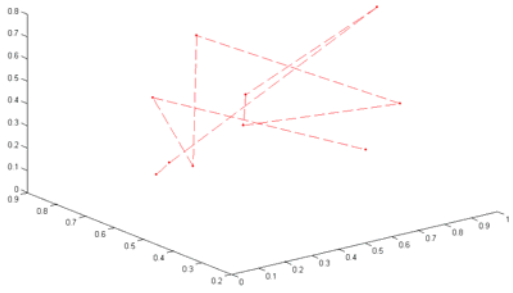
La combinazione meshc,surfc

- **Combina la visualizzazione della superfice o della griglia con il grafico delle curve di livello**
- **Per il resto è analoga a surf o mesh – meshc(x,y,z)**



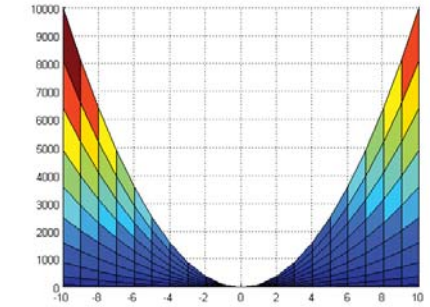
La funzione plot3

- **La funzione plot3 Disegna linee e punti nello spazio 3D, in modo analogo alla plot, ma ora per terne di dati (x,y,z) generiche**
- **Plot3(x,y,z) assume che i 3 vettori abbiano la stessa lunghezza**
- **Ammette le stesse opzioni grafiche di plot.**



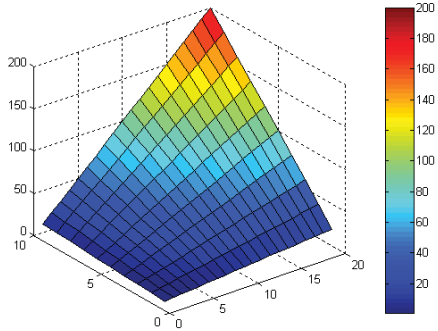
Gestione del punto di vista

- **E' possibile cambiare il punto da cui si osserva un grafico 3D con la funzione view**
  - view(az,el)
  - az = Azimut (rotazione attorno all'asse z)
  - el = Elevazione (rotazione rispetto al piano xy)
- **In modalità interattiva si usa rotate3D**

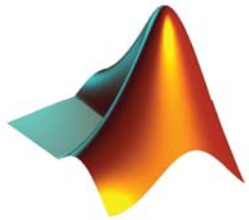


La colorbar

- **Nei grafici 3D è spesso utile affiancare al grafico una barra di colore che indica la corrispondeza tra colori e valori di altezza z, usando la fuznioen colorbar**
- **colorbar('vert') inserisce la colorbar in verticale**
- **colorbar('horiz) inserisce la colorbar in orizzontale**



L'ambiente Matlab
per le applicazioni industriali
(Parte 3 – Applicazioni numeriche)



Fitting di dati

- **In generale il fitting si pone l'obiettivo di adattare un generico modello matematico a dei dati sperimentali**
- **Tramite il modello è possibile**
  - Ricostruire dati mancanti (interpolazione)
  - Stabilire delle tendenze (trend analysis)
  - Fare previsioni (estrapolazione)
- **Nel caso più semplice i dati sono rappresentati da due vettori X (input) e Y (output), ed il modello descrive una relazione fra questi, del tipo Y=f(X), con f sconosciuta**
- **Molto spesso f è un polinomio**

Polinomi

- **Matlab dispone di un metodo particolarmente efficace per rappresentare i polinomi tramite vettori**
- **La variabile P=[a b c] rappresenta il polinomio $Y=aX^2+bX+c$**
- **I polinomi sono particolarmente utili per l'analisi dei dati perché sono un semplice modello relazionale tra X e Y**
- **I problemi di fitting con polinomi sono esprimibili con equazioni molto semplici**
- **Il valore del polinomio P nel punto X si calcola con**

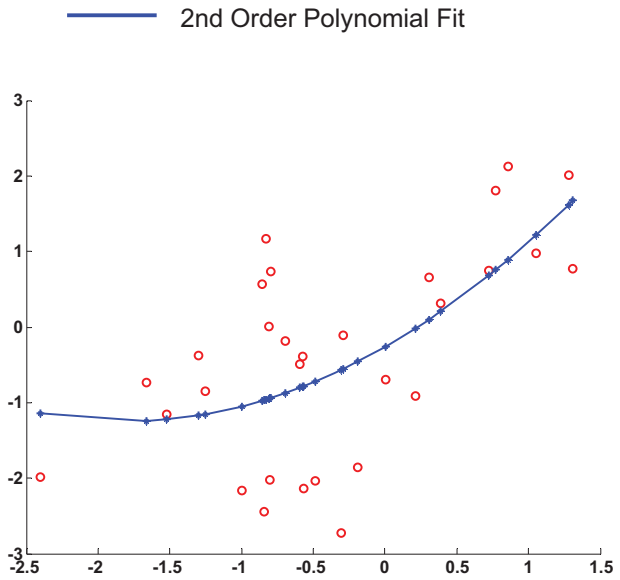
```
Y=polyval(P,X)
```

Radici di un polinomio

- **Un polinomio di grado N ha sempre N radici (complesse)**
- **Un polinomio è quindi individuato univocamente dalle sue radici, cioè i valori per cui si annulla (reali o complessi)**
  - R=root(P) estre le radici del polinomio P
  - P=poly(R) crea un polinomio con radici R
- **Le funzioni root e poly possono quindi essere intese una come l'inversa dell'altra (per radici calcolabili esattamente)**
- **Se N>4 non esistono soluzioni esatte e i dati generati da root sono ottenuti attraverso soluzioni numeriche**

Fitting

- **La tecnica tradizionale di fitting è quella dei minimi quadrati, in cui si cerca un polinomio P di grado N che minimizza la distanza tra tra i dati Y ed i valori P(X)**
- **P = polyfit(X,Y,N)**
- **Se N=1 il fitting è lineare, se N=2 è parabolico, etc. etc.**
- **Il tutto (come sempre accade in Matlab) senza scrivere una riga di codice (o quasi), e senza errori**
- **La finestra di plot fornisce questo ed altri metodi attivabili interattivamente**



Calcolo su funzioni

- **Matlab mette a disposizione molte funzionalità per operare su funzioni definite dall'utente per cercare:**
  - **Zeri, con la funzione fzero**
  - **Minimi o massimi con la funzione fmin**
- **Queste funzioni si appoggiano a librerie di calcolo numerico integrate appositamente in Matlab**
- **La funzione utente (definita in un M-file) viene passata come parametro**
  - X = fzero(@funzione,X0)
  - X = fmin(@funzione,X1,X2)
- **Ammettono entrambe un numero elevato di opzioni che condizionano la qualità e la precisione della ricerca**

Equazioni differenziali

- **Spesso ci si trova di fronte a sistemi il cui modello prevede l'uso di equazioni differenziali (o anche sistemi)**
- **Matlab mette a disposizione soluzioni numeriche di vario livello per integrare direttamente queste equazioni a partire da condizioni iniziali note**
- **Si prevede sempre la riduzione alla forma normale**
  - x' = f(t,x)
- **La funzione incognita x(t) può essere scalare o vettoriale (sistema)**

Equazioni differenziali

- **La sintassi di tutti i metodi (es. ode45) è la stessa:**

- **[t x]=ode45(@funzione,Tint,x0)**
  - [t,x] vettori di uscita per x(t)
  - @funzione punta alla funzione che descrive f
  - Tint è l'intervallo dove calcolare il risultato
  - X0 è la condizione iniziale

- **Il risultato è in una forma pronta per essere graficata con plot**

- **Esistono poi numerose parametrizzazioni definibili tramite il comando odeset('parametro',valore,….)**