Introduzione al Matlab

Felice Iavernaro

Dipartimento di Matematica
Università di Bari
http://dm.uniba.it/∼iavernaro

Ottobre 2011

INTRODUZIONE

MATLAB (MATrix LABoratory) è un software (a pagamento) per il calcolo scientifico, distribuito da Mathworks:

http://www.mathworks.it

INTRODUZIONE

MATLAB (MATrix LABoratory) è un software (a pagamento) per il calcolo scientifico, distribuito da Mathworks:

http://www.mathworks.it

I cofondatori, Jack Little and Cleve Moler, definiscono il MATLAB, *il linguaggio del calcolo tecnico*, ovvero un ambiente di programmazione per lo sviluppo di algoritmi, l'analisi, la visualizzare dei dati e l'esecuzione di calcoli numerici.

Contiene centinaia di funzioni matematiche predefinite, che implementano algoritmi per la risoluzione di diversi problemi nelle varie discipline delle scienze applicate.

Matlab è anche dotato di un linguaggio di programmazione *interpretato* molto elementare e quindi di facile comprensione ed utilizzo anche da parte dei non esperti.

INTRODUZIONE

MATLAB (MATrix LABoratory) è un software (a pagamento) per il calcolo scientifico, distribuito da Mathworks:

http://www.mathworks.it

I cofondatori, Jack Little and Cleve Moler, definiscono il MATLAB, *il linguaggio del calcolo tecnico*, ovvero un ambiente di programmazione per lo sviluppo di algoritmi, l'analisi, la visualizzare dei dati e l'esecuzione di calcoli numerici.

Contiene centinaia di funzioni matematiche predefinite, che implementano algoritmi per la risoluzione di diversi problemi nelle varie discipline delle scienze applicate.

Matlab è anche dotato di un linguaggio di programmazione *interpretato* molto elementare e quindi di facile comprensione ed utilizzo anche da parte dei non esperti.

Cloni gratuiti e open source di Matlab sono: Scilab (www.scilab.org), Octave (http://www.gnu.org/software/octave/)

OPERAZIONI DI BASE

L'utente può digitare i comandi all'interno di un'interfaccia denominata *command window*.
All'avvio del programma compare il prompt

>>

a destra del quale è possibile inserire e gestire espressioni ed istruzioni.

L'oggetto fondamentale su cui opera lo Matlab è la matrice, che esamineremo più avanti, mentre per il momento soffermiamoci sulla possibilità di utilizzare il Matlab per l'esecuzione delle operazioni elementari (come su una calcolatrice scientifica).

OPERAZIONI DI BASE

L'utente può digitare i comandi all'interno di un'interfaccia denominata *command window*.
All'avvio del programma compare il prompt

```
>>
```

a destra del quale è possibile inserire e gestire espressioni ed istruzioni.

L'oggetto fondamentale su cui opera lo Matlab è la matrice, che esamineremo più avanti, mentre per il momento soffermiamoci sulla possibilità di utilizzare il Matlab per l'esecuzione delle operazioni elementari (come su una calcolatrice scientifica).

Gli elementi principali per la costruzione delle espressioni sono:

- i numeri;
- le variabili;
- gli operatori elementari;
- le funzioni.

I numeri

Matlab usa la notazione decimale convenzionale, con un punto per separare la parte intera da quella decimale, ad esempio: $2,\ 1.23, -324.758$

Matlab permette anche la notazione scientifica o esponenziale (con mantissa ed esponente). Per specificare una potenza di 10 si utilizza la lettera e, ad esempio -3×10^8 lo si rappresenta digitando

```
>> -3e8
```

mentre il numero 2.34×10^{-12} lo si rappresenta digitando

```
>> 2.34e-12
```

I numeri

Matlab usa la notazione decimale convenzionale, con un punto per separare la parte intera da quella decimale, ad esempio: $2,\ 1.23, -324.758$

Matlab permette anche la notazione scientifica o esponenziale (con mantissa ed esponente). Per specificare una potenza di 10 si utilizza la lettera e, ad esempio -3×10^8 lo si rappresenta digitando

```
>> -3e8
```

mentre il numero 2.34×10^{-12} lo si rappresenta digitando

```
>> 2.34e-12
```

Il Matlab rappresenta i numeri reali in base binaria utilizzando la *doppia precisione*: ciascun numero è memorizzato in un campo da 64 bit (bit $\equiv$ binary digit, unità di informazione elementare). Ciò corrisponde, in base 10, a circa 16 cifre significative.
Ad esempio il numero 1.2345678901234567890
verrà rappresentato mediante 1.234567890123456.

Le variabili: assegnazione

L'assegnazione è un'operazione utilizzata in informatica per inserire un valore in una variabile. Ad esempio con l'istruzione

```
>> a=12.345
```

si assegna il valore scalare 12.345 alla variabile *a* che, da questo punto in poi, sarà disponibile nel command window per un successivo utilizzo, come ad esempio:

```
>> b=a+1
```

Matlab è *case sensitive*, cioè fa differenza tra variabili scritte in maiuscolo ed in minuscolo:

```
>> A=3.1

>>a+A
 ans  =

    15.445
```

Le variabili: formato di visualizzazione

Attraverso l'istruzione format è possibile modificare il formato di visualizzazione dei risultati ma NON la precisione con cui i calcoli vengono condotti.

Alcune possibilità sono:

- format short, formato *scaled fixed point* con 5 cifre (default);
- format long, formato *scaled fixed point* con 15 cifre;
- format short e, formato *floating point* con 5 cifre (default);
- format long e, formato *floating point* con 15 cifre;

Digitare

```
>>help format
```

per una spiegazione completa di questo comando.

Le variabili: formato di visualizzazione

Attraverso l'istruzione format è possibile modificare il formato di visualizzazione dei risultati ma NON la precisione con cui i calcoli vengono condotti.

Alcune possibilità sono:

- format short, formato *scaled fixed point* con 5 cifre (default);
- format long, formato *scaled fixed point* con 15 cifre;
- format short e, formato *floating point* con 5 cifre (default);
- format long e, formato *floating point* con 15 cifre;

Digitare

```
>>help format
```

per una spiegazione completa di questo comando.

IMPORTANTE: Il Matlab dispone di un help in linea completo. L'*help* può essere utilizzato per apprendere l'uso di qualsiasi funzione, comando o istruzione definiti in ambiente Matlab. Per saperne di più, digitare

```
>>help help
```

ESEMPIO

N.B. Per rendere l'esempio più significativo lavoreremo con un vettore di due elementi. La definizione dei vettori verrà data formalmente in seguito.

```
>> x=[-123.456789012345 0.0009876543210987 65]
x =
 -123.4568    0.0010

>> format short e
>> x
x =
 -1.2346e+002  9.8765e-004

>> format long
>> x
x =
  1.0e+002 *

  -1.23456789012345   0.00000987654321
```

Le variabili: proprietà (1/2)

- Le variabili sono sovrascrivibili, cioè, se digitiamo ora la stringa

  ```
  >> A = 0.5
  ```

  il precedente valore 3.1 viene definitivamente perso.
- Il simbolo ; inserito alla fine di un'assegnazione evita che il risultato della stessa venga visualizzato sul monitor;
- È possibile effettuare più assegnazioni sulla stessa riga, separate dal simbolo , oppure ;

  ```
  >>w1=2.3; w2=-3.5; t=0.12;
  ```
- una variabile può essere una qualsiasi combinazione di caratteri alfanumerici la cui lunghezza massima non superi 24 caratteri. Il nome di una variabile non può iniziare con un numero; inoltre vi sono alcuni caratteri non ammessi, poiché hanno diverso significato (*, +, /, =, ecc.).
- se si esegue un'espressione senza assegnare il risultato ad una variabile, Matlab assegna, per default il risultato alla variabile *ans* (diminutivo di answer).

Le variabili: proprietà (2/2)

Le variabili definite all'interno del command window, risiedono in una zona di memoria chiamata *workspace*.

- Per visualizzare il contenuto di una variabile, si digita la variabile e si preme invio o, alternativamente, mediante *disp(nomevariabile)*
- I comandi *who* e *whos* restituiscono un elenco delle variabili e delle funzioni presenti nel workspace.
- Per cancellare una variabile dal workspace si utilizza il comando *clear*. Esempi:

```
>>clear A
```

```
>>clear x y w1
```

clear da solo, cancella tutte le variabili definite durante la sessione di lavoro corrente.

- il comando *save nomefile*, salva le variabili presenti nel workspace nel file *nomefile*; il comando *load nomefile* carica all'interno del workspace le variabili precedentemente salvate nel file *nomefile*.

Variabili predefinite

In Matlab sono presenti le più comuni costanti matematiche, già preimpostate. Eccone alcune:

- pi: il numero *pi greco* 3.1415927....;
- i,j: l'unità immaginaria $\sqrt{-1}$;
- eps: il minimo valore tale che, per il calcolatore, risulti $(1 + \text{eps}) > 1$. Tale valore in Matlab vale $2^{-52} \simeq 2.22 \times 10^{-16}$. La *precisione di macchina* è eps/2.
- Inf, infinito;
- Nan, not-a-number.
- realmin, il più piccolo numero positivo (in doppia precisione) rappresentabile sul calcolatore.
- realmax, il più grande numero positivo (in doppia precisione) rappresentabile sul calcolatore.

Inoltre, le costanti logiche vero e falso sono rappresentate da:

- 1: vero
- 0: falso

Operazioni elementari

$+$	addizione;
$-$	sottrazione;
$*$	moltiplicazione;
$/$	divisione a destra;
$\backslash$	divisione a sinistra;
$\wedge$	elevamento a potenza;

Esempio:

```
>>2/4
 ans  =
    0.5

>>2\4
 ans  =

    2.
```

Alcune funzioni elementari

Funzioni trigonometriche:

sin	seno
cos	coseno
tan	tangente
asin	arcoseno
acos	arcocoseno
atan	arcotangente

sinh	seno iperbolico
cosh	coseno iperbolico
tanh	tangente iperbolica
asinh	arcoseno iperbolico
acosh	arcocoseno iperbolico
atanh	arcotangente iperbolica

Funzioni esponenziali e logaritmiche:

exp	esponenziale in base e
log2	logaritmo in base 2

log	logaritmo naturale
log10	logaritmo in base 10

Altre funzioni:

abs	valore assoluto o modulo
fix	parte intera

sqrt	radice quadrata
round	arrotondamento

Operatori logici e di relazione

Operatori logici

&	and
|	or
$\sim$	not

Operatori di relazione

$>$	maggiore	$>=$	maggiore o uguale
$<$	minore	$<=$	minore o uguale
$==$	uguale	$\sim=$	diverso

ESEMPIO

```
>>~(2<=1)
 ans  =

   1
```

ELEMENTI DI PROGRAMMAZIONE STRUTTURATA

Matlab mette a disposizione un linguaggio di programmazione di semplice utilizzo.

Esso dispone delle tre strutture fondamentali

- sequenza
- selezione
- iterazione

che consentono di tradurre in programma un qualsiasi algoritmo.

La *sequenza* non è altro che un blocco ordinato di istruzioni che verranno eseguite una dopo l'altra in successione.

Selezione

La selezione, nella sua forma più semplice, consente di eseguire un blocco di istruzioni a seconda che sia verificata o meno una data condizione.

if condizione
    istruzioni
end

ESEMPIO: calcolo del valore assoluto di un numero reale x.

```
if x<0
  x=-x;
end
```

È possibile inserire queste istruzioni direttamente nel command window elencandole su un unica riga e separandole dalla virgola o dal punto e virgola:

```
if x<0, x=-x; end
```

Alternativamente è possibile memorizzare le istruzioni all'interno di una *function* (questa possibilità verrà considerata la prossima lezione)

Selezione

L'uso completo del costrutto **if** è:

if prima condizione
istruzioni
elseif seconda condizione
istruzioni
⋮
else
istruzioni
end

ESEMPIO: calcolo del segno di un numero reale x.

```
 if x<0
   s=-1;
elseif x==0
   s=0;
else
   s=1;
end
```

```
Dal command window:
if x<0, s=-1; elseif x==0, s=0; else, s=1; end
```

Selezione

In luogo dell' **if**, in alcuni casi è possibile usare il costrutto **switch**. Ad esempio ciò risulta conveniente quando l'argomento assume dei valori prestabiliti.

switch variabile
case valore1
    istruzioni
case valore2
    istruzioni
        ⋮
otehrwise
    istruzioni
end

(l'uso di **otehrwise** è opzionale).

ESERCIZIO: Consideriamo $f(x) = \begin{cases} \frac{\sin x}{x}, & \text{se } x \neq 0, \\ 1, & \text{se } x = 0. \end{cases}$

Scrivere il codice Matlab che, dato il valore di x calcola $y = f(x)$.

Ripetizione

La ripetizione o iterazione permette l'esecuzione di un blocco di istruzioni un numero di volte prestabilito.

for ind = val1 : step : val2
istruzioni
end

che esegue in maniera ripetuta il gruppo di istruzioni interne al ciclo per un numero di volte prestabilito, uguale al numero di volte in cui varia l'indice *ind* (detto contatore) fra il valore *val*1 e il valore *val*2 con un incremento pari a *step*. Se l'incremento *step* non è specificato

esplicitamente (cioè se il ciclo è del tipo *for ind* = *val*1 : *val*2) esso viene scelto per default uguale a $+1$.
ESEMPIO: calcolo della somma dei primi dieci numeri interi.

```
somma=0; for i=1:10
  somma=somma+i;
end
```

Ripetizione

Un altro costrutto che implementa l'iterazione è il seguente:

while condizione
istruzioni
end

che esegue in maniera ripetuta il gruppo di istruzioni interne al ciclo fino a quando la condizione resta verificata. Ovviamente, qualora la condizione risulti non verificata in partenza, Matlab salta tutto il blocco delle istruzioni, passando alla linea successiva all'*end*.
ESEMPIO: quanti numeri interi successivi occorre sommare per ottenere un numero maggiore di 354?

```
somma=0; i=0; while somma<=354
    i=i+1;
    somma=somma+i;
end disp(i)
```

ES. 2. Calcolo del M.C.D. tra due numeri. (1/2)

Siano m ed n due numeri positivi ed MCD(m, n) il loro Massimo Comune Divisore. Vale la seguente proprietà:

$$\text{MCD}(m,n) = \begin{cases} \text{MCD}(m-n,n), & \text{se } m > n, \\ \text{MCD}(m,n-m), & \text{se } n > m, \\ m, & \text{se } n = m. \end{cases} \qquad (*)$$

Infatti, supponendo ad esempio $m > n$, vale evidentemente la seguente equivalenza (d, p, q, r sono interi positivi):

$$\begin{cases} m = d\,p \\ n = d\,q \end{cases} \iff \begin{cases} m - n = d\,(p-q) \equiv d\,r \\ n = d\,q \end{cases}$$

Cioè: tutti e soli i divisori comuni di m ed n sono i divisori comuni di $m - n$ ed n, e quindi anche il massimo comune divisore di queste due coppie di numeri dovrà coincidere. I vantaggi della (*) sono:

- la (*) non fa uso della scomposizione in fattori primi dei due interi m ed n;
- la (*) induce un semplice algoritmo implementabile in Matlab.

ES. 2. Calcolo del M.C.D. tra due numeri. (2/2)

ESEMPIO:

$\mathrm{MCD}(30, 18) = \mathrm{MCD}(12, 18) = \mathrm{MCD}(12, 6) = \mathrm{MCD}(6, 6) = 6.$

ALGORITMO:

ES. 2. Calcolo del M.C.D. tra due numeri. (2/2)

ESEMPIO:

$\mathrm{MCD}(30, 18) = \mathrm{MCD}(12, 18) = \mathrm{MCD}(12, 6) = \mathrm{MCD}(6, 6) = 6.$

ALGORITMO:

```
function m=mcd(m,n)
% calcola il M.C.D. tra m ed n
while m~=n
   if m>n
      m=m-n;
   else
      n=n-m;
   end
end
```

Funzioni: motivazione

Consideriamo le righe di codice che calcolano le radici di un'equazione di secondo grado della forma $ax^2 + bx + c = 0$:

```
>>a=1;b=-5;c=6;
>>delta=b^2-4*a*c;
>>x1=(-b-sqrt(delta))/(2*a); x2=(-b+sqrt(delta))/(2*a);
```

Se questa sequenza di istruzioni deve essere ripetuta più volte:

- in corrispondenza di diversi valori dei coefficienti a, b, c,
- ovvero rispetto ad una diversa terna di variabili (ad es. se vogliamo risolvere l'equazione $px^2 + qx + r = 0$),

saremmo costretti, ogni volta, a rieseguire ciascuna delle istruzioni (si pensi che gli algoritmi più sofisticati si traducono in centinaia di righe di codice).

Converrà, in tal caso, riscrivere le stesse istruzioni all'interno di un file che, una volta salvato sul disco rigido, potrà essere richiamato ed eseguito.

Funzioni: definizione

I file che ci interesserà compilare sono chiamati *function file* o più semplicemente *funzioni*.

Una funzione Matlab ammette dei dati (o parametri) di input (ingresso) e restituisce dei dati di output (uscita).

Informalmente possiamo pensare che i parametri di input rappresentino i dati del problema che vogliamo risolvere, mentre i parametri di output rappresentino le soluzioni del nostro problema.

Ad esempio, nel caso dell'equazione di secondo grado, i parametri di input saranno i coefficienti dell'equazione, mentre i parametri di output saranno le radici della stessa.

Cominciamo con un esempio: vediamo come scrivere la funzione che calcola le radici di un'equazione di secondo grado.

Funzioni: esempio

```
function [x1,x2,delta]=eq2(a,b,c)
% calcola le radici di un'equazione di secondo grado
% SINTASSI: [x1,x2]=eq2(a,b,c)
% Dati di input
%              a,b,c: coefficienti di ax^2+bx+c=0
% Dati di output
%              x1,x2: radici dell' equazione

delta=b^2-4*a*c;

x1=(-b-sqrt(delta))/(2*a);
x2=(-b+sqrt(delta))/(2*a);
```

Funzioni: sintassi

Una funzione Matlab comincia con la riga:

```
function [c,d,...] = nome_funzione(a,b,...)
```

Riconosciamo:

- la parola chiave function;
- il nome della funzione;
- le variabili di input a,b,...;
- le variabili di output c,d,...;

Tutte le variabili usate all'interno di una function sono variabili locali, cioè esistono solo durante l'esecuzione della funzione e non modificano il workspace. Ad esempio, la variabile *a* usata all'interno di una function sarà diversa dalla variabile *a* usata in un'altra function o nel command window.

Funzioni: come scriverle, come eseguirle.

Matlab dispone di un editor di testo che si apre dalle icone della barra degli strumenti nel command window.

Una volta scritta la funzione, la si salva in un file, mediante la barra degli strumenti dell'editor. È buona consuetudine assegnare al file lo stesso nome della funzione. Ogni function file dovrà avere l'estensione .m per essere riconosciuto da Matlab. Ad esempio, alla funzione dell'esempio precedente assegneremo il nome *eq2.m*

A questo punto siamo pronti per eseguire la funzione:

```
>>[x1,x2]=eq2(1,-5,6)
 x1  =
    2.
 x2  =
    3.
```

Osservazione: è possibile richiedere in output solo una parte dei parametri in output definiti all'interno della funzione.

Vettori (1/2)

Un vettore è un insieme ordinato di numeri, della forma:

- $x = [x_1,\ x_2, \ldots, x_n]$ (vettore riga), oppure
- $\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ (vettore colonna)

Ad esempio, dal prompt di Matlab, digitando:

```
>>x=[2 1 0 -3]      oppure
>>x=[2, 1, 0, -3]
```

si assegna alla variabile x il vettore di 4 elementi: $x = [2, 1, 0, -3]$. Invece digitando

```
>>x=[2; 1; 0; -3]      oppure
>>x=[2 1 0 -3]'
```

si ottiene l'analogo vettore colonna.

Vettori (2/2)

Il numero di elementi di un vettore si chiama *lunghezza* del vettore: in Matlab si ottiene mediante la function *length*:

```
>>length(x)
 ans  =

    4.
```

Per accedere, ad esempio, al secondo elemento del vettore x, scriveremo:

```
>>x(2)
 ans  =

    1.
```

- Cosa succede se scriviamo $x(5)$?
- Cosa succede se scriviamo $x(5) = 1$?
- Cosa succede se scriviamo $x(8) = -4$?

Esercizi

Definiamo il vettore $x = [1.5,\ -0.2,\ -3.1,\ 2.6]$ Se vogliamo calcolare la somma degli elementi del vettore x, ed assegnare il risultato alla variabile s potremo scrivere:

```
>>x=[1.5, -0.2, -3.1,2.6];
>>s=0;for i=1:4,s=s+x(i);end,disp(s)

    0.8
```

ESERCIZIO: il libretto di uno studente riporta i seguenti voti:
26, 24, 28, 30, 27, 18, 26, 30, 29. Calcolarne la media aritmetica e la media geometrica.
ESERCIZIO: Scrivere una function Matlab che calcola la varianza degli elementi di un vettore x. Si ricordi che:

$$\mathrm{var}(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2, \qquad \text{dove } \bar{x} \text{ è il valor medio degli elementi di } x$$

Esercizi

ESERCIZIO: Scrivere una function Matlab che ha in input due vettori reali x ed y di uguale lunghezza n ed in output il loro prodotto scalare

$$\mathbf{x}^T \mathbf{y} = \sum_{i=1}^{n} x_i \, y_i .$$

ESERCIZIO: Scrivere una function Matlab che ha in input un vettore **x** ed in output il minimo elemento del vettore e l'indice della componente corrispondente.

ESERCIZIO: Scrivere una function Matlab che ha in input un vettore **x** ed in output il massimo elemento del vettore e l'indice della componente corrispondente.

ESERCIZIO: Scrivere una function Matlab che ha in input un vettore **x** ed in output ed in output il vettore che si ottiene da **x** ordinando le sue componenti in senso crescente.

Definizione di matrice reale

Definizione

Dati due interi positivi m ed n, una matrice A reale $m \times n$ è un array bidimensionale avente m righe ed n colonne così definito

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & \cdots & a_{2n} \\ \vdots & \vdots & & & \vdots \\ a_{m1} & a_{m2} & \cdots & \cdots & a_{mn} \end{pmatrix}$$

Più formalmente possiamo dire che una matrice è un'applicazione

$$A : \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\} \longrightarrow \mathbb{R}$$

tale che $A(i, j) = a_{ij} \in \mathbb{R}$.

Notazioni

Una matrice verrà solitamente denotata con le lettere maiuscole dell'alfabeto, mentre gli elementi di una matrice con le lettere minuscole; ad esempio la scrittura

$$A = \{a_{ij}\}_{\substack{i=1,\dots m \\ j=1,\dots n}}, \text{ ovvero } \quad A = \{a_{ij}\}, \quad i = 1, \dots m, \quad j = 1, \dots n,$$

denoterà una matrice ad m righe ed n colonne il cui generico elemento è a_{ij}. Denotiamo con $\mathbb{R}^{m\times n}$ l'insieme delle matrici con m righe ed n colonne.

Esempio ($A \in \mathbb{R}^{3\times 4}$)

$$A = \begin{pmatrix} 2 & 0 & -1 & \sqrt{3} \\ \pi & \log(3) & -1/3 & 1 \\ 2/3 & 0 & \sin(\pi/7) & 4/3 \end{pmatrix},$$

è una matrice 3×4*, ovvero a* 3 *righe e* 4 *colonne.*

Definire una matrice in Matlab

In Matlab, una matrice può essere definita elencando, tra parentesi quadrate, le sue righe. Gli elementi su una stessa riga vanno separati da una virgola o da uno spazio, mentre per passare da una riga alla successiva si usa il punto e virgola. Ad esempio:

```
>>A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
A =

    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

In questo esempio A è una matrice quadrata di dimensione 4.

Matrici particolari (1/4)

- Se $m = n$, (num. di righe= num. di colonne), la matrice è detta quadrata di dimensione n (se $m \neq n$, la matrice è detta rettangolare);
- se $m = n = 1$, la matrice si riduce ad un unico elemento e dunque coincide con uno scalare: $A = (a_{11})$;
- se $m = 1$, la matrice possiede un'unica riga, pertanto si riduce ad un vettore riga: $A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \end{pmatrix}$
- se $n = 1$, la matrice possiede un'unica colonna, pertanto si riduce ad un vettore colonna:

$$A = \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{pmatrix} \equiv \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \end{pmatrix}^T.$$

Matrici particolari (2/4)

- La matrice $m \times n$ i cui elementi sono tutti nulli si chiama matrice nulla e si denota con $0_{m \times n}$ o più semplicemente con 0.

  Per ottenere una matrice nulla in Matlab, anziché elencare i suoi elementi, si può utilizzare la function predefinita *zeros*:

```
>>zeros(2,3)
 ans  =

    0     0     0
    0     0     0
```

  Se invece usiamo la function *ones*:

```
>>ones(2,3)
 ans  =

    1     1     1
    1     1     1
```

Matrici particolari (3/4)

- Si chiama matrice identica, ogni matrice quadrata avente elementi diagonali uguali ad 1 ed elementi extra-diagonali nulli:

$$I = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \end{pmatrix}$$

Per ottenere una matrice identica in Matlab si usa la function predefinita *eye*:

```
>>eye(4,4)
 ans  =

    1     0     0     0
    0     1     0     0
    0     0     1     0
    0     0     0     1
```

Matrici quadrate particolari (4/4)

- Una matrice quadrata A è detta:
  - diagonale se tutti i suoi elementi extra-diagonali sono nulli: $a_{ij} = 0,\ \forall i, j = 1, \ldots n, \quad i \neq j$;
  - triangolare inferiore se tutti i suoi elementi al di sopra della diagonale principale sono nulli: $a_{ij} = 0,\ \forall i < j$;
  - triangolare superiore se tutti i suoi elementi al di sotto della diagonale principale sono nulli: $a_{ij} = 0,\ \forall i > j$;
- Esempio:

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 3 \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 0 & 0 \\ -1 & -2 & 0 \\ 2 & -4 & 3 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & -2 \\ 0 & 0 & 3 \end{pmatrix}.$$

Addizione tra matrici (1/2)

Definizione

Se $A = \{a_{ij}\}$ *e* $B = \{b_{ij}\}$ *sono matrici* $m \times n$ *si definisce somma tra* A *e* B *la matrice*

$$A + B = \{a_{ij} + b_{ij}\} \in \mathbb{R}^{m \times n}$$

Esempio

$$A = \begin{pmatrix} 1 & 2 & 3 \\ -1 & 0 & 1 \\ 2/3 & 1 & -2 \end{pmatrix}, \; B = \begin{pmatrix} 1 & 0 & 1 \\ -1 & -2 & 0 \\ 1/3 & -4 & 3 \end{pmatrix}, \; A + B = \begin{pmatrix} 2 & 2 & 4 \\ -2 & -2 & 1 \\ 1 & -3 & 1 \end{pmatrix}.$$

Addizione tra matrici (2/2): Esempio Matlab

```
>>A=[1 2 3;-1 0 1]
 A  =
    1    2    3
  - 1    0    1

>>B=[1 0 1; -1 -2 0]
 B  =
    1     0    1
  - 1   - 2    0

>>C=A+B
 C  =
    2    2    4
  - 2  - 2    1
```

Moltiplicazione di uno scalare per una matrice (1/2)

Definizione

Se $A = \{a_{ij}\} \in \mathbb{R}^{m\times n}$ *e* $\lambda \in \mathbb{R}$, *si definisce prodotto di* λ *per* A *la matrice*

$$\lambda \cdot A = \{\lambda a_{ij}\} \in \mathbb{R}^{m\times n}$$

Esempio

$$A = \begin{pmatrix} 1 & 2 & 3 \\ -1 & 0 & 1 \\ 2/3 & 1 & -2 \end{pmatrix}, \quad 2 \cdot A = \begin{pmatrix} 1 & 4 & 6 \\ -2 & 0 & 2 \\ 4/3 & 2 & -4 \end{pmatrix}.$$

Moltiplicazione di uno scalare per una matrice (2/2): Esempio Matlab

```
>>A=[1 2 3;-1 0 1]
 A  =
    1    2    3
  - 1    0    1

>>lambda=2.5
 lambda  =

    2.5

>>lambda*A
 ans  =
    2.5    5    7.5
  - 2.5    0    2.5
```

Trasposta di una matrice (1/2)

Se $A \in \mathbb{R}^{m\times n}$, la trasposta di A, denotata con A^T, è la matrice ottenuta da A scambiando le righe con le colonne (o viceversa), ovvero

$$A^T = \{a_{ji}\}, \qquad i = 1, \ldots m,\ j = 1, \ldots n.$$

Pertanto $A^T \in \mathbb{R}^{n\times m}$.

Esempio

$$A = \begin{pmatrix} 1 & 2 & 3 \\ -1 & 0 & 1 \end{pmatrix} \Longrightarrow A^T = \begin{pmatrix} 1 & -1 \\ 2 & 0 \\ 3 & 1 \end{pmatrix}.$$

Trasposta di una matrice (1/2): Esempio Matlab

In Matlab per ottenere il trasposto si usa l'apice:

```
>>A=[1 2 3;-1 0 1]
 A  =

    1    2    3
   -1    0    1

>>B=A'
 B  =

    1   -1
    2    0
    3    1
```

Prodotto di matrici (righe per colonne)

- Ricordiamo che se $\mathbf{a}$ e $\mathbf{b}$ sono due vettori (colonna) di lunghezza n, il prodotto scalare di $\mathbf{a}$ e $\mathbf{b}$ denotato con $\mathbf{a}^T\,\mathbf{b}$ è così definito:

$$\mathbf{a}^T\,\mathbf{b} \equiv \sum_{k=1}^{n} a_k\, b_k.$$

- Siano $A \in \mathbb{R}^{m\times p}$ e $B \in \mathbb{R}^{p\times n}$. Si definisce prodotto (righe per colonne) tra A e B la matrice $C = A \cdot B \in \mathbb{R}^{m\times n}$ il cui elemento generico c_{ij} è il prodotto scalare tra la riga i-esima di A e la colonna j-esima di B:

$$c_{ij} = \mathbf{a}_i^T\,\mathbf{b}_j = \sum_{k=1}^{p} a_{ik}\, b_{kj}, \quad i = 1, \ldots m, \quad j = 1, \ldots, n.$$

$\mathbf{a}_i^T \rightarrow$ i-esima riga di A; $\quad \mathbf{b}_j \rightarrow$ j-esima colonna di B.

ESEMPIO

Osservazione

Il prodotto tra due matrici è possibile solo se il numero di colonne del primo fattore coincide con il numero di righe del secondo fattore.

$$A = \left(\begin{array}{cccc} 1 & 2 & 3 & 4 \\ \hline -1 & 0 & 1 & 2 \end{array}\right) = \begin{pmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \end{pmatrix}, \quad B = \left(\begin{array}{c|c|c} 1 & -1 & 3 \\ -1 & 0 & 1 \\ 0 & 2 & -1 \\ 2 & 1 & -2 \end{array}\right) = \begin{pmatrix} \mathbf{b}_1, & \mathbf{b}_2, & \mathbf{b}_3 \end{pmatrix}$$

$$A \cdot B = \begin{pmatrix} \mathbf{a}_1^T \mathbf{b}_1 & \mathbf{a}_1^T \mathbf{b}_2 & \mathbf{a}_1^T \mathbf{b}_3 \\ \mathbf{a}_2^T \mathbf{b}_1 & \mathbf{a}_2^T \mathbf{b}_2 & \mathbf{a}_2^T \mathbf{b}_3 \end{pmatrix} = \begin{pmatrix} 7 & 9 & -6 \\ 3 & 5 & -8 \end{pmatrix}$$

$B \cdot A$ non è possibile.

Ulteriori esempi (1/2)

- $\left(\begin{array}{ccc} 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline 7 & 8 & 9 \end{array}\right) \cdot \begin{pmatrix} -1 \\ 1 \\ -2 \end{pmatrix} = \begin{pmatrix} -5 \\ -11 \\ -17 \end{pmatrix}$
- $\begin{pmatrix} -1 & 1 & -2 \end{pmatrix} \cdot \left(\begin{array}{c|c|c} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array}\right) = \begin{pmatrix} -11 & -13 & -15 \end{pmatrix}$
-

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}^2 = \left(\begin{array}{ccc} 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline 7 & 8 & 9 \end{array}\right) \cdot \left(\begin{array}{c|c|c} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array}\right) = \begin{pmatrix} 30 & 36 & 42 \\ 66 & 81 & 96 \\ 102 & 126 & 150 \end{pmatrix}$$

Ulteriori esempi (2/2)

- $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 0 & -1 \\ -2 & 1 \end{pmatrix} = \begin{pmatrix} -4 & 1 \\ -8 & 1 \end{pmatrix}$
- $\begin{pmatrix} 0 & -1 \\ -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} -3 & -4 \\ 1 & 0 \end{pmatrix}$

Osservazione

Dunque, se A e B sono quadrate dello stesso ordine, $A \cdot B$ e $B \cdot A$ sono ben definite, tuttavia, in generale A e B non sono permutabili cioè, in generale, $A \cdot B \neq B \cdot A$.

Ne segue che la moltiplicazione tra matrici non è commutativa.

Prodotto tra matrici: Esempio Matlab

```
>>A=[1 2 3; 4 5 6]
 A  =
    1    2    3
    4    5    6

>>B=[1 2 1 -1;1 -1 0 1;0 1 -2 1]
 B  =
    1    2    1  - 1
    1  - 1    0    1
    0    1  - 2    1

>>C=A*B
 C  =
    3    3  - 5    4
    9    9  - 8    7
```

La funzione size

- **size**.
  L'istruzione $size(A)$ applicata alla matrice A di dimensioni $m \times n$ restituisce il vettore riga di due elementi $[m, n]$ contenente il numero m di righe e il numero n di colonne della matrice A. In Matlab:

```
>>A=[1 2 3;4 5 6];

>>[m,n]=size(A)
 n  =

    3
 m  =

    2
```

Esercizi (1/2)

ESERCIZIO: Scrivere una function Matlab che ha in input una matrice A ed in output il suo massimo elemento e gli indici (di riga e colonna) corrispondenti. Confrontare il risultato con quello ottenuto mediante la function predefinita *max*.

ESERCIZIO: Scrivere una function Matlab che ha in input una matrice A ed in output il suo minimo elemento e gli indici (di riga e colonna) corrispondenti. Confrontare il risultato con quello ottenuto mediante la function predefinita *min*.

Esercizi (2/2)

ESERCIZIO: Scrivere una function Matlab che ha in input una matrice A ed un vettore $\mathbf{x}$ ed in output il vettore $\mathbf{y} = A\mathbf{x}$:

$$y(i) = \sum_{j=1}^{n} A(i,j)x(j), \qquad i = 1, \ldots, m$$

essendo A di dimensioni $m \times n$ ed $\mathbf{x}$ di lunghezza n.

ESERCIZIO: Scrivere una function Matlab che ha in input due matrici A e B ed in output la matrice prodotto $C = A \cdot B$:

$$C(i,j) = \sum_{k=1}^{p} A(i,k)B(k,j), \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

essendo A di dimensioni $m \times p$ e B di dimensioni $p \times n$.

L'operatore ":" (1/3)

L'operatore ":" è uno dei più importanti di Matlab. Analizziamo alcuni dei suoi molteplici usi.

- Se $n1 \in \mathbb{N}$ ed $n2 \in \mathbb{N}$, con $n1 < n2$, mediante l'espressione $n1 : n2$ si ottiene un vettore riga che contiene tutti i numeri interi compresi tra $n1$ e $n2$. ESEMPI in Matlab:

```
>>1:10
 ans  =
    1    2    3    4    5    6    7    8    9    10

>>2:2
 ans  =
    2

>>10:1
 ans  =
     []
```

L'operatore ":" (2/3)

- Più in generale vale la seguente regola. Se $a \in \mathbb{R}$, $b \in \mathbb{R}$ e $h \in \mathbb{R}$, l'istruzione $a : h : b$ restituisce un vettore riga i cui elementi sono

$$a,\ a+h,\ a+2h,\ \ldots\ ,a+mh$$

dove m è un numero intero tale che

$$a+mh \leq b \qquad e \qquad a+(m+1)h > b.$$

Questo significa che gli elementi del vettore di output vanno da a a b con incremento h, arrestandosi al numero che non supera b.
L'incremento h può essere un numero reale positivo o negativo.

L'operatore ":" (3/3) ESEMPI in Matlab

```
>>10:2:20
 ans  =
    10    12    14    16    18    20

>>10:3:21
 ans  =
    10    13    16    19

>>100:-5:78
 ans  =
   100    95    90    85    80

>>0:.1:pi/4
 ans  =
    0   0.1   0.2   0.3   0.4   0.5   0.6   0.7
```

Estrazione di sottomatrici (1/2)

```
>>A=[1 2 3 4;5 6 7 8; 9 10 11 12]
 A  =

    1    2     3     4
    5    6     7     8
    9    10    11    12

>>A(2,:) %seconda riga di A
 ans  =

    5    6    7    8

>>A(:,3) %terza colonna di A
 ans  =

    3
    7
    11
```

Estrazione di sottomatrici (2/2)

```
>>A(1:2,1:2) %sottomatrice principale di testa di dimensione 2
 ans  =

    1    2
    5    6

>>A(2:3,3:4)
 ans  =

    7     8
    11    12

>>A([1 3],[2 4])
 ans  =

    2     4
    10    12
```

Costruzione di matrici a blocchi (1/2)

```
>>A11=[1 2; 3 4]
 A11  =

    1    2
    3    4

>>A12=[-2 3]'
 A12  =

  - 2
    3

>>A21=[1 0]
 A21  =

    1    0

>>A22=5
 A22  =

    5

>>A=[A11 A12; A21 A22]
 A  =

    1    2  - 2
    3    4    3
    1    0    5
```

Costruzione di matrici a blocchi (2/2)

```
>>A=[1 2 3;4 5 6]
 A  =

    1    2    3
    4    5    6

>>A=[A;[7 8 9]] % aggiunta di una nuova riga
 A  =

    1    2    3
    4    5    6
    7    8    9

>>A=[A zeros(3,1)] % aggiunta di una nuova colonna
 A  =

    1    2    3    0
    4    5    6    0
    7    8    9    0
```

Esercizio: Trasformare la matrice A di dimensioni 3×4 sopra riportata, in una matrice 4×4, inserendo come terza riga il seguente vettore: $x = [-1,\ 2,\ -3,\ 2]$.

La funzione *linspace*

Permette di ottenere lo stesso risultato raggiunto con l'operatore ":" , prefissando però il numero di punti anziché il passo.

La funzione *linspace* serve per costruire un vettore di punti equidistanti: mediante $linspace(x1, x2)$ si ottiene un vettore riga di 100 punti equidistanti compresi tra $x1$ e $x2$, mentre con $linspace(x1, x2, n)$ si ottiene un vettore riga di n elementi equidistanti compresi tra $x1$ e $x2$. Esempio

```
>>linspace(0,1,11)
 ans  =

    0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1

>>linspace(0,pi,8)
 ans  =
    0  0.448  0.897  1.346  1.795  2.243  2.692  3.141
```

Le operazioni .∗, ./ e .^

Anziché effettuare la moltiplicazione nel senso righe per colonne tra due matrici (o vettori), l'operazione ".∗" effettua la moltiplicazione *elemento per elemento* restituendo una matrice i cui elementi sono il prodotto degli elementi omonimi dei due fattori.

Ad esempio, considerati $\mathbf{x} = [x_1,\ x_2,\ x_3]$ ed $\mathbf{y} = [y_1,\ y_2,\ y_3]$, avremo:

$$\mathbf{x}. * \mathbf{y} = [x_1 y_1,\ x_2 y_2,\ x_3 y_3]$$

Analogamente, avremo:

$$\mathbf{x}./\mathbf{y} = [x_1/y_1,\ x_2/y_2,\ x_3/y_3]$$

e

$$\mathbf{x}.^{\wedge}\mathbf{y} = [x_1^{y_1},\ x_2^{y_2},\ x_3^{y_3}]$$

ESEMPIO in Matlab

```
>>A=[1 2 3;4 5 6]
 A  =
    1    2    3
    4    5    6
>>B=[-2 4 2; -1 3 -2]
 B  =
  - 2    4    2
  - 1    3  - 2

>>A.*B
 ans  =
  - 2    8     6
  - 4    15  - 12
>>A./B
 ans  =
   -0.5000    0.5000    1.5000
   -4.0000    1.6667   -3.0000
```

Tabulare una funzione (1/2)

Consideriamo una funzione reale di variabile reale $y = f(x)$. Sia $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ un vettore di elementi appartenenti al dominio di f. Vogliamo costruire il vettore delle valutazioni di f, cioè

$$\mathbf{y} = [f(x_1),\ f(x_2), \ldots, f(x_n)].$$

Esempio: definiamo in Matlab il vettore

```
>>x=linspace(0,pi,5)
 x  =
    0    0.78540    1.5708    2.35619    3.14159
```

e corrispondentemente valutiamo le seguenti funzioni.

- $y = \sin(x)$:

```
>>y=sin(x)
 y  =
    0    0.70711    1    0.70711    1.2D-16
```

Tabulare una funzione: esempi (2/2)

- $y = \sin(x)\cos(x)$:

```
>>y2=sin(x).*cos(x)
 y2  =
    0    0.5    6.1D-17  - 0.5  - 1.2D-16
```

- $y = x^2\, e^{-x}$:

```
>>y=(x.^2).*exp(-x)
 y  =
    0    0.28125    0.51292    0.52619    0.42650
```

- $y = \dfrac{x}{\cos(x)}$:

```
>>y=x./cos(x)
 y  =
    0    1.11072    2.6D+16  - 3.33216  - 3.14159
```

Grafico di una funzione: esempio Matlab (1/2)

Si voglia rappresentare il grafico della funzione $y = \sin(x)e^{-x}$ nell'intervallo $[0, 2\pi]$. Le righe di codice:

```
>>x=linspace(0,2*pi,100);
>>y=sin(x).*exp(-x);
>>plot(x,y)
```

producono il grafico

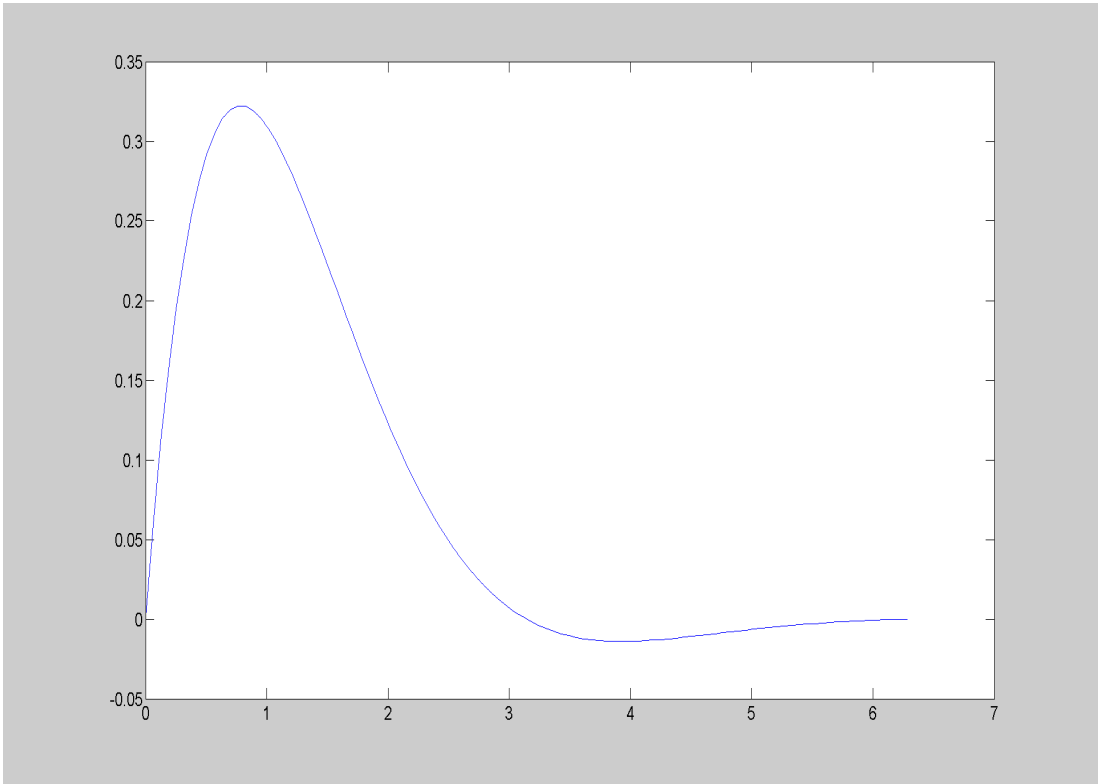


Grafico di una funzione: esempio Matlab (2/2)

Si vogliano rappresentare sugli stessi assi i grafici delle funzioni $y = \sin(x)e^{-x}$, $y = \sin(3x)e^{-x}$, $y = \sin(5x)e^{-x}$. Le righe di codice:

```
>>x=linspace(0,2*pi,100);
>>y=sin(x).*exp(-x);
>>y1=sin(3*x).*exp(-x);
>>y2=sin(5*x).*exp(-x);
>>plot(x,y,x,y1,x,y2)
>>legend('y=sin(x)*exp(-x)','y=sin(3*x)*exp(-x)','y=sin(5*x)*exp(-x)')
```

producono il grafico

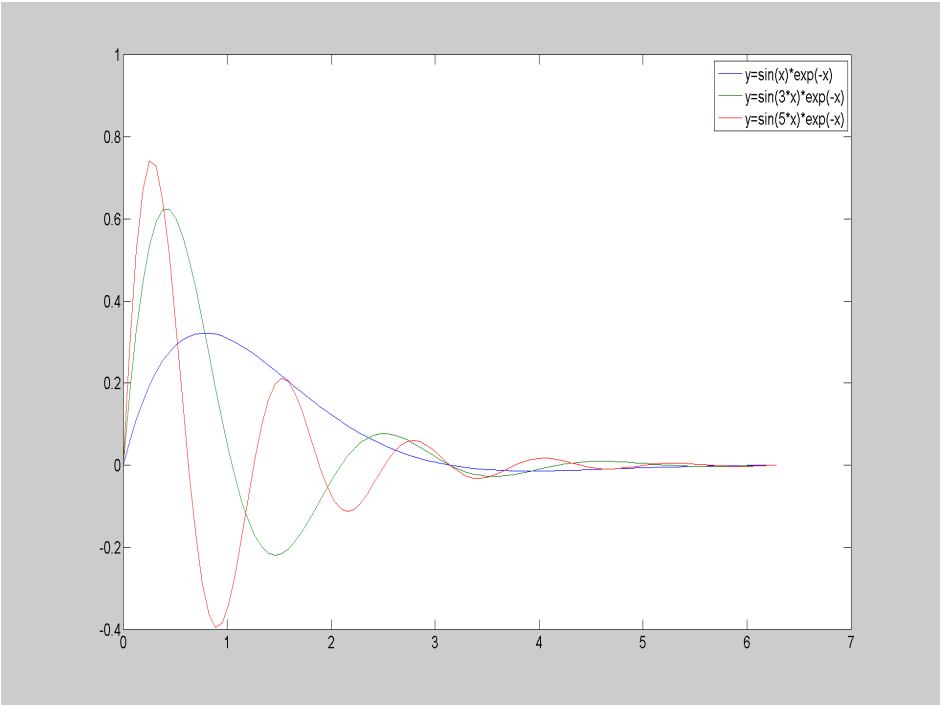


Grafico di una funzioni a due variabili (1/2)

Si voglia rappresentare il grafico della funzione di due variabili $f(x, y)$ su un dominio rettangolare $D = [a, b] \times [c, d]$. Occorre preliminarmente generare una griglia di punti che ricopra il dominio. A tal fine si utilizza la funzione `meshgrid`.

Esempio: Supponiamo di considerare due vettori $x = [x_1, x_2, x_3]$, $y = [y_1, y_2, y_3, y_4]$. Allora $[X, Y] = meshgrid(x, y)$ genera due matrici X e Y di dimensione 4×3 tale che:

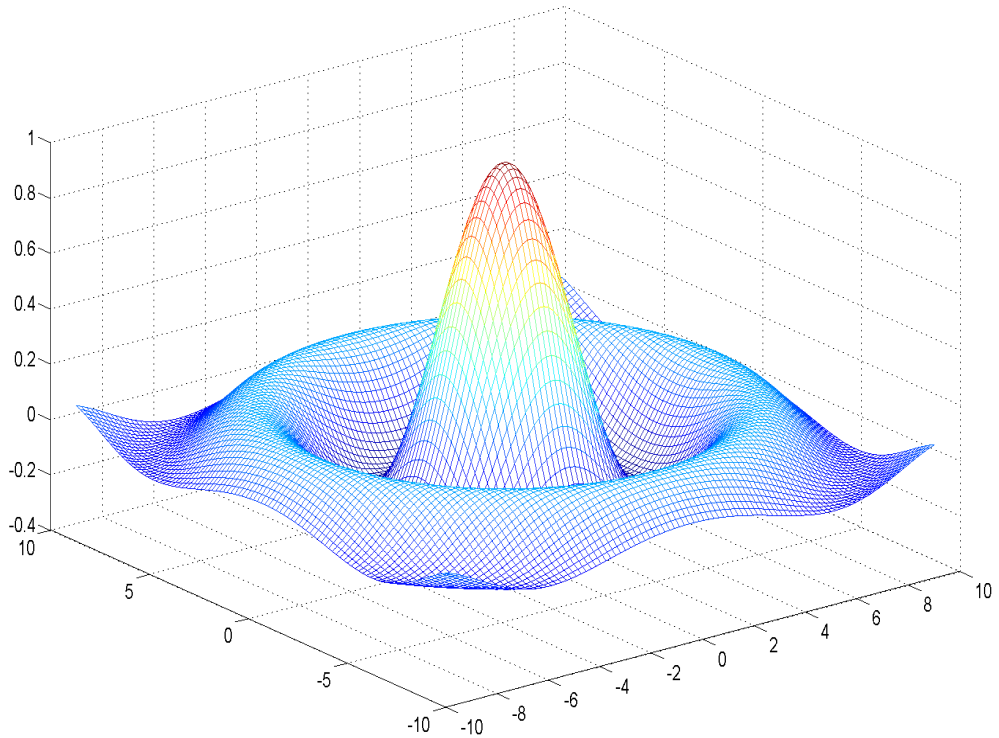
$$X = \begin{pmatrix} x_1 & x_2 & x_3 \\ x_1 & x_2 & x_3 \\ x_1 & x_2 & x_3 \\ x_1 & x_2 & x_3 \end{pmatrix}, \qquad Y = \begin{pmatrix} y_1 & y_1 & y_1 \\ y_2 & y_2 & y_2 \\ y_3 & y_3 & y_3 \\ y_4 & y_4 & y_4 \end{pmatrix}$$

In questo modo, si individua una griglia definita dai punti nel piano $(X(i,j), Y(i,j))$, al variare di $i = 1, \ldots, 4$ e $j = 1, \ldots, 3$.

Grafico di una funzioni a due variabili (2/2)

Si voglia rappresentare il grafico della funzione $f(x,y) = sin(\sqrt{x^2+y^2})/\sqrt{x^2+y^2}$.
Scriviamo:

```
>> x=linspace(-3*pi,3*pi);
>> y=linspace(-3*pi,3*pi);
>> [X,Y]=meshgrid(x,y);
>> R=sqrt(X.^2+Y.^2);
>> Z=sin(R)./R;
>> mesh(X,Y,Z)
```



Provare anche:

```
>> surf(X,Y,Z)
>> surfl(X,Y,Z)
>> shading interp
>> colormap copper
>> contour(X,Y,Z)
```