



Laboratorio informatico per l'ingegneria elettrica

Università di Pisa Dipartimento di
Sistemi Elettrici e Automazione

Luca Sani

Dipartimento di Sistemi Elettrici e Automazione

Università di Pisa

tel. 050 2217364

email luca.sani@dsea.unipi.it

Laboratorio informatico per l'ingegneria elettrica

Crediti 3 CFU di laboratorio

Numero totale ore di lezione (L): 0

Numero totale di ore di Laboratorio (Lab): 36

Obiettivi

Fornire agli studenti la conoscenza di alcuni pacchetti software utilizzati nell'ambito dell'ingegneria elettrica.

Programma di massima

I° Parte **MATLAB**

Introduzione. Operazioni su matrici e vettori. Operazioni su polinomi. Interpolazione. Operazioni su stringhe. Grafica 2D e 3D. Studio di funzioni. Gli script file e le funzioni. Importazioni di dati. Operazioni su strutture. Il calcolo simbolico. La Graphics User Interface (GUI). Introduzione a Simulink. Le librerie. Rappresentazione di un modello dinamico. La simulazione. I sottosistemi e le maschere. Introduzione allo State Flow. Il Control System Toolbox. Simulazione di circuiti e macchine elettriche con il System Power blockset.

II° Parte **ORCAD- PSPICE**

Introduzione. Rappresentazione di un circuito elettrico. La simulazione. Esempi. (Lab: 4)

III° Parte **LAB VIEW**

Introduzione. Il linguaggio G. Esempi.

Testi di riferimento

- I manuali di MATLAB (in inglese)
- A. Cavallo, R. Setola e F. Vasca '*La nuova guida a MATLAB Simulink e Control Toolbox*', Liguori Editore, 2004, Napoli.
- I manuali di Orcad-Spice
- I manuali di Labview

Università di Pisa - Dipartimento di
Sistemi Elettrici e Automazione

Linguaggio di programmazione

Un linguaggio di programmazione è costituito da un insieme di regole grammaticali e sintattiche, attraverso le quali è possibile definire le caratteristiche dei dati da elaborare e la successione delle operazioni che il computer deve svolgere per risolvere un determinato problema.

Più le regole sono vicine a quelle del linguaggio umano, più il linguaggio si dice ad **alto livello** (C, Pascal, Matlab). Se invece seguono da vicino il modo di operare della macchina si dice a **basso livello** (Assembler).

Linguaggio di programmazione

I linguaggi ad alto livello hanno i vantaggi della **leggibilità** (la facilità con cui, leggendo un codice sorgente, si può capire cosa fa e come funziona) e della **portabilità** (la possibilità che portando il codice scritto su una certa piattaforma (CPU + architettura + sistema operativo) su un'altra, questo funzioni subito, senza doverlo modificare).

Quelli a basso livello sfruttano meglio le caratteristiche della macchina (dimensioni ridotte e velocità di esecuzione più alta).

Sintassi

In gergo informatico, si indica con **sintassi** la correttezza dei comandi impartiti al computer.

Università di Pisa - Dipartimento di
Sistemi Elettrici e Automazione

Linguaggi interpretati e compilati

Un **linguaggio interpretato** è un linguaggio di programmazione i cui programmi vengono eseguiti da un opportuno programma, detto **interprete**.

Ad esempio il Basic o Matlab sono programmi interpretati (non si crea un eseguibile).

Un linguaggio compilato (C, Pascal) è un linguaggio di programmazione nel quale il programma viene compilato da un apposito software (**compilatore**) e tradotto in un linguaggio macchina direttamente eseguibile dal processore della macchina.

Linguaggi interpretati e compilati

L'approccio interpretato comporta una minore efficienza a run-time; un programma interpretato, in esecuzione, richiede più memoria ed è meno veloce.

Durante l'esecuzione, l'interprete deve infatti analizzare le istruzioni a partire dal livello sintattico, identificare le azioni da eseguire (eventualmente trasformando i nomi simbolici delle variabili coinvolte nei corrispondenti indirizzi di memoria), ed eseguirle; mentre le istruzioni del codice compilato, già in linguaggio macchina, vengono caricate e istantaneamente eseguite dal processore.

Linguaggi interpretati e compilati

In un linguaggio interpretato è decisamente più potente la fase di debugging:

la maggior parte degli interpreti consentono all'utente di agire sul programma in esecuzione sospendendolo, ispezionando o modificando i contenuti delle sue variabili, e così via, in modo spesso più flessibile e potente di quanto si possa ottenere, per il codice compilato, da un debugger.

Algoritmo

Con un linguaggio di programmazione è possibile affrontare solo problemi che hanno una ragionevole soluzione e cioè riconducibili ad algoritmi.

Un **algoritmo** si può definire come un *procedimento* che consente di *ottenere* un dato *risultato* eseguendo, in un determinato ordine, un insieme finito di *passi semplici* corrispondenti ad azioni scelte solitamente da un insieme finito. Non deve essere ambiguo, nel senso che l'interpretazione delle istruzioni deve essere unica e deve essere generale, ossia deve risolvere una classe di problemi e non unico specifico.

Programma

Un programma non è altro che la traduzione di un algoritmo in un determinato linguaggio di programmazione.

Università di Pisa - Dipartimento di
Sistemi Elettrici e Automazione

Programma

Il termine programma deve essere distinto da quello, più generico, di **software**; per programma si intende infatti un oggetto software che può essere caricato nella memoria di un computer ed eseguito in un nuovo processo.

Università di Pisa - Dipartimento di
Sistemi Elettrici e Automazione

Diagramma a blocchi

Un **diagramma a blocchi** (detto anche **diagramma di flusso** o **flow chart**) è un linguaggio formale di tipo grafico per rappresentare un algoritmo.

Esso consente di descrivere le differenti operazioni sotto forma di uno schema in cui le diverse fasi del processo e le differenti condizioni che devono essere rispettate vengono rappresentati da simboli grafici detti blocchi elementari. I blocchi sono collegati tra loro tramite frecce che indicano la cronologia.

Diagramma a blocchi

Esistono 5 tipi di blocchi elementari:

Blocco iniziale

Università di Pisa
Sistemi Elettrici e Automazione

Blocco di lettura/scrittura

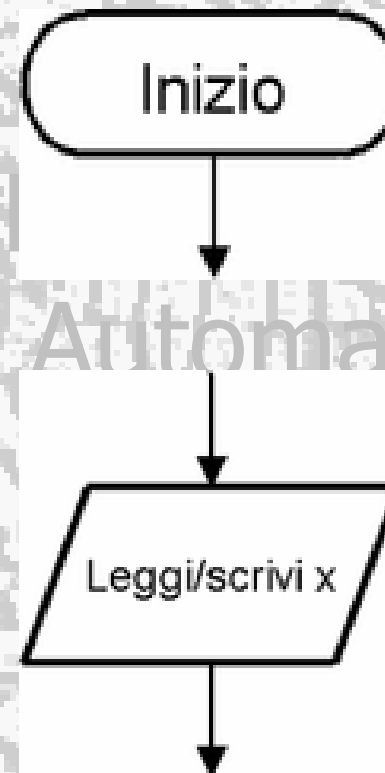
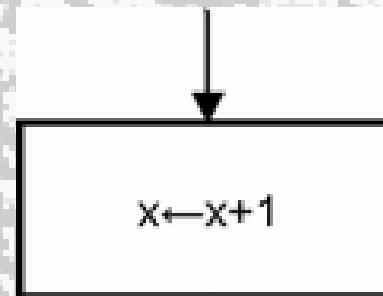


Diagramma a blocchi

Blocco azione



Blocco di controllo

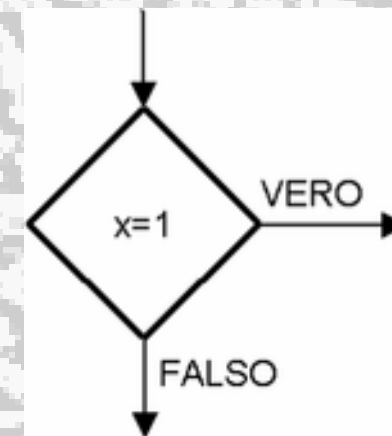


Diagramma a blocchi

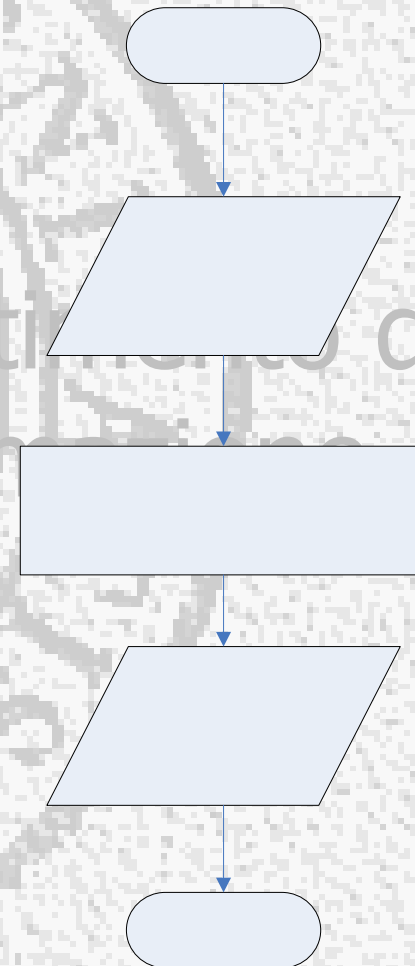
Blocco finale



Università di Pisa - Dipartimento di
Sistemi Elettrici e Automazione

Esempio di algoritmo

Calcolo del volume e della superficie di un cilindro



Elementi di un programma

Un programma è costituito dall'insieme di **algoritmi** e **dati**.

I dati rappresentano tutte le informazioni che intervengono in un modo o nell'altro nell'elaborazione di un programma.

I dati possono essere di **ingresso** nel senso che vengono 'letti' dal programma (da tastiera, da disco, ecc.), di **uscita** quando il programma, invece, li 'scrive' su un dispositivo esterno alla CPU, sia esso il video, sia la stampante ecc... oppure sono detti interni, cioè prodotti dall'elaborazione come risultati provvisori.

Elementi di un programma

I dati si dividono in due classi: **variabili** e **costanti**.

Variabile. Una variabile è un dato o un insieme di dati, noti o ignoti, già memorizzati o da memorizzare che può cambiare valore sia durante l'esecuzione del programma, sia quando cambiano le condizioni esterne.

Una variabile è sempre identificata con un nome e ad essa corrisponde sempre, da qualche parte, un certo numero (fisso o variabile) di locazioni di memoria che vengono ***allocate***, cioè riservate, per contenere i dati stessi. Molti linguaggi inoltre attribuiscono alle variabili un **tipo**, con differenti proprietà (stringhe di testo, numeri, liste, ecc.).

Elementi di un programma

Costanti: sono i dati il cui valore non cambia mai ogni volta che viene eseguito il programma. Per loro la definizione del tipo è implicita. Come le variabili anche le costanti sono identificate con un nome e corrispondono ad un determinato numero di locazioni di memoria.

Università di Pisa - Dipartimento di
Sistemi Elettrici e Automazione

Elementi di un programma

Istruzione: un comando, una azione concreta, oppure una regola descrittiva: anche il concetto di istruzione è molto variabile fra classi di linguaggi diverse. A prescindere dal particolare linguaggio però, ogni volta che un'istruzione viene eseguita, lo stato interno del calcolatore (che sia lo stato reale della macchina oppure un ambiente virtuale, teorico, creato dal linguaggio) cambia.

Elementi di un programma

Espressione: una combinazione di variabili e costanti, unite da operatori; le espressioni sono state introdotte inizialmente per rappresentare le espressioni matematiche, ma in seguito la loro funzionalità si è estesa. Una espressione viene valutata per produrre un valore, e la sua valutazione può produrre "effetti collaterali" sul sistema e/o sugli oggetti che vi partecipano.

Elementi del programma

Strutture di controllo: sono costrutti sintattici di un linguaggio di programmazione relativi al controllo del flusso di esecuzione di un programma, ovvero che servono a specificare se, quando, in quale ordine e quante volte devono essere eseguite le istruzioni che lo compongono.

Sistemi Elettrici e Automazione

Elementi del programma

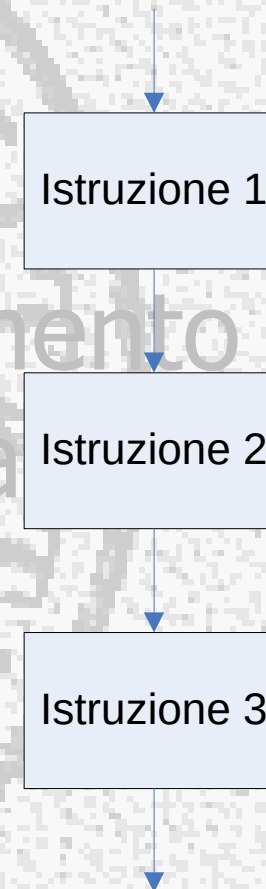
Sottoprogramma: un blocco di codice che può essere richiamato da qualsiasi altro punto del programma.

Strutture dati: meccanismi che permettono di organizzare e gestire dati complessi.

La sequenza

La **sequenza** è la struttura di controllo fondamentale di qualsiasi linguaggio, inclusi i linguaggi macchina.

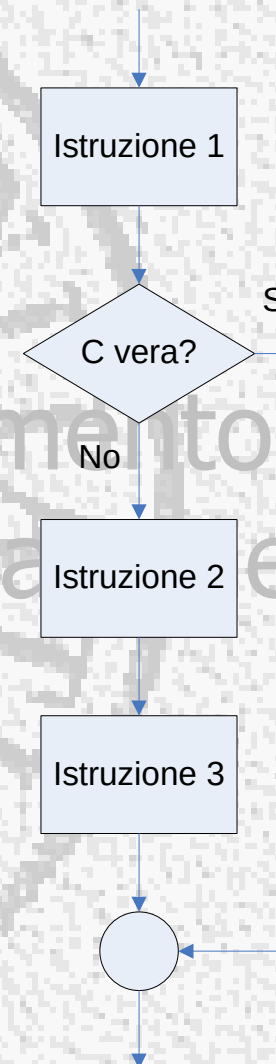
Essa si applica quando si vuole, semplicemente, che le istruzioni di un programma siano eseguite nell'ordine in cui compaiono nel testo del programma stesso. Di norma, essa non ha una espressione sintattica esplicita; in altre parole, laddove non vengano utilizzate esplicitamente strutture di controllo di altro genere, per "default" la macchina virtuale di un linguaggio esegue le istruzioni del programma sequenzialmente.



Il salto (Goto)

Insieme alla sequenza, il **goto** (vai a) è la struttura di controllo più semplice; anch'essa appare, in qualche forma, in tutti i linguaggi macchina.

Il significato generale del goto è quello di far "passare" il controllo a una istruzione specificata, che può trovarsi in un punto qualsiasi del programma. Il goto ammette sempre anche (o solo) una forma condizionata, il cui significato può essere parafrasato come segue: "se è vera una condizione C, vai alla istruzione I". Un buon programma non dovrebbe avere goto.



Blocchi decisionali

I blocchi decisionali costituiscono le strutture di controllo "alternative". Consentono di specificare che una data istruzione o un dato blocco di istruzioni devono essere eseguiti "(solo) se" vale una certa condizione.

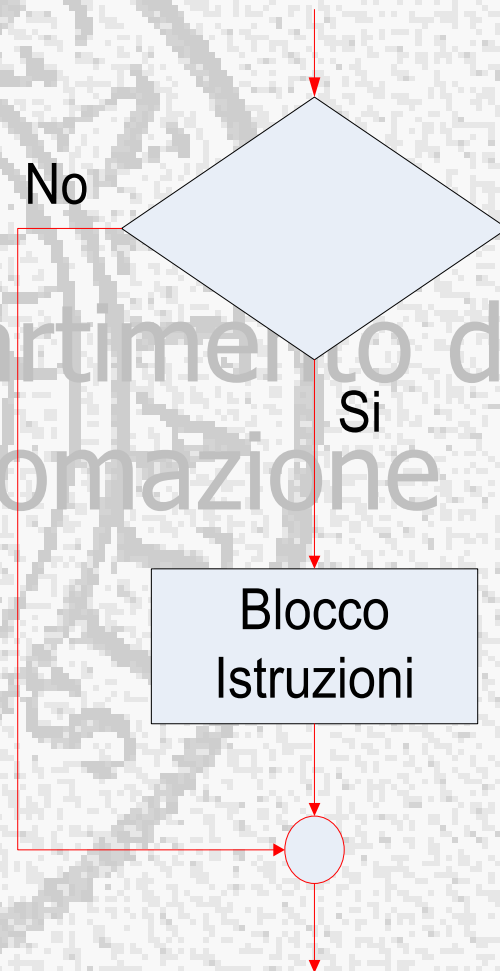
Blocco If

Blocco If – Else

Blocco Case

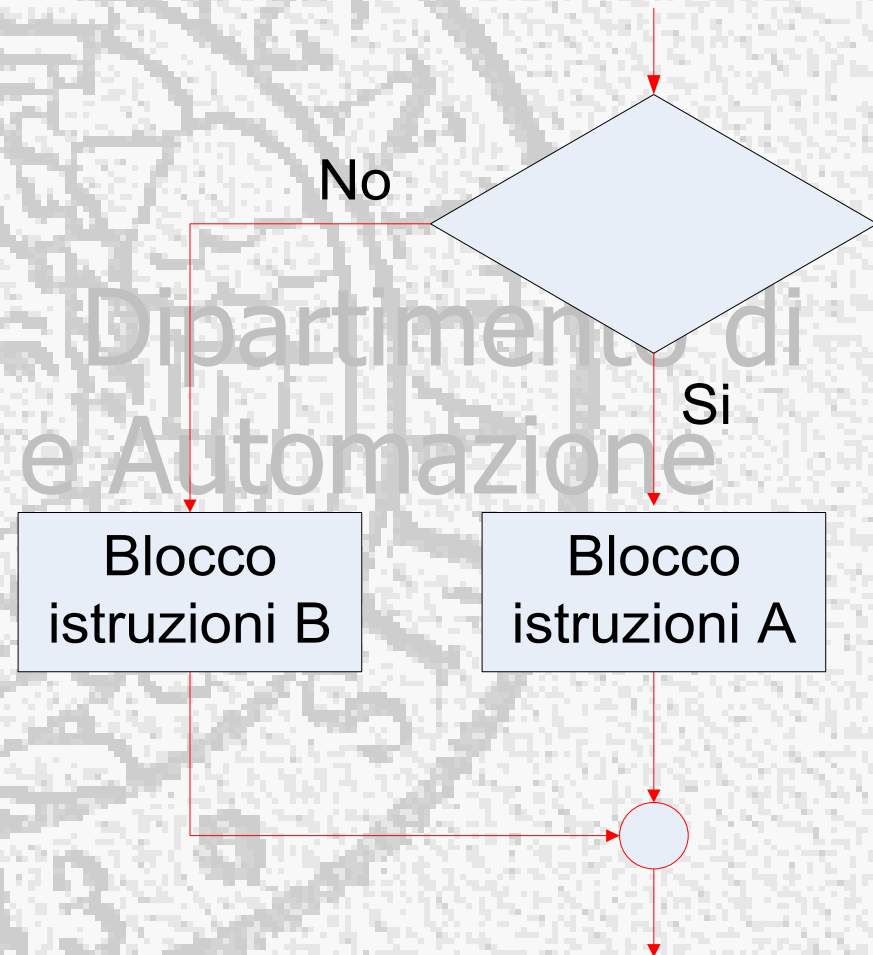
Blocco If - then

L'alternativa if-then (se-allora) è la più semplice forma di alternativa. Il suo significato può essere parafrasato con la frase "se vale la condizione C, esegui l'istruzione (blocco) I".



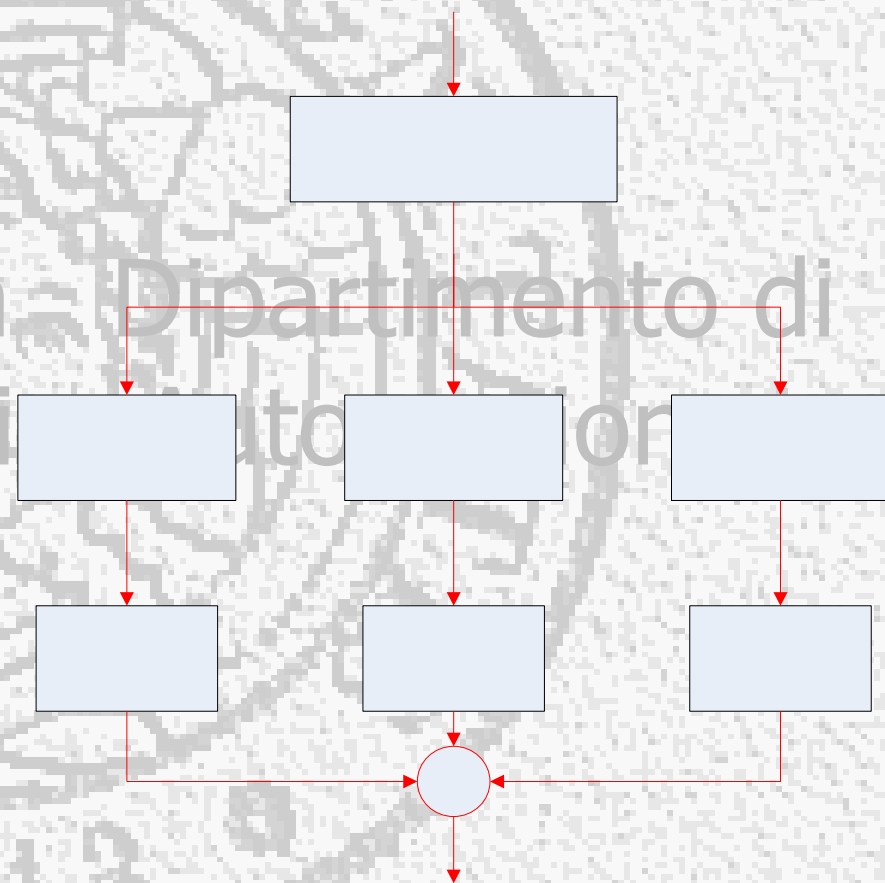
Blocco If-Then-Else

La forma più articolata **if-then-else** (*se-allora-altrimenti*), che si può parafrasare come: "se vale la condizione C esegui l'istruzione (blocco A); altrimenti esegui l'istruzione (blocco B)".



Blocco Case

L'alternativa case può essere assimilata a una catena di if-then-else con certe restrizioni. In questo caso, la scelta di uno fra N istruzioni o blocchi alternativi viene fatta sulla base del valore di una determinata variabile o espressione, normalmente di tipo intero. Essa può essere parafrasata come segue: "valuta il valore N; nel caso in cui il suo valore sia V_1 , esegui I_1 ; nel caso in cui sia V_2 , esegui I_2 (ecc.)".



Iterazioni

Le strutture di controllo "iterative" consentono di specificare che una data istruzione o un dato blocco di istruzioni devono essere eseguiti ripetutamente. Esse vengono anche dette **cicli**. Ogni struttura di controllo di questo tipo deve consentire di specificare sotto quali condizioni l'iterazione (ripetizione) di tali istruzioni debba terminare, ovvero la condizione di terminazione del ciclo oppure, equivalente mente, la condizione di permanenza nel ciclo. Di seguito si esaminano le strutture di controllo più note di questa categoria.

Ciclo For

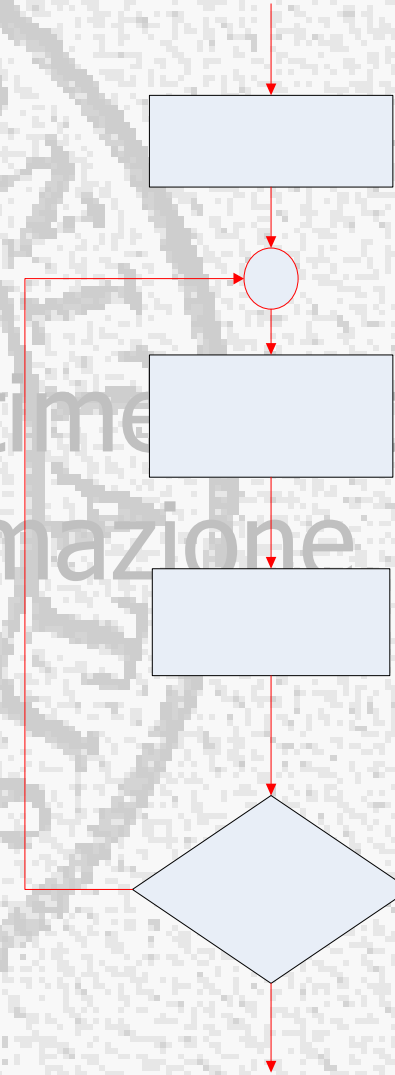
Ciclo While

Ciclo Repeat-Until

Ciclo For

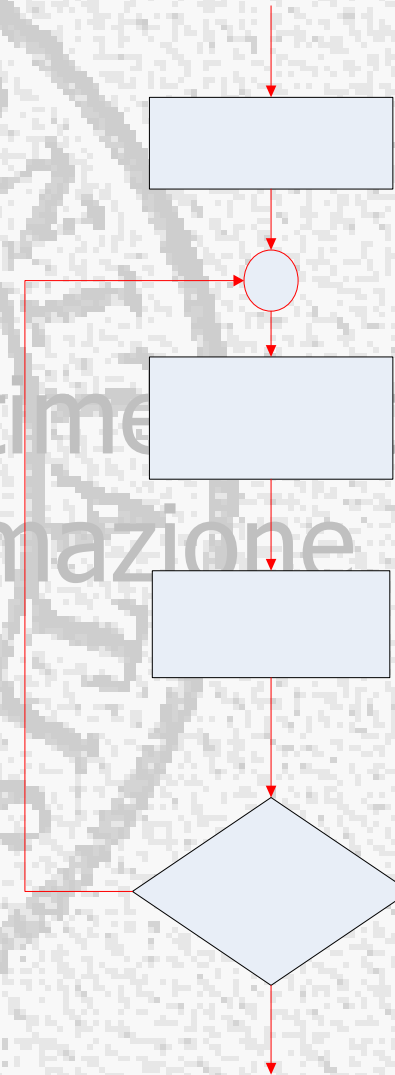
Il modo più naturale per esprimere la condizione di permanenza in un ciclo consiste nello specificare quante volte debbano essere ripetuti l'istruzione o il blocco controllati dal ciclo.

Le forme tradizionali di ciclo for possono essere parafrasate come "ripeti (il codice controllato) per i che va da un certo valore iniziale a un certo valore finale, con un certo passo". **i** è in generale una variabile di tipo intero, detta **contatore**.



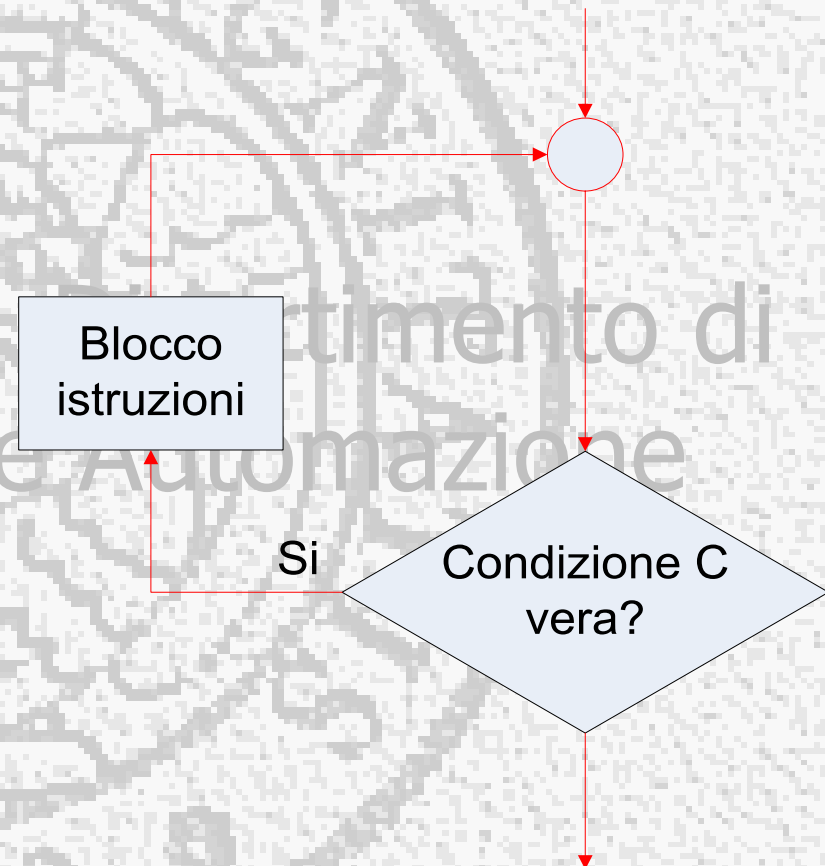
Ciclo For

La maggior parte dei programmi che utilizzano array si servono della struttura di controllo "cicli for" per attraversare gli array, ovvero per accedere sequenzialmente alle celle



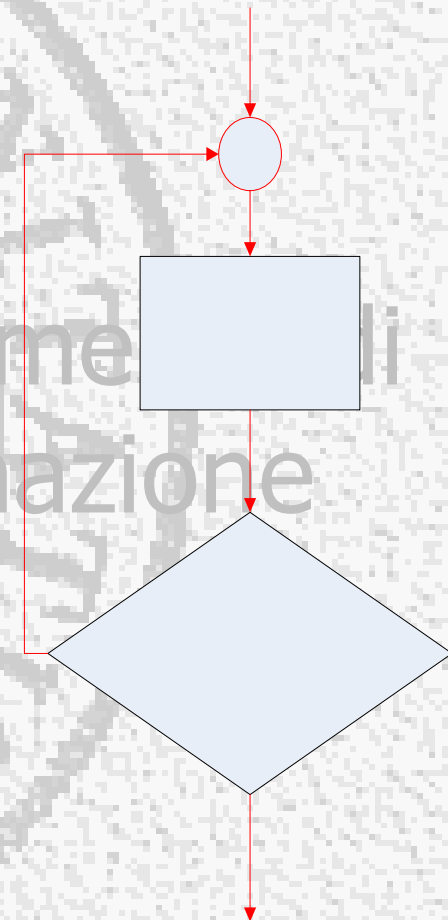
Ciclo While

Il ciclo **while** (*mentre, o fintantoché*) è indicato quando la condizione di permanenza in un ciclo è una generica condizione booleana, indipendente dal numero di iterazioni eseguite. Le forme tradizionali di ciclo while possono essere parafrasate come "ripeti (il codice controllato) fintantoché resta vera la condizione C". Un esempio tipico è la lettura di dati da un file di cui non si conosca a priori la dimensione; esso potrebbe assumere la forma "leggi il prossimo dato finché non incontri la fine del file".



Ciclo Repeat - Until

Il ciclo **repeat-until** (ripeti finché) differisce dal while per due aspetti. Innanzitutto, esso garantisce che venga eseguita sempre almeno una iterazione del ciclo (la condizione che controlla il ciclo viene verificata dopo aver concluso la prima iterazione). In secondo luogo, viene espressa la condizione di terminazione del ciclo anziché quella di permanenza nel ciclo (quest'ultima differenza non ha comunque un impatto concettuale molto importante, essendo le due condizioni esprimibili semplicemente come negazione l'una dell'altra)



Tipo di dati

In informatica un **tipo di dato** (o semplicemente **tipo**) è un nome che indica l'insieme di valori che una variabile, o il risultato di un'espressione, possono assumere, e le operazioni che su tali valori si possono effettuare.

Dire per esempio che la variabile X è di tipo "numero intero" significa affermare che X può assumere come valori solo numeri interi (appartenenti ad un certo intervallo) e che su tali valori sono ammesse solo certe operazioni (ad esempio le operazioni aritmetiche elementari).

Per ogni tipo di dato si ha una diversa occupazione di memoria e la loro elaborazione richiede regole diverse.

Tipo di dati

Nei linguaggi **tipizzati** è necessario associare alle variabili, alle espressioni e più in generale ai termini dei programmi delle annotazioni o **dichiarazioni di tipo**.

Queste annotazioni di tipo, a seconda del linguaggio, devono essere specificate esplicitamente dal programmatore, oppure possono essere generate in modo automatico.

In linguaggi come il Pascal o il C occorre dichiarare esplicitamente il tipo associato alla variabile. In Matlab avviene automaticamente con la prima istruzione di assegnamento.

Tipo di dati

I tipi di dato si distinguono in semplici e strutturati:

Semplici

- Booleani
- Numerici
- Carattere

Strutturati

- Array
- Record
- Stringhe

Tipo di dati - Booleano

Il tipo booleano prevede due soli valori: **true** ("vero") e **false** ("falso"). Questi valori vengono utilizzati in special modo nelle espressioni condizionali per controllare il flusso di esecuzione; inoltre possono essere manipolati con gli operatori booleani and, or, not e così via.

Anche se in teoria basterebbe un solo bit per memorizzare un valore booleano, per motivi di efficienza si usa in genere un' intera parola di memoria, come per i numeri interi "piccoli" (una parola di memoria a 8 bit, per esempio, può memorizzare numeri da 0 a 255, ma il tipo booleano utilizza solo i valori 0 e 1).

Tipo di dati - Numero

I tipi di dati numerici includono i numeri **interi (integer)** e i numeri razionali in **virgola mobile (float)**, che sono astrazioni dei corrispondenti insiemi di numeri della matematica. Quasi tutti i linguaggi includono tipi di dati numerici come tipi predefiniti e forniscono un certo numero di operatori aritmetici e di confronto su di essi.

A differenza degli insiemi numerici della matematica, i tipi di dati numerici sono spesso limitati (includono cioè un massimo e un minimo numero rappresentabile), dovendo essere contenuti in una singola parola (word) di memoria.

Tipo di dati - Carattere

Il tipo **carattere** (**char**) contiene, per l'appunto, un carattere: generalmente si riferisce ad un carattere ASCII e viene memorizzato in un byte. Tuttavia in questi anni si sta affermando il nuovo standard Unicode per i caratteri, che prevede 16 bit (che generalmente corrisponde a una parola di memoria) per la rappresentazione di un singolo carattere.

Le **stringhe** sono sequenze di caratteri di lunghezza finita. I linguaggi possono fornire operazioni per la concatenazione di stringhe, la selezione di sottostringhe di una stringa data, ecc.

Tipo di dati - Puntatore

I valori di tipo **puntatore** sono indirizzi di memoria (in genere indirizzi di variabili o di altri elementi del programma come funzioni o metodi. Nella programmazione procedurale, l'uso di puntatori è necessario per costruire strutture dati complesse e dalla forma non prevedibile a priori (ed eventualmente variabile nel tempo), come grafi, alberi, liste ecc. (in molti linguaggi esistono tipi puntatore specifici per ogni altro tipo di dato: in questi casi i puntatori possono essere considerati tipi derivati).

Tipo di dati - Array

Il tipo **array** (o vettore) è un'aggregazione di dati di tipo omogeneo sotto forma di tabella ad una o più dimensioni. Ciascun elemento dell'array è detta anche cella e costituisce una variabile o una costante.

Per ogni vettore devono essere definite : le dimensioni, il tipo di dato e un nome identificatore. Si parlerà perciò di tipi come "array di interi", "array di stringhe", "array di caratteri" e così via.

Ciascuna delle celle dell'array è identificata da un valore di **indice**. L'indice è generalmente numerico e i valori che gli indici possono assumere sono numeri interi contigui che partono da 0 o da 1. Si potrà quindi parlare della cella di indice 0, di indice 1, e, in generale, di indice N, dove N è un intero compreso fra 0 (o 1) e il valore massimo per gli indici dell'array

Tipo di dati - Record

Il tipo **record** è un'aggregazione di dati di tipo più semplici, la cui composizione può essere definita dall'utente.

Un record è necessario per mantenere informazioni eterogenee correlate: potrebbero ad esempio essere usati per modellare le schede dell'archivio di una biblioteca, che devono contenere stringhe per il titolo di un libro e il nome del suo autore, ma anche un valore numerico indicante la collocazione; ciascuna di queste informazioni (**campi** del record) può essere acceduta in modo indipendente specificandone il nome.

Le funzioni (o sottoprogrammi)

Una **funzione** (detta anche subroutine, routine, procedura o sottoprogramma) è una porzione di programma, costituita da un insieme di istruzioni che complessivamente eseguono una determinata operazione o risolvono un determinato problema.

Le subroutine sono spesso raccolte in librerie. Esse possono essere "**chiamate**" ("richiamate", "invoke", "attivate") da diversi punti del programma complessivo; in questo senso, esse svolgono il ruolo di "macro-istruzioni" e rendono possibile la fattorizzazione del software.

Per esempio, una subroutine progettata per disporre in ordine crescente un insieme di numeri interi può essere richiamata in tutti i contesti in cui questa operazione sia utile o necessaria, e supplisce alla mancanza di una vera e propria "istruzione" dedicata allo scopo, consentendo al contempo di descrivere il corrispondente algoritmo di ordinamento in un unico punto del programma.

Le funzioni (o sottoprogrammi)

Quasi tutti i linguaggi di programmazione supportano le funzioni, fornendo una sintassi per definire una funzione, ovvero scriverne il codice, ed una per richiederne l'esecuzione (invocazione o chiamata della funzione).

La sintassi con cui si definisce una funzione presenta sempre un'intestazione del tipo:

[Variabili di uscita] = nome_funzione(variabili di ingresso)

Una variabile si dice globale se è stata definita nel programma principale e se è visibile anche all'interno di una funzione.

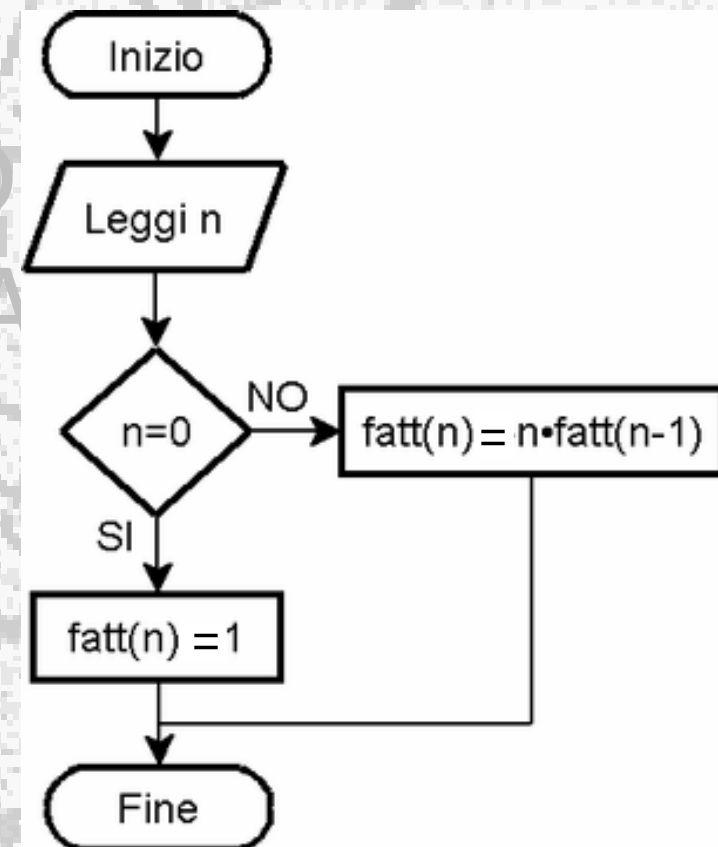
Una variabile si dice locale se è definita all'interno di una funzione e non è visibile all'esterno di questa. Quindi viene creata quando la funzione è chiamata e viene eliminata quando la funzione è terminata.

Le funzioni (o sottoprogrammi)

Esempio di funzione

Calcolo del fattoriale di un numero intero

$$fatt(n) = n! = \begin{cases} 1, & \text{se } n = 0 \\ n \cdot (n-1)!, & \text{se } n \neq 0 \end{cases}$$



The background of the slide features a large, faint, circular watermark of the University of Pisa seal. The seal contains the text "UNIVERSITAS PISANA" at the top and "1343" at the bottom, with a central emblem.

MATLAB e Simulink

Università di Pisa Dipartimento di
Sistemi Elettrici e Automazione

Corso introduttivo

Che cosa è MATLAB?

Prodotto dalla Mathworks, MATLAB è ormai uno dei pacchetti più celebri nell'ambito del calcolo scientifico.

E' stato sviluppato a partire dall'inizio degli anni '90.

Attualmente viene distribuita la versione 7.5.

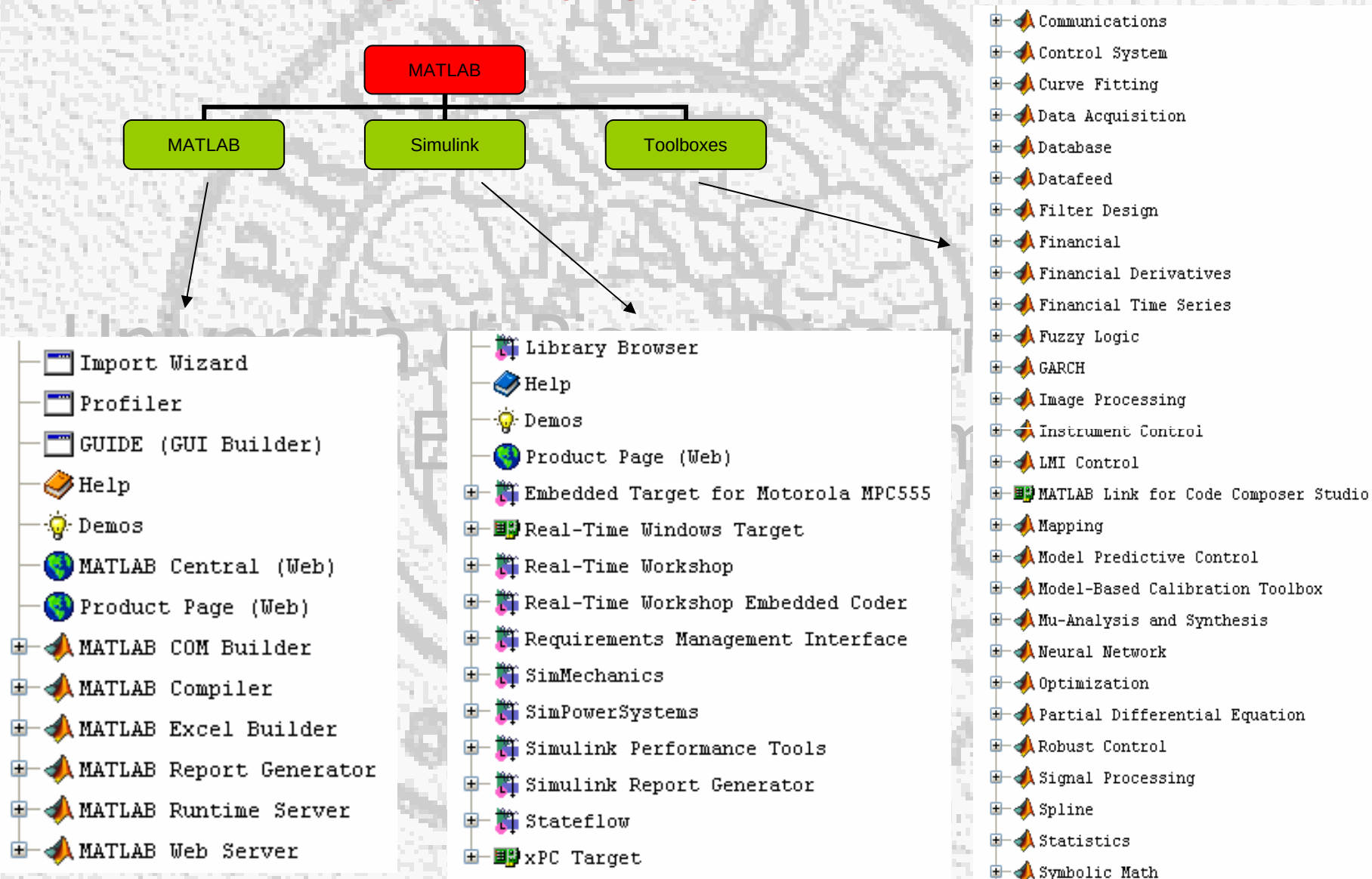
Il nome MATLAB deriva da MATRIX LABORATORY.

MATRIX perchè il tipo di dato base è la matrice.

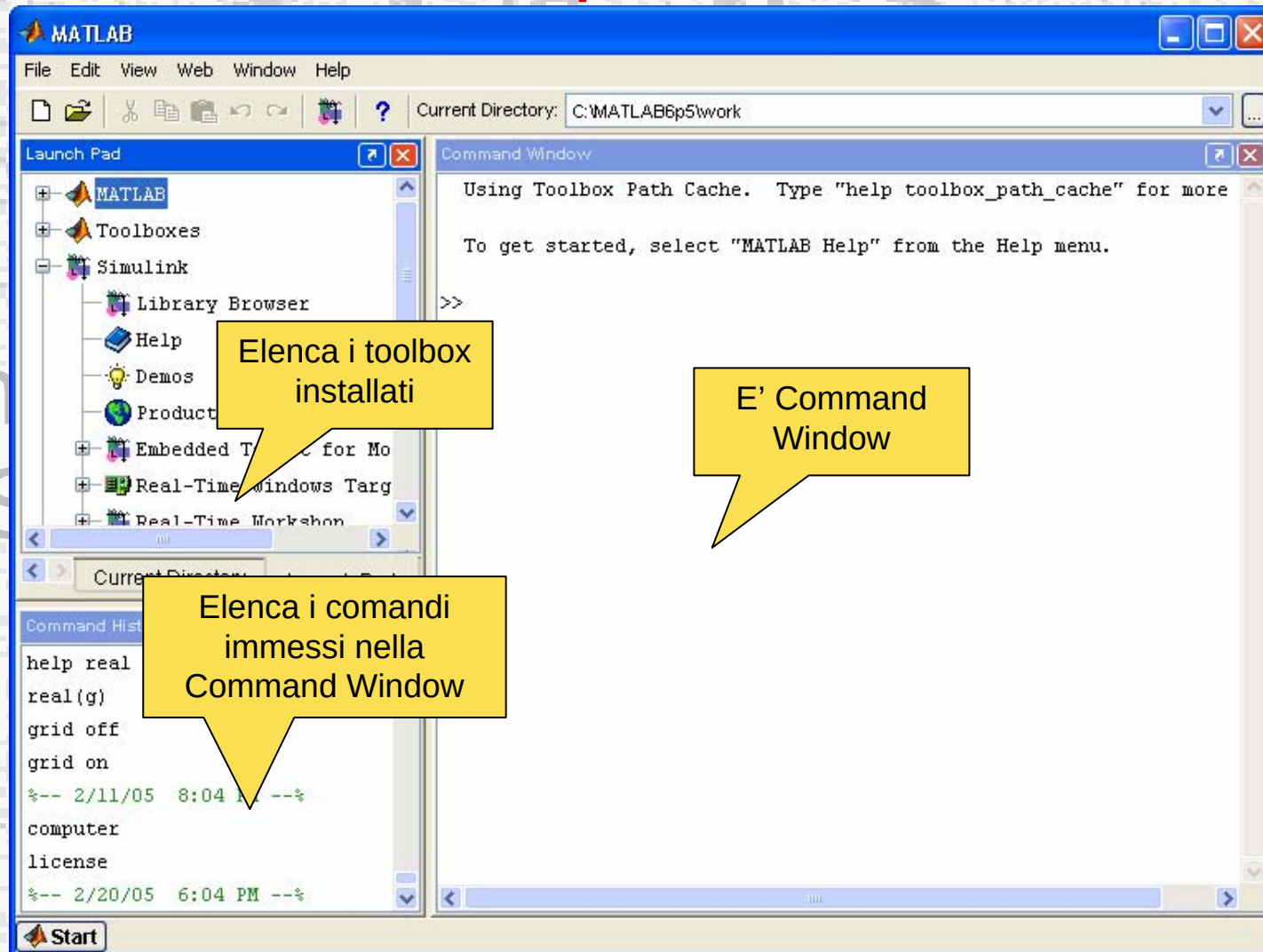
LABORATORY sta ad indicare la tipicità di impiego per la didattica e la ricerca.

MATLAB non è solo un linguaggio di programmazione, è soprattutto un ambiente interattivo estremamente complesso in cui è possibile affrontare i più svariati problemi in ambito ingegneristico e della matematica applicata.

Struttura di MATLAB



Il desktop di MATLAB



La Command Window

MATLAB può eseguire le varie istruzioni in due modalità:

- **Sessione Interattiva** (attraverso la Command Window)
- **Esecuzione di file script**

La Command Window è la finestra che permette all'utente di comunicare con il programma MATLAB (sessione Interattiva). In questa finestra l'utente digita i nomi dei comandi, delle funzioni e le istruzioni da eseguire.

Il prompt di MATLAB (`>>`) sta ad indicare che il programma è pronto a ricevere istruzioni.

Inserimento istruzioni

E' sufficiente digitare il comando al prompt di MATLAB.

Esempio

```
>> 8/10
```

```
ans =
```

```
0.8000
```

MATLAB per default visualizza il risultato di un'operazione con quattro cifre decimali.

MATLAB assegna la risposta a una variabile temporanea chiamata **ans**, che è l'abbreviazione di answer (risposta).

In **ans** è memorizzato sempre il risultato dell'ultima operazione eseguita da MATLAB per la quale non era specificata la variabile di uscita.

Operazione di assegnamento

Il risultato di un'operazione viene assegnato ad una variabile con l'operatore di assegnazione (=) nel seguente modo:

nome_var=operazione

esempio

```
>>x=10+2
```

Sul lato sinistro di = può esserci una sola variabile. Non è valida `>>x+y=10`.

Il lato destro dell'operatore di assegnazione deve essere un valore calcolabile.

Il punto e virgola (;) alla fine dell'operazione non fa visualizzare il risultato nella Command Window.

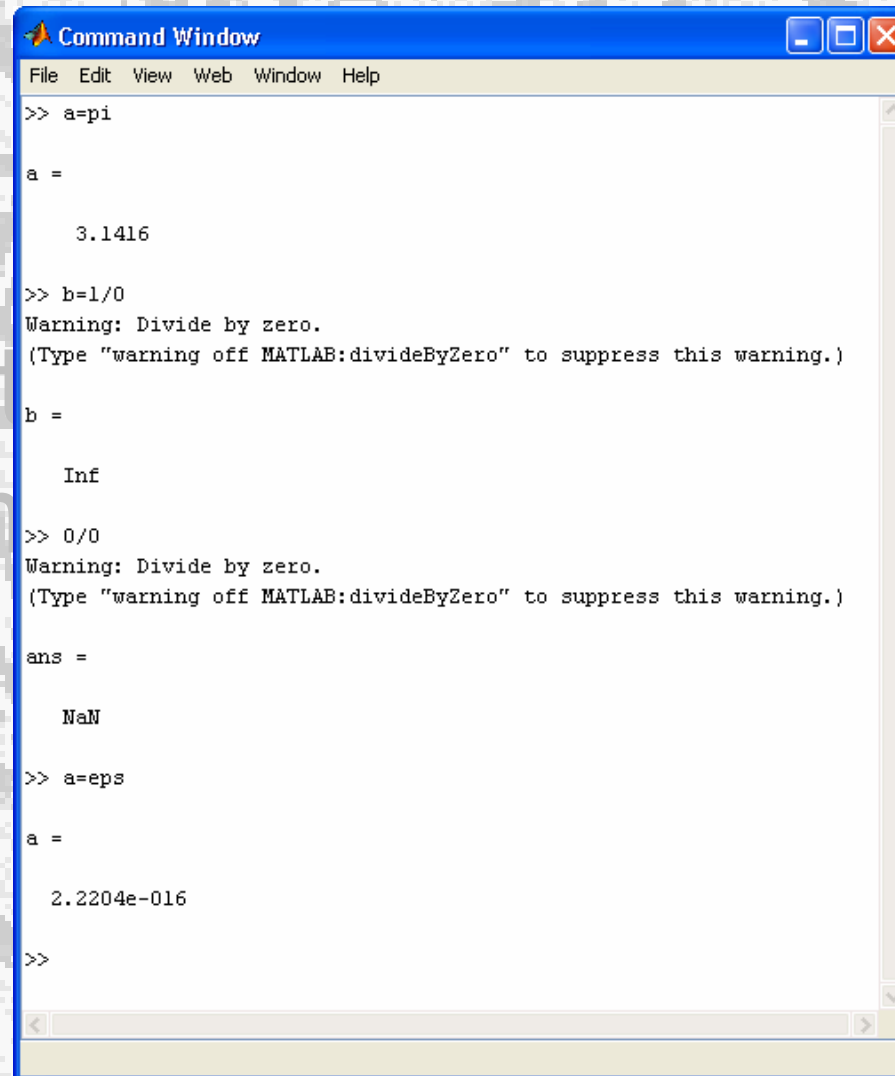
MATLAB è **case sensitive** (distingue tra maiuscolo e minuscolo), quindi, ad esempio, A e a sono due variabili diverse.

Costanti predefinite

Esistono in MATLAB una serie di costanti predefinite:

- **pi** è il π
- **ans** variabile temporanea che contiene il risultato più recente
- **i,j** l'unità immaginaria
- **inf** infinito
- **NaN** risultato numerico non definito
- **eps** zero macchina

Costanti Predefinite Esempio

A screenshot of the MATLAB Command Window. The window has a blue title bar with the text 'Command Window' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Web', 'Window', and 'Help'. The main area contains the following text:

```
>> a=pi  
  
a =  
  
    3.1416  
  
>> b=1/0  
Warning: Divide by zero.  
(Type "warning off MATLAB:divideByZero" to suppress this warning.)  
  
b =  
  
    Inf  
  
>> 0/0  
Warning: Divide by zero.  
(Type "warning off MATLAB:divideByZero" to suppress this warning.)  
  
ans =  
  
    NaN  
  
>> a=eps  
  
a =  
  
    2.2204e-016  
  
>>
```

Costanti Predefinite Esempio

Le variabili presenti nel workspace possono essere memorizzate su file con la funzione **save**.

Sintassi

save nome_file var1 var2 ... [-ASCII] [-TABS]

Le variabili sono memorizzate in formato MATLAB.

Con l'opzione **-ASCII** sono salvate in formato ASCII.

Con l'opzione **-ASCII -TABS** sono salvate in formato ASCII e separate da spazi di tabulazione.

Le variabili memorizzate in un file possono essere caricate nel workspace con la funzione **load**.

Sintassi

load nome_file

Costanti Predefinite Esempio

E' possibile importare dati prodotti da altri software con l'importazione guidata di Matlab, invocando dal menu File il comando Import Data....

Questo comando attiva l'Import Wizard che segue l'importazione in vari passi.

Università di Pisa - Dipartimento di
Sistemi Elettrici e Automazione

Array

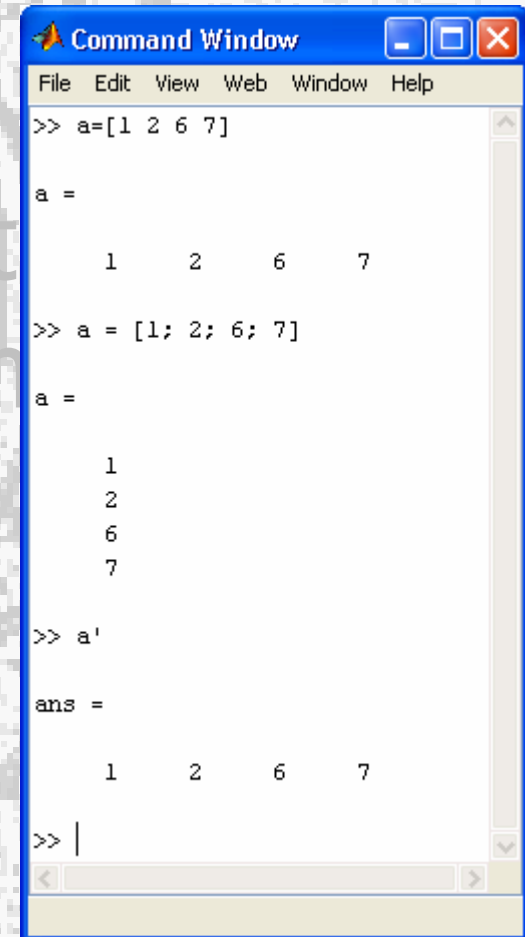
La memorizzazione di un vettore in una variabile è semplice. Non occorre dichiarare la variabile e le sue dimensioni.

Sintassi

nome_var = [elementi del vet.]

Se gli elementi sono separati da spazi o virgole (,) si crea un vettore riga, se sono separati da punti e virgole (;) si crea un vettore colonna.

Si può passare da una forma all'altra con l'operatore apice (').



```
Command Window
File Edit View Web Window Help
>> a=[1 2 6 7]

a =

     1     2     6     7

>> a = [1; 2; 6; 7]

a =

     1
     2
     6
     7

>> a'

ans =

     1     2     6     7

>> |
```

Rappresentazione di intervalli

Per memorizzare array con elementi equispaziati si può procedere in modo più rapido.

- Operatore (`:`)

crea un vettore di elementi a partire da `inf`, con passo `step` fino a `sup`.

Sintassi

`nome_var = inf:step:sup`

step default = 1

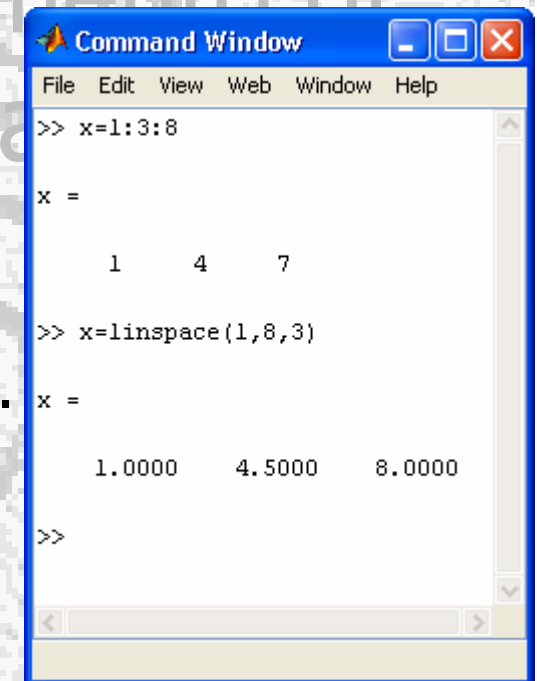
- Funzione `linspace`

crea un vettore di `N` elementi equispaziati tra `inf` e `sup`.

Sintassi

`nome_var = linspace(inf, sup, N)`

default `N = 50`



```
Command Window
File Edit View Web Window Help
>> x=1:3:8
x =
     1     4     7
>> x=linspace(1,8,3)
x =
  1.0000  4.5000  8.0000
>>
```

Rappresentazione di intervalli Esempio

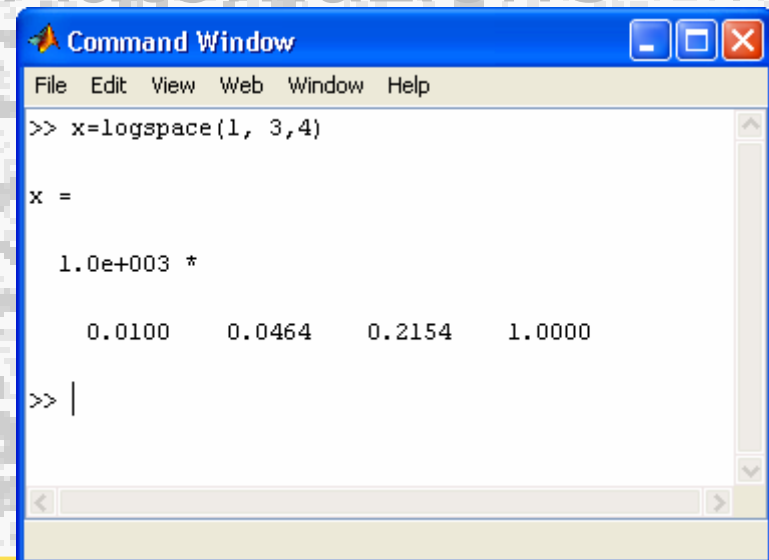
- Funzione **logspace**

crea un vettore di **N** elementi equispaziati in modo logaritmico tra 10^{inf} e 10^{sup}

Sintassi

nome_ar=logspace(inf, sup, N)

default N = 50



The screenshot shows a MATLAB Command Window with the following content:

```
File Edit View Web Window Help
>> x=logspace(1, 3,4)

x =

    1.0e+003 *
    0.0100    0.0464    0.2154    1.0000
>> |
```

Matrici

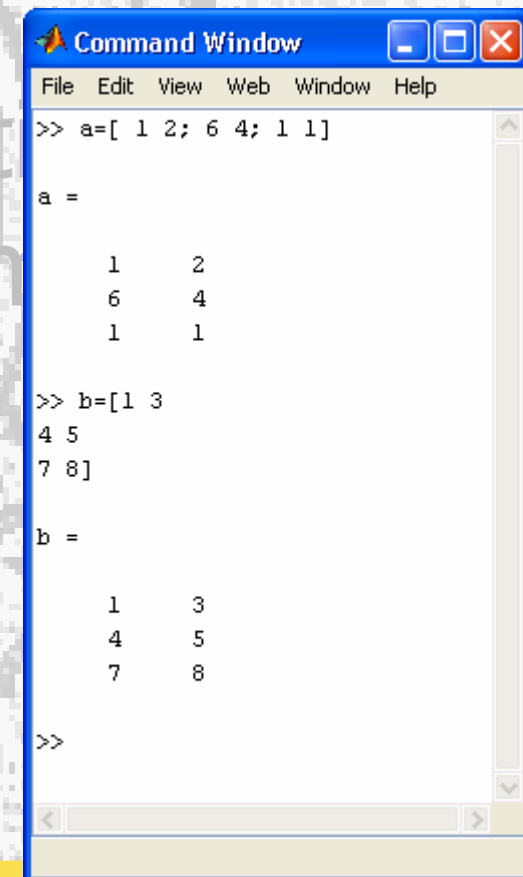
La memorizzazione di una matrice in una variabile è semplice. Non occorre dichiarare la variabile e le sue dimensioni.

Sintassi

nome_var = [1° riga; 2° riga; ...]

Gli elementi di una stessa riga sono separati da spazi o da virgole, mentre le colonne sono separate da punti e virgole (;) o con il tasto Enter.

Si possono costruire matrici ad N dimensioni.



```
Command Window
File Edit View Web Window Help
>> a=[ 1 2; 6 4; 1 1]

a =

     1     2
     6     4
     1     1

>> b=[1 3
4 5
7 8]

b =

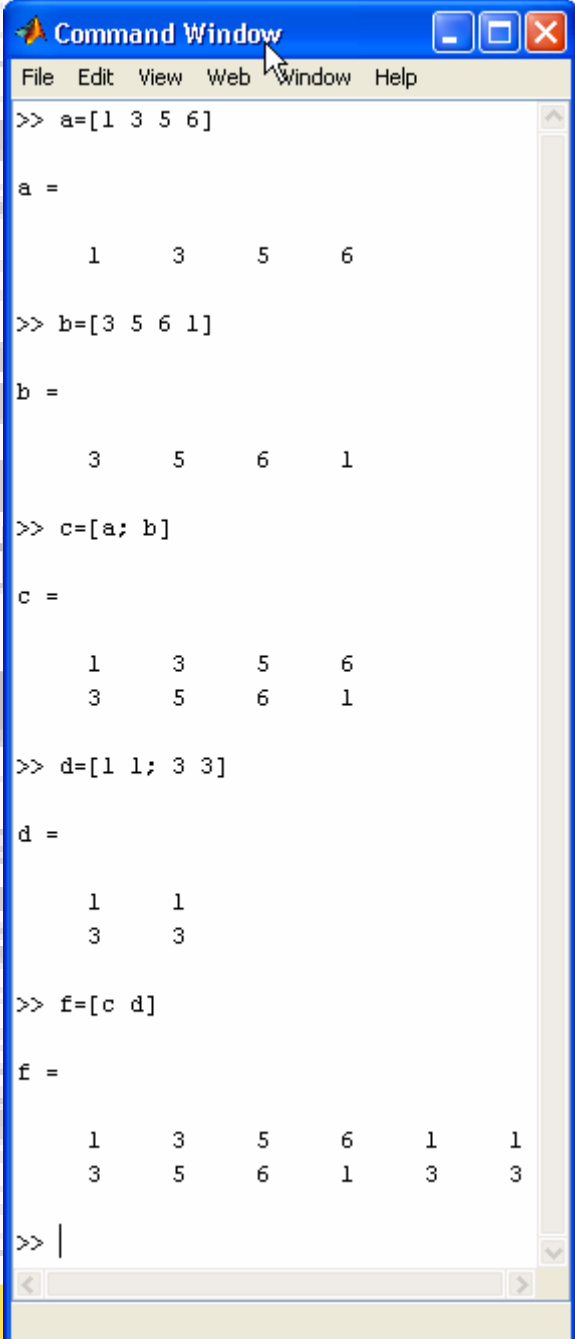
     1     3
     4     5
     7     8

>>
```

Matrici

Una matrice può essere creata anche combinando vettori riga e colonna o matrici precedentemente definite.

Università di Pisa - Dipar
Sistemi Elettrici e Auto

A screenshot of the MATLAB Command Window. The window has a blue title bar with the text 'Command Window' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Web', 'Window', and 'Help'. The main area shows a series of MATLAB commands and their outputs. The commands are: 'a=[1 3 5 6]', 'b=[3 5 6 1]', 'c=[a; b]', 'd=[1 1; 3 3]', and 'f=[c d]'. The outputs show the matrices 'a', 'b', 'c', 'd', and 'f' in a formatted way. The background of the slide features a large, faint watermark of the University of Pisa seal.

```
Command Window
File Edit View Web Window Help
>> a=[1 3 5 6]

a =

     1     3     5     6

>> b=[3 5 6 1]

b =

     3     5     6     1

>> c=[a; b]

c =

     1     3     5     6
     3     5     6     1

>> d=[1 1; 3 3]

d =

     1     1
     3     3

>> f=[c d]

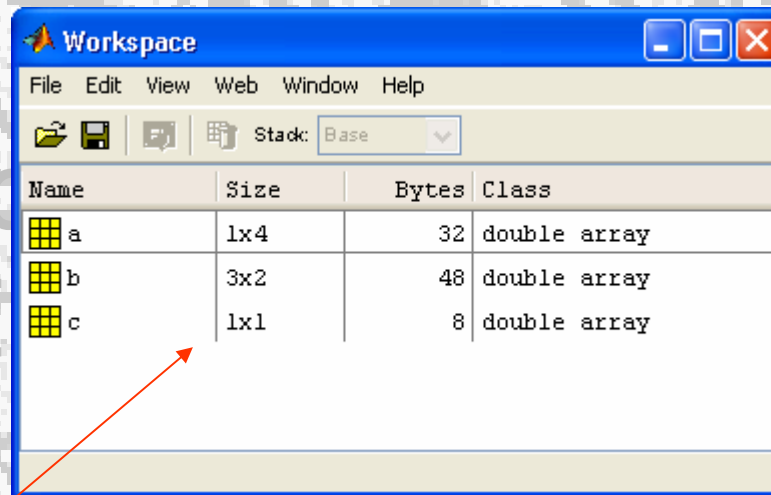
f =

     1     3     5     6     1     1
     3     5     6     1     3     3

>> |
```

Contenuto del Workspace

Nella sezione Workspace sono indicate le variabili memorizzate con le rispettive dimensioni.



The screenshot shows the MATLAB Workspace window. It has a menu bar (File, Edit, View, Web, Window, Help) and a toolbar with icons for saving, deleting, and refreshing. A 'Stack' dropdown menu is set to 'Base'. Below is a table listing variables in the workspace.

Name	Size	Bytes	Class
a	1x4	32	double array
b	3x2	48	double array
c	1x1	8	double array

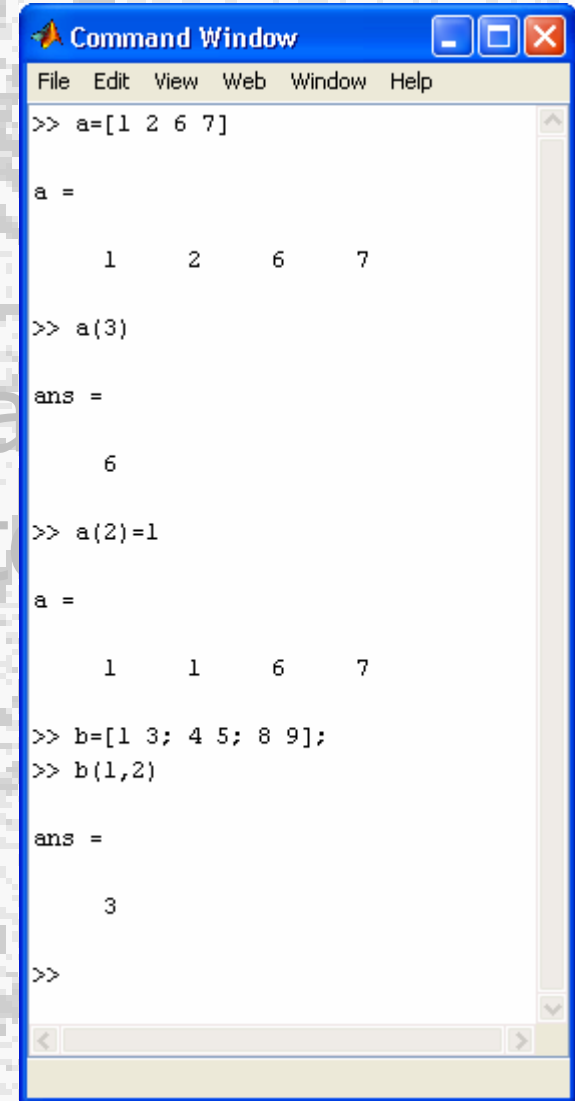
A red arrow points from the word 'Osservazione' below to the row for variable 'c' in the table.

Osservazione

Anche uno scalare è memorizzato come una matrice [1X1].

Indicizzazione degli elementi di una matrice

L'accesso ad un elemento generico di un vettore o di una matrice avviene per mezzo di indici. Si deve indicare un indice per ognuna delle dimensioni della variabile.



```
Command Window
File Edit View Web Window Help
>> a=[1 2 6 7]

a =

     1     2     6     7

>> a(3)

ans =

     6

>> a(2)=1

a =

     1     1     6     7

>> b=[1 3; 4 5; 8 9];
>> b(1,2)

ans =

     3

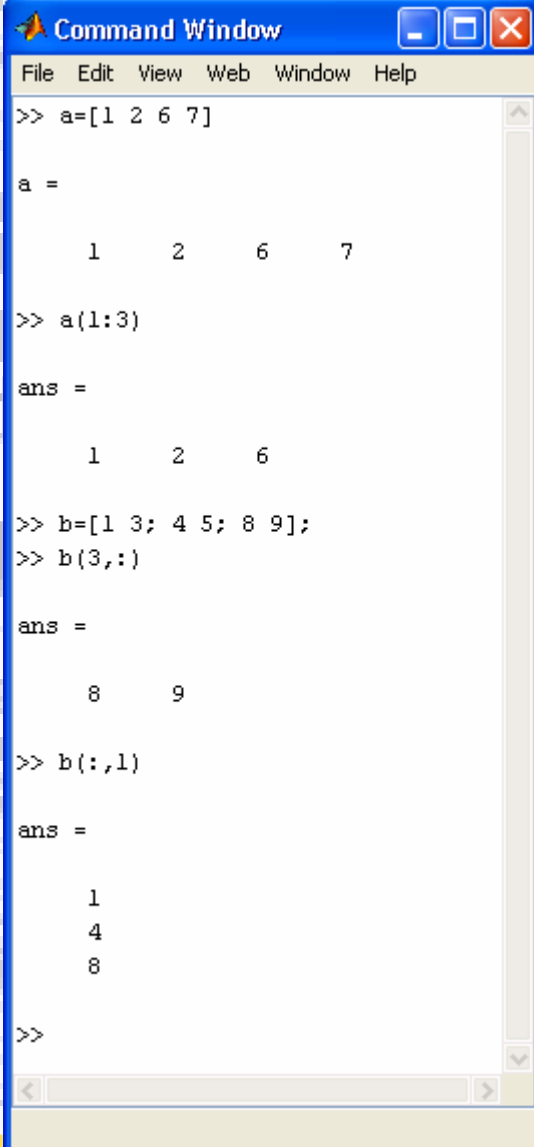
>>
```

Indicizzazione degli elementi di una matrice

Attraverso l'operatore (`:`) si può accedere ad un sottointervallo di un vettore o ad una parte di una matrice.

Sintassi

- Elementi tra l'indice *i* e l'indice *j*
nome_var(i:j)
- Tutti gli elementi della j-esima riga
nome_var(j,:)
- Tutti gli elementi della j-esima colonna
nome_var(:,j)



```
Command Window
File Edit View Web Window Help
>> a=[1 2 6 7]

a =

     1     2     6     7

>> a(1:3)

ans =

     1     2     6

>> b=[1 3; 4 5; 8 9];
>> b(3,:)

ans =

     8     9

>> b(:,1)

ans =

     1
     4
     8

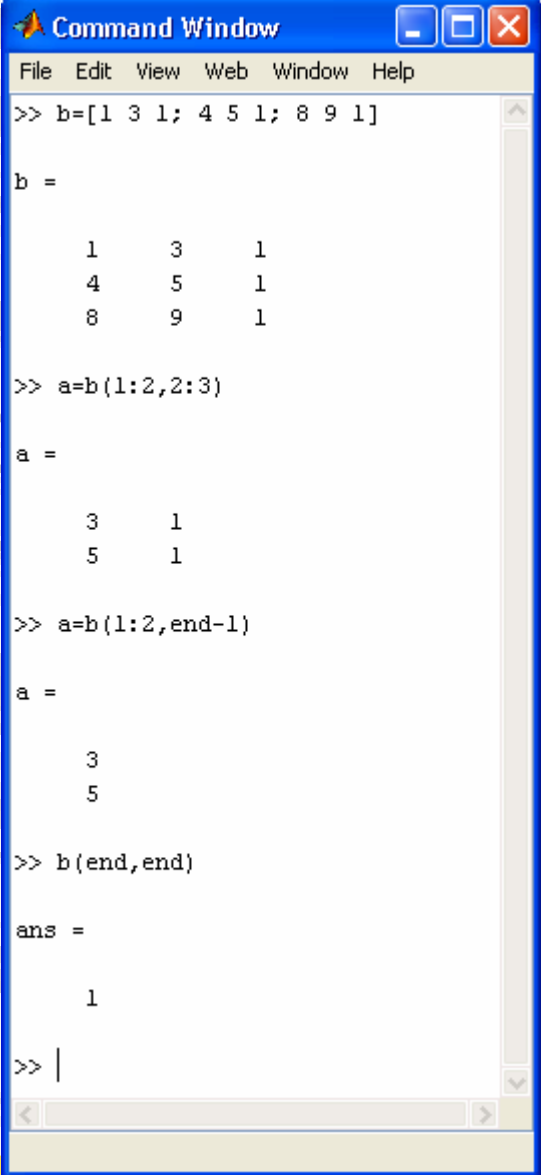
>>
```

Indicizzazione degli elementi di una matrice

- Elementi che sono compresi tra le righe **i** e **j** e tra le colonne **h** e **k**.

nome_var(i:j, h:k)

Con l'operatore **end** si può accedere agli elementi caratterizzati dal valore estremo dell'indice considerato.



```
Command Window
File Edit View Web Window Help
>> b=[1 3 1; 4 5 1; 8 9 1]

b =

     1     3     1
     4     5     1
     8     9     1

>> a=b(1:2,2:3)

a =

     3     1
     5     1

>> a=b(1:2,end-1)

a =

     3
     5

>> b(end,end)

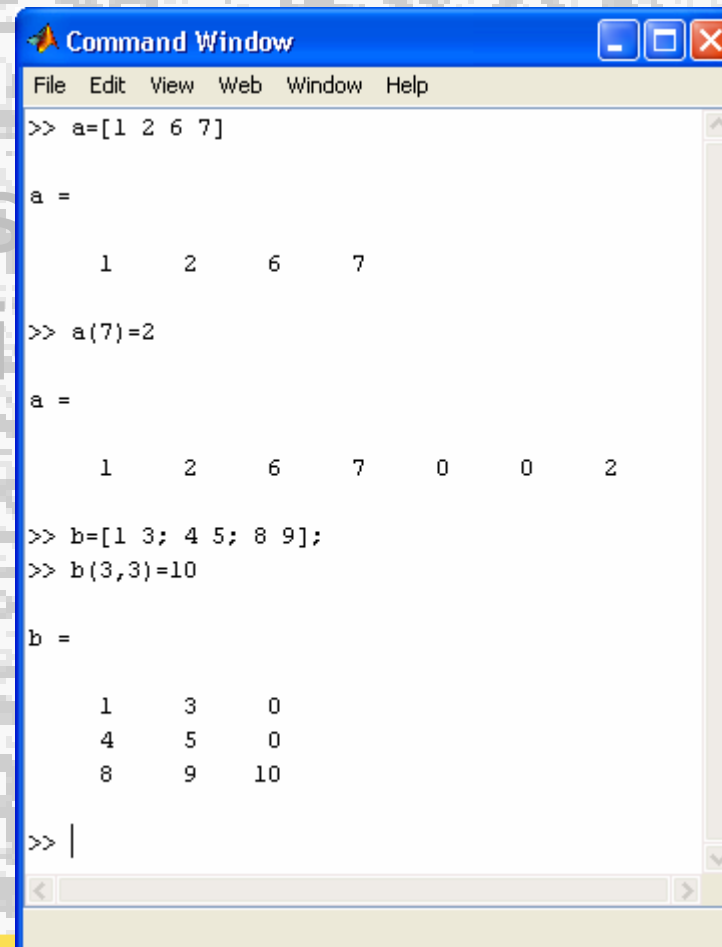
ans =

     1

>> |
```

Indicizzazione degli elementi di una matrice

Se, per mezzo dell'indicizzazione, si assegna un valore ad un elemento non definito, la matrice (o vettore) viene ridimensionata in modo automatico.



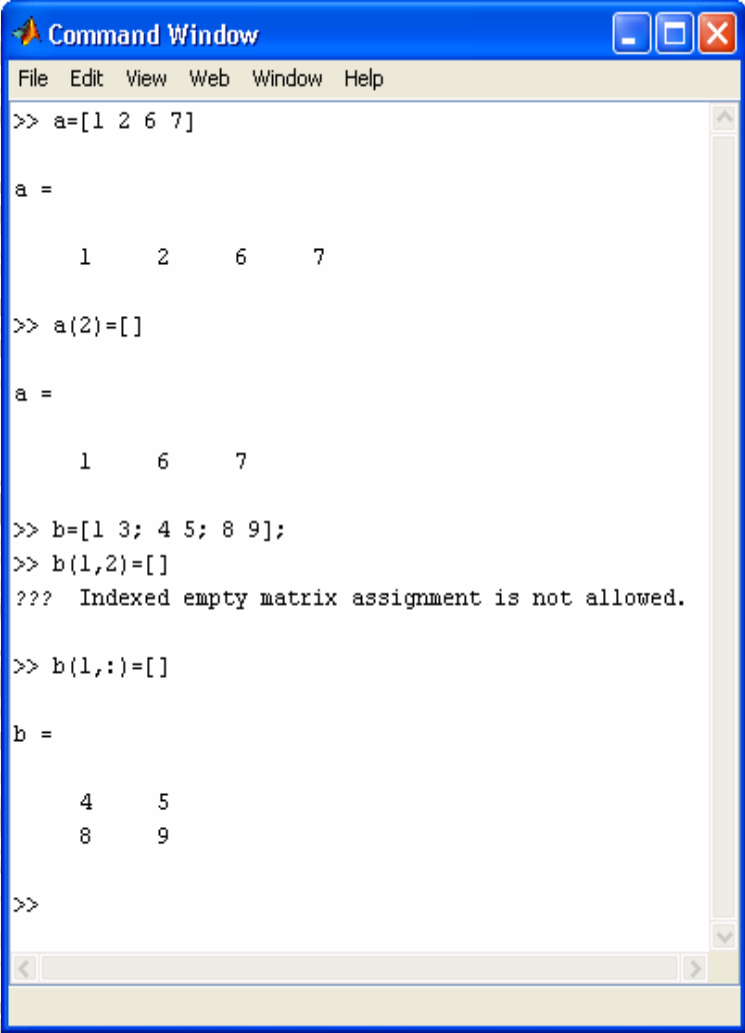
```
Command Window
File Edit View Web Window Help
>> a=[1 2 6 7]
a =
    1     2     6     7
>> a(7)=2
a =
    1     2     6     7     0     0     2
>> b=[1 3; 4 5; 8 9];
>> b(3,3)=10
b =
    1     3     0
    4     5     0
    8     9    10
>> |
```

Eliminazione di un elemento

Per cancellare un elemento di una matrice si usa l'operatore `[]`.

In una matrice non è possibile eliminare un singolo elemento, ma occorre eliminare un'intera dimensione (o una riga o una colonna).

La matrice viene ridimensionata in modo automatico.

A screenshot of the MATLAB Command Window. The window has a blue title bar with the text 'Command Window' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Web', 'Window', and 'Help'. The main area shows the following commands and outputs:

```
>> a=[1 2 6 7]

a =

     1     2     6     7

>> a(2)=[]

a =

     1     6     7

>> b=[1 3; 4 5; 8 9];
>> b(1,2)=[]
??? Indexed empty matrix assignment is not allowed.

>> b(1,:)=[]

b =

     4     5
     8     9

>>
```

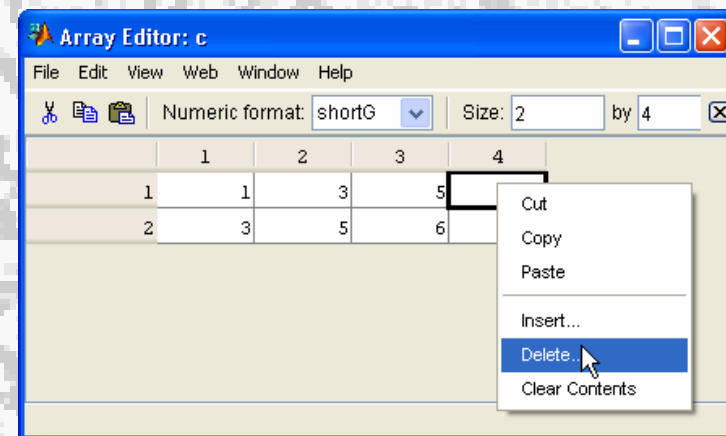
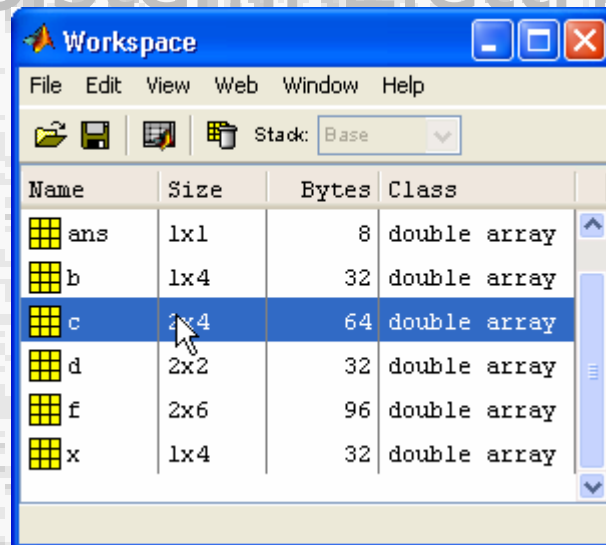
Editor di array

MATLAB mette a disposizione un editor (**Array Editor**), un'interfaccia grafica appositamente ideata per manipolare (modificare, cancellare, aggiungere) gli elementi di un array.

Vi si accede dal Workspace Browser, facendo doppio clic sulla variabile da modificare, oppure dalla Command Window con il comando **open**.

Sintassi

open('var')

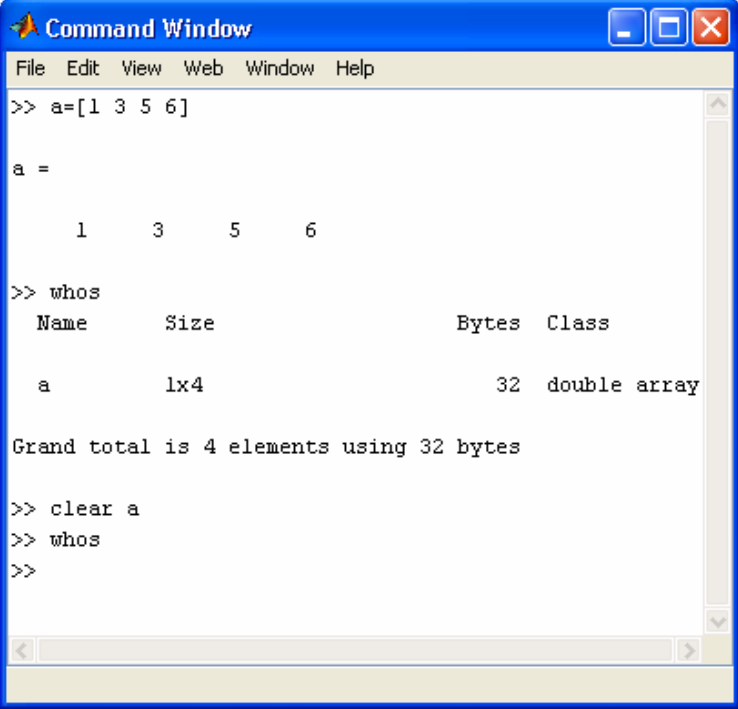


Cancellazione di una matrice

Si può cancellare una variabile nel Workspace o da riga di comando con la funzione **clear**.

Sintassi

- *clear nome_var*
- *clear all*
(cancella tutte le variabili)



The screenshot shows a MATLAB Command Window with the following content:

```
File Edit View Web Window Help
>> a=[1 3 5 6]

a =

     1     3     5     6

>> whos

  Name      Size      Bytes  Class

  a         1x4         32  double array

Grand total is 4 elements using 32 bytes

>> clear a
>> whos
>>
```

Matrici speciali

MATLAB offre alcuni comandi per creare matrici particolari:

- Matrice Identità $n \times n$

Funzione **eye**

Sintassi

nome_var=eye(n)

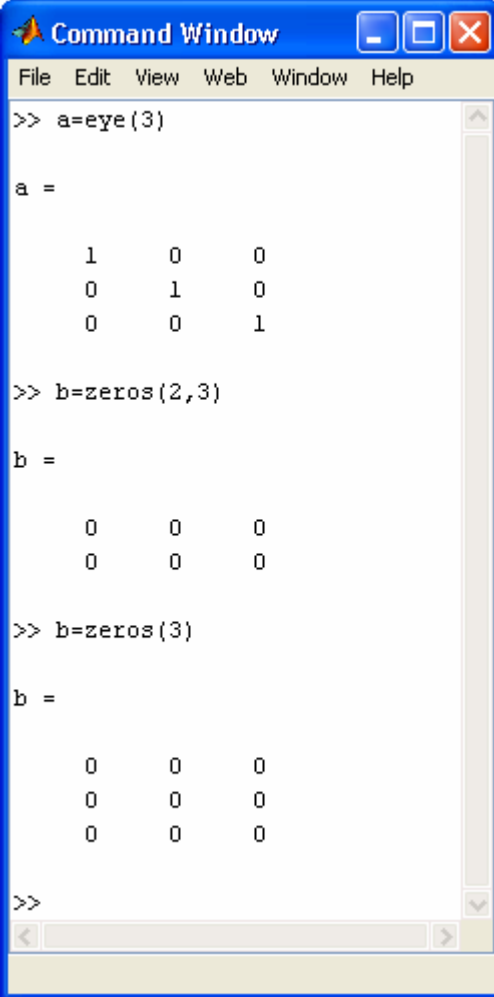
- Matrice di zero $n \times m$

Funzione **zeros**

Sintassi

nome_var=zeros(n)

nome_var=zeros(n,m)



```
Command Window
File Edit View Web Window Help
>> a=eye(3)

a =

     1     0     0
     0     1     0
     0     0     1

>> b=zeros(2,3)

b =

     0     0     0
     0     0     0

>> b=zeros(3)

b =

     0     0     0
     0     0     0
     0     0     0

>>
```

Matrici speciali

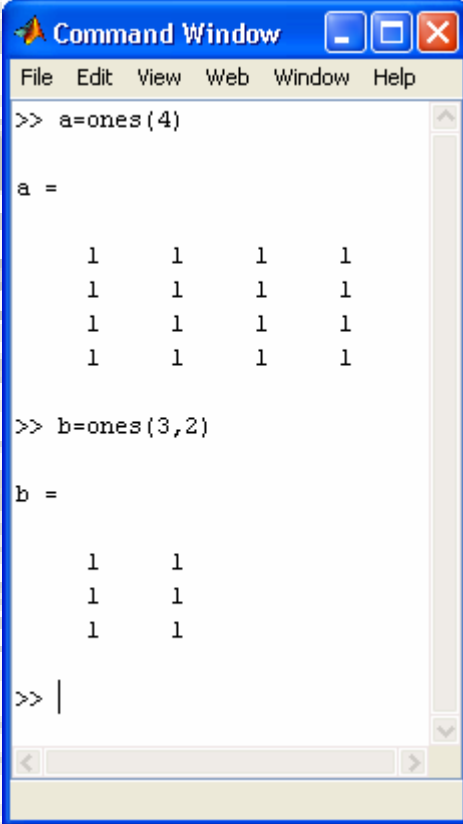
Matrice $n \times m$ di elementi uguali a 1.

Funzione **ones**

Sintassi

nome_var=ones(n) mat. quadrata

nome_var=ones(n,m)



```
Command Window
File Edit View Web Window Help
>> a=ones(4)

a =

     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1

>> b=ones(3,2)

b =

     1     1
     1     1
     1     1

>> |
```

Dimensione di una matrice

Si possono determinare le dimensioni di un vettore o di una matrice con le funzioni **size** e **length** (solo per i vettori).

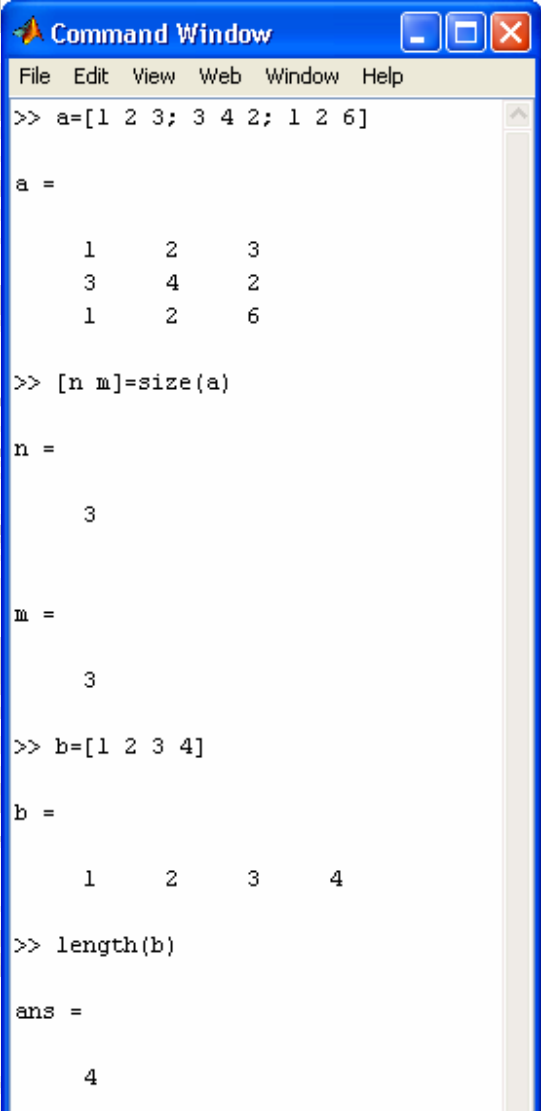
Sintassi

- $[n,m]=\text{size}(\text{var_matrice})$

n = numero di righe

m = numero di colonne

- $n=\text{length}(\text{var_array})$



```
Command Window
File Edit View Web Window Help
>> a=[1 2 3; 3 4 2; 1 2 6]

a =

     1     2     3
     3     4     2
     1     2     6

>> [n m]=size(a)

n =

     3

m =

     3

>> b=[1 2 3 4]

b =

     1     2     3     4

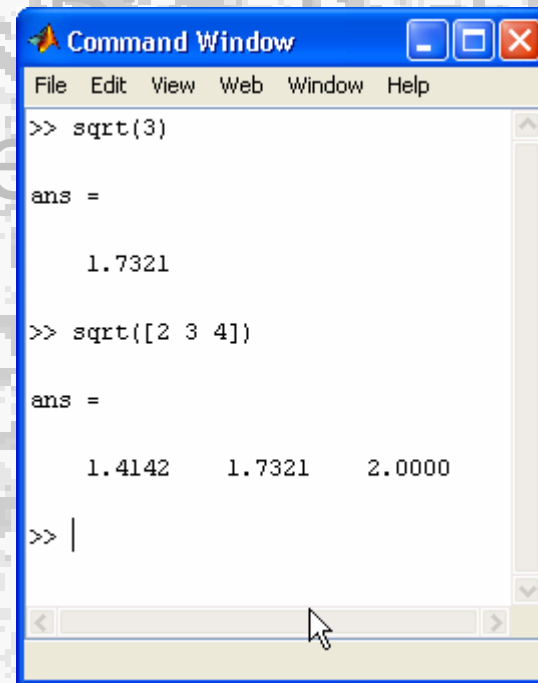
>> length(b)

ans =

     4
```

Operazioni scalari

Per operazioni scalari non si intendono solo quelle in cui gli operandi sono scalari, ma tutte quelle operazioni in cui gli operandi sono trattati come scalari. Ovvero sono operatori che si possono applicare a vettori e matrici e il risultato è lo stesso che si ottiene applicando lo stesso operatore ad ogni singolo elemento del vettore.



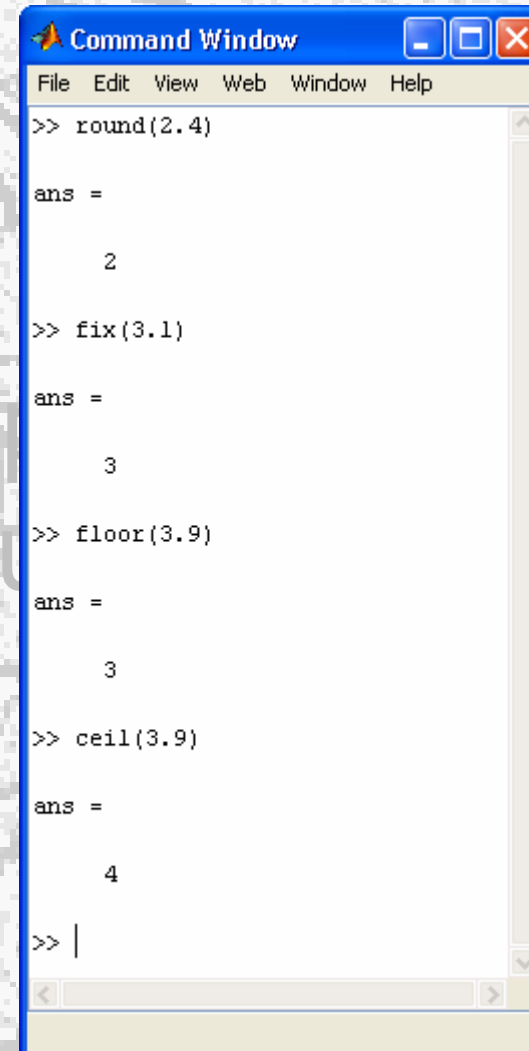
```
Command Window
File Edit View Web Window Help
>> sqrt(3)
ans =
    1.7321
>> sqrt([2 3 4])
ans =
    1.4142    1.7321    2.0000
>> |
```

- Operazioni aritmetiche

+	somma
-	differenza
*	prodotto
/	divisione
^	potenza
sqrt	radice quadrata

- Arrotondamento

round	all'intero più vicino
fix	verso lo zero
floor	per difetto all'intero più vicino
ceil	per eccesso all'intero più vicino



```
Command Window
File Edit View Web Window Help
>> round(2.4)
ans =
     2
>> fix(3.1)
ans =
     3
>> floor(3.9)
ans =
     3
>> ceil(3.9)
ans =
     4
>> |
```

- Approssimazioni razionali

rem resto di una divisione

rat espansione razionale

rats approssimazione razionale

- Operatori esponenziali e logaritmici

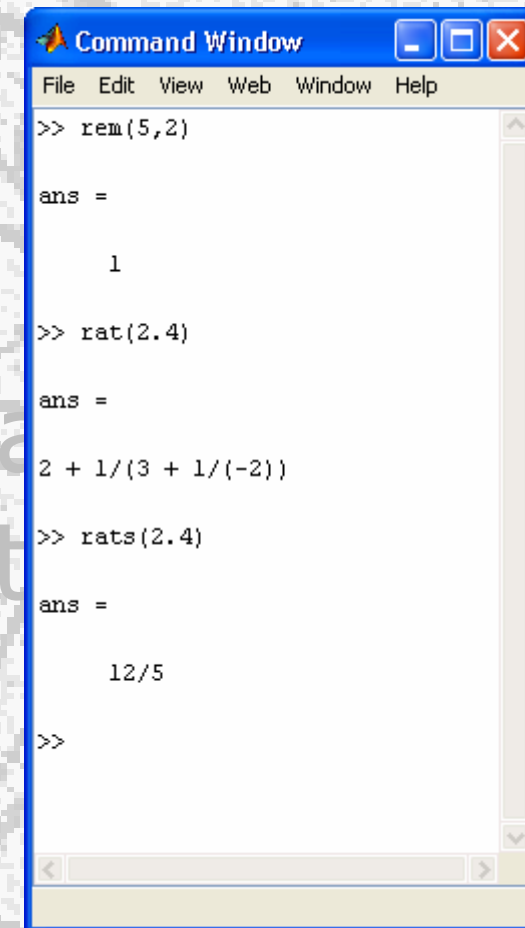
pow2 esponenziale base 2

exp esponenziale in base e

log logaritmo naturale

log2 logaritmo in base 2

log10 logaritmo in base 10

A screenshot of the MATLAB Command Window. The window has a blue title bar with the text 'Command Window' and standard window control buttons. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Web', 'Window', and 'Help'. The main area of the window shows the following text:

```
>> rem(5,2)

ans =

     1

>> rat(2.4)

ans =

2 + 1/(3 + 1/(-2))

>> rats(2.4)

ans =

    12/5

>>
```

- Funzioni trigonometriche dirette

sin

cos

tan

L'argomento deve essere in
radianti

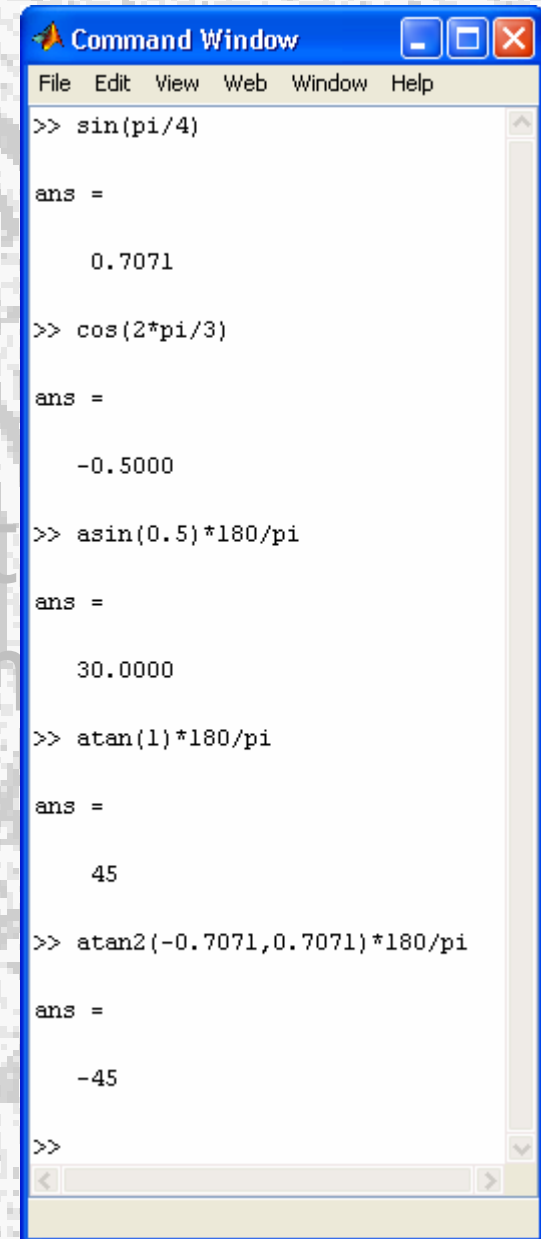
- Funzioni trigonometriche inverse

asin

acos

atan

atan2 arcotangente su 4 quadranti

A screenshot of the MATLAB Command Window. The window has a blue title bar with the text 'Command Window' and standard window control buttons. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Web', 'Window', and 'Help'. The main area contains a series of commands and their outputs. The commands are: 'sin(pi/4)', 'cos(2*pi/3)', 'asin(0.5)*180/pi', 'atan(1)*180/pi', and 'atan2(-0.7071,0.7071)*180/pi'. The outputs are: 'ans = 0.7071', 'ans = -0.5000', 'ans = 30.0000', 'ans = 45', and 'ans = -45'. The window ends with '>>' on the last line.

```
Command Window
File Edit View Web Window Help
>> sin(pi/4)
ans =
    0.7071
>> cos(2*pi/3)
ans =
   -0.5000
>> asin(0.5)*180/pi
ans =
   30.0000
>> atan(1)*180/pi
ans =
    45
>> atan2(-0.7071,0.7071)*180/pi
ans =
   -45
>>
```

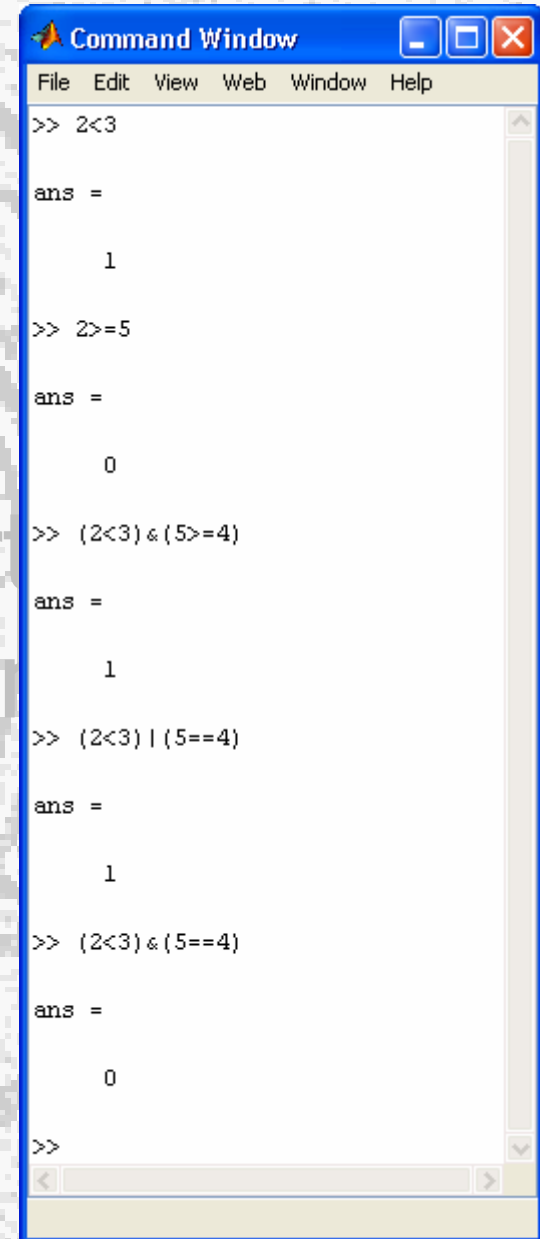
- Operatori di relazioni

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

- Operatori logici

- & and
- | or
- xor or esclusivo
- ~ not

Il risultato del test è 1 se vero e 0 (zero) se falso.



```
Command Window
File Edit View Web Window Help
>> 2<3
ans =
    1
>> 2>=5
ans =
    0
>> (2<3) & (5>=4)
ans =
    1
>> (2<3) | (5==4)
ans =
    1
>> (2<3) & (5==4)
ans =
    0
>>
```

Applicazione degli operatori aritmetici a matrici e vettori

La somma (+) e la differenza (-) si possono eseguire solo se le matrici hanno le stesse dimensioni.

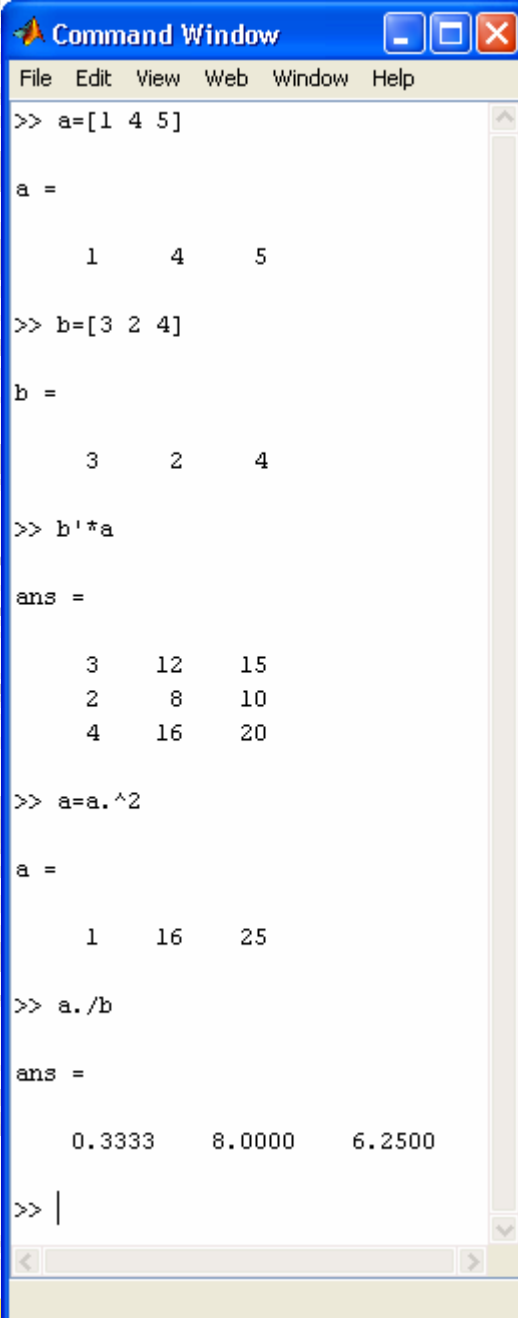
Per il prodotto (*) vale la regola del prodotto tra matrici: il numero di colonne della prima matrice deve essere uguale al numero di righe della seconda.

Se si vuole operare sui singoli elementi del vettore o della matrice occorre utilizzare l'operatore (.).

- . * prodotto tra elementi corrispondenti di due matrici;
- ./ quoziente tra elementi corrispondenti di due matrici;
- . ^ elevazione a potenza degli elementi di una matrice.

Esempi

Università di Pisa - Dipartimento
Sistemi Elettrici e Autor



A screenshot of the MATLAB Command Window interface. The window has a blue title bar with the text 'Command Window' and standard window control buttons. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Web', 'Window', and 'Help'. The main area contains the following MATLAB commands and their outputs:

```
>> a=[1 4 5]

a =

     1     4     5

>> b=[3 2 4]

b =

     3     2     4

>> b'*a

ans =

     3    12    15
     2     8    10
     4    16    20

>> a=a.^2

a =

     1    16    25

>> a./b

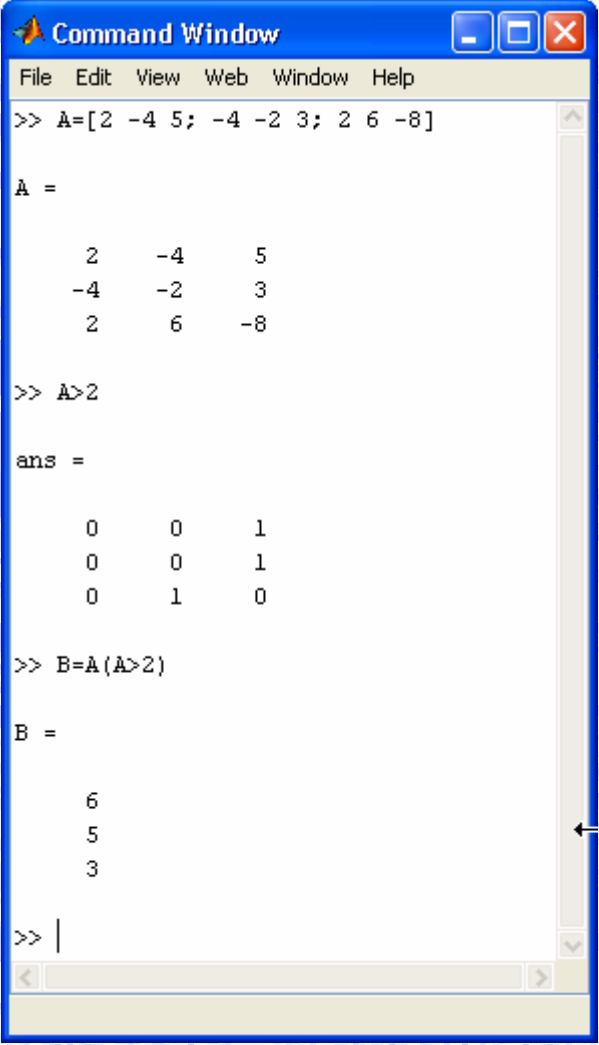
ans =

    0.3333    8.0000    6.2500

>> |
```

Operazioni logiche e relazionali applicate a matrici

MATLAB esegue il test relazionale o logico su ogni singolo elemento della matrice e quindi il risultato è una matrice delle stesse dimensioni i cui elementi sono 1 (se per l'elemento corrispon-dente il test è vero) o 0 (se per l'elemento corrispondente il test è falso).



```
Command Window
File Edit View Web Window Help
>> A=[2 -4 5; -4 -2 3; 2 6 -8]

A =

     2     -4     5
    -4     -2     3
     2      6    -8

>> A>2

ans =

     0     0     1
     0     0     1
     0     1     0

>> B=A(A>2)

B =

     6
     5
     3

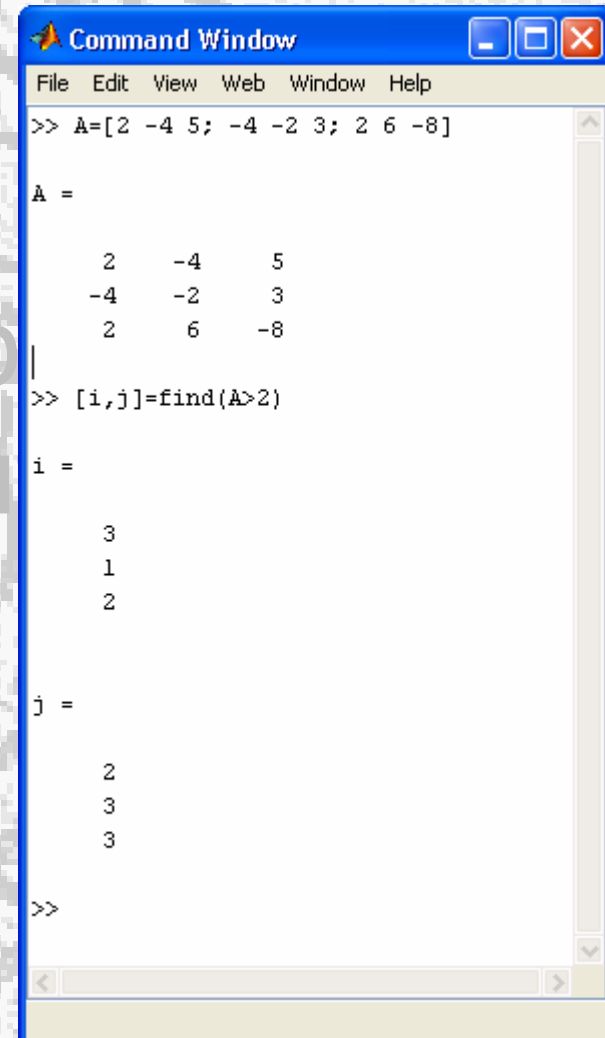
>> |
```

Operazioni logiche e relazionali applicate a matrici

La funzione **find** permette di trovare gli indici degli elementi che soddisfano un test logico o relazionale

Sintassi

$[i,j]=\text{find}(\text{var_test})$



```
Command Window
File Edit View Web Window Help
>> A=[2 -4 5; -4 -2 3; 2 6 -8]

A =

     2     -4     5
    -4     -2     3
     2      6    -8

>> [i,j]=find(A>2)

i =

     3
     1
     2

j =

     2
     3
     3

>>
```

Trasposta di una matrice

Se A è una matrice $n \times m$, la trasposta A^T è la matrice $m \times n$ con

$$a_{i,j}^T = a_{j,i}$$

- Operatore (')

Se la matrice è formata da numeri complessi la traspone e ne calcola il coniugato.

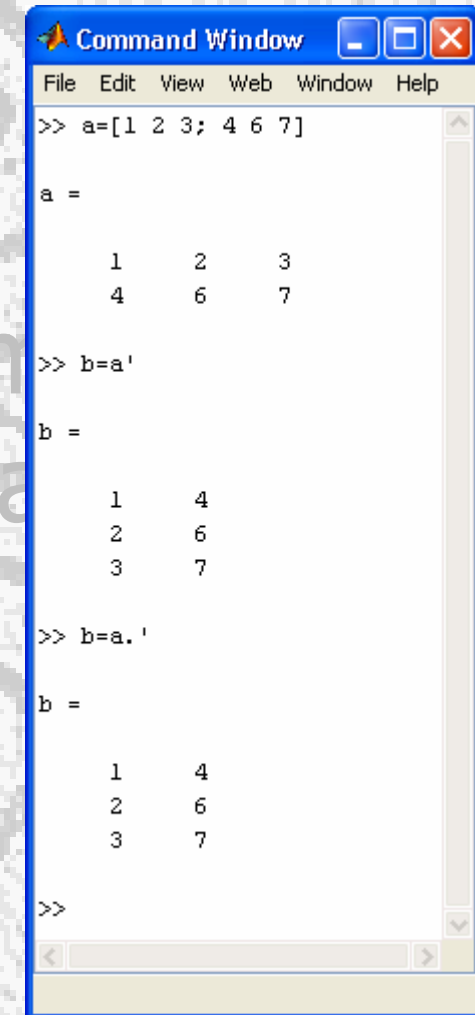
Sintassi

var_a = var_mat'

- Operatore (.')

Sintassi

var_a = var_mat.'



```

Command Window
File Edit View Web Window Help
>> a=[1 2 3; 4 6 7]

a =

     1     2     3
     4     6     7

>> b=a'

b =

     1     4
     2     6
     3     7

>> b=a.'

b =

     1     4
     2     6
     3     7

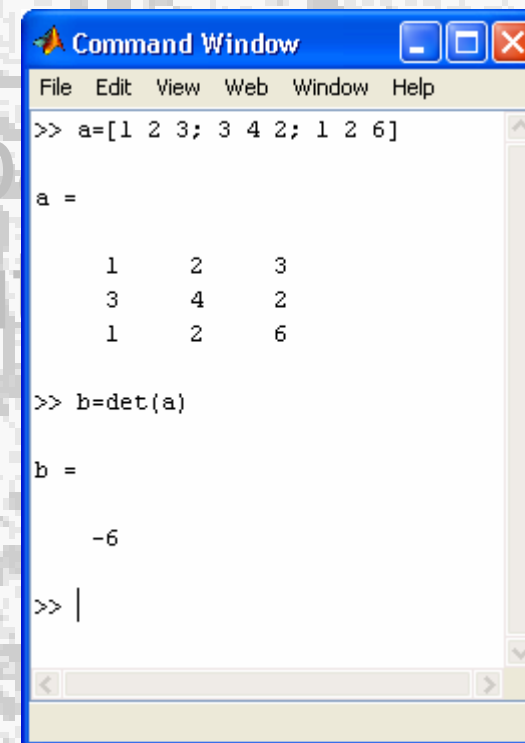
>>
  
```

Determinante di una matrice

Se A è una matrice $n \times n$ invertibile il suo determinante si ottiene con la funzione **det**.

Sintassi

var_a=det(var_mat)



```
Command Window
File Edit View Web Window Help
>> a=[1 2 3; 3 4 2; 1 2 6]

a =

     1     2     3
     3     4     2
     1     2     6

>> b=det(a)

b =

    -6

>> |
```

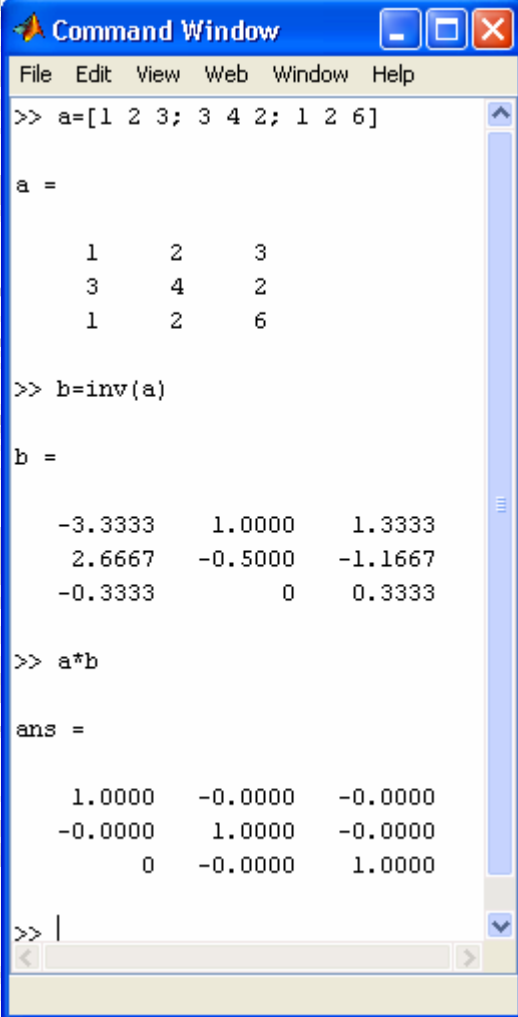
Inversa di una matrice

Se A è una matrice $n \times n$ invertibile ($\det \neq 0$), l'inversa A^{-1} è la matrice $n \times n$ tale che $A A^{-1} = A^{-1} A = I$

Funzione **inv**

Sintassi

var_a = inv(var_mat)

A screenshot of the MATLAB Command Window. The window has a blue title bar with the text 'Command Window' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Web', 'Window', and 'Help'. The main area shows the following commands and outputs:

```
>> a=[1 2 3; 3 4 2; 1 2 6]

a =

     1     2     3
     3     4     2
     1     2     6

>> b=inv(a)

b =

   -3.3333    1.0000    1.3333
    2.6667   -0.5000   -1.1667
   -0.3333     0      0.3333

>> a*b

ans =

    1.0000   -0.0000   -0.0000
   -0.0000    1.0000   -0.0000
         0   -0.0000    1.0000

>> |
```

Valore massimo

- Massimo modulo di un array

Funzione **max**

Sintassi

$var_a = max(var_b)$

Nel caso di una matrice il risultato è un vettore riga i cui elementi sono il massimo modulo di ogni colonna della matrice.

$[var_a, I] = max(var_mat)$

I è un vettore che contiene gli indici del massimo modulo di ogni colonna.

Il max modulo di una matrice si trova con il comando *$max(max(var_mat))$*

```

Command Window
File Edit View Web Window Help
>> b=[1 2 3 4]

b =

     1     2     3     4

>> max(b)

ans =

     4

>> a=[1 2 3; 3 4 2; 1 2 6]

a =

     1     2     3
     3     4     2
     1     2     6

>> [m,I]=max(a)

m =

     3     4     6

I =

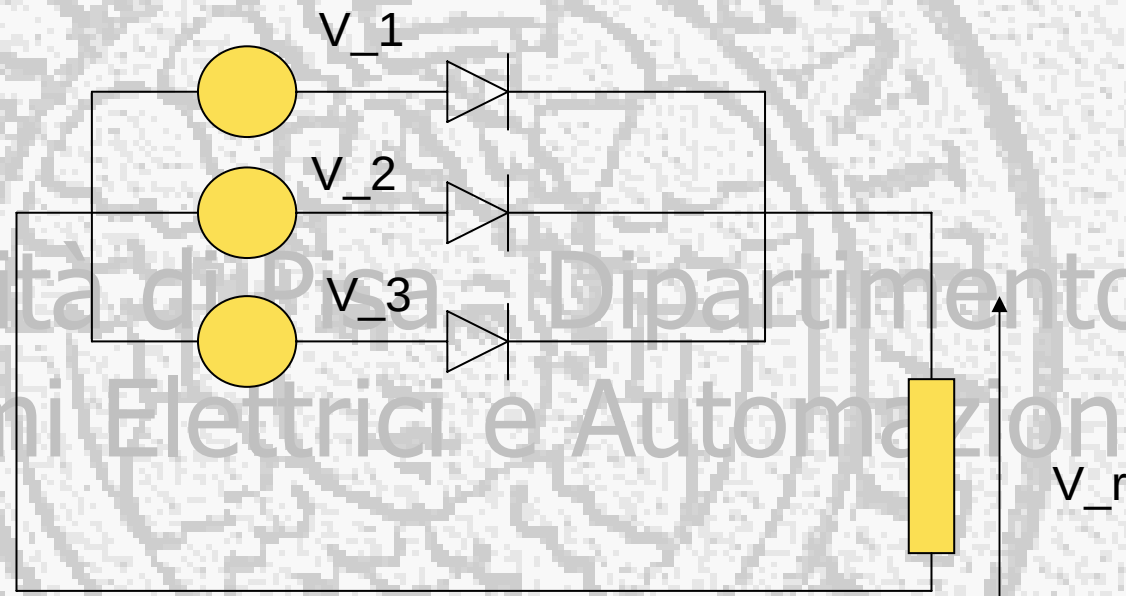
     2     2     3

>> max(max(a))

ans =

     6
  
```

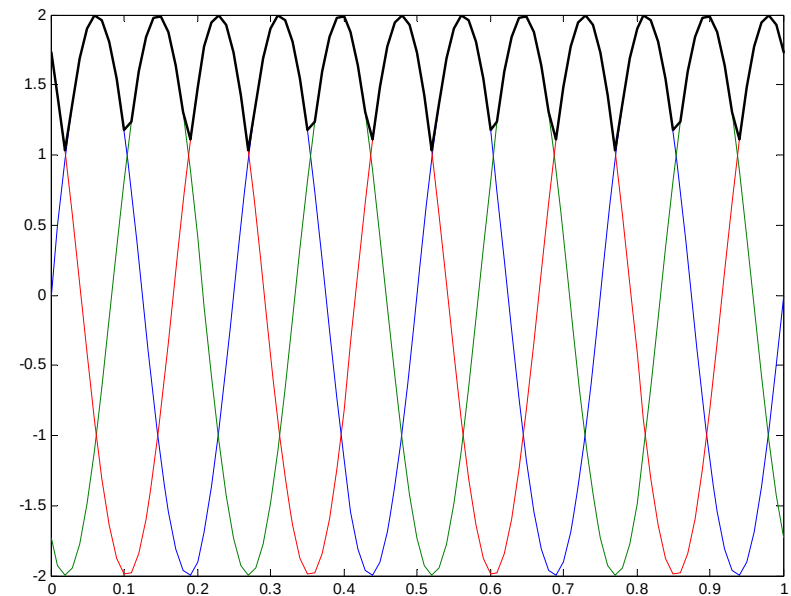
Esempio: raddrizzatore trifase ad anodo comune



$$\begin{cases} v_1 = A \sin(2\pi ft) \\ v_2 = A \sin(2\pi ft - 2\pi/3) \\ v_3 = A \sin(2\pi ft - 4\pi/3) \end{cases} \quad V_r = \max(v_1, v_2, v_3)$$

Esempio: raddrizzatore trifase ad anodo comune

```
Command Window
File Edit View Web Window Help
>> % si definisce una scala dei tempi
>> t=0:.01:1;
>> % si definisce la terna simmetrica di tensioni
>> v1=2*sin(2*pi*4*t);
>> v2=2*sin(2*pi*4*t-2*pi/3);
>> v3=2*sin(2*pi*4*t-4*pi/3);
>> v=[v1; v2; v3];
>> % si determina la tensione raddrizzata
>> v_r=max(v);
>> plot(t,v,t,v_r,'k')
>> |
```



Valore minimo

- Minimo modulo di un array

Funzione **min**

Sintassi

$var_a = min(var_b)$

Nel caso di una matrice il risultato è un vettore riga i cui elementi sono il minimo modulo di ogni colonna della matrice.

$[var_a, I] = min(var_mat)$

I è un vettore che contiene gli indici del minimo modulo di ogni colonna.

Il minimo di una matrice si trova con il comando:

$min(min(var_a))$

```

Command Window
File Edit View Web Window Help
>> b=[1 2 3 4]

b =

     1     2     3     4

>> min(b)

ans =

     1

>> a=[1 2 3; 3 4 2; 1 2 6]

a =

     1     2     3
     3     4     2
     1     2     6

>> [m,I]=min(a)

m =

     1     2     2

I =

     1     1     2

>> min(min(a))

ans =

     1
  
```

Valore medio

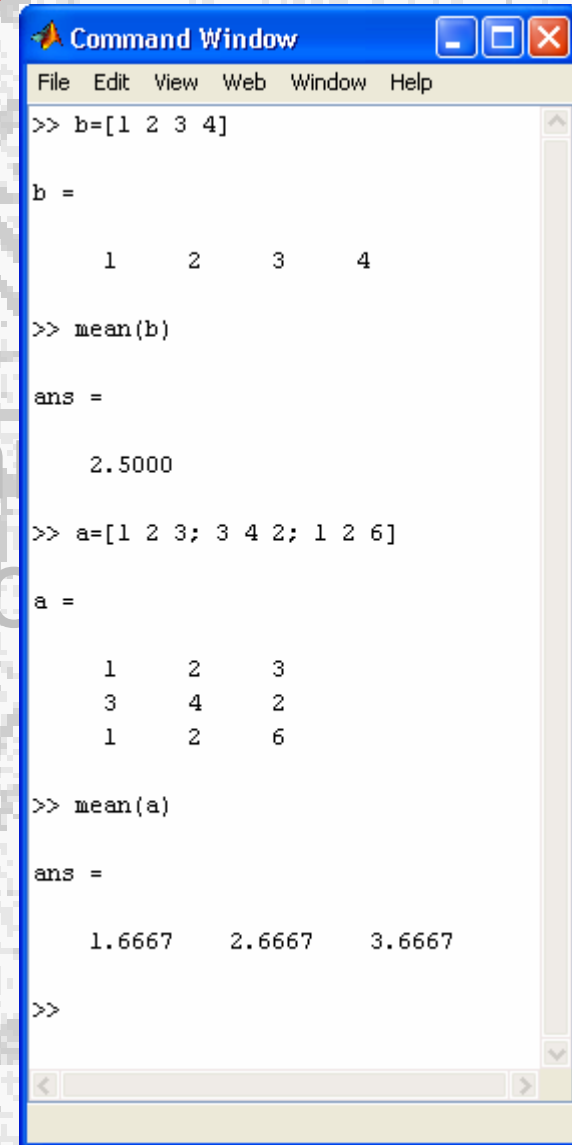
- Valore medio di un array

Funzione **mean**

Sintassi

var_a = mean(var_b)

Nel caso di una matrice il risultato è un vettore riga i cui elementi sono il valore medio di ogni colonna della matrice.



```
Command Window
File Edit View Web Window Help

>> b=[1 2 3 4]

b =

     1     2     3     4

>> mean(b)

ans =

     2.5000

>> a=[1 2 3; 3 4 2; 1 2 6]

a =

     1     2     3
     3     4     2
     1     2     6

>> mean(a)

ans =

     1.6667     2.6667     3.6667

>>
```

Somma

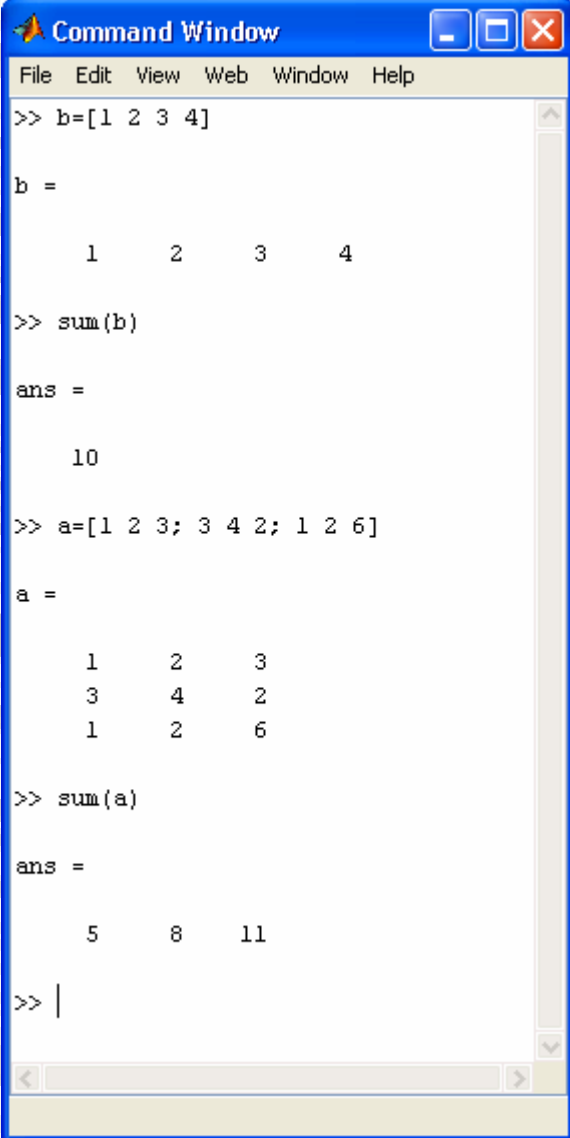
- Somma degli elementi di un array.

Funzione **sum**

Sintassi

$var_a = sum(var_b)$

Nel caso di una matrice il risultato è un vettore riga i cui elementi sono la somma di ogni colonna della matrice.



```
Command Window
File Edit View Web Window Help

>> b=[1 2 3 4]

b =

     1     2     3     4

>> sum(b)

ans =

    10

>> a=[1 2 3; 3 4 2; 1 2 6]

a =

     1     2     3
     3     4     2
     1     2     6

>> sum(a)

ans =

     5     8    11

>> |
```

Ordinamento

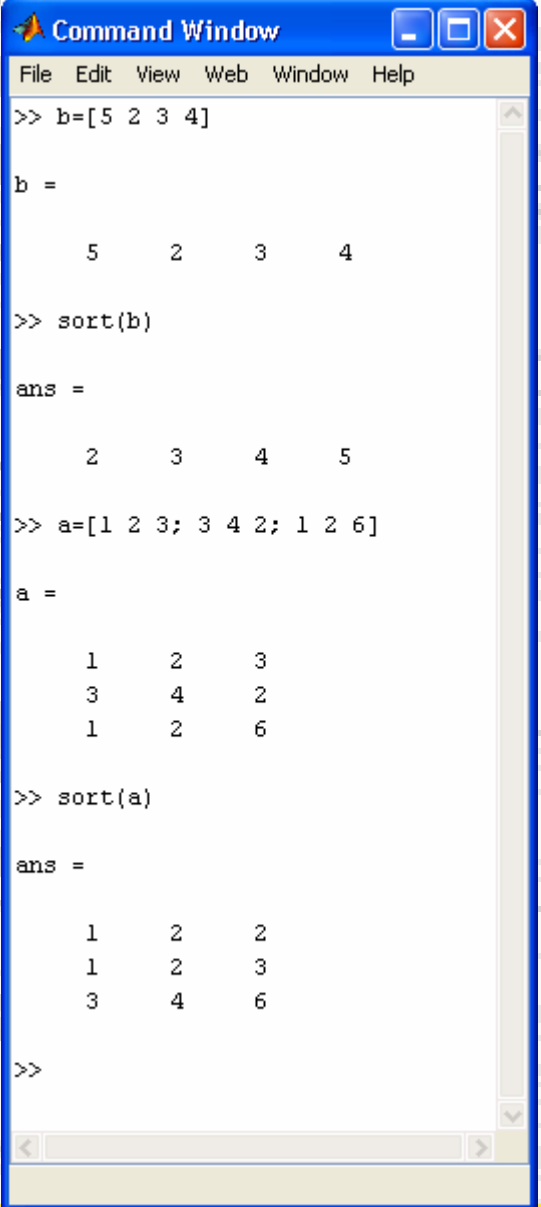
- Ordinamento degli elementi di un array

Funzione **sort**

Sintassi

var_a = sort(var_b)

nel caso di una matrice gli elementi sono ordinati per colonna. Nel caso di numeri complessi, viene ordinata la matrice `abs(var_b)`



```
Command Window
File Edit View Web Window Help

>> b=[5 2 3 4]

b =

     5     2     3     4

>> sort(b)

ans =

     2     3     4     5

>> a=[1 2 3; 3 4 2; 1 2 6]

a =

     1     2     3
     3     4     2
     1     2     6

>> sort(a)

ans =

     1     2     2
     1     2     3
     3     4     6

>>
```

Norma di un vettore

- Norma di un vettore

Funzione **norm**

Sintassi

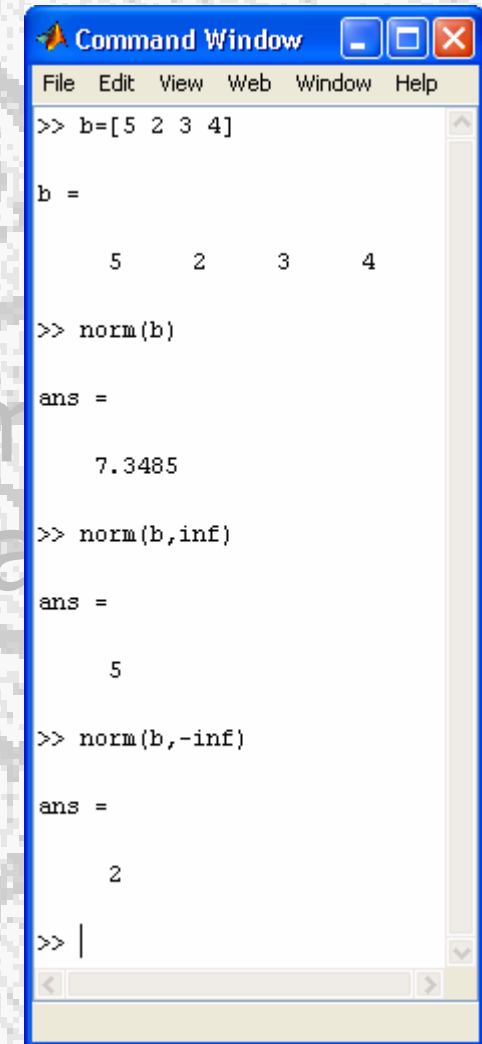
var_a = norm(var_b, p)

Dove:

se $p=[]$, var_a è la norma euclidea;

se $p=inf$, var_a è il massimo modulo;

se $p=-inf$, var_a è il minimo modulo.

A screenshot of the MATLAB Command Window. The window has a blue title bar with the text 'Command Window' and standard window controls. The menu bar includes 'File', 'Edit', 'View', 'Web', 'Window', and 'Help'. The command history shows: 1. 'b=[5 2 3 4]' followed by the display of the vector 'b' as '5 2 3 4'. 2. 'norm(b)' followed by the result 'ans = 7.3485'. 3. 'norm(b,inf)' followed by the result 'ans = 5'. 4. 'norm(b,-inf)' followed by the result 'ans = 2'. The prompt '>>' is visible at the bottom.

```
>> b=[5 2 3 4]
b =
    5    2    3    4
>> norm(b)
ans =
    7.3485
>> norm(b,inf)
ans =
     5
>> norm(b,-inf)
ans =
     2
>> |
```

Norma di una matrice

- Norma di una matrice

Funzione **norm**

Sintassi

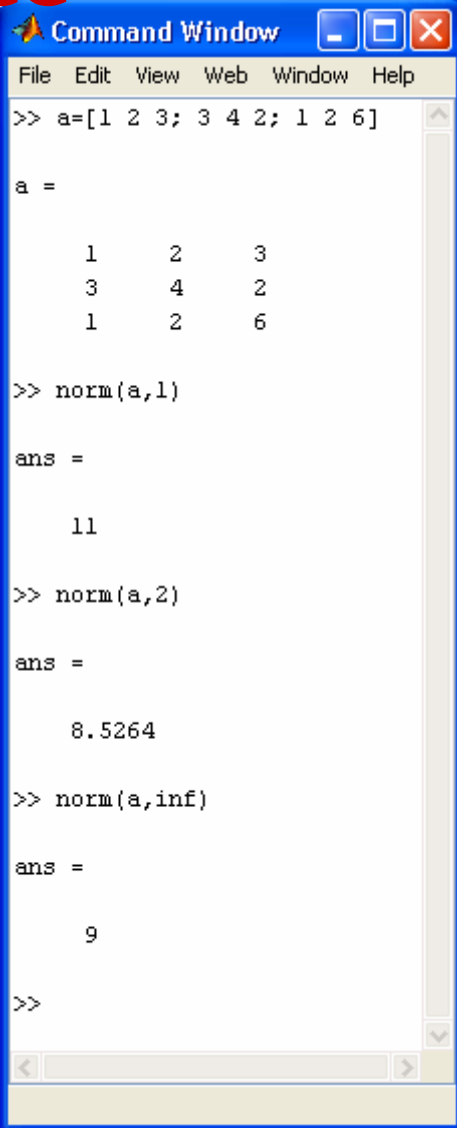
var_a = norm(var_mat, p)

dove:

se , var_a è la massima somma per colonne;

se $p=2$, var_a è il max valore singolare;

se $p=\text{inf}$, var_a è il max per righe.



```
Command Window
File Edit View Web Window Help
>> a=[1 2 3; 3 4 2; 1 2 6]

a =

     1     2     3
     3     4     2
     1     2     6

>> norm(a,1)

ans =

    11

>> norm(a,2)

ans =

    8.5264

>> norm(a,inf)

ans =

     9

>>
```

Autovalori e Autovettori

- Polinomio caratteristico $\det(A-\lambda I)$

funzione **poly**

Sintassi

$var_a = poly(var_mat)$

- Autovalori e autovettori

$Ax = \lambda x$

λ = autovalore x = autovettore

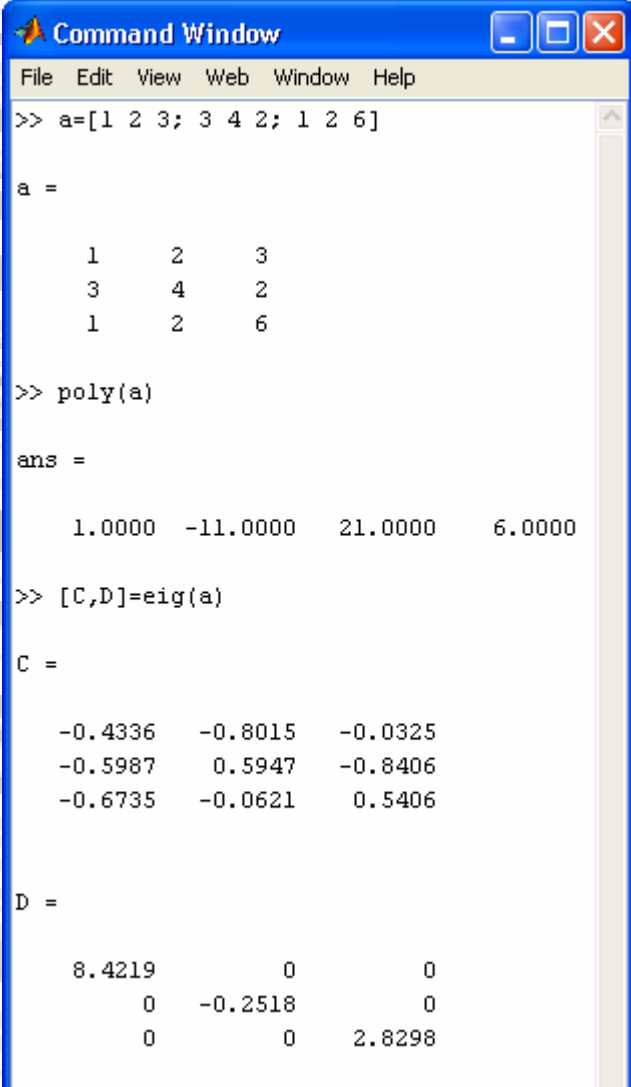
funzione **eig**

Sintassi

$[A, B] = eig(var_mat)$

A matrice degli autovettori;

B matrice diagonale degli autovalori.



```

Command Window
File Edit View Web Window Help
>> a=[1 2 3; 3 4 2; 1 2 6]

a =

     1     2     3
     3     4     2
     1     2     6

>> poly(a)

ans =

     1.0000    -11.0000    21.0000     6.0000

>> [C,D]=eig(a)

C =

   -0.4336   -0.8015   -0.0325
   -0.5987    0.5947   -0.8406
   -0.6735   -0.0621    0.5406

D =

    8.4219         0         0
         0   -0.2518         0
         0         0    2.8298
  
```