Lezione 1

Avvio di Matlab

Matlab ≡ *Matrix Laboratory*

Matlab è un ambiente di sviluppo per il calcolo numerico e simbolico che implementa tutte le operazioni definite in algebra matriciale più operazioni elemento ad elemento.

Equazioni di II grado

Soluzione Simbolica

$$ax^2 + bx + c = 0 \rightarrow x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Soluzione Numerica

$$x^2 - 3x + 2 = 0 \rightarrow \begin{cases} x_1 = 1 \\ x_2 = 2 \end{cases}$$

NOTA BENE: in questo corso non verranno trattate le funzioni per il calcolo simbolico

Modalità Funzionamento

Matlab può funzionare in modalità *console*, o come interprete degli *Scripts*

MATLAB

MODALITÀ CONSOLE

Matlab si comporta come una normale calcolatrice programmabile, i comandi devono essere inseriti in sequenza, uno dopo l'altro e vengono immediatamente eseguiti

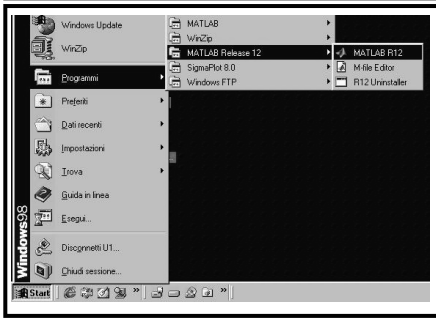
INTERPRETE

I comandi Matlab possono essere scritti in un file di testo ASCII, detti ***Scripts***, e possono essere eseguiti (*interpretati*) da Matlab in un secondo momento.

Eseguire Matlab

Per mandare il programma Matlab in esecuzione si può:

Fare doppio click con il tasto destro del mouse dopo aver posizionato il cursore sull'icona del programma che si trova sul desktop



1. Attivare il menu Start (o Avvio) cliccando con il tasto sinistro del mouse sul bottone relativo nella *taskbar* del *desktop;*
2. cercare nel menu "*Programmi*" il la voce relativa all'ambiente Matlab;
3. cliccare sul nome del programma Matlab.

La Finestra di Comando

Se Matlab è nella conformazione "*solo finestra di comando*" allora appare come in figura:

Barra di Sistema

I tre pulsanti sulla destra servono a:
- ridurre a icona
- ingrandire a tutto schermo
- chiudere il programma

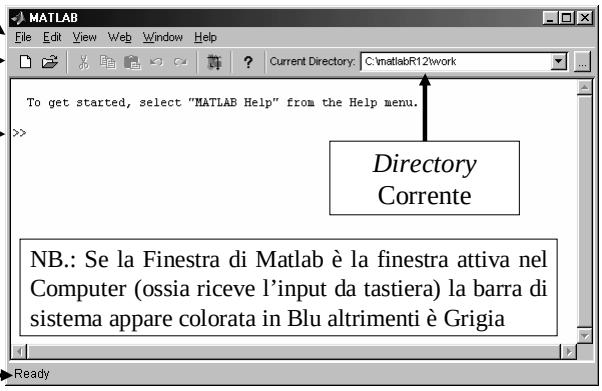
Menù Principale

Barra degli Strumenti

Prompt dei comandi

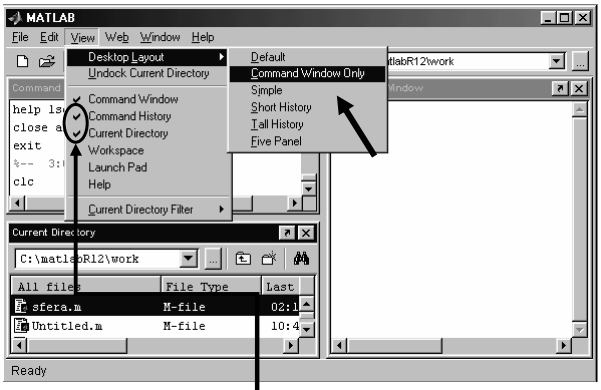
è il punto di inserimento dove digitare tramite tastiera i comandi da far eseguire in modalità console.

Barra di stato



Directory Corrente

NB.: Se la Finestra di Matlab è la finestra attiva nel Computer (ossia riceve l'input da tastiera) la barra di sistema appare colorata in Blu altrimenti è Grigia



Se Matlab non si trova nella conformazione: *Solo Finestra di Comando*, allora si selezioni dal menu

View

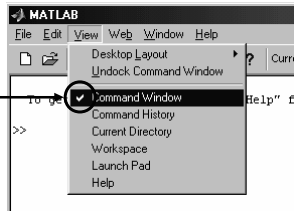
la voce

Desktop Layout

e quindi l'opzione

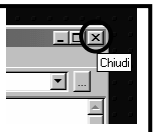
Command Window Only

Questo equivale a deselezionare (scompare il segno di spunta in nero) tutte le altre finestre dal menu **View** al di fuori della Finestra di Comando



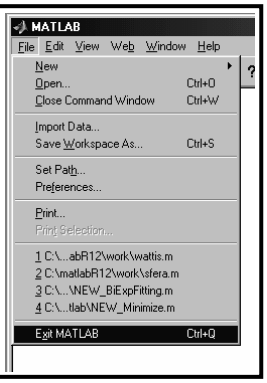
4 modi per chiudere Matlab

1 Cliccare con il tasto sx del mouse sul bottoncino ad x della barra di sistema

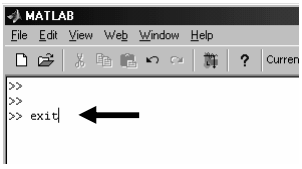


2

Selezionare la voce *Exit* del menu *File* nella barra del menu principale della finestra di comando



3



Digitare il comando *exit* seguito dal tasto invio (*Enter*) al *prompt* della finestra di comando

4 Premere contemporaneamente i tasti CTRL e Q sulla tastiera quando la finestra di comando è la finestra attiva

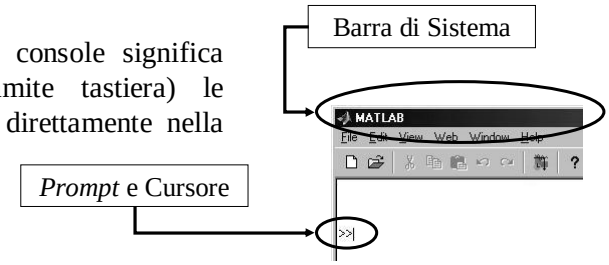
Verifica

Lo studente deve essere in grado di:

- Accendere il Computer
- Lanciare Matlab nella conformazione *Command Window Only*
- Ingrandire/ridurre la finestra di Matlab
- Uscire da Matlab
- Spegnere il Computer

Lavorare in modalità *console*

Lavorare in Modalità console significa digitare (inserire tramite tastiera) le istruzioni da eseguire direttamente nella finestra di comando.



⇨ Questo è possibile solo se la finestra di Matlab è la finestra attiva ossia se la sua Barra di Sistema è *evidenziata* (se non sono state cambiate le impostazioni di *default* deve apparire di colore blu e non grigio)

⇨ Per rendere attiva la finestra di Matlab (quando la barra di sistema è di color grigio) basta cliccarci su con il tasto sinistro del Mouse, il cursore inizierà a lampeggiare affianco al prompt e la finestra sarà pronta a ricevere l'input da tastiera.

Modalità *console*

In modalità console Matlab può essere usato come una qualsiasi calcolatrice programmabile per eseguire calcoli fra valori numerici scritti in diverso formato. Una volta inserita l'espressione da calcolare bisogna digitare il tasto ENTER (o INVIO) perché Matlab effettui il calcolo:

1 Matlab eseguirà l'operazione e restituirà il risultato (*answer*=risposta) nelle righe seguenti

2 Quindi restituisce il prompt (>>) ed il cursore (|) per un altro inserimento

```
>> 6 * 5
ans =
      30
>>
```

NB.: E' possibile inserire o modificare espressioni per il calcolo solo sull'ultima riga della finestra di comando di Matlab le altre righe, precedentemente inserite, anche se visibili non sono editabili. Per poterle modificare vanno richiamate con i tasti FRECCIA SU (↑) FRECCIA GIU' (↓) della tastiera.

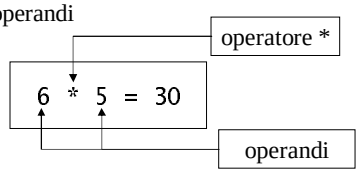
Operatori aritmetici

Per poter effettuare operazioni matematiche devono essere usati gli *operatori aritmetici binari* riportati di fianco.

```
^ Elevamento a Potenza
* Prodotto
/ Divisione
+ Somma
- Differenza
```

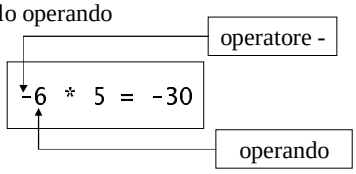
Operatore Binario

Un operatore si dice binario se opera su due operandi



Operatore Unario

Un operatore si dice unario se opera su un solo operando



Formati Numerici: input

I valori numerici costanti possono essere inseriti in Matlab come:

Gli interi sono valori numerici senza parte decimale

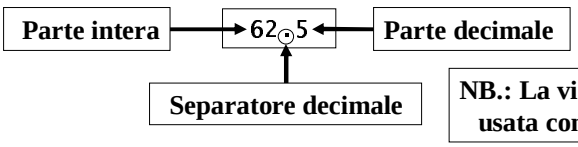
INTERI

```
>> 6 * 5
ans =
      30
```

I razionali *fixed point* sono valori numerici che presentano una parte intera ed una parte decimale separate da un punto: “.”

RAZIONALI
FIXED POINT

```
>> 62.5 * 5
ans =
      312.5000
```



NB.: La virgola “,” non può essere usata come separatore decimale

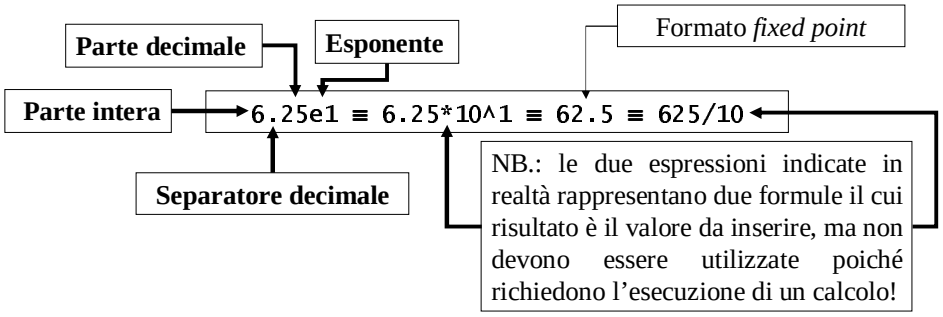
Formati Numerici: input

I valori razionali in formato *floating point* (formato esponenziale) presentano:

una solo cifra intera
nessuna o più cifre decimali
l'esponente della potenza del dieci da moltiplicare preceduta dal simbolo "e" o "E"

RAZIONALI FLOATING POINT

```
>> 6.5e1 * 5
ans =
        312.500
```



Esempi formati numerici

Formule Matematiche

Espressioni Matlab alternative

$1.2e3 \times \frac{5.2}{2}$ ⇨

```
>> 1.2e3*5.2/2
ans =
        3120
```

```
>> 1200*5.2/2
ans =
        3120
```

$25.04 \times 2.7 \cdot 10^{-4}$ ⇨

```
>> 25.04*2.7e-4
ans =
        0.0067608
```

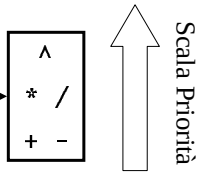
```
>> 2.504e1*2.7e-4
ans =
        0.0067608
```

$25.04 \times 2.7 \cdot 10^{-4} + 0.01$ ⇨

```
>> 2.504e1 * 2.7e-4 + 1e-2
ans =
        0.0167608
```

Priorità Operatori aritmetici

Nell'utilizzo degli operatori è sempre molto importante sapere quale è la scala delle priorità con cui essi vengono applicati nell'esecuzione di un calcolo.



L'operatore ^ elevamento a potenza ha una priorità maggiore dell'operatore prodotto * e viene applicato prima

```
>> 2^3*4
ans
      32
```

→ 8*4 → 32 Sequenza operazioni corretta

✗ 2^12 ✗ 4096

L'operatore / di divisione ha una priorità maggiore dell'operatore somma + e viene quindi applicato prima

```
>> 2+8/4
ans
      4
```

→ 2+2 → 4 Sequenza operazioni corretta

✗ 10/4 ✗ 2.5

L'operatore / di divisione e quello prodotto * hanno uguale priorità in questo caso le operazioni vengono eseguite da sinistra a destra

```
>> 2*8/4
ans
      4
```

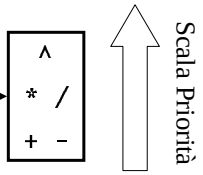
→ 16/4 → 4 Sequenza operazioni corretta

✗ 2*2 ✗ 4

N.B.: in questo caso il risultato sarebbe in entrambi i casi coincidente

Uso delle parentesi tonde

Le Parentesi tonde possono essere utilizzate come in algebra per variare la priorità degli operatori aritmetici.



Le parentesi tonde forzano l'esecuzione del prodotto prima dell'elevamento a potenza nonostante quest'ultimo abbia una priorità maggiore del primo.

```
>> 2^(3*4)
ans
        4096
```

8*4 → 32 — Sequenza operazioni senza parentesi

2^12 → 4096 — Sequenza operazioni con le parentesi


```
>> (2+8)/4
ans
        2.5
```

2+2 → 4

10/4 → 2.5

Sequenza operazioni corretta

```
>> 8/(2+2)
ans
        2
```

4+2 → 6

8/4 → 2

Sequenza operazioni corretta

N.B.: le uniche parentesi che possono essere utilizzate in Matlab per variare la priorità degli operatori aritmetici sono le parentesi tonde ( ), le quadre e le graffe { } sono utilizzate per altri scopi.

Formula Matematica

$$2\left\{3\left[(2+4)(3-5)+1\right]+7\right\}$$

Espressione Matlab

```
>> 2*(3*((2+4)*(3-5)+1)+7)
ans
        -52
```

Espressione Matlab Corretta

Formule Matematiche

Nel trascrivere una formula matematica in un'espressione Matlab bisogna fare attenzione al fatto che:

- le uniche parentesi che possono essere utilizzate in Matlab sono le parentesi tonde ( ), le quadre [ ] e le graffe { } sono utilizzate per altri scopi;
- l'operatore prodotto non può essere omesso

Formula Matematica

$$2\{3[(2+4)(3-5)+1]+7\}$$

```
>> 2*(3*((2+4)*(3-5)+1)+7)
ans
        -52
```

corretta

```
>> 2*(3((2+4)*(3-5)+1)+7)
??? 2*(3((2+4)*(3-5)+1)+7)
                |
Error: ")" expected, "(" found.
```

errata

Manca l'operatore prodotto

NOTA BENE

Bisogna sempre ricordare che a parità di priorità le istruzioni vengono eseguite da sinistra a destra

Formula Matematica

$$\frac{1}{3*5}$$

```
>> 1/(3*5)
ans
        0.0667
```

corretta

Le parentesi forzano il prodotto 3*5 ad essere eseguito per primo

1/15 → 0.0667

```
>> 1/3/5
ans
        0.0667
```

corretta

Le operazioni vengono effettuate da sinistra a destra poiché la priorità è la stessa

0.33/5 → 0.0667

```
>> 1/3*5
ans
        1.6667
```

errata

Le operazioni vengono effettuate da sinistra a destra poiché la priorità è la stessa, ma la logica è errata

0.33*5 → 1.6667

NOTA BENE

Bisogna sempre distinguere fra ERRORI LOGICI e ERRORI DI SINTASSI

ERRORI LOGICI	Il calcolo viene eseguito ma non nella maniera voluta ed il risultato ottenuto è sbagliato. Sono anche detti BUGS.
ERRORI DI SINTASSI	Il calcolo non può essere eseguito, non si ottiene un risultato, ma un messaggio di errore da Matlab

Manca l'operatore prodotto

```
>> 2*(3((2+4)*(3-5)+1)+7)
??? 2*(3((2+4)*(3-5)+1)+7)
                |
Error: ")" expected, "(" found.
```


Verifica

Lo studente deve essere in grado di calcolare correttamente le seguenti formule:

1 $2\left(\frac{3+4}{2*5}\right)$

2 $3\left[2.0\cdot10^{-1}\left(\frac{1}{3+5}+\frac{3}{5}\right)+0.1\right]$

3 $2\left[\frac{1.5\cdot10^{-2}+0.1}{3.5\left(0.1+\frac{3}{5}\right)}+2.0\cdot10^{-3}\right]-10^{-2}$

Soluzione 1 e 2

Parentesi ridondanti

1 $2\left(\frac{3+4}{2*5}\right)$

```
>> 2 * (3+4)/(2*5)
ans =
     1.4000
```
corretta

```
>> 2*((3+4)/(2*5))
ans =
     1.4000
```
sconsigliata

```
>> 2 * (3+4)/2/5
ans =
     1.4000
```
corretta

```
>> 2 * (3+4)/2*5
ans =
     35
```
errata

2 $3\left[2.0\cdot10^{-1}\left(\frac{1}{3+5}+\frac{3}{5}\right)+0.1\right]$

```
>> 3 *( 2.0e-1 * (1/(3+5) + 3/5) + 0.1 )
ans =
    0.7350
```


Soluzione 3

$$2\left[\frac{1.5\cdot10^{-2}+0.1}{3.5\left(0.1+\frac{3}{5}\right)}+2.0\cdot10^{-3}\right]-10^{-2}$$

```
>> 2*( (1.5e-2+0.1) / 3.5 / (0.1+3/5) + 2.0e-3) - 1e-2

ans =

    0.0879
```

Separatori di istruzioni.

È possibile scrivere due istruzioni matlab in sequenza sulla stessa riga di comando separandole con una virgola (,) o un punto e virgola (;):

, virgola

```
>> 2*3, 2+3
ans =
    6
ans =
    5
```

; punto e virgola

```
>> 2*3; 2+3
ans =
    5
```

N.B.: Il punto e virgola agisce non solo come separatore di istruzioni ma anche come soppressore dell'output testuale, infatti il risultato del primo calcolo non viene più mostrato.

Formato di output

È possibile modificare il formato numerico di output, ossia il formato con cui Matlab rappresenta i risultati dei calcoli usando il comando `format`.

Formato di default

Di *default* il formato utilizzato è un formato con solo 5 cifre (*short)* ed il valore viene rappresentato come *fixed point*, se possibile, altrimenti come *floating point* (formato esponenziale).

```
>> 1/100
ans =
    0.0100
```

```
>>1/1e4
ans =
   1.0000e-004
```

Il comando `format long` aumenta la precisione del formato numerico a 15 cifre

```
format long, 1/100
ans =
   0.01000000000000
```

Formato di output

FORMATO ESPONENZIALE

Il comando `format short e` forza il formato ad esponenziale con una precisione di 5 cifre

```
>>format short e, 1/100
ans =
   1.0000e-002
```

Il comando `format long e` forza il formato ad esponenziale con una precisione di 15 cifre

```
>>format long e, 1/100
ans =
   1.000000000000000e-002
```

FORMATO DI DEFAULT

Il comando `format` riporta il formato di output a quello di default

```
>>format, 1/100
ans =
   0.0100
```


Formato di output

FORMATO OTTIMIZZATO

Il comando `format short g` forza il formato ad un essere ottimizzato rispetto alla precisione a 5 cifre: solo gli zeri significativi vengono mostrati.

```
>>format short g, 1/100, 1/3e7
ans =
   0.01
ans =
   3.3333e-008
```

Il comando `format long g` forza il formato ad un essere ottimizzato rispetto alla precisione a 15 cifre: solo gli zeri significativi vengono mostrati.

```
>>format long g, 1/100, 1/3e7
ans =
   0.01
ans =
   3.333333333333333e-008
```

Funzioni Matematiche

In matlab esistono librerie di funzioni matematiche che possono essere facilmente utilizzate dall'utente:

Funzioni trigonometriche

NB.: Le funzioni trigonometriche dirette prendono l'argomento esclusivamente in radianti e non in gradi

$$\alpha_{rad} = \frac{\pi}{180}\alpha_{\deg}$$

Angolo in radianti

Angolo in gradi

Funzioni dirette

```
>>sin(3.14)
ans =
    0.0016
>>cos(pi)
ans =
    -1
>>tan(pi/2)
ans =
    1.0000
```

Funzioni inverse

```
>>asin(0.5)
ans =
    0.5236
>>acos(3/4)
ans =
    0.7227
>>atan(1.0)
ans =
    0.7854
```


Funzioni e Comandi

Una funzione è un'istruzione che riceve un certo elenco di valori (argomenti della funzione) fra parentesi tonde, separati da virgole, in un ordine opportuno, e restituisce un o più valori numerici:

```
>>nome_Funzione(arg1, arg2,…)
ans =
        valore funzione
```

Argomenti funzione

Valore funzione

Un comando è un'istruzione che riceve un certo numero di parametri separati da spazi e non restituisce un valore ma serve ad impostare una qualche proprietà del sistema

```
>>nome_Comando par1 par2 …
>>
```

parametri comando

Comando help

Serve ad ottenere informazioni da Matlab circa l'uso di una funzione matematica o di un comando:

```
>>help cos
 cos    Cosine.
    cos(X) is the cosine of the elements of X.
 Overloaded methods
    help sym/cos.m
```


Lezione 2

Scalari e Stringhe

Tipi di variabili in Matlab

In matlab è possibile creare delle variabili i cui valori possono essere da utilizzare e aggiornati durante i calcoli sia in modalità *console* che modalità *interprete*

Nell'ambiente Matlab esistono solo due tipi di variabili:

Variabili numeriche: possono essere scalari, vettori, o matrici di numeri interi, reali, o complessi.

Variabili stringa: sono sequenze di caratteri alfa numerici (es. 'Hello world')

Creazione di Variabili

Un variabile, sia numerica che stringa viene creata automaticamente attraverso un'istruzione di *assegnazione*, ossia una linea di comando in cui a sinistra del segno uguale sia riportato il nome di una variabile e sulla destra il suo valore:

```
Nome_variabile = valore
```

Istruzione di assegnazione

⇨ Il nome di una variabile può essere una qualsiasi sequenza alfanumerica (di lettere o numeri)

⇨ Il valore di una variabile può essere esplicito o il risultato del un calcolo di un espressione matematica

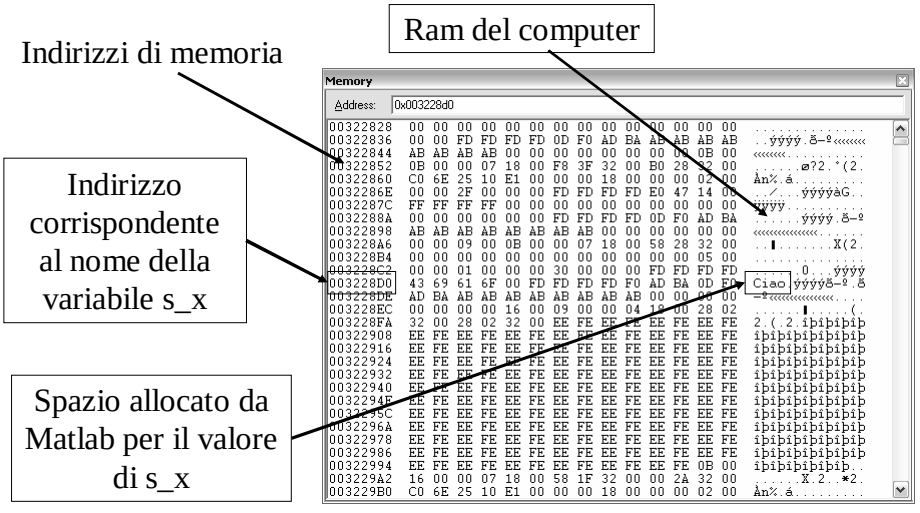
Creazione di Variabili

Creare una variabile significa concretamente richiedere all'ambiente Matlab di allocare una certa quantità di memoria nella RAM del computer in cui scrivere il suo valore.

Il nome della variabile non è altro che un *alias*, ossia un nome alternativo, semplice da ricordare, dell'indirizzo di memoria, ossia dell'esatta posizione in memoria in cui è stato scritto il valore della variabile.


```
s_x = 'Ciao'
```

creazione della variabile stringa s_x il cui nome è l'alias per l'indirizzo esadecimale 003228D0



Regole Nome Variabile

Il nome di una variabile può essere una qualsiasi sequenza alfanumerica (di lettere o numeri) che rispetti le seguenti regole:

- la lunghezza massima è di 19 caratteri;
- il primo carattere non deve essere un numero;
- non deve contenere spazi;
- non deve contenere segni di interpunzione (; , . ! ?) o operatori aritmetici (* + : / -)
- può contenere il segno "_"

Utilizzo Variabili

Una variabile non può essere utilizzata se prima non è stata creata, ossia non può trovarsi sulla destra di un'istruzione di assegnazione prima della sua creazione !

Esempi espliciti di creazioni di variabili verranno riportati nelle sezioni riguardanti i diversi tipi di variabili

Variabili Stringa in Matlab

Creazione

Operazioni

Stringhe

Una variabile stringa è una sequenza alfanumerica di caratteri che vengono manipolati da Matlab come un testo e non come una variabile numerica.

Creazione di Stringhe

Per creare una variabile stringa in MatLab è necessario semplicemente definirla, ossia porlo sulla sinistra di un'equazione di assegnazione:

```
S_x = . . .
```

dove a destra dovrà comparire un'espressione che possa essere valutata come appunto una stringa. Esistono vari modi per definire l'espressione di destra:

1. **Per valori**
2. **Usando la funzione di libreria** → *sprintf*

NOTA BENE: Utilizzeremo la convenzione di far precedere i nomi delle variabili strnga dal prefisso "s_" se vettori riga e "c_" se colonna

Stringhe create per valori

Per creare una stringa per valori basta porre sulla destra di un'istruzione di assegnazione la sequenza di caratteri alfanumerici delimitata da due apici:

```
>> S_x = 'Hello World'
S_x =
Hello world
```

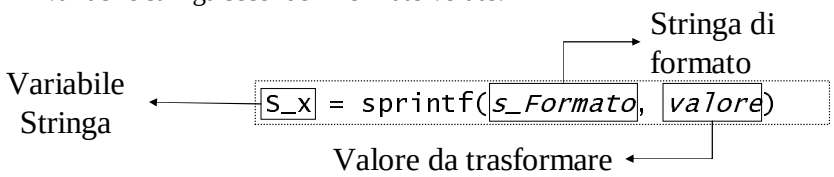
```
>> S_x = '1234'
S_x =
1234
```

Per visualizzare il valore di una variabile stringa senza il nome ad essa associato si può usare il comando `disp`:

```
>> S_x = 'Hello World'; disp (s_x)
Hello world
```

Funzione `sprintf`

La funzione `sprintf` serve a trasformare un valore numerico in una variabile stringa secondo il formato voluto.




```
S_x = sprintf('%f', 5.5)
5.500000
```
Formato fixed point

```
S_x = sprintf('%e', 5.5)
5.500000e+000
```
Formato esponenziale

```
S_x = sprintf('%d', round(5.5))
5
```
Formato intero

Variabili Numeriche in Matlab

Scalari

Vettori

Matrici

Variabili Numeriche in Matlab

In Matlab tutte le variabili numeriche vengono trattate come matrici, e possono essere reali o complesse.

Per semplicità in questa esposizione verranno considerati separatamente:

- Le variabili scalari (ad un solo valore)
- I vettori (riga o colonna)
- Le matrici

Scalari in Matlab

Creazione di scalari
Operazione fra Scalari

Creazione di Scalari

Sono variabili numeriche ad un sol valore. Possono essere create attraverso una semplice istruzione di assegnazione:

```
>> x = 5;
```
→ È stata creata la variabile x a cui è stato assegnato il valore 5

```
>> y = 1.2e-2;
```
→ È stata creata la variabile y a cui è stato assegnato il valore: $1.2 \cdot 10^{-2}$

```
>> z = log(0.1);
```
→ È stata creata la variabile z a cui è assegnato il risultato:

log(0.1)=-2.3026


```
>> x = 5*7/(2*10);

>> y = 5*7/2*10;
```

Sono state create le variabili x ed y a cui è stato assegnato come valore il risultato del calcolo sulla destra del segno di uguaglianza.

```
>> x = 5*8;

>> y = x/2;
```

E' stata creata prima la variabile x (uguale a 40) ed poi è stata utilizzata per la creazione della variabile y il cui valore risulta 20.

Si noti come in quest'ultimo caso l'uso delle parentesi porti a due risultati differenti:

⇨ x = 1.75
y = 175

Operazioni fra Scalari

Con le variabili scalari sono possibili tutte le normali operazioni aritmetiche:

```
>> x = 7;
>> y = 2;
>> z1 = x + y;
>> z2 = x – y;
>> z3 = x * y;
>> z4 = x / y;
>> z5 = y^x;
```

operatori

+ somma
- sottrazione
* prodotto
/ divisione
^ elevamento a potenza

risultati

```
z1 = 9
z2 = 5
z3 = 14
z4 = 3.5
z5 = 49
```


Verifica

Si crei un script Matlab per il calcolo del peso molecolare del NitroBenzene:

$C_6H_5NO_2$ e del DiNitroBenzene $C_6H_4(NO_2)_2$ creando le variabili numeriche per i pesi atomici dei singoli atomi e si usi il comando `sprintf` per ottenere un output testuale così formattato:

Carbonio = 12.011
Idrogeno = 1.008
Ossigeno = 15.999
Azoto = 14.007

Lezione 3

Vettori e Matrici
Parte I

Università degli Studi di Bari
Laurea in Chimica
Di spense di Informatica - Dott. F. Mavelli

Matrici: introduzione matematica

Cenni di Algebra Matriciale

Algebra Matriciale

L'algebra matriciale è una branca della matematica che si occupa delle proprietà delle matrici è definisce le seguenti operazioni fra matrici:

Somma

Sottrazione

Prodotto

Divisione (solo per matrici quadrate)

Trasposizione

Con il termine matrice si intendono in questo caso anche i vettori

Definizione di Matrice

Tutte le variabili numeriche in MATLAB vengono trattate come **matrici**, ossia come tabelle bidimensionali di numeri, organizzate in righe e colonne:

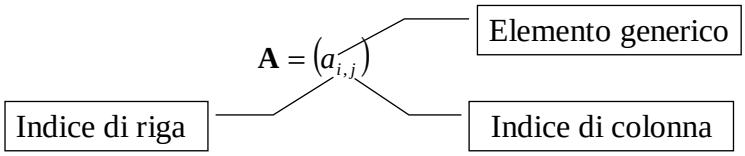
A è una matrice di ordine (n x m) in quanto è formata da n righe ed m colonne

$$\mathbf{A} = \left(a_{i,j}\right) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m-1} & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m-1} & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,m-1} & a_{n-1,m} \\ a_{n,1} & a_{n,2} & \cdots & a_{n-1,m} & a_{n,m} \end{pmatrix}_{\text{n x m}}$$

Ogni elemento $a_{i,j}$ della matrice **A** è contraddistinto da un indice di riga (i) e di colonna (j) che ne individuano la posizione all'interno della matrice stessa.

Simbolismo e Ordine della Matrice

Una Matrice viene rappresentata con una lettere maiuscola in grassetto o con una lettera minuscola fra parentesi tonde:



L'Ordine di una Matrice è dato dal numero delle sue righe e delle sue colonne

$1 \leq i \leq \mathbf{n}$ = **numero di righe**

$1 \leq j \leq \mathbf{m}$ = **numero di colonne**

} **ordine della matrice (n x m)**

Matrice Rettangolari e Quadrate

$1 \leq i \leq$ n = numero di righe

$1 \leq j \leq$ m = numero di colonne

n ≠ m

Matrici rettangolari

$$\begin{pmatrix} 3 & 6 & 1 \\ 5 & 2 & 4 \end{pmatrix}$$

n = m

Matrici quadrate

$$\begin{pmatrix} 3 & 6 \\ 5 & 2 \end{pmatrix}$$

Vettori

Vengono chiamate **vettori** quelle matrici che hanno o numero di righe o di colonne unitario:

$$\mathbf{v} = \begin{pmatrix} v_{1,1} \\ v_{2,1} \\ \vdots \\ v_{n,1} \end{pmatrix}_{n\times 1}$$

Vettore Colonna

$$\mathbf{v} = \begin{pmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,n} \end{pmatrix}_{1\times n}$$

Vettore Riga

Gli scalari altro non sono che matrici formate da una solo riga ed una sola colonna

Matrice Reale e Matrice Complessa

Una matrice **A** è detta Matrice Reale se tutti i suoi elementi sono numeri reali:

A Matrice reale

$a_{i,j} \in \mathfrak{R}$

Una matrice **A** è detta Matrice Complessa se almeno uno dei suoi elementi è un numero complesso:

A Matrice complessa

$a_{i,j} \in \mathfrak{C}$

Numero complesso:

$z = a + i \cdot b$

Unità immaginaria

$i = \sqrt{-1} \quad \Leftrightarrow \quad i^2 = -1$

Parte reale

Parte Immaginaria

Matrice Complessa Coniugata

Data una matrice complessa **A,** è detta Matrice Complessa Coniugata di **A** e denotata con il simbolo **Ã,** la matrice che ha per elementi i complessi coniugati degli elementi di **A**:

$$\tilde{\mathbf{A}} = \left(\tilde{a}_{i,j}\right) = \left(\tilde{a}_{i,j}\right)$$

Corollario: la **m**atrice **A è** Reale se e solo se è uguale alla sua Complessa coniugata $\mathbf{A} = \tilde{\mathbf{A}}$

Il complesso coniugato di un numero complesso si ottiene cambiando di segno alla parte immaginaria

Numero Complesso	⇨	Complesso Coniugato
$z = a \pm ib \in \mathbf{C}$		$\tilde{z} = a \mp ib \in \mathbf{C}$

Matrici Trasposte

Data la matrice **A** la sua trasposta **A′** si ottiene scambiando le righe con le colonne:

$$\mathbf{A} = \begin{pmatrix} 3 & 6 & 1 \\ 5 & 2 & 4 \end{pmatrix}_{2x3} \qquad \mathbf{A}' = \begin{pmatrix} 3 & 5 \\ 6 & 2 \\ 1 & 4 \end{pmatrix}_{3x2}$$

$$\left[\mathbf{A}' = \left(a'_{i,j}\right) = \left(a_{i,j}\right)' = \left(a_{j,i}\right)\right] \longrightarrow \left[a'_{2,1} = 6 = a_{1,2}\right]$$

Matrici Rettangolari Trasposte

L'operazione di trasposizione di una matrice rettangolare inverte l'ordine della matrice:

$$\mathbf{A} = \begin{pmatrix} 3 & 6 & 1 \\ 5 & 2 & 4 \end{pmatrix}_{2x3} \qquad \mathbf{A}' = \begin{pmatrix} 3 & 5 \\ 6 & 2 \\ 1 & 4 \end{pmatrix}_{3x2}$$

L'operazione di trasposizione di un vettore colonna restituisce un vettore riga e viceversa:

$$\mathbf{a} = \begin{pmatrix} 2 \\ 9 \\ 4 \\ 11 \end{pmatrix}_{4 \times 1} \qquad \mathbf{a}' = \begin{pmatrix} 2 & 9 & 11 & 4 \end{pmatrix}_{1 \times 4}$$

Matrici Quadrate Trasposte

L'operazione di trasposizione di una matrice quadrata mantiene fissi gli elementi sulla diagonale principale e scambia quelli fuori diagonale

$$\mathbf{A} = \begin{pmatrix} 10 & 1 & 2 \\ 3 & 2 & 1 \\ 1 & 6 & 9 \end{pmatrix}_{3x3} \qquad \mathbf{A}' = \begin{pmatrix} 10 & 1 & 2 \\ 3 & 2 & 1 \\ 1 & 6 & 9 \end{pmatrix}'_{3x3} = \begin{pmatrix} 10 & 3 & 1 \\ 1 & 2 & 6 \\ 2 & 1 & 9 \end{pmatrix}_{3x3}$$

Gli elementi sulla diagonale principale sono quelli che hanno uguali l'indice di riga e di colonna → $a_{i,j}$ → i=j → elemento sulla diagonale; i≠j → elemento fuori diagonale

Prodotto Matrice Scalare

Data una matrice **A** di ordine (n x m) ed un numero *c*, reale o complesso, il prodotto:

$$\mathbf{B} = c\,\mathbf{A}$$

è una matrice di ordine (n x m) i cui elementi sono i corrispondenti elementi di **A** moltiplicati per il numero *c*

$$b_{i,j} = c \cdot a_{i,j}$$

$$\mathbf{B} = c\mathbf{A} = c\begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & \dots & a_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,m} \end{pmatrix}_{n\,x\,m} = \begin{pmatrix} c\cdot a_{1,1} & c\cdot a_{1,2} & \dots & c\cdot a_{1,m} \\ c\cdot a_{2,1} & c\cdot a_{2,2} & \dots & c\cdot a_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ c\cdot a_{n,1} & c\cdot a_{n,2} & \dots & c\cdot a_{n,m} \end{pmatrix}_{n\,x\,m}$$

Esempi:

$$c = 5 \quad \times \quad \mathbf{a} = \begin{pmatrix} 2 \\ 9 \\ 4 \\ 11 \end{pmatrix}_{4\,x\,1} \quad \rightarrow \quad c\,\mathbf{a} = \begin{pmatrix} 10 \\ 45 \\ 20 \\ 55 \end{pmatrix}_{4\,x\,1}$$

$$c = 5 \quad \times \quad \mathbf{A} = \begin{pmatrix} 3 & 5 \\ 6 & 2 \\ 1 & 4 \end{pmatrix}_{3x2} \quad \rightarrow \quad c\,\mathbf{A} = \begin{pmatrix} 15 & 25 \\ 30 & 10 \\ 5 & 20 \end{pmatrix}_{3x2}$$

Somma Algebrica di Matrici

Date due matrici **A** e **B** di uguale ordine (n x m), resta definita la matrice **C,** di ordine (n x m), ottenuta dalla somma algebrica delle matrici date:

$$\mathbf{C} = \mathbf{B} \pm \mathbf{A}$$

e i cui elementi sono dati dalla somma algebrica elemento a elemento degli elementi corrispondenti delle matrici **A** e **B**.

$$c_{i,j} = b_{i,j} \pm a_{i,j}$$

Due Matrici **A** e **B** possono essere sommate o sottratte solo se hanno lo stesso ordine.

Esempi:

$$\mathbf{C} = \mathbf{B} \pm \mathbf{A} = \begin{pmatrix} b_{1,1} \pm a_{1,1} & b_{1,2} \pm a_{1,2} & \dots & b_{1,m} \pm a_{1,m} \\ b_{2,1} \pm a_{2,1} & b_{2,2} \pm a_{2,2} & \dots & b_{2,m} \pm a_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n,1} \pm a_{n,1} & b_{n,2} \pm a_{n,2} & \dots & b_{n,m} \pm a_{n,m} \end{pmatrix}_{\text{n x m}}$$

Somma Matrici

$$\begin{pmatrix} 3 & 5 \\ 6 & 2 \\ 1 & 4 \end{pmatrix}_{3x2} + \begin{pmatrix} 15 & 25 \\ 30 & 10 \\ 5 & 20 \end{pmatrix}_{3x2} = \begin{pmatrix} 18 & 30 \\ 36 & 12 \\ 6 & 24 \end{pmatrix}_{3x2}$$

Somma Vettori riga

$$\begin{pmatrix} 2 & 9 & 11 & 4 \end{pmatrix}_{1x4} +$$

$$\begin{pmatrix} 7 & 5 & 2 & 19 \end{pmatrix}_{1x4} =$$

$$\begin{pmatrix} 9 & 14 & 13 & 23 \end{pmatrix}_{1x4}$$

Vettori in Matlab

Creazione

Creazione di Vettori

Per creare un vettore in MatLab è necessario semplicemente definirlo, ossia porlo sulla sinistra di un'equazione di assegnazione:

```
r_x = . . .
```

dove a destra dovrà comparire un'espressione che possa essere valutata come appunto un vettore. Esistono vari modi per definire l'espressione di destra:

1. **Per valori**
2. **Usando le funzioni di libreria** → *ones, zeros, linspace, logspace, rand*
3. **Usando la notazione *colon* ":"**

NOTA BENE: Utilizzeremo la convenzione di far precedere i nomi dei vettori dal prefisso "r_" se vettori riga e "c_" se colonna

Per Valori

Vettore riga

Un vettore **riga** può essere creato inserendo in sequenza i valori dei suoi elementi separati da spazi o da virgole e compresi fra parentesi quadre:

```
>> r_x = [ 1 2 3 4 5]
r_x =
     1   2   3   4   5
```

```
>> r_x = [1,2,3,4,5]
r_x =
     1   2   3   4   5
```

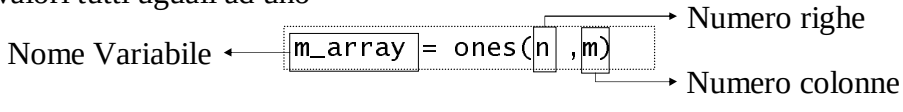
```
>> c_x = [1; 2; 3]

c_x =
     1
     2
     3
```

Vettore colonna

Un vettore **colonna** può essere creato inserendo in sequenza i valori dei suoi elementi separati da punti e virgola e compresi fra parentesi quadre:

Funzione *ones*

La funzione `ones(n,m)` restituisce una matrice di ordine (nxm) con valori tutti uguali ad uno



La funzione *ones* permette di creare un vettore riga o colonna delle dimensioni volute i cui elementi sono tutti unitari

Vettore colonna

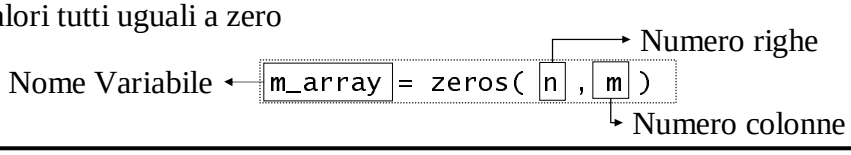
```
>> c_y = ones(3,1)
c_y =
     1
     1
     1
```

Vettore riga

```
>> r_x = ones(1,5)
r_x =
     1     1     1     1     1
```

Funzione *zeros*

La funzione `zeros(n,m)` restituisce una matrice di ordine (nxm) con valori tutti uguali a zero



La funzione *zeros* permette di creare un vettore riga o colonna delle dimensioni volute i cui elementi sono tutti nulli

Vettore colonna

```
>> c_y =
zeros(3,1)

c_y =

     0
     0
     0
```

3 righe

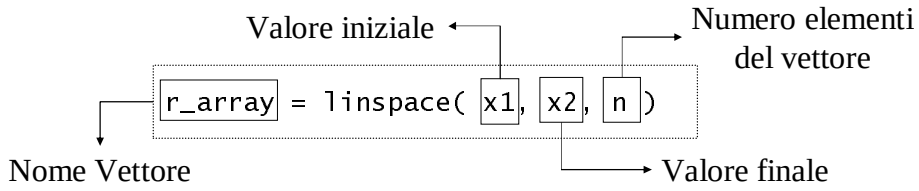
Vettore riga

```
>> r_x = zeros(1,5)

r_x =

     0     0     0     0     0
```

5 colonne

Funzione *linspace*

La funzione `linspace (x1,x2, n)` restituisce vettore riga di ordine (1xn) con valori linearmente equispaziati fra il valore iniziale x1 ed il valore finale x2:

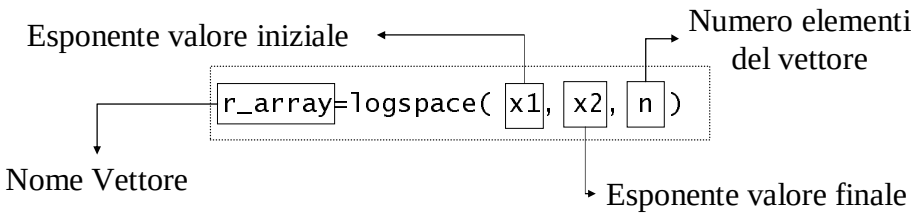



```
>> r_y = linspace(2.1, 4, 5)

r_y  =

    2.1000    2.5750    3.0500    3.5250    4.0000
```

Vettore di 5 elementi

Funzione *logspace*

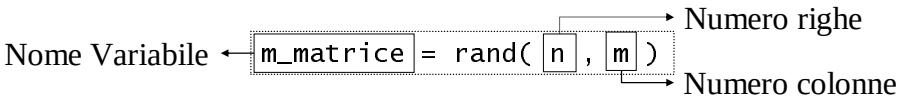
La funzione `logspace (x1,x2, n)` restituisce vettore riga di ordine (1xn) con valori logaritmicamente equispaziati fra il valore iniziale 10^{x1} ed il valore finale 10^{x2}:




```
>> r_z = logspace(1,4,4)
r_z =
          10         100        1000       10000
```


Funzione *rand*

La funzione `rand(n,m)` restituisce una matrice di ordine (nxm) con valori casuali nell'intervallo]0, 1[.



La funzione *rand* permette di creare un vettore riga o colonna delle dimensioni volute i cui elementi siano casualmente distribuiti fra 0 e 1

Vettore colonna

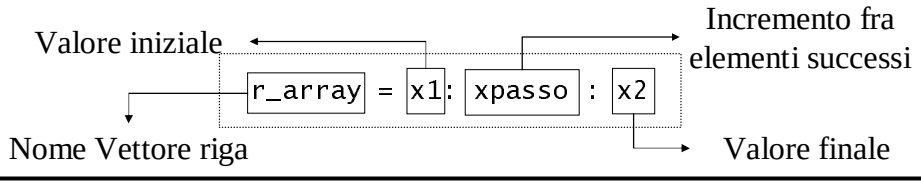
```
>> c_y = rand(3,1)
c_y =
    0.2722
    0.1988
    0.0153
```

Vettore riga

```
>> r_x = rand(1,4)
r_x =
 0.7468    0.4451    0.9318    0.4660
```

Notazione *colon* ":"

E' una notazione molto compatta per creare un vettore riga definendo il valore iniziale, l'incremento fra elementi successi ed il valore finale:



Se il passo è omesso viene assunto 1 automaticamente

```
>> r_x = 1.5 : 3.5
r_x  =
    1.5000    2.5000    3.5000
```

Non è detto che il valore finale sia x2

```
>> r_x = 1 : 0.6 : 3
r_x  =
  1.0000  1.6000  2.2000  2.8000
```


Funzioni e Vettori

Se ad una funzione viene passato come argomento una variabile vettore la funzione restituisce come valore un vettore di uguale dimensione i cui elementi sono ottenuti calcolando la funzione sui rispettivi elementi del vettore argomento:

```
>> r_x = 0:0.1: 0.5
r_x =
   0.0000    0.1000    0.2000    0.3000    0.4000    0.5000

>> r_y = cos(r_x)
r_y =
   1.0000    0.9950    0.9801    0.9553    0.9211    0.8776
```

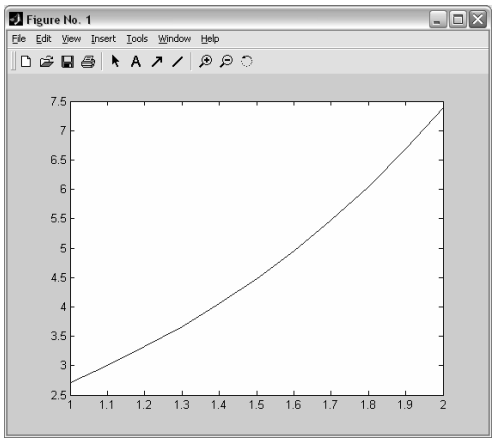
I singoli elementi del vettore `r_y` sono ottenuti calcolando il coseno sui rispettivi valori del vettore `r_x` passati alla funzione come parametro

```
cos(0.0000) = 1.0000
cos(0.1000) = 0.9950

 cos(0.5000) = 0.8776
```

Funzione plot (r_X, r_Y)

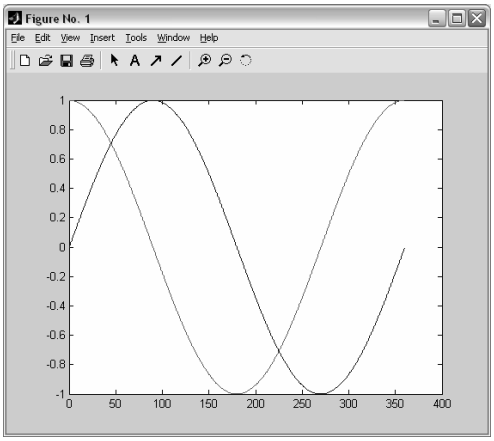
La funzione plot è una funzione che prende come argomenti due vettori entrambi riga o colonna di uguale lunghezza e restituisce come output il grafico del primo vettore (variabile indipendente asse X orizzontale) contro il secondo vettore (variabile dipendente asse Y verticale)

```
>> r_x = 1:0.1: 2;
>> r_y = exp(r_x);
>> plot (r_x, r_y)
```



Verifica

Si scriva un script Matlab che crei il grafico della funzione seno e coseno nell'intervallo 0-360 gradi



NB.: per poter far apparire più curve sullo stesso grafico si devono semplicemente passare alla funzione `plot` tutte le relative coppie di vettori di coordinate X, Y:

```
plot (r_x1, r_y1, r_x2, r_y2, r_x3, r_y3…)
```

Matrici in Matlab

Creazione

Creazione di Matrici

Le matrici possono essere create analogamente ai vettori

1. **Per valori**
2. **Usando le funzioni** di libreria
3. Creazione di Matrici a blocchi
4. **Matrici caricate da file**

Per valori

Una matrice può essere creata inserendo in sequenza i valori dei suoi elementi separati da spazi o da virgole e compresi fra parentesi quadre usando come separatore di riga il punto e virgole o andando a capo

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

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

```
>> m_A = [ 1, 2, 3
 4, 5, 6]
m_A =
     1   2   3
     4   5   6
```

N.B.: i tre modi proposti sono assolutamente equivalenti fra loro

Funzioni: *zeros e ones*

ones(n)

restituisce una matrice quadrata di ordine (nxn) in cui tutti gli elementi sono uguali ad uno: matrice unitaria.

```
>> m_A = ones(3)
m_A =

     1     1     1
     1     1     1
     1     1     1
```

ones(n, m)

una matrice rettangolare di ordine (nxn) in cui tutti gli elementi sono uguali ad uno: matrice unitaria

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

     1     1
     1     1
     1     1
```

Funzioni: *zeros*

zeros(n)

restituisce una matrice quadrata di ordine (nxn) in cui tutti gli elementi sono nulli.

```
>> m_A = zeros(3)
m_A =

     0     0     0
     0     0     0
     0     0     0
```

zeros(n,m)

una matrice rettangolare di ordine (nxm) in cui tutti gli elementi sono uguali a zero

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

     0     0     0
     0     0     0
```


Funzioni: *magic* e *eye*

`magic(n)` Solo per matrici quadrate

restituisce una matrice quadrata di ordine (nxn) in cui righe, colonne e diagonali hanno la somma degli elementi uguale.

```
>> m_A = magic(3)
m_A =

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

`eye(n,m)`

restituisce la matrice di ordine (nxm) che ha tutti valori nulli tranne quelli con indici fra loro uguali che sono unitari: $a_{i,i} = 1$

```
>> m_A = eye(2,3)
m_A =
     1     0     0
     0     1     0
```

`eye(n)`

restituisce la matrice identità di ordine (nxn).

```
>> m_A = eye(3)
m_A =
     1     0     0
     0     1     0
     0     0     1
```

Funzioni: *rand e randn*

rand(n, m)

restituisce una matrice rettangolare di ordine (nxm) i cui elementi hanno valori random uniformemente distribuiti fra 0 e 1.

```
>> A = rand(2, 3)

A =

   0.0535   0.0077   0.4175
   0.5297   0.3834   0.6868
```

randn(n, m)

restituisce una matrice rettangolare di ordine (nxm) i cui elementi valori distribuiti normalmente con media 0 e varianza 1.

```
>> A = randn(3,2)

A =

   1.1650   0.3516
   0.6268  -0.6965
   0.0751   1.6961
```


Operazioni con Scalari

Tutte le operazioni di somma (+), sottrazione (-), prodotto (*), divisione (/) di una matrice o vettore per uno scalare sono definite, in Matlab come operazioni elemento ad elemento: il risultato è una matrice o un vettore i cui elementi sono ottenuti sommando, sottraendo, moltiplicando, dividendo i singoli elementi della matrice o del vettore per lo scalare.

N.B.: Si noti come in algebra matriciale solo il prodotto scalare matrice sia in realtà definito

```
>> y = 10;
>> r_x = 1 : 4
r_x =
   1    2    3    4
```

somma

```
>> r_x + y
ans  =
11    12    13    14
```

sottraz.

```
>> y - r_x
ans  =
9    8    7    6
```

prodot.

```
>> r_x * y
ans  =
10    20    30    40
```

divisione

```
>> r_x / y
ans  =
0.1  0.2  0.3  0.4
```

Operazioni con Scalari

Tutte le operazioni fra matrici e scalari in matlab sono commutative eccetto la divisione

somma

```
>> m_x = [1 2; 3 4]; y = 10
>> m_x + y
ans =
    11    12
    13    14
>> y + m_x
ans =
    11    12
    13    14
```

divisione

```
>> m_x = [1 2; 3 4]; y = 10
>> m_x / y
ans =
    0.1000    0.2000
    0.3000    0.4000
>> y / m_x

?? Error using ==> /
Matrix dimensions must agree.
```

N.B.: E' possibile dividere un vettore o una matrice per uno scalare, ma non uno scalare per un vettore o una matrice

Lezione 4

Vettori e Matrici

Parte II

Matrici: introduzione matematica

Cenni di Algebra Matriciale

Definizione di Matrice

Tutte le variabili numeriche in MATLAB vengono trattate come **matrici**, ossia come tabelle bidimensionali di numeri, organizzate in righe e colonne:

A è una matrice di ordine (n x m) in quanto è formata da n righe ed m colonne

$$\mathbf{A} = \left(a_{i,j}\right) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m-1} & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m-1} & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,m-1} & a_{n-1,m} \\ a_{n,1} & a_{n,2} & \cdots & a_{n-1,m} & a_{n,m} \end{pmatrix}_{\text{n x m}}$$

Ogni elemento $a_{i,j}$ della matrice **A** è contraddistinto da un indice di riga (i) e di colonna (j) che ne individua la posizione all'interno della matrice stessa.

Vettori

Vengono chiamate **vettori** quelle matrici che hanno o numero di righe o di colonne unitario:

$$\mathbf{v} = \begin{pmatrix} v_{1,1} \\ v_{2,1} \\ \vdots \\ v_{n,1} \end{pmatrix}_{n\times 1}$$

Vettore Colonna

$$\mathbf{v} = \begin{pmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,n} \end{pmatrix}_{1\times n}$$

Vettore Riga

Gli scalari altro non sono che matrici formate da una solo riga ed una sola colonna

Prodotto Matrice Scalare

Data una matrice **A** di ordine (n x m) ed un numero *c*, reale o complesso, il prodotto:

$$\mathbf{B} = c\,\mathbf{A}$$

è una matrice di ordine (n x m) i cui elementi sono i corrispondenti elementi di **A** moltiplicati per lo scalare *c*

$$b_{i,j} = c \cdot a_{i,j}$$

$$\mathbf{B} = c\mathbf{A} = c\begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & \dots & a_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,m} \end{pmatrix}_{n\,x\,m} = \begin{pmatrix} c\cdot a_{1,1} & c\cdot a_{1,2} & \dots & c\cdot a_{1,m} \\ c\cdot a_{2,1} & c\cdot a_{2,2} & \dots & c\cdot a_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ c\cdot a_{n,1} & c\cdot a_{n,2} & \dots & c\cdot a_{n,m} \end{pmatrix}_{n\,x\,m}$$

Esempi:

$$c = 5 \quad \times \quad \mathbf{a} = \begin{pmatrix} 2 \\ 9 \\ 4 \\ 11 \end{pmatrix}_{4\,x\,1} \quad \rightarrow \quad c\,\mathbf{a} = \begin{pmatrix} 10 \\ 45 \\ 20 \\ 55 \end{pmatrix}_{4\,x\,1}$$

$$c = 5 \quad \times \quad \mathbf{A} = \begin{pmatrix} 3 & 5 \\ 6 & 2 \\ 1 & 4 \end{pmatrix}_{3x2} \quad \rightarrow \quad c\,\mathbf{A} = \begin{pmatrix} 15 & 25 \\ 30 & 10 \\ 5 & 20 \end{pmatrix}_{3x2}$$

Somma Algebrica di Matrici

Date due matrici **A** e **B** di uguale ordine (n x m), resta definita la matrice **C,** di ordine (n x m), ottenuta dalla somma algebrica delle matrici date:

$$\mathbf{C} = \mathbf{B} \pm \mathbf{A}$$

e i cui elementi sono dati dalla somma algebrica elemento a elemento degli elementi corrispondenti delle matrici **A** e **B**.

$$c_{i,j} = b_{i,j} \pm a_{i,j}$$

Due Matrici **A** e **B** possono essere sommate o sottratte solo se hanno lo stesso ordine.

Esempi:

$$\mathbf{C} = \mathbf{B} \pm \mathbf{A} = \begin{pmatrix} b_{1,1} \pm a_{1,1} & b_{1,2} \pm a_{1,2} & \dots & b_{1,m} \pm a_{1,m} \\ b_{2,1} \pm a_{2,1} & b_{2,2} \pm a_{2,2} & \dots & b_{2,m} \pm a_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n,1} \pm a_{n,1} & b_{n,2} \pm a_{n,2} & \dots & b_{n,m} \pm a_{n,m} \end{pmatrix}_{\text{n x m}}$$

Somma Matrici

$$\begin{pmatrix} 3 & 5 \\ 6 & 2 \\ 1 & 4 \end{pmatrix}_{3x2} + \begin{pmatrix} 15 & 25 \\ 30 & 10 \\ 5 & 20 \end{pmatrix}_{3x2} = \begin{pmatrix} 18 & 30 \\ 36 & 12 \\ 6 & 24 \end{pmatrix}_{3x2}$$

Somma Vettori riga

$$\begin{pmatrix} 2 & 9 & 11 & 4 \end{pmatrix}_{1 \text{x } 4} +$$

$$\begin{pmatrix} 7 & 5 & 2 & 19 \end{pmatrix}_{1 \text{x } 4} =$$

$$\begin{pmatrix} 9 & 14 & 13 & 23 \end{pmatrix}_{1 \text{x } 4}$$

Trasposizione di Matrici

Data la matrice **A** la sua trasposta **A′** si ottiene scambiando le righe con le colonne:

$$\mathbf{A} = \begin{pmatrix} 3 & \textcircled{6} & 1 \\ 5 & 2 & 4 \end{pmatrix}_{2x3} \qquad \mathbf{A}' = \begin{pmatrix} 3 & 5 \\ \textcircled{6} & 2 \\ 1 & 4 \end{pmatrix}_{3x2}$$

$$\mathbf{A}' = \left(a'_{i,j}\right) = \left(a_{i,j}\right)' = \left(a_{j,i}\right) \longrightarrow a'_{2,1} = 6 = a_{1,2}$$

Prodotto di Matrici

Il prodotto di una matrice **A,** di ordine ($n \times s$), per la matrice **B** di ordine ($s \times m$), è la matrice **C,** di ordine ($n \times m$):

$$\mathbf{C_{(n\times m)} = A_{(n\times s)} \times B_{(s\times m)}}$$

il cui elemento generico $c_{i,j}$ è dato dalla somma dei prodotti degli elementi della riga i-esima della matrice A per i corrispondenti elementi della j-esima colonna della matrice **B**.

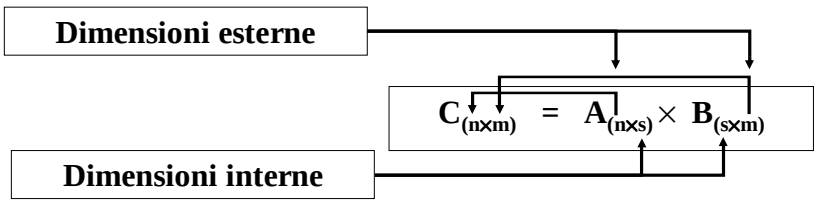
$$c_{i,j} = \sum_{k=1}^{S} a_{i,k} b_{k,j} = a_{i,1} b_{1,j} + a_{i,2} b_{2,j} + \cdots + a_{i,S} b_{S,j}$$

Due Matrici **A** e **B** possono essere moltiplicate fra loro **solo se** il numero di colonne di **A** è uguale al numero di righe di **B**.

Prodotto di Matrici

Il prodotto di matrici così definito viene anche detto:

prodotto righe per colonne



- Le dimensioni interne devono essere uguali
- La matrice risultato **C** ha le dimensioni esterne

Esempi

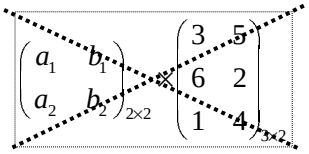
$$c_{1,2} = \sum_{k=1}^{S} a_{1,k} \cdot b_{k,2}$$

$$\mathbf{A} \times \mathbf{B} = \begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} + \ldots + a_{1,S}b_{S,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} + \ldots + a_{1,S}b_{S,2} & \ldots & a_{1,1}b_{1,m} + a_{1,2}b_{2,m} + \ldots + a_{1,S}b_{S,m} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} + \ldots + a_{2,S}b_{S,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} + \ldots + a_{2,S}b_{S,2} & \ldots & a_{2,1}b_{1,m} + a_{2,2}b_{2,m} + \ldots + a_{2,S}b_{S,m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1}b_{1,1} + a_{n,2}b_{2,1} + \ldots + a_{n,S}b_{S,1} & a_{n,1}b_{1,2} + a_{n,2}b_{2,2} + \ldots + a_{n,S}b_{S,2} & \ldots & a_{n,1}b_{1,m} + a_{n,2}b_{2,m} + \ldots + a_{n,S}b_{S,m} \end{pmatrix}_{\text{n x m}}$$

$$\begin{pmatrix} 3 & 5 \\ 6 & 2 \\ 1 & 4 \end{pmatrix}_{3\times 2} \times \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix}_{2\times 2} = \begin{pmatrix} 3a_1 + 5a_2 & 3b_1 + 5b_2 \\ 6a_1 + 2a_2 & 6b_1 + 2b_2 \\ a_1 + 4a_2 & b_1 + 4b_2 \end{pmatrix}_{3\text{x}2}$$

! Anche se esiste $\mathbf{C} = \mathbf{A} \times \mathbf{B}$ non è detto che sia definito il prodotto:

$\mathbf{B} \times \mathbf{A}$



Prodotti righe per colonne

Prodotto
matrice $\times$ vettore

$$\begin{pmatrix} \quad \end{pmatrix}_{n\times s} \times \begin{pmatrix} \\ \\ \end{pmatrix}_{s\text{x}1} = \begin{pmatrix} \\ \\ \end{pmatrix}_{n\times 1}$$

Prodotto
vettore riga $\times$ vettore colonna

$$(\quad)_{1\times s} \times \begin{pmatrix} \\ \\ \end{pmatrix}_{s\text{x}1} = c$$

Prodotto
vettore colonna $\times$ vettore riga

$$\begin{pmatrix} \\ \\ \end{pmatrix}_{s\times 1} \times (\quad)_{1\times p} = \begin{pmatrix} \quad \end{pmatrix}_{s\times p}$$

Matrice identità **E**

Si definisce matrice identità **E** la matrice quadrata che ha elementi tutti nulli eccetto quelli sulla diagonale principale che sono uguali ad 1.

$$\mathbf{E} = (e_{i,j}) = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

Proprietà della matrice identità

Il prodotto della matrice identità E per una qualsiasi matrice quadrata A restituisce la matrice A stessa

$$\mathbf{A}\times\mathbf{E} = \mathbf{E}\times\mathbf{A} = \mathbf{A}$$

$$\mathbf{A}\times\mathbf{E} = \begin{pmatrix} a_{1,1}1+a_{1,2}0+\ldots+a_{1,n}0 & a_{1,1}0+a_{1,2}1+\ldots+a_{1,n}0 & \ldots & a_{1,1}0+a_{1,2}0+\ldots+a_{1,n}1 \\ a_{2,1}1+a_{2,2}0+\ldots+a_{2,n}0 & a_{2,1}0+a_{2,2}1+\ldots+a_{2,n}0 & \ldots & a_{2,1}b_{1,m}+a_{2,2}b_{2,m}+\ldots+a_{2,n}1 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}1+a_{n,2}0+\ldots+a_{n,n}0 & a_{n,1}0+a_{n,2}1+\ldots+a_{n,n}0 & \ldots & a_{n,1}b_{1,m}+a_{n,2}b_{2,m}+\ldots+a_{n,n}1 \end{pmatrix}_{\text{n x n}}$$

Matrice Inversa $\mathbf{A^{-1}}$

Data una matrice quadrata **A** viene definita matrice inversa di **A** e denotata con il simbolo $\mathbf{A}^{-1}$ la matrice che soddisfa la seguente relazione:

$$\mathbf{A}\times\mathbf{A}^{-1} = \mathbf{A}^{-1}\times\mathbf{A} = \mathbf{E}$$

ossia quella matrice che moltiplicata per la matrice **A** restituisce la matrice identità (Si noti che in questo caso il prodotto è commutativo).

- La matrice inversa resta definita solo per matrici quadrate
- Non tutte le matrici quadrate sono dotate di inversa
- Si dimostra che le matrici quadrate dotate di inversa sono quelle a determinante non nullo e sono dette non singolari

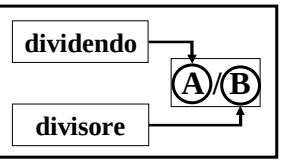
Divisione

Date due Matrici $\mathbf{A}=(a_{i,j})$ e $\mathbf{B}=(b_{i,j})$ viene definita l'operazione di divisione della matrice A per B come il prodotto della matrice A per l'inversa della matrice B:

$$\mathbf{A} / \mathbf{B} = A \times B^{-1}$$

dove B^{-1} è la matrice inversa della matrice B e resta definita solo per matrici quadrate, non singolari, ossia a determinante non nullo.

Per cui matrici rettangolari e vettori non possono essere i divisori in un operazione di divisione fra matrici



Operazioni		**Requisiti**	**Modalità**
Prodotto	$c\mathbf{A}$	--	Operazione elemento a elemento
Somma Algebrica	$\mathbf{A} \pm \mathbf{B}$	Uguali dimensioni: stesso numero di righe e di colonne	Operazione elemento a elemento
Prodotto	$\mathbf{A} \times \mathbf{B}$	Numero di colonne di **A** uguale al numero di righe di **B**	Prodotto righe per colonne
Divisione	$\mathbf{A} / \mathbf{B}$	**B** matrice non singolare, ossia dotata di inversa $\mathbf{B}^{-1}$ $\mathbf{B}\times\mathbf{B}^{-1} = \mathbf{B}^{-1}\times\mathbf{B} = \mathbf{E}$	$\mathbf{A} / \mathbf{B} = \mathbf{A} \times \mathbf{B}^{-1}$
Elevamento a potenza	$\mathbf{A}^n$	Solo per matrici quadrate	$\mathbf{A}\times\mathbf{A}\times \ldots(n \text{ volte})\ldots\times\mathbf{A}$

Vettori e Matrici in Matlab

Operazioni con Scalari

Operazioni fra Vettori e Matrici

Operazioni con Scalari

Tutte le operazioni di somma (+), sottrazione (-), prodotto (*), divisione (/) di una matrice o vettore per uno scalare sono definite, in Matlab come operazioni elemento ad elemento: il risultato è una matrice o un vettore i cui elementi sono ottenuti sommando, sottraendo, moltiplicando, dividendo i singoli elementi della matrice o del vettore per lo scalare.

N.B.: Si ricordi che in algebra matriciale solo il prodotto scalare per matrice è in realtà definito

```
>> y = 10;
>> r_x = 1 : 4
r_x =
   1    2    3    4
```

somma

```
>> r_x + y
ans  =
11    12    13    14
```

sottraz.

```
>> y - r_x
ans  =
9    8    7    6
```

prodot.

```
>> r_x * y
ans  =
10    20    30    40
```

divisione

```
>> r_x / y
ans  =
0.1  0.2  0.3  0.4
```

Operazioni con Scalari

Tutte le operazioni fra matrici e scalari in matlab sono commutative eccetto la divisione

somma

```
>> m_x = [1 2; 3 4]; y = 10
>> m_x + y
ans =
    11    12
    13    14
>> y + m_x
ans =
    11    12
    13    14
```

divisione

```
>> m_x = [1 2; 3 4]; y = 10
>> m_x / y
ans =
    0.1000    0.2000
    0.3000    0.4000
>> y / m_x

?? Error using ==> /
Matrix dimensions must agree.
```

N.B.: E' possibile dividere un vettore o una matrice per uno scalare, ma non uno scalare per un vettore o una matrice

Operazioni fra Vettori e Matrici

Matlab permette di effettuare facilmente operazioni fra vettori e matrici rispettando le regole dell'algebra matriciale, ma implementa anche degli operatori che permettono di effettuare operazioni di tipo diverso:

Operatori che seguono le regole dell'algebra matriciale →

- (+) Somma
- (-) Sottrazione
- (*) Prodotto Righe per Colonne
- (/) Divisione: matrici non singolari

L' Operatore punto '.' forza le operazioni ad essere effettuate elemento ad elemento →

- (.*) Prodotto elemento a elemento
- (./) Divisione elemento a elemento
- (.^) Elevamento a Potenza elemento a elemento

Operazioni fra Vettori e Matrici

Operatori che seguono le regole dell'algebra matriciale

Somma (+) e Sottrazione (-)

La somma e la sottrazione di vettori o matrici sono definite come in algebra matriciale operazioni elemento a elemento:

- i vettori (o matrici) operandi devono quindi avere uguali dimensioni
- il vettore (o matrice) risultante è dato dalla somma o sottrazione elemento a elemento dei vettori (o matrici) addendi

```
» r_x = 1:4
r_x =
     1     2     3     4

» r_y = 10:10:40
r_y =
    10    20    30    40

» r_z = r_x + r_y
r_z =
    11    22    33    44
```

Matlab restituisce un messaggio di errore se si cerca di sommare o sottrarre vettori con dimensioni non corrette

```
» r_x = 1:3; r_y = 10:10:40;
» r_z = r_x + r_y
??? Error using ==> +
Matrix dimensions must agree.
```

(*) Prodotto righe per colonne

Il prodotto fra vettori segue le regole dell'algebra matriciale due vettori possono essere moltiplicati con l'operatore * solo se:
- il numero di colonne del primo vettore è uguale al numero di righe del secondo,
- il prodotto è effettuato righe per colonne.

Possono essere moltiplicati fra loro con *:

Vettore Riga * Vettore Colonna → Scalare

Vettore Colonna * Vettore Riga → Matrice

```
» r_x = 1:4;
r_x =
    1    2    3    4
» c_y = [10;20;30; 40]
c_y =
    10
    20
    30
    40
» z = r_x * c_y
z =
    300
» m_z = c_y * r_x
m_z =
    10    20    30    40
    20    40    60    80
    30    60    90   120
    40    80   120   160
```


(*) Prodotto Righe per Colonne

Se si cerca di moltiplicare due matrici di dimensioni non corrette: allora Matlab restituisce un messaggio di errore

```
» r_x = 1:4;
r_x =
    1    2    3    4
» r_y = [10 20 30 40]
r_y =
    10   20    30    40
» r_y * r_x
??? Error using ==> *
Inner matrix dimensions
must agree.
```

$r\_y_{(1\times4)}$ * $r\_x_{(1\times4)}$

```
??? Errore usando ==> *
Le dimensioni interne delle
matrici devono essere uguali
```

Operazioni fra Vettori e Matrici

Operatori elemento a elemento

(.*) Prodotto Elemento a Elemento

In Matlab è definita anche l'operazione Prodotto elemento a elemento che segue quindi le stesse regole della somma e della sottrazione:

- i vettori operandi devono avere uguali dimensioni
- il vettore risultante è dato dal prodotto elemento a elemento dei vettori operandi

```
» r_x = 1:4
r_x =
     1     2     3     4

» r_y = 10:10:40
r_y =
    10    20    30    40

» r_z = r_x .* r_y
r_z =
    10    40    90   160
```

In Matlab l'operatore punto "." forza un'operazione matriciale di moltiplicazione (.*), divisione (./) e elevamento a potenza (.^) ad essere effettuata in modalità elemento a elemento.

(./) Divisione Elemento a Elemento

Anche la Divisione fra due vettori, in Matlab, può essere effettuata elemento a elemento utilizzando l'operatore (./):

- i vettori operandi devono avere uguali dimensioni
- il vettore risultante è dato dalla divisione elemento a elemento dei vettori operandi

```
» r_x = 1:4
r_x =
     1     2     3     4

» r_y = 10:10:40
r_y =
    10    20    30    40

» r_z = r_x ./ r_y
r_z =
    0.1   0.1   0.1   0.1
```

Utilizzando l'operatore ./ è anche possibile effettuare la divisione di uno scalare per un vettore dividendo lo scalare per i singoli elementi del vettore divisore

```
» r_x = 1:4;
» r_y = 1./r_x
r_y =
   1.0000   0.5000   0.3333   0.2500
```


(.^) Elevamento a Potenza Elemento a Elemento

L'elevamento a potenza elemento a elemento (.^) permette di calcolare:

```
» r_x = 1:4
r_x =
    1     2     3     4
```

1 l'elevamento a potenza di tutti gli elementi di un vettore ad uno stesso esponente scalare;

```
» r_y = r_x .^2
r_y =
    1     4     9     16
```

2 l'elevamento a potenza di uno scalare a tutti i valori di un vettore, presi come esponenti;

```
» r_z = 2.^ r_x
r_z =
     2     4     8     16
```

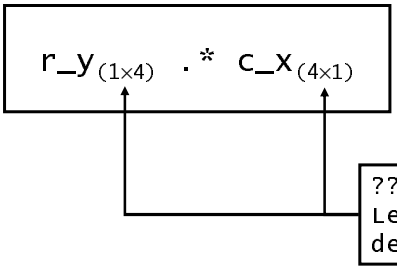
3 l'elevamento a potenza degli elementi di un vettore agli elementi si un altro vettore presi come esponenti,

```
» r_q = r_x.^[1 2 3 2]
r_q =
    1     4     27     16
```

Operazioni Elemento a Elemento

Nelle Operazioni elemento a elemento le matrici o i vettori operandi devono avere uguali dimensioni, ossia devono essere uguali sia il numero delle righe che delle colonne.

```
» r_x = 1:4;
r_x =
    1    2    3    4
» c_y = [10; 20; 30; 40]
c_y =
    10
    20
    30
    40
» r_y .* c_x
??? Error using ==> .*
Matrix dimensions must
agree.
```




```
??? Errore usando ==> *
Le dimensioni delle matrici
devono essere uguali
```


Verifica

- Si creino due matrici rettangolari con il comando `rand` e si verifichino le regole di addizione sottrazione prodotto.
- Si creino due matrice quadrate con il comando `rand` e si verifichi la regola di divisione

Riepilogo

Operazioni in Matlab
fra variabili numeriche

Operazioni Matlab: Scalare-Matrice

Operazioni		**Modalità**	**Commuta**	**Requisiti**
Somma	`s+m_A`	elemento a elemento	SI	
Differenza	`s-m_A`	elemento a elemento	SI	
Prodotto	`s*m_A`	elemento a elemento	SI	
Divisione	`m_A/s`	elemento a elemento	NO	
Elevamento a potenza	`m_A^s`	elemento a elemento	NO	solo matrici quadrate

Un operatore binario commuta se il risultato non cambia invertendo l'ordine degli operandi

```
s + m_A = m_A + s
```

Operazioni Matlab: Matrice-Matrice

Operazioni		Modalità	Commuta	Requisiti
Somma	`m_A+m_B`	elemento a elemento	SI	matrici di uguali dimensioni
Differenza	`m_A-m_B`	elemento a elemento	SI	matrici di uguali dimensioni
Prodotto	`m_A*m_B`	righe per colonne	NO	N. colonne di `m_A` = N. righe di `m_B`
Divisione	`m_A/m_B`	`m_A*inv(m_B)`	NO	`m_A` e `m_B` quadrate `m_B` dotata di inversa

Operazioni Matlab aggiuntive

Operazioni matrice scalare		Modalità	Commuta	Requisiti
Divisione	`s./m_A`	elemento a elemento	NO	
Operazioni matrice-matrice		**Modalità**	**Commuta**	**Requisiti**
Prodotto	`m_A.*m_B`	elemento a elemento	SI	matrici di uguali dimensioni
Divisione	`m_A./m_B`	elemento a elemento	NO	matrici di uguali dimensioni
Elevamento a potenza	`m_A.^m_B`	elemento a elemento	NO	matrici di uguali dimensioni

Vettori e Matrici

Parte III

Università degli Studi di Bari
Laurea in Chimica
Di spense di Informatica - Dott. F. Mavelli

Creazione Matrici

Matrici a blocchi

Caricamento dati da file

Matrici a blocchi

Matrici a blocchi (o composte) possono essere facilmente costruite in MatLab considerando i singoli blocchi alla stregua di elementi semplici:

```
>> m_A = rand(3);
>> m_B = [m_A, ones(3,2); zeros(2,3), eye(2)]
m_B =

    0.9304    0.0920    0.7012         1         1
    0.8462    0.6539    0.9103         1         1
    0.5269    0.4160    0.7622         1         1
         0         0         0    1.0000         0
         0         0         0         0    1.0000
```

N:B.: I blocchi devono avere dimensioni opportune.

```
>> m_B = [m_A, ones(3,3); zeros(2,3), eye(2)]
???  All rows in the bracketed expression must have the same
number of columns.
```

In questo caso il blocco creato con la funzione *ones* era eccedente di una colonna rispetto al blocco creato con *eye*.

Matrici caricate da file *.m

Sia “Dati.txt” un file ASCII con estensione “.txt”, scritto come una serie di valori separati da spazi o da virgole (*Coma Separated Values*)

File “Dati.txt”

```
0.9304  0.0920  0.7012
0.8462  0.6539  0.9103
0.5269  0.4160  0.7622
```

È possibile caricare i dati del file che resteranno associati in MatLab alla matrice che ha come nome lo stesso nome del file.

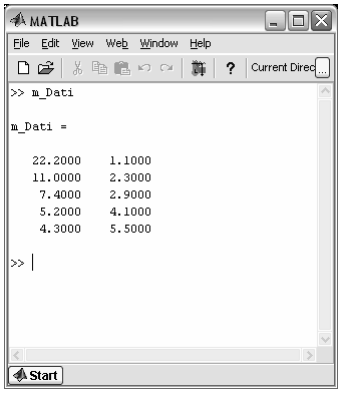
```
>> load Dati.txt
>> Dati

Dati =
   0.9304   0.0920   0.7012
   0.8462   0.6539   0.9103
   0.5269   0.4160   0.7622
```

I file ASCII in formato CSV sono file generati dalle strumentazioni di misura, usate comunemente nei laboratori, e possono essere rielaborati da numerosi programmi di elaborazione dati (tipo Excell, SigmaPlot ecc.) e possono essere letti da semplici editor di testo (tipo notepad.exe o lo stesso editor degli script Matlab).

Verifica

Scaricate il file: Datipress.txt dal sito e scrivete un script per il caricamento dei dati dal file in una matrice di nome m_Dati.

Si visualizzi il valore dei dati nella finestra di comando per essere sicuri di avere caricato correttamente i dati



Indirizzamento diretto

Indirizzamento diretto

Una volta che sia stata creata in Matlab, una matrice od un vettore, si possono utilizzare i valori dei suoi elementi in espressioni numeriche o possono essere assegnati nuovi valori ad ogni singolo elemento della matrice o del vettore.

La tecnica che consente tutto ciò prende il nome di *indirizzamento diretto* e consiste nello scrivere il nome della matrice o del vettore seguito dagli indici della posizione dell'elemento considerato fra parentesi tonde.

```
>> A = [10 20; 3  4]
A =
    10        20
     3         4
>> x = A(2,2)*A(1,2)
x =

        80
```

Il primo indice rappresenta la riga ed il secondo la colonna dell'elemento considerato.

Indirizzamento diretto

Usando l'indirizzamento diretto sulla sinistra di un'istruzione di assegnazione invece che a destra è possibile ridefinire il valore dell'elemento, indicato dalla coppia di indici fra parentesi tonde.

```
>> A = [10 20; 3  4];
>> A(2,2) = 100
A =
    10        20
     3       100
```

Per i vettori sia riga che colonna è possibile anche esplicitare solo l'indice della dimensione diversa da uno sia per riassegnare che per richiamare il valore degli elementi

```
>> r_x = [10 20  3  4];
>> r_x(1,4) = 100
r_x =
    10        20       3      100
>> r_x(2) =  5
 r_x =
    10        5        3       100
>> x = r_(1)*2
x =
     20
```

Indirizzamento diretto

Sulla Sinistra di un'istruzione di assegnazione

All'elemento sulla quinta riga, terza colonna della matrice **A** viene riassegnato il valore 5

```
>> A(5,3) = 5
```

Sulla Destra di un'istruzione di assegnazione

L'elemento sulla quinta riga, terza colonna della matrice A viene utilizzato nella dichiarazione della variabile x:

```
>> x = A(5,3)*5
```


Indirizzamento diretto

Se si assegna un valore ad un elemento inesistente di un vettore o di una matrice, (indirizzamento diretto a sinistra di un'assegnazione), Matlab aumenterà le dimensioni del vettore o della matrice fino a comprendere il nuovo elemento definito e ponendo a zero il valore dei nuovi elementi non definiti:

```
>> A = [10 20; 3  4];
>> A(3,4) = 100
A =
    10    20     0     0
     3   100     0     0
     0     0     0   100
```

```
>> c_x = [10 20]';
>> c_x(5) = 16
c_x =
    10
    20
     0
     0
    16
```

NOTA BENE: Matlab gestisce la memoria in maniera dinamica.

Indirizzamento diretto

Se si cerca di utilizzare il valore di un elemento inesistente di un vettore o di una matrice, (indirizzamento diretto a destra di un'assegnazione), Matlab restituisce un messaggio di errore che indica che sono state superate le dimensioni del vettore o della matrice:

```
>> A = [10 20; 3  4]
A  =
     10      20
      3       4

>> x = A(3,4) * 100

???  Index exceeds matrix dimensions.
```

NOTA BENE: E' stato commesso un errore di sconfinamento

Indirizzamento per vettori di indici

Si può anche usare la tecnica dell'indirizzamento diretto utilizzando vettori di indici.

Nell'esempio a lato, dopo avere creato il vettore riga di 4 elementi r_x:

- vengono riassegnati i valori agli elementi in posizione 1 e 4 mediante il vettore di indici [1 4],
- viene creato un vettore r_y di 2 elementi con valori uguali a quelli degli elementi in posizione 1 e 3 di r_x.

```
>> r_x = [5 20  7  4];
r_x =
    5      20     7      4

>> r_x([1 4]) = 100
r_x =
   100     20     7    100

>> r_y = r_x([1 3])
r_y
       100      7
```

Indirizzamento per vettori di indici

```
>> c_x = 10*(1:10)'
c_x
    10
    20
    30
    40
    50
    60
    70
    80
    90
   100
```

Tutte le posizioni pari del vettore colonna sono state poste a zero mediante l'utilizzo del vettore di indici 2:2:end.

```
>> c_x (2:2:end) = 0
c_x
    10
     0
    30
     0
    50
     0
    70
     0
    90
     0
```

NOTA BENE: L'utilizzo della variabile Matlab end restituisce automaticamente il valore del massimo indice del vettore (10 nell'esempio).

Indirizzamento diretto per vettori di indici

```
>> c_x = 10*(1:10)'
c_x
    10
    20
    30
    40
    50
    60
    70
    80
    90
   100
```

Tutte i valori nelle posizioni dispari del vettore colonna sono stati invertiti di posizione con l'uso dei vettori di indici:

1:2:9 e 9:-2:1

```
>> c_x (1:2:9) = c_x(9:-2:1)
c_x
    90
    20
    70
    40
    50
    60
    30
    80
    10
   100
```

NOTA BENE: L'istruzione c_x (1:2:9) = c_x(9:-2:1) corrisponde al comando:

c_x ([1 3 5 7 9]) = c_x([9 7 5 3 1])

dove i vettori degli indici sono scritti in forma esplicita.

Indirizzamento per vettori di indici

Creata la matrice quadrata m_A è possibile ridefinire tutti gli elementi di una singola riga o colonna

```
>> m_A = magic(3)
m_A =

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

tutti i valori della 2° colonna

```
>> m_A(1:end, 2) = 3
m_A =

     8     3     6
     3     3     7
     4     3     2
```

tutti i valori della 1° riga

```
>> m_A(1, :) = 0
m_A =

     0     0     0
     3     3     7
     4     3     2
```

Si noti come non aver esplicitato gli estremi di variazione nella notazione colon ":" equivalga al vettore di indici 1:end

Indirizzamento per vettori di indici

Creata la matrice quadrata A

```
>> m_A = magic(3)
A =

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

Possono essere creati:

- il vettore colonna c_x con gli elementi della seconda colonna di A
- il vettore riga r_x con gli elementi della prima riga di A
- il vettore riga r_y con gli elementi della seconda riga di A rovesciati

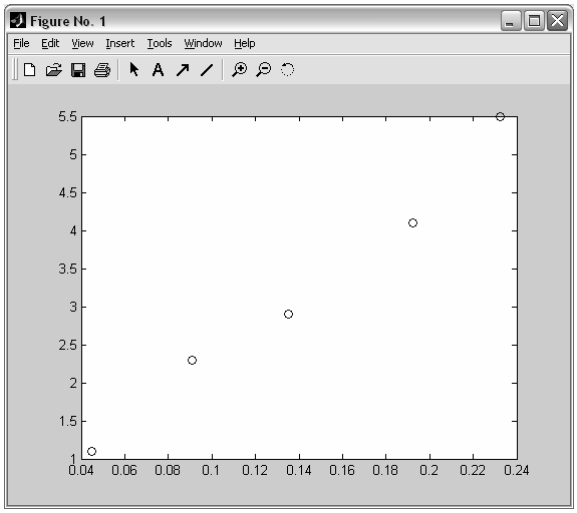
```
>> c_x = A(1:end, 2)
c_x =

     1
     5
     9
```

```
>> r_x = A(1, :)
r_x =

     8     1     6
```

```
>> r_y = A(1, end:-1:1)
r_y =

     7     1     5
```

Verifica

Si ottenga il grafico dei valori della matrice m_Dati riportando l'inverso dei valori della prima colonna contro valori della seconda.



Funzioni per Matrici e Vettori

- max e min
- size e length
- sum e prod

v_X vettore riga o colonna

Funzioni: max(v_X) and min(v_X)

max(v_X)

Restituisce il valore massimo degli elementi di vettore riga o colonna

```
>> r_x = rand(1,3)
r_x =
   0.9501    0.2311    0.6068
>> max(r_x)
 =
   0.9501
```

min(v_X)

Restituisce il valore minimo degli elementi di un vettore riga o colonna

```
>> r_x = rand(1,3)
r_x =
   0.9501    0.2311    0.6068
>> x = min(r_x)
x =
   0.2311
```


m_A Matrice

Funzioni: max(m_A) and min(m_A)

max(m_A)

Restituisce un vettore riga con i valori massimi degli elementi nelle colonne della matrice

```
>> m_A = rand(3,2)
m_A =
   0.4860    0.4565
   0.8913    0.0185
   0.7621    0.8214
```

min(m_A)

Restituisce un vettore riga con i valori minimi degli elementi nelle colonne della matrice

```
>> r_x = max(m_A)
r_x =
   0.8913    0.8214
```

```
>> r_y = min(m_A)
r_y =
   0.4860    0.0185
```

Applicando due volte le funzioni alla matrice m_A è possibile ottenere il valore del massimo e del minimo valore di tutta la matrice

```
>> x = max(max(m_A))
x =
   0.8913
```

```
>> y = min(min(m_A))
y =
   0.0185
```

Funzioni size(A)

```
>> A = rand(3,2)
A =
   0.4860   0.4565
   0.8913   0.0185
   0.7621   0.8214
```

Restituisce le dimensioni di una matrice:

`[r,c] = size(A)` Permette di assegnare alle variabili r e c rispettivamente il numero di righe e colonne della matrice A

```
>> [r, c] = size(A)
r =
   3
c =
   2
```

`r_s = size(A)` Assegna al vettore riga r_s il numero di righe e colonne della matrice A

```
>> r_s = size(A)
r_s =
    3     2
```

`size(A, 1)` Restituisce il numero di righe di una matrice

`size(A, 2)` Restituisce il numero di colonne di una matrice

```
>> r = size(A, 1)
r =
   3
>> c = size(A, 2)
c =
   2
```


Esempio size(m_A)

```
>> m_A = rand(3,2)
m_A =
   0.4860   0.4565
   0.8913   0.0185
   0.7621   0.8214
```

La funzione size può essere utilizzata per creare una matrice delle stesse dimensioni della matrice A con valori tutti nulli.

```
>> m_B = zeros(size(m_A))

m_B =
    0    0
    0    0
    0    0
```

La funzione size può essere utilizzata per creare una matrice delle stesse dimensioni della matrice A con valori tutti unitari.

```
>> m_C = ones(size(m_A))

m_C =
    1    1
    1    1
    1    1
```

Funzioni length(…)

```
>> m_A = rand(3,2)
m_A =
   0.4860   0.4565
   0.8913   0.0185
   0.7621   0.8214
```

m_A Matrice

length (m_A)

Restituisce la dimensione massima di una matrice:

```
>> x = length(m_A)
x =
   3
>> n = max(size(m_A))
n =
    3
```

Equivale ad applicare la funzione max al risultato della funzione size(A)

length (v_X)

v_X vettore riga o colonna

Se applicato ad un vettore riga o colonna restituisce la dimensione del vettore, rispettivamente il numero di righe o di colonne

```
>> r_x = 1:2:10
r_x =
   1   3   5   7   9
>> r = length (r_x)
r =
    5
```

```
>> c_x = [1; 2; 3];
c_x =
    1
    2
    3
>> c = length (r_x)
c =
    3
```


v_X vettore riga o colonna

Funzioni: sum(X) and prod(X)

sum(v_X)

Restituisce la somma dei valori di tutti gli elementi di un vettore riga o colonna

```
>> r_x = (1:2:10)*2
r_x =
    2     6     10     14     18
>> somma = sum(r_x)
somma =
    50
```

prod(v_X)

Restituisce il prodotto dei valori di tutti gli elementi di un vettore riga o colonna

```
>> r_x = (1:2:10)*2
r_x =
    2     6     10     14     18
>> prodotto = prod(r_x)
prodotto =
    30240
```

Funzioni: sum(A) and prod(A)

sum(m_A)

Restituisce un vettore riga con i prodotti massimi degli elementi nelle colonne della matrice

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

prod(m_A)

Restituisce un vettore riga con i prodotti massimi degli elementi nelle colonne della matrice

```
>> r_somma = sum(m_A)
r_somma =
   9   12
```

```
>> r_prodotto = prod(m_A)
r_prodotto =
   15   48
```

Applicando due volte le funzioni sum e prod alla matrice A è possibile ottenere la somma ed il prodotto di tutti gli elementi della matrice

```
>> x = sum(sum(m_A))
x =
   21
```

```
>> y = prod(prod(m_A))
y =
   720
```