Analisi di Immagini e Dati Biologici

Introduzione al linguaggio di MATLAB/OCTAVE

Octave

Download

http://octave.sourceforge.net/

Mac:

Package disponibili per diversi package manager

Linux:

Usare il package manager della propria distribuzione per aggiungere componenti aggiuntive al *core language*

Octave

Windows binaries:

Scaricare Octave 6.2.0 da https://www.gnu.org/software/octave/download

I package si possono scaricare con il comando **pkg**

Octave

- Numerosi 'tutorial' disponibili su Internet
- Manuale (scaricabile o consultabile in rete):
  - `http://www.gnu.org/software/octave/doc/interpreter/`
- Libro:
  - J.W.Eaton, D.Bateman,S.Hauberg "GNU Octave Manual version 3"

La versione per Windows ha anche un manuale in PDF oltre all'intero manuale in formato HTML

Matlab

Disponibile una licenza per studenti

http://imaging.biol.unipr.it/download/Matlab-studenti.pdf

La procedura prevede i seguenti passi

Creare la propria login su mathworks.com (usate l'email assegnata dall'Università come identificativo!)

Selezionare l'opzione "Teaching or research in school"

Associare la licenza

Scaricare il prodotto

L'installazione richiede di nuovo le credenziali attivate sul sito Mathworks

Matlab

Per il corso è necessario installare

MATLAB (ultima versione disponibile)

Fuzzy Logic Toolbox

Image Processing Toolbox

Signal Processing Toolbox

Introduzione a Matlab/Octave

La rappresentazione di base degli elementi simbolici del linguaggio sono le MATRICI

Le matrici sono oggetti matematici che organizzano elementi, oggetti od espressioni secondo una disposizioni tabellare (righe e colonne)

In Matlab/Octave gli elementi numerici sono automaticamente rappresentati come

numeri in virgola mobile in doppia precisione

come coppie di tali numeri (numeri complessi)

Octave/Matlab

La shell: Interprete di comandi

Si presenta con un 'prompt' all'inizio della linea aspettando un comando

```
Octave:1> (Octave)
>> (Matlab)
```

Nella sua forma più immediata la shell esegue il calcolo di espressioni algebriche

Nota: non esiste 'localizzazione' dei numeri per questi strumenti. E.g. il carattere del separatore decimale è il '.' (punto)

```
octave:1> 5+3*4/15
ans =  5.8000
octave:2> 2*log(abs((3+4.5)/(3-2.12)))
ans =  4.2855
```

Octave/Matlab: le 4 operazioni

Operatori che rappresentano alcune delle operazioni fondamentali tra numeri reali

Addizione +

Sottrazione -

Moltiplicazione *

Divisione /

Elevazione a potenza ^

Espressioni vengono valutate da sinistra verso destra tenendo conto delle parentesi

In mancanza di parentesi l'ordine di priorità è

```
^, * /,  + -
```

Rappresentazione Interna dei Dati

Numeri interi con o senza segno

- uint8 intero di 8-bit senza segno (0-255)
- uint16 intero di 16-bit senza segno (0-65535)
- uint32 intero di 32-bit senza segno (0-~4295000000)
- uint64
- int8 intero di 8-bit con segno (-128...+127)
- int16 intero di 16-bit con segno (-32768...+32767)
- int32
- int64

Esercizio: usate le funzioni intmax e intmin per determinare gli estremi rappresentabili

- Eg: intmax('uint8')

Numeri Interi

Possono essere con o senza segno

Usano la *rappresentazione complemento a 2* dove MSB viene usato per rappresentare numeri negativi

8-bit two's-complement integers

Bits	Unsigned value	2's complement value
0000 0000	0	0
0000 0001	1	1
0000 0010	2	2
0111 1110	126	126
0111 1111	127	127
1000 0000	128	−128
1000 0001	129	−127
1000 0010	130	−126
1111 1110	254	−2
1111 1111	255	−1

Numeri in Virgola Mobile

Internamente rappresentati base 2

IEEE-754 fissa oltre ad altri aspetti tecnici

- Il numero dei bit da riservare per significante ed esponente ad ogni precisione
- La rappresentazione di
  - +Inf, -Inf (più infinito, meno infinito)
  - NaN (not a number)

Rappresentazione di default per Octave: virgola mobile a doppia precisione (double:64-bit complessivi)

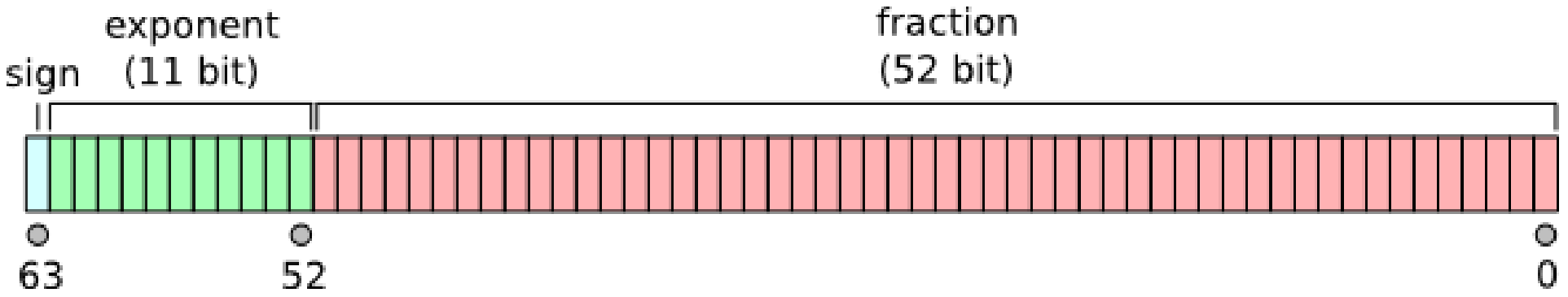
Numeri in Virgola Mobile

Usati per immagini di intensità, soprattutto ad alta precisione

Rappresentazione di *default* per Octave e Matlab

Rappresentazione analoga alla notazione scientifica:

$$(-1)^{\text{segno}} \times (\text{fraction}) \times 2^{\text{exponent}-1023}$$



Caveats

Occorre prestare attenzione agli effetti della rappresentazione

- In algoritmi complessi bisogna tenere conto degli errori che possono propagarsi a causa dell'arrotondamento causato da condizioni di *underflow ff*
- E così anche per le condizioni di *overflow.* Con numeri in virgola mobile bisogna prestare attenzione a non confrontare tra loro o manipolare numeri troppo diversi in scala
- Arrotondamenti sono frequenti, anche operazioni semplici ripetute portano a risultati approssimati
- Si deve prestare attenzione al formato dei dati ed eventualmente usare le funzioni di convesione

Tabella Riassuntiva Tipi di Dati Fondamentali

Name	Description
`double`	Double-precision, floating-point numbers in the approximate range -10^{308} to 10^{308} (8 bytes per element).
`uint8`	Unsigned 8-bit integers in the range [0, 255] (1 byte per element).
`uint16`	Unsigned 16-bit integers in the range [0, 65535] (2 bytes per element).
`uint32`	Unsigned 32-bit integers in the range [0, 4294967295] (4 bytes per element).
`int8`	Signed 8-bit integers in the range [−128, 127] (1 byte per element).
`int16`	Signed 16-bit integers in the range [−32768, 32767] (2 bytes per element).
`int32`	Signed 32-bit integers in the range [−2147483648, 2147483647] (4 bytes per element).
`single`	Single-precision floating-point numbers with values in the approximate range -10^{38} to 10^{38} (4 bytes per element).
`char`	Characters (2 bytes per element).
`logical`	Values are 0 or 1 (1 byte per element).

Variabili

1. Creazione di una variabile
2. Assegnazione ad una variabile del risultato di una funzione
3. Assegnazione ad una variabile del risultato di una espressione

```
octave:1> a = 1
a =  1

octave:2> esp = exp(a)
esp =  2.7183

octave:3> tang_ip = (exp(a) – exp(-a)) / (exp(a) + exp(-a))
tang_ip =  0.76159

octave:4> tang_ip
tang_ip =  0.76159
```

Variabili di Octave

Il valore di ogni entità può essere “salvato” in variabili

I nomi di variabili sono formati dalle lettere dell'alfabeto e da numeri più il carattere “_”

I nomi di variabili non possono iniziare con una cifra numerica

Ogni variabile viene inizializzata con l'operatore '=' che assegna alla variabile a sinistra dell'operatore il risultato dell'espressione o della funzione alla destra di esso

Se una variabile non esiste viene creata

Se già esistente viene sovrascritta

Funzioni

Funzioni: accettano liste di argomenti.

Talvolta il numero di argomenti ammissibili può essere variabile

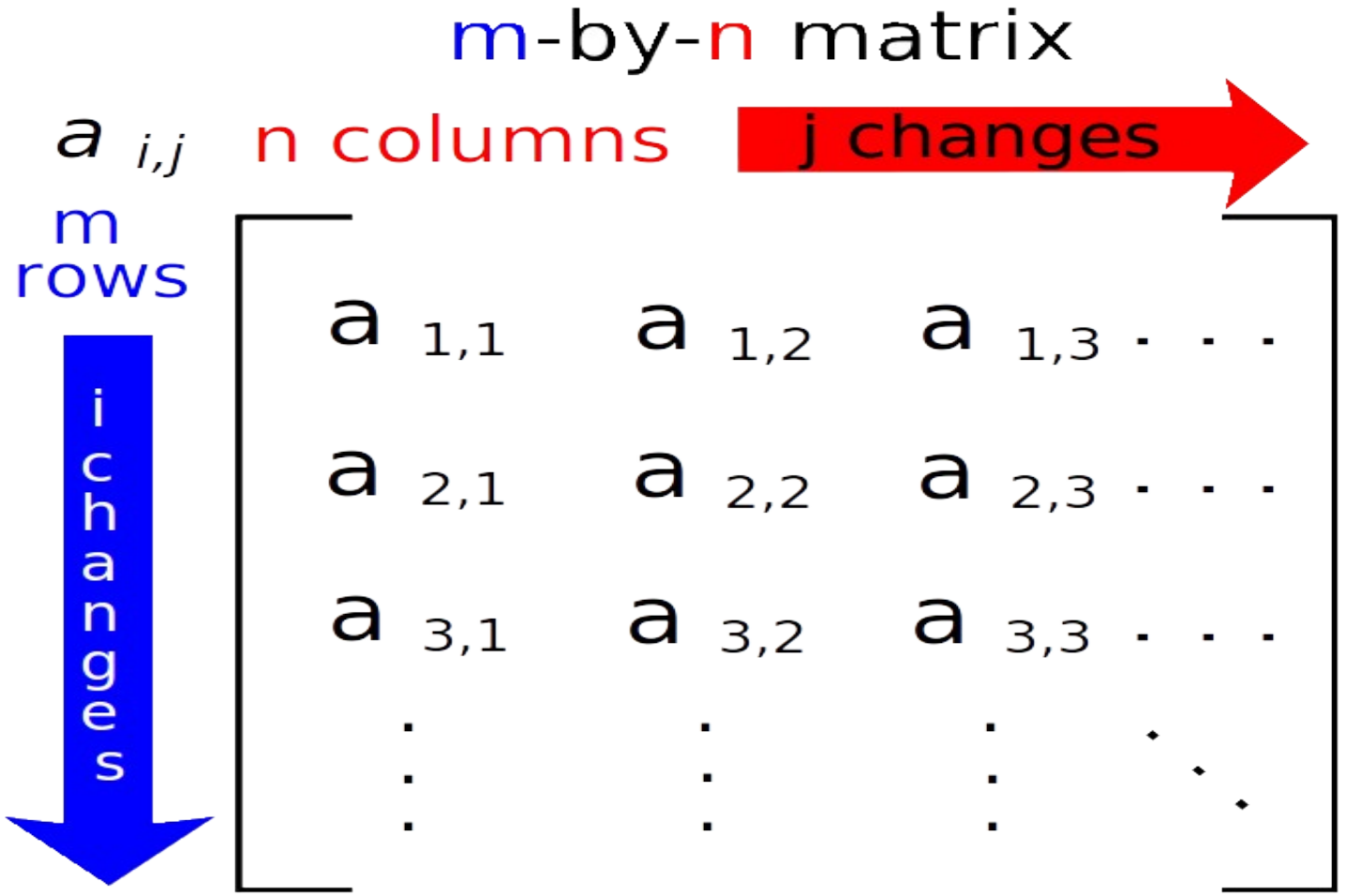
Le funzioni possono comparire in espressioni

Le funzioni possono ritornare un numero variabile risultati

E' sempre possibile avere documentazione su una funzione invocando il comando

```
'octave:2> help <nome funzione>'
```

Introduzione ad Octave/Matlab
Matrici



Matrici

Esempio di matrice

```
Octave:1> m = [1 2 -3; -4 5 -6; 7 -8 9]
matrice =

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

octave:2> size(matrice)

ans = 3 3
```

Somma di Matrici

L'operazione di somma di matrici è definita quando 2 matrici hanno la stessa dimensione

Stessa dimensione per 2 matrici A e B significa

Numero di colonne di A = numero di colonne di B

Numero di righe di A = numero di righe di B

Non è necessario che il numero di righe = numero di colonne (matrici quadrate)

Somma di Matrici

Esempio

```
octave:8> A=[1 -5.4 3; 2 2.8 9; 12 5 3]
A =

    1.0000   -5.4000    3.0000
    2.0000    2.8000    9.0000
   12.0000    5.0000    3.0000

octave:9> B=[2 6 8.3; -3 4 5.6; 1 1.2 -2.5]
B =

   2.0000   6.0000   8.3000
  -3.0000   4.0000   5.6000
   1.0000   1.2000  -2.5000

octave:10> A+B
ans =

    3.00000    0.60000   11.30000
   -1.00000    6.80000   14.60000
   13.00000    6.20000    0.50000
```

Octave/Matlab: Matrici e Array

Caso speciale: Array (o **vettori)**

Sono matrici che hanno una sola colonna o una sola riga

```
octave:1> riga = [1 2 3 4 5 6]
riga =

   1   2   3   4   5   6

octave:2> size(riga)
ans =

   1   6
```

Scalare

Caso speciale: una matrice 1x1 è detta scalare

Se uno scalare moltiplica una matrice ogni elemento della matrice viene moltiplicato

Così se uno scalare viene sommato ad una matrice il risultato è la matrice i cui elementi sono stati tutti sommati ad esso

Operazioni con Scalari

Esempi

```
octave:1> scalare=10
scalare =  10
octave:2> scalare*matrice
ans =

   10   20  -30
  -40   50  -60
   70  -80   90

octave:3> scalare+matrice
ans =

   11   12    7
    6   15    4
   17    2   19
```

Trasposizione di Matrici

L'operatore di trasposizione scambia righe e colonne di una matrice

```
octave:14> matrix=[1 2 -3; -4 5 -6; 7 -8 9]
matrix =

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

octave:15> transpose(matrix)
ans =

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

Più comunemente l'operatore di trasposizione viene reso con l'apice posto dopo il riferimento alla matrice (matrix')

Trasposizione di un Array

La trasposizione trasforma un vettore riga in un vettore colonna

```
octave:1> riga = [1 2 3 4 5 6]
riga =

   1   2   3   4   5   6

octave:2> colonna=riga'
colonna =

   1
   2
   3
   4
   5
   6
octave:3> size(colonna)
ans =

   6   1
```

Matrici

Altri operatori

`fliplr(x)` ritorna una matrice con l'ordine delle colonne scambiato

`flipud(x)` ritorna una matrice con l'ordine delle righe scambiato

`rot90(x,n)` ritorna una matrice ruotata in senso antiorario n volte. Se n negativo allora ruota in senso orario. L'argomento n è facoltativo, se non specificato prende il valore 1

Elementi di una matrice

Singoli elementi di una matrice possono essere isolati e usati in espressioni come variabili scalari con la notazione A(i,j) dove i è l'indice di riga e j l'indice di colonna

```
octave:1> matrice=[1 2 -3; -4 5 -6; 7 -8 9]
matrice =

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

octave:2> matrice(2,3)
ans = -6
octave:3> 5*matrice(2,3)
ans = -30
octave:4> matrice(2,3) = 5 * matrice(2,3)
```

Elementi di un Vettore

Anche se i vettori sono casi speciali di matrici è sufficiente un solo indice come riferimento ad un elemento di essi

Non c'è differenza tra vettori riga e vettori colonna

```
octave:1> v=[1 2 3 4 5 6 7 8 9 10]
v =

    1    2    3    4    5    6    7    8    9   10

octave:2> v(5)
ans =  5
octave:3> v(9)
ans =  9
octave:4> v(11)
error: A(I): index out of bounds; value 11 out of bound 10
```

Elementi Sparsi di Vettori e Matrici

Il linguaggio di Matlab e Octave hanno una notazione molto potente per selezionare parti, anche sparse, di una matrice

Elementi Sparsi di Vettori e Matrici

Esempio: clamping dei valori di una matrice nell'intervallo [0-255]

```
>> r
r =
    41   256    23   126   147
   261   187    72    15   102
   174   106    37   271   271
   165   154    56   284   111
    44   121    72   148    34
>> grt = r > 255
grt =
     0     1     0     0     0
     1     0     0     0     0
     0     0     0     1     1
     0     0     0     1     0
     0     0     0     0     0
>> r(grt) = 255;
>> r
r =
    41   255    23   126   147
   255   187    72    15   102
   174   106    37   255   255
   165   154    56   255   111
    44   121    72   148    34
```

Estrazione di Sottomatrici

Una porzione di matrice può essere estratta usando la notazione r1:r2 per indicare l'intervallo di righe e/o colonne da selezionare

```
octave:1> quadrato_magico=magic(5)
quadrato_magico =

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

octave:2> quadrato_magico(2:4,3:5)
ans =

    7   14   16
   13   20   22
   19   21    3
```


http://it.wikipedia.org/wiki/Quadrato_magico

Sequenze

La notazione n1:n2 serve a generare sequenze di interi compresi tra gli estremi indicati

```
octave:7> 2:4
ans =

   2   3   4

octave:8> 3:5
ans =

   3   4   5
```

Estrazione di sottomatrici

Notazione rapida

Estrarre tutti gli elementi della prima dimensione (righe) e un sottoinsieme della seconda

```
octave:8> quadrato_magico(1:end,1:2)
ans =

   17   24
   23    5
    4    6
   10   12
   11   18

In breve anche

octave:8> quadrato_magico(:,1:2)
```

Generazione di Vettori

La notazione di 'intervallo' di indici fornisce un modo elementare per generare vettori che contegono semplici sequenze numeriche

```
octave:6> v=[1:10]
v =

    1    2    3    4    5    6    7    8    9   10

octave:7> v=[10:2:30]
v =

   10   12   14   16   18   20   22   24   26   28   30

octave:8> v=[0.1:0.2:2]
v =

   0.10000   0.30000   0.50000   0.70000   0.90000   …
1.10000   1.30000   1.50000   1.70000   1.90000
```

Composizione di Vettori e Matrici

Concatenazione: un modo di comporre tra loro matrici più grandi a partire da altri elementi (matrici o vettori)

Utile quando si costruisce un vettore in uno script del quale non si sa a priori la dimensione finale

Nel campo dell'image processing consente di comporre con facilità immagini

Si deve essere rigorosi però con le dimensioni degli oggetti che vengono composti tra loro

Concatenazione di Vettori

Esempio: costruzione di un vettore riga

```
% vettore vuoto
octave:1> V = [ ];

% concateniamo un elemento alla volta
octave:2> V = [V 1];
octave:3> V = [V 2];
octave:4> V = [V 3]
V =

   1   2   3

% concateniamo la riga a se stessa usando il punto e virgola
octave:5> V = [V; 2*V; 3*V]
V =

   1   2   3
   2   4   6
   3   6   9
```

Concatenazione di Matrici

Ancora un esempio con matrici 2x2

```
octave:1> A=[1 2; 9 3];
octave:2> B=[2 5; 4 3];
octave:3> [A  B]
ans =

   1   2   2   5
   9   3   4   3

octave:4> [A; B]
ans =

   1   2
   9   3
   2   5
   4   3
```

Concatenazione con ‘cat’

- La funzione ‘cat’ fornisce un metodo generale per unire vettori e matrici
- Il primo argomento della funzione è un intero che indica secondo quale indice la concatenazione deve avvenire
- Un esempio di applicazione: la costruzione di un immagine RGB da 3 immagini grayscale

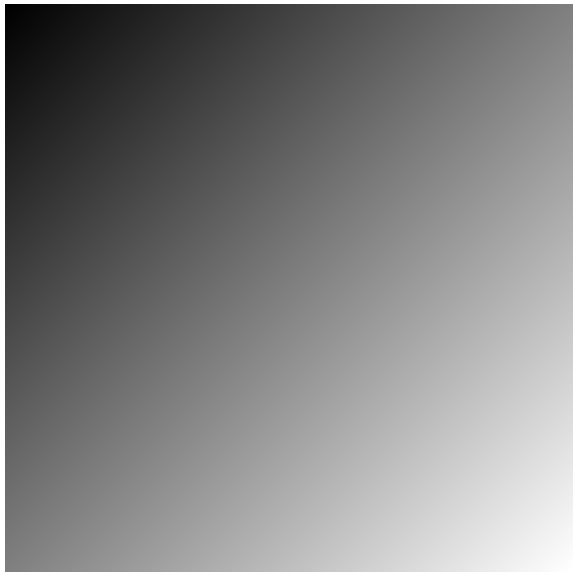
Concatenazione con ‘cat’

```
>> help cat
 cat Concatenate arrays.
    cat(DIM,A,B) concatenates the arrays A and B along
    the dimension DIM.
    cat(2,A,B) is the same as [A,B].
    cat(1,A,B) is the same as [A;B].

    B = cat(DIM,A1,A2,A3,A4,...) concatenates the input
    arrays A1, A2, etc. along the dimension DIM.
```

Concatenazione con ‘cat’

```
>> [xx,yy]=meshgrid(1:512,1:512);
>> z = mat2gray(xx+yy);
>> imshow(z)
>> rgb=cat(3,z,rot90(z),rot90(z,2));
>> imshow(rgb)
```



Operazione su vettori o matrici

Le funzioni di Octave/Matlab agiscono sui vettori / matrici in modi differenti che dipendono dal contesto e dalla funzione

Quando ha senso agiscono su tutti i valori di un vettore (o matrice) e restituiscono un altro vettore (o matrice)

Esempio: operazione su sequenza numeri compresi tra -5 e 5 con incremento 0.5

```
octave:18> x=[-5:0.5:5];
octave:19> y=tanh(x)
 y =

 Columns 1 through 13:

  -0.99991  -0.99975  -0.99933  -0.99818  -0.99505  -0.98661…
  -0.96403  -0.90515  -0.76159  -0.46212   0.00000   0.46212   0.76159

 Columns 14 through 21:

   0.90515   0.96403   0.98661   0.99505   0.99818   0.99933…
0.99975   0.99991
```

Modalità di esecuzione delle funzioni

`sum` ritorna la somma degli elementi di un array.

Se l'argomento è una matrice opera su di essa sommando gli elementi delle colonne e ritornando un vettore riga.

```
octave:2> quadrato_magico = magic(5);
octave:3> sum(quadrato_magico)
ans =

   65   65   65   65   65
```

Esempio: ‘max’ e ‘min’

Le funzioni `max` e `min` hanno vari modi di elaborazione

1, 2 o 3 parametri di input

1 o 2 valori di output

Con un solo argomento di *output* e un solo vettore come *input* max (min) ritorna il valore massimo (minimo) nel vettore

Se invocata con 2 argomenti di ritorno allora nella seconda variabile viene scritto l'indice nell'array al valore massimo (minimo)

```
octave:1> min([1 4 -3 2 -6 10])
ans = -6
octave:2> [minv,mind] = min([1 4 -3 2 -6 10])
minv = -6
mind =  5
```

Esempio: ‘max’ e ‘min’

Se l'argomento è una matrice allora viene ritornato un vettore riga contentente i minimi di ogni colonna

Con 2 matrici di identiche dimensioni la funzione ritorna una matrice contente il valore massimo (minimo) di elementi corrispondenti (non ammette 2 valori di ritorno)

Esercizio: digitate 'help min' per avere una descrizione del comportamento delle funzioni quando gli argomenti di input sono 2 o 3

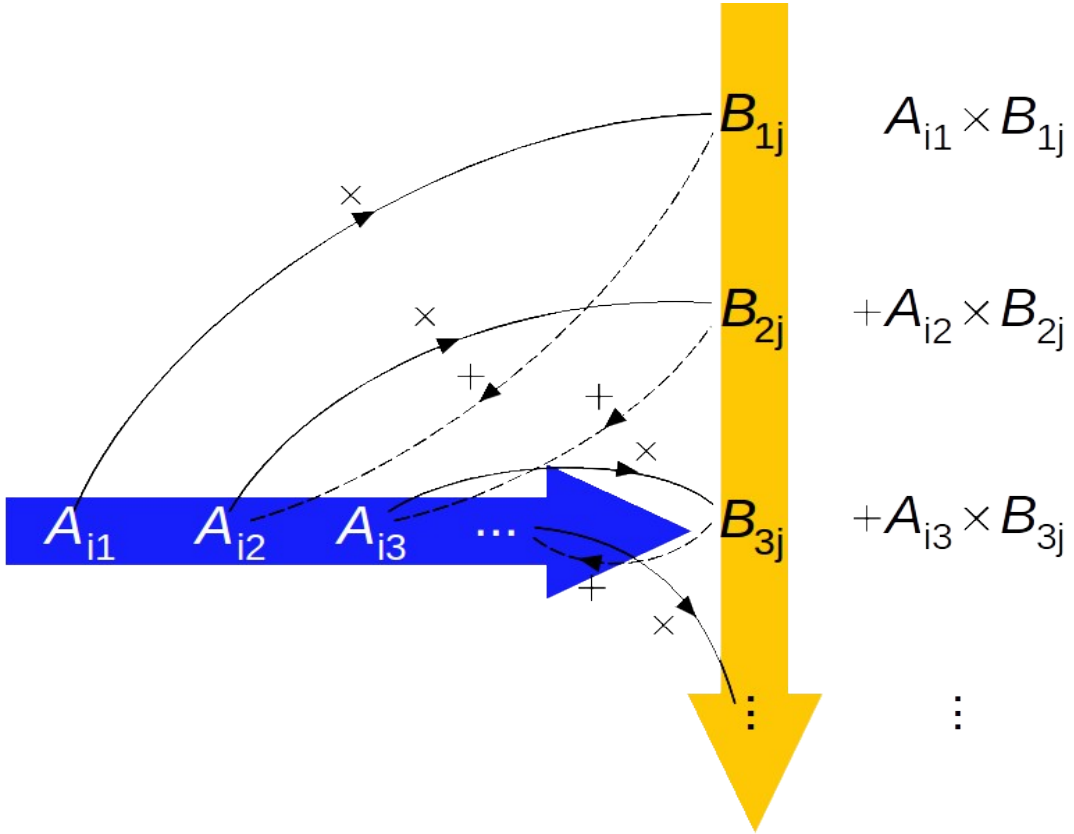
Moltiplicazione di Matrici

L'operazione di moltiplicazione tra matrici è detto **prodotto righe per colonne**

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1p} \\ B_{21} & B_{22} & \cdots & B_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mp} \end{pmatrix}$$

Moltiplicazione Righe per Colonne

$$(AB)_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}.$$



Moltiplicazione Righe per Colonne

Per Octave/Matlab la moltiplicazione righe per colonne è la moltiplicazioni 'naturale' tra matrici.

Per essa viene usato semplicemente l'operatore *

In generale (quando i fattori sono matrici quadrate) la moltiplicazione righe x colonne non è in generale commutativa, ma è associativa

```
octave:19> A
A =

    1.0000   -5.4000    3.0000
    2.0000    2.8000    9.0000
   12.0000    5.0000    3.0000

octave:20> B
B =

    7.0000   11.0000   13.3000
    2.0000    9.0000   10.6000
    6.0000    6.2000    2.5000

octave:21> A*B
ans =

    14.200   -19.000   -36.440
    73.600   103.000    78.780
   112.000   195.600   220.100
```

Matrice Unità

$$I_1 = \begin{bmatrix} 1 \end{bmatrix},\ I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ \cdots,\ I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

Per una matrice A con n righe e m colonne

$$I_m \cdot A = A \cdot I_n = A$$

Matrice Inversa

$$A \cdot B = B \cdot A = I_n$$

Definita per matrici quadrate

In analogia con gli scalari la matrice inversa è indicata dalla notazione **`A^-1`**

La funzione che ritorna la matrice inversa (se esiste) in Octave/Matlab è `inv(A)`

Esempio

La trasformazione RGB a YC_bC_r e la sua inversa

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1.000 & 0.000 & 1.403 \\ 1.000 & -0.344 & -0.714 \\ 1.000 & 1.773 & 0.000 \end{pmatrix} \cdot \begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix}$$

Moltiplicazione con Vettori

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

$$A \cdot x = b$$

Sistemi di Equazioni

$$\begin{aligned} 3x + 2y - \ \ z &= \ \ 1 \\ 2x - 2y + 4z &= -2 \\ -x + \tfrac{1}{2}y - \ \ z &= \ \ 0 \end{aligned}$$

```
octave:23> A = [3 2 -1; 2 -1 4; -1 0.5 -1]

A =

   3.00000   2.00000  -1.00000
   2.00000  -1.00000   4.00000
  -1.00000   0.50000  -1.00000

octave:24> b = [1 -2 0]

 B = 1 -2 0
```

Soluzione di Sistemi Lineari

OCTAVE/MATLAB: Usare la funzione `inv(A)`:

$$Ax = b$$

$$A^{-1}A = AA^{-1} = \mathbb{1}$$

$$x = A^{-1}Ax = A^{-1}b$$

funzione `linsolve(A,b)`

`linsolve` usa diversi algoritmi per risolvere sistemi di equazioni

Proprietà Associativa

Esercizio

Generiamo 3 matrici quadrate a,b,c usando la funzione rand(n,m) (Esempio: n = m = 10)

Calcoliamo due matrici prodotto

M1 = a * (b * c);

M2 = (a * b) *c;

Osservate il contenuto della matrice differenza

d = m1 – m2

Osserviamo il valore massimo e minimo della matrice differenza con i comandi

```
max(d(:))

min(d(:))
```

Operazione Elemento x Elemento

Operazioni sui singoli elementi corrispondenti

Addizione: $(A+B)_{ij} = A_{ij} + B_{ij}$

Nel caso degli operatori aritmetici questo comportamento si ottiene modificando l' espressione dell'operatore

Esempi

A.*B: prodotto elemento per elemento

A./B: divisione

A.^B: elevazione a potenza

Generazione di Matrici Speciali

Funzioni di libreria per matrici speciali

`ones(n,m[,p])`: Ritorna una matrice n x m [x p] avente solo '1' come elementi

`zeros(n,m[,p])`: Ritorna una matrice n x m [x p] avente solo '0' come elementi

- Queste funzioni posso creare matrici anche con 3 indici. Matrici che hanno una 'terza' dimensione: saranno utili per costruire matrici che rappresentano immagini “vuote”

Funzioni per Matrici Particolari

- `diag(v)`: Compone una matrice diagonale usando il vettore v come elementi della diagonale
- Nella forma diag(v,n) produce matrici sopradiagonali o sottodiagonali
- La matrice unità di ordine 5 si può generare così

```
octave:1> diag(ones(1,5))
ans =

Diagonal Matrix

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

Avvertenze sulla rappresentazione interna

```
octave:1> unsigned = uint16(100)
unsigned = 100
octave:2> class(unsigned)
ans = uint16
octave:3> whos
Variables in the current scope:

   Attr Name        Size                     Bytes  Class
   ==== ====        ====                     =====  =====
        ans         1x6                          6  char
        unsigned    1x1                          2  uint16

Total is 7 elements using 8 bytes

octave:4> unsigned/2
ans = 50
octave:5> unsigned = unsigned + 1
unsigned = 101
octave:6> unsigned/2
ans = 51
octave:7> double(unsigned)/2
ans =  50.500
```

Avvertenze sulla rappresentazione interna

Effetti di un numero negativo su `unsigned`

```
octave:10> unsigned
unsigned = 101
octave:11> unsigned=-1
unsigned = -1
octave:12> class(unsigned)
ans = double
octave:13> uint16(unsigned)
ans = 0
octave:14> unsigned=uint16(unsigned)
unsigned = 0
octave:15> class(unsigned)
ans = uint16
```

Altri Tipi di Dati: stringhe

Altri formati di dati

–Stringhe: sequenze di caratteri. I singoli caratteri possono essere isolati come elementi di un array

- Le stringhe devono essere delimitate dal carattere ‘
- Esistono funzioni per concatenare, confrontare e generare stringhe speciali
- Funzioni per creare matrici da stringhe e viceversa
- Le stringhe sono indispensabili in alcune funzioni come `printf`
- Con argomenti tipo stringa si può controllare l'output della funzione `plot`

Altri Tipi di Dati: strutture

Raccolgono dati anche disomogeni tra loro che descrivono però la stessa entità

- Utili per creare 'oggetti'
- Esempio: descrizione di un cerchio in un diagramma cartesiano

```
octave:1> cerchio.x = 1.5;
octave:2> cerchio.y = -0.5;
octave:3> cerchio.raggio = 2;
octave:4> cerchio
cerchio =

  scalar structure containing the fields:

    x =  1.5000
    y = -0.50000
    raggio =  1

octave:5>
```

Strutture

Possono essere create con la funzione struct

```
>> c=struct('x',1,'y',2,'raggio',1)
c =

  scalar structure containing the fields:

    x =  1
    y =  2
    raggio =  1
```

Altri Tipi di Dati: Cell Arrays

Cell Arrays sono come vettori che contengono dati anche disomogeni tra loro (scalari, matrici, stringhe) che vengono indicizzati con un numero come per gli array

Un Cell Array viene creato con l'operatore {...} e usa la stessa notazione per accedere ai singoli elementi dell'array

```
% cell array di dimensione 3x2 contenente scalari, stringhe e matrici

myCell = {1, 2, 3; 'text', rand(5,10,2), {11; 22; 33}}
>> myCell{2,1}
ans = text
>> myCell{2,3}
ans =
{
  [1,1] =  11
  [2,1] =  22
  [3,1] =  33
}
```

Workspace

Ogni riferimento simbolico mantiene un riferimento all'interno del workspace di Octave

Un elenco delle variabili registrate nel workspace può essere ottenuto con il comando whos

```
octave:9> whos
Variables in the current scope:

   Attr Name        Size                     Bytes  Class
   ==== ====        ====                     =====  =====
        a           1x1                          8  double
        ans         1x70                       763  cell
        esp         1x1                          8  double
        tang_ip     1x1                          8  double

Total is 72 elements using 779 bytes
```

Workspace

Il workspace di Matlab/Octave può essere salvato con il comando save

In una sessione di lavoro successiva il workspace salvato può essere letto con il comando load

Le pagine di manuale per save e load

Octave

https://octave.org/doc/v4.2.1/Simple-File-I_002fO.html

Matlab

https://it.mathworks.com/help/matlab/ref/save.html

https://it.mathworks.com/help/matlab/ref/load.html

Workspace

Esercizio:

Dopo aver letto le pagine di manuale di “save” e “load” salvate in un file alcune delle variabili del vostro workspace

Cancellate il workspace con il comando “clear”

Verificate con “whos” l’effetto del comando precedente

Rigenerate il workspace con “load”

‘save’ - ‘load’ su Matlab

Salva / Carica il workspace delle variabili

Permettono di salvare (e caricare successivamente) in modo selettivo il contenuto di variabili presenti nel workspace

Ci sono differenze tra Matlab e Octave

Formati di file utilizzati: ASCII (caratteri), ‘MAT-file’ (binario)

```
save(filename)
save(filename,variables)
save(filename,variables,fmt)
save(filename,variables,version)
save(filename,variables,version,'nocompression')
save(filename,variables,'-append')
save(filename,variables,'-append','-nocompression')
```

```
load(filename)
load(filename,variables)
load(filename,'-ascii')
load(filename,'-mat')
load(filename,'-mat',variables)
```

‘save’ - ‘load’ su Matlab

Nella sua forma più semplice ‘save’ ammette una lista di argomenti separati da spazi

```
riso=imread(‘Rice-small.png’);
risogr = rgb2gray(riso);
riso = mat2gray(risogr);

save workspace-riso.mat risogr riso
```

Le forme del comando più articolate hanno la struttura delle funzioni

Permettono di controllare il formato del file

Nel caso binario permettono di disabilitare la compressione

```
save ('myworkspace.mat','riso','risogr')
```

Funzioni per scambio di dati

Utility ‘importdata’

Usa una euristica di criteri per determinare quale tipo di file state leggendo

Grafica

XML

Audio

File in formato tabulare (eg: esportati da Excel o Access)

```
A = importdata(filename)
A = importdata('-pastespecial')
A = importdata(___,delimiterIn)
A = importdata(___,delimiterIn,headerlinesIn)
[A,delimiterOut,headerlinesOut] = importdata(___)
```