MATLAB

```
fprintf(‘ quello che vuoi ’)
warning (‘ qualcosa è andato storto ’)
a = input(Inserisci il dato:’);

clear //cancella workspace e non command window
who //indica le variabili
clc //pulisce la command window ma non il workspace o il diary

eps = 2^(1-53)

round(X, N) arrotondamento per le prime N cifre decimali
round(X, N, “significant”)  arrotondamento per le prime N cifre (anche non decimali),
normalizza

abs: calcola il valore assoluto

Calcolo dell’errore relativo
                rho = abs(RisultatoFinaleFlottato - RisultatoRealeNonFlottato) /
                                abs(RisultatoRealeNonFlottato)

switch n
        case 1
                fprintf(‘’)
        case 2
                fprintf(‘’)
        case 3
                fprintf(‘’)
        otherwise
                warning(‘’)
end

if condizione
        blocco di istruzioni
elseif
        blocco di istruzioni
else
        blocco di istruzioni
end


for i=val iniziale : passo : val finale


while (condizione)
        blocco di istruzioni
end
```

Matrici

Trasposta: aggiungi alla fine della matrice/vettore l'apostrofo
//NB: l'apice fa la trasposta dei complessi coniugati, quindi se la matrice è reale allora risulta uguale alla trasposta

length() restituisce il numero di elementi del vettore
size() restituisce il numero di righe e il numero di colonne
Matrice identità
I = eye(DIM)

Matrice di zero
A = zeros(RIGHE,COLONNE) //Matrice rettangolare
A = zeros(N) oppure A = zeros(N,N) //Matrice quadrata

Matrice di uno
A = ones(RIGHE,COLONNE) //Matrice rettangolare
A = ones(N) oppure A = zeros(N,N) //Matrice quadrata

Matrice di 23
A = 23 * ones(RIGHE,COLONNE)

Generazione di numeri pseudocasuali
R = rand(RIGHE,COLONNE) //Genera una matrice RIGHExCOLONNE con elementi casuali tra **0 e 1**
R = (b-a).*rand(n)+a //Genera numeri casuali nell'intervallo a e b

Generazione di numeri interi pseudocasuali
R = randi ([MIN,MAX], RIGHE, COLONNE)
S = randi(MAX,RIG,COL) //in questo caso, non avendo messo il primo numero dell'intervallo, lui mette 0

T = randn(RIGHE,COLONNE) //n sta per normal distribution, ovvero la gaussiana a campana. Si ha probabilità maggiore nel valor medio, e la prob è minore allontanandoci da esso
//in questo caso abbiamo molti valori vicini a zero, perché la media è quella
//se volessi cambiare il punto centrale, basta sommare a randn

Operazioni tra matrici/vettori

A = B + C //matrici

a = b + c //vettori

//nota bene: matlab fa operazioni non consentite! Ovvero somma un vettore colonna ad un vettore riga

Se vuoi effettuare operazioni elemento per elemento, ad esempio una divisione tra vettori, metti il . prima dell'operatore: b ./ d

Moltiplicazione tra matrici
Quando si fanno moltiplicazioni tra due matrici di dimensione diversa, l'operazione si può effettuare solo se i due numeri interni sono uguali
ES
T = 2x3 B=3x3
T*B = va bene B*T = non va bene (3x**3** * **2**x3)

Potenza di matrici quadrate (NO vettori e matrici non quadrate)
C^2 = C*C
NB: non è uguale a fare ogni elemento della matrice al quadrato

Potenza elemento per elemento (anche per matrici non quadrate e vettori)
C.^2

Divisioni
A/B = A*INV(B) //questo comando fa A moltiplicato l'inverso di B
A\B = INV(A)*B //questo comando fa l'inverso di A per B

Estrazione di elementi dalle matrici
C(1,2) //estraggo l'elemento riga 1 col 2

NB: le matrici partono dall'elemento 1 non dallo zero

Proprietà delle colonne
C(2,1:5) //Estraggo riga 2, colonne dalla 1 alla 5
C(2,:) //per estrarre tutte le colonne posso anche omettere l'indice di colonna
C(2,1:end) //posso anche dire di estrarre dall'elemento 1 a end
C(2,3:end) //posso anche estrarre porzioni di colonna, ad esempio da 3 alla fine

Le stesse proprietà delle colonne possono essere utilizzate per le righe
C(:,3) //ovviamente gli indici possono essere dei range anche per le righe

C([2,4],2:3) //posso anche selezionare elementi non contigui, ad esempio la riga 2 e la riga 4 e le colonne da 2 a 3

find(C>0.5) //restituisce un vettore colonna con i numeri degli elementi(NON IL VALORE!)in cui il valore è maggiore di 0.5 (Contando gli elementi in verticale, partendo da 1)

1	4	7
2	5	8
3	6	9

[i,j] = find(C>0.5) //In questo modo ci restituisce due vettori colonna, uno per le righe e uno per le colonne, delle coordinate degli elementi

Concatenazione di elementi - scrittura di colonne (matrici e vettori)

[i,j]

Concatenazione di elementi - scrittura di righe (matrici e vettori)
[i;j] //in questo caso ottieni un vettore unico di dimensione i+j

ES
Q = [C,x] //posso affiancare C con x perché hanno lo stesso numero di righe
Q = [C;w] //posso affiancare w sotto C, perché w è un vettore riga

Matrice complessa

//costruisco una matrice formata da elementi complessi sommando tra loro una matrice intera e una matrice formata da numeri immaginari

R = rand(5) + i*rand(5)

Facendo R' faccio l'operazione del complesso coniugato (cambio segno alla parte immaginaria)

Facendo R.' faccio solo il trasposto senza fare il complesso coniugato

v = C > 0.5
//ottengo una matrice di elementi logici, con 0 dove non è verificata la condizione, 1 dove è verificata

C^(-1)
//maniera + stabile per calcolare l'inversa

C\eye(5)
//inversa di C moltiplicato identità = inversa di C. Useremo questo metodo

det(C)
//Determinante di C

rank(A)
//Rango di A

norm(x,1)
//Calcolo della norma 1 del vettore x

norm(x,inf)
//Calcolo della norma infinito del vettore x

norm(x,2) o norm(x)
//Calcolo della norma 2 del vettore x

norm(x,3/2)
//Calcolo della norma 3/2 del vettore x

cond(matrice, indice di condizionamento)

//calcolo del numero di condizionamento: indica quanto un errore sui dati influisce sulla soluzione

Simmetria

issymmetric(A)
//Se restituisce 1 la matrice è simmetrica, altrimenti restituisce 0

A = A' * A
//Forziamo la simmetria

eig(A)
//Calcolo autovalori, genera un vettore colonna

[V,D] = eig(A)
//Calcolo degli autovettori
//V contiene in ogni colonna gli autovettori
//D è una matrice diagonale che contiene gli autovalori

//A questo punto se facessi V*D*V' otterrei A
//Per controllare se due matrici/vettori sono uguali, faccio la norma infinito della differenza:
norm(A - V*D*V' , inf)
//Se il risultato è molto vicino allo zero, le due matrici/vettori sono uguali

Calcolo del raggio spettrale: è il massimo degli autovalori in valore assoluto
rho = max(abs(eig(B)))

Se una matrice è simmetrica, raggio spettrale = norma 2

Funzione max()
Calcola il massimo, se applicato a un vettore mi restituisce il valore dell'elemento massimo
Se mi interessa la posizione in cui si trova il massimo mi faccio restituire due output
[mx,imx] = max(x)
//mx è il valore massimo
//imx è la posizione nel vettore (ad esempio elemento 4)

Se applico max a una matrice mi restituisce il massimo di ogni colonna
Se chiamo max con il doppio output, restituisce il massimo di ogni colonna e la posizione in ogni colonna
[mxB , imxB] = max(B)

Per trovare il massimo valore di una matrice:
max(max(B))

Per calcolare il massimo di ogni riga, faccio il max della trasposta di B

max(B')

Matrice hermitiana: ishermitian(C) //Se restituisce 1 lo è, altrimenti 0

Matrice aggiunta, 2 modi:

1) C = rand(3) - i*rand(3) e dopo C.'
2) C = rand(3) + i*rand(3) e dopo C'

Per creare la matrice hermitiana, devo fare la moltiplicazione tra la matrice aggiunta e la matrice originale: E = C' * C

Matrici ortogonali

Nelle matrici ortogonali la trasposta coincide con l'inversa quindi, facendo l'ortogonale di una matrice:

Q = orth(B)

e poi:

Q * Q' (Q * Q trasposta = Q * Q inversa)

otteniamo la matrice identità

Proprietà delle matrici ortogonali

- Il determinante di una matrice ortogonale può essere +-1
- creo un vettore colonna casuale: x = randn(5,1)
  Se calcolo la norma di un vettore, e successivamente calcolo la norma del vettore moltiplicato con quella di una matrice ortogonale, il risultato della norma sarà uguale in entrambi i casi.

Matrice triangolare

Vengono generate da una matrice di partenza

U = triu(B) //triangolare superiore
L = tril(B) //triangolare inferiore

Posso usare queste funzioni anche con due input
L = tril(B,1) //in questo caso, sopra la diagonale principale c'è un'altra diagonale diversa da zero.

L = tril(B,-1) //vale anche per numeri negativi, per eliminare diagonali visibili. Nel caso -1 ho la diagonale principale a 0

Per verificare se la matrice è triangolare con output 1/0

istriu(U) //verificare se è triangolare superiore
istril(U) //verificare se è triangolare inferiore

//ATTENZIONE: se ottengo una matrice tramite calcolo, non avrà mai gli elementi a 0 perfetto, ma avrà un numero molto vicino allo 0!
infatti, se almeno uno dei termini è un numero vicino allo zero e non 0, istriu e istril restituiscono 0
<u>Quindi il verificatore di triangolarità lo riscriveremo a mano</u>

Proprietà matrici triangolari:

- autovalori sono gli elementi sulla diagonale
- determinante è il prodotto degli autovalori

Matrici diagonali

il comando diag si comporta in maniera diversa se applicato alle matrici o ai vettori
-se gli diamo una matrice come argomento: diag crea un vettore che contiene gli elementi diagonali
-se diamo un vettore come argomento: diag crea una matrice con il vettore come diagonale

Quindi per ricavare la matrice diagonale:

diag(diag(B))

Per controllare se la matrice è diagonale:

isdiag(D)

ATTENZIONE: non funziona con numeri molto piccoli, funziona solo se gli altri numeri sono effettivamente 0

GRAFICI

ho due vettori colonna **x** e **y**, per creare il grafico:

plot(x,y)

A questo punto il grafico sarà spezzettato, per ottenere un andamento più curvo dobbiamo usare la funzione linspace(primo estremo, secondo estremo, punti di divisione intervallo)
ATTENZIONE: trasporre il tutto altrimenti otterremo un vettore riga

x = (linspace(-pi, pi, 10))'

a quel punto ricalcoli y in base al valore di x e ottieni un grafico più curvo
Più aumenti i punti di divisione intervallo, maggiormente curvo sarà il grafico

Per aggiungere le proprietà al grafico:

plot(x,y, 'proprietà')

Proprietà utilizzabili

b	blue	.	point	-	solid

---

g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
w	white	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

Per scrivere più di una funzione nello stesso grafico:

plot(x,y, 'proprietà1', *x, y2, 'proprietà2'*)

Per capire come distinguere i grafici, bisogna scrivere sempre la legenda con il plot aperto:

legend('prima funzione',*'seconda funzione'*)

Per aggiungere un titolo al plot:

title(‘funzioni trigonometriche’)

Per mettere le etichette agli assi:

xlabel(‘x’)
ylabel(‘y’)

Sistemi lineari

Funzioni

script della funzione:

function [output1, output2] = nome_scriptfunction(input1,input2)

//NB: tutto ciò che non metti in output, non puoi utilizzarlo / visualizzarlo nella command window

script principale:

[output1, output2] = nome_scriptfunction(input1, input2)

l’intestazione della function è l’intestazione generale, che deve corrispondere alle variabili usate ALL’INTERNO della funzione. Gli output e gli input devono avere gli stessi nomi di quelli all’interno della funzione.
Se però richiamiamo la funzione con variabili con diverso nome, matlab si basa sulla posizione dei parametri.

Creazione tabelle

- allocare un array: tab = zeros(10,2)
- inserire gli elementi a ogni ciclo : tab(i, :) = [i,err]
- all’uscita del ciclo: tabella = array2table(tab,”VariableNames”,{‘Dimensione’, ‘Errore’})
- Stampa la tabella: disp(tabella)