

MATLAB RIFERIMENTO RAPIDO

Array, matrici, grafici, I/O su file, controllo del flusso

Basi

Finestra di Comando

```
x = 5; % assign (semicolon suppresses output)
x = 5; % assign and display result
disp('Hello') % print to console
clc % clear command window
clear % clear all variables
```

Aiuto e Info

```
help sin % quick help for function
doc sin % open documentation
who % list variables in workspace
whos % list with details (size, type)
```

Operatori

```
+ - * / ^ Aritmetica (operazioni tra matrici)
.* ./ ./ ./ ./ Operazioni elemento per elemento
== ~= < > <= >= Operatori di confronto
&& | ~ AND, OR, NOT logici (scalari)
& | ~ AND, OR, NOT elemento per elemento (array)
```

Variabili e Tipi

Tipi Numerici

```
x = 3.14; % double (default)
n = int32(42); % 32-bit integer
z = 2 + 3i; % complex number
tf = true; % logical
```

Verifica del Tipo

```
class(x) Restituisce il nome del tipo come stringa
isa(x, 'double') Verifica se x è di un tipo specifico
isnumeric(x) Vero se tipo numerico
ischar(x) Vero se array di caratteri
islogical(x) Vero se tipo logico
```

Costanti Speciali

```
pi 3.14159...
Inf / -Inf infinito
NaN Not a Number
eps Epsilon macchina (~2.2e-16)
i / j Unità immaginaria
```

Array e Matrici

Creazione di Array

```
v = [1 2 3 4 5]; % row vector
v = [1; 2; 3]; % column vector
A = [1 2; 3 4]; % 2x2 matrix
r = 1:5; % [1 2 3 4 5]
r = 0:0.5:2; % [0 0.5 1 1.5 2]
```

Costruttori Built-in

```
zeros(3) % 3x3 of zeros
ones(2, 4) % 2x4 of ones
eye(3) % 3x3 identity
rand(2, 3) % 2x3 uniform random
linspace(0,1,5) % 5 evenly spaced [0..1]
```

Indicizzazione e Slicing

```
A(2, 3) % row 2, col 3
A(i, :) % entire first row
A(:, 2) % entire second column
A(1:2, 1:2) % submatrix
A(end, :) % last row
```

Operazioni su Matrici

```
A' Trasposta (coniugata)
A.' Trasposta (senza coniugato)
inv(A) Inversa della matrice
det(A) Determinante
eig(A) Autovalori e autovettori
A \ b Risolve Ax=b
size(A) Dimensioni [righe colonne]
numel(A) Numero totale di elementi
```

Controllo del Flusso

if / elseif / else

```
if x > 0
    disp('positive')
elseif x == 0
    disp('zero')
else
    disp('negative')
end
```

for e while

```
for i = 1:10
    fprintf('i = %d\n', i);
end
while x > 0
    x = x - 1;
end
```

switch

```
switch grade
    case 'A'
        disp('Excellent')
    case {'B', 'C'}
        disp('Good')
    otherwise
        disp('Try harder')
end
```

Controllo dei Cicli

```
break Esci dal ciclo più interno
continue Salta all'iterazione successiva
return Esci immediatamente dalla funzione
```

Funzioni

File di Funzione

```
% Save as myfunc.m
function result = myfunc(x, y)
    result = x.^2 + y.^2;
end
```

Output Multipli

```
function [m, mx] = minmax(v)
    mn = min(v);
    mx = max(v);
end
```

```
[lo, hi] = minmax([3 1 4 1 5]);
```

Funzioni Anonime

```
f = @(x) x.^2 + 1; % returns 10
f(3)
g = @(x,y) x + y;
arrayfun(f, [1 2 3]) % apply to each element
```

Built-in Utili

```
sum(v) Somma degli elementi
mean(v) Valore medio
max(v) / min(v) Massimo / minimo
sort(v) Ordina in modo crescente
find(v > 3) Indici dove la condizione è vera
length(v) Lunghezza del vettore
```

Grafici

Grafici 2D

```
x = 0:0.1:2*pi;
plot(x, sin(x), 'r-', 'LineWidth', 2)
xlabel('x'); ylabel('sin(x)')
title('Sine Wave'); grid on
legend('sin(x)')
```

Grafici Multipli

```
hold on
plot(x, sin(x), 'b-')
plot(x, cos(x), 'r--')
hold off
subplot(1,2,1); plot(x, sin(x))
subplot(1,2,2); plot(x, cos(x))
```

Altri Tipi di Grafico

```
bar(x, y) Grafico a barre
histogram(data) Istogramma
scatter(x, y) Grafico a dispersione
pie(data) Grafico a torta
surf(X, Y, Z) Grafico 3D di superficie
imagesc(A) Visualizza matrice come immagine
```

Salva Figura

```
saveas(gcf, 'plot.png')
exportgraphics(gcf, 'plot.pdf')
```

I/O su File

File di Testo

```
data = readmatrix('data.csv');
writematrix(A, 'output.csv')
T = readtable('data.csv');
writetable(T, 'output.csv')
```

File MAT

```
save('workspace.mat') % save all variables
save('data.mat', 'x', 'y') % save specific vars
load('data.mat') % load into workspace
S = load('data.mat'); % load into struct
```

I/O su File di Basso Livello

```
fid = fopen('log.txt', 'w');
fprintf(fid, 'Value: %f\n', 3.14);
fclose(fid);
lines = readlines('log.txt');
```

Operazioni su Stringhe

String vs Array di Caratteri

```
s = "Hello"; % string (double quotes)
c = 'Hello'; % char array (single quotes)
s + ' World' % "Hello World" (string)
[c, ' World'] % 'Hello World' (char concat)
```

Funzioni per Stringhe

```
strlength(s) Lunghezza della stringa
upper(s) / lower(s) Conversione maiuscolo/minuscolo
contains(s, pat) Vero se il pattern è trovato
replace(s, old, new) Sostituisce sottostringa
split(s, delim) Divide in array
join(arr, delim) Unisce array di stringhe
strip(s) Rimuove spazi iniziali/finali
```

Formattazione

```
sprintf('x = %.2f', 3.14159) % "x = 3.14"
fprintf('i = %d\n', 42) % print to console
num2str(3.14) % number to string
str2double("3.14") % string to number
```

Cell e Struct

Array Cell

```
C = {1, 'hello', [1 2 3]}; % mixed types
C{2} % access: 'hello'
C{end+1} = true; % append element
cellfun(@length, C) % apply func to each
```

Struct

```
s.name = 'Alice';
s.age = 30;
s.scores = [90 85 92];
fieldnames(s) % {'name', 'age', 'scores'}
rmfield(s, 'age') % remove field
```

Array di Struct

```
people(1).name = 'Alice'; people(1).age = 30;
people(2).name = 'Bob'; people(2).age = 25;
{people.name} % {'Alice', 'Bob'}
{people.age} % {30, 25}
```

Pattern Comuni

Operazioni Vettorializzate

```
% Avoid loops - use vectorization
v = 1:1000;
result = sum(v.^2); % fast
idx = v(v > 500 & v < 600); % logical indexing
```

Operazioni su Tabelle

```
T = table([25;30], {'A';'B'}, 'VariableNames', ...
    {'Age','Grade'});
T.Age % access column
T(T.Age > 25, :) % filter rows
```

Gestione degli Errori

```
try
    result = riskyFunction(x);
catch ME
    fprintf('Error: %s\n', ME.message);
end
```

Misurazione del Tempo

```
tic
heavyComputation();
toc % prints elapsed time
```