

REFERENCIA RÁPIDA DE MATLAB

Arrays, matrices, gráficas, E/S de archivos, flujo de control

Fundamentos

Ventana de comandos

```
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
```

Ayuda e información

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

Operadores

```
+ - * / ^ Aritmética (operaciones matriciales)
.* ./ ./ ./ ./ Operaciones elemento a elemento
== ~= < > <= >= Operadores de comparación
&& | ~ Lógico AND, OR, NOT lógico (escalares)
& | ~ Lógico elemento a elemento (arrays)
```

Variables y tipos

Tipos numéricos

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

Verificación de tipo

```
class(x) Retorna el nombre del tipo como cadena
is(x, 'double') Verificar si x es un tipo específico
isnumeric(x) Verdadero si es tipo numérico
ischar(x) Verdadero si es array de caracteres
islogical(x) Verdadero si es tipo lógico
```

Constantes especiales

```
pi 3.14159...
Inf / -Inf Infinito
NaN No es un número
eps Epsilon de máquina (~2.2e-16)
i / j Unidad imaginaria
```

Arrays y matrices

Crear arrays

```
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]
```

Constructores integrados

```
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]
```

Indexado y 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
```

Operaciones matriciales

```
A.' Transpuesta (conjugada)
A. Transpuesta (sin conjugar)
inv(A) Inversa matricial
det(A) Determinante
eig(A) Valores y vectores propios
A \ b Resolver Ax=b
size(A) Dimensiones [filas cols]
numel(A) Número total de elementos
```

Flujo de control

if / elseif / else

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

for y 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
```

Control de bucles

```
break Salir del bucle más interno
continue Saltar a la siguiente iteración
return Salir de la función inmediatamente
```

Funciones

Archivo de función

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

Múltiples salidas

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

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

Funciones anónimas

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

Funciones integradas útiles

```
sum(v) Suma de elementos
mean(v) Valor medio
max(v) / min(v) Máximo / mínimo
sort(v) Ordenar ascendente
find(v > 3) Índices donde la condición es verdadera
length(v) Longitud del vector
```

Gráficas

Gráficas 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)')
```

Múltiples gráficas

```
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))
```

Otros tipos de gráfica

```
bar(x, y) Gráfico de barras
histogram(data) Histograma
scatter(x, y) Diagrama de dispersión
pie(data) Gráfico circular
surf(X, Y, Z) Superficie 3D
imagesc(A) Mostrar matriz como imagen
```

Guardar figura

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

E/S de archivos

Archivos de texto

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

Archivos 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
```

E/S de archivos de bajo nivel

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

Operaciones con cadenas

String vs array de caracteres

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

Funciones de cadena

```
strlen(s) Longitud de la cadena
upper(s) / lower(s) Conversión de mayúsculas/minúsculas
contains(s, pat) Verdadero si se encuentra el patrón
replace(s, old, new) Reemplazar subcadena
split(s, delim) Dividir en array
join(arr, delim) Unir array de cadenas
strip(s) Eliminar espacios al inicio/final
```

Formato

```
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 y Struct

Arrays de celdas

```
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
```

Structs

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

Arrays de structs

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

Patrones comunes

Operaciones vectorizadas

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

Operaciones con tablas

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

Manejo de errores

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

Medir el tiempo de ejecución

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