RÉFÉRENCE RAPIDE MATLAB

Tableaux, matrices, tracés, E/S fichiers, flux de contrôle

Bases

Fenêtre de commande

```
x = 5;            % affecter (point-virgule supprime la sortie)
x = 5             % affecter et afficher le résultat
disp('Hello')     % afficher dans la console
clc               % effacer la fenêtre de commande
clear             % effacer toutes les variables
```

Aide et informations

```
help sin          % aide rapide pour la fonction
doc sin           % ouvrir la documentation
who               % lister les variables de l'espace de travail
whos              % lister avec détails (taille, type)
```

Opérateurs

`+ - * / ^`	Arithmétique (opérations matricielles)
`.* ./ .^`	Opérations élément par élément
`== ~= < > <= >=`	Opérateurs de comparaison
`&& || ~`	AND, OR, NOT logiques (scalaires)
`& | ~`	Logique élément par élément (tableaux)

Variables et types

Types numériques

```
x = 3.14;              % double (par défaut)
n = int32(42);         % entier 32 bits
z = 2 + 3i;            % nombre complexe
tf = true;             % logique
```

Vérification de type

`class(x)`	Retourner le nom du type en chaîne
`isa(x, 'double')`	Vérifier si x est d'un type spécifique
`isnumeric(x)`	Vrai si type numérique
`ischar(x)`	Vrai si tableau de caractères
`islogical(x)`	Vrai si type logique

Constantes spéciales

`pi`	3,14159...
`Inf / -Inf`	Infini
`NaN`	Pas un nombre
`eps`	Epsilon machine (~2,2e-16)
`i / j`	Unité imaginaire

Tableaux et matrices

Créer des tableaux

```
v = [1 2 3 4 5];         % vecteur ligne
v = [1; 2; 3];           % vecteur colonne
A = [1 2; 3 4];          % matrice 2x2
r = 1:5;                 % [1 2 3 4 5]
r = 0:0.5:2;             % [0 0.5 1 1.5 2]
```

Constructeurs intégrés

```
zeros(3)         % 3x3 de zéros
ones(2, 4)       % 2x4 de uns
eye(3)           % identité 3x3
rand(2, 3)       % aléatoire uniforme 2x3
linspace(0,1,5) % 5 valeurs équidistantes [0..1]
```

Indexation et découpage

```
A(2, 3)          % ligne 2, colonne 3
A(1, :)          % première ligne entière
A(:, 2)          % deuxième colonne entière
A(1:2, 1:2)      % sous-matrice
A(end, :)        % dernière ligne
```

Opérations matricielles

`A'`	Transposée (conjuguée)
`A.'`	Transposée (sans conjugaison)
`inv(A)`	Inverse de la matrice
`det(A)`	Déterminant
`eig(A)`	Valeurs propres et vecteurs propres
`A \ b`	Résoudre Ax = b
`size(A)`	Dimensions [lignes colonnes]
`numel(A)`	Nombre total d'éléments

Flux de contrôle

if / elseif / else

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

for et 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
```

Contrôle de boucle

`break`	Quitter la boucle la plus interne
`continue`	Passer à l'itération suivante
`return`	Quitter la fonction immédiatement

Fonctions

Fichier de fonction

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

Sorties multiples

```
function [mn, mx] = minmax(v)
    mn = min(v);
    mx = max(v);
end
[lo, hi] = minmax([3 1 4 1 5]);
```

Fonctions anonymes

```
f = @(x) x.^2 + 1;
f(3)                     % retourne 10
g = @(x,y) x + y;
arrayfun(f, [1 2 3])     % appliquer à chaque élément
```

Fonctions intégrées utiles

`sum(v)`	Somme des éléments
`mean(v)`	Valeur moyenne
`max(v) / min(v)`	Maximum / minimum
`sort(v)`	Trier par ordre croissant
`find(v > 3)`	Indices où la condition est vraie
`length(v)`	Longueur du vecteur

Tracés

Tracés 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)')
```

Tracés multiples

```
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))
```

Autres types de tracés

`bar(x, y)`	Diagramme en barres
`histogram(data)`	Histogramme
`scatter(x, y)`	Nuage de points
`pie(data)`	Diagramme circulaire
`surf(X, Y, Z)`	Surface 3D
`imagesc(A)`	Afficher la matrice comme image

Enregistrer une figure

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

E/S fichiers

Fichiers texte

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

Fichiers MAT

```
save('workspace.mat')          % enregistrer toutes les variables
save('data.mat', 'x', 'y')     % enregistrer des variables spécifiques
load('data.mat')               % charger dans l'espace de travail
S = load('data.mat');          % charger dans un struct
```

E/S fichiers bas niveau

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

Opérations sur les chaînes

String vs tableau de caractères

```
s = "Hello";        % string (guillemets doubles)
c = 'Hello';        % char array (guillemets simples)
s + " World"        % "Hello World" (string)
[c, ' World']       % 'Hello World' (concat char)
```

Fonctions de chaînes

`strlength(s)`	Longueur de la chaîne
`upper(s) / lower(s)`	Conversion de casse
`contains(s, pat)`	Vrai si le motif est trouvé
`replace(s, old, new)`	Remplacer une sous-chaîne
`split(s, delim)`	Diviser en tableau
`join(arr, delim)`	Joindre un tableau de chaînes
`strip(s)`	Supprimer les espaces en début/fin

Formatage

```
sprintf('x = %.2f', 3.14159)   % "x = 3.14"
fprintf('i = %d\n', 42)         % afficher dans la console
num2str(3.14)                   % nombre vers chaîne
str2double("3.14")              % chaîne vers nombre
```

Cell et Struct

Tableaux de cellules

```
C = {1, 'hello', [1 2 3]};  % types mixtes
C{2}                         % accès : 'hello'
C{end+1} = true;             % ajouter un élément
cellfun(@length, C)          % appliquer une fonction à chaque
```

Structs

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

Tableaux 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]
```

Modèles courants

Opérations vectorisées

```
% Éviter les boucles — utiliser la vectorisation
v = 1:1000;
result = sum(v.^2);          % rapide
idx = v(v > 500 & v < 600);  % indexation logique
```

Opérations sur les tables

```
T = table([25;30], ["A";"B"], 'VariableNames', ...
          {'Age','Grade'});
T.Age          % accéder à la colonne
T(T.Age > 25, :)  % filtrer les lignes
```

Gestion des erreurs

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

Chronométrer le code

```
tic
heavyComputation();
toc              % affiche le temps écoulé
```