# Getting Started with MATLAB

December 4, 2025

This introduction is an English translation of the original French version used for the "Travaux Pratiques" of the Numerical Analysis course held at the University of Geneva during the academic year 2019/2020.

General MATLAB documentation: https://www.mathworks.com/help/matlab/.

## 1 Manipulation of vectors and matrices

MATLAB can be used as a super-calculator that operates on matrices (real or complex numbers are considered as $1 \times 1$ matrices). Adding a semicolon at the end of a command line suppresses the display.

### Definition of vectors and matrices

```
>> x = [ 1, 2, 3 ]
>> y = [ 4; 5; 6 ]
>> Z = [ 1, 2, 3; 4, 5, 6 ]
>> t = 1:4
>> u = 0:0.1:1
```

### Special Matrices

```
>> I = eye(5,5)
>> X = zeros(5,3)
>> Y = zeros(size(y))
>> A = ones(6,2)
```

### Coefficient Extraction

```
>> Z(2,2)
>> x(2:3)
>> Z(1:1,1:2)
>> u(t)
>> Z(t(1:2),x(2))
>> Z(3)
>> Z(2,:)
>> Z(:,2)
```

### Linear Algebra Operations

```
>> Z'
>> x+y'
>> Z*y
```

### Element-wise Operations

```
>> x.^2
>> x./y'
>> 1./x
>> Z+1
```

### Operations with Matrices

```
>> A = [ 1 2 3; 0 0 1; 1 0 0 ]
>> B = [ -1 -2 -3; 0 0 -1; -1 0 0 ]
>> A*B
>> A.*B
>> A^2
>> A^(-2)
>> A'
```

### Functions on Matrices

```
>> det(A)
>> trace(A)
>> rank(A)
>> norm(A)
```

**Exercise 1.1** *Construct the $4 \times 4$ matrices $A$ and $B$ with coefficients $a_{ij}$, $b_{ij}$ defined by*

$$a_{ij} = \begin{cases} 2 & \text{if } i \neq j, \\ 1 & \text{if } i = j, \end{cases} \quad \text{and} \quad b_{ij} = j.$$

## 2 Plotting Tools

Powerful plotting tools are provided by MATLAB; a few examples are given here.

**Function Graphs**

```
>> x = linspace(0,4,100);
>> y = sin(x);
>> z = cos(x);
>> plot(x,y)
>> loglog(x,abs(y))
>> semilogx(x,y)
>> semilogy(x,y)
>> plot(x,z)
>> hold on
>> plot(x,y)
>> clf
>> plot(x,y,x,z)
>> clf
>> plot(x,y,'r',x,z,'b--')
```

**Text and Legends**

```
>> legend('sine','cosine')
>> xlabel('x')
>> ylabel('f(x)')
>> title('Trigonometric Graphs')
```

**Axis Management**

```
>> axis off
>> axis on
>> axis equal
>> grid on
```

**Figure Window Partitioning**

```
>> figure
>> subplot(1,3,1)
>> plot(x,y)
>> subplot(1,3,2)
>> plot(x,y,'r--')
>> subplot(1,3,3)
>> plot(x,y,'go')
```

## 3 Programming with MATLAB

It is possible to use files to group MATLAB instructions. To write them, you use a text editor: either the one integrated into MATLAB, or an external editor; MATLAB commands allow these files to be used.

**Script Execution** Create the file `script.m` containing the following instructions:

```
n = 5;
A = zeros(n, n);
for i = 1 : n
    A(i,i) = i;
end
A
```

Execution in MATLAB is done as follows.

```
>> script
```

**Function Execution** Create the file `f.m`:

```
function y = f(x)
y = x.^3;
```

We use `f` directly in MATLAB:

```
f([1,2,3])
```

In the case of functions as well as in the case of scripts, the files must be contained in the current directory.

## 4 Usage of Anonymous Functions

Anonymous functions (or function handles) are a very useful tool in MATLAB. They provides an easy way to define functions inside scripts. The syntax is

```
function_name = @(variable_name) matlab_expression;
```

Be particularly careful to avoid naming your functions with names that conflict with native MATLAB commands (like `sin`). In general, prefer long and quite explicit names.

**Exercise 4.1** *Define an anonymous function* ***`afun1`*** *to evaluate $f(x) = \sin(x)/x$ by writing*

```
>>  afun1 = @(x) sin(x)/x
>>  afun1(2)
```

*It is also possible to put several arguments in these functions as well as returning vectors.*

```
>>  afun2 = @(x, y) [2*x*sin(y); x^2*cos(y)]
>>  afun2(1, 2)
```

*It is not mandatory put every variable used as a parameter in an anonymous function.*

```
>>  a = 1;
>>  b = -2;
>>  c = 1;
>>  afun3 = @(x) a*x^2 + b*x + c;
>>  afun3(2)
>>  a = 10;
>>  afun3(2)
```

*Notice that, in this case, the values of variables* `a`*,* `b`*, and* `c` *used in the function are those defined just before it. Changing them afterward does not modify the anonymous function.*

An even more important property of anonymous functions is the ability to pass them as arguments to other functions. First, write a MATLAB program `doplot.m` to make the graph as shown below.

```
function doplot(afun, xrange)
% doplot creates the plot for y = afun(x) with x in the
%  interval xrange = [la, lb].
```

Then, to plot the function `afun3` on the interval $[-2, 2]$, simply write

```
>>  doplot(afun3, [-2,2])
```

Finally, try to apply `doplot` for an already implemented MATLAB function, for example `sin`.

## 5 Using the Online Help

The online help is available using the instruction `help command_name` or `doc command_name`.

**Exercise 5.1** *We are given a matrix* `A = rand(5,5)`. *By using the command* `diag`, *construct (in a single instruction line) the diagonal matrix $D$ with coefficients defined by $D_{ii} = A_{ii}$. If the exact command name is not known, the instruction can be used*

```
>> doc keyword
```

*which performs a search in the headers of the help topics, for example,*

```
>> doc diagonal
```

**Exercise 5.2** *Indicate which commands allow you to:*

- *calculate the eigenvalues of a matrix;*
- *change the number display format;*
- *plot a graph in polar coordinates;*
- *create a Vandermonde matrix;*
- *calculate the length of a vector;*
- *generate a random matrix;*
- *calculate the execution time of a task.*

**Exercise 5.3** *Construct the following tridiagonal matrix using, in particular, the* `diag` *function.*

$$\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 \end{pmatrix}.$$