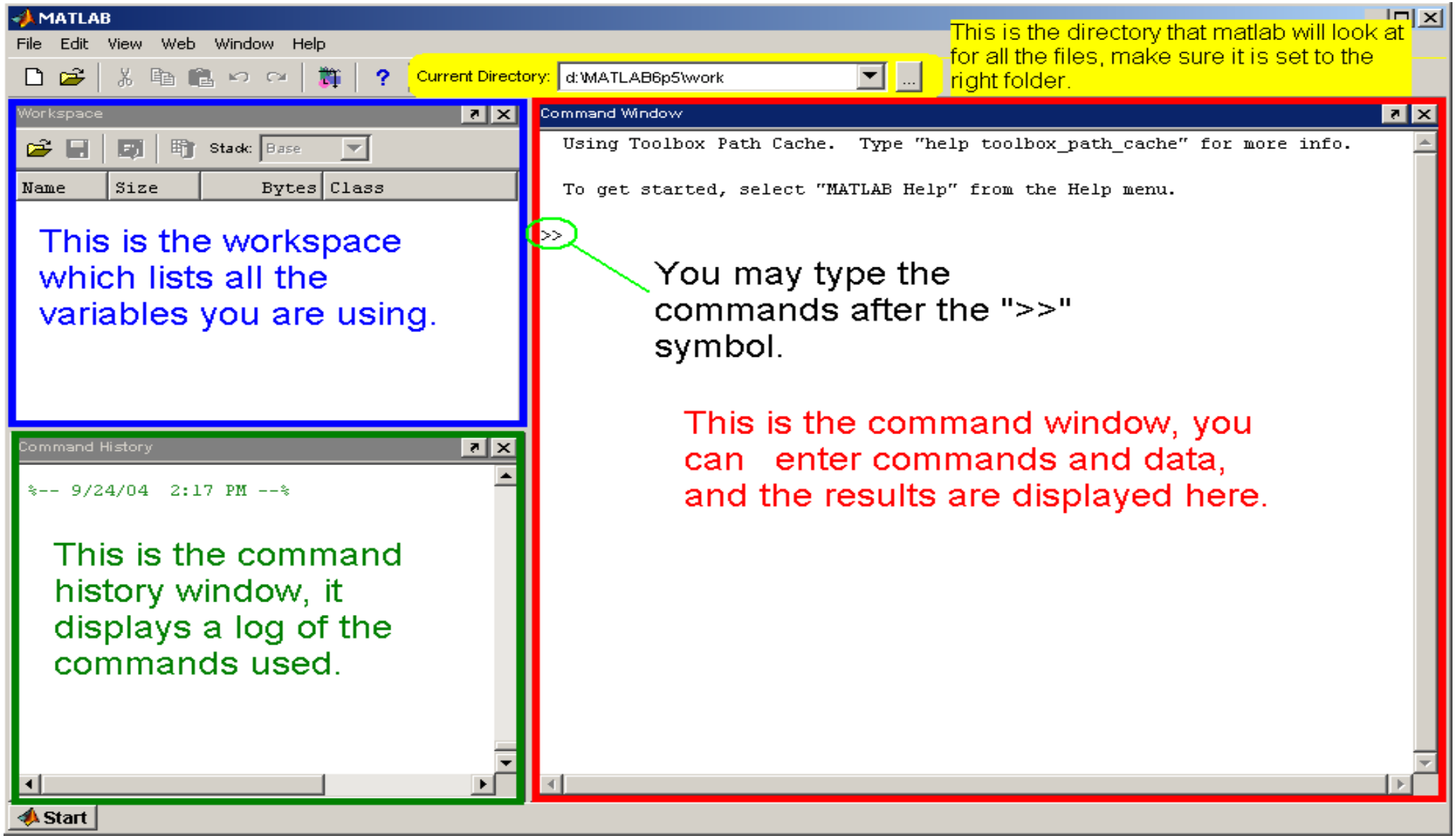
Matlab

MATLAB stands for **"MATrix LABoratory"**.
It was originally developed in the **late 1970s** by **Cleve Moler** to provide easy access to matrix software for engineers and scientists. MATLAB is widely used for **numerical computing, data analysis, algorithm development, and visualization**.
Since MATLAB is optimized for **matrix and vector operations**, it is especially popular in fields like **engineering, physics, finance, and machine learning**.

Matlab


MATLAB
File Edit View Web Window Help
Current Directory: d:\MATLAB6p5\work
This is the directory that matlab will look at for all the files, make sure it is set to the right folder.
Workspace
Stack: Base
Name Size Bytes Class
This is the workspace which lists all the variables you are using.
Command History
%-- 9/24/04 2:17 PM --%
This is the command history window, it displays a log of the commands used.
Command Window
Using Toolbox Path Cache. Type "help toolbox_path_cache" for more info.
To get started, select "MATLAB Help" from the Help menu.
>>
You may type the commands after the ">>" symbol.
This is the command window, you can enter commands and data, and the results are displayed here.
Start

Variables in MATLAB

In **MATLAB**, a variable is used to store data, such as numbers, arrays, or characters, for processing and calculations. Here's a quick guide to understanding and using variables in MATLAB:

1. Declaring Variables

In MATLAB, you don't need to declare a variable type explicitly. You just assign a value to a name.

```
x = 5;     % Assigning the value 5 to variable x
y = 10;    % Assigning the value 10 to variable y
z = x +y; % Adding x and y and storing in z
```

Variables

- No need for types. i.e.,

```
int a;
double b;
float c;
```

- All variables are created with double precision unless specified and they are matrices.

```
Example:
>>x=5;
>>x1=2;
```

- After these statements, the variables are 1x1 matrices with double precision

Variables in MATLAB

2. Naming Rules for Variables

- Must start with a **letter** (A-Z, a-z).
- Can contain **letters, numbers, and underscores** (_).
- Case-sensitive (`Var1` is different from `var1`).
- Cannot use **MATLAB reserved keywords** (like `if`, `for`, `while`).

✅ **Valid names:**

```
a1, speed, temperature_sensor
```

❌ **Invalid names:**

```
1var, if, x-y
```

Variables in MATLAB

4. Different Data Types in MATLAB Variables

MATLAB supports different types of variables:

```
A = 10;           % Integer (double by default)
B = 3.14;         % Floating point
C = 'Hello';      % Character array (string)
D = [1 2 3 4];    % Row vector
E = [1; 2; 3; 4]; % Column vector
F = [1 2; 3 4];   % Matrix (2x2)
```

Variables in MATLAB

5. Special Variables in MATLAB

MATLAB provides built-in variables that you can use:

- `pi` → 3.1416
- `inf` → Infinity
- `NaN` → Not a Number
- `eps` → Smallest difference between two numbers

Example:

```
radius = 5;
area = pi * radius^2;  % Calculate area of a circle
```

Variables in MATLAB

6. Variable Conversion

You can convert variables to different types:

```
A = 5.5;
B = int32(A);    % Convert to integer
C = num2str(A); % Convert number to string
```

Variables in MATLAB

7. Saving and Loading Variables

- **Save workspace variables to a file:**

```
save('myData.mat')
```

- **Load saved variables:**

```
load('myData.mat')
```

Variables in MATLAB

8. Global Variables

Use global to share a variable between different functions.

```
global x
x = 50;
```

Summary

- **Variables store values for computations.**
- **MATLAB assigns types automatically.**
- **You can check, clear, and convert variables.**
- **global allows variable sharing between functions**.

Vectors, Matrices, and Arrays in MATLAB

MATLAB is built for handling **vectors, matrices, and arrays** efficiently. Let's break them down:

1. Vectors in MATLAB

A **vector** is a one-dimensional array. It can be either **a row vector** or **a column vector**.

Creating a Row Vector (1×n matrix)

A row vector is a horizontal array of elements.

```
A = [1 2 3 4 5];    % Row vector
B = [1, 2, 3, 4, 5]; % Also a row vector (commas or spaces work)
```

Creating a Column Vector (n×1 matrix)

A column vector is a vertical array of elements.

```
C = [1; 2; 3; 4; 5];  % Column vector (use semicolon `;`)
```

Array, Matrix

- a vector

```
x = [1 2 5 1]

x =
     1    2    5    1
```

- a matrix

```
y = [1 2 3; 5 1 4; 3 2 -1]

y =
     1     2     3
     5     1     4
     3     2    -1
```

- transpose

```
y = x'

y =
     1
     2
     5
     1
```

Long Array, Matrix

- `t =1:10`

```
t =
    1    2    3    4   5   6    7   8    9    10
```

- `k =2:-0.5:-1`

```
k =
    2   1.5   1   0.5   0   -0.5   -1
```

- `B = [1:4; 5:8]`

```
x =
    1      2      3     4
    5      6      7     8
```

Generating Vectors from functions

- zeros(M,N) MxN matrix of zeros

```
x = zeros(1,3)
x =
   0     0     0
```

- ones(M,N) MxN matrix of ones

```
x = ones(1,3)
x =
   1     1     1
```

- rand(M,N) MxN matrix of uniformly distributed random numbers on (0,1)

```
x = rand(1,3)
x =
 0.9501  0.2311 0.6068
```

Matrix Index

- The matrix indices begin from 1 (not 0 (as in C))
- The matrix indices must be positive integer

Given:

```
A =

     3     5     3
     6     8     2
     2     7     3
```

```
>> A(6)

ans =

     7
```

```
>> A(3,2)

ans =

     7
```

```
>> A(2,:)

ans =

     6     8     2
```

```
>> A(1:2,2)

ans =

     5
     8
```

A(-2), A(0)

Error: ??? Subscript indices must either be real positive integers or logicals.

A(4,2)
Error: ??? Index exceeds matrix dimensions.

Concatenation of Matrices

- `x = [1 2], y = [4 5], z=[ 0 0]`

```
A = [ x y]

     1   2   4   5

B = [x ; y]

     1 2
     4 5
```

```
C = [x y ;z]
Error:
??? Error using ==> vertcat CAT arguments dimensions are not consistent.
```

Operators (arithmetic)

+ addition

- subtraction

* multiplication

/ division

^ power

‘ complex conjugate transpose

Matrices Operations

Given A and B:

```
>> A = [1 2 3;4 5 6;7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9
```

```
>> B = [3 5 2; 5 2 8; 3 6 9]

B =

     3     5     2
     5     2     8
     3     6     9
```

Addition

```
>> X = A + B

X =

     4     7     5
     9     7    14
    10    14    18
```

Subtraction

```
>> Y = A - B

Y =

    -2    -3     1
    -1     3    -2
     4     2     0
```

Product

```
>> Z = A * B

Z =

    22    27    45
    55    66   102
    88   105   159
```

Transpose

```
>> T = A'

T =

     1     4     7
     2     5     8
     3     6     9
```

Operators (Element by Element)

.* element-by-element multiplication

./ element-by-element division

.^ element-by-element power

The use of "." – "Element" Operation

```
A = [1 2 3; 5 1 4; 3 2 1]
    A =
          1    2    3
          5    1    4
          3    2   -1
```

```
x = A(1,:)

x=
     1   2   3
```

```
y = A(3 ,:)

y=
     3  4  -1
```

```
b = x .* y

b=
     3  8 -3
```

```
c = x . / y

c=
   0.33   0.5   -3
```

```
d = x .^2

d=
      1    4    9
```

```
K= x^2
Erorr:
 ??? Error using ==> mpower  Matrix must be square.
B=x*y
Erorr:
??? Error using ==> mtimes Inner matrix dimensions must agree.
```

Special Vectors

- **Linspace**: Creates a vector with evenly spaced elements.

```
x = linspace(1, 10, 5); % Generates [1 3.25 5.5 7.75 10]
```

- **Colon Operator (:)**: Defines a range with a step size.

```
y = 1:2:10;  % Generates [1 3 5 7 9] (start:step:end)
```

Vectors, Matrices, and Arrays in MATLAB

- **Zero vector and One vector**:

```
Z = zeros(1,5); % Row vector of 5 zeros

O = ones(5,1);  % Column vector of 5 ones.
```

Vectors, Matrices, and Arrays in MATLAB

2. Matrices in MATLAB

A **matrix** is a two-dimensional array of numbers.

Creating a Matrix

```
M = [1 2 3; 4 5 6; 7 8 9];  % 3×3 matrix
```

- Numbers in **each row** are separated by spaces or commas.
- Rows are separated by **semicolons (;)**.

Vectors, Matrices, and Arrays in MATLAB

Accessing Elements in a Matrix

```
M(2,3)   % Access the element in row 2, column 3 (Output: 6)
M(:,2)   % Get all rows in column 2 (Output: [2; 5; 8])
M(1,:)   % Get all columns in row 1 (Output: [1 2 3])
```

Special Matrices

```
I = eye(3);       % 3×3 Identity matrix
Z = zeros(3,4);   % 3×4 Zero matrix
O = ones(2,3);    % 2×3 Matrix of ones
R = rand(2,2);    % 2×2 Matrix with random values (0 to 1)
```

Vectors, Matrices, and Arrays in MATLAB

3. Arrays in MATLAB

An **array** is a generalized collection of elements (1D, 2D, or multi-dimensional).

Creating an N-Dimensional Array

```
A = rand(3,3,2);   % 3×3×2 3D array
B = zeros(2,3,4);  % 2×3×4 3D array of zeros
```

Accessing Elements in an Array

```
A(2,3,1)  % Get the element at row 2, column 3, layer 1
```

Vectors, Matrices, and Arrays in MATLAB

4. Vector vs. Matrix vs. Array

Type	Dimensions	Example
Vector	1D	[1 2 3] or [1; 2; 3]
Matrix	2D	[1 2; 3 4]
Array	3D or more	rand(2,2,3)

Conclusion

- **Vectors** are **1D arrays** (row or column).
- **Matrices** are **2D arrays** (rows × columns).
- **Arrays** are **N-dimensional structures** that extend beyond matrices.

Questions

- ?
- ?
- ?
- ?
- ?

Thanks