

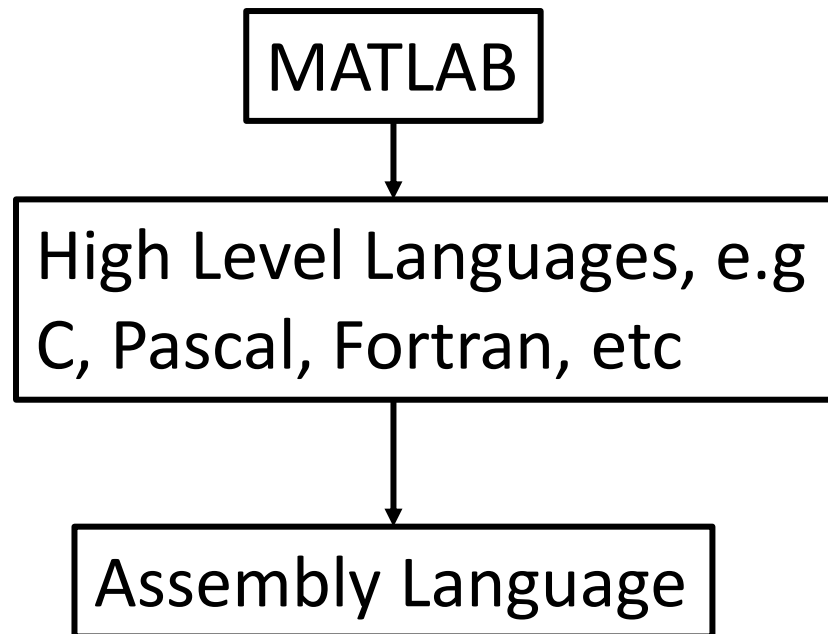
INTRODUCTION TO MATLAB

ECE 321 – PRINCIPLES OF COMMUNICATION

Tuesday, 09 December 2025

WHAT IS MATLAB?

MATLAB is a high level language which has many specialized toolboxes for making program much easier.



(a) In Fortran we multiply two matrices by the code:

```
real*8 A(10,10), B(10,10), C(10,10)
```

```
..
```

```
...
```

```
do i=1,10
```

```
do j=1,10
```

```
C(i,j) = A(i,j) + B(i,j)
```

```
continue
```

```
continue
```

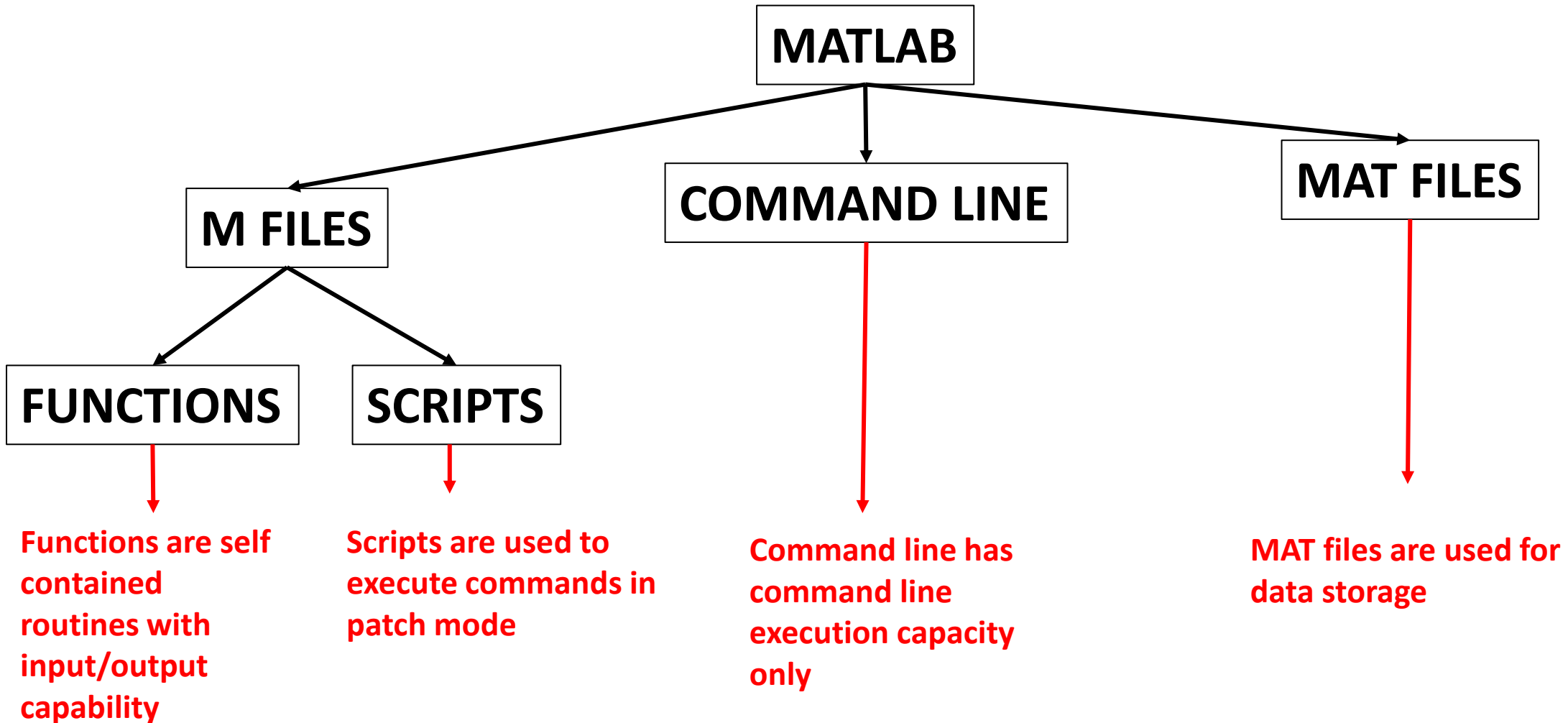
(b) In Matlab we achieve this without declaration and using in one line code:

```
C = A*B
```

BRIEF HISTORY OF MATLAB

- **Matlab** was founded in 1984 by Jack Little and Cleve Moler who recognized the need among engineers and scientists for more powerful and productive computation environments beyond that provided existing high level languages, i.e Fortran and C.
- It is distributed by a company **called Mathworks whose Flagship products are MATLAB and SIMULINK**

USING MATLAB



BASIC OPERATORS IN MATLAB

- Basic arithmetic operator in matlab are: $= + - * / ^ ()$

- Examples:

```
>> 2+3/4*5
```

```
>> 3^2*4
```

```
>> 3-4/4-2
```

```
>> (1+i)*(-1+3*i)
```

```
>> (1+i)/(-1+3*i)
```

NUMBER FORMATS IN MATLAB

Matlab supports short and long number formats.

Examples:

```
>> Format short
```

```
>> pi = 3.1416
```

```
>> format long
```

```
>> pi = 3.1415926535897....
```

VARIABLES IN MATLAB

1. **Variable names** can contain up to 63 characters
2. **Variable names** must start with a letter followed by letters, digits, and underscores.
3. **Variable names** are case sensitive.
4. **Allowed:** a , x1, z2453, A, com_c.
5. **Not allowed:** com-c, 2p, %x, @sign
6. Avoid using special names, e.g. pi

SPECIAL VARIABLES

1. **ans** default variable name for results
2. **Pi** Value of 3.1459...
3. **eps** Smallest incremental number
4. **inf** Infinity
5. **NaN** Not a number e.g. 0/0
6. **realmin** The smallest usable positive real number
7. **realmax** The largest usable positive real number

NUMBER TYPES

- There is no need to declare types. The following is not allowed.

~~Int a;~~

~~Float b;~~

~~Double c~~

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

~~x=1~~

~~y = 3~~

~~x1 = 5~~

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

BUILD-IN FUNCTIONS

1. Trigonometric functions:

$\sin, \cos, \tan, \sec = 1/\cos, \csc = 1/\sin, \cot = 1/\tan$

2. inverse trigonometric functions

$\arcsin, \arccos, \arctan$; answer returned in radians,

3. Exponential

$y = \exp(x)$

4. Logarithm:

$\log(X)$ to base e, $\log_{10}(X)$: log to base 10

5. Square root:

$\sqrt{x}$

INBUILT FUNCTIONS

1. **mean(A)**: mean value of a vector
2. **max(A), min (A)**: maximum and minimum
3. **sum(A)**: summation **sort(A)**: sorted vector
4. **median(A)**: median value
5. **std(A)**: standard deviation
6. **det(A)** : determinant of a square matrix
7. **dot(a,b)**: dot product of two vectors
8. **Cross(a,b)**: cross product of two vectors
9. **inv(A)**: Inverse of a matrix A
10. **abs(z)** : magnitude of a number

INBUILT FUNCTIONS

fft,ifft,fft2,ifft2: Fast Fourier

dct,idct,dct2,idct2: Discrete Cosine

czf: Chirp-z

radon,iradon: Radon

hilbert: Hilbert

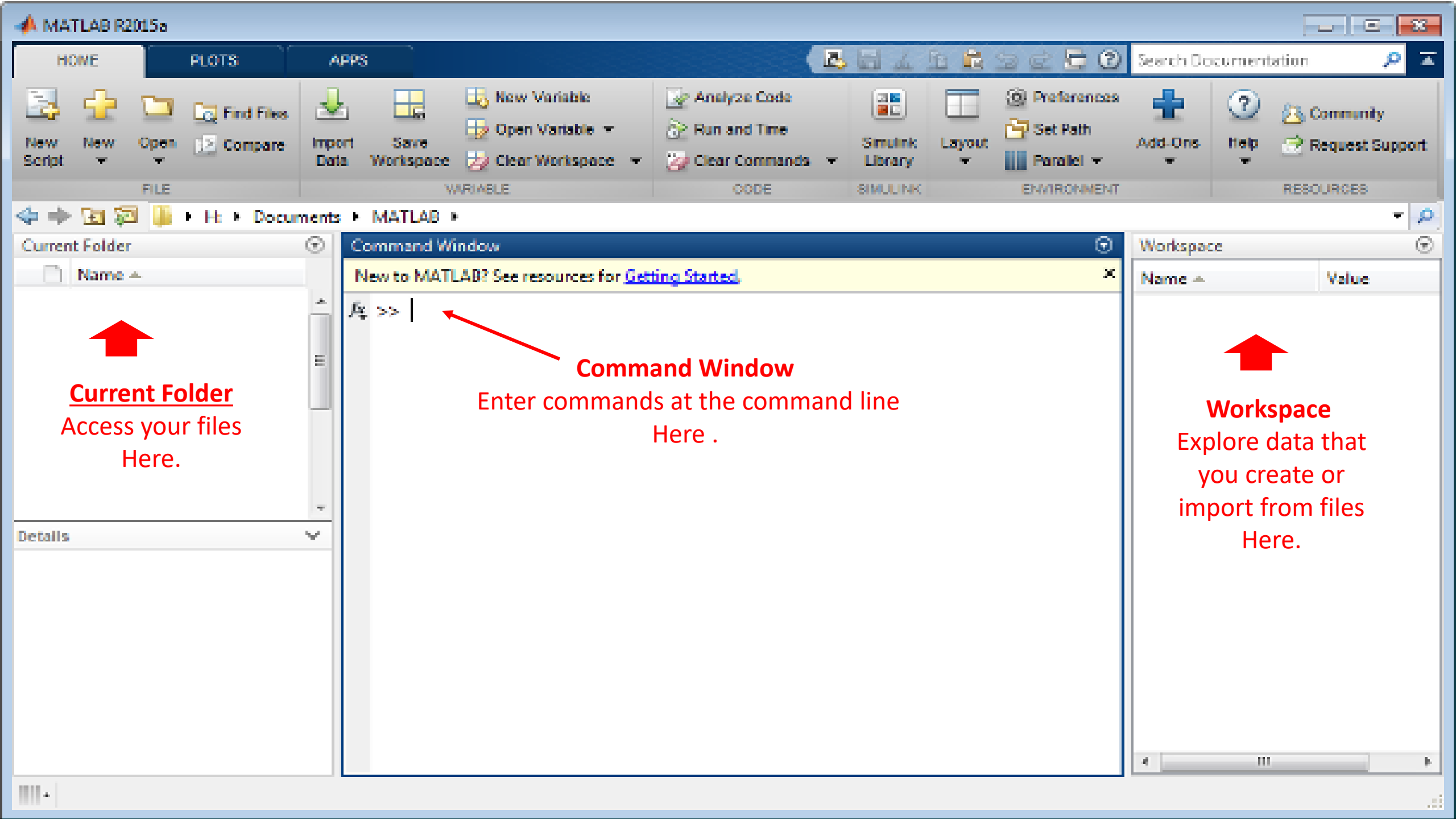
dftmtx: Discrete Fourier matrix

fftshift: Swap vector halves

There are more functions in the signal processing toolbox.

READING & PLAYING AUDIO FILES

1. MATLAB can read, play, write, and even record audio files.
2. You can use the following Matlab commands to read and play audio files:
 - a) **wavread** or **auread** to load a file or create a signal yourself.
 - b) **sound**, **soundsc**, or **wavplay** to play audio.
 - c) **wavwrite** or **auwrite** to save to audio file.
 - d) **wavrecord** to record sound from Windows audio input device.



Current Folder
Access your files
Here.

Command Window
Enter commands at the command line
Here .

Workspace
Explore data that
you create or
import from files
Here.

CREATING VARIABLES AND FUNCTION CALLS

Enter the following at the command line:

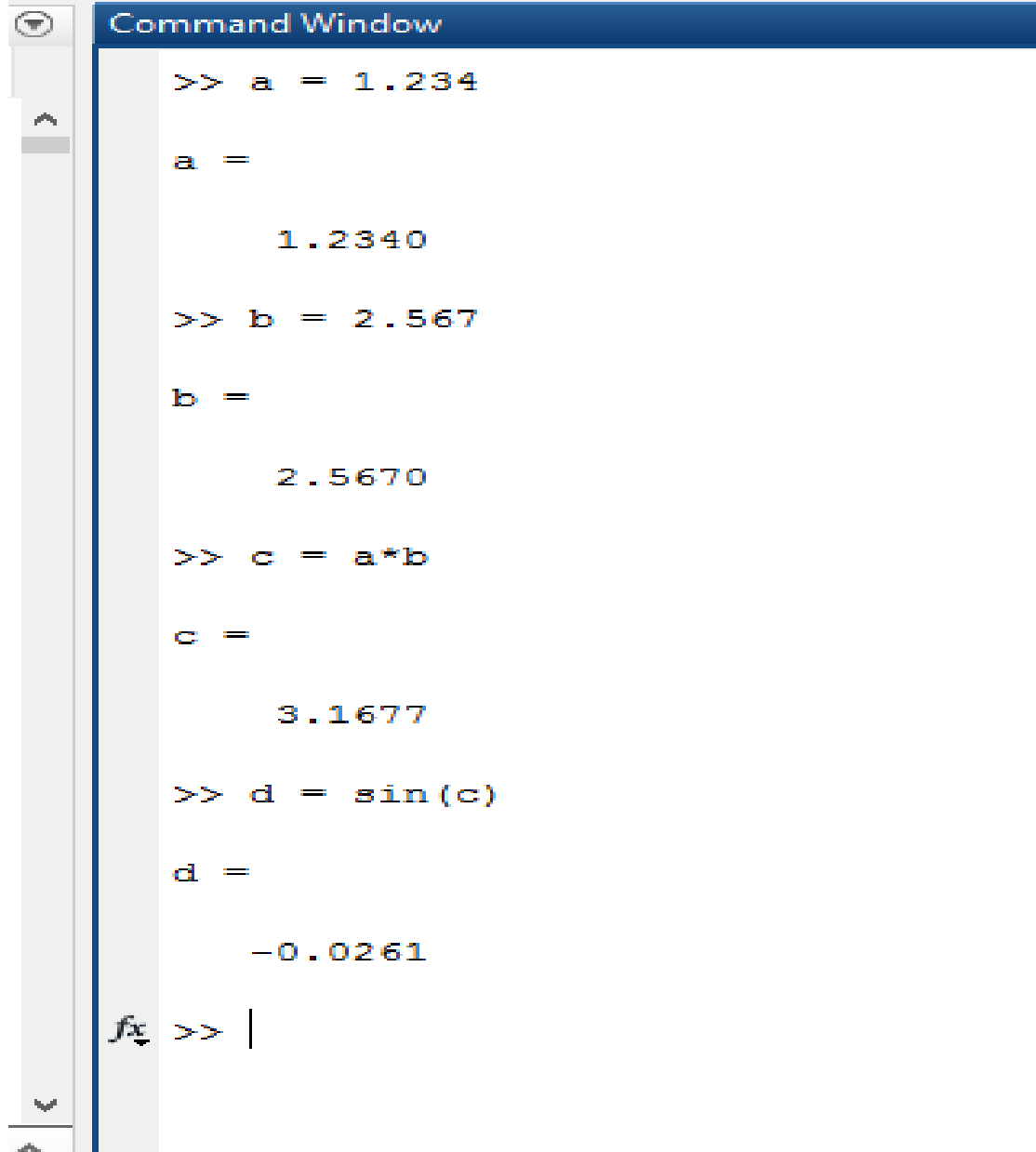
a = 1.234

b = 2.567

c = a * b

d = sin(d)

sin(a)

A screenshot of the MATLAB Command Window. The window has a blue title bar that says "Command Window". On the left side, there is a vertical toolbar with icons for undo, redo, and search. The main area of the window shows a series of commands and their outputs. The commands are: >> a = 1.234, >> b = 2.567, >> c = a*b, and >> d = sin(c). The outputs are: a = 1.2340, b = 2.5670, c = 3.1677, and d = -0.0261. At the bottom, there is a prompt >> followed by a vertical bar, indicating that the user can enter more commands. The window also has a scrollbar on the left side.

```
>> a = 1.234  
  
a =  
  
    1.2340  
  
>> b = 2.567  
  
b =  
  
    2.5670  
  
>> c = a*b  
  
c =  
  
    3.1677  
  
>> d = sin(c)  
  
d =  
  
   -0.0261  
  
fx >> |
```

ENTERING MATRICES

1. SINGLE-ROW MATRIX

Enter single Row matrix as follows:

C = [1 2 3]

Command Window

```
>> C = [1 2 3]
```

C =

1	2	3
---	---	---

 >> |

2. MULTIPLE-ROW MATRIX

Enter Multiple Row matrix as follows:

C2 = [1 2 3; 4 5 6; 7 8 10]

Command Window

```
>> C2=[1 2 3;4 5 6;7 8 10]
```

C2 =

1	2	3
4	5	6
7	8	10

 >> |

MULTIPLYING A MATRIX WITH A CONSTANT

Enter

$C3 = 10 * C2$

Command Window

```
>> C2=[1 2 3;4 5 6;7 8 10]
```

```
C2 =
```

1	2	3
4	5	6
7	8	10

```
>> C3 = 10*C2
```

```
C3 =
```

10	20	30
40	50	60
70	80	100

fx >> |

MULTIPLYING TWO MATRICES

Enter:

$C4 = C2 * C3$

Command Window

```
>> C2=[1 2 3;4 5 6;7 8 10]
```

```
C2 =
```

1	2	3
4	5	6
7	8	10

```
>> C3 = 10*C2
```

```
C3 =
```

10	20	30
40	50	60
70	80	100

```
>> C4 = C2*C3
```

```
C4 =
```

300	360	450
660	810	1020
1090	1340	1690

TRANSPOSING A MATRIX

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

Enter

$C5 = C4'$

Command Window

```
C3 =
```

```
    10    20    30
    40    50    60
    70    80   100
```

```
>> C4 = C2*C3
```

```
C4 =
```

```
    300    360    450
    660    810   1020
   1090   1340   1690
```

```
>> C5 = c4'
```

```
Undefined function or variable 'c4'.
```

```
Did you mean:
```

```
>> C5 = C4'
```

```
C5 =
```

```
    300    660   1090
    360    810   1340
    450   1020   1690
```

GETTING THE INVERSE OF A MATRIX

Type:

$C6 = C5 * \text{inv}(C5)$

Command Window

```
300      300      100
660      810     1020
1090     1340     1690
```

```
>> C5 = c4'
```

```
Undefined function or variable 'c4'.
```

```
Did you mean:
```

```
>> C5 = C4'
```

```
C5 =
```

```
300      660     1090
360      810     1340
450     1020     1690
```

```
>> C6 = C5*INV(C5)
```

```
Undefined function 'INV' for input arguments of type 'double'.
```

```
Did you mean:
```

```
>> C6 = C5*inv(C5)
```

```
C6 =
```

```
1.0000    0.0000   -0.0000
-0.0000    1.0000   -0.0000
-0.0000    0.0000    1.0000
```

CHANGING NUMBER FORMAT USING LONG TO SHORT COMMANDS

Type:

>> Format short

>> C7=C5*inv(C5)

>> Format long

>>> C7=C5*inv(C5)

```
>> format short
```

```
>> C7 = C5*inv(C5)
```

```
C7 =
```

1.0000	0.0000	-0.0000
-0.0000	1.0000	-0.0000
-0.0000	0.0000	1.0000

```
>> format long
```

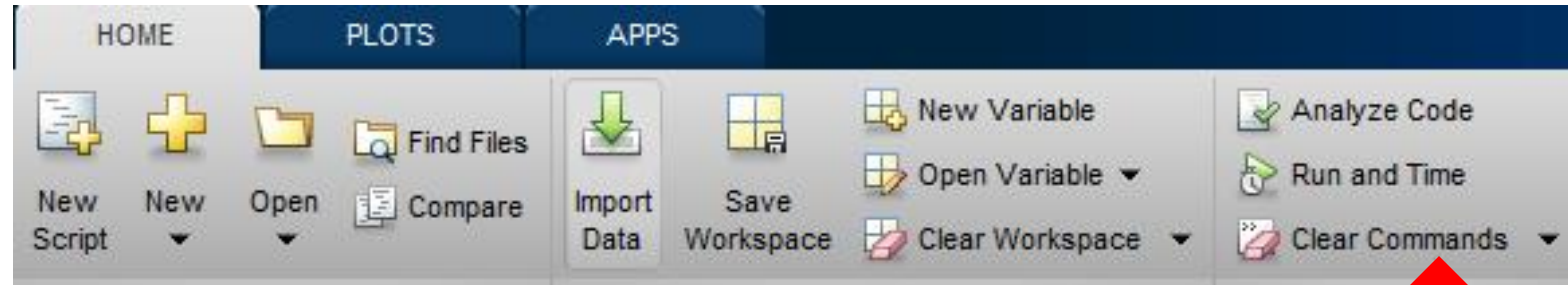
```
>> C8=C5*inv(C5)
```

```
C8 =
```

0.9999999999999965	0.0000000000000203	-0.0000000000000184
-0.0000000000000001	0.9999999999999936	-0.0000000000000115
-0.00000000000000044	0.0000000000000018	0.9999999999999912

CARRYING OUT BITWISE MATRIX MULTIPLICATION

1. Clear Command window



2. Enter

$A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 10]$

$C9 = A.*A$

```
Command Window

>> A=[1 2 3; 4 5 6; 7 8 10]

A =

     1     2     3
     4     5     6
     7     8    10

>> C9 = A.*A

C9 =

     1     4     9
    16    25    36
    49    64   100

fx >> |
```

DIVIDING A MATRIX WITH A CONSTANT

Enter

C10 = C9/4

```
>> A=[1 2 3; 4 5 6; 7 8 10]
```

```
A =
```

```
     1     2     3
     4     5     6
     7     8    10
```

```
>> C9 = A.*A
```

```
C9 =
```

```
     1     4     9
    16    25    36
    49    64   100
```

```
>> C10 = C9/4
```

```
C10 =
```

```
    0.2500000000000000    1.0000000000000000    2.2500000000000000
    4.0000000000000000    6.2500000000000000    9.0000000000000000
   12.2500000000000000   16.0000000000000000   25.0000000000000000
```

RAISING A MATRIX TO A POWER OF X

Enter

$C1 = A^3$

```
>> C1 = A^3
```

```
C1 =
```

489	600	756
1104	1353	1704
1828	2240	2821

CONCATATION OF MATRICES

Enter

$C = [A, A]$

```
>> C2 = [A, A]
```

```
C2 =
```

1	2	3
4	5	6
7	8	10

1	2	3
4	5	6
7	8	10

Enter

$D = [A; A]$

```
>> C3 = [A; A]
```

```
C3 =
```

1	2	3
4	5	6
7	8	10

1	2	3
4	5	6
7	8	10

COMPLEX MATRICES

1. Clear Command Window

2. Enter

>> Sqrt(-1)

Command Window

```
>> sqrt(-1)
```

```
ans =
```

```
0.0000000000000000 + 1.0000000000000000i
```

3. Enter

C = [3+4i, 4+3i; -i, 10i]

```
>> C = [3+4i, 4+3i; -i, 10i]
```

```
C =
```

```
3.0000000000000000 + 4.0000000000000000i  4.0000000000000000 + 3.0000000000000000i  
0.0000000000000000 - 1.0000000000000000i  0.0000000000000000 + 10.0000000000000000i
```

COLON OPERATOR

The colon (:) is use in defining a vector range as illustrated below.

```
>> x = 7:12
```

```
x =
```

```
7    8    9   10   11   12
```

```
>> x = 7:2.5:12
```

```
x =
```

```
7.0000  9.5000 12.0000
```

```
>> x = 12:-1:7
```

```
x =
```

```
12   11   10    9    8    7
```