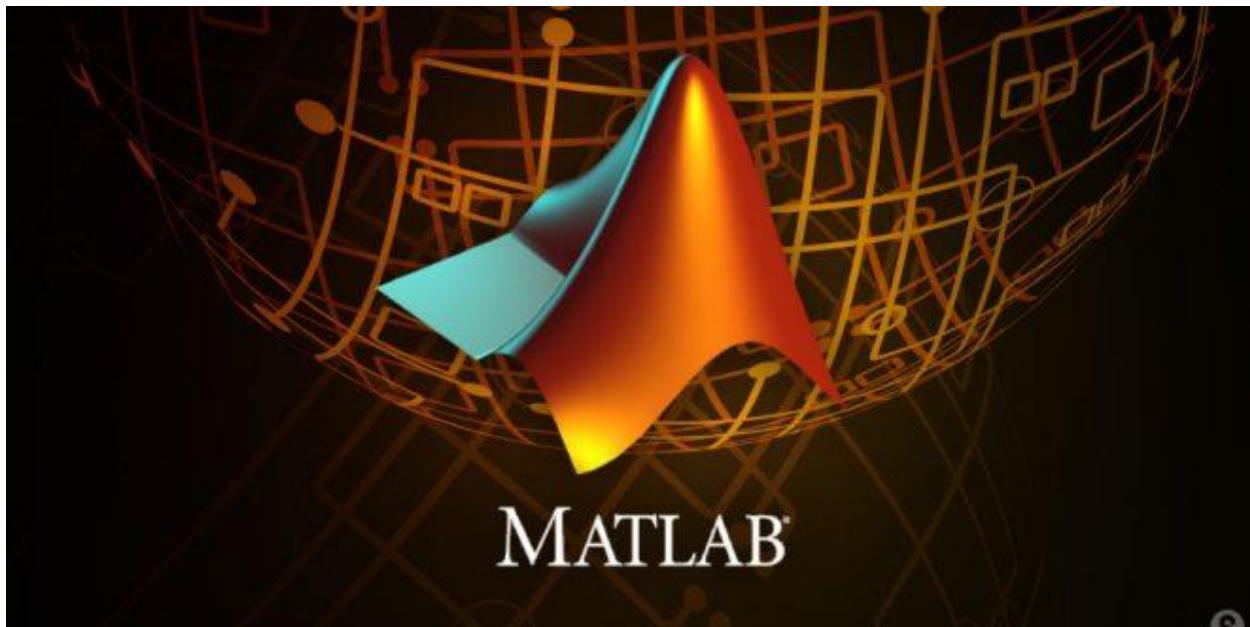




Course N°04

Matrix in

MATLAB



Dr. Salah Djerouni



1. Definition a matrix

A matrix is surrounded by **brackets** and may have an **arbitrary number of rows and columns**; for example, the matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad (1)$$

To create a such **matrix** in **MATLAB**, the following basic conventions must be followed:

- ✓ Separate the elements of a row with **spaces or commas** `,`
- ✓ Use a **semicolon** `;` to indicate the end of each **line or row**
- ✓ Surround the entire list of items with **square brackets** `[]`.

The image shows the MATLAB R2014a interface. The Command Window displays two methods for creating matrix A:

```

>> % first method
>> A = [1 2 3 ; 4 5 6 ; 7 8 9]
A =
    1.00    2.00    3.00
    4.00    5.00    6.00
    7.00    8.00    9.00

>> % second method
>> A = [1,2,3;4,5,6;7,8,9]
A =
    1.00    2.00    3.00
    4.00    5.00    6.00
    7.00    8.00    9.00

```

The Workspace browser on the right shows matrix A with the value `[1,2,3,4,5,6,7,8,9]`.

Fig 1. Different methods to create or define a matrix



2. Other useful MATLAB functions

For matrices, to find or extract the highest and/or lowest value or number in the whole matrix, we use the command/function `max(.)` and/or `min(.)` two times because these are matrix with two dimensions not one dimension. Otherwise, will indicate the highest values or the lowest values for each column in the matrix.

Fig 2. Find the highest and lowest value in the whole matrix

Again, in order to find or evaluate the summation and/or production of such element in the matrix, we use the command/function `sum(.)` and `prod(.)`



```

>> s = magic(4)
s =
16.00 2.00 3.00 13.00
5.00 11.00 10.00 8.00
9.00 7.00 6.00 12.00
4.00 14.00 15.00 1.00
>> sum(sum(s))
ans =
136.00
>> prod(prod(s))
ans =
2092278988000.00
fx >> |

```

Fig 3. Find the **summation** as well as the **product** of the all of element in the whole matrix

The **mean** of a **matrix**, also known as the **average** which equal the **sum** of the **numbers** in **each row** in the **matrix divided** by the **number of rows** and then the result will **sum again** and **divided** on the **number of columns**, using the command/function **mean(.)**

```

>> s = magic(4)
s =
16.00 2.00 3.00 13.00
5.00 11.00 10.00 8.00
9.00 7.00 6.00 12.00
4.00 14.00 15.00 1.00
>> mean(mean(s))
ans =
8.50
fx >> |

```

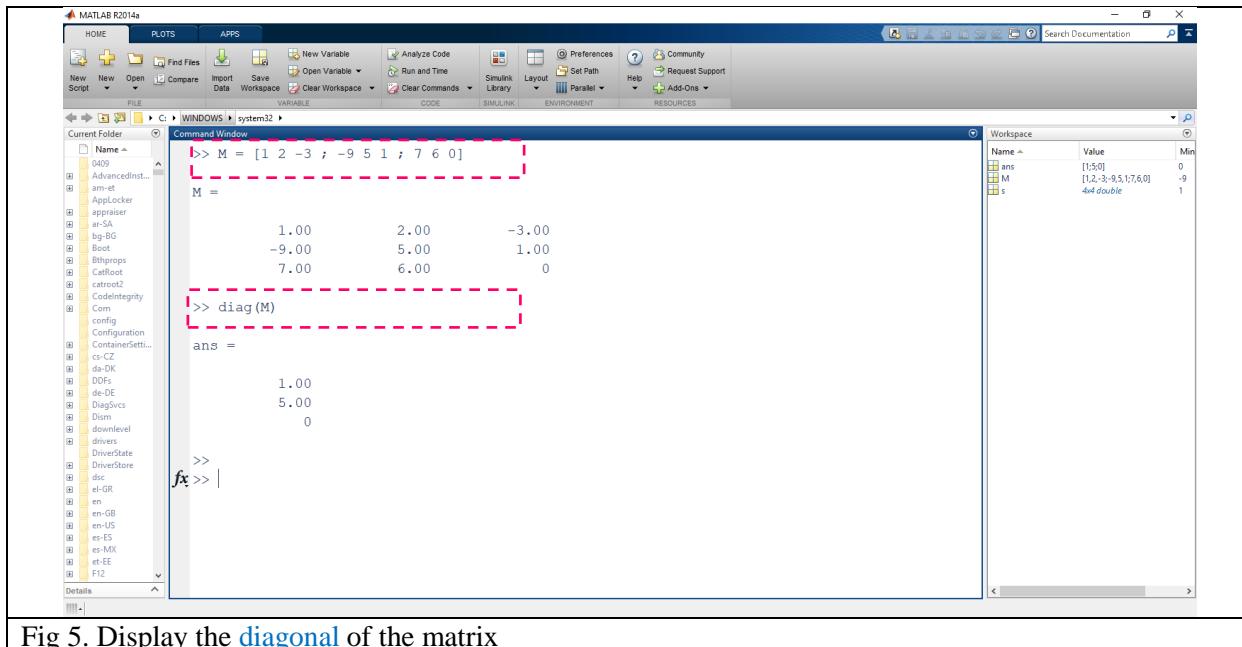
Fig 4. Find the **average** of the values that contain in the matrix



3. MATLAB output

3.1. The *diag* command

Displaying or to generate a **diagonal** of a **matrix**, using the command/function **diag(.)**



The screenshot shows the MATLAB R2014a interface. The Command Window displays the following code and output:

```

>> M = [1 2 -3 ; -9 5 1 ; 7 6 0]
M =
    1.00    2.00    -3.00
   -9.00    5.00    1.00
    7.00    6.00    0.00

>> diag(M)
ans =
    1.00
    5.00
    0.00

```

The Workspace browser on the right shows the variables defined:

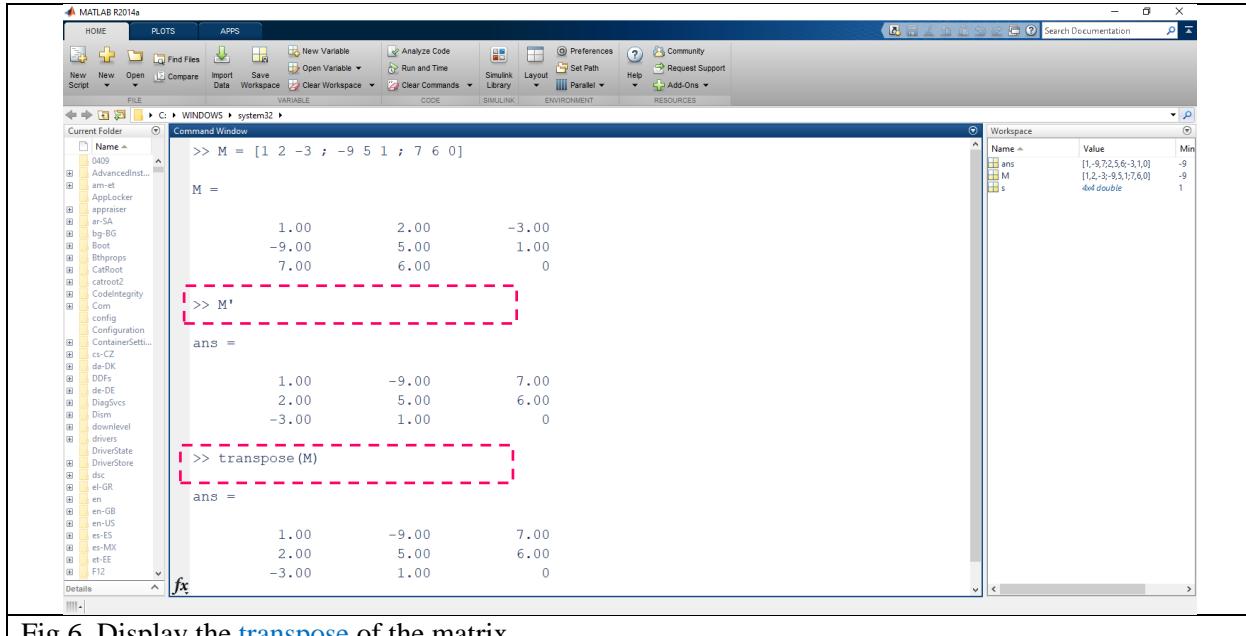
Name	Value	Min
ans	[1;5;0]	0
M	[1,2,-3;-9,5,1;7,6,0]	-9
s	4x3 double	1

Fig 5. Display the **diagonal** of the matrix



3.2.The transpose and/or ' and/or transp command

Transpose a matrix, rows become a columns and columns become rows.



The screenshot shows the MATLAB R2014a interface. In the Command Window, a matrix M is defined as:

```
>> M = [1 2 -3 ; -9 5 1 ; 7 6 0]
```

The matrix M is displayed as:

$$M = \begin{bmatrix} 1.00 & 2.00 & -3.00 \\ -9.00 & 5.00 & 1.00 \\ 7.00 & 6.00 & 0 \end{bmatrix}$$

The transpose of M is then calculated and displayed:

```
>> M'
```

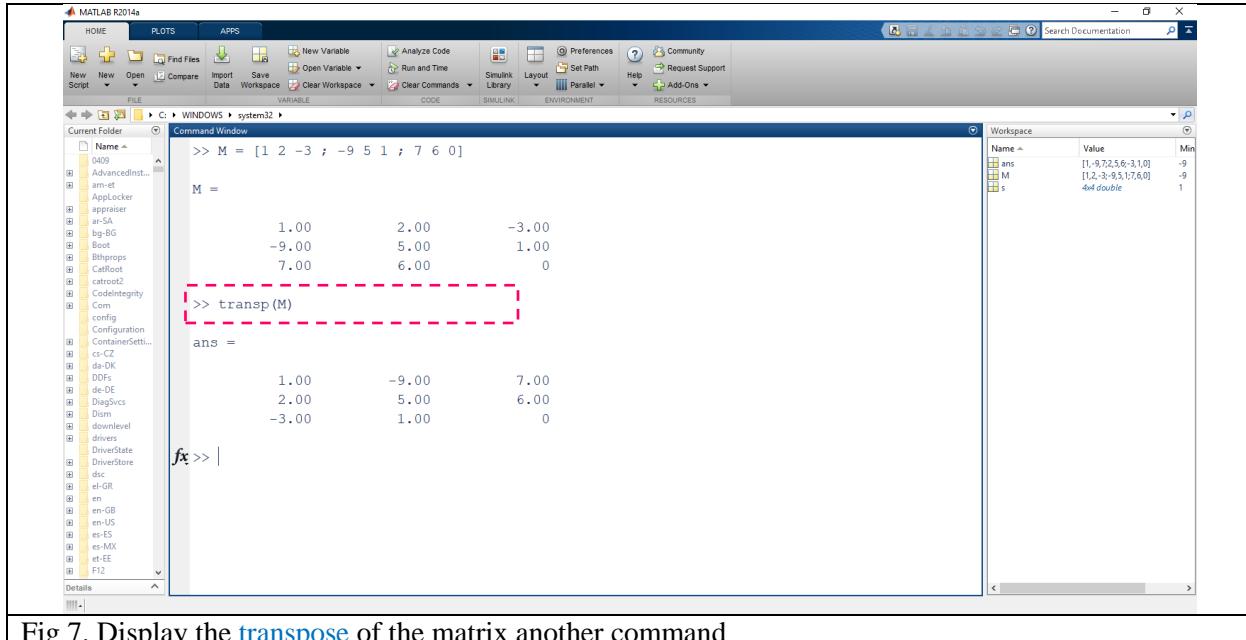
The transpose M' is displayed as:

$$ans = \begin{bmatrix} 1.00 & -9.00 & 7.00 \\ 2.00 & 5.00 & 6.00 \\ -3.00 & 1.00 & 0 \end{bmatrix}$$

The Workspace browser on the right shows the variables ans , M , and s .

Fig 6. Display the transpose of the matrix

Or



The screenshot shows the MATLAB R2014a interface. In the Command Window, a matrix M is defined as:

```
>> M = [1 2 -3 ; -9 5 1 ; 7 6 0]
```

The matrix M is displayed as:

$$M = \begin{bmatrix} 1.00 & 2.00 & -3.00 \\ -9.00 & 5.00 & 1.00 \\ 7.00 & 6.00 & 0 \end{bmatrix}$$

The transpose of M is then calculated and displayed using the `transp` command:

```
>> transp(M)
```

The transpose M' is displayed as:

$$ans = \begin{bmatrix} 1.00 & -9.00 & 7.00 \\ 2.00 & 5.00 & 6.00 \\ -3.00 & 1.00 & 0 \end{bmatrix}$$

The Workspace browser on the right shows the variables ans , M , and s .

Fig 7. Display the transpose of the matrix another command



3.3.The *inv* command

To display the inverse of such matrix just type the command/function *inv(.)* and the inverse of the matrix will be printed in the screen

The screenshot shows the MATLAB R2014a interface. In the Command Window, a matrix *M* is defined as $\begin{bmatrix} 1 & 2 & -3 \\ -9 & 5 & 1 \\ 7 & 6 & 0 \end{bmatrix}$. The inverse of *M* is then calculated using the command *inv(M)*. The resulting inverse matrix *ans* is displayed as follows:

```

>> M = [1 2 -3 ; -9 5 1 ; 7 6 0]
M =
    1.00    2.00   -3.00
   -9.00    5.00    1.00
    7.00    6.00    0.00
>> inv(M)
ans =
   -0.02   -0.07    0.06
    0.03    0.08    0.09
   -0.32    0.03    0.08
>> M^-1
ans =
   -0.02   -0.07    0.06
    0.03    0.08    0.09
   -0.32    0.03    0.08

```

Fig 8. Display the *inverse* of the matrix

3.4.The *size* command

To display the *dimension* of a matrix, just type the command/function *size(.)* and the dimension will be printed in the screen.

The screenshot shows the MATLAB R2014a interface. A matrix *C* is defined as $\begin{bmatrix} 5 & 4 & 5 & 5 & 2 & 5 & 5 & 5 & 4 & 1 & 2 & 1 & 1 & 1 \end{bmatrix}$. The dimension of *C* is then calculated using the command *size(C)*. The result shows the matrix has 3 rows and 15 columns.

```

>> C = [5 4 5 5 2 ; 5 5 5 5 4 ; 1 2 1 1 1]
C =
    5.00    4.00    5.00    5.00    2.00
    5.00    5.00    5.00    5.00    4.00
    1.00    2.00    1.00    1.00    1.00
>> size(C)
ans =
         3.00    5.00

```

Fig 9. Display the *dimension* of the matrix



4. Special or particular matrices

In MATLAB, there are **functions** that automatically generate specific matrices for example containing all **ones**, **zero elements**, **ones** in the **diagonal only**, **arbitrary number**.

```

>> ones(3,3)
ans =
1.00 1.00 1.00
1.00 1.00 1.00
1.00 1.00 1.00

>> zeros(3,3)
ans =
0 0 0
0 0 0
0 0 0

>> eye(3,3)
ans =
1.00 0 0
0 1.00 0
0 0 1.00

```

Fig 10. Create a matrix contain number « 1 » or « 0 » in all the column and row and number « 1 » in diagonal only

And

```

>> magic(3)
ans =
8.00 1.00 6.00
3.00 5.00 7.00
4.00 9.00 2.00

```



Fig 11. Create a random matrix

5. Matrix and submatrix manipulations

5.1. Pick and/or replace command

To show or **peak** or **extract** only one value from matrix already written by MATLAB, we need to know

- ✓ The **name** of the **matrix** first,
- ✓ Second, the **position** of that **element** in **row** and **column** of the matrix

Similarly, to **replace one value from matrix**, we need to know

- ✓ The **name** of the **matrix**
- ✓ Second, specify the **position** of the element by number of **row** and **column** in the matrix
- ✓ Third, give the **new number** or **value**

The screenshot shows the MATLAB R2014a interface. The Command Window on the left shows the following sequence of commands and outputs:

```

>> M = [1 2 -3 ; -9 5 1 ; 7 6 0]
M =
    1.00    2.00   -3.00
   -9.00    5.00    1.00
    7.00    6.00    0.00

>> M(2, 3)
ans =
    1.00

>> M(2, 3) = 99
M =
    1.00    2.00   -3.00
   -9.00    5.00   99.00
    7.00    6.00    0.00

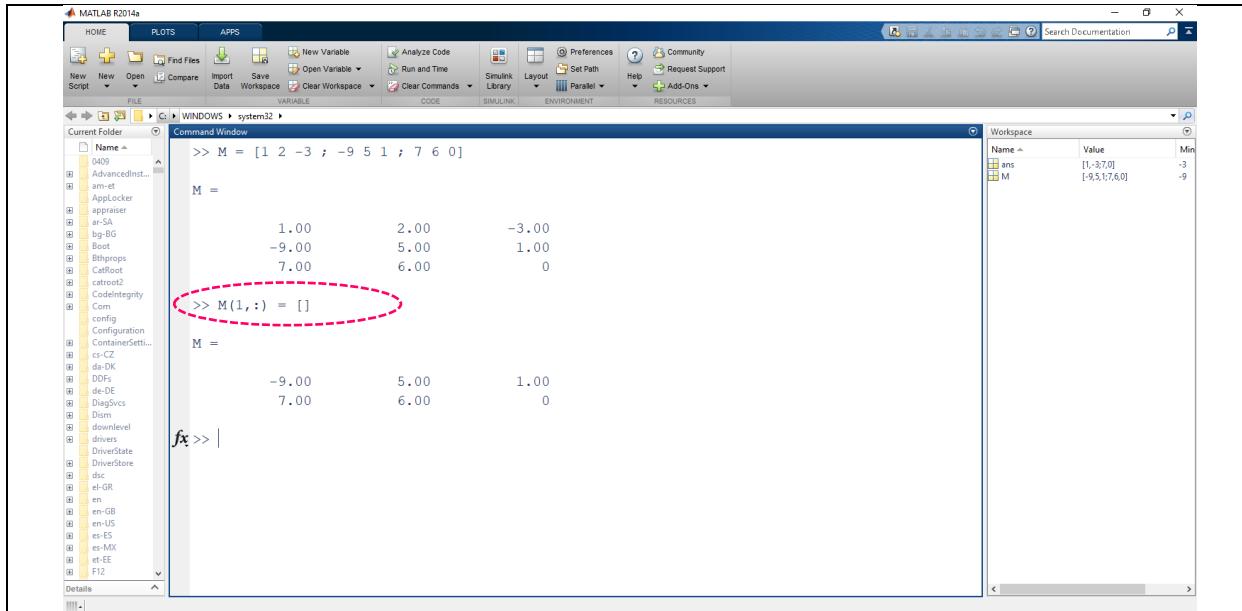
```

The Workspace browser on the right shows the variables ans and M. The variable M is defined as a 3x3 matrix with values [1,2,-3;-9,5,99;7,6,0].

Fig 12. Display or replace such element from a matrix

5.2. Remove a row from a matrix

To **remove** a **row** from a matrix already written by MATLAB, we need to know the **position** or the **order** of this **row** in that matrix

The screenshot shows the MATLAB R2014a interface. In the Command Window, the user has entered the following code:

```
>> M = [1 2 -3 ; -9 5 1 ; 7 6 0]
M =
    1.00    2.00   -3.00
   -9.00    5.00    1.00
    7.00    6.00    0.00
>> M(1,:) = []
M =
   -9.00    5.00    1.00
    7.00    6.00    0.00
fx >> |
```

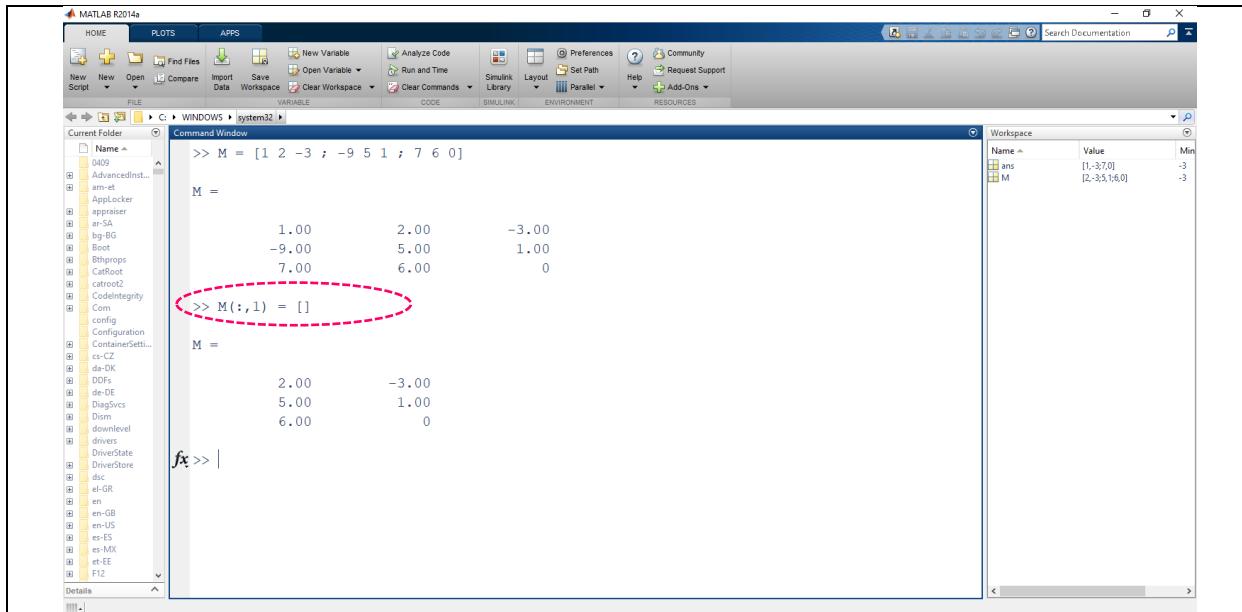
The line `>> M(1,:) = []` is highlighted with a red oval. In the Workspace browser, the variable `M` is shown as a 2x3 matrix:

Name	Value	Min
ans	[1;-3;0]	-3
M	[-9,5,1;6,0]	-9

Fig 13. Display how to **remove a row** from a matrix

5.3.Remove column from a matrix

To **remove a column** from a matrix already written by **MATLAB**, we need to know the **position** or the **order** of this **column** in that matrix



The screenshot shows the MATLAB R2014a interface. In the Command Window, the user has entered the following code:

```
>> M = [1 2 -3 ; -9 5 1 ; 7 6 0]
M =
    1.00    2.00   -3.00
   -9.00    5.00    1.00
    7.00    6.00    0.00
>> M(:,1) = []
M =
    2.00   -3.00
    5.00    1.00
    6.00    0.00
fx >> |
```

The line `>> M(:,1) = []` is highlighted with a red oval. In the Workspace browser, the variable `M` is shown as a 3x2 matrix:

Name	Value	Min
ans	[1;-3;0]	-3
M	[2,-3,5,1;6,0]	-3

Fig 14. Display how to **remove a column** from a matrix



5.4. Add a row to a matrix

To add a **row** from a **matrix** already written by **MATLAB**, we need to respect the dimension of the row in the matrix

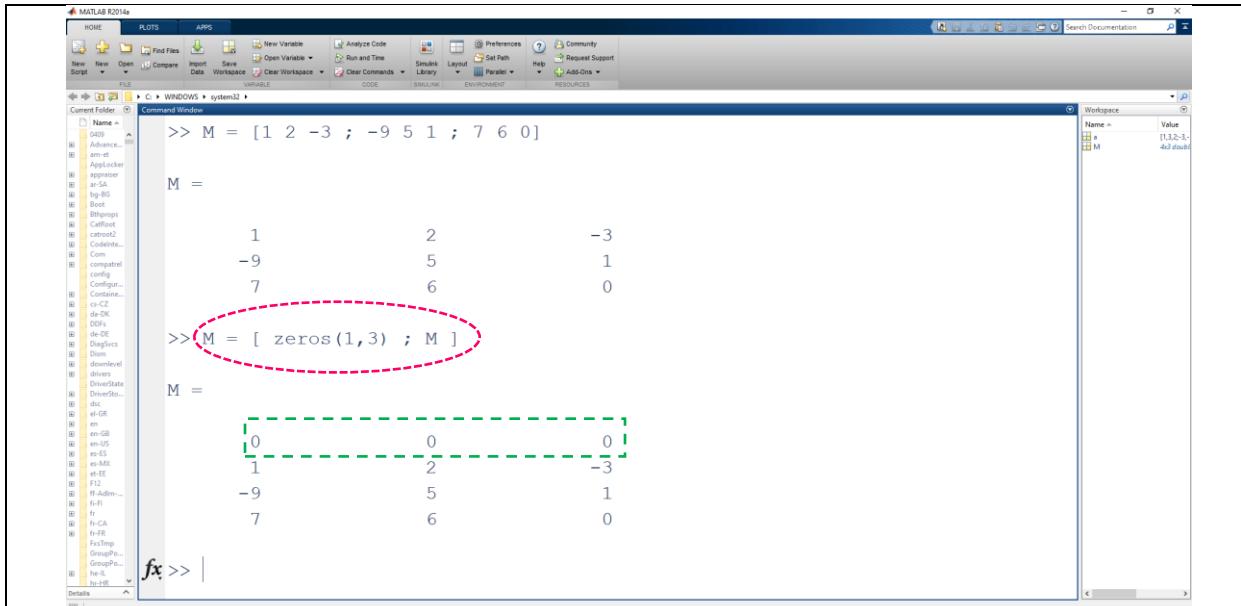


Fig 15. Display how to add a row to a matrix at begin

```

>> M = [ 1 2 -3 ; -9 5 1 ; 7 6 0 ]
M =
1 2 -3
-9 5 1
7 6 0
>> M = [ zeros(1,3) ; M ]
M =
0 0 0
1 2 -3
-9 5 1
7 6 0

```

And

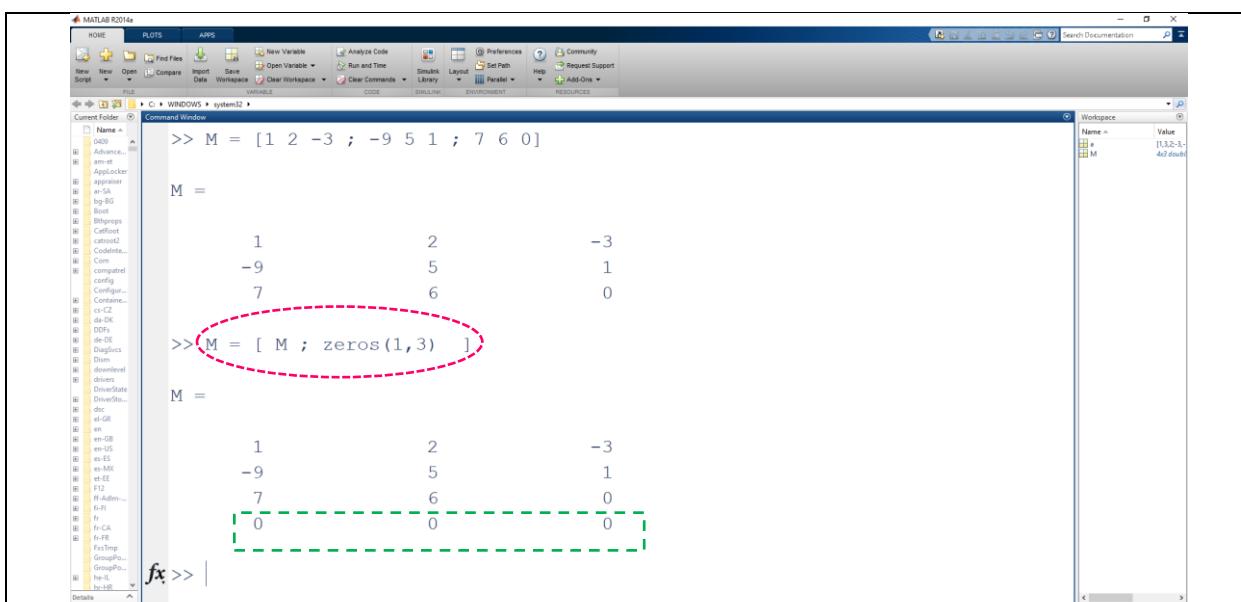


Fig 16. Display how to add a row to a matrix at end

```

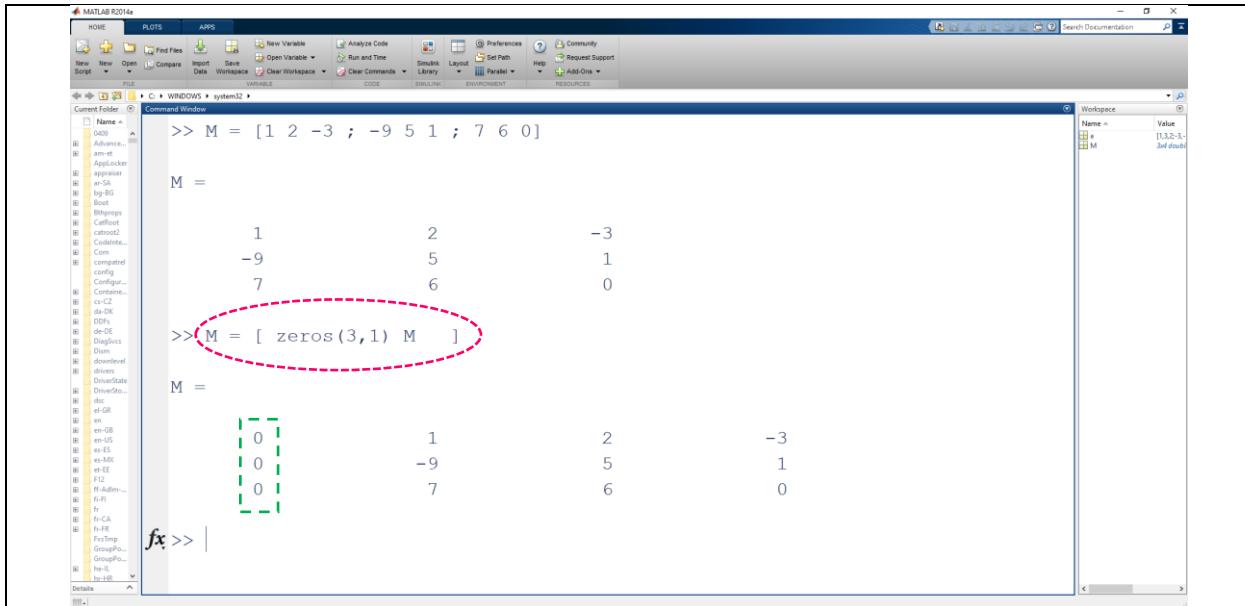
>> M = [ 1 2 -3 ; -9 5 1 ; 7 6 0 ]
M =
1 2 -3
-9 5 1
7 6 0
>> M = [ M ; zeros(1,3) ]
M =
1 2 -3
-9 5 1
7 6 0
0 0 0

```



5.3. Add a column to a matrix

To add a **column** from a **matrix** already written by **MATLAB**, we need to respect the dimension of the column in the matrix



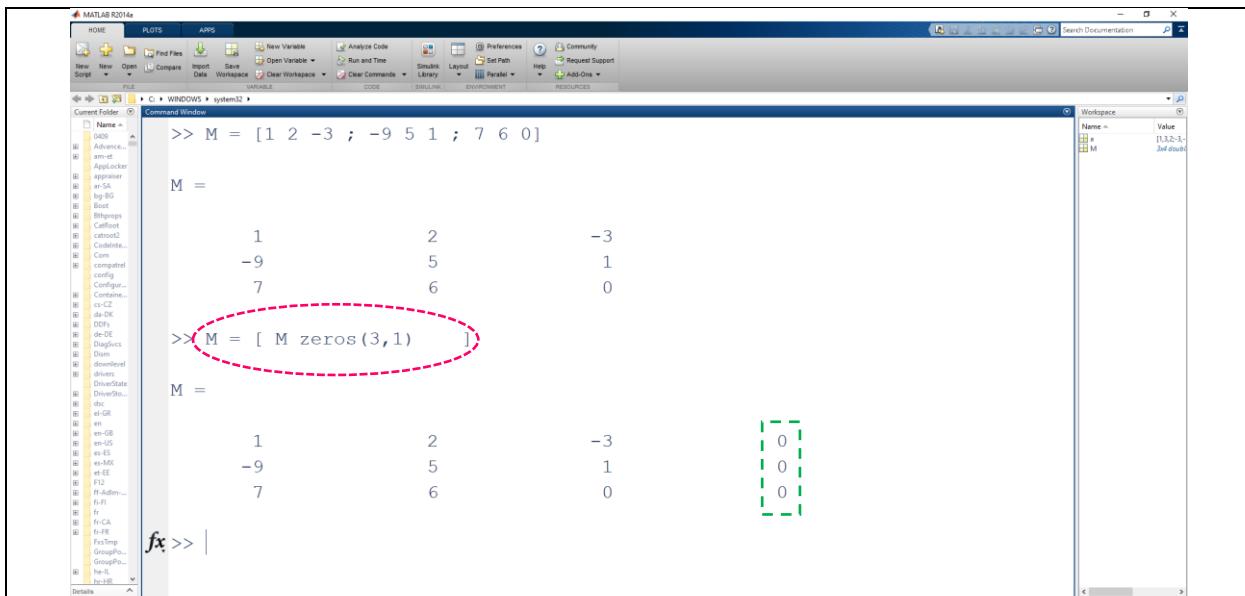
```

MATLAB R2014a
HOME PLOTS APPS
New Script Open Find Files Import Data Save Workspace Open Variable Analyze Code Run and Time Clear Commands Simscape Layout Parallel Help Get Path Preferences Community Request Support
FILE CODE SIMULINK ENVIRONMENT RESOURCES
Current Folder C:\WINDOWS\system32\ Command Window
Name M
>> M = [ 1 2 -3 ; -9 5 1 ; 7 6 0 ]
M =
1 2 -3
-9 5 1
7 6 0
>> M = [ zeros(3,1) M ]
M =
0 1 2 -3
0 -9 5 1
0 7 6 0
fx >> |

```

The code in the Command Window shows the creation of a 3x3 matrix M and then the addition of a new column at the beginning using the command `M = [zeros(3,1) M]`. The resulting matrix M is displayed with a green dashed box highlighting the new column of zeros.

Fig 17. Display how to add a column to a matrix at begin



```

MATLAB R2014a
HOME PLOTS APPS
New Script Open Find Files Import Data Save Workspace Open Variable Analyze Code Run and Time Clear Commands Simscape Layout Parallel Help Get Path Preferences Community Request Support
FILE CODE SIMULINK ENVIRONMENT RESOURCES
Current Folder C:\WINDOWS\system32\ Command Window
Name M
>> M = [ 1 2 -3 ; -9 5 1 ; 7 6 0 ]
M =
1 2 -3
-9 5 1
7 6 0
>> M = [ M zeros(3,1) ]
M =
1 2 -3
-9 5 1
7 6 0
0 0 0
fx >> |

```

The code in the Command Window shows the creation of a 3x3 matrix M and then the addition of a new column at the end using the command `M = [M zeros(3,1)]`. The resulting matrix M is displayed with a green dashed box highlighting the new column of zeros.

Fig 18. Display how to add a column to a matrix at end



6. List of References

Kattan, Peter Issa. Matlab for Beginners: A gentle approach. Petra books, 2008.

Etter, Delores M., David C. Kuncicky, and Douglas W. Hull. Introduction to MATLAB. Vol.4. Hoboken, NJ, USA: Prentice Hall, 2002.

Attaway, Stormy. Matlab: a practical introduction to programming and problem solving. Butterworth-Heinemann, 2013.

Driscoll, Tobin A. Learning Matlab. Society for Industrial and Applied Mathematics, 2009.

Butt, Rizwan. Introduction to numerical analysis using MATLAB. Laxmi Publications, Ltd.,2008.

Sigmon, Kermit. Matlab: aide-mémoire. Springer Science & Business Media, 1999.

Chapman, Stephen J. Essentials of MATLAB programming. Cengage Learning, 2016.