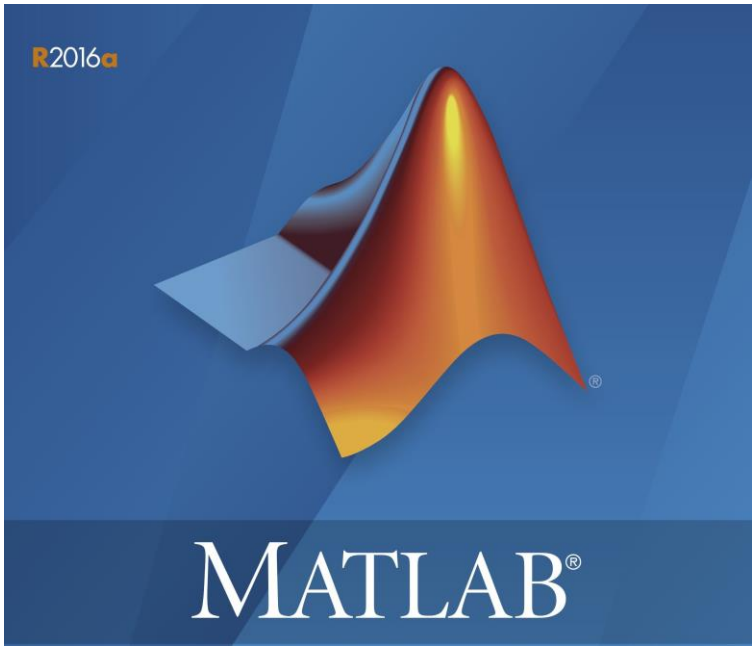
MATLAB

a short primer



Ovidiu Ivanov
'Gheorghe Asachi' Technical University,
Electrical Engineering Faculty
Iasi, Romania
www.tuiasi.ro

Part Four

Matrix processing

MATLAB is for matrices

- MATLAB stands for MATrix LABoratory.
- MATLAB can easily create and modify matrices.

- How to write a matrix directly:
  - Enclose in square brackets
  - Separate columns with space or comma, and rows with semicolons
  - All rows and columns dimensions must agree

  - For vectors, use only one column or one row

Writing vectors and matrices

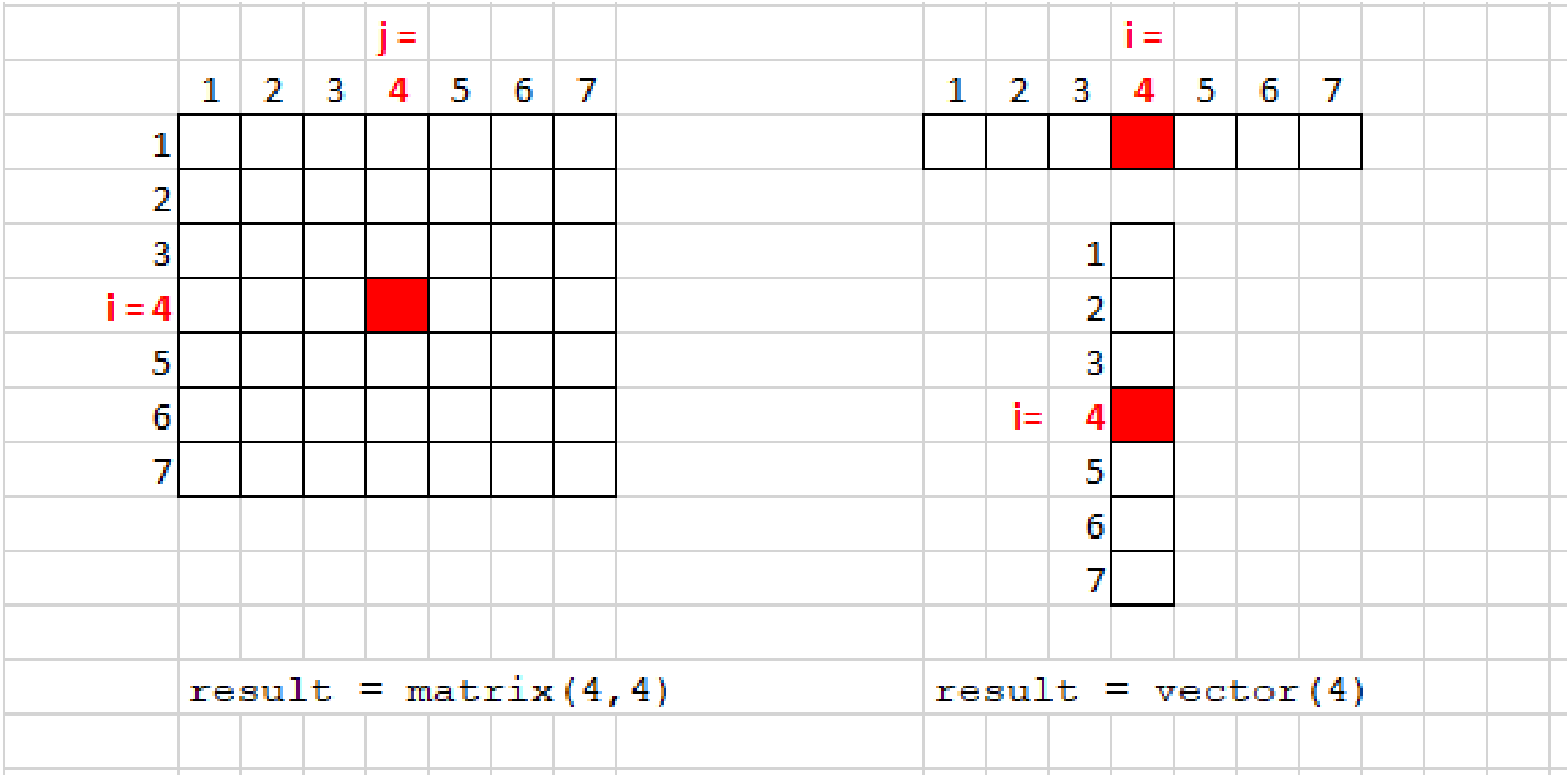
```
>> VEC_ROW=[1 2 3 4 5]

VEC_ROW =

     1     2     3     4     5
```

```
>> VEC_COL=[1;2;3;4;5]

VEC_COL =

     1
     2
     3
     4
     5
```

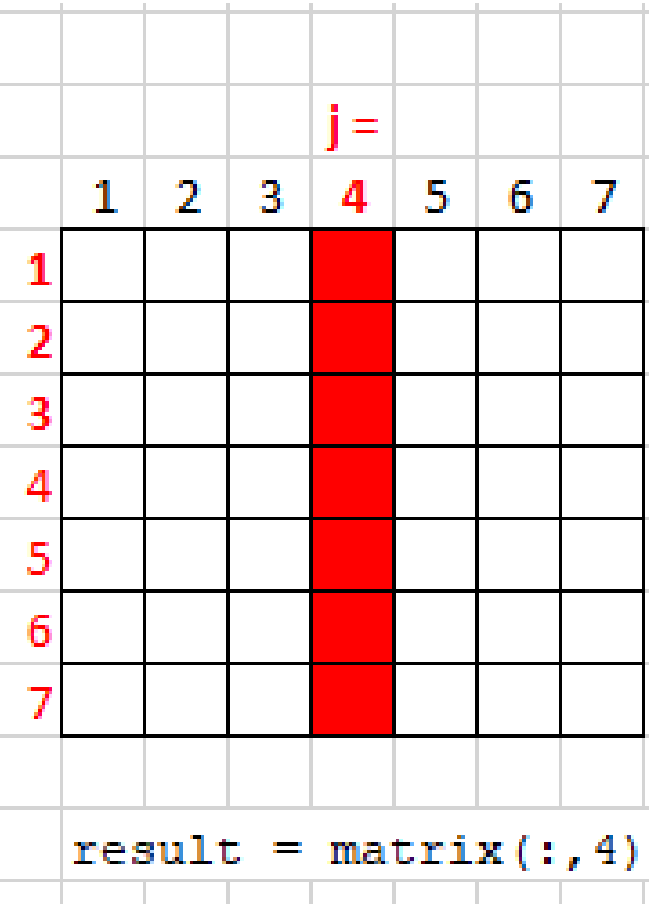
```
>> MATRIX=[1 -1 5 6; 0.14 285 3 1; 46 19 -0.5 2]

MATRIX =

    1.0000   -1.0000    5.0000    6.0000
    0.1400  285.0000    3.0000    1.0000
   46.0000   19.0000   -0.5000    2.0000

fx >>
```

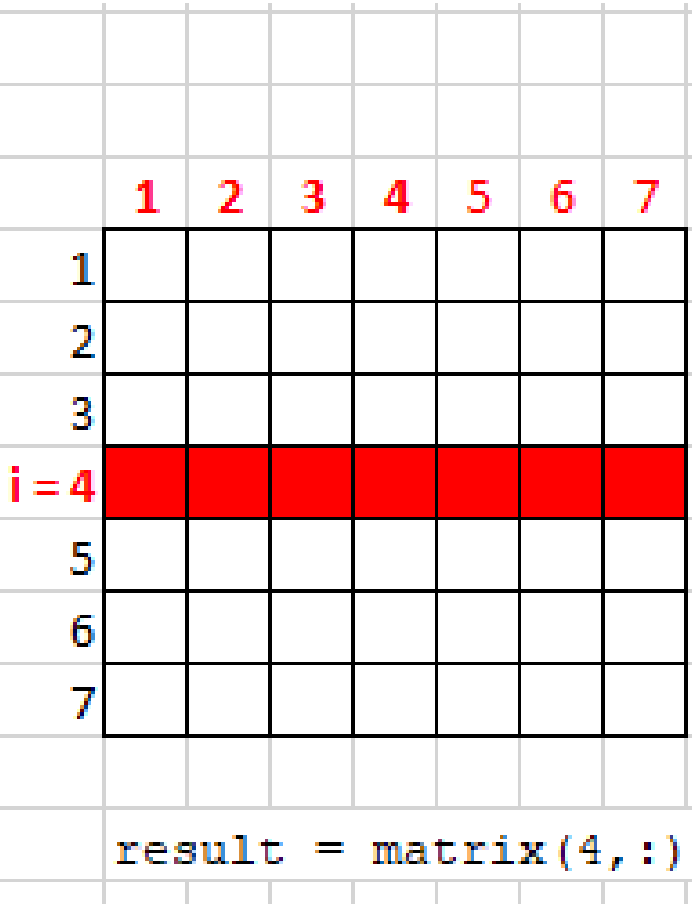
Better store and read them from files !

Addressing one element

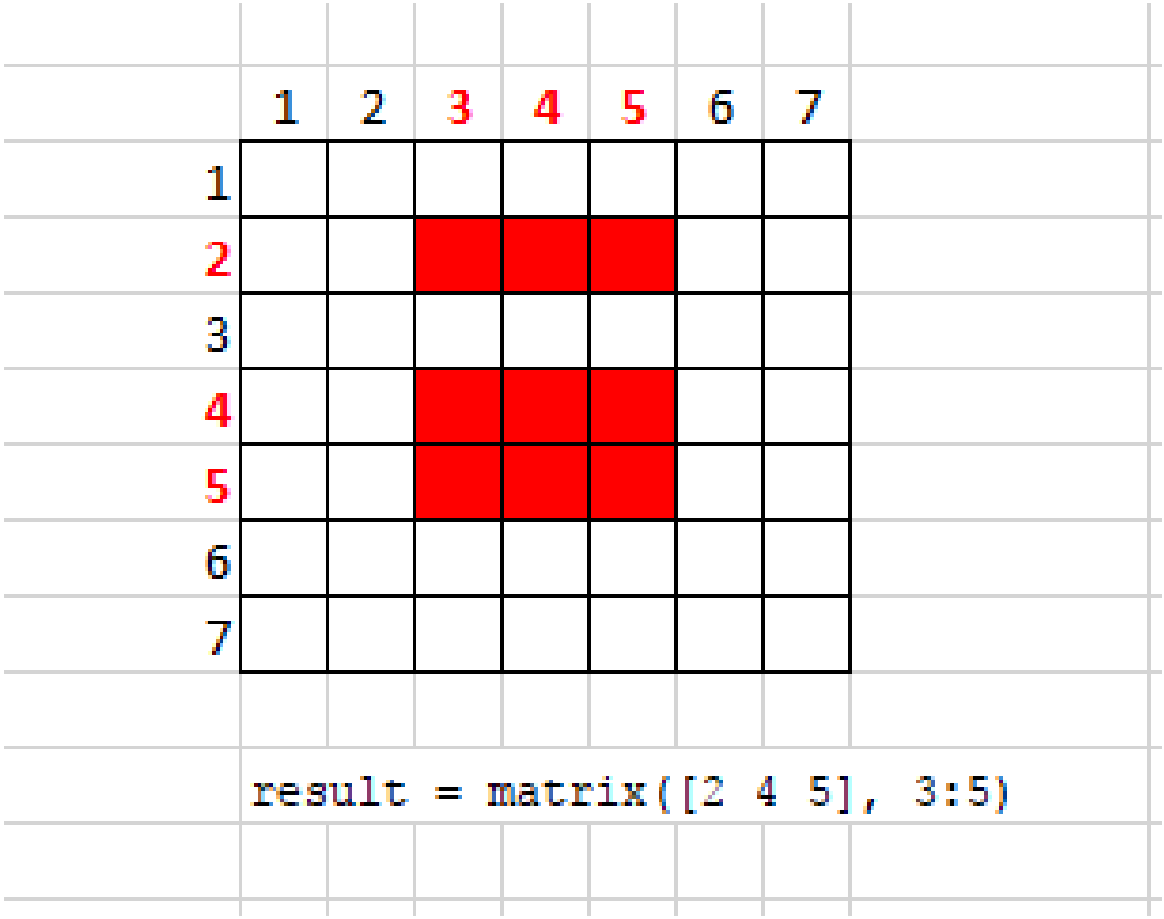


Addressing one column or row





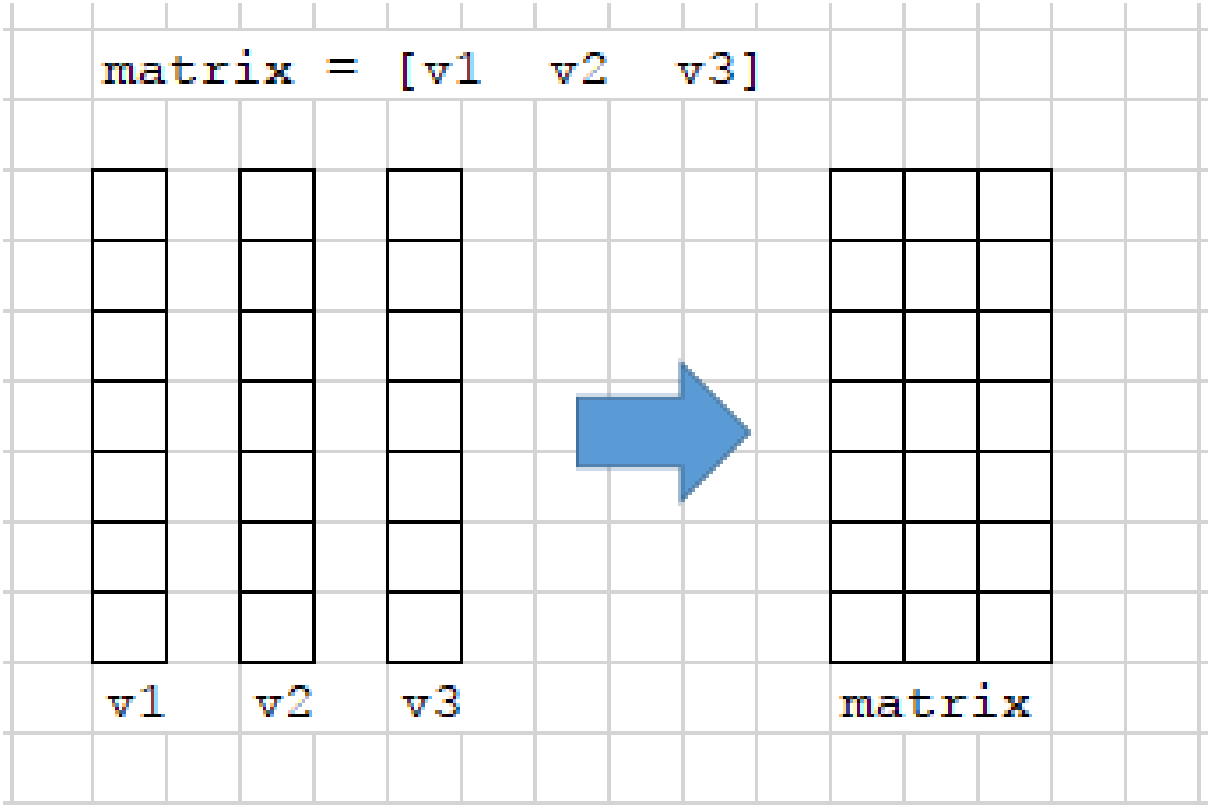
Addressing parts from rows and columns



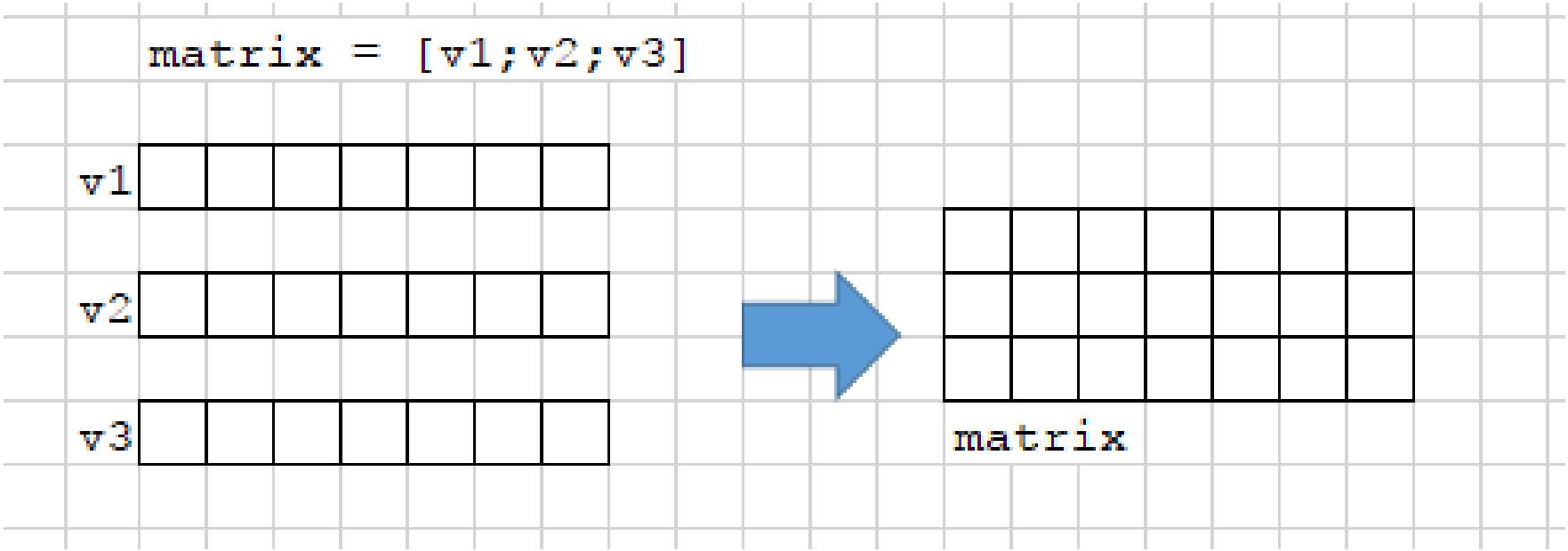
discrete and continuous ranges combined

Building matrices from column vectors or adding columns to a matrix

Dimensions must agree !

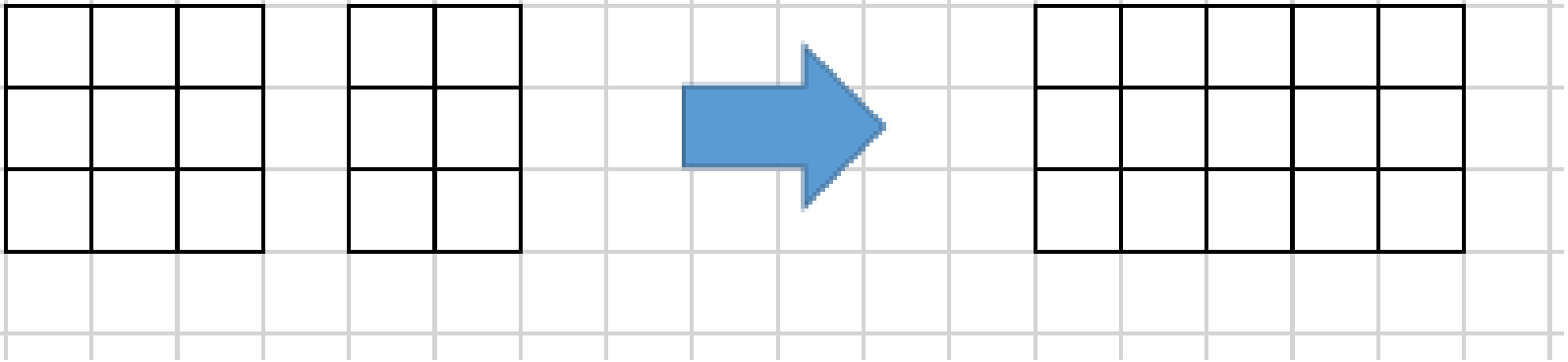


Building matrices from row vectors or adding rows to a matrix



Building matrices from other matrices or extending matrices

```
matrix = [mat1   mat2]
```



Matrix size

size

```
>> MATRIX=[1 -1 5 6; 0.14 285 3 1; 46 19 -0.5 2]

MATRIX =

    1.0000   -1.0000    5.0000    6.0000
    0.1400  285.0000    3.0000    1.0000
   46.0000   19.0000   -0.5000    2.0000

>>
>>
>> size(MATRIX)

ans =

     3     4
```

Vector length

- Actually, the maximum dimension size of a matrix: length

```
>> VEC_ROW=[1 2 3 4 5]

VEC_ROW =

     1     2     3     4     5

>>
>>
>> length(VEC_ROW)

ans =

     5
```

Matrix or vector transpose, rotation

' (prime) – transpose (for complex numbers, use **.'**)

rot90 - rotation

```
>> MATRIX=[1 -1 5 6; 0.14 285 3 1; 46 19 -0.5 2]

MATRIX =

    1.0000   -1.0000    5.0000    6.0000
    0.1400  285.0000    3.0000    1.0000
   46.0000   19.0000   -0.5000    2.0000

>> MATRIX'

ans =

    1.0000    0.1400   46.0000
   -1.0000  285.0000   19.0000
    5.0000    3.0000   -0.5000
    6.0000    1.0000    2.0000
```

```
>> MATRIX=[1 -1 5 6; 0.14 285 3 1; 46 19 -0.5 2]

MATRIX =

    1.0000   -1.0000    5.0000    6.0000
    0.1400  285.0000    3.0000    1.0000
   46.0000   19.0000   -0.5000    2.0000

>>
>> rot90(MATRIX)

ans =

    6.0000    1.0000    2.0000
    5.0000    3.0000   -0.5000
   -1.0000  285.0000   19.0000
    1.0000    0.1400   46.0000
```

Delete rows or columns from matrices

- **Delete column 3 from MATRIX, use [] (the empty variable)**

```
>> MATRIX=[1 -1 5 6; 0.14 285 3 1; 46 19 -0.5 2]

MATRIX =

    1.0000   -1.0000    5.0000    6.0000
    0.1400  285.0000    3.0000    1.0000
   46.0000   19.0000   -0.5000    2.0000

>> MATRIX(:,3)=[]

MATRIX =

    1.0000   -1.0000    6.0000
    0.1400  285.0000    1.0000
   46.0000   19.0000    2.0000
```

Vectors and matrices with zero, one and random elements

zeros(nlines,ncols)

ones(nlines,ncols)

rand(nlines,ncols)

```
>> zeros(2,3)

ans =

     0     0     0
     0     0     0
```

sum, min, max, mean

- works on columns

```
>> MATRIX=[1 -1 5 6; 0.14 285 3 1; 46 19 -0.5 2]

MATRIX =

    1.0000   -1.0000    5.0000    6.0000
    0.1400  285.0000    3.0000    1.0000
   46.0000   19.0000   -0.5000    2.0000

>> sumcol=sum(MATRIX)

sumcol =

   47.1400  303.0000    7.5000    9.0000

>> sumall=sum(sum(MATRIX))

sumall =

  366.6400

fx >>
```

Vectors with consecutive values

hint: just like in a for statement syntax
init_value : step : final_value

```
>> v1=1:10

v1 =

     1     2     3     4     5     6     7     8     9    10

>>
>>
>> v2=-10:2:10

v2 =

   -10    -8    -6    -4    -2     0     2     4     6     8    10

fx >>
```

Two ways to build a vector/matrix

- **Start with empty, then add elements**
  - **Slower, but useful when you don't know the number of elements of your vector**

```
% initialize the vector
vect=[];
i=1;
% add elements to the vector
while i>0.5
    i=rand; % generate a random
number
    vect=[vect i]
end;
```

Two ways to build a vector/matrix

- **Start with some elements, then replace them with other values**
  - **Faster**

```
% initialize the vector
vect=zeros(1,5);
% add elements to the vector
for i=1:5
  vect(i)=i
end;
```

Quickly solve linear equations systems

- The \ (backslash) operator

if [A][x]=[b], then [x]=[A]\[b] is the same as x=inv(A)*b

```
>> A=[2 1 1;-1 1 -1;1 2 3]

A =

     2     1     1
    -1     1    -1
     1     2     3

>> B=[2;3;-10]

B =

     2
     3
   -10
```

```
>> x=A\B

x =

     3
     1
    -5

>> x=inv(A)*B

x =

     3
     1
    -5

fx >>
```

- Matrix manipulation
  - Addressing elements
  - Adding, extracting and deleting rows and columns
  - Size, length
  - Special matrices (zeros, ones, random, eye)

End of Part Four - Review

- **Feel free to contact me with questions !**

Ovidiu Ivanov

Lecturer, PhD

The ‘Gheorghe Asachi’ Technical University

The Electrical Engineering Faculty

www.tuiasi.ro

ovidiuivanov@tuiasi.ro

THANK YOU !