EEE 302 - CONTROL LABORATORY

Res. Asst. Halit Örenbaş

halitorenbas@gmail.com

Res. Asst. Uğur Soydemir

soymedir_ugur@hotmail.com

WHY DO WE USE MATLAB ?

- It is a high-level language for technical calculation
- It offers a development environment for managing code, files and data
- It features interactive tools for exploration, design and iterative solving
- It supports mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
- It can produce high quality two-dimensional and three-dimensional graphics to aid data visualization
- It includes tools to create custom graphical user interfaces
- It can be integrated with external languages, such as C/C++, FORTRAN, Java, COM, and Microsoft Excel

VARIABLES

MATLAB does not require a command to declare variables. A variable is created simply by directly allocating a value to it. For example:

```
>> v = 3

v =

3
```

The variable *v* will take the value 3 and using a new mapping will not change its value. Once the variable is declared, we can use it in calculations.

```
>> v ^ 3

ans =

27

>> v + 5

ans =

8
```

VARIABLES

The value assigned to a variable remains fixed until it is explicitly changed or if the current MATLAB session is closed.

If we now write:

```
>> v = 3 + 7

v =

10
```

then the variable *v* has the value 10 from now on, as shown in the following calculation:

```
>> v ^ 4

ans =

10000
```

VECTOR VARIABLES

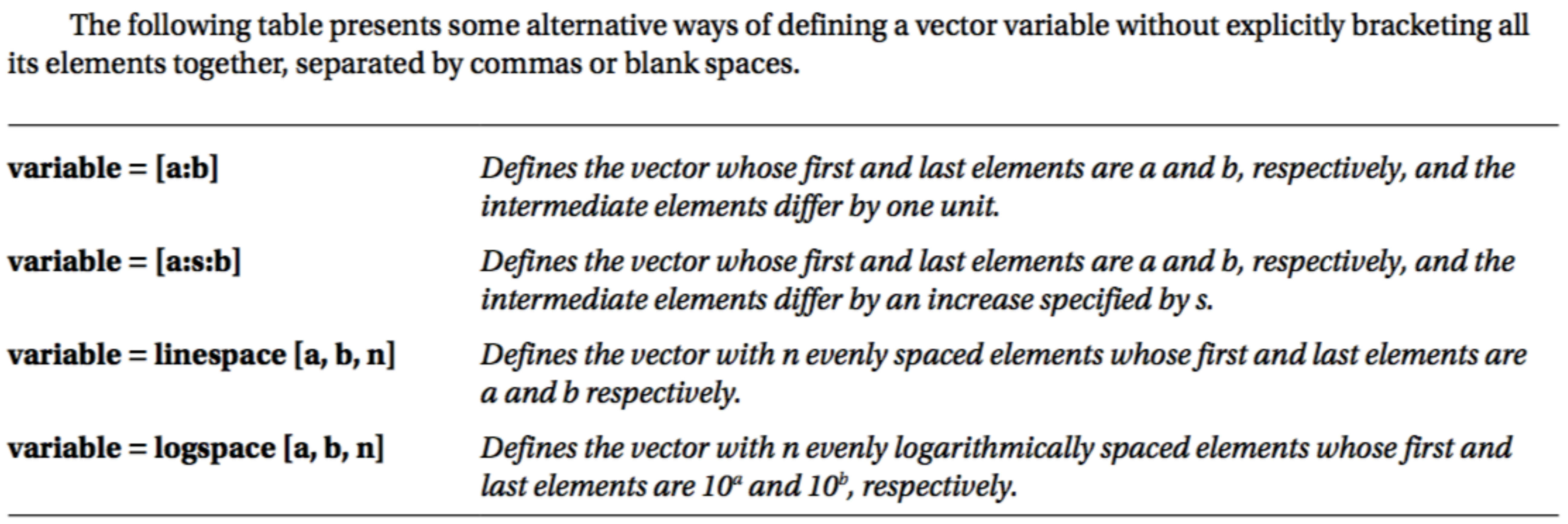
A vector variable of *n* elements can be defined in MATLAB in the following ways:

```
V = [v1, v2, v3,..., vn]

V = [v1 v2 v3... vn]
```

When most MATLAB commands and functions are applied to a vector variable the result is understood to be that obtained by applying the command or function to each element of the vector:

```
>> vector1 = [1,4,9,2.25,1/4]

vector1 =

1.0000 4.0000 9.0000 2.2500 0.2500


>> sqrt (vector1)

ans =

1.0000 2.0000 3.0000 1.5000 0.5000
```

VECTOR VARIABLES



The following table presents some alternative ways of defining a vector variable without explicitly bracketing all its elements together, separated by commas or blank spaces.

variable = [a:b]	*Defines the vector whose first and last elements are a and b, respectively, and the intermediate elements differ by one unit.*
variable = [a:s:b]	*Defines the vector whose first and last elements are a and b, respectively, and the intermediate elements differ by an increase specified by s.*
variable = linespace [a, b, n]	*Defines the vector with n evenly spaced elements whose first and last elements are a and b respectively.*
variable = logspace [a, b, n]	*Defines the vector with n evenly logarithmically spaced elements whose first and last elements are 10^a and 10^b, respectively.*

VECTOR VARIABLES

Below are some examples:

```
>> vector2 = [5:5:25]

vector2 =

5 10 15 20 25
```

This yields the numbers between 5 and 25, inclusive, separated by 5 units.

```
>> vector3=[10:30]

vector3 =

Columns 1 through 13

10    11    12    13    14    15    16    17    18    19      20    21    22

Columns 14 through 21

23 24 25 26 27 28 29 30
```

This yields the numbers between 10 and 30, inclusive, separated by a unit.

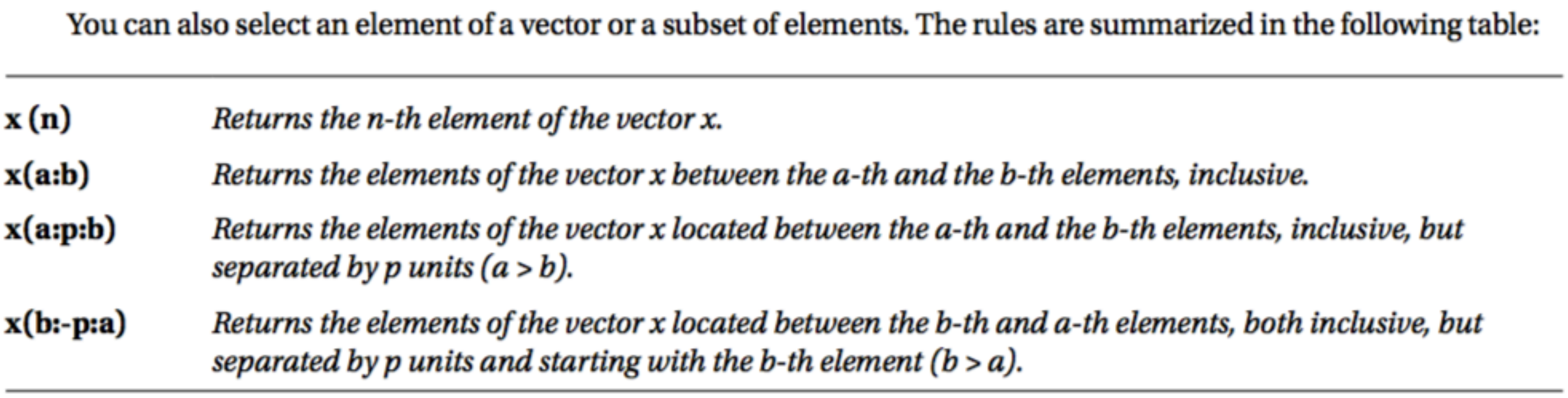
VECTOR VARIABLES

One can also consider row vectors and column vectors in MATLAB. A column vector is obtained by separating its elements by semicolons, or by transposing a row vector using a single quotation mark at the end of its definition.

```
>> a = [10;20;30;40]

a =

10
20
30
40
```

```
>> a = (10:14);b = a'

b =

10
11
12
13
14
```

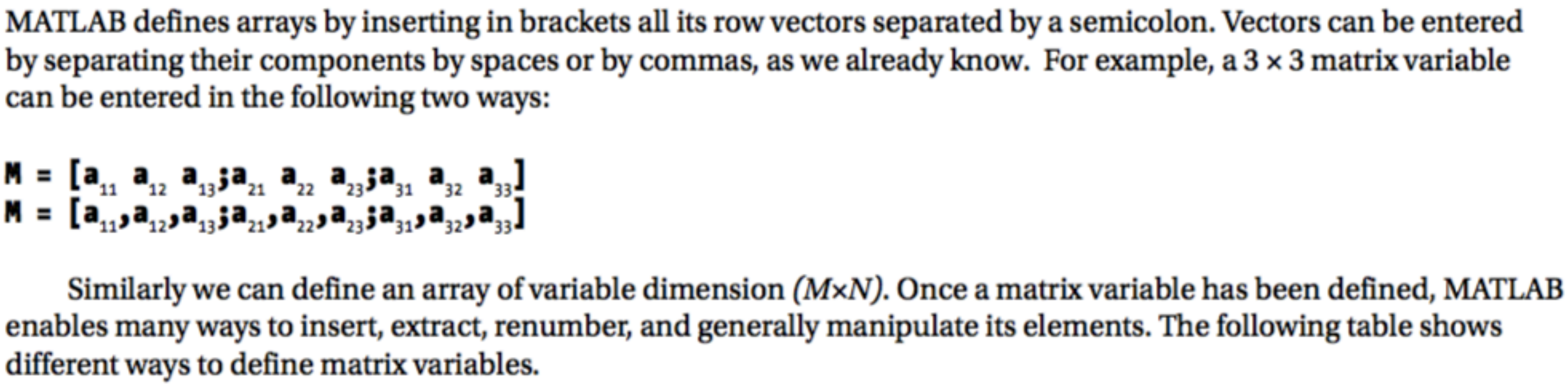
VECTOR VARIABLES



You can also select an element of a vector or a subset of elements. The rules are summarized in the following table:

x (n)	*Returns the n-th element of the vector x.*
x(a:b)	*Returns the elements of the vector x between the a-th and the b-th elements, inclusive.*
x(a:p:b)	*Returns the elements of the vector x located between the a-th and the b-th elements, inclusive, but separated by p units (a > b).*
x(b:-p:a)	*Returns the elements of the vector x located between the b-th and a-th elements, both inclusive, but separated by p units and starting with the b-th element (b > a).*

MATRIX VARIABLES

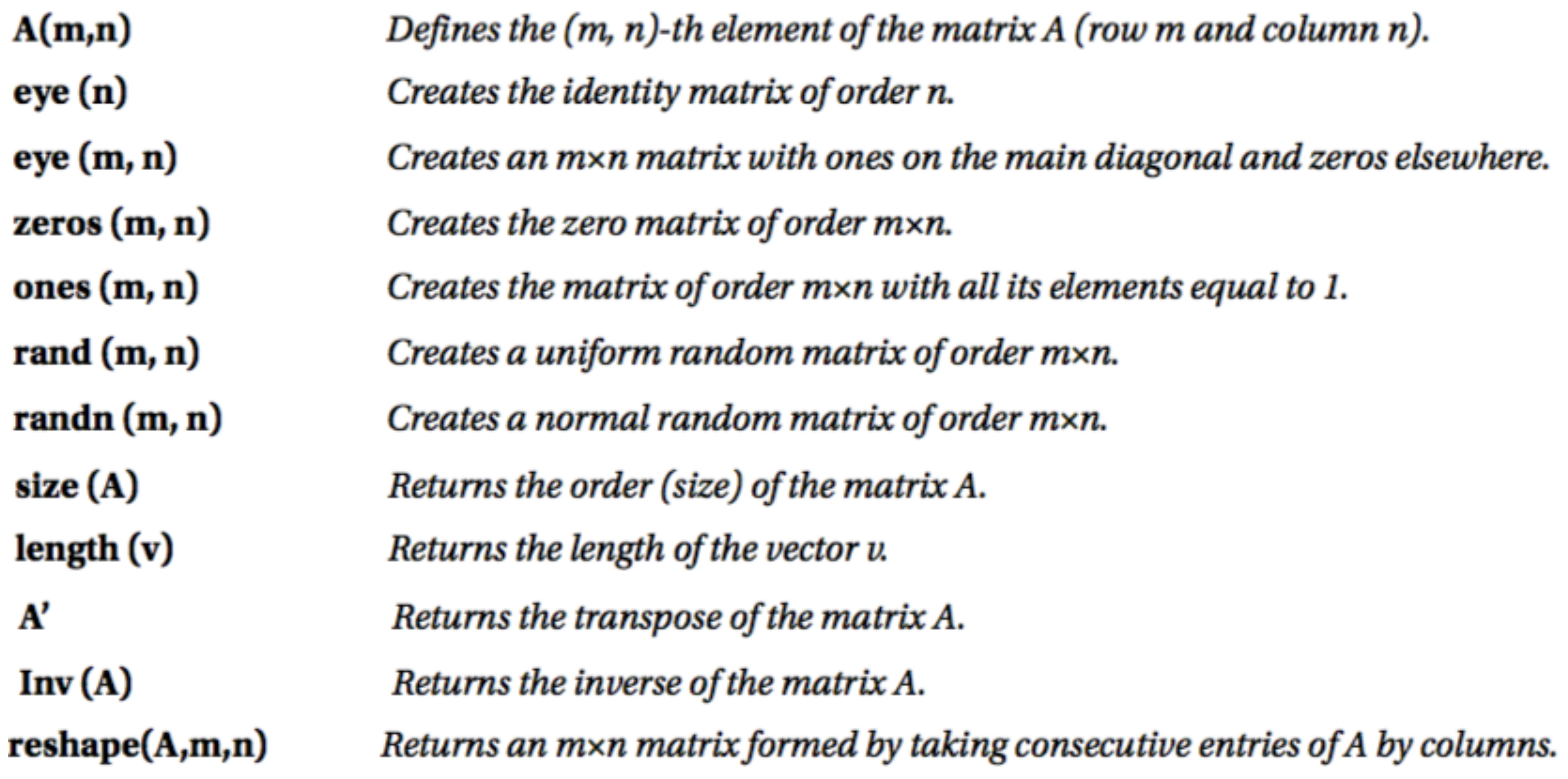


MATLAB defines arrays by inserting in brackets all its row vectors separated by a semicolon. Vectors can be entered by separating their components by spaces or by commas, as we already know. For example, a 3 × 3 matrix variable can be entered in the following two ways:

$$\mathtt{M = [a_{11}\ \ a_{12}\ \ a_{13};a_{21}\ \ a_{22}\ \ a_{23};a_{31}\ \ a_{32}\ \ a_{33}]}$$
$$\mathtt{M = [a_{11},a_{12},a_{13};a_{21},a_{22},a_{23};a_{31},a_{32},a_{33}]}$$

Similarly we can define an array of variable dimension *(M×N)*. Once a matrix variable has been defined, MATLAB enables many ways to insert, extract, renumber, and generally manipulate its elements. The following table shows different ways to define matrix variables.

MATRIX VARIABLES



A(m,n)	*Defines the (m, n)-th element of the matrix A (row m and column n).*
eye (n)	*Creates the identity matrix of order n.*
eye (m, n)	*Creates an m×n matrix with ones on the main diagonal and zeros elsewhere.*
zeros (m, n)	*Creates the zero matrix of order m×n.*
ones (m, n)	*Creates the matrix of order m×n with all its elements equal to 1.*
rand (m, n)	*Creates a uniform random matrix of order m×n.*
randn (m, n)	*Creates a normal random matrix of order m×n.*
size (A)	*Returns the order (size) of the matrix A.*
length (v)	*Returns the length of the vector v.*
A'	*Returns the transpose of the matrix A.*
Inv (A)	*Returns the inverse of the matrix A.*
reshape(A,m,n)	*Returns an m×n matrix formed by taking consecutive entries of A by columns.*

MATRIX VARIABLES

Here are some examples:
We consider first the 2×3 matrix whose rows are the first six consecutive odd numbers:

```
>> A = [1 3 5; 7 9 11]

A =

1 3 5
7 9 11
```

Now we are going to change the *(2,3)-th* element, i.e. the last element of *A*, to zero:

```
>> A(2,3) = 0

A =

1 3 5
7 9 0
```

MATRIX VARIABLES

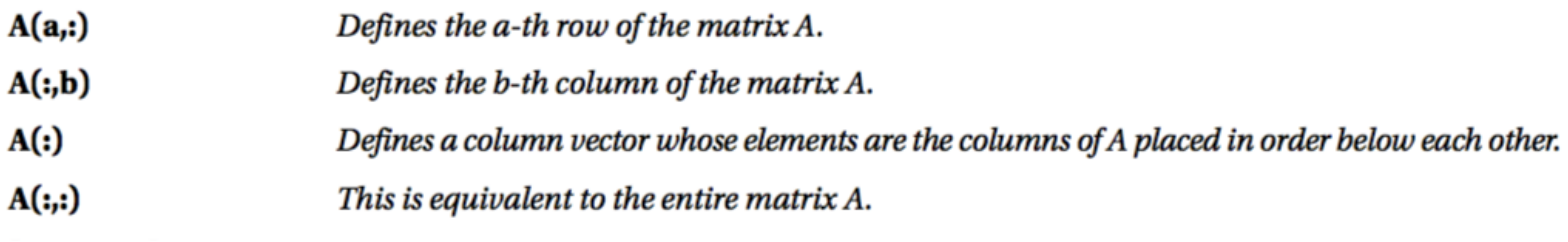
We now define the matrix *B* to be the transpose of *A*:

```
>> B = A'

B =

1 7
3 9
5 0
```

We now construct a matrix *C*, formed by attaching the identity matrix of order 3 to the right of the matrix *B*:

```
>> C = [B eye (3)]

C =

1     7     1     0     0
3     9     0     1     0
5     0     0     0     1
```

MATRIX VARIABLES



A(a,:)	*Defines the a-th row of the matrix A.*
A(:,b)	*Defines the b-th column of the matrix A.*
A(:)	*Defines a column vector whose elements are the columns of A placed in order below each other.*
A(:,:)	*This is equivalent to the entire matrix A.*

Build an array I formed by the identity matrix of order 5×4, appending the zero matrix of the same order to its right and to the right of that the unit matrix, again of the same order. Then extract the first row of I and, finally, form the matrix J comprising the odd rows and even columns of I and calculate its order (size).

question1

Construct a random matrix K of order 3×4, reverse the order of the rows of K, reverse the order of the columns of K and then perform both operations simultaneously. Find the matrix L of order 4×3 whose columns are obtained by taking the elements of K sequentially by columns.

question2