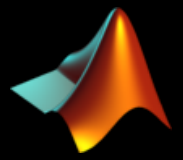

Programming with MATLAB

Operations with matrices

Dimitris Voudouris, PhD
d.s.voudouris@gmail.com

Creating a matrix

Create a matrix with zeros, ones, or threes :

```
a = zeros(3, 5)          % matrix a has 3 rows, 5 columns, all zeros
a = ones(2, 4)           % 2 rows, 4 columns, all ones
a = ones(2, 5) * 3       % 2 rows, 5 columns, all threes
```

Create a matrix with pseudorandom elements:

```
a = rand(5, 3)           % 5x3 matrix of uniform random values
a = randn(5, 3)          % 5x3 matrix of normal random values
a = randi( [3, 9], 4, 6) % 4x6 matrix with random integers
                         between 3 and 9
```

Look up the *help* for three above-mentioned functions to get more insights and details about how they work

Creating a matrix

Reset the seed generation

The “random” values that MATLAB presents, are not really random. They are chosen from a pre-defined set of “random” numbers, so if you call, say, **randi(1, 5)** three times every time you start up your computer, you will be getting the same three sequences of five numbers.

To re-set the “seed” that generates these “random” values, use **rng**

```
rng(1)          % will re-set the seed generator so your rand, randn,
                randi commands will produce the same order of
                outputs.
                You can also use other integers in the brackets
```

help rng

Creating a matrix

Create your own matrix with elements of different values:

```
m1 = [1, 2; 3, 4]          % values 1, 2 in row 1, values 3, 4 in row 2
m2 = [10, 20; 30, 40]      % similarly to the previous command
m3 = [2:2:12; 14:2:24]     % a range of numbers in each row

m4 = repmat(m3, 1, 3)      % copies 3 times the matrix m3 and puts
                           these copies next to each other

m5 = repmat(m3, 2, 1)      % copies 2 times the matrix m3 and puts
                           these copies below each other
```

Creating a matrix

Find out the size of your matrix:

```
[r, c] = size(m3)        % will return two values about matrix m3
                         one for the number of rows (r), and
                         one for the number of columns (c)

length(m3)               % will give you the length (one dimension)
                         this will be the length of the largest dimension!

numel(m3)                % number of elements in matrix m3
length( m3(:) )          % same as numel, because operation (:) will
                         convert matrix to a single column vector
                         This will be done by adding each column
                         below each other
```

Operations with matrices

You can add and subtract matrices

```
m1 + m2        % will perform an element-by-element addition
m1 – m2        % will perform an element-by-element subtraction
```

MATLAB considers a scalar (single value) as a 1x1 matrix, so:

```
m1 + 5         % will add the value of 5 to each element of m1
m2 - 3         % will add the value of 5 to each element of m1
```

Operations with matrices

Multiplication and division of matrices

m1 * m2 % will do a matrix multiplication based on rules of linear algebra

Optional and short:
In the above operation, the result will be [70, 100; 150, 220] because:

1*10 + 2*30 = **70** 1*20 + 2*40 = **100**

3*10 + 4*30 = **150** 3*20 + 4*40 = **220**

You can read more about matrix multiplication by executing:
help mtimes

Understanding this is ***beyond* the scope of this seminar**

Operations with matrices

Element-by-element multiplication of two matrices

```
m1 .* m2        % make sure you use the dot before the
                multiplication sign

m1 ./ m2        % for division

m1 .^ m2        % to raise each element of m1 to the power of each
                element of matrix m2
m1 .^ 3         % to raise each element of matrix m1 to the power of 3
                (compare to m1 ^ 3 –without the dot…)
```

Access elements

Choose elements of your matrix

Basic rules:

1. Open normal brackets after typing variable name
2. Within brackets, provide two inputs, separated by a comma:
   - First input requests the row(s)
   - Second input requests the column(s)
3. Each input can be a scalar or a vector ***(which requires additional square brackets!)***

For a simple example:

```
m1(1, 2)        % chooses element in row 1, column 2
m1(2, 1)        % element in row 2, column 1
```

Access elements

Choose elements of your matrix

```
m1(:, 1)          % all rows from column 1 (so, complete column 1)
m1(2, :)          % all columns from row 2 (so, complete row 2)
```

You can also use a ***vector*** as input. For this, provide the desired vector (within square brackets) as input to chooses rows and/or columns.

```
m3(1, [2, 3, 5])      % row 1, columns 2, 3 and 5
m3(1, [2:2:end])      % row 1, columns 2 till the end, in steps of 2

m3(2, end) = NaN% will replace the last element of row 2 with a NaN
```

Edit matrices

Create a new 4x3 matrix M

```
M = [1, 2, 3; 4, 5, 6; 7, 8, 9; 10, 11, 12]
```

Create new variables consisting of a selection of matrix M

```
S1 = M([2, 3], :)          % S1 has the elements of M that are in all
                           columns of rows 2 and 3

S2 = M(2, [1, 3])          % S2 has the elements of M that are in row 2
                           but only in columns 1 and 3

new_M = M([2 1 4 3], :)    % new_M has the values of M but with the
                           rows rearranged (row 1 of new_M has the
                           values that are in row 2 of M, etc.)
```

Merge matrices

Combine matrices next to each other, or on top of each other

```
[M, new_M]      % next to each other (use comma or space)
[M; new_M]      % on top of each other (use semicolon)
```

When doing so, the to-be-combined matrices need to have the same number of elements along the 'critical' dimension

You canNOT do

```
[m1; m3]                % because m3 has more columns than m1
```

But, you can do

```
[m1, m3]
```

Calculate max values

Calculate the maximal value of matrix m3

```
max(m3)              % will give the maximal value of each column
max(m3, [], 2)       % ...and of each row
[M, I] = max( m3(:) )        % converts m3 to a column vector and then
                             takes the maximal value of this vector. The
                             output M is a single value (the maximal
                             value), the output I is the index of this value
                             within this column vector

[row, column] = ind2sub ( size(m3), I)          % convert the previously
                    calculated linear index to a subscript for a matrix.
                    Variables row and column will output the row and
                    column of this linear index in your matrix m3
```

Note: the variables ***M, I, row, column*** can have any valid name!

Good luck!

Tips:

- Try out the commands and observe what happens
- Change the commands and try to make MATLAB give you errors:
  Try to understand what and why causes this error.