PROGRAMMING WITH MATLAB

DATA TYPES - 1

WEEK 2

NUMERIC DATA TYPES

MATLAB provides several options for storing numbers as bits (default : double)

Type	Description	Range
uint8	8 bit unsigned integer	Integers from 0 to 255
int8	8 bit signed integer	Integers from -128 to 127
uint16	16 bit unsigned integer	Integers from 0 to 65535
int16	16 bit signed integer	Integers from -32768 to 32767
uint32	32 bit unsigned integer	Integers from 0 to 4294967295
int32	32 bit signed integer	Integers from -2147483648 to 2147483647
single	32 bit floating point	-3.402823E38 to -1.401298E-45 1.401298E-45 to 3.402823E38
double	64 bit floating point	-1.7976931348623E308 to -4.9406564584124E-324 4.9406564584124E-324 to 1.7976931348623E308

STRINGS IN MATLAB

- MATLAB stores strings as an array of characters

```
>> name = 'Cemil‘
>> double(name)
ans =
    67   101   109   105   108
```

- Each letter in the string is represented by a decimal number in the ASCII table

VECTORS

- The elements are separated by commas to create a row vector
- Use square brackets:

```
 >> x = [1, 3, 5]
x =
    1     3     5
```

- a column vector can be created with transpose notation (‘)

```
>> y = [1, 3, 5]'
y =
    1
    3
    5
```

VECTORS

- a column vector can also be created by separating elements with semicolons

```
>> z = [7; 11; 13]
z =
     7
    11
    13
```

- a new vector can be created by appending one vector to another

```
>> x = [1, 3, 5]
>> y = [7, 11, 13]
>> z = [x, y]
z =
     1     3     5     7    11    13
```

VECTORS

- The column operator creates vectors consisting of equally spaced elements

Syntax: x = p : q : r or x = (p : q: r) Do not use sqare brackets!

```
>> x = 1 : 2:  9
x =
    1     3     5     7     9
>> x = 1 : 2: 8
x =
    1     3     5     7
```

- If p - r is an integer multiple of q, then the final value is r. Otherwise, the final value is less than r.
- Step size does not have to be an integer

```
>> y = 1 : 0.5: 4
y =
   1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000
```

- Default step size is 1.

```
>> z = 1:5
z =
    1     2     3     4     5
```

VECTORS

- With the linspace command, evenly spaced row vectors can be created. However, instead of incrementing, the number of elements is specified.

Syntax: x = linspace(p : r : n)

```
>> x = linspace(1,5,7)
x =
   1.0000    1.6667    2.3333    3.0000    3.6667    4.3333    5.0000
```

If n is not specified, the spacing is 1.

- The logspace command creates an array of logarithmically spaced elements

Syntax: x = logspace(p : r : n)

```
>> x = logspace(-0.5,1,4)
x =
   0.3162    1.0000    3.1623   10.0000
```

(4 points between 10^p and 10^q)

If n is not specified, the default number of points is 50.

- The length command gives the number of elements in the vector.

Syntax: length(x)

```
>> x = [17, 19, 23]
>> length(x)
ans =
    3
```

MATRICES

- Spaces or commas separates elements in different columns. Semicolons separate elements in different rows.

```
>> X = [1, 3, 5; 7, 11, 13]
X =
    1     3     5
    7    11    13
```

- Matrices can also be generated from vectors.

```
>> y = [1, 3, 5]
>> z = [7, 11, 13]
>> A = [y z]
A =
    1     3     5     7    11    13
```

or

```
>> A = [y;z]
A =
    1     3     5
    7    11    13
```

or

```
>> A = [[1, 3];[5, 7];[11, 13]]
A =
    1     3
    5     7
   11    13
```

ARRAY INDEXING, ARRAY ADDRESSING

- The colon (:) operator

```
>> x = [11, 13, 17, 19, 23]
```

x(:), returns all the row or column elements of the x vector.

```
>> x(:)
ans =
    11
    13
    17
    19
    23
```

x(a:b), returns the elements a through b of the vector x.

```
>> x(2:4)
ans =
    13    17    19
```

ARRAY ADDRESSING

```
>> A = [1, 3, 5; 7, 11, 13; 17, 19, 23]
A =
     1     3     5
     7    11    13
    17    19    23
```

A(:,2) returns all the elements in the second column of the matrix A.

```
>> A(:,2)
ans =
     3
    11
    19
```

A(:,1:3), returns all the elements in the second through third columns of A.

```
>> A(:,2:3)
ans =
     3     5
    11    13
    19    23
```

ARRAY ADDRESSING

Try the following expressions

```
>> A(2:3,1:2)
ans =
     7    11
    17    19
>> A(:,end)
ans =
     5
    13
    23
>> c = A(:)
c =
     1
     7
    17
     3
    11
    19
     5
    13
    23
```

SOME ARRAY FUNCTIONS

- find: Finds indices of nonzero elements of an array.

Syntax: I = find(X) returns the linear indices corresponding to the nonzero entries of the array X

```
>> X = [1, 0, 5, 0, 0, 7, 9]
>> I = find(X)
I =
    1    3    6    7
```

[I,J] = find(X) returns the row and column indices instead of linear indices into X.

```
>> A = [1, 0, 5; 0, 11, 0; 17, 0, 23]
>> [I, J] = find(A)
I =
    1
    3
    2
    1
    3
J =
    1
    1
    2
    3
    3
```

SOME ARRAY FUNCTIONS

[I,J,V] = find(X) also returns a vector V containing the values that correspond to the row and column indices I and J.

- length: Computes either the number of elements of A if A is a vector or the largest value of m or n if A is an m × n matrix.

Syntax: I = length(A)

```
>> A = [1, 3, 5; 7, 11, 13]
>> length(A)
ans =
    3
```

- max: Largest component. For vectors, max(X) is the largest element in X. For matrices, max(X) is a row vector containing the maximum element from each column.

Syntax: [Y,I] = max(X) returns the indices of the maximum values in vector I.

```
>> max(A)
ans =
    7    11    13
>> x = [11, 13, 17, 19, 23]
>> max(x)
ans =
   23
```

SOME ARRAY FUNCTIONS

- size: Returns a row vector [m n] containing the sizes of the m x n array A.

Syntax: size(A)

```
>> A = [1, 3, 5; 7, 11, 13]
>> size(A)
ans =
    2     3
```

- sort: Sorts each column of the array A in ascending order and returns an array the same size as A.

Syntax: sort(A)

```
>> sort(A)
ans =
    1     3     5
    7    11    13
```

- sum: Sorts each column of the array A in ascending order and returns an array the same size as A.

Syntax: sort(A)

```
>> sum(A)
ans =
    8    14    18
```

MULTIDIMENSIONAL ARRAYS

```
>> A(:,:,1) = [1, 3, 5;7, 11, 13]
>> A(:,:,2) = [17, 19, 23;27, 29, 31]
>> A
A(:,:,1) =
   1     3     5
   7    11    13
A(:,:,2) =
  17    19    23
  27    29    31
```