3.5 USING ARRAYS IN MATLAB BUILT-IN MATH FUNCTIONS

The built-in functions in MATLAB are written such that when the argument (input) is an array, the operation that is defined by the function is executed on each element of the array. (One can think of the operation as element-by-element application of the function.) The result (output) from such an operation is an array in which each element is calculated by entering the corresponding element of the argument (input) array into the function. For example, if a vector with seven elements is substituted in the function `cos(x)`, the result is a vector with seven elements in which each element is the cosine of the corresponding element in `x`. This is shown below.

```
>> x=[0:pi/6:pi]
x =
    0   0.5236   1.0472   1.5708   2.0944    2.6180    3.1416
>>y=cos(x)
y =
   1.0000    0.8660    0.5000    0.0000   -0.5000   -0.8660   -
1.0000
>>
```

An example in which the argument variable is a matrix is:

```
>> d=[1 4 9; 16 25 36; 49 64 81]
d =
     1     4     9
    16    25    36
    49    64    81
>> h=sqrt(d)
h =
     1     2     3
     4     5     6
     7     8     9
```

Creating a 3×3 array.

`h` is a 3×3 array in which each element is the square root of the corresponding element in array `d`.

The feature of MATLAB in which arrays can be used as arguments in functions is called vectorization.

3.6 BUILT-IN FUNCTIONS FOR ANALYZING ARRAYS

MATLAB has many built-in functions for analyzing arrays. Table 3-1 lists some of these functions.

Table 3-1: Built-in array functions

Function	Description	Example

`mean(A)`	If `A` is a vector, returns the mean value of the elements of the vector.	`>> A=[5 9 2 4];`
		`>> mean(A)`
		`ans =`
		`5`

`C=max(A)`
 	
`[d,n]=max(A)`
	If `A` is a vector, `C` is the largest element in `A`. If `A` is a matrix, `C` is a row vector containing the largest element of each column of `A`.
	
	If `A` is a vector, `d` is the largest element in `A`, and `n` is the position of the element (the first if several have the max value).
		`>> A=[5 9 2 4 11 6 11 1];`
		`>> C=max(A)`
		`C =`
		`11`
		
		`>> [d,n]=max(A)`
		`d =`
		`11`
		`n =`
		`5`

`min(A)`
 	
`[d,n]=min(A)`
	The same as `max(A)`, but for the smallest element.
	
	The same as `[d,n]=max(A)`, but for the smallest element.
		`>> A=[5 9 2 4];`
		`>> min(A)`
		`ans =`
		`2`

`sum(A)`	If `A` is a vector, returns the sum of the elements of the vector.	`>> A=[5 9 2 4];`
		`>> sum(A)`
		`ans =`
		`20`

`sort(A)`	If `A` is a vector, arranges the elements of the vector in ascending order.	`>> A=[5 9 2 4];`
		`>> sort(A)`
		`ans =`
		`2 4 5 9`

`median(A)`	If `A` is a vector, returns the median value of the elements of the vector.	`>> A=[5 9 2 4];`
		`>> median(A)`
		`ans =`
		`4.5000`