Table 3-1: Built-in array functions (Continued)

Function	Description	Example

`std(A)`	If `A` is a vector, returns the standard deviation of the elements of the vector.	`>> A=[5 9 2 4];`
		`>> std(A)`
		`ans =`
		`2.9439`

`det(A)`	Returns the determinant of a square matrix `A`.	`>> A=[2 4; 3 5];`
		`>> det(A)`
		`ans =`
		`-2`

`dot(a,b)`	Calculates the scalar (dot) product of two vectors `a` and `b`. The vectors can each be row or column vectors.	`>> a=[1 2 3];`
		`>> b=[3 4 5];`
		`>> dot(a,b)`
		`ans =`
		`26`

`cross(a,b)`	Calculates the cross product of two vectors `a` and `b`, (`axb`). The two vectors must have each three elements.	`>> a=[1 3 2];`
		`>> b=[2 4 1];`
		`>> cross(a,b)`
		`ans =`
		`-5 3 -2`

`inv(A)`	Returns the inverse of a square matrix `A`.	`>> A=[2 -2 1; 3 2 -1; 2 -3 2];`
		`>> inv(A)`
		`ans =`
		`0.2000 0.2000 0`
		`-1.6000 0.4000 1.0000`
		`-2.6000 0.4000 2.0000`

3.7 *Generation of Random Numbers*

Simulations of many physical processes and engineering applications frequently require using a number (or a set of numbers) with a random value. MATLAB has three commands—`rand`, `randn`, and `randi`—that can be used to assign random numbers to variables.

The `rand` command:

The `rand` command generates uniformly distributed random numbers with values between 0 and 1. The command can be used to assign these numbers to a scalar, a vector, or a matrix, as shown in Table 3-2.

Table 3-2: The `rand` command

Command	Description	Example

`rand`	Generates a single random number between 0 and 1.	`>> rand`
		`ans =`
		`0.2311`

`rand(1,n)`	Generates an n-element row vector of random numbers between 0 and 1.	`>> a=rand(1,4)`
		`a =`
		`0.6068 0.4860 0.8913 0.7621`

`rand(n)`	Generates an n × n matrix with random numbers between 0 and 1.	`>> b=rand(3)`
		`b =`
		`0.4565 0.4447 0.9218`
		`0.0185 0.6154 0.7382`
		`0.8214 0.7919 0.1763`

`rand(m,n)`	Generates an m × n matrix with random numbers between 0 and 1.	`>> c=rand(2,4)`
		`c =`
		`0.4057 0.9169 0.8936 0.3529`
		`0.9355 0.4103 0.0579 0.8132`

`randperm(n)`	Generates a row vector with n elements that are random permutation of integers 1 through n.	`>> randperm(8)`
		`ans =`
		`8 2 7 4 3 6 5 1`

Sometimes there is a need for random numbers that are distributed in an interval other than (0,1), or for numbers that are integers only. This can be done using mathematical operations with the `rand` function. Random numbers that are distributed in a range (*a*,*b*) can be obtained by multiplying `rand` by (*b* – *a*) and adding the product to *a*:

$$(b - a)*\texttt{rand} + a$$

For example, a vector of 10 elements with random values between –5 and 10 can be created by (*a* = –5, *b* = 10):

```
>> v=15*rand(1,10)-5
v =
   -1.8640    0.6973    6.7499    5.2127    1.9164    3.5174
6.9132   -4.1123    4.0430   -4.2460
```

The `randi` command:

The `randi` command generates uniformly distributed random integer. The command can be used to assign these numbers to a scalar, a vector, or a matrix, as shown in Table 3-3.

Table 3-3: The `randi` command

Command	Description	Example

`randi(imax)`
 	
(imax is an integer)
	Generates a single random number between 1 and imax.	`>> a=randi(15)`
		`a =`
		`9`

`randi(imax,n)`	Generates an `n × n` matrix with random integers between 1 and imax.	`>> b=randi(15,3)`
		`b =`
		`4 8 11`
		`14 3 8`
		`1 15 8`

`randi(imax,m,n)`	Generates an `m × n` matrix with random integers between 1 and imax.	`>> c=randi(15,2,4)`
		`c =`
		`1 1 8 13`
		`11 2 2 13`

The range of the random integers can be set to be between any two integers by typing `[imin imax]` instead of `imax`. For example, a 3×4 matrix with random integers between 50 and 90 is created by:

```
>> d=randi([50 90],3,4)
d =
    57    82    71    75
    66    52    67    61
    84    66    76    67
```

The `randn` command:

The `randn` command generates normally distributed numbers with mean 0 and standard deviation of 1. The command can be used to generate a single number, a vector, or a matrix in the same way as the `rand` command. For example, a 3×4 matrix is created by:

```
>> d=randn(3,4)
d =
   -0.4326    0.2877    1.1892    0.1746
   -1.6656   -1.1465   -0.0376   -0.1867
    0.1253    1.1909    0.3273    0.7258
```

The mean and standard deviation of the numbers can be changed by mathematical operations to have any values. This is done by multiplying the number generated by the `randn` function by the desired standard deviation, and adding the desired mean. For example, a vector of six numbers with a mean of 50 and standard deviation of 6 is generated by: