

# zeros

Create array of all **zeros**

## Syntax

```
B = zeros(n)
B = zeros(m,n)
B = zeros([m n])
B = zeros(m,n,p,...)
B = zeros([m n p ...])
B = zeros(size(A))
zeros(m, n, ...,classname)
zeros([m,n,...],classname)
```

## Description

`B = zeros(n)` returns an  $n$ -by- $n$  matrix of **zeros**. An error message appears if  $n$  is not a scalar.

`B = zeros(m,n)` or `B = zeros([m n])` returns an  $m$ -by- $n$  matrix of **zeros**.

`B = zeros(m,n,p,...)` or `B = zeros([m n p ...])` returns an  $m$ -by- $n$ -by- $p$ -by-... array of zeros.

**Note** The size inputs  $m$ ,  $n$ ,  $p$ , ... should be nonnegative integers. Negative integers are treated as 0.

`B = zeros(size(A))` returns an array the same size as `A` consisting of all **zeros**.

`zeros(m, n, ...,classname)` or `zeros([m,n,...],classname)` is an  $m$ -by- $n$ -by-... array of **zeros** of data type `classname`. `classname` is a string specifying the data type of the output. `classname` can have the following values: 'double', 'single', 'int8', 'uint8', 'int16', 'uint16', 'int32', 'uint32', 'int64', or 'uint64'.

## Example

```
x = zeros(2,3,'int8');
```

## Remarks

The MATLAB language does not have a dimension statement; MATLAB automatically allocates storage for matrices. Nevertheless, for large matrices, MATLAB programs may execute faster if the `zeros` function is used to set aside storage for a matrix whose elements are to be generated one at a time, or a row or column at a time. For example

```
x = zeros(1,n);  
for i = 1:n, x(i) = i; end
```

## See Also

[eye](#), [ones](#), [rand](#), [randn](#), [complex](#)

← xslt

zip →

© 1984–2006 The MathWorks, Inc. · [Terms of Use](#) · [Patents](#) · [Trademarks](#) · [Acknowledgments](#)