



Scipy.org (<http://scipy.org/>) Docs (<http://docs.scipy.org/>)

NumPy v1.9 Manual ([../../index.html](http://docs.scipy.org/doc/numpy-1.9.0/index.html)) NumPy Reference ([../index.html](http://docs.scipy.org/doc/numpy-1.9.0/reference/index.html))

Routines ([../routines.html](http://docs.scipy.org/doc/numpy-1.9.0/routines.html)) Linear algebra (`numpy.linalg`) ([../routines.linalg.html](http://docs.scipy.org/doc/numpy-1.9.0/routines.linalg.html))

index ([../../genindex.html](http://docs.scipy.org/doc/numpy-1.9.0/genindex.html)) next ([numpy.tensordot.html](http://docs.scipy.org/doc/numpy-1.9.0/routines.linalg.html)) previous ([numpy.inner.html](http://docs.scipy.org/doc/numpy-1.9.0/routines.linalg.html))

numpy.outer

numpy.outer(*a*, *b*, *out*=None)

[[source](#)]

(<http://github.com/numpy/numpy/blob/v1.9.1/numpy/core/numeric.py#L998>)

Compute the outer product of two vectors.

Given two vectors, *a* = [*a*₀, *a*₁, ..., *a*_{*M*}] and *b* = [*b*₀, *b*₁, ..., *b*_{*N*}], the outer product [R55] is:

```
[[a0*b0  a0*b1  ...  a0*bN ]
 [a1*b0      .
 [ ...      .
 [aM*b0      aM*bN ]]
```

Parameters: *a* : (*M*,) *array_like*

First input vector. Input is flattened if not already 1-dimensional.

b : (*N*,) *array_like*

Second input vector. Input is flattened if not already 1-dimensional.

out : (*M*, *N*) *ndarray*, *optional*

A location where the result is stored
New in version 1.9.0.

Returns:

out : (*M*, *N*) *ndarray*

out[*i*, *j*] = *a*[*i*] * *b*[*j*]

See also:

[inner](#) ([numpy.inner.html#numpy.inner](#)), [einsum](#) ([numpy.einsum.html#numpy.einsum](#))

References

[R55] (1, 2) : G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed., Baltimore, MD, Johns Hopkins University Press, 1996, pg. 8.

Examples

Make a (very coarse) grid for computing a Mandelbrot set:

```

>>> r1 = np.outer(np.ones((5,)), np.linspace(-2, 2, 5))
>>> r1
array([[ -2.,  -1.,   0.,   1.,   2.],
       [ -2.,  -1.,   0.,   1.,   2.],
       [ -2.,  -1.,   0.,   1.,   2.],
       [ -2.,  -1.,   0.,   1.,   2.],
       [ -2.,  -1.,   0.,   1.,   2.]])
>>> im = np.outer(1j*np.linspace(2, -2, 5), np.ones((5,)))
>>> im
array([[ 0.+2.j,  0.+2.j,  0.+2.j,  0.+2.j,  0.+2.j],
       [ 0.+1.j,  0.+1.j,  0.+1.j,  0.+1.j,  0.+1.j],
       [ 0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j],
       [ 0.-1.j,  0.-1.j,  0.-1.j,  0.-1.j,  0.-1.j],
       [ 0.-2.j,  0.-2.j,  0.-2.j,  0.-2.j,  0.-2.j]])
>>> grid = r1 + im
>>> grid
array([[ -2.+2.j,  -1.+2.j,   0.+2.j,   1.+2.j,   2.+2.j],
       [ -2.+1.j,  -1.+1.j,   0.+1.j,   1.+1.j,   2.+1.j],
       [ -2.+0.j,  -1.+0.j,   0.+0.j,   1.+0.j,   2.+0.j],
       [ -2.-1.j,  -1.-1.j,   0.-1.j,   1.-1.j,   2.-1.j],
       [ -2.-2.j,  -1.-2.j,   0.-2.j,   1.-2.j,   2.-2.j]])

```

An example using a “vector” of letters:

```

>>> x = np.array(['a', 'b', 'c'], dtype=object)
>>> np.outer(x, [1, 2, 3])
array([[a, aa, aaa],
       [b, bb, bbb],
       [c, cc, ccc]], dtype=object)

```

Previous topic

[numpy.inner \(numpy.inner.html\)](#)

Next topic

[numpy.tensordot \(numpy.tensordot.html\)](#)