numpy.linalg.eigh

numpy.linalg.eigh(*a*, *UPLO='L'*) **[source] (http://github.com/numpy/numpy/blob/v1.9.1/numpy/linalg/linalg.py#L1116)**

Return the eigenvalues and eigenvectors of a Hermitian or symmetric matrix.

Returns two objects, a 1-D array containing the eigenvalues of *a*, and a 2-D square array or matrix (depending on the input type) of the corresponding eigenvectors (in columns).

Parameters: **A** : *(..., M, M) array*

Hermitian/Symmetric matrices whose eigenvalues and eigenvectors are to be computed.

UPLO : *{'L', 'U'}, optional*

Specifies whether the calculation is done with the lower triangular part of *a* ('L', default) or the upper triangular part ('U').

Returns: **w** : *(..., M) ndarray*

The eigenvalues, not necessarily ordered.

v : *{(..., M, M) ndarray, (..., M, M) matrix}*

The column `v[:, i]` is the normalized eigenvector corresponding to the eigenvalue `w[i]`. Will return a matrix object if *a* is a matrix object.

Raises: **LinAlgError** :

If the eigenvalue computation does not converge.

See also:

eigvalsh **(numpy.linalg.eigvalsh.html#numpy.linalg.eigvalsh)** eigenvalues of symmetric or Hermitian arrays.

eig **(numpy.linalg.eig.html#numpy.linalg.eig)** eigenvalues and right eigenvectors for non-symmetric arrays.

eigvals **(numpy.linalg.eigvals.html#numpy.linalg.eigvals)** eigenvalues of non-symmetric arrays.

Notes

Broadcasting rules apply, see the numpy.linalg documentation for details.

The eigenvalues/eigenvectors are computed using LAPACK routines _ssyevd, _heevd

The eigenvalues of real symmetric or complex Hermitian matrices are always real. [R38] The array *v* of (column) eigenvectors is unitary and *a*, *w*, and *v* satisfy the equations `dot(a, v[:, i]) = w[i] * v[:, i]`.

References

[R38] *(1, 2)* G. Strang, *Linear Algebra and Its Applications*, 2nd Ed., Orlando, FL, Academic Press, Inc., 1980, pg. 222.

Examples

```
>>> from numpy import linalg as LA
>>> a = np.array([[1, -2j], [2j, 5]])
>>> a
array([[ 1.+0.j,  0.-2.j],
       [ 0.+2.j,  5.+0.j]])
>>> w, v = LA.eigh(a)
>>> w; v
array([ 0.17157288,  5.82842712])
array([[-0.92387953+0.j        , -0.38268343+0.j        ],
       [ 0.00000000+0.38268343j,  0.00000000-0.92387953j]])
```

```
>>> np.dot(a, v[:, 0]) - w[0] * v[:, 0] # verify 1st e-val/vec pair
array([2.77555756e-17 + 0.j, 0. + 1.38777878e-16j])
>>> np.dot(a, v[:, 1]) - w[1] * v[:, 1] # verify 2nd e-val/vec pair
array([ 0.+0.j,  0.+0.j])
```

```
>>> A = np.matrix(a) # what happens if input is a matrix object
>>> A
matrix([[ 1.+0.j,  0.-2.j],
        [ 0.+2.j,  5.+0.j]])
>>> w, v = LA.eigh(A)
>>> w; v
array([ 0.17157288,  5.82842712])
matrix([[-0.92387953+0.j        , -0.38268343+0.j        ],
        [ 0.00000000+0.38268343j,  0.00000000-0.92387953j]])
```


Previous topic

numpy.linalg.eig (numpy.linalg.eig.html)

Next topic

numpy.linalg.eigvals (numpy.linalg.eigvals.html)