**Linear Algebra : scipy.linalg**

Linear algebra is concerned with the representation of linear equations in vector spaces and matrices. SciPy is based on the ATLAS LAPACK and BLAS libraries, and it is extraordinarily fast at solving linear algebra tasks. In addition to all of numpy.linalg's functions, scipy.linalg additionally has a number of advanced functions. Additionally, scipy.linalg is faster than numpy.linalg if ATLAS LAPACK and BLAS support are not employed.

**Example 1 : Finding the Inverse**

*import numpy as np*

*from scipy import linalg*

*A = np.array([[1,2,3],[4,5,6],[2,3,8]])*

*print(A)*

*print(linalg.inv(A))*

*Output:*

*[[-1.83333333 0.58333333 0.25 ]*

*[ 1.66666667 -0.16666667 -0.5 ]*

*[-0.16666667 -0.08333333 0.25 ]]*

**Example 2 : Finding the Determinants**

The determinant of a square matrix is the value calculated arithmetically from the coefficients of the matrix.

*import numpy as np*

*from scipy import linalg*

*A = np.array([[1,2], [3,4]])*

*B = linalg.det(A)*

*print(B)*

*Output : -2.0*

**Example 3 : Sparse Eigenvalues**

Eigenvalues are a set of scalars that are related to linear equations. Eigenvalues (eigenvectors) may be found quickly with ARPACK. The scipy.sparse.linalg.eigs and scipy.sparse.linalg.eigsh high-level interfaces include all of ARPACK's capabilities. The eigenvalues of real or complex non symmetric square matrices may be found using the eigs interface, whereas the eigenvalues of real or complex hermitian matrices can be found using the eigsh interface.

For a complex Hermitian or real symmetric matrix, the eigh function solves the generalised eigenvalue issue.

*from scipy.linalg import eigh*

*import numpy as np*

*A = np.array([[1, 2, 3, 4], [4, 3, 2, 1], [1, 4, 6, 3], [2, 3, 2, 5]])*

*a, b = eigh(A)*

*print("Selected eigenvalues :", a)*

*print("Complex ndarray :", b)*

*Output:*

*Selected eigenvalues : [-2.53382695 1.66735639 3.69488657 12.17158399]*

*Complex ndarray : [[ 0.69205614 0.5829305 0.25682823 -0.33954321]*

*[-0.68277875 0.46838936 0.03700454 -0.5595134 ]*

*[ 0.23275694 -0.29164622 -0.72710245 -0.57627139]*

*[ 0.02637572 -0.59644441 0.63560361 -0.48945525]]*

**Example 4 : Single Value Decomposition**

A Singular Value Decomposition (SVD) is an extension of the eigenvalue problem to non-square matrices.

The scipy.linalg.svd module factorises the matrix 'a' into two unitary matrices 'U' and 'Vh', as well as a 1-D array's' of singular values (real, non-negative) such that a == U\*S\*Vh, where 'S' is an appropriately formed zeros matrix with the major diagonal's'.

*#importing the scipy and numpy packages*

*from scipy import linalg*

*import numpy as np*

*#Declaring the numpy array*

*a = np.random.randn(3, 2) + 1.j\*np.random.randn(3, 2)*

*#Passing the values to the eig function*

*U, s, Vh = linalg.svd(a)*

*# printing the result*

*print (U)*

*print (Vh)*

*print (s)*

*Output:*

[[ 0.13738879-0.15696908j 0.55004187+0.7346568j -0.32590374-0.08947127j]

[ 0.83592645-0.06043994j 0.02712865+0.01186676j 0.33643876+0.42837619j]

[-0.3207546 +0.38881293j 0.28183584+0.27825367j 0.76562266+0.05391092j]]

[[ 0.43523679+0.j 0.8137356 -0.3852315j ]

[-0.90031602+0.j 0.39338151-0.18623119j]]

[2.50436725 1.5700067 ]