## Implementing Matrices with Numpy (Numerical Python)

import numpy as np

A = np.array([[1,2,3],[3,4,2]]) // 2 rows and 3 columns, first set of brackets indicate matrix, second set is each row

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

A[1,1] == 4 // 4 is in second row and second column

A.shape == (2,3) // rows by columns

A = np.zeroes((2,3)) // makes matrix full of zero floats

A == array([[0.,0.,0.],[0.,0.,0.]])

b = np.array([[2],[1],[2]]) // vector (3 rows, 1 column)

b == array([[2],

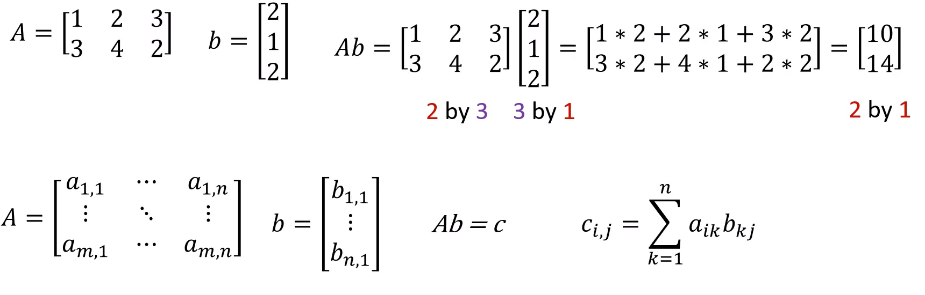
[1],

[2]])

b = np.array([[2,1,2]]) // row vector

b == array([[2,1,2]])

b = np.array([2,1,2]) // row vector implemented 1 dimensionally

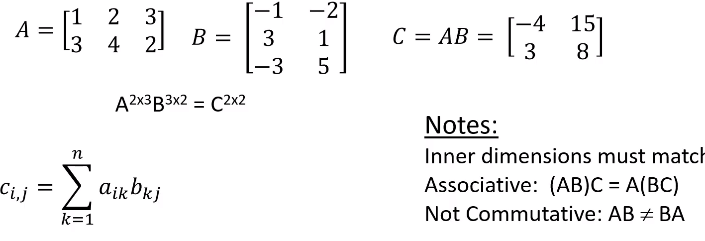


**Multiply a matrix times a vector**

c = np.dot(A,b) // as in dot product

c == array([[10],

[14]])

**Multiply a matrix times a matrix**

d = np.dot(A,B)

d == ([[-4, 15],

[3, 8]])

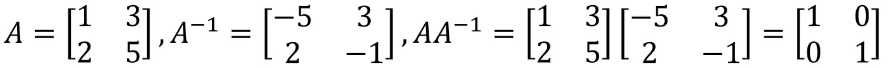
**Identity Matrix and Inverses**

* Identity matrix is a square matrix with 1s on the main diagonal and 0s elsewhere
* 1 0 0

0 1 0

0 0 1

* If A is a square matrix, then AI = IA = A
* Inverse of a square matrix, A-1, has the property that AA-1 = A-1A = I



**Inverse and Identity in numpy**

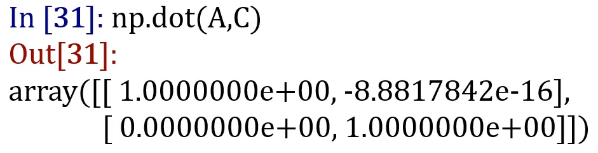
M = np.identity(3)

M == array([1.,0.,0.],

[0.,1.,0.].

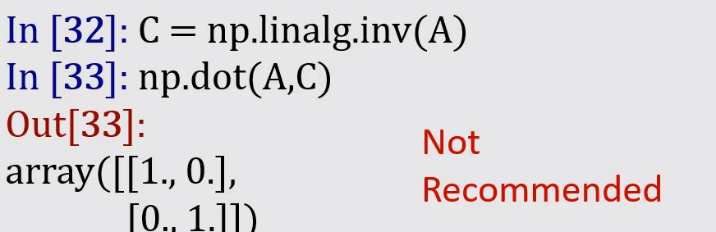
[0.,0.,1.]

A = np.array([[1, 3],

 [2, 5]])

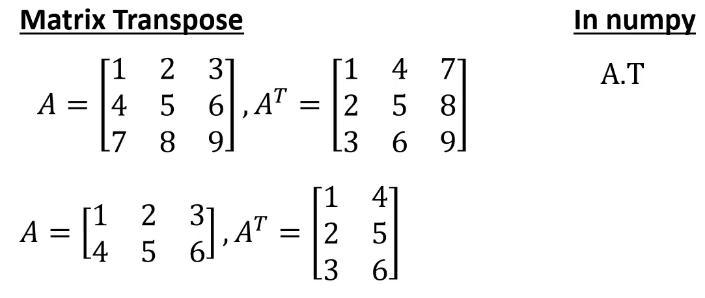
C = np.linalg.pinv(A) // pinv as in pseudo inverse

C == array([[-5.,3.],

 [2.,-1]]) // notice how some arent exactly 0

C = np.linalg.inv(A) // inv as in not psuedo, not recommended

**Matrix transpose**

b = np,array([[1,2,3]])

np.shape(b) == (1, 3)

b.T == array([[1],

[2],

[3]])

**Addition and Subtraction of Matrices**

