

**Algèbre linéaire et
réduction**

Programmation Python Partie 2 : Algèbre linéaire et opérations de réduction

Algèbre linéaire et réduction

Algèbre linéaire Réduction

Algèbre linéaire

```
In [98]: print(A)
```

```
[[ 0  1  2  3]
 [10 11 12 13]
 [20 21 22 23]
 [30 31 32 33]]
```

```
In [99]: print(A.T)  # transposition
```

```
[[ 0 10 20 30]
 [ 1 11 21 31]
 [ 2 12 22 32]
 [ 3 13 23 33]]
```

```
In [100]: (A + A.T) / 2  # addition terme-à-terme
```

```
Out[100]: array([[ 0,  5, 11, 16],
 [ 5, 11, 16, 22],
 [11, 16, 22, 27],
 [16, 22, 27, 33]])
```

```
In [101]: print(np.trace(A))  # trace de A
```

Algèbre linéaire et réduction

Algèbre linéaire Réduction

Multiplications de matrices

```
In [102]: print(A.shape)
```

```
(4, 4)
```

```
In [103]: print(A * A)  # multiplication élément par élément
```

```
[[  0   1   4   9]
 [ 100  121  144  169]
 [ 400  441  484  529]
 [ 900  961 1024 1089]]
```

```
In [104]: print(np.dot(A, A))  # multiplication de matrices
```

```
[[ 140  146  152  158]
 [ 740  786  832  878]
 [1340 1426 1512 1598]
 [1940 2066 2192 2318]]
```

Algèbre linéaire et réduction

Algèbre linéaire Réduction

Multiplication matrice vecteur

```
In [105]: print(A)
          print(np.dot(A, v))
```

```
[[ 0  1  2  3]
 [10 11 12 13]
 [20 21 22 23]
 [30 31 32 33]]
```

```
-----
-----
ValueError                                Traceback (most recent call last)
<ipython-input-105-3bef7781e90d> in <module>()
      1 print(A)
----> 2 print(np.dot(A, v))

ValueError: objects are not aligned
```

```
In [106]: np.dot(A, v[:4])  # OK
```

```
Out[106]: array([ 14,  74, 134, 194])
```

Algèbre linéaire et réduction

Algèbre linéaire Réduction

Inversion de matrice

```
In [107]: B = random.standard_normal((3, 3))  
          Binv = np.linalg.inv(B)  
          print(Binv)  
  
[[-0.17547656  1.11964367  1.06455073]  
 [-0.57835448 -0.53209909  0.71979545]  
 [-0.61317879 -0.23518517 -1.82875743]]
```

```
In [108]: print(np.dot(B, Binv))  
  
[[ 1.00000000e+00  8.32667268e-17  0.00000000e+00]  
 [ 0.00000000e+00  1.00000000e+00  0.00000000e+00]  
 [ 0.00000000e+00  1.38777878e-17  1.00000000e+00]]
```

- Calcul de vecteurs/valeurs propres `np.linalg.eig`
- Résolution de système linéaire `np.linalg.solve`

Algèbre linéaire et réduction

Algèbre linéaire
Réduction

Opérations de réduction (ou d'aggrégation)

La performance des programmes écrits en Python/Numpy dépend de la capacité à vectoriser les calculs (les écrire comme des opérations sur des vecteurs/matrices) en évitant au maximum les boucles `for/while`

Algèbre linéaire et réduction

Algèbre linéaire Réduction

La somme : np.sum

```
In [109]: print(A)
```

```
[[ 0  1  2  3]
 [10 11 12 13]
 [20 21 22 23]
 [30 31 32 33]]
```

```
In [110]: np.sum(A)  # somme de tous les éléments
```

```
Out[110]: 264
```

```
In [111]: np.sum(A[1, :])  # somme de la 2ième ligne
```

```
Out[111]: 46
```

```
In [112]: np.sum(A, axis=1)  # somme de toutes les lignes
```

```
Out[112]: array([ 6, 46, 86, 126])
```

```
In [113]: np.sum(A, axis=0)  # somme de toutes les colonnes
```

```
Out[113]: array([60, 64, 68, 72])
```

Algèbre linéaire et réduction

Algèbre linéaire Réduction

La moyenne : np.mean

```
In [114]: print(A)
```

```
[[ 0  1  2  3]
 [10 11 12 13]
 [20 21 22 23]
 [30 31 32 33]]
```

```
In [115]: np.mean(A)  # moyenne de tous les éléments
```

```
Out[115]: 16.5
```

```
In [116]: np.mean(A[1, :])  # moyenne de la 2ième ligne
```

```
Out[116]: 11.5
```

```
In [117]: np.mean(A, axis=1)  # moyenne de toutes les lignes
```

```
Out[117]: array([ 1.5, 11.5, 21.5, 31.5])
```

```
In [118]: np.mean(A, axis=0)  # moyenne de toutes les colonnes
```

```
Out[118]: array([ 15., 16., 17., 18.])
```


Algèbre linéaire et réduction

Algèbre linéaire Réduction

D'autres fonctions de réduction existent: `np.var` pour la variance, `np.std` pour l'écart type, `np.min` pour le minimum, `np.max` pour le maximum, `np.prod` pour le produit, `np.cumsum` pour la somme cumulée ou encore `np.cumprod` pour le produit cumulé.

```
In [119]: print(A)
```

```
[[ 0  1  2  3]
 [10 11 12 13]
 [20 21 22 23]
 [30 31 32 33]]
```

```
In [120]: np.max(A, axis=0)  # maximum de toutes les colonnes
```

```
Out[120]: array([30, 31, 32, 33])
```