

# Calcul Scientifique

## Cours 1: Représentation des données

Alexis Lechervy



# Sommaire

- 1 Présentation du cours
- 2 Représentation des données pour le calcul scientifique

# Présentation du cours de Calcul Scientifique

## Objectifs

- Acquérir une méthodologie de travail pour **modéliser** et **résoudre** à l'aide de l'informatique des problèmes nécessitant la mise en place de calculs scientifiques,
- Savoir **modéliser les données**, les éléments d'un **problème concret**, à l'aide d'outils à la fois **mathématiques et informatiques**,
- **Mettre en œuvre des solutions** mathématiques et algorithmique apportant une solution numérique à ces problèmes,
- Maîtriser la vectorisation d'un problème afin de faire de la **programmation par tableau (array programming)** pour notamment bénéficier des accélérations matérielles des processeurs vectoriels et des GPU.

## Organisation

- 10 CM de 1h15 le mercredi 11h-12h15,
- 5 TP de 2h30 une semaine sur deux,
- 1/2 Note de CC sur les TP + 1/2 Note de CT 1h.
- Ressources : <https://ecampus.unicaen.fr/course/view.php?id=14888>

# Sommaire

## 1 Présentation du cours

## 2 Représentation des données pour le calcul scientifique

- Introduction
- Les vecteurs
- Les matrices
- Les tenseurs

# Outils informatiques utilisés dans ce cours

## Langage de Programmation

- **Python 3** : <https://www.python.org/>
- Documentation : <https://docs.python.org/3/>

## Bibliothèque de calcul

- **Numpy** : <http://www.numpy.org/>
- Documentation : <https://docs.scipy.org/doc/numpy/index.html>
- Numpy permet la **manipulation de données sous forme de tableau** et met à disposition des outils mathématiques pour les manipuler.
- Déclaration en début de fichier : `import numpy as np`

## Bibliothèque d'affichage

- **Matplotlib** : <https://matplotlib.org/>
- **Matplotlib** est une librairie *open source* python permettant de visualiser des données sous formes de graphiques.
- Déclaration en début de fichier : `import matplotlib.pyplot as plt`

# Les vecteurs

## Les vecteurs en informatique

Un vecteur est un tableau de nombre (à une dimension). Il permet de regrouper sous une même appellation une liste de nombre.

## Création de vecteur en numpy

- Vecteur  $v$  de 10 zéros :  $v = \text{np.zeros}(10)$
- Vecteur  $v$  de 10 un :  $v = \text{np.ones}(10)$
- Vecteur  $v$  des nombres entiers de 3 à 10 compris :  $v = \text{np.arange}(3,11)$
- Vecteur  $v$  de 10 valeurs comprises entre 0 et 1 :  $v = \text{np.linspace}(0,1,10)$
- Vecteur  $v$  de valeur  $[42,13,38,51]$  :  $v = \text{np.array}([42,13,38,51])$

## Lien avec les listes python

- Transformer une liste  $l$  en vecteur :  $v = \text{np.array}(l)$
- Transformer un vecteur  $v$  en liste python :  $l = \text{list}(v)$
- Attention les opérations mathématiques ne marche pas sur les listes python.

# Opérations sur les vecteurs

## Dimension d'un vecteur $v$

- $\text{len}(v)$  : Retourne le nombre d'élément dans  $v$ .
- $v.\text{shape}$  : Retourne une liste d'un élément, contenant la dimension du vecteur  $v$ .

## Opération arithmétique entre un nombre et un vecteur

**Règle générale :** L'opération arithmétique est appliquée entre le nombre et chaque valeur du vecteur. Cela correspond à l'opération mathématique classique correspondante.

## Exemples

- Additionner 2 à toutes les dimensions du vecteur :  $v2 = v + 2$
- Multiplier 2 à toutes les dimensions du vecteur :  $v2 = 2*v$
- Diviser 2 à toutes les dimensions du vecteur :  $v2 = v/2$

# Opérations sur les vecteurs

## Opération arithmétique entre deux vecteurs

**Règle générale :** L'opération arithmétique est appliquée terme à terme entre les valeurs des deux vecteurs. Attention cela ne correspond pas toujours aux notations mathématiques.

### Exemples

- Additionner les valeurs des vecteurs  $v1$  et  $v2$  :  $v3 = v1 + v2$
- Multiplier les valeurs des vecteurs  $v1$  et  $v2$  :  $v3 = v1 * v2$
- Diviser les valeurs des vecteurs  $v1$  et  $v2$  :  $v3 = v1 / v2$

## Fonction mathématique vectorisées (s'appliquant à toutes les valeurs d'un vecteur)

Liste des fonctions disponibles : <https://docs.scipy.org/doc/numpy/reference/routines.math.html>

- `np.sin`, `np.cos`, `np.tan`, `np.arcsin` ...
- `np.sinh`, `np.cosh`, `np.tanh`, `np.arcsinh` ...
- `np.exp`, `np.log`, `np.log10`, ...
- `np.sqrt`, `np.fabs`, `np.sign`, `np.square` ...

Exemple : `v2 = np.sin(v)`

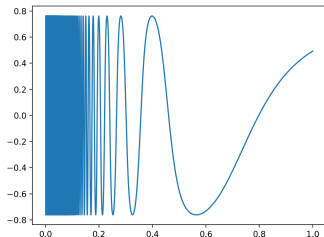


# Exemple d'application

## Tracer une fonction donnée

- Fonction à tracer :  
 $f(x) = \tanh(\cos(\frac{1}{x^2}))$
- Intervalle : Entre 0.001 et 1
- Code :

```
x = np.arange(1e-3,1,1e-6)
fx = np.tanh(np.cos(1/x**2))
plt.plot(x,fx)
plt.show()
```



⇒ L'utilisation de vecteur permet d'effectuer le calcul de la fonction en même temps sur toutes les valeurs du vecteur sans utiliser de boucle for.

# Opérations sur les vecteurs

## Les fonctions d'agrégation

**Principe :** Agrège les valeurs d'un vecteur en une seule valeur.

**Remarque :** Si les valeurs sont des boolean,  $\text{True} \iff 1$  et  $\text{False} \iff 0$ .

**Exemples :**

- `np.sum` , `np.prod`, `np.mean`, `np.std`,`np.var` ...
- `np.min`, `np.max`, `np.argmax`, `np.argmin`,...
- `np.count_nonzero` , `np.all`, `np.any`,...

## Opérations sur des ensembles

- `np.sort`, `np.argsort`, `np.unique`, `np.cumsum`, `np.cumprod` ...

## Opération booléenne sur les valeurs d'un vecteur

- Test d'égalité : `x == 1`
- Test d'inégalité : `x > 1`
- Comparaison des valeurs de vecteur de même taille : `x==y`

# Accès aux valeurs d'un vecteur

## Indexation

- $a[i]$  permet d'accéder à la  $i^{\text{ème}}$  valeur.
- L'indexation commence par la valeur 0.
- $a[-i]$  permet d'accéder à la  $i^{\text{ème}}$  valeur en partant de la fin.
- la dernière valeur est à l'indexe -1.

## Slicing

- $a[m:n]$  permet de récupérer un sous-vecteur des valeurs comprises entre l'indice  $m$  et l'indice  $n$ .
- $a[m:n:p]$  permet de récupérer un sous-vecteur des valeurs comprises entre l'indice  $m$  et l'indice  $n$  en prenant une valeur tout les  $p$  valeurs.

# Accès aux valeurs d'un vecteur

## Slicing avec des vecteurs

- Soit  $v$  un vecteur d'entier,  $a[v]$  permet de récupérer dans les valeurs de  $a$  aux indices donnés par  $v$ .  $v$  peut contenir plusieurs fois le même indice.
- Soit  $v$  un vecteur de boolean de même dimension que  $a$ ,  $a[v]$  permet de récupérer les valeurs dans  $a$  au indice où  $v$  est vrai.

## Exemple d'utilisation

```
x=np.array([5,4,3,2,1,0])
```

- $x[[1,1,0,2]] \rightarrow \text{array}([4, 4, 5, 3])$
- $x[x>2] \rightarrow \text{array}([3, 4, 5])$

## Exemple d'utilisation avancé

Soit  $x$  un vecteur contenant les notes d'examen de la promo.

- Compter le nombre d'étudiant au-dessus de la moyenne de la classe :  
`np.count_nonzero(x>np.mean(x))`

# Les matrices

## Les matrices en informatique

Une matrice est un tableau de nombre en 2D. On peut par exemple les utilisés pour stocker des images.

## Création d'une matrice en numpy

- Matrice m de 10x5 zéros :  $m = \text{np.zeros}((10,5))$
- Matrice m de 10x5 un :  $m = \text{np.ones}((10,5))$
- Matrice identité m de taille 11 deux :  $m = \text{np.eye}(11)$
- Matrice m de valeur  $\begin{bmatrix} 31 & 29 \\ 16 & -1 \end{bmatrix}$  :  $m = \text{np.array}([[31,29],[16,-1]])$

## Attention aux vecteurs

Attention une matrice a une seule dimension est différent d'un objet vecteur sous numpy.

Convertir une matrice 1D en vecteur :

$m[:,0]$  ou  $m[0,:]$

Convertir un vecteur en matrice 1D :

$v[:,\text{np.newaxis}]$  ou  $v[\text{np.newaxis},:]$

$v[:,\text{None}]$  ou  $v[\text{None},:]$

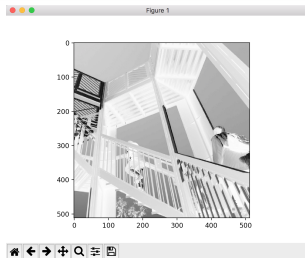
# Une image vu comme une matrice

## Lecture d'une image

- Utilisation de la librairie matplotlib :
- `import matplotlib.pyplot as plt`  
`m = plt.imread('mon_image.png')`

## Affichage d'une matrice comme une image noir et blanc

```
plt.imshow(m, cmap='gray')  
plt.show()
```



# Opérations sur les matrices

## Dimension d'une matrice $v$

- `m.size` : Retourne le total d'élément dans la matrice  $m$ .
- `len(m)` : Retourne le nombre de lignes de la matrice  $m$ .
- `m.shape` : Retourne une liste de deux éléments, contenant les dimensions de la matrice  $m$ .

## Opération arithmétique entre un nombre et une matrice

**Règle générale** : L'opération arithmétique est appliquée entre le nombre et chaque valeur de la matrice. Cela correspond à l'opération mathématique classique.

- Additionner 2 à toutes les dimensions de la matrice :  $m2 = m + 2$
- Multiplier 2 à toutes les dimensions de la matrice :  $m2 = 2 * m$
- Diviser 2 à toutes les dimensions de la matrice :  $m2 = m / 2$

# Opérations sur les matrices

## Opération arithmétique entre deux matrices

**Règle générale :** L'opération arithmétique est appliqué entre les valeurs des deux matrices. Attention, cela ne correspond pas toujours aux notations mathématiques.

- Additionner des valeurs de deux matrices :  $m3 = m1 + m2$
- Multiplier des valeurs de deux matrices :  $m3 = m1 * m2$
- Diviser les valeurs de deux matrices :  $m3 = m1 / m2$

## Fonction mathématiques vectorisées sur matrice

Liste des fonctions disponibles : <https://docs.scipy.org/doc/numpy/reference/routines.math.html>

- `np.sin`, `np.cos`, `np.tan`, `np.arcsin`, `np.arccos` ...
- `np.sinh`, `np.cosh`, `np.tanh`, `np.arcsinh`, `np.arccosh` ...
- `np.exp`, `np.log`, `np.log10`, ...
- `np.sqrt`, `np.fabs`, `np.sign`, `np.square` ...

Exemple : `m2 = np.cos(m)`



# Opérations sur les matrices

## Les fonctions d'agrégation

**Principe :** Agrège les valeurs d'une matrice en une valeur (par défaut) ou un vecteur (en précisant un axe d'agrégation).



- `np.sum` , `np.prod`, `np.mean`, `np.std`, `np.var` ...
- `np.min`, `np.max`, `np.argmax`, `np.argmin`, ...
- `np.count_nonzero`, `np.all`, `np.any`...

## Opération sur un axe donné

Exemple : `np.min(m,axis=0)`

# Accès aux valeurs d'une matrice

## Indexation

- $m[i,j]$  permet d'accéder à la valeur sur la  $i^{\text{ème}}$  ligne et la  $j^{\text{ème}}$  colonne.
- $m[i]$  ou  $m[i, :]$  permet d'accéder aux valeurs sur la  $i^{\text{ème}}$  ligne. La valeur retournée est un vecteur et non une matrice ligne.
- $m[:,j]$  permet d'accéder aux valeurs sur la  $j^{\text{ème}}$  colonne. La valeur retournée est un vecteur et non une matrice colonne.
- Test d'inégalité :  $x > 1$
- Comparaison de matrice de même taille :  $x==y$
- Sélection des valeurs d'une matrice selon une valeur de vérité :  $x[x>2]$
- Sélection des valeurs d'un axe selon un vecteur de boolean :  $x[v \neq 0, :]$  ;
- Sélection des valeurs d'une matrice uniquement sur certaine ligne  $x[v, :]$ .

# Accès aux valeurs d'une matrice

## Slicing avec des vecteurs

```
m = np.array([[0,1,2],[3,4,5]])
```

- `m[np.array([1,0]), :]` récupère les lignes 1 et 0 de `m`.
- `m[:,np.array([1,0])]` récupère les colonnes 1 et 0 de `m`.
- `m[np.array([0,1]),np.array([2,1])]` récupère les valeurs aux coordonnées (0,2) et (1,1)
- `m[np.array([True,False]), :]` récupère la première ligne. Doit être de la même taille que le nombre de ligne.
- `m[:,np.array([True,True,False])]` récupère les deux premières colonnes. Doit être de la même taille que le nombre de colonne.
- `m[np.array([True,False]),np.array([True,True,False])]` récupère les colonnes 0 et 1 de la ligne 0.

# Exemple d'utilisation

Binarisation d'une image noir et blanc.

On souhaite avoir une image où :

- Tous les pixels  $> 128$  sont à 255,
- Tous les pixels  $\leq 128$  sont à 0,

## Code

```
im[im>128] = 255
```

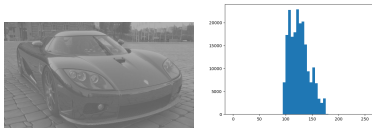
```
im[im<=128] = 0
```

## Résultat Avant/Après



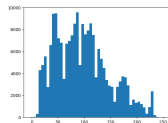
# Exemple d'utilisation : Étirement d'histogramme

Avant



`np.min(x) -> 93` `np.max(x) -> 185`

Après



`np.min(x) -> 0` `np.max(x) -> 255`

Code

```
x2=255*(x-np.min(x))/(np.max(x)-np.min(x))
```

# Les tenseurs

## Les tenseurs en informatique

Un tenseur peut être vu comme un tableau de nombre en n-dimension. On peut par exemple les utiliser pour stocker des images couleurs ou pour stocker une base d'image de même dimension.

## Création d'un tenseur en numpy

- Tenseur  $t$  de  $10 \times 5 \times 3$  zéros :  $t = \text{np.zeros}((10,5,3))$
- Tenseur  $s$  de  $10 \times 5 \times 3 \times 9$  uns :  $m = \text{np.ones}((10,5,3,9))$
- Tenseur  $t$  composé des valeurs  $\begin{bmatrix} 31 & 29 \\ 16 & -1 \end{bmatrix}$ ,  $\begin{bmatrix} 13 & 92 \\ 61 & 1 \end{bmatrix}$  :  
 $m = \text{np.array}([[[[31,29],[16,-1]], [[13,92],[61,1]]]])$

## Attention aux dimensions

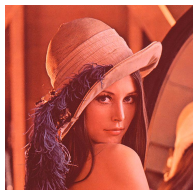
Un tenseur de dimension  $(10,5,2)$  se comportera différemment du tenseur de dimension  $(10,5,2,1)$ .

# Exemple d'utilisation : Augmenter le rouge d'une image

Avant



Après



## Code

```
import numpy as np
import imageio
im=np.array(imageio.imread('lena.jpg'),dtype='uint16') # lecture de l'image
im[:, :,0] += 50 # rougir
im[im>255] = 255 # seuillage
```