MANIPULATION DE FICHIERS

B. Landelle

Table des matières

Dans tout ce qui suit, on importera les bibliothèques `numpy` et `matplotlib.pyplot` avec les alias `np` et `plt` :

```
import numpy as np, matplotlib.pyplot as plt
```

I Manipulation de fichiers

1 Les instructions

Pour manipuler les fichiers, on utilise :
- l'instruction `open` pour l'ouverture ou la création ;
- la méthode `close` pour la fermeture ;
- la méthode `read` pour la lecture intégrale ;
- la méthode `readline` pour lire la ligne courante ;
- la méthode `write` pour l'écriture.

Pour la lecture et l'écriture dans un fichier, on recommande de travailler sur des fichiers `*.csv` [1] avec le caractère `';'` comme séparateur. Pour lire dans un fichier `file.csv`, on utilise :

```
data=open('file.csv')   # ouverture du fichier dans la variable data
contenu=data.read()     # contenu reçoit l'intégralité de data, format str
ligne=data.readline()   # ligne reçoit la ligne courante, format str
data.close()            # fermeture du fichier
```

La lecture de l'intégralité du fichier positionne le curseur de lecture en fin de fichier. La lecture d'une ligne du fichier positionne le curseur sur la ligne suivante ou la fin du ficher le cas échéant. Pour écrire dans un fichier, on l'ouvre avec l'option d'écriture `'w'` et on utilise :

```
data=open('file.csv','w') # option 'w'=write pour écriture
data.write('blabla\n')    # écriture d'une ligne 'blabla' dans data
data.close()
```

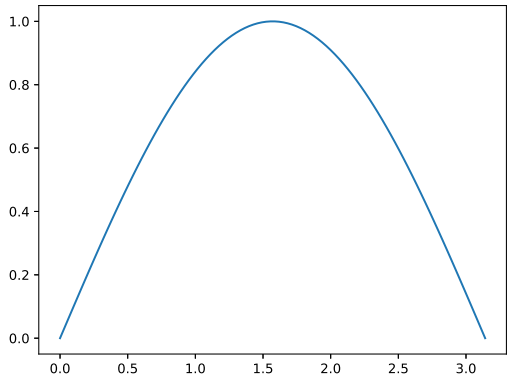
Le caractère spécial `'\n'` correspond à un saut de ligne.

2 Lecture dans un fichier

Le fichier `pts.csv` présent dans le répertoire de travail contient une suite de coordonnées de points que l'on souhaite tracer. Les points sont stockés ligne par ligne et le caractère ';' sépare abscisse et ordonnée.

1. Le format `csv` pour *comma-separated values* est un format de texte ouvert représentant des données tabulaires sous formes de valeurs séparées par des virgules ou points-virgules.

0.0	0.0
0.0317332591272	0.0317279334981
0.0634665182543	0.0634239196566
0.0951997773815	0.0950560433042
0.126933036509	0.126592453574
0.158666295636	0.158001395973
...	...



On peut parcourir directement le fichier ligne par ligne avec une simple boucle `for`. Chaque ligne est lue comme chaîne de caractères.

```
points=open('pts.csv')
tx,ty=[],[]
for coords in points:           # lecture ligne par ligne
    xy=coords.split(";")        # coupe la chaine "x;y" en ("x","y")
    tx.append(float(xy[0]))
    ty.append(float(xy[1]))
points.close()
plt.plot(tx,ty);plt.show()
```

Ainsi, lors du deuxième passage dans la boucle `for`, on réalise les affectations suivantes :

```
coords ← '0.0317332591272;0.0317279334981\n'
xy ← ['0.0317332591272', '0.0317279334981\n']
tx ← tx + [0.0317332591272]
ty ← ty + [0.0317279334981]
```

3 Écriture dans un fichier

Pour sauvegarder des données calculées sous python dans un fichier, on ouvre celui-ci avec l'option `'w'` pour autoriser l'écriture. Ainsi, le fichier `pts.csv` considéré précédemment a été généré par le code suivant :

```
fichier=open('pts.csv','w')    # option 'w'=write pour écriture
tt=np.linspace(0,np.pi,100)
ts=np.sin(tt)
for k in range(len(tt)):
    fichier.write(str(tt[k])+";"+str(ts[k])+"\n")   # ';' comme séparateur
fichier.close()
```

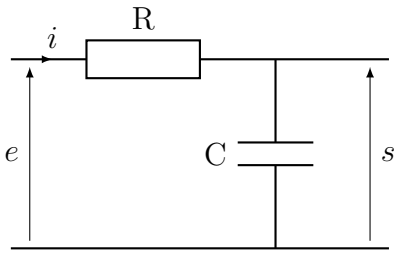
Son exécution réalise l'écriture des lignes suivantes dans le fichier `pts.csv` :

```
0.0;0.0
0.0317332591272;0.0317279334981
0.0634665182543;0.0634239196566
...
```

II Filtrage d'un signal

1 Réalisation d'un filtre passe-bas

On s'intéresse à l'implémentation d'un filtre analogique passe-bas réalisé par un circuit RC.



Exercice : Déterminer l'équation différentielle reliant la sortie s à l'entrée e.

Corrigé : On trouve $\boxed{\mathrm{RC}\dfrac{\mathrm{d}s}{\mathrm{d}t} + s = e}$

2 Extraction des données

On souhaite extraire du fichier `data.csv` les données organisées sur deux colonnes, la première colonne contenant les instants de mesure et la deuxième contenant les mesures bruitées (bruit haute fréquence) d'un signal (l'entrée e). Sur chaque ligne, les données sont séparées par le caractère `","`. On pourra utiliser la méthode `split` qui permet de séparer une chaîne de caractère selon le schéma `texte.split(",")` où `texte` désigne la chaîne de caractères.

Exercice : Écrire des instructions afin de construire les listes d'instants et de mesures bruitées qu'on nommera respectivement `tt` et `te` à partir des données du fichier `data.csv`.

Corrigé : On saisit :

```
fichier=open("data.csv")
tt,te=[],[]
for ligne in fichier:
    aux=ligne.split(",")
    tt.append(float(aux[0]))
    te.append(float(aux[1]))
fichier.close()
```

3 Débruitage

On souhaite enfin utiliser le filtre passe-bas présenté précédemment pour réaliser un débruitage des mesures bruitées extraites du fichier `data.csv`.

Exercice : Écrire une fonction `debruit(tau,e,t)` d'arguments $\tau = \mathrm{RC}$ avec R et C respectivement résistance et capacité du filtre, `e` la liste des valeurs bruitées du signal, `t` la liste des instants de mesure et qui renvoie la liste des valeurs de sortie du filtre RC. On implémentera un schéma d'Euler explicite.

Corrigé : On saisit :

```
def debruit(tau,e,t):
    """debruit(tau:float,e:list,t:list)->list
    Débruitage du signal d'entrée e par un filtre tau=RC"""
    s=[e[0]]
    for k in range(1,len(t)):
        h=t[k]-t[k-1]
        s.append(s[k-1]+h*(e[k-1]-s[k-1])/tau)
    return x
```

On saisit :

```
plt.plot(tt,te)
ts=debruit(.2,te,tt)
plt.plot(tt,ts,'r',linewidth=2)
plt.grid();plt.show()

plt.plot(tt,te)
ts=debruit(.5,te,tt)
plt.plot(tt,ts,'r',linewidth=2)
plt.grid();plt.show()
```

On observe :

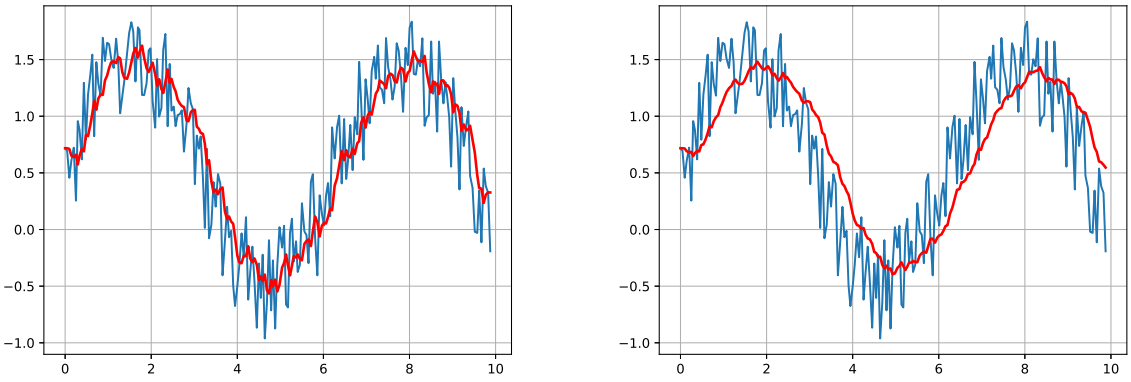


FIGURE 1 – Débruitage par filtre passe-bas RC pour différentes valeurs de τ

On constate clairement l'effet du filtre passe-bas : les perturbations hautes fréquences sont « lissées » et l'on « paye » l'effet du filtrage par un léger retard dans le temps.

III Manipulation d'images

1 Les instructions

Pour manipuler des images, on importera le module `Image` de la bibliothèque `PIL` :

```
from PIL import Image
```

On utilisera les fonctions suivantes :

```
def importer(image):
    """importer(image->str)->list
    Renvoie une liste de listes de points de l'image en niveaux de gris"""
    tab=np.array(Image.open(image).convert('L'))
    return [list(tab[k]) for k in range(len(tab))]

def afficher(tab):
    """afficher(tab->list)->None
    Réalise l'affichage de l'image
    codée par la liste de listes de points tab"""
    im=Image.fromarray(np.array(tab))
    im.show()
```

La fonction `importer(image)` d'argument un fichier d'image renvoie une liste de listes, chaque sous-liste représentant une ligne de points de l'image où les points sont des valeurs entre 0 et 255 qui désignent un niveau de gris. La fonction `afficher(tab)` d'argument une liste de listes représentant une image réalise l'affichage de celle-ci.

Si la configuration machine ne permet pas l'installation de la bibliothèque `PIL`, on peut se rabattre sur une version obsolète en utilisant :

```
def importer(image):
    tab=plt.imread(image)
    tab=(tab[:,:,0]+tab[:,:,1]+tab[:,:,2])/3
    return [list(tab[k]) for k in range(len(tab))]

def afficher(tab):
    plt.imshow(tab,cmap="gray")
    plt.show()
```

Avec le fichier `Auvergne.jpg` présent dans le répertoire de travail, on saisit :

```
>>> tab=importer('Auvergne.jpg')
>>> afficher(tab)
```

qui produit l'affichage

FIGURE 2 – Vue depuis le Puy de Sancy, Auvergne

Pour afficher l'image en négatif, on inverse le niveau de gris de chaque point en le passant d'une valeur x à sa valeur complémentaire $255 - x$.

```
def negatif(tab):
    """negatif(tab:list)->list
    Renvoie la liste de listes de points
    de l'image décrite par tab en négatif"""
    L,C=len(tab),len(tab[0])
    return [[255-tab[i][j] for j in range(C)] for i in range(L)]

res=negatif(tab)
afficher(res)
```

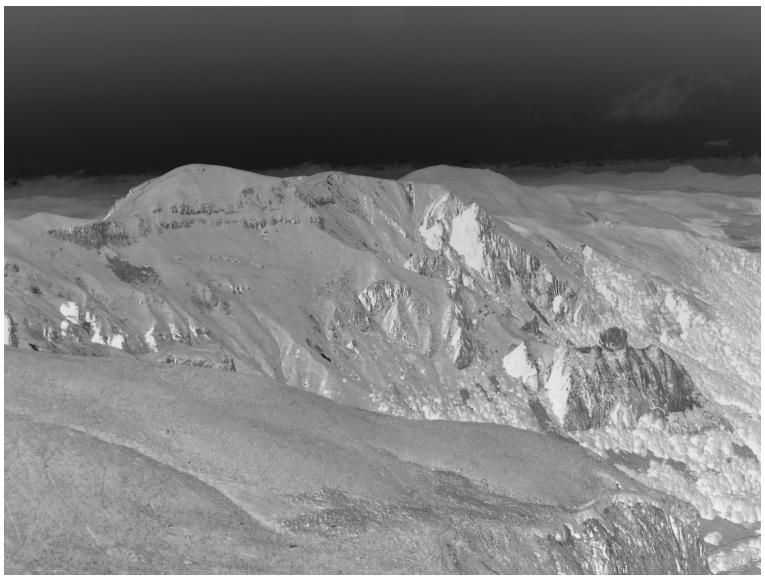


FIGURE 3 – Négatif de l'image

2 Changement de coordonnées

Pour appliquer des transformations géométriques à une image, on se propose de passer du système de repérage par ligne et colonne (i, j) à un système de coordonnées classiques (x, y). On note (i_0, j_0) les numéros de ligne et colonne d'un point central de l'image .

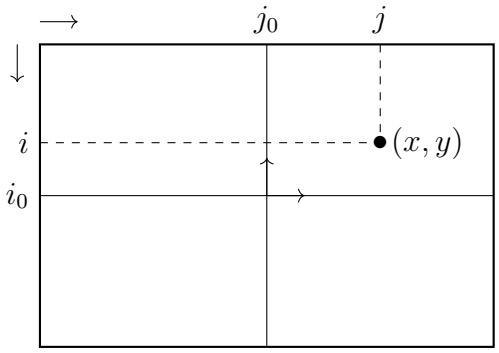


FIGURE 4 – Changement de coordonnées

Pour le passage $(i, j) \to (x, y)$, on a

$$\begin{cases} x = j - j_0 \\ y = i_0 - i \end{cases}$$

Comme on prévoit d'effectuer des calculs sur (x, y) dont le format est appelé à basculer en flottant, le passage $(x, y) \to (i, j)$ ne sera pas l'application réciproque du passage $(i, j) \to (x, y)$ mais tiendra compte du fait que les lignes et colonnes sont des entiers :

$$\begin{cases} i = \lfloor i_0 - y \rfloor \\ j = \lfloor j_0 + x \rfloor \end{cases}$$

3 Homothétie

Soit $\mu > 0$. Réaliser la dilatation ou rétraction d'une image du facteur μ consiste pour un point de l'image de coordonnées (x, y) à le transformer en $(\mu x, \mu y)$.

On note (X, Y) les coordonnées du point transformé et (I, J) ces numéros de ligne et colonne correspondants. Les nombres de ligne et de colonnes de l'image transformée sont multipliés par ce facteur μ et comme ce sont des entiers, on a respectivement $\lfloor \mu \mathrm{L} \rfloor$ lignes et $\lfloor \mu \mathrm{C} \rfloor$ colonnes dans la nouvelle image.

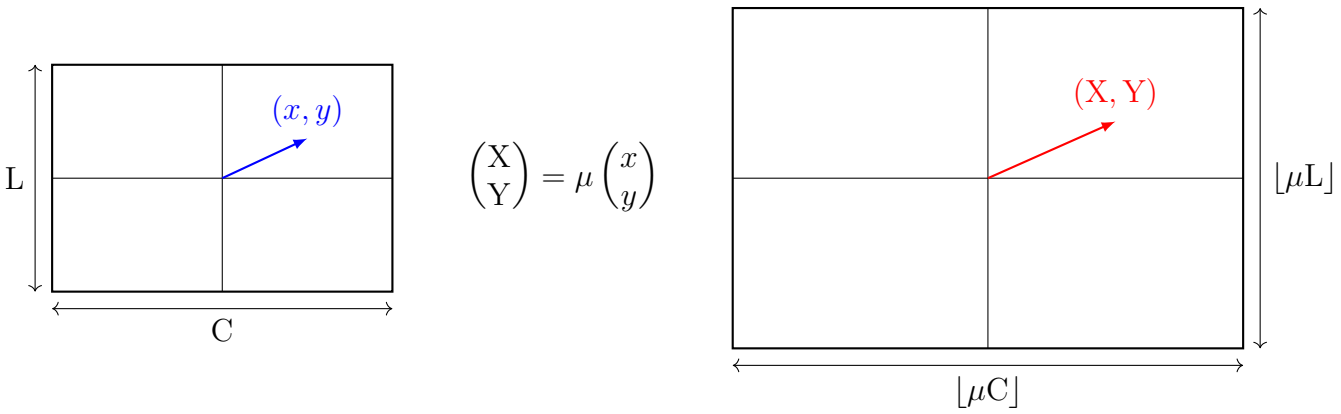


FIGURE 5 – Homothétie de rapport μ

Pour la construction de la nouvelle image, un point en ligne I colonne J dont les coordonnées sont (X, Y) correspond à un point (x, y) de l'image d'origine avec

$$x = \frac{1}{\mu} X \qquad y = \frac{1}{\mu} Y$$

de ligne i et colonne j, les relations entre (I, J) et (X, Y), et (x, y) et (i, j) étant décrites à la section précédente.

Pour la lisibilité de l'implémentation, on définit des fonctions locales `xy_ij` et `IJ_XY` qui réalisent les conversions ligne/colonne $\longleftrightarrow$ coordonnées.

```
def homothetie(tab,mu):
    """homothetie(tab:list,mu:float)->list
    Réalise l'homothétie de facteur mu sur l'image
    décrite par la liste de listes de points tab"""
    L,C=len(tab),len(tab[0])
    mL,mC=int(mu*L),int(mu*C)
    res=[[0]*mC for k in range(mL)]
    i0,j0=L//2,C//2
    I0,J0=mL//2,mC//2
    def xy_ij(x,y):
        return int(i0-y),int(j0+x)
    def IJ_XY(I,J):
        return J-J0,I0-I
    for I in range(mL):
        for J in range(mC):
            X,Y=IJ_XY(I,J)
            x,y=1/mu*X,1/mu*Y
            i,j=xy_ij(x,y)
            res[I][J]=tab[i][j]
    return res

res=homothetie(tab,.5)
afficher(res)
```

FIGURE 6 – Image réduite d'un facteur $\frac{1}{2}$

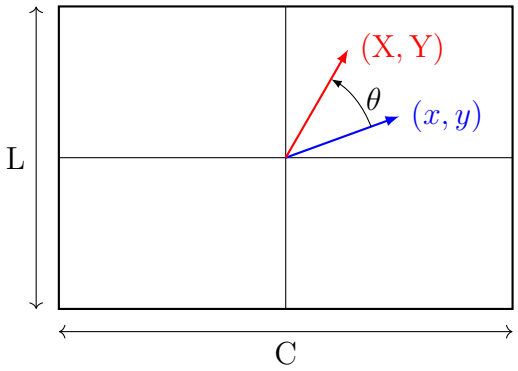
À l'affichage, on ne voit pas réellement de différence avec l'image d'origine mais comme le nombre de lignes et de colonne a été divisé par deux, on gagne donc un facteur quatre dans l'écriture en mémoire de cette image réduite quand elle est au format liste de listes.

4 Rotation

Soit θ réel. Réaliser la rotation d'une l'image d'un angle θ consiste pour un point de l'image de coordonnées (x, y) à le transformer en (X, Y) suivant la relation

$$\begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \end{pmatrix} = \mathrm{R}(\theta) \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{avec} \quad \mathrm{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

La matrice $\mathrm{R}(\theta)$ est la matrice de la rotation d'angle θ. Elle est inversible d'inverse $\mathrm{R}(-\theta)$.



$$\begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \end{pmatrix} = \mathrm{R}(\theta) \begin{pmatrix} x \\ y \end{pmatrix}$$

FIGURE 7 – Rotation d'angle θ

Pour la construction de la nouvelle image, on considère qu'elle a les mêmes dimensions que l'image d'origine. Si un point sort du cadre après rotation, il est « perdu ». Un point de l'image pivotée en ligne I colonne J dont les coordonnées sont (X, Y) correspond à un point (x, y) de l'image d'origine avec

$$\begin{pmatrix} x \\ y \end{pmatrix} = \mathrm{R}(-\theta) \begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \end{pmatrix}$$

autrement dit $\begin{cases} x = \cos(\theta)\mathrm{X} + \sin(\theta)\mathrm{Y} \\ y = -\sin(\theta)\mathrm{X} + \cos(\theta)\mathrm{Y} \end{cases}$

Pour la lisibilité de l'implémentation, on définit des fonctions locales `xy_ij` et `ij_xy` qui réalisent les conversions ligne/colonne $\longleftrightarrow$ coordonnées. On définit également une fonction `deborde` qui détermine si un point ligne `i` colonne `j` déborde du cadre de l'image ou pas.

```
def rotation(tab,a):
    """rotation(tab:list,a:float)->list
    Réalise la rotation d'angle a sur l'image
    décrite par la liste de listes de points tab"""
    L,C=len(tab),len(tab[0])
    i0,j0=L//2,C//2
    def ij_xy(i,j):
        return j-j0,i0-i
    def xy_ij(x,y):
        return int(i0-y),int(j0+x)
    def deborde(i,j):
        return i<0 or i>=L or j<0 or j>=C
    res=[[0]*C for k in range(L)]
    ca,sa=np.cos(a),np.sin(a)
    for I in range(L):
        for J in range(C):
            X,Y=ij_xy(I,J)
            x,y=ca*X+sa*Y,-sa*X+ca*Y
            i,j=xy_ij(x,y)
            if not deborde(i,j):
                res[I][J]=tab[i][j]
    return res

res=rotation(tab,np.pi/6)
afficher(res)
```

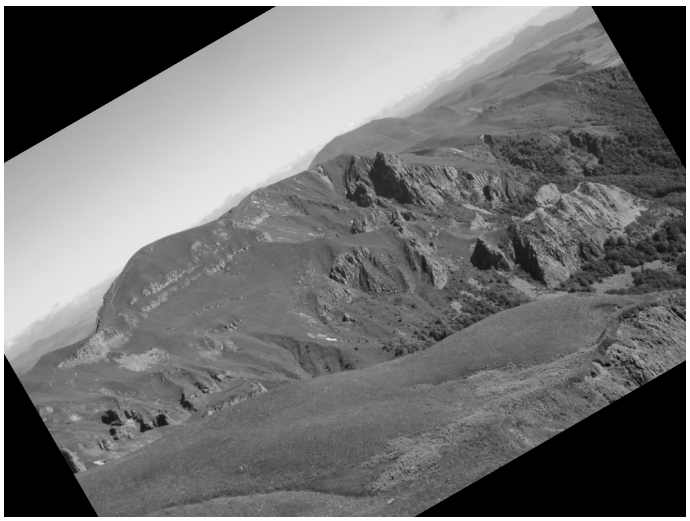


FIGURE 8 – Image pivotée d'un angle de $\frac{\pi}{6}$