Partie 6

Vade-mecum Python

Ce vade-mecum n'est pas exhaustif : n'hésitez pas à y ajouter les fonctions supplémentaires que vous trouvez intéressantes.

6.1 Introduction à Python et au calcul matriciel

6.1.1 Prise en main

- `#` : délimiteur de commentaires.
- `print()` : pour d'afficher la valeur des variables.
- `help(commande)` : affiche l'aide de la commande.
- `s.dtype` : affiche le type des données contenues dans `s` (*e.g.*, `numpy.uint8`).
- `%whos` : affiche la liste des variables de l'espace de travail.
- `time.time()` : associé à des variables, cela permet de connaître le temps d'exécution d'un programme.
- `abs(z)` : module de `z`.
- `numpy.angle(z)` : argument/phase de `z` en radians.
- `z.conjugate()` : complexe conjugué de `z`.
- `numpy.rad2deg(a)` : transforme en degrés la quantité `a` exprimée en radians.
- `numpy.degrees(a)` : transforme en degrés la quantité `a` exprimée en radians.
- `numpy.deg2rad(a)` : transforme en radians la quantité `a` exprimée en degrés.
- `numpy.radians(a)` : transforme en radians la quantité `a` exprimée en degrés.

6.1.2 Vecteurs et matrices

- `t = numpy.arange(1, 10.1, 0.1)` : définit un vecteur `t` composé de nombres entre 1 et 10 avec un pas de 0,1 (par défaut, la valeur du pas est 1).
- `t = numpy.linspace(1, 10, 100)` : définit un vecteur `t` composé de 100 nombres répartis uniformément entre 1 et 10.
- `a = numpy.array([1, 3, 5, 2, -1])` : définit un vecteur ligne `a` contenant les éléments 1, 3, 5, 2 et -1.
- `a = numpy.array([[1], [3], [5], [2], [-1]])` : définit un vecteur colonne `a` contenant les éléments 1, 3, 5, 2 et -1.
- `m = numpy.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])` : définit la matrice `m` de dimension 3×3 :
$$m = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$
- `c = numpy.array([a, b])` : concatène des vecteurs `a` et `b`.
- `c = a.flatten()` : retourne l'array `a` « applati » selon une dimension.
- `t[1]` : permet d'accéder au deuxième élément de l'array `t`.

- `t[-1]` : permet d'accéder au dernier élément de l'array `t`.
- `t[:5]` : permet d'accéder aux cinq premiers éléments de l'array `t`.
- `t[1:4]` : renvoie du deuxième au quatrième élément de l'array `t`.
- `m[1, 2]` : permet d'accéder à l'élément de la seconde ligne et de la troisième colonne de la matrice `m`.
- `m[1, :]` : renvoie la deuxième ligne de la matrice `m`.
- `m[:, 2]` : renvoie la troisième colonne de la matrice `m` sous forme d'un vecteur ligne.
- `len(m)` : renvoie le nombre d'éléments de l'array `m`.
- `numpy.size(m)` : renvoie le nombre d'éléments de l'array `m`, possiblement selon un axe donné.
- `numpy.shape(m)` : renvoie les dimensions de l'array `m`, c'est-à-dire son nombre de lignes et son nombre de colonnes.
- `numpy.min(m)`/`numpy.max(m)` : renvoie la valeur minimum/maximum de l'array `m`.
- `numpy.mean(m)` : renvoie la valeur moyenne des entrées de l'array `m`.

Importation de signaux audio

- `Fs, s = scipy.io.wavfile.read('filename.wav')` : ouvre le fichier `filename.wav` et stocke les échantillons dans le vecteur `s`. `Fs` contient la fréquence d'échantillonnage du signal.
- `scipy.io.wavfile.write('filename.wav', Fs, data)` : enregistre les données du vecteur `data` dans le fichier `filename.wav`. `Fs` contient la fréquence d'échantillonnage du signal.
- `sounddevice.play(s, Fs)` : joue le signal `s` avec une fréquence d'échantillonnage `Fs`.
- `sounddevice.wait()` : attend que le son joué soit terminé avant d'exécuter la suite du programme.

6.1.3 Programmation avec Python

- `for i in range(1, 11): <instructions>` : boucle `for` (remarque : attention aux performances par rapport au calcul matriciel et attention à l'indentation des instructions).
- `while condition: <instructions>` : boucle `while`.
- `if condition1: <instructions> elif condition2: <instructions> else: <instructions>` : condition.
- `<`, `<=`, `>`, `>=`, `==`, `!=` : opérateurs de comparaison.
- `and`, `or`, `not` : opérateurs logiques.

6.1.4 Éléments de calcul matriciel

- `numpy.zeros((m, n))` : envoie une matrice de dimension $m \times n$ composée de 0 partout.
- `numpy.ones((m, n))` : envoie une matrice de dimension $m \times n$ composée de 1 partout.

- `numpy.eye(m, n)` : envoie une matrice de dimension $m \times n$ composée de 0 à l'exception de la diagonale principale qui contient des 1.
- `A @ B` ou `numpy.matmul(A, B)` : renvoie le produit matriciel de A par B (fonctionne uniquement si le nombre de lignes de `A` est égal au nombre de colonnes de `B`).
- `A * B` : renvoie le produit terme par terme de A par B (fonctionne uniquement si `A` et `B` sont de mêmes dimensions).
- `A.T` : renvoie la transposée de la matrice `A`.
- `numpy.linalg.inv(A)` : renvoie la matrice inverse de la matrice `A` si elle existe.
- `numpy.linalg.matrix_power(A,2)` : renvoie le carré de la matrice `A` (donc `A @ A`).
- `numpy.square(A)` : renvoie une matrice dans laquelle chaque élément de `A` a été mis au carré (donc `A * A`).
- `numpy.random.rand(s1, s2)` : génère un array de dimension `s1`×`s2` peuplé de nombres aléatoires générés à partir d'une distribution uniforme sur $[0, 1[$.
- `numpy.random.randint(low, high, (s1, s2))` : génère un array de dimension `s1`×`s2` peuplé de nombres aléatoires générés à partir d'une distribution uniforme discrète sur `[low, high[`.
- `numpy.random.randn(s1, s2)` : génère un array de dimension `s1`×`s2` peuplé de nombres aléatoires générés à partir d'une distribution normale standard.
- `numpy.diag(A)` : renvoie les éléments de la diagonale de la matrice `A`.

6.2 Représentation des signaux

- `matplotlib.pyplot.figure()` : crée une nouvelle fenêtre/figure.
- `matplotlib.pyplot.show()` : afficher toutes les figures ouvertes.
- `matplotlib.pyplot.close('all')` : ferme toutes les fenêtres ouvertes.
- `matplotlib.pyplot.title('Titre')` : ajoute un titre au graphe.
- `matplotlib.pyplot.xlabel("Nom de l'axe x")` ou `matplotlib.pyplot.ylabel("Nom de l'axe y")` : ajoute un titre à l'axe correspondant.
- `matplotlib.pyplot.xlim((p, q))` : permet de limiter les valeurs affichées en abscisses entre `p` et `q` (à préciser avant le `matplotlib.pyplot.show()`).
- `matplotlib.pyplot.tight_layout()` : permet d'ajuster la disposition du graphique (à préciser avant le `matplotlib.pyplot.show()`).
- `matplotlib.pyplot.subplot(a, b, c)` : permet d'afficher plusieurs figures dans la même fenêtre. Les paramètres `a` et `b` définissent le nombre de figures en lignes et en colonnes et `c` spécifie la position du graphe créé par la commande.
- `fig, axs = matplotlib.pyplot.subplots(a, b)` : crée une figure avec plusieurs axes (un par sous-figure). Les paramètres `a` et `b` définissent le nombre d'axes en lignes et en colonnes. D'autres paramètres additionnels permettent par exemple de partager une même graduation d'axe pour toutes les sous-figures.

Signaux 1D

- `matplotlib.pyplot.plot(t, y)` : affiche le graphe du vecteur `y` en fonction du vecteur `t`.

- `matplotlib.pyplot.stem(x, y)` : permet de dessiner uniquement les points de coordonnées `(x, y)` sans les relier (à la manière d'un graphe discret).

Signaux 2D

- `matplotlib.pyplot.imshow(I, cmap='gray')` : affiche dans une figure l'image représentée par la matrice `I` en niveaux de gris. La carte des couleurs par défaut est `'viridis'`.
- `matplotlib.pyplot.set_cmap('map')` : permet de changer la carte des couleurs courante (valeurs possibles : `'viridis'`, `'jet'`, `'hot'`, `'cool'`, `'summer'`, `'hsv'`, `'gray'`, `'Pastel1'`, etc.).
- `matplotlib.pyplot.colorbar()` : affiche une barre représentant les niveaux de couleurs de la colormap de l'image représentée dans la figure courante.

6.3 Représentation fréquentielle des signaux

- `numpy.fft.fft(s)` : renvoie la transformée de Fourier discrète du signal `s`.
- `numpy.fft.fftfreq(len(t), 1/Fs)` : renvoie les fréquences auxquelles correspondent les valeurs de la FFT d'un signal à `len(t)` points et de fréquence d'échantillonnage `Fs`.
- `numpy.fft.ifft(X)` : calcule la transformée de Fourier inverse du spectre `X`.
- `b, a = scipy.signal.butter(n, Wn, 'ftype')` : calcule les coefficients d'un filtre de Butterworth d'ordre `n` avec une fréquence de coupure normalisée `Wn` ($W_n = \frac{F_c}{F_s/2}$, avec F_c la fréquence de coupure en Hz). `ftype` est le type du filtre (passe-bas par défaut, les valeurs spécifiables sont `'high'`, `'low'`, `'stop'` et `'pass'`).
- `scipy.signal.filtfilt(b, a, s)` : filtre le signal `s` avec un filtre de coefficients `a`, `b`.
- `f, t, Sxx = scipy.signal.spectrogram(s, Fs, window='hamming', nperseg=1024, noverlap=512)` : calcule le spectrogramme du signal `s`. `Sxx` est la matrice du spectrogramme de dimensions données par les longueurs des vecteurs `f` et `t`. `window` est la forme du fenêtrage temporel, `nperseg` est la taille des segments du signal sur lesquels sera calculée la FFT. `noverlap` donne le nombre d'échantillons de superposition entre deux positions successives de la fenêtre glissante (ici 50%). Finalement, `Fs` est la fréquence d'échantillonnage du signal. Remarque : on affiche le spectrogramme sous la forme d'un nuage de points colorés avec l'instruction `matplotlib.pyplot.pcolormesh(t, f, np.log10(Sxx))`. L'affichage logarithmique pour améliore la visualisation des valeurs faibles du spectrogramme.

6.4 Traitement d'images

6.4.1 Ouverture, affichage et types d'images

- `matplotlib.pyplot.imread('filename')` : ouvre le fichier `filename` (par exemple .tif, .jpg, .bmp, etc.) et renvoie soit une matrice bidimensionnelle (image en niveaux de gris) soit une matrice tridimensionnelle (image en couleurs).

- `I.copy()` : renvoie une copie de l'array `I`.
- `I.asType(np.float64)` : permet de convertir les données de `I` dans un autre type (ici vers des nombres à virgule flottante).
- `skimage.measure.profile_line(I, point_start, point_end)` : renvoie un tableau contenant les valeurs RGB des pixels sur une ligne allant de `point_start` à `point_end` dans l'image `I`.

6.4.2 Opérations arithmétiques sur une image

- `skimage.color.rgb2gray(I)` : convertit l'image RGB `I` en niveaux de gris en utilisant la formule `gray = 0.2125*R + 0.7154*G + 0.0721*B`.
- `skimage.exposure.equalize_hist(I)` : égalise l'histogramme de l'image `I` (*i.e.*, répartit les valeurs des pixels sur toute la gamme disponible pour améliorer le contraste) et renvoie l'image résultante.
- `skimage.exposure.rescale_intensity(I)` : ajuste l'intensité ou la carte de couleurs de l'image `I` et renvoie l'image résultante.
- `skimage.filters.sobel(I)` : renvoie une image contenant les contours de l'image `I` obtenus grâce au filtre de Sobel. D'autres filtres peuvent être utilisés pour détecter les contours, *e.g.*, `skimage.filters.prewitt(I)` ou `skimage.filters.roberts(I)`.
- `scipy.signal.convolve2d(I, h, mode='same')` : renvoie le produit de convolution des deux arrays bidimensionnels `I` et `h`. Le paramètre `mode='same'` fait en sorte que le résultat a la même taille que `I`.
- `scipy.signal.correlate2d(I, h, mode='same')` : renvoie la corrélation des deux arrays bidimensionnels `I` et `h`. Le paramètre `mode='same'` fait en sorte que le résultat a la même taille que `I`.
- `scipy.signal.medfilt2d(I, kernel_size=[M, N])` : renvoie l'image `I` modifiée suite à l'application d'un filtre médian dont le noyau est de taille `M`×`N` (avec `M` et `N` impairs). La taille du noyau par défaut est 3×3.

6.4.3 Opérations sur une image binaire

Binarisation et labellisation

- `I >= thresh` : transforme l'image `I` en image binaire avec un seuil `thresh`.
- `skimage.filters.threshold_otsu(I)` : renvoie un seuil calculé automatiquement sur base de l'image `I` et selon la méthode d'Otsu.
- `L, n = skimage.measure.label(I, return_num=True)` : identifie les objets (régions connectées) présents dans l'image binaire `I`. La matrice `L` a la même taile que `I` et contient des valeurs entières. Tous les pixels appartenant au $i^{\text{ème}}$ objet détecté ont la valeur i. La paramètre `return_num=True` permet de renvoyer également le nombre d'objets détectés.
- `skimage.segmentation.find_boundaries(L)` : renvoie une image booléenne basée sur l'image labellisée `L` et contenant `True` sur les frontières entre régions.
- `skimage.measure.regionprops(L)` : mesure une série de propriétés des objets présents dans une image labellisée `L`. La fonction renvoie un tableau contenant

autant de structures que d'objets. Chaque champ d'une structure correspond à une propriété (par exemple, `data[1].area` contient la surface du deuxième objet).
- `numpy.where(condition, x, y)` : renvoie `x` ou `y` dépendant si `condition` est `True` ou `False`. Les deux derniers arguments sont optionnels. Par exemple, `numpy.where(L==2, 1, 0)` renvoie une image binaire contenant uniquement le deuxième objet de l'image labellisée `L`.
- `numpy.isin(X, a)` : renvoie une matrice binaire de même taille que `X` contenant des valeurs `True` si la valeur correspondante de `X` est présente dans le array `a` et `False` dans le cas contraire.

Opérations morphologiques

- `skimage.morphology.diamond(s)` : crée un élément structurant pour les opérations d'érosion et de dilatation. Ici, il s'agit d'une forme en diamant de taille `s` (entier) contrôlant la taille du losange. D'autres formes sont possibles, *e.g.*, un disque avec `skimage.morphology.disk(s)` ou un carré avec `skimage.morphology.square(s)`.
- `skimage.morphology.erosion(I, SE)` : renvoie une nouvelle image sur base de l'image `I` (en niveaux de gris ou binaire) à laquelle l'opération d'érosion a été appliquée avec un élément structurant `SE`.
- `skimage.morphology.dilation(I, SE)` : renvoie une nouvelle image sur base de l'image `I` (en niveaux de gris ou binaire) à laquelle l'opération de dilatation a été appliquée avec un élément structurant `SE`.
- `skimage.morphology.remove_small_objects(I, min_size=10)` : renvoie une image binaire identique à l'image de départ `I` à l'exception des régions connectées plus petites que `min_size` pixels qui sont supprimées.
- `skimage.morphology.skeletonize(I)` : renvoie une image contenant le squelette de l'image binaire `I`.
- `scipy.ndimage.binary_fill_holes(I)` : remplit les trous de l'image binaire `I`.

6.5 Liens vers les documentations en ligne

Les documentations complètes se trouvent aux adresses suivantes :

- Matplotlib : `https://matplotlib.org/`
- NumPy : `https://numpy.org/`
- scikit-image : `https://scikit-image.org/`
- SciPy : `https://scipy.org/`
- sounddevice : `https://python-sounddevice.readthedocs.io/`
- time : `https://docs.python.org/3/library/time.html`
- OpenCV : `https://opencv.org/`