ENSTA BRETAGNE

Visualisation de données 3D

UE 1.1 - Analyse de données spatiales

La réalisation lors de la séance de TP des deux derniers items (1.3 et 2.2) de chaque exercice est facultative.

1. Visualisation de données cartographiques

On considère des données du modèle numérique altimétrique Litto3D® sur l'Île de Sein (Finistère).

Les données sont stockées dans le fichier `sein.npy` sous la forme d'une matrice contenant les coordonnées x (m) dans la première colonne, y (m) dans la seconde et l'altitude z (m) dans la troisième.

1. Visualisation du nuage de points : pour visualiser ces données, on utilise la fonction `scatter`[1] de `matplotlib`.
   (a) Afficher les données à l'aide de la fonction `scatter` ; chaque point défini par ses coordonnées (x, y) doit être représenté par un cercle de taille de 2 pt (option `s`) et coloré en fonction de son altitude z (option `c`).
   Le chargement du fichier et l'accès aux variables se réalise de la manière suivante :

   Python
   ```
   npy = np.load('sein.npy')
   x = npy[:,0]
   y = npy[:,1]
   z = npy[:,2]
   ```

   Il n'existe plus de commande permettant d'ajuster une échelle identique sur les axes x et y. Un moyen détourné est de le faire manuellement en fixant la même amplitude sur les axes des abscisses et des ordonnées avec les méthodes `set_xlim` et `set_ylim`.
   Pour une meilleure lisibilité il est conseillé d'ajouter l'option `edgecolors='none'` à la fonction `scatter`.
   (b) La coloration des données dépend de la carte de couleur choisies (`colormap`[2]). Appliquer la carte de couleur `gist_earth` (option `cmap`) pour la visualisation de ces données et contraindre les valeurs minimale et maximale à -15 et $+15$ (options `vmin` et `vmax`).
   (c) Compléter la figure en ajoutant une barre de couleur à l'aide de la fonction `colorbar`[3].

1. https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html
2. https://matplotlib.org/users/colormaps.html
3. https://matplotlib.org/api/_as_gen/matplotlib.pyplot.colorbar.html

2. Visualisation sous forme d'une surface : différentes fonctions permettent de visualiser des données en grille sous forme de surface.

   (a) Les données contenues dans le fichier `sein.npy` sont fournies pour des pas réguliers en x et y. Pour la visualisation de ces données sous forme de surface, il est nécessaire de les réorganiser sous forme de grille.

$$\underbrace{\begin{pmatrix} x_1 & y_1 & z_{11} \\ \vdots & \vdots & \vdots \\ x_n & y_1 & z_{n1} \\ x_1 & y_2 & z_{12} \\ \vdots & \vdots & \vdots \\ x_n & y_2 & z_{n2} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ x_1 & y_p & z_{1p} \\ \vdots & \vdots & \vdots \\ x_n & y_p & z_{np} \end{pmatrix}}_{3 \text{ vecteurs } p \cdot n} \rightarrow \underbrace{\begin{pmatrix} x_1 & \dots & x_n \\ \vdots & & \vdots \\ x_1 & \dots & x_n \end{pmatrix}, \begin{pmatrix} y_1 & \dots & y_1 \\ \vdots & & \vdots \\ y_p & \dots & y_p \end{pmatrix}, \begin{pmatrix} z_{11} & \dots & z_{n1} \\ \vdots & & \vdots \\ z_{1p} & \dots & z_{np} \end{pmatrix}}_{3 \text{ matrices } p \times n}$$

   Utiliser la fonction `reshape` pour transformer les vecteurs x, y et z de taille $n \cdot p$ en matrices de taille $p \times n$ ($p = n = 200$).

   (b) Utiliser la fonction `pcolormesh`[4] pour visualiser les données sous forme de surface. Utiliser la carte de couleur `gist_earth` et limiter l'étendue des données à $\pm$ 15 m.
   L'option `shading='auto'` permet d'avoir le rendu le plus esthétique.

   (c) Ajouter à la figure précédente des lignes de niveaux à l'aide de la fonction `contour`[5] ; tracer les lignes de niveaux en traits continus noirs et les espacer de -15 à 15 m par pas de 5 m.

3. Estompage d'un relief : il est possible d'accentuer les reliefs en appliquant un estompage (ou ombrage) sur les données. L'ombrage optimal consiste à considérer une source d'éclairement située à 45°W et 35° d'élévation (ce type d'estompage est appliqué sur les cartes de randonnées 1/25000e éditées en France par l'IGN).
   La classe `LightSource`[6] de `matplotlib.colors` permet de réaliser ce type d'ombrage. il faut pour cela :

   (a) Créer la source d'éclairement ;

   (b) Combiner la source d'éclairement aux données, à l'aide de la méthode `shade` avec :
   - Une exagération verticale de 5.
   - Des valeurs minimale et maximale à -15 et $+15$.

4. https://matplotlib.org/api/_as_gen/matplotlib.pyplot.pcolormesh.html
5. https://matplotlib.org/api/_as_gen/matplotlib.pyplot.contour.html
6. https://matplotlib.org/api/_as_gen/matplotlib.colors.LightSource.html

- Une fusion adoucie de l'ombrage avec la carte d'élévation (`blend_mode=soft`)

(c) Il en résulte un tenseur de dimensions $n \times m \times 4$ représentant l'image ombrée. Celle-ci peut alors être visualisée à l'aide de la fonction `imshow` [7].

2. Visualisation d'un objet tridimensionnel

La librairie `pyplot` propose une boite à outils pour la visualisation d'objet en trois dimensions, `mpl_toolkits.mplot3d` [8].
Pour le tracé en 3D, il suffit de préciser le type projection des axes après la création de la figure.

```
Python
# Import de la librairie
from mpl_toolkits.mplot3d import Axes3D
# Création de la figure
fig = plt.figure()
#Axes en 3D
ax = fig.gca(projection='3d')
```

Nous allons utiliser cette boite à outil pour le tracé d'un modèle de topographie terrestre appliqué sur une sphère.

1. Tracé d'une sphère

(a) On utilise la définition paramétrique de la sphère :

$$\begin{cases} x(\lambda,\varphi) &= a\cos\varphi\cos\lambda \\ y(\lambda,\varphi) &= a\cos\varphi\sin\lambda \\ z(\lambda,\varphi) &= a\sin\varphi \end{cases}$$

avec $a = 6{,}4$, $\lambda \in [0; 2\pi]$ et $\varphi \in [-\frac{\pi}{2}; \frac{\pi}{2}]$

i. La fonction `meshgrid` [9] permet de définir un maillage régulier de coordonnées de points : à partir des vecteurs $u = (u_1, \ldots, u_n)$ et $v = (v_1, \ldots, v_p)$, la fonction renvoie les matrices de dimension $p \times n$:

$$U = \begin{pmatrix} u_1 & \ldots & u_n \\ \vdots & & \vdots \\ u_1 & \ldots & u_n \end{pmatrix} \qquad V = \begin{pmatrix} v_1 & \ldots & v_1 \\ \vdots & & \vdots \\ v_p & \ldots & v_p \end{pmatrix}$$

Combiner les fonctions `linspace` et `meshgrid` pour définir un maillage régulier de 100×100 en (λ, φ).

ii. En déduire l'ensemble des points (x, y, z) de la sphère correspondant à ce maillage régulier.

7. https://matplotlib.org/api/_as_gen/matplotlib.pyplot.imshow.html
8. https://matplotlib.org/mpl_toolkits/index.html
9. https://docs.scipy.org/doc/numpy/reference/generated/numpy.meshgrid.html

(b) La méthode `plot_wireframe`[10] permet un tracé sous forme de fils de fer d'une structure 3D.
Appliquer cette méthode à l'objet `ax` pour visualiser la sphère.

(c) La méthode `plot_surface`[11] permet, comme son nom l'indique, de tracer une surface.
Tracer la sphère ; la colorer en bleu.
Il est possible de supprimer la coloration des contours des facettes en spécifiant l'option `linewidth=0`.

2. Visualisation de la topographie terrestre : le fichier `world.npz` contient un modèle topographique terrestre à une résolution de 0,6°, sous la forme de 3 variables : longitude [°] (λ), latitude [°] (φ) et élévation par rapport au niveau moyen des mers [m].
Le chargement du fichier et l'accès au variables se font de la manière suivante :

```Python
npz = np.load('world.npz')
lat = npz['lat']
lon = npz['lon']
elev = npz['elev']
```

(a) En utilisant la fonction `meshgrid`, calculer les points du globe à partir des latitudes et longitudes contenues dans le fichier.

(b) Pour colorer la surface, il faut que les valeurs d'élévation soient normalisées entre 0 et 1.
Calculer la grille d'élévation normalisée, `elev_n` à partir des valeurs minimale et maximale d'élévation.

(c) Utiliser la fonction `plot_surface` pour appliquer la topographie sur le globe terrestre :
- L'option `facecolors` permet de coloriser les facettes de la sphère ; on lui affecte une carte de couleur générée à partir des données d'élévation normalisée : `plt.get_cmap('gist_earth')(elev_n)` ;
- Fixer la résolution de la coloration des facettes sur la sphère à 2 (10 par défaut) à l'aide des options `cstride` et `rstride` ;
- Pour un résultat plus rapide (et plus esthétique !) , désactiver l'anticrénelage : `antialiased=False`.

10. https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html
11. https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html