TD Modélisation - Informatique:

Analyse de MNT et Exercice de Géologie Structurale (4h)

Objectifs pédagogiques de cette séance:

- Charger un MNT avec la bibliothèque **GDAL**
- Savoir manipuler un MNT: applications de filtres, calcul de pente, affichage des courbes de niveaux, etc.
- Exercice de géologie structurale: calcul de plan passant par un nuage de points (x, y, z), calcul de pendage et d'azimut.

Préambule sur les MNT LiDAR de l'IGN

L'IGN diffuse une cartographie 3D complète du sol et du sursol de la France via le programme LiDAR HD : `https://diffusion-lidarhd.ign.fr/mnx`.

Les données disponibles :

- Nuages de points 3D (bruts ou classifiés)
- Modèles Numériques de Terrain (MNT)
- Modèles Numériques de Surface (MNS)

Classification du LiDAR HD : sol, eau, végétation, bâtiments, ponts, sursol pérenne…

Partie 1: Analyse d'un MNT

Vous disposez sur **Arche** d'un extrait de MNT LiDAR HD de l'IGN aux alentours de Ribaute (Corbières): `MNT_LiDAR_Ribaute_crop.tif`.

Étape 1 – Exploration du fichier

- Ouvrir le fichier avec GDAL (reprendre TD analyse d'image pour les commandes).

👉 Quelle est la taille de l'image, sa résolution spatiale, sa résolution radiométrique (type de données `band.DataType`), ainsi que son système de coordonnées ?

- Lire la bande unique dans un tableau NumPy que vous appellerez `topo`. 👉 Où est situé le premier point du tableau NumPy (coin en haut à gauche ou coin en bas à droite)? Afficher l'image en niveaux de gris avec la bonne option `origin`: plt.imshow(array, origin='upper'/'lower') (https://matplotlib.org/stable/users/explain/artists/imshow_extent.html).
- Enregistrer le georeferencement et la projection dans les deux variables `gt` et `proj` / gt = `ds.GetGeoTransform()` et `proj = ds.GetProjection()`. Pour rappel, le géoréférencement renvoie 6 éléments: `gt = [originX, pixelWidth, rotX, originY, rotY, pixelHeight]`

Étape 2 – Extraction des coordonnées

- À partir du `GeoTransform`, calculer dans deux vecteurs `x` et `y` les coordonnées en x (axe des colonnes) et y (axe des lignes), de chaque pixel: faire une liste en compréhension ou utiliser la fonction `arange` de `numpy`.

- Rappels :
  - `gt[1] = pixelWidth > 0` (vers l'est)
  - `gt[5] = pixelHeight < 0` (vers le sud)
- À partir du `GeoTransform`, calculer les limites du raster (`xmin, xmax, ymin, ymax`).

Étape 3 – Filtrage et calcul du gradient

- Appliquer un filtre Gaussien pour lisser le MNT et nettoyer de bruit potentiel avec:

```
import scipy
filtered_input = scipy.ndimage.gaussian_filter (input, sigma)
```

- avec `input`, le tableau numpy à filtrer, et `sigma=(filter_x, filter_y)` l'écart type du noyau gaussien (plus sigma est for, plus le lissage est fort) en x et en y.
- Calculer le gradient de la topographie avec:

```
import numpy as np
dsm_dy, dsm_dx = np.gradient(filtered_topo, ±res_y, ±res_x)
```

- `np.gradient` calcule les dérivées partielles (pentes locales) d'un champ 2D en x (`dsm_dx : dz/dx`) et y (`dsm_dy: dz/dy`). Faite en sorte que le gradient en y soit positif vers le nord et le gradient en x soit positif vers l'est en ajustant les signes des résolutions en x et en y (res_y, res_x).
- À partir du calcul de gradient précédent (`dz/dx`, `dz/dy`), calculer la pente (`slope`) ainsi que l'orientation (`aspect`) de tous les pixels du MNT. On définira l'aspect à partir du nord, positif dans le sens horaire (voir figure).
- Il est aussi préférable de définir l'aspect entre 0 et 360° afin d'éviter les discontinuités brutales en +180° et -180°:

```
aspect_geo = (np.rad2deg(aspect) + 360) % 360
```

- Afficher les résultats sous forme d'images (`plt.imshow()`) avec des échelles de couleur appropriées. Utiliser la fonction:

```
fig = plt.figure(figsize=(…))
ax = fig.add_subplot(2, 2 , 1)
```

ou

```
fig, axes = plt.subplots(2, 2, figsize=(…))
```

pour afficher plusieurs graphiques sur une même figure et combiner ainsi, gradients, aspects et pentes sur une même figure. Vérifier les signes des gradients et de l'aspect en fonction de l'orientation des versants.

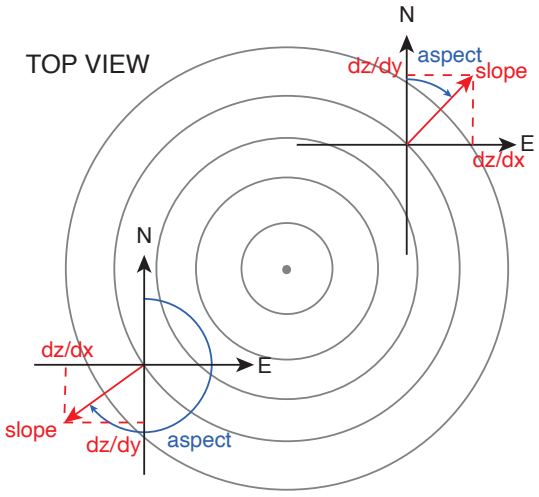

Figure: Représentation des angles slope et aspect sur une montagne schématique

Étape 4 – Calcul de l'ombrage

L'ombrage (hillshade en anglais) correspond à une simulation d'éclairage du relief selon une position du soleil (azimut et hauteur) qui se calcule à partir de la pente et de l'orientation (aspect).

Si:

- `slope` = pente (en radians)

- `aspect` = orientation (en radians)
- `azimuth_rad` = azimut du soleil en radians (0°: nord, 90°: sud)
- `altitude_rad` = hauteur du soleil en radians (entre 0° et 90°)

Alors:

$$hillshade = 255.[cos(altitude_{rad}).cos(slope) + sin(altitude_{rad}).sin(slope).cos(azimuth_{rad} - aspect)]$$

- Calculer l'ombrage du MNT

Étape 5 – Sauvegarde des résultats

Vous allez maintenant sauvegarder vos résultats sous forme de rasters GeoTIFF.
Les cartes de **pente**, **orientation (aspect),** et **ombrage (hillshade)** peuvent, en effet, être enregistrées sur disque afin de pouvoir être ouvertes dans un SIG (QGIS, ArcGIS).

Pour cela, créez une fonction `save_raster()` qui prend en entrée :

- le **nom de fichier** de sortie,
- un **tableau** NumPy à enregistrer,
- ainsi que les **métadonnées** de l'image (géoréférencement - `gt`, projection - `proj`).

Cette fonction utilisera GDAL pour :

1. Créer un *driver* adapté au format souhaité (GeoTIFF) : `drv = gdal.GetDriverByName('GTiff')`
2. Créer un fichier raster vide avec les dimensions de votre MNT : `dst = drv.Create(filename, ncols, nlines, nb_bandes, gdal.GDT_Float32),` en spécifiant ses dimensions, son nombre de bandes, et le type de données de l'image.
3. Copier les informations de géoréférencement et de projection depuis le MNT source :`dst.SetGeoTransform(gt)` et `dst.SetProjection(proj)`
4. Écrire vos résultats dans la première bande : `dst.GetRasterBand(1).WriteArray(array)`

Étape 6 – Lignes de niveau (contours)

- Tracer des **lignes de niveau** (`plt.contour()` – `https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.contour.html`) sur le MNT ombré.
- Superposer les contours sur le MNT coloré pour visualiser le relief.
  ⚠️ Attention :
  - Par défaut, `imshow()` considère que l'origine se situe en haut à gauche (`origin="upper"`).
  - À l'inverse, `contour()` et `contourf()` supposent une origine en bas à gauche (`origin="lower"`).
    Cela peut conduire à un décalage entre le raster et les courbes.

👉 Pour éviter ce piège, il est **recommandé** de toujours fournir les limites et l'origine du raster :

```
ax.imshow(…, origin='upper', extent=[xmin, xmax, ymin, ymax])
contours = ax.contour(…, origin='upper', extent=[xmin, xmax, ymin, ymax])
```

De cette façon, l'alignement est assuré, quelle que soit l'option origin.

- Rajouter des etiquettes aux lignes de niveau avec la commande:

`ax.clabel(contours, …)` – https://matplotlib.org/stable/gallery/images_contours_and_fields/contour_demo.html#sphx-glr-gallery-images-contours-and-fields-contour-demo-py

Partie 2: Exercice de Géologie Structurale

Vous disposez sur **Arche** de deux fichiers CSV contenant une cartographie de deux bancs de la formation de Besson à partir du MNT LiDAR : `besson_banc1.csv`, `besson_banc2.csv`.

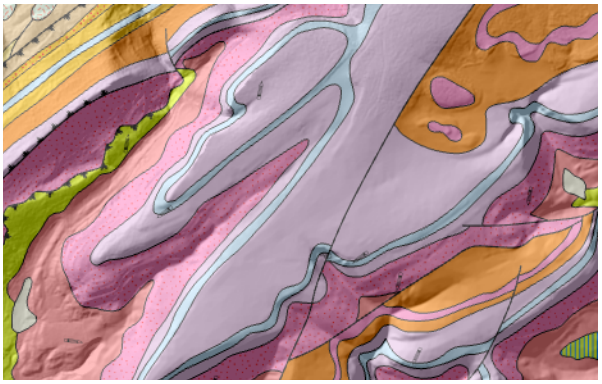

Figure: Carte géologique de la zone d'étude

Étape 1 – Lecture des données

- Charger les deux fichiers CSV avec NumPy. ⚠️ Utiliser l'argument `skiprows` pour sauter les premières lignes.
- Afficher les points de mesure sur le MNT ombré.

Étape 2 – Ajustement d'un plan

- Écrire une fonction `fit_plane(x, y, z)` qui ajuste un plan aux mesures de chaque banc par régression linéaire (méthode des moindres carrés vu dans le TD1):

$$z = ax + by + c.$$

Avec x, y, z, les positions east, nord, et verticales des mesures, et a,b les coefficients du plan.

- Donner l'équation du plan ajusté pour les deux bancs.
- Le vecteur normal au plan est:

$$\vec{n} = (a, b, -1).$$

- Le **strike** (azimut) correspond à la direction de la ligne d'intersection du plan avec l'horizontale :

$$strike = atan2(b, a)$$

- Le **pendage** est l'angle entre le plan et l'horizontale: $dip = acos(\frac{-1}{sqrt(a^2 + b^2 + (-1)^2)})$

- Donner l'azimuth et le pendage des deux bancs. Vérifier la relation entre la topographie et l'orientation des bancs.

Étape 3 – Visualisation 3D

- Afficher les résultats en 3D avec:

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
```

- Ajouter le plan ajusté pour chaque banc avec `ax.plot_surface(X, Y, Z, ...)`, où X et Y correspondent à une grille générée pour les plans (https://numpy.org/doc/2.1/reference/generated/numpy.meshgrid.html), et `Z` les altitudes des plans pour ces positions `(X,Y)`.

- Superposer les points de mesure des bancs avec des couleurs différentes (`ax.scatter(x1, y1, z1, c= ...)`, `ax.scatter(x2, y2, z2, c=…)`).

Étape 4 – Pour aller plus loin…

- Calculer l'orientation et le pendage des failles à l'aide de la carte géol. puis réaliser une coupe structurale.