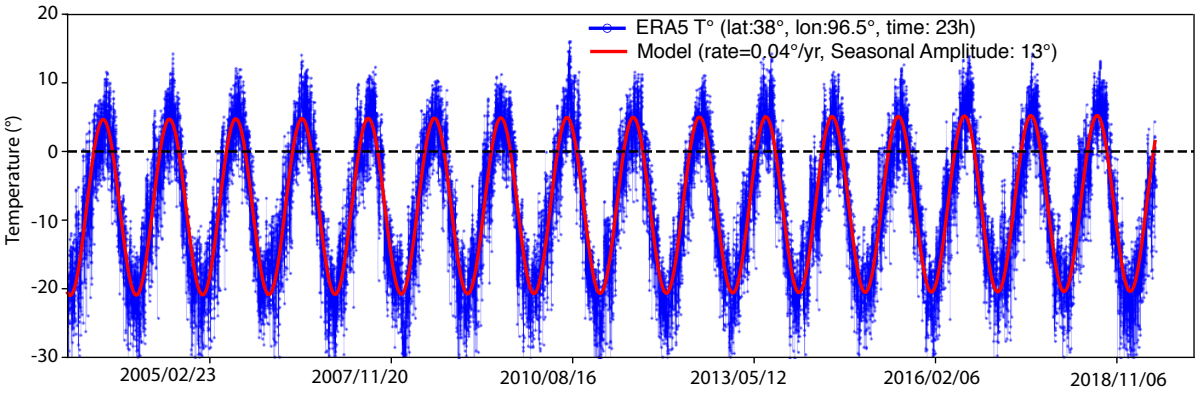
Examen d' Informatique / Modélisation (1h30)

Documents autorisés. Ordinateurs, tablettes, téléphones interdits.

AF 1: Être capable de formuler un problème sous forme d'un modèle. Savoir faire appel aux outils numériques adaptés au problème.

Partie 1: Structure thermique du sol



Séries temporelles de température (T°) sur le plateau Tibétain aux coordonnés 38° N et 96.5°E issues du modèle météorologique ERA-5 (points bleus) et du modèle théorique (courbe rouge). La température moyenne de surface (MAGST, pour Mean Annual Ground Surface Temperature) est de -8°C.

Vous disposez de températures journalières mesurées à 2m au-dessus du sol, à 23h, pour la période 2003-2019 (Figure 1) que vous avez ouvert dans un code Python avec la librairie NumPy.

Question 1: Décrire une méthode permettant d'ajuster un modèle linéaire et saisonnier (oscillations annuelles) aux données (courbe rouge), et ainsi d'extraire une tendance linéaire ainsi qu'un comportement saisonnier des T°.

Cf. TD1

$T(t) = Acos(2\pi t) + Bsin(2\pi t) + Ct + D$, avec:

C'est un problème inverse linéaire, avec T les données et A, B, C, D les inconnus, qui se résout par moindres carrés.

Question 2: À quoi correspondent les valeurs « rate » et « seasonal amplitude » issues de cette modélisation sur la figure 1 ? Comment serait-il possible de calculer le jour de l'année où le pic de température est atteint ?

- $\sqrt{A^2 + B^2}$: amplitude saisonniere des températures,
- $\arctan B/A$: phase correspondant à la date des maxima saisonniers,
- C: tendance linéaire (augmentation ou diminution des températures moyennes),
- D : température moyenne.

Vous souhaitez maintenant étudier la structure thermique d'un sol au dessus d'un pergélisol, schématisé en Figure 2 et contrôlée par la loi de Fourrier:

$$Q = -k\frac{\partial T}{\partial z}$$

Dans laquelle Q est le flux de chaleur par unité de surface en W/m^2, k la conductivité thermique du milieu en $W\,.\,m^{-1}\,.\,K^{-1}$ toujours positive , et T est le champ scalaire de températures.

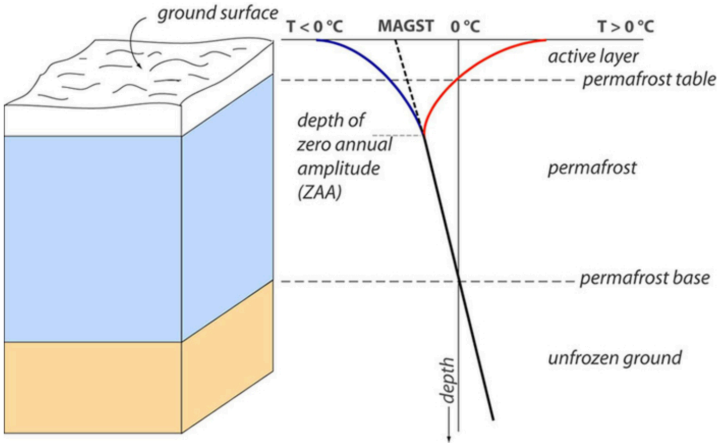


Figure 2: Profil schématique de T° en fonction de la profondeur (z) au travers du pergélisol. La température de surface et le gradient géothermique (Q/k) contrôle le profil géo-thermique (droite noire).

Pour rappel des notions vues dans l'UE Surfaces Continentales du S6, la température doit rester sous 0 °C pour que le sol soit considéré comme du pergélisol. La couche active est quant à elle la partie superficielle du sol qui gèle en hiver et dégèle en été.

On souhaite maintenant étudier quantitativement comment se propage une anomalie de chaleur saisonnière annuelle en fonction de la profondeur. Le problème étant unidimensionnel, l'équation générale de diffusion s'écrit:

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial z^2}$$

où z est la profondeur.

La diffusivité thermique des sols varie selon leur composition (argileux, sablonneux, humifère, etc.), leur teneur en eau, et leur densité. En général, les valeurs de diffusivité thermique se situent dans l'intervalle suivant:

- Sols secs: $\kappa = 0.5.10^{-6} m^2/s$.
- Sols humide: $\kappa = 1.10^{-6} m^2/s$.

Question 3: En combien de temps les variations saisonnières de température se propagent-elles dans un sol humide sur une profondeur de 2 mètres ? Donner le résultat en jour.

Une variation de température se propagera sur une profondeur de 2 m en 4000000.0 s (soit 46 jours).

Question 4: Compléter le script Python ci-dessous permettant de construire un maillage T(z, t) fonction de l'espace et du temps, avec en ligne la profondeur z de 0 à 25m et en colonne le temps de t=0 jusqu'au dernier temps t choisi pour notre calcul (365 jours).

```
# Paramètres du maillage selon la profondeur
zmax = 25  # profondeur maximale en mètres
dz = .1  # pas de profondeur en mètres
Nz = int(zmax / dz)   # nombre de point en profondeur

# Discrétisation temporelle
d2s = 60 * 60 * 24  # conversion des jours en secondes
dt = 0.01 * d2s  # pas temporel en secondes
calcul_days = 365  # durée de la simulation en jours
Nt = int(calcul_days * d2s / dt)           # nombre d'itération

# Initialisation du maillage des température T(z, t).
T = np.zeros((Nz + 1, Nt + 1))             # maillage des T°

# Maillage en profondeur z.
z = np.arange(0, zmax + dz , dz)           # axe des profondeurs
Ou z = np.linspace(0, zmax, Nz + 1)

# Maillage en temps t.
t = np.linspace(0, calcul_days * d2s, Nt)      # axe du temps
Ou t = np.arange(0, tmax + dt , dt)
```

Question 5: Quelle sont les profondeurs aux indices 0 et 12 de l'axe des abscisses ?
0 mètre (surface) et 1.2 mètres (car 12 × 0.1 = 1.2).

Question 6: Quelles conditions limites en surface seront ici utilisées pour résoudre le schéma explicite et étudier comment se propagent des variations saisonnières de température ? Comment est il possible d'initialiser la température en fonction de la profondeur sans anomalies de température ?

En surface: temperatures de surfaces saisonnières mesurées ou modélisées dans la question 1.
En profondeur: En profondeur et sans anomalies les température suivent le gradient géothermique / T(z) = MAGST + $\frac{\partial T}{\partial z}$

Question 7: Quelle condition de stabilité du schéma explicite doit on vérifier sur le pas de temps ?

La stabilité du schéma explicite est donné pour un pas de temps et d'espace qui satisfont la relation $dt < \frac{1}{2}\frac{dz^2}{\kappa}$ = 0.058 jours.

Question 8: D'après les résultats de la Figure 3, que pouvez-vous dire sur la profondeur de la couche active ? Donner approximativement le décalage en jour entre les T° de surface et les T° à la profondeur 1.8m

En dessous de la couche active, les T° sont toujours sous 0°C => 1.4m
Á la profondeur 1.8, les T° maximales sont atteintes 40-50jours apres le maximum des T° de surface.

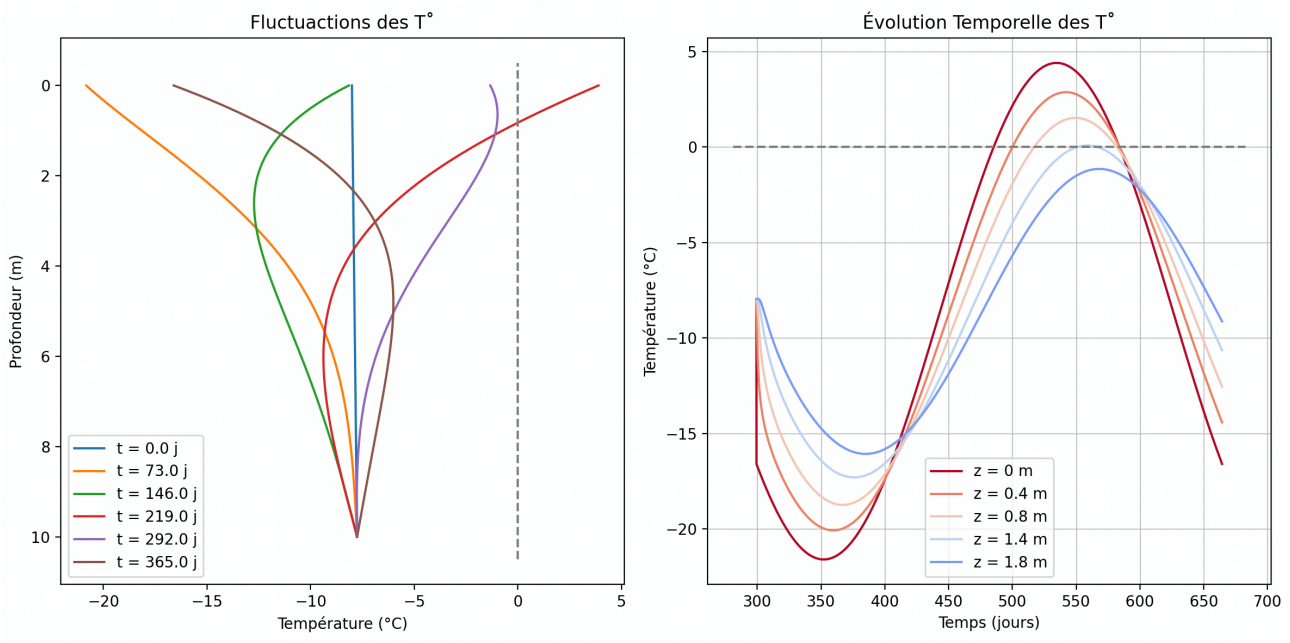


Figure 3: Résultats obtenus après résolution numérique de l'équation différentielle de la chaleur ave un schéma explicite. À gauche: Fluctuation de température en fonction de la profondeur pour différents temps. À droite: Évolution de la T° en fonction du temps pour différentes profondeurs.

AF 2: Être capable de développer des programmes structurés et robustes en Python, intégrant des bibliothèques scientifiques, des interfaces graphiques et des visualisations avancées.

Merci de répondre aux questions suivantes sur le document.

Question 9: On considère le tableau NumPy suivant:

```
A = np.array([
    [1, 2, 3, 4],
    [5, 6, 7, 8],
    [9, 10, 11, 12]
])
```

Que renvoie :

- `A[0:1, 1]: 2`
- `A[-1, -1]: 12`
- `np.shape(A.reshape(4,3)): (4,3)`
- `A[-1,:] > 10: False, False, True, True`

Question 10: Charger les deux dernières colonnes du fichier texte suivant avec la bibliothèque Numpy dans un tableau que vous appellerez `data`.

```
# Permanent GNSS network from INGV
# Country stations longitudes latitudes
Italy CIVI 11.9386 44.0066
Italy PENX 12.2651 43.8173
Italy SENI 13.2150 43.7076
Italy PITI 11.6731 42.6339
Italy ACQU 11.8648 42.7440
….
```

```
import numpy as np
data = np.loadtxt("fichier.txt", skiprows=2, usecols=(2, 3), dtype=float, unpack = False)
```

Vous disposez d'un vecteur représentant les altitudes (en mètres) de points le long d'un sentier de randonnée pris tous les 2 mètres :

```
altitudes = np.array([1251, 1430.3, 1182, 1621, 989.4, 952, 1021])
```

Question 11: Utiliser NumPy pour trouver l'indice du point culminant du sentier:

```
indice_culminant = np.argmax(altitudes)
```

Question 12: Les données présentent quelques erreurs de mesure. Utiliser une fonction SciPy pour filter le vecteur avec une fenêtre de 2 et ainsi atténuer les erreurs de mesure:

```
from scipy.ndimage import gaussian_filter
altitudes_filtrees = gaussian_filter(altitudes, sigma=2)
```

Question 13: Calculer la pente le long du chemin de randonnée:

```
pentes = np.gradient(altitudes)
```

Question 14: Compléter le code suivant permettant l'affichage d'un Modèle Numérique d'Élévation avec la librairie `Matplotlib`.

```
from osgeo import gdal
import numpy as np
import matplotlib.pyplot as plt

# -------------------
# Ouverture de l'image
# -------------------

ds = gdal.Open("MNT.tiff", gdal.GA_ReadOnly)
band = ds.GetRasterBand(1)
image =    band.ReadAsArray()                          À compléter

# extraction du nombre de ligne et de colonne
ncols, nlines = ds.RasterXSize, ds.RasterYSize         À compléter
print("Taille (X,Y) :", ncols, nlines)

# --- Visualisation ---
fig, ax = plt.subplots(figsize=(8, 6))

# Affichage de l'image
```

```
im =  ax.imshow(image)                                  À compléter


# Affichage de la palette de couleur
fig.colorbar(im, ax=ax, label="Altitude (m)")           À compléter


# Ajouter le titre Modèle « Numérique d’Élévation »
# à la figure
ax.set_title("Modèle Numérique d'Élévation")            À compléter
```