DM 1 modélisation

Question 1

La taille totale d'un fichier est de

$$64 \times 64 \times 352 \times 12 = 17301504 bits \approx 2.16 Mo$$

Question 2

Le taux de compression vaut alors :

$$\tau = \frac{T - Tc}{T} = \frac{2.16 - 1}{2.16} = 0.5376$$

La réduction de la taille du fichier est donc de 53.76%

Question 3

Soit $S_n = 4 - 5 - 7 - 0 - 7 - 8 - 1 - 7 - 4$
l'entropie $H(S_n)$ est définie de la manière suivante et vaut alors:
$$H(S_n) = -\sum_{i=1}^{N_v} p_i \times log_2(p_i) = -(2 \times 0.3 \times log_2(0.3) + 4 \times 0.1 \times log_2(0.1)) = 2.37$$
La longueur moyenne de $2.4 bits$ par caractère correspond à la valeur trouvée

Question 4,5,6

```
#### DEFINITION DE LA FONCTION ENTROPIE

def entropie (S):

    #1. crée une liste de valeurs sans doublons (liste contenant une seule fois chaque valeur de la liste)

    #Question 5
    valeurs = list (set (S))
    occ_i , proba = [] , []
    for i in valeurs
        somme=0
        for j in S:
```

```
            if i==j:
                somme+=1
        occ_i.append(somme)
        proba.append(somme/len(S))

    #Question 6
    H=-sum(proba[i]*log2(proba[i]) for i in range(len(valeurs)))

    return H
```

Question 7

```
def tau (bloc_image):
    T=12*len(bloc_image)
    Tc= entropie(bloc_image)*len(bloc_image)
    return (T-Tc)/T
```

Question 8

utilisation de Numpy `import numpy as np`

```
def pretraitement12(donnees_brutes):
    luminance_moy = np.mean(donnees_brutes[[1, 11, 21, 31],:])
    for j in range(len(luminance_moy)):
        if luminance_moy[j] > luminance_max:
            luminance_max , j_max = luminance_moy[j], j
    return(luminance_moy, luminance_max, j_max)
```

Question 9

```
def pretraitement34(donnees_brutes, luminance_max, j_max):
    spectre_max = donnees_brutes[:, j_max]
    spectre_max/=luminance_max
    return(spectre_max)
```

Question 10

```python
def pretraitement5(luminance_moy,spectre_max):
    matrice_modele = np.dot(spectre_max,luminance_moy)
    return(matrice_modele)
```

Question 11

Pour l'étape 12, fait intervenir un boucle range et un np.mean devant parcourir l'ensemble de la matrice pour effectuer une moyenne. On peut donc supposer que la complexité de l'étape 12 est un $O(2n)$

Pour l'étape 34 on effectue également une boucle avec n itérations. On peut donc supposer que cette étape possède une complexité $O(n)$

Pour l'étape 5 on effectue un produit de matrices. on peut assimiler la fonction np.dot à deux boucles imbriquées. Il s'agit donc surement d'une complexité $O(n^2)$

Question 12

```python
def prediction(x):
    erreur=[x[i]-prediction[i-1] for i in range(1,len(X))]
    return(erreur)
```

Question 13

```python
def mappage(erreur,x):
    delta=[]
    for i in range(1,len(x))
        theta=min(x[i-1],4095-x[i-1])
        erreur=x[i]-x[i-1]
        if 0<=erreur and erreur<=theta:
            delta.append(2*erreur)
        if -theta<=erreur<=0:
            delta.append(2*abs(erreur)-1)
        else:
            delta.append(theta+abs(erreur))
    return(delta)
```

Question 14

décimal	rice p = 8
4	0 100
10	10 010
18	110 010
18	110 010
6	0 110
1	0 001
3	0 011

Question 15,16

```
def codage(delta_k,p_opt):
    quotient = delta_k//p_opt
    codage_unaire=str()
    for i in range (quotient):
        codage_unaire+="1"
    codage_unaire+="0"

    reste=delta_k%p_opt
    codage_rice=int(codage_unaire+str(bin(reste)))

    return([codage_rice])
```

Question 17

Cette équation a été obtenue à l'aide du principe fondamental de la dynamique. Ici seule la force d'attraction gravitationnelle de Vénus sur la sonde.

$$-G \times \frac{m_v \times m_{sonde}}{||\vec{r} - \vec{r}_v||^2} \times \frac{\vec{r} - \vec{r}_v}{||\vec{r} - \vec{r}_v||}$$

et la force d'attraction gravitationnelle du soleil sur la sonde ont été prises en compte.

$$-G \times \frac{m_s \times m_{sonde}}{||\vec{r}||^2} \times \frac{\vec{r}}{||\vec{r}||}$$

Question 18

S_x et S_y correspondent d'après l'équation ci dessus:

$$
\begin{aligned}
S_x(x(t), y(t), t) = & - G \times m_s \times \frac{x(t)}{(x(t)^2 + y(t)^2)^{\frac{3}{2}}} \\
& - G \times m_v \times \frac{x(t) - x_v(t)}{((x(t) - x_v(t))^2 + (y(t) - y_v(t))^2)^{\frac{3}{2}}} \\
S_y(x(t), y(t), t) = & - G \times m_s \times \frac{y(t)}{(x(t)^2 + y(t)^2)^{\frac{3}{2}}} \\
& - G \times m_v \times \frac{y(t) - y_v(t)}{((x(t) - x_v(t))^2 + (y(t) - y_v(t))^2)^{\frac{3}{2}}}
\end{aligned}
$$

Or d'après l'énoncé:

$$
\begin{aligned}
x_v(t) = r_v cos(\Omega t + \Phi) \\
y_v(t) = r_v sin(\Omega t + \Phi)
\end{aligned}
$$

Ainsi :

$$
\begin{aligned}
S_x(x(t), y(t), t) = & - G \times m_s \times \frac{x(t)}{(x(t)^2 + y(t)^2)^{\frac{3}{2}}} \\
& - G \times m_v \times \frac{x(t) - r_v cos(\Omega t + \Phi)}{((x(t) - r_v cos(\Omega t + \Phi))^2 + (y(t) - r_v sin(\Omega t + \Phi))^2)^{\frac{3}{2}}} \\
S_y(x(t), y(t), t) = & - G \times m_s \times \frac{y(t)}{(x(t)^2 + y(t)^2)^{\frac{3}{2}}} \\
& - G \times m_v \times \frac{y(t) - r_v sin(\Omega t + \Phi)}{((x(t) - r_v cos(\Omega t + \Phi))^2 + (y(t) - r_v sin(\Omega t + \Phi))^2)^{\frac{3}{2}}}
\end{aligned}
$$

Le code qui renvoie Sx et Sy est:

```
def eval_sm(x,y,t):
    # Constantes
    r_v = 1.08e11  # Distance (en mètres)
    Omega = 3.2e-7  # Fréquence angulaire (en rad/s)
    Gm_v = 3.2e14  # G*m_v (en m^3/s^2)
    Gm_s = 1.3e20  # G*m_s (en m^3/s^2)
    Phi = 0  # Phase initiale (en radians, si nécessaire)

    r_norme = x_t**2 + y_t**2
    rv_cos = x_t - r_v * cos(Omega * t + Phi)
    rv_sin = y_t - r_v * sin(Omega * t + Phi)

    # Équation pour S_x
    S_x = -Gm_s * x_t / r_norme**(3/2)-Gm_v * rv_cos / (rv_cos**2
+ rv_sin**2)**(3/2)
```

```python
    # Équation pour S_y
    S_y = -Gm_s * y_t / r_norme**(3/2)-Gm_v * rv_sin / (rv_cos**2
+ rv_sin**2)**(3/2)

    return (S_x,S_y)
```

Question 19

En appliquant la méthode d'Euler, on obtient deux relations de récurrence entre u_{i+1} et u_i et entre v_{i+1} et v_i

$$u_{i+1} = u_i + \Delta t \times S_x(x(t), y(t), t)$$
$$v_{i+1} = v_i + \Delta t \times S_y(x(t), y(t), t)$$

Question 20

D'après la méthode explicite d'Euler, on obtient deux relations de récurrence entre x_{i+1} et x_i et entre y_{i+1} et y_i

$$x_{i+1} = x_i + \Delta t \times u_i$$
$$y_{i+1} = y_i + \Delta t \times v_i$$

Question 21,22

```python
import numpy as np
temps=int(input("durée de la simulation"))
t= np.linspace(0,temps,temps/n)
x,y,u,v=[0],[0],[0],[0]
for i in range(len(t)):
    u.append(u[i]+deltaT*eval_sm(x[i],y[i],t[i])[0])
    v.append(u[i]+deltaT*eval_sm(x[i],y[i],t[i])[1])
    x.append(x[i]+deltaT*u[i])
    y.append(y[i]+deltaT*v[i])
```

Question23

La quantité de mémoire nécessaire pour réaliser cette simulation numérique pour une durée de 30 jours avec un pas de 1 seconde et x,y,u,v des listes contenant des flottants codés sur 8 octes est de: $4 \times 8 \times 30 \times 24 \times 3600 = 103.68 Mo$

Question 24

```python
import numpy as np
import matplotlib.pyplot as plt
def vitesse_sonde(u,v,t):
    vitesse=sqrt(u**2+v**2)*10**(-3)
    plt.plot(t,vitesse)
    plt.xlabel('Temps (s)')
    plt.ylabel('vitesse (km/s)')
    plt.show()
```