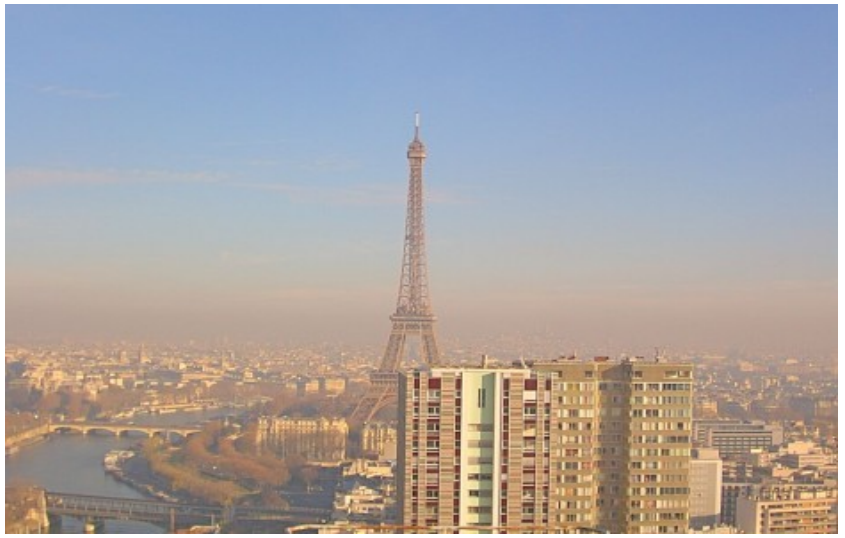
CHESTERIKOFF
Aymeric
MP
N°32167

TIPE : Santé et prévention

Titre : Modélisation simplifiée de l'évolution des concentrations d'ozone au niveau du sol

Sommaire :

Objectif : A l'aide d'une modélisation simplifiée de l'atmosphère, prévoir le nombre de jours de beau temps nécessaires avant de dépasser les seuils d'alerte de pollution à l'ozone fixés par l'OMS (organisation mondiale de la santé).



Source : thelocal.fr

INTRODUCTION :

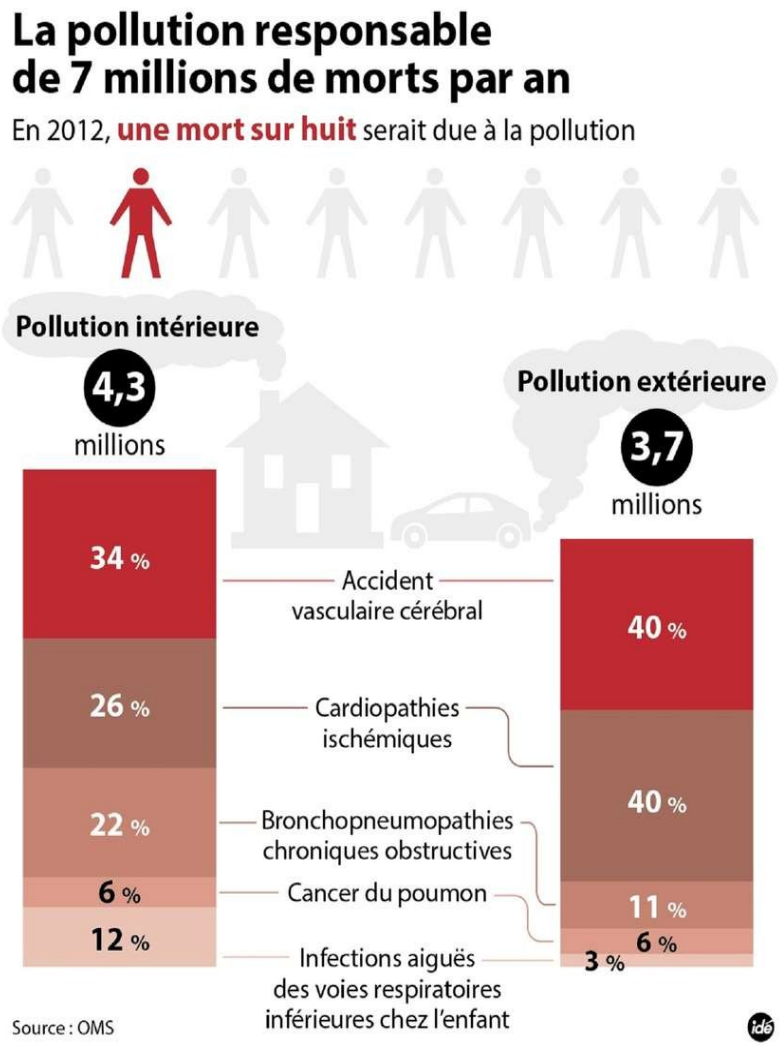
- Lien avec le thème :

Pollution = 1ère cause de mortalité dans le monde

Problème :

Les polluants :
- ignorent les frontières
- lourd coût économique
- dégradent la biodiversité (pluie acide)

► Nécessité de prévoir et gérer les futurs pics de pollution



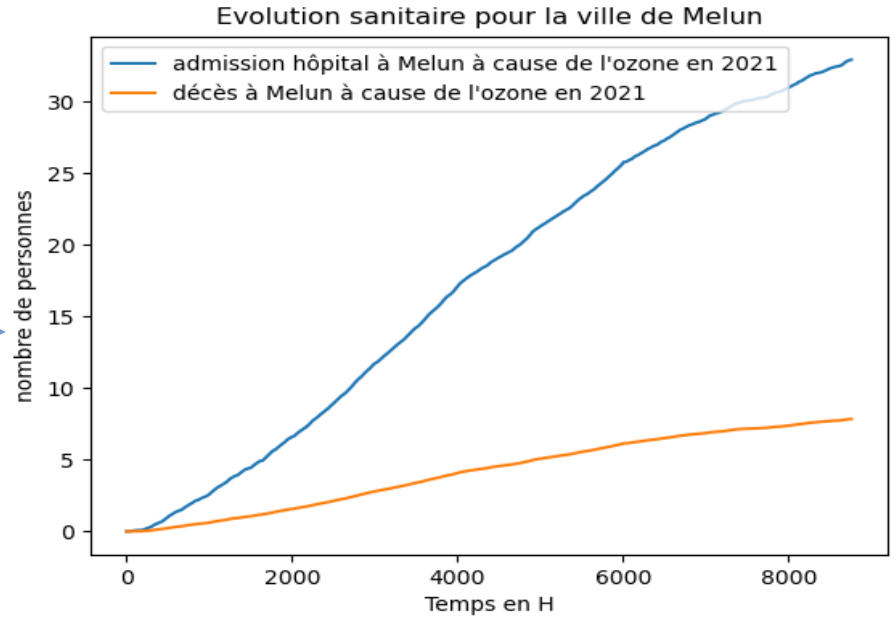
INTRODUCTION :

- Impact sanitaire de la pollution à l'ozone à Melun (40 000 habitants) :

En France en 2021 :
- 660361 morts en France
- 12.9 millions d'admissions à l'hôpital

Statistiques santé INSEE + relevés ozone Airparif (station Melun-Villaroche)

► *Saisie des données python :*



► Résultats modélisation pour 2021:

Impact ozone	Nombres Hospitalisations	Hospitalisations %	Nombres Morts	Morts %
Melun	32	0,43 %	7	2 %
IDF	10046	0,52 %	2395	2 %

Conclusion : Pollution à l'ozone = dangereuse pour la santé

I) Modélisation de l'angle zénithal

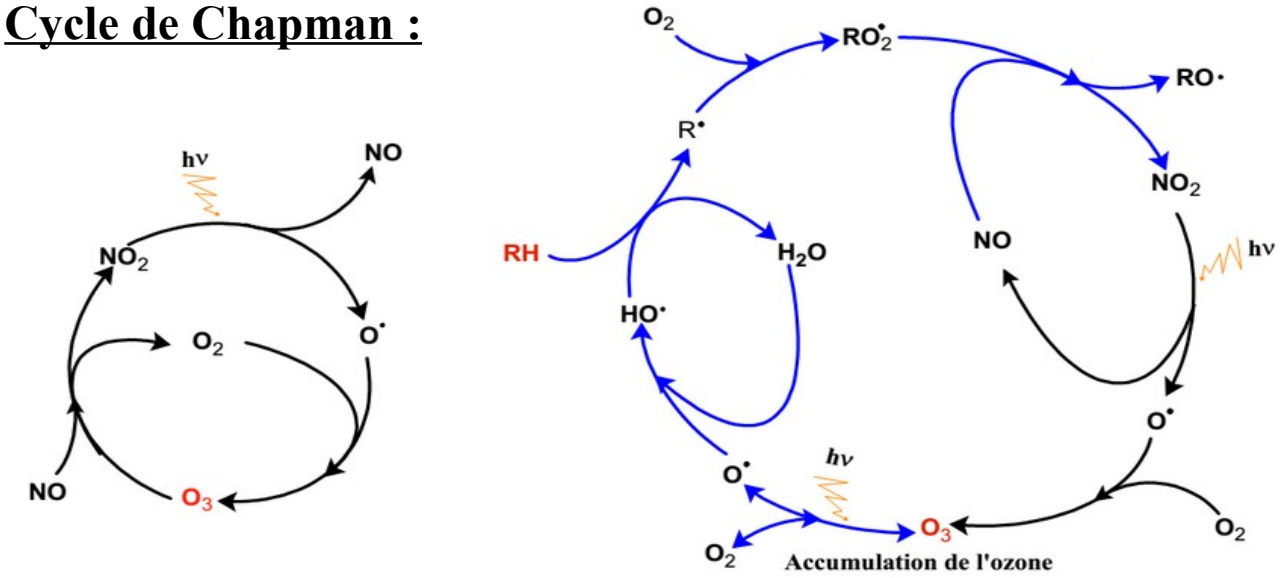
- Qu'est ce que le cycle de Chapman ?

Double rôle de l'ozone (O_3) :

- Ozone stratosphérique = couche d'ozone, protège des rayons UV-B et UV-C (100 à 315nm)

- Ozone troposphérique = polluant de la troposphère, troisième gaz à effet de serre

Cycle de Chapman :



a) Cycle de Chapman sans COV

b) En présence de COV=RH

Source: researchgate.net

►Concentration d'ozone varie en fonction :
- De l'**ensoleillement** (hν)
- Des **concentrations des autres polluants** (NO2...)

Autres facteurs ?

I) Modélisation de l'angle zénithal

- Influence des différents paramètres météo sur le cycle de Chapman :

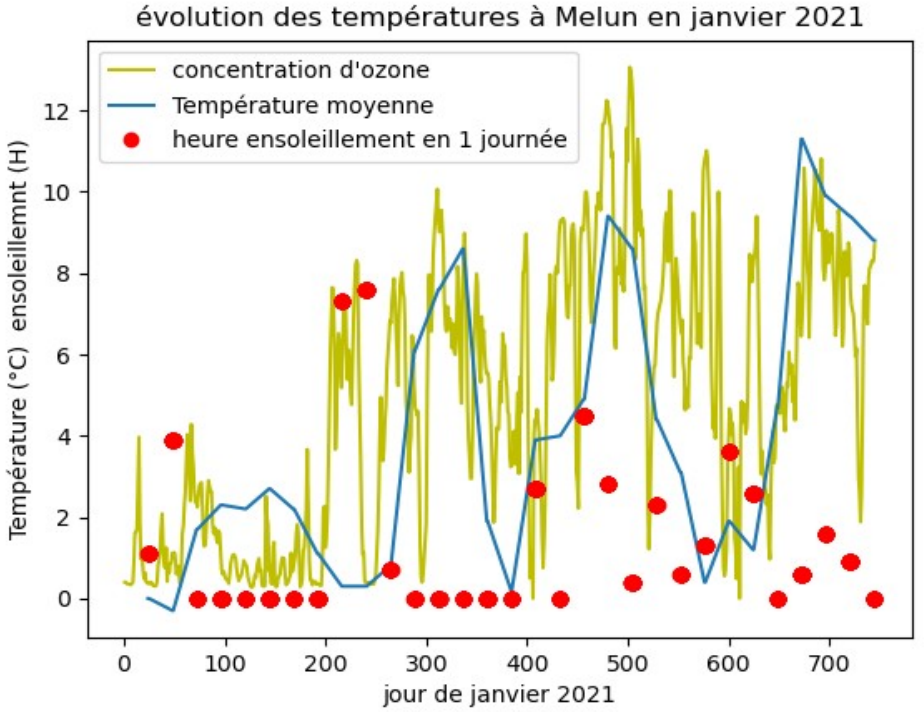
Influence lieux/activités humaines :

Pollution NO_2 :

Source : Airparif

► Lieux influence la pollution d'ozone

Mise en relations des différents paramètres (température, ensoleillement) sous python :



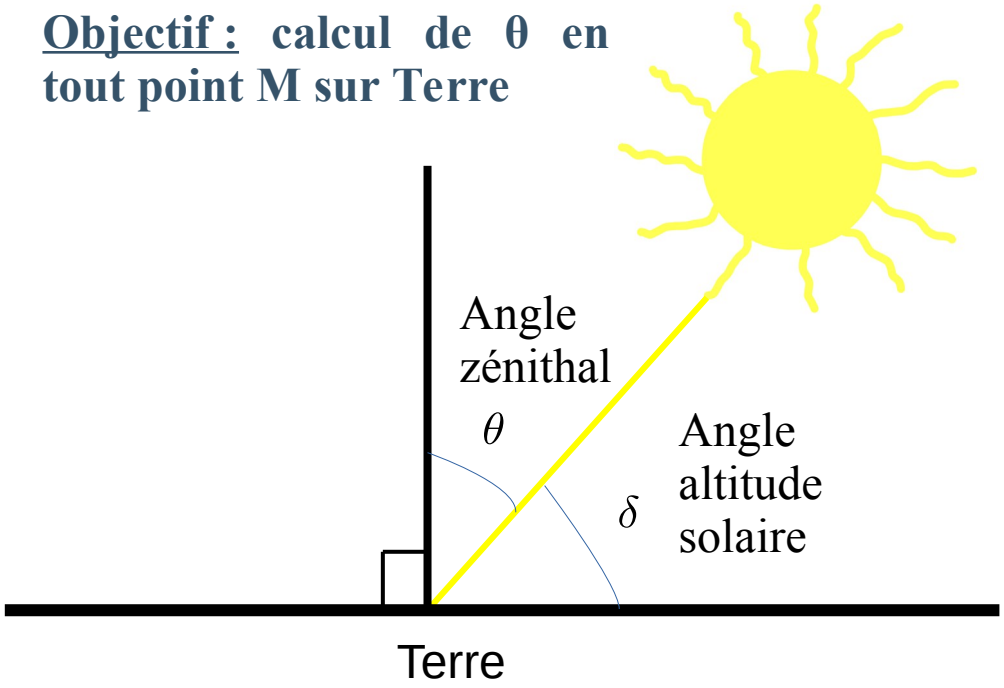
► Température et ensoleillement (énergie solaire) influencent la concentration d'ozone

⇨ **Modélisation journalière des variations de l'angle zénithal pour modéliser le cycle de Chapman**

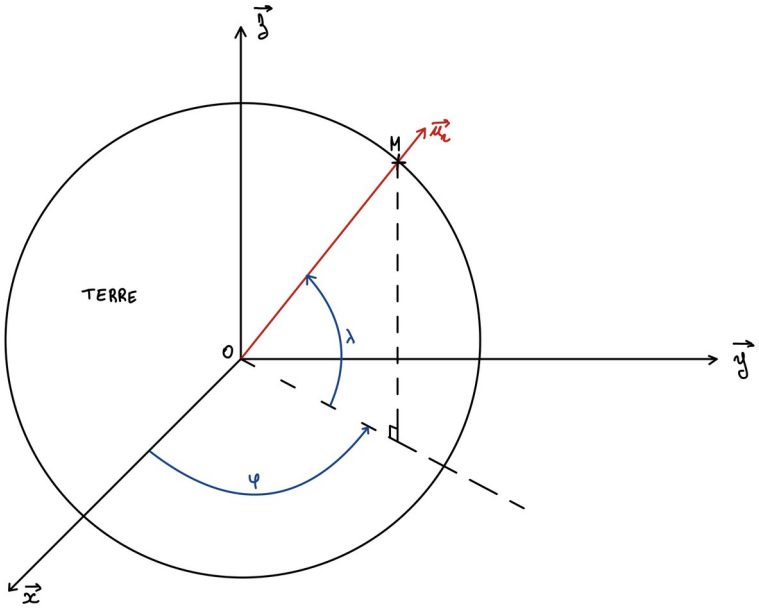
I) Modélisation de l'angle zénithal

- Modélisation de l'angle zénithal :

Objectif : calcul de θ en tout point M sur Terre



Référentiel :
(Oxyz) géocentrique



Données :

φ0 : longitude de M
λ : latitude de M
Melun :

- $\varphi 0 = 2,66°$
- $\lambda = 48,5°$

Soit t (seconde), origine des temps : équinoxe de printemps heure de Greenwich

► Pour tout t : $\begin{cases} \varphi(t) = \varphi 0 + \omega t \\ \omega = \dfrac{2\pi}{1jour} = \dfrac{2\pi}{86164} \end{cases}$

$$\overrightarrow{ur} = \begin{pmatrix} \cos\lambda \times \cos\varphi\, \overrightarrow{x} \\ \cos\lambda \times \sin\varphi\, \overrightarrow{y} \\ \sin\lambda\, \overrightarrow{z} \end{pmatrix}$$

I) Modélisation de l'angle zénithal

- Modélisation de l'angle zénithal :

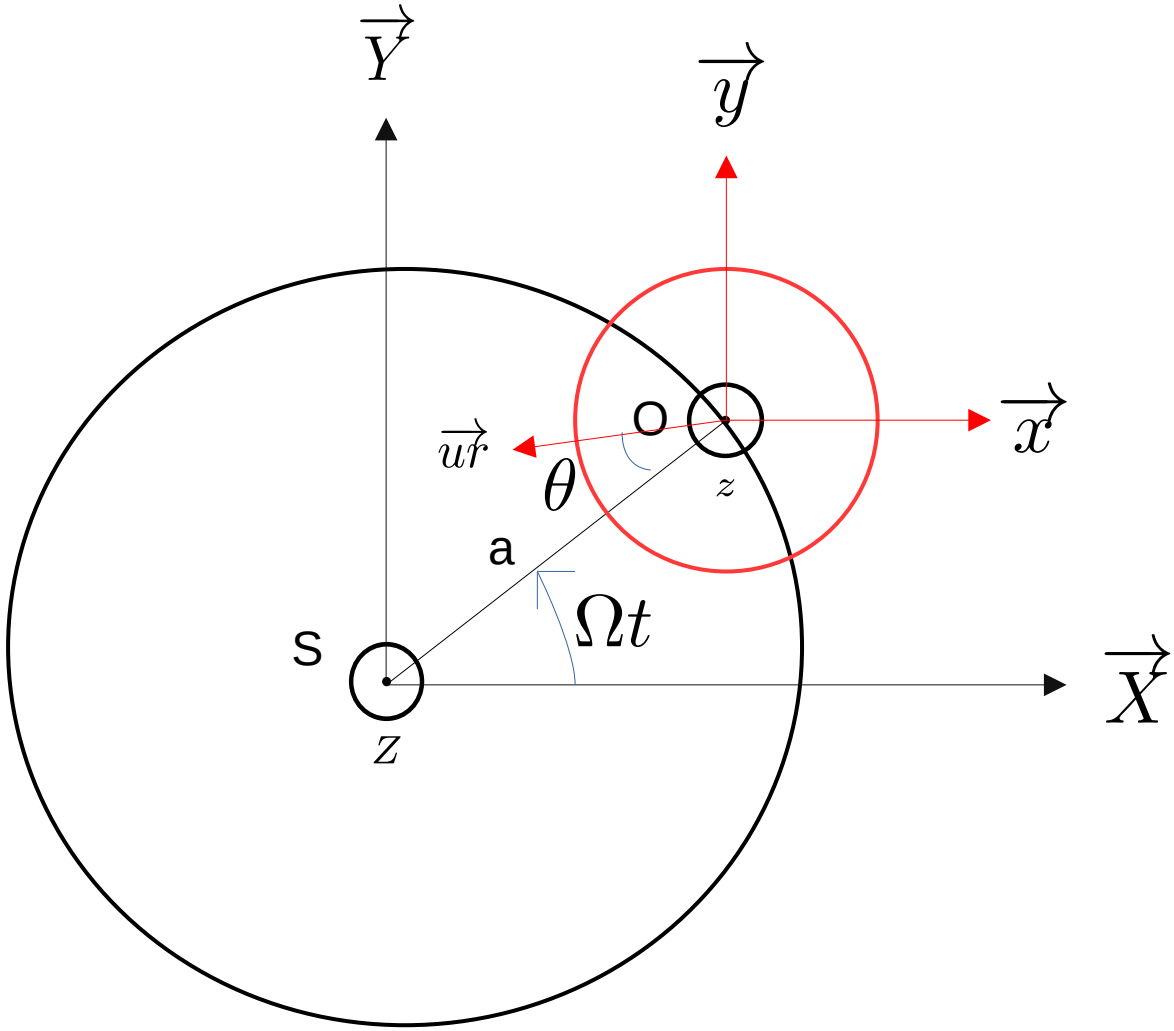
Objectif : calcul de θ en tout point M sur Terre

Référentiel : (SXYZ) héliocentrique

Données géométriques :

- $\theta = (\overrightarrow{ur}, \overrightarrow{OS})$
- $\Omega = \frac{2\pi}{1an}$
- $\overrightarrow{SO} = a \times (\cos(\Omega t)\overrightarrow{X} + \sin(\Omega t)\overrightarrow{Y})$

$$\rightarrow \cos\theta = \frac{\overrightarrow{ur}.\overrightarrow{OS}}{a}$$



▶ 1ère Approximation sans saisons :

- $\overrightarrow{X} = \overrightarrow{x}$
- $\overrightarrow{Y} = \overrightarrow{y}$

→ Approximation insuffisante

⇒ **2ème approximation prise en compte des saisons**

I) Modélisation de l'angle zénithal

- Modélisation de l'angle zénithal :

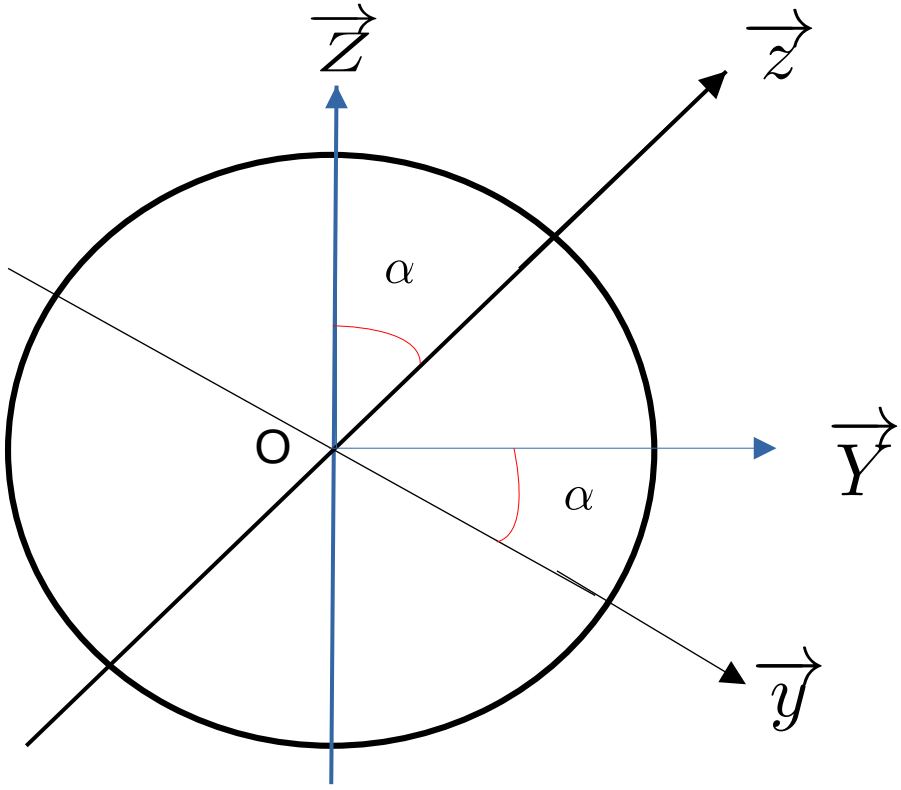
Objectif : calcul de θ en tout point M sur Terre

► 2ème approximation prise en compte des saisons:

α : angle entre le plan de l'écliptique et le plan de l'équateur

Données géométriques :

- $\alpha = 23,4$
- $\overrightarrow{X} = \overrightarrow{x}$
- $\overrightarrow{Y} = \cos(\alpha)\overrightarrow{y} - \sin(\alpha)\overrightarrow{z}$

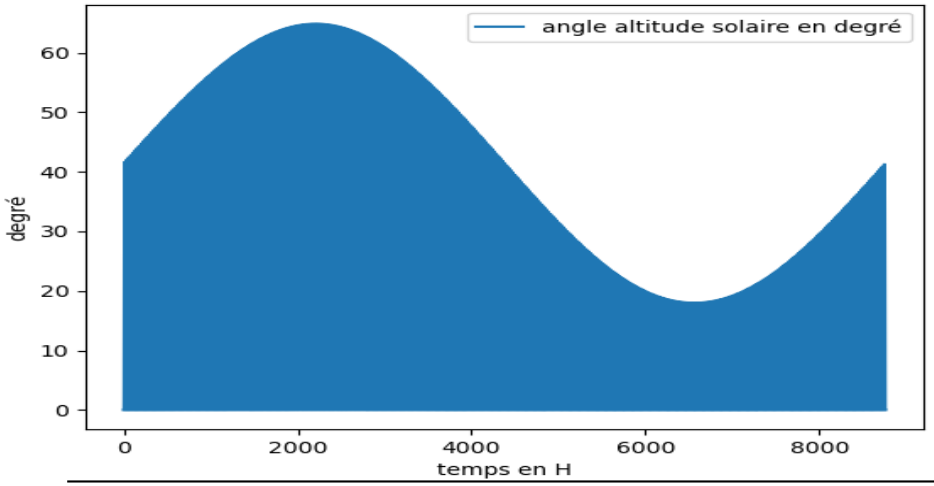


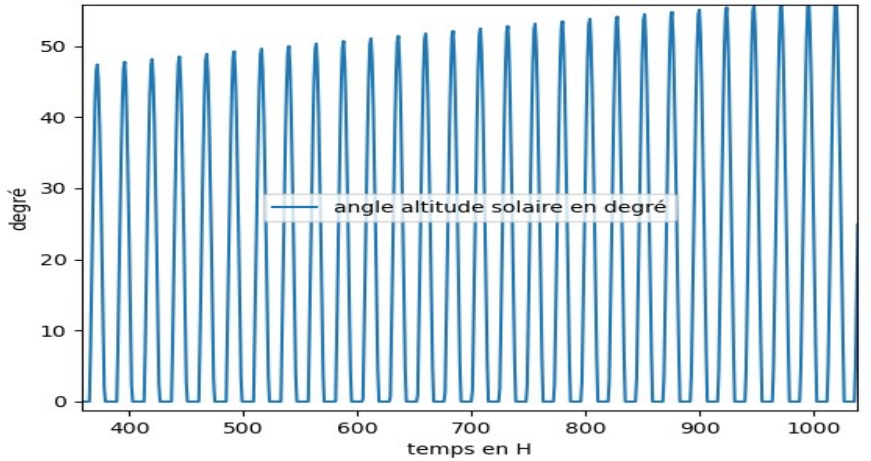
Modélisation Python de θ et δ sur 1 an

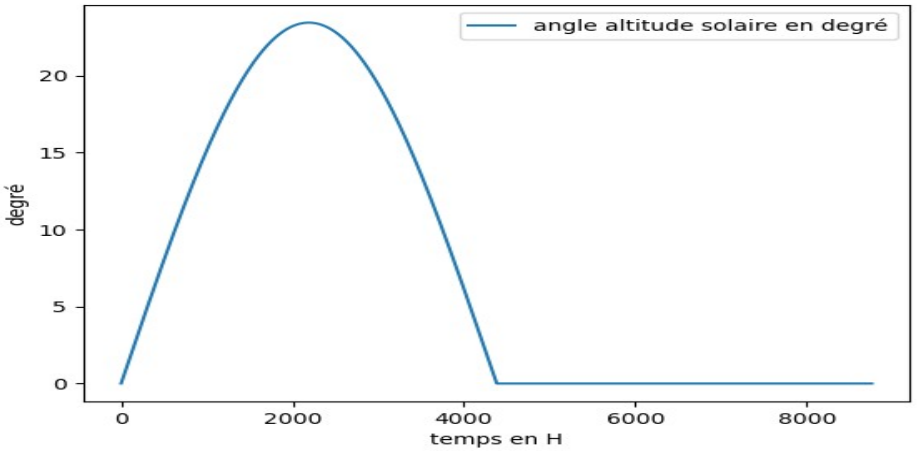
I) Modélisation de l'angle zénithal

- Modélisation de l'angle zénithal :

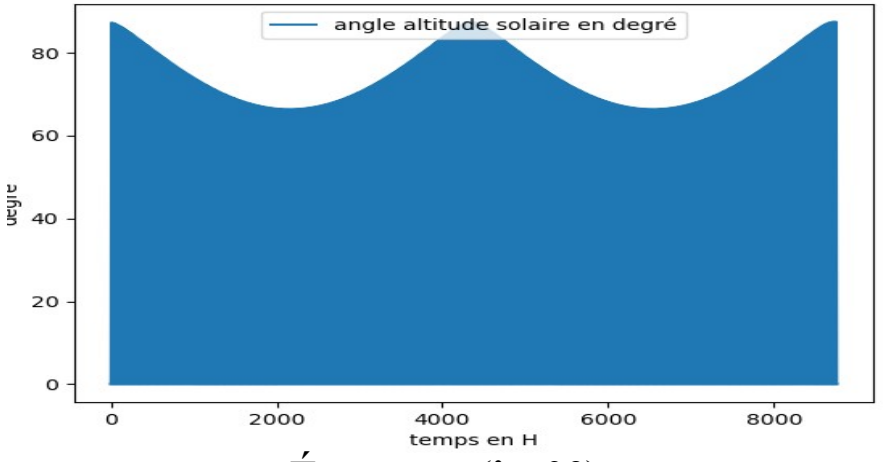
Résultat modélisation à Melun angle δ:







Pôle Nord (λ=90)°



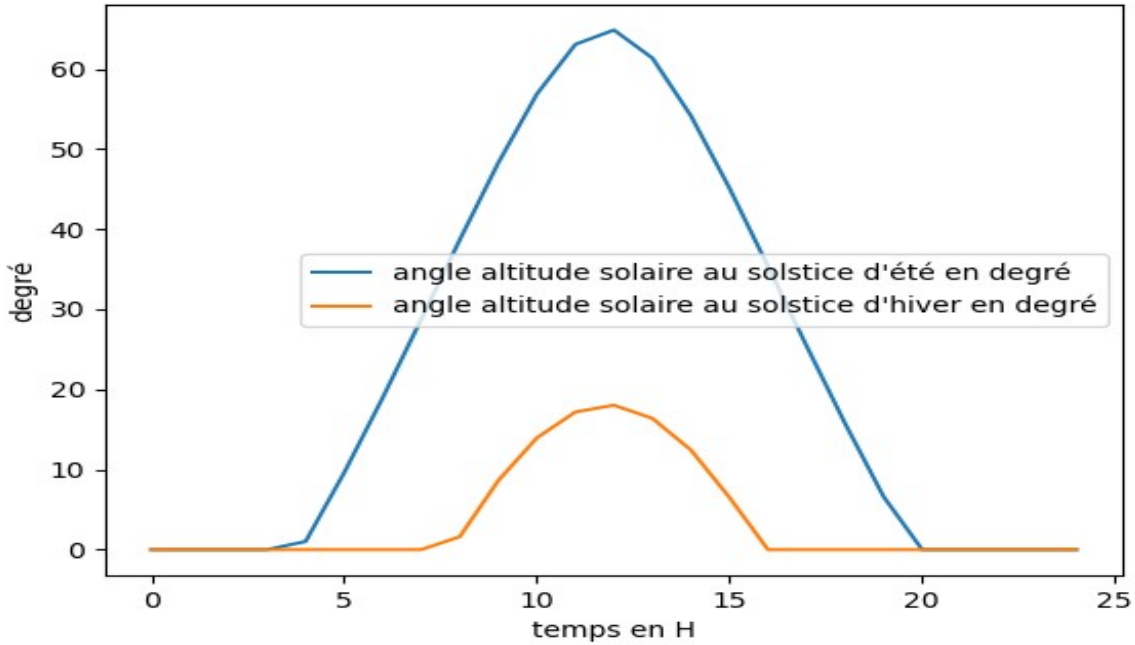
Équateur (λ=0°)

⇨ **Objectif angle zénithal en tout point M atteint**

I) Modélisation de l'angle zénithal

- Modélisation de l'angle zénithal :

Résultat pour Melun aux solstices :



Heure de Greenwich

Réaction chimique cycle de Chapman varie selon les saisons

II) Modélisation cycle de Chapman sans émission, difficultés rencontrées

- Modélisation cycle de Chapman méthode d'Euler : $V(t+dt) = V(t) + dt \times \frac{\partial V}{\partial t}(t)$

Hypothèse de la simulation :

- ▶Vent négligé
- ▶Émission des voitures négligé
- ▶Température supposée constante (298K)
- ▶Temps ensoleillé

Réaction chimique :

$$NO_2 + (O_2) + h\nu \longrightarrow NO + O_3 \quad (1)$$

$$NO + O_3 \longrightarrow NO_2 + O_2 \quad (2)$$

$$O_2 + \dot{O} \rightleftarrows O_3 + h\nu \quad (3)$$

➡ Calcul constantes de vitesse des réactions (1) et (2) :

$$\begin{cases} v1 = JNO_2(t) \times [NO_2] \\ v2 = k \times [NO] \times [O_3] \end{cases}$$

Données réaction (en annexe):
Unités choisie pour la simulation des concentrations molécules.cm-3

On a :

$$JO_3(t) = JO_3 \times \cos(\theta(t))$$
$$JO_2(t) = JO_2 \times \cos(\theta(t))$$
$$JNO_2(t) = JNO_2 \times \cos(\theta(t))$$

II) Modélisation cycle de Chapman sans émission, difficultés rencontrées

- Modélisation cycle de Chapman méthode d'Euler : $$V(t+dt) = V(t) + dt \times \frac{\partial V}{\partial t}(t)$$

Mise en équation sous python méthode d'Euler :

Condition initiale :
-[O_0]=0 molécules.cm-3 (hypothèse)
-[NO_2],[NO],[O_3] : concentration mesurée équinoxe de printemps
-[O_2] : concentration dans l'air

On pose : $$V(t) = \begin{pmatrix} [O_3] \\ [O_2] \\ [O] \\ [NO] \\ [NO_2] \end{pmatrix} \Rightarrow \frac{\partial V}{\partial t}(t) = \begin{pmatrix} \frac{\partial [O_3]}{\partial t} \\ \frac{\partial [O_2]}{\partial t} \\ \frac{\partial [O]}{\partial t} \\ \frac{\partial [NO]}{\partial t} \\ \frac{\partial [NO_2]}{\partial t} \end{pmatrix}$$

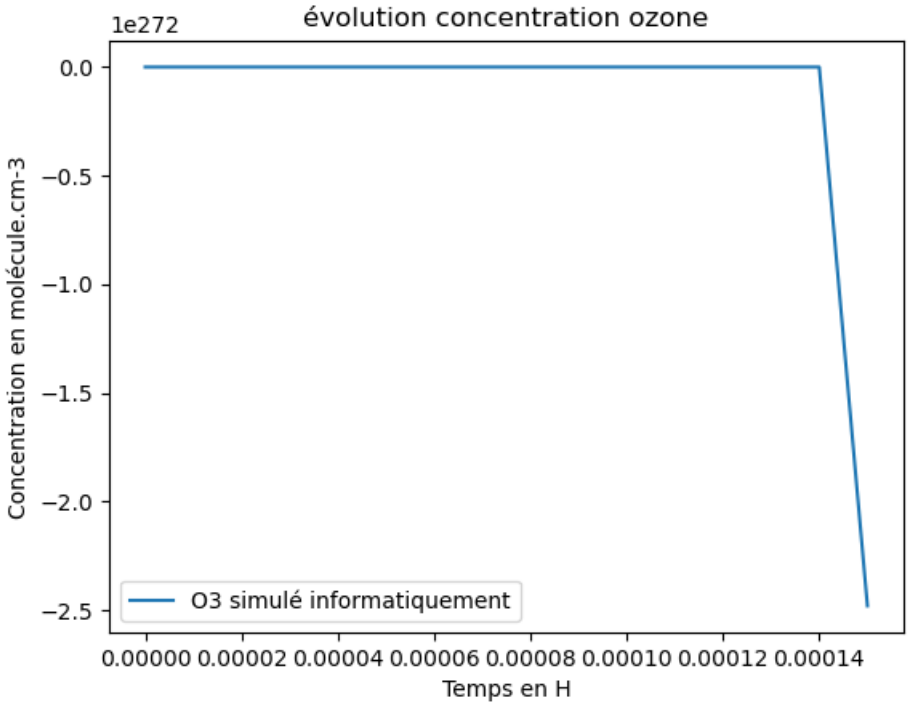
$$= \begin{pmatrix} c \times [M] \times [O_2] \times [O] - JO_3(t) \times [O_3] - d \times [O] \times [O_3] + v1 - v2 \\ a \times [M] \times [O]^2 + JO_3(t) \times [O_3] + 2d \times [O] \times [O_3] - JO_2(t) \times [O_2] - c \times [M] \times [O_2] \times [O] + v2 - v1 \\ -2a \times [M] \times [O]^2 + JO_3(t) \times [O_3] - d \times [O] \times [O_3] + 2JO_2(t) \times [O_2] - c \times [M] \times [O_2] \times [O] \\ v1 - v2 \\ v2 - v1 \end{pmatrix}$$

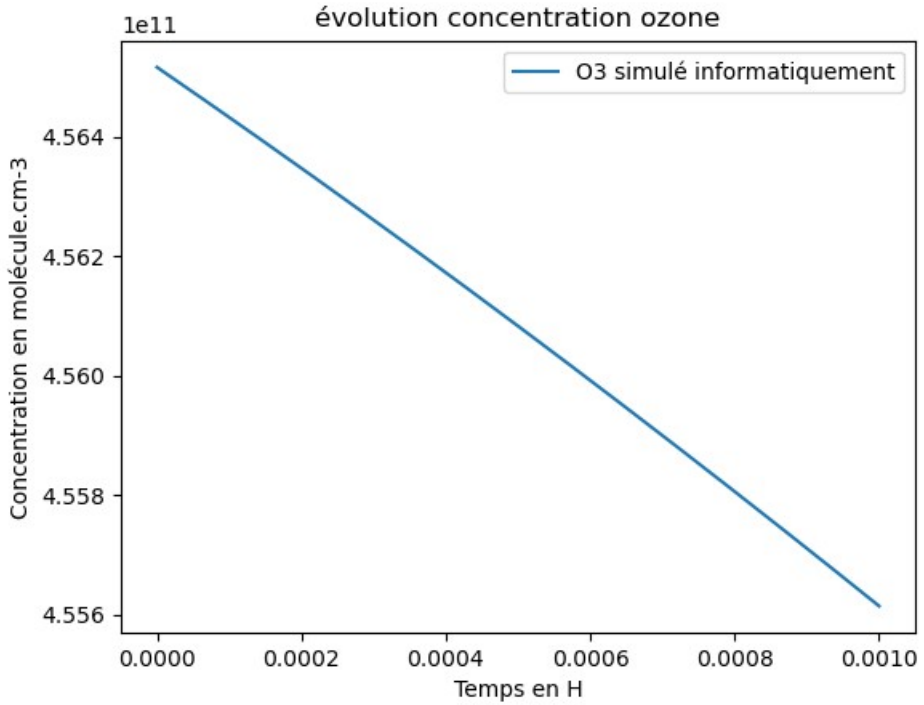
II) Modélisation cycle de Chapman sans émission, difficultés rencontrées

- Divergence de la méthode d'Euler:

$$V(t+dt) = V(t) + dt \times \frac{\partial V}{\partial t}(t)$$

Résultat méthode d'Euler aberrant :





Divergence d'Euler avec un pas de temps de 1/100 000 s.
Raison : Euler = mauvaise approximation

Pas de divergence pour un pas de temps <1/1 000 000 000 s ⇒ trop long temps de calcul (3 min de simulation = 4s simulé)

⇒ Besoin d'utiliser un autre méthode pour notre simulation

II) Modélisation cycle de Chapman sans émission, difficultés rencontrées

- Modélisation cycle de Chapman méthode Runge-Kutta-4:

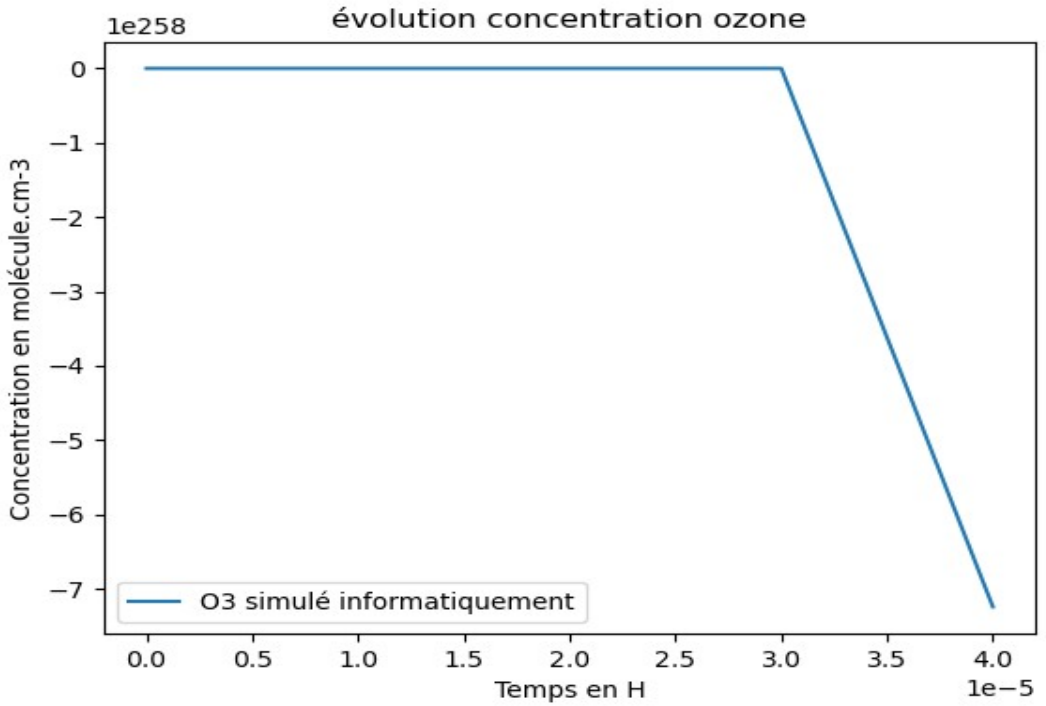
Runge-Kutta 4 : $$V(t+dt) = V(t) + \frac{dt(k1 + 2k2 + 2k3 + k4)}{6}$$

Données :

- Où k1, k2, k3, k4 sont des approximations de :

$$\frac{\partial V}{\partial t}(t)$$

Même hypothèse de simulation qu'avec la méthode d'Euler :



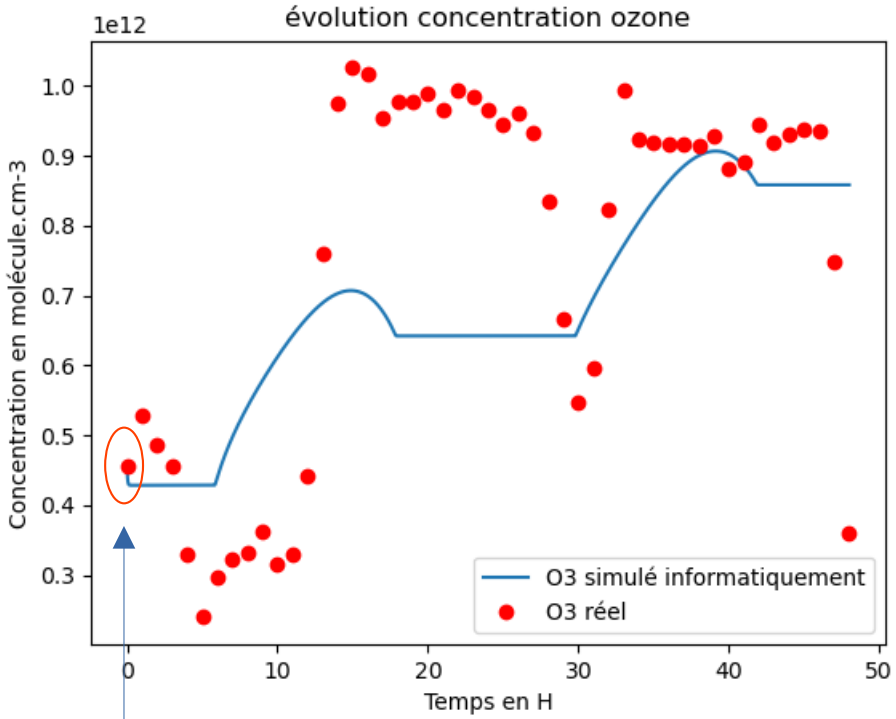
Même problème de divergence qu'avec la méthode d'Euler pour dt > 1/1 000 000 000 s

⇒ **Runge-Kutta = meilleure précision mais insuffisant**

II) Modélisation cycle de Chapman sans émission, difficultés rencontrées

- Modélisation cycle de Chapman en utilisant odeint :

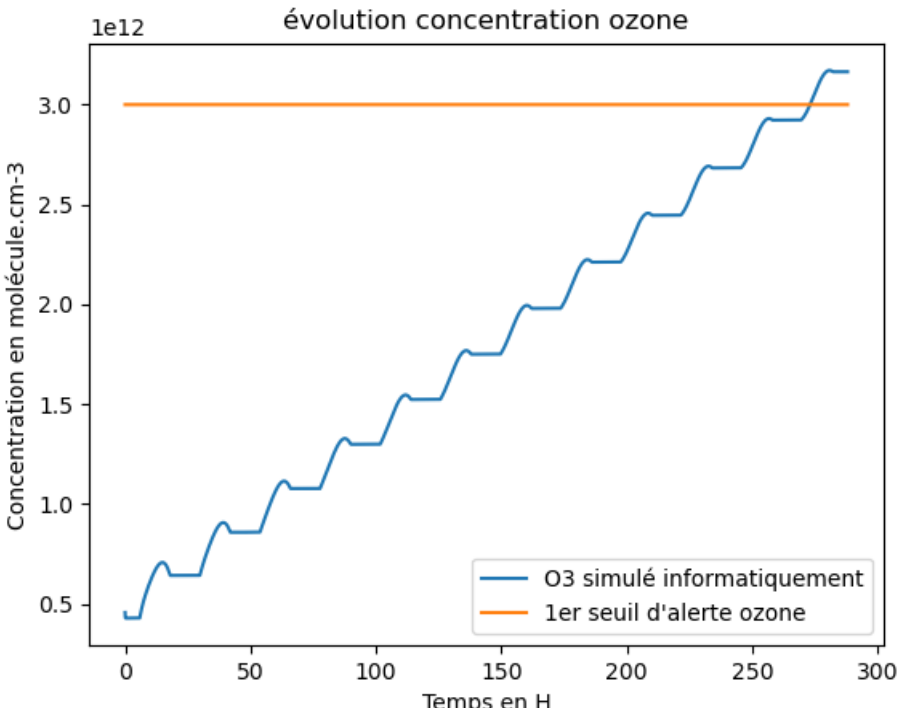
Résultat pour Melun :



- ▶ Pas de divergence
- ▶ [O3] varie avec θ
- ▶ Régime stationnaire la nuit

Variation due aux conditions initiales

1er seuil d'alerte : 240µg.m-3



Seuils d'alerte atteint après 12 jours dans ces conditions

⇨ Résultat du bon ordre de grandeur mais erreur ~ 100 %

III) Modélisation cycle de Chapman avec émission

- Modélisation cycle Chapman voiture 60 % diesel/40 % essence :

Hypothèse polluant :

- Émission constante
- Polluants évoluent dans un volume de 100 m de haut et surface petite couronne Ile de France (657 km carré)

→ $V = 6,57 \times 10^{10} m^3$

IDF chaque jour:

~40 % voiture essence
~60 % voiture diesel
~4 millions de voiture
~30 km de trajet par voiture

Émission (mg.km-1)	Essence	Diesel
NO2	40	140
NO	1	40

Données parc INRETS

►Émission quotidienne :

- 180µg.m-3 NO2
- 40µg.m-3 de NO

→ eno : terme d'émission de NO
eno2 : terme d'émission de NO2

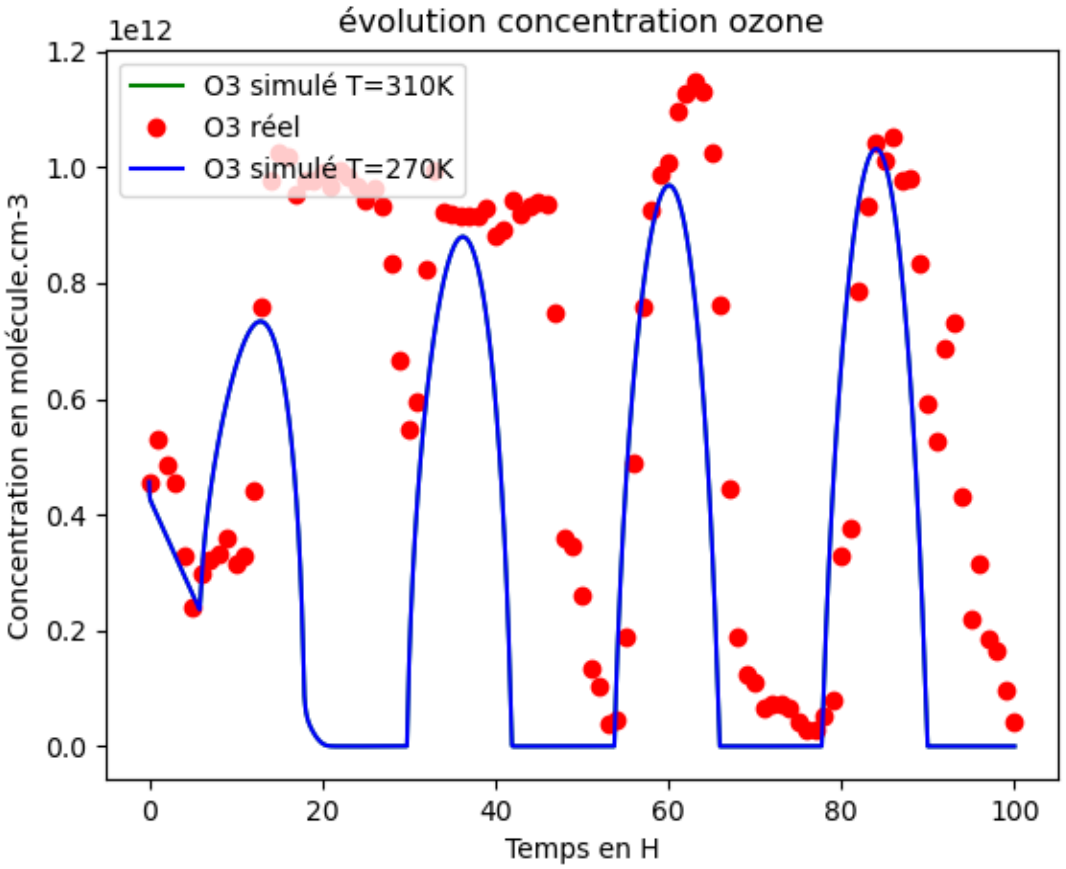
On injecte dans notre système d'équation : $\frac{\partial V}{\partial t}(t) =$

$$\begin{pmatrix} c \times [M] \times [O_2] \times [O] - JO_3(t) \times [O_3] - d \times [O] \times [O_3] + v1 - v2 \\ a \times [M] \times [O]^2 + JO_3(t) \times [O_3] + 2d \times [O] \times [O_3] - JO_2(t) \times [O_2] - c \times [M] \times [O_2] \times [O] + v2 - v1 \\ -2a \times [M] \times [O]^2 + JO_3(t) \times [O_3] - d \times [O] \times [O_3] + 2JO_2(t) \times [O_2] - c \times [M] \times [O_2] \times [O] \\ v1 - v2 + eno \\ v2 - v1 + eno2 \end{pmatrix}$$

III) Modélisation cycle de Chapman avec émission

- Modélisation cycle Chapman voiture 60 % diesel/40 % essence :

Comparaison réel/simulation :



Résultat :

- Jour 3/4 : bonne approximation
- Déphasage car heure de Greenwich
- Même résultat pour T=270 K ou T=310 K ⟹ T supposé constant : bonne approximation

Source d'erreur :

- Vent
- Émission non constante

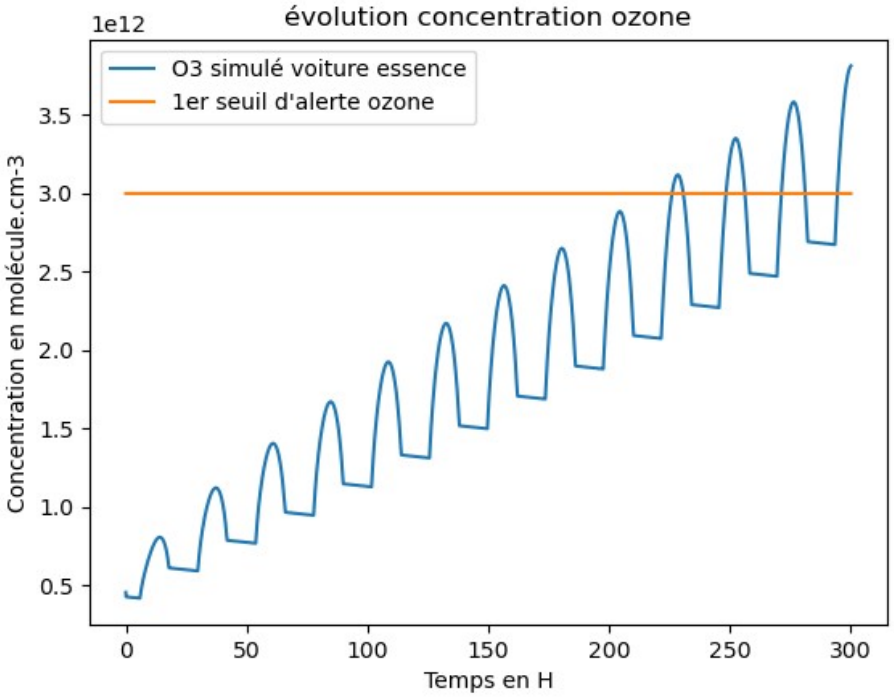
III) Modélisation cycle de Chapman avec émission

- Modélisation cycle Chapman voiture 100 % diesel/100 % essence :

Cas voiture 100 % essence :

►Émission quotidienne :

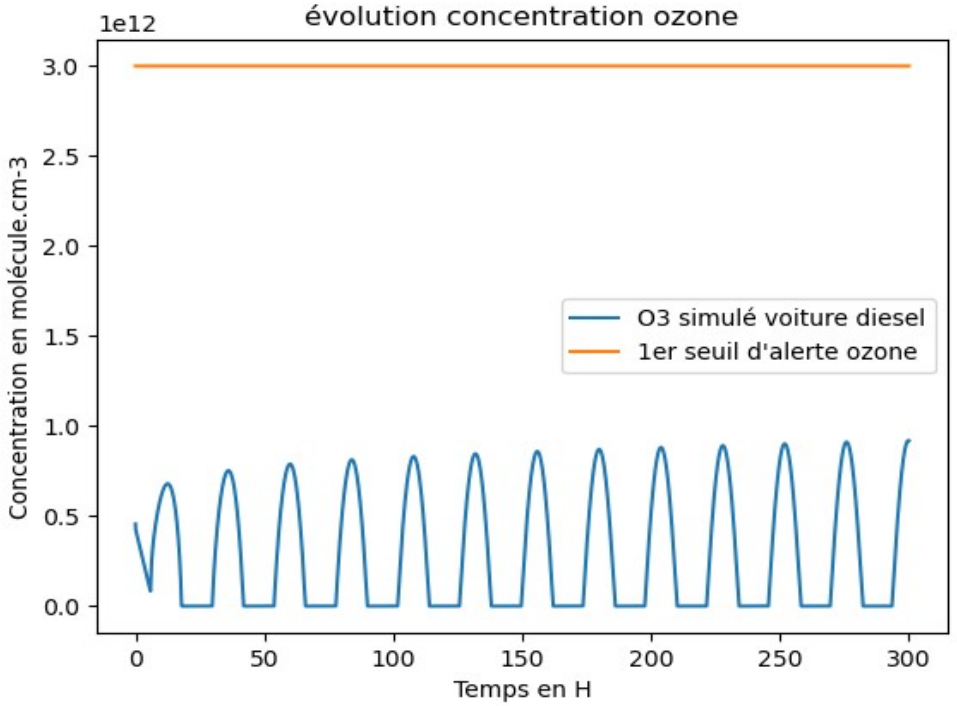
- 73µg.m-3 NO_2
- 2µg.m-3 de NO



Cas voiture 100 % diesel :

►Émission quotidienne :

- 255µg.m-3 NO_2
- 73µg.m-3 de NO

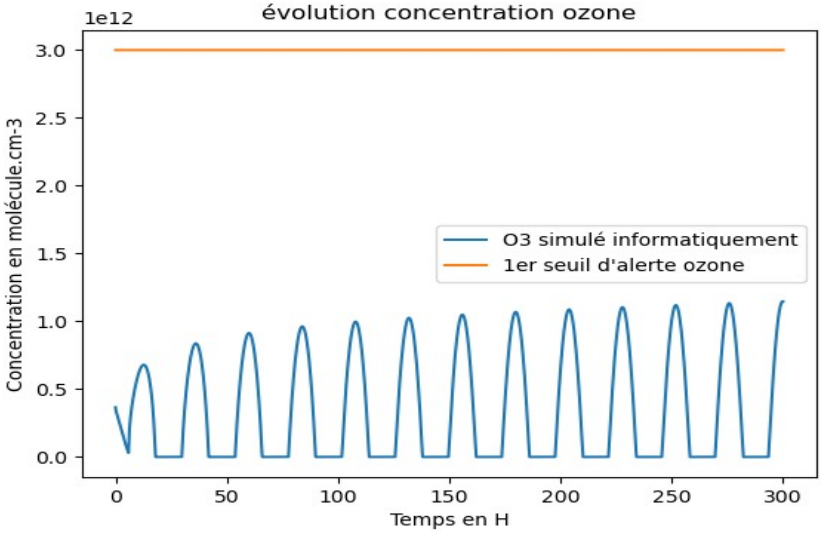


→ Voitures diesel = mieux pour éviter les pics de pollution à l'ozone mais émet plus de particule fine

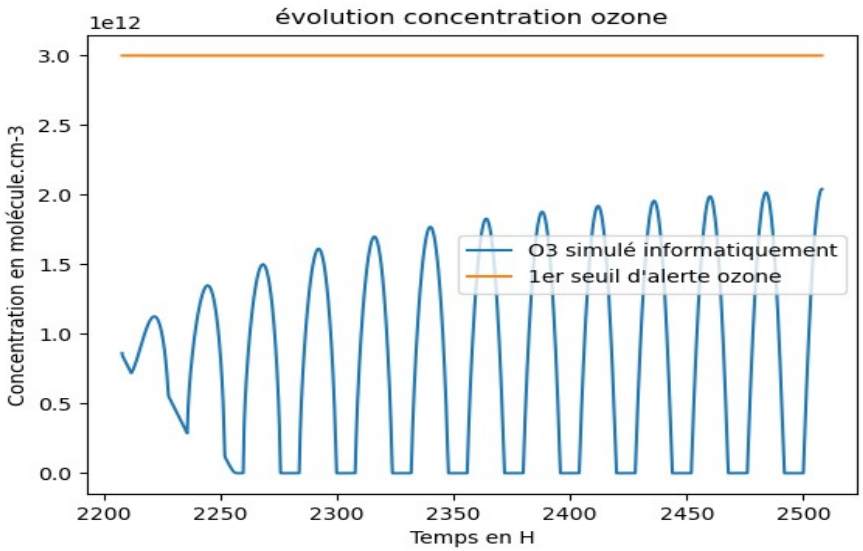
III) Modélisation cycle de Chapman avec émission

- Modélisation cycle Chapman cas réel en fonction des saisons :

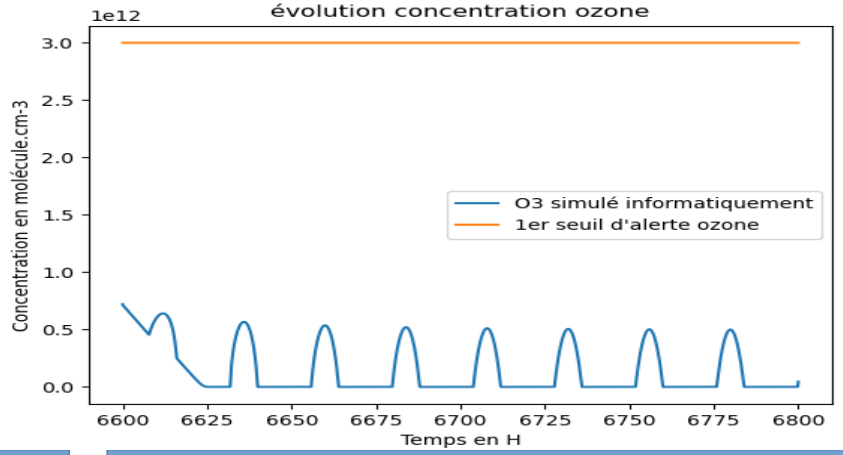
Équinoxe de printemps



Solstice d'été



Solstice d'hiver



Conclusion :

Pic pollution à l'ozone favorisé par beau temps en été

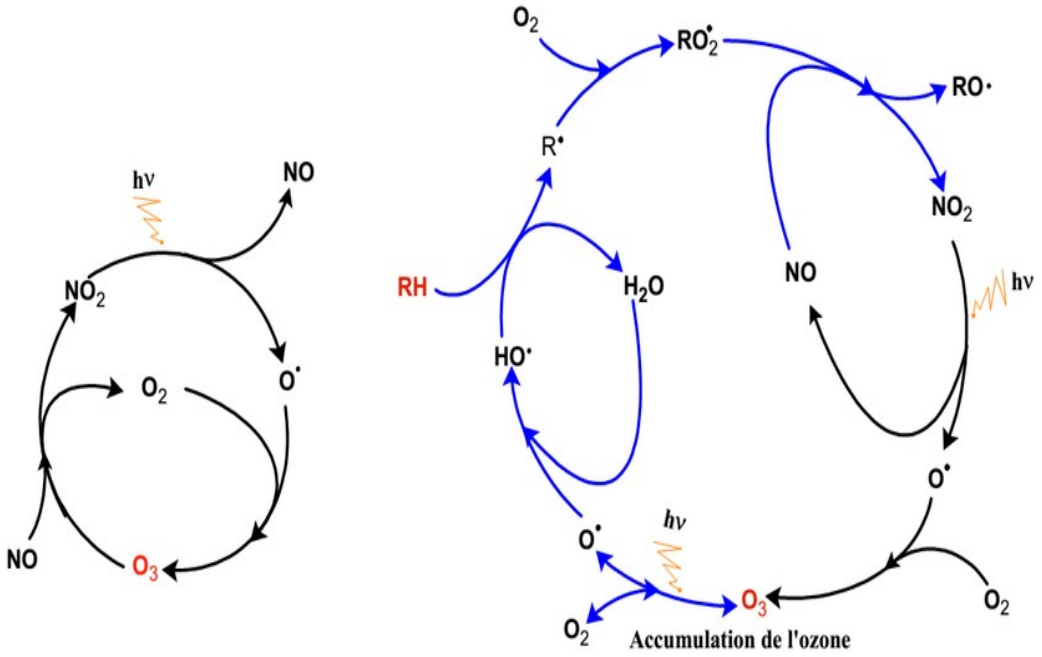
Seuil d'alerte jamais atteint

➡ Limite de notre modélisation

CONCLUSION :

Limite du modèle :

- Vent négligé
- Temps supposé toujours ensoleillé
- Émission supposée constante
- Présence d'autres sources d'émission : le chauffage...
- Cycle de Chapman modélisé sans COV
- Émission non uniformément répartie dans l'espace



a) Cycle de Chapman sans COV

b) En présence de COV=RH

Source: researchgate.net

ANNEXE :

- Programme python :

```python
### TIPE simulation de la pollution à l'ozone à Melun

## Lecture de fichier données expérimentales de la station météo de Melun-Villaroche grâce à Airparif

import matplotlib.pyplot as plt

plt.close('all')

fichier = open("C:\\Users\\aches\\Documents\\TIPE\
\TIPE_mesure.txt","r")

NO=[]
NO2=[]
O3=[]
Dates=[]
for Ligne in fichier:
    L=Ligne.split(",")
    if len(L)==5:
        if L[1]=='': # test en cas d'absence de données car station météo en panne sur une tranche horaire, dans ce cas la plage horaire ne seras pas utilisée dans la simulation
            NO2.append(0)
        else:
            NO2.append(L[1])
        if L[2]=='':
            NO.append(0)
        else:
            NO.append(L[2])
        if L[3]=='':
            O3.append(0)
        else:
            O3.append(L[3])
        Dates.append(L[0])

D=[]
H=[]
for k in range(len(Dates)):
    Date=Dates[k].split()
    D.append(Date[0])
    H.append(Date[1])

H1=[]
for k in range(len(H)):
    Temps=H[k].split(":")
    H1.append(float(Temps[0]))

Jour=[]
Mois=[]
for k in range(len(D)):
    J=D[k].split("/")
    Jour.append(int(J[2]))
    Mois.append(int(J[1]))


M=[0,31,31+28,31+28+31,31+28+30+31,31+28+30+31+31,31+28+30+31+30+31,31+28+30+31+30+31+31,31+28+30+31+30+31+31+31,31+28+30+31+30+31+31+31+30,31+28+30+31+30+31+31+31+30+31,31+28+30+31+30+31+31+31+30+31+30,31+28+30+31+30+31+31+31+30+31+30+31] #jour en plus du mois i

H2=[]
for k in range(len(Jour)):
    i=Mois[k]
    H=[24*(float(Jour[k])-1)+H1[k]+24*(M[i-1])]
    H2.append(H)

CNO2=[]
CNO=[]
CO3=[]
H3=[]
for k in range(len(NO2)):
    CNO2.append(float(NO2[k]))
    CNO.append(float(NO[k]))
    CO3.append(float(O3[k]))
    H3.append(H2[k])

def f_affiche(fig_i,X,Y,legende,abscisse,ordonnée,titre):
    plt.figure(fig_i)
    plt.plot(X,Y,label=legende)
    plt.xlabel(abscisse)
    plt.ylabel(ordonnée)
    plt.title(titre)
    plt.legend()
    plt.show()

```

ANNEXE :

- Programme python :

```
f_affiche(0,H3,CNO2,"NO2","Temps","concentration en mg.m^-3","")
f_affiche(0,H3,CNO,"NO","Temps","concentration en mg.m^-3","")
f_affiche(0,H3,CO3,"O3","Temps en H","concentration en mg.m^-3","Evolution concentration ozone")

## Impact sanitaire en ODG de la pollution à l'ozone sur la ville de Melun et l'Ile de France en utilisant les données de l'INSEE

import matplotlib.pyplot as plt

plt.close('all')

patient_malade=0
patient_mort=0
Mal=[]
Mort=[]
for k in range(len(CO3)):
    mal=(0.56+0.198-0.259)*4*CO3[k]/(1000*24)  #(0.56+0.198-0.259) pour 10 millions d'habitants par jour or il y a 40000 habitants à Melun
    mort=(-0.066+0.124+0.061)*4*CO3[k]/(1000*24)
    patient_malade+=mal
    patient_mort+=mort
    Mal.append(patient_malade)
    Mort.append(patient_mort)

patient_malade_idf=0
patient_mort_idf=0
Mal_idf=[]
Mort_idf=[]
for k in range(len(CO3)):
    mal=(0.56+0.198-0.259)*1.22*CO3[k]/24  #(0.56+0.198-0.259) pour 10 millions d'habitants par jour or il y a 12,2 millions d'habitants en IDF
    mort=(-0.066+0.124+0.061)*1.22*CO3[k]/24
    patient_malade_idf+=mal
    patient_mort_idf+=mort
    Mal_idf.append(patient_malade_idf)
    Mort_idf.append(patient_mort_idf)

# 660361 morts en France en 2021 en France selon l'INSEE sur 67.4 millions d'habitants
# 12.9 millions d'admisions à l'hôpital en France toutes causes confondues en 2021

mort_IDF=660361*12.2/67.4 # Calcul moyenné par rapport à la population en Ile de France avec 12.2 millions d'individus
mort_Melun=660361*4/6740
admission_IDF=12900000*10/67.4
admission_Melun=12900000*4/6740
m_idf=mort_IDF/len(CO3) #mort par heure en moyenne
m_melun=mort_Melun/len(CO3)
a_idf=admission_IDF/len(CO3) #admission par heure en moyenne
a_melun=admission_Melun/len(CO3)
M_IDF=[]
M_Melun=[]
A_IDF=[]
A_Melun=[]
for k in range(len(CO3)):
    m_i=k*m_idf # nombre de mort au bout de k heures
    m_m=k*m_melun
    a_i=k*a_idf
    a_m=k*a_melun
    M_IDF.append(m_i)
    M_Melun.append(m_m)
    A_IDF.append(a_i)
    A_Melun.append(a_m)

# Melun

print("patient malade et hospitalisé à cause de l'ozone à Melun en 2021:",int(patient_malade))
print("patient mort à cause de l'ozone à Melun en 2021:",int(patient_mort))
print("admission hôpital à Melun toutes causes confondues en 2021':",int(admission_Melun))
print("taux d'hospitalisation à cause de la pollution à l'ozone à Melun",(patient_malade/admission_Melun)*100,"%")
print("décès toutes causes confondues à Melun en 2021:",int(mort_Melun))
print("taux de décès à cause de la pollution à l'ozone à Melun en 2021",(patient_mort/mort_Melun)*100,"%")

f_affiche(1,H3,Mal,"admission hôpital à Melun à cause de l'ozone en 2021","Temps en H","nombre de personnes","Evolution sanitaire pour la ville de Melun")
f_affiche(1,H3,Mort,"décès à Melun à cause de l'ozone en 2021","Temps en H","nombre de personnes","Evolution sanitaire pour la ville de Melun")
```

ANNEXE :

- Programme python :

```
#f_affiche(1,H3,A_Melun,"admission hôpital à Melun en 2021","H","nombre de personnes","Evolution sanitaire pour la ville de Melun")
#f_affiche(1,H3,M_Melun,"décès à Melun en 2021","H","nombre de personnes","Evolution sanitaire pour la ville de Melun")

# IDF

print("patient malade et hospitalisé à cause de l'ozone en IDF en 2021:",int(patient_malade_idf))
print("patient mort à cause de l'ozone en IDF en 2021:",int(patient_mort_idf))
print("admission hôpital en IDF toutes causes confondues en 2021':",int(admission_IDF))
print("taux d'hospitalisation à cause de la pollution à l'ozone en IDF",(patient_malade_idf/admission_IDF)*100,"%")
print("décès toutes causes confondues en IDF en 2021:",int(mort_IDF))
print("taux de décès à cause de la pollution à l'ozone en IDF en 2021",(patient_mort_idf/mort_IDF)*100,"%")

#f_affiche(1,H3,Mal_idf,"admission hôpital en IDF à cause de l'ozone en 2021","H","","")
#f_affiche(1,H3,Mort_idf,"décès en IDF à cause de l'ozone en 2021","H","","")
#f_affiche(1,H3,A_IDF,"admission hôpital en IDF en 2021","H","","")
#f_affiche(1,H3,M_IDF,"décès en IDF en 2021","H","nombre de personnes","Evolution sanitaire en IDF")

## Influence des différents paramètres météorologiques (Température, ensolleillement) sur la concentration d'ozone à Melun en Janvier 2021

import matplotlib.pyplot as plt

plt.close('all')

H4=[H3[k]for k in range(745)] # Liste des heures au mois de Janvier
kO3=[CO3[k]/6 for k in range(745)] # calibrage pour mieux voir l'influence des paramètres sur la courbe
plt.figure(2)
plt.plot(H4,kO3,'y',label="concentration d'ozone")
plt.show()
#f_affiche(2,H4,kO3,"concentration d'ozone","heure de prélévement","concentration d'ozone en mg.m^3","évolution de la concentration d'ozone en fonction du temps à Melun à partir du 1er janvier 2021")


# Impact de la Température

fichier=open("C:\\Users\\aches\\.conda\\TIPE_meteo.txt","r")
Tmax=[]
Tmin=[]
Tmoy=[]
Jour=[]
for Ligne in fichier:
    L=Ligne.split()
    if len(L)==7:
        for k in range(24):
            Tmax.append(float(L[2]))
            Tmin.append(float(L[1]))
            Tmoy.append(float(L[3]))
            Jour.append(float(L[0])*24+(1/24)*k)

#f_affiche(2,Jour,Tmax,"Température max","","","")
#f_affiche(2,Jour,Tmin,"Température min","","","")
f_affiche(2,Jour,Tmoy,"Température moyenne","jour de janvier 2021","Température (°C)  ensoleillemnt (H)","évolution des températures à Melun en janvier 2021")

# Impact de l'ensoleillement

fichier=open("C:\\Users\\aches\\.conda\\TIPE_meteo.txt","r")

Sh=[]
Sm=[]
Jour=[]
for Ligne in fichier:
    L=Ligne.split()
    if len(L)==7:
        for k in range(24):
            Sh.append(L[5])
            Sm.append(L[6])
            Jour.append(float(L[0])*24+(1/24)*k)

Sh1=[]
Sm1=[]
```

ANNEXE :

- Programme python :

```
for k in range(len(Sh)):
    heure=Sh[k].split("h")
    minute=Sm[k].split("min")
    Sh1.append(float(heure[0]))
    Sm1.append(float(minute[0]))

S=[]
for k in range(len(Sh1)):
    s=Sh1[k]*60+Sm1[k]
    S.append(s/60)

plt.figure(2)
plt.plot(Jour,S,'ro',label='heure ensoleillement en 1 journée')
plt.legend()
plt.show()

## Calcul de l'angle zénithal théta au cours du temps à Melun, 1ère approximation sans tenir compte des saisons référentiel géocentrique

import numpy as np
from math import cos,sin,pi
from matplotlib import pyplot as plt
plt.close('all')

#Données:

latitude=45*pi/180 # en radians
longitude=2.66*pi/180 # en radians
w=2*pi/86164 # pulsation rotation de la Terre
omega=2*pi/(365*60*60*24) # pulsation de révolution de la Terre autour du soleil
T=np.linspace(0,8759,8760) # Liste des temps en H à partir de 12H heure de Greenwich à l'équinoxe de printemps

def phi(T):
    n=len(T)
    phit=[]
    for k in range(n):
        phi=w*T[k]*60*60+longitude
        phit.append(phi)
    return phit

def teta(T):
    tetat=[]
    n=len(T)
    phit=phi(T)
    for k in range(n):
        ur=[cos(latitude)*cos(phit[k]),cos(latitude)*sin(phit[k]),sin(latitude)]
        us=[-cos(omega*T[k]*60*60),-sin(omega*T[k]*60*60),0]
        pscalaire=ur[0]*us[0]+ur[1]*us[1]+ur[2]*us[2]
        teta=pscalaire # angle en radians
        tetat.append(teta)
    return tetat

tetat=teta(T)
plt.figure(2)
plt.plot(T,tetat,label="angle théta en radian")
plt.xlabel("Temps en heures")
plt.ylabel("radians")
plt.legend()
plt.show()

## Calcul de l'angle zénithal théta au cours du temps à Melun, 2ème approximation prise en compte des saisons en référentiel héliocentrique

import numpy as np
from math import cos,sin,pi,acos
from matplotlib import pyplot as plt
plt.close('all')

#Données:

latitude=48.5*pi/180 # en radians
longitude=2.66*pi/180 # en radians
w=2*pi/86164 # pulsation rotation de la Terre en radians.s-1
alpha0=23.45*pi/180 # angle plan écliptique/plan équatorial en radians
omega=2*pi/(365*24) # pulsation de révolution de la Terre autour du soleil en radians.h-1


def fphi(t):
    return w*t*60*60+longitude
```

ANNEXE :

- Programme python :

```
## Méthode d'Euler cycle de Chapman soleil uniquement sans polluant

plt.close('all')
import numpy as np
from math import cos,sin,pi,acos,exp
from matplotlib import pyplot as plt

#Données:

Tp=298 # Température en K
Na=6.02*10**23 # en mol-1
a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
jO3=3*10**-5 # en s-1
jO2=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
x0=0*Na*10**-12*CO3[1896]/48 # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
y0=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène de l'air
z0=0 #hypothèse pas d'oxygène atomique initialement
V0=np.array([x0,y0,z0])

# Simulation

def fJO3(t):
    return jO3*cos(fteta(t))

def fJO2(t):
    return jO2*cos(fteta(t))

def F(V,t):
    x,y,z=V
    dx=(c*M*y*z-fJO3(t)*x-d*z*x)*3600
    dy=(a*M*z**2+2*d*z*x+fJO3(t)*x-fJO2(t)*y-c*M*y*z)*3600
    dz=(2*fJO2(t)*y+fJO3(t)*x-c*M*y*z-2*a*z**2*M-d*z*x)*3600
#Divergence d'Euler a cause du terme c*M*y*z (pas de temps trop grand)
    return np.array([dx,dy,dz])

def fteta(t):
    ur=[cos(latitude)*cos(fphi(t)),cos(latitude)*sin(fphi(t)),sin(latitude)]
    us=[-cos(omega*t),-cos(alpha0)*sin(omega*t),sin(alpha0)*sin(omega*t)]
    pscalaire=ur[0]*us[0]+ur[1]*us[1]+ur[2]*us[2]
    costeta=pscalaire # cosinus angle zénithal en radians
    return acos(costeta*(costeta>=0))

def falt(t):
    ur=[cos(latitude)*cos(fphi(t)),cos(latitude)*sin(fphi(t)),sin(latitude)]
    us=[-cos(omega*t),-cos(alpha0)*sin(omega*t),sin(alpha0)*sin(omega*t)]
    pscalaire=ur[0]*us[0]+ur[1]*us[1]+ur[2]*us[2]
    costeta=pscalaire # cosinus angle zénithal en radians
    return ((pi/2)-acos(costeta*(costeta>=0)))*180/pi

# Affichage:

T=np.linspace(0,8759,8760) # Liste des temps en H à partir de 12H heure de Greenwich à l'équinoxe de printemps
teta=[fteta(t) for t in T]
alti=[falt(t) for t in T]
shiver=[falt(t) for t in range(6600,6625)] #jour solstice d'hiver
sete=[falt(t) for t in range(2208,2233)] #jour solstice d'été
TH=[i for i in range(25)]

plt.close('all')
plt.figure(3)
#plt.plot(T,teta,label="angle zénithal en radians")
#plt.plot(T,alti,label="angle altitude solaire en degré")
plt.plot(TH,sete,label="angle altitude solaire au solstice d'été en degré")
plt.plot(TH,shiver,label="angle altitude solaire au solstice d'hiver en degré")
plt.xlabel("temps en H")
plt.ylabel("degré")
plt.legend()
plt.show()
```

ANNEXE :

- Programme python :

```python
def Euler_Explicite(f,y0,t0,dt,t1):
    y=y0
    Y=[y]
    t=t0
    T=[t]
    while t<t1:
        t+=dt
        yp=f(y,t)
        y=y+yp*dt
        Y.append(y)
        T.append(t)
    return T,Y

# Résolution

t0=0
dt=1/100000 # dt=1/100000000 pour éviter la divergence d'euler mais temps de calcul énorme (3 min pour 4s de simulation)
t1=12
T1,Y=Euler_Explicite(F,V0,t0,dt,t1)
Y=np.array(Y)
LO3=Y[:,0]
LO2=Y[:,1]
LO=Y[:,2]
LJO3=[fJO3(t) for t in T1]
plt.figure(4)
plt.plot(T1,LO3,label="O3")
#plt.plot(T1,LJO3,label="JO3")
plt.legend()
plt.show()

## Méthode d'Euler cycle de Chapman avec polluant sans émission des voitures

plt.close('all')
import numpy as np
from math import cos,sin,pi,acos,exp
from matplotlib import pyplot as plt

# Données:

Tp=298 # Température en K
Na=6.02*10**23 # en mol-1
a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
k=1.8*10**-14 # en cm3.molécules-1.s-1
M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
jO3=3*10**-5 # en s-1
jO2=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
jNO2=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
o30=Na*10**-12*CO3[1896]/48 # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène dans l'air
o0=0 #hypothèse pas d'oxygene atomique initialement dans l'air
no0=Na*10**-12*CNO[1896]/30  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
no20=Na*10**-12*CNO2[1896]/46  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
V0=np.array([o30,o20,o0,no0,no20]) # condition initiale

# Simulation:

def fJO3(t):
    return jO3*cos(fteta(t))

def fJO2(t):
    return jO2*cos(fteta(t))

def fJNO2(t):
    return jNO2*cos(fteta(t))

def F(V,t):
    o3,o2,o,no,no2=V
    v1=fJNO2(t)*no2
    v2=no*o3*k
    do3=(c*M*o2*o-fJO3(t)*o3-d*o*o3+v1-v2)*3600
    do2=(a*M*o**2+2*d*o*o3+fJO3(t)*o3-fJO2(t)*o2-c*M*o2*o-v1+v2)*3600
    do=(2*fJO2(t)*o2+fJO3(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600 # Divergence d'euler pas de temps trop grand
    dno=(v1-v2)*3600
    dno2=(v2-v1)*3600
    return np.array([do3,do2,do,dno,dno2])
```

ANNEXE :

- Programme python :

```
442
443 def Euler_Explicite(f,y0,t0,dt,t1):
444     y=y0
445     Y=[y]
446     t=t0
447     T=[t]
448     while t<t1:
449         t+=dt
450         yp=f(y,t)
451         y=y+yp*dt
452         Y.append(y)
453         T.append(t)
454     return T,Y
455
456 # Résolution
457
458 t0=0
459 dt=1/100000
460 t1=12
461 T1,Y=Euler_Explicite(F,V0,t0,dt,t1)
462 Y=np.array(Y)
463 LO3=Y[:,0]
464 LO2=Y[:,1]
465 LO=Y[:,2]
466 LNO=Y[:,3]
467 LNO2=Y[:,4]
468
469 plt.figure(5)
470 plt.plot(T1,LO3,label="O3 simulé informatiquement")
471 #plt.plot(T1,LO2,label="O2 simulation")
472 #plt.plot(T1,LO,label="O simulation")
473 #plt.plot(T1,LNO2,label="NO2 simulation")
474 #plt.plot(T1,LNO,label="NO simulation")
475 plt.xlabel("Temps en H")
476 plt.ylabel("Concentration en molécule.cm-3")
477 plt.title("évolution concentration ozone")
478 plt.legend()
479 plt.show()
480
481 ## Méthode d'Euler cycle de Chapman avec polluant sans émission des
    voitures
```

```
483 plt.close('all')
484 import numpy as np
485 from math import cos,sin,pi,acos,exp
486 from matplotlib import pyplot as plt
487
488 # Données:
489
490 Tp=298 # Température en K
491 Na=6.02*10**23 # en mol-1
492 a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
493 c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
494 d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
495 k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
496 M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
497 jO3=3*10**-5 # en s-1 pour un angle zénithal de 0°
498 jO2=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
499 jNO2=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
500 o30=Na*10**-12*CO3[1896]/48 # concentration en molécules.cm-3 à
    l'équinoxe du printemps (20 mars 2021)
501 o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène
    dans l'air
502 o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans
    l'air
503 no0=Na*10**-12*CNO[1896]/30  # concentration en molécules.cm-3 à
    l'équinoxe du printemps (20 mars 2021)
504 no20=Na*10**-12*CNO2[1896]/46  # concentration en molécules.cm-3 à
    l'équinoxe du printemps (20 mars 2021)
505 V0=np.array([o30,o20,o0,no0,no20]) # condition initiale
506
507 # Simulation:
508
509 def fJO3(t):
510     return jO3*cos(fteta(t))
511
512 def fJO2(t):
513     return jO2*cos(fteta(t))
514
515 def fJNO2(t):
516     return jNO2*cos(fteta(t))
517
```

ANNEXE :

- Programme python :

```
518 def F(V,t):
519     o3,o2,o,no,no2=V
520     v1=fJNO2(t)*no2
521     v2=no*o3*k
522     do3=(c*M*o2*o-fJO3(t)*o3-d*o*o3+v1-v2)*3600
523     do2=(a*M*o**2+2*d*o*o3+fJO3(t)*o3-fJO2(t)*o2-c*M*o2*o-
    v1+v2)*3600
524     do=(2*fJO2(t)*o2+fJO3(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600 #
    Divergence d'euler à cause du terme c*M*o2*o
525     dno=(v1-v2)*3600
526     dno2=(v2-v1)*3600
527     return np.array([do3,do2,do,dno,dno2])
528
529 def Euler_Explicite(f,y0,t0,dt,t1):
530     y=y0
531     Y=[y]
532     t=t0
533     T=[t]
534     while t<t1:
535         t+=dt
536         yp=f(y,t)
537         y=y+yp*dt
538         Y.append(y)
539         T.append(t)
540     return T,Y
541
542 # Résolution
543
544 t0=0
545 dt=1/1000000000
546 t1=0.001
547 T1,Y=Euler_Explicite(F,V0,t0,dt,t1)
548 Y=np.array(Y)
549 LO3=Y[:,0]
550 LO2=Y[:,1]
551 LO=Y[:,2]
552 LNO=Y[:,3]
553 LNO2=Y[:,4]
554
555 plt.figure(6)
```

```
556 plt.plot(T1,LO3,label="O3 simulé informatiquement")
557 #plt.plot(T1,LO2,label="O2 simulation")
558 #plt.plot(T1,LO,label="O simulation")
559 #plt.plot(T1,LNO2,label="NO2 simulation")
560 #plt.plot(T1,LNO,label="NO simulation")
561 plt.xlabel("Temps en H")
562 plt.ylabel("Concentration en molécule.cm-3")
563 plt.title("évolution concentration ozone")
564 plt.legend()
565 plt.show()
566
567 ## Méthode Runge-kutta Cycle de Chapman avec polluant sans émission
    voiture
568
569 plt.close('all')
570 import numpy as np
571 from math import cos,sin,pi,acos,exp
572 from matplotlib import pyplot as plt
573
574 # Données:
575
576 Tp=298 # Température en K
577 Na=6.02*10**23 # en mol-1
578 a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
579 c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
580 d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
581 k=1.8*10**-14 # en cm3.molécules-1.s-1
582 M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
583 jO3=3*10**-5 # en s-1
584 jO2=10**-12 # en s-1 hypothese lineaire de la figure 3
585 jNO2=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
586 o30=Na*10**-12*CO3[1896]/48 # concentration en molécules.cm-3 à
    l'équinoxe du printemps (20 mars 2021)
587 o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène
    dans l'air
588 o0=0 #hypothèse pas d'oxygene atomique initialement dans l'air
589 no0=Na*10**-12*CNO[1896]/30  # concentration en molécules.cm-3 à
    l'équinoxe du printemps (20 mars 2021)
590 no20=Na*10**-12*CNO2[1896]/46  # concentration en molécules.cm-3 à
    l'équinoxe du printemps (20 mars 2021)
591 V0=np.array([o30,o20,o0,no0,no20]) # condition initiale
592
593 # Simulation:
594
595 def fJO3(t):
596     return jO3*cos(fteta(t))
597
```

ANNEXE :

- Programme python :

```
def fJO2(t):
    return jO2*cos(fteta(t))

def fJNO2(t):
    return jNO2*cos(fteta(t))

def F(V,t):
    o3,o2,o,no,no2=V
    v1=fJNO2(t)*no2
    v2=no*o3*k
    do3=(c*M*o2*o-fJO3(t)*o3-d*o*o3+v1-v2)*3600
    do2=(a*M*o**2+2*d*o*o3+fJO3(t)*o3-fJO2(t)*o2-c*M*o2*o-v1+v2)*3600
    do=(2*fJO2(t)*o2+fJO3(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600 # Même problème de divergence que précédemment
    dno=(v1-v2)*3600
    dno2=(v2-v1)*3600
    return np.array([do3,do2,do,dno,dno2])

def Rk4(f,y0,t0,dt,t1):
    y=y0
    Y=[y]
    t=t0
    T=[t]
    while t<t1:
        t+=dt
        k1=f(y,t)
        k2=f(y+dt*k1/2,t+dt/2)
        k3=f(y+k2*dt/2,t+dt/2)
        k4=f(y+dt*k3,t+dt)
        y=y+dt*(k1+2*k2+2*k3+k4)/6
        Y.append(y)
        T.append(t)
    return T,Y

# Résolution

t0=0
dt=1/100000
t1=1
T1,Y=Rk4(F,V0,t0,dt,t1)
Y=np.array(Y)
LO3=Y[:,0]
LO2=Y[:,1]
LO=Y[:,2]
LNO=Y[:,3]
LNO2=Y[:,4]

plt.figure(7)
plt.plot(T1,LO3,label="O3 simulé informatiquement")
#plt.plot(T1,LO2,label="O2")
#plt.plot(T1,LO,label="O")
#plt.plot(T1,LNO2,label="NO2")
#plt.plot(T1,LNO,label="NO")
plt.xlabel("Temps en H")
plt.ylabel("Concentration en molécule.cm-3")
plt.title("évolution concentration ozone")
plt.legend()
plt.show()

## Résolution odeint sans émission

plt.close('all')
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from math import cos,sin,pi,acos,exp

#Données:

Tp=298 # Température en K
Na=6.02*10**23 # en mol-1
a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
jO3=3*10**-5 # en s-1 pour un angle zénithal de 0°
jO2=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
jNO2=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
o30=Na*10**-12*CO3[1896]/48 # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène dans l'air
o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans l'air
```

ANNEXE :

- Programme python :

```
679 no0=Na*10**-12*CNO[1896]/30  # concentration en molécules.cm-3 à
    l'équinoxe du printemps (20 mars 2021)
680 no20=Na*10**-12*CNO2[1896]/46  # concentration en molécules.cm-3 à
    l'équinoxe du printemps (20 mars 2021)
681 V0=np.array([o30,o20,o0,no0,no20]) # condition initiale
682
683 # Simulation:
684
685 def fJO3(t):
686     return jO3*cos(fteta(t))
687
688 def fJO2(t):
689     return jO2*cos(fteta(t))
690
691 def fJNO2(t):
692     return jNO2*cos(fteta(t))
693
694 def F(V,t):
695     o3,o2,o,no,no2=V
696     v1=fJNO2(t)*no2
697     v2=no*o3*k
698     do3=(c*M*o2*o-fJO3(t)*o3-d*o*o3+v1-v2)*3600
699     do2=(a*M*o**2+2*d*o*o3+fJO3(t)*o3-fJO2(t)*o2-c*M*o2*o-
    v1+v2)*3600
700     do=(2*fJO2(t)*o2+fJO3(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600 #
    Problème de divergence résolu
701     dno=(v1-v2)*3600
702     dno2=(v2-v1)*3600
703     return np.array([do3,do2,do,dno,dno2])
704
705 # Résolution
706
707 T1=np.linspace(0,288,100025) #12 Jour pour dépassement
708 Y=odeint(F,V0,T1)
709 Y=np.array(Y)
710 LO3=Y[:,0]
711 LO2=Y[:,1]
712 LO=Y[:,2]
713 LNO=Y[:,3]
714 LNO2=Y[:,4]
715
```

```
716 # Liste des concentrations réellement mesuré
717
718 TT=[k for k in range(49)]
719 CrO3=[]
720 CrNO=[]
721 CrNO2=[]
722 for k in range(len(TT)):
723     CrO3.append(10**-12*Na*CO3[1896+k]/48)
724     CrNO.append(10**-12*Na*CNO[1896+k]/30)
725     CrNO2.append(10**-12*Na*CNO2[1896+k]/46)
726
727 # 1er seuil d'alerte pollution ozone: 240 microgramme.m-3 (liste A1)
728 # 2ème seuil d'alerte pollution ozone: 300 microgramme.m-3 (liste
    A2)
729 # 3ème seuil d'alerte pollution ozone: 360 microgramme.m-3 (liste
    A3)
730
731 A1=[]
732 A2=[]
733 A3=[]
734 for k in range(len(T1)):
735     A1.append(30*10**11) #conversion en molécules.cm-3
736     A2.append(37*10**11)
737     A3.append(45*10**11)
738
739 plt.figure(8)
740 plt.plot(T1,LO3,label="O3 simulé informatiquement")
741 #plt.plot(T1,LO2,label="O2 simulation")
742 #plt.plot(T1,LO,label="O simulation")
743 #plt.plot(T1,LNO2,label="NO2 simulation")
744 #plt.plot(T1,LNO,label="NO simulation")
745 plt.plot(TT,CrO3,'ro',label="O3 réel")
746 #plt.plot(TT,CrNO,'ro',label="NO réel")
747 #plt.plot(TT,CrNO2,'ro',label="NO2 réel")
748 #plt.plot(T1,A1,label="1er seuil d'alerte ozone")
749 #plt.plot(T1,A2,label="2eme seuil d'alerte ozone")
750 #plt.plot(T1,A3,label="3eme seuil d'alerte ozone")
751 plt.xlabel("Temps en H")
752 plt.ylabel("Concentration en molécule.cm-3")
753 plt.title("évolution concentration ozone")
754 plt.legend()
755 plt.show()
```

ANNEXE :

- Programme python :

```
## Résolution odeint avec émission voiture 40% essence/60% diesel

plt.close('all')
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from math import cos,sin,pi,acos,exp

#Données:

Tp=298 # Température en K
Na=6.02*10**23 # en mol-1
a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
jO3=3*10**-5 # en s-1 pour un angle zénithal de 0°
jO2=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
jNO2=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
o30=Na*10**-12*CO3[1896]/48 # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène dans l'air
o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans l'air
no0=Na*10**-12*CNO[1896]/30  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
no20=Na*10**-12*CNO2[1896]/46  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
V0=np.array([o30,o20,o0,no0,no20]) # condition initiale

# Hypothèse pour les émissions: temps calme et polluant évoluant dans un volume de 100m d'altitude et de surface la superficie de la petite couronne parisienne
eno2=Na*10**-12*180/(46*24) # emission de NO2 du au trafic routier en petite couronne en molécule.cm-3.h-1
eno=Na*10**-12*40/(30*24) # emission de NO du au trafic routier en petite couronne en molécule.cm-3.h-1

#Simulation:

def fJO3(t):
    return jO3*cos(fteta(t))

def fJO2(t):
    return jO2*cos(fteta(t))

def fJNO2(t):
    return jNO2*cos(fteta(t))

def F(V,t):
    o3,o2,o,no,no2=V
    v1=fJNO2(t)*no2
    v2=no*o3*k
    do3=(c*M*o2*o-fJO3(t)*o3-d*o*o3+v1-v2)*3600
    do2=(a*M*o**2+2*d*o*o3+fJO3(t)*o3-fJO2(t)*o2-c*M*o2*o-v1+v2)*3600
    do=(2*fJO2(t)*o2+fJO3(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
    dno=(v1-v2)*3600+eno
    dno2=(v2-v1)*3600+eno2
    return np.array([do3,do2,do,dno,dno2])

# Résolution

T1=np.linspace(0,100,100000)
Y=odeint(F,V0,T1)
Y=np.array(Y)
LO3=Y[:,0]
LO2=Y[:,1]
LO=Y[:,2]
LNO=Y[:,3]
LNO2=Y[:,4]

# Liste des concentrations réellement mesuré

TT=[k for k in range(101)]
CrO3=[]
CrNO=[]
CrNO2=[]
for k in range(len(TT)):
    CrO3.append(10**-12*Na*CO3[1896+k]/48)
    CrNO.append(10**-12*Na*CNO[1896+k]/30)
    CrNO2.append(10**-12*Na*CNO2[1896+k]/46)

# 1er seuil d'alerte pollution ozone: 240 microgramme.m-3 (liste A1)
# 2ème seuil d'alerte pollution ozone: 300 microgramme.m-3 (liste A2)
```

ANNEXE :

- Programme python :

```
# 3ème seuil d'alerte pollution ozone: 360 microgramme.m-3 (liste A3)

A1=[]
A2=[]
A3=[]
for k in range(len(T1)):
    A1.append(30*10**11) #conversion en molécules.cm-3
    A2.append(37*10**11)
    A3.append(45*10**11)

plt.figure(9)
plt.plot(T1,LO3,label="O3 simulé informatiquement")
#plt.plot(T1,LO2,label="O2 simulation")
#plt.plot(T1,LO,label="O simulation")
#plt.plot(T1,LNO2,label="NO2 simulation")
#plt.plot(T1,LNO,label="NO simulation")
plt.plot(TT,CrO3,'ro',label="O3 réel")
#plt.plot(TT,CrNO,'ro',label="NO réel")
#plt.plot(TT,CrNO2,'ro',label="NO2 réel")
#plt.plot(T1,A1,label="1er seuil d'alerte ozone")
#plt.plot(T1,A2,label="2eme seuil d'alerte ozone")
#plt.plot(T1,A3,label="3eme seuil d'alerte ozone")
plt.xlabel("Temps en H")
plt.ylabel("Concentration en molécule.cm-3")
plt.title("évolution concentration ozone")
plt.legend()
plt.show()

## Odeint avec émission solstice été

plt.close('all')
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from math import cos,sin,pi,acos,exp

#Données:

Tp=298 # Température en K
Na=6.02*10**23 # en mol-1
a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
jO3=3*10**-5 # en s-1 pour un angle zénithal de 0°
jO2=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
jNO2=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
o30=Na*10**-12*CO3[2208]/48 # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène dans l'air
o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans l'air
no0=Na*10**-12*CNO[2208]/30  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
no20=Na*10**-12*CNO2[2208]/46  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
V0=np.array([o30,o20,o0,no0,no20]) # condition initiale

# Hypothèse pour les émissions: temps calme et polluant évoluant dans un volume de 100m d'altitude et de surface la superficie de la petite couronne parisienne
eno2=Na*10**-12*180/(46*24) # emission de NO2 du au trafic routier en petite couronne en molécule.cm-3.h-1
eno=Na*10**-12*40/(30*24) # emission de NO du au trafic routier en petite couronne en molécule.cm-3.h-1

#Simulation:

def fJO3(t):
    return jO3*cos(fteta(t))

def fJO2(t):
    return jO2*cos(fteta(t))

def fJNO2(t):
    return jNO2*cos(fteta(t))
```

ANNEXE :

- Programme python :

```
def F(V,t):
    o3,o2,o,no,no2=V
    v1=fJNO2(t)*no2
    v2=no*o3*k
    do3=(c*M*o2*o-fJO3(t)*o3-d*o*o3+v1-v2)*3600
    do2=(a*M*o**2+2*d*o*o3+fJO3(t)*o3-fJO2(t)*o2-c*M*o2*o-
v1+v2)*3600
    do=(2*fJO2(t)*o2+fJO3(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
    dno=(v1-v2)*3600+eno
    dno2=(v2-v1)*3600+eno2
    return np.array([do3,do2,do,dno,dno2])

# Résolution

T1=np.linspace(2208,2508,100000)
Y=odeint(F,V0,T1)
Y=np.array(Y)
LO3=Y[:,0]
LO2=Y[:,1]
LO=Y[:,2]
LNO=Y[:,3]
LNO2=Y[:,4]

# Liste des concentrations réellement mesuré

TT=[k for k in range(301)]
CrO3=[]
CrNO=[]
CrNO2=[]
for k in range(len(TT)):
    CrO3.append(10**-12*Na*CO3[2208+k]/48)
    CrNO.append(10**-12*Na*CNO[2208+k]/30)
    CrNO2.append(10**-12*Na*CNO2[2208+k]/46)

# 1er seuil d'alerte pollution ozone: 240 microgramme.m-3 (liste A1)
# 2ème seuil d'alerte pollution ozone: 300 microgramme.m-3 (liste
A2)
# 3ème seuil d'alerte pollution ozone: 360 microgramme.m-3 (liste
A3)

A1=[]
A2=[]
A3=[]
for k in range(len(T1)):
    A1.append(30*10**11) #conversion en molécules.cm-3
    A2.append(37*10**11)
    A3.append(45*10**11)

plt.figure(10)
plt.plot(T1,LO3,label="O3 simulé informatiquement")
#plt.plot(T1,LO2,label="O2 simulation")
#plt.plot(T1,LO,label="O simulation")
#plt.plot(T1,LNO2,label="NO2 simulation")
#plt.plot(T1,LNO,label="NO simulation")
#plt.plot(TT,CrO3,'ro',label="O3 réel")
#plt.plot(TT,CrNO,'ro',label="NO réel")
#plt.plot(TT,CrNO2,'ro',label="NO2 réel")
plt.plot(T1,A1,label="1er seuil d'alerte ozone")
#plt.plot(T1,A2,label="2eme seuil d'alerte ozone")
#plt.plot(T1,A3,label="3eme seuil d'alerte ozone")
plt.xlabel("Temps en H")
plt.ylabel("Concentration en molécule.cm-3")
plt.title("évolution concentration ozone")
plt.legend()
plt.show()

## Odeint avec émission solstice hiver

plt.close('all')
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from math import cos,sin,pi,acos,exp

#Données:

Tp=298 # Température en K
Na=6.02*10**23 # en mol-1
a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
```

ANNEXE :

- Programme python :

```
jO3=3*10**-5 # en s-1 pour un angle zénithal de 0°
jO2=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
jNO2=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
o30=Na*10**-12*CO3[6600]/48 # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène dans l'air
o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans l'air
no0=Na*10**-12*CNO[6600]/30  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
no20=Na*10**-12*CNO2[6600]/46  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
V0=np.array([o30,o20,o0,no0,no20]) # condition initiale

# Hypothèse pour les émissions: temps calme et polluant évoluant dans un volume de 100m d'altitude et de surface la superficie de la petite couronne parisienne
eno2=Na*10**-12*180/(46*24) # emission de NO2 du au trafic routier en petite couronne en molécule.cm-3.h-1
eno=Na*10**-12*40/(30*24) # emission de NO du au trafic routier en petite couronne en molécule.cm-3.h-1

#Simulation:

def fJO3(t):
    return jO3*cos(fteta(t))

def fJO2(t):
    return jO2*cos(fteta(t))

def fJNO2(t):
    return jNO2*cos(fteta(t))

def F(V,t):
    o3,o2,o,no,no2=V
    v1=fJNO2(t)*no2
    v2=no*o3*k
    do3=(c*M*o2*o-fJO3(t)*o3-d*o*o3+v1-v2)*3600
    do2=(a*M*o**2+2*d*o*o3+fJO3(t)*o3-fJO2(t)*o2-c*M*o2*o-v1+v2)*3600
    do=(2*fJO2(t)*o2+fJO3(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
    dno=(v1-v2)*3600+eno
    dno2=(v2-v1)*3600+eno2
    return np.array([do3,do2,do,dno,dno2])

# Résolution

T1=np.linspace(6600,6800,100000)
Y=odeint(F,V0,T1)
Y=np.array(Y)
LO3=Y[:,0]
LO2=Y[:,1]
LO=Y[:,2]
LNO=Y[:,3]
LNO2=Y[:,4]

# Liste des concentrations réellement mesuré

TT=[k for k in range(301)]
CrO3=[]
CrNO=[]
CrNO2=[]
for k in range(len(TT)):
    CrO3.append(10**-12*Na*CO3[6600+k]/48)
    CrNO.append(10**-12*Na*CNO[6600+k]/30)
    CrNO2.append(10**-12*Na*CNO2[6600+k]/46)

# 1er seuil d'alerte pollution ozone: 240 microgramme.m-3 (liste A1)
# 2ème seuil d'alerte pollution ozone: 300 microgramme.m-3 (liste A2)
# 3ème seuil d'alerte pollution ozone: 360 microgramme.m-3 (liste A3)

A1=[]
A2=[]
A3=[]
for k in range(len(T1)):
    A1.append(30*10**11) #conversion en molécules.cm-3
    A2.append(37*10**11)
    A3.append(45*10**11)

plt.figure(11)
plt.plot(T1,LO3,label="O3 simulé informatiquement")
#plt.plot(T1,LO2,label="O2 simulation")
#plt.plot(T1,LO,label="O simulation")
#plt.plot(T1,LNO2,label="NO2 simulation")
#plt.plot(T1,LNO,label="NO simulation")
#plt.plot(TT,CrO3,'ro',label="O3 réel")
#plt.plot(TT,CrNO,'ro',label="NO réel")
#plt.plot(TT,CrNO2,'ro',label="NO2 réel")
plt.plot(T1,A1,label="1er seuil d'alerte ozone")
```

ANNEXE :

- Programme python :

```
#plt.plot(T1,A2,label="2eme seuil d'alerte ozone")
#plt.plot(T1,A3,label="3eme seuil d'alerte ozone")
plt.xlabel("Temps en H")
plt.ylabel("Concentration en molécule.cm-3")
plt.title("évolution concentration ozone")
plt.legend()
plt.show()

## Odeint avec émission, influence température négligeable

plt.close('all')
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from math import cos,sin,pi,acos,exp

#Données:

Tp=310 # Température en K
Na=6.02*10**23 # en mol-1
a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
jO3=3*10**-5 # en s-1 pour un angle zénithal de 0°
jO2=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
jNO2=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
o30=Na*10**-12*CO3[1896]/48 # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène dans l'air
o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans l'air
no0=Na*10**-12*CNO[1896]/30  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
no20=Na*10**-12*CNO2[1896]/46  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
V0=np.array([o30,o20,o0,no0,no20]) # condition initiale

# Hypothèse pour les émissions: temps calme et polluant évoluant dans un volume de 100m d'altitude et de surface la superficie de la petite couronne parisienne
eno2=Na*10**-12*180/(46*24) # emission de NO2 du au trafic routier en petite couronne en molécule.cm-3.h-1
eno=Na*10**-12*40/(30*24) # emission de NO du au trafic routier en petite couronne en molécule.cm-3.h-1

#Simulation:

def fJO3(t):
    return jO3*cos(fteta(t))

def fJO2(t):
    return jO2*cos(fteta(t))

def fJNO2(t):
    return jNO2*cos(fteta(t))

def F(V,t):
    o3,o2,o,no,no2=V
    v1=fJNO2(t)*no2
    v2=no*o3*k
    do3=(c*M*o2*o-fJO3(t)*o3-d*o*o3+v1-v2)*3600
    do2=(a*M*o**2+2*d*o*o3+fJO3(t)*o3-fJO2(t)*o2-c*M*o2*o-v1+v2)*3600
    do=(2*fJO2(t)*o2+fJO3(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
    dno=(v1-v2)*3600+eno
    dno2=(v2-v1)*3600+eno2
    return np.array([do3,do2,do,dno,dno2])

# Résolution

T1=np.linspace(0,100,100000)
Y=odeint(F,V0,T1)
Y=np.array(Y)
LO3=Y[:,0]
LO2=Y[:,1]
LO=Y[:,2]
LNO=Y[:,3]
LNO2=Y[:,4]

# Liste des concentrations réellement mesuré

TT=[k for k in range(101)]
CrO3=[]
CrNO=[]
CrNO2=[]
for i in range(len(TT)):
    CrO3.append(10**-12*Na*CO3[1896+i]/48)
```

ANNEXE :

- Programme python :

```
    CrNO.append(10**-12*Na*CNO[1896+i]/30)
    CrNO2.append(10**-12*Na*CNO2[1896+i]/46)


plt.figure(12)
plt.plot(T1,LO3,'g',label="O3 simulé T=310K")
plt.plot(TT,CrO3,'ro',label="O3 réel")
plt.xlabel("Temps en H")
plt.ylabel("Concentration en molécule.cm-3")
plt.title("évolution concentration ozone")
plt.legend()
plt.show()

# Nouvelle données:

Tp=270 # Température en K
a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1

#Simulation:

def fJO3(t):
    return jO3*cos(fteta(t))

def fJO2(t):
    return jO2*cos(fteta(t))

def fJNO2(t):
    return jNO2*cos(fteta(t))

def F(V,t):
    o3,o2,o,no,no2=V
    v1=fJNO2(t)*no2
    v2=no*o3*k
    do3=(c*M*o2*o-fJO3(t)*o3-d*o*o3+v1-v2)*3600
    do2=(a*M*o**2+2*d*o*o3+fJO3(t)*o3-fJO2(t)*o2-c*M*o2*o-
v1+v2)*3600
    do=(2*fJO2(t)*o2+fJO3(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
    dno=(v1-v2)*3600+eno
    dno2=(v2-v1)*3600+eno2
    return np.array([do3,do2,do,dno,dno2])

# Résolution
```

```
T1=np.linspace(0,100,100000)
Y=odeint(F,V0,T1)
Y=np.array(Y)
LO3=Y[:,0]
LO2=Y[:,1]
LO=Y[:,2]
LNO=Y[:,3]
LNO2=Y[:,4]

plt.figure(12)
plt.plot(T1,LO3,'b',label="O3 simulé T=270K")
plt.xlabel("Temps en H")
plt.ylabel("Concentration en molécule.cm-3")
plt.title("évolution concentration ozone")
plt.legend()
plt.show()

## Test odeint avec émission voiture 100% essence

plt.close('all')
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from math import cos,sin,pi,acos,exp

#Données:

Tp=298 # Température en K
Na=6.02*10**23 # en mol-1
a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
jO3=3*10**-5 # en s-1 pour un angle zénithal de 0°
jO2=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
jNO2=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
o30=Na*10**-12*CO3[1896]/48 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène
dans l'air
o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans
l'air
no0=Na*10**-12*CNO[1896]/30  # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
```

ANNEXE :

- Programme python :

```
no20=Na*10**-12*CNO2[1896]/46  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
V0=np.array([o30,o20,o0,no0,no20]) # condition initiale

# Hypothèse pour les émissions: temps calme et polluant évoluant dans un volume de 100m d'altitude et de surface la superficie de la petite couronne parisienne
eno2=Na*10**-12*73/(46*24) # emission de NO2 du au trafic routier en petite couronne en molécule.cm-3.h-1
eno=Na*10**-12*2/(30*24) # emission de NO du au trafic routier en petite couronne en molécule.cm-3.h-1

#Simulation:

def fJO3(t):
    return jO3*cos(fteta(t))

def fJO2(t):
    return jO2*cos(fteta(t))

def fJNO2(t):
    return jNO2*cos(fteta(t))

def F(V,t):
    o3,o2,o,no,no2=V
    v1=fJNO2(t)*no2
    v2=no*o3*k
    do3=(c*M*o2*o-fJO3(t)*o3-d*o*o3+v1-v2)*3600
    do2=(a*M*o**2+2*d*o*o3+fJO3(t)*o3-fJO2(t)*o2-c*M*o2*o-v1+v2)*3600
    do=(2*fJO2(t)*o2+fJO3(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
    dno=(v1-v2)*3600+eno
    dno2=(v2-v1)*3600+eno2
    return np.array([do3,do2,do,dno,dno2])

# Résolution

T1=np.linspace(0,300,100000)
Y=odeint(F,V0,T1)
Y=np.array(Y)
LO3=Y[:,0]
LO2=Y[:,1]
LO=Y[:,2]
LNO=Y[:,3]
LNO2=Y[:,4]
```

```
# Liste des concentrations réellement mesuré

TT=[k for k in range(101)]
CrO3=[]
CrNO=[]
CrNO2=[]
for k in range(len(TT)):
    CrO3.append(10**-12*Na*CO3[1896+k]/48)
    CrNO.append(10**-12*Na*CNO[1896+k]/30)
    CrNO2.append(10**-12*Na*CNO2[1896+k]/46)

# 1er seuil d'alerte pollution ozone: 240 microgramme.m-3 (liste A1)
# 2ème seuil d'alerte pollution ozone: 300 microgramme.m-3 (liste A2)
# 3ème seuil d'alerte pollution ozone: 360 microgramme.m-3 (liste A3)

A1=[]
A2=[]
A3=[]
for k in range(len(T1)):
    A1.append(30*10**11) #conversion en molécules.cm-3
    A2.append(37*10**11)
    A3.append(45*10**11)

plt.figure(9)
plt.plot(T1,LO3,label="O3 simulé voiture essence")
#plt.plot(T1,LO2,label="O2 simulation")
#plt.plot(T1,LO,label="O simulation")
#plt.plot(T1,LNO2,label="NO2 simulation")
#plt.plot(T1,LNO,label="NO simulation")
#plt.plot(TT,CrO3,'ro',label="O3 réel")
#plt.plot(TT,CrNO,'ro',label="NO réel")
#plt.plot(TT,CrNO2,'ro',label="NO2 réel")
plt.plot(T1,A1,label="1er seuil d'alerte ozone")
#plt.plot(T1,A2,label="2eme seuil d'alerte ozone")
#plt.plot(T1,A3,label="3eme seuil d'alerte ozone")
plt.xlabel("Temps en H")
plt.ylabel("Concentration en molécule.cm-3")
plt.title("évolution concentration ozone")
plt.legend()
plt.show()

## Test odeint avec émission voiture 100% diesel

plt.close('all')
import numpy as np
```

ANNEXE :

- Programme python :

```
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from math import cos,sin,pi,acos,exp

#Données:

Tp=298 # Température en K
Na=6.02*10**23 # en mol-1
a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
jO3=3*10**-5 # en s-1 pour un angle zénithal de 0°
jO2=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
jNO2=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
o30=Na*10**-12*CO3[1896]/48 # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène dans l'air
o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans l'air
no0=Na*10**-12*CNO[1896]/30  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
no20=Na*10**-12*CNO2[1896]/46  # concentration en molécules.cm-3 à l'équinoxe du printemps (20 mars 2021)
V0=np.array([o30,o20,o0,no0,no20]) # condition initiale

# Hypothèse pour les émissions: temps calme et polluant évoluant dans un volume de 100m d'altitude et de surface la superficie de la petite couronne parisienne
eno2=Na*10**-12*255/(46*24) # emission de NO2 du au trafic routier en petite couronne en molécule.cm-3.h-1
eno=Na*10**-12*73/(30*24) # emission de NO du au trafic routier en petite couronne en molécule.cm-3.h-1

#Simulation:

def fJO3(t):
    return jO3*cos(fteta(t))

def fJO2(t):
    return jO2*cos(fteta(t))

def fJNO2(t):
    return jNO2*cos(fteta(t))

def F(V,t):
    o3,o2,o,no,no2=V
    v1=fJNO2(t)*no2
    v2=no*o3*k
    do3=(c*M*o2*o-fJO3(t)*o3-d*o*o3+v1-v2)*3600
    do2=(a*M*o**2+2*d*o*o3+fJO3(t)*o3-fJO2(t)*o2-c*M*o2*o-v1+v2)*3600
    do=(2*fJO2(t)*o2+fJO3(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
    dno=(v1-v2)*3600+eno
    dno2=(v2-v1)*3600+eno2
    return np.array([do3,do2,do,dno,dno2])

# Résolution

T1=np.linspace(0,300,100000)
Y=odeint(F,V0,T1)
Y=np.array(Y)
LO3=Y[:,0]
LO2=Y[:,1]
LO=Y[:,2]
LNO=Y[:,3]
LNO2=Y[:,4]

# Liste des concentrations réellement mesuré

TT=[k for k in range(101)]
CrO3=[]
CrNO=[]
CrNO2=[]
for k in range(len(TT)):
    CrO3.append(10**-12*Na*CO3[1896+k]/48)
    CrNO.append(10**-12*Na*CNO[1896+k]/30)
    CrNO2.append(10**-12*Na*CNO2[1896+k]/46)

# 1er seuil d'alerte pollution ozone: 240 microgramme.m-3 (liste A1)
# 2ème seuil d'alerte pollution ozone: 300 microgramme.m-3 (liste A2)
# 3ème seuil d'alerte pollution ozone: 360 microgramme.m-3 (liste A3)

A1=[]
A2=[]
A3=[]
```

ANNEXE :

- Programme python/ Données de la simulation :

```
1392 for k in range(len(T1)):
1393     A1.append(30*10**11) #conversion en molécules.cm-3
1394     A2.append(37*10**11)
1395     A3.append(45*10**11)
1396
1397 plt.figure(9)
1398 plt.plot(T1,LO3,label="O3 simulé voiture diesel")
1399 #plt.plot(T1,LO2,label="O2 simulation")
1400 #plt.plot(T1,LO,label="O simulation")
1401 #plt.plot(T1,LNO2,label="NO2 simulation")
1402 #plt.plot(T1,LNO,label="NO simulation")
1403 #plt.plot(TT,CrO3,'ro',label="O3 réel")
1404 #plt.plot(TT,CrNO,'ro',label="NO réel")
1405 #plt.plot(TT,CrNO2,'ro',label="NO2 réel")
1406 plt.plot(T1,A1,label="1er seuil d'alerte ozone")
1407 #plt.plot(T1,A2,label="2eme seuil d'alerte ozone")
1408 #plt.plot(T1,A3,label="3eme seuil d'alerte ozone")
1409 plt.xlabel("Temps en H")
1410 plt.ylabel("Concentration en molécule.cm-3")
1411 plt.title("évolution concentration ozone")
1412 plt.legend()
1413 plt.show()
```

Données simulation diapo 12/13 :

Soit T : température en K

$$a = 9,9 \times \exp(\frac{470}{T}) \times 10^{-34} cm^6.s^{-1}$$
$$c = 1,1 \times \exp(\frac{510}{T}) \times 10^{-34} cm^6.s^{-1}$$
$$d = 1,9 \times \exp(\frac{-2300}{T}) \times 10^{-11} cm^3.s^{-1}$$

- $k = \frac{1}{\tau_{NO} \times [O_3 0]}$
  $= 1,8 \times 10^{-14} cm^3.molécules^{-1}.s^{-1}$

[O30] : concentration initiale
τNO : temps de vie de NO

- $JNO_2 = 1/\tau NO_2$
  $= 8,2 \times 10^{-3} s^{-1}$ (angle zénithal 0°)

τNO2 : temps de vie NO2
[M] : Concentration en molécule d'air (80% N2 20% O2)

- $JO_3 = 3 \times 10^{-5} s^{-1}$ (angle zénithal 0°)

- $JO_2 = 10^{-12} s^{-1}$ (angle zénithal 0° avec hypothèse linéaire)

ANNEXE :

- Document utilisé :

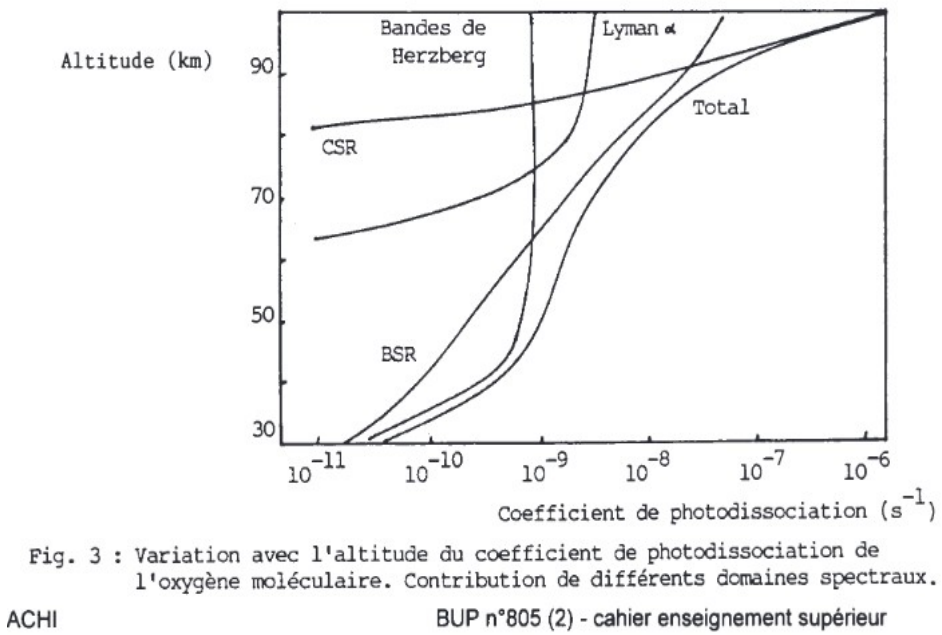


Fig. 3 : Variation avec l'altitude du coefficient de photodissociation de l'oxygène moléculaire. Contribution de différents domaines spectraux.

ACHI BUP n°805 (2) - cahier enseignement supérieur

Source : ENS-lyon physique M Ayachi

	Admissions aux urgences			Mortalité		
	Respiratoire	Cardiovasculaire	Digestive	Respiratoire	Cardiovasculaire	Digestive
Particules fines de moins de 2,5 µm	0,087	0,084	-0,356	0,186	**0,420****	0,102
	(0,235)	(0,206)	(0,182)	(0,112)	**(0,161)**	(0,068)
Monoxyde de carbone	0,022	**0,068****	-0,007	0,008	0,014	-0,002
	(0,028)	**(0,026)**	(0,021)	(0,013)	(0,018)	(0,008)
Dioxyde d'azote	0,256	-0,282	0,103	0,040	0,073	-0,067
	(0,268)	(0,263)	(0,208)	(0,142)	(0,189)	(0,075)
Ozone	**0,560****	0,198	-0,259	-0,066	0,124	0,061
	(0,194)	(0,184)	(0,173)	(0,100)	(0,146)	(0,057)

Source : INSEE

Approximation linéaire pour JO2