

TIPE : Santé et prévention

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

Sommaire :

Introduction :

Problématique :

Combien de jours ensoleillés et sans vent consécutifs faut-il en été pour franchir le seuil d'alerte de pic de pollution à l'ozone dans une ville de l'agglomération parisienne comme Melun?

I) Modélisation de l'angle zénithal

II) Modélisation cycle de Chapman sans émission, difficultés rencontrées

III) Modélisation cycle de Chapman avec émission

Conclusion :

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 :

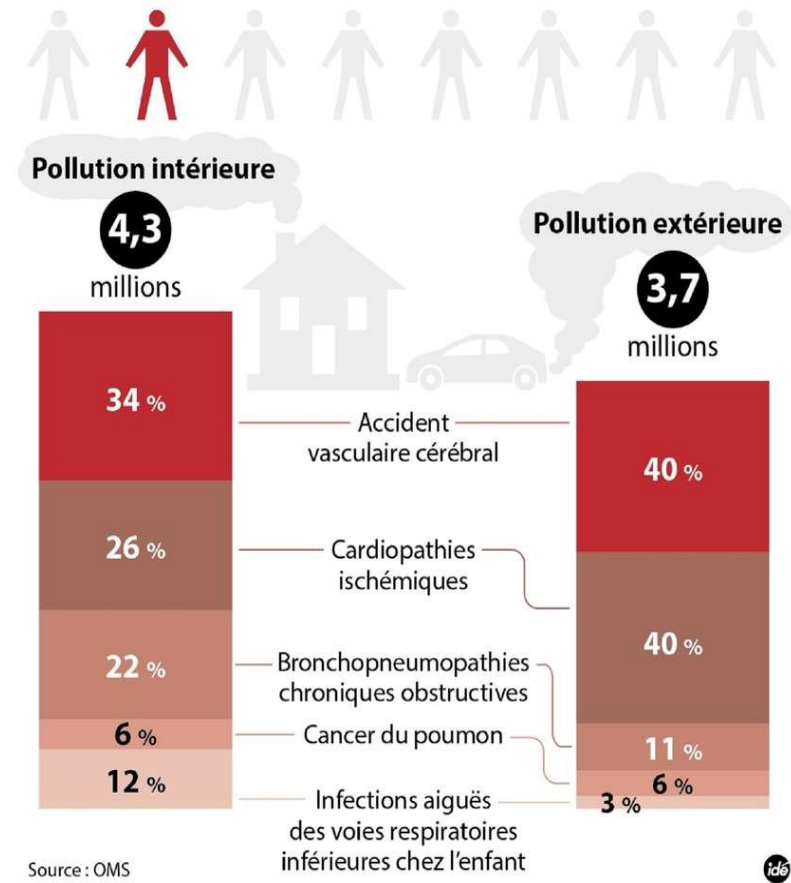
Les polluants :

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

► Nécessité de prévoir et gérer les futurs pics de pollution

La pollution responsable de 7 millions de morts par an

En 2012, **une mort sur huit** serait due à la pollution



INTRODUCTION :

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

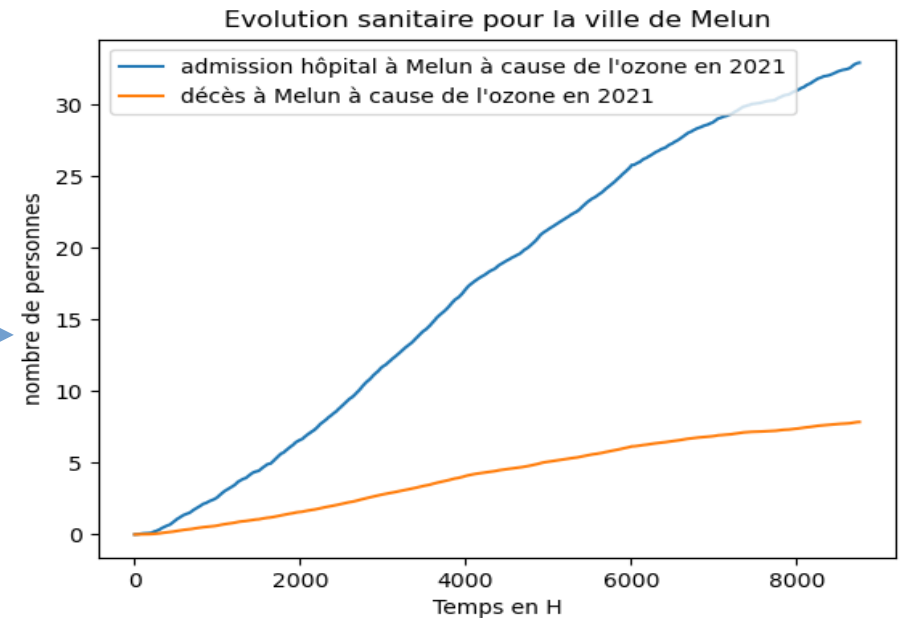
En France en 2021 :

- 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

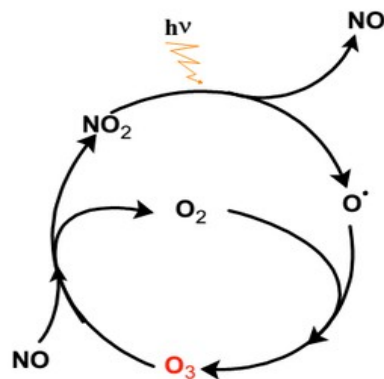
- Qu'est ce que le cycle de Chapman ?

Double rôle de l'ozone (O₃) :

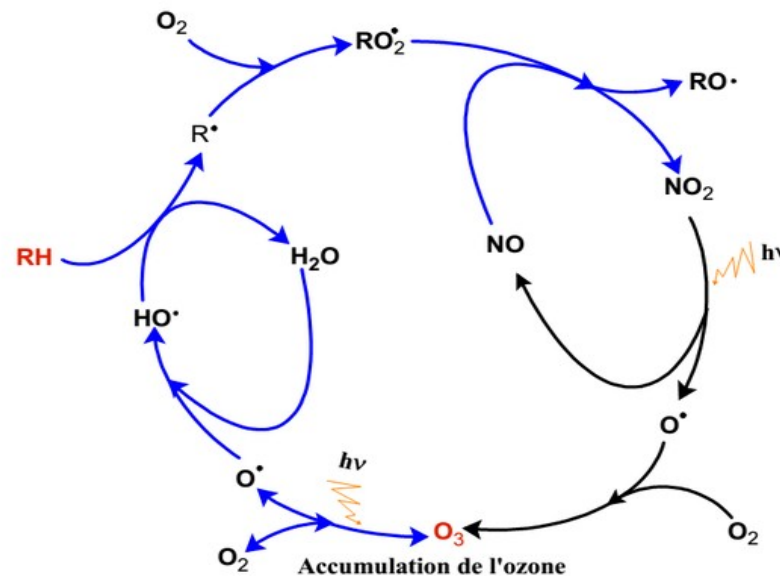
- Ozone stratosphérique = couche d'ozone, protège des rayons UV-B et UV-C (100 à 315nm)

- 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

► Concentration d'ozone varie en fonction :

- De l'**ensoleillement** (hv)
- Des **concentrations des autres polluants** (NO₂...)

Autres facteurs ?

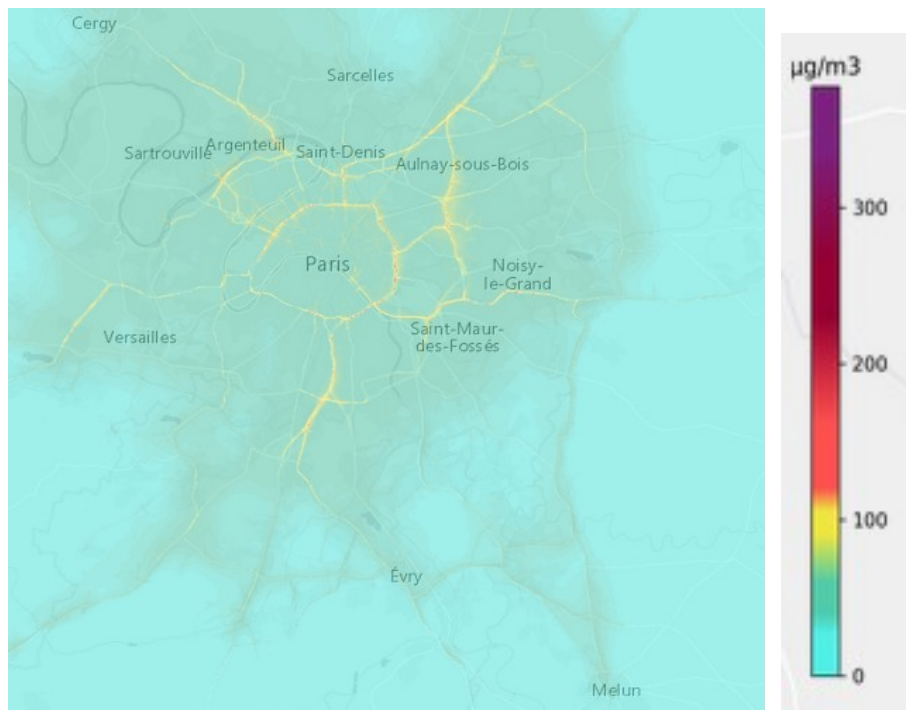
Source: researchgate.net

I) Modélisation de l'angle zénithal

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

Influence lieux/activités humaines :

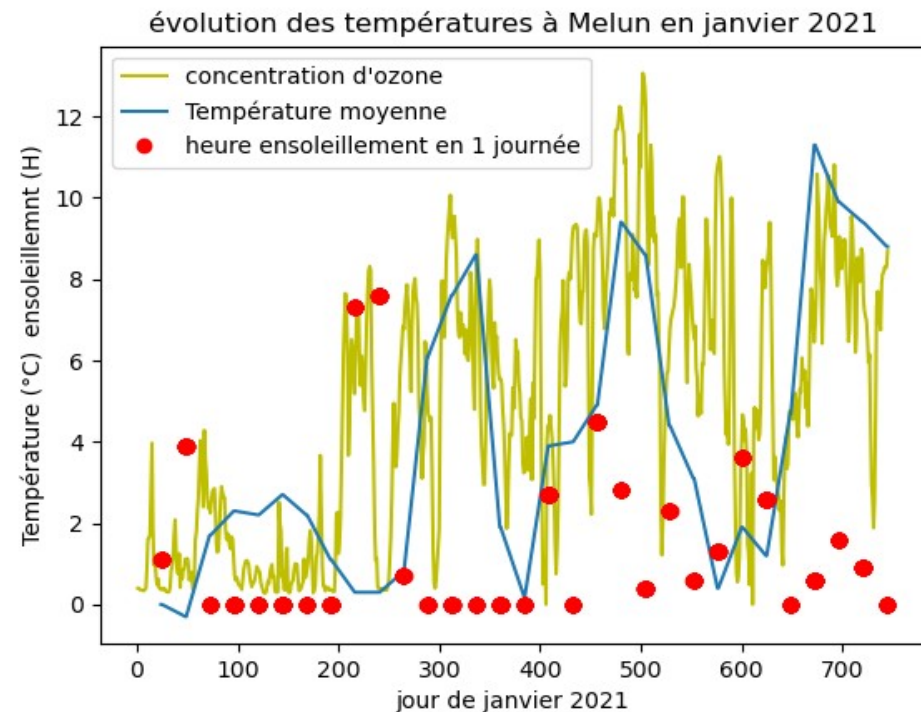
Pollution NO₂ :



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

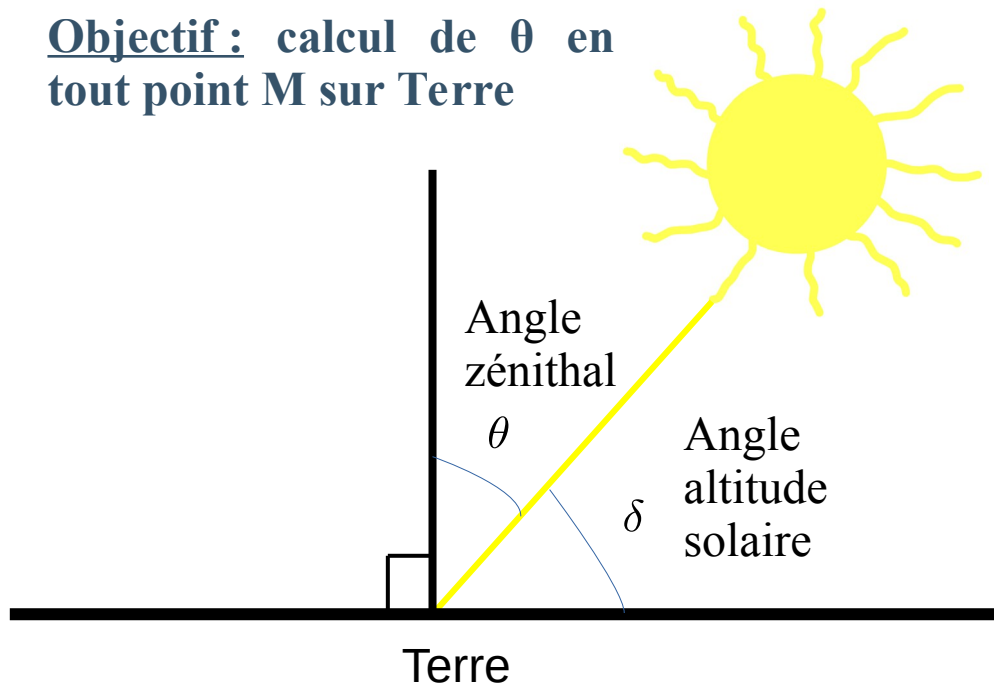


Modélisation journalière des variations de l'angle zénithal pour modéliser le cycle de Chapman

I) Modélisation de l'angle zénithal

- Modélisation de l'angle zénithal :

Objectif : calcul de θ en tout point M sur Terre



Données :

φ_0 : longitude de M

λ : latitude de M

Melun :

- $\varphi_0 = 2,66^\circ$

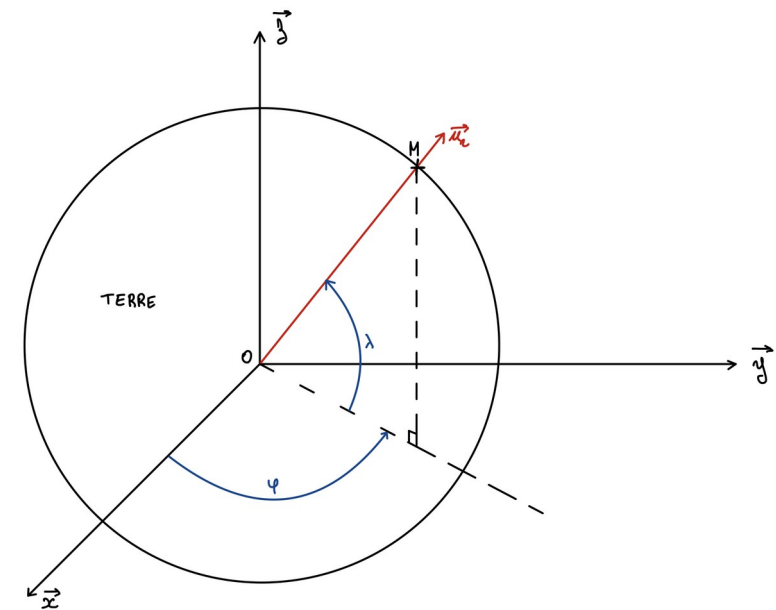
- $\lambda = 48,5^\circ$

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

► Pour tout t :

$$\begin{cases} \varphi(t) = \varphi_0 + \omega t \\ \omega = \frac{2\pi}{1\text{jour}} = \frac{2\pi}{86164} \end{cases}$$

Référentiel :
(Oxyz) géocentrique



$$\vec{u}_r = \begin{pmatrix} \cos \lambda \times \cos \varphi \vec{x} \\ \cos \lambda \times \sin \varphi \vec{y} \\ \sin \lambda \vec{z} \end{pmatrix}$$

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:

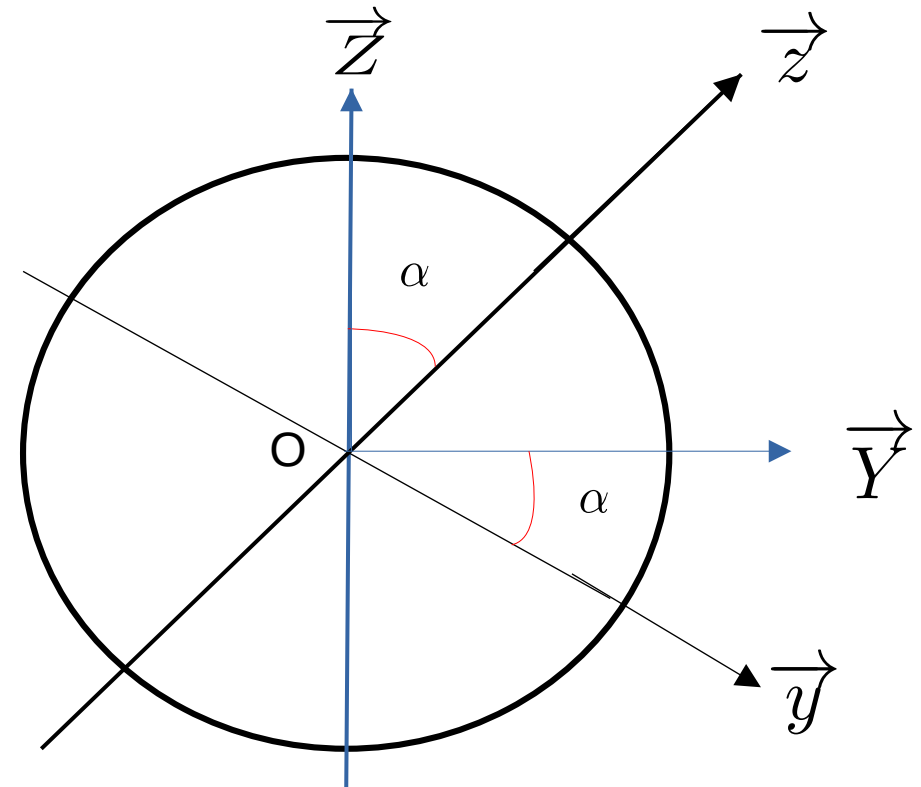
α : angle entre le plan de l'écliptique et le plan de l'équateur

Données géométriques :

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



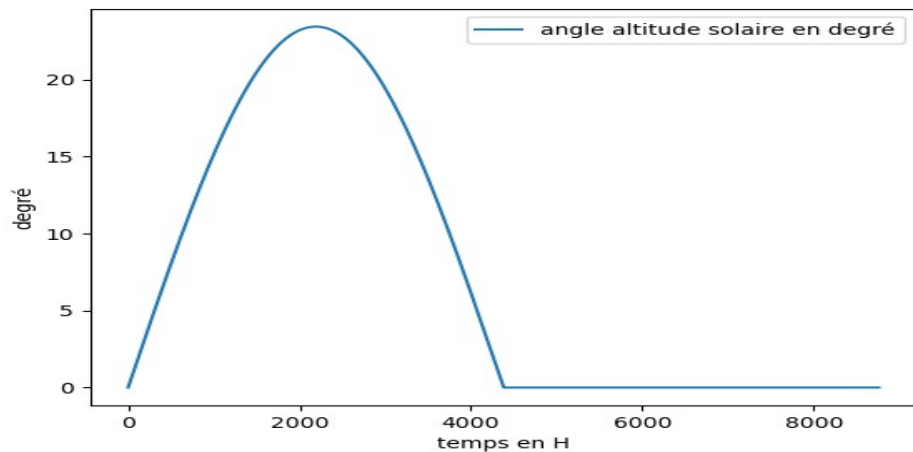
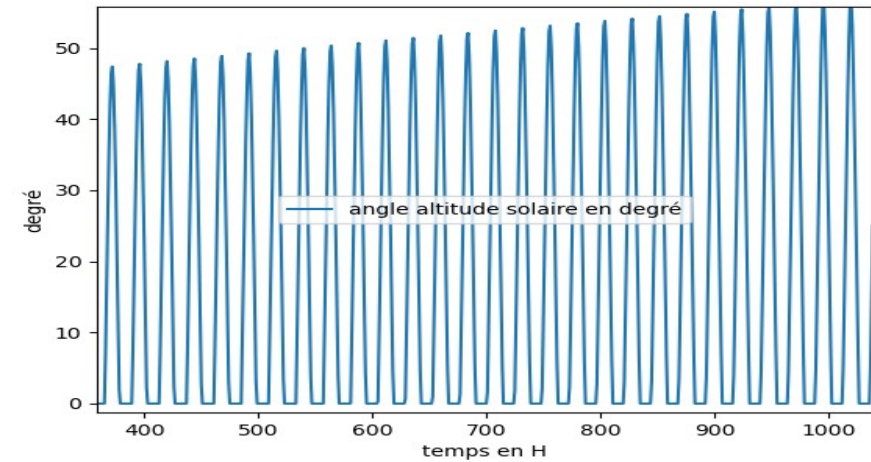
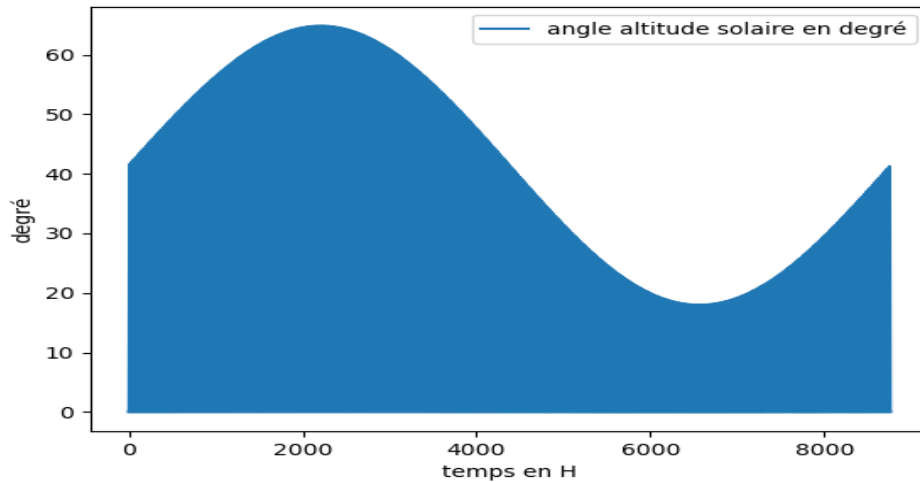
Modélisation Python de θ et δ sur 1 an



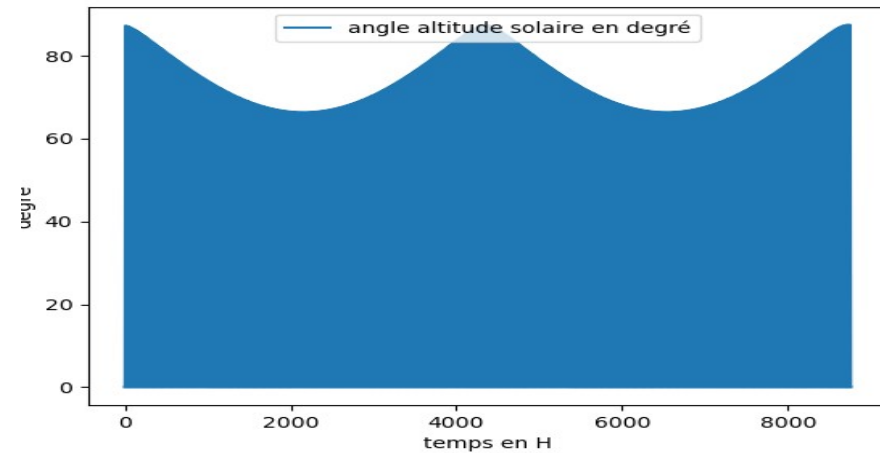
I) Modélisation de l'angle zénithal

- Modélisation de l'angle zénithal :

Résultat modélisation à Melun angle δ :



Pôle Nord ($\lambda=90^\circ$)



Équateur ($\lambda=0^\circ$)

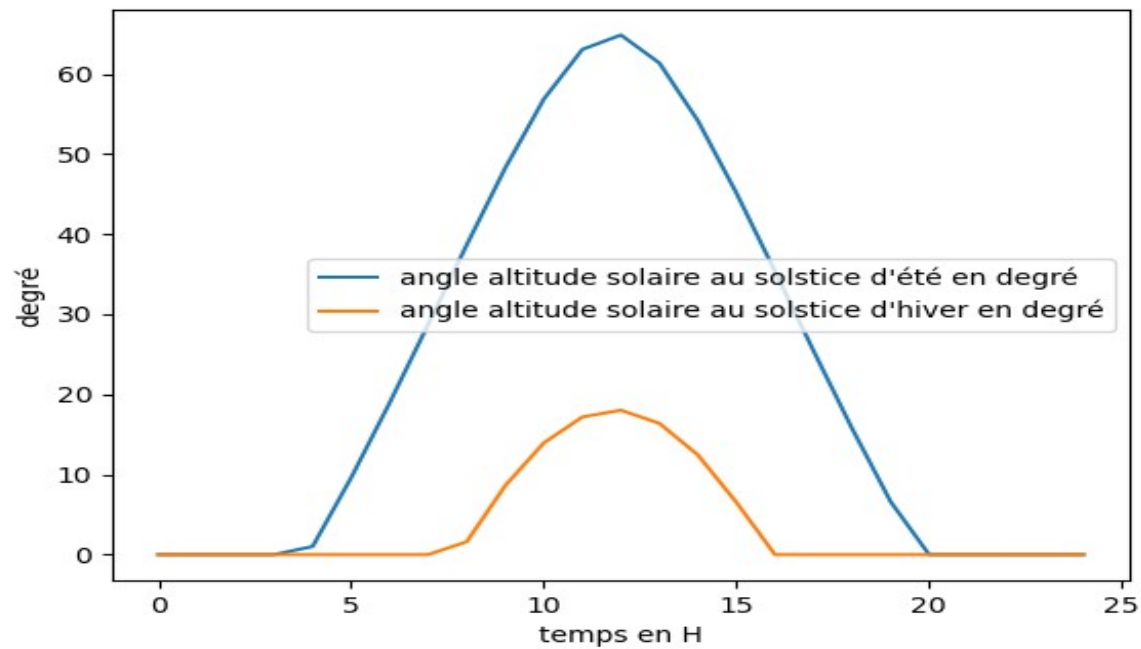


Objectif angle zénithal en tout point M atteint

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

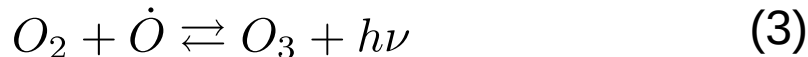
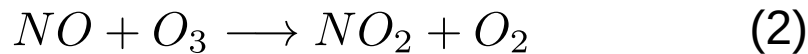
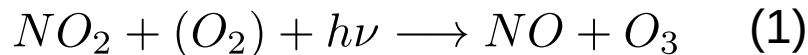
- Modélisation cycle de Chapman méthode d'Euler :

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

Hypothèse de la simulation :

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

Réaction chimique :



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

$$\begin{cases} v_1 = JNO_2(t) \times [NO_2] \\ v_2 = k \times [NO] \times [O_3] \end{cases}$$

Données réaction (en annexe):

Unités choisie pour la simulation des concentrations molécules.cm-3

On a :

$$\begin{aligned} JO_3(t) &= JO_3 \times \cos(\theta(t)) \\ JO_2(t) &= JO_2 \times \cos(\theta(t)) \\ JNO_2(t) &= JNO_2 \times \cos(\theta(t)) \end{aligned}$$

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 :

- [O₀]=0 molécules.cm⁻³ (hypothèse)
- [NO₂],[NO],[O₃] : concentration mesurée équinoxe de printemps
- [O₂] : concentration dans l'air

On pose :

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

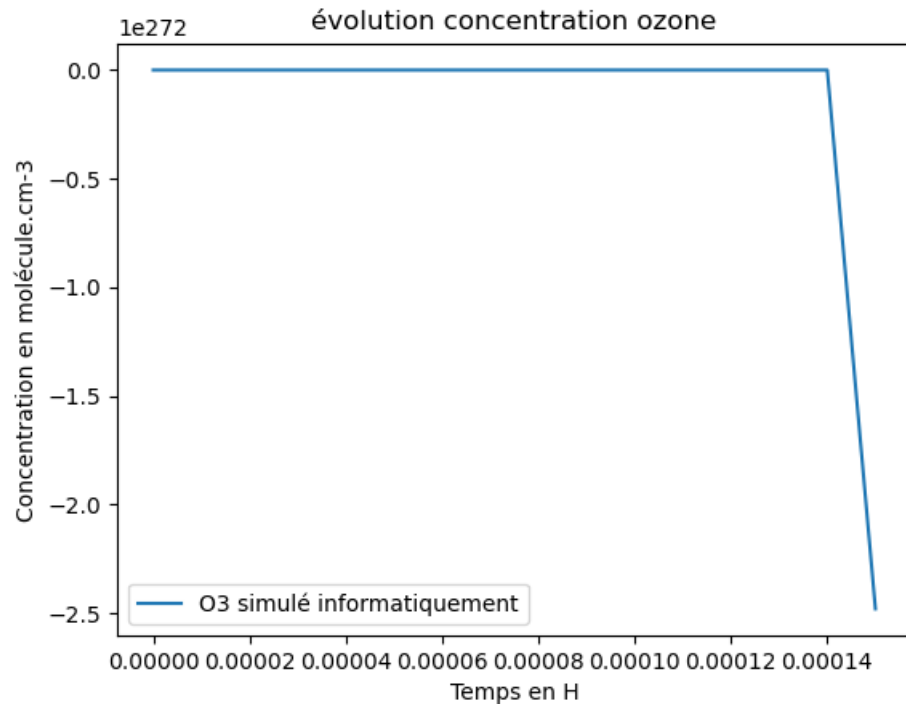
$$= \begin{pmatrix} c \times [M] \times [O_2] \times [O] - JO_3(t) \times [O_3] - d \times [O] \times [O_3] + v1 - v2 \\ a \times [M] \times [O]^2 + JO_3(t) \times [O_3] + 2d \times [O] \times [O_3] - JO_2(t) \times [O_2] - c \times [M] \times [O_2] \times [O] + v2 - v1 \\ -2a \times [M] \times [O]^2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

II) Modélisation cycle de Chapman sans émission, difficultés rencontrées

- Divergence de la méthode d'Euler:

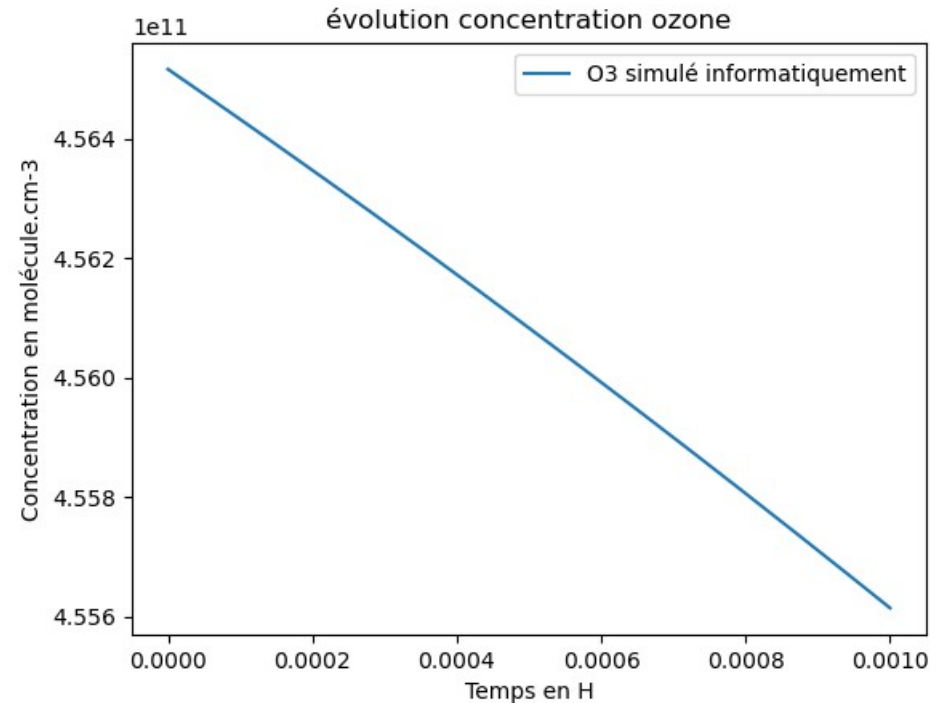
$$V(t + dt) = V(t) + dt \times \frac{\partial V}{\partial t}(t)$$

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 une 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 :

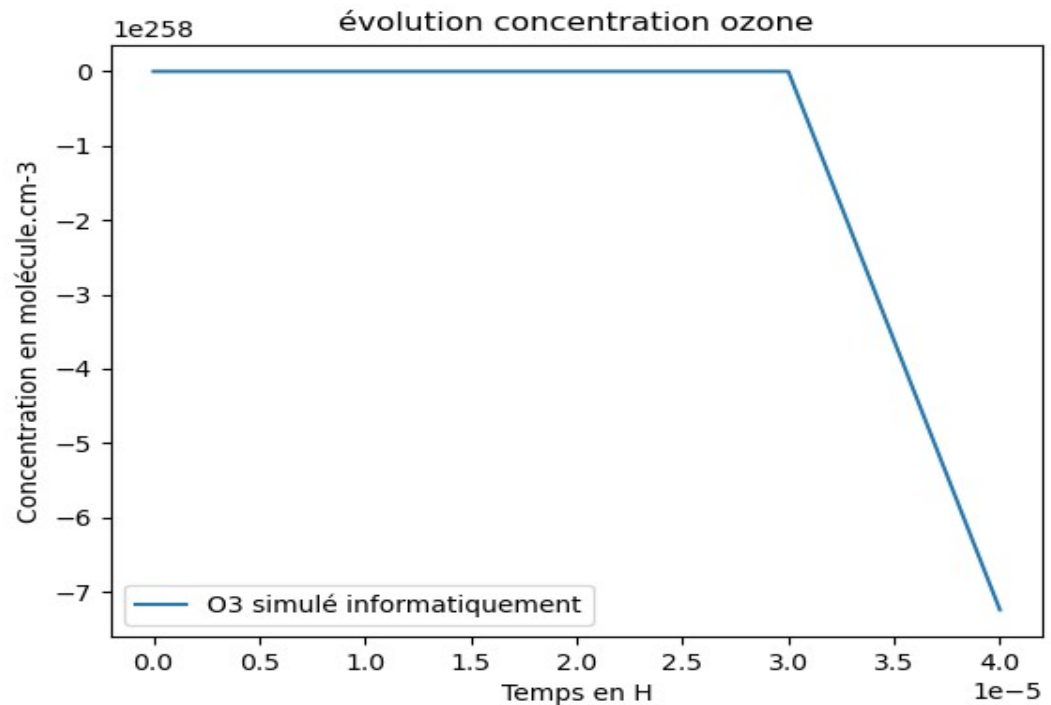
$$V(t + dt) = V(t) + \frac{dt(k1 + 2k2 + 2k3 + k4)}{6}$$

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 $dt > 1/1\ 000\ 000\ 000\ s$

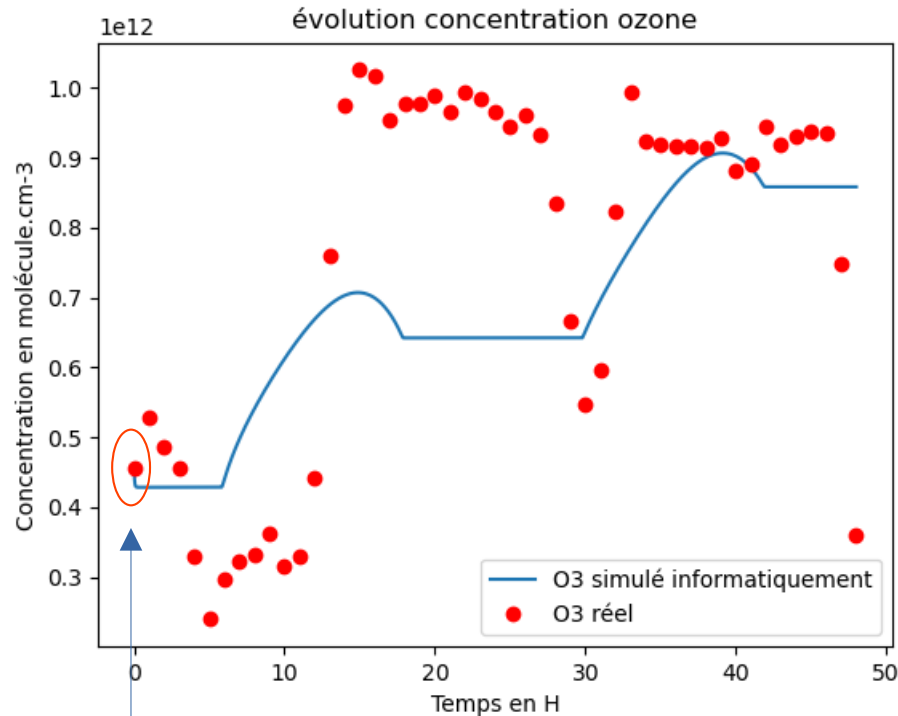


Runge-Kutta = meilleure précision mais insuffisant

II) Modélisation cycle de Chapman sans émission, difficultés rencontrées

- Modélisation cycle de Chapman en utilisant odeint :

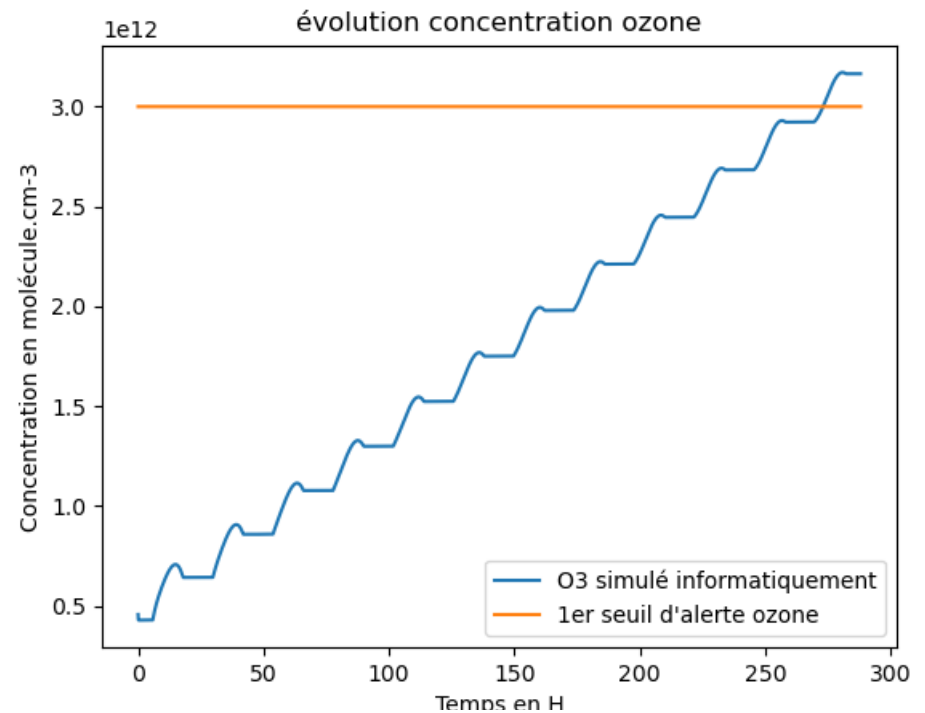
Résultat pour Melun :



- Pas de divergence
- $[O_3]$ varie avec θ
- Régime stationnaire la nuit

Variation due aux conditions initiales

1^{er} seuil d'alerte : $240\mu\text{g.m}^{-3}$



Seuils d'alerte atteint après 12 jours dans ces conditions

➡ Résultat du bon ordre de grandeur mais erreur $\sim 100\%$

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
NO ₂	40	140
NO	1	40

Données parc
INRETS

► Émission quotidienne :

- 180 µg.m-3 NO₂
- 40 µg.m-3 de NO

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

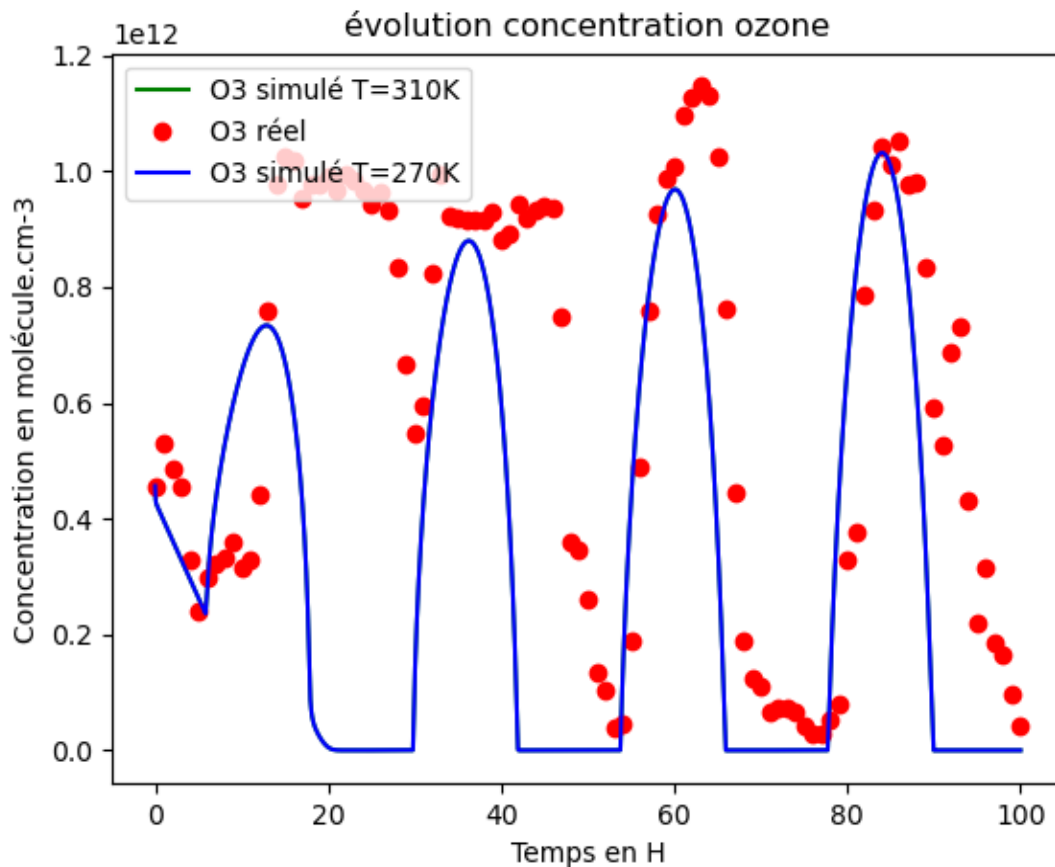
On injecte dans notre système d'équation : $\frac{\partial V}{\partial t}(t) =$

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

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

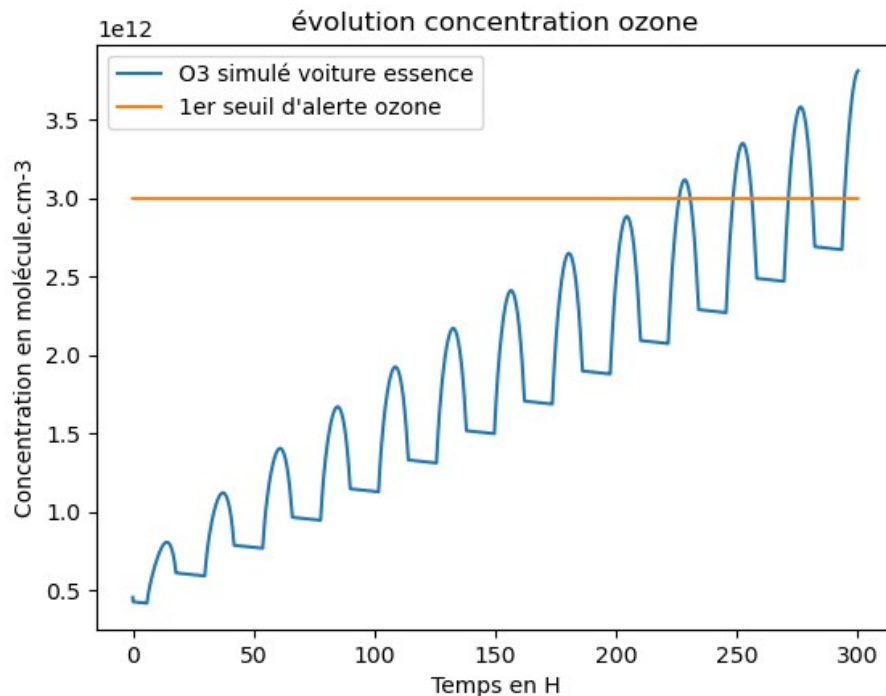
III) Modélisation cycle de Chapman avec émission

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

Cas voiture 100 % essence :

► Émission quotidienne :

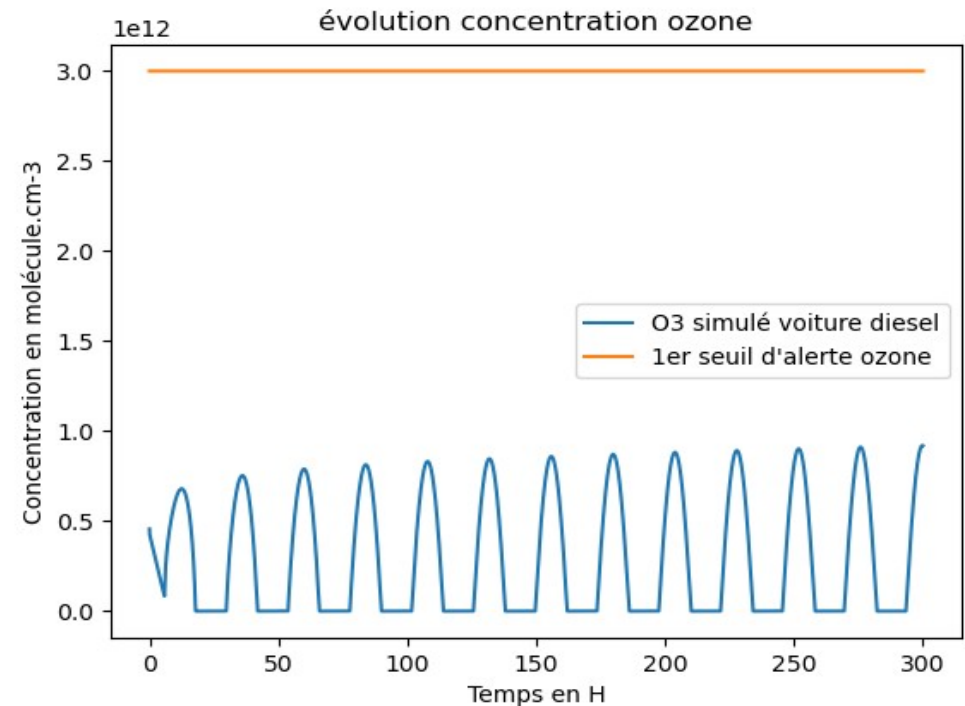
- $73\mu\text{g.m}^{-3}$ NO_2
- $2\mu\text{g.m}^{-3}$ de NO



Cas voiture 100 % diesel :

► Émission quotidienne :

- $255\mu\text{g.m}^{-3}$ NO_2
- $73\mu\text{g.m}^{-3}$ de NO

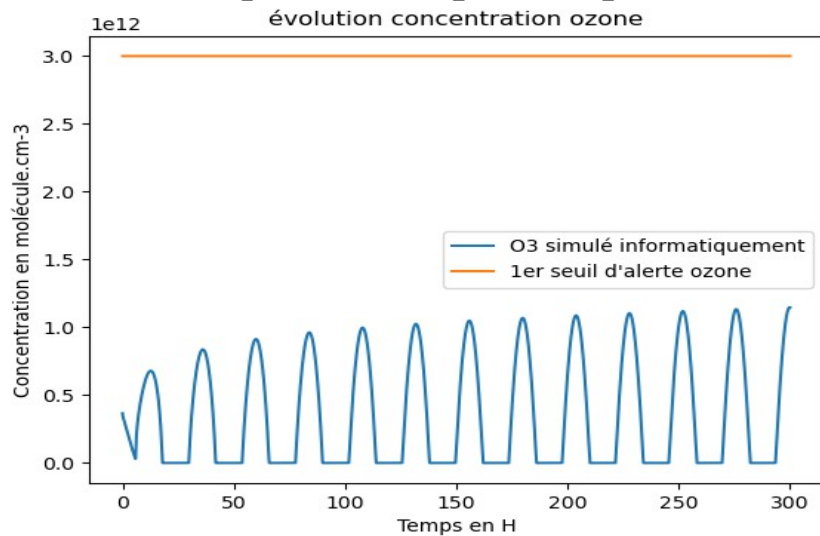


Voitures diesel = mieux pour éviter les pics de pollution à l'ozone mais émet plus de particule fine

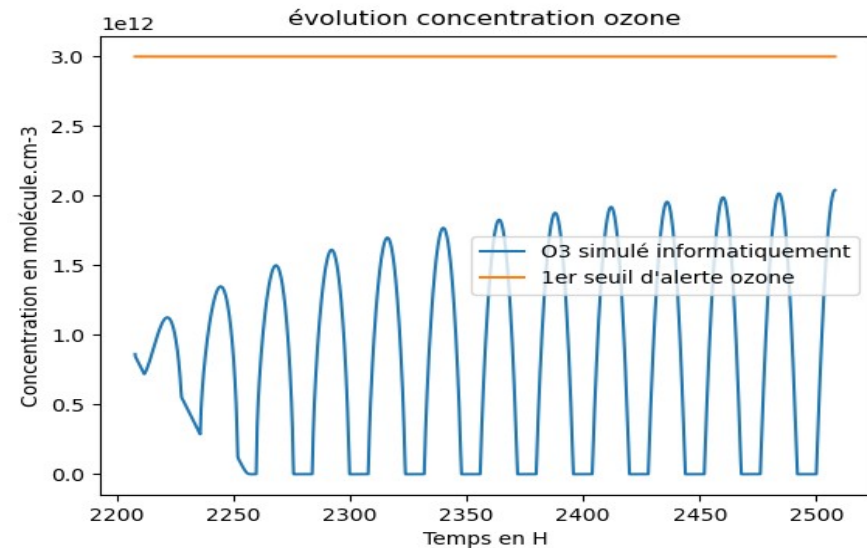
III) Modélisation cycle de Chapman avec émission

- Modélisation cycle Chapman cas réel en fonction des saisons :

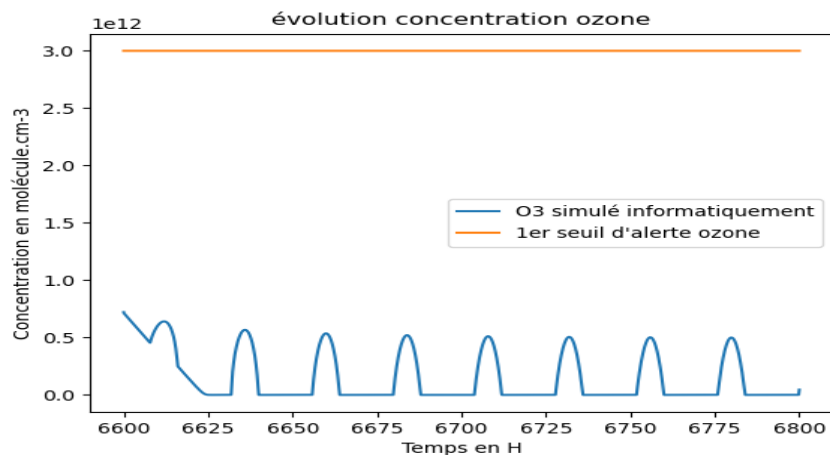
Équinoxe de printemps



Solstice d'été



Solstice d'hiver



Conclusion :

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

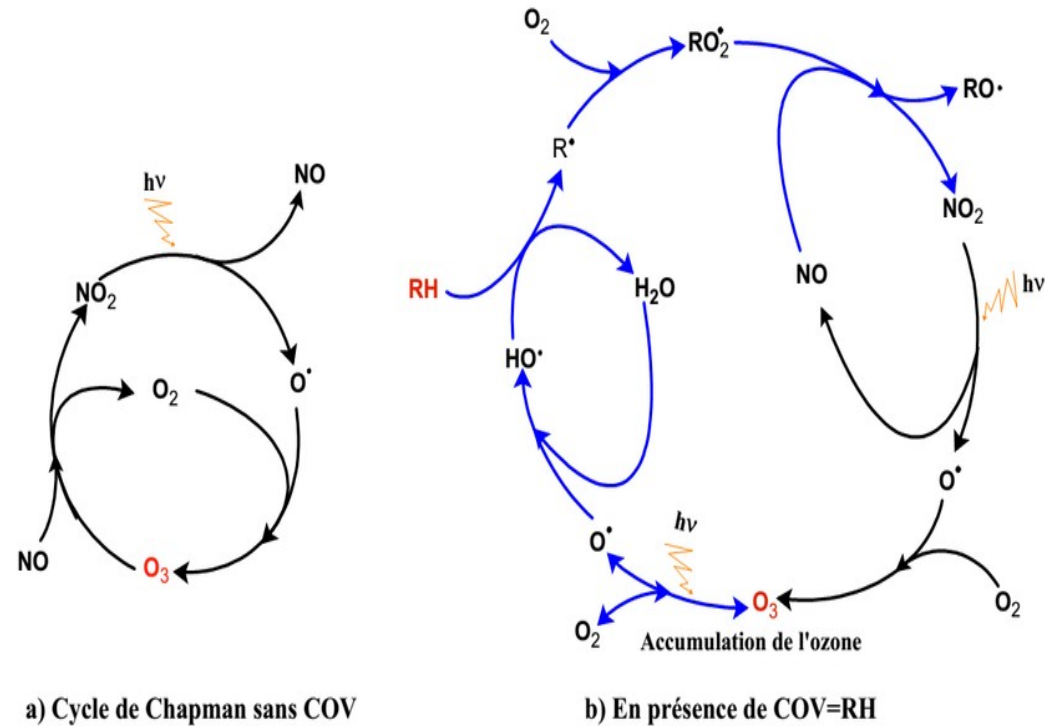
Seuil d'alerte jamais atteint

➡ 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



Source: researchgate.net

ANNEXE :

● Programme python :

```
1  ### TIPE simulation de la pollution à l'ozone à Melun
2
3  ## Lecture de fichier données expérimentales de la station météo de
   Melun-Villaroche grâce à Airparif
4
5  import matplotlib.pyplot as plt
6
7  plt.close('all')
8
9  fichier = open("C:\\Users\\aches\\Documents\\TIPE\\
   \\TIPE_mesure.txt", "r")
10
11  NO=[]
12  NO2=[]
13  O3=[]
14  Dates=[]
15  for Ligne in fichier:
16      L=Ligne.split(",")
17      if len(L)==5:
18          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 sera pas utilisée dans la simulation
19              NO2.append(0)
20          else:
21              NO2.append(L[1])
22          if L[2]=='':
23              NO.append(0)
24          else:
25              NO.append(L[2])
26          if L[3]=='':
27              O3.append(0)
28          else:
29              O3.append(L[3])
30          Dates.append(L[0])
31
32  D=[]
33  H=[]
34  for k in range(len(Dates)):
35      Date=Dates[k].split()
36      D.append(Date[0])
37      H.append(Date[1])
38
```

```
39  H1=[]
40  for k in range(len(H)):
41      Temps=H[k].split(":")
42      H1.append(float(Temps[0]))
43
44  Jour=[]
45  Mois=[]
46  for k in range(len(D)):
47      J=D[k].split("/")
48      Jour.append(int(J[2]))
49      Mois.append(int(J[1]))
50
51
52  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+30,3
   1+28+30+31+30+31+31+31+30+31+30+31] #jour en plus du mois i
53
54  H2=[]
55  for k in range(len(Jour)):
56      i=Mois[k]
57      H=[24*(float(Jour[k])-1)+H1[k]+24*(M[i-1])]
58      H2.append(H)
59
60  CNO2=[]
61  CNO=[]
62  CO3=[]
63  H3=[]
64  for k in range(len(NO2)):
65      CNO2.append(float(NO2[k]))
66      CNO.append(float(NO[k]))
67      CO3.append(float(O3[k]))
68      H3.append(H2[k])
69
70  def f_affiche(fig_i,X,Y,legende,abscisse,ordonnée,titre):
71      plt.figure(fig_i)
72      plt.plot(X,Y,label=legende)
73      plt.xlabel(abscisse)
74      plt.ylabel(ordonnée)
75      plt.title(titre)
76      plt.legend()
77      plt.show()
78
```

ANNEXE :

● Programme python :

```
79 f_affiche(0,H3,CN02,"N02","Temps","concentration en mg.m^-3","")
80 f_affiche(0,H3,CN0,"NO","Temps","concentration en mg.m^-3","")
81 f_affiche(0,H3,C03,"O3","Temps en H","concentration en
mg.m^-3","Evolution concentration ozone")
82
83 ## 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
84
85 import matplotlib.pyplot as plt
86
87 plt.close('all')
88
89 patient_malade=0
90 patient_mort=0
91 Mal=[]
92 Mort=[]
93 for k in range(len(C03)):
94     mal=(0.56+0.198-0.259)*4*C03[k]/(1000*24) #(0.56+0.198-0.259)
pour 10 millions d'habitants par jour or il y a 40000 habitants à
Melun
95     mort=(-0.066+0.124+0.061)*4*C03[k]/(1000*24)
96     patient_malade+=mal
97     patient_mort+=mort
98     Mal.append(patient_malade)
99     Mort.append(patient_mort)
100
101 patient_malade_idf=0
102 patient_mort_idf=0
103 Mal_idf=[]
104 Mort_idf=[]
105 for k in range(len(C03)):
106     mal=(0.56+0.198-0.259)*1.22*C03[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
107     mort=(-0.066+0.124+0.061)*1.22*C03[k]/24
108     patient_malade_idf+=mal
109     patient_mort_idf+=mort
110     Mal_idf.append(patient_malade_idf)
111     Mort_idf.append(patient_mort_idf)
112
113 # 660361 morts en France en 2021 en France selon l'INSEE sur 67.4
millions d'habitants
114 # 12.9 millions d'admissions à l'hôpital en France toutes causes
confondues en 2021
115
116 mort_IDF=660361*12.2/67.4 # Calcul moyenné par rapport à la
population en Ile de France avec 12.2 millions d'individus
117 mort_Melun=660361*4/6740
118 admission_IDF=12900000*10/67.4
119 admission_Melun=12900000*4/6740
120 m_idf=mort_IDF/len(C03) #mort par heure en moyenne
121 m_melun=mort_Melun/len(C03)
122 a_idf=admission_IDF/len(C03) #admission par heure en moyenne
123 a_melun=admission_Melun/len(C03)
124 M_IDF=[]
125 M_Melun=[]
126 A_IDF=[]
127 A_Melun=[]
128 for k in range(len(C03)):
129     m_i=k*m_idf # nombre de mort au bout de k heures
130     m_m=k*m_melun
131     a_i=k*a_idf
132     a_m=k*a_melun
133     M_IDF.append(m_i)
134     M_Melun.append(m_m)
135     A_IDF.append(a_i)
136     A_Melun.append(a_m)
137
138 # Melun
139
140 print("patient malade et hospitalisé à cause de l'ozone à Melun en
2021:",int(patient_malade))
141 print("patient mort à cause de l'ozone à Melun en
2021:",int(patient_mort))
142 print("admission hôpital à Melun toutes causes confondues en
2021:",int(admission_Melun))
143 print("taux d'hospitalisation à cause de la pollution à l'ozone à
Melun",(patient_malade/admission_Melun)*100,"%")
144 print("décès toutes causes confondues à Melun en
2021:",int(mort_Melun))
145 print("taux de décès à cause de la pollution à l'ozone à Melun en
2021",(patient_mort/mort_Melun)*100,"%")
146
147 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")
148 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 :

```
149 #f_affiche(1,H3,A_Melun,"admission hôpital à Melun en
2021","H","nombre de personnes","Evolution sanitaire pour la ville
de Melun")
150 #f_affiche(1,H3,M_Melun,"décès à Melun en 2021","H","nombre de
personnes","Evolution sanitaire pour la ville de Melun")
151
152 # IDF
153
154 print("patient malade et hospitalisé à cause de l'ozone en IDF en
2021:",int(patient_malade_idf))
155 print("patient mort à cause de l'ozone en IDF en
2021:",int(patient_mort_idf))
156 print("admission hôpital en IDF toutes causes confondues en
2021:",int(admission_IDF))
157 print("taux d'hospitalisation à cause de la pollution à l'ozone en
IDF",(patient_malade_idf/admission_IDF)*100,"%")
158 print("décès toutes causes confondues en IDF en
2021:",int(mort_IDF))
159 print("taux de décès à cause de la pollution à l'ozone en IDF en
2021",(patient_mort_idf/mort_IDF)*100,"%")
160
161 #f_affiche(1,H3,MaI_idf,"admission hôpital en IDF à cause de l'ozone
en 2021","H","","")
162 #f_affiche(1,H3,Mort_idf,"décès en IDF à cause de l'ozone en
2021","H","","")
163 #f_affiche(1,H3,A_IDF,"admission hôpital en IDF en 2021","H","","")
164 #f_affiche(1,H3,M_IDF,"décès en IDF en 2021","H","nombre de
personnes","Evolution sanitaire en IDF")
165
166 ## Influence des différents paramètres météorologiques (Température,
ensolleillement) sur la concentration d'ozone à Melun en Janvier
2021
167
168 import matplotlib.pyplot as plt
169
170 plt.close('all')
171
172 H4=[H3[k]for k in range(745)] # Liste des heures au mois de Janvier
173 k03=[CO3[k]/6 for k in range(745)] # calibrage pour mieux voir
l'influence des paramètres sur la courbe
174 plt.figure(2)
175 plt.plot(H4,k03,'y',label="concentration d'ozone")
176 plt.show()
177 #f_affiche(2,H4,k03,"concentration d'ozone","heure de
prélèvement","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")
178
179 # Impact de la Température
180
181 fichier=open("C:\\Users\\aches\\.conda\\TIPE_meteo.txt","r")
182 Tmax=[]
183 Tmin=[]
184 Tmoy=[]
185 Jour=[]
186 for Ligne in fichier:
187     L=Ligne.split()
188     if len(L)==7:
189         for k in range(24):
190             Tmax.append(float(L[2]))
191             Tmin.append(float(L[1]))
192             Tmoy.append(float(L[3]))
193             Jour.append(float(L[0])*24+(1/24)*k)
194
195 #f_affiche(2,Jour,Tmax,"Température max","","","")
196 #f_affiche(2,Jour,Tmin,"Température min","","","")
197 f_affiche(2,Jour,Tmoy,"Température moyenne","jour de janvier
2021","Température (°C)  ensolleillemnt (H)","évolution des
températures à Melun en janvier 2021")
198
199 # Impact de l'ensolleillement
200
201 fichier=open("C:\\Users\\aches\\.conda\\TIPE_meteo.txt","r")
202
203 Sh=[]
204 Sm=[]
205 Jour=[]
206 for Ligne in fichier:
207     L=Ligne.split()
208     if len(L)==7:
209         for k in range(24):
210             Sh.append(L[5])
211             Sm.append(L[6])
212             Jour.append(float(L[0])*24+(1/24)*k)
213
214 Sh1=[]
215 Sm1=[]
```


ANNEXE :

• Programme python :

```
216 for k in range(len(Sh)):
217     heure=Sh[k].split("h")
218     minute=Sm[k].split("min")
219     Sh1.append(float(heure[0]))
220     Sm1.append(float(minute[0]))
221
222 S=[]
223 for k in range(len(Sh1)):
224     s=Sh1[k]*60+Sm1[k]
225     S.append(s/60)
226
227 plt.figure(2)
228 plt.plot(Jour,S,'ro',label='heure ensoleillement en 1 journée')
229 plt.legend()
230 plt.show()
231
232 ## Calcul de l'angle zénithal théta au cours du temps à Melun, 1ère
approximation sans tenir compte des saisons référentiel géocentrique
233
234 import numpy as np
235 from math import cos,sin,pi
236 from matplotlib import pyplot as plt
237 plt.close('all')
238
239 #Données:
240
241 latitude=45*pi/180 # en radians
242 longitude=2.66*pi/180 # en radians
243 w=2*pi/86164 # pulsation rotation de la Terre
244 omega=2*pi/(365*60*60*24) # pulsation de révolution de la Terre
    autour du soleil
245 T=np.linspace(0,8759,8760) # Liste des temps en H à partir de 12H
    heure de Greenwich à l'équinoxe de printemps
246
247 def phi(T):
248     n=len(T)
249     phit=[]
250     for k in range(n):
251         phi=w*T[k]*60*60+longitude
252         phit.append(phi)
253     return phit
254
```

```
255 def teta(T):
256     tetat=[]
257     n=len(T)
258     phit=phi(T)
259     for k in range(n):
260         ur=[cos(latitude)*cos(phit[k]),cos(latitude)*sin(phit[k]),si
n(latitude)]
261         us=[-cos(omega*T[k]*60*60),-sin(omega*T[k]*60*60),0]
262         pscalaire=ur[0]*us[0]+ur[1]*us[1]+ur[2]*us[2]
263         teta=pscalaire # angle en radians
264         tetat.append(teta)
265     return tetat
266
267 tetat=teta(T)
268 plt.figure(2)
269 plt.plot(T,tetat,label="angle théta en radian")
270 plt.xlabel("Temps en heures")
271 plt.ylabel("radians")
272 plt.legend()
273 plt.show()
274
275 ## Calcul de l'angle zénithal théta au cours du temps à Melun, 2ème
approximation prise en compte des saisons en référentiel
héliocentrique
276
277 import numpy as np
278 from math import cos,sin,pi,acos
279 from matplotlib import pyplot as plt
280 plt.close('all')
281
282 #Données:
283
284 latitude=48.5*pi/180 # en radians
285 longitude=2.66*pi/180 # en radians
286 w=2*pi/86164 # pulsation rotation de la Terre en radians.s-1
287 alpha0=23.45*pi/180 # angle plan écliptique/plan équatorial en
    radians
288 omega=2*pi/(365*24) # pulsation de révolution de la Terre autour du
    soleil en radians.h-1
289
290
291 def fphi(t):
292     return w*t*60*60+longitude
293
```

ANNEXE :

• Programme python :

```
328 ## Méthode d'Euler cycle de Chapman soleil uniquement sans polluant
329
330 plt.close('all')
331 import numpy as np
332 from math import cos,sin,pi,acos,exp
333 from matplotlib import pyplot as plt
334
335 #Données:
336
337 Tp=298 # Température en K
338 Na=6.02*10**23 # en mol-1
339 a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
340 c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
341 d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
342 M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
343 j03=3*10**-5 # en s-1
344 j02=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
345 x0=0*Na*10**-12*C03[1896]/48 # concentration en molécules.cm-3 à
    l'équinoxe du printemps (20 mars 2021)
346 y0=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène
    de l'air
347 z0=0 #hypothèse pas d'oxygène atomique initialement
348 V0=np.array([x0,y0,z0])
349
350 # Simulation
351
352 def fJ03(t):
353     return j03*cos(fteta(t))
354
355 def fJ02(t):
356     return j02*cos(fteta(t))
357
358 def F(V,t):
359     x,y,z=V
360     dx=(c*M*y*z-fJ03(t)*x-d*z*x)*3600
361     dy=(a*M*z**2+2*d*z*x+fJ03(t)*x-fJ02(t)*y-c*M*y*z)*3600
362     dz=(2*fJ02(t)*y+fJ03(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)
363     return np.array([dx,dy,dz])
364
```

```
298
299 def fteta(t):
300     ur=[cos(latitude)*cos(fphi(t)),cos(latitude)*sin(fphi(t)),sin(la
    titude)]
301     us=[-cos(omega*t),-
    cos(alpha0)*sin(omega*t),sin(alpha0)*sin(omega*t)]
302     pscalaire=ur[0]*us[0]+ur[1]*us[1]+ur[2]*us[2]
303     costeta=pscalaire # cosinus angle zénithal en radians
304     return acos(costeta*(costeta>=0))
305
306 def falt(t):
307     ur=[cos(latitude)*cos(fphi(t)),cos(latitude)*sin(fphi(t)),sin(la
    titude)]
308     us=[-cos(omega*t),-
    cos(alpha0)*sin(omega*t),sin(alpha0)*sin(omega*t)]
309     pscalaire=ur[0]*us[0]+ur[1]*us[1]+ur[2]*us[2]
310     costeta=pscalaire # cosinus angle zénithal en radians
311     return ((pi/2)-acos(costeta*(costeta>=0)))*180/pi
312
313 # Affichage:
314
315 T=np.linspace(0,8759,8760) # Liste des temps en H à partir de 12H
    heure de Greenwich à l'équinoxe de printemps
316
317 teta=[fteta(t) for t in T]
318 alti=[falt(t) for t in T]
319 shiver=[falt(t) for t in range(6600,6625)] #jour solstice d'hiver
320 sete=[falt(t) for t in range(2208,2233)] #jour solstice d'été
321 TH=[i for i in range(25)]
322
323 plt.close('all')
324 plt.figure(3)
325 #plt.plot(T,teta,label="angle zénithal en radians")
326 #plt.plot(T,alti,label="angle altitude solaire en degré")
327 plt.plot(TH,sete,label="angle altitude solaire au solstice d'été en
    degré")
328 plt.plot(TH,shiver,label="angle altitude solaire au solstice d'hiver
    en degré")
329 plt.xlabel("temps en H")
330 plt.ylabel("degré")
331 plt.legend()
332 plt.show()
333
```

ANNEXE :

• Programme python :

```
365 def Euler_Explicite(f,y0,t0,dt,t1):
366     y=y0
367     Y=[y]
368     t=t0
369     T=[t]
370     while t<t1:
371         t+=dt
372         yp=f(y,t)
373         y=y+yp*dt
374         Y.append(y)
375         T.append(t)
376     return T,Y
377
378 # Résolution
379
380 t0=0
381 dt=1/100000 # dt=1/100000000 pour éviter la divergence d'euler mais
382             # temps de calcul énorme (3 min pour 4s de simulation)
383 t1=12
384 T1,Y=Euler_Explicite(F,V0,t0,dt,t1)
385 Y=np.array(Y)
386 L03=Y[:,0]
387 L02=Y[:,1]
388 L0=Y[:,2]
389 LJO3=[fJO3(t) for t in T1]
390 plt.figure(4)
391 plt.plot(T1,L03,label="O3")
392 #plt.plot(T1,LJO3,label="JO3")
393 plt.legend()
394 plt.show()
395
396 ## Méthode d'Euler cycle de Chapman avec polluant sans émission des
397 voitures
398 plt.close('all')
399 import numpy as np
400 from math import cos,sin,pi,acos,exp
401
402 # Données:
403
404 Tp=298 # Température en K
405 Na=6.02*10**23 # en mol-1
406 a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
407 c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
408 d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
409 k=1.8*10**-14 # en cm3.molécules-1.s-1
410 M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
411 j03=3*10**-5 # en s-1
412 j02=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
413 jN02=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
414 o30=Na*10**-12*CO3[1896]/48 # concentration en molécules.cm-3 à
415   l'équinoxe du printemps (20 mars 2021)
416 o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène
417   dans l'air
418 o0=0 #hypothèse pas d'oxygene atomique initialement dans l'air
419 no0=Na*10**-12*CN0[1896]/30 # concentration en molécules.cm-3 à
420   l'équinoxe du printemps (20 mars 2021)
421 no20=Na*10**-12*CN02[1896]/46 # concentration en molécules.cm-3 à
422   l'équinoxe du printemps (20 mars 2021)
423 V0=np.array([o30,o20,o0,no0,no20]) # condition initiale
424
425 # Simulation:
426
427 def fJO3(t):
428     return j03*cos(fteta(t))
429
430 def fJO2(t):
431     return j02*cos(fteta(t))
432
433 def fJN02(t):
434     return jN02*cos(fteta(t))
435
436 def F(V,t):
437     o3,o2,o,no,no2=V
438     v1=fJN02(t)*no2
439     v2=no*o3*k
440     do3=(c*M*o2*o-fJO3(t)*o3-d*o*o3+v1-v2)*3600
441     do2=(a*M*o**2+2*d*o*o3+fJO3(t)*o3-fJO2(t)*o2-c*M*o2*o-
442           v1+v2)*3600
443     do=(2*fJO2(t)*o2+fJO3(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600 #
444     Divergence d'euler pas de temps trop grand
445     dno=(v1-v2)*3600
446     dno2=(v2-v1)*3600
447     return np.array([do3,do2,do,dno,dno2])
```

ANNEXE :

● Programme python :

```
442 def Euler_Explicite(f,y0,t0,dt,t1):
443     y=y0
444     Y=[y]
445     t=t0
446     T=[t]
447     while t<t1:
448         t+=dt
449         yp=f(y,t)
450         y=y+yp*dt
451         Y.append(y)
452         T.append(t)
453     return T,Y
454
455 # Résolution
456 t0=0
457 dt=1/100000
458 t1=12
459 T1,Y=Euler_Explicite(F,V0,t0,dt,t1)
460 Y=np.array(Y)
461 L03=Y[:,0]
462 L02=Y[:,1]
463 L0=Y[:,2]
464 LNO=Y[:,3]
465 LNO2=Y[:,4]
466
467 plt.figure(5)
468 plt.plot(T1,L03,label="O3 simulé informatiquement")
469 #plt.plot(T1,L02,label="O2 simulation")
470 #plt.plot(T1,L0,label="O simulation")
471 #plt.plot(T1,LNO2,label="NO2 simulation")
472 #plt.plot(T1,LNO,label="NO simulation")
473 plt.xlabel("Temps en H")
474 plt.ylabel("Concentration en molécule.cm-3")
475 plt.title("évolution concentration ozone")
476 plt.legend()
477 plt.show()
478
479
480
481 ## Méthode d'Euler cycle de Chapman avec polluant sans émission des
482 voitures
483
484 plt.close('all')
485 import numpy as np
486 from math import cos,sin,pi,acos,exp
487 from matplotlib import pyplot as plt
488
489 # Données:
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 j03=3*10**-5 # en s-1 pour un angle zénithal de 0°
498 j02=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
501 l'équinoxe du printemps (20 mars 2021)
502 o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène
503 dans l'air
504 o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans
505 l'air
506 no0=Na*10**-12*CN0[1896]/30 # concentration en molécules.cm-3 à
507 l'équinoxe du printemps (20 mars 2021)
508 no20=Na*10**-12*CN02[1896]/46 # concentration en molécules.cm-3 à
509 l'équinoxe du printemps (20 mars 2021)
510 V0=np.array([o30,o20,o0,no0,no20]) # condition initiale
511
512 # Simulation:
513
514 def fJ03(t):
515     return j03*cos(fteta(t))
516
517 def fJ02(t):
518     return j02*cos(fteta(t))
519
520 def fJNO2(t):
521     return jNO2*cos(fteta(t))
522
```

ANNEXE :

● Programme python :

```
518 def F(V,t):
519     o3,o2,o,no,no2=V
520     v1=fJN02(t)*no2
521     v2=no*o3*k
522     do3=(c*M*o2*o-fJ03(t)*o3-d*o*o3+v1-v2)*3600
523     do2=(a*M*o**2+2*d*o*o3+fJ03(t)*o3-fJ02(t)*o2-c*M*o2*o-
v1+v2)*3600
524     do=(2*fJ02(t)*o2+fJ03(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 L03=Y[:,0]
550 L02=Y[:,1]
551 L0=Y[:,2]
552 LN0=Y[:,3]
553 LN02=Y[:,4]
554
555 plt.figure(6)
556 plt.plot(T1,L03,label="O3 simulé informatiquement")
557 #plt.plot(T1,L02,label="O2 simulation")
558 #plt.plot(T1,L0,label="O simulation")
559 #plt.plot(T1,LN02,label="NO2 simulation")
560 #plt.plot(T1,LN0,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 j03=3*10**-5 # en s-1
584 j02=10**-12 # en s-1 hypothese lineaire de la figure 3
585 jN02=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*CN0[1896]/30 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
590 no20=Na*10**-12*CN02[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 fJ03(t):
596     return j03*cos(fteta(t))
597
```


ANNEXE :

● Programme python :

```
598 def fJ02(t):
599     return j02*cos(fteta(t))
600
601 def fJN02(t):
602     return jN02*cos(fteta(t))
603
604 def F(V,t):
605     o3,o2,o,no,no2=V
606     v1=fJN02(t)*no2
607     v2=no*o3*k
608     do3=(c*M*o2*o-fJ03(t)*o3-d*o*o3+v1-v2)*3600
609     do2=(a*M*o**2+2*d*o*o3+fJ03(t)*o3-fJ02(t)*o2-c*M*o2*o-
v1+v2)*3600
610     do=(2*fJ02(t)*o2+fJ03(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600 #
Même problème de divergence que précédemment
611     dno=(v1-v2)*3600
612     dno2=(v2-v1)*3600
613     return np.array([do3,do2,do,dno,dno2])
614
615 def Rk4(f,y0,t0,dt,t1):
616     y=y0
617     Y=[y]
618     t=t0
619     T=[t]
620     while t<t1:
621         t+=dt
622         k1=f(y,t)
623         k2=f(y+dt*k1/2,t+dt/2)
624         k3=f(y+k2*dt/2,t+dt/2)
625         k4=f(y+dt*k3,t+dt)
626         y=y+dt*(k1+2*k2+2*k3+k4)/6
627         Y.append(y)
628         T.append(t)
629     return T,Y
630
631 # Résolution
632
633 t0=0
634 dt=1/100000
635 t1=1
```

```
636 T1,Y=Rk4(F,V0,t0,dt,t1)
637 Y=np.array(Y)
638 L03=Y[:,0]
639 L02=Y[:,1]
640 L0=Y[:,2]
641 LNO=Y[:,3]
642 LNO2=Y[:,4]
643
644 plt.figure(7)
645 plt.plot(T1,L03,label="O3 simulé informatiquement")
646 #plt.plot(T1,L02,label="O2")
647 #plt.plot(T1,L0,label="O")
648 #plt.plot(T1,LNO2,label="NO2")
649 #plt.plot(T1,LNO,label="NO")
650 plt.xlabel("Temps en H")
651 plt.ylabel("Concentration en molécule.cm-3")
652 plt.title("évolution concentration ozone")
653 plt.legend()
654 plt.show()
655
656 ## Résolution odeint sans émission
657
658 plt.close('all')
659 import numpy as np
660 import matplotlib.pyplot as plt
661 from scipy.integrate import odeint
662 from math import cos,sin,pi,acos,exp
663
664 #Données:
665
666 Tp=298 # Température en K
667 Na=6.02*10**23 # en mol-1
668 a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
669 c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
670 d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
671 k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
672 M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
673 j03=3*10**-5 # en s-1 pour un angle zénithal de 0°
674 j02=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
675 jN02=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
676 o30=Na*10**-12*C03[1896]/48 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
677 o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène
dans l'air
678 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*CN02[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 fJ03(t):
686     return j03*cos(fteta(t))
687
688 def fJ02(t):
689     return j02*cos(fteta(t))
690
691 def fJN02(t):
692     return jN02*cos(fteta(t))
693
694 def F(V,t):
695     o3,o2,o,no,no2=V
696     v1=fJN02(t)*no2
697     v2=no*o3*k
698     do3=(c*M*o2*o-fJ03(t)*o3-d*o*o3+v1-v2)*3600
699     do2=(a*M*o**2+2*d*o*o3+fJ03(t)*o3-fJ02(t)*o2-c*M*o2*o-
    v1+v2)*3600
700     do=(2*fJ02(t)*o2+fJ03(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 L03=Y[:,0]
711 L02=Y[:,1]
712 L0=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*CN02[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,L03,label="O3 simulé informatiquement")
741 #plt.plot(T1,L02,label="O2 simulation")
742 #plt.plot(T1,L0,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 :

```
757 ## Résolution odeint avec émission voiture 40% essence/60% diesel
758
759 plt.close('all')
760 import numpy as np
761 import matplotlib.pyplot as plt
762 from scipy.integrate import odeint
763 from math import cos,sin,pi,acos,exp
764
765 #Données:
766
767 Tp=298 # Température en K
768 Na=6.02*10**23 # en mol-1
769 a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
770 c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
771 d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
772 k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
773 M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
774 j03=3*10**-5 # en s-1 pour un angle zénithal de 0°
775 j02=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
776 jN02=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
777 o30=Na*10**-12*C03[1896]/48 # concentration en molécules.cm-3 à
778   l'équinoxe du printemps (20 mars 2021)
779 o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène
780   dans l'air
781 o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans
782   l'air
783 no0=Na*10**-12*CNO[1896]/30 # concentration en molécules.cm-3 à
784   l'équinoxe du printemps (20 mars 2021)
785 no20=Na*10**-12*CNO2[1896]/46 # concentration en molécules.cm-3 à
786   l'équinoxe du printemps (20 mars 2021)
787 V0=np.array([o30,o20,o0,no0,no20]) # condition initiale
788
789 # Hypothèse pour les émissions: temps calme et polluant évoluant
790   dans un volume de 100m d'altitude et de surface la superficie de la
791   petite couronne parisienne
792 eno2=Na*10**-12*180/(46*24) # emission de NO2 du au trafic routier
793   en petite couronne en molécule.cm-3.h-1
794 eno=Na*10**-12*40/(30*24) # emission de NO du au trafic routier en
795   petite couronne en molécule.cm-3.h-1
796
797 #Simulation:
798
799 def fJ03(t):
800     return j03*cos(fteta(t))
801
802 def fJ02(t):
803     return j02*cos(fteta(t))
804
805 def fJN02(t):
806     return jN02*cos(fteta(t))
807
808 def F(V,t):
809     o3,o2,o,no,no2=V
810     v1=fJN02(t)*no2
811     v2=no*o3*k
812     do3=(c*M*o2*o-fJ03(t)*o3-d*o*o3+v1-v2)*3600
813     do2=(a*M*o**2+2*d*o*o3+fJ03(t)*o3-fJ02(t)*o2-c*M*o2*o-
814           v1+v2)*3600
815     do=(2*fJ02(t)*o2+fJ03(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
816     dno=(v1-v2)*3600+eno
817     dno2=(v2-v1)*3600+eno2
818     return np.array([do3,do2,do,dno,dno2])
819
820 # Résolution
821
822 T1=np.linspace(0,100,100000)
823 Y=odeint(F,V0,T1)
824 Y=np.array(Y)
825 L03=Y[:,0]
826 L02=Y[:,1]
827 L0=Y[:,2]
828 LNO=Y[:,3]
829 LNO2=Y[:,4]
830
831 # Liste des concentrations réellement mesuré
832
833 TT=[k for k in range(101)]
834 CrO3=[]
835 CrNO=[]
836 CrNO2=[]
837 for k in range(len(TT)):
838     CrO3.append(10**-12*Na*C03[1896+k]/48)
839     CrNO.append(10**-12*Na*CNO[1896+k]/30)
840     CrNO2.append(10**-12*Na*CNO2[1896+k]/46)
841
842 # 1er seuil d'alerte pollution ozone: 240 microgramme.m-3 (liste A1)
843 # 2ème seuil d'alerte pollution ozone: 300 microgramme.m-3 (liste
844   A2)
```


ANNEXE :

• Programme python :

```
834 # 3ème seuil d'alerte pollution ozone: 360 microgramme.m-3 (liste
A3)
835
836 A1=[]
837 A2=[]
838 A3=[]
839 for k in range(len(T1)):
840     A1.append(30*10**11) #conversion en molécules.cm-3
841     A2.append(37*10**11)
842     A3.append(45*10**11)
843
844 plt.figure(9)
845 plt.plot(T1,L03,label="O3 simulé informatiquement")
846 #plt.plot(T1,L02,label="O2 simulation")
847 #plt.plot(T1,L0,label="O simulation")
848 #plt.plot(T1,LN02,label="NO2 simulation")
849 #plt.plot(T1,LN0,label="NO simulation")
850 plt.plot(TT,CrO3,'ro',label="O3 réel")
851 #plt.plot(TT,CrNO,'ro',label="NO réel")
852 #plt.plot(TT,CrNO2,'ro',label="NO2 réel")
853 #plt.plot(T1,A1,label="1er seuil d'alerte ozone")
854 #plt.plot(T1,A2,label="2eme seuil d'alerte ozone")
855 #plt.plot(T1,A3,label="3eme seuil d'alerte ozone")
856 plt.xlabel("Temps en H")
857 plt.ylabel("Concentration en molécule.cm-3")
858 plt.title("évolution concentration ozone")
859 plt.legend()
860 plt.show()
861
862 ## Odeint avec émission solstice été
863
864 plt.close('all')
865 import numpy as np
866 import matplotlib.pyplot as plt
867 from scipy.integrate import odeint
868 from math import cos,sin,pi,acos,exp
869
870 #Données:
871
872 Tp=298 # Température en K
873 Na=6.02*10**23 # en mol-1
874 a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
875 c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
876 d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
877 k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
878 M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
879 j03=3*10**-5 # en s-1 pour un angle zénithal de 0°
880 j02=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
881 jN02=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
882 o30=Na*10**-12*CO3[2208]/48 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
883 o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène
dans l'air
884 o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans
l'air
885 no0=Na*10**-12*CN0[2208]/30 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
886 no20=Na*10**-12*CN02[2208]/46 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
887 V0=np.array([o30,o20,o0,no0,no20]) # condition initiale
888
889 # 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
890 eno2=Na*10**-12*180/(46*24) # emission de NO2 du au trafic routier
en petite couronne en molécule.cm-3.h-1
891 eno=Na*10**-12*40/(30*24) # emission de NO du au trafic routier en
petite couronne en molécule.cm-3.h-1
892
893 #Simulation:
894
895 def fJ03(t):
896     return j03*cos(fteta(t))
897
898 def fJ02(t):
899     return j02*cos(fteta(t))
900
901 def fJN02(t):
902     return jN02*cos(fteta(t))
903
```

ANNEXE :

● Programme python :

```
904 def F(V,t):
905     o3,o2,o,no,no2=V
906     v1=fJN02(t)*no2
907     v2=no*o3*k
908     do3=(c*M*o2*o-fJ03(t)*o3-d*o*o3+v1-v2)*3600
909     do2=(a*M*o**2+2*d*o*o3+fJ03(t)*o3-fJ02(t)*o2-c*M*o2*o-
v1+v2)*3600
910     do=(2*fJ02(t)*o2+fJ03(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
911     dno=(v1-v2)*3600+eno
912     dno2=(v2-v1)*3600+eno2
913     return np.array([do3,do2,do,dno,dno2])
914
915 # Résolution
916
917 T1=np.linspace(2208,2508,100000)
918 Y=odeint(F,V0,T1)
919 Y=np.array(Y)
920 L03=Y[:,0]
921 L02=Y[:,1]
922 L0=Y[:,2]
923 LNO=Y[:,3]
924 LNO2=Y[:,4]
925
926 # Liste des concentrations réellement mesuré
927
928 TT=[k for k in range(301)]
929 CrO3=[]
930 CrNO=[]
931 CrNO2=[]
932 for k in range(len(TT)):
933     CrO3.append(10**-12*Na*CO3[2208+k]/48)
934     CrNO.append(10**-12*Na*CNO[2208+k]/30)
935     CrNO2.append(10**-12*Na*CN02[2208+k]/46)
936
937 # 1er seuil d'alerte pollution ozone: 240 microgramme.m-3 (liste A1)
938 # 2ème seuil d'alerte pollution ozone: 300 microgramme.m-3 (liste A2)
939 # 3ème seuil d'alerte pollution ozone: 360 microgramme.m-3 (liste A3)
940
941 A1=[]
942 A2=[]
943 A3=[]
944
945 for k in range(len(T1)):
946     A1.append(30*10**11) #conversion en molécules.cm-3
947     A2.append(37*10**11)
948     A3.append(45*10**11)
949
950 plt.figure(10)
951 plt.plot(T1,L03,label="O3 simulé informatiquement")
952 #plt.plot(T1,L02,label="O2 simulation")
953 #plt.plot(T1,L0,label="O simulation")
954 #plt.plot(T1,LNO2,label="NO2 simulation")
955 #plt.plot(T1,LNO,label="NO simulation")
956 #plt.plot(TT,CrO3,'ro',label="O3 réel")
957 #plt.plot(TT,CrNO,'ro',label="NO réel")
958 #plt.plot(TT,CrNO2,'ro',label="NO2 réel")
959 plt.plot(T1,A1,label="1er seuil d'alerte ozone")
960 #plt.plot(T1,A2,label="2eme seuil d'alerte ozone")
961 #plt.plot(T1,A3,label="3eme seuil d'alerte ozone")
962 plt.xlabel("Temps en H")
963 plt.ylabel("Concentration en molécule.cm-3")
964 plt.title("évolution concentration ozone")
965 plt.legend()
966 plt.show()
967
968 ## Odeint avec émission solstice hiver
969
970 plt.close('all')
971 import numpy as np
972 import matplotlib.pyplot as plt
973 from scipy.integrate import odeint
974 from math import cos,sin,pi,acos,exp
975
976 #Données:
977 Tp=298 # Température en K
978 Na=6.02*10**23 # en mol-1
979 a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
980 c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
981 d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
982 k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
983 M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
```

ANNEXE :

• Programme python :

```
984 j03=3*10**-5 # en s-1 pour un angle zénithal de 0°
985 j02=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
986 jN02=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
987 o30=Na*10**-12*C03[6600]/48 # concentration en molécules.cm-3 à
    l'équinoxe du printemps (20 mars 2021)
988 o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène
    dans l'air
989 o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans
    l'air
990 no0=Na*10**-12*CNO[6600]/30 # concentration en molécules.cm-3 à
    l'équinoxe du printemps (20 mars 2021)
991 no20=Na*10**-12*CNO2[6600]/46 # concentration en molécules.cm-3 à
    l'équinoxe du printemps (20 mars 2021)
992 V0=np.array([o30,o20,o0,no0,no20]) # condition initiale
993
994 # 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
995 eno2=Na*10**-12*180/(46*24) # emission de NO2 du au trafic routier
    en petite couronne en molécule.cm-3.h-1
996 eno=Na*10**-12*40/(30*24) # emission de NO du au trafic routier en
    petite couronne en molécule.cm-3.h-1
997
998 #Simulation:
999
1000 def fJ03(t):
1001     return j03*cos(fteta(t))
1002
1003 def fJ02(t):
1004     return j02*cos(fteta(t))
1005
1006 def fJN02(t):
1007     return jN02*cos(fteta(t))
1008
1009 def F(V,t):
1010     o3,o2,o,no,no2=V
1011     v1=fJN02(t)*no2
1012     v2=no*o3*k
1013     do3=(c*M*o2*o-fJ03(t)*o3-d*o*o3+v1-v2)*3600
1014     do2=(a*M*o**2+2*d*o*o3+fJ03(t)*o3-fJ02(t)*o2-c*M*o2*o-
        v1+v2)*3600
1015     do=(2*fJ02(t)*o2+fJ03(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
1016     dno=(v1-v2)*3600+eno
1017     dno2=(v2-v1)*3600+eno2
1018     return np.array([do3,do2,do,dno,dno2])
1019
1020 # Résolution
1021
1022 T1=np.linspace(6600,6800,100000)
1023 Y=odeint(F,V0,T1)
1024 Y=np.array(Y)
1025 L03=Y[:,0]
1026 L02=Y[:,1]
1027 L0=Y[:,2]
1028 LN0=Y[:,3]
1029 LN02=Y[:,4]
1030
1031 # Liste des concentrations réellement mesuré
1032
1033 TT=[k for k in range(301)]
1034 CrO3=[]
1035 CrNO=[]
1036 CrNO2=[]
1037 for k in range(len(TT)):
1038     CrO3.append(10**-12*Na*C03[6600+k]/48)
1039     CrNO.append(10**-12*Na*CNO[6600+k]/30)
1040     CrNO2.append(10**-12*Na*CNO2[6600+k]/46)
1041
1042 # 1er seuil d'alerte pollution ozone: 240 microgramme.m-3 (liste A1)
1043 # 2ème seuil d'alerte pollution ozone: 300 microgramme.m-3 (liste
    A2)
1044 # 3ème seuil d'alerte pollution ozone: 360 microgramme.m-3 (liste
    A3)
1045
1046 A1=[]
1047 A2=[]
1048 A3=[]
1049 for k in range(len(T1)):
1050     A1.append(30*10**11) #conversion en molécules.cm-3
1051     A2.append(37*10**11)
1052     A3.append(45*10**11)
1053
1054 plt.figure(11)
1055 plt.plot(T1,L03,label="O3 simulé informatiquement")
1056 #plt.plot(T1,L02,label="O2 simulation")
1057 #plt.plot(T1,L0,label="O simulation")
1058 #plt.plot(T1,LN02,label="NO2 simulation")
1059 #plt.plot(T1,LN0,label="NO simulation")
1060 #plt.plot(TT,CrO3,'ro',label="O3 réel")
1061 #plt.plot(TT,CrNO,'ro',label="NO réel")
1062 #plt.plot(TT,CrNO2,'ro',label="NO2 réel")
1063 plt.plot(T1,A1,label="1er seuil d'alerte ozone")
```

ANNEXE :

● Programme python :

```
1064 #plt.plot(T1,A2,label="2eme seuil d'alerte ozone")
1065 #plt.plot(T1,A3,label="3eme seuil d'alerte ozone")
1066 plt.xlabel("Temps en H")
1067 plt.ylabel("Concentration en molécule.cm-3")
1068 plt.title("évolution concentration ozone")
1069 plt.legend()
1070 plt.show()
1071
1072 ## Odeint avec émission, influence température négligeable
1073
1074 plt.close('all')
1075 import numpy as np
1076 import matplotlib.pyplot as plt
1077 from scipy.integrate import odeint
1078 from math import cos,sin,pi,acos,exp
1079
1080 #Données:
1081
1082 Tp=310 # Température en K
1083 Na=6.02*10**23 # en mol-1
1084 a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
1085 c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
1086 d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
1087 k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
1088 M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
1089 j03=3*10**-5 # en s-1 pour un angle zénithal de 0°
1090 j02=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
1091 jN02=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
1092 o30=Na*10**-12*C03[1896]/48 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
1093 o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène
dans l'air
1094 o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans
l'air
1095 no0=Na*10**-12*CNO[1896]/30 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
1096 no20=Na*10**-12*CNO2[1896]/46 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
1097 V0=np.array([o30,o20,o0,no0,no20]) # condition initiale
1098
1099 # 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
1100 eno2=Na*10**-12*180/(46*24) # emission de NO2 du au trafic routier
en petite couronne en molécule.cm-3.h-1
1101 eno=Na*10**-12*40/(30*24) # emission de NO du au trafic routier en
petite couronne en molécule.cm-3.h-1
1102
1103 #Simulation:
1104
1105 def fJ03(t):
1106     return j03*cos(fteta(t))
1107
1108 def fJ02(t):
1109     return j02*cos(fteta(t))
1110
1111 def fJN02(t):
1112     return jN02*cos(fteta(t))
1113
1114 def F(V,t):
1115     o3,o2,o,no,no2=V
1116     v1=fJN02(t)*no2
1117     v2=no*o3*k
1118     do3=(c*M*o2*o-fJ03(t)*o3-d*o*o3+v1-v2)*3600
1119     do2=(a*M*o**2+2*d*o*o3+fJ03(t)*o3-fJ02(t)*o2-c*M*o2*o-
v1+v2)*3600
1120     do=(2*fJ02(t)*o2+fJ03(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
1121     dno=(v1-v2)*3600+eno
1122     dno2=(v2-v1)*3600+eno2
1123     return np.array([do3,do2,do,dno,dno2])
1124
1125 # Résolution
1126
1127 T1=np.linspace(0,100,100000)
1128 Y=odeint(F,V0,T1)
1129 Y=np.array(Y)
1130 L03=Y[:,0]
1131 L02=Y[:,1]
1132 L0=Y[:,2]
1133 LNO=Y[:,3]
1134 LNO2=Y[:,4]
1135
1136 # Liste des concentrations réellement mesuré
1137
1138 TT=[k for k in range(101)]
1139 CrO3=[]
1140 CrNO=[]
1141 CrNO2=[]
1142 for i in range(len(TT)):
1143     CrO3.append(10**-12*Na*C03[1896+i]/48)
```

ANNEXE :

● Programme python :

```
1144 CrNO.append(10**(-12*Na*CrNO[1896+i]/30))
1145 CrNO2.append(10**(-12*Na*CrNO2[1896+i]/46))
1146
1147
1148 plt.figure(12)
1149 plt.plot(T1,L03,'g',label="O3 simulé T=310K")
1150 plt.plot(TT,CrO3,'ro',label="O3 réel")
1151 plt.xlabel("Temps en H")
1152 plt.ylabel("Concentration en molécule.cm-3")
1153 plt.title("évolution concentration ozone")
1154 plt.legend()
1155 plt.show()
1156
1157 # Nouvelle données:
1158
1159 Tp=270 # Température en K
1160 a=9.9*exp(470/Tp)*10**(-34) #en cm6.s-1
1161 c=1.1*exp(510/Tp)*10**(-34) #en cm6.s-1
1162 d=1.9*exp(-2300/Tp)*10**(-11) #en cm3.s-1
1163
1164 #Simulation:
1165
1166 def fJ03(t):
1167     return j03*cos(fteta(t))
1168
1169 def fJ02(t):
1170     return j02*cos(fteta(t))
1171
1172 def fJN02(t):
1173     return jN02*cos(fteta(t))
1174
1175 def F(V,t):
1176     o3,o2,o,no,no2=V
1177     v1=fJN02(t)*no2
1178     v2=no*o3*k
1179     do3=(c*M*o2*o-fJ03(t)*o3-d*o*o3+v1-v2)*3600
1180     do2=(a*M*o**2+2*d*o*o3+fJ03(t)*o3-fJ02(t)*o2-c*M*o2*o-
1181     v1+v2)*3600
1182     do=(2*fJ02(t)*o2+fJ03(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
1183     dno=(v1-v2)*3600+eno
1184     dno2=(v2-v1)*3600+eno2
1185     return np.array([do3,do2,do,dno,dno2])
1186
1187 # Résolution
```

```
1188 T1=np.linspace(0,100,100000)
1189 Y=odeint(F,V0,T1)
1190 Y=np.array(Y)
1191 L03=Y[:,0]
1192 L02=Y[:,1]
1193 L0=Y[:,2]
1194 LN0=Y[:,3]
1195 LN02=Y[:,4]
1196
1197 plt.figure(12)
1198 plt.plot(T1,L03,'b',label="O3 simulé T=270K")
1199 plt.xlabel("Temps en H")
1200 plt.ylabel("Concentration en molécule.cm-3")
1201 plt.title("évolution concentration ozone")
1202 plt.legend()
1203 plt.show()
1204
1205 ## Test odeint avec émission voiture 100% essence
1206
1207 plt.close('all')
1208 import numpy as np
1209 import matplotlib.pyplot as plt
1210 from scipy.integrate import odeint
1211 from math import cos,sin,pi,acos,exp
1212
1213 #Données:
1214
1215 Tp=298 # Température en K
1216 Na=6.02*10**23 # en mol-1
1217 a=9.9*exp(470/Tp)*10**(-34) #en cm6.s-1
1218 c=1.1*exp(510/Tp)*10**(-34) #en cm6.s-1
1219 d=1.9*exp(-2300/Tp)*10**(-11) #en cm3.s-1
1220 k=1.8*10**(-14) # constante de vitesse en cm3.molécules-1.s-1
1221 M=Na*1.292*(10**(-3))/29 # concentration en molécules.cm-3 de l'air
1222 j03=3*10**(-5) # en s-1 pour un angle zénithal de 0°
1223 j02=10**(-12) # en s-1 hypothese lineaire de la figure 3 doc ENS
1224 jN02=8.2*10**(-3) # en s-1 pour un angle zénithal de 0°
1225 o30=Na*10**(-12)*CrO3[1896]/48 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
1226 o20=Na*10**(-3)*0.31/32 #concentration en molécules.cm-3 de dioxygène
dans l'air
1227 o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans
l'air
1228 no0=Na*10**(-12)*CrNO[1896]/30 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
```


ANNEXE :

• Programme python :

```
1229 no20=Na*10**-12*CN02[1896]/46 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
1230 V0=np.array([o30,o20,o0,no0,no20]) # condition initiale
1231
1232 # 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
1233 eno2=Na*10**-12*73/(46*24) # emission de N02 du au trafic routier en
petite couronne en molécule.cm-3.h-1
1234 eno=Na*10**-12*2/(30*24) # emission de N0 du au trafic routier en
petite couronne en molécule.cm-3.h-1
1235
1236 #Simulation:
1237
1238 def fJ03(t):
1239     return j03*cos(fteta(t))
1240
1241 def fJ02(t):
1242     return j02*cos(fteta(t))
1243
1244 def fJN02(t):
1245     return jN02*cos(fteta(t))
1246
1247 def F(V,t):
1248     o3,o2,o,no,no2=V
1249     v1=fJN02(t)*no2
1250     v2=no*o3*k
1251     do3=(c*M*o2*o-fJ03(t)*o3-d*o*o3+v1-v2)*3600
1252     do2=(a*M*o**2+2*d*o*o3+fJ03(t)*o3-fJ02(t)*o2-c*M*o2*o-
v1+v2)*3600
1253     do=(2*fJ02(t)*o2+fJ03(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
1254     dno=(v1-v2)*3600+eno
1255     dno2=(v2-v1)*3600+eno2
1256     return np.array([do3,do2,do,dno,dno2])
1257
1258 # Résolution
1259
1260 T1=np.linspace(0,300,100000)
1261 Y=odeint(F,V0,T1)
1262 Y=np.array(Y)
1263 L03=Y[:,0]
1264 L02=Y[:,1]
1265 L0=Y[:,2]
1266 LN0=Y[:,3]
1267 LN02=Y[:,4]
1268
1269 # Liste des concentrations réellement mesuré
1270
1271 TT=[k for k in range(101)]
1272 CrO3=[]
1273 CrN0=[]
1274 CrN02=[]
1275 for k in range(len(TT)):
1276     CrO3.append(10**-12*Na*CO3[1896+k]/48)
1277     CrN0.append(10**-12*Na*CN0[1896+k]/30)
1278     CrN02.append(10**-12*Na*CN02[1896+k]/46)
1279
1280 # 1er seuil d'alerte pollution ozone: 240 microgramme.m-3 (liste A1)
1281 # 2ème seuil d'alerte pollution ozone: 300 microgramme.m-3 (liste
A2)
1282 # 3ème seuil d'alerte pollution ozone: 360 microgramme.m-3 (liste
A3)
1283
1284 A1=[]
1285 A2=[]
1286 A3=[]
1287 for k in range(len(T1)):
1288     A1.append(30*10**11) #conversion en molécules.cm-3
1289     A2.append(37*10**11)
1290     A3.append(45*10**11)
1291
1292 plt.figure(9)
1293 plt.plot(T1,L03,label="O3 simulé voiture essence")
1294 #plt.plot(T1,L02,label="O2 simulation")
1295 #plt.plot(T1,L0,label="O simulation")
1296 #plt.plot(T1,LN02,label="N02 simulation")
1297 #plt.plot(T1,LN0,label="N0 simulation")
1298 #plt.plot(TT,CrO3,'ro',label="O3 réel")
1299 #plt.plot(TT,CrN0,'ro',label="N0 réel")
1300 #plt.plot(TT,CrN02,'ro',label="N02 réel")
1301 plt.plot(T1,A1,label="1er seuil d'alerte ozone")
1302 #plt.plot(T1,A2,label="2eme seuil d'alerte ozone")
1303 #plt.plot(T1,A3,label="3eme seuil d'alerte ozone")
1304 plt.xlabel("Temps en H")
1305 plt.ylabel("Concentration en molécule.cm-3")
1306 plt.title("évolution concentration ozone")
1307 plt.legend()
1308 plt.show()
1309
1310 ## Test odeint avec émission voiture 100% diesel
1311
1312 plt.close('all')
1313 import numpy as np
```

ANNEXE :

● Programme python :

```
1314 import matplotlib.pyplot as plt
1315 from scipy.integrate import odeint
1316 from math import cos,sin,pi,acos,exp
1317
1318 #Données:
1319
1320 Tp=298 # Température en K
1321 Na=6.02*10**23 # en mol-1
1322 a=9.9*exp(470/Tp)*10**-34 #en cm6.s-1
1323 c=1.1*exp(510/Tp)*10**-34 #en cm6.s-1
1324 d=1.9*exp(-2300/Tp)*10**-11 #en cm3.s-1
1325 k=1.8*10**-14 # constante de vitesse en cm3.molécules-1.s-1
1326 M=Na*1.292*(10**-3)/29 # concentration en molécules.cm-3 de l'air
1327 j03=3*10**-5 # en s-1 pour un angle zénithal de 0°
1328 j02=10**-12 # en s-1 hypothese lineaire de la figure 3 doc ENS
1329 jN02=8.2*10**-3 # en s-1 pour un angle zénithal de 0°
1330 o30=Na*10**-12*C03[1896]/48 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
1331 o20=Na*10**-3*0.31/32 #concentration en molécules.cm-3 de dioxygène
dans l'air
1332 o0=0 #hypothèse: pas d'oxygene atomique initialement présent dans
l'air
1333 no0=Na*10**-12*CNO[1896]/30 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
1334 no20=Na*10**-12*CNO2[1896]/46 # concentration en molécules.cm-3 à
l'équinoxe du printemps (20 mars 2021)
1335 V0=np.array([o30,o20,o0,no0,no20]) # condition initiale
1336
1337 # 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
1338 eno2=Na*10**-12*255/(46*24) # emission de NO2 du au trafic routier
en petite couronne en molécule.cm-3.h-1
1339 eno=Na*10**-12*73/(30*24) # emission de NO du au trafic routier en
petite couronne en molécule.cm-3.h-1
1340
1341 #Simulation:
1342
1343 def fJ03(t):
1344     return j03*cos(fteta(t))
1345
1346 def fJ02(t):
1347     return j02*cos(fteta(t))
1348
1349 def fJN02(t):
1350     return jN02*cos(fteta(t))
1351
1352 def F(V,t):
1353     o3,o2,o,no,no2=V
1354     v1=fJN02(t)*no2
1355     v2=no*o3*k
1356     do3=(c*M*o2*o-fJ03(t)*o3-d*o*o3+v1-v2)*3600
1357     do2=(a*M*o**2+2*d*o*o3+fJ03(t)*o3-fJ02(t)*o2-c*M*o2*o-
v1+v2)*3600
1358     do=(2*fJ02(t)*o2+fJ03(t)*o3-c*M*o2*o-2*a*o**2*M-d*o*o3)*3600
1359     dno=(v1-v2)*3600+eno
1360     dno2=(v2-v1)*3600+eno2
1361     return np.array([do3,do2,do,dno,dno2])
1362
1363 # Résolution
1364
1365 T1=np.linspace(0,360,100000)
1366 Y=odeint(F,V0,T1)
1367 Y=np.array(Y)
1368 L03=Y[:,0]
1369 L02=Y[:,1]
1370 L0=Y[:,2]
1371 LNO=Y[:,3]
1372 LNO2=Y[:,4]
1373
1374 # Liste des concentrations réellement mesuré
1375
1376 TT=[k for k in range(101)]
1377 CrO3=[]
1378 CrNO=[]
1379 CrNO2=[]
1380 for k in range(len(TT)):
1381     CrO3.append(10**-12*Na*C03[1896+k]/48)
1382     CrNO.append(10**-12*Na*CNO[1896+k]/30)
1383     CrNO2.append(10**-12*Na*CNO2[1896+k]/46)
1384
1385 # 1er seuil d'alerte pollution ozone: 240 microgramme.m-3 (liste A1)
1386 # 2ème seuil d'alerte pollution ozone: 300 microgramme.m-3 (liste
A2)
1387 # 3ème seuil d'alerte pollution ozone: 360 microgramme.m-3 (liste
A3)
1388
1389 A1=[]
1390 A2=[]
1391 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,L03,label="O3 simulé voiture diesel")
1399 #plt.plot(T1,L02,label="O2 simulation")
1400 #plt.plot(T1,L0,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

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

$$\bullet k = \frac{1}{\tau_{NO} \times [O_30]}$$
$$= 1,8 \times 10^{-14} \text{cm}^3 \cdot \text{molécules}^{-1} \cdot \text{s}^{-1}$$

[O30] : concentration initiale

τ_{NO} : temps de vie de NO

$$\bullet J_{NO_2} = 1/\tau_{NO_2}$$

$$= 8,2 \times 10^{-3} \text{s}^{-1} \text{ (angle zénithal } 0^\circ)$$

τ_{NO_2} : temps de vie NO₂

[M] : Concentration en molécule d'air (80% N₂ 20% O₂)

$$\bullet J_{O_3} = 3 \times 10^{-5} \text{s}^{-1} \text{ (angle zénithal } 0^\circ)$$

$$\bullet J_{O_2} = 10^{-12} \text{s}^{-1} \text{ (angle zénithal } 0^\circ \text{ avec hypothèse linéaire)}$$

ANNEXE :

● Document utilisé :

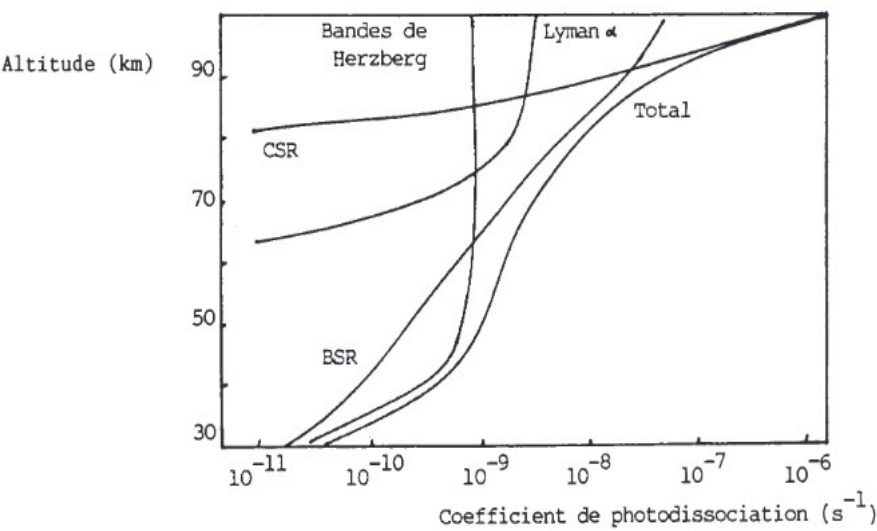


Fig. 3 : Variation avec l'altitude du coefficient de photodissociation de l'oxygène moléculaire. Contribution de différents domaines spectraux.

ACHI

BUP n°805 (2) - cahier enseignement supérieur

	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 : ENS-lyon physique M Ayachi

Source : INSEE

Approximation linéaire pour JO2