Modélisation de la diffusion de pollen

EL FATIHI YOUSSEF
NUM DE DOSSIER :35170
THEME : SANTE ET PREVENTION

PLAN

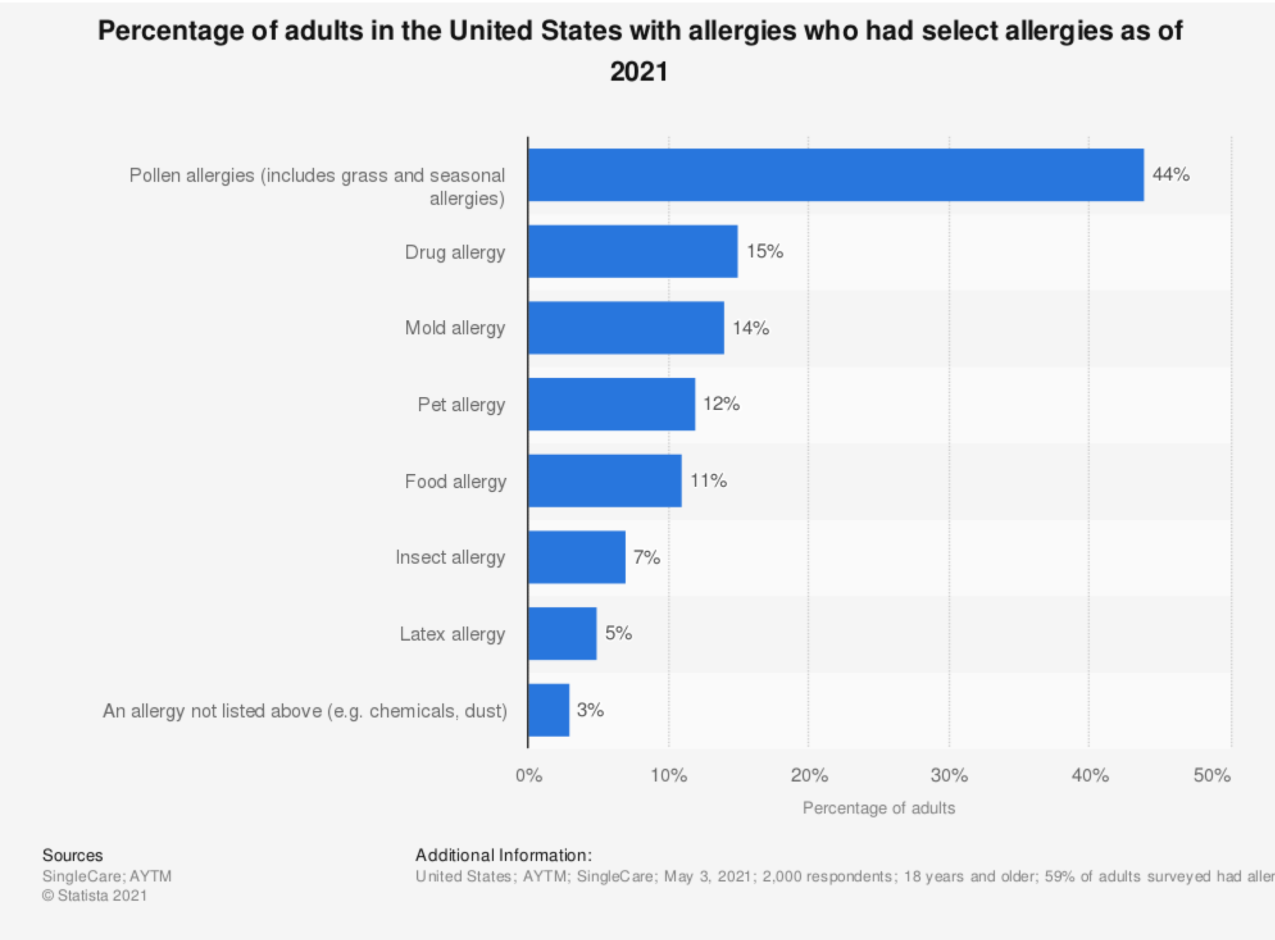


Figure 1: pourcentage des américain ayant une allergie [1]

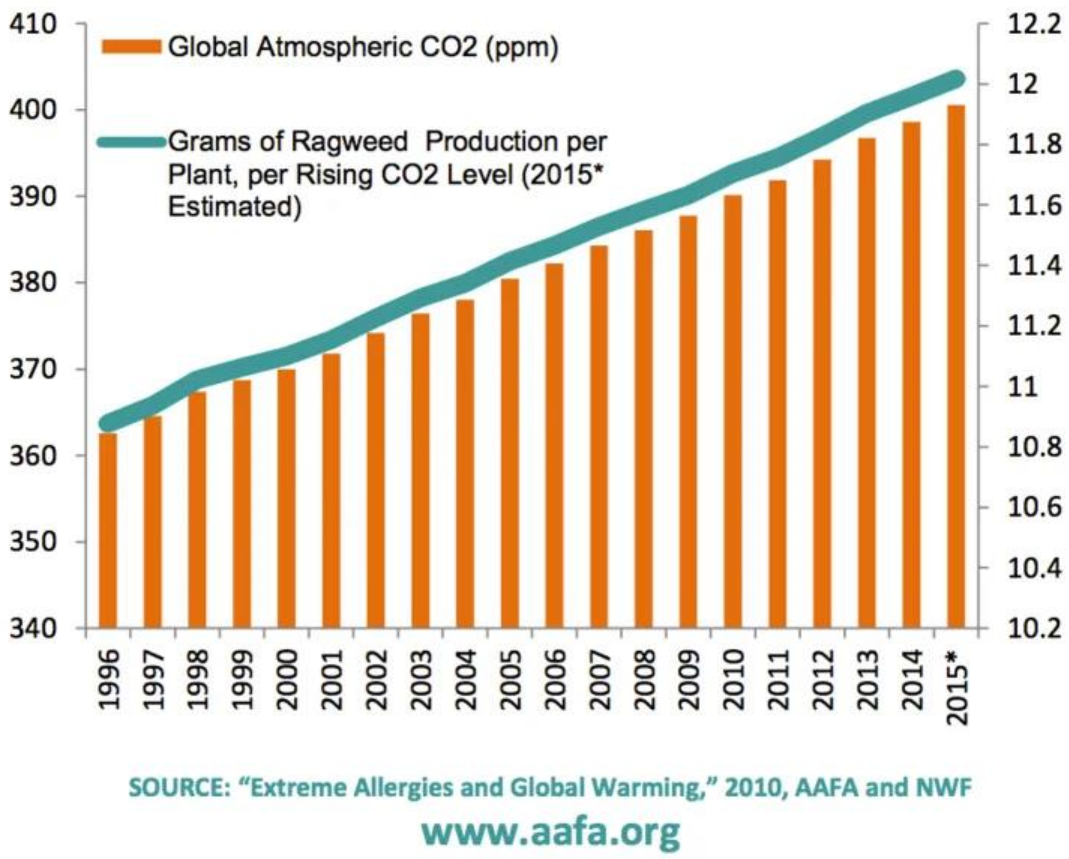


Figure 2: relation entre l'émission du CO_2 et la production du pollen d'ambroisie

Problématique :

modéliser la transport du pollen et tracer la carte de concentration et des zones à risques

Loi de conservation de masse

La modélisation de la diffusion commence par une mise en équation.
On exprime la loi de conservation de masse sous sa forme différentiel :

$$\frac{\partial C}{\partial t} + div(\vec{j}) = P \qquad (1)$$

C: concentration des particules ou nombre de particules par unité de volume en m^{-3}
$\vec{j}$: vecteur densité de courant des particules en $m^{-2}s^{-1}$
P: terme de production en $m^{-3}s^{-1}$

Vecteur de densité de courant

Le vecteur densité résulte d'une superposition d'une diffusion atmosphérique et d'une convection (par le vent et la sédimentation)

$$\vec{j} = \overrightarrow{j_d} + \overrightarrow{j_c}$$

- La diffusion atmosphérique suit la loi de Fick : $\overrightarrow{j_d} = -\boldsymbol{k}\overrightarrow{grad}(C)$

avec $\boldsymbol{k} = diag(k_x, k_y, k_z)$ la matrice contenant les coefficient de diffusivité des particules.

- La contribution convection linéaire s'exprime comme : $\overrightarrow{j_c} = \vec{v}C$ avec $\vec{v}$ la vitesse imposée par la convection .

Vecteur de densité de courant de convection

$$\overrightarrow{j_c} = \begin{cases} C(\vec{u} + \overrightarrow{v_{\text{lim}}}) & \text{Avec sédimentation} \\ C.\vec{u} & \text{Sans sédimentation} \end{cases}$$

$\overrightarrow{v_{lim}}$:vitesse limite de sédimentation en $m.s^{-1}$

$\overrightarrow{v_{lim}} = -\frac{\rho g d^2}{18\mu}\vec{z}$ avec ρla masse volumique de la particule, g l'accélération de pesanteur, d le diamètre de la particule et μ la viscosité dynamique de l'air

$\vec{u}$: vitesse du vent en $m.s^{-1}$

Hypothèse de travail

- Le débit d'émission des particules Q $[s^{-1}]$ est constant
  on modélise le terme de production par $P(x,y,z) = Q\delta(x)\delta(y)\delta(z-H)$
  avec H: la hauteur du point d'émission en m

$$\delta(x) = \begin{cases} 0, & x \neq 0 \\ 1, & x = 0 \end{cases} \quad [m^{-1}]$$

- La vitesse du vent est constante et alignée avec l'axe $\vec{x}$.
- On se place en régime permanent .
- La diffusivité ne dépend que de x .
- La vitesse du vent est suffisamment importante pour négliger la diffusion atmosphérique suivant x donc on peut négliger le terme en k_x

Equation de diffusion

L'équation (1) devient donc :

Sans sédimentation

$$u\frac{\partial C}{\partial x} = k_y\frac{\partial^2 C}{\partial y^2} + k_z\frac{\partial^2 C}{\partial z^2} + Q\delta(x)\delta(y)\delta(z-H) \qquad (2)$$

Avec sédimentation

$$u\frac{\partial C}{\partial x} = k_y\frac{\partial^2 C}{\partial y^2} + k_z\frac{\partial^2 C}{\partial z^2} + Q\delta(x)\delta(y)\delta(z-H) + v_{lim}\frac{\partial C}{\partial z} \qquad (3)$$

Solution de l'équation sans sédimentation

La solution de l'équation (2) est (la démarche de résolution est proposé dans l'annexe):

$$C = \frac{Q}{2\pi u \sigma_Z \sigma_Y} \exp\left(\frac{-y^2}{2{\sigma_y}^2}\right)\left[\exp\left(\frac{-(z-H)^2}{2{\sigma_z}^2}\right) + \exp\left(\frac{-(z+H)^2}{2{\sigma_z}^2}\right)\right]$$

$\sigma_{y,z}$:déviation standard de la distribution normale en m^{-1} définie par :

${\sigma_z}^2(x) = \frac{2}{u}\int_0^x k_z(t)dt = ax^b$ et ${\sigma_y}^2(x) = \frac{2}{u}\int_0^x k_y(t)dt == x465.11628\tan(\theta)$

La détermination des coefficients a et b et θ se passe à travers la méthode de Pasquille-Guifford-Turner (qui est proposé dans l'annexe)

Solution de l’équation sans sédimentation

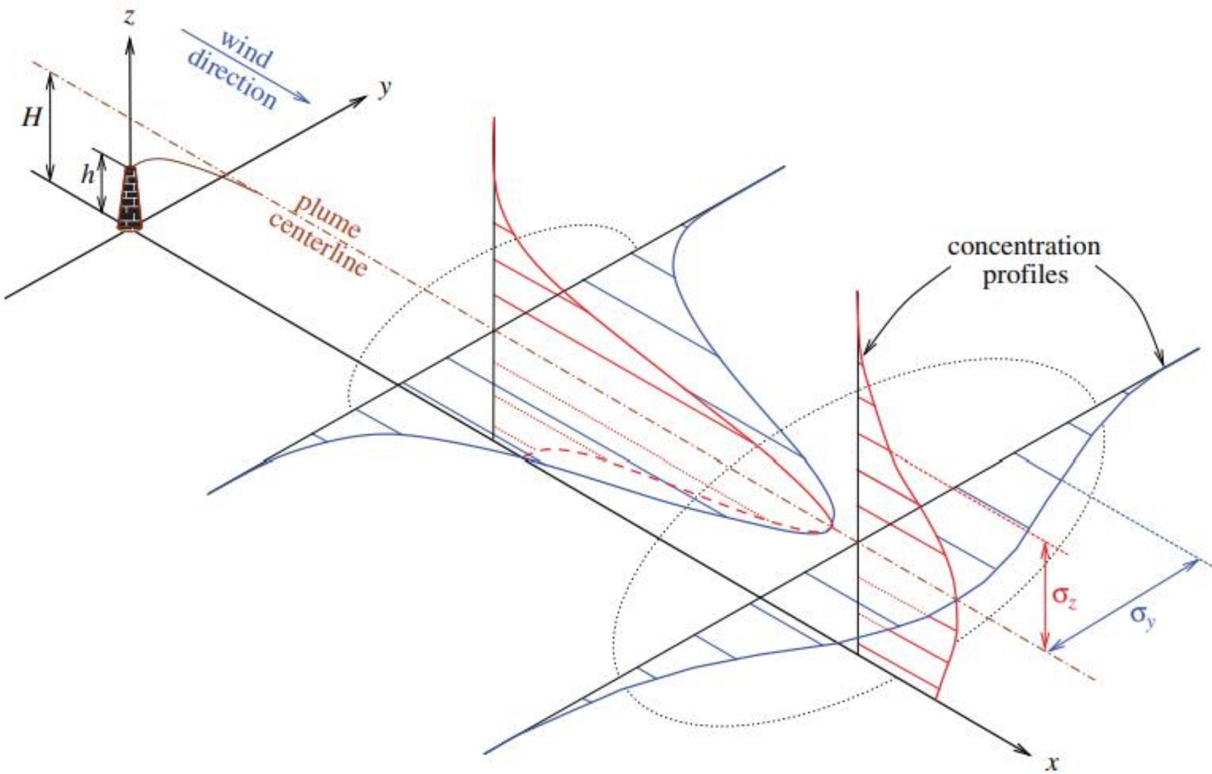


Figure 3: la concentration suit une distribution normale suivant y et z d’écart type σ_y et σ_z respectivement [2]

Solution de l'équation avec sédimentation

Les résultats expérimentales indiquent que la norme du vecteur densité de courant verticale au niveau de la surface est proportionnelle à la concentration au niveau de la surface:[3]

$$(k_z \frac{\partial C}{\partial z} + v_{lim} C)_{\ z=0} = V_d C(z=0)$$

V_d: vitesse de déposition en ms^{-1} avec : $V_d = v_{lim} + \frac{1}{R_{a+} R_{p+} R_a R_p v_{lim}}$ [8]

R_a: résistance aérodynamique en sm^{-1}

R_p: résistance de la sous-couche quasi-laminaire en sm^{-1}

Solution de l'équation avec sédimentation

La solution de l'équation (3) est:

$$C = \frac{Q}{2\pi u \sigma_Z \sigma_Y} \exp\left(\frac{-y^2}{2{\sigma_y}^2}\right) \exp\left(\frac{-v_{lim}(z-h)}{2k_z} - \frac{{v_{lim}}^2{\sigma_z}^2}{8{k_z}^2}\right) \left[\exp\left(\frac{-(z-H)^2}{2{\sigma_Z}^2}\right) + \exp\left(\frac{-(z+H)^2}{2{\sigma_Z}^2}\right) - (2\pi)^{1/2}\frac{V_1\sigma_z}{k_z}\exp\left(\frac{V_1(z+h)^2}{k_z} + \frac{{V_1}^2{\sigma_z}^2}{2k_z}\right)\mathrm{erfc}\left(\frac{V_1\sigma_z}{\sqrt{2}k_z} + \frac{z+h}{\sqrt{2}\sigma_z}\right)\right]$$

$$k_z = \frac{{\sigma_z}^2 u}{2x}$$

avec $V_1 = V - \frac{1}{2}v_{lim}$

erfc(x)=$\frac{2}{\sqrt{\pi}}\int_x^{+\infty}\exp(-t^2)\,dt$ la fonction d'erreur complémentaire

Solution de l’équation avec sédimentation

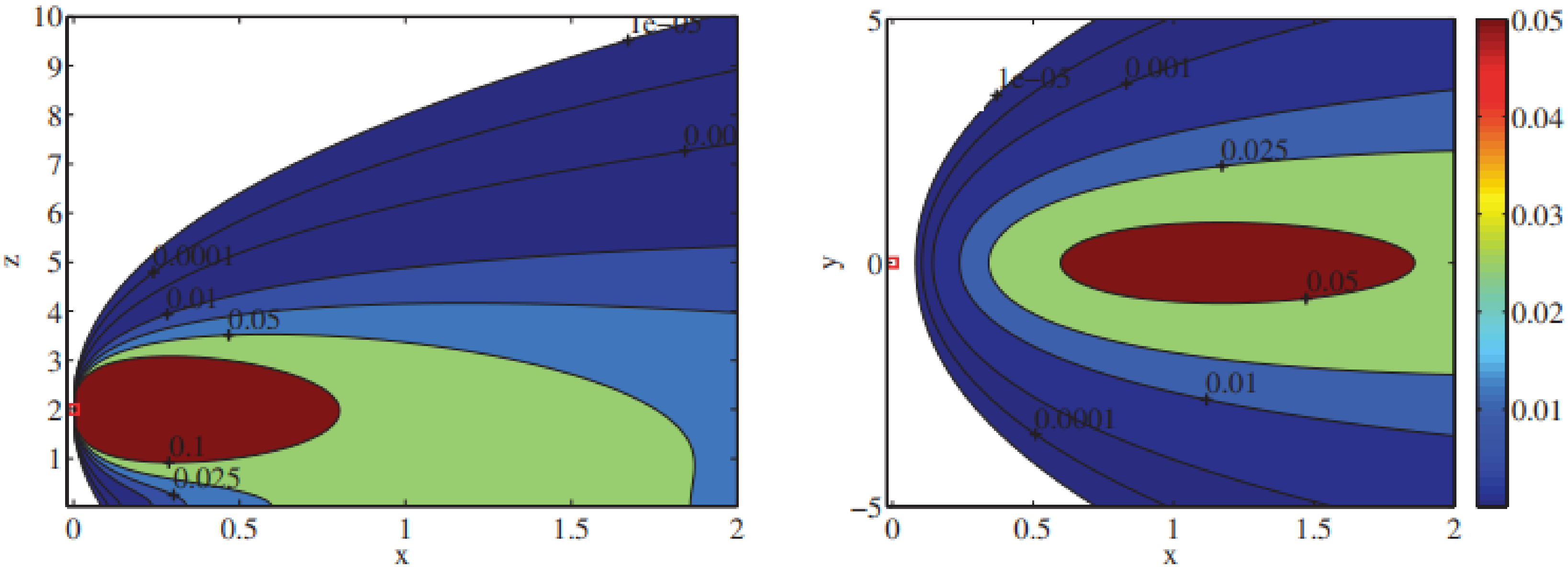


Figure 4: tracé du contour du profil de la concentration avec H=2 Q=1 u=1 k_z=1

Pollen de l'olivier

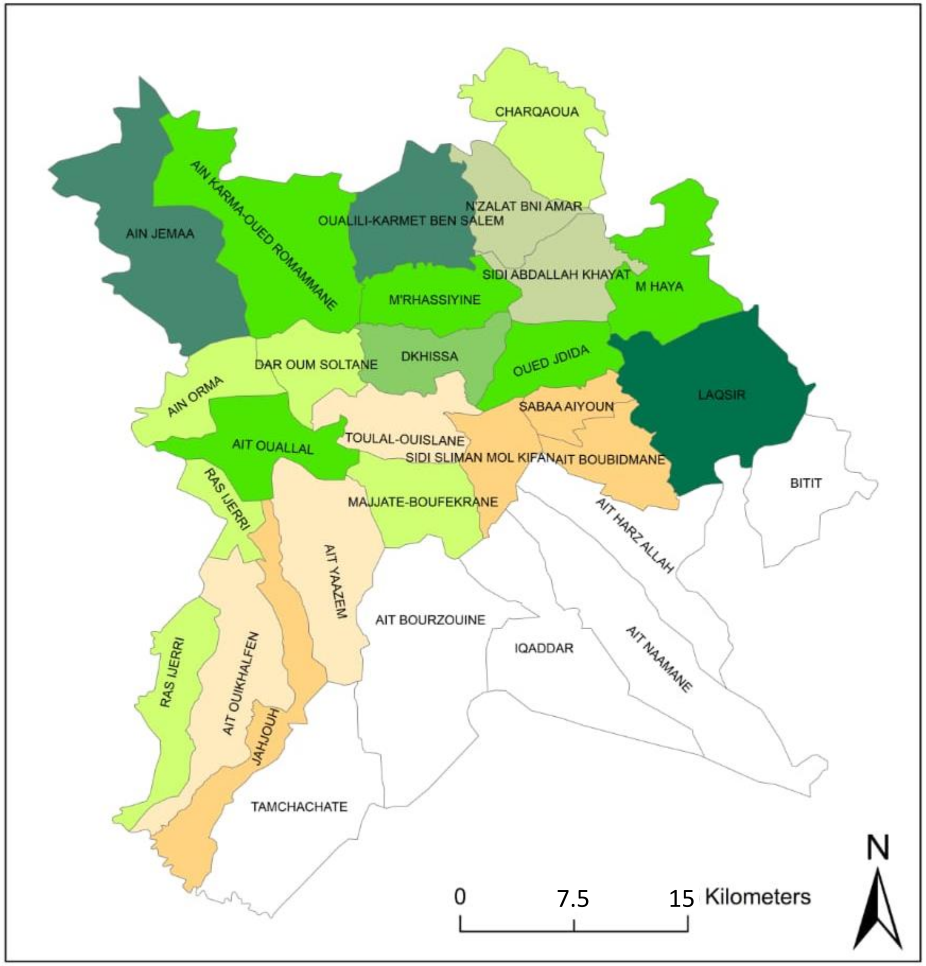
L'olivier est une espèce principalement anémophile(les grains de pollens sont transportés par le vent) ,avec une période de floraison(libération du pollen) qui dure environ 3 semaines entre avril et juin, avec un pendant la deuxième semaine (80% des grains libérés pendant , 10% pendant la première et troisième semaines).Et la libération se fait majoritairement pendant la journée (on va prendre une période entre 7h00 et 19h00). Avec un production totale moyenne de 4×10^{10} grains par an .[4][`7]
Donc on va modélisé le débit d'émission Q par la formule suivante :

$$Q = \frac{production\ totale}{7 \times 12 \times 3600} \times \mathrm{m}$$

Avec m= $\begin{cases} 0{,}8 \\ 0{,}1 \end{cases}$ dépendant de quelle semaine on parle .

les grains de pollen sont modélisés par des sphères de diamètre 21 μm er de densité $1.02 \mu g. cm^{-3}$

Carte des cultures



Légende

Arbre planté par km²(x100)

- 0 - 500
- 500 - 1000
- 1000 - 1500
- 1500 - 2000
- 2000 - 2500
- 2500 - 3000
- 3000 - 3500
- 3500 - 4000

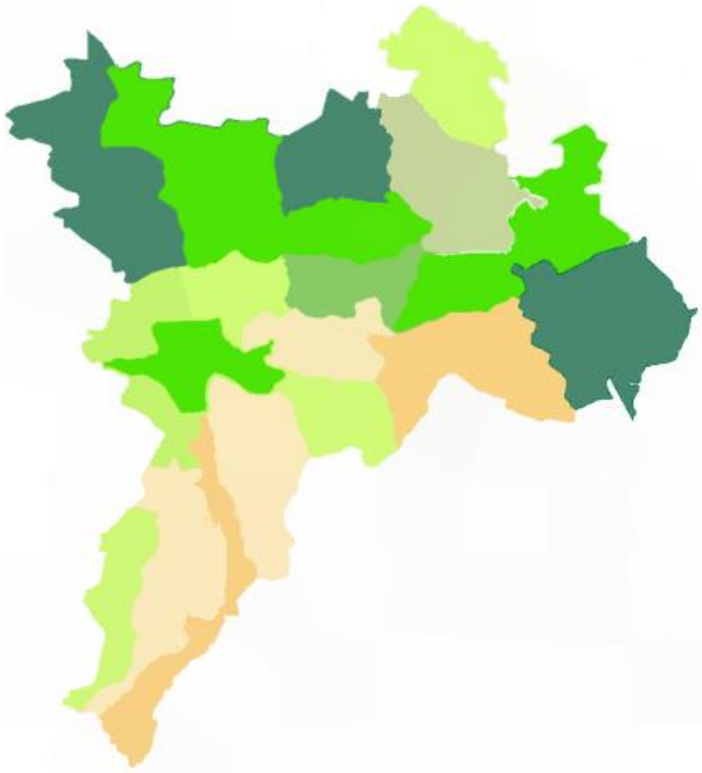
Figure 5: carte des cultures de l’olivier proche de Meknes [6]

Carte des cultures

On doit effectuer une segmentation sur l'image en utilisant python (La segmentation d'image est une opération de traitement d'images qui a pour but de rassembler des pixels entre eux suivant des critères prédéfinis).

Carte des cultures

Figure 6:carte après traitement



L'image après segmentation: Les régions sont clairement séparés et la collecte de données par la reconnaissance des couleurs est faisable.

Débit et vent

Cet année la floraison dans la Région Meknès ,était pendant les 3 premières semaines de mai . La journée qu'on va modélisé est :

- ❑ Le 12 mai : jour de la pleine floraison (pic).
  - Vitesse du vent: u=8.33 m/s[5]
  - Direction du vent:180°
  - Débit d'un seul arbre :

$$Q = \frac{4\times 10^{10}}{7\times 12\times 3600} \times 0.8 = 107367 \text{ grains/seconde}$$

Les sources

Les sources seront modélisés seront modélisés par une sources placés chaque 125 m de débit $Q_s = NQ$ avec N le nombre d'arbre dans un carré de dimension 125m×125m.

Cette disposition introduit une incertitude de l'ordre de 100m sur les distances.

On doit vérifier la sensibilité du modèle pour s'assurer que l'erreur introduit est négligeable .

La simulation suivante est exécuté pour Q=100000grain/s , H=8m/s, z=2m, u= 8m/s

Analyse de la sensibilité

	x	y	C
0	10	10	15.629712
1	10	50	0.833485
2	10	100	0.788158
3	10	250	0.788158
4	10	500	0.788158
5	50	10	1.845159
6	50	50	1.609724
7	50	100	1.216948
8	50	250	0.991426
9	50	500	0.991231
10	100	10	1.166364
11	100	50	1.151350
12	100	100	1.112567
13	100	250	1.013434
14	100	500	0.998869
15	250	10	1.010728
16	250	50	1.010530
17	250	100	1.009934
18	250	250	1.006606
19	250	500	1.001527
20	500	10	1.001351
21	500	50	1.001343
22	500	100	1.001320
23	500	250	1.001170
24	500	500	1.000759

Figure 7.a : sensibilité du modèle sans sédimentation

	x	y	C
0	10	10	19.387358
1	10	50	2.059500
2	10	100	1.208040
3	10	250	1.013372
4	10	500	1.001682
5	50	10	0.974276
6	50	50	1.766514
7	50	100	1.189355
8	50	250	1.013125
9	50	500	1.001672
10	100	10	0.917869
11	100	50	1.277726
12	100	100	1.141092
13	100	250	1.012384
14	100	500	1.001644
15	250	10	0.917869
16	250	50	0.997077
17	250	100	1.017727
18	250	250	1.008242
19	250	500	1.001457
20	500	10	0.917869
21	500	50	0.996834
22	500	100	0.999601
23	500	250	1.001921
24	500	500	1.000946

Figure 7.b: Sensibilité du modèle avec sédimentation

Analyse de la sensibilité

L'analyse de la sensibilité montre que la différence de concentration en changeant la distance de la source d'une centaine de mètres est négligeable spécialement au-delà de 10m de la source.
L'approximation est donc envisageable.

Résultat de la simulation sans sédimentation

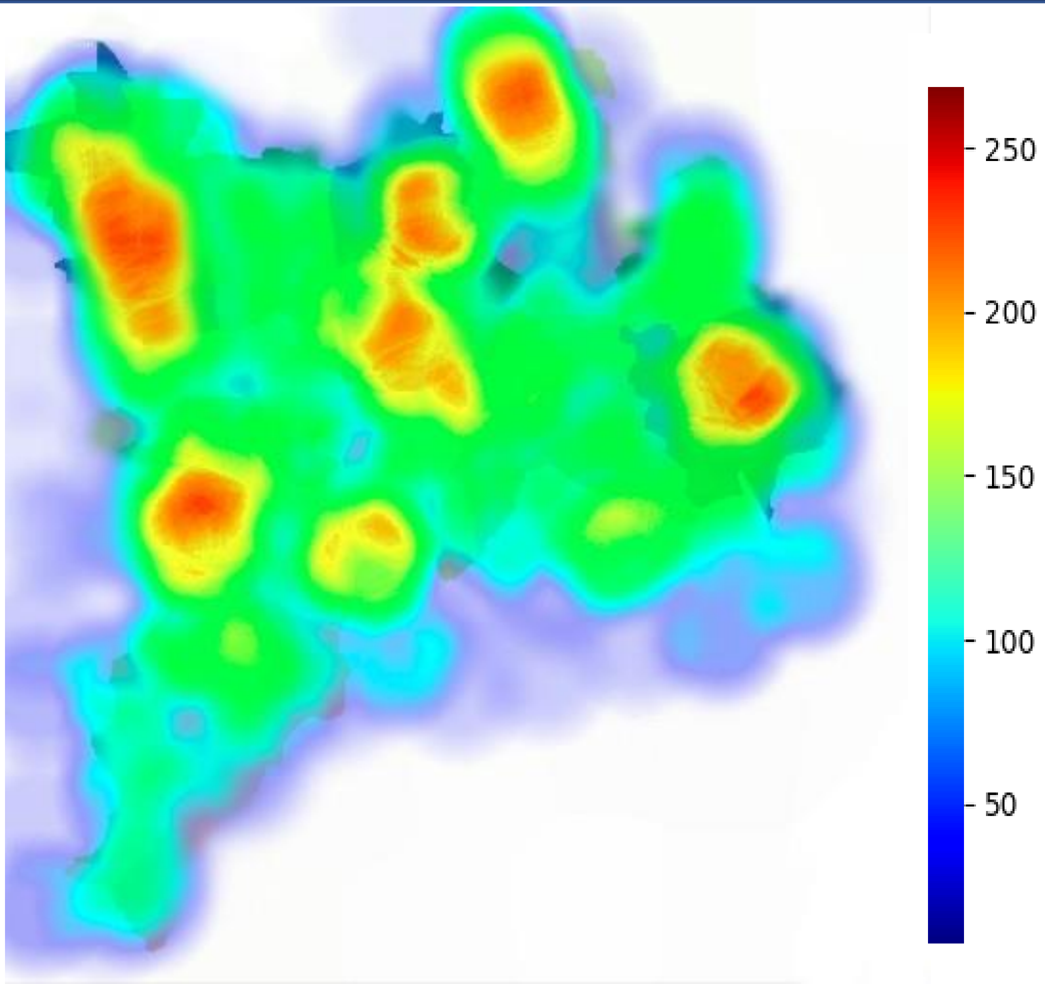


Figure 8.a: carte de concentration

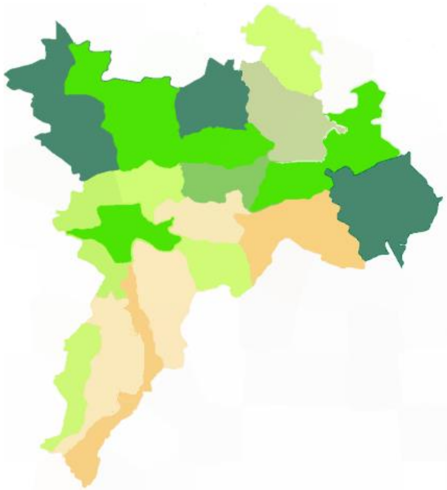


Figure 8.b:carte originale (Template)

Résultat de la simulation avec sédimentation

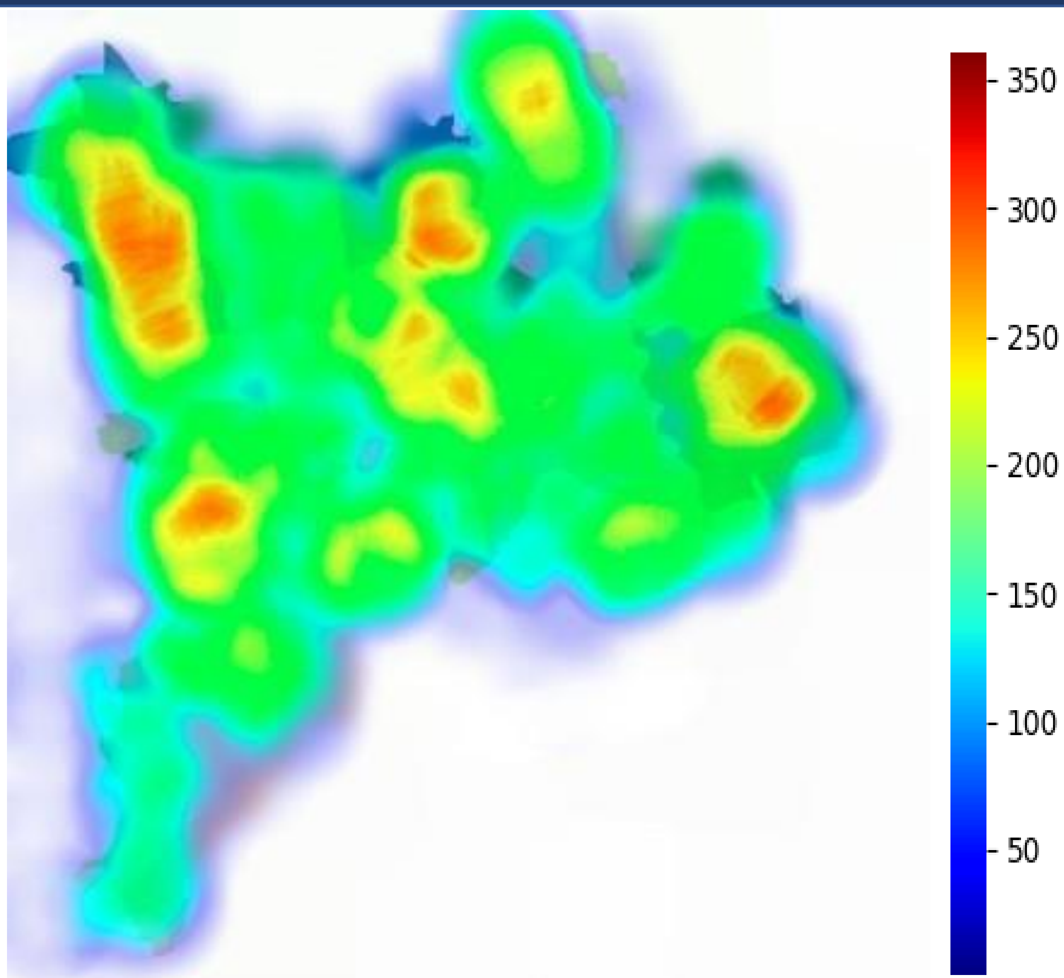


Figure 8.a: carte de concentration

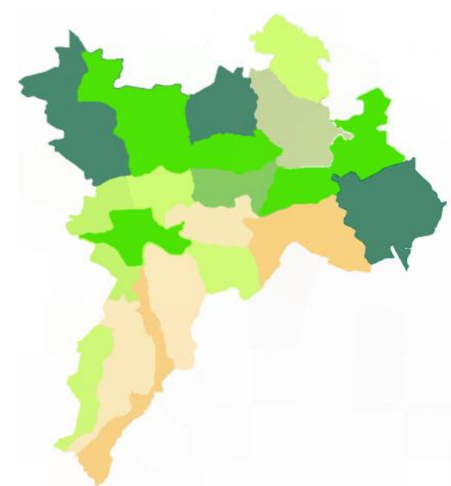
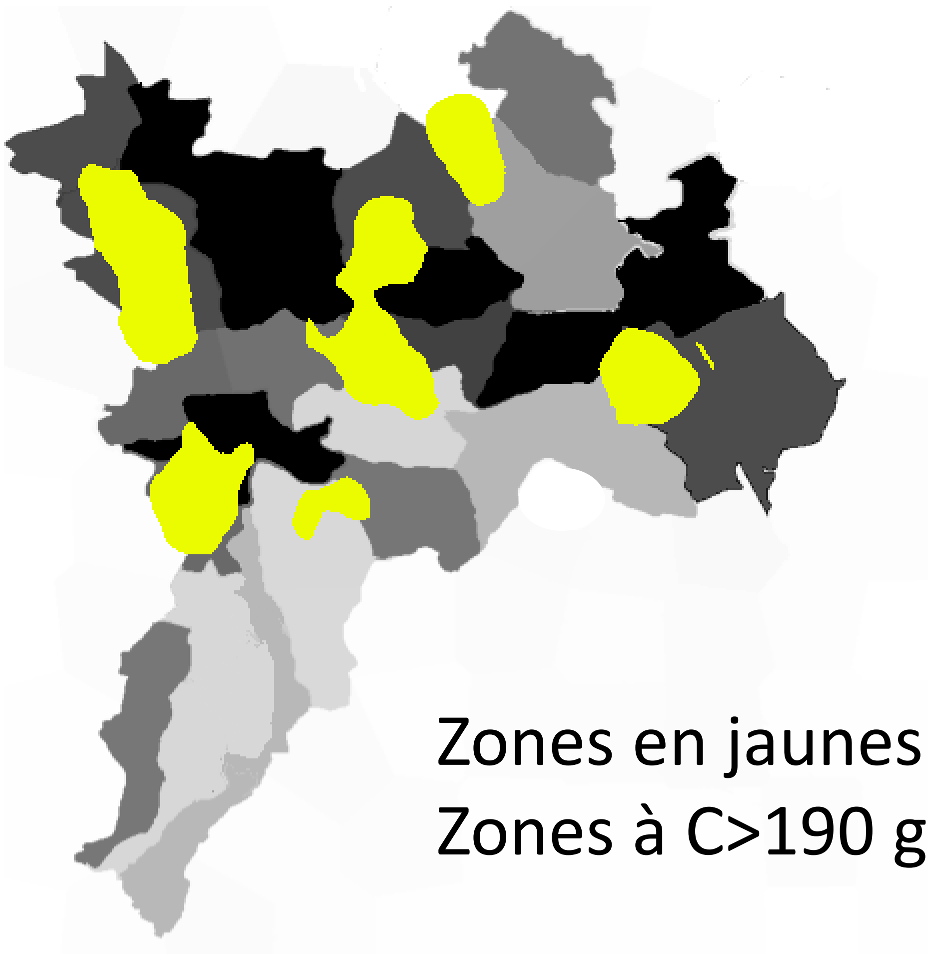


Figure 8.b:carte originale (Template)

Zones à risques



Zones en jaunes = zones à risques=
Zones à C>190 grains par mètres carrée

Conclusion

✓ Résultat: la carte avec les zones à risques

X Limitations :

- le modèle n’est pas valables dans des grandes distance (fluctuation des vitesse et direction de vent)
- l’algorithme informatique est très complexe

Merci pour votre attention

ANNEXE

Référence additionnelle

[1]:https://www.statista.com/ (consulté le 25 mai)
[2]:T1 - Simulation of I-131 Dispersion around KNS (Kawasan Nuklir Serpong) using Gaussian Plume Model DO - 10.1109/QIR.2017.8168521
[3]:Chamberlain, A. C., 1953: Aspects of travel and deposition of aerosol and vapor clouds. Atomic Energy Research Establishment, HP/R 1261
[4]:memoir de fin d'étude , Aadil Bajoub,2006 , prevision du rendement des récoltes de l'Olivier à partir de l'étude du contenu pollinique, l'Ecole national d'agriculture de Meknes
[5]:La direction provincial d'agriculture de Meknès
[6]:École national d'agriculture de Meknès
[7]:Rafael Tormo Molina , Adolfo Muñoz Rodríguez , Inmaculada Silva Palaciso & Francisco Gallardo López (1996) Pollen production in anemophilous trees, Grana, 35:1, 38-46, DOI: 10.1080/00173139609430499
[8]:modele de diffusion AERMOD developpé par " the American Meteorological Society (AMS)/United States Environmental Protection Agency (EPA) Regulatory Model Improvement Committee "
[9]: l'université de Washington

Démarche de résolution de l’équation (2)

(1) : $u \frac{\partial C}{\partial x} = K_y \frac{\partial^2 C}{\partial y^2} + K_z + Q\,\delta(x)\,\delta(y)\,\delta(z-H)$

D'après le théorème de Stackgold (Démonstration ci-dessous)

(1) est équivalente à $\Leftrightarrow$ $u \frac{\partial C}{\partial x} = K_y \frac{\partial^2 C}{\partial y^2} + \frac{\partial^2 C}{\partial z^2}$ (2).

avec $C(0,y,z) = \frac{Q}{u}\,\delta(y)\,\delta(z-H)$

- On cherche une solution sous la forme $C(x,y,z) = \frac{Q}{u} f(x,y)\, g(x,z)$

On obtient alors $u \frac{\partial f}{\partial x} = K_y \frac{\partial^2 f}{\partial y^2}$ (3.a) $u \frac{\partial g}{\partial x} = K_z \frac{\partial^2 g}{\partial z^2}$ (3.b)

- $x \in [0, +\infty[$
- $y \in]-\infty, +\infty[$

Démarche de résolution de l’équation (2)

- $y \in]-\infty, +\infty[$
- $z \in]0, +\infty[$

- On introduit les 2 changements de variables $\sigma^2_{y,z}(x) = \frac{2}{u}\int_0^x K(t)\,dt$

(3,a), (3,b) deviennent :

(4,a) $\frac{1}{2}\frac{\partial g}{\partial r} = \frac{\partial^2 g}{\partial y^2}$ (4,b) $\frac{1}{2}\frac{\partial g}{\partial r} = \frac{\partial^2 g}{\partial z^2}$

- On passe donc par la transformation de Laplace, et d'après les conditions aux limites :
  - $c(0,y,z) = \frac{Q}{u}\delta(y)\delta(z-H)$
  - $c(\infty,y,z) = c(x,\infty,z) = c(x,y,\infty) = 0$
  - $K_z \frac{\partial c}{\partial z}(x,y,0) = 0$

On retrouve le résultat.

- La démarche est similaire pour l'équation avec sédimentation.

Théorème de Stackgold

DEFINITION. *The causal fundamental solution $C(x, t)$ is the particular solution of (5.134) which vanishes identically for $t < 0$. Thus $C(x, t)$ satisfies*

$$\frac{\partial C}{\partial t} - a\nabla^2 C = \delta(x)\delta(t), \qquad C \equiv 0 \text{ for } t < 0. \tag{5.135}$$

The causal fundamental solution $C(x, t)$ has a direct physical interpretation; it is the temperature distribution in a medium which is at zero temperature up to time $t = 0$, when a concentrated source is introduced at $x = 0$, this source instantaneously releasing a unit of heat. Although C is defined for all t and x, its calculation presents a problem only for $t > 0$ ($C = 0$ for $t < 0$). This immediately suggests a slightly different point of view; for $t > 0$ no sources are present, so that C satisfies the homogeneous equation and must reduce, at $t = 0+$, to a certain initial temperature. This initial temperature is the one to which the medium has been raised just after the introduction of an instantaneous concentrated source of unit strength. We now show that this initial temperature is $\delta(x)$.

Theorem. *The causal fundamental solution coincides for $t > 0$ with the solution of the initial value problem*

Théorème de Stackgold

$$\frac{\partial u}{\partial t} - a\nabla^2 u = 0, \quad t > 0; \qquad u(x, 0+) = \delta(x). \tag{5.136}$$

Proof. Let $u(x, t)$ be the solution for $t > 0$ of system (5.136) and let $C(x, t)$ be defined by

$$C(x, t) = \begin{cases} u(x, t), & t > 0; \\ 0, & t < 0. \end{cases} \tag{5.137}$$

We must show that $C(x, t)$ satisfies (5.135). The requirement $C \equiv 0$ for $t < 0$ is obviously satisfied. We can write

$$C(x, t) = H(t)u(x, t),$$

where $H(t)$ is the Heaviside function which is 1 for $t > 0$ and vanishes for $t < 0$. Then, proceeding formally,

$$\nabla^2 C = H(t)\nabla^2 u,$$

$$\frac{\partial C}{\partial t} = H(t)\frac{\partial u}{\partial t} + u(x, t)\frac{dH}{dt}(t) = H(t)\frac{\partial u}{\partial t} + u(x, 0+)\delta(t).$$

Thus

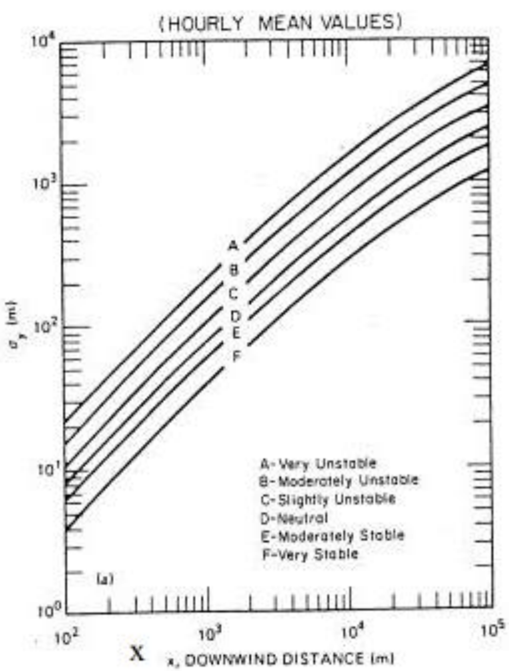
$$\frac{\partial C}{\partial t} - a\nabla^2 C = u(x, 0+)\delta(t) = \delta(x)\delta(t),$$

which completes the proof.

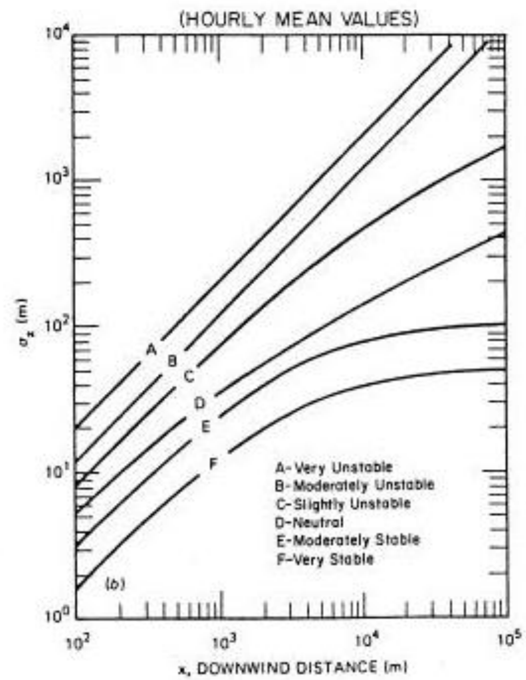
La méthode empirique de Pasquille-Guifford-Turner

Les valeurs des écart-type sont basées sur la classe de stabilité de l'atmosphère . Plusieurs méthodes sont développées pour calculer ces valeurs, mais la plus adopté est celle de Pasquille-Guifford-Turner . Qui détermine les coefficients à partie de ces courbes .[10]

Sigma-y



Sigma-z



La méthode empirique de Pasquille-Guifford-Turner

Ces courbes sont approximés par les expressions suivantes :

Plume sigma formulas from EPA's ISC Model

Vertical distribution: $$\sigma_z = ax^b$$

x is in *kilometers*
σ_z is in *meters*
a, b depend on x

Cross-wind distribution: $$\sigma_y = 465.11628x(\tan\Theta)$$

$$\Theta = 0.017453293(c - d\ln(x))$$

x is in *kilometers*
σ_y is in *meters*
Θ is in *radians*

La méthode empirique de Pasquille-Guifford-Turner

Les coefficients a,b,c,d sont déterminées à partir des tableau suivant :

$$\sigma_z = ax^b$$

Pasquill Stability Category	x (km)	a	b
A*	<.10	122.800	0.94470
	0.10 - 0.15	158.080	1.05420
	0.16 - 0.20	170.220	1.09320
	0.21 - 0.25	179.520	1.12620
	0.26 - 0.30	217.410	1.26440
	0.31 - 0.40	258.890	1.40940
	0.41 - 0.50	346.750	1.72830
	0.51 - 3.11	453.850	2.11660
	>3.11	**	**

* If the calculated value of σ_z exceed 5000 m, σ_z is set to 5000 m.

$$\sigma_z = ax^b$$

Pasquill Stability Category	x (km)	a	b
B*	<.20	90.673	0.93198
	0.21 - 0.40	98.483	0.98332
	>0.40	109.300	1.09710
C*	All	61.141	0.91465
D	<.30	34.459	0.86974
	0.31 - 1.00	32.093	0.81066
	1.01 - 3.00	32.093	0.64403
	3.01 - 10.00	33.504	0.60486
	10.01 - 30.00	36.650	0.56589
	>30.00	44.053	0.51179

* If the calculated value of σ_z exceed 5000 m, σ_z is set to 5000 m.

** σ_z is equal to 5000 m.

La méthode empirique de Pasquille-Guifford-Turner

$$\sigma_z = ax^b$$

Pasquill Stability Category	x (km)	a	b
E	<.10	24.260	0.83660
	0.10 - 0.30	23.331	0.81956
	0.31 - 1.00	21.628	0.75660
	1.01 - 2.00	21.628	0.63077
	2.01 - 4.00	22.534	0.57154
	4.01 - 10.00	24.703	0.50527
	10.01 - 20.00	26.970	0.46713
	20.01 - 40.00	35.420	0.37615
	>40.00	47.618	0.29592
F	<.20	15.209	0.81558
	0.21 - 0.70	14.457	0.78407
	0.71 - 1.00	13.953	0.68465
	1.01 - 2.00	13.953	0.63227
	2.01 - 3.00	14.823	0.54503
	3.01 - 7.00	16.187	0.46490
	7.01 - 15.00	17.836	0.41507
	15.01 - 30.00	22.651	0.32681
	30.01 - 60.00	27.074	0.27436
	>60.00	34.219	0.21716

$$\Theta = 0.017453293\left(c - d\ln(x)\right)$$

Pasquill Stability Category	c	d
A	24.1670	2.5334
B	18.3330	1.8096
C	12.5000	1.0857
D	8.3330	0.72382
E	6.2500	0.54287
F	4.1667	0.36191

La méthode empirique de Pasquille-Guifford-Turner

La détermination des classes de stabilité ce passe à travers les tableaux suivants

Sky Cover	Solar elevation angle>60 degrees	Solar elevation angle >35 degrees ≤60 degrees	Solar elevation angle > 15 degrees ≤35 degrees
High thin clouds	strong	moderate	slight
middle clouds *	moderate	slight	slight
low clouds **	slight	slight	slight

* middle clouds – base at 7,000 to 15,000 feet
** low clouds – base less than 7,000 feet

Surface wind speed (m/s)	Daytime insolation			Nighttime	
	strong	moderate	slight	Thin overcast or > 4/8 low cloud cover	< 3/8 cloud cover
< 2	A	A-B	B	-	-
2-3	A-B	B	C	E	F
3-5	B	B-C	C	D	E
5-6	C	C-D	D	D	D
> 6	C	D	D	D	D

Codes python : le modèle principale

```
def concentration(Q,x,y,z,ul,H,classe):
    (dev_y,dev_z)=deviation(classe,x)

    C=Q/(2.*np.pi*ul*dev_y*dev_z) * np.exp(-y**2./(2.*dev_y**2.)) *(np.exp(-(z
   -H)**2./(2.*dev_z**2.))) +np.exp(-(z+H)**2./(2.*dev_z**2.))
    return(C)


#####
def concentration_sedimentation(Q,x,y,z,ul,H,classe):
    (dev_y,dev_z)=deviation(classe,x)
    k=(dev_z*ul)/2*x

    C=Q/(2.*np.pi*ul*dev_y*dev_z) * np.exp(-y**2./(2.*dev_y**2.)) *((np.exp(-(z
   -H)**2./(2.*dev_z**2.))) +np.exp(-(z+H)**2./(2.*dev_z**2.))-np.sqrt(2*np.pi)*(V*dev_z/
k)*np.exp((V*(z+H)**2/k)+((V*dev_z)**2)/2*k)*erfc((((V*dev_z)/np.sqrt(2)*k)+(z+H)/
np.sqrt(2)*dev_z))*np.exp((-vl*(z-h)/k*2)-(((vl*dev_z)**2)/8*(k**2)))
    return(C)

    ####
```

Codes python : le modèle principale

```
def deviation(CLASSE,x1):

    x=np.mgrid[0:10000:5]

    a=np.zeros(np.shape(x));
    b=np.zeros(np.shape(x));
    c=np.zeros(np.shape(x));
    d=np.zeros(np.shape(x));

    if CLASSE == "A":

        temp=np.where((x<100.) & (x>0.));
        a[temp]=122.800
        b[temp]=0.94470

        temp=np.where((x>=100.) & (x<150.));
        a[temp]=158.080
        b[temp]=1.05420

        temp=np.where((x>=150.) & (x<200.));
        a[temp]=170.220
        b[temp]=1.09320

        temp=np.where((x>=200.) & (x<250.));
        a[temp]=179.520
        b[temp]=1.12620

        temp=np.where((x>=250.) & (x<300.));
        a[temp]=217.410
        b[temp]=1.26440

        temp=np.where((x>=300.) & (x<400.));
        a[temp]=258.89
        b[temp]=1.40940

        temp=np.where((x>=400.) & (x<500.));
        a[temp]=346.75
        b[temp]=1.7283

        temp=np.where((x>=500.) & (x<3110.));
        a[temp]=453.85
        b[temp]=2.1166
```

```
        temp=np.where((x>=3110.));
        a[temp]=453.85
        b[temp]=2.1166


        c[:]=24.1670;
        d[:]=2.5334;
    elif CLASSE == "B":

        temp=np.where((x<200.) & (x>0.));
        a[temp]=90.673
        b[temp]=0.93198

        temp=np.where((x>=200.) & (x<400.));
        a[temp]=98.483
        b[temp]=0.98332

        temp=np.where(x>=400.);
        a[temp]=109.3
        b[temp]=1.09710


        c[:]=18.3330;
        d[:]=1.8096;

    elif CLASSE == "C":

        a[:]=61.141;
        b[:]=0.91465;

        c[:]=12.5;
        d[:]=1.0857;
    elif CLASSE == "D":

        temp=np.where((x<300.) &(x>0.));
        a[temp]=34.459;b[temp]=0.86974;

        temp=np.where((x>=300.) & (x<1000.));
        a[temp]=32.093;b[temp]=0.81066;

        temp=np.where((x>=1000.) & (x<3000.));
        a[temp]=32.093;b[temp]=0.64403;
```

Codes python : le modèle principale

```
    c[:]=12.5;
    d[:]=1.0857;
elif CLASSE == "D":

    temp=np.where((x<300.) &(x>0.));
    a[temp]=34.459;b[temp]=0.86974;

    temp=np.where((x>=300.) & (x<1000.));
    a[temp]=32.093;b[temp]=0.81066;

    temp=np.where((x>=1000.) & (x<3000.));
    a[temp]=32.093;b[temp]=0.64403;

    temp=np.where((x>=3000.) & (x<10000.));
    a[temp]=33.504;b[temp]=0.60486;

    temp=np.where((x>=10000.) & (x<30000.));
    a[temp]=36.650;b[temp]=0.56589;

    temp=np.where(x>=30000.);
    a[temp]=44.053;b[temp]=0.51179;


    c[:]=8.3330;
    d[:]=0.72382;
elif CLASSE == "E":

    temp=np.where((x<100.) & (x>0.));
    a[temp]=24.26;b[temp]=0.83660;

    temp=np.where((x>=100.) & (x<300.));
    a[temp]=23.331;b[temp]=0.81956;

    temp=np.where((x>=300.) & (x<1000.));
    a[temp]=21.628;b[temp]=0.75660;

    temp=np.where((x>=1000.) & (x<2000.));
    a[temp]=21.628;b[temp]=0.63077;

    temp=np.where((x>=2000.) & (x<4000.));
    a[temp]=22.534;b[temp]=0.57154;
```

```
elif CLASSE == "F":

    temp=np.where((x<200.) & (x>0.));
    a[temp]=15.209;b[temp]=0.81558;

    temp=np.where((x>=200.) & (x<700.));
    a[temp]=14.457;b[temp]=0.78407;

    temp=np.where((x>=700.) & (x<1000.));
    a[temp]=13.953;b[temp]=0.68465;

    temp=np.where((x>=1000.) & (x<2000.));
    a[temp]=13.953;b[temp]=0.63227;

    temp=np.where((x>=2000.) & (x<3000.));
    a[temp]=14.823;b[temp]=0.54503;

    temp=np.where((x>=3000.) & (x<7000.));
    a[temp]=16.187;b[temp]=0.46490;

    temp=np.where((x>=7000.) & (x<15000.));
    a[temp]=17.836;b[temp]=0.41507;

    temp=np.where((x>=15000.) & (x<30000.));
    a[temp]=22.651;b[temp]=0.32681;

    temp=np.where((x>=30000.) & (x<60000.));
    a[temp]=27.074;b[temp]=0.27436;

    temp=np.where(x>=60000.);
    a[temp]=34.219;b[temp]=0.21716;


    c[:]=4.1667;
    d[:]=0.36191;
sig_z=a*(x/1000.)**b;
sig_z[np.where(sig_z[:]>5000.)]=5000.;

theta=0.017453293*(c-d*np.log(np.abs(x+1e-15)/1000.));
sig_y=465.11628*x/1000.*np.tan(theta);

(dev_y,dev_z)=(sig_y[x1],sig_z[x1])

return  (dev_y,dev_z)
```

Codes python : le modèle principale

```
def model1(Q,xs,ys):
    g=[[[0]*10]*10000 for i in range (10000)]



    for i in trange (10000):
        for j in range(10000):
            for k in range(10):
                y=np.abs(ys-j*5)
                x=np.abs(xs-i*5)
                z=k*2
                g[i][j][k]=concentration(Q,u,x,y,z,H,classe)


                ####
def model2(Q,xs,ys):
    g=[[[0]*10]*10000 for i in range (10000)]



    for i in trange (10000):
        for j in range(10000):
            for k in range(10):
                y=np.abs(ys-j*5)
                x=np.abs(xs-i*5)
                z=k*2
                g[i][j][k]=concentration_sedimentation (Q,u,x,y,z,H,classe)
```

Codes python : le modèle principale

```
C=np.zeros((10000,10000))
A=C
for i in range(len(g)):
    for j in range(len(g[i])):
        A=np.asarray(model1(Q[i][j],[i],[j]))
        C=C+A
```

```
####
```

```
C=np.zeros((10000,10000))
A=C
for i in range(len(g)):
    for j in range(len(g[i])):
        A=np.asarray(model2(Q[i][j],[i],[j]))
        C=C+A
```

Codes python : la sensibilité

```
import pandas as pd
(Q,u,z,H)=(100000,8,2,3)

def my_model(x,y):

    return concentration(Q,x,y,z,u,H,"A")
X=[10,50,100,250,500]
Y=[10,50,100,250,500]
outputs = []
for x1 in X:
    for x2 in Y:
        y_i = my_model(x1, x2)
        outputs.append((x1, x2, y_i))
outputs

df = pd.DataFrame(outputs, columns=['x', 'y', 'C'])
df


####
import pandas as pd
(Q,u,z,H)=(100000,8,2,3)

def my_model(x,y):

    return concentration_sedimentation(Q,x,y,z,u,H,"A")
X=[10,50,100,250,500]
Y=[10,50,100,250,500]
outputs = []
for x1 in X:
    for x2 in Y:
        y_i = my_model(x1, x2)
        outputs.append((x1, x2, y_i))
outputs

df = pd.DataFrame(outputs, columns=['x', 'y', 'C'])
df
```