SOMMAIRE

- **Présentation générale -** Modèle choisi, objectifs à réaliser …
- **(1) Modélisation** - Dynamique de la population d'une ruche au cours de l'année
- **(2) Modélisation** de la **propagation** des abeilles et des pesticides
- **Mise en relation** des différentes modélisations
- **Conclusion**

AU SEIN D'UNE RUCHE DOMESTIQUE

APIS MELLIFERA

Jusqu'à **2000** œufs pondues par la reine par jour

85% butineuses (récoltent le miel)

5% bourdons mâles (reproduction)

10% pour le reste

45 jours de temps de vie

Sources
Thèses de TOMA et al (2009) – MALLICK A. (2013)

ORGANISATION AU SEIN D'UNE RUCHE

PRINTEMPS

- Augmentation de la population
- Ponte de la reine suivant une fonction explicitée ci-après

HIVER

- Hibernation, le temps de vie des abeilles est rallongée
- Une partie de la population a disparu (bourdons mâles)

Seuil minimal d'abeilles en entrée à déterminer

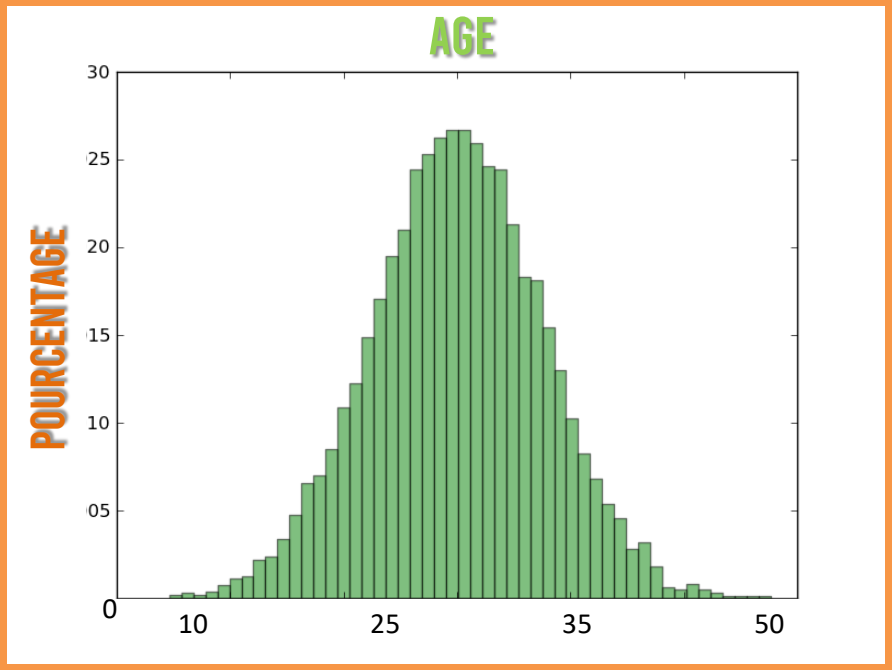
ETE - AUTOMNE

- Préparation pour l'hibernation, temps de vie commence à augmenter
- Les bourdons mâles commencent à disparaître
- La reine arrête la ponte.

MODELISATION 1.
PREQUIS – REPARTITION DE LA POPULATION

```
f = lambda x:(1/np.sqrt(2*np.pi))*np.exp((-x**2)/2)
```

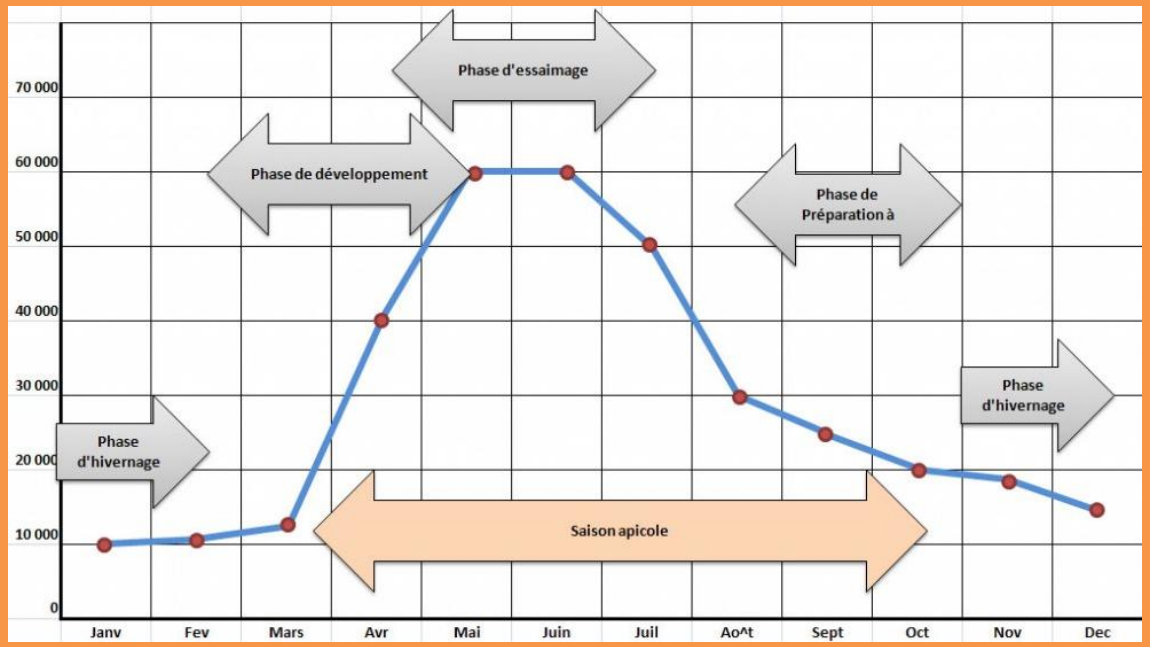
```
def fpop_init(pop):
    h= 10/jvie ; a=-5
    pop_tot=[]; ouv=[]; mal=[]; res=[]
    for i in range (jvie):
        pop_tot.append(int((f(a+i*h)*h)*pop))
        ouv.append(int((f(a+i*h)*h)*pop*0.85))
        mal.append(int((f(a+i*h)*h)*pop*0.05))
        res.append(int((f(a+i*h)*h)*pop*0.10))
    return (pop_tot,ouv,mal,res)
```



Graphique indiquant **le pourcentage d'abeilles** présentes dans la ruche **en fonction de leur âge** au jour 0

MODELISATION 1.
PREQUIS – PONTE DE LA REINE

```
g = lambda x:((-80/81)*x**2+(800/9)*x)/1.7
```



issue de la thèse d'Alice Mallick, 2013

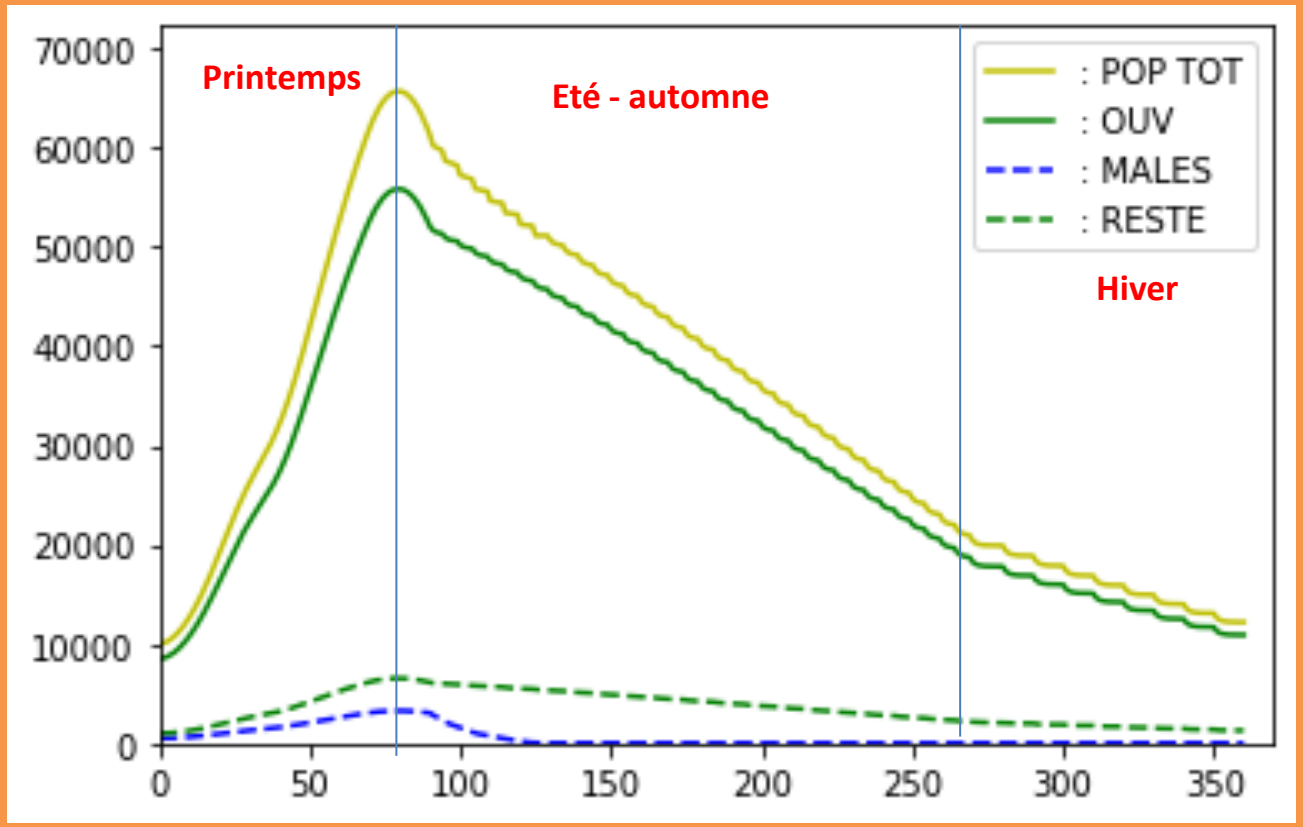
MODELISATION 1.
PARTIE PRINTEMPS

```
def ruche (jour,duree):
    #Définition de toutes les constantes
    compteur=no_var_compteur[0]
    nb_pol = tot_pop
    no_var_jour = jour
    no_var_duree = duree
    saison = fsaison(jour)
    cycle = no_var_cycle[0]
    while cycle > -1:
        if saison == 'printemps' and (no_var_jour+no_var_duree) >90+360*compteur:
            print('PRINTEMPS ++')
            if sum(tot_pop) != 0:
                for i in range (no_var_jour-(360*compteur),91):
                    nb_pol[0] = nb_pol[0] + int(g(i))
                    ouv[0] = ouv[0] +  int(g(i)*0.85)
                    mal[0] = mal[0] + int(g(i)*0.05)
                    res[0] = res[0] + int(g(i)*0.10)
                    list_graph_ouv.append(sum(ouv))
                    list_graph_tot_pop.append(sum(tot_pop))
                    list_graph_res.append(sum(res))
                    list_graph_mal.append(sum(mal))
                    """



                    nb_pol[len(nb_pol)-1]=0
                    ouv[len(ouv)-1]=0
                    res[len(res)-1]=0
                    mal[len(mal)-1]=0
                    nb_pol.insert(0,nb_pol.pop())
                    ouv.insert(0,ouv.pop())
                    mal.insert(0,mal.pop())
                    res.insert(0,res.pop())
                    duree = duree - 1
            else:
                for i in range (no_var_jour-(360*compteur),91):
                    list_graph_ouv.append(0)
                    list_graph_tot_pop.append(0)
                    list_graph_res.append(0)
                    list_graph_mal.append(0)
                    duree = duree - 1
            jour=jour + ((no_var_glo_cycle*360)-(cycle*360)+90-no_var_jour)
            print(jour)
            print(duree)
            return ruche(jour,duree)
```

> Parties identiques dans la construction pour les parties **été – automne – hiver**

> Complexité en **O(durée * âge_abeilles)**

DYNAMIQUE DE LA POPULATION D'UNE RUCHE AU COURS DE L'ANNEE

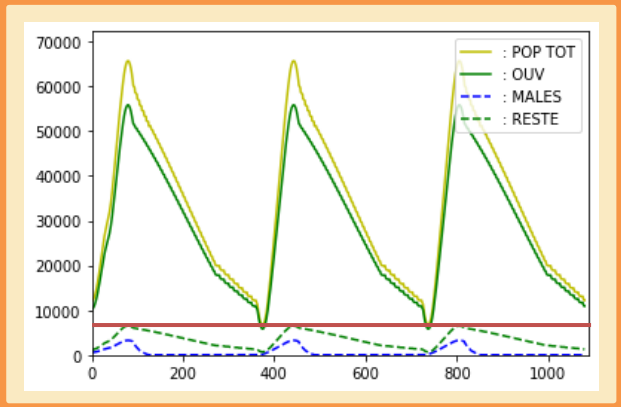


> Modélisation commençant mi mars

> 10.000 abeilles en entrée et on laisse tourner une année.

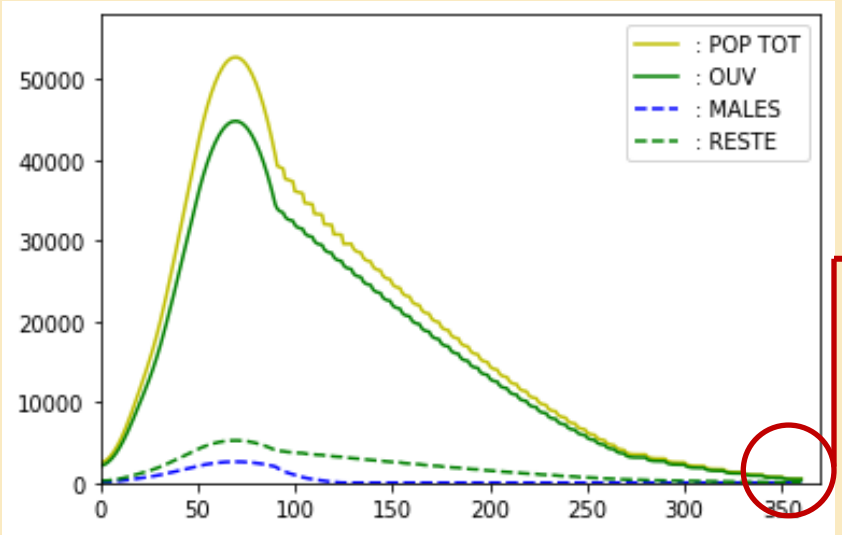
QUE POUVONS NOUS TIRER DE CE MODELE AFIN DE REPONDRE A LA PROBLEMATIQUE ?

Obtention d'un **taux de croissance** annuel dans une colonie de X abeilles



3 CYCLES

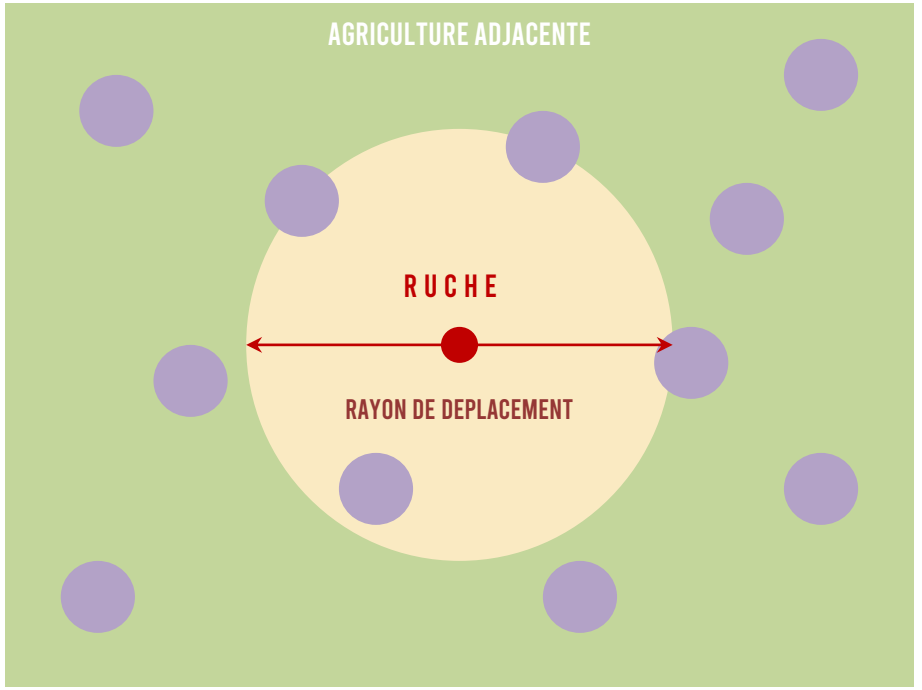
Dépendance entre **croissance de la ruche** et **population totale**



SEUIL CRITIQUE

Population d'environ **2000 à 2500 individus**

MODELISATION DE L'ACTION DU PESTICIDE



- Interaction entre les **abeilles** et les **pesticides**
- Prise en compte **des effets du pesticide** sur le taux de **croissance annuel** des abeilles

IMPACT DES PESTICIDES SUR LES RUCHES

Présent en **grande quantité** dans les champs

Cause des **problèmes cérébraux** rendant ainsi les abeilles **inaptes à rentrer dans la ruche**

Une faible dose suffit pour leur causer ces problèmes cérébraux

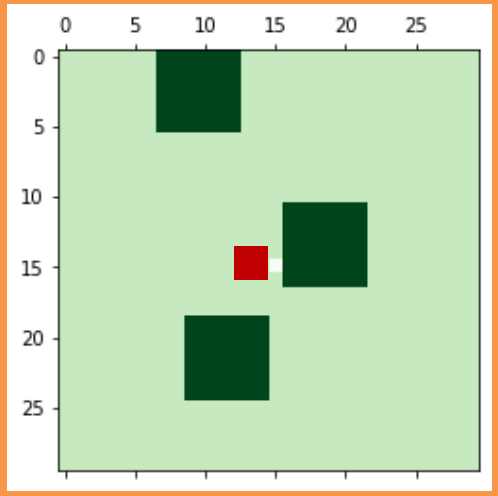
Néonicotinoïdes

MODELISATION 2.
PREREQUIS - CREATION DES CHAMPS

```python
def initialise_monde(t,ruche_co,u,taux_pesticide):
    A=np.zeros((t,t))
    x,y=np.shape(A)
    A=A.tolist()
    print(type(A))
    for i in range (t):
        for j in range (t):
            A[i][j]=[0,0.3,0]
    r=rn.randint(1,5)
    i=0
    while i<=r:
        champs(A)
        i=i+1
    A[t//2][t//2]=[ruche_co,0.1,0]
    return initialisation_abeilles(A,int(len(A)*0.2),u,taux_pesticide)
```

```python
def champs(A):
    x=0.9*len(A)
    a=rn.randint(1,x)
    b=rn.randint(1,x)
    if A[a][b] != [0,0.3,0]:
        return champs(A)
    else:
        bo=len(A)*0.1
        if bo//2 == 1:
            for i in range (int(a-bo),int(a+bo)):
                for j in range (int(b-bo),int(b+bo)):
                    A[i][j]=[0,0.9,0]
        else:
            bo=bo-1
            for i in range (int(a-bo),int(a+bo)):
                for j in range (int(b-bo),int(b+bo)):
                    A[i][j]=[0,0.9,0]
    return A
```

- Complexité en O(**taille de la matrice**)
- la suite



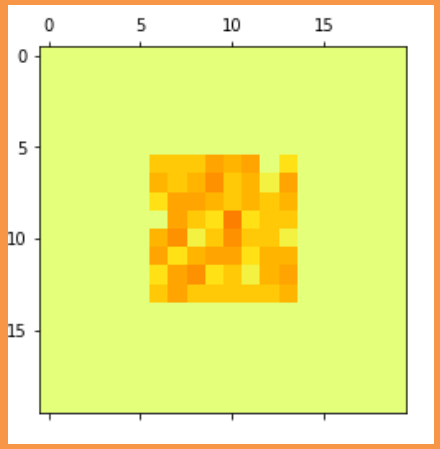
En **rouge**, la ruche et en **vert foncé**, les champs aux alentours

MODELISATION 2.
PREREQUIS - MIS EN PLACE DE LA RUCHE

```python
def initialisation_abeilles(A,r_0,u,taux_pesticide):
    b=A[len(A)//2][len(A)//2][0]
    B=A[:]
    A[len(A)//2][len(A)//2][0]=0
    r=u
    while 0<b:
        for i in range (int((len(A)//2)-r_0),int((len(A)//2)+r_0)):
            for j in range (int((len(A)//2)-r_0),int((len(A)//2)+r_0)):
                m=rn.randint(0,1)
                if m==1 and 0<b:
                    B[i][j][0]=B[i][j][0]+1
                    b=b-1
    return deplacement_abeilles(B,r,r_0,taux_pesticide)
```

```python
def deplacement_abeilles(A,u,r_0,taux_pesticide):
    B=A[:]
    if 0<u:
        affichage_abeille(B)
        ch=0
        for i in range (int((len(B)//2)-r_0),int((len(B)//2)+r_0)):
            for j in range (int((len(B)//2)-r_0),int((len(B)//2)+r_0)):
                al=rn.randint(1,10*B[i][j][1])
                if al==1:
                    ch=B[i][j][0]+ch
                    B[i][j][0]=0
        b=ch
        while 0<b:
            for i in range (int((len(B)//2)-r_0),int((len(B)//2)+r_0)):
                for j in range (int((len(B)//2)-r_0),int((len(B)//2)+r_0)):
                    m=rn.randint(0,1)
                    if m==1 and 0<b:
                        B[i][j][0]=B[i][j][0]+1
                        b=b-1
        return deplacement_abeilles(pesticide(B,taux_pesticide),u-1,r_0,taux_pesticide)
```

> Complexité en O(**taille de la matrice* nombre d'abeilles**)

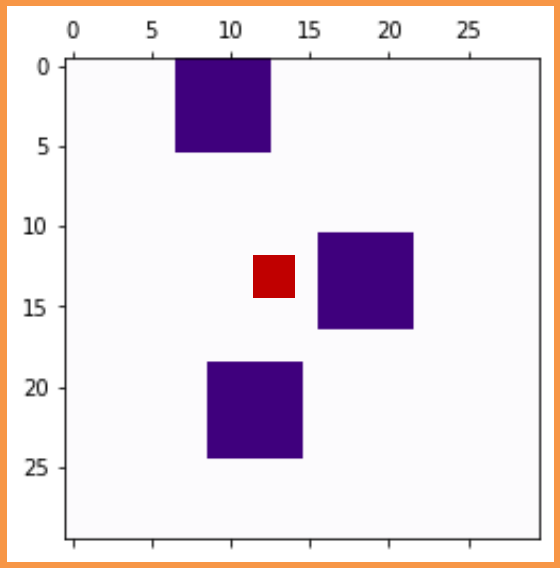


Plus la zone est **foncée**, plus il y a d'abeilles.

MODELISATION 2.
ACTION DES PESTICIDES

```
def pesticide (matrice_initiale,taux_pesticide):
    B=matrice_initiale[:]
    a,b=len(B),len(B)
    print(somme_abeille(B)), cr_abe.append(somme_abeille(B))
    for i in range (a):
        for j in range (b):
            B[i][j][0]=B[i][j][0]*(1-taux_pesticide)
    return B
```

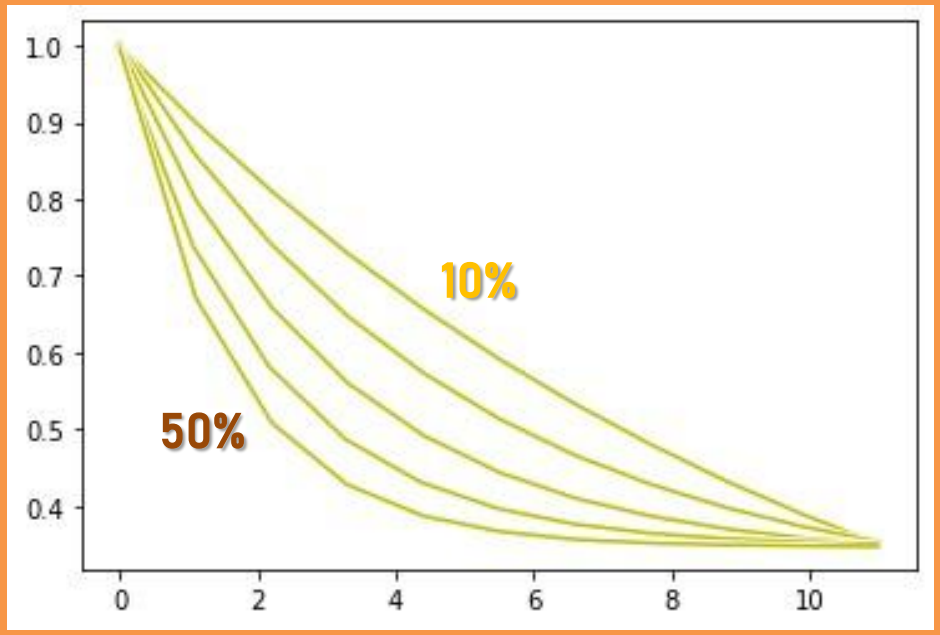
- Complexité en O(taille de la matrice)



En **rouge**, la ruche et en **violet**, les champs contaminés par les pesticides

ACTION DES PESTICIDES - APPLICATION

Graphique indiquant **le pourcentage d'abeilles** présentes dans la ruche au fil des jours pour un **taux de pesticides précis**



- 2000 ABEILLES
- 10 JOURS
- TAUX DE PESTICIDES +10% A CHAQUE COURBE ↓

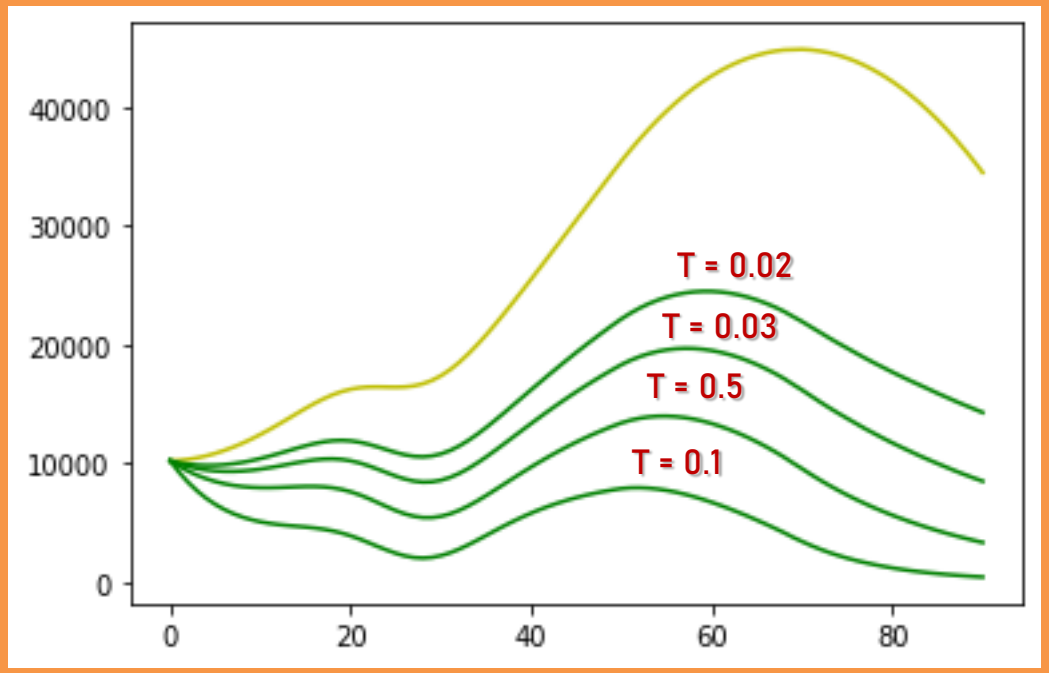
MODELISATION 3.
MISE EN RELATION DES PROGRAMMES

```python
def coalition (taille_M,taux_pesticide):

    nouv_ouv=[ouvriere[0]]
    A=np.zeros((taille_M,taille_M))
    x,y=np.shape(A)
    A=A.tolist()
    print(type(A))
    for i in range (taille_M):
        for j in range (taille_M):
            A[i][j]=[0,0.3,0]
    i=0
    while i<=4:
        champs(A)
        i=i+1
    A[taille_M//2][taille_M//2]=[ouvriere[0],0.1,0]
    B=initialisation_abeilles(A,int(len(A)*0.2),duree,taux_pesticide)
    C=B[:]
    for i in range (0,90):
        if ouvriere[i+1]-ouvriere[i]>=0:
            C[len(A)//2][len(A)//2][0]=C[len(A)//2][len(A)//2][0]+ouvriere[i+1]-ouvriere[i]
            C=initialisation_abeilles(C,int(len(A)*0.2),duree,taux_pesticide)
            C=deplacement_abeilles(C,duree,int(len(A)*0.2), taux_pesticide)
            C=pesticide(C,taux_pesticide)
            nouv_ouv.append(somme_abeille(C))
        else:
            C=initialisation_abeilles_pertes(C,int(len(A)*0.2),duree,taux_pesticide,ouvriere[i]-ouvriere[i+1])
            C=deplacement_abeilles(C,duree,int(len(A)*0.2), taux_pesticide)
            C=pesticide(C,taux_pesticide)
            nouv_ouv.append(somme_abeille(C))
    return nouv_ouv
```

- Complexité en O(taille_matrice*duree)

Prise en compte des effets des pesticides seulement pendant le printemps

MISE EN RELATION – EFFET SUR LES PESTICIDES

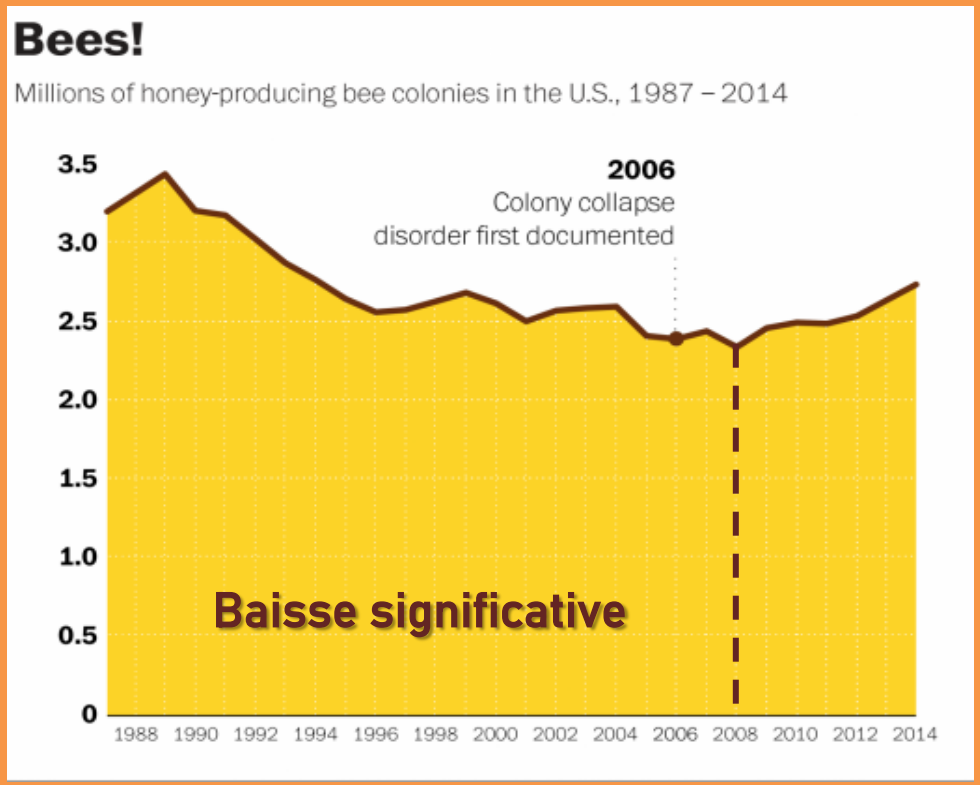


Graphique indiquant **le nombre d'abeilles butineuses avant et après application des pesticides** en fonction du temps

En vert, les abeilles butineuses soumises aux pesticides. En jaune, celles qui ne le sont pas.

COMPARAISON AVEC DES DONNEES REELLES (1)

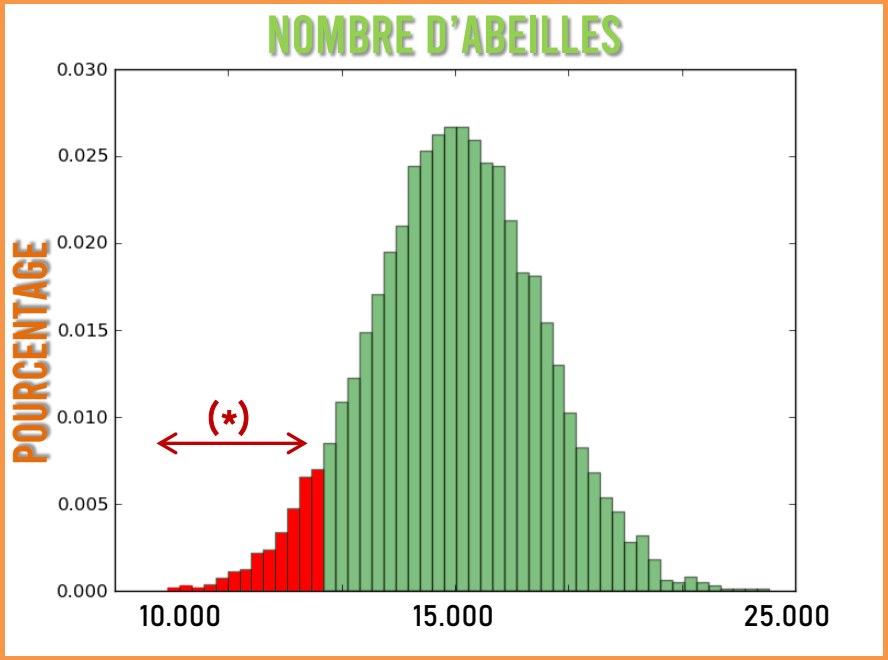
Graphique représentant **le nombre de ruches d'abeilles aux Etats Unis** au fil des années



➢ Quelle **évolution possible** avec le modèle informatique si la tendance ne s'était pas arrêtée en 2008 ?

Sources - Département de l'Agriculture des Etats Unis (USDA)

COMPARAISON AVEC DES DONNEES REELLES (2)



Graphique indiquant **le pourcentage de ruches d'abeilles** en fonction du nombre d'abeilles présent dedans au début du printemps

HYPOTHESES

- 40% des ruches disparaissent à cause des pesticides
- En dessous du seuil critique, la ruche est considérée comme morte

OBJECTIF

- Associer à chaque année un taux de pesticide, responsable de cette perte de ruches

(*) Exemple montrant le nombre de ruches dont la population comprise dans l'intervalle en rouge ne survit pas à un taux de pesticides T égale à 0.2

COMPARAISON AVEC DES DONNEES REELLES (3)

TABLEAU ASSOCIANT **A CHAQUE ANNEE** LA VALEUR DU **TAUX DE PESTICIDE** RESPONSABLE DE LA **PERTE DES RUCHES**

Années	% pesticide
1990 - 1991	0.028
1992 - 1993	0.018
1993 - 1994	0.017
1994 - 1995	0.018
1995 - 1996	0.019
1999 – 2000	0.015
2004 - 2005	0.024

Si l'on prend la valeur moyenne du pourcentage de pesticide, et que l'on suppose que la baisse continue encore après 2008, on obtiendrait **environ 1.95 millions de ruches** en 2014

CONCLUSIONS

- Lien entre **seuil critique** et **effets des pesticides** fortement ressentis. Nécessairement, **un taux de pesticides faible implique une baisse de chance d'attendre ce seuil**.

- Possibilité de **quantifier ce seuil critique** en fonction du taux de pesticides T.

- On peut rester **optimiste** au vu des courbes issues des données réelles. On constate quand même que les restrictions faites vis-à-vis des pesticides ont un impact.

```
def ruche (jour,duree):
    #Définition de toutes les constantes
    compteur=no_var_compteur[0]
    nb_pol = tot_pop
    no_var_jour = jour
    no_var_duree = duree
    saison = fsaison(jour)
    cycle = no_var_cycle[0]
    print('nb_pol vaut ',nb_pol)
    print('no_var_jour vaut ',no_var_jour)
    print('no_var_duree vaut ',no_var_duree)
    #Saison
    # Printemps été, saison optimal pour la fécondation
    while cycle > -1:
        if saison == 'printemps' and (no_var_jour+no_var_duree) >90+360*compteur:
            print('PRINTEMPS ++')
            if sum(tot_pop) != 0:
                for i in range (no_var_jour-(360*compteur),91):
                    nb_pol[0] = nb_pol[0] + int(g(i))
                    ouv[0] = ouv[0] +  int(g(i)*0.85)
                    mal[0] = mal[0] + int(g(i)*0.05)
                    res[0] = res[0] + int(g(i)*0.10)
                    list_graph_ouv.append(sum(ouv))
                    list_graph_tot_pop.append(sum(tot_pop))
                    list_graph_res.append(sum(res))

                    nb_pol[len(nb_pol)-1]=0
                    ouv[len(ouv)-1]=0
                    res[len(res)-1]=0
                    mal[len(mal)-1]=0
                    nb_pol.insert(0,nb_pol.pop())
                    ouv.insert(0,ouv.pop())
                    mal.insert(0,mal.pop())
                    res.insert(0,res.pop())
                    duree = duree - 1
            else:
                for i in range (no_var_jour-(360*compteur),91):
                    list_graph_ouv.append(0)
                    list_graph_tot_pop.append(0)
                    list_graph_res.append(0)
                    list_graph_mal.append(0)
                    duree = duree - 1
            jour=jour + ((no_var_glo_cycle*360)-(cycle*360)+90-no_var_jour)
            print(jour)
            print(duree)
            return ruche(jour,duree)
        elif saison == 'printemps' and (no_var_jour+no_var_duree) <= 90+360*compteur:
            print('PRINTEMPS --')
            if sum(tot_pop) != 0:
                for i in range (duree+1):
                    nb_pol[0] = nb_pol[0] + int(g(i))
                    ouv[0] = ouv[0] +  int(g(i)*0.85)
                    mal[0] = mal[0] + int(g(i)*0.05)
                    res[0] = res[0] + int(g(i)*0.10)
                    list_graph_ouv.append(sum(ouv))
                    list_graph_tot_pop.append(sum(tot_pop))
                    list_graph_res.append(sum(res))
                    list_graph_mal.append(sum(mal))
                    nb_pol[len(nb_pol)-1]=0
                    ouv[len(ouv)-1]=0
                    res[len(res)-1]=0
                    mal[len(mal)-1]=0
                    nb_pol.insert(0,nb_pol.pop())                    #Décalage
                    ouv.insert(0,ouv.pop())
                    mal.insert(0,mal.pop())
                    res.insert(0,res.pop())
            else:
                for i in range (duree+1):
                    list_graph_ouv.append(0)
                    list_graph_tot_pop.append(0)
                    list_graph_res.append(0)
                    list_graph_mal.append(0)
                jour = jour + 1
        elif (saison == 'automne_ete') and (no_var_jour+no_var_duree)>270+360*compteur:
            chgmt = 0
            ajus_pol = 0
            print('AUTOMNE_ETE ++')
            for i in range (no_var_jour,no_var_jour+181):
                chgmt = chgmt + 1
                if chgmt >= mult:
                    nb_pol[len(nb_pol)-1]=0
                    ouv[len(ouv)-1]=0
                    res[len(res)-1]=0
                    mal[len(mal)-1]=0
                    nb_pol.insert(0,nb_pol.pop())                    #Décalage
                    mal.insert(0,mal.pop())
                    res.insert(0,res.pop())
                    ouv.insert(0,ouv.pop())
                    chgmt = chgmt - mult
                for j in range (len(mal)):
                    ajus_pol = mal[j]-int(0.95*mal[j])
                    mal[j]=int(0.95*mal[j])
                    nb_pol[j]=nb_pol[j]-ajus_pol
                list_graph_ouv.append(sum(ouv))
                list_graph_tot_pop.append(sum(tot_pop))
                list_graph_res.append(sum(res))
                list_graph_mal.append(sum(mal))
                duree = duree - 1
            jour=jour + ((no_var_glo_cycle*360)-(cycle*360)+270-no_var_jour)
            return ruche(jour,duree)
        elif (saison == 'automne_ete') and (no_var_jour+no_var_duree)<=270+360*compteur:
            chgmt = 0
            ajus_pol = 0
            print('AUTOMNE_ETE --')
            for i in range (duree+1):
                chgmt = chgmt + 1
                if chgmt >= mult:
                    nb_pol[len(nb_pol)-1]=0
                    ouv[len(ouv)-1]=0
                    res[len(res)-1]=0
                    mal[len(mal)-1]=0
                    nb_pol.insert(0,nb_pol.pop())
                    mal.insert(0,mal.pop())
                    res.insert(0,res.pop())
                    ouv.insert(0,ouv.pop())
                    chgmt = chgmt - mult
                for j in range (len(mal)):
                    ajus_pol = mal[j]-int(0.95*mal[j])
                    mal[j]=int(0.95*mal[j])
                      nb_pol[j]=nb_pol[j]-ajus_pol
                list_graph_ouv.append(sum(ouv))
                list_graph_tot_pop.append(sum(tot_pop))
                list_graph_res.append(sum(res))
                list_graph_mal.append(sum(mal))
                jour = jour + 1
                duree = duree - 1
```

```
    elif saison == 'hiver' and (no_var_jour+no_var_duree) >360+360*compteur:
        chgmt = 0
        # print('HIVER ++');print('HIVER ++');print('HIVER ++');print('HIVER ++')
        for i in range (no_var_jour,no_var_jour+91):
            chgmt = chgmt + 0.5
            if chgmt == mult:
                chgmt = chgmt - mult
                nb_pol[len(nb_pol)-1]=0
                ouv[len(ouv)-1]=0
                res[len(res)-1]=0
                mal[len(mal)-1]=0
                nb_pol.insert(0,nb_pol.pop())
                mal.insert(0,mal.pop())
                ouv.insert(0,ouv.pop())
                res.insert(0,res.pop())
            list_graph_ouv.append(sum(ouv))
            list_graph_tot_pop.append(sum(tot_pop))
            list_graph_res.append(sum(res))
            list_graph_mal.append(sum(mal))
            duree = duree - 1
        jour=jour + ((no_var_glo_cycle*360)-(cycle*360)+360-no_var_jour)
        no_var_cycle[0] = no_var_cycle[0] - 1
        no_var_compteur[0]=no_var_compteur[0]+1
        print('Nous sommes en',2010+no_var_compteur[0])
        return ruche(jour,duree)
    elif saison == 'hiver' and (no_var_jour+no_var_duree) <=360+360*compteur:
        chgmt = 0
        print('HIVER --');print('HIVER --');print('HIVER --');print('HIVER --')
        for i in range (0,duree+1):
            chgmt = chgmt + 0.5
            if chgmt == mult:
                chgmt = chgmt - mult
                nb_pol[len(nb_pol)-1]=0
                ouv[len(ouv)-1]=0
                res[len(res)-1]=0
                mal[len(mal)-1]=0
                nb_pol.insert(0,nb_pol.pop())
                mal.insert(0,mal.pop())
                ouv.insert(0,ouv.pop())
                res.insert(0,res.pop())
            list_graph_ouv.append(sum(ouv))
            list_graph_tot_pop.append(sum(tot_pop))
            list_graph_res.append(sum(res))
            list_graph_mal.append(sum(mal))
            jour = jour + 1
            duree = duree - 1
    return (list_graph_tot_pop,list_graph_ouv,list_graph_mal,list_graph_res)
```

DYNAMIQUE POPULATION

```
def maximum_pop(list_graph_tot_pop):
    max=list_graph_tot_pop[0]
    for i in list_graph_tot_pop:
        if i>max:
            max=i
    return max

def lissage (T):
    for i in range (1,len(T)-2):
        if T[i] == T[i+1]:
            T[i] = int((T[i-1]+T[i+1])/2)
    return T

liss_tot_pop=lissage(list_graph_tot_pop)
liss_ouv=lissage(list_graph_ouv)
liss_mal=lissage(list_graph_mal)
liss_res=lissage(list_graph_res)


max_list_graph_tot_pop=maximum_pop(list_graph_tot_pop)

plt.ylim(-10,max_list_graph_tot_pop+int(0.1*max_list_graph_tot_pop))
plt.xlim(jour,jour+duree+10)
```

```
max_list_graph_tot_pop=maximum_pop(list_graph_tot_pop)

plt.ylim(-10,max_list_graph_tot_pop+int(0.1*max_list_graph_tot_pop))
plt.xlim(jour,jour+duree+10)



tabl_x=range(jour,jour+duree+1,1)
x=np.array(tabl_x)
y_pop=np.array(liss_tot_pop)
y_ouv=np.array(liss_ouv)
y_mal=np.array(liss_mal)
y_res=np.array(liss_res)
(plt.plot(x,y_pop,'y-'),plt.plot(x,y_ouv,'g-'),plt.plot(x,y_mal,'b--'),plt.plot(x,y_res,'g--'))

plt.legend(['POP TOT','OUV','MALES','RESTE',],loc='upper right')
plt.show()
print(list_graph_tot_pop[def_duree-1])
print(list_graph_tot_pop[def_duree-1])
print(list_graph_tot_pop[def_duree-1])
print(list_graph_tot_pop[def_duree-1])
```

AFFICHAGE – DYNAMIQUE POPULATION

AFFICHAGE CHAMP

```python
def champs(A):
    x=0.9*len(A)
    a=rn.randint(1,x)
    b=rn.randint(1,x)
    if A[a][b] != [0,0.3,0]:
        return champs(A)
    else:
        bo=len(A)*0.1
        if bo//2 == 1:
            for i in range (int(a-bo),int(a+bo)):
                for j in range (int(b-bo),int(b+bo)):
                    A[i][j]=[0,0.9,0]
        else:
            bo=bo-1
            for i in range (int(a-bo),int(a+bo)):
                for j in range (int(b-bo),int(b+bo)):
                    A[i][j]=[0,0.9,0]
    return A

def affichage_environnement (A):
    B=np.zeros((len(A),len(A)))
    B=B.tolist()
    for i in range (len(A)):
        for j in range (len(A)):
            B[i][j]=A[i][j][1]
    return plt.matshow(B, cmap=plt.cm.Greens)

def affichage_abeille (A):
    B=np.zeros((len(A),len(A)))
    B=B.tolist()
    for i in range (len(A)):
        for j in range (len(A)):
            B[i][j]=A[i][j][0]
    return plt.matshow(B, cmap=plt.cm.Wistia)
```

```python
def affichage_pesticide(A,taux_pesticide):
    B=np.zeros((len(A),len(A)))
    B=B.tolist()
    for i in range (len(A)):
        for j in range (len(A)):
            if A[i][j][1]==0.9:
                B[i][j]=taux_pesticide
    return plt.matshow(B, cmap=plt.cm.Purples)
```

```python
def somme_abeille(A):
    a=0
    for i in range (len(A)):
        for j in range (len(A)):
            a=A[i][j][0]+a
    return int(a)
```

COMPORTEMENT PESTICIDE

```python
def max_abeille(A):
    a=0
    for i in range (len(A)):
        for j in range (len(A)):
            if A[i][j][0] >= a:
                a=A[i][j][0]
    return int(a)
```

```python
def initialise_monde(t,ruche_co,u,taux_pesticide):
#    assert 100 <= t <= 300
    A=np.zeros((t,t))
    x,y=np.shape(A)
    A=A.tolist()
    print(type(A))
    for i in range (t):
        for j in range (t):
            A[i][j]=[0,0.3,0] # On attribue à toutes les cases une valeur minimal pour indice environnement
    # On génère jusqu'à 4 champs au max
    i=0
    while i<=4:
        champs(A)
        i=i+1
    A[t//2][t//2]=[ruche_co,0.1,0]
    return initialisation_abeilles(A,int(len(A)*0.2),u,taux_pesticide)
```

```python
def pesticide (matrice_initiale,taux_pesticide):
    B=matrice_initiale[:]
    a,b=len(B),len(B)
    print(somme_abeille(B)), cr_abe.append(somme_abeille(B))
    for i in range (a):
        for j in range (b):
            B[i][j][0]=B[i][j][0]*(1-taux_pesticide)
    return B
```

```python
def deplacement_abeilles(A,u,r_0,taux_pesticide):
    B=A[:]
    if 0<u:
        affichage_abeille(B)
        ch=0
        for i in range (int((len(B)//2)-r_0),int((len(B)//2)+r_0)):
            for j in range (int((len(B)//2)-r_0),int((len(B)//2)+r_0)):
                al=rn.randint(1,10*B[i][j][1])
                if al==1:
                    ch=B[i][j][0]+ch
                    B[i][j][0]=0
#                    print(B[i][j][1])
#        print('jour',u)
#        B[len(B)//2][len(B)//2][0]=ch
        b=ch
#        print('b vaut',b)
        while 0<b:
            for i in range (int((len(B)//2)-r_0),int((len(B)//2)+r_0)):
                for j in range (int((len(B)//2)-r_0),int((len(B)//2)+r_0)):
                    m=rn.randint(0,1)
                    if m==1 and 0<b:
                        B[i][j][0]=B[i][j][0]+1
                        b=b-1
#        print('Il y a ',somme_abeille(B))

        return deplacement_abeilles(pesticide(B,taux_pesticide),u-1,r_0,taux_pesticide)
    else:
#        print('jour',u)
        # np.linspace(ruche_co,somme_abeille(B),100)
        return affichage_abeille(B),print('le maximum est',max_abeille(B)),affichage_environnement(B)
```

```
def coalition (taille_M,taux_pesticide):
    duree= 0
    nouv_ouv=[ouvriere[0]]
    A=np.zeros((taille_M,taille_M))
    x,y=np.shape(A)
    A=A.tolist()
    print(type(A))
    for i in range (taille_M):
        for j in range (taille_M):
            A[i][j]=[0,0.3,0]                                   # Indice environnement
    i=0
    while i<=4:
        champs(A)
        i=i+1
    A[taille_M//2][taille_M//2]=[ouvriere[0],0.1,0]
    B=initialisation_abeilles(A,int(len(A)*0.2),duree,taux_pesticide)
    C=B[:]
#assert if valeur positive pour le nombre d'abeille butineuses initialisation_abeilles(A,r_0,u,taux_pesticide)
    for i in range (0,90):
        if ouvriere[i+1]-ouvriere[i]>=0:
            C[len(A)//2][len(A)//2][0]=C[len(A)//2][len(A)//2][0]+ouvriere[i+1]-ouvriere[i]
            C=initialisation_abeilles(C,int(len(A)*0.2),duree,taux_pesticide)
            C=deplacement_abeilles(C,duree,int(len(A)*0.2), taux_pesticide)
            C=pesticide(C,taux_pesticide)
            nouv_ouv.append(somme_abeille(C))
            print('étape ++',i)
        else:
            C=initialisation_abeilles_pertes(C,int(len(A)*0.2),duree,taux_pesticide,ouvriere[i]-ouvriere[i+1])
            C=deplacement_abeilles(C,duree,int(len(A)*0.2), taux_pesticide)
            C=pesticide(C,taux_pesticide)
            nouv_ouv.append(somme_abeille(C))
            print('étape --',i)
    while len(nouv_ouv)<=90:
        nouv_ouv.append(0)
    for i in range (len(nouv_ouv)):
        if nouv_ouv[i]<0:
            nouv_ouv[i]=0
    return nouv_ouv
```

MISE EN RELATION DE TOUS LES PROGRAMMES