DUPONT TRISTAN

Thème : Transport

Numéro d’inscription : 36825

OPTIMISATION DE LA POLLINISATION PAR LES ABEILLES SELON LE PLACEMENT DE LA RUCHE

PROBLÉMATIQUE

COMMENT PLACER UNE RUCHE D'ABEILLES POUR OPTIMISER LA POLLINISATION DE CHAMPS DE FLEURS ?

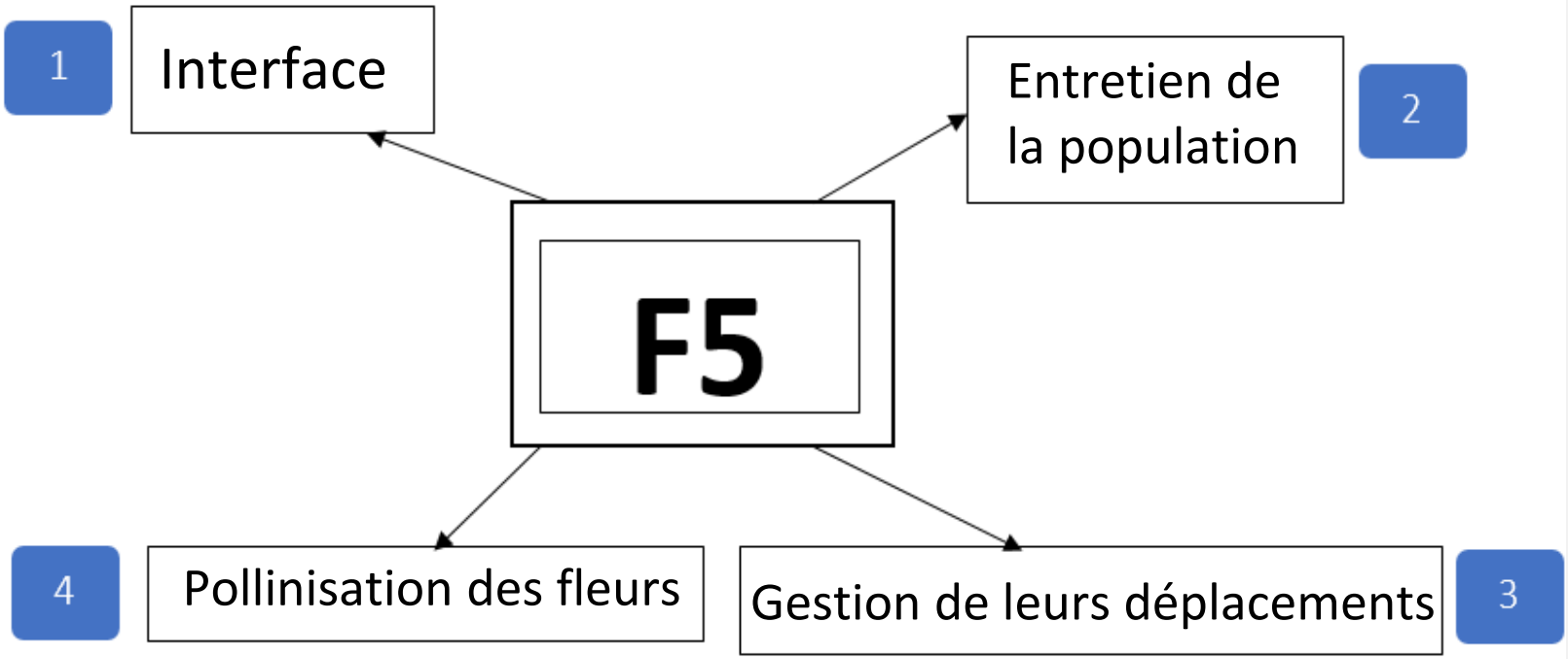
SOMMAIRE

- I) INTRODUCTION
- II) PRESENTATION DE L'ALGORITHME
- III) PLACEMENT DE LA RUCHE
  - II.i) METHODE BARYCENTRE
  - II.ii) METHODE CENTRE RADICAL
  - II.iii) METHODE INTEGRALE

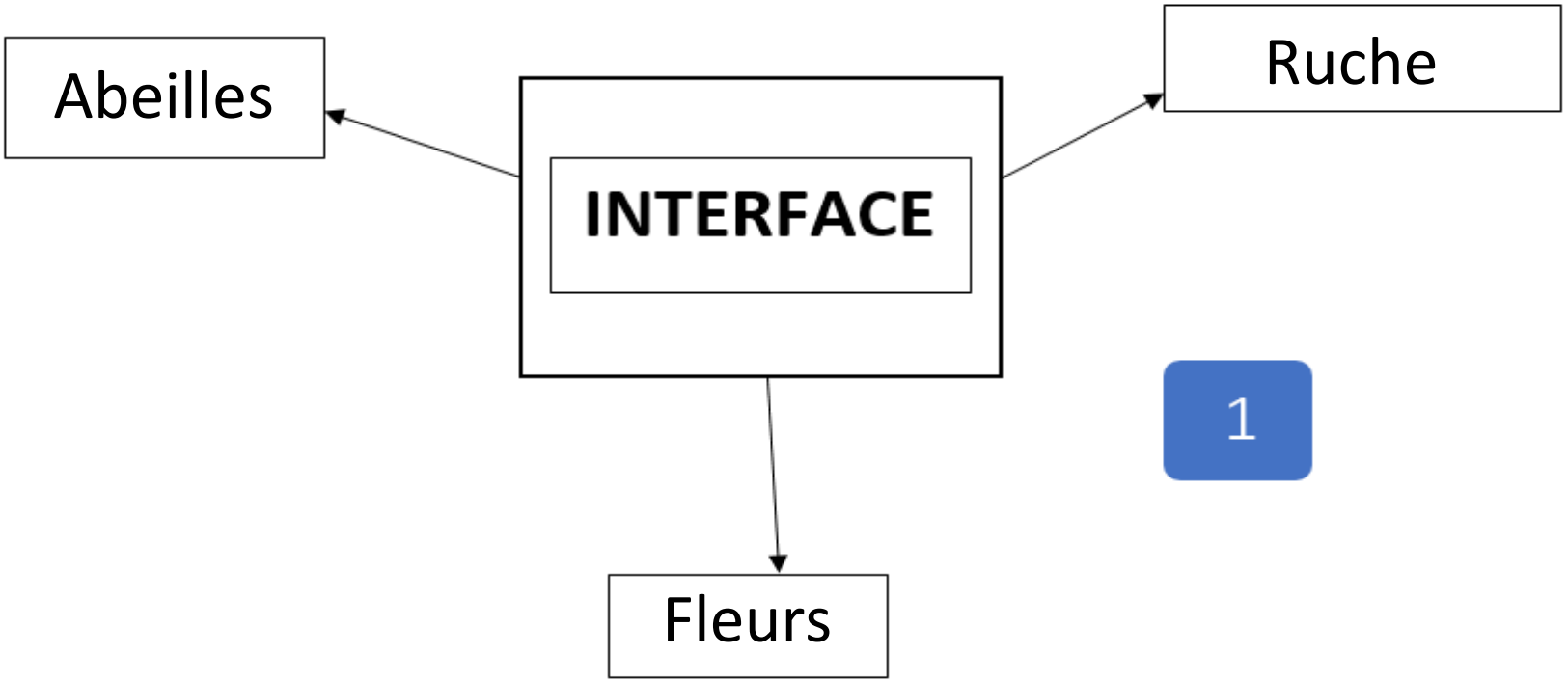
INTRODUCTION

- Pollinisation uniquement par les abeilles
- Fleurs de la même espèce
- Abeilles ouvrières
- Champs de fleurs

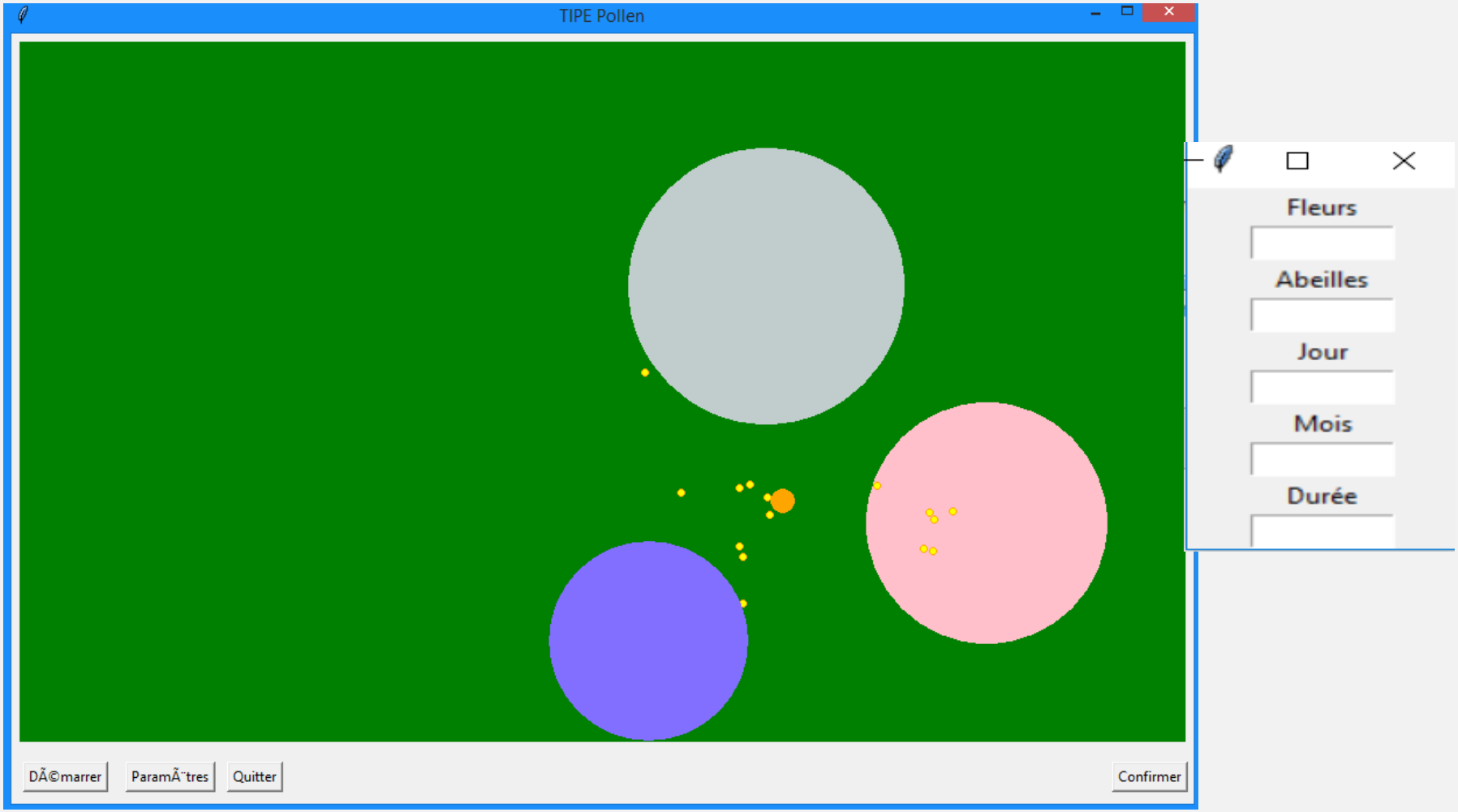
PRÉSENTATION DE L'ALGORITHME



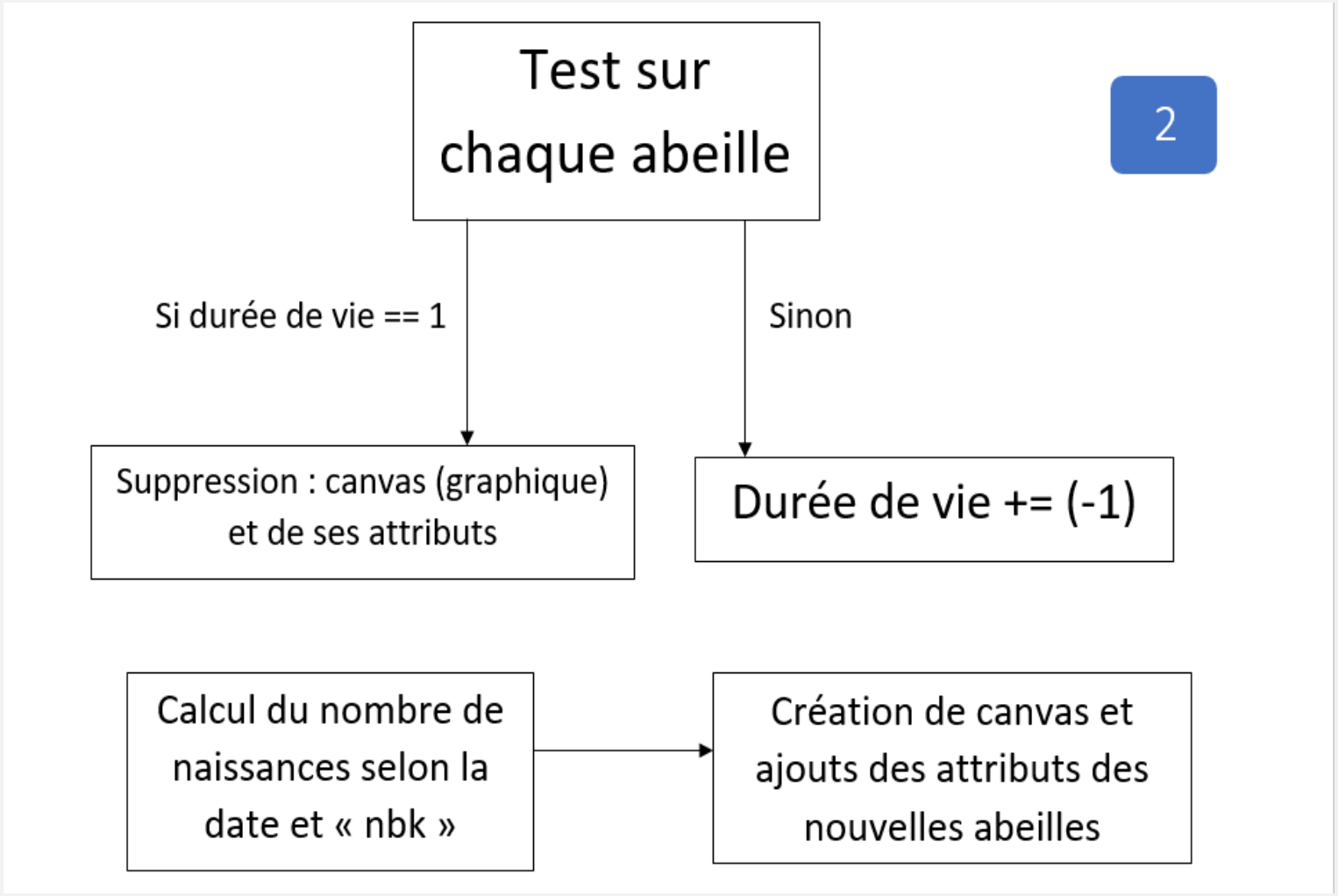
PRÉSENTATION DE L'ALGORITHME



PRÉSENTATION DE L'ALGORITHME



PRÉSENTATION DE L'ALGORITHME



PRÉSENTATION DE L'ALGORITHME

3

Recherche du champ le plus proche de chaque abeille

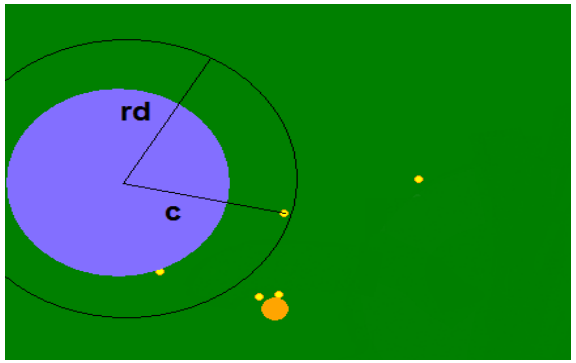
Si c<rd

Sinon

Test sur la position de l'abeille par rapport aux champs

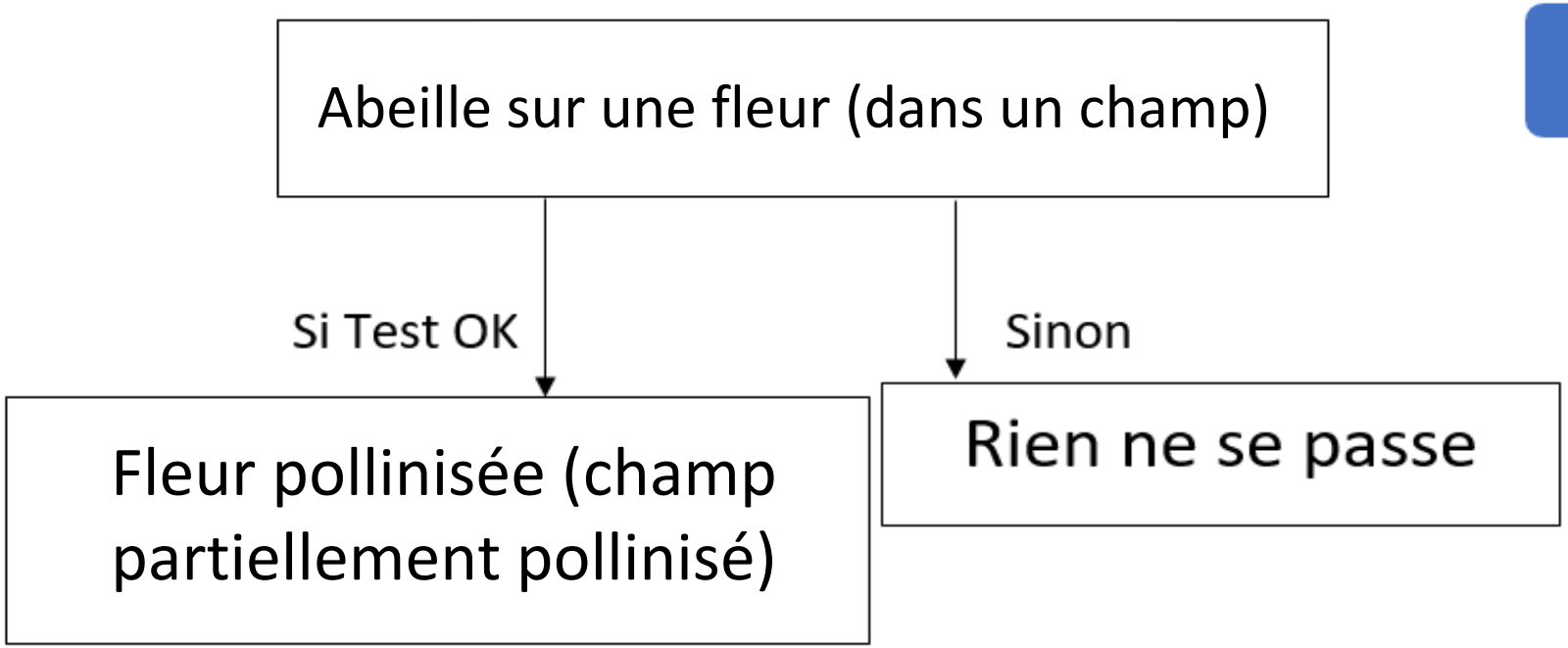
Mouvement aléatoire

Mouvement rectiligne uniforme en direction du champ



PRÉSENTATION DE L'ALGORITHME

4



PLACEMENT DE LA RUCHE

- I) Méthode barycentre
- II) Méthode centre radical
- III) Méthode intégrale

MÉTHODE BARYCENTRE

- Soit (a1, . . . , an) ∈ $(\mathbb{R}_+^*)^n$, n coefficients ou *poids*, soit (A1, . . . , An) ∈ $(\mathbb{R}^2)^n$, n points du plan.
- On a :

$$\text{bar}(A_i, a_i)_{i=1,\dots,n} = \frac{1}{\sum a_i} \sum a_i A_i$$

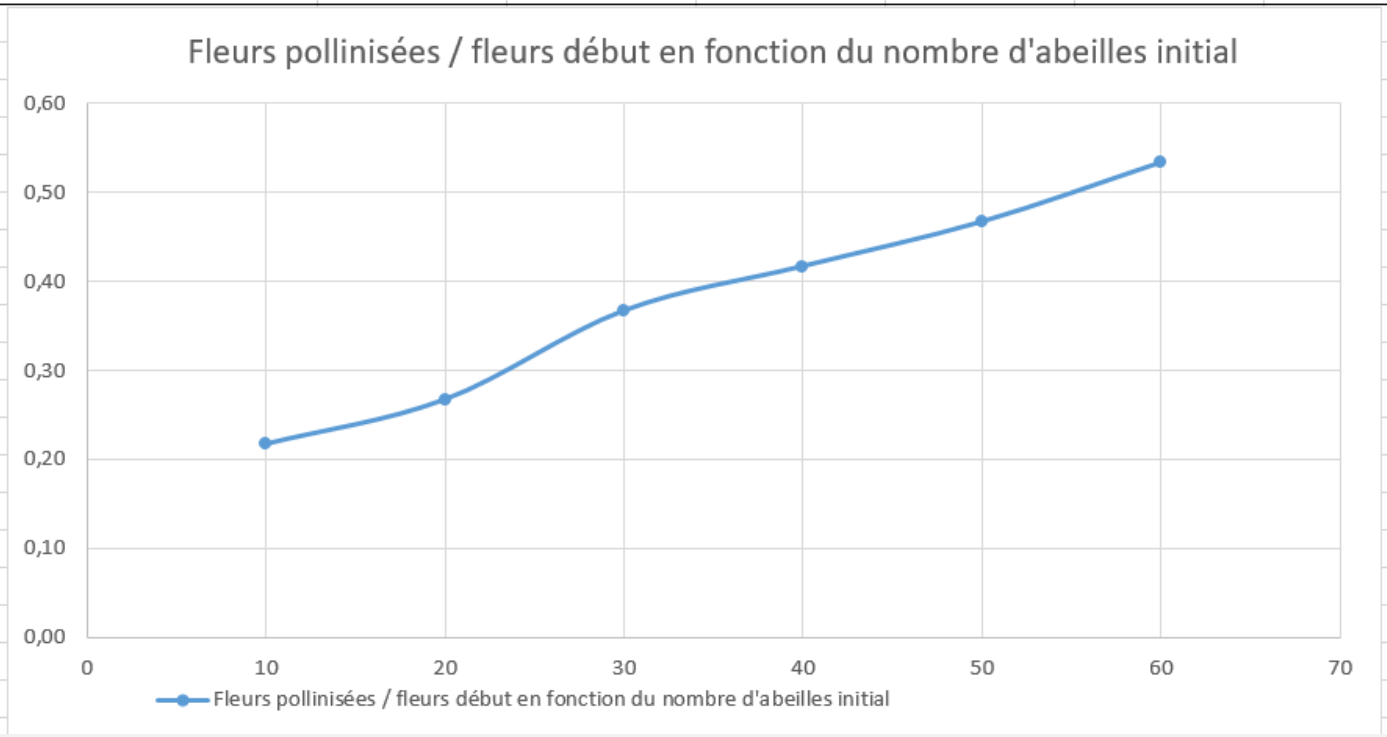
Ici, nous avons :

$$\text{bar}(A_i, a_i)_{i=1,\dots,n} = \frac{1}{\sum \pi r_i^2} \sum \pi r_i^2 A_i$$

MÉTHODE BARYCENTRE

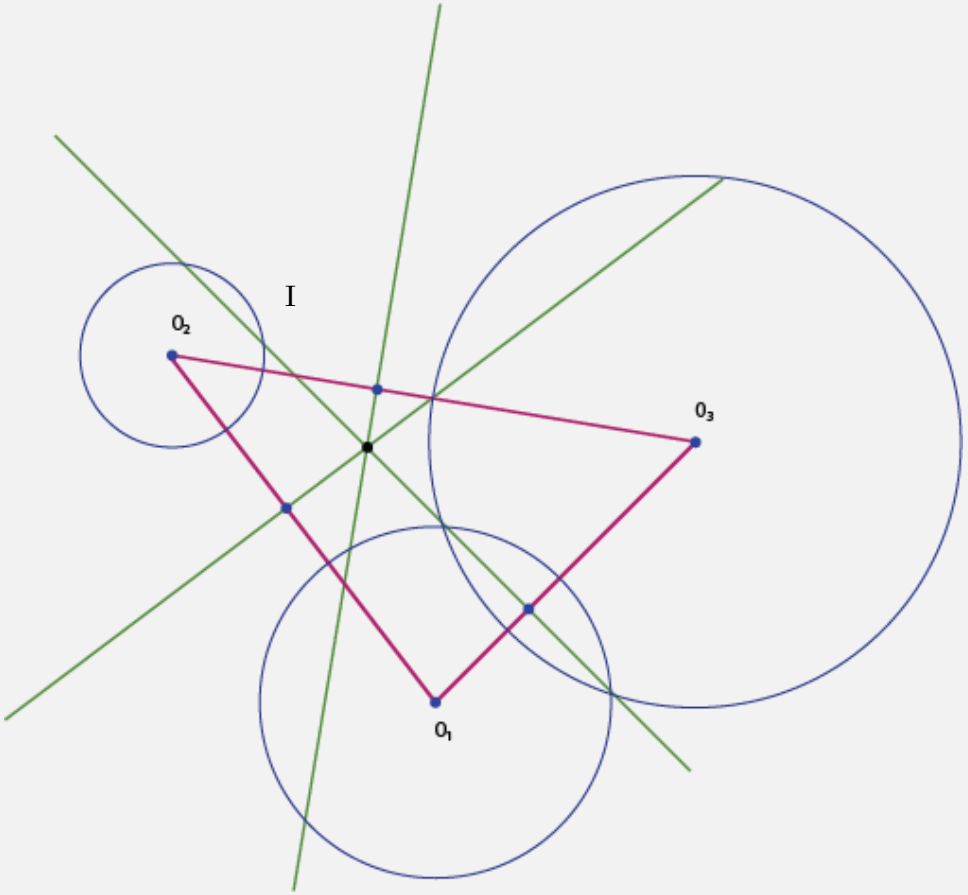
Nombre abeilles début	10	20	30	40	50	60
Nombre fleurs début	60	60	60	60	60	60
Nombre fleurs pollinisées	13	16	22	25	28	32
Fleurs pollinisées / fleurs début	0,22	0,27	0,37	0,42	0,47	0,53

Date : AVRIL-JUIN



MÉTHODE CENTRE RADICAL

- Centre radical de 3 cercles



MÉTHODE CENTRE RADICAL

- Données : $O_i = (x_i, y_i)$, $r_i > 0$ *pour* $i \in [\![1,3]\!]$. $M = (x, y) \in$ à l'axe radical de C_i et C_j noté $\Delta_{i,j}$ équivaut à :

$$P_i(M)=P_j(M) \Leftrightarrow P\begin{pmatrix} x \\ y \end{pmatrix} =$$

$$\begin{pmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}=\begin{pmatrix} \frac{-(x_1-x_2)(x_1+x_2)-(y_1-y_2)(y_1+y_2)+R_1^2-R_2^2}{2} \\ \frac{-(x_1-x_3)(x_1+x_3)-(y_1-y_3)(y_1+y_3)+R_1^2-R_3^2}{2} \end{pmatrix}=\begin{pmatrix} a \\ b \end{pmatrix}.$$

Qui est un système de Cramer, on a donc :

$$x = \frac{1}{\det(P)}\begin{vmatrix} a & y_2 - y_1 \\ b & y_3 - y_1 \end{vmatrix} = \frac{a(y_3-y_1)-b(y_2-y_1)}{(x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1)}$$

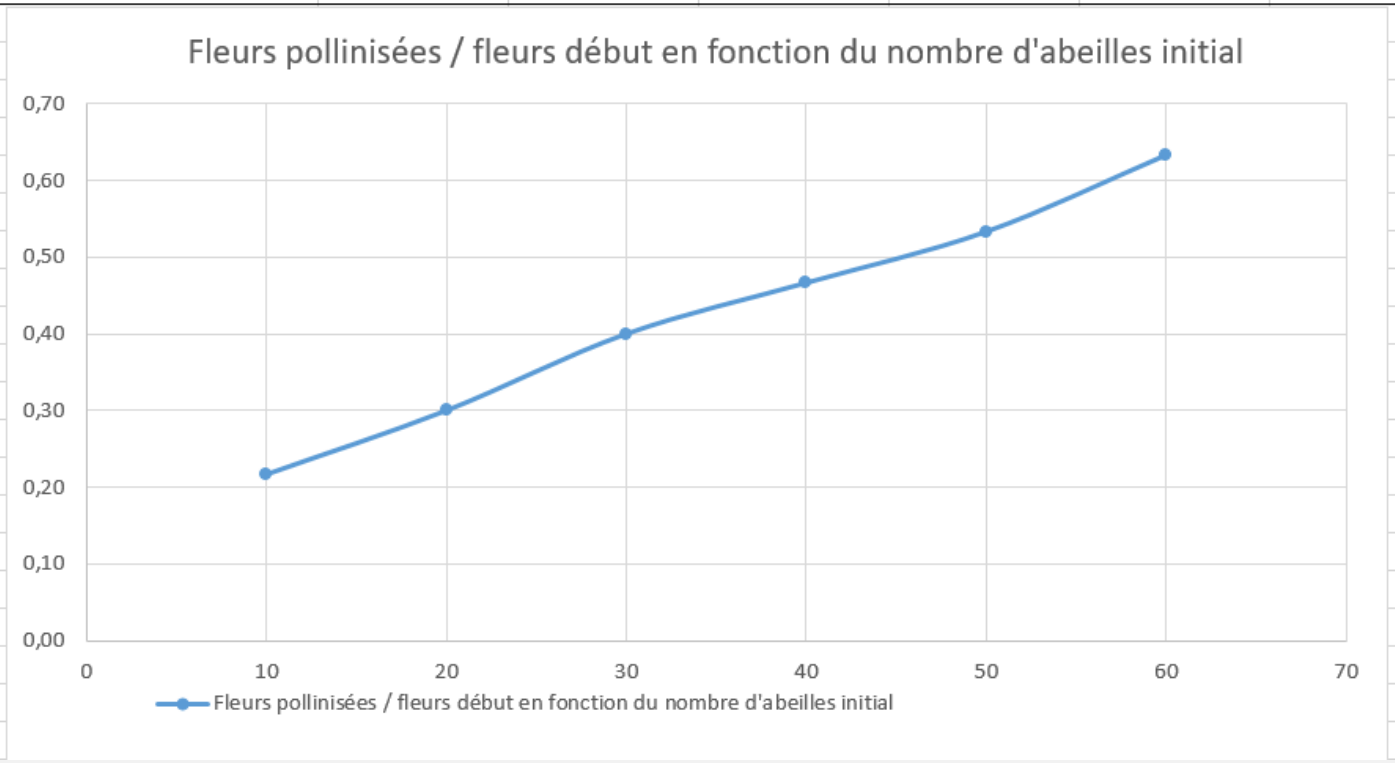
$$y = \frac{1}{\det(P)}\begin{vmatrix} x_2 - x_1 & a \\ x_3 - x_1 & b \end{vmatrix} = \frac{b(x_2-x_1)-a(x_3-x_1)}{(x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1)}$$

D'où les coordonnées du centre radical de nos trois cercles.

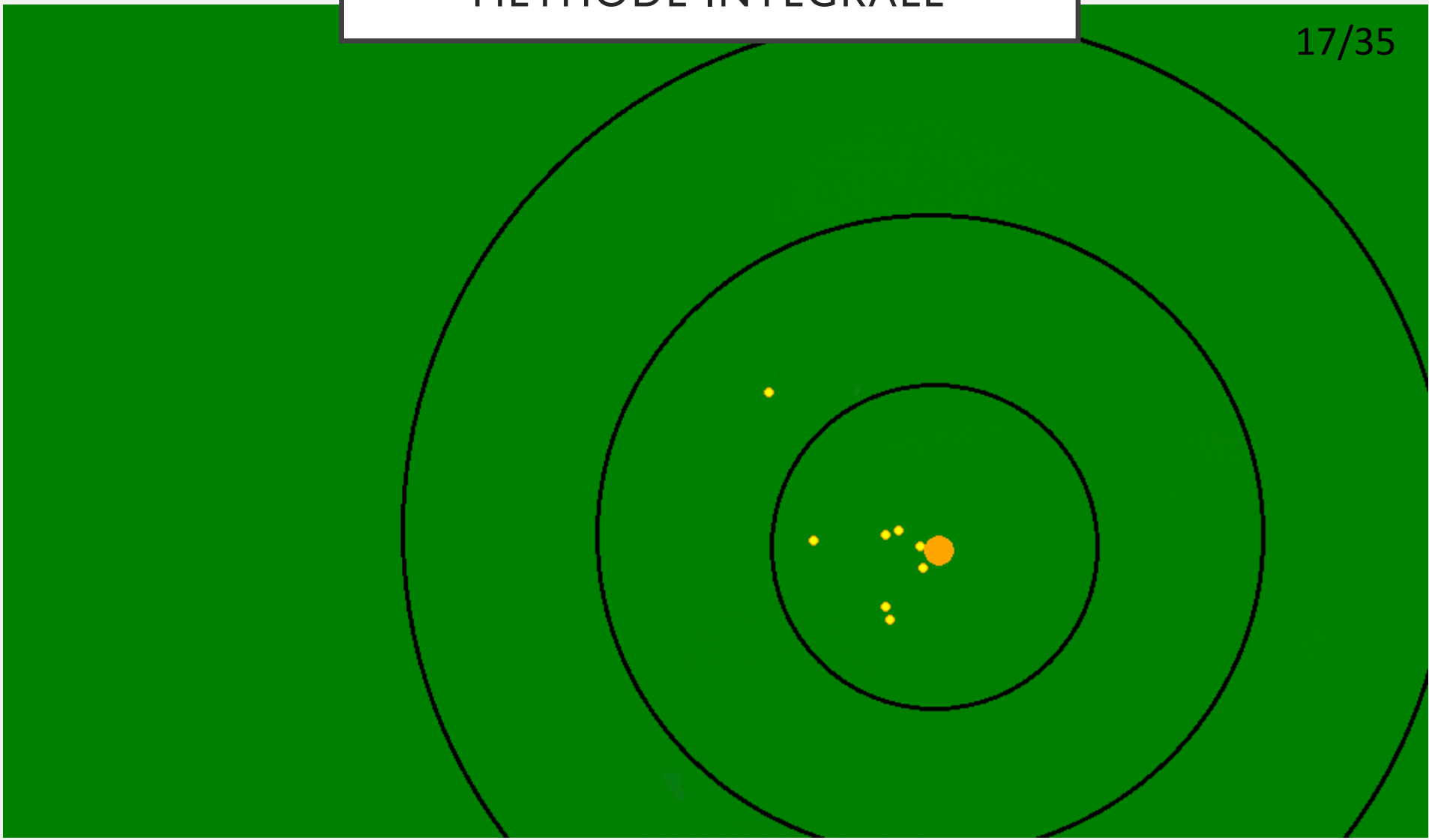
MÉTHODE CENTRE RADICAL

Nombre abeilles début	10	20	30	40	50	60
Nombre fleurs début	60	60	60	60	60	60
Nombre fleurs pollinisées	13	18	24	28	32	38
Fleurs pollinisées / fleurs début	0,22	0,30	0,40	0,47	0,53	0,63

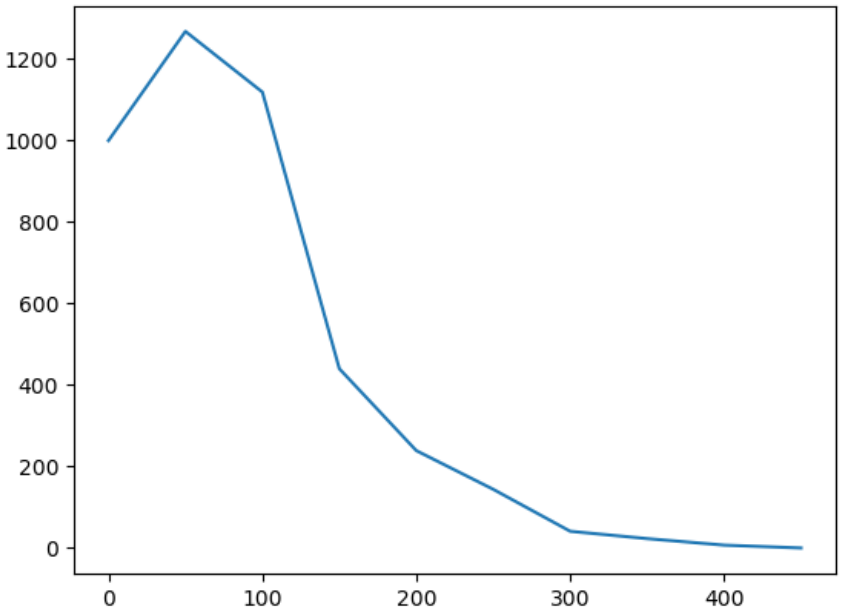
Date : AVRIL-JUIN



MÉTHODE INTEGRALE

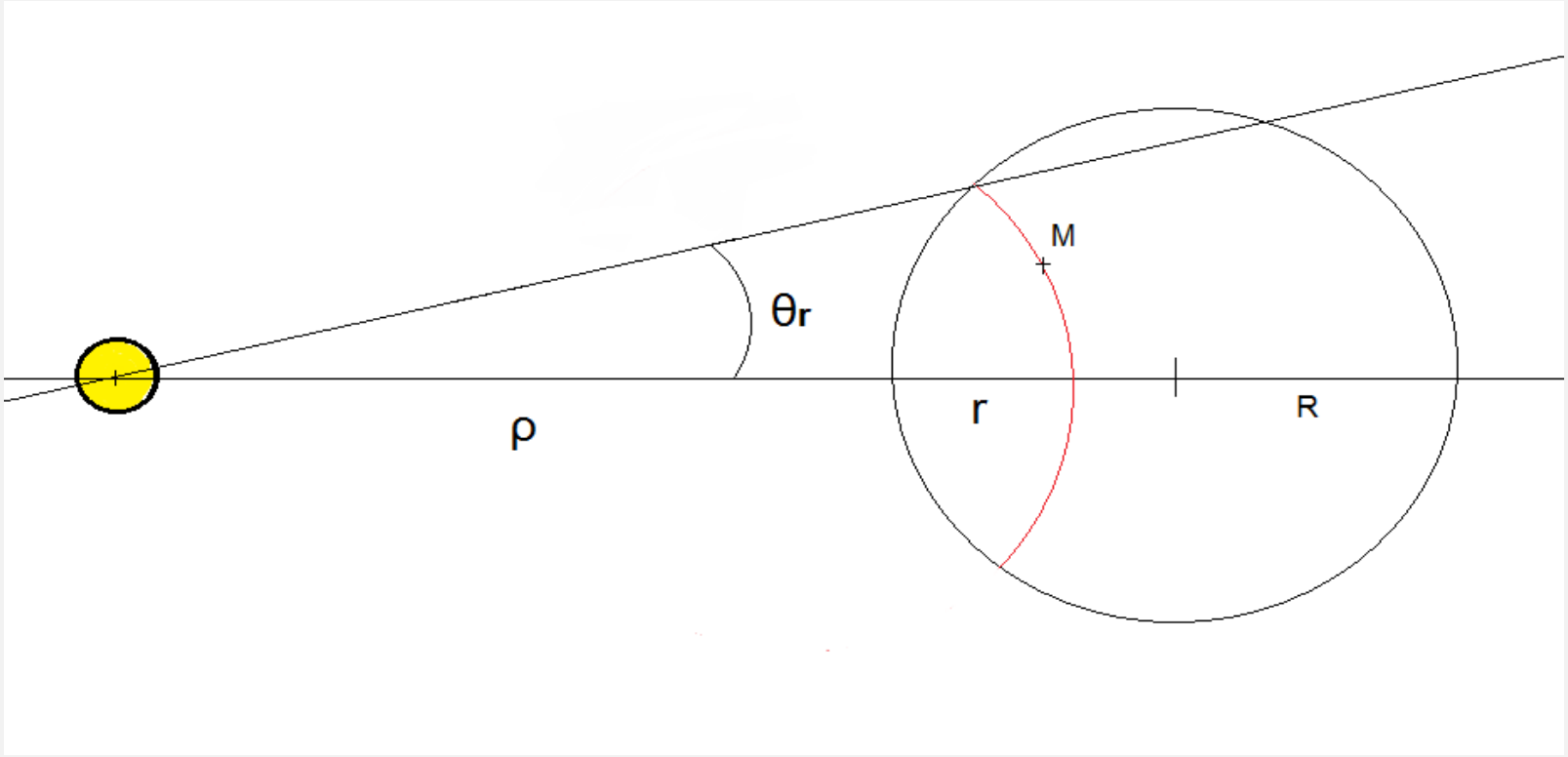


MÉTHODE INTEGRALE



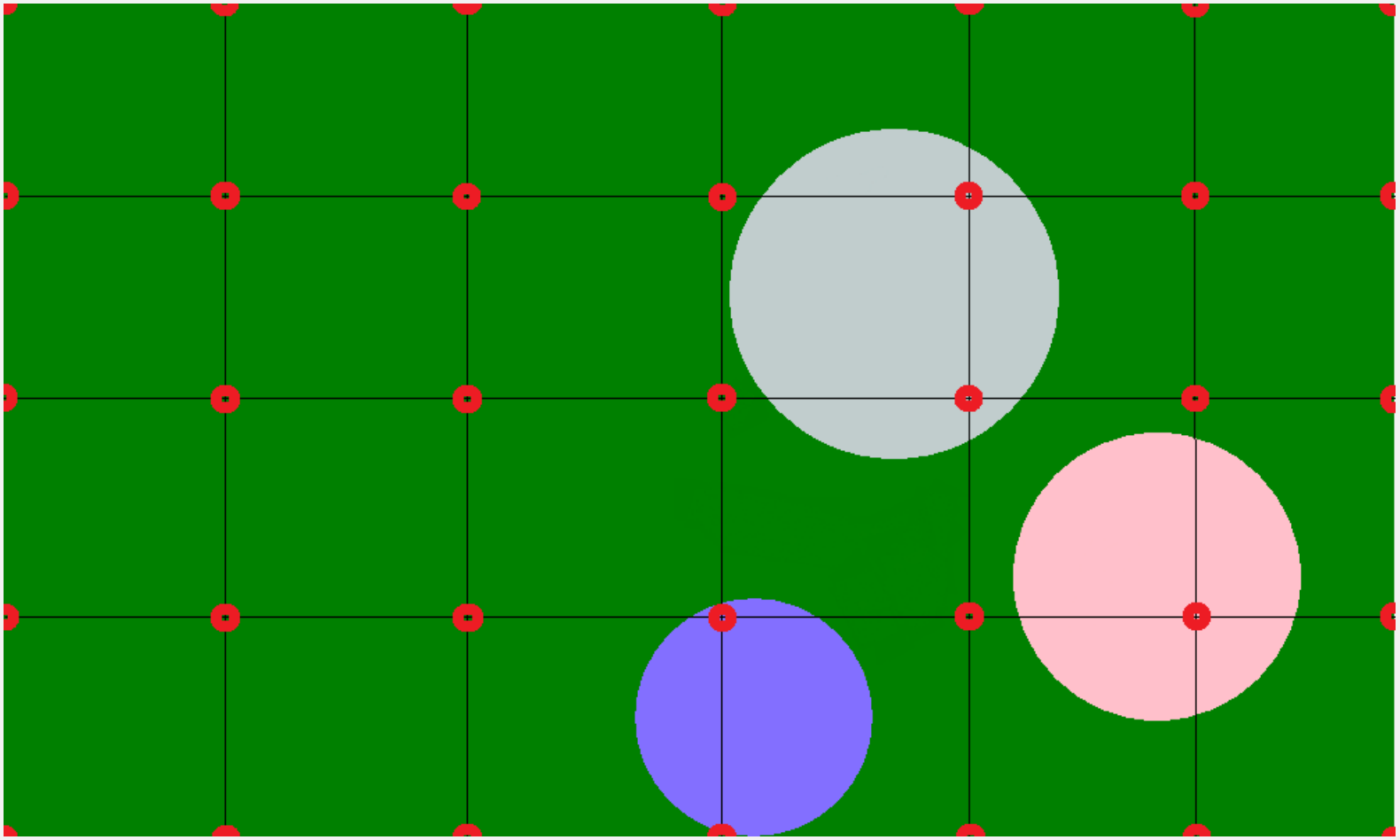
$$P(X) = 1000 - 55{,}2429286X + 2{,}93900171X^2 - 0{,}0523200389X^3 + 0{,}00047984861X^4 - 0{,}0000210784889X^5 + 0{,}000000055943778X^6 - 0{,}000000000786679365X^7 + 0{,}000000000000456190476X^8$$

MÉTHODE INTEGRALE



$$\int_{\rho-R}^{\rho+R}\int_{-\theta_r}^{\theta_r} P(r)\,r\,d\theta\,dr \qquad \text{Où } \theta_r = Arccos\left(\frac{r^2+\rho^2-R^2}{2\rho r}\right)$$

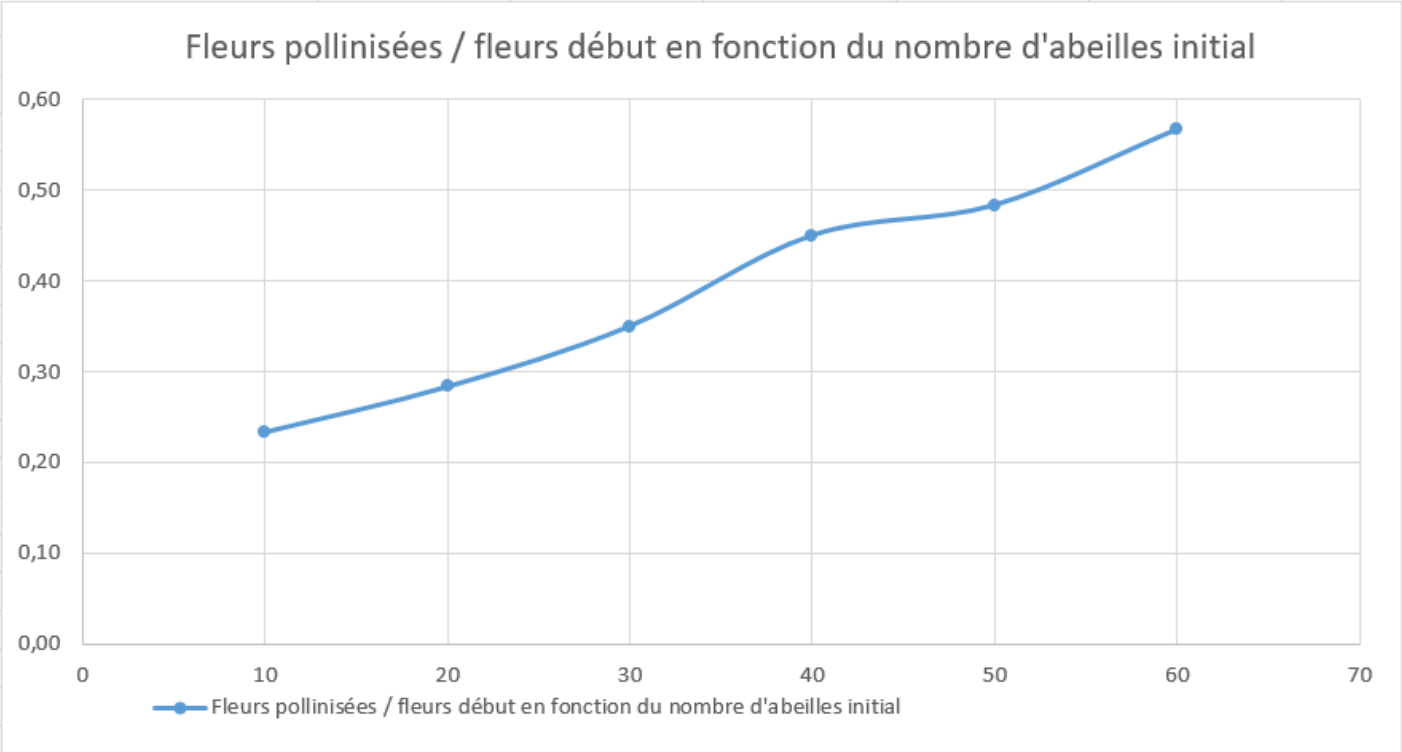
MÉTHODE INTEGRALE



MÉTHODE INTEGRALE

Nombre abeilles début	10	20	30	40	50	60
Nombre fleurs début	60	60	60	60	60	60
Nombre fleurs pollinisées	14	17	21	27	29	34
Fleurs pollinisées / fleurs début	0,23	0,28	0,35	0,45	0,48	0,57

Date : AVRIL-JUIN



CONCLUSION

- Bilan
- Critique / Améliorations possibles

ANNEXES

BIBLIOGRAPHIE DETAILLEE

[1] https://fr.wikipedia.org/wiki/Pollinisation, consultation le 13/08/2018
[2] Olivier Faure : Simulation du mouvement brownien et des diffusions : https://pastel.archivesouvertes.fr/tel-00523258/document, consultation le 10/09/2018
[3] Enrique Munoz : RMS, Etude qualitative d'un système de Lotka-Volterra : Volume 112, numéro 3, 2001/2002
[4] http://debart.pagesperso-orange.fr/ts/geometrie_cercle.html, consultation le 03/12/2018

REFERENCE DU CONTACT

Jean-Baptiste MALRAUX, technicien, animateur chez l'ADAFC (Association pour le développement de l'apiculture en Franche-Comté)
Téléphone : 03.81.54.71.57
Adresse mail : jean-baptiste.malraux@adafr.adafrance.org

ALGORITHMES

```
from tkinter import *
import math, random
import time
from numpy.polynomial import Polynomial
from sympy import binomial
from math import *
"""from colors import *"""

Fleurs = []    #Liste des fleurs
Fx=[]    #Liste des abscisses des fleurs
Fy=[]    # '''  des ordonnées ''''''
M=[]    #Liste des coordonnées de la ruche
C=[500000]  #"Liste auxiliaire qui va contenir les fleurs pollinisées meme plusieurs fois
Cp=[]    #Liste qui va contenir les fleurs pollinisées de façon distincte
P=[] #liste des poids de chaque fleur
Fr=[]
Lc1=[]
Lc2=[]
Lc3=[]
Couleur=COLORS = ['LavenderBlush3','azure3', 'SlateBlue1', 'RoyalBlue2',  'blue4',
    'DodgerBlue2', 'SteelBlue1', 'DeepSkyBlue3', 'SkyBlue1', 'LightSkyBlue1']
W = [pi/2,0,pi/4,0, (3*pi)/16, 0, (15*pi)/96, 0, (105*pi)/768]
A = [1000, -55.2429286, 2.93900171, -0.0523200389, 0.00047984861, -0.00000210784889, 0.00000000559437778,
-0.00000000000786679365, 0.000000000000000456190476]


def Fleur():
    """ Dessine des fleurs de centre (x,y) et de rayon r """
    global Arret, f
    if Arret == False:
        v = 0
        while v < f:
            r = random.randint(10,20)
            x = random.randint(100,300)
            y = random.randint(100,300)
            Lc1.append((r**2)/300)
            Lc2.append(0)
            Lc3.append(0)
            Fx.append(x)
            Fy.append(y)
            Fr.append(r)
            Fleurs.append(Canevas.create_oval(x-r, y-r, x+r, y+r, outline='pink', fill='pink'))
            P.append(1)
            v += 1
        else:
            Ruche()
            return(Fleurs)
            return(Fx)
            return(Fy)

```

```
def Ruche():
    """ Dessine une ruche de centre (x,y) et de rayon r """
    global Arret
    if Arret == False:
        x,y=position_ruche(Fx,Fy,Fr)
        r = 10
        Canevas.create_oval(x-r, y-r, x+r, y+r, outline='orange', fill='orange')
        M.append(x)
        M.append(y)
        Abeille()

def Abeille():
    """ Dessine des abeilles de centre (X[p],Y[p]) et de rayon RAYON"""
    global Arret,a,u,g,G,J,k,j,m,t,Abeilles,x,X,Y,DX,DY,RAYON,LARGEUR,HAUTEUR
    if Arret == False:
        cdf=0
        k = 0
        u = 0
        g = 0
        num_ab = a
        Abeilles = []
        G=[]
        J=[]
        X = []
        Y = []
        L = []
        C = []
        DX = []
        DY = []
        vitesse = []
        angle = []
        RAYON = 3
        for p in range (num_ab) : #Création des abeilles initiales
            X.append(M[0])
            Y.append(M[1])
            L.append(duree_vie(j,m))
            G.append(random.randint(0,80))
            J.append(0)
            Abeilles.append(Canevas.create_oval(X[p]-RAYON, Y[p]-RAYON, X[p]+RAYON, Y[p]+RAYON, outline='orange',
fill='yellow'))
            vitesse.append(random.uniform(1.8,2)*3)
            angle.append(random.uniform(0,2*math.pi))
            DX.append(vitesse[p]*math.cos(angle[p]))
            DY.append(vitesse[p]*math.sin(angle[p]))
        while k<t: #Gestion du temps, des naissances et des morts
            k+=0.05
            g+=1
            r=0
            if g==10:
```

```
                g=0
                while r!=len(L):
                    if L[r]==1:
                        del L[r]
                        Canevas.delete(Mafenetre, Abeilles[r])
                        del Abeilles[r]
                        del vitesse[r]
                        del angle[r]
                        del DX[r]
                        del DY[r]
                        del X[r]
                        del Y[r]
                        del G[r]
                        del J[r]
                        num_ab -= 1
                    else:
                        L[r]=L[r]-1
                        r+=1
                for o in range(0,rep_abeilles(date1(j,m,k),date2(j,m,k),len(L))):
                    L.append(duree_vie(date1(j,m,k),date2(j,m,k)))
                    X.append(M[0])
                    Y.append(M[1])
                    G.append(random.randint(0,80))
                    J.append(0)
                    Abeilles.append(Canevas.create_oval(X[len(X)-1]-RAYON, Y[len(Y)-1]-RAYON, X[len(X)-1]+RAYON,
Y[len(Y)-1]+RAYON, outline='orange', fill='yellow'))
                    vitesse.append(random.uniform(1.8,2)*3)
                    angle.append(random.uniform(0,2*math.pi))
                    DX.append(vitesse[len(vitesse)-1]*math.cos(angle[len(angle)-1]))
                    DY.append(vitesse[len(vitesse)-1]*math.sin(angle[len(angle)-1]))
                    num_ab += 1


            for p in range (num_ab) : #Gestion des déplacements des abeilles
                rd=200
                c=10000
                for kk in range(len(Fx)):

                    if ((X[p]-Fx[kk])**2+(Y[p]-Fy[kk])**2)**0.5 < c :
                        c=((X[p]-Fx[kk])**2+(Y[p]-Fy[kk])**2)**0.5
                        jj=kk
                        alpha=math.acos((X[p]-Fx[jj])/((X[p]-Fx[kk])**2+(Y[p]-Fy[kk])**2)**0.5)

                if c<rd and G[p]<40:
                    G[p]+=1
                    if X[p]-Fx[jj] > 0:
                        if Y[p]-Fy[jj] > 0:
                            vitesse[p] = random.uniform(1.8,2)*2
                            angle[p] = math.pi+alpha
                            DX[p] = vitesse[p]*math.cos(angle[p])
                            DY[p] = vitesse[p]*math.sin(angle[p])
```

```
148                     else :
149                         vitesse[p] = random.uniform(1.8,2)*2
150                         angle[p] =math.pi-alpha
151                         DX[p] = vitesse[p]*math.cos(angle[p])
152                         DY[p] = vitesse[p]*math.sin(angle[p])
153                 else :
154                     if Y[p]-Fy[jj] > 0:
155                         vitesse[p] = random.uniform(1.8,2)*2
156                         angle[p] = -(math.pi-alpha)
157                         DX[p] = vitesse[p]*math.cos(angle[p])
158                         DY[p] = vitesse[p]*math.sin(angle[p])
159                     else:
160                         vitesse[p] = random.uniform(1.8,2)*2
161                         angle[p] =-(math.pi+alpha)
162                         DX[p] = vitesse[p]*math.cos(angle[p])
163                         DY[p] = vitesse[p]*math.sin(angle[p])
164             else:
165                 if G[p]>=40:
166                     if J[p]<50:
167                         J[p]+=1
168                         vitesse[p] = random.uniform(1.8,2)*7
169                         angle[p] = random.uniform(0,2*math.pi)
170                         DX[p] = vitesse[p]*math.cos(angle[p])
171                         DY[p] = vitesse[p]*math.sin(angle[p])
172                     else:
173                         G[p]=0
174                         J[p]=0
175                         vitesse[p] = random.uniform(1.8,2)*7
176                         angle[p] = random.uniform(0,2*math.pi)
177                         DX[p] = vitesse[p]*math.cos(angle[p])
178                         DY[p] = vitesse[p]*math.sin(angle[p])
179                 else:
180                     vitesse[p] = random.uniform(1.8,2)*7
181                     angle[p] = random.uniform(0,2*math.pi)
182                     DX[p] = vitesse[p]*math.cos(angle[p])
183                     DY[p] = vitesse[p]*math.sin(angle[p])
184 
185 
186             Canevas.move(Abeilles[p], DX[p], DY[p])
187 
188             X[p] = X[p]+DX[p]
189             Y[p] = Y[p]+DY[p]
190 
191             for i in range(len(Fx)): #Pollinisation des fleurs
192                 numal = random.randint(0,13)
193                 if abs(X[p]-Fx[i]) < (3/2)*Fr[i] and abs(Y[p]-Fy[i]) < (3/2)*Fr[i] and numal == 2:
194                     if Lc3[i]<9:
195                         Cp.append(Fx[i])
196                     Lc2[i]+=1
197                     r = Fr[i]
198                     x = Fx[i]
```

```
                        y = Fy[i]
                        if Lc2[i]>=Lc1[i] and Lc3[i]<9:
                            Lc2[i]=0
                            Lc3[i]+=1
                            Fleurs.append(Canevas.create_oval(x-r, y-r, x+r, y+r, outline=Couleur[Lc3[i]],
fill=Couleur[Lc3[i]]))
                            del Fleurs[i]


            Mafenetre.update()
            time.sleep(0.1)


def Parametres():
    global l,f,b,a,o,j,u,m,p,t
    l.pack()
    f.pack()
    b.pack()
    a.pack()
    o.pack()
    j.pack()
    u.pack()
    m.pack()
    p.pack()
    t.pack()
    # Création d'un widget Button (bouton Confirmer)
    BoutonCo = Button(Mafenetre, text ='Confirmer', command = Confirmer)
    BoutonCo.pack(side = RIGHT, padx = 5, pady = 5)
    return(f)
    return(a)
    return(j)
    return(m)
    return(t)
```

```
def date(j,m,t):
    for aa in range(0,t):
        if m in [1,3,5,7,8,10] and j==31:
            m+=1
            j=1
        elif m in [4,6,9,11] and j==30:
            m+=1
            j=1
        elif m==2 and j==28:
            m+=1
            j=1
        elif m==12 and j==31:
            j=1
            m=1
        else:
            j+=1
        """print(j,m)"""
    return(j,m)

def date1(j,m,t):
    for bb in range(0,int(t)):
        if m in [1,3,5,7,8,10] and j==31:
            m+=1
            j=1
        elif m in [4,6,9,11] and j==30:
            m+=1
            j=1
        elif m==2 and j==28:
            m+=1
            j=1
        elif m==12 and j==31:
            j=1
            m=1
        else:
            j+=1
        """print(j,m)"""
    return(j)

def date2(j,m,t):
    for oo in range(0,int(t)):
        if m in [1,3,5,7,8,10] and j==31:
            m+=1
            j=1
        elif m in [4,6,9,11] and j==30:
            m+=1
            j=1
        elif m==2 and j==28:
            m+=1
            j=1
        elif m==12 and j==31:
            j=1
```

```
            m=1
        else:
            j+=1
        """print(j,m)"""
    return(m)

def duree_vie(j,m):
    d=0
    if m in [1,2,3]:
        d=random.randrange(35,40,1)*(1/2)
    if m in [4,5,6]:
        d=random.randrange(14,20,1)
    if m in [10,11,12]:
        d=random.randrange(70,90,1)*(1/2)
    if m in [7,8]:
        d=random.randrange(38,46,1)*(1/2)
    if m==9:
        if j<15:
            d=random.randrange(38,46,1)*(1/2)
        else:
            d=random.randrange(70,90,1)*(1/2)
    return(d)

"""def duree_vie(j,m):
    d=0
    if m in [1,2,3]:
        d=random.randrange(70,80,1)
    if m in [4,5,6]:
        d=random.randrange(28,40,1)
    if m in [10,11,12]:
        d=random.randrange(140,180,1)
    if m in [7,8]:
        d=random.randrange(77,92,1)
    if m==9:
        if j<15:
            d=random.randrange(77,92,1)
        else:
            d=random.randrange(140,180,1)
    return(d)"""

def rep_abeilles(j,m,n): #Facteur des naissances
    nbk=0
    if m in [1,2,3]:
        nbk=0
    if m in [4,5,6]:
        nbk=int(0.08*n)*random.randrange(1,2,1)
    if m in [7,8,9]:
        nbk=int(0.045*n)*random.randrange(1,2,1)
    if m in [10,11,12]:
        nbk=int(0.01*n)*random.randrange(1,2,1)
    return(nbk)
```

```
375
376
377 def Confirmer():
378     global l,f,b,a,o,j,u,m,p,t
379     f = int(f.get())
380     a = int(a.get())
381     j = int(j.get())
382     m = int(m.get())
383     t = int(t.get())
384     return(f)
385     return(a)
386     return(j)
387     return(m)
388     return(t)
389     root.destroy
390
391 def Quitter():
392     global Arret
393     print('le nombre de fleurs pollinisées est : ', len(Cp))
394     print('abeilles de départ : ', a )
395     print('abeilles à la fin : ', len(Abeilles) )
396     print('la date de la fin de la simulation est : ',date(j,m,int(k)) )
397     print('la date de début de la simulation est : ', j, '/', m)
398     Arret = True
399     root.destroy()
400     Mafenetre.destroy()
401
402 def Demarrer():
403     """ Démarre l'animation """
404     global Arret
405     Canevas.delete(ALL)
406     if Arret == True:
407         Arret = False
408         Fleur()
409
410
411 Arret = True
412
413 # Création de la fenêtre principale (main window)
414 Mafenetre = Tk()
415 Mafenetre.title('TIPE Pollen')
416
417 # Création d'un widget Canvas
418 LARGEUR = 1000
419 HAUTEUR = 600
420 Canevas = Canvas(Mafenetre, width = LARGEUR, height =HAUTEUR, bg ='green')
421 Canevas.pack(padx = 5, pady = 5)
422
423 # Création d'un widget Button (bouton Démarrer)
```

```
# Création d'un widget Button (bouton Démarrer)
BoutonGo = Button(Mafenetre, text ='Démarrer', command = Demarrer)
BoutonGo.pack(side = LEFT, padx = 10, pady = 10)

# Création d'un widget Button (bouton Paramètres)
BoutonPa = Button(Mafenetre, text ='Paramètres', command = Parametres)
BoutonPa.pack(side = LEFT, padx = 5, pady = 5)

# Création d'un widget Button (bouton Quitter)
BoutonQuitter = Button(Mafenetre, text ='Quitter', command = Quitter )
BoutonQuitter.pack(side = LEFT, padx = 5, pady = 5)

#root
root = Tk()
root.title('Parametres')

#Label Et Entry
l = Label(root, text='Fleurs', width=10)
b = Label(root, text='Abeilles', width=10)
o = Label(root, text='Jour', width=10)
u = Label(root, text='Mois', width=10)
p = Label(root, text='Durée', width=10)
f = Entry(root, width =10)
a = Entry(root, width=10)
j = Entry(root, width=10)
m = Entry(root, width=10)
t = Entry(root, width=10)


Mafenetre.mainloop()
```

```python
def init_liste(n):
    G=[]
    for k in range(n):
        G.append(0)
    return(G)

Linit=init_liste(10)

def map_prob(x,y,G,n):
    H=[]
    for k in range(n):
        H.append(k*500/n)
    c=0
    while (x-500)**2+(y-300)**2>H[c]**2:
        c+=1
    G[c-1]+=1

def Interpolation(W,B):
    Lagrange=[]
    Interpol=Polynomial([0])
    for k in range(0,9):
        Pol=Polynomial(1)
        for i in range(0,9):
            if i!=k:
                Pol=(1/(W[k]-W[i]))*Pol*Polynomial([W[i],-1])
        Lagrange.append(Pol)
    for j in range(0,9):
        Interpol+=B[j]*Lagrange[j]
    return(Interpol.coef)

def barycentre(Fx,Fy,P):
    Pf = 0
    bx = 0
    by = 0
    for i in range(len(Fx)) :
        bx += Fx[i]*P[i]
        by += Fy[i]*P[i]
        Pf += P[i]
    return(bx/Pf,by/Pf)

def centre_radical(x1,x2,x3,y1,y2,y3,R1,R2,R3):
    a = (-(x1-x2)*(x2+x1)-(y1-y2)*(y1+y2)+R1*R1-R2*R2)/2
    b = (-(x1-x3)*(x3+x1)-(y1-y3)*(y1+y3)+R1*R1-R3*R3)/2
    x = (a*(y3-y1)-b*(y2-y1))/((x2-x1)*(y3-y1)-(x3-x1)*(y2-y1))
    y = (b*(x2-x1)-(x3-x1)*a)/((x2-x1)*(y3-y1)-(x3-x1)*(y2-y1))
    return(x,y)
```

```
239 def integrale(xf,yf,Rx,Ry,R):
240     S=0
241     pas=10
242     rho = sqrt((xf-Rx)**2 + (yf-Ry)**2)
243     for k in range(0,9):
244         c=0
245         r = rho - R
246         while r < rho + R:
247             c+=math.acos((r**2 + rho**2 - R**2)/(2*rho*r))*r**(k+1)
248             r+=pas
249         S+=A[k]*c
250     return(2*S)
251 
252 def somme_integrales_fleurs(Rx,Ry,Fx,Fy,Fr):
253     SIF=0
254     for i in range(0,3):
255         SIF+=integrale(Fx[i],Fy[i],Rx,Ry,Fr[i])
256     return(SIF)
257 
258 
259 def position_ruche(Fx,Fy,Fr):
260     pos_ruche=[0,0]
261     som=somme_integrales_fleurs(0,0,Fx,Fy,Fr)
262     for xtest in range(100,301):
263         for ytest in range(100,301):
264             if somme_integrales_fleurs(xtest,ytest,Fx,Fy,Fr)>som:
265                 pos_ruche=[xtest,ytest]
266     return(pos_ruche[0],pos_ruche[1])
267 
```