

ANNEXE A : 1^{ER} Modèle simple

```
1 from random import *
2 from numpy import *
3 from matplotlib.pyplot import *
4 from numpy import linalg as LA
5 #constantes : lambda1 (paramètre de la Fatt) et alpha (paramètre de la Frep)
6 def MODELE_SIMPLE(lambda1,alpha):
7     #Nombre d'itération
8     n=600
9     #Nombre de Boîds
10    nbr=2
11    #Position initiale
12    A0=array([-10,3])
13    B0=array([10,0])
14    #initialisation des matrices de stockage de position
15    A=zeros((2,n))
16    B=zeros((2,n))
17    #initialisation de la vitesse initiale aléatoire
18    j1=random.random()
19    j2=random.random()
20    V=array([[cos(j1*2*pi),cos(j2*2*pi)],[sin(j1*2*pi),sin(j2*2*pi)]])
21    #Affectation des positions initiales
22    A[:,0]=A0
23    B[:,0]=B0
24    #Boucle de calcul des positions successives
25    for i in range(n-1):
26        V[:,0]=V[:,0]+(lambda1-(4*alpha)/LA.norm(A[:,i]-B[:,i])**2)*(B[:,i]-A[:,i])/2
27        V[:,1]=V[:,1]+(lambda1-(4*alpha)/LA.norm(A[:,i]-B[:,i])**2)*(A[:,i]-B[:,i])/2
28        A[:,i+1]=A[:,i]+V[:,0]
29        B[:,i+1]=B[:,i]+V[:,1]
30
31    #Affichage direct des trajectoires
32    plot(A[0,:],A[1:], 'r')
33    plot(B[0:],B[1:], "b")
34    show()
```

Annexe B : Décroissance de vitesse

```
1 from random import *
2 from numpy import *
3 from matplotlib.pyplot import *
4 from numpy import linalg as LA
5 #Coefficient de la décroissance des vitesses noté par K >1
6 def DECROISSANCE_VITESSE(lambda1,alpha,K):
7     #Nombre d'itération
8     n=600
9     #Nombre de Boids
10    nbr=2
11    #Position initiale
12    A0=array([-10,3])
13    B0=array([10,0])
14    #initialisation des matrices de stockage de position
15    A=zeros((2,n))
16    B=zeros((2,n))
17    #initialisation de la vitesse initiale aléatoire
18    j1=random.random()
19    j2=random.random()
20    V=array([[cos(j1*2*pi),cos(j2*2*pi)],[sin(j1*2*pi),sin(j2*2*pi)]])
21    #Affectation des positions initiales
22    A[:,0]=A0
23    B[:,0]=B0
24    #Boucle de calcul des positions successives
25    for i in range(n-1):
26        V[:,0]=V[:,0]/K+(lambda1-(4*alpha)/LA.norm(A[:,i]-B[:,i])**2)*(B[:,i]-A[:,i])/2
27        V[:,1]=V[:,1]/K+(lambda1-(4*alpha)/LA.norm(A[:,i]-B[:,i])**2)*(A[:,i]-B[:,i])/2
28        A[:,i+1]=A[:,i]+V[:,0]
29        B[:,i+1]=B[:,i]+V[:,1]
30
31    #Affichage direct des trajectoires
32    plot(A[0,:],A[1:], 'r')
33    plot(B[0:],B[1:], "b")
34    show()
```

Annexe C : Vitesse Limite

```
1 from random import *
2 from numpy import *
3 from matplotlib.pyplot import *
4 from numpy import linalg as LA
5 def VITESSE_LIMITE(lambda1,lambda2,K,Vlim):
6     #Nombre d'itération
7     n=600
8     #Nombre de Boids
9     nbr=2
10    #Position initiale
11    A0=array([-10,3])
12    B0=array([10,0])
13    #initialisation des matrices de stockage de position
14    A=zeros((2,n))
15    B=zeros((2,n))
16    #initialisation de la vitesse initiale aléatoire
17    j1=random.random()
18    j2=random.random()
19    V=array([[cos(j1*2*pi),cos(j2*2*pi)],[sin(j1*2*pi),sin(j2*2*pi)]])
20    #Affectation des positions initiales
21    A[:,0]=A0
22    B[:,0]=B0
23    #direction de la vitesse limite
24    Ulim=Vlim/LA.norm(Vlim)
25    #Boucle de calcul des positions succesives
26    for i in range(n-1):
27        if LA.norm(V[:,0])> LA.norm(Vlim) or LA.norm(V[:,1])> LA.norm(Vlim):
28            V[:,0]=V[:,0]/K+(lambda1-(4*alpha)/LA.norm(A[:,i]-B[:,i])**2)*(B[:,i]-A[:,i])/2
29            V[:,1]=V[:,1]/K+(lambda1-(4*alpha)/LA.norm(A[:,i]-B[:,i])**2)*(A[:,i]-B[:,i])/2
30            A[:,i+1]=A[:,i]+V[:,0]
31            B[:,i+1]=B[:,i]+V[:,1]
32        elif LA.norm(V[:,0])<= LA.norm(Vlim) or LA.norm(V[:,1])<= LA.norm(Vlim):
33            V[:,0]=V[:,0]/K+(lambda1-(4*alpha)/LA.norm(A[:,i]-B[:,i])**2)*(B[:,i]-A[:,i])/2+[LA.norm(Vlim)-LA.norm(V[:,0])]*Ulim
34            V[:,1]=V[:,1]/K+(lambda1-(4*alpha)/LA.norm(A[:,i]-B[:,i])**2)*(A[:,i]-B[:,i])/2 + [LA.norm(Vlim)-LA.norm(V[:,0])]*Ulim
35            A[:,i+1]=A[:,i]+V[:,0]
36            B[:,i+1]=B[:,i]+V[:,1]
37    #Affichage direct des trajectoires
38    plot(A[0,:],A[1,:],'r')
39    plot(B[0,:],B[1,:],"b")
40    show()
```

Annexe D : Généralisation des forces

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from math import *
4 def FattrGaussA(lambda1,lambda2,R0):
5     t=np.linspace(0,10,1000)
6     Fattr = lambda1*np.exp(-(((t/2)-R0/2)/lambda2)**2)
7     plt.plot(t,Fattr)
8     plt.xlabel("distance")
9     plt.ylabel("intensité de la force")
10    plt.show()
```

```

1 from random import *
2 from matplotlib.pyplot import *
3 from numpy import linalg as LA
4 import numpy as np
5 def GENERALISATION_FORCES(lambda1,lambda2,alpha,R0,K,Vlim):
6     n=600
7     nbr=2
8     A0=array([-10,3])
9     B0=array([10,0])
10    A=zeros((2,n))
11    B=zeros((2,n))
12    j1=random.random()
13    j2=random.random()
14    V=array([[cos(j1*2*pi),cos(j2*2*pi)],[sin(j1*2*pi),sin(j2*2*pi)]])
15    Ulim=Vlim/LA.norm(Vlim)
16    A[:,0]=A0
17    B[:,0]=B0
18    for i in range(n-1):
19        distAB=(B[:,i]-A[:,i])/2
20        distBA=(A[:,i]-B[:,i])/2
21        FattrAB=lambda1*np.exp(-((LA.norm(distAB)-R0/2)/lambda2)**2)*distAB/LA.norm(distAB)#Force d'attraction en A par B
22        FattrBA=lambda1*np.exp(-((LA.norm(distBA)-R0/2)/lambda2)**2)*distBA/LA.norm(distBA)#Force d'attraction en B par A
23        #on annule la répulsion pour une certaines distance
24        if LA.norm(distAB)<10 :
25            FrepAB=-(alpha/LA.norm(distAB)**2)*distAB
26        else :
27            FrepAB=0
28        if LA.norm(distBA)<10 :
29            FrepBA=-(alpha/LA.norm(distBA)**2)*distBA
30        else:
31            FrepBA=0
32        V[:,0]=V[:,0]/K+FattrAB+FrepAB
33        V[:,1]=V[:,1]/K+FattrBA+FrepBA
34        if LA.norm(V[:,0])> LA.norm(Vlim) or LA.norm(V[:,1])> LA.norm(Vlim):
35            V[:,0]=V[:,0]/K+FattrAB+ FrepAB
36            V[:,1]=V[:,1]/K+FattrBA+ FrepBA
37            A[:,i+1]=A[:,i]+V[:,0]
38            B[:,i+1]=B[:,i]+V[:,1]
39        elif LA.norm(V[:,0])<= LA.norm(Vlim) or LA.norm(V[:,1])<= LA.norm(Vlim):
40            V[:,0]=V[:,0]/K+FattrAB+ FrepAB + [LA.norm(Vlim)-LA.norm(V[:,0])]*Ulim
41            V[:,1]=V[:,1]/K+FattrBA+ FrepBA + [LA.norm(Vlim)-LA.norm(V[:,1])]*Ulim
42            A[:,i+1]=A[:,i]+V[:,0]
43            B[:,i+1]=B[:,i]+V[:,1]
44        A[:,i+1]=A[:,i]+V[:,0]
45        B[:,i+1]=B[:,i]+V[:,1]
46    plot(A[0,:],A[1,:],"r")
47    plot(B[0,:],B[1,:],"b")
48    show()

```

Annexe E : Force extérieur

```

1 from random import *
2 from matplotlib.pyplot import *
3 from numpy import linalg as LA
4 import numpy as np
5 #constantes : lambdap( paramètre de la Force (poteau) en position xp,yp
6 def FORCES_POTEAU(lambda1,lambda2,alpha,R0,K,Vlim,lambdap,xp,yp):
7     n=600
8     nbr=2
9     A0=array([-10,3])
10    B0=array([10,0])
11    A=zeros((2,n))
12    B=zeros((2,n))
13    j1=random()
14    j2=random()
15    V=array([[cos(j1*2*pi),cos(j2*2*pi)],[sin(j1*2*pi),sin(j2*2*pi)]])
16    Ulim=Vlim/LA.norm(Vlim)
17    A[:,0]=A0
18    B[:,0]=B0
19    for i in range(n-1):
20        distAB=(B[:,i]-A[:,i])/2
21        distBA=(A[:,i]-B[:,i])/2
22        FattrAB=lambda1*np.exp(-((LA.norm(distAB)-R0/2)/lambda2)**2)*distAB/LA.norm(distAB)#Force d'attraction en A par B
23        FattrBA=lambda1*np.exp(-((LA.norm(distBA)-R0/2)/lambda2)**2)*distBA/LA.norm(distBA)#Force d'attraction en B par A
24        P0=array([xp,yp])
25        distAP=(P0-A[:,i])
26        distBP=(P0-B[:,i])
27        #On annule la répulsion pour une certaines distance
28        if LA.norm(distAB)<10 :
29            FrepAB=-(alpha/LA.norm(distAB)**2)*distAB
30        else :
31            FrepAB=0
32        if LA.norm(distBA)<10 :
33            FrepBA=-(alpha/LA.norm(distBA)**2)*distBA
34        else:
35            FrepBA=0
36        #On annule l'influence de l'obstacle pour une certaines distance
37        if LA.norm(distAP)<10 :
38            poteauAP=-(lambdap/LA.norm(distAP)**2)*distAP
39        else :
40            poteauAP=0
41        if LA.norm(distBP)<10 :
42            poteauBP=-(lambdap/LA.norm(distBP)**2)*distBP
43        else:
44            poteauBP=0
45        V[:,0]=V[:,0]/K+FattrAB+FrepAB +poteauAP
46        V[:,1]=V[:,1]/K+FattrBA+FrepBA +poteauBP
47        if LA.norm(V[:,0])> LA.norm(Vlim) or LA.norm(V[:,1])> LA.norm(Vlim):
48            V[:,0]=V[:,0]/K+ FattrAB + FrepAB + poteauAP
49            V[:,1]=V[:,1]/K+ FattrBA + FrepBA + poteauBP
50            A[:,i+1]=A[:,i]+V[:,0]
51            B[:,i+1]=B[:,i]+V[:,1]
52        elif LA.norm(V[:,0])<= LA.norm(Vlim) or LA.norm(V[:,1])<= LA.norm(Vlim):
53            V[:,0]=V[:,0]/K+FattrAB+ FrepAB + [LA.norm(Vlim)-LA.norm(V[:,0])]*Ulim
54            V[:,1]=V[:,1]/K+FattrBA+ FrepBA + [LA.norm(Vlim)-LA.norm(V[:,1])]*Ulim
55            A[:,i+1]=A[:,i]+V[:,0]
56            B[:,i+1]=B[:,i]+V[:,1]
57    plot(A[0,:],A[1,:],"r")
58    plot(B[0,:],B[1,:],"b")
59    plot(xp,yp,"v")
60    show()

```

Annexe F : Modèle général

```

1 from matplotlib.pyplot import *
2 from numpy import linalg as LA
3 import numpy as np
4 from numpy import *
5 def MODELE_GENERAL(n,nbr,lambda1,lambda2,alpha,R0,K,Vlim,lambdap,yp,yp): #nbr = nombre de boids et n =nombre d'itération
6     #Conditions initiales
7     Boids0=20*np.random.rand(2,nbr)
8     Vboids0=0.2*np.random.rand(2,nbr)
9     #Initialisation
10    Boids=zeros((2,n,nbr))
11    Vboids=zeros((2,nbr))
12    Fatt=zeros((2,nbr))
13    Frep=zeros((2,nbr))
14    Boids[:,0,:]=Boids0
15    Vboids=Vboids0
16    P0=array([xp,yp])#position du poteau
17    Vinfty=array([1,Vlim])#Vlim vitesse à partir de laquelle on tend vers Vinfty
18    for i in range (n-1) : # i nombre de tour
19        Fatt=zeros((2,nbr)) #on remet à zéro à chaque tour
20        Frep=zeros((2,nbr))
21        for j in range (nbr):# j numéro de boids
22            k=0
23            while (k<nbr):
24                if(k!=j): # on cumule les interactions pour chaque boids
25                    dist=Boids[:,i,k]-Boids[:,i,j]
26                    norme=LA.norm(dist)
27                    Fatt[:,j]=Fatt[:,j]+lambda1*np.exp(-((norme-R0/2)/lambda2)**2)*(dist/norme)
28                    Frep[:,j]=Frep[:,j]-(alpha/norme)*(dist/norme)
29                k=k+1
30            distp=P0-Boids[:,i,j]
31            if LA.norm(distp)<5:
32                FpoteaujP=-(lambdap/LA.norm(distp)**2)*distp
33            else :
34                FpoteaujP =0
35            Vboids[:,j]=Vboids[:,j]/K+Frep[:,j]+Fatt[:,j]+ FpoteaujP #mise en place de la vitesse
36            if(LA.norm(Vboids[:,j])<Vlim): #recherche d'une vitesse inférieure à vlim
37                Vboids[:,j]=Vboids[:,j]+[Vlim-LA.norm(Vboids[:,j])]*Vinfty
38            Boids[:,i+1,j]=Boids[:,i,j]+Vboids[:,j]#calcul de la position
39            plot(Boids[0,0 :i,j],Boids[1,0 :i,j],linewidth=0.1) #affichage des boids
40    if lambdap!=0 :
41        plot(xp,yp,"v")
42    show()

```

Annexe G : Modèle 3D

```

1 from mpl_toolkits.mplot3d import Axes3D
2 from matplotlib.pyplot import *
3 from numpy import linalg as LA
4 import numpy as np
5 from numpy import *
6 def MODELE_3D(n,nbr,lambda1,lambda2,alpha,R0,K,Vlim,lambdap,xp,yp,zp): #nbr = nombre de boids et n =nombre d'itération
7     #Conditions initiales
8     Boids0=20*np.random.rand(3,nbr)
9     Vboids0=0.2*np.random.rand(3,nbr)
10    #Initialisation
11    Boids=zeros((3,n,nbr))
12    Vboids=zeros((3,nbr))
13    Fatt=zeros((3,nbr))
14    Frep=zeros((3,nbr))
15    Boids[:,0,:]=Boids0
16    Vboids=Vboids0
17    P0=array([xp,yp,zp])#position du poteau
18    Vinfty=array([0,0,Vlim])#Vlim vitesse à partir de laquelle on tend vers Vinfty
19    for i in range (n-1) : # i nombre de tour
20        Fatt=zeros((3,nbr)) #on remet à zéro à chaque tour
21        Frep=zeros((3,nbr))
22        theta1=random.rand()
23        phi1=random.rand()
24        if(random.rand()>0.5):# ajout d'une vitesse de direction aléatoire
25            Vinfty=Vinfty+array([sin(pi*theta1)*cos(2*pi*phi1),sin(pi*theta1)*cos(2*pi*phi1),cos(pi*theta1)])/2
26            Vinfty=Vinfty/LA.norm(Vinfty)
27        for j in range (nbr):# j numéro de boids
28            k=0
29            while (k<nbr):
30                if(k!=j): # on cumule les interactions pour chaque boids
31                    dist=Boids[:,i,k]-Boids[:,i,j]
32                    norme=LA.norm(dist)
33                    Fatt[:,j]=Fatt[:,j]+lambda1*np.exp(-((norme-R0/2)/lambda2)**2)*(dist/norme)
34                    Frep[:,j]=Frep[:,j]-(alpha/norme)*(dist/norme)
35                    k=k+1
36                dist=P0-Boids[:,i,j]
37                if LA.norm(dist)<5:
38                    FpoteaujP=-(lambdap/LA.norm(dist)**2)*dist
39                else :
40                    FpoteaujP =0
41                Vboids[:,j]=Vboids[:,j]/K+Frep[:,j]+Fatt[:,j]+ FpoteaujP #mise en place de la vitesse
42                if(LA.norm(Vboids[:,j])<Vlim):#recherche d'une vitesse inférieure à Vlim
43                    theta2=random.rand()
44                    phi2=random.rand()
45                    if(random.rand()>0.5):
46                        Vboids[:,j]=Vboids[:,j]+array([Vlim-LA.norm(Vboids[:,j]))*array([sin(pi*theta2)*cos(2*pi*phi2),sin(pi*theta2)*cos(2*pi*phi2),cos(pi*theta2)])))/2
47                    else:
48                        Vboids[:,j]=Vboids[:,j]+[Vlim-LA.norm(Vboids[:,j]))*Vinfty
49                Boids[:,i+1,j]=Boids[:,i,j]+Vboids[:,j]#calcul de la position
50                gca(projection='3d').plot(Boids[0,0:i,j],Boids[1,0:i,j],Boids[2,0:i,j], "--",linewidth=0.5)
51            if lambdap!=0 :
52                gca(projection='3d').plot([xp],[yp],[zp],"v")
53            show()

```