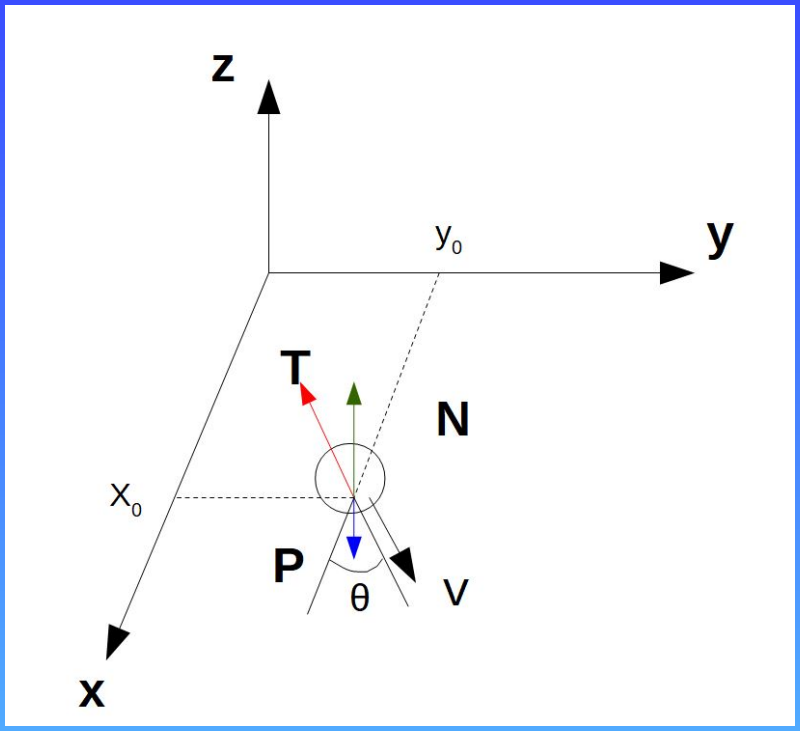
Mouvement d'une balle dans un baby-foot

Sommaire

I. Étude mécanique



$$\vec{N} = N \,.\, \vec{z}$$

$$\vec{P} = -P \,.\, \vec{z}$$

$$\vec{T} = -T \,.\, \cos(\theta) \,.\, \vec{x} - T \,.\, \sin(\theta) \,.\, \vec{y}$$

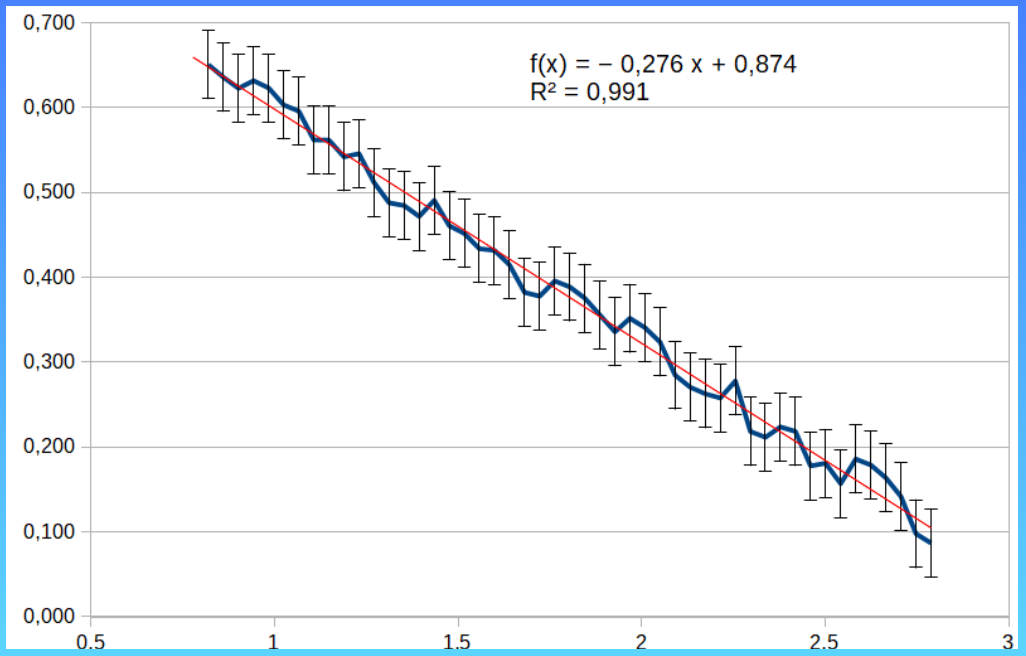
$$\vec{v} = v \,.\, \cos(\theta) \,.\, \vec{x} + v \,.\, \sin(\theta) \,.\, \vec{y}$$

$$x(t) = -\left(\frac{1}{2}\right) fg\cos(\theta) t^2 + v\cos(\theta) t + x_0$$

$$y(t) = -\left(\frac{1}{2}\right) fg\sin(\theta) t^2 + v\sin(\theta) t + y_0$$

I. Étude mécanique

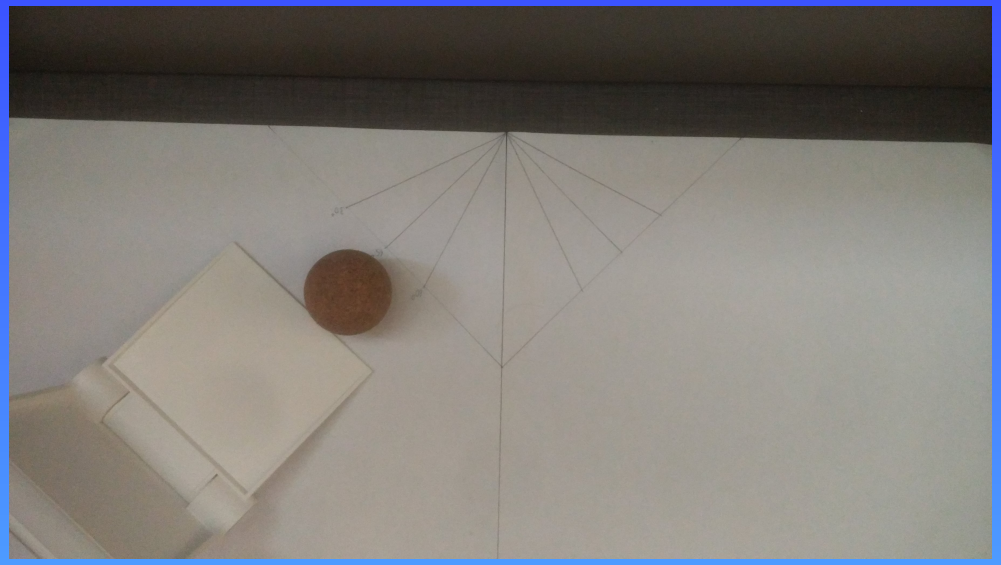
Détermination du frottement entre la balle et le tapis de billard



Ainsi :

	Coefficient de frottement
Balle/Béton	0,01
Balle/Bois de Baby	0,01
Balle/Tapis billard	0,03

I. Étude mécanique

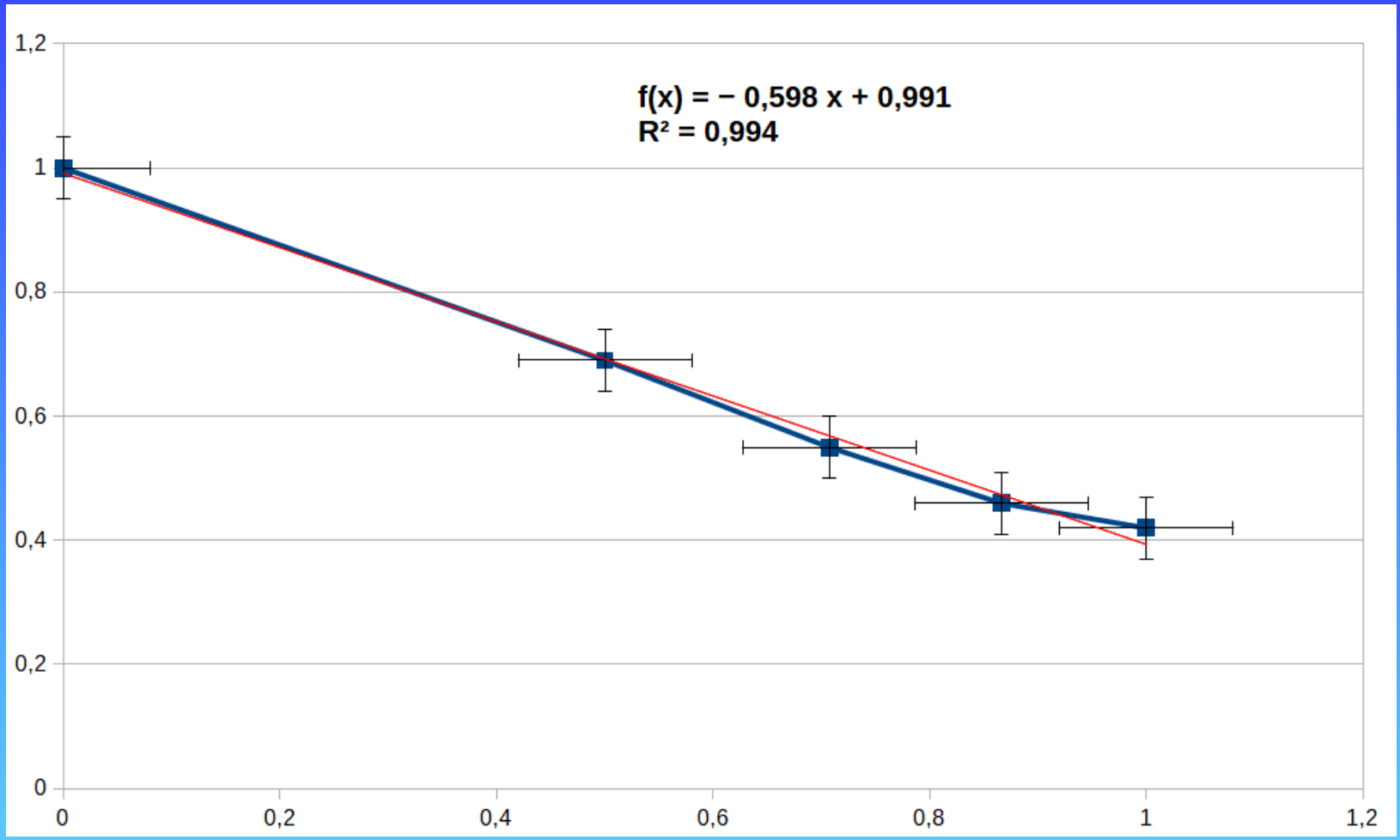


Coefficient de restitution :

$$e = \frac{Vitesse\ après\ collision}{Vitesse\ avant\ collision}$$

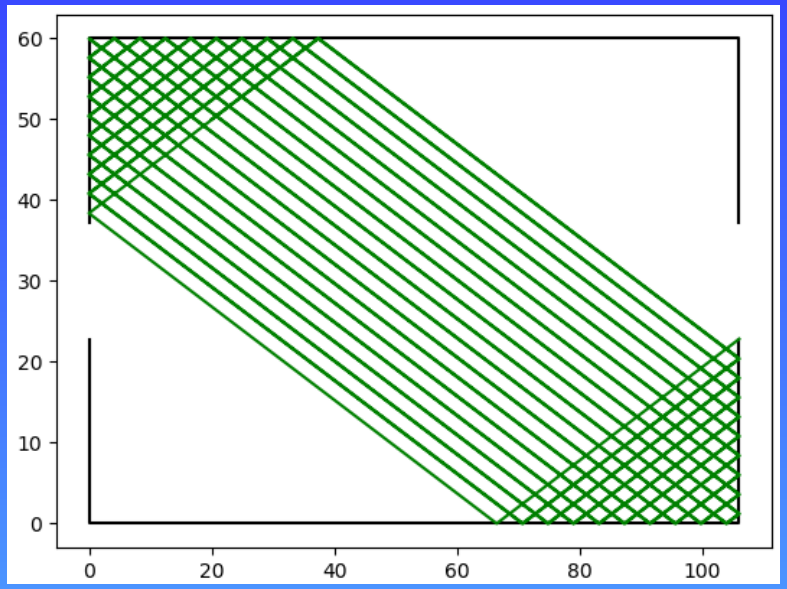
Angle (en radians)	0	pi/6	pi/4	pi/3	pi/2
e	1	0,62	0,55	0,49	0,42

I. Étude mécanique

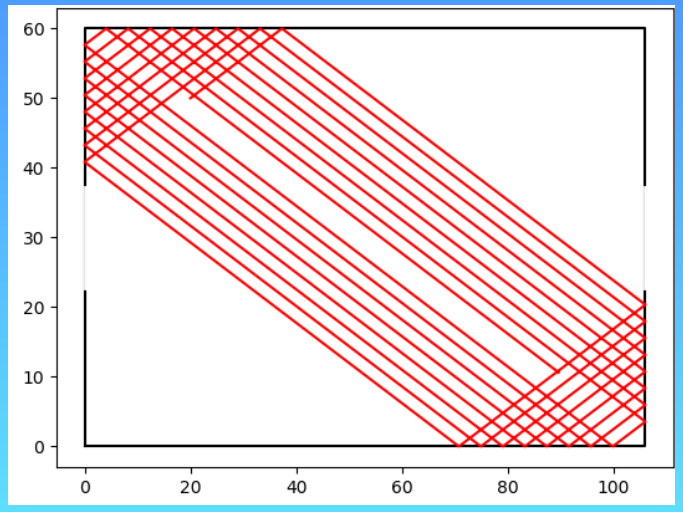


Coefficient de restitution en fonction de l'angle d'impact θ

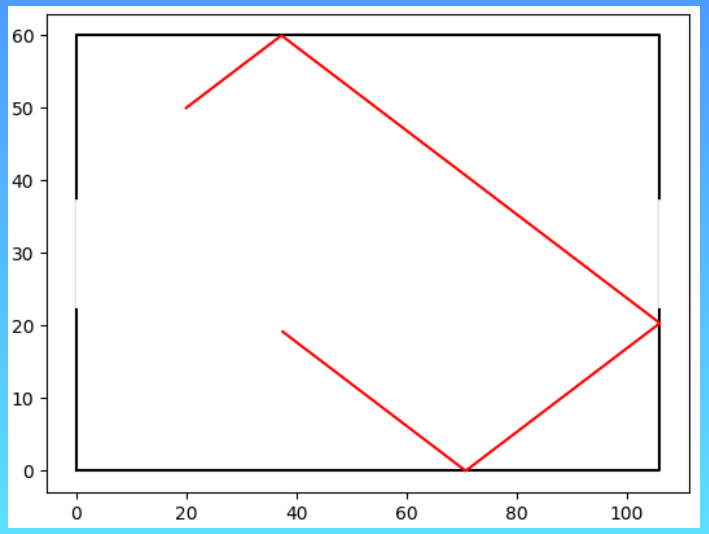
I. Étude mécanique



Simulation sans frottement et e=1



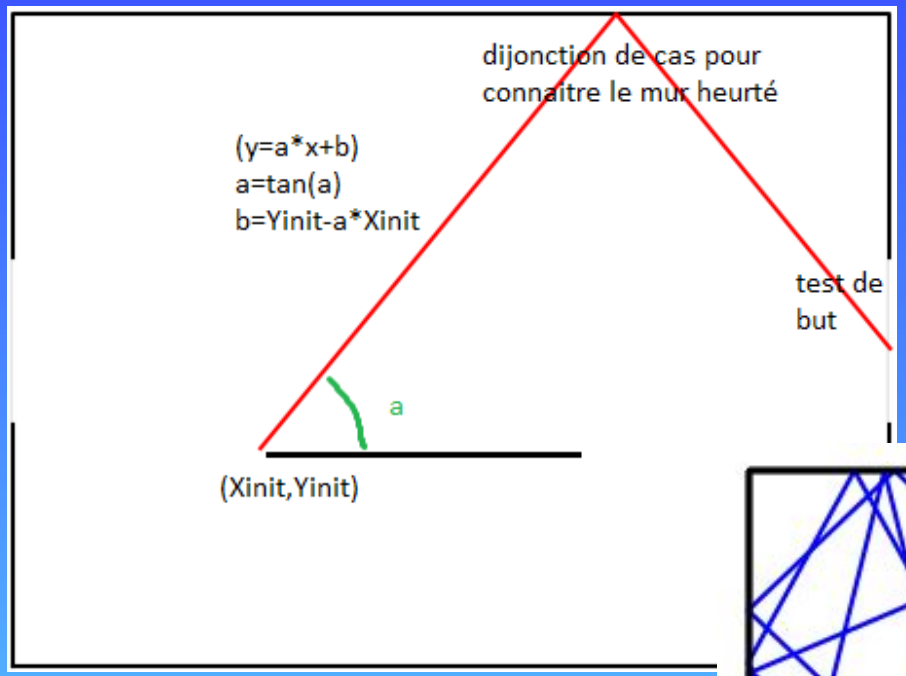
Simulation avec f=0,02 et e=1

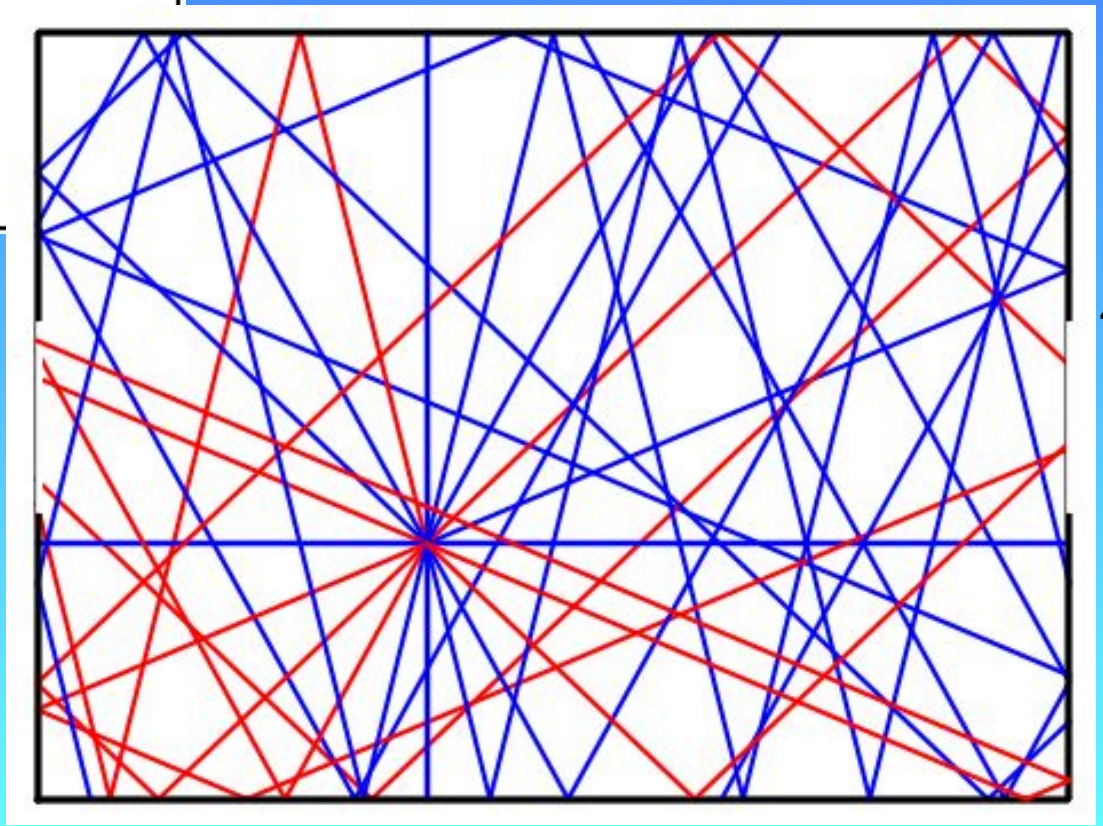


Simulation avec f=0,02 et en prenant compte des chocs

II. Étude statistiques

Principe du lancer de rayon

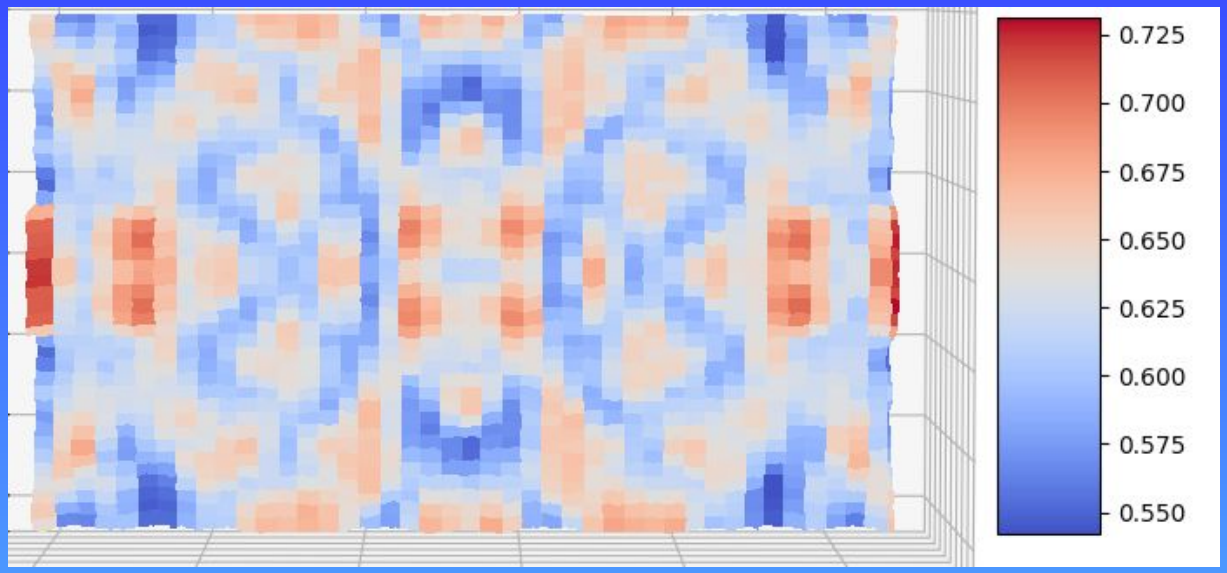




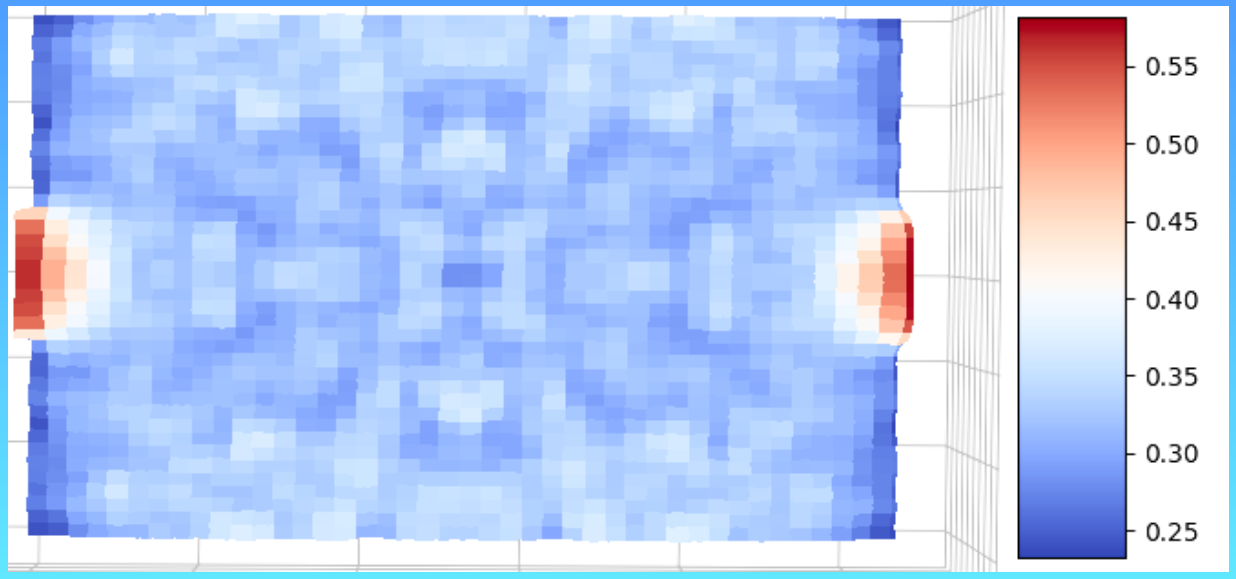
Lancer de
40 rayons

II. Étude statistiques

Approximation sans joueurs



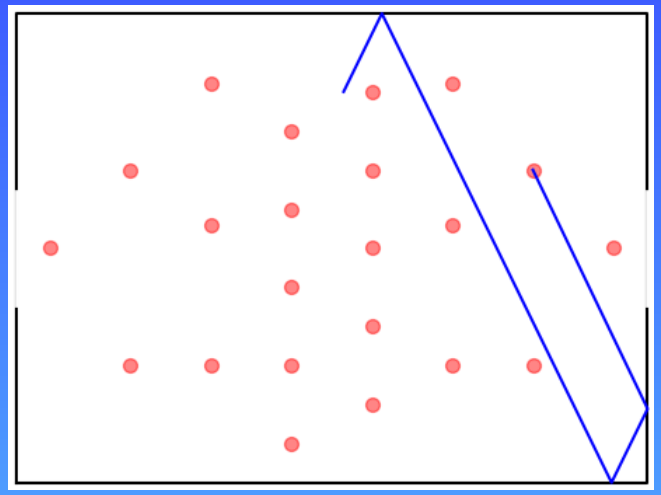
-120x207 points
-Arrêt à 10 rebonds
-Lancer de 100 rayons

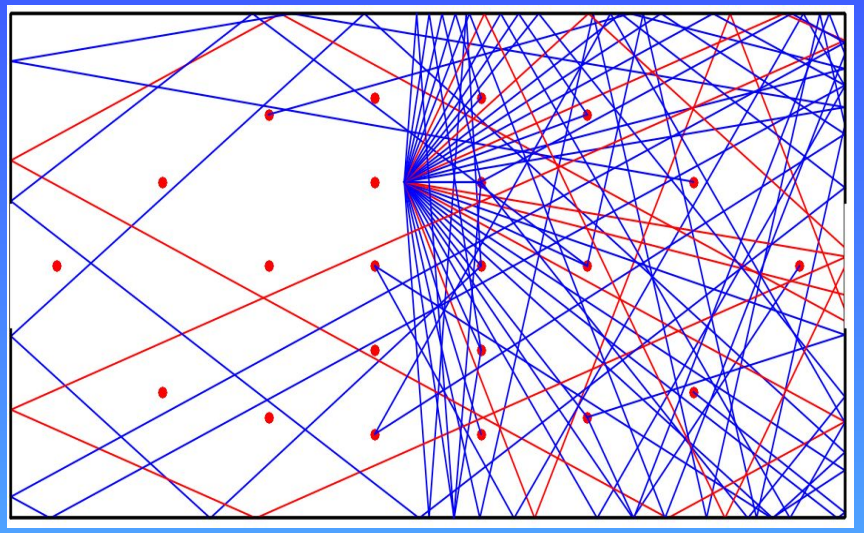


-120x207 points
-Arrêt à 3 rebonds
-Lancer de 100 rayons

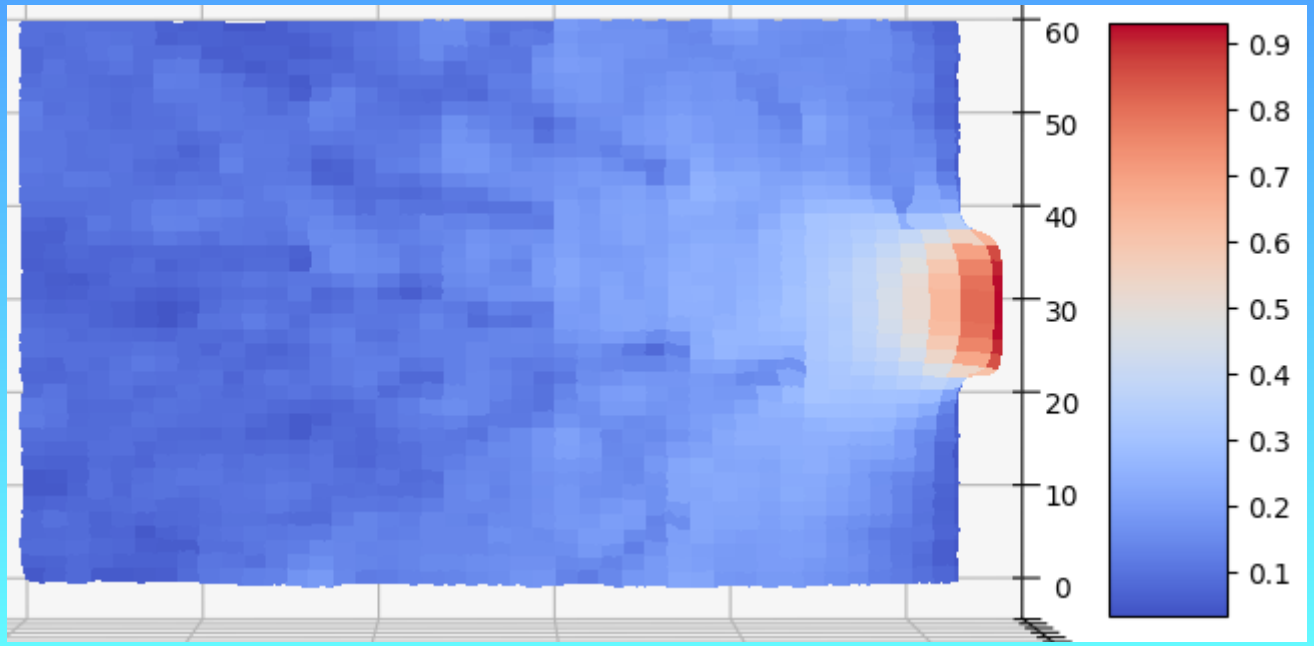
II. Étude statistiques

Approximation avec joueurs





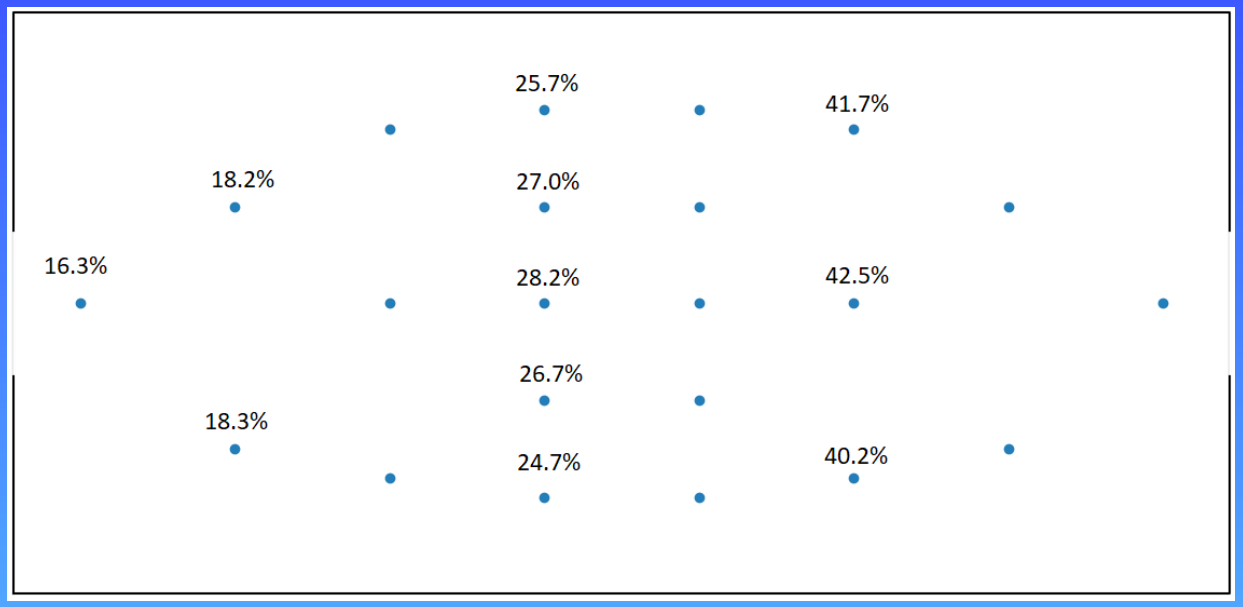
Lancer de
40 rayons



Lancer de 100 rayons sur
120x207 pts
3 rebonds max

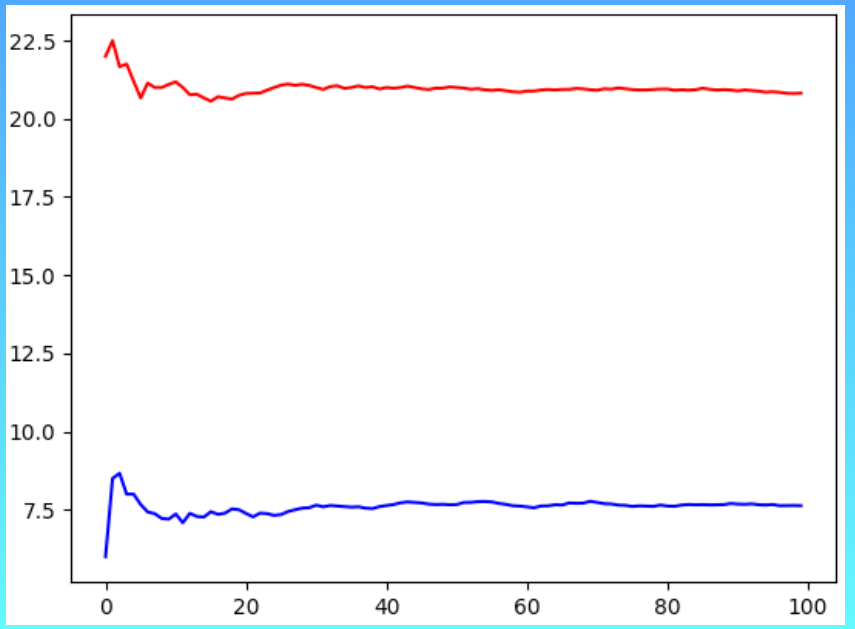
II. Étude statistiques

Résultat pour chaque joueur

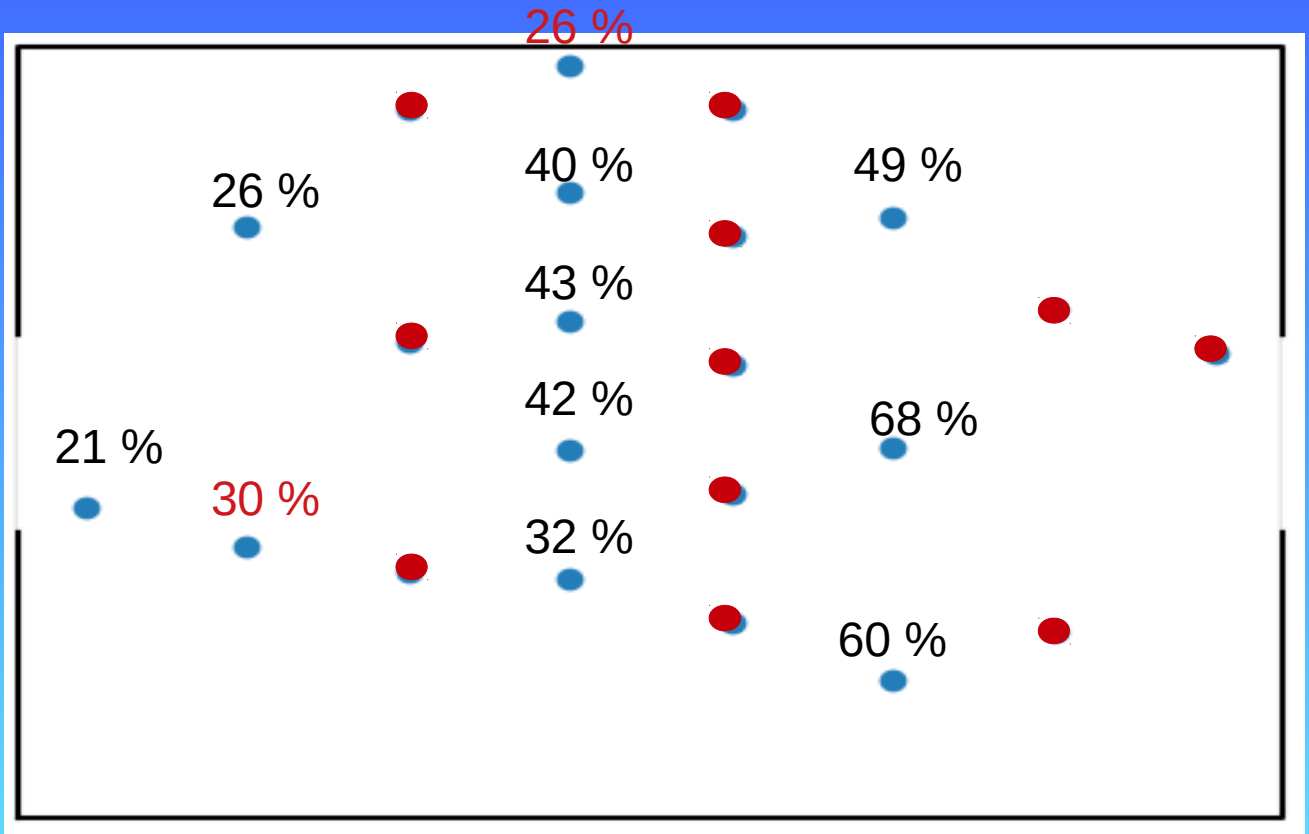


Lancer de 200 rayons pour 1000 positions de joueurs différentes

En rouge : attaquant centre
En bleu : goal



Conclusion



Annexes

Partie Mécanique

```
from random import *
from math import *
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np


largeurbut=14.5
xmin=0
xmax=106
ymin=0
ymax=60
entrebar=13.5
entredef=25
entreatt=18
entremil=10
largeurjoueur=1
g=981
f=0.01



def rebond(vf,xf,yf,angf,bord):

    if (bord=="haut" and cos(angf)>0)or (bord=="bas" and cos(angf)<0):
        tetavoulu=angf%(pi)
        angi=-angf
    elif (bord=="haut" and cos(angf)<0)or (bord=="bas" and cos(angf)>0):
        tetavoulu=pi-(angf%(pi))
        angi=-angf
    elif bord=="droite" or bord=="gauche":
        tetavoulu=(pi/2)- (abs(angf)%(pi))
        angi=pi-angf
    vi=vf*(-0.598*tetavoulu+1)
    return(vi,xf,yf,angi)
```

```
def choc(vi,xi,yi,angi):
    global largeurbut
    global xmin
    global xmax
    global ymin
    global ymax
    global g
    global f
    affichage=True
    but=False
    dparcouru=0
    fction=[tan(angi),yi-xi*tan(angi)]
    X=[xi]
    Y=[yi]
    #droite
    if xmax*fction[0]+fction[1] < ymax and xmax*fction[0]+fction[1] > ymin and cos(angi) > 0 :
        mur="droite"
        ptface=[xmax,fction[0]*xmax+fction[1]]
    #gauche
    elif xmin*fction[0]+fction[1] < ymax and xmin*fction[0]+fction[1] > ymin and cos(angi)<0 :
        mur="gauche"
        ptface=[xmin,fction[0]*xmin+fction[1]]
    #haut
    elif fction[0]!=0 and (ymax-fction[1])/fction[0] >xmin and (ymax-fction[1])/fction[0] < xmax and sin(angi)>0:
        mur="haut"
        ptface=[(ymax-fction[1])/fction[0],ymax]
    #bas
    elif fction[0]!=0 and (ymin-fction[1])/fction[0] >xmin and (ymin-fction[1])/fction[0] <xmax and sin(angi)<0:
        mur="bas"
        ptface=[(ymin-fction[1])/fction[0],ymin]
    xf=ptface[0]
    yf=ptface[1]
    dmax=(vi**2)/(2*f*g)
    atteindbord= dmax> sqrt(((yf-yi)**2)+(xf-xi)**2)
    d=min(dmax,sqrt(((yf-yi)**2)+(xf-xi)**2))
    dparcouru=dparcouru+d
    vf=sqrt((vi**2) - 2*d*f*g)

    while atteindbord :
        if affichage :
            X.append(xf)
            Y.append(yf)

        vi,xi,yi,angi=rebond(vf,xf,yf,angi,mur)

        fction=[tan(angi),yi-xi*tan(angi)]

        if xmax*fction[0]+fction[1] < ymax and xmax*fction[0]+fction[1] > ymin and cos(angi) > 0 :
            mur="droite"
            ptface=[xmax,fction[0]*xmax+fction[1]]
        #gauche
        elif xmin*fction[0]+fction[1] < ymax and xmin*fction[0]+fction[1] > ymin and cos(angi)<0 :
            mur="gauche"
            ptface=[xmin,fction[0]*xmin+fction[1]]
        #haut
        elif fction[0]!=0 and (ymax-fction[1])/fction[0] >xmin and (ymax-fction[1])/fction[0] < xmax and sin(angi)>0:
            mur="haut"
            ptface=[(ymax-fction[1])/fction[0],ymax]
        #bas
        elif fction[0]!=0 and (ymin-fction[1])/fction[0] >xmin and (ymin-fction[1])/fction[0] <xmax and sin(angi)<0:
            mur="bas"
            ptface=[(ymin-fction[1])/fction[0],ymin]
```

```
    xf=ptface[0]
    yf=ptface[1]
    dmax=(vi**2)/(2*f*g)
    atteindbord= dmax> sqrt(((yf-yi)**2)+(xf-xi)**2)
    d=min(dmax,sqrt(((yf-yi)**2)+(xf-xi)**2))
    dparcouru=dparcouru+d
    #test but
    if (mur=='droite' or mur=='gauche') and yf>(ymax/2)-(largeurbut/2) and  yf<(ymax/2)+(largeurbut/2) and atteindbord :
        X.append(xf)
        Y.append(yf)
        but=True
        break


    vf=sqrt(abs((vi**2) - 2*d*f*g))

if but :
    xarret=xf
    yarret=yf
    if affichage :
            plt.plot([0,106,106,0,0],[0,0,60,60,0],c='black')
            plt.plot([0,0],[22.75,37.25],'w-',lw=2)
            plt.plot([106,106],[22.75,37.25],'w-',lw=2)
            plt.plot(X,Y,color='green')
            plt.show()
            return(xarret,yarret,dparcouru,but)
```

```
else :
xarret=dmax*cos(angi)+xi
yarret=dmax*sin(angi)+yi
X.append(xarret)
Y.append(yarret)
if affichage :
        plt.plot([0,106,106,0,0],[0,0,60,60,0],c='black')
        plt.plot([0,0],[22.75,37.25],'w-',lw=2)
        plt.plot([106,106],[22.75,37.25],'w-',lw=2)
        plt.plot(X,Y,color='red')
        plt.show()
        return(xarret,yarret,dparcouru,but)
```

Partie Statistiques

```
positionbars=[]
a=random()
a=a*30+15
positionbars.append(a)
a=random()
a=a*35
positionbars.append(a)
a=random()
a=a*24
positionbars.append(a)
a=random()
a=a*20
positionbars.append(a)
a=random()
a=a*20
positionbars.append(a)
a=random()
a=a*24
positionbars.append(a)
a=random()
a=a*35
positionbars.append(a)
a=random()
a=a*30+15
positionbars.append(a)


premierebar=5.75
matricejoueurs=[]
matricejoueurs.append([premierebar,positionbars[0],"gauche"])
for i in range (0,2):
    matricejoueurs.append([premierebar + entrebar,positionbars[1]+i*entredef,"gauche"])
for i in range (0,3):
    matricejoueurs.append([premierebar + entrebar*2,positionbars[2]+i*entreatt,"droite"])
for i in range (0,5):
    matricejoueurs.append([premierebar+entrebar*3,positionbars[3]+i*entremil,"gauche"])
for i in range (0,5):
    matricejoueurs.append([premierebar+entrebar*4,positionbars[4]+i*entremil,"droite"])
for i in range (0,3):
    matricejoueurs.append([premierebar+entrebar*5,positionbars[5]+i*entreatt,"gauche"])
for i in range (0,2):
    matricejoueurs.append([premierebar+entrebar*6,positionbars[6]+i*entredef,"droite"])
matricejoueurs.append([premierebar+entrebar*7,positionbars[7],"droite"])
```

```
def pointderebond (x,y,angle):
    global largeurbut
    global xmin
    global xmax
    global ymin
    global ymax
    global matricejoueurs
    global largeurjoueur
    fction=[tan(angle),y-tan(angle)*x]
    X,Y=x,y
    #test rencontre joueur
    for i in range (len(matricejoueurs)):
        x=matricejoueurs[i][0]
        y=matricejoueurs[i][1]
        if fction[0]*x+fction[1] <= y + largeurjoueur and fction[0]*x+fction[1] >= y - largeurjoueur and (X-x)*cos(angle)<=0 :
            return([x,y,0,"arreté"])
        if fction[0]==0 and fction[1]<= x + largeurjoueur and fction[1]>= x - largeurjoueur:
            return([x,y,0,"arreté"])
        if fction[0]!=0 :
            if (y-fction[1])/fction[0] <= x + largeurjoueur and (y-fction[1])/fction[0] >= x - largeurjoueur and (X-x)*cos(angle)<=0 :
                return([x,y,0,"arreté"])



    #test but
    #but à droite
    if xmax*fction[0]+fction[1]<(ymax/2)+(largeurbut/2) and xmax*fction[0]+fction[1]>(ymax/2)-(largeurbut/2) and cos(angle)>0:

        return([xmax,xmax*fction[0]+fction[1],0,True])

    #la balle tape le bord de droite
    elif xmax*fction[0]+fction[1] < ymax and xmax*fction[0]+fction[1] > ymin and cos(angle) > 0 :
        return([xmax,xmax*fction[0]+fction[1],pi-angle,False])
    #gauche
    elif xmin*fction[0]+fction[1] < ymax and xmin*fction[0]+fction[1] > ymin and cos(angle)<0 :
        return([xmin,xmin*fction[0]+fction[1],pi-angle,False])
    #haut
    elif fction[0]!=0 and (ymax-fction[1])/fction[0] >xmin and (ymax-fction[1])/fction[0] < xmax and sin(angle)>0:
        return([(ymax-fction[1])/fction[0],ymax,-angle,False])
    #bas
    elif fction[0]!=0 and (ymin-fction[1])/fction[0] >xmin and (ymin-fction[1])/fction[0] <xmax and sin(angle)<0:
        return([(ymin-fction[1])/fction[0],ymin,-angle,False])
```

```
def trajectoire (xinit,yinit , angle ):
    global matricejoueurs
    if xinit<xmin or xinit>xmax or yinit<ymin or yinit >ymax :
        return ("la balle n'est pas dans le terrain")
    stop=False
    nbrebond=0
    x=xinit
    y=yinit
    a=angle
    X=[x]
    Y=[y]
    affichage=True
    while not(stop) :

        liste = pointderebond (x,y,a)
        X.append(liste[0])
        Y.append(liste[1])

        if affichage and (liste[3] =="arreté" or liste[3]==True or nbrebond > 3) :
            plt.plot([0,106,106,0,0],[0,0,60,60,0],c='black')
            plt.plot([0,0],[22.75,37.25],'w-',lw=2)
            plt.plot([106,106],[22.75,37.25],'w-',lw=2)
            plt.plot(X,Y,color='blue')
            x=[]
            y=[]
            for i in range (0,len(matricejoueurs)):
                x.append(matricejoueurs[i][0])
                y.append(matricejoueurs[i][1])
            plt.scatter(x,y,s=50,c='red',alpha=0.5)
            plt.show()


        if liste[3] =="arreté" :
            return([nbrebond,liste[0],liste[1],False])

        if liste[3]==True :
            return([nbrebond,liste[0],liste[1],liste[3]])

        else:
            nbrebond = nbrebond +1
            x=liste[0]
            y=liste[1]
            a=liste[2]
        if nbrebond > 10 :
            return([nbrebond,liste[0],liste[1],liste[3]])
```

```
def testpoint (xinit,yinit,nbdetest):
    angleinter=pi/nbdetest
    angle=-pi/2
    nbdereussite=0
    for i in range (0,nbdetest) :
        liste=trajectoire(xinit,yinit,angle)
        if liste [3]==True:
            nbdereussite=nbdereussite+1
        angle=angle+angleinter
    return (nbdereussite)



def figure():
    fig = plt.figure()
    ax = fig.gca(projection='3d')

    # Make data.
    X = np.arange(0.0000001, 106, 0.5)
    Y = np.arange(0.0000001, 60, 0.5)
    X, Y = np.meshgrid(X, Y)
    Z=np.zeros((len(X),len(X[0])))


    for i in range (len(X)):
        for j in range (len(X[i])):

            print(int((Y[i][j]/60)*100),"%")
            Z[i][j]=testpoint(X[i][j],Y[i][j],100)/100

    print(Z)
    # Plot the surface.
    surf = ax.plot_surface(X, Y, Z,cmap=cm.coolwarm,linewidth=0,antialiased=False)

    # Customize the z axis.
    ax.set_zlim(0, 1)
    ax.zaxis.set_major_locator(LinearLocator(10))
    ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

    # Add a color bar which maps values to colors.
    fig.colorbar(surf, shrink=0.5, aspect=5)

    plt.show()
```

Loi des grands nombres

```
def touslespoints (nbdex , nbdey ):
    retour=[]
    pasx=xmax/nbdex
    pasy=ymax/nbdey
    for i in range (0,nbdex):
        for j in range (0,nbdey):
            nbdereussite=testpoint(i*pasx+0.00000000001,j*pasy+0.00000000000001,100)
            retour.append([i*pasx,j*pasy,nbdereussite])
    return(retour)

#tire uniquement depuis les joueurs
def testpointjoueur (xinit,yinit,nbdetest):
    angleinter=pi/nbdetest
    angle=pi/2
    nbdereussite=0
    for i in range (0,nbdetest) :
        liste=trajectoire(xinit,yinit,angle)
        if liste [3]==True:
            nbdereussite=nbdereussite+1
        angle=angle-angleinter
    return (nbdereussite)


def touslespointjoueurs (nbdetest):
    global largeurbut
    global xmin
    global xmax
    global ymin
    global ymax
    global matricejoueurs
    global largeurjoueur
    matriceretour=[]
    for i in range (len(matricejoueurs)):
        if matricejoueurs[i][2]=="gauche":
            nbdereussites=testpointjoueur(matricejoueurs[i][0]+0.00000000001,matricejoueurs[i][1]+0.00000000001,nbdetest)
            print(nbdereussites)
            matriceretour.append([matricejoueurs[i][0],matricejoueurs[i][1],nbdereussites])
    return(matriceretour)
```

```
def testmatricejoueuraleatoire (nbdetest):
    #creation d'une matrice aleatoire des joueurs de babyfoot
    affichage=True
    positionbars=[]
    a=random()
    a=a*30+15
    positionbars.append(a)
    a=random()
    a=a*35
    positionbars.append(a)
    a=random()
    a=a*24
    positionbars.append(a)
    a=random()
    a=a*20
    positionbars.append(a)
    a=random()
    a=a*20
    positionbars.append(a)
    a=random()
    a=a*24
    positionbars.append(a)
    a=random()
    a=a*35
    positionbars.append(a)
    a=random()
    a=a*30+15
    positionbars.append(a)


    premierebar=5.75
    matricejoueurs=[]
    matricejoueurs.append([premierebar,positionbars[0],"gauche"])
    for i in range (0,2):
        matricejoueurs.append([premierebar + entrebar,positionbars[1]+i*entredef,"gauche"])
    for i in range (0,3):
        matricejoueurs.append([premierebar + entrebar*2,positionbars[2]+i*entreatt,"droite"])
    for i in range (0,5):
        matricejoueurs.append([premierebar+entrebar*3,positionbars[3]+i*entremil,"gauche"])
    for i in range (0,5):
        matricejoueurs.append([premierebar+entrebar*4,positionbars[4]+i*entremil,"droite"])
    for i in range (0,3):
        matricejoueurs.append([premierebar+entrebar*5,positionbars[5]+i*entreatt,"gauche"])
```

```
    for i in range (0,3):
        matricejoueurs.append([premierebar+entrebar*5,positionbars[5]+i*entreatt,"gauche"])
    for i in range (0,2):
        matricejoueurs.append([premierebar+entrebar*6,positionbars[6]+i*entredef,"droite"])
    matricejoueurs.append([premierebar+entrebar*7,positionbars[7],"droite"])

    if affichage:
        X=[]
        Y=[]
        plt.plot([0,106,106,0,0],[0,0,60,60,0],c='black')
        plt.plot([0,0],[22.75,37.25],'w-',lw=2)
        plt.plot([106,106],[22.75,37.25],'w-',lw=2)
        for i in range (0,len(matricejoueurs)):
            X.append(matricejoueurs[i][0])
            Y.append(matricejoueurs[i][1])
        plt.scatter(X,Y)
        plt.show()

#calcul de la matrice de retour
    matriceretour=[]
    matricereussite=[]
    for i in range (len(matricejoueurs)):
        if matricejoueurs[i][2]=="gauche":
            nbdereussites=testpointjoueur(matricejoueurs[i][0]+0.00000000001,matricejoueurs[i][1]+0.00000000001,nbdetest)
            #print(nbdereussites)
            matriceretour.append([matricejoueurs[i][0],matricejoueurs[i][1],nbdereussites])
            matricereussite.append(nbdereussites)
    print(matricereussite)
    print('llllll')
    return(matriceretour)



#donne la moyenne des testmatricealeatoires
def testfinal (nbdetest,nbiteration):
    matricemoyenne=[0,0,0,0,0,0,0,0,0,0,0]
    for i in range (nbiteration):
        matriceretour=testmatricejoueuraleatoire (nbdetest)
        print((i/nbiteration)*100,'%')
        for j in range (len(matricemoyenne)):
            matricemoyenne[j]=(i*matricemoyenne[j]+matriceretour[j][2])/(i+1)
        # print("m=",matricemoyenne)
    return(matricemoyenne)
```

```
def affichagegrandnombre(nbdetest,nbiteration):
    matricemoyenne=[0,0,0,0,0,0,0,0,0,0,0]
    x=[]
    y1=[]
    y2=[]
    y3=[]
    y4=[]
    y5=[]
    y6=[]
    y7=[]
    y8=[]
    y9=[]
    y10=[]
    y11=[]
    for i in range (nbiteration):
        matriceretour=testmatricejoueuraleatoire (nbdetest)
        print("i=",i)
        for j in range (len(matricemoyenne)):
            matricemoyenne[j]=(i*matricemoyenne[j]+matriceretour[j][2])/(i+1)

        print("m=",matricemoyenne)
        x.append(i)
        y1.append(matricemoyenne[0])
        y2.append(matricemoyenne[1])
        y3.append(matricemoyenne[2])
        y4.append(matricemoyenne[3])
        y5.append(matricemoyenne[4])
        y6.append(matricemoyenne[5])
        y7.append(matricemoyenne[6])
        y8.append(matricemoyenne[7])
        y9.append(matricemoyenne[8])
        y10.append(matricemoyenne[9])
        y11.append(matricemoyenne[10])
    plt.plot(x,y1,c='b')
    #plt.plot(x,y2)
    #plt.plot(x,y3)
    #plt.plot(x,y4)
    #plt.plot(x,y5)
    #plt.plot(x,y6)
    #plt.plot(x,y7)
    #plt.plot(x,y8)
    #plt.plot(x,y9)
    plt.plot(x,y10,c='r')
    #plt.plot(x,y11)
    plt.show()
    return(x,y1)
```