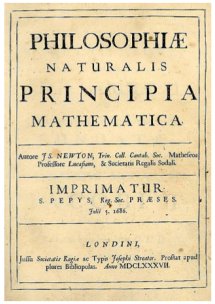
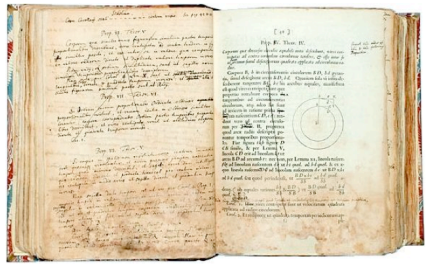
Matlab: applications en mécanique

LA207, Université Pierre et Marie Curie.

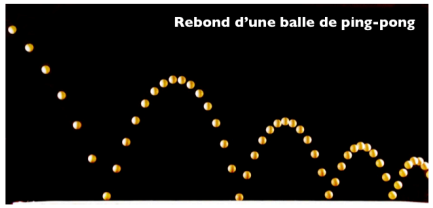
2.4 TP3: Newton et la balle de ping-pong

Newton décrit dans les "principia", une des premières mathématisations d'une loi de la physique, le second principe de la dynamique, qui dit que un corps soumis à des forces est entraîné dans un mouvement: la somme des forces appliquées est égale à la masse fois l'accélération. Avec cette loi, une fois admis que les corps célestes exercent les uns sur les autres des forces d'attraction, on peut comprendre et prédire le mouvement des étoiles, des planètes et de leurs satellites. La balle de ping-pong elle aussi est soumise à cette loi. Entre les rebonds, la seule force appliquée est son poids, et pendant le rebond, la réaction du support joue. Il y a aussi d'autres forces, dont l'action est moins prépondérante: les efforts dus au déplacement de l'air: efforts aérodynamiques, et efforts de déformation de la balle pendant le rebond.





Isaac Newton:
les principes mathématiques de la philosophie naturelle
(la physique)



Dans ce TP, nous allons utiliser la photo du rebond de la balle de ping-pong comme une expérimentation physique pour vérifier la loi de Newton, et estimer les pertes énergétiques au cours du mouvement.

Compétences techniques:

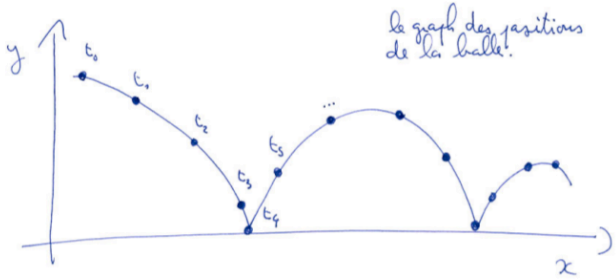
- Mesurer la position de points sur une image.
- Calcul de la vitesse et de l'accélération à partir des positions successives.
- Tracer une trajectoire théorique, en prenant en compte les rebonds.

Données: Masse de la balle: m=5 grammes; largeur et hauteur d'un pixel de l'image: dx=1.7 millimètre; Intervalle entre les prises de vue: dt=0.04 secondes; Accélération de la gravité: g=9.8 mètres par seconde au carré.

2.4.1 Manipulations

1. L'image de la balle de ping-pong est stockée sur le disque avec pour nom de fichier: `pingpong.png`. Chargez la dans matlab avec la commande `imread`, puis affichez là, avec la commande `image`.

2. Avec la fonction `ginput` mesurez sur l'image la position du centre de la balle aux temps successifs que vous stockerez dans un fichier sur le disque. Lisez ce ficher pour avoir les données dans votre workspace, et mettez les coordonnées dans deux tableaux `x` et `y`. Choisissez une origine pour votre référentiel, et transformez les coordonnées pour avoir les distances en mètres. Tracez le graphique de la trajectoire avec la fonction `plot`.



3. Calculez le vecteur vitesse: la vitesse selon x pour un temps donné, peut être calculée comme la position selon x au temps suivant moins la position selon x au temps précédent, divisé par l'intervalle de temps entre ces deux instants:

$$v_x(t) \approx \frac{x(t+dt) - x(t-dt)}{2dt}$$

Tracez le graph de la vitesse v_x selon x et v_y selon y en fonction du temps.

4. On calcule l'accélération. Pour la calculer, on se souviendra que l'accélératio c'est la "vitesse de la vitesse". Tracez le graph de l'accélération selon x et selon y en fonction du temps. A quoi est égale l'accélération verticale? A quoi est égale l'accélération horizontale? Que se passe-t'il lors des rebonds?

2.4.2 Etude

L'énergie mécanique de notre balle, c'est la somme de l'énergie cinétique $E_c = m\|v\|^2/2$ et de l'énergie potentielle $E_p = mgy$. Au cours du mouvement, l'énergie peut être transférée de l'énergie cinétique vers l'énergie potentielle et réciproquement, par contre, s'il n'y a pas de frottement, l'énergie totale reste constante.

Tracez un graph qui montre comment les énergies varient dans le temps. Grâce à ce graph, analysez les pertes d'énergie de notre balle. Pour cela, posez-vous les questions: l'énergie est-elle constante ou pas? Quels sont les effets physiques qui peuvent faire varier l'énergie? Que se passe-t'il lors du rebond?

Vérifiez que le coefficient α de restitution d'énergie lors du rebond est le même pour tous les rebonds: énergie après le rebond=α fois l'énergie avant le rebond. Donnez la valeur de ce coefficient de restitution.

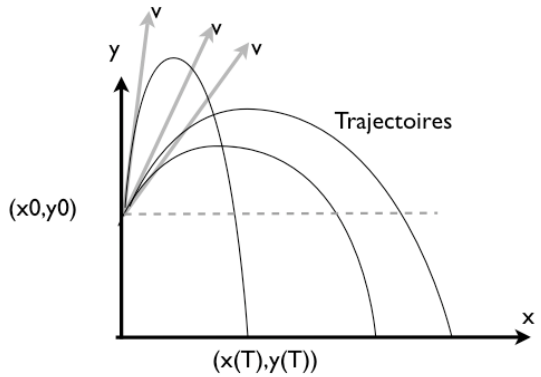
Nous voulons maintenant comparer la trajectoire mesurée avec une trajectoire théorique qui obéit à la loi de Newton. On va prendre en compte la perte d'énergie pendant le rebond, mais pas la résistance aérodynamique. Selon la loi de Newton, la trajectoire est

$$x(t) = x_0 + v_x t, \quad y(t) = y_0 + v_y t - gt^2/2$$

avec (x_0, y_0) et (v_x, v_y) les position et vitesse initiales de la balle. Ecrivez une fonction `rebondfonc` qui calcule la trajectoire de la balle pour (x_0, y_0) et (v_x, v_y) donnés. La syntaxe de création de fonctions dans Matlab est décrite dans les notes de cours. Cette fonction donnera en arguments de sortie la trajectoire de la balle, le temps T auquel la balle touche le sol ($y(T) = 0$) et la vitesse $(v_x(T), v_y(T))$ et la position de la balle $(x(T), y(T))$ à cet instant T.

Tracez sur un graphique la superposition des trajectoires pour $x_0 = 0, y_0 = 1$, et $\|v\| = 10$, ou la vitesse initiale fait un angle θ avec l'horizontale.

Vous prendrez plusieurs valeurs pour θ entre $\pi/2$ et $-\pi/2$.



2.4.3 Pour aller plus loin

Comparaison mesures/théorie: lors d'un rebond, la vitesse change comme ceci:

$$v_x \to \alpha \times v_x, \quad v_y \to -\alpha \times v_y,$$

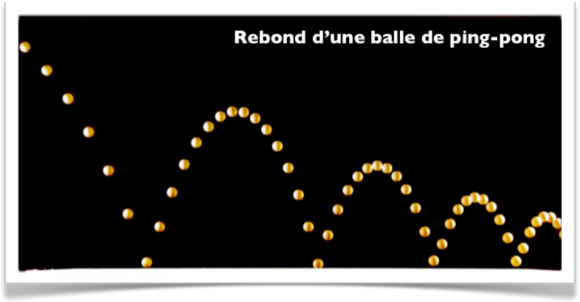
c'est à dire, les deux composantes de la vitesse sont multipliées par un facteur de restitution, et la vitesse selon y change de signe. Utiliser votre fonction `rebondfonc` pour comparer la trajectoire mesurée avec la trajectoire théorique pour trois rebond successifs. D'après ce graphique, donnez la valeur du coeficient de restitution en procédant par approximations successives.

Matlab: applications en mécanique

LA207. Université Pierre et Marie Curie

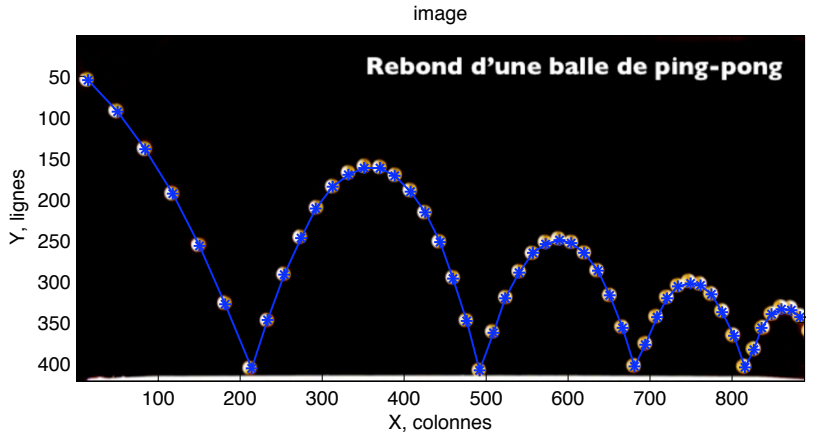
www.lmm.jussieu.fr/~hoepffner/enseignement

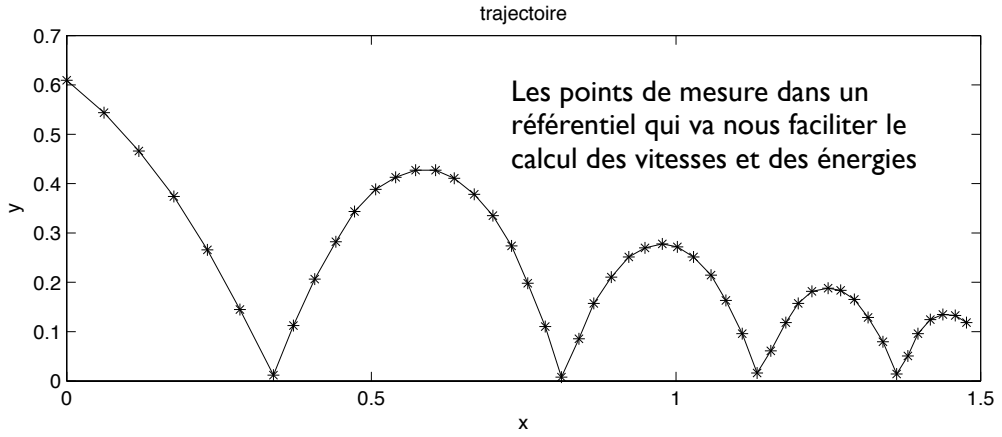
TP4: le rebond de la balle de ping-pong



On trace la figure originelle avec en superposition les points de mesure dans le référentiel des pixels de l'image. on vérifie ainsi la qualité des points de mesure. on effectue ensuite un changement de référentiel. ici y=0 correspond à la position la plus basse de la balle, et x=0 correspond à la position la plus à gauche sur l'image. On a utilisé l'information que la taille d'un pixel est 0.0017 mètre (c'est la variable dx dans le script).

```
clear all; clf;

% affichage de l'image
subplot(2,1,1);
a=imread('pingpong.png');
image(a); axis equal tight

% on trace sur l'image les points de mesure
hold on
d=load('pingpong.dat');
x=d(:,1); y=d(:,2);
plot(x,y,'b*-','linewidth',1)
xlabel('X, colonnes');
ylabel('Y, lignes');
title('image')


% changement de référentiel
subplot(2,1,2);
y0=413; % position de y=0
x0=x(1); % position de x=0
dx=0.0017;

x=dx*(x-x0);
y=dx*(y-y0);
y=-y; % inversion de l'axe des y

% on trace la trajectoire
plot(x,y,'k*-');
xlabel('x');
ylabel('y');
title('trajectoire')
```

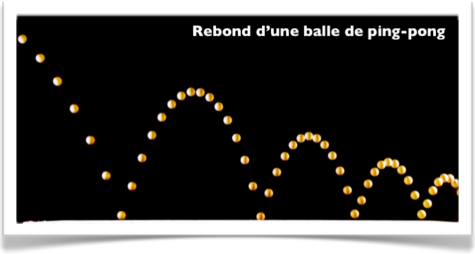




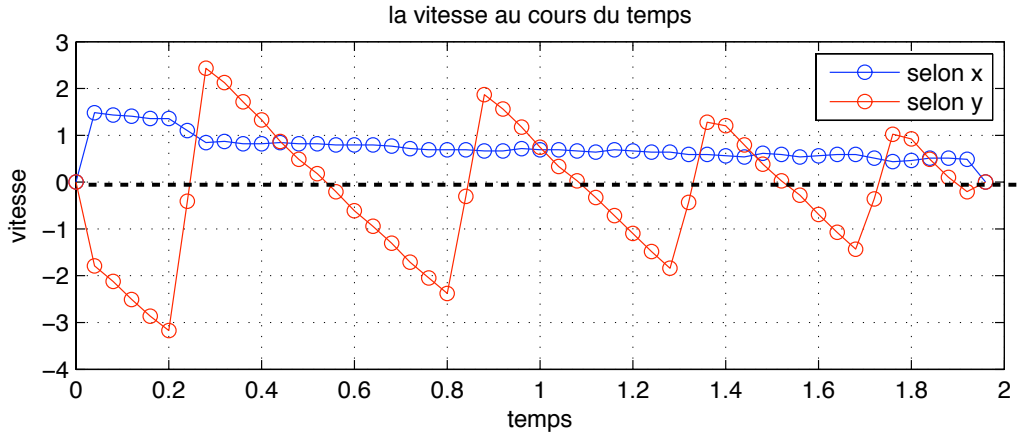
Vitesse et accélération

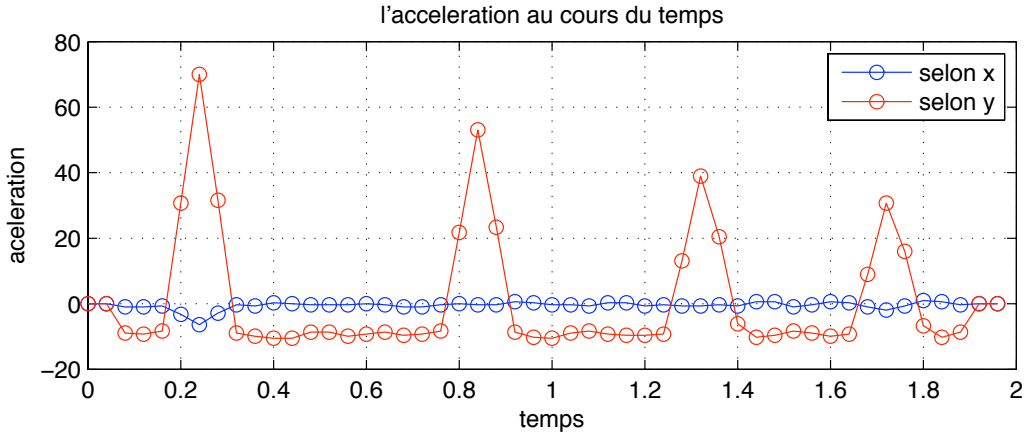
Pour le calcul de la vitesse et de l'accélération, on utilise la formule donnée dans l'énoncé de TP. On crée deux vecteurs remplis de zéros, avec autant de lignes que de positions successives de la balle. On ne pourra pas calculer la vitesse pour la première ni la dernière position parce que la formule demande la position avant et après. On aurais pu utiliser une formule décentrée. Vous apprendrez cela l'année prochaine.

```
g=9.81; % gravité
m=0.005; % masse de la balle
n=length(x); % nombre de points de mesure
dt=0.04; % intervale de temps entre les images
tvec=0:dt:(n-1)*dt; % vecteur du temps

% calcul de la vitesse
vx=zeros(n,1);
vy=zeros(n,1);
for ind=2:n-1
    vx(ind)=(x(ind+1)-x(ind-1))/(2*dt);
    vy(ind)=(y(ind+1)-y(ind-1))/(2*dt);
end


% calcul de l'acceleration
ax=zeros(n,1);
ay=zeros(n,1);
for ind=3:n-2
    ax(ind)=(vx(ind+1)-vx(ind-1))/(2*dt);
    ay(ind)=(vy(ind+1)-vy(ind-1))/(2*dt);
end

% on trace la vitesse et l'acceleration au cours du temps
subplot(2,1,1);
plot(tvec,vx,'bo-',tvec,vy,'ro-')
xlabel('temps');
ylabel('vitesse');
title('la vitesse au cours du temps')
legend('selon x','selon y')
grid on

subplot(2,1,2);
plot(tvec,ax,'bo-',tvec,ay,'ro-')
xlabel('temps');
ylabel('aceleration');
title('l''acceleration au cours du temps')
legend('selon x','selon y')
grid on
```



La vitesse selon x est quasiment constante: la gravité ne s'applique que selon la verticale, cependant il y a une petite force de frottement qui ralentit la balle. La vitesse selon y change de signe à chaque rebond, et entre les rebonds, elle décroit linéairement avec le temps, à cause de la gravité.

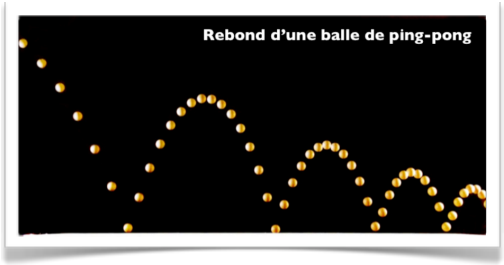


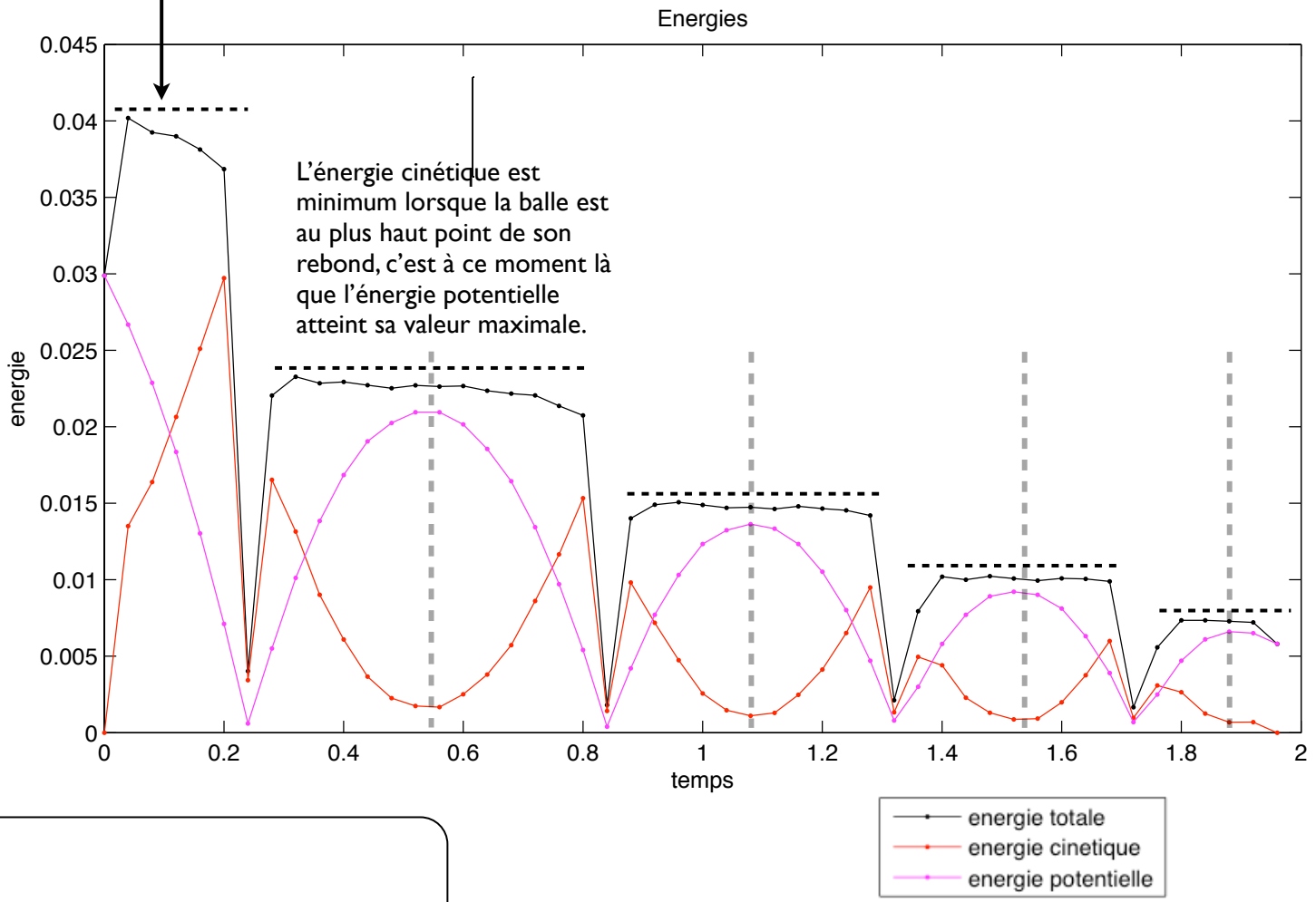


L'accélération selon x est quasiment nulle. Selon y elle est plus ou moins constante, de valeur à peu près -10 ... C'est l'accélération de la gravité.

Energies

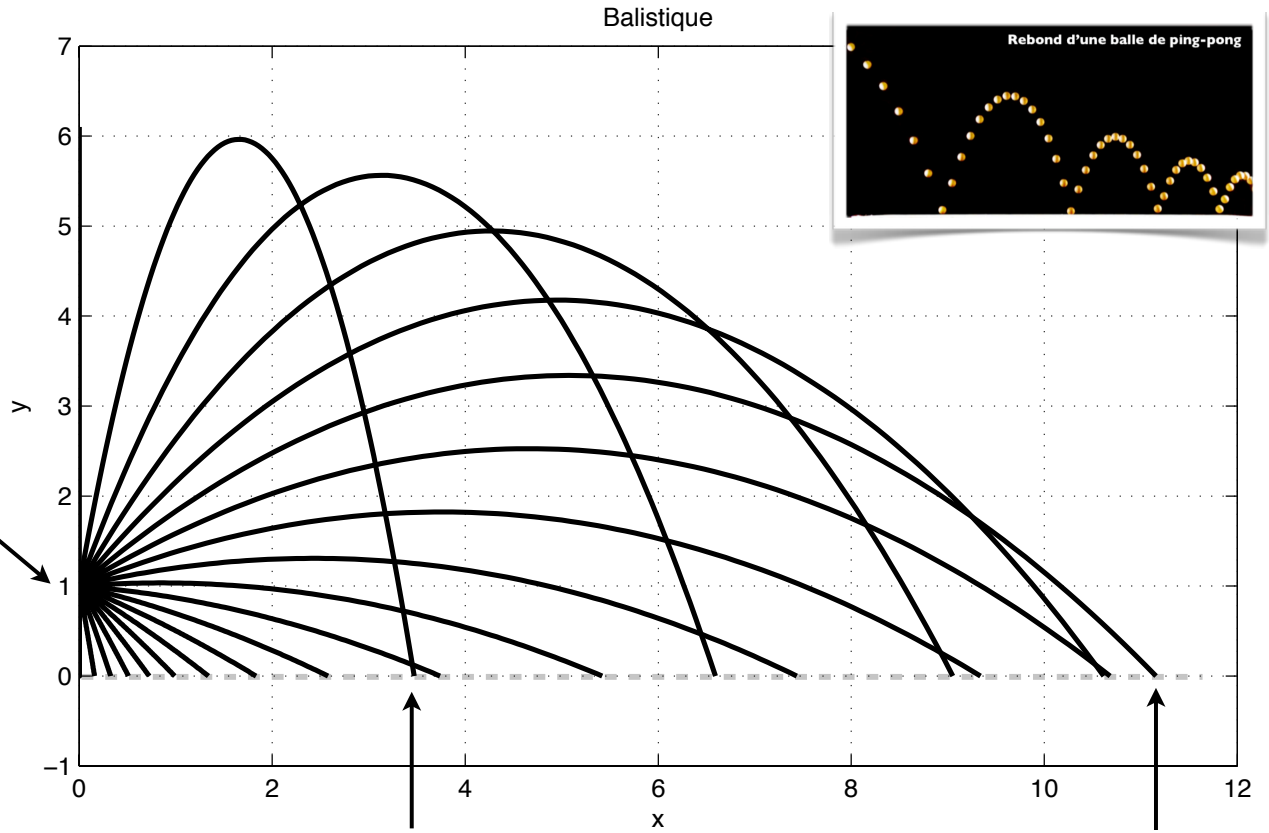
Il est maintenant facile de calculer les différentes contributions à l'énergie: cinétque et potentielle. On a l'énergie totale en faisant la somme. On voit que entre les rebonds, l'énergie totale est quasiment constante. Elle décroit un peu à cause des forces de frottement aérodynamique. par contre l'énergie diminue beaucoup à chaque rebond.






```
% energie cinetique et potentielle
ec=m*(vx.^2+vy.^2)/2;
ep=m*y*g;

% on trace les energies
plot(tvec,ec+ep,'k.-',tvec,ec,'.-r',tvec,ep,'m.-')
legend('energie totale','energie cinetique','energie potentielle');
xlabel('temps'); ylabel('energie'); title('Energies')
```

Balistique



La position initiale est (0,1) pour toutes les trajectoires

On fait une boucle sur l'angle initial, et on garde la même position initiale et la même intensité de vitesse initiale.

On calcule dans notre fonction le temps et la position lorsque la balle touche le sol avec une formule théorique.

Il y a un angle optimal pour lancer la balle le plus loin possible, c'est 45° s'il n'y a pas de frottement aérodynamique.

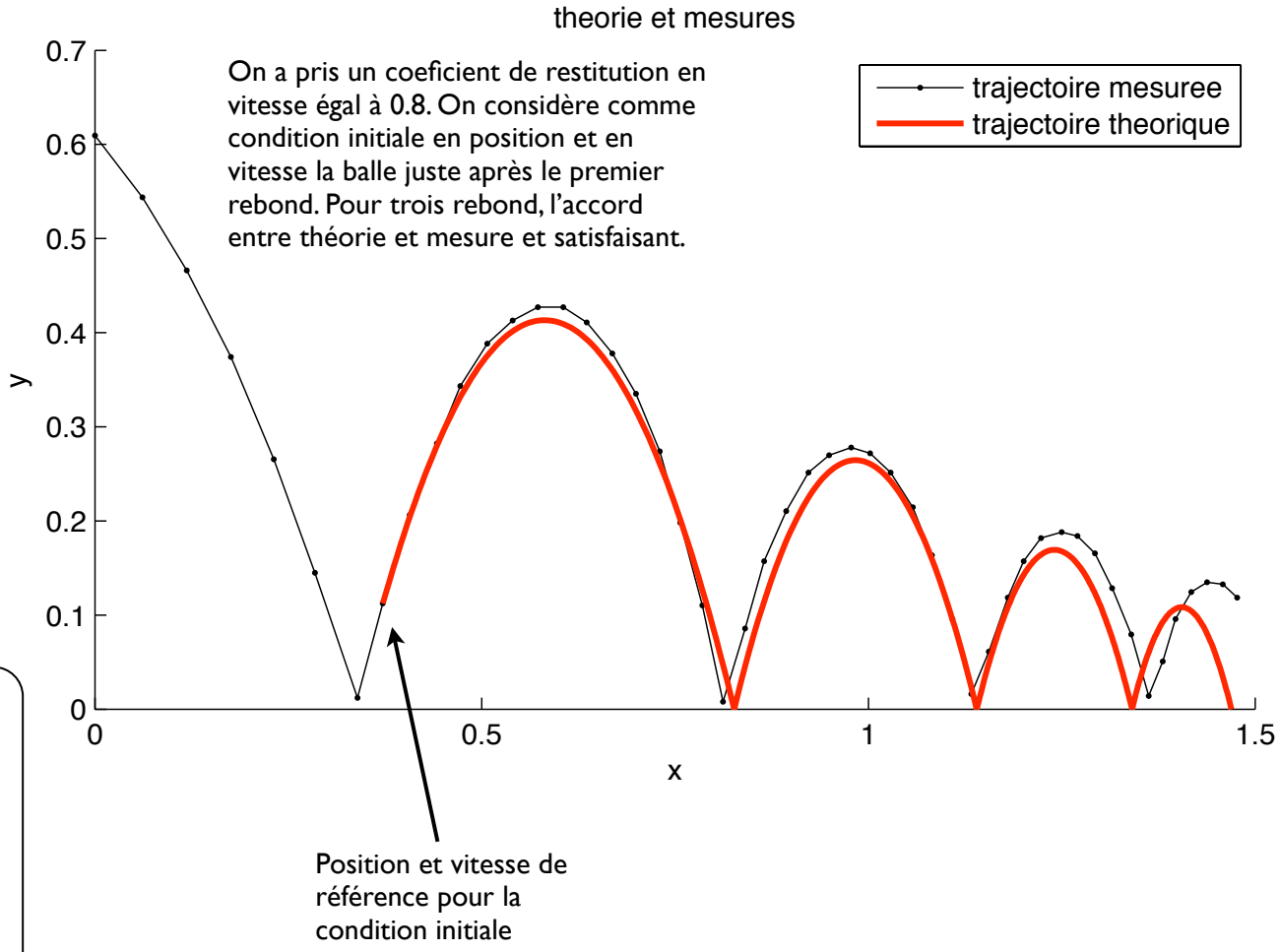
```
%%%%%%%% trajectoires balistique:
% avec la fonction rebondfonc.m
thetavec=linspace(-pi/2,pi/2,20);
v=10; % amplitude de la vitesse initiale


% Boucle sur l'angle initial theta
for ind=1:length(thetavec);
    theta=thetavec(ind);
    vvx=v*cos(theta);
    vvy=v*sin(theta);

    % Position initiale et calcul de la trajectoire
    x0=0;
    y0=1;
    [x0,y0,vvx,vvy,xx,yy]=rebondfonc(x0,y0,vvx,vvy);
    plot(xx,yy,'k','linewidth',2)
    hold on
end
grid on
xlabel('x'); ylabel('y'); title('Balistique')
```

La fonction rebondfonc qui donne la trajectoire pour une condition initiale donnée en position et en vitesse: elle calcule le temps ou la balle va toucher le sol.

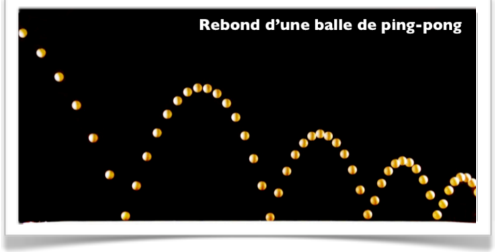
```
function [x0,y0,vx,vy,x,y]=f(x0,y0,vx,vy)
% pour le TP du rebond de la balle

g=9.8;

% on calcule le temp ou la balle touche le sol
tmax=(vy+sqrt(vy^2+4*y0*g/2))/g;
t=linspace(0,tmax,100);

% la trajectoire
x=vx*t+x0;
y=-g*t.^2/2+t*vy+y0;

% la vitesse et la position finale
vy=-g*tmax+vy;
x0=x0+vx*tmax;
y0=0;
```

Trajectoires théoriques

```
% position et vitesse initiale de notre balle
% prises à la position loc
loc=8;
y0=y(loc);
x0=x(loc);
vvx=vx(loc);
vvy=vy(loc);

% coef de restitution
resti=0.8;

% on trace les points de mesure
plot(x,y,'k.-'); hold on

for ind=1:4
% on calcule et on trace la parabole
[x0,y0,vvx,vvy,xx,yy]=rebondfonc(x0,y0,vvx,vvy);
plot(xx,yy,'r','linewidth',2);

% transformation de la vitesse lors du rebond
vvx=resti*vvx;
vvy=-resti*vvy;

end

xlabel('x'); ylabel('y'); title('theorie et mesures')
legend('trajectoire mesuree','trajectoire theorique')
```



On fait une boucle pour les rebonds: à chaque itération on calcule la trajectoire d'un rebond, et on donne la position et vitesse de la balle lorsque celle-ci touche de nouveau le sol. On peut ainsi calculer autant de rebond que l'on veut.



Précision

Ici une petite étude suplémentaire: on prend comme condition initiale plein de positions successives de la balle pour voir comment la précision que nous avons sur la mesure de la vitesse va influer sur la dispersion des trajectoires. Ici on a pris comme position et vitesses initiales les positions de 8 à 20, ce sont toutes des positions lors du premier rebond.