Table of Contents

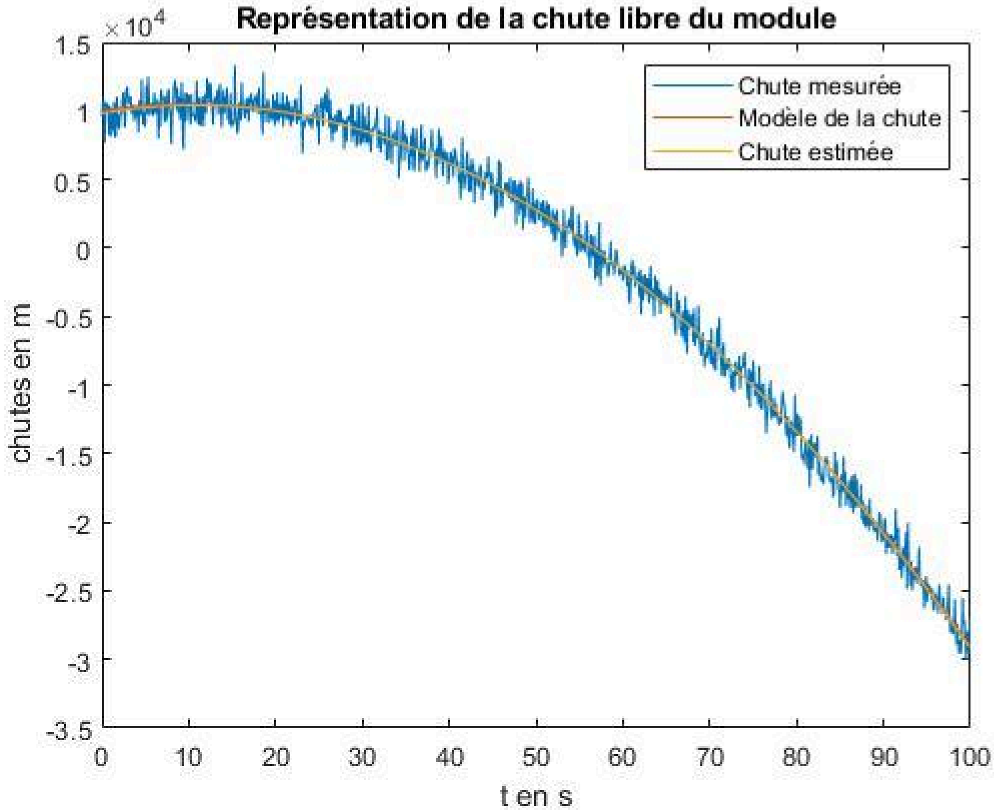
Simulation de la situation Apollo

Simulation de x(t)- la chute de la capsule-, v(t) - le bruit - et y(t) le signal capté par le radar.

```
%Initialisation du modèle
a = -1/2*9.81 ; v0 = 100; x0 = 10000; K = 1000;
t = (0:0.1:(K-1)*0.1)';

v_t = 1000*randn(K,1); %Bruit blanc
x_t = a*t.^2 + v0*t+x0; %Modèle de la chute
y_t = x_t + v_t; %Signal mesuré
```

Filtrage du bruit

```
M = [t.^2 , t , ones(size(t))]; %Récupération de la matrice M
theta = M\y_t; %Calcul des paramètres optimaux
x_estim = M*theta; %filtrage optimal du bruit

figure(1),clf,plot(t,[y_t x_t x_estim]),xlabel("t en
 s"),ylabel("chutes en m")
legend("Chute mesurée","Modèle de la chute","Chute
 estimée"),title("Représentation de la chute libre du module")
%Observations des résultats: L'estimation de la chute est proche du
 modèle.
```



Calcul de l'instant t_plouf pour lequel la capsule touchera l'eau

```
racines = roots(theta);
t_plouf = racines(racines>0)

% t_{plouf} = 56.3 secondes
```

```
t_plouf =

   56.5384
```

MONTE CARLO - Répétition de la simulation avec Monté-Carlo et visualisation de l'évolution de t_plouf

```
%On réalise l'expérience 10 000 fois et on observe l'évolution de
%t_{plouf}.
```

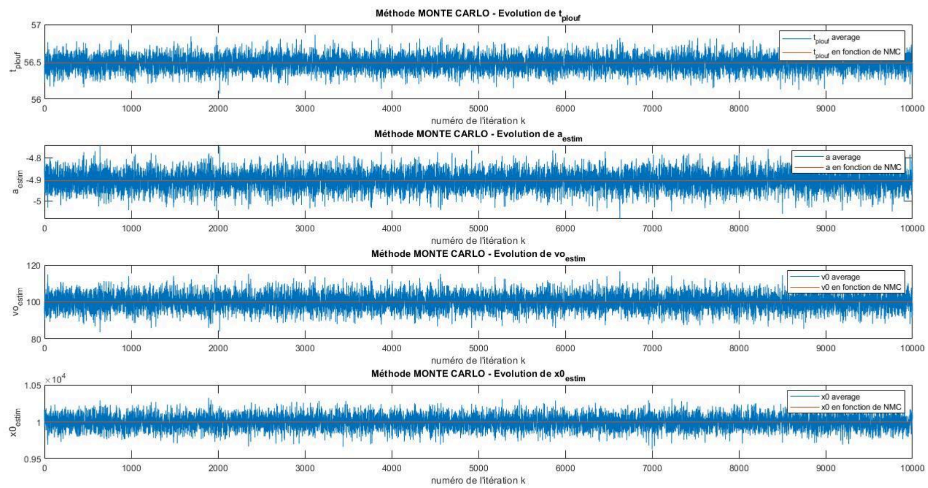
```
NMC = 10000;

%Initialisation des vecteurs t_ploufs, a_estim, v0_estim et x0_estim
t_ploufs = zeros(NMC,1);
a_estim = zeros(NMC,1);
vo_estim = zeros(NMC,1);
x0_estim = zeros(NMC,1);

for k=1:NMC

    v_t = 1000*randn(K,1); %Génère un nouveau bruit
    x_t = a*t.^2 + v0*t+x0;
    y_t = x_t + v_t; %Signal mesuré à l'expérience k
    theta = M\y_t; % Calcul des paramètres optimaux
    a_estim(k) = theta(1);
    vo_estim(k) = theta(2);
    x0_estim(k) = theta(3);
    x_estim = M*theta; %filtrage optimal du bruit
    racines = roots(theta);
    t_plouf = racines(racines>0);
    t_ploufs(k) = t_plouf;
end
```

Calcul de l'espérence mathématique et de la variance des t_ploufs

```
esperance_t_ploufs = mean(t_ploufs);
esperance_a_estim = mean(a_estim);
esperance_v0_estim = mean(vo_estim);
esperance_x0_estim = mean(x0_estim);

variance_t_ploufs = var(t_ploufs)
variance_a_estim = var(a_estim)
variance_vo_estim = var(vo_estim)
variance_x0_estim = var(x0_estim)


k = (1:NMC)';
figure(2),clf,
subplot(411),plot(k,t_ploufs,k,ones(NMC,1)*esperance_t_ploufs)
title("Méthode MONTE CARLO - Evolution de t_{plouf}"),xlabel("numéro
 de l'itération k"),ylabel("t_{plouf}"),legend("t_{plouf}
 average","t_{plouf} en fonction de NMC")
subplot(412),plot(k,a_estim,k,ones(NMC,1)*esperance_a_estim)
title("Méthode MONTE CARLO - Evolution de a_{estim}"),xlabel("numéro
 de l'itération k"),ylabel("a_{estim}"),legend("a average","a en
 fonction de NMC")
subplot(413),plot(k,vo_estim,k,ones(NMC,1)*esperance_v0_estim)
title("Méthode MONTE CARLO - Evolution de vo_{estim}"),xlabel("numéro
 de l'itération k"),ylabel("vo_{estim}"),legend("v0 average","v0 en
 fonction de NMC")
subplot(414),plot(k,x0_estim,k,ones(NMC,1)*esperance_x0_estim)
```

```
title("Méthode MONTE CARLO - Evolution de x0_{estim}"),xlabel("numéro
 de l'itération k"),ylabel("x0_{estim}"),legend("x0 average","x0 en
 fonction de NMC")

%Observations : On remarque que les valeurs de t_ploufs, a_estim,
 v0_estim et x0_estim
%oscillent autour de leur espérance qui est très proche de la valeur
%théorique.
```

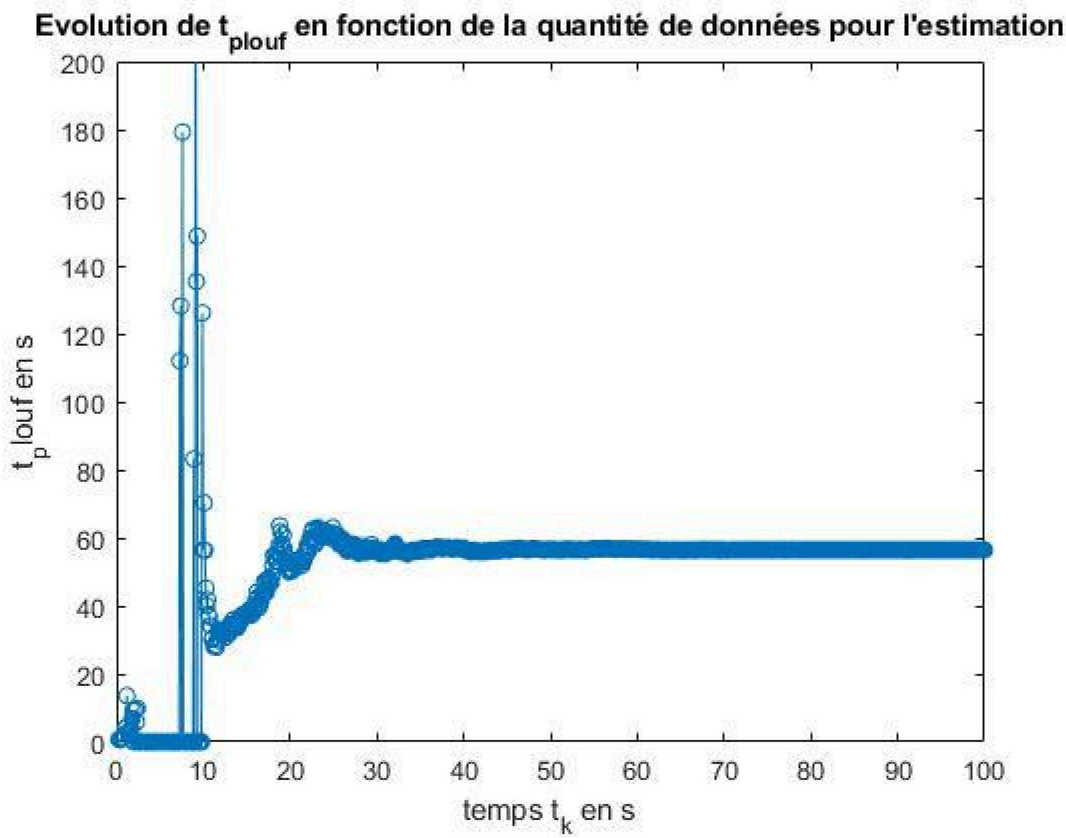
```
variance_t_ploufs =

    0.0106


variance_a_estim =

    0.0018


variance_vo_estim =

   19.0330


variance_x0_estim =

   8.8891e+03
```



Deuxième approche : Méthode lourde

```
v_t = 1000*randn(K,1); %Génère le bruit blanc
x_t = a*t.^2 + v0*t+x0; %Génère le modèle de la chute
y_t = x_t + v_t; %Signal mesuré
```

```
t_ploufs_2 = zeros(K-2,1);
nb_ech_vect = (3:1:K); % On fait varier le nombre d'échantillons de 3
 à K=1000

for nb_ech=3:K
    t_i = t(1:nb_ech); %Définition du vecteur temps lié au k
 échantillons
    M_i = [t_i.^2, t_i, ones(size(t_i))]; %Création de la matrice
    theta_i = M_i\y_t(1:nb_ech); %Calcul des paramètres optimaux
    x_estim_i = M_i*theta_i; % Estimation de la chute
    racines_i = roots(theta_i); %Calcul des racines pour trouver
 t_plouf
    t_plouf = racines_i(racines_i>0); % On récupère la valeur positive
    if imag(t_plouf)== 0 %On pose garde t_plouf c'est un réel sinon
 valeur 0
        t_ploufs_2(nb_ech-2) = t_plouf;
    end
end
t_k = nb_ech_vect.*0.1;
figure(3), clf, plot(t_k,t_ploufs_2, '-o'),ylim([0,200])
title("Evolution de t_{plouf} en fonction de la quantité de données
 pour l'estimation")
xlabel("temps t_k en s"),ylabel("t_plouf en s")

%Observations : La valeur de t_plouf se stabilise autour de la valeur
 de
%t_plouf trouvée auparavant après 35 secondes de chutes. On est donc
%capable d'informer l'astronaute de l'instant de contact à partir de
 35
%secondes de chute soit 20 secondes avant l'impact.
```



Deuxième approche : Méthode lourde +NMC

```
v_t = 1000*randn(K,1);
x_t = a*t.^2 + v0*t+x0;
y_t = x_t + v_t;

t_ploufs_2 = zeros(K-2,NMC);
nb_ech_vect = (3:1:K);
esperances = zeros(NMC,1);
variances = zeros(NMC,1);

% Session commentée car trop lourde pour mon ordinateur

% for i=1:NMC
%     for nb_ech=3:K
%         t_i = t(1:nb_ech);
%         M_i = [t_i.^2, t_i, ones(size(t_i))];
%         theta_i = M_i\y_t(1:nb_ech);
%         x_estim_i = M_i*theta_i;
%         racines_i = roots(theta_i);
%         t_plouf = racines_i(racines_i>0);
%         if imag(t_plouf)== 0
%             t_ploufs_2(nb_ech-2,i) = t_plouf;
%         end
%     end
```

```
%     esperances(i) = mean(t_ploufs_2(:,i));
%     variances(i) = var(t_ploufs_2(:,i));
% end
% t_k = nb_ech_vect.*0.1;
% figure(4), clf, plot(t_k,t_ploufs_2(:,1), '-o'),ylim([0,200])
% title("Evolution de t_{plouf} en fonction de la quantité de données
 pour l'estimation")
% xlabel("temps t_k en s"),ylabel("t_plouf en s")
```

Deuxième approche : Récursivité - Algo de Kalman (NON FINI)

```
g = 9.81;a = -1/2*g; v0 = 100; x0 = 10000; K = 1000;

t = (0:0.1:(K-1)*0.1)';

x_t = a*t.^2 + v0*t+x0;
x2_t = -g * t+v0;
x2_point_t = 0*t + -g;
v_t = 1000*randn(K,1);

%Equation  d'état
A = [0 1;0 0];
X_t = [x_t'; x2_t'];
X_point_t = A*X_t + [0; -g];

%Equation d'observation
C = [1 0];
Y_t = C* [x_t'; x2_t'] ;

%Filtrage de Kalman à réaliser plus tard.
```

Published with MATLAB® R2018a