-9 => 41 / 50

UNIVERSITÉ DE SHERBROOKE

Faculté de genie

Département de genie électrique et informatique

RÉSULTATS ET VALIDATION PROJET 3

GEI 720 Commande multivariable appliquée à l'aérospatiale

PRÉSENTÉ À

Jean DE LAFONTAINE

PAR

Oussama BOUSSELSAL

Sebastien DEMERS

Sherbrooke (Québec) Canada Novembre 2016

Table des matières

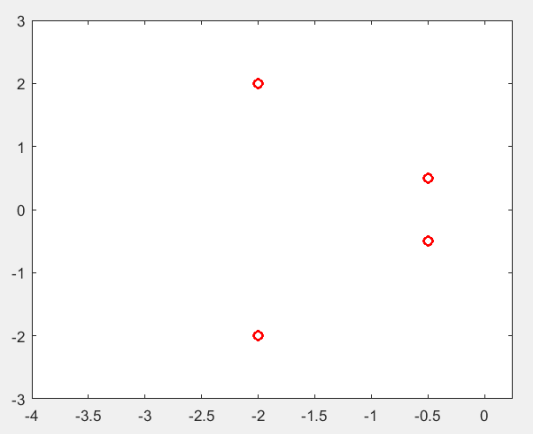
PARTIE A : Design du régulateur

P3-1

On choisit les pôles dominants de sorte que le temps de stabilisation de $t_s \leq 8s$ et le facteur d'amortissement est $\sqrt{2}/2$. Les pôles du deuxième mode sont placés selon un "scale" de facteur 4. Voir figure ci-dessous.

```
Ts = 8;
Zeta = sqrt(2)/2;
Wn = 4/(Zeta*Ts);
Poled_1 = -Zeta*Wn + Wn*sqrt(1-(Zeta^2))*1j;
Poled_2 = -Zeta*Wn - Wn*sqrt(1-(Zeta^2))*1j;
Facteur_pole = 1/4;
Wn2 = 4/(Zeta*Ts*Facteur_pole);
Poled_3 = -Zeta*Wn2 + Wn2*sqrt(1-(Zeta^2))*1j;
Poled_4 = -Zeta*Wn2 - Wn2*sqrt(1-(Zeta^2))*1j;
Poled = [Poled_1, Poled_2, Poled_3,Poled_4];
```

Poles desires : $-0.5 \pm 0.5i$ & $-2 \pm 2i$



OK

Figure 1: Poles

P3-2

OK

```
% Question P3-2
% COMMANDABILITÉ PAR RETOUR D'ÉTAT
%
% Methode 1 :
M = ctrb(A,B);
M = rank (M);
fprintf('\nLe rang de la matrice M = %d\n', M);
fprintf('La dimention de la matrice A = %d\n', length(A));

%Methode 2 :
[P,V]=eig(A);
M2 = inv(P)*B;
disp('Aucune range de inv(P) * B est nulle ');
disp(M2);

%Methode 3 : Vérification s'il y a annulation pôle-zéro
Bdelta = B(:,1);
Ddelta = D(:,1)
```

```
[num,den] = ss2tf(A, Bdelta, C, Ddelta);

[r, c] = size(num);
  for i = 1:r
    Zero = roots(num(i,:));
  end
 Zero
 Pole = roots(den)
 disp('Tous les pôles sont différents des zéros:')
 disp('Resultat : complètement commandable')
```

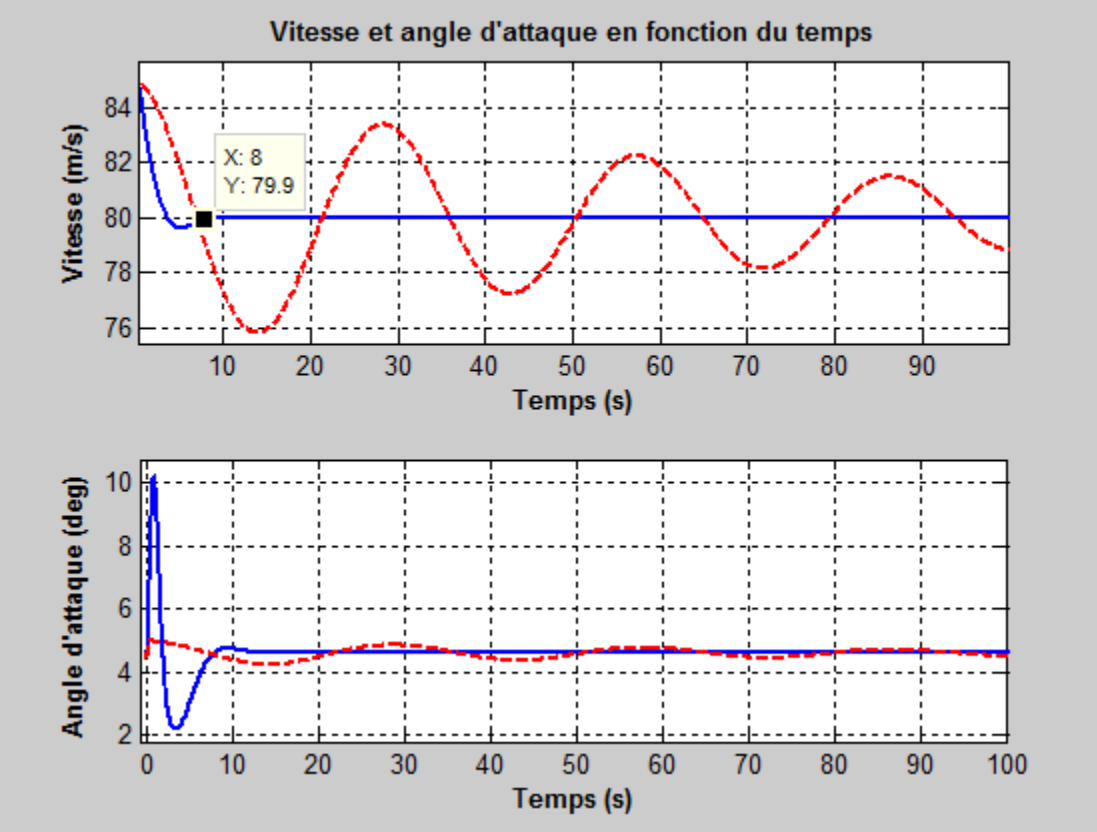
P3-3 OK

```
% Calcul de la matrice de gain K
K = place(A,Bdelta,Poled);
disp('Matrice K pour les poles desirees choisis ')
disp(K)
disp('Verification que mon calcul marche eig(A-B*K)= P')
tmp=eig(A-Bdelta*K);
disp(Poled);
disp(tmp);
K = [0.0317    3.2770   -0.6819    0.1107]
```

P3-4 : Validation du design

Mettre le switch case a 1 pour simuler le fichier « AVION_CONTROL » et on peut voir la différence entre le système asservie et non asservie

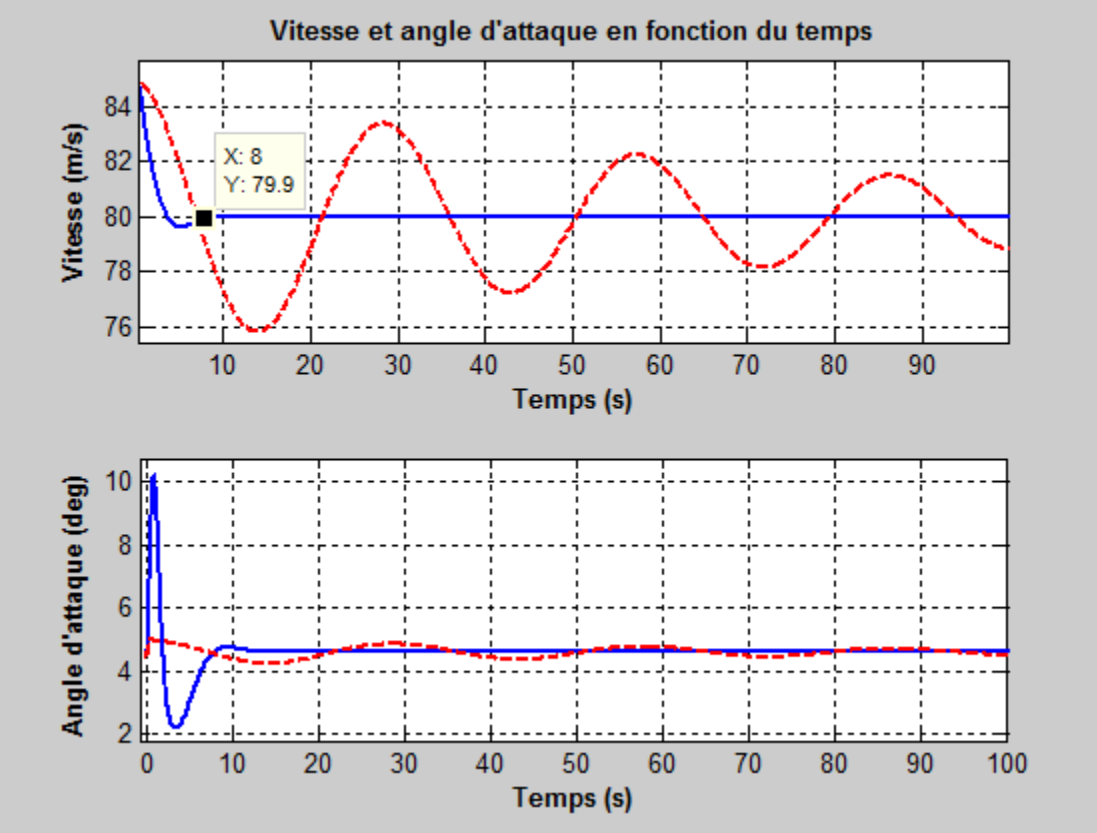
Le modèle AVION_CONTROL est dans la version 2016a donc je ne peux pas l'exécuter.



Vous pouvez vérifier le reste des simulations, mais on peut voir qu'on l'avion se stabilise à 80 km/h après seulement 8s.

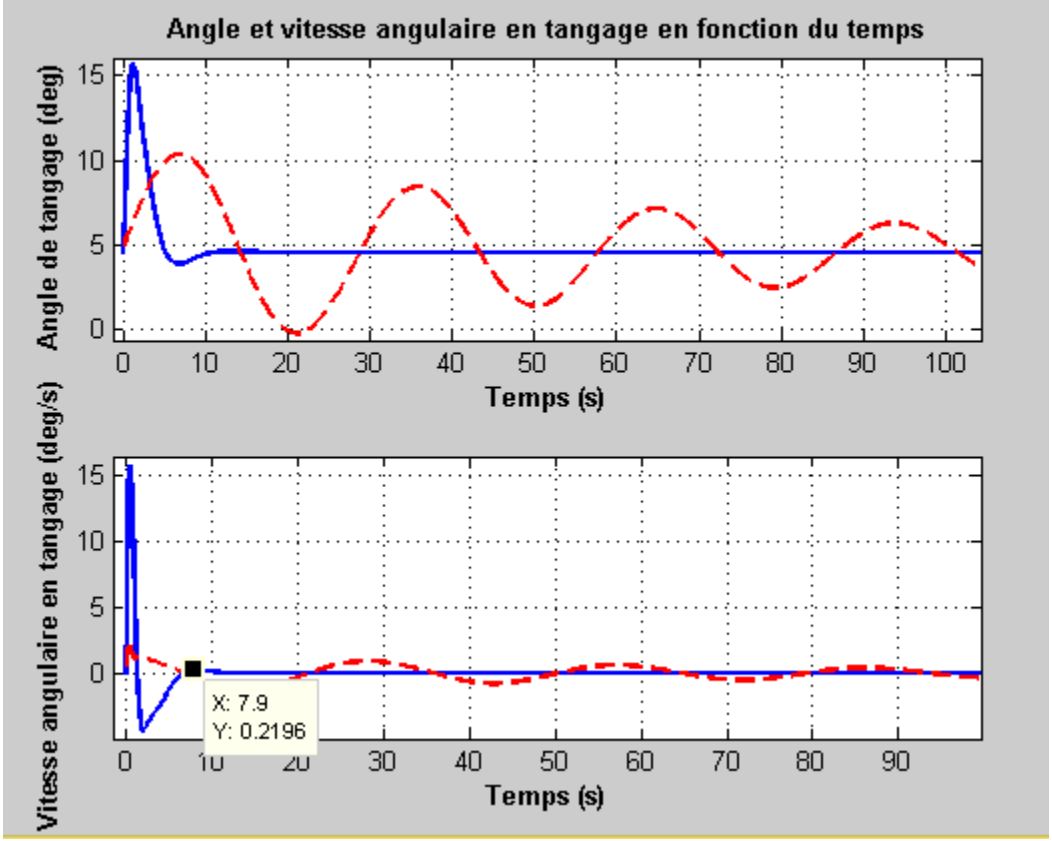
P3-5 : La comparaison des performances des deux systèmes :

a. La vitesse



On peut voir que le système asservie (en bleu) se stabilise à 79.9 Km/h après 8 secondes. Il respecte belle est bien les spécifications demandées, par contre pour atteindre ces spécification l'angle d'attaque passe à une valeur de 10 degrés.

b. Vitesse angulaire de tangage :



On peut voir aussi que 8s est le temps nécessaire pour atteindre les spécifications pour la vitesse angulaire.

PARTIE B : Design de l'observateur

P3-6

```
%P3-6 : Donner et justifier le choix des pôles de l'observateur Pe.
% On choisit de placer les poles de l'observateur a un facteur 2 des
poles
% du regulateur.
Facteur_Pe = 2;
Pe = Poled * Facteur_Pe;
```

Il faut que tous les pôles de l'observateur soient à gauche de ceux du régulateur. -2

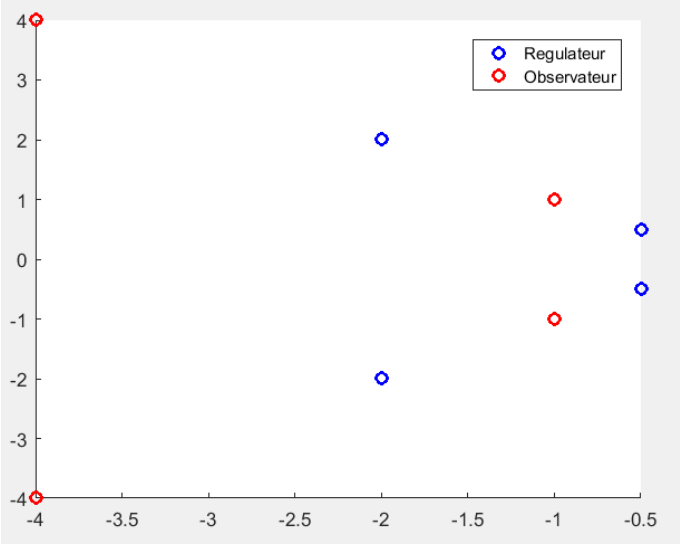


Figure 2

P3-7 : L'observabilité du système avec :

a. Obsv(A,C)

Il fallait déterminer l'observabilité avec la seule mesure de la vitesse de tangage donc avec Cmes = [0 0 0 1] et non pas avec toutes les sorties dans C. -2

```
'Ob = obsv(A,C);
unob = length(A)-rank(Ob);'
```

Qui donne un résultat de 0 dans notre cas c'est ce qui est attendu.

b. Les 2 fonction ctrb et obsv

On utilise le système suivant :

Il fallait déterminer l'observabilité par dualité avec ctrb, pas la commandabilité. La commande était donc: ctrb(A', C'). -2

```
Co = ctrb(A,B);
unco = length(A) - rank(Co);
Qui donne qu le system Est contrôlable et observable.
```

P3-8

```
%P3-8 : Calcul de la matrice Ke
CC = [0 0 0 1];
Ke = place(A',CC',Pe')';
disp('Matrice Ke pour les poles desirees choisis ')
disp(Ke)
disp('Verification que mon calcul marche eig(A-Ke*)= P')
tmp=eig(A-Ke*CC);
```

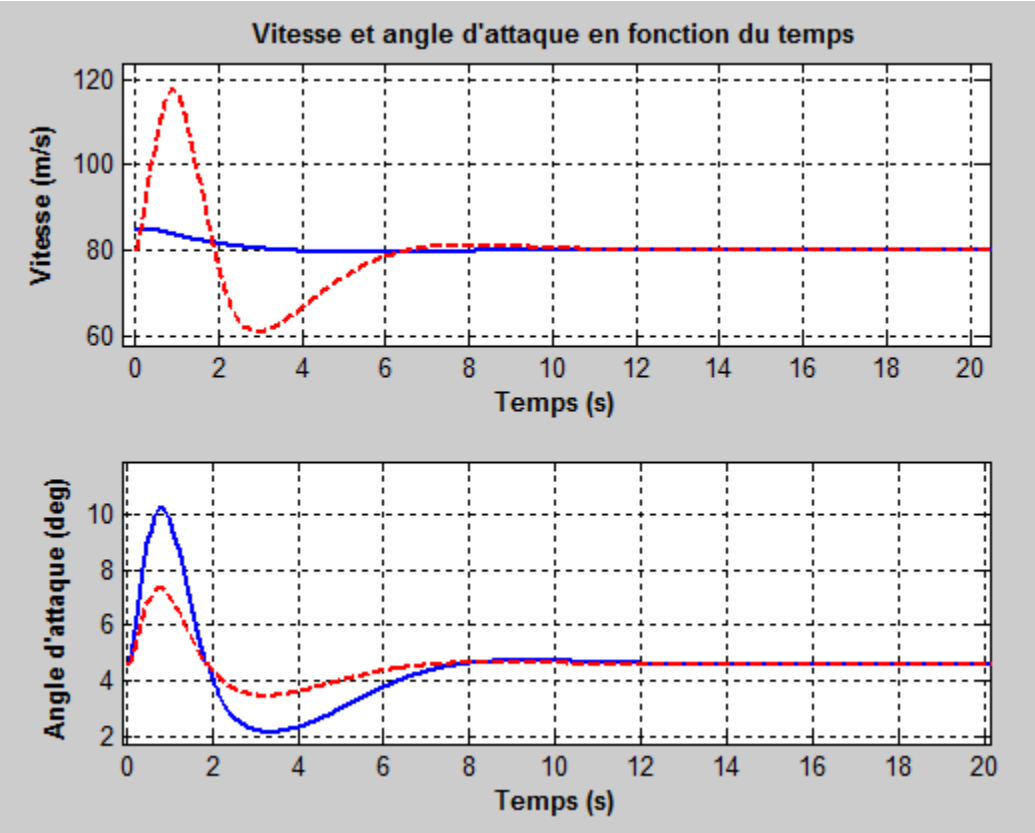
```
disp(Pe');
disp(tmp);
```

P3-9

```
Aobs = A - Ke*CC;
Bobs = [B(:,1) Ke];
Cobs = eye(4);
Dobs = [0 0 0 0; 0 0 0 0]';
```

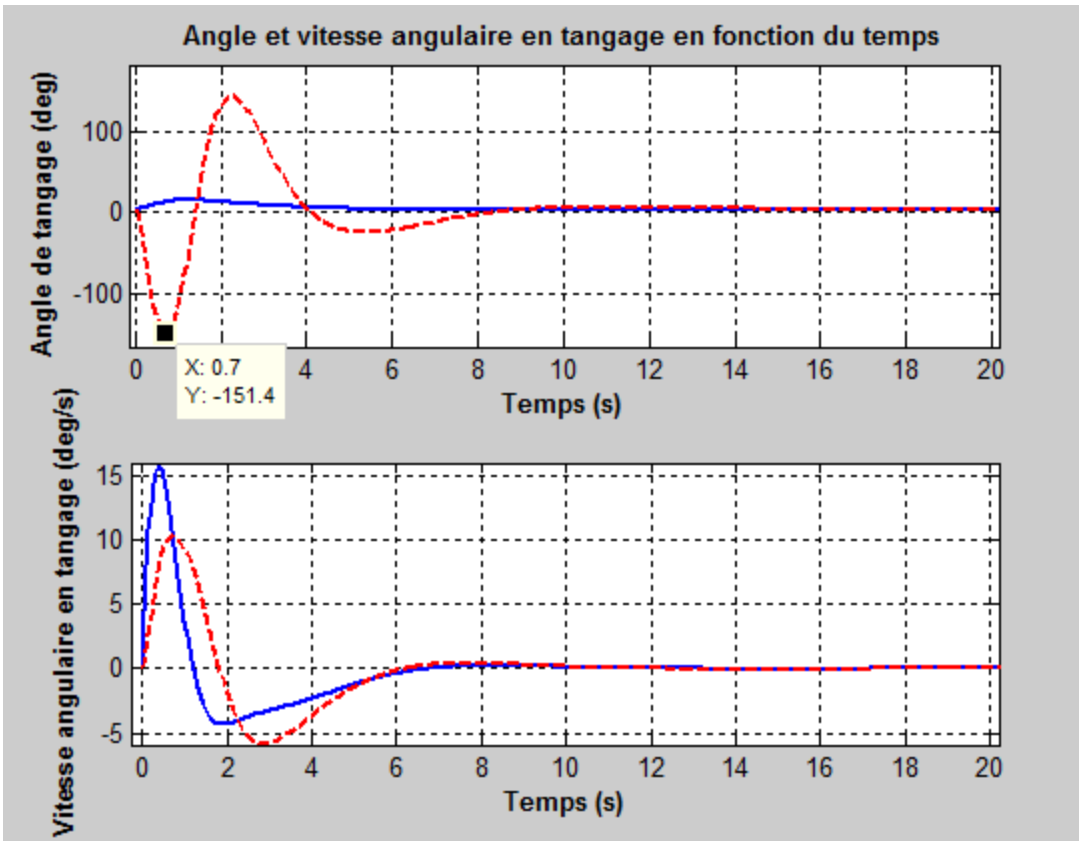
Le modèle AVION_OBSERV est dans la version 2016a donc je ne peux pas l'exécuter.

Les résultats :



On voit une bonne différence entre le système observé et réel puisque le système observé pense que l'avion est à 80 km/h au moment zéro et de cette façon il agit drastiquement pour corriger ce problème et de cette manière on remarque il atteint une vitesse de presque 120 km/h en deux secondes.
Par contre, l'angle d'attaque est plus petit durant la période de stabilisation.

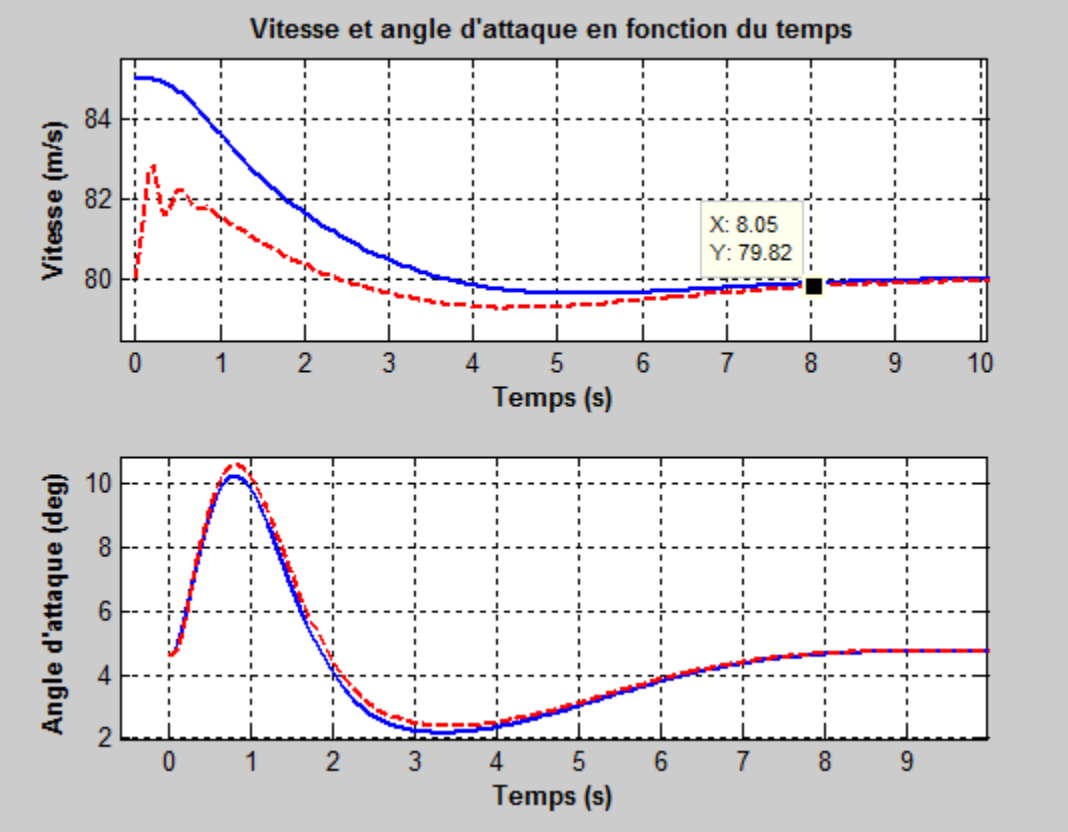
Pour ce qui est de l'angle de tangage, l'observateur passe d'un angle -150 degrés à 150 degrés en quelques secondes pour pouvoir assurer une stabilisation durant les 8 premier secondes. Ce qui dans un cas réel est impossible je crois. Certes, la vitesse en tangage est réduite mais le changement dans l'angle de tangage et trop important.


Angle et vitesse angulaire en tangage en fonction du temps
Angle de tangage (deg)
100
0
-100
X: 0.7
Y: -151.4
0
2
4
6
8
10
12
14
16
18
20
Temps (s)
Vitesse angulaire en tangage (deg/s)
15
10
5
0
-5
0
2
4
6
8
10
12
14
16
18
20
Temps (s)

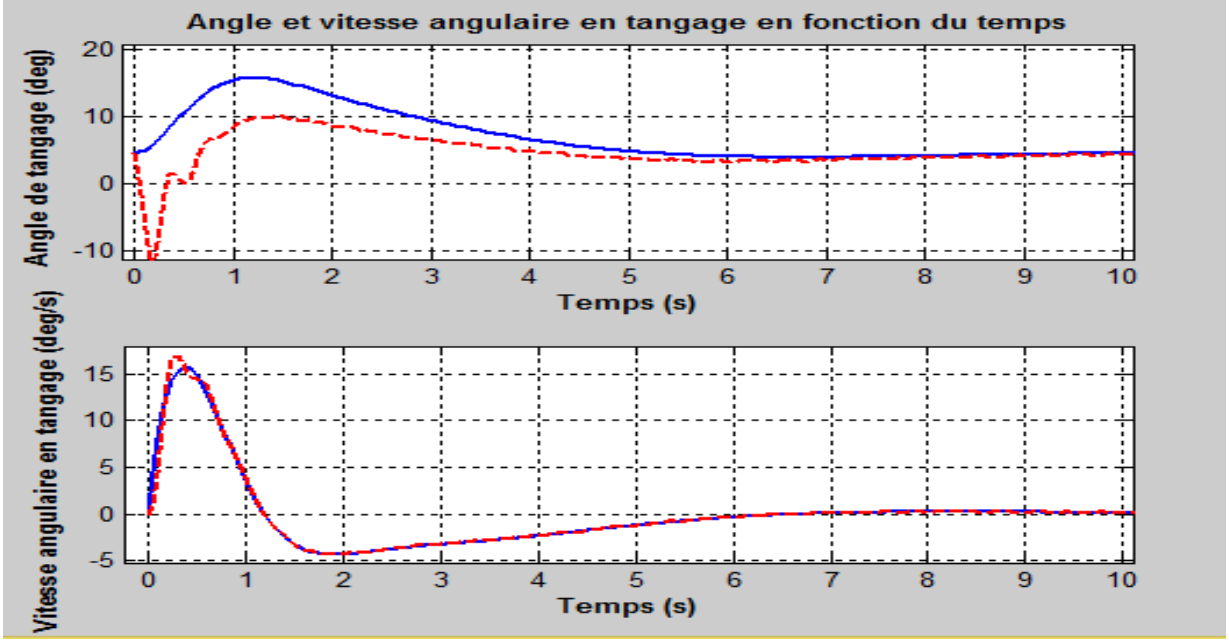
PARTIE C : Couplage de l'observateur et du régulateur

P3-10 : les graphiques demandés :

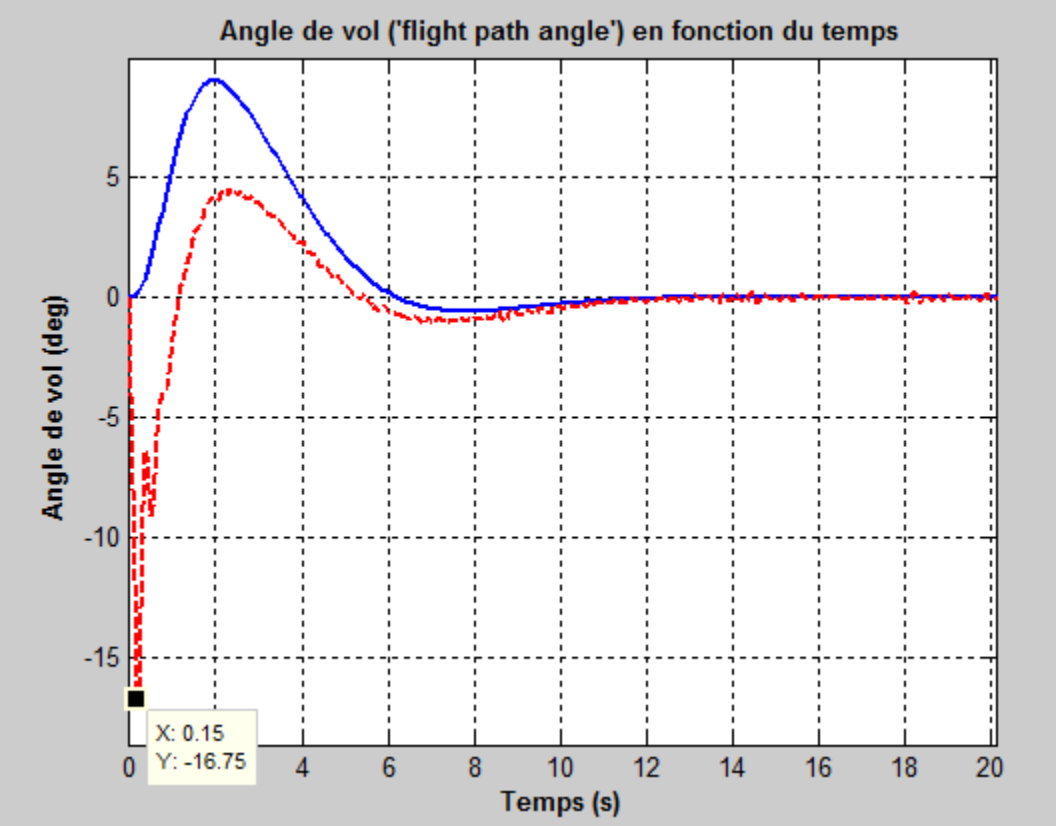
a. Vitesse & angle d'attaque en fonction du temps



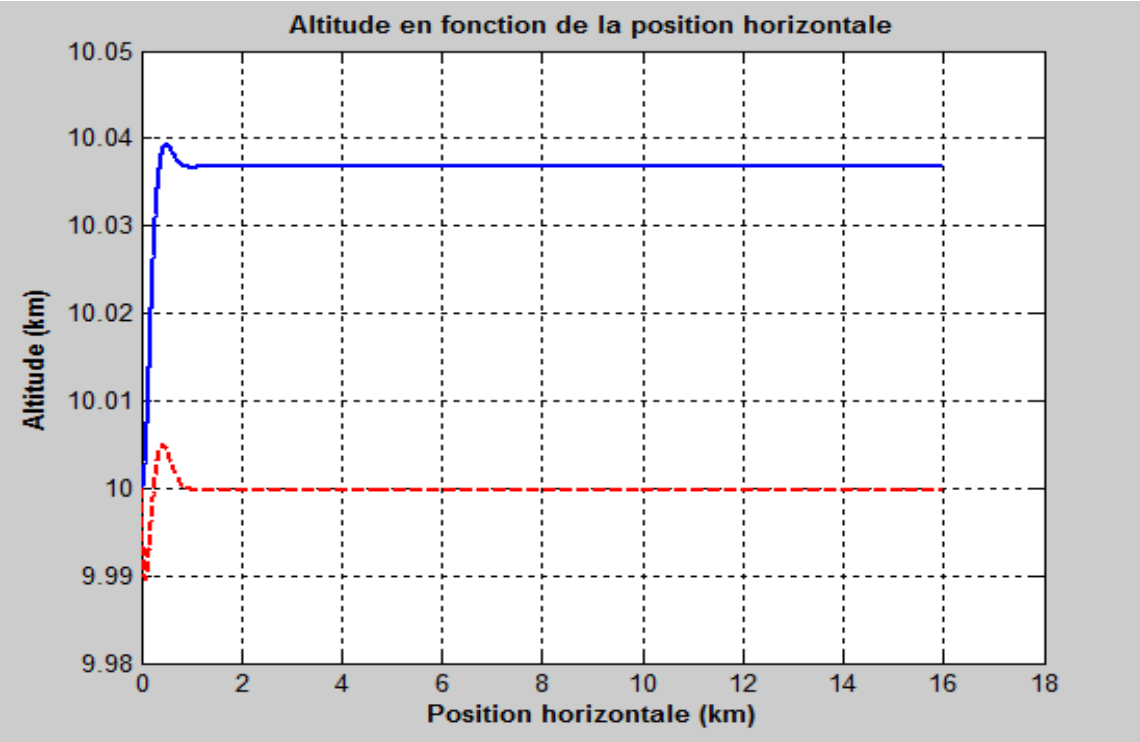
b. Angle et vitesse angulaire en tangage



c. Angle de vol



d. Altitude en fonction de la position horizontale



La boucle du régulateur utilise les vrais états du simulateur plutôt que les états générés par l'observateur.
C'est le but de ce dernier de fournir les états au régulateur. La structure n'est donc pas correcte.