TP1 : Appontage automatique d'un avion de chasse

...

Cohen --Urrutia Florian . Baconnais Olivier . Badanin Roman . Blanc Indie

Introduction

Un appontage est une manoeuvre qui consiste à un avion d'atterrir une plateforme. C'est une action délicate en raison des conditions dans lesquelles elle est exécutée : visibilité du pilote, mouvement du porte avion, conditions climatiques...

Dans ce TP, nous nous intéressons à la manoeuvre d'appontage d'un avion de chasse sur un porte avion. Les avions de chasse sont la catégorie d'avion qui enregistre le plus d'échec pour cette manoeuvre.

La manoeuvre d'appontage se déroule en 3 phases décrites plus loin.

On considère que l'orientation d'un avion en vol est déterminée par 3 axes fixes.

Axe 1 : L'axe longitudinale

La rotation de cet axe correspond au ROULIS.

On le définit par Ψ.

Axe 2 : L’axe vertical

La rotation de cet axe correspond au LACET.

On le définit par θ.

Axe 3 : L'axe latéral

La rotation de cet axe correspond au TANGAGE.

On le définit par φ.

Appontage - Phase 1 :

Le pilote place l'appareil en position d'approche. Il manoeuvre pour atteindre 3 objectifs :

1. Aligner l'axe longitudinal de l'avion à l'axe d'appontage
2. Ramener les paramètres de navigation à une configuration initiale
3. Enclencher le pilote d'appontage automatique

Appontage - Phase 2 :

Lancement de l'algorithme à bord du porte avion qui va générer en temps réel la trajectoire d'appontage. Il transmet aussi les paramètres suivants :

- $V\Psi$: la vitesse de l'avion suivant l'axe longitudinal de l'avion
- Ψ : angle d'inclinaison de l'axe longitudinal de l'avion vers le porte avion

Appontage - Phase 3 :

Fin de la manoeuvre d'appontage. Les paramètres de l'avion sont ramenés à la configuration suivante :

- $\Psi = 0$
- $V\Psi = 0$
- $\varphi = 0$
- $\theta = 0$

Cela signifie que l'avion apponte et finit par s'immobiliser sur la piste du porte avion, le long de l'axe d'appontage.

Le but de ce projet est de développer une application qui permet d'automatiser la manoeuvre d'appontage d'un avion de chasse sur un porte avion.

Partie 1

Nous avons recours à un développement basé sur une méthode formelle telle que la méthode B car les méthodes formelles permettent d'obtenir une très forte assurance de l'absence de bug dans les logiciels, c'est pourquoi elles sont utilisées pour les logiciels critiques. L'application pour automatiser la manœuvre d'appontage d'un avion de chasse sur un porte avion est une application critique qui requière 0 bug car une simple erreur pourrait avoir des conséquences catastrophiques. Elles permettent aussi de donner une spécification détaillée du système à développer, qui nous permettra de vérifier si la réalisation final est conforme à ce qui était demandé au départ.

Partie 2

Question 1 : Analyse du système d'appontage

Système **Pilote** :

Représente le système de contrôle automatique de l'appontage.

Etat :

- Enclenchement du pilote automatique : **PiloteAutomatique**
- Sous-système **Appontage** avec :
  - Fonction : **MaintienPlanApproche()**
  - Fonction : **MaintienTrajectoire()**

Fonction :

- **ActivationPA()** : Si PiloteAutomatique est à TRUE, on appelle les fonctions MaintienPlanApproche() et MaintienTrajectoire().

Sous-système **Contexte** :

Définit le contexte du fonctionnement global du système.

Etat :

- Axe longitudinal de l'avion : **roulis**
- Axe vertical de l'avion : **lacet**
- Axe latéral de l'avion : **tangage**
- Vitesse de l'avion : **vitesse**

Fonction :

- **Initialisation(roulis, lacet, tangage, vitesse)** : Met les variables à 0 et ne retourne rien.

Sous-système **Appontage** :

Système opérationnel chargé de coordonner le maintien et la trajectoire de descente de l'avion dans le plan d'approche.

Etat :

- Sous-système **PlanApproche** avec :
  - Fonction : **EnvoiRoulis()**
  - Fonction : **EnvoiLacet()**
- Sous-système **Trajectoire** avec :
  - Toutes les fonctions de ce sous-système

Fonctions :

- **MaintienPlanApproche()** : Appelle les fonctions EnvoiRoulis() et EnvoiLacet().
- **MaintienTrajectoire()** : Appelle toutes les fonctions du sous-système Trajectoire.

Sous-système **PlanApproche** :

Garantit le maintien de l'avion dans le plan d'approche.

Etat :

- Sous-système **CapteurPlan** avec :
  - Fonction : **RecupRoulis()**
  - Fonction : **RecupLacet()**
- Sous-système **RegulateurPlan** avec :
  - Fonction : **ModifRoulis(valeur)**
  - Fonction : **ModifLacet(valeur)**

Fonctions :

- **EnvoiRoulis()** : Appelle de la fonction ModifRoulis(valeur), valeur égale à (0 - le retour de RecupRoulis()).
- **EnvoiLacet()** : Appelle la fonction ModifLacet(valeur), valeur égale à (0 - le retour de RecupLacet()).

Sous-système **Trajectoire** :

Contrôle la trajectoire de descente de l'avion maintenu dans le plan d'approche.

Etat :

- Tangage mesuré par le capteur : **tangage**
- Vitesse mesurée par le capteur : **vitesse**
- Tangage mesuré par l'aiguilleur : **tangageA**
- Vitesse mesurée par l'aiguilleur : **vitesseA**
- Sous-système **CapteurTrajectoire** avec :
  - Fonction : **RecupTangage()**
  - Fonction : **RecupVitesse()**
- Sous-système **Aiguillage** avec :
  - Fonction : **CalculTangage(valeur)**
  - Fonction : **CalculVitesse(valeur)**
- Sous-système **RegulateurTrajectoire** avec :
  - Fonction : **ModifTangage(valeur)**
  - Fonction : **ModifVitesse(valeur)**

Fonctions :

- **TangageCapteur()** : Met dans tangage le retour de la fonction RecupTangage().
- **VitesseCapteur()** : Met dans vitesse le retour de la fonction RecupVitesse().
- **AiguilTangage()** : Met dans tangageA le retour de la fonction CalculTangage(tangage).
- **AiguilVitesse()** : Met dans vitesseA le retour de la fonction CalculVitesse(vitesse).
- **EnvoiTangage()** : Appelle la fonction ModifTangage(tangageA).
- **EnvoiVitesse()** : Appelle la fonction ModifVitesse(vitesseA).

Sous-système **Aiguilleur** :

Reçoit les données réelles sur les paramètres vitesse et tangage de l'avion et retourne les données pour corriger en temps réel la trajectoire d'appontage.

Fonctions :

- **CalculTangage(valeur)** : calcul le tangage nécessaire à l'avion durant la manoeuvre grâce au tangage actuel passé en paramètres. Retourne le tangage nécessaire.
- **CalculVitesse(valeur)** : calcul la vitesse nécessaire à l'avion durant la manoeuvre grâce à la vitesse actuelle passée en paramètres. Retourne la vitesse nécessaire.

Sous-système **BASIC_IO** :

Gère les entrées/sorties du simulateur.

Etat :

- Données en entrée du simulateur : **roulis**, **lacet**, **tangage**, **vitesse**
- Données en sortie du simulateur : **roulis**, **lacet**, **tangage**, **vitesse**

Fonctions :

- **RecepDonnee(donnee)** : récupère la donnée passée en paramètre à l'entrée du simulateur.
- **EnvoiDonnee(donnee, nombre)** : envoi nombre à la donnée passée en paramètre par la sortie du simulateur.

Sous-système **CapteurPlan** :

Récupère les données réelles fournies par les capteurs roulis et lacet de l'avion.

Etat :

- Sous-système **CapteurRoulis** avec :
  - Fonction : **MesureRoulis()**
- Sous-système **CapteurLacet** avec :
  - Fonction : **MesureLacet()**

Fonctions :

- **RecupRoulis()** : Appelle la fonction MesureRoulis(). Retourne le roulis réceptionné.
- **RecupLacet()** : Appelle la fonction MesureLacet(). Retourne le lacet réceptionné.

Sous-système **CapteurRoulis** :

Mesure le roulis de l'avion.

Etat :

- Sous-système **BASIC_IO** avec :
  - Donnée en entrée du simulateur : **roulis**
  - Fonction : **RecepDonnee(donnee)**

Fonction :

- **MesureRoulis()** : Récupère la donnée roulis du simulateur. Appelle la fonction RecepDonnee(roulis). Retourne la valeur du roulis mesurée.

Sous-système **CapteurLacet** :

Mesure le lacet de l'avion.

Etat :

- Sous-système **BASIC_IO** avec :
  - Donnée en entrée du simulateur : **lacet**
  - Fonction : **RecepDonnee(donnee)**

Fonction :

- **MesureLacet()** : Récupère la donnée lacet du simulateur. Appelle la fonction RecepDonnee(lacet). Retourne la valeur du lacet mesurée.

Sous-système **CapteurTrajectoire** :

Récupère les données réelles sur la vitesse et le tangage de l'avion.

Etat :

- Sous-système **CapteurTangage** avec :
  - Fonction : **MesureTangage()**
- Sous-système **CapteurVitesse** avec :
  - Fonction : **MesureVitesse()**

Fonctions :

- **RecupTangage()** : Appelle la fonction MesureTangage(). Retourne le tangage reçu.
- **RecupVitesse()** : Appelle la fonction MesureVitesse(). Retourne la vitesse reçue.

Sous-système **CapteurTangage** :

Mesure l'angle de tangage réel de l'avion.

Etat :

- Sous-système **BASIC_IO** avec :
  - Donnée en entrée du simulateur : **tangage**
  - Fonction : **RecepDonnee(donnee)**

Fonction :

- **MesureTangage()** : Récupère la donnée tangage du simulateur. Appelle la fonction RecepDonnee(tangage). Retourne la valeur du tangage mesurée.

Sous-système **CapteurVitesse** :

Mesure la vitesse réelle de l'avion.

Etat :

- Sous-système **BASIC_IO** avec :
  - Donnée en entrée du simulateur : **vitesse**
  - Fonction : **RecepDonnee(donnee)**

Fonction :

- **MesureVitesse()** : Récupère la donnée vitesse du simulateur. Appelle la fonction RecepDonnee(vitesse). Retourne la valeur de la vitesse mesurée.

Sous-système **RegulateurPlan** :

Corrige en temps réel les paramètres (lacet-roulis) de sorte à maintenir l'avion dans le plan d'approche.

Etat :

- Sous-système **RegulateurRoulis** avec :
  - Fonction : **CorrRoulis(valeur)**
- Sous-système **RegulateurLacet** avec :
  - Fonction : **CorrLacet(valeur)**

Fonctions :

- **ModifRoulis(valeur)** : Appelle CorrRoulis(valeur).
- **ModifLacet(valeur)** : Appelle CorrLacet(valeur).

Sous-système **RegulateurRoulis** :

Corrige le paramètre roulis de sorte que l'avion soit maintenu dans le plan d'approche.

Etat :

- Sous-système **BASIC_IO** avec :
  - Donnée en sortie du simulateur : **roulis**
  - Fonction : **EnvoiDonnee(donnee, nombre)**

Fonction :

- **CorrRoulis(valeur)** : Appelle EnvoiDonnee(roulis, valeur).

Sous-système **RegulateurLacet** :

Corrige le paramètre lacet de sorte que l'avion soit maintenu dans le plan d'approche.

Etat :

- Sous-système **BASIC_IO** avec :
  - Donnée en sortie du simulateur : **lacet**
  - Fonction : **EnvoiDonnee(donnee, nombre)**

Fonction :

- **CorrLacet(valeur)** : Appelle EnvoiDonnee(lacet, valeur).

Sous-système **RegulateurTrajectoire** :

Commande les régulateurs de vitesse et de tangage afin de corriger la trajectoire de l'avion.

Etat :

- Sous-système **RegulateurTangage** avec :
  - Fonction : **CorrTangage(valeur)**
- Sous-système **RegulateurVitesse** avec :
  - Fonction : **CorrVitesse(valeur)**

Fonctions :

- **ModifTangage(valeur)** : Appelle CorrTangage(valeur).
- **ModifVitesse(valeur)** : Appelle CorrVitesse(valeur).

Sous-système **RegulateurVitesse** :

Corrige la vitesse de l'avion en appliquant la consigne de l'aiguilleur automatique.

Etat :

- Sous-système **BASIC_IO** avec :
  - Donnée en sortie du simulateur : **vitesse**
  - Fonction : **EnvoiDonnee(donnee, nombre)**

Fonction :

- **CorrVitesse(valeur)** : Appelle EnvoiDonnee(vitesse, valeur).

Sous-système **RegulateurTangage** :

Corrige le tangage de l'avion en appliquant la consigne de l'aiguilleur automatique.

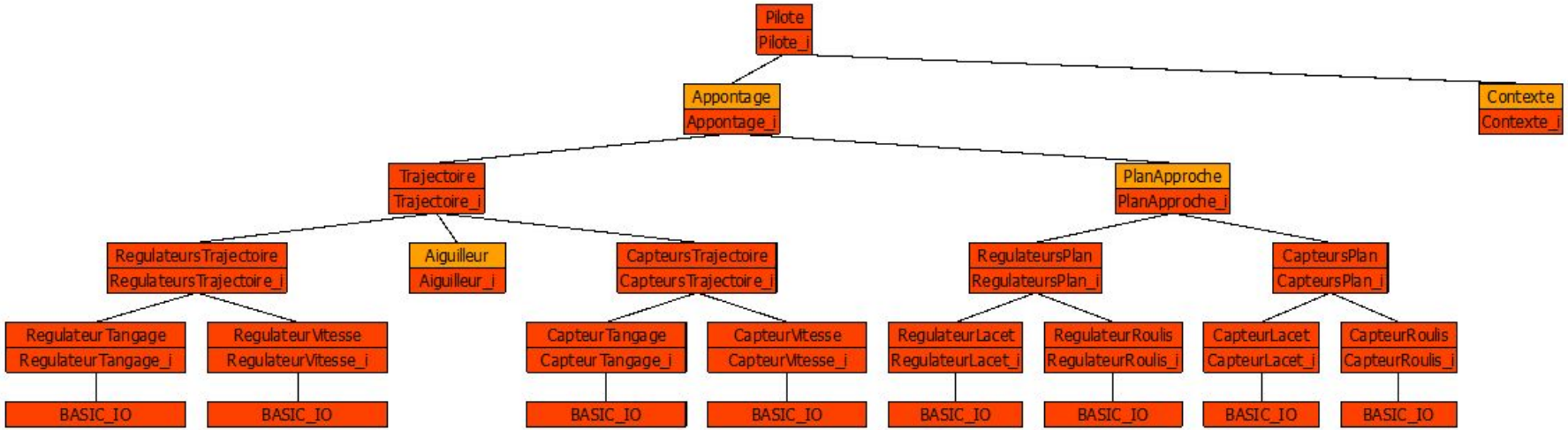
Etat :

- Sous-système **BASIC_IO** avec :
  - Donnée en sortie du simulateur : **tangage**
  - Fonction : **EnvoiDonnee(donnee, nombre)**

Fonction :

- **CorrTangage(valeur)** : Appelle EnvoiDonnee(tangage, valeur).

Question 2 : Modélisation en B du système



Question 3 : Validation du modèle précédent

Question 4 : Génération du code source C++ du logiciel