« Systèmes à événements discrets »
TP : Le traitement numérique des signaux pour le contrôle du tangage du drone didactique D2C

Cette fiche détaille le travail à réaliser par le *groupe 3* dans ce TP :

Préalable :
L'objectif général du travail est l'amélioration progressive des performances du système de contrôle du tangage du drone ;
→ lire la description générale des activités du TP, p6 de la ***fiche « TP-systemes-evenements-discrets-presentation »*** pour comprendre le rôle de chaque groupe dans cette première phase, et pour comprendre l'utilité des différentes phases.

La séance de travail comporte trois phases qui sont à peu près de 30 minutes chacune ; chaque groupe prendra soin de réaliser des copies d'écran, pour présenter ses conclusions.
Des réponses écrites seront données pour les résultats qui ne pourront apparaître sur les copies d'écran.
A la fin de chaque phase, une synthèse de 5 minutes permettra à chaque groupe de prendre connaissance des résultats obtenus par les deux autres groupes.

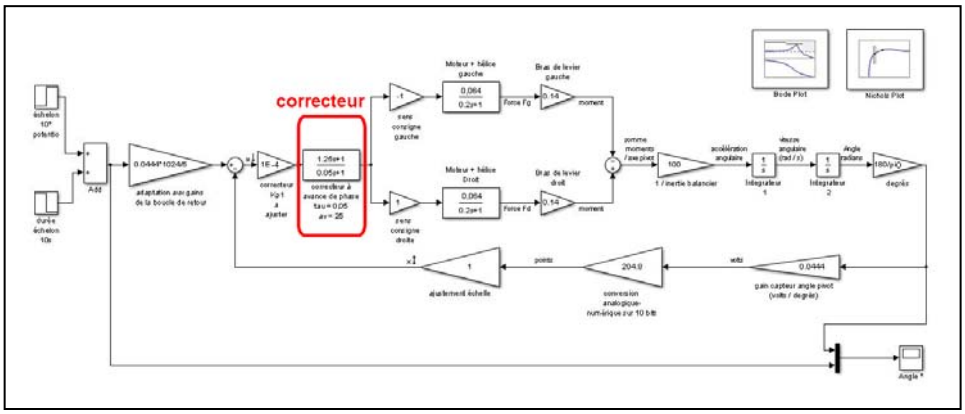
Important : les travaux proposés conduiront à modifier des fichiers de simulation (exploités sous Matlab-Simulink ou Scilab-Xcos) ou des fichier de programmation (exploités dans l'interface de programmation Arduino) ; il faudra dès leur ouverture, enregistrer ces fichiers avec votre nom, avant toute modification.

Phase 1 : « l'étude de l'asservissement à une seule boucle de position »

Dans cette phase, le travail du groupe est centré autour de la mise en place de l'équation de récurrence du correcteur à avance de phase qui intervient dans la boucle d'asservissement ; l'équation sera implantée dans le programme de commande de tangage du drone didactique, pour la réalisation d'expérimentations.

Travail 3-1-1 : démonstration de commande en position
→ Tester la commande en position angulaire à l'aide des commandes du pupitre pour le cas où la commande est réalisée avec le micro-contrôleur DSPic et les réglages proposés par défaut à partir de la ***« Fiche_pilotage-D2C-par-DSPic »*** ;
→ Faire la démonstration aux autres groupes, en réalisant quelques allers-retours entre -30° et + 30°.

Travail 3-1-2 : Ecriture de l'équation de récurrence du correcteur
Le ***groupe 2*** a la charge de réaliser un travail de simulation logicielle pour valider l'utilisation d'un correcteur à avance de phase dans l'objectif d'obtenir un système stable en tangage (schéma-bloc ci-contre).



Aéroparc Saint Martin – 12 rue de Caulet – 31300 TOULOUSE – ☎ : + 33 (0)5 62 88 72 72 📖 : + 33 (0)5 62 88 72 79
Site internet : www.dmseducation.com Email : info@dmseducation.com

la ***« Fiche_equations-de-recurrence »*** fournit la méthode pour obtenir l'équation de récurrence qui va être utilisée pour programmer ce correcteur.

$$S_n(avance\ de\ phase) = \frac{1}{Te+\tau}\left[\tau.S_{n-1} + (Te + a.\tau).E_n - a.\tau.E_{n-1}\right]$$

→ démontrer l'équation de récurrence de ce correcteur à avance de phase proposée dans la fiche (on utilisera la méthode des rectangles).

Travail 3-1-3 : implémentation du correcteur dans le programme de commande
Ouvrir le programme ***« _1_asservit_1boucle_position_AVPH_a_completer.ino »***, l'enregistrer avec votre nom ; il a été préparé de la façon suivante :
(Nota : le numéro de ligne où est positionné le curseur s'affiche en bas à gauche de l'écran) :

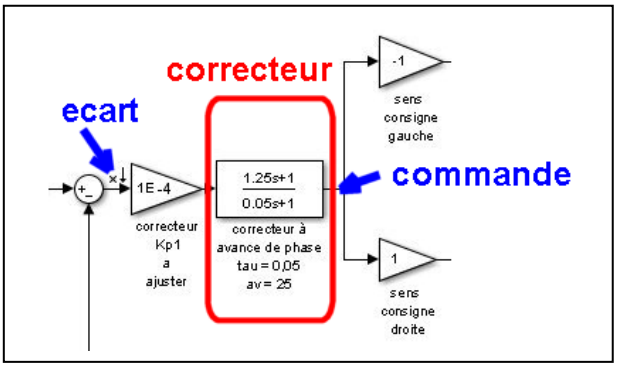
- les premières lignes (25 à 30) définissent les variables pour le correcteur :

```
//*************** variables pour le correcteur ****************************************
long periode_echantillonnage = 20;    // frequence de mise à jour du correcteur (en millisecondes) ;
float precedente_entree = 0;          // variable globale pour la fonction AVPH (correcteur à avance de phase)
float precedente_sortie = 0;          // variable globale pour la fonction AVPH (correcteur à avance de phase)
float kp1 = 0;      // valeur à régler
float av = 0;      // valeur à régler
float tau = 0;     // reglage du correcteur à avance de phase
```

- Pour écrire le correcteur, on utilise une fonction nommée « AVPH_out » ;
les lignes suivantes (33 à 41) permettent d'écrire la fonction du correcteur à avance de phase « AVPH_out » qui sera appelée plus loin pour le calcul de la commande des moteurs.

```
//*************************************** la fonction pour le correcteur*************************
float AVPH_out(float entree, long dt) {
  float sortie;
  // cas de la méthode des rectangles pour exprimer le correcteur à avance de phase :
  sortie = // c'est ici qu'il faut écrire l'équation de récurrence du correcteur à avance de phase
  sortie = constrain(sortie,-1023,1023);     // bornage des valeurs
  precedente_entree = entree;        // pour le prochain calcul
  precedente_sortie = sortie;       // pour le prochain calcul
  return sortie;                 //retourne la sortie du correcteur pour exploitation de la fonction
}
```

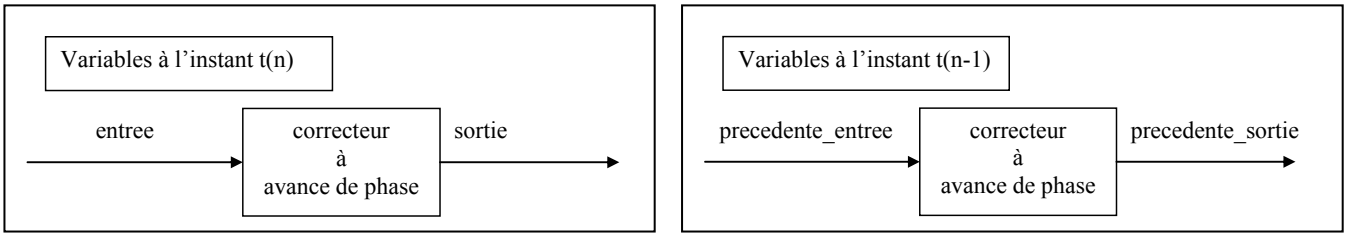
- la boucle « loop » qui se répète indéfiniment est écrite entre les lignes 89 et 142.
L'utilisation de la fonction « AVPH_out » est faite à la ligne 99, où la variable « commande » récupère la « sortie » de la fonction correcteur « AVPH_out », pour laquelle l'entrée est « Kp1 x ecart » (schéma ci-contre).




```
// ******************************   asservissement ***********************************
//******************traitement de la commande du tangage
mesure_potentio_droit = analogRead(broche_potentio_droit);                              // acquisition de la tension du potentio 0 à 5V sur 10 bits
ecart = float(mesure_potentio_droit - 512) - float(mesure_angle_pivot - mesure_angle_pivot_nul);   // calcul de l'écart entre consigne et mesure
commande = AVPH_out(kp1*ecart, periode_echantillonnage);                                 // application de la fonction du correcteur à avance de pha
commande_unitarisee = commande / 1023.0f;              // réechelonnement de la commande de tangage : valeur de la commande (-1023 à 1023) ramenée sur une échel
```


Société DMS
Aéroparc Saint Martin – 12 rue de Caulet – 31300 TOULOUSE – ☎ : + 33 (0)5 62 88 72 72 📖 : + 33 (0)5 62 88 72 79
Site internet : www.dmseducation.com Email : info@dmseducation.com

Le travail à réaliser porte sur l'écriture mise en place à l'intérieur de la fonction « AVPH_out ».
On utilise dans cette fonction, les variables suivantes aux deux instants t(n) et t(n-1) :



→ Réécrire l'équation de récurrence du correcteur en utilisant les variables d'entrée-sortie ci-dessus pour exprimer la sortie à l'instant t(n).

Nota : il ne faut pas d'accents sur les variables !
Il faudra utiliser aussi la période d'échantillonnage « dt » sous la forme « float(dt)/1000.0f » pour tenir compte du fait qu'elle est exprimée en millisecondes et que l'on travaille en secondes, et aussi du fait qu'elle doit être exprimée en type « flottants » pour ce programme en langage C.

→ Mettre en place l'équation de récurrence du correcteur à la ligne 36 du programme ***« _1_asservit_1boucle_position_AVPH_a_completer.ino »*** (enregistré avec votre nom), pour permettre de générer la commande de pilotage des moteurs ;

```
//************************************** la fonction pour le correcteur**************************************
float AVPH_out(float entree, long dt) {
  float sortie;
  // cas de la méthode des rectangles pour exprimer le correcteur à avance de phase :
  sortie = // c'est ici qu'il faut écrire l'équation de récurrence du correcteur à avance de phase
  sortie = constrain(sortie,-1023,1023);      // bornage des valeurs
  precedente_entree = entree;          // pour le prochain calcul
  precedente_sortie = sortie;         // pour le prochain calcul
  return sortie;                               //retourne la sortie du correcteur pour exploitation de la fonction
}
```

→ Récupérer auprès du groupe 2, les valeurs adaptées du correcteur et les placer aux lignes 28 à 30.

```
//************** variables pour le correcteur ***************************************
long periode_echantillonnage = 20;     // frequence de mise à jour du correcteur (en millisecondes) ;
float precedente_entree = 0;           // variable globale pour la fonction AVPH (correcteur à avance de phase)
float precedente_sortie = 0;           // variable globale pour la fonction AVPH (correcteur à avance de phase)
float kpl = 0;       // valeur à régler
float av = 0;       // valeur à régler
float tau = 0;      // reglage du correcteur à avance de phase
```

→ Téléverser le programme dans la mémoire du microcontrôleur de l'Arduino.

Travail 3-1-4 : pilotage du drone didactique
→ Vérifier la mise en place des câblages entre l'Arduino-box et les moteurs du système D2C (voir la ***« fiche_connexions-pupitre-arduino-box »***) et tester la programmation en mettant progressivement les moteurs en route jusqu'à 20% avec le bouton gauche, puis la commande d'inclinaison avec le bouton droit.
(Nota : en cas de non fonctionnement, si le déboogage s'éternise après vérification des câblages, on téléversera le programme corrigé ***« _1_asservit_1boucle_position_AVPH_solution_.ino »).***

Aéroparc Saint Martin – 12 rue de Caulet – 31300 TOULOUSE – ☎ : + 33 (0)5 62 88 72 72 🕮 : + 33 (0)5 62 88 72 79
Site internet : www.dmseducation.com Email : info@dmseducation.com

→ Réaliser une manipulation approximative d'un échelon de 10° avec le bouton droit, pour montrer que le critère de rapidité du cahier des charges n'est pas respecté. Commenter ces résultats du point de vue des autres critères du cahier des charges.
Nota : Le paragraphe 4 de la « ***Fiche_pilotage-D2C-par-DSPic*** » explique comment réaliser des acquisitions de ces expérimentations.

Conclusion de la phase 1 :
Rédiger, dans le compte-rendu commun aux trois groupes, une courte synthèse des démarches réalisées et des résultats obtenus, en respectant (ou en créant) l'organisation logique de ce compte-rendu de façon à présenter les détails la fonction "traiter" réalisée par le microcontrôleur.
Une synthèse présentée sous forme de synoptique sera bien appréciée.

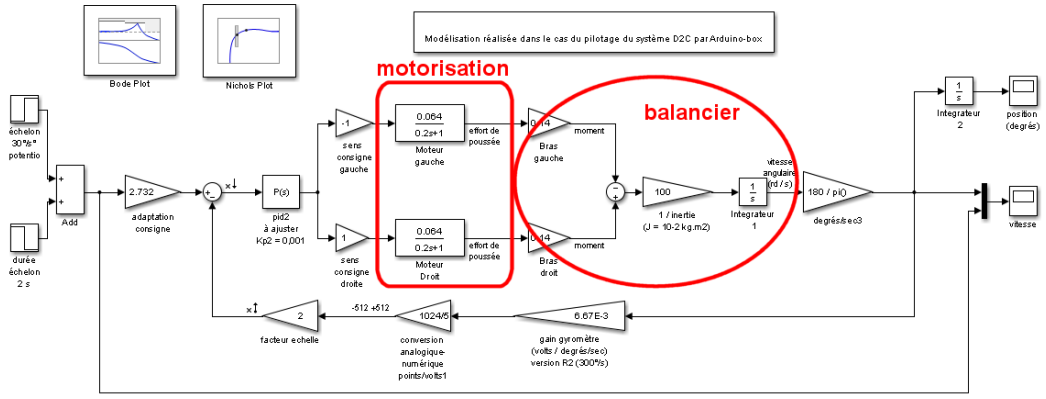
Société DMS
Aéroparc Saint Martin – 12 rue de Caulet – 31300 TOULOUSE – ☎ : + 33 (0)5 62 88 72 72 🕮 : + 33 (0)5 62 88 72 79
Site internet : www.dmseducation.com Email : info@dmseducation.com

Phase 2 : « l'étude de l'asservissement en boucle de vitesse »

Dans cette phase, le travail du groupe est centré sur la modélisation de la boucle d'asservissement en en vitesse du drone didactique D2C. Une simulation logicielle permettra de valider les coefficients du correcteur PID filtré utilisé pour optimiser cette boucle.

Travail 2-1-1 : analyse du schéma-bloc

→ Ouvrir le fichier ***« simu_D2C_Arduino_1boucle_vitesse_a_ajuster »*** avec le logiciel de simulation du laboratoire (Matlab-Simulink ou Scilab-Xcos) ; l'enregistrer avec votre nom.



a) Analyse de la boucle de retour :

→ ***vérifier auprès du groupe 2, que les coefficients de la boucle de retour correspondent au matériel et aux solutions d'acquisition choisis pour le gyromètre*** ;

b) Analyse de la chaîne directe :

→ ***Obtenir auprès du groupe 2, la justification de la fonction de transfert de la motorisation*** ;

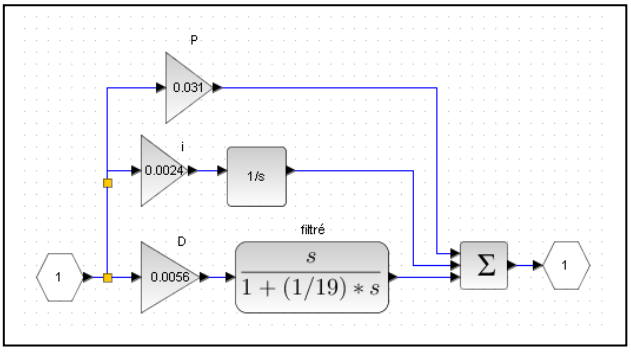
→ Indiquer quelle est l'équation qui a été utilisée pour modéliser la zone désignée par « Balancier » sur le schéma-bloc ci-dessus, et comment elle a été obtenue.

→ Le correcteur proposé est un correcteur proportionnel de valeur 0,001 ; effectuer des simulations pour trouver le gain optimal qui permet de valider le critère de stabilité du cahier des charges, tout en optimisant la rapidité.

→ montrer que ce correcteur proportionnel n'est pas suffisant pour valider le critère de rapidité du cahier des charges.

Pour obtenir la rapidité souhaitée, on propose dans la suite, d'utiliser un correcteur dérivé ;
Et pour obtenir une bonne résistance aux perturbations sur la vitesse angulaire (turbulences dans le cas du drone réel, frottements de la liaison pivot dans le cas du drone didactique), on ajoute un correcteur intégral.

L'ensemble réalise un correcteur PID filtré comme le montre le schéma-bloc ci contre.



Des simulations non récapitulées ici ont permis d'obtenir les valeurs optimisées suivantes :
KP2 = 0,031 ; Ki2 = 0,0024 ; Kd2 = 0,0056 ; N = 19 (N : pulsation de coupure du filtre dérivé).

→ Montrer que le correcteur PID filtré, avec ces coefficients permet de respecter tous les critères du cahier des charges concernant l'asservissement en vitesse de tangage.

Aéroparc Saint Martin – 12 rue de Caulet – 31300 TOULOUSE – ☎ : + 33 (0)5 62 88 72 72 🕮 : + 33 (0)5 62 88 72 79
Site internet : www.dmseducation.com Email : info@dmseducation.com

Phase 3 : Etude de l'asservissement à deux boucles imbriquées ;

L'objectif du travail est d'étudier l'influence du choix de la période d'échantillonnage sur le comportement du système :

- Il faudra vérifier que le calculateur du microcontrôleur Arduino effectue les traitements dans un temps inférieur à la valeur de la période d'échantillonnage choisie.
- il faudra vérifier expérimentalement que la durée de l'échantillonnage choisie ne perturbe pas la stabilité du système.

Travail 3-3-1 : mesure du temps de traitement du code ; comparaison avec la période d'échantillonnage choisie ;

Se placer sous l'interface de programmation Arduino, et ouvrir et enregistrer avec votre nom, le programme **« _3_asservit_2_boucles_PIDF1_PIDF2_solution_avec_retards_optim.ino »** ;

Il s'agit d'une programmation avec deux boucles d'asservissement et deux correcteurs PIDfiltrés complets optimisés par simulation numérique.

Le programme est conçu de la manière suivante :

(Nota : le numéro de ligne où se trouve le curseur s'affiche en bas à gauche)

Lignes 1 à 117 : pour déclarer les variables et définir les fonctions PID ;

La ligne 24 fixe la période d'échantillonnage à 20 millisecondes.

Lignes 118 à 125 : fonction lancée une fois au démarrage du programme ;

Lignes 128 à 194 : boucle principale du programme qui se répète indéfiniment.

A la ligne 129 on utilise la **variable « temps_debut_cycle »** au début de cette boucle pour enregistrer le temps au début du traitement (**c'est la fonction « micros() » qui retourne le temps courant en microsecondes**)

```
//***********************************
//boucle principale
void loop(){
129 temps_debut_cycle = micros();
```

De la ligne 186 à 190, on utilise une procédure d'attente (do–while) pour attendre la durée de la période d'échantillonnage, avant de relancer la boucle principale.

Pendant cette attente, on détermine en permanence la **variable « temps_cycle_calcule » (ligne 188)** et on termine la procédure d'attente si cette variable devient supérieure à la période d'échantillonnage exprimée en microsecondes.

```
*/
185 temps_fin_du_calcul = 0;  // c'est ici qu'il faut remplcer le 0 par le calcul du temps de traitement
/* ************* attente du temps de cycle fixé  ********************** */
do {
188 temps_cycle_calcule = micros() - temps_debut_cycle;
} while (temps_cycle_calcule < (periode_echantillonnage * 1000));
// fin do-while
Serial.print(temps_fin_du_calcul);
Serial.print(" = t_calcul ; t_cycle = ");
Serial.println(temps_cycle_calcule);
} // fin de la boucle loop
```

A la ligne 193, on affiche cette durée (en microsecondes) dans la fenêtre du port série pour contrôle.

→ compléter la ligne 185 (en remplaçant la valeur « 0 ») pour permettre la mesure et l'affichage du temps consacré aux calculs par le microcontrôleur dans la variable « temps_fin-calcul ».

→ téléverser le programme dans la mémoire du microcontrôleur et cliquer sur l'afficheur du « moniteur série » pour analyser le résultat. Vérifier que le microcontrôleur effectue bien ses calculs dans un temps inférieur à la période d'échantillonnage fixée.

Travail 3-3-2 : analyse du comportement du système avec période d'échantillonnage élevée

La ligne 24 fixe la période d'échantillonnage à 20 millisecondes.

```
      //*************** variables pour les correcteurs ****************************************************
      int integrateur_max = 100 ;          // plafond du calcul d'integration
24 →  long periode_echantillonnage = 20;   // période d'exécution de la boucle principale (en millisecondes) ;
```

→ Placer successivement les valeurs 50, 100 et 150 millisecondes pour la période d'échantillonnage ; Pour chaque valeur, téléverser le programme dans la mémoire du microcontrôleur de l'arduino-box. Vérifier les connexions à l'aide de la **« Fiche_connexions-pupitre-arduino-box.pdf »** ;
Tester la stabilité de l'asservissement avec le point de fonctionnement ayant le potentiomètre « commande moteurs » à 20 % et une commande en échelon de 10° approximativement réalisée avec le potentiomètre « commande tangage ».

→ Conclure en justifiant le comportement de l'asservissement ***par l'analyse des diagrammes fréquentiels réalisée par le groupe 1*** (si le logiciel de simulation du laboratoire a permis cette analyse des retards).

Conclusion de la phase 3 :
→ ***Compléter le compte-rendu commun aux trois groupes***, en ajoutant les résultats associés à cette commande en double boucle.
→ A l'aide de la **« Fiche_ecarts_simu-reel.pdf »** choisir deux ou trois explications pertinentes aux écarts constatés entre les résultats de simulation et les résultats d'expérimentation.

Société DMS
Aéroparc Saint Martin – 12 rue de Caulet – 31300 TOULOUSE – ☎ : + 33 (0)5 62 88 72 72 🕮 : + 33 (0)5 62 88 72 79
Site internet : www.dmseducation.com Email : info@dmseducation.com