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

Cette fiche détaille le travail réalisé par le *groupe 1* 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 essentiellement réalisé autour des systèmes d'acquisition de données de position angulaire du drone didactique D2C ; les acquisitions autour des interfaces homme/machine (potentiomètres) seront aussi abordées.

Travail 1-1-1 : analyse des signaux analogiques
La **fiche technique « Capteur-angulaire-sans-contact-Ametes.pdf »** fournie par le constructeur, présente le « capteur d'angle pivot » ; le modèle utilisé est la version 0-90° ;

→ analyser la courbe de réponse fournie par le constructeur à la page 3, et en déduire la plage de tension dans laquelle se situera le signal généré par le capteur, ainsi que le gain « Kap » du capteur ;

→ Transmettre cette valeur de gain Kap au groupe 2 pour qu'il puisse valider le schéma-bloc avant d'effectuer les simulations.

réponse : plage : 0,5 volt à 4,5 volts → 4 volts pour 90° d'où le gain : Kap = 0,0444 V/degré

- lorsque le balancier du drone didactique est à l'horizontale, il a été décidé que ce signal serait exactement au milieu de la plage de variation ;
→ en déduire la valeur de la tension générée par le capteur lorsque le balancier est en position horizontale.

réponse : 2,5 volts

→ Ouvrir la porte du côté droit du système D2C, positionner le bloqueur sur « TANGAGE LIBRE » et manipuler le balancier pour vérifier les réponses précédentes ; on utilisera le voltmètre placé entre la borne « MASSE » et la borne « MESURE ANGLE TANGAGE » du pupitre.
→ effectuer la même mesure pour les deux potentiomètres « COMMANDE MOTEURS » et « COMMANDE TANGAGE » du pupitre ; noter la plage d'évolution du signal.

Les deux potentiomètres donnent un signal qui évolue entre 0 et 5 volts

Travail 1-1-2 : analyse de la conversion analogique / numérique
Le microcontrôleur qui va effectuer la conversion analogique / numérique des signaux possède un convertisseur qui fonctionne sur 10 bits pour une plage de signaux d'entrée de 0 à 5 volts ;
→ en déduire :

- la plage de variation des signaux convertis (en « points » de calcul) ;
  réponse : 0 à 1023 (1024 points correspondent à 2^{10})
- la valeur obtenue en position horizontale, lors de la mesure de l'angle de tangage par le « capteur d'angle pivot ».
  réponse : 512
- la valeur obtenue en position centrale lors de la commande de tangage effectuée avec le potentiomètre « COMMANDE TANGAGE ».
  réponse : 512

→ Calculer la « résolution » de l'acquisition (ou « quantum ») en degrés, de la mesure du « capteur d'angle pivot », et comparer celle-ci à la valeur du critère de précision C-2-1 du cahier des charges (***« fiche_cahier-des-charges-asservissements »***).
Rappel : la résolution est la plus petite variation mesurable.

Réponse :
La variation d'angle de tangage de 90° correspond à une variation de 4 volts ;
Si la plage pouvait aller jusqu'à 5 volts, il faudrait parcourir un angle de 5 x 90 / 4 = 112,5°
Cet angle correspondrait alors à 1024 « points » du microcontrôleur,
donc la résolution de la mesure est : 112,5° / 1024 = **0,1 degré ;**
cette valeur est largement inférieure à la valeur prise en compte pour évaluer la précision dans le cahier des charges.

Travail 1-1-3 : mise en œuvre de la programmation des acquisitions

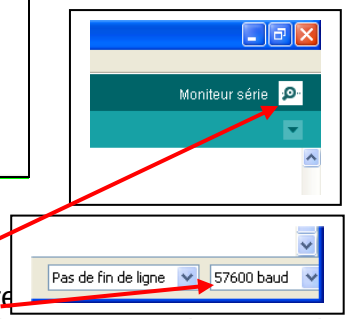
→ Après avoir identifié le pavé des entrées analogiques sur l'Arduino-box, mettre en place (ou vérifier) le câblage des entrées (capteur angulaire + potentiomètre des gaz + potentiomètre de consigne angulaire) entre le pupitre et le microcontrôleur de l'Arduino-box
(voir la **« fiche connexion-pupitre-arduino_box »**) ;

→ Utiliser le programme ***« _0_Test-entrees.ino »*** dans l'interface de programmation Arduino ; l'enregistrer avec votre nom ; supprimer les « // » devant les lignes qui permettront de visualiser les valeurs des entrées « capteur angle pivot » + « potentiomètre des gaz » + « potentiomètre de consigne angulaire » ;

Résultat :

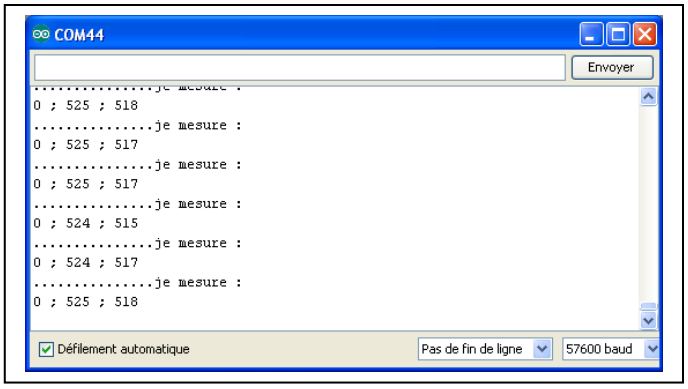
```
//*************************************************************************************************************************
//boucle principale qui se répète indéfiniment
void loop() {
  // ************* lecture des entrées capteurs ***************************************
  mesure_gyro=analogRead(broche_gyro);                          // acquisition de la mesure du gyromètre
  mesure_accelero = analogRead(broche_accelero);                // acquisition de la mesure de l'accéléromètre
  mesure_potentio_gauche = analogRead(broche_potentio_gauche);  // acquisition de la mesure du potentio gauche
  mesure_potentio_droit = analogRead(broche_potentio_droit);    // acquisition de la mesure du potentio droit
  mesure_angle_pivot = analogRead(broche_angle_pivot);          // acquisition de la mesure du capteur d'angle pivot

  //********************** envoi sur le port série pour tests *********************************
  if (nb_cycles >= 10) {                        // on n'affiche que tous les 10 cycles pour ne pas surcharger le port série
    Serial.println(".............je mesure : ");
    //Serial.print(mesure_gyro);
    //Serial.print(" ; ");
    //Serial.print(mesure_accelero);
    //Serial.print(" ; ");
    Serial.print(mesure_potentio_gauche);
    Serial.print(" ; ");
    Serial.print(mesure_potentio_droit);
    Serial.print(" ; ");
    Serial.print(mesure_angle_pivot);
    Serial.println();
    nb_cycles = 0;
  } // fin if
nb_cycles = nb_cycles + 1;
delay(10); // on attend une dizaine de millisecondes pour ne pas surcharger le processeur
} // fin de la boucle loop
```



→ Charger le programme dans la mémoire de l'Arduino ;
→ Cliquer sur l'icône du port série (en haut à droite) pour obtenir l'affichage ;
→ Ajuster la vitesse du port série à 57600 Bauds (en bas à droite de la fenêtre) ;
→ Valider les résultats du travail 1-1-2 concernant la plage de variation des signaux convertis et pour les valeurs centrales.

Résultats : La rotation des potentiomètres et du balancier provoquent l'évolution des grandeurs entre 0 et 1023 ; les valeurs centrales sont d'environ 512 :



Travail 1-1-4 : analyse des saturations de la commande
Les grandeurs d'entrée qui sont en dehors de la plage d'action du convertisseur analogique numérique sont tronquées aux valeurs limites 0 ou 1024 lors de la conversion ; c'est le phénomène de saturation en entrée du microcontrôleur.
Donnée : les convertisseurs « numérique / analogique » qui réalisent la commande des signaux de sortie vers les cartes de puissance des moteurs sont des convertisseurs 8 bits ;
→ Expliquer ce qu'il se passe si le signal de commande généré par le calculateur dépasse la valeur 255.
→ Analyser le schéma-bloc ***étudié par le groupe 2 qui travaille sur la simulation de la boucle d'asservissement*** et proposer en conséquence une explication à d'éventuels écarts entre les résultats de simulation et les résultats d'expérimentation.

Réponse : il se produit aussi des saturations de la grandeur de sortie ; ces saturations ne sont pas modélisées sur les schéma-blocs ; ceci peut générer des écarts entre les résultats de simulation et les résultats d'expérimentation.

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.

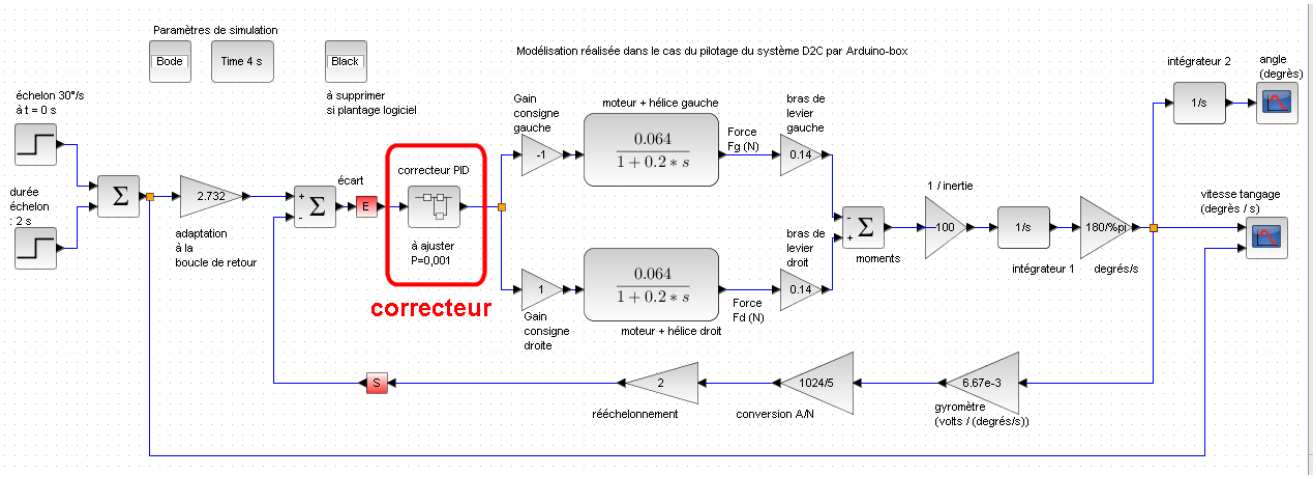
Phase 2 : Etude de l'asservissement de la vitesse de tangage ;

Dans cette phase, le travail du groupe est centré autour de la mise en place de l'équation de récurrence du correcteur dérivé filtré, 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 1-2-1 : démonstration de pilotage en vitesse de tangage
→ Tester le pilotage en vitesse 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 au paragraphe 3 de la **« fiche_pilotage-D2C-par-DSPic »** ;
→ Faire la démonstration aux autres groupes du « pilotage en vitesse » du drone didactique, en réalisant quelques aller-retours entre -30° et + 30°.

Travail 1-2-2 : Ecriture de l'équation de récurrence du correcteur

Le ***groupe 3*** a la charge de réaliser un travail de simulation logicielle pour valider l'utilisation d'un correcteur proportionnel intégral dérivé (PID) dans l'objectif d'obtenir un système stable en tangage (schéma-bloc ci-dessous).



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

$$S_n(\text{dérivé filtré}) = \frac{1}{Te+\tau}\left[\tau.S_{n-1} + E_n - E_{n-1}\right]$$

→ Démontrer l'équation de récurrence du correcteur « dérivé filtré » proposée au paragraphe 2-5 de la **« fiche_equations-de-recurrence »** (on utilisera la méthode des rectangles) ;

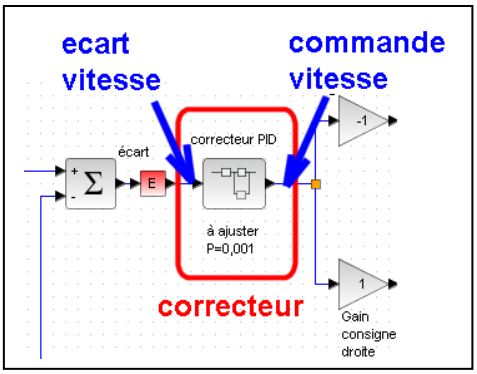
Travail 1-2-3 : implémentation du correcteur dans le programme de commande
Ouvrir le programme ***« _2_asservit_1boucle_vitesse_PIDF_a_completer.ino »***, l'enregistrer avec votre nom ; il a été préparé de la façon suivante (le numéro de ligne où est positionné le curseur s'affiche en bas à gauche de l'écran) :
- les premières lignes (23 à 31) définissent les variables pour le correcteur, dont la période d'échantillonnage fixée à 20 millisecondes ;

```
//*************** variables pour le correcteur PID filtré ****************************************
int integrateur_max = 100;           // plafond du calcul d'integration
long periode_echantillonnage = 20;   // période d'exécution de la boucle principale et de mise à jour du PID
float integrateur = 0;                // variable globale pour la fonction PID
float precedente_entree = 0;          // variable globale pour la fonction PID
float precedente_sortie = 0;          // variable globale pour la fonction PID
float kp = 0.031;         // correction proportionnelle
float ki = 0.0024;        // correction intégrale
float kd = 0.0056;        // correction dérivée
float tau = 1/19;         // réglage de la pulsation de coupure du filtre dérivé
```

- Pour écrire le correcteur, on utilise une fonction nommée « PID_out » ;
les lignes suivantes (33 à 46) permettent d'écrire la fonction du correcteur dérivé filtré « PID_out » qui sera appelée plus loin pour le calcul de la commande des moteurs.

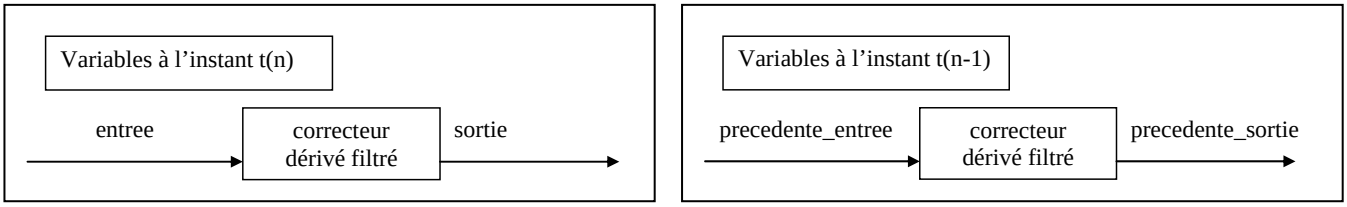
```
//********************************** la fonction du correcteur PID filtré ******************
float PID_out(float entree, long dt) {
  float sortie;
  float derive_filtre = // c'est ici qu'il faut écrire l'équation de la partie "dérivé filtré"
  integrateur = integrateur + entree * (float(dt))/1000.0f;
  integrateur = constrain(integrateur,-integrateur_max,integrateur_max);      // plafonne la valeur de l'integrateur
  sortie = (kp * entree);          // calcule la partie proportionnelle
  sortie = sortie + integrateur * ki;         //ajoute la partie integrale
  sortie = sortie + derive_filtre * kd;          //ajoute la partie dérivée
  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 PID
}
```

- la boucle « loop » qui se répète indéfiniment est écrite entre les lignes 97 et 145, et l'utilisation de la fonction du correcteur est faite à la ligne 109, où la variable « commande_vitesse » récupère la sortie de la fonction correcteur « PID_out », pour laquelle l'entrée est « ecart_vitesse » (schéma ci-contre).




```
// ******************************   asservissement ***********************************
//****************** asservissement de la vitesse du tangage
vitesse = float(mesure_gyro - gyro_nul)  * facteur_dechelle_gyro;
mesure_potentio_droit = analogRead(broche_potentio_droit);          // acquisition de la tension du potentio 0 à 5V sur 10 bits (valeurs de 0 à 1023 )
ecart_vitesse = 0.5*(float(mesure_potentio_droit - 512)) - vitesse;    // calcul de l'écart entre consigne et mesure (0,5 pour augmenter l'amplitude du pot
commande_vitesse = PID_out(ecart_vitesse, periode_echantillonnage);    // on calcule la sortie du PID avec la fonction "PID_out"
commande_unitarisee = commande_vitesse / 50.0f;                        // rééchelonnement de la commande de tangage
```

Le travail à réaliser porte sur l'écriture mise en place à l'intérieur de la fonction « PID_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_vitesse_PIDF_a_completer.ino »*** (enregistré avec votre nom) pour permettre de générer la commande de pilotage des moteurs ;

```
//********************************** la fonction du correcteur PID filtré ******************
float PID_out(float entree, long dt) {
  float sortie;
  float derive_filtre = // c'est ici qu'il faut écrire l'équation de la partie "dérivé filtré"
  integrateur = integrateur + entree * (float(dt))/1000.0f;
  integrateur = constrain(integrateur,-integrateur_max,integrateur_max);      // plafonne la valeur de l'integrateur
  sortie = (kp * entree);          // calcule la partie proportionnelle
  sortie = sortie + integrateur * ki;          //ajoute la partie integrale
  sortie = sortie + derive_filtre * kd;           //ajoute la partie dérivée
  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 PID
}
```

Corrigé :

```
//********************************** la fonction du correcteur PID filtré ******************
float PID_out(float entree, long dt) {
  float sortie;
  float derive_filtre = (1/((float(dt))/2000.0f + tau)) * ((tau + (float(dt))/2000.0f)*precedente_sortie + ((float(dt))/2000.0f)*(entree + precedente_entree));
  integrateur = integrateur + entree * (float(dt))/1000.0f;
  integrateur = constrain(integrateur,-integrateur_max,integrateur_max);      // plafonne la valeur de l'integrateur
  sortie = (kp * entree);          // calcule la partie proportionnelle
  sortie = sortie + integrateur * ki;          //ajoute la partie integrale
  sortie = sortie + derive_filtre * kd;           //ajoute la partie dérivée
  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 PID
}
```

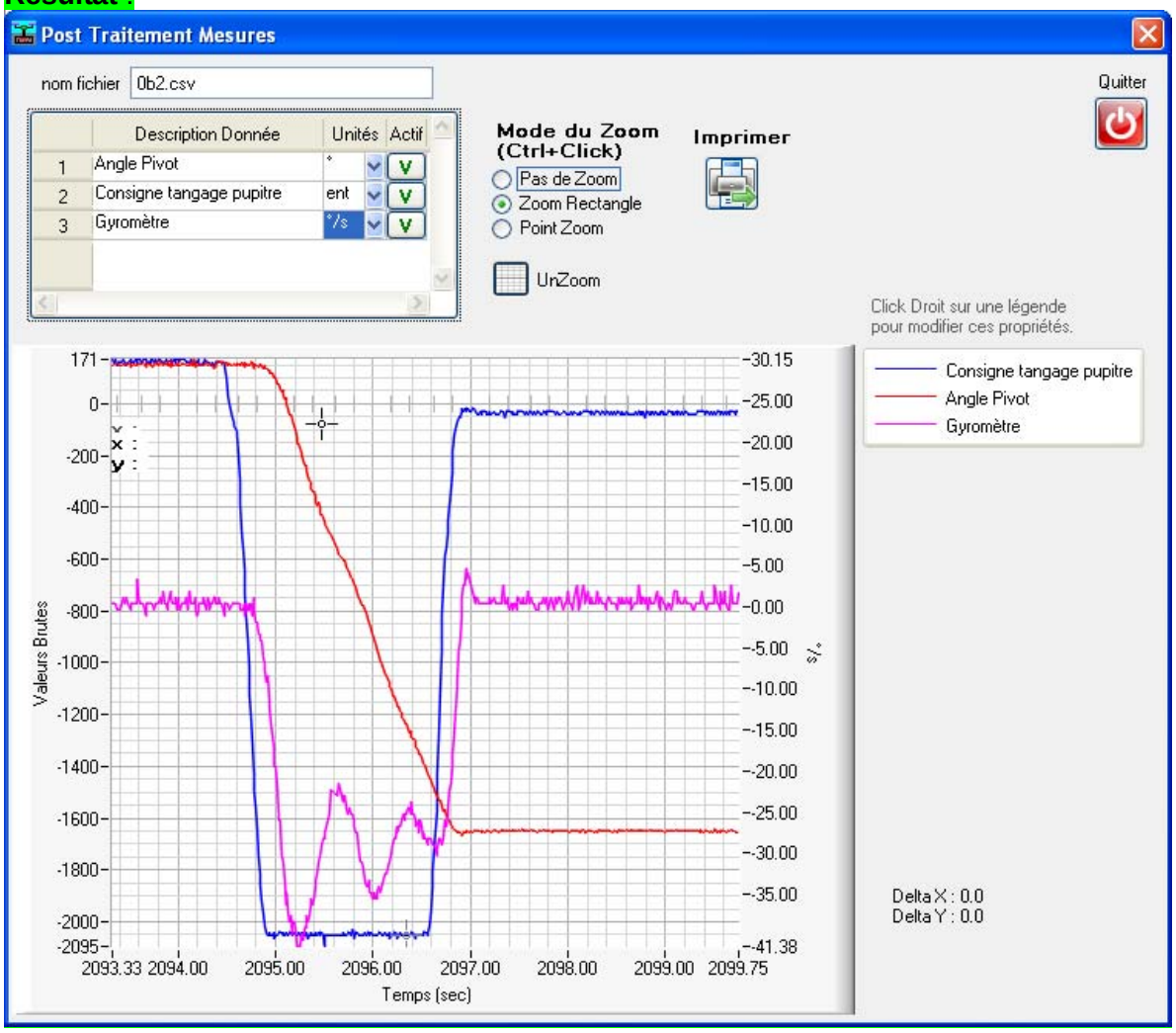
→ Vérifier auprès du groupe 2 qui effectue la simulation logicielle, que les valeurs du correcteur placées aux lignes 28 à 30 pourraient convenir vis-à-vis du cahier des charges (voir la ***« fiche_cahier-des-charges-asservissements »***).

```
//*************** variables pour le correcteur PID filtré ****************************************
int integrateur_max = 100;          // plafond du calcul d'integration
long periode_echantillonnage = 20;   // période d'exécution de la boucle principale et de mise à jour du PID
float integrateur = 0;                // variable globale pour la fonction PID
float precedente_entree = 0;          // variable globale pour la fonction PID
float precedente_sortie = 0;          // variable globale pour la fonction PID
float kp = 0.031;       // correction proportionnelle
float ki = 0.0024;      // correction intégrale
float kd = 0.0056;      // correction dérivée
float tau = 1/19;       // réglage de la pulsation de coupure du filtre dérivé
```

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

Travail 1-2-4 : pilotage du drone didactique
→ Vérifier la mise en place des câblages entre l'Arduino-box et les moteurs du système D2C (voir la **« fiche connexion-pupitre-arduino_box »**) ;
→ Tester la commande en vitesse angulaire programmée dans l'Arduino-box, à l'aide des commandes du pupitre ; vérifier la stabilité de cette commande.

Résultat :



L'acquisition de la réponse à un créneau de consigne réalisé avec le potentiomètre du pupitre montre que le système n'est pas instable, même si la réponse (en vitesse angulaire) est un peu oscillante.

Conclusion de la phase 2 :
→ Rédiger dans le compte-rendu commun aux trois groupes, une courte synthèse des démarches réalisées et des résultats obtenus dans ce mode de pilotage en vitesse (respecter l'organisation logique de ce compte-rendu de façon pour présenter les détails de la fonction "traiter" réalisée par le microcontrôleur dans ce mode vitesse).
Une synthèse présentée sous forme de synoptique sera bien appréciée.

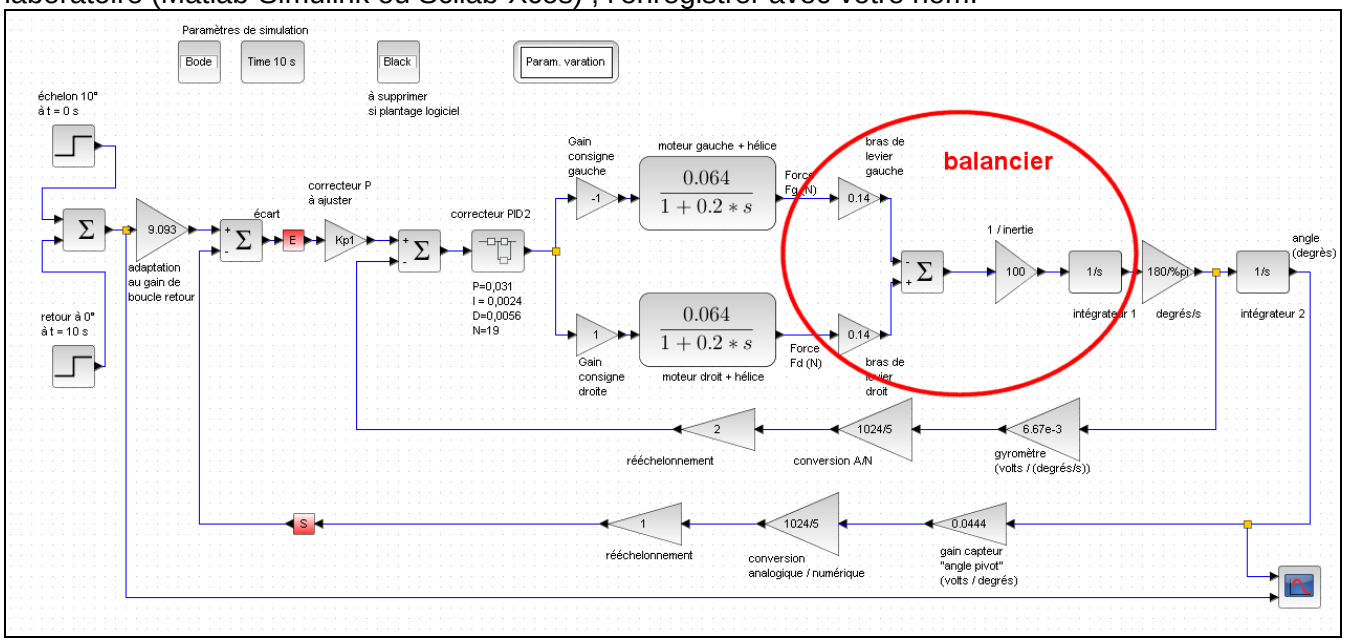
Société DMS
Aéroparc Saint Martin – 12 rue de Caulet – 31300 TOULOUSE – ☎ : + 33 (0)5 62 88 72 72 🕮 : + 33 (0)5 62 88 72 79
Site internet : www.dmseducation.com Email : info@dmseducation.com

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

L'objectif est maintenant de travailler en simulation sur le modèle du système à deux boucles, pour :
- d'une part, vérifier qu'il est plus performant que le modèle du système à une boucle ;
- d'autre part, de régler le coefficient proportionnel de la boucle externe.

Il s'agira de prendre en compte les imperfections dues, entre autres, à l'échantillonnage.

Travail 1-3-1 : Analyse du schéma-bloc
Ouvrir le fichier ***« simu_D2C_Arduino_2boucles_a_ajuster »*** avec le logiciel de simulation du laboratoire (Matlab-Simulink ou Scilab-Xcos) ; l'enregistrer avec votre nom.



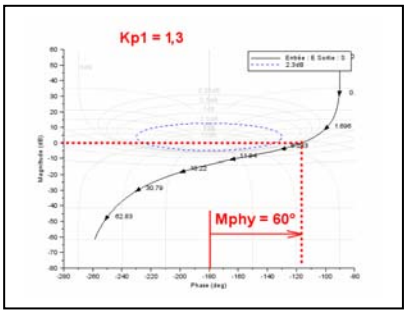
a) validation des deux boucles de retour
→ vérifier les coefficients de la boucle interne de vitesse à partir du travail effectué par le groupe 2, dans la phase 2.
→ vérifier les coefficients de la boucle externe à partir du travail effectué dans la phase 1 ; justifier le coefficient 9,093 du bloc d'adaptation.

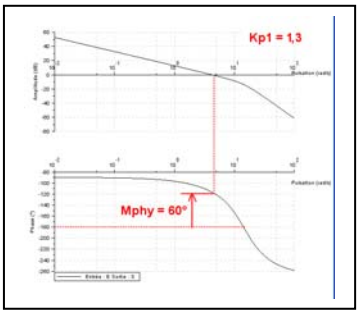
b) validation de la chaîne directe
→ Indiquer quelle est l'équation qui a été utilisée pour modéliser la zone désignée par « Balancier » sur le schéma-bloc ci-dessus, et comment elle a été obtenue.
→ En approximant le modèle de la boucle interne à une fonction de transfert du second ordre, déterminer la « classe » de la fonction de transfert en boucle ouverte.
Comparer la valeur obtenue, à celle du schéma-bloc qui ne contient pas de boucle interne de vitesse (***voir le schéma-bloc analysé en phase 1 par le groupe 2***) ; en déduire l'effet bénéfique apporté par la boucle interne dans la recherche du compromis stabilité / rapidité.

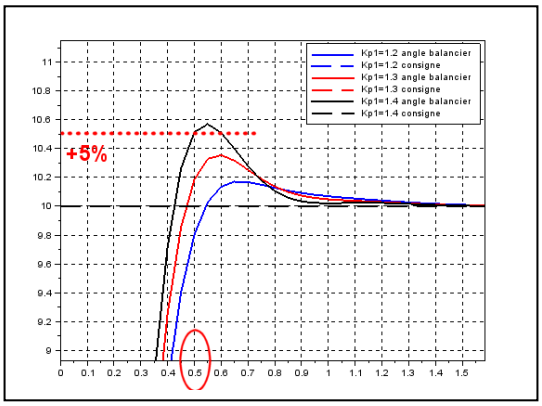
Solution :
- Le schéma-bloc analysé en phase 1 était un schéma bloc avec une fonction de transfert en boucle ouverte de classe 2 (présence de deux intégrateurs) ; le déphasage apporté dans une analyse fréquentielle est donc inférieur à -180°, ce qui nécessitait d'ajouter une avance de phase pour la stabilité ;

- la fonction de transfert en boucle ouverte étudiée maintenant est une fonction de transfert de classe 1 (présence de « l'intégrateur 2 ») seulement ;
le déphasage apporté dans une analyse fréquentielle est de -90° en basses fréquences ; ceci donnera plus de facilité pour obtenir la stabilité et sera bénéfique dans la recherche de rapidité.

Travail 1-3-2 : Ajustement du correcteur
→ réaliser des simulations pour ajuster le coefficient proportionnel « Kp1 » du correcteur de la boucle de position du modèle du système piloté par Arduino, de façon à respecter les critère de stabilité du cahier des charges (voir la **« fiche_cahier-des-charges-asservissements »**) ;
Solution : un coefficient **Kp1 = 1,35** permet de respecter le critère du dépassement < 5% :
La marge de phase est alors de 60° environ.







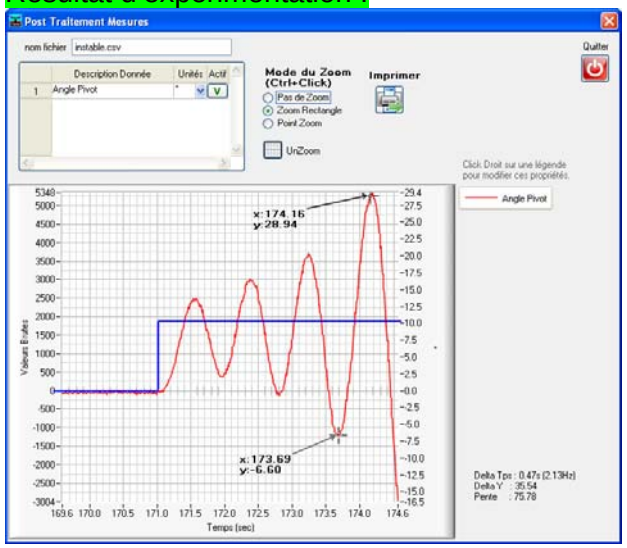
→ vérifier que l'ajout de la boucle interne a permis de rendre le système plus rapide par rapport au cas de la commande à une seule boucle de position étudié dans la phase 1, et de répondre au cahier des charges sur le critère de rapidité.

→ Fournir au groupe 2 la valeur du coefficient proportionnel Kp1, pour qu'il l'intègre dans la programmation de la commande à deux boucles ;

→ Participer à l'analyse des écarts entre les résultats de simulation et les résultats d'expérimentation.

Réponse : le coefficient de 1,3 mis en place dans l'expérimentation effectuée par le groupe 2 ne donne pas satisfaction ; en effet il apparaît que le système est instable (contrairement au résultat de la simulation) :

Résultat d'expérimentation :

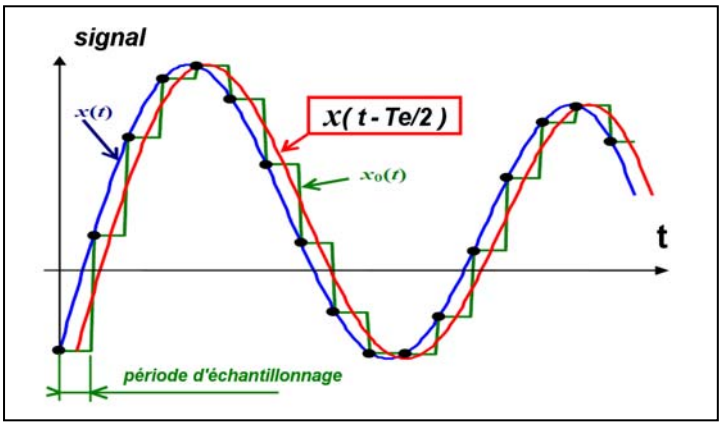


Travail 1-3-3 : Ajustement du modèle de simulation

Pour que le modèle de simulation soit plus proche du comportement du système réel, on propose de tenir compte de trois phénomènes non-linéaires, négligés dans la précédente analyse :

- la modification apportée par le fonctionnement « échantillonné » de la commande programmée ;
- le retard apporté par le fonctionnement des contrôleurs numériques des moteurs ;
- le retard apporté par le « capteur angle pivot ».

a) données sur la modélisation de la commande échantillonnée :

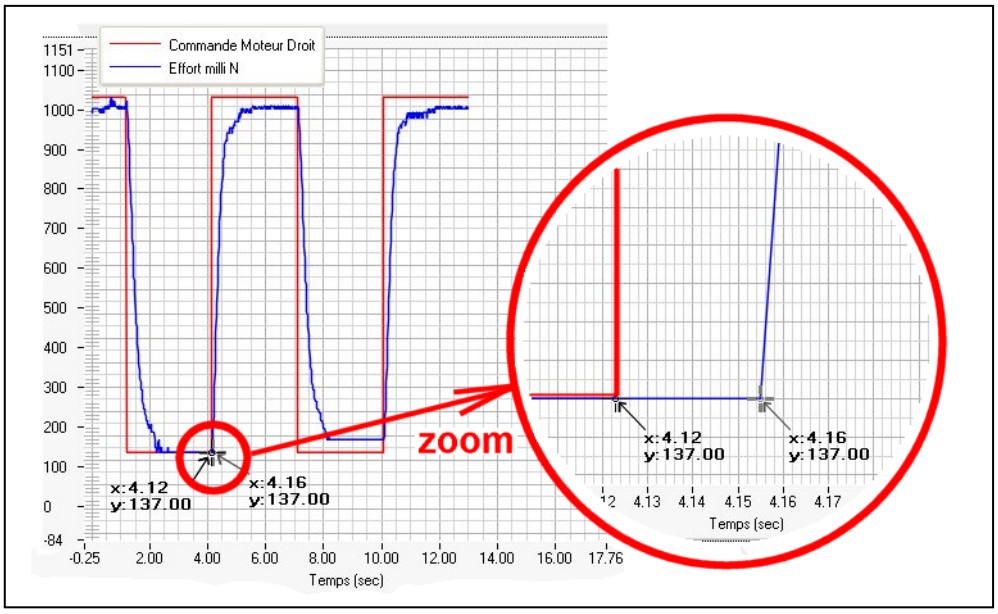


Lorsque le programme effectue une séquence de traitement d'informations, il procède par lecture des entrées du microcontrôleur aux intervalles définis par la période d'échantillonnage.
On considère qu'une bonne approximation continue de l'échantillonnage est un retard d'une demi-période (voir la figure ci-contre).

La période d'échantillonnage choisie dans les programmes Arduino a pour valeur **20 millisecondes**.

b) données sur la modélisation des retards dus au fonctionnement des contrôleurs numériques des moteurs

Une mesure précise de la réponse de la motorisation à une sollicitation en échelon, a permis de détecter un retard par rapport à la consigne ; les figures ci-dessous mettent en évidence ce retard.



Une valeur de **0,04 seconde** peut être retenue pour ce retard.

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

c) données sur le retard de réponse du « capteur angle pivot »

Le capteur d'angle pivot est un capteur qui doit effectuer du traitement numérique avant de délivrer sa réponse analogique ;
le temps mis pour effectuer ce traitement est donné par la grandeur « Response time » à la page 2 de la fiche technique du capteur :
il a pour valeur : **17 millisecondes**.

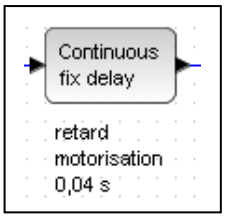
Electrical Characterisitics
Operating Conditions: V_{Sup} = 5.0 V and T_A= 25° C
Note (1): Outputs are ratiometric to the supply voltage V_{SUP}

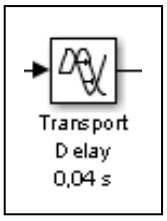
Outputs	Selectable Range	Confg Pin A	Config Pin B	Output Voltage Range	Remarks
Analog Output Voltage Ranges	1	COM	COM	0.5 V to 4.5V(1)	0 to 360 deg-See output plot (p3)
	2	COM	V_{Sup}	0.5 V to 4.5V(1)	0 to 30 deg-See output plot (p3)
	3	V_{Sup}	COM	0.5 V to 4.5V(1)	0 to 90 deg-See output plot (p3)
	4	V_{Sup}	V_{Sup}	2.5 V ± 2.0V(1)	0 ± 90 deg-See output plot (p3)
	All ranges			≤ 50mV	When magnet out of range
Analog output current				± 1mA max	
Inputs					
Zero Angle Set		Active High		>4.75 V prior to V_{Sup} application	Max voltage not to exceed 6V
"Zero Angle Set" output level		360ASMC-01		0.50 V±0.01 V(1)	Ranges 1, 2 & 3
		360ASMC-01		2.50 V±0.01 V(1)	Range 4
Response time		≈ 17mS		Time required to sample inputs and update output	
Resolution		10 bits			
Accuracy		0-360 Deg		± 1 deg	Selectable Range 1
		0-30 Deg		± 0.1 deg	Selectable Range 2
		0-90Deg		± 0.25 deg	Selectable Range 3
		0 +/- 90 Deg		± 0.5 deg	Selectable Range 4
Magnetic Specifications					
Max horizontal field		50 mT (500G)		At surface of module	
Min horizontal field		6 mT (60G)		Below 6 mT, Magnet "Out of Range" will be activated	

Revision Date: 07 May 2007 - 360ASMC SPEC
Europe: **Senis GmbH: Technoparkstrasse 1, 8005 Zurich, Switzerland. www.senis.ch. +41 (79) 366-8756**
The Americas: GMW Associates: 955 Industrial Road, San Carlos, CA 94070, USA. www.gmw.com. +1 (650) 802-8292
2

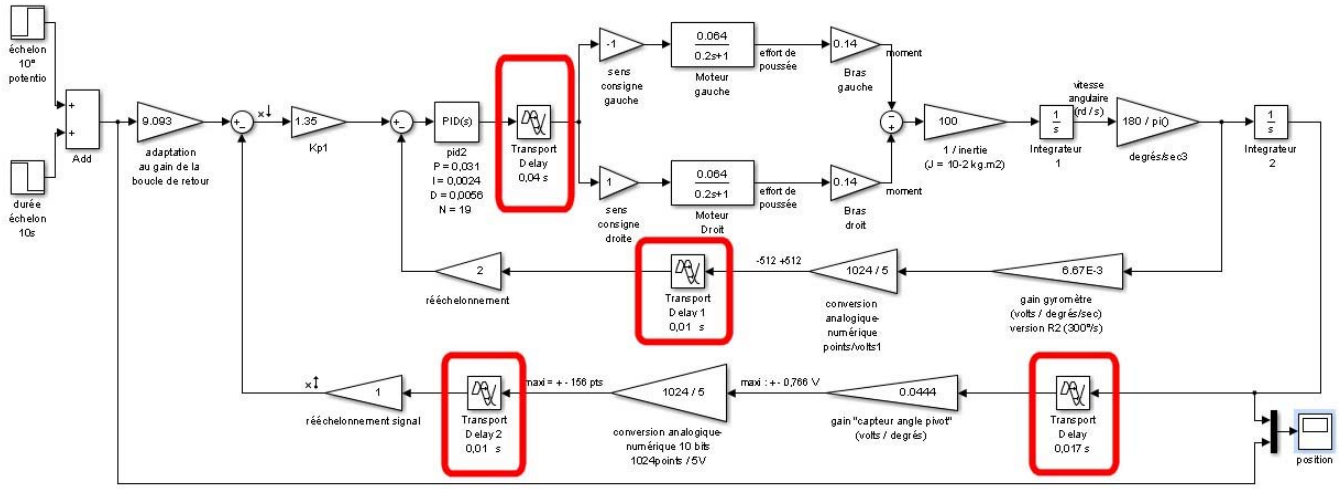
→ Modifier le schéma-bloc pour y placer les différents retards listés précédemment ;

- pour Scilab-Xcos : bloc « Continuous fix delay » de la bibliothèque CPGE - UIODelay ;
- pour Matlab-Simulink : bloc « Transport Delay » de la bibliothèque Simulink - Continuous

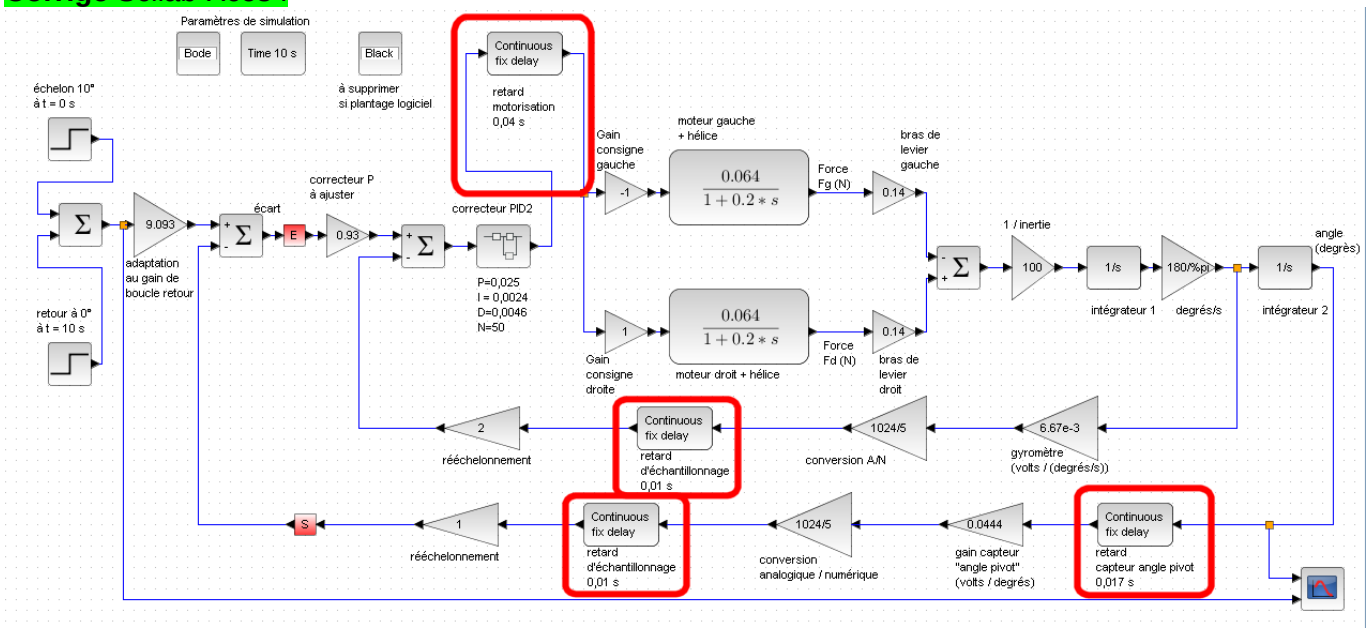




Corrigé matlab-Simulink :

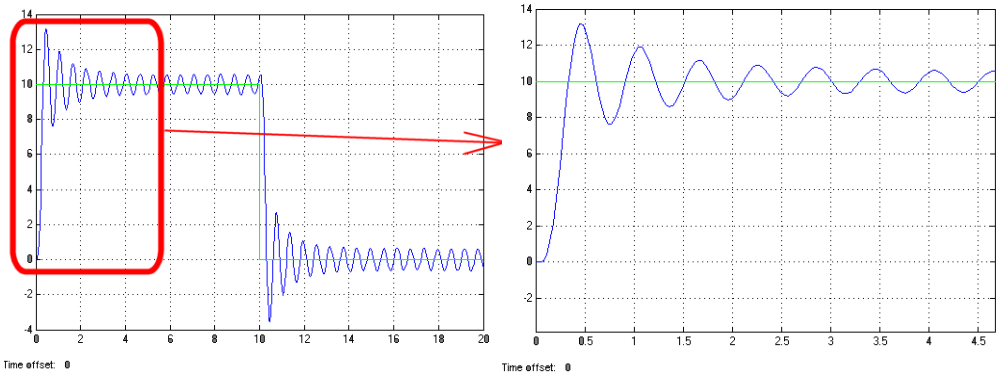


Corrigé Scilab-Xcos :

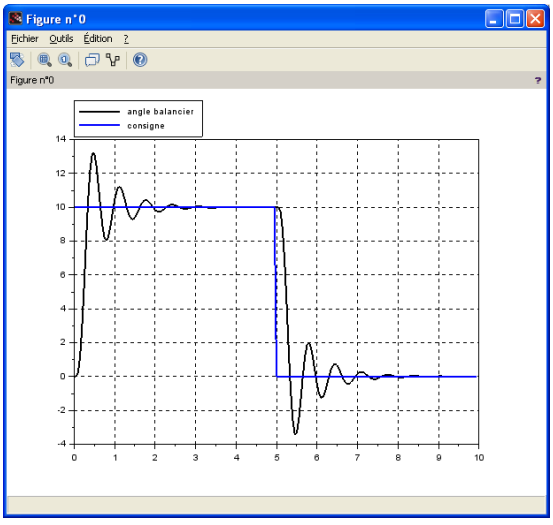


→ Effectuer la simulation pour montrer que le résultat de la simulation temporelle de ce nouveau modèle s'est approché du résultat de l'***expérimentation effectuée par le groupe 2 (travail 2-3-3)***.

Corrigé : La simulation Matlab montre un comportement très proche de l'instabilité.



La simulation Scilab est plus oscillante aussi :



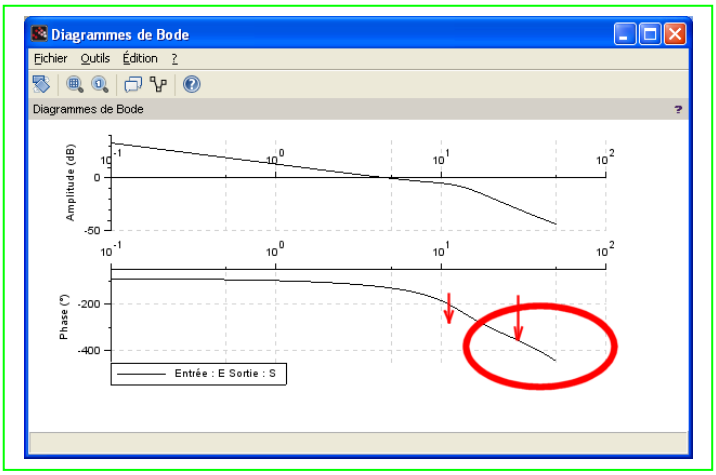
Société DMS
Aéroparc Saint Martin – 12 rue de Caulet – 31300 TOULOUSE – ☎ : + 33 (0)5 62 88 72 72 📖 : + 33 (0)5 62 88 72 79
Site internet : www.dmseducation.com Email : info@dmseducation.com

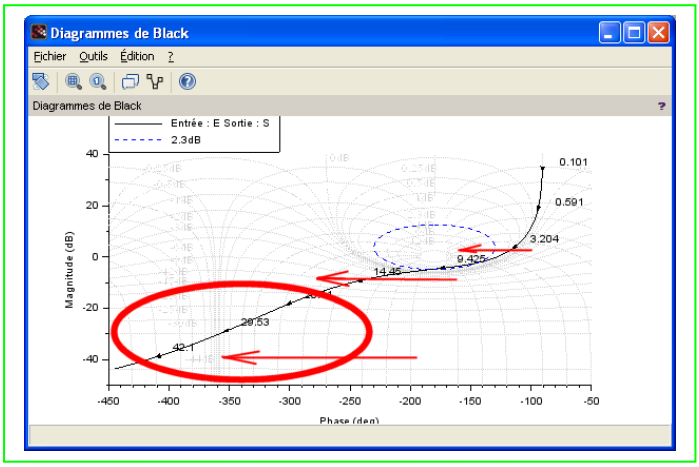
Attention avec l'analyse fréquentielle :

- Matlab-Simulink (2013a) ne traite pas les retards dans l'analyse fréquentielle des blocs « Bode plot » et « Nichols plot » de la bibliothèque « Simulink Control Design » ; il ne faudra donc travailler qu'avec la réponse temporelle.
- Scilab-Xcos ne traite les retards dans la réponse fréquentielle qu'à partir de la version 5.5 et avec la bibliothèque CPGE 1.6.

→ si l'analyse fréquentielle est possible avec le logiciel de votre laboratoire, analyser l'effet des retards sur la courbe de phase et justifier l'écart entre le comportement du système simulé au travail 1-3-2 et le système réel programmé avec les mêmes coefficients des correcteurs.

Corrigé : les différents retards provoquent un déplacement négatif de la courbe de phase, ce qui augmente l'instabilité.

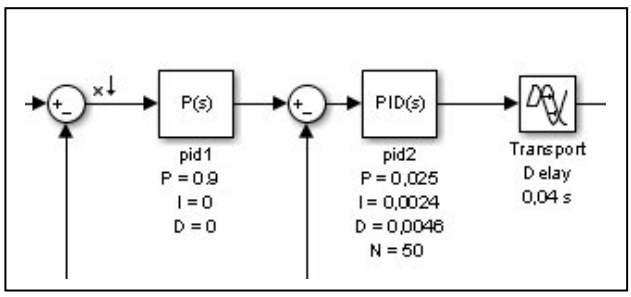




Travail 1-3-4 : Réajustement des paramètres du correcteur
La prise en compte des retards nécessite un retour sur les réglages de la boucle interne de vitesse puis un réajustement du correcteur proportionnel de la boucle externe de position.
Le fichier ***« simu_D2C_Arduino_2boucles_solution_retards_rectifie »*** propose un nouveau réglage adapté à la prise en compte des différents retards évoqués précédemment.

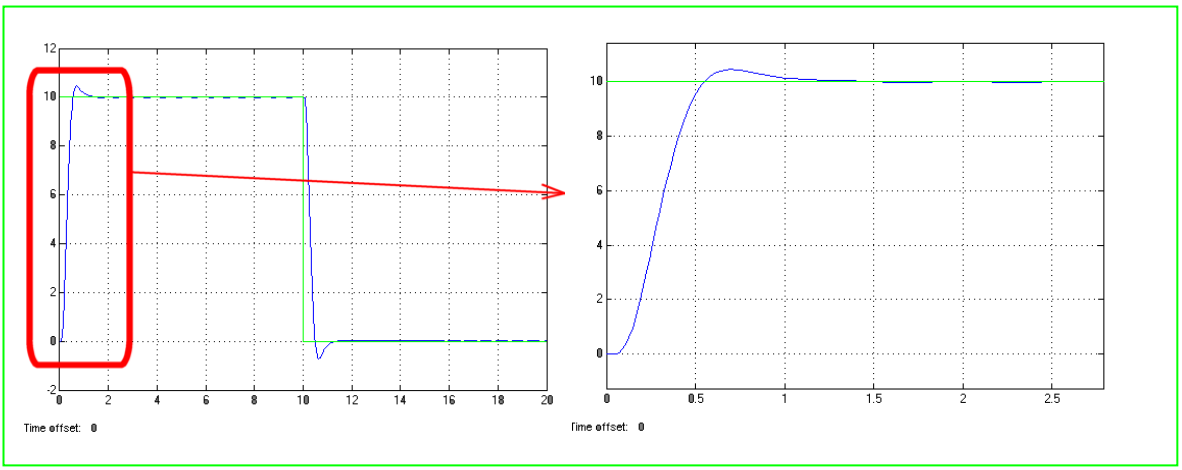
→ Effectuer la simulation pour valider (par rapport au cahier des charges) les choix proposés dans ce fichier et ***transmettre les nouvelles valeurs des correcteurs au groupe 2*** pour qu'il valide expérimentalement le résultat.

Corrigé :
Nouvelles valeurs des correcteurs :



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

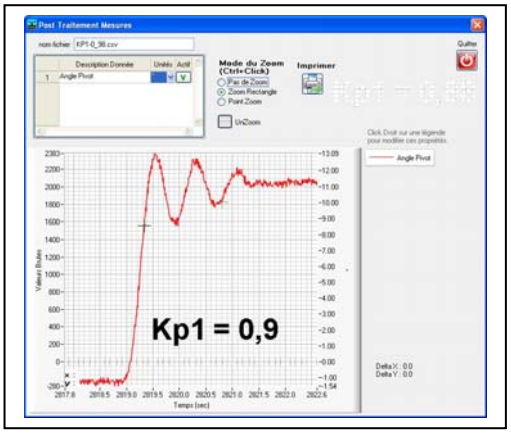
Nouvelle réponse temporelle :

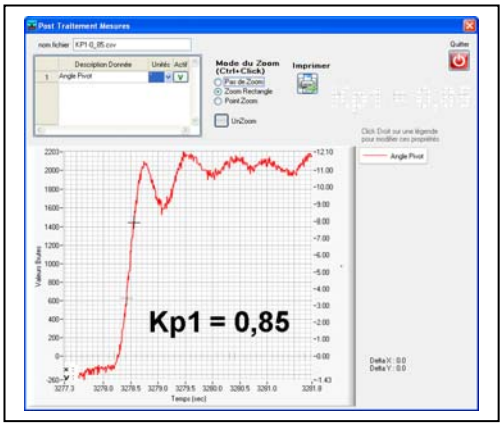


La stabilité est obtenue (dépassement < 5%) ; le temps de réponse à 5% est voisin de 0,5 seconde.

→ Comparer les résultats de cette simulation aux ***résultats d'expérimentation obtenus par le groupe 2*** ; A l'aide de la **« Fiche_ecarts_simu-reel.pdf »** choisir deux ou trois explications pertinentes aux écarts qui pourraient subsister entre expérimentation et simulation.

Corrigé :





On observe sur le système réel est un peu plus oscillant, et qu'en premier recours, le coefficient Kp1 peut être abaissé à 0,85 environ pour obtenir un comportement plus satisfaisant du point de vue des oscillations ;
Les causes de cet écart sont multiples, et proviennent surtout de la difficulté d'obtenir un modèle qui représente avec précision le système réel ;
La modélisation proposée pour le système échantillonné est aussi très approximative.

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