# Travaux pratiques d'automatique – Master 1 MNE

## 1 Préparation

Il est impératif de préparer chaque séance de travaux pratiques. On estime à maximum 2 heures le temps nécessaire pour préparer un TP. L'essentiel du travail consiste à lire et à comprendre ce qui est demandé et, le cas échéant, à rechercher les informations qui pourront être nécessaires pendant la séance. Deux questions sont également explicitement posées pour la préparation du TP2 (section préparation).

## 2 Présentation des sujets

1. Le TP 1 peut être considéré comme un TD sur machine (simulation sous Matlab et Simulink ). Il servira à mettre en évidence l'effet de l'échantillonnage et de la quantification lors de la transposition de correcteurs et dans l'immunité aux bruits.
2. Le TP 2 porte sur l'asservissement en position d'une maquette moteur de faible puissance. Vous mettrez en évidence la différence de comportement des correcteurs synthétisés par transposition et des correcteurs synthétisés directement en numérique.
3. Le TP 3 porte sur l'asservissement de température d'une maquette pédagogique d'un système thermique. Deux méthodes de synthèse seront mises en oeuvre dans le cas très courant en pratique d'un système à retard : synthèse fréquentielle continue et transposition, et synthèse numérique directe.

## 3 Déroulement des TPs

Les TPs seront réalisés en binômes ou seul et il y aura une rotation entre les sujets (se rapporter au planning des rotations qui vous a été remis). Il vous est demandé de respecter le planning. L'encadrant se réserve le droit de refuser la présence en TP des étudiants se présentant en dehors des créneaux prévus. En cas de problème il vous est demandé de contacter Florent Nageotte avant la séance à l'adresse `Nageotte@unistra.fr` .

## 4 Evaluation

L'évaluation des TPs d'automatique est répartie entre contrôle continu (1/3 de la note totale de TP) et un examen oral final pouvant porter sur tous les sujets abordés en TPs, cours et TDs (2/3 de la note totale). L'évaluation en contrôle continu se fait sur la base :

1. du travail effectué pendant chacunes des 3 séances de 4heures ;
2. des comptes-rendus de TP, qui doivent notamment comporter les réponses aux questions théoriques de la préparation. Un compte-rendu parmi les 3 TPs sera demandé par étudiant.

## 5 Conseils pour la rédaction des comptes-rendus

L'objectif du compte-rendu est de montrer la bonne compréhension du travail qui a été réalisé pendant la séance. Il s'agit donc d'un travail d'analyse des résultats.
Par conséquent le compte-rendu **COMPORTERA** :

– l'étude théorique éventuellement demandée en préparation

- le relevé synthétique des résultats obtenus accompagné de légendes
- les réponses aux questions posées dans l'énoncé
- l'analyse des résultats obtenus : est-ce ce qui était attendu, pourquoi, comment améliorer les résultats, etc.
- les explications théoriques et pratiques

Mais le compte-rendu **NE CONTIENDRA PAS** :

- un recopié de l'énoncé et des questions posées
- des descriptions des commandes matlab tapées
- des dizaines de courbes non commentées et non légendées
- des résultats non observés durant la séance et obtenus de sources non autorisées (annales, autres binômes, etc.)

# TP 1 – Asservissement numérique d'un système continu

*L'objectif de ce TP est de mettre en évidence les effets de l'échantillonnage et de la numérisation des correcteurs continus. Ce TP s'effectuera entièrement en simulation avec* Matlab *et la boîte à outils* Simulink .

Le système étudié possède une seule sortie $y$ et une seule entrée $u$ (il s'agit de tensions, mesurées en Volts). Sa fonction de transfert, identifiée au point de fonctionnement, s'écrit :

$$G(s) = \frac{Y(s)}{U(s)} = \frac{2,3}{s(1+0,1s)},$$

où $s$ désigne la variable de Laplace.

## 1 Correction proportionnelle

### 1.1 Correcteur analogique

*Cette* $1^{ère}$ *partie est un rappel de l'utilisation de matlab pour les systèmes continus. Vous ne devez pas y passer plus d'une demie heure*

1. Définir la fonction de transfert du système à l'aide de la commande :
   ```
   >>G=tf([2.3],[0.1 1 0])
   ```
   et lancer l'utilitaire de tracé du lieu d'Evans avec la commande :
   ```
   >>rltool(G)
   ```
   Les points rouges sur le lieu d'Evans correspondent à la position des pôles du système bouclé pour la valeur du gain indiquée dans l'onglet `Compensator Editor` de la fenêtre `Control and Estimation Tools Manager`. Ces pôles peuvent être déplacés le long du lieu d'Evans. La valeur du gain est alors réactualisée.
2. Faire la synthèse d'un correcteur proportionnel analogique de sorte que le système asservi ait un facteur d'amortissement égal à 0,707.
3. Simuler à l'aide de `rltool` la réponse indicielle du système asservi pour un échelon unitaire. Identifier les éléments caractéristiques de la réponse confirmant le bon réglage du correcteur.
4. Lancer Simulink avec la commande :
   ```
   >>simulink
   ```
   Construire un modèle Simulink du système analogique en boucle fermée comprenant le correcteur proportionnel déterminé. Pour cela, on recherchera dans les menus de Simulink les différents blocs nécessaires (fonction de transfert continue, sommateur, gain, oscilloscope pour visualiser, etc.), que l'on fera glisser sur le modèle. La fonction de transfert $G(s)$ de même que le gain déterminés précédemment seront utilisés dans ce modèle, Simulink partageant les variables de l'environnement de Matlab (son `Workspace` en anglais dans le logiciel).
5. Ce modèle construit, retrouver la réponse indicielle du système en boucle fermée pour un échelon unitaire.

### 1.2 Correcteur numérique

1. Créer un nouveau modèle. Transformer l'asservissement analogique en un asservissement numérique par transposition du correcteur. Ajouter le convertisseur numérique analogique sur le schéma bloc. On supposera également que la chaîne directe intègre **un retard d'une période d'échantillonnage** servant à modéliser le temps de calcul de la commande.
2. Simuler la réponse indicielle du système asservi pour différentes périodes d'échantillonnage : 10, 25, 50, 100, 250 et 500 $ms$. Comparer ces réponses à celle l'asservissement analogique et commenter. Pour quelle période d'échantillonnage le système asservi devient-il instable ? Pour quelles périodes le comportement continu est-il bien approché ? Vérifiez la règle empirique vue en cours.

3. Nous faisons maintenant l'hypothèse que le système asservi est soumis à un bruit de mesure dans sa boucle de retour. Ce bruit est sinusoïdal, de fréquence 122 $Hz$ et d'amplitude unité (Attention, dans matlab, frequency signifie en fait pulsation). Comparer, dans ces conditions, la réponse indicielle du système asservi avec le correcteur analogique à celle du système asservi numériquement avec une période d'échantillonnage de 25 $ms$. Expliquer la différence de comportement face à ce bruit. (Attention, si vous observez un comportement erratique des signaux dans le cas analogique, ce peut être le signe que la résolution de calcul de simulink est insuffisante. Il faut alors modifier les paramètres de simulation dans le menu simulation).
4. Refaire l'expérience avec un bruit à 150 $Hz$ et commenter.
5. Ajouter un filtre de Butterworth (commande `butter` ) d'ordre 2 à l'asservissement numérique de manière à rejeter correctement les bruits de mesure. Vérifier l'efficacité du filtrage par un essai indiciel.

# 2 Rejet de perturbations

## 2.1 Rejet d'une perturbation d'entrée

Le convertisseur numérique/analogique (CNA) a un offset dépendant de la température ambiante. Cet offset (d'amplitude inconnue) peut-être modélisé comme une perturbation d'entrée constante sur la commande.

1. Créer un nouveau modèle. Simuler la réponse indicielle du système asservi avec le correcteur numérique précédent et un offset de 0,2 $V$.
   *On souhaite synthétiser un correcteur d'ordre réduit permettant de rejeter cette perturbation d'entrée. La synthèse sera faite par transposition d'un correcteur continu.*
2. A l'aide de `rltool`, déterminer un correcteur analogique du second ordre (au plus) tel que le système bouclé respecte le cahier des charges suivant :
   – erreur statique nulle malgré l'offset du CNA ;
   – dépassement inférieur à 15% ;
   – temps d'établissement à 5% inférieur à 1 $s$.
   Pour cela on notera que dans l'utilitaire `rltool`, il est possible d'ajouter des pôles et des zéros au correcteur (dans `Tools` → `Edit compensator` ou simplement en cliquant sur le bloc K). Il est également possible d'utiliser le bouton `Add zero` (flèche pointant un cercle) ou le bouton `Add pole` (flèche pointant une croix) puis de cliquer à l'endroit du plan complexe où on désire placer le zéro ou le pôle. Ces derniers peuvent être déplacés par la suite en cliquant sur le bouton drag pole/zero (flèche seule) ; le lieu d'Evans est alors réactualisé.
3. Appliquer la transformation bilinéaire :
$$s \to \frac{2}{T_e}\frac{z-1}{z+1}$$
   où $T_e$ est la période d'échantillonnage, pour passer du correcteur continu au correcteur numérique. Ceci est réalisé en utilisant la fonction `c2d` de Matlab . Expliquez comment choisir la période d'échantillonnage.
4. Simuler la réponse indicielle de l'asservissement numérique ainsi obtenu.

## 2.2 Effet de la quantification

Le convertisseur numérique/analogique (CNA) et le convertisseur analogique/numérique (CAN) introduisent respectivement une quantification de la commande et une quantification de la mesure de position. Nous faisons l'hypothèse que le pas de quantification est de 0,5.

1. Simuler l'effet de la quantification au niveau de la commande sur la réponse indicielle du système avec le correcteur numérique précédent. Utiliser pour cela le bloc `quantizer` de la bibliothèque non linéaire de Simulink .
2. Simuler l'effet de la quantification au niveau de la mesure (CAN) sur une réponse indicielle.
3. Expliquer pourquoi ces effets sont très différents. Si la mesure de position est réalisée avec un codeur incrémental, calculer quelle devrait être la résolution minimale de ce codeur pour que l'amplitude du bruit de quantification soit inférieure à un 0,1 $V$.

# TP 2 – Correction numérique d'un moteur à courant continu

*Le but de ce TP est de mettre en place des correcteurs numériques pour asservir la position d'une maquette construite autour d'un moteur à courant continu et d'un codeur optique incrémental.*

Le TP comporte plusieurs cahiers des charges à satisfaire. Pour chacun des jeux de contraintes les phases à réaliser sont les suivantes :

1. Synthèse d'un correcteur à l'aide de l'outil `rltool` de Matlab (lieu des racines) ;
2. Test du correcteur à l'aide de la boîte à outils de simulation `Simulink` ;
3. Codage en langage C de la loi de commande ;
4. Test du correcteur sur la maquette ;
5. Etude et analyse des résultats obtenus.

Cette démarche est la méthodologie classique de synthèse et d'implantation sur un système réel. Il est bien sûr inutile de passer d'une étape à la suivante sans l'avoir validée.

Les correcteurs seront implantés sur un PC muni d'une carte assurant l'interface avec le banc moteur. Pour ce qui concerne le logiciel, les fonctions d'entrée–sortie sont fournies, ainsi que le programme assurant le cadencement de la commande. Seul le correcteur est à programmer.

# 1 Préparation

## 1.1 Modélisation du système étudié

Le système étudié est un banc moteur Quanser. Ce banc est composé d'un moteur à courant continu de faible puissance ($< 40\ W$), sans réducteur, couplé à un volant d'inertie. Le codeur incrémental qui fournit la mesure de la position est couplé à l'arbre moteur. La grandeur de commande du système est la tension $U$ d'entrée d'un amplificateur à transistors qui est proportionnelle à la tension appliquée à l'induit du moteur.

La fonction de transfert théorique, reliant l'entrée $U$ du système à la mesure de position $\Theta$, s'écrit :

$$G(s) = \frac{\Theta(s)}{U(s)} = \frac{K}{s(1+\tau s)} \tag{1}$$

Les valeurs numériques sont $K = 3100\ deg.V^{-1}$ et $\tau = 0,08\ s$.

Le système est piloté à l'aide d'un ordinateur réalisant l'asservissement numérique. Le CNA utilisé est modélisé par un bloqueur d'ordre zéro $B_0(s)$. En supposant que l'on échantillonne de façon synchrone la commande et la mesure, et que le gain du CNA est unitaire, la représentation du système en boucle ouverte est celle décrite par la figure 1.

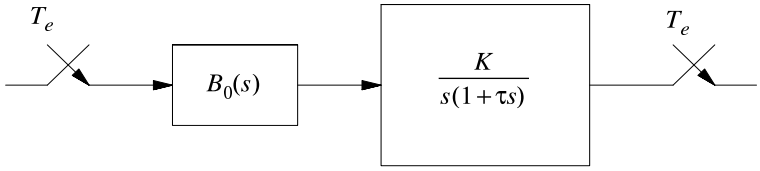


FIGURE 1 – Système à commander.

Le PC qui réalise l'asservissement temps réel est également celui qui est utilisé pour développer et compiler les logiciels. Cette machine possède une carte convertisseur numérique/analogique pour l'application des commandes ainsi qu'une carte compteur pour convertir les impulsions du codeur incrémental en position angulaire. Le système d'exploitation utilisé est `Xenomai` (linux temps-réel).

**Question** : Donner la fonction de transfert $G(z)$ entre la commande $U(z)$ et la sortie $\Theta(z)$, pour une période d'échantillonnage $T_e$ quelconque.

## 1.2 Algorithme de commande

On souhaite réaliser la commande numérique de ce système. La majorité du logiciel est déjà écrite. La partie réalisant la commande est globalement organisée conformément à la description de la figure 2.

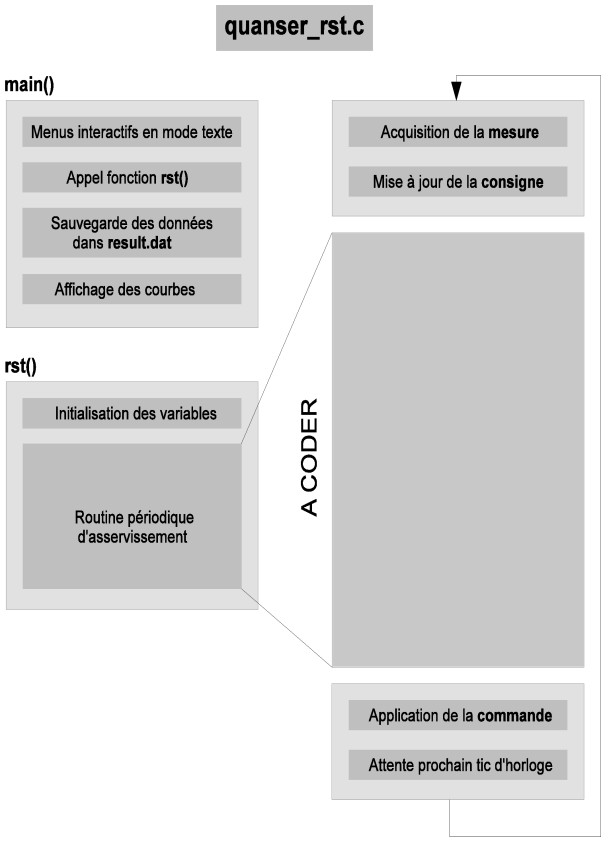


FIGURE 2 – Organigramme du logiciel de commande.

On supposera par la suite que le temps nécessaire pour effectuer la mesure, le calcul, et l'envoi de la commande est négligeable (très petit devant la période d'échantillonnage).

Le programme effectue des instructions en boucle infinie. La boucle est organisée en cinq étapes :

1. La mesure de la position est acquise au moyen d'un codeur optique incrémental. Les fonctions assurant la lecture du compteur et la conversion en degrés sont implémentées. La variable **mesure** contient la mesure courante.
2. La variable **consigne** donne la consigne courante. Cette consigne est définie par l'utilisateur avant le lancement de l'asservissement (échelon ou rampe).
3. La partie à coder est le calcul de la commande (variable **commande**).
   **On demande en préparation** d'écrire l'algorithme d'un correcteur numérique d'ordre $n$ en supposant que l'on connaît :
   – les $n+1$ coefficients des polynômes $N(z^{-1})$ et $D(z^{-1})$ (Numérateur et Dénominateur de la fonction de transfert du correcteur exprimés en puissances de $z^{-1}$), qui sont rangés dans des variables (de type tableau) $b$ et $a$ dans l'ordre des puissances croissantes de $z^{-1}$ :

$$\begin{aligned} N(z^{-1}) &= b[0] + b[1]z^{-1} + \ldots + b[n]z^{-n} \\ D(z^{-1}) &= a[0] + a[1]z^{-1} + \ldots + a[n]z^{-n} \end{aligned}$$

– les $n+1$ dernières mesures (dont la mesure courante) rangées dans le tableau $mes$, de telle sorte que :

$$mes[0] = \theta(k),\ mes[1] = \theta(k-1),\ \ldots$$

– les $n+1$ dernières consignes rangées dans le tableau $cons$, de telle sorte que :

$$cons[0] = \theta_d(k),\ cons[1] = \theta_d(k-1), \ldots$$

– les $n+1$ dernières commandes rangées dans le tableau $com$, de telle sorte que :

$$com[0] = u(k),\ com[1] = u(k-1),\ \ldots$$

L'algorithme doit calculer la commande $u(k)$ en fonction des données, stocker cette commande dans la variable **commande** et mettre à jour les différents éléments des tableaux.

4. La commande stockée dans **commande** est ensuite envoyée dans le CNA.
5. On attend enfin le prochain top d'horloge avant de reboucler (fonctions de gestion de l'horloge existantes en bibliothèque)

# 2 Manipulation

**ATTENTION : L'électronique du banc moteur n'est pas protégée. Veillez à ne pas toucher les contacts ou les composants de la carte électronique de la maquette. Une décharge électro-statique pourrait créer des dommages irréversibles.**

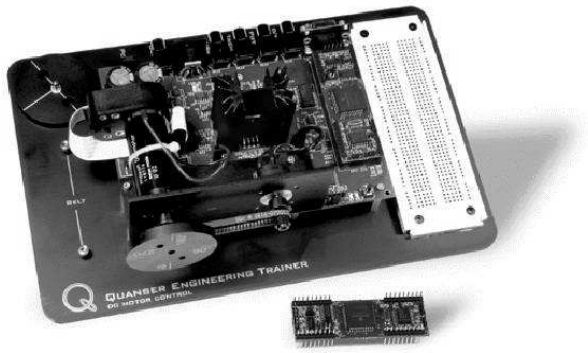


FIGURE 3 – Maquette Quanser QET.

## 2.1 Connexion informatique

A la fenêtre de login le nom d'utilisateur est `ensps` et le mot de passe `ulp`. Ouvrir un terminal en cliquant l'icône de la barre des tâches.

Taper la commande `.ini_quanser` pour initialiser le TP (ne pas oublier le point). Cette commande crée un répertoire nommé `quanser_rst` contenant plusieurs fichiers dont `iofunctions.h` (à ne pas modifier) et `quanser_rst.c`. Pour aller dans ce répertoire, taper `cd quanser_rst`. Pour éditer le fichier `quanser_rst.c`, taper la commande `nedit quanser_rst.c &`. Ce fichier contient le code du programme d'asservissement.

Lancer `Matlab` à partir de l'icône du bureau ou en tapant `matlab` dans un terminal. Cette version de `Matlab` ne dispose pas d'éditeur intégré. Vous pouvez directement entrer vos commandes dans le terminal `matlab` qui fait office de workspace ou écrire vos programmes en utilisant un éditeur externe tel que `nedit`. **Attention !** : Pour vous éviter de perdre du temps, n'écrasez pas vos schémas et résultats. Ils sont utilisés à plusieurs reprise tout au long du TP.

## 2.2 Calcul d'un correcteur par transposition

1. A l'aide de `rltool` déterminez un correcteur analogique d'ordre minimal tel que :
   – Le comportement en boucle fermée soit du second ordre avec $\omega_n = 25\ rad.s^{-1}$ et $\zeta = 0,7$.
   – L'erreur statique vis à vis de la consigne soit nulle.
2. Transposez ce correcteur en un correcteur numérique en utilisant l'approximation bilinéaire. La période d'échantillonnage est imposée à 10 $ms$. Cette période d'échantillonnage est-elle suffisante pour garantir un bon comportement de l'asservissement numérique ? Justifiez.
3. Simuler avec Simulink le comportement du système analogique asservi avec votre correcteur numérique. Comparez les résultats avec le comportement attendu. Comment expliquez vous les différences observées ?
4. Ecrivez la loi de commande temporelle correspondant à ce correcteur
5. Implémenter le correcteur en langage C. Seul le calcul de la commande doit être programmé, le reste est déjà fait (la partie à compléter est signalée par un commentaire) (voir figure 2 et 4 (TP3)). Il vous est également demandé de réaliser la saturation logicielle de la commande entre les valeurs `-CNA_SATURATION` et `CNA_SATURATION` qui sont les valeurs limites que peut transmettre le CNA. Pourquoi est-il important d'écrire ces saturations logicielles ? Que peut-il se passer si on ne sature pas correctement la commande ? **ATTENTION ! Vérifiez également bien la période d'échantillonnage du système définie en haut du fichier par la macro `#define SAMPLING_PERIOD` et exprimée en nano-secondes !**
   Pour compiler le programme `quanser_rst.c`, taper simplement `make`.
6. Tester le correcteur :
   – lancer le programme d'asservissement en tapant `quanser_rst` ;
   – à l'invite, entrer le type de consigne souhaitée (échelon ou rampe), l'amplitude ou la pente voulue et la durée de l'asservissement (après cette durée les commandes ne sont plus envoyées au système). La question du choix des coefficients rst concerne une autre manipulation et n'a donc pas d'importance ici : tapez le choix 2.
   – Tester le comportement du système pour différentes valeurs d'échelon (notamment pour de faibles amplitudes ($< 5^o$) et pour une rampe.
   Vous pouvez observer le comportement
   – directement sur la maquette ;
   – sur l'affichage écran à la fin de l'asservissement ;
   – en utilisant le fichier `result.dat` créé pendant l'essai et que vous pouvez charger dans le workspace `matlab` par la commande `load result.dat`.
7. Commenter les résultats obtenus. Quelles sont les caractéristiques problématiques de la réponse indicielle ? Pourquoi n'est-ce pas conforme à ce qui était prévu ? Quel type de perturbation (perturbation d'entrée, de sortie ou de mesure) est responsable de ce phénomène ? Quelle est leur origine physique ?

## 2.3 Synthèse directe en numérique

On veut réaliser l'asservissement du moteur selon le même cahier des charges que précédemment :
– Le comportement en boucle fermée soit du second ordre avec $\omega_n = 25\ rad.s^{-1}$ et $\zeta = 0,7$.
– L'erreur permanente vis à vis de l'entrée soit nulle.

En revanche, on propose de réaliser la synthèse du correcteur directement en numérique sans passer par une synthèse analogique.

La période d'échantillonnage choisie est toujours de 10 $ms$.

1. En utilisant la fonction `c2d` de Matlab et les paramètres adéquats, vérifiez le calcul de $G(z)$ fait en préparation.
2. Déterminez un correcteur à l'aide de `rltool`.
   Attention ! rltool est buggé pour les systèmes numériques. Lorsque vous cliquez sur les pôles de la boucle fermée, rltool affiche sous la fenêtre l'amortissement et la pulsation naturelle correspondants à la position des pôles. Les valeurs sont justes mais deviennent erronées dès que vous déplacez la souris. D'autre part les contraintes de pulsation sont fausses et ne doivent donc pas être utilisées.

3. Simuler avec Simulink le comportement du système analogique asservi avec votre correcteur numérique. Comparez les résultats avec le comportement attendu et avec les résultats obtenus avec la première synthèse (par transposition).
4. Ecrivez la loi de commande temporelle correspondant à ce correcteur
5. Implémenter le correcteur sur le système réel.
6. Tester le correcteur pour différentes valeurs d'échelon et pour une rampe.

## 2.4 Correction à faible fréquence d'échantillonnage

Pour des raisons de temps de calcul, il est parfois nécessaire de d'augmenter la période d'échantillonnage. Pour cette partie nous prendrons $T_e = 100ms$.

1. Reprenez le correcteur analogique synthétisé au début du TP et transposez le en numérique avec la nouvelle période d'échantillonnage. Testez le avec Simulink sur le modèle du système. Que se passe-t'il ? Quelle règle n'est plus respectée ?
2. Faites maintenant une synthèse directement en numérique avec la nouvelle période d'échantillonnage. Où sont les pôles de la boucle fermée ? Simulez l'effet de ce correcteur à l'aide de Simulink. Observez le signal de sortie et le signal de commande.
3. Implantez ce correcteur sur le système réel et testez-le (pensez à modifier la période d'échantillonnage du système).

## 2.5 Amélioration du correcteur

La période d'échantillonnage est maintenant reprise à $T_e = 10ms$. L'objectif de cette partie est de remédier au problème apparu lors du test des correcteurs sur le système réel : l'erreur statique n'est pas nulle. On considérera en première approximation que les frottements secs à l'origine de l'erreur statique peuvent être modélisés par une perturbation d'entrée constante.

- Synthétisez un nouveau correcteur directement en numérique vérifiant le nouveau cahier des charges :
  - Erreur statique vis-à-vis de la consigne nulle
  - Rejet des perturbations d'entrée d'ordre 0
  - Dépassement de l'ordre de 20%
  - Temps d'établissement à 5% inférieur à 300 ms

  Cette synthèse est délicate et il est utile de se référer aux conseils de l'utilisation du lieu d'Evans vus en cours. On prendra notamment soin de ne pas positionner de pôles du correcteur dans la partie réelle négative afin d'éviter des commandes alternées qui peuvent être néfastes à haute fréquence d'échantillonnage. On fera également attention à ne pas placer de zéro trop près de 1. Enfin, on notera que lorsqu'un système a 1 pôle réel et 2 pôles complexes conjugués, la configuration la plus rapide est obtenue lorsque les 3 pôles sont alignés (même valeur réelle).

  Pour faciliter la synthèse vous pouvez répondre aux questions d'aide suivantes :
  - Un correcteur d'ordre 1 est-il suffisant ?
  - Quel est l'ordre du système en boucle fermée ? Peut-on alors utiliser les abaques et les contraintes de rltool ?
  - Où faut-il placer les zéros du correcteur pour faire rentrer les branches dans le cercle unité ?
  - Où faut-il théoriquement placer les pôles de la boucle fermée pour ne pas avoir de dépassement ? Pourquoi cela ne fonctionne-t-il pas ici ? Quelle est l'origine du dépassement ?
- Testez ce correcteur à l'aide de Simulink. Vérifiez notamment que les perturbations d'entrée d'ordre zéro sont correctement rejetées et testez la réponse à des rampes.
- Implantez ce correcteur et testez le comportement du système en réponse à différents échelons et en réponse à des rampes.
- Commentez les résultats obtenus.

## 2.6 Question bonus : correcteur à réponse pile

On reprend $T_e = 0.1s$ et on fait la synthèse directement en numérique. Le cahier des charges est le suivant :

- Erreur statique nulle
- Temps de convergence minimal
- Où faut-il placer les pôles de la boucle fermée pour que la réponse soit la plus rapide possible ? Essayez de positionner les pôles de cette façon sans compenser les zéros du système.
- Déterminez ensuite la fonction de transfert de la boucle fermée. Déduisez-en l'équation aux différences qui relie entrée et sortie et calculez les premières valeurs de la sortie. En combien de pas d'échantillonnage la sortie converge-t'elle vers la valeur désirée ? Vérifiez à l'aide de `rltool`.
- Testez ce correcteur sur la maquette. **Attention !** : ce type de correcteur, appelé correcteur à réponse pile, a un comportement très violent et nécessite un modèle précis du procédé. Le test sera réalisé avec prudence (amplitudes réduites des échelons, temps d'asservissement court) et sous la supervision des encadrants afin de ne pas dégrader le matériel.

# TP 3 – Asservissement numérique de température

*L'objectif de ce TP est de réaliser un asservissement numérique de température.* Le système à réguler est une maquette d'un système thermique *Feedback*. Les particularités de ce système sont :
- une constante de temps lente ;
- un retard dû au transport de l'air.

## 1 Moyens

On dispose :
- d'une maquette *Feedback* ;
- d'un ordinateur compatible PC équipé du système d'exploitation temps réel (`xenomai`). Il possède par ailleurs une carte d'interface qui comporte en entrée un convertisseur analogique numérique et en sortie un convertisseur numérique analogique modélisé par un bloqueur d'ordre 0.
- un programme réalisant l'interface graphique entre l'utilisateur et le module d'asservissement (`temp`) ;
- Matlab .

Après démarrage du système, entrer le mot de passe `ulp`. Ouvrir un terminal en cliquant l'icône de la barre des tâches.

Taper la commande `ini_temp` pour initialiser le TP. Cette commande crée un répertoire nommé `temp` contenant, entre autres, le fichier `temp.c`. Pour aller dans ce répertoire, taper `cd temp`. Pour éditer le fichier `temp.c`, taper la commande `nedit temp.c &`. Ce fichier contient le code du programme d'asservissement. Seul le calcul de la commande doit être programmé, le reste est déjà fait (la partie à compléter est signalée par un commentaire) (voir figure 4).

Pour compiler `temp.c`, taper simplement `make`.

## 2 Expérimentation

La fonction de transfert du système lorsque l'ouverture du clapet est à 35 degrés ou *throttle* sur 4 (suivant le modèle de maquette) et lorsque le capteur est placé à l'extrémité du conduit est la suivante (résultat d'une identification par analyse de réponse indicielle) :

$$G(s) = \frac{e^{-0,18s}}{(1+0,25s)^2} \tag{2}$$

Dans ce qui suit, on suppose que le temps nécessaire au calcul de la commande est négligeable devant la période d'échantillonnage.

### 2.1 Méthode de synthèse basée sur le correcteur continu

Pour cette première synthèse, on déterminera tout d'abord un correcteur analogique qui sera ensuite transposé en numérique.

1. Donnez la forme générale d'un correcteur analogique d'ordre 2.
2. Faire la synthèse d'un correcteur $C(s)$ continu du second ordre tel que le système corrigé bouclé satisfasse au cahier des charges suivant :
   - Erreur permanente nulle pour une consigne en échelon,
   - Système plus rapide qu'en boucle ouverte. On compensera en particulier les pôles dominants du système,
   - Marge de phase de 45 deg.

   On demande de ne pas intégrer le retard dans la fonction de transfert mais d'ajouter "manuellement" son effet aux tableaux de valeurs obtenus par la fonction `bode` de `matlab`. Cette fonction trace les diagrammes de bode de la fonction de transfert G lorsqu'elle est appelée par :
   - bode(G)

   Elle retourne des tableaux de points pris sur les diagrammes lorsqu'elle est appelée par :

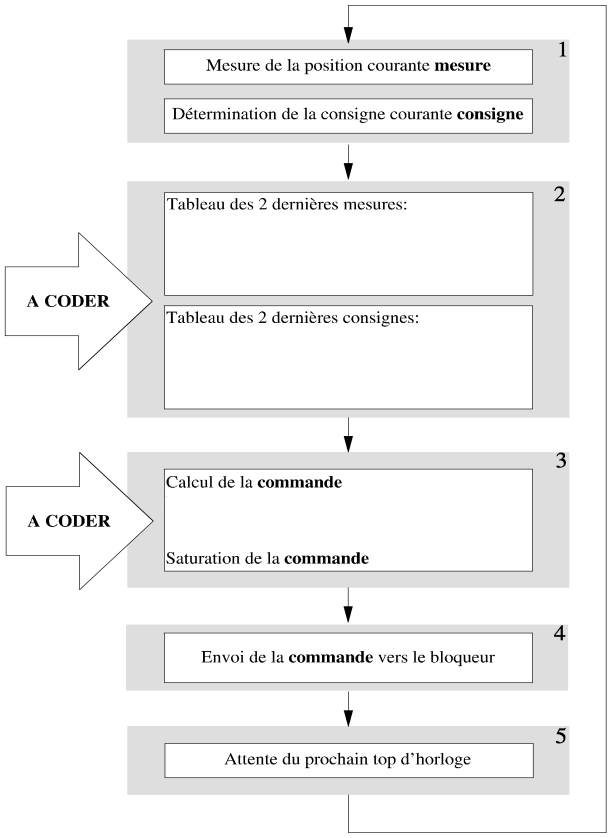


FIGURE 4 – Organigramme du logiciel de commande.

– [ ampl, ph, w ] = bode(G)
Lors de la synthèse du correcteur analogique vous devrez garder à l'esprit qu'il doit ensuite être transposé en numérique. La période d'échantillonnage minimale est de 10 ms.

3. Appliquer la transformation bilinéaire pour passer de $C(s)$ à $C(z)$ :

$$s \rightarrow \frac{2}{T_e} \frac{z-1}{z+1} \tag{3}$$

avec $T_e$, la période d'échantillonnage du correcteur numérique (fonction `c2d` de Matlab ). On choisira $T_e$ suffisamment faible pour bien approcher le correcteur continu (pas moins de 10 ms).

4. Etudier en simulation le système analogique avec son correcteur numérique en utilisant Simulink . Utiliser le bloc *"transport delay"* de la bibliothèque Simulink pour modéliser le retard.
5. Programmer le correcteur obtenu dans `temp.c` à l'emplacement prévu à cet effet. Ne pas oublier de définir la période d'échantillonnage en haut du programme à la ligne (`#define SAMPLING_PERIOD`). Compiler le module en tapant `make`.
6. Tester le correcteur en lançant `temp`. Commenter les résultats obtenus.

## 2.2 Synthèse directe du correcteur numérique

On souhaite réaliser une synthèse directement en numérique.

1. Calculer la transmittance échantillonnée $G(z)$ du système en boucle ouverte (commande `c2d` avec les arguments adéquats).
2. Expliquer pourquoi on peut utiliser le lieu des racines pour calculer un correcteur alors que cela n'était pas possible pour la synthèse par transposition.

3. Faire la synthèse d'un correcteur numérique du second ordre à l'aide du lieu des racines (fonction `rltool` de Matlab ) qui respecte le cahier des charges suivant :
   – Erreur permanente nulle pour une consigne en échelon,
   – Compensation des pôles dominants du système,
   – Dépassement nul,
   – temps de réponse le plus court possible.
4. Simuler le correcteur numérique sur le modèle du système analogique. Ensuite, implanter le correcteur numérique sur le système réel et mesurer les caractéristiques du système réel corrigé.

# Aide mémoire pour l'utilisation de Matlab et Simulink

## 1 Utilisation générale de matlab

Il existe deux versions de Matlab en salle de TPs d'automatique que vous serez amenés à utiliser. Matlab 6.5 sur les machines sous environnement Windows et Matlab 5.2 pour les machines sous environnement Linux. Les fonctions disponibles sont sensiblement les mêmes, mais l'utilisation du logiciel diffère.

### 1.1 Lancement

**Sous windows** : double cliquez sur l'icône Matlab6.5 sur le bureau. Matlab ouvre alors plusieurs fenêtres (ou une fenêtre contenant plusieurs sous-fenêtres). Les deux plus importantes sont la fenêtre de commande ("Command Window") et la fenêtre de l'éditeur de script.

**Sous linux** : cliquez sur l'icône Matlab. Matlab démarre dans un terminal qui devient la fenêtre de commande, repérable au prompt >>. Pour travailler avec des scripts ou des fonctions ".m", vous avez besoin d'un éditeur de texte séparé. Pour cela, cliquez sur l'icône de nedit ou tapez `nedit` dans un terminal.

## 2 Généralités

Matlab est un logiciel de calcul matriciel. Il peut être utilisé simplement comme une calculatrice pour tous les calculs scalaires et dispose de nombreuses fonctions de calcul matriciel (déterminant, inverse, etc.). En automatique, en plus des fonctions de base de calcul, vous utiliserez des fonctions appartenant à la boîte à outils "Control Toolbox" et à la boîte à outil "Simulink".

Il existe 2 principales façon d'utiliser Matlab :

– en ligne de commande : l'utilisateur tape des commandes dans la fenêtre de commande ("command Window" pour matlab 6.5 sous Windows). Celles-ci sont directement executées par matlab. Une commande se termine par un retour chariot ("entrée"). Il est possible d'exécuter plusieurs commandes successivement en les écrivant sur la même ligne et en les séparant par le caractère " ;". Lorsqu'une commande se termine par le caractère " ;" elle est muette : le résultat n'est pas affiché à l'écran. Le caractère % indique des commentaires. Tout ce qui se trouve après ce caractère sur la même ligne n'est pas interpreté par Matlab. Exemples :

```
>> a = 2  % Affectation de 2 à la variable a
  a =
2
>> b=1; % Affectation de 1 à la variable b sans affichage

>> a+b
ans =
3
>> c = a + b; d = c + 1;
>> d   % permet d'afficher le contenu de la variable d
d =
4
```

Les variables créées sont stockées dans l'espace de travail (workspace) et peuvent être réutilisées à tout moment.

En ligne de commande, les commandes tapées précédemment peuvent être rappelées en utilisant les flèches vers le haut et vers le bas.

– par l'intermédiaire de scripts ou de fonctions. Les commandes sont écrites dans un fichier texte de la même façon qu'en ligne de commande. Le fichier doit être sauvegardé avec une extension ".m" pour signifier qu'il s'agit d'un script Matlab. Pour exécuter ce script il suffit de taper son nom (sans l'extension ".m") dans la fenêtre de commande. Toutes les commandes du script sont alors exécutées successivement. Attention, pour que les fichiers ".m" puissent être lancés depuis matlab vous devez vous placer dans le répertoire où se trouve le fichier.
Exemples :
Fichier exemple.m **********
```
a = 2  % Affectation de 2 à la variable a
b=1; % Affectation de 1 à la variable b sans affichage
a+b
c = a + b; d = c + 1;
d   % permet d'afficher le contenu de la variable d
**********
>> exemple

a =
2
ans =
3
d =
4
```
Les variables créées dans un script sont stockées dans le workspace lors de l'exécution du script. Elles peuvent donc ensuite être utilisées en ligne de commande.
Afin de garder simplement une trace de vos travaux, il est conseillé de travailler à l'aide de scripts.

Matlab dispose d'une aide en ligne permettant d'obtenir les détails d'utilisation d'une commande : paramètres d'entrée, arguments de sortie, etc. Par exemple, pour obtenir des informations sur la commande d'inversion de matrice "inv", il suffit de taper

```
>> help inv
```

### 2.1 Quelques fonctions mathématiques utiles

– cos, sin, tan
– acos, atan, asin
– exponentielle : exp
– logarithme : Attention !! logarithme népérien : log, logarithme décimal : log10

## 3 Fonctions de la "Control toolbox"

Pour utiliser ces fonctions, il suffit de les taper soit dans la fenêtre de commande, soit dans un script.
Voici la plupart des fonctions qui vous seront utiles en TP.
– créer une fonction de transfert : ex : $G(s) = \frac{s+2}{s^2+3s}$
```
>> G = tf([1 2], [1 3 0])
```
– créer une fonction de transfert numérique : ex : $Gz(z) = \frac{z+0.5}{z-0.2}$
```
>> Gz = tf([1 0.5], [1 -0.2], Te)
```
où Te est la valeur de la période d'échantillonnage.
– créer une fonction de transfert par les pôles, zéros et gain : ex : $G(s) = \frac{5(s+2)}{s^2+3s} = \frac{5(s+2)}{(s+3)s}$
```
>> G = zpk([-2], [0 -3], 5)
```
– mettre une fonction de transfert G sous forme de pôles et zéros
```
>> zpk(G)
```
– Transposer une fonction de transfert continue en numérique
```
>> Gz = c2d(G, Te, 'method')
```

où Te est la période d'échantillonnage et où method est la méthode de transposition choisie (zoh, tustin, prewarp ou matched)

- Tracer la réponse indicielle d'un système donné par sa fonction de transfert G
  ```
  >> step(G)
  ```
- Obtenir la carte des pôles et des zéros d'un système
  ```
  >> pzmap(G)
  ```
- Diagramme de bode :
  ```
  >> bode(G) %trace les diagrammes de Bode de G

  >> bode(G, {w_min, w_max})  % trace les diagrammes entre les pulsations
   %w_min et w_max
  %par exemple
  >> bode(G, {0.1, 100})
  %trace les diagrammes de Bode entre 0.1 et 100 rad/s

  >> [gain, phase, w] = bode(G) % Ne trace pas les courbes, mais rend trois tableaux :
  %le gain sans dimension (pas en décibels) et la phase (en degrés)
  %déterminés aux pulsations w
  ```
- Diagramme de Nyquist :
  ```
  >> Nyquist(G)
  ```

  Trace le diagramme de Nyquist. Attention, les éventuels cercles à l'infini ne sont pas représentés. Il faut donc analyser la fonction de transfert pour pouvoir déterminer avec certitude la stabilité d'une boucle fermée.
- Lieu des racines :
  ```
  >> rltool(G)
  ```
  Trace le lieu des racines de G, avec retour unitaire. Les pôles et zéros du système G sont représentés par des croix et cercles bleus. La position des pôles de la boucle fermée est représentée par des carrés rouges. En cliquant sur ces carrés, il est possible de les déplacer le long du lieu d'Evans. Le gain affiché en haut de la fenêtre est alors modifié simultanément.
  `rltool` est un utilitaire permettant de régler un correcteur par la méthode du lieu des racines. Une fois le lieu des racines de G tracé, l'utilisateur a de nombreux outils.
  Il est ainsi possible d'ajouter des pôles et des zéros sur la carte des pôles et zéros. Ces zéros et pôles sont affectés au correcteur. Ils peuvent être déplacés par des "cliqué - glissé".
  Pour éditer le correcteur, cliquer sur le bloc C rouge (ou K cyan selon la version) dans le schéma bloc représenté en haut à droite
  On peut afficher les abaques d'iso-amortissement et d'iso-pulsation sur la carte des pôles soit en cliquant "grid on" (version Linux), soit en cliquant avec le bouton droit de la souris et en choisissant afficher la grille.
  - file → import : permet d'affecter des fonctions de transfert aux différents éléments du modèle. Ces fonctions de transfert doivent avoir été définies dans le workspace. Attention, les blocs de gain doivent être entrés sous la forme de fonction de transfert (par exemple, pour H = 5, il faut entrer H = tf([5], [1]))
  - file → export : permet d'exporter un bloc vers le workspace (par exemple le correcteur)
  - Analysis → response to step command (ou cliquer sur le bouton "step" en bas à gauche) : trace la réponse indicielle du système en boucle fermée pour le gain donné.
  - Clic droit dans la fenêtre rltool (ou tools → Add grid / boundary) : permet de définir des contraintes pour le système bouclé : dépassement, temps d'établissement, etc. Attention : ces contraintes ne sont valables que si le système en BF est équivalent à un système du $2^{eme}$ ordre.

# 4 Utilisation de Simulink

Simulink est un utilitaire de simulation permettant de représenter les systèmes à partir de schémas bloc. Pour lancer simulink, tapez

```
>> simulink
```

dans la fenêtre de commande.

L'utilisation de Simulink est assez intuitive. Elle consiste à sélectionner des blocs représentant des fonctions de transfert, des gains, etc. et à les glisser sur le schéma de simulation. Les blocs sont reliés entre eux par des traits orientés tracés à l'aide de la souris.

- file → new → model : crée un nouveau modèle simulink
- Les fonctions de transfert continues se trouvent dans le menu "Continuous" ou "Linear"
- Les fonctions de transfert numériques se trouvent dans le menu "discrete" ou "discontinuous"
- Les convertisseurs numériques analogiques (BOZ) sont dans le menu "discrete" ou "discontinuous"
- Les gains sont dans le menu "math operations" ou "Linear"
- Les comparateurs sont dans le menu "math operations" ou "Linear"
- Les outils de mesure (scope) sont dans le menu "sink"
- Les sources (échelons (step), rampes, etc.) sont dans le menu "source"
- Les retards à utiliser sont appelés "transport delay" et se trouvent (et c'est une erreur !) dans le menu "non linear" ou "Continuous"
- Les saturations sont dans le menu "non linear"
- Les quantificateurs sont dans le menu "non linear"

**Remarques**

- Il n'y a pas de bloc "CAN". Simulink détermine la nature des signaux en fonction de la nature des blocs dans lesquels ils entrent. Il est nécessaire pour les blocs de nature "numérique" de spécifier la période d'échantillonnage en double cliquant dessus. Attention, Matlab n'est pas dérangé par des périodes d'échantillonnage diférentes selon les blocs et ne vous avertira pas en cas d'erreur.
- Par défaut, les échelons démarrent à l'instant $t = 1s$. Il est parfois utile de modifier en $t = 0$.

Une fois les blocs placés sur le schéma, vous pouvez les modifier en double cliquant dessus. Par exemple, en cliquant sur un bloc fonction de transfert vous pouvez ajouter des pôles et des zéros, modifier les gains, etc.

Il est possible d'utiliser dans ces blocs des variables définies dans le workspace. Par exemple, si vous avez défini

```
>> K = 5
```

alors en mettant K dans un bloc simulink, sa valeur sera 5

La simulation est ensuite lancée par simulation→start. Les résultats de la simulation peuvent être obtenus en double cliquant sur les scopes. Une fenêtre s'ouvre alors avec les tracés des signaux mesurés.

Les paramètres de la simulation peuvent être modifiés en faisant simulation → parameters. On peut notamment modifier la durée de la simulation. Lorsque les courbes des "scopes" sont tronquées, il suffit d'ouvrir le bloc "scope" et d'augmenter le paramètre "Data history" de l'onglet "Settings".

# 5 Dangers

Les versions de Matlab disponibles en salle de TP comportent plusieurs erreurs, bugs ou difficultés qu'il convient de connaître. Parmi celles-ci vous rencontrerez les suivantes :

- Dans Simulink , rltool et les fonctions de la control toolbox, ce qui est appelé `frequency` correspond à des **pulsations** (donc en rad/s).
- Pour les simulations de systèmes continus, il est important d'utiliser une résolution de calcul suffisante. Lorsque les signaux observés semblent bruités (bruit de résolution numérique des équations différentielles régissant les systèmes analogiques), les paramètres de simulation peuvent être modifiés dans le menu simulation → simulation parameters, puis dans l'onglet "solver".
- les blocs retard ("transport delay") se trouvent pour certaines versions dans les bibliothèques "non linear". Il s'agit d'une erreur car les retards sont des systèmes linéaires.
- Sur les versions 5.2 de Matlab sous linux, l'affichage des amortissements et pulsations propres par rltool est buggé pour les systèmes numériques. Lorsqu'on clique sur les pôles de la BF, l'affichage est juste mais il devient faux dès qu'on déplace la souris. De plus les contraintes de pulsations pour les systèmes numériques sont fausses et ne doivent donc pas être utilisées.