	BUT2 **SAE FOUR** *2022 - 2023*	**Modélisation, Identification** **Correcteur PID**

Objectifs :

1. Savoir construire la consigne de température
2. Programmer un correcteur PID numérique
3. Modéliser le four et estimer les paramètres du modèle
4. Régler le correcteur PID

Table des matières

1 Objectifs

Cette séance de 4h est un TP dans lequel vous allez programmer une correction PID et générer une consigne.

Délivrables/Evaluation :

1. Génération de la consigne toutes les secondes (10 points)
2. Programmation du correcteur (10 points)
3. Teste du fonctionnement du four en BF avec une correction proportionnelle (10 points)
4. Mesure des performances en BF avec une correction proportionnelle (10 points)

2 Introduction

Pour souder des composants CMS on peut utiliser différentes pâtes à souder qui nécessitent le suivi d'un profil de température adapté à la composition de la pâte. Ce profil de température correspond à la consigne.

Pour que la température du four soit la plus proche possible de la consigne il faut obligatoirement travailler en boucle fermée :

1. On mesure la température du four

2. En fonction de l'écart entre la consigne et la température mesurée on décide de chauffer plus ou moins ou bien de refroidir. Dit autrement on adapte la commande du four en fonction de l'écart. C'est le correcteur qui va construire cette commande

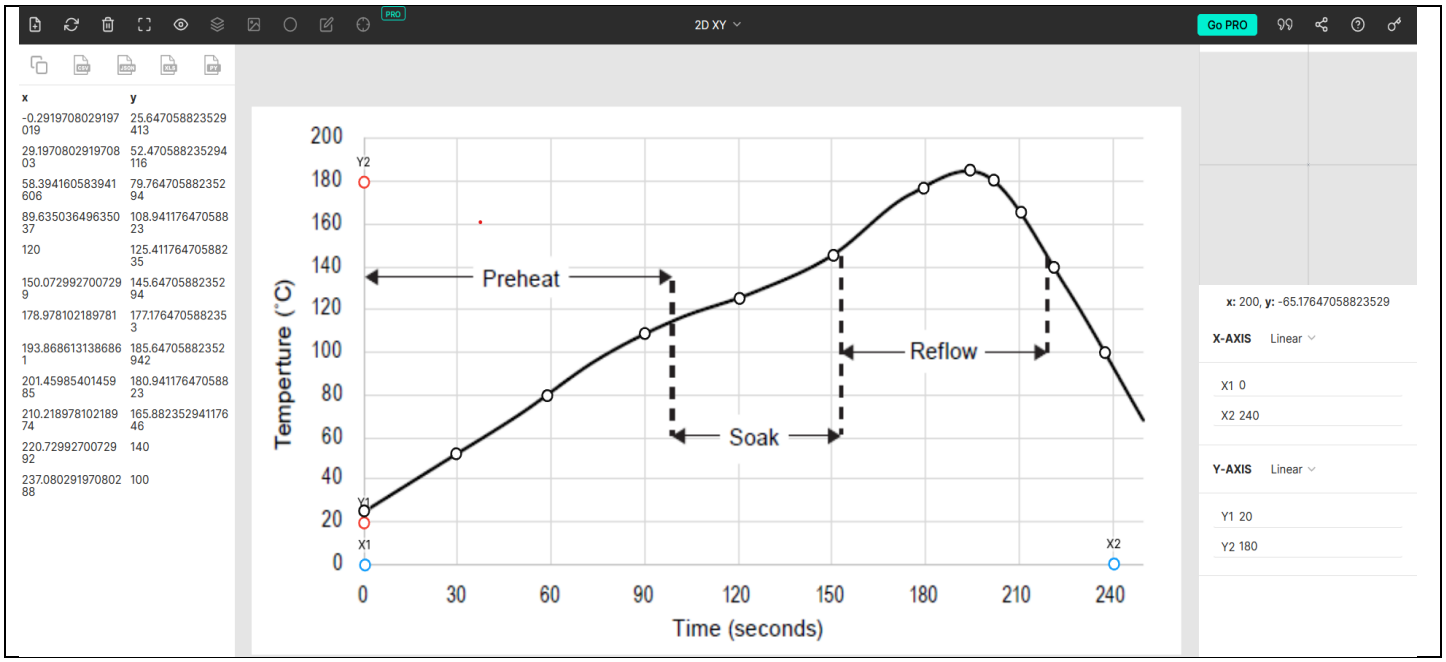
Les différentes étapes conduisant au fonctionnement en boucle fermée sont :

1. Génération de la consigne
2. Programmation du correcteur
3. Réglage du correcteur en fonction des caractéristiques du four et des performances attendues

3 Génération de la consigne

Dans les documentations techniques les constructeurs de pâte à souder fournissent le profil de température sous la forme d'une figure dont l'abscisse représente le temps et l'ordonnée la température. Pour générer la consigne de température il faut que l'on ait un tableau de points. Pour passer de la figure au tableau de point nous allons utiliser une application en ligne : https://plotdigitizer.com/app

A partir de la documentation technique on fait une capture d'écran de la courbe de température et on importe cette figure dans l'application.



On positionne les repères x1 et x2 (bleu) sur l'axe des abscisses et y1 et y2 (rouge) sur l'axe des ordonnées. Dans la zone à droite on entre les temps correspondants aux positions des repères x1 et x2. Par exemple x1 :0 et x2 :240. On fait de même pour les températures correspondantes aux positions des repères y1 et y2. Ensuite on positionne la souris sur la courbe pour sélectionner des points. Il apparait alors des ronds noirs et dans la zone de gauches les coordonnées de ces points. On sauvegarde ces coordonnées puis on arrondi les valeurs à l'entier le plus proche (on peut le faire « manuellement »)

A	B
x	y
0	25
30	53
58	80
90	109
120	125
150	145
180	177
193	186
201	181
210	166
220	140
237	100

On peut maintenant entrer ces valeurs dans le programme gérant le fonctionnement du four sous forme d'un tableau en C pour générer la consigne. Mais il y a un problème.

En effet la commande du four va être réactualisée toute les Te secondes (par exemple toutes les secondes). Dans notre tableau de valeurs on a par exemple la valeur de la consigne au temps 90 [s] (qui est 109°C) et au temps 120[s] (qui vaut 125°C) mais on ne connait pas les valeurs de la consigne au temps 91, 92, 93, 94, …….119. Il faut utiliser une fonction qui calcule ces valeurs intermédiaires en faisant une interpolation linéaire. Mais vous savez faire, vous l'avez utilisée précédemment dans la chaine de mesure de la température.

A FAIRE :

- Génération du tableau « consigne » en utilisant l'application plotdigitizer
- Génération dans le microcontrôleur de la consigne toutes les Te secondes (par exemple Te=1[s])

4 Programmation du correcteur

Le correcteur doit construire la commande en utilisant la consigne et la valeur mesurée de la température.

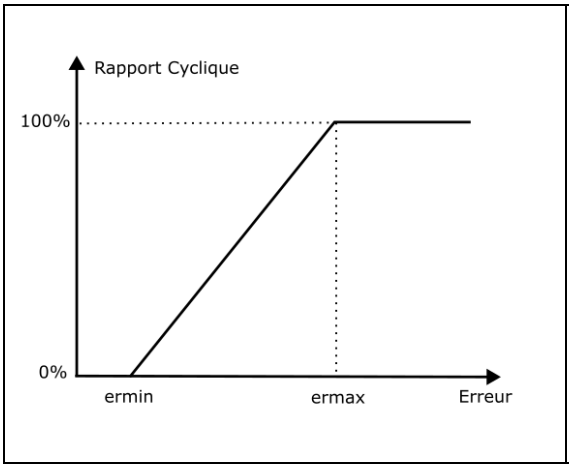
4.1 Correction proportionnelle

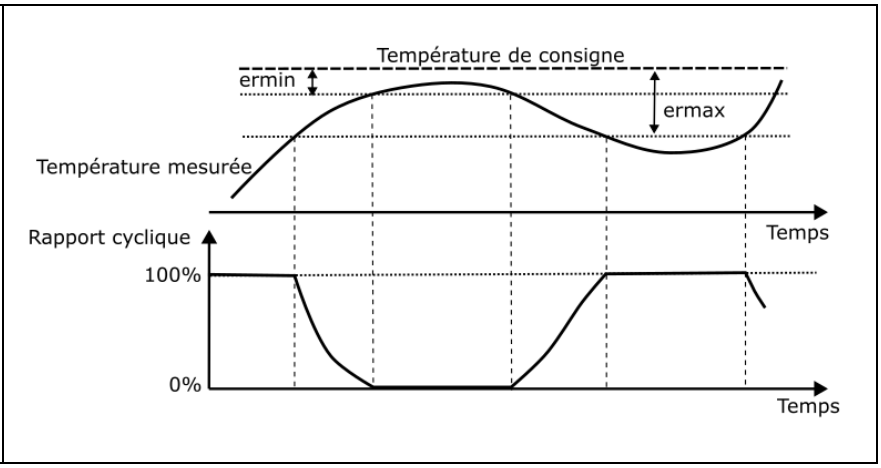
La correction proportionnelle est la correction la plus simple qu'il faut essayer en premier. Nous allons utiliser la définition industrielle de ce correcteur dite « Proportionnal Band »

Pour notre projet l'entrée du correcteur correspond à l'erreur (notée er(t)) soit la différence entre la température de consigne et la température mesurée. La sortie du correcteur correspond au rapport cyclique du signal pilotant le relai statique de l'élément chauffant.

Les deux paramètres du correcteur sont : l'erreur minimale (ermin) et l'erreur maximale (ermax). Si l'erreur est supérieure à ermax alors le rapport cyclique est fixé à 100% et si l'erreur est inférieure à ermin alors le rapport cyclique est fixé à 0%. Si l'erreur est comprise entre ermin et ermax alors le rapport cyclique (r%) vaut :

$$r\% = 100(er(t) - ermin)/(ermax - ermin)$$





Il faut déterminer expérimentalement les valeurs de ermin et ermax. On peut par exemple essayer ermin = 5[°C] et ermax = 30 [°C]

A FAIRE :

1. Programmer le correcteur et le tester
2. Faire des essais avec le four pour différentes valeurs de ermin et ermax. Sauvegarder les données dans des fichiers.
3. Analyser les performances obtenues

4.2 PID sous la forme « parallèle »

Lorsque la correction proportionnelle ne donne pas les performances attendues en boucle fermée il faut utiliser une correction PID. Cette correction utilise la combinaison de trois actions : l'action Proportionnelle, l'action Intégrale et l'action Dérivée

Cette correction PID peut s'écrire sous plusieurs formes : la forme « série » et la forme « parallèle ». Ici on utilisera la correction parallèle.

Nous allons nous intéresser à la réalisation numérique d'une correction PID (sous la forme « parallèle) dont la fonction de transfert (U(p)=commande et er(p)=erreur) est

$$PID(p) = \frac{U(p)}{er(p)} = G * (1 + \frac{1}{Ti * p} + \frac{Td * p}{1 + a * Td * p})$$

C'est la somme d'une action proportionnelle, d'une action intégrale et d'une action dérivée-filtrée

Le calcul de la commande suit les étapes suivantes :

1. Action intégrale

$$v_I(k) = v_I(k-1) + aw * \frac{Te}{Ti} * er(k)$$

- La variable $v_I(k)$ représente la valeur de l'action intégrale calculée à l'instant courant kTe et $v_I(k-1)$ celle calculée à l'instant précédent (k-1)Te
- La variable er(k) représente la valeur de l'erreur calculée à l'instant courant kT_e
- La variable aw vaut 0 ou 1. Si le signal de commande a atteint ses limites c'est-à-dire 0% ou bien 100% alors aw=0. Dans le cas contraire aw=1.

2. Action dérivée-filtrée

$$v_D(k) = \frac{1}{1 + a * \frac{Td}{Te}} \left(a * \frac{Td}{Te} * v_D(k-1) + \frac{Td}{Te} * \left(er(k) - er(k-1) \right) \right)$$

- Les variables er(k) et er(k-1) représentent les valeurs de l'erreur à l'instant courant kTe et à l'instant précédent (k-1)Te
- Les variables $v_D(k)$ représente la valeur de l'action dérivée-filtrée calculée à l'instant courant kTe et $v_D(k-1)$ celle calculée à l'instant précédent (k-1)Te
-

3. Construction du signal de commande

$$v_c(k) = G * \left(1 + v_I(k) + v_D(k) \right)$$

Remarque importante : Si on ne veut pas utiliser l'action dérivée il suffit d'imposer Td=0. Mais si on ne veut pas utiliser l'action intégrale il faudrait mettre Ti égal à l'infini !! On ne va pas faire cela. On remarque que prendre Ti=0 n'est pas possible (pourquoi ??). Donc on va appliquer la règle suivante : si l'utilisateur choisit Ti=0 c'est qu'il ne veut pas utiliser l'action intégrale. Donc dans le programme on va rajouter une structure condition : si Ti=0 alors $v_I(k)=0$ sinon $v_I(k) = av_I(k-1) + aw\frac{Te}{Ti}er(k)$

A FAIRE :
- Programmation du PID sous forme parallèle

5 Réalisation du montage en boucle fermée

La réalisation du montage en boucle fermée passe par le choix du correcteur et le réglage de ses paramètres. Pour pouvoir régler les paramètres du correcteur il faut avoir des informations sur le comportement du système que l'on commande. Mais il faut également définir les performances que l'on désire avoir en boucle fermée

5.1 Modélisation et identification

Cette étape permet de capturer les informations sur le comportement du système.

Premièrement il faut choisir un modèle permettant de reproduire au mieux le comportement du système : modèle du premier ordre, modèle du premier ordre avec retard (FirstOrderPlusTimeDelay), modèle du second ordre, modèle du second ordre avec retard (SecondOrderPlusTimeDelay),…. On choisira le modèle le plus simple c'est-à-dire avec le moins de paramètres

Deuxièmement il faut déterminer les valeurs numériques des paramètres du modèle

Pour faire ce qui vient d'être décrit il faut que l'on ait observé la réponse du système lorsque l'on a fait varier la commande suivant une forme déterminée : par exemple un échelon, une rampe, une sinusoïde, ….

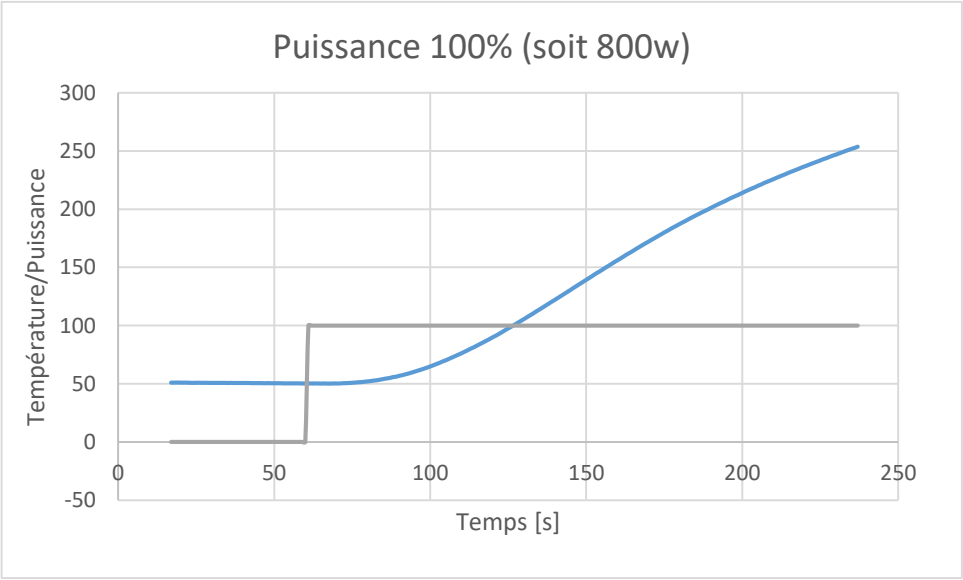
Donc concrètement la première action à réaliser est d'enregistrer l'évolution de la température lorsque la puissance des éléments chauffants varie

5.2 Acquisition de réponse expérimentale

Cette étape va nous permettre de construire un modèle comportemental du four. Mais elle nous permettra également de vérifier que la température du four peut suivre le profil de température désirée. Plus précisément si le profil de température passe par une température maximale d'environ 250°C, le four peut-il atteindre cette température ? De même si le profil de température passe d'une température de 140°C à 180°C en 30[s] la température du four peut-elle suivre cette variation ?

On doit donc lorsque la puissance dissipée est maximale savoir si la température peut atteindre 250°C et on doit mesurer la vitesse de variation de la température

Voici un exemple de relevé expérimental obtenu sur un four :

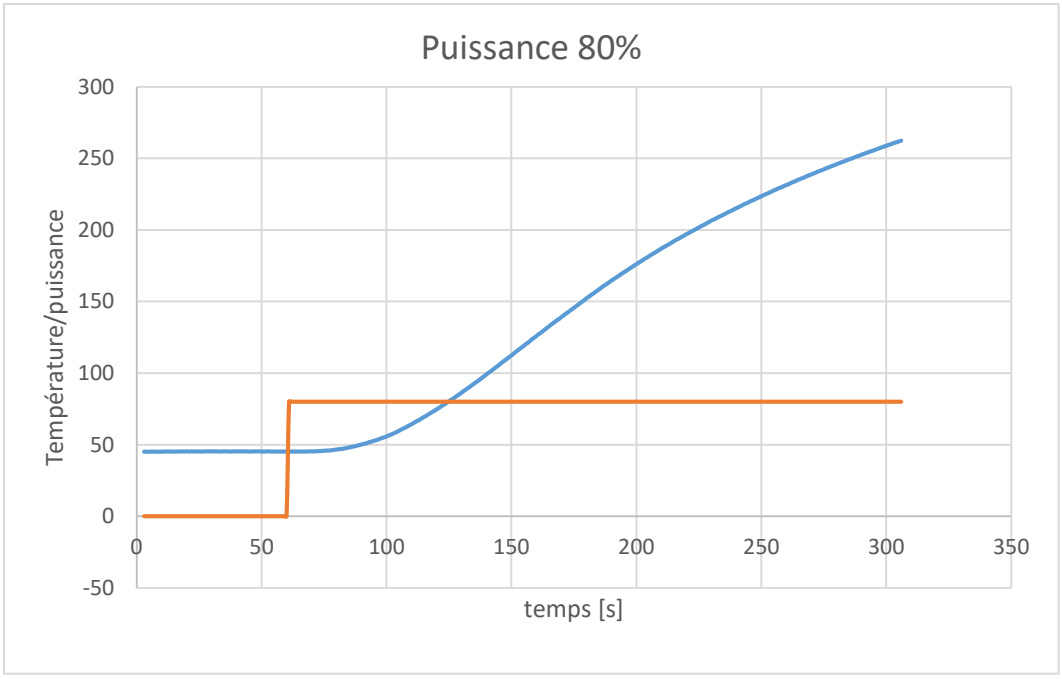


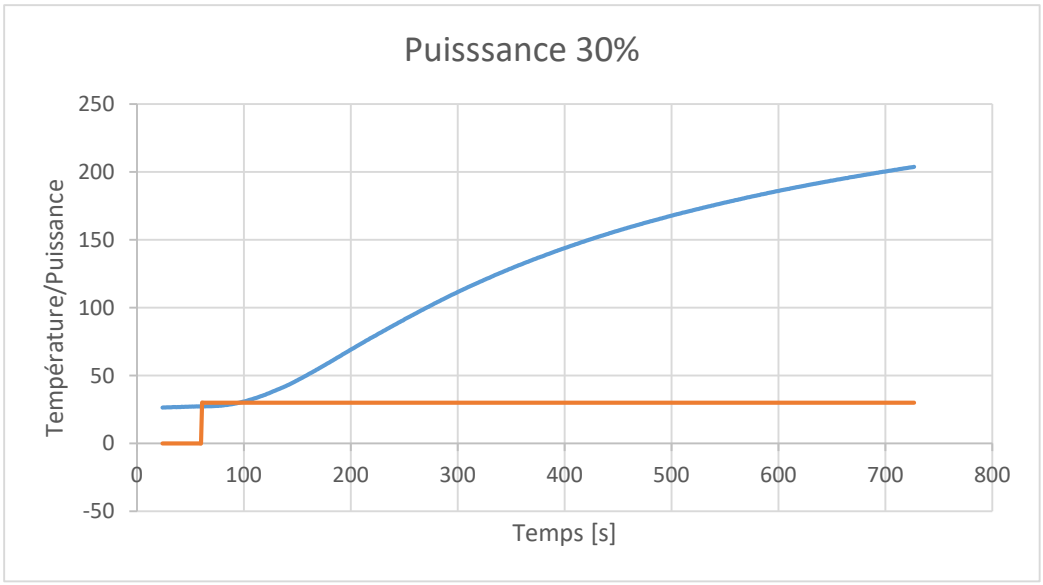
On constate donc que l'on peut atteindre la température de 250°C et que l'on a une pente moyenne de 70/50[°C/s]=1.4[°C/s]. La pente imposée par le profil étant de 40/30[°C/s]=1.33[°C/s]. Ouf !! c'est faisable.

Remarque importante : Dans l'exemple précédent la durée de l'acquisition était d'environ 250[s] soit 4[min], c'est long ! Lorsque l'on va faire des acquisitions avec des puissances inférieures à 100% cela sera encore plus

long !!!!! Il faut donc s'assurer que toute la chaine d'acquisition fonctionne correctement avant de faire une acquisition complète.

Relevés expérimentaux avec des puissances de 80% et 30% :





5.3 Choix du modèle

En observant les réponses on remarque que la pente à l'origine de la température est nulle on ne peut donc pas choisir comme modèle une fonction de transfert du premier ordre.

On va donc travailler avec un modèle possédant la fonction de transfert

$$F_A(p) = \frac{K}{(1+\tau_1 p)(1+\tau_2 p)}$$

Ou bien avec un modèle possédant la fonction de transfert

$$F_B(p) = \frac{Ke^{-d*p}}{1+\tau p}$$

Dans la suite on prendra $F_B(p)$ comme modèle car on veut utiliser la méthode AMIGO pour régler le PID.

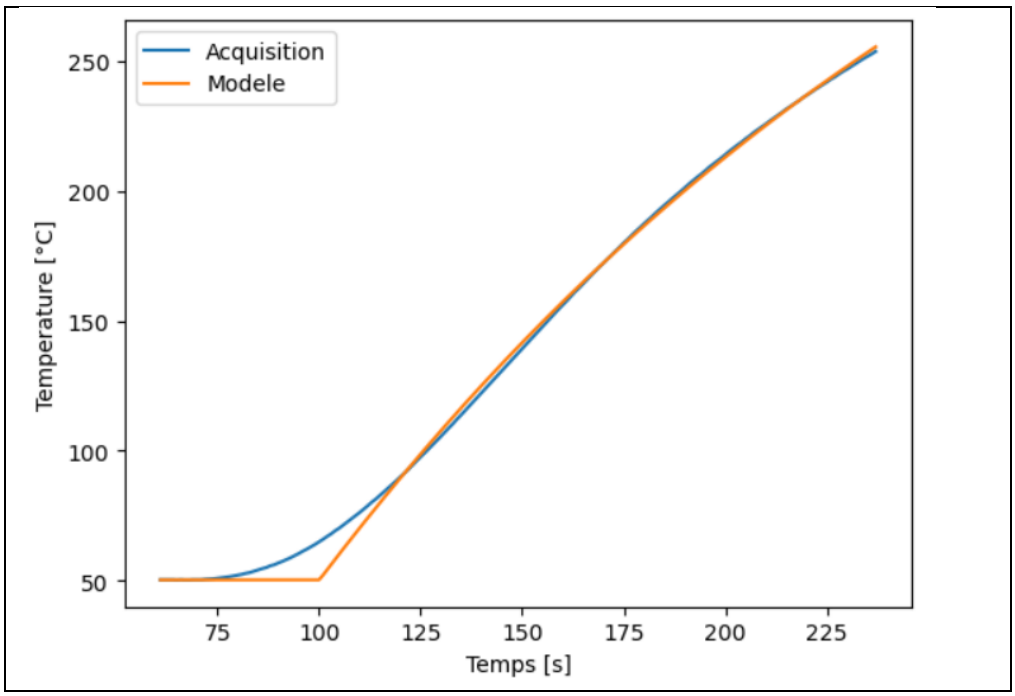
5.4 Estimation des paramètres

Bon, on a une « petite » idée du modèle maintenant. Il reste à déterminer les valeurs numériques des paramètres du modèle.

La première difficulté est que l'on n'a pas la valeur à laquelle la température se stabilise car elle est trop élevée. C'est un problème car d'habitude on trouve la valeur de K en faisant le rapport entre la variation de la température (température finale stable – température initiale stable) et la variation de la commande (par exemple : 100%=1, 80%=0.8, 30%=0.3). De plus pour mesurer la constante de temps τ présente dans $F_B(p)$ on a envie d'utiliser la règle des 63% mais comme on ne connait pas la valeur finale (le 100%) on ne peut pas. Bref cela ne vas pas être simple !

Pour avancer on vous donne un notebook Jupyter (application disponible dans Anaconda) (Identification_SAE_Four.ipynb) qui va lire dans un fichier Excel les valeurs enregistrées du temps, de la température et de la commande. Pour ensuite calculer les paramètres du modèle en minimisant l'écart quadratique moyen entre les températures enregistrées et celles données par le modèle pour la même valeur de la commande

Par exemple avec les données enregistrées pour une variation de la commande de 100% on obtient les paramètres suivants : K=4.1, τ=200[s] et d=39[s]



Il faut faire l'estimation des paramètres pour différentes valeurs de la commande par exemple 100%, 80%, 60%.

On ne trouvera pas exactement les mêmes valeurs pour les paramètres du modèle c'est tout à fait normal. Car ce n'est qu'un modèle, les conditions expérimentales ne sont pas identiques et les paramètres sont déterminés à l'aide d'un algorithme de minimisation dont le résultat dépend de son initialisation

Pour la suite on prendra comme paramètres du modèle une moyenne des paramètres obtenus.

Remarque importante : Avant de faire varier la valeur de la commande il faut s'assurer que la température soit la plus stable possible. La température ne doit donc pas être en train de varier notablement.

5.5 Réglage du correcteur

Nous allons utiliser une méthode empirique qui donne en général de bons résultats. Cette méthode s'appelle AMIGO (Astrom, K.J., and Hagglund, T. (2004), "Revisiting the Ziegler-Nichols step response method for PID control," Journal of Process Control, pp. 635–650)

La structure du correcteur PID (dite forme ISA) utilisé pour la méthode AMIGO est

$$PID(p) = G\left(1 + \frac{1}{T_i p} + \frac{T_d p}{1 + aT_d p}\right) \quad avec\ a \ll 1$$

Une fois que l'on a déterminer les paramètres (K, τ, d) du modèle $F_B(p)$ on applique les règles suivantes :

PID :	**PI :**
$G = \frac{1}{K}\left(0.2 + 0.45\ \frac{\tau}{d}\right)$ $T_i = \frac{0.4 * d + 0.8 * \tau}{d + 0.1 * \tau} * d$ $T_d = \frac{0.5 * \tau * d}{0.3 * d + \tau}$	$G = \frac{1}{K}\left(0.15 + 0.35\ \frac{\tau}{d} - \frac{\tau^2}{(d+\tau)^2}\right)$ $T_i = \left(0.35 + \frac{13 * \tau^2}{\tau^2 + 12 * \tau * d + 7 * d^2}\right) * d$ $T_d = 0$

Remarque : On prendra a=0.1

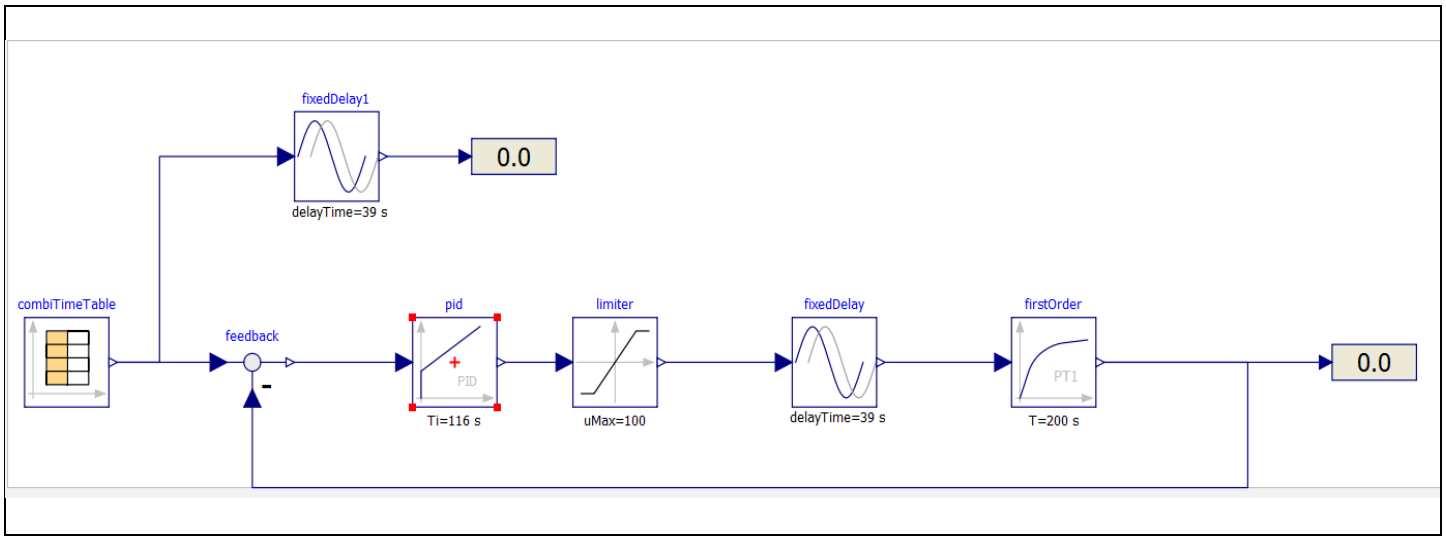
A FAIRE (et à refaire si nécessaire) :

- Acquisition de données expérimentales
- Identification des paramètres du correcteur
- Mesure des performances en boucle fermée
- Suivi d'une consigne de température permettant la soudure d'une carte CMS

5.6 Simulation

Une fois le modèle trouvé et le correcteur déterminé on va simuler le comportement du montage en boucle fermée. Il faut bien garder à l'esprit que l'on va utiliser le modèle pour faire la simulation donc il ne faut pas s'attendre à avoir les mêmes performances avec le four réel, tout va dépendre de la qualité du modèle que vous avez obtenu.

Pour faire la simulation vous allez utiliser OpenModelica.

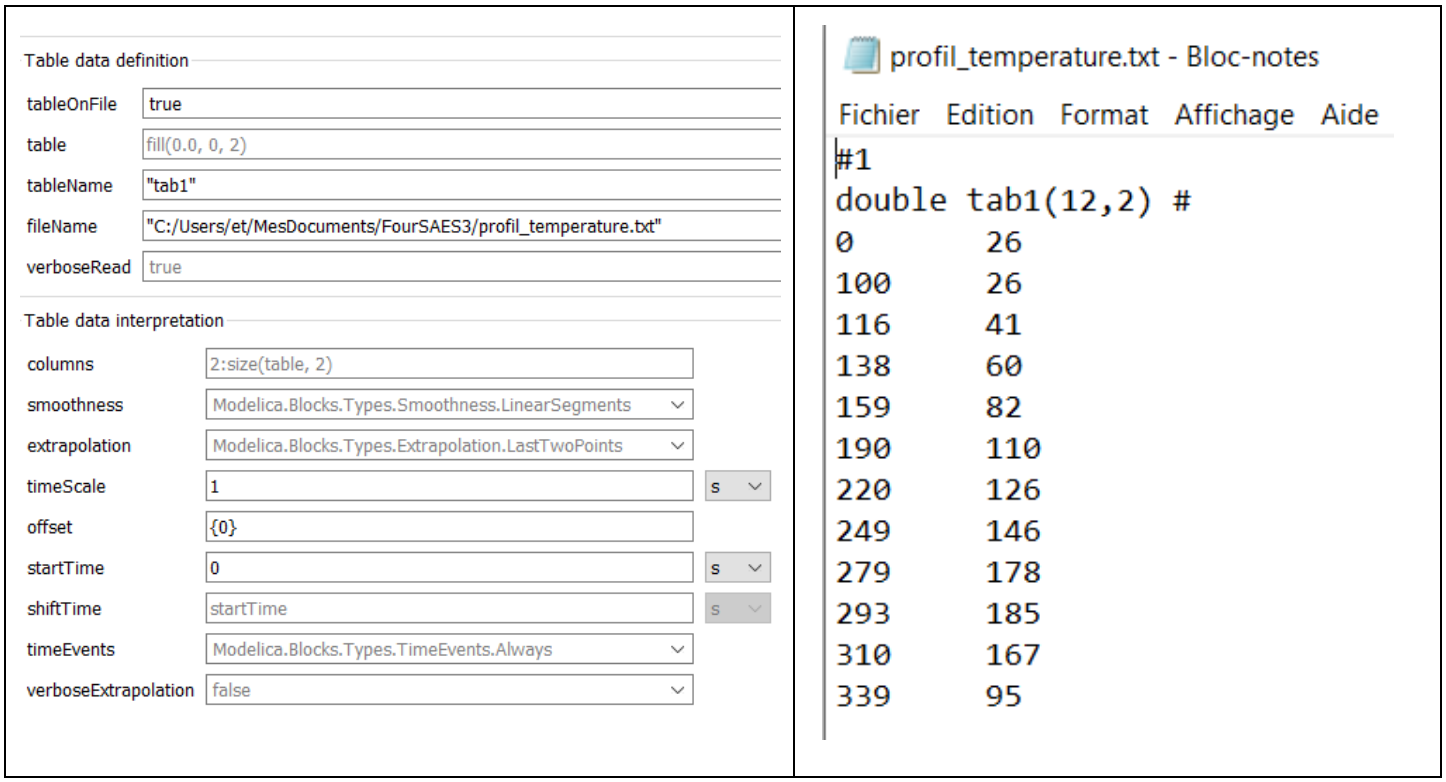


Dans le bloc « fixedDelay » on entre la valeur de « d » et dans le bloc « firstOrder » on entre les valeurs de « K » et « tau ».

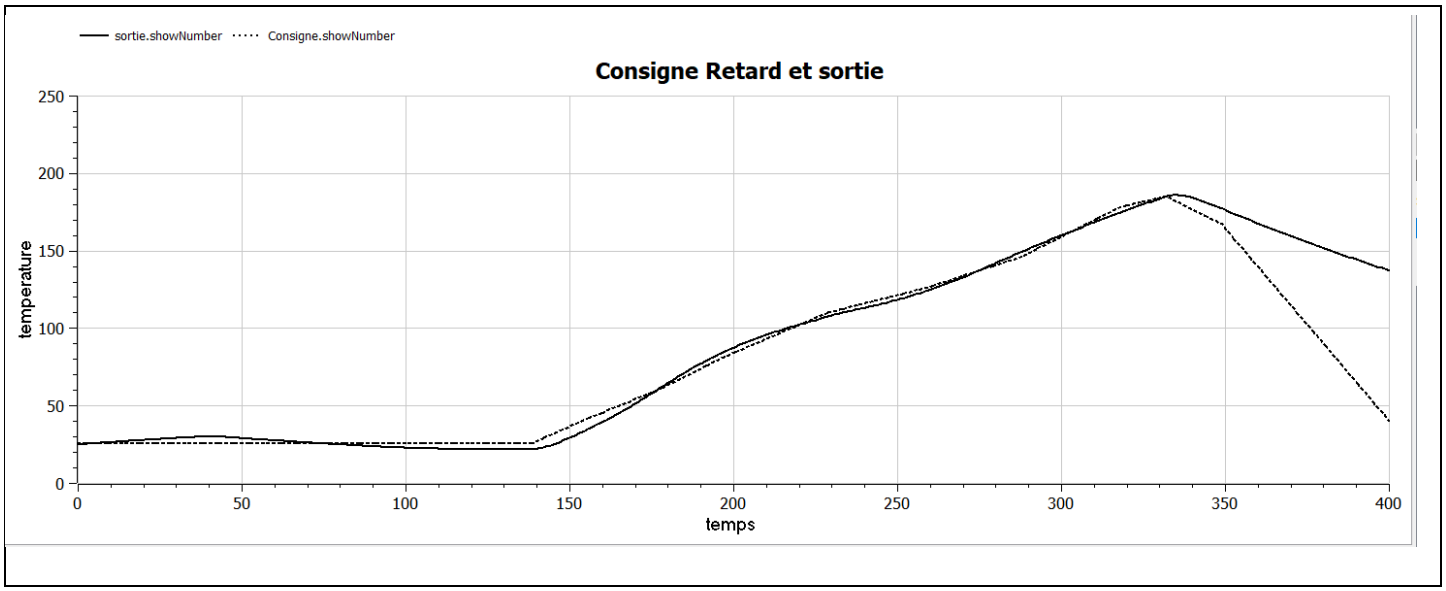
Le bloc « limiter » permet de prendre en compte la saturation de la commande soit uMin=0 et uMax=100.

Dans le bloc « PID » on entre les valeurs trouvées pour le PID

Le bloc « combiTimeTable » permet de mettre le profil comme consigne.



Après ajustement des paramètres du correcteur (il faut augmenter un peu le gain et Td) on arrive à avoir des performances acceptables


sortie.showNumber
Consigne.showNumber
Consigne Retard et sortie
temperature
temps
0
50
100
150
200
250
300
350
400