# Mise au point d'un algorithme auto adaptatif

**Janvier 2003**

ARNATRONIC

**Rémy LUCAS**

Promotion 2003

SOMMAIRE


I. Recherche de documentation - Synthèse Page 4

Introduction Page 5
Principe de la commande adaptative Page 6
Commande adaptative directe Page 7
Commande adaptative indirecte Page 8
Les réseaux de neurones Page 11
Conclusion Page 17

II. Identification et régulation d'un processus de 2ième ordre Page 19

Introduction Page 20
Principe Page 21
Implantation sous MATLAB Page 23
Essais et conclusions Page 24

III. Etude en continu Page 31

Introduction Page 32
Mesure de la réponse impulsionnelle Page 33
Identification Page 35
Synthèse du régulateur Page 37
Implantation sous MATLAB Page 39
Tests Page 57
Conclusion Page 60

IV. Etude en discret Page 62

Introduction Page 63
Démarche pour la synthèse du régulateur Page 64
1er essai Page 66
2ième essai Page 72
3ième essai Page 89
4ième essai Page 94
Conclusion Page 96

V. Maquette "Régulation thermique" Page 97

Introduction Page 98
Principe de fonctionnement Page 99
Alimentation Page 101
Ventilation Page 102
Chauffage Page 103
Mesure de température Page 104
Réalisation Page 105

VI. Informations utiles à propos de MATLAB Page 106

VII. Remerciements Page 107

Dernière mise à jour : 4 février 2003

# Recherche de documentation - Synthèse

SOMMAIRE


| | |
|---|---|
| Introduction | Page 5 |
| Principe de la commande adaptative | Page 6 |
| Commande adaptative directe | Page 7 |
| Commande adaptative indirecte | Page 8 |
| Les réseaux de neurones | Page 11 |
| Conclusion | Page 17 |

# Introduction

Dans l'industrie d'aujourd'hui, quasiment tous les processus sont automatisés. La commande automatisée permet d'atteindre les objectifs de coût, de qualité et de sécurité.

Cependant, la mise en place d'une régulation nécessite toujours un travail relativement important, travail ne pouvant être réalisé que par du personnel hautement qualifié.

Aussi, l'évolution à long terme des systèmes due à leur usure et leur vieillissement font que les différents régulateurs doivent être périodiquement réglés. Certaines opérations de maintenance impliquent également de refaire les réglages.

Ainsi, la mise en place et le maintient opérationnel d'une régulation nécessitent du temps et du personnel spécialement formé. Pour les entreprises dont le faible nombre d'équipements ne justifie pas la création d'un service dédié, cela implique :

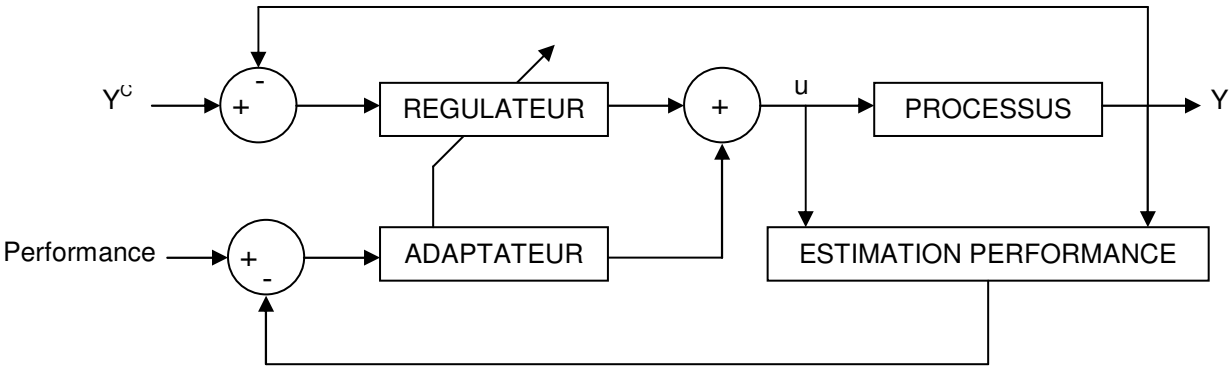
- de recourir à des intervenants extérieurs, mais cette solution est coûteuse et manque de souplesse par rapport à un service interne,
- d'abandonner ces tâches à d'autres services ; l'installation ainsi confiée à du personnel non qualifié fonctionne alors en permanence en dessous de ses performances optimales.

L'idée de concevoir un régulateur qui ajuste automatiquement ses paramètres est alors séduisante, car elle permet d'obtenir le maximum de performance sans nécessiter l'intervention régulière de techniciens spécialisés.

Il y a aussi le cas des processus dont les paramètres sont difficiles à estimer et/ou évoluent avec le temps. Dans ce cas la commande avec un régulateur traditionnel ne peut aboutir qu'à de mauvais résultats.

Ce projet va donc étudier la commande adaptative.

# Principe de la commande adaptative




Ce dispositif permet de commander des processus dont les paramètres sont inconnus et/ou évoluent dans le temps.

Nous voyons ici que la conception d'une commande adaptative suppose d'abord de définir un indice de performance désirée, qui devra qui plus est être mesurable, ou au minimum estimable.

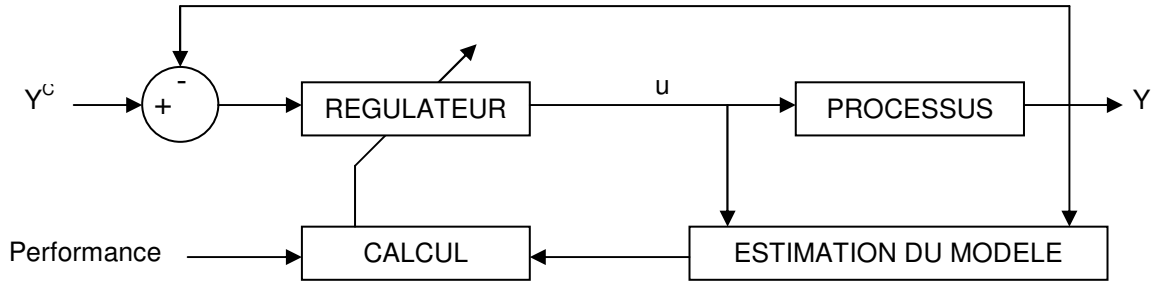
Il existe deux approches pour réaliser cette de commande adaptative :

- la **commande adaptative indirecte** : une procédure d'estimation permet d'identifier les paramètres.

- la **commande adaptative directe** : les paramètres sont ajustés directement et en temps réel.

La mise en œuvre de cette commande implique de nombreux calculs complexes. L'utilisation d'un calculateur numérique s'impose; donc nous ne traiterons par la suite que le cas discret.

# Commande adaptative indirecte




Le processus est décrit par $\underline{M}(q) = \underline{B}(q)\, q^{-d} / \underline{A}(q)$. Evidement, $\underline{A}$ et $\underline{B}$ sont inconnus.

Voici comment s'effectue la commande :

## 1. Estimation :

A partir des valeurs disponibles de u et y, $A_k$ et $B_k$, les estimations de $\underline{A}$ et $\underline{B}$, sont calculées.

## 2. Calcul :

La performance en poursuite est spécifiée par les polynômes $A_p$ et $B_p$ :
$A_p(q)\, y^d_{k+d} = B_p(q)\, y^c_k$
Ces deux polynômes permettent de calculer une sortie désirée $y^d$ réaliste tenant compte des constantes de temps du processus. Il est judicieux de choisir $A_p$ et $B_p$ pour éliminer les variations trop rapides de la consigne $y^c$, notamment celles due à du bruit. En effet, ces variations n'ont aucune chance d'être suivies, et entraînent une dépense énergétique inutile, voire une détérioration des actionneurs.

La performance en régulation est spécifiée par $A_R$, qui indique la dynamique d'annulation des perturbations :
$A_R(q)\, (y_{k+d} - y^d_{k+d}) = 0$

Le calcul se déroule comme suit :

On détermine $y^d_{k+d}$ à partir de la consigne et de $A_p$ et $B_p$,

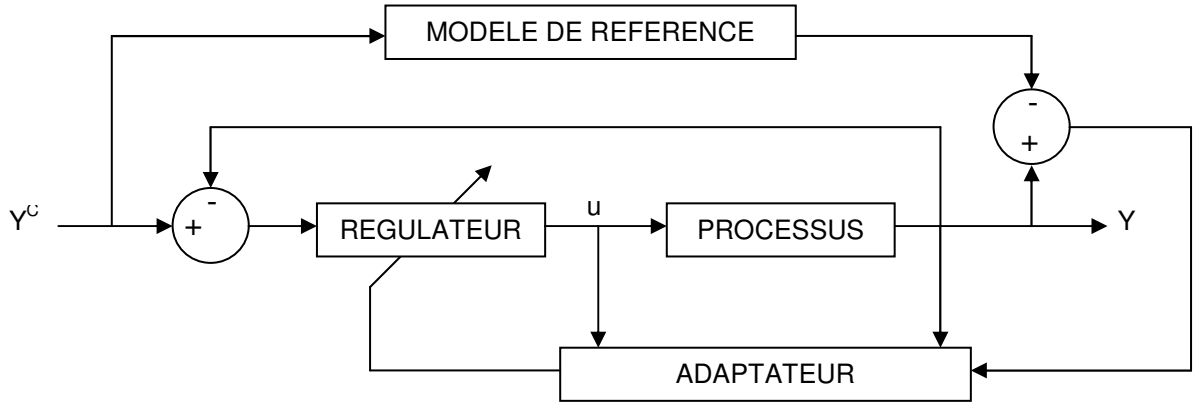
Ensuite on calcule $S_k$ et $R_k$ tel que :
$A_R(q) = S_k(q)\, A_k(q) + q^{-d}\, \mathbf{B_k(q)}\, R_k(q)$
On retrouve la méthode dite du modèle interne. Il s'agit en fait de placement de pôles **mais sans placement de zéros**. En effet, pour placer des zéros, ceux-ci doivent être stables ; or, le processus étant inconnu, nous ne sommes pas sûrs que cette condition soit toujours vérifiée.

## 3. Commande :

On applique alors la commande : $u_k = S_k(q)^{-1}\, [A_R(q)\, y^d_{k+d} - R_k(q)\, y_k]$

La quantité de calculs à effectuer va peut être nécessiter de retarder la commande d'une période d'échantillonnage (soit appliquer le $u_k$ ci-dessus à l'instant k+1). Il ne faudra alors pas oublier de tenir compte de ce retard supplémentaire dans le modèle.

# Commande adaptative directe




Dans ce modèle, nous allons chercher à optimiser la performance en régulation ; le critère à minimiser est donc :
$J(u_k) = [A_R(q)\ (y_{k+d} - y^d_{k+d})]^2$

La sortie désirée (aussi appelée trajectoire de référence) est $y^d_{k+d}$ toujours définie par :
$A_p(q)\ y^d_{k+d} = B_p(q)\ y^c_k$

Ici aussi nous partons de la méthode du modèle interne. Celle-ci précise qu'il existe des polynômes $\underline{R}$ et $\underline{S}$ tel que :
$A_R(q)\ y^d_{k+d} = \underline{B}(q)\ \underline{R}(q)\ y_k + \underline{S}(q)\ u_k$
Nous allons utiliser les estimations B, R et S sans chercher à identifier $\underline{A}$ et $\underline{B}$ proprement dit, car on ne dispose pas d'un modèle. Il s'agit en fait de la généralisation de la méthode à modèle de référence.

**Principe :**

Soit $\Phi_{0k}$ le vecteur des mesures disponibles à l'instant k et $\underline{\Theta}_0$ le vecteur des paramètres inconnus :
$\Phi_{0k} = [u_{k-1}, \ldots, u_{k-ns}, y_k, \ldots, y_{k-nr-nb}]^T$
$\underline{\Theta}_0 = [\underline{s}_1, \ldots, s_{ns}, \underline{b}^r_0, \ldots, \underline{b}^r_{nr+nb}]^T$

La commande $u_k$ optimale devant annuler $J(u_k)$ est définie par :
$\underline{s}_0\ u_k = -\ \Phi_{0k}{}^T\ \underline{\Theta}_0 + A_R(q)\ y^d_{k+d}$

$\underline{\Theta}_0$ et $\underline{s}_0$ étant inconnus, on utilise leurs estimations à l'instant k : $\Theta_{0k}$ et $s_{0k}$, soit :
$s_{0k}\ u_k = -\ \Phi_{0k}{}^T\ \Theta_{0k} + A_R(q)\ y_{k+d}$

En notant $\Phi^T_k = [u_k, \Phi^T_{0k}]$ et $\Theta^T_k = [s_0, \Theta^T_{0k}]$ on aboutit à : $0 = \Phi^T_k\ \Theta_k - A_R(q)\ y_{k+d}$

Soit $v_{0,k+d} = A_R(q)\ (y^d_{k+d} - y_{k+d})$, on a :
$v_{0,k+d} = -\ \Phi^T_k\ (\Theta_k - \underline{\Theta})$

**Procédure d'adaptation :**

Etant donné que $v_{0,k} = A_R(q)\,(y^d_k - y_k)$, $v_{0,k}$ est mesurable à l'instant k, nous pouvons calculer $\Theta_k$ :

$$v_k = [v_{0,k} + A_R(q)\, y^d_k - \Phi^T_{k-d}\, \Theta_{k-1}] \,/\, [1 + \Phi^T_{k-d}\, P_k\, \Phi_{k-d}]$$

$$P_k = (1/\lambda_{1k})\, [P_{k-1} - (P_{k-1}\, \Phi_{k-d}\, \Phi^T_{k-d}\, P_{k-1}) \,/\, (\lambda_{1k}/\lambda_{2k} + \Phi^T_{k-d}\, P_{k-1}\, \Phi_{k-d})]$$

$$\Theta_k = \Theta_{k-1} + P_k\, \Phi_{k-d}\, v_k$$

Les variables $\lambda_{1k}$ et $\lambda_{2k}$ sont des réels compris entre 0 et 1.
La matrice $P_k$ est la matrice de gain, définie positive et bornée. Son calcul en apparence complexe évite de devoir calculer $P_k^{-1}$.

**Mise en application :**

On peut décomposer la commande optimale sous la forme R, S, T, avec :

$$R(q) = \Sigma^{nr}_{i=0}\,(r_i\, q^{-i});\;\; S(q) = \Sigma^{ns}_{i=0}\,(s_i\, q^{-i});\;\; T(q) = \Sigma^{nt}_{i=0}\,(t_i\, q^{-i})$$

On a alors :

$$s_0\, u_k = - \Sigma^{ns}_{i=1}\,(s_i\, u_{k-i}) - \Sigma^{nr}_{i=0}\,(r_i\, y_{k-i}) - \Sigma^{nt}_{i=0}\,(t_i\, y^d_{k+d-i})$$

En notant :

$$\Phi_{0k}{}^T = [u_{k-1}, \ldots, u_{k-ns}, -y_k, \ldots, -y_{k-nr}, y^d_k, \ldots, y^d_{k-nt}]$$
$$\Theta_0{}^T = [s_1, \ldots, s_{ns}, r_0, \ldots, r_{nr}, t_0, \ldots, t_{nt}]^T$$

$$\Phi^T_k = [u_k, \Phi^T_{0k}]$$
$$\Theta^T = [s_0, \Theta^T_0]$$

On a : $0 = - \Phi^T_k\, \Theta + T(q)\, y^d_{k+d}$

Voici la séquence d'opérations à effectuer, les coefficients $\lambda_{1k}$ et $\lambda_{2k}$ étant fixés (voir paragraphe suivant) :

1. Mesure de $y_k$
2. Calcul de $v_{0,k}$
3. Calcul de $v_k$
4. Actualisation de $\Theta_k$
5. Commande : $u_k = (1/s_0)\, [\Phi^T_{0k}\, \Theta_{0k} + T(q)\, y^d_{k+d}]$
6. Calcul de $P_{k+1}$
7. Calcul de $y^d_{k+d+1}$ et de $T(q)\, y^d_{k+d+1}$

Là aussi, la quantité de calculs à effectuer va peut être nécessiter de retarder la commande d'une période d'échantillonnage (soit appliquer le $u_k$ ci-dessus à l'instant k+1). Il ne faudra alors pas oublier de tenir compte de ce retard supplémentaire dans le modèle.

**Choix des coefficients $\lambda_{1k}$ et $\lambda_{2k}$ :**

**$\lambda_{1k} = \lambda_{2k} = 1$ :**

Cela abouti à : $P_k^{-1} = P_0^{-1} + \Sigma^k_{i=0} (\Phi_i \Phi^T_i)$, $P_k$ décroît alors vers 0.

Or, précédemment, nous avons établi : $\Theta_k = \Theta_{k-1} + P_k \Phi_{k-d} v_k$

Cette estimation va donc tendre vers une constante. Il s'agit en fait d'une **aide au réglage**, puisque les variations des paramètres du processus ne sont pas suivies.

**$\lambda_{1k} = \lambda < 1, \lambda_{2k} = 1$ :**

Dans ce cas, $P_k^{-1} = \lambda^k P_0^{-1} + \Sigma^k_{i=0} (\lambda^i \Phi_{k-d-i} \Phi^T_{k-d-i})$

Il s'agit en fait d'un algorithme à facteur d'oubli exponentiel dont la fenêtre d'exploration est $H = \Sigma^\infty_{i=0} \lambda^i$

La valeur entière de H corresponds au nombre d'échantillons qui permettent d'obtenir une estimation correcte de $P_k$. Ainsi cet algorithme permet de suivre l'évolution des paramètres en conservant les H derniers échantillons dans une ligne à retard.

Mais lorsque la régulation est quasi parfaite (régime établit), $P_k$ est instable car dans ce cas : $P_{k+1} = P_k / \lambda$

Une réponse à ce problème consiste à définir une bande de bruit admissible dans laquelle l'évolution de $P_k$ est bloquée ($\lambda_{1k} = 1$ et $\lambda_{2k} = 0$). Aussi, pour éviter la décroissance de la trace de $P_k$, on peut la multiplier et ajouter des quantités positives aux termes diagonaux pour éviter qu'ils ne s'annulent.

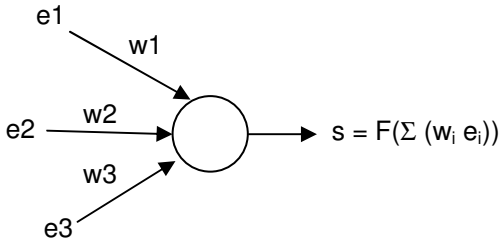
Voici un algorithme qui peut être utilisé :

Si trace($P_k^*$) > $t_{r0}$ alors
    $\lambda_{1k} = 1$
    $\lambda_{2k} = 0$
    $P_k = P_{k-1}$
Sinon
    $\lambda_{1k} = 1$
    $\lambda_{2k} = \text{trace}(P_k^*/t_{r0})$
    Si det($P_k$) ≤ d0 alors
        $P_k = P_k + \sigma \text{ trace}(P_k^*) I$
    Sinon
        $P_k = (1/\lambda_{1k}) [P_{k-1} - (P_{k-1} \Phi_{k-d} \Phi^T_{k-d} P_{k-1}) / (\lambda_{1k}/\lambda_{2k} + \Phi^T_{k-d} P_{k-1} \Phi_{k-d})]$
    Fin Si
Fin Si

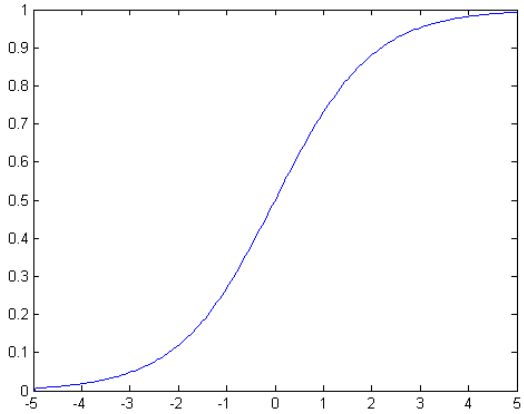
# Les réseaux de neurones :

## Principe de fonctionnement :

Voici la représentation d'un neurone :




F(x) = 1/(1 + exp(-d*x))



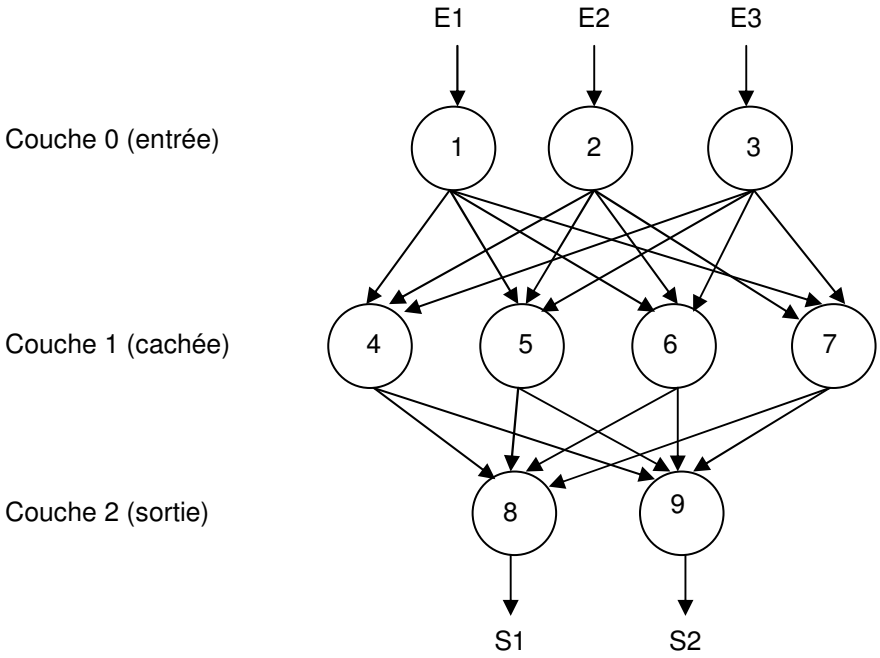

Un neurone est un dispositif comportant plusieurs entrées $e_i$ pondérées par des coefficients $w_i$ et une sortie s. Pour éviter des phénomènes de débordement, les valeurs des entrées sont toujours comprises entre -1 et +1. La fonction F(x), qui est une sigmoïde, est appelée fonction seuil et garantit que la sortie reste dans l'intervalle [-1,+1]. Enfin, les poids $w_i$ sont des réels quelconques, pouvant être nuls, négatifs ou positifs.

Cela implique que pour utiliser les réseaux de neurones il faut travailler avec des variables réduites.
Pour les variables continues et bornées, une simple règle de trois suffira.
Pour les variables bornées ayant une grande dynamique, on pourra travailler avec leur racine cubique.
Pour les variables non bornées, on utilisera une sigmoïde.
Pour les variables correspondant à un choix multiple, on codera le choix en binaire en utilisant autant d'entrées que nécessaire.

Ce fonctionnement corresponds à la modélisation des neurones biologiques. C'est pour cette raison que les réseaux de neurones sont qualifiés d'intelligence artificielle.
Remarque : l'asynchronisme (chaque neurone calcule et met à jour sa sortie à des instants indépendants des autres neurones) constaté dans le fonctionnement réel des neurones n'est pas modélisé.

En effet, un neurone seul ne peut pas faire grand chose mais lorsqu'on en associe plusieurs, comme dans un cerveau, le réseau obtenu permet d'identifier des systèmes dans une phase d'apprentissage et ensuite d'en simuler le comportement.

La topologie de réseau la plus souvent utilisée est le "perceptron multicouches" :




Le réseau est constitué de plusieurs couches : la couche d'entrée (les neurones 1, 2 et 3), la couche de sortie (les neurones 8 et 9) et une couche cachée (les neurones 4, 5, 6 et 7). Il peut y avoir plusieurs couches cachées.

Le réseau est construit de telle sorte que chaque neurone d'une d'une couche soit relié avec ses entrées à toutes les sorties des neurones de la couche précédente.

**Phase de fonctionnement :**

Le cycle de fonctionnement normal du réseau (après apprentissage) est le suivant :

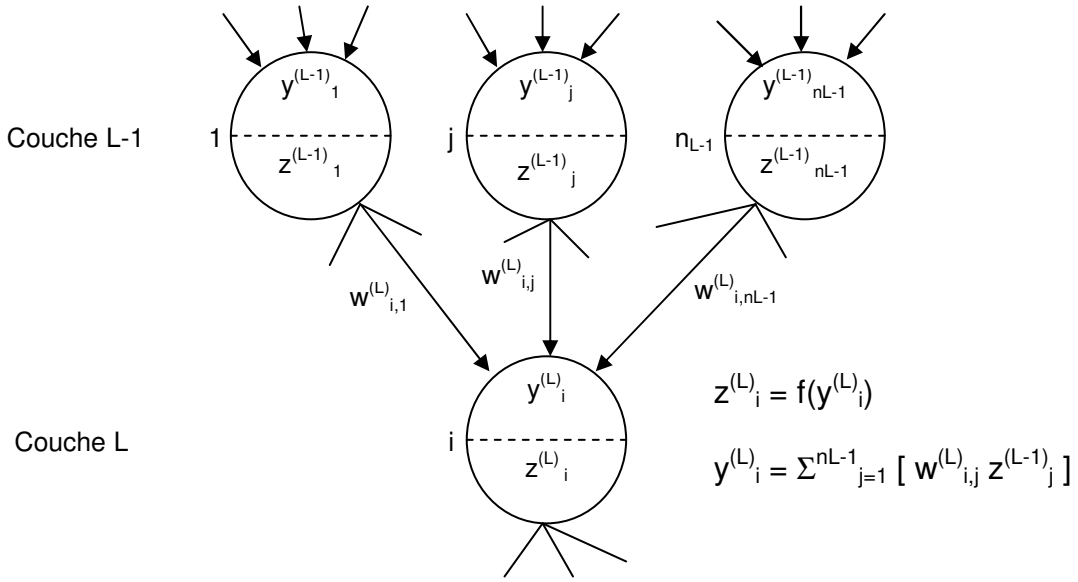
- Capture simultanée des entrées et réduction
- Les neurones de la 1ière couche calculent leur sortie
- Ensuite, les neurones de la couche suivante font de même
- ...
- Et ainsi de suite jusqu'à la dernière couche
- Enfin, il faut «dé-réduire» les sorties : on fait subir aux sorties la fonction réciproque de celle ayant servit à les réduire pendant la phase d'apprentissage.

**Phase d'apprentissage :**

Notations :

L'apprentissage est la phase pendant laquelle le réseau identifie le système en ajustant ses poids. D'une façon imagée, on montre plusieurs exemples au réseau, en y appliquant les valeurs des entrées et en comparant la sortie donnée par le réseau et la sortie attendue. La différence entre la sortie donnée et la sortie attendue est rétro propagée dans le réseau; les poids sont modifiés proportionnellement à leur contribution à l'erreur globale.

Voici les notations que nous allons utiliser par la suite :




$\mathbf{n^L}$ est le nombre de neurones de la couche L

$\mathbf{y^{(L)}_i}$ est l'entrée totale (aussi appelée activité) du neurone i de la couche L

$\mathbf{z^{(L)}_i}$ est la sortie (aussi appelée état) du neurone i de la couche L

$\mathbf{w^{(L)}_{i,j}}$ est le poids de l'entrée du neurone i de la couche L reliée avec le neurone j de la couche L-1.

Si L=0, les états $\mathbf{z^{(0)}_i}$ correspondent aux **entrées** du réseau et si L=q, les états $\mathbf{z^{(q)}_i}$ correspondent aux **sorties** du réseau.

On note $\mathbf{t^{(q)}_j}$ la valeur désirée de la sortie j.

**f()** est la fonction sigmoïde définie précédemment : f(x) = 1/(1 + exp(-d*x)) ; on remarque que f'(x) = f(x) (1-f(x))

Algorithme :

On applique à l'entrée du réseau les données $\{y^{(0)}_1 \ldots y^{(0)}_{n0}\}$ d'un des exemples qui sert à l'apprentissage. On récupère les sorties $\{z^{(q)}_1 \ldots z^{(q)}_{n0}\}$. La sortie désirée de l'exemple est quant à elle $\{t^{(q)}_1 \ldots t^{(q)}_{n0}\}$. Comme le sont les entrées, les sorties désirées doivent subir une réduction. La réciproque de cette fonction de réduction sera utilisée pour «traduire» les sorties lors de l'utilisation du réseau.

Nous pouvons exprimer l'erreur comme suit :

$$E = 0.5 \, \Sigma^{nq}_{j=1} (z^{(q)}_j - t^{(q)}_j)^2$$

L'objectif est de changer chaque coefficient w proportionnellement à sa contribution à l'erreur. Mais comme il y beaucoup de coefficients w, il vaut mieux d'abord calculer la contribution des erreurs totales y et ensuite en déduire la contribution des w.

Il faut donc calculer $\partial E / \partial y^{(L)}_j$

Pour L=q, et compte tenu de la remarque sur f'(x), on a :

$$\partial E / \partial y^{(q)}_j = z^{(q)}_j (1 - z^{(q)}_j) ( z^{(q)}_j - t^{(q)}_j)$$

Pour L<q, on a :

$$\partial E / \partial y^{(L)}_j = z^{(L)}_j (1 - z^{(L)}_j) \, \partial E / \partial z^{(L)}_j$$

et

$$\partial E / \partial z^{(L)}_j = \Sigma^{nL+1}_{k=1} ( w^{(L+1)}_{k,j} \, \partial E / \partial y^{(L+1)}_k )$$

(Les termes $\partial E / \partial y^{(L+1)}_k$ sont connus puisque calculés précédemment)

On peut alors calculer les la contribution à l'erreur des coefficients $w^{(L)}_{j,i}$ :

$$\partial E / \partial w^{(L)}_{j,i} = z^{(L-1)}_i \, \partial E / \partial y^{(L)}_j$$

En appliquant la méthode du gradient il vient :

$$\Delta w^{(L)}_{j,i} = - \rho \, \partial E / \partial w^{(L)}_{j,i}$$

$\rho$ est le pas de la méthode. C'est un nombre positif qui doit être de faible valeur. Son choix est délicat : une valeur trop petite l'apprentissage est très long et nécessite donc de nombreux exemples ; par contre une valeur trop grande risque de ne pas faire converger les coefficients.

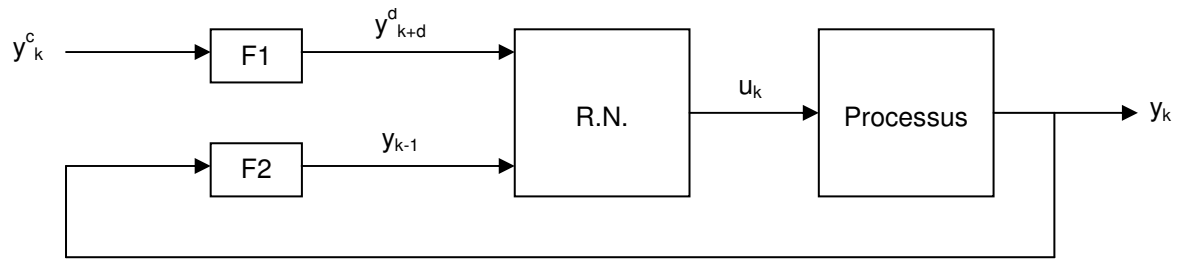
Nous pouvons alors réajuster le coefficient :

$$w^{(L)}_{j,i} = w^{(L)}_{j,i} + \Delta w^{(L)}_{j,i}$$

Tous ces calculs doivent êtres refaits pour chaque exemple servant à l'apprentissage.

**Application en automatisme :**

Voici comment je pense utiliser un réseau de neurone (R.N.) pour réguler un processus :



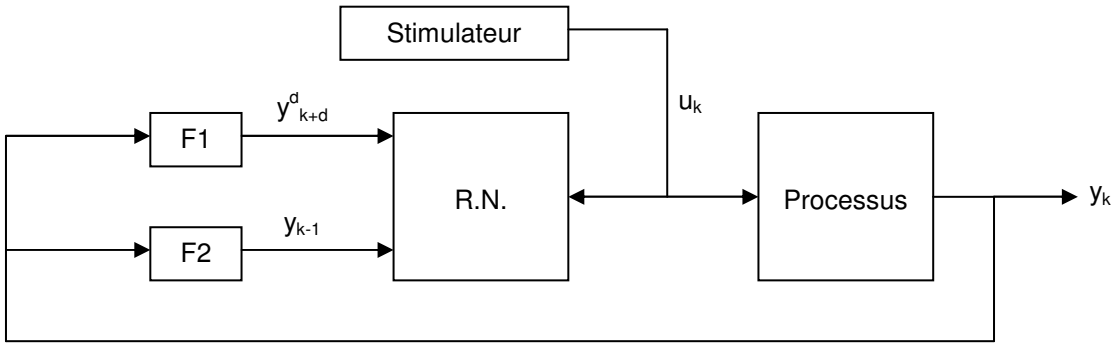

Le rôle de F1 est de filtrer $y^c_k$ pour y éliminer le bruit et de construire $y^d_{k+d}$.

Le rôle de F2 est de filtrer $y_k$ pour y éliminer le bruit et de mémoriser cette valeur pour donner en sortie $y_{k-1}$(échantillonneur bloqueur).

S'il y a beaucoup de bruit, le filtrage pourra se faire en sur-échantillonnant les différentes grandeurs et en calculant une moyenne glissante.

Nous supposons que le processus intègre un mécanisme de sécurité qui évite au système de dysfonctionner lorsque des valeurs brutales de $u_k$ lui sont appliquées. Sinon, il faudra prévoir un tel dispositif.

Voici comment se déroule l'apprentissage :

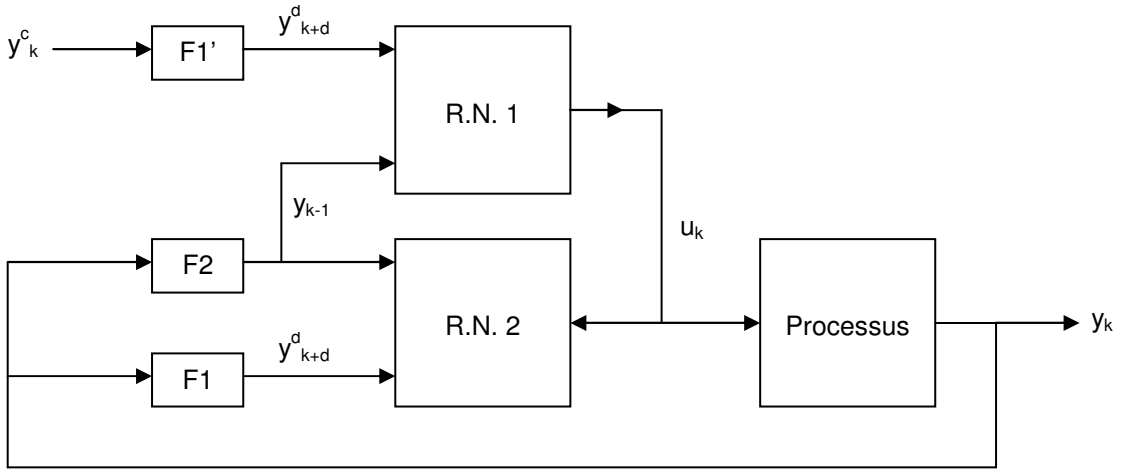



Le réseau de neurone fonctionne alors en rétro propagation de l'erreur.

Le stimulateur délivre une commande $u_k$ suffisamment riche (dynamique importante, spectre étendu) pour que le réseau de neurone puisse apprendre comment fonctionne le processus.

L'inconvénient est que le processus n'est plus régulé mais stimulé. L'idéal serai de pouvoir mener l'apprentissage tout en continuant à réguler le système.

Une idée qui vient à l'esprit est d'utiliser non pas un mais deux réseaux de neurones. Pendant que l'un régule le système, l'autre est en phase de ré apprentissage.




Le réseau 1 s'occupe de réguler le processus tandis que le 2 est en ré apprentissage. Les filtres F1 et F1' sont strictement identiques. A des intervalles réguliers, les deux réseaux sont permutés.

Quelques remarques :

- Les deux réseaux n'étant pas identiques au moment de la permutation, il convient de vérifier si le changement de réseau ne risque pas de produire un transitoire gênant.

- Il faut bloquer l'apprentissage dès que la commande $u_k$ n'a pas une dynamique et un contenu fréquenciel suffisants. En d'autre terme, l'apprentissage se fera avec des salves d'exemples capturées au moment où $u_k$ est assez riche.

- Les réseaux de neurones sont très performants et permettent de modéliser de nombreux problèmes complexes là où les méthodes classiques échouent. Mais ils ne sont que des boîtes noires, et ne sont absolument pas explicatifs.

# Conclusion :

Nous allons d'abord passer en revue les avantages et inconvénients des différentes méthodes évoquées dans ce rapport.

**Commande adaptative indirecte :**

- Ce type de commande est à la base de nombreux régulateurs auto réglables ; ces régulateurs font aujourd'hui l'unanimité dans l'industrie.
- Facilité de mise en œuvre

- Il faut connaître la forme du modèle pour en estimer les paramètres
- Le modèle doit être linéaire aux environ d'un point de fonctionnement
- La commande adaptative d'un processus linéaire et stationnaire est non linéaire ou non stationnaire.
- L'aide au réglage n'est pas la commande auto adaptative.

**Commande adaptative directe :**

- Il n'y a pas besoin de connaître la forme du modèle

- Le modèle doit être linéaire aux environ d'un point de fonctionnement
- Pour que l'algorithme du mécanisme d'adaptation aboutisse à des trajectoires de u et y stables, il faut que le comportement linéaire de l'asservissement soit stable
- Le paramètre $\sigma$ de la procédure évitant la dégénérescence de $P_k$ doit être déterminé à partir de plusieurs essais.
- Il faut déterminer le paramètre $\lambda$ de l'algorithme à facteur d'oubli.

**Réseaux de neurones :**

- Les réseaux de neurones sont capables d'apprendre le comportement de n'importe quel système, linéaire ou pas.

- Il faut travailler avec des variables réduites.
- Il faut définir le nombre de couches, le nombre de neurones par couche cachée, le paramètre d de la sigmoïde et le pas $\rho$ de la méthode d'apprentissage. Ces différents paramètres sont étroitement liés à la nature du système et au nombre de données pour l'apprentissage.
- Il faut mettre au point F1 et F2.
- La performance n'est pas constante ; elle peut fluctuer au cours du temps.
- L'apprentissage en continu non supervisé peut dégrader les réseaux de neurones.
- Il est impossible de déduire un modèle d'après les valeurs des différents coefficients du réseau après apprentissage

Dans tous les cas, il est nécessaire d'encadrer la commande adaptative par un dispositif de supervision, qui doit détecter les dérives et instabilités de la sortie. Ce dispositif devra alors agir sur les paramètres de la commande adaptative, quitte à accepter une performance moindre, ou dans le pire des cas passer la main à une commande traditionnelle peu performante mais robuste.

De plus, la mise en œuvre d'une commande adaptative ne dispense pas d'étudier au préalable le processus. L'idée de concevoir un régulateur universel pouvant réguler un processus quelconque sans réglages est donc une utopie.

Mais en réduisant le champ d'applications à une classe de problèmes de régulations définis, on peut tout à fait concevoir un régulateur auto adaptatif. Par exemple, on pourra proposer un régulateur polyvalent pour une catégorie bien précise d'équipement après avoir conclut que les processus en question se comportent comme des premiers ou deuxièmes ordres avec retard pur.

Pour les processus plus délicats, comme les systèmes non linaires ou dont les paramètres évoluent rapidement, seule une étude personnalisée du processus permettra de mettre en œuvre la commande adaptative.

La complexité des calculs a mettre en œuvre et le temps disponible sont à prendre en compte. Les processus trop rapides risquent de nécessiter une puissance de calcul énorme et donc un calculateur d'autant plus coûteux.

C'est ainsi que les asservissements dans les circuits électroniques rapides sont encore réalisés aujourd'hui en analogique. C'est le cas notamment des alimentations à découpage ou encore des circuits HF.

# Identification et régulation d'un processus de type deuxième ordre

SOMMAIRE


| | |
|---|---|
| Introduction | Page 20 |
| Principe | Page 21 |
| Implantation sous MATLAB | Page 23 |
| Essais et conclusions | Page 24 |

# Introduction

Considérons le problème suivant :

Nous avons à réguler un processus inconnu, dont nous savons juste qu'il se comporte comme un deuxième ordre. L'objectif est de réguler ce processus de manière a obtenir certaines performances.

Pour ce faire, il faudra d'abord identifier le système et ensuite synthétiser le correcteur qui permettra d'atteindre les performances désirées.

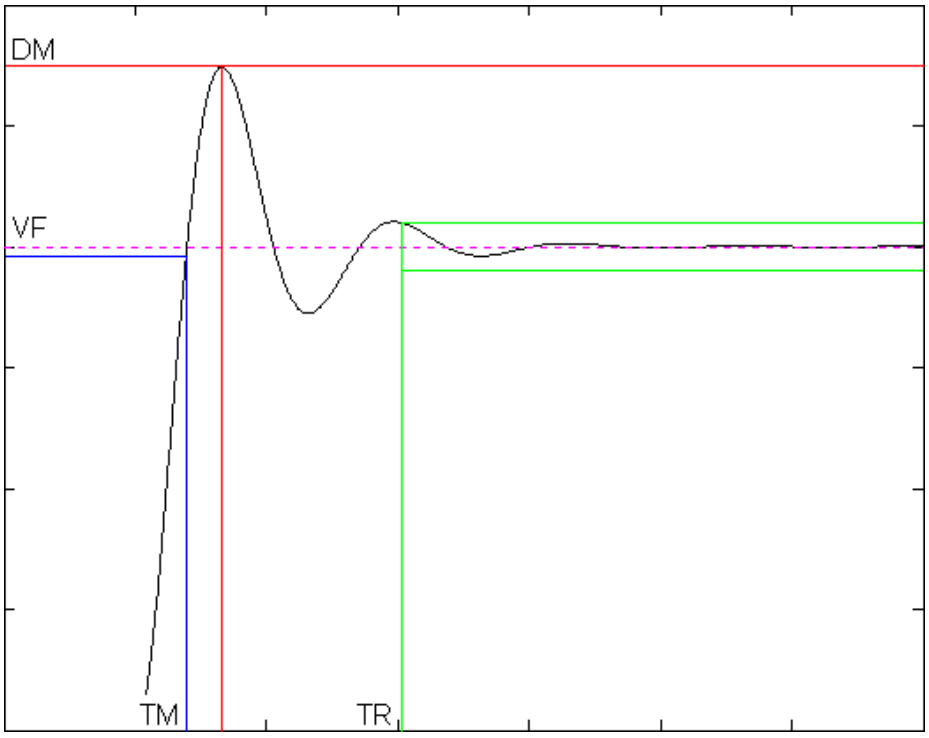
Dans la suite de ce document, nous concevrons une méthode, qui sera ensuite mise en œuvre avec MATLAB.

# Principe

## Identification du système :

Pour identifier le système, nous allons le soumettre à un stimuli de type échelon puis nous mesurerons sa réponse indicielle.
Comme la mesure comporte du bruit, il faudra filtrer le signal obtenu. Ensuite nous rechercherons dans ce signal certains points caractéristiques :

- la valeur finale VF,
- le dépassement maximal DM,
- le temps de monté TM,
- le temps de réponse TR.




Ici, TM est définit comme le temps que mets la sortie pour passer de 0% à 98% de VF. TR est le temps que mets la sortie pour se stabiliser à VF +/-5%.

Bien sûr, TM et TR sont définit par rapport au temps écoulé depuis le départ de l'échelon.

Le processus, de type deuxième ordre, a une fonction de transfert du type :
$F(p) = K * W0^2 / (p^2 + 2 * E * W0 * p + W0^2)$

On peut retrouver ces paramètres à partir des grandeurs caractéristiques de la réponse indicielle :

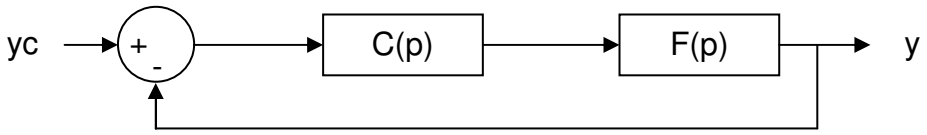
K = VF (si l'échelon est d'amplitude 1)
$E = \sqrt{ [ (\text{Ln } DM)^2 / ((\text{Ln } DM)^2 + \Pi^2)]}$

W0 = Ln(1/n) / (TR * E) avec n = 0.05 si TR est définit à VF +/- 5%
W0 = (Π - ArcCos(E)) / (TM * √(1 - E&))

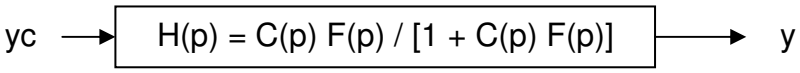
**Synthèse du correcteur :**

L'objectif est de réguler le processus pour atteindre une certaine performance en boucle fermée :

Le système en boucle fermée ci-dessous :




Est équivalent à :




On exprime les performances sous la forme d'une fonction de transfert H(p) désirée : Hd(p).

Le correcteur est donc C(p) = Hd(p) / [F(p) (1 - Hd(p))]

Par exemple, on peut désirer que Hd(p) soit un deuxième ordre dont on spécifie un dépassement plus faible et un temps de montée plus court que le processus en boucle ouverte F(p).

La performance se traduit dans ce cas par des valeurs de TM et DM désirées. A partir de ces valeurs on calcule Hd(p), puis C(p).

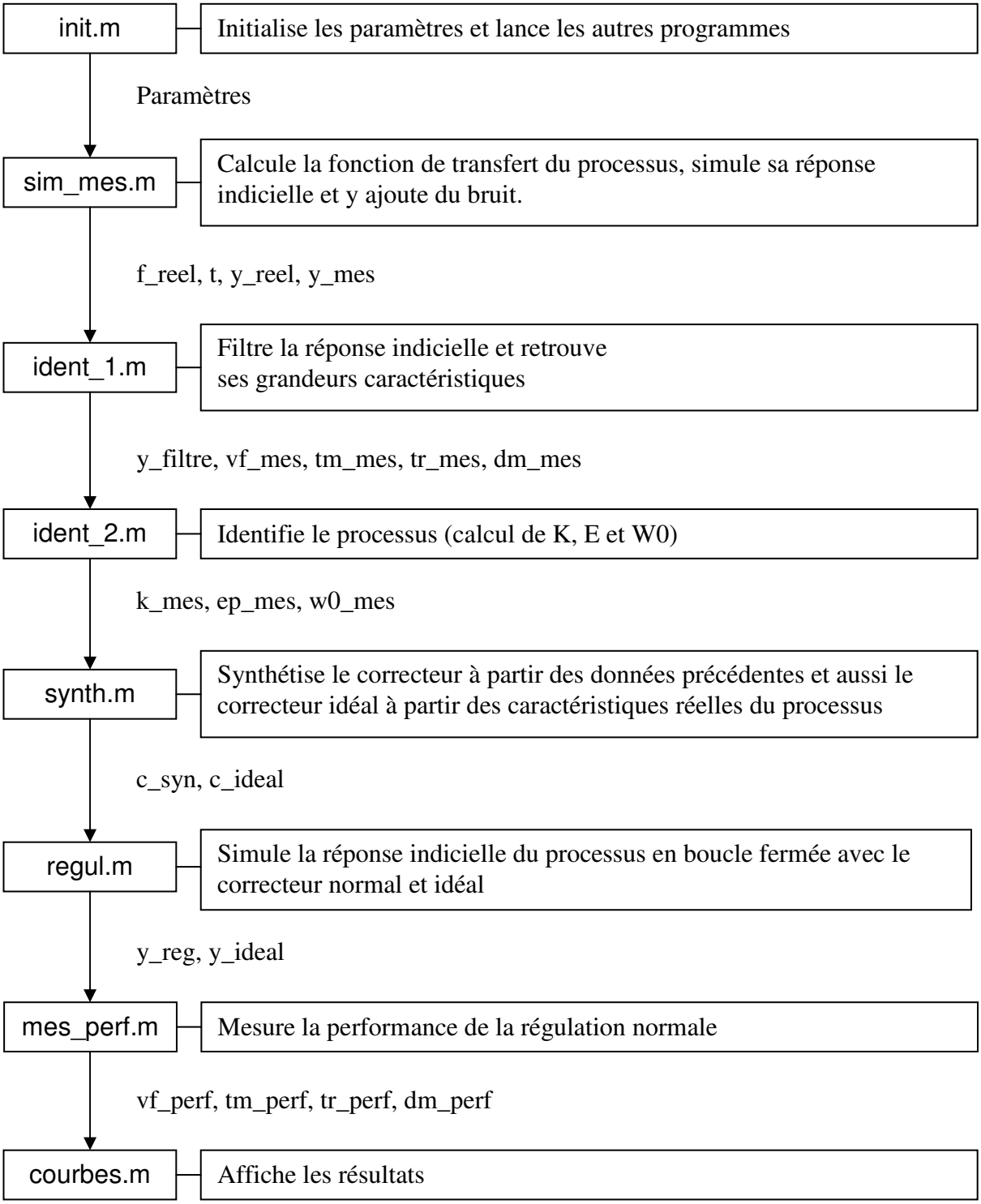
Bien évidemment, le gain statique K de Hd(p) vaut 1 (erreur statique de position nulle).

Cette manière de calculer un correcteur est connu sous le nom de "méthode du modèle".

# Implantation sous MATLAB

[Dossier MATLAB1]

Voici le rôle des scripts MATLAB contenus dans le dossier MATLAB1 :




Pour utiliser les programmes, éditez le fichier init.m. Vous pouvez modifier les valeurs des paramètres à votre guise. Ensuite, enregistrez le fichier et tapez "init" dans la fenêtre de commande. Chaque programme contient de nombreux commentaires.

# Essais et conclusions

Impact du filtrage sur l'allure de la courbe

Le filtrage mis en place dans le fichier ident_1.m est une moyenne glissante. Nous allons faire quelques essais sans bruit pour étudier l'impact du filtrage.

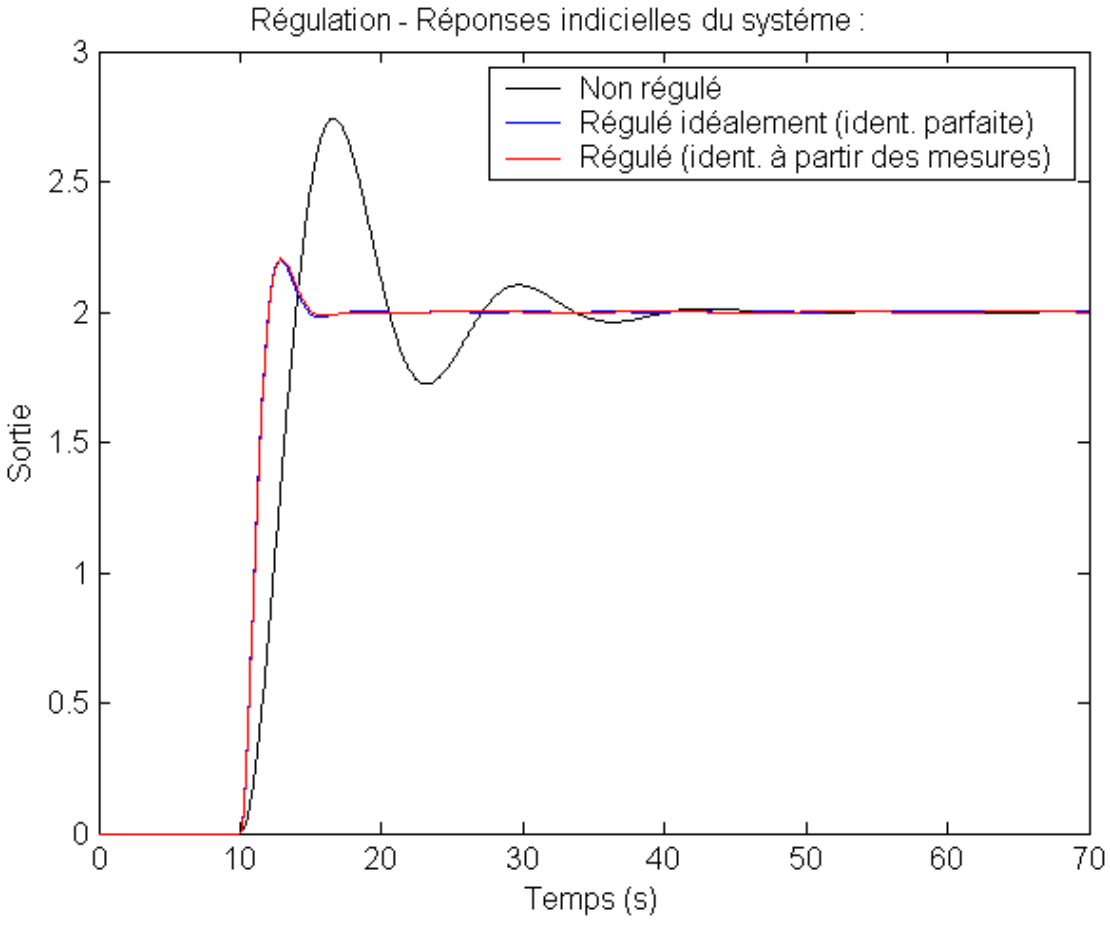
En précisant lfg = 1, nous obtenons un filtrage très limité, sur une fenêtre glissante de 3 points. Voici les résultats :

La réponse indicielle théorique (y_reel) et filtrée (y_filtree) se superposent quasiment (sans utiliser le zoom, la différence entre les deux courbes est invisible).

L'estimation du système est excellente :

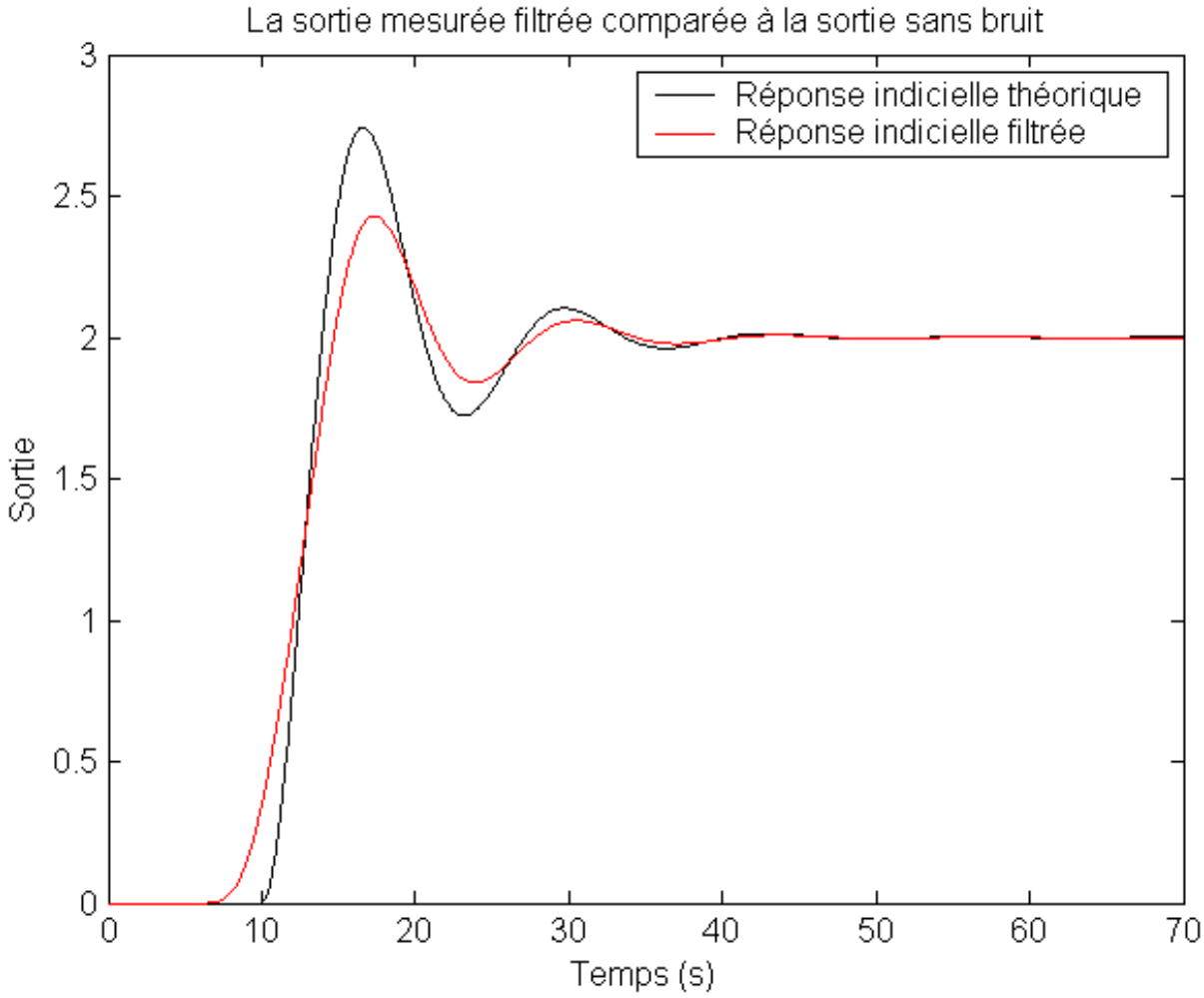
| Système réel : | Système estimé : |
|---|---|
| K = 1 | K = 0.99994 |
| W0 = 0.5 (rad/s) | W0 = 0.50475 (rad/s) |
| EP = 0.3 | EP = 0.29997 |

Cependant, une légère différence entre les deux réponses en boucle fermée (y_reg et y_ideal) est perceptible :



Faisons un nouvel essai en précisant lfg = 100 (comme nous le verrons par la suite une telle valeur de lfg rejette très bien le bruit) :

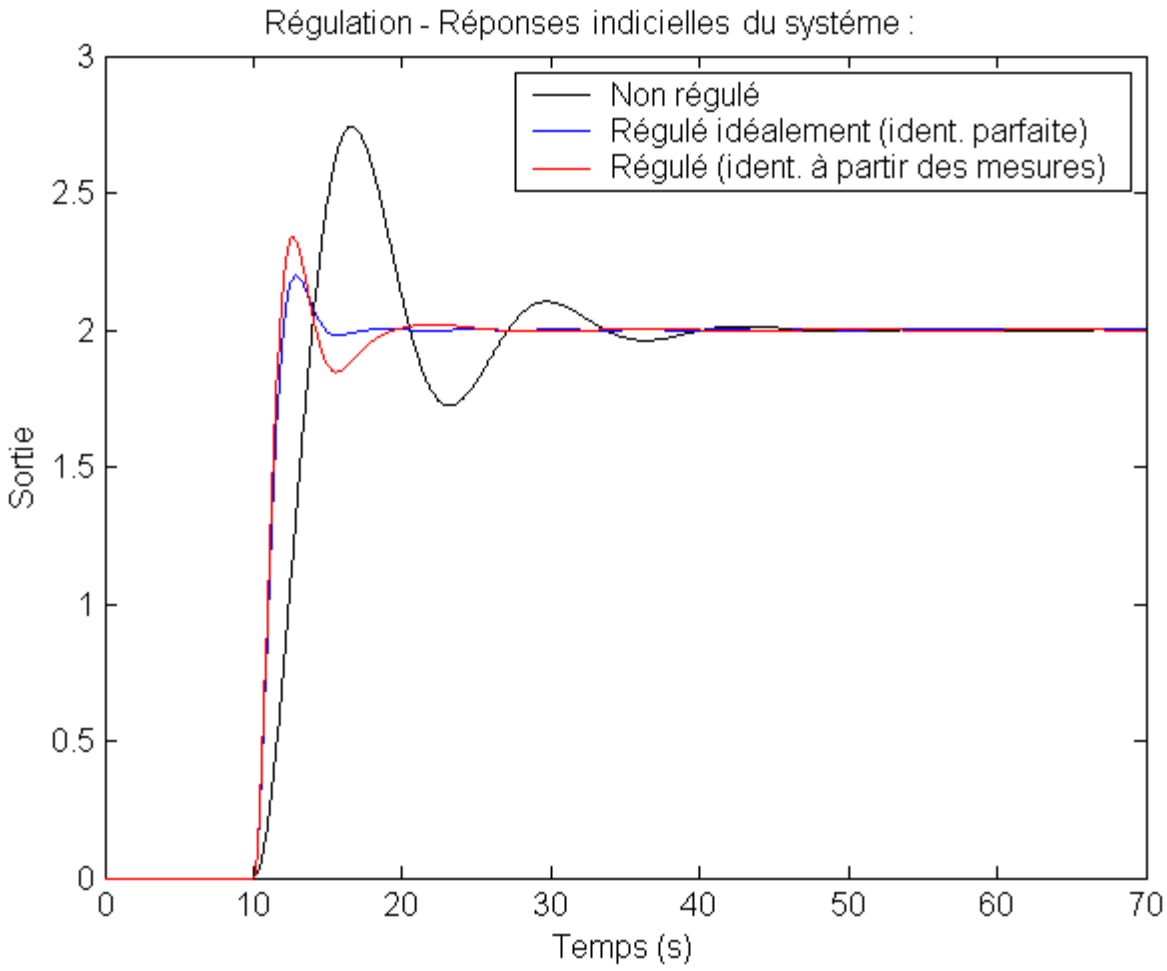
La différence entre la réponse indicielle théorique (y_reel) et filtrée (y_filtree) est assez importante :




C'est surtout le coefficient d'amortissement EP qui est affecté :

| | |
|---|---|
| Système réel : | Système estimé : |
| K = 1 | K = 0. 99989 |
| W0 = 0.5 (rad/s) | W0 = 0. 4906 (rad/s) |
| EP = 0.3 | EP = 0. 43908 |

La différence entre les deux réponses en boucle fermée (y_reg et y_ideal) est assez nette :




Y_ideal, en bleu sur le graphique, est la réponse du système en boucle fermée que l'on souhaite obtenir. Y_reg, en rouge, est la réponse effectivement obtenue.

Cette différence est uniquement due à un filtrage trop important. Par la suite il faudra donc trouver un compromis entre le niveau de bruit résiduel et la déformation due au filtrage.

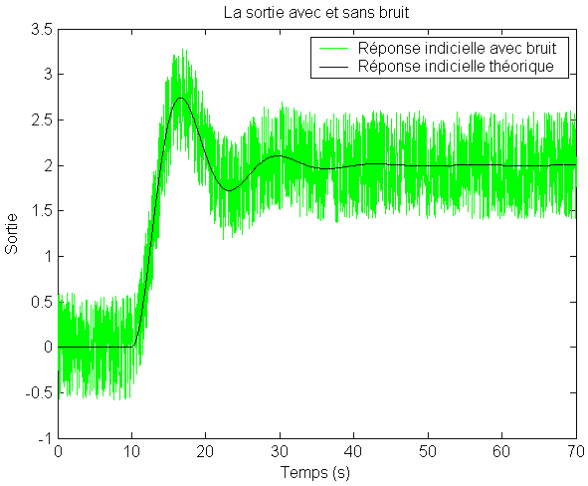
**Impact du bruit - Choix de l'importance du filtrage :**

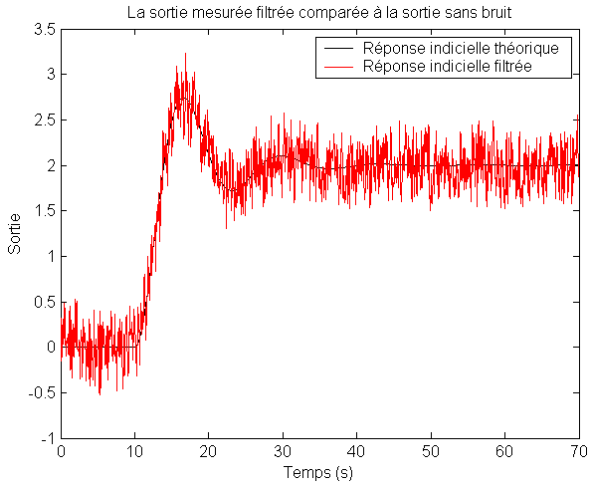
Nous allons cette fois ci introduire un bruit de 30% lors de l'identification du système.

Faisons un premier essai avec un filtrage minimal, lfg = 1.

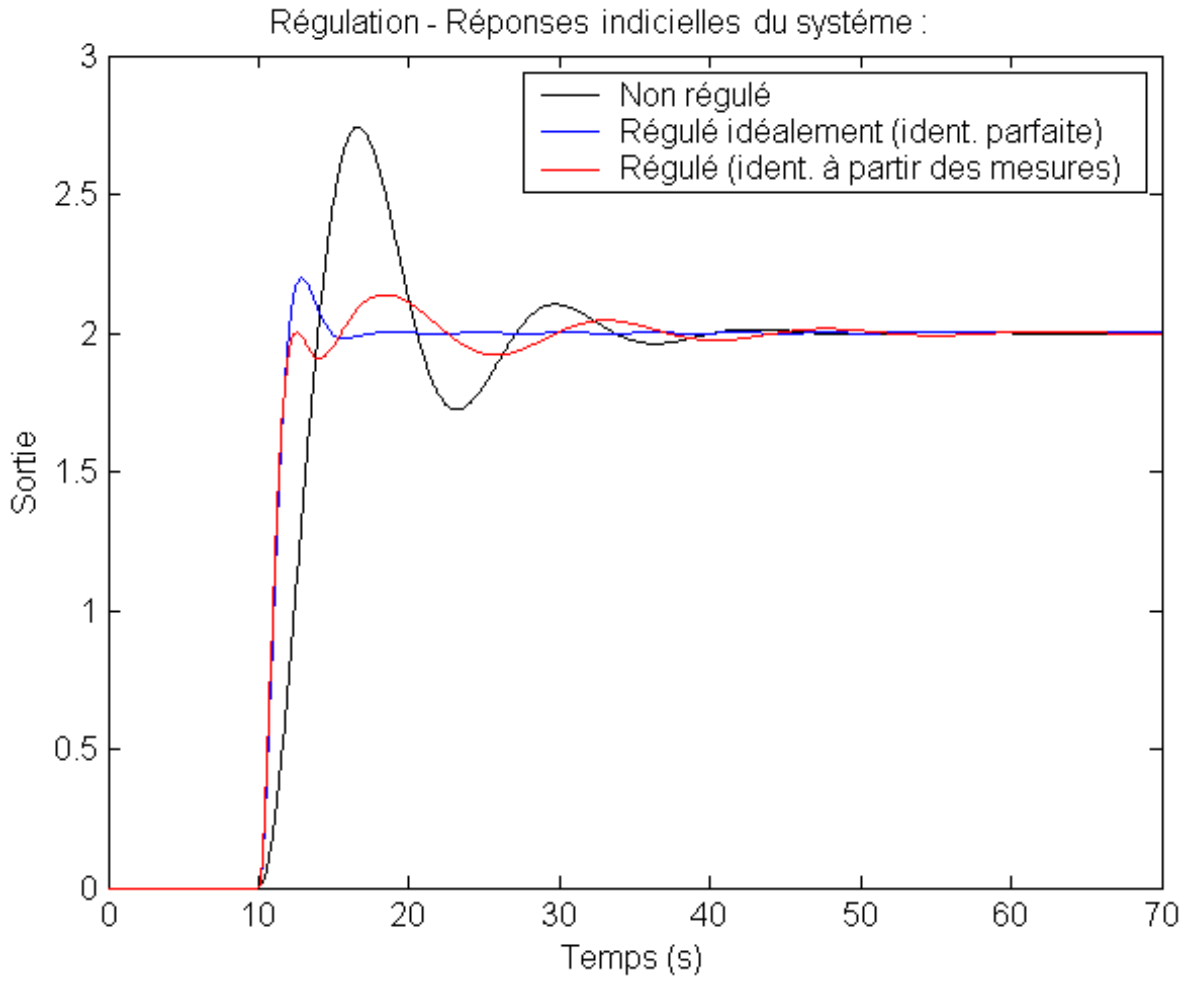
Comme on peut le voir sur les deux images ci-dessous, le bruit est faiblement rejeté :

La réponse non filtrée est à gauche (vert) et la réponse filtrée est à droite (rouge) :



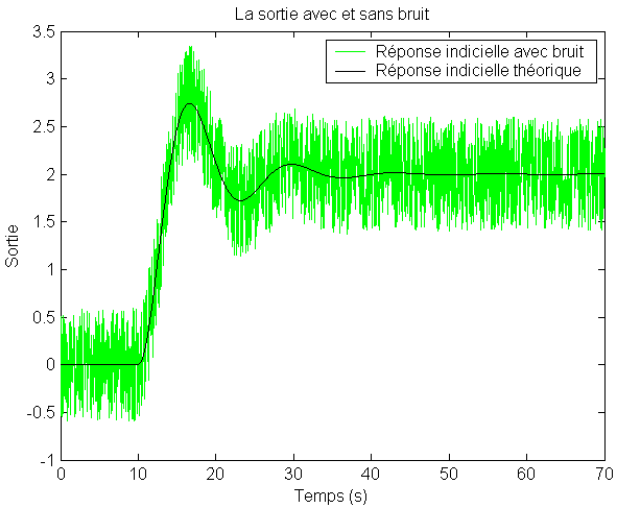



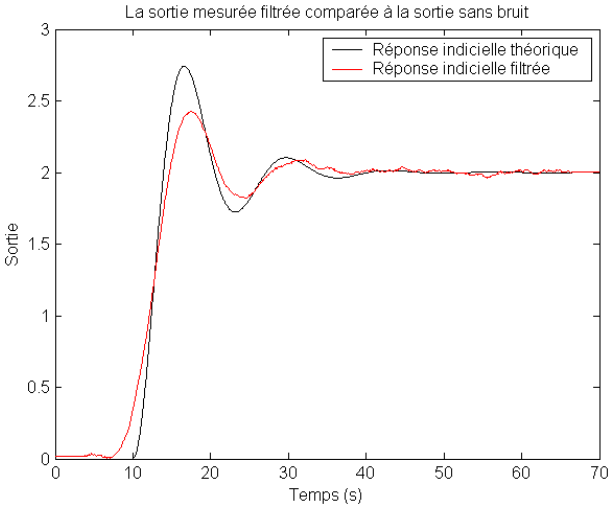

La régulation est donc mauvaise :



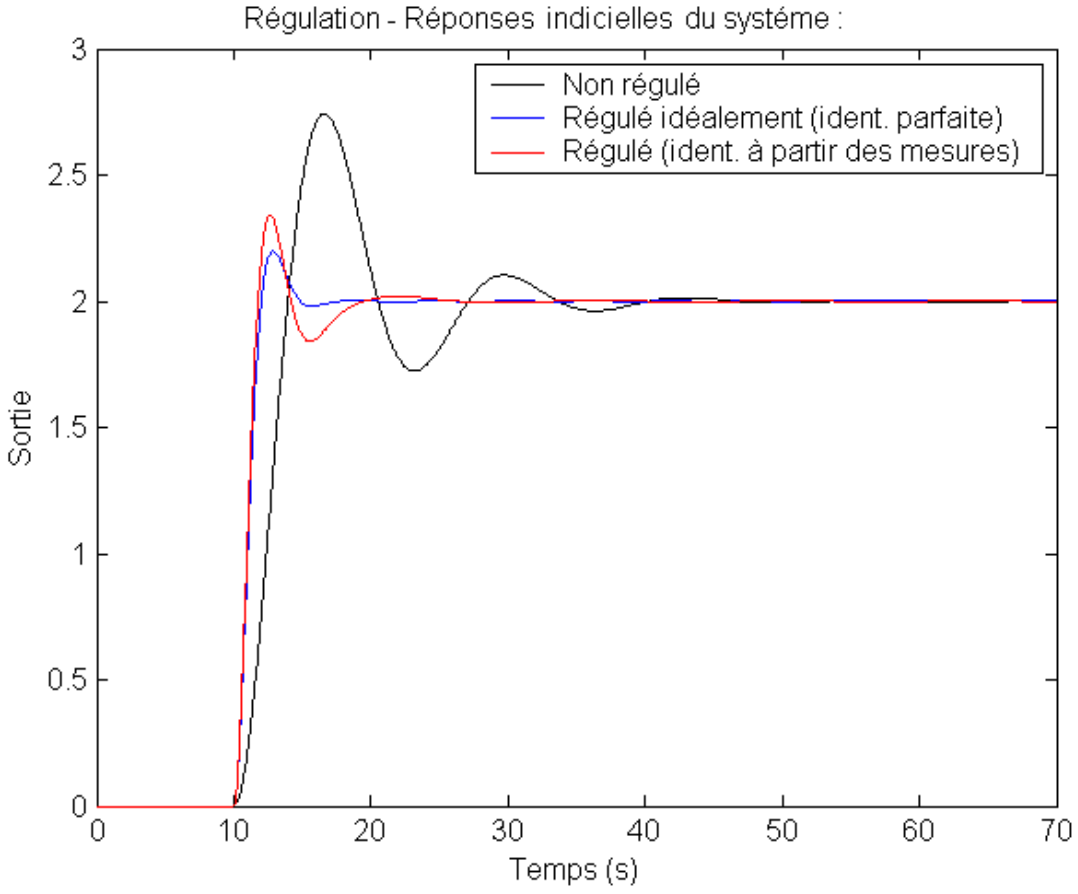
Essayons avec un filtrage plus puissant : lfg = 100.

Comme on peut le voir sur les deux images ci-dessous, le bruit est très bien rejeté; la réponse non filtrée est à gauche (vert) et la réponse filtrée est à droite (rouge) :



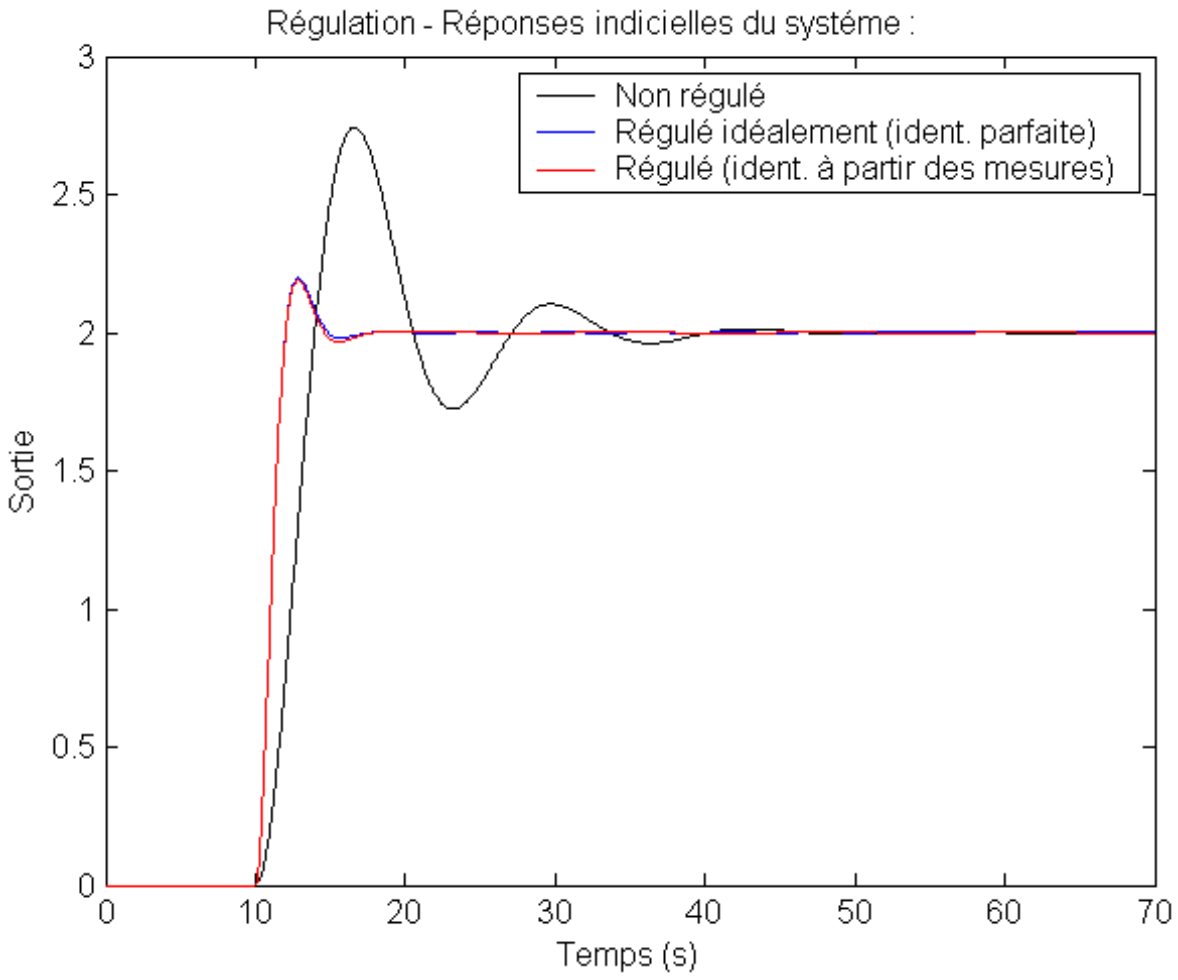

Mais comme nous l'avons vu plus haut, ce filtrage important déforme la courbe; la régulation ne corresponds pas non plus à ce que nous voulions :



Essayons maintenant avec un filtrage moins prononcé : lfg = 30 (fenêtre de 61 points)

Le résultat est ici excellent :

| Système réel : | Système estimé : |
|---|---|
| K = 1 | K = 1.0251 |
| W0 = 0.5 (rad/s) | W0 = 0. 49393 (rad/s) |
| EP = 0.3 | EP = 0. 30549 |




En refaisant une autre série d'essais avec un niveau de bruit de 100%, on trouve que c'est avec lfg = 40 que les meilleurs résultats sont obtenus.

Ces observations nous amènent à une première conclusion :

Le choix de l'intensité du filtrage est primordial pour la qualité de la régulation; lors de la mise en place d'un tel système, il faut donc avoir une idée de la qualité des mesures lors de l'identification du processus.

**Influence du coefficient d'amortissement sur la qualité de l'identification :**

Nous allons faire plusieurs essais en faisant varier ep_reel et en gardant les autres coefficients constants :

lfg = 30
Niveau de bruit 30%
k_reel = 1
w0_reel = 0.5

| ep_reel | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.75 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ep_mes | 0.114 | 0.198 | 0.301 | 0.386 | 0.492 | 0.631 | 0.661 | 0.676 | 0.676 | 0.725 | 0.694 |
| w0_mes | 0.514 | 0.505 | 0.496 | 0.482 | 0.506 | 0.499 | 0.459 | 0.477 | 0.439 | 0.359 | 0.298 |
| k_mes | 1.022 | 0.986 | 0.996 | 0.993 | 1.000 | 1.013 | 1.006 | 1.002 | 1.005 | 1.009 | 0.997 |

Nous constatons que dès que le coefficient d'amortissement dépasse 0.7, l'identification du système conduit systématiquement à des résultats faux : ep_mes reste autour de 0.7 et w0_mes diminue au fur et à mesure que ep_reel augmente.

Cela vient du fait que le dépassement maximal devient très difficile à retrouver lorsque le système n'est plus résonant.

Il faudrait donc mettre en place une autre procédure d'identification qui serai appelée lorsque la valeur du coefficient d'amortissement estimée dépasse 0.65.

**Autres problèmes :**

Tout d'abord, dans la pratique le système est complètement inconnu. Ici nous n'avons pas vraiment joué le jeu puisque le temps de simulation dépends de l'ordre de grandeur de w0 qui est inconnu. En d'autres termes, lors de la mise en place de cette méthode, il faudra faire une pré-identification du système pour déterminer l'ordre de grandeur de ses constantes de temps.

Ensuite, appliquer une entrée en échelon d'amplitude suffisante pour identifier convenablement le système n'est pas envisageable sur certains processus. Une telle commande risque en effet de poser des problèmes de sécurité et d'endommager le matériel.

Enfin, notre programme permet de calculer le correcteur de manière à obtenir les performances désirées. Mais des performances désirées trop sévères (Hd(p) proche de 1) conduisent à envoyer au processus des commandes trop brutales. Une telle commande risque d'endommager le système ou de ne pas être réalisée du fait de saturations.

Finalement, la difficulté est de trouver un algorithme capable de choisir les valeurs optimales des paramètres contenus dans le fichier init.m.

# Etude en continu

SOMMAIRE


| | |
|---|---|
| Introduction | Page 32 |
| Mesure de la réponse impulsionnelle | Page 33 |
| Identification | Page 35 |
| Synthèse du régulateur | Page 37 |
| Implantation sous MATLAB | Page 39 |
| Tests | Page 57 |
| Conclusion | Page 60 |

# Introduction

Considérons le problème suivant :

Nous avons un processus totalement inconnu, d'un ordre quelconque, avec ou sans retard. L'objectif est de réguler ce processus de manière a obtenir certaines performances (caractéristiques de la réponse indicielle).

Voici comment nous allons résoudre ce problème :

## 0. Hypothèses simplificatrices

Nous nous plaçons dans le cas de l'aide au réglage, c'est à dire avec une phase d'identification et une phase de régulation distinctes.

Nous ne considérerons que les ordres suivants : gain pur, $1^{er}$ ordre, $2^{ième}$ ordre.

## 1. Mesure de la réponse impulsionnelle

Nous netterrons en œuvre une méthode permettant de retrouver la réponse impulsionnelle du système (cette-ci n'étant pas directement mesurable). Nous verrons également que nous pouvons effectuer cette mesure pendant que le processus est régulé, ce qui nous permettra d'envisager en conclusion de ce rapport une commande auto adaptative.

## 2. Identification du processus

A partir des mesures précédentes, il faudra retrouver la forme et les valeurs des paramètres de la fonction de transfert du processus.

## 3. Synthèse du régulateur

Le correcteur sera calculé en fonction des performances désirées et de la forme de la fonction de transfert du processus.

# Mesure de la réponse impulsionnelle

Un processus est entièrement caractérisé par sa réponse impulsionnelle. Si celle-ci est finie (systèmes stables en boucle ouverte), la mesure de cette réponse pendant un intervalle de temps suffisamment long est suffisante pour caractériser le processus.

Nous rappelons que la réponse impulsionnelle est la sortie obtenue lorsqu'à l'entrée du processus on applique une impulsion de Dirac $\delta(t)$. En effet :

$y(t) = h(t) \otimes u(t)$

Si $u(t) = \delta(t)$ alors $y(t) = h(t)$

$Y(p) = F(p) * U(p)$

y(t) est la sortie du processus, u(t) l'entrée et h(t) la réponse impulsionnelle. Y(p), U(p) et F(p) sont les transformées de Laplace de y(t), u(t) et h(t).
$\otimes$ est le produit de convolution, * est le produit habituel.

Mais le signal $\delta(t)$ ne peut pas être en pratique appliqué à un système. Aussi, nous allons devoir retrouver h(t) à partir d'une mesure indirecte.

NB : $\delta(t) = 1$ si $t = 0$ et $\delta(t) = 0$ si $t \neq 0$.

Pour cela, nous allons utiliser les propriétés du bruit blanc; si b(t) est un bruit blanc :

- b(t) est orthogonal à tous les autres signaux
- $b(t) \otimes b(-t) = K * \delta(t)$ avec K un coefficient réel positif.

Appliquons un tel signal à l'entrée du processus, on a :

$y(t) = b(t) \otimes h(t)$

Définissons l'intercorrélation $R_{YB}(t)$ et l'autocorrélation $R_B(t)$ :

$R_{YB}(t) = y(t) \otimes b(-t)$
$R_B(t) = b(t) \otimes b(-t)$

On a :

$R_{YB}(t) = b(t) \otimes h(t) \otimes b(-t) = b(t) \otimes b(-t) \otimes h(t) = R_B(t) \otimes h(t)$

$R_B(t) = K * \delta(t)$

Compte tenu des propriétés de $\delta(t)$, $K = R_B(0)$

$R_{YB}(t) = R_B(t) \otimes h(t) = K * \delta(t) \otimes h(t) = R_B(0) * \delta(t) \otimes h(t)$

Or, $\delta(t)$ est l'élément neutre de $\otimes$, c'est à dire que $\delta(t) \otimes x(t) = x(t)$ donc :

$R_{YB}(t) = R_B(0) * h(t)$

Ce qui nous donne :

$h(t) = R_{YB}(t) / R_B(0)$

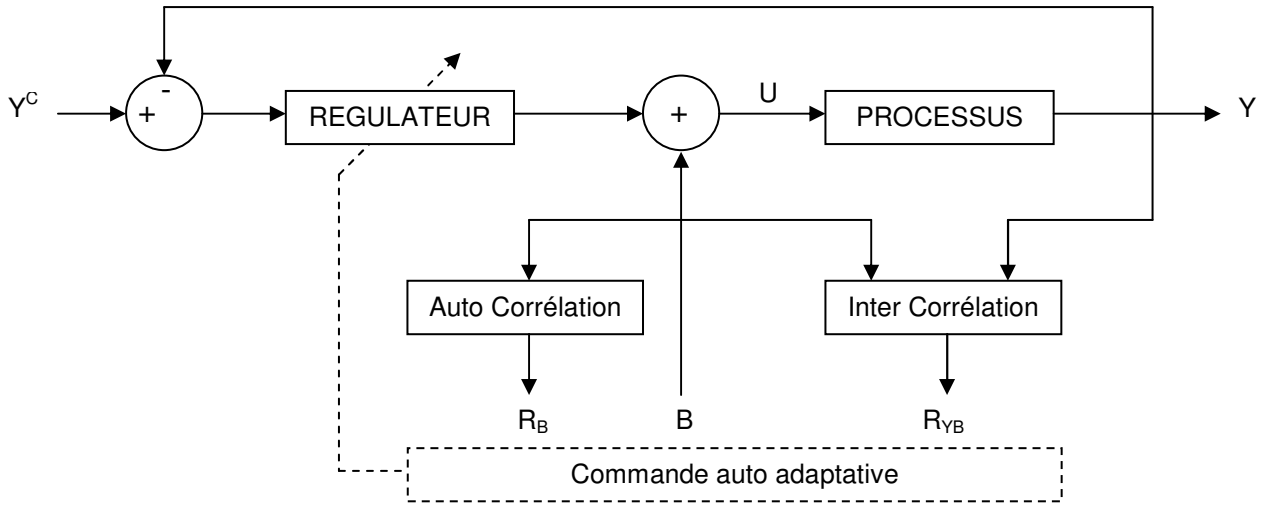
Or, b(t) et y(t) étant mesurables, $R_{YB}(t)$, $R_B(t)$ et h(t) peuvent être calculés.

Pour déterminer h(t), il suffit donc d'appliquer un bruit blanc à l'entrée du système et d'effectuer quelques calculs.

La relation $h(t) = R_{YB}(t) / R_B(0)$ est "statistiquement" bonne, cela signifie que dans la pratique, il faudra réaliser plusieurs mesures de $R_{YB}$ et $R_B$ et en faire la moyenne pour obtenir de bons résultats.

Appliquer un bruit blanc à un processus est faisable, mais conduit à un comportement imprévisible de celui-ci !

Il serai donc tentant de vouloir effectuer cette mesure pendant que le système est régulé (en utilisation). Pour ce faire, il suffit de superposer le bruit blanc b(t) à la commande u(t) envoyée au processus, comme le montre le schéma ci-dessous :




La mesure fonctionne toujours car b(t) est orthogonal avec tous les autres signaux. L'amplitude de b(t) devra être suffisamment faible pour ne pas trop perturber la régulation.

De plus cette technique permet d'envisager une commande auto adaptative.

# Identification

Il faut maintenant retrouver la forme et les valeurs des paramètres de la fonction de transfert du système.

Pour ce faire, il faut :
- répertorier les différentes formes possibles,
- pour chaque forme, calculer l'expression de h(t) en fonction des paramètres,
- chercher à identifier chaque h(t) théorique à h(t) mesuré,
- choisir la meilleure identification,

En introduction, nous nous sommes limités aux ordres 0, 1 et 2, avec ou sans retard.

Comme nous le verrons par la suite, la régulation d'un système avec retard est effectuée avec le correcteur calculé pour le système intégré dans un régulateur de Smith. Donc le premier traitement à effectuer est d'estimer le retard. Ensuite, les calculs s'effectuerons avec h(t + d).

La détection du retard s'effectue simplement : il est égal au temps mis par h(t) pour dépasser un certain seuil ε proche de 0.

Voici les expressions de h(t) correspondant aux différentes formes possibles (sans retard) :

**Gain pur :**

$F(p) = K$

$h(t) = K\,\delta(t)$

**1er ordre :**

$F(p) = K / (1 + \tau p)$

$h(t) = K/\tau \exp(- t/\tau)$

**2ième ordre :**

$F(p) = K\,\omega^2 / (p^2 + 2\,\xi\,\omega\,p + \omega^2)$

Cas $\xi < 1$ :

$h(t) = (K\,\omega / \sqrt{(1 - \xi^2)}) \exp(- \omega\,\xi\,t) \sin(\omega \sqrt{(1 - \xi^2)}\,t)$

Cas $\xi = 1$ :

$h(t) = K\,\omega^2\,t \exp(- \omega\,t)$

Cas $\xi > 1$ :

$h(t) = (K\,\omega / 2\sqrt{(\xi^2 - 1)}) [\exp(-\omega\,(\xi - \sqrt{(\xi^2 - 1)})\,t) - \exp(-\omega\,(\xi + \sqrt{(\xi^2 - 1)})\,t)]$

Au total, il y a donc 5 expressions possibles pour h(t). Nous allons essayer d'identifier h(t) mesuré avec chacune des 5 expressions possibles, avec une méthode des moindres carrés (data fiting).

La meilleure identification est celle pour laquelle la norme des résidus est la plus faible.

Ces calculs sont longs et complexes, pour les effectuer nous aurons recours à des méthodes optimisées d'analyse numérique.

# Synthèse du régulateur

Nous avons précédemment obtenu une estimation de la fonction de transfert de notre processus en boucle ouverte. Nous voulons maintenant synthétiser un correcteur pour que le système en boucle fermée aie une certaine forme.

Dans la régulation, la prise en compte du retard pur de la fonction de transfert du processus amène à un traitement particulier en deux étapes.

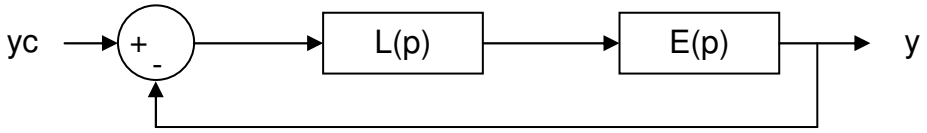
Soit : F(p) = R(p) E(p) # D(p) G(p)

F(p) est la fonction de transfert réelle (inconnue) du processus, décomposée en un retard pur R(p) et une fonction de transfert sans retard E(p).
D(p) est l'estimation de R(p).
G(p) est l'estimation de E(p).

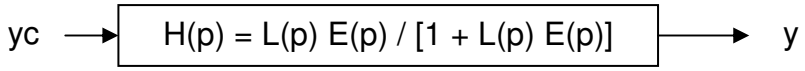
D(p) et G(p) sont l'estimation de F(p).

## 1. Synthèse du régulateur pour le processus sans retard :

Calculons dans cette première étape le régulateur L(p) permettant de réguler le processus dans retard. L(p) est calculé avec la méthode du modèle :




Est équivalent à :




On exprime les performances sous la forme d'une fonction de transfert H(p) désirée : Hd(p). On veux que H(p) = Hd(p), et sachant que E(p) est une inconnue estimé par G(p), on a :

L(p) = Hd(p) / [G(p) (1 - Hd(p))]

Rappelons qu'un Hd(p) trop proche de 1 (performances trop pointues) conduisent à une commande du processus irréaliste. En général, on choisi pour Hd(p) un deuxième ordre dont on spécifie un dépassement plus faible et un temps de réponse plus rapide que le processus en boucle ouverte.

La fonction de transfert en boucle fermée réellement obtenue est :

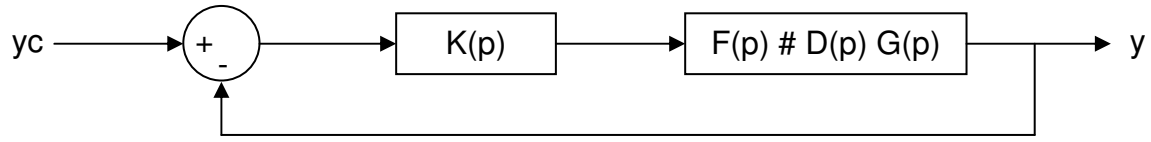
Hréel(p) = Hd(p) E(p) / [G(p) + Hd(p) (E(p) – G(p))]

Hréel(p) = Hd(p) si G(p) = E(p) c'est à dire si l'identification est parfaite.

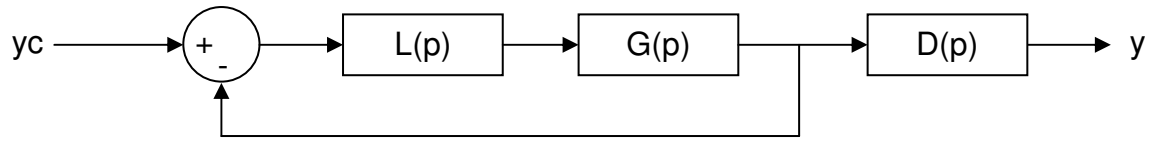
## 2. Prise en compte du retard :

Nous allons utiliser un régulateur de Smith, qui permet d'utiliser le correcteur précédemment calculé en envoyant le retard du système à la sortie.

Il faut calculer K(p) tel que :

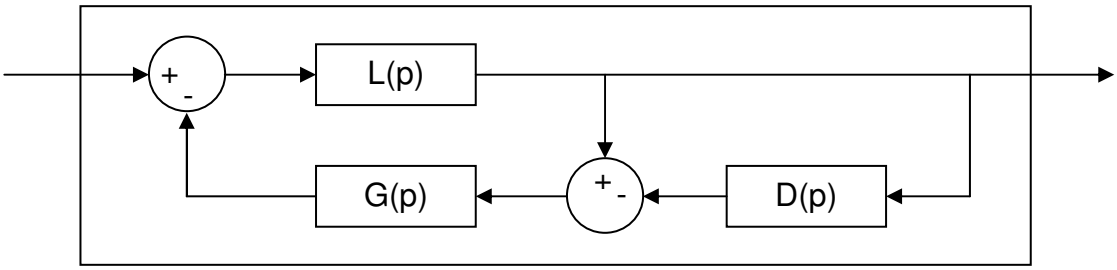



Soit équivalent à :




Par rapport à la méthode du modèle, cela revient à dire que l'on veux que H(p) = Hd(p) D(p).

Voici le correcteur de Smith K(p) :




Il vient : $K(p) = L(p) / [1 + L(p) G(p) (1 – D(p))]$

La fonction de transfert en boucle fermée réellement obtenue est :

$Hréel(p) = L(p) F(p) / [1 + L(p) (F(p) + G(p) – G(p) D(p))]$

$Hréel(p) = Hd(p) D(p)$ si l'identification est parfaite.

# Implantation sous MATLAB

[Dossier MATLAB3]

Le programme MATLAB concernant l'étude en continu se compose des fichiers suivants :


- init.m Page 40

- mesure.m Page 42

- identification.m Page 44
  - forme1.m Page 47
  - forme2.m Page 47
  - forme3a.m Page 47
  - forme3b.m Page 47
  - forme3c.m Page 47

- choix.m Page 48

- synth.m Page 49

- regul.m Page 50
  - regul_bo.mdl Page 50
  - regul_syn.mdl Page 50
  - regul_syn_smith.mdl Page 51
  - regul_ideal.mdl Page 51
  - regul_ideal_smith.mdl Page 51

- mes_perf.m Page 52

- resultats.m Page 53


Tous les fichiers se trouvent dans le dossier MATLAB3.

Ce programme effectue :

- la simulation de la mesure du la réponse impulsionnelle par corrélation,
- l'identification,
- la synthèse du régulateur à partir des mesures précédentes et la synthèse du régulateur idéal,
- la simulation du processus en boucle ouverte, en boucle fermée avec le régulateur idéal et non idéal,
- l'affichage de résultats et le traçage de courbes permettant de voir les différences entre le système identifié et le système réel, ainsi qu'entre le régulateur idéal et non idéal.

Tous les paramètres et réglages se trouvent dans le fichier init.m. Pour lancer le programme, taper "init".

Ce programme est prévu pour tourner avec MATLAB R12 ou supérieur.

## Fichier init.m

```
% Ce fichier initialise les paramètres et lance les autres programmes :

% REGLAGES DES PARAMETRES ------------------------------------------

% PARAMETRES DE LA SIMULATION :
% Durée d'une simulation :
duree = 40;
% Nombre de simulations (Durée totale de mesure = nbsim * duree)
nbsim = 400;
% Nombre de points d'une simulation ( = nombre de points
% de l'intercorrelation) :
nbpas = 400;
% Amplitude du bruit servant à la mesure :
ampl_b = 1;

% FORME DU PROCESSUS A IDENTIFIER :
% 1 = Gain [K]
% 2 = 1er ordre [K / (1 + TX p)]
% 3 = 2ième ordre [K W0² / (p² + 2 EP W0 p + W0²)]
% 4 = Gain avec retard
% 5 = 1er ordre avec retard
% 6 = 2ième ordre avec retard
% 7 = 3ième ordre pour voir [K W0² / (1 + TX p)(p² + 2 EP W0 p + W0²)]
% 8 = 3ième ordre avec retard
forme_reel = 6;

% PARAMETRES (Ne pas supprimer les paramètres non utilisés) :
% Gain K :
k_reel = 3;
% Constante de temps (1er ordre) :
tx_reel = 2;
% Pulsation (2ième ordre) :
w0_reel = 1.5;
% Coefficient d'amortissement (2ième ordre) :
ep_reel = 0.2;
% Retard :
dl_reel = 2;

% OPTIONS DE REGRESSION :
options = optimset('MaxFunEvals', 5000);

% DETECTION DU RETARD :
% Ce paramètre évite de détecter un retard trop petit à cause du bruit
% Le signal est considéré comme non nul si > Vmax * pourcent_dl_id_tol
pourcent_dl_id_tol = 0.02;

% Spécifier ici la performance que l'on désire pour le
% système en boucle fermée :
% Mettre les deux paramètres que l'on choisi de spécifier :
choix_perf = 'DM TM';
% Valeur des paramètres (ne pas effacer les lignes concernant les
% paramètres inutilisés) :
% Dépassement maximal (en % de VF) :
dm_des = 0.10;
% Temps de réponse défini à VM +/- pourcent_tr :
tr_des = 10;
% Temps de monté défini à pourcent_tm * VM :
tm_des = 1;
% Erreur en vitesse :
```

```
ev_des = 0.5;


% PARAMETRES POUR MES_PERF :
% Pourcentage definissant le nombre de valeurs prises pour calculer
% la valeur finale :
pourcent_nb_dernieres = 0.10;

% Le temps de monté est calculé de 0% à pourcent_tm de la valeur finale :
% 98% est la valeur déterminée empiriquement qui donne les
% meilleurs résultats
pourcent_tm = 0.98;

% Le temps d'établissement est définit lorsque la sortie reste autour de
% la valeur finale à +/- pourcent_tm :
pourcent_tr = 0.05;


% EXECUTION DES PROGRAMMES ------------------------------------------

% Simulation de la mesure de la réponse impulsionnelle par corrélation :
mesure;
% => t(nbpas), h_reel(nbpas)

% Identification de la réponse impulsionnelle :
identification;
% => k_id(f), tx_id(f), w0_id(f), ep_id(f), dl_id(f), norres(f)

% Choix de la meilleure identification :
choix;
% => forme_id, f_id, h_id(nbpas)

% Synthèse du correcteur conformément aux performances spécifiées :
synth;
% => c_syn, c_ideal

% Simulation du processus en boucle ouverte, et en boucle fermée :
regul;
% => y_bf_reg, y_bf_ideal, y_bo

% Mesure de la performance obtenue :
mes_perf;
% => vf_perf, tm_perf, tr_perf, dm_perf

% Affichage des résultats :
resultats;
% => Données numériques et courbes
```

## Fichier mesure.m

```
% Simulation de la mesure de la réponse impulsionnelle

pas = duree / nbpas;
t = 0:pas:duree;
rand('state',1);

% Fonction de transfert du système à identifier (h_reel est calculé
% pour comparaison) :
s = tf('s');
switch upper(forme_reel)
case {1} % F(p) = K
    f_reel = tf([k_reel],[1]);
    % h_reel = impulse(f_reel,t);
    h_reel = k_reel .* (t==0);
case {2} % F(p) = K / (1 + TX p)
    f_reel = k_reel / (1 + tx_reel * s);
    h_reel = impulse(f_reel,t);
case {3} % F(p) = K W0² / (p² + 2 EP W0 p + W0²)
    f_reel = k_reel * w0_reel^2 / (s^2 + 2*ep_reel*w0_reel*s + w0_reel^2);
    h_reel = impulse(f_reel,t);
case {4} % F(p) = exp(- p D) K
    f_reel = tf([k_reel],[1]);
    f_reel.ioDelay = dl_reel;
    % h_reel = impulse(f_reel,t);
    h_reel = k_reel .* (t-dl_reel==0);
case {5} % F(p) = exp(- p D) K / (1 + TX p)
    f_reel = k_reel / (1 + tx_reel * s);
    f_reel.ioDelay = dl_reel;
    h_reel = impulse(f_reel,t);
case {6} % F(p) = exp(- p D) K W0² / (p² + 2 EP W0 p + W0²)
    f_reel = k_reel * w0_reel^2 / (s^2 + 2*ep_reel*w0_reel*s + w0_reel^2);
    f_reel.ioDelay = dl_reel;
    h_reel = impulse(f_reel,t);
case {7} % F(p) = K W0² / (1 + TX p)(p² + 2 EP W0 p + W0²)
    f_reel = k_reel * w0_reel^2 / ((s^2 + 2*ep_reel*w0_reel*s + w0_reel^2)
* (1 + tx_reel * s));
    h_reel = impulse(f_reel,t);
case {8} % F(p) =  exp(- p D) K / (p^3 + TX p^2 + EP p + W0)
    f_reel = k_reel * w0_reel^2 / ((s^2 + 2*ep_reel*w0_reel*s + w0_reel^2)
* (1 + tx_reel * s));
    f_reel.ioDelay = dl_reel;
    h_reel = impulse(f_reel,t);
end

if f_reel.ioDelay == 0
    dl_reel = 0;
end

% Options de corrélation - Ne pas les changer, important ! :
intercorr_option = 'none';
autocorr_option = 'unbiased';

for i = 1:nbsim,
    % Simulation du système :
    u_sim = zeros(size(t)); % Pour l'instant pas de commande
    %u_sim = sin(t); % Une commande pour vérifier que ca marche

    b_sim = 2 .* ampl_b .* (rand(size(t)) - 0.5 .* ones(size(t)));
    y_sim = lsim(f_reel, u_sim + b_sim, t);
```

```
    % On retouve la réponse impusionnelle par correlation :
    if i==1
        Ryb = xcorr(y_sim, b_sim, nbpas, intercorr_option);
        Rb = xcorr(b_sim, nbpas, autocorr_option);
    else
        Ryb = Ryb + xcorr(y_sim, b_sim, nbpas, intercorr_option);
        Rb = Rb + xcorr(b_sim, nbpas, autocorr_option);
    end
end

Ryb = Ryb ./ nbsim;
Rb = Rb ./ nbsim;

xcorr_start = fix(size(Ryb,1) / 2) + 1;
h_mes = Ryb(xcorr_start:size(Ryb,1)) ./ (duree .* Rb(xcorr_start));

h_reel = h_reel';
h_mes = h_mes';
```

## Fichier identification.m

```
% Identification de la réponse impulsionnelle
% Utilise les fichiers formeXX.m

% On essaye d'identifier la courbe avec chaque forme possible :

% Les paramètres de la forme 'f' sont XX_id(f) :
k_id = zeros(6,1);  % Gain K
tx_id = zeros(6,1); % Constante de temps (1er ordre)
w0_id = zeros(6,1); % Pulsation (2ième ordre)
ep_id = zeros(6,1); % Coefficient d'amortissement (2ième ordre)
dl_id = zeros(6,1); % Retard

% Norme des résidus :
resnor = zeros(6,1);

% FORME 1 : Gain pur
f = 1;
% X = [k];
x_init = [1];
[x_sol, NormeResidus] = lsqcurvefit(['forme' num2str(f)], x_init, t, h_mes,
-Inf, Inf, options);
k_id(f) = x_sol(1);
tx_id(f) = 0;
w0_id(f) = 0;
ep_id(f) = 0;
dl_id(f) = 0;
resnor(f) = NormeResidus;

% FORME 2 : 1er ordre
f = 2;
% X = [k tx]
x_init = [1 0.1];
[x_sol, NormeResidus] = lsqcurvefit(['forme' num2str(f)], x_init, t, h_mes,
-Inf, Inf, options);
k_id(f) = x_sol(1);
tx_id(f) = x_sol(2);
w0_id(f) = 0;
ep_id(f) = 0;
dl_id(f) = 0;
resnor(f) = NormeResidus;

% FORME 3 : 2ième ordre
f = 3;
%
% FORME 3a : ep = 1
% X = [k w0]
x_init = [1 0.1];
[x_sol, NormeResidus] = lsqcurvefit(['forme' num2str(f) 'a'], x_init, t,
h_mes, -Inf, Inf, options);
resnor(f) = NormeResidus;
k_id(f) = x_sol(1);
tx_id(f) = 0;
w0_id(f) = x_sol(2);
ep_id(f) = 1;
dl_id(f) = 0;
%
% FORME 3b : ep < 1
% X = [k w0 ep]
x_init = [1 0.1 0.5];
```

```
[x_sol, NormeResidus] = lsqcurvefit(['forme' num2str(f) 'b'], x_init, t,
h_mes, -Inf, Inf, options);
if NormeResidus < resnor(f) % Forme 3b meilleure que 3a :
    resnor(f) = NormeResidus;
    k_id(f) = x_sol(1);
    tx_id(f) = 0;
    w0_id(f) = x_sol(2);
    ep_id(f) = x_sol(3);
    dl_id(f) = 0;
end
%
% FORME 3c : ep > 1
% X = [k w0 ep]
x_init = [1 0.1 1.5];
[x_sol, NormeResidus] = lsqcurvefit(['forme' num2str(f) 'c'], x_init, t,
h_mes, -Inf, Inf, options);
if NormeResidus < resnor(f) % Forme 3c meilleure que les deux autres :
    resnor(f) = NormeResidus;
    k_id(f) = x_sol(1);
    tx_id(f) = 0;
    w0_id(f) = x_sol(2);
    ep_id(f) = x_sol(3);
    dl_id(f) = 0;
end

% Recherche du retard :
h_max = h_mes(1);
for i = 2:size(t,2),
    if h_mes(i)>h_max
        h_max = h_mes(i);
    end
end
dl_id_tol = pourcent_dl_id_tol * h_max;
for i = 1:size(t,2),
    if h_mes(i)>dl_id_tol
        dl_id_i = i;
        break;
    end
end

if dl_id_i<3
    resnor(4:6) = Inf;
else
    t_retard = t(dl_id_i:size(t,2));
    t_retard = t_retard - t(dl_id_i) .* ones(size(t_retard));
    h_retard = h_mes(dl_id_i:size(t,2));

    % FORME 4 : Gain pur retardé = FORME 1 avec retard
    f = 1;
    % X = [k];
    x_init = [1];
    [x_sol, NormeResidus] = lsqcurvefit(['forme' num2str(f)], x_init,
t_retard, h_retard, -Inf, Inf, options);
    f = 4;
    k_id(f) = x_sol(1);
    tx_id(f) = 0;
    w0_id(f) = 0;
    ep_id(f) = 0;
    dl_id(f) = t(dl_id_i);
    resnor(f) = NormeResidus;
```

```
    % FORME 5 : 1er ordre retardé = FORME 2 avec retard
    f = 2;
    % X = [k tx]
    x_init = [1 0.1];
    [x_sol, NormeResidus] = lsqcurvefit(['forme' num2str(f)], x_init,
t_retard, h_retard, -Inf, Inf, options);
    f = 5;
    k_id(f) = x_sol(1);
    tx_id(f) = x_sol(2);
    w0_id(f) = 0;
    ep_id(f) = 0;
    dl_id(f) = t(dl_id_i);
    resnor(f) = NormeResidus;

    % FORME 6 : 2ième ordre retardé = FORME 3 avec retard
    % FORME 6a : ep = 1 retardé = FORME 3a avec retard
    f = 3;
    % X = [k w0]
    x_init = [1 0.1];
    [x_sol, NormeResidus] = lsqcurvefit(['forme' num2str(f) 'a'], x_init,
t_retard, h_retard, -Inf, Inf, options);
    f = 6;
    resnor(f) = NormeResidus;
    k_id(f) = x_sol(1);
    tx_id(f) = 0;
    w0_id(f) = x_sol(2);
    ep_id(f) = 1;
    dl_id(f) = t(dl_id_i);
    %
    % FORME 3b : ep < 1 retardé = FORME 3a avec retard
    f = 3;
    % X = [k w0 ep]
    x_init = [1 0.1 0.5];
    [x_sol, NormeResidus] = lsqcurvefit(['forme' num2str(f) 'b'], x_init,
t_retard, h_retard, -Inf, Inf, options);
    f = 6;
    if NormeResidus < resnor(f) % Forme 3b meilleure que 3a :
        resnor(f) = NormeResidus;
        k_id(f) = x_sol(1);
        tx_id(f) = 0;
        w0_id(f) = x_sol(2);
        ep_id(f) = x_sol(3);
        dl_id(f) = t(dl_id_i);
    end
    %
    % FORME 3c : ep > 1 retardé = FORME 3a avec retard
    f = 3;
    % X = [k w0 ep]
    x_init = [1 0.1 1.5];
    [x_sol, NormeResidus] = lsqcurvefit(['forme' num2str(f) 'c'], x_init,
t_retard, h_retard, -Inf, Inf, options);
    f = 6;
    if NormeResidus < resnor(f) % Forme 3c meilleure que les deux autres :
        resnor(f) = NormeResidus;
        k_id(f) = x_sol(1);
        tx_id(f) = 0;
        w0_id(f) = x_sol(2);
        ep_id(f) = x_sol(3);
        dl_id(f) = t(dl_id_i);
    end
end
```

**Fichier forme1.m**

```
% Fonction pour identification.m
function F = fonction(x, t)
% X = [k];
% F est la forme de la fonction à identifier
F = x(1) .* (t==0);
% h_reel = k_reel .* (t==0);
```

**Fichier forme2.m**

```
% Fonction pour identification.m
function F = fonction(x, t)
% X = [k tx];
% F est la forme de la fonction à identifier
F = x(1) .* exp(- t ./ x(2)) ./ x(2);
% h_reel = k_reel .* exp(- t ./ tx_reel) ./ tx_reel;
```

**Fichier forme3a.m**

```
% Fonction pour identification.m
function F = fonction(x, t)
% X = [k w0];
% F est la forme de la fonction à identifier
F = x(1) .* x(2)^2 .* t .* exp(- x(2) .* t);
% h_reel = k_reel .* w0_reel^2 .* t .* exp(- w0_reel .* t);
```

**Fichier forme3b.m**

```
% Fonction pour identification.m
function F = fonction(x, t)
% X = [k w0 ep];
% F est la forme de la fonction à identifier
F = (x(1).*x(2)./sqrt(1-x(3)^2)) .* exp(-x(3).*x(2).*t) .*
sin(x(2).*sqrt(1-x(3)^2).*t);
% h_reel = (k_reel.*w0_reel./sqrt(1-ep_reel^2)) .* exp(-
ep_reel.*w0_reel.*t) .* sin(w0_reel.*sqrt(1-ep_reel^2).*t);
```

**Fichier forme3c.m**

```
% Fonction pour identification.m
function F = fonction(x, t)
% X = [k w0 ep];
% F est la forme de la fonction à identifier
F = (0.5.*x(1).*x(2)./sqrt(x(3)^2-1)) .* ( exp(-x(2).*(x(3)-sqrt(x(3)^2-
1)).*t) - exp(-x(2).*(x(3)+sqrt(x(3)^2-1)).*t) );
% h_reel = (0.5.*k_reel.*w0_reel./sqrt(ep_reel^2-1)) .* ( exp(-
w0_reel.*(ep_reel-sqrt(ep_reel^2-1)).*t) - exp(-
w0_reel.*(ep_reel+sqrt(ep_reel^2-1)).*t) );
```

## Fichier choix.m

```
% Choix de la meilleure identification
resnor_min = resnor(1);
forme_id = 1;
for i = 2:6,
    if resnor(i)<resnor_min
        resnor_min = resnor(i);
        forme_id = i;
    end
end

% On calcule la réponse impulsionnelle du système identifié :
s = tf('s');
f = forme_id;
switch upper(f)
case {1} % F(p) = K
    f_id = tf([k_id(f)],[1]);
    %h_reel = impulse(f_id,t);
    h_id = k_id(f) .* (t==0);
case {2} % F(p) = K / (1 + TX p)
    f_id = k_id(f) / (1 + tx_id(f) * s);
    h_id = impulse(f_id,t);
case {3} % F(p) = K W0² / (p² + 2 EP W0 p + W0²)
    f_id = k_id(f) * w0_id(f)^2 / (s^2 + 2*ep_id(f)*w0_id(f)*s +
w0_id(f)^2);
    h_id = impulse(f_id,t);
case {4} % F(p) = exp(- p D) K
    f_id = tf([k_id(f)],[1]);
    f_id.ioDelay = dl_id(f);
    %h_reel = impulse(f_id,t);
    h_id = k_id(f) .* (t-dl_id(f)==0);
case {5} % F(p) = exp(- p D) K / (1 + TX p)
    f_id = k_id(f) / (1 + tx_id(f) * s);
    f_id.ioDelay = dl_id(f);
    h_id = impulse(f_id,t);
case {6} % F(p) = exp(- p D) K W0² / (p² + 2 EP W0 p + W0²)
    f_id = k_id(f) * w0_id(f)^2 / (s^2 + 2*ep_id(f)*w0_id(f)*s +
w0_id(f)^2);
    f_id.ioDelay = dl_id(f);
    h_id = impulse(f_id,t);
end
h_id = h_id';
```

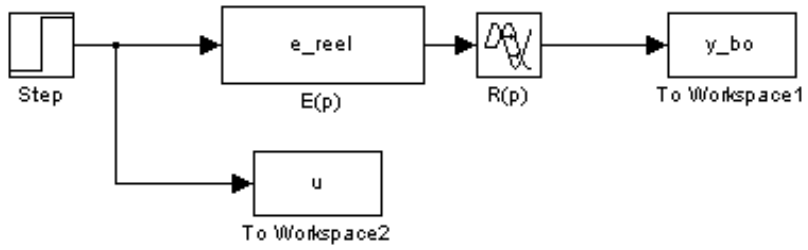
## Fichier synth.m

```
% Ce programme synthétise un correcteur en fonction des performances
désirées :

% CALCUL DE w0 ET DE ep DESIRES EN FONCTION DES PERFORMANCES DESIREES :
% Evidemment, k désiré vaut 1 !
switch upper(choix_perf)
case {'DM TM','TM DM'}
    ep_des = sqrt( ((log(dm_des))^2) ./ ((log(dm_des))^2 + pi^2) );
    w0_des = (pi - acos(ep_des)) ./ (tm_des * sqrt(1 - ep_des^2));
case {'DM TR','TR DM'}
    ep_des = sqrt( ((log(dm_des))^2) ./ ((log(dm_des))^2 + pi^2) );
    w0_des = (log (1/pourcent_tr)) ./ (tr_des * ep_des);
case {'DM EV','EV DM'}
    ep_des = sqrt( ((log(dm_des))^2) ./ ((log(dm_des))^2 + pi^2) );
    w0_des = 2 * ep_des ./ ev_des
case {'EV TR','TR EV'}
    ep_des = sqrt( 0.5 * ev_des * log(1/pourcent_tr) ./ tr_des );
    w0_des = 2 * ep_des ./ ev_des;
case {'EV TM','TM EV'}
    % Dans ce cas il faut résoudre une équation
case {'TR TM','TM TR'}
    % Dans ce cas il faut résoudre une équation
end

% IL FAUT CONCEVOIR LE CORRECTEUR POUR QUE LE SYSTEME EN B.F. SE
% COMPORTE COMME UN DEUXIEME ORDRE fbf_des DE GAIN STATIQUE 1,
% D'AMORTISSEMENT ep_des ET DE PULSATION w0_des

% Pour ce faire on utilise la méthode du modèle :
% C = sysbf_des / (F * (1 - sysbf_des))

% E(p) est F(p) sans retard :

e_reel = f_reel;
e_reel.ioDelay = 0;

e_id = f_id;
e_id.ioDelay = 0;

% On calcule les correcteurs avec F(p) sans retard :

s = tf('s');
fbf_des = w0_des^2 / (s^2 + 2*ep_des*w0_des*s + w0_des^2);
c_syn = fbf_des / (e_id * (1 - fbf_des));
c_ideal = fbf_des / (e_reel * (1 - fbf_des));

% Si il y a un retard, on utilise un correcteur de smith
% (voir fichier regul.m)
```

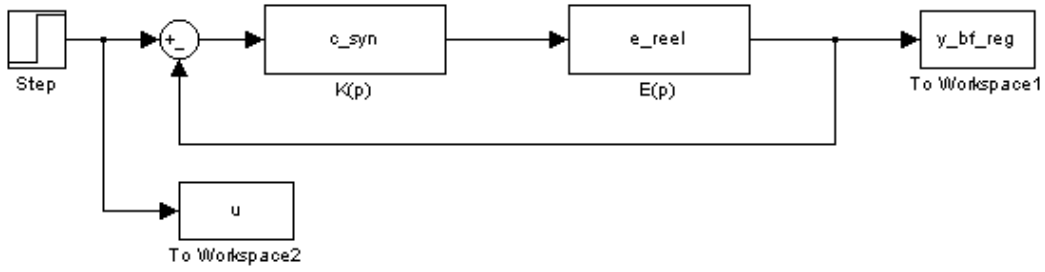
## Fichier regul.m

```
% Simulation des réponses indicielles - Utilise les fichiers regul_xxx.mdl
% Réponse indicielle système en boucle ouverte :
sim regul_bo;

% Réponse indicielle système régulé :
if f_id.ioDelay == 0
    sim regul_syn;
else
    sim regul_syn_smith;
end

% Réponse indicielle système régulé avec identification parfaite
% (cas idéal) :
if f_reel.ioDelay == 0
    sim regul_ideal;
else
    sim regul_ideal_smith;
end
```

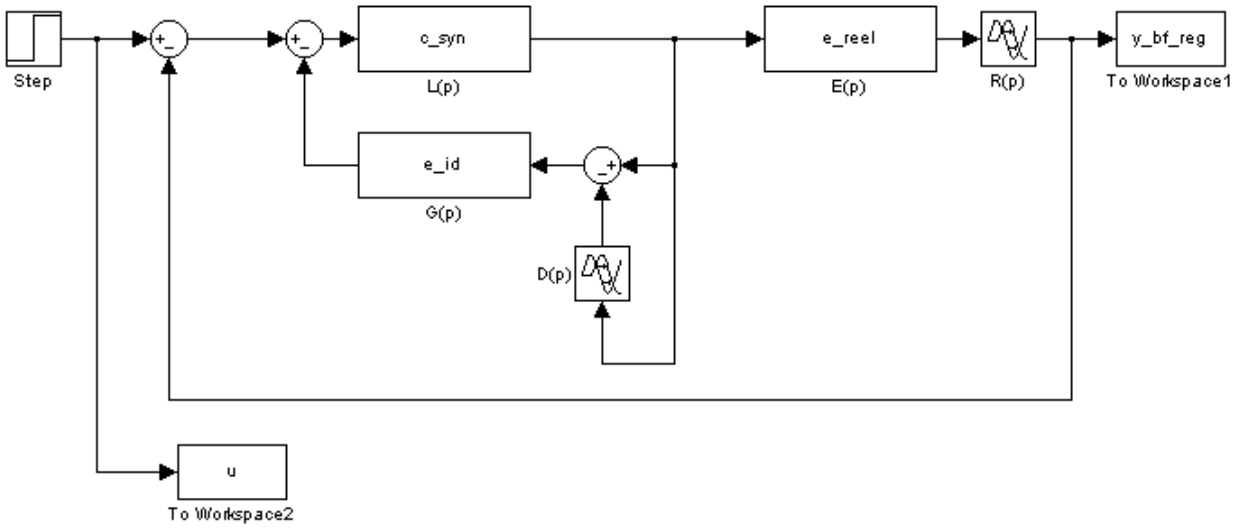
## Fichier regul_bo.mdl



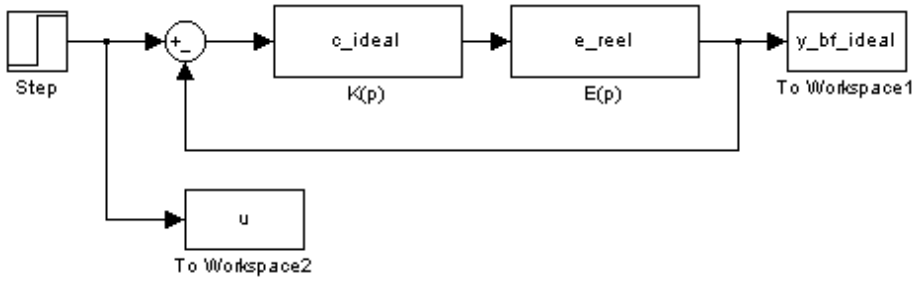

## Fichier regul_syn.mdl



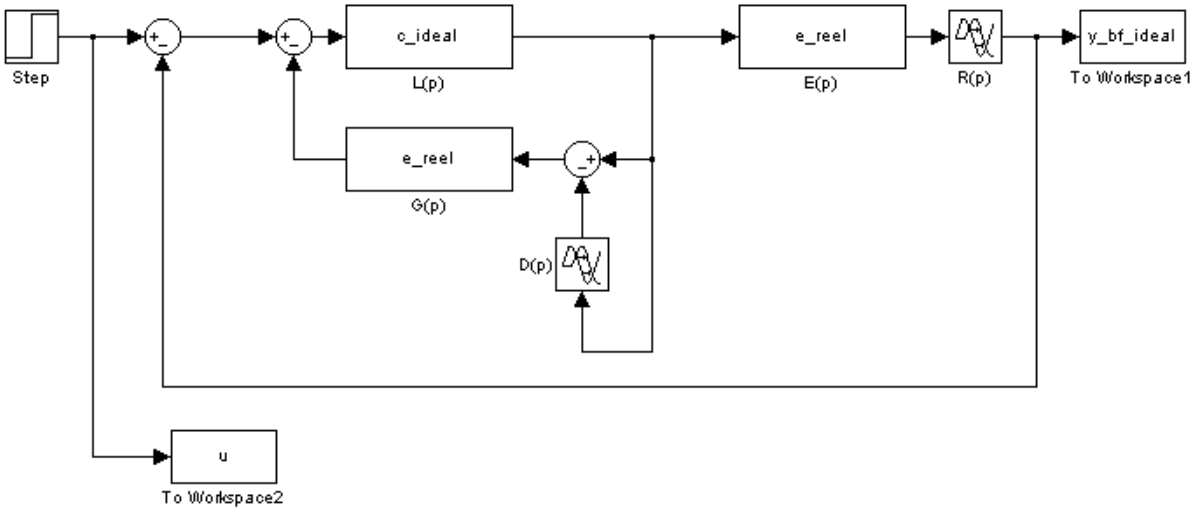
**Fichier regul_syn_smith.mdl**




**Fichier regul_ideal.mdl**




**Fichier regul_ideal_smith.mdl**



## Fichier mes_perf.m

```
% Ce programme mesure les paramètres de la courbe régulée

% RECHERCHE DE LA VALEUR FINALE :
nb_dernieres_val = round(pourcent_nb_dernieres * size(t,2));
vf_perf = sum(y_bf_reg(size(t,2)-
nb_dernieres_val+1:size(t,2)))/(nb_dernieres_val);

% RECHERCHE DE DU TEMPS DE MONTE :
for i = 1:size(t,2),
    if y_bf_reg(i)>=pourcent_tm*vf_perf
        tm_perf = t(i) - dl_id(forme_id);
        break
    end
end

% RECHERCHE DU TEMPS DE REPONSE :
for i = size(t,2):-1:1,
    if y_bf_reg(i)>=vf_perf*(1+pourcent_tr) | y_bf_reg(i)<=vf_perf*(1-
pourcent_tr)
        tr_perf = t(i) - dl_id(forme_id);
        break
    end
end

% RECHERCHE DU DEPASSEMENT MAXIMAL :
dm_perf_ymax = y_bf_reg(1);
dm_perf_t = t(1);
for i = 2:size(t,2),
    if y_bf_reg(i)>dm_perf_ymax
        dm_perf_ymax = y_bf_reg(i);
        dm_perf_t = t(i) - dl_id(forme_id);
    end
end
dm_perf = (dm_perf_ymax - vf_perf) / vf_perf;
```

## Fichier resultats.m

```
% Affichage des résultats

close all;

texte = ['----------------------------------------------------------------
---\n'];
texte = [texte 'SYSTEME REEL - '];
switch upper(forme_reel)
case {1} % F(p) = K
    texte = [texte 'Gain pur :\n'];
    texte = [texte 'K = ' num2str(k_reel) '\n'];
case {2} % F(p) = K / (1 + TX p)
    texte = [texte '1er ordre :\n'];
    texte = [texte 'K = ' num2str(k_reel) '\n'];
    texte = [texte 'TX = ' num2str(tx_reel) ' (s)\n'];
case {3} % F(p) = K W0² / (p² + 2 EP W0 p + W0²)
    texte = [texte '2ième ordre :\n'];
    texte = [texte 'K = ' num2str(k_reel) '\n'];
    texte = [texte 'W0 = ' num2str(w0_reel) ' (rad/s)\n'];
    texte = [texte 'EP = ' num2str(ep_reel) '\n'];
case {4} % F(p) = exp(- p D) K
    texte = [texte 'Gain pur retardé :\n'];
    texte = [texte 'K = ' num2str(k_reel) '\n'];
    texte = [texte 'd = ' num2str(dl_reel) ' (s)\n'];
case {5} % F(p) = exp(- p D) K / (1 + TX p)
    texte = [texte '1er ordre retardé :\n'];
    texte = [texte 'K = ' num2str(k_reel) '\n'];
    texte = [texte 'TX = ' num2str(tx_reel) ' (s)\n'];
    texte = [texte 'd = ' num2str(dl_reel) ' (s)\n'];
case {6} % F(p) = exp(- p D) K W0² / (p² + 2 EP W0 p + W0²)
    texte = [texte '2ième ordre retardé :\n'];
    texte = [texte 'K = ' num2str(k_reel) '\n'];
    texte = [texte 'W0 = ' num2str(w0_reel) ' (rad/s)\n'];
    texte = [texte 'EP = ' num2str(ep_reel) '\n'];
    texte = [texte 'd = ' num2str(dl_reel) ' (s)\n'];
case {7} % F(p) = K W0² / (1 + TX p)(p² + 2 EP W0 p + W0²)
    texte = [texte '3ième ordre :\n'];
    texte = [texte 'K = ' num2str(k_reel) '\n'];
    texte = [texte 'TX = ' num2str(tx_reel) ' (s)\n'];
    texte = [texte 'W0 = ' num2str(w0_reel) ' (rad/s)\n'];
    texte = [texte 'EP = ' num2str(ep_reel) '\n'];
case {8} % F(p) =  exp(- p D) K / (p^3 + TX p^2 + EP p + W0)
    texte = [texte '3ième ordre retardé :\n'];
    texte = [texte 'K = ' num2str(k_reel) '\n'];
    texte = [texte 'TX = ' num2str(tx_reel) ' (s)\n'];
    texte = [texte 'W0 = ' num2str(w0_reel) ' (rad/s)\n'];
    texte = [texte 'EP = ' num2str(ep_reel) '\n'];
    texte = [texte 'd = ' num2str(dl_reel) ' (s)\n'];
end
texte = [texte '\n'];

f = 1;
texte = [texte 'IDENTIFICATION - Gain pur [' num2str(resnor(f)) ']'];
if f==forme_id
    texte = [texte ' <<<--- Meilleure identification'];
end
texte = [texte '\n'];
texte = [texte 'K = ' num2str(k_id(f)) '\n'];
texte = [texte '\n'];
```

```
f = 2;
texte = [texte 'IDENTIFICATION - 1er ordre [' num2str(resnor(f)) ']'];
if f==forme_id
    texte = [texte ' <<<--- Meilleure identification'];
end
texte = [texte '\n'];
texte = [texte 'K = ' num2str(k_id(f)) '\n'];
texte = [texte 'TX = ' num2str(tx_id(f)) ' (s)\n'];
texte = [texte '\n'];

f = 3;
texte = [texte 'IDENTIFICATION - 2ième ordre [' num2str(resnor(f)) ']'];
if f==forme_id
    texte = [texte ' <<<--- Meilleure identification'];
end
texte = [texte '\n'];
texte = [texte 'K = ' num2str(k_id(f)) '\n'];
texte = [texte 'W0 = ' num2str(w0_id(f)) ' (rad/s)\n'];
texte = [texte 'EP = ' num2str(ep_id(f)) '\n'];
texte = [texte '\n'];

f = 4;
texte = [texte 'IDENTIFICATION - Gain pur retardé [' num2str(resnor(f))
']'];
if f==forme_id
    texte = [texte ' <<<--- Meilleure identification'];
end
texte = [texte '\n'];
texte = [texte 'K = ' num2str(k_id(f)) '\n'];
texte = [texte 'd = ' num2str(dl_id(f)) ' (s)\n'];
texte = [texte '\n'];

f = 5;
texte = [texte 'IDENTIFICATION - 1er ordre retardé [' num2str(resnor(f))
']'];
if f==forme_id
    texte = [texte ' <<<--- Meilleure identification'];
end
texte = [texte '\n'];
texte = [texte 'K = ' num2str(k_id(f)) '\n'];
texte = [texte 'TX = ' num2str(tx_id(f)) ' (s)\n'];
texte = [texte 'd = ' num2str(dl_id(f)) ' (s)\n'];
texte = [texte '\n'];

f = 6;
texte = [texte 'IDENTIFICATION - 2ième ordre retardé [' num2str(resnor(f))
']'];
if f==forme_id
    texte = [texte ' <<<--- Meilleure identification'];
end
texte = [texte '\n'];
texte = [texte 'K = ' num2str(k_id(f)) '\n'];
texte = [texte 'W0 = ' num2str(w0_id(f)) ' (rad/s)\n'];
texte = [texte 'EP = ' num2str(ep_id(f)) '\n'];
texte = [texte 'd = ' num2str(dl_id(f)) ' (s)\n'];

texte = [texte '\n'];
texte = [texte 'PERFORMANCE DESIREE EN BOUCLE FERMEE (Réponse indicielle)
:\n'];
texte = [texte 'K  = 1\n'];
```

```
switch upper(choix_perf)
case {'DM TM','TM DM'}
    texte = [texte 'DM = ' num2str(dm_des*100) ' (%%VF)\n'];
    texte = [texte 'TM (0 -> ' num2str(pourcent_tm*100) '%%) = '
num2str(tm_des) ' (s)\n'];
case {'DM TR','TR DM'}
    texte = [texte 'DM = ' num2str(dm_des*100) ' (%%VF)\n'];
    texte = [texte 'TR (VF +/- ' num2str(pourcent_tr*100) '%%) = '
num2str(tr_des) ' (s)\n'];
case {'DM EV','EV DM'}
    texte = [texte 'DM = ' num2str(dm_des*100) ' (%%VF)\n'];
    texte = [texte 'EV = ' num2str(ev_des) ' \n'];
case {'EV TR','TR EV'}
    texte = [texte 'TR (VF +/- ' num2str(pourcent_tr*100) '%%) = '
num2str(tr_des) ' (s)\n'];
    texte = [texte 'EV = ' num2str(ev_des) ' \n'];
case {'EV TM','TM EV'}
    texte = [texte 'TM (0 -> ' num2str(pourcent_tm*100) '%%) = '
num2str(tm_des) ' (s)\n'];
    texte = [texte 'EV = ' num2str(ev_des) ' \n'];
case {'TR TM','TM TR'}
    texte = [texte 'TM (0 -> ' num2str(pourcent_tm*100) '%%) = '
num2str(tm_des) ' (s)\n'];
    texte = [texte 'TR (VF +/- ' num2str(pourcent_tr*100) '%%) = '
num2str(tr_des) ' (s)\n'];
end
texte = [texte '\n'];
texte = [texte 'PERFORMANCE OBTENUE :\n'];
texte = [texte 'K  = ' num2str(vf_perf/1) '\n'];
switch upper(choix_perf)
case {'DM TM','TM DM'}
    texte = [texte 'DM = ' num2str(dm_perf*100) ' (%%VF)\n'];
    texte = [texte 'TM (0 -> ' num2str(pourcent_tm*100) '%%) = '
num2str(tm_perf) ' (s)\n'];
case {'DM TR','TR DM'}
    texte = [texte 'DM = ' num2str(dm_perf*100) ' (%%VF)\n'];
    texte = [texte 'TR (VF +/- ' num2str(pourcent_tr*100) '%%) = '
num2str(tr_perf) ' (s)\n'];
case {'DM EV','EV DM'}
    texte = [texte 'DM = ' num2str(dm_perf*100) ' (%%VF)\n'];
    texte = [texte 'EV = ' num2str(ev_perf) ' \n'];
case {'EV TR','TR EV'}
    texte = [texte 'TR (VF +/- ' num2str(pourcent_tr*100) '%%) = '
num2str(tr_perf) ' (s)\n'];
    texte = [texte 'EV = ' num2str(ev_perf) ' \n'];
case {'EV TM','TM EV'}
    texte = [texte 'TM (0 -> ' num2str(pourcent_tm*100) '%%) = '
num2str(tm_perf) ' (s)\n'];
    texte = [texte 'EV = ' num2str(ev_perf) ' \n'];
case {'TR TM','TM TR'}
    texte = [texte 'TM (0 -> ' num2str(pourcent_tm*100) '%%) = '
num2str(tm_perf) ' (s)\n'];
    texte = [texte 'TR (VF +/- ' num2str(pourcent_tr*100) '%%) = '
num2str(tr_perf) ' (s)\n'];
end

sprintf(texte)

plot (t, h_reel, 'b');
hold on;
plot (t, h_mes, 'r');
```

```
title('Qualité de la mesure de h(t) par intercorrélation');
legend('Réponse impuslionnelle théorique','Réponse impulsionnelle
mesurée');
xlabel('Temps (s)');
ylabel('h-reel(t), h-mes(t)');

figure;
plot (t, h_reel, 'b');
hold on;
plot (t, h_id, 'r');
title('Reponse du systéme identifié VS Réponse théorique');
legend('Réponse impuslionnelle théorique','Réponse impulsionnelle
identifiée');
xlabel('Temps (s)');
ylabel('h-reel(t), h-id(t)');

figure;
plot (t, y_bf_ideal, 'b');
hold on;
plot (t, y_bf_reg, 'r');
plot (t, y_bo, 'k');
title('Efficacité régulation - Réponse indicielle');
legend('Réponse en B.F. idéale', 'Réponse en B.F. obtenue', 'Réponse en
B.O.');
xlabel('Temps (s)');
ylabel('y(t)');
```

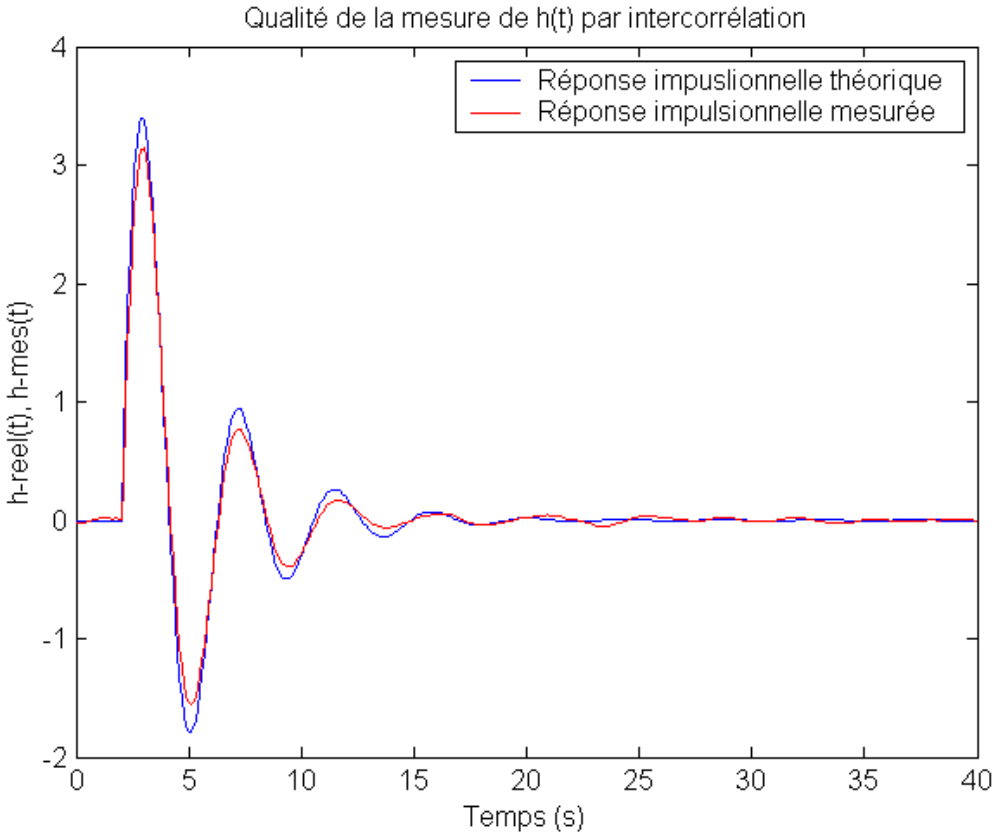
# Tests

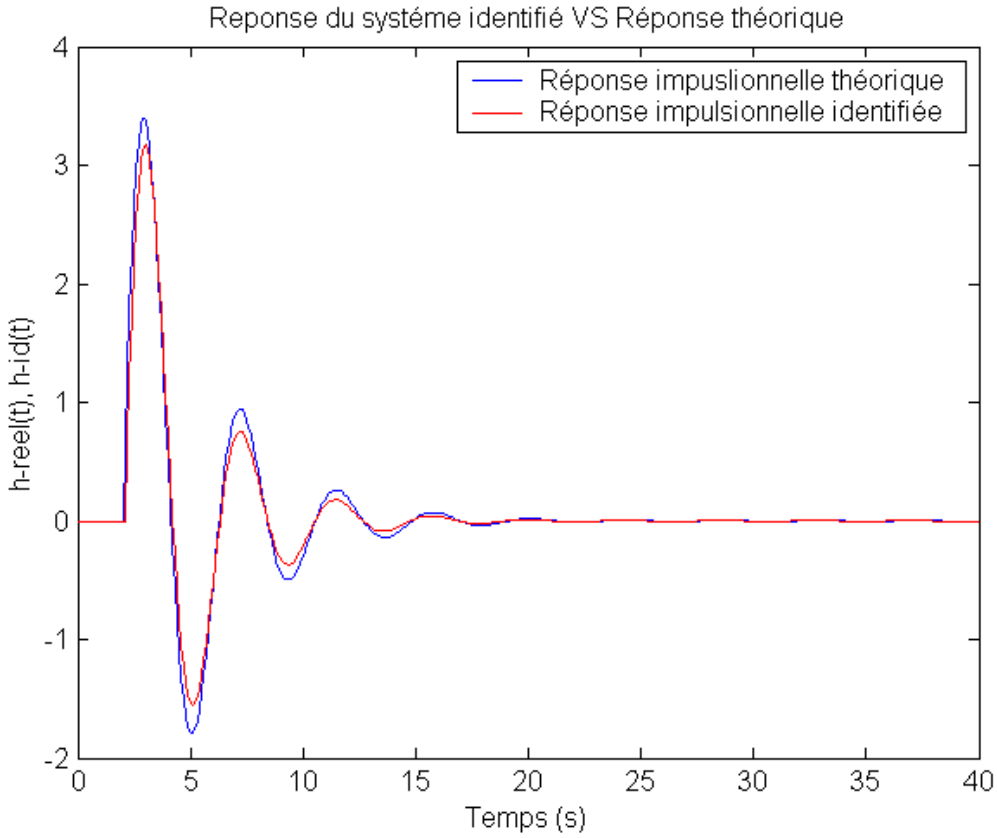
Lançons le fichier init.m sans changer les paramètres :

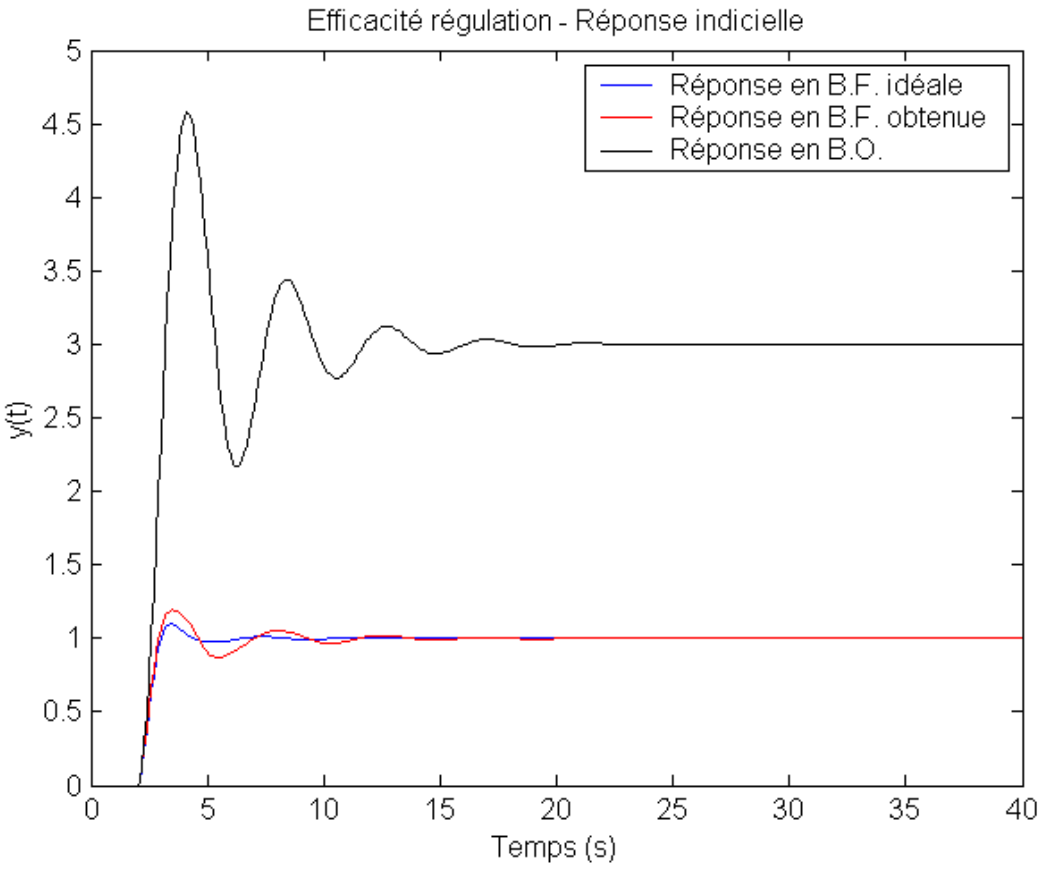
Les résultats suivant s'affichent dans la fenêtre de commande :

```
-----------------------------------------------------------------
SYSTEME REEL - 2ième ordre retardé :
K = 3
W0 = 1.5 (rad/s)
EP = 0.2
d = 2 (s)

IDENTIFICATION - Gain pur [137.8266]
K = -0.025854

IDENTIFICATION - 1er ordre [137.8248]
K = -0.0090889
TX = 0.24634 (s)

IDENTIFICATION - 2ième ordre [102.7434]
K = 2.6171
W0 = 0.67577 (rad/s)
EP = 0.33037

IDENTIFICATION - Gain pur retardé [137.7181]
K = 0.32002
d = 2.1 (s)

IDENTIFICATION - 1er ordre retardé [81.5538]
K = 3.6822
TX = 1.2999 (s)
d = 2.1 (s)

IDENTIFICATION - 2ième ordre retardé [0.64696] <- Meilleure identification
K = 2.8308
W0 = 1.5283 (rad/s)
EP = 0.22319
d = 2.1 (s)

PERFORMANCE DESIREE EN BOUCLE FERMEE (Réponse indicielle) :
K  = 1
DM = 10 (%VF)
TM (0 -> 98%) = 1 (s)

PERFORMANCE OBTENUE :
K  = 0.99999
DM = 19.3415 (%VF)
TM (0 -> 98%) = 0.8 (s)
```

Les graphiques suivants apparaissent à l'écran :






**Commentaires :**

La réponse impulsionnelle identifiée est assez proche de celle du processus. Cette faible différence a un impact relativement important sur la qualité de la régulation, comme le montre le troisième graphique et la différence entre les performances désirées et les performances obtenues.

La mesure par corrélation est très longue; mais si nous essayons de diminuer le temps de mesure, en baissant dans init.m les valeurs de nbsim, la qualité de l'identification s'en ressens.

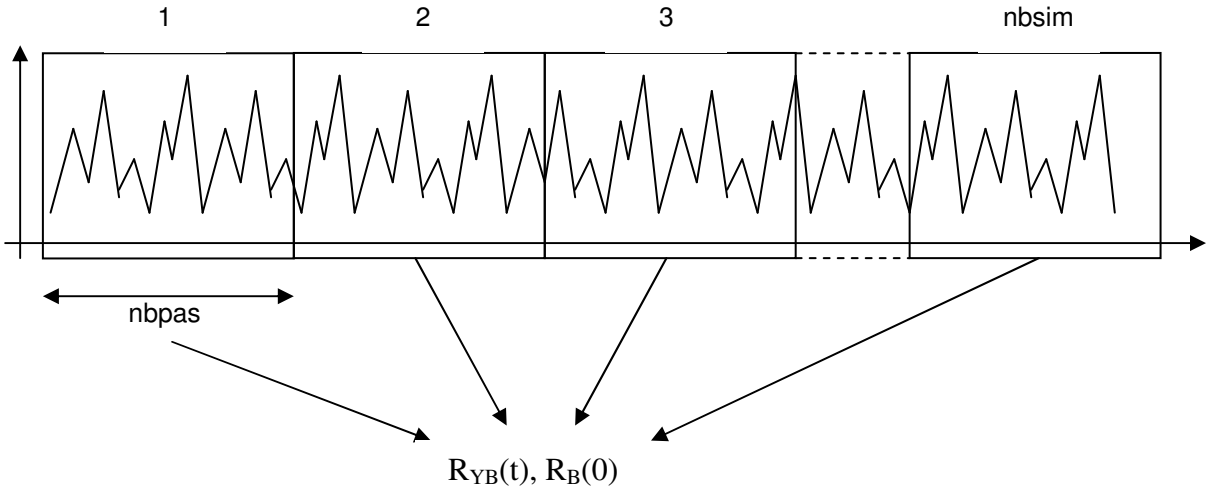
Une valeur trop faible de nbsim peut conduire à une régulation instable ! En effet, la méthode du modèle n'est pas robuste si le système est mal identifié.

Ce problème lié à la qualité de l'identification est d'autant plus important que nous ne cherchons à identifier que un système d'ordre 0, 1 ou 2. Mais dans la pratique, de nombreux processus sont d'ordre quelconque.

# Conclusion

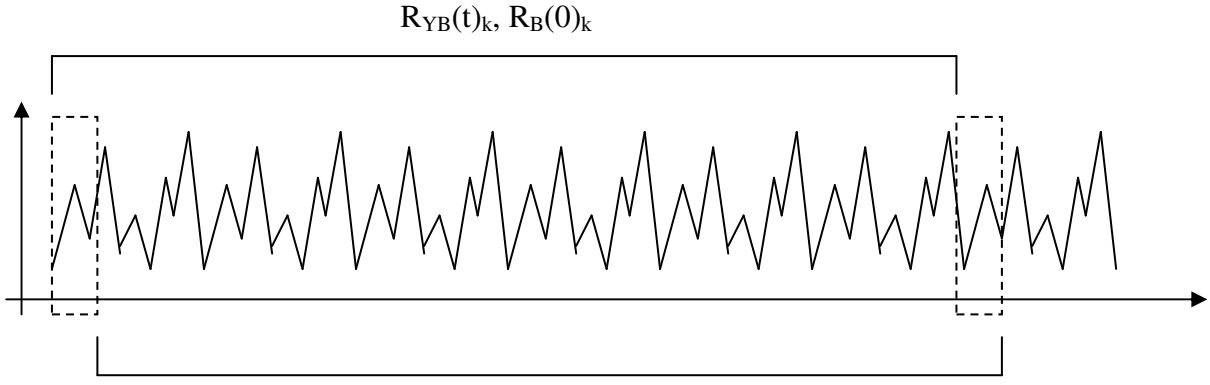
## 1. Vers un régulateur auto adaptatif

Pour calculer $R_{YB}(t)$ et $R_B(0)$, le programme effectue d'abord nbsim mesures de nbpas chacune, puis il en fait la moyenne :




Implanter cette méthode telle quelle sur un automate revient à dire qu'il faut d'abord mesurer et stocker nbsim * nbpas points, puis ensuite faire les calculs.

Mais si nous exécutons le programme sur un automate en temps réel, nous pouvons mettre en œuvre un algorithme à facteur d'oubli : A l'instant k, la mesure est intégrée dans le calcul de la moyenne alors que la contribution de la mesure de l'instant (k - nbpas * nbsim) est retranchée à la moyenne.




$R_{YB}(t)_{k+1}$, $R_B(0)_{k+1}$ =
$R_{YB}(t)_k$, $R_B(0)_k$ + (contribution du point k + 1) - (contribution du point k - nbsim * nbpas)

Ces deux opérations donnent le même résultat que si l'on effectuait le calcul d'identification en intégralité à chaque instant d'échantillonnage. De cette façon, on aboutit à un régulateur auto adaptatif, puisque le correcteur est synthétisé à partir du processus identifié à chaque période d'échantillonnage.

## 2. Limites du continu

L'avantage de travailler en continu est de toujours manipuler des grandeurs physiques, pouvant être facilement interprétées.
L'inconvénient vient du nombre et de la complexité des calculs à effectuer.

Ainsi, la procédure d'identification exécute six régressions non linéaires, et elle n'est capable de traiter uniquement les processus d'ordre égal ou inférieur à 2.

La technique de régulation que nous venons de mettre au point n'est donc qu'a utiliser dans le cas où le régulateur est analogique, constitué de filtres actifs utilisant AOP résistances et condensateurs. Dans ce cas, les techniques mises en œuvre dans le programme MATLAB pourront servir à identifier le processus et à concevoir le filtre analogique.

La commande analogique d'un processus avec un calculateur est inenvisageable. En fait, ce que nous avons fait avec MATLAB n'est ni plus moins que de la commande analogique discrétisée.

La deuxième partie de ce projet sera donc l'étude en discret.

# Etude en discret

SOMMAIRE


| | |
|---|---|
| Introduction | Page 63 |
| Démarche pour la synthèse du régulateur | Page 64 |
| 1er essai | Page 66 |
| 2ième essai | Page 72 |
| 3ième essai | Page 89 |
| 4ième essai | Page 94 |
| Conclusion | Page 96 |

# Introduction

Le problème est le même qu'en continu, à savoir :

Nous avons un processus totalement inconnu, d'un ordre quelconque, avec ou sans retard. L'objectif est de réguler ce processus de manière a obtenir certaines performances (caractéristiques de la réponse indicielle).

Mais ici, nous allons chercher à mettre en œuvre un algorithme travaillant en discret, c'est à dire avec des transformées en z et des équations récurrentes.

Voici comment nous allons résoudre ce problème :

## 0. Hypothèses simplificatrices

Comme précédemment, nous nous plaçons dans le cas de l'aide au réglage, c'est à dire avec une phase d'identification et une phase de régulation distinctes.

Nous ne considérerons que les ordres suivants : gain pur, $1^{er}$ ordre, $2^{ième}$ ordre.

## 1. Mesure de la réponse impulsionnelle

Nous utiliserons la même mesure que dans l'étude en continu (mesure par corrélation).

## 2. Identification du processus.

A partir des mesures précédentes, il faudra retrouver la forme et les valeurs des paramètres de la fonction de transfert du processus. L'identification se fera en utilisant des méthodes de résolutions d'algèbre linéaire. Différentes méthodes seront testées au cours de plusieurs essais.

## 3. Synthèse du régulateur

Le correcteur sera calculé en fonction des performances désirées et de la forme de la fonction de transfert du processus. Pour cela nous emploierons une démarche déduite de la méthode du modèle.

## Quelques rappels :

Transformée en z : $Z(x(t)) = x(0) + x(1)\, z^{-1} + \ldots + x(k)\, z^{-k} + \ldots$

La fonction de transfert d'un système est définie par deux polynômes A et B :
$H = B / A = (b_0 + b_1\, z + \ldots + b_m\, z^m) / (a_0 + a_1\, z + \ldots + a_{n-1}\, z^{n-1} + z^n)$

Ce qui corresponds à l'équation récurrente :
$a_0\, y_k + \ldots + a_{n-1}\, u_{k+n-1} + y_{k+n} = b_0\, u_k + \ldots\, b_m\, u_{k+m}$

Le terme de plus haut degré de A est égal à 1 ($a_n = 1$).
Pour que H soit causal, il faut que $n \geq m$.

# Démarche pour la synthèse du régulateur

Voici la démarche que nous allons suivre pour la synthèse du régulateur. Cette démarche dérive de la méthode du modèle, et aboutit à un régulateur de type RST.

## 1. Données de départ :

**Le processus identifié** : H = B / A

*Les différentes méthodes d'identification testées seront vues en détail dans des parties suivantes (1$^{er}$ essai à 4$^{ième}$ essai).*

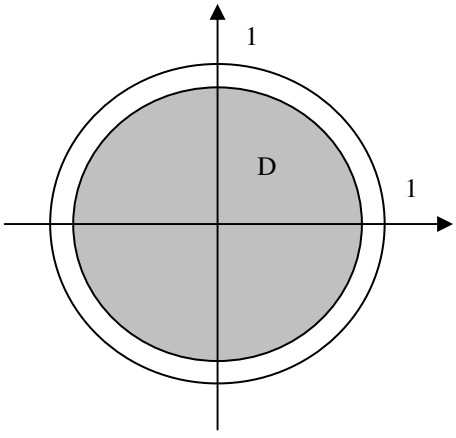
**Le modèle** (fonction de transfert désirée en boucle fermée) : Hd = Bm / Am

Le modèle doit être choisi tel que d° Am - d° Bm ≥ d° A - d° B

**Le nombre d'intégration i souhaités dans le correcteur** (assure la précision pour un type d'entrée donnée).

**Le domaine D définissant la zone des zéros compensables.** En effet, lors de la synthèse du régulateur, on compense les zéros de H en ajoutant des pôles au correcteur, mais il ne faut pas introduire de pôle instable. Un zéro qui serai compensé par un pôle instable est improprement appelé "zéro instable". D permet de définir une zone dans laquelle on considère que les zéros peuvent être compensés sans risquer d'introduire un pôle instable dans le correcteur.

D doit donc être inclus dans le cercle unité. Il faut tenir compte des erreurs liées à l'identification et ne pas inclure dans D les zéros situés trop proches de la zone limite, comme le montre le schéma ci-dessous :



## 2. Calcul :

D'abord, il faut factoriser B sous la forme B = Bs * Bi avec :

$Bs = (z - z_0) (z - z_1) \ldots$

$Bi = K (z - z'_0) (z - z'_1) \ldots$

Les nombres $z_i$ sont les zéros de B compensables (inclus dans D), et les $z'_i$ sont les zéros non compensables.

Pour permettre la synthèse du régulateur, nous devons introduire dans la suite des calculs un polynôme $A_0$, de degré suffisamment élevé pour des raisons de causalité. Il faut que :

$d° A0 \geq 2 * d° A - d° Am - d° Bs + i - 1$

Ensuite il faut calculer le polynôme Bm' tel que Bm = Bi Bm'. En effet, comme les zéros non compensables ne seront pas compensés par le correcteur, ils apparaîtront dans la fonction de transfert en boucle fermée. Il faut donc les inclure dans le modèle. En fait, Bm / Am est la fonction de transfert désirée, et Bi * Bm' / Am est la fonction de transfert réalisable.

Le calcul de Bm' se fait par identification.

Puis il faut résoudre l'identité de Bézou :

$(z - 1)^i A R' + Bi S = A0 Am$

Avec :

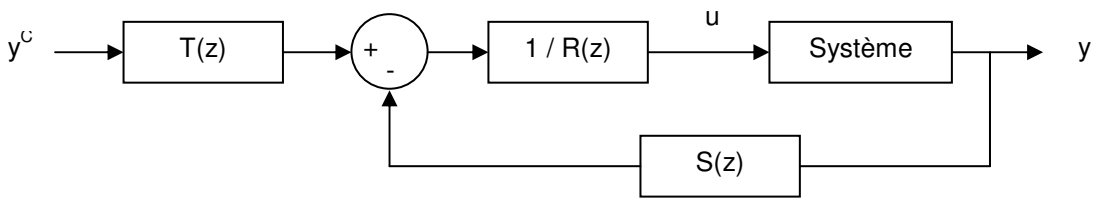
$d° R' = d° A0 + d° Am - d°A - i$
$d° S \leq d° A + i - 1$
Le coefficient du terme de plus haut degré de R' doit valoir 1.

Finalement on calcule les polynômes R et T :

$R = Bs (z - 1)^i R'$

$T = Bm' A0$

Voici la loi de commande : $u(z) = [y^C(z) T(z) - S(z) y(z)] / R(z)$




*Les moyens mis en œuvres pour réaliser ces calculs seront vus en détail dans des parties suivantes (1er essai à 4ième essai).*

# 1er essai

[Dossier MATLAB5]

## 1. Identification :

La méthode de mesure par corrélation (voir la partie "étude en continu") nous donne une estimation de la réponse impulsionnelle h(t) échantillonnée.

Nous allons devoir identifier le système à partir de cette mesure, c'est à dire trouver les polynômes A et B.

A et B sont totalement inconnus. Partons de l'équation récurrente :

$a_0\ y_k + \ldots + a_{n-1}\ u_{k+n-1} + y_{k+n} = b_0\ u_k + \ldots\ b_m\ u_{k+m}$

ou

$y_{k+n} = b_0\ u_k + \ldots\ b_m\ u_{k+m} - (a_0\ y_k + \ldots + a_{n-1}\ u_{k+n-1})$

Pour chaque valeur de yk+n mesurée, cette équation est vérifiée. Si nous avons p mesures, nous obtenons donc un système à p équations et à m + n + 1 inconnues.

Les degrés m et n vérifient $m \leq n$. Comme A et B sont quelconques, en choisissant m = n nous sommes sûrs de prendre en compte tous les cas de figures.

h(t) représente la valeur de la sortie y lorsque l'entrée u est une impulsion de Dirac. Nous disposons de la valeur de h(t) sur p points; en choisissant n tel que $p \geq 2\ n + 1$ nous obtenons un système PQR, sur conditionné si $p > 2\ n + 1$.

| P | = | | | | Q | | | | | * | R |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $y_0$ | | $u_{-n}$ | $u_{-n+1}$ | ... | $u_0$ | $y_{-n}$ | $y_{-n+1}$ | ... | $y_{-1}$ | | $b_0$ |
| $y_1$ | | $u_{-n+1}$ | $u_{-n+2}$ | ... | $u_1$ | $y_{-n+1}$ | $y_{-n+2}$ | ... | $y_0$ | | ... |
| $y_2$ | | $u_{-n+2}$ | $u_{-n+3}$ | ... | $u_2$ | $y_{-n+2}$ | $y_{-n+3}$ | ... | $y_1$ | | $b_n$ |
| ... | = | ... | ... | ... | ... | ... | ... | ... | ... | * | $-a_0$ |
| $y_{p-1}$ | | $u_{p-1-n}$ | $u_{p-n}$ | ... | $u_{p-1}$ | $y_{p-1-n}$ | $y_{p-n}$ | ... | $y_{p-2}$ | | ... |
| | | | | | | | | | | | $-a_{n-1}$ |

Le résultat est $R = Q^{-1}\ P$ (ou $R = (Q^T\ .\ Q)^{-1}\ .\ Q^T\ .\ P$ si le système est sur conditionné)
Si le système est sur conditionné, l'erreur commise sur chaque $y_i$ est $E = P - Q\ .\ R$

Voici les valeurs de u et de y :
$u_i = 1$ si $i = 0$, sinon $u_i = 0$
$y_i = h(t = Te * i)$, avec Te la période d'échantillonnage.

On constate immédiatement que la matrice Q va contenir un nombre important de zéros dans sa partir droite. Cela implique un gaspillage de l'espace mémoire et du temps de calcul perdu.

On va donc procéder plus intelligemment en divisant ce système en deux sous systèmes :

1er sous système :

| P | = | Q | | | | * | R |
|---|---|---|---|---|---|---|---|
| $y_{n+1}$ | | $y_1$ | $y_2$ | ... | $y_n$ | | $-a_0$ |
| $y_{n+2}$ | | $y_2$ | $y_3$ | ... | $y_{n+1}$ | | ... |
| ... | = | ... | ... | ... | ... | | $-a_{n-1}$ |
| $y_{p-1}$ | | $y_{p-1-n}$ | $y_{p-n}$ | ... | $y_{p-2}$ | | |

Le résultat $R = (Q^T . Q)^{-1} . Q^T . P$ donne les coefficients de A.

2ième sous système :

Le 2ième système est de type triangulaire et permet d'obtenir les coefficients de B :

$b_n = y_0$
$b_{n-1} = y_1 + a_{n-1} y_0$
...
$b_{n-i} = y_i + a_{n-1} y_{i-1} + a_{n-2} y_{i-2} + ... + a_{n-i} y_0$
...
$b_0 = y_n + a_{n-1} y_{n-1} + a_{n-2} y_{n-2} + ... + a_0 y_0$

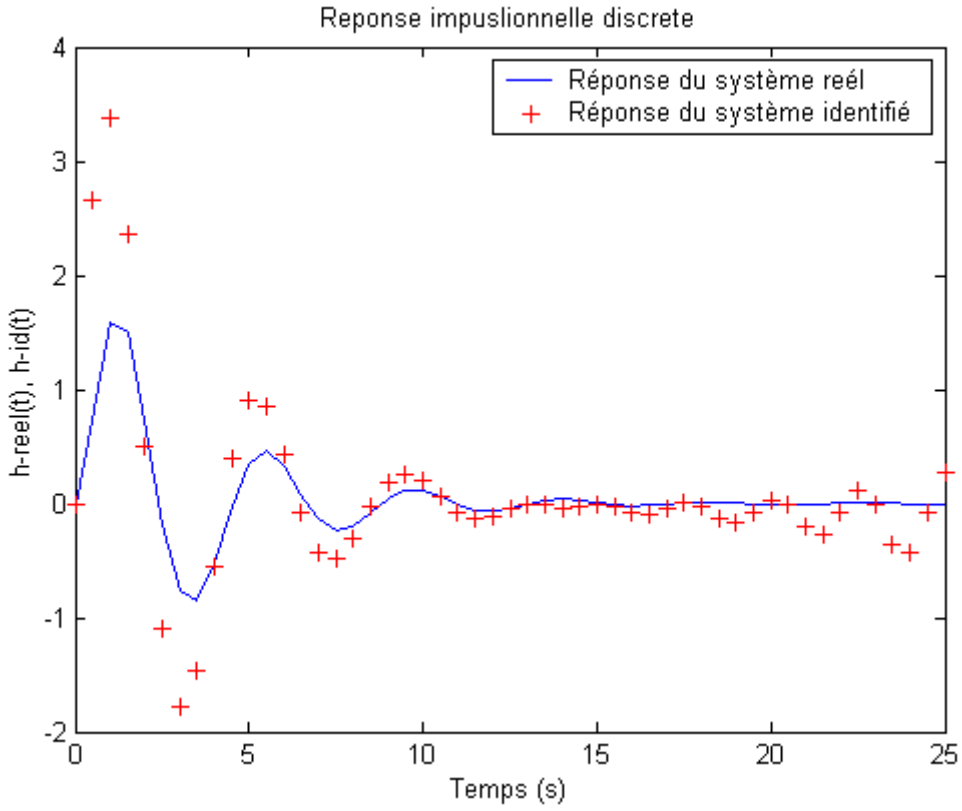
## 2. Tests avec MATLAB :

Les fichiers concernant cet essai se trouvent dans le dossier MATLAB5. Le script à exécuter est init.m.

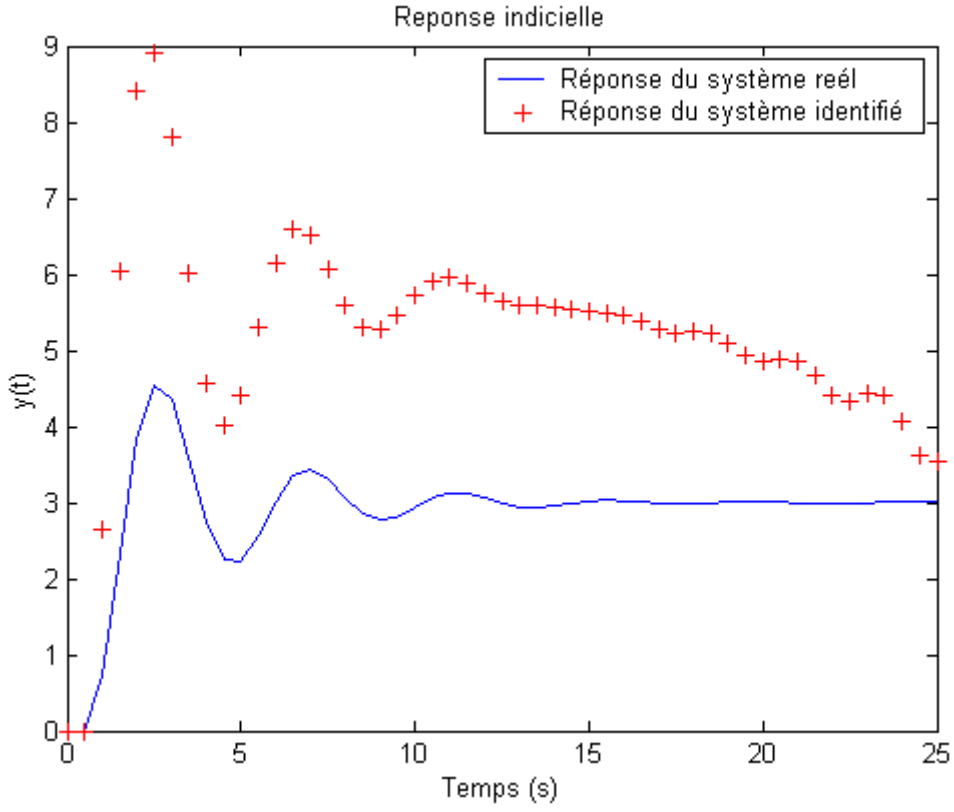
Pour tester la méthode d'identification, nous comparons les réponses impuslionnelles, indicielles et à une rampe du système réel et du système identifié.

Les résultats obtenus sont très mauvais. Voyons d'abord le cas idéal, c'est à dire avec une mesure parfaite (h_mes = h_reel) :

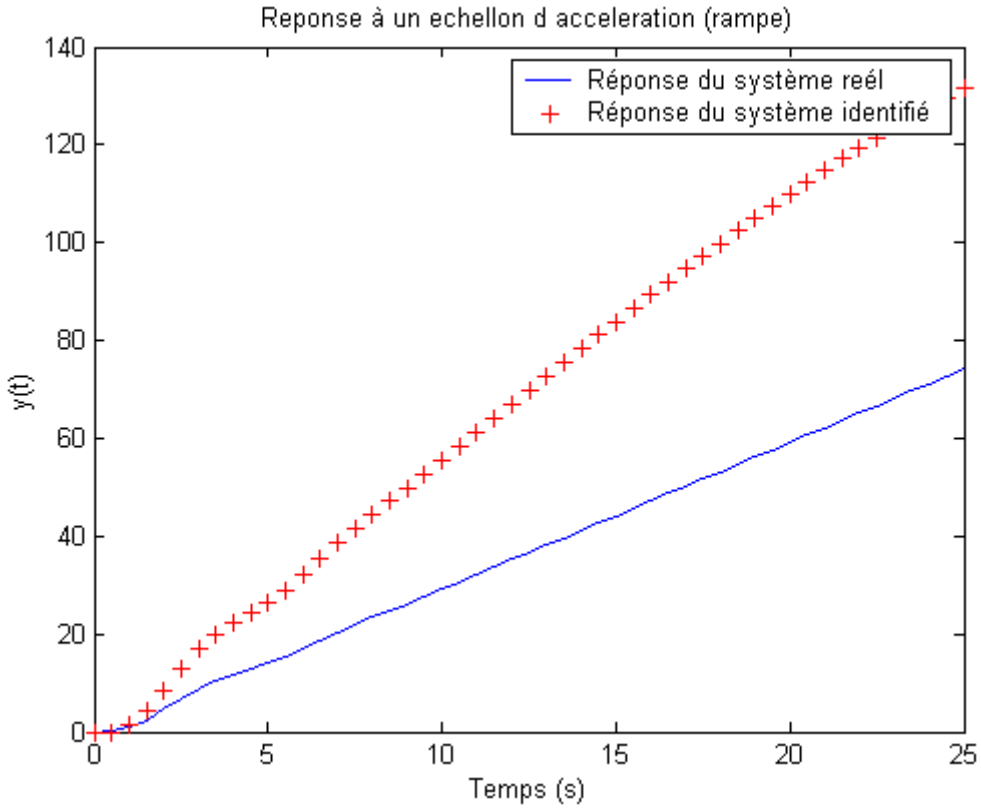
Réponses impulsionnelles :



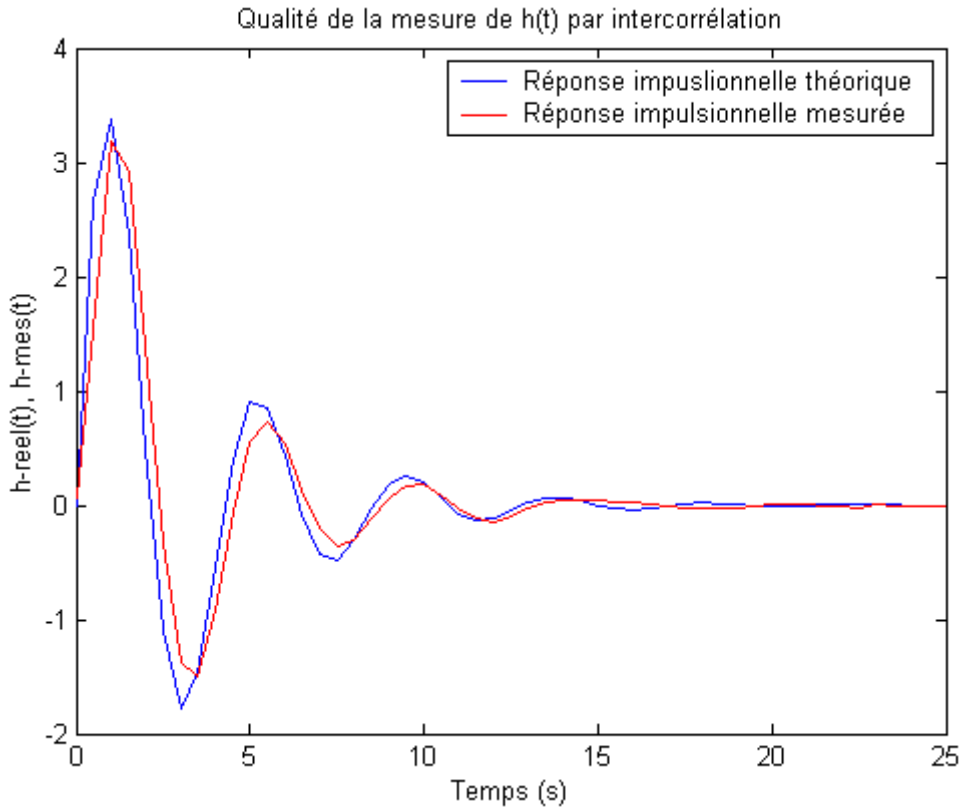

Réponses indicielles :



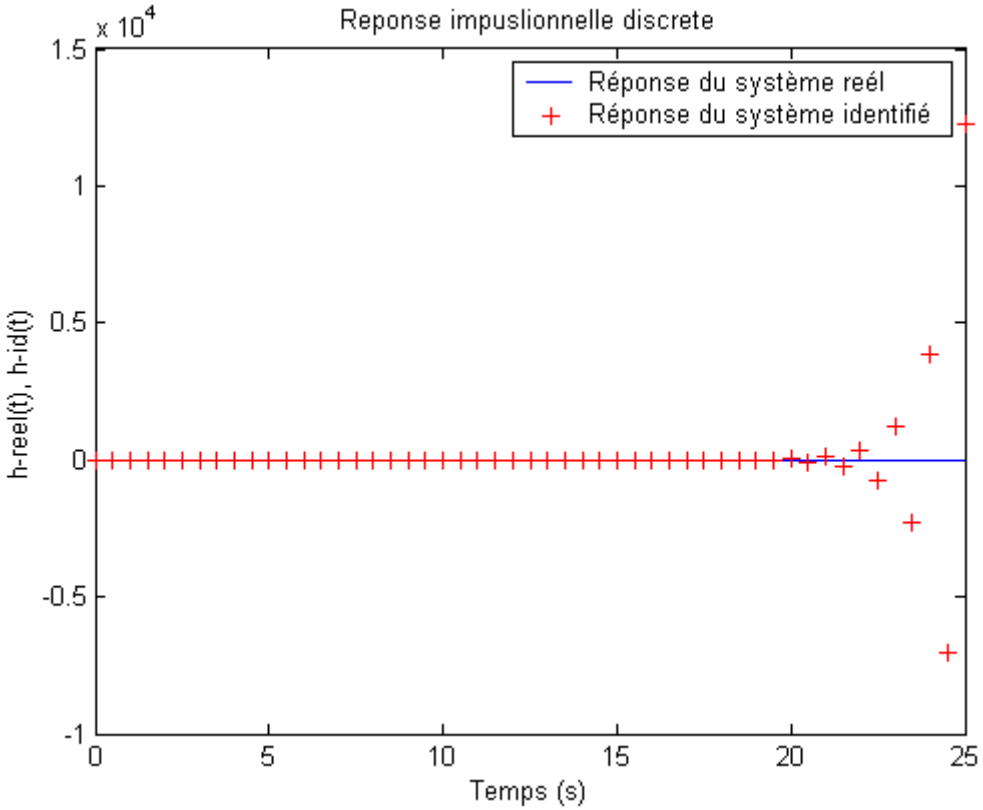
Réponses à une rampe :



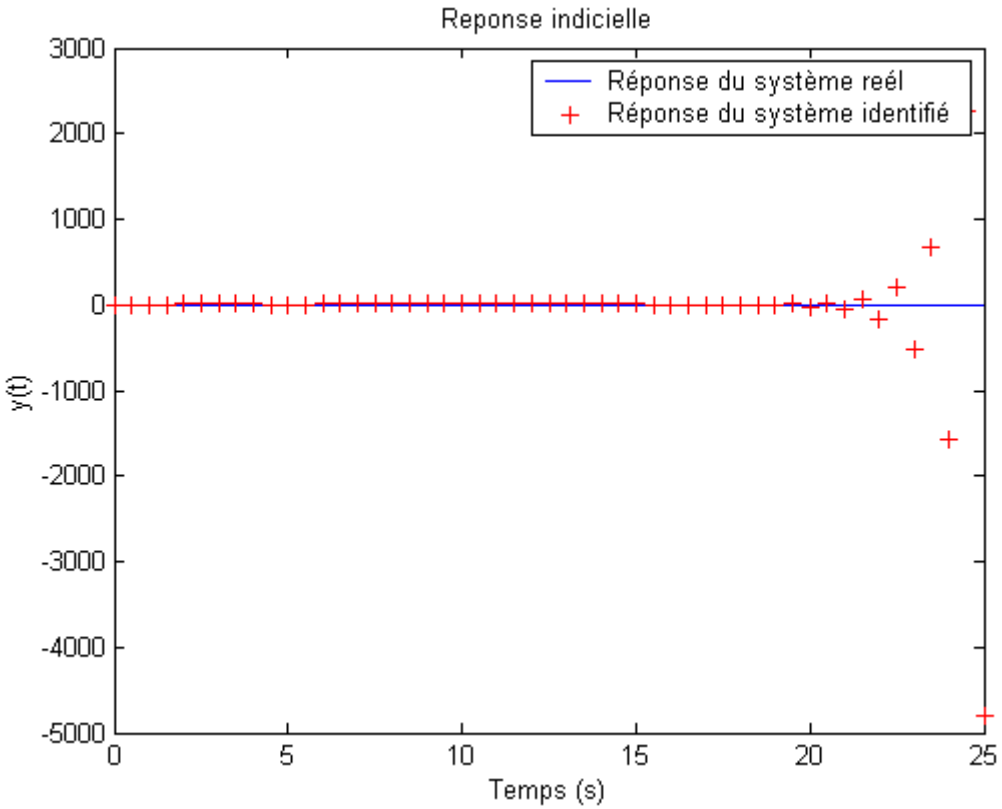

Les résultats sont encore pires lorsque la mesure n'est plus idéale :



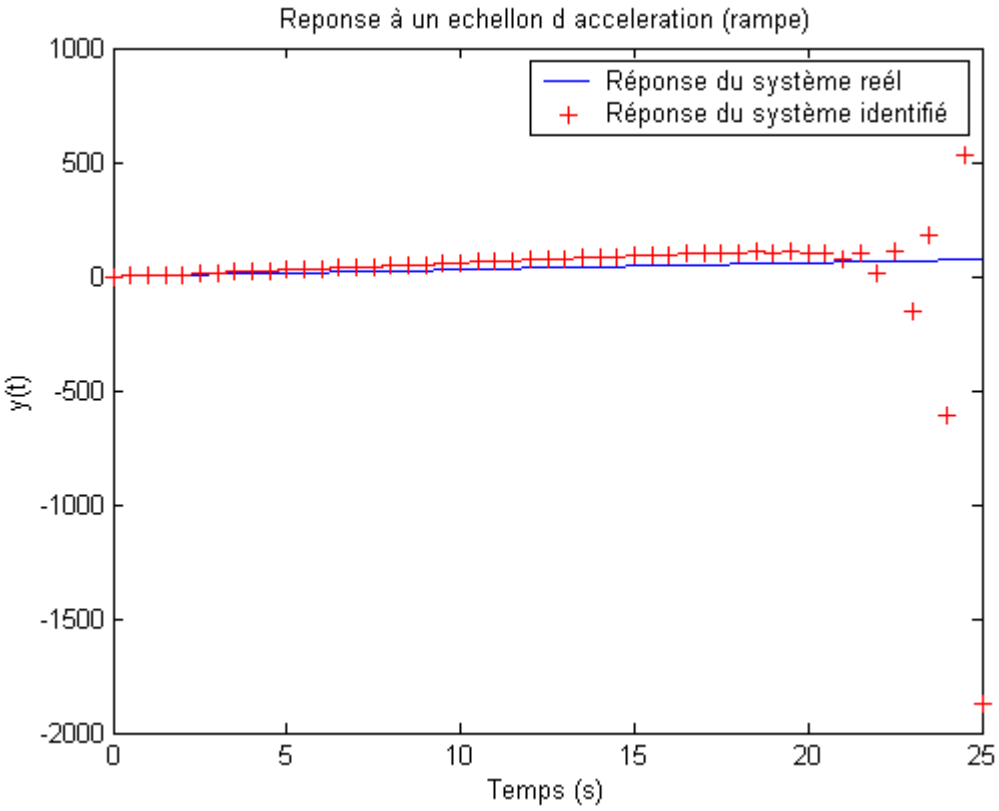
Réponses impulsionnelles :




Réponses indicielles :



Réponses à une rampe :




Cette méthode d'identification n'est donc pas bonne. Nous allons nous orienter vers une autre voie.

# 2ième essai

[Dossier MATLAB6]

## 1. Identification :

Pour ce deuxième essai, nous allons approximer le système par un FIR (filtre à réponse impulsionnelle finie). Cela implique que la mesure de h(t) devra être suffisamment longue pour que h(t) tende vers 0. Comme H(z) = B(z) / A(z), il suffit de calculer la transformée en Z de h, puis d'en déduire A et B.

Donc, en posant $h_i = h(t = Te * i)$, on a :

$$H(z) = h_0 + h_1 z^{-1} + h_1 z^{-2} + ... + h_n z^{-n}$$

Or, on a H = B / A avec :

$$A = a_0 + a_1 z + a_2 z^2 + ... + a_{n-1} z^{n-1} + z^n$$

$$B = b_0 + b_1 z + b_2 z^2 + ... + b_m z^m$$

On en déduit :

$$B = h_n + h_{n-1} z + h_{n-2} z^2 + ... + h_0 z^n$$

$$A = z^n$$

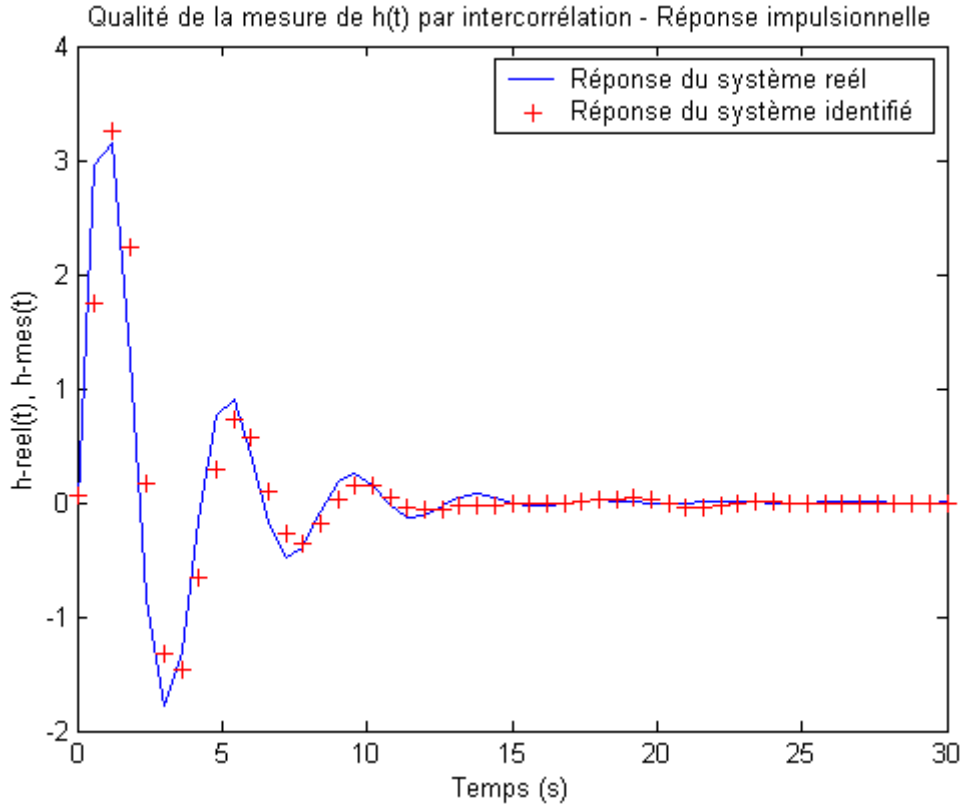
## 2. Tests avec MATLAB :

Les fichiers concernant cet essai se trouvent dans le dossier MATLAB6. Le script à exécuter est init.m.

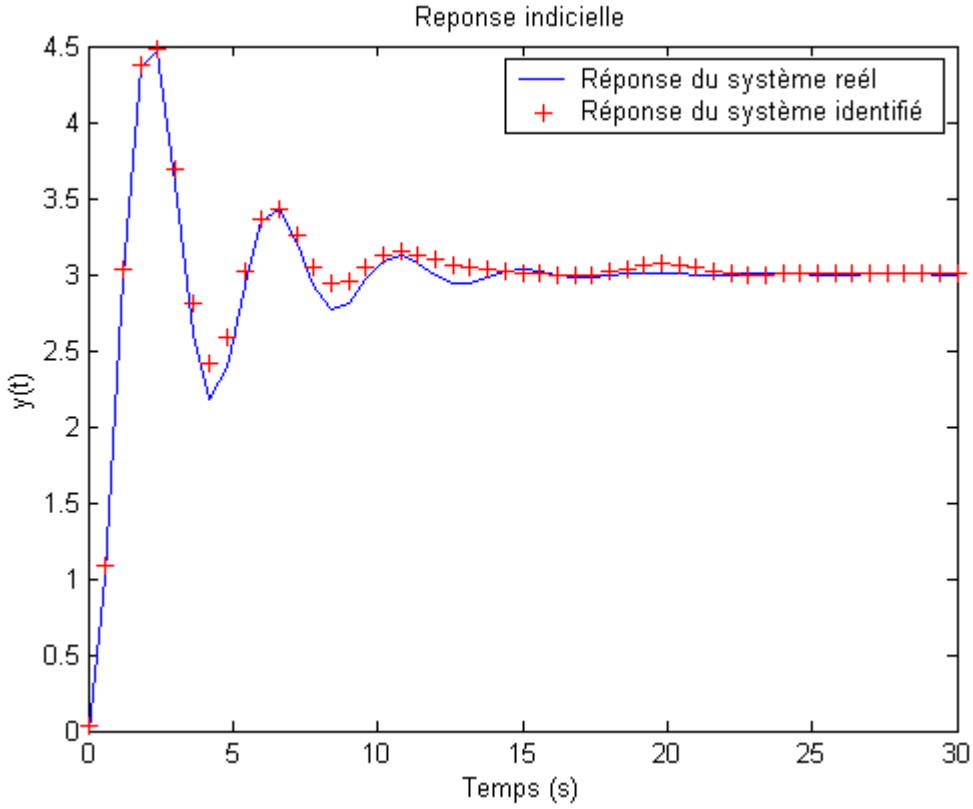
Pour tester la méthode d'identification, nous comparons les réponses impuslionnelles, indicielles et à une rampe du système réel et du système identifié.

Les résultats sont satisfaisants :

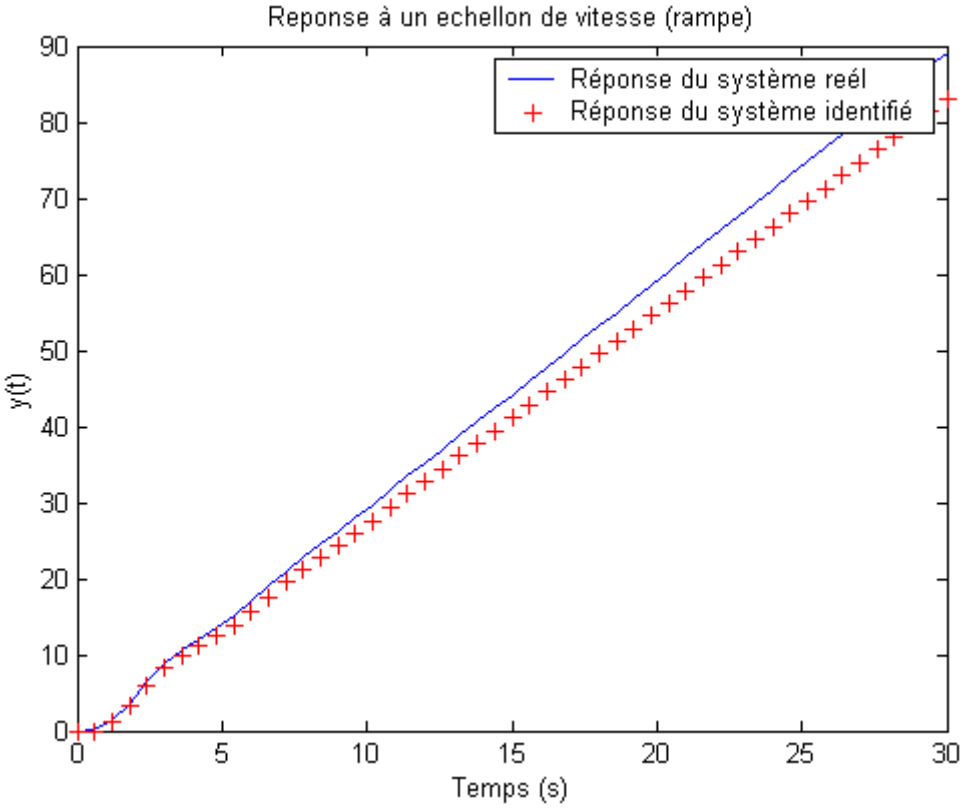
Qualité de la mesure :




Réponses indicielles :



Réponses à une rampe :




## 3. Synthèse :

Nous allons alors essayer de mettre en œuvre la synthèse.

Pour factoriser B sous la forme B = Bs * Bi, nous calculons les racines de B (c'est à dire les zéros du système). Ensuite, il suffit de regrouper ces racines dans deux tableaux, l'un contenant les zéros compensables et l'autre les zéros non compensables. On en déduit alors les coefficients de Bs et de Bi.

Voici le code MATLAB correspondant :

```
r_B_mes = roots(B_mes);
r_Bi = [];
r_Bs = [];
for i = 1:size(r_B_mes,1),
    if abs(r_B_mes(i)) < r_domaine
        r_Bs = [r_Bs; r_B_mes(i)];
    else
        % Zéro ne pouvant être compensé :
        r_Bi = [r_Bi; r_B_mes(i)];
    end
end
Bi = poly(r_Bi) .* B_mes(1);
Bs = poly(r_Bs);
```

Pour permettre la synthèse du régulateur, nous devons introduire dans la suite des calculs un polynôme $A_0$, de degré suffisamment élevé pour des raisons de causalité. Il faut que :

$d° A0 \geq 2 * d° A - d° Am - d° Bs + i - 1$

Nous choisissons $A0 = z^{(2 * d° A - d° Am - d° Bs + i - 1)}$

Le calcul de Bm' tel que Bm = Bm' * Bi se fait par identification, ce qui revient à résoudre un système sur conditionné PQR.

Prenons un exemple :

$Bi = b'_0 + b'_1 z + b'_2 z^2$
$Bm = d_0 + d_1 z + d_2 z^2 + d_3 z^3 + d_4 z^4 + d_5 z^5 + d_6 z^6$

Comme Bm = Bm' Bi, on a d° Bm' = d° Bm - d° Bi alors :

$Bm' = b_0 + b_1 z + b_2 z^2 + b_3 z^3 + b_4 z^4$

Ecrivons l'équation à résoudre :

$(b_0 + b_1 z + b_2 z^2 + b_3 z^3 + b_4 z^4) (b'_0 + b'_1 z + b'_2 z^2)$
$= d_0 + d_1 z + d_2 z^2 + d_3 z^3 + d_4 z^4 + d_5 z^5 + d_6 z^6$

Développons-la :

$b_0 b'_2 \mathbf{z^2}$ $+ b_1 b'_2 \mathbf{z^3}$ $+ b_2 b'_2 \mathbf{z^4}$ $+ b_3 b'_2 \mathbf{z^5}$ $+ b_4 b'_2 \mathbf{z^6}$
$+ b_0 b'_1 \mathbf{z^1}$ $+ b_1 b'_1 \mathbf{z^2}$ $+ b_2 b'_1 \mathbf{z^3}$ $+ b_3 b'_1 \mathbf{z^4}$ $+ b_4 b'_1 \mathbf{z^5}$
$+ b_0 b'_0 \mathbf{z^0}$ $+ b_1 b'_0 \mathbf{z^1}$ $+ b_2 b'_0 \mathbf{z^2}$ $+ b_3 b'_0 \mathbf{z^3}$ $+ b_4 b'_0 \mathbf{z^4}$
=
$d_0 \mathbf{z^0} + d_1 \mathbf{z^1} + d_2 \mathbf{z^2} + d_3 \mathbf{z^3} + d_4 \mathbf{z^4} + d_5 \mathbf{z^5} + d_6 \mathbf{z^6}$

Nous pouvons maintenant écrire le système matriciel PQR :

| P | = | | Q | | | | * | R | + | E |
|---|---|---|---|---|---|---|---|---|---|---|
| $d_0$ | | $b'_0$ | 0 | 0 | 0 | 0 | | $b_0$ | | $\varepsilon_0$ |
| $d_1$ | | $b'_1$ | $b'_0$ | 0 | 0 | 0 | | $b_1$ | | $\varepsilon_1$ |
| $d_2$ | | $b'_2$ | $b'_1$ | $b'_0$ | 0 | 0 | | $b_2$ | | $\varepsilon_2$ |
| $d_3$ | = | 0 | $b'_2$ | $b'_1$ | $b'_0$ | 0 | * | $b_3$ | + | $\varepsilon_3$ |
| $d_4$ | | 0 | 0 | $b'_2$ | $b'_1$ | $b'_0$ | | $b_4$ | | $\varepsilon_4$ |
| $d_5$ | | 0 | 0 | 0 | $b'_2$ | $b'_1$ | | | | $\varepsilon_5$ |
| $d_6$ | | 0 | 0 | 0 | 0 | $b'_2$ | | | | $\varepsilon_6$ |

Ici on voit nettement apparaître la convolution entre les coefficients des polynômes.

Résolution :
$R = (Q^T . Q)^{-1} . Q^T . P$
$E = P - Q . R$

Voici le code MATLAB correspondant :

```
% CALCUL DE Bmm TQ Bm # Bmm * Bi
P = zeros(degre(Bm)+1, 1);
Q = zeros(degre(Bm)+1, degre(Bm)+1 - degre(Bi));
for i = 1:size(P,1),
    P(i,1) = Bm(1, degre(Bm)+2-i);
end
for j = 1:size(Q,2),
    for i = j:j+degre(Bi),
        Q(i,j) = Bi(1, degre(Bi)+1-i+j);
    end
end
R = (Q'*Q)^-1*Q'*P;
E = P-Q*R;
%
Bmm = zeros(1, size(R,1));
for i = 1:size(Bmm,2),
    Bmm(1, i) = R(size(Bmm,2)+1-i, 1);
end
```

Puis il faut résoudre l'identité de Bézou :

$(z - 1)^i$ A R' + Bi S = A0 Am

Avec :

d° R' = d° A0 + d° Am - d°A - i
d° S ≤ d° A + i - 1
Le coefficient du terme de plus haut degré de R' doit valoir 1.

On choisit d° S = d° A + i - 1

Là aussi, la résolution de l'identité de Bézou sera mise en œuvre avec un système PQR.

Posons C = $(z - 1)^i$ A et P = A0 Am. L'équation de Bézou s'écrit alors :

C * R' + Bi * S = P

On reconnaît une forme PQR où Q contient C et Bi, et où R contient R' et Bi.

Prenons un exemple avec d°A = 5, d° Bi = 3, i = 2

On a :

d° R' = d° Bi - 1 = 2
d° S = d° A + i - 1 = 6
d° P = d° Bi + d° A + i -1 = 9
d° C = d° A + i = 7

$C = c_7 z^7 + c_6 z^6 + c_5 z^5 + c_4 z^4 + c_3 z^3 + c_2 z^2 + c_1 z + c_0$ ($c_7 = 1$)
$R' = z^2 + r_1 z + r_0$ (Le terme de plus haut degré de R' doit être égal à 1)
$Bi = b_3 z^3 + b_2 z^2 + b_1 z + b_0$
$S = s_6 z^6 + s_5 z^5 + s_4 z^4 + s_3 z^3 + s_2 z^2 + s_1 z + s_0$
$P = p_9 z^9 + p_8 z^8 + p_7 z^7 + p_6 z^6 + p_5 z^5 + p_4 z^4 + p_3 z^3 + p_2 z^2 + p_1 z + p_0$ ($p_9 = 1$, $p_{i<9} = 0$)

Développons l'équation matricielle C * R' + Bi * S = P :

$$\mathbf{c_7 z^9 + c_6 z^8 + c_5 z^7 + c_4 z^6 + c_3 z^5 + c_2 z^4 + c_1 z^3 + c_0 z^2}$$
$$+ c_7 r_1 z^8 + c_6 r_1 z^7 + c_5 r_1 z^6 + c_4 r_1 z^5 + c_3 r_1 z^4 + c_2 r_1 z^3 + c_1 r_1 z^2 + c_0 r_1 z$$
$$+ c_7 r_0 z^7 + c_6 r_0 z^6 + c_5 r_0 z^5 + c_4 r_0 z^4 + c_3 r_0 z^3 + c_2 r_0 z^2 + c_1 r_0 z + c_0 r_0$$
$$+$$
$$b_3 s_6 z^9 + b_2 s_6 z^8 + b_1 s_6 z^7 + b_0 s_6 z^6$$
$$+ b_3 s_5 z^8 + b_2 s_5 z^7 + b_1 s_5 z^6 + b_0 s_5 z^5$$
$$+ b_3 s_4 z^7 + b_2 s_4 z^6 + b_1 s_4 z^5 + b_0 s_4 z^4$$
$$+ b_3 s_3 z^6 + b_2 s_3 z^5 + b_1 s_3 z^4 + b_0 s_3 z^3$$
$$+ b_3 s_2 z^5 + b_2 s_2 z^4 + b_1 s_2 z^3 + b_0 s_2 z^2$$
$$+ b_3 s_1 z^4 + b_2 s_1 z^3 + b_1 s_1 z^2 + b_0 s_1 z$$
$$+ b_3 s_0 z^3 + b_2 s_0 z^2 + b_1 s_0 z + b_0 s_0$$
$$=$$
$$p_9 z^9 + p_8 z^8 + p_7 z^7 + p_6 z^6 + p_5 z^5 + p_4 z^4 + p_3 z^3 + p_2 z^2 + p_1 z + p_0$$

Ce système est équivalent à :

$$c_7 r_1 z^8 + c_6 r_1 z^7 + c_5 r_1 z^6 + c_4 r_1 z^5 + c_3 r_1 z^4 + c_2 r_1 z^3 + c_1 r_1 z^2 + c_0 r_1 z$$
$$+ c_7 r_0 z^7 + c_6 r_0 z^6 + c_5 r_0 z^5 + c_4 r_0 z^4 + c_3 r_0 z^3 + c_2 r_0 z^2 + c_1 r_0 z + c_0 r_0$$
$$+$$
$$b_3 s_6 z^9 + b_2 s_6 z^8 + b_1 s_6 z^7 + b_0 s_6 z^6$$
$$+ b_3 s_5 z^8 + b_2 s_5 z^7 + b_1 s_5 z^6 + b_0 s_5 z^5$$
$$+ b_3 s_4 z^7 + b_2 s_4 z^6 + b_1 s_4 z^5 + b_0 s_4 z^4$$
$$+ b_3 s_3 z^6 + b_2 s_3 z^5 + b_1 s_3 z^4 + b_0 s_3 z^3$$
$$+ b_3 s_2 z^5 + b_2 s_2 z^4 + b_1 s_2 z^3 + b_0 s_2 z^2$$
$$+ b_3 s_1 z^4 + b_2 s_1 z^3 + b_1 s_1 z^2 + b_0 s_1 z$$
$$+ b_3 s_0 z^3 + b_2 s_0 z^2 + b_1 s_0 z + b_0 s_0$$
$$=$$
$$(p_9 - \mathbf{c_7}) z^9 + (p_8 - \mathbf{c_6}) z^8 + (p_7 - \mathbf{c_5}) z^7 + (p_6 - \mathbf{c_4}) z^6 + (p_5 - \mathbf{c_3}) z^5 + (p_4 - \mathbf{c_2}) z^4 + (p_3 - \mathbf{c_1}) z^3 + (p_2 - \mathbf{c_0}) z^2 + p_1 z + p_0$$

Nous pouvons maintenant écrire le système matriciel PQR :

| P | = | | | | | Q | | | | | * | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p_0$ | | $d_0$ | 0 | 0 | 0 | 0 | 0 | 0 | $c_0$ | 0 | | $s_0$ |
| $p_1$ | | $d_1$ | $d_0$ | 0 | 0 | 0 | 0 | 0 | $c_1$ | $c_0$ | | $s_1$ |
| $p_2 - c_0$ | | $d_2$ | $d_1$ | $d_0$ | 0 | 0 | 0 | 0 | $c_2$ | $c_1$ | | $s_2$ |
| $p_3 - c_1$ | | $d_3$ | $d_2$ | $d_1$ | $d_0$ | 0 | 0 | 0 | $c_3$ | $c_2$ | | $s_3$ |
| $p_4 - c_2$ | | 0 | $d_3$ | $d_2$ | $d_1$ | $d_0$ | 0 | 0 | $c_4$ | $c_3$ | * | $s_4$ |
| $p_5 - c_3$ | = | 0 | 0 | $d_3$ | $d_2$ | $d_1$ | $d_0$ | 0 | $c_5$ | $c_4$ | | $s_5$ |
| $p_6 - c_4$ | | 0 | 0 | 0 | $d_3$ | $d_2$ | $d_1$ | $d_0$ | $c_6$ | $c_5$ | | $s_6$ |
| $p_7 - c_5$ | | 0 | 0 | 0 | 0 | $d_3$ | $d_2$ | $d_1$ | $c_7$ | $c_6$ | | $r_0$ |
| $p_8 - c_6$ | | 0 | 0 | 0 | 0 | 0 | $d_3$ | $d_2$ | 0 | $c_7$ | | $r_1$ |
| $p_9 - c_7$ | | 0 | 0 | 0 | 0 | 0 | 0 | $d_3$ | 0 | 0 | | |

Résolution : $R = (Q^T . Q)^{-1} . Q^T . P$ (l'erreur sur chaque élément de P est $E = P - Q . R$)

Voici le code MATLAB correspondant :

```
% RESOLUTION DE L'IDENTITE DE BEZOUT :
% (z-1)^n_i A_mes RR + Bi S = A0 Am
% C RR + D S = P
% On construit C, D et P :
% C =(z-1)^n_i A_mes :
C = A_mes;
for i=1:n_i,
    C = conv(C, [1 -1]);
end
D = Bi;
PP = conv(A0, Am);

% On résout C RR + D S = P sous la forme P = Q R + E
% Le terme de plus haut degré de RR est égal à 1
% Remplissage de Q :
Q = zeros(degre(Bi) + n_mes + n_i, degre(Bi) + n_mes + n_i - 1);
for j = 1:n_mes+n_i,
    for i = j:j+degre(Bi),
        Q(i, j) = D(degre(Bi) + 1 - i + j);
    end
end
for j = 1:degre(Bi)-1,
    for i = j:j+n_mes+n_i,
        Q(i, j+n_mes+n_i) = C(n_mes+n_i+1-i+j);
    end
end
% Remplissage  de P :
P = zeros(degre(Bi) + n_mes + n_i, 1);
for i=1:degre(Bi)+n_mes+n_i,
    P(i) = PP(degre(Bi)+n_mes+n_i+1-i);
end
for i=degre(Bi):degre(Bi)+n_mes+n_i,
    P(i) = P(i) - C(n_mes+n_i+1-i+degre(Bi));
end

R = (Q'*Q)^-1*Q'*P;
E = P-Q*R;
```

Finalement on calcule les polynômes R et T :

$R = Bs\ (z - 1)^i\ R'$

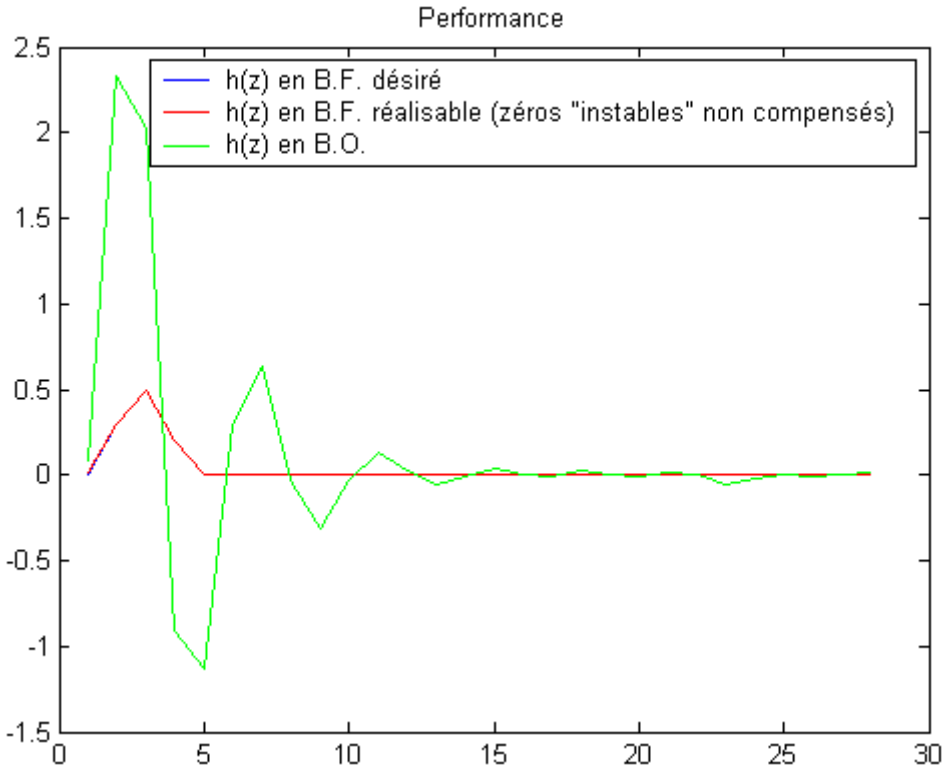
$T = Bm'\ A0$

## 4. Tests avec MATLAB (suite) :

Le programme précédant teste la synthèse en plus de l'identification. Etudions les résultats de deux tests (juste la période d'échantillonnage diffère) :

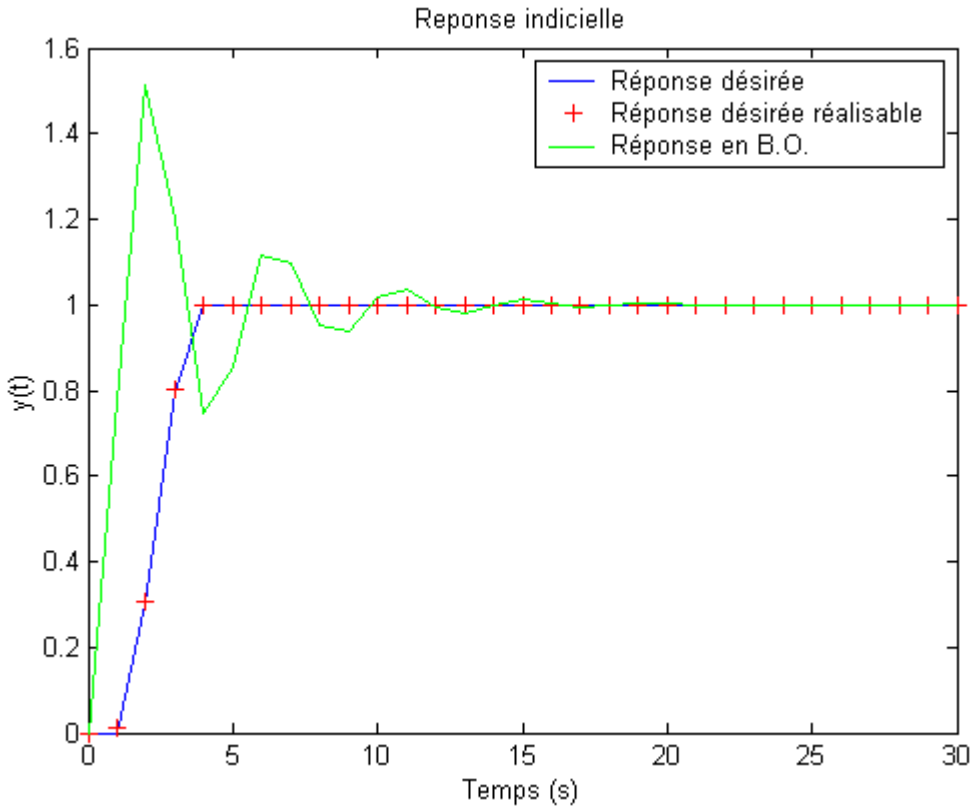
Test avec une période d'échantillonnage de 1 seconde :

Voici d'abord la réponse impulsionnelle du système, la réponse impulsionnelle désirée (modèle Bm / Am) et la réponse impulsionnelle réalisable (Bi * Bm' / Am).

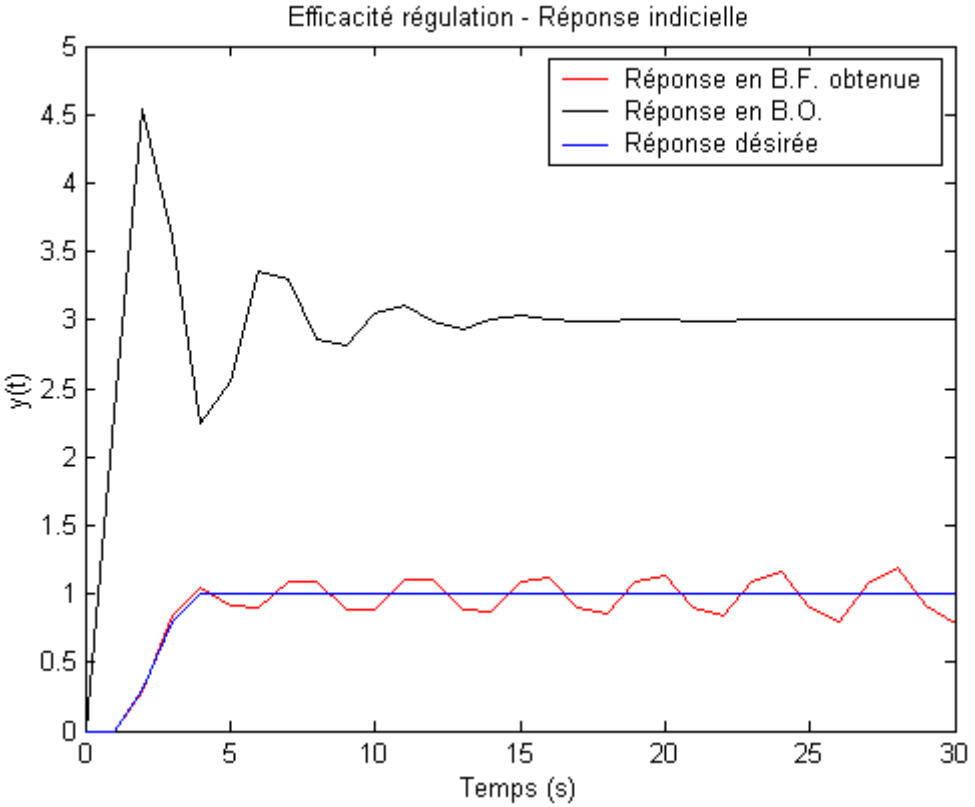



On voit que la réponse réalisable est très proche de celle désirée.

Même constatation ici :

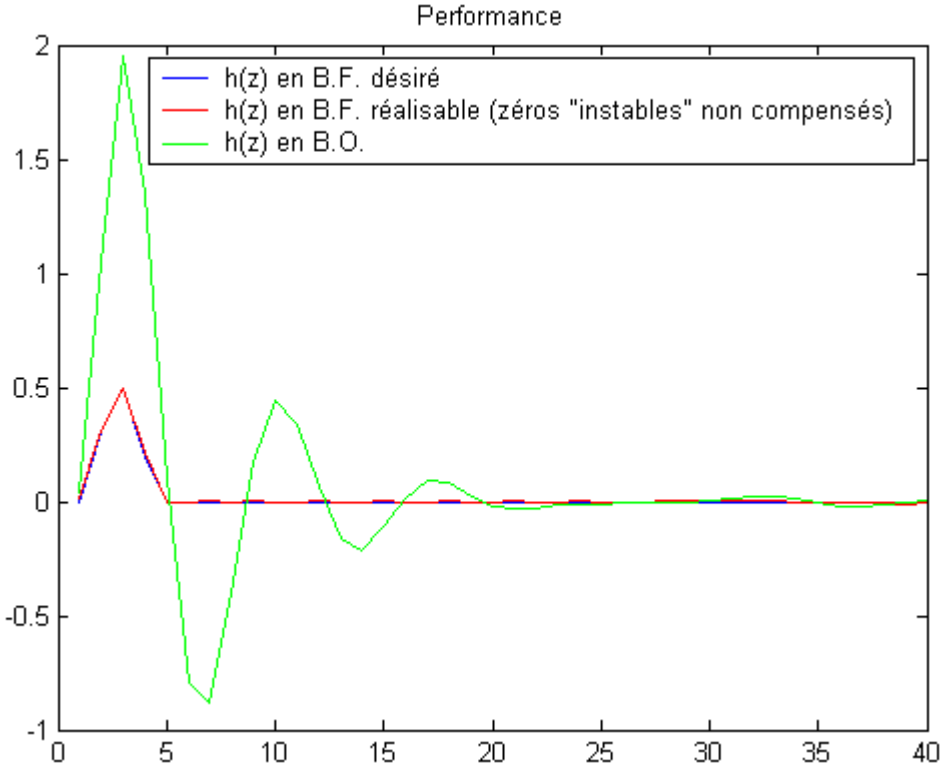


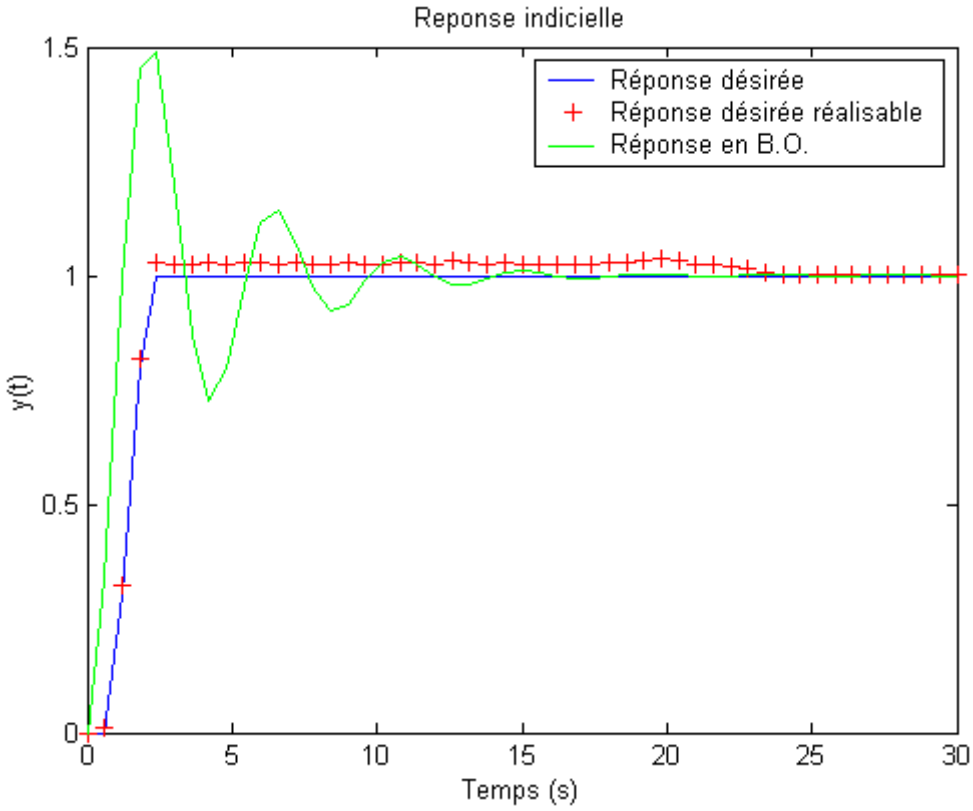

Mais le résultat est décevant :



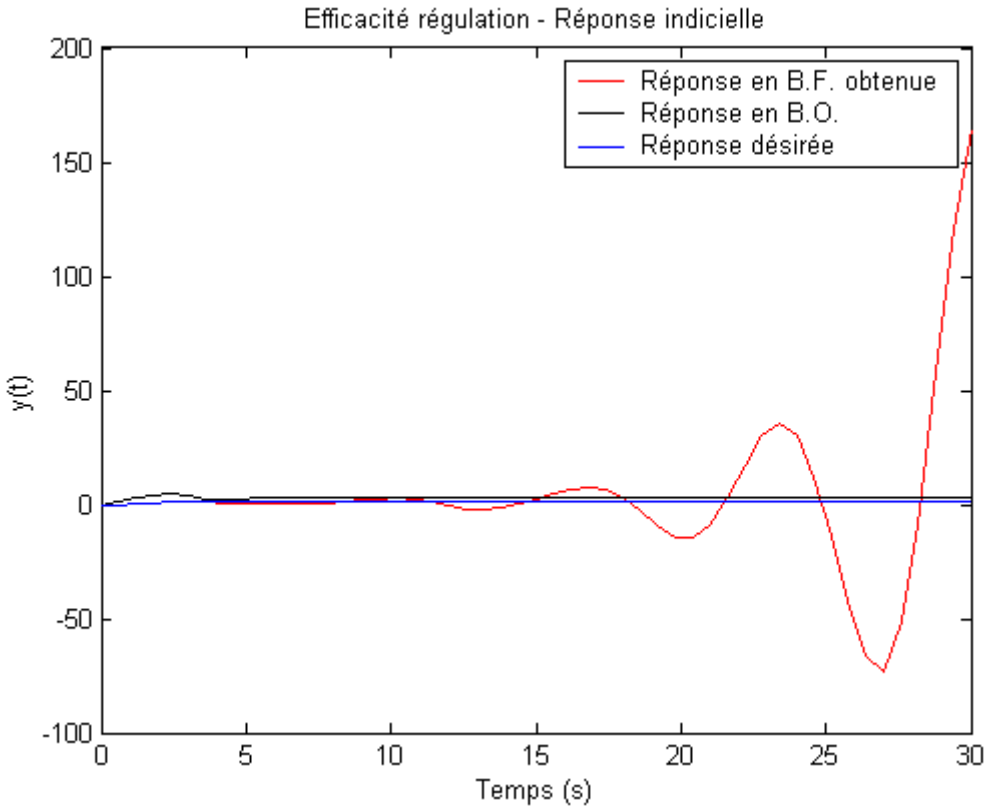
Test avec une période d'échantillonnage de 0,6 seconde :

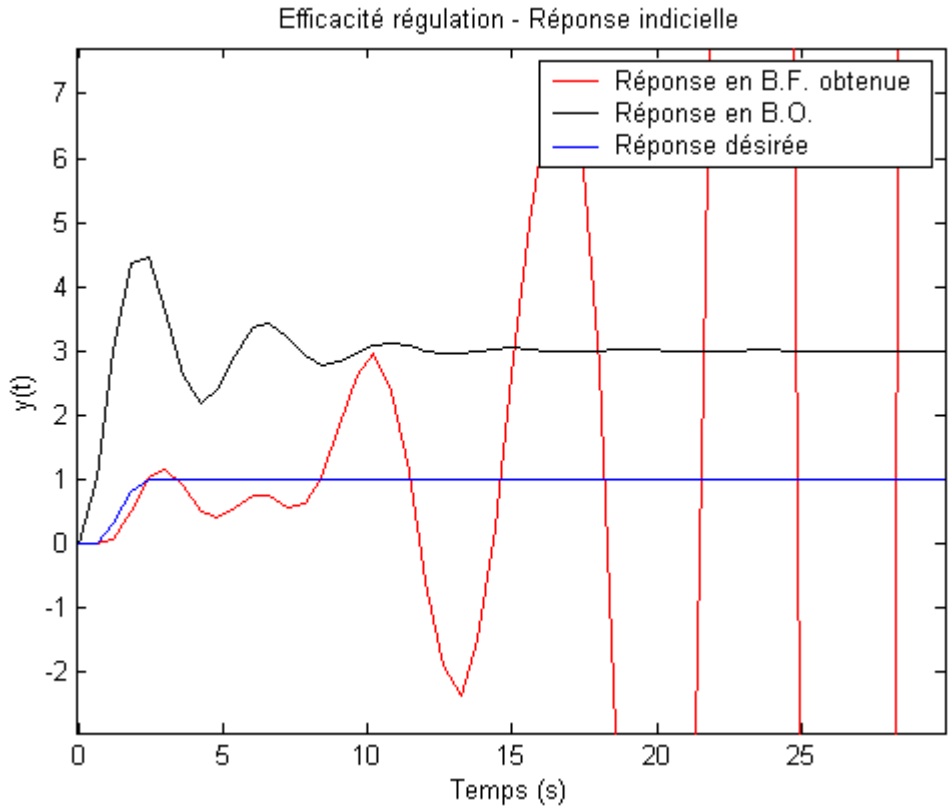
On remarque que la réponse réalisable présente de légères fluctuations autour de la réponse désirée :



Ici, le résultat est très mauvais, la régulation est instable :



## 5. Modification du programme :

[Dossier MATLAB7]

Quand on reprend les calculs, on se rend compte que le système PQR permettant de résoudre l'équation de Bézou est sur conditionné. En fait, si on choisit un degré plus élevé pour A0 le système n'est plus sur conditionné, et de ce fait le vecteur d'erreur E est nul.

Reprenons les calculs de l'exemple précédent avec cette fois-ci : $A0 = z^{(2 * d° A - d° Am - d° Bs + i )}$ au lieu de $z^{(2 * d° A - d° Am - d° Bs + i - 1)}$

d°A = 5, d° Bi = 3, i = 2

On a :

d° R' = d° Bi = 3 (le degré de R' a augmenté du fait de A0)
d° S = d° A + i - 1 = 6
d° P = d° Bi + d° A + i = 10 (le degré de P a augmenté du fait de A0)
d° C = d° A + i = 7

$C = c_7 z^7 + c_6 z^6 + c_5 z^5 + c_4 z^4 + c_3 z^3 + c_2 z^2 + c_1 z + c_0$ ($c_7 = 1$)
$R' = z^2 + r_1 z + r_0$ (Le terme de plus haut degré de R' doit être égal à 1)
$Bi = b_3 z^3 + b_2 z^2 + b_1 z + b_0$
$S = s_6 z^6 + s_5 z^5 + s_4 z^4 + s_3 z^3 + s_2 z^2 + s_1 z + s_0$
$P = p_9 z^9 + p_8 z^8 + p_7 z^7 + p_6 z^6 + p_5 z^5 + p_4 z^4 + p_3 z^3 + p_2 z^2 + p_1 z + p_0$ ($p_{10} = 1$, $p_{i<10} = 0$)

Développons l'équation matricielle C * R' + Bi * S = P :

$$\mathbf{c_7 z^{10} + c_6 z^9 + c_5 z^8 + c_4 z^7 + c_3 z^6 + c_2 z^5 + c_1 z^4 + c_0 z^3}$$
$$+ c_7 r_2 z^9 + c_6 r_2 z^8 + c_5 r_2 z^7 + c_4 r_2 z^6 + c_3 r_2 z^5 + c_2 r_2 z^4 + c_1 r_2 z^3 + c_0 r_1 z^2$$
$$+ c_7 r_1 z^8 + c_6 r_1 z^7 + c_5 r_1 z^6 + c_4 r_1 z^5 + c_3 r_1 z^4 + c_2 r_1 z^3 + c_1 r_1 z^2 + c_0 r_1 z$$
$$+ c_7 r_0 z^7 + c_6 r_0 z^6 + c_5 r_0 z^5 + c_4 r_0 z^4 + c_3 r_0 z^3 + c_2 r_0 z^2 + c_1 r_0 z + c_0 r_0$$
$$+$$
$$b_3 s_6 z^9 + b_2 s_6 z^8 + b_1 s_6 z^7 + b_0 s_6 z^6$$
$$+ b_3 s_5 z^8 + b_2 s_5 z^7 + b_1 s_5 z^6 + b_0 s_5 z^5$$
$$+ b_3 s_4 z^7 + b_2 s_4 z^6 + b_1 s_4 z^5 + b_0 s_4 z^4$$
$$+ b_3 s_3 z^6 + b_2 s_3 z^5 + b_1 s_3 z^4 + b_0 s_3 z^3$$
$$+ b_3 s_2 z^5 + b_2 s_2 z^4 + b_1 s_2 z^3 + b_0 s_2 z^2$$
$$+ b_3 s_1 z^4 + b_2 s_1 z^3 + b_1 s_1 z^2 + b_0 s_1 z$$
$$+ b_3 s_0 z^3 + b_2 s_0 z^2 + b_1 s_0 z + b_0 s_0$$
$$=$$
$$p_{10} z^{10} + p_9 z^9 + p_8 z^8 + p_7 z^7 + p_6 z^6 + p_5 z^5 + p_4 z^4 + p_3 z^3 + p_2 z^2 + p_1 z + p_0$$

Ce système est équivalent à :

$$
\begin{aligned}
&c_7 r_2 z^9 + c_6 r_2 z^8 + c_5 r_2 z^7 + c_4 r_2 z^6 + c_3 r_2 z^5 + c_2 r_2 z^4 + c_1 r_2 z^3 + c_0 r_1 z^2 \\
&+ c_7 r_1 z^8 + c_6 r_1 z^7 + c_5 r_1 z^6 + c_4 r_1 z^5 + c_3 r_1 z^4 + c_2 r_1 z^3 + c_1 r_1 z^2 + c_0 r_1 z \\
&+ c_7 r_0 z^7 + c_6 r_0 z^6 + c_5 r_0 z^5 + c_4 r_0 z^4 + c_3 r_0 z^3 + c_2 r_0 z^2 + c_1 r_0 z + c_0 r_0 \\
&+ \\
&b_3 s_6 z^9 + b_2 s_6 z^8 + b_1 s_6 z^7 + b_0 s_6 z^6 \\
&+ b_3 s_5 z^8 + b_2 s_5 z^7 + b_1 s_5 z^6 + b_0 s_5 z^5 \\
&+ b_3 s_4 z^7 + b_2 s_4 z^6 + b_1 s_4 z^5 + b_0 s_4 z^4 \\
&+ b_3 s_3 z^6 + b_2 s_3 z^5 + b_1 s_3 z^4 + b_0 s_3 z^3 \\
&+ b_3 s_2 z^5 + b_2 s_2 z^4 + b_1 s_2 z^3 + b_0 s_2 z^2 \\
&+ b_3 s_1 z^4 + b_2 s_1 z^3 + b_1 s_1 z^2 + b_0 s_1 z \\
&+ b_3 s_0 z^3 + b_2 s_0 z^2 + b_1 s_0 z + b_0 s_0 \\
&= \\
&(p_{10} - \mathbf{c_7}) z^{10} + (p_9 - \mathbf{c_6}) z^9 + (p_8 - \mathbf{c_5}) z^8 + (p_7 - \mathbf{c_4}) z^7 + (p_6 - \mathbf{c_3}) z^6 + (p_5 - \mathbf{c_2}) z^5 + (p_4 - \mathbf{c_1}) z^4 + \\
&(p_3 - \mathbf{c_0}) z^3 + p_2 z^2 + p_1 z + p_0
\end{aligned}
$$

Nous pouvons maintenant écrire le système matriciel PQR :

| P | | Q | | | | | | | | | | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p_0$ | | $d_0$ | 0 | 0 | 0 | 0 | 0 | 0 | $c_0$ | 0 | 0 | $s_0$ |
| $p_1$ | | $d_1$ | $d_0$ | 0 | 0 | 0 | 0 | 0 | $c_1$ | $c_0$ | 0 | $s_1$ |
| $p_2$ | | $d_2$ | $d_1$ | $d_0$ | 0 | 0 | 0 | 0 | $c_2$ | $c_1$ | $c_0$ | $s_2$ |
| $p_3 - c_0$ | | $d_3$ | $d_2$ | $d_1$ | $d_0$ | 0 | 0 | 0 | $c_3$ | $c_2$ | $c_1$ | $s_3$ |
| $p_4 - c_1$ | | 0 | $d_3$ | $d_2$ | $d_1$ | $d_0$ | 0 | 0 | $c_4$ | $c_3$ | $c_2$ | * $s_4$ |
| $p_5 - c_2$ | = | 0 | 0 | $d_3$ | $d_2$ | $d_1$ | $d_0$ | 0 | $c_5$ | $c_4$ | $c_3$ | $s_5$ |
| $p_6 - c_3$ | | 0 | 0 | 0 | $d_3$ | $d_2$ | $d_1$ | $d_0$ | $c_6$ | $c_5$ | $c_4$ | $s_6$ |
| $p_7 - c_4$ | | 0 | 0 | 0 | 0 | $d_3$ | $d_2$ | $d_1$ | $c_7$ | $c_6$ | $c_5$ | $r_0$ |
| $p_8 - c_5$ | | 0 | 0 | 0 | 0 | 0 | $d_3$ | $d_2$ | 0 | $c_7$ | $c_6$ | $r_1$ |
| $p_9 - c_6$ | | 0 | 0 | 0 | 0 | 0 | 0 | $d_3$ | 0 | 0 | $c_7$ | $r_2$ |
| $\mathbf{p_{10} - c_7}$ | | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |

Etant donné que p10 et c7 sont tous deux égal à 1, la dernière ligne de P et de Q peut être supprimé, ce qui aboutit à un système PQR non sur conditionné :

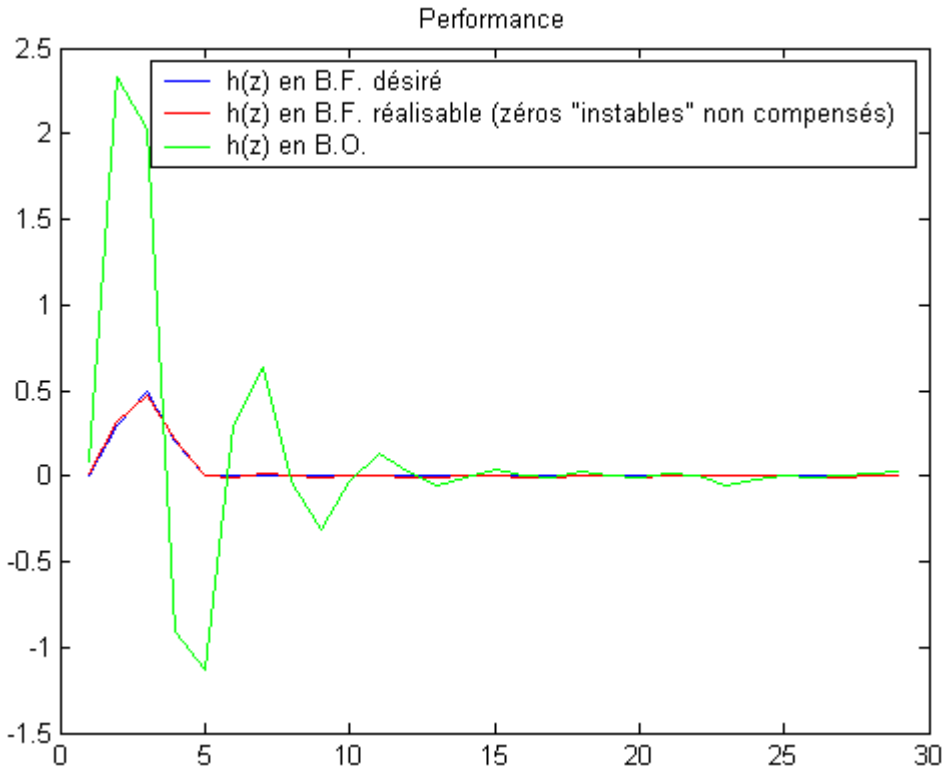
| P | | Q | | | | | | | | | | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p_0$ | | $d_0$ | 0 | 0 | 0 | 0 | 0 | 0 | $c_0$ | 0 | 0 | $s_0$ |
| $p_1$ | | $d_1$ | $d_0$ | 0 | 0 | 0 | 0 | 0 | $c_1$ | $c_0$ | 0 | $s_1$ |
| $p_2$ | | $d_2$ | $d_1$ | $d_0$ | 0 | 0 | 0 | 0 | $c_2$ | $c_1$ | $c_0$ | $s_2$ |
| $p_3 - c_0$ | | $d_3$ | $d_2$ | $d_1$ | $d_0$ | 0 | 0 | 0 | $c_3$ | $c_2$ | $c_1$ | $s_3$ |
| $p_4 - c_1$ | | 0 | $d_3$ | $d_2$ | $d_1$ | $d_0$ | 0 | 0 | $c_4$ | $c_3$ | $c_2$ | * $s_4$ |
| $p_5 - c_2$ | = | 0 | 0 | $d_3$ | $d_2$ | $d_1$ | $d_0$ | 0 | $c_5$ | $c_4$ | $c_3$ | $s_5$ |
| $p_6 - c_3$ | | 0 | 0 | 0 | $d_3$ | $d_2$ | $d_1$ | $d_0$ | $c_6$ | $c_5$ | $c_4$ | $s_6$ |
| $p_7 - c_4$ | | 0 | 0 | 0 | 0 | $d_3$ | $d_2$ | $d_1$ | $c_7$ | $c_6$ | $c_5$ | $r_0$ |
| $p_8 - c_5$ | | 0 | 0 | 0 | 0 | 0 | $d_3$ | $d_2$ | 0 | $c_7$ | $c_6$ | $r_1$ |
| $p_9 - c_6$ | | 0 | 0 | 0 | 0 | 0 | 0 | $d_3$ | 0 | 0 | $c_7$ | $r_2$ |

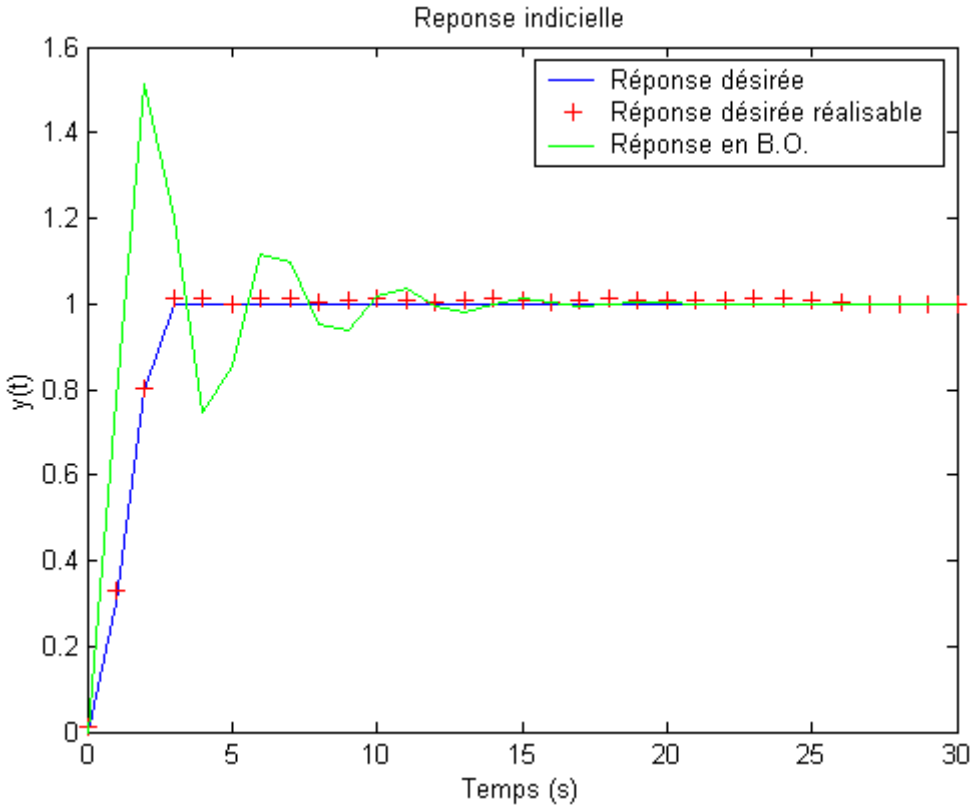
## 6. Tests avec MATLAB :

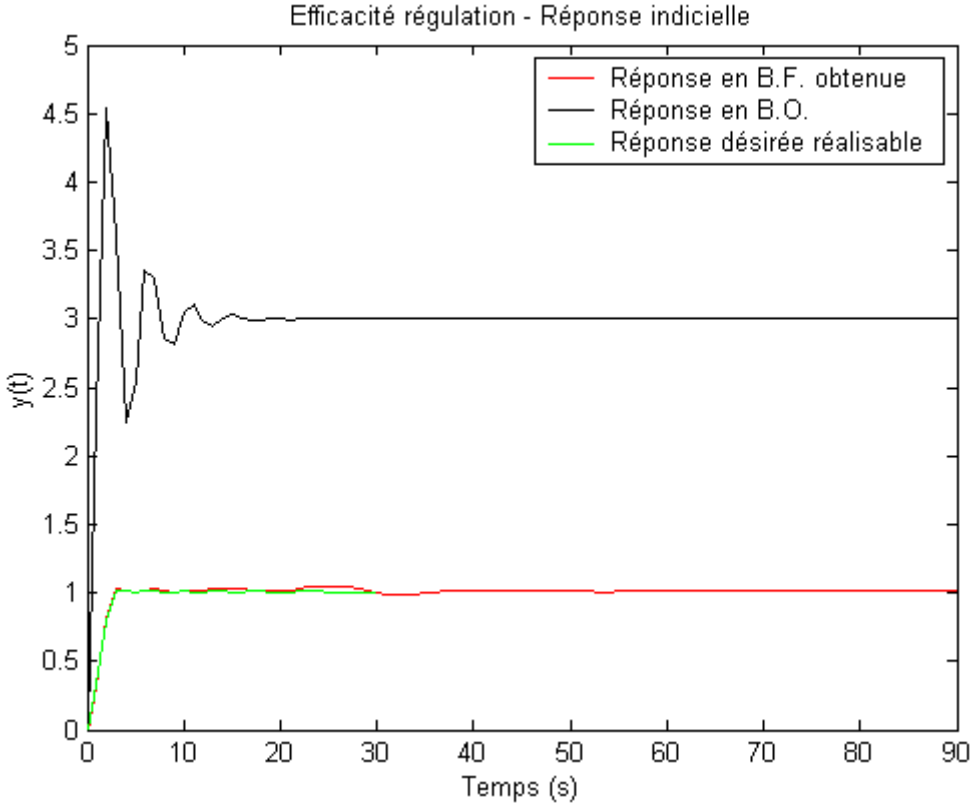
Les fichiers concernant cet essai se trouvent dans le dossier MATLAB7. Le script à exécuter est init.m.

Nous allons refaire les deux tests précédents avec le programme modifié :

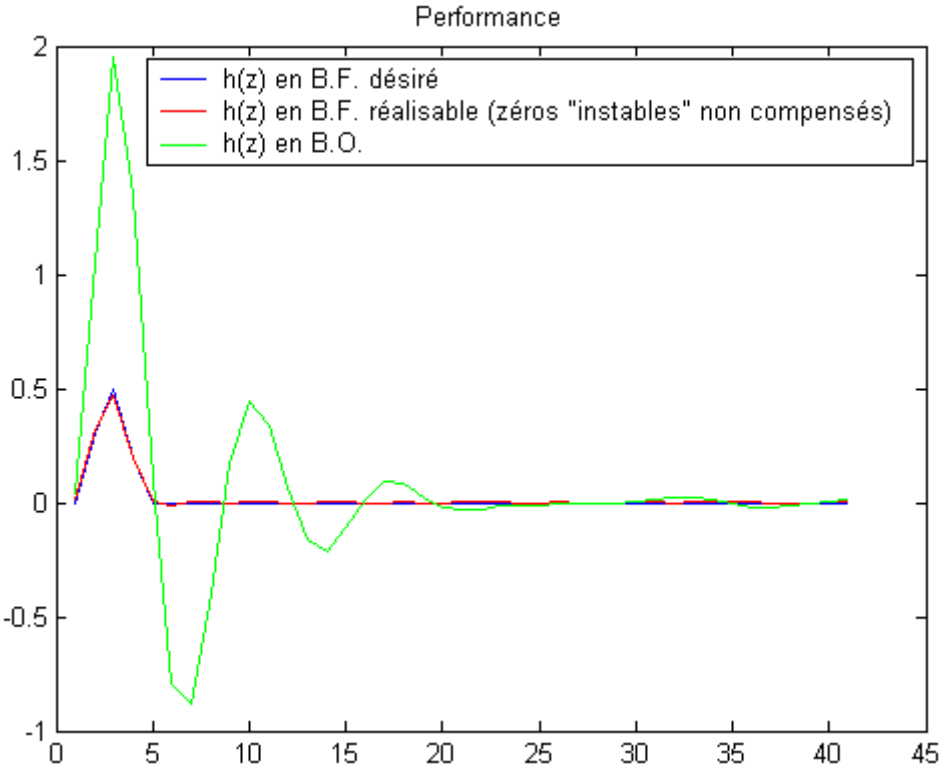
Test avec une période d'échantillonnage de 1 seconde :

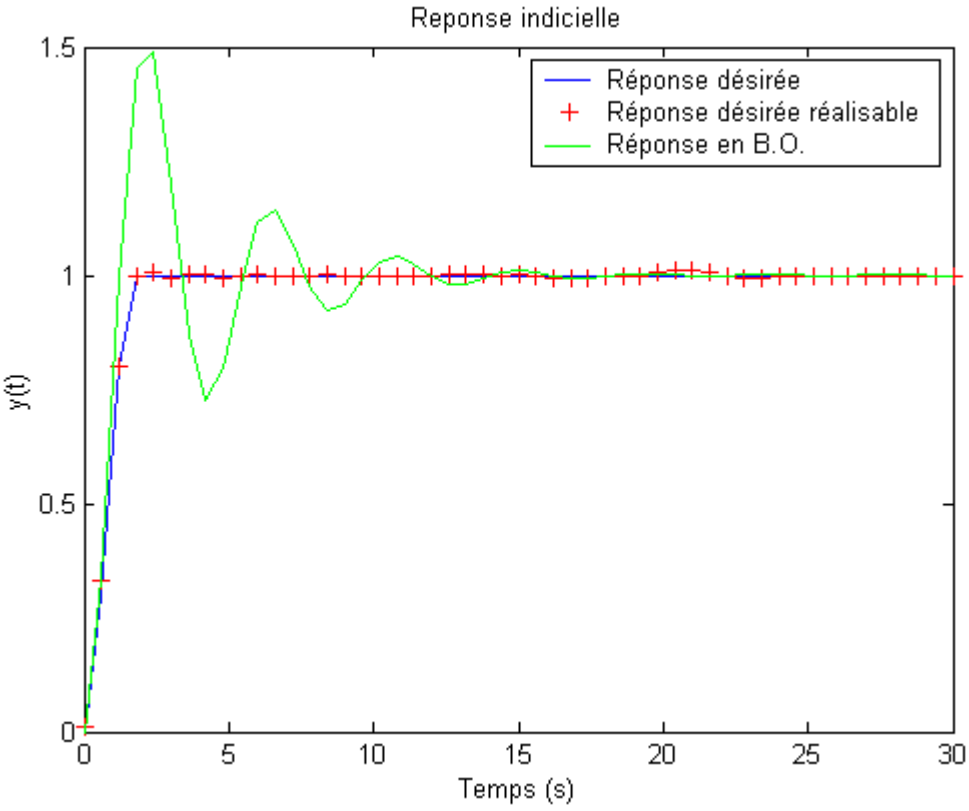


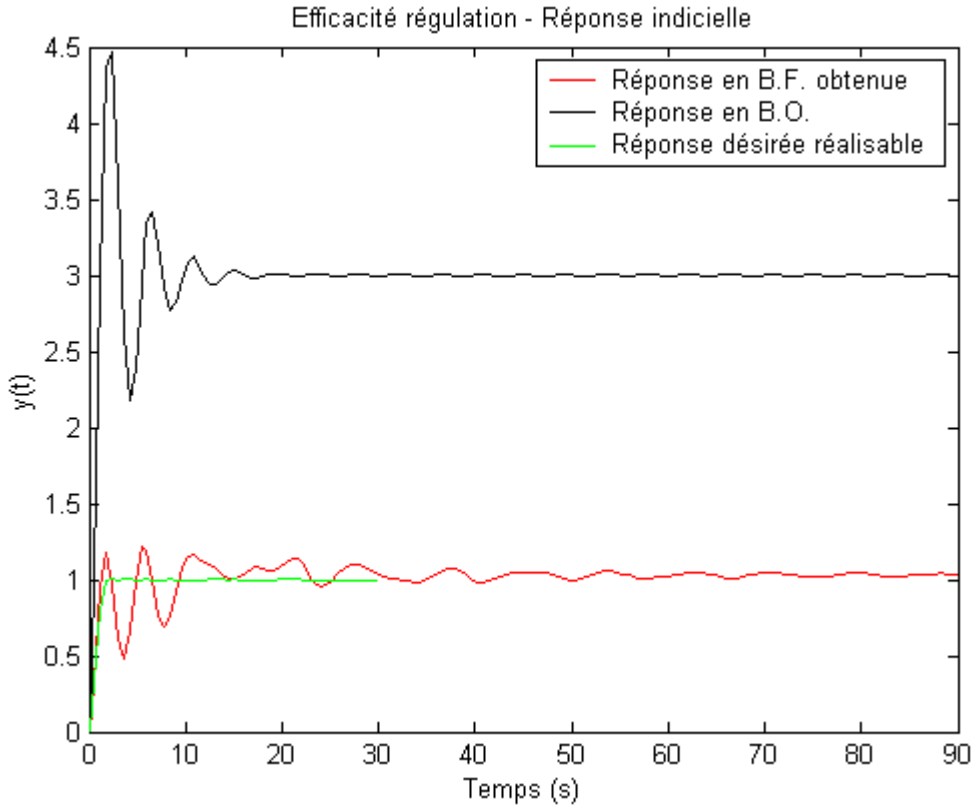

Reponse indicielle
Réponse désirée
Réponse désirée réalisable
Réponse en B.O.
y(t)
Temps (s)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
0
5
10
15
20
25
30


Efficacité régulation - Réponse indicielle
Réponse en B.F. obtenue
Réponse en B.O.
Réponse désirée réalisable
y(t)
Temps (s)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
10
20
30
40
50
60
70
80
90

Test avec une période d'échantillonnage de 0,6 seconde :






Les résultats sont meilleurs que dans le cas précédant, du fait de l'absence d'erreur liée à la résolution de l'équation de Bézou.

Mais ces résultats ne peuvent pas être considérés comme bons. Même si le régulateur est plus stable, on constate la présence d'ondulations de forme complexe (ordre élevé) qui se prolongent bien au delà du régime établit.

Nous avons amélioré l'implantation de la méthode du modèle en évitant de résoudre un système sur conditionné, mais en gardant le même principe pour l'identification, à savoir l'approximation du système par un FIR.

En fait le point commun entre les deux premiers essais (MATLAB5, MATLAB6 et MATLAB7) est d'obtenir des ordres important pour A et B après identification.

En réalité, les systèmes physiques rencontrés dans la nature sont d'ordre plus faible; par exemple un deuxième ordre (en continu) aura pour équivalent discret un système B/A avec B d'ordre 1 et A d'ordre 2. Dans les prochains essais nous allons donc essayer d'identifier le système avec une fonction de transfert B / A avec A et B de degré faible.

Cette façon de procéder va nous ramener à ce qui a été fait lors de l'étude en continu.

# 3ième essai

[Dossier MATLAB9]

## 1. Identification :

Nous allons essayer de retrouver A et B à partir de la mesure de la réponse impulsionnelle de la même façon que dans le premier essai, mais avec des degrés de A et de B faibles.

Posons n = d°A et m = d°B. Nous devons choisir des valeurs de n et de m faibles, et respectant la condition de causalité m ≤ n. Par exemple, n = 2 et m = 1, ce qui corresponds à l'équivalent discret d'un deuxième ordre.

Nous obtenons un système PQR fortement sur conditionné :

| P | = | Q | | | | | | | | * | R |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $y_0$ | | $u_{-m}$ | $u_{-m+1}$ | ... | $u_0$ | $y_{-n}$ | $y_{-n+1}$ | ... | $y_{-1}$ | | $b_0$ |
| $y_1$ | | $u_{-m+1}$ | $u_{-m+2}$ | ... | $u_1$ | $y_{-n+1}$ | $y_{-n+2}$ | ... | $y_0$ | | ... |
| $y_2$ | | $u_{-m+2}$ | $u_{-m+3}$ | ... | $u_2$ | $y_{-n+2}$ | $y_{-n+3}$ | ... | $y_1$ | | $b_n$ |
| ... | = | ... | ... | ... | ... | ... | ... | ... | ... | * | $- a_0$ |
| ... | | ... | ... | ... | ... | ... | ... | ... | ... | | ... |
| ... | | ... | ... | ... | ... | ... | ... | ... | ... | | $- a_{n-1}$ |
| $y_p$ | | $u_{p-m}$ | $u_{p-m+1}$ | ... | $u_p$ | $y_{p-n}$ | $y_{p-n+1}$ | ... | $y_{p-1}$ | | |

Le résultat est $R = (Q^T . Q)^{-1} . Q^T . P$
L'erreur commise sur chaque $y_i$ est $E = P - Q . R$

Voici les valeurs de u et de y :
ui = 1 si i = 0, sinon ui = 0
yi = h(t = Te * i), avec Te la période d'échantillonnage.

Comme précédemment on économise de la mémoire et du temps de calcul en divisant ce système en deux sous systèmes :

1er sous système :

| P | = | Q | | | | * | R |
|---|---|---|---|---|---|---|---|
| $y_{m+1}$ | | $y_{m-n+1}$ | $y_{m-n+2}$ | ... | $y_m$ | | $- a_0$ |
| $y_{m+2}$ | | $y_{m-n+2}$ | $y_{m-n+3}$ | ... | $y_{m+1}$ | * | ... |
| ... | = | ... | ... | ... | ... | | $- a_{n-1}$ |
| ... | | ... | ... | ... | ... | | |
| $y_p$ | | $y_{p-n}$ | $y_{p-n+1}$ | ... | $y_{p-1}$ | | |

Le résultat $R = (Q^T . Q)^{-1} . Q^T . P$ donne les coefficients de A.

$2^{ième}$ sous système :

Le $2^{ième}$ système est de type triangulaire et permet d'obtenir les coefficients de B :

$b_m = y_0$
$b_{m-1} = y_1 + a_{n-1}\ y_0$
...
$b_{m-i} = y_i + a_{n-1}\ y_{i-1} + a_{n-2}\ y_{i-2} + ... + a_{n-i}\ y_0 = yi + \Sigma_{k=1:i}\ (a_{n-k}\ y_{i-k})$
...
$b_0 = y_m + a_{n-1}\ y_{n-1} + a_{n-2}\ y_{n-2} + ... + a_{n-m}\ y_0 = y_m + \Sigma_{k=1:m}\ (a_{n-k}\ y_{i-k})$

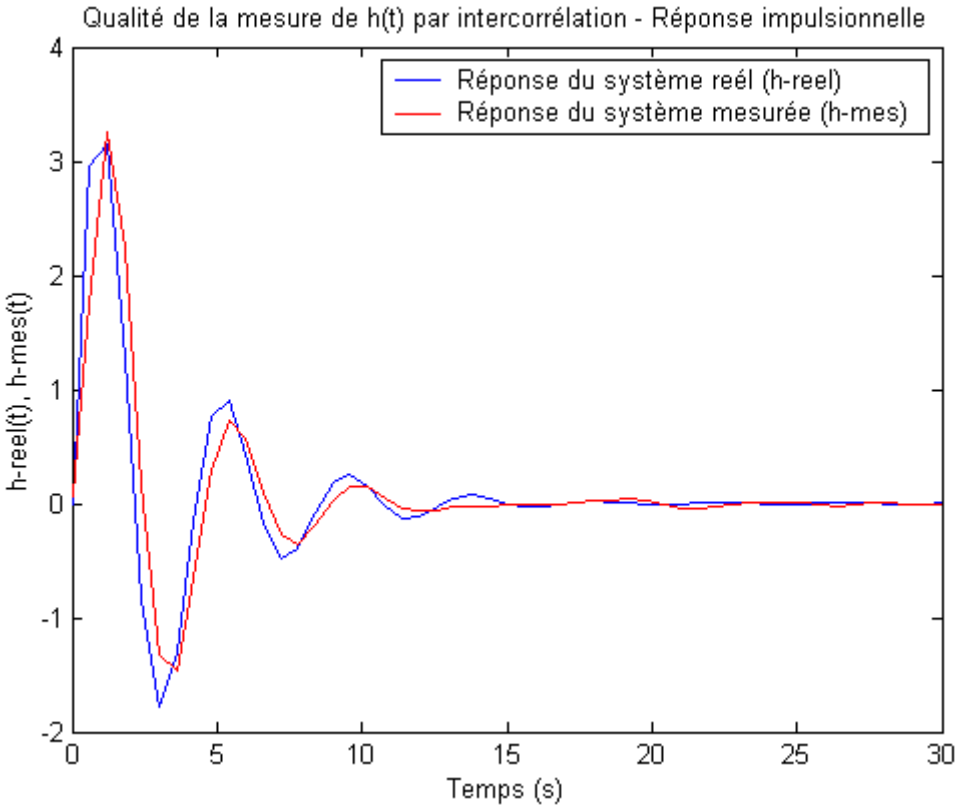
## 2. Tests avec MATLAB :

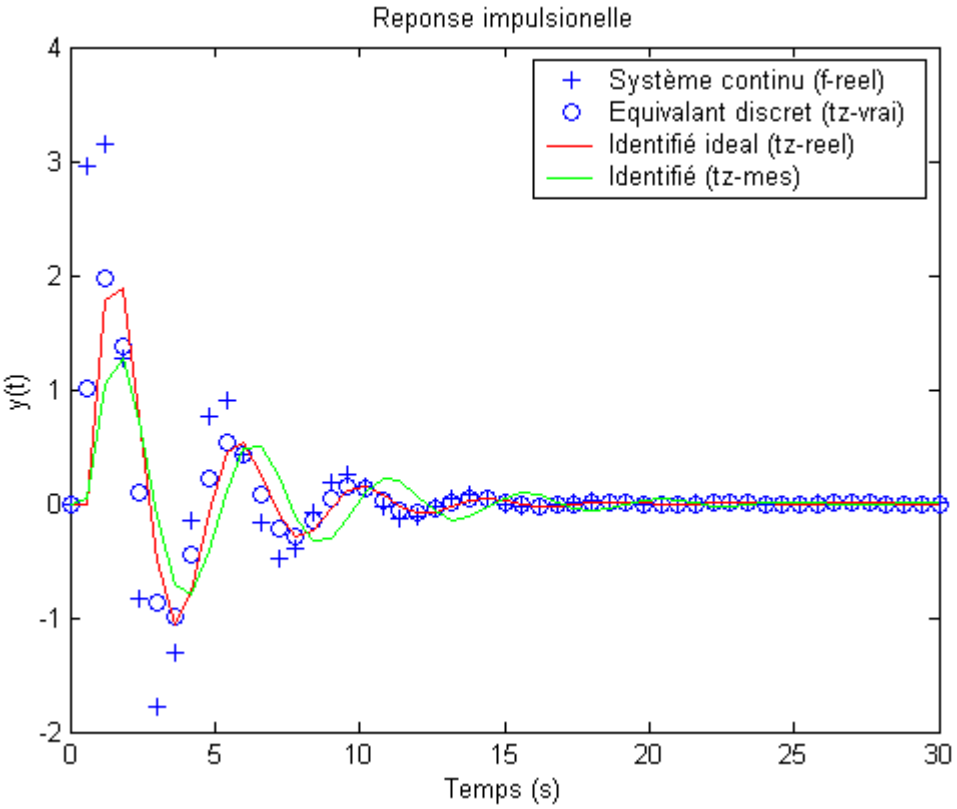
Les fichiers concernant cet essai se trouvent dans le dossier MATLAB9. Le script à exécuter est init.m.

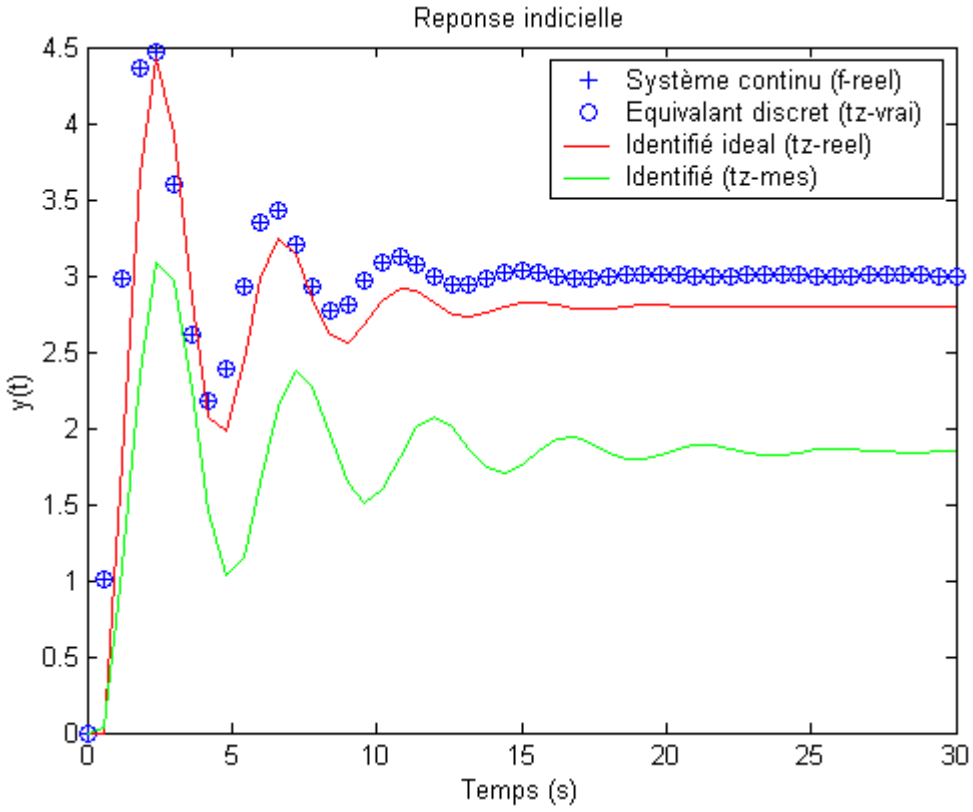
Dans ces tests, nous allons comparer :
- Les réponses du système réel (le système continu à identifier)
- Les réponses de son équivalent discret (obtenu avec C2D)
- Les réponses du système identifié idéal (c'est à dire à partir de h)
- Les réponses du système identifié (c'est à dire à partir de la mesure de h par corrélation)

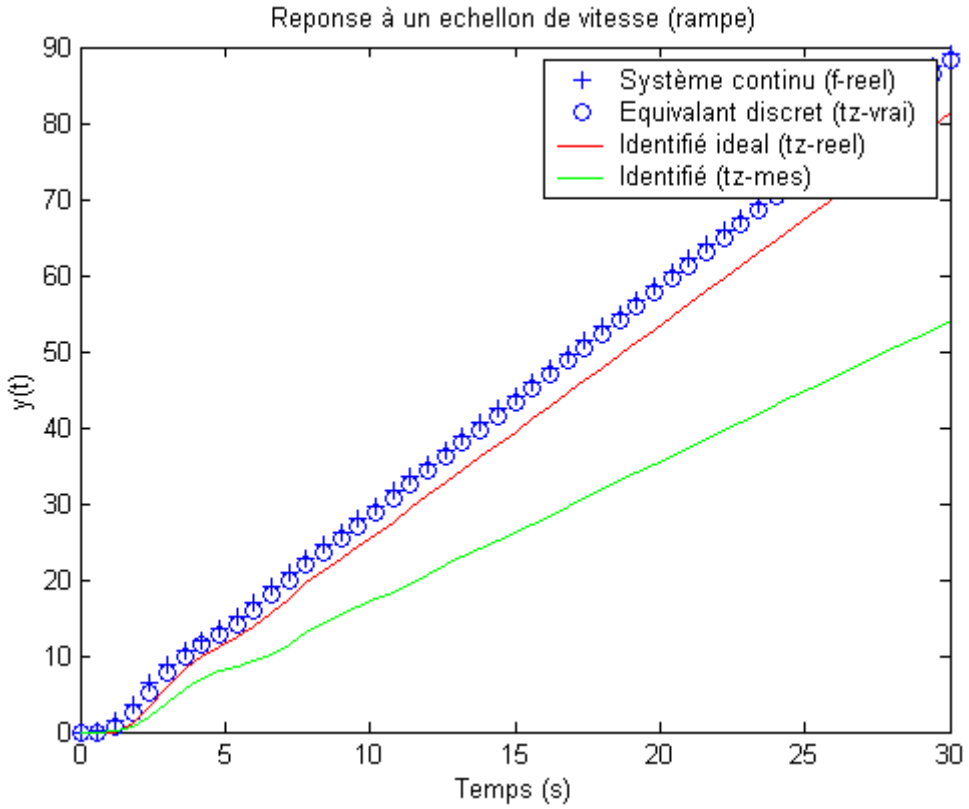
Les réponses testées sont les réponses impulsionnelles, indicielles, et à une rampe.

Voyons d'abord la différence entre h et sa mesure par corrélation :




Reponse impulsionelle
Système continu (f-reel)
Equivalant discret (tz-vrai)
Identifié ideal (tz-reel)
Identifié (tz-mes)
y(t)
Temps (s)


Reponse indicielle
Système continu (f-reel)
Equivalant discret (tz-vrai)
Identifié ideal (tz-reel)
Identifié (tz-mes)
y(t)
Temps (s)




Le programme inclus également un affichage texte :

```
SYSTEME REEL - 2ième ordre :
K = 3
W0 = 1.5 (rad/s)
EP = 0.2

Fonction de transfert réelle (continu) :
       6.75
------------------
s^2 + 0.6 s + 2.25

Equivalent discret (C2D) :
   1.012 z + 0.8948
----------------------
z^2 - 1.062 z + 0.6977
Sampling time: 0.6

TZ identifiée à partir de la réponse impulsionnelle théorique :
        1.777
----------------------
z^2 - 1.062 z + 0.6977
Sampling time: 0.6

TZ identifiée à partir de la réponse impulsionnelle mesurée (corrélation) :
  0.03641 z + 1.001
--------------------
z^2 - 1.238 z + 0.799
Sampling time: 0.6
```

Avant de commenter les résultats, on remarque d'abord des différences faibles mais non négligeables entre le système continu et son équivalent discret. Le simple fait de travailler en discret constitue donc bien intrinsèquement une approximation.

Les résultats sont mauvais, même à partir de la réponse impulsionnelle théorique (mesure parfaite) on n'arrive pas à trouver une bonne approximation de B.

Cela vient peut être du fait que l'entrée u est une impulsion de Dirac. Une entrée plus "riche" donnera peut être de meilleurs résultats.

D'où l'idée d'effectuer le calcul d'identification directement sur les mesures brutes, sans retrouver h par corrélation. En effet, les mesures brutes sont réalisées en injectant du bruit en entrée, donc par définition un contenu fréquenciel riche.

# 4ième essai

[Dossier MATLAB10]

## 1. Identification :

Nous allons essayer d'effectuer le même calcul que dans le 3ième essai, mais en prenant comme données les mesures brutes, et non plus l'estimation de h.

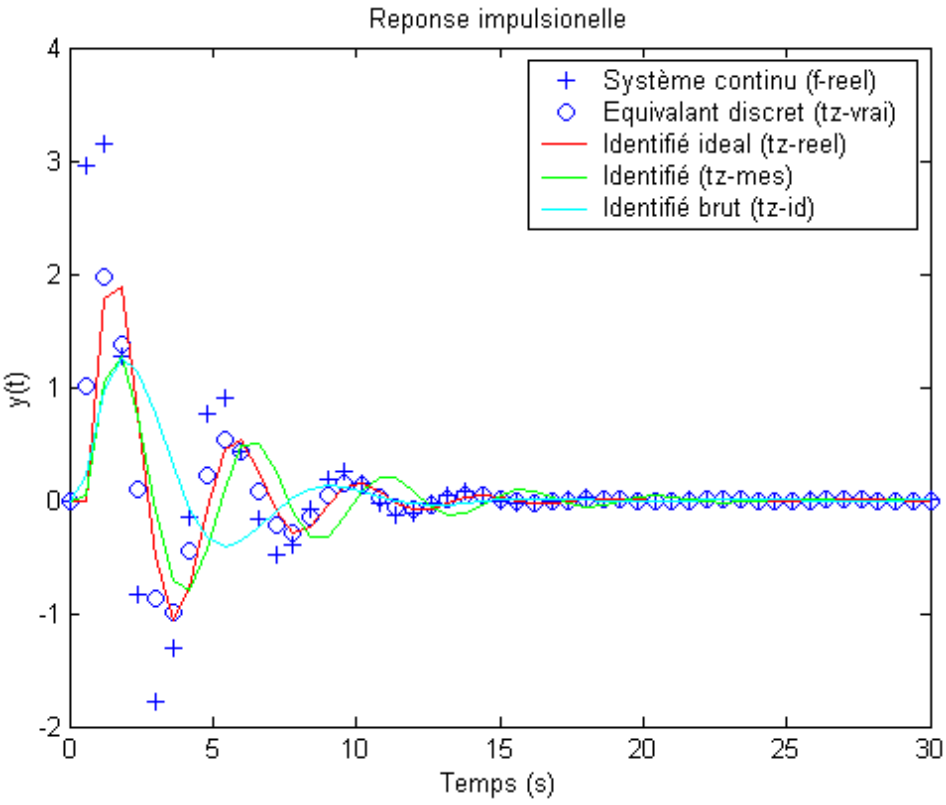
Ainsi, u sera le signal appliqué à l'entrée du système et y la sortie.

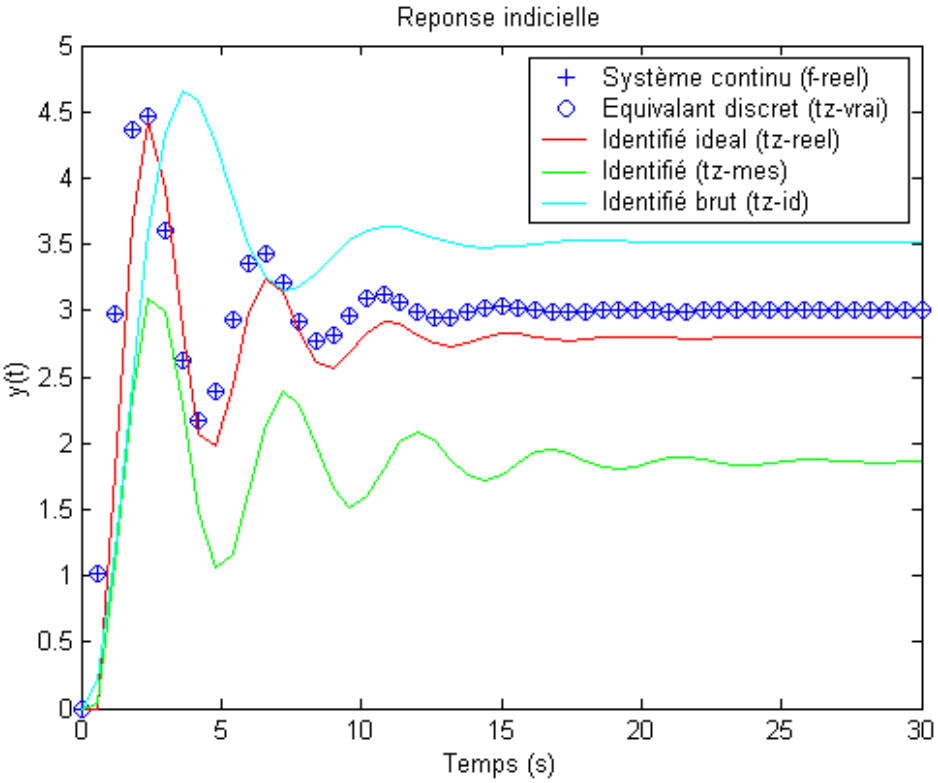
## 2. Tests avec MATLAB :

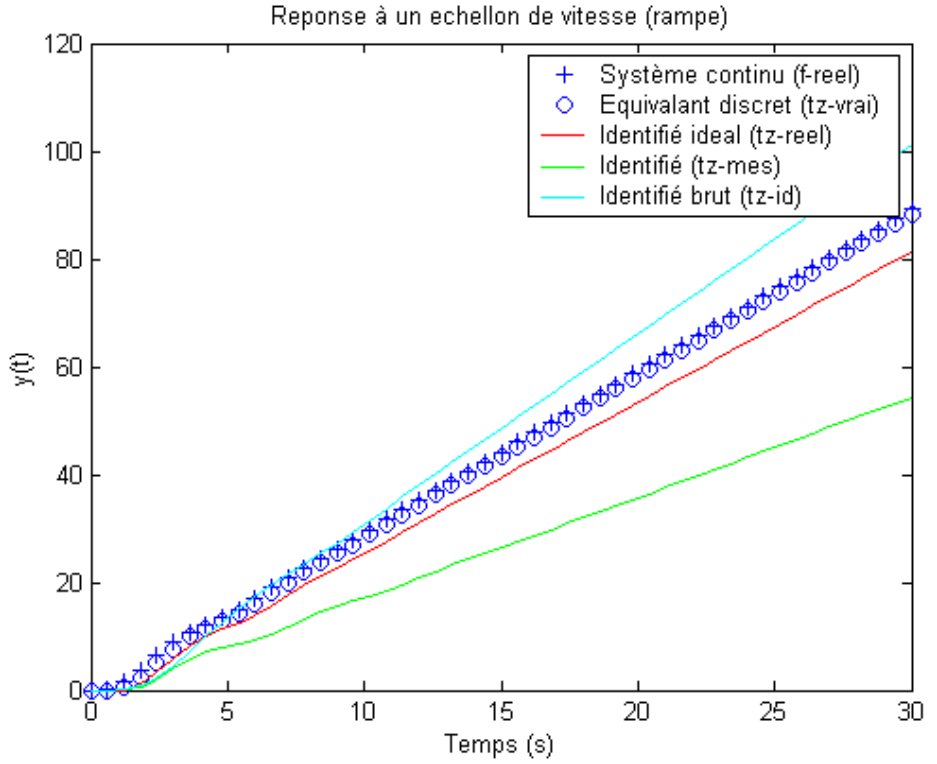
Les fichiers concernant cet essai se trouvent dans le dossier MATLAB9. Le script à exécuter est init.m.

Dans ces tests, nous allons comparer :
- Les réponses du système réel (le système continu à identifier)
- Les réponses de son équivalent discret (obtenu avec C2D)
- Les réponses du système identifié idéal (c'est à dire à partir de h)
- Les réponses du système identifié (c'est à dire à partir de la mesure de h par corrélation)
- Les réponses du système identifié à partir des mesures brutes

Les réponses testées sont les réponses impulsionnelles, indicielles, et à une rampe.






Le résultat la aussi est mauvais. La transformé en z identifiée à partir des données brutes est fortement influencée par la forme du signal d'entrée; en effet dans le programme le signal d'entrée est une sinusoïde superposée à du bruit blanc.

# Conclusion

L'étude en discret n'a pas abouti, mais si mon travail de recherche pour ce projet s'achève, c'est pour des raisons de calendrier, et non parce que le projet est dans une impasse.

En effet, il reste d'autres pistes à explorer, que ce soit en analyse numérique, pour mener à bien l'identification, ou en automatique pure, pour la synthèse du régulateur.

Néanmoins lors de ce projet j'ai exploré de nombreuses pistes, certaines fructueuses et d'autres non; elles seront toutes utiles à qui travaillera à la mise au point d'un régulateur auto-adaptatif.

Enfin, en travaillant pour ce projet, j'ai pu approfondir mes connaissances en automatique et acquérir une certaine habilité avec MATLAB.

# Maquette "Régulation thermique"

SOMMAIRE


| | |
|---|---|
| Introduction | Page 98 |
| Principe de fonctionnement | Page 99 |
| Alimentation | Page 101 |
| Ventilation | Page 102 |
| Chauffage | Page 103 |
| Mesure de température | Page 104 |
| Réalisation | Page 105 |

# Introduction

L'algorithme auto adaptatif est développé à partir de simulations MATLAB. Mais essayer cet algorithme sur une maquette réelle est toujours plus intéressant. En effet, sur un processus réel apparaissent les phénomènes suivants :

- non linéarités,
- bruits divers,
- saturations,
- zones mortes,
- hystérésis (souvent associé à la zone morte dans le cas de moteurs),
- transitoires plus ou moins complexes lors de la mise en route,
- ...

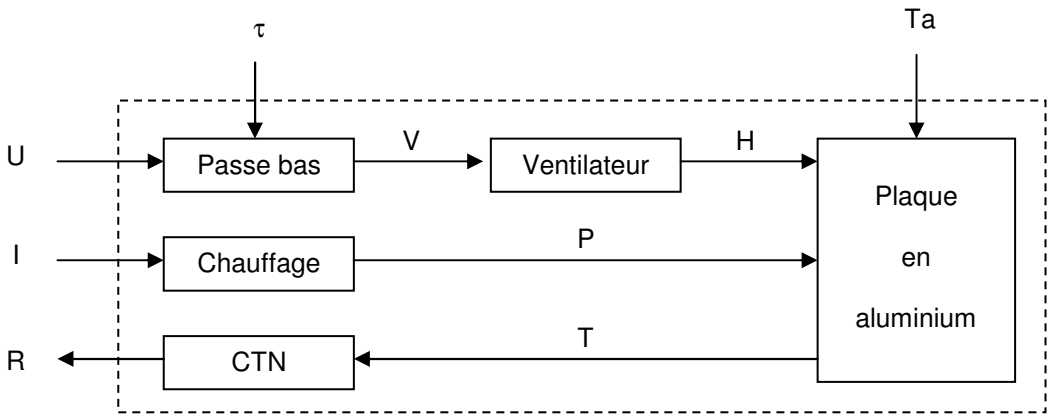
Certains de ces phénomènes peuvent êtres simulés avec MATLAB, mais seule une simulation "hardware" peut mettre en évidence les problèmes pratiques.

Parallèlement à mes travaux sous MATLAB, j'ai donc réfléchi à une maquette.

# Principe de fonctionnement

J'ai choisi pour la maquette un processus thermique, qui offre l'avantage de posséder des constantes de temps suffisamment lentes pour pouvoir voir ce qu'il se passe en temps réel.

Voici la structure de la maquette :




Description des variables :

U est la tension de commande pour la ventilation
V est la tension effectivement appliquée aux bornes du ventilateur
H est le coefficient de convection (résultat de l'action du ventilateur) entre la plaque et l'air
I est la consigne du courant traversant les résistances de chauffage
P est la puissance que transmettent les résistances de chauffage à la plaque
T est la température de la plaque
R est la résistance de la CTN
$\tau$ est la constante de temps du filtre passe bas
Ta est la température ambiante

Les variables V, H, P et T sont des variables internes au processus.

R est la sortie du processus. Une tension image de R est disponible sur une broche de sortie de la maquette pour acquisition par MUPSI. La valeur de R est parallèlement visualisée sur un Vumètre.

U et I sont des entrées du système; selon ce que l'on désire expérimenter, on pourra considérer U comme la commande et I comme une perturbation ou l'inverse. U et I peuvent être commandées au choix par un potentiomètre ou par une sortie du MUPSI.

Remarque : comme les résistances de chauffage reçoivent directement des impulsions PWM, P est proportionnel à Imoy. En fait, I est la consigne du courant moyen qui est ici différent du courant efficace. Pour plus de détails voir les schémas plus loin dans ce rapport.

Ta est une perturbation qui n'est pas commandable et qu'il n'est pas prévu de mesurer. En fait il aurait été plus sérieux de réguler T-Ta mais d'une part la maquette étant destinée à un usage

en laboratoire la dynamique de Ta n'est pas très importante, et d'autre part réguler T-Ta ne corresponds pas à un problème industriel concret.

$\tau$ est un paramètre du système. Les variations de $\tau$ constituent une perturbation paramétrique, ce qui permettra de tester la commande adaptative de ce processus dans le cas ou U est la commande. La valeur de $\tau$ est commandée par un potentiomètre.
La plage de variation de $\tau$ s'étale de 0.5 jusqu'à 20. Ainsi $\tau$ pourra être choisi pour être plus rapide ou plus lent que la constante de temps thermique (résistances + plaque + CTN).

Cette maquette est intéressante car elle possède un certain nombre de défauts rencontrés sur différents systèmes dans l'industrie :

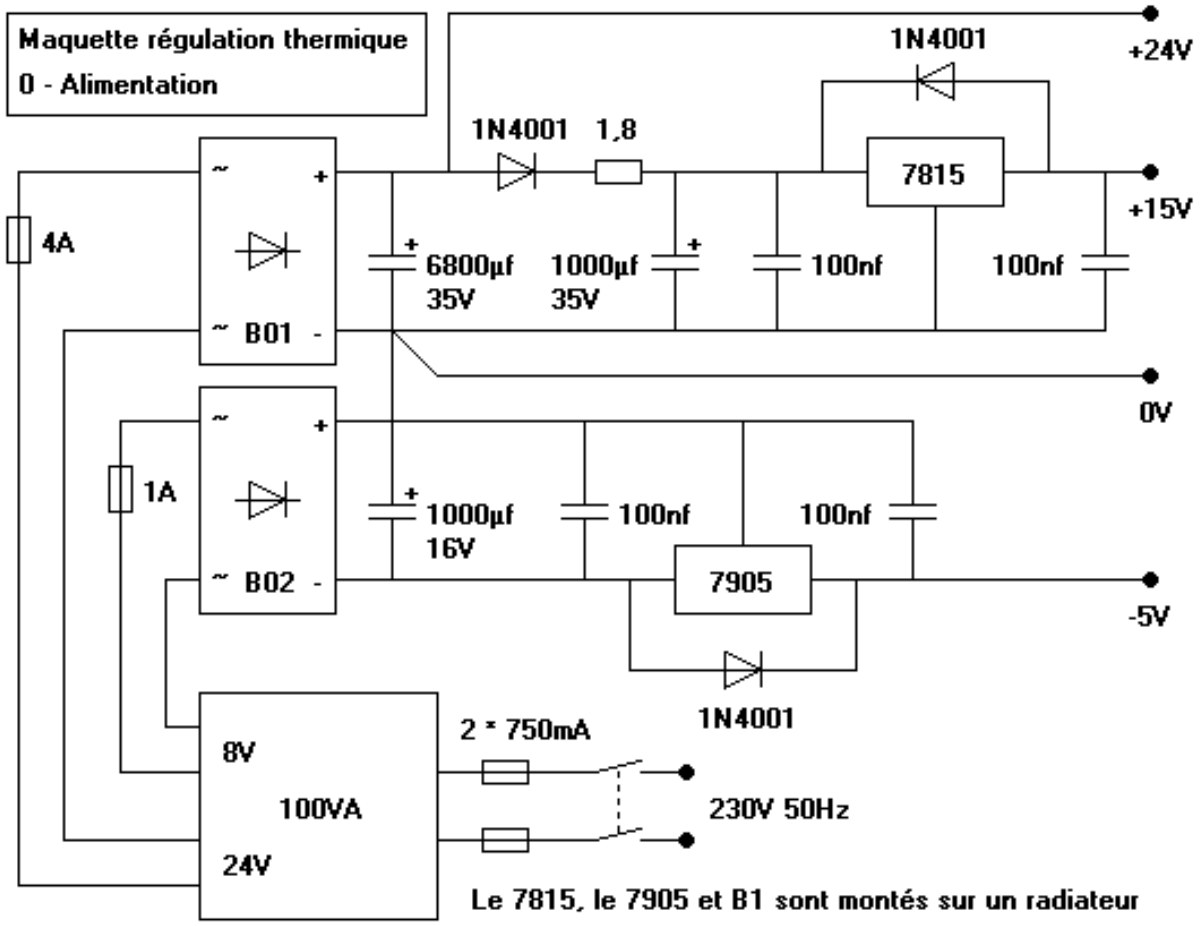
- présence de bruit (la commande PWM du chauffage sans lissage et le non blindage du circuit de mesure aboutirons à une mesure assez bruité),
- point mort et hystérésis pour le ventilateur,
- saturations des commandes,
- le transfert de chaleur abouti à un système d'ordre infini,
- non linéarités dues notamment à la CTN et à la relation entre V et H.

De plus cette maquette corresponds à un problème réel rencontré notamment en informatique : le refroidissement des semi-conducteurs. En effet, la puissance dissipée étant fonction du nombre de calculs, elle est donc inconnue;  pour prolonger la durée de vie des équipements informatique il faut maintenir la température des composants constante.

Enfin, comme nous le verrons plus en détail dans les schémas, la maquette est munie de nombreux potentiomètres ajustables qui permettent d'ajuster le niveau des sorties et des plages d'entrées. La maquette pourra ainsi être reliée au MUPSI mais aussi à toutes sortes d'équipements possédant des entrées/sorties analogiques.

# Alimentation

Commençons par étudier l'alimentation de la maquette. Voici le schéma :




Le schéma est tout à fait classique. Les tensions +15V et -5V servent de tensions de références et de tension d'alimentation pour les amplificateurs opérationnels. La tension +24V sert à alimenter la partie puissance : le ventilateur et les résistances de chauffage.

Le transformateur est "fabriqué" à partir d'un transformateur Legrand 24V auquel on bobine quelques spires de fil au-dessus du bobinage d'origine pour obtenir le secondaire de 8V.

La résistance de 1,8 Ohms avec le condensateur et la diode qui lui sont associé servent à découpler la partie +15V de la partie puissance.

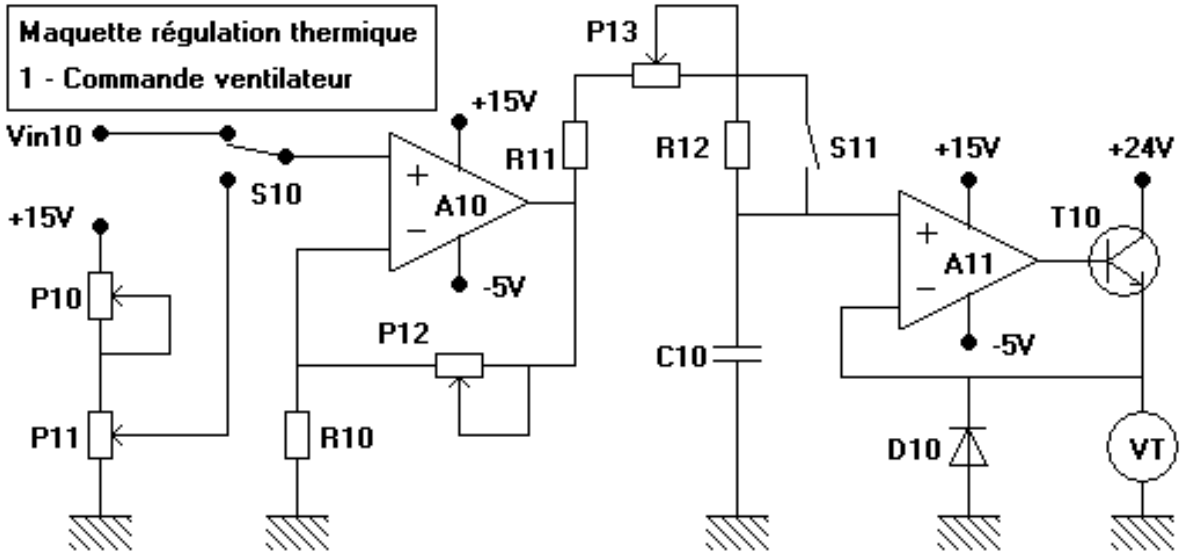
Les autres diodes protègent les régulateurs contre les inversions de tension.

Le radiateur sera une simple plaque en aluminium de 3mm d'épaisseur et d'une surface de 1dm².

Pour les fusibles et l'interrupteur 230V, on pourra utiliser un bloc "Beeling Lee" qui inclus le tout plus un filtre antiparasite dans un même boîtier.

# Ventilation

Voici le schéma de la partie qui commande le ventilateur :




Sont accessibles en face avant : S10, P11, S11, P13 et une douille banane pour Vin10.

S10 sert à choisir le type de commande : manuel (par P11) ou par MUSPI (dont une sortie doit alors être connectée à Vin10).

S11 et P13 permettent de régler la constante de temps. Elle est minimale si S11 est fermé et P13 en butée droite.

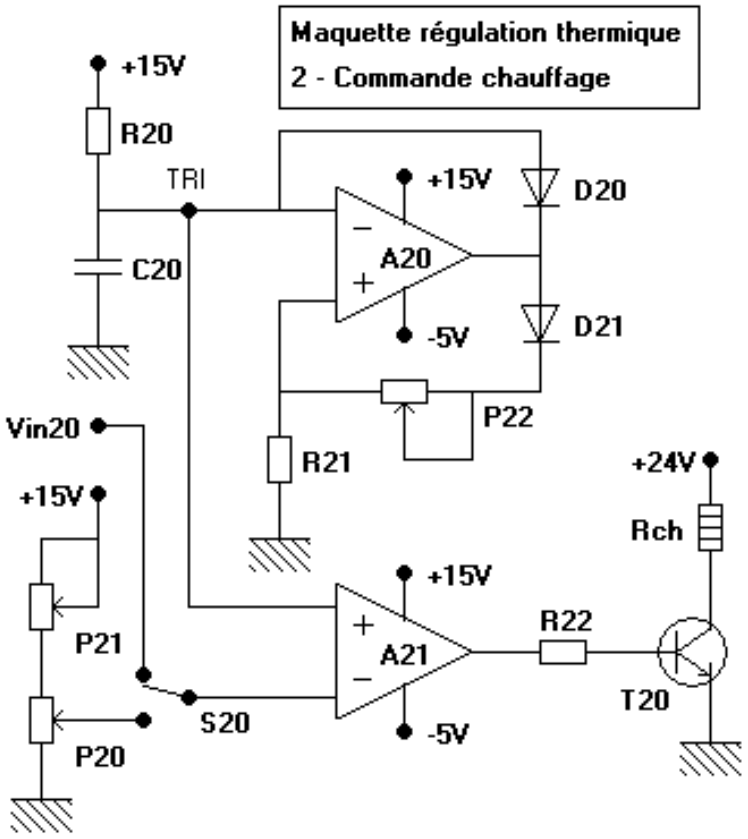
P12 doit être réglé de telle sorte que la tension de sortie de A10 soit de 12V quand la tension délivrée par le MUPSI est maximale (4,096V).

Ensuite, il faut régler P10 de telle sorte que la tension de sortie de A10 soit de 12V quand P11 est en butée haute.

P10 = 100K
P11 = 10K
R10 = 10K
P12 = 47K
R11 = 470
R12 = 10K
P13 = 10K
C10 = 1000µf
D10 = 1N4001
T10 = S53-3 ou équivalent
VT = Ventilateur 12V 80 mm équipant les alimentations de compatibles P.C.

# Chauffage

Voici le schéma de la partie qui commande le chauffage de la plaque :




R20 = 10K
C20 = 100nf
R21 = 6,2K
P22 = 22K
P21 = 100K
P20 = 10K
R22 = 330
Rch = 8
D20 = 1N4148
D21 = 1N4148
T20 = 2N3055

Sont accessibles en face avant : S20, P20 et une douille banane pour Vin20.

S20 sert à choisir le type de commande : manuel (par P20) ou par MUSPI (dont une sortie doit alors être connectée à Vin20).

L'ensemble R20, C20, R21, P22, A20, D20 et D21 constitue un générateur de dents de scies rudimentaire (il faudrait remplacer R20 par une source de courant). Les dents de scies démarrent d'environ 0V et montent jusqu'à un seuil réglable par P22. La fréquence est d'environ 1KHz avec un seuil de 5V.

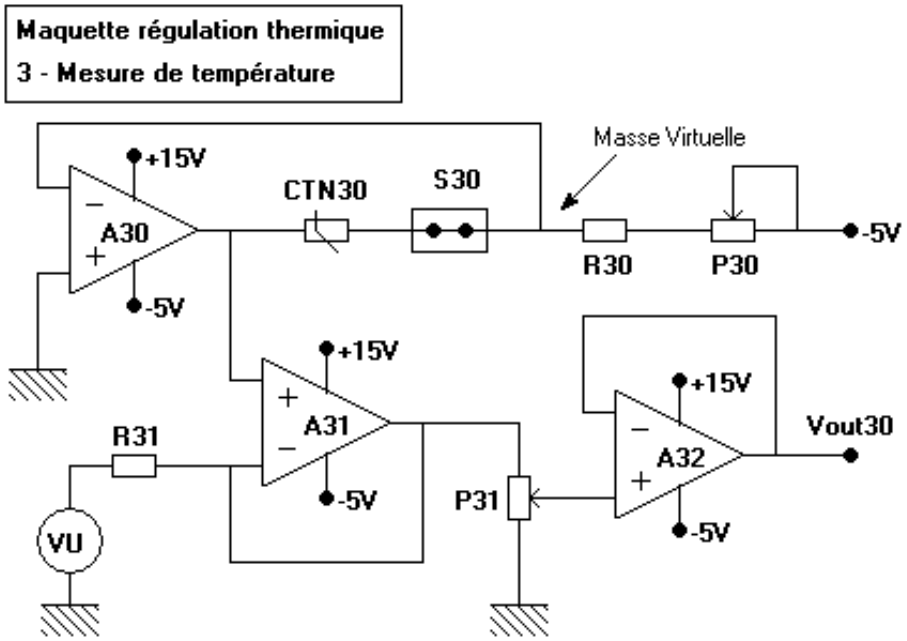
P22 doit être réglé de telle sorte que les dents de scies (mesurables entre le point "TRI" et la masse) montent jusqu'à la tension maximale pouvant être délivrée par le MUPSI (4,096V).

Ensuite il faut régler P21 de telle sorte que la tension entre l'entrée - de A21 et la masse corresponde au seuil précédent (4,096V) quand P20 est en butée haute.

Rch et T20 seront câblés en "volant"; l'émetteur de T20 sera relié directement à la masse de la partie alimentation par un fil de section suffisante. Rch est bien entendu fixé sur la plaque refroidie par le ventilateur.

# Mesure de température

La température de la plaque est mesurée avec une CTN. Le montage suivant délivre une tension proportionnelle à la résistance de la CTN :




Sont accessibles en face avant : VU et une douille banane pour Vout30.

Le principe de la mesure consiste à mesurer la tension aux bornes de la CTN qui est parcourue par un courant constant. La mesure a courant constant à deux avantages :
- Vout30 est proportionnel à la résistance de la CTN (U = R * I !)
- Pas d'emballement thermique : $\mathbf{T_{CTN}}\uparrow \Rightarrow R_{CTN}\downarrow \Rightarrow U_{CTN}\downarrow \Rightarrow P_{CTN}\downarrow \Rightarrow \mathbf{T_{CTN}}\downarrow$

La valeur du courant constant parcourant la CTN est réglable avec P30. En effet, on a :
$I_{CTN} = 5 / (R30 + P30)$.
Pour procéder au réglage, retirer le cavalier S30 et le remplacer par un milliampèremètre.

Pour les CTN employées comme capteur de température dans les cartes mères PII/PIII, choisir $I_{CTN}$ = 1 mA. En effet, ces CTN ont une résistance de l'ordre de 10K à 20°C, ce qui donne une tension de 10V, et une dissipation (auto échauffement) de 10mW. De plus, elles sont munies d'une languette en métal qui facilite leur montage contre la plaque.

P31 sert à adapter l'amplitude de Vout30 au niveau d'entrée du MUPSI (2.5V max.).

R31 est à choisir en fonction de VU.

R30 = 1K
P30 = 10K
R31 = 2,2K - (Pour $I_{VU}$ = 0,5mA à pleine échelle et $R_{VU}$ = 2300 Ohms)
P31 = 1K
CTN30 = Sonde de température pour carte mère PII/PIII

# Réalisation

Pour la réalisation de la maquette j'ai choisi deux circuits LM348 qui fournissent les 7 AOP nécessaires pour un encombrement minimal.

Une autre douille banane sera placée en face avant pour le 0V.

La plaque pouvant être très chaude, il faudra l'entourer d'un grillage empêchant la main de l'opérateur de venir en contact avec elle.

De même, il faudra éviter que le flux d'air chaud provenant du ventilateur ne soit dirigé vers les autres composants.

Disposant de quelques composants, j'ai déjà testé chaque partie du montage sur des plaquettes à montage rapide et pu vérifier leur bon fonctionnement.

Faute de temps, je n'ai pas réalisé la maquette. Ce document permettra à Arnatronic de la réaliser. Cette maquette peut aussi intéresser les enseignants de l'E.S.S.T.I.N. qui s'occupent des travaux pratiques.

# Informations utiles à propos de MATLAB

Quand j'ai commencé à coder des scripts MATLAB, j'ai tout de suite cherché à savoir s'il n'était pas possible de compiler des scripts pour obtenir un programme exécutable, pouvant être distribué librement.

Une telle fonctionnalité permettra de créer des outils informatiques directement exploitables par le commanditaire du projet.

Donc, j'ai contacté un ingénieur de la compagnie commercialisant MATLAB, M. Daniel MARTIN (daniel.martin@mathworks.fr).

Voilà les informations que j'ai pu obtenir :

MATLAB permet de créer des interfaces graphiques, appelées GUI. MATLAB dispose également d'un compilateur, qui permet de créer des programmes exécutables à partir de scripts et d'une GUI.

Mais le compilateur ne gère pas les objets MATLAB, les objets étant les types de données structurés. Cela est très gênant car MATLAB utilise de nombreux objets. Par exemple une fonction de transfert est un objet.

Aussi, Mathworks propose une alternative : le Run-time server. Celui-ci permet d'exécuter des programmes MATLAB sans restriction sur des ordinateurs ne disposant pas de MATLAB. Mais pour utiliser ce Run-time, il faut débourser 7000 € pour le kit de développement et rétribuer à Mathworks 1250 € de royalties par copie du Run-time distribuée.

Cependant, M. MARTIN m'a laissé entendre que dans les prochaines versions de MATLAB, le compiler prendrai en charge les objets. C'est une affaire à suivre, qui pourra se révéler très intéressante pour les projets E.S.S.T.I.N.

# Remerciements

Je tiens d'abord à remercier M. ORY qui m'a proposé ce projet et dont l'aide fut précieuse tout au long de ce semestre.

Je remercie M. JACQUES, de la société ARNATRONIC, qui a commandité ce projet et qui nous a apporté son point de vue d'industriel.

Je remercie également tous les enseignants du département automatique de l'E.S.S.T.I.N., notamment M. BLOCH qui m'a donné de nombreux conseils.

Enfin, je remercie Mme Fennerich qui m’a aider à réaliser ce rapport.