CYCLE 5

RESOLUTION DE PROBLEMES PAR UTILISATION DE L'INGENIERIE NUMERIQUE OU L'APPRENTISSAGE AUTOMATISE

TP2 - PSI

TP 2.1

CONTROL X – RESEAUX DE NEURONES

L'objectif de ce TP est de piloter le Control'X par un correcteur de type réseau de neurones.

1 MISE EN SITUATION

Le système industriel duquel est extrait Control'X est un robot portique 3 axes Lexium Max R du constructeur Schneider Electric. Ce robot portique est constitué :

- d'un axe portique double Lexium MAX S assurant un déplacement selon la direction X
- d'un axe portique double Lexium MAX H assurant un déplacement selon la direction Y.
- d'un axe Cantilever Lexium CAS 4 ou Lexium CAS 3 assurant un déplacement selon la direction Z.

L'application choisie pour contextualiser Control'X et celle du « pick and place » dans le domaine du placement de composants électroniques. Il s'agit d'un processus de précision consistant à positionner des composants électroniques sur des circuits imprimés.
Dans ce contexte d'utilisation, la particularité mécanique tient au fait que les efforts résistants extérieurs exercés sur l'axe sont nuls : le moteur sert uniquement à vaincre les efforts inertiels ainsi que les résistances passives. Le moteur est souvent en prise directe avec la poulie motrice ou, s'il y a un réducteur, le rapport de réduction est généralement faible.

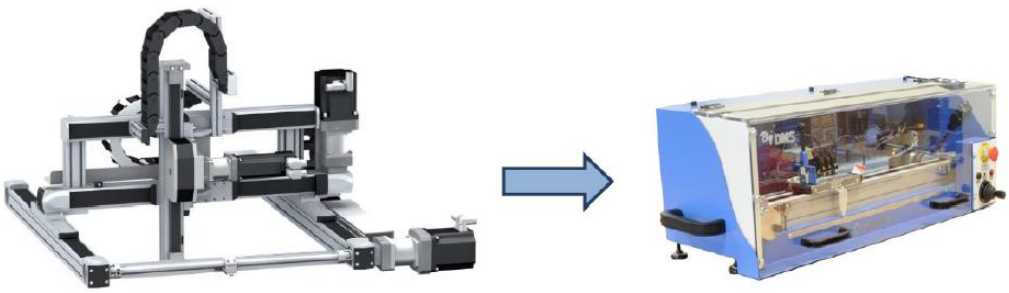


FIGURE 1 – Du Max R au Control'X

Le cahier des charges fixe les paramètre suivants pour une entrée échelon :

- pas de dépassement ;
- écart statique nul ;
- temps de réponse à 5 % de 0,2 s.

2 PREMIERE APPROCHE DU RESEAU DE NEURONE

2.1 Modélisation d'un schéma classique

Activité 1

Prendre connaissance du schéma-blocs proposé dans le fichier `Activite_01_Maquette.slx`. Retrouver l'ensemble des théorèmes, principes et hypothèses afin de construire ce schéma-blocs du système en boucle ouverte. Identifier l'influence de la perturbation sur la réponse.

On cherche à identifier quelques paramètres parmi l'ensemble des paramètres physiques.

Activité 2

Proposer l'ensemble des expériences à mettre en place afin de déterminer K_m, J_{eq}, R, L, f_v, m, r et d.

Afin d'améliorer les valeurs numériques des estimations précédentes des paramètres trop éloignées de leur valeurs pratiques, on cherche à superposer une réponse expérimentale à une réponse simulée.
Un essai en BO a été réalisé avec 9V en entrée pendant 5s.

Activité 3

Charger le fichier Activite_03_maquette_Corrige.m
Puis observer les courbes obtenues dans Activite_03_maquette_Corrige.xls

Vous pouvez également observer plusieurs tests dans Activite_03_maquette_Courbes.m

À la première vue des réponses, la pente en régime permanent n'est pas identique. Par conséquent, la valeur du coefficient K_m n'est pas correcte. De plus les oscillations des réponses ne sont pas en phase ; cela est probablement dû à une mauvaise estimation de l'inertie équivalente J_{eq}.
Afin de réduire les écarts entre le modèle et le réel, un algorithme de types $f(x) = 0$ va être utilisé sur les deux paramètres identifiés précédemment. L'algorithme proposé est celui de Newton introduit sous la forme :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{5}$$

qui se généralise pour un système d'équations à plusieurs variables :

$$\underline{x}_{n+1} = \underline{x_n} - \mathbf{J}^{-1}(\underline{x}_n) \cdot \underline{f}(\underline{x}_n) \tag{6}$$

où $\mathbf{J}$ représente la matrice jacobienne définie par :

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_{p-1}} & \frac{\partial f_1}{\partial x_p} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_{p-1}} & \frac{\partial f_2}{\partial x_p} \\ \cdots & & \cdots & & \cdots \\ \frac{\partial f_{m-1}}{\partial x_1} & \frac{\partial f_{m-1}}{\partial x_2} & \cdots & \frac{\partial f_{m-1}}{\partial x_{p-1}} & \frac{\partial f_{m-1}}{\partial x_p} \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_{p-1}} & \frac{\partial f_m}{\partial x_p} \end{bmatrix} \tag{7}$$

Activité 4

À la lecture du code proposé dans le fichier `Activite_04_Maquette.m`, retrouver les étapes suivantes :

- construction de la matrice jacobienne – comment est-elle construite ?
- première itération ;
- boucle ;
- conditions de fin d'itération.

Activité 5

Exécuter le code et vérifier que la correction des deux paramètres identifiés permet de coller au mieux à la réponse expérimentale. Conclure quant à la qualité du modèle à prédire une réponse du système en BO à une sollicitation à un échelon.

L'algorithme de Newton permet d'obtenir deux nouvelles valeurs de K_m et J_{eq} qui permettent à la simulation de coller à l'expérimentation. Le modèle proposé semble assez robuste afin de prévoir des réponses à une entrée en échelon.

2.2 Identification par un réseau de neurones

Activité 6

Faire une synthèse des réseaux de type NARX.

Consulter le document.

Il s'agit d'entrainer le réseau de neurones. On retrouve les étapes classiques d'un entrainement :

- choix des données ;
- structure du réseau ;
- entrainement ;
- analyse des résultats
- itération
- fermeture du modèle.

On rappelle ici que la modélisation par un réseau de neurones est un processus itératif.

2.3 Pré-entrainement

Activité 7

Proposer un type de signal d'entrée à imposer au système afin d'entrainer celui-ci. préciser les amplitudes de variation des caractéristiques de ce signal : amplitude, durée, nombres et échantillonnage. Penser également à l'amplitude du signal de sortie.
Générer alors un signal d'entrée permettant d'entrainer convenablement le système en BO.
Piloter le système/modèle par ce signal d'entrée.

Utiliser le fichier Activite_07_GenerationEchelon.m afin de construire ce signal en tapant dans la console :

```
[V, S,temps_discret,pwm_discret,signe_discret,pwm_signe_discret] = Activite_07_GenerationEchelons(0.01, 50, 1, 20)
```

L'instruction permet de générer un signal contenant 50 échelons de durée aléatoire (durée comprise entre 1 ×0,01s et 20×0,01s) et d'amplitudes aléatoires comprises entre −255 et +255.

On propose le fichier `Activite_07_Modele.slx` afin de piloter le modèle.

Activité 8

En s'appuyant sur le cours, proposer une première structure du réseau de neurones avec le nombre d'entrées différées, nombre de couches, nombres de neurones et fonctions d'activation.

L'équation différentielle, proposée lors de l'activité 1, est une équation différentielle d'ordre 3. Par conséquent, dans une première analyse, il semble important de fixer le nombre d'entrées retardées à 3. Le cours sur les NARX précise qu'on peut approximer tous le systèmes par un réseau comportant 2 couches. Pour le nombre de neurones, il faut se fixer une valeur pour débuter. Il n'existe pas de règles pour prédire cette valeur. Nous allons dans un premier temps essayer avec 20 neurones. La fonction d'activation entre la couche intermédiaire et la couche de sortie sera une fonction ReLU (`poslin` sous Matlab), puis une fonction linéaire en sortie du réseau de neurones.

2.4 Entrainement

On entrainera le modèle avec deux types d'entrainement :

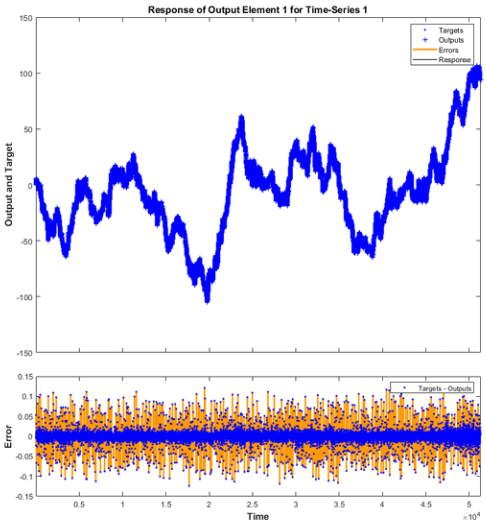
— régularisation de Bayes ;

— Levenberg-Marquardt.

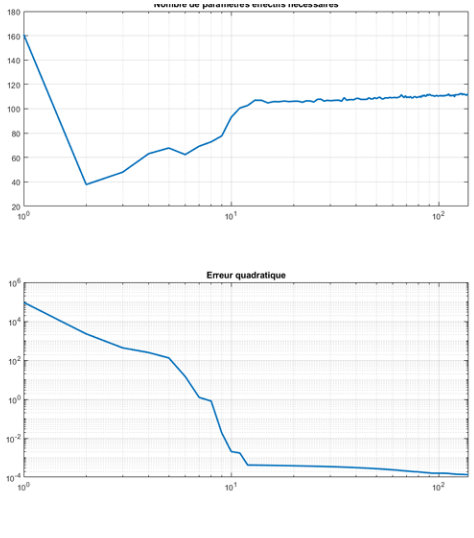
Dans un premier temps, la régularisation de Bayes est utilisé afin d'exploiter son résultat intéressant de proposition du nombre de paramètres utilisés par rapport par rapport nombre de paramètres disponibles.

Activité 9

Mettre à jour le script Matlab `Activite_09_NN.mlx` avec les paramètres fixés aux activités 7 et 8. Exécuter ce script. Observer la qualité prévisionnelle du modèle. Quel est le nombre de paramètres utilisés ? Est-il nécessaire d'augmenter ou de diminuer le nombre de neurones ?



(a) Comparaison modèle – données



(b) Évolution du nombre de paramètres effectifs et de qualité du modèle

FIGURE 4 – Résultats du premier entrainement

Activité 10

Ré-itérer le processus d'entrainement afin de valider la modélisation.

Il s'agit ici de jouer avec :

— le nombre de neurones;

— vérifier la convergence vers un minimum global et non local.

2.5 Post entrainement

Activité 11

Observer les produits de corrélation. En déduire si le nombre d'entrées retardées est suffisant. Puis ré-itérer l'ensemble du processus afin d'améliorer la qualité du modèle.

La qualité doit être à cette étape très bonne. Néanmoins, il a été construit sur l'hypothèse que le modèle était parfait et que par conséquent la deuxième entrée du modèle était exactement la sortie. Afin de prendre en compte un retour non parfait, il faut « boucler » réellement le NARX.

Activité 12

Boucler le modèle par réseau de neurones. Commenter la précision de celui-ci modèle. Entrainer-le de nouveau si nécessaire.

3 RETOUR AU CONTROLX :

3.1 Identification du modèle de la BO par un réseau de neurones

Activité 1

Prendre connaissance du schéma-blocs proposé dans le fichier `Activite_01_CX_Lin.slx`. Identifier ce que modélise le système d'ordre 1. Préciser ce qui ne semble pas avoir été modélisé.

Activité 2

Sachant que $T_{eq} = 0{,}022\,s$ et que la commande est saturée à $\pm 10\,V$, proposer des entrées permettant d'entraîner un réseau de neurones permettant de modéliser la boucle ouverte.

Activité 3

On se propose d'entraîner le modèle avec une succession aléatoire d'échelons. Proposer une durée minimale et maximale des échelons ainsi qu'une amplitude minimale et maximale. Pour générer une séquence d'échelons aléatoires, on utilisera le fichier `Activite_03_GenerationEchelons.m`.

Taper dans la console :

```
[U_discret] = Activite_03_GenerationEchelons(0.1,10,1,10,5)
plot(U_discret)
```

- Sur le Control'X la tension de seuil permettant de modéliser les frottement secs est de l'ordre de $1{,}5\,V$. Expliquer pourquoi un entraînement entre $-2\,V$ et $2\,V$ est peu judicieux.
- Expliquer pourquoi il est judicieux que, pour l'entraînement, la position estimée en utilisant le modèle de la boucle ouverte du Control'X soit comprise entre $-250\,mm$ et $250\,mm$.

Activité 4

En utilisant le fichier `Activite_04_NN_CX.mlx` configurer le réseau de neurones en vu de son entraînement (choix des données d'entrée, choix des paramètres du réseau).

3.2 Correction par réseau de neurones

Il s'agit en préambule d'avoir lu le cours sur les réseaux de type NARX et la section concernant le contrôle des systèmes dynamiques.

Activité 5

Faire une synthèse des Modele Reference Adaptative Controller (MRAC).

3.3 Pré-entrainement

Activité 6

Proposer un modèle idéal à identifier afin de satisfaire les exigences du cahier des charges ? On pourra se demander quel est le comportement souhaité, en régime permanent et en régime transitoire, quel ordre de fonction permet de vérifier ceci, etc.

Une clé de la bonne modélisation par réseau de neurones réside dans les données.

Activité 7

Proposer et générer des données permettant un bon entraînement du réseau de neurones correcteur. Préciser les amplitudes de variation des caractéristiques de ce signal : amplitude, durée, nombres et échantillonnage.

3.4 Entrainement

Activité 8

À l'aide du fichier `Activite_08_Train_Controller.mlx`, créer le réseau de neurone du contrôleur. Pour cela :

- renseigner les caractéristiques des données d'entrée ;
- configurer le configuration du système attendu en boucle fermée ;
- configurer les caractéristiques des échelons d'entrée ;
- configurer les caractéristiques du réseau de neurone associé au contrôleur.

Lancer alors l'entraînement du réseau avec le contrôleur.

La structure proposée pour le réseau de neurone donnée figure 2.

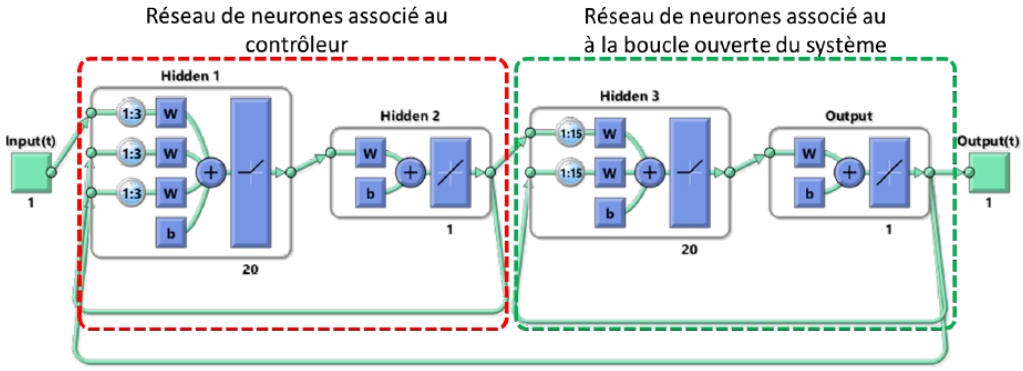


FIGURE 2 – Structure du MRAC

APPELER LE PROFESSEUR

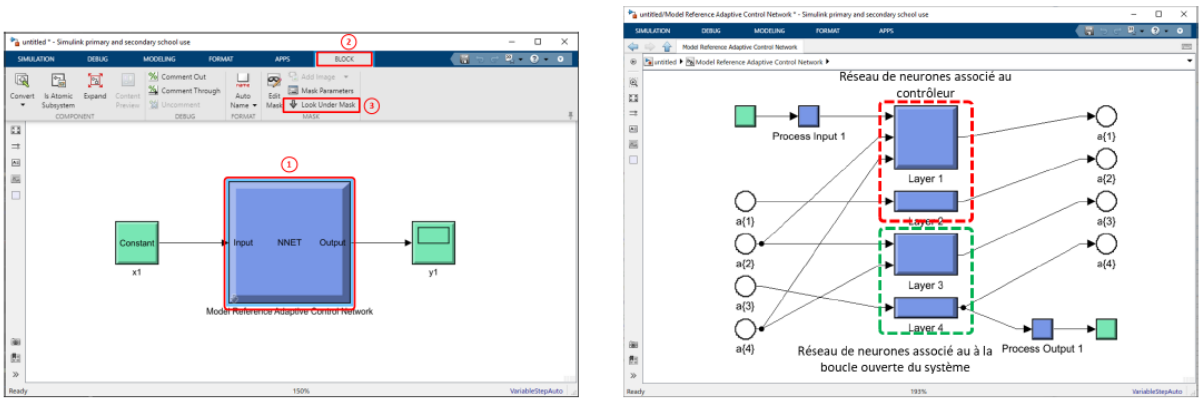
Activité 9

En utilisant les consignes suivantes, générer le fichier Simulink permettant de piloter le système par un réseau de neurones. On utilisera la fichier `Activite_09_PilotageANN_Vide.slx`.

Pour implémenter le réseau de neurone associé au contrôleur, nous allons copier les blocs Simulink associés au réseau de neurone du contrôleur et les coller sur une feuille permettant de piloter le système. Commençons par générer le bloc Simulink associé au MRAC en utilisant la commande `gensim(mrac_net)` et affichons ensuite la structure du réseau (Figure 4). Pour cela :

1. cliquer sur bloc bleu (MRACN) ;
2. cliquer sur l'onglet « Block » ;
3. cliquer sur « Look Under Mask » (raccourci clavier : Ctrl+u).

Copier alors les couches 1 et 2 dans le MRAC associé au pilotage du Control'X.



(a) Bloc Simulink du MRAC (b) Structure Simulink du réseau de neurones

FIGURE 4 – Récupération du réseau de neurones du contrôleur

La figure 5 (il faut avouer qu'elle est super belle) illustre la zone où copier le réseau de neurones dans le fichier `Activite_09_PilotageANN_Vide.slx`.

Activité 10

Implémenter sur le système réel en utilisant le bouton « Run in Real Time » et évaluer les écarts.

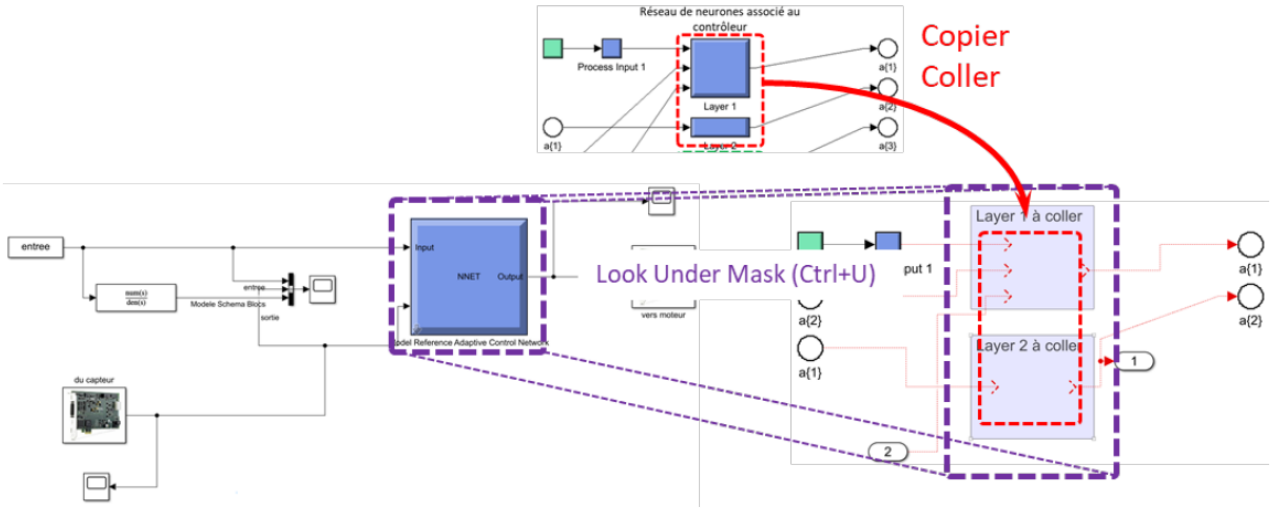


FIGURE 5 – Copie du contrôleur

4 SYNTHESE

Activité 11

Proposer une synthèse sur les avantages et les inconvénients sur la correction mise en œuvre dans ce TP par rapport à l'utilisation d'un correcteur PI(D).