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.

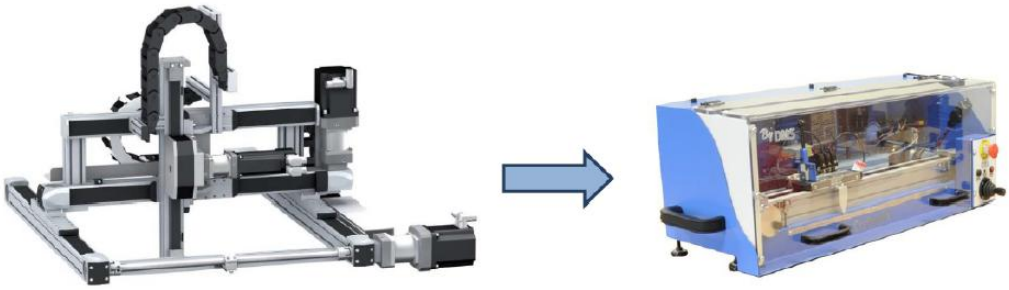


FIGURE 1 – Du Max R au Control'X

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.

La modélisation de la machine à courant continu reprend les équations classiques avec prise en compte de l'inductance et d'un frottement visqueux :

$$u(t) = L\frac{\mathrm{d}i(t)}{\mathrm{d}t} + Ri(t) + e(t) \quad (1)$$

$$C_m(t) = K_m i(t) \quad (2)$$

$$J_{eq}\frac{\mathrm{d}\omega(t)}{\mathrm{d}t} = C_m(t) - f_v\omega_m(t) - C_r(t) \quad (3)$$

$$e(t) = K_m\omega_m(t) \quad (4)$$

L'équation électrique est issue de la loi des mailles ; l'équation mécanique est issu d'un théorème de l'énergie cinétique. Puis la transmission est classique par introduction du train d'engrenages et de son rapport de réduction r et l'intégration de la vitesse en position. Le couple résistant $C_r(t) = -mgdr\sin\theta_b(t)$, avec $\frac{\mathrm{d}\theta_b(t)}{\mathrm{d}t} = \omega_b(t)$ la vitesse de rotation du bras, est ramené sur l'arbre moteur.
Les conditions initiales sont supposées nulles.
La perturbation, quant à elle, engendre des oscillations mais en plus créé un phénomène non-linéaire assimilable à une dissymétrie du comportement lors de la descente de la charge par rapport à sa phase de montée.

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.

Les différentes expériences possibles à mettre en place sont :

- K_m : échelon de tension en entrée, mesure de la position $\theta_b(t)$; la pente en régime permanent est l'image de K_m.
- J_{eq} : somme des inerties individuelles.
- R : moteur seul ; rotor bloqué. Mesures de u et i pour différentes valeurs de u. La pente est R.
- L : la mesure de $i(t)$ en régime permanent par une pince ampèremétrique. La constante de temps vaut $\frac{L}{R}$; en ayant identifié R, on détermine L.
- f_v : essai à vide, mesure de i avec la pente et K_m on peut remonter à f_v
- m : balance.
- r : sur le moteur démonté, on compte le nombre de dents des différents niveaux. Voir photos des engrenages.
- d : mètre.

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.

Le découpage se retrouve de cette manière :

- construction de la matrice jacobienne (lignes 39 à 61) : cette matrice, eq. 7, se construit par colonne, la première correspond à la variation du résidu par rapport à la variation du premier paramètre d'étude K_m, la seconde par rapport au second paramètre J_{eq}. Cette matrice est obtenue par différences finies où le terme $\frac{\partial f_m}{\partial x_p}$ est approximé par :

$$\frac{\partial f_m}{\partial x_p} \simeq \frac{f_m(x_p + \epsilon) - f_m(x_p - \epsilon)}{2\epsilon} \tag{8}$$

- première itération (lignes 77 à 78) : on retrouve la formule 6. On note que le terme $\mathbf{J}^{-1} \cdot$ est remplacé dans Matlab par une opération \ qui représente une minimisation au sens des moindre carrés.
- boucle (lignes 81 à 134) : on répète la construction de la matrice jacobienne puis la détermination du vecteur inconnu.
- conditions de fin d'itération (ligne 83) : 2 conditions de fin sont présentes, une en nombre d'itérations et l'autre sur la variation du résidu (norme de la variation du résidu inférieur à une valeur limite).

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.

Tout d'abord, listons les différentes formes de signaux que l'on peut exploiter : échelon, rampe, sinus, exponentiel, etc. On sait en avance qu'il faudra que la commande soit une forme exponentielle (pour une entrée en échelon ou en rampe). Par conséquent, une entrée de la forme $K\left(1-e^{\frac{t}{\tau}}\right)$ serait idéale. Cela élimine de facto un entrainement avec des signaux en sinus. Il est par contre assez facile d'approximer ces fonctions exponentielles par une « somme » d'échelons (ou de rampe). Une façon donc assez pratique est d'entrainer notre réseau de neurones par des fonctions échelon ayant des durées variables entre une valeur faible par rapport au temps de réponse caractéristique et et une valeur de l'ordre du temps caractéristique. Par conséquent, le réseau, pourra traduire le régime transitoire ainsi que le régime permanent. De, plus il faut balayer l'ensemble des tensions de commande du moteur, soit de 0 V à 9 V. ces deux paramètres doivent disposer d'une répartition aléatoire mais suivant une distribution uniforme pour avoir autant de chance d'apparaître.
Pour le nombre d'échantillons à générer, plus il y a de données meilleur sera le réseau de neurones. Par conséquent, il ne faut pas hésiter à exploiter les capacités matériel.

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 ;

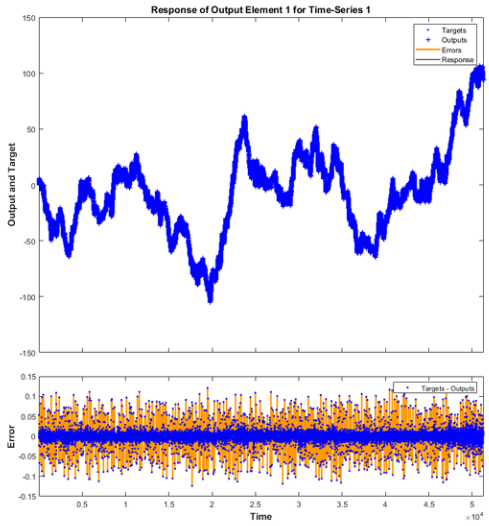
— Levenberg-Marquardt.

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.

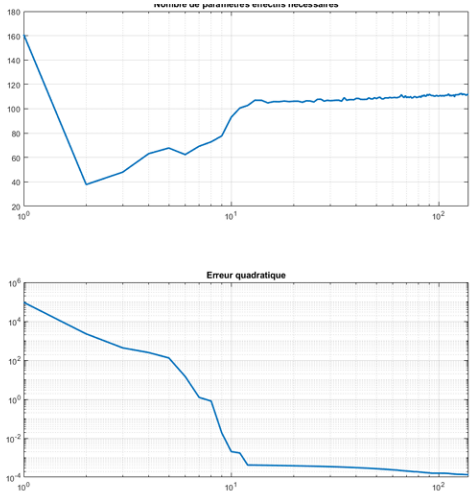
Activité 9

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 ?

L'exécution du script montre que le modèle par réseau de neurones colle très bien aux données si vous avez respecté les valeurs précédentes (fig. 4a).
On remarque également que le nombre de paramètres effectifs converge vers une valeur de 40, ce qui représente les 2/3 des paramètres disponibles (fig. 4b). À priori vu que l'on ne souhaite pas améliorer les performances d'exécution, il est tout à fait possible de laisser ce surplus de paramètres (surtout que l'on n'a pas encore ajusté le nombre d'entrées retardées, voir activité 11.



(a) Comparaison modèle – données



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

FIGURE 4 – Résultats du premier entrainement

Activité 10

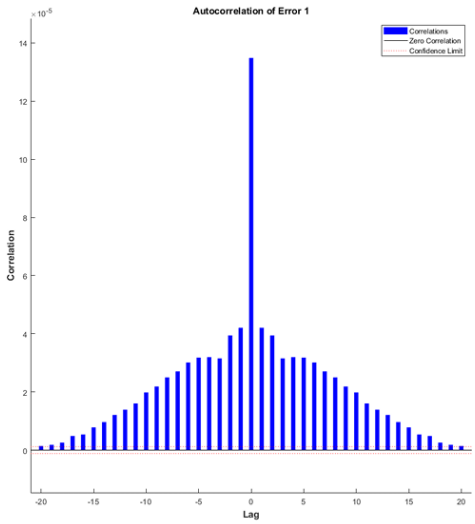
Ré-itérer le processus d'entrainement afin de valider la modélisation.

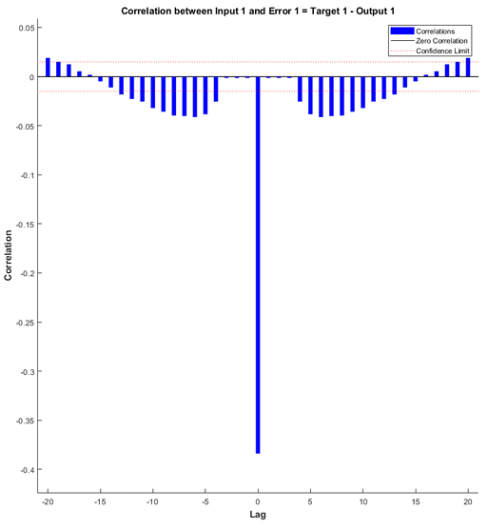
Il s'agit ici de jouer avec :
— le nombre de neurones;
— vérifier la convergence vers un minimum global et non local.

2.5 Post entrainement

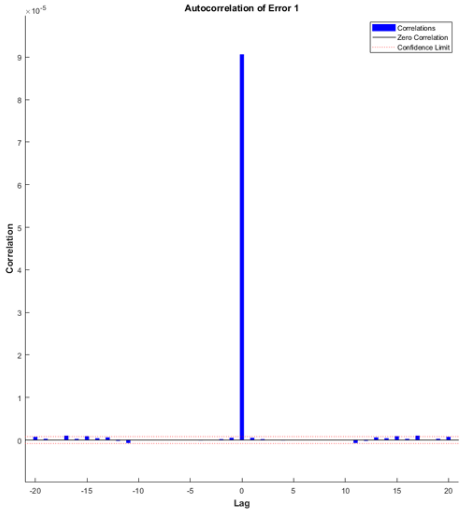
Activité 11

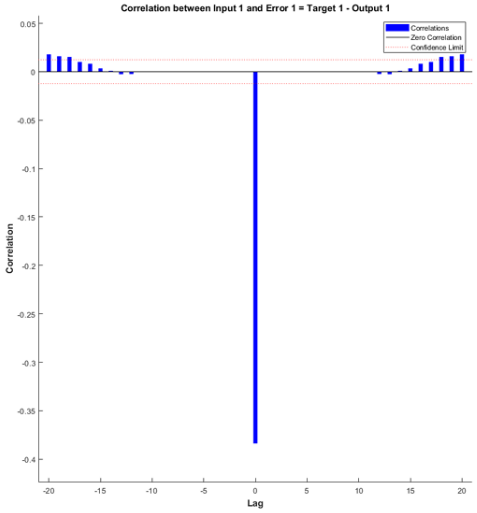
Observer les produits de corrélation. En déduire si le nombre d'entrées retardées est suffisant. Puis ré-itérer l'ensemble du processus afin d'améliorer la qualité du modèle.





Les lignes rouges pointillées des figures 5a et 5b indiquent les limites de confiance du modèle. Nous pouvons constater que les deux fonctions de corrélation se situent en dehors de ces limites à plusieurs endroits. Cela indique que nous devrions augmenter le nombre d'entrées retardées.
En réalisant cette augmentation, on améliore ces fonctions, figures 6a et 6b sans pour autant garantir une prédiction totalement correcte.





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.

Le fait de construire le modèle sur les données renvoyés par le modèle lui-même montre que celui-ci perd en qualité. À cette étape il est alors possible de ré-entrainer le modèle sur le modèle réellement bouclé afin d'améliorer ses qualités de prédiction pour un coût de calcul raisonnable.

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.

Le système d'ordre 1 modélise le variateur, le moteur, le réducteur et le système poulie-courroie.
Les frottements secs et visqueux ainsi que le saturations ne sont pas modélisées.

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 $250\,mm$.

Un entraînement sur des tensions trop souvent inférieures à la tension de seuil (en valeur absolue) ne sera pas pertinent car on sera dans une plage de travail où le modèle est peu fidèle à la réalité.
De même, si les échelons générés provoquent des déplacements en dehors de la plage de déplacement du Control'X, on s'entraînera sur une zone qui n'est pas la zone d'utilisation.
Enfin, il faudra veiller à ce que les déplacements couvrent toute la zone de travail afin que le réseau de neurones puisse s'entraîner sur l'ensemble de la plage accessible.

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.

La réponse temporelle d'un système d'ordre 1 ou d'ordre 2 avec $z > 1$ permet de valider l'exigence de non dépassement. L'exigence de précision peut être atteinte quel que soit l'ordre avec une amplification statique de 1 ; enfin la rapidité n'influence pas le choix. Les auteurs proposent de choisir plutôt l'ordre 2 par la non-discontinuité de sa dérivée en 0^+ afin de rendre plus souple le comportent du correcteur par réseau de neurones. Ainsi la fonction de transfert objectif à suivre est :

$$H_{obj}(p) = \frac{1}{1 + 2 \times 1\frac{p}{30} + \frac{p^2}{30^2}} \tag{1}$$

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.

Le système sera sollicité essentiellement en réponse à des échelons de positions. On se fixe une période d'échantillonnage de $T_e = 0{,}01\,\text{s}$. La durée de ces échelons doit être comprise entre $0{,}1\,\text{s}$ (choix arbitraire) et $1{,}5\,\text{s}$ (choix permettant de mettre en évidence le régime permanent). Les amplitudes de sortie correspondent aux possibilités du système. Ici, nous souhaitons entraîner le réseau de neurones sur une amplitude $[-100; 100]$ mm. Le nombre de 20 000 points permet d'obtenir globalement une petite centaine d'échelons, ce qui semble être un nombre correct vis-à-vis de entraînement qui suit.
L'identification du système par un réseau de neurone a été effectué lors de l'activité 4. Par conséquent, on peut passer à l'étape suivante : l'entraînement du contrôleur.

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.

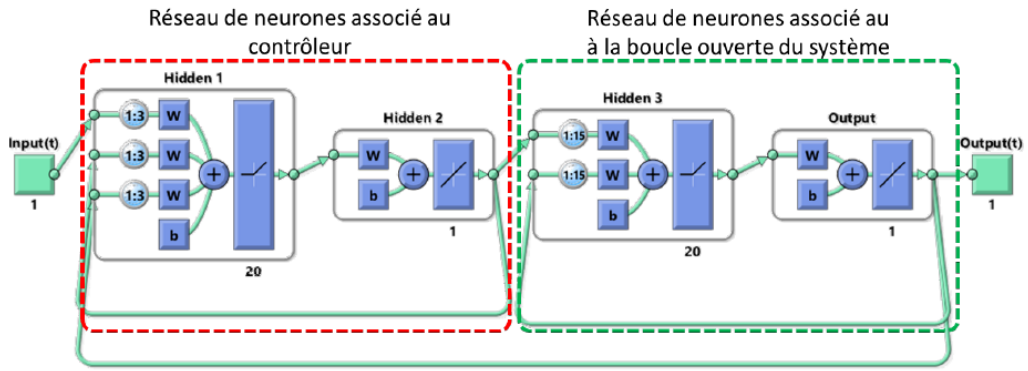
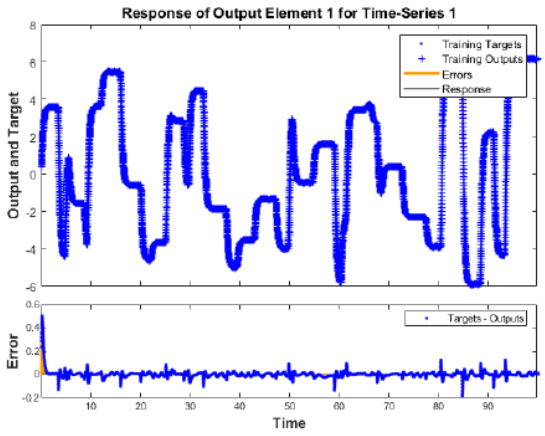
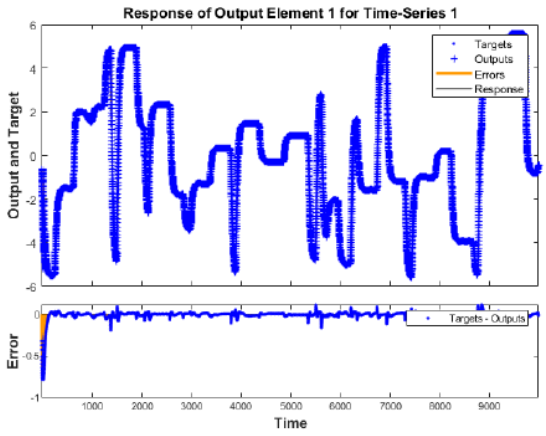


FIGURE 2 – Structure du MRAC



(a) Résultat de l'entraînement du contrôleur



(b) Simulation du réseau avec des données de test

FIGURE 3 – Entraînement et inférence du MRAC

APPELER LE PROFESSEUR

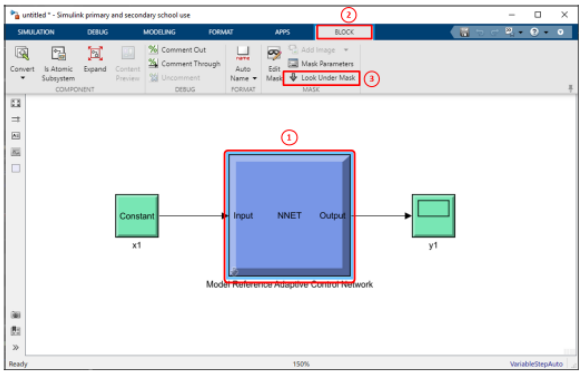
Activité 9

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`.

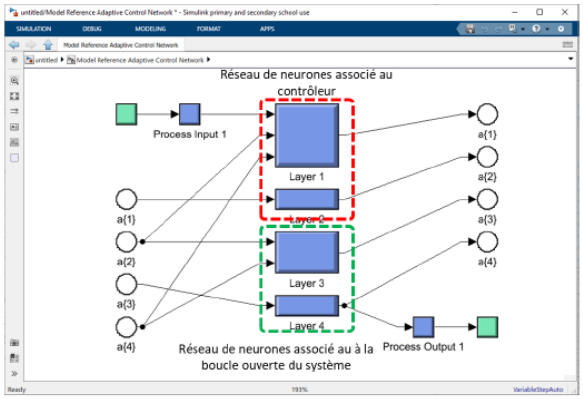
Pour implémenter le réseau de neurone associé au contrôleur, nous allons copier les blocs Simulink associés au réseau de neurone du contrôleur et les coller sur une feuille permettant de piloter le système. Commençons par générer le bloc Simulink associé au MRAC en utilisant la commande `gensim(mrac_net)`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

Sur le premier échelon :

- écart statique souhaité : 0 mm. Écart statique mesuré : 8 mm.
- temps de réponse souhaité 0,2 s. Temps de réponse mesuré : $< 0{,}2$ s.

On constate qu'au début du régime transitoire les allures de la courbe cible et de la mesure évoluent de façon similaire. La commande du contrôleur ANN n'est ensuite plus satisfaisante car la tension délivrée est inférieure à la tension de seuil du système. En effet, ce phénomène n'a pas été correctement modélisé. Il est donc normal que la commande soit moins satisfaisante.

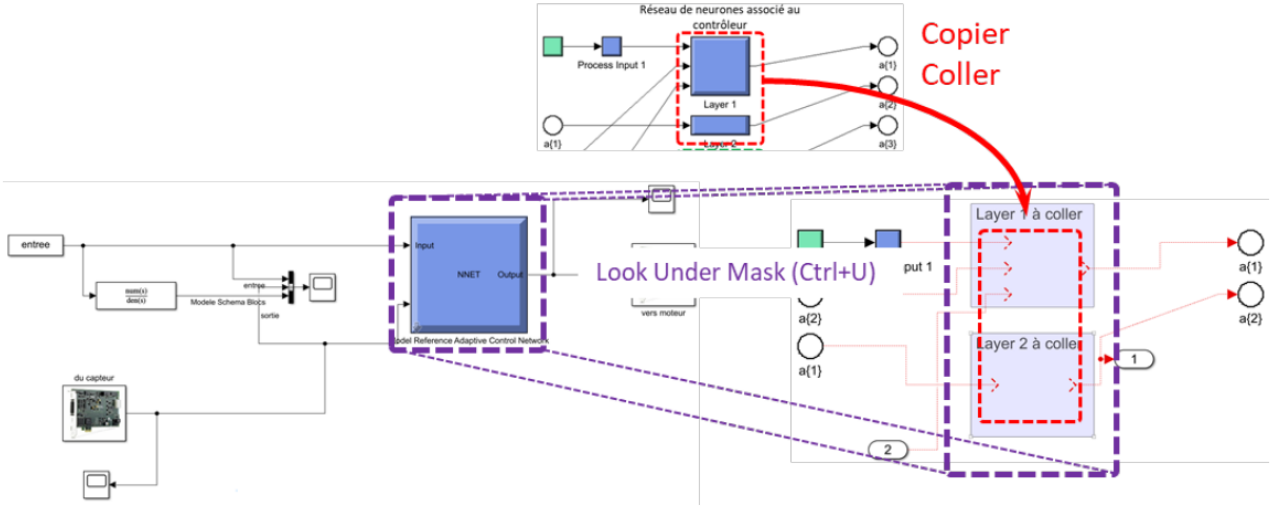


FIGURE 5 – Copie du contrôleur

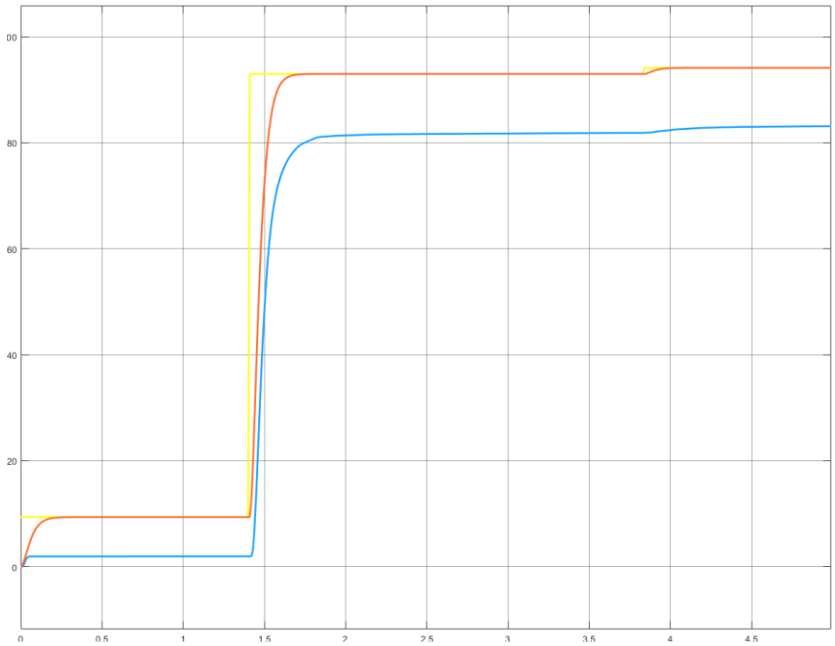


Figure 6 – Résultat donné par le système contrôlé par un réseau de neurones (*Jaune : consigne, Rouge : cible, Bleu : mesure sur le Control'X*)

Sur le deuxième échelon, la variation de consigne est plus grand, ce qui donne une impression plus satisfaisante. Cependant les mêmes phénomènes sont visibles : régime transitoire satisfaisant et régime permanent qui ne valide pas le cahier de charges.

Évaluation des écarts par rapport au cahier des charges :

- écart statique souhaité : 0 mm. Écart statique mesuré : 8 mm.
- temps de réponse souhaité 0,2 s. Temps de réponse mesuré : $< 0{,}2$ s.

On observe un écart, notamment sur la valeur de l'écart statique. Cela peut s'expliquer par une « mauvaise » identification du comportement du système lors de l'entraînement du réseau de neurones. Il est en effet difficile d'obtenir des résultats satisfaisants avec une identification sur un modèle non linéaire. Une mauvaise modélisation de la tension de seuil peut donc expliquer que le contrôleur ne parvienne pas à supprimer l'écart statique.
Il faudrait donc déterminer des paramètres de chacun des réseaux de neurones permettant d'obtenir de meilleurs résultats.

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).

Avantages :

- Relative facilité de mise en œuvre du processus de corrections.
- Possibilité de réglage sur PC avant implémentation sur cible.
- Pas/Peu d'essais nécessaires.
- Possibilité d'entraîner le correcteur avec une fonction de transfert directement choisie par le concepteur.

Inconvénients :

- Nécessité de disposer d'un modèle relativement fiable.
- Difficulté de choisir les caractéristiques du réseau de neurones.
- Difficulté de choisir le type d'entrées.
- Processus d'entraînement peu répétable et qui peut échouer.