CYCLE 5

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

TD2 - PSI

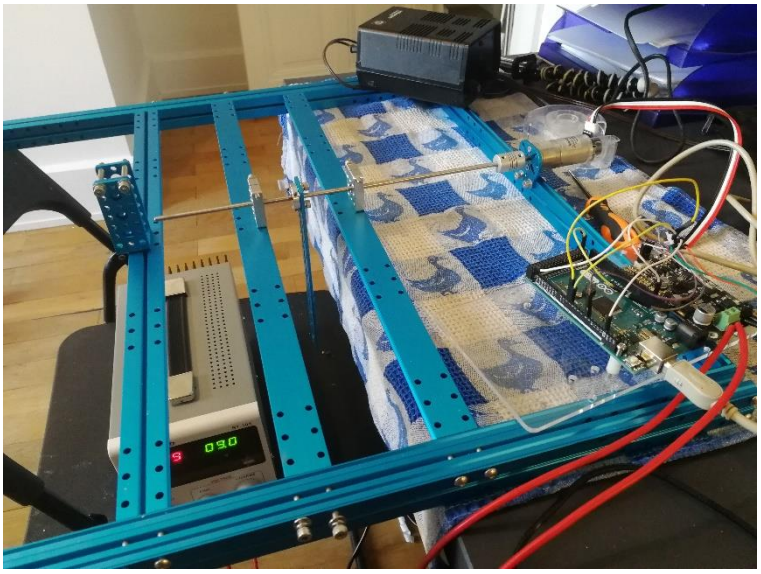
TD 2

RESEAUX DE NEURONES - IDENTIFICATION

L'objectif de ce TD est de tester et de comprendre comment marche les réseaux de neurones avec Matlab, ici NARX. Vous avez à votre disposition un cours plus complet sur les réseaux de neurones et l'approfondissement des NARX.

La problématique sera ici de proposer un modèle de comportement d'un système multiphysique par identifications temporelles. La première identification sera « classique » au sens d'une analyse causale : la deuxième se fera par un réseau de neurone de type NARX.

1 MISE EN SITUATION



Un bras manipulateur est le bras d'un robot généralement programmable, avec des fonctions similaires à un bras humain. Les liens de ce manipulateur sont reliés par des axes permettant, soit de mouvement de rotation (comme dans un robot articulé) et/ou de translation (linéaire) de déplacement.

Dans le cas d'une imitation complète d'un bras humain, un bras manipulateur a donc 3 mouvements de rotation et 3 mouvements de translation sur son élément terminal.

Il peut être autonome ou contrôlé manuellement et peut être utilisé pour effectuer une variété de tâches avec une grande précision.

Les bras manipulateurs peuvent être fixes ou mobiles (c'est-à-dire à roues) et peuvent être conçus pour des applications industrielles.

La maquette présentée ci-dessus permet de comprendre les tenants et les aboutissants d'un tel système.

2 MODELISATION PAR SCHEMA BLOC

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 :

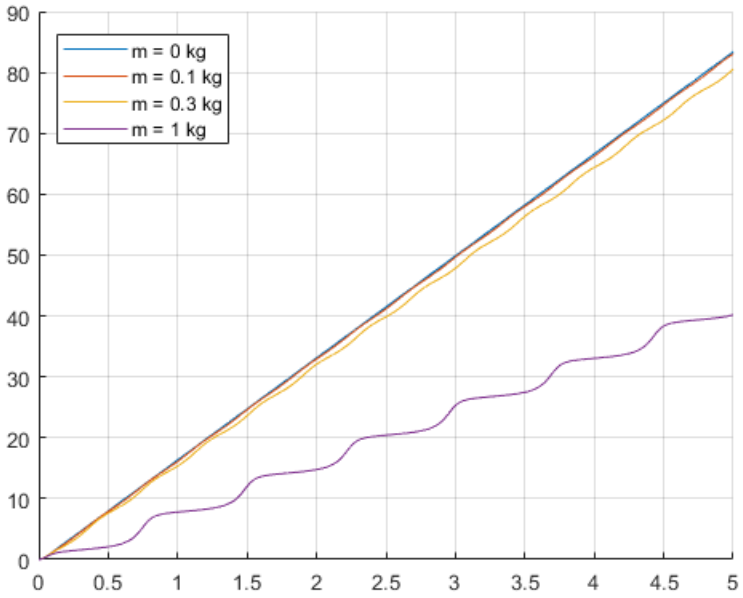
$$u(t) = L\frac{\mathrm{d}i(t)}{\mathrm{d}t} + Ri(t) + e(t) \quad (1)$$

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

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

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

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.

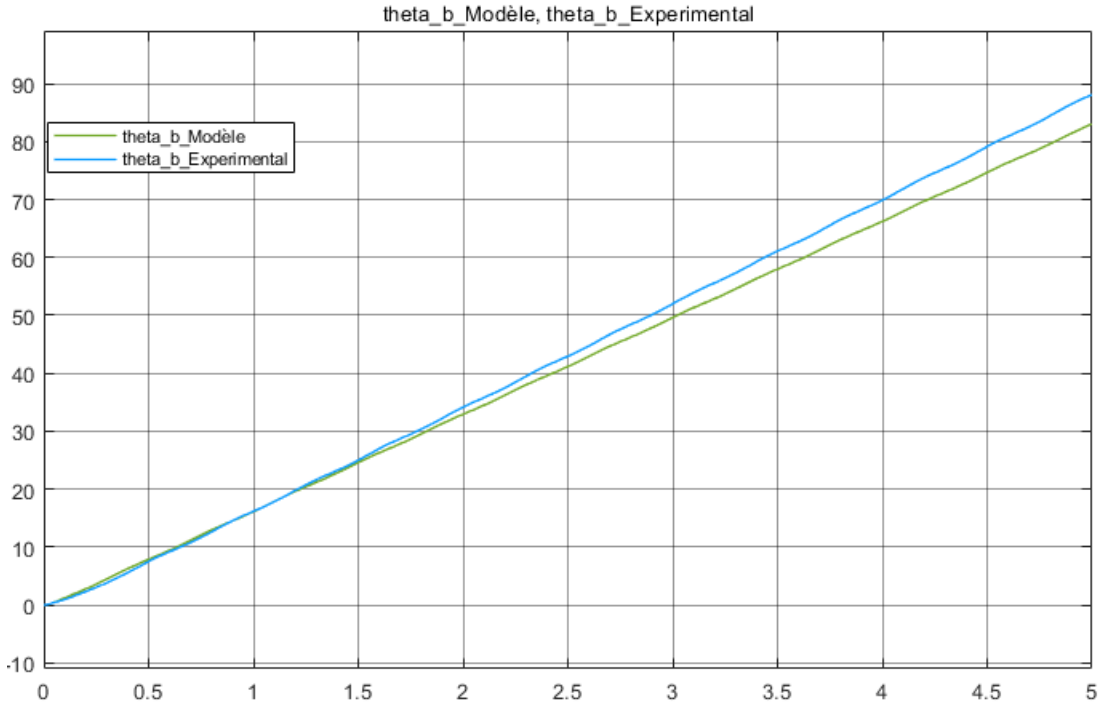
Les différentes expériences possibles à mettre en place sont :

- 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. Fichier `Activite_02_Maquette.slx`.
- 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 : pas de mesure possible ici.
- 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.

Activité 3

En exploitant le fichier `Activite_03_Maquette`, réaliser un essai en BO pour une entrée d'une tension de 9 V pendant 5 s et superposer cette réponse à la réponse simulée. En déduire les deux principaux paramètres mal évalués.



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.

3 Modélisation par un réseau de neurones

Il s'agit en préambule d'avoir lu le cours sur les réseaux de type NARX.

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.

3.1 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. Dans ce TP, la carte Arduino limite le nombre de points (espace mémoire), on se retrouve à pouvoir générer une centaine d'échelons (difficilement prévisible). Enfin, si l'on souhaite à la fin créer un asservissement sur une plage $[-\pi;\pi]$, il n'est alors pas indispensable de générer un signal de sortie qui n'est pas compris entre ces bornes. Il faut alors limiter la durée des échelons ou leurs amplitudes.
On propose le fichier `Activite_07_GenerationEchelon.m` afin de construire ce signal.
L'instruction suivante permet de générer un signal contenant 50 échelons de durée aléatoire (durée comprise entre $1\times0{,}01$ s et $20\times0{,}01$ s) et d'amplitudes alétoires comprises entre -255 et $+255$.

```
[V, S,temps_discret,pwm_discret,signe_discret,pwm_signe_discret] = Activite_07_GenerationEche
50, 1, 20)
```

On propose le fichier `Activite_07_Maquette.slx` afin de piloter le système (à valider avec le système).
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.

3.2 Entrainement

On entrainera le modèle avec deux types d'entrainement :

— régularisation de Bayes ;

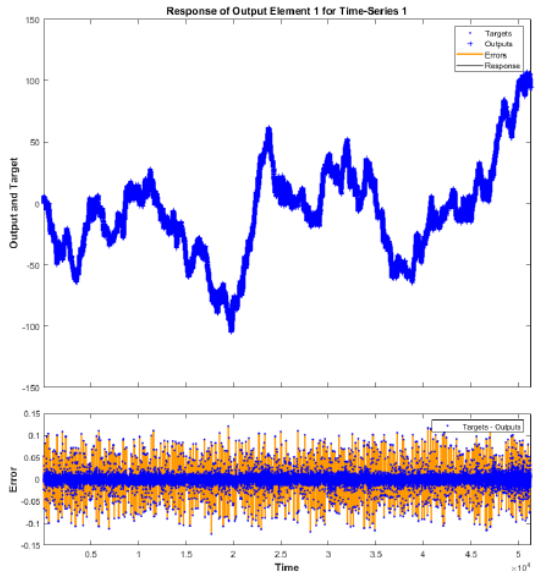
— Levenberg-Marquardt.

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.

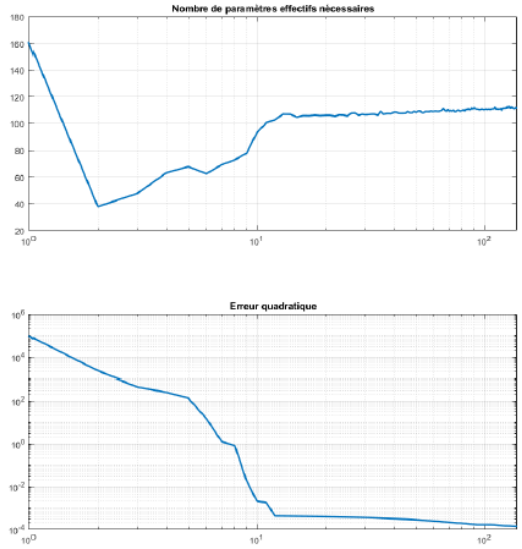
Activité 9

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

Activité 10

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

Il s'agit ici de jouer avec :

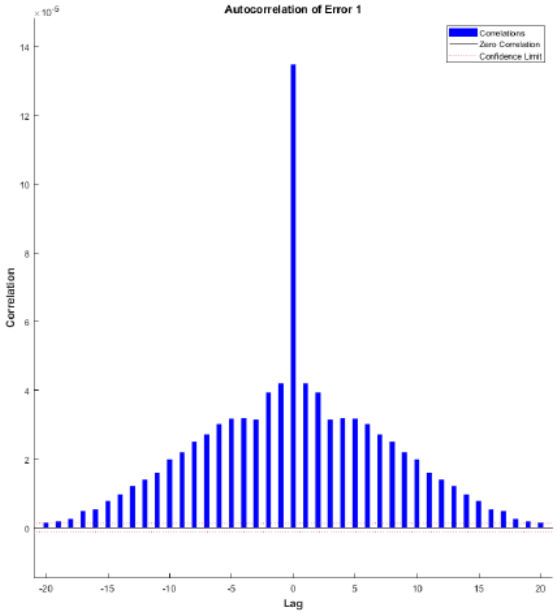
— le nombre de neurones ;

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

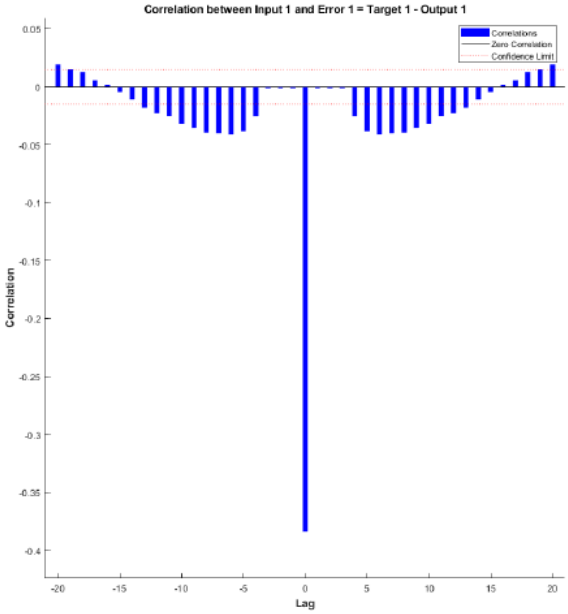
3.3 Post-entrainement

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.

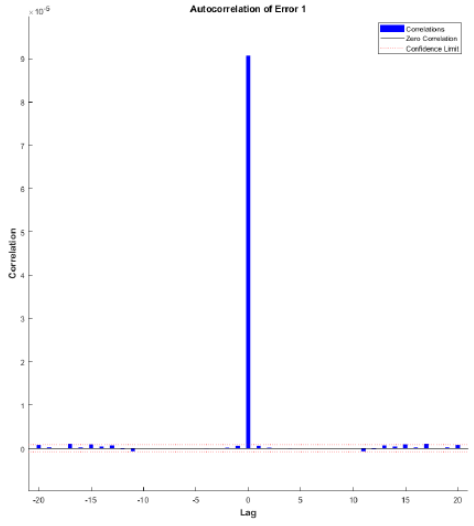


(a) Auto-corrélation initiale

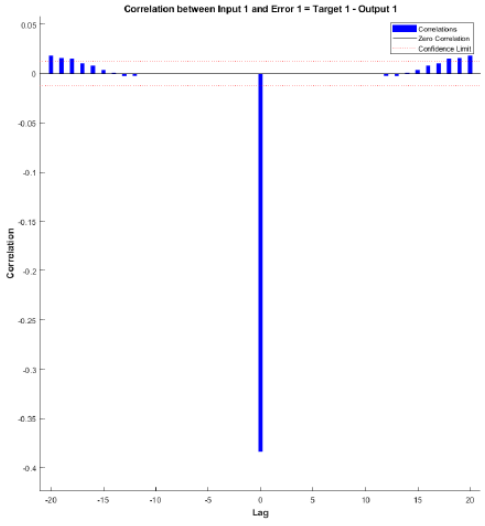


(b) Corrélation sortie entrée initiale

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.



(a) Auto-corrélation après augmentation



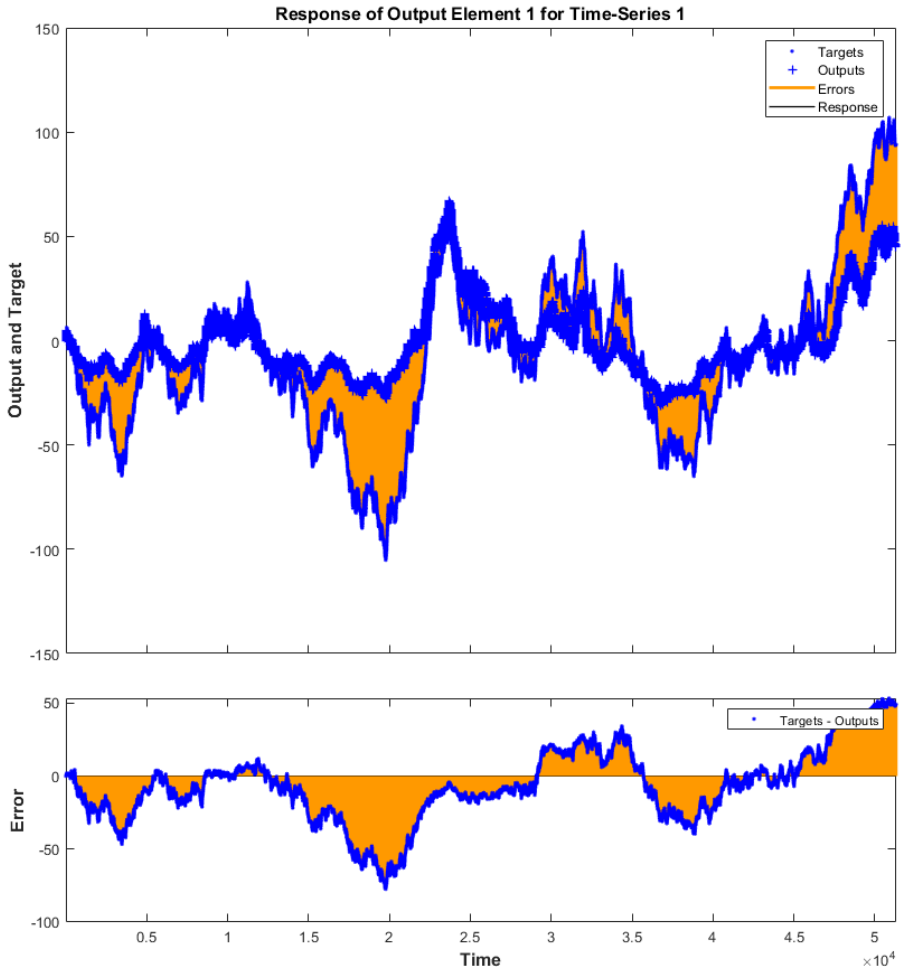
(b) Corrélation sortie entrée après augmentation

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.

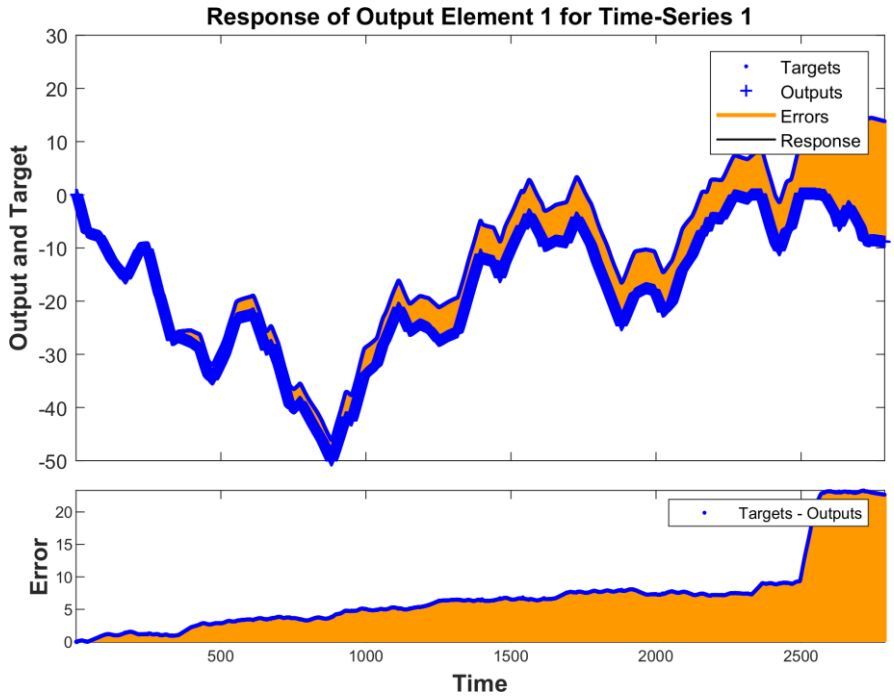


4 Conclusion

Activité 13

Tester le réseau de neurones entraîne sur un nouveau jeu de données.

Le réseau prédit correctement les nouvelles valeurs cibles pendant les premiers instants. Il accumule ensuite de l'erreur tout en ayant un comportement global acceptable.



Activité 14

Proposer une synthèse sur les avantages et les inconvénients des deux types de modélisations mises en place dans ce TP.