Commande robuste de systèmes non-linéaires basée sur un modèle hybride-physique neuronal

1er congrès annuel de la SAGIP

A. Hache, M. Thieffry, M. Yagoubi, P. Chevrel

IMT Atlantique, LS2N UMR CNRS N° 6004, Nantes, France

June 23, 2023

Sommaire

Sommaire

Contexte

Essor de l'intelligence artificielle/réseaux de neurones :

- Une augmentation du nombre de données
- Des algorithmes performants [Kingma and Ba, 2017]
- Des bons résultats pour l'identification de systèmes dynamiques [Schoukens, 2021, Rivals, 1995, Raissi et al., 2019, Revay et al., 2020]
- **Problématique :** Comment à l'aide de l'apprentissage par réseaux de neurones projeter les données sur une classe de modèles qui facilite ensuite la synthèse d'une commande robuste ?
  - appprentissage donc méthodologie de génération de données
  - modèle donc différent d'une approche purement "Data-driven" [Kergus, 2021, van Waarde et al., 2020, Martin and Allgöwer, 2021]

  Deux axes :
  - Paramétrisation
  - Intégration de contraintes "système" à prendre en compte dans l'optimisation

Sommaire

Modèle d'états neuronaux

Paramétrisation d'un modèle d'état par réseaux de neurones :

- Neural State-space models [SUYKENS et al., 1995] :

$$\begin{cases} x_{k+1} &= f_\theta(x_k, u_k) \\ y_k &= g_\theta(x_k, u_k) \end{cases} \tag{1}$$

  Quelle forme pour f_θ, g_θ ?

- **Quelle classe choisir ?**
- **A partir de la classe choisie comment orienter l'apprentissage pour la commande ?**

Introduction d'une partie linéaire explicite

- Introduire une partie linéaire donne un bon point d'initialisation. [Sjoberg, 1997][Schoukens, 2021] :

$$\begin{cases} \dot{x} &= Ax + Bu + W_x\sigma(W_{f_x}x + W_{f_u}u + b_f) + b_x \\ y &= Cx + Du + W_y\sigma(W_{g_x}x + W_{g_u}u + b_g) + b_y \end{cases} \tag{2}$$

- On peut l'initialiser grâce aux techniques linéaires telles que *best linear approximation*, [Pintelon et al., 2020] ou la méthode des sous-espaces [Van Overschee and De Moor, 1994].
- Cela permet d'inclure des dynamiques linéaires et de faciliter l'optimisation.

Quelle cible pour la commande robuste ?

- Robustesse des systèmes linéaires : champ mature.

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Gw_k \\ y_k &= C_1 x_k + D_{11} u_k + D_{12} w_k \\ z_k &= C_2 x_k + D_{21} u_k + D_{22} w_k \end{aligned} \tag{3}$$

- Indicateur de performance et de robustesse

$\Longrightarrow$ Volonté de projeter sur une classe linéarisable

Sommaire

Linéarisation exacte par retour d'état.

On considère un système affine en la commande :

$$\begin{cases} \dot{x} &= Ax + Bu + f_{NN}(x) &= f(x) + g(x)u \\ y &= Cx + g_{NN}(x) &= h(x) \end{cases} \tag{4}$$

Un problème bien étudié pour cette classe de modèles est de trouver une loi de commande :

$$u_{lin} = \alpha(x) + \beta(x)v$$

et un difféomorphisme $z = \Phi(x)$ telle que (4) soit équivalent en boucle fermée à un système linéaire :

$$\begin{cases} \dot{z} &= A_z z + B_z v \\ y &= C_z z \end{cases} \tag{5}$$

Méthodologie

De cette idée, on propose la procédure d'apprentissage suivante :
Soit un jeu de données(u_D, y_D, x_D)

1. On détermine $(A_{BLA}, B_{BLA}, C_{BLA})$ à partir d' (u_D, y_D)
2. On choisit A_{CL} pour placer les pôles de la boucle fermée à partir de (A_{BLA}, B_{BLA}) et on la fixe.
3. On paramétrise le modèle suivant (M) :

$$\begin{cases} \dot{z} = A_{CL} z + B\beta_{NN}(x_D)(u - \alpha_{NN}(x_D)) \\ y_M = Cz \end{cases} \tag{6}$$

   - α_{NN}, β_{NN} étant 2 réseaux de neurones
4. On minimise la fonction de coût $\mathcal{L}(y_M, y_D) = \frac{1}{N_B T} \sum_{i=0}^{N-1} \sum_{j=0}^{T-1} (y_D^{(i,j)} - y_M^{(i,j)})^2$
   - N_B étant la taille du batch, T la période d'intégration
   - $y^{(i,j)}$ étant l'instant j du $i^{\text{ème}}$ batch.

Identification : Silverbox

- Le jeu de données Silverbox provient d'une implémentation électronique d'un oscillateur non-linéaire. [Wigren and Schoukens, 2013]
- Il est décrit par :

$$\begin{cases} m\ddot{y} + c\dot{y} + k(y)y = u \\ k(y) = a + by^2 \end{cases}$$

  - Linéaire en u
- Donneés de nonlinearbenchmark.org
- Implémenté avec PyTorch [Forgione and Piga, 2021] et RK4 en solver d'intégration.
- Optimiseur ADAM
  - L'état interne du réseau est une variable d'optimisation.

Validation

On entraîne 2 modèles dont un qui nous servira de simulateur pour validation :

- Modèle "P" :

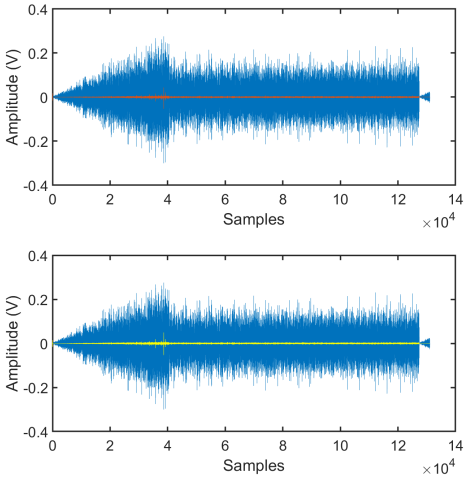
$$\begin{cases} \dot{x}_P &= Ax_P + Bu + f_{NN}(x_P) \\ y_P &= Cx_P + g_{NN}(x_P) \end{cases} \quad \text{(P)}$$

- Avec $x_D = \begin{pmatrix} y_P \\ \dot{y}_P \end{pmatrix}$
- Modèle "M":

$$\begin{cases} \dot{z} = A_{CL}z + B\beta_{NN}(x_D)^{-1}(u - \alpha_{NN}(x_D)) \\ y_M = Cz \end{cases} \quad \text{(M)}$$

- Soient λ_1, λ_2 les valeurs propres de A_{BLA}. On choisit $sp(A_{CL}) = \{-|\lambda_1|, -|\lambda_2|\}$

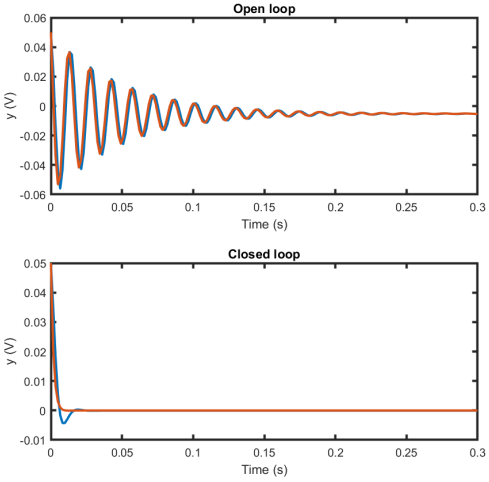
Identification : Silverbox



Résultats d'entraînement :

- Bleu : y^D.
- Rouge (haut) : erreur entre les données et le modèle de simulation P (MSE : $2.12.10^{-6}$).
- Jaune (bas) : erreur entre les données et le modèle M (MSE : $2.20.10^{-6}$).

Réponse aux conditions initiales



Comparaison entre P et M.
Haut : boucle ouverte,
Bas : boucle fermée.

- Bleu : modèle P
- Rouge : modèle M

Limites de l'exemple

Premiers résultats intéressants mais un exemple limité :

- Monovariable
- Peu non-linéaire
- Difficile par nature, de déterminer des objectifs de commande
- Linéarisable par retour d'état ("plat")

Comment étendre cette démarche à une classe de systèmes plus large ?

Sommaire

Projection sur un modèle quasi-linéarisable

On cherche donc un modèle quasi-linéarisable par retour de sortie (RF-SSNN):

$$\begin{aligned} x_{k+1} &= Ax_k + B(u_k + \alpha(y_k)) + g_n(x_k, u_k) \\ y_k &= Cx_k \end{aligned} \tag{7}$$

$$\begin{aligned} \alpha(y) &= W_h \sigma_\alpha(W_y y + b_y) + b_\alpha \\ g_n(x, u) &= W_g \sigma_g(W_x x + W_u u + b_x) + b_g \end{aligned} \tag{8}$$

Modèle non-linéaire mais comment minimiser le résidu non-linéaire $\|g_n\|$?
$\Longrightarrow$ maximiser l'explicabilité par la partie linéaire

Procédure d'apprentissage

On cherche à minimiser $\|g_n(x,u)\|$ par rapport $\|Ax+Bu\|$:

- Initialisation linéaire A_0, B_0, C_0
- $C = C_0$ pendant l'entraînement

Fonction de coût :

$$\begin{aligned} \hat{\theta} &= \arg\min J_N(\theta) \\ J_N(\theta) &= \frac{1}{N}\sum_{k=1}^{N}\left((y(k)-y_\theta(k))^2 + \gamma g_n^\theta(x(k),u(k))^2\right) \end{aligned} \tag{9}$$

Pondération de γ à définir comme un hyperparamètre.

Exemples

Illustration sur 2 benchmarks :

- Wiener-Hammerstein (donneés de nonlinearbenchmark.org)
- Proie-prédateur

$$\begin{aligned} \frac{dx_1}{dt} &= a_1 x_1 - b_1 x_1 x_2 - c_1 x_1 x_3 + d_1 u_1^2 \\ \frac{dx_2}{dt} &= a_2 x_2 - b_2 x_1 x_2 - c_2 x_1 x_3 + d_2 u_2^2 \\ \frac{dx_3}{dt} &= -e x_3 + f x_1 x_3 + g x_2 x_3 \end{aligned} \tag{10}$$

Comparaison avec une forme générale (GR-SSNN):

$$\begin{aligned} x_{k+1} &= A x_k + B u_k + f_n(x_k, u_k) \\ y_k &= C x_k \end{aligned} \tag{11}$$

Résultats d'identification

- Wiener-Hammerstein $\gamma = 1$

RMSE	RF-SSNN (This work, Eq. (7))	GR-SSNN (Eq. (11))	LTI
Training	5.3×10^{-4}	3.6×10^{-4}	2.7×10^{-2}
Test	5.5×10^{-4}	3.7×10^{-4}	2.8×10^{-2}

- Proie-prédateur $\gamma = 2$

RMSE	RF-SSNN (model (7))	GR-SSNN (model (11))	LTI
Train	9.5×10^{-2}	6.7×10^{-2}	1.9
Test	9.3×10^{-2}	6.5×10^{-2}	1.9

Résultats d'identification

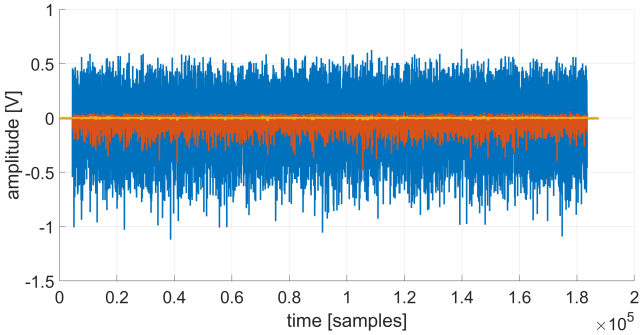


Figure: Résultats d'identification pour Wiener-Hammerstein.
Blue: données,
Red: erreur avec le LTI,
Yellow: erreur avec RF-SSNN (7).

Résultats d'identification

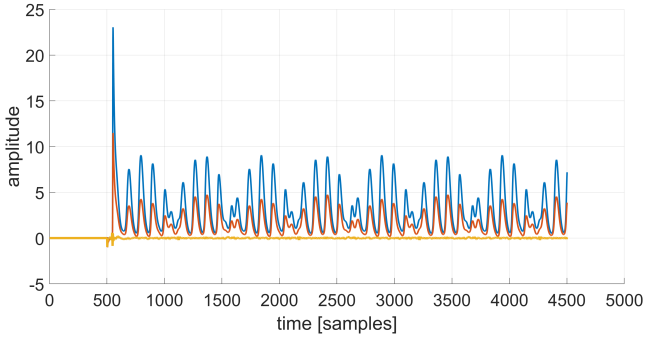


Figure: Résultats d'identification pour proie-prédateur.
Blue: données,
Red: erreur avec le LTI,
Yellow: erreur avec RF-SSNN (7).

Influence des hyperparamètres

- Influence du paramètre γ

	$\gamma = 0.01$	$\gamma = 1$	$\gamma = 100$
RMS error $(y - y_\theta)$	2.8×10^{-4}	5.5×10^{-4}	7.1×10^{-3}
$\texttt{mean}(\frac{\|g_n(x,u)\|}{\|Ax+Bu\|})$	5.6×10^{-3}	1.9×10^{-4}	1.9×10^{-5}

- Norme de la partie linéaire vs résidu (boucle fermée)

	$\|Ax + Bv\|$		$\|g_n(x, v - \alpha(y))\|$		$\frac{\|g_n(x,v-\alpha(y))\|}{\|Ax+Bv\|}$	
	max	mean	max	mean	max	mean
W.H.	8.6	1.9	5.3×10^{-3}	2.7×10^{-4}	4.1×10^{-3}	2.0×10^{-4}
P.P.	2.07	0.75	7.8×10^{-2}	1.4×10^{-2}	0.71	1.3×10^{-1}

Sommaire

Résumé et perspectives

- Résumé
  - Il existe des classes de modèles sur lesquels projeter les données par apprentissage et qui sont utiles à la commande robuste.
- Perspectives :
  - Démarche à prolonger sur un exemple fil rouge
  - Extension au cas du rejet de perturbation
  - Aller jusqu'à la synthèse et la définition du schéma de commande :

$$\begin{array}{lcl} x_{k+1} & = & Ax_k + B(u - \alpha(y_k, d_k)) + Gd_k \\ y & = & C_1 x_k \\ z_k & = & C_2 x_k + D_{21} u_k + D_{22} d_k \end{array} \tag{12}$$

  - Formalisation des objectifs de performance/robustification
  - Ajout de contraintes sur le modèle linéarisé en boucle fermée

Forgione, M. and Piga, D. (2021).
Continuous-time system identification with neural networks: Model structures and fitting criteria.
European Journal of Control, 59:69–81.

Kergus, P. (2021).
Data-driven control of infinite dimensional systems: Application to a continuous crystallizer.
In *2021 American Control Conference (ACC)*, pages 1438–1443.

Kingma, D. P. and Ba, J. (2017).
Adam: A Method for Stochastic Optimization.
arXiv:1412.6980 [cs].

Martin, T. and Allgöwer, F. (2021).
Dissipativity verification with guarantees for polynomial systems from noisy input-state data.
In *2021 American Control Conference (ACC)*, pages 3963–3968.

Pintelon, R., Schoukens, M., and Lataire, J. (2020).
Best Linear Approximation of Nonlinear Continuous-Time Systems Subject to Process Noise and Operating in Feedback.
IEEE Transactions on Instrumentation and Measurement, 69(10):8600–8612.
Conference Name: IEEE Transactions on Instrumentation and Measurement.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019).
Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.
Journal of Computational Physics, 378:686–707.

Revay, M., Wang, R., and Manchester, I. R. (2020).
A Convex Parameterization of Robust Recurrent Neural Networks.
arXiv:2004.05290 [cs, eess, math, stat].
arXiv: 2004.05290.

Rivals, I. (1995).

Modélisation et commande de processus par réseaux de neurones ; application au pilotage d'un véhicule autonome.
Theses, Université Pierre et Marie Curie - Paris VI.


Schoukens, M. (2021).
Improved Initialization of State-Space Artificial Neural Networks.
arXiv:2103.14516 [cs, eess].
arXiv: 2103.14516.


Sjoberg, J. (1997).
On estimation of nonlinear black-box models: how to obtain a good initialization.
In *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop*, pages 72–81.
ISSN: 1089-3555.


SUYKENS, J. A. K., MOOR, B. L. R. D., and VANDEWALLE, J. (1995).
Nonlinear system identification using neural state space models, applicable to robust control design.
International Journal of Control, 62(1):129–152.

Van Overschee, P. and De Moor, B. (1994).
N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems.
Automatica, 30(1):75–93.
Special issue on statistical signal processing and control.

van Waarde, H. J., Eising, J., Trentelman, H. L., and Camlibel, M. K. (2020).
Data informativity: A new perspective on data-driven analysis and control.
IEEE Transactions on Automatic Control, 65(11):4753–4768.

Wigren, T. and Schoukens, J. (2013).
Three free data sets for development and benchmarking in nonlinear system identification.
In *2013 European Control Conference (ECC)*, pages 2933–2938.