TOULOUSE INP N7

5A ModIA

Apprentissage sous contraintes physiques
Implementation of DAN

Auteur :
Rémi Colin

Superviseur :
Sixin Zhang

7 avril 2024

1 Introduction

L'objectif de ce TP est d'implémenter un Data Assimilation Network (DAN) pour modéliser un système linéaire 2D. Ce système simule un mouvement circulaire. Ce travail comprend l'implémentation d'un module gaussien, le pré-entraînement pour ajuster la distribution initiale, et un entraînement complet du réseau. Nous évaluons le modèle sur sa capacité à prédire les états futurs et son adaptation à de nouvelles données.

2 Description du modèle Physique

Le modèle physique sur lequel ce TP se base est défini par l'équation de propagation suivante :

$$\mathbf{x}_t = \mathbf{M}\mathbf{x}_{t-1} + \boldsymbol{\eta}_t \tag{1}$$

où $\mathbf{x}_t$ représente l'état du système à l'instant t, $\mathbf{M}$ est une matrice de rotation 2x2 définie comme

$$\mathbf{M} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \tag{2}$$

avec $\theta = \frac{\pi}{100}$, et $\boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \sigma_p \mathbf{I})$ représente le bruit du processus, avec $\sigma_p = 0.01$ indiquant la variance du bruit.

Les observations du système, $\mathbf{y}_t$, sont directement liées aux états par l'équation suivante :

$$\mathbf{y}_t = \mathbf{H}\mathbf{x}_t + \boldsymbol{\epsilon}_t \tag{3}$$

où $\mathbf{H} = \mathbf{I}$ est la matrice d'observation, et $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \sigma_o \mathbf{I})$ représente le bruit d'observation, avec $\sigma_o = 0.1$ étant dix fois la variance du bruit du processus.

3 Implémentation du module Gaussien

3.1 Définition de la fonction coût L_0

La fonction de perte L_0 est définie par :

$L_0(q_0^a) = \int \frac{(x_0-\mu_0^a)^T(\Lambda_0^a(\Lambda_0^a)^T)^{-1}(x_0-\mu_0^a)}{2} p(x_0)dx_0 + \log\left[|2\pi\Lambda_0^a(\Lambda_0^a)^T|^{1/2}\right]$

L'estimation de Monte-Carlo est quant à elle définie par :

$L_0(q_0^a) = \frac{1}{I}\sum_{i\leq I} \frac{(x_0(i)00-\mu_0^a)^T(\Lambda_0^a(\Lambda_0^a)^T)^{-1}(x_0(i)-\mu_0^a)}{2} p(x_0)dx_0 + \log\left[|2\pi\Lambda_0^a(\Lambda_0^a)^T|^{1/2}\right]$

avec I la taille du batch.

3.2 Amélioration de la rapidité du batch mode

Pour accélérer l'entraînement, nous avons optimisé le calcul matriciel en utilisant des opérations batch, telles que `torch.bmm` et `torch.triangular_solve`. Ces améliorations ont permis de réduire considérablement le temps nécessaire pour l'entraînement et l'évaluation du modèle.

4 Résultats sur le pretrain

4.1 Résultat sur la fonction de coût L_0

Le pré-entraînement a permis de réduire la fonction de coût L_0 jusqu'à -7.886421311891298, ce qui peut indiquer une bonne estimation initiale des états.

4.2 Moyenne et covariance de pdf a_0

La distribution des états initiaux, caractérisée par sa moyenne et sa covariance, a été ajustée avec précision au cours du pré-entraînement.
Nous avons ainsi obtenu une moyenne pdf a_0 de (3.0005, 2.9998), ce qui correspond quasiment à la moyenne empirique de x_0.

Nous avons également obtenu la matrice de covariance suivante : $\begin{bmatrix} 1.2785e-4 & -1.4395e-6 \\ -1.4395e-6 & 1.0988e-4 \end{bmatrix}$.

5 Implémentation de l'entraînement complet

5.1 Comment fixer fc zero ?

Nous avons corrigé la fonction `FcZero` en transformant les zéros initiaux en paramètres apprenables. Cette modification a permis au modèle de mieux s'ajuster lors de l'entraînement en considérant ces valeurs dans le processus d'optimisation.

5.2 Module DAN

Le module DAN a été structuré pour intégrer les phases de propagation et d'assimilation afin d'utiliser à la fois les observations et les états prédits pour ajuster ses prédictions futures comme en Assimilation de Données classique.

5.3 Fonction de coût complète

La fonction de coût complète combine les erreurs de prédiction à chaque étape, ainsi que l'erreur initiale ajustée par L_0, ce qui assure que le modèle minimise les écarts, toujours en restant fidèle à l'état initial estimé.
Ainsi, comme dans le cours, nous avons :

$$\mathcal{L}_{\text{complète}} = \frac{1}{T} \sum_{t=1}^{T} \left(L_t(q_t^b) + L_t(q_t^a) \right) + L_0(q_0^a) \tag{4}$$

où :

— T est le nombre total d'instants temporels,
— q_t^b et q_t^a représentent respectivement les distributions de probabilité à l'étape t avant et après l'assimilation des données,
— $L_t(q_t^b)$ et $L_t(q_t^a)$ sont les fonctions de perte calculées pour les distributions q_t^b et q_t^a à chaque instant t,
— $L_0(q_0^a)$ est la fonction de perte associée à la distribution initiale des états.

6 Résultats sur l'entraînement complet

6.1 Fonction de coût d'entrainement et de test log loss

Les résultats montrent une diminution continue de la fonction de coût, aussi bien pour l'entraînement que pour le test. Le modèle arrive donc bien à généraliser. On retrouve les résultats dans le tableau qui suit :

Métrique	1	150	600	1000
RMSE b	4.793114713777984	0.7465688468950695	0.04526061910656278	0.03014208224129928
RMSE a	4.764920842812691	0.32673993105986643	0.049277637550380754	0.02795331152722182
LOSS	738436.1255549039	4.339098086403024	-5.566865569615356	-7.886421311891298

TABLE 1 – Evolution de la RMSE de a et b et de la fonction de perte au cours de l'entraînement

6.2 RMSE

Les valeurs de l'erreur quadratique moyenne (RMSE) pour les prédictions a et b montrent une diminution au cours de l'entraînement.

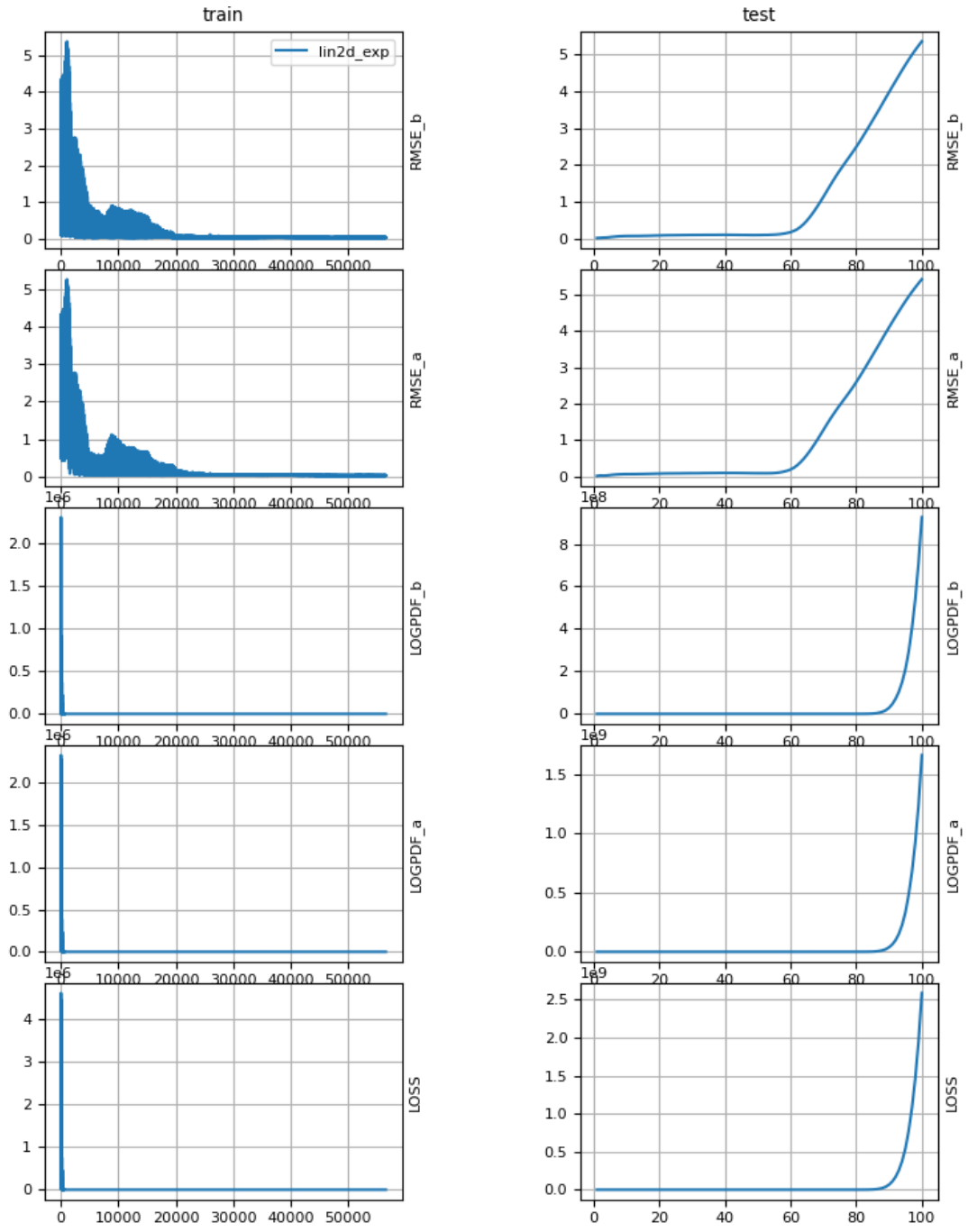

FIGURE 1 – RMSE de a et b et de la fonction de perte au cours des itérations

6.3 Plot t=T sur les données d'entrainement

Les graphiques à $t = T$ pour les données d'entraînement montrent que le modèle peut prédire fidèlement la dynamique du système linéaire 2D.

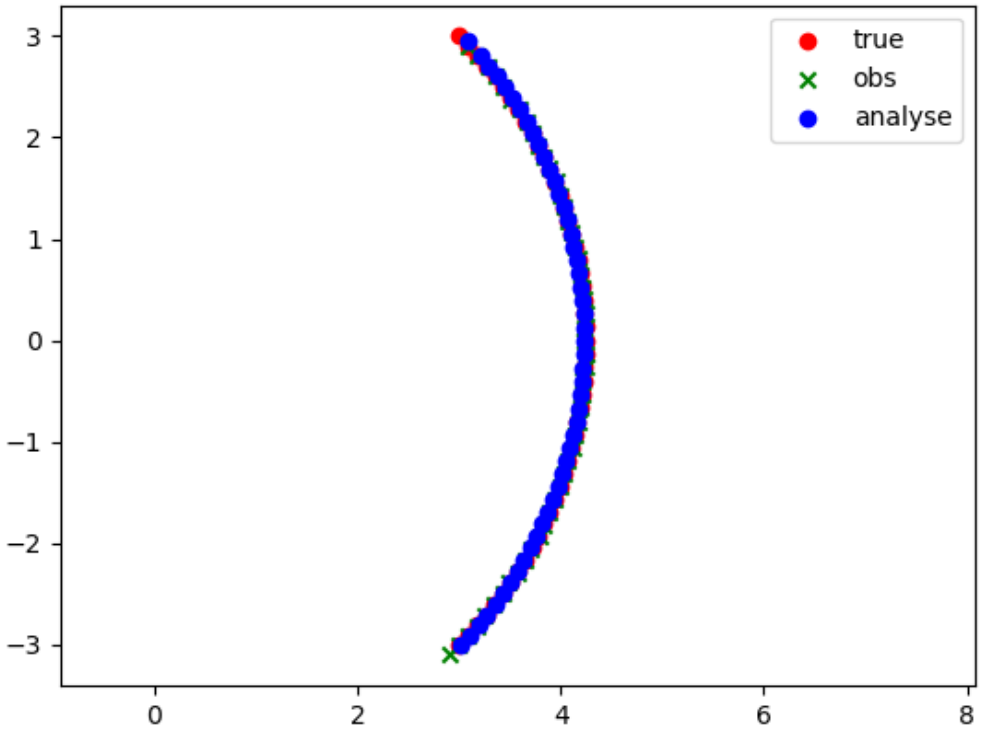

FIGURE 2 – Dynamique de x_t et y_t dans le modèle Lin2D et une des trajectoires de q_{a_t} pour $t = T$.

6.4 Plot t=2T sur les données de test

Pour $t = 2T$ sur les données de test, les graphiques montrent les limites du modèle pour de l'extrapolation. Ainsi, le modèle a du mal à prédire la trajectoire avec précision pour des temps plus longs, ce qui peut suggérer qu'il faut améliorer la généralisation (tout comme pour les modèles basiques d'assimilation de données).

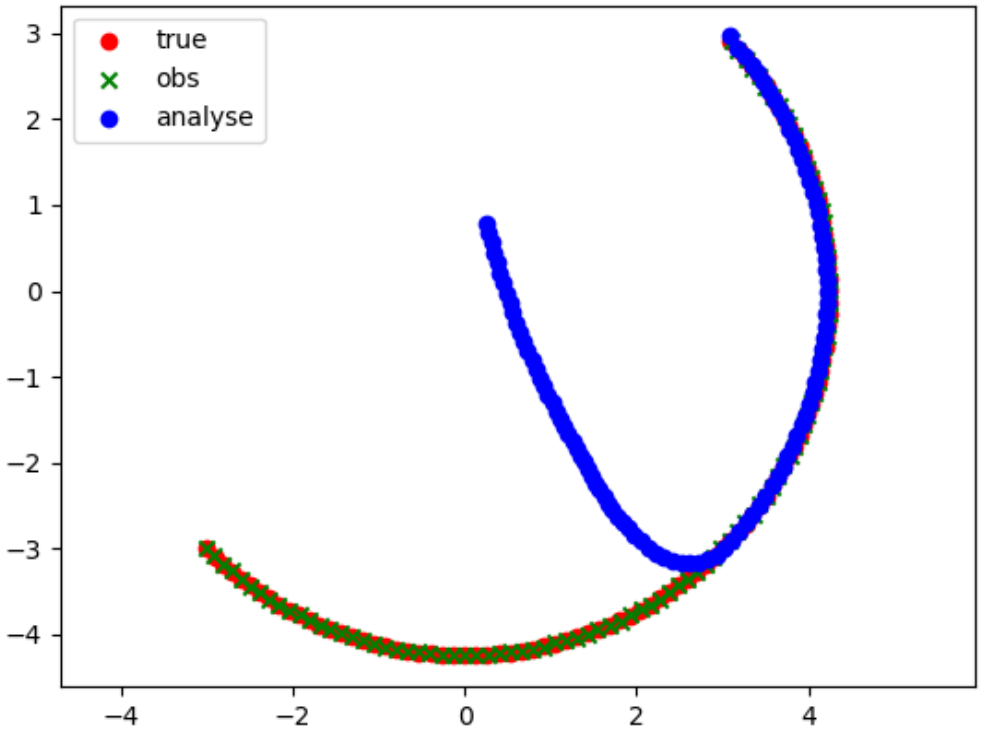

FIGURE 3 – Dynamique de x_t et y_t dans le modèle Lin2D et une des trajectoires de q_{a_t} pour $t = 2T$.

6.5 Impact de la profondeur

Les tableaux suivants montrent qu'un nombre accru de couches améliore la précision des prédictions puisque le modèle a une complexité plus grande avec plus de couches et peut donc capturer des dynamiques plus nuancées.

Deep	1	5	10	15
RMSE_b	3.7985182944482543	4.2978142323838995	4.253854905140644	4.056643075166632
RMSE_a	3.771380459671618	4.299926504037143	4.241988210751739	4.054047636542647
LOGPDF_b	413553.2841719676	486457137.90708613	2172381.026423223	734170.7958471051
LOGPDF_a	366487.52617057884	456483949.4445792	2274564.721774252	592621.9718959781
LOSS	780040.8103425464	942941087.3516653	4446945.748197475	1326792.7677430832

TABLE 2 – Scores initiaux pour différentes valeurs de deep

Deep	1	5	10	15
$RMSE_b$	0.03497577833633456	0.032159274660420054	0.030582998128623536	0.03014208224129928
$RMSE_a$	0.03234755857104206	0.03530238827146526	0.028952224123445065	0.02795331152722182
$LOGPDF_b$	-3.47995987500301	-3.6884184422360233	-3.7614310711656196	-3.8418846170110053
$LOGPDF_a$	-3.751143448665976	3.6209722588810984	-3.981993377152123	-4.044536694880293
LOSS	-7.231103323668986	-7.309390701117122	-7.743424448317743	-7.886421311891298

TABLE 3 – Scores finaux pour différentes valeurs de deep

7 Conclusion

Ce travail a présenté l'implémentation et l'entraînement d'un modèle DAN pour la prédiction d'un système dynamique linéaire 2D. Les résultats montrent que l'approche est efficace, particulièrement dans la précision des prédictions à court terme et la capacité à s'adapter à des conditions initiales variées. Cependant, la capacité du modèle à généraliser pour des temps plus longs n'est pas très bonne.