INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Machine learning under physical constraints
Implementation of DAN

Dylan Lebreton

2th year of Modelisation and Artificial Intelligence double degree
at INSA and ENSEEIHT - Toulouse, France

Supervisor: Sixin Zhang

7 April 2023

ModIA
Promotion 2024

2022 - 2023

Contents

1 Notations

$\mathbb{1}_n$ denotes the vector of ones of size n.
I_n denote the identity matrix of size n.

2 Introduction

Data assimilation is a technique used for estimating the state of dynamical systems based on noisy and partial observations. Traditional data assimilation methods, such as the Kalman filter, have been widely used for linear systems. However, for complex and nonlinear systems, these methods may not provide the desired level of accuracy and efficiency.

Data Assimilation Networks (DANs) combine the power of deep learning with traditional data assimilation techniques to improve the estimation of a system's state. In this report, we focus on the implementation of a Data Assimilation Network for the following Observed Dynamical System (ODS):

$$\text{Propagation step} : \begin{cases} x_t = Mx_{t-1} + \eta_t & (1) \\ x_0 \sim \mathcal{N}\left(3 \times \mathbb{1}_{\texttt{x\_dim}}, \sigma_0 I_{\texttt{x\_dim}}\right) & (2) \\ \eta_t \sim \mathcal{N}\left(0, \sigma_p I_{\texttt{x\_dim}}\right) & (3) \end{cases}$$

$$\text{Observation step} : \begin{cases} y_t = Hx_t + \epsilon_t & (4) \\ H = I_{\texttt{x\_dim}} & (5) \\ \epsilon_t \sim \mathcal{N}\left(0, \sigma_o I_{\texttt{x\_dim}}\right) & (6) \end{cases}$$

3 Link with Bayesian data assimilation

Rephrasing the goal of the project, we seek to estimate the state x_t of the ODS for any $t \leq T$ from the current observation y_t and the past observations $y_1, \ldots, y_{t-1}$. This corresponds mathematically to finding the following probability density: $p(x_t \mid y_1, \ldots, y_t)$.

In a Bayesian approach, two quantities are then generally defined :

- The prior, or background probability density: $p_t^b(x_t) = p(x_t \mid y_1, \ldots, y_{t-1})$
- The posterior, or analysis probability density: $p_t^a(x_t) = p(x_t \mid y_1, \ldots, y_t)$

Using Bayes rule and Markov property, we can then define these two steps:

- Analysis step: incorporation of a new observation y_t to compute the posterior p_t^a from the prior p_t^b

$$p_t^a\left(x_t \mid y_t\right) = \frac{p\left(y_t \mid x_t\right) p_t^b\left(x_t\right)}{\int p\left(y_t \mid x\right) p_t^b(x) dx}$$

- Propagation step: propagate the dynamical model to compute the next prior p_{t+1}^b from the current posterior p_t^a

$$p_{t+1}^b\left(x_{t+1}\right) = \int p\left(x_{t+1} \mid x_t\right) p_t^a\left(x_t \mid y_t\right) dx_t$$

In the case where the operators M and H are linear, the Kalman filter equations allow to realize these two steps in an optimal way. In a non-linear framework, we can introduce the use of deep learning for conditional probability estimation. This is the principle of DAN.

4 Principle of Data Assimilation Network

When the operators M and H are non-linear, we can use deep learning to approximate:

- The background probability density p_t^b by $q_t^b : (\mathbb{R}^{\texttt{H\_dim}})^{t-1} \longrightarrow \text{Prob}\,(\mathbb{R}^{\texttt{x\_dim}})$
- The analysis probability density p_t^a by $q_t^a : (\mathbb{R}^{\texttt{H\_dim}})^{t} \longrightarrow \text{Prob}\,(\mathbb{R}^{\texttt{x\_dim}})$

Remark: as $M = I_{\texttt{x\_dim}}$, we have $\texttt{H\_dim} = \texttt{x\_dim}$. The observed output is the same as the state vector.

Considering a, b and c three neural networks, we constrain the DAN to the following Markovian structure:

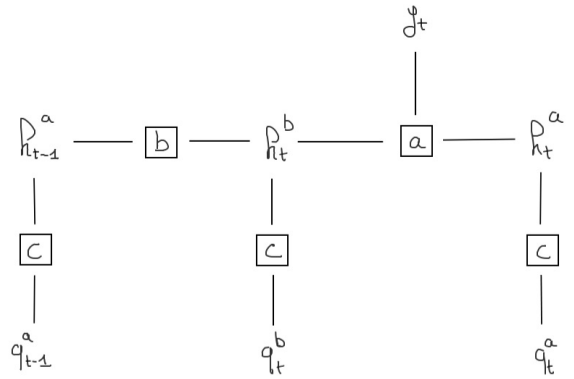

Figure 1: Chosen Markovian structure to implement DAN

In the following subsections, $\mathbb{H}$ denotes the memory space where lives h_i^j.

4.1 Analyzer a

The analyzer a assimilates new observation, it transforms the prior state in a posterior state:

$$\begin{aligned} a : \mathbb{H} \times \mathbb{R}^{\texttt{H\_dim}} &\longrightarrow \mathbb{H} \\ \left(h_t^b, y_t\right) &\longmapsto h_t^a \end{aligned}$$

In the project, it is chosen as an augmented residual network with linear layers and different possible depths.

4.2 Propagator b

The propagator b propagates the model. It is defined as:

$$\begin{aligned} b : \mathbb{H} &\longrightarrow \quad \mathbb{H} \\ h_t^a &\longmapsto h_{t+1}^b \end{aligned}$$

In the project, it is chosen as a non-linear residual network.

4.3 Procoder c

The procoder estimates the distribution of a given state. It is defined as:

$$c : \mathbb{H} \longrightarrow \text{Prob}\left(\mathbb{R}^{\texttt{x\_dim}}\right) \quad \text{such as} \quad q_t^a = c(h_t^a) \quad \text{and} \quad q_t^b = c(h_t^b)$$

In the project, it is chosen as a simple linear layer.

5 Application on period Hamiltonian dynamics

Let $x_t \in \mathbb{R}^2$ ($\texttt{x\_dim} = 2$) and $\theta = \frac{\pi}{100}$. We consider M as the following rotation matrix:

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

The chosen parameters are as follows:

- Networks batch size : 256
- Standard deviation of initial state $\sigma_0 \in \{0.01, 0.1\}$
- Standard deviation of propagation $\sigma_p = 0.01$
- Standard deviation of observations $\sigma_o = 0.1$
- Depth of analyzer and propagator depth $\in \{1, 5, 10, 20, 100\}$

5.1 Pre-training of procoder

First, in order to best estimate the density q_t^a at $t = 0$, we train the procoder alone, minimizing the loss function $\mathcal{L}_0\left(q_0^a\right)$ using the initial state x_0 and the initial hidden state $h_{t=0}^a$.

The chosen solver is LBFGS with 1000 max iterations, strong wolfe line search and a tolerance of $1e-14$. Here are some results of convergence for different depths and σ_0.

Table 1: $\mathcal{L}_0\left(q_0^a\right)$ at initial and final iteration during procoder pretaining

	depth	σ_0	number of iterations	$\mathcal{L}_0\left(q_0^a\right)$ initial	$\mathcal{L}_0\left(q_0^a\right)$ final
experiment 1	1	0.01	214	8.77	-6.46
experiment 2	1	0.1	65	16.66	-1.85
experiment 3	5	0.01	146	12.27	-6.46
experiment 4	5	0.1	76	12.29	-1.85
experiment 5	10	0.01	139	12.01	-6.46
experiment 6	20	0.01	126	22.13	-6.46
experiment 7	100	0.01	170	29.67	-6.46

We can see the convergence towards a (negative?) loss in all cases. We can also see that the higher the depth, the higher the initial loss, which is counter-intuitive because the depth only concerns the analyzer and the propagator, not the procoder. σ_0 has an impact on the number of iterations and the final loss, a higher σ_0 leads to less iterations but a higher final loss.

The figure below represents the evolution of the loss during the iterations. The curves have been plotted for the first 4 experiments and are available in the project file. We will present here only the curves of the experiment 1 and 2.

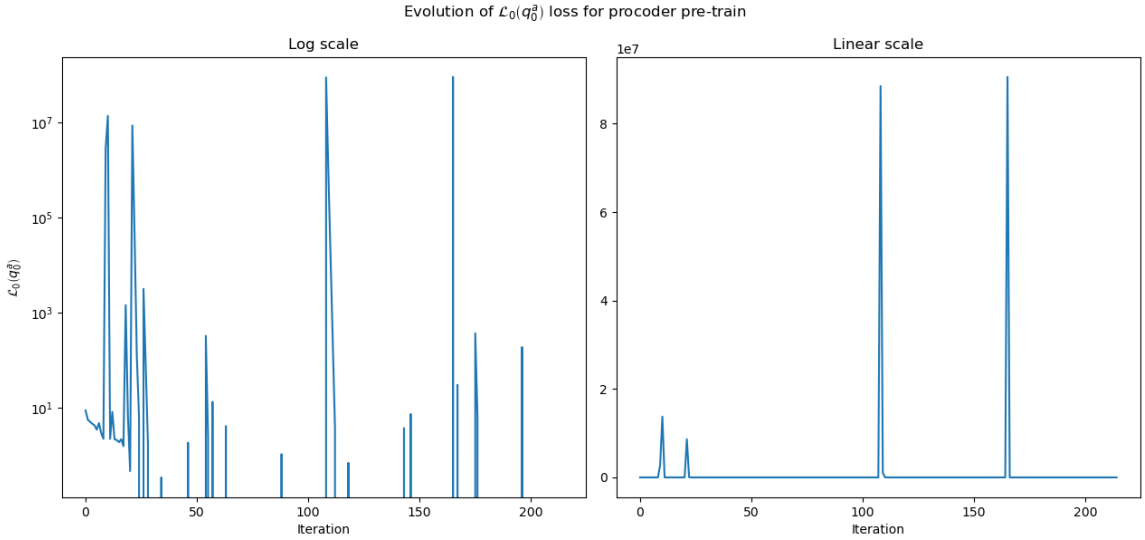

Figure 2: Evolution of $\mathcal{L}_0\left(q_0^a\right)$ for $\sigma_0 = 0.01$ and depth $= 1$ during procoder training

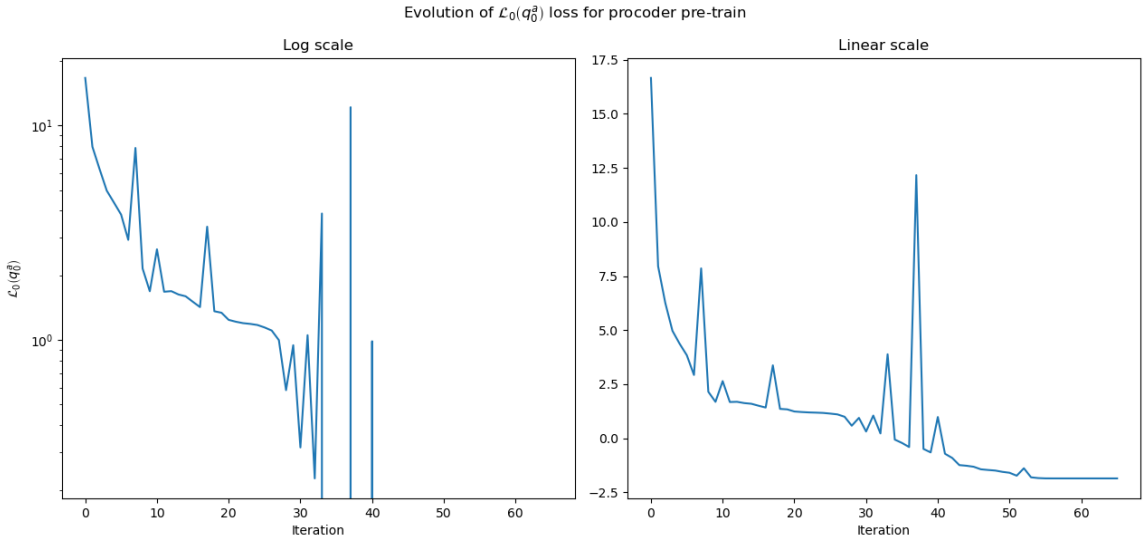

Figure 3: Evolution of $\mathcal{L}_0\left(q_0^a\right)$ for $\sigma_0 = 0.1$ and depth $= 1$ during procoder training

We can see the decrease to zero (if we do not take into account the final negative loss) in both cases. However, we notice that, for $\sigma_0 = 0.01$, the convergence is unstable and presents oscillations of amplitude $1e7$, whereas the convergence for $\sigma_0 = 0.1$ is stable, and presents oscillations of amplitude equal to 10 at the most.

5.2 Training of the DAN

Once the procoder training is done, we can start training the DAN. So we make several passes on time steps $t \in [\![0; T]\!]$ to train analyzer, propagator and procoder to transform priors into posteriors, posteriors into priors, and priors and posteriors into densities. The loss minimized during this training includes the three components of the DAN and is equal to:

$$\frac{1}{T}\sum_{t=1}^{T}\left[\mathcal{L}_t\left(q_t^a\right) + \mathcal{L}_t\left(q_t^b\right)\right] + \mathcal{L}_0\left(q_0^a\right)$$

During this training, some metrics (called scores) are recorded:

- RMSE of the analyzer and propagator
- Loss functions of the analyzer ($\mathcal{L}_t\left(q_t^a\right)$) and propagator ($\mathcal{L}_t\left(q_t^b\right)$)
- Total loss function

The state of these scores during the initial iteration and the final iteration for the different experiments is available in the following table.

Table 2: Scores during training of the DAN for different experiments

experiment number	1	2	3	4	5	6	7
depth	1	1	5	5	10	20	100
σ_0	0.01	0.1	0.01	0.1	0.01	0.01	0.01
number of iterations	1147	1134	1134	1146	1117	1159	1121
training time (hours)	0.84	0.59	0.87	0.86	1.21	1.98	7.64
initial analyzer RMSE	0.43	0.24	0.21	0.22	0.58	0.67	1.47
final analyzer RMSE	0.05	0.05	0.05	0.05	0.05	0.05	0.05
$\mathcal{L}_t\left(q_t^a\right)$ initial	3470252	4.36	97247	5.37	6193338	683	102112
$\mathcal{L}_t\left(q_t^a\right)$ final	-3.12	-3.16	-3.13	-3.18	-2.97	-3.26	-3.24
initial propagator RMSE	0.09	0.12	0.09	0.12	0.09	0.09	0.09
final propagator RMSE	0.06	0.05	0.06	0.05	0.06	0.06	0.06
$\mathcal{L}_t\left(q_t^b\right)$ initial	86.07	-0.92	86.07	-0.92	86.07	86.07	86.07
$\mathcal{L}_t\left(q_t^b\right)$ final	-2.98	-3.07	-2.99	-3.08	-2.85	-3.04	-3.06
initial total loss	3470338	3.44	97333	4.44	6193424	769.07	102198
final total loss	-6.09	-6.23	-6.12	-6.26	-5.82	-6.3	-6.3

We notice at first, without surprise, that the training time increases considerably when the depth of the networks used for the analyzer and the propagator increases, going from less than one hour to 7 hours.

We notice that the initial losses (of the analyzer, of the propagator and total) are always much lower for $\sigma_0 = 0.1$ than for $\sigma_0 = 0.01$, which is counter-intuitive, and may suggest an error in the code. Nevertheless, the initial loss is always higher than the final loss, generally going, for $\sigma_0 = 0.01$, from an order of magnitude of power of 10 to a value close to zero. This indicates that despite everything, we succeeded in reducing the losses during training. We also see that the RMSEs decrease, but remain of the same order of magnitude as the initial RMSEs.

The figure below represents the evolution of the scores during the iterations. As for pre-training, the curves have been plotted for the first 4 experiments and are available in the project file. We will present here only the curves of the experiment 3 (depth = 5 and $\sigma_0 = 0.01$).

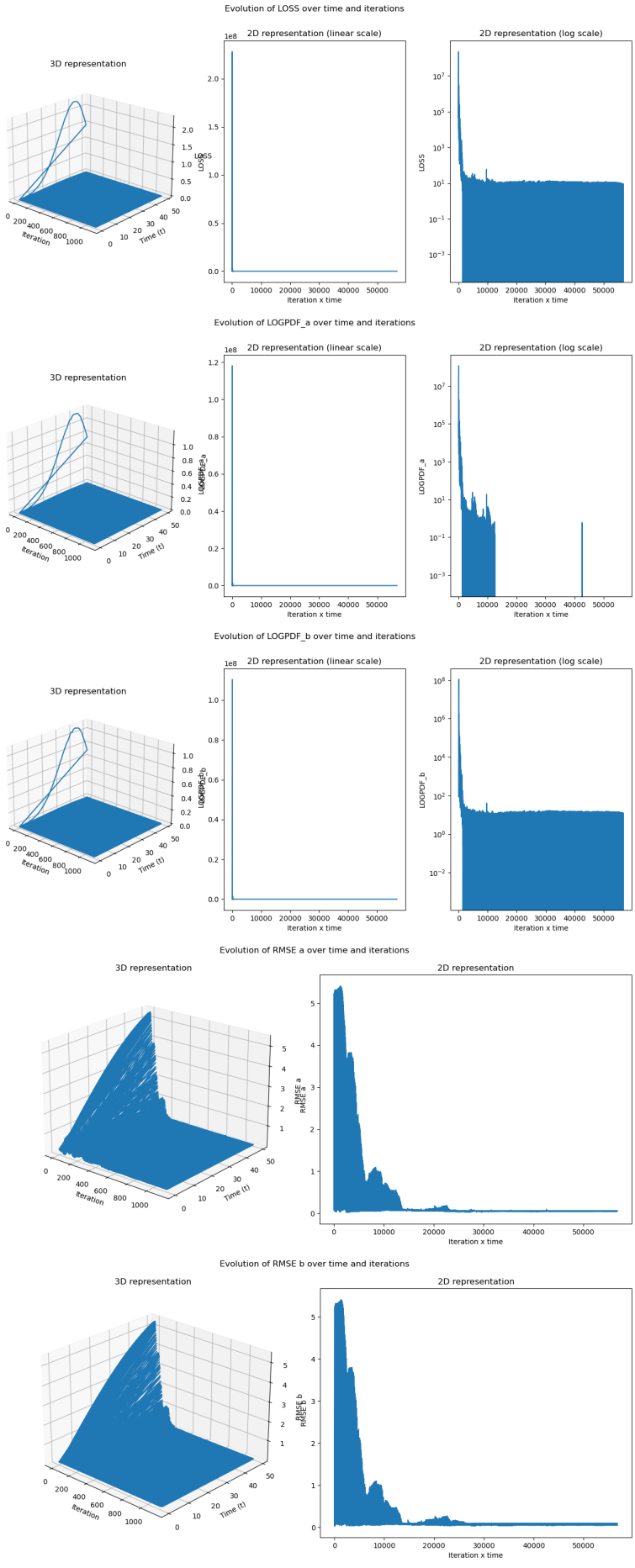

Figure 4: Evolution of scores during DAN training

We can clearly see a decrease in most scores on the training games. For the RMSEs, we see that the RMSEs are large for high t, but also decreases as the iterations progress.

This convergence of the scores is also confirmed by the following graphs, which represent the average of the density estimated from the posteriors q_t^a with the observations and the values of the real model.

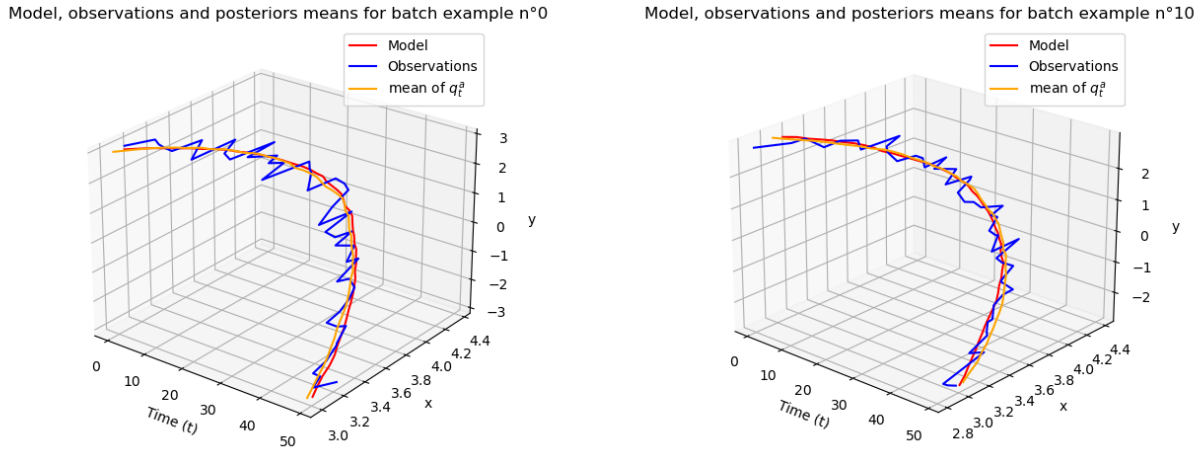

Figure 5: Evolution of model, observations and mean of q_t^a over time on train data

We see that, despite moderately noisy observations, the DAN provides a very good approximation of the model. But given that DAN was trained on this same dataset, the results are not surprising. In the next section, we test the DAN on a test dataset.

5.3 Test of the DAN

As for training data, the next table shows the scores on the test data for different depths and σ_0. The test set is derived from the same model as the training set, but with a different initial prior x_0 and a model and observations over twice as long.

Table 3: Scores of the DAN on test data for different experiments

experiment number	1	2	3	4	5	6	7
depth	1	1	5	5	10	20	100
σ_0	0.01	0.1	0.01	0.1	0.01	0.01	0.01
initial analyzer RMSE	0.05	0.09	0.05	0.09	0.05	0.05	0.04
final analyzer RMSE	0.05	0.04	0.05	0.05	0.05	0.08	0.07
$\mathcal{L}_t\left(q_t^a\right)$ initial	-3.12	-1.7	-3.53	-1.66	-3.7	-3.78	-3.93
$\mathcal{L}_t\left(q_t^a\right)$ final	82.64	2.29	275.09	2.45	5762.63	2688.82	5302.54
initial propagator RMSE	0.09	0.13	0.09	0.13	0.09	0.09	0.09
final propagator RMSE	0.14	0.07	0.09	0.07	0.08	0.08	0.08
$\mathcal{L}_t\left(q_t^b\right)$ initial	10.52	-0.6	13.0	-0.59	11.03	11.64	13.89
$\mathcal{L}_t\left(q_t^b\right)$ final	419.76	4.32	126.57	1.63	50339.73	224.21	580.65
initial total loss	7.41	-2.3	9.47	-2.24	7.34	7.86	9.97
final total loss	502.4	6.62	401.66	4.09	56102.36	2913.04	5883.19

There is no significant difference on the RMSE, nevertheless, we can see that the final losses are higher than the initial losses, showing that the algorithm has more difficulties on a data set on which it has not been trained.

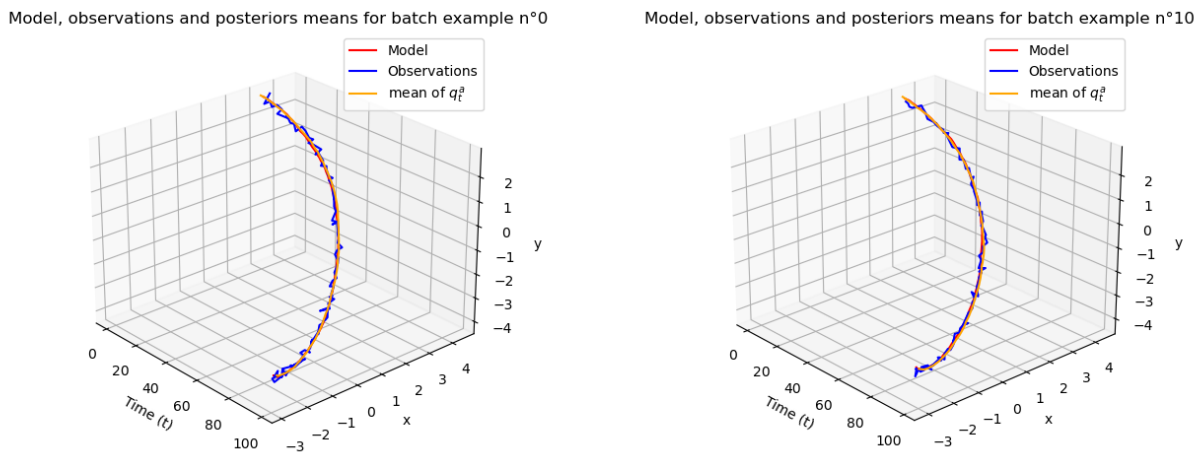

Figure 6: Evolution of model, observations and mean of q_t^a over time on test data

Despite the previous results, we notice that the DAN manages to predict correctly the model trajectory with noisy observations. This can be explained by the fact that, despite a different initial prior, the model is linear and therefore simpler to predict than a non-linear model. An increase in the different standard deviations could have shown more limitations of the DAN.

6 Conclusion

Following our project, we were able to deepen our understanding of the implementation of DAN. The Hamiltonian dynamics experiment showed the ability of the DAN to model a simple system, even on data which has not been used for training.

As a further step, one could think of varying with more finesse the different standard deviations to see the impact on the training and the results of the DAN. On the other hand, in a practical case such as weather forecasting, it would have been difficult to obtain as much data as those used in the batch to train the DAN. A possible extension would be to study the impact of the batch size on the DAN and its results.