TP: Introduction to DAN

1 Prepare dataset for DAN

The goal is to implement the propagation and observation steps in two observed dynamical systems. In both cases, we assume

- Propagation step : $x_t = Mx_{t-1} + \eta_t$
  - η_t is Gaussian white noise $\mathcal{N}(0, \sigma_p I)$
- Observation step : $y_t = Hx_t + \epsilon_t$
  - Identity case : $H = I$
  - ϵ_t is Gaussian white noise $\mathcal{N}(0, \sigma_o I)$

1.1 Linear 2d : periodic Hamiltonian dynamics

Let $x_t \in \mathbb{R}^2$ and $\theta = \pi/100$. Implement the module Lin2d in the code filters.py, with the following 2×2 rotation matrix,

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

Note the that state x_t is stored in a batch form whose size is $mb \times 2$. This allows one to process mb simulations in parallel.

• Initialize x_0 by using the function get_x0 in manage_exp.py. Choose $\sigma = \sigma_0$. The parameters $\sigma_0, \sigma_p, \sigma_o$ are given in lin2d_exp.py

• Make a figure to show the dynamics of x_t for $t \leq 50$, staring from a random initialization of x_0. Use $mb = 2$ to show two simulations.

1.2 Integration with DAN

Use the code of DAN in `code_ elevesMoodle.zip`

Based on the dynamical operator and x_0 which you have just implemented, we shall now build the propagator and observer in order to generate x_t and y_t. They are constructed in the following way (in the function experiment),

```
prop = filters.ConstructorProp(**prop_kwargs)
obs = filters.ConstructorObs(**obs_kwargs)
```

• Preparation : first test the code of DAN on your machine by running

```
python main.py --save lin2d_exp.py --run
```

• The M operator is implemented in the module Lin2d in the file filters.py. In order to generate x_{t+1}, you will sample a Gaussian distribution $\mathcal{N}(Mx_t, \sigma_p I)$. To generate mb samples, we store mb points of $x_t \in \mathbb{R}^n$ in a matrix whose size is $mb \times n$.

• In the Lin 2d case, make a figure to show the dynamics of y_t for $t \leq 50$, staring from a random initialization of x_0. Use $mb = 2$ to show two simulations.

1.3 (Optional) Lorentz 40d : non-linear Chaotic dynamics

Let $x = (x[1], \cdots x[40]) \in \mathbb{R}^{40}$. Each state variable $x[i]$ of the Lorentz system is evolved under the following ODE :

$$\frac{dx[i]}{dt} = (x[i+1] - x[i-2])x[i-1] - x[i] + F$$

This ODE is then discretized to generate x over $t \geq 0$. More information can be found in `https://en.wikipedia.org/wiki/Lorenz_96_model`, e.g. $F = 8$.

• Implement the function edo, which compute $\frac{dx[i]}{dt}$ for $1 \leq i \leq 40$ at any given state x, in the module EDO of filters.py.

• Make an image plot of the dynamics of x_t for $t \leq 50$, starting from $x[i] = 0, i \leq 40$. Use $mb = 1$ to show one simulation.

• Reproduce Figure 1 to generate chaotic dynamics.

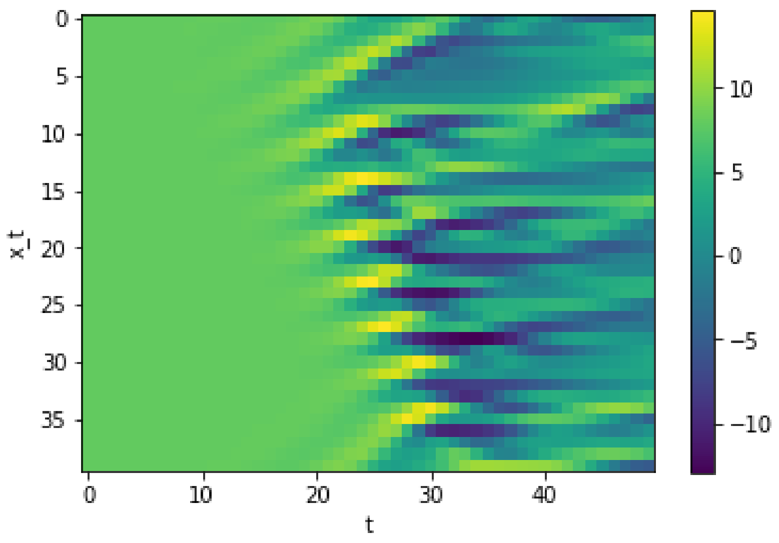

Figure 1 – The dynamics of x_t in Lorentz 40d, starting from $x[1] = F + 0.01$ and $x[i] = F$ for $i > 1$