Part I Intro

① Parametric PDE in Data driven dynamics

→ $\dot{u} = \underset{\sim}{N}(u, x, t, \mu)$ where $u(x,t)$ is the solution.

μ ← Parameter

N ↑ unknown, learned by NN (Neural Network)

Q: How to learn $N(\cdot)$?

→ By training. Therefore we need training data

- synthetic data : generated data when governing equations are known ; less noisy
- experimental data : observed data ; noisier

In synthetic data, we usually call it snapshots. Often,

snapshots: training data, test data, validation data

Synthetic data is time and spatially discretized (of a continuous solution)

i.e.) $u(x,t) = u(t_k) \in \mathbb{R}^{N_h}$ for $k = 1, \cdots, N_t$

N_h ← Dimension of Data (high); N_t ← # of time steps

※: In machine learning community, spatial coordinates ≈ sensors (In this paper, sensors are selected spatial coordinates to reduce its total number)

$$
\begin{array}{c}
\text{Space } 1 \\ \text{Space } 2 \\ \text{Space } 3 \\ \vdots \\ \text{Space } N_h
\end{array}
\begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_{N_t} \\ | & | & & | \end{bmatrix} = X \in \mathbb{C}^{N_h \times N_t}
$$

⟶ Time

If $u(x,t)$ is parametrized,

$$
X = \left[\begin{bmatrix} | & & | \\ u_1 & \cdots & u_{N_t} \\ | & & | \end{bmatrix}, \begin{bmatrix} | & & | \\ u_1 & \cdots & u_{N_t} \\ | & & | \end{bmatrix}, \cdots \begin{bmatrix} | & & | \\ u_1 & \cdots & u_{N_t} \\ | & & | \end{bmatrix}_{N_p} \right] \in \mathbb{C}^{N_h \times N_t N_p}
$$

N_p ← # of parameters

② ROM (Reduced Order Modeling)

1) POD (Proper Orthogonal Decomposition)

→ This is in fact, SVD.

$X = \Psi \Sigma V^*$; we assume that all the variables of $U_k^{\mu_p}$ is "not" independent but the snapshot matrix X is a lower rank.

← collection of all the temporal solution

; the true, latent dynamics reside in linear subspace.

$\Psi \in \mathbb{C}^{N_h \times r}$ ✓, $\Sigma \in \mathbb{R}^{r \times r}$, $V^* \in \mathbb{C}^{r \times N_t N_p}$

Then $u \approx \Psi a$ where a is the reduced state. (The governing PDE evolution $a(t, \mu)$ is in the r-rank subspace spanned by Ψ).

$u \in \mathbb{C}^{N_h \times 1}$, $\Psi \in \mathbb{C}^{N_h \times r}$, $a \in \mathbb{C}^{r \times 1}$

⇒ Ψ can be obtained easily by SVD of the snapshots. Therefore, if we learn $a(t, \mu)$, we can get the solution $u(x,t)$.

Q: How to learn $a(x,t)$?

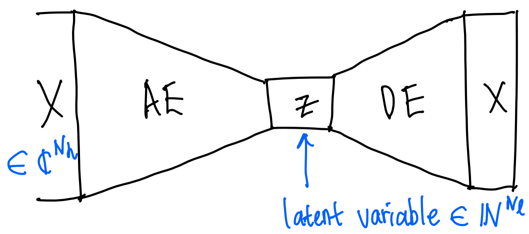
→ We predict $a_{k+1}^{\mu} = f_{\underset{\sim}{\theta}}(a_k^{\mu})$ using NN ; "one step prediction"

θ ↑ weights and biases

pros) easy and simple, results are interpretable

cons) linear subspace may not be optimal for latent dynamics

2) Autoencoder – Decoder

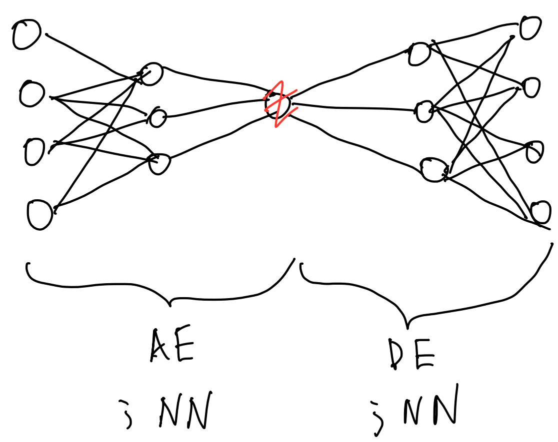


; latent dynamics reside in nonlinear manifold

↳ $\dot{z} = f(z, t, \mu)$

⇒ We train AE – DE by minimizing the reconstruction error, i.e., $\| X_{before} - X_{after} \| = \| X_{before} - DE(AE(X_{before})) \|$

$\approx \| X_{before} - \underbrace{D_\theta}_{NN} (\underbrace{A_\theta}_{NN} (X_{before})) \|$



; z is learned by one-step prediction.

i.e., $z_{k+1} = f(z_k, \mu)$

Part II Shallow Recurrent Decoder – Based ROM

preliminary) Separation of variable for PDEs : $u(x, t, \mu) = T(t, \mu) X(x, \mu)$

If we transform a PDE to spectral expansion, $u(x,t) = \sum_{n=1}^{N} a_n(t) \phi_n(x)$ ··· $\boxed{1}$

Plugging $\boxed{1}$ back to $\dot{u} = N(u)$ yields N coupled ODEs for $a_n(t)$: $\dot{a}_n = f_n(a_1, a_2, \cdots, a_N)$

for $n = 1, \cdots, N$.

→ Then discretized u is, $u(x,t) \approx u(x_s, t_k) = \sum_{n=1}^{N} a_n(t_k) \phi_n(x_s)$ for $\begin{cases} k = 1, \cdots, N_t \\ s = 1, \cdots, N_h \end{cases}$

Since, $a_n(0) = \langle \underbrace{u_0(x)}_{\text{measurable}}, \phi_n(x) \rangle$, we can solve $\dot{a}_n = f_n(a_1, a_2, \cdots, a_N)$

by forward method – iterative integration –

and minimize $\| \underbrace{u(x_s, t_k)}_{\text{included in the snapshots}} - \sum a_n(t_k) \phi_n(x_s) \|^2$

where $\phi_n(x)$ is the n-th column vector of $\phi(x)$ from $\psi(x)$ of SVD of X.

The structure of SHRED

In practice, we set this value and tune by the performance of the model.

X | LSTM (Long short time memory) $f_T : \underbrace{\mathbb{R}^{N_s} \times \cdots \times \mathbb{R}^{N_s}}_{L \text{ times}} \to \mathbb{R}^{N_\ell}$ | $h_k{}^{\mu} \in \mathbb{R}^{N_\ell} = f_T({}S_{k-L}{}^{\mu}, \cdots, S_k{}^{\mu})$ learned by NN | SHRED $f_X : \mathbb{R}^{N_\ell} \to \mathbb{R}^{N_h}$ | learned solution $\hat{u}_k{}^{\mu} \in \mathbb{R}^{N_h} = f_X(h_k{}^{\mu}) \approx u_k{}^{\mu}$ (learned by NN; True Solution)

X : sensor measurements over last $L_{\le N_t}$ time steps ; temporal history of sensor values.

Sensor 1 $\quad ({}^1S_{k-L}{}^{\mu}, {}^1S_{k-(L-1)}{}^{\mu}, \cdots, {}^1S_k{}^{\mu})$

Sensor 2 $\quad ({}^2S_{k-L}{}^{\mu}, {}^2S_{k-(L-1)}{}^{\mu}, \cdots, {}^2S_k{}^{\mu})$

$\vdots$

Sensor N_s $\quad ({}^{N_s}S_{k-L}{}^{\mu}, {}^{N_s}S_{k-(L-1)}{}^{\mu}, \cdots, {}^{N_s}S_k{}^{\mu})$

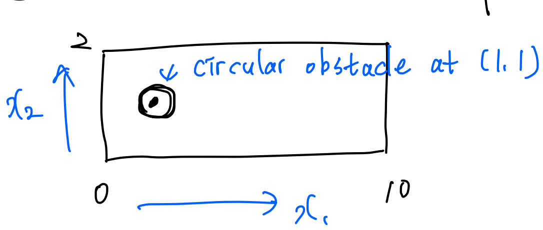
※ N_s is not nessarily equal to N_h.
They indeed select a few sensors and set $N_s < N_h$.

Note that the separation of variables doesn't work for all kinds of PDEs, where the general use of this architecture is limited. However, they show heuristically with substantial experiments that this can be widly used and helpful for varieties of PDEs.

Part III Numerical Experiment (5 in total)

E) Flow Around an obstacle (high dim, non-linear, parametric PDE)

① Problem set-up



, $\mu = [\alpha_{in}, \gamma_{in}, r_l, r_r]^T$

flow angle of attack (α_{in}), Inlet velocity (γ_{in}), Left obstacle extension (r_l), right obstacle extension (r_r)

→ The flow of the fluid is described by 2D Navier Stokes equations

① $\begin{cases} V_t - \nu \Delta t + (v \cdot \nabla) v + \nabla p = 0 \\ \nabla \cdot v = 0 \end{cases}$ → Governing interior dynamics of the domain

② I.C : $V(x, 0) = (\overset{V_1}{0}, \overset{V_2}{0})$; velocities in direction of x_1, x_2

③ B.C : $V((0,x_2),t) = (r_{in}\cos(\alpha_{in}),\ \underbrace{x_2}_{\text{locations}}(2-x_2)\, r_{in}\sin(\alpha_{in}))$

② Generate the snapshots for 40,296 nodes and obtain horizontal velocities V_1 and vertical velocities V_2.

$$
X = \left[\underbrace{\left[\begin{bmatrix} V_{11} \\ V_{12} \\ \vdots \\ V_{1\,40296} \\ \hline V_{21} \\ V_{22} \\ \vdots \\ V_{2\,40296} \end{bmatrix}_{t=0} \begin{bmatrix} V_{11} \\ V_{12} \\ \vdots \\ V_{1\,40296} \\ \hline V_{21} \\ V_{22} \\ \vdots \\ V_{2\,40296} \end{bmatrix}_{\underset{0.05\,sec}{dt}} \cdots \begin{bmatrix} V_{11} \\ V_{12} \\ \vdots \\ V_{1\,40296} \\ \hline V_{21} \\ V_{22} \\ \vdots \\ V_{2\,40296} \end{bmatrix}_{T = 10\,\text{seconds}} \right]_{\mu=1}}_{200\ \text{time steps}} \cdots \left[\begin{bmatrix} V_{11} \\ V_{12} \\ \vdots \\ V_{1\,40296} \\ \hline V_{21} \\ V_{22} \\ \vdots \\ V_{2\,40296} \end{bmatrix}_{t=0} \begin{bmatrix} V_{11} \\ V_{12} \\ \vdots \\ V_{1\,40296} \\ \hline V_{21} \\ V_{22} \\ \vdots \\ V_{2\,40296} \end{bmatrix}_{t=dt} \cdots \begin{bmatrix} V_{11} \\ V_{12} \\ \vdots \\ V_{1\,40296} \\ \hline V_{21} \\ V_{22} \\ \vdots \\ V_{2\,40296} \end{bmatrix}_{t=T} \right]_{\mu = N_p \,\sim\, 200\ \text{scenarios}} \right]
$$

$\in \mathbb{R}^{80592 \times 40000}$

$$= \quad \Sigma \quad V^*$$

$\Psi \in \mathbb{R}^{80,592 \times r}$ ← at most 40,000

$\Psi = \Psi[:, :150]$ ← columnwise truncation by the latent dim, N_l that we want.

$\in \mathbb{R}^{80592 \times 150}$

} truncated POD Basis

Then, $u = \Psi \underbrace{a}_{}$ $\longleftrightarrow$ $a = \Psi^T u$

we are going to learn this and predict!

③ Construct training data from snapshots

→ We choose 3 spatial locations (sensors) and 50 time steps from each sensor.

	LSTM		SHRED	
X	$f_T : \underbrace{\mathbb{R}^{N_s} \times \cdots \times \mathbb{R}^{N_s}}_{L \text{ times}} \to \mathbb{R}^{N_\ell}$	$h_k^{\mu} \in \mathbb{R}^{N_\ell} = f_T(S_{k-L}^{\mu}, \cdots, S_k^{\mu})$	$f_X : \mathbb{R}^{N_\ell} \to \mathbb{R}^{N_h}$	$\hat{u}_k^{\mu} \in \mathbb{R}^{N_h} = f_X(h_k^{\mu}) \approx u_k^{\mu}$

Fixed, implicitly given by data

Does not change with time.

$$X = S\_history = \begin{bmatrix} [S_{k-50}, S_{k-49}, \cdots, S_k ; \mu], \\ [S_{k-50}, S_{k-49}, \cdots, S_k ; \mu], \\ [S_{k-50}, S_{k-49}, \cdots, S_k ; \mu], \end{bmatrix} \overset{NN}{\Rightarrow} \underbrace{h_k^{\mu}}_{\text{Latent variable}} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N_\ell} \end{bmatrix}_{N_\ell \le 150} \overset{NN}{\Rightarrow} a(t) = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{150} \end{bmatrix} \in \mathbb{R}^{150 \times 1} \Rightarrow u(x,t) = \psi(x) a(t)$$

L2 error : 6.63%

VS

POD Reconstruction error: 1.16%

$$\left(\left\| \frac{u - \psi \psi^T a}{u} \right\| \right)$$

In fact, they set the dimension of h as 64 in their implementation.