Deep learning and uncertain dynamical systems

Tenavi Nakamura-Zimmerer    Daniele Venturi    Qi Gong

Applied Mathematics Department
University of California, Santa Cruz

SIAM Conference on Computational Science and Engineering
February 25, 2019
Nonlinear control system with random initial condition (note that this framework *includes parameter uncertainty*):

\[ \dot{x} = f(x, u), \quad x(0) = x_0 \sim p_0(x_0), \quad t \in [0, t_f]. \]

\(u(x, t) \in \mathcal{U} \subset \mathbb{R}^m\) is the deterministic control, \(x_0 \sim p_0(x_0)\) is the random initial condition which causes uncertainty in the state \(x(t; x_0) \in \mathbb{R}^n\).
Nonlinear control system with random initial condition (note that this framework *includes parameter uncertainty*):

\[
\dot{x} = f(x, u), \quad x(0) = x_0 \sim p_0(x_0), \quad t \in [0, t_f].
\]

\(u(x, t) \in \mathcal{U} \subset \mathbb{R}^m\) is the deterministic control, \(x_0 \sim p_0(x_0)\) is the random initial condition which causes uncertainty in the state \(x(t; x_0) \in \mathbb{R}^n\).

Want to design an *optimal control* which is *robust to uncertainty* by solving, for example,

\[
\min_u \mathbb{E}\{F(x(t_f))\} = \int F(x)p(x, t_f)dx
\]
Applications

(De)synchronization of neural oscillators:

*From Monga et al. ACC 2018.
**From Peng and Li JGCD 2017.
Applications

(De)synchronization of neural oscillators:

*From Monga et al. ACC 2018.

Trajectory planning for swarms of small satellites:

**From Peng and Li JGCD 2017.
Existing methods (e.g. Phelps et al. SICON 2016, Ross et al. JGCD 2015) rely on massive amounts of sampling and numerical integration from the initial PDF.
Existing methods (e.g. Phelps et al. SICON 2016, Ross et al. JGCD 2015) rely on massive amounts of sampling and numerical integration from the initial PDF.

$\Rightarrow$ **Curse of dimensionality**, typically limited to just a few uncertain variables.
Existing methods (e.g. Phelps et al. SICON 2016, Ross et al. JGCD 2015) rely on massive amounts of sampling and numerical integration from the initial PDF.

⇒ **Curse of dimensionality**, typically limited to just a few uncertain variables.

We propose an alternative approach to the forward UQ problem based on combining *PDF equations* with *deep learning*.
The PDF $p(x, t)$ for a dynamical system $\dot{x} = f(x, u)$ with random initial condition $x(0) = x_0 \sim p_0(x_0)$ is known to satisfy the Liouville equation:

$$\frac{\partial p(x, t)}{\partial t} + \nabla_x \cdot [p(x, t)f(x, u)] = 0.$$
The Liouville equation

The PDF $p(x, t)$ for a dynamical system $\dot{x} = f(x, u)$ with random initial condition $x(0) = x_0 \sim p_0(x_0)$ is known to satisfy the **Liouville equation**:

$$\frac{\partial p(x, t)}{\partial t} + \nabla_x \cdot [p(x, t)f(x, u)] = 0.$$

**Analytical solution:**

$$p(x, t) = p_0(\Phi_0(x, t; u)) \exp\left(-\int_0^t \nabla_x \cdot f(\Phi(x_0, \tau; u), u) d\tau\right),$$

where

$$\Phi(x_0, t; u) \quad \text{and} \quad \Phi_0(x, t; u)$$

are the forward and inverse **flow maps** for a given control $u$. 
If there is an analytical solution, isn't this problem already solved?
If there is an analytical solution, isn't this problem already solved?

No! The analytical solution requires the flow map $\Phi(x_0, t)$ and inverse flow map $\Phi_0(x, t)$.

However, the analytical solution makes computing the density at individual points straightforward.
If there is an analytical solution, isn't this problem already solved?

No! The analytical solution requires the flow map \( \Phi(x_0, t) \) and inverse flow map \( \Phi_0(x, t) \).

However, the analytical solution makes computing the density at individual points straightforward.

\( \Rightarrow \) Suggests an approach which combines data and knowledge of physics (Liouville PDE).
If there is an analytical solution, isn’t this problem already solved?

No! The analytical solution requires the flow map $\Phi(x_0, t)$ and inverse flow map $\Phi_0(x, t)$.

However, the analytical solution makes computing the density at individual points straightforward.

$\Rightarrow$ Suggests an approach which combines data and knowledge of physics (Liouville PDE).

$\Rightarrow$ Alternatively, develop efficient data-driven approximators of the forward and inverse flow maps.
“Physics informed neural nets” (PINNs) augment training data by fitting the PDE at collocation points (Raissi et al. (arXiv 2017)).
“Physics informed neural nets” (PINNs) augment training data by fitting the PDE at collocation points (Raissi et al. (arXiv 2017)).

Automatic differentiation $\frac{\partial \hat{p}}{\partial t}$ and $\nabla_x \hat{p}$, the partial derivatives of the NN predictions $\hat{p}(x, t) \approx p(x, t)$. 

$0 = \frac{\partial p(x, t)}{\partial t} + \nabla_x [p(x, t)f(x, t)]$. 

Define the residual $R(\hat{p})$ as $R(\hat{p}(x, t)) := \frac{\partial \hat{p}(x, t)}{\partial t} + \nabla_x [\hat{p}(x, t)f(x, t)]$. 

Quantifies how well predictions $\hat{p}$ fit the Liouville equation.
“Physics informed neural nets” (PINNs) augment training data by fitting the PDE at collocation points (Raissi et al. (arXiv 2017)).

Automatic differentiation $\frac{\partial \hat{p}}{\partial t}$ and $\nabla_x \hat{p}$, the partial derivatives of the NN predictions $\hat{p}(x, t) \approx p(x, t)$.

For the true probability $p(x, t)$,

$$0 = \frac{\partial p(x, t)}{\partial t} + \nabla_x \cdot [p(x, t) f(x, t)].$$
“Physics informed neural nets” (PINNs) augment training data by fitting the PDE at collocation points (Raissi et al. (arXiv 2017)).

Automatic differentiation $\rightarrow \partial \hat{p}/\partial t$ and $\nabla_x \hat{p}$, the partial derivatives of the NN predictions $\hat{p}(x, t) \approx p(x, t)$.

For the true probability $p(x, t)$,

$$0 = \frac{\partial p(x, t)}{\partial t} + \nabla_x \cdot [p(x, t) f(x, t)].$$

$\Rightarrow$ Define the residual $R(\hat{p})$ as

$$R(\hat{p}(x, t)) := \frac{\partial \hat{p}(x, t)}{\partial t} + \nabla_x \cdot [\hat{p}(x, t) f(x, t)].$$

$\Rightarrow$ Quantifies how well predictions $\hat{p}$ fit the Liouville equation.
Define a new loss metric for training the neural net:

\[
\text{loss}_{\text{PINN}}(\theta) := \text{loss}_p(\theta) + \mu \text{loss}_L(\theta).
\]

\(\theta := \{W_j, \mathbf{b}_j\}_{j=1}^M\) are the parameters of the NN, \(\mu\) is a scalar weight,
PDF approximation with physics informed neural nets

Define a new loss metric for training the neural net:

\[
\text{loss}_{\text{PINN}}(\theta) := \text{loss}_p(\theta) + \mu \text{loss}_L(\theta).
\]

\[\theta := \{\mathbf{W}_j, \mathbf{b}_j\}_{j=1}^M\]

are the parameters of the NN, \(\mu\) is a scalar weight, \(\text{loss}_p\) is the loss with respect to the training data (we use weighted MSE on log probability):

\[
\text{loss}_p(\theta) = \frac{1}{N_d} \sum_{i=1}^{N_d} p(x^{(i)}, t) \left[ \log \hat{p}(x^{(i)}, t) - \log p(x^{(i)}, t) \right]^2,
\]
Define a new loss metric for training the neural net:

$$\text{loss}_{\text{PINN}}(\theta) := \text{loss}_p(\theta) + \mu \text{loss}_L(\theta).$$

$\theta := \{W_j, b_j\}_{j=1}^M$ are the parameters of the NN, $\mu$ is a scalar weight, $\text{loss}_p$ is the loss with respect to the training data (we use weighted MSE on log probability):

$$\text{loss}_p(\theta) = \frac{1}{N_d} \sum_{i=1}^{N_d} p \left( x^{(i)}, t \right) \left[ \log \hat{p} \left( x^{(i)}, t \right) - \log p \left( x^{(i)}, t \right) \right]^2,$$

$\text{loss}_L$ measures how well predictions $\hat{p}$ fit the Liouville equation on a set of $N_c$ collocation points (we use MSE):

$$\text{loss}_L(\theta) = \frac{1}{N_c} \sum_{i=1}^{N_c} \left[ R \left( \hat{p} \left( x^{(i)}, t^{(i)} \right) \right) \right]^2.$$
Training physics informed neural nets

\[(x, t) \rightarrow \text{NN}_p \rightarrow \hat{p}(x, t) \rightarrow \text{loss}_p \rightarrow p(x, t) \rightarrow \text{loss}_{\text{PINN}} \rightarrow R(\hat{p}(x, t)) \rightarrow \text{loss}_\mathcal{L} \rightarrow \text{update NN parameters} \]
Prediction with physics informed neural nets

prediction with trained NN

\((x, t)\) \rightarrow NN_p \rightarrow \hat{p}(x, t) \rightarrow loss_p

\(\partial \hat{p}/\partial t, \nabla_x \hat{p}\) \rightarrow R(\hat{p}(x, t)) \rightarrow loss_{\mathcal{L}}

\(p(x, t)\)

\(\text{LOSS}_{\text{PIN}}\)
Example: Van der Pol oscillator

\[
\begin{aligned}
\dot{x} &= y \\
\dot{y} &= (1 - x^2)y - x,
\end{aligned}
\]

\[p_0(x, y) = \mathcal{N}_x(0, 0.25) \mathcal{N}_y(0, 0.25), \]

\[t \in [0, 4].\]

Plot courtesy of wikipedia.
PDF data generation
Forward sampling

1. Sample $N_s$ points $\{x_0^{(i)}\}$ from the initial PDF $p_0(x)$. 

![Initial PDF samples (t = 0.0)](image-url)
PDF data generation
Forward sampling

1. Sample $N_s$ points $\{x_0^{(i)}\}$ from the initial PDF $p_0(x)$.
2. Numerically integrate to get the state $\{x^{(i)}(t_k)\}$ at times $\{t_k\} \subset [0, t_f]$.
PDF data generation
Forward sampling

1. Sample $N_s$ points $\{x_0^{(i)}\}$ from the initial PDF $p_0(x)$.

2. Numerically integrate to get the state $\{x^{(i)}(t_k)\}$ at times $\{t_k\} \subset [0, t_f]$.

3. Compute density by analytical Liouville solution,

$$p(x, t) = p_0(\Phi_0(x, t; u)) \exp \left( - \int_0^t \nabla_x \cdot f(\Phi(x_0, \tau; u), u) \, d\tau \right)$$
“Convex hull stratification algorithm” from Ziegelmeier et al. (SIAM REV 2017) to construct an approximate boundary of the trajectory tube/PDF support at final time (generally highly non-convex).

2 Weight data by its distance to the boundary.
“Convex hull stratification algorithm” from Ziegelmeier et al. (SIAM REV 2017) to construct an approximate boundary of the trajectory tube/PDF support at final time (generally \textit{highly non-convex}).

2 Weight data by its distance to the boundary.

3 Sample new data points near the boundary and integrate backwards to time $t_0$. 
PDF data generation

PDF data (t = 0.0)

PDF data (t = 4.0)
Original PINNs optimize without any weight on the PDE penalty term, using L-BFGS (full-batch Quasi-Newton)…
Original PINNs optimize without any weight on the PDE penalty term, using L-BFGS (full-batch Quasi-Newton)...

Physics is only *encouraged* by a single penalty
Easy progressive batch L-BFGS

Original PINNs optimize without any weight on the PDE penalty term, using L-BFGS (full-batch Quasi-Newton)...

⇒ Physics is only *encouraged* by a single penalty

⇒ Ideally, force $R(\hat{\rho}(\mathbf{x}, t)) \to 0$ using *constrained optimization*. *Difficult in practice...*
Original PINNs optimize without any weight on the PDE penalty term, using L-BFGS (full-batch Quasi-Newton)...

⇒ Physics is only *encouraged* by a single penalty

⇒ Ideally, force \( R(\hat{\rho}(\mathbf{x}, t)) \to 0 \) using *constrained optimization*. *Difficult in practice*...

⇒ **Simple idea:** Train in multiple rounds:

- Each round, increase the penalty weight \( \mu \)
- Train on random subsets of the collocation and training data; increase the size of the subsets over the training rounds
Results

Van der Pol oscillator

\[
\begin{aligned}
\dot{x} &= y, \\
\dot{y} &= (1 - x^2)y - x,
\end{aligned}
\]

\[p_0(x, y) = \mathcal{N}_x(0, 0.25)\mathcal{N}_y(0, 0.25),
\]

\[t \in [0, 4].
\]

- 554 training trajectories evaluated at \(N_t = 160\) time snapshots
- Normalized root mean square error measured with respect to 133 test trajectories not seen during training

<table>
<thead>
<tr>
<th>optimizer</th>
<th>PDE</th>
<th>training time</th>
<th>test NRMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>full batch L-BFGS</td>
<td>-</td>
<td>210 s</td>
<td>7.17 e−03</td>
</tr>
<tr>
<td>progressive L-BFGS</td>
<td>-</td>
<td>123 s</td>
<td>5.27 e−03</td>
</tr>
<tr>
<td>full batch L-BFGS penalty</td>
<td>penalty</td>
<td>509 s</td>
<td>3.74 e−03</td>
</tr>
<tr>
<td>progressive L-BFGS penalty</td>
<td>penalty</td>
<td>106 s</td>
<td>4.75 e−03</td>
</tr>
<tr>
<td>full batch IPOPT</td>
<td>-</td>
<td>117 s</td>
<td>2.54 e−02</td>
</tr>
<tr>
<td>progressive IPOPT constrained</td>
<td>constrained</td>
<td>480 s</td>
<td>6.86 e−03</td>
</tr>
</tbody>
</table>
Approximated PDF $\hat{p}(x, y, t)$ of the Van der Pol equation. Each frame is $500 \times 500$; predicting all 1.5 million outputs takes about $\sim 0.06$ seconds.
\[ \dot{x} = y, \quad \dot{y} = -\delta y - x (\alpha + \beta x^2) + \gamma \cos(\omega t). \]

Means: \( \mu_x = 0, \mu_y = 0, \mu_\delta = 0.5, \mu_\alpha = -1, \mu_\beta = 1, \mu_\omega = 1, \mu_\gamma = 0.5 \)

Variances: \( \sigma_x^2 = \sigma_y^2 = 1, \quad \sigma_\delta^2 = \sigma_\alpha^2 = \sigma_\beta^2 = \sigma_\omega^2 = \sigma_\gamma^2 = 0.25^2 = 0.0625 \)

- 1186 training trajectories evaluated at \( N_t = 77 \) time snapshots
- 283 test trajectories

<table>
<thead>
<tr>
<th>optimizer</th>
<th>PDE</th>
<th>training time</th>
<th>test NRMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>full batch L-BFGS</td>
<td>-</td>
<td>465 s</td>
<td>2.95 e–03</td>
</tr>
<tr>
<td>progressive L-BFGS</td>
<td>-</td>
<td>399 s</td>
<td>3.13 e–03</td>
</tr>
<tr>
<td>full batch L-BFGS</td>
<td>penalty</td>
<td>1586 s</td>
<td>2.01 e–03</td>
</tr>
<tr>
<td>progressive L-BFGS</td>
<td>penalty</td>
<td>802 s</td>
<td>4.17 e–03</td>
</tr>
</tbody>
</table>
Approximated PDF \( \hat{p}(x, y, t|\delta = 0.5, \alpha = -1, \beta = 1, \omega = 1, \gamma = 0.5) \) of the uncertain forced Duffing equation for mean parameter values. Predicting all 1.5 million outputs below takes about \( \sim 0.12 \) seconds.
We also consider the problem of approximating the flow map and inverse flow map:

\[ \hat{\Phi}(x_0, t) \approx \Phi(x_0, t), \quad \hat{\Phi}_0(x, t) \approx \Phi(x, t). \]
We also consider the problem of approximating the flow map and inverse flow map:

\[ \hat{\Phi}(x_0, t) \approx \Phi(x_0, t), \quad \hat{\Phi}_0(x, t) \approx \Phi(x, t). \]

**Two interpretations of the flow map:**

1. “Step” (short-time) flow map with RNN, studied in e.g. Lusch *et al.* (Nat. Commun. 2018): \( \hat{\Phi}(x(t), t) \approx x(t + \Delta t). \)
Flow map approximation

We also consider the problem of approximating the flow map and inverse flow map:

\[ \hat{\Phi}(x_0, t) \approx \Phi(x_0, t), \quad \hat{\Phi}_0(x, t) \approx \Phi(x, t). \]

Two interpretations of the flow map:

1. “Step” (short-time) flow map with RNN, studied in e.g. Lusch et al. (Nat. Commun. 2018): \( \hat{\Phi}(x(t), t) \approx x(t + \Delta t) \).

2. “Jump” (long-time) flow map with PINN: \( \hat{\Phi}(x_0, t) \approx x(t) \).
Three degree-of-freedom fixed-wing UAV

Model introduction

\[
\begin{align*}
\dot{x} &= v \cos \gamma \cos \sigma \\
\dot{y} &= v \cos \gamma \sin \sigma \\
\dot{z} &= v \sin \gamma \\
\dot{v} &= \frac{1}{m} (-D + T \cos \alpha) - g \sin \gamma \\
\dot{\gamma} &= \frac{1}{mv} (L \cos \mu + T \cos \mu \sin \alpha) - \frac{g}{v} \cos \gamma \\
\dot{\sigma} &= \frac{1}{mv \cos \gamma} (L \sin \mu + T \sin \mu \sin \alpha) \\
\dot{T} &= u_T \\
\dot{\alpha} &= u_\alpha \\
\dot{\mu} &= u_\mu \\
\dot{C}_{x_0} &= \dot{C}_{xa} = \dot{C}_{z_0} = \dot{C}_{za} = 0
\end{align*}
\]

Lift: \( L = \frac{1}{2} \rho v^2 SC_L \)

Drag: \( D = \frac{1}{2} \rho v^2 SC_D \)

Air pressure: \( \rho = 1.21 e^{-z/8000} \)

Wing area: \( S = 0.982 \)

\[
\begin{align*}
C_L &= (C_{x_0} + C_{xa} \alpha) \sin \alpha - (C_{z_0} + C_{za} \alpha) \cos \alpha \\
C_D &= - (C_{x_0} + C_{xa} \alpha) \cos \alpha - (C_{z_0} + C_{za} \alpha) \sin \alpha
\end{align*}
\]
Use a control designed for the deterministic system.
Three degree-of-freedom fixed-wing UAV

Results

Use a control designed for the deterministic system, add uncertainty in everything!!!
Use a control designed for the deterministic system, **add uncertainty in everything!!!**

200 sample trajectories for training, 50 for testing. Test error measured using mean relative $L_2$

<table>
<thead>
<tr>
<th>approximator</th>
<th>training time</th>
<th>test error</th>
</tr>
</thead>
<tbody>
<tr>
<td>forward flow map</td>
<td>373 s</td>
<td>3.81 e–04</td>
</tr>
<tr>
<td>inverse flow map</td>
<td>220 s</td>
<td>1.39 e–03</td>
</tr>
</tbody>
</table>
Three degree-of-freedom fixed-wing UAV
Results

Forward flow map data reconstructions

- true
- reconstruction
If we need $p(x, t)$ in high dimensions and high resolution, training and evaluating a NN can be more efficient than Monte Carlo sampling or solving Liouville PDE.
If we need $p(x, t)$ in high dimensions and high resolution, training and evaluating a NN can be more efficient than Monte Carlo sampling or solving Liouville PDE.

- No control yet 😞
Conclusions

- If we need $p(x, t)$ in high dimensions and high resolution, training and evaluating a NN can be more efficient than Monte Carlo sampling or solving Liouville PDE.

- No control yet 😞 ... but potential new tool for uncertainty quantification and evaluation of control robustness given limited sample data.
This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.


