Sandia National Laboratories

Test Cases for a Translating Hyperbolic Function

Simone Venturi, Tiernan Casey

Extreme-Scale Data Science & Analytics (8739)

Part of the Code Documentation for Neural Networks for Reduced Order Modeling (ROMNet)

U.S. DEPARTMENT OF ENERGY | National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

A Translating Hyperbolic Tangent Test Case

Equations of motion:

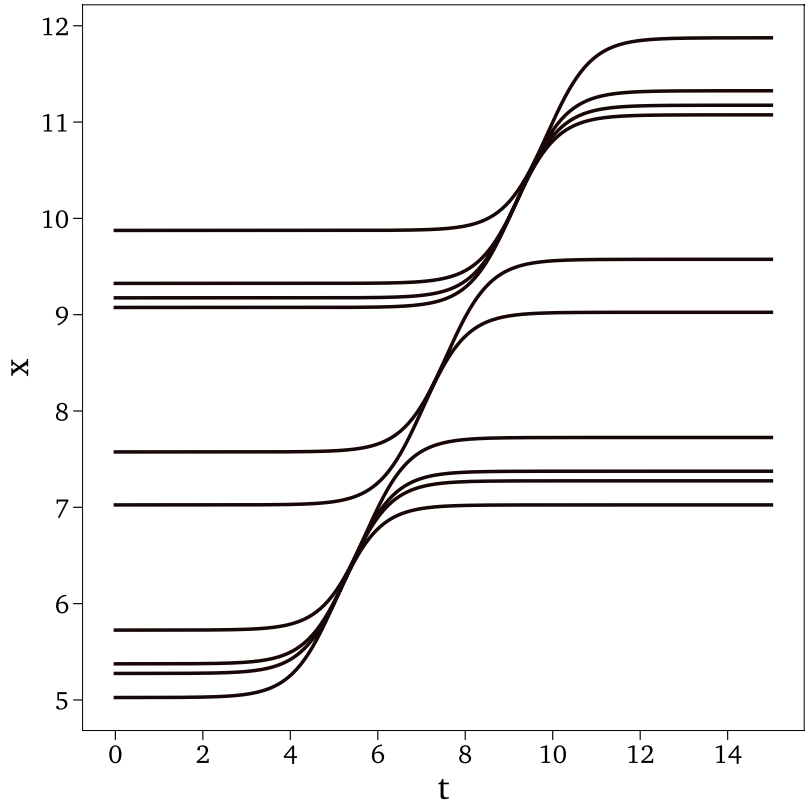
$$\begin{cases} \dot{x} = \operatorname{sech}^2\left(\frac{x_0}{a} - t\right) \\ x(t=0) = x_0 \end{cases}$$

which has solution:

$$x(t) = \tanh\left(t - \frac{x_0}{a}\right) + \tanh\left(\frac{x_0}{a}\right) + x_0,$$

being a a parameter set to $a = 1$

Some training scenarios



The physical system is implemented in
$WORKSPACE_PATH/ROMNet/romnet/romnet/pinn/system/transtanh.py

The m, c, and k parameters can be found in
$WORKSPACE_PATH//ROMNet/romnet/database/TransTanh/Params/

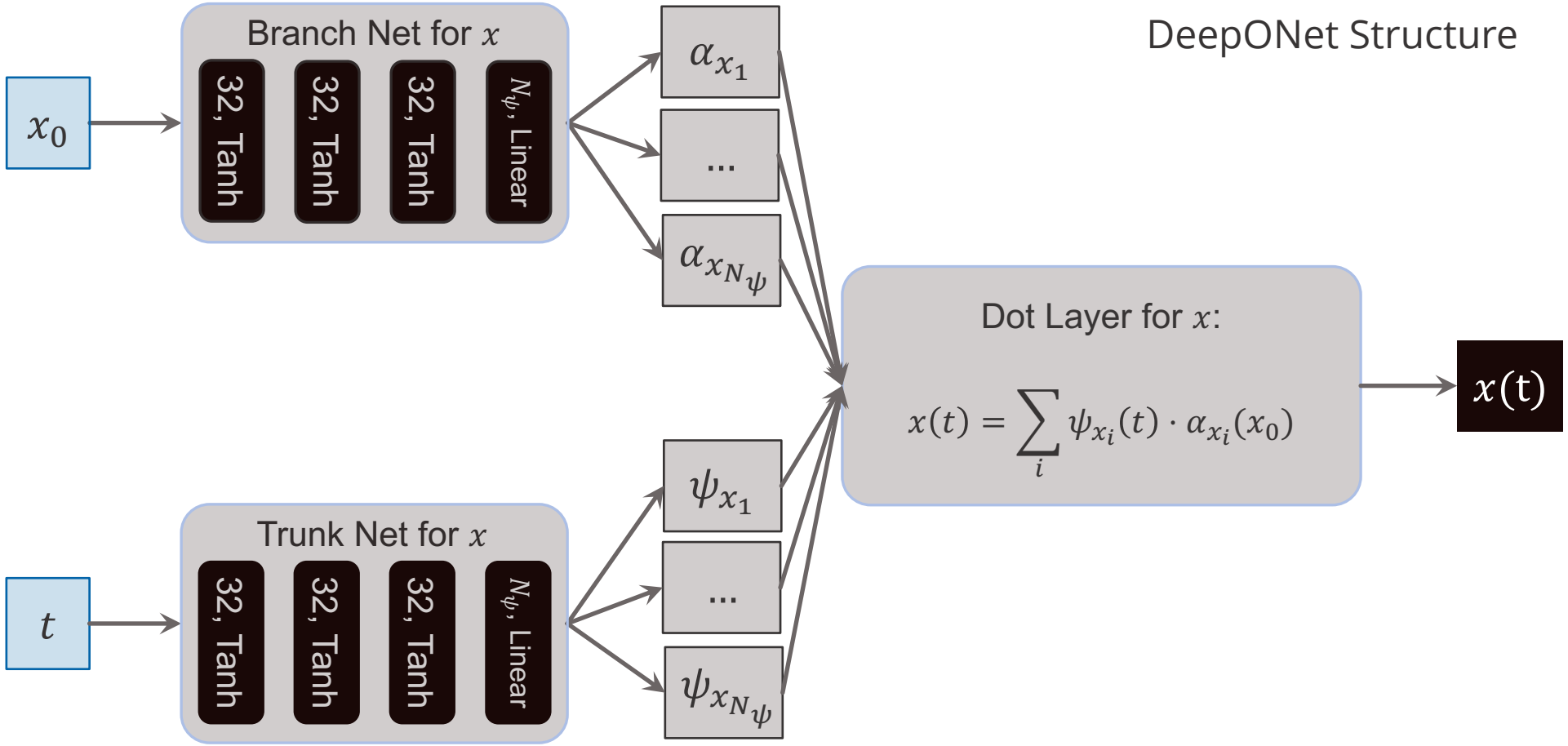
A Translating Hyperbolic Tangent Test Case

Run Jupyter Notebook /ROMNet/romnet/scripts/generating_data/TransTanh/Generate_Data_1.ipynb for generating training and test data

A Translating Hyperbolic Tangent Test Case

Test Cases 1 & 2

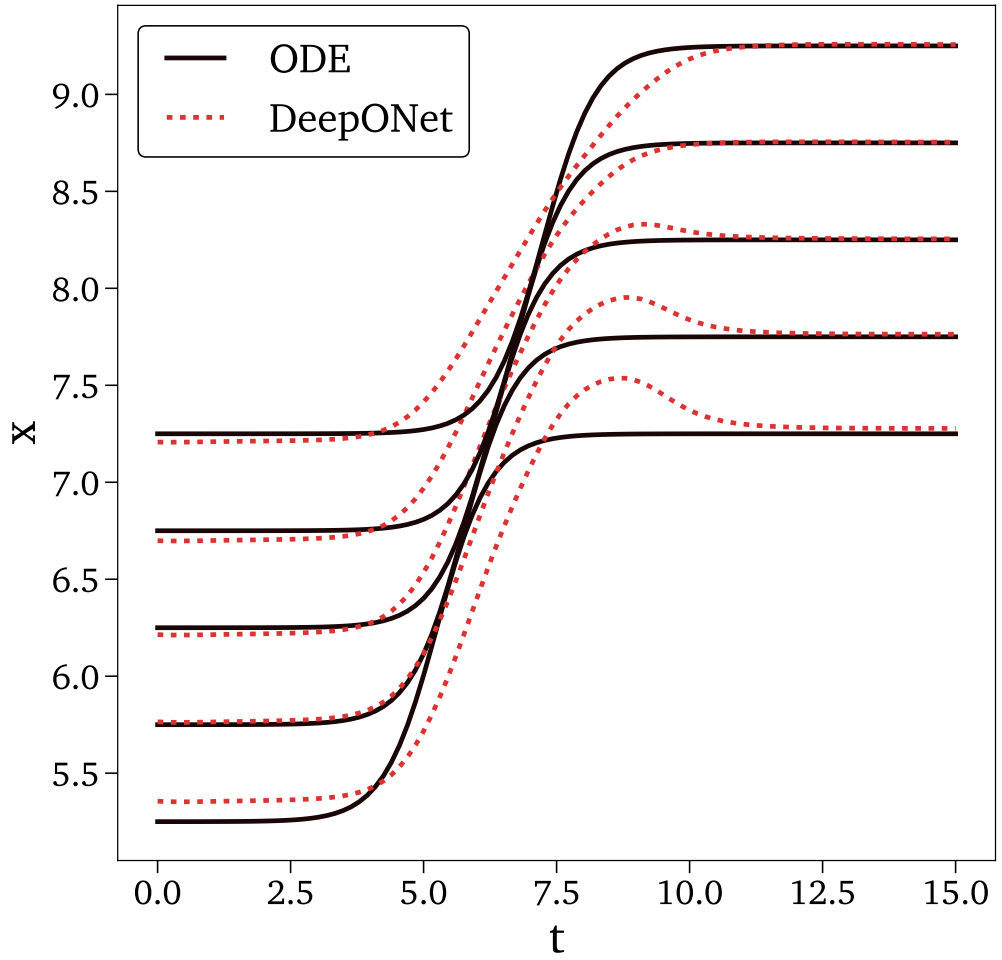
A Translating Hyperbolic Tangent Test Case



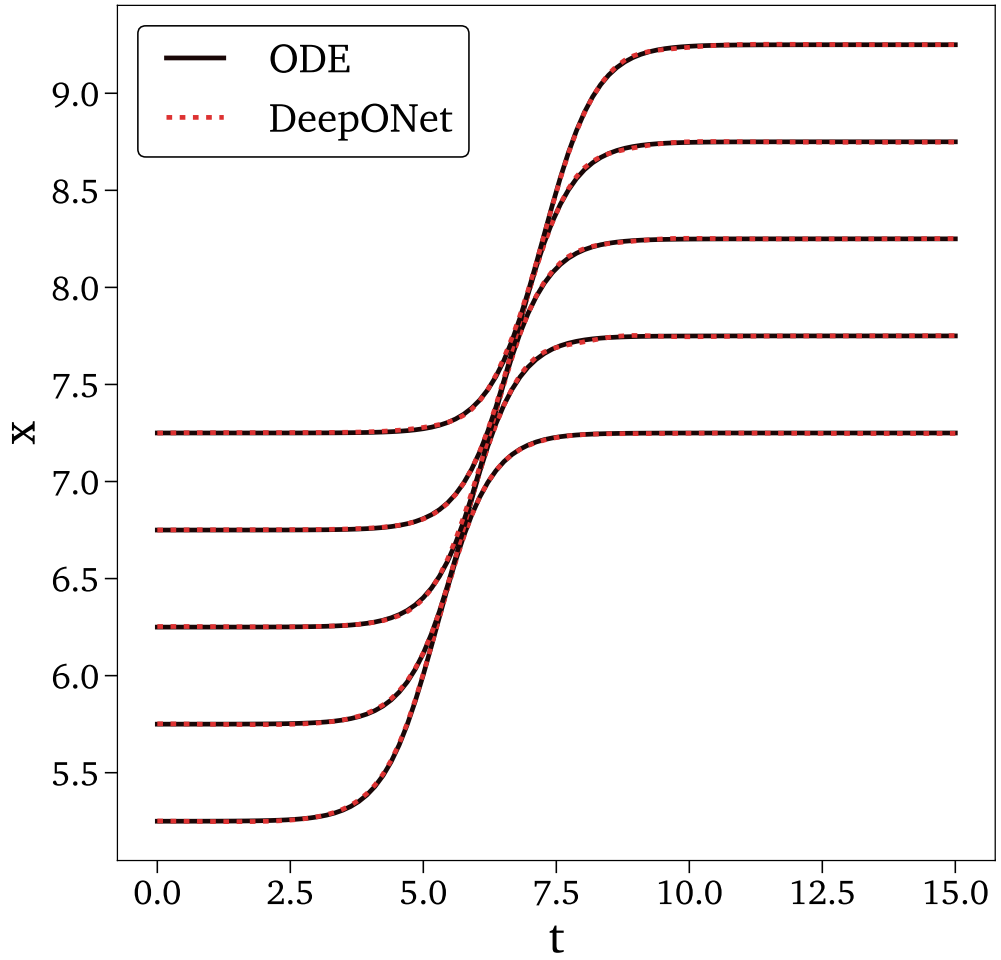
A Translating Hyperbolic Tangent Test Case

Results on training scenarios

$N_\psi = 2$



$N_\psi = 8$



A Translating Hyperbolic Tangent Test Case

Test Case 1: Data-driven deep operator network (DeepONet) for predicting position

1.1. Copy $WORKSPACE_PATH/ROMNet/romnet/input/TransTanh/DeepONet/TransTanh_TestCase1/ROMNet_Input.py
to $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py

1.2. In $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py, change:
1.2.1. "self.WORKSPACE_PATH = ..."

1.3. Move to $WORKSPACE_PATH/ROMNet/romnet/app/

1.4. Run: "python3 ROMNet.py ../input/

1.5. Postprocess results via: $WORKSPACE_PATH/ROMNet/romnet/scripts/postprocessing/TransTanh/DeepONet/Predict_DeepONet.ipynb

A Translating Hyperbolic Tangent Test Case

Test Case 2: Physics-Informed deep operator network (DeepONet) for predicting position

2.1. Copy $WORKSPACE_PATH/ROMNet/romnet/input/TransTanh/DeepONet/TransTanh_TestCase2/ROMNet_Input.py
to $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py

2.2. In $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py, change:
6.2.1. "self.WORKSPACE_PATH = ..."

2.3. Move to $WORKSPACE_PATH/ROMNet/romnet/app/

2.4. Run: "python3 ROMNet.py ../input/

2.5. Postprocess results via: $WORKSPACE_PATH/ROMNet/romnet/scripts/postprocessing/TransTanh/DeepONet/Predict_DeepONet.ipynb

A Translating Hyperbolic Tangent Test Case

A scenario-aggregated Principal Component Analysis

By aggregating the training scenarios for $x_i(t)$, where i represents the scenario index:

$$\boldsymbol{X} = \begin{bmatrix} | & | & & | & | \\ \boldsymbol{x_1} & \boldsymbol{x_2} & \cdots & \boldsymbol{x_{99}} & \boldsymbol{x_{100}} \\ | & | & & | & | \end{bmatrix}$$

$$\mathbf{dim}(\boldsymbol{X}) = \boldsymbol{N_t} \times \boldsymbol{N_s}$$

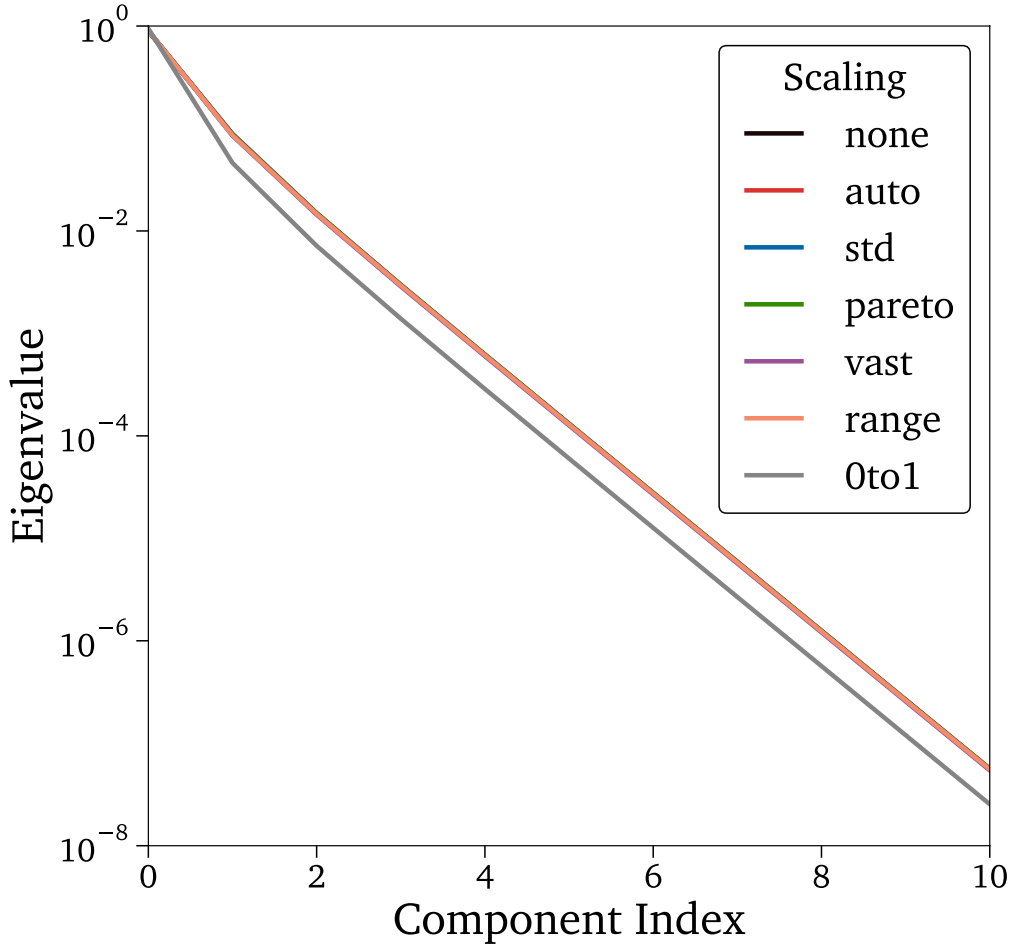
No of time instants

No of scenarios

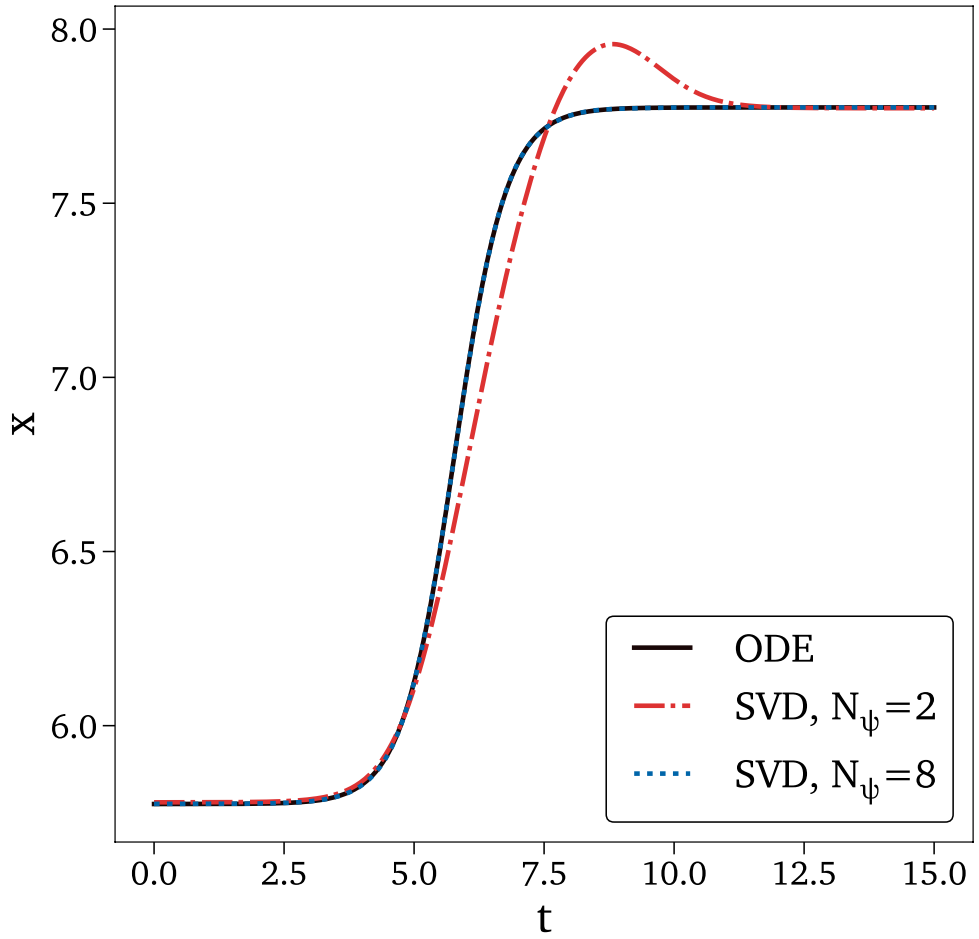
A Translating Hyperbolic Tangent Test Case

By performing SVD on X (PCA of $\mathbf{R_X}$): $\boldsymbol{\Psi} = \dfrac{\boldsymbol{X} - \boldsymbol{C}}{\boldsymbol{D}} \boldsymbol{A}$

Eigenvalues of $\boldsymbol{X}$

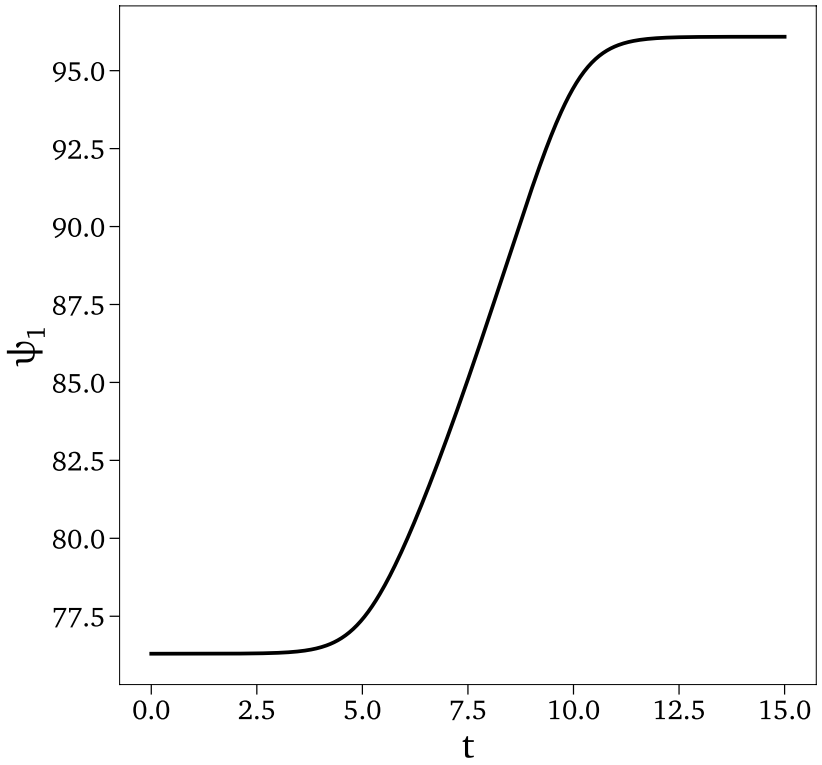


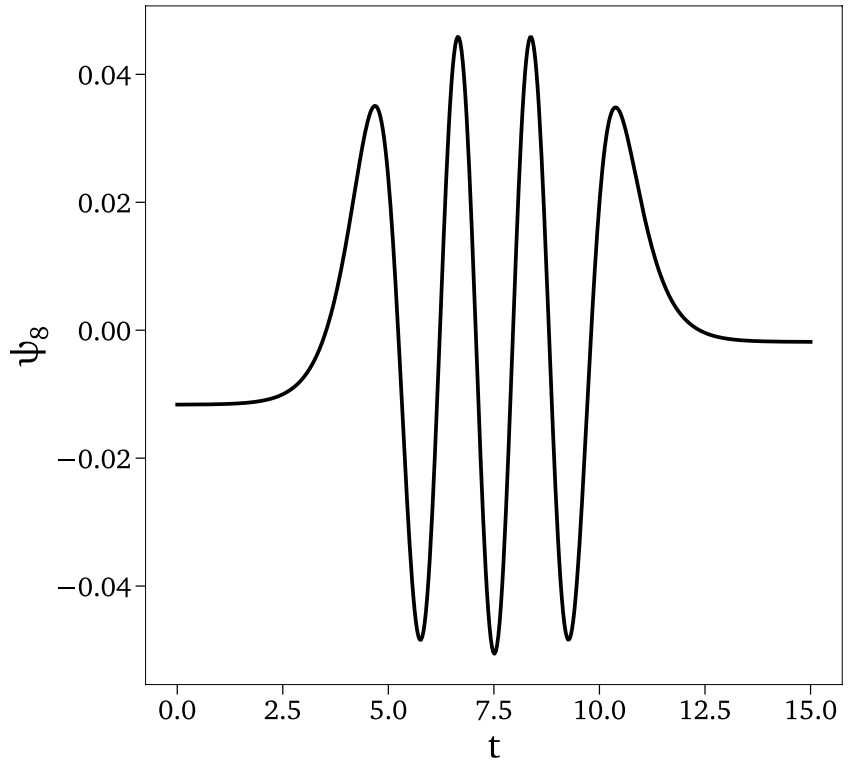
Scenario 1 (i.e., first column of $\boldsymbol{X}$) encoded and decoded via SVD



A Translating Hyperbolic Tangent Test Case

1st and 8th eigenmodes of $\mathbf{R}_X$





Note: The high-frequency character of the low-energy modes makes them harder to be fitted with high accuracy

However, we know from the analytical solution that the number of modes can be reduced to one by properly taking care of the translational invariance

A Translating Hyperbolic Tangent Test Case

Run Jupyter Notebook /ROMNet/romnet/scripts/generating_data/TransTanh/Generate_Data_2.ipynb for generating SVD data

A Translating Hyperbolic Tangent Test Case

We suggest two improvements to the original DeepONet Structure:

Improvement 1:

Effective SVD/PCA must be preceded by centering and scaling [1-5].

“Since the variables are characterized by different units and ranges, preprocessing in the form of centering and scaling is a mandatory operation [1, 2].
Data centering consists of subtracting the mean value of each variable to all dataset observations: in this way, all the observations can be seen as fluctuations from a mean value.
Scaling is achieved by dividing each variable by a given scaling factor, which can be different depending on the adopted scaling criterion.
…
The way data are preprocessed can have a strong influence on the data analysis and the reduced-order modeling for combustion applications [3], as the scaling technique can be more or less sensitive to the presence of outliers or it can highlight a specific pattern in the data” [6]

[1] Rasmus Bro and Age K. Smilde - Centering and scaling in component analysis - 2001
[2] Robert A van den Berg et al. - Centering, scaling, and transformations: improving the biological information content of … - 2006
[3] A. Parente and J. C. Sutherland - Principal component analysis of turbulent combustion data: Data pre-processing and … - 2012
[4] A. Armstrong and J. C. Sutherland - A technique for characterising feature size and quality of manifolds – 2021
[5] K. Zdybal et al - PCAfold: Python software to generate, analyze and improve PCA-derived low-dimensional manifold – 2021
[6] G. D’Alessio et al - Analysis of Turbulent Reacting Jets via Principal Component Analysis - 2020

A Translating Hyperbolic Tangent Test Case

We suggest two improvements to the original DeepONet Structure:

Improvement 1:

Effective SVD/PCA must be preceded by centering and scaling [1-5].

The vectors $\boldsymbol{C}$ and $\boldsymbol{D}$ have not equivalents in the original DeepONet formulation.

Note: Scaling and centering parameters need to be dependent on initial conditions
(i.e., $\boldsymbol{C}$ and $\boldsymbol{D}$ are vectors of shape $N_s \times 1$)

[1] Rasmus Bro and Age K. Smilde - Centering and scaling in component analysis - 2001
[2] Robert A van den Berg et al. - Centering, scaling, and transformations: improving the biological information content of ... - 2006
[3] A. Parente and J. C. Sutherland - Principal component analysis of turbulent combustion data: Data pre-processing and ... - 2012
[4] A. Armstrong and J. C. Sutherland - A technique for characterising feature size and quality of manifolds – 2021
[5] K. Zdybal et al - PCAfold: Python software to generate, analyze and improve PCA-derived low-dimensional manifold – 2021
[6] G. D'Alessio et al - Analysis of Turbulent Reacting Jets via Principal Component Analysis - 2020

A Translating Hyperbolic Tangent Test Case

Note 1: Bias term

Multiple references [1,2] introduce the bias term, but:

- In none it is justified (e.g., "... where we include an additional bias term b0 as a trainable variable as it may reduce the generalization error" [2])
- In none it depends on the initial conditions (e.g., "... \phi_0(\epsilon is the mean function of u(\epsilon) computed from the training data-set" [1])

[1] L. Lu et al - Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators - 2021
[2] M. Yin et al - Simulating progressive intramural damage leading to aortic dissection using an operator-regression neural network - 2022

A Translating Hyperbolic Tangent Test Case

We suggest two improvements to the original DeepONet Structure:

Improvement 1:

Effective SVD/PCA must be preceded by centering and scaling [1-5].

The vectors $\boldsymbol{C}$ and $\boldsymbol{D}$ have not equivalents in the original DeepONet formulation.

Note: Scaling and centering parameters need to be dependent on initial conditions
(i.e., $\boldsymbol{C}$ and $\boldsymbol{D}$ are vectors of shape $N_s \times 1$)

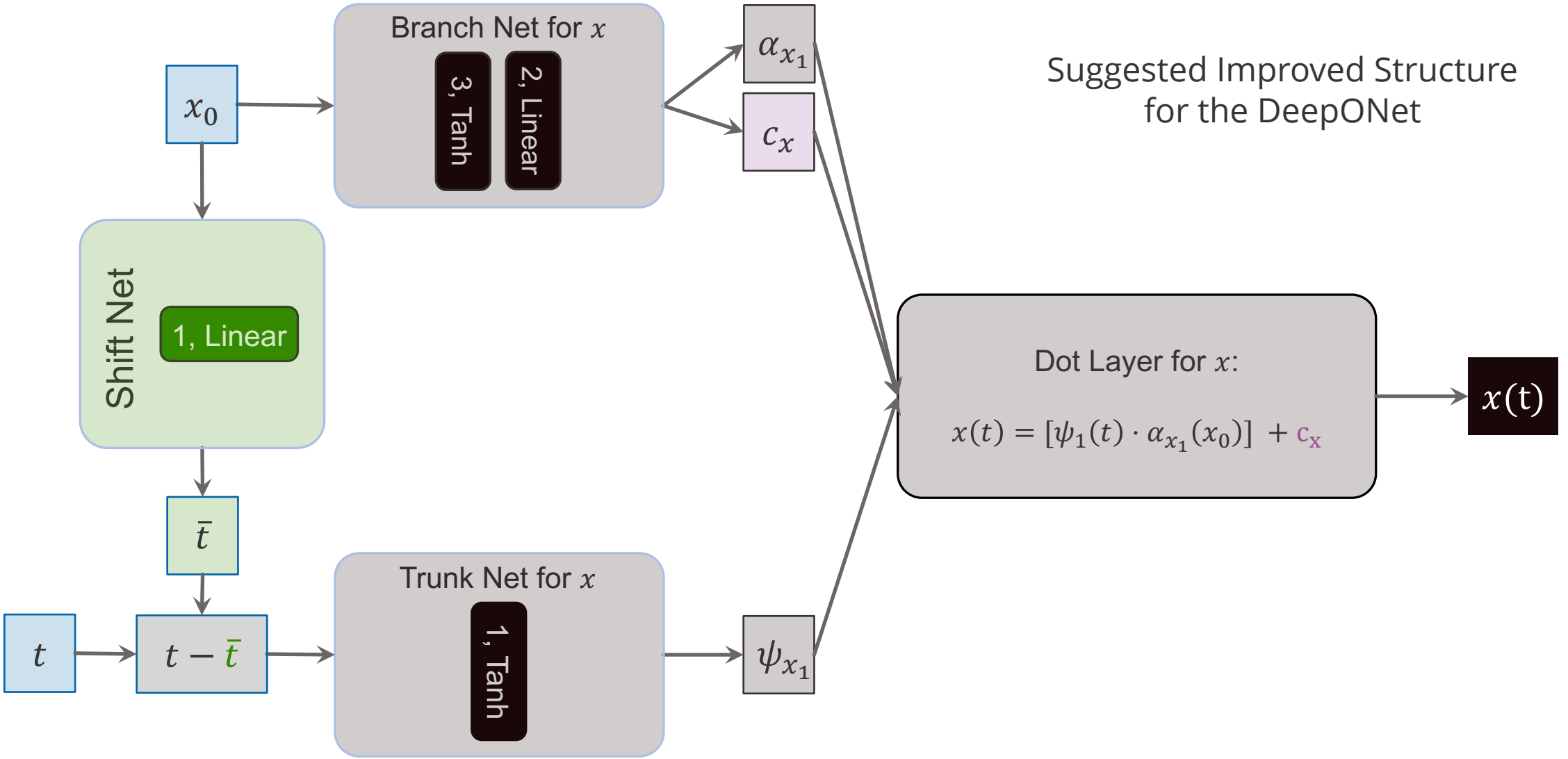
Improvement 2:

Effective SVD/PCA must be preceded by removal of rigid transformations (e.g., time shifts)

A Translating Hyperbolic Tangent Test Case

Test Cases 3 & 4

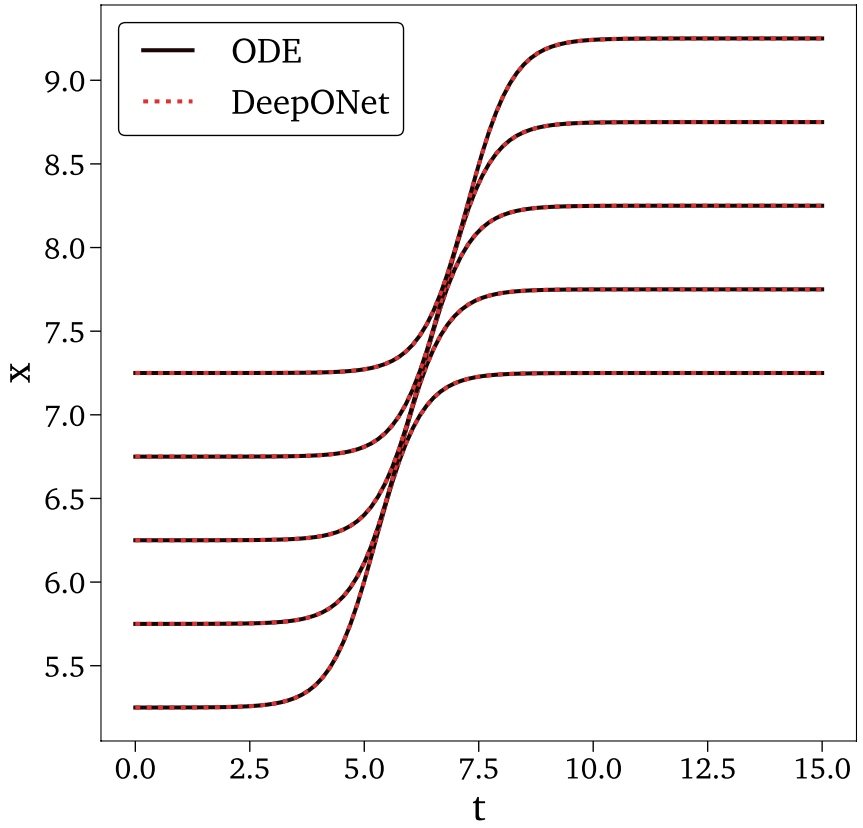
A Translating Hyperbolic Tangent Test Case



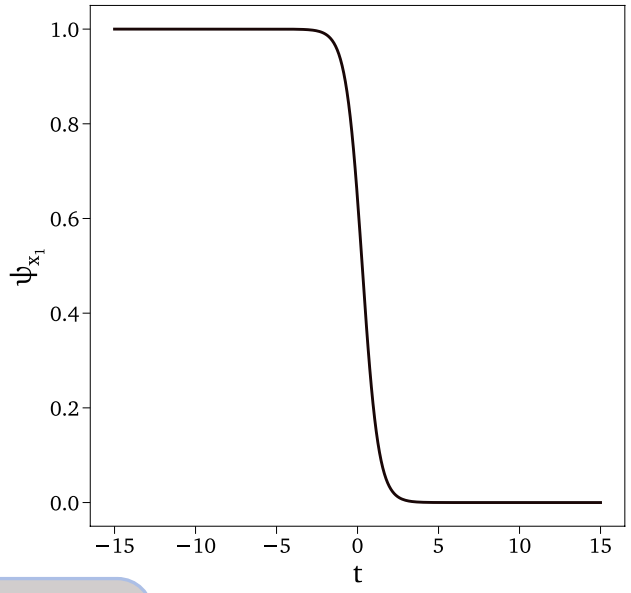
A Translating Hyperbolic Tangent Test Case

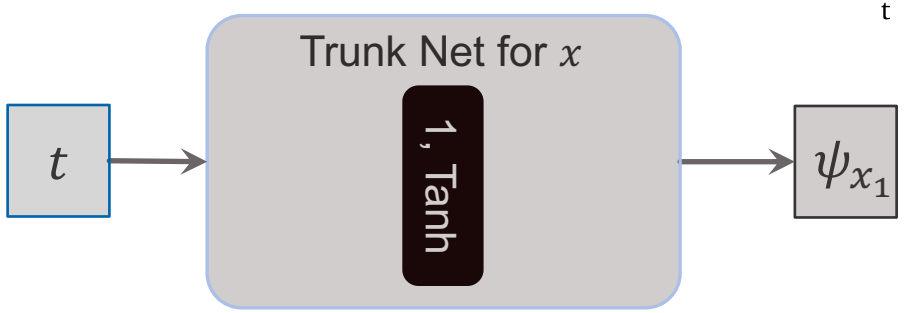
Results from the improved structure

Some test scenarios



Trunk output
(w/o including shifting
block's pre-transformation)

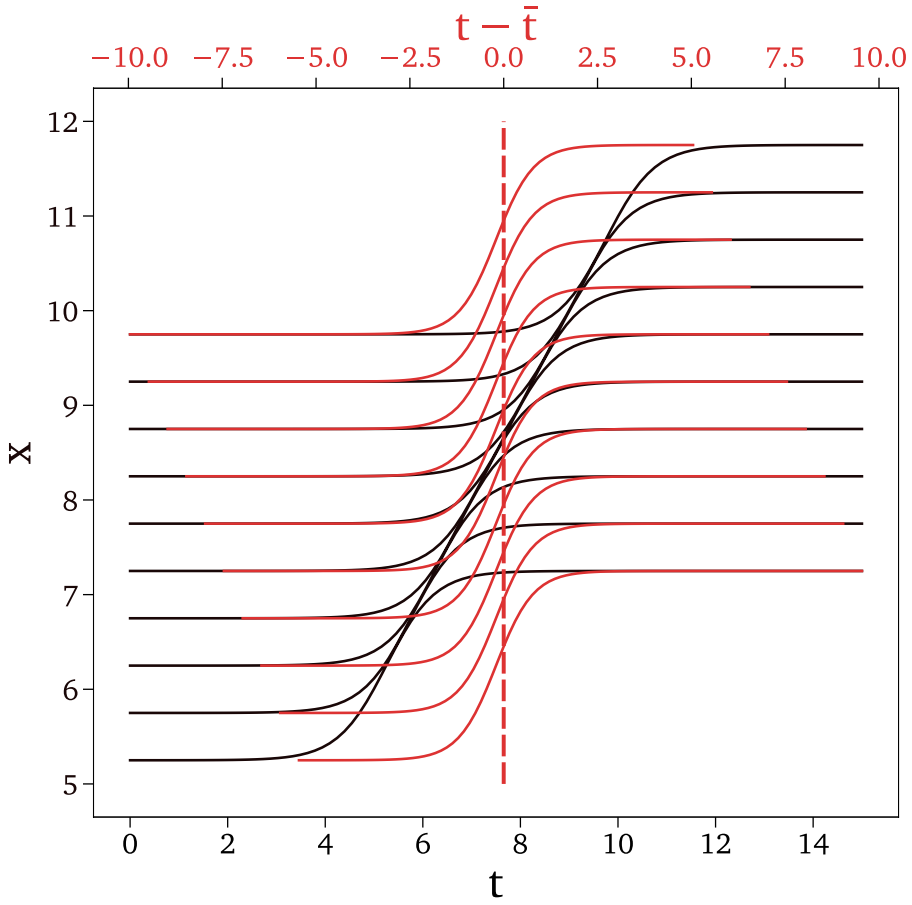


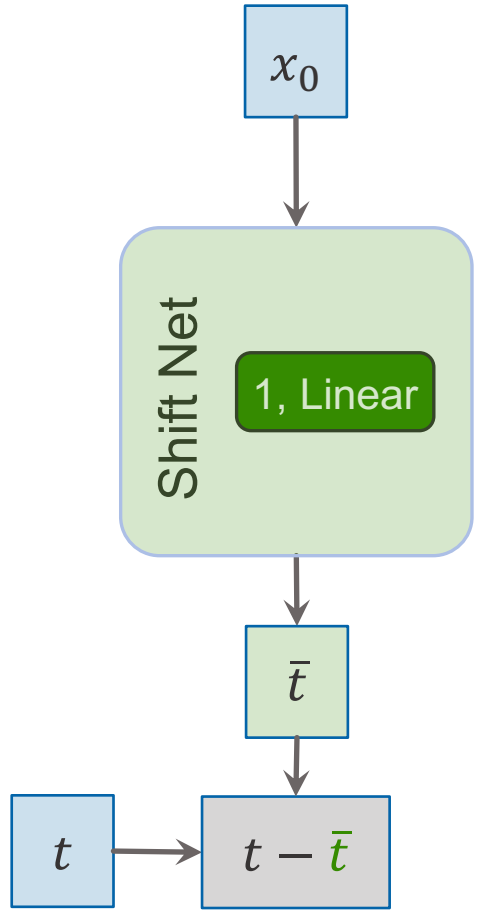


A Translating Hyperbolic Tangent Test Case

Results from the improved structure

Test scenarios vs t and vs $(t - \bar{t})$





A Translating Hyperbolic Tangent Test Case

Test Case 3: Data-driven improved deep operator network (DeepONet) for predicting position

3.1. Copy $WORKSPACE_PATH/ROMNet/romnet/input/TransTanh/DeepONet/TransTanh_TestCase3/ROMNet_Input.py
to $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py

3.2. In $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py, change:
7.2.1. "self.WORKSPACE_PATH = ..."

3.3. Move to $WORKSPACE_PATH/ROMNet/romnet/app/

3.4. Run: "python3 ROMNet.py ../input/

3.5. Postprocess results via: $WORKSPACE_PATH/ROMNet/romnet/scripts/postprocessing/TransTanh/DeepONet/Predict_DeepONet.ipynb

A Translating Hyperbolic Tangent Test Case

Test Case 4: Physics-Informed improved deep operator network (DeepONet) for predicting position

4.1. Copy $WORKSPACE_PATH/ROMNet/romnet/input/TransTanh/DeepONet/TransTanh_TestCase4/ROMNet_Input.py
to $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py

4.2. In $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py, change:
8.2.1. "self.WORKSPACE_PATH = ..."

4.3. Move to $WORKSPACE_PATH/ROMNet/romnet/app/

4.4. Run: "python3 ROMNet.py ../input/

4.5. Postprocess results via: $WORKSPACE_PATH/ROMNet/romnet/scripts/postprocessing/TransTanh/DeepONet/Predict_DeepONet.ipynb

The input file for Test Case 4 differs from the one of Test Case 3:

self.**n_branch_out**: We added other two neurons to the last layer of the brunch net (i.e., centering and scaling)

Note:

- If self.n_branch_out == self.n_modes: No centering, no scaling
- If self.n_branch_out == self.n_modes+1: Only centering
- If self.n_branch_out == self.n_modes+2: Centering and scaling

self.**structure**: We added a rigid block to the structure …

self.**rigid_type**: … and it's of shifting type