



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)**



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:

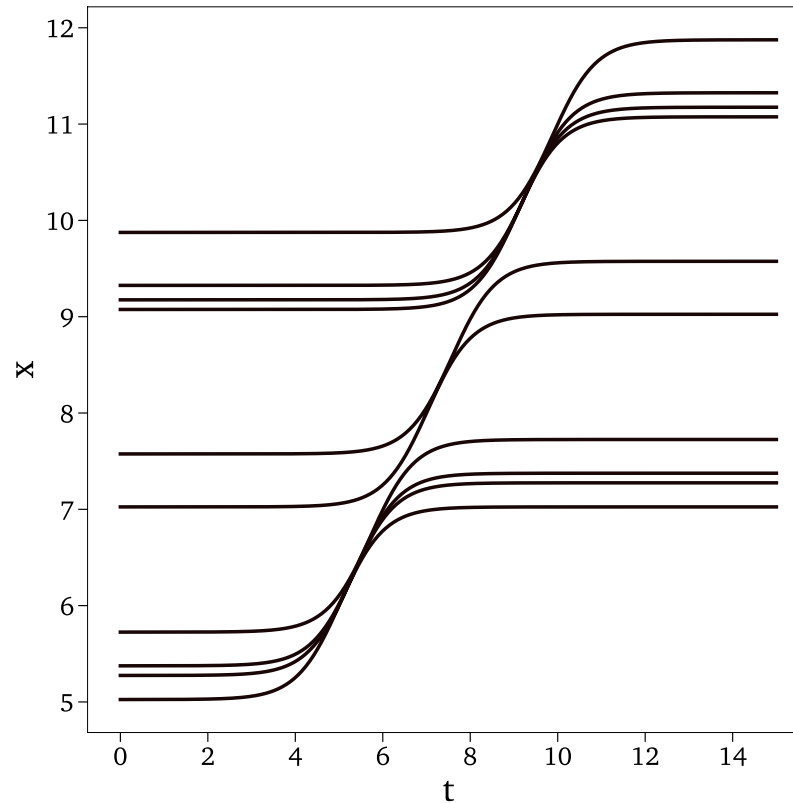
$$\begin{cases} \dot{x} = \text{sech}^2\left(\frac{x_0}{a} - t\right) \\ x(t=0) = x_0 \end{cases}$$

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



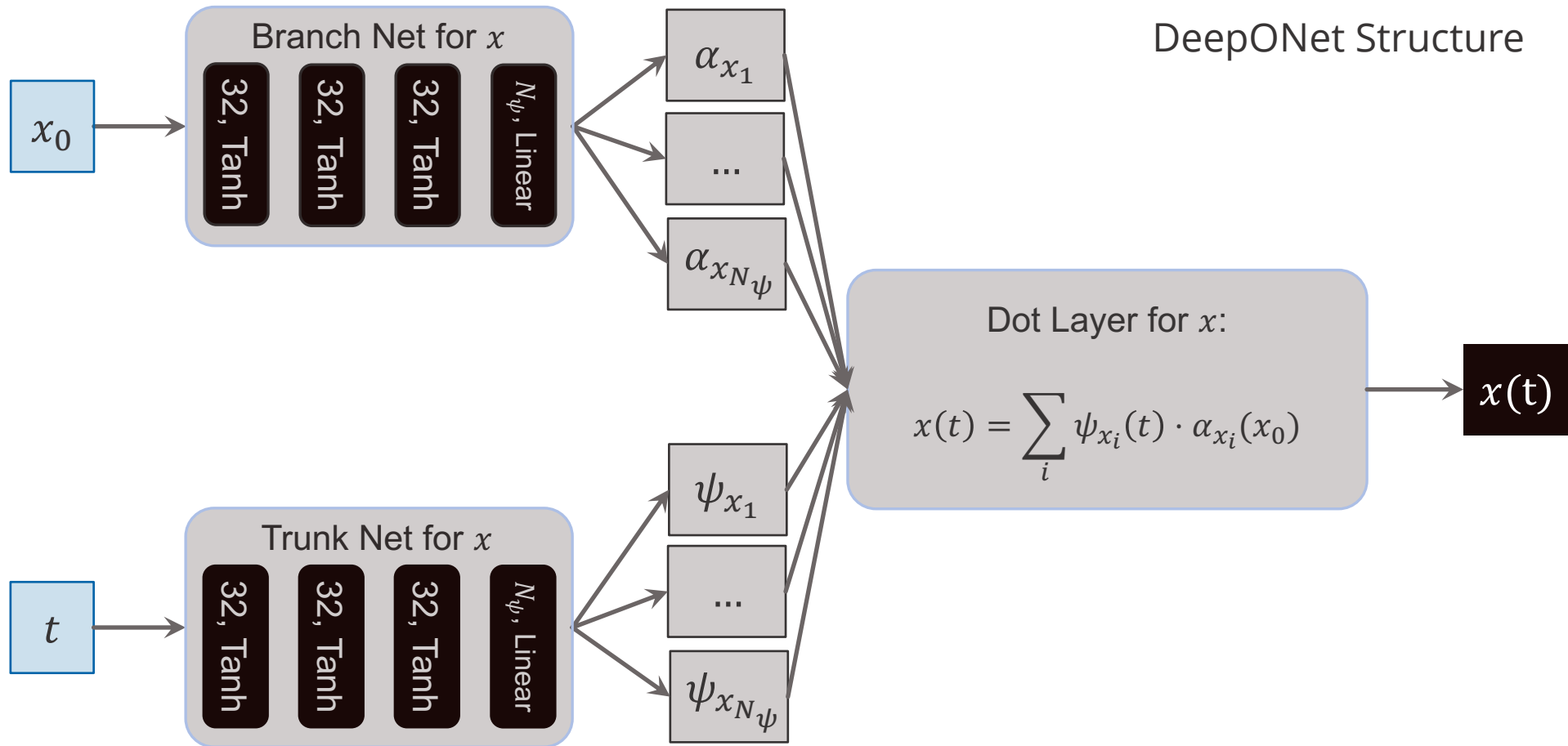
Run Jupyter Notebook `/ROMNet/romnet/scripts/generating_data/TransTanh/Generate_Data_1.ipynb` for generating training and test data

# A Translating Hyperbolic Tangent Test Case



**Test Cases 5 & 6**

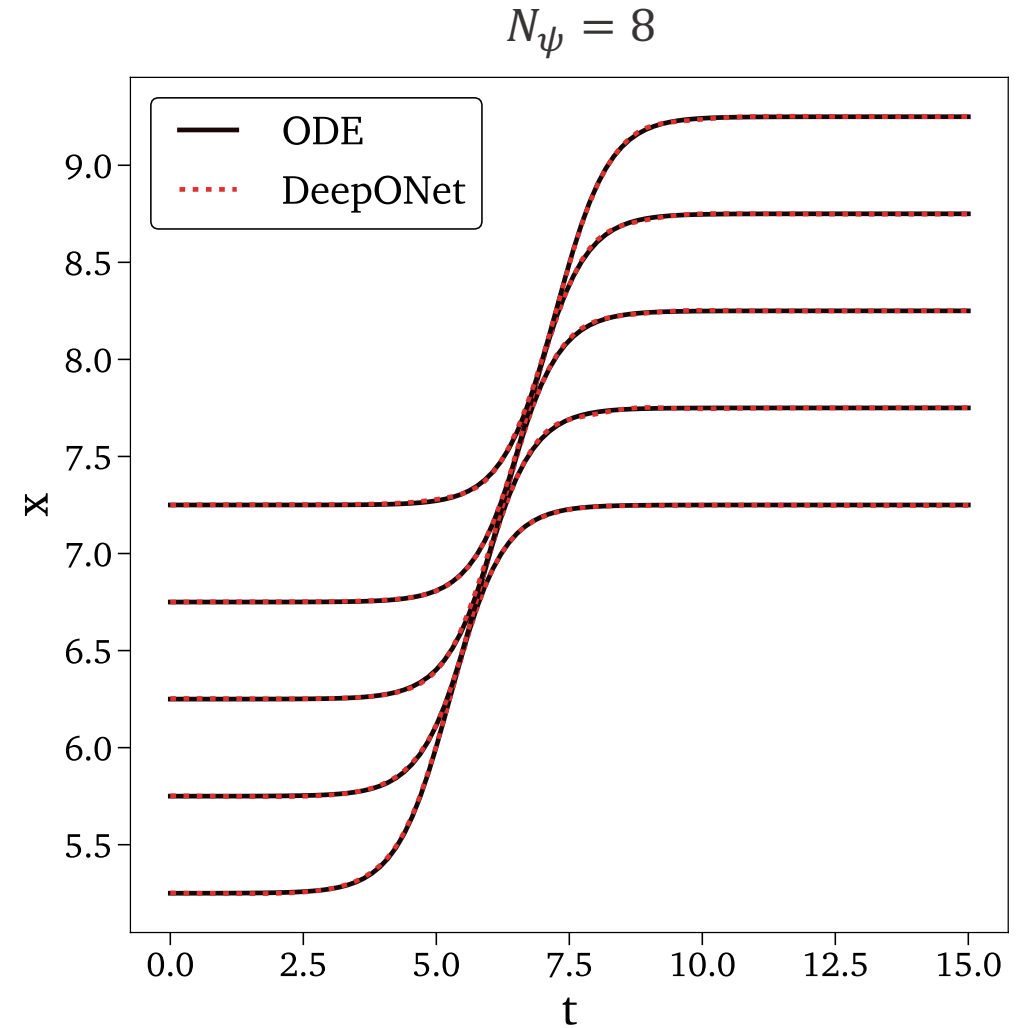
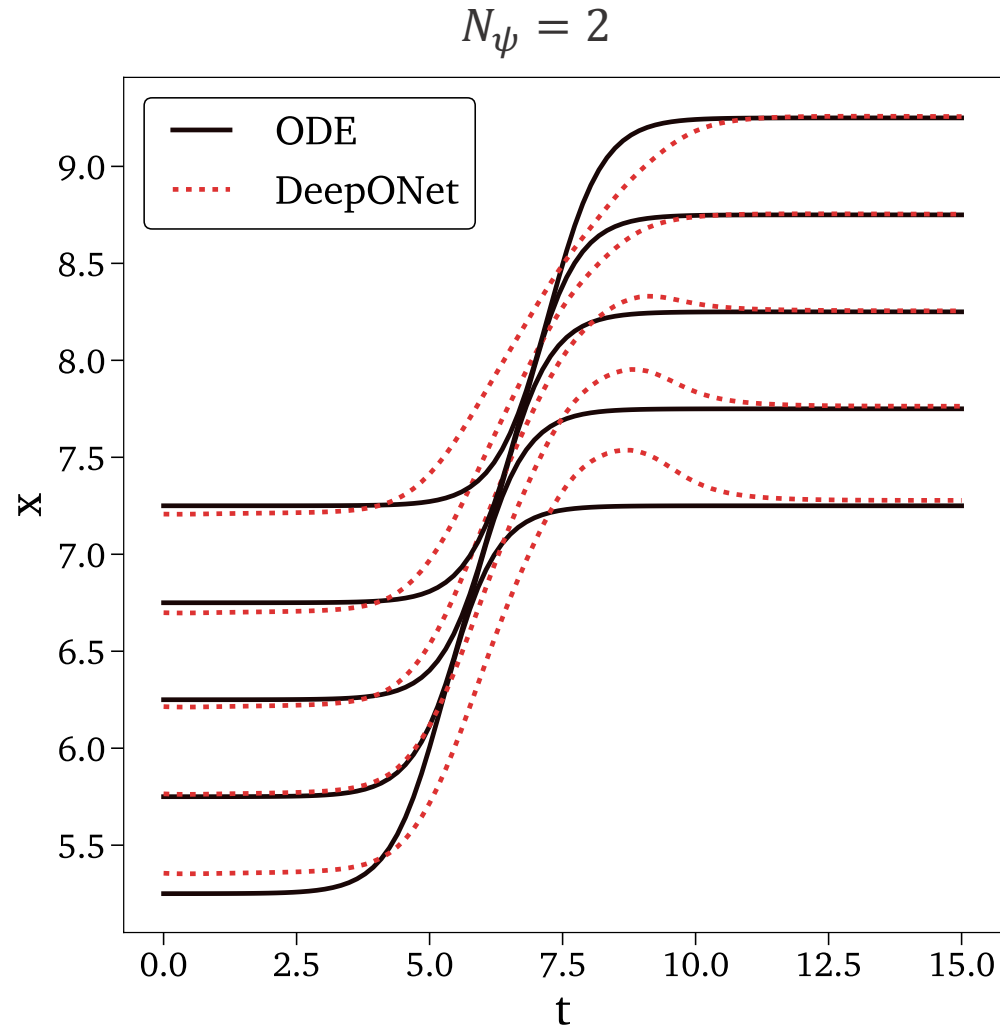
# A Translating Hyperbolic Tangent Test Case



# A Translating Hyperbolic Tangent Test Case



Results on training scenarios



# A Translating Hyperbolic Tangent Test Case



## Test Case 5: Data-driven deep operator network (DeepONet) for predicting position

- 5.1. Copy \$WORKSPACE\_PATH/ROMNet/romnet/input/TransTanh/DeepONet/TestCase5/ROMNet\_Input.py to \$WORKSPACE\_PATH/ROMNet/romnet/input/ROMNet\_Input.py
- 5.2. In \$WORKSPACE\_PATH/ROMNet/romnet/input/ROMNet\_Input.py, change:
  - 5.2.1. "self.WORKSPACE\_PATH = ..."
- 5.3. Move to \$WORKSPACE\_PATH/ROMNet/romnet/app/
- 5.4. Run: "python3 ROMNet.py ../input/"
- 5.5. Postprocess results via: \$WORKSPACE\_PATH/ROMNet/romnet/scripts/postprocessing/TransTanh/DeepONet/Predict\_DeepONet.ipynb

# A Translating Hyperbolic Tangent Test Case



## Test Case 6: Physics-Informed deep operator network (DeepONet) for predicting position

- 6.1. Copy `$WORKSPACE_PATH/ROMNet/romnet/input/TransTanh/DeepONet/TestCase6/ROMNet_Input.py`  
to `$WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py`
- 6.2. In `$WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py`, change:
  - 6.2.1. `"self.WORKSPACE_PATH = ..."`
- 6.3. Move to `$WORKSPACE_PATH/ROMNet/romnet/app/`
- 6.4. Run: `"python3 ROMNet.py ../input/"`
- 6.5. 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:

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

$$\dim(X) = N_t \times N_s$$

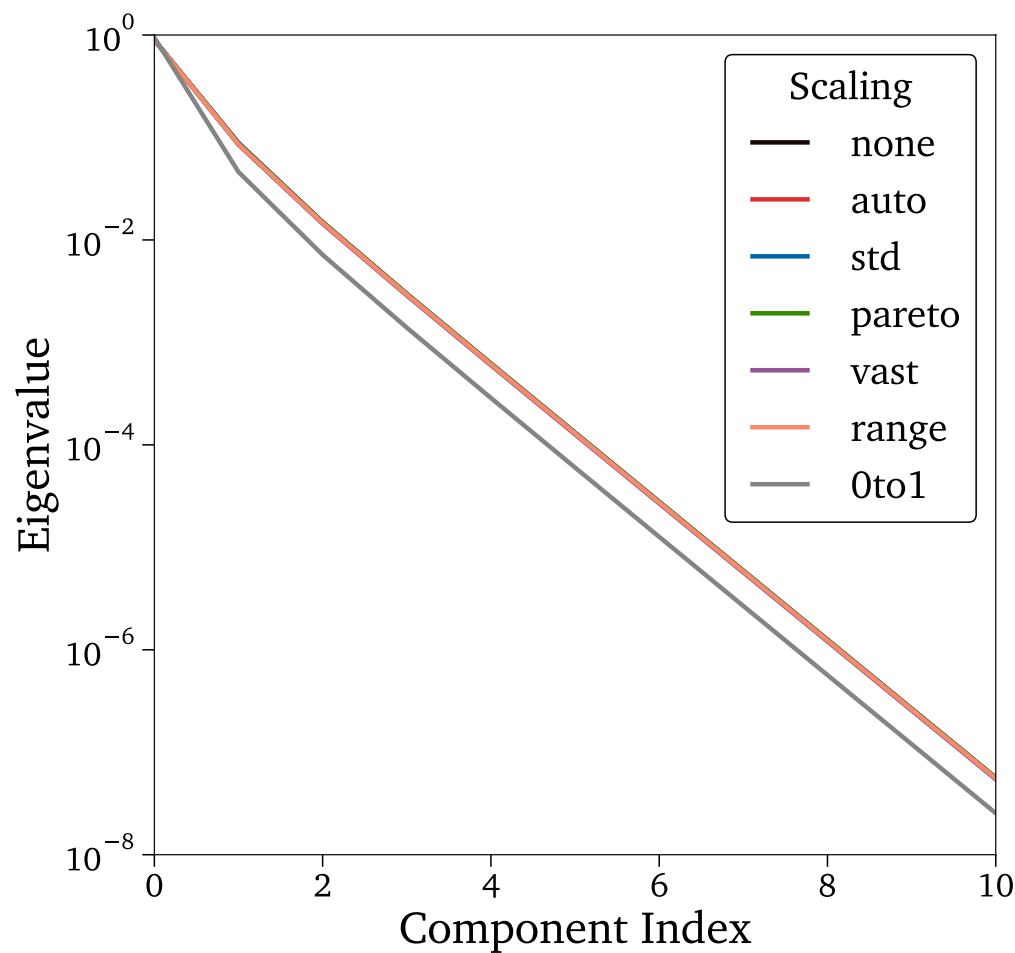
No of time      No of  
instants      scenarios

# A Translating Hyperbolic Tangent Test Case

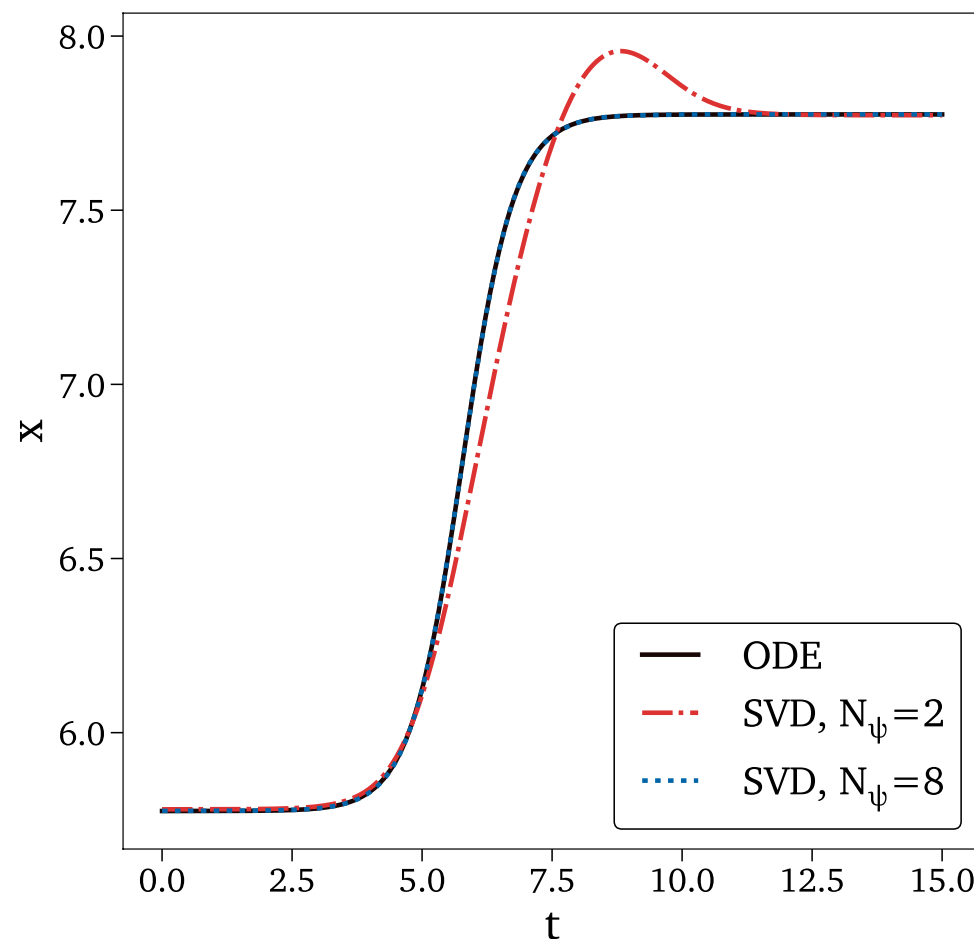


By performing PCA of  $\mathbf{R}_X$ :  $\Psi = \frac{X - \mathcal{C}}{D} A$

Eigenvalues of  $X$



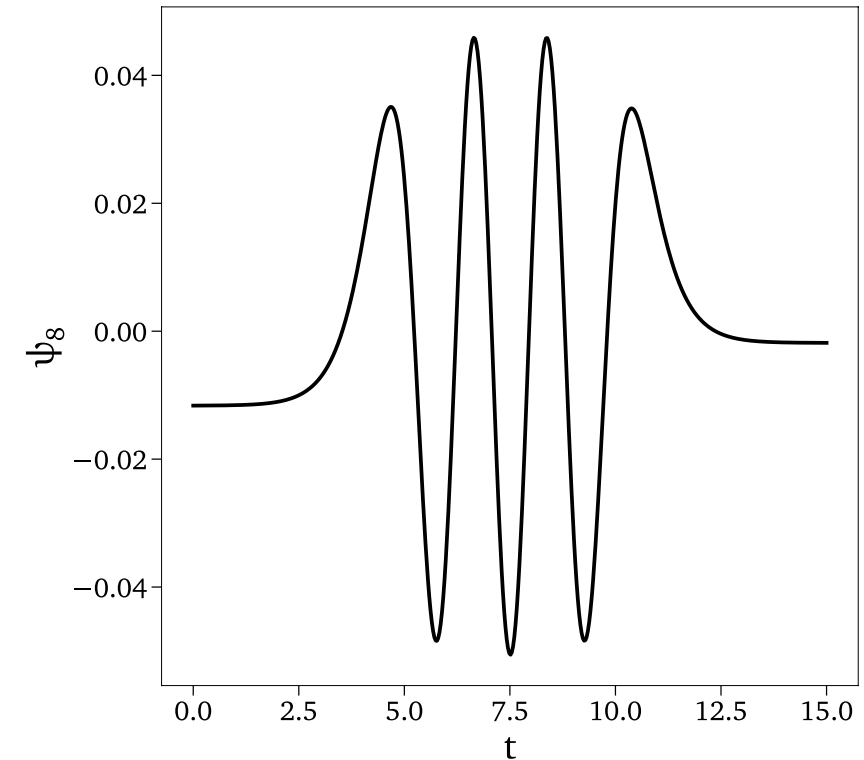
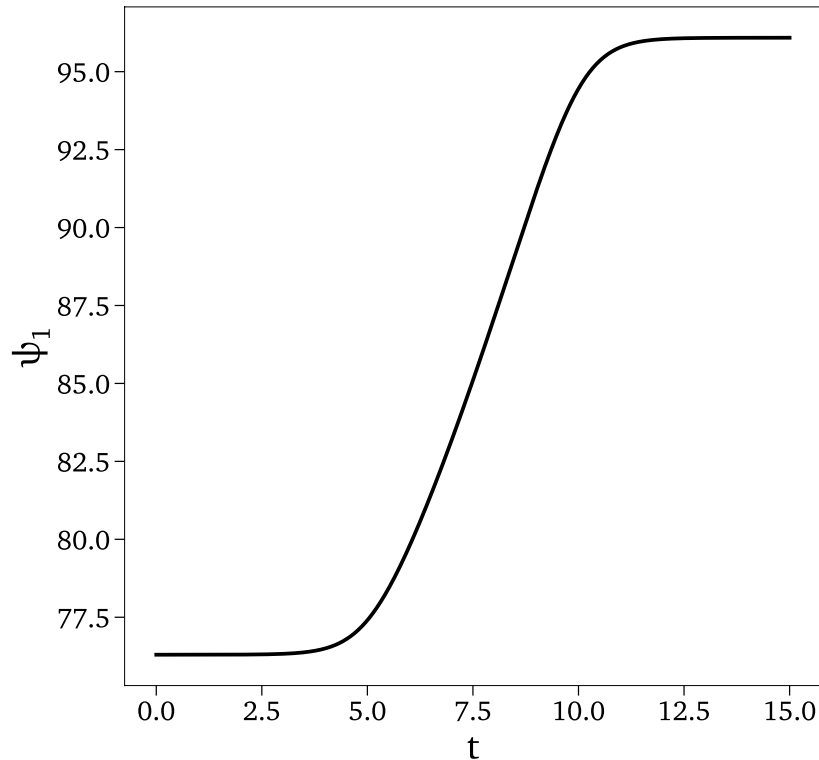
Scenario 1 (i.e., first column of  $X$ ) encoded and decoded via PCA



# A Translating Hyperbolic Tangent Test Case



1<sup>st</sup> and 8<sup>th</sup> 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 PCA data

# A Translating Hyperbolic Tangent Test Case



## We suggest two improvements to the original DeepONet Structure:

### Improvement 1:

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

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

Note: Scaling and centering parameters need to be dependent on initial conditions (i.e.,  $\mathbf{C}$  and  $\mathbf{D}$  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  $b_0$  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 PCA must be preceded by centering and scaling [1-5].

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

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

### Improvement 2:

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

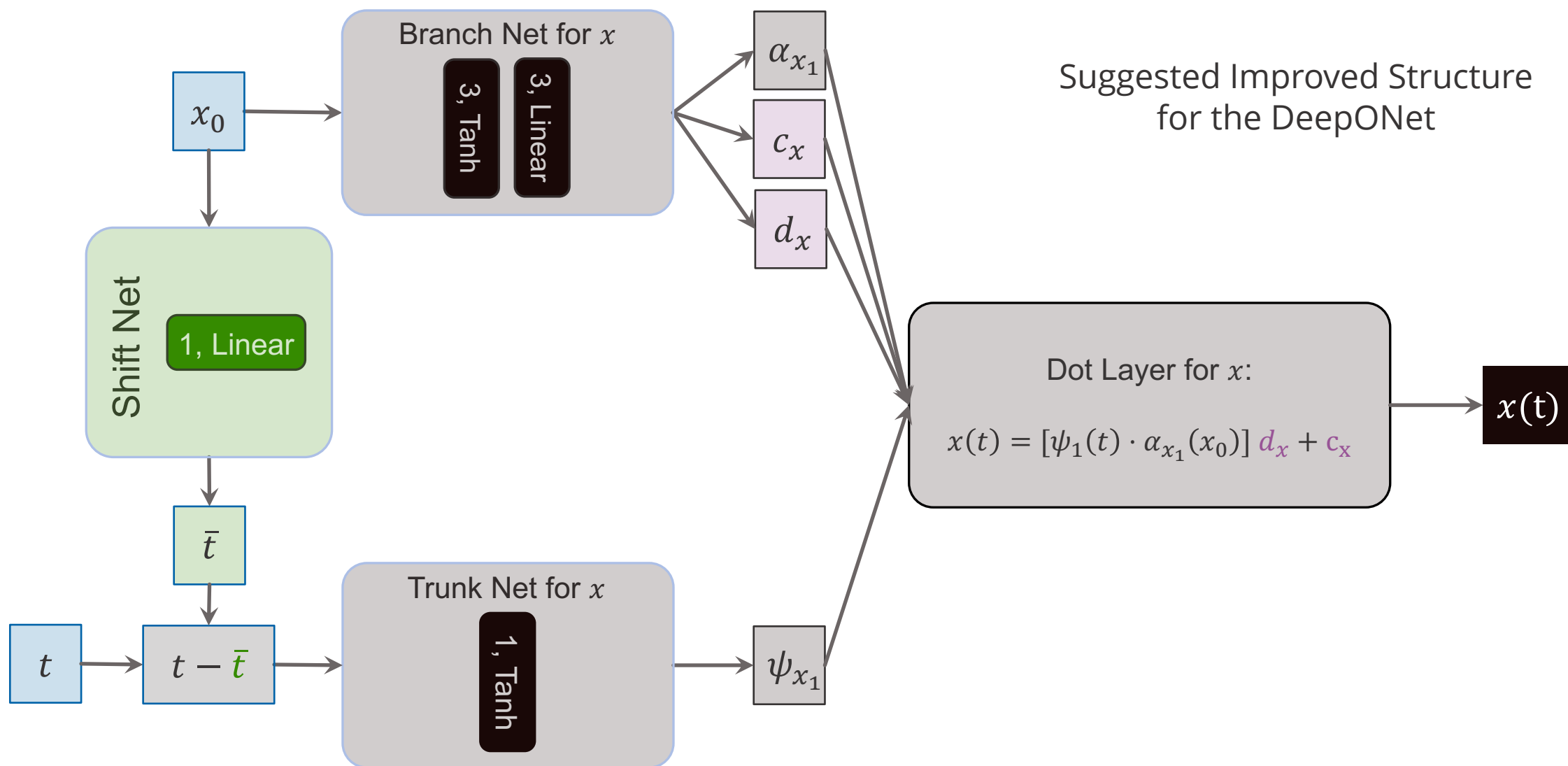


# A Translating Hyperbolic Tangent Test Case



**Test Cases 7 & 8**

# A Translating Hyperbolic Tangent Test Case

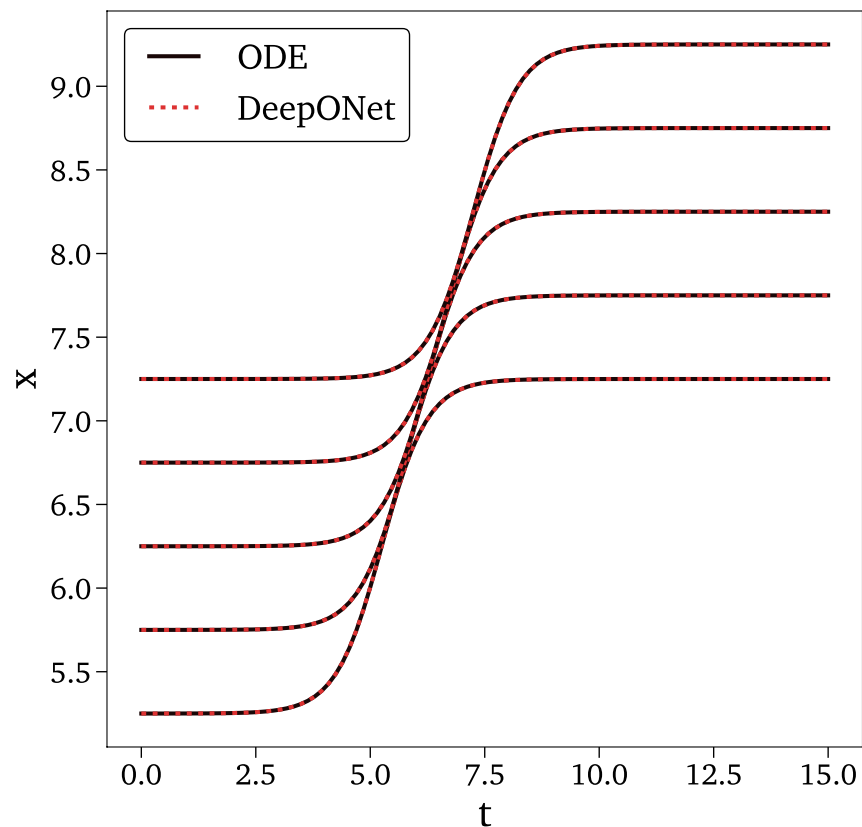


# A Translating Hyperbolic Tangent Test Case

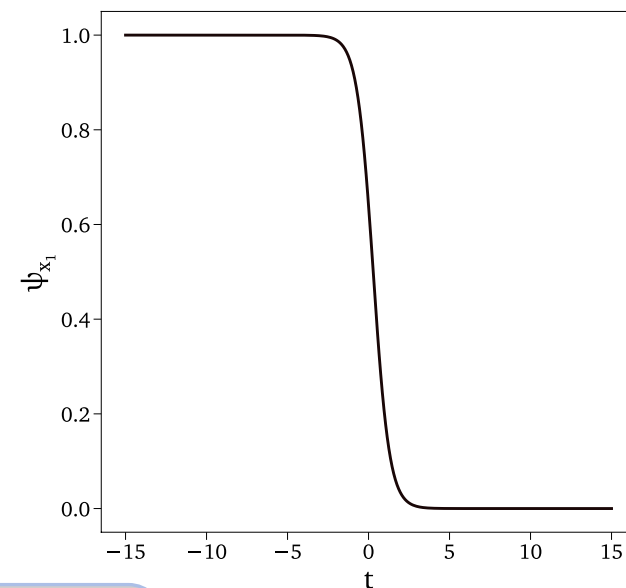


Results from the improved structure

Some test scenarios



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

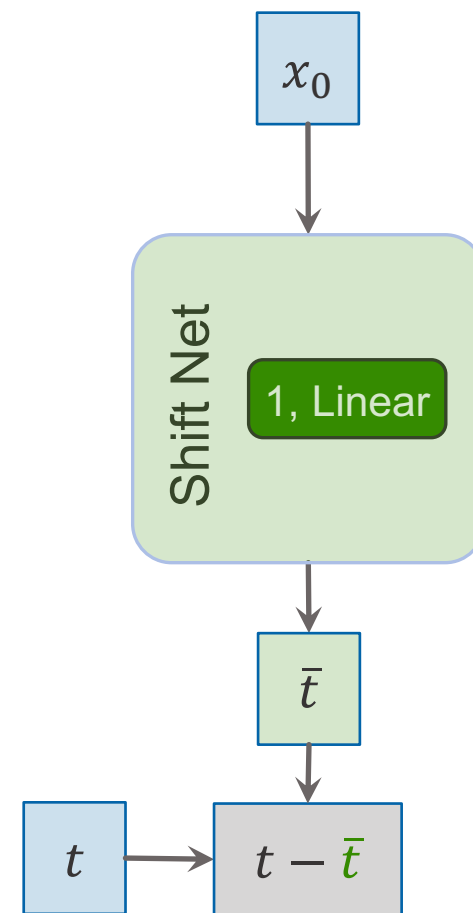
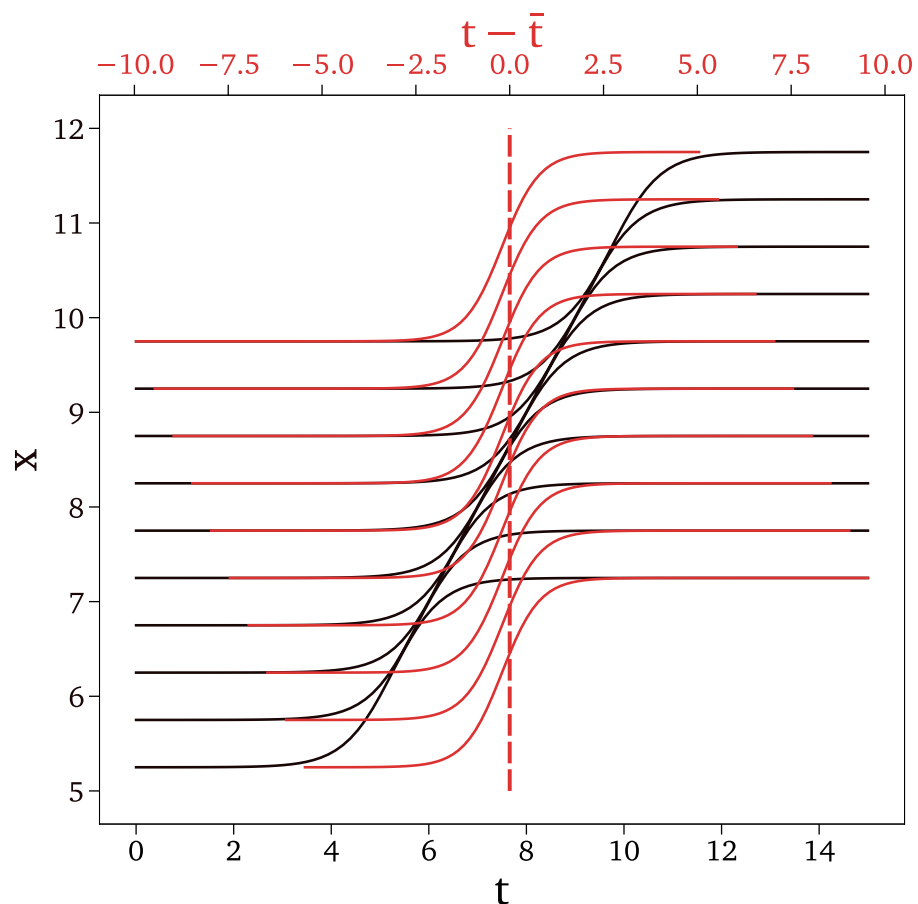


# A Translating Hyperbolic Tangent Test Case



Results from the improved structure

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



# A Translating Hyperbolic Tangent Test Case



## Test Case 7: Data-driven improved deep operator network (DeepONet) for predicting position

- 7.1. Copy `$WORKSPACE_PATH/ROMNet/romnet/input/TransTanh/DeepONet/TestCase7/ROMNet_Input.py`  
to `$WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py`
- 7.2. In `$WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py`, change:
  - 7.2.1. `"self.WORKSPACE_PATH = ..."`
- 7.3. Move to `$WORKSPACE_PATH/ROMNet/romnet/app/`
- 7.4. Run: `"python3 ROMNet.py ../input/`
- 7.5. Postprocess results via: `$WORKSPACE_PATH/ROMNet/romnet/scripts/postprocessing/TransTanh/DeepONet/Predict_DeepONet.ipynb`

# A Translating Hyperbolic Tangent Test Case



## **Test Case 8: Physics-Informed improved deep operator network (DeepONet) for predicting position**

- 8.1. Copy \$WORKSPACE\_PATH/ROMNet/romnet/input/TransTanh/DeepONet/TestCase8/ROMNet\_Input.py to \$WORKSPACE\_PATH/ROMNet/romnet/input/ROMNet\_Input.py
- 8.2. In \$WORKSPACE\_PATH/ROMNet/romnet/input/ROMNet\_Input.py, change:
  - 8.2.1. "self.WORKSPACE\_PATH = ..."
- 8.3. Move to \$WORKSPACE\_PATH/ROMNet/romnet/app/
- 8.4. Run: "python3 ROMNet.py ../input/"
- 8.5. Postprocess results via: \$WORKSPACE\_PATH/ROMNet/romnet/scripts/postprocessing/TransTanh/DeepONet/Predict\_DeepONet.ipynb

The input file for Test Case 7 differs from the one of Test Case 5 Part 4 for:

**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