Test Cases for the Mass-Spring-Damper System

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 Mass-Spring-Damper Test Case

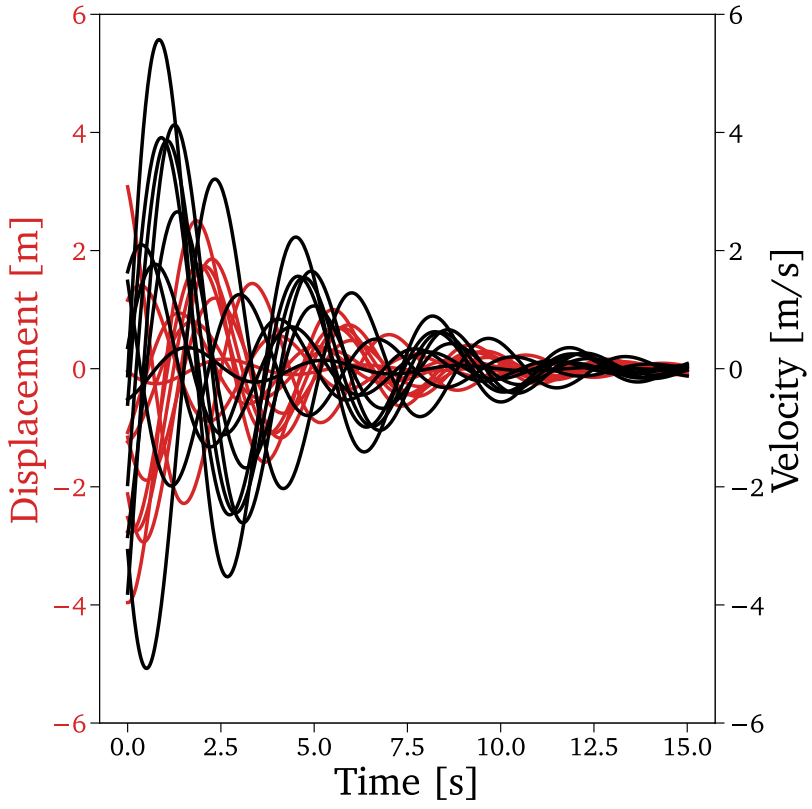
Equations of motion:

$$\begin{cases} m\ddot{x} + c\dot{x} + kx = 0, \\ x(t=0) = x_0, \\ \dot{x}(t=0) = v_0, \end{cases}$$

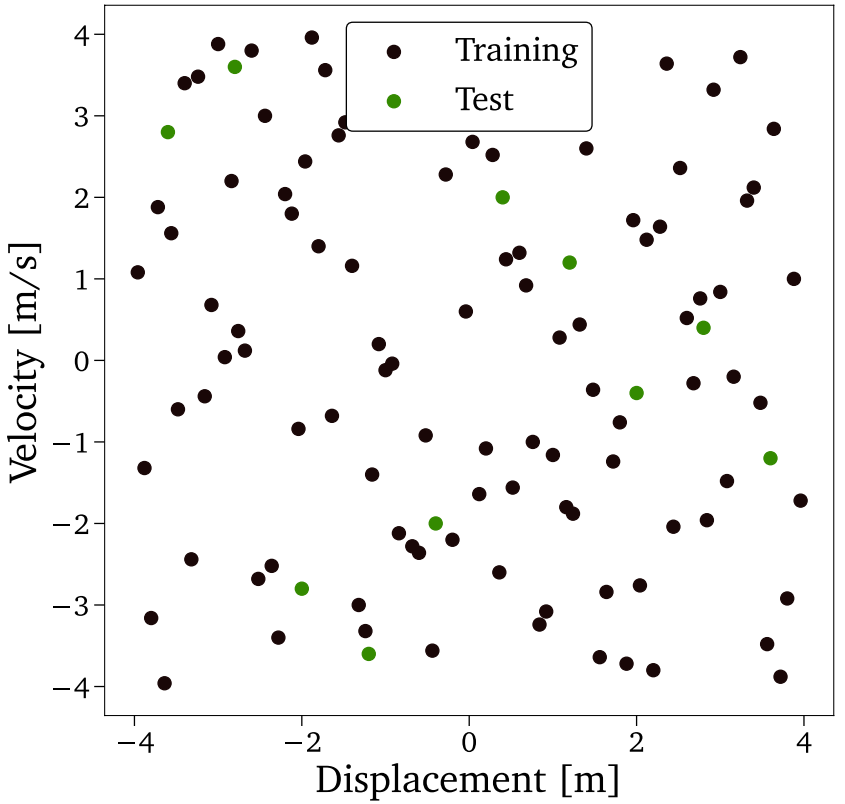
which can be rewritten as:

$$\begin{cases} \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \\ x(t=0) = x_0, \\ \dot{x}(t=0) = v_0. \end{cases}$$

Some training scenarios



Initial conditions



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

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

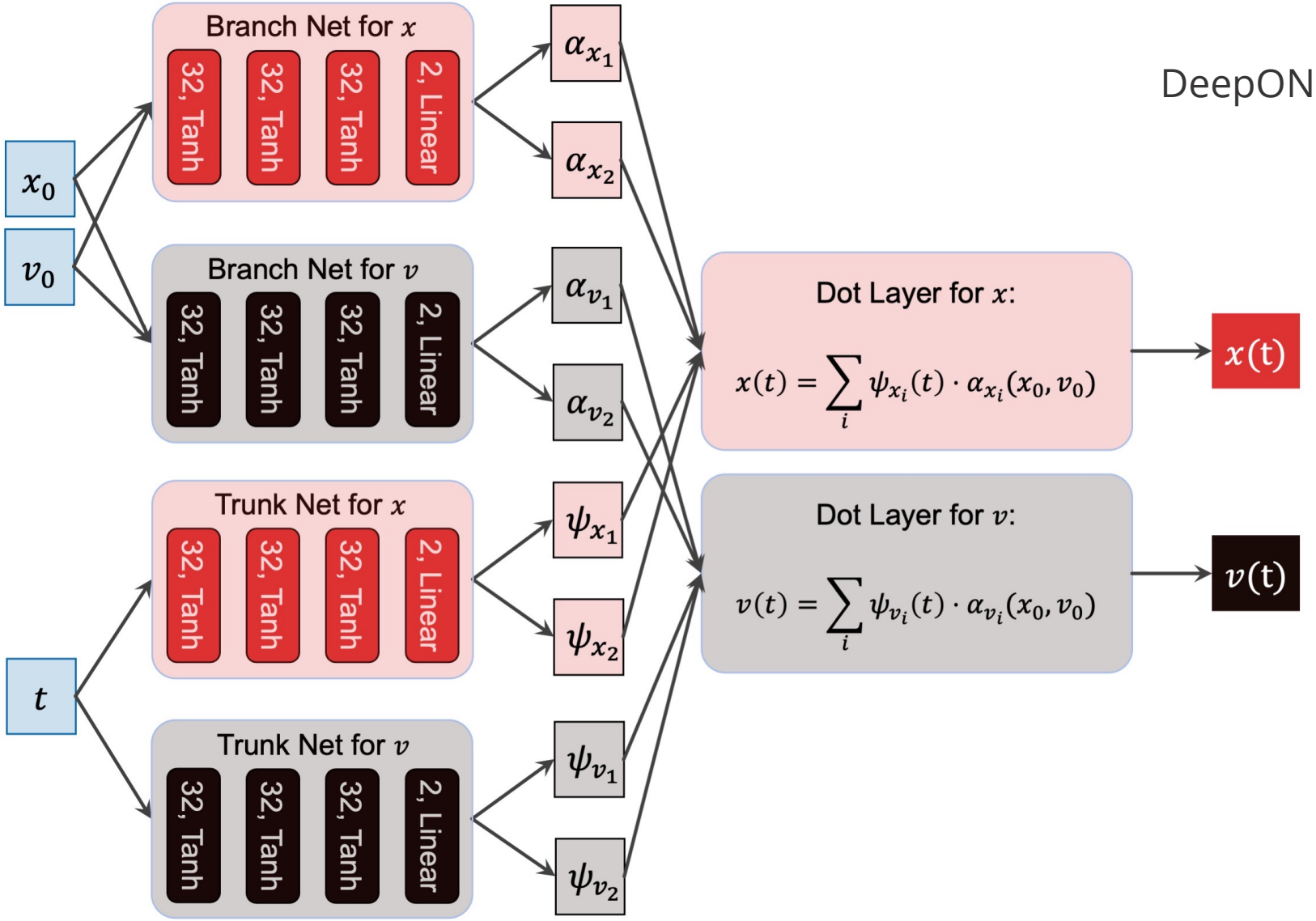
A Mass-Spring-Damper Test Case

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

A Mass-Spring-Damper Test Case

Test Cases 1 & 2

A Mass-Spring-Damper Test Case

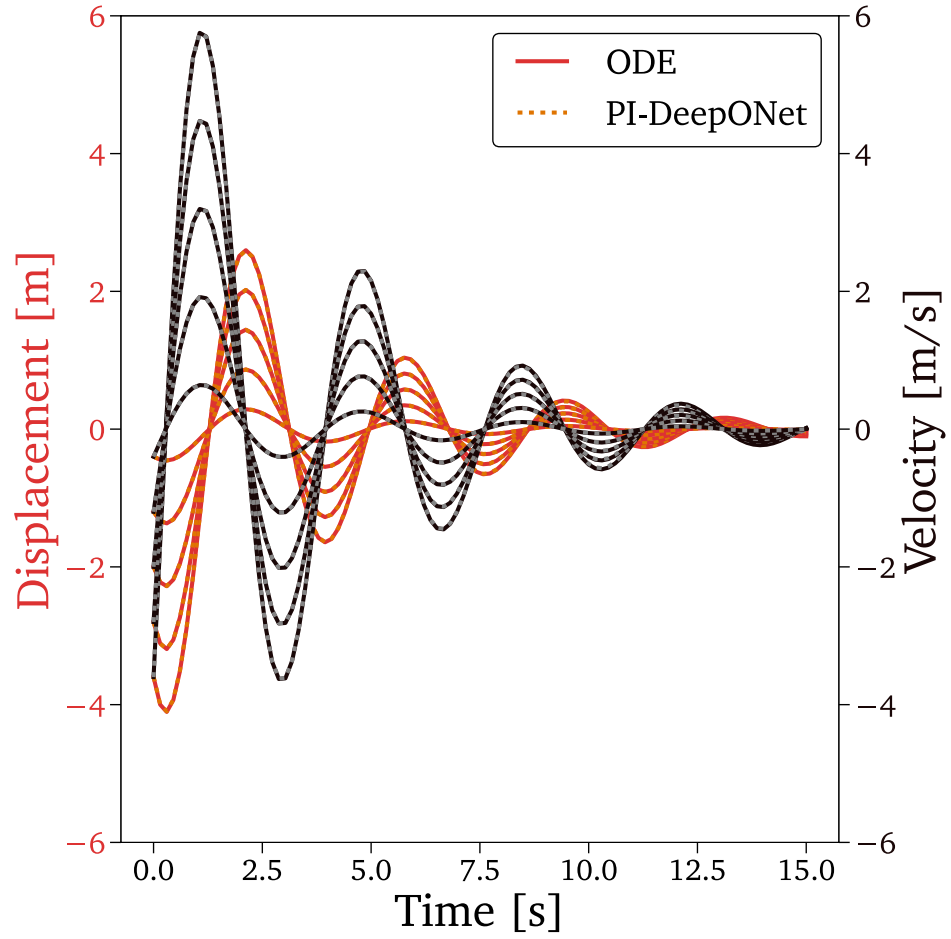


DeepONet Structure

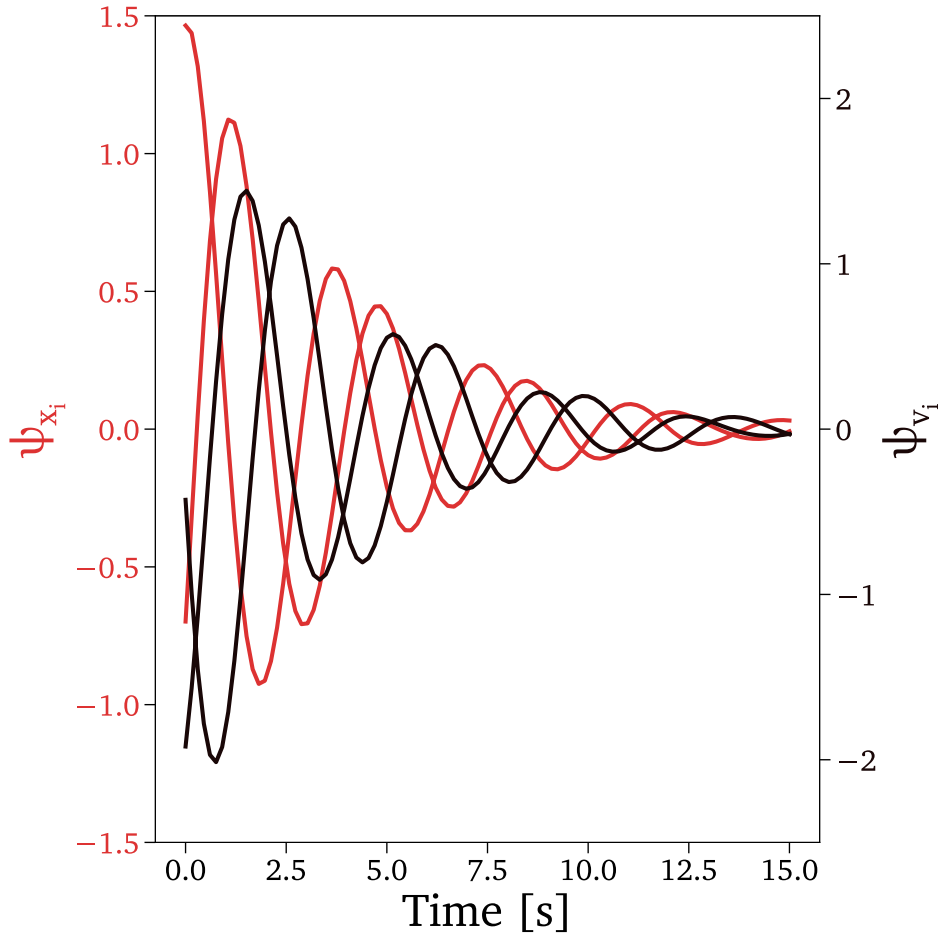
A Mass-Spring-Damper Test Case

Test Case 1: Fully data-drive training
Test Case 2: Trained with a physics-informed loss

Test cases



Trunk nets' outputs (~dynamical modes)



A Mass-Spring-Damper Test Case

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

1.1. Copy $WORKSPACE_PATH/ROMNet/romnet/input/MassSpringDamper/DeepONet/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/MassSpringDamper/DeepONet/Predict_DeepONet.ipynb

A Mass-Spring-Damper Test Case

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

2.1. Copy $WORKSPACE_PATH/ROMNet/romnet/input/MassSpringDamper/DeepONet/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:
2.2.1. "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/MassSpringDamper/DeepONet/Predict_DeepONet.ipynb

In the input file:

self. **surrogate_type** controls the type of surrogate

self.**structure** is a dictionary that controls the structure of the surrogate

Surrogates:

- FNN: Feed-Forward Neural Network
- DeepONet: Deep Operator Network
- Double_DeepONet: Two DeepONets in Series

System of Components:

- FNN
- DeepONet

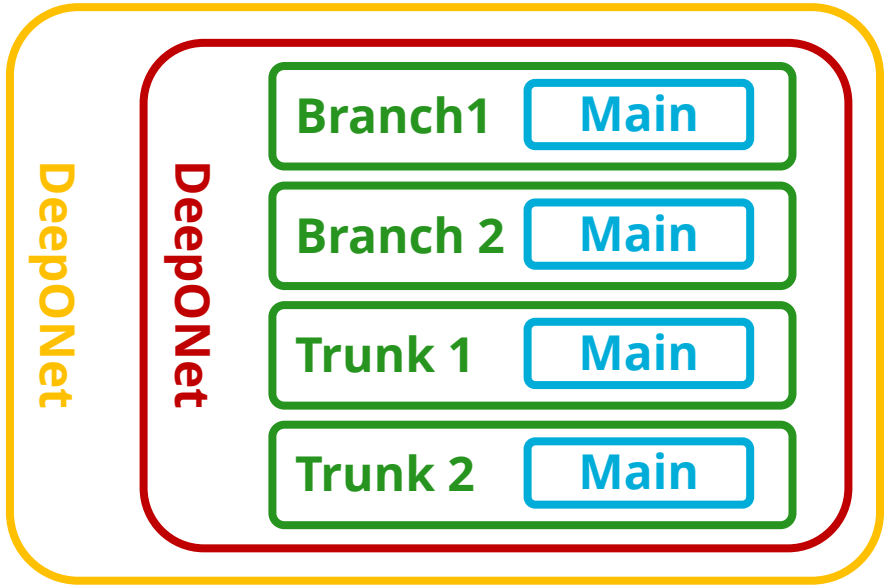
Components:

- FNN
- Branch
- Branch_i
- Trunk
- Trunk_i

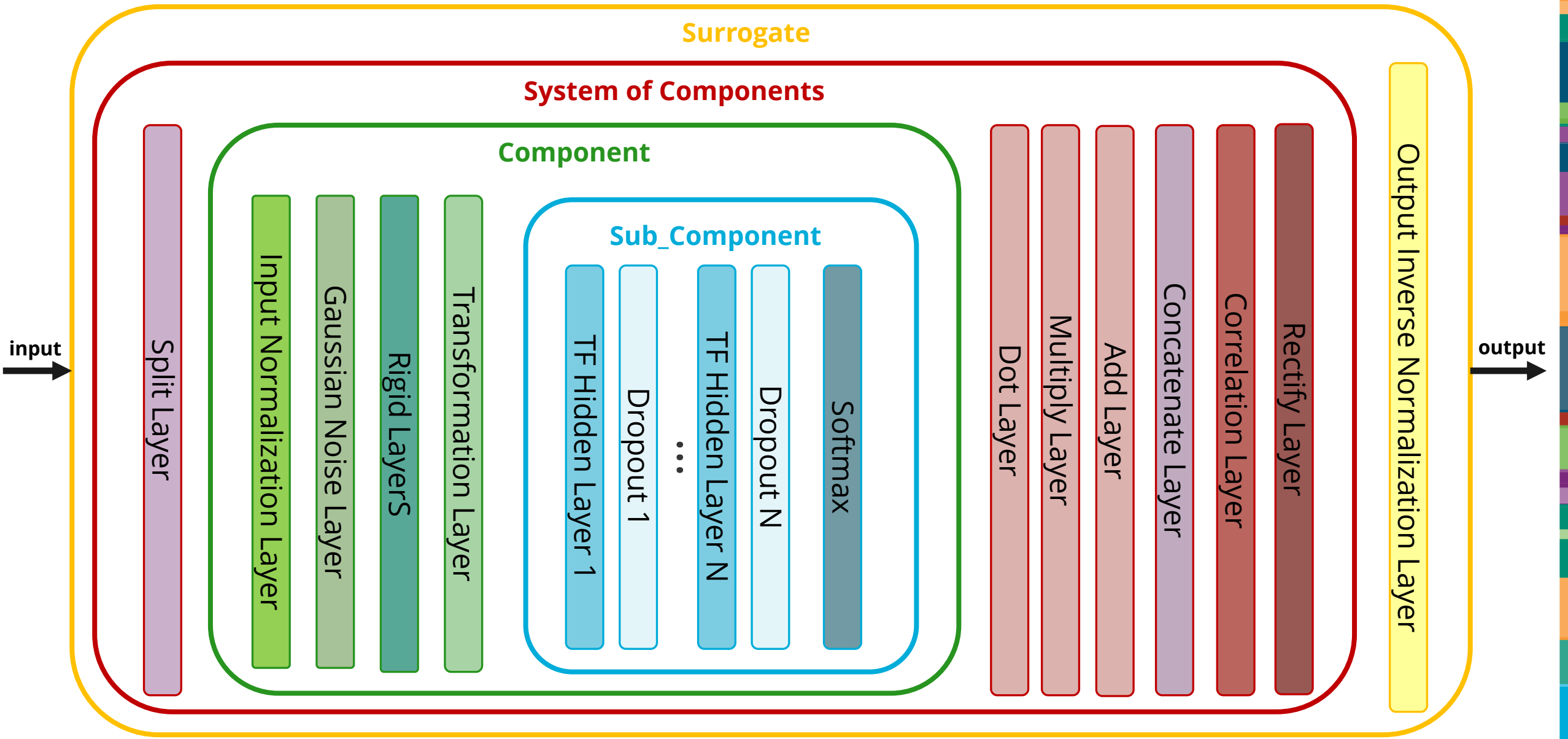
Sub-Components:

- Main
- U
- V

In these test cases:



The related classes can be found in
$WORKSPACE_PATH/ROMNet/romnet/romnet/nn/



The input files for Test Case 1 and 2 differ only for:

self.**n_train (**i.e. Type/No of Data Points)

- 'pts': data point
- 'ics': ODE's initial conditions
- 'res': ODE residual

self. **losses** (i.e., Dictionary Containing Loss Functions for Each Data Type)

self.**loss_weights** (i.e., Dictionary Containing Weights for Each Data Type)

A Mass-Spring-Damper Test Case

Test Case 3

A Mass-Spring-Damper Test Case

A scenario-aggregated principal component analysis (PCA) analogy

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

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

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

No of time instants

No of scenarios

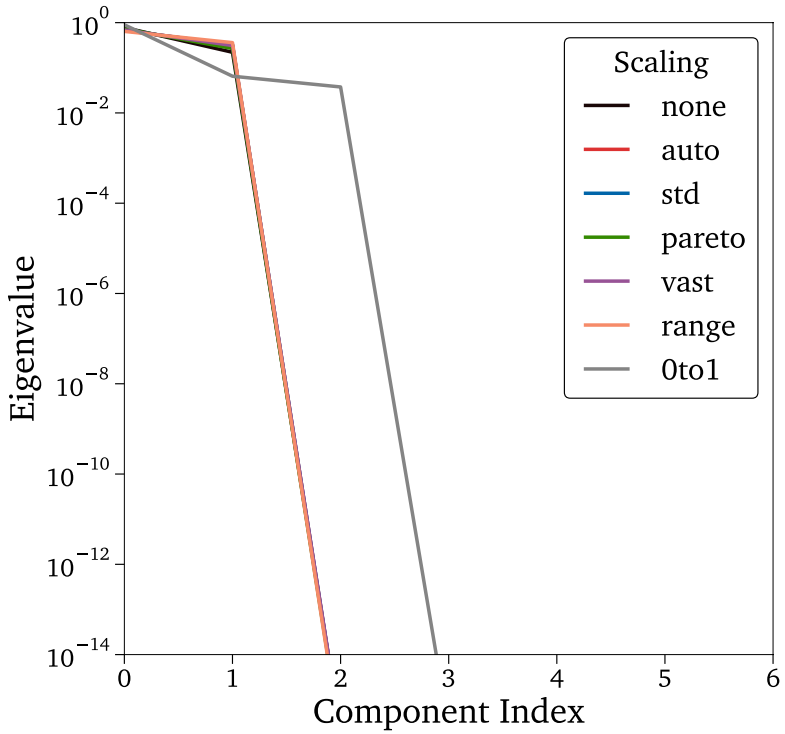
$$\boldsymbol{V} = \begin{bmatrix} | & | & & | & | \\ \boldsymbol{v_1} & \boldsymbol{v_2} & \cdots & \boldsymbol{v_{99}} & \boldsymbol{v_{100}} \\ | & | & & | & | \end{bmatrix}$$

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

A Mass-Spring-Damper Test Case

Eigenvector Decomposition of the Covariance Matrix of X ($\mathbf{R_X} = \frac{\mathbf{XX^T}}{\mathbf{N_S - 1}}$):

Analyzing the eigenvalues:



Eigenvalues of $\mathbf{R_x}$

$$\mathbf{\Lambda_x = \Psi_x^{-1} R_x \Psi_x}$$

Orthonormal eigenvectors of $\mathbf{R_x}$

Two principal components ($N_\psi = 2$) are sufficient for fully characterizing all the 100 scenarios

Note: Equivalent results obtained for the PCA of $\mathbf{R_V}$

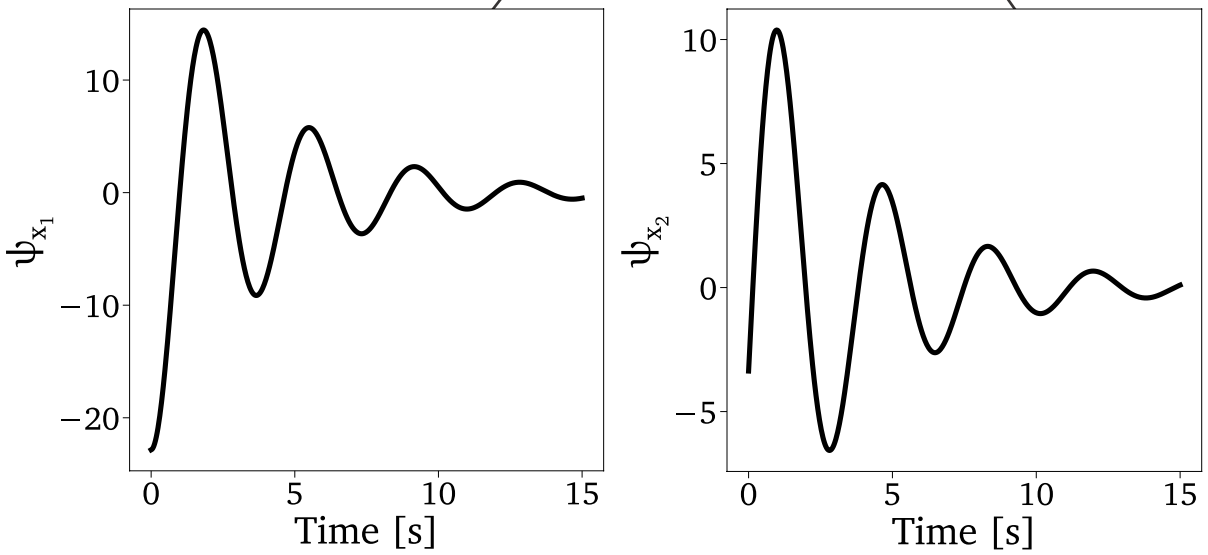
A Mass-Spring-Damper Test Case

Eigenvector Decomposition of the Covariance Matrix of X ($R_x = \frac{XX^T}{N_S - 1}$):

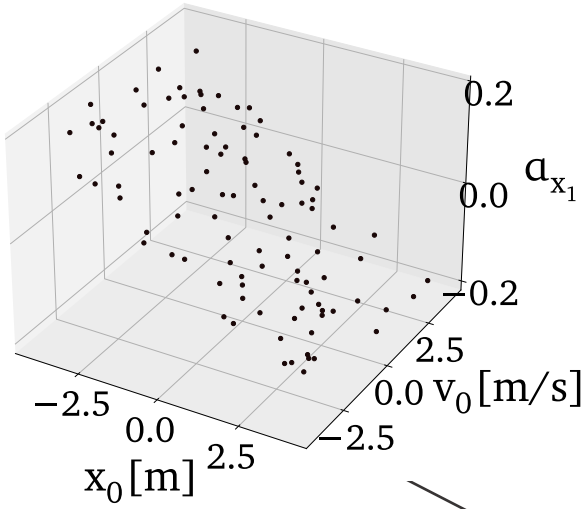
$$\Psi_x = (X - C_x) \cdot D_x^{-1} A_x = \begin{bmatrix} \psi_{x_1} & \psi_{x_2} \end{bmatrix}$$

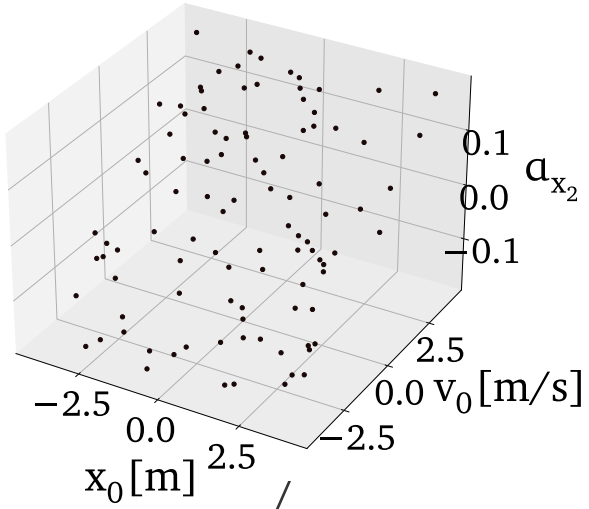
Where:

$$\dim(\Psi_x) = N_t \times N_\psi = 500 \times 2$$



Analyzing the modes and the projection matrix





$$A_x = \begin{bmatrix} \alpha_{x_1} & \alpha_{x_2} \end{bmatrix}$$

Where:

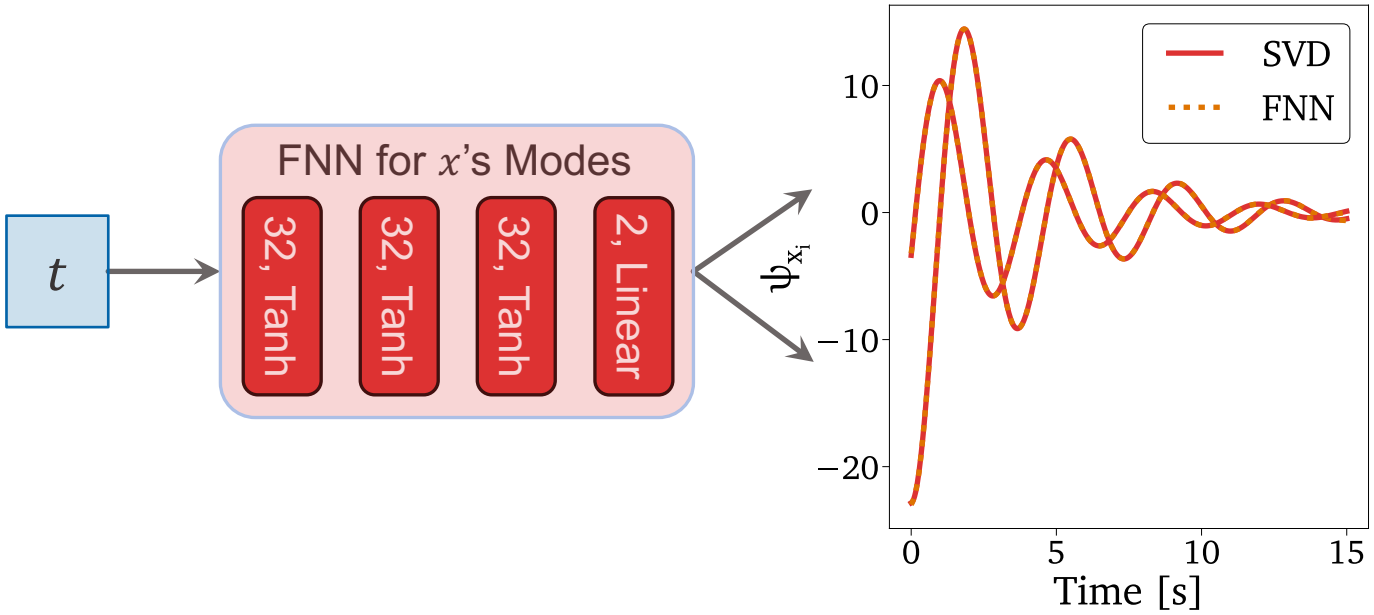
$$\dim(A_x) = N_s \times N_\psi = 100 \times 2$$

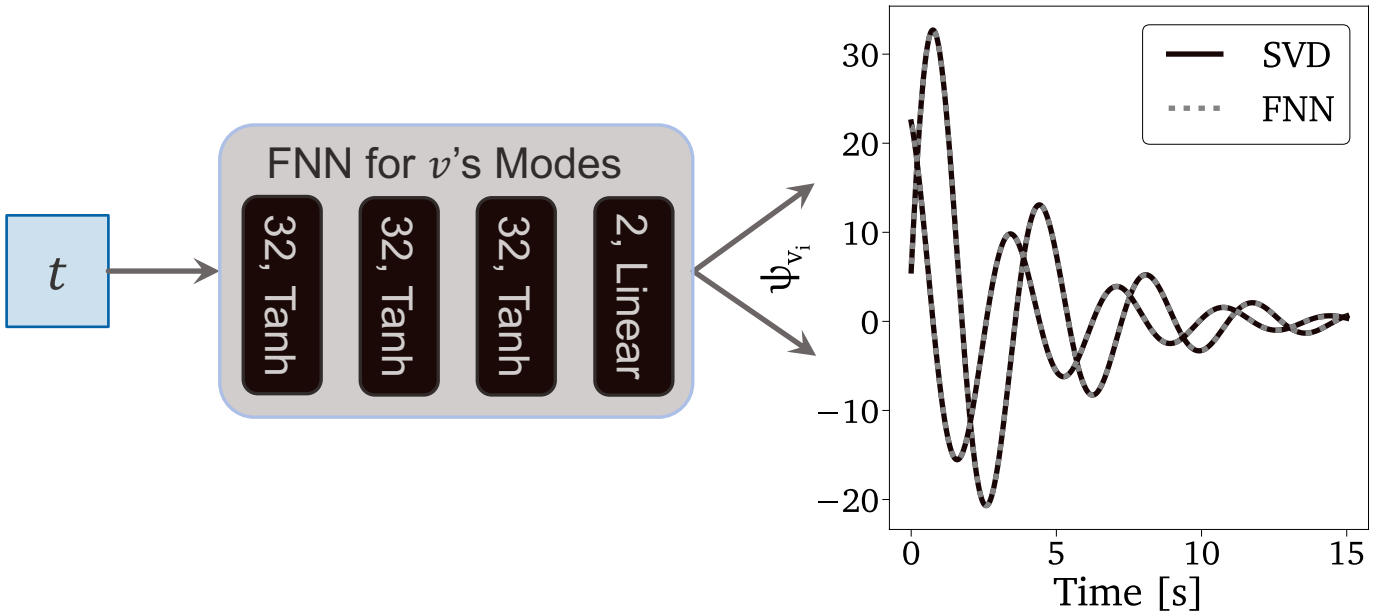
A Mass-Spring-Damper Test Case

Run Jupyter Notebook
$WORKSPACE_PATH/ROMNet/romnet/scripts/generating_data/MassSpringDamper/Generate_Data_2.ipynb
for generating PCA training and test data

Note: The script needs to be run twice, the second time after changing mode_name and i_var

A Mass-Spring-Damper Test Case





Fitted the modes of x and v with two independent feed-forward neural networks

A Mass-Spring-Damper Test Case

Test Case 3: PCA-based interpreation of DeepONets:

Parts I and II: Train Trunks

3.1. Copy $WORKSPACE_PATH/ROMNet/romnet/input/PCA/MassSpringDamper/FNN/Trunk/TestCase3_Part1/ROMNet_Input.py
to $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py

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

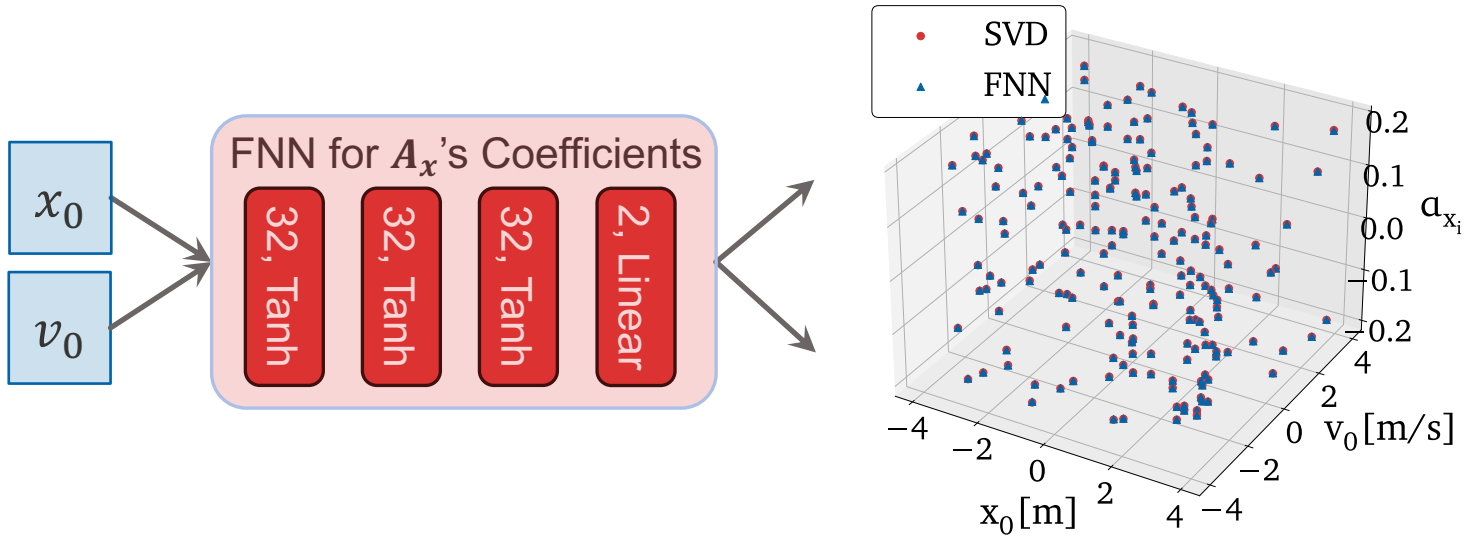
3.3. Move to $WORKSPACE_PATH/ROMNet/romnet/app/

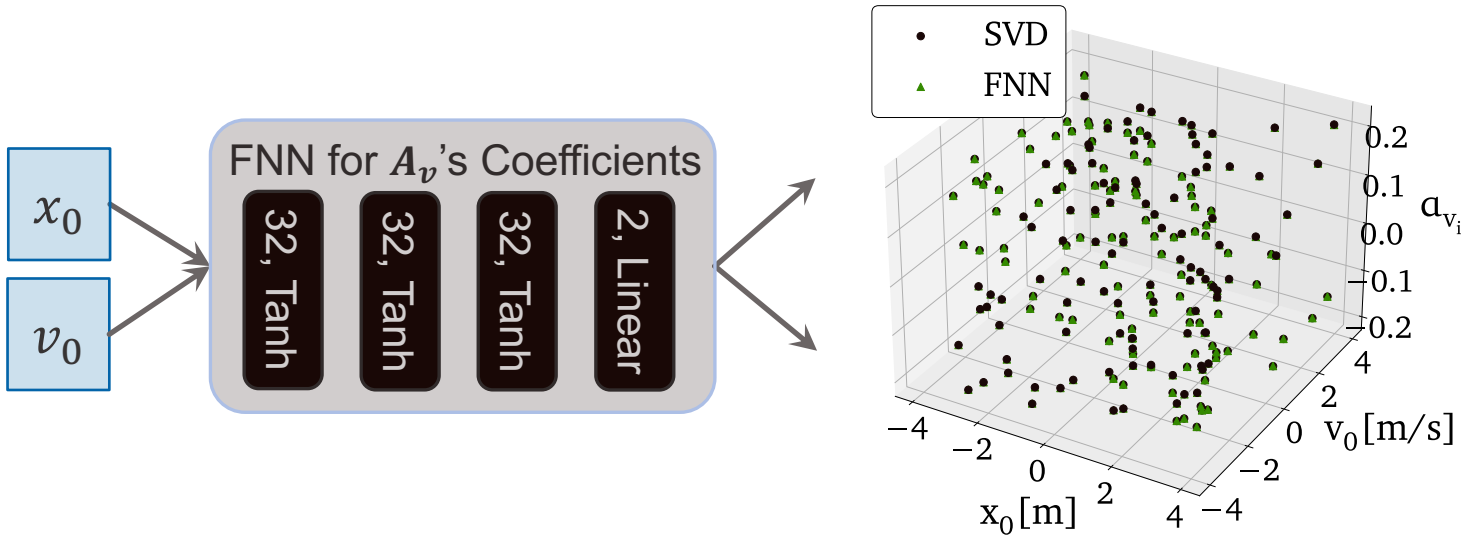
3.4. Run: "python3 ROMNet.py ../input/"

3.5. Postprocess results via: $WORKSPACE_PATH/ROMNet/romnet/scripts/postprocessing/PCA/MassSpringDamper/FNN/Predict_FNN_Trunk.ipynb

REPEAT for the second Trunk (i.e., $WORKSPACE_PATH/ROMNet/romnet/input/PCA/MassSpringDamper/FNN/Trunk/TestCase3_Part1/ROMNet_Input.py)

A Mass-Spring-Damper Test Case





Fitted the $\mathbf{A}_x$ and $\mathbf{A}_v$ components with two independent feed-forward neural networks

A Mass-Spring-Damper Test Case

Test Case 3: PCA-based interpreation of DeepONets:

Parts III and IV: Train Branches

3.1. Copy $WORKSPACE_PATH/ROMNet/romnet/input/PCA/MassSpringDamper/FNN/Branch/TestCase3_Part3/ROMNet_Input.py
to $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py

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

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

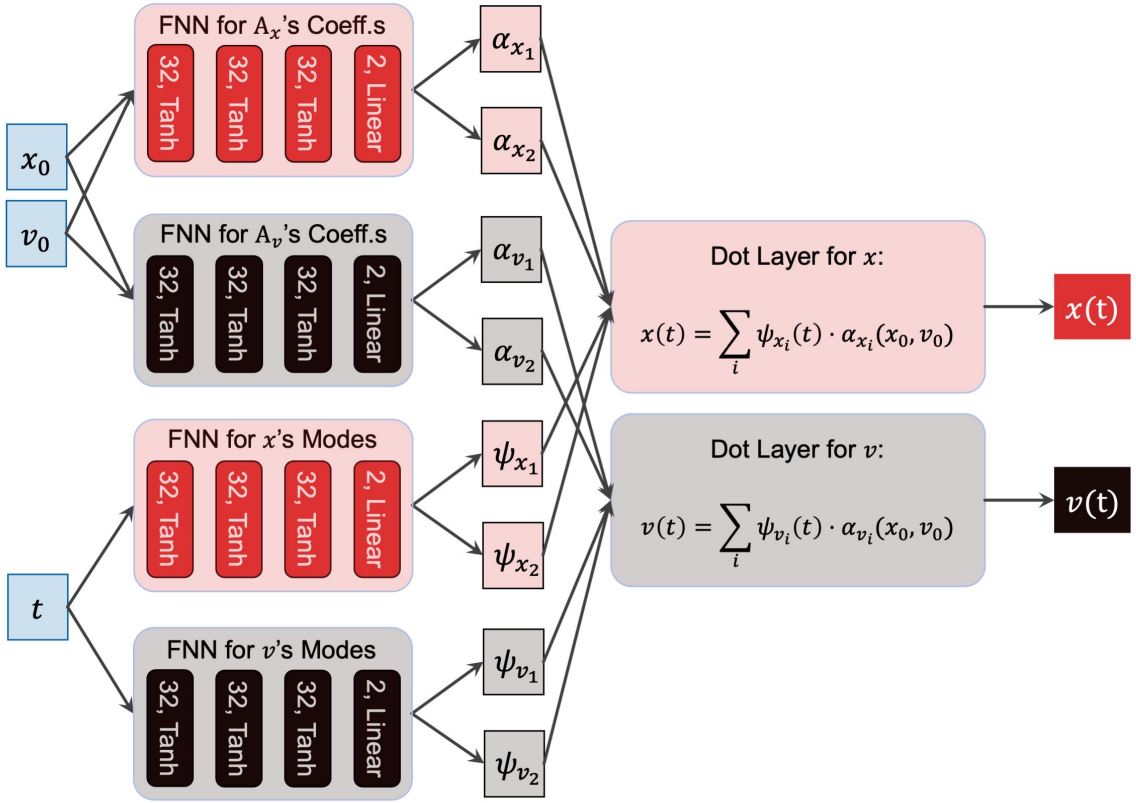
3.4. Run: "python3 ROMNet.py ../input/"

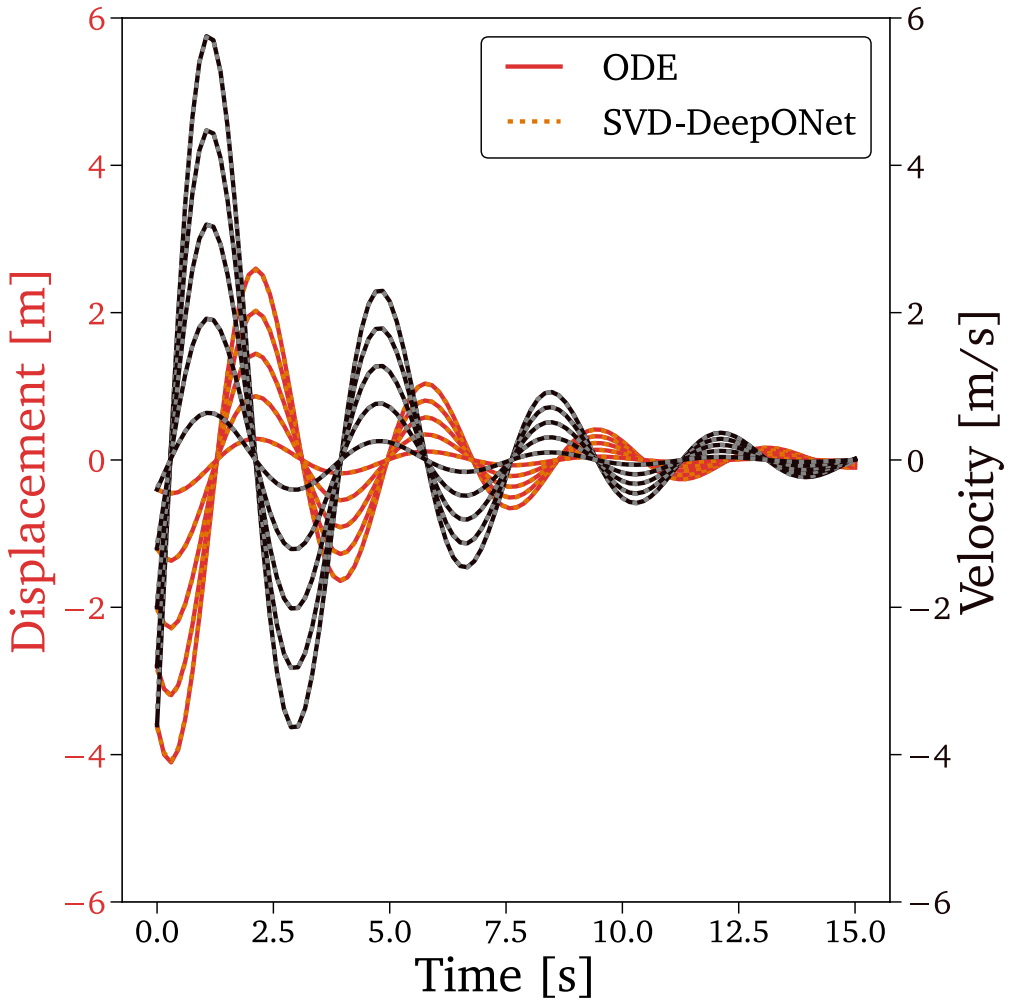
3.5. Postprocess results via: $WORKSPACE_PATH/ROMNet/romnet/scripts/postprocessing/PCA/MassSpringDamper/FNN/Predict_FNN_Branch.ipynb

REPEAT for the second Branch (i.e., $WORKSPACE_PATH/ROMNet/romnet/input/PCA/MassSpringDamper/FNN/Branch/TestCase3_Part4/ROMNet_Input.py)

A Mass-Spring-Damper Test Case

Generative DeepONet:
uploaded the parameters of the four FNN's
as weights and biases of
the corresponding trunk and branch nets
and predicted test scenarios (w/o any additional training)





A Mass-Spring-Damper Test Case

Note: If correctly executed, Predict_FNN_Branch.ipynb and Predict_FNN_Trunk.ipynb created the file:
$WORKSPACE_PATH/ROMNet/Data/MSD_100Cases/Orig/OneByOne/FNN/Final.h5,
which contains the trained parameter values for branches and trunk.

Test Case 3: PCA-based interpreation of DeepONets:

Part V: Generate the DeepONet

3.1. Copy $WORKSPACE_PATH/ROMNet/romnet/input/MassSpringDamper/DeepONet/TestCase3_Part5/ROMNet_Input.py
to $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py

3.2. In $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py, change:
3.2.1. "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/MassSpringDamper/DeepONet/Predict_DeepONet.ipynb

The input files for Test Case 1 differs from the one of Test Case 3 Part 5 for:

self.**path_to_load_fld**: we are now including a path for uploading pre-trained weights

self.**trainable_flg**: we are not training the parameters

A Mass-Spring-Damper Test Case

Test Case 4

A Mass-Spring-Damper Test Case

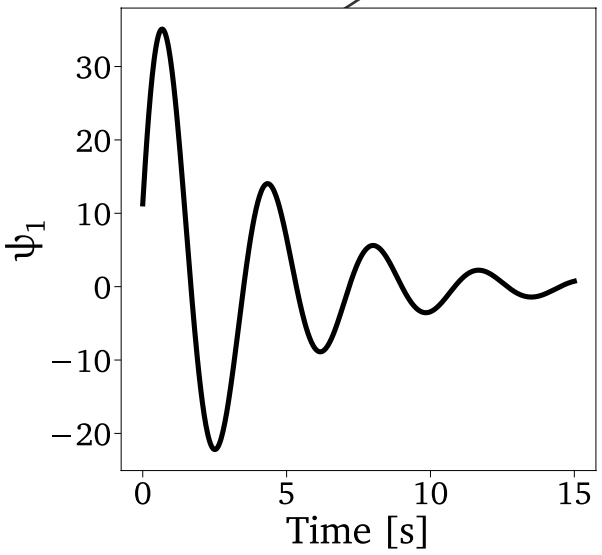
The scenario aggregation can be performed based on one single QoI (as shown so far), resulting in N_y separate trunks, or it can be executed by including all the QoIs simultaneously, resulting in one single trunk.

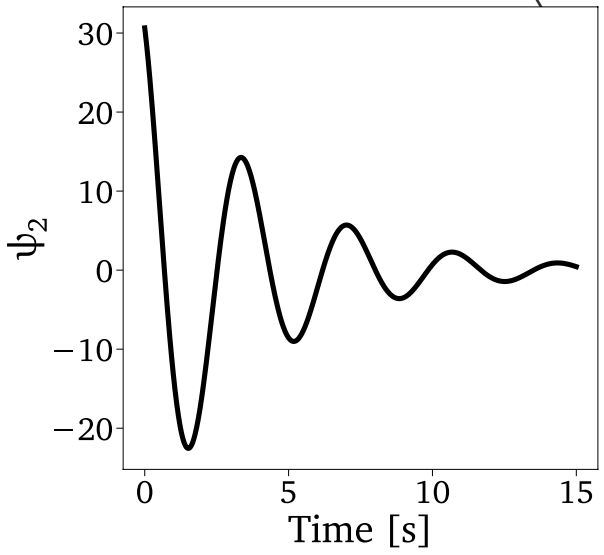
$$\boldsymbol{XV} = \begin{bmatrix} | & | & & | & | & | & | & & | & | \\ \boldsymbol{x_1} & \boldsymbol{x_2} & \cdots & \boldsymbol{x_{99}} & \boldsymbol{x_{100}} & \boldsymbol{v_1} & \boldsymbol{v_2} & \cdots & \boldsymbol{v_{99}} & \boldsymbol{v_{100}} \\ | & | & & | & | & | & | & & | & | \end{bmatrix}$$

$$\mathbf{dim}(\boldsymbol{XV}) = \boldsymbol{N_t} \times \boldsymbol{2N_s}$$

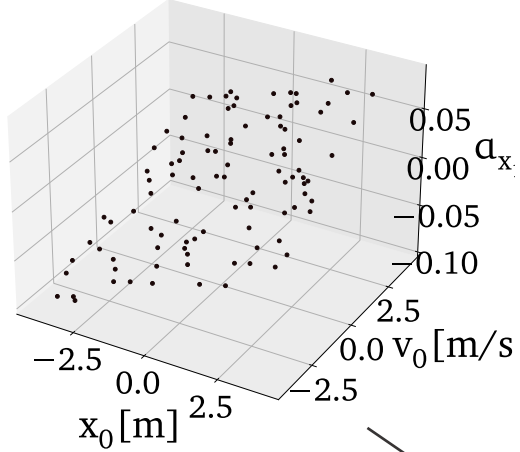
A Mass-Spring-Damper Test Case

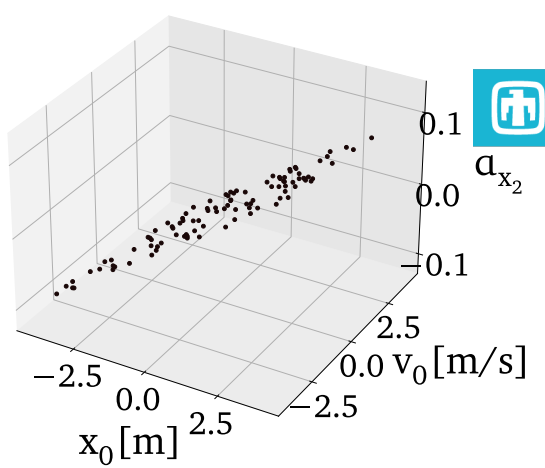
$$\boldsymbol{\Psi_{xv}} = (\boldsymbol{XV} - \boldsymbol{C_{xv}}) \cdot \boldsymbol{D_{xv}^{-1}}\, \boldsymbol{A_{xv}} = \begin{bmatrix} | & | \\ \psi_1 & \psi_2 \\ | & | \end{bmatrix}$$

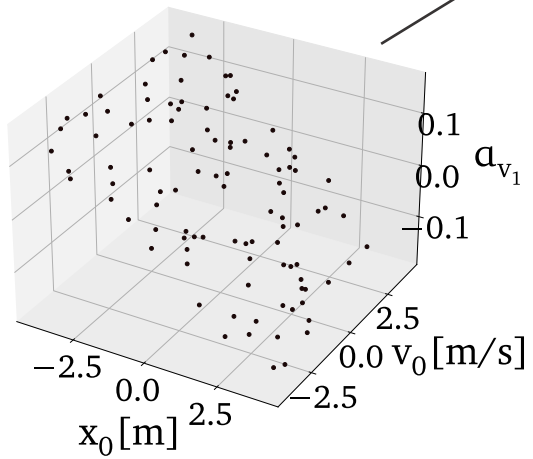


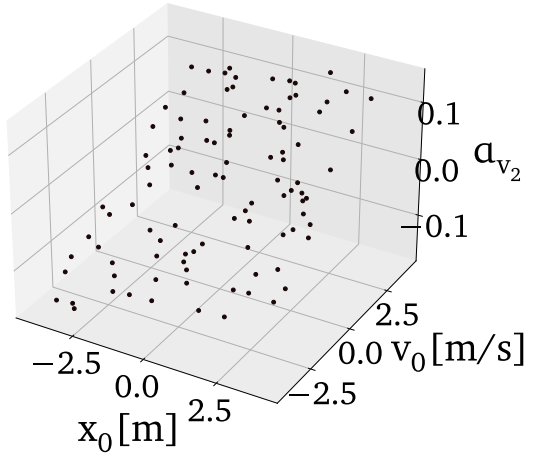


$$\boldsymbol{A_{xv}} = \begin{bmatrix} | & | \\ \alpha_{xv_1} & \alpha_{xv_2} \\ | & | \end{bmatrix} = \begin{bmatrix} | & | \\ \alpha_{x_1} & \alpha_{x_2} \\ | & | \\ \alpha_{v_1} & \alpha_{v_2} \\ | & | \end{bmatrix}$$





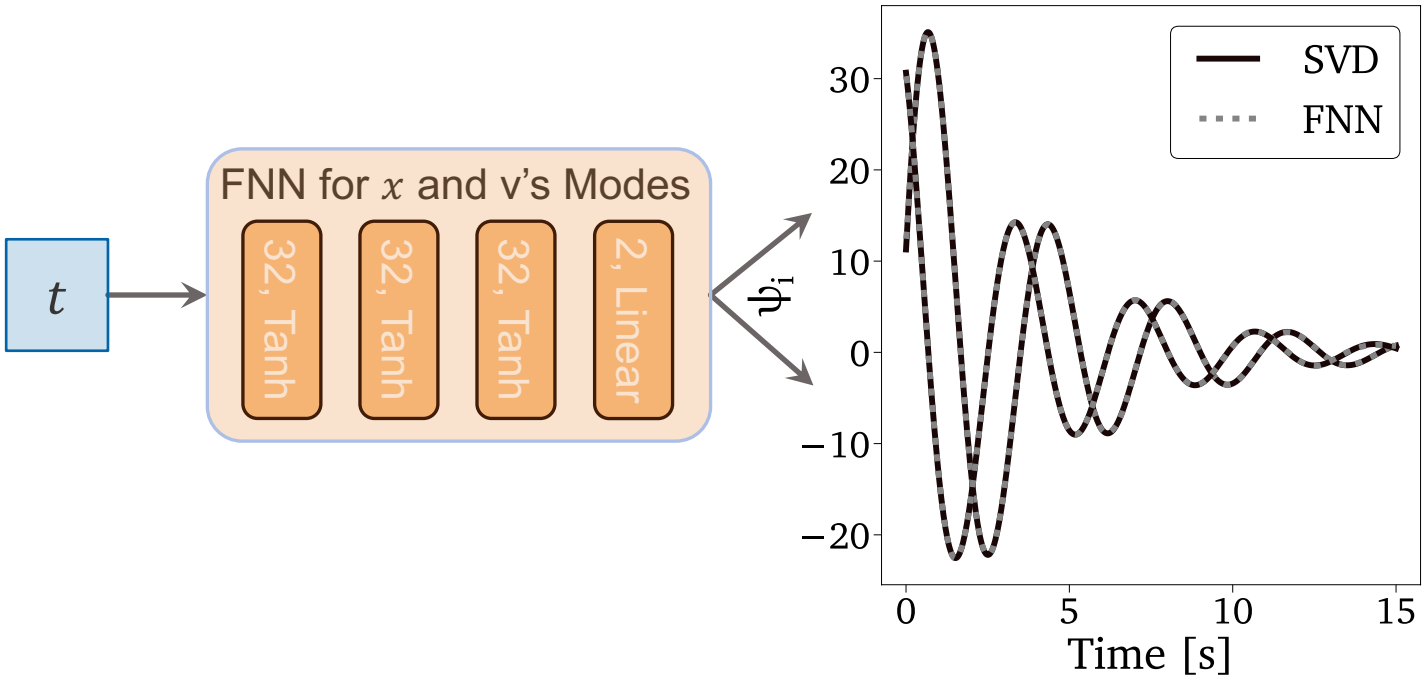




A Mass-Spring-Damper Test Case

Run Jupyter Notebook
$WORKSPACE_PATH/ROMNet/romnet/scripts/generating_data/MassSpringDamper/Generate_Data_2_All.ipynb
for generating PCA training and test data

A Mass-Spring-Damper Test Case



Fitted the modes of XV
with one
feed-forward neural network

A Mass-Spring-Damper Test Case

Test Case 4: PCA-based interpreation of DeepONets – Shared Trunk:

Part 1: Train the Trunk

4.1. Copy $WORKSPACE_PATH/ROMNet/romnet/input/PCA/MassSpringDamper/FNN/Trunk/TestCase4_Part1/ROMNet_Input.py
to $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py

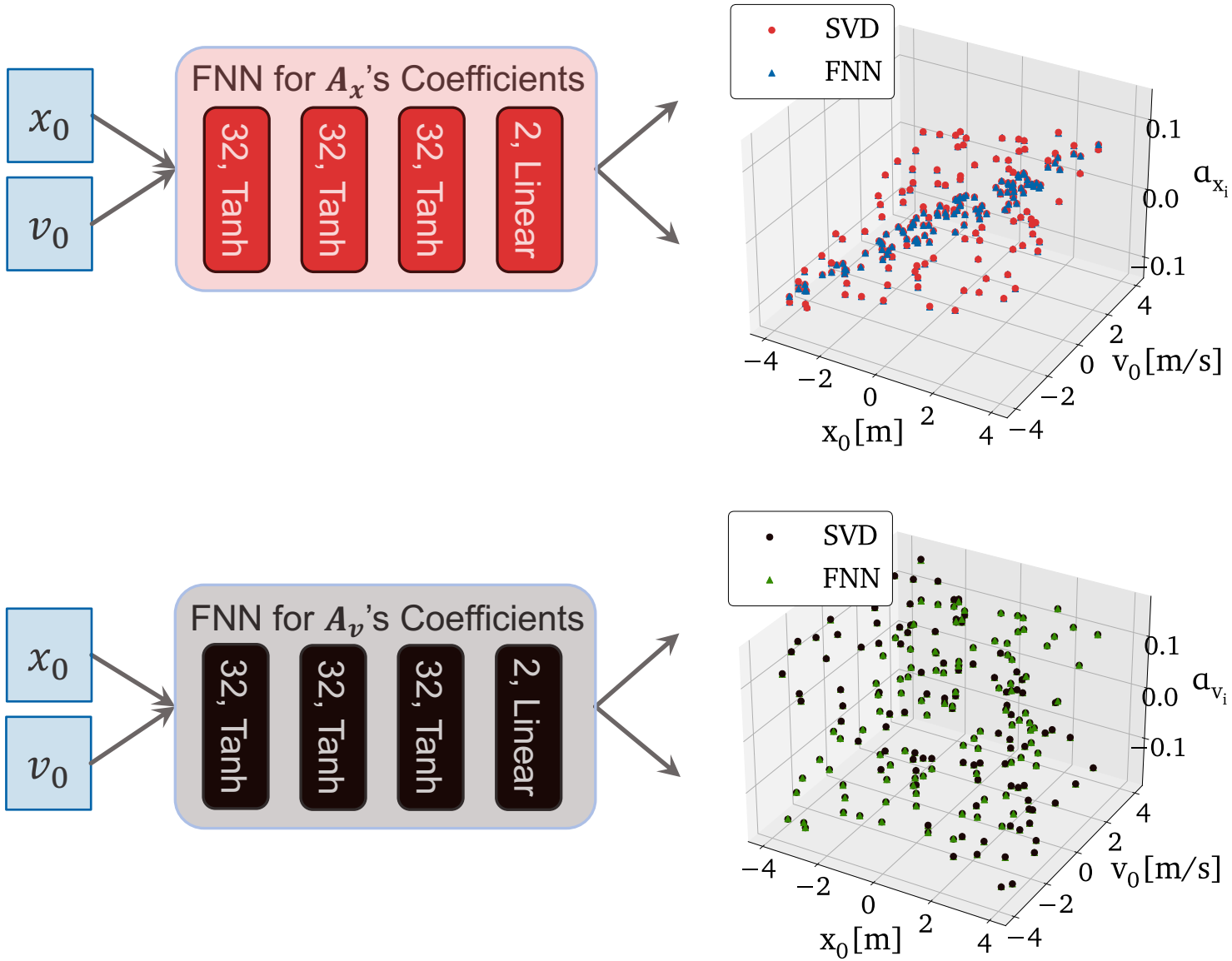
4.2. In $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py, change:
4.2.1. "self.WORKSPACE_PATH = ..."

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/PCA/MassSpringDamper/FNN/Predict_FNN_Trunk.ipynb

A Mass-Spring-Damper Test Case



Fitted the $\mathbf{A}_x$ and $\mathbf{A}_v$ components with two independent feed-forward neural networks

A Mass-Spring-Damper Test Case

Test Case 4: PCA-based interpreation of DeepONets – Shared Trunk:

Parts II and III: Train Branches

4.1. Copy $WORKSPACE_PATH/ROMNet/romnet/input/PCA/MassSpringDamper/FNN/Branch/TestCase4_Part2/ROMNet_Input.py
to $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py

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

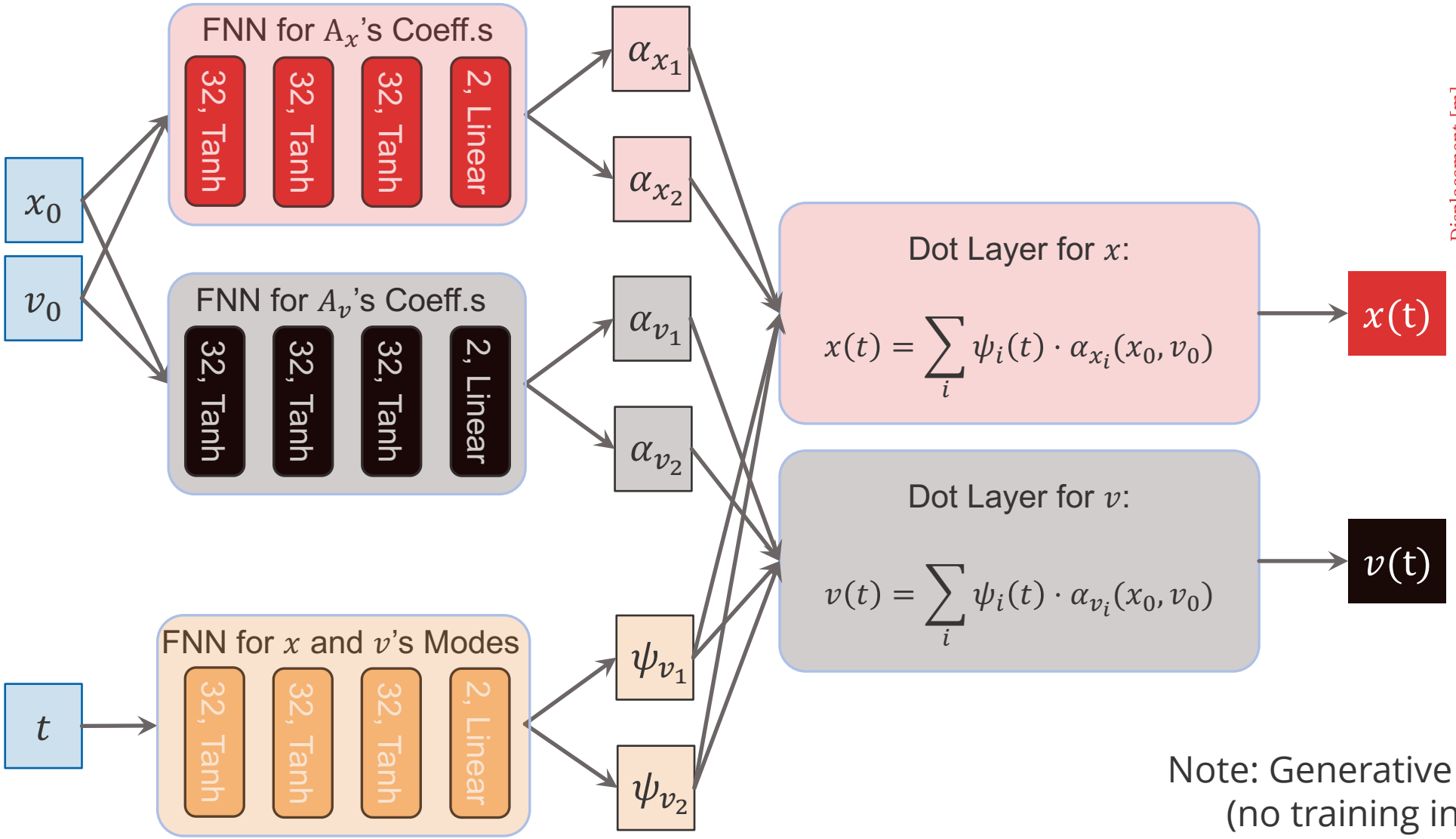
4.3. Move to $WORKSPACE_PATH/ROMNet/romnet/app/

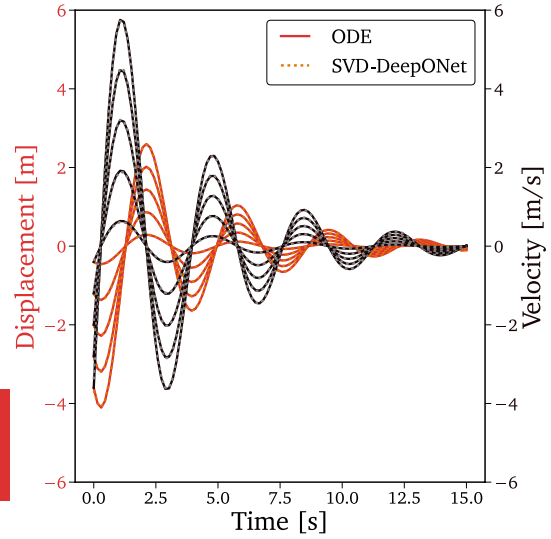
4.4. Run: "python3 ROMNet.py ../input/"

4.5. Postprocess results via: $WORKSPACE_PATH/ROMNet/romnet/scripts/postprocessing/PCA/MassSpringDamper/FNN/Predict_FNN_Branch.ipynb

REPEAT for the second Branch (i.e., $WORKSPACE_PATH/ROMNet/romnet/input/PCA/MassSpringDamper/FNN/Branch/TestCase4_Part3/ROMNet_Input.py)

A Mass-Spring-Damper Test Case





Note: Generative DeepONet
(no training involved)

A Mass-Spring-Damper Test Case

Note: If correctly executed, Predict_FNN_Branch.ipynb and Predict_FNN_Trunk.ipynb created the file:
$WORKSPACE_PATH/ROMNet/Data/MSD_100Cases/Orig/All/FNN/Final.h5,
which contains the trained parameter values for branches and trunk.

Test Case 4: PCA-based interpreation of DeepONets – Shared Trunk:

Part IV: Generate the DeepONet

4.1. Copy $WORKSPACE_PATH/ROMNet/romnet/input/MassSpringDamper/DeepONet/TestCase4_Part4/ROMNet_Input.py
to $WORKSPACE_PATH/ROMNet/romnet/input/ROMNet_Input.py

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

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

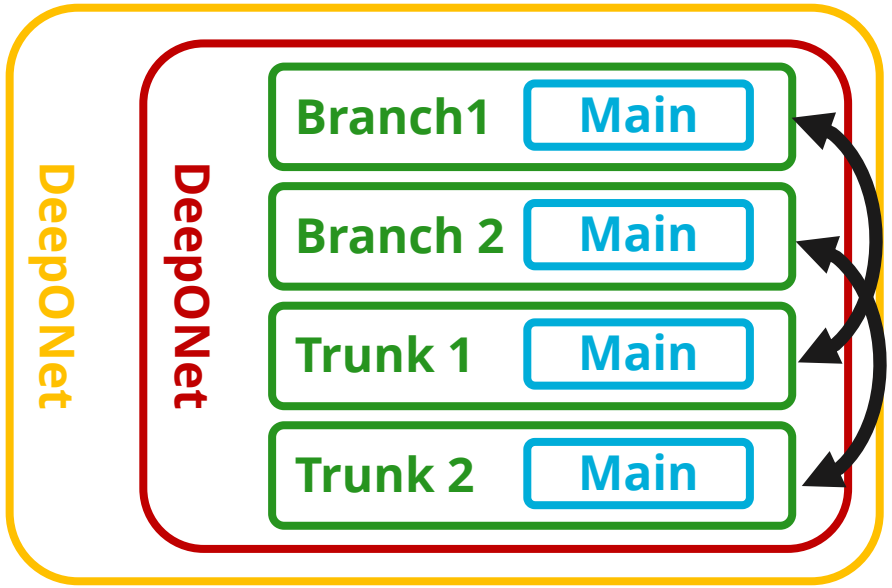
4.4. Run: "python3 ROMNet.py ../input/"

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

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

self.**branch_to_trunk**: DeepONet Branch-to-Trunk Type of Mapping ('one_to_one'/'multi_to_one')

Test Case 3 Part 5:



Test Case 4 Part 4: