Multibody System Dynamics manuscript No.
(will be inserted by the editor)

Data-driven simulation for general purpose multibody dynamics using deep neural networks

Hee-Sun Choi · Junmo An · Jin-Gyun Kim · Jae-Yoon Jung · Juhwan Choi · Grzegorz Orzechowski · Aki Mikkola · Jin Hwan Choi

Received: date / Accepted: date

Abstract In this paper, a machine learning based simulation framework of general purpose multibody dynamics is introduced. The aim of the framework is to generate a well trained meta-model of multibody dynamics (MBD) systems. To this end, deep neural network (DNN) is employed to the framework so as to construct data based meta model representing multibody systems. Constructing well defined training data set with time variable is essential to get accurate and reliable motion data such as displacement, velocity, acceleration, and forces. As a result of the introduced approach, the meta-model provides motion estimation of system dynamics without solving the analytical equations of motion. The performance of the proposed DNN meta-modeling was evaluated to represent several MBD systems.

Keywords Multibody dynamics · Meta-model · Deep neural network · Feed forward network · Data-driven simulation

Hee-Sun Choi, Jin-Gyun Kim, Jin Hwan Choi (Corresponding author)
Department of Mechanical Engineering, Kyung Hee University 1732, Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do 17104, Republic of Korea
E-mail: mdefptheia@gmail.com, jingyun.kim@khu.ac.kr, jhchoi@khu.ac.kr

Junmo An, Jae-Yoon Jung
Department of Industrial and Management Systems Engineering, Kyung Hee University 1732, Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do 17104, Republic of Korea
E-mail: ajm9306@khu.ac.kr, jyjung@khu.ac.kr

Juhwan Choi
R&D Center, FunctionBay, Inc. 5F, Pangyo Seven Venture Valley 1 danji 2 dong, 15, Pangyo-ro 228 beon-gil, Bundang-gu, Seongnam-si, Gyeonggi-do, Republic of Korea
E-mail: juhwan@functionbay.co.kr

Grzegorz Orzechowski, Aki Mikkola
Department of Mechanical Engineering, LUT University, Yliopistonkatu 34, 53850 Lappeenranta, Finland
E-mail: grzegorz.orzechowski@lut.fi, aki.mikkola@lut.fi

1 Introduction

Using Machine Learning (ML) with big data is an important subject matter in science and engineering. This is because ML is effective to handle and interpret big data sets for the purpose of finding certain patterns from the data.

In particular, Deep Neural Network (DNN), which is based on an Artificial Neural Network (ANN) with multiple hidden layers between input and output layers allows to handle complex shapes with nonlinear functions with multi-dimensional input data. DNN has been successfully used in a large number of practical applications. Well trained neural network then provides precise pattern recognition based on data sets in real time.

These features, big data recognition and real time estimation of nonlinear functions, of ML approaches are attractive to dynamics and control engineers who are handling nonlinear system dynamics with real world data. There have been several previous studies on applying ML, DNN, or other big-data handling techniques to rigid multibody system problems. For example, Bayesian formulation [4, 13, 15] in combination with Markov random field approximation, Kalman filter, or particle filter has been applied to various multibody dynamics (MBD) problems to handle noise data effectively in real-life applications, generate reliable modeling with efficient computational cost, estimate multibody system in probabilistic sense, or identify nonlinear parameters in governing equations. ML approaches [14, 16, 19, 17, 18] such as regression methods, reinforcement learning algorithms, and surrogate models have also been employed. Regression methods have many different types that can be performed in ML. In addition to the simple linear regression model, one can select and use techniques such as polynomial regression, support vector regression, decision tree regression, and random forest regression to suit a given problem. Based on the investigated input-label values, surrogate models perform a probabilistic estimate for an unknown objective function. This is an approach that uses an interpretable model to describe complex models. The most commonly used model in surrogate models is the Gaussian process. The proposed method has enhanced accuracy of prediction, especially in the long time scales, and increased computational efficiency in simulating dynamic response of multibody system. Moreover, neural networks [19, 20, 21, 22, 23] have been suggested as effective alternatives to multibody dynamics simulation in comparison with conventional algorithms. The approaches have been proved to be fast and reliable to describe and predict characteristics of multibody systems.

It is important to note that previous studies [4, 17, 18, 20, 21, 22, 23] are focused on particular MBD problems, mainly on contact, railways, vehicles, gaits, robotics, or tracking. Accordingly, a general MBD problem has not been introduced and analyzed through DNN technique.

To address these shortcomings, this study introduces a procedure to generate a solver based on *DNN meta-model* for *general* purpose multibody system, which allows us to predict MBD with high accuracy in real time. Among the various ML methods, a supervised learning technique is used for the mathematical and/or numerical data set of the MBD model in the training process. Data preparation and training process are called *off-line* stage, and its trained result is known as *meta-model*. Using the meta-model, the time varying results can be estimated such as displacement,

velocity, and acceleration of the multibody system without directly solving the governing equations of MBD, and then this estimation process is called *on-line* stage. In particular, *the feed forward networks* (FFN) with hidden layers and non-linear activation functions are employed among the various DNN methods since it can efficiently represent continuous functions. Three representative MBD problems, single pendulum, double pendulum, and slider crank mechanism, were considered to evaluate the performance of the proposed DNN based meta-modeling framework. To get the reliable meta-model, sufficient and accurate training data set of MBD is prerequisite, and random search is also important to define appropriate hyper-parameters of MBD problems such as the number of hidden layers, the size of batches, the number of epochs, optimizer, etc. In particular, numerical results imply that a position of time variable as input or output data is crucial to get the usable transient response of MBD.

In Section 2, the governing equations of MBD is briefly reviewed. In Section 3, the overview of neural networks of MBD and its meta-modeling process is presented. It should be noted that the framework of the proposed meta-modeling provides fundamental ideas of handling experimental or real-world data and exploiting their structures and relations to understand dynamics of general multibody systems. Not depending on complexity of MBD systems, the present meta-modeling helps us to achieve real-time and robust simulations with accurate motion results. In addition, high level of engineering simulations can be employed for not only engineering designs, but also motion related Internet of Things (IoT). Section 4 describes the case studies of the meta-modeling process using single pendulum, double pendulum, and slider crank mechanism. Conclusions are given in Section 5.

2 Brief Review on Common General Purpose MBD Governing Equations

Multibody system dynamics offers a straightforward approach to construct and solve equations of motion for mechanical systems. Multibody system dynamics includes a large number of procedures those can be categorized based on the used coordinates. In topological approaches, such like semi-recursive formulation, relative coordinates between the bodies are used. In the global approaches, in turn, the set of coordinates defines each body of the system. It is important to note that although topological and global approaches both lead to identical dynamic responses, the numerical performance differs. In this section often used global methods are briefly reviewed.

In *the augmented formulation*, constraint equations are accounted in the equation of motion by employing Lagrange multipliers. In this approach the equations of motion can be written as

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_{\mathbf{q}}^{T} \\ \mathbf{C}_{\mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{a} \\ \mathbf{F}_{c} \end{bmatrix} \tag{1}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ is the constraint vector, $\mathbf{C}_{\mathbf{q}}$ is the the Jacobian matrix of the constraint vector $\mathbf{C}$, $\mathbf{F}_a$ is the vector of applied generalized forces, and $\mathbf{F}_c$ the vector can be obtained by differential constraint twice with respect to time. The equation of motion is solved to obtain the generalized coordinates $\mathbf{q}$ and the Lagrange multipliers λ.

The other commonly used form of equations of motion for multibody system can be achieved from applying *the embedding technique* to global coordinates (1). The embedding technique reduces the generalized coordinates to be solved from $\ddot{\mathbf{q}}$ to a set of independent generalized $\ddot{\mathbf{q}}_{ind}$. In practice, this reduction can be accomplished using a transformation matrix $\mathbf{T}$:

$$\ddot{\mathbf{q}} = \mathbf{T}\ddot{\mathbf{q}}_{ind} + \mathbf{r}, \tag{2}$$

where $\mathbf{r}$ is a remainder vector. Substituting (2) into the augmented system (1) yields

$$\begin{cases} \mathbf{M}\mathbf{T}\ddot{\mathbf{q}}_{ind} + \mathbf{M}\mathbf{r} + \mathbf{C}_{\mathbf{q}}^{T}\boldsymbol{\lambda} = \mathbf{F}_{a}, \\ \mathbf{C}_{\mathbf{q}}\mathbf{T}\ddot{\mathbf{q}}_{ind} + \mathbf{C}_{\mathbf{q}}\mathbf{r} = \mathbf{F}_{c}. \end{cases} \tag{3}$$

By applying an identity $\mathbf{T}^{T}\mathbf{C}_{\mathbf{q}}^{T} = \mathbf{0}$, the equation (3) can be simplified into

$$\tilde{\mathbf{M}}\ddot{\mathbf{q}}_{ind} = \tilde{\mathbf{F}}, \tag{4}$$

where

$$\begin{aligned} \tilde{\mathbf{M}} &:= \mathbf{T}^{T}\mathbf{M}\mathbf{T}, \\ \tilde{\mathbf{F}} &:= \mathbf{T}^{T}\mathbf{F}_{a} - \mathbf{T}^{T}\mathbf{M}\mathbf{r}. \end{aligned} \tag{5}$$

3 Deep Neural Network for Multibody Dynamics Systems

In this section, a brief introduction to DNN that will be used in numerical examples is presented, and training of the DNN for MBD systems is also described.

Machine Learning (ML) aims to develop technologies and algorithms that enables computers to analyze and predict mechanisms of a system by learning structures of big amount of data. ML allows important tasks to be performed by generalizing from examples [12]. ML has already powered many aspects of modern society from web searches and item recognition to image classification, speech recognition [11], and cyber-physical systems (CPS)."

Being a part of ML, *Artificial Neural Networks (ANN)* are clusters of *nodes* (or *neurons*), which is designed to mimic the decision-making process of human brain, see Fig. 1. Nodes form *layers*, i.e. the input layer, the hidden layer, and the output layer. The input and output layers consist input and output parameters, respectively, of a meta-model. Containing information, nodes of each layer interchange the information through *weights*. One of the main purposes of ANN is to find the best weights to maximize the performance of a given neural network. Rumelhart et al. [2] developed an *error back-propagation algorithm* to find weights and improve neural networks efficiently.

To describe and represent more complicated and intricate data, more than one hidden layer can be considered. In this case, the ANN is referred to as *Deep Neural Networks (DNN)*. The increased number of hidden layers increases the number of nodes and weights, which requires an expensive computational cost and makes it difficult to train a model. Despite the shortcomings, DNN yields better meta-models

for solving complex *nonlinear* problems.

Structure of DNN can be specified in more details by the *hyper-parameters* such as the number of layers, the number of nodes for each layer, the batch size, the activation functions, the regulatory method, and the optimizers. The performance of DNN highly relies on the proper choice of hyper-parameters. Some important hyper-parameters mentioned in the numerical tests (Section 4) are briefly summarized as follows:

- *Batch size*
  The batch size is the number of training data samples in one pass for updating weights. Due to memory limitations, it is not recommended to perform training with all available data samples at once. The larger the batch size, the less computational cost a training requires.
- *Activation function*
  In DNN, values specified to nodes of a layer are not transferred directly to the next layer, but transformed through a nonlinear function, called *activation function*. It helps the values of nodes not to diverge during training and allows to solve complex problems with a small number of nodes. If an unsuitable activation function is chosen, gradients of DNN (in the error back-propagation process) can be vanishing, which makes learning speed severely slow. Activation functions such as *tanh*, *sigmoid*, and *ReLU*, are known to appropriate choices.
- *Optimizers*
  Weights of DNN are found by error back-propagation process, which sequentially updates the weights to minimize a *loss function* defined by a given error, such as $\mathscr{E}_{mse}$ described in (7). In this process, a local minimum problem needs to be solved and an efficient *optimizer* helps to reduce solution time. Representative techniques are stochastic gradient descent, Adam [1], RMSprop [3].

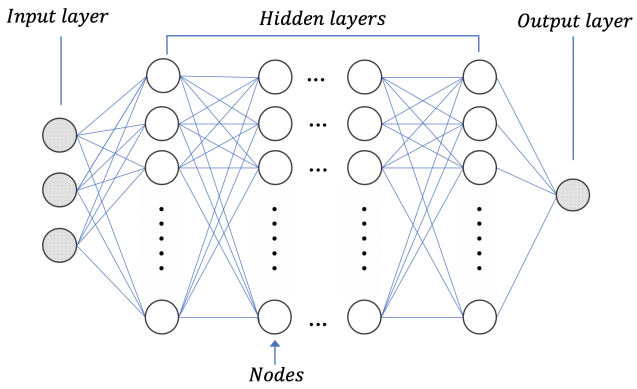

Fig. 1: Structure of Artifical Neural Networks (ANN). If there are multiple hidden layers, ANN is referred to as Deep Neural Networks (DNN).

3.1 Overview of Neural Networks for MBD

Meta-model using Neural Networks

ML methods can be categorized in viewpoint of learning styles into three: supervised, unsupervised, and reinforcement learning. Supervised learning trains a meta-model by considering both reference response features called *labels* and predictive features, and by gradually improving the model to fit the given training data. There are mainly classification and regression methods in supervised learning. In unsupervised learning, in contrary to supervised learning, label (or reference) features are not designated. It focuses on how training data is structured. Reinforcement learning is an effective algorithm for optimization analysis. It learns data by making decisions to maximize user-specified reward. Users need to design appropriate model conditions such as environments, actions, rewards.

MBD problems can be mainly dealt with supervised or reinforcement learning techniques since many MBD problems aim to seek robust and optimal design considering a set of design parameters.

To apply supervised learning, training data need to be prepared afore-hand for learning the model. The training data for MBD meta-models can be obtained in a few manners, usually by computational methods. In case of reinforcement learning, a multibody systems simulation environment is requisite to train an agent according to cumulated reward for each action. Both learning approaches require time-consuming tasks to learn the meta-models of MBD: data preparation task for supervised learning and simulation task for reinforcement learning. However, once the meta-model is built, it resolves MBD problems in real-time and yields dynamics responses.

In this research, the supervised learning of MBD meta-model based on training data is mainly considered. Supervised learning finds an approximation function $\mathscr{M}$ that minimizes a loss $L(x;\mathscr{M})$ over samples x. An algorithm $\mathscr{A}_\alpha$ produces $\mathscr{M}$ for a training set $\mathbf{X}^{train}$ through the optimization of a training criterion with respect to a set of parameters, given hyper-parameters α [10]. The built function

$$\mathscr{M} = \mathscr{A}(\mathbf{X}^{train};\alpha)$$

is called a *meta-model* in this research.

A neural network algorithm is one of the powerful machine learning algorithms of minimizing the loss

$$L\left(x;\mathscr{A}(\mathbf{X}^{train};\alpha)\right).$$

Specifically, the algorithm uses a network structure and optimizes the parameters of the networks, weights and biases, by utilizing the back-propagation algorithms, which is an extension of the gradient descent method for neural network structures.

In this research, neural networks are adopted to build the meta-models of MBD problems, since it is subject to be generalized to fit various shapes of nonlinear functions with multi-dimensional input data. In particular, *the feed forward networks (FFN)* with hidden layers and nonlinear activation functions are considered, which are the universal approximators that can represent effectively continuous functions. Owing to the characteristics of FFN, it is a powerful candidate of implementing the

meta-models of general purposed MBD problems. Moreover, many techniques for DNN including accelerated activation functions such as ReLU, dropout, regularization, and batch normalization have strengthened the potential of FFN with deep layers for modeling general purpose MBD problems.

The flowchart in Fig. 2 shows brief outlines of meta-modeling of MBD problems and its benefits.

Design of Neural Networks for Meta-models

MBD problems rarely have high dimensionality of input or output data, compared to common DNN applications such as image, speech, and text data. Rather than high dimensionality, in general, MBD considers complicated nonlinear functions and requires accurate and robust solutions.

If an MBD problem is given, the design of input and output layers is typically decided. For example, each variable of input (or output) data is mapped to a single node of the input (or output) layer in case that the variable is numeric one, but if the variable is nominal one, it should be mapped to multiple nodes through one-hot encoding. In one-hot encoding, each value of the nominal variable is transformed to one of one-hot vectors,

$$\{(1,0,\cdots,0),(0,1,\cdots,0),\cdots,(0,0,\cdots,1)\}.$$

Different from input and output layers, the design of hidden layers is volatile. The number of hidden layers and the number of nodes are the most critical hyper-parameters, and their best design must be decided along with other hyper-parameters at the step of hyper-parameter tuning. Empirically, it is known that deeper hidden layers are more effective than larger nodes of shallower hidden layers if two FFN models have similar numbers of parameters such as weights and biases.

To build expressive MBD meta-models, FFN models with enough width and depth are necessary. However, proper regularization methods such as L_1 and L_2 regularization, dropout and batch-normalization are required to achieve the generalized meta-models because FFN models with too many parameters are often overfitted to the given training data [9].

Hyper-parameters Optimization of meta-models

Similar to typical ML algorithms, the neural network algorithm does not provide a method to find the optimal hyper-parameters α. Hyper-parameters of DNN are critical to the accuracy and robustness of the meta-model. Unfortunately, there is no perfect scheme of building the most accurate and robust DNN model from a given training data. One must search the best set of hyper-parameters such as the number of hidden layers, the number of nodes in each hidden layers, activation function, optimization function, learning rate, and the number of epochs.

Generally, two kinds of search methods are often used for the purpose of hyper-parameter optimization; a given set of candidate values for each hyper-parameter are investigated with the *grid search* method, or randomly selected values for hyper-parameters are evaluated with the *random search* method. It is known that random search is more efficient to find optimal hyper-parameters than grid search [10]. Recently, AutoML is actively researched in academic and practical fields to find the

best design of DNN. When the AutoML techniques are mature, it is expected that the optimal design of the DNN-based meta-models can be found in easier and faster manners [8].

Generation of MBD Training Data
In this paper, it is assumed that one can obtain as many MBD sample data as is need to train the meta-model and achieve a reliable model. In other words, a case with an insufficient training data set is not considered. Nevertheless, since the process of MBD data collection takes so long time in case of complex multibody systems, a more efficient manners of collecting training data is needed.

First, the amount of training samples can be determined according to some criteria. Incremental learning methods can be applied to learn the meta-models. For instance, a certain level of performance measures such as the root mean squared errors or the mean absolute percentage errors can be adopted for the criteria to stop feeding more samples to the meta-model. In case of the random search method, simply more random samples can be provided to the less trained meta-model, and in case of the grid search method, finer-grained grid samples can be done [7].

Second, the range of each design parameter for more training samples can be adjusted after seeking less accurate ranges of design parameters of the meta-model. It is under an assumption that model complexity is often different in many ranges of nonlinear hyperplanes. In such cases, adaptive sampling methods such as focused grid search can be less exhaustive than uniform design of the typical grid search method [6].

3.2 Detailed Assumptions and Conditions for Meta-modeling Process

The followings are some assumptions and comments on the meta-modeling that is developed for MBD problems. The same conditions are applied to the numerical tests in Section 4.

1. *Training Data*

   • *Sufficiently many sets of training data*
   As mentioned in Section 3.1, it is assumed that there are as many sets of data for training and tests as one wants. Since the most important objective of this research is to achieve a highly accurate meta-model, the other issues such as computational efficiency and problems of insufficient training data are not mainly concerned.

   • *Uniform Meshes*
   Training data for input parameters are uniformly meshed in a given finite range.

   • *Data without Noise*
   Training data for output responses such as displacements, veloicities, or acclerations are *exactly* calculuated from governing equations for MBD problems. In other words, training data are artifically generated without any noise.

   • *Time Variable and Structurues of Training Data*

An important question in meta-modeling for dynamic problems is whether time variable t needs to be handled as an input parameter or not.
Table 1 shows an example of training data set, where time variable t is considered as an input. All the discrete time instants are contained in the set of training data. On the other hand, if time variable is not considered as an input parameter, there are $\#\{t_n\}$ sets of training data, where time is fixed to $t = t_n$, as shown in Table 2. The two types of training data structures are referred to as S_{full} and S_{fixed}.

It may seem that S_{fixed} is simpler than S_{full}, in that the former considers a fixed time instant $t = t_n$ and has a much smaller size of training data set compared to S_{full}, especially when the number of discrete time instants is very large. However, handling time variable as a non-input (S_{fixed}) is not adequate for MBD analysis in two following major aspects:

(a) It requires to make as many meta-models as the number of discrete time instants t_n, $n = 0, 1, \cdots$. Moreover, if grid search is performed for each meta-model to find out the best hyper-parameters, this approach can be computationally infeasible.
(b) Each resulting meta-model provides predictions only for a specific time $t = t_n$, which makes it difficult to figure out time-varying tendency of MBD.

Thus, in this research, it is concluded that a meta-model for MBD problems need to be generated from training data of form S_{full}, where time variable is considered as an input. More details on training data structure and its results are described in Section 4.

2. *Test Data*

• *Unseen Data*
The performance of a resulting meta-model is evaluated with some sets of test data which are unseen from training process.

• *Randomly Distributed Data*
Unlike training data, input parameters for test are not uniformly meshed. They are randomly distributed in the same given range.

3. *Grid Search and Hyper-parameters*

Grid search is performed to find out appropriate hyper-parameters for each MBD example, which helps to yield a highly accurate meta-model. From grid search, the number of hidden layers, the number of nodes for each layer, the size of batches, the number of epochs, optimizer, and loss functions need to be decided. Still, there can be other sets of hyper-parameters that result in similar or better performance.

4. *Evaluation of Performance*

The performance of a resulting meta-model $\mathscr{M}$ is evaluated in terms of two mea-

sures: R-squared value and absolute mean-squared error (MSE), denoted by $\mathscr{R}^2$ and $\mathscr{E}_{mse}$, respectively. When an output label y is given for a set of test data, and the mata-model $\mathscr{M}$ yields a prediction $\hat{y}$ for the test set, the performance measures are defined by

$$\mathscr{R}^2(y,\hat{y}) := 1 - \frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{N}(y_i-\bar{y})^2}, \tag{6}$$

$$\mathscr{E}_{mse}(y,\hat{y}) := \frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2, \tag{7}$$

where $y=(y_1,\cdots,y_N)$, $\hat{y}=(\hat{y}_1,\cdots,\hat{y}_N)$, and $\bar{y} := \sum_{i=1}^{N} y_i/N$. As the solution $\hat{y}$ of the meta-model predicts the label y more accurately, the value of $\mathscr{R}^2$ closes to 1, and the error $\mathscr{E}_{mse}$ closes to 0.

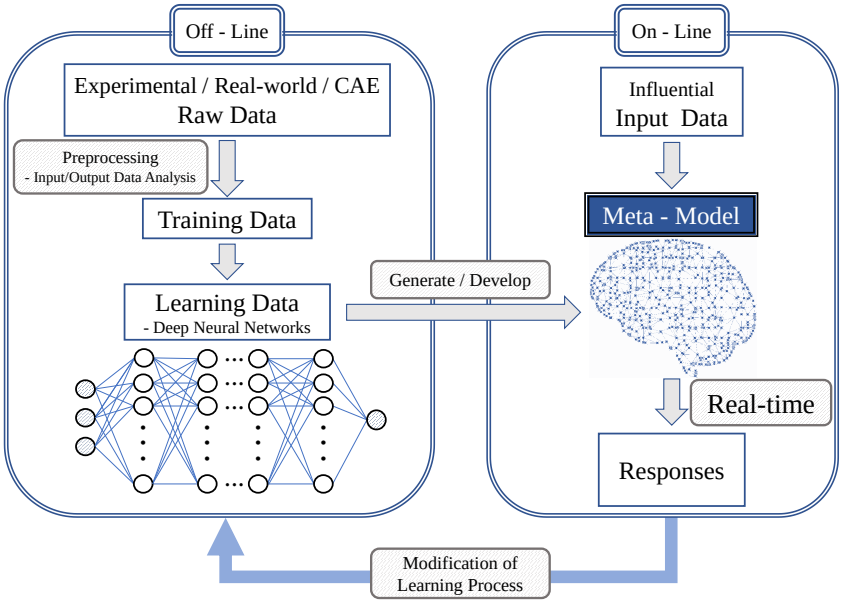

Fig. 2: Flows of meta-modeling for MBD. By analyzing and learning data on MBD, a meta-model can be generated. The meta-model is intended to yield *real-time* dynamic responses of given MBD problems. Performance of the meta-model can be evaluated by comparing its results with experimental or real-world data. The evaluation helps to reconstruct or improve the off-line learning algorithm.

4 Case Studies

In this section, three fundamental MBD examples, single pendulum, double pendulums, and slider crank mechanisms, are investigated. For each example, a data-driven

Input			Output	
L	c	t	θ	$\dot{\theta}$
0.0	0.00	0.00	1.57080	0.00000
0.0	0.00	0.01	1.56589	-0.98100
0.0	0.00	0.02	1.55118	-1.96192
0.10	0.05	0.00	1.57080	0.00000
0.10	0.05	0.01	1.56590	-0.97936
0.10	0.05	0.02	1.55122	-1.95540
0.20	0.10	0.00	1.57080	0.00000
0.20	0.10	0.01	1.56590	-0.97773
0.20	0.10	0.02	1.55126	-1.94890

Table 1: Structure of training data set for DNN, where time variable t is considered as an *input*. This type of training data structure is denoted by S_{full}. In this case, a single meta-model is generated.

	Input		Output	
	L	c	θ	$\dot{\theta}$
$t = 0.00$	0.0	0.00	1.57080	0.00000
	0.10	0.05	1.57080	0.00000
	0.20	0.10	1.57080	0.00000

	Input		Output	
	L	c	θ	$\dot{\theta}$
$t = 0.01$	0.0	0.00	1.56589	-0.98100
	0.10	0.05	1.57080	0.00000
	0.20	0.10	1.56590	-0.97773

	Input		Output	
	L	c	θ	$\dot{\theta}$
$t = 0.02$	0.0	0.00	1.55118	-1.96192
	0.10	0.05	1.55122	-1.95540
	0.20	0.10	1.55126	-1.94890

Table 2: Structure of training data set for DNN, where time variable t is fixed and not considered as an input. This type of training data structure is denoted by S_{fixed}. In this case, $\#\{t_n\}$ numbers of meta-models are generated corresponding to $\#\{t_n\}$ sets of training data.

meta-model is generated through FFN, and its performance is evaluated in various ways, as described in Section 3.2.

4.1 Damped Single Pendulum

A damped single pendulum problem shown in Fig. 3 can be expressed in the following mathematical governing equation:

$$\begin{cases} \ddot{\theta} + \dfrac{g}{L}\sin(\theta) + \dfrac{c}{mL}\dot{\theta} = 0, & \text{where } \theta = \theta(t), \quad t \in [0, t_f], \\ \theta(t) = \theta^0, \dot{\theta}(t) = \dot{\theta}^0, & \text{where } t = 0, \end{cases} \tag{8}$$

where g is the gravity acceleration, L is the length of the massless rod, m is the mass, and c is the damping coefficient, respectively. The variables θ and $\dot{\theta}$ are time-varying angle and its velocity, whose initial values are specified as θ^0 and $\dot{\theta}^0$, respectively.

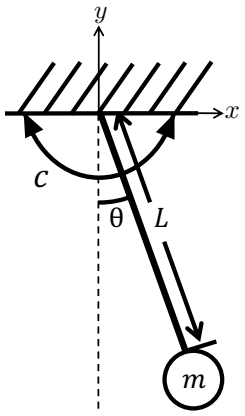

Fig. 3: Dampled single pendulum problem. Gravity acceleration g, the mass m, and the initial angle θ^0 are fixed to $g = 9.81[m/s^2], m = 0.3[kg]$, and $\theta^0 = \pi/2[rad]$. The length of the massless rod $L[m] \in [0.1, 0.2]$, the damping coefficient $c[kg \cdot m/s] \in [0, 1]$, and the initial angular velocity $\dot{\theta}^0[rad/s] \in [0, 5]$ are arbitrarily determined within the given ranges.

Although all the input parameters $(g, L, m, c, \theta^0, \dot{\theta}^0)$ affect dynamics of the single pendulum in Fig. 3, it is empirically noticed that the parameters $(L, c, \dot{\theta}^0)$ make a major influence on the dynamic response characteristics. Thus, it is assumed that the relatively insignificant parameters (g, m, θ^0) are fixed to values ($9.81[m/s^2]$, $0.3[kg]$, $\pi/2[rad]$), while the parameters $(L, c, \dot{\theta}^0)$ are not determined specifically. It is the objective of this example to generate a meta-model which yields the dynamics of damped single pendulum as outputs when a particular set of input parameters $(L, c, \dot{\theta}^0)$ are given.

For an efficient learning, it is assumed that $(L, c, \dot{\theta}^0)$ are chosen within finite ranges:

$$\begin{aligned} L[m] &\in [0.1, 0.2] \ \ (\Delta L = 0.01), \\ c[kg \cdot m/s] &\in [0, 0.15] \ \ (\Delta c = 0.01), \\ \dot{\theta}^0[rad/s] &\in [0, 5] \ \ (\Delta \dot{\theta}^0 = 0.5). \end{aligned}$$

Here, $(\Delta L, \Delta c, \Delta \dot{\theta}^0)$ denote uniform meshsizes for training data. In evaluating a meta-model, the uniform meshes are not applied, and arbitrarily chosen input values are used.

To describe dynamics of the damped single pendulum, the time-varying solutions $\theta(t)$, $\dot{\theta}(t)$, and $\ddot{\theta}(t)$ are achieved as outputs of a meta-model. For time variable t, discrete time instants $\{t_n\}$ with a uniform meshsize Δt is considered in an interval $[0, t_f]$, where $t_f = 2$:

$$t_n[s] := n\Delta t \in [0, 2] \quad \left(\Delta t = 10^{-2}\right), \tag{9}$$

for $n = 0, 1, \cdots, 200$.

As described in Section 3.2, time variable t can be handled as an input (S_{full}) or fixed to a certain instant (S_{fixed}). Results from the two structures are compared. S_{full} case generates only one meta-model, while S_{fixed} case $\#\{t_n\} = 201$ meta-models. Thus, for S_{full}, the input and output of meta-model are four and three dimenional, repectively. The total number of training data is $267,531$. S_{fixed}has three dimensional input and the number of its training data is $1,331$ for each model.

Hyper-parameters found from grid search are shown in Table 3.

Hyper-parameters	Choice
The number of hidden layers	2
The number of nodes in each layer	128
The size of batch	64
The number of epochs	400
Loss function	$\mathscr{E}_{mse}$
Optimizer	Adam

Table 3: Hyper-parameters for the damped single pendulum problem

Fig. 4 displays the scatter plots where *labels*, i.e. reference solutions, and predictions of outputs $(\theta, \dot{\theta}, \ddot{\theta})$ are compared. The results are achieved from a set of test data, which are unseen from training. The $\mathscr{R}^2$ scores are around 0.997, which implies that the DNN model predicts the outputs with high accuracy.

Fig. 5 shows dynamics of angle(θ) (Top), angular velocity($\dot{\theta}$) (Middle), and angular acceleration($\ddot{\theta}$) (Bottom), for a specific case: $L = 0.1911[m]$, $c = 3.78[kg \cdot m/s]$, $\dot{\theta}^0 = 0.055[rad/s]$. Labels (blue dashed, crosses) and predictions (red solid, circles) are shown for each solution. Results of S_{fixed}(Left) and S_{full}(Right) are compared. Although both S_{fixed} and S_{full} yields highly accurate results, some oscillations

	$L\,[m]$	$c\,[kg\cdot m/s]$	$\dot{\theta}^0\,[rad/s]$
Case 1	0.123	2.53	0.055
Case 2	0.1583	0.52	0.055
Case 3	0.1758	0.52	0.109
Case 4	0.1911	4.52	0.109

Table 4: Input parameters of multiple cases for Fig. 6

are observed in case of S_{fixed}(Left). On the other hand, S_{full}(Right) gives relatively smooth solutions.

In Fig. 6, performance comparison of S_{fixed}(Left) and S_{full}(Right) for other input parameters are summarized in Table 4: Similarly as in Fig. 5, oscillatory waves are observed in case of S_{fixed}. Some are more severe than others, which makes prediction error greater. On the other hand, S_{full} yields smooth and accurate predictions for all cases.

4.1.1 Hyper-parameters for S_{full} and S_{fixed}

In the damped single pendulum problem, the same hyper-parameters are used to both types of training data S_{full} and S_{fixed}, where the hyper-parameters are found from a grid search for S_{full}. Since the data structures of S_{full} and S_{fixed} are different, it would be the best to carry out independent grid search for each structure, in comparing results of S_{full} and S_{fixed}. Obviously, the performance of S_{fixed} will be improved if more appropriate hyper-parameters are applied. To clarify positives and negatives of employing better hyper-parameters for S_{fixed}, independent grid searches for S_{fixed} models are performed. Since there are $\#\{t_n\} = 201$ models in S_{fixed}, $\#\{t_n\}$ grid searches are required. The hyper-parameters found for S_{fixed} are listed in Table 5.

Obviously, compared to the hyper-parameters for S_{full} in Table 3, those in Table 5 improves the performance of S_{fixed}. The improved results corresponding to Fig. 5 (Left) and 6 (Left) are shown in Fig. 7 (Left) and Fig. 7 (Right), respectively. Compared to the results shown in Fig. 5 (Left) and 6 (Left), the accuracies of solutions from independent grid searches are clearly enhanced, which can be confirmed by the orders of $\mathscr{E}_{mse}$.

However, the oscillations are still observed, which yield less smooth solutions compared to the results of S_{full}, shown in Fig. 5 (Right) and 6 (Right). In addition, $\#\{t_n\}$ numbers of grid searches for S_{fixed} requires a heavy computational burden. The normalized clock time for grid search for S_{full} and S_{fixed} are compared in Table 6.

Thus, the usage of the same hyper-parameters to both S_{full} and S_{fixed} is not a serious hindrance to comparing performance of the two types of training data sets. For simplicity and computational feasibility, the hyper-parameters found from S_{full} for both S_{full} and S_{fixed} are employed, in the numerical examples in Sections 4.2 and 4.3.

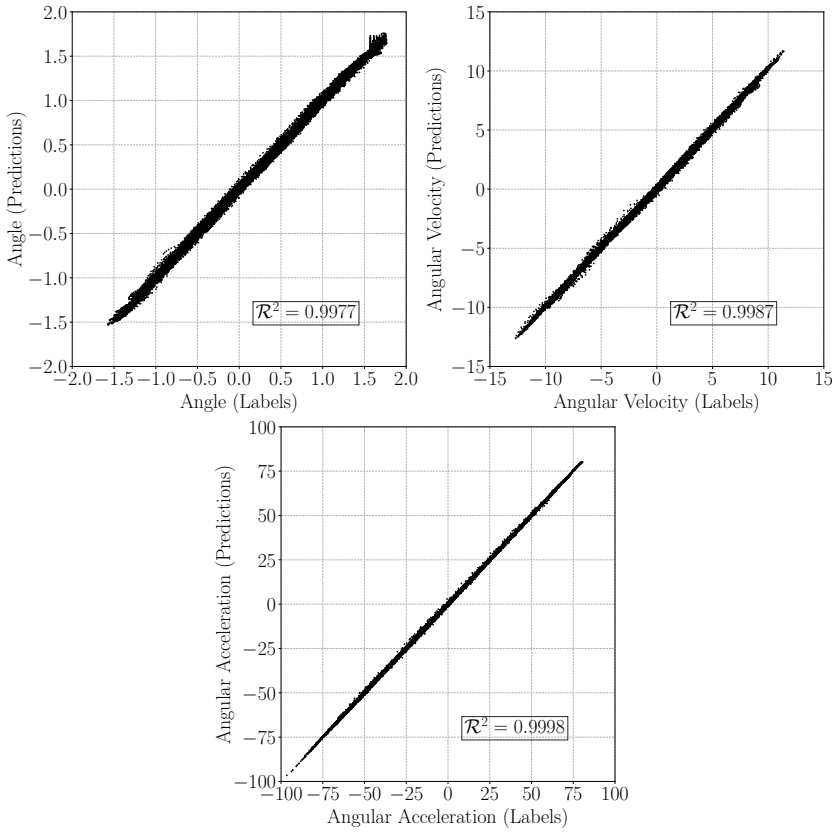

Fig. 4: Labels vs. Predictions for test data. The meta-model for the damped single pendulum problem is generated from S_{full} type of training set. Test data are *unseen* from training. The $\mathscr{R}^2$ values are almost 1, which implies that the meta-model predicts output solutions with high accuracy.

4.2 Double Pendulum

A double pendulum problem in Fig. 8 follows the given mathematical governing equation:

$$\begin{cases} (m_1+m_2)L_1\ddot{\theta}_1 + m_2L_2\ddot{\theta}_2\cos(\theta_1-\theta_2) + m_2L_2\dot{\theta}_2^2\sin(\theta_1-\theta_2) + (m_1+m_2)g\sin(\theta_1) = 0, \\ m_2L_2\ddot{\theta}_2 + m_2L_1\ddot{\theta}_1\cos(\theta_1-\theta_2) - m_2L_1\dot{\theta}_1^2\sin(\theta_1-\theta_2) + m_2g\sin(\theta_2) = 0, \\ \theta_i(t) = \theta_i^0,\ \dot{\theta}_i(t) = \dot{\theta}_i^0, \quad \text{where} \quad t = 0, \quad i = 1,2. \end{cases} \tag{10}$$

where $\theta_i = \theta_i(t)$ and $t \in [0, t_f]$, $i = 1,2$, represent the time-varying angles of the links as shown in Fig. 8. Parameters g is the gravity constant, L_i is the length of the massless rod i, m_i is the mass, θ_i^0 is the initial angle, $\dot{\theta}_i^0$ is the the initial angular velocity, and $i = 1,2$, body notation, respectively.

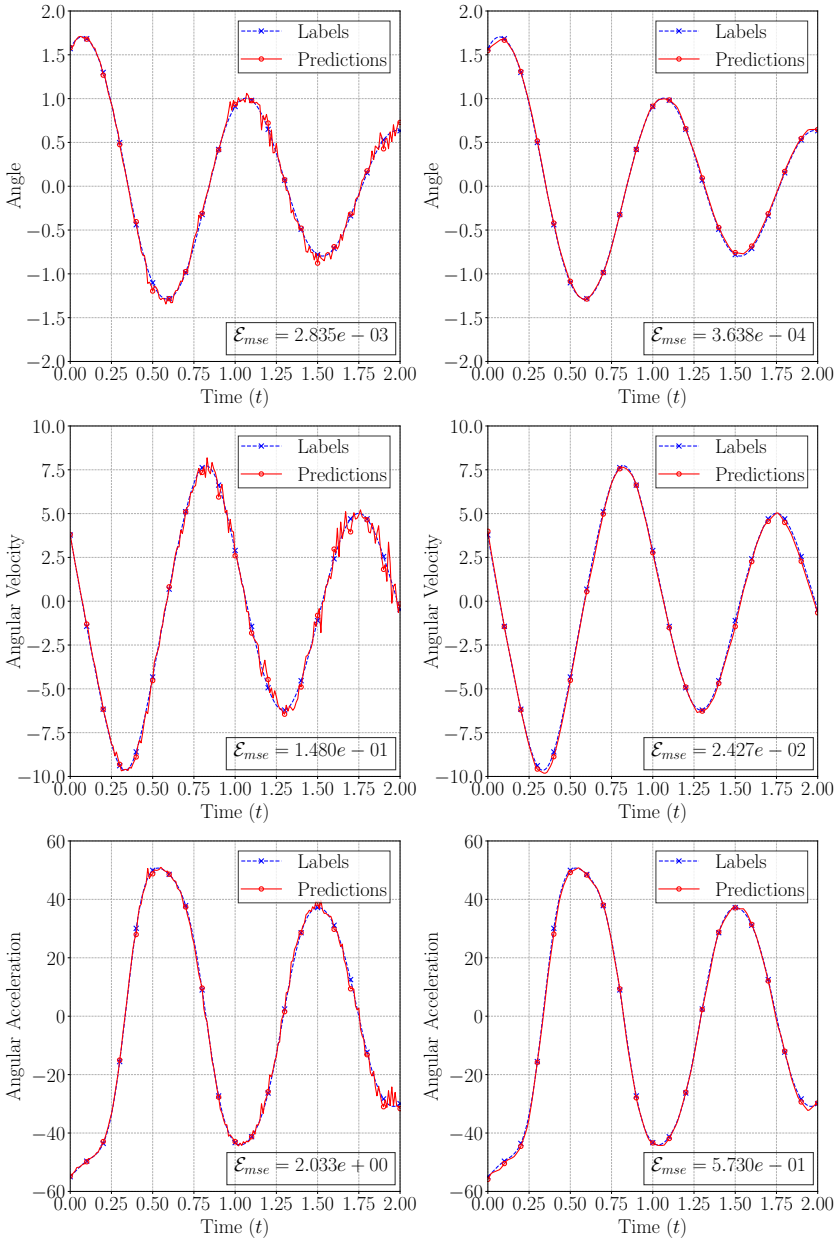

Fig. 5: Dynamic responses of the damped single pendulum for specific input $L = 0.1911[m]$, $c = 3.78[kg \cdot m/s]$, $\dot{\theta}^0 = 0.055[rad/s]$. Labels(blue dashed, crosses) and predictions(red solid, circles) are compared for test data. Left: $\#\{t_n\}$ numbers of meta-models are generated for each fixed time $t = t_n$ (S_{fixed}). Some oscillations are observed. Right:When time variable t is considered as an input parameter (S_{full}). Relatively smooth solutions are achieved.

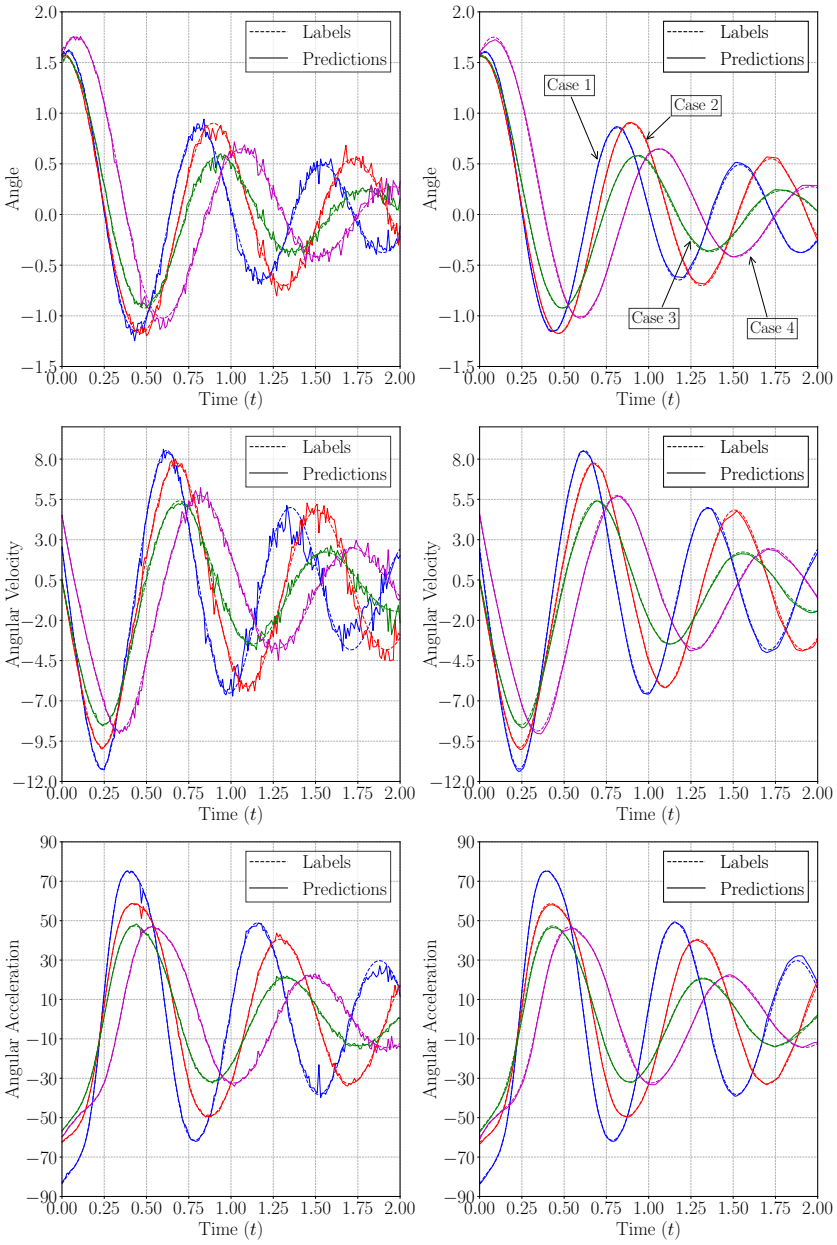

Fig. 6: Dynamic responses of single pendulum for multiple inputs $L = 0.123[m]$, $c = 2.53[kg \cdot m/s]$, $\dot{\theta}^0 = 0.055[rad/s]$ (blue), $L = 0.1583[m]$, $c = 0.52[kg \cdot m/s]$, $\dot{\theta}^0 = 0.055[rad/s]$ (red), $L = 0.1758[m]$, $c = 0.52[kg \cdot m/s]$, $\dot{\theta}^0 = 0.109[rad/s]$ (green), $L = 0.1911[m]$, $c = 4.52[kg \cdot m/s]$, $\dot{\theta}^0 = 0.109[rad/s]$ (magenta). Labels(dashed) and predictions(solid) are compared for test data. Left:#$\{t_n\}$ numbers of meta-models are generated for each fixed time $t = t_n$.(S_{fixed}). Some oscillations are observed. Right:When time variable t is considered as an input parameter (S_{full}). Relatively smooth solutions are achieved.

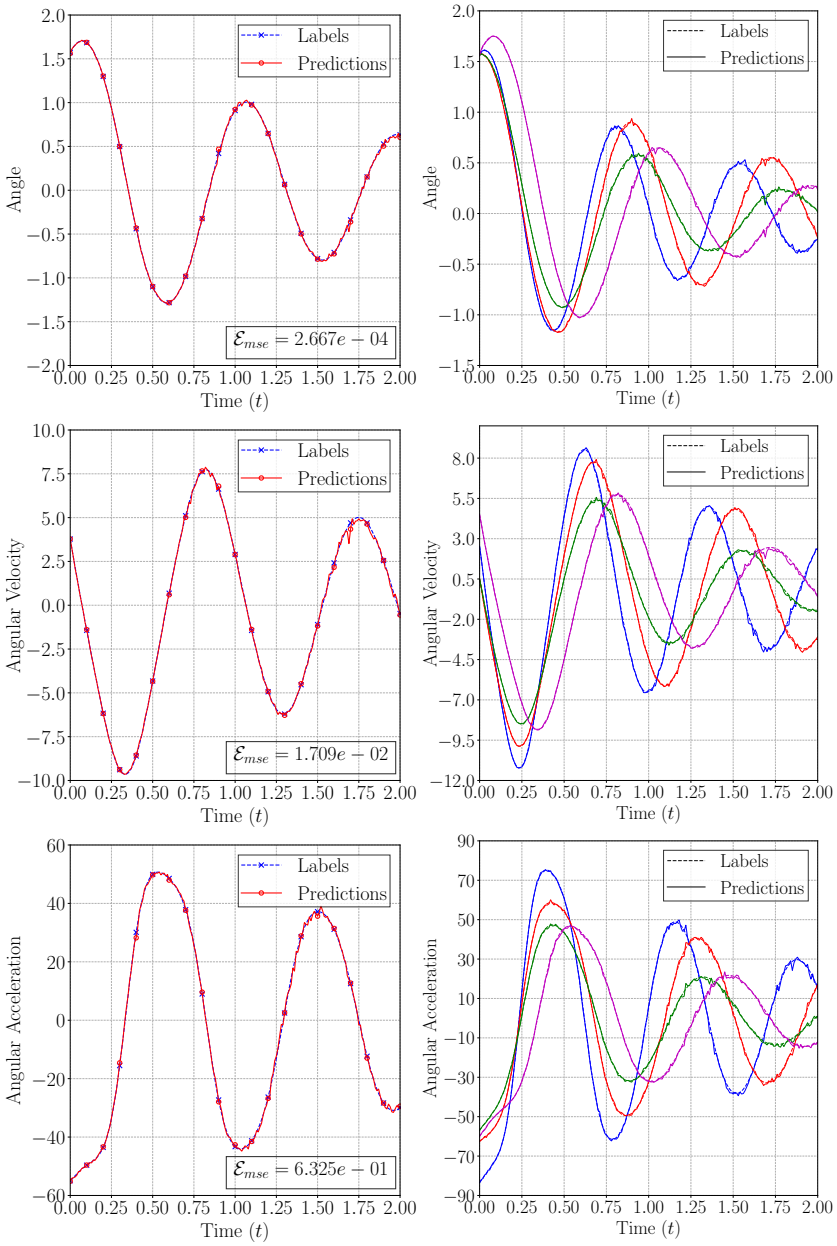

Fig. 7: Dynamic responses of the damped single pendulum achieved from S_{fixed} training data with hyper-parameters in Table 5. While the results in Fig. 5 (Left) and 6 (Left) employs the hyper-parameters of S_{full}, the present results uses the hyper-parameters from independent grid searches on $\#\{t_n\} = 201$ numbers of S_{fixed} models. While the accuracies of solutions are improved, the oscillations are still observed.

Model for $t = t_n$	The number of hidden layers	The number of nodes per a hidden layer	The size of batch
$t = 0.00$	2	256	128
$t = 0.01$	2	256	64
$t = 0.02$	2	256	128
$t = 0.03$	2	256	64
$t = 0.04$	2	256	128
$t = 0.05$	2	256	128
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$t = 1.00$	3	256	128
$t = 1.01$	4	256	128
$t = 1.02$	4	256	64
$t = 1.03$	3	256	128
$t = 1.04$	4	256	128
$t = 1.05$	2	256	128
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$t = 1.95$	4	128	128
$t = 1.96$	4	128	128
$t = 1.97$	3	256	64
$t = 1.98$	3	256	128
$t = 1.99$	3	256	128
$t = 2.00$	4	256	128

Table 5: Hyper-parameters for S_{fixed} training data, which are achieved from independent grid searches for $\#\{t_n\} = 201$ S_{fixed}models.

Training data	The number of models	The number of training data per model	Normalized clock time for grid searches
S_{full}	1	267,531	1
S_{fixed}	201	1,331	18.3458

Table 6: Comparison of data structures S_{full} and S_{fixed}, and normalized clock times taken for independent grid searches. Grid searches for S_{fixed}requires a heavy computational cost.

In the meta-modeling, it is assumed that $(L_1, L_2, \dot{\theta}_1^0, \dot{\theta}_2^0)$ are independent input parameters and $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$ are output parameters. As in the single pendulum problem (8), inputs are chosen within some ranges. The other parameters are fixed to given constants. More details on ranges and mesh sizes of parameters are summarized in Table 9.

As in the previous numerical example, two types of training data, i.e. S_{fixed} and S_{full} are compared. For S_{fixed}, there are $\#\{t_n\} = 501$ meta-models, where each model

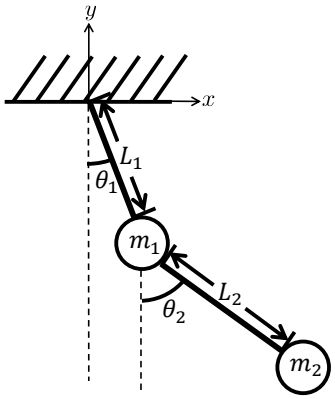

Fig. 8: Double pendulum problem. Gravity acceleration g, the masses m_1,m_2, and the initial angles θ_1^0,θ_2^0,are fixed to $g = 9.81[m/s^2]$, $m_1 = 2.0[kg]$, $m_2 = 1.0[kg]$, $\theta_1^0 = 1.6[rad]$, and $\theta_2^0 = 1.6[rad]$. The lengths of the massless rods $L_1[m] \in [1,2]$, $L_2[m] \in [2,3]$, and the initial angular velocities $\dot{\theta}_1^0[rad/s] \in [0,0.1]$, $\dot{\theta}_2^0[rad/s] \in [0.3,0.5]$ are arbitrarily determined within the given ranges.

is trained from $14,641$ numbers of data set. For S_{full}, there is only one meta-model trained from $14,641 \times 501 = 7,335,141$ numbers of data set. For both S_{fixed} and S_{full} types of training data, hyper-parameters are found as in Table 7.

Hyper-parameters	Choice
The number of hidden layers	4
The number of nodes in each layer	64
The size of batch	1024
The number of epochs	400
Loss function	$\mathscr{E}_{mse}$
Optimizer	Adam

Table 7: Hyper-parameters for the double pendulum problem

The scatter plots in Fig. 9 show that a meta-model from S_{full} predicts output parameters $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$ with a great accuracy. The $\mathscr{R}^2$ values are over 0.997 in all cases of solutions.

Performances of meta-models from S_{fixed} and S_{full} types of training data are compared in Fig. 10 and 11. It shows dynamic changes of predictions (solid) from meta-models in comparison with their labels (dashed), for multiple cases as shown in Table 8.

As observed in single pendulum cases shown in Fig. 5 and 6, the meta-model from S_{fixed} shows lots of oscillations in its dynamic responses. Here the oscillations are quite severe, especially when t is large. Though these results can be improved if

	L_1 $[m]$	L_2 $[m]$	$\dot{\theta}_1^0$ $[rad/s]$	$\dot{\theta}_2^0$ $[rad/s]$
Case 1	1.010	2.130	0.00	0.300
Case 2	1.500	2.410	0.03	0.330
Case 3	1.620	2.560	0.044	0.384
Case 4	1.330	2.820	0.062	0.412
Case 5	1.980	2.940	0.087	0.470

Table 8: Input parameters of multiple cases for Fig. 10 and 11.

	Parameters	Ranges	Meshsizes for Training Data	Meshsizes for Test Data
Fixed constants	g $[m/s^2]$	9.81	·	·
	m_1 $[kg]$	2.0	·	·
	m_2 $[kg]$	1.0	·	·
	θ_1^0 $[rad]$	1.6	·	·
	θ_2^0 $[rad]$	1.6	·	·
Inputs	L_1 $[m]$	[1, 2]	$\Delta L_1 = 0.1$	arbitrary(not uniform)
	L_2 $[m]$	[2, 3]	$\Delta L_2 = 0.1$	arbitrary(not uniform)
	$\dot{\theta}_1^0$ $[rad/s]$	[0, 0.1]	$\Delta \dot{\theta}_1^0 = 0.01$	arbitrary(not uniform)
	$\dot{\theta}_2^0$ $[rad/s]$	[0.3, 0.5]	$\Delta \dot{\theta}_2^0 = 0.02$	arbitrary(not uniform)
Time instants	$\{t_n\}$ $[s]$	[0,5]	$\Delta t = 0.01$ $(t_0 = 0)$	$\Delta t = 0.01$ $(t_0 = 0)$

Table 9: Summary on parameters of double pendulum problem. In S_{fixed}, a fixed time instant is considered. In S_{full}, all the time instants are treated as inputs.

more appropriate hyper-parameters are employed for each of $\#\{t_n\}$ number of meta-models, the grid searches are computationally infeasible. On the other hand, meta-model from S_{full} yields more accurate and smooth dynamic responses.

Difference between two training data set S_{fixed} and S_{full} is shown more clearly in Fig. 12, where trajectories of two masses m_1 and m_2 are shown. Labels (m_1: black solid, m_2: black dashed) and predictions (m_1:blue solid,circles, m_2:red solid,circles) are given for the results from S_{fixed}(Left) and S_{full}(Right). Each plot is from a particular input parameters: $L_1 = 1.500[m]$, $L_2 = 2.410[m]$, $\dot{\theta}_1^0 = 0.03[rad/s]$, $\dot{\theta}_2^0 = 0.330[rad/s]$ (Top), $L_1 = 1.980[m]$, $L_2 = 2.940[m]$, $\dot{\theta}_1^0 = 0.087[rad/s]$, $\dot{\theta}_2^0 = 0.470[rad/s]$ (Middle), $L_1 = 1.400[m]$, $L_2 = 2.500[m]$, $\dot{\theta}_1^0 = 0.060[rad/s]$, $\dot{\theta}_2^0 = 0.380[rad/s]$ (Bottom).

4.3 Slider Crank Mechanism

Consider a slider crank in Fig. 13, where parameters $(r, L, \theta(t), \phi(t))$ represent, respectively, the length of the massless crank shaft$[m]$, the length of the massless connecting rod$[m]$, the angle of the crank shaft$[rad]$, and the angle of the connecting rod$[rad]$. The initial angle $\theta^0[rad]$ and the initial velocity $\dot{\theta}^0[rad/s]$ are assumed as

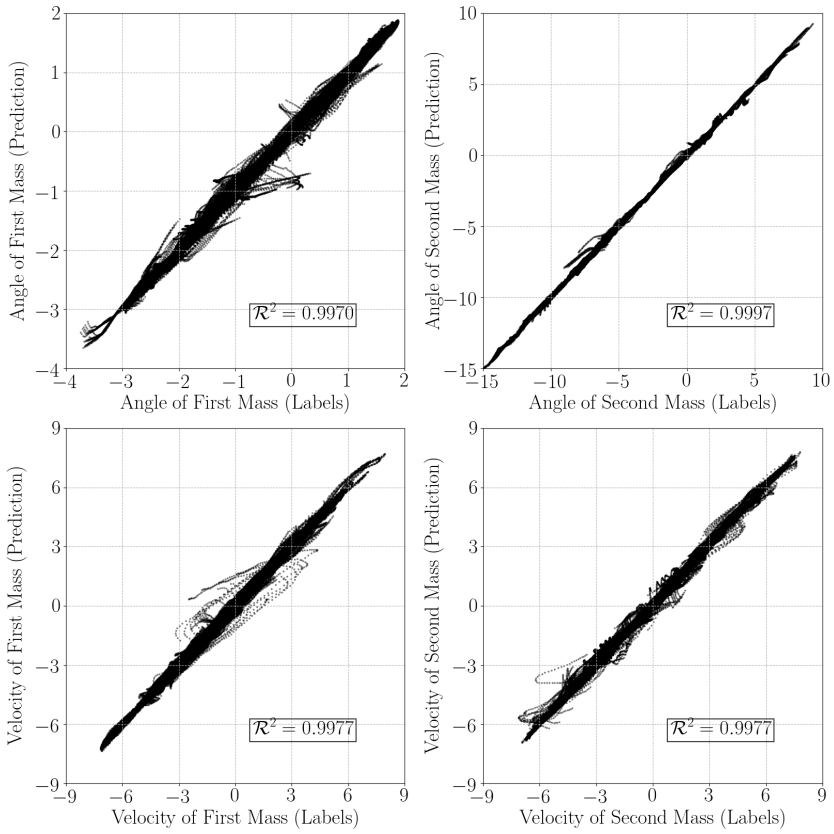

Fig. 9: Labels vs. Predictions for test data. The meta-model for the double pendulum problem is generated from S_{full} type of training set. Test data are *unseen* from training. The $\mathscr{R}^2$ scores are almost 1, which implies that the meta-model yields accurate solutions.

zeros, and the angular acceleration of the crank shaft $\ddot{\theta}(t)[rad/s^2]$ is given as

$$\begin{aligned} \theta(t) &= \theta^0 = 0, \quad \text{where } t = 0, \\ \dot{\theta}(t) &= \dot{\theta^0} = 0, \quad \text{where } t = 0, \\ \ddot{\theta}(t) &= \sin(\tau t), \quad \text{where } t \in [0, t_f], \end{aligned} \tag{11}$$

for some constant $\tau \in \mathbb{R}$.
Then the angle of the crank shaft $\theta(t)$ and its temporal derivatives can be rewritten explicitly, for $t \in [0, t_f]$,

$$\begin{aligned} \theta(t) &= -\frac{1}{\tau^2}\sin(\tau t) + \frac{t}{\tau} + \theta^0 = -\frac{1}{\tau^2}\sin(\tau t) + \frac{t}{\tau}, \\ \dot{\theta}(t) &= -\frac{1}{\tau}\cos(\tau t) + \frac{1}{\tau} + \dot{\theta}^0 = -\frac{1}{\tau}\cos(\tau t) + \frac{1}{\tau}, \end{aligned} \tag{12}$$

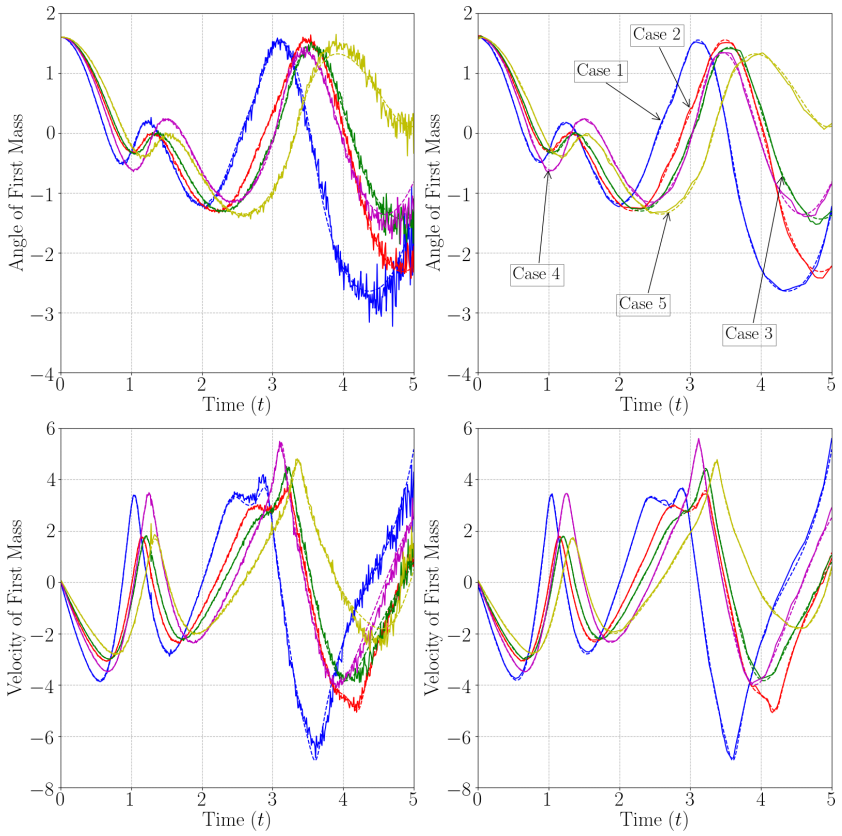

Fig. 10: Dynamic responses of double pendulum for multiple cases of input parameters: $L_1 = 1.010[m]$, $L_2 = 2.130[m]$, $\dot{\theta}_1^0 = 0.0[rad/s]$, $\dot{\theta}_2^0 = 0.3[rad/s]$ (blue), $L_1 = 1.500[m]$, $L_2 = 2.410[m]$, $\dot{\theta}_1^0 = 0.03[rad/s]$, $\dot{\theta}_2^0 = 0.330[rad/s]$ (red), $L_1 = 1.620[m]$, $L_2 = 2.560[m]$, $\dot{\theta}_1^0 = 0.044[rad/s]$, $\dot{\theta}_2^0 = 0.384[rad/s]$ (green), $L_1 = 1.330[m]$, $L_2 = 2.820[m]$, $\dot{\theta}_1^0 = 0.062[rad/s]$, $\dot{\theta}_2^0 = 0.412[rad/s]$ (magenta), $L_1 = 1.980[m]$, $L_2 = 2.940[m]$, $\dot{\theta}_1^0 = 0.087[rad/s]$, $\dot{\theta}_2^0 = 0.470[rad/s]$ (yellow). Labels(dashed) and predictions(solid) are given for test data. Results from two types of training set S_{fixed}(Left) and S_{full}(Right) are compared. Oscillations from S_{fixed} becomes more severe than the case of single pendulum shown in Fig. 6.

In DNN modeling, three independent parameters $(\tau, r, L/r)$ are considered as inputs, while time variable t can be fixed to an instant (S_{fixed}) or considered as an input (S_{full}). More details on ranges and mesh sizes of parameters are summarized in Table 11.

Although the slider crank mechanism is not a dynamic problem, this kinematic example is a good example because the kinematics should be treated as a special case of dynamic problems. To describe kinematics of the slider crank, seven kinematic

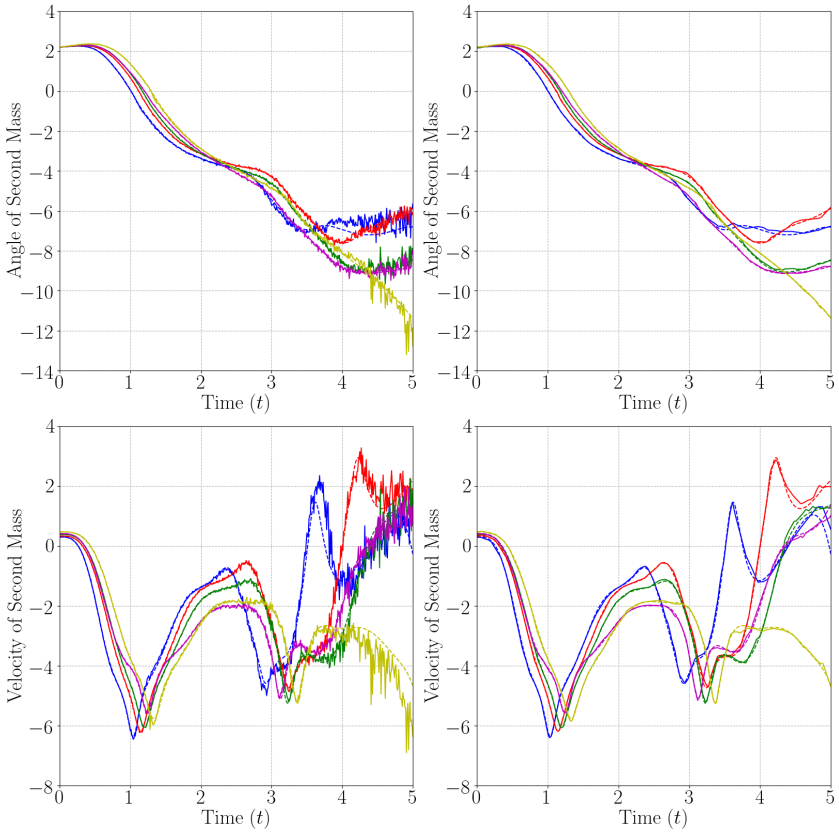

Fig. 11: Dynamic responses of double pendulum for multiple cases of input parameters: $L_1 = 1.010[m]$, $L_2 = 2.130[m]$, $\dot{\theta}_1^0 = 0.0[rad/s]$, $\dot{\theta}_2^0 = 0.3[rad/s]$ (blue), $L_1 = 1.500[m]$, $L_2 = 2.410[m]$, $\dot{\theta}_1^0 = 0.03[rad/s]$, $\dot{\theta}_2^0 = 0.330[rad/s]$ (red), $L_1 = 1.620[m]$, $L_2 = 2.560[m]$, $\dot{\theta}_1^0 = 0.044[rad/s]$, $\dot{\theta}_2^0 = 0.384[rad/s]$ (green), $L_1 = 1.330[m]$, $L_2 = 2.820[m]$, $\dot{\theta}_1^0 = 0.062[rad/s]$, $\dot{\theta}_2^0 = 0.412[rad/s]$ (magenta), $L_1 = 1.980[m]$, $L_2 = 2.940[m]$, $\dot{\theta}_1^0 = 0.087[rad/s]$, $\dot{\theta}_2^0 = 0.470[rad/s]$ (yellow). Labels(dashed) and predictions(solid) are given for test data. Results from two types of training set S_{fixed}(Left) and S_{full}(Right) are compared. Oscillations from S_{fixed} becomes more severe than the case of single pendulum shown in Fig. 6

solutions θ, ϕ, $\dot{\phi}$, $\ddot{\phi}$, x_B, $\dot{x}_B$, and $\ddot{x}_B$ are considered as an output parameters, where x_B denotes the x-directional translation of the slider.

The output solutions other than $(\theta, \dot{\theta}, \ddot{\theta})$ can be found from kinematic equations as follows:

$$\begin{bmatrix} \phi \\ x_B \end{bmatrix} = \begin{bmatrix} \sin^{-1}\left(-(r/L)\sin\theta\right) \\ r\cos\theta + L\cos\phi \end{bmatrix}, \tag{13}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{x}_B \end{bmatrix} = \begin{bmatrix} L\sin\phi & 1 \\ -L\cos\phi & 0 \end{bmatrix}^{-1} \begin{bmatrix} -r\dot{\theta}\sin\theta \\ r\dot{\theta}\cos\theta \end{bmatrix}, \tag{14}$$

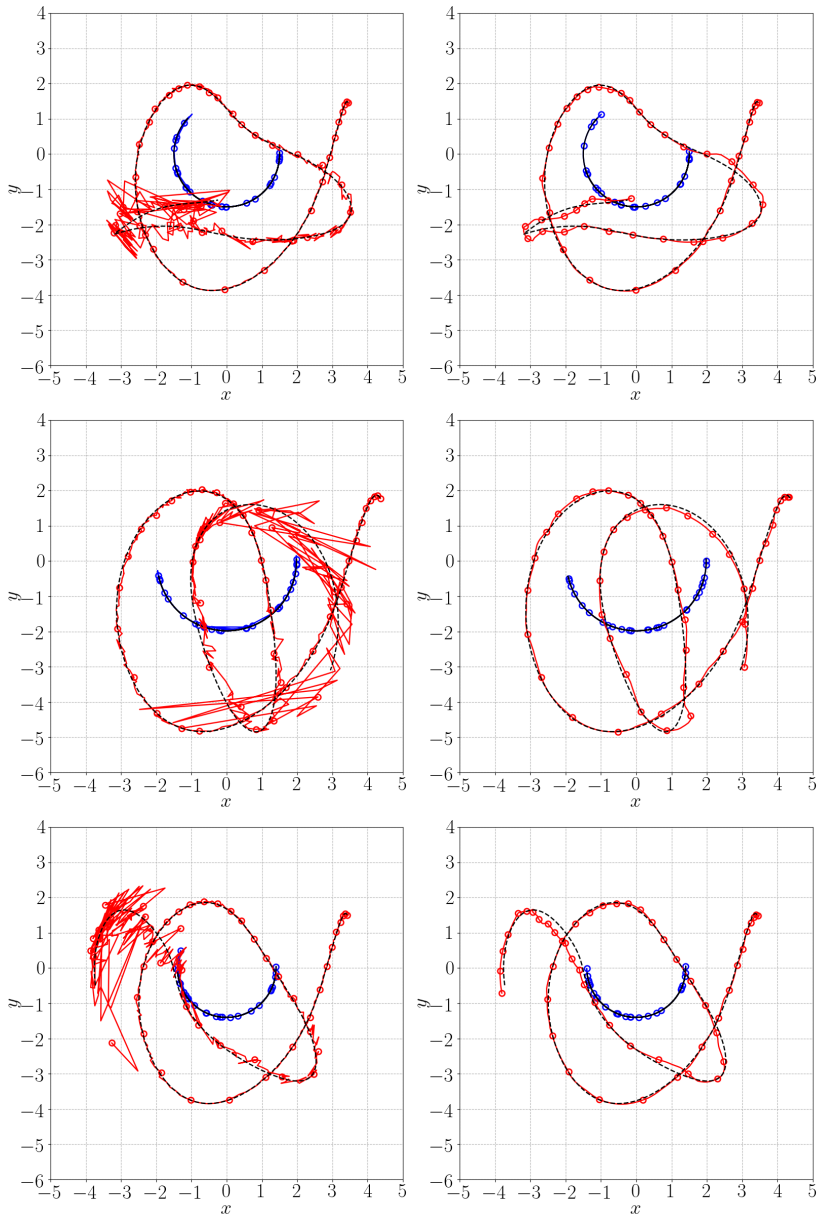

Fig. 12: Trajectories of masses m_1 and m_2 for double pendulum problems: Labels(m_1: black solid, m_2: black dashed) vs. Predictions(m_1:blue solid,circles, m_2:red solid,circles) for multiple inputs $L_1 = 1.500[m]$, $L_2 = 2.410[m]$, $\dot{\theta}_1^0 = 0.03[rad/s]$, $\dot{\theta}_2^0 = 0.330[rad/s]$ (Top), $L_1 = 1.980[m]$, $L_2 = 2.940[m]$, $\dot{\theta}_1^0 = 0.087[rad/s]$, $\dot{\theta}_2^0 = 0.470[rad/s]$ (Middle), $L_1 = 1.400[m]$, $L_2 = 2.500[m]$, $\dot{\theta}_1^0 = 0.060[rad/s]$, $\dot{\theta}_2^0 = 0.380[rad/s]$ (Bottom). Results from two types of training set S_{fixed}(Left) and S_{full}(Right) are compared.

and

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{x_B} \end{bmatrix} = \begin{bmatrix} L\sin\phi & 1 \\ -L\cos\phi & 0 \end{bmatrix}^{-1} \left(\begin{bmatrix} -L\dot{\phi}\cos\phi & 0 \\ -L\dot{\phi}\sin\phi & 0 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{x_B} \end{bmatrix} + \begin{bmatrix} -r\dot{\theta}^2\cos\theta \\ -r\dot{\theta}^2\sin\theta \end{bmatrix} + \begin{bmatrix} -r\ddot{\theta}\sin\theta \\ r\ddot{\theta}\cos\theta \end{bmatrix} \right). \tag{15}$$

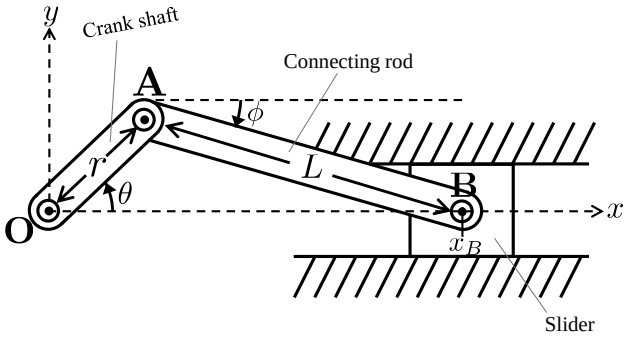

Fig. 13: Slider crank mechanism. Gravity acceleration g is fixed to $g = 9.81[m/s^2]$. The constant $\tau \in [1,2]$, the length of crank shaft $r[m] \in [1,3]$, the ratio of the lengths between the connecting rod and the crank shaft $L/r \in [2.5,3.5]$ are arbitrarily determined within the given ranges.

Two meta-models are generated from S_{fixed} and S_{full} types of training data, by employing the hyper-parameters found from grid searches for the case of S_{full}shown in Table 10.

Hyper-parameters	Choice
The number of hidden layers	2
The number of nodes in each layer	128
The size of batch	64
The number of epochs	200
Loss function	$\mathscr{E}_{mse}$
Optimizer	Adam

Table 10: Hyper-parameters for the slider crank problem

The scatter plots in Fig. 14 compares labels and predictions of the meta-model from S_{full}, and verifies that the meta-model produces almost accurate results. Its performance is much better than the other meta-models of previous examples, which seems to be caused by a simple form of kinematic equations (13) and a sufficient training data set. The $\mathscr{R}^2$ values are over 0.999 for the kinematic responses θ, ϕ, $\ddot{\phi}$, x_B, and $\dot{x}_B$.

	Parameters	Ranges	Meshsizes for Training Data	Meshsizes for Test Data
Fixed constants	$\theta^0[rad]$	0	·	·
Inputs	τ	[1, 2]	$\Delta\tau = 0.1$	arbitrary(not uniform)
	$r[m]$	[1, 3]	$\Delta r = 0.2$	arbitrary(not uniform)
	L/r	[2.5, 3.5]	$\Delta(L/r) = 0.1$	arbitrary(not uniform)
Time instants	$\{t_n\}[s]$	[0, 5]	$\Delta t = 0.01$ ($t_0 = 0$)	$\Delta t = 0.01$ ($t_0 = 0$)

Table 11: Summary on parameters of slider crank problem. In S_{fixed}, a fixed time instant is considered. In S_{full}, all the time instants are treated as inputs.

Since the predictions for test data are highly accurate as confirmed in Fig. 14, Fig. 15, 16, and 17 present results only for a specific case of test data: $\tau = 1.780, r = 1.360, L/r = 3.050$. Fig. 15 shows changes of translation and velocities of the slider mass B in time t. As shown in previous Sections 4.1 and 4.2, S_{fixed}(Left) shows oscillatory waves, while S_{full} yields smooth solutions. The error $\mathscr{E}_{mse}$ compares the difference of their accuracies more clearly.

Fig. 16 displays time-varying relations between the angle of connecting rod $\phi(t)$ and its temporal derivatives $(\dot{\phi}(t), \ddot{\phi}(t))$. The oscillations from the case of S_{fixed} (Left) are observed. Fig. 17 shows relations between the displacement of slider x_B and its derivatives. Performance of two training data set S_{fixed} (Left) and S_{full} (Right) is more clear than Fig. 16. S_{full} yields more smooth and accurate results than S_{fixed}.

5 Conclusions

The present study introduces a procedure to combine a machine learning and solution of general purpose multibody dynamics. The paper contributes to data-driven modeling for multibody systems in two meaningful aspects. The first is that Deep Neural Network learning is applied, not to a specified particular type, but a *general* multibody dynamic problem. The generality makes it possible for the proposed DNN algorithm to be employed for other multibody system problems in future research. The second is that the present work analyzes and suggests how training data need to be structured for more effective DNN learning. In particular, it is found out that treating time variable as an input parameter enhances accuracy and smoothness of resulting predictions. The observation is worthwhile to notice, since the smoothness of physical variables in time direction is significant in dynamic problems. The paper demonstrates that the accurate solution of general purpose multibody dynamics can be achieved by DNN procedure. Despite the introduced numerical results, the present data-based learning algorithm can be improved through further studies. For one thing, performing *smart sampling* which decides more suitable ranges and non-uniform mesh sizes of data will improve computational efficiency in generating a meta-model. Moreover, to make fundamental progress in data-driven design of MBD, further studies are required on other various subjects, from theories on probability, uncertainties, and physics, to brand-new data-handling techniques.

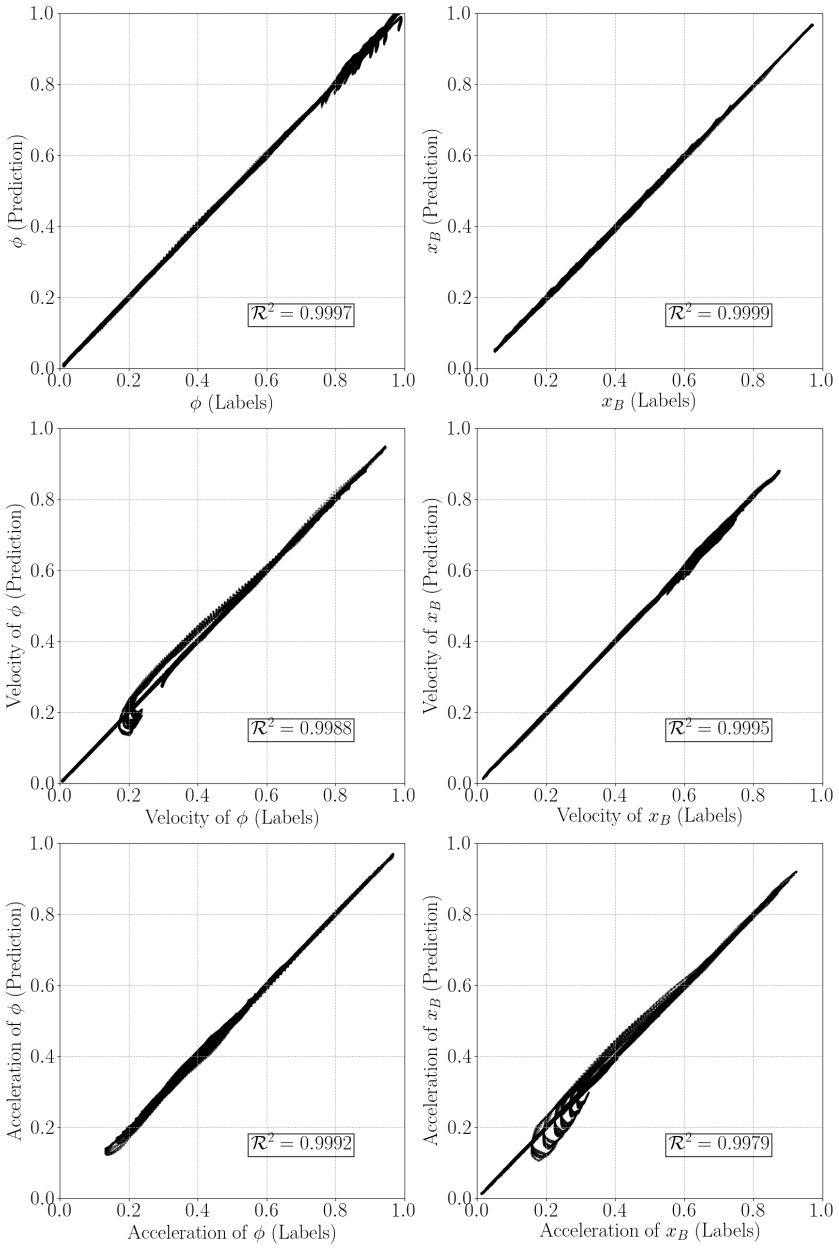

Fig. 14: Labels vs. Predictions for normalized test data. The meta-model for the slider crank problem is generated from S_{full} type of training set. Test data are *unseen* from training. The $\mathscr{R}^2$ scores are almost 1, which implies that the DNN model predicts output solutions with high accuracy.

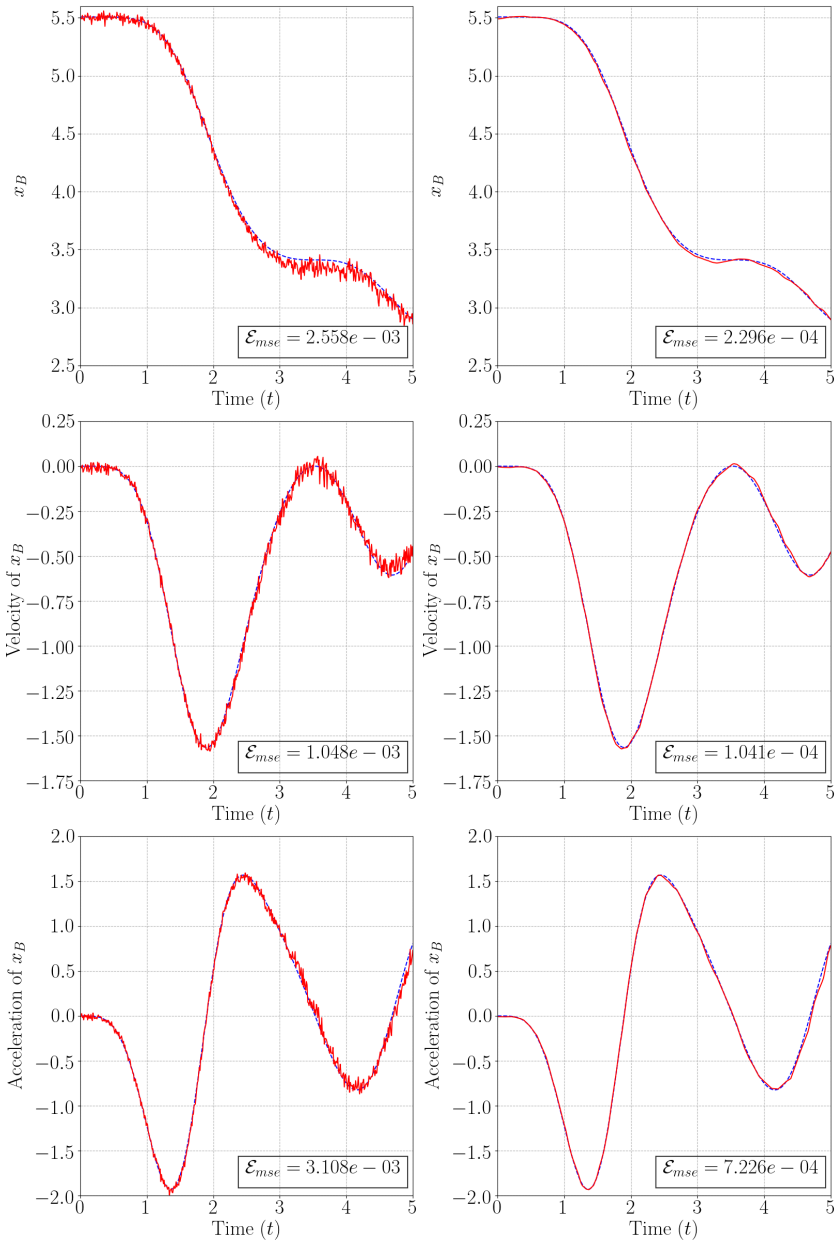

Fig. 15: Dynamic responses of slider crank: Labels(blue dashed) vs. Predictions(red solid) for specific input $\tau = 1.780$, $r = 1.360$, and $L/r = 3.050$. Left:$\#\{t_n\}$ numbers of meta-models are generated for each fixed time $t = t_n$ (S_{fixed}). Some oscillations are observed. Right:When time variable t is considered as an input parameter (S_{full}). Relatively smooth solutions are achieved.

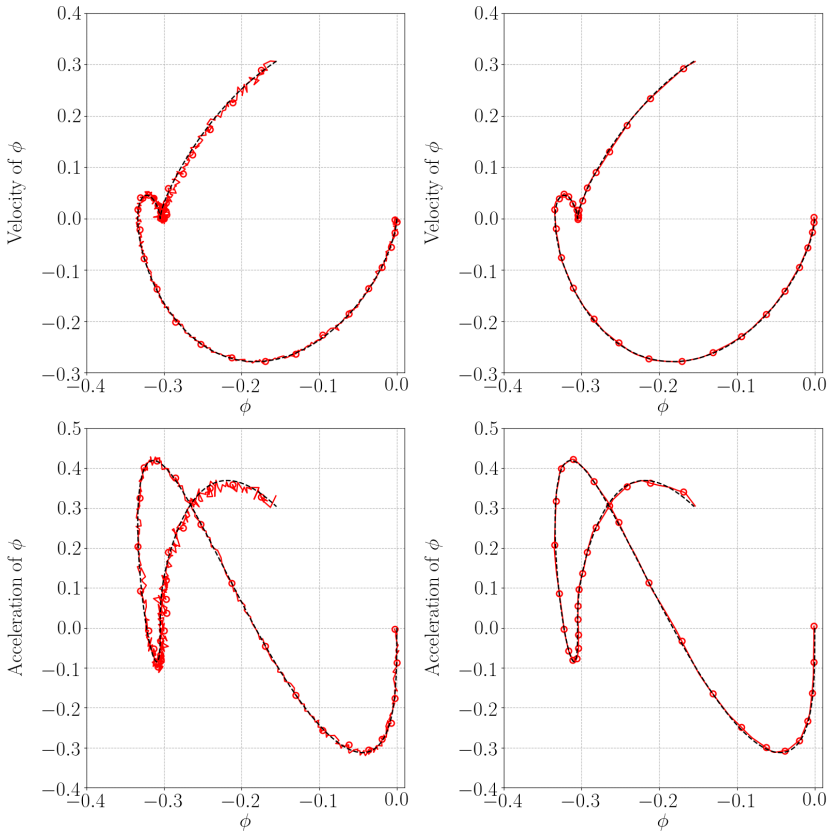

Fig. 16: Relations between dynamic responses of slider crank problem when $\tau = 1.780$, $r = 1.360$, and $L/r = 3.050$: Labels(black dashed) and predictions (red solid, circles) are given. Results from different types of training data set S_{fixed}(Left) and S_{full}(Right) are compared. S_{full} yields more smooth and accurate dynamic results.

Acknowledgements This research is supported by 2018-2019 KyungHee University Research Support Program.

References

1. Kingma, Diederik P., and Ba, Jimmy Lei, Adam: A method for stochastic optimization, arXiv:1412.6980v9, (2014)
2. Rumelhart, David E., Hinton, Geoffrey E., Williams, Ronald J., Learning Representations by Back-propagating Errors, Nature, 323, 533–536, (1986)
3. Hinton, G. Neural Networks for Machine Learning - Lecture 6a - Overview of mini-batch gradient descent, (2012)
4. Lanz, O., Approximate Bayesian Multibody Tracking, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 28, 9, (2006)
5. Shabana, A.A., Dynamics of Multibody Systems, Cambridge University Press, Cambridge (2005)
6. Pontes, F. J., Amorim, G. F., Balestrassi, P. P., Paiva, A. P., Ferreira, J. R., Design of experiments and focused grid search for neural network parameter optimization, Neurocomputing, 186, 22-34, (2016)

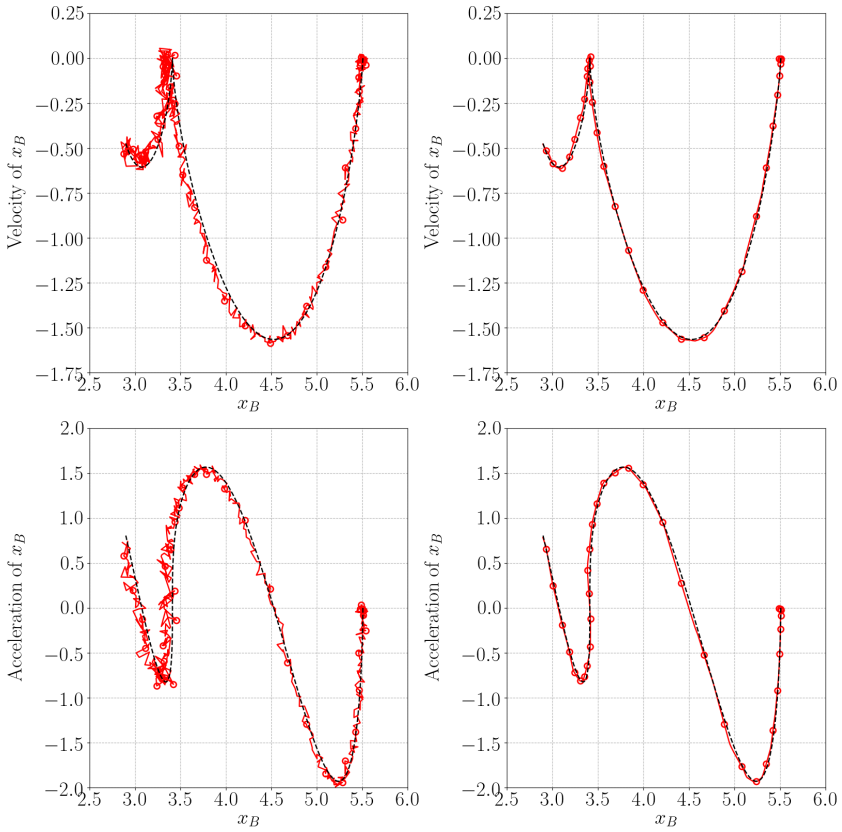

Fig. 17: Relations between dynamic responses of slider crank problem when $\tau = 1.780$, $r = 1.360$, and $L/r = 3.050$: Labels(black dashed) and predictions (red solid, circles) are given. Results from different types of training data set S_{fixed}(Left) and S_{full}(Right) are compared. S_{full} yields more smooth and accurate dynamic results.

7. Huang, C. M., Lee, Y. J., Lin, D. K., Huang, S. Y., Model selection for support vector machines via uniform design, Computational Statistics & Data Analysis, 52(1), 335-346, (2007)
8. Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., Hutter, F, Practical automated machine learning for the automl challenge 2018, In International Workshop on Automatic Machine Learning at ICML (pp. 1189-1232), (2018, July)
9. Goodfellow, I., Bengio, Y., Courville, A, Deep learning, MIT press, (2016)
10. Bergstra, J., Bengio, Y, Random search for hyper-parameter optimization, Journal of Machine Learning Research, 13(Feb), 281-305, (2012)
11. LeCun, Y., Bengio, Y., Hinton, G, Deep learning, Nature, 521(7553), 436, (2015)
12. Domingos, P. M., A few useful things to know about machine learning, Commun. acm, 55(10), 78-87, (2012)
13. Blanco-Claraco, J.L., Torres-Moreno, J.L., Giménez-Fernández, A., Multibody dynamic systems as Bayesian networks: Applications to robust state estimation of mechanisms, Multibody Syst Dyn, 34, 103-128, (2015)
14. Li, Y., Wu, J., Tedrake, R., Tenenbaum, J.B., Torralba, A., Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids, ICLR 2019.

15. Ting, J-A., Mistry, M., Peters, J., Schaal, S., Nakanishi, J., A Bayesian Approach to Nonlinear Parameter Identification for Rigid Body Dynamics, Robotics: Science and Systems II., 247-254, (2007)
16. Tutsoy, O., Brown, M., Wang, H., Reinforcement learning algorithm application and multi-body system design by using MapleSim and Modelica, International Journal of Advanced Mechatronic Systems, (2012)
17. Lin, Y-C., Haftka, R.T., Queipo, N.V., Fregly, B.J., Surrogate articular contact models for computationally efficient multibody dynamic simulations, Medical Engineering and Physics, 32, 6, 584-594, (2010)
18. Halloran, J.P., Erdemir, A., van den Bogert, A.J., Adaptive Surrogate Modeling for Efficient Coupling of Musculoskeletal Control and Tissue Deformation Models, J. Biomech. Eng., 131(1) (2009)
19. Ansari, H., Tupy, M., Datar, M., Negrut, D., Construction and Use of Surrogate Models for the Dynamic Analysis of Multibody Systems, SAE International by Columbia Univ, (2018)
20. Kraft,S., Causse, J., Martinez, A., Black-box modelling of nonlinear railway vehicle dynamics for track geometry assessment using neural networks, International Journal of Vehicle Mechanics and Mobility, (2018)
21. Falomi, S., Malvezzi, M., Meli, E., Multibody modeling of railway vehicles: Innovative algorithms for the detection of wheel-rail contact points, Wear, 271, 453-461, (2011)
22. Martin, T.P., Zaazaa, K.E., Whitten, B., Tajaddini, A., USING A MULTIBODY DYNAMIC SIMULATION CODE WITH NEURAL NETWORK TECHNOLOGY TO PREDICT RAILROAD VEHICLE-TRACK INTERACTION PERFORMANCE IN REAL TIME, Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, (2007)
23. Byravan, A., Fox, D., SE3-nets: Learning rigid body motion using deep neural networks, IEEE International Conference on Robotics and Automation (ICRA), (2017)