Mechanical Systems and Signal Processing 240 (2025) 113220

Contents lists available at ScienceDirect

Mechanical Systems and Signal Processing

journal homepage: www.elsevier.com/locate/ymssp

Full length article

Real-time structural dynamics estimation in hydraulically actuated systems using 3D flexible multibody simulation and SLIDE networks

Qasim Khadim [a],[*], Peter Manzl [b], Emil Kurvinen [a], Aki Mikkola [c], Grzegorz Orzechowski [c], Johannes Gerstmayr [b]

[a] *University of Oulu, Pentti Kaiteran katu 1, Oulu, 90570, Finland*
[b] *University of Innsbruck, Technikerstraße 13, Innsbruck, 6020, Austria*
[c] *LUT University, Yliopistonkatu 34, Lappeenranta, 53850, Finland*

ARTICLE INFO

Communicated by G. Totis

Keywords:
Structural dynamics estimation
Neural network model
Flexible multibody systems
Hydraulics
Robot manipulator
Machine learning

ABSTRACT

The precision, stability, and performance of lightweight high-strength steel structures in heavy machinery is affected by their highly nonlinear structural dynamics. This, in turn, makes control more difficult, simulation more computationally intensive, and achieving real-time autonomy, using standard approaches, impossible. Machine learning through data-driven, physics-informed and physics-inspired networks, however, promises more computationally efficient and accurate solutions to nonlinear dynamic problems. This study proposes a physics-inspired real-time structural dynamics estimation framework using novel SLIDE-neural networks for the hydraulically actuated three-dimensional systems.[1] It learns the dynamics of a system by utilizing physically known damping properties in a SLIDE window. For data acquisition, an algorithm is introduced from randomized initial configurations and hydraulic pressures in a system, along with a method to compute the SLIDE size. The new framework was evaluated across varying geometries, and 1-DOF and 2-DOF hydraulically actuated systems. Different sensor configurations while lifting various payloads were also tested. The SLIDE-trained network accelerated structural dynamics estimation solutions by a factor of 10^3 in reference to flexible multibody simulation batches and provided reasonable accuracy. Performance of new framework is compared with the sequential architectures such as RNN, LSTM and CNN. The SLIDE network was successfully trained in less time using standard parameters from PyTorch, ADAM optimizer. These results support the studies goal of providing robust, real-time solutions for control, robotic manipulators, structural health monitoring, and automation problems.

1. Introduction

1.1. Research background: Structural deflection—Impacting automation in heavy machines

Heavy machinery is important for primary production and transportation to enable a modern lifestyle [1]. Most of these machines are human-operated and typically tailored for specific hydraulic actuation [2]. However, there is a growing trend towards increasing automation to enhance operational safety, productivity, and efficiency [3–5]. Increasing the level of automation, leading eventually to autonomous operation, requires accurate information on physical dynamics, particularly structural dynamics during

* Corresponding author.
E-mail address: qasim.khadim@oulu.fi (Q. Khadim).
1 GitHub source, https://github.com/qkhadim22/SLIDE-EstimatingStructuralDeflection.git

https://doi.org/10.1016/j.ymssp.2025.113220
Received 24 February 2025; Received in revised form 28 June 2025; Accepted 10 August 2025
Available online 23 September 2025

operation [6,7]. Flexible behavior, a result of material properties, geometry and other factors [8], can significantly affect the precision and accuracy needed to achieve autonomous operation [9–11].

1.2. Research motivation: Electrification and its impact on structural deflections in heavy machinery

Additionally, emerging trends in electrification are driving engineers to innovate lighter structural designs to compensate for the weight of batteries and other heavy components [12,13]. This goal can be achieved using lightweight materials like modern metal alloys, *e.g.*, ultra-high-strength steel or aluminum, composites, and sandwich structures [13]. However, this shift towards lighter structures comes with new challenges. As the machinery becomes lighter, its structural components also become more flexible, which can affect stability and performance, especially under dynamic conditions [9,14,15].

1.3. State of art methods and their limitations

The increased flexibility makes it more difficult to maintain precise control over the machine, particularly when it experiences varying or sudden loads [14,15]. State-of-the-art methods for interpreting structural flexibility include direct measurement [16,17], dynamic models [9,18,19], and Machine Learning (ML) [20–23]. The direct measurement methods—such as digital image correlation [24], fiber optic sensors [25], wireless sensor networks [26], ultrasonic nondestructive testing [27], laser scanning, and LIDAR [28,29]—are commonly used in civil engineering applications. Real-time processing demands and implementation constraints are limitations of these methods.

In heavy machines conventional strain sensors can interpret flexible behavior at specific locations on the structure. However, these sensors have several disadvantages, such as high cost, insufficient reliability, limited measuring points, and poor real-time capability due to challenging working conditions. To overcome these challenges, alternative approaches such as dynamic modeling and ML offer promising solutions for accurately capturing the flexible behavior of components in heavy machines.

1.4. Research gaps: Conventional dynamic modeling and ML approaches in controlling flexibility

In flexible systems, dynamic modeling involves links and joints or both [10]. The Assumed Mode Method (AMM) [30], Lumped Parameter Model (LPM) [9], Transfer Matrix Method (TMM) [31], and Finite Elements Method (FEM) [15] are commonly used to model flexible links. Broadly speaking, these methods demonstrate limitations in real-time control due to high computational cost, complex-geometry handling [32], many Degrees Of Freedom (DOFs) [15], and improper boundary conditions and modes [10,18,33]. A detailed description of flexible systems modeling methods and their pitfalls can be found in [10].

On the other hand, in flexible multibody systems, the Floating Frame of Reference Formulation (FFRF) is one of the most widely used methods in engineering applications involving large translations and rotations [34]. Across several FFRF versions, generalized Component Mode Synthesis (CMS) has been proposed as a suitable way to analyze 3D flexible systems [35,36]. The Equations Of Motion (EOMs) in CMS are further simplified using modal reduction techniques based on eigenmodes, which yields reasonable computational efficiency and accuracy [37].

From the application perspective, hydraulic actuators, valves, pumps and their modeling methods are crucial factors [38]. The outlook on faults, diagnostics, and prognosis of hydraulic actuators is discussed in Kumar et al. [39]. Hydraulics are coupled with the multibody systems using the lumped fluid theory [40–42]. It computes hydraulic pressure derivatives in a volume by dividing the effective bulk modulus with a small volume size [40]. This method leads to numerically stiff EOMs for a coupled system negatively affecting computational burden [41,42]. Due to this reason, most of the literature in this field has focused on coupling rigid bodies with hydraulics [41–43], ignoring flexible bodies. Nevertheless, the real-time processing of structural dynamics in flexible systems remains challenging due to model complexity and the handling of flexible bodies with the hydraulics and electric components, contacts, and friction.

On the contrary—ML offers computationally efficient and accurate solutions in diverse fields such as image processing [44,45], playing Atari games [46], natural language processing [47], fluid mechanics [48], heat transfer [49], and the real-time estimation of rigid [50–53] and flexible multibody dynamics [54–56]. In ML, neural networks often act as black-box models that effectively map input–output relationships [57]. Neural networks have been used to develop Hamiltonian Neural Networks (HNNs) [58] and Physics-Inspired Neural Network (PI-NN) [59] for rigid multibody systems, to estimate structural deformations [23] and deflections [21,22,60], and to replace flexible multibody simulation [54,55,61,62]. In hydraulics, ML has also been employed to replace pressures [63], cylinder friction forces [64], control valve [65] and hydraulic forces [43,66].

As universal approximators, neural networks often overlook the underlying system mechanics. For flexible systems, the studies in [54–56] utilize neural networks within a time-stepping scheme [67], which involves estimating a state vector at Tth step. The solution of the previous step is used autoregressively, which significantly affects training data size and computational training resources. To accelerate supervised learning, Principal Component Analysis (PCA) was used to reduce the output vector and data size in flexible multibody systems [54]. In a hydraulically driven application [62], a data-driven deep neural network (DNN) modeling approach was introduced for real-time flexible multibody dynamics simulations using coarse sample data. However, data acquisition was performed using mesh-based material parameters and position level inputs while ignoring actuator dynamics. The employed DNN architecture was also complex [62].

Summary of identified research gaps — current neural network methods for hydraulically flexible multibody systems: (i) often ignore actuator dynamics, (ii) require large high-fidelity training datasets, and (iii) lack validation across varying geometries and system configurations. When actuator dynamics are considered, each configuration of a hydraulic actuator in a coupled system requires unique chamber pressures, thereby introducing computational challenges. It challenges the data collection, extensive preprocessing, identifying exact input–output, and data quality [47] for hydraulically actuated flexible systems. Neural networks, such as Physics-Informed neural networks (PINNs) [49] and PI-NNs [59], can also address computationally intensive problems by incorporating physical laws, leading to improved performance and efficiency.

1.5. Research outline

A PI-NN, the SLiding-window Initially-truncated Dynamic-response Estimator (SLIDE) method, is introduced [61] to capture the underlying physics of a system using the SLIDE window $\mathrm{t_d}$. In damped systems, $\mathrm{t_d}$ demonstrates the time required by a system to reach the steady-state condition under forced excitation. The approximation of $\mathrm{t_d}$ is based on the complex eigenvalues of the system's linearized EOMs [61]. EOM-based $\mathrm{t_d}$ computing approach might be challenging for a hydraulically actuated system due to the highly nonlinear dynamics and discontinuities. The SLIDE method provided good performance in estimating the position, velocity and structural deflection vectors of bodies for the flexible multibody systems [61] with the standard PyTorch parameters [68] for Adam optimizer [69]. However, the SLIDE method has not yet been explored in the framework of hydraulically actuated flexible systems accounting actuator dynamics.

This study introduces a physics-inspired real-time structural dynamics estimation framework based on a novel SLIDE approach for hydraulically actuated 3D flexible multibody systems. A computationally efficient data acquisition approach is proposed to generate data from random initial configurations and a method to compute $\mathrm{t_d}$ without the EOMs in hydraulically actuated systems. This framework is evaluated on the hydraulically actuated flexible boom and a 2-DOF forestry crane. Additionally, different hidden layers, sensor combinations and lifting various payloads are tested to verify the robustness of approach. The trained neural network enabled the accurate and 10^3 faster estimate of structural deflection, von Mises stress and strain compared to the reference solution. The scientific contributions of this study are as follows.

- A computationally efficient data acquisition approach for hydraulically actuated flexible multibody systems from random initial configurations and a method to compute $\mathrm{t_d}$ without the EOMs
- A physics-inspired real-time structural dynamics estimation framework for hydraulically actuated 3D flexible multibody systems with SLIDE networks using various sensor and lifting loads combinations
- Validation of proposed the framework across varying geometries, 1-DOF and 2-DOF hydraulically actuated systems and sensor configurations

To evaluate effectiveness, the performance of SLIDE is compared with state-of-the-art sequential architectures Recurrent Neural Networks (RNNs) [70], Long Short-Term Memory networks (LSTMs) [71] and Convolutional Neural Networks (CNNs) [72]. The final trained neural network model enables the real-time estimate of structural performance, which is particularly beneficial in applications where quick decision-making is critical, such as control, robotic manipulators, structural health monitoring, and automation.

2. Deep neural network (DNN) model

The general framework of the SLIDE approach is illustrated in Fig. 1. The proposed neural network model learns the nature of $\mathcal{Y}$ through regression tasks using an input layer $\mathcal{X}$ using $\mathrm{t_d}$.

The composition of $\mathcal{X}$ and $\mathcal{Y}$ for a damped mechanical system can be described as

$$\mathcal{X} = \begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \dots & \boldsymbol{x}_{\mathrm{t_d}} & \dots & \boldsymbol{x}_{\mathrm{t_d+k}} \end{bmatrix}^\top, \quad \text{and} \tag{1}$$

$$\mathcal{Y} = \begin{bmatrix} \boldsymbol{y}_{\mathrm{t_d}} & \boldsymbol{y}_{\mathrm{t_d+1}} & \dots & \boldsymbol{y}_{\mathrm{t_d+k}} \end{bmatrix}^\top, \tag{2}$$

where $\boldsymbol{x}_\mathrm{i}$ and $\boldsymbol{y}_\mathrm{i}$ are ith input and target vectors. In [61], $\mathrm{t_d}$ is approximated using the complex eigenvalues of the system's linearized EOMs. A statistics-based method in Section 4.4 describes the computing of $\mathrm{t_d}$ without the EOMs. The regression relationship in $\mathcal{X}$ and $\mathcal{Y}$ can be formulated as follows [73].

$$\hat{\mathcal{Y}}_{0:\mathrm{n_{train}}} = \mathcal{N}(\mathcal{X}_{0:\mathrm{n_{train}}}; \boldsymbol{\Psi}), \tag{3}$$

where $\hat{\mathcal{Y}}_{0:\mathrm{n_{train}}}$ is the estimated output layer, $\mathcal{N}$ is the neural network model, $\boldsymbol{\Psi}$ is the vector of trainable parameters, and $\mathrm{n_{train}}$ is the number of training samples. In supervised learning, the objective of the neural network model is to minimize the loss function $\mathcal{L}$ and keep it below a threshold value $\mathcal{L}_{\min}$ using the optimal training parameters. It can be described as

$$\operatorname{argmin} \ \mathcal{L} \quad \text{subject to} \quad \mathcal{L} \le \mathcal{L}_{\min} \tag{4}$$

The loss function is computed between the estimated output layer $\hat{\mathcal{Y}}$ and the ground truth output layer $\mathcal{Y}$ using the Mean Square Error (MSE) as follows.

$$\mathcal{L} = \frac{1}{\mathrm{n_{train}}} \sum_{i=1}^{\mathrm{n_{train}}} (\hat{\mathcal{Y}}_i - \mathcal{Y}_i)^2 \tag{5}$$

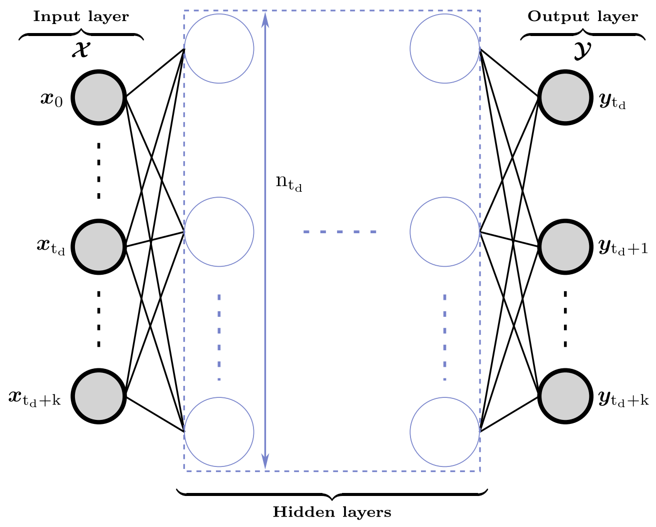


Fig. 1. Proposed method—A general framework of SLIDE-neural network model.

Adam optimizer [69] minimizes the loss function in $\mathcal{N}$ training. The hidden layers size is according to the number of steps t_d, highlighted in Fig. 1 as n_{t_d}. The standard neural network parameters Ψ^* are selected to achieve optimum training performance. In the evaluation phase, the output layer is estimated in a continuous-time frame with the trained neural network as

$$\hat{\mathcal{Y}}^{n_{eval}}_{t_d:t_f} = \mathcal{N}(\mathcal{X}^{n_{eval}}_{i:t_d+i}), \tag{6}$$

where $i = 0, 1, \dots t_f - t_d$, and t_f is the final time in the evaluation data n_{eval}.

3. General purpose flexible multibody dynamics

The equations of motion for a constrained mechanical system can be described as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + (\mathbf{g_q})^\top \lambda = \mathbf{Q}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}, t), \tag{7}$$

$$\mathbf{g}(\mathbf{q}, t) = \mathbf{0}, \quad \text{and} \tag{8}$$

$$\dot{\mathbf{y}} + (\frac{\partial \mathbf{g}}{\partial \mathbf{y}})^\top \lambda = \mathbf{f}_{\text{ODE}_1}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}, t), \tag{9}$$

where $\mathbf{q} \in \mathbb{R}^n$, $\dot{\mathbf{q}} \in \mathbb{R}^n$, and $\ddot{\mathbf{q}} \in \mathbb{R}^n$ are the position, velocity, and acceleration vectors of the generalized coordinates, respectively. $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the system mass matrix, $\mathbf{Q} \in \mathbb{R}^n$ is generalized force vector, $\mathbf{g}$ represents the holonomic constraint equations, and $\mathbf{f}_{\text{ODE}_1}$ is the vector of first-order differential equations. Here, n is the number of coordinates in the system. The left hand side of Eq. 3 includes the inertial and constraint forces in the system, whereas the right hand side represents the generalized external forces.

To explain deformable bodies, Fig. 2 describes a hydraulically actuated flexible multibody system. A flexible body A is defined with the generalized coordinate vector $\mathbf{q}_A = \begin{bmatrix} \mathbf{q}_{t_A}^\top & \theta_A{}^\top & (\overline{\boldsymbol{\Psi}}_A \boldsymbol{\zeta}_A)^\top \end{bmatrix}^\top$ [34] with $n_m = \dim(\boldsymbol{\zeta}_A) \ll \dim(\bar{\mathbf{c}}_{f_A}) = 3n_n$ in CMS [37], where $\boldsymbol{\zeta}_A \in \mathbb{R}^{3n_m \times 1}$ is the vector of modal coordinates, $\overline{\boldsymbol{\Psi}}_A \in \mathbb{R}^{3n_n \times n_m}$ is the column-wise modes reduction-basis, n_m is the modal coordinates and n_n is the number of nodes. The definition of joint and constraint equations for the multiple interconnected flexible bodies is further detailed in [74] using the modally reduced CMS. The components of Eq. (7) for a flexible multibody system can be derived from the generalized coordinate vector $\mathbf{q}$ according to [37] as follows .

$$\mathbf{M} = \begin{bmatrix} \hat{\mathbf{M}}_{tt} & \hat{\mathbf{M}}_{tr} & \hat{\mathbf{M}}_{tf} \\ & \hat{\mathbf{M}}_{rr} & \hat{\mathbf{M}}_{rf} \\ \text{sym.} & & \hat{\mathbf{M}}_{ff} \end{bmatrix}, \tag{10}$$

$$(\mathbf{g_q})^\top \lambda = \underbrace{\begin{bmatrix} \hat{\mathbf{Q}}_{c_t} \\ \hat{\mathbf{Q}}_{c_r} \\ \hat{\mathbf{Q}}_{c_f} \end{bmatrix}}_{\text{constraint force vector}}, \quad \text{and} \tag{11}$$

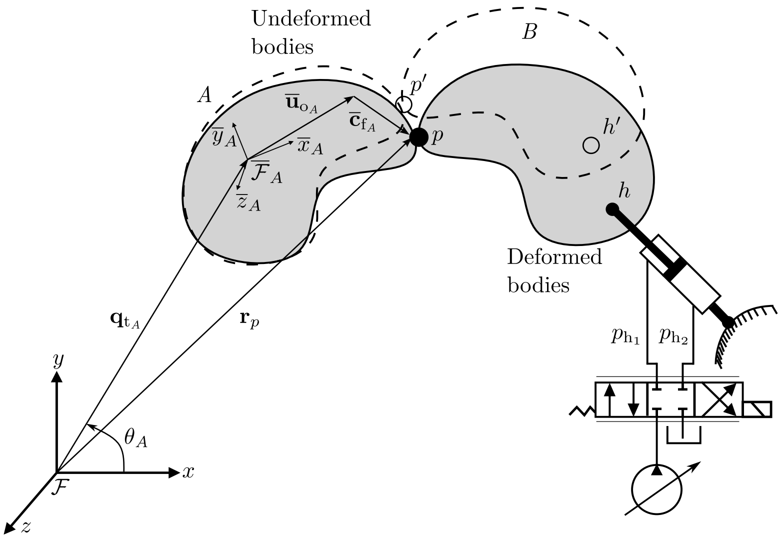


Fig. 2. Flexible system – A system consisting of two flexible bodies A and B connected at the joint p, controlled by hydraulic actuation with in $\mathcal{F}$ and floating $\overline{\mathcal{F}}_A$ frame of references – The position of the joint is $\mathbf{r}_p = \mathbf{q}_{t_A} + \mathbf{A}_A(\bar{\mathbf{u}}_{o_A} + \bar{\mathbf{c}}_{f_A})$, where $\mathbf{q}_{t_A} \in \mathbb{R}^{3\times1}$ is the translational coordinates vector, $\mathbf{A}_A = \mathbf{A}(\theta_A)$ is the rotation matrix, $\boldsymbol{\theta}_A \in \mathbb{R}^{n_r\times1}$ is the rotational parametrization vector, $\bar{\mathbf{u}}_{o_A}$ is the local coordinate vector, and $\bar{\mathbf{c}}_{f_A} \in \mathbb{R}^{3\times1}$ is the local flexible coordinates vector in body A. Additionally, n_r is the rotational DOFs.

$$\mathbf{Q} = \underbrace{\begin{bmatrix} \hat{\mathbf{Q}}_{e_t} \\ \hat{\mathbf{Q}}_{e_r} \\ \hat{\mathbf{Q}}_{e_f} \end{bmatrix}}_{\substack{\text{elastic force}\\ \text{vector}}} + \underbrace{\begin{bmatrix} \hat{\mathbf{Q}}_{v_t} \\ \hat{\mathbf{Q}}_{v_r} \\ \hat{\mathbf{Q}}_{v_f} \end{bmatrix}}_{\substack{\text{quadratic velocity}\\ \text{vector}}} + \underbrace{\begin{bmatrix} \hat{\mathbf{Q}}_{a_t} \\ \hat{\mathbf{Q}}_{a_r} \\ \hat{\mathbf{Q}}_{a_f} \end{bmatrix}}_{\substack{\text{applied force}\\ \text{vector}}} . \tag{12}$$

Further details of terms in $\mathbf{M}$ and $\mathbf{Q}$ can be found in [37]. The components of applied force vector are

$$\hat{\mathbf{Q}}_{a_t} = \sum_{i=1}^{n_n} \mathbf{f}_a{}^{(i)}, \quad \mathbf{f}_a{}^{(i)} \neq 0, \tag{13}$$

$$\hat{\mathbf{Q}}_{a_r} = \overline{\boldsymbol{G}}^\top \sum_{i=1}^{n_n} \left[\tilde{\bar{\boldsymbol{x}}}^{(i)} + \sum_{m=1}^{n_m} \tilde{\overline{\boldsymbol{\Psi}}}^{(i)}_m \zeta_m \right] \mathbf{A}^\top \mathbf{f}_a{}^{(i)}, \quad \mathbf{f}_a{}^{(i)} \neq 0, \quad \text{and} \tag{14}$$

$$\hat{\mathbf{Q}}_{a_f} = \sum_{i=1}^{n_n} \begin{bmatrix} \overline{\boldsymbol{\Psi}}^{(i)\top}_1 \\ \vdots \\ \overline{\boldsymbol{\Psi}}^{(i)\top}_{n_m} \end{bmatrix} \mathbf{A}^\top \mathbf{f}_a{}^{(i)}, \quad \mathbf{f}_a{}^{(i)} \neq 0, \tag{15}$$

where $\mathbf{f}_a{}^{(i)}$ is the applied force vector at an arbitrary node i, and $\tilde{\bar{\boldsymbol{x}}}^{(i)}$ is the undeformed (reference) nodal coordinates. In hydraulic actuation, $\mathbf{f}_a{}^{(i)}$ can be computed from the hydraulic force vector $\mathbf{F}_h$ as follows.

$$\mathbf{f}_a{}^{(i)} = w^{(i)} \mathbf{F}_h, \tag{16}$$

where $w^{(i)} = \frac{1}{3A_B} \sum_j A_j$ and $\sum_i w^{(i)} = 1$ is the nodal weighting, A_j is the area of jth triangle, and A_B is the total boundary area at RBE2 interface. This weighting ensures nearly constant strain distribution with equally distributed axial forces. The force produced by the hydraulic cylinder is calculated as $F_h = p_{h_1} A_{h_1} - p_{h_2} A_{h_2} - F_\mu(\dot{s})$, where p_{h_1} and p_{h_2} correspond to the hydraulic pressures on cylinder areas A_{h_1} and A_{h_2}, respectively. A velocity-dependent Stribeck friction model is used to calculate friction force, $F_\mu(\dot{s})$, in the cylinder according to [75,76]. The hydraulic force is converted into the vector form as $\mathbf{F}_h = F_h \mathbf{r}_h$. Following Fig. 2, the position vector $\mathbf{r}_h$ is computed between the nodal point H and ground. The hydraulic pressure, p_h, in a volume V_h, can be modeled using lumped fluid theory [40] in differential form as follows.

$$\dot{p}_h = \frac{B_{e_h}}{V_h} \left(-\frac{dV_h}{dt} + Q_s \right) \quad \text{and} \tag{17}$$

$$B_{e_h} = \left(\frac{1}{B_o} + \sum_{c=1}^{n_h} \frac{V_c}{V_h B_c} \right)^{-1}, \tag{18}$$

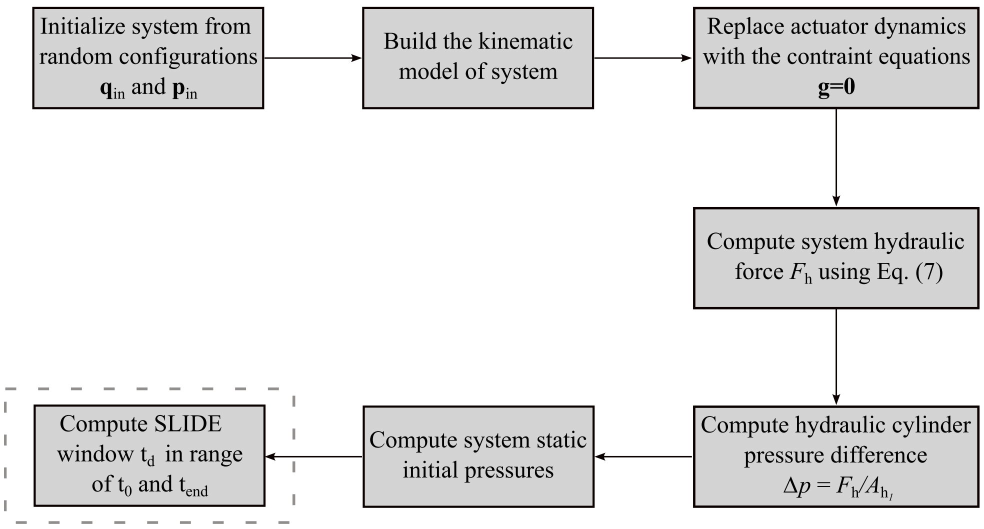

Initialize system from random configurations $\mathbf{q}_{\text{in}}$ and $\mathbf{p}_{\text{in}}$
Build the kinematic model of system
Replace actuator dynamics with the contraint equations $\mathbf{g=0}$
Compute system hydraulic force F_{h} using Eq. (7)
Compute hydraulic cylinder pressure difference $\Delta p = F_{\text{h}}/A_{\text{h}_l}$
Compute system static initial pressures
Compute SLIDE window t_{d} in range of t_0 and t_{end}

Fig. 3. SLIDE data acquisition – A method for generating data from hydraulically actuated mechanical systems at random initial configurations – Initial pressures are calculated based on the static equilibrium force for each system configuration. The SLIDE window t_{d} is also determined during the data acquisition, and the data is arranged accordingly.

where $B_{\text{e}_{\text{h}}}$ is the effective bulk modulus, Q_{s} is the sum of the incoming and outgoing flows, B_{o} is the oil bulk modulus, n_{h} is the total number of hydraulic volumes, V_{c} denotes the sub-volume, and B_{c} is the bulk modulus of the sub-volume. A semi-empirical method can be used to compute the flow rate [40]. Note that the non-linear effects such as pipeline pressure fluctuations and valve leakages have not been considered in this study [38]. The proposed network needs training data from various initial configurations of the system. Specifically for hydraulic systems, this results in computational challenges because each system's initial configuration must reach static equilibrium. To this end, the algorithm in Fig. 3 is introduced to achieve static equilibrium at random system configuration $\mathbf{q}_{\text{in}}$ and initial pressure $\mathbf{p}_{\text{in}}$.

For a random $\mathbf{q}_{\text{in}}$, the actuator dynamics in the system are replaced with the equivalent constraint equations $\mathbf{g}$ in a closed kinematic loop. At static equilibrium, the EOMs are reduced into $(\mathbf{g}_{\mathbf{q}})^{\top}\lambda = \mathbf{Q}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}, t_0)$ and $\mathbf{g}(\mathbf{q}, t_0) = \mathbf{0}$, which results in the computation of equivalent force F_{h} for a hydraulic system. The pressure difference Δp, computed from F_{h}, enables the computation of initial pressures in the hydraulic cylinder. Note that, at static equilibrium, $F_{\mu}(\dot{s})$ contributes a negligible friction force to F_{h}. Further, the proposed algorithm facilitates computing t_{d}. In the closed-loop configuration, the full set of system coordinates is computed from the independent coordinates. For such systems, the data acquisition algorithm is complemented by the standard coordinate partitioning method.

4. Flexible simulation setup

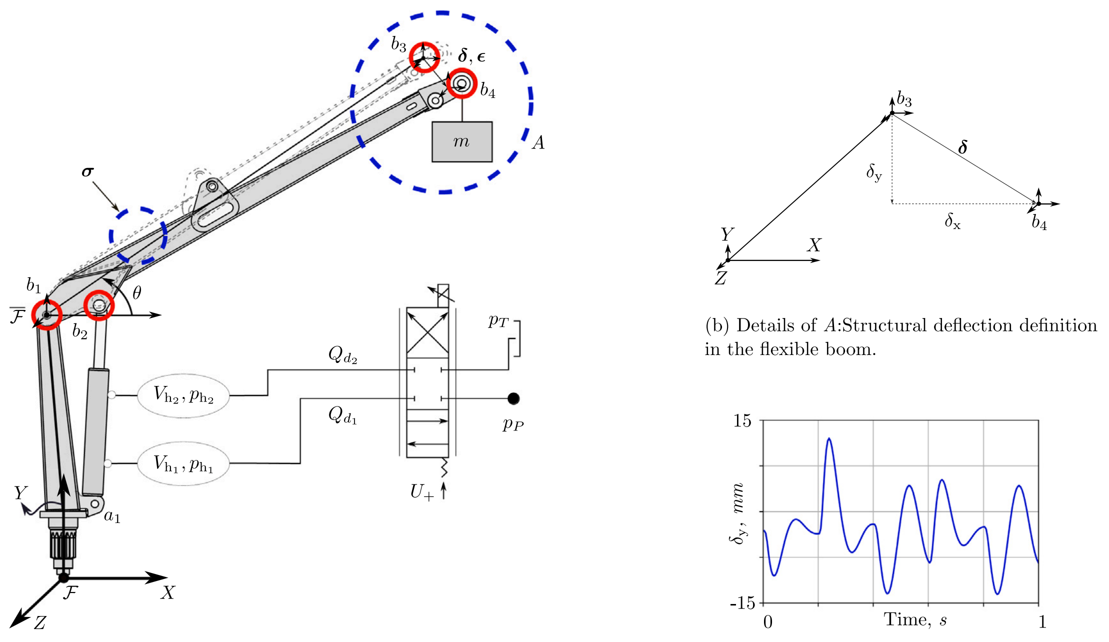
Application of the proposed method is demonstrated by estimating the structural dynamics in a hydraulically actuated flexible boom. The simulation setup of the flexible boom is described in Fig. 4a. This setup is derived from practical case studies involving forestry and material handling booms—common in heavy-duty mobile machinery. It comprises a rigid pillar, flexible boom, known payload m, and the hydraulics. High stress $\boldsymbol{\sigma}$, structural deflection $\boldsymbol{\delta}$ and strain ε on the deformed boom are shown in Fig. 4a. It is measured relative to the global coordinate system XYZ. As an example, the components of structural deflection are represented by δ_{x}, δ_{y} and δ_{z} in Fig. 4a. In this study, only few components of $\boldsymbol{\sigma}$, $\boldsymbol{\delta}$, and ε, such as σ_{xx}, δ_{y}, and ε_{xy}, are estimated due to payload forces, nonlinear friction force F_{μ} in the hydraulic cylinder, and the highly nonlinear dynamics from an application point of view. Fig. 4c further describes the uncertain and non-linear nature of tip deflection δ_{y} during a working cycle.

4.1. Flexible multibody system

The simulation setup was modeled in the open-source software Exudyn [77]. Exudyn[2] is a general-purpose flexible multibody dynamics systems modeling software in the Python environment. It includes various versions of the FFRF, including the modally reduced CMS and lumped fluid theory. The 3D boom geometry is discretized into 15101 first-order tetrahedral elements (C3D4) in the commercial FE software Abaqus®.

Fig. 5 shows the meshed flexible boom in Abaqus®. The material properties set in the Abaqus® software include Young's modulus $E = 2.1 \times 10^{11}$ Pa, Poisson's ratio $\nu = 0.3$, and density $\rho = 7850$ kg/m^3. The mass and stiffness matrices, required in Eq. (12) for

[2] Version 1.8, https://github.com/jgerstmayr/EXUDYN



(a) Simulation setup of a hydraulically actuated flexible boom.

(b) Details of A:Structural deflection definition in the flexible boom.

(c) An example of uncertain deflections.

Fig. 4. Simulation setup – A hydraulically actuated flexible boom lifting a payload m – The flexible boom is attached to rigid pillar at node b_1. It is actuated by a hydraulic cylinder, which has been added between the nodes a_1 and b_2. The known payload m is connected to the flexible boom at node b_4, and tip deflection is measured at node b_3.

Exudyn, were exported from Abaqus®. The nodal points b_1, b_2 and b_3 describe the location of joints and forces on the flexible boom. Eight Hurty Craig-Bampton modes are computed in software Exudyn using the specified boundary conditions. The eigen-frequencies of first bending and normal modes are 29 Hz and 31 Hz, respectively. The accuracy of δ_y in Exudyn was confirmed with Abaqus® during a static analysis with an approximated error of 0.5%. The flexible boom connects to the pillar at node b_1 via a revolute joint through an RBE2 standard interface.

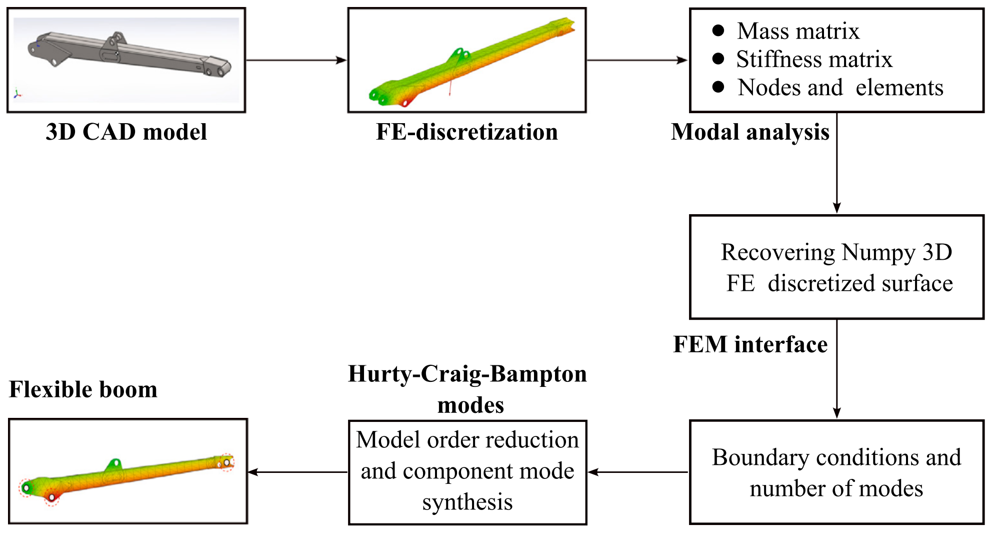

3D CAD model
FE-discretization
Mass matrix
Stiffness matrix
Nodes and elements
Modal analysis
Recovering Numpy 3D FE discretized surface
FEM interface
Flexible boom
Hurty-Craig-Bampton modes
Model order reduction and component mode synthesis
Boundary conditions and number of modes

Fig. 5. Workflow for modeling flexible boom in the Exudyn software [77] – The commercial Abaqus® environment is used to perform FE discretization of the three-dimensional lift boom model. The nodes b_1, b_2 and b_3 on the flexible boom describe the boundary conditions in the flexible multibody system.

The nodal points b_2 and b_4 on the flexible boom defines the locations of hydraulics and payload. The flexible boom was hydraulically actuated by a 4/3 directional control valve through hydraulic volumes V_1 and V_2, constant pressure sources from pump, and tank p_P and p_T. The details of the hydraulic parameters in this example are taken from an experimental setup [78]. Some hydraulic parameters can be found in Table 1.

Table 1
Hydraulic parameters of the lift boom.

Parameter	Value	Parameter	Value
Pump pressure	140 bar	Tank pressure	1 bar
Cylinder diameter	100 mm	Piston diameter	56 mm
Cylinder length (l_{cyl})	535 mm	Cylinder stroke (l_{pist})	820 mm

4.2. SLIDE data acquisition

In Exudyn, the coupled differential equations of the hydraulically actuated flexible boom are solved using the generalized-alpha integration scheme. The coupled system is solved for 1 s simulation time with a fixed time step of 5 ms, comprising each simulation of 200 steps. To generate varied training data, the flexible boom is actuated between $\theta_{\mathrm{min}} = -10°$ and $\theta_{\mathrm{max}} = +50°$, which are determined according to the minimum and maximum actuator lengths. The randomized initial angle θ_{in} for each simulation is computed as follows.

$$\theta_{\mathrm{in}} = \mathrm{rand}^{*}(\theta_{\mathrm{min}}, \theta_{\mathrm{max}}) + \theta_{min}, \tag{19}$$

where rand* is the NumPy function `numpy.random.rand()` [79], which populates random samples using a uniform distribution. For each θ_{in}, the hydraulic pressures p_{h_1} and p_{h_2} are computed through the algorithm introduced in Fig. 3 during a static analysis. The relative tolerances of 1×10^{-7} m at the position level were used in the Newton–Raphson solver for the static analysis. The simulations were run in batches of 40, 80, 160, 320, 640 and 1280 to create data in 50 s, 81.2 s, 131 s, 273.4 s, 569.3 s and 1111.8 s. The resulting data saved in the standard Numpy format. The simulation batches include 80% training data and 20% validation data.

4.3. Designing the control signal

In each simulation, the hydraulic valve was actuated using a randomly generated control signal U, as described in Fig. 6. The random signal varies between the minimum spool position $U_{\mathrm{min}} = -1$, the neutral spool position 0, and maximum spool position $U_{\mathrm{max}} = +1$. It comprises 20% values at 0, 20% values at +1, 20% values at −1, 20% linear ramped values between −1 and +1 and 20% random values between −1 and +1. The length of each ramp signal is randomly determined between the points to ensure that the total ramped signal contains 20% of total simulation steps. The different segments of the control signal are randomly shuffled to formulate the control signal, as described in Fig. 6.

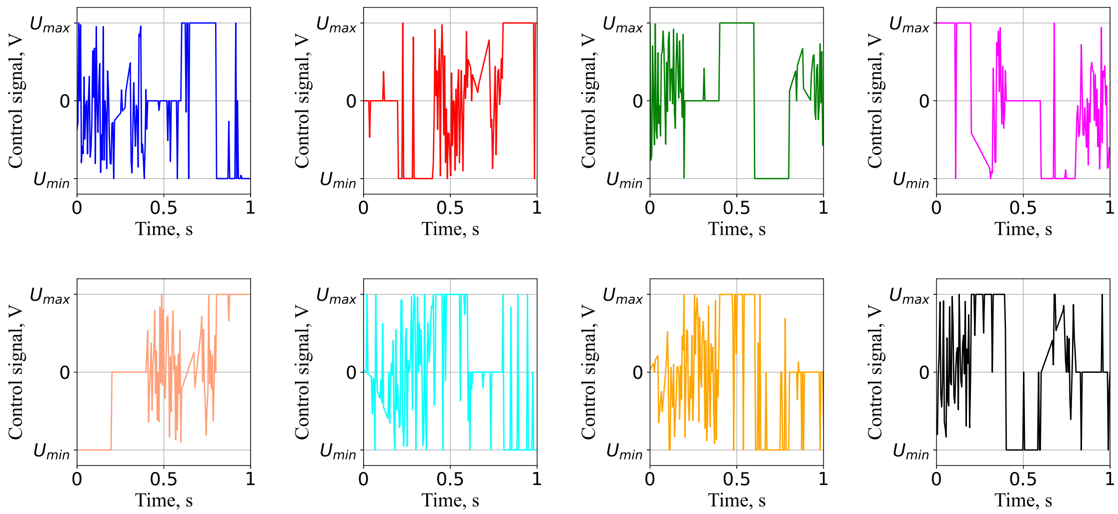


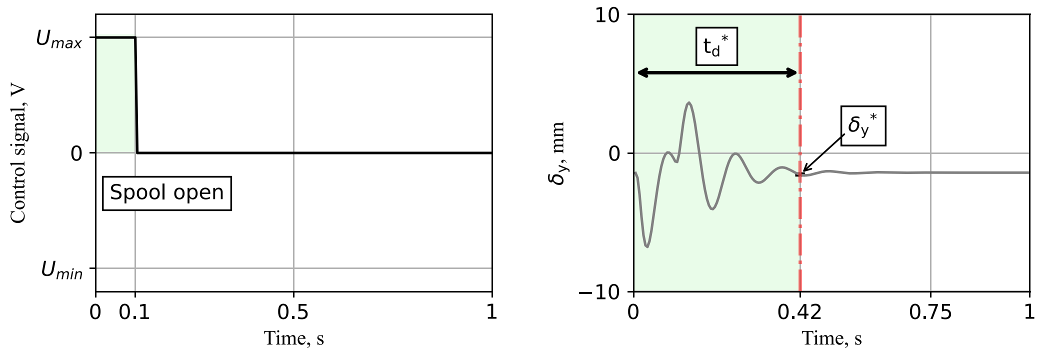
Fig. 6. Random control signal—An example of control signal in the training data.

4.4. Calculating SLIDE window

SLIDE data acquisition arranges data according to $\mathrm{t_d}$. The variable $\mathrm{t_d}^{*}$ represents the SLIDE window during a training sample. To calculate $\mathrm{t_d}^{*}$, the flexible boom is actuated from a random configuration θ_{in} using the control signal described in Fig. 7(a).

The control signal involves opening the hydraulic valve for 10% of simulation time and closing. Note that the choice of control signal distribution is in reference to EOM-based $\mathrm{t_d}$ computing method [61]. However, this distribution might change for different system configuration. ∇ is the δ_{y} data in discrete-time format from Fig. 7.

$$\nabla = \begin{bmatrix} t_0 & t_1 & t_2 & \cdots & t_{\mathrm{end}} \\ \delta_{\mathrm{y}_0} & \delta_{\mathrm{y}_1} & \delta_{\mathrm{y}_2} & \cdots & \delta_{\mathrm{y}_{\mathrm{end}}} \end{bmatrix}^{\top} \tag{20}$$



(a) Actuator valve control signal   (b) The SLIDE window is the time in computing $\delta_y{}^*$

Fig. 7. Computing SLIDE window – $t_d{}^*$ is computed from the threshold deflection $\delta_y{}^*$ by actuating the hydraulic valve with a control signal. The threshold deflection is 1% dampened value of deflection, presented in (b).

The threshold deflection $\delta_y{}^*$ is calculated from the mean structural deflection as $\delta^*_y = (1 - 2^{\frac{\log(A_{\text{rel},1\%})}{100}}\text{mean}(\delta_y))$. However, the expression for δ^*_y is case dependent and might change for different system configuration. The size of $t_d{}^*$ is determined by finding n^* step, where $\delta_y < \delta_y{}^*$ holds for $n^* = n_{\text{end}}/10$ steps, as follows.

$$\left|\delta_{y_k}\right|, \left|\delta_{y_{k+1}}\right|, \left|\delta_{y_{k+2}}\right|, \ldots, \left|\delta_{y_{k+n^*-1}}\right| < \delta_y{}^*, \tag{21}$$

where k is the first step, for $\delta_y < \delta_y{}^*$ in ∇. For n_{train} training samples, t_d is computed as

$$t_d = \frac{\sum_{n=1}^{n_{\text{train}}} t_d{}^*}{n_{\text{train}}}. \tag{22}$$

The SLIDE window is computed as $t_d{}^* = k + n^* - 1$. The size of $t_d{}^*$ depends on the payload and the Rayleigh damping factor used in the simulation. See Fig. 15 for more details. Note that this method is performed without the EOMs, which demonstrates its industrial applications with the experimental data. For the flexible boom, the statistical-based solution resulted in values of 29, 51, and 85, which are consistent with the EOM-based solutions [61].

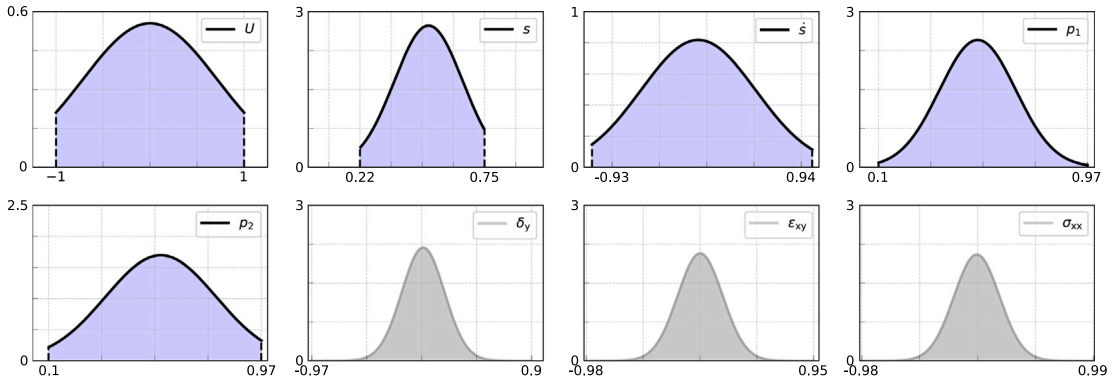


Fig. 8. Data distribution – Representation of $\mathcal{X}$ (🟦) and $\mathcal{Y}$ (⬜) on the normal probability distribution curves – Each curve is calculated using the mean and standard deviation.

4.5. Feature scaling

The virtual sensors recorded actuator position s, actuator velocity $\dot{s}$, pressures p_{h_1} and p_{h_2}, and structural deflection δ_y, strain ε_{xy} and stress σ_{xx} from the simulation setup. The sensor values are scaled for $\mathcal{X}$ as follows.

$$\mathcal{X} = \frac{1}{S_{\mathcal{X}}}\left[\boldsymbol{x}_0 \quad \boldsymbol{x}_1 \quad \ldots \quad \boldsymbol{x}_{t_d} \quad \ldots \quad \boldsymbol{x}_{t_d+k}\right]^\top, \quad \text{and} \tag{23}$$

$$\mathcal{Y} = \frac{1}{S_{\mathcal{Y}}}\left[\boldsymbol{y}_{t_d} \quad \boldsymbol{y}_{t_d+1} \quad \ldots \quad \ldots \quad \boldsymbol{y}_{t_d+k}\right]^\top, \tag{24}$$

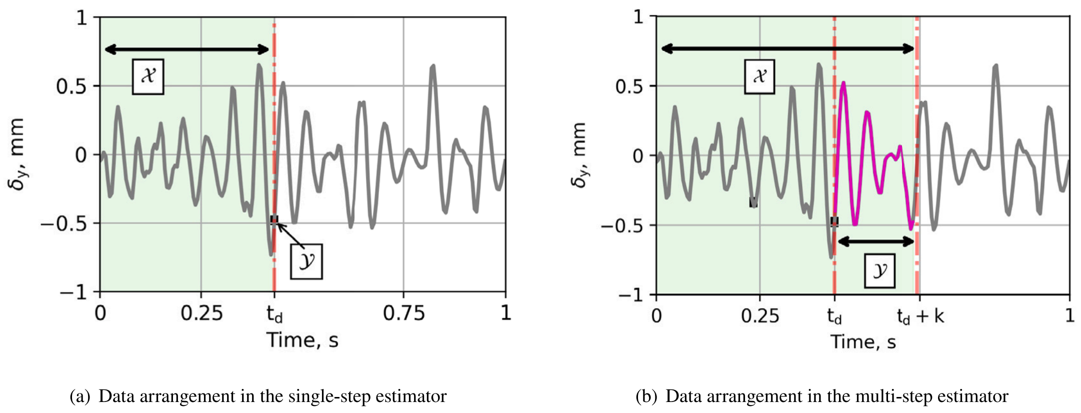
where $\boldsymbol{x}_i = \begin{bmatrix} U_i & s_i & \dot{s}_i & p_{h_{1\,i}} & p_{h_{2\,i}} \end{bmatrix}$ is the input vector, and $\boldsymbol{y}_i = \begin{bmatrix} \delta_{y_i} & \varepsilon_{xy_i} & \sigma_{xx i} \end{bmatrix}$ is the target vector at the ith step. In Eqs. (23)–(24), $S_{\mathcal{X}} = \begin{bmatrix} S_{\mathcal{X}_0} & S_{\mathcal{X}_1} & \ldots & S_{\mathcal{X}_{t_d}} & \ldots & S_{\mathcal{X}_{t_d+k}} \end{bmatrix}$ and $S_{\mathcal{Y}} = \begin{bmatrix} S_{\mathcal{Y}_{t_d}} & S_{\mathcal{Y}_{t_d+1}} & \ldots & \ldots & S_{\mathcal{Y}_{t_d+k}} \end{bmatrix}$ are the scaling matrices for $\mathcal{X}$ and

$\mathcal{Y}$. The scaling factor for actuator position is calculated using $l_{\text{cyl}} + l_{\text{pist}}$. The actuator velocity and hydraulic pressures are scaled according to the maximum velocity, 6.67 m/s, and maximum pressures, 200 bar, produced by the hydraulic cylinder. Similarly, the output scaling factor is determined based on the maximum capacity of the strain measurement sensor, which is 30 mm.

The quality of the training data is further accessed by analyzing $\mathcal{X}$ (🟦) and $\mathcal{Y}$ (⬜) on the normal probability distribution curves. See Fig. 8. The x-axis in these plots demonstrates the range of scaled measurements, and the y-axis is the probability density of the fitted normal distribution. As shown, the specified scaling factors scale s between $+0.22$ and $+0.75$, $\dot{s}$ between -0.93 and $+0.94$, pressures between 0.1 and $+0.97$, δ_{y} between -0.97 and $+0.9$, ε_{xy} between -0.97 and $+0.9$ and σ_{xx} between -0.97 and $+0.9$. Positive pressures in the hydraulic cylinder are ensured during the data acquisition process.

4.6. Multi-step estimation

Using t_{d}, the SLIDE-neural networks can be used for both single-step and multi-step estimations. Figs. 9(a)–9(b) illustrate the data arrangement of $\mathcal{X}$ and $\mathcal{Y}$ for both estimation approaches.



(a) Data arrangement in the single-step estimator (b) Data arrangement in the multi-step estimator

Fig. 9. SLIDE data arrangement – Arrangement of $\mathcal{X}$ and $\mathcal{Y}$ in the single-step and multi-step estimation schemes.

The network model learns the dynamics of $\mathcal{Y}$ from $\mathcal{X}$ in t_{d}, which is representing the steady-state condition under the forced excitations. In single-step estimation, the output layer $\mathcal{Y}$ contains the target measurements at t_{d}. However, the multi-step estimator generates estimations starting from t_{d} for k steps forward. The applications of multi-step estimation can also be found in the reference study [61].

4.7. DNN training parameters

The proposed DNN model is implemented using the ML PyTorch.[3] library [68] The model computations are accelerated with the CUDA toolkit.[4] The parameters of the DNN are listed in Table 2. The size of the hidden layers is according to $\text{n}_{\text{t}_{\text{d}}}$. The training network comprises the linear layer (L), sigmoid layer (S), tangent hyperbolic layer (T), and ReLU layer (R).

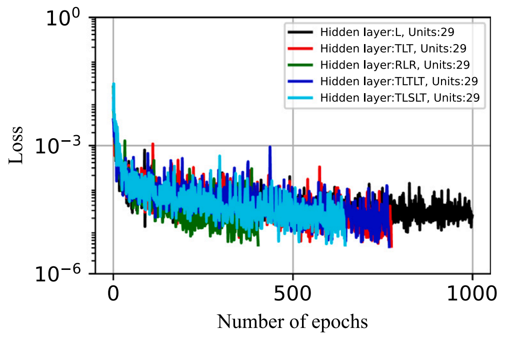
Table 2
Standard parameters from PyTorch are used for the ADAM optimizer.

Parameter	Value	Parameter	Value
Optimizer	ADAM [69]	Variable type	float32
Learning rate	1×10^{-3}	Batch size	$\text{n}_{\text{train}}/8$
Training set size n_{train}	80 ... 2560	Validation set size n_{val}	20% of n_{train}
Validation frequency	every 20 epochs	$\mathcal{L}_{\text{min}}$	5×10^{-6}

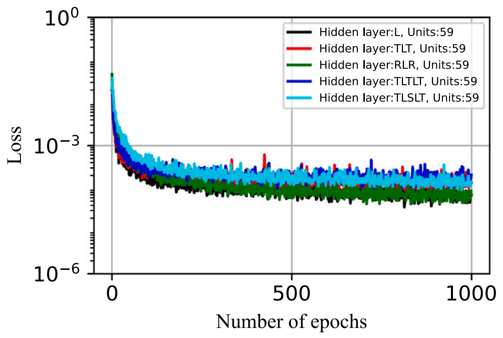
5. DNN training and its evaluation performance

The DNN model was trained on a computer with an 11th generation Intel® Core™ i5-11500H CPU running at a base speed of 2.92 GHz, complemented by 32 GB RAM. This machine has 6 operating cores and 12 logical processors. It also has an NVIDIA T1200 GPU. The operating system is 64-bit Windows 11. The Exudyn parameter variation function is used to facilitate supervised learning in various hidden layer combinations, leveraging parallel computing power. Following paragraphs describe the DNN training and evaluation performance in detail.

[3] Version 2.3, https://github.com/pytorch/pytorch
[4] Version 12.4, https://developer.nvidia.com/cuda-downloads



(a) Training loss with single-step estimation.



(b) Training loss with multi-steps estimation.

Fig. 10. Training performance—Mean squared error (MSE) loss for the single-step and multi-steps SLIDE-neural network models evaluated with five sensors and hidden layers combinations without payload.

5.1. *Training performance*

In supervised learning, various combinations of data size, input–output arrangement, and hidden layer size are explored. Fig. 10 shows the training performance of the single-step and multi-step SLIDE-networks, evaluated with five sensors and hidden layer combinations without payload. To ensure generalization and detect potential overfitting or underfitting, both training and validation losses were monitored during the training procedure. For each model, the training and validation losses remained comparable, indicating stable learning behavior. For clarity and brevity, however, Fig. 10 presents only the training loss curves.

The different hidden layer combinations include 'L', 'TLT', 'RLR', 'TLTLT' and 'TLSLT'. The five sensors providing data were —U, p_{h_1}, s, p_{h_2} and $\dot{s}$. SLIDE-network 'L' shows good training results, assisted by the availability of all data in $\mathcal{X}$ for the force computation. In single-step estimation, other hidden layer combinations meet the $\mathcal{L}_{\text{min}}$ criteria for five sensors combinations in the supervised learning. In Fig. 10(b), 30 steps forward in t_d was considered during the training process. This results in the smooth supervised learning of the SLIDE-networks. However, none of hidden layer combinations meet the $\mathcal{L}_{\text{min}}$ criteria. The relatively poor performance of multi-step SLIDE networks is due to the complexity of training multiple parameters during supervised learning.

Table 3
Sequential architectures—Training performance comparison of SLIDE with RNN, LSTM and CNN for 'TLSLT' hidden layers.

Neural networks	Minimum loss	Training time
RNN	1.8×10^{-4}	181.3 s
LSTM	2.4×10^{-4}	118.4 s
CNN	3.9×10^{-4}	51 s
SLIDE	5×10^{-6}	38.6 s

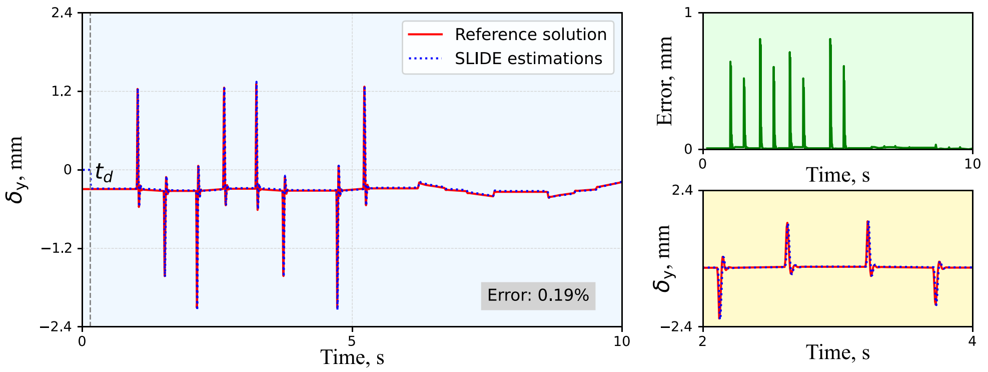
Table 3 demonstrates a comparison of SLIDE network with the sequential architectures RNN, LSTM and CNN for 'TLSLT' hidden layers. Standard parameters in Table 2 for ADAM optimizer had been used for this comparison. Among these, SLIDE-TLSLT network meets the $\mathcal{L}_{\text{min}}$ criteria in 718 epochs, also shown in Fig. 11(a). However, other sequential architectures could not reach to optimal training demonstrated overfitting during the supervised learning process. Further, the training time of SLIDE network is less than other networks.

5.2. *Evaluation performance*

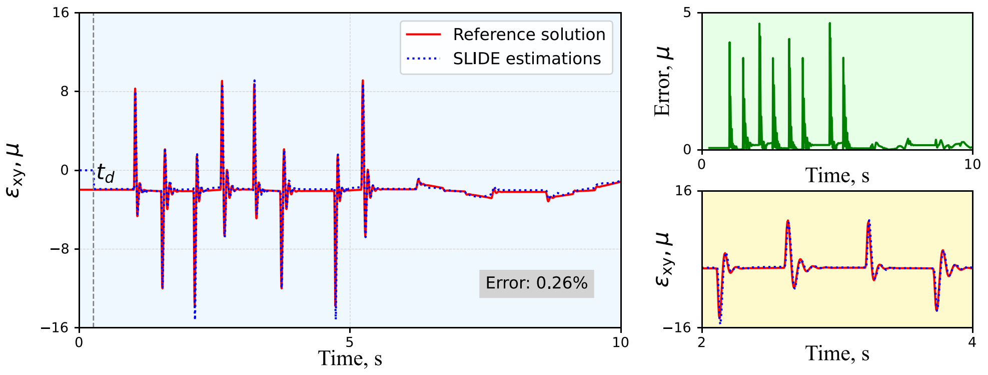
In evaluation phase, the performance of trained single-step SLIDE models is tested in estimating δ_y, ε_{xy} and σ_{xx} with an unseen control signal. Reference simulation are represented by (━━), and (- - - - -) corresponds to the SLIDE estimates. The SLIDE estimations are shown in Fig. 11 in reference to a 10 s dynamic simulation. The actuator control signal during simulation is presented in Fig. 16(b). Variations in the control signal also affect δ_y, ε_{xy} and σ_{xx} in the reference solution. The SLIDE estimations accurately capture the dynamic changes of control signal on the reference solutions.

For no payload, the SLIDE-TLT model makes no estimations of δ_y till 0.145 s, due to the unavailability of $\mathcal{X}$ for this period. After t_d, the model starts making single-step estimations with 0.19% mean-absolute-percentage-error (MAPE). The error with respect to the reference solution is further explained in mean-absolute-error (MAE) (━━━), which remains below 1 mm. In zoomed-in plot, the estimated δ_y is exactly following the patterns of reference solution during 2–4 s. With SLIDE-RLR, the estimation of ε_{xy} is described in Fig. 11(b) for 50 kg. It uses a sensor configuration of $U, s, \dot{s}$ and p_{h_2}. The network captured changes of ε_{xy} despite its small numerical values in the microstrain range.

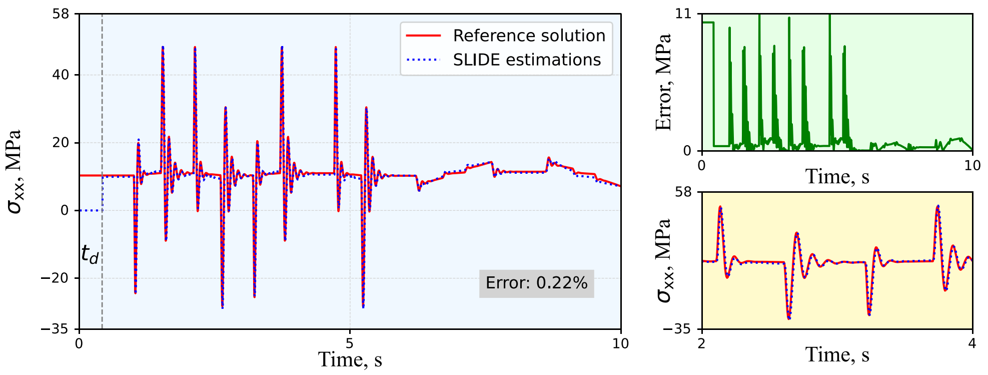
The estimation of σ_{xx} for payload 100 kg is described in Fig. 11(c). The maximum stress of approximately 58 MPa occurs in this case. A sensor combination of U, s, p_{h_1} and p_{h_2} enabled SLIDE-L to accurately estimate σ_{xx} in the hydraulically actuated flexible boom. It provided the estimation accuracy of 0.22% MAPE in the stress estimation.



(a) **No payload:** Estimating δ_y using 'TLT' network with the sensor configuration of $U, s, \dot{s}, p_{h_1}$ and p_{h_2}.



(b) **Payload 50 kg:** Estimating ε_{xy} using 'RLR' network with the sensor configuration of $U, s, \dot{s}$ and p_{h_2}.



(c) **Payload 100 kg:** Estimating σ_{xx} using 'L' network with the sensor configuration of U, s, p_{h_1} and p_{h_2}.

Fig. 11. Estimation accuracy—SLIDE-neural networks estimating δ_y, ε_{xy} and σ_{xx} in a hydraulically actuated flexible boom, verifying various payloads and sensor combinations. Mean-absolute-percentage error (MAPE) and mean-absolute error (MAE) demonstrate the estimation accuracy with respect to the reference solutions.

The zoomed-in plot reveals that the SLIDE estimations follows the patterns of ε_{xy} and σ_{xx} from the reference solutions between 2–4 s. This demonstrates the robustness of the SLIDE approach to estimating the targets over longer simulations. Note, in [62], flexible multibody estimation required a DNN, while SLIDE-L can achieve effective estimation performance for σ_{xx} with the sensor configurations of U, s, p_{h_1} and p_{h_2}. Further, this study demonstrates the numerical estimations of δ_y, ε_{xy} and σ_{xx}, whereas the estimations were presented on the finite element scale in [62].

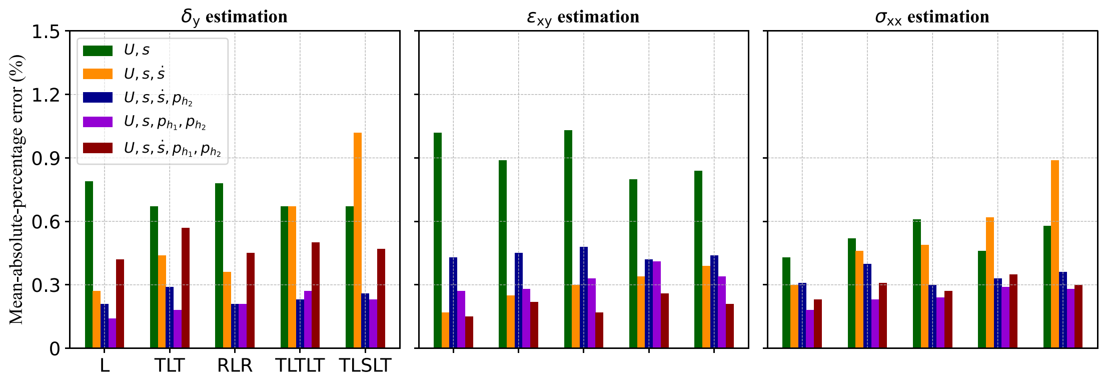


Fig. 12. Estimation summary – A summary of SLIDE-neural networks performance in single-step estimation of δ_y, ε_{xy} and σ_{xx} with different sensor configurations and lifting payload 100 kg.

The robustness of SLIDE networks is evaluated on a 2-DOF hydraulically actuated flexible forestry crane. Fig. 16(a) shows the forestry crane along with control signals used in the evaluation phase. The data acquisition from the crane is briefly explained in Appendix C. The single-step and multi-step estimations of SLIDE networks in the crane are demonstrated in Fig. 17. These estimations use control signals, actuator positions and hydraulic pressures in $\mathcal{X}$ during the training process. Fig. 16(a)–(i) demonstrate the accuracy of SLIDE networks in estimating δ_y, ε_{xy} and σ_{xx}. Further details about the SLIDE estimates in forestry crane example can be found in Appendix D.

Fig. 12 summarizes the performance of SLIDE neural networks in estimating δ_y, ε_{xy}, and σ_{xx} for a 100 kg payload with different sensor configurations. The maximum MAPE occurs when using 'TLSLT' with the sensor combination of U, s, and $\dot{s}$, and remains below 1.1%. The two-sensor case also provides accurate estimation results across all hidden layer combinations. This demonstrates that SLIDE neural network models can provide accurate estimations using actuator position and control signal data. Additionally, the use of pressure sensors further increases estimation accuracy, as seen with the sensor combinations $U, s, \dot{s}, p_{h_2}$ and U, s, p_{h_1}, p_{h_2}.

5.3. Computational efficiency

Python's function `timeit` was used to determine the relative run time of SLIDE-neural network estimation and simulation solutions. Average per-step computational time for the flexible boom problem is 4.98 ms in solving a 10 s Exudyn simulation, whereas the simulation step time is 5 ms . However, the single-step SLIDE-TLSLT takes 30 μs for computing per-step structural dynamics of the hydraulically actuated flexible boom. It demonstrates the computational superiority of SLIDE networks with respect to the hydraulically actuated flexible multibody simulations.

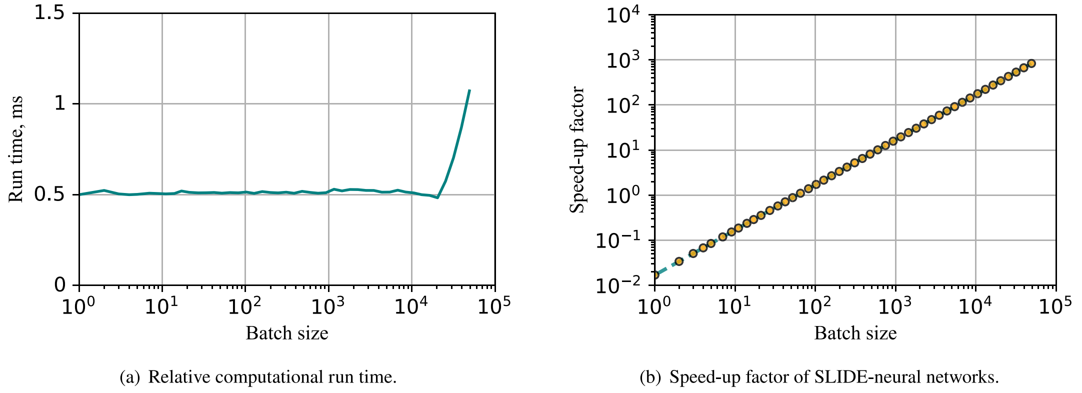


Fig. 13. Computational efficiency–Relative computational run time and speed-up factor of single-step SLIDE-neural networks compared to the flexible multibody simulation of case example in running simulation batches for δ_y.

Fig. 13 shows the computational efficiency benefits of SLIDE-neural networks compared to flexible multibody simulation for estimating structural dynamics of boom. CUDA takes 0.5 ms to initialize the system GPUs. As Fig. 13(a) also shows, the relative computational run time of the neural networks starts from 0.5 ms s for creating the equivalent data to the simulation batches. The relative computational run time of the neural networks when running simulation batches is faster than the simulations.

Note that the simulations solutions are faster in the Exudyn framework. The SLIDE approach will provide more computational efficiency as compared to FE [15] simulations. Further, Fig. 13 demonstrates the computational speed-up factor, S, of neural network training and run time, t_{NN}, with respect to the simulation solutions run time, t_{sim}. It is measured as [61]

$$S = \frac{t_{\mathrm{sim}}}{t_{\mathrm{NN}}} \frac{n_{\mathrm{out}}}{n_{\mathrm{in}}}, \tag{25}$$

where n_{in} and n_{out} are the input number of simulations and output number of steps. Note, in Fig. 13, the neural network training time of 20356 s for 10240 simulations has been considered for estimating the structural dynamics of flexible boom problem. The computational efficiency be further enhanced with the multi-step networks.

6. Conclusion

A physics-inspired real-time structural dynamics estimation framework using novel SLIDE-neural network model has been proposed. The framework learns the behavior of structural deflections at arbitrary location on the full-scale 3D FE model of a system within the SLIDE window size. For hydraulically actuated systems, a computationally efficient data acquisition approach has been proposed to generate data from random initial configurations, along with a method to determine $\mathrm{t_d}$. The structural dynamics of hydraulically actuated systems such as $\boldsymbol{\sigma}$, $\boldsymbol{\varepsilon}$ and $\boldsymbol{\delta}$ are estimated with less training data and standard PyTorch parameters from the Adam optimizer. The new framework is validated across varying geometries, 1-DOF and 2-DOF hydraulically actuated systems and sensor configurations. Effectiveness can be confirmed as follows.

- The data acquisition algorithm generated training data for the studied systems using randomized initial configurations, while also determining $\mathrm{t_d}$ without the EOMs. The $\mathrm{t_d}$ computing method introduced here can also be applied to the experimental data.
- The new framework is evaluated across different geometries and system configurations. The accuracy of SLIDE networks with various hidden layers, payloads and sensor combinations demonstrate their robustness. The trained networks accelerate the structural dynamics estimations by a factor of 10^3 compared to reference simulations for the hydraulically actuated 3D flexible multibody system batches.
- Effective supervised learning of SLIDE networks can be achieved with less data and standard parameters from PyTorch for the Adam optimizer compared to the standard RNN, LSTM and CNN architectures.

Unlike PINNs [48,49] and HNNs [58], which explicitly enforce physical constraints (e.g., energy conservation, symmetry, or compatibility conditions) during training, the SLIDE network utilizes physically known damping properties. The SLIDE networks learn the underlying physics of a system using $\mathrm{t_d}$, which can be computed using EOMs [61] and statistics-based method. SLIDE network settings work well for the simulation setup without such enforcements. However, the enforcements of such constraints might be required for the implementation of SLIDE networks in other applications. The findings highlight the potential industrial applications of SLIDE network in areas such as robotic manipulations, control, structural health monitoring, and automation.

6.1. Implications and future work

The effectiveness of proposed method is demonstrated through a simulation setup derived from a practical forestry crane [78]. To demonstrate the robustness, the structural dynamics of a 2-DOF forestry crane are estimated to validate the SLIDE performance across different structural geometries and system configurations. Training data is obtained from an Exudyn-based model that reflects the crane's dynamic behavior. Hydraulic parameters and geometric configuration align with an industrial machine [78]. However, the SLIDE method has yet not been applied to estimation tasks in an experimental setup due to the lack of measurement data.

Setting up the experimental setup and data acquisition according to the SLIDE method can be time-consuming. As an alternative, a proportion of training data can also be obtained from the simulation data. However, developing and calibrating accurate models of hydraulically actuated systems require additional time. This includes modeling and verification of hydraulic cylinder friction, valve leakage, joint friction, and contact forces. It also involves selecting appropriate boundary conditions and modes for the flexible structures. Nevertheless, in an experimental setup, the statistics-based method can be implemented to determine $\mathrm{t_d}$.

Application of SLIDE network in an experimental setup can be considered within the scope of a future study. The requirement of simulation and experimental data should be demonstrated in details. The modeling details of the hydraulically actuated system should be taken into account to calibrate the model according to experimental data. These can contribute to the uncertainty of robotic manipulation and control [10]. Furthermore, for large deformation estimation, the dynamics of lightweight structures could be controlled using the SLIDE approach in future work. This may require modeling the flexible structure using the Absolute Nodal Coordinate Formulation. As documented in the literature, these problem have been studied in control systems using approximated dynamic models [9,30], FE [15], and direct measurement methods [21]. Because of nonlinearity, the detailed FE models may not be applicable to industrial systems. However, the proposed approach offers an efficient tool to study flexible structures with complex geometries.

The SLIDE-driven approach would require less data and works with standard PyTorch parameters in real-time estimation tasks for industrial systems, allowing sensors with less frequency. It is able to capture the dynamics of the hydraulic system effectively. Due to the known damping properties of the system, the relevant sensor information is contained within the input window, making recurrency unnecessary for dynamic prediction. While the RNN also performs well in capturing the dynamics, the FNN's computational efficiency provides a significant advantage, making it a practical choice. The SLIDE-based ML approach could have potential applications across various fields including engineering, medicine, and business. It could be employed in scenarios where data exhibits damping, because this would require fewer target values. The data generation tool developed here offers a versatile solution for hydraulically actuated systems, effectively facilitating research in ML, control, and structural health monitoring across various FE structures.

CRediT authorship contribution statement

Qasim Khadim: Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Peter Manzl:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Formal analysis, Conceptualization. **Emil Kurvinen:** Writing – review & editing, Writing – original draft, Validation, Resources, Project administration, Investigation, Conceptualization. **Aki Mikkola:** Writing – review & editing, Writing – original draft, Resources, Project administration, Formal analysis, Conceptualization. **Grzegorz Orzechowski:** Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Johannes Gerstmayr:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

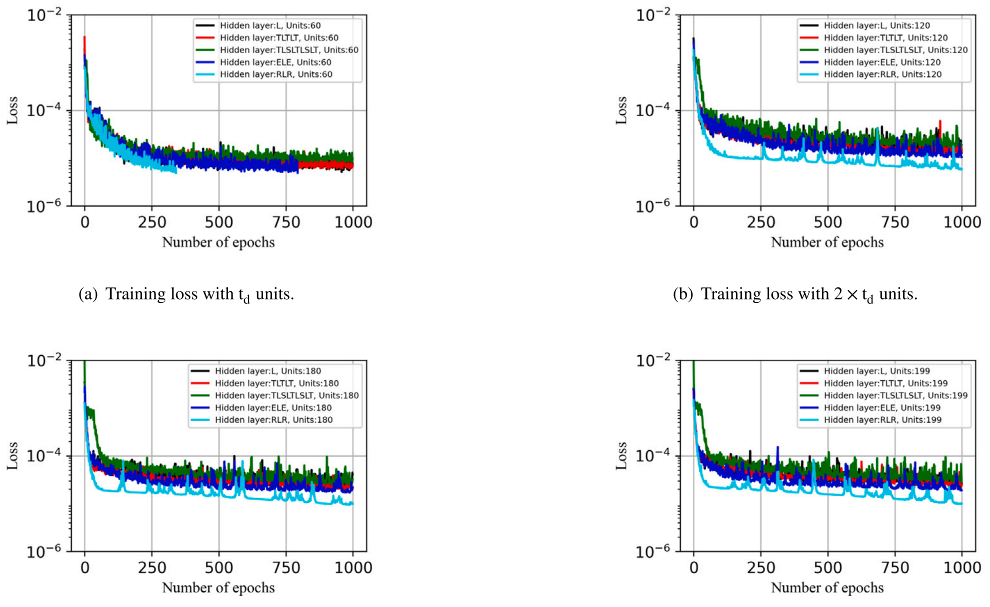
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Business Finland (SANTTU-Oulu 8896/31/2021 and SANTTU-LUT 8859/31/2021). We want to thank the open-source projects Exudyn [77], PyTorch [68], CUDA toolkit and Numpy [79]. The computational results presented here have been achieved in partly using the LEO HPC[5] infrastructure of the University of Innsbruck, as well as Puhti[6] and LUMI[7] provided by CSC. The authors would also like to acknowledge Dr. Scott Semken from the Laboratory of Machine Design, Department of Mechanical Engineering, LUT University for the language check for this article.

Appendix A. Training losses with different units

Fig. 14 demonstrates the superior training performance of single-step SLIDE-neural network models, evaluated with different number of units in the hidden layers.



(a) Training loss with t_d units.

(b) Training loss with $2 \times t_d$ units.

(c) Training loss with $3 \times t_d$ units.

(d) Training loss with 200 units.

Fig. 14. Training performance— Comparing the mean squared error (MSE) loss of single-step SLIDE-neural network models, evaluated with different number of units in the hidden layers.

[5] High Performance Computing resources at the University of Innsbruck, https://www.uibk.ac.at/en/mechatronik/mekt/research/research-profile/

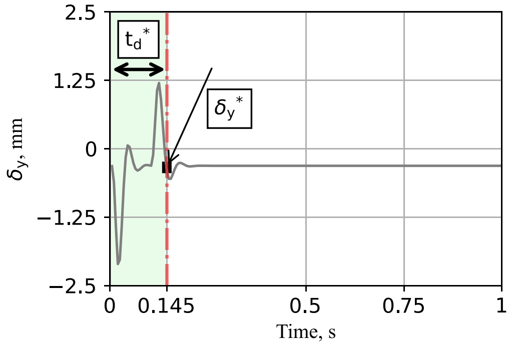
[6] Puhti supercomputer, CSC - IT Center for Science, Finland, https://www.csc.fi/en/puhti

[7] LUMI supercomputer, part of the EuroHPC Joint Undertaking and hosted by CSC - IT Center for Science, Finland, https://www.lumi-supercomputer.eu

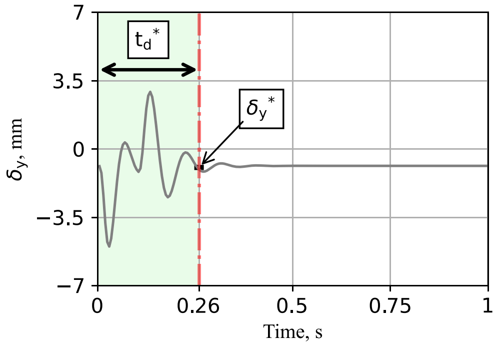
The neural networks require more data in the supervised learning with $2\times t_d$, $3\times t_d$ and 200 units in the hidden layer. As shown in Fig. 14(b)– Fig. 14(d), the supervised learning of neural networks demonstrate overfitting. However, the SLIDE-network arrangement in Fig. 14(a) shows stable supervised learning with relatively less training data and standard PyTorch parameters.

Appendix B. Computing SLIDE window

For no and 50 kg payload cases, the computing of $t_d{}^*$ is shown in Fig. 15 with the control signal, described in Sec. 4.4. Rayleigh damping $\mathbf{D} = 3.35 \times 10^{-3}$ Nsm^{-1} is employed in modeling the flexible boom, which effects the steady-state behavior of structural deflection under the forced excitation.



(a) The SLIDE window is $t_d{}^* = 0.145$ s with no payload.

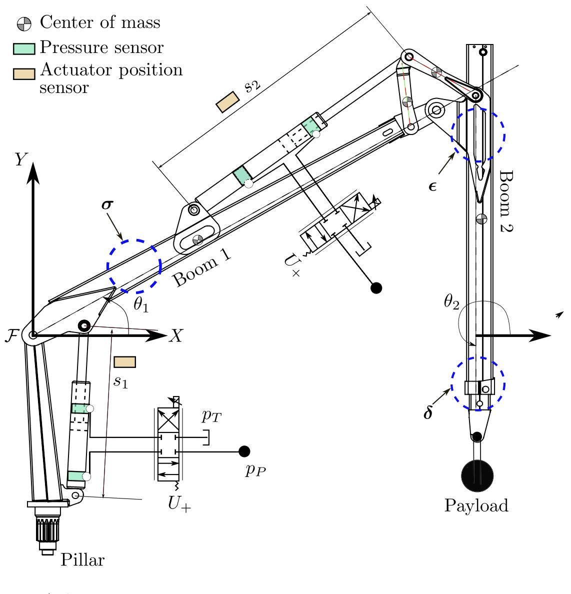


(b) The SLIDE window is $t_d{}^* = 0.255$ s with 50 kg.

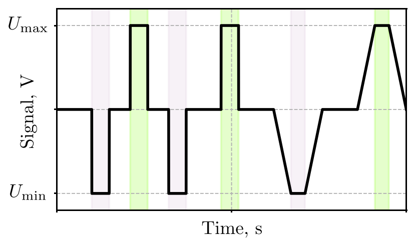
Fig. 15. Computing SLIDE window–$t_d{}^*$ is computed with no payload and 50 kg payload in the flexible boom.

Appendix C. Robustness of SLIDE networks

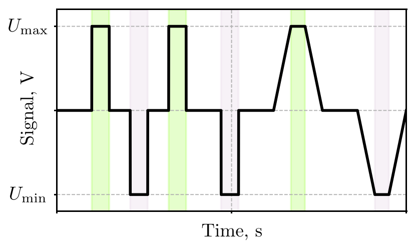
In the evaluation phase, the hydraulically actuated flexible boom is actuated with the control signal shown in Fig. 16(b). It ensures the movement of flexible boom up and down during the working cycle. This control signal was not provided during the supervised learning of SLIDE-neural network. To demonstrate the robustness, the structural dynamics of a 2-DOF forestry crane are estimated. This crane is shown in Fig. 16(a). It comprises two booms, pillar, two brackets, and two hydraulic cylinders [78]. The locations to estimate δ_y, ε_{xy} and σ_{xx} on the forestry crane are also highlighted.



a) A 2-DOF hydraulically actuated flexible forestry crane.



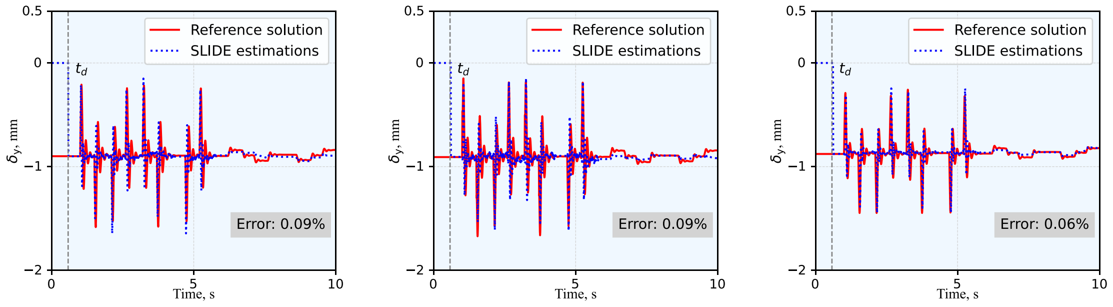
b) Lift control valve signal.



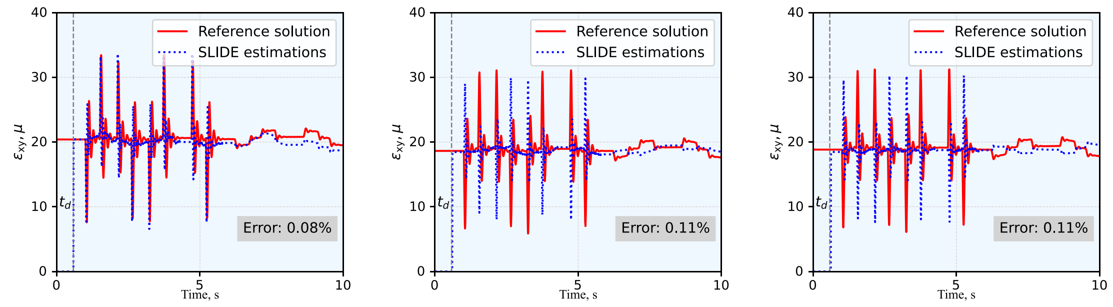
c) Tilt control valve signal.

Fig. 16. Application of SLIDE networks for estimating the structural dynamics of a 2-DOF forestry crane. Lift and tilt control valve signals illustrate the crane's working cycle during the evaluation phase.

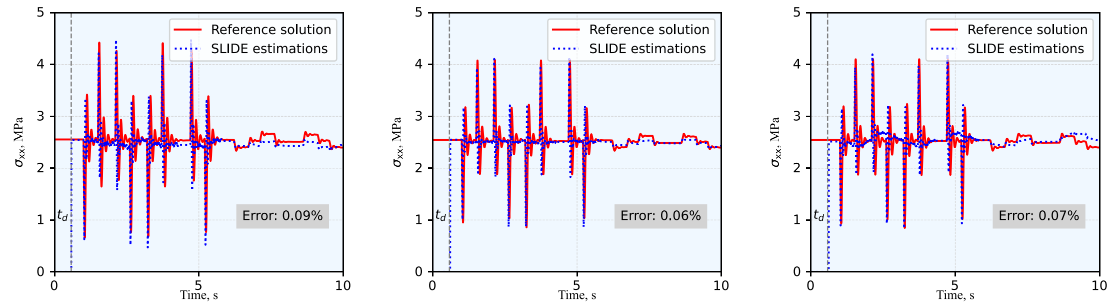
Both booms are modeled as the flexible bodies, and 16 modes are used in creating the training data. However, other bodies are considered as the rigid bodies. The hydraulic parameters are taken from the reference study [78], reflecting actual dynamic characteristics of machine. For the forestry crane, each simulation generates data for 400 steps in 2 s and the size of SLIDE window is 231 without payload. The training data is generated from 2560 simulations with randomized initial configurations. Accordingly, the hydraulic pressures in the cylinders are computed using the algorithm described in Fig. 3. The actuator strokes, pressures and control signals are used in $\mathcal{X}$ to train SLIDE networks, whereas $\mathcal{Y} = [\delta_y \quad \sigma_{xx} \quad \varepsilon_{xy}]$. Fig. 16(b) and Fig. 16(c) demonstrate lift and tilt control signals in the evaluation phase.



(a) Estimation of single-step SLIDE-TLT. (b) Estimation of 10-steps SLIDE-TLSLT. (c) Estimation of 15-steps SLIDE-TLSLT.



(d) Estimation of single-step SLIDE-TLT. (e) Estimation of 10-steps SLIDE-TLSLT. (f) Estimation of 15-steps SLIDE-TLSLT.



(g) Estimation of single-step SLIDE-TLT. (h) Estimation of 10-steps SLIDE-TLSLT. (i) Estimation of 15-steps SLIDE-TLSLT.

Fig. 17. Estimation in forestry crane—Summary of SLIDE-networks performance in single-step and multi-steps structural dynamics estimations in the forestry crane without payload.

Appendix D. Estimations in forestry crane

The estimates of δ_y, ε_{xy} and σ_{xx} by the SLIDE-networks for the forestry crane without payload are shown in Fig. 17. Standard PyTorch parameters, described in Table 2, have been used in the supervised learning of networks. These estimations are obtained by implementing the single-step, 10-steps and 15-steps SLIDE networks. For multi-steps, 10-steps and 15-steps have been estimated forward in t_d with SLIDE-TLSLT, whereas single-step estimations are made by SLIDE-TLT. All these estimates utilize control signals, actuator positions and pressures in $\mathcal{X}$ to estimate δ_y, ε_{xy} and σ_{xx}.

For δ_y, the SLIDE estimations closely follow the reference solutions in both single-step and multi-steps evaluations. However, ε_{xy} estimations show discrepancies, as shown in Figs. 17(d)–17(f). Specifically, Fig. 17(e) and Fig. 17(e) show noticeable differences compared to the reference solutions. These minor differences are attributed to overfitting during the training phase. Further accuracy of ε_{xy} estimations might require setting of hyperparameters in Table 2. On contrary, σ_{xx} estimations are relatively stable for single-step and multi-step evaluations. The maximum MAPE is 0.09% for σ_{xx} estimations in the forestry crane application.

Data availability

GitHub source code, 3D CAD models and meshes, datasets, trained neural network model, and evaluation results. Additionally, all result files are available on the GitHub repository.

References

[1] Kossi Dodzi Bissadu, Salleh Sonko, Gahangir Hossain, Society 5.0 enabled agriculture: Drivers, enabling technologies, architectures, opportunities, and challenges, Inf. Process. Agric. (2024).
[2] Qasim Khadim, Esa-Pekka Kaikko, Eero Puolatie, Aki Mikkola, Targeting the user experience in the development of mobile machinery using real-time multibody simulation, Adv. Mech. Eng. 12 (6) (2020) 1687814020923176.
[3] Aude Billard, Danica Kragic, Trends and challenges in robot manipulation, Science 364 (6446) (2019) eaat8414.
[4] Felipe Sánchez, Philipp Hartlieb, Innovation in the mining industry: Technological trends and a case study of the challenges of disruptive innovation, Min. Met. Explor. 37 (5) (2020) 1385–1399.
[5] Katja Mann, Lukas Püttmann, Benign effects of automation: New evidence from patent texts, Rev. Econ. Stat. 105 (3) (2023) 562–579.
[6] Emil Kurvinen, Antero Kutvonen, Juhani Ukko, Qasim Khadim, Yashar Shabbouei Hagh, et al., Physics-based digital twins merging with machines: Cases of mobile log crane and rotating machine, IEEE Access 10 (2022) 45962–45978.
[7] David A. Haggerty, Michael J. Banks, Ervin Kamenar, Alan B. Cao, Patrick C. Curtis, Igor Mezić, Elliot W. Hawkes, Control of soft robots with inertial dynamics, Sci. Robot. 8 (81) (2023) eadd6864.
[8] David Wagg, Simon Neild, Nonlinear vibration with control: for flexible and adaptive structures, Springer, 2010.
[9] Prasenjit Sarkhel, Mithilesh K. Dikshit, Vimal Kumar Pathak, Kuldeep K. Saxena, C. Prakash, Dharam Buddhi, Robust deflection control and analysis of a fishing rod-type flexible robotic manipulator for collaborative robotics, Robot. Auton. Syst. 159 (2023) 104293.
[10] Bai Li, Xinyuan Li, Hejia Gao, Fei-Yue Wang, Advances in flexible robotic manipulator systems—Part I: Overview and dynamics modeling methods, IEEE/ASME Trans. Mechatronics 29 (2) (2024) 1100–1110.
[11] Leilei Cui, Hesheng Wang, Weidong Chen, Trajectory planning of a spatial flexible manipulator for vibration suppression, Robot. Auton. Syst. 123 (2020) 103316.
[12] Antti Lajunen, Panu Sainio, Lasse Laurila, Jenni Pippuri-Mäkeläinen, Kari Tammi, Overview of powertrain electrification and future scenarios for non-road mobile machinery, Energies 11 (5) (2018) 1184.
[13] Frank Czerwinski, Current trends in automotive lightweighting strategies and materials, Materials 14 (21) (2021) 6631.
[14] Gerasimos Rigatos, Krishna Busawon, Robotic manipulators and vehicles: control, estimation and filtering, vol. 152, Springer, 2018.
[15] M. Uyar, Levent Malgaca, Implementation of active and passive vibration control of flexible smart composite manipulators with genetic algorithm, Arab. J. Sci. Eng. 48 (3) (2023) 3843–3862.
[16] Onur Avci, Osama Abdeljaber, Serkan Kiranyaz, Mohammed Hussein, Moncef Gabbouj, Daniel J. Inman, A review of vibration-based damage detection in civil structures: From traditional methods to machine learning and deep learning applications, Mech. Syst. Signal Process. 147 (2021) 107077.
[17] Arman Malekloo, Ekin Ozer, Mohammad AlHamaydeh, Mark Girolami, Machine learning and structural health monitoring overview with emerging technology and high-dimensional data source highlights, Struct. Heal. Monit. 21 (4) (2022) 1906–1955.
[18] Santosha Kumar Dwivedy, Peter Eberhard, Dynamic analysis of flexible manipulators, a literature review, Mech. Mach. Theory 41 (7) (2006) 749–777.
[19] Tian Soon Lee, Esmail Ali Alandoli, A critical review of modelling methods for flexible and rigid link manipulators, J. Braz. Soc. Mech. Sci. Eng. 42 (2020) 1–14.
[20] Fadi Aldakheel, Ramish Satari, Peter Wriggers, Feed-forward neural networks for failure mechanics problems, Appl. Sci. 11 (14) (2021) 6483.
[21] Konrad Johan Jensen, Morten Kjeld Ebbesen, Michael Rygaard Hansen, Online deflection compensation of a flexible hydraulic loader crane using neural networks and pressure feedback, Robotics 11 (2) (2022) 34.
[22] Asko Rouvinen, Heikki Handroos, Deflection compensation of a flexible hydraulic manipulator utilizing neural networks, Mechatronics 7 (4) (1997) 355–368.
[23] Jiazeng Shan, Peican Huang, Cheng Ning Loong, Mukun Liu, Rapid full-field deformation measurements of tall buildings using UAV videos and deep learning, Eng. Struct. 305 (2024) 117741.
[24] Hadi Pezeshki, Hojjat Adeli, Dimitrios Pavlou, Sudath C Siriwardane, State of the art in structural health monitoring of offshore and marine structures, in: Proceedings of the Institution of Civil Engineers-Maritime Engineering, vol. 176, (2) Thomas Telford Ltd, 2023, pp. 89–108.
[25] Ignasi Fernandez, Carlos G. Berrocal, Rasmus Rempling, Long-term performance of distributed optical fiber sensors embedded in reinforced concrete beams under sustained deflection and cyclic loading, Sensors 21 (19) (2021) 6338.
[26] Eike Grundkötter, Joachim Melbert, Precision blade deflection measurement system using wireless inertial sensor nodes, Wind. Energy 25 (3) (2022) 432–449.
[27] Patryk Kot, Magomed Muradov, Michaela Gkantou, George S Kamaris, Khalid Hashim, David Yeboah, Recent advancements in non-destructive testing techniques for structural health monitoring, Appl. Sci. 11 (6) (2021) 2750.
[28] Yuanchang Chen, D. Todd Griffith, Experimental and numerical full-field displacement and strain characterization of wind turbine blade using a 3D scanning laser Doppler vibrometer, Opt. Laser Technol. 158 (2023) 108869.
[29] Søren Rasmussen, Jørgen A. Krarup, Gregers Hildebrand, Non-contact deflection measurement at high speed, in: Bearing Capacity of Roads Volume 1, CRC Press, 2022, pp. 53–60.
[30] Changyin Sun, Hejia Gao, Wei He, Yao Yu, Fuzzy neural network control of a flexible robotic manipulator using assumed mode method, IEEE Trans. Neural Netw. Learn. Syst. 29 (11) (2018) 5214–5227.
[31] Mingming Shi, Bao Rong, Jing Liang, Wenlong Zhao, Hongtao Pan, Dynamics analysis and vibration suppression of a spatial rigid-flexible link manipulator based on transfer matrix method of multibody system, Nonlinear Dynam. 111 (2) (2023) 1139–1159.
[32] Kshetrimayum Lochan, Binoy Krishna Roy, B. Subudhi, Robust tip trajectory synchronisation between assumed modes modelled two-link flexible manipulators using second-order PID terminal SMC, Robot. Auton. Syst. 97 (2017) 108–124.
[33] Qingxin Meng, Xuzhi Lai, Ze Yan, Chun-Yi Su, Min Wu, Motion planning and adaptive neural tracking control of an uncertain two-link rigid–flexible manipulator with vibration amplitude constraint, IEEE Trans. Neural Netw. Learn. Syst. 33 (8) (2021) 3814–3828.
[34] Ahmed A. Shabana, Dynamics of Multibody Systems, Cambridge University Press, Cambridge, 2020.
[35] Andreas Zwölfer, Johannes Gerstmayr, Co-rotational formulations for 3D flexible multibody systems: a nodal-based approach, Contrib. Adv. Dyn. Contin. Mech. (2019) 243–263.
[36] Andreas Zwölfer, Johannes Gerstmayr, A concise nodal-based derivation of the floating frame of reference formulation for displacement-based solid finite elements: Avoiding inertia shape integrals, Multibody Syst. Dyn. 49 (3) (2020) 291–313.

[37] Andreas Zwölfer, Johannes Gerstmayr, The nodal-based floating frame of reference formulation with modal reduction: How to calculate the invariants without a lumped mass approximation, Acta Mech. 232 (2021) 835–851.
[38] M. Galal Rabie, Fluid Power Engineering, McGraw-Hill Education, New York, 2009.
[39] Prashant Kumar, Sechang Park, Yongli Zhang, Soo-Ho Jo, Heung Soo Kim, Taejin Kim, A review of hydraulic cylinder faults, diagnostics, and prognostics, Int. J. Precis. Eng. Manufacturing-Green Technol. 11 (5) (2024) 1637–1661.
[40] John Watton, Fluid Power systems: Modeling, Simulation, Analog, and Microcomputer Control, Prentice Hall, New York, 1989.
[41] Miguel A Naya, Cuadrado Javier, Daniel Dopico, Urbano Lugris, An efficient unified method for the combined simulation of multibody and hydraulic dynamics: Comparison with simplified and co-integration approaches, Arch. Mech. Eng. 58 (2) (2011) 223–243.
[42] Suraj Jaiswal, Jarkko Rahikainen, Qasim Khadim, Jussi Sopanen, Aki Mikkola, Comparing double-step and penalty-based semirecursive formulations for hydraulically actuated multibody systems in a monolithic approach, Multibody Syst. Dyn. 52 (2021) 169–191.
[43] Qasim Khadim, Emil Kurvinen, Aki Mikkola, Grzegorz Orzechowski, Simulation-driven universal surrogates of coupled mechanical systems: Real-time simulation of a forestry crane, J. Comput. Nonlinear Dyn. 19 (7) (2024) 071003.
[44] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 60 (6) (2012) 84–90.
[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller, Playing atari with deep reinforcement learning, 2013, ArXiv Preprint:1312.5602.
[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[48] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, George Em Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: A review, Acta Mech. Sin. 37 (12) (2021) 1727–1738.
[49] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, George Em Karniadakis, Physics-informed neural networks for heat transfer problems, J. Heat Transf. 143 (6) (2021) 060801.
[50] Arash Hashemi, Grzegorz Orzechowski, Aki Mikkola, John McPhee, Multibody dynamics and control using machine learning, Multibody Syst. Dyn. 58 (3) (2023) 397–431.
[51] Johannes Gerstmayr, Peter Manzl, Michael Pieber, Multibody Models Generated from Natural Language, Multibody Syst. Dyn. 62 (2) (2024) 249–271.
[52] Hee-Sun Choi, Junmo An, Seongji Han, Jin-Gyun Kim, Jae-Yoon Jung, Juhwan Choi, Grzegorz Orzechowski, Aki Mikkola, Jin Hwan Choi, Data-driven simulation for general-purpose multibody dynamics using deep neural networks, Multibody Syst. Dyn. 51 (2021) 419–454.
[53] Andrea Angeli, Wim Desmet, Frank Naets, Deep learning for model order reduction of multibody systems to minimal coordinates, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113517.
[54] Myeong-Seok Go, Young-Bae Kim, Jeong-Hoon Park, Jae Hyuk Lim, Jin-Gyun Kim, A rapidly trained DNN model for real-time flexible multibody dynamics simulations with a fixed-time increment, Eng. Comput. 40 (5) (2024) 3131–3156.
[55] Tomas Slimak, Andreas Zwölfer, Bojidar Todorov, Daniel Rixen, A machine learning approach to simulate flexible body dynamics, Multibody Syst. Dyn. (2025) 1–27.
[56] Seongji Han, Hee-Sun Choi, Juhwan Choi, Jin Hwan Choi, Jin-Gyun Kim, A DNN-based data-driven modeling employing coarse sample data for real-time flexible multibody dynamics simulations, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113480.
[57] Kurt Hornik, Maxwell Stinchcombe, Halbert White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[58] Fei Simiao, Huo Lin, Sun Zhixiao, Wang He, Lu Yuanjie, He Jile, Luo Qing, Su Qihang, Hamiltonian neural network 6-dof rigid-body dynamic modeling based on energy variation estimation, Complexity 2023 (1) (2023) 8882781.
[59] Michael Lutter, Jan Peters, Combining physics and deep learning to learn continuous-time dynamics models, Int. J. Robot. Res. 42 (3) (2023) 83–107.
[60] Olatunji Mumini Omisore, Lei Wang, Kinematics constraint modeling for flexible robots based on deep learning, in: 2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society, EMBC, IEEE, 2021, pp. 4940–4943.
[61] Peter Manzl, Alexander Humer, Qasim Khadim, Johannes Gerstmayr, SLIDE: A machine-learning based method for forced dynamic response estimation of multibody systems, 2024, ArXiv Preprint:2409.18272. (submitted for publication).
[62] Seongji Han, Hee-Sun Choi, Juhwan Choi, Jin Hwan Choi, Jin-Gyun Kim, A DNN-based data-driven modeling employing coarse sample data for real-time flexible multibody dynamics simulations, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113480.
[63] Hao Feng, Hao Zhou, Donghui Cao, Chenbo Yin, Chenxi Zhou, Data-driven hydraulic pressure prediction for typical excavators using a new deep learning SCSSA-LSTM method, Expert Syst. Appl. 275 (2025) 127078.
[64] Seongji Han, Grzegorz Orzechowski, Jin-Gyun Kim, Aki Mikkola, Data-driven friction force prediction model for hydraulic actuators using deep neural networks, Mech. Mach. Theory 192 (2024) 105545.
[65] Wei Ren, Wenbin Su, Xinrong Mu, Hongbo Wei, A flow rate soft sensor for hydraulic proportional directional valves, IEEE Sensors J. 24 (24) (2024) 42281–42288.
[66] Qasim Khadim, Emil Kurvinen, Aki Mikkola, Grzegorz Orzechowski, Efficient and accurate real-time simulations of coupled mechanical systems using the universal hydraulic surrogate, in: Proceedings of the ASME IDETC/CIE, 2023, number DETC2023-117402, page V010T10A030.
[67] Tomas Slimak, Andreas Zwölfer, Bojidar Todorov, Daniel J Rixen, Overview of design considerations for data-driven time-stepping schemes applied to nonlinear mechanical systems, J. Comput. Nonlinear Dyn. 19 (7) (2024).
[68] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, et al., Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation, in: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, 2024, pp. 929–947.
[69] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, ArXiv Preprint:1412.6980.
[70] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, Soham De, Resurrecting recurrent neural networks for long sequences, in: International Conference on Machine Learning, PMLR, 2023, pp. 26670–26698.
[71] Alex Graves, Alex Graves, Long short-term memory, Supervised Seq. Label. Recurr. Neural Netw. (2012) 37–45.
[72] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (2002) 2278–2324.
[73] James Bergstra, Yoshua Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2) (2012) 281–305.
[74] Johannes Gerstmayr, Stefan Holzinger, Andreas Zwölfer, From 3D solid finite elements to reduced flexible multibody bodies with constraint interfaces: A holistic approach, J. Sound Vib. (2025) (submitted for publication).
[75] Rolf Isermann, Mechatronic systems—Innovative products with embedded control, Control Eng. Pract. 16 (1) (2008) 14–29.
[76] Brian Armstrong-Helouvry, Control of machines with friction, vol. 128, Springer Science & Business Media, 1991.
[77] Johannes Gerstmayr, Exudyn– A C++ based Python package for flexible multibody systems, Multibody Syst. Dyn. 60 (4) (2023) 533–561.
[78] Qasim Khadim, Yashar Shabbouei Hagh, Dezhi Jiang, Lauri Pyrhönen, Suraj Jaiswal, Victor Zhidchenko, Xinxin Yu, Emil Kurvinen, Heikki Handroos, Aki Mikkola, Experimental investigation into the state estimation of a forestry crane using the unscented Kalman filter and a multiphysics model, Mech. Mach. Theory 189 (2023) 105405.
[79] Charles R. Harris, K. Jarrod Millman, Stéfan J. Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, et al., Array programming with numpy, Nature 585 (7825) (2020) 357–362.