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

Multibody Dynamics and Control using Machine Learning

Arash Hashemi · Grzegorz Orzechowski · Aki Mikkola · John McPhee

Received: date / Accepted: date

Abstract Artificial intelligence and mechanical engineering are two mature fields of science that intersect more and more often. Computer-aided mechanical analysis tools, often multibody system software, are very versatile and have revolutionalized many industries. However, as shown by the literature presented in this review, combining the advantages of multibody system dynamics and machine learning creates new and exciting possibilities. For example, the multibody method can assist machine learning by providing synthetic data, while machine learning can provide fast and accurate subsystem models. The intersection of both approaches results in surrogate and hybrid modeling techniques, advanced control algorithms, and optimal design applications. A notable example is the development of autonomous systems for vehicles, robots, and mobile machinery. In our review we have found nontrivial, innovative, and even surprising applications of machine learning and multibody dynamics. This review focuses on applying neural networks, mainly deep learning, in connection with the multibody system method. Over one hundred and fifty papers are covered, and three main research areas are identified and introduced: data-driven modeling, model-based control and estimation, and data-driven control. The paper starts with a primer on machine learning and concludes with future research directions. The main goal is to provide a comprehensive and up-to-date review of existing literature to inspire further research.

Keywords Machine learning · Multibody system · Deep learning · Data-driven modeling · Model-based control · Data-driven control

Arash Hashemi
University of Waterloo, Canada
Grzegorz Orzechowski
LUT University, Finland
Aki Mikkola
LUT University, Finland
John McPhee
University of Waterloo, Canada

Acronyms

AI Artificial Intelligence. 3, 4, 13, 14, 22, 32
ANN Artificial Neural Network. 5, 6, 15, 20, 23, 35

CNN Convolutional Neural Network. 5–7, 15, 16, 20, 29, 35

DeLaN Deep Lagrangian Network. 34
DL Deep Learning. 4, 7, 8, 10, 15, 22, 24, 29, 31, 32
DNN Deep Neural Network. 4, 11, 27, 30
DOF Degree of Freedom. 28, 33, 36
DRL Deep Reinforcement Learning. 12, 27, 28, 31

EKF Extended Kalman Filter. 25

FFNN Feed-Forward Neural Network. 5, 15, 17

GAN Generative Adversarial Network. 13, 15, 18, 30
GNS Graph Network-Based Simulator. 36

IL Imitation Learning. 29
IRL Inverse Reinforcement Learning. 29, 30, 33

KF Kalman Filter. 25

LQR Linear Quadratic Regulator. 23, 25, 26, 29, 31
LSTM Long Short-Term Memory. 8, 22, 35

MBRL Model-Based Reinforcement Learning. 23, 24, 33
MFRL Model-Free Reinforcement Learning. 27–29, 33
MHE Moving Horizon Estimator. 25, 26
ML Machine Learning. 4, 8–10, 14, 16–23, 25, 26, 31, 33, 35, 36
MLP Multilayer Perceptron. 5, 6, 22
MPC Model Predictive Control. 21, 23, 24, 26–28, 31, 33
MSD Multibody System Dynamics. 3, 13, 15, 27, 34–37

NARX Nonlinear Auto-Regressive with Exogenous Input. 25
NMPC Nonlinear Model Predictive Control. 26, 27, 31
NN Neural Network. 5, 9, 10, 13, 24, 25, 31–34, 36

PDE Partial Differential Equation. 34
PID Proportional Integral Derivative. 27, 31, 33

RBF Radial Basis Function. 32, 33
ReLU Rectified Linear Unit. 5, 21, 24
RL Reinforcement Learning. 11, 12, 23, 27, 28, 30, 31
RNN Recurrent Neural Network. 7, 8, 15, 18, 20, 32, 35

SMC Sliding Mode Controller. 26, 31, 33

1 Introduction

Rapid growth in Artificial Intelligence (AI) has radically altered a number of mechanical engineering applications. Autonomous vehicles, automated harbor cranes, robotic systems, and unmanned mining are examples of applications that have been significantly impacted. The development of sufficient artificial intelligence for autonomy is critically dependent on the capability of the machinery to describe its actions and to percept its environment. This pathway will lead to machines capable of autonomous operation (*e.g.*, performing tasks automatically, individually, or as a fleet) enabled by wireless machine communication. A critical step towards autonomously-operated systems relates to fully or partly automated actions, which can be accomplished by employing AI-based solutions. The aim of automated actions is to simplify the control and operation of the machines with semi-autonomous operator assistance systems. This can be used to enhance safe operation of a machine and to reduce cognitive stress on the operator and consequently improve an operator's well-being. Simplifying the control and operation also reduces stress on the machine, which will extend machine life, availability, and productivity.

AI can be connected to Multibody System Dynamics (MSD) such that a well-trained neural network can describe a sub-model such as friction. In this way, AI-based technologies in the form of hybrid models can take multibody simulations to another level in terms of computational efficiency and accuracy. This surrogate model can be based on practical experiments and/or detailed models. Surrogate models can also be used to replace an entire multibody model. This makes it possible to combine data from experiments with data obtained from multibody-based simulations. The combination of measured and simulated data could also evolve over a machine's lifetime as more is learned about its usage.

A critical bottleneck in the widespread implementation of AI in mechanical engineering and autonomy, in particular, relates to data production. Many AI-based technologies, such as deep and reinforcement learning, rely on large amounts of training data to function well. For example, more than ten million work cycles are needed to teach an excavator to operate autonomously for a single work cycle using a reinforced learning procedure [1]. Although in some cases, the data can be obtained from measurements, in general cases, it may be cumbersome to obtain [2]. The use of high-efficiency multibody system simulation can be connected to AI, and multibody system dynamics can offer data for AI training. To achieve this end, physics-based multibody simulation can produce data for AI needs accurately and efficiently. Multibody tools will then encourage more effective and more extensive AI implementations. As multibody system simulation can produce data more quickly than experiments on prototypes, it is possible to efficiently evaluate, tune and select an AI-based algorithm for each application.

This paper presents AI technologies in the framework of multibody system dynamics. In particular, the objective is to provide a comprehensive, up-to-date review of existing literature that explores the combination of AI and

multibody system dynamics. The paper also discusses the enormous potential of an AI and multibody system dynamics combination in terms of hybrid models. This combination has been investigated for more than a decade. Well over 100 journal articles covering this topic are discussed here. There is a myriad of AI, Machine Learning (ML), and Deep Learning (DL) methods applied to multibody dynamics and control. However, this paper considers only the application of shallow and deep neural networks to multibody system dynamics.

2 Primer on machine learning

Artificial intelligence, a term which was coined in 1956 [3], refers to any technique that enables a machine to mimic human behavior. The definition applies to any rule-based or data-driven approach. Machine learning, which is a subset of AI, includes using mathematics, statistics, and optimization rules to come up with a data-driven approach for the previous goal; in this sense, ML does not consider rule-based approaches. Lastly, deep learning utilizes a Deep Neural Network (DNN) to empower ML data-driven methods to solve more complex problems (Fig. 1). It should be noted that shallow neural networks also provide good solutions in particular problems and these classes of solutions are considered under the category of ML.

DL has recently gained the spotlight in the fields of science and engineering. With improved sensor technology and massive computer storage capabilities, large-scale acquisition of data has been made possible. This amount of data, coupled with advanced computing powers and recently developed complex algorithms, has been offering practical solutions to big real-world problems. The "learning" component in DL is the process of training the underlying algorithms to perform a desirable task.

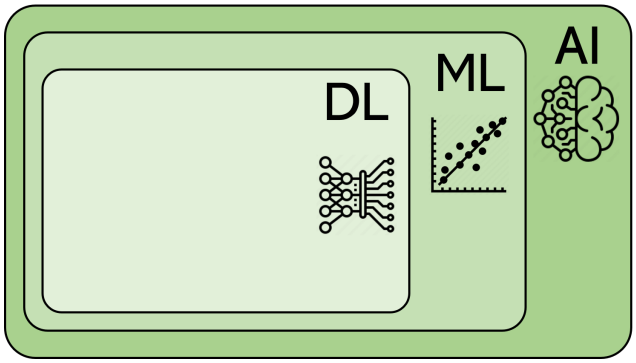

Fig. 1 AI vs ML vs DL

2.1 Different neural network structures

A Neural Network (NN) is a nonlinear function approximator formulated by a group of parameters $\boldsymbol{\theta}$. The optimal set of parameters are obtained by optimization algorithms. Researchers have come up with different structures of NNs for solving various problems.

Artificial neural networks

The structure of an Artificial Neural Network (ANN) is inspired by the human brain. It includes artificial neurons arranged as multiple layers; the ANN consists of three types of layers: the *input layer* (which includes the input of the network), the *output layer* (which includes the network output), and the *hidden layers* (which include the middle layers of the network between input and output). The input is fed through these layers and processed by multiple mathematical operations. The output of each layer acts as the input to the next layer; the operation continues until the final output, see Fig. 2. Networks with up to three hidden layers are often called *shallow* and the ones with more than three hidden layers are often called *deep* neural networks. Furthermore, the network with a feed-forward structure, *i.e.*, with no loops or recurrent connections, are called Feed-Forward Neural Network (FFNN). A vanilla NN, often called Multilayer Perceptron (MLP), has a feed-forward structure and is fully connected with the nodes in one layer being connected to all nodes in adjacent layers. The input of each layer goes through an initial linear filter, consisting of multiplication by a weight matrix and summation with a bias matrix; the weights and biases compose the overall parameters of the NN. The filter is followed by a nonlinear activation function acting on the result:

$$\mathbf{y}_i = \mathbf{f}\left(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i\right) \tag{1}$$

where $\mathbf{x}_i$ and $\mathbf{y}_i$ are the input and output vectors to the ith layer. In equation (1), $\mathbf{W}_i$ and $\mathbf{b}_i$ denote the weight matrix and the bias vector for the ith layer, respectively. Note that the ANN's input should always be arranged as a 1-dimensional (1D) vector. In equation (1), $\mathbf{f}$ is the nonlinear activation function; various functions including but not limited to the sigmoid, Rectified Linear Unit (ReLU) (zero output for negative input and a linear function for positive input), and tangent hyperbolic have been used in the deep learning community based on the specific purpose of the network [4].

Convolutional neural networks

Convolutional Neural Network (CNN) is one of most common NNs currently used in research and industry [5]. Firstly introduced by LeCun et. al [6], the main advantage of this structure is that the feature extraction is automatically done by the network, meaning that no human knowledge/supervision is required for the determination of the features. In addition, CNNs include

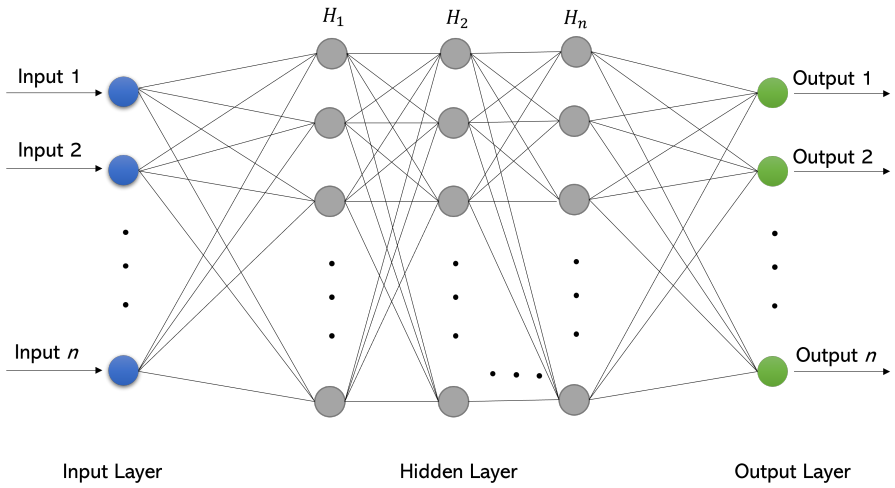

Fig. 2 The scheme of an ANN. The inputs go through the hidden layers where they are linearly transformed by weights and biases and nonlinearly transformed by the activation functions to generate the outputs. Note that all the nodes in hidden layers are fully connected to the inputs to that layer in a conventional ANN.

the *weight sharing* characteristic; multiple parts of the network have the same parameters. This important distinction from MLPs results in fewer overall parameters and simplified training, which makes CNNs applicable to larger datasets. The CNN has been applied in many fields including object detection and face recognition [7], [8], image classification [9], environment recognition [10], pose detection [11], and speech processing [12].

The structure of a CNN consists of convolution filters (kernels) with size $n_K \times n_K$. This filter convolves with a small part of a $n_I \times n_I$ input at a time:

$$S_I = \mathbf{W}_K * \mathbf{X}_I \tag{2}$$

where S_I is a scalar filtered part of the data after each convolution, $\mathbf{W}_K$ are the filter parameters, $\mathbf{X}_I$ is the part of the input on which the filter is acting, and $*$ denotes convolution operation. Then depending on the chosen stride (which determines the number of input elements that the filter passes after each move), the filter slides on the other parts of the input. Given the structure of the filter, the output after the filter has a smaller size than the input. This limits the use of many kernels in deeper networks as the input size turns to zero. Plus, the standard convolution implementation uses edge and corner data values less than middle values and the final network may lose performance. To solve these problems, a padding technique is applied before the filter, in which a zero layer is added to the edge of the data, hence preserving the size of the output. The last layers include multiple fully connected (FC) layers for the classification purpose. The scheme of a CNN can be seen in Fig. 3.

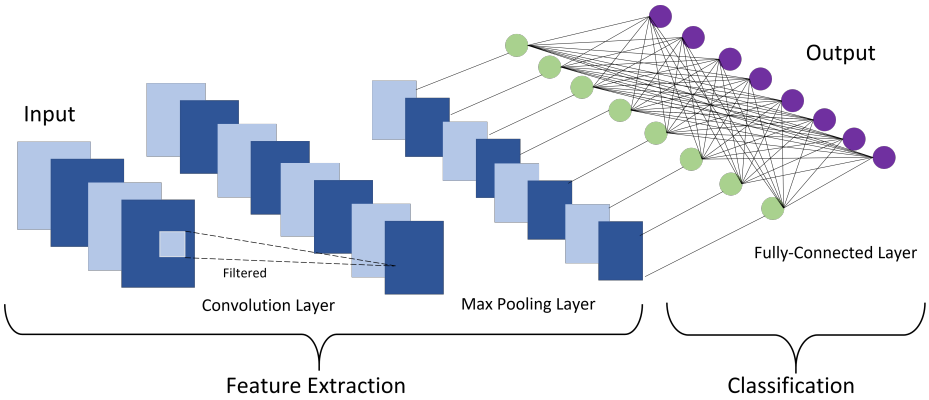

Fig. 3 The scheme of a CNN. The feature extraction layers are responsible for obtaining meaningful groups from raw input data; for instance, feature extractors in a face recognition CNN extract face components like eyes, nose, etc. The classification layers map the extracted features to the desired classes (in the face recognition example, the features are mapped to the corresponding individual). The input goes through multiple convolution layers (here, one is depicted) where a filter is swept through the input to generate a smaller data. Often, a pooling layer is also added to take the maximum (or average in case of average pooling) of neighboring data elements and make the data even more compact. The final filtered data is then fed through a fully-connected network to generate the output.

Recurrent neural networks

A Recurrent Neural Network (RNN) is tailored towards sequential data or time series. This networks is designed for applications in which the temporal characteristics of data is a necessary factor for the success of DL algorithms. Its unique format makes it applicable for speech segmentation [13] and natural language processing [14]. Also, due to its temporal aspect and hidden implementation of feedback in their network, RNN is a good candidate for modelling and control of dynamical systems [15], [16]. Unlike other structures, in RNN there is no assumption that the inputs and outputs are independent, meaning that each input does not only contribute to one output. In RNN, the outputs are divided into time-steps. The output of the time-step t is dependent on the inputs from previous time-steps $t-1$, $t-2$, $t-3$, etc (Fig. 4). While this structure facilitates the implementation of networks that can predict the next word in a sentence (or similar applications), it imposes a major problem, commonly known as vanishing/exploding gradients [17]. As the gradients backpropagate through the hidden layers (the gradient is calculated backward through the layers using the chain rule), depending on their initial values, they can get very large or small for the previous time-step weights. In the former case, the training becomes unstable. In the latter case, the weights of the previous time-steps are updated much slower than the later time-steps, making the training much slower for those sections of the network. As the previous time-step information is used in later time-step layers, the untrained parts of the network affect the training of the faster parts as well. Long Short-Term

Memory (LSTM) and gate recurrent units have been developed to solve this problem [18], [19], [20].

2.2 Optimization algorithms for ML/DL

To find the parameters of a neural network, namely weights and biases that determine the mapping between the input and output, an optimization problem is solved to minimize (or maximize in some cases) an objective function. Generally speaking, a logarithmic/cross-entropy cost function is used for classification problems and mean squared error or similar functions are utilized for regression problems. Depending on the convexity of the optimization problem, the solution might have one (global) or multiple (local) optima. Gradient-based methods [21] are one of the main approaches for solving optimization problems. Stochastic gradient descent [22] along with adaptive momentum optimizer [23] have been used extensively in the literature. While gradient-based methods are a common solution in the deep learning community, they are susceptible to converging to local optima and saddle points. In addition, the performance of these methods degrade considerably with noisy gradients, a common feature in dynamical system and controls applications. Streams of research have focused on combining evolutionary algorithms with gradient-based approaches for solving optimization problems, especially in reinforcement learning applications, semi-supervised data-driven methods, where the aforementioned problems have larger impact on the performance of the algorithm [24].

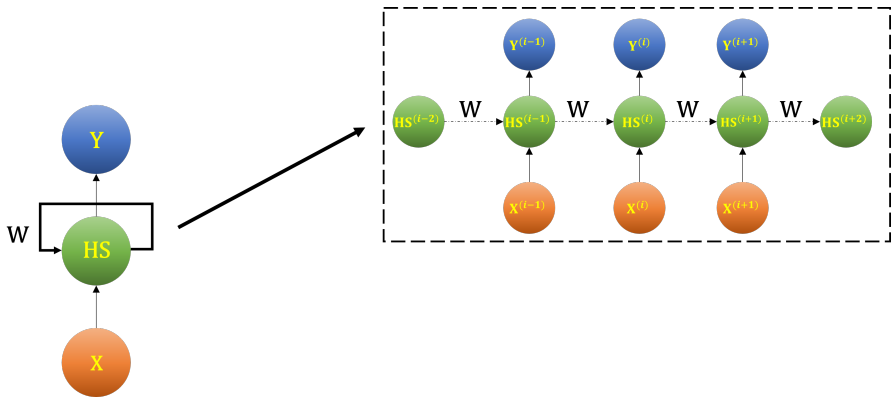

Fig. 4 The scheme of RNN. **X**, **HS**, **W**, **Y** refer to the input, hidden state, weight, and output vector, respectively. i denotes the time step at which the vectors are represented. When an input sequence is fed through the network, each input element is used to generate the output, while going through a hidden state. The hidden state from previous input elements are then fed into the next layers. This way, the inputs will have information about previous input/outputs.

2.3 Important issues in NN training

Overfitting

In the ML research, the data is split in three sections: the training data is used to train the network for a given task; the validation data is used to tune the network hyperparameters (number of layers, number of hidden units, etc.) for the highest performance; and finally, the test data is used to evaluate the performance of the network on unseen data. A successfully-trained network offers good accuracy on all data. Although a high training accuracy is valued on the training data, fitting a NN exactly to this sample data is not desired. Overfitting results in very good accuracy on the training data and poor performance on the test data. In other words, the model only memorizes the structure of the training data and cannot generalize well to new unseen data, which is ultimately the goal of using data-driven approaches. Overfitting stems from multiple factors; usually, if the model is trained too much or the algorithm is too complex for the sample data, the NN even learns the irrelevant structures on the data, like the noise, and hence loses it generalizability. Various methods exist for avoiding overfitting including but not limited to:

- Simplifying the network structure by reducing the number of layers or the number of hidden units.
- Stopping the training earlier whenever a certain training/testing accuracy is reached; also known as "early stopping".
- Adding a regularization term to the cost function to penalize high weight and bias values.
- Adding dropout layers, which randomly disable part of the network by setting the corresponding weights and biases to zero during training.

Underfitting

Underfitting has the opposite effect to overfitting and happens when the model has a poor performance on the training and test samples. It usually occurs when the model is not complex enough for the structure of the data or the model is not trained enough. As a result, adding more layers/units to the model and/or adding more training epochs are common solutions. Based on the previous section, there is an optimum complexity of the model and the training epochs (a measure for the number of times the training vectors are used once before updating the weights) for each problem. Using validation and test sets and trying different network settings is therefore of utmost importance. Moreover, a stream of research is focused on optimization-based hyperparameter tuning [25].

2.4 Learning schemes in ML/DL

Depending on the application, DL algorithms make use of three general learning schemes: *supervised learning*, *unsupervised learning*, and *semi-supervised learning*, which differ in terms of whether the true outputs are available and how they are used.

Supervised learning

Supervised learning is a suitable method when the true outputs (also called expert labels) are available for a set of inputs. An approximate mapping between input-output pairs is of interest to replace the experiments or computationally intractable models. To this end, NNs act as parameterized nonlinear function approximators and the mapping is found by acquiring a suitable set of parameters $\boldsymbol{\theta}$. Hence, supervised learning obtains the following approximate function:

$$\hat{\mathbf{y}} = \boldsymbol{\psi}(\mathbf{x}; \boldsymbol{\theta}) \tag{3}$$

where $\hat{\mathbf{y}}$ is the approximate output to replace the real output $\mathbf{y}$, $\boldsymbol{\psi}$ is the overall NN function, which depends on the input $\mathbf{x}$ and parameters $\boldsymbol{\theta}$. Broadly speaking, studies that use supervised learning can be *classification* or *regression* problems; the former predicts a discrete output, like whether an image is a cat or dog, and the latter predicts a continuous quantity, like the exoskeleton assistive torque for a given human pose.

Supervised learning is the dominant method used in the DL community. Many applications, like object detection, image segmentation, and natural language processing make use of this approach. Moreover, most of the DL research in multibody dynamics takes advantage of this method, since the true labels are usually available from physical models or experiments.

Unsupervised learning

In some applications, the labels are not available and only a set of input data exists. However, it is of interest to find/change the underlying structure of data, for example, to group the emails into spam and non-spam categories from the data or to reduce the dimension of the input data for faster training. Unsupervised learning algorithms are utilized for this purpose. Depending on the structure of data, different subcategories of unsupervised learning can be used. If the goal is to categorize the data into distinct groups, the problem is considered as a *clustering*. K-means, mean-shift, DBSCAN, mixture models, and hierarchical clustering are among the most common clustering algorithms [26]. Another category of unsupervised learning problems are called *embedding*, where the data has a continuous distribution. The linear versions of this approach lead to the quite well-known singular value decomposition and principal component analysis methods [27]. The nonlinear versions include AutoEncoder networks [28]. Embedding algorithms are a powerful tool for model and dimensionality reduction.

Semi-supervised learning

There are two categories to this method. Semi-supervised learning can be associated with the class of learning methods where the expert labels are not available prior to the training but are generated (approximated) during the training; a function approximator is responsible for generating/correcting the labels and another approximator is responsible for creating the correct output based on the approximate labels. The second category looks into the ways to improve accuracy where the amount of labeled data is substantially smaller than the unlabeled data. The two most well-known methods that are based on semi-supervised learning are *reinforcement learning* and *generative adversarial networks*.

Reinforcement learning

Reinforcement Learning (RL) is based on the first category of semi-supervised learning. In this method, the agent (the algorithm, controller, or the brain) interacts with an environment by generating an action and observing the environment's next state. Each agent interaction with the environment is evaluated; being in the state $\mathbf{S}_t$ and executing the action $\mathbf{A}_t$, where the subscript t is the time index, obtains an immediate reward from the environment (See Fig. 5). The goal of the RL algorithm is to maximize the cumulative reward, *i.e.*, the reward obtained starting from the initial state $\mathbf{S_{t_0}}$ and proceeding until the terminal state $\mathbf{S_{t_f}}$ is met. This is achieved by obtaining the value function $\mathbf{V}(\mathbf{S})$ for each state, which determines the quality of a given state. This value can be considered as the true label, which is usually not known before the training. As a result, it is estimated; the algorithm aims to make the estimations close to the true values. The value functions are updated at each iteration by the Bellman equation [29]:

$$\mathbf{V}(\mathbf{S}_t) = \mathbb{E}[R_{t+1} + \gamma \mathbf{V}(\mathbf{S}_{t+1}) \mid \mathbf{S}_t = \mathbf{S}] \tag{4}$$

where $\mathbf{V}(\mathbf{S}_t)$ is the value function for the state $\mathbf{S}_t$ at the iteration t; R_{t+1} is the immediate reward for being in the state and $\mathbf{V}(\mathbf{S}_{t+1})$ is the value of the successor state. γ is the discount factor used to emphasize on short-term rewards. This formulation can also be rewritten for state-action pairs instead of just the states.

Multiple function approximators have been utilized to estimate the value functions. For instance, linear functions have been selected as follows:

$$\mathbf{V(S)} \approx \hat{\mathbf{V}}(\mathbf{S}; \mathbf{W}) = \boldsymbol{\phi}(\mathbf{S})^T \mathbf{W} \tag{5}$$

In this representation, $\boldsymbol{\phi}(\mathbf{S})$ can be any nonlinear function of the states; however, the value function approximation with the parameters $\mathbf{W}$ is linear. A common RL method in control applications that uses linear approximation is least square policy iteration [30]. The specific use of DNNs as a nonlinear function approximation has led to the emergence of Deep Reinforcement

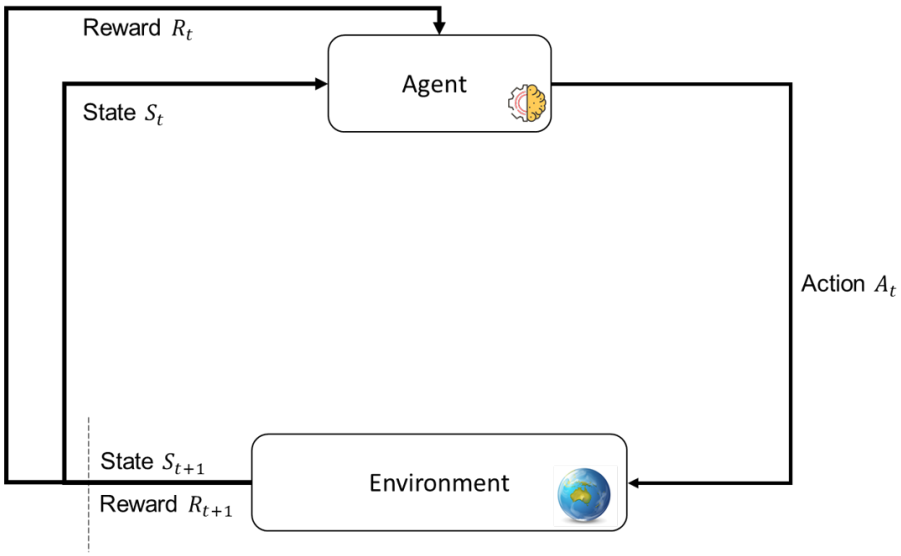

Fig. 5 The schematic of RL. The agent refers to the component that does the decision-making based on a policy, which is the algorithm that defines the mapping between the states and actions. Based on the current policy $\boldsymbol{\pi}_t$, the current state $\mathbf{S}_t$, and the current reward signal R_t, the agent calculates the current action $\mathbf{A}_t$ and applies it to the environment. It then receives the next state $\mathbf{S}_{t+1}$ and reward signal R_{t+1} for the next iteration.

Learning (DRL) research, which dominates the current studies in the field of computer science and engineering.

RL has recently gained the spotlight with the seminal work of David Silver, in which an agent beat a human master in the game of AlphaGo [31]. The use of deep neural networks as function approximators has been popularized in RL by the introduction of deep Q networks [32]. Recently, modifications to actor-critic methods have been proposed to work with continuous state-action environments; these approaches use an actor to approximate the policy (the controller), which is updated based on the value function approximate value coming from the critic. Deep deterministic policy gradient and twin-delayed policy gradient methods are among these approaches [33], [34]. This achievement has made RL algorithms directly applicable to continuous physical domains, like multibody dynamic systems, without the need for state-action discretization.

Generally speaking, RL can be categorized into model-free and model-based methods, both of which rely on interactions with the environment. Having said that, model-based methods use a model of the environment for prediction/planning, while model-free methods do not require a model representation of the environment and depend solely on environment interactions. Both variations will be discussed in more detail in future sections.

Generative adversarial networks

Developed by Ian Goodfellow [35], Generative Adversarial Network (GAN) is an interesting category of deep learning algorithms that is used for generative modeling. At its core, GAN is an unsupervised learning task with unlabeled data. There are no expert labels provided with the input data. Having said that, an updated version of GANs adds a small amount of labeled data and uses the second category of semi-supervised learning. The goal is to extract patterns of the input data and generate new data containing the same pattern; the first part fits well with the unsupervised learning scheme previously discussed. Having said that, this problem is posed as a supervised learning problem in GANs. There are two networks in any GAN, namely a generator network and a discriminator network; as the name suggests, the generator network is trained to produce new examples that are similar to the input data. The discriminator network is responsible for determining whether the output of the generator network is a real example (sampled from the input distribution), or a fake example. Drawing inspiration from game theory, these two networks are trained together as a zero-sum game in an adversarial manner, i.e. the generator network keeps improving on generating good fake examples and the discriminator network keeps improving on detecting fake examples. A well-trained GAN contains a discriminator network that labels fake outputs as real examples half the time erroneously. This means that the generator is capable of producing examples that are highly similar to the input data. GANs have shown potential in MSD community. They have been used, for example, to decrease the computation for data-driven inverse-kinematics and inverse-dynamics approximation [36].

3 AI-powered high-fidelity multibody modeling

This section examines studies in which data-based models are incorporated into multibody formulations to enhance computational efficiency and/or accuracy. Data required to create such models are generated synthetically or acquired by measurements. A data-driven model made with simulation-based, synthetic data trades intensive precomputations and a large memory footprint for the improved efficiency of the data-driven solution. Correspondingly, real-life measurement data allows for improved accuracy by model identification. A neural network based on a trained data set can provide accurate solutions for multibody applications. This makes it ideal for building surrogate representations of a submodel within the multibody framework, or for building entire multibody models. A global approximation procedure that covers points from the whole domain of interest can be used in the multibody framework. This chapter introduces studies where a multibody model or its submodel is replaced by surrogate models based on the NN representation.

Firstly, some overview works on data-driven modeling will be presented in Section 3.1. Next, papers devoted to black-box data-driven surrogate submod-

els are discussed in Section 3.2. Following, work dedicated to the data-driven representation of complete multibody system models in inverse and forward dynamic simulations is shown in Sections 3.3 and 3.4, respectively. Finally, Section 3.5 presents hybrid methods, combining the deep learning technique with a physics-based modeling approach. Table 1 reviews some paper examples of data-driven modeling of multibody systems.

3.1 Thematic reviews

ML has a long history in computational mechanics, and several domain-specific reviews are already available. As shown below, this ranges from personalized medicine [59], through aerospace applications [60], to autonomous driving [61].

Saxby *et al.* [59] have reviewed personal musculoskeletal modeling that employs the data-driven approach. Their study has targeted the personalized medicine domain – to match the individual and the biomechanical model. To this end, the investigation has taken the AI-based approach, which has resulted in fast, personalized computational models for a human neuromusculoskeletal system. In addition, the study has discussed types of computational models, a framework for customized model generation and operation, ML applications in model personalization, and implementation details.

Brunton *et al.* [60] have introduced a comprehensive literature review on data-driven approaches in aerospace applications. Their study has taught concept-based strategies for various production processes, including design and design optimization. They have highlighted multiple possibilities on how ML can enhance the solution of engineering optimization problems. Similarly, they have explained how ML can be applied in aerospace design, manufacturing, service, and the concept of digital twins. The paper offered several case studies.

Similarly, the paper by Hashemi *et al.* [61] has discussed the application of various artificial learning techniques. The study focuses on "black-box" modeling, model-based control (including deep reinforcement learning), and vision-based systems (including human pose recognition) in autonomous driving and biomechatronics.

3.2 Surrogate subsystem models in the multibody framework

This section presents papers that utilize black-box ML techniques to approximate multibody model subsystems. Condition monitoring studies were also included under the surrogate model category as they must internally represent the main characteristic of the monitored system. Studies have shown good accuracy and efficiency in their domains.

Ardeh *et al.* [37] and Han *et al.* [40] have reported that their surrogates trained on synthetic data offered high accuracy and efficiency compared to the full, nonlinear simulation models. Ardeh *et al.* [37] have built a surrogate of a tire using a feed-forward neural network. The surrogate model was designed

Table 1 Summary of papers on data-driven modeling of multibody systems using deep learning. The right-most columns refer to data types employed to train and verify data-driven models: synthetic, simulation-based data (SD), measurement-based data (MD), and a hybrid approach, where synthetic data are used for model training while measurement data are used in model verification (SM). In the methods column, FFNN, RNN, CNN, and DCNN refers to, respectively, feed-forward, recurrent, convolutional, and deconvolutional neural networks. SOUL is a self-organizing unsupervised learning method. GAN refers to the generative adversarial network, RAE is recurrent auto-encoder architecture, SVM is support vector machine, while NNDTW is nearest-neighbor dynamic time warping.

Paper	Topic	DL Methods	SD	MD	SM
Surrogate models					
Ardeh *et al.* [37]	Submodel of a tire	FFNN	✓		
Azzam *et al.* [38]	Virtual sensor for loads on wind turbine gearbox	FFNN	✓		
García Peyrano *et al.* [39]	Mechanical unbalance of the flexible rotor of a steam turbine	FFNN, SVM	✓		
Han *et al.* [40]	Submodel of a flexible component based on the floating frame of reference	FFNN	✓		
Kahr *et al.* [41]	Condition monitoring of roller bearings	CNN			✓
Ma *et al.* [42]	Contact/impact model between the barrel and bourrelet	FFNN		✓	
Sobie *et al.* [43]	Condition monitoring and failure detection of roller bearings	CNN, NNDTW			✓
Ye *et al.* [44]	Diagnosis of wheel out-of-roundness in high-speed trains	CNN, FFNN	✓		
Inverse dynamics					
Polydoros *et al.* [45]	Inverse dynamics of robotic manipulators	SOUL, RNN		✓	
Rane *et al.* [46]	Internal forces in musculoskeletal models during motion	CNN, FFNN		✓	
Ren and Ben-Tzvi [36]	Inverse kinematics and dynamics of robotic manipulators	GAN		✓	
Nasr *et al.* [47]	Inverse muscle dynamics in musculoskeletal models	RNN		✓	
Forward dynamics					
Byravan and Fox [48]	Approximate rigid body motion due to SE(3) transformation from the raw point cloud data	FFNN, CNN, DCNN			✓
Choi *et al.* [49]	A general solution to replace rigid MSD simulations with a data-driven approach	FFNN	✓		
Hegedüs *et al.* [50]	Road vehicle model for real-time trajectory planning	FFNN	✓		
Kraft *et al.* [51]	Railway vehicle acceleration approximation	RNN	✓		
Martin *et al.* [52]	Wheel-rail force approximation	RNN	✓		
Pan *et al.* [53]	Longitudinal dynamics of a vehicle	FFNN	✓		
Nasr *et al.* [54]	Muscle dynamics approximation	ANN, CNN, RNN, RCNN		✓	
Hybrid approaches					
Angeli *et al.* [55]	Reduction of multibody system's dimensionality from full to minimal coordinates	RAE	✓		
Hosking and McPhee [56]	Model and control of a powertrain of a hybrid vehicle	FFNN		✓	
Oishi and Yagawa [57]	Extract rules inherent in a computational mechanics application	FFNN	✓		
Ye *et al.* [58]	General MSD simulation without contact with railway applications	CNN, RNN, FFNN	✓		

for use in a quarter-car vehicle model. Han *et al.* [40] have proposed a deep learning approach to flexible multibody simulations. However, massive data is required to achieve high-fidelity models for deformable bodies, mainly due to fine model discretization in time and space. To solve the issue, the authors have proposed a two-stage learning procedure. In the first stage, the training was performed using small-size, randomly selected data. In the second stage, data-driven model performance was improved by training a second network designed to provide error corrections. The new approach was tested on several models. In addition, Han *et al.* [40] reports on the computational time of trained data-driven models in comparison with physics-based simulations. It is noted that the efficiency of the data-driven solution depends mostly on the number of prediction points (nodes) requested. All analyzed data-driven models have lower execution times than classical simulations. For the most complex model (excavator's arm) and prediction for 10 nodes, the data-driven model was 80 times faster than classical simulation on average.

Ma *et al.* [42] have introduced a deep learning model based on experimental data for the normal contact force. The proposed solution can approximate contact forces on complex surfaces and offers high accuracy and good generalization properties. The study, motivated by the lack of an appropriate physics-based model for such an application, has analyzed contact between a gun barrel and a bourrelet projectile.

Multibody simulations often generate synthetic data for training ML-based condition monitoring systems. Monitored systems include bearing fault detection (Sobie *et al.* [43], Kahr *et al.* [41]), wind turbine gearbox (Azzam *et al.* [38]), flexible rotors (García Peyrano *et al.* [39]), and diagnosis of wheel out-of-roundness in high-speed trains (Ye *et al.* [44]). The main reason to use synthetic data instead of measurements for model learning is the difficulty in data acquisition for various failure scenarios.

Sobie *et al.* [43] developed a detailed bearing simulation model to generate training data for a given physical realization. The study compares feature-based methods (such as random forests, shallow neural networks, and logistic regression) with deep learning convolutional neural networks and the nearest-neighbor classifier using dynamic time warping. The nearest-neighbor classifier is a supervised learning method that classifies data based on what class the known data points nearest to it belong to. The algorithm relies on distance for classification, and to consider the varying speed of the time series, a dynamic time warping similarity measure is employed. The trained models were verified using measurements of the speed faults. Simulation data makes it possible to build a classifier for any model and bearing configuration, giving good classification results. However, as the authors pointed out, experimental data should be included to improve classification accuracy. Moreover, the traditional feature-based data-driven models were outperformed by newly introduced models that lacked features.

Similarly, Karh *et al.* [41] have used a three-dimensional multibody model of the bearing to simulate various failure scenarios. The failure classification was based on CNN fed with an image of wavelet transform generated from the

data. The model was verified on measurement data. The trained network had easily distinguished between healthy and defective bearings, but the source of failure was not always appropriately identified.

Azzam *et al.* [38] created a virtual sensor to estimate all six components of loads on a wind turbine gearbox because an actual sensor is costly. All data for training and verifying neural network-based virtual sensors were generated in multibody simulation. Six networks were designed, one for each gearbox's load component.

García Peyrano *et al.* [39] have analyzed the mechanical imbalance of the flexible rotor of a steam turbine. The multibody model considers various imbalance conditions and bearing stiffness. In addition, two ML models were trained based on synthetic data: a feed-forward neural network and a support vector machine. Support vector machine is a robust supervised learning model for classification and regression analysis. Both methods have shown accurate predictions of the system imbalance.

Ye *et al.* [44] have studied the problem of the train's wheel out-of-roundness. Traditional out-of-roundness measurement is laborious and time-consuming. Therefore, the authors presented a concept of a ML-based online monitoring system of wheel out-of-roundness. Axle box acceleration is used as the input. Proposed network architectures include five neural networks, convolutional and fully connected, to process time and frequency domain signals. The solution has shown good accuracy in condition monitoring of wheel out-of-roundness.

3.3 Inverse dynamics applications

ML methods applied to inverse dynamics problems are mostly related to the modeling of robotic manipulators. The works of Polydoros *et al.* [45], Zhou and Schoellig [62], and Ren and Ben-Tzvi [36] fall into this category. Various ML techniques are used in this context.

The solution by Polydoros *et al.* [45] was based on a network architecture with a self-organized layer (which employs the Generalized Hebbian Learning algorithm – a learning rule for linear FFNN in unsupervised learning primarily used to approximate principal components of the input) and a recursive layer. Self-organizing unsupervised learning is a type of FFNN used to produce a discrete, low-dimensional representation of a higher-dimensional data set while preserving the relative distance between the points. Moreover, the Bayesian linear regression was applied to update the weights between the recursive layer and the outputs. Validation was performed on five measurement data sets from four robots. The proposed approach illustrated a state-of-the-art generalization ability and better adaptability than the existing real-time state-of-the-art learning solutions. Moreover, it robustly adapted to changes in the robot and its environment due to, *e.g.*, mechanical wear or moving objects.

Similarly, Zhou and Schoellig [62] introduced a framework for trajectory generation for a robotic manipulator to train deep learning inverse dynamics models. The authors have discussed the problem of insufficient coverage

of experimental data for data-driven model training. They have proposed an adaptive approach for data collection to train deep neural networks. The introduced procedure was based on a ML technique called active learning (not to be confused with the method of automatic label generation), closely related to optimal experimental design, which makes it possible for the learner to query the training data to increase information content actively. Their data-driven model's data efficiency and training quality were superior to that obtained using a trial-and-error methodology where the trajectories are selected manually.

Ren and Ben-Tzvi [36] recognized that data collection for data-driven inverse kinematics and dynamics models are the most time-consuming task in model development. Therefore, they have proposed an algorithm based on generative adversarial networks (consult Section 2.4) already widely applied in the computer vision community to overcome this concern. The authors used several GAN architectures to aid data creation in inverse kinematics and dynamics problems and performed experimental validation on two robotic manipulators.

Rane *et al.* [46] used deep learning techniques to compute internal forces during motion in musculoskeletal models based on kinematic inputs. In musculoskeletal modeling, data-driven models offer three main advantages: low computational cost (once the model is trained), recognition of the most significant model inputs (which may reduce the number of required measurements), and development of a data-driven model that makes it possible to assess relationships at the population level. The positions of the lower limb markers (18 in total) and ground reaction forces with center of pressure data from a force plate served as input for the network. Internal forces for the network labels were computed using existing modeling software. Moreover, the electromyographic signal of an electrical activity produced by skeletal muscles was also used as network output based on processed raw data. As a result, the final model has shown good accuracy and superior efficiency. However, the authors have emphasized that limited data decreases the performance of the data-driven model in scenarios where subject information was used only in the testing phase and not in training.

Similarly, Nasr *et al.* [47] used deep learning to approximate inverse muscle activation dynamics. In their black-box model called InverseMuscleNet, biomechanical kinematic (joint angle, joint velocity, joint acceleration) and dynamic (joint torque, activation torque) variables are fed into an RNN, which estimates the electromyographic signals. To decrease the number of inputs and find the most dominant ones, a sensitivity analysis on the input space is done using a backward selection algorithm. Moreover, the network configuration is optimized by trying different input sets and model weights. The resulting RNN can replace the inefficient static optimization that is often used for this purpose, and is computationally efficient enough for real-time applications like rehabilitation and sports engineering. Also, this model bypasses the need for time-consuming and invasive electromyographic electrode setup on subjects.

3.4 Forward dynamic applications

Studies in this section focus on time-varying responses of complete multibody systems without directly solving the equations of motion. The primary motivation behind those papers is increased efficiency compared to the original simulation model. The works of Urda *et al.* [63] and Byravan and Fox [48] build the system-level model from experimental data. Compared with surrogates in Section 3.2, those models represent the whole system.

The development of ML models has had a long history in railway applications. In 2007, Martin *et al.* [52] proposed substituting the multibody model with a neural network to approximate the forces on a wheel-rail interface based on track geometry and train characteristics. The study used a recurrent neural network and multibody simulation software to generate synthetic training data. Good agreement was observed between simulation results and neural network computations.

Similarly, Kraft *et al.* [51] studied the possibility of using black-box modeling in railway vehicle simulation in the presence of track irregularities. The authors evaluated several linear and nonlinear models against multibody simulation responses and measurements. While all models have reproduced vertical (primarily linear) motion, the highly nonlinear lateral movement required nonlinear modeling; the recurrent neural networks provided the most accurate results. Moreover, relative to the standard multibody simulation, computational times were shorter for the data-based model. The study emphasized that the performance of the data-driven model depends on the availability of training data.

Urda *et al.* [63] introduced a neural network-based estimation method for lateral wheel-rail contact forces. Their findings have shown that deep learning techniques are computationally efficient and easy to apply to the description of wheel-rail contact forces. They have compared two methods for lateral force estimation: the classical approach based on a harmonic elimination technique and the application of a deep neural network. The paper also offered an experimental validation of the data-driven model compared to simulation results. The authors concluded that the deep learning approach, compared with the classical harmonic elimination technique, is computationally more efficient and requires fewer sensor inputs.

Alternatively, Choi *et al.* [49] proposed a method to replace the multibody-based equations of motion of arbitrary systems using a deep neural network-based modeling technique. As for other findings, the main goal of their procedure was to improve computational efficiency. In addition, the authors have emphasized the smoothness of the solution at the displacement, velocity, and acceleration levels. Trained data-driven models are valid for the range of design variables and initial configurations of the system under consideration. The study has applied the procedure to three simple academic examples and one practical case.

An interesting perspective is introduced by Byravan and Fox [48], where rigid body motion was predicted based on point cloud data from a depth cam-

era. The motivation for this work was to observe physics rather than derive it from first principles. The study represents the kinematics of the rigid bodies as a SE(3) (Special Euclidean Group representing 3D rotations and translations) transformation. The authors have used a deep learning architecture with convolutional, fully connected, and deconvolutional layers (deconvolutional neural networks, often called transposed convolutional networks, use convolution to perform upsampling to find lost features) to predict how the environment changes due to applied forces. This results in higher accuracy and is more general than solutions obtained from data-driven models developed with no assumptions regarding the motion under investigation. The ML solution was verified with experimental data.

Hegedüs *et al.* [50] and Pan *et al.* [53] proposed a neural-network-based road vehicle model that can substitute a classic road vehicle model. In both studies, the data-driven solution was based on standard deep neural networks and showed good agreement with the full multibody model. Adequate efficiency and accuracy made the vehicle model suitable for real-time model-based motion planning and control. Hegedüs *et al.* [50] modeled a single-track vehicle and used ML to predict the overall behavior of the car, whereas Pan *et al.* [53] modeled a complete vehicle but predicted only fundamental quantities like the distance traveled and output velocity.

Nasr *et al.* [54] developed MuscleNet for fast approximation of forward muscle models. Taking electromyographic signals and delayed joint kinematics as input, MuscleNet estimates the current joint kinematics (angle, velocity, acceleration) and joint/muscle dynamics (net joint torque, joint activation torque). Multiple network structures including ANNs, CNNs, RNNs, and Recurrent CNNs are tested for this purpose. Two configurations of electromyographic inputs (raw, filtered) are also considered to evaluate the filtering capabilities of networks. RNNs with filtered electromyographic inputs showed the best performance among all network structures. MuscleNet represents muscle activation dynamics, muscle contraction dynamics, musculoskeletal geometry, and skeletal dynamics; it bypasses complex muscle modelling and model parameter identification and is computationally feasible for real-time application.

One of the main reasons for using data-driven models instead of classical simulations is a decrease in computational time. Therefore, it is worth pointing out that Choi *et al.* [49] and Hegedüs *et al.* [50] introduced computational time comparisons between physics-based simulations and data-driven models. Choi *et al.* [49] reports on computational time for their most complicated mechanical system (vibrating transmission with bearing contact) as it is not expected to note the computational advantage of the data-driven model for very simple mechanical systems. However, for the reported example, the data-driven model was over 110 times faster than the physics-based simulation. Hegedüs *et al.* [50] also reported that their data-driven models (of a planar, rigid, single-track vehicle) are faster than physics-based simulation. The computational advantage of the data-driven models depends on total simulation times. It ranges from around 3 times speedup for 1 s simulation to 1.3 speedup for 20 s.

3.5 Hybrid modeling techniques

Including expert knowledge of the system results in so-called gray-box or hybrid models. When available and applicable, specialist knowledge can enhance the model's properties and allow for easier ML model creation. Expert knowledge can be included as a regularization term (like in the work of Angeli *et al.* [55]), as a part of the model (as in Hosking and McPhee [56]), or can be exploited to design an appropriate structure for the ML model (as was done by Ye *et al.* [58] and Oishi and Yagawa [57]). This section presents a variety of applications and modeling designs.

Angeli *et al.* [55] proposed a deep learning architecture to reduce the dimensionality of the multibody system from full to minimal coordinates to enable the use of space observers. While the description of the system in minimal coordinates has many advantages, obtaining suitable representation is generally challenging and depends critically on the multibody formulation used. Therefore, the authors have proposed using a recurrent auto-encoder deep learning architecture to approximate the mapping between full system coordinates and a minimal set of coordinates. Recurrent auto-encoder implements an auto-encoder for sequential data using an encoder-decoder recurrent neural network. The encoder structure compresses (encodes) input into a fixed-length vector. The original sequence can then be reconstructed through the decoder structure. A critical characteristic of the proposed solution is that the learning process uses the multibody model directly. Therefore, it was not based only on data. Moreover, the authors selected sigmoid activation functions for their network instead of the more commonly used ReLU because the sigmoid functions provide smooth first and second-order derivatives. Their proposed solution was verified using two numerical examples. It showed good accuracy for the range of motion within the training data.

Hosking and McPhee [56] proposed a mixed approach to model the powertrain of a hybrid passenger car (Lincoln MKZ Hybrid). The car uses a complex power-split powertrain, which is challenging to model robustly using analytic modeling or experimental identification. Therefore, the authors have proposed combining analytical and experimental approaches. The drivetrain model was physics-based, while the control system and power source were approximated using neural networks, and parameters were obtained from measurements. The model was developed with limited knowledge of the system, and identification was made end-to-end based on vehicle road testing. The proposed solution proved to be an adequate approximation of the actual vehicle, and supported the development of model-based controllers for the autonomous vehicle.

Nasr *et al.* [64] used InverseMuscleNet [47] (see Section 3.3) as part of the high-fidelity model to which a Model Predictive Control (MPC) is applied. In this paper, MPC acts as the human central nervous system controller. The human model is coupled with an upper-limb exoskeleton, which uses a mid-level controller based on model-based and fuzzy logic techniques to adjust the exoskeleton assistance. InverseMuscleNet estimates the approximated electromyographic signals in the high-fidelity model.

Ye *et al.* [58] proposed a general deep learning model for multibody systems. Their solution comprises three parts: a 3D convolutional neural network, a recurrent Long Short-Term Memory (LSTM) network, and a standard MLP feed-forward, fully connected network. The design of their networks was physics-informed, that is, based on the fact that the system is described with second-order differential equations and that the resulting state depends on previous states and external interference. The resulting DL model was applied to a vehicle-track system in which the vehicle had 10 degrees of freedom, and no constraints were present in the system.

Oishi and Yagawa [57] proposed a general framework for extracting rules inherent to computational mechanics. The introduced procedure was based on deep learning techniques. While interesting and applicable to phenomena existing in multibody systems, the authors have demonstrated their method by developing a new numerical quadrature for a finite element stiffness matrix calculation. The number and placement of the quadrature points were optimized for each finite element in the model to achieve superior efficiency as compared with a classical approach.

4 Control-oriented DL models for model-based control and estimation

Model-based controllers have been extensively used in multibody dynamics research. These controllers utilize a model internally, usually referred to as the "control-oriented" model. The performance and stability of model-based controllers rely heavily on the accurate representation of the system dynamics. For complex multibody systems, obtaining an accurate physical model is not always feasible. Even in cases when a model can be acquired, it may not be utilized for real-time control due to computational complexity. Moreover, the inherent dynamics and model parameters may change over time, for example in different control scenarios. While obtaining a physics-based model requires doing extensive experiments to find the model parameters, for each scenario and for each system (when modeling vehicles for example, each vehicle portrays a different dynamic behavior which necessitates individual tests on each vehicle [65]), online estimation algorithms suffer from high computation for complex systems. Given that a great amount of data is available when these systems are operated by human experts, ML methods offer a viable solution to approximate the control-oriented model offline/online. These approaches are capable of extracting hidden patterns in the data and hence can capture higher-order and varying dynamics.

In this section, we will discuss how ML has been used to approximate the control-oriented model. Different controllers that have used AI-driven internal models will be reviewed. In addition, research on data-driven models for estimation will be studied.

4.1 Model-based reinforcement learning

Model-Based Reinforcement Learning (MBRL) is one of the main categories of RL. This method is dependent on the interactions with the environment but also uses an internal model to predict state trajectories (often called "roll-outs") and plan future action sequences [66]. A powerful method to facilitate this prediction and planning is to use receding horizon control; the optimal sequence of future actions is calculated in a defined horizon window but only the first action is applied to the environment, after which the same procedure is continued for future horizons. This is achieved by solving the following optimization for each prediction horizon:

$$\max_{[\mathbf{A}_{t_0}, \mathbf{A}_{t_1}, \mathbf{A}_{t_2}, ..., \mathbf{A}_{t_f}]} \quad \sum_{t=t_0}^{t_f} r(\mathbf{S}_t, \mathbf{A}_t) \tag{6}$$

where $[\mathbf{A}_{t_0}, \mathbf{A}_{t_1}, \mathbf{A}_{t_2}, ..., \mathbf{A}_{t_f}]$ is the action sequence in the horizon window and $r(\mathbf{S}_t, \mathbf{A}_t)$ is the reward function. This optimization can be solved using derivative-free and sample-based approaches like random shooting [67] and cross entropy method [68]; it can also be tackled using ideas from optimal control and trajectory optimization [69], where the first and second order derivative of the reward function and the internal model are utilized [70], [71]. This idea is the main structure used in Linear Quadratic Regulator (LQR) [72], iterative LQR [73], and MPC [74]. While MPC is sometimes interchangeably used with MBRL, it is not the only method for applying model-based planning. It should be noted that the aforementioned methods can be, and have been, integrated. More information on the MBRL can be found in the review papers [70], [71], and [75].

As discussed, a crucial component of MBRL is the use of an internal model; this model can be known a priori using physical rules. It can also be learned using ML, where a surrogate model is used for planning (similar to the discussions in Section 3.2). In addition, only part of the model can be approximated, like an aerodynamic drag coefficient or a static friction term (refer to Section 3.5). In this section, we discuss the MBRL research in the field of multibody dynamics.

To develop a variable impedance controller for efficient human-robot interaction, Roveda *et al.* [76] used a neural network approach to approximate the human-robot dynamics. To avoid overfitting in low-data regimes and capture the uncertainty in the data, an ensemble of ANNs has been utilized to learn the model; the ANNs will produce different outputs in data regimes where there is uncertainty. The model is trained offline based on previously-gathered data but is also updated online to account for human motor adaptation. The resulting human-robot interaction model is utilized within an MPC optimized by cross entropy method to dynamically obtain the stiffness and damping parameters of the impedance controller by minimizing the human effort.

Model-free approaches, which we review in Section 5.2, do not require a mathematical representation of the system dynamics. While these approaches

can obtain the optimal policy based on data patterns, they are highly sample-inefficient. On the other hand, MBRL uses model information, rendering better sample-efficiency but less accurate controllers. To combine the capabilities of these two methods, Nagabandi *et al.* [77] used MBRL with model-free fine-tuning. First, a deep neural network-based surrogate model is trained to approximate the dynamics. Then an MBRL algorithm is utilized to obtain an acceptable policy. This policy is then used as the initial solution of a model-free learner to compute the near-optimal solution. The integrated method achieved high tracking accuracy and sample-efficiency on multiple complex locomotion tasks of the openAI gym environment [78].

MPC has been utilized for real-time aggressive driving maneuvers in [79]. The model predictive path integral approach, which is usually applied with a control-affine assumption, was combined with general nonlinear dynamics with the aid of information-theoretic control. As this approach requires a model of the system, a multi-layered NN is used to approximate the dynamics. While the NN-approximated multibody dynamics is initially trained using random policies, it is later re-trained by the data gathered from applying model predictive path integral to the current dynamic model. The process is repeated until no further improvement is observed in the model accuracy. The resulting model-based controller achieves good empirical tracking for a complex mobile robot driving task.

4.2 Other model-based controllers

Aside from learning the control-oriented models (surrogate models) for MBRL (MPC in particular), a limited stream of research has focused on other controllers. Richards *et al.* [80] have used a feed-forward neural network in a meta-learning scheme to approximate an integrated model and adaptive controller. The approximate multibody model of a planar fully actuated rotorcraft includes the uncertain dynamic terms; it also comprises the aerodynamic disturbance. The learning is done on past data in an offline manner. The meta-learning (or the colloquially used term "learning how to learn") tracks several trajectories with desirable performance. Each trajectory is assigned to a base-learner, which learns the specific task. The meta-learner is then defined to minimize the average trajectory error in all tasks.

In [81], the model of a helicopter was obtained using a ReLU network, which combines a quadratic lag model (mapping the current state and multiple past inputs to the current helicopter linear/angular acceleration) and a two-layer neural network with ReLU activation functions. This multibody model has been successful in capturing the nonlinear dynamic terms, aerodynamic terms, engine model, vibration, and external disturbances. State-action pairs from expert demonstrations were used for training the network. Contrary to simple linear models, the NN-based model is able to handle complex aerobatic maneuvers. The DL-based model has enabled the use of model-based controllers.

A simple two-layer fully-connected NN was used to approximate the translational and rotational components of a quadrotor dynamics [82]. To collect the training data, the quadrotor moved along multiple trajectories using an LQR with a low-level proportional-derivative controller. The LQR controller uses a near-hover linear approximate model to derive the feedback control law. The trained NN model was utilized within a sequential convex optimization to calculate the open-loop control signal. The authors have shown that the combination of the feedforward controller (with an NN-based control-oriented model) with the feedback LQR-proportional-derivative controller works better than only the feedback term on the test trajectories (unseen in training). Moreover, the simple structure of NN enables the high online computation of sequential convex optimization.

Researchers have implemented a Nonlinear Auto-Regressive with Exogenous Input (NARX) network to approximate nonlinear vehicle dynamics [83]. Instead of using an internal feedback signal, as in recurrent neural networks, NARX feeds the delayed control inputs and outputs as extra inputs to a classic multi-layer perceptron. They have studied the effect of initial weights and the optimum network size on the network accuracy. Both offline and online training schemes are tested and it is concluded that it is possible, with the addition of a greater online computational cost, to use NNs for online system identification and control of complex vehicle systems.

The same problem was studied in [65] but for changing vehicle dynamics at the limit of multiple friction surfaces; these conditions affect the vehicle stability and controllability and must be captured in the model to ensure good controller performance in all driving scenarios. A two-layer NN was trained to approximate vehicle dynamics. Exploiting the data from both low- and high-friction driving scenarios, this model was able to perform well on different road surfaces and bypass the need for explicit friction estimation. The NN-based model was utilized within a feedforward-feedback controller, rendering better control results than with a physics-based alternative.

4.3 Model-based estimators

A subset of estimation techniques requires a model of the system. Kalman Filter (KF) and Extended Kalman Filter (EKF) are among these approaches. ML approaches are then utilized when obtaining a model is not possible or convenient. The work of Angeli *et al.* [55] on obtaining minimal coordinates using AutoEncoders was presented in Section 3.5. The minimal ordinary differential equations obtained from this method were used in an EKF to estimate the states of a slider-crank mechanism [84].

KFs are limited to linear systems and assume Gaussian distributions for the noise, disturbance, and state estimates. Also EKFs, the nonlinear version of KFs, require iterative local linearization of the plant around the new mean and covariance estimates. To overcome these issues, researchers have been developing the Moving Horizon Estimator (MHE) [85], [86]. While KFs are the

reformulation of LQRs, MHEs are the reformulation of MPCs; using the past states in a finite horizon window, MHEs estimate the current state by solving an optimization problem. Similar to MPC, MHE enables the use of a nonlinear plant model and explicit inclusion of constraints. To approximate the control-oriented model in the MHE estimator, Sun *et al.* [87] used a type-2 Fuzzy neural network. In this research, MHE estimates the external force/torque in a bilateral telerobotic system. The output of the estimator is used in a Sliding Mode Controller (SMC) for nonlinear control of the plant. Type-2 Fuzzy neural network enables the application of model-based estimators on a complex robotic system without the need to derive an accurate mathematical model.

5 Data-driven control

An alternative to using model-based controllers with learned models (as mentioned in Section 3), is directly using model-free data-driven control approaches that do not explicitly require a dynamics model for design. This method is applicable to situations in which learning a model requires an extensive and challenging data acquisition.

5.1 ML-based MPC Optimization

In Section 4.1, we discussed how ML methods can be used to train a control-oriented model for MPC implementation. There exists another challenging problem in real-time application of this controller; the online optimization of the MPC may be computationally intensive and hence solving it on embedded systems and micro-controllers may not be feasible, especially for high-state dynamic systems. Some of the derivative-free and sample-based methods for MPC optimization are mentioned in Section 4.1. Here, however, we discuss the research on approximating the optimization solution specifically using neural networks.

Inspired by explicit MPC [88], where the feedback policy is obtained offline by solving the optimization problem for various ranges of operation, neural networks have been used to train a controller that mimics an MPC controller based on synthetic data generated from an offline MPC. Winqvist *et al.* [89] presented a general framework for training and validation of neural networks for MPC. They proposed using PyTorch with the CVXPY optimization tool [90] for constrained explicit MPC. They also offered a hit-and-run sampler to systematically generate the input data for the network. Lastly, the authors explore the idea of adding domain knowledge to the network training.

In [91], Varshney *et al.* presented DeepControl: a two-layer fully connected neural network that replaces a Nonlinear Model Predictive Control (NMPC) for online quadrotor control. The training synthetic data is generated from a simulated model of the quadrotor using Gazebo. This model is obtained

by matching the simulation transfer function with that of the real physical system for some manual movements of the quadrotor. Offline MPC is used in simulation as the high-level controller to generate the corresponding roll, pitch, yaw rate, and thrust commands for following a given desired trajectory. A low-level Proportional Integral Derivative (PID) is then applied to obtain motor speed commands. The DNN controller has a close performance to NMPC for three different desired trajectories; on the other hand, it has a much lower online computation compared to NMPC. As a result, the DNN controller was applied in real-time, which led to less power consumption for computations and more flight time, compared to NMPC implementation.

Data-driven NMPC has been applied to other dynamic systems and the same ideas can be implemented in MSD. Pan *et al.* [92] iteratively linearized the nonlinear system model and changed the optimization problem from non-convex nonlinear to convex quadratic programming; they then used a recurrent neural network to approximate the solution to the new optimization. This controller was applied to a polymerization reaction problem. Cao *et al.* [93] presented the idea of "optimize and train" for NMPC approximation with deep feed-forward networks. In this approach, the process of generating the training data by offline NMPC and training the network is done simultaneously. As they discussed, the conventional "optimize then train" methods tend to lead to divergent (or inaccurate) networks when there exists multiple solution for a given initial state. The data-driven controller is applied to a quadtank system for numerical case study.

5.2 Model-free reinforcement learning

Model-Free Reinforcement Learning (MFRL) has been recently studied for the control of multibody systems. As mentioned in Section 2.4, MFRL finds the optimal policy based on the interaction with the environment and does not require an internal model. Therefore, if the environment reflects the changing dynamics, the DRL controller can generalize to multiple situations.

To review, RL formulates the problem as a Markov Decision Process (MDP), in which the transition dynamics to the next state depend only on the previous state. Mathematically speaking:

$$\mathbf{P}(\mathbf{S}_{t+1}|\mathbf{A}_t, \mathbf{S}_t) = \mathbf{P}(\mathbf{S}_{t+1}|\mathbf{A}_t, \mathbf{S}_t, \mathbf{S}_{t-1}, \mathbf{S}_{t-2}, ..., \mathbf{S}_1) \tag{7}$$

where $\mathbf{P}$ is the transition probability function, which is a general formulation for stochastic systems. $\mathbf{S}$ and $\mathbf{A}$ are the states and actions in the corresponding time indexes, respectively. The same formulation can be applied to deterministic dynamics.

DRL has been recently suggested for feedback control systems [94], [95], and has been applied to robotic applications [96], [97], [98].

A stream of research has focused on using DRL for the purpose of controlling musculoskeletal systems. In particular, a team of researchers at Stanford University have integrated an openSim musculoskeletal environment with

DRL-based controller platforms. The resulting platform has been used in NeuralIPS 2019 challenge: "Learn to Move-Walk Around" [99]. Multiple researchers have since studied DRL algorithms to control a full musculoskeletal system with 8 Degree of Freedom (DOF) and 22 muscles.

Combining RL with imitation learning, which we discuss in later sections, has enabled realistic human movements on a complex muskuloskeletal system with 346 muscles [100]. This data-driven controller has led to predictive simulations of various human movements. In addition, multiple anatomical conditions including bone deformity, muscle weakness, and contracture were studied. This method has successfully simulated pathological gait patterns and human-prosthethis movement. In [101], DRL was coupled with curriculum learning [102] for simulating natural human gait. This approach splits the overall reward function into smaller portions and trains the agent sequentially; hence, it is highly applicable to human movement problems in which the human (is assumed) to optimize a different function at each stage of movement (standing up, walking, picking up objects for example). In this manner, curriculum learning is similar to the general multi-stage optimization method.

DRL has also been used in autonomous vehicle research. In [103], DRL was applied to an autonomous vehicle for adaptive cruise control. The results show comparable performance with MPC when there are no uncertainties and better performance in the presence of delays, disturbances, and unmodelled dynamics.

From the control-theoretic perspective, the focus is mostly on stability; however, the computer science community has concentrated on convergence properties [104]. This is one of main research gaps between the control-theoretic approach to reinforcement learning, often categorized as approximate dynamic programming, and the conventional RL used in the computer science community. Moreover, there is no explicit constraint handling in MFRL formulation. As a result, there are still safety concerns for MFRL, mainly on safety-critical applications. Recent studies have examined the possibility of combining MPC and MFRL to address this issue. For instance, the idea of safe learning-based MPC was presented in a recent review by Hewing *et al.* [105]. Based on this idea, the optimal control input is calculated based on MFRL and another optimization is solved in series. This optimization minimizes the deviation between the solution and the MFRL output; the problem constraints are included in this optimization to ensure safety.

MFRL is a powerful method as it can operate without a system transition model and can learn a policy from experiments. However, a main drawback is that a reward function should be defined; in some cases, this function is not straightforward to acquire, i.e. it is not clear what metric is being optimized to perform a certain task. Fortunately, two other research streams can offer a solution: imitation learning (Section 5.3) and inverse reinforcement learning (Section 5.4).

5.3 Imitation learning

Imitation Learning (IL) attempts to learn a policy from human demonstrations. In contrast to MFRL, which generates the training data using random (and partially-trained) policies as the experience, IL uses the data from a human expert performing a task so as to mimic that behavior [106]. Having access to optimal human demonstration bypasses the need to set a reward function.

Wu *et al.* [107] used imitation learning for robot impedance regulation. They stated that the problem of Cartesian impedance control can be formulated as an LQR with the stiffness coinciding with the LQR weighting matrix. Inspired by how humans dynamically regulate their arm impedance while interacting with the environment, IL was utilized with human demonstration data to learn to adaptively modify the stiffness parameter.

In [108], researchers utilized IL to increase the adaptive capacity of robots to unseen situations. Human demonstrations can be used to teach the robots various motor skills. Invariant motion patterns in the data are then extracted and reproduced in new unseen scenarios. This method was used for trajectory modulation in various unseen situations.

Similarly, Finn *et al.* [109] used IL to improve the generalizability of robots to unstructured complex environments. As it is impractical to train a DL for each specific task, they proposed training a meta-imitation learning scheme to teach the robots how to learn new skills more efficiently. They used a one-shot learning approach, which is learning a policy from a single demonstration. The authors used a CNN structure, which takes camera data and robot configuration as inputs and outputs the actions (robot torques) for reaching scenarios. Meta-learning has resulted in faster and more efficient learning of new skills by robots.

Kebria *et al.* [110] used imitation learning on an autonomous vehicle vision problem. Using the simulation data from three different camera views of a vehicle controlled by a human, a CNN structure was trained to map the raw image data to the final steering angle. Since the configuration of CNNs has a great impact on their performance, the researchers on this study compared 96 CNN configurations with different layers, number of filters, and filter sizes in terms of mean squared error to find the best configuration and explore the effect of each parameter. Finally, an ensemble of CNN models was used for steering angle prediction.

5.4 Inverse reinforcement learning

So far, the bulk of research has focused on obtaining accurate and fast controllers that can minimize or maximize the intuitively-defined metrics. Recent advances in optimization-based control and deep learning, however, have led to posing a new question: What is the underlying metric that an expert minimizes or maximizes to do a certain task? This research direction aspires to obtain more accurate cost/reward functions. Inverse Reinforcement Learning (IRL),

which is the model-free version of inverse optimal control, has been utilized for this purpose [111]. This approach follows a reverse methodology of RL; the already running policy is considered as an optimal controller and the input-output tuples generated from this policy are used to find the reward/cost function that is optimized to find the policy.

Autonomous driving can take good advantage of IRL. To design reliable controllers for autonomous vehicles, it is imperative to mimic human behavior in driving by deriving the human reward function. Nevertheless, this function is complex, stochastic, and varies greatly between drivers [112], [113], [114]. To remedy this, IRL has been used based on gathered human expert data. You *et al.* [112] used a DNN to approximate the reward function that humans minimize in driving using an IRL setting. The optimal DNN parameters were obtained by gradient-based optimization and maximum entropy principle (MEP). Wu *et al.* [113] improved upon previous IRL algorithms for autonomous driving by introducing a continuous trajectory sampler, which uses prior knowledge on vehicle kinematics and motion planning algorithms. This addition enables the use of IRL in a continuous formulation, making it applicable to large-scale continuous problems. In addition, this study has used real traffic data; while previous research has assumed optimal (near-optimal) policies for real data, this study considers the uncertainty in the human demonstrations to account for the terms that interfere with ideal human driving.

Woodworth *et al.* [115] took advantage of IRL to find the preferences of different users to an assistive robot. While previous applications of robots in healthcare have focused on imposing a certain task, like finishing a certain exercise by the user, this research has focused on finding a user-specific preference while performing each task. The underlying algorithm should be able to adapt to a multitude of tasks and subjects. This research has used observational repeated inverse reinforcement learning to perform long-term, task-independent preference learning in scenarios with incomplete information (only partial knowledge about the task and the environment is available). The algorithm observes each subject's data while doing multiple rehabilitation exercises to infer their motor learning scheme. This study enables the subject-specific control of these devices and their compatibility to each subject.

IRL is applied to a very complex and highly redundant snake-like robot in [116]. An RL controller was first used to find energy-efficient and realistic gait at various velocities. An adversarial IRL was then trained based on the demonstrations from the RL controller using a GAN. By improving upon the initial reward function, the IRL controller is able to preserve the efficiency and natural gait resulting from the RL controller while adapting to changing dynamics due to damaged body parts. In this sense, the IRL controller outperforms both the RL and model-based controllers.

5.5 Other applications

So far, the papers in Section 5 have discussed data-driven methods to substitute a control policy. The summary of these approaches are provided in Table 2. There are, however, other sections that complete/augment the closed-loop system; controller tuners, state observers, and disturbance observers are among the important components of a closed-loop system. In this section, we discuss the ML research on these components.

Data-driven controller weights adjustment

In many cases, existing control approaches are sufficient for obtaining the desired performance in multibody dynamics systems. However, every control method has multiple parameters (gains) that require meticulous tuning. This process is usually done manually, which results in a time-intensive manual trial and error adjustment; moreover, the manual tuning does not achieve the optimal set of weights. Current tuning techniques are limited to simple controllers and cannot generalize to all approaches. Plus, they still suffer from repetitive trial and error iterations [117]. Recently, automatic tuning methods have been studied; general data-driven methods are exploited for this purpose. In [118], the bees algorithm was used to tune an LQR controller on an inverted pendulum control problem. Researchers in [119] took advantage of Gaussian optimization, namely simultaneous perturbation stochastic approximation, to tune the same controller for a robot arm balancing an inverted pole. Particle swarm optimization was also investigated to tune a PID controller [120]. The resulting controller has been applied to a ball on plate trajectory tracking system. Recently, a genetic algorithm was integrated with an SMC for rehabilitation robot control [121].

While the above approaches work on a variety of problems, more complex approximators are usually required for large-scale problems. In addition, most of above data-driven approaches utilize static optimization, which cannot be used for adaptive dynamic weight adjustment. DL, and more specifically DRL, methods have been used to remedy these issues. Several works have taken advantage of DRL for tuning a PID controller in single-agent [122], [123], [117], and multi-agent problems [124]. DRL has also been applied to more advanced controllers. Specifically, a learning automata RL algorithm has been integrated with a linear MPC for autonomous vehicle control and mobile robotics. The resulting controller has been tested in simulation [125] and experiments [126], [127]. An NMPC has also been automatically tuned using incremental DRL for unmanned aerial vehicle control [128]. Liu *et al.* [129] used a NN approach to dynamically approximate the switching gain of a SMC for an uncertain robot control. The neural network SMC adapts to unknown dynamics and disturbance, and can resolve the chattering and high-input issue of SMC.

The cited papers have focused on low-level controller tuning. On the other hand, several studies have used DRL for tuning mid-level controllers. These controllers are extensively used in rehabilitation robotics and assistive devices

as they directly adjust the compliance in human-robot interactions. In particular, online tuning of mid-level controllers are investigated in the form of impedance [98], [130], [131] and admittance [132] controllers. These intelligent controllers have enabled online personalization of assistive devices and exoskeletons to different subjects, resulting in less muscle effort and from the subject and more comfort during use.

NN-based observers and estimators

Various issues may arise in real-time control of multibody dynamics systems. First, exact kinematic and dynamic parameters are usually not available; the uncertainty around the parameter values can degrade the performance of the controller. Moreover, the values of some of the parameters may vary during system operation. Second, external disturbances can affect the system stability (and performance). In the control community, robust control methods are utilized to mitigate this issue. The controller is usually augmented with a disturbance estimator, which usually requires some knowledge about the disturbance characteristics (e.g. its model). NN-based estimators rely solely on the sensor data to approximate the disturbance and are advantageous when no knowledge about the disturbance model can be obtained. Third, while most controllers work with system states (state-feedback controllers for example), measured outputs cannot obtain all the states in a variety of problems. As a solution, state observers are developed to approximate some (or all) of the states. We talked about state-based observers using DL-based AI models in Section 3.5 with the work of [55]. This work is applicable to problems where an approximate system model can be acquired. Researchers have also worked on complete data-driven observers, discussed below, that do not require a model.

Wu *et al.* [133] presented the idea of using neural-network-based observers in rehabilitation robotics. Radial Basis Function (RBF) network was used as a disturbance observer. The network was also utilized to account for robot modeling errors. Using this scheme, the authors were able to implement an adaptive admittance controller on the rehabilitation robot without concerns about closed-loop stability.

A recurrent neuro-adaptive state observer was proposed in [134]. This RNN-based scheme was applied to a flexible-joint manipulator and was trained by modified backpropagation; to ensure the stability of the observer, an e-modification term was added to the optimization; this term is derived from robust adaptive control theory for systems with bounded disturbances. The advantage of the RNN-based observer is the fact that it does not require a model; hence it is applicable to complex nonlinear systems for which an accurate representation of the system is not easy to acquire.

RBF networks were also utilized to estimate the unknown parameters of cooperative multiple manipulators [135]. They were also incorporated as the disturbance observers when the torques reach their upper-bounds. Almost all actuated systems are susceptible to input saturation. As a result, this research proposed a data-driven approach to resolve this problem.

Adaptive data-driven controllers

In most real-time control applications, both the dynamics of the plant and the environment change during system operation. We previously discussed how data-driven approaches can estimate the change in the dynamics. Alternatively, the changing conditions can be taken into consideration by using an adaptive controller, therefore improving the performance of the closed-loop system. Here, we discuss the research on data-driven adaptive controllers.

Luan *et al.* [136] applied an RBF network adaptive controller to a SCARA robot. This controller was combined with SMC to guarantee robustness against uncertainties and disturbance. The same idea was also applied to three-link industrial robot manipulators [137]. The resulting controller outperformed PID and fuzzy controllers both in terms of tracking accuracy and input expenditure. An adaptive NN tracking controller was designed in [138] for robotic manipulators with dead-zones. A RBF neural network was used to augment a back-stepping controller by approximating the intermediate control functions. Neural networks were also used in [139] for adaptive control of a 2-DOF helicoper with input dead-zone and output constraint. Adaptive control theory was used to update the NN weights and hence, introduce adaptivity to the closed-loop system. The integral barrier Lyapunov function was utilized to ensure output constraint satisfaction. Another study focused on robotic manipulators with external disturbances and time-varying outputs [140]. A Model-based backstepping controller was augmented to an adaptive NN-based controller.

Table 2 Summary of the data-driven control methods. ML Opt refers to ML-based MPC Optimization discussed in Section 5.1

Method	Model?	Data efficiency	Description	Example papers
ML Opt	✓	high	Approximates the the MPC completely by training a neural network on synthetic data from a simulated MPC	[89], [91], [93]
MBRL	✓	high	MPC is implemented by approximating the control-oriented model using a neural network	[76], [78], [77]
MFRL	✗	low	Uses the data from interaction with the environment to find the optimal policy	[98], [99], [103]
IM	✗	low	Trains a neural network based on expert data to replicate the expert action	[107], [110], [141]
IRL	✗	low	Uses the input/output from the optimal policy to find the underlying reward/cost function	[112], [115], [116]

6 Future directions and ideas

Although there has been a myriad of recent research in using deep learning in multibody dynamics and control, many opportunities exist for future studies. In this section, we examine two promising research paths that can be explored by the multibody dynamics community. These areas have been investigated in other fields; we believe they have the potential to advance the field of multibody system dynamics.

6.1 Physics-informed neural networks

Raissi *et al.* [142] have recently introduced physics-informed neural networks, in which laws of physics are considered in the training of a neural network as the surrogate model. They have shown that the neural network-based approximation of any physical system that is described by a nonlinear Partial Differential Equation (PDE) can exploit the physics of the system (as prior knowledge) in the training. To this end, they have proposed a second neural network with the same weights as the primary network that approximates the system variable. The second neural network enforces the right hand side of the PDE to be zero. In other words, the same weights that approximate the system variable enforce the PDE equation.

As any multibody system follows certain dynamic laws, we suggest using the system physics (model equations) as a regularization scheme; the additional constraint would avoid unrealistic networks, and would enable the training with fewer samples. The idea of physics-informed models has already been tested in MSD and it has shown potential opportunities for future research. Lutter *et al.* [143] have introduced Deep Lagrangian Network (DeLaN), which impose Lagrangian mechanics on the training. The augmentation has led to more robust neural networks, capable of generalizing to new samples with fewer training cases; the resulting network is also physically plausible. DeLaN has been used to represent the inverse dynamics of a robot for a trajectory tracking problem; the lower-triangular matrix $\mathbf{L}(\mathbf{q})$ composing the inertia matrix $\mathbf{M}(\mathbf{q})$ is represented by a deep neural network as:

$$\hat{\mathbf{M}}(\mathbf{q}) = \hat{\mathbf{L}}(\mathbf{q};\boldsymbol{\theta})\hat{\mathbf{L}}(\mathbf{q};\boldsymbol{\theta})^T \tag{8}$$

where $\mathbf{q}$ is the robot joint angle and $\boldsymbol{\theta}$ are the NN parameters. An optimization problem is then solved to find the best set of parameters while ensuring the positive-definiteness of the inertia matrix. The inertia matrix, resulted from the matrix $\hat{\mathbf{L}}(\mathbf{q};\boldsymbol{\theta})$, is utilized to generate the system equations. The optimization then aims to minimize the error between the multibody model and torque sample data.

DeLaN is one example of general physics-informed networks and many other solutions exist. For example, Hamiltonian Neural Networks presented by Greydanus *et al.* [144] can learn an arbitrary conservation law based on Hamiltonian mechanics. While DeLaN requires existence of the mass matrix

$\mathbf{M}(\mathbf{q})$, Cranmer *et al.* [145] provides a neural network architecture for learning arbitrary Lagrangians. However, as pointed out by Krishnapriyan *et al.* [146], special learning techniques may be required for accurate training of physically-informed neural networks in many real-life scenarios.

6.2 More complex networks

Most research in data-driven multibody dynamics and control has used common network structures: ANNs, CNNS, RNNs, and autoencoders. With recent advances in network configurations, we believe that the MSD community can benefit from new structures. In this section, we present the application of new neural networks to multibody dynamic modeling and control.

Temporal convolutional neural networks

To combine the feature extraction and expression capabilities of CNNs with the temporal modelling capabilities of RNNs, temporal convolutional neural networks have recently been developed [147]. While solving the gradient vanishing/explosion problem (similar to LSTM networks), Temporal CNNs also introduce more efficiency by adding more parallel computations; unlike the conventional dependency on previous timesteps for current timestep calculation (as in RNNs, LSTMs, gate recurrent units), temporal CNNs use the same filter for the whole sequence, hence enabling parallel computations. Temporal CNNs have been investigated for system identification in [148]. The resulting data-driven approach has been successfully used for the Silverbox dataset, which is the electronic version of the Duffing oscillator, and to identify an F-16 ground vibration test.

Transformers

With more focus on the attention mechanism, recently developed transformer networks have added considerable improvements to machine translation, computer vision, and other applications [149], [150]. While LSTM networks solve the problem of vanishing/exploding gradients to some extent, for some applications like machine translation where longer output dependency to earlier elements in the sequence is observed, this issue is not completely resolved. To this end, the attention mechanism utilizes all the hidden states, instead of only the last one, and decides what elements of the hidden states are relevant to generating the output; roughly speaking, the attention mechanism learns where to focus when given a long sequence of data (hence the name). In addition to resolving the long-dependency issue, similar to TCNNs, the unique structure of transformers allows for parallelization of computations on sequential data. Due to their temporal component, transformers have the potential to improve the previous work on ML-based MSD modelling and control, especially in cases where the output from the current timestep has a high dependency on input/output from timesteps in the past.

Graph network-based simulators

Modelling challenges are expected as the number of components in a multibody system increases. Hence, solutions with more expressive capabilities should be used. For complex physical systems, like MSD systems with many DOF and components, finding an accurate model is difficult and simulating it is computationally expensive. Using NN-based surrogate models (Section 3) with conventional network structures may still lead to high computations during inference.

Hence, a network with an inherent structure to handle higher input components is preferred. Recently, an interesting class of networks called Graph Network-Based Simulator (GNS) was developed for modelling complex physical systems with very high input size [151]. These networks are a special kind of graph neural networks. A unique feature of graph neural networks is that they can accept an unstructured graph as input, i.e. no constraint on the topology or size of the input is imposed. This enables structuring the physical system as a graph and feeding it to the network. In the GNS network, the physical states are represented as graphs of interacting particles structured as nodes and the message-passing between nodes are learned using data. GNS has been implemented on a range of complex physical systems including fluid dynamics, rigid solids, and deformable materials. We believe GNS has potential for modelling complex multibody dynamics with higher states and components.

7 Conclusions

The use of ML in multibody system dynamics is growing rapidly. In this paper, we discussed the multiple streams of research on the integration of these two fields and the ways in which data-driven solutions can provide improvements to MSD research.

Neural networks may provide surrogate dynamic models with comparable accuracy to multibody models but with higher computational efficiency. Moreover, they can be utilized in hybrid models where part of the system is difficult to model analytically and is replaced with a data-driven blackbox alternative.

ML-based solutions can also improve the controllers applied to multibody systems. Efficient control-oriented models within a model-based controller can be obtained from data. In addition, the control policy itself can be approximated using data-driven approaches. Other aspects of control design can also be improved by ML. Neural networks can replace the conventional state/disturbance estimators. They can also be used for online controller parameter adjustment, facilitating the use of adaptive controllers.

Although ML methods have improved the accuracy/efficiency of multibody system models and controllers, there is a huge potential for further integration of these two fields. While new network structures and data-driven solutions can potentially improve the current state of this integration, physical models can also be used to improve the network learning. More research is needed

to move towards the optimal combination of physics-based and data-driven solutions in MSD systems.

Acknowledgements The first and last authors gratefully acknowledge funding provided by the Canada Research Chairs program and the Natural Sciences and Engineering Research Council of Canada. The second and third authors gratefully acknowledge funding provided by Business Finland (Santtu project).

Declarations

Ethical approval

Not applicable.

Competing interests

Not applicable.

Authors' contributions

- Arash Hashemi: conceptualization, visualization, writing (Sections 2,4,5,6,7), review and editing.
- Grzegorz Orzechowski: conceptualization, visualization, writing (Abstract, Sections 1,3), review and editing.
- Aki Mikkola: conceptualization, funding acquisition, supervision, writing (Abstract, Sections 1,3), review and editing.
- John McPhee: conceptualization, funding acquisition, supervision, writing (Sections 2,4,5,6,7), review and editing.

Funding

The first and last authors gratefully acknowledge funding provided by the Canada Research Chairs program and the Natural Sciences and Engineering Research Council of Canada. The second and third authors gratefully acknowledge funding provided by Business Finland's "Kumppanuusmalli – SANTTU – LUT" project under grant 8859/31/2021.

Availability of data and materials

Not applicable (review paper).

References

1. I. Kurinov, G. Orzechowski, P. Hamalainen, and A. Mikkola, "Automated Excavator Based on Reinforcement Learning and Multibody System Dynamics," *IEEE Access*, vol. 8, pp. 213998–214006, 2020.
2. D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman, "MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale," Apr. 2021. arXiv:2104.08212 [cs].
3. R. Cordeschi, "AI turns fifty: revisiting its origins," *Applied Artificial Intelligence*, vol. 21, no. 4-5, pp. 259–279, 2007.
4. C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning," pp. 1–20, 2018.
5. J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.
6. Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, contour and grouping in computer vision*, pp. 319–345, Springer, 1999.
7. A. Dhillon and G. K. Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," *Progress in Artificial Intelligence*, vol. 9, no. 2, pp. 85–112, 2020.
8. H. C. Li, Z. Y. Deng, and H. H. Chiang, "Lightweight and resource-constrained learning network for face recognition with performance optimization," *Sensors (Switzerland)*, vol. 20, no. 21, pp. 1–20, 2020.
9. H. Salman, J. Grover, and T. Shankar, "Hierarchical Reinforcement Learning for Sequencing Behaviors," vol. 2733, no. March, pp. 2709–2733, 2018.
10. B. Laschowski, W. McNally, A. Wong, and J. McPhee, "Preliminary design of an environment recognition system for controlling robotic lower-limb prostheses and exoskeletons," *IEEE International Conference on Rehabilitation Robotics*, vol. 2019-June, pp. 868–873, 2019.
11. W. McNally, K. Vats, A. Wong, and J. McPhee, "EvoPose2D: Pushing the Boundaries of 2D Human Pose Estimation using Neuroevolution," 2020.
12. D. Palaz, M. Magimai-Doss, and R. Collobert, "End-to-end acoustic modeling using convolutional neural networks for HMM-based automatic speech recognition," *Speech Communication*, vol. 108, no. June 2016, pp. 15–32, 2019.
13. Ö. Batur Dinler and N. Aydin, "An optimal feature parameter set based on gated recurrent unit recurrent neural networks for speech segment detection," *Applied Sciences (Switzerland)*, vol. 10, no. 4, 2020.
14. W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative Study of CNN and RNN for Natural Language Processing," 2017.
15. S. Chandar and H. Sunder, "Dynamic Systems Simulation and Control Using Consecutive Recurrent Neural Networks," *Communications in Computer and Information Science*, vol. 1290, pp. 92–103, 2020.
16. A. P. Trischler and G. M. D'Eleuterio, "Synthesis of recurrent neural networks for dynamical system simulation," *Neural Networks*, vol. 80, pp. 67–78, 2016.
17. Y. Hu, A. Huber, J. Anumula, and S.-C. Liu, "Overcoming the vanishing gradient problem in plain recurrent networks," no. Section 2, pp. 1–20, 2018.
18. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
19. C. Gao, J. Yan, S. Zhou, P. K. Varshney, and H. Liu, "Long short-term memory-based deep recurrent neural networks for target tracking," *Information Sciences*, vol. 502, pp. 279–296, 2019.
20. H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent Neural Networks for Time Series Forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021.
21. S. Wright, J. Nocedal, *et al.*, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

22. L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*, pp. 421–436, Springer, 2012.
23. Z. Zhang, "Improved Adam Optimizer for Deep Neural Networks," *2018 IEEE/ACM 26th International Symposium on Quality of Service, IWQoS 2018*, pp. 1–2, 2019.
24. F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning," 2017.
25. J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," pp. 1–9.
26. X. Wu, V. Kumar, Q. J. Ross, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z. H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, *Top 10 algorithms in data mining*, vol. 14. 2008.
27. X. Xu, T. Liang, J. Zhu, D. Zheng, and T. Sun, "Review of classical dimensionality reduction and sample selection methods for large-scale data processing," *Neurocomputing*, vol. 328, pp. 5–15, 2019.
28. G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.
29. Richard Bellmann, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, pp. 503–515, 1954.
30. M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, no. 6, pp. 1107–1149, 2004.
31. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
32. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
33. J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Ontinuous learning control with deep reinforcement," 2016.
34. S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," *35th International Conference on Machine Learning, ICML 2018*, vol. 4, pp. 2587–2601, 2018.
35. I. J. Goodfellow, J. Pouget-abadie, M. Mirza, B. Xu, and D. Warde-farley, "Generative Adversarial Nets," pp. 1–9.
36. H. Ren and P. Ben-Tzvi, "Learning inverse kinematics and dynamics of a robotic manipulator using generative adversarial networks," *Robotics and Autonomous Systems*, vol. 124, p. 103386, 2020.
37. H. A. Ardeh, M. Tupy, and D. Negrut, "On the Construction and Use of Surrogate Models for the Dynamic Analysis of Multibody Systems," in *Volume 13: New Developments in Simulation Methods and Software for Engineering Applications; Safety Engineering, Risk Analysis and Reliability Methods; Transportation Systems*, vol. 13, pp. 17–26, ASMEDC, 1 2009.
38. B. Azzam, R. Schelenz, B. Roscher, A. Baseer, and G. Jacobs, "Development of a wind turbine gearbox virtual load sensor using multibody simulation and artificial neural networks," *Forschung im Ingenieurwesen/Engineering Research*, vol. 85, pp. 241–250, 6 2021.
39. O. García Peyrano, J. Vignolo, R. Mayer, and M. Marticorena, "Online unbalance detection and diagnosis on large flexible rotors by svr and ann trained by dynamic multibody simulations," *Journal of Dynamics, Monitoring and Diagnostics*, Jun. 2022.
40. S. Han, H. S. Choi, J. Choi, J. H. Choi, and J. G. Kim, "A DNN-based data-driven modeling employing coarse sample data for real-time flexible multibody dynamics simulations," *Computer Methods in Applied Mechanics and Engineering*, vol. 373, p. 113480, 2021.

41. M. Kahr, G. Kovács, M. Loinig, and H. Brückl, "Condition monitoring of ball bearings based on machine learning with synthetically generated data," *Sensors*, vol. 22, no. 7, 2022.
42. J. Ma, S. Dong, G. Chen, P. Peng, and L. Qian, "A data-driven normal contact force model based on artificial neural network for complex contacting surfaces," *Mechanical Systems and Signal Processing*, vol. 156, p. 107612, 2021.
43. C. Sobie, C. Freitas, and M. Nicolai, "Simulation-driven machine learning: Bearing fault classification," *Mechanical Systems and Signal Processing*, vol. 99, pp. 403–419, 1 2018.
44. Y. Ye, B. Zhu, P. Huang, and B. Peng, "Oornet: A deep learning model for on-board condition monitoring and fault diagnosis of out-of-round wheels of high-speed trains," *Measurement*, vol. 199, p. 111268, 2022.
45. A. S. Polydoros, L. Nalpantidis, and V. Kruger, "Real-time deep learning of robotic manipulator inverse dynamics," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, no. October, pp. 3442–3448, 2015.
46. L. Rane, Z. Ding, A. H. McGregor, and A. M. Bull, "Deep Learning for Musculoskeletal Force Prediction," *Annals of Biomedical Engineering*, vol. 47, no. 3, pp. 778–789, 2019.
47. A. Nasr, K. A. Inkol, S. Bell, and J. McPhee, "Inversemusclenet: Alternative machine learning solution to static optimization and inverse muscle modeling," *Frontiers in Computational Neuroscience*, vol. 15, 2021.
48. A. Byravan and D. Fox, "SE3-nets: Learning rigid body motion using deep neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 173–180, IEEE, 5 2017.
49. H. S. Choi, J. An, S. Han, J. G. Kim, J. Y. Jung, J. Choi, G. Orzechowski, A. Mikkola, and J. H. Choi, "Data-driven simulation for general-purpose multibody dynamics using Deep Neural Networks," *Multibody System Dynamics*, vol. 51, no. 4, pp. 419–454, 2021.
50. F. Hegedüs, P. Gáspár, and T. Bécsi, "Fast Motion Model of Road Vehicles with Artificial Neural Networks," *Electronics*, vol. 10, p. 928, 4 2021.
51. S. Kraft, J. Causse, and A. Martinez, "Black-box modelling of nonlinear railway vehicle dynamics for track geometry assessment using neural networks," *Vehicle System Dynamics*, vol. 57, no. 9, pp. 1241–1270, 2019.
52. T. P. Martin, K. E. Zaazaa, B. Whitten, and A. Tajaddini, "Using a Multibody Dynamic Simulation Code With Neural Network Technology to Predict Railroad Vehicle-Track Interaction Performance in Real Time," in *Volume 5: 6th International Conference on Multibody Systems, Nonlinear Dynamics, and Control, Parts A, B, and C*, vol. 5 PART C, pp. 1881–1891, ASMEDC, 1 2007.
53. Y. Pan, X. Nie, Z. Li, and S. Gu, "Data-driven vehicle modeling of longitudinal dynamics based on a multibody model and deep neural networks," *Measurement*, vol. 180, p. 109541, 2021.
54. A. Nasr, S. Bell, J. He, R. L. Whittaker, N. Jiang, C. R. Dickerson, and J. McPhee, "Musclenet: mapping electromyography to kinematic and dynamic biomechanical variables by machine learning," *Journal of Neural Engineering*, vol. 18, no. 4, p. 0460d3, 2021.
55. A. Angeli, W. Desmet, and F. Naets, "Deep learning for model order reduction of multibody systems to minimal coordinates," *Computer Methods in Applied Mechanics and Engineering*, vol. 373, p. 113517, 2021.
56. B. A. Hosking and J. McPhee, "Powertrain Modeling and Model Predictive Longitudinal Dynamics Control for Hybrid Electric Vehicles," *SAE Technical Papers*, vol. 2018-April, pp. 1–10, 2018.
57. A. Oishi and G. Yagawa, "Computational mechanics enhanced by deep learning," *Computer Methods in Applied Mechanics and Engineering*, vol. 327, pp. 327–351, 12 2017.
58. Y. Ye, P. Huang, Y. Sun, and D. Shi, "MBSNet: A deep learning model for multibody dynamics simulation and its application to a vehicle-track system," *Mechanical Systems and Signal Processing*, vol. 157, p. 107716, 2021.
59. D. J. Saxby, B. A. Killen, C. Pizzolato, C. P. Carty, L. E. Diamond, L. Modenese, J. Fernandez, G. Davico, M. Barzan, G. Lenton, S. B. da Luz, E. Suwarganda, D. Devaprakash, R. K. Korhonen, J. A. Alderson, T. F. Besier, R. S. Barrett, and D. G.

Lloyd, "Machine learning methods to support personalized neuromusculoskeletal modelling," *Biomechanics and Modeling in Mechanobiology*, vol. 19, no. 4, pp. 1169–1185, 2020.
60. S. L. Brunton, J. Nathan Kutz, K. Manohar, A. Y. Aravkin, K. Morgansen, J. Klemisch, N. Goebel, J. Buttrick, J. Poskin, A. W. Blom-Schieber, T. Hogan, and D. McDonald, "Data-Driven Aerospace Engineering: Reframing the Industry with Machine Learning," *AIAA Journal*, vol. 59, no. 8, pp. 1–26, 2021.
61. A. Hashemi, Y. Lin, W. McNally, B. Laschowski, B. Hosking, A. Wong, and J. McPhee, "Integration of Machine Learning with Dynamics and Control : From Autonomous Cars to Biomechatronics," *Canadian Society for Mechanical Engineering (CSME) Bulletin*, pp. 9–10, 2019.
62. S. Zhou and A. P. Schoellig, "Active Training Trajectory Generation for Inverse Dynamics Model Learning with Deep Neural Networks," *Proceedings of the IEEE Conference on Decision and Control*, vol. 2019-Decem, no. i, pp. 1784–1790, 2019.
63. P. Urda, J. F. Aceituno, S. Muñoz, and J. L. Escalona, "Artificial neural networks applied to the measurement of lateral wheel-rail contact force: A comparison with a harmonic cancellation method," *Mechanism and Machine Theory*, vol. 153, p. 103968, 2020.
64. A. Nasr, A. Hashemi, and J. McPhee, "Model-based mid-level regulation for assist-as-needed hierarchical control of wearable robots: A computational study of human-robot adaptation," *Robotics*, vol. 11, no. 1, p. 20, 2022.
65. N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network vehicle models for high-performance automated driving," *Science Robotics*, vol. 4, no. 28, 2019.
66. D. Bertsekas, *Reinforcement learning and optimal control*. Athena Scientific, 2019.
67. A. G. Richards, *Robust constrained model predictive control*. PhD thesis, Massachusetts Institute of Technology, 2005.
68. P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.
69. M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.
70. C.-V. Pal and F. Leon, "Brief survey of model-based reinforcement learning techniques," in *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 92–97, IEEE, 2020.
71. T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking model-based reinforcement learning," *arXiv preprint arXiv:1907.02057*, 2019.
72. H. Kwakernaak, R. Sivan, and B. N. D. Tyreus, "Linear optimal control systems," 1974.
73. Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913, IEEE, 2012.
74. J. H. Lee, "Model predictive control: Review of the three decades of development," *International Journal of Control, Automation and Systems*, vol. 9, no. 3, pp. 415–424, 2011.
75. M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327–1349, 2021.
76. L. Roveda, J. Maskani, P. Franceschi, A. Abdi, F. Braghin, L. Molinari Tosatti, and N. Pedrocchi, "Model-Based Reinforcement Learning Variable Impedance Control for Human-Robot Collaboration," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 100, no. 2, pp. 417–433, 2020.
77. A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 7579–7586, 2018.
78. G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," pp. 1–4, 2016.

79. G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic MPC for model-based reinforcement learning," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1714–1721, 2017.
80. S. Richards, N. Azizan, J.-J. Slotine, and M. Pavone, "Adaptive-Control-Oriented Meta-Learning for Nonlinear Systems," 2021.
81. A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3223–3230, 2015.
82. S. Bansal, F. J. Jiang, C. J. Tomlin, and S. Y. Oct, "Learning Quadrotor Dynamics Using Neural Network for Flight Control," no. 0931843.
83. S. J. Rutherford and D. J. Cole, "Modelling nonlinear vehicle dynamics with neural networks," vol. 53, no. 4, pp. 260–287, 2010.
84. A. Angeli, W. Desmet, and F. Naets, "Deep learning of multibody minimal coordinates for state and input estimation with Kalman filtering," *Multibody System Dynamics*, vol. 53, no. 2, pp. 205–223, 2021.
85. A. Alessandri, M. Baglietto, G. Battistelli, and V. Zavala, "Advances in Moving Horizon Estimation for Nonlinear Systems," pp. 5681–5688, 2010.
86. C. Jin, A. Maitland, and J. McPhee, "Hierarchical Nonlinear Moving Horizon Estimation of Vehicle Lateral Speed and Road Friction Coefficient," *ASME Letters in Dynamic Systems and Control*, 2020.
87. D. Sun, Q. Liao, T. Stoyanov, A. Kiselev, and A. Loutfi, "Bilateral telerobotic system using Type-2 fuzzy neural network based moving horizon estimation force observer for enhancement of environmental force compliance and human perception," *Automatica*, vol. 106, pp. 358–373, 2019.
88. A. Alessio and A. Bemporad, "A survey on explicit model predictive control," *Lecture Notes in Control and Information Sciences*, vol. 384, pp. 345–369, 2009.
89. R. Winqvist, A. Venkitaraman, and B. Wahlberg, "On Training and Evaluation of Neural Network Approaches for Model Predictive Control," 2020.
90. S. Diamond and S. Boyd, "Cvxpy: A python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.
91. P. Varshney, G. Nagar, and I. Saha, "DeepControl: Energy-Efficient Control of a Quadrotor using a Deep Neural Network," *IEEE International Conference on Intelligent Robots and Systems*, pp. 43–50, 2019.
92. Y. Pan and J. Wang, "Nonlinear model predictive control using a recurrent neural network," *Proceedings of the International Joint Conference on Neural Networks*, no. July, pp. 2296–2301, 2008.
93. Y. Cao and R. B. Gopaluni, "Deep neural network approximation of nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 11319–11324, 2020.
94. J. W. Roberts, I. R. Manchester, and R. Tedrake, "Feedback controller parameterizations for Reinforcement Learning," *IEEE SSCI 2011: Symposium Series on Computational Intelligence - ADPRL 2011: 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pp. 310–317, 2011.
95. F. L. Lewis and D. Vrabie, "Adaptive dynamic programming for feedback control," *Proceedings of 2009 7th Asian Control Conference, ASCC 2009*, pp. 1402–1409, 2009.
96. J. Kober, J. A. Bagnell, and J. Peters, "Kober_IJRR_2013," pp. 1–73, 2013.
97. C. Sun, J. Orbik, C. Devin, B. Yang, A. Gupta, G. Berseth, and S. Levine, "Fully Autonomous Real-World Reinforcement Learning for Mobile Manipulation," pp. 1–16, 2021.
98. M. Li, Y. Wen, X. Gao, J. Si, and H. Huang, "Toward Expedited Impedance Tuning of a Robotic Prosthesis for Personalized Gait Assistance by Reinforcement Learning Control," *IEEE Transactions on Robotics*, pp. 1–13, 2021.
99. S. Song, Ł. Kidziński, X. B. Peng, C. Ong, J. Hicks, S. Levine, C. G. Atkeson, and S. L. Delp, "Deep reinforcement learning for modeling human locomotion control in neuromechanical simulation," *Journal of NeuroEngineering and Rehabilitation*, vol. 18, no. 1, pp. 1–17, 2021.
100. S. Lee, M. Park, K. Lee, and J. Lee, "Scalable muscle-actuated human simulation and control," *ACM Transactions on Graphics*, vol. 38, no. 4, 2019.

101. J. Weng, E. Hashemi, and A. Arami, "Natural Walking with Musculoskeletal Models Using Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4156–4162, 2021.
102. Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.
103. Y. Lin, J. McPhee, and N. L. Azad, "Comparison of Deep Reinforcement Learning and Model Predictive Control for Adaptive Cruise Control," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 221–231, 2021.
104. L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018.
105. L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-Based Model Predictive Control: Toward Safe Learning in Control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, 2020.
106. A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–35, 2017.
107. Y. Wu, F. Zhao, T. Tao, and A. Ajoudani, "A Framework for Autonomous Impedance Regulation of Robots Based on Imitation Learning and Optimal Control," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 127–134, 2020.
108. Y. Huang, L. Rozo, J. Silverio, and D. G. Caldwell, "Non-parametric imitation learning of robot motor skills," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 5266–5272, 2019.
109. C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-Shot Visual Imitation Learning via Meta-Learning," no. CoRL, pp. 1–12, 2017.
110. P. M. Kebria, A. Khosravi, S. M. Salaken, and S. Nahavandi, "Deep imitation learning for autonomous vehicles based on convolutional neural networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 1, pp. 82–95, 2020.
111. N. Ab Azar, A. Shahmansoorian, and M. Davoudi, "From inverse optimal control to inverse reinforcement learning: A historical review," *Annual Reviews in Control*, vol. 50, pp. 119–138, 2020.
112. C. You, J. Lu, D. Filev, and P. Tsiotras, "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning," *Robotics and Autonomous Systems*, vol. 114, pp. 1–18, 2019.
113. Z. Wu, L. Sun, W. Zhan, C. Yang, and M. Tomizuka, "Efficient Sampling-Based Maximum Entropy Inverse Reinforcement Learning with Application to Autonomous Driving," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5355–5362, 2020.
114. S. Sharifzadeh, I. Chiotellis, R. Triebel, and D. Cremers, "Learning to Drive using Inverse Reinforcement Learning and Deep Q-Networks," no. Nips, pp. 1–7, 2016.
115. B. Woodworth, F. Ferrari, T. E. Zosa, and L. D. Riek, "Preference Learning in Assistive Robotics: Observational Repeated Inverse Reinforcement Learning," *Proceedings of Machine Learning Research*, vol. 85, pp. 1–19, 2018.
116. Z. Bing, C. Lemke, L. Cheng, K. Huang, and A. Knoll, "Energy-efficient and damage-recovery slithering gait design for a snake-like robot based on reinforcement learning and inverse reinforcement learning," *Neural Networks*, vol. 129, pp. 323–333, 2020.
117. E. Okafor, D. Udekwe, Y. Ibrahim, M. Bashir Mu'azu, and E. G. Okafor, "Heuristic and deep reinforcement learning-based PID control of trajectory tracking in a ball-and-plate system," *Journal of Information and Telecommunication*, vol. 5, no. 2, pp. 179–196, 2021.
118. H. H. Bilgic, M. A. Sen, and M. Kalyoncu, "Tuning of LQR controller for an experimental inverted pendulum system based on the bees algorithm," *Journal of Vibroengineering*, vol. 18, no. 6, pp. 3684–3694, 2016.
119. A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic LQR tuning based on Gaussian process global optimization," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 270–277, 2016.
120. S. U. Hussein, B. M. Muhammed, T. H. Sikiru, I. J. Umoh, and A. T. Salawudeen, "Trajectory tracking control of ball on plate system using weighted Artificial Fish

Swarm Algorithm based PID," *2017 IEEE 3rd International Conference on Electro-Technology for National Development, NIGERCON 2017*, vol. 2018-Janua, no. June 2018, pp. 554–561, 2018.
121. A. Hashemi and J. McPhee, "Assistive sliding mode control of a rehabilitation robot with automatic weight adjustment," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 4891–4896, 2021.
122. W. J. Shipman and L. C. Coetzee, "Reinforcement Learning and Deep Neural Networks for PI Controller Tuning," *IFAC-PapersOnLine*, vol. 52, no. 14, pp. 111–116, 2019.
123. I. Carlucho, M. De Paula, and G. G. Acosta, "An adaptive deep reinforcement learning approach for MIMO PID control of mobile robots," *ISA Transactions*, vol. 102, pp. 280–294, 2020.
124. A. El Hakim, H. Hindersah, and E. Rijanto, "Application of reinforcement learning on self-tuning PID controller for soccer robot multi-agent system," *Proceedings of the 2013 Joint International Conference on Rural Information and Communication Technology and Electric-Vehicle Technology, rICT and ICEV-T 2013*, 2013.
125. N. K. Ure, M. U. Yavas, A. Alizadeh, and C. Kurtulus, "Enhancing situational awareness and performance of adaptive cruise control through model predictive control and deep reinforcement learning," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2019-June, no. IV, pp. 626–631, 2019.
126. P. T. Jardine, S. N. Givigi, and S. Yousefi, "Experimental results for autonomous model-predictive trajectory planning tuned with machine learning," *11th Annual IEEE International Systems Conference, SysCon 2017 - Proceedings*, 2017.
127. P. T. Jardine, M. Kogan, S. N. Givigi, and S. Yousefi, "Adaptive predictive control of a differential drive robot tuned with reinforcement learning," *International Journal of Adaptive Control and Signal Processing*, vol. 33, no. 2, pp. 410–423, 2019.
128. M. Mehndiratta, E. Camci, and E. Kayacan, "Automated Tuning of Nonlinear Model Predictive Controller by Reinforcement Learning," *IEEE International Conference on Intelligent Robots and Systems*, no. October, pp. 3016–3021, 2018.
129. C. Liu, G. Wen, Z. Zhao, and R. Sedaghati, "Neural-Network-Based Sliding-Mode Control of an Uncertain Robot Using Dynamic Model Approximated Switching Gain," *IEEE Transactions on Cybernetics*, vol. 51, no. 5, pp. 2339–2346, 2021.
130. Y. Wen, J. Si, A. Brandt, X. Gao, and H. H. Huang, "Online Reinforcement Learning Control for the Personalization of a Robotic Knee Prosthesis," *IEEE Transactions on Cybernetics*, vol. 50, no. 6, pp. 2346–2356, 2020.
131. W. Wu, K. R. Saul, and H. Huang, "Using reinforcement learning to estimate human joint moments from electromyography or joint kinematics: An alternative solution to musculoskeletal-based biomechanics," *Journal of Biomechanical Engineering*, vol. 143, no. 4, pp. 1–9, 2021.
132. X. Tu, M. Li, M. Liu, J. Si, He, and Huang, "A Data-Driven Reinforcement Learning Solution Framework for Optimal and Adaptive Personalization of a Hip Exoskeleton," 2020.
133. Q. Wu, B. Chen, and H. Wu, "Adaptive admittance control of an upper extremity rehabilitation robot with neural-network-based disturbance observer," *IEEE Access*, vol. 7, pp. 123807–123819, 2019.
134. F. Abdollahi, H. A. Talebi, and R. V. Patel, "A stable neural network observer with application to flexible-joint manipulators," *ICONIP 2002 - Proceedings of the 9th International Conference on Neural Information Processing: Computational Intelligence for the E-Age*, vol. 4, no. 1, pp. 1910–1914, 2002.
135. W. He, Y. Sun, Z. Yan, C. Yang, Z. Li, and O. Kaynak, "Disturbance Observer-Based Neural Network Control of Cooperative Multiple Manipulators with Input Saturation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 5, pp. 1735–1746, 2020.
136. F. Luan, J. Na, Y. Huang, and G. Gao, "Adaptive neural network control for robotic manipulators with guaranteed finite-time convergence," *Neurocomputing*, vol. 337, pp. 153–164, 2019.
137. V. T. Yen, W. Y. Nan, and P. Van Cuong, "Robust Adaptive Sliding Mode Neural Networks Control for Industrial Robot Manipulators," *International Journal of Control, Automation and Systems*, vol. 17, no. 3, pp. 783–792, 2019.

138. Q. Zhou, S. Zhao, H. Li, R. Lu, and C. Wu, "Robotic Manipulators With Dead Zone," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 12, pp. 1–10, 2018.
139. Y. Ouyang, L. Dong, L. Xue, and C. Sun, "Adaptive control based on neural networks for an uncertain 2-DOF helicopter system with input deadzone and output constraints," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 807–815, 2019.
140. Y. Wu, R. Huang, X. Li, and S. Liu, "Adaptive neural network control of uncertain robotic manipulators with external disturbance and time-varying output constraints," *Neurocomputing*, vol. 323, pp. 108–116, 2019.
141. Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile Autonomous Driving using End-to-End Deep Imitation Learning," 2018.
142. M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.
143. M. Lutter, C. Ritter, and J. Peters, "Deep Lagrangian networks: Using physics as model prior for deep learning," *7th International Conference on Learning Representations, ICLR 2019*, pp. 1–17, 2019.
144. S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.
145. M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, "Lagrangian Neural Networks," July 2020. arXiv:2003.04630 [physics, stat].
146. A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, "Characterizing possible failure modes in physics-informed neural networks," in *Advances in Neural Information Processing Systems*, vol. 34, pp. 26548–26560, Curran Associates, Inc., 2021.
147. C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 156–165, 2017.
148. C. Andersson, A. H. Ribeiro, K. Tiels, N. Wahlstrom, and T. B. Schon, "Deep Convolutional Networks in System Identification," *Proceedings of the IEEE Conference on Decision and Control*, vol. 2019-Decem, pp. 3670–3676, 2019.
149. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
150. T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *arXiv preprint arXiv:2106.04554*, 2021.
151. A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *International Conference on Machine Learning*, pp. 8459–8468, PMLR, 2020.