Received 30 11 2020; | Revised Day Month Year; | Accepted Day Month Year

DOI: xxx/xxxx

ARTICLE

Structured learning of rigid-body dynamics: A survey and unified view

A. René Geist[1] | Sebastian Trimpe[1,2]

[1]Intelligent Control Systems Group, Max Planck Institute for Intelligent Systems, Stuttgart, Germany
[2]Institute for Data Science in Mechanical Engineering, RWTH Aachen University, Aachen, Germany

Correspondence
A. René Geist, Heisenbergstr. 3, 70569 Stuttgart Germany, Email: geist@is.mpg.de

Abstract

Accurate models of mechanical system dynamics are often critical for model-based control and reinforcement learning. Fully data-driven dynamics models promise to ease the process of modeling and analysis, but require considerable amounts of data for training and often do not generalize well to unseen parts of the state space. Combining data-driven modelling with prior analytical knowledge is an attractive alternative as the inclusion of structural knowledge into a regression model improves the model's data efficiency and physical integrity. In this article, we survey supervised regression models that combine rigid-body mechanics with data-driven modelling techniques. We analyze the different latent functions (such as kinetic energy or dissipative forces) and operators (such as differential operators and projection matrices) underlying common descriptions of rigid-body mechanics. Based on this analysis, we provide a unified view on the combination of data-driven regression models, such as neural networks and Gaussian processes, with analytical model priors. Further, we review and discuss key techniques for designing structured models such as automatic differentiation.

KEYWORDS:
Hybrid modeling, Analytical mechanics, Robotics, Machine learning

1 | INTRODUCTION

In recent decades, increasing interest has been put on the control of mobile robots and similar agile mechanical systems. Promising frameworks for the synthesis of control policies are model-predictive control [1] and model-based reinforcement learning [2]. The performance of these algorithms rely on, or at least significantly benefit from an accurate model of the mechanical system dynamics. A dynamics model $\hat{f}(x, \theta)$ seeks to minimize the error to the real dynamics function $f(x)$, writing

$$\epsilon_f(x) = f(x) - \hat{f}(x, \theta), \tag{1}$$

with model parameters θ and model inputs x. Many interesting mechanical systems possess complex non-linear dynamics and operate in large regions of their state-space. Therefore, numerous works on dynamics modelling choose the direct identification

Abbreviations: ML: Machine learning; GP: Gaussian Process; NN: Neural network; EOM: Equations of motion; ODE: Ordinary differential equation; ALM: Analytical latent modelling; ARM: Analytical (output) residual modelling; AD: Automatic differentiation; APN: Analytical parametric networks; COM: Center of mass

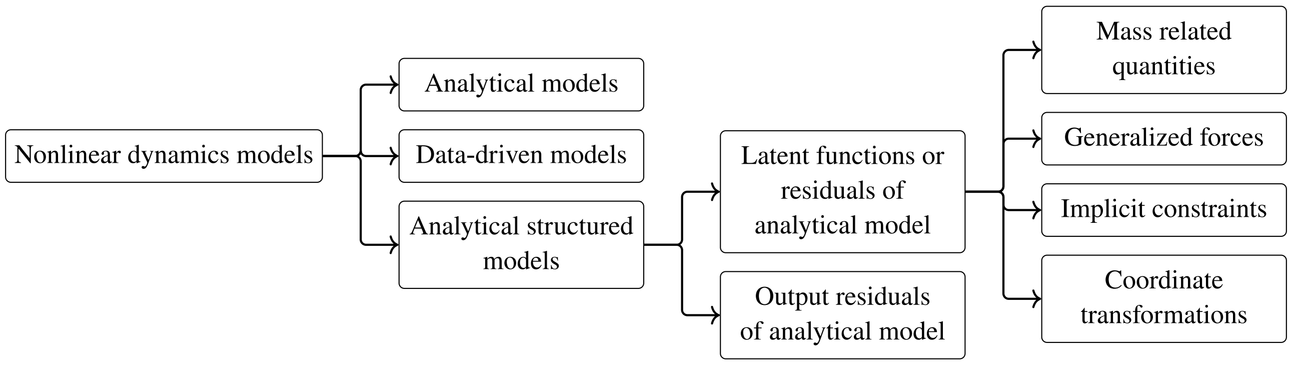
Nonlinear dynamics models
Analytical models
Data-driven models
Analytical structured models
Latent functions or residuals of analytical model
Output residuals of analytical model
Mass related quantities
Generalized forces
Implicit constraints
Coordinate transformations

FIGURE 1 Overview of different types of nonlinear dynamics models with a focus on structured modeling. The synthesis of a structured model requires several hierarchical decisions on which part of an analytical model is estimated via data-driven models.

of the system's non-linear dynamics instead of local linear approximations. For learning a system's nonlinear dynamics, several different model types have been suggested (cf. Fig. 1), namely:

- *Analytical models (white-box models)*, which consist of functional relationships motivated by first-principles.
- *Data-driven models (black-box models)*, which consist of solely data-driven models.
- *Analytical structured models (gray-box models)*, which denote combinations of analytical with data-driven models.

However, the identification of the dynamics of mechanical systems from data is oftentimes challenging because real-world systems are high-dimensional and subject to complex physical phenomena such as friction and contacts. In addition, data collection on physical systems is expensive and time-consuming. These aspects in combination with the curse of dimensionality [3, p. 190] aggravate pure data-driven modeling.

For these reasons, recent works aim at combining data-driven modeling techniques with a-priori available knowledge, which we will denote as *structured models*. A rich source for a-priori structural knowledge in numerous mechanical systems constitutes analytical rigid-body dynamics. While the bodies of a mechanical system are usually slightly elastic, it has been shown in literature that the assumption of rigid-bodies yields a useful model prior for nonlinear dynamics modeling [4, 5, 6, 7].

In this survey, we give a unified view of the state-of-the-art in structured learning using rigid-body dynamics. Structured models inherit the potential to combine the advantages of both analytical and data-driven modeling while avoiding the shortcoming of these modeling approaches when being used individually. However, a model will be rarely used in practice if its synthesis requires significant effort and its optimization is laborious. Therefore, we emphasize in this survey that recent automatic differentiation libraries such as JAX [8] and PyTorch [9] considerably simplify the synthesis of structured models and also enable GPU accelerated estimation of the model's parameters. In turn, structured modeling has the potential to emerge as one of the defining model classes that is used in the control synthesis and operation of future generations of robotic systems.

Before we present our view on structured modeling of rigid-body mechanics, we first define what we understand as dynamics functions as well as further discuss the strengths and shortcomings of analytical and data-driven modeling which are also summarized in Table 1.

Nonlinear dynamics functions of mechanical systems

The field of dynamics consists of kinematic equations describing the motion of a system via position, velocity, and acceleration, as well as kinetics, which evolves around the causal effects of generalized forces on the motion of body masses. We assume that we have a thorough understanding of the system's kinematics which yields a generalized coordinate description as well as implicit constraint equations. When it comes to structured modeling we are interested in utilizing knowledge on the causal relationships between the system's mass, impressed forces, and its motion. In most modern analytical mechanics textbooks [10, 11], the term *dynamics* is used as a direct substitute for kinetics.

The configuration of a rigid-body mechanical system is described via the generalized position, velocity, and acceleration, which are denoted by the vectors $q(t)$, $\dot{q}(t)$, and $\ddot{q}(t)$, respectively. Although these vectors depend on time t, we generally omit the time dependence for these and other variables if this is clear from the context. Further, a specific discrete-time instant is denoted by the subscript k, the subsequent discrete-time step by $k+1$. In addition, the system is subject to a control input $u(t)$ or $u_k \in \mathbb{R}^w$. The control input induces forces onto the system, which change the system's total energy. Further, we assume that the mechanical system is *fully observable*. That is, we can directly measure the state variables or, use state estimation techniques to obtain a reasonable state estimate.

The *transition dynamics* $f_\mathrm{T}(q_k, \dot{q}_k, u_k)$ denotes a function from the current state-control vector to the next state, writing

$$\begin{bmatrix} q_{k+1} \\ \dot{q}_{k+1} \end{bmatrix} = f_\mathrm{T}(q_k, \dot{q}_k, u_k). \tag{2}$$

While the transition dynamics are readily amenable to data-driven models, analytical rigid-body dynamics are usually described as ODEs. In turn, the transition dynamics can be split up into a continuous *direct dynamics* function $f_{\ddot{q}} : \mathbb{R}^{2n+w} \to \mathbb{R}^n$, that maps the state-control vector to the system's acceleration,

$$\ddot{q} = f_{\ddot{q}}(q, \dot{q}, u), \tag{3}$$

and a numerical integration method predicting the next state using $\ddot{q}$. Commonly used integration methods such as Runge-Kutta-45 predict the next state as a function of several state-acceleration vectors at different time steps [12].

Alternatively, one can rearrange the forward dynamics to obtain the system's *inverse dynamics* function $f_u : \mathbb{R}^{3n} \to \mathbb{R}^l$, which computes the (feed-forward) generalized actuation force that is required to achieve a state-acceleration vector, writing

$$Q_\mathrm{u} = f_u(q, \dot{q}, \ddot{q}). \tag{4}$$

In the case of robots with actuated joints, Q_u represent all forces that are caused by friction and actuation inside the joints. In this case, Q_u is referred to in robotics as joint torques (usually being denoted as τ). We note that the inverse dynamics mapping in (4) can be non-injective. In this case, the identification of inverse dynamics via supervised learning models requires additional mathematical techniques [13], which are not discussed in this survey.

The inverse dynamics enables us to describe which Q_u is required to steer a system on a given trajectory. However, to use a dynamics description in terms of Q_u, we need to know how the desired actuation force $Q_\mathrm{u,desired}$ is altered by Q'_u, writing

$$Q_\mathrm{u} = Q_\mathrm{u,desired} + Q'_\mathrm{u}. \tag{5}$$

We refer to the term $Q'_\mathrm{u}(x, Q_\mathrm{u,desired})$ as the *actuation dynamics* which is a complex function that can depend on nonlinear and discontinuous dissipation inside the motor, flexibilities in a gear-belt or attached shaft, free-play in an attached transmission, and internal dynamics of cascaded feedback control loops. For example, the generation of torque inside a robot arm's brush-less motors requires to transform a control error into $Q_\mathrm{u,desired}$; then $Q_\mathrm{u,desired}$ is transformed by a low-level control into the desired current; the desired current is transformed by a field-orientation controller into the motor torque; the motor torque is altered by a potential gear train as well as dissipation in the robot's joints; and finally Q_u is measurable in the joints.

To simplify the discussion, we use the term *dynamics* interchangeably for functions modeling either the transition, forward, or inverse dynamics functions of a mechanical system. To this end, we consider the dynamics of a mechanical system be described by the general function $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ where $f(x) = y$ with input vector $x \in \mathbb{R}^{n_x}$ and output vector $y \in \mathbb{R}^{n_y}$. For example, if f denotes the system's forward dynamics, then x is a state-control vector and y the system's acceleration.

Supervised regression

In this survey, we assume that a model $\hat{f}(x;\theta) = \hat{y}$ with model parameters θ shall be trained on informative data $\mathcal{D}\{\tilde{x}_k, \tilde{y}_k\}_{k=1}^N$ where N denotes the number of data points, $\tilde{x}$ denotes noisy measurements of x, and $\tilde{y}$ denotes noisy measurements of f. The term training or learning refers to the estimation of θ from data. The term supervised refers to a model being trained on input-output data pairs of the dynamics function.

1.1 | Analytical models

Analytical models are commonly used in the identification of the dynamics of mechanical systems [14, 15, 16, 17, 18, 5]. Analytical models of rigid-body dynamics are based on physical axioms and principles underlying the motion of mechanical systems, *e.g.*, Newton's axioms of motion. These axioms and principles spawn dynamics equations such as the Newton-Euler equations and the Euler-Lagrange equations. Analytic dynamics equations are a combination of numerous physically motivated functions such as coordinate-transformations, forces, and the inertia matrix which depend on physical parameters. Physical parameters are for example the mass of a rigid-body, a friction coefficient, the length of a kinematic link, or the stiffness coefficient of a spring.

TABLE 1 Pros and cons of analytical rigid-body models compared to data-driven models.

Analytical models	**Data-driven models**
+ Data-efficient	- Data-hungry
+ Human-interpretable	- Usually black-box model
+ Out-of-sample generalization	- Usually data-point interpolation
- Large prediction errors	+ Less restrictive assumptions
- Laborious derivation	+ Less prior knowledge required
∘ Deterministic	∘ Possibly probabilistic

The numerous functions that form an analytical model can usually not be observed individually which is why we refer to them as being *latent*. Notably, the latent functions and parameter estimates of analytical models yield a physical interpretation that can guarantee *out of sample generalization* (i.e., the validity of the model in regions where no data has been observed) and earns them the name white-box models. However, on real systems, physical phenomenons such as *friction*, *damping*, and *contacts* aggravate an accurate identification of an analytical dynamics model. In addition, other physical phenomenons such as *body elasticities* cannot be modeled by analytical rigid-body dynamics in the first place and therefore lead to errors. In practice, some of the parameters of an analytical model are estimated from data using linear regression or gradient-based optimization. However, the analytical model errors and the limited representative power of analytical force models can lead to physically inconsistent parameter estimates such as negative body masses [15]. Analytical models whose parameters are estimated via gradient-based optimization could be seen as parametric data-driven models which is why we refer to them as analytical parametric networks (APM). Analytical parametric networks provide profound insights for the synthesis of analytical structured models, which is why we briefly discuss these works in Section 3.2.2.

1.2 | Data-driven models

When analytical modeling is not feasible (e.g, because the physical process is unknown), or the effort required to obtain a sufficiently small prediction error appears too large, the nonlinear dynamics can be directly learned from data. The design of the data-driven model places prior assumptions on the functions it can approximate. To reduce the prediction error of these models, one needs to estimate their parameters, which is referred to as training. Parameters can be either the hyperparameters of the kernel of a non-parametric method, such as a Gaussian process (GP) or, the parameters of a parametric method such as a neural network (NN). As the same optimization methods are commonly used for training parametric models as well as GPs, we refer in what follows to hyper-parameters also as parameters. However, training hyperparameters of a kernel determines an entire population of features while training the parameters of a parametric model usually determines a function approximation.

Data-driven models contrast analytical models as most often their parameter estimates do not yield a physically insightful interpretation. Moreover, the sheer amount of parameters and interlinked latent functions in most data-driven models such as neural networks eludes human comprehension. Thus they are commonly referred to as black-box models.

One of the largest concern with the identification of dynamics via data-driven models is *sample complexity*; that is, the number of training points a data-driven model requires to successfully approximate a function. On physical systems, data is often sparse because the data collection is subject to life-time and cost constraints. While it has been proposed to identify structural knowledge directly from data [19, 20], the identification of structure requires significant amounts of data. In the next section, we detail our understanding of structure, and why it is beneficial to combine analytical structure with data-driven modeling.

1.3 | Structured models

In this survey, we interpret the term structure as either mathematical properties of functions or, alternatively, causal dependencies between functions. In the field of nonlinear system identification, a *gray-box model* denotes the combination of analytical with data-driven modeling [21, 3]. In slight contrast, a structured model denotes the combination of data-driven models with *some form of* structural prior knowledge that reduces the complexity of the model class (cf. [3, p. 192]). Such structural knowledge can be an analytical model or alternatively, the way in which different data-driven models are being connected with each other.

Therefore, a gray-box model is a structured model, but not all structured models are gray-box models. As suggested by Nelles [3], structured models can be further categorized as:

- *Hybrid structures*: The combination of different sub-models to a single structured model.[1] This model type is further divided into:
  - *Parallel model*: The outputs of all sub-models are summed up to yield the output of the hybrid model.
  - *Series model*: The sub-models are connected in series, that is the output of one sub-model forms the input to another sub-model and so forth.
  - *Parameter scheduling model*: The parameters of a sub-model are scheduled by the output of another sub-model.
- *Projection-based structures*: The input space is projected into a lower-dimensional latent space.
- *Additive structures*: The input dimensions are grouped into several sub-spaces which act as input to sub-models. The additive model's output is the sum of the sub-models' outputs.
- *Hierarchical structures*: A model is composed of parallel, series, and additive structures yielding hierarchical causal relationships between the inputs and outputs of the sub-models.
- *Input space decompositions*: The input space is decomposed into several regions that each constitutes a sub-space to a sub-model. Note that the input space is split up region-wise rather than dimension-wise as in additive structures.

Notably, analytical rigid body dynamics can yield many of the above types of structural knowledge, which makes it a versatile toolbox for structured learning.

In the field of robotics, the term structured learning usually refers to the combination of analytical mechanics with data-driven modeling [22, 23, 24]. The term structured mechanics model has also been used in robotics literature [23]. However, it is the analytical prior knowledge that structures a data-driven model. Therefore, we use the term *analytical structured model* to refer to a structured model that uses analytical rigid-body mechanics as a model prior to data-driven modelling.

A unified view on analytical structured models

The core idea of analytical structured modeling is to complement an analytical model via data-driven modeling to infer its errors $\epsilon_\mathrm{A}(x)$ from data. An analytical model $\hat{f}_\mathrm{A}(x;\theta_\mathrm{A})$ with parameters θ_A defines a hierarchical structure between several latent functions such as forces and inertias. In principle, one can directly use the analytical model to faithfully estimate θ_A by minimizing a suitable metric $\|\tilde{y} - \hat{f}_\mathrm{A}(\tilde{x};\theta_\mathrm{A})\|$. However, an analytical model is only an approximation of $f(x)$ such that

$$f(x) = \hat{f}_\mathrm{A}(x;\theta_\mathrm{A}) + \epsilon_\mathrm{A}(x;\theta_\mathrm{A}). \tag{6}$$

To reduce ϵ_A and potentially improve the estimate of θ_A, one can add a data-driven model $\hat{\epsilon}_\mathrm{A}(x;\theta_\mathrm{D})$ with parameters θ_D to the analytical model, writing

$$\hat{f}(x;\theta_\mathrm{A},\theta_\mathrm{D}) = \hat{f}_\mathrm{A}(x;\theta_\mathrm{A}) + \hat{\epsilon}_\mathrm{A}(x;\theta_\mathrm{D}). \tag{7}$$

We refer to the parallel model structure in (7) as *analytical (output) residual modeling* (ARM). Usually, the measurements obtained from the real dynamics are subject to additional measurement noise and bias $\epsilon_y(\tilde{x})$, such that a system measurement follows from $\tilde{y} = \hat{f}_\mathrm{A}(\tilde{x};\theta_\mathrm{A}) + \epsilon_\mathrm{A}(\tilde{x}) + \epsilon_y(\tilde{x})$. Therefore, the data driven model usually models the residual $\epsilon_\mathrm{A}(\tilde{x}) + \epsilon_y(\tilde{x})$ jointly. ARM forms a natural extension to analytical modeling and is conceptually easy to implement (while there can be of course practical challenges such as noise). However, if $\hat{f}_\mathrm{A}(x;\theta_\mathrm{A})$ is inaccurate, the data-efficiency of the resulting structured model suffers significantly. In what follows, we omit the arguments in the notation of a function if these are clear from context.

Consequently, the question arises on how we can improve the prediction accuracy of $\hat{f}_\mathrm{A}$ itself by using a data-driven model. An important insight for structured learning is that the cause of parts of the analytical model error ϵ_A lies hidden in the functions that form the analytical model $\hat{f}_\mathrm{A}$. For example, parts of ϵ_A can be caused by an inaccurate kinematics model or an inaccurate model for the friction forces. Therefore, instead of estimating directly ϵ_A, one can also substitute unknown latent functions or their residuals *inside* $\hat{f}_\mathrm{A}$ with a data-driven model (to be made precise in Section 3). We refer to a structured model in which data-driven models substitute/augment parts of $\hat{f}_\mathrm{A}$ as *analytical latent modeling* (ALM).

[1]The term hybrid structures/models should not be confused with hybrid systems which commonly denote the combination of continuous and discrete-time systems.

1.4 | Overview

The remainder of this article is organized as follows. Section 2 introduces the reader to the analytical mechanics of rigid-body systems. This section introduces different types of structural knowledge while emphasizing the importance of linear operators in rigid-body mechanics. While we do not aim at a comprehensive overview of rigid-body dynamics herein, Section 2 outlines important aspects of analytical mechanics that are useful for the synthesis of structured models and needed for the purpose of this survey. Section 3 discusses the current state of the art in analytical structured models. Via the discussion of the previous sections, we develop a unified view on ARM as well as ALM. Section 4 details the key mathematical techniques required to combine the most common data-driven models, namely GPs and NNs, with analytical equations. Further, we emphasize the importance of recent developments on automatic differentiation libraries for a straightforward design and training of analytical structured models. Finally, we discuss a case study of modeling the dynamics of a robot arm to illustrate the presented concepts.

2 | A GLIMPSE ON ANALYTICAL RIGID-BODY MECHANICS

In this section, we discuss common types of rigid-body dynamics equations. The equations describing rigid-body dynamics can be formulated in many ways. While each formulation yields an equivalent description of motion, they differ in the form of the resulting mathematical equations. For the sake of brevity, we limit the discussion to the Newton-Euler and Lagrange equations described in generalized coordinates. These formulations are frequently used for the description of robot dynamics [25] and the simulation of rigid-body systems [11]. Further, we illustrate how to include implicit constraint knowledge into these dynamic formulations.

A rigid-body system consists of rigid bodies that are connected by joints and which are subject to external forces. We assume that the joints and other physical mechanisms enforce $r + m$ independent constraints onto the system's motion, of which r holonomic constraints are eliminated from the description of the dynamics via a suitable choice of a generalized coordinate vector $q \in \mathbb{R}^n$ [11, p. 45-46]. In addition, there are m implicit constraint equations such that the system has $d = n - m$ degrees of freedom. The m (holonomic or non-holonomic) constraint equations are described as implicit constraint equations which are enforced via the addition of an ideal constraint force Q_I (cf. Section 2.3).

2.1 | Newton-Euler equations: Structural knowledge between forces and acceleration

The (projected) Newton-Euler equations of a general rigid-body system [11, p. 44] read

$$M\ddot{q} = Q + Q_\mathrm{I}, \tag{8}$$

with the generalized inertia matrix $M(q)$, generalized forces $Q(q, \dot{q}, t)$, and ideal constraint force $Q_\mathrm{I}(q, \dot{q}, t)$. The inverse of the inertia matrix, $M(q)^{-1}$, is a mapping from the n-dimensional force space to the n-dimensional generalized acceleration space. We assume that the generalized coordinates q are chosen such that the generalized inertia matrix $M(q)$ and its inverse are positive-definite (and hence symmetric), writing $\ddot{q}^\top M(q)\ddot{q} > 0$. The vector of generalized forces in (8) can be further split up as

$$Q = Q_C + Q_G + Q_D. \tag{9}$$

Here the inertia or fictitious force $Q_\mathrm{C}(q, \dot{q})$ denotes the coriolis and centrifugal forces. The force Q_C arises if the system's configuration is described in an accelerated frame of reference. The conservative force $Q_\mathrm{G}(q, \dot{q})$ arises from a physical potential $V(q, \dot{q})$, such as a spring or a gravitational force. The non-conservative forces $Q_\mathrm{D}(q, \dot{q}, t) = Q_\mathrm{d}(q, \dot{q}) + Q_\mathrm{u}(q, \dot{q}, t)$ denote forces that can change the system's total energy via external forces Q_d and the actuation forces Q_u, respectively. Note that Q_d is often a dissipative force, that is, it can only reduce the system's total energy.

By rearranging (8) one obtains the system's inverse dynamics as

$$Q_\mathrm{u} = M\ddot{q} - Q_\mathrm{C} - Q_\mathrm{d} - Q_\mathrm{I}. \tag{10}$$

2.2 | Lagrange equations: Structural knowledge via energy conservation

The dynamics equations presented in this section are used to build energy conservation into an analytical structured model. An introduction to Lagrangian mechanics is given in [10] and specifically for robotic systems in [25]. As we show in Section 3.4.1, one can place a data-driven model either on the Lagrangian or the entries of M.

The energy of a rigid-body system can be denoted by a Lagrangian function $\mathcal{L}(q, \dot{q})$ as the sum of the kinetic energy $T(q, \dot{q})$ and the potential energy $V(q, \dot{q})$, writing

$$\mathcal{L} = T - V = \frac{1}{2}\dot{q}^{\top} M \dot{q} - V. \tag{11}$$

The Euler-Lagrange equation of a rigid-body system's i-th dimension can be derived via the calculus of variations as

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = Q_{D,i} + Q_{I,i}. \tag{12}$$

Rewriting this equation in vector notation yields

$$\frac{d}{dt}\nabla_{\dot{q}}\mathcal{L} - \nabla_{q}\mathcal{L} = Q_{\mathrm{D}} + Q_{\mathrm{I}}, \tag{13}$$

with $(\nabla_{\dot{q}})_i = \frac{\partial}{\partial \dot{q}_i}$. The Euler-Lagrange equation is invariant to linear transformations of the Lagrangian. That is, the Lagrangian $\mathcal{L}^* = \alpha\mathcal{L} + \beta$ is also a valid Lagrangian if $\alpha \neq 0$. One can apply the chain rule to expand the *time-derivative* of the Lagrangian's partial derivative as

$$\frac{d}{dt}\nabla_{\dot{q}}\mathcal{L} = \left(\nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L}\right)\ddot{q} + \left(\nabla_{q}\nabla_{\dot{q}}^{\top}\mathcal{L}\right)\dot{q}, \tag{14}$$

with the $n \times n$ matrix $\left(\nabla_{q}\nabla_{\dot{q}}^{\top}\mathcal{L}\right)_{ij} = \frac{\partial^2 \mathcal{L}}{\partial q_i \partial \dot{q}_j}$. Therefore, one obtains a parametrization of the system's acceleration solely in terms of the Lagrangian function as

$$\ddot{q} = \left(\nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L}\right)^{-1}\left(-\left(\nabla_{q}\nabla_{\dot{q}}^{\top}\mathcal{L}\right)\dot{q} + \nabla_{q}\mathcal{L} + Q_{\mathrm{D}} + Q_{\mathrm{I}}\right). \tag{15}$$

In what follows, it is assumed that $V(q)$ only depends on q to keep the expressions concise. The previous equation can also be written in terms of M to yield an alternative description of the forward dynamics as

$$\ddot{q} = M^{-1}\left(-\nabla_{q}(\dot{q}^{\top}M)\dot{q} + \frac{1}{2}\nabla_{q}\left(\dot{q}^{T}M\dot{q}\right) - \nabla_{q}V + Q_{\mathrm{D}} + Q_{\mathrm{I}}\right), \tag{16}$$

with $M(q) = \left(\nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L}\right)$ and $Q_{\mathrm{G}} = -\nabla_{q}V(q)$. The above equations yield parametrizations of the fictitious force as

$$Q_{\mathrm{C}} = -\left(\nabla_{q}\nabla_{\dot{q}}^{\top}T\right)\dot{q} + \nabla_{q}T = -\nabla_{q}(\dot{q}^{\top}M)\dot{q} + \frac{1}{2}\nabla_{q}\left(\dot{q}^{T}M\dot{q}\right). \tag{17}$$

By rearranging both (15) and (16) one obtains expressions for the system's inverse dynamics as

$$Q_{\mathrm{u}} = \left(\nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L}\right)\ddot{q} + \left(\nabla_{q}\nabla_{\dot{q}}^{\top}\mathcal{L}\right)\dot{q} - \nabla_{q}\mathcal{L} - Q_{\mathrm{d}} - Q_{\mathrm{I}}, \tag{18}$$

$$= M\ddot{q} + \nabla_{q}(\dot{q}^{\top}M)\dot{q} - \frac{1}{2}\nabla_{q}\left(\dot{q}^{T}M\dot{q}\right) + \nabla_{q}V - Q_{\mathrm{d}} - Q_{\mathrm{I}}. \tag{19}$$

2.3 | Implicit constraint equations: Structural knowledge on the direction of motion

The presence of m implicit motion constraints applies forces Q_{I} onto the system. The forces Q_{I} ensure that the system's acceleration lies in a $d = n - m$ dimensional manifold. For example, a foot of a quadruped robot pressed onto a surface (Figure 2 , left) can only slide tangential to the surface. Also, a pneumatic-actuated robot leg (Figure 2, right) introduces kinematic loops that can be modeled via the addition of implicit constraints. These implicit constraints induce a dissipative force being part of Q_{d}, named $Q_{\mathrm{d,I}}$, which is often causally related to Q_{I}. For example, the friction force $Q_{\mathrm{d,I}}$ between a robot's foot and a surface is caused by the normal force Q_{I} that the foot applies onto the surface. It is common practice to model $Q_{\mathrm{d,I}}$ as well as Q_{I} jointly with Q_{D}. However in this section, we detail the dynamics description that expresses Q_{I} in terms of Q by using additional knowledge about the implicit constraint equations. Implicit constraint equations are usually easy to obtain a-priori and in return form a practical way to introduce structural knowledge to dynamics models. Further, these dynamics formulations allow us to add or remove implicit motion constraints to the EOMs at will which is particularly useful if a dynamical system is subject to contacts.

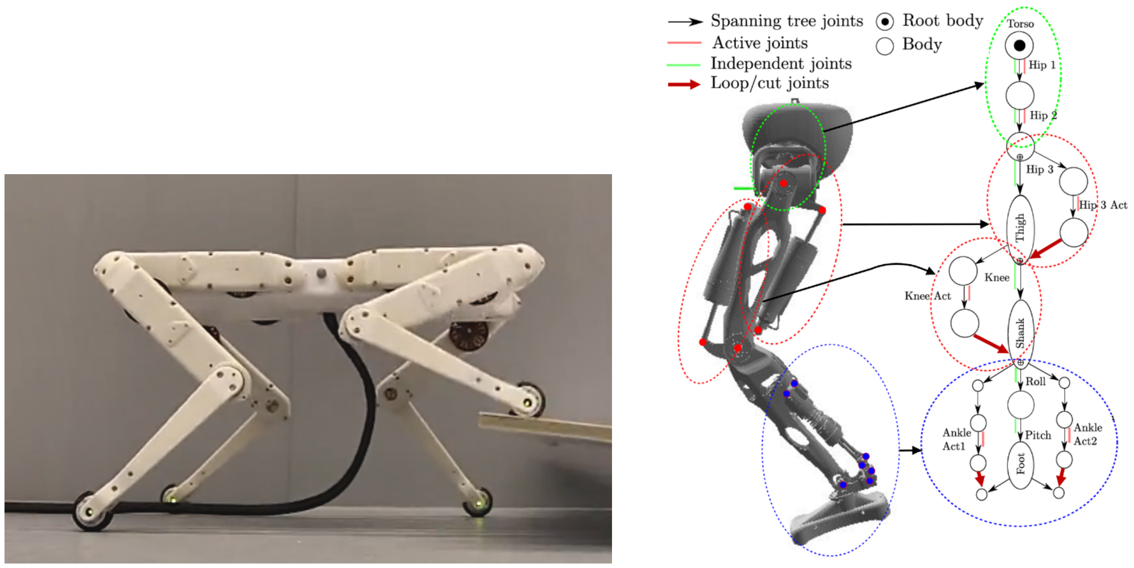

FIGURE 2 A quadruped (Left: Open Dynamic Robot, used with courtesy of [26]) is subject to contact forces both in the robot's feet. A pneumatic-actuated leg (Right: RH5 leg, adapted with courtesy of [27]) has several kinematic loops that can be modeled via implicit constraints which induce constraint forces at the cut joints.

D'Alembert's Principle

Motion constraints are algebraic constraints on the system's states [11, p. 44] such that implicit motion constraints can be expressed algebraically as $c_h(q,t) = 0$ if they are holonomic, or $c_{nh}(q,\dot{q},t) = 0$ if they are non-holonomic. While it is possible to find a transformation to a set of generalized coordinates such that the implicit holonomic constraints reduce to identities [11, 45-46], this is not possible for non-holonomic constraints. Alternatively, one can take the second time-derivative of the implicit holonomic constraints $\frac{d^2 c_h}{dt^2}$ and the first time-derivative of the non-holonomic constraints $\frac{dc_{nh}}{dt}$ and denote these derivatives jointly via the so called constraining equation

$$A\ddot{q} = b, \tag{20}$$

with $A(q,\dot{q},t) \in \mathbb{R}^{m\times n}$ and $b(q,\dot{q},t) \in \mathbb{R}^m$. Note that (A^+A) denotes an orthogonal projection into $\mathsf{R}(A^+)$ and $(I - A^+A)$ denotes an orthogonal projection into $\mathsf{N}(A)$, where $\mathsf{N}(A)$ is the null space of A, $\mathsf{R}(A^+)$ denotes the range space of A^+, and A^+ is the Moore-Penrose inverse of A [28]. As $\mathbb{R}^n = \mathsf{R}(A^T) \oplus \mathsf{N}(A)$, any $\ddot{q}$ possesses a unique orthogonal decomposition [28] as $\ddot{q} = A^+A\ddot{q} + (I - A^+A)\ddot{q}$. It is commonly assumed that the force impressed by the implicit constraints Q_I is *ideal*, that is, Q_I does no work under virtual displacements δq, writing

$$\delta q^\top Q_\mathrm{I} = 0. \tag{21}$$

Here, a virtual displacement δq denotes the difference between the current displacement at time t and a possible displacement at the *same* time t [29, p. 133]. Equation (21) is referred to as the *D'Alembert(-Lagrange)'s principle* [30, 31]. Udwadia et al. [32] extended the discussion on what constitutes a virtual displacement δq by concluding that $A(q,\dot{q},t)\delta q = 0$. This definition differs from classical formulations of D'Alembert's principle as in [33, p. 23]. With the previous definition of δq, this extension of D'Alembert's principle emphasises that

$$Q_\mathrm{I} \in \mathsf{R}(A^+). \tag{22}$$

Oftentimes an implicit constraint also induces a force $Q_{\mathrm{d,I}}$ that *does virtual work*; this force is referred to in literature as the *non-ideal* constraint force [34]. As $Q_{\mathrm{d,I}}$ is a dissipative force, it is assumed to be a part of Q_d as long as the constraints are active. Udwadia and Kalaba [35] proposed a further generalization of D'Alembert's principle which states that the part of the forces Q that does virtual work, Q', has the same direction as δq and hence

$$Q' \in \mathsf{N}(A). \tag{23}$$

As $\mathsf{R}(A^+) = \mathsf{R}(A^\top)$, D'Alemberts' principle yields a common parametrization of the ideal constraint force in terms of the Lagrange multiplier vector $\lambda(q,\dot{q},t) \in \mathbb{R}^m$ as

$$Q_\mathrm{I} = A^\top \lambda. \tag{24}$$

Gauss' principle and the Udwadia-Kalaba equation

Many different dynamic formulations for Q_I in terms of implicit constraints and the unconstrained dynamics equations have been proposed in literature. For example, Aghili [36] details several dynamics equations of implicitly holonomic constrained systems. In the following, we show the use of implicit constraint knowledge to describe an alternative dynamics formulation based on Gauss' principle. Gauss [37] observed that the acceleration caused by an ideal constraint force $\ddot{q}_I = M^{-1}Q_\mathrm{I}$ minimizes a quadratic functional

$$\arg\min_{\ddot{q}_I} G(q, \dot{q}, t) = \ddot{q}_I^\top M \ddot{q}_I. \tag{25}$$

The above equation, being referred to as *Gauss' principle of least constraint*, uniquely defines the length of the vector $\ddot{q}_\mathrm{I}$. The combination of Gauss' principle with D'Alembert's principle lead to the *Udwadia-Kalaba equation* [34, 29, 38] which provides an EOM for an implicitly constrained Lagrangian system as

$$\ddot{q} = M^{-1}\Big(Q + \underbrace{A^T(AM^{-1}A^T)^{-1}(b - AM^{-1}Q)}_{Q_\mathrm{I}}\Big). \tag{26}$$

Note that $Q_\mathrm{I} = A^T(AM^{-1}A^T)^{-1}(b - AM^{-1}Q)$ yields an explicit form for the Lagrange multiplier parametrization in terms of the constraining equation (20). Rearranging this equations yields

$$\ddot{q} = M^{-1}\left(PQ + Q_b\right), \tag{27}$$

with the weighted constraint projection $P(q, \dot{q}, t) = I_n - A^T(AM^{-1}A^T)^{-1}AM^{-1}$ and the constraining force $Q_b = A^T(AM^{-1}A^T)^{-1}b$. The matrix P is a weighted projection from the n-dimensional force space to $\mathsf{N}(A)$ such that Gauss' principle is fulfilled, where $\mathsf{N}(A)$ is a d-dimensional manifold inside $\mathbb{R}^n$.

3 | ANALYTICAL STRUCTURED LEARNING

In the following section, we propose a unified view on analytical structured modeling. For this, we leverage the fact that the dynamics descriptions presented in Section 2 consist of sums of latent vector-valued functions (*e.g.*, forces) that are multiplied with matrix-valued functions (*e.g.*, the inertia matrix). We then show that this perspective enables us to decompose and discuss the error functions inherent in an analytical model. As data-driven models are a substantial part of analytical structured modeling, we then proceed by giving a brief introduction to common data-driven dynamics models and analytical models in parametric network form. Finally, we first discuss selected literature on ARM in Section 3.3 and subsequently detail recent works on ALM in Section 3.4.

3.1 | A unified view

To understand the pros and cons of different analytical structured models, we require a thorough understanding of the cause of the analytical model errors. As shown in Section 2, the direct dynamics expressed either in Newton-Euler (8), Euler-Lagrange (13), or implicit constraint (27) formulations consist of a sum of generalized forces ($Q + Q_\mathrm{I}$) that are multiplied by $M^{-1}(q, \theta_\mathrm{A})$. In comparison, the inverse dynamics in the Newton-Euler (19) or the Euler-Lagrange (18) formulations are sums of latent functions that are transformed by $M(q, \theta_\mathrm{A})$ or simply identity matrices. To unify the discussion, we therefore assume that rigid-body dynamics equations can be written as a sum of latent vector-valued functions $\hat{f}_i(x, \theta_\mathrm{A})$ that are transformed by latent matrix-valued functions $C_i(x; \theta_\mathrm{A})$, such that the analytical model becomes

$$\hat{f}_\mathrm{A}(x; \theta_\mathrm{A}) = \sum_i C_i \hat{f}_i. \tag{28}$$

As analytical models of a mechanical system dynamics are erroneous, one can substitute (28) into (6) to obtain an expression of the system's dynamics in terms of the analytical model and its error functions, writing

$$f(x) = \epsilon_\mathrm{R} + \hat{f}^*_\mathrm{A} = \epsilon_\mathrm{R} + \sum_i \left(C_i + \epsilon^i_C\right)\left(\hat{f}_i + \epsilon_i\right), \tag{29}$$

with the ideal rigid-body dynamics model $\hat{f}^*_\mathrm{A}(x; \theta^*_\mathrm{A})$, the vector-valued error functions $\epsilon_\mathrm{R}(x)$ and $\epsilon_i(x)$, and the matrix-valued error function $\epsilon^i_C(x)$. Here, ϵ_R denotes the error between $\hat{f}^*_\mathrm{A}$ and f.

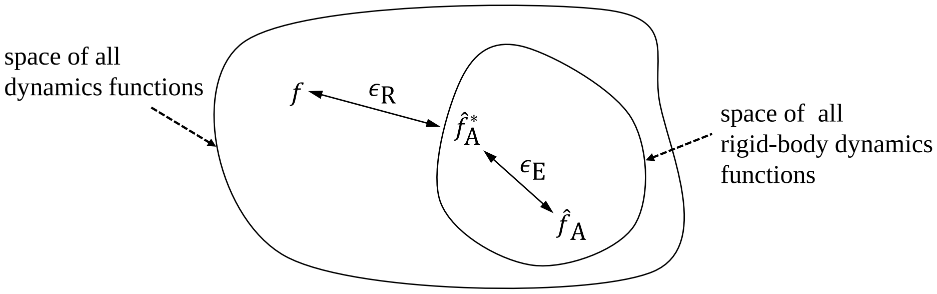
space of all dynamics functions
f
ϵ_{R}
$\hat{f}_{\mathrm{A}}^{*}$
ϵ_{E}
$\hat{f}_{\mathrm{A}}$
space of all rigid-body dynamics functions

FIGURE 3 Illustration of the errors of a rigid-body dynamics model.

Ideal models and latent error functions

Figure 3, being inspired by [39], illustrates the errors between the systems dynamics and an analytical approximation. The *ideal rigid-body dynamics model*

$$\hat{f}_{\mathrm{A}}^{*}(x;\theta_{\mathrm{A}}^{*}) = \sum_i C_i^* \hat{f}_i^* . \tag{30}$$

corresponds to the analytical model in which the asterisk symbol $(^*)$ denotes adequate choices of C_i,$\hat{f}_i$, and θ_{A} such that $\|\epsilon_{\mathrm{R}}(x)\|$ is minimized. In other words, ϵ_{R} denotes all errors that cannot be captured using rigid-body dynamics modeling. For example, ϵ_{R} can be caused by elasticities in the system's bodies or disturbances caused by attached cables. In contrast, ϵ_i and ϵ_C^i cause errors between the rigid-body dynamics model prior $\hat{f}_{\mathrm{A}}$ and the ideal rigid-body dynamics model $\hat{f}_{\mathrm{A}}^*$. If $C_i \mathrel{\widehat{=}} I$, then ϵ_C^i is a null matrix. In turn, the analytical model error can be written as

$$\epsilon_{\mathrm{A}} = \epsilon_{\mathrm{R}} + \epsilon_{\mathrm{E}} = \epsilon_{\mathrm{R}} + \sum_i \epsilon_C^i \hat{f}_i + \sum_i \epsilon_C^i \epsilon_i + \sum_i C_i \epsilon_i . \tag{31}$$

The foremost goal behind the combination of data-driven models with an analytical structured model is to reduce ϵ_{A}. Therefore, analytical structured models can be distinguished by how ϵ_{A} is reduced using data-driven modeling, namely using:

- ARM, which seeks to minimize ϵ_{A} by placing a data-driven model on the residual $\epsilon_{\mathrm{A}} + \epsilon_y$,
- ALM of latent analytical functions, in which data-driven models approximate $\hat{f}_i$ and/or C_i in (29) if the respective analytical model is inaccurate, or alternatively,
- ALM of latent residual functions, in which data-driven models approximate ϵ_C^i and/or ϵ_i in (29) hence also using the analytical functions $\hat{f}_i$ and C_i.

Using the unified view to compare direct and inverse dynamics

To illustrate the implications underlying (29), consider that a system's direct dynamics are expressed via the Newton-Euler equation (8). Then (29) reduces to

$$\ddot{q} = \epsilon_{\mathrm{R}} + \left(M^{-1} + \epsilon_{M^{-1}}\right)\left(Q + Q_{\mathrm{I}} + \epsilon_Q\right), \tag{32}$$

where the error functions in the entries of M^{-1} are denoted by $\epsilon_{M^{-1}}(q)$, and error functions in the entries of $Q + Q_{\mathrm{I}} \mathrel{\widehat{=}} \sum_i \hat{f}_i$ are denoted by $\epsilon_Q(q,\dot{q},t)$. In comparison, the system's inverse dynamics read

$$Q_{\mathrm{u}} = \epsilon_{\mathrm{R}} + \left(M + \epsilon_M\right)\ddot{q} - \left(Q_{\mathrm{C}} + Q_{\mathrm{G}} + Q_{\mathrm{d}} + Q_{\mathrm{I}} + \epsilon_{\mathrm{CGd}}\right), \tag{33}$$

with the force errors $\epsilon_{\mathrm{CGd}}(q,\dot{q},t)$. While we chose the Newton-Euler equations for the purpose of illustration, the discussion can be straightforwardly applied to other descriptions of motion.

A model deviates from the dynamics either trough model errors as in (32) and (33) or, through observation noise. The *model errors* occurring in (33) can be denoted jointly as $\epsilon_{\mathrm{R}} + \epsilon_M \ddot{q} + \epsilon_{\mathrm{CGd}}$. As these errors directly occur in the output of the inverse dynamics they can straightforwardly approximated using ARM. In comparison, direct dynamics formulations as in (32), nonlinearly transform the error ϵ_Q by $\left(M^{-1} + \epsilon_{M^{-1}}\right)$. As illustrated in Example 3.1, this can render ARM significantly more challenging for direct dynamics compared to inverse dynamics.

Ex. 3.1. Structured modeling of pendulum dynamics
The direct dynamics of an undamped-uncontrolled pendulum in terms of its angle q reads

$$\ddot{q}(q) = M^{-1}Q_G, \tag{34}$$

with the inverse of the inertia matrix $M^{-1} = 1/mL^2$ *and the gravitational torque* $Q_G = -\sin(q)Lmg$ *consisting of the gravitational acceleration g, the pendulum's mass m, and the length of the pendulum's rod L. For the sake of this illustration, assume that the estimate for the gravitational acceleration g is erroneous such that an a-priori available model reads* $\hat{Q}_g = -\sin(q)Lm(g+\epsilon_g)$ *where* ϵ_g *denotes a constant function. Even though* ϵ_g *is constant, if its propagated through the dynamics function the output prediction error becomes a nonlinear function*

$$\epsilon_{\ddot{q}}(q) = \frac{-\sin(q)}{L}\epsilon_g. \tag{35}$$

Therefore, if the model designer is confident that the inverse inertia matrix $M(q, m, L)^{-1}$ *and the kinematic dependency* $\sin(q)L$ *denote suiting parametrizations of the real system's physics, one can place a data-driven model on* ϵ_g *instead of* $\epsilon_{\ddot{q}}$*. In turn, a less complex data-driven model can be used. Alternatively, one can use the additional prior knowledge that* $g > 0$*, such that* $\hat{Q}_g = -\sin(q)Lm\hat{g}^2$ *where* $\hat{g}(x; \theta_D)$ *denotes a suitable data-driven model.*

Another significant quantity to consider is the *acceleration noise*. Estimates of $\ddot{q}$ are usually obtained via numerical time-differentiation of velocity or position measurements. In return, the measurement noise is amplified in the acceleration estimate. As it is often easier for data-driven models to approximate the comparably large noise in the model's *output* compared to the model's input, direct dynamics models can be preferred to inverse dynamics models. This is also a reason that might explain why the majority of data-driven models approximate transition dynamics (2) as state measurements are usually readily available. However, as direct dynamics formulations are usually used to compute trajectory predictions via numerical integration methods, these trajectory predictions contain an additional integration error.

Another important aspect forms the measurement of Q_u. Recall that the actuation dynamics $Q'_\text{u}(x, Q_\text{u,desired})$ denote the difference between $Q_\text{u,desired}$ and Q_u. The identification of Q'_u is therefore critical if we want to use a dynamics model for control or simulation. However, the estimation of Q_u via current measurements in electric motors is inaccurate. In return, if we learn an inverse dynamics model, the model's output error ϵ_y will also contain the difference between Q_u and its estimation. Alternatively, we can use the end-effector force to estimate Q_u. However, such estimates depend on mechanical parameters. In comparison, directly measuring Q_u through force sensors in the actuators yields accurate measurements. Albeit, sensors, such as joint torque sensors, are costly. Compared to learning inverse dynamics with ARM, learning forward dynamics with ALM has the advantage that we do not need to measure Q_u but instead approximate it directly with a data driven-model.

Combining ARM with ALM: An uncharted territory

While we separate analytical structured modeling in between the lines of ARM and ALM, these two modeling approaches can be combined if the dynamics are affected by phenomenons that are not describable by rigid-body dynamics.

As shown in Section (3.4.2), some of the recent works that apply ALM on direct dynamics equations assume that M^{-1} in (32) accurately depicts the causal map between the forces and acceleration of a system. Accordingly, the error $\epsilon_{M^{-1}}$ is only caused by a wrong estimate of θ_A. Placing a data-driven model on the error-prone parts of Q (ALM) as well as on the output residual of the structured model (ARM), potentially reduces ϵ_Q and ϵ_R. If the reduction in the former error terms causes θ_A to be closer to θ^*_A, $\epsilon_{M^{-1}}$ also reduces. Consequently, the prediction performance significantly improves. However, this is currently solely a hypothesis based on the empirical validations made in the literature discussed in Section 3.4. To which extend modeling of direct dynamics benefits from a combination of ALM and ARM requires further future analysis.

3.2 | Data-driven dynamics models and parametric analytical networks

In this section, we briefly discuss solely data-driven dynamics models as these models are important for structured modeling. Further, we discuss works that learn the parameters of analytical models using gradient-based optimization. The insights about analytical models are useful for improving future generations of analytical structured models.

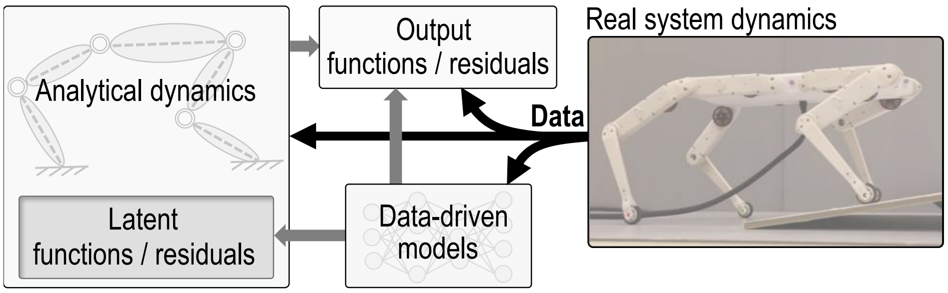

FIGURE 4 Schematic of structured dynamics models. Data-driven models can approximate the output error of the analytical model (ARM) and/or reduce latent errors inside the analytical model (ALM). Image with courtesy of [26].

3.2.1 | Data-driven dynamics models

Common data-driven models used for the identification of dynamics are (Bayesian) linear regression, neural networks (NNs), and Gaussian processes (GPs). Most works that learn dynamics function using solely data-driven models use transition dynamics or inverse transition dynamics [40]. Nguyen-Tuong and Peters [40] surveys these approaches for robot control. In the following, we briefly discuss NNs and GPs as examples of data-driven models being used for analytical structured learning.

NNs achieved breakthroughs in big-data domains such as computer vision [41], the game of go [42], and control of computer games [43, 44]. NNs have also been deployed for the inference of physical system dynamic's [45, 46, 47, 48]. Notably, NNs have learned successfully transition dynamics and control-policies on contact rich domains [49, 50]. Remarkably, Nagabandi et al. [50] demonstrated that NNs enable the inference of specific control policies for handling objects inside a robotic hand. However, in [50] the training of a control policy for a specific task-object combination requires several hours of data.

In comparison, GPs are non-parametric and probabilistic models that define a normal distribution over functions. GPs provide a measure of uncertainty of the estimation result in form of their posterior variance. In addition, GPs convert Bayesian inference into numerically efficient linear algebraic equations. An introduction to multivariate GP regression is given in [51]. In what follows, we denote a multivariate GP as $\hat{f} \sim \mathcal{GP}\left(\mathbf{m}(x), \mathbf{k}(x, x')\right)$ with the vector-valued mean function $\mathbf{m}(x)$ and matrix-valued kernel $\mathbf{k}(x, x')$. Deisenroth and Rasmussen [52] learned the system's transition dynamics with a GP. This GP dynamics model was then used for nonlinear control of real mechanical systems with compared to NNs small amounts of data. One key take away of [52] has been the usage of a probabilistic model which allows propagating uncertainty in the observed state-actions through the system dynamics. In turn, this approach improved significantly the robustness of a control policy trained on the GP dynamics model. Other applications of dynamics modeling with GPs for control includes [4, 53, 54, 55, 56, 57, 58]. The main disadvantage of GPs is their computational complexity, which typically scales cubically with the number of data points. Even though their computational complexity can be reduced using sparse GPs [59], the computational effort required to work with such models is considerably larger than using NNs.

3.2.2 | Analytical parametric networks

Recent works propose to formulate analytical models as analytical parametric networks (APN) that are trained with gradient-based optimization methods. While technically these kinds of models are analytical models, the techniques used are closely related to techniques from data-driven modeling which in return blurs the line between analytical and solely data-driven modeling.

Ledezma and Haddadin [7] reformulated the inverse dynamics of a robot arm such that both the kinematic and dynamic parameters can be estimated via gradient-based optimization. In this approach, the inverse dynamics of the robot arm are separated into a kinematic and dynamic network which allows to estimate the kinematic parameters before estimating the dynamic parameters. This approach has been extended by Ledezma and Haddadin [18] towards the identification of inverse dynamics of humanoid robots.

Similar to [7], Sutanto et al. [5] described a recursive formulation of the Newton-Euler inverse dynamics as a differentiable computational graph. The dynamics parameters are estimated via automatic differentiation. In this work, the authors placed special emphasis on the incorporation of additional structural knowledge contained in the mass parameters as discussed in [16]. Traversaro et al. [16] show that not every positive-definite matrix constitutes a physically plausible inertia matrix as the rotational

TABLE 2 Summary of works on analytical structured models that are detailed in this survey. We use the abbreviation DD for direct dynamics and ID for inverse dynamics. The entries in rightmost corner link to code repositories that have been submitted alongside the publications.

	Publication	Year	Dynamics	Data-driven model type	Data-driven approximation	Optimization library
APN	Ledezma et al. [7]	2017	ID [60]	×	×	*fmincon* (Matlab)
	Ledezma et al. [18]	2018	ID [60]	×	×	*fmincon* (Matlab)
	Sutanto et al. [5]	2020	DD [61]	×	×	PyTorch (Python)
ARM	Nguyen-Tong et al. [4]	2010	ID	GP	ϵ_{A}	×
	De La Cruz et al. [62]	2011	ID	LWPR	ϵ_{A}	×
	Um et al. [63]	2014	ID	GP	ϵ_{A}	×
	Grandia et al. [64]	2018	ID	GP/LWPR	$\tilde{\epsilon}_u$	GPy (Python)
ALM	Cheng et al. [65]	2016	ID	GP	$\mathcal{L}$	×
	Geist et al. [24]	2020	DD	GP	Q, Q_{I}	scikit-learn (Python)
	Hwangbo et al. [6]	2019	DD [66]	NN	Q_{u}	×
	Greydanus et al. [67]	2019	DD	NN	$\mathcal{H}$	PyTorch (Python)
	Lutter et al. [22]	2019	ID/DD	NN	M, Q_{G}	×
	Lutter et al. [68]	2019	ID/DD	NN	M, V	×
	Gupta et al. [23]	2020	ID/DD	NN	M, V, B, Q_{d}	PyTorch (Python) / Flux (Julia)
	Lutter et al. [69]	2020	DD [70]	NN	Q_{u}	×
	Toth et al. [71]	2020	DD	NN	$\mathcal{H}$	×
	Cranmer et al. [72]	2020	DD	NN	$\mathcal{L}$	JAX (Python)

inertia matrix must also fulfill *triangular inequalities* with respect to the principal moments of inertia. These insights led Sutanto et al. [5] to a parametrization of the rotational inertia matrix of each body respecting triangular inequalities. It should be noted that the experiments detailed in [5] show similar training results for the models with parametrization of the rotational inertia matrix either solely in terms of positive definiteness or by taking the triangular inequality property into account.

3.3 | Analytical output residual modeling

In this section, we discuss selected literature on ARM. In practice, in ARM a data-driven model simply approximates the analytical residual function $\epsilon_{\mathrm{A}} + \epsilon_y$.

From linear-regression of analytical models to semi-parametric ARM models

Historically, ARM originated from the need for accurate dynamic models on robot arms. Here, Siciliano et al. [25, p. 280] referrers to [14] as the standard approach used for the identification of the dynamics parameters of robot arms. Atkeson et al. [14] leverage the *linearity* of the (rigid) robot arm's inverse dynamics with respect to the dynamical parameters [25, p. 259], writing

$$Q_{\mathrm{u}} = \Phi(q, \dot{q}, \ddot{q})\theta_{\mathrm{A}}. \tag{36}$$

The least squares estimate of θ_{A} is obtained as

$$\hat{\theta}_{\mathrm{A}} = (\Phi^{\top}\Phi)^{-1}\Phi^{\top}\tilde{Q}_u, \tag{37}$$

with measurements of Q_{u} being denoted as $\tilde{Q}_u$ and $(\Phi^{\top}\Phi)^{-1}\Phi^{\top}$ being the left pseudo-inverse matrix of Φ. Some of these parameters denote linear combinations of dynamical parameters [25, p. 280]. However, the least-squares approach requires a good prior model of the kinematic parameters and dynamic parameters (masses and inertias) acting on the systems. Therefore, Nguyen-Tuong and Peters [4] proposed to combine (37) with GP regression resulting in a semi-parametric or fully-parametric structured model. The semi-parametric modeling approach simply approximates the analytical model's residual via a zero-mean GP, writing $\hat{\epsilon}_{\mathrm{A}} \sim \mathcal{GP}\left(0, \mathbf{k}(x, x')\right)$ such that

$$\hat{Q}_u \sim \Phi(x)\hat{\theta}_{\mathrm{A}} + \mathcal{GP}\left(0, \mathbf{k}(x, x')\right) = \mathcal{GP}\left(\Phi(x)\hat{\theta}_{\mathrm{A}}, \mathbf{k}(x, x')\right), \tag{38}$$

with $\mathbf{k}(x, x')$ denoting a diagonal matrix-valued kernel function. A Gaussian distributed estimate of the system's dynamic parameters $\hat{\theta}_A$ can then be inferred via the posterior distribution of (38), writing $\hat{Q}_u|\mathcal{D}$ [73, p. 27-29].

The second model proposed in [4] uses the kernel trick to obtain an analytical kernel of the inverse dynamics', writing

$$\mathbf{k}_A(x, x') = \Phi^\top W \Phi + \Sigma_y, \tag{39}$$

with Σ_y denoting a diagonal matrix of observation noise variances and W being a diagonal matrix denoting the prior variance on θ_p. An algorithm that is defined solely in terms of inner products in input space is lifted by the kernel trick into feature space [73, p. 12]. The analytical kernel GP can then be combined with $\hat{\epsilon}_A \sim \mathcal{GP}\left(0, k(x, x')\right)$ to yield

$$\hat{Q}_u = \mathcal{GP}\left(0, k_A(x, x') + k(x, x')\right). \tag{40}$$

Nguyen-Tuong and Peters [4] showed that (38) and (40) achieved comparable prediction accuracy on a real robot arm while (38) was slightly faster in computing predictions. Similar to [4], De La Cruz et al. [62] combined prior analytical knowledge of a robot arm's inverse dynamics with locally weighted parametric regression (LWPR) such that its receptive field is a first-order approximation of the analytical model. De La Cruz et al. [62] assumed the analytical parameters as fixed. Other semi-parametric models for learning a robot arms inverse dynamics are found in [74].

Combining a recursive Newton-Euler formulation with GP regression

Um et al. [63] extended the semi-parametric model proposed in [4]. In this work, the authors emphasize that for general kinematic trees – such as robot arms – one can obtain the system's inverse dynamics in form of a recursive formulation of the Newton-Euler equations [25]. The model assumes an accurate prior analytical model whose parameters are assumed known. The proposed modeling procedure splits into a forward and backward pass computation scheme. First, via assumed knowledge of the kinematic and inertial parameters, the joint velocities as well $\hat{Q}_C$ and $\hat{Q}_g$ at each joint are computed in a forward pass through the kinematic tree. Secondly, in a backward pass, the joint torques residuals of each joint are computed using the respective joint velocity as well as the generalized force acting on the parent joints. The generalized force acting on the parent joints is itself a prediction by another GP. To pass knowledge recursively from joint to joint as well as using only a subset of relevant states as GP inputs are both important propositions.

Using kinematics to derive a contact-invariant formulation of errors

Grandia et al. [64] learned the residual of a quadruped robot's inverse dynamics in a formulation that stays invariant under *changes in the contact configuration*. Here, if the point-feet of the quadruped are pressed onto the surface, constraints are activated which are expressed via (20). As first step, the authors eliminate Q_I from the quadrupeds dynamics using D'Alemberts principle (22) such that

$$(I - A^+A)\hat{\epsilon}_A = (I - A^+A)(Q_d + Q_u + Q_C - M\ddot{q}). \tag{41}$$

Secondly, while the constraint is active, every part of a force pointing in $\mathsf{R}(A^+)$ causes a reaction force Q_I which in return can cause a jump in *every* component of $\ddot{q}$. As learning the jump in $\ddot{q}$ is difficult, Grandia et al. [64] use a coordinate transformation J_u to express the force error via

$$\epsilon_A = A^\top \tilde{\epsilon}_c + J_u \tilde{\epsilon}_u \tag{42}$$

with $A^\top \tilde{\epsilon}_c \in \mathsf{R}(A^+)$ and $J_u \tilde{\epsilon}_u \in \mathsf{N}(A)$. A careful choice of J_u ensures that the activation of constraints only changes the error $\tilde{\epsilon}_c$. Therefore, with (41) one obtains $(I - A^+A)\hat{\epsilon}_A = (I - A^+A)J_u\tilde{\epsilon}_u$ and in return a *constraint invariant formulation* of the force error as

$$\tilde{\epsilon}_u = \left((I - A^+A)J_u\right)^+ (I - A^+A)(Q_d + Q_u + Q_C - M\ddot{q}). \tag{43}$$

The authors approximated $\tilde{\epsilon}_u$ using LWPR as well as GP regression.

3.4 | Analytical latent modeling

In analytical models, it is often known which of its latent functions are prone to modeling errors. ALM seeks to reduce the error of an analytical model by placing data-driven models on the unknown latent functions of an analytical model. Further, ALM allows to incorporate prior knowledge on the mathematical properties of an analytical latent function into the design of its data-driven approximation. In the following, we detail different works on ALM. These works differ in which part of the analytical model is approximated with a data-driven model, namely: *(i)* The entries of M or M^{-1} or alternatively the Lagrangian function, *(ii)* the entries of Q_u, or *(iii)* the entries of Q which are transformed using constraint knowledge.

3.4.1 | Latent modeling using energy conservation

The generalized inertia matrix M is of great significance for rigid-body dynamics modeling. The fictitious force Q_{C} can be derived in terms of M, see (17). Moreover, if Q_{G} denotes the gravitational force then this force is also a function of the mass parameters. Furthermore, in forward dynamics, M^{-1} is multiplied with $Q + Q_{\mathrm{I}}$ while in inverse dynamics the inertial force $M\ddot{q}$ has a significant impact on the final estimation results. In the works presented in the previous section, it is a common assumption that the inertia matrix and even its parameters are known a-priori. However, this can lead to large errors in ϵ_M / $\epsilon_{M^{-1}}$ and the part of ϵ_Q / ϵ_{CGd} that is caused by Q_{C}. Therefore, recent works propose to approximate ϵ_M / $\epsilon_{M^{-1}}$ via a data-driven model by either directly modeling the entries of M or by parametrization of the inertia matrix in terms of a Lagrangian. The works on Langrangian and Hamiltonian NN were inspired by the seminal work of Chen et al. [75] on neural differential equations.

Learning the Lagrangian

Often analytical parametrizations of M^{-1} and F_C are either not available or the effort required obtaining these is considered too large. Instead, one can express the system's dynamics in terms of the Lagrangian function $\mathcal{L}$. If the Lagrangian is not explicitly time-dependent one obtains an expression for the forward-dynamics in (15) and for the inverse-dynamics in (18). One of the first works that modeled the Lagrangian function via a GP is [65]. Cheng et al. [65] placed a GP prior on $\mathcal{L}$ writing $\hat{\mathcal{L}} \sim \mathcal{GP}(0, k(x, x'))$ and then transformed the GP prior by the operators in (18) to obtain a structured model for the inverse dynamics equation of a conservative system. However, it is currently not clear how to insert such an $\hat{\mathcal{L}}$ into the forward dynamics equation (15) as efficient multi-output GP regression requires that $\hat{\mathcal{L}}$ is solely linearly transformed. In comparison, Cranmer et al. [72] models $\mathcal{L}$ via a NN. In return, the authors obtain structured models both for (15) as well as (18).

Learning the Hamiltonian

Unlike the Newtonian and Lagrangian formulations of classical mechanics, Hamiltonian mechanics is rarely used for describing the motion of rigid-body systems. Yet, Hamiltonian dynamics is of utmost importance in other branches of mechanics such as quantum mechanics, celestial mechanic, and thermodynamics (cf. [67]). Hamiltonian mechanics is a reformulation of classical mechanics using the Legendre transform into $2n$ first-order ODEs in terms of position coordinates $q \in \mathbb{R}^n$ and a canonical impulse $p \in \mathbb{R}^n$, writing

$$\dot{q}_i = \frac{\partial \mathcal{H}}{\partial p_i}, \qquad \dot{p}_i = -\frac{\partial \mathcal{H}}{\partial q_i}, \tag{44}$$

with $\mathcal{H}(q, p, t)$, $\mathcal{H} \; :\in \; \mathbb{R}^{2n} \to \mathbb{R}$ denoting the Hamiltonian function. Similar to the Lagrangian NN, Greydanus et al. [67] parameterized the Hamiltonian in the above equation via a NN. This model was extended by Toth et al. [71] using a generative NN structure which enables the inference of Hamiltonian dynamics from high-dimensional observations (such as images).

Learning the inertia matrix

As shown in (11), the kinetic energy of a rigid-body system is described in terms of the generalized inertia matrix M. Thereby, the forward dynamics (16) as well as inverse dynamics (19) can be denoted in terms of M instead of $\mathcal{L}$. However, it is inexpedient to directly model the function entries of M using a data-driven model. As also discussed by [16], M must be positive definite. Therefore, Lutter et al. [22] proposed a parametrization of M in terms of a lower triangular matrix $L(q, \dot{q})$ such that $\hat{M} = \hat{L}(q, \dot{q})\hat{L}(q, \dot{q})^T$ ensures that $\hat{M}$ is symmetric. In addition, the diagonal of $\hat{L}(q, \dot{q})$ is enforced to be positive such that all eigenvalues of $\hat{M}$ are positive (cf. Section 3.2.2). To do so, the output layer of the NN that forms the diagonal entries of $\hat{L}(q, \dot{q})$ uses a non-negative activation function such as ReLU or Softplus onto which a small positive number is added to prevent numerical instabilities. Lutter et al. [22] modeled the potential forces Q_{G} as well as non-conservative forces F_D jointly via a NN. The authors named the resulting structured model a *Deep Lagrangian Neural Network (DELAN)*. Lutter et al. [68] showed that DELAN can be used for the energy-based control of a Furuta pendulum in which they used the fact that the conservative force can be written in terms of a NN parametrization of the potential energy function $\hat{V}(q)$, writing $Q_{\mathrm{G}} = -\nabla_q \hat{V}(q)$.

Gupta et al. [23] extended DELAN by leveraging that the control force is affine in the control signal u, writing $F_u = B(q)u$. Gupta et al. [23] added NN parametrizations of $B(q)$ and $V(q)$, such that the deep Lagrangian network becomes

$$\ddot{q} = (\hat{L}\hat{L}^T)^{-1} \left(-\nabla_q (\dot{q}^\top \hat{L}\hat{L}^T)\dot{q} + \frac{1}{2} \left(\nabla_q \left(\dot{q}^T \hat{L}\hat{L}^T \dot{q} \right) \right)^T - \nabla_q \hat{V}(q) + \hat{B}(q)u + \hat{F}_d \right). \tag{45}$$

3.4.2 | Latent modeling of joint torques

For many analytical descriptions of the system's dynamics, one can assume that some of the analytical latent functions form better approximations of the real physics than others. For example, for a robot arm, one can argue that the inertia matrix M, the fictitious force Q_C, and gravitational force Q_G form good parametrizations of the respective physics. The parameters θ_A of these analytical functions are most likely unknown but can be estimated alongside the parameters of a data-driven model. In this case, the system would still respect energy conservation with respect to the model's energy $\hat{E}(q, \dot{q}; \theta_\mathrm{A}) = \frac{1}{2}\dot{q}^T \hat{M}(q; \theta_\mathrm{A})\dot{q} + \hat{V}(q, \dot{q}; \theta_\mathrm{A})$ similarly to the structured Lagrangian models. Under this assumption, the majority of the errors ϵ_Q / ϵ_{CGd} in (32) and (33) stem from an inaccurate description of the joint torques Q_u. The following works learn the dynamics of robots with motor torques being applied inside the robot's joints.

Direct identification of joint torques and combination with a contact model

Hwangbo et al. [6] identified the joint torques Q_u induced by a quadruped's electric motors via a NN *a-priori*. The authors used a simple control scheme to let a quadruped trot and meanwhile measured Q_u using joint-torque sensors as well as position errors and velocities. Then a NN was trained to predict Q_u given a sequence of position errors and velocities. In this manner, the authors learned the complete mapping of Q_u including complex interlaced control routines of the motors (PD torque control, PID current control, field-oriented control) as well as transmission and friction disturbances.

Afterward, the trained NN joint torque model is combined with a rigid-body simulation [66]. The simulator uses a hard contact model that respects Coulomb friction cone constraints. Then, a NN based reinforcement learning algorithm was trained to control the quadruped in simulation. Notably, the kinematic and mass parameters of the analytical model were randomly initialized to increase the robustness of the control policy during training.

The fact that the trained control policy achieved impressive results on the real quadruped, indicates that the gap between modeled and real dynamics can be bridged via randomization of physical parameters (body length and mass) of a good analytical model in combination with a prior identification of Q_u. This work was further extended by Lee et al. [76] who trained an unprecedented robust control policy for a quadruped robot traversing challenging terrain.

Modeling of joint torques inside forward dynamics

Lutter et al. [69] (being the authors of DELAN) combined Newton-Euler dynamics in Lie Algebra form [70] with a NN parametrization of Q_u. The authors compared several models for Q_u, with f_NN denoting a NN model, namely

$$\text{Viscous: } \hat{Q}_\mathrm{u} \mathrel{\hat{=}} Q_\mathrm{u,desired} - \theta_v \dot{q}, \tag{46}$$

$$\text{Stribeck: } \hat{Q}_\mathrm{u} \mathrel{\hat{=}} Q_\mathrm{u,desired} - \mathrm{sign}(\dot{q}) \left(f_s + f_d \exp\left(-\theta_s \dot{q}^2\right)\right) - \theta_v \dot{q}, \tag{47}$$

$$\text{NN Friction: } \hat{Q}_\mathrm{u} \mathrel{\hat{=}} Q_\mathrm{u,desired} - \mathrm{sign}(\dot{q}) \left\| f_\mathrm{NN}(q, \dot{q}) \right\|_1, \tag{48}$$

$$\text{NN Residual: } \hat{Q}_\mathrm{u} \mathrel{\hat{=}} Q_\mathrm{u,desired} - f_\mathrm{NN}(q, \dot{q}), \tag{49}$$

$$\text{FF-NN: } \hat{Q}_\mathrm{u} \mathrel{\hat{=}} f_\mathrm{NN}(Q_\mathrm{u,desired}, q, \dot{q}). \tag{50}$$

Equations (46),(47), and (48) are guaranteed to be solely *dissipative*, while the more classical NN parametrization in (49) and (50) do not respect energy dissipativity. However, the first three models assume that the internal motor control routines do not cause an overshoot such that the real Q_u is actually larger than $Q_\mathrm{u,desired}$.

The different structured direct dynamics models were trained on data from simulated and real pendulums. Additionally, these models where compared to NN black-box modeling as well as the linear regression model denoted in (36) and (37) (cf. [14]). The training results show that a random initialization of the link parameters compares similarly to having a good prior knowledge of the link parameters. This indicates that it is possible to learn analytical and NN parameters *jointly* inside a structured model. Further, the joint torque models which enforce dissipativity of $\hat{Q}_\mathrm{u}$ gave significantly better long-term predictions. The long-term predictions were computed by feeding the models' acceleration predictions to a Runge-Kutta-4 solver.

3.4.3 | Latent modeling using implicit constraint knowledge

For some mechanical systems, such as a robot arm whose end-effector touches a surface or a quadruped robot walking over terrain, it can be desirable to identify the force acting in an implicit constraint directly from data. As detailed in Section 2.3, motion constraints induced by surfaces and wheels can be formulated as implicit constraint equations which impress an ideal force Q_I and a non-ideal force $Q_\mathrm{d,I}$. While these forces can be modeled jointly with Q_d, the constraining equation (20) allows to place additional prior knowledge onto the direction of the force vectors.

To the best of our knowledge, Geist and Trimpe [24] were the first that proposed the usage of constrained knowledge for ALM. Here, the authors assumed that an analytical parametrization of M^{-1}, and the terms of the constraining equation A, and b is given. The parameters of these analytical functions are estimated alongside the data-driven-model's parameters. In [24], a GP prior is placed onto $(M^{-1}Q) \sim \mathcal{GP}(0, k(x, x'))$ inside the Udwadia-Kalaba equation (27), that is, the acceleration that would be caused by Q if the constraints were absent. However, it is more straightforward to place a prior on Q rather than its acceleration $M^{-1}Q$. With this small adaptation, the model obtained in [24] reads

$$\hat{\ddot{q}} \sim \mathcal{GP}(M^{-1}Q_b + (M^{-1}P)m_Q,\ (M^{-1}P)k_Q(M^{-1}P)^\top), \tag{51}$$

with analytical mean function $m_Q(x)$ and kernel function $k_Q(x, x')$. Here, one can include analytical prior knowledge of Q_G and Q_C via the analytical mean function, writing $m_Q = Q_\mathrm{G}+Q_\mathrm{C}+Q_\mathrm{u}$. If such a prior mean function is chosen, $k_Q(x, x')$ models Q_d as well as the residual of m_Q. While the combination of the Udwadia-Kalaba equation with a parametric model is straightforward, using a GP has several advantages. For example, one can denote the joint distribution between $\ddot{q}$ and Q, writing

$$\begin{bmatrix} \hat{Q} \\ \hat{\ddot{q}} \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} m_Q \\ M^{-1}Q_b + (M^{-1}P)m_Q \end{bmatrix}, \begin{bmatrix} k_Q & k_Q(M^{-1}P)^\top \\ (M^{-1}P)k_Q & (M^{-1}P)k_Q(M^{-1}P)^\top \end{bmatrix} \right). \tag{52}$$

This allows to infer $\hat{Q}$ from observations of $\ddot{q}$. Secondly, in GP^2 one can learn on one constraint configuration $\{A, b\}$ and then change to a different constraint configuration $\{A', b'\}$ without the need for retraining if both constraints induce the same dissipative force $Q_{\mathrm{d,I}}$ inside the constraint.

4 | KEY TECHNIQUES FOR ANALYTICAL STRUCTURED MODELING

In order to design an analytical structured model, the following steps are required:

1. Derive an rigid-body dynamics model $\hat{f}_\mathrm{A}(x; \theta_\mathrm{A})$.
2. Obtain a structured model $\hat{f}(x, \theta)$, by the combination of data-driven models with the analytical model.
3. Collect informative data and estimate the parameters of the structured model.

If the parameters of an analytical model are assumed to be known, one can simply obtain an ARM model by learning $\epsilon_\mathrm{A} + \epsilon_y$ using a data-driven model. However, ARM becomes significantly more challenging if θ_A is to be learned jointly with θ_D. This case is particularly interesting as reducing ϵ_A potentially improves the estimate of θ_A.

In this section, we detail key techniques for the design and training of analytical structured models, in which θ_A and θ_D are estimated jointly as $\theta = \{\theta_\mathrm{A}, \theta_\mathrm{D}\}$. The first key technique is the optimization algorithm itself. Note that, all works in Table 2 use gradient-based optimization. Therefore, in Section 4.1, we discuss gradient-based optimization and illustrate why automatic differentiation is particularly useful to compute the gradient of structured models. By discussing the optimization method, we see which requirement the optimization poses onto the design of an analytical structured model. Subsequently, we detail in Section 4.2 important aspects that must be considered when combining analytical models with data-driven models.

4.1 | Gradient-based optimization

Analytical structured models are used if standard analytical or data-driven approaches yield unsatisfactory predictions. Usually, this occurs if the system is high-dimensional and subject to nonlinear physical phenomenons. Therefore, most works utilizing analytical structured models require the training of high-dimensional models on large datasets. [2] Subsequently, recent works on analytical structured learning predominantly use gradient-based optimization algorithms to compute the model's parameters.

Gradient-based optimization forms a cornerstone of solely data-driven modeling [77] and constitutes one of the most common nonlinear local optimization techniques [3, p. 90]. Table 2 summarizes the literature presented in this survey that utilizes gradient-based optimization. Gradient-based optimization techniques require the computation of the partial derivative $\nabla_\theta \ell \mathrel{\widehat{=}} \frac{\partial \ell}{\partial \theta}$ of an objective function $\ell(\hat{f}(\tilde{x}; \theta), \tilde{y}) \mathrel{\widehat{=}} \ell(\theta)$, $\ell \in \mathbb{R}$. Often it is useful to include higher-order derivatives of the objective function in

[2] A dynamics model of a quadruped robot usually has more than 14 output dimensions (*e.g.*, Six DOF for the floating base plus two DOF per leg). In this scenario, a large dataset denotes just several thousands of points.

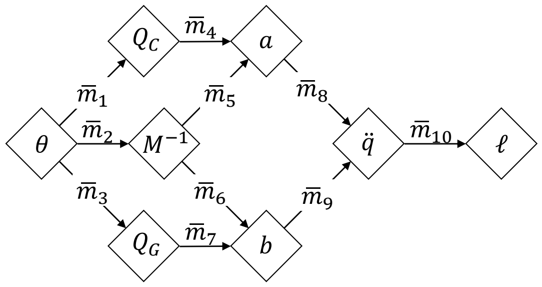

FIGURE 5 Graph illustrating backward mode automatic differentiation of the objective function of a forward dynamics model.

the parameter update rule. As gradient-based optimization is a local optimization technique and $\ell(\theta)$ is usually non-linear, it is indispensable to restart the optimization several times with random initialization of θ, keeping only the best optimization result. In addition, as models based on NN have a large number of parameters, the gradient update is computed only for a randomly selected batch of data. This optimization procedure is called stochastic gradient-descent in machine learning literature.

A typically used objective function for NN models is the L_2 loss function with an L_2 regularization term on the network's weights. For GP models the objective function is commonly chosen to be the logarithmic likelihood function. Both of these objective functions denote forward functions mapping θ to a cost $\ell(\theta)$.

As detailed by Baydin et al. [78], gradients of a forward function can be obtained via:

- *Analytical derivation and implementation*, which is time-consuming and error-prone.
- *Numerical differentiation*, which is inaccurate due to round-off and truncation errors and scales poorly with the size of θ.
- *Symbolic differentiation* yielding functions for the analytical gradient. However, these expressions are also closed-form which hinders GPU acceleration and quickly become cryptic due to an "expression swell" [79].
- *Automatic differentiation*, which computes accurate gradients, is considerably faster than the aforementioned methods, and thanks to steady improvements in the usability of optimization libraries is straightforward to use with ALM.

Structured learning with automatic differentiation

At the core of automatic differentiation (AD), which is also called algorithmic differentiation or ”autodiff“, lies the insight that forward functions are compositions of elementary operations whose derivatives are known. In return, the derivative of a function can be constructed using the elementary operators' gradient expressions with the chain rule of differentiation. AD computes a numerical value of the derivative of a computational graph with branching, recursions, loops, and procedure calls [78].

The two basic forms of AD are forward-mode AD and reverse-mode AD. In what follows, we denote the objective function as $\ell(\theta) = \ell(c(b(a(\theta))))$ where $a(\theta)$, $b(a)$, and $c(b)$ are functions of appropriate size. Then, in *forward-mode AD*, the gradient of a function is obtained via forward accumulation, *e.g.*, writing $\nabla_\theta \ell(\theta) = \nabla_c \ell(\nabla_b c(\nabla_a b \nabla_\theta a))$ where $\nabla_\theta b = \nabla_a b \nabla_\theta a$ denotes a Jacobian matrix.

Alternatively, *reverse-mode AD* being also referred to as ”backprop“ in machine learning literature, computes gradients via reverse accumulation, *e.g.*, writing $\nabla_\theta \ell(\theta) = ((\nabla_c \ell \nabla_b c)\nabla_a b)\nabla_\theta a$. For functions $\ell : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ with $n_1 >> n_2$ reverse mode AD is preferred as it requires less operation counts for the computation of vector-Jacobian products [78]. However, reverse-mode AD can have increased memory requirements compared to forward-mode AD. Many automatic differentiation packages allow to combine forward and reverse accumulation. In the following example, we illustrate reverse-mode AD for ALM.

Ex. 4.1. Computing the gradient of a structured model with reverse-mode AD

Assume that the dynamics function of a mechanical system is modeled via a structured analytical model as

$$\hat{\ddot{q}}(x;\theta) = \underbrace{M^{-1}(x;\theta)\hat{Q}_C(x;\theta)}_{a} + \underbrace{M^{-1}(x;\theta)\hat{Q}_G(x;\theta)}_{b}, \tag{53}$$

in which $\hat{Q}_G(x;\theta)$ and $\hat{Q}_C(x;\theta)$ denote either analytical and/or data-driven models, and intermediate function expressions are abbreviated as $a = M^{-1}\hat{Q}_C$, $b = M^{-1}\hat{Q}_G$. Note that Figure 5 depicts the computational graph formed by (53)*. The parameters θ are estimated jointly using an objective function of the form $\ell(\hat{f}(x;\theta),\tilde{y}) \hat{=} \ell(\theta)$. In reverse mode AD, the gradient of ℓ with respect to θ is decomposed as*

$$\nabla_\theta \ell(\theta) = \underbrace{\nabla_{\hat{Q}_C}\ell \nabla_\theta \hat{Q}_C}_{\bar{m}_1} + \underbrace{\nabla_{M^{-1}}\ell \nabla_\theta M^{-1}}_{\bar{m}_2} + \underbrace{\nabla_{\hat{Q}_G}\ell \nabla_\theta \hat{Q}_G}_{\bar{m}_3}, \tag{54}$$

with

$$\bar{m}_1 = \underbrace{\nabla_a \ell \nabla_{\hat{Q}_C} a}_{\bar{m}_4} \nabla_\theta \hat{Q}_C, \quad \bar{m}_2 = \big(\underbrace{\nabla_a \ell \nabla_{M^{-1}} a}_{\bar{m}_5} + \underbrace{\nabla_b \ell \nabla_{M^{-1}} b}_{\bar{m}_6} \big) \nabla_\theta M^{-1}, \quad \bar{m}_3 = \underbrace{\nabla_b \ell \nabla_{\hat{Q}_G} b}_{\bar{m}_7} \nabla_\theta \hat{Q}_G, \tag{55}$$

$$\bar{m}_4 = \underbrace{\nabla_{\hat{\ddot{q}}}\ell \nabla_a \hat{\ddot{q}}}_{\bar{m}_8} \nabla_{\hat{Q}_C} a, \quad \bar{m}_5 = \underbrace{\nabla_{\hat{\ddot{q}}}\ell \nabla_a \hat{\ddot{q}}}_{\bar{m}_8} \nabla_{M^{-1}} a, \quad \bar{m}_6 = \underbrace{\nabla_{\hat{\ddot{q}}}\ell \nabla_b \hat{\ddot{q}}}_{\bar{m}_9} \nabla_{M^{-1}} b, \quad \bar{m}_7 = \underbrace{\nabla_{\hat{\ddot{q}}}\ell \nabla_b \hat{\ddot{q}}}_{\bar{m}_9} \nabla_{\hat{Q}_G} b. \tag{56}$$

Therefore, the gradient of an analytical structured model can be computed with AD. Yet, this requires the computation of the numerical expressions for $\nabla_\theta M^{-1}$, $\nabla_\theta \hat{Q}_G$, and $\nabla_\theta \hat{Q}_C$. Ideally, an AD library will compute these terms for us by also decomposing the gradients into product-sums of basic gradient functions.

Example 4.1 shows that the computation of $\nabla_\theta \ell(\theta)$ requires the partial differentiation of analytical latent functions. However, to do this efficiently, several requirements must be fulfilled by an AD library, namely:

- Analytical structured models denote forward functions that consist of numerous basic mathematical operators. Ideally, the AD library should only require that the analytical structured model is expressed in terms of a computer library for basic linear algebraic operations such as Numpy [80] or Torch [9]. In turn, the model designer is only required to convert the derived analytical functions into the programming language of the respective AD library.
- As detailed in Section 3.4.1, models for $\hat{Q}_\text{C}$, $\hat{Q}_\text{G}$, or $\hat{M}$ can be obtained by transformation of an analytical or data-driven function via the partial derivative operators ∇_q or $\nabla_{\dot{q}}$. Therefore, the AD library must be able to automatically compute the gradients (w.r.t. to θ) of gradients (w.r.t. q or $\dot{q}$) of latent functions.

Fortunately, recent developments in AD packages such as ”AutoGrad“ [81] and ”Torch autograd“ allow to compute $\nabla_\theta \ell(\theta)$ in which $\ell(\theta)$ is expressed in either native python (Numpy) code or python (torch) code. Importantly in these AD packages, $\nabla_\theta \ell(\theta)$ can itself include higher-order partial derivatives without braking the AD routines. These AD packages have been further improved in packages such as JAX [8] and PyTorch [9], which additionally combine AD with compilers for GPU acceleration such as XLA. These packages tremendously simplify the synthesis of an analytical structured model, enables easy debugging of the respective computer code, as well as provide computationally efficient implementations with GPU acceleration.

4.2 | Model construction of analytical latent models

In the previous section, we detailed how an analytical structured model is trained using modern libraries for AD. These AD libraries only require that a forward function $\ell(\hat{f}(\tilde{x};\theta),\tilde{y})$ is expressed in terms of a specified programming language (*e.g.*, Python using solely Numpy expressions). In this section, we detail important aspects that must be considered when constructing an analytical structured model. An analytical model $\hat{f}_\text{A}(x;\theta_\text{A})$ can be derived using for example a rigid-body dynamics software package such as Pinocchio [82].

If a data-driven model is used as an approximation for a function inside an analytical mode, this usually places additional requirements on the properties of the data-driven model. ALM requires to place data-driven models on unknown functions inside an analytical model. As discussed in Section 3, these unknown quantities are either vector-valued functions f_i (*e.g.*, forces) or matrix-valued functions C_i (*e.g.*, matrices). Depending on the analytical mechanics formulation, these data-driven functions are usually transformed by operators which in return poses additional mathematical requirements onto the data-driven models. These transformations of data-driven models can be distinguished into:

Matrix transformations, writing

$$\hat{f}_i^{C}(x) = C_i(x)\hat{f}_i(x). \tag{57}$$

e.g., in forward dynamics a model can be placed on Q_d which is transformed by M^{-1}, writing $\hat{\ddot{q}}_\mathrm{d} = M^{-1}\hat{F}_D$.
Partial derivative transformations, writing

$$\hat{f}_i^{\nabla_x}(x) = \nabla_x f_i(x)|_{x=z}, \tag{58}$$

which we denote by abuse of notation as

$$\hat{f}_i^{\nabla_x}(x) = \nabla_x f_i(x), \tag{59}$$

e.g., modeling Q_C in terms of a model for the kinetic energy $\hat{T}$ (cf. (17)) reads $\hat{Q}_\mathrm{C}(q,\dot{q}) = -\left(\nabla_q \nabla_{\dot{q}}^\top \hat{T}\right)\dot{q} + \nabla_q \hat{T}$.
Substitution of input variables by a nonlinear mapping $x = u(z)$ such that

$$\hat{f}_i^{u}(x) = f_i(z)|_{z=u(x)}, \tag{60}$$

e.g., instead of using an angle coordinate $x \,\hat{=}\, \phi$ for describing the pose of a pendulum, the pendulums pose can be expressed as $z = [\cos(\phi), \sin(\phi)]$. This coordinate transformation has the advantage that the entries of z are bounded to $[-1, 1]$. Another example is the descriptions of z in terms of a NN with inputs x.

Note that all of the above operators are linear operators. We illustrate the additional requirements that the above operators pose onto a data-driven model on the examples of NN and GPs.

Linearly transformed neural networks

Hendriks et al. [83] details linear transformations of NNs such that an equality constrained is fulfilled. Additional details on ARMs with linear transformed NNs is found in [22, 68, 23, 72, 67, 71] (cf. Section 3.4). If the NN outputs parameterize the entries of M, the positive definiteness of M poses additional requirements on the last layer of the deep network. Moreover, transforming a NN with partial differential operators requires a careful selection of the activation functions. For example, the Lagrangian NN developed by Cranmer et al. [72] contains the Hessian $\left(\nabla_q \nabla_{\dot{q}}^\top \hat{\mathcal{L}}\right)$ of the NN $\hat{\mathcal{L}}$. The second-order derivative of a RELU activation with respect to its inputs is zero. Therefore, the authors used and compared RELU^2, RELU^3, tanh, sigmoid, and softplus activation functions. For their Lagrangian NN the authors chose the softplus activation function. The usage of higher-order gradient-based optimization methods additionally affects the choice of the activation functions.

Linearly transformed Gaussian processes

Jidling et al. [84] and Lange-Hegermann [85] detail linear transformations of GPs such that an equality constrained is fulfilled. Additional details on ARMs with linear transformed GPs is found in [65, 24] (cf. Section 3.4). Gaussian processes are closed under linear operators, such that with $\hat{f}_i \sim \mathcal{GP}(\mathbf{m}(x), \mathbf{k}(x, x'))$, (57) yields

$$\hat{f}_i^{C} \sim \mathcal{GP}\left((C_i(x)\mathbf{m}(x),\ C_i(x)\mathbf{k}(x, x')C_i^\top(x')\right), \tag{61}$$

with the vector-valued mean function $\mathbf{m}(x)$ and matrix-valued (diagonal) kernel function $\mathbf{k}(x, x')$. The resulting GP $\hat{f}_i^{C}$ is a multi-output GP [86]. Most works that do ARM with GPs, model all output dimensions via independent GPs as the computation of the inverse covariance matrices of n one-dimensional GPs scales with $\mathcal{O}\big(nN^3\big)$. In comparison, the computation of the covariance matrix of an n-dimensional multitask GP scales with $\mathcal{O}\big((nN)^3\big)$. However, a big advantage of multi-output GPs obtained from ALM modeling is that the transformation $C_i(x;\theta)$ is often known a-priori from analytical mechanics. For example, in case of the Newton-Euler equation $\ddot{q} = M^{-1}Q$, we can place a GP prior on Q, writing $\hat{Q} \sim \mathcal{GP}(\mathbf{m}(x), \mathbf{k}(x, x'))$, such that

$$\hat{\ddot{q}} \sim \mathcal{GP}\left((M^{-1}\mathbf{m},\ M^{-1}\,\mathbf{k}\,[M^{-1}]^\top\right). \tag{62}$$

In turn, the prior knowledge of M^{-1} specifies how the individual dimensions of $\hat{Q}$ are correlated with each other to yield $\hat{\ddot{q}}$. As detailed in a Jidling et al. [84] and Lange-Hegermann [85], a GP can be transformed by a matrix of partial derivatives to yield another GP. For example, for $n = 2$, the matrix of operators

$$C_x = (\nabla_q \nabla_{\dot{q}}^\top) = \begin{bmatrix} \frac{\partial^2}{\partial q_1 \partial \dot{q}_1} & \frac{\partial^2}{\partial q_1 \partial \dot{q}_2} \\ \frac{\partial^2}{\partial q_2 \partial \dot{q}_1} & \frac{\partial^2}{\partial q_2 \partial \dot{q}_2} \end{bmatrix} \tag{63}$$

transforms a prior $\hat{\mathcal{L}} \sim \mathcal{GP}(\mathbf{m}(x), \mathbf{k}(x, x'))$ (cf. [65]) as

$$C_x \hat{\mathcal{L}} \sim \mathcal{GP}(C_x \mathbf{m}(x),\ C_x \mathbf{k}(x, x') C_x^\top). \tag{64}$$

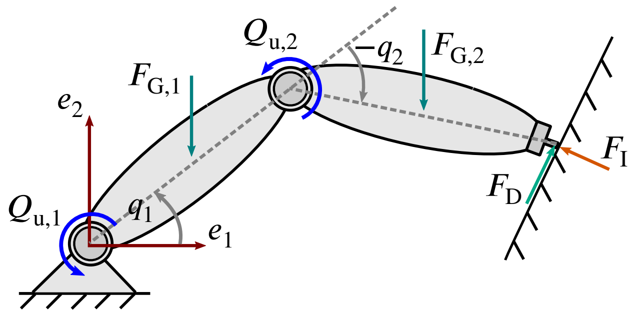

FIGURE 6 Sketch of a two-link robot arm who's endeffector touches a surface.

Therefore, if a GP is transformed by a matrix of partial derivatives, $\mathbf{m}(x)$ and $\mathbf{k}(x, x')$ must be sufficiently often differentiable with respect to the kernel's inputs. Further, as the sum of GPs are itself a GP (cf. [87, p. 13]), a model for $Q \mathrel{\widehat{=}} Q_G + Q_C$ can be obtained as

$$\hat{q} \sim \mathcal{GP}(\mathbf{m}_C + \mathbf{m}_G,\ \mathbf{k}_C + \mathbf{k}_G), \tag{65}$$

with $\hat{Q}_C \sim \mathcal{GP}(\mathbf{m}_C(x), \mathbf{k}_C(x, x'))$ and $\hat{Q}_G \sim \mathcal{GP}(\mathbf{m}_G(x), \mathbf{k}_G(x, x'))$.

5 | CASE STUDY: STRUCTURED MODELING OF A ROBOT ARM

To illustrate the concepts presented in this survey, let us assume that we want to design an analytical structured model for a robot arm whose end-effector is in contact with a surface. We chose this example, as robot arms are widely used mechanical systems that yield complex dynamics functions. Moreover, a robot arm shares similar mechanical characteristics to a robotic leg which is an important component in robotics. While a 2D-system forms an abstraction of a real robot arm, similar modeling assumptions as in the 3D case are made while keeping the discussion concise. This discussion is based on the robot arm equations derived by Siciliano et al. [25, p. 265-269]. Figure 6 depicts a schematic of the 2D robot arm. Here, Cartesian forces are denoted as F. The arm orientation is described via the generalized angle coordinates q_1 and q_2. As described in Section 2, the arm's rigid-body dynamics model takes the form

$$\underbrace{M\ddot{q} = Q_C + Q_G}_{\text{mass-related quantities}} + Q_u + \underbrace{Q_d + Q_I}_{\text{constraint forces}}\ . \tag{66}$$

In this example, $Q_d \mathrel{\widehat{=}} Q_{d,I}$ denotes the friction force that is applied by the surface onto the endeffector while Q_u denotes the joint torques. While for all of these terms, analytical functions have been proposed, we need to ask ourselves if these models form good descriptions of the real system dynamics. As discussed in Section 3, how we model the analytical functions heavily depends on the system and which quantities we can observe accurately.

Defining the analytical parameters

It is assumed that the center of mass (COM) of the first body lies on an axis of length a_1 between the base joint and second joint. The COM of the second body lies on the axis of length a_2 between the second joint and the end-effector tip. The distance from the respective joints to a body's COM is denoted by L_i. The mass of each body is denoted by m_i and the inertia around axis e_3 at the COM of the body by I_{L_i}. The robot arm is actuated by motors located in the joints. It is assumed that the COG of the motors is located at the origin of the respective generalized coordinate frames. The mass of the motor's rotors is m_{m_i} and its inertia amounts to I_{m_i}. The gear reduction ratio of each motor is k_{ri}.

Modeling the mass related quantities

The inertia matrix of the robot arm follows from an analytical derivation as

$$M(q) = \begin{bmatrix} b_{11}\left(q_2\right) & b_{12}\left(q_2\right) \\ b_{21}\left(q_2\right) & b_{22} \end{bmatrix} \text{ with } \begin{aligned} b_{11} =& I_{L_1} + m_{L_1} L_1^2 + k_{r1}^2 I_{m_1} + I_{L_2} \\ & + m_{L_2}\left(a_1^2 + L_2^2 + 2a_1 L_2 \cos(q_2)\right) + I_{m_2} + m_{m_2} a_1^2, \\ b_{12} =& b_{21} = I_{L_2} + m_{L_2}\left(L_2^2 + a_1 L_2 \cos(q_2)\right) + k_{r2} I_{m_2}, \\ b_{22} =& I_{L_2} + m_{L_2} L_2^2 + k_{r2}^2 I_{m_2}. \end{aligned} \tag{67}$$

The fictitious force follows from (17) as

$$Q_\mathrm{C} = -\nabla_q(\dot{q}^\top M)\dot{q} + \frac{1}{2}\nabla_q\left(\dot{q}^T M \dot{q}\right) = -C(q,\dot{q})\dot{q}, \text{ with } C(q,\dot{q}) = \nabla_q(\dot{q}^\top M) = \begin{bmatrix} h\dot{q}_2 & h\left(\dot{q}_1 + \dot{q}_2\right) \\ -h\dot{q}_1 & 0, \end{bmatrix} \tag{68}$$

while the model for the gravitational force reads

$$Q_\mathrm{G} = \begin{bmatrix} \left(m_{\ell_1}\ell_1 + m_{m_2}a_1 + m_{\ell_2}a_1\right) g\cos(q_1) + m_{\ell_2}\ell_2 g\cos(q_1 + q_2) \\ m_{\ell_2}\ell_2 g\cos(q_1 + q_2) \end{bmatrix}. \tag{69}$$

To simplify this derivation, several assumptions were made which are also typically found in other analytical models, namely:

- The COM of the bodies as well as of the motor's rotors lies on a predefined axis.
- Elasticities of bodies and compliance inside the joints are not modeled.

The latter point cannot be avoided using rigid-body modeling and hence can contribute to ϵ_R (cf. Section 3). To reduce ϵ_R we can use ARM. However, learning these errors is difficult as we have only limited sensory information, *e.g.*, measurements of q, $\dot{q}$, and Q_u ($\ddot{q}$ is often estimated by filtering measurements of $\dot{q}$ [25, p. 281]). If we do not want to model ϵ_R solely as a noise process, one can either take the history of the dynamics into account to learn on a streak of data points as done in [6] or add additional sensors that capture information about the elasticities.

Errors that stem from a wrong description of the bodies COM positions should ideally be avoided by improving the analytical model itself. It is tempting to directly resort to the data-driven methods presented in Section (3.4.1) which model M, Q_G, and Q_C via a NNs or GPs. However, for many mechanical systems, an accurate depiction of M, Q_G, and Q_C can be obtained from analytical modeling. The question is how much time and expertise are we willing to invest to obtain a sufficient analytical description of M, Q_G, and Q_C and in return potentially improve the sample-efficiency of an analytical structured mdoel.

Modeling the joint torques

The identification of the non-conservative forces constitutes a big challenge in robotic system. As discussed in Section 3, how the joint torques Q_u are modeled depends on how they are measured.

If the joint torques are directly measured through joint-torque sensors, one can directly learn the function Q_u. However, while theoretically, it should be possible to learn Q_u from data of $\{q, \dot{q}, \ddot{q}\}$, the acceleration $\ddot{q}$ is usually not directly observed and is therefore considerably noisy. In return, we suggest to learn Q_u using the techniques discussed in Section **??**. For example, Q_u can be directly learned using a NN that has as input a sequence of position errors and velocities or $Q_\mathrm{u,desired}$.

Alternatively, if no accurate measurements of Q_u are available one can design a structured inverse dynamics model as discussed in Section 3.3. However, this model will learn the mapping from the current state-acceleration observation to the measured Q_u. In return, such a model is not necessarily suited for a simulation of the joint torques. A second approach is to directly learn the forward dynamics via ALM. By doing so, we need to only specify a model for Q_u.

Modeling the implicit constraint forces

The identification of Q_d and Q_I is also challenging as Q_d is highly environmental dependent and Q_I is a contact force. We observed two different philosophies on how to model these forces in robotic systems.

The first approach, which Lee et al. [76] used to achieve seminal results for the control a quadruped, is to directly measure and identify Q_u. Then we can insert the model for Q_u in a rigid-body dynamics simulation where we can simulate different Q_d and Q_I. In return, one can design or learn a control strategy that yields robust performance over a large set of different Q_d. However, even in this scenario, it can desirable to identify a specific Q_d as robustness of a specific control policy to different dissipative forces can come at the cost of losing control performance.

The second approach, aims at modeling Q_d inside the inverse or forward dynamics of the system. As shown in Section 2.3 one can include constrained knowledge to eliminate Q_I from the dynamics equations. For example, assuming that the robot arm's endeffector is pressing on a flat surface such that the Cartesian coordinate description of the endeffector is $x_{e,2} = 0$. Then, this surface can be described by an implicit constraint equation as

$$0 = x_{e,2} = a_1 \sin(q_1) + a_2 \sin(q_1 + q_2). \tag{70}$$

The the second time-derivative of (71) yields the constraining equation (20) as

$$\left[a_1 \cos(q_1) + a_2 \cos(q_1 + q_2),\ a_2 \cos(q_1 + q_2)\right] \ddot{q} = a_1 \sin(q_1)\dot{q}^2 + a_2 \sin(q_1 + q_2)(\dot{q}_1 + \dot{q}_2)^2. \tag{71}$$

This equation can be used to obtain an analytical structured model as detailed in Section 3.3 and Section 3.4.3.

Constructing the structured model and training

Now we assume that all mechanical functions and error functions of the dynamics model are either modeled analytically or as a data-driven model. As discussed in Section 4.1, we suggest that these functions are then expressed in the programming language of a suitable automatic differentiation library to yield the analytical structured model. After the specification of the objective function, one can estimate the unidentified parameters of the structured model using collected data.

6 | CONCLUSION AND FUTURE OUTLOOK

In this survey, we propose a unified view on structured modeling by showing that output and latent modeling are the key building blocks for analytical structured models. We distinguish learning the residuals of an analytical model in the model's output, which is referred to as analytical (output) residual modeling (ARM), and analytical latent modeling (ALM), in which a data-driven model approximates latent residuals or functions inside the analytical model. To this effect, we consolidated the proposed view by illustrating that unknown analytical latent functions are often transformed by a known linear operator, which advocates ALM. The review of the literature (Sec. 3), as well as needed techniques (Sec. 4), revealed a shift in the optimization techniques used in works on structured modeling. Namely, we observed a shift from analytical models that are optimized via linear regression towards structured latent models that are optimized via automatic differentiation. This shift in combination with improvements in the functionality of automatic differentiation software libraries causes analytical mechanics to potentially take a pivotal role in the development of data-efficient and practical nonlinear dynamics models of mechanical systems.

As analytical structured modeling is a relatively recent field, many open questions on the design of these models remain. Below we provide a list of what we perceive as potential directions that require further analysis in the field of analytical structured modeling, but that were beyond the scope of this survey.

Rigid-body dynamics

Many of the discussed works use the recursive Newton-Euler algorithm as thoroughly discussed in [11]. However, some works also resorted to alternative dynamics descriptions such as the Newton-Euler equation in Lie algebra formulation [70] or the Udwadia-Kalaba equation [34]. It seems that the recent works on analytical structured learning only touched the surface of the vast field of rigid-body simulations and it is worth further discussion which descriptions of motion are particularly suited for the combination with data-driven learning techniques.

Partially observable systems

There exist numerous works on data-driven modeling of partially observable systems [88, 58, 89]. Yet, works on analytical structured learning do typically not consider that for many mechanical systems, the state is not directly observable and only partial state information is given via the observation equation $o_k = h(q^T, \dot{q}^T, u_k, \epsilon_o)$. Here $h(q^T, \dot{q}^T, u_k, \epsilon_o)$ most often denotes a simple algebraic equation and ϵ_o an additional noise process. The question remains if the existing insights from data-driven modeling can be extended to analytical structured modeling of partially observable systems.

Noise on model inputs

The observations of the input variables x are often noisy and biased. In dynamics modeling this is often not taken into account during training. The question arises if input noise can and should be addressed. Notably, there are works that propagate input noise through the trained dynamics model when doing predictions [52, 90].

Active learning

We assumed that informative system data is given. However, the collection of informative data from dynamical systems is itself a subject of ongoing research efforts [91, 92, 93, 94]. The question arises as to which extend structural knowledge can be used to make data-collection more efficient as well as how to use this data to train analytical structured models.

Interpretable data-driven models

While data-driven models are often referred to as black-box models, there are efforts made to provide guaranties for data-driven model predictions [95, 96] as well as carefully designing data-driven models to make them more interpretable [97]. These works are critical if the application of a data-driven model requires prediction guarantees.

Combination of forward dynamics models with numerical integration

As previously mentioned, a disadvantage of analytical structured learning is the often considerably large noise on the acceleration estimates. However, direct dynamics are usually modeled to obtain a trajectory prediction via numerical integration. Therefore, an open question is to which extend analytical structured models can be combined with numerical integration schemes such that training of such a combined model is done solely on trajectory data. Recent works discuss how to approximate ODEs using GPs [98, 99, 100] and NNs [75, 101] as well as solve initial-value problems using GPs [102, 103].

Classical mechanics provides us with an understanding of the structural relationships underlying the motion of mechanical systems. One such insight is that motion is a consequence of the interaction of mass and force. Yet, to predict motion we need to describe the effects of mass and force mathematically. This turns out to be problematic in practice as the distribution of mass, as well as the applied forces, are itself the product of convoluted physical phenomenons that are difficult for us humans to describe a-priori. To address this and similar bottlenecks in a mathematical description of the world, we forged tools to learn functions from data. However, using data-driven learning methods comes at the price of requiring significant amounts of data as well as losing insight on how a prediction of such a model came about. In this survey, we show that the combination of rigid-body dynamics with data-driven learning yields not only a more data-efficient dynamics model but also enables us to keep our structural understanding of the cause of motion when making predictions. To this extent, we discussed in this survey different sources of structural knowledge of rigid-body dynamics. Here, we showed that the errors of a rigid-body dynamics model is located either inside latent functions of the model such as forces or, stem from phenomena that lie beyond rigid-body dynamics descriptions. In turn, we categorized current works that combine rigid-body mechanics with data-driven learning. Finally, we emphasize recent improvements in automatic differentiation libraries that significantly ease the synthesis of analytical structured models. Recently, an analytical structured model [76] enabled the synthesis of a quadruped robot's control policy with so far unseen and ineffable robustness to changes in the environment and carrying-load. This work emphasized how effective the combination of analytical with data-driven models can be. We believe that this and the other works presented in this survey form the beginning of a broad application of analytical structured models in wide ranges of engineering, and we are excited to see what kind of technical applications these class of models might enable in the future.

Acknowledgments

We thank Alexander von Rohr, Steve Heim and Christian Fiedler for their valuable feedback and many interesting discussions. This work has been supported in part by the Max Planck Society and in part by the Cyber Valley initiative. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting A. René Geist.

References

[1] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

[2] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[3] Oliver Nelles. *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer Science & Business Media, 2013.

[4] Duy Nguyen-Tuong and Jan Peters. Using model knowledge for learning inverse dynamics. In *International Conference on Robotics and Automation (ICRA)*, pages 2677–2682. IEEE, 2010.

[5] Giovanni Sutanto, Austin Wang, Yixin Lin, Mustafa Mukadam, Gaurav Sukhatme, Akshara Rai, and Franziska Meier. Encoding physical constraints in differentiable newton-euler algorithm. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 804–813, The Cloud, 10–11 Jun 2020. PMLR.

[6] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

[7] Fernando Díaz Ledezma and Sami Haddadin. First-order-principles-based constructive network topologies: An application to robot inverse dynamics. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 438–445. IEEE, 2017.

[8] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[10] Thomas R Kane and David A Levinson. *Dynamics, theory and applications*. McGraw Hill, 1985.

[11] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2008.

[12] Philip J Davis and Philip Rabinowitz. *Methods of numerical integration*. Courier Corporation, 2007.

[13] Michael I Jordan. Constrained supervised learning. *Journal of Mathematical Psychology*, 36(3):396–425, 1992.

[14] Christopher G Atkeson, Chae H An, and John M Hollerbach. Estimation of inertial parameters of manipulator loads and links. *The International Journal of Robotics Research*, 5(3):101–119, 1986.

[15] Jo-Anne Ting, Michael N Mistry, Jan Peters, Stefan Schaal, and Jun Nakanishi. A bayesian approach to nonlinear parameter identification for rigid body dynamics. In *Robotics: Science and Systems*, pages 32–39. Philadelphia, USA, 2006.

[16] Silvio Traversaro, Stanislas Brossette, Adrien Escande, and Francesco Nori. Identification of fully physical consistent inertial parameters using optimization on manifolds. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5446–5451. IEEE, 2016.

[17] Patrick M Wensing, Sangbae Kim, and Jean-Jacques E Slotine. Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution. *IEEE Robotics and Automation Letters*, 3(1): 60–67, 2017.

[18] Fernando Díaz Ledezma and Sami Haddadin. Fop networks for learning humanoid body schema and dynamics. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9. IEEE, 2018.

[19] Subham S. Sahoo, Christoph H. Lampert, and Georg Martius. Learning equations for extrapolation and control. In *Proc. 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, 2018*, volume 80, pages 4442–4450. PMLR, 2018. URL http://proceedings.mlr.press/v80/sahoo18a.html.

[20] Dominik Baumann, Friedrich Solowjow, Karl H Johansson, and Sebastian Trimpe. Identifying causal structure in dynamical systems. *arXiv preprint arXiv:2006.03906*, 2020.

[21] Ljung Lennart. System identification: theory for the user. *PTR Prentice Hall, Upper Saddle River, NJ*, pages 1–14, 1999.

[22] M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. May 2019.

[23] Jayesh K. Gupta, Kunal Menda, Zachary Manchester, and Mykel Kochenderfer. Structured mechanical models for robot learning and control. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 328–337, The Cloud, 10–11 Jun 2020. PMLR.

[24] Andreas Geist and Sebastian Trimpe. Learning constrained dynamics with gauss' principle adhering gaussian processes. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 225–234. PMLR, 10–11 Jun 2020.

[25] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[26] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5(2):3650–3657, 2020. .

[27] Shivesh Kumar. *Modular and Analytical Methods for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots*. PhD thesis, Universität Bremen, 2019.

[28] Randal W Beard. Linear operator equations with applications in control and signal processing. *IEEE Control Systems Magazine*, 22(2):69–79, 2002.

[29] Firdaus E Udwadia and Robert Kalaba. *Analytical dynamics: a new approach*. Cambridge University Press, 2007.

[30] Jean Le Rond d'Alembert. *Traité de dynamique*. 1743.

[31] Joseph Louis Lagrange. *Méchanique analytique*. Mme. De Courcier, Paris, 1787.

[32] Firdaus E Udwadia, Robert E Kalaba, and Hee-Chang Eun. Equations of motion for constrained mechanical systems and the extended d'alembert's principle. *Quarterly of Applied Mathematics*, 55(2):321–331, 1997.

[33] Leopold Alexander Pars. A treatise on analytical dynamics(treatise on analytical dynamics covering lagrange equations, gibbs-appell equations, hamilton- jacobi theorem, contact transformations, etc). *NEW YORK, JOHN WILEY AND SONS, INC., 1965. 641 P*, 1965.

[34] Firdaus E Udwadia and Robert E Kalaba. On the foundations of analytical dynamics. *International Journal of non-linear mechanics*, 37(6):1079–1090, 2002.

[35] Firdaus E Udwadia and Robert E Kalaba. Nonideal constraints and lagrangian dynamics. *Journal of Aerospace Engineering*, 13(1):17–22, 2000.

[36] Farhad Aghili. A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: applications to control and simulation. *IEEE Transactions on Robotics*, 21(5):834–849, 2005.

[37] Carl Friedrich Gauß. Über ein neues allgemeines grundgesetz der mechanik. *Journal für die reine und angewandte Mathematik*, 4:232–235, 1829.

[38] Firdaus E Udwadia and Robert E Kalaba. A new perspective on constrained motion. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1906):407–410, 1992.

[39] Ulrike Von Luxburg and Bernhard Schölkopf. Statistical learning theory: Models, concepts, and results. In *Handbook of the History of Logic*, volume 10, pages 651–706. Elsevier, 2011.

[40] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NeurIPS)*, pages 1097–1105, 2012.

[42] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[43] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[44] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, et al. Alphastar: Mastering the real-time strategy game starcraft ii. *DeepMind blog*, page 2, 2019.

[45] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806, 1993.

[46] John G Kuschewski, Stefen Hui, and Stanislaw H Zak. Application of feedforward neural networks to dynamical system identification and control. *IEEE Transactions on Control Systems Technology*, 1(1):37–49, 1993.

[47] M Jansen. Learning an accurate neural model of the dynamics of a typical industrial robot. In *Int. Conf. on Artificial Neural Networks*, pages 1257–1260, 1994.

[48] Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9258–9268, 2018.

[49] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4754–4765, 2018.

[50] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112, 2020.

[51] Mauricio A Alvarez, Lorenzo Rosasco, and Neil D Lawrence. Kernels for vector-valued functions: A review. *arXiv preprint arXiv:1106.6251*, 2011.

[52] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[53] Juš Kocijan, Agathe Girard, Blaž Banko, and Roderick Murray-Smith. Dynamic systems identification with Gaussian processes. *Mathematical and Computer Modelling of Dynamical Systems*, 11(4):411–424, 2005.

[54] Roger Frigola, Fredrik Lindsten, Thomas B Schön, and Carl Edward Rasmussen. Bayesian inference and learning in Gaussian process state-space models with particle MCMC. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3156–3164, 2013.

[55] César Lincoln C Mattos, Andreas Damianou, Guilherme A Barreto, and Neil D Lawrence. Latent autoregressive Gaussian processes models for robust system identification. *IFAC-PapersOnLine*, 49(7):1121–1126, 2016.

[56] Andreas Doerr, Christian Daniel, Duy Nguyen-Tuong, Alonso Marco, Stefan Schaal, Marc Toussaint, and Sebastian Trimpe. Optimizing long-term predictions for model-based policy search. In *Conference on Robot Learning*, pages 227–238, 2017.

[57] Stefanos Eleftheriadis, Tom Nicholson, Marc Deisenroth, and James Hensman. Identification of Gaussian process state space models. In *Advances in neural information processing systems (NeurIPS)*, pages 5309–5319, 2017.

[58] Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian. Probabilistic recurrent state-space models. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1280–1289, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[59] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[60] Chae H An, Christopher G Atkeson, and John M Hollerbach. Estimation of inertial parameters of rigid body links of manipulators. In *1985 24th IEEE Conference on Decision and Control*, pages 990–995. IEEE, 1985.

[61] John YS Luh, Michael W Walker, and Richard PC Paul. On-line computational scheme for mechanical manipulators. 1980.

[62] Joseph Sun De La Cruz, Dana Kulić, and William Owen. Online incremental learning of inverse dynamics incorporating prior knowledge. In *International Conference on Autonomous and Intelligent Systems*, pages 167–176. Springer, 2011.

[63] Terry Taewoong Um, Myoung Soo Park, and Jung-Min Park. Independent joint learning: A novel task-to-task transfer learning scheme for robot models. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5679–5684. IEEE, 2014.

[64] Ruben Grandia, Diego Pardo, and Jonas Buchli. Contact invariant model learning for legged robot locomotion. *IEEE Robotics and Automation Letters*, 3(3):2291–2298, 2018.

[65] Ching-An Cheng, Han-Pang Huang, Huan-Kun Hsu, Wei-Zh Lai, and Chih-Chun Cheng. Learning the inverse dynamics of robotic manipulators in structured reproducing kernel hilbert space. *IEEE transactions on cybernetics*, 46(7):1691–1703, 2015.

[66] Jemin Hwangbo, Joonho Lee, and Marco Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018.

[67] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15379–15389, 2019.

[68] Michael Lutter, Kim Listmann, and Jan Peters. Deep lagrangian networks for end-to-end learning of energy-based control for under-actuated systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7718–7725. IEEE, 2019.

[69] Michael Lutter, Johannes Silberbauer, Joe Watson, and Jan Peters. A differentiable newton euler algorithm for multi-body model learning. *arXiv preprint arXiv:2010.09802*, 2020.

[70] Junggon Kim. Lie group formulation of articulated rigid body dynamics. Technical report, Technical Report. Carnegie Mellon University, 2012.

[71] Peter Toth, Danilo J Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. In *International Conference on Learning Representations*, 2019.

[72] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[73] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[74] Raffaello Camoriano, Silvio Traversaro, Lorenzo Rosasco, Giorgio Metta, and Francesco Nori. Incremental semiparametric inverse dynamics learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 544–550. IEEE, 2016.

[75] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems (NeurIPS)*, pages 6571–6583, 2018.

[76] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020.

[77] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[78] Atılım Günes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, 18(1):5595–5637, 2017.

[79] George F Corliss. Applications of differentiation arithmetic. In *Reliability in Computing*, pages 127–148. Elsevier, 1988.

[80] Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. .

[81] Dougal Maclaurin, David Duvenaud, and Ryan P Adams. Autograd: Effortless gradients in numpy. In *ICML 2015 AutoML Workshop*, volume 238, page 5, 2015.

[82] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *IEEE International Symposium on System Integrations (SII)*, 2019.

[83] Johannes Hendriks, Carl Jidling, Adrian Wills, and Thomas Schön. Linearly constrained neural networks. *arXiv preprint arXiv:2002.01600*, 2020.

[84] Carl Jidling, Niklas Wahlström, Adrian Wills, and Thomas B Schön. Linearly constrained gaussian processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1215–1224, 2017.

[85] Markus Lange-Hegermann. Algorithmic linearly constrained gaussian processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2137–2148, 2018.

[86] Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for vector-valued functions: A review. *Foundations and trends in machine learning*, 4(3):195–266, 2012. ISSN 1935-8237.

[87] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[88] Wee Lee, Nan Rong, and David Hsu. What makes some pomdp problems easy to approximate? *Advances in neural information processing systems (NeurIPS)*, 20:689–696, 2007.

[89] Sebastian Curi, Silvan Melchior, Felix Berkenkamp, and Andreas Krause. Structured variational inference in partially observable unstable gaussian process state space models. In Alexandre M. Bayen, Ali Jadbabaie, George Pappas, Pablo A. Parrilo, Benjamin Recht, Claire Tomlin, and Melanie Zeilinger, editors, *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 147–157, The Cloud, 10–11 Jun 2020. PMLR.

[90] Paavo Parmas, Carl Edward Rasmussen, Jan Peters, and Kenji Doya. Pipps: Flexible model-based policy search robust to the curse of chaos. In *International Conference on Machine Learning*, pages 4065–4074. PMLR, 2018.

[91] Thomas B Schön, Adrian Wills, and Brett Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, 2011.

[92] Max Simchowitz, Horia Mania, Stephen Tu, Michael I. Jordan, and Benjamin Recht. Learning without mixing: Towards a sharp analysis of linear system identification. In *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 439–473. PMLR, 06–09 Jul 2018.

[93] Johan Schoukens and Lennart Ljung. Nonlinear system identification: A user-oriented road map. *IEEE Control Systems Magazine*, 39(6):28–99, 2019.

[94] Mona Buisson-Fenet, Friedrich Solowjow, and Sebastian Trimpe. Actively learning gaussian process dynamics. In *Learning for Dynamics and Control*, pages 5–15. PMLR, 2020.

[95] Jonas Umlauft, Armin Lederer, and Sandra Hirche. Learning stable Gaussian process state space models. In *American Control Conference (ACC)*, pages 1499–1504. IEEE, 2017.

[96] Thomas Beckers, Jonas Umlauft, Dana Kulic, and Sandra Hirche. Stable Gaussian process based tracking control of Lagrangian systems. In *Conference on Decision and Control (CDC)*, pages 5180–5185. IEEE, 2017.

[97] Amir Ali Ahmadi and Bachir El Khadir. Learning dynamical systems with side information. In Alexandre M. Bayen, Ali Jadbabaie, George Pappas, Pablo A. Parrilo, Benjamin Recht, Claire Tomlin, and Melanie Zeilinger, editors, *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 718–727, The Cloud, 10–11 Jun 2020. PMLR.

[98] Markus Heinonen, Cagatay Yildiz, Henrik Mannerström, Jukka Intosalmi, and Harri Lähdesmäki. Learning unknown ode models with gaussian processes. In *International Conference on Machine Learning*, pages 1959–1968, 2018.

[99] Cagatay Yildiz, Markus Heinonen, Jukka Intosalmi, Henrik Mannerstrom, and Harri Lahdesmaki. Learning stochastic differential equations with gaussian processes without gradient matching. In *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2018.

[100] P. Wenk, G. Abbati, M. A. Osborne, B. Schölkopf, A. Krause, and S. Bauer. Odin: Ode-informed regression for parameter and state inference in time-continuous dynamical systems. In *Proceedings of the 34th Conference on Artificial Intelligence (AAAI)*, volume 34, pages 6364–6371. AAAI Press, February 2020. AAAI Technical Track: Machine Learning.

[101] Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3870–3882, Online, 26–28 Aug 2020. PMLR.

[102] M. Schober, D. Duvenaud, and P. Hennig. Probabilistic ODE solvers with runge-kutta means. In *Advances in Neural Information Processing Systems 27*, pages 739–747. Curran Associates, Inc., 2014.

[103] Michael Schober, Simo Särkkä, and Philipp Hennig. A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29(1):99–122, 2019.

□