Using Model Knowledge for Learning Forward Dynamics

Anonymous Author(s)
Affiliation
Address
email

Abstract: If a robot's dynamics function is difficult to model analytically, it is an attractive option to directly learn it from data. Yet, solely data-driven approaches require considerable amounts of data for training and do not extrapolate well to unseen regions of the system's state space. In this work, we emphasize that when a robot's links are sufficiently rigid, many analytical functions such as kinematics, inertia functions, and surface constraints encode informative prior knowledge on its dynamics. To this effect, we propose a framework for learning probabilistic forward dynamics that combines analytical knowledge with Gaussian processes utilizing automatic differentiation with GPU acceleration. Compared to solely data-driven modeling, the model's data efficiency improves while the model also respects physical constraints. We illustrate the proposed structured model on a seven joint robot arm in PyBullet. An efficient implementation of the framework is enclosed to the supplementary material.

Keywords: Machine learning, Robotics, Analytical mechanics

1 Introduction

An accurate nonlinear dynamics model is often essential for the synthesis of a robot's actions via model predictive control [1–4] or model-based reinforcement learning [5–10]. Yet, in many robotic systems including legged robots and robot arms, physical phenomena such as friction and elasticities considerably affect the dynamics, are often environmental dependent, and unknown a-priori which in turn aggravates analytical modeling. As an alternative to analytical modeling, recent works resort to learning dynamics solely from input-output data using regression models. However, the collection of data from robotic systems must be done in real-time using potentially sub-optimal control strategies that only operate in small regions of the system's state-space. Therefore, data sets that stem from physical systems are often limited in size and information value. This raises a considerable obstacle

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

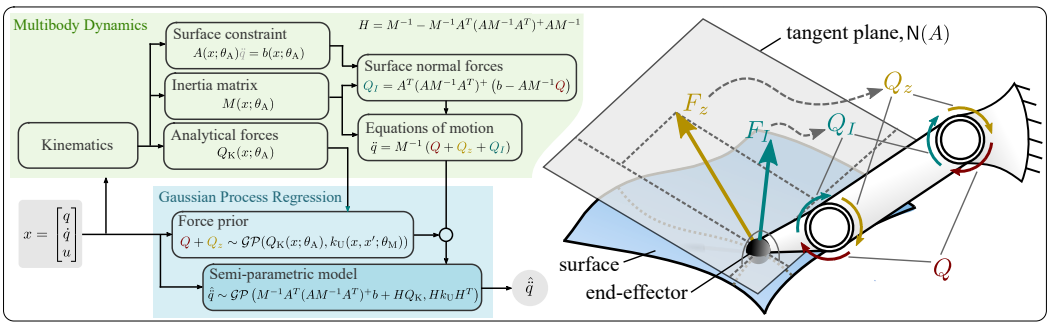


Figure 1: Computational graph of the proposed framework. A robot arm's forward dynamics model is augmented by a GP approximating unknown forces Q_U. The semi-parametric model predicts acceleration $\ddot{q}$ while potentially including knowledge in form of analytical functions $\{M, A, b, Q_K\}$.

for successfully training data-driven models such as neural network (NN) or a Gaussian process (GP), as these are data-hungry compared to analytical models and usually do not extrapolate.

To overcome the limitations of data-driven modeling, recent works combine analytical with data-driven models resulting in so-called structured models or graybox models. In structured models, the target function, that is the parts of an analytical model that are difficult to model a-priori, is being approximated by a carefully designed data-driven model. By use of analytical prior knowledge, the data efficiency of a structured model is potentially improved by: i) Reducing the complexity of the target function; ii) Reducing the dimensionality of the target function; or iii) Building prior knowledge on physical properties of the target function into the data-driven model. The target function ideally contains all erroneous parts of the analytical model while its sample-complexity is smaller compared to simply learning the residual function resulting from the analytical model's output. Here, sample-complexity denotes the number of training points a data-driven model requires to successfully approximate the target function.

In general, finding a suitable candidate for a target function inside an analytical model is not trivial. In the following, we suggest that for sufficiently rigid bodies the majority of the analytical model errors arise from an inaccurate depiction of forces. As many of these forces are potentially uncertain and stochastic in nature, we propose a probabilistic structured dynamics model in which we approximate unknown forces inside an analytical dynamics model by a data-driven model. In particular, our work makes the following contributions:

- We propose a modelling framework that combines implicitly constrained analytical dynamics with data-driven regression. As a particular example, we combine a GP with analytical equations of motion of an implicitly constrained kinematic chain subsequently improving the data efficiency of the model compared to standard GP regression as well as guaranteeing that the model's predictions satisfy implicit acceleration-level constraints.
- We show that analytical parameters can be efficiently estimated alongside the GP's hyper-parameters using automatic differentiation with GPU acceleration. We analyze a proposed structured forward dynamics model using data from a high-dimensional robot arm whose end-effector touches a surface in PyBullet.
- By conditioning the resulting structured model on acceleration measurements, predictions of various analytical quantities can be obtained such as surface normal forces and the system's unconstrained acceleration.

1.1 Problem formulation

We assume that the following assumptions apply to a robot whose forward dynamics (FD) we aim to identify:

Assumption 1 *The underlying kinematics function of the robot arm and kinematic surface equation, as well as their parameters, are known a-priori.*

Assumption 2 *The bodies (aka links) of the mechanical system are sufficiently rigid such that the inverse of the inertia matrix M forms an accurate functional representation of how forces acting on the system cause acceleration $\ddot{q}$.*

As discussed in Section 1.3, these assumptions are commonly found in literature on nonlinear robot dynamics identification [11–15]. Given noisy input points $x_k = [q_k, \dot{q}_k, u_k]^\mathsf{T}$, acceleration measurements are modeled as

$$y_k(x_k) = \ddot{q}(x_k) + \epsilon_k \;\text{ with }\; \epsilon_k \sim \mathcal{N}\left(0, \Sigma_y\right), \tag{1}$$

where Σ_y denotes a diagonal matrix containing the measurements variances. Further, an informative data-set $\mathcal{D} = \{X, Y\}$ is available consisting of the input data vector $X = [x_1^\mathsf{T} \dots x_N^\mathsf{T}]^\mathsf{T}$ and the output data vector $Y = [y_1^\mathsf{T} \dots y_N^\mathsf{T}]^\mathsf{T}$.

The main objective of this work is, for a high-dimensional robot arm, to propose a modeling framework that enables the data-efficient prediction of $\mu^{\ddot{q}}_{X^\star}|\mathcal{D} = \mathbb{E}(\ddot{q}(\boldsymbol{x}_*)|\mathcal{D})$ and $K^{\ddot{q}}_{X^\star,X^\star}|\mathcal{D} = \text{cov}(\ddot{q}(\boldsymbol{x}_*)|\mathcal{D})$ at a test input $\boldsymbol{x}_*$ with a structured GP model $\hat{\ddot{q}} \sim \mathcal{GP}(m_{\ddot{q}}, K_{\ddot{q}})$. We refer to $\hat{\ddot{q}}$ as structured as this model results from combining analytical prior knowledge underlying the system's EOM (6) with a GP prior placed on the system's unknown forces $\hat{Q}_\text{U} \sim \mathcal{GP}(0, K_\text{U})$.

1.2 Forward dynamics of robot arms

In what follows, we omit the arguments of functions to avoid cluttered notation if these are clear from the context. Assume that the state of a robot arm is described by $q \in \mathbb{R}^{n_q}$, $\dot{q}$, $\ddot{q}$ being the joint angles, velocity, and acceleration, respectively. In this work, we limit the discussion to Lagrangian FD of robot arms as described by equations of motion (EOM) taking the form

$$\ddot{q} = f_{\mathrm{A}}(x; \theta_{\mathrm{A}}) = M^{-1}\left(Q + Q_z + Q_I\right), \tag{2}$$

with the analytical model parameters θ_{A}, control input $u \in \mathbb{R}^{n_u}$, symmetric and positive definite inertia matrix $M(q; \theta_{\mathrm{A}})$, torques acting on the unconstrained system $Q(x; \theta_{\mathrm{A}})$, torques caused by friction forces F_z acting on the end-effector $Q_z(x; \theta_{\mathrm{A}})$, and torques caused by surface normal forces F_I also acting on the end-effector $Q_I(x; \theta_{\mathrm{A}})$. The joint-related torques $Q(x; \theta_{\mathrm{A}}) = Q_D + Q_u + Q_G + Q_C$ consist of bias forces Q_C, gravitational forces Q_G, dissipative forces Q_D, and actuation forces Q_u. Often in literature $Q_u + Q_D$ is denoted by τ being the joint torques that are measurable in a robot joints. For kinematic trees, such as most robot arms, the FD $f_{\mathrm{A}}(x; \theta_{\mathrm{A}})$ are straightforwardly obtained using the articulated body algorithm. Alternatively, inverse dynamics (ID) seek a mapping from $\{q, \dot{q}, \ddot{q}\}$ to u. However, unlike ID, FD allow the direct simulation of a system's motion. In turn, one could use an accurate structured FD model to train an RL policy in simulation. Moreover, the largest noise usually lies on acceleration estimates denoting the output of FD models. In the case of GPs it is simpler to model noise on the outputs than inputs [16].

Implicitly constrained dynamics In the absence of end-effector forces, the coordinates q uniquely define every admissible state of the robot arm. However, the presence of Q_I enforces that the end-effector's motion during contact respects the holonomic surface constraints

$$c(q) = 0, \text{ with } c(q) : \mathbb{R}^{n_q} \to \mathbb{R}^m, \tag{3}$$

effectively reducing the set of admissible states the robot arm may acquire. For the robot arm depicted in Figure 1 we have that $m = 1$. Instead of working directly with (3), one can compute its second time-derivative to obtain a constraint equation that is linear in $\ddot{q}$, writing

$$A\ddot{q} = b, \tag{4}$$

with $A(q, \dot{q}; \theta_{\mathrm{A}}) : \mathbb{R}^{2n_q} \to \mathbb{R}^{m \times n_q}$, $b(q, \dot{q}; \theta_{\mathrm{A}}) : \mathbb{R}^{2n_q} \to \mathbb{R}^m$, and $m < n_q$. As detailed in [17], the part of all forces doing virtual work, $Q' \in Q$, and the implicit constraint forces Q_I doing no virtual work, must be in $Q' \in \mathsf{N}(A)$, and $Q_I \in \mathsf{R}(A^{\mathsf{T}})$, with the null space of $A \in \mathbb{R}^{m \times n_q}$ being defined as $\mathsf{N}(A) = \{\ddot{q} \in \mathbb{R}^{n_q} : A\ddot{q} = 0\}$, its range space as $\mathsf{R}(A) = \{b \in \mathbb{R}^m : \exists \ddot{q} \in \mathbb{R}^{n_q} \text{ such that } b = A\ddot{q}\}$, and further we have that $\mathbb{R}^{n_q} = \mathsf{R}(A^T) \oplus \mathsf{N}(A)$ and $\mathbb{R}^m = \mathsf{R}(A) \oplus \mathsf{N}(A^T)$ [18]. The fact that $Q_I \in \mathsf{R}(A^{\mathsf{T}})$ motivates its parametrization in terms of a vector-valued function λ, usually being referred to as Lagrange multipliers, writing $Q_I = A^\top \lambda$. As detailed in [19] one obtains an explicit formula for the Lagrange multipliers as

$$\lambda = (AM^{-1}A^T)^+ \left(b - AM^{-1}Q\right), \tag{5}$$

where the A^+ denotes the Moore-Penrose (MP) pseudo inverse of A. Subsequently, the EOM of the implicitly constrained system are obtained as

$$\ddot{q} = M^{-1}\left(Q_b + P\bar{Q}\right) = M^{-1}Q_b + H\bar{Q}, \tag{6}$$

with $\bar{Q} = Q + Q_z$, $Q_b(q, \dot{q}; \theta_{\mathrm{A}}) = A^T(AM^{-1}A^T)^+ b$, $Q_z(q, \dot{q})$ being in $\mathsf{N}(A)$, $P(q, \dot{q}; \theta_{\mathrm{A}}) = I_n - A^T(AM^{-1}A^T)^+ AM^{-1}$, and $H(q, \dot{q}; \theta_{\mathrm{A}}) = M^{-1}P$. As detailed in the Section 3 of the supplementary material, P denotes an oblique projection into $\mathsf{N}(A)$, with $P = P^2$ but $P \neq P^T$.

1.3 Related work

The identification of analytical model errors in a robot's EOM often form a metaphorical Gordian knot consisting of errors that arise from the interplay of wrong analytical parameters (*e.g.*, inertia parameters), wrong analytical functions (*e.g.*, friction forces), and in the worst case the analytical model class itself being an inaccurate depiction of the real dynamics (*e.g.*, describing flexible bodies with rigid-body dynamics). In what follows, we discuss recent works that combine data-driven with analytical modeling as well as the assumptions the resulting models place on the origin of the analytical errors.

Analytical parameter estimation Pioneering works on robot dynamics identification focused on the estimation of analytical parameters. For example, the well-known approach of Atkeson et al. [11] uses the linearity of a rigid robot arm's ID with respect to the dynamical parameters to estimate these via linear regression. With the increased popularity of libraries for automatic differentiation, recent works [12–15] use gradient-based optimization to estimate the physical parameters inside the analytical model from data. Such approaches work well if the robot arm is rigid and the forces acting on it are sufficiently known. However, the parameter estimation of an insufficient analytical representation of a robot's dynamics leads to physically inconsistent parameter estimates as well as unsatisfactory model accuracy. While we assume that the analytical model differs from the real dynamics, we also implement a differentiable analytical model in PyTorch such that we can estimate analytical parameters *alongside* the parameters of a data-driven model.

Building physical knowledge into neural networks As an alternative approach to analytical modeling, [20–23] model dynamics via the Euler-Lagrange equations in which a NN $n(\theta_M, x)$ approximates the system's Lagrangian or the entries of the inertia matrix M. In turn, such a Lagrangian NN increases the flexibility of the functional approximation of mass-related dynamic terms $\{M, Q_G, Q_C\}$ compared to analytical models while still respecting energy conservation. Lagrangian NNs also allow one to model ID (in the absence of end-effector forces) as $\hat{\tau} = \hat{M}(n)\ddot{q} - \hat{Q}_C(n) - \hat{Q}_G(n)$. These works often assume $Q_u \hat{=} \tau$ thereby neglecting the joint friction Q_D. If forces are the major sources of analytical errors and mass related quantities are not, then Lagrangian NNs add flexibility through the NN in the functional representation of a robot's dynamics where it might not be required. So far this hypothesis has not been refuted, as the works on Lagrangian NNs test their algorithms solely on pendulums or low-dimensional robot arms without end-effector contacts.

A recent branch of structured models combines analytical models of mass-related quantities and presumably known forces with data-driven modeling. For example, Lutter et al. [24] proposed to augment an analytical FD model with a NN approximating Q_τ. The parameters of the analytical model and NN are estimated jointly using automatic differentiation. Such an approach enables the incorporation of the dissipativity of Q_D into the NN. Recently, [10] and subsequently [25] combined an analytical simulation with a pre-trained NN – predicting measured joint torques Q_τ. Notably, the forces Q_I and Q_z have been modeled analytically as detailed in [9] while their physical parameters have been varied to robustify the trained control policy towards parameter uncertainties. Significantly, NN control policies that had been trained on such structured models achieved so far unseen robustness[1] of a quadruped's locomotion policy. These seminal works indicate that the gap between modeled and real dynamics (aka sim2real gap) can be closed using a conscious combination of an analytical model with a data-driven model. Albeit, for some applications it can be more practical to not rely on joint torque measurements and instead approximate Q_I and Q_z through a data-driven model using constraint knowledge as we propose in this work.

Building physical knowledge into Gaussian Processes NNs are a natural choice for the combination with analytical parametric models. However, standard NNs do not provide a measure of uncertainty with regards to the learned function which is problematic if noise and uncertainty significantly impact the robot's dynamics. For example, a friction force denotes a macroscopic abstraction of microscopic tribologic phenomena such that it is often difficult to model deterministically. In addition, depending on the robot's sensors, the changes in the joint torques due to elasticities and backlash are often not observed accurately and subsequently also introduce uncertainty into the dynamics. As shown in [6], the synthesis of a robot's control policy can significantly benefit from the availability of an uncertainty measure for a planned motion. A popular model for the identification of uncertain and noisy dynamics are Gaussian processes [8, 26–30]. As GPs allow for the incorporation of various model assumptions through the covariance function (kernel), they are often more data-efficient compared to NNs. From a probabilistic ML perspective, GPs transform the equations for Bayesian inference – usually involving the computation of complicated integrals – into linear algebraic equations. Yet, vanilla GP regression requires the computation of the inverse covariance matrix (in practice one uses the Cholesky decomposition) which demands a computational complexity of $\mathcal{O}((DN)^3)$ and a memory requirement of $\mathcal{O}((DN)^2)$ where D denotes the number of the GP's correlated outputs. In addition, the inversion of the covariance matrix is also prone to numerical problems and until recently only a few libraries existed to ease the implementation of

[1] https://www.youtube.com/watch?v=9j2a1oAHDL8

probabilistic non-parametric models. Despite these challenges, several models have been proposed that combine GP regression with analytical dynamics which we briefly discuss in the following.

Nguyen-Tuong et al. [31] used [11] as a linear parametric mean of a GP model (cf. [16, Chapter 2.7]). If Assumption 2 applies onto the robot arm, the GP in [31] effectively approximates the joint torques τ (in the absence of end-effector forces). As outlined in Section 2, we also approximate unknown forces through a GP. Albeit, structured GP regression is more challenging for FD compared to ID as a robot arm's FD are rarely linear with respect to analytical parameters.

Similarly to Lagrangian NNs, Cheng et al. [32] placed a GP on the Lagrangian inside a robot arm's Euler-Lagrangian ID. Yet, for GPs it is not clear how to formulate such a model for FD and to which extent the end-effector forces can be incorporated. A Lagrangian GP leverages that the derivative of a GP if it exists, is itself a GP [33]. Jidling et al. [34] emphasized that more generally GPs are closed under linear functionals. In turn, the authors proposed a linearly transformed GP such that its predictions fulfil a linear operator equation *e.g.*, the mechanical stress field inside a linear elastic material. Our work also transforms a GP by a linear operator, that is, the matrix-valued function H and in turn enforce that predictions respect the linear-affine equation (4).

Methodically, our work builds on the work of Geist and Trimpe [35] who suggested the combination of GP regression with implicitly constrained EOM. However, unlike [35], we suggest placing a GP on the unknown *forces* of the system rather than the unknown unconstrained *acceleration*. By doing so, we leverage the model knowledge underlying Assumption 2 to potentially improve the sample-efficiency of the proposed structured GP model as well as ease the inclusion of prior knowledge on forces into the structured GP. In addition, [35] only tested their model on low-dimensional systems such as a point-mass sliding over a surface. In turn, they did not run into the challenges that usually accompanies multi-output GP regression. In contrast, we propose a framework for learning high-dimensional robot dynamics. The proposed framework is significantly faster than [35] by resorting to automatic differentiation with GPU accelerated computation of gradients.

2 Semiparametric regression for forward dynamics with Gaussian processes

In this section, we extend the findings of Geist and Trimpe [35] and use knowledge on the parametric functions $\{M, A, b\}$ as stated in (2) and (4) to linearly transform a GP. Here, we leverage that analytical mechanics defines the acceleration of a system in terms of affine transformations of force functions which we approximate by a GP. To be able to train the resulting structured GP on high-dimensional data through GPU accelerated automatic differentiation, we suggest a framework that combines GP regression with multibody dynamics modeling in PyTorch.

2.1 Multi-output Gaussian processes

A Gaussian process defines a collection of random variables such that every finite set of these random variables are being normally distributed [16]. More concretely, we can model the FD $\boldsymbol{f} = [f_1 \dots f_n]^\mathsf{T}$ as a multi-task GP, writing $\hat{\boldsymbol{f}} \sim \mathcal{GP}(\boldsymbol{m}(\boldsymbol{x}), K(\boldsymbol{x}, \boldsymbol{x}'))$, where $\boldsymbol{m}(\boldsymbol{x})$ denotes a vector-valued mean function and $K(\boldsymbol{x}, \boldsymbol{x}')$ a matrix-valued kernel (covariance function), such that

$$\boldsymbol{m}(\boldsymbol{x}) = \begin{bmatrix} m_1(\boldsymbol{x}) & \dots & m_n(\boldsymbol{x}) \end{bmatrix}^\mathsf{T}, \quad K(\boldsymbol{x}, \boldsymbol{x}') = \begin{bmatrix} k_{1,1}(\boldsymbol{x}, \boldsymbol{x}') & \cdots & k_{1,n'}(\boldsymbol{x}, \boldsymbol{x}') \\ \vdots & \ddots & \vdots \\ k_{n,1}(\boldsymbol{x}, \boldsymbol{x}') & \cdots & k_{n,n'}(\boldsymbol{x}, \boldsymbol{x}') \end{bmatrix}, \tag{7}$$

where $m_j(\boldsymbol{x})$ is the scalar mean function of $f_j(\boldsymbol{x})$ and $k_{i,j}(\boldsymbol{x}, \boldsymbol{x}')$ corresponds to the kernel between $f_i(\boldsymbol{x})$ and $f_j(\boldsymbol{x}')$. A multi-task GP defines a multivariate normal distribution on a given input data set X as $\hat{\boldsymbol{f}}(X) \sim \mathcal{N}(\boldsymbol{\mu}_X, \Sigma_{X,X})$, with the mean vector and covariance matrix respectively

$$\boldsymbol{\mu}_X = \begin{bmatrix} \boldsymbol{m}(\boldsymbol{x}_1)^\mathsf{T} & \dots & \boldsymbol{m}(\boldsymbol{x}_N)^\mathsf{T} \end{bmatrix}^\mathsf{T}, \quad \Sigma_{X,X'} = \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1') & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_{N'}') \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_N, \boldsymbol{x}_1') & \cdots & K(\boldsymbol{x}_N, \boldsymbol{x}_{N'}') \end{bmatrix}. \tag{8}$$

For X, (1) reads $Y(X) = \hat{\boldsymbol{f}}(X) + \boldsymbol{\epsilon}_Y$, with $\boldsymbol{\epsilon}_Y \sim \mathcal{N}(\boldsymbol{0}\,, \Sigma_Y)$, with $\Sigma_Y = I_N \otimes \Sigma_y$. In turn, with the prediction locations $X' = [x_1'^\mathsf{T} \dots x_{N_\mathrm{p}}'^\mathsf{T}]^\mathsf{T}$, the joint distribution of $Y(X)$ and $\hat{\boldsymbol{f}}(X')$ reads

$$\begin{bmatrix} \hat{\boldsymbol{f}}(X') \\ Y(X) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{X'} \\ \boldsymbol{\mu}_X \end{bmatrix}, \begin{bmatrix} \Sigma_{X',X'} & \Sigma_{X',X} \\ \Sigma_{X,X'} & \Sigma_{X,X} + \Sigma_Y \end{bmatrix} \right). \tag{9}$$

Conditioning on the observations yields the predictive posterior distribution

$$\hat{\boldsymbol{f}}(X')|\mathcal{D}, \Sigma_Y \sim \mathcal{N}(\boldsymbol{\mu}^\star, \Sigma^\star), \quad \begin{aligned} \boldsymbol{\mu}^\star &= \boldsymbol{\mu}_{X'} + \Sigma_{X',X}\left(\Sigma_{X,X} + \Sigma_Y\right)^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_X), \\ \Sigma^\star &= \Sigma_{X',X'} - \Sigma_{X',X}\left(\Sigma_{X,X} + \Sigma_Y\right)^{-1}\Sigma_{X,X'}. \end{aligned} \tag{10}$$

Notably, GPs are closed under linear affine transformations [16] such that transforming a GP by a matrix of functions B yields also a GP as

$$B\boldsymbol{f} \sim \mathcal{GP}\left(B(\boldsymbol{x})\boldsymbol{m}(\boldsymbol{x}),\ B(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{x}')B^\mathsf{T}(\boldsymbol{x}')\right). \tag{11}$$

2.2 Combining data-driven with analytical modeling

In this section, we propose a framework that combines Gaussian process regression with parametric analytical modeling. To develop a thorough understanding of the underlying assumptions we first discuss the main building blocks of the analytical EOM (6) as also shown schematically in Figure 1. As the dynamics (6) are expressed in generalized coordinates, the entries of M, Q_G, Q_C, A, and b depend on *kinematic functions*. The kinematic functions themselves depend on the system's state $\{q, \dot{q}\}$ and kinematic parameters such as the length of the bodies. In practice, precise estimates for the kinematic parameters are obtained from computer-aided design (CAD) or kinematic parameter estimation techniques. In Cartesian space, an accurate model for (4) that is $\{A, b\}$ can be straightforwardly obtained by *e.g.*, using a camera system. In turn, (4) is obtained in generalized coordinates using kinematic coordinate transformations. Besides kinematic expressions, the *inertia matrix* M contains inertia parameters, namely the COG positions, inertias, and masses. We assume that a prior for the inertia parameters can be obtained using also CAD. Yet, for FD it is critical that we mitigate errors in inertial parameters θ_A as all forces in (6) are being multiplied by M^{-1}.

As pointed out in Section 1, it is often difficult to analytically describe force functions that arise from elasticities inside joints and friction phenomena. Subsequently, we divide the analytical force vector $\bar{Q}$ into a known part Q_K and an unknown part Q_U. The term *known* is defined in the following as the existence of an optimal θ_A, $\theta_\mathrm{A}^\star$, such that the error between a function and its analytical description approaches zero. Knowledge on $\theta_\mathrm{A}^\star$ can be unavailable a-priori such that these often need to be estimated from data. In turn, we directly model the generalized forces as a GP, writing

$$\hat{\bar{Q}} \sim \mathcal{GP}(Q_\mathrm{K}(x;\theta_\mathrm{A}),\ k_{Q_\mathrm{U}}(x,x';\theta_\mathrm{M})), \tag{12}$$

in which $k_{Q_\mathrm{U}}(x,x')$ denotes an appropriately chosen kernel function and θ_M its hyper-parameters. By inserting (12) into (6), we obtain a structured model for the system's FD similarly to (11) as

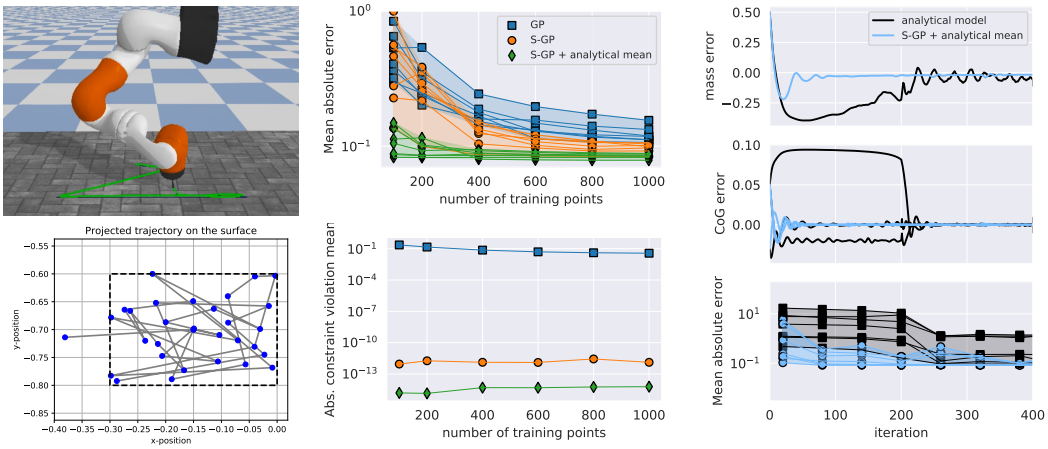
$$\hat{\ddot{q}} \sim \mathcal{GP}(m_{\ddot{q}},\ k_{\ddot{q}}), \tag{13}$$

with $m_{\ddot{q}} = M^{-1}Q_b + HQ_\mathrm{K}$ and $k_{\ddot{q}} = Hk_{Q_\mathrm{U}}H^\top$. The above model is itself a GP as $\ddot{q}$ follows from an affine transformation of $\bar{Q}$. To make predictions with (13), the GP conditional posterior formula (10) is used. Note that in (13), we placed the GP prior on the unknown forces instead of placing it on the systems unknown acceleration $M^{-1}Q_\mathrm{U}$ as initially proposed by [35]. Since M is known and its entries are usually nonlinear functions, learning a function after it has been transformed by M is likely to require a more expressive data-driven model to achieve a comparable prediction accuracy [17]. In addition, as elaborated in Section 4 of the supplementary material, a GP force prior can be designed as a sum of GPs which each model a different type of analytical force. In turn, the proposed structured model (13) enables a straightforward inclusion of prior knowledge on the system's unknown forces into the GP. Moreover, many dynamical quantities result from a linear transformation of the system's forces. A trained GP force prior (12) can provide predictions on other analytical processes such as the system's unconstrained dynamics or its Lagrange multipliers.

2.3 Type-II MLE of structured Gaussian processes

While a merit of the model presented in (13) forms its simplicity, the optimization of both analytical and kernel parameters poses a significant challenge that so far has not been properly addressed in prior work. In this work, we resort to Type-II maximum likelihood estimation (MLE). MLE forms a standard approach for estimating GP parameters. Given $p(Y|X;\theta) \sim \mathcal{N}(\mu_X^{\ddot{q}}, \Sigma_{X,X}^{\ddot{q}} + \Sigma_Y)$ resulting from (13), type-II MLE seeks a $\theta^\star = \{\theta_\mathrm{A}^\star, \theta_\mathrm{M}^\star\}$ minimizing the negative log likelihood, writing

$$\theta^\star = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left[\left(Y - \mu_X^{\ddot{q}}\right)^\top \left(\Sigma_{X,X}^{\ddot{q}} + \Sigma_Y\right)^{-1}\left(Y - \mu_X^{\ddot{q}}\right) + \log\left|\Sigma_{X,X}^{\ddot{q}} + \Sigma_Y\right|\right] + \frac{N}{2}\log 2\pi. \tag{14}$$



(a) (b) (c)

Figure 2: a) Robot arm in PyBullet (Top) and end-effector trajectory during collection of training data (Bottom). b) The hyper-parameters of GP FD models are trained on an increasing number of training points, then the mean absolute error for each joint dimension as well as acceleration-level constraint error on the test data set are compared. c) Comparison of parameter optimization results of an analytical model (black) with the proposed structured GP model (blue).

The left term inside the bracket in (14) assigns a cost to the data-fit while the log-determinant term penalises the function's complexity [16, p. 113]. Due to the inversion of $\Sigma^{\ddot{q}}_{X,X} + \Sigma_Y$, the computation of (14) scales with $\mathcal{O}((DN)^3)$. As (13) denotes a multi-output GP such that $D > 1$, the computational cost increases significantly compared to standard GP regression. Moreover, as pointed out in [16], the objective function in (14) is highly nonlinear such that if local optimization techniques are considered we need to restart the optimization several times with random initial parameter values. To solve (14) using restarts for high-dimensional dynamics in reasonable amounts of time, we require a fast computation of analytical parameter gradients. However, such an approach to (14) requires the computation of partial gradients of the analytical parametric function $\mu^{\ddot{q}}_X(x;\theta_\mathrm{A})$ as well as $\Sigma^{\ddot{q}}_{X,X}$ being a transformation of $k_{\ddot{q}}(x,x';\theta_\mathrm{M})$ by $H(x,\theta_\mathrm{A})$. Fortunately, recent ML libraries such as JAX [36] and PyTorch [37] allow the optimization of functions via automatic differentiation (building on AutoGrad [38]) in combination with libraries for GPU accelerated computation such as XLA. In turn, JAX and PyTorch provide a fast computation of analytical function gradients as long as the function is expressed in terms of an underlying numerical linear algebra library such as Numpy or PyTorch's torch functions. Therefore, we first implement a multi-body library using native PyTorch functions resulting in $\{A, b, M, Q_\mathrm{K}\}$ as well as $\{\mu^{\ddot{q}}_X, H\}$. Then, we combine these functions with a multi-output GP implementation provided by GPyTorch [39] to obtain the likelihood in (14). The optimization is done using automatic differentiation [40] and GPU acceleration resulting in a significant speed-up compared to numerical gradient approximation techniques.

3 Experimental Results

In this section, the proposed structured GP (13) is analyzed and compared to different baseline models. The chosen dynamical system is a KUKA arm with 7 rotational joints simulated in PyBullet [41] as depicted in Figure 2a. A detailed description of the experimental setup is given in Section 1 of the supplementary material. Data is collected by controlling the robot's end-effector along linear trajectories while pressing onto the surface. This task can be seen as an abstraction to a robot arm performing welding, cutting, or marking maneuvers on surfaces. The contact between the end-effector and the surface creates ideal-constraint forces, which avoids the end-effector penetrating the surface, as well as friction forces which are challenging to be modeled beforehand. While running the experiment, we measured the joint angles q and the control inputs u. The joint velocity $\dot{q}$ and acceleration $\ddot{q}$ are obtained by using a low-pass filter and appealing to numerical differentiation.

After data collection, we applied Farthest Point Sampling [42] to remove adjacent data points as well as reduce the data set's size. The post-processed data is split into a training and a test data set.

Figure 2b: Prediction accuracy and data efficiency As a baseline for comparison, we trained the parameters of (6) with $Q_z = 0$ as well as a feed-forward NN on ten thousand data points. The trained analytical model and the NN achieved on the test data set a root mean absolute error (MAE) ranging from $[0.1, 1.2]$ and $[0.1, 0.15]$, respectively. The analytical baseline model does not contain a function describing the surface friction which causes large prediction errors as illustrated in Section 2 of the supplementary material. For all GP models, we choose a squared exponential (SE) kernel. The 154 hyper-parameters of each GP were estimated according to (14) using Adam [43] without parameter constraints. Figure 2b illustrates the GPs' test-data MAE of each joint-dimension over an increasing size of the training data set. In this figure, the term "S-GP" refers to the model in (13) with $Q_{\mathrm{K}} = 0$ while the term "S-GP + analytical mean" assumes that an analytical model $Q_{\mathrm{K}} = Q_G + Q_C + Q_u$ with accurate analytical parameters is given. The structured GP models are compared to standard GP regression in which each acceleration function is modelled by a single independent SE GP. As shown in Figure 2b (Top), analytical prior knowledge improves the data efficiency of the GP. Moreover, the incorporation of implicit constraint knowledge in (13) significantly reduces the constraint error $A\hat{\ddot{q}} - b$ as illustrated in Figure 2b (Bottom). An illustration of different acceleration predictions made with these models is given in Section 2 of the supplementary material. As a consequence of Assumption 1 and 2, one can guarantee that trajectory predictions computed with (13) converge onto the surface (3) by adjusting b through Baumgarte stabilization [44] as detailed in Section 5 of the supplementary material.

Figure 2c: Joint estimation of inertia and kernel parameters Another important aspect of the proposed structured modeling framework forms the simultaneous estimation of θ_{M} and θ_{A}. To illustrate how θ_{M} and θ_{A} are estimated jointly, we train the same S-GP with $Q_{\mathrm{K}} = 0$ as in the previous experiment on thousand data points. Yet, instead of assuming that we obtained a good prior for θ_{A}, we now estimate the end-effector's inertia parameters alongside the GP's 154 hyper-parameters. For example, one could imagine that a tool such as a brush or milling machine is being fixed to the end-effector changing its inertia parameters. Figure 2c illustrates the optimization results. Compared to training the parameters of an analytical model without surface friction, the parameter estimates of the proposed S-GP converge faster while the prediction's MAE for all joint-dimensions improves.

4 Conclusion and future prospects

In this paper, we introduce a modeling framework that combines differentiable Lagrangian dynamics with data-driven modeling. This framework incorporates structural prior knowledge in form of kinematic, inertia and implicit constraint equations to improve the data efficiency of a GP model. Further, we illustrate that the hyper-parameters of the GP and parameters of an analytical model can be estimated jointly using automatic differentiation. The framework requires that we have access to an accurate kinematics description of the system as well as that its bodies are sufficiently rigid. Notably, we illustrate that the dynamics of high-dimensional robot arms whose end-effector is subject to forces can be estimated using multi-output GPs.

Particularly in the combination with GP regression, the proposed modeling framework opens up exciting research directions in the field of robotics modeling. So far, by letting the data-driven model approximate the forces inside an analytical model, the data efficiency of the model has been increased as we do not need to learn the inertia and constraint functions from scratch. Yet, as we emphasized in the introduction, structured modeling can also improve a model's data efficiency by decreasing the dimensionality of the target function. In particular, one can use knowledge on a robot's differential kinematics to approximate with a GP the end-effector forces $\{F_z, F_I\}$ being in the Cartesian space as shown in Figure 1.

For the sake of illustration, we chose a squared-exponential kernel. In practice, such a kernel might not suffice as it assumes that the process is stationary and smooth. Both of these assumptions will likely not be met in practice. Instead, one can parameterize kernels using NNs [45, 46]. Further, rather than assigning all states as inputs to the kernel force prior, one can use additional prior model knowledge and feed only physically relevant states as kernel inputs.

References

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

[2] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter. Whole-body mpc for a dynamically stable mobile manipulator. *IEEE Robotics and Automation Letters*, 4(4):3687–3694, 2019. doi:10.1109/LRA.2019.2927955.

[3] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe. Safe and fast tracking on a robot manipulator: Robust mpc and neural network control. *IEEE Robotics and Automation Letters*, 5(2):3050–3057, 2020. doi:10.1109/LRA.2020.2975727.

[4] S. Kleff, A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti. High-Frequency Nonlinear Model Predictive Control of a Manipulator. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, May 2021. URL https://hal.archives-ouvertes.fr/hal-02993058.

[5] R. S. Sutton, A. G. Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[6] M. Deisenroth and C. 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.

[7] P. Parmas, C. E. Rasmussen, J. Peters, and K. Doya. PIPPS: Flexible model-based policy search robust to the curse of chaos. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4065–4074. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/parmas18a.html.

[8] A. Doerr, C. Daniel, D. Nguyen-Tuong, A. Marco, S. Schaal, M. Toussaint, and S. Trimpe. Optimizing long-term predictions for model-based policy search. In *Conference on Robot Learning*, pages 227–238, 2017.

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

[10] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

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

[12] F. D. Ledezma and S. 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.

[13] F. D. Ledezma and S. 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.

[14] G. Sutanto, A. Wang, Y. Lin, M. Mukadam, G. Sukhatme, A. Rai, and F. 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.

[15] M. Lutter, J. Silberbauer, J. Watson, and J. Peters. Differentiable physics models for real-world offline model-based reinforcement learning. *arXiv preprint arXiv:2011.01734*, 2020.

[16] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[17] A. R. Geist and S. Trimpe. Structured learning of rigid-body dynamics: A survey and unified view from a robotics perspective. *GAMM Mitteilungen*, 44(2):34, 2021.

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

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

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

[21] M. Lutter, K. Listmann, and J. 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.

[22] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[23] J. K. Gupta, K. Menda, Z. Manchester, and M. 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] M. Lutter, J. Silberbauer, J. Watson, and J. Peters. A differentiable newton euler algorithm for multi-body model learning. *arXiv preprint arXiv:2010.09802*, 2020.

[25] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020.

[26] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith. Dynamic systems identification with Gaussian processes. *Mathematical and Computer Modelling of Dynamical Systems*, 11(4): 411–424, 2005.

[27] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen. Bayesian inference and learning in Gaussian process state-space models with particle MCMC. In *Advances in Neural Information Processing Systems*, pages 3156–3164, 2013.

[28] C. L. C. Mattos, A. Damianou, G. A. Barreto, and N. D. Lawrence. Latent autoregressive Gaussian processes models for robust system identification. *IFAC-PapersOnLine*, 49(7):1121–1126, 2016.

[29] S. Eleftheriadis, T. Nicholson, M. Deisenroth, and J. Hensman. Identification of Gaussian process state space models. In *Advances in neural information processing systems*, pages 5309–5319, 2017.

[30] A. Doerr, C. Daniel, M. Schiegg, N.-T. Duy, S. Schaal, M. Toussaint, and S. Trimpe. Probabilistic recurrent state-space models. volume 80 of *Proceedings of Machine Learning Research*, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[31] D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Schölkopf. Learning inverse dynamics: A comparison. In *Advances in Computational Intelligence and Learning: Proceedings of the European Symposium on Artificial Neural Networks*, pages 13–18. d-side, Apr. 2008.

[32] C.-A. Cheng, H.-P. Huang, H.-K. Hsu, W.-Z. Lai, and C.-C. Cheng. Learning the inverse dynamics of robotic manipulators in structured reproducing kernel hilbert space. *IEEE transactions on cybernetics*, 46(7):1691–1703, 2015.

[33] E. Solak, R. Murray-Smith, W. E. Leithead, D. J. Leith, and C. E. Rasmussen. Derivative observations in gaussian process models of dynamic systems. 2003.

[34] C. Jidling, N. Wahlström, A. Wills, and T. B. Schön. Linearly constrained gaussian processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1215–1224, 2017.

[35] A. Geist and S. 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.

[36] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *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.

[38] D. Maclaurin, D. Duvenaud, and R. P. Adams. Autograd: Reverse-mode differentiation of native python. In *ICML workshop on Automatic Machine Learning*, 2015.

[39] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. Gpytorch: Black-box matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[40] R. E. Wengert. A simple automatic derivative evaluation program. *Commun. ACM*, 7(8): 463–464, Aug. 1964. ISSN 0001-0782. doi:10.1145/355586.364791. URL https://doi.org/10.1145/355586.364791.

[41] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[42] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5105–5114, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[43] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, Dec. 2014. URL http://arxiv.org/abs/1412.6980. arXiv: 1412.6980.

[44] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer methods in applied mechanics and engineering*, 1(1):1–16, 1972.

[45] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

[46] S. W. Ober, C. E. Rasmussen, and M. van der Wilk. The promises and pitfalls of deep kernel learning. *arXiv preprint arXiv:2102.12108*, 2021.