Physics-informed Regression of Implicitly-constrained Robot Dynamics

Von der Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik
der Universität Stuttgart zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von

Andreas René Geist

aus Hamburg

Hauptberichter:	Prof. Dr.-Ing. Frank Allgöwer
Mitberichter:	Prof. Dr. Sebastian Trimpe
	Prof. Dr. C. David Remy

Tag der mündlichen Prüfung: 1st June 2022

Institut für Systemtheorie und Regelungstechnik

Universität Stuttgart

2022

Declaration | Erklärung

I hereby declare that the presented work was accomplished independently and only by the use of the specified resources and literature.

Ich erkläre hiermit, dass die vorgestellte Arbeit unabhängig und nur unter Verwendung der angegebenen Ressourcen und Literatur durchgeführt wurde.

Stuttgart, April 2022

Andreas René Geist

Acknowledgements

In what follows, I would like to express gratitude to the many people who provided support and encouragement in my life. First of all, I am deeply grateful to my advisor Prof. **Sebastian Trimpe** who patiently supported me. Sebastian's advice helped me at countless points throughout my Ph.D. studies and profoundly shaped my way of reasoning. I would like to extend this level of gratitude to **Jonathan Fiene** for sharing his expertise on mechatronics and who helped me to stay focused on research when my group transitioned from Stuttgart to Aachen. I am sincerely grateful to Prof. **Frank Allgöwer** who shared valuable advice throughout my Ph.D., was part of my IMPRS-IS thesis advisory committee, and also acted as the main examiner of my Ph.D. thesis. Additional thanks goes to Prof. **Marc Toussaint** for being part of my IMPRS-IS thesis advisory committee, Prof. **David Remy** for being on my dissertation committee, and Prof. **Peter Eberhard** for serving as the chair of the committee.

My deepest gratitude extends to Prof. **Edwin Kreuzer** and Prof. **Robert Seifried** for igniting a lasting fascination for mechanics and for their help when I decided to pursue my doctoral studies. Moreover, during almost six years as a student assistant at the Institute of Mechanics and Ocean Engineering, **Marc-André Pick** provided indispensable support during my studies. I am also sincerely grateful to **Eugen Solowjow** who spurred my interest in statistics and probability theory while also introducing me to the arts of academic writing and basketball. Further, I would like to extend my sincere thanks to **Daniel Dücker** for his advice on robot design and **Andreas Hansen** who provided mentor-ship during my time at the University of California Berkeley.

To my colleagues and friends at the Intelligent Control Systems Group, namely **Friedrich Solowjow**, **Alexander von Rohr**, **Christian Fiedler**, **Mona Buisson-Fenet**, **Dominik Baumann**, **Alonso Marco Valle**, and **Steve Heim**, I want to express gratitude for the many joyful moments. Friedrich Solowjow and Alexander von Rohr considerably helped to shape the direction of my Ph.D. research and improve my backhand serve in table tennis. Representing all the excellent students I had the pleasure to supervise, I would like to thank **Zheng Jia** and **Lucas Rath**. Additional thanks shall also go to my friends and colleagues at the Max Planck Institute for Intelligent Systems as well as the University of Stuttgart. In particular, I want to thank **Henning Schlüter**, **Lukas Schwenkel**, **Sebastian Schlor**, **Felix Grimminger**, **Felix Grüninger**, **Julian Martus**, and **Naomi Tashiro**.

I am immensely grateful to my family. My father **Karlheinz** spent many days spurring in me a fascination for the natural sciences. My mother **Beatrix** filled my childhood with northern German joie de vivre. My brother **Jan** brought countless joyful memories into my life that I would not have wanted to miss. My wife **Vaishnavi** provided love and support that at times spanned continents.

Stuttgart, August 2022
Andreas René Geist

Abstract

The ability to predict a robot's motion through a dynamics model is critical for the development of fast, safe, and efficient control algorithms. Yet, obtaining an accurate robot dynamics model is challenging as robot dynamics are typically nonlinear and subject to environment-dependent physical phenomena such as friction and material elasticities. The respective functions often cause analytical dynamics models to have large prediction errors. An alternative approach to analytical modeling forms the identification of a robot's dynamics through data-driven modeling techniques such as Gaussian processes or neural networks. However, solely data-driven algorithms require considerable amounts of data, which on a robotic system must be collected in real-time. Moreover, the information stored in the data as well as the coverage of the system's state space by the data is limited by the controller that is used to obtain the data. To tackle the shortcomings of analytical dynamics and data-driven modeling, this dissertation investigates and develops models in which analytical dynamics is being combined with data-driven regression techniques. By combining prior structural knowledge from analytical dynamics with data-driven regression, physics-informed models show improved data-efficiency and prediction accuracy compared to using the aforementioned modeling techniques in an isolated manner.

In the first part of this dissertation, we give a concise presentation of important concepts that give rise to structural knowledge in rigid body dynamics. In particular, we discuss how the solution to robot dynamics formulations follows from projection operations or Lagrangian optimization. These concepts enable us to give a detailed overview of how the equations of motion of a robotic system emerge from fundamental principles, and how these principles provide valuable knowledge on the direction of dissipative forces, energy conservation, and the length of constraint forces. In this exposition of rigid body mechanics, a special emphasis is placed on implicit constraint equations, which despite their utility for robot dynamics modeling are rarely detailed in machine learning literature. To this effect, we leverage the Udwadia-Kalaba equations of motion and show their connection to the derivation of implicitly-constrained dynamics using Lagrange multipliers.

In the second part, we use the preceding discussion on rigid body dynamics to develop a unified view of the errors inside an analytical dynamics model. Moreover, we look at analytical dynamics models as parametric networks, which like statistical models give rise to estimation errors and approximation errors. Moreover, we develop the notion of latent model errors that arise inside an analytical dynamics model. This discussion enables us to systematically categorize existing literature on physics-informed regression of robot dynamics into analytical dynamics modeling, analytical output residual modeling, and analytical latent modeling. In the latter modeling approach, a data-driven model approximates a latent unknown function inside an analytical dynamics model, whereas in analytical output residual modeling, the output residual of an analytical dynamics model is approximated by a data-driven model.

In the third part, we propose a framework for analytical latent modeling in which Gaussian process regression is combined with implicitly-constrained rigid body dynamics.

In this framework, the unknown forces (or in an alternative formulation, the acceleration that these forces cause) are modeled by a multi-output Gaussian process. This Gaussian process is linearly transformed by rigid body dynamics to yield probabilistic predictions of the system's acceleration that by design must fulfill implicit constraints. Moreover, as many latent functions inside a rigid body dynamics model are obtained from an affine transformation of the unknown forces, we show how to predict the force inside a robot arm's end-effector using acceleration measurements.

The gathered insights on rigid body dynamics modeling and physics-informed regression benefited from and contributed to, the design of a novel reaction wheel-driven unicycle robot for learning control. The last part of this dissertation revolves around this new testbed which we named "Wheelbot". We detail the Wheelbot's mechanical and electronics design as well as propose a first state estimator and balancing controller. Moreover, we demonstrate the Wheelbot's ability to jump on its wheel from any initial position. Finally, we discuss potential research questions on physics-informed robot regression that may be investigated on this novel robotic testbed.

Zusammenfassung

Deutscher Titel der Dissertation:
Physikalisch Strukturierte Regression Implizit-eingeschränkter Roboter Dynamiken

Die Fähigkeit, die Bewegung eines Roboters anhand eines Dynamikmodells vorherzusagen, ist für die Entwicklung schneller, sicherer und effizienter Regelungsalgorithmen von entscheidender Bedeutung. Es ist allerdings in vielen Fällen eine große Herausforderung ein genaues Modell der Roboterdynamik zu erhalten, da diese oft nichtlinear ist und von umgebungsabhängigen physikalischen Phänomenen wie Reibung und Materialelastizität beeinflusst wird. Solche umgebungsabhängigen Funktionen führen häufig dazu, dass analytische Dynamikmodelle große Vorhersagefehler aufweisen. Ein alternativer Ansatz zur analytischen Modellierung besteht in der Identifizierung der Roboterdynamik durch datengesteuerte Modellierungstechniken wie Gaußsche Prozesse oder neuronale Netze. Ausschließlich datengesteuerte Algorithmen erfordern jedoch erhebliche Datenmengen, die bei einem Robotersystem in Echtzeit gesammelt werden müssen. Darüber hinaus werden die in den Daten enthaltenen Informationen sowie die Abdeckung des Systemzustandsraums durch die Daten, durch den zur Datenerfassung verwendeten Regler begrenzt. Um die Schwachstellen analytischer Modelle und datengesteuerter Modellierung zu beheben, werden in dieser Dissertation Modelle untersucht und entwickelt, bei denen die analytische Dynamik mit datengesteuerten Regressionsverfahren kombiniert wird. Durch die Kombination von strukturellem Vorwissen aus der analytischen Dynamik mit datengetriebenen Modellen zeigen physikalisch informierte Modelle eine verbesserte Dateneffizienz und Vorhersagegenauigkeit auf im Vergleich zur isolierten Verwendung der vorgenannten Modellierungstechniken.

Im ersten Teil dieser Dissertation werden wichtige Konzepte, die zu strukturellem Wissen in der Starrkörperdynamik führen, kurz vorgestellt. Insbesondere erörtern wir, wie die Formulierung einer Roboterdynamikfunktion aus Projektionsoperationen oder Lagrangescher Optimierung hervorgeht. Diese Konzepte ermöglichen es uns, einen detaillierten Überblick darüber zu geben, wie die Bewegungsgleichungen eines Robotersystems auf Basis grundlegender Prinzipien, wertvolles Wissen über die Richtung dissipativer Kräfte, die Energieerhaltung und die Länge der Zwangskräfte liefern. In dieser Darstellung der Starrkörpermechanik wird ein besonderer Schwerpunkt auf die impliziten Zwangsgleichungen gelegt. Zu diesem Zweck nutzen wir die Bewegungsgleichungen in der Udwadia-Kalaba Formulierung und zeigen ihre Verbindung zur Herleitung der implizit beschränkten Dynamik mit Hilfe von Lagrange-Multiplikatoren.

Im zweiten Teil nutzen wir die vorangegangene Diskussion über die Starrkörperdynamik, um eine einheitliche Sichtweise auf die Fehler innerhalb eines analytischen Dynamikmodells zu entwickeln. In dieser Sichtweise betrachten wir analytische Dynamikmodelle als parametrische Netzwerke, die wie statistische Modelle zu Schätz- und Approximationsfehlern führen. Darüber hinaus entwickeln wir den Begriff der latenten Modellfehler, die innerhalb eines analytischen Dynamikmodells auftreten. Anhand dieser Diskussion können wir die

vorangegange Literatur zur physikalisch informierten Regression von Roboterdynamiken systematisch in die analytische Dynamikmodellierung, die analytische Ausgangs-Residuen-Modellierung und die analytische latente Modellfehler-Modellierung einteilen. Bei letzterem Modellierungsansatz approximiert ein datengetriebenes Modell eine latente unbekannte Funktion innerhalb eines analytischen Dynamikmodells, während bei der einfacheren analytischen Ausgangs-Residuen-Modellierung das Residuum eines analytischen Dynamikmodells durch ein datengetriebenes Modell approximiert wird.

Im dritten Teil schlagen wir ein neues Analysemodell für die analytische latente Modellfehler Modellierung vor, in dem Gauß-Prozess-Regression mit implizit eingeschränkter Starrkörperdynamik kombiniert wird. In diesem Rahmen werden die unbekannten Kräfte (oder, in einer alternativen Formulierung, die Beschleunigung, die diese Kräfte verursachen) durch einen mehrdimensionalen Gauß-Prozess modelliert. Dieser Gauß-Prozess wird durch die Starrkörperdynamik linear transformiert, um probabilistische Vorhersagen über die Beschleunigung des Systems zu erhalten. Da viele latente Funktionen innerhalb eines Starrkörperdynamikmodells aus einer affinen Transformation der unbekannten Kräfte gewonnen werden, zeigen wir außerdem, wie man die Kraft im Endeffektor eines Roboterarms anhand von Beschleunigungsmessungen vorhersagen kann.

Die gewonnenen Erkenntnisse über physikalisch-informierte Regression trugen zu der Entwicklung des "Wheelbot" bei, einem neuartigen reaktionsradgetriebenen Einradroboter für die Entwicklung lernender Dynamikmodelle und Regler. Im letzten Teil dieser Arbeit, wird das mechanische und elektronische Design des Wheelbots beschrieben und ein erster Entwurf der Zustandsschätzer und Regler vorgeschlagen. Außerdem demonstrieren wir die Fähigkeit des Wheelbots, aus jeder beliebigen Ausgangsposition auf sein Rad zu springen. Schließlich erörtern wir potenzielle Forschungsfragen zur physikinformierten Roboterregression, die auf diesem neuartigen Robotertestand in Zukunft untersucht werden können.

Acronyms

AD	automatic differentiation
ALM	analytical latent modeling (cf. Section 4.4)
APN	analytical parametric network (cf. Section 4.2)
ARM	analytical (output) residual modeling (cf. Section 4.3)
COG	center of gravity
DELAN	deep Lagrangian neural network
DOF	degrees of freedom
EOM	equations of motion
FD	forward dynamics
ID	inverse dynamics
KKT	Karush-Kuhn-Tucker
GP	Gaussian process
GP^2	Gauss principle adhering Gaussian process (cf. Section 5.1.2)
MP	Moore-Penrose
NN	neural network
RNEA	recursive Newton-Euler algorithm
SVD	singular value decomposition
SE	squared-exponential

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Before stepping onto a frozen lake, our brain must anticipate a whole host of physical phenomena that determine whether we slip and subsequently fall or stay balanced and start walking. In the blink of an eye, the human brain imagines the interaction between a shoe and an icy surface, while also predicting how to flex the leg's muscles such that the desired movement is executed. The difference between a child stepping onto an icy surface for the first time and an ice skating champion are the years of experience and training. During these years, the professional athlete not only improved muscle strength and sensorimotor coordination, but also developed intuition about how certain muscle movements steer the body through space. It is this accurate prediction of motion that enables an actuated dynamical system to perform maneuvers that otherwise remain unattainable.

A particularly interesting class of mechanical systems in which the range of potential applications is often limited by the accuracy of the motion predictions are *robots*. Over the last decades, the field of robotics flourished owing to advancements in electronics, software, and control engineering. Robots such as the Boston Dynamic's "Spot" and "Atlas", and ETH Zürich's "ANYmal" (Hutter et al., 2016) have demonstrated impressive feats of agility. These works show that what a robot can do is not merely determined by the cost of its hardware, the strength of its actuators, or computation power, but foremost by its control policy. The computation of a control policy is closely intertwined with the prediction of the system's motion. In mathematics and physics, the description of a system's motion over time is described through a *dynamics function* $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$, which in its general form is denoted in this work as

$$y = f(x), \tag{1.1}$$

with the input vector $x \in \mathbb{R}^{n_x}$ and output vector $y \in \mathbb{R}^{n_y}$. The acquisition of an accurate model $\hat{f}(x, \theta_\mathrm{A})$ – where θ_A denotes the model's parameters – is by no means trivial on real-world robotic systems. Due to unmodelled physical phenomena such as friction, body elasticities, and motor dynamics, the dynamics model may deviate from the system's real dynamics by an error function

$$\epsilon(x, \theta_\mathrm{A}) = f(x) - f(x, \theta_\mathrm{A}). \tag{1.2}$$

This error function between a system's simulation and its real dynamics is of such utmost importance to robotics in particular and engineering in the whole that it received its own nickname being called the *sim2real gap* (Höfer et al., 2021).

With new robot designs aiming at reducing costs (Grimminger et al., 2020), increasing compliance for operation near humans, improving energy efficiency by leveraging natural dynamics (Badri-Spröwitz et al., 2022), or using soft bodies to unlock new potential

applications, closing the sim2real gap on the increasingly complex and intricate robot systems is more than ever a significant challenge.

Approaches that model the dynamics of robotic systems either evolve around analytical mechanics or data-driven modeling. As we will explore in this work, both approaches have significant drawbacks. These drawbacks motivated a few recent works to combine physics with data-driven learning under the collective heading of "physics-informed machine learning" also referred to as structured learning, gray-box modeling, or hybrid modeling. To this day, the existing works that combine physics models with machine learning, typically see the physics model as given while all attention is placed on the design of the machine learning algorithm which seeks to approximate the physics model's errors. However, if we do not question where and why the errors arise inside the physics model, we disregard a rich source of information that could prove utile towards closing the sim2real gap.

In this work, we explore existing approaches in physics-informed regression of robot dynamics. By focusing on implicitly constrained rigid body dynamics, we provide a systematic view of the potential errors arising in analytical mechanics models and how to approximate these with machine learning.

> *In particular, we look at the errors of a dynamics model not as merely given, but instead investigate where these errors arise inside rigid body dynamics models and how this knowledge can benefit the design of physics-informed machine learning algorithms.*

Further, we want to encourage the reader to envision physics models as parametric functions that depending on their parameters span a function space. Similarly, machine learning models such as neural networks or Gaussian processes also span a function space. Albeit, in comparison to physics models, the function space that machine learning models span can be straightforwardly enlarged by *e.g.*, adjusting the number of network layers in a neural network or resorting to a different covariance function in a Gaussian process. However, increasing the representational power of these models comes with the caveat of also requiring more data to find a point in this space which sufficiently reduces the model's error (Von Luxburg and Schölkopf, 2011). If we want to find such a point for a neural network, we need to find a point in the model's parameter space. Finding this particular point is metaphorically spoken a search for the needle in the haystack where the amount of hay one has to search through corresponds to the function space that the model spans. To follow the needle in a haystack analogy, prior knowledge from physics may enable us to alter the haystack in which we need to search for a good model approximation.

> *In this work, we argue that a significant part of a physics model's errors often arises in a potentially simpler and better-understood function space such as the space of all possible friction functions. In this case, it is more data-efficient to learn these latent errors before they have been transformed by known nonlinear transformations arising from rigid body dynamics.*

Finally, we present a design of a novel robotic testbed for nonlinear and learning control.

Dynamics functions in robotics

The dynamics of robotic systems are typically described in terms of a vector of control inputs $u(t)$, which for example, can include the electric currents that are sent to motors in a robot's joint, as well as vectors of position, velocity, and acceleration denoted as $q(t)$, $\dot{q}(t)$ and $\ddot{q}(t)$ respectively. In what follows, we generally omit the dependency of a function on variables, writing $f(x) := f$, if this is clear from the context. A common approach for the computation of control inputs in relation to its motion variables is *inverse dynamics* (ID)

$$u = f_u(q, \dot{q}, \ddot{q}). \tag{1.3}$$

Inverse dynamics yields the control input that is required to maintain the motion variables $\{q, \dot{q}, \ddot{q}\}$. In turn, a common approach for controlling a robot is to plan a desired path $\{q_{\text{des}}, \dot{q}_{\text{des}}, \ddot{q}_{\text{des}}\}_{i=1}^{N_q}$ and then compute the feed-forward control inputs via (1.3). These feed-forward control inputs can be combined with a feedback controller, which ensures that the system remains close to the desired path despite errors in (1.3).

For complicated maneuvers such as a robot playing table tennis (Koç et al., 2018) or a quadrupedal robot recovering from a fall (Hwangbo et al., 2019), it is laborious to engineer a path planning algorithm by hand. Instead, one can search for a viable control input as the solution to an optimization problem that seeks to find extrema of a performance index. Such a performance index is also referred to as the objective function in optimization theory, loss in statistics, cost function in control theory, or reward function in reinforcement learning. For example, the performance index can be designed such that an optimal control signal $\{u_i\}_{i=1}^{N_u}$ minimizes the control energy as well as the Euclidean distance to a desired state $\{q_{\text{des}}, \dot{q}_{\text{des}}, \ddot{q}_{\text{des}}\}$. Typically, such a performance index heavily depends on the system's *transition dynamics*, which is a discrete-time function modeling the effect of the control input at a given state onto the next state, written as

$$x_{k+1} = f_T(x_k, u_k) \tag{1.4}$$

with discrete input vector $x_k = [q_k, \dot{q}_k]$ at discrete time step k. In robotics, the motion variables are often chosen such that $\dot{q} = \frac{dq}{dt}$ and $\ddot{q} = \frac{d^2q}{dt^2}$, and the transition dynamics can be obtained through the numerical integration of a differential equation, the so-called *forward dynamics* (FD)

$$\ddot{q} = f_{\ddot{q}}(q, \dot{q}, u). \tag{1.5}$$

Numerous variations of the aforementioned dynamics models exist. For example, one may also find a robot's path as a solution to an optimization problem which is then fed to a model of the robot's ID. Also, in many data-driven dynamics models such as (Hwangbo et al., 2019) and (Doerr et al., 2017), a sequence of states is chosen as both the inputs and outputs of the dynamics model. Nevertheless, these approaches rely heavily on a dynamics model that, in some form or another, describes the motion of the system.

Table 1.1: Pros and cons of analytical rigid body models compared to data-driven models.

Physics models	**Data-driven models**
+ Data-efficient	- Data-hungry
+ Human-interpretable	- Usually black-box
+ Out-of-sample generalization	- Usually data-point interpolation
- Large prediction errors	+ Flexible function approximations
- Expert knowledge required	+ Less prior knowledge required
○ Deterministic	○ Possibly probabilistic

1.1 Motivation

To grasp the problems that underlie the computation of robot dynamics, we first discuss what distinguishes a physics model from a data-driven model. Subsequently, we examine data in the context of robotics as a scarce resource that may not be as straightforward to acquire as it may initially seem. The main differences between analytical dynamics and data-driven models are summarized in Table 1.1. The following discussion takes text passages from (Geist and Trimpe, 2021) and (Rath et al., 2021).

Physics-based Models

Rigid body dynamics models are based on physical principles underlying the motion of mechanical systems, *e.g.*, Newton's axioms of motion. These axioms spawn dynamics equations such as the Newton-Euler equations and the Euler-Lagrange equations, as detailed in Section 3. Analytical dynamics models are combinations of numerous physically motivated functions such as coordinate transformations, forces, and the inertia matrix, which depend on physical parameters such as the mass of a rigid body, a friction coefficient, the length of a kinematic link, or the stiffness coefficient of a spring. The numerous functions that form an analytical model can usually not be observed individually and are thus referred to 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, in real systems, physical phenomena such as *friction*, *damping*, and *contacts* aggravate the accurate identification of an analytical dynamics model. Furthermore, other physical phenomena 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 (Ting et al., 2006) which we discuss in detail in Section 4.3.

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 to be too

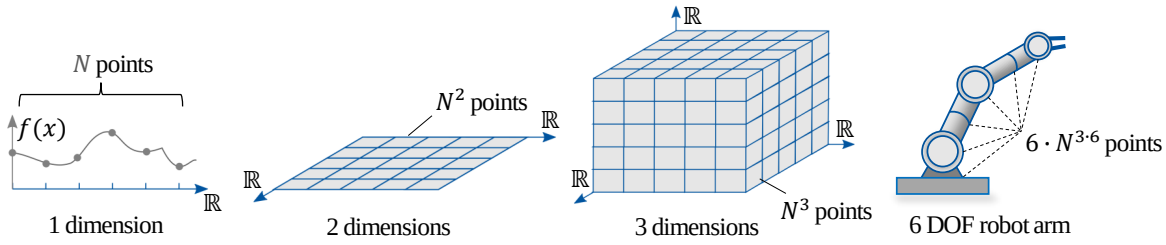


Figure 1.1: An illustration of the curse of dimensionality.

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, a process 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). Training hyperparameters of a kernel determines an entire population of functions while training the parameters of a parametric model usually determines a specific function approximation. Data-driven models contrast analytical models as their parameter estimates often do not yield a physically insightful interpretation. Moreover, the sheer amount of parameters and interlinked latent functions in many data-driven models eludes human comprehension. Thus, they are commonly referred to as black-box models.

Challenges in Robot Dynamics Identification

Machine learning and NNs in particular have achieved impressive breakthroughs in areas such as text-to-speech, language translation, image classification, 3D reconstruction, art generation, and even playing computer games. Typically, the data for solving related problems can either be scraped from the internet, generated by an accurate simulation, or readily available in large-scale datasets such as ImageNet (Deng et al., 2009). In comparison, the identification of robot dynamics challenges solely data-driven regression models by the sheer complexity of the target function in relation to the properties and available amount of the data. The main concern with the identification of dynamics via data-driven models is *sample complexity*, i.e., the number of training points a data-driven model requires to approximate a function sufficiently. The sample complexity of a data-driven model that approximates robot dynamics is determined by the following aspects:

Curse of dimensionality As initially pointed out by Bellman (1961), the number of points required to cover a space with an equidistant grid grows exponentially with the number of dimensions of the space (Nelles, 2013, p. 190). This property of multi-dimensional functions is referred to as the "curse of dimensionality". As an example, we consider a robot arm with six actuated revolute joints as illustrated in Figure 1.1. An ID model of this robot has at least six output variables (observed joint torques) and at least 18 input variables (position, velocity, and acceleration). Assuming the dataset forms an equidistant grid over the input space with just 5 points in each direction, the resulting dataset has $6 \cdot 5^{18} \approx 22,9 \cdot 10^{12}$ data points. In practice, the curse of dimensionality is slightly alleviated as not every region of the state space is relevant or can be reached, some process dimensions

are correlated, and other dimensions are redundant. Nevertheless, a data-driven model may significantly benefit if the dimensionality of the problem can be reduced.

Properties of dynamics The dynamics of quadrupedal robots, as well as robot arms, are nonlinear. As detailed in Chapter 3, dynamics functions are the result of a superposition and nonlinear transformation (*e.g.*, through kinematics) of nonlinear functions (*e.g.*, friction forces) that may each have a different scale in their input and output dimension. Moreover, dynamics functions may be discontinuous or even jump if being subject to friction phenomena or contacts. In turn, the prediction accuracy of a data-driven model may considerably deteriorate at points that lie outside the convex hull of the training points. Moreover, dynamics may vary over time and be subject to a considerable amount of noise. These aspects increase the amount of data that a data-driven model requires to successfully approximate a robot's dynamics.

While the above points may increase the amount of data required to learn a dynamics function, they would not pose a problem if enough data is available in the first place. After all, many data-driven models such as NNs (Hornik, 1993) or a GP with a squared exponential kernel are universal function approximators that, given sufficient data closely approximate many different types of target functions. Yet, such approximation guarantees do not specify the amount of data required for learning. Therefore, data collection comprises another key aspect that renders robot dynamics identification particularly challenging due to the problems listed below.

Real time constraints On physical systems, data has to be collected in real time while the system is subject to wear and tear. In turn, acquiring a data set requires a considerable amount of time and may result in high financial costs. Some works that rely on data of a robot's dynamics to synthesize a controller suggest training the data-driven controller in simulation. This tackles real time constraints to a certain extent as data collection can be significantly sped up. However, if a simulator or a robot dynamics model is erroneous, the control policy trained on this simulator may not work on the real system. One possible route towards bridging the sim2real gap is domain adaption, wherein a data-driven control policy is first trained in simulation and then re-trained on the real system. While domain adaptation might be viable, it significantly benefits or, may not be necessary if the robot's dynamics model is an accurate representation of the actual dynamics.

Information value It is not only the amount of data that determines the approximation result of a data-driven model but also the information that is present in the data. It does not suffice to collect hours of data from a resting robot, but instead, the system must be *excited i.e.*, moved on carefully selected trajectories. The design of excitation signals for collecting informative data is in itself an ongoing field of research (Buisson-Fenet et al., 2020).

Control Dilemma Steering a robot through its state space to collect data requires a controller. This potentially spawns a chicken or egg causality dilemma, as the motive for collecting data in the first place is to learn a dynamics model that enables the synthesis of a controller. Fortunately, on most robotic systems one has access to a linear controller that

enables the collection of data in a small part of the state space. The interpretation of a state space being "small" depends on the sample complexity of the data-driven model.

It is possible to tackle the control dilemma by utilizing approaches such as model-based RL. These methods alternate between using a control policy to collect data which is used to improve the dynamics model, and using the dynamics model to improve the control policy. Using a data-driven model for describing the dynamics as well as the controller may require substantial amounts of data.

For robotic systems in which data collection is not an issue, one may still face the following practical concerns that advocate against dynamics identification with solely data-driven methods.

Resource consumption If a robot's dynamics are considerably nonlinear, the data-driven model must be able to approximate the target function closely. In turn, a parametric data-driven model, *e.g.*, a NN, will have many parameters. Training such a NN for several days, often using multiple GPUs, consumes significant amounts of time, financial resources, and energy.

Adjustability of black-box models If a robot interacts with its environment, its dynamics may be subject to change. For example, a change in the dynamics occurs if a robot arm picks up an object, a wheeled robot drives over different surfaces, or the joints of a quadrupedal robot are subject to wear and tear. In a physics model, one could identify and subsequently adjust the respective latent function, while in a data-driven model, it is unclear how to incorporate the change in the dynamics without fully re-training the model.

Hardware requirements Making predictions with a data-driven model may require more computational resources such as processing power and memory of a micro-controller than using a physics-informed regression model.

In summary, robot dynamics often consists of involved nonlinear functions, while data collection on robotic systems is cumbersome, and the amount of training data is limited.

> *Therefore, the main incentive for combining a data-driven model with prior physics knowledge is to improve its sample efficiency, whereas the prime motive for augmenting a physics model with data-driven learning is to increase its prediction accuracy.*

Improving sample efficiency or increasing prediction accuracy are two sides of the same coin that evolves around mitigating the shortcomings of both modeling frameworks.

1.2 Current State of Research

Countless works exist on obtaining robot dynamics models. In what follows, a brief overview of selected literature related to physics-informed regression of robot dynamics is given. The presented works on physics-informed regression will be revisited in Section 4 after a unified view of the errors in rigid body dynamics has been developed. This section is taken from *(Geist and Trimpe, 2021)* and *(Rath et al., 2021)*.

Literature on data-driven models

Common data-driven models used for the identification of dynamics are (Bayesian) linear regression, NNs, and GPs. Most works that learn dynamics functions using data-driven models solely use transition dynamics or inverse transition dynamics. Nguyen-Tuong and Peters (2011) surveys these approaches for robot control. NN have been used for the direct identification of fluid flows (Morton et al., 2018), ODEs (via recurrent NNs) (Funahashi and Nakamura, 1993), and actuated dynamics (Kuschewski et al., 1993). NNs were able to learn transition dynamics and control policies on contact-rich domains (Chua et al., 2018a; Nagabandi et al., 2020). Remarkably, Nagabandi et al. (2020) demonstrated that NNs enable the inference of specific control policies for handling objects inside a robotic hand. However, in Nagabandi et al. (2020) the training of a control policy for a specific task-object combination requires several hours of data.

Deisenroth and Rasmussen (2011) learned the system's transition dynamics with a GP. This GP dynamics model was then used for the nonlinear control of real mechanical systems with to NNs comparably small amounts of data. One key takeaway of Deisenroth and Rasmussen (2011) 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 significantly improved the robustness of a control policy trained on the GP dynamics model. Other applications of dynamics modeling with GPs for control include (Doerr et al., 2017, 2018; Eleftheriadis et al., 2017; Frigola et al., 2013; Kocijan et al., 2005; Mattos et al., 2016; Nguyen-Tuong and Peters, 2010). 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 (Quiñonero-Candela and Rasmussen, 2005), the computational effort required to work with such models is considerably larger than using NNs.

Literature on rigid body dynamics models

Pioneering works on robot dynamics identification focused on the estimation of analytical parameters. For example, the well-known approach of Atkeson et al. (1986) uses the linearity of a rigid robot arm's ID with respect to its parameters to estimate these via linear regression. With the increased popularity of libraries for automatic differentiation, recent works (Ledezma and Haddadin, 2017, 2018; Lutter et al., 2020; Sutanto et al., 2020) 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. We further discuss the problems of parameter estimation in analytical dynamics models in Section 4.2.2.

Literature on Physics-informed Models

In what follows, we discuss recent works that combine data-driven models with rigid body dynamics.

Building rigid body dynamics into neural networks As an alternative approach to analytical modeling, Cranmer et al. (2020); Gupta et al. (2020); Lutter et al. (2019a,b)

model dynamics via the Euler-Lagrange equations in which a NN approximates the system's Lagrangian or the entries of the inertia matrix. In turn, such a Lagrangian NN increases the flexibility of the functional approximation of mass-related dynamic terms compared to analytical dynamics models while still respecting energy conservation. If forces are the major sources of analytical errors and mass-related quantities are not, then Lagrangian NNs may 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 in structured modeling combines analytical models of mass-related quantities and presumably known forces with data-driven modeling. Lutter et al. (2020) augments an analytical FD model with a NN approximating a robot arm's joint torques. The parameters of the analytical model and NN are estimated jointly via automatic differentiation. Recently, Hwangbo et al. (2019) and subsequently Lee et al. (2020) combined an analytical simulation with a pre-trained NN – predicting measured joint torques. Notably, the contact forces were modeled analytically as detailed by Hwangbo et al. (2018) while their physical parameters are varied to robustify the trained control policy towards parameter uncertainties. Significantly, NN control policies that were trained on such structured models achieved so far unseen robustness of a quadruped's locomotion policy. These seminal works indicate that the sim2real gap can be closed by consciously combining analytical mechanics with data-driven modeling.

Building rigid body dynamics into Gaussian Processes In real robotic systems, noise and uncertainty are often significantly impacting the robot's dynamics. For example, a friction force denotes a macroscopic abstraction of microscopic tribologic phenomena, which aggravates deterministic modeling. 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 introduce uncertainty into the dynamics. As shown in Chua et al. (2018b); Deisenroth and Rasmussen (2011), the synthesis of a robot's control policy can significantly benefit from the availability of an uncertainty measure for the planned motion. Popular models for the identification of uncertain and noisy dynamics are either building on Bayesian linear regression as *e.g.*, (Ting et al., 2006), or often resort to GPs (Doerr et al., 2017, 2018; Eleftheriadis et al., 2017; Frigola et al., 2013; Kocijan et al., 2005; Mattos et al., 2016). GPs incorporate various model assumptions through the covariance function (kernel) and are often more data-efficient than NNs. Yet, vanilla GP regression requires the computation of the inverse covariance matrix, 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 prone to numerical problems, and deriving an efficient computational implementation of multi-output GPs forms a considerable obstacle. Despite these challenges, several models have been proposed that combine GP regression with analytical dynamics.

Nguyen-Tuong and Peters (2010) used Atkeson et al. (1986) as a linear parametric mean of a GP model (cf. Chapter 2.7 in Williams and Rasmussen (2006)). Conceptually being similar to Nguyen-Tuong and Peters (2010), Saveriano et al. (2017) approximate the residuals of an analytical transition dynamics model via one-dimensional GPs.

Cheng et al. (2015) 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 (Solak et al., 2003). Jidling et al. (2017) emphasized that more generally GPs are closed under linear functionals. In turn, the authors proposed a linearly transformed GP such that its predictions fulfill a linear operator equation *e.g.*, the mechanical stress field inside a linear elastic material.

1.3 Contribution

This dissertation is based on the following publications:

1. A. René 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 (L4DC)*, volume 120 of *Proceedings of Machine Learning Research (PMLR)*, pages 225–234, June 2020

2. A. René Geist and Sebastian Trimpe. Structured learning of rigid-body dynamics: A survey and unified view from a robotics perspective. *GAMM Mitteilungen*, 44(2, Special Issue: Scientific Machine Learning):34, June 2021

3. Lucas Rath*, A. René Geist*, and Sebastian Trimpe. Using physics knowledge for learning rigid body forward dynamics with Gaussian process force priors. In *Proceedings of the 5th Conference on Robot Learning (CORL)*, volume 164 of *Proceedings of Machine Learning Research (PMLR)*, pages 101–111, 08–11 Nov 2021

4. A. René Geist, Jonathan Fiene, Naomi Tashiro, Zheng Jia, and Sebastian Trimpe. The wheelbot: A jumping reaction wheel unicycle. *IEEE Robotics and Automation Letters*, 7(4):9683–9690, 2022

This dissertation uses text from the above publications. If a figure from the above publications is used in this dissertation, we do not explicitly cite the publication in the figure's description. The copyright to this text passages and figures if not belonging to the authors remains with the respective publishers.

In what follows, we give a brief overview of the individual sections of this dissertation and point out its main contributions.

Chapter 2 This chapter is based on *Geist and Trimpe (2021)* and *Rath et al. (2021)*. In Section 2.1, we introduce supervised regression as the general problem setting. As recent works on physics-informed regression substantially benefited from modern libraries on automatic differentiation, we briefly introduce this particular method for the computation of analytical gradients at specific data points. Section 2.2 provides intuition on the solution of linear matrix equations, which is an important concept to grasp how constraint equations provide structural knowledge on the directions of forces. We briefly discuss Lagrangian optimization in Section 2.3 as throughout this work, we encounter the concept of a Lagrangian function whose partial derivatives, if carefully arranged, yield an alternative

* Authors contributed equally.

perspective on implicitly constrained dynamics and subsequently Lagrangian NNs. As this work also details the combination of GP regression with implicitly constrained rigid body dynamics, an extensive treatment of multi-output GP regression is given.

Chapter 3 This chapter is based on *Geist and Trimpe (2021)*. Through a careful bisection of some of the most common EOM in robotics, such as the Newton-Euler EOM and Euler-Lagrange EOM, we arrive at a broader picture of how analytical mechanics provides structural knowledge. An important part of this work is the discussion of implicitly constrained rigid body dynamics in the Udwadia-Kalaba formulation. Further, we visualize the structure that underlies constraint-related transformations by resorting to diagrams illustrating the fundamental subspaces of a matrix as found in the literature on functional analysis. Moreover, we show that both the Udwadia-Kalaba formulation of implicitly-constrained EOMs as well as its formulation in terms of Lagrangian optimization can be derived with Gauss' principle as a starting point. In summary, Chapter 3 provides a comprehensive overview of structural knowledge in rigid body dynamics that proved utile for physics-informed regression of robot dynamics.

Chapter 4 This chapter is based on *Geist and Trimpe (2021)* and *Rath et al. (2021)*. We investigate how functions underlying rigid body dynamics have different causes, while our knowledge of each of these functions may considerably differ. Borrowing concepts from statistical learning theory, we then arrive at a concise description of potential errors in an analytical dynamics model that distinguishes estimation errors from approximation errors. This discussion enables us to compare the pros and cons of using ID and FD models in physics-informed regression. Here, we emphasize that errors in physics-informed models are 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). Afterward, using the developed unified view on modeling errors in rigid body dynamics, selected works on physics-informed regression are categorized into analytical dynamics models, analytical (output) residual modeling, and analytical latent modeling.

Chapter 5 This chapter is based on *Geist and Trimpe (2020)* and *Rath et al. (2021)*. We investigate the combination of GP regression with implicitly-constrained rigid body dynamics. Through a series of simulation experiments, we show that placing a GP prior on the unknown forces is superior to placing a GP prior on the unknown unconstrained accelerations. Moreover, we illustrate how placing a GP on unknown latent functions inside a rigid body dynamics model enables the identification of numerous other latent quantities such as the system's implicit constraint force using measurements of the constrained acceleration. Moreover, we show that using Baumgarte stabilization in the proposed probabilistic model enables long-term trajectory predictions that respect implicit constraint knowledge.

Chapter 6 This chapter is based on *Geist et al. (2022)*. The research of the previous chapters was carried out in parallel with the development of a novel robotic testbed for learning control named "Wheelbot". In turn, the theoretical discussion of physics-informed

modeling benefited from and contributed to, the development of a small robot testbed that possesses interesting properties that are commonly encountered in robotics. Such properties include being subject to holonomic and non-holonomic constraints, being under-actuated, and having naturally unstable dynamics that aggravate data collection. In addition, the proposed robot uses brushless motors as these are commonly found on other robotic systems as well as occupy only a small operating space to enable research in a confined laboratory environment. This chapter details the mechanical, electronics, and firmware design.

Chapter 2

Mathematical Preliminaries

This section provides a comprehensive overview of mathematical preliminaries that are helpful throughout this work. Section 2.1 is based on a discussion of gradient-based optimization in physics-informed ML as found in *Geist and Trimpe (2021)*. The introduction to multi-output GPs is based on the supplementary material of our work on combining GPs with implicitly constrained rigid body dynamics *(Rath et al., 2021)*.

2.1 Supervised Regression: Model, Loss, and Optimizer

Machine learning methods can be categorized into supervised, unsupervised, and reinforcement learning. Supervised learning assumes that the target function's inputs and some of its outputs can be observed. Unsupervised learning assumes that no output values are given and instead, the model must infer structure from the input data. In reinforcement learning, the desired outputs cannot be observed, but instead, some form of information on the model's performance is given.

Supervised learning can be divided into classification and regression. In classification, the model predicts a probability that the input belongs to a certain discrete class, whereas in regression, the model maps the input to a continuous quantity, which in our case is the real number line $\mathbb{R}$.

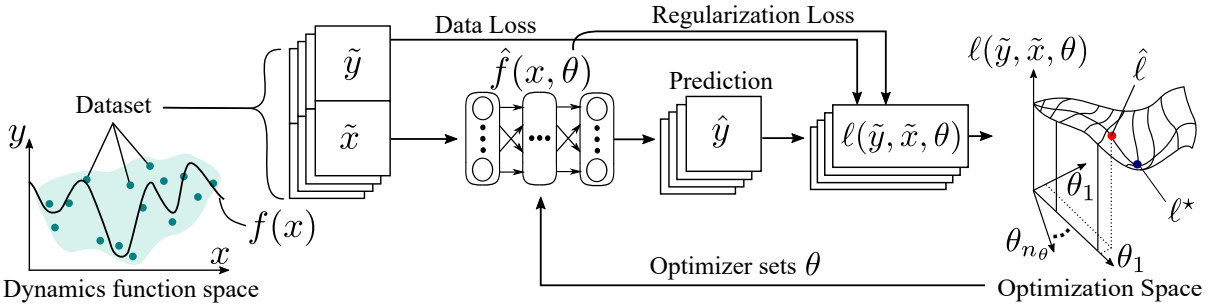


Figure 2.1: Illustration of supervised regression using a parametric model. Adapted from Geist et al. (2017).

In this work, data-driven modeling solely refers to the supervised regression of a dynamics function using data collected from the robotic system. The training data set

$$\mathcal{D} := \{X, Y\} \tag{2.1}$$

consists of a set of N input vectors $X := \{\tilde{x}_i\}_{i=1}^N$ and output vectors $Y := \{\tilde{y}_i\}_{i=1}^N$ with $\tilde{x}_i \in \mathbb{R}^{n_x}$ and $\tilde{y}_i \in \mathbb{R}^{n_y}$ being noisy observations of the i-th input x or output y of the dynamics (1.1). We assume that all states of the dynamics can be observed.

Before delving into non-parametric regression in Section 2.4, we first assume that the robot's dynamics shall be approximated by a parametric model taking the form

$$\hat{y} = \hat{f}(x, \theta), \tag{2.2}$$

with $\theta \in \mathbb{R}^{n_\theta}$ being a vector of model parameters and the model's prediction $\hat{y} \in \mathbb{R}^{n_y}$. Both, analytical dynamics models, as well as neural networks, are parametric models. Typically, the prediction accuracy of a parametric model is measured with respect to an objective function which we simply refer to as *loss*. In machine learning, a commonly encountered loss function is the quadratic loss with quadratic parameter regularization

$$\ell(\mathcal{D}, \theta) = \left(\sum_{i=1}^{N} \alpha_i \|(\tilde{y}_i - f(\tilde{x}_i, \theta))\|^p \right)^{1/p} + \alpha_0 \|\theta\|^2, \tag{2.3}$$

where besides $p = 2$ being the vector-2 norm, other common choices are $p = 1$ being the sum of absolute values, and $p = \infty$ being the maximum value norm. The $\alpha_1, ..., \alpha_N$ are hyper-parameters that weigh the importance of each data point while α_0 weighs the parameter regularization term (Nelles, 2013, p. 28). The parameter regularization term $\|\theta\|^2$ adds an incentive for the parameter values to be zero if they do not affect the loss function. When a model is being *trained*, an optimization algorithm determines the model's parameters such that ideally the loss is minimized.

A supervised regression algorithm consists of a model, loss, and optimizer in some form or another. For example, in linear regression as detailed in Section 2.2.1, the model is a linear equation, the loss is the norm $\|Ax - b\|^2$, and the optimization technique forms the analytical computation of the solution via the Moore-Penrose inverse. For considerably nonlinear models such as NNs, it is rarely possible to find their parameters analytically. Finding a set of parameters that reduces the loss to an acceptable value is often a difficult problem due to the dimensionality of the parameter space. Analytical dynamics models quickly have more than 100 parameters, while neural networks easily have several thousand parameters. In turn, the optimization of a parametric model suffers under the curse of dimensionality. In addition, physics-informed dynamics 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 phenomena. Therefore, most works utilizing physics-informed models require the training of high-dimensional models on large datasets. In turn, the models also have a considerable amount of parameters such that recent works on physics-informed learning predominantly resort to gradient-based optimization instead of using global optimization techniques.

2.1.1 Gradient-based Optimization

Gradient-based optimization forms a cornerstone of solely data-driven modeling as well as constitutes one of the most common nonlinear local optimization techniques (Bottou et al., 2018). Gradient-based optimization techniques require the computation of the partial derivative of the loss with respect to the parameters, writing

$$g = \nabla_\theta \ell = \frac{\partial \ell}{\partial \theta} = [\frac{\partial \ell}{\partial \theta_1}, \frac{\partial \ell}{\partial \theta_2}, ..., \frac{\partial \ell}{\partial \theta_{n_\theta}}]^\mathsf{T}. \tag{2.4}$$

A gradient-based optimizer takes the general form

$$\theta_k = \theta_{k-1} - \eta_{k-1} p_{k-1} \quad \text{with} \quad p_{k-1} = R_{k-1} g_{k-1}, \tag{2.5}$$

with θ_k being the vector of model parameters at optimization iteration k, p_{k-1} is a direction vector, η_{k-1} is the step size, and R_{k-1} is a gradient transformation matrix (Nelles, 2013, p. 90). Choosing $R_{k-1} = I_{n_\theta}$ performs a parameter update in the opposite direction of the gradient or in other words the direction of steepest *descent* in the loss. Two commonly used gradient-based optimizers are gradient descent with momentum and ADAM (Kingma and Ba, 2014a). An astounding exposition of momentum is given in Goh (2017). Often it is useful to include higher-order derivatives of the objective function in the parameter update rule. As gradient-based optimization is a local optimization technique and ℓ is usually nonlinear, it is indispensable to restart the optimization several times with random initialization of θ, keeping only the best optimization results. 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.

As detailed by Baydin et al. (2017), 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 becomes cryptic due to an "expression swell" as pointed out by Corliss (1988).
- *Automatic differentiation*, which computes accurate gradients at given data points, is considerably faster than the aforementioned methods and thanks to steady improvements in the usability of optimization libraries, is straightforward to use.

2.1.2 Automatic Differentiation

At the core of automatic differentiation (AD), 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 (Baydin et al., 2017).

The two basic forms of AD are forward-mode AD and reverse-mode AD. In what follows, we assume a loss $\ell(\theta) = \ell(c(b(a(\theta))))$ where $a(\theta)$, $b(a)$, and $c(b)$ are functions of appropriate size. In *forward-mode AD*, the gradient of a function is obtained via forward accumulation, *e.g.*, writing

$$\nabla_\theta \ell(\theta) = \nabla_c \ell(\nabla_b c(\nabla_a b \nabla_\theta a)), \tag{2.6}$$

where $\nabla_\theta b = \nabla_a b \nabla_\theta a$ denotes a Jacobian matrix.

In *reverse-mode AD* being also referred to as "backprop" in machine learning literature, computes gradients via reverse accumulation, *e.g.*, writing

$$\nabla_\theta \ell(\theta) = ((\nabla_c \ell \nabla_b c) \nabla_a b) \nabla_\theta a. \tag{2.7}$$

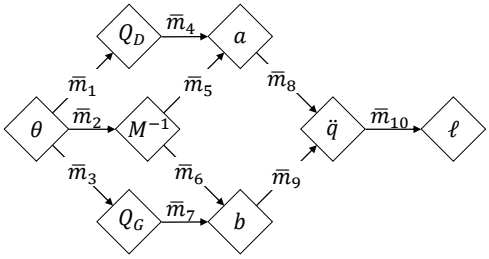


Figure 2.2: Graph illustrating backward mode automatic differentiation of the loss of a forward dynamics model. Figure adapted from *Geist and Trimpe (2021)*.

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 fewer operation counts for the computation of vector-Jacobian products (Baydin et al., 2017). 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. The following example illustrates the reverse-mode AD of a rigid body dynamics model.

Example 2.1 (Computing the gradient of a structured model with reverse-mode AD)**.** *Assume that the dynamics of a robotic system are modeled via FD as*

$$\hat{\ddot{q}}(x;\theta) = \underbrace{M^{-1}(x;\theta)\hat{Q}_D(x;\theta)}_{a} + \underbrace{M^{-1}(x;\theta)\hat{Q}_G(x;\theta)}_{b}, \tag{2.8}$$

in which $\hat{Q}_G(x;\theta)$ and $\hat{Q}_D(x;\theta)$ denote either analytical and/or data-driven models, and intermediate function expressions are abbreviated as $a = M^{-1}\hat{Q}_D$, $b = M^{-1}\hat{Q}_G$. Note that Figure 2.2 depicts the computational graph formed by (2.8)*. The parameters θ are estimated jointly using an objective function $\ell(\mathcal{D},\theta) := \ell(\theta)$ where the dependency on the data $\mathcal{D}$ is being omitted in the notation. In reverse mode AD, the gradient of ℓ with respect to θ is decomposed as*

$$\nabla_\theta \ell(\theta) = \underbrace{\nabla_{\hat{Q}_D}\ell \nabla_\theta \hat{Q}_D}_{\bar{m}_1} + \underbrace{\nabla_{M^{-1}}\ell \nabla_\theta M^{-1}}_{\bar{m}_2} + \underbrace{\nabla_{\hat{Q}_G}\ell \nabla_\theta \hat{Q}_G}_{\bar{m}_3}, \tag{2.9}$$

with

$$\bar{m}_1 = \underbrace{\nabla_a \ell \nabla_{\hat{Q}_D} a}_{\bar{m}_4} \nabla_\theta \hat{Q}_D, \quad \bar{m}_2 = \big(\underbrace{\nabla_a \ell \nabla_{M^{-1}} a}_{\bar{m}_5} + \underbrace{\nabla_b \ell \nabla_{M^{-1}} b}_{\bar{m}_6}\big) \nabla_\theta M^{-1}, \quad \bar{m}_3 = \underbrace{\nabla_b \ell \nabla_{\hat{Q}_G} b}_{\bar{m}_7} \nabla_\theta \hat{Q}_G, \tag{2.10}$$

$$\bar{m}_4 = \underbrace{\nabla_{\hat{\ddot{q}}}\ell \nabla_a \hat{\ddot{q}}}_{\bar{m}_8} \nabla_{\hat{Q}_D} a, \quad \bar{m}_5 = \underbrace{\nabla_{\hat{\ddot{q}}}\ell \nabla_a \hat{\ddot{q}}}_{\bar{m}_8} \nabla_{M^{-1}} a, \quad \bar{m}_6 = \underbrace{\nabla_{\hat{\ddot{q}}}\ell \nabla_b \hat{\ddot{q}}}_{\bar{m}_9} \nabla_{M^{-1}} b, \quad \bar{m}_7 = \underbrace{\nabla_{\hat{\ddot{q}}}\ell \nabla_b \hat{\ddot{q}}}_{\bar{m}_9} \nabla_{\hat{Q}_G} b. \tag{2.11}$$

Therefore, the gradient of a physics-informed 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 $\nabla_\theta \hat{Q}_D$. Ideally, an AD library will compute these terms for us by also decomposing the gradients into product sums of basic gradient functions.

Example 2.1 illustrates that the computation of $\nabla_\theta \ell(\theta)$ requires the partial differentiation of analytical latent functions. Yet, to do this efficiently, several requirements must be fulfilled by an AD library, namely:

- Physics-informed models denote forward functions that consist of numerous basic mathematical operators. Ideally, the AD library should only require that the physics-informed model is expressed in terms of a computer library for basic linear algebraic operations such as Numpy (Harris et al., 2020) or Torch (Paszke et al., 2019). In turn, the model designer must only convert the derived analytical functions into the programming language of the respective AD library.
- As detailed in Section 4.4.1, models for $\hat{Q}_\mathrm{C}$, $\hat{Q}_\mathrm{G}$, 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" (Maclaurin et al., 2015) and Torch's autograd library 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 breaking the AD routines. These AD packages have been further improved in libraries such as JAX (Bradbury et al., 2018) and PyTorch (Paszke et al., 2019), which additionally combine AD with compilers for GPU acceleration such as XLA[1]. These packages tremendously simplify the synthesis of a physics-informed model, enable easy debugging of the respective computer code, and provide computationally efficient implementations with GPU acceleration.

2.2 Linear Matrix Equations

At several points in this work, the solution to a linear equation is determined and provides insight into the structure underlying a physical equation. For example, in Section 3.2, the Newton-Euler equations of a rigid body system are transformed into the generalized coordinate space through a linear transformation. In Section 3.3, we summarize the Euler-Lagrange equations, which are obtained after the transformation of a Lagrangian function with a matrix of partial differential operators. In Section 3.4.3, implicit constraint equations of a rigid body system are linear with respect to the system's acceleration which allows to obtain expression of the implicit constraint forces. This latter point, we will use in Chapter 5 to linearly transform a GP prior that approximates the unknown forces inside a rigid body dynamics model. The important role of linear transformations in rigid body mechanics enables us to further discuss in Section 4.1 how errors in rigid body dynamics can be approximated through data-driven models. In Section 6.3.2, we detail a state estimator that uses the linearity of the system's kinematics with respect to the gravitational acceleration vector.

The following sections evolving around linear matrix equations are entirely based on Beard (2002). Before discussing the mathematical structure underlying linear equations, we briefly define some basic terminology. There are two important vector spaces for the description of analytical dynamics, namely

[1] `https://www.tensorflow.org/xla`

- $\mathbb{R}^n$: The set of n-tuples of real numbers.
- L^n : The set of measurable functions $f : \mathbb{R}^n \to \mathbb{R}$, which includes piecewise continuous functions and the set of all physically realizable signals.

While analytical dynamics could be discussed in terms of matrix-valued functions whose entries are elements of L^n, we discuss such matrix-valued functions after inserting a specific point in state space. In turn, the matrix-valued functions underlying analytical mechanics become real matrices at specific points in the state space. In Beard (2002), a more general discussion in terms of Hilbert spaces is given where a linear operator may also denote a convolution or differentiation.

To be able to compare elements of a vector space, an inner product is defined yielding an inner product space. An inner product that is often used in $\mathbb{R}^n$ with $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ is the *dot product*

$$\langle x, y \rangle = y^\mathsf{T} x = \sum_{i=1}^{n} x_i y_i, \tag{2.12}$$

or alternatively, the weighted inner product

$$\langle x, y \rangle_W = y^\mathsf{T} W x = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} x_j y_i, \tag{2.13}$$

where $W \in \mathbb{R}^n$ is a positive definite matrix denoted as $W > 0$. The following definitions are excerpts from (Beard, 2002), in which the definitions in terms of the more general case of linear transformations between Hilbert spaces have been adjusted to the specific case of matrix transforms between $\mathbb{R}^n$ and $\mathbb{R}^m$.

Definition 2.2 (Orthogonal subspace)**.** *Let $V, W \subset \mathbb{R}^n$, then V is orthogonal to W, written $V \perp W$, if for all $v \in V$ and $w \in W$, $\langle v, w \rangle = 0$.*

Definition 2.3 (Orthogonal complement)**.** *Let $V \subset \mathbb{R}^n$, then the orthogonal complement of V is the set $V^\perp = \{x \in \mathbb{R}^n : \forall v \in V,\ \langle x, v \rangle = 0\}$.*

Definition 2.4 (Orthogonal sum)**.** *If V and W are orthogonal subspaces of $\mathbb{R}^n$, then their orthogonal sum is $V \oplus W = \{x \in \mathbb{R}^n : x = v + w, v \in V, w \in W\}$.*

From the definitions above it follows that if V is a subspace of $\mathbb{R}^n$, then $\mathbb{R}^n = V \oplus V^\perp$. In other words, a vector space $\mathbb{R}^n$ may be split up in two sets of vectors whose elements are orthogonal to each other.

A real-valued matrix $A \in \mathbb{R}^{m \times n}$ is a linear operator as with $x_1, x_2 \in \mathbb{R}^n$ and $\alpha, \beta \in \mathbb{R}$ it holds that

$$A(\alpha x_1 + \beta x_2) = \alpha A x_1 + \beta A x_2. \tag{2.14}$$

Definition 2.5 (Adjoint)**.** *Given $A \in \mathbb{R}^{m \times n}$, a matrix $A^* \in \mathbb{R}^{m \times n}$ is called an adjoint, if*

$$\langle Ax, y \rangle = \langle x, A^* y \rangle \tag{2.15}$$

for any $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$.

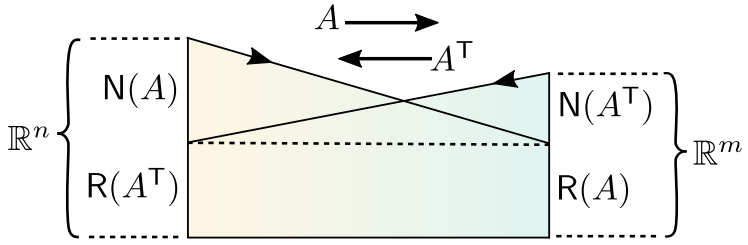


Figure 2.3: Fundamental subspaces of a matrix A. Adapted from Beard (2002).

From Definition 2.5 it directly follows with (2.12) that the adjoint of A is its transposed matrix A^T, written $A^* = A^\mathsf{T}$. The null space and range of a matrix and its adjoint are referred to as the *fundamental subspaces* and form an important form of structural knowledge that we will encounter throughout in this work.

Definition 2.6 (Null space)**.** *Given $A \in \mathbb{R}^{m\times n}$, the null space of A is defined as*

$$\mathsf{N}(A) = \{x \in \mathbb{R}^n : \, Ax = 0\}. \tag{2.16}$$

Definition 2.7 (Range space)**.** *Given $A \in \mathbb{R}^{m\times n}$, the range space of A is defined as*

$$\mathsf{R}(A) = \{b \in \mathbb{R}^m : \, \exists x \in \mathbb{R}^n \text{ such that } b = Ax\}. \tag{2.17}$$

As shown in more detail in Beard (2002), it holds that $[\mathsf{R}(A)]^\perp = \mathsf{N}(A^*)$, $[\mathsf{R}(A^*)]^\perp = \mathsf{N}(A)$, and $\mathbb{R}^n = V \oplus V^\perp$, such that the theorems below follow.

Theorem 2.8 (Fundamental subspaces)**.** *Given $A \in \mathbb{R}^{m\times n}$:*

$$\begin{aligned}
&1)\ \mathbb{R}^n = \mathsf{R}(A^*) \oplus \mathsf{N}(A).\\
&2)\ \mathbb{R}^m = \mathsf{R}(A) \oplus \mathsf{N}(A^*).\\
&3)\ n = \dim(\mathsf{R}(A^*)) + \dim(\mathsf{N}(A)).\\
&4)\ m = \dim(\mathsf{R}(A)) + \dim(\mathsf{N}(A^*)).
\end{aligned} \tag{2.18}$$

Theorem 2.9 (Range space dimensions)**.** *Given $A \in \mathbb{R}^{m\times n}$:*

$$\dim(\mathsf{R}(A^*)) = \dim(\mathsf{R}(A)). \tag{2.19}$$

In the above expression, we denoted the dimension of a vector space as $\dim(\mathbb{R}^n) = n$. With Theorem 2.8 and Theorem 2.9, the fundamental subspaces of a linear equation of the form

$$Ax = b, \tag{2.20}$$

with $A : \mathbb{R}^n \to \mathbb{R}^m$, $x \in \mathbb{R}^n$ and $\mathbb{R}^m$ can be illustrated in a subspace diagram as depicted in Figure 2.3.

To shed further light on the fundamental subspaces underlying linear matrix transformations, the following sections detail two special cases, the *least-squares* linear equation, and the *minimum-norm* linear equation. Afterward, we briefly discuss how the singular value decomposition (SVD) of a A gives rise to the general solution of (2.20).

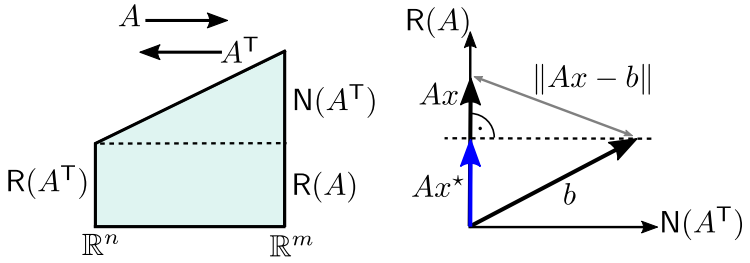


Figure 2.4: Subspaces of a least-squares linear equation. Adapted from Beard (2002).

2.2.1 Least-squares Linear Equation

The least-squares linear equation takes the form

$$Ax = b, \text{ with } A \in \mathbb{R}^{m\times n}, \text{ rank}(A) = n \text{ and } m > n. \tag{2.21}$$

In turn, the fundamental subspaces of A are depicted in Figure 2.4 (Right), where $\mathsf{N}(A)$ simply reduces to the null vector, while the m-vector b potentially consists of a null space part $b_\mathsf{N} \in \mathsf{N}(A^\mathsf{T})$ and a range space part $b_\mathsf{R} \in \mathsf{R}(A)$ such that $b = b_\mathsf{R} + b_\mathsf{N}$.

If $b \in \mathsf{R}(A)$, equation (2.21) can be solved directly. Left-multiplication of (2.21) by A^T yields $A^\mathsf{T}b = A^\mathsf{T}Ax$. As the square matrix $A^\mathsf{T}A$ has full rank, computing its inverse gives rise to the solution for x as

$$x^\star = (A^\mathsf{T}A)^{-1}A^\mathsf{T}b = A^+b, \tag{2.22}$$

where $A^+ = (A^\mathsf{T}A)^{-1}A^\mathsf{T}$ denotes the Moore-Penrose (MP) inverse of A for $\text{rank}(A) = n$. If $b \notin \mathsf{R}(A)$, no direct solution to (2.22) exists. Instead, the problem setting can be changed to find the unique solution $x^\star$ that satisfies (2.21) while minimizing

$$x^\star = \arg\min \|Ax-b\|^2 = \arg\min (Ax-b)^\mathsf{T}(Ax-b). \tag{2.23}$$

As any non-zero vector Ax must lie in $\mathsf{R}(A)$, the x minimizing $\|Ax-b\|^2$ is obtained when $b_\mathsf{R} = Ax^\star$. In turn, the optimal $x^\star$ is given by (2.22). Figure 2.4 (Left) visualizes the least squares optimization problem in terms of the fundamental subspaces of A, while Figure 2.4 (Right) sketches the corresponding vector diagram in $\mathbb{R}^m$.

Weighted least-squares linear equation

The solution of the weighted least-squares optimization problem

$$\tilde{x}^\star = \arg\min \|Ax-b\|_W^2 = \arg\min (Ax-b)^\mathsf{T}W(Ax-b), \tag{2.24}$$

with $W > 0$, is found by exchanging terms with $\tilde{b} = W^{1/2}b$ and $\tilde{A} = W^{1/2}A$ to obtain the standard-least squares problem (Beard, 2002, p. 78), (Udwadia and Kalaba, 2007, p.59). In turn, equation (2.25) reduces to

$$\tilde{x}^\star = \text{argmin} \|\tilde{A}x - \tilde{b}\|^2. \tag{2.25}$$

By comparison with (2.22), the solution to the weighted least-squares problem is obtained via (2.25) as

$$\tilde{x}^\star = (AWA)^{-1}AWb. \tag{2.26}$$

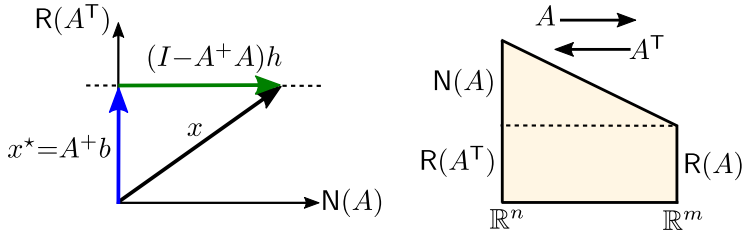


Figure 2.5: Subspaces of a minimum-norm linear equation. Adapted from Beard (2002).

2.2.2 Minimum-norm Linear Equation

The minimum-norm linear equation takes the form

$$Ax = b, \text{ with } A \in \mathbb{R}^{m\times n}, \text{ rank}(A) = m \text{ and } m < n. \tag{2.27}$$

Therefore, the fundamental subspaces corresponding to (2.27) are depicted by Figure 2.5 (Right). As $m < n$ and $\text{rank}(A) = m$ the null space $\mathsf{N}(A)$ is non-trivial such that x splits up into a range space part and null space part, writing $x = x_\mathsf{R} + x_\mathsf{N}$, while $b \in \mathsf{R}(A^\mathsf{T})$. Due to the existence of $x_\mathsf{N} \in \mathsf{N}(A)$, (2.27) has an infinite number of solutions x as illustrated in Figure 2.5 (Left).

To find a unique solution of (2.27), one may wish to find

$$\begin{aligned} &\text{minimize } \|x\|^2 \\ &\text{subject to } Ax = b. \end{aligned} \tag{2.28}$$

The optimal solution of (2.28) is found by first noting that there must exist a $\zeta \in \mathbb{R}^m$ such that $x^\star = A^\mathsf{T}\zeta$. With (2.27), one obtains $AA^\mathsf{T}\zeta = b$ such that the solution to (2.28) becomes

$$x^\star = A^\mathsf{T}(AA^\mathsf{T})^{-1}b = A^+b, \tag{2.29}$$

where $A^+ = A^\mathsf{T}(AA^\mathsf{T})^{-1}$ denotes the MP inverse of A for $\text{rank}(A) = m$. In other words, (2.29) provides the vector that is shortest with respect to $\|x\|^2$ that if transformed by A yields b. If b is replaced with Ax, one obtains the shortest part of the vector x with respect to $\|x\|^2$ that is in $\mathsf{R}(A^\mathsf{T})$ as

$$x_\mathsf{R} = A^\mathsf{T}(AA^\mathsf{T})^{-1}Ax = A^+Ax. \tag{2.30}$$

In turn, the shortest part of the vector x with respect to $\|x\|^2$ that is in $\mathsf{N}(A)$ is obtained as

$$x_\mathsf{N} = x - x_\mathsf{R} = (I_n - A^+A)x. \tag{2.31}$$

The matrices (A^+A) and $(I - A^+A)$ are orthogonal projections as they are idempotent and symmetric. With (2.31), the general solution to (2.28) is given by

$$x = A^+b + (I_n - A^+A)h, \tag{2.32}$$

where h denotes an arbitrary n-vector.

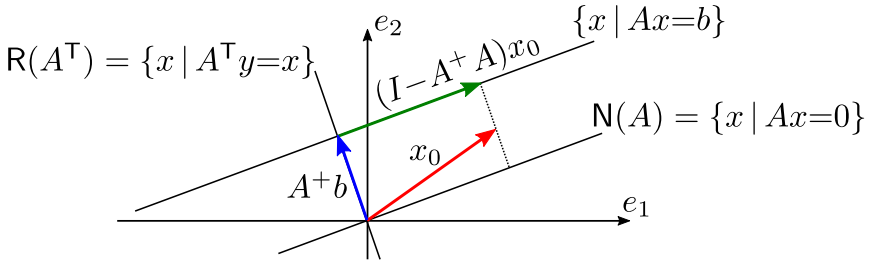


Figure 2.6: Illustration of the solution to a minimum-norm linear equation in $\mathbb{R}^2$ where the distance $\|x - x_0\|^2$ is minimized. Adapted from Boyd et al. (2004).

Weighted minimum-norm linear equation

The weighted minimum-norm optimization problem reads

$$\begin{aligned} &\text{minimize } \|x - x_0\|_W^2, \\ &\text{subject to } Ax = b, \end{aligned} \tag{2.33}$$

with $W > 0$ and the bias vector $x_0 \in \mathbb{R}^n$. Similar to Section 2.2.1, the problem in (2.33) can be simplified to (2.28) by the change of variables $\tilde{x} = W^{1/2}(x - x_0)$, $\tilde{A} = AW^{-1/2}$, and $\tilde{b} = b - Ax_0$ one obtains

$$\begin{aligned} &\text{minimize } \|\tilde{x}\|^2, \\ &\text{subject to } \tilde{A}\tilde{x} = \tilde{b}. \end{aligned} \tag{2.34}$$

Subsequently, by comparison with (2.32), the optimal solution to (2.34) reads

$$x = W^{-1}A^\mathsf{T}(AW^{-1}A^\mathsf{T})^{-1}b + \left(I_n - W^{-1}A^\mathsf{T}(AW^{-1}A^\mathsf{T})A\right)x_0. \tag{2.35}$$

For the special case that $W := I_n$ and $x \in \mathbb{R}^2$, the optimal solution can be visualized as depicted in Figure 2.6.

2.2.3 Singular Value Decomposition

The Singular-Value Decomposition (SVD) of a matrix $A \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(A) = p$ is given by

$$A = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^\mathsf{T} \\ V_2^\mathsf{T} \end{bmatrix} = U_1 \Sigma V_1^\mathsf{T}, \tag{2.36}$$

with the diagonal matrix of singular values $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, ..., \sigma_p)$, $\sigma_1 \geq \sigma_2 \geq \sigma_p > 0$, and $U = [U_1, U_2]$ as well as $V = [V_1, V_2]$ are orthogonal matrices with $U^\mathsf{T}U = I_m$ and $V^\mathsf{T}V = I_n$. Moreover, we have $U_1 \in \mathbb{R}^{m\times p}$ and $V_1 \in \mathbb{R}^{n\times p}$. The factorization of a positive definite matrix

$$W = W^{1/2}W^{1/2} \tag{2.37}$$

is obtained as $W^{1/2} = U_1\Sigma^{1/2}V_1^\mathsf{T}$ with $\Sigma^{1/2} = \operatorname{diag}(\sigma_1^{1/2}, \sigma_2^{1/2}, ..., \sigma_p^{1/2})$. In practice, one can often use the Cholesky decomposition $W = LL^\mathsf{T}$ using L instead of $W^{1/2}$.

Figure 2.7 extends Figure 2.3 to illustrate the fundamental subspaces of A in terms of the SVD. When a vector x is multiplied with A, $x_\mathsf{R} = V_1\Sigma^{-1}U_1^\mathsf{T}x$ is first transformed by V_1^T

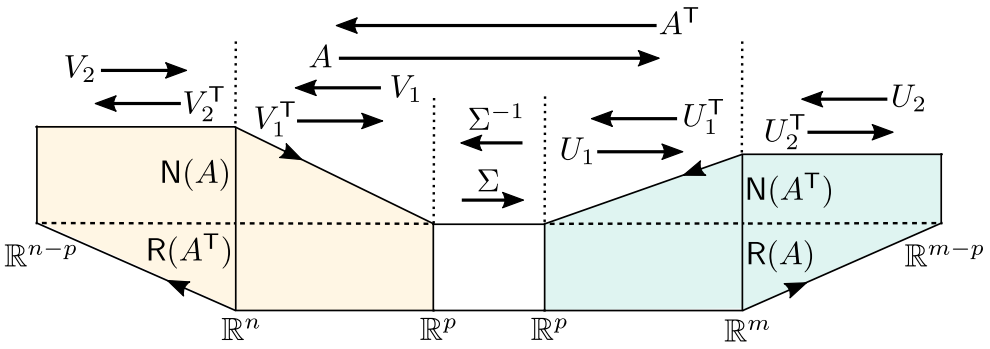


Figure 2.7: The singular value decomposition (SVD) describes A in terms of transformations between several subspaces. Adapted from Beard (2002).

into $\mathbb{R}^p$, then scaled by Σ, and finally transformed by U_1 to reach $\mathbb{R}^m$. During the journey through the fundamental subspaces of A, the vector x lost the part $x_\mathsf{N} = V_2^\mathsf{T} x$. We refer to x_N as "lost", as it cannot be recovered via multiplication of Ax_R with A^+.

A comparison of Figure 2.7 with Figure 2.4 and Figure 2.5 shows that the aforementioned linear equations are special cases inherent in the SVD of a matrix with $p = \text{rank}(A)$ and $p < \min(m, n)$. Therefore, the solution to the linear matrix equation $Ax = b$ can be divided into solving the least squares equation $V_1^\mathsf{T} x = \Sigma^{-1}\zeta$, with $\zeta \in \mathbb{R}^p$ for x, as well as solving the minimum norm equation $U_1\zeta = b$ for ζ. The minimum-norm solutions to the above equations are stated in (2.22) and (2.29) such that

$$x^\star = (V_1 V_1^\mathsf{T})^{-1} V_1 \Sigma^{-1}\zeta, \text{ and } \zeta^\star = U_1^\mathsf{T}(U_1 U_1^\mathsf{T})^{-1} b. \tag{2.38}$$

Since V_1 and U_1 are orthogonal matrices, one obtains

$$x^\star = A^+ b, \tag{2.39}$$

with the MP inverse of A,

$$A^+ = V_1 \Sigma^{-1} U_1^\mathsf{T} b. \tag{2.40}$$

If $p < m$, $\mathsf{N}(A^\mathsf{T})$ contains non-zero vectors, such that the general solution to the linear matrix equation reads

$$x = A^+ b + (I_n - A^+ A)h, \tag{2.41}$$

with h being an arbitrary n-vector. A matrix A^+ is called the Moore-Penrose (MP) inverse of a matrix A if the following conditions are fulfilled, cf. (Udwadia and Kalaba, 2007, p. 45):

$$\begin{aligned}
&1)\ AA^+A = A, \\
&2)\ A^+AA^+ = A^+, \\
&3)\ AA^+ = (AA^+)^\mathsf{T}, \text{ that is } AA^+ \text{ is symmetric}, \\
&4)\ A^+A = (A^+A)^\mathsf{T}, \text{ that is } A^+A \text{ is symmetric.}
\end{aligned} \tag{2.42}$$

(2.43)

Generalized inverses that only fulfill the first condition in (2.42) are referred to as G-inverses, and generalized inverses that fulfill the first and third condition in (2.42) are referred to as L-inverses. Therefore, the MP inverse is also an L-inverse and the L-inverse is also a

G-inverse. However, several L-inverses and G-inverses of a matrix A may exist whereas the MP inverse of any matrix is *unique* and its existence is guaranteed (Udwadia and Kalaba, 2007, p. 46). The inverse A^{-1} of a square nonsingular matrix A fulfills all conditions in (2.42) and hence could be deemed as a special case of the MP inverse. Udwadia and Kalaba (2007) prove some important properties for an MP inverse A^+ of a matrix $A \in \mathbb{R}^{m\times n}$ of any rank, namely

$$A^+ = A^\mathsf{T}(AA^\mathsf{T})^+, \tag{2.44}$$
$$A^+ = (A^\mathsf{T}A)^+A^\mathsf{T}, \tag{2.45}$$
$$\mathsf{R}(A^\mathsf{T}) = \mathsf{R}(A^+), \tag{2.46}$$
$$\mathsf{R}(A) = \mathsf{R}(AA^+), \tag{2.47}$$
$$\mathsf{R}(A^+) = \mathsf{R}(A^+A), \tag{2.48}$$
$$\mathsf{N}(A) = \mathsf{R}(I - A^+A). \tag{2.49}$$
$$\tag{2.50}$$

and if $\operatorname{rank}(A) = n$, then

$$A^+ = (A^\mathsf{T}A)^{-1}A^\mathsf{T} \text{ and } A^+A = I_n. \tag{2.51}$$

2.3 Lagrangian Optimization

In Section 2.2, we obtained expressions for the solutions of linear equations solely by means of algebra. In what follows, we outline the solution of a linear matrix equation by additional means of calculus. In the later sections, these differing views on the same problem provide additional insight into the structure underlying rigid body mechanics. The following exposition of Lagrangian optimization is adapted from (Boyd et al., 2004, Chapter 5) unless noted otherwise.

2.3.1 Duality

Consider the primal optimization problem

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g_i(x) = 0,\ i = 1,\dots,m \\ & h_j(x) \le 0,\ j = 1,\dots,k \end{aligned} \tag{2.52}$$

with optimization variable $x \in \mathbb{R}^n$, the non-empty domain of f being $\mathcal{D} = \bigcap_{i=1}^m \mathbf{dom}\ g_i \cap \bigcap_{j=1}^k \mathbf{dom}\ h_j$. The variable $x \in \mathbb{R}^n$ is a *feasible* point with respect to the constraints $g_i(x)$ and $h_(x)$, if $g_i(x') = 0$ and $h_i(x') \le 0$. The solution to (2.52) is denoted by $x^\star$ and its function value by $p^\star = f(x^\star)$. As a first step towards finding a solution of (2.52), define the *Lagrangian* as

$$L(x,\lambda) = f(x) + \lambda^\mathsf{T}g(x) + \nu^\mathsf{T}h(x), \tag{2.53}$$

with $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}$, $\mathbf{dom}\ L = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^k$, $g(x) = \operatorname{vec}(g_1,\dots,g_m)$, and $h(x) = \operatorname{vec}(h_1,\dots,h_m)$. The vectors λ, ν are referred to as *Langrange mutliplier* vectors or dual variables. Moreover, define the dual function

$$\ell(\lambda,\nu) = \inf_{x\in\mathcal{D}} L(x,\lambda,\nu). \tag{2.54}$$

The dual function $\ell : \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}$ is concave even if (2.52) is non-convex and forms a lower-bound bound on $p^\star$ for any $\nu_j \geq 0$, writing

$$\ell(\lambda, \nu) \leq L(x', \lambda, \nu) \leq f(x'), \tag{2.55}$$

where x' is a feasible point. The second inequality follows from $\lambda^\mathsf{T} g(x') = 0$ and $\nu^\mathsf{T} h(x') \leq 0$ when $\nu \geq 0$. The goal of the *dual optimization problem* is to find the largest lower bound as in (2.55)

$$\max_{\lambda,\nu} \ell(\lambda, \nu) \text{ subject to } \nu_j \geq 0, \lambda_i \in \mathbb{R}. \tag{2.56}$$

The solution of (2.56) is denoted by $\{\nu^\star, \lambda^\star\}$. The primal problem (2.52) seeks to minimize f resulting in $p^\star$, while its dual problem (2.56) seeks to maximize $\ell(\lambda, \nu)$ yielding $d^\star := \ell(\lambda^\star, \nu^\star)$. If the *duality gap* $p^\star - d^\star$ is zero, the optimization problem is referred to as having *strong duality*. Strong duality allows us to find $p^\star$ solely by optimization of the convex optimization problem (2.56). We note that not every problem in the form of (2.52) possesses strong duality. However, while not every convex problem has strong duality, the problems we consider in this work such as linear problems do. If strong duality holds then $\{x^\star, \lambda^\star, \nu^\star\}$ constitute a saddle-point of the Lagrangian L. The converse also holds that is if $\{x^\star, \lambda^\star, \nu^\star\}$ is a saddle point of L, then the duality gap is zero. *Constraint qualifications* are conditions that ensure whether strong duality holds in an optimization problem.

2.3.2 Karush-Kuhn-Tucker Conditions

For an optimization problem of the form in (2.52) with differentiable functions f, g, h, and a zero duality gap, the primal and dual optimal points $\{x^\star, \lambda^\star, \nu^\star\}$ must satisfy the *Karush-Kuhn-Tucker* (KKT) conditions

$$\begin{aligned} \nabla f(x^\star) + (\lambda^\star)^\mathsf{T} \nabla g(x^\star) + (\nu^\star)^\mathsf{T} \nabla h(x^\star) &= 0 \\ h(x^\star) &\leq 0 \\ g(x^\star) &= 0 \\ \nu_i^\star &\geq 0,\ i = 1, \ldots, k \\ \nu_i^\star h_i(x^\star) &= 0,\ i = 1, \ldots, k. \end{aligned} \tag{2.57}$$

The first three equations of (2.57) follow from setting the gradient of the Lagrangian (2.53) to zero that is $\nabla L = [\nabla_x L^\mathsf{T}, \nabla_\lambda L^\mathsf{T}, \nabla_\nu L^\mathsf{T}]^\mathsf{T} = 0$. The last equation of (2.57) is a complementary condition enforcing that if $h_i(x^\star) > 0$ then $\nu_i = 0$. The KKT conditions are cardinal for constrained optimization and many convex optimization algorithms make use of the KKT conditions.

Example 2.10. *Assume the following optimization problem*

$$\begin{aligned} &\textit{maximize } f(x) = x_1 x_2 + 1 \textit{ with } x_1 \in \mathbb{R}, x_2 \in \mathbb{R} \\ &\textit{subject to } g(x) = x_1^2 + x_2^2 - 1 = 0. \end{aligned} \tag{2.58}$$

The Lagrangian of (2.58) *reads*

$$\mathcal{L} = x_1 x_2 + 1 + \lambda(x_1^2 + x_2^2 - 1) \textit{ with } \lambda \in \mathbb{R}, \tag{2.59}$$

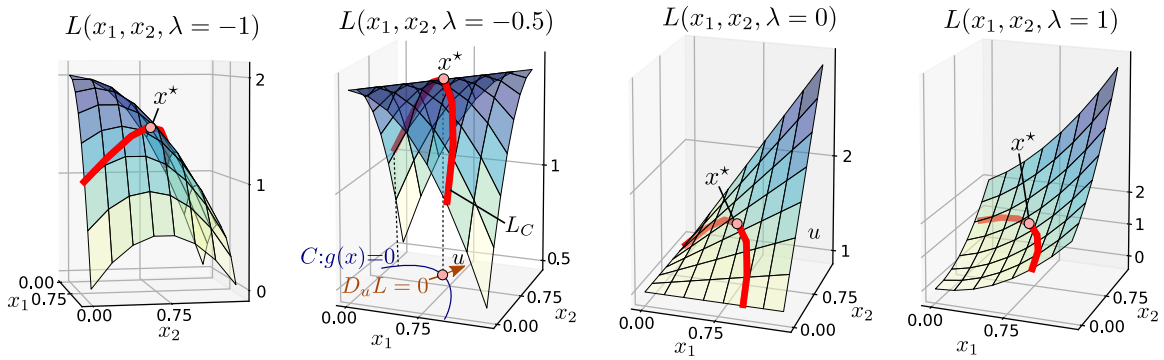


Figure 2.8: 3D plot of the Lagrangian in Example 2.10 for different Lagrange multipliers λ. The red line denotes all values of $L(x, \lambda)$ such that $g(x) = 0$.

such that the KKT conditions[2] *are*

$$\nabla L = \begin{bmatrix} x_2 + \lambda 2x_1 & x_1 + \lambda 2x_2 & (x_1^2 + x_2^2 - 1) \end{bmatrix}^\mathsf{T} = 0. \tag{2.60}$$

By solving (2.60) *the optimal solution is obtained at* $x_1 = \pm\frac{1}{\sqrt{2}}$, $x_2 = \pm\frac{1}{\sqrt{2}}$, *and* $\lambda = -0.5$. *As pointed out in (Kalman, 2009, p. 194), the term* $\lambda g(x_1, x_2)$ *"levels" L such that for the optimal points* $\{x_1^\star, x_2^\star, \lambda^\star\}$ *the KKT condition is fulfilled.*

Figure 2.8 illustrates the change of L as a function of λ. *Here* $C = \{x \in \mathbb{R}^2 : g(x) = 0\}$ *denotes the curve in the* x_1-x_2 *space for which the equality constraint is zero. Further,* $u \in \mathbb{R}^2$ *denotes a vector normal to the curve C. As pointed out by Kalman (2009), at the point* $x^\star = \{x_1^\star, x_2^\star\}$, *the directional derivative of L in direction of u must be zero*

$$D_u L = \nabla L^\mathsf{T} u = 0. \tag{2.61}$$

With $D_u L = D_u(f + \lambda g)$, *it follows from* (2.61) *that* $\lambda = -D_u f / D_u g$.

2.3.3 Norm Minimization with Equality Constraints

We finally return to the optimization problem

$$\begin{aligned} &\text{minimize } \|Cx - d\| \\ &\text{subject to } Ax = b. \end{aligned} \tag{2.62}$$

The minimization of (2.62) is equivalent to optimizing

$$\begin{aligned} &\text{minimize } \frac{1}{2}\|Cx - d\|^2, \\ &\text{subject to } Ax = b. \end{aligned} \tag{2.63}$$

As special cases, (2.63) includes the aforementioned least-squares problem as in (2.26) ($C = A$, $d = b$, and no constraints) and least-norm problem as in (2.35) ($C = W^{1/2}$ and $d = W^{1/2}x_0$). The Lagrangian (2.53) of (2.63) is obtained as

$$L(x, \lambda) = \frac{1}{2}x^\mathsf{T}Wx - d^\mathsf{T}Cx + \frac{1}{2}d^\mathsf{T}d + \lambda^\mathsf{T}(Ax - b), \tag{2.64}$$

[2] We note that if an optimization problem is solely subject to equality constraints the conditions in (2.60) are usually referred to as the first order optimality condition.

with $W := C^\mathsf{T}C$, $W \geq 0$, and the vector of Lagrange multipliers $\lambda \in \mathbb{R}^m$. Subsequently, the KKT conditions (2.57) for finding an optimal point of (2.64) read

$$\nabla L = \begin{bmatrix} \nabla_x L \\ \nabla_\lambda L \end{bmatrix} = \begin{bmatrix} Wx - C^\mathsf{T}d + A^\mathsf{T}\lambda \\ Ax - b \end{bmatrix} = 0. \tag{2.65}$$

The above equation can be rewritten as the block matrix equation

$$\begin{bmatrix} W & A^\mathsf{T} \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} C^\mathsf{T}d \\ b \end{bmatrix}. \tag{2.66}$$

The solution to (2.66) depends on the properties of matrices and vectors involved (Boyd et al., 2004, p. 522). In optimization literature, the block-matrix on the left side of (2.66) is referred to as the KKT matrix. If the KKT matrix is non-singular, the matrix inversion lemma yields a unique solution to (2.66), cf. (Schön and Lindsten, 2011, p. 10) and (Boyd et al., 2004, p. 650), as

$$\begin{bmatrix} x^\star \\ \lambda^\star \end{bmatrix} = \begin{bmatrix} W^{-1} - W^{-1}A^\mathsf{T}(AW^{-1}A^\mathsf{T})^{-1}AW^{-1} & W^{-1}A^\mathsf{T}(AW^{-1}A^\mathsf{T})^{-1} \\ (AW^{-1}A^\mathsf{T})^{-1}AW^{-1} & -(AW^{-1}A^\mathsf{T})^{-1} \end{bmatrix} \begin{bmatrix} C^\mathsf{T}d \\ b \end{bmatrix}. \tag{2.67}$$

Alternatively, (2.67) can be rewritten as

$$x^\star = W^{-1}\left(C^\mathsf{T}d - A^\mathsf{T}(AW^{-1}A^\mathsf{T})^{-1}(AW^{-1}C^\mathsf{T}d + b)\right), \tag{2.68}$$

$$\lambda^\star = (AW^{-1}A^\mathsf{T})^{-1}\left(AW^{-1}C^\mathsf{T}d - b\right). \tag{2.69}$$

For $C := W^{1/2}$ and $d := W^{1/2}x_0$, (2.35) yields the general solution to the weighted minimum norm problem (2.35), that is

$$x = W^{-1}A^\mathsf{T}(AW^{-1}A^\mathsf{T})^{-1}b + \left(I_n - W^{-1}A^\mathsf{T}(AW^{-1}A^\mathsf{T})^{-1}A\right)x_0. \tag{2.70}$$

2.4 Gaussian Process Regression

A Gaussian process (GP) defines a distribution over functions such that every selection of function values $\{f(x_1), f(x_2), ..., f(x_N)\}$ are jointly Gaussian distributed (Williams and Rasmussen, 2006, Chapter 2). In other words, a GP is often referred to as a probabilistic model that defines a normal distribution over functions. GPs are non-parametric and provide a measure of uncertainty of the estimation result in form of their posterior variance. As GPs define normal distributions over functions, they reduce Bayesian inference which often requires the numerical approximation of complicated integrals into a series of computational efficient linear algebraic operations. A function $f(x)$ is modeled through a GP as

$$f(x) \sim \mathcal{GP}\left(m(x), k(x, x')\right) \tag{2.71}$$

with a mean function also being referred to as *mean*

$$m(x) = \mathbb{E}[f(x)], \tag{2.72}$$

and a covariance function also being referred to as *kernel*

$$k(x, x') = \mathrm{Cov}[f(x), f(x')] = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]. \tag{2.73}$$

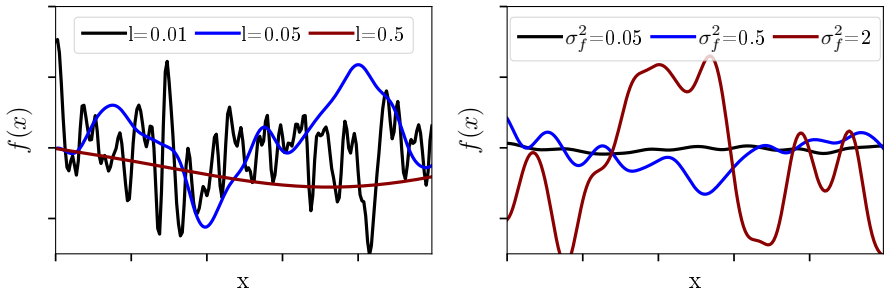


Figure 2.9: GP function samples for different SE kernel hyper-parameters.

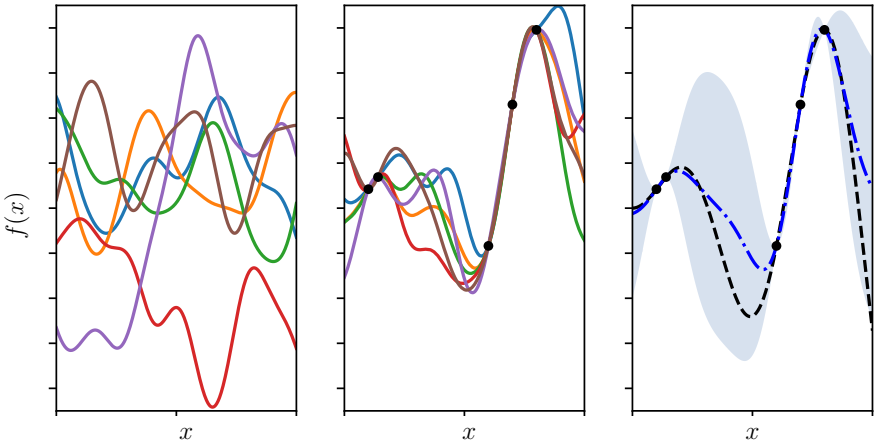


Figure 2.10: **Left**: Samples drawn from a GP prior with SE kernel. **Center:** GP samples after conditioning on observations (black dots). **Right:** Real function (black dash line), GP posterior mean (blue dot dash line), and the posterior GP's 95-percent confidence interval. Figure adapted from Geist (2018).

After the specification of the mean, the kernel determines the mathematical properties of the GP's function approximation and how it extrapolates to new data (Duvenaud, 2014). Following from Mercer's theorem, a kernel $k : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ evaluated at $k(x, x')$ could be thought of as an entry in an "infinitely large" matrix (Hennig, 2020). Not every function with inputs $\{x, x'\}$ qualifies as a kernel. A kernel $k(x, x')$ must be positive semidefinite such that the Gram matrix $\Sigma \in \mathbb{R}^{n \times n}$ with entries $\Sigma_{i,j} = k(x_i, x_j)$ is positive semidefinite writing $v^\mathsf{T} \Sigma v \geq 0$ for all vectors $v \in \mathbb{R}^n$ and hence Σ is a covariance matrix (Williams and Rasmussen, 2006, p. 80).

A commonly used kernel in robot dynamics identification is the squared-exponential (SE) function

$$k\left(x, x'\right) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}\left(x - x'\right)^2\right), \tag{2.74}$$

with the *length-scale* l and the signal variance σ_f^2. Kernel parameters such as $\{l, \sigma_f^2\}$ of a non-parametric model are referred to as hyper-parameters. These hyper-parameters shape the GP's functions space, while in a non-probabilistic parametric model changing the parameters determines a particular element inside the model's function space. Figure 2.9 illustrates how the change in the hyper-parameters of a SE kernel affects the shape of the GP samples.

2.4.1 Multi-output Gaussian Processes

A multi-output GP denotes the extension of the previously discussed one-dimensional GP to a multi-dimensional process. The following exposition of multi-output GPs is taken from *Rath et al. (2021)* and its supplementary material. In a multi-dimensional function $f(x) = [f_1(x) \dots f_n(x)]^\mathsf{T}$ as a multi-task GP, writing $\hat{f} \sim \mathcal{GP}(m(x), K(x, x'))$, where $m(x)$ denotes a vector-valued mean function and $K(x, x')$ a matrix-valued kernel, such that

$$m(x) = \begin{bmatrix} m_1(x) & \dots & m_n(x) \end{bmatrix}^\mathsf{T}, \;\; K(x, x') = \begin{bmatrix} k_{1,1}(x, x') & \cdots & k_{1,n'}(x, x') \\ \vdots & \ddots & \vdots \\ k_{n,1}(x, x') & \cdots & k_{n,n'}(x, x') \end{bmatrix}, \tag{2.75}$$

where $m_j(x)$ is the scalar mean function of $f_j(x)$ and $k_{i,j}(x, x')$ corresponds to the scalar kernel between $f_i(x)$ and $f_j(x')$. Given a data set $\mathcal{D}$ as in (2.1), a multi-output GP defines a multivariate normal distribution over X as $\hat{f}(X) \sim \mathcal{N}(\mu_X, \Sigma_{X,X})$, with the mean vector and covariance matrix respectively

$$\mu_X = \begin{bmatrix} m(x_1)^\mathsf{T} & \dots & m(x_N)^\mathsf{T} \end{bmatrix}^\mathsf{T}, \;\; \Sigma_{X,X'} = \begin{bmatrix} K(x_1, x'_1) & \cdots & K(x_1, x'_{N'}) \\ \vdots & \ddots & \vdots \\ K(x_N, x'_1) & \cdots & K(x_N, x'_{N'}) \end{bmatrix}, \tag{2.76}$$

assuming $X = X'$. The process measurements y are assumed to arise from i. i. d. Gaussian noise, writing

$$y_i = f_i + \epsilon_y, \;\; \text{with} \;\; \epsilon_y \sim \mathcal{N}(0, \Sigma_y), \; \Sigma_y = \text{diag}(\sigma_1^2, ...\sigma_n^2), \tag{2.77}$$

with the noise variance of the i process being σ_i^2. With $Y(X) = \hat{f}(X) + \epsilon_Y$, with $\epsilon_Y \sim \mathcal{N}(0\,, \Sigma_Y)$ and $\Sigma_Y = I_N \otimes \Sigma_y$, the the joint distribution of $Y(X)$ and $\hat{f}(X^*)$ at the prediction points $X^* = \{x_1^* \dots x_{N_\mathrm{P}}^*\}$, reads

$$\begin{bmatrix} \hat{f}(X^*) \\ Y(X) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_{X^*} \\ \mu_X \end{bmatrix}, \begin{bmatrix} \Sigma_{X^*,X^*} & \Sigma_{X^*,X} \\ \Sigma_{X,X^*} & \Sigma_{X,X} + \Sigma_Y \end{bmatrix} \right). \tag{2.78}$$

Conditioning on the observations yields the predictive posterior distribution

$$\hat{f}(X^*)|\mathcal{D}, \Sigma_y \sim \mathcal{N}(\mu^\star, \Sigma^\star), \quad \begin{aligned} \mu^* &= \mu_{X^*} + \Sigma_{X^*,X} \left(\Sigma_{X,X} + \Sigma_Y\right)^{-1} (y - \mu_X), \\ \Sigma^* &= \Sigma_{X^*,X^*} - \Sigma_{X^*,X} \left(\Sigma_{X,X} + \Sigma_Y\right)^{-1} \Sigma_{X,X^*}. \end{aligned} \tag{2.79}$$

In practice, working with multi-output GPs does not significantly differ from one-dimensional GPs if one gets acquainted with the covariance matrix entries at every input point pair being also multi-dimensional. In this regard, if the process is one dimensional that is $n = 1$, the above equations reduce to standard (one-dimensional) GP regression. Figure 2.10 illustrates how conditioning affects the samples of a one-dimensional GP with SE kernel.

Samples $\hat{f} \sim \mathcal{N}(\mu_X, \Sigma_{X,X})$ are drawn from a normal-distribution by computing the Cholesky decomposition L of the positive definite matrix $\Sigma_{X,X} = LL^\mathsf{T}$ and noting that

$$\hat{f} = \mu_X + Lu, \;\; \text{with} \;\; u \sim \mathcal{N}(0, I_n). \tag{2.80}$$

In practice, an additional noise variance ϵ is added to the diagonal of the covariance matrix as this matrix is often nearly rank-deficient leading to numerical problems when computing its Cholesky decomposition (Williams and Rasmussen, 2006, p. 201). In Section 5.3.2, we briefly detail how using the Farthest Point Sampling algorithm to sparsify a data set improves the conditioning number of the covariance matrix.

2.4.2 Linearly Transformed Gaussian Processes

In recent decades, several works suggested using structural knowledge to transform GPs and improve the models sample-efficiency. In this section, we discuss how sums of functions, as well as linear operators acting on functions can be modeled with GP regression.

Sums of functions

Duvenaud (2014) provides a concise discussion on how to construct and combine kernels. In particular, in (Duvenaud, 2014, Section 2.4) it is shown how to model sums of functions with GP regression, leading to the following theorem:

Theorem 2.11. *The sum of GPs* $\hat{f}_i \sim \mathcal{GP}(m_i(x), K_i(x, x'))$ *yields the GP*

$$\hat{f} \sim \mathcal{GP}\left(\sum m_i(x),\ \sum K_i(x, x')\right). \tag{2.81}$$

As shown in Section 5.1.3, in analytical mechanics one often encounters sums of forces which subsequently can be modeled with GPs. Given the GPs $\hat{f}^{\mathrm{a}} \sim \mathcal{GP}(m^{\mathrm{a}}(x), K^{\mathrm{aa}}(x, x'))$, $\hat{f}^{\mathrm{b}} \sim \mathcal{GP}(m^{\mathrm{b}}(x), K^{\mathrm{bb}}(x, x'))$, and $\hat{f}^{\mathrm{c}} \sim \mathcal{GP}(m^{\mathrm{c}}(x), K^{\mathrm{cc}}(x, x'))$, where

$$m^{\mathrm{a}}(x) = m^{\mathrm{b}}(x) + m^{\mathrm{c}}(x), \quad \text{and} \quad K^{\mathrm{aa}}(x, x') = K^{\mathrm{bb}}(x, x') + K^{\mathrm{cc}}(x, x'), \tag{2.82}$$

then these processes define a joint distribution as

$$\begin{bmatrix} \hat{f}^{\mathrm{a}} \\ \hat{f}^{\mathrm{b}} \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} m^{\mathrm{b}}(x) + m^{\mathrm{c}}(x) \\ m^{\mathrm{b}}(x) \end{bmatrix}, \begin{bmatrix} K^{\mathrm{bb}}(x, x') + K^{\mathrm{cc}}(x, x') & K^{\mathrm{bb}}(x, x') \\ K^{\mathrm{bb}}(x, x') & K^{\mathrm{bb}}(x, x') \end{bmatrix}\right). \tag{2.83}$$

Subsequently, one can predict $\hat{f}^{\mathrm{b}}(X^*)$ given measurements $Y = \hat{f}^{\mathrm{a}}(X) + \epsilon_Y$ as

$$\hat{f}^{\mathrm{b}}(X^*) \mid \mathcal{D}, \Sigma_y \sim \mathcal{N}(\mu^{\mathrm{b},*}, \Sigma^{\mathrm{b},*}), \tag{2.84}$$

$$\mu^{\mathrm{b},*} = \mu^{\mathrm{b}}_{X^*} + \Sigma^{\mathrm{ba}}_{X^*,X} \left(\Sigma^{\mathrm{aa}}_{X,X} + \Sigma_Y\right)^{-1} (Y - \mu^{\mathrm{a}}_X), \tag{2.85}$$

$$\Sigma^{\mathrm{b},*} = \Sigma^{\mathrm{bb}}_{X^*,X^*} - \Sigma^{\mathrm{ba}}_{X^*,X} \left(\Sigma^{\mathrm{aa}}_{X,X} + \Sigma_Y\right)^{-1} \Sigma^{\mathrm{ab}}_{X,X^*}. \tag{2.86}$$

Linear operators

Another important form of structural knowledge are linear operations. Here, Solak et al. (2003) showed that one may derive the joint distribution of a GP $\hat{f}$ and its derivative $\partial\hat{f}/\partial x$ such that the process $\hat{f}$ can be conditioned on observations of $\partial\hat{f}/\partial x$. Similar to normal distributions being closed under linear transformations (Schön and Lindsten, 2011), Jidling et al. (2017) emphasize that GPs are closed under linear operations and provided the following Theorem 2.12.

Theorem 2.12. *A GP* $f \sim \mathcal{GP}(m(x), K(x, x'))$ *is closed under a linear operator* $\mathcal{B}_x$*, cf. Williams and Rasmussen (2006), writing*

$$\mathcal{B}_x f(x) \sim \mathcal{GP}\left(\mathcal{B}_x m(x),\ \mathcal{B}_x K(x, x') \mathcal{B}_{x'}^{\mathsf{T}}\right). \tag{2.87}$$

As detailed in the supplementary material of Jidling et al. (2017), the covariance function of a linear transformed GP reads

$$\begin{aligned}\operatorname{Cov}(\mathcal{B}_x f(x), \mathcal{B}_{x'} f(x')) &= \mathbb{E}\left[\big(\mathcal{B}_x f(x) - \mathcal{B}_x m(x)\big)\big(\mathcal{B}_{x'} f(x') - \mathcal{B}_{x'} m(x')\big)^{\mathsf{T}}\right], \\ &= \mathcal{B}_x \mathbb{E}\left[\big(f(x) - m(x)\big)\big(f(x') - m(x')\big)^{\mathsf{T}}\right] \mathcal{B}_{x'}^{\mathsf{T}}, \\ &= \mathcal{B}_x K(x, x') \mathcal{B}_{x'}^{\mathsf{T}} \end{aligned} \tag{2.88}$$

As initially pointed out in the supplementary material of Jidling et al. (2017) and further discussed in (Geist and Trimpe, 2021), the following linear operators are often found in rigid body dynamics:

Matrix transformations, writing

$$\hat{f}_i^{\mathcal{C}}(x) = \mathcal{C}_i(x) \hat{f}_i(x). \tag{2.89}$$

Differentiation of a function, writing

$$\hat{f}_i^{\nabla_x}(x) = \nabla_x f_i(x)|_{x=z}, \tag{2.90}$$

which we denote by an abuse of notation and assuming $x \in \mathbb{R}^n$ as

$$\hat{f}_i^{\nabla_x}(x) = \nabla_x f_i(x) = \left[\frac{\partial f_i}{\partial x_1}, \ldots, \frac{\partial f_i}{\partial x_n}\right]^{\mathsf{T}}. \tag{2.91}$$

Substitution of input variables by a nonlinear mapping $x = u(z)$ such that

$$\hat{f}_i^{u}(x) = f_i(z)|_{z=u(x)}, \tag{2.92}$$

e.g., instead of using an angle coordinate $x \mathrel{\widehat{=}} \phi$ 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 (Calandra et al., 2016).

Assume that two processes are modeled as

$$\hat{f} \sim \mathcal{GP}(m(x), K(x, x')) \;\text{ and }\; \mathcal{B}_x \hat{f} \sim \mathcal{GP}(\mathcal{B}_x m(x), \mathcal{B}_x K(x, x') \mathcal{B}_{x'}^{\mathsf{T}}). \tag{2.93}$$

As due to 2.12, the above processes are both GPs, one can define their joint distribution as

$$\begin{bmatrix} \hat{f} \\ \mathcal{B}_x \hat{f} \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} m(x) \\ \mathcal{B}_x m(x) \end{bmatrix}, \begin{bmatrix} K(x, x') & K(x, x') \mathcal{B}_{x'}^{\mathsf{T}} \\ \mathcal{B}_x K(x, x') & \mathcal{B}_x K(x, x') \mathcal{B}_{x'}^{\mathsf{T}} \end{bmatrix}\right). \tag{2.94}$$

In particular, the covariance function becomes $\operatorname{Cov}(\mathcal{B}_x \hat{f}, \hat{f}) = \mathcal{B}_x K(x, x')$.

2.4.3 Maximum Likelihood Estimation

Finding a good prior for a GP's hyper-parameters is a significant challenge. As we point out in *(Rath et al., 2021)*, a standard method for estimating a GP's hyper-parameter is Type-II maximum likelihood estimation (MLE). Given $Y \sim \mathcal{N}(\mu_X, \Sigma_{X,X} + \Sigma_Y)$, type-II MLE seeks a $\theta^\star = \{\theta^\star_{\mathrm{A}}, \theta^\star_{\mathrm{M}}\}$ minimizing the negative log likelihood, writing

$$\theta^\star = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left[(Y-\mu_X)^\mathsf{T}\left(\Sigma_{X,X}+\Sigma_Y\right)^{-1}(Y-\mu_X) + \log\left|\Sigma_{X,X}+\Sigma_Y\right|\right] + \frac{N}{2}\log 2\pi. \tag{2.95}$$

The left term inside the bracket in (2.95) assigns a cost to the data-fit while the log-determinant term penalizes the function's complexity (Williams and Rasmussen, 2006, p. 113). For example, assume $\Sigma_{X,X}$ is defined in terms of an SE kernel, if the lengthscale l approaches zero the process becomes very flexible such that observations are over-fitted. In this case, $\Sigma_{X,X} + \Sigma_Y$ becomes a diagonal matrix and the log-determinant attains its maximum value. Whereas, if the lengthscale l approaches infinity, the change in the process approaches zero such that the observations are under-fitted. For this case, the determinant becomes nearly rank deficient such that the log-determinant acquires a large negative value. In practice, the effect of the log-determinant term on the likelihood can be underwhelming such that it is common practice to resort to maximum a posteriori estimation in which an additional prior parameter distribution is multiplied with the likelihood.

Chapter 3

Rigid Body Dynamics

At the beginning of this work, we highlighted some of the challenges of robot dynamics identification. In particular, we emphasized that a promising direction toward improving the sample efficiency of a data-driven dynamics model forms the inclusion of additional prior knowledge into the model. Prior knowledge or, using a less Bayesian lingo, "structural knowledge", we defined as functional relationships that are known to us and that we must not identify from data. For robot dynamics identification, a useful source of structural knowledge forms rigid body dynamics. The following exposition of Lagrangian mechanics is, to a large part, identical to *Geist and Trimpe (2021)*[1] while being rearranged and being significantly extended. Here, we provide *one* perspective on how in Lagrangian mechanics, the system's equations of motion (EOM) emerge from the interplay of fundamental principles. Albeit, algorithms for the derivation of EOM may differ in the principles on which they are built and the system's properties onto which these frameworks apply. Other formulations of rigid body mechanics that are not detailed in this dissertation include Gibbs–Appell's EOM, Kane's EOM, or Hamiltonian Dynamics. Although, if we stick to Lagrangian mechanics, algorithms that derive a robot's dynamic equations may possess over computational routines that make them particularly fast to execute, *e.g.*, the articulated-body algorithm (Featherstone, 2008, p. 128) or, even use different Algebras such as Lie Algebra (Murray et al., 1994), Spatial Vector Algebra (Featherstone, 2008, p. Chapter 2) or Geometric Algebra (Hitzer et al., 2013). The EOM derived through these frameworks are equivalent in the sense that they must describe the same unique motion of the mechanical system.

While the following exposition is limited to robotic systems, rigid body dynamics is a vast field. To keep the discussion concise and provide a consecutive narrative, we first limit the discussion to the Newton-Euler equations of a system of N_b rigid bodies subject to holonomic constraints. By resorting to generalized coordinates and explicit coordinate transformations, we eliminate constraint forces from the EOM. Then, we briefly discuss how the inertia matrix and conservative forces can be rewritten in terms of potential functions in the Euler-Lagrange equations. Afterward, we discuss how additional implicit constraints are incorporated into the EOM. With the discussion of explicit and implicit constraints, we illustrate how constraint equations and potential functions provide structural knowledge of the vector spaces in which certain forces are bound to lie. These insights, we then use to discuss and extend recent works on physics-informed regression in the subsequent chapter.

[1] Wiley remains the copyrights holder for all texts and figures which have been taken from *Geist and Trimpe (2021)*.

3.1 Kinematics and Constraints

To derive the EOM of a multibody system, the motion variables – position, velocity, and acceleration – of its N_b rigid bodies must be described with respect to an inertial frame. The position of every point of the i-th rigid body can be described by a position vector $r_i(t) \in \mathbb{R}^3$ pointing to the origin of a body-fixed frame with respect to an inertial frame and a rotation matrix $R_i(t) \in SO(3)$ describing the rotation of the body-fixed coordinate frame with respect to the inertial frame. $SO(3)$ denotes the subgroup of orthogonal matrices of size three with determinant $+1$. Practically speaking, $SO(3)$ describes the space of all plausible physical rotations in $\mathbb{R}^3$. Subsequently, the multibody system has at most $6N_\mathrm{b}$ degrees of freedom (DOF).

The velocity of a body is described by the translational velocity $\dot{r}_i(t) = \frac{dr_i}{dt}$ and the rotational velocity $\omega_i(t) = \frac{d\varphi_i}{dt}$ with the infinitesimal instantaneous rotation vector $\varphi_i \in \mathbb{R}^3$ (Woernle, p. 67). Alternatively, the rotational velocity $\omega_i(t) = [\omega_1, \omega_2, \omega_3]^\mathsf{T}$ can be described in terms of $R_i(t)$ via the Poisson equation, cf. (Woernle, p. 37) and (Schiehlen and Eberhard, 2014, p. 28), as

$$S_i(t) = \mathrm{crossp}\{\omega_i\} = \dot{R}_i(t) R_i(t)^\mathsf{T}, \text{ with } \mathrm{crossp}\{\omega\} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{3.1}$$

If a constant vector p' is transformed by a rotation such that $p(t) = R_i(t)p'$, then the time-derivative of $p(t)$ follows as, cf. (Woernle, p. 37) and (Siciliano et al., 2010, p. 107),

$$\dot{p}(t) = \dot{R}_i(t)p' = S_i(t)R_i(t)p' = \omega_i \times R_i(t)p'. \tag{3.2}$$

The rotation matrix $R_i(t)$ is expressed in terms of a generalized coordinate vector $q(t)$, writing $R_i(t) := R(q)$. Depending on the given mechanical system, different choices for q exist. For example, one could use spherical coordinates to denote a point in space (Schiehlen and Eberhard, 2014, p. 14) or Cardano angles to describe rotations (Schiehlen and Eberhard, 2014, p. 24). The vector q is termed *minimal* if it consists of independent coordinates that equal the system's DOF. Equation (3.1) conceptually connects a body's vector of instantaneous change in rotation $\dot{\varphi}_i$ with the description of rotation by finite coordinates q.

Constraints

In multibody systems, the rigid bodies are usually subject to mechanisms that apply constraint forces. These constraint forces reduce the system's DOF or constrain the system's motion. In this work, the term constraints refers to algebraic equations that describe the system's admissible states (Featherstone, 2008, p. 44).

Constraints can be either holonomic or nonholonomic. A constrained system is holonomic if its position variables are integrals of the velocity variables; otherwise, the system is nonholonomic (Featherstone, 2008, p. 41). Alternatively put, a constraint that depends on velocity variables is holonomic if it can be obtained by differentiating another constraint equation (that only depends on position variables and time) with respect to time. Holonomic constraints allow finding explicit coordinate transformation, which when inserted into the original implicit constraint of the form $c(q) = 0$ reduces this equation to an identity. The following example helps to visualize the previous statement.

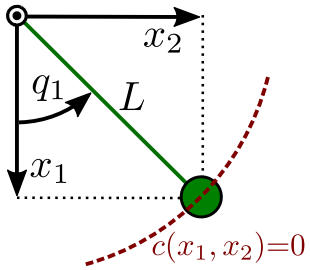


Figure 3.1: Coordinates and constraint of a point-mass pendulum.

Example 3.1 (Pendulum constraints)**.** *Consider an ideal pendulum as shown in Figure 3.1. A point-mass m is constrained by a rod of length L to move on a circular path which can be described via the implicit constraint equation*

$$x_1^2 + x_2^2 = L^2. \tag{3.3}$$

The above constraint is holonomic. Instead of using the Cartesian position coordinates $\{x_1, x_2\}$, the motion of the system can be described using the angle coordinate q_1. The angle coordinate q_1 denotes one possible minimal generalized coordinate description of the system. Every point in the generalized coordinate space spanned by q_1 is admissible with the constraint (3.3)*. To see this, the explicit constraint transformation*

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = L \begin{bmatrix} \cos(q_1) \\ \sin(q_2) \end{bmatrix} \tag{3.4}$$

can be inserted into (3.3) *to yield the identity $L^2(\cos(q_1)^2 + \sin(q_1)^2) = L^2$.*

3.2 Explicitly Constrained Dynamics

To keep the discussion concise, we make the following assumptions for Section 3.2 and Section 3.3:

- The EOM of a system of N_b rigid bodies shall be expressed in terms of n_q minimal generalized coordinates.
- The effect of the constraint forces onto the system's motion are modeled through n_E independent and consistent holonomic constraint equations such that $n_q = 6N_\mathrm{b} - n_\mathrm{E}$.
- The origin of the body-fixed frame lies at the body's center of gravity (COG).

While these assumptions are common for the derivation of robot dynamics, a more comprehensive description of multibody dynamics is given in Schiehlen and Eberhard (2014).

With the above assumptions, the Newton-Euler EOM of the i-th rigid body with respect to the COG read, cf. (Schiehlen and Eberhard, 2014, p. 76),

$$\tilde{M}_i \ddot{p}_i = F_{\mathrm{C},i} + F_{\mathrm{e},i} + F_{\mathrm{E},i} \tag{3.5}$$

with $\ddot{p}_i = \mathrm{vec}\{\ddot{r}_i, \dot{\omega}_i\}$, the block-diagonal matrix as $\tilde{M}_i = \mathbf{diag}\{m_i I_3, \Theta_{S,i}\}$ consisting of the body's mass m_i and its inertia matrix $\Theta_{S,i}$, the bias force $F_{\mathrm{C},i} = F_{\mathrm{G},i} + \mathrm{vec}\{\mathbf{0}, -\tilde{\omega}_i \Theta_{S,i} \omega_i\}$ with the conservative forces $F_{\mathrm{G},i}$, the impressed forces $F_{\mathrm{e},i} = \mathrm{vec}\{f_{\mathrm{e},i}, \tau_{\mathrm{e},i}\}$, and the explicit constraint forces $F_{\mathrm{E},i} = \mathrm{vec}\{f_{\mathrm{E},i}, \tau_{\mathrm{E},i}\}$ (Schiehlen and Eberhard, 2014, p. 76).

3.2.1 Explicit Constraints

As the system is by assumption subject to n_E independent holonomic constraints, it has $n_\mathrm{q} = 6N_\mathrm{b} - n_\mathrm{E}$ DOF. In turn, the i-th bodies position and orientation can be written in terms of explicit constraints as

$$r_i(t) = r_i(q,t), \qquad R_i(t) = R_i(q,t). \tag{3.6}$$

Note that a free system (that is no constraint forces act onto the bodies) can be seen as a special case of a holonomic constrained system in which $n_\mathrm{E} = 0$, q is of dimension $6N_\mathrm{b}$, and (3.6) denotes a suitable coordinate transformation (Schiehlen and Eberhard, 2014, p. 51). Differentiation of (3.6) with respect to time yields constraint equations for the system's velocities and accelerations as

$$\dot{r}_i(q,\dot{q},t) = J_{r,i}\dot{q} + \frac{\partial r_i}{\partial t}, \qquad \omega_i(q,\dot{q},t) \quad = J_{R,i}\dot{q} + \frac{\partial \varphi_i}{\partial t}, \tag{3.7}$$

$$\ddot{r}_i(q,\dot{q},\ddot{q},t) = J_{r,i}\ddot{q} + \dot{J}_{r,i}\dot{q} + \frac{\partial \dot{r}_i}{\partial t}, \qquad \dot{\omega}_i(q,\dot{q},\ddot{q},t) = J_{R,i}\ddot{q} + \dot{J}_{R,i}\dot{q} + \frac{\partial \dot{\varphi}_i}{\partial t}, \tag{3.8}$$

with the translational Jacobian matrix $J_{r,i}(q,t) \in \mathbb{R}^{3\times n_\mathrm{q}}$ and the rotational Jacobian matrix $J_{R,i}(q,t) \in \mathbb{R}^{3\times n_\mathrm{q}}$ (Woernle, p. 195). One can rewrite (3.8) more compactly as

$$\ddot{p}_i(q,\dot{q},\ddot{q},t) = J_i\ddot{q} + \tilde{J}_i, \tag{3.9}$$

with the Jacobian matrix $J_i = [J_{r,i}^\mathsf{T}, J_{R,i}^\mathsf{T}]^\mathsf{T}$ and $\tilde{J}_i = \mathrm{vec}\{\dot{J}_{r,i}\dot{q} + \frac{\partial \dot{r}_i}{\partial t},\ \dot{J}_{R,i}\dot{q} + \frac{\partial \dot{\varphi}_i}{\partial t}\}$.

3.2.2 D'Alembert's Principle with Explicit Constraints

The D'Alembert principle (also referred to as the D'Alembert-Lagrange principle) is one of many constraint-related principles of analytical mechanics. These principles connect constraint forces to the constraint equations that they evoke. Other principles are Jourdain's principle of least constraint (Jourdain, 1909), Kane's principle (being closely connected to Jourdaine's principle (Piedboeuf, 1993)), or Gauss principle of least constraint. The main object of interest in D'Alembert's principle forms virtual displacements. Virtual displacements are infinitesimal vectors that are compatible with the constraints while by definition not varying the time variable (Udwadia and Kalaba, 2007, p. 133), (Schiehlen and Eberhard, 2014, p. 85). A thorough introduction to virtual displacements and their connection to the system's actual and possible displacements is provided in (Layton, 2012, Section 2.5). In Section 3.4, we further extend the discussion on virtual displacements when in addition to explicit constraints, also implicit constraints are incorporated into the EOM.

To eliminate the constraint forces from (3.5), it is assumed that the constraints are *ideal*. A constraint is called ideal, if its constraint forces do zero work under the virtual displacements $\delta p_i = \mathrm{vec}\{\delta r_i, \delta\varphi_i\}$. In turn, it is postulated that the ideal constraint forces $F_{\mathrm{E},i}$ respect the following inner product

$$\sum_{i=1}^{N_\mathrm{b}} \delta p_i^\mathsf{T} F_{\mathrm{E},i} = \delta p^\mathsf{T} F_\mathrm{E} = 0, \tag{3.10}$$

with the δp_i of all bodies being denoted jointly as $\delta p = \mathrm{vec}\{\delta p_1, \delta p_2, \dots \delta p_{N_\mathrm{b}}\}$ as well as the explicit constraint forces of all bodies being denoted as $F_\mathrm{E} = \mathrm{vec}\{F_{\mathrm{E},1}, F_{\mathrm{E},2}, \dots, F_{\mathrm{E},N_\mathrm{b}}\}$.

Note that, if the i-th body is not subject to any constraint forces then $F_{\mathrm{E},i} = 0$. Equation (3.10) is being referred to as the D'Alembert principle (d'Alembert, 1743; Lagrange, 1787), which in its multibody systems extended form (Schiehlen and Eberhard, 2014, p. 92) by insertion of (3.5) reads

$$\sum_{i=1}^{N_\mathrm{b}} \delta p_i^\mathsf{T} \left(\tilde{M}_i \ddot{p}_i - F_{c,i} - F_{e,i} \right) = 0. \tag{3.11}$$

Due to the presence of F_E, the components of δp_i are dependent on each other.

3.2.3 Equations of Motion

In what follows, we outline how to obtain the EOM from (3.11) by resorting to explicit constraint equations that are expressed in terms of a minimal set of generalized coordinates. Importantly, the vector of virtual displacements δp_i can be expressed in terms of the generalized virtual displacement vector $\delta q \in \mathbb{R}^{n_q}$ using (3.7), cf. (Schiehlen and Eberhard, 2014, p. 50), such that

$$\delta p_i = J_i \delta q. \tag{3.12}$$

By inserting (3.12) into (3.10) one obtains

$$\sum_{i=1}^{N_\mathrm{b}} \delta q^\mathsf{T} J_i^\mathsf{T} F_{\mathrm{E},i} = \delta q^\mathsf{T} J^\mathsf{T} F_\mathrm{E} = 0, \tag{3.13}$$

with $J = [J_1^\mathsf{T}, J_2^\mathsf{T}, \dots J_{N_\mathrm{b}}^\mathsf{T}]^\mathsf{T}$. As (3.13) must hold for an *arbitrary* δq, we also have $J^\mathsf{T} F_\mathrm{E} = 0$ such that

$$F_\mathrm{E} \in \mathsf{N}(J^\mathsf{T}). \tag{3.14}$$

Further, with (3.14) and $\mathbb{R}^{6N_\mathrm{b}} = \mathsf{R}(J) \oplus \mathsf{N}(J^\mathsf{T})$, cf. Beard (2002), from (3.10) follows

$$\delta p \in \mathsf{R}(J). \tag{3.15}$$

By transformation of (3.11) into generalized coordinate form using (3.6), (3.7), (3.8), and (3.12), one obtains

$$\delta q^\mathsf{T} \sum_{i=1}^{N_\mathrm{b}} J_i^\mathsf{T} \left(\tilde{M}_i (J_i \ddot{q} + \tilde{J}_i) - F_{\mathrm{C},i} - F_{\mathrm{e},i} \right) = 0. \tag{3.16}$$

As q is assumed to be minimal, (3.16) must hold for any δq such that the local EOM of the i-th rigid body are obtained as $M_i \ddot{q} = Q_{C,i} + Q_{e,i}$, with $M_i(q,t) = J_i^\mathsf{T} \tilde{M}_i J_i$, $Q_{\mathrm{C},i}(q,\dot{q},t) = J_i^\mathsf{T}(F_{\mathrm{C},i}(q,\dot{q},t) - \tilde{M}_i \tilde{J}_i)$, and $Q_{\mathrm{e},i}(q,\dot{q},t) = J_i^\mathsf{T} F_{\mathrm{e},i}$ (Schiehlen and Eberhard, 2014, p. 100). In return, one obtains the EOM of the multibody system as

$$M \ddot{q} = Q, \tag{3.17}$$

with $Q = Q_\mathrm{C} + Q_\mathrm{e}$, the generalized inertia matrix $M = \sum_{i=1}^{N_\mathrm{b}} M_i$, the generalized bias force $Q_\mathrm{C} = \sum_{i=1}^{N_\mathrm{b}} Q_{\mathrm{C},i} = (Q_\mathrm{F} + Q_\mathrm{G})$ including the fictitious forces $Q_\mathrm{F} = \sum_{i=1}^{N_\mathrm{b}} J_i^\mathsf{T} \mathrm{vec}\{\mathbf{0}, -\tilde{\omega}\Theta_{S,i}\omega_i\} - \tilde{M}_i \tilde{J}_i)$ and the conservative forces $Q_\mathrm{G} = \sum_{i=1}^{N_\mathrm{b}} J_i^\mathsf{T} F_{\mathrm{G},i}$, as well as generalized impressed forces $Q_\mathrm{e} = \sum_{i=1}^{N_\mathrm{b}} Q_{\mathrm{e},i}$ (Schiehlen and Eberhard, 2014, p. 107). As the impressed forces

possess different properties depending on their source of origin, we further split up the impressed forces as

$$Q_{\mathrm{e}} = Q_{\mathrm{D}} + Q_{\mathrm{u}}, \tag{3.18}$$

with the dissipative forces $Q_{\mathrm{D}}(q, \dot{q}, t)$ and actuation forces $Q_{\mathrm{u}}(q, \dot{q}, t)$. These forces are discussed in further detail in Section 3.5.

Moreover, it is assumed that q is chosen such that $M(q,t)$ and its inverse $M^{-1}(q,t)$ are symmetric and positive-definite. Then, $M^{-1}(q,t)$ maps the n_q-dimensional generalised force space to the n_q-dimensional generalized acceleration space.

3.3 Euler-Lagrange Equations

In this section, we detail how some of the terms in (3.18) can be obtained in terms of potential functions. The resulting equations can be used to include energy conservation into an analytical structured model as discussed in Section 4.4.1. A detailed introduction to the Lagrange equations is given in (Layton, 2012, p. 68) and more specifically for robot arms in (Siciliano et al., 2010, p. 247).

The Lagrangian function $\mathcal{L}(q, \dot{q})$ can be obtained as the difference between the kinetic energy $T(q, \dot{q})$ and the potential energy $V(q, \dot{q})$, writing

$$\mathcal{L} = T - V = \frac{1}{2}\dot{q}^{\top} M \dot{q} - V. \tag{3.19}$$

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

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = 0. \tag{3.20}$$

Note that the Lagrangian differs from the system's total energy $E = T + V$. The Lagrangian is used to mathematically express that a conservative system must take a path of stationary action via (3.20) such that E remains constant. Rewriting (3.20) in vector notation and adding the non-conservative forces Q_{e} yields

$$\frac{d}{dt}\nabla_{\dot{q}}\mathcal{L} - \nabla_q \mathcal{L} = Q_{\mathrm{e}}, \tag{3.21}$$

with $(\nabla_{\dot{q}})_i = \frac{\partial}{\partial \dot{q}_i}$. One can apply the chain rule to expand the *time-derivative* of the Lagrangian's partial derivative as

$$\frac{d}{dt}\nabla_{\dot{q}}\mathcal{L} = \left(\nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L}\right)\ddot{q} + \left(\nabla_q \nabla_{\dot{q}}^{\top}\mathcal{L}\right)\dot{q}, \tag{3.22}$$

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 (3.17) in terms of the Lagrangian as

$$\ddot{q} = \left(\nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L}\right)^{-1}\left(-\left(\nabla_q \nabla_{\dot{q}}^{\top}\mathcal{L}\right)\dot{q} + \nabla_q \mathcal{L} + Q_{\mathrm{e}}\right). \tag{3.23}$$

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 $M = \nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L}$ to yield an alternative description of the forward dynamics as

$$\ddot{q} = M^{-1}\left(-\nabla_q(\dot{q}^{\top}M)\dot{q} + \frac{1}{2}\nabla_q\left(\dot{q}^T M \dot{q}\right) - \nabla_q V + Q_{\mathrm{e}}\right). \tag{3.24}$$

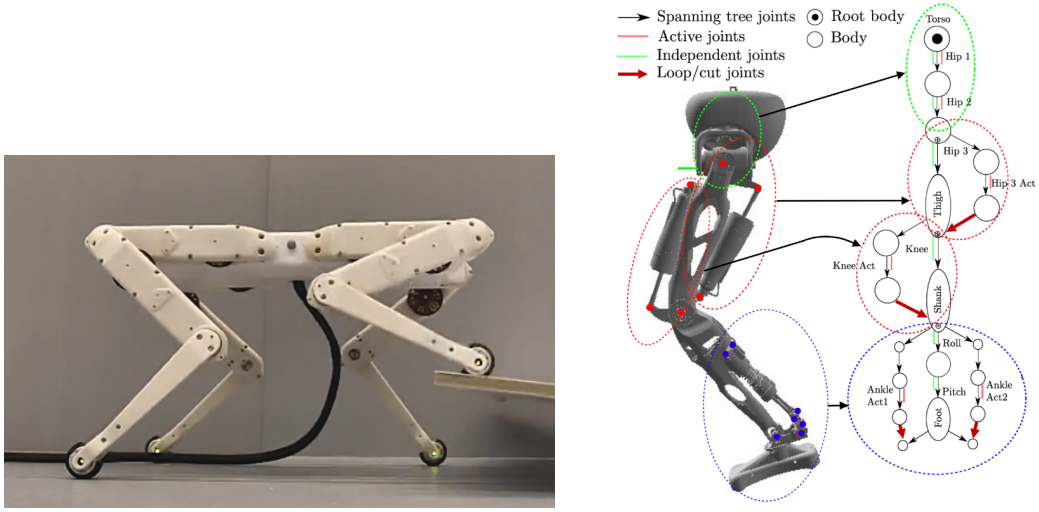


Figure 3.2: Left: The feet of a quadruped (Open Dynamic Robot) are subject to contact forces (Image used with courtesy of Grimminger et al. (2020)). Right: A pneumatic-actuated leg (RH5 leg) has several kinematic loops that can be modeled via implicit constraints which induce constraint forces at the cut joints (Robot image adapted with courtesy from Kumar (2019)). Figure adapted from *Geist and Trimpe (2021)*.

The above equations contain parametrizations of the fictitious force as

$$Q_{\mathrm{F}} = -\left(\nabla_q \nabla_{\dot{q}}^{\top} T\right) \dot{q} + \nabla_q T = -\nabla_q (\dot{q}^{\top} M)\dot{q} + \frac{1}{2}\nabla_q \left(\dot{q}^T M \dot{q}\right), \tag{3.25}$$

and of the conservative force

$$Q_{\mathrm{G}} = -\nabla_q V(q). \tag{3.26}$$

Remark 3.2 (Energy conservation in EOM formulations)**.** *The EOM derived via D'Alembert's principle as stated in* (3.17) *are identical to the EOM* (3.23) *that were derived via the principle of least action. In particular, the functions* $\{M, Q_C\}$ *forming the energy-conserving part of the EOM are the same. Both of these formalisms arrive at EOMs that in the absence of* Q_e *are conservative. The significant algorithmic difference is that the derivation of* (3.17) *starts with the formulation of forces, while* (3.23) *formulates* $\{M, Q_C\}$ *in terms of the Lagrangian* (3.19)*.*

3.4 Implicitly Constrained Dynamics

Section 3.2 outlined how constraint forces $F_{\mathrm{E},i}$ can be excluded from the EOM by a suitable choice of generalized coordinates q that yield explicit holonomic constraints. This section details how implicit constraints can be used to determine constraint forces inside the system's EOM.

Implicit constraints can be particularly useful for nonholonomic systems, systems with kinematic loops, and systems subject to inequality constraints. For example, when the foot of a quadruped presses onto a surface, the surface applies a reaction force. This force reduces the DOF of the system. The force arising from the contact can be modeled using

an implicit holonomic constraint. In return, one obtains an analytical expression for the respective constraint force, which can be straightforwardly removed from the EOM if the constraint is inactive. As another example, a hydraulic-actuated robot leg (Figure 2, right) introduces kinematic loops that can be modeled via the addition of implicit constraints.

We base the following discussion on the EOM as in (3.17). However, we broaden the problem setting compared to Section 3.2 by making the following assumptions:

- The system of rigid bodies is subject to constraint forces that now impose $n_\mathrm{E} + n_\mathrm{I}$ independent constraint equations with n_I denoting the number of implicit constraints. In return, the system has $(n_q - n_\mathrm{I})$ DOF (Layton, 2012, p. 43).
- The constraint forces that impose the n_E explicit holonomic constraints are eliminated from the EOM via a suitable choice of generalized coordinates $q \in \mathbb{R}^{n_q}$ as detailed in Section 3.2.

In turn, (3.17) becomes

$$M\ddot{q} = Q + Q_\mathrm{I} + Q_z, \tag{3.27}$$

where Q_I denotes implicit constrained forces whose value must be determined using $\{M, Q\}$ and the implicit constraint equations. The non-ideal constraint forces Q_z denote the non-ideal part of the implicit constraint forces and are further discussed in Section 3.5.3.

These assumptions cover by no means all possible descriptions of the EOM of a rigid body system. For example, Aghili (2005) details several dynamics equations of implicitly holonomic constrained systems which we do not cover in this work. Also, constraint equations can be redundant, which requires a more extensive treatment on the connection between forces and the vector spaces that constraint-related matrices span. A more general discussion is given in Featherstone (2008) as well as Koganti and Udwadia (2016). Nevertheless, the following discussion illustrates the interplay between many of the building blocks that are frequently encountered when deriving dynamics equations and which yield structure to a regression model as detailed in Section 5.

Implicit constraints can be expressed algebraically as $c(q,t) = 0$ if they are holonomic, or more generally as $c(q,\dot{q},t) = 0$ if they are nonholonomic. We assume that differentiation with respect to time yields implicit constraint equations on the system's position, velocity, and acceleration as detailed below in (3.28) and (3.29).

$$\begin{array}{rlll} & \text{position} & \text{velocity} & \text{acceleration} \\ \text{holonomic:} & c(q,t) = 0 \overset{\mathrm{d}/\mathrm{d}t}{\rightarrow} & \tilde{A}(q,t)\dot{q} = \tilde{b}(q,t) \overset{\mathrm{d}/\mathrm{d}t}{\rightarrow} & A(q,t)\ddot{q} = b(q,\dot{q},t) \end{array} \tag{3.28}$$

$$\begin{array}{rll} \text{nonholonomic:} & c(q,\dot{q},t) = 0 \overset{\mathrm{d}/\mathrm{d}t}{\rightarrow} & A(q,\dot{q},t)\ddot{q} = b(q,\dot{q},t) \end{array} \tag{3.29}$$

The terms such as $A(q,t) = \tilde{A}(q,t) = \frac{\partial c(q,t)}{\partial q}$ and $b(q,\dot{q},t) = \frac{\partial c(q,\dot{q},t)}{\partial t}$ denote partial derivatives with respect to q or t, respectively. Comparing (3.28) and (3.29), one sees that both holonomic and nonholonomic constraints can be denoted jointly on the acceleration level through the *constraining equation*

$$A\ddot{q} = b, \tag{3.30}$$

where we assume that only n_I constraints are expressed implicitly such that $A \in \mathbb{R}^{n_\mathrm{I} \times n_q}$ and $b \in \mathbb{R}^{n_\mathrm{I}}$. In what follows, the implicit constraints can be *rheonomic* that is explicitly

dependent on time or alternatively, *scleronomic* such that the constraints are not explicitly time-dependent.

Remark 3.3 (Minimal state description of nonholonomic systems)**.** *To obtain a minimal state description, a nonholonomic system requires more position variables than velocity variables. For example, a unicyclist riding on a plane can reach any potential position on the plane. Yet, as an ideally rolling wheel cannot slide sideways, the wheel's translational Cartesian velocities at the plane's contact point can be described as a single velocity variable pointing along a position-dependent axis. However, in this work, we limit the discussion to the description of the system's EOM using as many velocity variables $\dot{q}$ as position variables q. An introduction to the derivation of EOM of nonholonomic systems with a minimal state representation is given in (Featherstone, 2008, p. 41) and (Schiehlen and Eberhard, 2014, p. 114).*

3.4.1 Virtual Displacements

To be able to eliminate constraint forces using the D'Alembert principle for multibody systems (3.10), one must define what constitutes a virtual displacement vector δq. Oftentimes, the nonholonomic implicit constraints are assumed Pfaffian taking the form $\tilde{A}(q,t)\dot{q} = \tilde{b}(q,t)$ with $\tilde{A}(q,t)$ and $\tilde{b}(q,t)$ not being partial derivatives of a position-level constraint. In this case, for holonomic and Pfaffian nonholonomic constrained systems the virtual displacement is often defined as the infinitesimal vector δq that fulfills $\tilde{A}\delta q = 0$, cf. (Udwadia and Kalaba, 2007, p. 131) and (Layton, 2012, p. 50). However, this definition is not applicable for nonholonomic constraints of the form in (3.29). In this case, many works resort to virtual velocity vectors which denote the infinitesimal variation in the velocity that agrees with the constraints while not varying position and time (Schiehlen and Eberhard, 2014, p. 55). In return, one can turn to Jourdain's principle of virtual power to discuss the effect of nonholonomic constraint forces on the EOM. However, as our previous discussion of explicit constraints was centered around the concept of virtual work, we instead define virtual displacements in terms of acceleration variations as proposed by Udwadia et al. (1997), such that virtual displacement denotes any infinitesimal vector $\delta q \in \mathbb{R}^{n_q}$ fulfilling the equation

$$A\delta q = 0. \tag{3.31}$$

The proof for (3.31) was initially proposed by Udwadia et al. (1997) and is further discussed in (Udwadia and Kalaba, 2007, p. 223) and (Bauchau, 2011, p. 457). A brief outline of the proof of (3.31) is given in Section A.1.

Unlike Section 3.2, in which δq denoted any infinitesimal vector inside $\mathbb{R}^{n_q}$, (3.31) implies that in the presence of the additional implicit constraint forces, the virtual displacement denotes any infinitesimal vector that fulfills

$$\delta q \in \mathsf{N}(A), \tag{3.32}$$

such that $\delta q \in \mathbb{R}^{(n_q - n_\mathrm{I})}$. The above derivation of $\delta q := \delta\ddot{q}$ in terms of a variation of the system's accelerations occupies the same role as virtual velocities as discussed (Woernle, p. 202). (Udwadia and Kalaba, 2007, p. 223) as well as (Bauchau, 2011, p. 457) outline the relation of "Gauss principle" as also mentioned in (Schiehlen and Eberhard, 2014, p. 92) to the minimization of a quadratic function as discussed in Section 3.4.3.

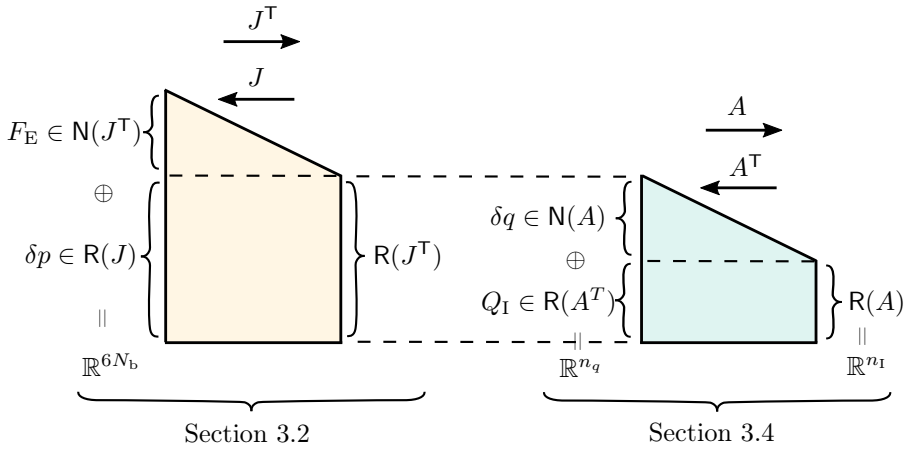


Figure 3.3: Under the assumption that the explicit and implicit constraints are independent, the constraint forces and virtual displacements must lie in the null and range spaces of J, J^T, A, and A^T as illustrated above. In the above diagram, being inspired by Beard (2002), vertical lines denote vector spaces. One can move a vector between two of these spaces by multiplication from the left with the constraint-related matrix transformation indicated on the respective arrow. Figure adapted from *Geist and Trimpe (2021)*.

3.4.2 D'Alembert's Principle with Implicit Constraints

The implicit constraints are the consequence of constraint forces and torques $F_{\mathrm{I},i} = \mathrm{vec}\{f_{\mathrm{I},i}, \tau_{\mathrm{I},i}\}$ acting onto each body. In turn, the D'Alembert's principle for the multi-body system (3.10) with the explicit transformation to generalized coordinates (3.12) as well as $\sum_{i=1}^{N_\mathrm{b}} J_i^\mathsf{T} F_{\mathrm{E},i} = 0$ due to the specific choice of q, reads

$$\sum_{i=1}^{N_\mathrm{b}} \delta p_i^\mathsf{T} (F_{\mathrm{E},i} + F_{\mathrm{I},i}) = \delta q^\mathsf{T} \sum_{i=1}^{N_\mathrm{b}} J_i^\mathsf{T} (F_{\mathrm{E},i} + F_{\mathrm{I},i}) = \delta q^\mathsf{T} Q_\mathrm{I} = 0, \tag{3.33}$$

with the generalized implicit constraint forces $Q_I = \sum_{i=1}^{N_\mathrm{b}} J_i^\mathsf{T} F_{\mathrm{I},i}$. As (3.31) requires $\delta q \in \mathsf{N}(A)$, (3.33) yields that

$$Q_\mathrm{I} \in \mathsf{R}(A^\mathsf{T}). \tag{3.34}$$

Vector spaces and constrained dynamics

The assumption that the explicit and implicit constraint equations are independent, requires (AJ^T) to have full row-rank at every non-zero state $\{q, \dot{q}, t\}$ that respects the constraints. Under these assumptions, the insights on the direction of F_E in (3.14), δp in (3.15), δq in (3.31), and Q_I in (3.34) can be summarized in a single diagram as depicted in Figure 3.3. Figure 3.3 emphasizes that at an admissible point in the system's state-space $\{q, \dot{q}, t\}$, the virtual displacement vectors as well as constraint forces are bound to lie in spaces spanned by the constraint-related matrices J and A.

3.4.3 Implicitly Constrained EOM

In textbooks on multibody mechanics, (3.34) often directly motivates a parametrization of the ideal constraint force in terms of a Lagrange multiplier vector $\lambda(q, \dot{q}, t) \in \mathbb{R}^{n_\mathrm{I}}$ as

$$Q_\mathrm{I} = A^\top \lambda. \tag{3.35}$$

Yet, as discussed in Section 2.3, at the gist of Lagrangian optimization lies an optimization problem that we seek to minimize through a corresponding Lagrangian function. Naturally, the question arises as to what may be the optimization function whose solution determines the length of Q_I?

To find one possible answer to this question, we may resort to Gauß (1829), who observed that for a system of point masses, the acceleration caused by the ideal constraint forces appears to minimize a quadratic functional. Consecutively, Udwadia and Kalaba (1992) extended this observation to multibody systems, such that the acceleration caused by implicit constraint forces, $\ddot{q}_\mathrm{I} = M^{-1}Q_\mathrm{I}$ follows from the following optimization problem

$$\begin{aligned} &\text{minimize} \;\; G(q, \dot{q}, t) = \ddot{q}_I^\top M \ddot{q}_I \\ &\text{subject to} \;\; A\ddot{q} = b. \end{aligned} \tag{3.36}$$

The above quadratic function, being referred to as *Gauss' principle of least constraint*, uniquely defines the length of the vector $\ddot{q}_\mathrm{I}$. The minimum-norm solution to (3.36) yields the system's EOM in terms of an MP inverse as initially shown for a system of mass particles in Udwadia and Kalaba (1992), later being extended to rigid body systems by Arabyan and Wu (1998), and discussed in ample detail in (Udwadia and Kalaba, 2007). Following (Udwadia and Kalaba, 2007, 2002), we insert (3.27) into (3.36) using $M > 0$, such that

$$\begin{aligned} &\text{minimize} \;\; G(q, \dot{q}, t) = \ddot{q}_{\mathrm{I},s}^\mathsf{T} \ddot{q}_{\mathrm{I},s} \\ &\text{subject to} \;\; A_s \ddot{q}_{\mathrm{I},s} = b - A_s M^{-1/2} \bar{Q}, \end{aligned} \tag{3.37}$$

with the *normalized acceleration* $\ddot{q}_{\mathrm{I},s} = M^{1/2}\ddot{q}_\mathrm{I}$, $A_s = AM^{-1/2}$, and $\bar{Q} = Q + Q_z$. As detailed in Section 2.2, the solution to (3.37) yields the equation

$$\ddot{q} = M^{-1/2} A_s^+ b + M^{-1/2}(I_n - A_s^+ A_s) M^{-1/2} \bar{Q}, , \tag{3.38}$$

$$= M^{-1} A^\mathsf{T} (AM^{-1}A^\mathsf{T})^+ b + (I - M^{-1}A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+ A) M^{-1} \bar{Q}, \tag{3.39}$$

$$= M^{-1}\Big(Q + Q_z + \underbrace{A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+\big(b - AM^{-1}(Q}_{Q_\mathrm{I}} + Q_z)\big)\Big), \tag{3.40}$$

Equation (3.38) is referred to in literature as the *Udwadia-Kalaba Equations of Motion* (UKE) (Zhao et al., 2018). Importantly, Q_z denotes the non-ideal part of the implicit constraint forces and by definition remains in $\mathsf{N}(A)$ (Udwadia and Kalaba, 2000, 2001, 2002). The UKE also yield a unique solution if $\mathrm{rank}(A) < n_\mathrm{I}$. Inside the UKE the forces $\bar{Q}$ are first normalized by $M^{-1/2}$ and then orthogonal projected through the matrix $(I - A_s^+ A_s)$ into $\mathsf{N}(A_s)$ (Koganti and Udwadia, 2016). Figure 3.4 visualizes the inner workings of the UKE as initially shown by Koganti and Udwadia (2016).

While the above equations are valid for $\mathrm{rank}(A) \leq n_\mathrm{I}$, if $\mathrm{rank}(A) = n_\mathrm{I}$ then $(AM^{-1}A^\mathsf{T})$ is nonsingular such that $(AM^{-1}A^\mathsf{T})^+ = (AM^{-1}A^\mathsf{T})^{-1}$.

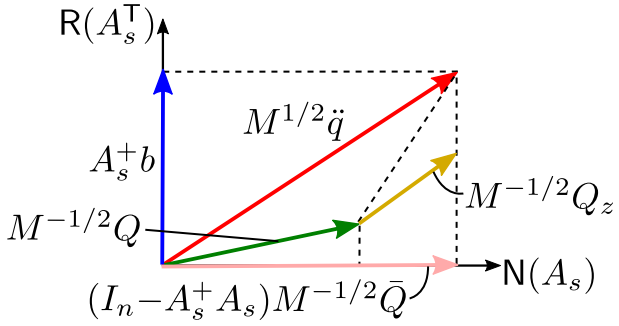


Figure 3.4: From the UKE follows that according to Gauss' principle the normalized acceleration of an implicitly-constrained system $M^{1/2}\ddot{q}$ consists of the vectors $(I_n - A_s^+ A_s)M^{-1/2}\bar{Q}$ and $A_s^+ b$. Adapted from Udwadia and Kalaba (2002).

As by definition $Q_z \in \mathsf{N}(A)$, an explicit formula for the Lagrange multipliers is obtained from (3.40) as

$$\lambda = (AM^{-1}A^\mathsf{T})^+\big(b - AM^{-1}Q\big). \tag{3.41}$$

We note that when the implicit constraints are active, the forces Q split up into $Q = Q_\mathsf{R} + Q_\mathsf{N}$ with $Q_\mathsf{R} \in \mathsf{R}(A^\mathsf{T})$ and $Q_\mathsf{N} \in \mathsf{N}(A)$. The force $Q_\mathsf{R} = -A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+AM^{-1}Q$ contributes to the implicit constraint force Q_I while $Q_\mathsf{N} = (I - A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+AM^{-1})Q$ contributes to the system's constrained acceleration. To simplify, the notation one can rewrite (3.39) as

$$\ddot{q} = Lb + TM^{-1}(Q + Q_z), \tag{3.42}$$
$$= Lb + M^{-1}S(Q + Q_z), \tag{3.43}$$

with the matrices $L = M^{-1}A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+$, $T = I - LA = I - M^{-1}A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+A$, and $S = T^\mathsf{T} = I - AL^\mathsf{T} = I - A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+AM^{-1}$. Moreover, assuming $M > 0$, T and S are oblique projection operators that map a vector into $\mathsf{N}(A)$ as detailed in Section A.2.

3.4.4 Implicit Constraints and Lagrangian Optimization

Further details on the UKE can be found in (Bauchau, 2011). A different approach that is commonly encountered in robotics resorts to Lagrangian optimization.

Gauss' principle as stated in (3.36) forms a special case of norm minimization with equality constraints as discussed in Section 2.3. Therefore, the multiplication of (3.36) by $\frac{1}{2}$ yields the Lagrangian

$$L = \frac{1}{2}\ddot{q}^\mathsf{T}M\ddot{q} - \bar{Q}^\mathsf{T}\ddot{q} + \frac{1}{2}\bar{Q}^\mathsf{T}M^{-1}\bar{Q} + \lambda^\mathsf{T}(A\ddot{q} - b). \tag{3.44}$$

To find the $\ddot{q}$ that minimizes (3.36), the KKT conditions of the Lagrangian read

$$\begin{bmatrix}\nabla_{\ddot{q}}L \\ \nabla_\lambda L\end{bmatrix} = \begin{bmatrix}M\ddot{q} - \bar{Q} + \lambda^\mathsf{T}A \\ A\ddot{q} - b\end{bmatrix} = 0, \tag{3.45}$$

which after rearranging yields an index-3 system of differential-algebraic equations, cf. (Schiehlen and Eberhard, 2014, p. 105), as

$$\begin{bmatrix}M & A^\mathsf{T} \\ A & \mathbf{0}\end{bmatrix}\begin{bmatrix}\ddot{q} \\ \lambda\end{bmatrix} = \begin{bmatrix}\bar{Q} \\ b\end{bmatrix}. \tag{3.46}$$

When M is positive definite and $\text{rank}(A) = n_\text{I}$, the solution to (3.46) is given according to (2.70) as

$$\ddot{q} = M^{-1}A^\mathsf{T}(AM^{-1}A^\mathsf{T})^{-1}b + (I - M^{-1}A^\mathsf{T}(AM^{-1}A^\mathsf{T})^{-1}A)M^{-1}\bar{Q}, \tag{3.47}$$

with

$$\lambda = (AM^{-1}A^\mathsf{T})^{-1}(AM^{-1}Q - b). \tag{3.48}$$

Equation (3.47) is identical to the EOM as in (3.39). We note that the definition of the Lagrange multiplier in (3.44) resulting in (3.48) is sign opposite to the Lagrange multiplier in (3.41). In turn, (3.48) the implicit constraint force becomes $Q_\text{I} = -A^\mathsf{T}\lambda$.

3.4.5 Singular Inertia Matrices

In some special cases, it is advantageous to derive the implicitly constrained EOM starting from EOM as in (3.17) in which the inertia matrix M is positive semi-definite. In what follows, we refer to EOM (3.17) before additional implicit constraint forces are introduced as the *unconstrained EOM*.

Practical examples that demonstrate the utility of a singular inertia matrix include:

- Deriving unconstrained EOM via the Euler-Lagrange equations, in which for convenience the potential energy is derived in Cartesian coordinates while the kinetic energy is formulated in terms of generalized coordinates (Udwadia and Phohomsiri, 2006, p. 2106), (Udwadia and Wanichanon, 2012, p. 333).
- Deriving unconstrained EOM using a redundant set of coordinates such that bodies can be removed from the system without significantly altering the EOM (Udwadia and Phohomsiri, 2006, p. 2109), (Udwadia and Wanichanon, 2012, p. 335)
- Deriving the rotational dynamics of a rigid body in terms of quaternions (Udwadia and Schutte, 2010, p. 123), (Udwadia and Wanichanon, 2013, p. 438).

One possible approach towards the derivation of EOM of a system with a singular inertia matrix has been proposed by Udwadia and Wanichanon (2013). First, the authors assume that an unconstrained system of the form in (3.17) with $M > 0$ is subject to implicit constraints that yield the constraining equation (3.30). As a next step, the authors define the *augmented mass matrix*

$$\bar{M} := M + \alpha^2 A^\mathsf{T} G A, \tag{3.49}$$

where $G(q,t) \in \mathbb{R}^{n_\text{I} \times n_\text{I}}$ denotes any positive definite matrix with sufficiently smooth functions ($\mathcal{C}^2$) as elements and $\alpha(t)$ any nowhere-zero real function that is also sufficiently smooth ($\mathcal{C}^2$). Then $\bar{M}$ is positive definite if and only if $\text{rank}([M, A^\mathsf{T}]) = n_q$ at each instant of time (Udwadia and Wanichanon, 2013, Lemma 3.1). Moreover, the authors define the augmented force as

$$\bar{Q}(q, \dot{q}, t) := Q(q, \dot{q}, t) + A^\mathsf{T}(q, \dot{q}, t)G(q, t)z(q, \dot{q}, t), \tag{3.50}$$

with z denoting an arbitrary vector. Subsequently, the EOM of the implicitly constrained system are obtained by replacing inside (3.38) the terms M by $\bar{M}$ and Q by $\bar{Q}$, reading

$$\ddot{q} = \bar{M}^{-1/2}(A\bar{M}^{-1/2})^{+}b + \bar{M}^{-1/2}(I - (A\bar{M}^{-1/2})^{+}(A\bar{M}^{-1/2}))\bar{M}^{-1/2}(\bar{Q} + Q_z)), \tag{3.51}$$

$$= \bar{M}^{-1}A^\mathsf{T}(A\bar{M}^{-1}A^\mathsf{T})^{-1}b + (I - \bar{M}^{-1}A^\mathsf{T}(A\bar{M}^{-1}A^\mathsf{T})^{-1}A)\bar{M}^{-1}(\bar{Q} + Q_z)). \tag{3.52}$$

Remarkably, besides $M > 0$, the constraints can be ideal or non-ideal, holonomic or nonholonomic, scleronomic or rheonomic, and dependent on each other.

Udwadia and Wanichanon (2013) points out (p. 441) that their EOM are simpler as the EOM proposed in Udwadia and Schutte (2010) as, *(i)* they only use A^T instead of A^+ to define the unconstrained auxiliary system and *(ii)*, they define the unconstrained auxiliary system in terms of the arbitrary positive function $\alpha(t)$ and positive definite matrix $G(q,t)$.

Deriving EOM with $M \geq 0$ via Lagrangian Optimization Alternatively to the derivation proposed by Udwadia and Wanichanon (2013), one arrives at a similar EOM via the inversion of the KKT matrix in (3.46). The KKT matrix in (3.46) is also referred to as the *bordered Gramian matrix* in optimization theory (Magnus and Neudecker, 2019, p. 66). As pointed out in (Magnus and Neudecker, 2019, p. 69, Theorem 23), the matrix equation (3.46) with $M \geq 0$ and $A \in \mathbb{R}^{n_\mathrm{I} \times n_q}$ has a solution if and only if

$$(Q + Q_z) \in \mathsf{R}([M, A^\mathsf{T}]) \;\text{ and }\; b \in \mathsf{R}(A), \tag{3.53}$$

which reads

$$\ddot{q} = \tilde{M}^+ A^\mathsf{T} (A\tilde{M}^+ A^\mathsf{T})^+ b + (I - \tilde{M}^+ A^\mathsf{T} (A\tilde{M}^+ A^\mathsf{T})^+ A)\tilde{M}^+ (Q + Q_z) + (I - \tilde{M}^+ \tilde{M})h, \tag{3.54}$$

with $\tilde{M} = M + A^\mathsf{T} A$ and the arbitrary vector h. If we further assume $\mathrm{rank}([M, A^\mathsf{T}]) = n_q$ such that $\tilde{M} > 0$ (Udwadia and Wanichanon, 2013, Lemma 3.1), we arrive at a special case ($\alpha = 1$, $G = I$) of (3.51).

3.5 Types of Forces

The previous discussion showed how constraint forces restrict the direction of a system's acceleration $\ddot{q}$ to follow a constraint path. Here, the size of the constrained acceleration, and hence the system's motion, is determined by the vector of forces $Q + Q_z$. These forces are caused by numerous physical phenomena such as tribologic effects, the interaction of mass with force, or magnetic fields.

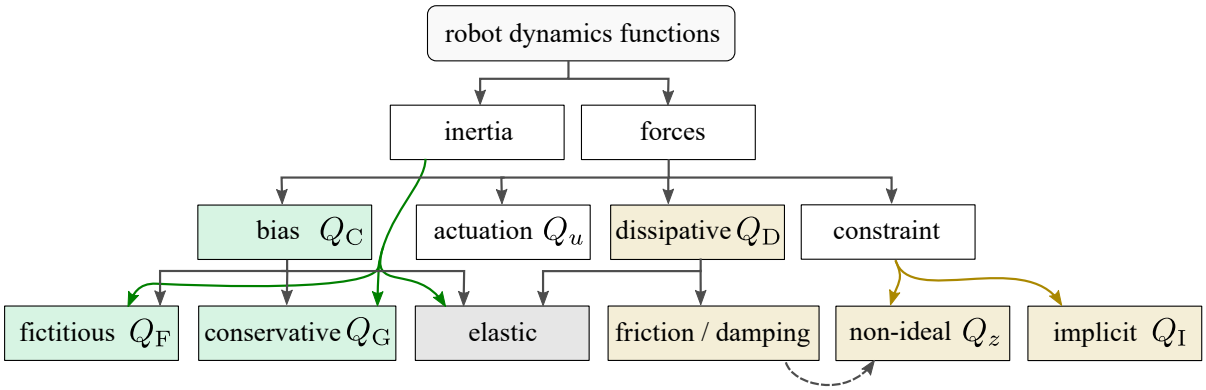


Figure 3.5: Overview on functions that arise in a robot's EOM. These functions differ in the type of system knowledge used for their derivation. Some functions depend on additional inertia terms (green), and others rely on external phenomena that are often not known a-priori (yellow).

Knowledge of the physical phenomena underlying forces enables informed decisions on how to learn these functions from data. In particular, with prior knowledge of the cause of a function, the inputs of a data-driven model approximating these functions may be chosen only to include relevant state information and informative environmental variables. In what follows, we briefly discuss the different types of forces that one may encounter in a robotic system. Figure 3.5 provides an overview of the different forces that one may encounter inside a rigid body system.

3.5.1 Conservative Forces

The bias force Q_{C} contains the conservative forces Q_{G} and the fictitious force Q_{F}. The term "bias force" has been adapted from (Featherstone, 2008, p. 40). The bias force includes all forces that only depend on the system's kinematics and inertia functions as described in Section 3.3. The fictitious force Q_{F} models the Coriolis and centrifugal forces. The conservative forces Q_{G} arise from a physical potential function $V(q, \dot{q})$ that may describe the system's potential energy when being subject to a spring or a gravitational field. The bias force does not include friction forces, even though some particular friction forces may be written in terms of a potential function (Kuypers, 2016, Chapter 6).

3.5.2 Dissipative Forces

The vector of dissipative forces $Q_{\mathrm{D}}(q, \dot{q}, t)$, denote forces that act against the system's motion and, in turn, reduce the system's kinetic energy. Dissipative forces include friction forces between bodies as well as dissipative phenomena during the deformation of bodies. Energy dissipation through body deformation is not discussed in this dissertation.

Direction of friction forces

A friction force that acts on a rigid body's particle being in contact with a surface opposes the particle's velocity relative to the surface. This directional principle of friction forces is useful structural knowledge. In practice, a friction force is a mathematical abstraction that describes the surface integral over infinitesimal forces acting inside a body's area of contact with another surface. In turn, friction effects in a revolute joint are often described via a torque vector. In robotic systems, the angle of revolute joints is often chosen as a generalized coordinate q_i such that the friction force inside a revolute joint is colinear to q_i. In particular, the friction force inside a revolute joint being colinear to q_i always opposes the velocity $\dot{q}_i$ and in turn may be modeled as

$$Q_{\mathrm{D},i}(q, \dot{q}, t, \rho) = -\mathrm{sign}(\dot{q}_i) \| f_{\mathrm{D}}(q, \dot{q}, t, \rho) \|_1, \tag{3.55}$$

with f being a parametric function of appropriate size and ρ denoting additional variables, cf. (Bauchau, 2011, p. 88) and (Lutter et al., 2020). At this point, the function f_{D} uses full state information as input as no additional structural knowledge has been assumed. Yet, friction force may depend on a Cartesian force that is applied normal to the contact area. Modeling friction is a challenging endeavor in itself as it depends on various parameters and may change over time while depending on many environmental parameters. Many empirical models such as the Coulomb friction model, the viscous friction model, the Dahl model, or the Lugre friction model have been proposed to describe friction phenomena.

3.5.3 Non-ideal Constraint Forces

Oftentimes, constraint forces are not ideal and do virtual work. In such a case, Udwadia and Kalaba (2002) proposed to divide the constraint force into an ideal part Q_{I} and a non-ideal part Q_z doing virtual work. The forces Q_z are usually dissipative, but can also denote actuation forces. Typically, Q_{I} causes a Cartesian Coulomb friction force F_z whose transformation into the generalized coordinate space yields Q_z. For example, the friction force Q_z between a robot's foot and a surface is caused by the normal force F_{I} that the foot applies onto the surface. As emphasized by Udwadia and Kalaba (2000), the part of any arbitrary force which is doing virtual work, Q', writing $\delta q^{\mathsf{T}} Q' \neq 0$, must have the same direction as δq and hence

$$Q' \in \mathsf{N}(A). \tag{3.56}$$

3.5.4 Actuation Forces

The actuation forces $Q_{\mathrm{u}}(q, \dot{q}, t)$, also referred to as control forces, can decrease as well as increase the system's total energy. In addition, the actuation forces can be directly controlled during the robot's operation. In many robots the actuation forces are linear with respect to the control inputs u, writing

$$Q_{\mathrm{u}} = B(q)u, \tag{3.57}$$

with the input Jacobian $B(q)$.

Actuation Dynamics In the control of a mechanical system, one usually calculates the desired actuation force $Q_{\mathrm{u,desired}}$ that is sent to computational routines, which in return cause the actuators to impress Q_{u} onto the system. Yet, Q_u likely deviates from $Q_{\mathrm{u,desired}}$ due to friction and other unmodeled physical phenomena inside the actuators, flexibilities in gear-belts or attached shafts, free-play in an attached transmission, and internal dynamics of cascaded feedback control loops. To account for this deviation, one may introduce the actuation error function

$$\epsilon_u(\rho) = Q_{\mathrm{u,desired}} - Q_{\mathrm{u}}, \tag{3.58}$$

with a suitably chosen vector of observables ρ, *e.g.*, state data $\{q, \dot{q}\}$ from several time steps, the error to a target state, or the current applied to the actuators. Note some parts of $\epsilon_u(\rho)$ can be modeled via the dissipative forces Q_{D}.

Example 3.4 (Creation of an actuation force inside a robot's joint)**.** *The generation of torque inside a robot arm's brushless motors requires that $Q_{u,desired}$ is transformed by low-level control routines into motor currents. The motor currents create a magnetic field that creates torque inside the motor. This motor torque is altered by a potential gear train as well as friction, and finally Q_u is applied to the joints. Here, the motor shaft can be subject to torsional deformation, and a gear belt may strain until the static friction inside the robot's joint is overcome.*

Joint Torques Currently, the most common robotic systems, such as many robot arms and quadrupedal robots, connect bodies through revolute joints in which motors can cause actuation forces inside the joint. In these systems, the generalized force being caused inside

the joint is referred to as *joint torque* τ. As the dissipative forces act all inside the joints, the joint torques for a robot arm's EOM described in minimal coordinates become

$$\tau := Q_{\mathrm{e}} = Q_{\mathrm{D}} + Q_{\mathrm{u}}. \tag{3.59}$$

The EOM of a robot arm whose end-effector is subject to a surface constraint may be written as

$$\tau = M\ddot{q} - \left(Q_{\mathrm{C}} + Q_{\mathrm{I}} + Q_z\right). \tag{3.60}$$

That is, the joint torque can be observed as the difference between the forces acting onto the system $\{Q_{\mathrm{C}} + Q_{\mathrm{I}} + Q_z\}$ and the inertial force $\{M\ddot{q}\}$.

Computing Control as an Implicit Constraint Force As initially proposed by Udwadia (2003), actuation forces may also be computed as an implicit constraint force required to enforce that a mechanical system respects a set of implicit constraints. Peters et al. (2008) used this idea to develop a framework that derives several well-known robot control laws using implicitly-constrained rigid body dynamics equations. Moreover, Udwadia and Koganti (2015) detail how some Lyapunov functions can be written to yield acceleration-level constraints as in (3.30). Koganti and Udwadia (2016) further details how implicitly-constrained forces arising from mechanical constraints can be combined with implicit constraint forces arising from control constraints.

Chapter 4

Physics Informed Regression

Analytical mechanics comprises an abundant source of structural knowledge for the regression of robot dynamics. Yet, if a practitioner is tasked to model a robot's dynamics function, questions emerge on how to leverage this structural knowledge for the identification of the error functions. To alleviate these questions, we look at rigid body dynamics as a parametric network that similarly to other parametric models is erroneous. This view on analytical models allows us to distinguish errors into errors that arise from a faulty specification of the dynamics and errors that arise from wrong analytical parameter estimates. This separation of a dynamic model's errors is done in the output space of the analytical model. We further extend this discussion by assuming that errors may arise in latent functions hidden inside the analytical model. This enables us to categorize current literature on physics-informed regression with respect to how the models' errors are reduced, namely into:

- **Analytical parametric networks** (APN): An analytical dynamics model is treated as a parametric model whose parameters are estimated from data.
- **Analytical (output) residual modeling** (ARM): The error between the system's data and the output of an analytical dynamics model is approximated by a solely data-driven model.
- **Analytical latent modeling** (ALM): A data-driven model replaces functions inside an analytical mechanics model to approximate latent errors in the mechanics before they are transformed by a-priori known transformations from physics.

This categorization of different physics-informed models allows us to develop a deeper understanding of how errors arise inside a rigid body dynamics model.

> *Subsequently, we suggest that a physics model's errors should be approximated by data-driven models before these latent functions are transformed by known transformations from physics. In this regard, the data-driven model may only enlarge the function space of the physics-informed model where necessary.*

The first paragraphs of this section evolving around the errors of physics-informed models is based on *(Geist and Trimpe, 2021)*[1]. Albeit, this discussion has been re-written and the notion of adventitious errors has been introduced. The discussion of existing literature in physics-informed modeling is mostly identical to *(Geist and Trimpe, 2021)* and *(Rath et al., 2021)*.

[1] Wiley remains the copyrights holder for all text excerpts and figures which have been taken from *Geist and Trimpe (2021)*.

4.1 Model Errors

As shown in Section 3, the FD can be expressed either in terms of force vectors that are derived using explicit coordinate transformations (3.17), additional potential functions (3.21), or resorting to implicit constraints (3.40), taking the form

$$\ddot{q} = M^{-1}(Q + Q_z + Q_I). \tag{4.1}$$

These formulations of the FD consist of sums of generalized forces that are multiplied by M^{-1}. In comparison, the system's ID may take the form

$$Q_u = M\ddot{q} - Q_{\mathrm{C}} - Q_{\mathrm{D}} - Q_z - Q_I, \tag{4.2}$$

in which the acceleration $\ddot{q}$ is transformed by M and the force vector by identity matrices.

To unify the discussion of FD and ID, it is assumed that the EOM may 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 $\mathcal{C}_i(x, \theta_{\mathrm{A}})$, *e.g.*, $\hat{f}_1 := Q_C$ and $\mathcal{C}_1 := M^{-1}$, such that the system's dynamics model read

$$\hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}}) = \sum_i \mathcal{C}_i \hat{f}_i, \tag{4.3}$$

where x denotes the model's input vector.

4.1.1 Approximation errors

Naturally, a dynamics model deviates from the system's real dynamics $f(x)$ by an error function

$$\epsilon_{\mathrm{A}}(x, \theta_{\mathrm{A}}) = f(x) - \hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}}). \tag{4.4}$$

A common approach towards the reduction of ϵ_{A} is to optimize the dynamics model parameters θ_{A} such that a loss $L(\hat{f}, \mathcal{D}; \theta_{\mathrm{A}}, \theta_L)$ over the training data X is minimized. Literature that follows this approach is discussed in Section 4.2. The intrinsic goal of such an optimization is to find *ideal dynamic model parameters*

$$\theta_{\mathrm{A}}^{\star} = \arg\min_{\theta_{\mathrm{A}}} \|\epsilon_{\mathrm{A}}(x, \theta_{\mathrm{A}})\|_1. \tag{4.5}$$

such that one obtains the *ideal dynamics model*

$$\hat{f}_{\mathrm{A}}^{\star}(x) = f_{\mathrm{A}}(x, \theta_{\mathrm{A}}^{\star}) = \sum_i \mathcal{C}_i^{\star}(x) \hat{f}_i^{\star}(x) = \sum_i \mathcal{C}_i(x, \theta_{\mathrm{A}}^{\star}) \hat{f}_i(x, \theta_{\mathrm{A}}^{\star}). \tag{4.6}$$

In turn, the vector-valued minimal error function of the dynamics model being referred to as *approximation error* in analogy to statistical learning theory, reads

$$\epsilon_{\mathrm{A}}^{\star}(x) = \epsilon_{\mathrm{A}}(x, \theta_{\mathrm{A}}^{\star}) = f(x) - \hat{f}_{\mathrm{A}}^{\star}(x). \tag{4.7}$$

By definition, the function $\epsilon_{\mathrm{A}}^{\star}$ does not depend on θ_{A} and denotes errors that cannot be captured by the dynamics model. For example, $\epsilon_{\mathrm{A}}^{\star}$ may be caused by elasticities in the system's bodies, disturbances impressed by attached cables, unaccounted torsion of a shaft inside a gearbox, or an insufficient analytical description of friction phenomena.

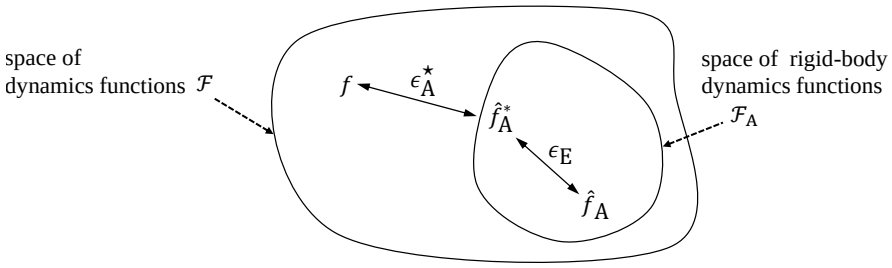


Figure 4.1: Illustration of the errors of a rigid body dynamics model.
Figure adapted from *Geist and Trimpe (2021)*.

As shown in Figure 4.1, the error functions underlying a dynamics model can be conceptually illustrated as being elements of a function space. Figure 4.1 is inspired by the survey of Von Luxburg and Schölkopf (2011) on statistical learning theory, which shares interesting similarities to the problem of nonlinear dynamics identification. Notably, every parameter vector θ_{A} defines with the dynamics model $\hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}})$ an element of a function space $\mathcal{F}_{\mathrm{A}} = \{\hat{f}_{\mathrm{A}} : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}\}$. The function space $\mathcal{F}_{\mathrm{A}}$ forms a subspace of the space of all possible dynamics models $\mathcal{F} = \{f' : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}\}$, writing $\mathcal{F}_{\mathrm{A}} \subset \mathcal{F}$. When a certain coordinate representation $\{x, y\}$ is chosen to model dynamics, it is implicitly assumed that there exists the real dynamics $y = f(x)$ such that $f \in \mathcal{F}$. In this case, changing θ_{A} can be seen as moving $\hat{f}_{\mathrm{A}}$ around in $\mathcal{F}_{\mathrm{A}}$ where $\hat{f}_{\mathrm{A}}^{\star}(x)$ denotes the element that is closest to $f(x)$ in terms of $\|\epsilon_{\mathrm{A}}\|_1$.

Physics-informed modeling approaches aim at reducing the approximation error of a physics model by enlarging the function space through the thoughtful incorporation of a data-driven model.

4.1.2 Estimation Errors

While $\epsilon_{\mathrm{A}}^{\star}$ does not depend on θ_{A}, optimization of θ_{A} seeks to minimize the *estimation error*

$$\epsilon_{\mathrm{E}}(x, \theta_{\mathrm{A}}) = \epsilon_{\mathrm{A}}(x, \theta_{\mathrm{A}}) - \epsilon_{\mathrm{A}}^{\star}(x) = \hat{f}_{\mathrm{A}}^{\star}(x) - \hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}}). \tag{4.8}$$

In practice, it is not feasible to observe all values of $\epsilon_{\mathrm{A}}(x, \theta_{\mathrm{A}})$. Instead, parameter identification of $\hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}})$ is carried out on a dataset $\mathcal{D} = \{X, Y\}$. The evaluation of $\epsilon_{\mathrm{A}}(x, \theta_{\mathrm{A}})$ on the input data X is denoted in an abuse of notation by the vector $\epsilon_{\mathrm{A}}(X, \theta_{\mathrm{A}})$, which with the output data Y yields the *model's output residual*

$$\tilde{\epsilon}_{\mathrm{A}}(\mathcal{D}, \theta_{\mathrm{A}}) = Y - \hat{f}_{\mathrm{A}}(X, \theta_{\mathrm{A}}). \tag{4.9}$$

By definition, the model parameter vector $\hat{\theta}_{\mathrm{A}}$ that achieves the minimal estimation error $\epsilon_{\mathrm{E}}(x, \hat{\theta}_{\mathrm{A}})$ is obtained as

$$\hat{\theta}_{\mathrm{A}} = \arg\min_{\theta_{\mathrm{A}}} L(\hat{f}_{\mathrm{A}}, \mathcal{D}; \theta_{\mathrm{A}}). \tag{4.10}$$

The limited amount and quality of training data hinder the identification of θ_{A}. For example, an optimizer is biased by the distribution of data in X. However, it is possible to adjust X to remove biases as well as shape the loss L such that data points with a large error considerably affect the estimate of θ_{A}, *e.g.*, using a quadratic loss. In addition to problems with data accumulation, the optimization of a nonlinear function can itself be a laborious

process. On one hand, the global optimization methods require large amounts of samples to cover significant parts of the optimization space whereas local optimization techniques are prone to get stuck in local optima. The limitations of parameter estimation causes θ_{A} to deviate from $\theta_{\mathrm{A}}^{\star}$ by a vector-valued *parameter error* function

$$\epsilon_{\theta_{\mathrm{A}}} = \theta_{\mathrm{A}}^{\star} - \theta_{\mathrm{A}}. \tag{4.11}$$

With the above equation, the estimation error can be rewritten in terms of its defining quantity $\epsilon_{\theta_{\mathrm{A}}}$ as

$$\epsilon_{\mathrm{E}}(x, \theta_{\mathrm{A}}) = \hat{f}_{\mathrm{A}}(x; \theta_{\mathrm{A}}^{\star}) - \hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}}^{\star} - \epsilon_{\theta_{\mathrm{A}}}). \tag{4.12}$$

The decomposition of the error function into approximation errors $\epsilon_{\mathrm{A}}^{\star}(x)$ and estimation errors $\epsilon_{\mathrm{A}}(x, \theta_{\mathrm{A}})$ reveals a dilemma in the design and optimization of a statistical model. On one hand, adding a data-driven model to an analytical dynamics model enlarges its function space and in turn, reduces $\epsilon_{\mathrm{A}}^{\star}(x)$. On the other hand, it is well known from statistics literature that a large function class increases *over-fitting* of the training data and subsequently increases $\epsilon_{\mathrm{E}}(x, \theta_{\mathrm{A}})$ (Von Luxburg and Schölkopf, 2011). If a model over-fits, then its ability to generalize to unseen parts of the state space diminishes. Moreover, increasing the model's number of parameters aggravates parameter optimization, increases the amount of data as well as time required to achieve a satisfactory optimization loss, increases the model's energy consumption, and increases the model's memory requirements. To make matters worse, if the model spans a too-small function space, such that it *under-fits*, then the overall error $\epsilon_{\mathrm{A}}(x, \theta_{\mathrm{A}})$ increases significantly while the physical parameters may attain physically inconsistent values. Physical inconsistent values form a particular challenge in the parameter optimization of analytical dynamics models and are discussed in more detail in Section 4.2.

4.1.3 Latent and Adventitious Errors

The significant difference between solely data-driven regression and physics-informed regression informs the inclusion of additional prior physics knowledge. So far, it has been assumed that structural knowledge takes the form of an analytical dynamics model $\hat{f}_{\mathrm{A}}(X, \theta_{\mathrm{A}})$ that is used in the computation of the residual function (4.9). However, the individual functions that form an analytical dynamics model $\{\mathcal{C}_i, \hat{f}_i\}$ possess unique mathematical properties while also our knowledge of these latent analytical dynamics functions differs considerably. It appears worthwhile to inspect the errors that arise in $\{\mathcal{C}_i, \hat{f}_i\}$ and how subsequently these latent errors shape ϵ_{A}. Following this vein of thought, we assume that the real dynamics function can be denoted in terms of the latent error functions $\{\epsilon_{\mathcal{C}_i}, \epsilon_{\hat{f},i}\}$ as

$$f(x) = \epsilon_{\mathrm{A,A}}^{\star} + \sum_i \big(\mathcal{C}_i + \epsilon_{\mathcal{C}_i}\big)\big(\hat{f}_i + \epsilon_{\hat{f},i}\big), \tag{4.13}$$

where the *adventitious* approximation error $\epsilon_{\mathrm{A,A}}^{\star}$ denotes error functions that cannot be captured by a parametric model as in (4.3) regardless of the model's chosen function class. For example, the elastic deformation of bodies cannot be captured by rigid body dynamics and in turn, introduces an error $\epsilon_{\mathrm{A,A}}^{\star}$ into a data-driven rigid body dynamics model. Another example for the creation of $\epsilon_{\mathrm{A,A}}^{\star}$ forms a cable that is attached to a robot's body and whose influence on the robot's dynamics cannot be modeled given the system's observable states.

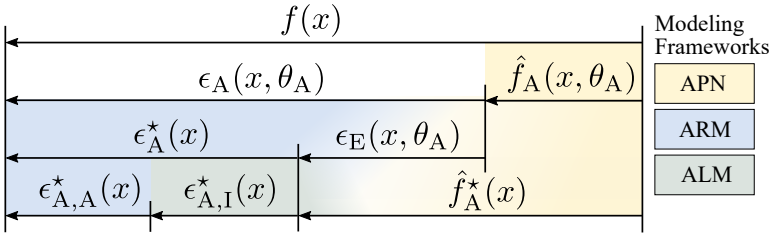


Figure 4.2: The additive relationship between the functions underlying a data-driven analytical dynamics model is visualized as a 1D vector diagram. APN solely focuses on the estimation of θ_{A} inside $\hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}})$. ARM additionally models $\epsilon^{\star}_{\mathrm{A}}(x)$ by a data-driven model, which also may reduce $\epsilon_{\mathrm{E}}(x, \theta_{\mathrm{A}})$. ALM uses specifically designed data-driven models to reduce approximation errors $\epsilon^{\star}_{\mathrm{A,I}}(x)$ that originate from the miss-specification of latent functions inside $\hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}})$.

In analogy to the previous sections, the latent errors can be divided into latent approximation errors $\{\epsilon_{\mathcal{C}_i,\mathrm{I}}, \epsilon_{\hat{f}_i,\mathrm{I}}\}$ and latent estimation errors $\{\epsilon_{\mathcal{C}_i,\mathrm{E}}, \epsilon_{\hat{f}_i,\mathrm{E}}\}$, writing

$$\epsilon_{\mathcal{C}_i}(x, \theta_{\mathrm{A}}) = \epsilon_{\mathcal{C}_i,\mathrm{I}}(x) + \epsilon_{\mathcal{C}_i,\mathrm{E}}(x, \theta_{\mathrm{A}}) \;\text{ and }\; \epsilon_{\hat{f}_i} = \epsilon_{\hat{f}_i,\mathrm{I}}(x) + \epsilon_{\hat{f}_i,\mathrm{E}}(x, \theta_{\mathrm{A}}). \tag{4.14}$$

Latent Estimation Errors

The latent functions $\{\mathcal{C}_i, \hat{f}_i\}$ give rise to estimation errors

$$\epsilon_{\mathcal{C}_i,\mathrm{E}}(x, \theta_{\mathrm{A}}) = \mathcal{C}^{\star}_i(x) - \mathcal{C}_i(x, \theta_{\mathrm{A}}), \;\text{ and }\; \epsilon_{\hat{f}_i,\mathrm{E}}(x, \theta_{\mathrm{A}}) = \hat{f}^{\star}_i(x) - \hat{f}_i(x, \theta_{\mathrm{A}}). \tag{4.15}$$

In analogy to Section 4.1.2, the latent estimation errors $\{\epsilon_{\mathcal{C}_i,\mathrm{E}}(x, \theta_{\mathrm{A}}), \epsilon_{\hat{f}_i,\mathrm{E}}(x, \theta_{\mathrm{A}})\}$ solely arise from an error in the model's analytical parameters $\epsilon_{\theta_{\mathrm{A}}} = \theta_{\mathrm{A}} - \theta^{\star}_{\mathrm{A}}$.

When designing a physics-informed regression model, the individual parameter errors $\epsilon_{\theta_{\mathrm{A}},i}$ differently affect the dynamics model's prediction. While a precise estimation of some parameters in θ_{A} strongly influences the model's prediction error ϵ_{A}, a large error in other parameters may not significantly impact the model's prediction. For example, for robots that have a gearbox inside their joints, some rotational inertias may have a negligible effect on the system's dynamics compared to the gear friction torques. To avoid estimating model parameters that barely affect the loss function, one can perform a parameter sensitivity analysis. If all analytical parameters are treated as unknown and must be estimated from data, the function space spanned by a parametric model increases and in turn, its sample efficiency reduces.

Latent Approximation Errors

In addition, the model's approximation error $\epsilon^{\star}_{\mathrm{A}}$ can be further divided into

$$\epsilon^{\star}_{\mathrm{A}} = \epsilon^{\star}_{\mathrm{A,A}} + \epsilon^{\star}_{\mathrm{A,I}}, \tag{4.16}$$

with the *intrinsic* approximation error $\epsilon^{\star}_{\mathrm{A,I}}$ denoting all errors that arise due to a miss-specification of the latent functions $\{\mathcal{C}_i, \hat{f}_i\}$. For example, due to an inaccurate friction model $\hat{f}_i$ an error $\epsilon_{\hat{f}_i,\mathrm{I}}(x)$ is introduced into the model that could be removed by adjusting the function class of $\hat{f}_i$. In particular, one could adjust the function class by the addition

of a neural network to the analytical dynamics function $\hat{f}_i$ approximating $\epsilon_{\hat{f}_i,\mathrm{I}}(x)$. By definition, the intrinsic approximation error $\epsilon^{\star}_{\mathrm{A,I}}$ arises from latent approximation errors $\{\epsilon_{\hat{f}_i,\mathrm{I}}(x), \epsilon_{\mathcal{C}_i,\mathrm{I}}(x)\}$. To understand the relationship between $\epsilon^{\star}_{\mathrm{A,I}}$ and $\{\epsilon_{\hat{f}_i,\mathrm{I}}(x), \epsilon_{\mathcal{C}_i,\mathrm{I}}(x)\}$, we consult Figure 4.2 which yields with (4.15) the equation

$$\epsilon^{\star}_{\mathrm{A,I}}(x) + \hat{f}^{\star}_{\mathrm{A}}(x) = \sum_i \Big(\big(\mathcal{C}_i + \epsilon_{\mathcal{C}_i,\mathrm{E}} + \epsilon_{\mathcal{C}_i,\mathrm{I}}\big)\big(\hat{f}_i + \epsilon_{\hat{f}_i,\mathrm{E}} + \epsilon_{\hat{f}_i,\mathrm{I}}\big)\Big), \tag{4.17}$$

$$= \sum_i \Big(\big(\mathcal{C}_i - \mathcal{C}_i + \mathcal{C}^{\star}_i + \epsilon_{\mathcal{C}_i,\mathrm{I}}\big)\big(\hat{f}_i - \hat{f}_i + \hat{f}^{\star}_i + \epsilon_{\hat{f}_i,\mathrm{I}}\big)\Big), \tag{4.18}$$

such that with (4.6) one arrives at

$$\epsilon^{\star}_{\mathrm{A,I}}(x) = \sum_i \Big(\mathcal{C}^{\star}_i \epsilon_{\hat{f}_i,\mathrm{I}} + \epsilon_{\mathcal{C}_i,\mathrm{I}} \epsilon_{\hat{f}_i,\mathrm{I}} + \epsilon_{\mathcal{C}_i,\mathrm{I}} \hat{f}^{\star}_i \Big). \tag{4.19}$$

4.1.4 Types of Knowledge

The distinction between latent approximation errors and latent estimation errors allows us to categorize $\{\hat{f}_i, \mathcal{C}_i\}$ based on the degree to which these functions are known during model synthesis. Namely, a latent analytical function $\hat{f}_i$ (and analogously a function $\mathcal{C}_i$) may be:

- known, $\epsilon_{\hat{f}_i,\mathrm{I}} = \epsilon_{\hat{f}_i,\mathrm{E}} = 0$,
- parametrically-unknown, $\epsilon_{\hat{f}_i,\mathrm{I}} = 0$, $\epsilon_{\hat{f}_i,\mathrm{E}} \neq 0$,
- unknown, $\epsilon_{\hat{f}_i,\mathrm{I}} \neq 0$, $\epsilon_{\hat{f}_i,\mathrm{E}} \neq 0$.

Typically, known robot dynamics functions are kinematics and constraint equations. Kinematic parameters can be often accurately determined using computer-aided design (CAD), by measuring on the real robot, or by performing parametric estimation solely on the system's kinematics functions.

Inertia parameters can be determined using CAD. Yet, developing a CAD model with a correct mass distribution is a laborious process. In turn, many robot manufacturers do not provide accurate priors for the system's inertia parameters. Instead, a manufacturer may provide parameter estimates that have been obtained using basic optimization techniques such as linear regression as detailed in Section 4.2. Therefore, inertia parameters are often not precisely known a-priori. More complex functions such as friction may also be known a-priori, if they have been identified in a separate previous experiment. From a data-driven learning perspective, it is helpful to distinguish *kinematic parameters* $\theta_{\mathrm{A,K}} \subset \theta_{\mathrm{A}}$ from *inertia parameters* $\theta_{\mathrm{A,I}} \subset \theta_{\mathrm{A}}$.

Remark 4.1 (On the separation of kinematic and inertia parameters.)**.** *In some rigid body dynamics algorithms, the kinematics functions are recursively computed using vectors that point from a preceding joint to a body's COG, and another vector that points from the body's COG to the next joint. Such a parameterization of the system's kinematics does not allow the estimation of the system's COG positions independently from the distance between the robot's joints. In turn, in such a kinematics description the robot's COG positions cannot be estimated independently from the distance between the joints.*

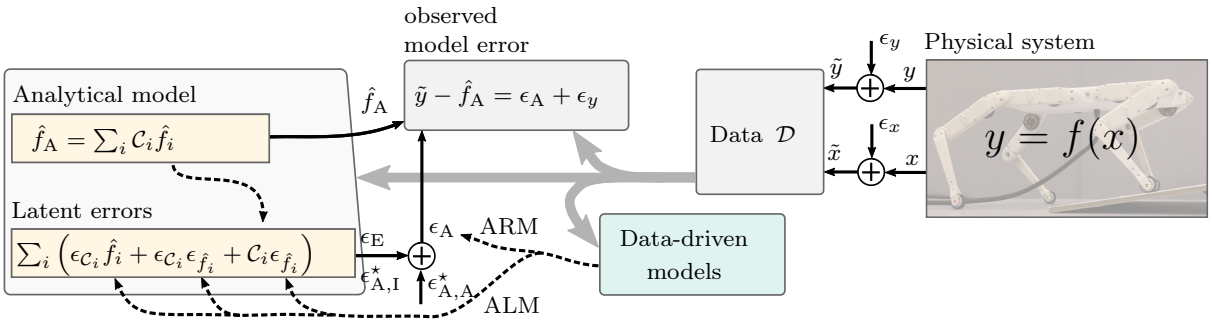


Figure 4.3: Schematic of analytical structured modeling. Data-driven models can approximate the output error of the analytical model (ARM) and/or reduce latent errors inside the analytical model (ALM). Robot image used with courtesy of Grimminger et al. (2020). Figure adapted from *Geist and Trimpe (2021)*.

Parametrically unknown functions in rigid systems are typical $\{M, Q_C\}$ as these functions are derived only in terms of known kinematics, as well as the gravitational acceleration constant. Unknown functions depend on a-priori unknown and complicated environmental phenomena. A notorious candidate for unknown functions form friction forces in a robot's joints as well as in a robot's end-effector. While a function may be unknown, we can still have an analytical model prior, such as a Coulomb friction model, to reduce $\{\epsilon_{\hat{f}_i,\mathrm{I}}, \epsilon_{\hat{f}_i,\mathrm{E}}\}$. Moreover, a function can be unknown, while the model designer has structural knowledge of the mathematical properties of the function. For example, one might wish to approximate an unknown friction function with a solely data-driven parametric function whose outputs can only be dissipative.

4.1.5 Modeling Frameworks

Prior works towards the identification of rigid body dynamics differ in the imposed assumptions on the system and its analytical description. In what follows, we propose a categorization of physics-informed regression models for rigid body dynamics identification and discuss how recently proposed models fit into these categories.

The first category of models that suggest improvements over solely analytical dynamics models are analytical dynamics models whose parameters are estimated from data. We refer to such approaches as "Analytical Parameter Optimization", which will be discussed in Section 4.2. Analytical parameter optimization cannot reduce the error ϵ_A that remains after parameter estimation.

As depicted in Figure 4.3, physics-informed regression models can be distinguished by how ϵ_A is approximated using data-driven modeling, namely into:

- ARM, in which data-driven models approximates ϵ_A directly,
- ALM of latent analytical functions, in which data-driven models approximate $\hat{f}_i$ and/or $\mathcal{C}_i$ in (4.13) if the respective analytical model is inaccurate, or alternatively,
- ALM of latent residual functions, in which data-driven models approximate $\epsilon_{\mathcal{C}_i}$ and/or $\epsilon_{\hat{f}_i}$ in (4.13) hence also using the analytical functions $\hat{f}_i$ and $\mathcal{C}_i$.

For both ARM and ALM, the *target function* which shall be approximated by the data-driven model must be specified. With the prior knowledge on where approximations errors occur inside the analytical dynamics model, ALM is potentially more sample-efficient than ARM as it allows to additionally:

- reduce the complexity of the target function,
- reduce the dimensionality of the target function,
- build prior knowledge on physical properties of the target function into the data-driven model.

For example, if we assume that it is known (through empirical validation) that a large part of the approximation errors arises from joint friction. Then a data-driven model can directly be built into the analytical dynamics model such that it approximates the joint friction. For the data-driven model, one can then only use inputs that are typically relevant for friction functions and enforce that its outputs must be dissipative (cf. Section 4.4.2).

4.1.6 Errors in Forward & Inverse Dynamics

Before designing a physics-informed regression model, the model's input and output variables must be chosen, i.e., given the task at hand, should one resort to inverse dynamics, forward dynamics, or transition dynamics? In many robotic textbooks, an ID formulation is recommended for feed-forward control, while an FD formulation seems suitable for simulation. However, apart from the model's application, there are additional aspects that should be considered when developing a physics-informed regression model.

Multiplicative Errors

A model deviates from observation of $f(x)$ either through model errors $\{\epsilon^{\star}_{\mathrm{A,A}}, \epsilon_{\mathcal{C}_1}, \epsilon_{\hat{f}_1}\}$ or through observation noise. Equation (4.19) emphasizes that the intrinsic approximation error $\epsilon^{\star}_{\mathrm{A,I}}$ follows from products between the ideal latent functions $\{\mathcal{C}^{\star}_i, \hat{f}^{\star}_i\}$ and the latent error functions $\{\epsilon_{\hat{f}_i,\mathrm{I}}, \epsilon_{\mathcal{C}_i,\mathrm{I}}\}$. An important implication that underlies (4.19) is that the structure of error functions in FD models differs substantially from ID dynamics.

The combination of the FD as in (3.17) with (4.13) yields

$$\ddot{q} = \epsilon^{\star}_{\mathrm{A,A}} + \left(M^{-1} + \epsilon_{\mathcal{C}_1}\right)\left(Q + \epsilon_{\hat{f}_1}\right). \tag{4.20}$$

For FD as in (4.20), the latent errors are nonlinearly transformed such that the error at the output of the analytical dynamics model reads

$$\epsilon_{\mathrm{A}} = \epsilon^{\star}_{\mathrm{A,A}} + M^{-1}\epsilon_{\hat{f}_1} + \epsilon_{\mathcal{C}_1} Q + \epsilon_{\mathcal{C}_1}\epsilon_{\hat{f}_1}. \tag{4.21}$$

In turn, a data-driven model approximating ϵ_{A} learns a nonlinear transformation of the latent error functions $\{\epsilon_{\mathcal{C}_1}, \epsilon_{\hat{f}_1}\}$.

In comparison, the system's ID read

$$Q_{\mathrm{u}} = \epsilon^{\star}_{\mathrm{A,A}} + \left(M + \epsilon_{\mathcal{C}_1}\right)\ddot{q} - \left(Q_{\mathrm{C}} + \epsilon_{\hat{f}_1}\right). \tag{4.22}$$

For ID as in (4.22), the error function reads

$$\epsilon_{\mathrm{A}} = \epsilon^{\star}_{\mathrm{A,A}} + \epsilon_{\mathcal{C}_1}\ddot{q} + \epsilon_{\hat{f}_1}. \tag{4.23}$$

In turn, the latent model errors arising inside the system's forces can be directly approximated via ARM. For FD, the nonlinear transformation of errors renders ARM considerably more challenging compared to using ID as also noted in the following remark.

Remark 4.2 (ARM of FD is often more challenging than of ID)**.** *ARM models ϵ_A directly through a data-driven model. In ID models, the error in* (4.23) *denotes the sum of adventitious modeling errors and latent model errors. In FD models, the error in* (4.21) *is the result of nonlinear transformations of the latent errors* $\{\epsilon_{\hat{f}_1}, \epsilon_{\mathcal{C}_1}\}$.

The following example shall illustrate the benefits of ALM inside FD.

Example 4.3 (Multiplicative errors in pendulum dynamics)**.** The forward dynamics of an undamped-uncontrolled pendulum in terms of its angle q reads

$$\ddot{q}(q) = M^{-1}Q_{\mathrm{G}}, \tag{4.24}$$

with the inverse of the inertia matrix $M^{-1} = 1/mL^2$ and the gravitational torque $Q_{\mathrm{G}} = -\sin(q)Lmg$ 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 $\hat{Q}_g = -\sin(q)Lm(g + \epsilon_{\hat{f}_1})$ where $\epsilon_{\hat{f}_1}$ denotes a constant function. Even though $\epsilon_{\hat{f}_1}$ is constant, if it is propagated through the dynamics function the output prediction error becomes a nonlinear function

$$\epsilon_{\mathrm{A}}(q) = \frac{-\sin(q)}{L}\epsilon_{\hat{f}_1}. \tag{4.25}$$

Therefore, if the model designer is confident that the inverse inertia matrix $M(q, m, L)^{-1}$ and the kinematic dependency $\sin(q)L$ are suitable parametrizations of the real system's physics, one can place a data-driven model on $\epsilon_{\hat{f}_1}$ instead of ϵ_{A}. In turn, a less complex data-driven model can be used. Alternatively, one can use the additional prior knowledge that $g > 0$, such that $\hat{Q}_{\mathrm{G}} = -\sin(q)Lm\hat{g}^2$ where $\hat{g}(x, \theta_{\mathrm{M}})$ denotes a suitable data-driven model.

Acceleration Noise and Transition Dynamics

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. As a consequence, 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, FD models can be preferred to ID models. However, as FD formulations are usually used to compute trajectory predictions via numerical integration methods, these trajectory predictions contain an additional integration error.

This is also one reason that might explain why the majority of data-driven models approximate transition dynamics (1.4) as position and velocity measurements are usually readily available. However, solely data-driven dynamics models that directly approximate the transition dynamics have $2n_q$ output dimensions compared to n_q outputs for an FD model. If system data is sparse, the larger number of output dimensions advocates against a direct approximation of transition dynamics (also referred to as discrete-time dynamics).

Table 4.1: Summary of works on physics-informed regression of robot dynamics that are detailed in this monograph. Table adapted from *Geist and Trimpe (2021)*.

	Publication	Dynamics	Data-driven model type	Data-driven approximation	Optimization library
APN	Ledezma et al. (2017)	ID	×	×	*fmincon* (Matlab)
	Ledezma et al. (2018)	ID$^{(1)}$	×	×	*fmincon* (Matlab)
	Sutanto et al. (2020)	FD$^{(2)}$	×	×	PyTorch (Python)
	Lutter et al. (2021)	FD/ID	×	×	×
ARM	Nguyen-Tuong et al. (2010)	ID	GP	ϵ_{A}	×
	De La Cruz et al. (2011)	ID	LWPR	ϵ_{A}	×
	Um et al. (2014)	ID	GP	ϵ_{A}	×
	Grandia et al. (2018)	ID	GP/LWPR	$\tilde{\epsilon}_u$	GPy (Python)
ALM	Cheng et al. (2015)	ID	GP	$\mathcal{L}$	×
	Geist and Trimpe (2020)	FD	GP	Q_{e}, Q_z	scikit-learn (Python)
	Rath et al. (2021)	FD	GP	Q_{e}, Q_z	PyTorch (Python)
	Hwangbo et al. (2019)	FD$^{(3)}$	NN	Q_{e}	×
	Greydanus et al. (2019)	FD	NN	$\mathcal{H}$	PyTorch (Python)
	Lutter et al. (2019a)	FD/ID	NN	M, Q_{G}	PyTorch (Python)
	Lutter et al. (2019b)	FD/ID	NN	M, V	PyTorch (Python)
	Gupta et al. (2020)	FD/ID	NN	M, V, B, Q_{e}	PyTor. / Flux (Julia)
	Lutter et al. (2020)	FD$^{(4)}$	NN	Q_{e}	×
	Toth et al. (2019)	FD	NN	$\mathcal{H}$	×
	Cranmer et al. (2020)	FD	NN	$\mathcal{L}$	JAX (Python)

Measuring Joint Torques

Another important aspect relates to the measurement of Q_{u}. As discussed in further detail in Section 3.5.4, the desired control force $Q_{\mathrm{u,desired}}$ may deviate from the observed control force Q_{u}. The identification of the mapping from $Q_{\mathrm{u,desired}}$ to Q_{u} is 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 may be inaccurate. Alternatively, we can use the end-effector force to estimate Q_{u}. Yet, such estimates depend on mechanical system parameters. In comparison, directly measuring Q_{u} through force sensors in the actuators yields accurate measurements. However, joint torque sensors are costly. Compared to learning ID, learning FD has the advantage that we do not need to measure Q_{u}.

In the subsequent sub-section, we discuss the different modeling frameworks and categorize related literature in accordance with these categories.

4.2 Analytical Parameter Optimization

Analytical parameter optimization seeks to determine an analytical model $\hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}})$ through optimization of its parameters θ_{A}. The insights about analytical models are useful for improving future generations of analytical structured models.

4.2.1 Linear Regression of Analytical Dynamics

Siciliano et al. (2010, p. 280) refers to Atkeson et al. (1986) as the standard approach used for the identification of the dynamics parameters of robot arms, which leverages the *linearity* of the (rigid) robot arm's ID with respect to the dynamical parameters (Siciliano et al., 2010, p. 259), writing

$$Q_{\mathrm{u}} = \Phi(q, \dot{q}, \ddot{q})\theta_{\mathrm{A}}. \tag{4.26}$$

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

$$\hat{\theta}_{\mathrm{A}} = (\Phi^{\top}\Phi)^{-1}\Phi^{\top}\tilde{Q}_u, \tag{4.27}$$

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 (Siciliano et al., 2010, p. 280).

Many rigid body dynamics models are not linear in their parameters. Moreover, even if a dynamics model is linear in its parameters, one might wish to distinguish known from unknown parameters as discussed in Section 4.1.4. Another caveat of such linear formulations of robot dynamics is that some entries of θ_{A} denote products of rigid body dynamics parameters (Siciliano et al., 2010, p. 264). Moreover, the ability to include dissipative force models is considerably limited.

A more flexible approach for analytical parameter optimization is to treat rigid body dynamics models as analytical parametric networks (APN) (Ledezma and Haddadin, 2017, 2018; Lutter et al., 2020; Sutanto et al., 2020), which we discuss in the following.

4.2.2 Analytical Parametric Networks

Similar to neural networks, rigid body dynamics denotes a combination of differentiable functions that can be trained with gradient-based optimization methods, in particular using automatic differentiation. While technically these types of models are analytical dynamics models, the techniques used are closely related to ones from data-driven modeling, which blurs the line between analytical mechanics and solely data-driven modeling.

The use of automatic differentiation for dynamics model parameter estimation has a long history. However, the success of deep learning in big-data applications spawned a series of computational libraries for automatic differentiation and optimization. These libraries significantly simplify the optimization of rigid body dynamics via automatic differentiation as well as reduce the effort that is required to combine analytical dynamics with solely data-driven models. As shown in Table 4.1, recent physics-informed regression models in robotics are implemented using PyTorch (Paszke et al., 2019) as well as JAX (Bradbury et al., 2018). These Python libraries allow the computation of analytical gradients of parametric functions in a few lines of computer code thanks to modern libraries on automatic differentiation such as AutoGrad (Maclaurin et al., 2015), while also enabling a fast computation of the gradient functions through libraries for GPU accelerated computation such as XLA.

Ledezma and Haddadin (2017) reformulated the ID of a robot arm such that both the kinematic and dynamic parameters can be estimated via gradient-based optimization. In this approach, the ID of the robot arm are separated into a kinematic and dynamic

[(1)]: An et al. (1985), [(2)]: Luh et al. (1980), [(3)]: Hwangbo et al. (2018), [(4)]: Kim (2012)

network which allows estimating the kinematic parameters before estimating the dynamic parameters. This approach has been extended by Ledezma and Haddadin (2018) towards the identification of ID of humanoid robots. The clear advantage of such an approach is that the optimizer cannot change the kinematic parameters to compensate for modeling errors in the system's dynamics.

Physical Inconsistent Parameters

Inaccurate analytical representation of a robot's dynamics often leads to physically inconsistent parameter estimates during parameter optimization, *e.g.*, negative masses or too short length parameters. A practical example forms robot arm dynamics in which friction is not modeled sufficiently. The friction force gives rise to an error that the optimizer could try to reduce by changing the model's mass parameters. However, if the physical parameters attain implausible values, then also the description of the analytical dynamics loses its physical interpretability. In practice, physical implausible parameter estimates often deteriorate the prediction error on the test data set.

Analytical Parameter Constraints

A common approach to prevent physically inconsistent parameters is to impose additional constraints on the parameters. These constraints range from simple bounds that enforce $\nabla_{\theta_{\mathrm{A},i}} L = 0$ at the constraint's boundary or impose additional structure into the dynamics model such as defining a mass parameter as $m_i := \theta_{\mathrm{A},i}^2$.

Similar to Ledezma and Haddadin (2017), Sutanto et al. (2020) described a recursive formulation of the Newton-Euler ID as a differentiable computational graph. Significantly, Traversaro et al. (2016) show that not every positive-definite matrix constitutes a physically plausible inertia matrix as the rotational inertia matrix must also fulfill *triangular inequalities* with respect to the principal moments of inertia. These insights led Sutanto et al. (2020) to a parametrization of the rotational inertia matrix of each body respecting triangular inequalities. It should be noted that the experiments detailed in Sutanto et al. (2020) 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.

Even if an approach towards the estimation of an analytical dynamics model combines constraints on the parameters, refrains from estimating "known" parameters, and trains kinematics independently from dynamic parameters, the problem remains that some important physical phenomena are often not sufficiently described by the analytical dynamics model. While parameter estimation of an analytical dynamics model often yields better results on small data sets than purely data-driven models (Ledezma and Haddadin, 2017; Sutanto et al., 2020; Traversaro et al., 2016), the question arises how a data-driven model can be incorporated into an analytical model to reduce ϵ_{A}.

4.3 Analytical Residual Modeling

A straightforward approach to account for ϵ_{A}, is to add a data-driven model $\hat{\epsilon}_{\mathrm{A}}(x, \theta_{\mathrm{M}})$ with parameters θ_{M} to the analytical dynamics model, writing

$$\hat{f}(x, \theta_{\mathrm{A}}, \theta_{\mathrm{M}}) = \hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}}) + \hat{\epsilon}_{\mathrm{A}}(x, \theta_{\mathrm{M}}). \tag{4.28}$$

We refer to the above model as an *analytical (output) residual model* (ARM). ARM forms a natural extension to solely resorting to rigid body dynamics models. Such an approach to physics-informed regression is conceptually easy to implement as the data-driven model can be directly trained on observations of $\epsilon_{\mathrm{A}}(x, \theta_{\mathrm{A}})$. There are, of course, practical challenges in ARM such as noise as well as optimizing the parameters θ_{A} alongside θ_{M}.

Many works on ARM exist in robotics literature. In what follows, we discuss a small selection of ARM models that provides a first glance on the different problem setting one can encounter in robot dynamics identification.

4.3.1 Semi-parametric GP Regression of Analytical Dynamics

To address the short-comings of the least-squares dynamics model (4.27), Nguyen-Tuong and Peters (2010) proposed to combine (4.27) with GP regression resulting either 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

$$\hat{Q}_u \sim \Phi(x)\hat{\theta}_{\mathrm{A}} + \mathcal{GP}\left(0, \mathbf{k}(x, x')\right) = \mathcal{GP}\left(\Phi(x)\hat{\theta}_{\mathrm{A}}, \mathbf{k}(x, x')\right), \tag{4.29}$$

with $\mathbf{k}(x, x')$ denoting a diagonal matrix-valued kernel function. A Gaussian distributed estimate of the system's dynamic parameters $\hat{\theta}_{\mathrm{A}}$ can then be inferred via the posterior distribution of (4.29) (Williams and Rasmussen, 2006, p. 27-29).

The second model proposed in Nguyen-Tuong and Peters (2010) uses the kernel trick to obtain an analytical kernel of the ID, writing

$$\mathbf{k}_{\mathrm{A}}(x, x') = \Phi^{\top} W \Phi + \Sigma_y, \tag{4.30}$$

with Σ_y denoting a diagonal matrix of observation noise variances and W being a diagonal matrix denoting the prior variance on θ_{A}. An algorithm that is defined solely in terms of inner products in input space is lifted by the kernel trick into feature space (Williams and Rasmussen, 2006, p. 12). The analytical kernel GP can then be combined with $\hat{\epsilon}_{\mathrm{A}} \sim \mathcal{GP}\left(0, \mathbf{k}(x, x')\right)$ to yield

$$\hat{Q}_u = \mathcal{GP}\left(0, \mathbf{k}_{\mathrm{A}}(x, x') + \mathbf{k}(x, x')\right). \tag{4.31}$$

Nguyen-Tuong and Peters (2010) showed using data of a real robot, that (4.29) and (4.31) achieved comparable prediction accuracy, while (4.29) was slightly faster in computing predictions. Similar to Nguyen-Tuong and Peters (2010), De La Cruz et al. (2011) combined prior analytical knowledge of a robot arm's ID 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. (2011) assumed the analytical parameters as fixed. Other semi-parametric models for learning robot arm ID are found in Camoriano et al. (2016).

While $k(x, x')$ in (4.29) and (4.31) can be chosen as a matrix-valued kernel function with off-diagonal entries, it is unclear how to design the off-diagonal kernel functions correlating the different process dimensions. Therefore, Nguyen-Tuong and Peters (2010) chose a diagonal $k(x, x')$ such that the dimensions of the dynamics function are modeled by independent one-dimensional GPs.

4.3.2 Combining Recursive Newton-Euler with GPs

Um et al. (2014) augment the recursive-Newton-Euler-algorithm (RNEA) (Featherstone, 2008, p. 98) by a GP. First, the RNEA computes the system's analytical ID using a forward recursion of kinematic functions and a backward recursion of forces. In turn, the system's analytical joint torques Q_e are computed. The observed joint torques $\tilde{Q}_\mathrm{e}$ deviate from Q_e in the i-th joint by an error $\epsilon_{\mathrm{A},i}$, writing

$$\tilde{Q}_{\mathrm{e},i} = Q_{\mathrm{e},i} + \epsilon_{\mathrm{A},i}. \tag{4.32}$$

The above equations assume that the error in the joint torque arises solely *inside* the joints, *e.g.*, through friction or shaft elasticities. In comparison to Nguyen-Tuong et al. (2008) where $\epsilon_{\mathrm{A},i}(q, \dot{q}, \ddot{q})$, it is assumed that the error only depends on the velocity variables of the i-th joint and force functions that act on the i-th body, writing $\epsilon_{\mathrm{A},i}(x_i)$ with $x_i := [\omega_i, \dot{\omega}_i, Q_{\mathrm{e},i+1}, F_{i+1}]$ where F_{i+1} denotes the Cartesian force that acts normal to the axis of joint $i+1$. This error function is then modeled through a GP, writing

$$\hat{\epsilon}_{\mathrm{A},i} \sim \mathcal{GP}\Big(0, k(x(t_1), x(t_2))\Big), \tag{4.33}$$

where $k(x(t_1), x(t_2))$ has been chosen as a squared exponential kernel.

The reduction of the dimensionality of the input space of the model is one important aspect of how the combination of physics knowledge improves the sample efficiency of a data-driven model. The forces $\{Q_{\mathrm{e},i+1}, F_{i+1}\}$ encode information underlying the analytical dynamics model in only two variables.

Remark 4.4 (On choosing input variables in physics-informed regression.)**.** *When a data-driven model approximates the errors in the prediction of an analytical dynamics model, physics often yields structural knowledge on which variables to choose as inputs to the data-driven model. The reduction of a data-driven model's input space to span only relevant input dimensions improves the model's sample efficiency.*

In summary, the fundamental principles underlying physics provide useful prior knowledge for dimensionality reduction in data-driven dynamics identification.

4.3.3 Deriving Contact-invariant Errors

Inspired by the implicitly constrained dynamics descriptions detailed by Aghili (2005), Grandia et al. (2018) learned the residual of a quadruped robot's ID in a formulation that stays invariant under *changes in the contact configuration*. Here, if the point-feet of the quadruped is pressed onto a surface, holonomic constraints are activated which are expressed via implicit constraints on the acceleration level (3.30). As the first step, the authors model a quadruped's dynamics as in (3.27)

$$M\ddot{q} = Q + Q_\mathrm{I} + \epsilon_\mathrm{A}, \tag{4.34}$$

with $Q_\mathrm{I} = A^\mathsf{T}\lambda$. In this particular example, A is chosen such that λ^T corresponds to a vector of Cartesian forces that act on the quadruped's feet while the respective constraints are active. The implicit constraint forces Q_I are eliminated from the EOM using D'Alembert's principle (3.33), yielding the projected force error $(I - A^+A)\epsilon_\mathrm{A}$ as

$$(I - A^+A)\epsilon_\mathrm{A} = (I - A^+A)(Q - M\ddot{q}). \tag{4.35}$$

As pointed out in Section 3.4, the activation of a constraint summons an implicit constraint force Q_I that opposes the part of Q that is in $\mathsf{R}(A^\mathsf{T})$, and subsequently changes the system's acceleration. In turn, when the quadruped's feet get in contact with the ground, $\ddot{q}$ changes to the constrained acceleration and a jump occurs in ϵ_A. As learning jumps in ϵ_A is difficult for many data-efficient regression models, Grandia et al. (2018) use a coordinate transformation J_u to express the force error via

$$\epsilon_\mathrm{A} = A^\top\tilde{\epsilon}_c + J_u\tilde{\epsilon}_u \tag{4.36}$$

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 (4.35) one obtains $(I - A^+A)\hat{\epsilon}_\mathrm{A} = (I - A^+A)J_u\tilde{\epsilon}_u$ and in return a *constraint invariant formulation* of the force error as

$$\tilde{\epsilon}_u = \big((I - A^+A)J_u\big)^+ (I - A^+A)(Q - M\ddot{q}). \tag{4.37}$$

The authors approximated $\tilde{\epsilon}_u$ using LWPR as well as GP regression. It is assumed that the robot's feet do not slip. If the robot's feet are subject to stick-slip effects, a friction force Q_z may cause the error in (4.37) to also show discontinuities.

4.4 Analytical Latent Modeling

ARM treats $\epsilon_\mathrm{A}(x, \theta_\mathrm{A})$ as a black box without incorporating additional structural knowledge. However, in many robotic systems, the function $\epsilon_\mathrm{A}(x, \theta_\mathrm{A})$ origins from numerous physical phenomena spawning error functions that considerably differ in their magnitude and mathematical properties.

Recent works on physics-informed regression suggest the approximation of unknown latent functions with data-driven models to effectively reduce $\epsilon^\star_\mathrm{A,I}$. We refer to physics-informed regression models in which a data-driven model approximates latent functions $\{\mathcal{C}_i, \hat{f}_i\}$ as analytical latent modeling (ALM). ALM allows to incorporate prior knowledge of 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 the type of latent functions that are approximated with a data-driven model, namely:

- the entries of M, M^{-1}, or a corresponding Lagrangian/ Hamiltonian,
- the entries of Q_e such that the data-driven model respects dissipativity, uses kinematic transformations, or learns in a function space with a smaller dimensionality compared to $f(x)$.

In addition, we proposed a novel approach to ALM in which implicitly constrained transformations are incorporated into the physics-informed regression model. This particular approach is detailed in Section 5.

4.4.1 Learning Mass-related Quantities

The generalized inertia matrix M is of great significance for rigid body dynamics modeling. The fictitious force Q_{F} can be derived in terms of M, cf. (3.25). Moreover, if Q_{G} denotes the gravitational force then this force is also a function of the mass parameters. As discussed in Section 4.1.6, in FD the force functions $(Q + Q_{\mathrm{I}})$ are multiplied by M^{-1}, while in ID 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, such an assumption may neglect significant errors in M or M^{-1} and in the bias force Q_{C}. Therefore, recent works model inertia-related quantities 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 / Hamiltonian. The works on Lagrangian and Hamiltonian NNs were inspired by the seminal work of Chen et al. (2018) on neural differential equations.

We note that these works often neglect errors in the dissipative forces Q_D. If Q_D is the major source for 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).

Learning the Lagrangian

Often analytical parametrizations of M^{-1} and Q_{C} are either not available or the effort required to obtain 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 FD in (3.23) and for the corresponding ID. One of the first works that modeled the Lagrangian function via a GP was proposed by Cheng et al. (2015). Cheng et al. (2015) placed a GP prior on $\mathcal{L}$ inside of (3.23), writing

$$\hat{\mathcal{L}} \sim \mathcal{GP}(0, k(x, x')), \tag{4.38}$$

to obtain a structured model for the ID of a conservative system. However, it is currently not clear how to insert such an $\hat{\mathcal{L}}$ for obtaining an FD model as efficient multi-output GP regression requires that $\hat{\mathcal{L}}$ is solely linearly transformed. In comparison, Cranmer et al. (2020) models $\mathcal{L}$ via a NN. In return, the authors obtain structured models both for FD as in (3.23) as well as ID.

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. Still, Hamiltonian dynamics is of utmost importance in other branches of mechanics such as quantum mechanics, celestial mechanics, and thermodynamics (cf. Greydanus et al. (2019)). 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

$$\dot{q}_i = \frac{\partial \mathcal{H}}{\partial p_i}, \qquad \dot{p}_i = -\frac{\partial \mathcal{H}}{\partial q_i}, \tag{4.39}$$

with $\mathcal{H}(q,p,t)$, $\mathcal{H} :\in \mathbb{R}^{2n} \rightarrow \mathbb{R}$ denoting the Hamiltonian function. Similar to the Lagrangian NN, Greydanus et al. (2019) models the Hamiltonian in the above equation via a NN. This model was extended by Toth et al. (2019) using a generative NN structure which enables the inference of Hamiltonian dynamics from high-dimensional observations such as images.

Learning the Entries of the Inertia Matrix

As shown in (3.19), the kinetic energy of a rigid body system is described in terms of the generalized inertia matrix M. Thereby, the FD as in (3.24), as well as its ID, 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 Traversaro et al. (2016), the inertia matrix M as derived in Section 3.2 must be positive definite. Therefore, Lutter et al. (2019a) 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 4.2.2). 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. (2019a) modeled the potential forces Q_{G} as well as non-conservative forces Q_{D} jointly via a NN. The authors named the resulting structured model a *Deep Lagrangian Neural Network (DELAN)*. Lutter et al. (2019b) 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 $Q_{\mathrm{G}} = -\nabla_q \hat{V}(q)$. Albeit, the prediction accuracy of DELAN approximating the dynamics of a real cart-pole and a real Furuta pendulum compare less favorable to white-box system identification as proposed by Atkeson et al. (1986) with the incorporation of viscous friction as in (Ting et al., 2006). Yet, the fact that DELAN achieved almost comparable prediction accuracy on these systems without using any prior knowledge of the system's underlying kinematics and mass parameters constitutes a notable achievement.

Gupta et al. (2020) extended DELAN by leveraging that the control force is affine in the control signal u, writing $Q_u = B(q)u$. Gupta et al. (2020) added NN parametrizations of $B(q)$ and $V(q)$, such that the deep Lagrangian network becomes

$$\ddot{q} = (\hat{L}\hat{L}^T)^{-1}\left(-\nabla_q(\dot{q}^\top \hat{L}\hat{L}^T)\dot{q} + \frac{1}{2}\left(\nabla_q\left(\dot{q}^T\hat{L}\hat{L}^T\dot{q}\right)\right)^T - \nabla_q\hat{V}(q) + \hat{B}(q)u + \hat{Q}_d\right). \quad (4.40)$$

4.4.2 Learning 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 the 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 conservative parts of the dynamics expressed in terms of $\{M, Q_{\mathrm{C}}\}$ would still conserve the model's total 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 dynamics' errors stem from an inaccurate description of the impressed forces Q_{e} , which in the following is referred to as joint torques.

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. (2019) identified the joint torques Q_e 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_e using joint-torque sensors as well as position errors and velocities. Then, a NN was trained to predict Q_e given a sequence of position errors and velocities. In this manner, the authors learned the complete mapping of Q_e 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 (Hwangbo et al., 2018). 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 a randomization of physical parameters (body length and mass) of a good analytical model in combination with a prior identification of Q_e. This work was further extended by Lee et al. (2020) who trained an unprecedented robust control policy for a quadruped robot traversing challenging terrain.

Modeling of joint torques inside forward dynamics

Lutter et al. (2020) (being the authors of DELAN) combined Newton-Euler dynamics in Lie Algebra form Kim (2012) with a NN parametrization of Q_u. The authors compared several models for Q_u, with f_NN denoting a NN model, namely

$$\text{Viscous: } \hat{Q}_\mathrm{e} = Q_\mathrm{u,desired} - \theta_v \dot{q}, \tag{4.41}$$

$$\text{Stribeck: } \hat{Q}_\mathrm{e} = Q_\mathrm{u,desired} - \mathrm{sign}(\dot{q}) \left(f_s + f_d \exp\left(-\theta_s \dot{q}^2\right)\right) - \theta_v \dot{q}, \tag{4.42}$$

$$\text{NN Friction: } \hat{Q}_\mathrm{e} = Q_\mathrm{u,desired} - \mathrm{sign}(\dot{q}) \left\| f_\mathrm{NN}\left(q, \dot{q}\right)\right\|_1, \tag{4.43}$$

$$\text{NN Residual: } \hat{Q}_\mathrm{e} = Q_\mathrm{u,desired} - f_\mathrm{NN}(q, \dot{q}), \tag{4.44}$$

$$\text{FF-NN: } \hat{Q}_\mathrm{e} = f_\mathrm{NN}(Q_\mathrm{u,desired}, q, \dot{q}). \tag{4.45}$$

Equations (4.41), (4.42), and (4.43) are guaranteed to be sole *dissipative*, while the more classical NN parametrization in (4.44) and (4.45) do not respect energy dissipativity. However, the first three models assume that the internal motor control routines do not cause an overshoot such that $|Q_\mathrm{u,i}| > |Q_\mathrm{u,desired,i}|$.

The different structured FD models were trained on data from simulated and real pendulums. Additionally, these models were compared to NN black-box modeling as well as the linear regression model denoted in (4.26) and (4.27) (cf. Atkeson et al. (1986)). 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 $\hat{Q}_{\mathrm{e}}$ made significantly better long-term predictions. The long-term predictions were computed by feeding the models' acceleration predictions to a Runge-Kutta-4 solver.

Chapter 5

Implicitly-constrained Gaussian Processes

As detailed in Section 3.2, the current literature on ARM and ALM evolves around the combination of data-driven regression with explicitly constrained rigid body dynamics. Only a few works that revolve around the dynamics identification of quadrupedal robots used implicit-constraint knowledge to structure a data-driven model, *e.g.*, (Grandia et al., 2018). As discussed in Section 3.4, for many mechanical systems including quadrupedal robots and robot arms, implicit constraint equations yield information on implicit constraint forces Q_{I} as well as on constrained motion as long as the implicit constraints are active.

In what follows, we present a framework that combines GP regression with implicitly-constrained rigid body dynamics. While the errors in these dynamics can also be identified using NNs, we resort to GP regression as it allows us to obtain probabilistic predictions of latent processes, such as the implicit constraint forces Q_{I}, through multivariate distributions as detailed in Section 5.1.3. These sections are based on *Geist and Trimpe (2020)* as well as *Rath et al. (2021)*.

5.1 Model Synthesis

For convenience, we briefly recall the EOM of implicitly constrained systems as in (3.42), providing an analytical dynamics model $\hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}})$ as

$$\ddot{q} = Lb + P\bar{Q},$$

with $M > 0$, the matrices $L = M^{-1}A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}$, $P = TM^{-1}$ with $T = (I - M^{-1}A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}A)$, and $\bar{Q} = Q + Q_z$. The above EOM always fulfills the implicit constraining equation (3.30) being

$$A(q, \dot{q}, t)\ddot{q} = b(q, \dot{q}, t),$$

where the terms $\{A, b\}$ are obtained from differentiating the implicit constraints a sufficient number of times as shown in (3.28) and (3.29). Note that predictions $\{q_{k+1}, \dot{q}_{k+1}\}$ made via integration of (3.42) only respect the implicit constraints on the position-level and velocity-level, if the inputs to the EOM as in (3.42) denoted by $\{q_k, \dot{q}_k\}$ themselves respect the implicit constraints on position- and velocity-level.

As discussed in Section 4.1, for robotic systems, the EOM deviates from the real dynamics by an error ϵ_{A}, such that

$$\tilde{y}(x) - \hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}}) = \epsilon_y(x) + \epsilon_{\mathrm{A}}(x, \theta_{\mathrm{A}}), \tag{5.1}$$

with the observations of the dynamics $\tilde{y}$, the input state vector $x = [q, \dot{q}, t]$ (if the system is autonomous $x = [q, \dot{q}]$), and the observation noise $\epsilon_y(x)$. The observation noise ϵ_y is modeled as

$$\hat{\epsilon}_y \sim \mathcal{N}(0, \Sigma_y), \tag{5.2}$$

where Σ_y denotes a diagonal matrix containing the measurement variances.

Similar to recent works on physics-informed ALM as detailed in Section 4.4.2, we place the following assumptions on the implicitly constrained EOM:

Assumption 5.1. *The bodies of the mechanical system are rigid.*

Assumption 5.2. *The robot's kinematics, implicit constraints, and their parameters are known a-priori.*

As noted in Section 4.1.4, the above assumptions imply that the mass-related quantities $\{M, Q_{\mathrm{C}}\}$ are parametrically-unknown ($\epsilon_{\mathcal{C},\mathrm{I}} = 0$, $\epsilon_{\mathcal{C},\mathrm{E}} \neq 0$) such that $\mathcal{C}^\star = P + \epsilon_{\mathcal{C},\mathrm{E}}$, whereas the forces $\{Q, Q_z\}$ introduce a latent error $\epsilon_{\hat{f}} = \epsilon_{\hat{f},\mathrm{I}} + \epsilon_{\hat{f},\mathrm{E}}$, and thereby, the process that generates the observations reads

$$\tilde{y} = \hat{\epsilon}_y + \epsilon^\star_{\mathrm{A,A}} + (P + \epsilon_{\mathcal{C},\mathrm{E}})(Q_{\mathrm{K}} + \epsilon_{\hat{f}}), \tag{5.3}$$

where Q_{K} refers to all force functions from which an accurate parametric model can be derived from analytical mechanics. If $\{M, Q_{\mathrm{C}}\}$ are assumed to be only parametrically-unknown and Q_u is known, then $Q_{\mathrm{K}} = Q_{\mathrm{C}} + Q_u$.

The latent estimation errors $\{\epsilon_{\mathcal{C},\mathrm{E}}, \epsilon_{\hat{f},\mathrm{E}}\}$ are a consequence of limited resources during optimization, such as the amount and distribution of data as well as the time available for training. One can attempt to reduce the estimation error through a conscious selection of the loss, by preprocessing data and collecting a sufficient amount of data in relevant parts of the state space.

The approximation errors $\{\epsilon^\star_{\mathrm{A,A}}, \epsilon_{\hat{f},\mathrm{I}}\}$ in (5.3) can be modeled through a combination of ARM and ALM (cf. Section 4). In principle, one could model the approximation errors in the above equations through a NN $f_{\mathrm{NN}}(x, \theta_{\mathrm{M}})$. As the combination of the analytical dynamics model $\hat{f}_{\mathrm{A}}(x, \theta_{\mathrm{A}})$ with a NN is also a differentiable parametric model, it is possible to estimate the model's parameters $\{\theta_{\mathrm{A}}, \theta_{\mathrm{M}}\}$ via automatic differentiation. However, we model the approximation error through GP regression as they provide a measure of prediction uncertainty in the form of the posterior variance. Additionally, they allow incorporating various model assumptions through the kernel. As pointed out by Deisenroth and Rasmussen (2011), the synthesis of a robot's control policy may significantly benefit from a measure of uncertainty in the planned motion.

5.1.1 GP Priors

To obtain a physics-informed regression model for the system's implicitly constrained dynamics, a multi-dimensional GP prior is placed on the errors of (5.3), writing

$$\hat{\epsilon}^\star_{\mathrm{A,A}} \sim \mathcal{GP}\left(0,\ k_{\epsilon^\star_{\mathrm{A,A}}}(x, x'; \theta_{\mathrm{M}})\right), \quad \text{and} \quad \hat{\epsilon}_{\hat{f}} \sim \mathcal{GP}\left(0,\ k_{\epsilon_{\hat{f}}}(x, x'; \theta_{\mathrm{M}})\right). \tag{5.4}$$

The matrix-valued kernels in (5.4) such as $k_{\epsilon_{\hat{f}}} : \mathbb{R}^{n_x \times n_{x'}} \mapsto \mathbb{R}^{n_q \times n_q}$ determine the covariance between the individual dimensions of the error functions.

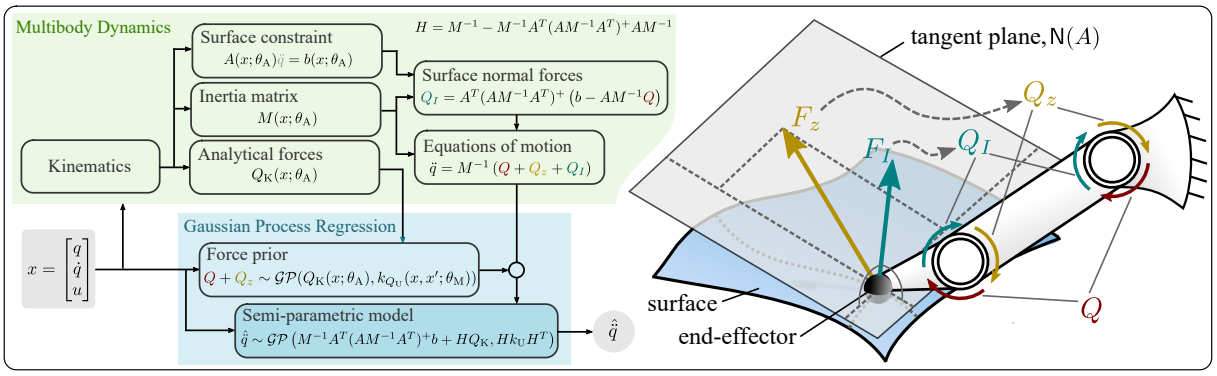


Figure 5.1: Computational graph of the proposed framework. A robot arm's forward dynamics model is augmented by a GP modeling the errors in the analytical forces $\bar{Q}$. The semi-parametric model predicts acceleration $\ddot{q}$ while potentially including knowledge in form of analytical functions $\{M, A, b, Q_K\}$. Figure adapted from *Rath et al. (2021)*.

Assumption 5.3. *No additional knowledge on the errors of the implicitly-constrained dynamics functions is available.*

With the above assumption, a suitable choice for the kernels $\{k_{\epsilon^\star_{A,A}}, k_{\epsilon_{\hat{f}}}, k_{\epsilon^\star_{\dot{A},A}}\}$ are diagonal matrices of squared-exponential kernels, as the estimation of off-diagonal covariance functions from data forms in itself a challenging endeavour (Alvarez et al., 2012).

The above GP prior yields a physics-informed model for $\bar{Q}$, as

$$\hat{\bar{Q}} \sim \mathcal{GP}\left(Q_K(x;\theta_A),\ k_{\epsilon_{\hat{f}}}(x,x';\theta_M)\right). \tag{5.5}$$

With the parametrically-unknown functions $\{M, Q_C\}$, the zero-mean GP $\mathcal{GP}(0, k_{\epsilon_{\hat{f}}})$ effectively models the unknown forces Q_D and Q_z.

5.1.2 Gauss Principle Adhering Gaussian Processes

To combine GP regression with (5.3), we leverage that $\tilde{y}$ is obtained in terms of sums of functions, in which $\epsilon_{\hat{f}}$ is being solely linearly transformed. As detailed in Section 2.4.2, the sum of GPs yield GPs (Duvenaud, 2014), while the transformation of a GP by a bounded linear operator is also a GP (Williams and Rasmussen, 2006). In turn, inserting the GP priors (5.4) into the implicitly-constrained dynamics (3.42) yields a physics-informed GP model as

$$\hat{\ddot{q}} \sim \mathcal{GP}\left(Lb + PQ_K,\ k_{\epsilon^\star_{A,A}} + Pk_{\epsilon_{\hat{f}}}P^\mathsf{T}\right). \tag{5.6}$$

The above model is referred to as a Gauss' Principle adhering Gaussian Process (GP^2) and is illustrated in Figure 5.1. By construction, the GP^2's predictions satisfy the implicit-constraint equations on acceleration-level (3.30). To make predictions with (5.6), the GP conditional posterior formula is used as denoted in (2.79).

Efficient Computation of Linear-transformed Kernel Functions

A considerable practical challenge forms the fast computation of the covariance matrix $\Sigma^{\ddot{q}}_{X,X'}$ over two input data sets $X = [x_1, x_2, ..., x_N]$ with $X \in \mathbb{R}^{n_{\ddot{q}} \times N}$ and $X' = [x'_1, x'_2, ..., x'_N]$

with $X' \in \mathbb{R}^{n_{\ddot{q}} \times N'}$ as detailed in Section 2.4. In our first work on GP^2 *(Geist and Trimpe, 2020)*, $\Sigma^{\ddot{q}}_{X,X'}$ is computed via a double for-loop, while the hyper-parameter of the GP's are computed via the numerical gradient of a maximum-likelihood. As discussed in Section 2.1.1, the computation through double for-loops and optimization using numerical gradients is slow.

In comparison, in *(Rath et al., 2021)*, we propose a significantly faster workflow by resorting to GPU acceleration and automatic differentiation. In this implementation of GP^2, the line of code that computes $\Sigma^{\ddot{q}}_{X,X'}$ can be found in code attached as supplementary material to *(Rath et al., 2021)*. The code is written using the "Python" programming language resorting to the "PyTorch" machine learning library. In particular, in this Python implementation of GP^2, the "SGPKernel" class computes $\Sigma^{\ddot{q}}_{X,X'}$ as

```
# Mulitask covariance linear transformation --> T1 * Cov * T2^\transp
  cov_quant1_quant2 = torch.einsum('iab,ijbc,jdc->ijad',
      linearoperator1,
      covar_F_F,
      linearoperator2
  )
  return cov_quant1_quant2
```

Here, we used Torch's "Einsum" function as it resulted in a fast computation of $\Sigma^{\ddot{q}}_{X,X'}$. Figure 5.2 illustrates how the above code computes the matrix $\Sigma^{\ddot{q}}_{X,X'}$. Here, $\Sigma^{\ddot{q}}_{X,X'}$ is stored as a four dimensional tensor of size $N \times N' \times n_{\ddot{q}} \times n_{\ddot{q}}$ which can be interpreted as a $N \times N'$ matrix of $n_{\ddot{q}} \times n_{\ddot{q}}$ matrices that depend on the respective i-th and j-th input points. This matrix is computed using the tensors "linearoperator1" of size $N \times \mathtt{a} \times \mathtt{b}$ and "linearoperator2" of size $N' \times \mathtt{c} \times \mathtt{d}$ as well as the covariance matrix prior "covar_F_F" of size $N \times N' \times \mathtt{b} \times \mathtt{c}$. Note that in Figure 5.2, "linearoperator1" denotes the three-dimensional array $[[H(x_1)], [H(x_2)], ..., [H(x_N)]]$ as in this example we compute $\Sigma^{\ddot{q}}_{X,X'}$. The above computation of a structured covariance matrix allows to use arbitrary matrix-valued parametric functions as "linearoperator1", *e.g.*, one can derive the joint distribution between the system acceleration and its implicit constraint forces. As detailed in Section 5.1 such a joint distribution can be used to predict the implicit constraint force solely using acceleration measurements.

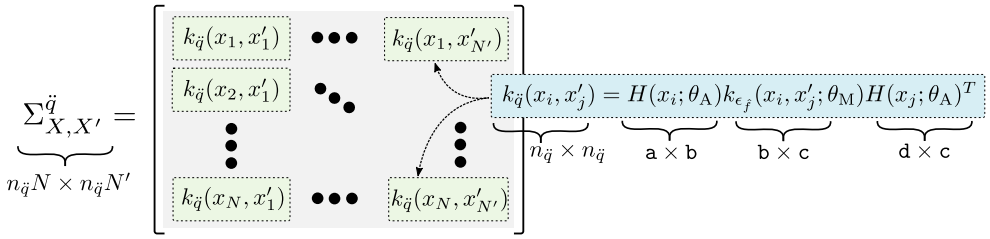


Figure 5.2: Illustration of the computation of the structured GP's covariance matrix $\Sigma^{\ddot{q}}_{X,X'}$. Figure adapted from *Rath et al. (2021)*.

With the above code, PyTorch can compute $\Sigma^{\ddot{q}}_{X,X'}$ using GPU acceleration, which significantly speeds up the computation. Moreover, if $\Sigma^{\ddot{q}}_{X,X'}$ is used to compute the GP's maximum likelihood, PyTorch allows GPU accelerated computation of the *analytical*

gradients with respect to the model parameters $\{\theta_{\mathrm{A}}, \theta_{\mathrm{M}}\}$.

5.1.3 Joint Distributions

A remarkable feature of structured GP regression is the probabilistic prediction of latent quantities inside the implicitly-constrained dynamics using acceleration measurements.

With the assumptions made at the beginning of this Chapter, the unknown forces $Q_z + Q$ are the major source of the approximation error $\epsilon_{\hat{f},\mathrm{I}}$. In (5.5), $\bar{Q}$ is modeled through a GP with a single kernel function. However, we can consider a slightly more general case in which Q and Q_z are modeled as individual GPs

$$\hat{Q} \sim \mathcal{GP}\left(Q_{\mathrm{K}},\, k_Q\right), \;\; \text{and} \;\; \hat{Q}_z \sim \mathcal{GP}\left(m_{Q_z},\, k_{Q_z}\right), \tag{5.7}$$

such that the GP model of $\bar{Q}$ reads

$$\hat{\bar{Q}} \sim \mathcal{GP}\left(m_{\bar{Q}},\, k_{\bar{Q}}\right), \tag{5.8}$$

with $m_{\bar{Q}} = Q_{\mathrm{K}} + m_{Q_z}$ and $k_{\bar{Q}} = k_Q + k_{Q_z}$. In this case, the parametric function prior Q_{K} could be chosen as a combination of an analytical model of Q_{C} and a Coulomb friction model. Moreover, k_{Q_z} could be itself a linearly transformed kernel that uses structural knowledge arising from the known implicit constraints.

We recall from (3.41) that the system's Lagrange multipliers are given as

$$\lambda = L'b - L^{\mathsf{T}}Q,$$

with $L' = (AM^{-1}A^{\mathsf{T}})^{+}$ and using the fact that

$$\left((AM^{-1}A^{\mathsf{T}})^{+}\right)^{\mathsf{T}} = \left((AM^{-1}A^{\mathsf{T}})^{\mathsf{T}}\right)^{+} = (AM^{-1}A^{\mathsf{T}})^{-1}. \tag{5.9}$$

In addition, as stated in (3.17), if the implicit constraints are inactive the system's unconstrained acceleration a is obtained as

$$a = M^{-1}Q. \tag{5.10}$$

In what follows, it assumed $\epsilon^{\star}_{\mathrm{A,A}} = 0$. As a robot's implicit dynamics result from a sum of linearly transformed processes which are modeled as either by parametric functions or GPs, the joint distribution between the processes $\{\ddot{q}, a,\, \lambda, \bar{Q}, Q, Q_z\}$ reads

$$\begin{bmatrix} \hat{\ddot{q}} \\ \hat{a} \\ \hat{\lambda} \\ \hat{\bar{Q}} \\ \hat{Q} \\ \hat{Q}_z \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} Lb{+}Pm_{\bar{Q}} \\ M^{-1}Q_{\mathrm{K}} \\ L'b{-}L^{\mathsf{T}}Q_{\mathrm{K}} \\ m_{\bar{Q}} \\ Q_{\mathrm{K}} \\ m_{Q_z} \end{bmatrix}, \begin{bmatrix} Pk_{\bar{Q}}P^{\mathsf{T}} & Pk_QM^{-1} & Pk_QL & Pk_{\bar{Q}} & Pk_Q & Pk_{Q_z} \\ M^{-1}k_QP^{\mathsf{T}} & M^{-1}k_QM^{-1} & M^{-1}k_QL & M^{-1}k_{\bar{Q}} & M^{-1}k_Q & 0 \\ L^{\mathsf{T}}k_QP^{\mathsf{T}} & L^{\mathsf{T}}k_QM^{-1} & L^{\mathsf{T}}k_QL & L^{\mathsf{T}}k_{\bar{Q}} & L^{\mathsf{T}}k_Q & 0 \\ k_{\bar{Q}}P^{\mathsf{T}} & k_QM^{-1} & k_QL & k_{\bar{Q}} & k_Q & k_{Q_z} \\ k_QP^{\mathsf{T}} & k_QM^{-1} & k_QL & k_Q & k_Q & 0 \\ k_{Q_z}P^{\mathsf{T}} & 0 & 0 & k_{Q_z} & 0 & k_{Q_z} \end{bmatrix} \right) \tag{5.11}$$

with $L' = (AM^{-1}A^{\mathsf{T}})^{+}$. A zero in the off-diagonal terms of the joint distribution implies that the two processes are uncorrelated.

Remark 5.1 (Joint distribution of dynamics processes)**.** *From the joint distributions* (5.11) *it follows that with* (2.84) *one can use the proposed framework to predict {q̈, a, λ, Q_I, Q̄, Q, Q_z} using measurements of either {q̈, a, λ, Q_I, Q̄, Q, Q_z}.*

Remark 5.1 assumes that during optimization the two GPs $\hat{Q}$ and $\hat{Q}_z$ solely approximate the latent function Q or Q_z, respectively. If the hyper-parameter estimates of $\hat{Q}$ are strongly influenced by the errors arising from Q_z, and vice versa, predictions made using Remark 5.1 can be erroneous. It remains an open question how one can ensure that a force GP prior which models a specific force function is not influenced by the errors that arise from other forces. One possible approach to solve this problem is to first estimate the hyper-parameters of $\hat{Q}$ on data of the unconstrained system, and afterward, solely train the hyper-parameters of $\hat{Q}_z$ on data of the constrained system. Another approach is to use additional structural knowledge to only feed relevant inputs to the GPs $\hat{Q}$ and $\hat{Q}_z$ to aggravate the approximation of errors arising from other physical phenomena.

As a first step toward more elaborate structured GP models, we modeled either the acceleration $\{a + a_z\}$ or forces $\{Q + Q_z\}$ inside the analytical model as a single GP, *e.g.*, yielding a force prior as in (5.5) using a diagonal matrix of squared-exponential kernels.

In the subsequent Sections, we are going to discuss first experiments on dynamics identification with GP^2 as in *(Geist and Trimpe, 2020)* and *(Rath et al., 2021)*. In Section 5.2.2, we show how to predict a given measurements of $\ddot{q}$ as well as make predictions of $\ddot{q}$ assuming that we obtained measurements from the system while being subject to differing implicit constraints $A'\ddot{q} = b'$. In Section 5.3.3, we briefly show the prediction of the end-effector contact force using measurements of $\ddot{q}$.

5.2 Learning Implicitly-constrained Accelerations

In our initial work on GP^2 *(Geist and Trimpe, 2020)*, it is suggested to place a GP prior on the unconstrained acceleration $\bar{a} = a + a_z = M^{-1}\bar{Q}$, writing

$$\hat{\epsilon}_{\bar{a}} \sim \mathcal{GP}\left(0,\ k_{\epsilon_{\hat{f}}}(x, x'; \theta_{\mathrm{M}})\right), \tag{5.12}$$

yielding a prior on $\bar{a}$ as

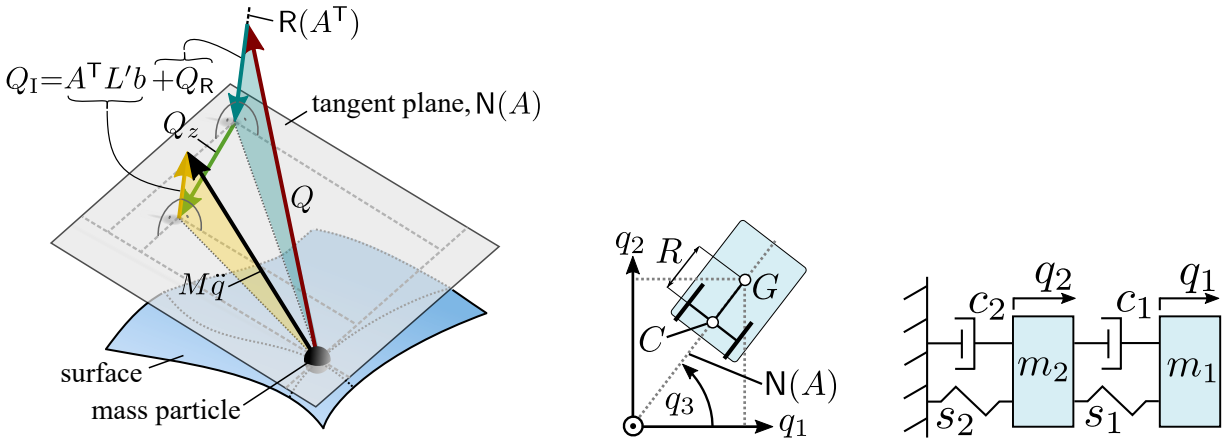
$$\hat{\bar{a}} \sim \mathcal{GP}\left(\bar{a}_{\mathrm{K}}(x; \theta_{\mathrm{A}}),\ k_{\epsilon_{\hat{f}}}(x, x'; \theta_{\mathrm{M}})\right), \tag{5.13}$$

with the known or parametrically-unknown acceleration functions $\bar{a}_{\mathrm{K}}$, *e.g.*, being modeled as $\bar{a}_{\mathrm{K}} := M^{-1}(Q_{\mathrm{C}} + Q_{\mathrm{u}})$. In turn, an alternative formulation to (5.6) of a GP^2 assuming $\epsilon^{\star}_{\mathrm{A,A}} = 0$ is obtained as

$$\hat{\ddot{q}}_{\bar{a}} \sim \mathcal{GP}\left(Lb + T\bar{a}_{\mathrm{K}}, k_{\epsilon^{\star}_{\mathrm{A,A}}} + Tk_{\epsilon_{\hat{f}}}T^{\mathsf{T}}\right). \tag{5.14}$$

From the exposition of errors as in Section 4.1 it follows that if M is parametrically-unknown **the approximation of $\bar{a}$ through a GP is inferior to placing a GP prior on $\bar{Q}$ as in** (5.5). After all, the main source of approximation error is assumed to arise from unidentified force functions. Clearly, the approximation of these force errors *after* the transformation by M^{-1} introduces additional nonlinearities into the error function. The superiority of GP^2 using (5.5) compared to (5.14) has been validated empirically in *(Rath et al., 2021)*, and it is discussed in Section 5.3.

Nonetheless, the experiments on dynamic's identification using the GP^2 proposed in *(Geist and Trimpe, 2020)* grant clarity on the benefits and limitations of an implicitly-constrained GP. Subsequently, these experiments are detailed in the following, starting with an exposition of the mechanical system benchmarks.



(a) Mass particle sliding on a surface. (b) 2D Unicycle. (c) Duffing oscillator.

Figure 5.3: Implicitly constrained rigid body systems.
Figure adapted from *Geist and Trimpe (2020)*.

5.2.1 Mechanical System Benchmarks

In *(Geist and Trimpe, 2020)*, three rigid body dynamics systems were chosen as simple benchmarks for testing the GP^2 model as in (5.14). These systems present three types of constraints typically encountered in robotic systems while being particularly simple systems to ease the analysis of the learning result. The first system denotes a particle sliding over a surface constraint. The second system is a rigid body subject to a 2D wheel constraint, which in a sense is a drastically simplified version of the Wheelbot's dynamics as presented in Section 6.2.2. The last system introduces control forces into a coupled oscillator through holonomic constraints.

Particle on surface

A particle, as illustrated in Figure 5.3a with the known mass m slides along a surface. While Figure 5.3a depicts the constraint forces as a consequence of orthogonal projections. However, the constraint projection in the system's force space is only orthogonal if the system's mass matrix is diagonal.

While the dynamics of the particle are unknown, we want to leverage the surface geometry. The unconstrained acceleration of the particle amounts to $a = M^{-1}[u_1, u_2, u_3 - mg]^\mathsf{T}$ with $M^{-1} = \mathrm{diag}(1/m)$, $g = 9.81\frac{\mathrm{m}}{\mathrm{s}^2}$, and control forces u_i. The mass slides on the surface $q_3 = p_1 q_1^2 + p_2 q_2^2 + p_3 q_1 + p_4 \cos(p_5 q_1)$, with the states and constraint parameters being denoted as $\{q_i, \dot{q}_i\}$ and $\theta_\mathrm{A} = [p_1, ..., p_5]$. The second time-derivative of the constraint yields the constraining equation (3.30) as

$$\underbrace{\begin{bmatrix} 2p_1 q_1 + p_3 - p_4 p_5 \sin(p_5 q_1), & 2p_2 q_2, & -1 \end{bmatrix}}_{A(x,\theta_\mathrm{A})} \ddot{q} = \underbrace{\begin{bmatrix} -2p_1 \dot{q}_1^2 - 2p_2 \dot{q}_2^2 + p_4 p_5^2 \dot{q}_1^2 \cos(p_5 q_1) \end{bmatrix}}_{b(x,\theta_\mathrm{A})}, \tag{5.15}$$

In addition, a velocity quadratic damping force Q_z decelerates the mass non-ideally such that $\ddot{q}_{z,i} = -a_0(v^2/|v|)\dot{q}_i$, with the translatory velocity $v(x)$ and damping coefficient a_0. One obtains the system's constrained dynamics by inserting (5.15) as well as a and $\ddot{q}_z$ into (3.42). While θ_A and u_i can be readily measured and $\{A, b, M\}$ are obtained from a brief mechanical analysis, modeling Q and Q_z pose a considerable challenge for a plethora of

mechanical systems. Surface constraints are common implicit constraints that occur if a robot arm's end-effector or the feet of a quadrupedal robot interact with their environment.

2D Unicycle

The unicycle as depicted in Figure 5.3b commonly describes the motion of simple wheeled robots (Siciliano et al., 2010, p. 478). This example is described in further detail in (Udwadia and Kalaba, 2007, p. 213). Here, a non-holonomic constraint $\dot{q}_2/\dot{q}_1 = \tan(q_3)$ enforces that the system's translation velocity must point along the line C-G. After rearranging the nonholonomic constraint, its differentiation yields the implicit acceleration-level constraint as

$$A = \begin{bmatrix} \sin(q_3) & -\cos(q_3) & 0 \end{bmatrix}, \text{ and } b = -\dot{q}_1\dot{q}_3\cos(q_3) - \dot{q}_2\dot{q}_3\sin(q_3). \tag{5.16}$$

The inertia matrix and forces of the implicitly unconstrained system are derived as

$$M = \begin{bmatrix} m & 0 & -mR\sin(q_3) \\ 0 & m & mR\cos(q_3) \\ -mR\sin(q_3) & mR\cos(q_3) & Ic \end{bmatrix}, \; Q = \begin{bmatrix} mR\dot{q_3}^2\cos(q_3) + \cos(q_3)u_1, \\ mR\dot{q_3}^2\sin(q_3) + \sin(q_3)u_1, \\ u_2 \end{bmatrix}, \tag{5.17}$$

with R denoting the distance between the points C and G, and I_C being the system's inertia at G around the e_3 axis. Further, u_1 denotes a control input in direction of C-G and u_2 is a control input around q_3. The system is decelerated in the driving direction by Q_z. Q_z is induced by velocity quadratic damping.

Controlled Duffing's Oscillator

The Duffing's oscillator as depicted in Figure 5.3c models the behavior of two masses that are subject to cubic spring forces and liner damping. This system is described in further detail in (Udwadia and Kalaba, 2007, p. 120). An external control force imposes the constraint $q_2 = q_1 + p_1\exp(-p_2t)\sin(p_3t)$. Differentiation of the constraint yields the implicit acceleration-level constraint via

$$A = \begin{bmatrix} -1 & 1 \end{bmatrix}, \tag{5.18}$$

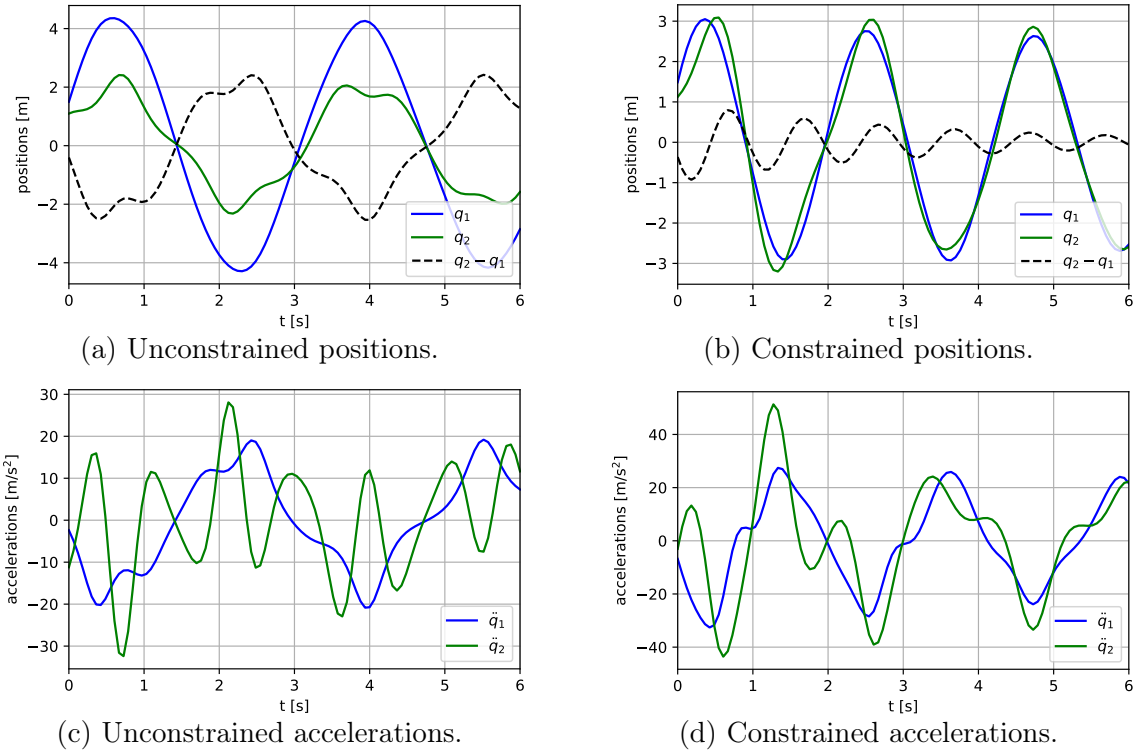
$$b = -p_1\exp(-p_2t)\left(p_3^2\sin(p_3t) + 2p_2p_3\cos(p_3t) - p_2^2\sin(p_3t)\right). \tag{5.19}$$

In turn, the control force arises as an implicit constraint force, which by enforcing the constraint ensures that over time the oscillation of the masses synchronizes as depicted in Figure 5.4. The uncontrolled dynamics are derived as

$$M = \begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix}, \text{ and } Q = K\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} + C\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} k_1^{nl}(q_1 - q_2)^3 \\ k_2^{nl}q_2^3 - k_1^{nl}(q_1 - q_2)^3 \end{bmatrix}, \tag{5.20}$$

where Q origins from spring and damping forces with

$$K = \begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 + k_2 \end{bmatrix}, \text{ and } C = \begin{bmatrix} c_1 & -c_1 \\ -c_1 & c_1 + c_2 \end{bmatrix}. \tag{5.21}$$



(a) Unconstrained positions.

(b) Constrained positions.

(c) Unconstrained accelerations.

(d) Constrained accelerations.

Figure 5.4: Duffing's oscillator's positions and accelerations plotted over time. Figure adapted from *Geist and Trimpe (2020)*.

5.2.2 Simulation Results

The benchmark systems detailed in Section 5.2.1 yield data that is used in the following to analyze the properties of the proposed GP^2 model. We compare the GP^2 to standard (multi-output) GPs.[1]

The system parameters are detailed in the supplementary material of *(Geist and Trimpe, 2020)*. The training data was generated using the system's analytic ODE and consists of randomly sampled observations lying inside the constrained state space fulfilling the respective position- (if the constraint is holonomic), velocity-, and acceleration-level constraints.

Predictions are made on an equidistant discretization of the constrained state space. The training data is normalized to have zero mean and unit standard deviation.

GP models

As a first baseline for model comparison, the individual $\ddot{q}_i$ are modeled independently as

$$\ddot{q}_i \sim \mathcal{GP}(0, k_{\text{SE}}(x, x')), \tag{5.22}$$

with squared exponential (SE) covariance function $k_{\text{SE}}(x, x')$. Further, we compare to a standard multi-output GP model, the (GPy, 2012) implementation of the LMC (Alvarez et al., 2012) with matrix

$$B_i = W_i W_i^{\mathsf{T}} + I_n \kappa, \; W_i \in \mathbb{R}^{n \times r}, \;\; \text{and} \;\; \kappa > 0, \tag{5.23}$$

[1]The simulation code is available on: `https://github.com/AndReGeist/gp_squared`

Table 5.1: Comparison of the normalized GPs' predicted mean RMSE, and maximum constraint error for 10 runs. For the RMSE, the mean, min. (subscript) and max. (superscript) values are shown. Table taken from *Geist and Trimpe (2020)*.

	RMSE			max. constraint error		
	Surface	Unicycle	Duffing	Surface	Unicycle	Duffing
Analy. ODE	—	—	—	$2{\cdot}10^{-15}$	$6{\cdot}10^{-17}$	$2{\cdot}10^{-14}$
SE	$.235_{.190}^{.272}$	$0.27_{0.20}^{0.40}$	$.011_{.004}^{.033}$	2.6	0.22	6.4
ICM	$.244_{.223}^{.277}$	$0.21_{0.15}^{0.27}$	$\mathbf{.003}_{.002}^{\mathbf{.005}}$	1.5	0.22	0.5
LMC	$.194_{.159}^{.233}$	$0.21_{0.15}^{0.28}$	$\mathbf{.003}_{\mathbf{.001}}^{.006}$	1.8	0.26	0.023
$\mathrm{GP}^2, \theta_\mathrm{A}{=}\theta_\mathrm{A}^*, \mu_{\bar{a}}{=}0$	$.058_{.045}^{.066}$	$0.10_{0.06}^{0.20}$	$.007_{.001}^{.019}$	$4{\cdot}10^{-12}$	$2{\cdot}10^{-12}$	$\mathbf{1{\cdot}10^{-8}}$
$\mathrm{GP}^2, \theta_\mathrm{A}{=}\theta_\mathrm{A}^*, \mu_{\bar{a}}{\neq}0$	$\mathbf{.023}_{\mathbf{.018}}^{\mathbf{.032}}$	$\mathbf{0.08}_{\mathbf{0.05}}^{\mathbf{0.15}}$	$.005_{.002}^{.029}$	$\mathbf{1{\cdot}10^{-13}}$	$\mathbf{6{\cdot}10^{-13}}$	$3{\cdot}10^{-8}$
GP^2, est. $\theta_\mathrm{A}, \mu_{\bar{a}}{=}0$	$.065_{.056}^{.071}$	$0.13_{\mathbf{0.05}}^{0.30}$	$.009_{.003}^{.028}$	0.12	$9{\cdot}10^{-13}$	0.013
GP^2, est. $\theta_\mathrm{A}, \mu_{\bar{a}}{\neq}0$	$.027_{.020}^{.037}$	$0.12_{0.06}^{0.20}$	$.020_{.004}^{.077}$	0.09	$9{\cdot}10^{-13}$	0.006

reading

$$K(x, x') = B_1 k_{\mathrm{SE}}(x, x') + B_2 k_{\mathrm{bias}}(x, x') + B_3 k_{\mathrm{linear}}(x, x'). \tag{5.24}$$

Inhere, k_{linear} and k_{bias} denote a linear and bias covariance function, respectively (Williams and Rasmussen, 2006). The model $K(x, x') = B_1 k_{\mathrm{SE}}(x, x')$ is referred to as the intrinsic model of coregionalization (ICM). The hyperparameters are optimized by maximum likelihood estimation via L-BFGS-b (Zhu et al., 1997). For the GP^2, we model $\bar{a} \sim \mathcal{GP}(0, k_{\mathrm{SE}}(x, x'))$, with the same optimization settings as for the other models.

For the GP^2 with "$\mu_{\bar{a}} \neq 0$", the analytical mean function $\bar{a}$ in (5.14) is set to a for the surface particle and 2D unicycle, while for the Duffing's oscillator $\bar{a}$ models the linear part of the acceleration induced by dampers and springs. Here, the spring and damping parameters θ_A are estimated alongside the other GP hyper-parameters θ_M.

Optimization and prediction

For each of the 10 optimization runs, 100 observations are sampled while the optimization is restarted 30 times for the benchmark GPs and five times for the GP^2 model. The prediction results after optimization are depicted in Table 5.1. For the surface particle and 2D unicycle, the GP^2 shows improved prediction accuracy. The performance can be further increased via the incorporation of additional structural knowledge in form of $\mu_{\bar{a}}$ ("$\mu_{\bar{a}} \neq 0$" in Table 5.1). If θ_A is estimated ("est. θ_A" in Table 5.1) the constraint error increases. Albeit, the constraint error is significantly smaller compared to the untransformed models. For the surface particle and Duffing's oscillator, θ_A was estimated accurately, whereas the 2D unicycle's parameters (I_c, R) converged to their correct ratio.

In the case of Duffing's oscillator, the implicit constraint force is a time-dependent control force. Unlike the unconstrained dynamics that show nonlinear oscillatory behavior, the constrained system dynamics move similarly to a single linearly damped oscillator. In this scenario, the GP^2 compares less favorably to the other models as it is learns the unconstrained dynamics. For all examples, the GP^2 demonstrates a considerable improvement regarding constraint satisfaction.

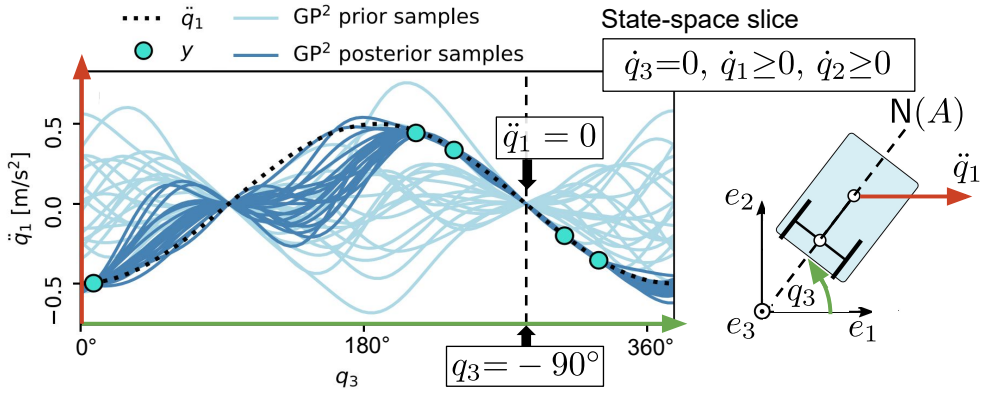


Figure 5.5: Samples of a GP^2 before and after conditioning on data of the 2D unicycle. Figure adapted from *Geist and Trimpe (2020)*.

Figure 5.5 illustrates how the GP^2 samples of $\ddot{q}_1(x)$ at $q_3 = 90\,\text{deg}$ and $q_3 = -90\,\text{deg}$ are forced to zero as the unicycle can only translate in driving direction.

Extrapolation

For an illustration of the prediction characteristics on the surface particle example, the constraint parameters are assumed as given, while θ_M is estimated on 200 observations. Figure 5.6a illustrates that the GP^2 model *extrapolates* the prediction result using the particle dynamics $\hat{h}$ at a velocity of zero such that $a_z = 0$. In this case, extrapolation is possible as $\bar{a}(X) = \bar{a}(x^*)$. That is the acceleration function $\bar{a}$ is the same at the training data and test data. This is not the case for velocity input dimensions when damping plays a dominant role. In comparison, after a certain distance, the SE-GP resorts back to the prior mean function, which has been chosen as zero.

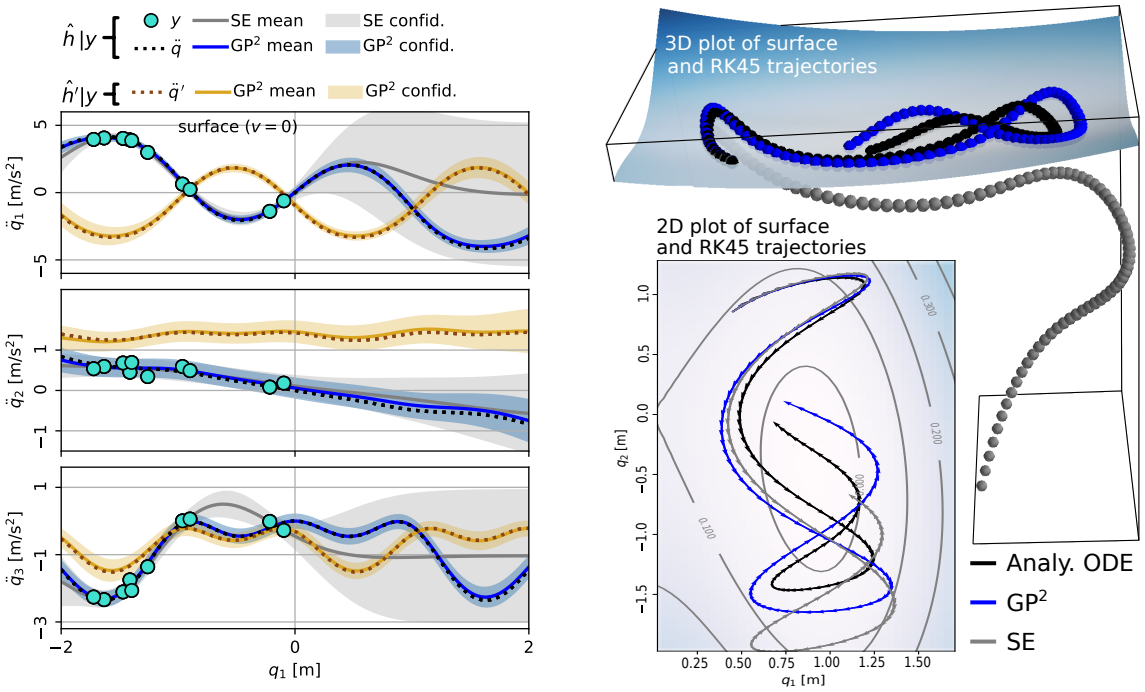
Transfer Learning

In many systems, altering the constraint configuration $\{A(x,\theta_p), b(x,\theta_p)\}$ to a different known configuration $\{A'(x,\theta'_p), b'(x,\theta'_p)\}$ does not change $\{\bar{a}, M\}$. For example, in the mass particle system, one could take the mass from one shape of surface to a different surface with the same tribological properties. In this case, it is possible to transfer knowledge in form of $\mathcal{D}$ from one system to a different system using the joint distribution

$$\begin{bmatrix}\hat{h}\\ \hat{h}'\end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix}\mu_{\hat{h}}(x|\theta_p)\\ \mu_{\hat{h}'}(x|\theta'_p)\end{bmatrix}, \begin{bmatrix}T(x|\theta_p)k_{\epsilon_{\hat{f}}}(x,x')T(x'|\theta_p)^\mathsf{T} & T(x|\theta_p)k_{\epsilon_{\hat{f}}}(x,x')T'(x'|\theta'_p)^\mathsf{T}\\ T'(x|\theta'_p)k_{\epsilon_{\hat{f}}}(x,x')T(x'|\theta_p)^\mathsf{T} & T'(x|\theta'_p)k_{\epsilon_{\hat{f}}}(x,x')T'(x'|\theta'_p)^\mathsf{T}\end{bmatrix}\right). \tag{5.25}$$

In other words, under the above assumptions, we condition a GP^2 modeling $\hat{h}'$ on data of $\hat{h}$ through (5.25). Figure (5.6a) illustrates that $\hat{h}'(x)$ resulting from a particle sliding over the surface $q_3 = 0.1q_1 - 0.15q_2 - 0.1\cos(3q_1)$ can be predicted by conditioning on observations $y(x)$ that stem from the structurally similar system $\hat{h}(x)$.

Trajectory prediction Figure 5.6b illustrates trajectory predictions of the surface particle computed by a Runke-Kutta-45 (RK45) ODE solver. The solver uses either the analytical dynamics model, the SE model, or the GP^2 model. While the SE trajectory



(a) (b)

Figure 5.6: Comparison of GP^2 predictions compared to a SE-GP on the surface particle system. *(a)* Given y from one surface (blue), predictions of $\ddot{q}$ on the same and $\ddot{q}'$ on another surface (yellow). *(b)* Trajectory prediction by integration of acceleration predictions using RK45.
Figures adapted from *Geist and Trimpe (2020)*.

prediction leaves the surface, the GP^2's prediction remains on the surface independent of the overall prediction performance. Here, the GP^2 predictions' Euclidean error to the surface increases in the same order of magnitude as with the analytical ODE.

It is not surprising that a standard GP FD model creates large errors when used for making trajectory predictions of an implicitly constrained system. After all, the GP is conditioned on data that stems from a constrained manifold inside the system's state-space. For example, for the surface particle, the positions lie on the surface, the velocities are tangent to this surface, and the acceleration fulfills the constraining equation. Even noisy observations remain close to this constrained manifold. When making trajectory predictions with a standard GP, the error in the predicted acceleration may cause the system's position to drift away from the values that have been observed in the training data. As we saw in Figure 5.6a, after a certain distance between training inputs and prediction inputs the SE-GP's predictions resorts back to its prior mean.

Inferring the Unconstrained Acceleration

As pointed out in Remark 5.1, the GP^2 framework enables the prediction of latent quantities that underlie the dynamics. For example, as the GP^2 results from a linear transformation of $\bar{a}$, the joint distribution is obtained as

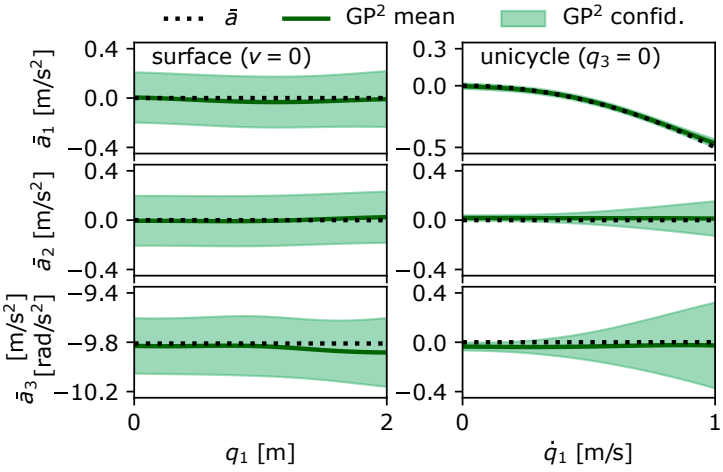


Figure 5.7: Prediction of $\bar{a}|y$. Figure adapted from *Geist and Trimpe (2020)*.

$$\begin{bmatrix} \hat{\bar{a}} \\ \hat{h} \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} \mu_{\bar{a}}(x, x') \\ \mu_{\hat{h}}(x, x') \end{bmatrix}, \begin{bmatrix} K_{\bar{a}}(x, x') & k_{\epsilon_{\hat{f}}}(x, x')T(x')^{\mathsf{T}} \\ T(x)K_{\bar{a}}(x, x') & T(x)k_{\epsilon_{\hat{f}}}(x, x')T(x')^{\mathsf{T}} \end{bmatrix} \right). \tag{5.26}$$

That is, $\bar{a}(x)$ can be inferred by conditioning it on observations of the system's acceleration y. Figure 5.5 shows the posterior distribution $\bar{a}|y$ using (5.26). For the surface particle with $v = 0$ and $u = 0$, $\bar{a}_3$ is simply $g = 9.81\frac{m}{s^2}$. For the 2D unicycle with $u = 0$, $\bar{a}$ solely consists of a damping force that increases with the translatory velocity.

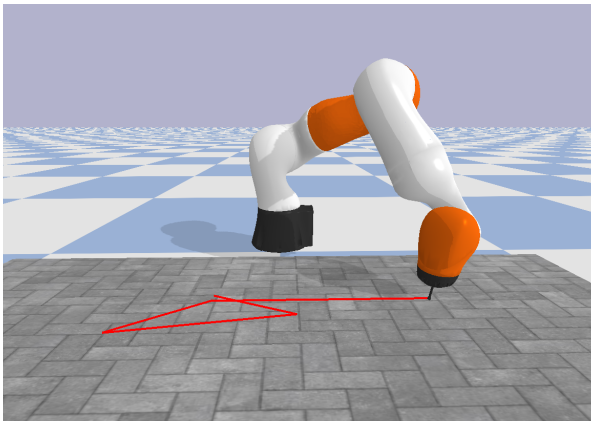
5.3 Learning Implicitly-constrained Forces

In this section, we address several shortcomings of the GP^2 framework as discussed in Section 5.2. In this regard, we propose a framework that allows for learning high-dimensional dynamics with GP^2. This framework is significantly faster compared to Section 5.2 by resorting to automatic differentiation with GPU accelerated computation of gradients and resorting to recursively computed rigid body dynamics. Notably, a GP prior is placed on the unknown forces of the system rather than its unconstrained acceleration, which eases the inclusion of prior knowledge on force properties into the GP. Through a series of simulation experiments, we show that this alternative approach improves the model's data efficiency compared to standard GP regression and GP^2 as in Section 5.2.

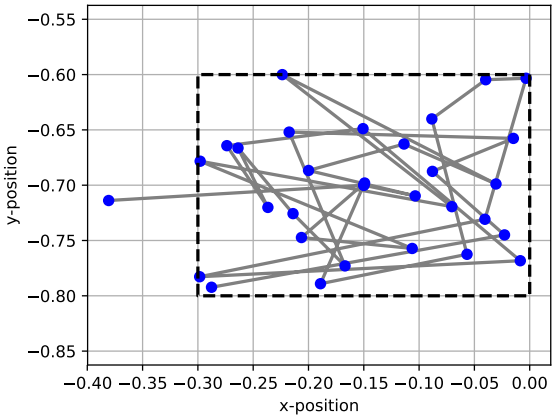
The GP^2 model as formulated in (5.6) is used throughout this section and has been derived at the beginning of Section 5.

5.3.1 Robot Arm Forward Dynamics Simulation

To explore the computational limits of structured GP regression a high-dimensional system, we chose a KUKA robot arm whose end-effector is in persistent contact with a surface as a benchmark. This robot arm possesses over seven rotational joints (as depicted in Figure 5.8a) and is simulated in "PyBullet" (Coumans and Bai, 2016–2021). With seven joints, the robot's FD model has 7 accelerations $\ddot{q}$ as outputs in which each process dimension depends on 21 inputs $\{q, \dot{q}, u\}$. Data is collected by controlling the robot's end-effector along linear trajectories while pressing onto the surface as depicted in Figure 5.8a. This task can be



(a) Robot arm in PyBullet.



(b) End-effector trajectory during the collection of training data.

Figure 5.8: Simulation of a robot arm whose end-effector is in contact with a surface. Figures adapted from *Rath et al. (2021)*.

seen as an abstraction of a robot arm performing welding, cutting, or marking maneuvers on surfaces. The contact between the end-effector and the surface creates ideal-constraint forces Q_{I} as depicted in Figure 5.1, which prevents the end-effector from penetrating the surface, as well as friction forces $\{Q_{\mathrm{D}}, Q_z\}$ which are challenging to be modeled beforehand.[2]

Dynamics Modeling

A key prerequisite for a fast implementation of GP^2 is the derivation of the robot's dynamics model using the same ML library used to optimize the model's (hyper-)parameters. In our case, the dynamics function and GP have been implemented in the "PyTorch" optimization library (Paszke et al., 2019). After all, the GP^2 model (5.6) requires the computation of the system's kinematics as well as the analytical functions M, Q_{K}, A, b. For example, with the robot arm having seven joints, just the calculation of the GP prior (5.5) over 1000 data points requires the computation of a $(7 \cdot 1000) \times (7 \cdot 1000)$ dimensional covariance matrix where every entry also requires the computation of analytical functions M, Q_{K}, A, b. Fortunately, the computational graph of the kinematics, dynamic terms, and subsequently the entries of the covariance matrix solely depend on the specific state inputs, such that the computation of these terms can be done in parallel.

The robot arm's FD has been derived based on (Featherstone, 2008, p. 119) and Remy (2019). The correctness of the PyBullet simulated dynamics and our PyTorch dynamics implementation is ensured by carefully comparing the simulated states, i.e., the position, velocity, and acceleration. Instead of relying on the contact dynamics model provided by PyBullet, we compute the surface normal force Q_I using our PyTorch dynamics library which is subsequently fed to the PyBullet simulation during data collection. Q_I is computed using Baumgarté Stabilization, as detailed in Section 5.3.4, to ensure that the end-effector does not leave the surface during the simulation due to small numerical errors. Moreover, we compute a viscous friction force F_z that is being applied to the end-effector, reading $F_z = -\theta_v \dot{p}_{\mathrm{E}}$ with the friction coefficient θ_v and the Cartesian end-effector velocity $\dot{p}_{\mathrm{E}}$.

[2]The simulation code of the proposed framework is available at: https://git.io/JP4Fs

Using our multibody library, Q_z is computed using F_z and subsequently integrated into the PyBullet simulation. The parameter of the friction function is set such that it significantly alters the motion of the robot arm. Without F_z, the analytical regression model matches the above-discussed analytical simulation model. In turn, the effect of F_z on the robot dynamics can be seen in the error between the analytical regression model and the simulated acceleration as depicted in Figure 5.11.

5.3.2 Data Generation

After setting up the simulation environments, running the experiment requires defining trajectory planning and control algorithms. While running the simulation, 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 (Qi et al., 2017) 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.

Trajectory planning

While the robot is sliding its end-effector over the surface, we want to collect informative data for training different regression models. Therefore, we uniformly sample the end-effector positions inside a rectangular region on the contact plane within the end-effector reachable space. We then coordinate the robot end-effector to sequentially connect the points following a straight-line trajectory, thereby ensuring that the end-effector remains in contact with the surface. Between each pair of points, a task-space trapezoidal velocity profile is generated for the end-effector position. At each point, the arriving and starting velocities are zero. In addition, this profile allows setting a travel time and a maximum task-space acceleration between points which is used to avoid exceeding the actuator limits. The generated end-effector task-space position, velocity, and acceleration trajectories are depicted in Figure 5.9. While the end-effector position trajectory is given by the trapezoidal profile, the desired orientation is set to a constant such that the rounded tip remains perpendicular to the surface.

Task space inverse dynamics control and nullspace control

To track the task-space reference trajectories with the robot arm, we use a task-space inverse dynamics controller (Siciliano et al., 2010, p. 347). The task of controlling a manipulator consists of finding a time history of the generalized control forces Q_u that are created by the actuator's control input u.

The robot arm has 7 degrees of freedom while we want to track a six-dimensional end-effector trajectory (3D position and 3 Euler angles). We can achieve the same reference with more than one configuration by changing the arm's elbow elevation. The question arises of how to set the extra DOF. In this work, we additionally use null-space control, which involves modification of the desired joint-space acceleration, such that the error dynamics remains untouched but constrain the acceleration through a secondary optimization objective. Here, this subspace is the null-space of a Jacobian matrix that follows from the robot's differential kinematics.

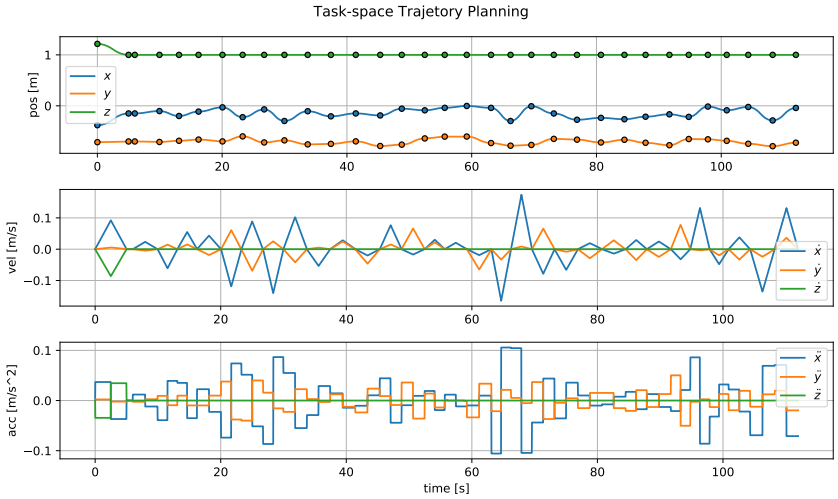


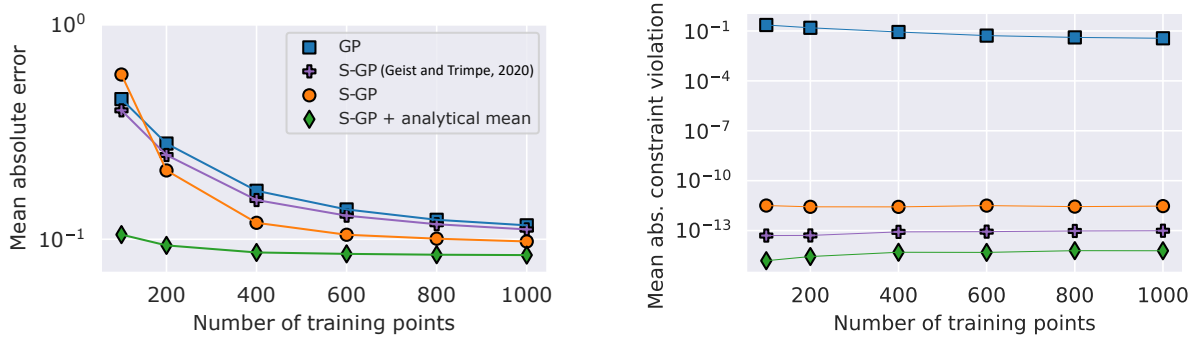
Figure 5.9: Task space trajectory of the robot arm's end-effector during data collection. Figure adapted from *Rath et al. (2021)*.

Data Selection

Sampling at 240 Hz generates a substantial amount of data points. During training, the computational complexity of GPs scales cubically with the number of data points which requires reducing the size of the initially large data set. Another problem arises when points in the training data set lie too close to each other inside the input space. This results in an almost singular covariance matrix potentially causing numerical problems during the computation of its Cholesky decomposition (Mohammadi et al., 2017). This problem can be avoided by selecting sufficiently distinct data set points to guarantee a numerically well-behaved covariance matrix. One practical way to select distinct data points is clustering or smart sampling techniques. In this work, a clustering algorithm divides the initial large amount of training points into clusters of similar data and only retains a single point per cluster. However, standard clustering techniques such as K-means clustering are computationally too slow for our data set size. Therefore, we use the Farthest Point Sampling algorithm (FPS), which has been used for Deep Learning with PointNet++ (Qi et al., 2017) for selecting a subset of relevant and informative points from a point cloud in a fast and efficient manner. The sampled points are chosen as the training dataset, while the removed points comprise the test dataset.

5.3.3 Simulation Results

In what follows, the proposed structured GP (S-GP) (5.6) is analyzed and compared to different baseline models. As a baseline for comparison, we trained the parameters of (3.42) 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 mean absolute error (MAE) of 0.57 and 0.13, respectively. For all GP models, we chose a squared exponential (SE) kernel. The 154 hyper-parameters of each GP were estimated according to (2.95) using Adam (Kingma and Ba, 2014b) without parameter constraints.



(a) (b)

Figure 5.10: The hyper-parameters of GP FD models are trained on an increasing number of training points, then the (a) mean absolute acceleration prediction error and the (b) acceleration-level constraint error on the test data set are compared. Figure adapted from *Rath et al. (2021)*.

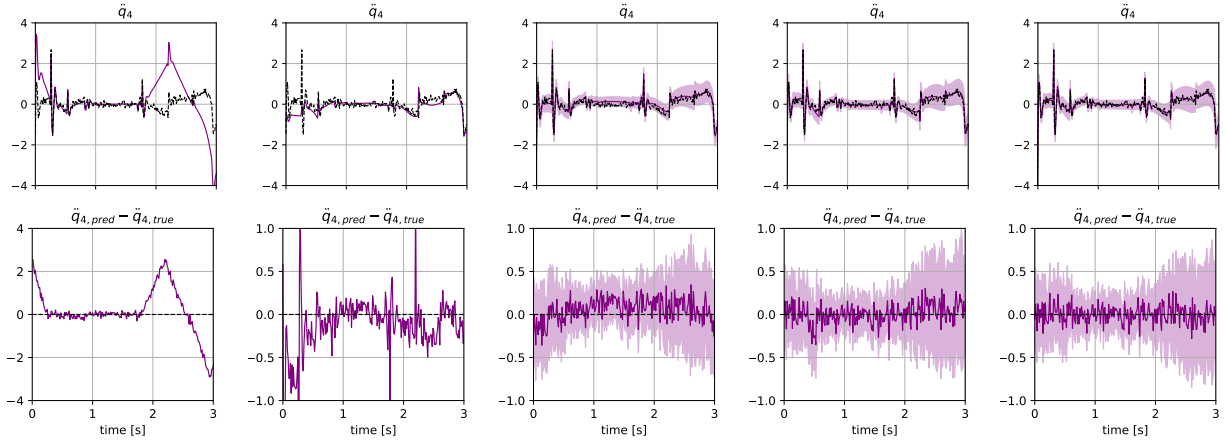
Prediction accuracy and data efficiency

Figure 5.10 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 (5.6) with $Q_{\mathrm{K}} = 0$ while the term "S-GP + analytical mean" assumes an analytical model $Q_{\mathrm{K}} = Q_G + Q_C + Q_u$ as the GP's prior mean function. For both S-GP models, we assume that accurate analytical parameters are given, that is $\theta_{\mathrm{A}} = \theta_{\mathrm{A}}^{\star}$. We assume $\{Q_{\mathrm{G}} + Q_{\mathrm{C}}\}$ as known as these analytical functions are being derived solely in terms of known kinematics and inertia functions and the gravitational acceleration constant *(Geist and Trimpe, 2021)*. The structured GP models are compared to standard GP regression in which each acceleration function is modeled by a single independent SE GP. As shown in Figure 5.10a, analytical prior knowledge improves the data efficiency of the GP. The proposed S-GP models compares also favorably to placing a GP prior on the system's unconstrained acceleration, as initially proposed by *Geist and Trimpe (2020)*. Moreover, the incorporation of implicit constraint knowledge in (5.6) significantly reduces the constraint error $\{A\hat{\ddot{q}} - b\}$ as illustrated in Figure 5.10b.

Figure 5.11 illustrates different acceleration predictions made with these models using 1000 training points. The analytical baseline model does not contain a function describing surface friction. This, in turn, causes the model to yield large prediction errors in the robot arm's joints that are close to the end-effector.

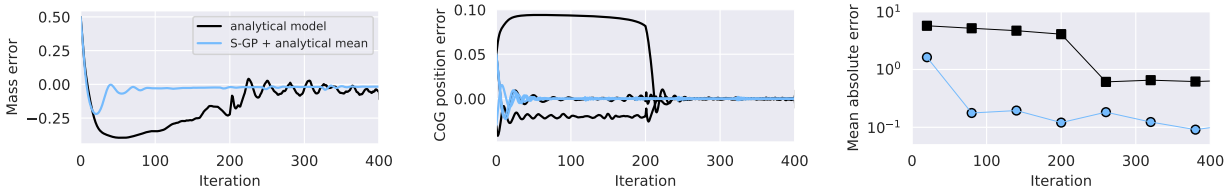
Estimation of Analytical Parameters

Another important aspect of the proposed structured modeling framework forms the simultaneous estimation of θ_{A} and θ_{M}. To illustrate how θ_{A} and θ_{M} are estimated jointly, we train the same S-GP with $Q_{\mathrm{K}} = 0$ as in the previous simulation 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 5.12 illustrates the optimization results.



(a) Analytical dynamics model (b) Feed-forward NN (c) GP (d) S-GP with zero mean (e) S-GP with analytical mean

Figure 5.11: Acceleration predictions $\ddot{q}_{4,\text{pred}}$ (Top) and prediction error (Bottom) of different forward dynamics models with the noisy acceleration over time (Black), the predicted mean acceleration (purple) and – if available – the ± 2 std. deviation confidence region (light purple). The y-axis scaling of Figure 5.11a deviates from the other figures. Figure adapted from *Rath et al. (2021)*.



(a) (b) (c)

Figure 5.12: Parameter optimization results of an analytical model (black) and the proposed structured GP model (blue). Figure adapted from *Rath et al. (2021)*.

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.

Prediction of Lagrange Multipliers

With the joint distributions between the GP^2's latent processes (5.11), one can predict the model's Lagrange multipliers λ at inputs $X^\star$ by conditioning on acceleration measurements at inputs X using the conditioning formula (2.84), writing

$$\hat{\lambda}(X^*)|\mathcal{D},\theta = \mu^{\text{b}}_{X^*} + \Sigma^{\text{ba}}_{X^*,X}\left(\Sigma^{\text{aa}}_{X,X} + \Sigma_Y\right)^{-1}\left(Y - \mu^{\text{a}}_X\right), \tag{5.27}$$

with $\mu^{\text{b}} = (AM^{-1}A^\mathsf{T})^{+}b - LQ_K$, $\mu^{\text{a}} = M^{-1}Q_b + PQ_{\text{K}}$, $\Sigma^{\text{ba}} = L(X^*)\Sigma^{\bar{Q}}_{X^*,X}P^\mathsf{T}(X)$ and $\Sigma^{\text{aa}}_{X,X} = P(X)k_{\bar{Q}}(X,X)P^\mathsf{T}(X)$. Figure 5.13 illustrates the prediction of λ by conditioning on $\mathcal{D}$ via (5.27) using the proposed structured GP model. As expected, the Lagrange

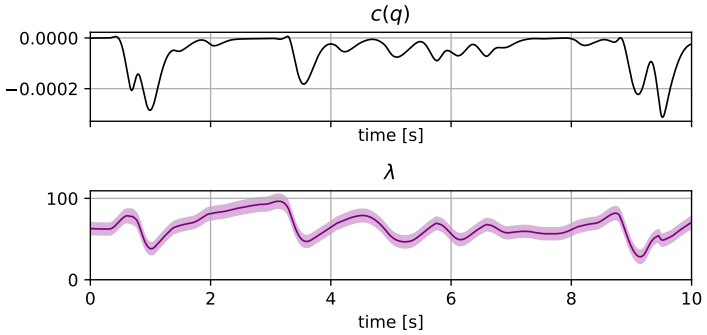


Figure 5.13: Top: Surface equation error in simulation. Bottom: Prediction of Lagrange multiplier λ by conditioning a GP^2 (5.6) on acceleration measurements. Figure adapted from *Rath et al. (2021)*.

multiplier remains positive such that the end-effector is impressing a positive force on the surface. The prediction of a loss of contact with structured GPs is left to future work. The estimation of the Lagrange multipliers as well as the surface normal force Q_I is useful for impedance control as commonly used in legged robotics.

5.3.4 Baumgarte Stabilization for Trajectory Predictions

Compared to differential-algebraic formulations of the EOM, the EOM (3.42) has the advantage that numerical ODE solvers can be used to compute a trajectory given initial conditions $\{q_0, \dot{q}_0\}$. Moreover, it is particularly straightforward to combine data-driven modeling with analytical equations. Yet, (3.42) only respects the acceleration-level constraints explicitly. In turn, a trajectory prediction in the absence of measurement noise only lies on the position-level constraint $c(q) = 0$ if the initial state $\{q_0, \dot{q}_0\}$ lies on the position-level constraint. In practice, a numerical integration method inevitably makes errors on the system's acceleration which causes the predicted position and velocity to drift. This integration drift is additionally aggravated by the noise in the state observations.

To ensure that trajectory predictions converge to the position-level surface constraint, one can resort to Baumgarte stabilization.

Baumgarte stabilization forms a common technique in multi-body dynamics to ensure that the trajectory predictions made with an ODE formulation of the EOM fulfill implicit position level-constraints. In what follows, we assume the implicit constraint equations $c(q)$ are holonomic with $\frac{dc(q)}{dt} = A\dot{q} = 0$. Given a measured/predicted state-acceleration pair $\{q', \dot{q}', t, \ddot{q}'\}$, the error made in the position-, velocity-, and acceleration-level constraints amounts to

$$e_c = c(q'), \tag{5.28}$$
$$e_{\dot{c}} = A(q')\dot{q}', \tag{5.29}$$
$$e_{\ddot{c}} = A(q')\ddot{q}' - b(q', \dot{q}'). \tag{5.30}$$

In this case, Baumgarte (1972) suggests extending the error dynamics to yield a stable damped oscillator equation for e_c as

$$e_{\ddot{c}} + 2\omega_{n,\ddot{q}}\xi_{n,\ddot{q}}\dot{e}_c + \omega_{n,\ddot{q}}^2 e_c = 0, \tag{5.31}$$

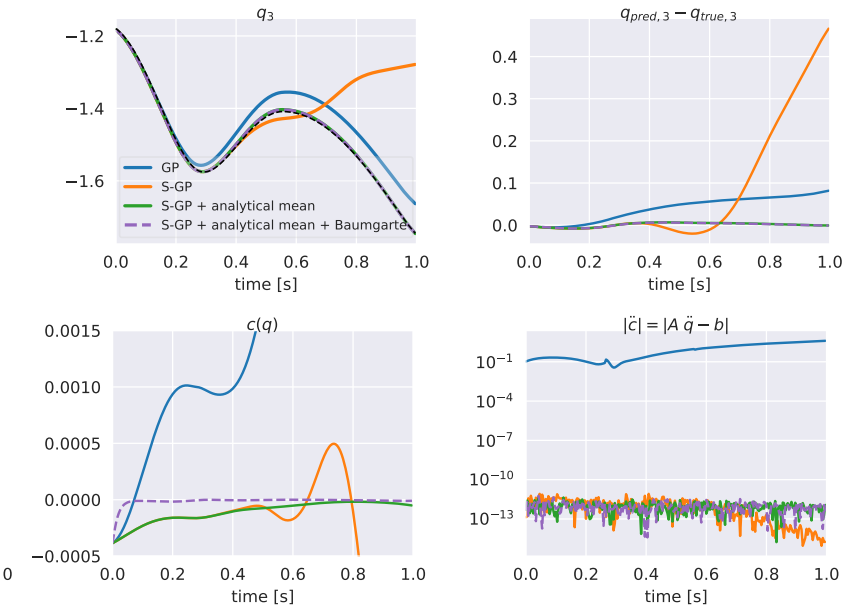


Figure 5.14: Long-term trajectory prediction and corresponding error functions obtained by numerical integration with an integration time step of 0.0024 s using the different GP models' acceleration predictions of the fourth output dimension. The true trajectory is depicted by the black dotted line. Figure adapted from *Rath et al. (2021)*.

with the natural frequency $\omega_{n,\ddot{q}} > 0$ and damping ratio $\xi_{n,\ddot{q}} > 0$. Usually $\xi_{n,\ddot{q}}$ is chosen to be unitary, such that the constraint error dynamics are critically damped. By inserting (5.28), (5.29), and (5.30) into (5.31) one obtains

$$A\ddot{q}' = \tilde{b}, \tag{5.32}$$

with $\tilde{b} = b + 2\omega_{n,\ddot{q}}\xi_{n,\ddot{q}}A\dot{q}' + \omega_{n,\ddot{q}}^2 e_{\ddot{q}}c(q')$.

Remark 5.2. *Substituting b in (3.42) with $\tilde{b}$ ensures that for long trajectory predictions, e_c converges to zero at the cost of introducing a small error to the predictions of $\ddot{q}$.*

Figure 4 illustrates trajectory predictions for one of the joint dimensions which are computed by resorting to symplectic Euler numerical integration, using the different GP models. The different GP models are the same as in Figure 5.10a using 1000 training points. Due to measurement noise in the initial state as well as numerical integration and prediction errors, the GPs diverge from the true trajectory over time. As a consequence of Assumptions 1 and 2, one can guarantee that trajectory predictions computed with (5.6) converge onto the surface equation $c(q) = 0$ by replacing b inside the GP^2 with $\tilde{b}$ as (5.32). In turn, the position-level surface constraint error converges to zero for the GP models using Baumgarte stabilization. Here, only in the prior mean at the prediction locations $\{L(X^\star)b + Pm_{\bar{Q}}(X^\star, \theta_\mathrm{A})\}$ and the function $P(x^\star, \theta_\mathrm{A})$ inside the covariance function $k_{\ddot{q}}(x^\star, x) = P(x^\star, \theta_\mathrm{A})k_{e_{\hat{f}}}(x^\star, x)P(x, \theta_\mathrm{A})$ the function b must be replaced by $\tilde{b}$. Figure 5.14 illustrates how the usage of Baumgarte stabilization ensures that the trajectory prediction of the robot's third joint respects the surface constraint $c(q) = 0$.

Chapter 6

The Wheelbot: Developing a Robot Testbed for Learning Control

The previous discussion of a unified view on physics-informed regression has been decisively influenced by the development of a novel robotic testbed named "Wheelbot". We developed this robot in parallel to conducting the research on physics-informed regression as presented in Chapter 3 and Chapter 5. Before we delved into the depth of mechatronics system engineering, little we knew on the sacrifice it takes to develop the first jumping unicycle robot. Yet, by venturing deep into the abyss of real mechatronic system design, we were able to cast a glance at typical challenges of robotics engineering and how to ease some of those using physics-informed regression. Albeit, we see the purpose of this robot as a relatively affordable platform for education and research on nonlinear dynamics and control. This section is based on *Geist et al. (2022)*[1].

Recent advancements in actuators, sensors, and embedded controllers, saw the explosion of low-cost 3D rapid prototyping; have enabled the development of a wide variety of novel robotic testbeds. Such systems enrich the controls and robotics community by exposing unique compositions of dynamical properties while also providing a platform for the exploration of novel mechatronic solutions.

When contemplating the architecture of a robotic testbed, the designer first encounters several system-level design decisions, including the number of degrees of freedom (DOF) and the layout of the mechanical and electrical components. While a variety of interesting arrangements exist in such a large design space, this work is specifically focused on the realm of nonlinearly-coupled, under-actuated systems, wherein the number of actuators is less than the DOF. A textbook example of such a system is the "cart-pole pendulum", with only two DOF (the position and pendulum angle) and one actuator controlling the cart's position. The cart-pole pendulum's low-dimensional dynamics eases the analysis of experimental data, while its simple design reduces cost and maintenance. Yet, for systems with more DOF, interesting questions arise on how to identify and leverage coupling terms for control. In what follows, we propose a control testbed that offers challenging under-actuated dynamics with interesting dynamical properties and unprecedented control capabilities while also being compact and relatively low-cost.

[1] IEEE remains the rights holder for all text excerpts and figures which have been taken from *Geist et al. (2022)*.

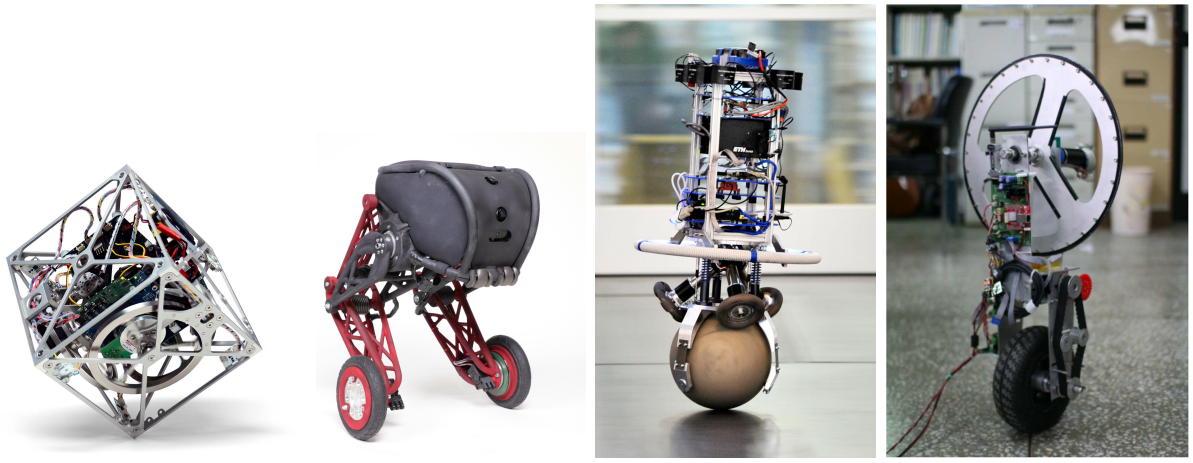


Figure 6.1: The design of the Wheelbot is inspired by several robotic testbeds. From left to right: The "Cubli" balancing cube (Muehlebach and D'Andrea, 2018), the "Ascento" robot (Klemm et al., 2019), the "Rezero" ballbot of the ETH Zürich (Hertig et al., 2013), and an orthogonal-configuration unicycle robot of the Pusan National University (Lee et al., 2013).

Self-balancing Wheeled Testbeds

Typically utilizing off-the-shelf electric motors, wheeled robots are interesting control testbeds that usually require a small operating space while also providing a wide range of different dynamical properties. Figure 6.1 illustrates a selection of robotic testbeds that decisively influenced the design of the Wheelbot. The wheels of such robots can be distinguished into rolling wheels and reaction wheels. Rolling wheels leverage friction forces for locomotion and are found on Ballbots Fankhauser and Gwerder (2010); Nagarajan et al. (2014) as well as Segway-esque robots such as the Ascento Klemm et al. (2019). Reaction wheels apply free torques and are used in simple wheeled pendulums such as Rouleau (2015) and more complex cube-like structures such as Gajamohan et al. (2013).

From a dynamics perspective, balancing with a reaction wheel requires fast changes in the motor's direction of rotation and may also evoke high motor velocities. In turn, the actuation of a reaction wheel with an electric motor summons rate-dependent control constraints as the maximum torque produced by an electric motor *in the direction of rotation* is inversely proportional to its rotational speed. In comparison, rolling wheels rotate at comparably lower speeds and may introduce nonholonomic kinematic constraints as in the case of the Ascento Klemm et al. (2019). Moreover, using rolling wheels may require the linearized closed-loop system dynamics to be non-minimum phase as in the case of Fankhauser and Gwerder (2010); Klemm et al. (2019); Nagarajan et al. (2014).

Brushless electric motors have enabled wheeled robots such as the Cubli (Gajamohan et al., 2013; Muehlebach and D'Andrea, 2017) and Ascento (Klemm et al., 2019) to perform fast-changing maneuvers that are subject to discontinuous contact dynamics. Since such dynamics are difficult to model a priori, linear control algorithms tend to perform below expectations, motivating the usage of different control strategies that either identify modeling errors from data or incorporate probabilistic nonlinear error functions into the control design.

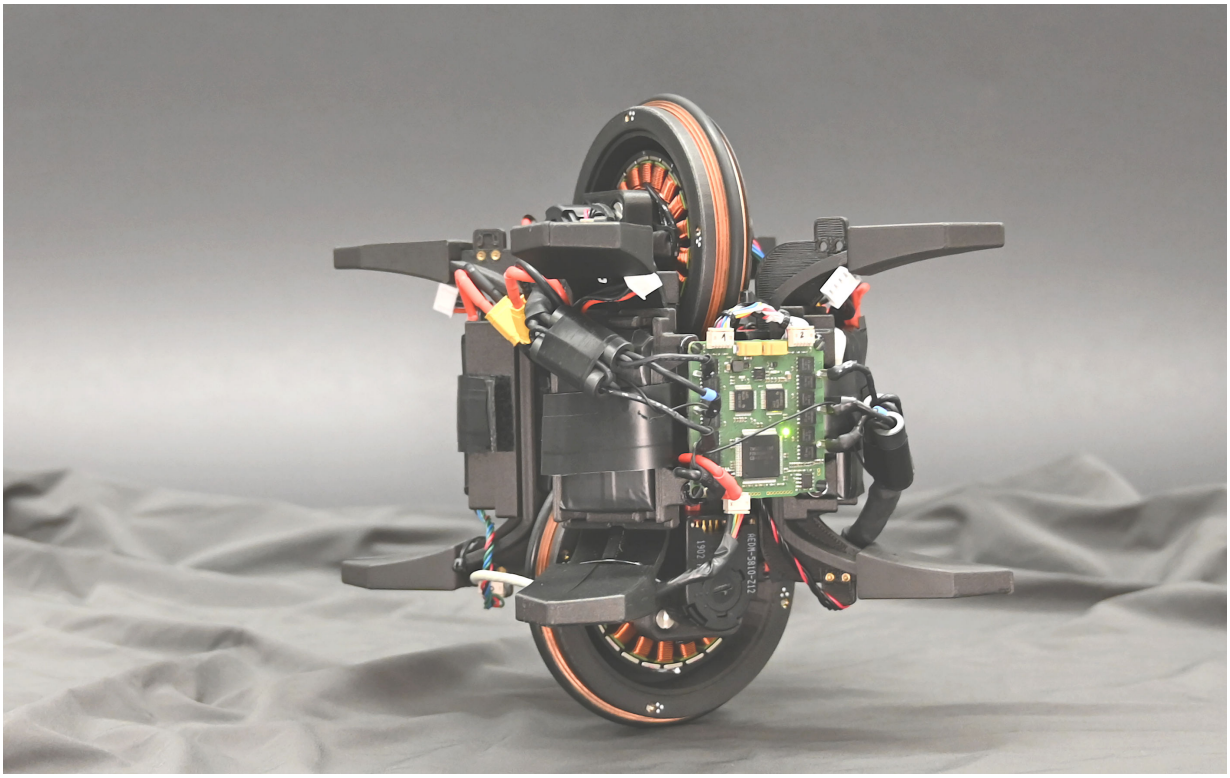


Figure 6.2: The Wheelbot is a reaction wheel-driven unicycle robot that uses brushless motors to self-erect after toppling.[2]

The Wheelbot Concept

The Wheelbot arose from the desire for a compact, under-actuated control testbed with nonholonomic and fast-changing discontinuous dynamics that could be operated in a relatively small space. To meet these objectives, the small unicycle robot shown in Fig. 6.2 was designed to include two actuated wheels attached to a rigid body, with one rolling on the ground and the other acting as a reaction wheel. A special feature of the robot is its symmetry in the sense that both wheels can act as a reaction wheel or a rolling wheel.

Previous literature proposes reaction wheel unicycle robots whose rotation axis is either coaxial to a line connecting the center of both wheels as in Deisenroth (2010); Rizal et al. (2015); Vos (1992); or it is orthogonal to such a line as in Lee et al. (2013); Rizal et al. (2015); Rosyidi et al. (2016). Unicycle robots with coaxially-oriented reaction wheels do not directly actuate the unstable roll DOF but instead turn the robot in the tilting direction while the rolling wheel prevents the robot from toppling. In contrast, an orthogonal-configuration unicycle directly actuates the roll DOF.

When designing the Wheelbot, we opted for an orthogonal configuration to directly control the unstable roll and pitch DOF. As the roll, pitch, and yaw dynamics are decoupled when linearized around the upright equilibrium, an orthogonal configuration allows one to tune the roll and pitch balancing controllers independently from each other. These controllers then act as a starting point for tackling more challenging research questions that revolve around the identification and control of the robot's yaw dynamics or, as a subsequent step, the control of the robot along a desired trajectory. Recently proposed

[2]A video of the Wheelbot is available at: `https://youtu.be/4XLB8JoPpZ8`

Table 6.1: System overview. Table taken from *Geist et al. (2022)*

Category	Value
Overall weight	1.4 kg
Wheel weight	0.24 g
Operating time	∼20 min
Max. battery voltage	25.2 V
Battery capacity	1.3 Ah
Max. motor current	19 A
Max. motor torque	∼1.3 Nm
Max. motor rate	300 rad/s

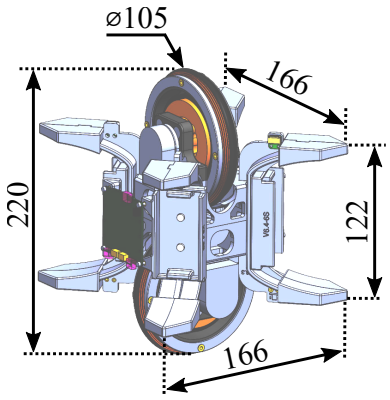


Figure 6.3: Dimensions (mm). Figure taken from *Geist et al. (2022)*.

designs for orthogonal-configuration unicycles (Lee et al., 2013; Rizal et al., 2015; Rosyidi et al., 2016) resort to large, high-inertia reaction wheels to reduce wheel acceleration and in turn avoid exposing the system's electronics to larger currents at faster wheel speeds. Yet, to be able to self-erect from any initial pose, the reaction wheel of the Wheelbot need to be considerably smaller than those of the aforementioned orthogonal-configuration unicycles, which led to a number of interesting design challenges. To the best of our knowledge, the Wheelbot forms the first unicycle robot that can self-erect from any initial position and is able to recover from significant disturbances around its roll and pitch axis.

6.1 System Description

The detailed design of the Wheelbot began with several high-level architectural decisions:

1. The outer geometry of the robot should allow it to self-erect from any initial position.
2. The robot's bottom and top half should be identically constructed, thereby reducing the number and complexity of the 3D-printed parts.
3. The robot should carry its own power supply.
4. The control of the robot should be shared between an onboard embedded processor and a wirelessly-connected supervisory controller.

The realization of these points required solving numerous mechanical, electrical, and programmatic challenges.

6.1.1 Mechanical Design

To build a safe, easy-to-maintain, and reasonably low-cost testbed, the mechanical design of the Wheelbot focused on the combination of off-the-shelf components with simple 3D-printable parts. The robot consists of two identical wheel assemblies mounted to a center frame. The majority of the mechanical components were created from Onyx ABS plastic,

Table 6.2: Component specifications.
Table taken from *Geist et al. (2022)*.

Component	Specification
Motors	T-Motor Antigravity 6007 KV160
Motor-controller	uDriver v2
Additional μController	Maevarm M2 (ATmega32U4 processor)
IMUs	TDK-Invensense ICM-20948 9-DOF
Encoders	Avago Technologies AEDM5810Z12
Batteries	LiPo - 11.1 V (3S), 650 mAh, 75 C

using a Markforged OnyxOne 3D printer. Threaded brass inserts are heat impressed into the 3D printed parts such that screws can be removed without wear and tear.

Center frame The center assembly consists of a 3D-printed chassis sheltering microcontrollers, four batteries, cables, four inertia measurement units (IMUs), and two wheel assemblies. The batteries are placed symmetrically with respect to the x and y axis of the body-fixed frame since they contribute considerably to the system's overall weight and inertia. As discussed in 6.4.2, the cubic design of the chassis allows the system to self-erect from any initial position. The width and height of the chassis cube structure are provided in Fig. 6.3.

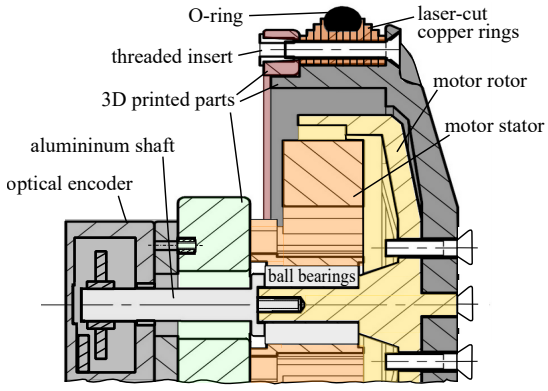


Figure 6.4: Sectional drawing of the wheel assembly.
Figure taken from *Geist et al. (2022)*.

Wheel assembly As depicted in Fig. 6.4, each wheel assembly consists of several 3D-printed parts, a reaction wheel, a brushless electric motor, and an optical encoder. Since the reaction wheel's mass and inertia determine the torque required for a stand-up maneuver as well as the time constant of the system's roll dynamics, the design focuses on maximizing the rotational inertia while keeping the mass as low as possible. To reduce the reaction wheel's mass while maximizing its inertia, we laser-cut 1 mm thick copper rings, which are then stacked onto a 3D printed hub attached to the reaction wheel's motor, as depicted in Fig. 6.4, yielding a 320 gram wheel with a rotational moment of inertia of $5 \cdot 10^{-4}$ kg-m^2. A rubber O-ring (75 mm inside diameter, 5 mm cross-section diameter) was fitted to the outer rim of the wheel to increase grip when touching the ground.

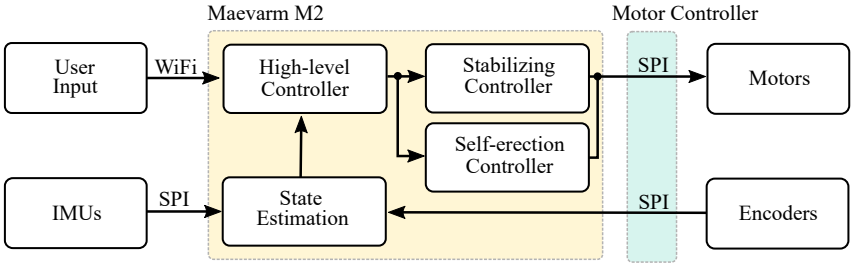


Figure 6.5: Overview of the software architecture.
Figure taken from *Geist et al. (2022)*.

6.1.2 Electronics Design

Table 6.1 gives a general system overview, while Table 6.2 provides a summary of the chosen electronic components. Based on the mass properties obtained from the robot's CAD model, we utilized the simulation detailed in Section 6.2.2 to determine the required motor torque for performing a stand-up maneuver. The chosen brushless motor has a stall torque of 1.3 Nm and maximum speed under load of 300 rad/s (2860 RPM) when operated at 24.5 V. Power is supplied by four 12.6 V, 650 mAh Lithium-Polymer (LiPo) batteries arranged as two parallel pairs of serially connected packs to provide a nominal fully-charged voltage of 25.2 V.

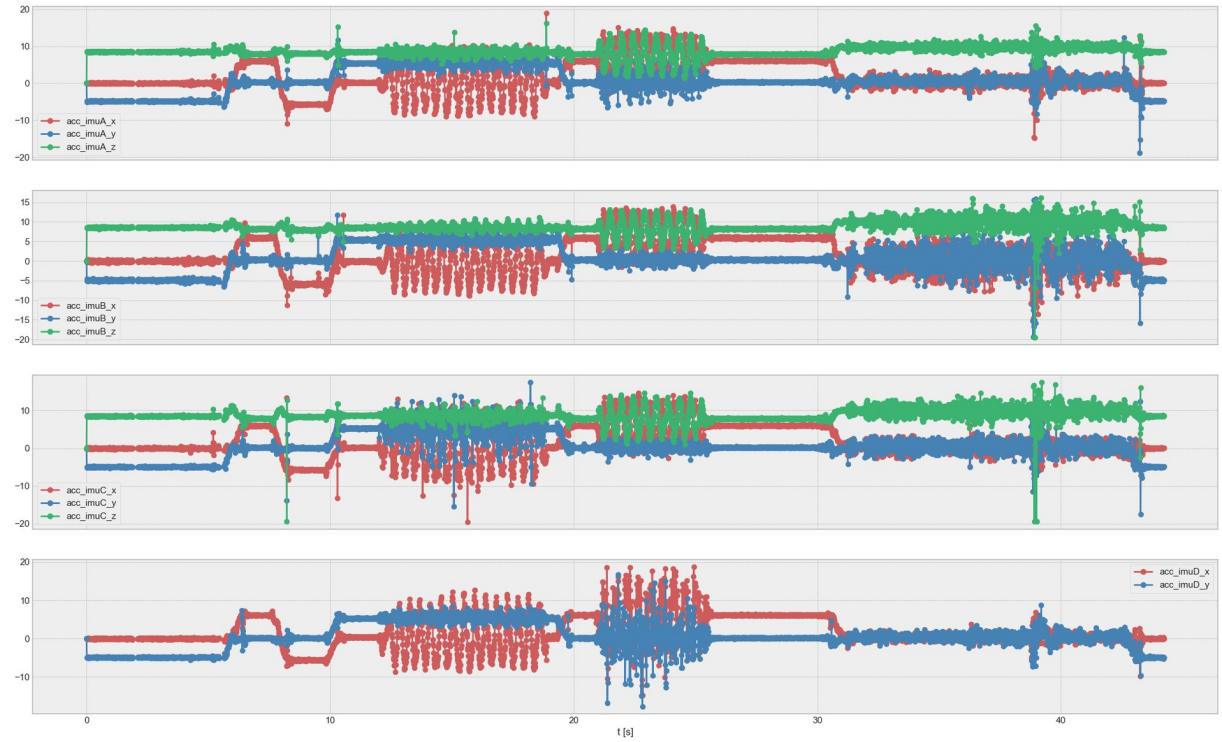
The motors are controlled by a "μDriver v.2", a compact dual-channel brushless-motor controller based on the TI TMS320F28069 micro-controller and a pair of TI DRV8305 smart three-phase gate drivers. The μDriver v.2 can operate up to a maximum of 44 V while delivering up to 30 A per channel. Motor commands and feedback signals are transmitted between the μDriver and a compact 16-MHz ATmega32U4-based MAEVARM M2 microcontroller via a high-speed Serial-Peripheral Interface (SPI) connection.

The SPI interface of the M2 is also used to collect data from four TDK-Invensense ICM-20948 9-DOF inertial measurement units (IMUs) attached to four corners of the cube structure. Each IMU provides the triaxial acceleration, angular rate, and magnetic field measurements, though in the present configuration, the magnetometer readings are not used due to high noise from the nearby brushless motors. To maximize sensitivity while avoiding saturation, the range of the accelerometers is set to $\pm 2g_0$ where g_0 denotes the gravitational acceleration constant $g_0 \approx 9.81\,\mathrm{m/s^2}$, while the gyroscopes are set to 500 deg/s. Figure 6.6 shows typical measurements of the IMUs. As illustrated in Fig. 6.5, the robot's state estimator and controllers are executed on the M2, which is also responsible for the receipt of user inputs via a Nordic nRF25LE1 2.4-GHz wireless link.

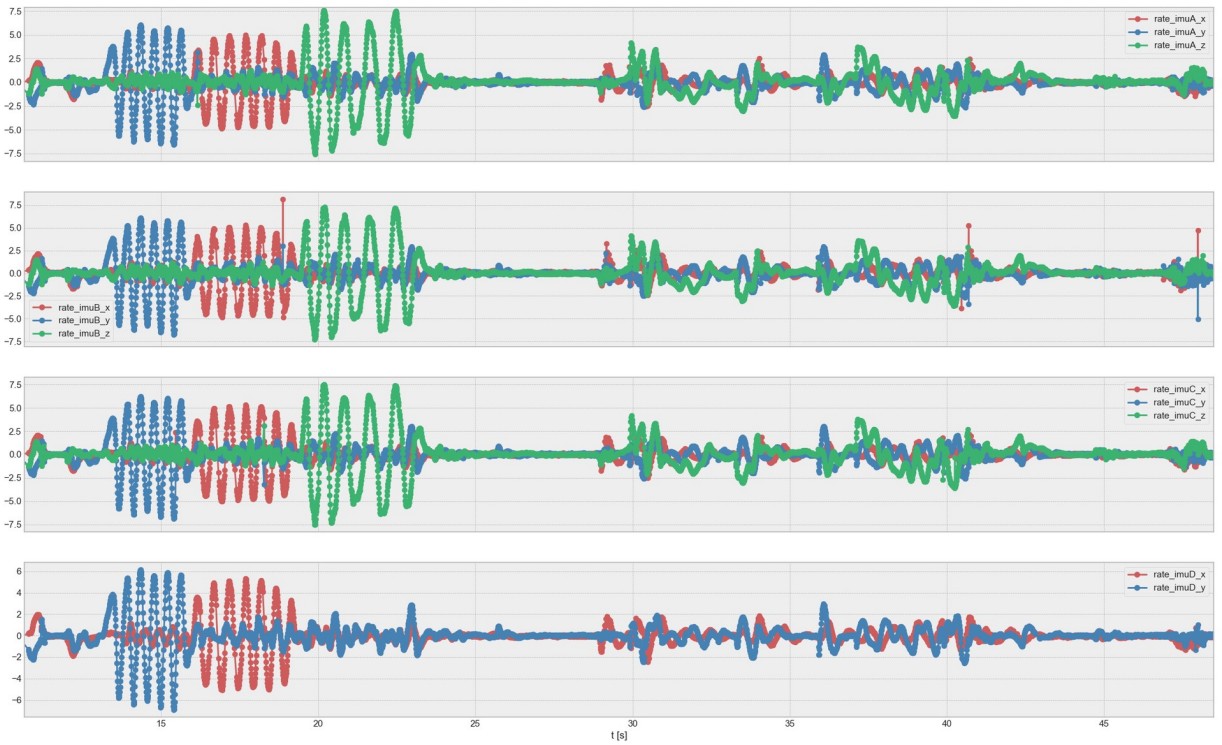
The use of brushless motors in such a compact design comes with the risk of significant Electro-Magentic Inference (EMI). To reduce the EMI emission, the motor cables are twisted and wrapped around ferrite rings to reduce high-frequency current oscillations. In addition, all SPI communication cables are shielded with a grounded copper mesh and located as far from the motor cables as possible.

6.1.3 Software Design

The embedded software for both the M2 and the motor driver is written in the "C" programming language. Running at a fixed loop rate of 100 Hz, the M2 receives user inputs



(a) Accelerometer data.



(b) Gyro data.

Figure 6.6: Data of the Wheelbot's four tri-axis IMUs. **Left:** Data is collected while the motor is switched off and the Wheelbot is rotated by hand. **Right:** Data is collected while the robot is balancing. The accelerometer and gyro data stem from two separate experiments.

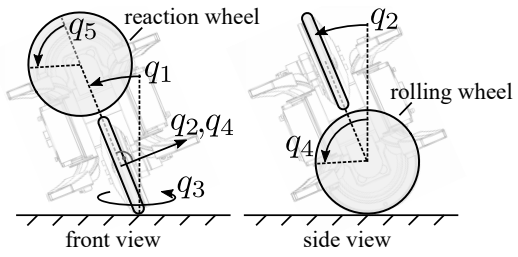


Figure 6.7: Generalized coordinates describing the system's pose.
Figure taken from *Geist et al. (2022)*.

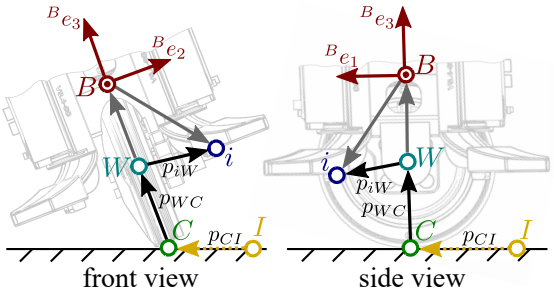


Figure 6.8: The locations of the robot's IMUs is described in terms of the coordinate systems $\{I\}$, $\{C\}$, $\{W\}$, and $\{B\}$.
Figure taken from *Geist et al. (2022)*.

over the wireless link, reads data from the four IMUs and two encoders, executes the state estimator and control routines, and outputs desired current values to the μDriver. The Wheelbot's user inputs and data are processed on a PC using the "Python" programming language via the "Tkinter" package.

6.2 Dynamics Modeling

Data-driven dynamics identification is the raison d'etre of this work. Yet, before delving into some of the fundamental principles underlying the dynamics of robotic systems in Section 3, we first approach robot dynamics modeling from a practitioner's point of view. In the concrete example of the Wheelbot, kinematic and dynamic models may be possibly used to:

- Understand the system's physical properties and test designs during the development phase.
- Obtain more accurate estimates of the system's state by incorporating the mechanical prior knowledge into an observer design.
- Provide a simulation environment for the development of state estimation and control routines before facing additional challenges that accompany embedded system design.
- Provide a model for a predictive controller to improve control performance.

6.2.1 System Coordinates

The system's pose is described via the generalized coordinates $q \in \mathbb{R}^5$ with $q = [q_1, ..., q_5]^\mathsf{T}$, the generalized velocity by $\dot{q} = \frac{dq}{dt}$, and the generalized acceleration as $\ddot{q} = \frac{d^2q}{dt^2}$. The body-fixed coordinate system is denoted as $\{B\}$ with coordinate axes $\{e_1^B, e_2^B, e_3^B\}$ and origin B. As illustrated in Fig. 6.7 and Fig. 6.8, the system coordinates are:

- *Contact point positions* x and y: The Cartesian positions of $\{C\}$'s origin with respect to the inertial frame $\{I\}$.

- *Roll angle* q_1: A rotation of $\{W\}$ around e_1^C.
- *Pitch angle* q_2: A rotation of $\{B\}$ around e_2^W.
- *Yaw angle* q_3: A rotation of $\{C\}$ around e_3^I.
- *Rolling wheel angle* q_4: Describing the rolling wheel's rotation around e_2^W.
- *Reaction wheel angle* q_5: Describing the reaction wheel's rotation around e_1^B.

A vector is transformed from the inertial frame $\{I\}$ to the body-fixed frame $\{B\}$ by a yaw-roll-pitch Euler rotation sequence, reading

$$R_{BI}(q_1, q_2, q_3) = R_2^\mathsf{T}(q_2) R_1^\mathsf{T}(q_1) R_3^\mathsf{T}(q_3). \tag{6.1}$$

The rotation from $\{B\}$ to $\{I\}$ is given as $R_{IB} = R_{BI}^\mathsf{T}$. With (6.1) one obtains kinematic expressions for the position and orientation of the system's three rigid bodies.

6.2.2 Rigid body Dynamics

The derivation of the rigid body dynamic equations is based on the following assumptions:

1. The system's bodies are rigid.
2. The rolling wheel ideally rolls without slip.
3. Motors are not subject to friction or hysteresis effects.
4. Motor dynamics are neglected, being considerably faster than other dynamic terms.

While some of these are clearly simplifying assumptions, the reality gap will be taken care of by feedback control, possibly combined with learning, which is, among others, an interesting challenge of this testbed. With Assumption 2), the friction forces that act on the rolling wheel enforce an implicit nonholonomic constraint

$$[\dot{x} - r_w \dot{q}_4 \cos(q_3),\ \dot{y} - r_w \dot{q}_4 \sin(q_3)]^\mathsf{T} = 0, \tag{6.2}$$

with wheel radius r_w. The above constraint is nonholonomic, reducing the generalized coordinates required to describe the system's configuration to q.

As outlined in further detail by Daud et al. (2017), the system's equations of motion are obtained via the Euler-Lagrange equations (3.21).

The Wheelbot's EOM was derived using Matlab's symbolic toolbox. These symbolic equations were then transferred to an s-function in Simulink.

Simulation Model

In Simulink, the s-function dynamics model is combined with controllers and state-estimation routines. The controller model includes time delays in the obtained reference signal while the simulated IMU measurements are perturbed by Gaussian noise, whose distribution closely resembles the characteristics of the real IMU noise. The Wheelbot's simulated motion is animated using a Simulink "VR sink".

t

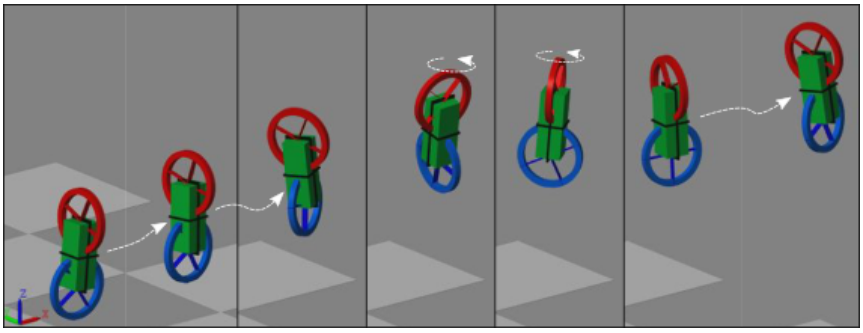


Figure 6.9: Simulink simulation of the Wheelbot. A feed-forward control signal is added to the balancing control, rotating the system three times around yaw in 4 s.

6.3 State Estimation

An accurate and drift-free estimate of the robot's tilt angles q_1 and q_2 is critical for good control performance. Tilt estimates are obtained by integrating the gyroscope's angular rates and using the accelerometer measurements. The integration of the gyroscopes rate measurements yields accurate (though drifting) tilt estimates. In comparison, the accelerometers provide drift-free yet noisy estimates of the tilt angles by using a carefully designed filter. In turn, the gyroscope and accelerometer tilt estimates are fused in a complementary fashion to yield accurate and drift-free tilt estimates as shown in Fig. 6.10.

The initial derivation of the tilt estimator Trimpe and D'Andrea (2010) and similarly Muehlebach and D'Andrea (2018) assume that the IMUs are attached to a body rotating around a non-accelerated pivot point. We show in Section 6.3.3 that for unicycle robots, the estimation algorithm can be extended to estimate the pivot point acceleration via motor encoder measurements. In turn, we propose a novel approach to tilt estimation that also applies to other types of rolling wheel robots, including Ballbots and Segway-esque robots.

6.3.1 Tilt and Rate Estimation using Gyroscopes

To obtain an estimate of the Euler rates $\{\dot{q}_1, \dot{q}_2, \dot{q}_3\}$, the measured gyro rates ${}^{i}\omega_i \in \mathbb{R}^3$ are first transformed into $\{B\}$, writing ${}^{B}\omega_i = R_{Bi}\,{}^{i}\omega_i$, and afterwards using the previous tilt estimate $\{\hat{q}_1(k-1), \hat{q}_2(k-1)\}$ transformed into $\{\mathrm{I}\}$, writing

$$\begin{bmatrix} \dot{q}_{1,\mathrm{G}} \\ \dot{q}_{2,\mathrm{G}} \\ \dot{q}_{3,\mathrm{G}} \end{bmatrix} = \begin{bmatrix} e_1^{\mathsf{T}} R_2 \\ e_2^{\mathsf{T}} \\ e_3^{\mathsf{T}} R_1 R_2 \end{bmatrix} \sum_{k=1}^{4} \frac{{}^{B}\omega_i(k)}{4}, \tag{6.3}$$

where here, as well as in what follows, we use the abbreviation $R_1 := R_1(\hat{q}_1(k-1))$ and $R_2 := R_2(\hat{q}_2(k-1))$ and omit the discrete time step k. Afterwards, the Euler rates in (6.3) are integrated to obtain drifting pose estimates $\{q_{1,\mathrm{G}}, q_{2,\mathrm{G}}, q_{3,\mathrm{G}}\}$.

6.3.2 Tilt Estimation using Accelerometers

A fundamental challenge in the computation of the tilt angle lies in distinguishing the gravitational acceleration from the acceleration that is caused by the system's motion. A

common method for estimation of a robot's tilt angle are extended, or unscented Kalman filters (Hertig et al., 2013; Klemm et al., 2019). These approaches usually rely on local approximations of the attitude dynamics, which model the temporal correlation between the observed data. In turn, Kalman filtering approaches are susceptible to errors of the dynamics and noise parameters.

As an alternative approach, Trimpe and D'Andrea (2010) proposed a tilt estimation algorithm for a rigid body with a non-accelerated pivot point that has been used on various balancing robots Gajamohan et al. (2013); Muehlebach and D'Andrea (2017); Trimpe and D'Andrea (2012). Given knowledge of the robot's kinematics model, the estimator in Trimpe and D'Andrea (2010) is computationally less demanding than Kalman filtering while providing a least-squares optimal estimation result. We thus follow this approach and extend it to a moving pivot in the next subsection.

Each accelerometer measures acceleration with respect to an observer in free fall. In turn, the acceleration measurement of the i-th sensor at position p_i with respect to $\{B\}$ reads

$$ {}^{B}m_i = {}^{B}\ddot{p}_i - {}^{B}g + {}^{B}n_i, \tag{6.4} $$

with the acceleration due to the IMU's movement $\ddot{p}_i$, gravitational acceleration g, and measurement errors n_i.

The approach in Trimpe and D'Andrea (2010) requires that the position vector p_{iW} between the non-accelerated pivot point and each of the IMUs remains constant with respect to $\{B\}$. Consequently, we select the rolling wheel's center W as the pivot point and estimate its acceleration $\ddot{p}_W$ as detailed in Section 6.3.3. By subtracting $\ddot{p}_W$ from ${}^{B}m_i$, we reduce (6.4) to the problem of estimating the pose of a rigid body with a non-accelerated pivot point, writing

$$ {}^{B}\hat{m}_i \hat{=} {}^{B}m_i - {}^{B}\ddot{p}_W = {}^{B}\ddot{p}_{iW} - {}^{B}g + {}^{B}n_i, \tag{6.5} $$

with

$$ {}^{B}\ddot{p}_{iW} = {}^{B}\Omega {}^{B}p_{iW}, \tag{6.6} $$

where p_{iW} points from the wheel center to the i-th IMU and ${}^{B}\ddot{p}_{iW} = R_{BI}\ddot{R}_{IB}{}^{B}p_{iW}$, with $\ddot{R}_{IB} = R_{IB}{}^{B}\Omega$. The matrix

$$ {}^{B}\Omega = {}^{B}\tilde{\omega}^2 + {}^{B}\dot{\tilde{\omega}}, \tag{6.7} $$

represents the body's angular as well as centripetal acceleration with the angular velocity of $\{B\}$ relative to $\{I\}$ being ω, and $\tilde{\omega}p_{iW} = \omega \times p_{iW}$. By inserting (6.6) into (6.5), one obtains a set of linear equations as

$$ M = QP + N, \tag{6.8} $$

$$ M = \begin{bmatrix} {}^{B}\hat{m}_1 & {}^{B}\hat{m}_2 & \dots & {}^{B}\hat{m}_L \end{bmatrix} \in \mathbb{R}^{3\times L}, \tag{6.9} $$

$$ Q = \begin{bmatrix} {}^{B}g & {}^{B}\Omega \end{bmatrix} \in \mathbb{R}^{3\times 4}, \tag{6.10} $$

$$ P = \begin{bmatrix} 1 & 1 & \dots & 1 \\ {}^{B}p_{1W} & {}^{B}p_{2W} & \dots & {}^{B}p_{LW} \end{bmatrix} \in \mathbb{R}^{4\times L}, \tag{6.11} $$

$$ N = \begin{bmatrix} {}^{B}n_1 & {}^{B}n_2 & \dots & {}^{B}n_L \end{bmatrix} \in \mathbb{R}^{3\times L}, \tag{6.12} $$

with the matrix of known sensor location P having full row-rank by design. As pointed out in Trimpe and D'Andrea (2010), the optimal Q, $Q^\star$, that minimizes

$$ \min_{\hat{Q}} \mathbb{E}\left[\|\hat{Q} - Q\|^2\right] \quad \text{subj. to} \quad \mathbb{E}[\hat{Q}] = Q, \tag{6.13} $$

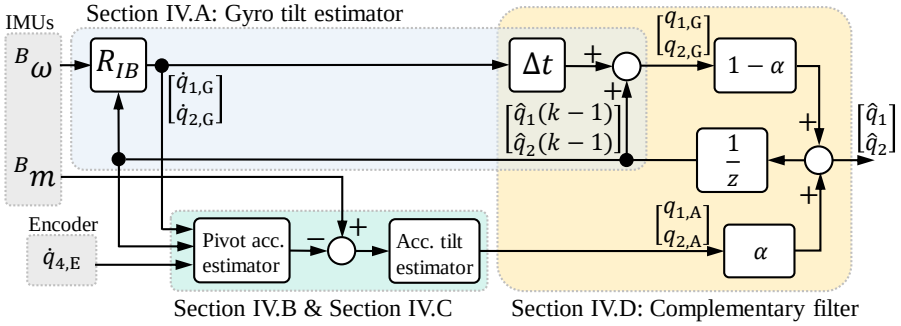


Figure 6.10: Block diagram of the tilt estimator.
Figure adapted from *Geist et al. (2022)*.

is given by

$$\hat{Q} = [{}^B\hat{g}, {}^B\hat{\Omega}] = MP^\mathsf{T}(PP^\mathsf{T})^{-1} = M[X_1^\star, X_2^\star], \tag{6.14}$$

with $X_1^\star \in \mathbb{R}^{L\times 1}$ being constant and only depending on the IMU positions. With (6.1), the gravitational acceleration reads

$${}^Bg = R_2^\mathsf{T} R_1^{\mathsf{T}I} g = g_0 \begin{bmatrix} -\cos(q_1)\sin(q_2) \\ \sin(q_1) \\ \cos(q_1)\cos(q_2) \end{bmatrix}. \tag{6.15}$$

Finally, with ${}^B\hat{g}$ as in (6.14), the tilt angle estimates are obtained as

$$q_{1,\mathrm{A}} = \arctan\left(\frac{{}^B\hat{g}_2}{\sqrt{{}^B\hat{g}_1^2 + {}^B\hat{g}_3^2}}\right), q_{2,\mathrm{A}} = \arctan\left(\frac{-{}^B\hat{g}_1}{{}^B\hat{g}_3}\right). \tag{6.16}$$

Muehlebach and D'Andrea Muehlebach and D'Andrea (2018) further extended Trimpe and D'Andrea (2010) by formulating (6.4) as a maximum likelihood estimation problem with the additional constraints that the gravitational acceleration lies on a sphere of radius g_0, ${}^B\tilde{\omega}^2$ is symmetric and ${}^B\dot{\tilde{\omega}}$ is skew-symmetric. This leads to a non-convex optimization problem which can be solved iteratively. The estimator proposed by Muehlebach and D'Andrea (2018) shows improved performance at the cost of a slightly higher computational load. The Wheelbot uses Trimpe and D'Andrea (2010) as it already provides a sufficiently accurate tilt estimate. Both Trimpe and D'Andrea (2010) and Muehlebach and D'Andrea (2018) can be used for wheeled balancing robots by estimating the pivot acceleration, as proposed in the next section.

6.3.3 Pivot Acceleration Estimation using Encoder

If we assume $\ddot{p}_W = 0$ in the tilt angle estimation via (6.14) and (6.16), a translational acceleration of the robot causes a non-zero tilt estimate even though the system did not tilt during the movement as shown in Figure 6.11 (Left, red line). We suggest to mitigate this error through the estimation of $\ddot{p}_W$ and its insertion into (6.5).

To estimate $\ddot{p}_W$, we decompose the acceleration into its partial components as depicted in Fig. 6.8, writing

$${}^B\ddot{p}_W = {}^B\ddot{p}_{CI} + {}^B\ddot{p}_{WC}, \tag{6.17}$$

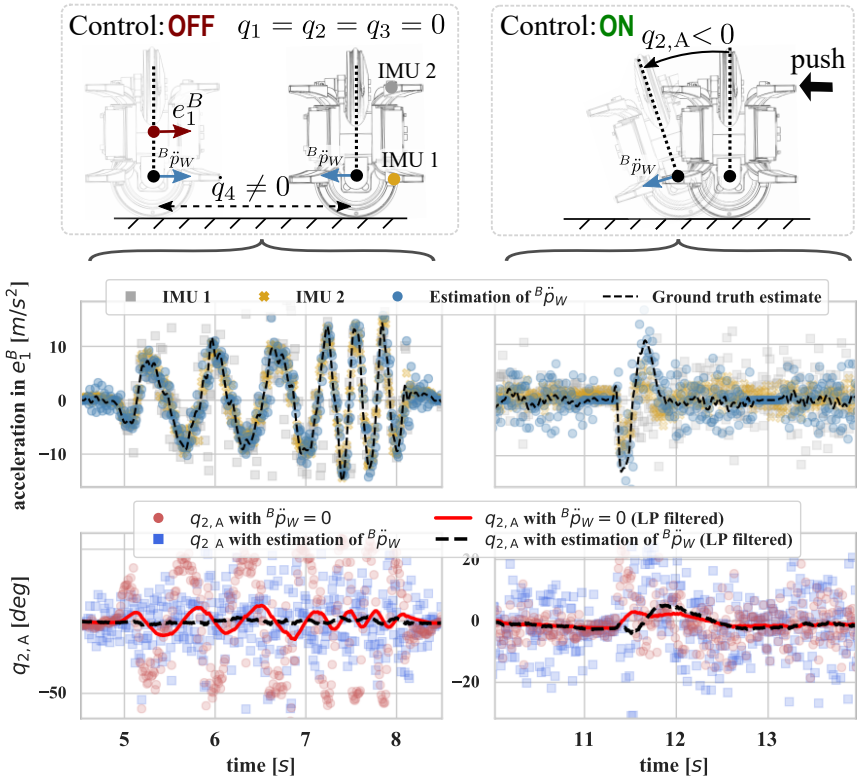


Figure 6.11: Left: The robot is translationally accelerated without rotating the system. Right: The balancing robot is pushed such that $q_2 < 0$. Top: Estimation of ${}^B\ddot{p}_W$ using $\ddot{q}_{4,\mathrm{E}}$. Bottom: Pitch estimate $q_{2,\mathrm{A}}$ with and without estimation of ${}^B\ddot{p}_W$. The lines are obtained by low pass filtering $q_{2,\mathrm{A}}$.
Figure adapted from *Geist et al. (2022)*.

where the vector p_{CI} points from I to the wheel contact point C, and p_{WC} points from C to W. As the rolling wheel is assumed to not slip, the acceleration at the ground contact point reads

$$ {}^B\ddot{p}_{CI} = R_2^\mathsf{T} R_1^\mathsf{T} \begin{bmatrix} r_w \ddot{q}_4 & r_w \dot{q}_3 \dot{q}_4 & 0 \end{bmatrix}^\mathsf{T} . \tag{6.18} $$

Note that the encoder measures $\dot{q}_{4,\mathrm{E}} = \dot{q}_4 - \dot{q}_2$. As $\dot{q}_4 \gg \dot{q}_2$, we assumed $\dot{q}_{4,\mathrm{E}} \approx \dot{q}_4$. An estimate for $\ddot{q}_{4,\mathrm{E}}$ is obtained by low-pass filtering $\dot{q}_{4,\mathrm{E}}$ and resorting to numerical differentiation. The acceleration due to the change of p_{WC} reads

$$ {}^B\ddot{p}_{WC} = R_2^\mathsf{T} R_1^\mathsf{T} \begin{bmatrix} 2 r_w \cos(q_1) \dot{q}_1 \dot{q}_3 + r_w \sin(q_1) \ddot{q}_3 \\ r_w \sin(q_1)(\dot{q}_1^2 + \dot{q}_3^2) - r_w \cos(q_1) \ddot{q}_1 \\ -r_w \cos(q_1) \dot{q}_1^2 - r_w \sin(q_1) \ddot{q}_1 \end{bmatrix} . \tag{6.19} $$

To determine the significance of the individual acceleration terms in (6.17), we ran a simulation of the Wheelbot in which the controller has been periodically excited, resulting in a pirouette-esque motion. This simulation as well as further experiments on the real robot indicate that in (6.17) only the term depending on $\ddot{q}_4$ significantly contributes to $\ddot{p}_w$ such that the other terms are being omitted for the estimation of the pivot acceleration.

Figure 6.11 illustrates the influence of the pivot acceleration estimation on the computation of the pitch angle. The ground truth estimate of ${}^B\ddot{p}_W$ is obtained by applying a non-causal low-pass filter with a cutoff frequency of 60 Hz to $r_w \ddot{q}_{4,\mathrm{E}}$. The first-order low-pass used

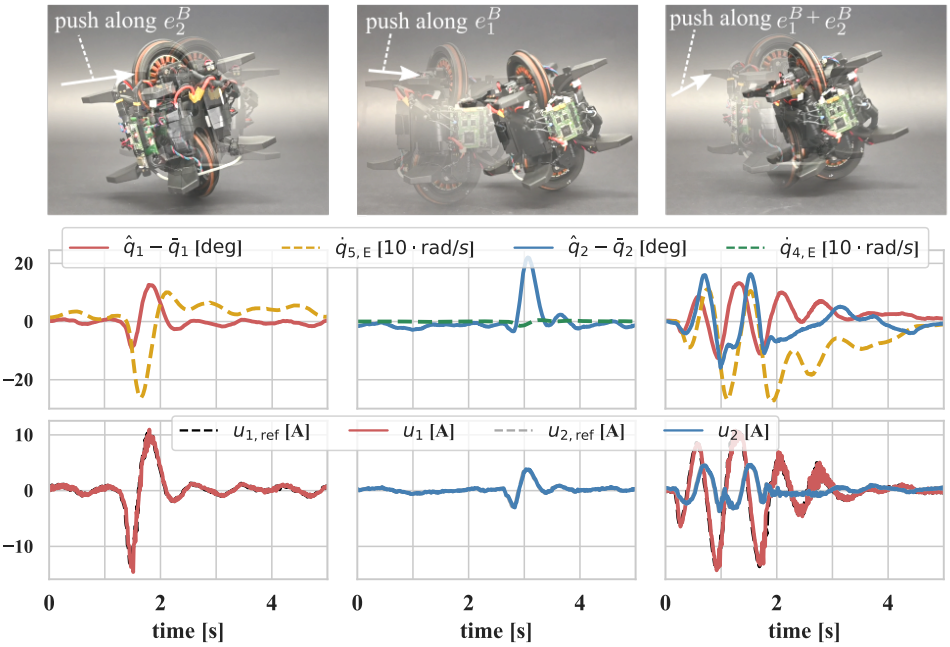


Figure 6.12: Balancing controller rejecting different pushes.
Figure adapted from *Geist et al. (2022)*.

to filter $q_{2,\mathrm{A}}$ has a cutoff frequency of 0.32 Hz. This cutoff frequency matches the cutoff frequency of the low-pass filter part of the complementary filter that is used to filter the $q_{2,\mathrm{A}}$ estimates. In Fig. 6.11 (Left), the robot is held at $q_1 = q_2 = q_3 = 0$ while being only translational accelerated. In turn, the IMUs directly measure ${}^B\ddot{p}_W = e_1 r_w \ddot{q}_4$. Notably, the low-pass filtered estimate of $q_{2,\mathrm{A}}$ with the estimation of ${}^B\ddot{p}_w$ remains close to zero. In Fig. 6.11 (Right), the robot is pushed such that q_2 turns negative and then positive while the controller overshoots. Notably, the low-pass filtered $q_{2,\mathrm{A}}$ with the estimation of ${}^B\ddot{p}_W$ reproduces the correct qualitative shape of q_2 during the push.

6.3.4 Sensor fusion via complementary filtering

The gyroscopes tilt estimate $\{q_{1,\mathrm{G}}, q_{2,\mathrm{G}}\}$ as in Section 6.3.1 is combined with the accelerometers tilt estimate $\{q_{1,\mathrm{A}}, q_{2,\mathrm{A}}\}$ as in Section 6.3.2 using the following formula

$$\begin{bmatrix} \hat{q}_1 \\ \hat{q}_2 \end{bmatrix} = \alpha \begin{bmatrix} q_{1,\mathrm{A}} \\ q_{2,\mathrm{A}} \end{bmatrix} + (1-\alpha) \begin{bmatrix} q_{1,\mathrm{G}} \\ q_{2,\mathrm{G}} \end{bmatrix}, \tag{6.20}$$

with fusion parameter $\alpha = 0.02$. Equation (6.20) corresponds to a discrete-time complementary filter that combines the high-frequent part of the gyroscope tilt estimate with the low-frequent part of the accelerometer tilt estimate Brown and Hwang (1997).

6.4 Control

In what follows, we present a straightforward design of a balancing controller as a proof of concept. In Section 6.4.2, we demonstrate the system's capability to jump onto one of its wheels by combining balancing control with cautiously engineered feed-forward control signals.

6.4.1 Balancing Control

Linear quadratic regulator (LQR) control synthesis has been used successfully for the control of unstable wheeled robots (Fankhauser and Gwerder, 2010; Klemm et al., 2019; Lee et al., 2013; Li et al., 2014). To obtain an LQR for balancing, the system's dynamics (**??**) are linearized around the upright equilibrium position ($q = \mathbf{0}, \dot{q} = \mathbf{0}$), to yield a linear state space model as

$$\frac{d}{dt}\begin{bmatrix} q_I \\ q_{II} \end{bmatrix} = \begin{bmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & A_2 \end{bmatrix}\begin{bmatrix} q_I \\ q_{II} \end{bmatrix} + \begin{bmatrix} B_1 & \mathbf{0} \\ \mathbf{0} & B_2 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{6.21}$$

with $q_I = [q_1, \dot{q}_1, q_5, \dot{q}_5]$, $q_{II} = [q_2, \dot{q}_2, q_4, \dot{q}_4]$, and system matrices of appropriate size. Due to the linearization around the upright equilibrium, the roll and pitch dynamics in (6.21) are decoupled. As in the linearized dynamics, the yaw angle q_3 and its velocity $\dot{q}_3$ are not controllable, the corresponding dynamic modes are being omitted from (6.21). We note that, in practical experiments, specific control actions lead to considerably agile movements along the yaw DOF. Additionally, the system's nonlinear dynamics model possesses numerous coupling terms between the yaw dynamics and the other states, indicating that viewed from a nonlinear system's perspective, the yaw dynamics can be controlled. However, an analysis of the system's nonlinear yaw dynamics exceeds the scope of this work.

The LQR's positive semi-definite weighting matrices for the roll controller $\{Q_1, R_1\}$ and pitch controller $\{Q_2, R_2\}$ are chosen as diagonal matrices. The diagonal elements of the Q and R matrices form tuning knobs that increase the quadratic cost of the respective state error or control input. With the A and B matrices from (6.21), carefully chosen $\{Q, R\}$ matrices, and the system's sample-time of $0.01s$, the LQR gains for the discrete-time system are obtained as $K_1 = \begin{bmatrix} 4.5 & 0.25 & 0.0003 & 0.0018 \end{bmatrix}$ and $K_2 = \begin{bmatrix} 1.6 & 0.14 & 0.04 & 0.0344 \end{bmatrix}$, such that

$$\begin{aligned} u_1 &= K_1 \begin{bmatrix} \hat{q}_1 - \bar{q}_1 & \dot{q}_{1,\mathrm{G}} & q_{5,\mathrm{E}} & \dot{q}_{5,\mathrm{E}} \end{bmatrix}^\mathsf{T}, \\ u_2 &= K_2 \begin{bmatrix} \hat{q}_2 - \bar{q}_2 & \dot{q}_{2,\mathrm{G}} & q_{4,\mathrm{E}} & \dot{q}_{4,\mathrm{E}} \end{bmatrix}^\mathsf{T}, \end{aligned} \tag{6.22}$$

with the estimator tilt angle bias $\{\bar{q}_1, \bar{q}_2\}$. To improve performance during non-steady state operation, the tilt angle bias has been estimated in a calibration routine before start.

Fig. 6.12 illustrates how the Wheelbot rejects pushes coming from different directions with respect to $\{B\}$. Fig. 6.12 (Bottom) depicts the motor current. With an estimated motor torque constant of $K_\mathrm{T} = 0.075$, the motors apply during disturbance rejection approximately $u = K_\mathrm{T} \cdot 13\mathrm{A} \approx 1\,\mathrm{Nm}$.

6.4.2 Jump-up Maneuvers

The jump-up maneuver for a Wheelbot is challenging as:

- Self-erection takes less than half a second at a control frequency of 100 Hz, while traversing a significant part of the system's state space, leaving a small error margin for computing control signals.
- During self-erection, the robot's ground contact points change, and in turn also its dynamics change.

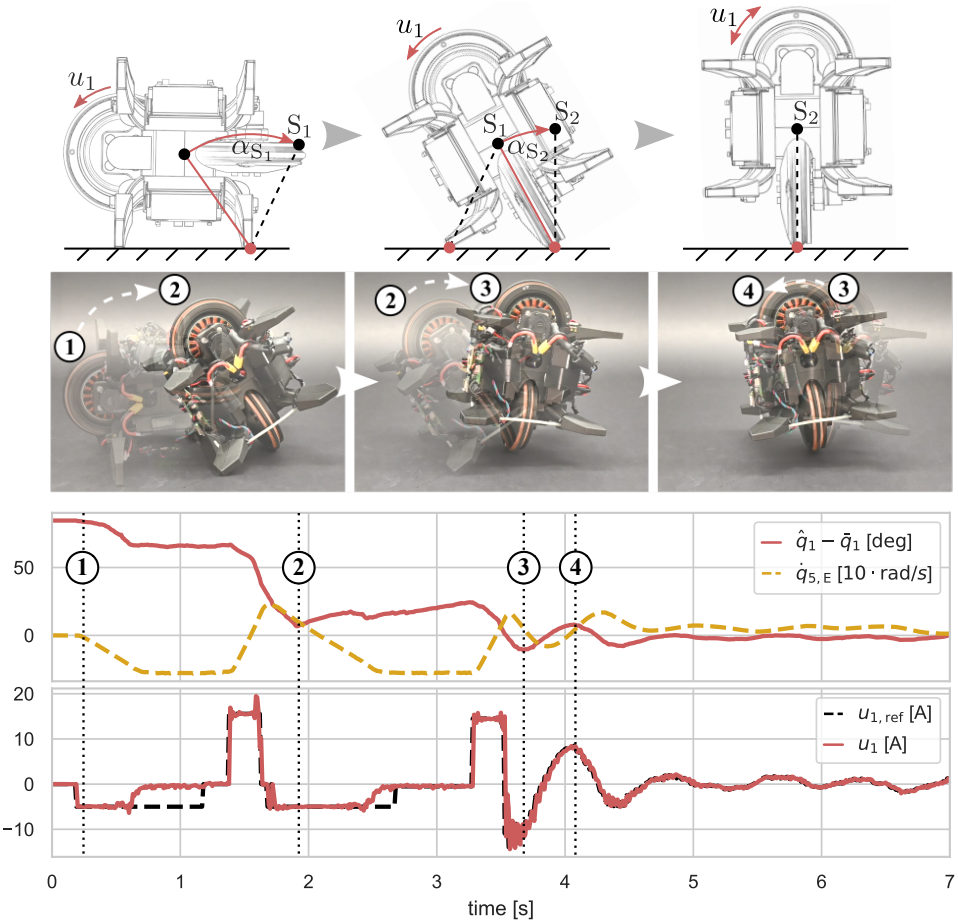


Figure 6.13: Stand-up experiment. Figure adapted from *Geist et al. (2022)*.

- When the wheel hits the ground, discontinuities and stick-slip effects may be introduced to the dynamics.
- The dynamics can be subject to forces arising from the deformation of mechanical components.

While similar challenges are often encountered in other systems, such as quadrupedal robots, the Wheelbot provides a comparably compact and low-dimensional system for research on iterative learning and repetitive control algorithms. Here, as soon as a balancing controller fails, the robot can automatically self-erect, and the experiment can be continued.

The robot self-erects either in a two-step or a one-step maneuver. The system first accelerates one wheel in the two-step maneuver until the rolling wheel has ground contact. Afterward, the robot rotates to its upright equilibrium position either by using its reaction wheel, which we refer to as *"stand-up"* or by its rolling wheel, which we refer to as *"roll-up"*.

Stand-up

Figure 6.13 (Top) illustrates a stand-up maneuver. First, the reaction wheel applies torques u_1 to rotate the robot such that its center of gravity (COG) lies at S_1 by $\alpha_{S_1} = 60\,\text{deg}$. Then, the reaction wheel moves the system from S_2 by $\alpha_{S_2} = 30\,\text{deg}$ such that $q_1 \approx 0\,\text{deg}$.

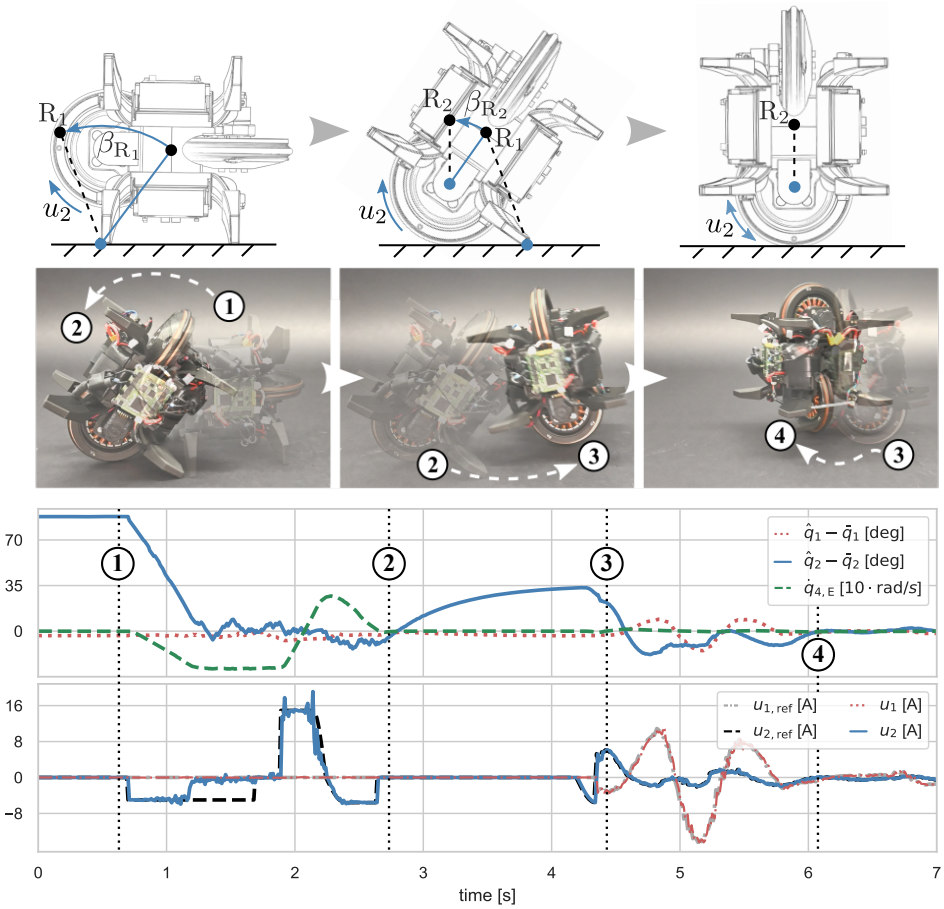


Figure 6.14: Roll-up experiment. Figure adapted from *Geist et al. (2022)*.

Figure 6.13 depicts the experimental results of a two-step stand-up. Here, the reaction wheel accelerates until $\hat{q}_1 \approx 30\,\text{deg}$ (Fig. 6.13, ②), then accelerates until $\hat{q}_1 \approx 0\,\text{deg}$ (Fig. 6.13, ③) and finally switches to the roll balancing controller (Fig. 6.13, ④). Notably, the torque produced by an electric motor in the direction of rotation is inversely proportional to its rotational speed. Therefore, preceding the stand-up, the reaction wheel is accelerated to $-290\,\text{rad/s}$. At the beginning of the stand-up, the normal force between the ground and the wheel does not suffice to prevent slip. Therefore, the pitch balancing controller is only switched on when the robot is nearly upright.

Roll-up

As depicted in Fig. 6.14 (Top), during a roll-up, the rolling wheel first applies torques u_2 to rotate the robot such that $\beta_{S_1} = 55\,\text{deg}$ and its COG lies at R_1. Secondly, the rolling wheel rotates the system by $\beta_{R_2} = 35\,\text{deg}$ to its upright equilibrium position with the system's COG being located at R_2. For the roll-up maneuver to be physically realizable, the normal force between the robot's wheel and the ground must be sufficiently large to prevent slip. Fig. 6.14 illustrates the experimental validation of a two-step roll-up maneuver. As the pivot acceleration estimation has not been switched off during the first step of the roll-up maneuver (Fig. 6.14, ① → ②), the pitch estimate drifts to zero. Importantly, the rolling wheel is quickly decelerated before establishing surface contact to prevent the wheel's O-ring

from melting due to frictional heat. Afterward, the roll-up maneuver pauses for 1.5 s such that the pitch estimate converges to 35 deg. In the second step of the roll-up (Fig. 6.14, ② → ④), $\hat{q}_2 - \bar{q}_2$ is used to activate the roll controller.

6.5 Summary

The Wheelbot's compact design and challenging composition of dynamical properties inspire research on the many unexplored areas of control theory. For example, one appealing goal is fast nonlinear trajectory control, *e.g.*, using model-predictive control. Moreover, due to its low-cost components and simple assembly, one can straightforwardly set up a robot fleet for research on networked control systems. As the Wheelbot can self-erect, it forms a platform for exploring algorithms that learn either dynamics or controller parameters while avoiding failure. Here, the system's low-dimensional dynamics ease the analysis and visualization of such safe learning algorithms.

In addition, the proposed testbed is interesting for a variety of learning control tasks including iterative/repetitive tasks of jumping up and continuous tasks of balancing. While we derived a dynamics model using solely first principles, the actual dynamics of the robot are subject to phenomena that are difficult to model analytically such as friction, motor dynamics, contacts, and material deformations. These phenomena render the Wheelbot a suitable testbed to study physics-informed learning approaches as detailed in Section 4.

Finally, as the Wheelbot's dynamics possess several interesting properties while being relatively simple in its design, we hope that this robot may assist in the education of students interested in robotics and control systems.

Chapter 7

Conclusion

This dissertation developed a unified view of physics-informed regression of robot dynamics. To this end, we analyzed how data-driven modeling techniques can be combined with implicitly-constrained rigid body dynamics. First, mechanical principles that lead to structural knowledge in rigid body dynamics were introduced in Chapter 3. These principles provided an overview of how a robot's EOM gives rise to valuable prior knowledge about the direction of dissipative forces, conservation of energy, and the length of constraint forces. We further discussed the utility of implicit constraint equations for robot dynamics modeling. In Chapter 4, a unified view of the errors within an analytical dynamics model was developed. In this view, analytical mechanics typically gives rise to function expressions in which linear operators (*e.g.*, the inertia matrix or differential operators) act on latent functions (*e.g.*, dissipative forces). Moreover, a rigid body dynamics model can be seen as a parametric network whose predictions deviate from a ground truth due to estimation and approximation errors. Based on this discussion, previous literature on physically-informed regression of robot dynamics was analyzed and categorized into parameter estimation of analytical dynamics models, ARM, and ALM. In ARM, an analytical dynamics model acts as a function prior to a data-driven model. In turn, the data-driven model approximates the error between the analytical dynamics model's predictions and observations of the dynamics. In comparison, ALM requires additional expert knowledge to distinguish between the functions forming an analytical dynamics model as known, parametrically-unknown, or unknown. In turn, a data-driven model can be built into the analytical dynamics model to approximate the unknown functions, while the parameters of the parametrically-unknown functions can be estimated from data. In Chapter 5, a new dynamics model is proposed in which GP regression is combined with implicitly constrained rigid body dynamics. The proposed algorithm is tested on several dynamical systems, including a 2D unicycle and a 6D robot arm. The insights gained on physics-informed regression of robot dynamics contributed to and benefited from the development of the "Wheelbot". The Wheelbot is a reaction wheel-driven unicycle robot designed for research on data-driven dynamics modeling and nonlinear control. In Chapter 6, the mechanical and electronic design of this robot was described, and a first design of a state estimator and controller is proposed. We further experimentally validated the robustness of the controller and demonstrated the Wheelbot's ability to self-erect from any initial position.

7.1 Discussion of Contributions

Section 1.3 provides a comprehensive overview on the contributions of this dissertation. These contributions can be summarized as

1. providing a concise introduction of implicitly-constrained rigid body mechanics linking algebraic derivations of a robot's EOM with Lagrangian optimization,
2. developing a unified view of physics-informed regression for robot dynamics modeling,
3. developing the GP^2 as a dynamics model that combines GP regression with implicitly-constrained rigid body dynamics,
4. and designing the Wheelbot, forming a novel robotic testbed for nonlinear control and data-driven modeling.

In the following section, we discuss these individual contributions and point to future research directions in the subsequent sections.

In Chapter 3, we gave a brief over on rigid body dynamics using the functional analysis perspective on linear operators as outlined in Beard (2002). We believe that the description of analytical mechanics in terms of linear operators and their fundamental subspaces provides helpful intuition on the structure underlying mechanical principles. It must be noted that the usual suspects for the derivation of robot dynamics such as the Euler-Lagrange equations and Newton-Euler equations seek to derive EOM that humans can easily comprehend. These formulations of a robot's dynamics have not been formulated to enable easy integration of data-driven modeling. For example, angle coordinate is often not a good choice as an input to a regression model. That is, the points 0 and 2π even though they may denote the same orientation of a coordinate system in Cartesian space, are two distinct points in $\mathbb{R}$ with non-zero norm. Therefore, one needs to carefully evaluate how, as well as which, generalized coordinates shall be fed to a data-driven model inside a physics-informed regression algorithm. The question arises if some analytical models of a robot's EOM are better suited for the combination with data-driven models either in terms of computational efficiency or mathematical properties improving sample efficiency.

7.2 Going Deeper into Analytical Latent Modeling

In Chapter 4, we developed a unified view of physics-informed models that distinguishes models into analytical dynamics models, ARM, and ALM. In ARM, the data-driven model directly approximates the residual between an analytical model and the observations. In comparison, in ALM, the data-driven model approximates latent functions inside the analytical model such that the outputs of the data-driven model are transformed by analytical expressions. Notably, the discussed literature on ALM approximates either a Lagrangian or a generalized force through a data-driven model. This also includes our work on GP^2 as detailed in **Chapter 5** in which a GP approximated unknown generalized forces inside a robot's analytical FD model. One possible route toward improving ALM models is to incorporate additional physics knowledge. We see two possible routes towards improving the sample-efficiency of ALM, namely *(i)* model the dynamics of a robot's joint torque

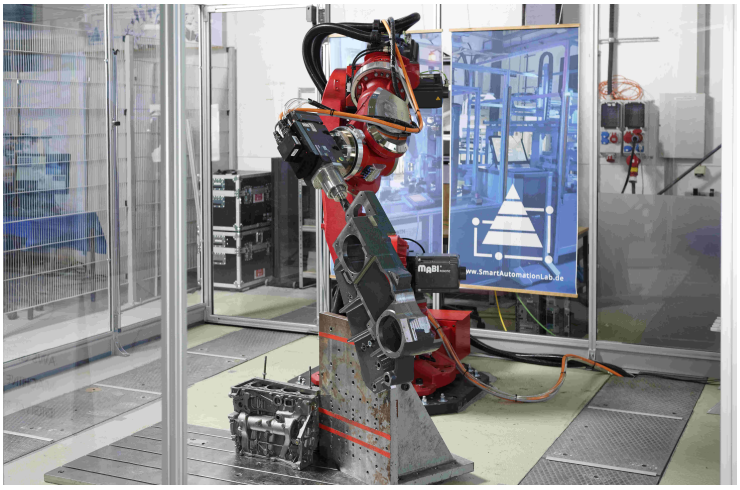


Figure 7.1: Serial robot MABI MAX 100. Image courtesy of Minh Trinh of the Laboratory for Machine Tools and Production Engineering of RWTH Aachen University.

through a physics-informed model as detailed in Section 7.2.1, or *(ii)* incorporate prior knowledge of differential kinematics and contact physics as outlined in Section 7.2.2.

7.2.1 ALM of Joint Torques

Our work led to an ongoing collaboration with the *Laboratory for Machine Tools and Production Engineering* of the RWTH Aachen University, in which we investigated how to design a physics-informed ID model of the "MABI MAX 100" industrial serial robot as depicted in Figure 7.1. With a milling tool attached to its end-effector, the MABI MAX 100 shall be used to mill metal parts, while requiring a considerably smaller space compared to conventional CNC milling machines. The precise control of the robot's end-effector during such a machining task requires an accurate dynamics model. Previous attempts at modeling the robot's ID showed insufficient prediction accuracy resorting either to an analytical model that is linear in θ_A as detailed in Section 4.2.1, directly trained a NN, or used a Lagrangian NN as discussed in Section 4.4.1. As an alternative approach, in the ongoing project, we combine insights from Section 4 and Section 5 to investigate how to combine RNEA (Featherstone, 2008, p. 98) with NN regression. Similar to Um et al. (2014) and Lutter et al. (2020), we assume that the largest error inside a robot's dynamics resides in the description of the joint torques, which subsequently shall be approximated by a NN. Here, we choose a NN instead of a GP as proposed in Section 5, as the combination of RNEA being a parametric model with NN is straightforward and the computational complexity of the algorithm does not grow with the number of input points. Our first experiments followed the approach of (Um et al., 2014) and directly approximated the errors in the joint torques through an individual NN per joint that only receives as inputs relevant state information.

ALM with an Analytical Joint Torque Model The joint torques arise due to forces acting on the bodies and the control force being caused inside the joint. These joint torques not only depend on dissipative forces, but also on elastic deformation of shafts inside the joint and potential slack inside a gear. We currently investigate how analytical

models of these phenomena may additionally benefit ALM with a RNEA. The basic idea is, instead of approximating the errors in the joint torques directly through a NN, one can use several smaller NN inside an analytical model of the joint torques. These NNs can then be specifically designed to account for the dissipativity of the friction torques, energy conservation in the motor shaft deformations, or slack in the gears of the robots.

Joint Optimization of Analytical and Data-driven Model Parameters Another aspect of this ongoing project is the estimation of unknown parameters of the analytical model. In principle, it is possible to learn unknown analytical parameters with the parameters of a NN using automatic differentiation. However, the parameter gradients of an analytical dynamics model differ considerably in their magnitude compared to the NN's parameter gradients. This difference in the size of the gradients aggravates the joint optimization of analytical parameters and NN parameters. One could in principle, mitigate this problem by switching between the optimization of the analytical parameters and NN parameters (cf. coordinate descent). Yet, it is unclear how such an approach affects the optimization result.

7.2.2 ALM of Robot-Surface Interactions

In Chapter 5, we modeled the surface friction forces arising between a robot's end-effector and a surface as Q_z which we approximated by a GP. Here, we modeled Q_z in the robot arm's generalized coordinate space $\mathbb{R}^{n_q}$ being referred to as the joint space. However, as shown in Figure 7.2, end-effector friction forces are caused in the *task space*, that is the six-dimensional space that describes the end-effector's position in Cartesian coordinates and its orientation. In particular, through a series of tribologic and elastomechanical phenomena, the end-effector force $F_\mathrm{I} \in \mathbb{R}$ being normal to the surface causes F_z.

As pointed out by Um et al. (2014), a data-driven model learning the friction forces acting in a robot's joint may only depend on the local joint coordinates and the forces acting in and on the joint. The same idea may also apply to a data-driven model that shall directly approximate F_z and whose output is then transformed through the "manipulator geometric Jacobian" J_E^T into the joint space, cf. (Siciliano et al., 2010, p. 148). Figure 7.2 illustrates this approach. A data-driven model that learns the friction force F_z may only have a single output dimension as the predicted friction force must oppose the end-effector's velocity. Moreover, one can ensure that the outputs of the data-driven model are dissipative as discussed in Section 4.4.2, and that this data-driven model receives as inputs the end-effector position relative to the surface, velocities, as well as F_I. In summary, using knowledge of the system's kinematics and dynamics may significantly reduce the dimensionality of the regression problem.

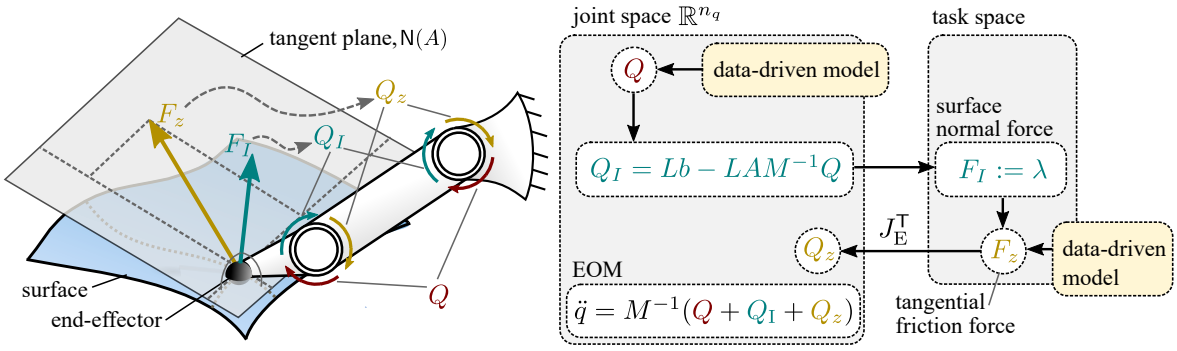


Figure 7.2: In a robot arm whose end-effector touches a surface, a force normal to the surface Q_I cause the non-ideal friction forces F_I being tangential to the surface. If the coordinate transformation J_E^T is known, a data-driven model can learn the end-effector friction in the lower-dimensional task space while potentially incorporating additional physics knowledge. Left half of the image is taken from *Rath et al. (2021)*.

Compliancy in End-effector and Surface Material Depending on the robotic system, it may not suffice to model the end-effector contact as a single point between a rigid surface and a rigid body. Instead, the end-effector may also be subject to deformation in the surface and to compliance in its end-effector. Under these assumptions, it is considerably more difficult to identify the forces acting on a robot's end-effector. Nonetheless, it may be possible to derive a suitable analytical model that could be augmented with a data-driven model.

Changes in the Contact Configuration For simplicity's sake, this dissertation assumed that the implicit constraints in a robot's dynamics description are equalities. Yet, a robot's leg or end-effector is typically modeled as being subject to inequality constraints. In turn, implicit constraints can become inactive which implies that parts of Q_I, Q_z become zero. On the other hand, if the contact closes, not only the implicit constraints will become active, but also a *collision* causes an instantaneous change in the system's velocities. The inclusion of collisions into physics-informed regression models is another interesting future research question.

7.3 Physics-informed Control of the Wheelbot

In the beginning of this dissertation, we pointed out that the control of a robotic system is closely intertwined with modeling its dynamics. As the logical next step, we plan to explore how the control of the Wheelbot may benefit from physics-informed regression. This includes steering the Wheelbot along a reference trajectory by using model predictive control as depicted in Figure 7.3. As a promising starting point, an LQR controller design could be developed that incorporates an implicitly-constrained dynamics model into the controller design as shown in Klemm et al. (2020). Another interesting aspect forms safe-learning control of a data-driven control policy, where during the exploration of the system's dynamics a data-driven controller is being optimized. Interesting approaches towards safely learning to drive the Wheelbot along a reference trajectory are (Berkenkamp

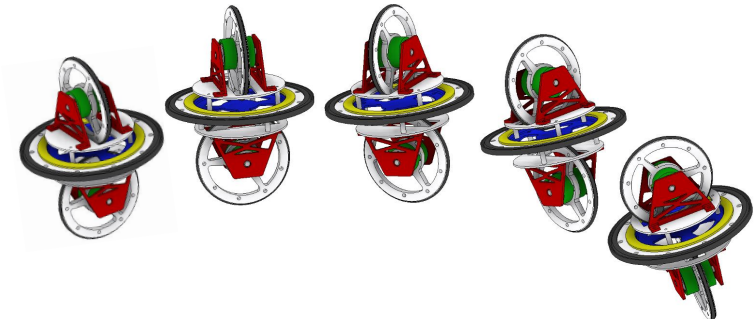


Figure 7.3: Illustration of an early prototype of the Wheelbot driving along a curve.

et al., 2017), (Berberich et al., 2020a), and (Berberich et al., 2020b).

An intriguing future research direction for physics-informed regression is how to transfer knowledge between several dynamical systems also being referred to as *transfer learning*. A discussion of transfer learning in the context of physics-informed regression requires that the notion of structural knowledge must be extended to several presumably structurally similar systems. In this regard, the Wheelbot is a particularly well-suited testbed for transfer learning. On one hand, it is straightforward to assemble a fleet of Wheelbots which allows to investigate how dynamics knowledge can be exchanged between several similar systems. On the other hand, one can easily replace the wheel assembly of the Wheelbot to obtain a coaxial-oriented reaction wheel or a pendulum reaction wheel unicycle robot. In these problem settings, questions arise on how to exchange knowledge between physics-informed regression models as well as between the closed-loop dynamics of these systems. Interesting works evolving around transfer learning in robotics from a control theoretic perspective are (Raimalwala et al., 2016) and (Helwa and Schoellig, 2017).

Appendix A

Technical Proofs

A.1 Derivation of Virtual Displacements

As shown by Udwadia et al. (1997), (3.31) follows from the Taylor expansion about time t of a displacement $q(t+dt)$ assuming that position and velocity are given fixed quantities at time t, such that

$$q(t+dt) = q(t) + (dt)\dot{q}(t) + \frac{(dt)^2}{2}\ddot{q}(t) + \mathcal{O}(dt^3), \tag{A.1}$$

with dt being an infinitesimal quantity. If the system would not be subject to any implicit constraints and neglecting the higher order terms $\mathcal{O}(dt^3)$, the particle would move to the position according with (3.17) to $q_{\mathrm{U}} = q(t) + (dt)\dot{q}(t) + \frac{(dt)^2}{2}M^{-1}Q$. With (A.1) one defines the virtual displacement at time $(t+dt)$ as the difference between the actual displacement $q(t+dt) = \mathrm{vec}\{q, \dot{q}, \ddot{q}\}$ and a possible displacement $s(t+dt) = \mathrm{vec}\{q, \dot{q}, \ddot{q}'\}$, writing

$$\delta q(t+dt) = q^b(t+dt) - q^a(t+dt) = \frac{(dt)^2}{2}[\ddot{q}^b(t) - \ddot{q}^a(t)] = \frac{(dt)^2}{2}\delta\ddot{q}(t). \tag{A.2}$$

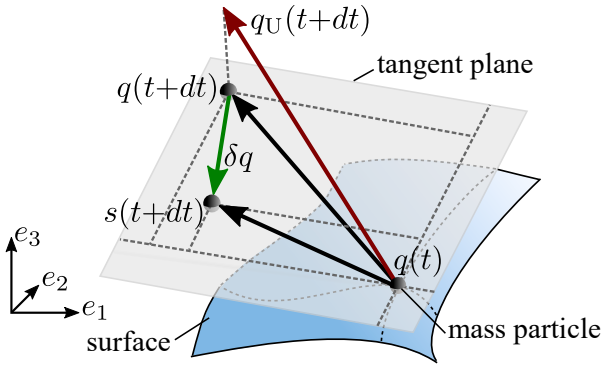


Figure A.1: Due to the presence of implicit constraints, the system (*e.g.*, a mass particle sliding over a surface) moves from $q(t)$ to $q(t+dt)$. Without the constraint the system moves to $q_{\mathrm{U}}(t+dt)$. In a hypothetical thought experiment, the system could move to $s(t+dt)$ differing from $q(t+dt)$ in the variation $\delta q(t+dt)$. Figure adapted from (Udwadia and Kalaba, 2007, p. 223).

Figure (A.1) illustrates how the virtual displacement emerges as a variation of the systems state from $q(t+dt)$ to $s(t+dt)$. As every possible motion must fulfill (3.30), we can insert

$q^a(t+dt)$ and $q^b(t+dt)$ into (3.30), and take the difference between the expressions to obtain

$$A(q(t),\dot{q}(t),t)\delta\ddot{q}(t) = A\delta q(t+dt) = 0, \tag{A.3}$$

which with (A.2) and $dt \to 0$ yields (3.31).

A.2 Oblique Projections inside the UKE

The matrices $T = I - M^{-1}A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}A$ and $S = I - A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}AM^{-1}$ with $A \in \mathbb{R}^{n_{\mathrm{I}} \times n_q}$, $M \in \mathbb{R}^{n_q \times n_q}$, and $M > 0$ are oblique projectors. That is these matrices are idempotent but not symmetric. T is idempotent as

$$T^2 = I + M^{-1}A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}AM^{-1}A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}A - 2M^{-1}A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}A, \tag{A.4}$$

$$= I - M^{-1}A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}A = T. \tag{A.5}$$

S is idempotent as

$$S^2 = I + A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}AM^{-1}A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}AM^{-1} - 2A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}AM^{-1}, \tag{A.6}$$

$$= I - A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}AM^{-1} = S. \tag{A.7}$$

It is starightforward to varify that the matrices S and T are not symmetric. T denotes a projector into $\mathsf{N}(A)$ as

$$0 = ATv = (A - AM^{-1}A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}A)v = (A - A)v = 0, \tag{A.8}$$

with $v \in \mathbb{R}^{n_q}$ and using the fact that if $M > 0$ then $AM^{-1}A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}A = A$ (Bernstein, 2018, p. 636, Fact 8.4.17). S denotes also a projector into $\mathsf{N}(A)$ as

$$0 = SA^{\mathsf{T}}v, \tag{A.9}$$

$$= (A^{\mathsf{T}} - A^{\mathsf{T}}(AM^{-1}A^{\mathsf{T}})^{+}AM^{-1}A^{\mathsf{T}})v, \tag{A.10}$$

$$= (A^{\mathsf{T}} - A^{\mathsf{T}})v, \tag{A.11}$$

with $v \in \mathbb{R}^{n_{\mathrm{I}}}$ shows that S maps a vector $A^{\mathsf{T}}v \in \mathsf{R}(A^{\mathsf{T}})$ to null.

Bibliography

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.

Mauricio A Alvarez, Lorenzo Rosasco, Neil D Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

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.

Ara Arabyan and Fei Wu. An improved formulation for constrained mechanical systems. *Multibody System Dynamics*, 2(1):49–69, 1998.

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.

Alexander Badri-Spröwitz, Alborz Aghamaleki Sarvestani, Metin Sitti, and Monica A. Daley. Birdbot achieves energy-efficient gait with minimal control using avian-inspired leg clutching. *Science Robotics*, 7:eabg4055, March 2022. doi: 10.1126/scirobotics.abg4055.

Olivier Andre Bauchau. *Flexible multibody dynamics*, volume 176. Springer, 2011.

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

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.

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

Richard E Bellman. *Adaptive control processes*. Princeton university press, New Jersey, 1961.

Julian Berberich, Anne Koch, Carsten W Scherer, and Frank Allgöwer. Robust data-driven state-feedback design. In *2020 American Control Conference (ACC)*, pages 1532–1538. IEEE, 2020a.

Julian Berberich, Johannes Köhler, Matthias A Müller, and Frank Allgöwer. Data-driven model predictive control with stability and robustness guarantees. *IEEE Transactions on Automatic Control*, 66(4):1702–1717, 2020b.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

Dennis S Bernstein. *Scalar, vector, and matrix mathematics*. Princeton university press, 2018.

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

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

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`.

Robert Grover Brown and Patrick YC Hwang. *Introduction to random signals and applied Kalman filtering*, volume 3rd ed. New York: John Wiley, 1997.

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

Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold gaussian processes for regression. In *International Joint Conference on Neural Networks (IJCNN)*, pages 3338–3345. IEEE, 2016.

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.

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.

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.

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, 2018a.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors,

Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018b.

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

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

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.

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

Yohanes Daud, Abdullah Al Mamun, and Jian-Xin Xu. Dynamic modeling and characteristics analysis of lateral-pendulum unicycle robot. *Robotica*, 35(3):537–568, Mar 2017. ISSN 0263-5747. doi: 10.1017/S0263574715000703.

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.

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.

Marc Peter Deisenroth. *Efficient reinforcement learning using Gaussian processes*, volume 9. KIT Scientific Publishing, 2010.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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.

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.

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

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.

Péter Fankhauser and Corsin Gwerder. Modeling and control of a ballbot. Technical report, 06 2010.

Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2008.

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.

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

Mohanarajah Gajamohan, Michael Muehlebach, Tobias Widmer, and Raffaello D'Andrea. The cubli: A reaction wheel based 3d inverted pendulum. *European Control Conference*, pages 268–274, 2013. doi: 10.23919/ECC.2013.6669562.

Carl Friedrich Gauß. Über ein neues allgemeines Grundgesetz der Mechanik. *Journal für die reine und angewandte Mathematik*, 4:232–235, 1829.

A. René Geist. Combining Gaussian processes and stochastic optimal control for non-myopic field exploration with autonomous mobile robots. Master's thesis, TU Hamburg, Institute of Mechanics and Ocean Engineering, 4 2018.

A. René 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 (L4DC)*, volume 120 of *Proceedings of Machine Learning Research (PMLR)*, pages 225–234, June 2020.

A. René Geist and Sebastian Trimpe. Structured learning of rigid-body dynamics: A survey and unified view from a robotics perspective. *GAMM Mitteilungen*, 44(2, Special Issue: Scientific Machine Learning):34, June 2021.

A René Geist, Andreas Hansen, Eugen Solowjow, Shun Yang, and Edwin Kreuzer. Data collection for robust end-to-end lateral vehicle control. In *ASME 2017 Dynamic Systems and Control Conference*, pages V001T45A007–V001T45A007. American Society of Mechanical Engineers, 2017.

A. René Geist, Jonathan Fiene, Naomi Tashiro, Zheng Jia, and Sebastian Trimpe. The wheelbot: A jumping reaction wheel unicycle. *IEEE Robotics and Automation Letters*, 7 (4):9683–9690, 2022.

Gabriel Goh. Why momentum really works. *Distill*, 2017. doi: 10.23915/distill.00006. URL `http://distill.pub/2017/momentum`.

GPy. A Gaussian process framework in python. `github.com/SheffieldML/GPy`, 2012.

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.

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

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. doi: 10.1109/LRA.2020.2976639.

Jayesh K. Gupta, Kunal Menda, Zachary Manchester, and Mykel Kochenderfer. Structured mechanical models for robot learning and control. 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 328–337. PMLR, 10–11 Jun 2020.

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. doi: 10.1038/s41586-020-2649-2.

Mohamed K Helwa and Angela P Schoellig. Multi-robot transfer learning: A dynamical system perspective. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4702–4708. IEEE, 2017.

Phillipp Hennig. *Probabilistic Inference and Learning*. University of Tübingen, 2020.

Lionel Hertig, Dominik Schindler, Michael Bloesch, C. David Remy, and Roland Siegwart. Unified state estimation for a ballbot. In *International Conference on Robotics and Automation*, pages 2471–2476. IEEE, May 2013. ISBN 978-1-4673-5643-5. doi: 10.1109/ICRA.2013.6630913.

Eckhard Hitzer, Tohru Nitta, and Yasuaki Kuroe. Applications of clifford's geometric algebra. *Advances in Applied Clifford Algebras*, 23(2):377–404, 2013.

Sebastian Höfer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Melissa Mozifian, Florian Golemo, Chris Atkeson, Dieter Fox, Ken Goldberg, John Leonard, et al. Sim2real in robotics and automation: Applications and challenges. *IEEE transactions on automation science and engineering*, 18(2):398–400, 2021.

Kurt Hornik. Some new results on neural network approximation. *Neural networks*, 6(8): 1069–1072, 1993.

Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 38–44. IEEE, 2016.

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.

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.

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.

Philip Edward Bertrand Jourdain. *Note on an analogue of Gauss' principle of least constraint*, volume XL. 1909.

Dan Kalman. Leveling with lagrange: An alternate view of constrained optimization. *Mathematics Magazine*, 82(3):186–196, 2009.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014a.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014b. URL `http://arxiv.org/abs/1412.6980`. arXiv: 1412.6980.

Victor Klemm, Alessandro Morra, Ciro Salzmann, Florian Tschopp, Karen Bodie, Lionel Gulich, Nicola Kung, Dominik Mannhart, Corentin Pfister, Marcus Vierneisel, Florian Weber, Robin Deuber, and Roland Siegwart. Ascento: A two-wheeled jumping robot. In *International Conference on Robotics and Automation*, pages 7515–7521. IEEE, May 2019. ISBN 978-1-5386-6027-0. doi: 10.1109/ICRA.2019.8793792.

Victor Klemm, Alessandro Morra, Lionel Gulich, Dominik Mannhart, David Rohr, Mina Kamel, Yvain de Viragh, and Roland Siegwart. Lqr-assisted whole-body control of a wheeled bipedal robot with kinematic loops. *IEEE Robotics and Automation Letters*, 5 (2):3745–3752, 2020. doi: 10.1109/LRA.2020.2979625.

Okan Koç, Guilherme Maeda, and Jan Peters. Online optimal trajectory generation for robot table tennis. *Robotics and Autonomous Systems*, 105:121–137, 2018.

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.

Prasanth B Koganti and Firdaus E Udwadia. Unified approach to modeling and control of rigid multibody systems. *Journal of Guidance, Control, and Dynamics*, 39(12):2683–2698, 2016.

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

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.

Friedhelm Kuypers. *Klassische Mechanik*. John Wiley & Sons, 2016.

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

Richard A Layton. *Principles of analytical system dynamics*. Springer Science & Business Media, 2012.

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.

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.

Jaeoh Lee, Seongik Han, and Jangmyung Lee. Decoupled dynamic control for pitch and roll axes of the unicycle robot. *Transactions on Industrial Electronics*, 60(9):3814–3822, Sep 2013. ISSN 0278-0046. doi: 10.1109/TIE.2012.2208431.

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

Zhijun Li, Chenguang Yang, and Liping Fan. *Advanced Control of Wheeled Inverted Pendulum Systems*. 04 2014. ISBN 978-1-4471-2962-2. doi: 10.1007/978-1-4471-2963-9.

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

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

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, 2019b.

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.

Michael Lutter, Johannes Silberbauer, Joe Watson, and Jan Peters. Differentiable physics models for real-world offline model-based reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4163–4170. IEEE, 2021.

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

Jan R Magnus and Heinz Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 2019.

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.

Hossein Mohammadi, Rodolphe Le Riche, Nicolas Durrande, Eric Touboul, and Xavier Bay. An analytic comparison of regularization methods for Gaussian Processes. *arXiv:1602.00853 [math, stat]*, May 2017. URL `http://arxiv.org/abs/1602.00853`. arXiv: 1602.00853.

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.

Michael Muehlebach and Raffaello D'Andrea. Nonlinear analysis and control of a reaction-wheel-based 3-d inverted pendulum. *Transactions on Control Systems Technology*, 25(1): 235–246, Jan 2017. ISSN 1063-6536. doi: 10.1109/TCST.2016.2549266.

Michael Muehlebach and Raffaello D'Andrea. Accelerometer-based tilt determination for rigid bodies with a nonaccelerated pivot point. In *Transactions on Control Systems Technology*, volume 26, pages 2106–2120. IEEE, Nov 2018. doi: 10.1109/TCST.2017.2753171.

Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

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.

Umashankar Nagarajan, George Kantor, and Ralph Hollis. The ballbot: An omnidirectional balancing mobile robot. *The International Journal of Robotics Research*, 33(6):917–930, May 2014. ISSN 0278-3649. doi: 10.1177/0278364913509126.

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

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, April 2008.

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.

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

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.

Jan Peters, Michael Mistry, Firdaus Udwadia, Jun Nakanishi, and Stefan Schaal. A unifying framework for robot control with redundant dofs. *Autonomous Robots*, 24(1):1–12, 2008.

J.-C. Piedboeuf. Kane's equations or jourdain's principle? In *Proceedings of 36th Midwest Symposium on Circuits and Systems*, pages 1471–1474 vol.2, 1993. doi: 10.1109/MWSCAS.1993.343389.

Charles R. Qi, Li Yi, Hao Su, and Leonidas 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.

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.

Kaizad V Raimalwala, Bruce A Francis, and Angela P Schoellig. A preliminary study of transfer learning between unicycle robots. In *2016 AAAI Spring Symposium Series*, 2016.

Lucas Rath, A. René Geist, and Sebastian Trimpe. Using physics knowledge for learning rigid-body forward dynamics with Gaussian process force priors. In *Proceedings of the 5th Conference on Robot Learning (CORL)*, volume 164 of *Proceedings of Machine Learning Research (PMLR)*, pages 101–111, 08–11 Nov 2021.

David Remy. *Computational Dynamics for Robotics*. University of Stuttgart, 2019.

Yusie Rizal, Chun-Ting Ke, and Ming-Tzu Ho. Point-to-point motion control of a unicycle robot: Design, implementation, and validation. In *International Conference on Robotics and Automation*, pages 4379–4384. IEEE, May 2015. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139804.

M Anfa'ur Rosyidi, Eko Henfri Binugroho, S Ekti Radin Charel, R Sanggar Dewanto, and Dadet Pramadihanto. Speed and balancing control for unicycle robot. In *International Electronics Symposium*, pages 19–24. IEEE, 2016.

Mike Rouleau. Self balancing stick - dual axis reaction wheel in- verted pendulum, 2015. URL `https://www.youtube.com/watch?v=woCdjbsjbPg`.

Matteo Saveriano, Yuchao Yin, Pietro Falco, and Dongheui Lee. Data-efficient control policy search using residual dynamics learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4709–4715, 2017. doi: 10.1109/IROS.2017.8206343.

Werner Schiehlen and Peter Eberhard. *Applied dynamics*, volume 57. Springer, 2014.

Thomas B Schön and Fredrik Lindsten. Manipulating the multivariate gaussian density. *Division of Automatic Control, Linköping University, Sweden, Tech. Rep*, 166, 2011.

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

Ercan Solak, Roderick Murray-Smith, William E Leithead, Douglas J Leith, and Carl Edward Rasmussen. Derivative observations in gaussian process models of dynamic systems. 2003.

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.

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.

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

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.

Sebastian Trimpe and Raffaello D'Andrea. Accelerometer-based tilt estimation of a rigid body with only rotational degrees of freedom. In *International Conference on Robotics and Automation*, pages 2630–2636. IEEE, 2010. doi: 10.1109/ROBOT.2010.5509756.

Sebastian Trimpe and Raffaello D'Andrea. The balancing cube: A dynamic sculpture as test bed for distributed estimation and control. *IEEE Control Systems*, 32(6):48–75, Dec 2012. ISSN 1066-033X. doi: 10.1109/MCS.2012.2214135.

FE Udwadia. A new perspective on the tracking control of nonlinear structural and mechanical systems. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 459(2035):1783–1800, 2003.

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

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.

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

Firdaus E Udwadia and Robert E Kalaba. Explicit equations of motion for mechanical systems with nonideal constraints. *J. Appl. Mech.*, 68(3):462–467, 2001.

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

Firdaus E Udwadia and Prasanth B Koganti. Optimal stable control for nonlinear dynamical systems: an analytical dynamics based approach. *Nonlinear Dynamics*, 82(1):547–562, 2015.

Firdaus E Udwadia and Phailaung Phohomsiri. Explicit equations of motion for constrained mechanical systems with singular mass matrices and applications to multi-body dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 462(2071):2097–2117, 2006.

Firdaus E Udwadia and Aaron D Schutte. Equations of motion for general constrained systems in lagrangian mechanics. *Acta mechanica*, 213(1):111–129, 2010.

Firdaus E Udwadia and Thanapat Wanichanon. Explicit equation of motion of constrained systems. In *Nonlinear approaches in engineering applications*, pages 315–347. Springer, 2012.

Firdaus E Udwadia and Thanapat Wanichanon. On general nonlinear constrained mechanical systems. *Numerical Algebra, Control & Optimization*, 3(3):425, 2013.

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.

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.

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.

David William Vos. *Nonlinear control of an autonomous unicycle robot: practical issues*. PhD thesis, MIT, 1992.

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

C Woernle. Mehrkörpersysteme: Eine Einführung in die Kinematik und Dynamik von Systemen starrer Körper. 2016.

Xiao-Min Zhao, Ye-Hwa Chen, Han Zhao, and Fang-Fang Dong. Udwadia–kalaba equation for constrained mechanical systems: formulation and applications. *Chinese journal of mechanical engineering*, 31(1):1–14, 2018.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.