Identifying Objects' Inertial Parameters with Robotic Manipulation to Create Simulation-Ready Assets

by

Andy Lambert

B.S. Electrical Engineering and Computer Science
Massachusetts Institute of Technology (2022)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2023

Authored by: Andy Lambert
Department of Electrical Engineering and Computer Science
May 12, 2023

Certified by: Russ Tedrake
Toyota Professor of EECS, Aero/Astro, MechE
Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Identifying Objects' Inertial Parameters with Robotic Manipulation to Create Simulation-Ready Assets

by

Andy Lambert

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 2023, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Real2Sim is the problem of simulating objects and scenes via real world data, allowing a robot to imagine future interactions with its environment. However, many existing approaches either do not consider the dynamics of objects being simulated or make assumptions about their mass distributions. In this work, we aim to make use of robotic arm payload identification techniques in order to enhance the dynamic accuracy of objects generated from a Real2Sim pipeline for manipulation tasks. While the payload identification literature is vast, applying these methods in practice has various challenges and limitations. Upon implementing these techniques, we gain understanding of best practices in the engineering sense. We hope that these methods can be used to provide ground truth data for other robot learning tasks on the road towards generalized dynamic intuition.

Thesis Supervisor: Russ Tedrake
Title: Toyota Professor of EECS, Aero/Astro, MechE

Acknowledgments

This work is dedicated to my high school robotics teammate and good friend Dominic Buraglio, who sadly passed away in 2022. Dom's kindness and passion for engineering inspired me each day that we worked together, and I may not have chosen to pursue robotics if our paths had not crossed at that time. Thank you for everything, Dom.

* * *

My deepest thanks go to my supervisor, Russ Tedrake, for teaching me so much about this optimization and dynamic side of robotics that I had not deeply explored before. It was an immense privilege to study robotics under his mentorship. I would also like to thank the rest of the Robot Locomotion Group for helping me grow into the role of a researcher. I especially want to acknowledge my collaborators, Nicholas Pfaff, Lirui Wang, and Terry Suh, who helped me manifest these ideas outside of my brain and into real systems. Many thanks to Eric Cousineau from Toyota Research Institute and Sadra Saddini from Dexai Robotics for sharing code and ideas. Additionally, thank you to the staff of 6.100 for both funding my MEng and being a great team to work with. Lastly, I would like to acknowledge my academic advisor John Fisher and the EECS Comm Lab for their support in the writing process.

Thank you so much to my family, for unconditionally loving and supporting me in every way they can. I send special gratitude to my grandparents, who always believed in me and graciously provided for my education throughout my life. I am also so grateful for my best friends, Sam Nitz and Lydia Light, who have been my rock since we came to MIT, and Silvia Knappe, who has been my Course 6 pset partner and good friend throughout our undergrad and MEng. I am also thankful for the MIT Sport Taekwondo Team and the rest of my Taekwondo community. MIT would not have been the same without Taekwondo, and Taekwondo would not have been the same without all of you! Lastly, I could not have done this without my furry friends, Pita and Pocket.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As humans, we gain much of our understanding of the world by interacting with the objects in it. The first time somebody picks up a large pole, they may be surprised by either its mass or the amount of torque needed to rotate it. We gain intuition about the physics of rigid objects throughout our lives, and develop the ability to imagine how they may behave. A long-term vision in the robotics community is for robots to have this same collective intuition.

This work is motivated by the problem of *Real2Sim*. Named after the research area of *Sim2Real*, which closes gaps between the performance of simulated models and real robotic systems, Real2Sim is the problem of modeling real objects in simulation with realistic dynamics. As we enter an era where simulation is an essential tool for robots to make planning and control decisions, having efficient methods for porting real world data into simulation is crucial.

A simulation-ready asset of a rigid object requires different properties, depending on the task being simulated. Our approach to Real2Sim is to obtain these properties from observations of the world in an efficient manner, specifically using only cameras and robot sensors. For visualization, an asset generally requires geometry, often represented as either a set of geometric primitives or a mesh, although newer representations are coming online. Optionally, a texture and material properties can be added to the visual representation for rendering purposes. For the dynamics, the object needs both a collision geometry, to describe where it can interact with other

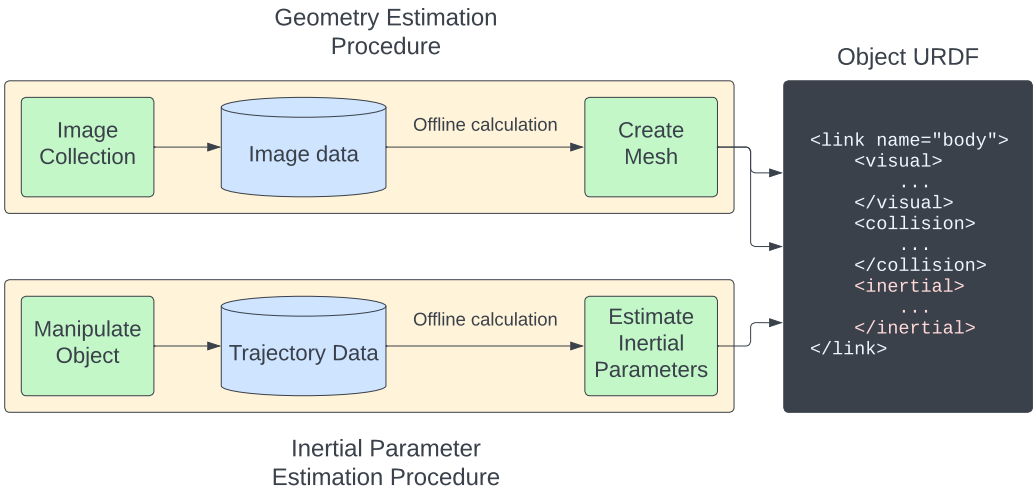

Figure 1-1: An example of a Real2Sim pipeline which estimates both the geometric and dynamic properties of an object for simulation. These properties are often stored in a Universal Robot Description Format (URDF).

objects, and *dynamic parameters* (or properties), which define how the object moves given applied forces at each time step. A subset of the dynamic parameters is the *inertial parameters*, which include the mass, center of mass, and inertia for a rigid object.

The envisioned pipeline is a robot arm, equipped with cameras, that first takes many images of an object in order to construct a mesh using 3D geometry tools. Then, the robot grasps the object, traversing some trajectory to gather torque and motion data. After using this data to identify the inertial parameters of the object, the result is an asset of the object that behaves similarly to the real object in simulation. In this work, we focus on the latter portion: estimating the inertial parameters.

Prior work in Real2Sim either makes assumptions about the object's mass distribution or estimates dynamic properties from video. However, robots are already suited for collecting data with sensors, and with a few extra steps, they could obtain physical, numerical information about the object. By using robot manipulators as measurement tools, we can begin to gather physical data about the world that can be used to generate assets for simulation. Once an object is created in simulation, the robot can imagine any arbitrary scenario involving that object. This is just one

example of the power Real2Sim technology can have on robot decision making.

In the next decade, as robotic manipulators begin to scale in industrial operations, and perhaps even grocery stores or homes, strong payload identification techniques could be capturing the properties of every previously unseen object, 24/7. This previously unavailable data could be used not only to create simulation-ready objects, but also to train many types of models, such as control policies or perception systems. For example, images of objects labeled with physical properties can train a model that can predict the dynamics of objects from visual information.

In this work we aim to understand and implement existing methods of robotic arm payload identification in the context of inertial property estimation for objects in Real2Sim pipelines. Specifically we consider rigid objects, due to their simple and well-understood properties and behavior, and methods that do not require additional force-torque sensors on a robot's wrist. We find that there advantages and disadvantages to different approaches, but the choice of identification procedure that is best for Real2Sim is that which can be seamlessly integrated with the geometric data collection steps.

There are a few questions we seek to answer. Firstly, of various payload identification methods presented in the literature, which would be best in a Real2Sim pipeline? Given a particular method, what trajectories are necessary for quality parameter estimates? Lastly, how tolerant can the estimation process be to noise from the robot's sensors? These questions will guide the engineering necessary to build a payload identification process into a Real2Sim system.

We begin with existing work in system identification for robotic manipulators, as the data we collect from the robot's sensors depends on the whole system. However, we find that it is challenging to implement without substantial engineering effort. Regardless, studying these works provide intuition for direct payload identification methods. When all that is needed is the payload identification, we propose an implementation of a method that requires no existing knowledge of the robot's system. We implement the preferred method of identification on the Franka Emika Panda hardware, and analyze various engineering strategies to improve upon the result.

Chapter 2

Preliminaries

Understanding where the inertial parameters appear in the equations used to describe the dynamics of robotic manipulators gives intuition for how they can be estimated. We first describe in detail the quantities we wish to estimate and show the patterns in which they exist in the models we use to describe and control these systems.

2.1 What Are the Inertial Parameters?

The *dynamic parameters*, sometimes referred to as the *physical parameters*, of a rigid body are the mass (m), center of mass ($\boldsymbol{c}$), inertia tensor ($\boldsymbol{I}$), and coefficients of friction ($\boldsymbol{\mu}$). In our setting, we will focus on the estimation of the *inertial parameters*, which includes mass, center of mass, and inertia, or the 0^{th}, 1^{st}, and 2^{nd} moments of mass, respectively. These should not be confused with the *geometric* or *kinematic* parameters that describe the shapes and links of a robotic system, such as the D-H parameters.

In this work, we will use the SI units of kilograms (kg) to describe mass and meters (m) to describe the center of mass. The inertia of a rigid body in three dimensions is often defined by an *inertia tensor*. The inertia tensor is a symmetric, positive-definite matrix containing six unique parameters, each with the SI units $kg \cdot m^2$. It can be calculated by integrating over the volume with respect to the mass distribution of the

object.

$$\boldsymbol{I} = \int_{\boldsymbol{x}\in\mathbb{R}^3} \phi(\boldsymbol{x})(\boldsymbol{x}^\top \boldsymbol{x}\mathbf{I}_3 - \boldsymbol{x}\boldsymbol{x}^\top)dm = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}. \tag{2.1}$$

Here, $\phi(\cdot)$ is a density field describing how mass is distributed across the volume, and $\mathbf{I}_3$ is the 3×3 identity matrix.

We also define the *unit inertia tensor* $\boldsymbol{G}$:

$$\boldsymbol{G} = \frac{\boldsymbol{I}}{m}. \tag{2.2}$$

We will occasionally use $m\boldsymbol{G}$ as an equivalent to $\boldsymbol{I}$ to emphasize that all of the inertial parameters appear in the equations of motion as bilinear terms with m.

For a 3-dimensional rigid body, the inertial parameters $\boldsymbol{\theta}$ are defined as

$$\boldsymbol{\theta} = [m,\, c_x,\, c_y,\, c_z,\, G_{xx},\, G_{yy},\, G_{zz},\, G_{xy},\, G_{xz},\, G_{yz}]. \tag{2.3}$$

An important note is that the inertial parameters, specifically the center of mass and inertia, must be defined with respect to some coordinate frame. Unless otherwise specified, they will be assumed to be about the origin frame of the respective body B, **not** about the body's center of mass frame, B_{cm}. Defining the parameters this way allows for special structure in the inverse dynamics equations, as we will see shortly. The mass parameter itself, being the 0^{th} moment, is invariant to frame of reference.

As further discussed below, the inertial parameters $\boldsymbol{\theta}$, except for mass, always appear in the structure of a rigid-body manipulator as bilinear terms with m. We will refer to these combinations as the *lumped inertial parameters*, or just *lumped parameters*, $\boldsymbol{\alpha}(\boldsymbol{\theta})$:

$$\boldsymbol{\alpha} = [m,\, mc_x,\, mc_y,\, mc_z,\, mG_{xx},\, mG_{yy},\, mG_{zz},\, mG_{xy},\, mG_{xz},\, mG_{yz}]. \tag{2.4}$$

Notice again that $m\boldsymbol{G}$ is equivalent to the entries of the inertia tensor.

In addition to the lumped parameters, it is also possible to describe a set of *base parameters* that are the minimum set of dynamic parameter combinations needed to describe the inverse dynamics of a system [3]. The base parameters are linear combinations of the lumped parameters, describing which lumped parameters cannot be identified independently of each other in practice.

2.1.1 Linearity of Composite Bodies

An important fact that we use in payload identification is that the inertial parameters of a composite body are the sum of the lumped inertial parameters in the bodies it is comprised of. Consider two rigid bodies B_1 an B_2, which are fixed with respect to each other, i.e. they are welded together. Each body has inertial properties $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, and lumped inertial properties $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$, respectively. They behave dynamically as one rigid body, B_{12}, which has it's own inertial properties and lumped inertial properties, $\boldsymbol{\theta}_{12}$ and $\boldsymbol{\alpha}_{12}$. Naturally, the mass m_{12} is equivalent to the sum of m_1 and m_2. The center of mass, with respect to a common reference frame F, is equivalent to the weighted sum of each individual center of mass vector, divided by m_{12}:

$$m_{12}\boldsymbol{c}_{12} = m_1\boldsymbol{c}_1 + m_2\boldsymbol{c}_2. \tag{2.5}$$

The same applies for the inertia, as long as both $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are defined with respect to the same frame F (this ensures that the *parallel axis theorem* holds):

$$m_{12}\boldsymbol{G}_{12} = m_1\boldsymbol{G}_1 + m_2\boldsymbol{G}_2. \tag{2.6}$$

Therefore, we see that the lumped parameters of the two bodies can be summed to obtain the lumped parameters of the composite body,

$$\boldsymbol{\alpha}_{12} = \boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2, \tag{2.7}$$

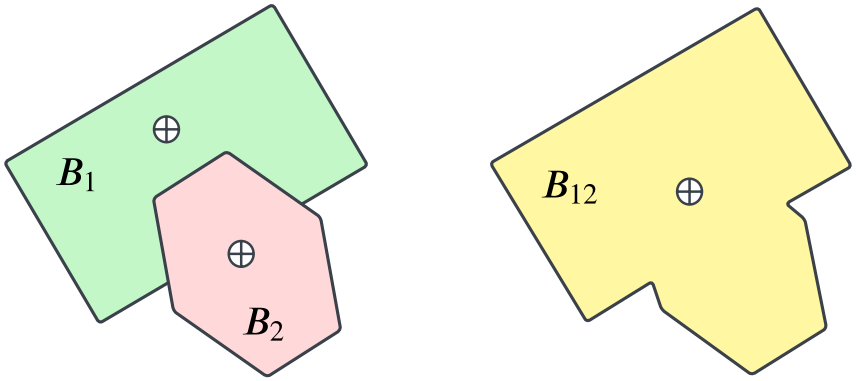

Figure 2-1: An example of a composite body comprised of two rigidly attached bodies. Their mass is summed, and the center of mass and inertia are weighted averages of the two, given the mass.

and more generally,

$$\boldsymbol{\alpha}_{\text{total}} = \sum_{i=1}^{n} \boldsymbol{\alpha}_i. \tag{2.8}$$

2.1.2 System Identification

Assume we know the inverse dynamics equations of a robotic system, parameterized by these unknowns, $\boldsymbol{\alpha}$:

$$\boldsymbol{\tau} = f(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}; \boldsymbol{\alpha}). \tag{2.9}$$

System identification solves the problem of finding an $\boldsymbol{\alpha}^*$ that minimizes the error between a set of observed states and torques ($\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, $\boldsymbol{\tau}$) and the model-predicted torques.

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \| f(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}; \boldsymbol{\alpha}) - \boldsymbol{\tau} \|_2^2. \tag{2.10}$$

In many methods, including those we explore in this work, this problem is solved via least-squares. However, it can also be solved with gradient or other non-linear optimization methods [4].

When performing system identification, $\boldsymbol{\alpha}$ typically represents a stacked combination of the inertial parameters for multiple bodies in a multi-body system. Here,

we distinguish between the inertial parameters of a robot arm $\boldsymbol{\alpha}_r$, and the inertial parameters of a payload $\boldsymbol{\alpha}_p$. We define $\boldsymbol{\alpha}_i$ for each link $i \in (1, N)$ as

$$\boldsymbol{\alpha}_i = [m_i,\ m_i c_{xi},\ m_i c_{yi},\ m_i c_{zi},\ I_{xxi},\ I_{yyi},\ I_{zzi},\ I_{xyi},\ I_{xzi},\ I_{yzi}], \tag{2.11}$$

and $\boldsymbol{\alpha}_r$ as the concatenated vector of $[\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_N]$. We state $\boldsymbol{\alpha}_r$ is a column vector of shape $(A \times N) \times 1$, where A is the number of lumped parameters per link and N is the number of links.

2.1.3 Constraints on Physical Feasibility

We aim to analyze when these parameters can and cannot be identified. Additionally, in order for these parameters to make sense physically, they must meet certain mathematical constraints of *physical feasibility*, also referred to as *physical consistency*. These constraints include $m > 0$, $\boldsymbol{I} \succ 0$, and the triangle inequality for the *principle moments of inertia*, or the three eigenvalues of the inertia tensor, I_1, I_2, I_3:

$$\begin{aligned} I_1 &\leq I_2 + I_3 \\ I_2 &\leq I_1 + I_3. \\ I_3 &\leq I_1 + I_2 \end{aligned} \tag{2.12}$$

Lastly, the inertia tensor must be symmetric.

2.2 The Equations of Motion

The inverse dynamics of a dynamic system are the differential equations that describe the forces acting upon the system given the system's state, it's derivatives, and the kinematic and dynamic parameters specific to the system. They are also referred to as the *equations of motion*. These are the equations that we want to study and understand if we want to estimate the dynamic parameters given observations of the state and forces acting upon it. Murray, et al. [5] and Long [6] are the resources we use to study and derive these equations, in order to give some intuition as to where

the dynamic parameters appear. Lastly, an important fact we want to highlight is that the equations of motion for a manipulator are linear in the lumped parameters, which was shown in [7] and [8].

2.2.1 One-Link Robot Case

First, we will analyze the equations of motion of a planar, single-link "robot arm" before moving to higher degrees of freedom. Lagrange's equations are a popular way of deriving the equations of motion, which begin from the kinetic and potential energy of the dynamic system. Take q to be the state of the system (in this case, the angle of the single joint, in radians), and $\dot{q}$ and $\ddot{q}$ to be the first and second derivatives, respectively. If $T(q, \dot{q})$ is the kinetic energy and $U(q)$ is the potential energy, then the *Lagrangian* is defined as

$$L(q, \dot{q}) = T(q, \dot{q}) - U(q). \tag{2.13}$$

Considering the case of a robot arm, we assume that all of the generalized forces acting on the robot can be written as joint torques, τ. This includes forces from the motors, gravity, and friction. Lagrange's equations state that

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \quad \forall i = 1, ..., N, \tag{2.14}$$

where N is the number of joints.

Because we are studying a rigid-body dynamic system, we can define T and U from the inertial properties of the arm. Let us abstract the values of mass, center of mass, and the inertia tensor as m, $\boldsymbol{c}$, and $\boldsymbol{I}$. In this case, $\boldsymbol{I}$ will be with respect to the center of mass, to simplify calculations, but $\boldsymbol{c}$ is with respect to the world frame. In our world frame of reference, x points to the right, z points up, and y points into the xz-plane. The single joint of the arm rotates about the y-axis. We can write the kinetic energy $T(q, \dot{q})$ as

$$T(q, \dot{q}) = \frac{1}{2} m \|\boldsymbol{v}\|^2 + \frac{1}{2} \omega^\top \boldsymbol{I} \omega, \tag{2.15}$$

where the norm of the translational velocity of the center of mass is $\|\boldsymbol{v}\| = \sqrt{\dot{x}^2 + \dot{z}^2}$ and the angular velocity is $\omega = \dot{q}$. We also have the following conversions from the joint angle q to the Cartesian coordinates, x, y, and z:

$$\begin{aligned} x &= r\cos q & z &= r\sin q \\ \dot{x} &= -\dot{q}r\sin q & \dot{z} &= \dot{q}r\cos q. \end{aligned} \tag{2.16}$$

r is the radial position of the center of mass from the world origin, which is equivalent to $\sqrt{x^2 + z^2}$ but will be written as r for simplicity. Now we write the kinetic energy as

$$T(q, \dot{q}) = \frac{1}{2}m(\dot{x}^2 + \dot{z}^2) + \frac{1}{2}I_{yy}\dot{q}^2. \tag{2.17}$$

Here, we only include the inertia about the y-axis, I_{yy}, since that is the axis of rotation of the joint. The body does not rotate about any other axis in this two-dimensional system. Further substituting, we obtain

$$\begin{aligned} T(q, \dot{q}) &= \frac{1}{2}m\left[r^2\sin^2 q \cdot \dot{q}^2 + r^2\cos^2 q \cdot \dot{q}^2\right] + \frac{1}{2}I_{yy}\dot{q}^2 \\ &= \frac{1}{2}\left[mr^2 + I_{yy}\right]\dot{q}^2. \end{aligned} \tag{2.18}$$

We can also define the kinetic energy in generalized coordinates. Using the generalized velocity vector $\boldsymbol{V}$ to describe the translational and rotational velocity, then

$$T(q, \dot{q}) = \frac{1}{2}\boldsymbol{V}^\top \begin{bmatrix} m\mathbf{I}_3 & -m \cdot \boldsymbol{c}\times \\ m \cdot \boldsymbol{c}\times & \boldsymbol{I} \end{bmatrix} \boldsymbol{V}, \tag{2.19}$$

where $\mathbf{I}_3$ is the 3×3 identity matrix and $\boldsymbol{c}\times$ is the skew-symmetric cross product operator for $\boldsymbol{c}$.

If we were strictly talking about the motion of the arm about it's origin, this would be sufficient. However, to describe the motion with respect to a body frame of a particular link, we will need to account for the kinematics of the system. If we want to describe the kinetic energy as a function of the joint states, then we must multiply

by the Jacobian that maps the joint angle to the generalized velocity vector:

$$\boldsymbol{V} = \boldsymbol{J}\dot{q}. \tag{2.20}$$

In the case of a single-link arm, where the base joint only rotates around the y-axis, the Jacobian is

$$\boldsymbol{J} = \begin{bmatrix} c_x & 0 & c_z & 0 & 1 & 0 \end{bmatrix}^\top, \tag{2.21}$$

and the kinetic energy of the system becomes

$$\begin{aligned} T(q,\dot{q}) =& \frac{1}{2}\dot{q} \begin{bmatrix} c_x \\ 0 \\ c_z \\ 0 \\ 1 \\ 0 \end{bmatrix}^\top \begin{bmatrix} m & 0 & 0 & 0 & mc_z & -mc_y \\ 0 & m & 0 & -mc_z & 0 & mc_x \\ 0 & 0 & m & mc_y & -mc_x & 0 \\ 0 & -mc_z & mc_y & I_{xx} & I_{xy} & I_{xz} \\ mc_z & 0 & -mc_x & I_{xy} & I_{yy} & I_{yz} \\ -mc_y & mc_x & 0 & I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \begin{bmatrix} c_x \\ 0 \\ c_z \\ 0 \\ 1 \\ 0 \end{bmatrix} \dot{q} \\ =& \frac{1}{2}\left[m(c_x^2 + c_z^2) + I_{yy}\right]\dot{q}^2. \end{aligned} \tag{2.22}$$

Clearly, these calculations become more complex with more linkages.

$U(q)$ is dependent on the angle of the joint, which gives the height of the center of mass:

$$U(q) = -mgr\sin q. \tag{2.23}$$

Now, we can form the Lagrangian,

$$\begin{aligned} L =& T - U \\ =& \frac{1}{2}(mr^2 + I_{yy})\,\dot{q}^2 + mgr\sin q, \end{aligned} \tag{2.24}$$

and the equations for the generalized forces via Eq. 2.14:

$$\begin{aligned} \tau =& \frac{d}{dt}\left[(mr^2 + I_{yy})\dot{q}\right] + mgr\cos q \\ =& (mr^2 + I_{yy})\ddot{q} + mgr\cos q. \end{aligned} \tag{2.25}$$

Notice that we see the term $(mr^2 + I_{yy})\ddot{q}$, which is not linear in the lumped parameters. However, remember that here, I_{yy} is the y-axis inertia **about the center of mass**. If we consider the *parallel axis theorem* and convert this representation of the inertia in the origin frame of the object, I_{yy}^B, then these non-linearities are eliminated, as

$$I_{yy}^B = I_{yy}^{B_{cm}} + m\|\boldsymbol{c}\|^2. \tag{2.26}$$

The resulting equation of motion is

$$\tau = I_{yy}\ddot{q} + m(c_x^2 + c_z^2)\mathrm{g}\cos q, \tag{2.27}$$

which is linear in $\boldsymbol{\alpha} = [mc_x, mc_z, I_{yy}]$.

We also notice that in this system, the mass term m does not appear in these equations independently from other inertial parameters. This means that we could not estimate the mass without already knowing the center of mass or the inertia. This makes sense intuitively, since there is no difference in torque produced between a heavy arm with a short center of mass and a lighter arm with a longer center of mass. This means that m is *unidentifiable*. The same applies to $\boldsymbol{c}$ and the unit inertia tensor $\boldsymbol{G}$. However, the lumped parameters mc and $m\boldsymbol{G}$ are identifiable, and in fact, they are the only inertial terms that matter when describing these equations of motion.

2.2.2 Two-Link Robot Case

Now, we examine the derivation of a two-link planar arm in order to see where the rest of the inertial parameters appear in the equations. This example is mainly for building intuition, there is no proof of the linearity in the lumped inertial parameters.

The structure of the equations now relies on the kinematic relationship between links 1 and 2. In the planar case, this is simply a translational offset dependent on the length of link 1, ℓ_1. Since these kinematic parameters are known in the system, we

see the term $m_2\ell_1$ appear in the equations of motion, allowing for the identification of m_2.

Indeed, we only need to derive the potential energy of the system to show where m_2 appears independently, although it also appears in the kinetic energy expressions. The potential energy is equivalent to

$$U(q_1, q_2) = U_1(q) + U_2(q_1, q_2), \tag{2.28}$$

$$U(q_1, q_2) = -m_1 g r_1 \sin q_1 + m_2 g(\ell_1 \sin q_1 + r_2 \sin(q_1 + q_2)). \tag{2.29}$$

Here we see the term $m_2 g \ell_1 \sin q_1$ which does not depend on any inertial parameters except for m_2. Hopefully this example provides some intuition for how the inertial parameters begin to enter the equations of motion as more links are added. More rigorous examples can be found in [5] and [6].

2.2.3 Linearity in Lumped Parameters

When the Lagrangian equations of motion for an N-link manipulator are derived, they take the form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \boldsymbol{\tau}_g(\mathbf{q}) = \boldsymbol{\tau}, \tag{2.30}$$

where $\mathbf{M}(\mathbf{q})$ is the generalized mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ contains the Coriolis and centrifugal terms, and $\boldsymbol{\tau}_g(\mathbf{q})$ are the generalized forces due to gravity. Atkeson, et al. (1985) [7] shows how the parameters of the terminal link are linear in the lumped parameters (Eq. 2.1), and An, et al. [8] extends this to the full robot case. Gautier and Khalil also propose the same result in 1988 [9].

This is a very important result that has implications for how the system identification problem can be solved; as we see later, this linear relationship that it specific to robot manipulators allows the problem to be solved via least-squares optimization, which is convex.

Chapter 3

Related Work

Highlighted here are works from both a novel research area, Real2Sim, and a well established research area, system identification. We note gaps in the Real2Sim literature with respect to dynamic parameter identification and pull from existing methods to fill them.

3.1 Real2Sim

Solving the Real2Sim problem is a topical research area, gaining traction with the rise of virtual reality and advances in computer vision and graphics. It encompasses a wide variety of tasks, all with the shared objective of representing a scene from the real world in a simulator. For example, an earlier work in Real2Sim involves modeling pegs from real, factory insertion tasks to train reinforcement learning policies [10]. The authors make use of generative adversarial networks to create a simulated model of the real scene, then allow the robot to learn in simulation.

Soon after, we see one of the most important contributions to computer vision and graphics in the past 3 years that has inspired Real2Sim research, Neural Radiance Fields (NeRFs) [11]. A NeRF is a neural network representation of a scene or object, where the inputs are 3D points at a desired viewing angle, and the outputs are RGB values with a given optical density, describing how much light is transmitted through that point. By using a differentiable volumetric renderer, the result is an image

of the scene at any queried viewpoint. They can be trained end-to-end on sets of RGB images, creating a virtually three dimensional representation of the scene in the images. In order to be simulated, a NeRF can be converted into a traditional mesh via the marching cubes algorithm [12] or simulated directly in a differentiable simulator.

Le Cleac'h et al. understands that neural fields have a capacity to not only encapsulate the way that light behaves, but also how mass is distributed on a object [13]. However, their method does not take advantage of real-world measurement to gather dynamic information, and instead uses video to learn the expected dynamics of the object. Additionally, learning a density neural field has many challenges, mainly that it is very over-parameterized given we can otherwise fully represent the dynamics of a rigid body using only 10 parameters. PAC-NeRF [14] is another work which expands neural field representations to contain dynamic information.

Heiden, et al. (2019) is very similar to our work, in that they estimate physical properties of mechanical systems for Real2Sim transfer [15]. However, they use motion capture techniques to collect data for system identification, where we use data directly from a robot's sensors. We are not aware of Real2Sim solutions that involve mechanical payload identification methods at this time.

3.2 Object Inertial Parameter Estimation

Identifying payloads of robotic manipulators has been well studied since the mid 1980s. This section will cover a subset of important works in this field as the literature is vast. Often, the method is derived as an extension of system identification techniques of robotic manipulators as a whole, so we will review these works as well.

3.2.1 Estimation as a Free Body

When presented with a standalone, rigid object, there are quite a few ways to estimate its inertial properties. Mass can be measured with a scale, center of mass via measuring where the object can balance, and inertia by analyzing motions when

hung from a pendulum [16]. However, when our goal is to identify these parameters of many unseen objects at scale, it is not efficient and difficult to automate. Mavrakis and Stolkin provide a thorough review of alternative methods, categorized by means of visual information and physical information [17]. They also describe methods for objects rigidly fixed to a robot manipulator, which we will discuss more below.

Some notable visual methods include Galileo [18], which identifies the physical properties of objects from video, and image2mass [19], which can estimate the mass of an unseen object, trained on thousands of examples of Amazon merchandise. What is impressive about these visual methods is that they are comparably accurate to a human performing the same physical parameter estimation task. Regardless, having a robot involved in the estimation process could allow us to do much better, given that they can take physical measurements.

Methods in which a manipulator is interacting with an object as a free body have also been proposed. For example, Yu, et al. aims to identify all of the inertial parameters of a robot via pushing with a 2-finger gripper [20]. However, their method requires specialized force sensor hardware at the fingertips, and many interactions are needed in order to identify the different moments. Mavrakis, et al. also focuses on planar pushing, which is bound to the xy-plane, meaning many inertial parameters are destined to be unidentifiable [21]. Overall, free body object identification is helpful for identifying dynamic parameters such as friction coefficients, but there are limitations in inertial parameter identification.

3.2.2 Estimation as a Payload (Extension of Robot Arm)

Rather than identifying the parameters of a free body, an object can be treated as an extension of a mechanical system, or in the case of a robotic manipulator specifically, the object is referred to as the payload. Performing system identification for a payload's dynamic parameters specifically is often referred to as *payload identification*. One of the first cited works on payload identification describes a method for mass estimation in a static setting, both using the joint torques on the manipulator and with a force sensor attached to the robot's wrist [22]. Soon after, methods were

developed to identify center of mass and inertial of the payloads, using specified test motions to measure one axis of motion at a time [23].

Atkeson, et al. (1985) contributed to the payload identification literature by applying the Newton-Euler equations, which combine the transitional and rotational components of the payloads dynamics into a single equation that can be used for identification [7]. As mentioned previously, they note that these equations are linear in the inertial parameters they wish to identify, and therefore the parameters can be estimated with the least-squares method. They tested this method on a real robotic system and handled sensor noise and other issues arising from experimental data.

Khalil, et al. presents four different least-squares based methods that can be used to identify a fixed payload, meaning that the object is rigidly attached to the end of the arm [24]. This regime is applicable not only when a robotic manipulator is wielding a tool, but also when it is grasping an object. One of the methods shows that the parameters of a payload can actually be identified by using the difference in torque measurements taken when the robot is carrying the payload and when it is not. A drawback to this method is that it requires both of these measured trajectories to be identical. Therefore, it may be more reasonable to assume a good dynamic model of the robot, if available. We study this method carefully in our work, and find it quite useful for the Real2Sim task.

In later years, Gaz, et al. (2017) revisit this problem and explore questions regarding the identifiability of the payload parameters after the base dynamic parameters of the robot are already identified [25]. They show that the payload parameters appear linearly in the dynamic parameters of the robot and can thus be estimated in the same least-squares approach. An interesting finding was that any estimate of the payload parameters could be used with any feasible estimate of the robot parameters and dynamically consistent results would be obtained.

3.3 Robot Dynamic Parameter Estimation

As pointed out by Khalil, et al. (2007), many payload identification methods require knowing either estimating or knowing the dynamic model of the robot *a priori*. This is because a rigid payload can be treated as the extension of a manipulator's terminal link, and having a strong underlying model of the robot allows us to extract the parameters of a payload from unexpected changes in the robot's joint torques. If this model is not available, which it often is not, the problem of estimating the dynamic parameters of the whole robotic system is necessary.

In the 1980s, multiple papers ground work for dynamic parameter estimation on robots [9][3][26][27]. An et al. proposes one of the methods we apply in this work. This paper points out and addresses the issues of unidentifiability in certain parameters and subtleties to the least-squares optimization that is ignored in prior work [8]. Going forward, this foundational method of system identification for robots is used widely in research and industrial settings.

3.3.1 Identifying Industrial Robots

The main motivation of identifying robotic systems is that when utilizing a robot built by an external manufacturer, the user does not have access to the dynamic model of the robot. These numbers are proprietary, but without them, it is challenging to build precise controllers. Thus, there is a paper for the identification of almost every commonly used robotic manipulator, and we will highlight a few examples.

The PUMA 560 arm was one of the first to have it's dynamic parameters estimated in the 1980s; in the work of Armstrong, et al. it was deconstructed completely to have the individual links measured [28]. More recently, Gaz, et. al (2019) aims to estimate the Franka Emika Panda robot [1], which we studied with care as it is the model that we chose for our real-world experiments. First, they find the optimal value of the base parameter set, via least-squares optimization. Then, they use a global optimization method to estimate a set of parameters from the optimal base parameters. The results are currently one of the most widely accepted Franka Emika Panda models

available for simulation and control, notably being used by the ROS interface to the Panda [29]. However, this estimate does not appear to be physically realistic, meaning that while the parameter estimates might produce the correct expected torques, they may not be the same as the actual values of the hardware. The popular KUKA iiwa LBR robot was analyzed by Stürs, et al [30]. They also use the least-squares optimization approach, and we found this work helpful when originally performing early experiments with a simulated KUKA robot.

3.3.2 Identifiability Analysis

Robotic system identification has been heavily studied, especially in the domain of robotic manipulators. Multiple robotic manipulation textbooks and papers discuss the use of rigid body dynamics analysis to find a set of base parameters that are suitable for identification in a system, and specifically for manipulators with rotational joints [16][22][31][32]. The correct set of base parameters allows the least-squares problem to be full-rank. Typically, these are numeric methods that involve analyzing either the QR or SVD compositions of the data matrices used in the least-squares estimation.

3.3.3 Trajectories for Excitation

Solving the dynamic estimation problem requires quality data collection. Early works use a sequence of isolated joint trajectories in order to estimate one link at a time [27]. Once the method of solving for the parameters via the Newton-Euler equations was proposed, the regression could be performed on all parameters simultaneously given a data set formed from all joint trajectories. We mostly see sinusoidal trajectories used in the literature; while we could not find the original proposition of this data collection technique, it likely stems from modal analysis of linear dynamical systems.

However, Rackl et al. performs an optimization for excitability over trajectories parameterized as B-splines [33]. Many types of trajectories can be parameterized in this way, but the ones found for excitation appeared to look very similar to sinusoidal

trajectories. This demonstrates that sinusoidal trajectories are good for excitation.

3.3.4 Estimation with Physical Feasibility

The identification techniques described above may give robot parameter estimates that mathematically evaluate to the correct dynamics, but the parameters themselves may not be physically feasible. To be physically feasible, the physical parameters must satisfy a series of constraints described in Section 2.1.3.

Previously mentioned, Gaz, et. al (2019) tackles the problem of physical feasibility on a Franka Emika Panda arm by performing an optimization in two steps [1]. First, they find the optimal value of the base parameter set, via least-squares optimization. Then, they use simulated annealing to estimate a set of physically feasible parameters from the optimal base parameters given the necessary constraints, a non-linear optimization problem. The resulting parameter values are currently one of the most widely accepted Panda models available for simulation and control, notably being used by the ROS interface to the Panda [29]. However, there is no guarantee on physical reality of these estimates, meaning that they may not be close to the values obtained by measurement techniques, such as deconstructing each link and analyzing them individually. That is because there are many possible solutions that produce the same inverse dynamics. This can be observed especially in the large difference in mass between links 1 and 2, which have equivalent shape and are likely to be very close in mass in reality (Figure 3-1).

Wensing, et al. encodes these constraints as Linear Matrix Inequalities and utilizes semi-definite programming to find the optimal solution [34]. Sutanto, et al. provides an alternative to this approach by encoding the constraints directly into the equations of motion via reformulation of the dynamic parameters [35]. More recently, Ledezma and Haddadin try a similar global optimization; however, instead of performing a non-linear optimization with constraints, they instead show that the set of physically feasible solutions lie on a Riemannian manifold, and perform gradient descent on that manifold [2].

There has been plenty of focus on physical feasibility for robot parameter esti-

$\boldsymbol{\theta}$	Link 1	Link 2	Link 3	Link 4	Link 5	Link 6	Link 7
m_i	4.9707	0.6469	3.2286	3.5879	1.2259	1.6666	1.4655
$m_i X_i$	0.0193	-0.0020	0.0888	-0.1908	-0.0147	0.1002	0.0004
$m_i Y_i$	0.0103	-0.0186	0.1267	0.3746	0.0503	-0.0235	-0.0031
$m_i X_i$	-0.4654	0.0023	-0.2147	0.0985	-0.0471	-0.0175	0.1453
XX_i	0.7470	0.0085	0.0565	0.0677	0.0394	0.0025	0.0308
XY_i	-0.0002	-0.0040	-0.0082	0.0277	-0.0015	0.0015	0.0004
XZ_i	0.0086	0.0103	-0.0055	0.0039	-0.0046	-0.0001	-0.0007
YY_i	0.7503	0.0281	0.0529	0.0324	0.0315	0.0106	0.0284
YZ_i	0.0201	0.0008	-0.0044	-0.0016	0.0022	0.0001	-0.0005
ZZ_i	0.0092	0.0265	0.0182	0.0776	0.0109	0.0118	0.0067

$\boldsymbol{\theta}$	Link 1	Link 2	Link 3	Link 4	Link 5	Link 6	Link 7
m_i	1.000	2.6473	3.1301	2.4053	1.8933	1.9949	1.4170
$m_i X_i$	0.000	-0.0143	0.1289	-0.1090	-0.0209	0.1173	0.0013
$m_i Y_i$	0.000	-0.3686	-0.0052	0.3392	0.0289	-0.1009	0.0139
$m_i X_i$	0.000	0.0673	-0.4106	-0.0002	-0.3443	-0.0451	0.0774
XX_i	1.006	0.0998	0.1051	0.1067	0.1078	0.0182	0.0161
XY_i	0.000	0.0049	0.0000	0.0300	0.0003	0.0066	-0.0001
XZ_i	0.000	-0.0029	0.0018	0.0000	-0.0035	0.0030	-0.0028
YY_i	1.006	0.0348	0.1179	0.0250	0.1065	0.0190	0.0183
YZ_i	0.000	0.0161	-0.0031	-0.0022	0.0044	-0.0022	-0.0006
ZZ_i	0.012	0.1020	0.0249	0.1169	0.0099	0.0243	0.0099

Figure 3-1: Estimated inertial parameters of the Franka Panda robot and associated inertial ellipsoid visualizations. On the left are estimates from Gaz, et al (2019) [1], and on the right are estimates from Ledezma and Haddadin [2]. Courtesy of Fernando Díaz Ledezma and Sami Haddadin. RIL: Riemannian Incremental Learning of the Inertial Properties of the Robot Body Schema. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* ©2021.

mation, but we could not find previous literature analyzing physical feasibility for payload parameter estimations exclusively. We assume that this means it is a non-issue for well-structured payload identification objectives, but refer to these works in robot identification when trying to understand unsatisfactory results.

While the theory of robotic arm system identification is strong, there are many implementation subtleties that make the problem challenging. The main motivation for most of this work is for better controller design, which means obtaining a dynamically consistent model with a margin of error suitable for feedback control. This does not always mean that we need to produce physically consistent or even physically accurate results. Fortunately, there are payload estimation techniques that do not require the estimation of the whole robot system.

Chapter 4

Methods

In this section, we present two main approaches to payload identification that we explored and share additional engineering details on implementation. The first, *full-arm symbolic decomposition*, involves learning the dynamic parameters of an entire robot arm, not just the payload. However, we later see that the *direct payload identification* method only estimates the payload parameters. The quality of data used for estimation also affects the quality of the solution, so we address design questions regarding the types of trajectories executed to collect data from the robot and post-processing to remove noise from the data.

An overview of each method, with their differences and similarities, is outlined in Table 4.1. Python implementations are available at:
https://github.com/alambert14/real2sim-payload-id [36].

Method	**Objective: $\arg\min_{\boldsymbol{\alpha_p}}$**	**# Parameters**	**Needs $\boldsymbol{\alpha}_r$?**	**Assumptions**
Full-Arm Symbolic Decomposition	$\|\mathbf{W}(\boldsymbol{\alpha}_r + \boldsymbol{\alpha}_p) - \boldsymbol{\tau}\|_2^2$	$(N+1) \times 10 +$ Friction	✓	Robot parameter estimates are physically consistent and realistic
Direct Payload Identification	$\|\mathbf{W}_p\boldsymbol{\alpha}_p - \Delta\boldsymbol{\tau}\|_2^2$	10	✗	Two trajectories have equivalent positions and velocities

Table 4.1: Comparison of both payload identification methods.

4.1 Method 1: Full-Arm Symbolic Decomposition

The full-arm symbolic decomposition method makes use of the linearity of the lumped inertial parameters in a robotic arm's equations of motion. It starts with either an estimation of the dynamic parameters of the robot's model, or the assumption that the parameters are already available.

We began exploring this work by the assumption that the robot's dynamic parameters would be made available by the manufacturer, but this is not the case for most industrial and research robots. The Panda robot, which we choose to use for our experiments, does not expose the dynamic parameters used in its control algorithms, nor parameters directly from the manufacturer. Its API exposes some functions that could help us extract these parameters, such as $\boldsymbol{\tau}_{\text{ext}}(\cdot)$, the "external" torques calculated from the measured joint torques minus expected torques from the internal model. This includes commanded joint torques and gravity compensation torques. However, we ran experiments to see if these external torques can be trusted and found them to be inaccurate. We observed that the external torques were much larger than zero when the robot was undisturbed. Regardless, because our Panda has elements that are unique to it, such as a camera, end-effector, and wires, re-calibrating the model to fit this specific robot is necessary. This motivated our primary problem of estimating the Panda's dynamic parameters.

4.1.1 Estimating the Robot's Dynamic Parameters

Assuming we know the kinematic and geometric parameters of the robot, we recall from Chapter 2 that the equations of motion defining the system's dynamics are

$$\mathbf{M}(\mathbf{q};\boldsymbol{\alpha}_r)\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}};\boldsymbol{\alpha}_r)\dot{\mathbf{q}} - \boldsymbol{\tau}_{\text{g}}(\mathbf{q};\boldsymbol{\alpha}_r) = \boldsymbol{\tau} \tag{4.1}$$

and are linear in the lumped inertial parameters of the robot, $\boldsymbol{\alpha}_r$, which we will further refer to as simply the lumped parameters.

By logging one or more trajectories from the robot, we obtain $\boldsymbol{\mathcal{D}}$, a dataset con-

sisting of

- $\mathbf{q}[\cdot]$: The measured joint angles (radians)
- $\dot{\mathbf{q}}[\cdot]$: The measured or calculated joint velocities (rad/s)
- $\ddot{\mathbf{q}}[\cdot]$: The calculated joint accelerations (rad/s^2)
- $\boldsymbol{\tau}[\cdot]$: The measured joint torques (Newton-meters)

At a given time t, each of these components give a vector in $\mathbb{R}^N$. With these known sequences, the only remaining quantities are the dynamic parameters we wish to estimate, $\boldsymbol{\alpha}_r$. Our goal is to find an $\boldsymbol{\alpha}_r$ which minimizes the torque estimation error, meaning our model is dynamically accurate:

$$\boldsymbol{\alpha}_r^* = \arg\min_{\boldsymbol{\alpha}_r} \|\mathbf{M}(\mathbf{q};\boldsymbol{\alpha}_r)\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}};\boldsymbol{\alpha}_r)\dot{\mathbf{q}} - \boldsymbol{\tau}_{\mathrm{g}}(\mathbf{q};\boldsymbol{\alpha}_r) - \boldsymbol{\tau}\|_2^2. \tag{4.2}$$

4.1.2 Friction Modeling

Unfortunately the inertial parameters are not enough for a model of a serial manipulator that will give reasonable model-predicted torques. There is also friction between the joints that dampens their motion. The challenge here is that there are many ways to model friction; however, the coefficients of friction still appear linearly in the equations of motion and are decoupled from the lumped inertial parameters.

One way of modeling friction is with 3 coefficients, described in [1]:

$$\boldsymbol{\tau}_\mu(\dot{\mathbf{q}}) = \boldsymbol{\mu}_v\dot{\mathbf{q}} + \boldsymbol{\mu}_c\mathrm{sign}(\dot{\mathbf{q}}) + \boldsymbol{\mu}_o. \tag{4.3}$$

$\boldsymbol{\mu}_v$ represents viscous friction, $\boldsymbol{\mu}_c$ represents Coloumb friction, and $\boldsymbol{\mu}_o$ is friction offset that encapsulates any unmodeled behavior. If $\boldsymbol{\tau}_\mu$ is considered with the equations of motion (Eq. 4.1),

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \boldsymbol{\tau}_{\mathrm{g}}(\mathbf{q}) - \boldsymbol{\tau}_\mu(\dot{\mathbf{q}}) = \boldsymbol{\tau}, \tag{4.4}$$

then $\boldsymbol{\mu}_v$, $\boldsymbol{\mu}_c$, and $\boldsymbol{\mu}_o$ may be appended to $\boldsymbol{\alpha}_r$, which now represents the lumped dynamic parameters.

4.1.3 Linear Decomposition and Data Matrix Construction

Since the equations of motions are linear in $\boldsymbol{\alpha}_r$, we can symbolically rewrite our objective in this form:

$$\boldsymbol{\alpha}_r^* = \arg\min_{\boldsymbol{\alpha}_r} \|\mathbf{W}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\alpha}_r - \boldsymbol{\tau}\|_2^2. \tag{4.5}$$

$\mathbf{W}$ is referred to as the *data matrix*, which is dependent on our kinematics variables $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$.

There are two tools in Drake that we can use to implement this decomposition, discussed below [37].

Symbolic Decomposition

One option is the function `DecomposeLumpedParameters` from Drake's symbolic toolkit. This recursive function takes a representation of a symbolic expression and a vector of symbolic parameters, $\boldsymbol{\theta}$, and returns the data matrix and a set of lumped parameters as they appear in the symbolic expression, $\boldsymbol{\alpha}(\boldsymbol{\theta})$. Since this algorithm runs agnostic to the structure of the multibody equations, it is not guaranteed that the lumped parameters that are identified will be the same set that we describe in Eq. 2.1. However, this is a good choice if we have a system where we do not know what the set of lumped parameters would be.

Ideally in computation, $\mathbf{W}$ remains symbolic, containing placeholders for the kinematics variables $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$. Unfortunately, at the time of this writing, the decomposition algorithm does not scale to systems with high degrees of freedom and takes prohibitive computation time when presented with the equations of motion of the 7-degree-of-freedom robotic arm we are interested in. Despite this issue, the problem can be made tractable. We found that substituting the kinematics variables with numeric values drastically reduces the number of expressions needed to represent the

equations, as many of the terms are able to be regrouped into numeric coefficients.

Instead of performing the decomposition once with all of symbolic kinematics variables as placeholders, we can decompose the expression at each time step after substituting the collected kinematic data. At each time step t, we collect $\mathbf{q}(t)$, $\dot{\mathbf{q}}(t)$, and $\ddot{\mathbf{q}}(t)$, and then decompose to obtain $\mathbf{W}_t$. With T total data points, we create a stacked vector $\mathbf{W}$ of size $(T \times N) \times A$. An issue with this approach is that A could change at each time step, for example, if some values of the kinematics observations cause the coefficients to become zero. We simply remove these data points if the decomposition of $\boldsymbol{\alpha}_r$ does not contain the parameters we expect, which is fortunately uncommon.

For a simple system with low degrees of freedom, we only need to decompose the equations of motion once, with a fully symbolic representation of the kinematics and the inertial parameters. However, on a real robotic arm with 6 or 7 joints, it is necessary to add the substitution step, as shown in Algorithm 1.

Algorithm 1 Symbolic Decomposition with Kinematics Substitution

Given: $\boldsymbol{\mathcal{D}}$, `plant`, $\boldsymbol{\theta}_{\text{sym}}$
$\mathbf{W}_{\text{data}} \leftarrow zeros(T \times N, A)$
$k \leftarrow 0$
for $t \leftarrow 1$ to T **do**
  $\mathbf{X}$, $\boldsymbol{\tau}_{\text{obs}} \leftarrow \boldsymbol{\mathcal{D}}[t]$ ▷ Observed $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \boldsymbol{\tau}$ at time t
  $\boldsymbol{\tau}_{\text{sym}} \leftarrow$ `plant.InverseDynamics`$(\mathbf{X})$
  $\mathbf{W}_t$, $\boldsymbol{\alpha}_{\text{sym}} \leftarrow$`DecomposeLumpedParameters`$(\boldsymbol{\tau}_{\text{sym}}, \boldsymbol{\theta}_{\text{sym}})$
  if $\boldsymbol{\alpha}_{\text{sym}}$ is invalid **then**
    continue
  end if
  $\mathbf{W}_{\text{data}}[k : k+N, :] \leftarrow \mathbf{W}_t$
  $\boldsymbol{\tau}_{\text{data}}[k : k+N] \leftarrow \boldsymbol{\tau}_{\text{obs}}$
  $k \leftarrow k + N$
end for
return $\mathbf{W}_{\text{data}}$, $\boldsymbol{\alpha}_{\text{sym}}$, $\boldsymbol{\tau}_{\text{data}}$ ▷ $\mathbf{W}_{\text{data}}\boldsymbol{\alpha}_{\text{sym}} = \boldsymbol{\tau}_{\text{data}}$

Auto-Differentiation

Another implementation is based on the observation that $\mathbf{W}$ describes the partial derivatives of the equations of motion with respect to $\boldsymbol{\alpha}_r$:

$$\mathbf{W} = \frac{\partial}{\partial \boldsymbol{\alpha}} \left[\mathbf{M}(\mathbf{q}; \boldsymbol{\alpha}_r)\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}; \boldsymbol{\alpha}_r)\dot{\mathbf{q}} - \boldsymbol{\tau}_g(\mathbf{q}; \boldsymbol{\alpha}_r) \right]. \tag{4.6}$$

Thus, $\mathbf{W}$ can also be constructed using auto-differentiation, which we implement using the `AutoDiff` functionality in Drake. Because the equations are linear in $\boldsymbol{\alpha}_r$, the partial derivatives are constant with respect to $\boldsymbol{\alpha}_r$. Essentially, $\mathbf{W}$ describes how much we expect the torque $\boldsymbol{\tau}$ to change when we change $\boldsymbol{\alpha}_r$. For this reason, $\mathbf{W}$ is sometimes referred to as the *sensitivity tensor*.

Here, we construct the inverse dynamics using `AutoDiff` variables to represent the lumped parameters, telling Drake that we can find the gradient with respect to these variables. They are initialized to arbitrary values, since the derivative is not dependent on the values of $\boldsymbol{\alpha}_r$. This gives an expression $\mathbf{e}_t$ at each time step, after substituting in the recorded kinematics values. $\mathbf{W}$ is found by stacking the partial derivative vectors of each $\mathbf{e}_t$ with respect to $\boldsymbol{\alpha}_r$, creating a matrix of size $(T \times N) \times A$.

Now that $\mathbf{W}$ has been constructed, this minimization problem is a candidate for least-squares optimization. The least-squares solution to this problem is

$$\boldsymbol{\alpha}_r = (\mathbf{W}^\top \mathbf{W})^{-1} \mathbf{W}^\top \boldsymbol{\tau}, \tag{4.7}$$

or

$$\boldsymbol{\alpha}_r = [\mathbf{W}]^+ \boldsymbol{\tau} \tag{4.8}$$

if we use the pseudo-inverse notation [38]. Note that the original dynamic parameters, $\boldsymbol{\theta}_{1\ldots N}$ can be recovered by the associated estimated $m_{1\ldots N}$.

4.1.4 Using an Initial Condition

Another advantage of the linearity in $\boldsymbol{\alpha}_r$ is that we can estimate $\boldsymbol{\alpha}_r$ with respect to an initial guess, $\boldsymbol{\alpha}_0$:

$$\boldsymbol{\tau} - \boldsymbol{\tau}_0 = \mathbf{W}(\boldsymbol{\alpha}_r - \boldsymbol{\alpha}_0). \tag{4.9}$$

This time, instead of solving for a new set of parameters, we solve for a delta $\boldsymbol{\delta}_{\boldsymbol{\alpha}}$ that describes the difference between $\boldsymbol{\alpha}_0$ and the $\boldsymbol{\alpha}_r^*$, the optimal $\boldsymbol{\alpha}$ that most accurately describes the system. Now, the bias vector is the difference between our observed torques $\boldsymbol{\tau}$, and the model-predicted torques of the $\boldsymbol{\alpha}_0$ system, $\boldsymbol{\tau}_0$.

$$\boldsymbol{\delta}_{\boldsymbol{\alpha}}^* = \arg\min_{\boldsymbol{\delta}_{\boldsymbol{\alpha}}} \|\mathbf{W}(\boldsymbol{\alpha}_{\boldsymbol{r}} - \boldsymbol{\alpha}_0) - (\boldsymbol{\tau} - \boldsymbol{\tau}_0)\|_2^2, \tag{4.10}$$

$$\boldsymbol{\delta}_{\boldsymbol{\alpha}}^* = [\mathbf{W}]^+ (\boldsymbol{\tau} - \boldsymbol{\tau}_0), \tag{4.11}$$

$$\boldsymbol{\alpha}_r^* = \boldsymbol{\alpha}_0 + \boldsymbol{\delta}_{\boldsymbol{\alpha}}^*. \tag{4.12}$$

In theory, we could choose any initial condition for $\boldsymbol{\alpha}_0$, but we choose to existing guesses for the Panda parameters, such as those in [1] and [2]. The advantage of using an initial condition is that we can add regularization to our least-squares problem, such as a ridge regularizer, that would keep the solution close to the initial guess:

$$\boldsymbol{\delta}_{\boldsymbol{\alpha}}^* = (\mathbf{W}^\top \mathbf{W} + \lambda \mathbf{I})^{-1} \mathbf{W}^\top (\boldsymbol{\tau} - \boldsymbol{\tau}_0). \tag{4.13}$$

This is beneficial when estimating the entire robot, as there are many possible solutions that are dynamically consistent.

4.1.5 Estimating the Inertial Parameters of the Payload

This method, when evaluated on data collected from a robot not carrying a payload, gives us an estimate of the robot's inertial parameters, $\hat{\boldsymbol{\alpha}}_r$. If we consider the payload to be a component of a composite body with the terminal link of the robot arm, then from Section 2.1.1 it is shown that the inertial parameters of the composite body are

equivalent to the sum of he two bodies it consists of, and thus

$$\hat{\boldsymbol{\alpha}}_p = \hat{\boldsymbol{\alpha}}'_N - \hat{\boldsymbol{\alpha}}_N. \tag{4.14}$$

$\hat{\boldsymbol{\alpha}}_N$ are the previously estimated (or known *a priori*) parameters of the terminal link when there is no payload, and $\hat{\boldsymbol{\alpha}}'_N$ is the estimated value of the last link while the robot is carrying the payload.

Going forward, we will use the notation $\boldsymbol{\alpha} = \boldsymbol{\alpha}_r + \boldsymbol{\alpha}_p$ to describe this relationship, as

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \vdots \\ \boldsymbol{\alpha}_N \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \boldsymbol{\alpha}_p \end{bmatrix}, \tag{4.15}$$

where $\boldsymbol{\alpha}_1 ... \boldsymbol{\alpha}_N$ are the lumped parameters of each link.

Implementation wise, there are two options for enforcing the linear relationship between $\boldsymbol{\alpha}_N$ and $\boldsymbol{\alpha}_p$:

1. **Re-estimate all robot parameters $\boldsymbol{\alpha}_r$:** If we proceed with the same procedure as before to obtain a new estimate of $\hat{\boldsymbol{\alpha}}_r$, then the contribution by $\boldsymbol{\alpha}_p$ can be extracted by subtracting the old estimate of $\hat{\boldsymbol{\alpha}}_r$ from the new estimate. An issue with this method is that if there is error in the initial estimate of $\hat{\boldsymbol{\alpha}}_r^N$, it will be propagated to the estimate of the payload.

2. **Create new variables $\boldsymbol{\alpha}_p$ and sum with $\boldsymbol{\alpha}_N$:** Previously, we represented the inertial parameters as variables (either symbolic or `AutoDiff`) in Drake's `MultibodyPlant`. Here, we create a new version of the plant, initialized with the previously estimated $\hat{\boldsymbol{\alpha}}_r$. Then, $\hat{\boldsymbol{\alpha}}_N$ is replaced by $\hat{\boldsymbol{\alpha}}_N + \boldsymbol{\alpha}_p$, and the equations are decomposed, this time with respect to the symbolic $\boldsymbol{\alpha}_p$:

$$\boldsymbol{\tau} = \mathbf{W}_p \boldsymbol{\alpha}_p + \mathbf{w}. \tag{4.16}$$

$\mathbf{W}_p$ and $\mathbf{w}$ are symbolic only in the kinematics variables $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$, which can

either remain symbolic or be substituted as in Algorithm 1. $\mathbf{w}$ is a bias term that holds all of the terms that are not dependent on $\boldsymbol{\alpha}_p$ (including $\hat{\boldsymbol{\alpha}}_r$).

Method two is preferred, as the problem becomes easier to solve when less parameters are being estimated. Regardless, in either technique, $\boldsymbol{\alpha}_p$ will need to be expressed in the same frame as $\boldsymbol{\alpha}_N$, namely the body frame of link N, L_N. We discuss how to perform this transformation in Section 4.4.1.

In summary, Method 1 provides a means of estimating the dynamic parameters of a robot and then proceeding to estimate the inertial parameters of a payload. This method has the advantage of requiring practically no knowledge of the system aside from the kinematic parameters of the robot. It also promises a much easier problem of estimating the object's parameters with less data once a robust model of the robot is obtained. By using the structure of the equations of motions, the optimal estimate can be found with a least-squares quadratic program. While it has practical challenges that become apparent as we test the method on systems in simulation, we learn a lot about least-squares system identification as a whole. The decomposition methods described here will be relevant in the next method.

4.2 Method 2: Direct Payload Identification with Torque Residuals

In order to avoid challenges with identifying a whole manipulator system, we consider a method that does not require an accurate robot model *a priori*. This method is proposed by Khalil et al. (2007) [24]. This method is similar to the first in that it involves symbolic decomposition of the equations of motion into the data matrix and lumped parameters.

We again use the fact that $\boldsymbol{\alpha}'_r = \boldsymbol{\alpha}_r + \boldsymbol{\alpha}_p$, where $\boldsymbol{\alpha}'_r$ are the lumped parameters of the robot when the robot is carrying the object. This means the system can be

decomposed this way:

$$\boldsymbol{\tau}' = \mathbf{W}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})(\boldsymbol{\alpha}_r + \boldsymbol{\alpha}_p) = \mathbf{W}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\alpha}_r + \mathbf{W}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\alpha}_p, \tag{4.17}$$

where $\boldsymbol{\tau}'$ are the observed torques, while the robot is carrying the payload. Noticing that $\mathbf{W}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\alpha}_r$ is equivalent to the model-predicted torque measurements when the robot is not carrying the payload, we state

$$\boldsymbol{\tau}' - \boldsymbol{\tau} = \Delta\boldsymbol{\tau} = \mathbf{W}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\alpha}_p, \tag{4.18}$$

where $\Delta\boldsymbol{\tau}$ are what we refer to as the *residual torques*, being the difference in torques from the robot holding the object and without the object. In lieu of an estimate for $\boldsymbol{\alpha}_r$ to obtain model-predicted torques, we can simply take another observation from the robot executing the same trajectory without holding an object to obtain $\boldsymbol{\tau}$.

Thus, for this method, we collect two sets of data, one without holding the object and one while holding it. The trajectories themselves, $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$, are the same in both data, assuming we have a controller with minimal tracking error, even with an unknown payload. This is easy in simulation, as we have the ground truth model of our system (with the payload) available to us. In a real system, we would need a robust controller that can stabilize in the presence of deviations from the internal model.

This decomposition is calculated symbolically, like in Method 1. We use $\mathbf{W}_p$ to represent the last few columns of $\mathbf{W}$ that correspond to $\boldsymbol{\alpha}_p$, as the first columns and inconsequential. Again, this only is valid if $\boldsymbol{\alpha}_p$ is within the same frame as $\boldsymbol{\alpha}_r$.

Now, our objective is

$$\min_{\boldsymbol{\alpha}_p} \|\mathbf{W}_p\boldsymbol{\alpha}_p - \Delta\boldsymbol{\tau}\|_2^2, \tag{4.19}$$

and least-squares optimization is used to solve for $\boldsymbol{\alpha}_p$:

$$\boldsymbol{\alpha}_p = [\mathbf{W}_p]^+\Delta\boldsymbol{\tau}. \tag{4.20}$$

Once $\boldsymbol{\alpha}_p$ is transformed to be with respect to the body frame of the payload and divided by the estimated mass, m_p, we have obtained an estimate of the payload's inertial parameters.

This method is simplistic and more intuitive; the parameters of the payload should only depend on the discrepancy observed when the robot is carrying the object as to when it is not. However, one drawback of this method is that it requires the collection of two trajectories, one with the object in hand and one without. To make matters worse, the trajectories must be the same within a small error for the method to be successful, meaning that the robot's controller must be well tuned. Nonetheless, it has the major advantage of allowing us to skip the step of robot identification, which is much more challenging. Only estimating the parameters of the payload significantly reduces the complexity of the optimization problem, and requires less data overall to solve, as discovered in [24].

4.2.1 Trajectory Alignment

As mentioned, one of the challenges of the torque residual decomposition method is the need for two trajectories that are nearly identical. This is challenging for a real robot. When a payload is added to the system that is not accounted for in the model used by the controller, there will be errors either in the positions themselves or the time it takes to reach the target positions in the commanded trajectory.

The approach to solving this problem is to tune the controller so that it can quickly manage discrepancies in the robot model and closely track a given position trajectory. We briefly experiment with different gains, both in simulation and on the real system, to see if this is sufficient, but find that we need a more robust method.

Another strategy is to align the two trajectories in post-processing, matching the positions as closely as possible. We can do this using the Dynamic Time Warping (DTW) algorithm, which finds the optimal displacement path between a pair of 2-D

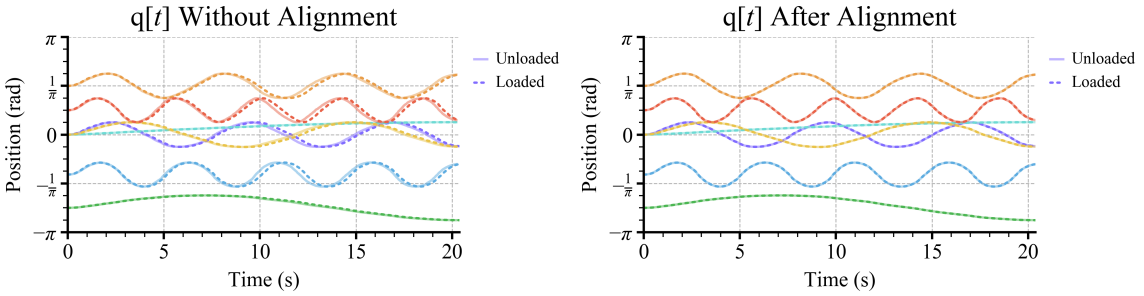

Figure 4-1: The graph on the left shows raw joint position data before trajectory alignment, with and without a 0.9kg payload, where each color represents one joint trajectory. There is lag due to imperfect position control. The right graph shows the trajectories after aligning with dynamic time warping.

signals [39][40]. The objective is

$$\inf_{w} \sum_{(i,j)\in w} (\mathbf{q}[i] - \mathbf{q}'[j])^2, \tag{4.21}$$

where $\mathbf{q}$ is the joint trajectory data collected while the robot is not carrying the object and $\mathbf{q}'$ is the joint trajectory measured when the robot is carrying it.

We use a Python implementation of this algorithm in order to obtain the warping path [41]. The "lagging" trajectory is assumed to be the one of larger length. We construct a new trajectory by adjusting each entry in the lagging trajectory to what it would be at the index corresponded to the leading trajectory. We then use the same mapping to remap all of the data, including the velocities, accelerations, and torque observations. This way, the derivatives do not rely on the adjusted data and each data point used for estimation is conserved.

This approach has notable issues, mainly that we know the velocities and accelerations of the two trajectories will not match exactly. Regardless, this method got us the closest to reasonable payload estimates, as the alignment of the joint positions was quite successful, as shown in Figure 4-1. At the time of writing, this step allows Method 2 to be feasible on hardware.

4.3 Identifiability Analysis

Above, we have seen that the problem of inertial parameter estimation can be formulated as a least-squares optimization. There is lots of theory from linear algebra that we can apply to understanding where this method can succeed and where it might face challenges or impossibilities.

During this process, we empirically observed some parameters to be difficult or impossible to estimate. This motivates an analysis of which parameters in a robotic arm system are *identifiable*. Firstly, we can easily see which parameters are impossible to estimate by looking at the structure of the equations of motion. If a parameter does not appear in the equations, then we cannot identify it using the methods proposed. However, even if a parameter does appear in the equations, it may still be difficult to estimate.

The sensitivity of a least-squares solution will be dependent on the *condition number* of the data matrix $\mathbf{W}$ [16]. The condition number, κ is defined as the ratio of the largest singular value of the matrix ($\sigma_{\max}$) to the smallest singular value ($\sigma_{\min}$):

$$\kappa(\mathbf{W}) = \frac{\sigma_{\max}(\mathbf{W})}{\sigma_{\min}(\mathbf{W})}. \tag{4.22}$$

We run into trouble when $\mathbf{W}$ has singular values that are very close to zero. This may mean that one or more of the parameters are unidentifiable independently of others, and small changes to $\mathbf{W}$ can have large effects on the resulting estimates. This could also occur either as a result of the structure of the equations of motion, meaning that the parameter associated with that singular value is nearly unidentifiable, or from data with insufficient excitation for some parameters. It is not wise to simply remove these parameters from the estimation problem, as other parameters may depend on them. The solution is instead to find a set of base lumped parameters that defines these linear dependencies. This is okay for robot estimation, as the base parameters are the minimum set that are needed to model the robot's behavior, but could cause issues when using the robot's dynamic parameters to identify a payload.

4.4 Data Collection Details

4.4.1 Object Body Frames

The center of mass and inertia of a rigid body are defined with respect to some frame, B, so this frame must be defined. In a simulated system, we are aware of the exact pose of each body in the system at all times. However, in a real system, knowing where the object body is with respect to the rest of the robot is non-trivial.

A simple solution is to hand-pick a frame P to anchor to each object that is defined at the position where the object is grasped by the robot's fingers. Then, we define the pose ${}^{L_N}X^P$ to be the measured grasp pose with respect to the terminal link's frame, L_N. We use monogram notation to express this pose as described in [42]. This means that for the same object grasped at two different poses, the estimated inertial parameters will be different. However, the inertial parameters can be transformed to any coordinate frame via the parallel axis theorem if there is a canonical frame specified for the object. An example of this could be a frame determined by the first step of the Real2Sim pipeline, such as aligning the z-axis with the longest axis on the bounding box of a mesh.

If we want to transform the inertial properties of the object from frame L_N to P, we perform the following steps. The mass is invariant to transformation, and remains constant. The center of mass can be re-expressed in the objects body frame P like any point in 3D space. We define the point of the center of mass in the body frame P in monogram notation as ${}^{P}p^{P_{cm}}$, and re-express via

$$ {}^{P}p^{P_{cm}} = [{}^{P}X^{L_N}]{}^{L_N}p^{P_{cm}} = {}^{L_N}p^{P} + [{}^{P}R^{L_N}]{}^{L_N}p^{P_{cm}}. \tag{4.23} $$

The inertia tensor can be transformed via the parallel axis theorem. First, we must rotate the inertia tensor so that it is expressed in a frame that is parallel to P:

$$ I^P = [{}^{P}R^{L_N}]I^{L_N}[{}^{L_N}R^{P}]. \tag{4.24} $$

Then, it can be translated, now that it is expressed in a parallel frame [43]. When dealing with the inertia tensor, which is an integral over the distribution of mass, it is easier to calculate the translation through the center of mass before translating to another point in space. Fortunately, the `SpatialInertia` module in Drake allows these transformations to take place easily [37].

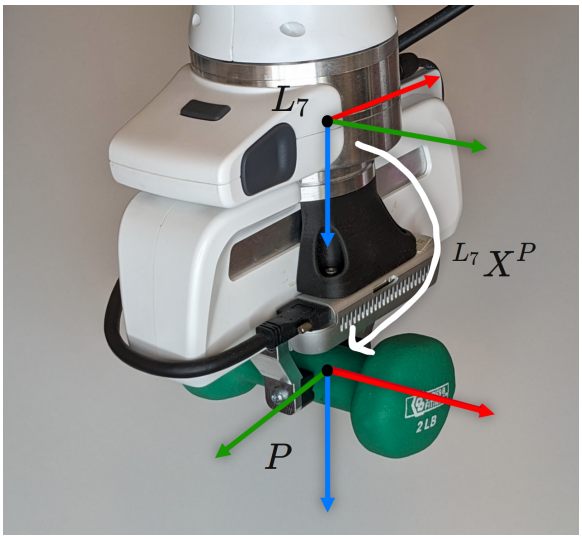

Figure 4-2: Example of a coordinate frame transformation from the terminal link of a Franka Panda (link 7) to the object coordinate frame P.

4.4.2 Trajectory Type

Existing system identification literature recommends sinusoidal excitation of the arm links being identified, because it improves noise reduction and ensures proper dynamic parameter excitation. When collecting data, we command each joint i to move in the pattern of a sinusoid of varying frequency $\boldsymbol{f}(i)$.

$$q_i[t] = 0.3 \sin(\boldsymbol{f} \cdot (i)t). \tag{4.25}$$

Sinusoidal trajectories have advantages, but they are not the types of trajectories that are doing tasks we care about. For the Real2Sim application, we consider pick-and-place trajectories for two reasons. Firstly, a picking robot station has the ability to reset itself, meaning more data collection with less researcher supervision.

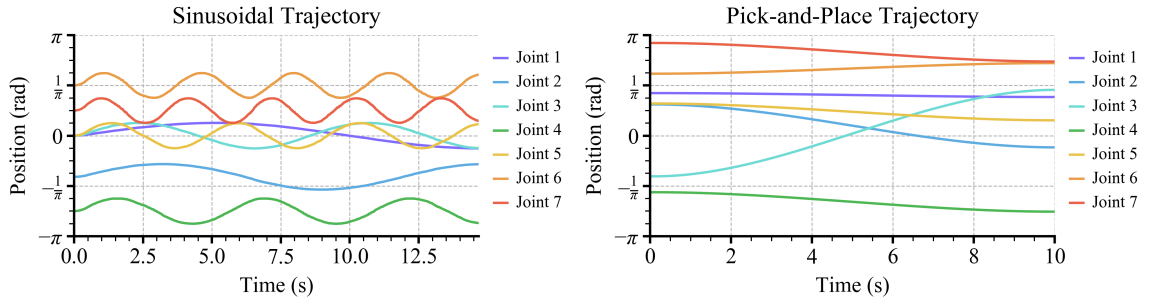

Figure 4-3: Examples of a sinusoidal and pick-and-place trajectory on a 7-link robotic arm

Second, if we assume that pick-and-place trajectories categorize a large portion of the work that manipulators currently perform in industries such as logistics and manufacturing, then understanding how to identify payloads in this regime permits object identification at scale.

The issue with pick-and-place trajectories for system identification is that they are very simple and might not result in proper excitation of the system. A general blueprint of a pick and place trajectory consists of a grasp, a pull-away motion, the movement to the target, and the final placement. We will simplify this blueprint by only considering the movement from a post-grasp point to a pre-place point. This trajectory is commonly represented as a piece-wise polynomial, or even more simply, a linear trajectory in configuration space [42]. We represent the trajectory between two points in task space as a piece-wise cubic polynomial in joint space. This allows us to easily add more breakpoints if necessary to avoid collisions with the environment.

4.4.3 Filtering

Naturally, real data from real robot sensors is accompanied by noise. When solving a problem that is sensitive to the data used for regression, we need to ensure the data is continuous and contains no outliers. In the next section, we perform experiments to have a sense of how much having noise in the data we collect affects the quality of our estimates. Many system identification works use low-pass filtering to clean the data. We use a second-order Butterworth low-pass filter [44] with a cutoff frequency of 10Hz and sampling frequency of 200Hz on the torque sensor output (Figure 4-4).

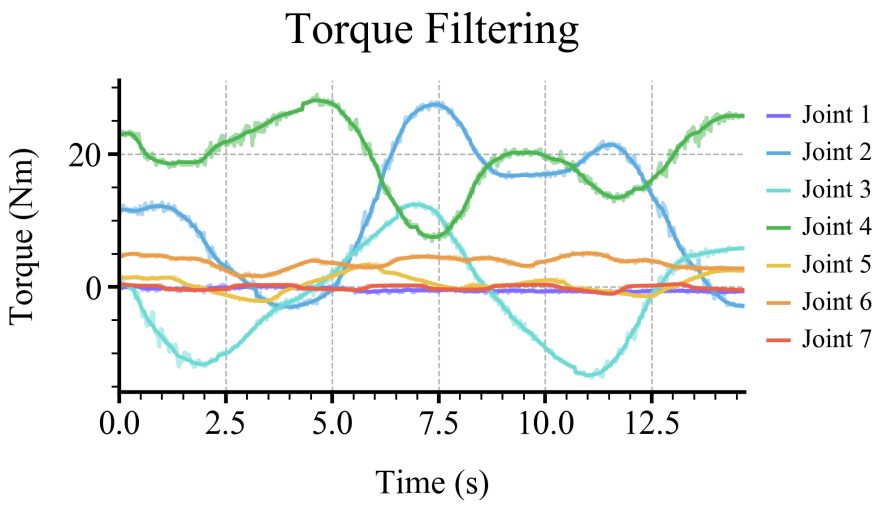

Figure 4-4: Unfiltered (translucent) and filtered (opaque) joint torque measurements. They are filtered using a second-order Butterworth filter with cutoff frequency of 10Hz and sampling frequency of 200Hz.

Additionally, we filter the calculated accelerations with the same filter, since they are calculated by numerically differentiating the measured joint velocities that could contain discontinuities.

Chapter 5

Experiments & Results

5.1 Validating Each Method

In order to test our implementation of the methods described in Chapter 4, it is crucial to first build systems in simulation. The benefit of using these simple systems is that we can fully understand them mathematically; for the most part, their equations of motions within one blackboard and can be derived by hand, as in Chapter 2. Knowing in advance that the general problem may be sensitive to real-world artifacts such as noise, we create a polished environment to leave no room for uncertainty when debugging.

5.1.1 Full-Arm Symbolic Decomposition

1-Link Robot Arm

We define a simulated, planar, robotic arm with one rotary joint at it's base. In our system, the x-axis points to the right, the z-axis points up, and the y-axis points into the screen as constrained by their cross product. The origin of the body frame is located at the base of the link, where it meets the world frame. The joint rotates about the y-axis.

Given that the motion of the system is constricted to the xz-plane, the only inertial

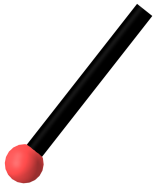

Figure 5-1: A one-link arm, which rotates about a single joint at the base, shown in red.

parameters that appear in the equations of motion are

$$\boldsymbol{\theta} = [m_1,\, c_{x1},\, c_{z1},\, G_{yy1}]. \tag{5.1}$$

Recalling our derivation of the inverse dynamics from Chapter 2 (Eq. 2.2.1), we find the lumped parameters to be

$$\boldsymbol{\alpha} = [m_1 c_{x1},\, m_1 c_{z1},\, m_1 G_{yy1}], \tag{5.2}$$

noting that the mass, m_1 is missing. The ground truth values (units are as defined in Chapter 2) specified in the systems description file are

$$\boldsymbol{\theta} = [0.6,\, 0.0,\, -0.5,\, 0.2]. \tag{5.3}$$

We collect the data from three sinusoidal trajectories, each with increasing frequency, then estimate the robot parameters $\boldsymbol{\alpha}_r$ of the arm, following the procedure in Chapter 4. The results are shown in Table 5.1. For all tables in this section, the error is specified as the absolute error from the ground truth. A reasonable question

$\boldsymbol{\alpha}_r$	**G.T.**	**Estimate**	**Error**
$m_1 c_{x1}$	0.0	$-1.71\text{e}{-5}$	-1.71×10^{-5}
$m_1 c_{z1}$	-0.3	-0.300	$< 1 \times 10^{-3}$
$m_1 G_{yy1}$	0.2	0.201	1×10^{-3}

Table 5.1: Lumped parameter estimates of a planar, 1-link robotic arm in simulation. G.T. is the ground truth parameters, and the error is the absolute error.

to ask is: why are the errors not zero if we have a perfectly simulated environment? We believe the answer is due to how the data points from the trajectory are sampled. We hypothesize that given infinite data collected along the trajectory, the limit of the error goes to zero.

While we obtain accurate estimates of the identifiable lumped parameters, there is still no way of decomposing the individual parameters from the mass, as the mass itself is unidentifiable in this setting. Therefore, there is no way to proceed with the meaningful estimation of a payload's parameters, as there would be no way to identify the mass of the object, only the identifiable lumped parameters.

This experiment proved that our implementation was correct. However, we cannot isolate all of the inertial parameters of the link. This is okay for control applications, but we cannot perform payload identification. Fortunately, in systems with higher degrees of freedom, the terminal link becomes fully identifiable, thus, we move on towards the 2-link robot arm.

2-Link Robot Arm

The natural extension of the 1-link system is a 2-link robot arm. This arm has two rotary joints that move about the y-axis, while the x-axis points right and the z-axis points up. Each link has a mass and a center of mass about it's origin, which coincides with the geometric center of the link. Its moments of inertia are assumed to be that of a rod rotating about a pin pointing into the screen.

The inertial parameters of each link i of this system are

$$\boldsymbol{\alpha}_i = [m_i,\, c_{xi},\, c_{zi},\, m_i G_{yyi}]. \tag{5.4}$$

The ground truth values for the first link are the same as the 1-link arm. The ground truth inertial parameters of link 2 are

$$\boldsymbol{\alpha}_2 = [0.8,\, 0,\, -0.75,\, 0.6]. \tag{5.5}$$

As shown in Chapter 2, this system's inverse dynamics equations reveal more inertial parameters to be estimated, notably the mass of the second link. We found the lumped parameters of the arm to be

$$\boldsymbol{\alpha}_r = \begin{bmatrix} m_2, & m_1 G_{yy1} + m_2 G_{yy2}, & m_1 c_{x1}, & m_1 c_{z1}, & m_2 c_{x2}, & m_2 c_{z2}, & m_2 G_{yy2} \end{bmatrix} \quad (5.6)$$

Notice that there is a redundancy where $m_2 G_{yy2}$ appears in two of the lumped parameters. This is an artifact of the decomposition algorithm grouping terms with the same coefficients. The estimates of these lumped parameters are highly inaccurate, as shown in Table 5.2.

$\boldsymbol{\alpha}_r$	**G.T.**	**Estimate**	**Error**
m_2	0.8	1.10	0.30
$m_1 G_{yy1} + m_2 G_{yy2}$	0.8	0.509	0.291
$m_1 c_{x1}$	0.0	-6.82×10^{-5}	6.82×10^{-5}
$m_1 c_{z1}$	-0.3	3.71×10^{-16}	0.300
$m_2 c_{x2}$	0.0	-1.84×10^{-4}	1.84×10^{-4}
$m_2 c_{z2}$	-0.6	-0.600	0.000
$m_2 G_{yy2}$	0.6	0.602	0.002

Table 5.2: Estimates of a 2-link planar robotic arm in simulation

The first thing we note from this result is the poor estimate of m_2, yet many of the parameters which contain m_2 are correct. We then note that the condition number $\kappa(\mathbf{W})$ is very high, on the order of 1×10^{15}. While almost all of the singular values of $\mathbf{W}$ are on the order of 1×10^2, one of the singular values was on the order of 1×10^{-13}. This suggests that at least one of the lumped parameters identified by the decomposition algorithm is actually not well-suited for estimation independently and are linearly dependent with another parameter. It would require that we either find a new set of base parameters via methods described in Section 3.3.2.

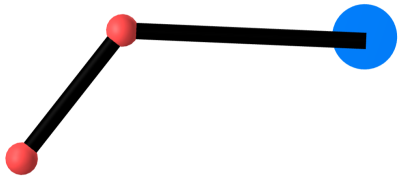

Figure 5-2: A simulated 2-link planar manipulator with an object (in blue) rigidly attached to the terminal link

5.1.2 Torque Residual Decomposition

2-Link Robot Arm

In this method, we would expect that the identifiability issues in the links of the arm are abstracted away, as this information is encoded in the torques observed when the robot is not carrying a payload. If we perform the decomposition using Drake, we see that the remaining payload parameters are both fully identifiable and identifiable independently of each other.

To validate this method in simulation, we gather data from 10 simulated sinusoidal trajectories, each with increasing frequency, for 17421 data points in total. The results were sufficient, and we decide that this method is more reliable for our purposes, at least in simulation.

$\boldsymbol{\alpha}_p$	G.T.	Estimate	Error
m_p	1.0	1.000	0.000
$m_p c_{xp}$	0.025	0.0250	0.000
$m_p c_{zp}$	0.01	0.00907	9.30×10^{-4}
$m_p G_{yyp}$	0.0457	0.0445	2.00×10^{-4}

Table 5.3: Results of payload identification on a 2-link arm using the direct payload identification method.

Simulated Panda Arm

Now we attempt to estimate an object grasped by a simulated Franka Panda arm, to see if the method holds in three dimensions. Similarly to the previous experiment,

we execute three sinusoidal trajectories for data collection. Since the robot's controller has a perfect model of the object in simulation, there is no need for trajectory alignment, as the controller can track the executed position exactly while holding the object. We also do not perform any filtering, as there is no noise in the simulation. We run these experiments on a simulated object that has a mass of 0.37 [kg], center of mass (0.0, 0.0, 0.0) [m], moments of inertia (8.51e-3, 5.96e-4, 7.65e-4) [kg·m^2], and products of inertia (1.11e-4, 1.70e-4,-1.28e-4) [kg·m^2].

The results are what we expect, with very low absolute error, validating this method on a 7-degree-of-freedom system.

$\boldsymbol{\alpha}$	**G.T.**	**Estimate**	**Error**
m_p	0.370	0.370	$< 1 \times 10^{-3}$
$m_p c_{xp}$	0.0	-4.37×10^{-6}	-4.37×10^{-6}
$m_p c_{yp}$	0.0	8.15×10^{-6}	8.15×10^{-6}
$m_p c_{zp}$	0.0	5.65×10^{-6}	5.65×10^{-6}
$m_p G_{xxp}$	3.17×10^{-4}	3.43×10^{-4}	2.6×10^{-5}
$m_p G_{yyp}$	4.20×10^{-4}	4.33×10^{-4}	1.3×10^{-5}
$m_p G_{zzp}$	5.33×10^{-4}	5.29×10^{-4}	4.0×10^{-6}
$m_p G_{xyp}$	0.0	1.62×10^{-5}	1.62×10^{-5}
$m_p G_{xzp}$	0.0	-6.58×10^{-6}	-6.58×10^{-6}
$m_p G_{yzp}$	0.0	-6.76×10^{-6}	-6.76×10^{-6}

Table 5.4: Results of payload identification on a simulated Panda arm. The data is collected from three sinusoidal trajectories, and the resulting absolute error is very low.

5.2 Sensitivity Analysis

After showing that we can obtain a reasonable estimate of an object's inertial parameters in three dimensions, we want to understand how sensitive these estimates are to noise in the robot's sensors before moving on to hardware. Beginning with a baseline of a near-perfect estimate in simulation, we can inject random noise in the inputs and measure the discrepancy in performance. We run these experiments using the same object as the previous section.

5.2.1 Sensitivity to Noise in State Measurements

In this experiment, we inject Gaussian noise with a standard deviation $\sigma = 0.01$ into both of the $\mathbf{q}$ and $\dot{\mathbf{q}}$ logs from the simulation. We expect to see noise in these measurements, as the joint position encoders are imperfect and susceptible to noise. We run the same simulation 18 times, sampling an array of Gaussian noise to add to the trajectory in each trial. This value of sigma was chosen because it modeled the observed discrepancies between the loaded and unloaded trajectories in our real data most closely.

We find that small errors in the state observations lead to large errors in the acceleration calculations, likely due to using finite differences to carry out the calculation. This suggests that we should try another method for calculating the accelerations in the presence of noise.

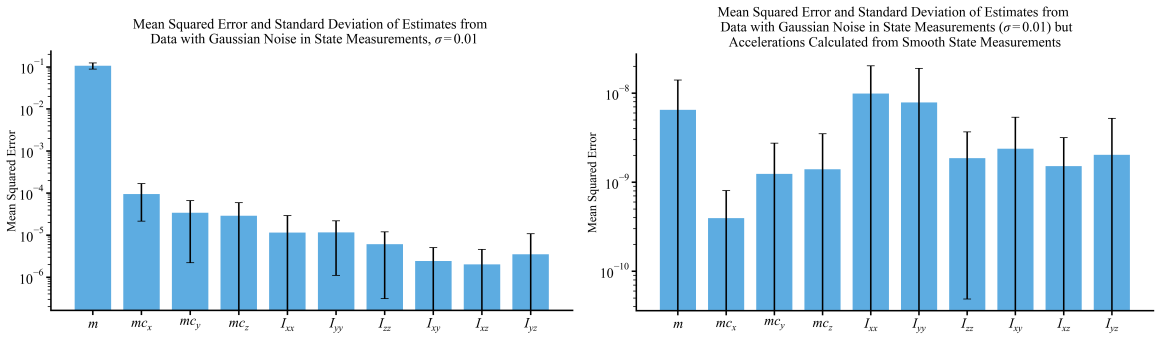

Figure 5-3: The mean squared error and standard deviation for each lumped inertial parameter when random Gaussian noise with $\sigma = 0.01$ is added to the joint position and velocity observations in simulation. The left plot shows when the accelerations are calculated from the noisy measurements, and the right shows when they are calculated from smooth measurements.

What we see is that the estimator cannot estimate even the mass of an object when there is that much noise in the joint torque sensors. This could be due to large discontinuities in the accelerations that are large enough to cause biases in the estimation. However, it is still unclear whether this is due to the two trajectories used to calculate the torque residuals are becoming unaligned due to the noise, or if it is because of the volatile nature of the accelerations. To test this, we run the same experiment but instead calculate the accelerations using the joint state values before

the noise is added, so that they are smooth. The estimates have very little error in comparison, as seen in Figure 5-3.

This experiment shows that while the estimation procedure is relatively robust to noise in the state observations and even discrepancies between the loaded and unloaded trajectories, there is trouble when the noise is propagated into the calculated accelerations. Thus, having smooth accelerations is crucial for obtaining good estimates, and important to take care in how we are differentiating the signals, especially when noise is involved.

5.2.2 Sensitivity to Noise in Torque Measurements

We expect that adding noise to the torque observations will cause significant errors in estimation, and test this theory. In this experiment, we run 10 trials, each with the same trajectory and no torque observation filtering. First, we inject Gaussian noise with $\sigma = 0.01$ into the joint torque observation. Looking at how mass is estimated over time, we see that while there is initially more volatility, the result does converge (Figure 5-4). Across multiple trials, we see that the error is very small and robust to noise.

Next, we increase the standard deviation of the noise by an order of magnitude, to $\sigma = 0.1$. Now we begin to see that the estimation procedure is not as robust, and while the estimate converges on some trials, it often does not, leading to higher error. An example of a trial where the mass estimate does not converge is shown in Figure 5-4.

These experiments have shown us that filtering the observed data is a necessary part of the estimation process. When we return to the problem on hardware, we keep these design choices in mind.

5.3 Robot Experiments

We now experiment on the Panda robot, with real objects, using the insights from prior experiments in simulation.

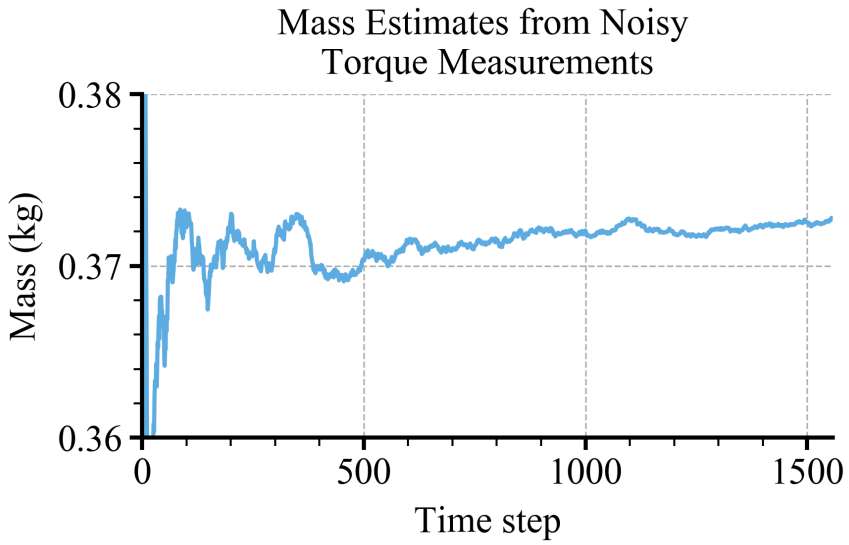

Figure 5-4: Plotted on the y-axis are the mass estimates at each step of the data matrix construction. The data was collected in simulation with Gaussian noise with $\sigma = 0.1$ added to the torque observations. While the estimates begin by jumping around, they eventually stabilize but do not converge to the correct value of 0.37kg.

5.3.1 Identifying the Panda

Noting the issues of conditioning that we have uncovered, we still attempt to identify a real Panda robot. The benefits of having a fully-calibrated system mean that we need less information to identify object with our pipeline going forward. Given the issues with consistent equation decomposition, we experiment to see if beginning with an initial condition. Fortunately, there are two physically feasible initial conditions

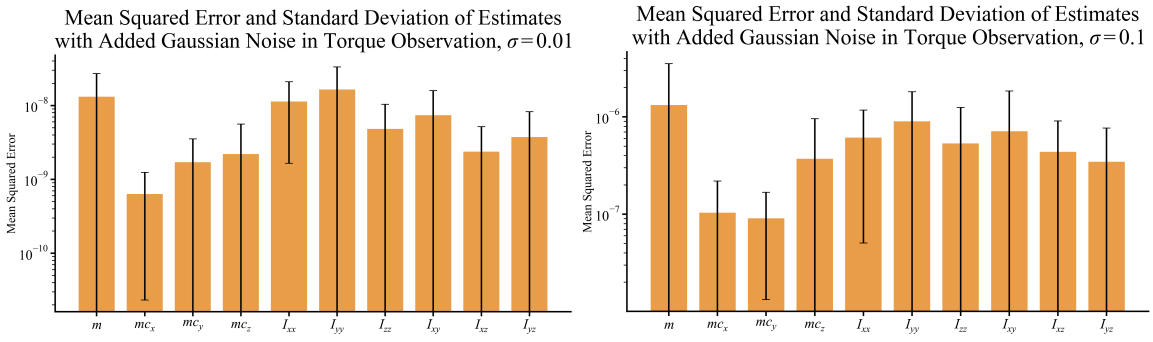

Figure 5-5: The mean squared error and standard deviation for each lumped inertial parameter when random Gaussian noise with $\sigma = 0.01$ and $\sigma = 0.1$ is added to the joint torque observations in simulation.

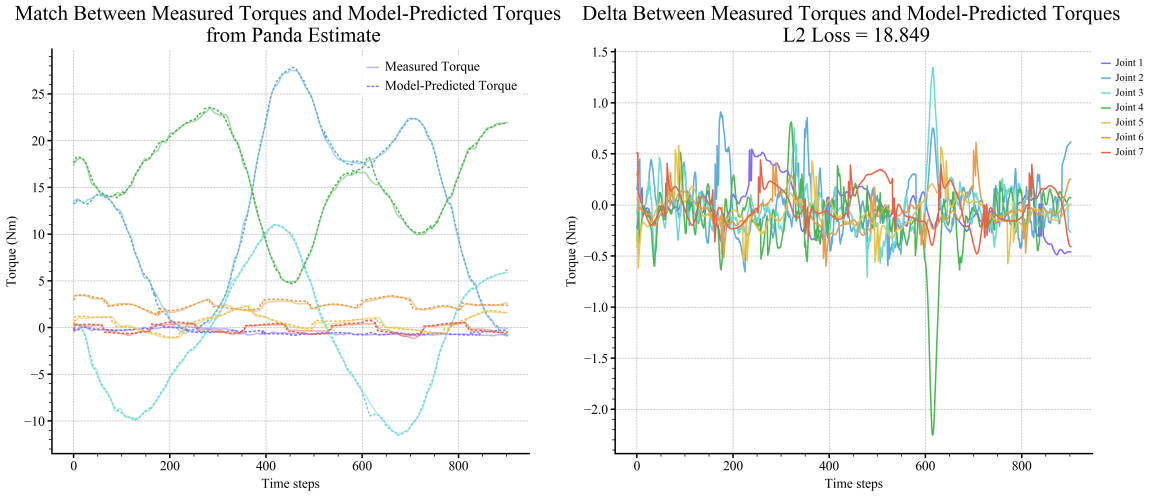

Figure 5-6: Comparison of the measured torques from the Panda vs. the model predicted torques of our dynamic model. The outlier in the right plot is due to an outlier in the calculated accelerations.

provided from [1] and [2] (Fig. 3-1). We choose to use [2] as an initial condition, as their results appear to be closer to physical reality.

We collect 10 trajectories for training, using the sinusoidal trajectory scheme and filter both the calculated accelerations and the observed joint torques. We include joint friction coefficients to the lumped parameters and use auto-differentiation in order to recover the data matrix.

Without a ground truth set of inertial parameters for the Panda, the error metric we must rely on is the tracking error between the filtered measured torques from the robot and the new model's torques when estimating the same trajectory. What we notice is that the estimate we obtain results in low error between the measured torques and the model-predicted torques (Figure 5-6. However, the actual results of the inertial parameters are not physically feasible, which we demonstrate by highlighting some of the estimated parameters in bold in Table 5.5.

Specifically, we notice that the mass estimate for the 7th link, which would ultimately be used to identify the payload, is negative. This is unsuitable to proceed with payload estimation and demonstrates how physical feasibility is not guaranteed, even when using a physically feasible initial condition. In order to make this method work, we would need a procedure for ensuring physically feasible estimates via semi-definite

$\boldsymbol{\alpha}_r$	**Link 1**	**Link 2**	**Link 3**	**Link 4**	**Link 5**	**Link 6**	**Link 7**
m	1.000	2.647	1.965	0.833	1.447	1.548	**-0.170**
c_x	0.000	-0.011	0.233	-0.398	-0.002	0.149	-0.095
c_y	0.000	-1.123	0.005	**1.988**	0.001	-0.053	-0.105
c_z	0.000	0.067	-0.654	-0.001	-0.451	-0.058	-0.574
I_{xx}	1.006	0.469	0.629	0.375	0.324	0.041	-0.047
I_{xy}	0.000	-0.006	-0.000	0.121	-0.001	0.025	-0.002
I_{xz}	0.000	0.001	0.154	0.001	-0.002	0.008	0.024
I_{yy}	1.006	0.046	0.700	0.053	0.341	0.045	-0.045
I_{yz}	0.000	0.082	-0.011	-0.002	0.016	-0.007	-0.010
I_{zz}	0.008	0.463	0.071	0.430	0.015	0.068	-0.054
μ_v	0.035	0.242	0.140	-0.099	-0.007	-0.087	-0.100
μ_c	-0.310	-0.431	-0.257	-0.406	-0.454	-0.401	-0.384
μ_o	0.400	-0.645	-0.360	0.752	-0.049	-0.227	-0.010

Table 5.5: Estimated dynamic parameters of the Franka Emika Panda using an initial condition from [2]. While these values produce good dynamic results, they are not physically feasible or realistic, as highlighted by a few parameters in bold.

constraints.

5.3.2 Object Estimation

While we were initially unsuccessful in obtaining a model of the Panda, we proceed to estimate a payload using the torque residual method.

Ground Truth Dataset

A significant challenge in validating our results is finding a dataset of objects that have known, ground truth inertial parameters. Finding the mass of any object is trivial given common weight measurement tools, but finding the center of mass and inertia requires either very tedious experimentation or special measurement tools.

For objects where we have a mesh or CAD model available, we can measure the mass, then integrate over a mesh of the object to find the center of mass and inertia, assuming a constant density. This method is not perfect, but it allowed us to compare our results to some baseline. If the object is simple enough, we can hand-calculate ground truth values using primitives.

Figure 5-7: The 2lb dumbbell that we attempt to estimate with payload identification

Even generating a wide variety of objects to test our method in simulation has challenges. We only care about an object's inertial parameters for testing, the visual and geometric properties of the simulated object are insignificant. However, creating a random inertial profile is very difficult, as the parameters need to satisfy the physical feasibility constraints.

Estimating a Dumbbell

To test our method on a real world object, we choose to use a Series 8 Fitness™ 2lb dumbbell (Figure 5-7). The material of the dumbbell is unknown, but we assume that it has uniform density. We began with this object because we can estimate ground truth inertial properties by hand by approximating the volume as three cylinders.

In this real-world experiment, we first collect ten sinusoidal trajectories of various frequencies while the robot is not carrying the object. Then, we collect data from the same trajectories again, this time with the object manually placed within the robot's grasp. We note that by doing the placement manually, we cannot guarantee that the object's origin frame, which is defined between the robot fingers, will be the same each time. We expect this to result in slightly inconsistent center of mass and inertia estimations. The data is aligned using dynamic time warping and the payload parameters are computed using the torque residual method.

In this primary real-world experiment, we also analyzed how the estimates changed with respect to noise, by filtering the observations in one estimation and leaving them unfiltered in another.

		No Filtering		Filtering	
$\boldsymbol{\alpha_p}$	**Ground Truth**	**Estimate**	**Error**	**Estimate**	**Error**
m_p	0.902	0.9079	5.9×10^{-3}	0.9086	6.6×10^{-3}
$m_p c_{xp}$	0.0	1.359×10^{-3}	1.359×10^{-3}	-5.596×10^{-4}	5.596×10^{-4}
$m_p c_{yp}$	0.0	-5.359×10^{-3}	5.359×10^{-3}	-5.166×10^{-3}	5.166×10^{-3}
$m_p c_{zp}$	0.0	-1.978×10^{-3}	1.978×10^{-3}	3.031×10^{-4}	3.031×10^{-4}
$m_p G_{xxp}$	2.440×10^{-4}	-4.882×10^{-3}	$5.126 \times 10{-}3$	5.759×10^{-3}	5.515×10^{-3}
$m_p G_{yyp}$	2.026×10^{-3}	-0.05246	0.0545	0.01987	0.0178
$m_p G_{zzp}$	2.026×10^{-3}	-9.934×10^{-3}	7.908×10^{-3}	6.153×10^{-3}	4.127×10^{-3}
$m_p G_{xyp}$	0.0	-3.802×10^{-3}	3.802×10^{-3}	-1.130×10^{-3}	1.130×10^{-3}
$m_p G_{xzp}$	0.0	-2.983×10^{-3}	2.983×10^{-3}	2.112×10^{-3}	2.112×10^{-3}
$m_p G_{yzp}$	0.0	-3.322×10^{-3}	3.322×10^{-3}	4.765×10^{-3}	4.765×10^{-3}

Table 5.6: Results of estimating a dumbbell on hardware using the direct payload identification method, comparing the results using unfiltered data vs. filtered data.

We first notice that for each parameter except mass, we see better results with respect to the ground truth when the data is filtered. Regardless, we can still expect to see absolute error within 1×10^{-2} for each parameter, which is not ideal. This is up to 1cm or 10g. If we analyzed the relative error instead of absolute, we would see that $m_p G_{yyp}$, for instance, is very high, nearly 800%. However, there are quite a few explanations for why we may see error this high on the real system. Firstly, the ground truth values are not an accurate representation of the real properties of the object; they are only an approximation. It is possible that our estimate is a better measure of the object's inertial properties than our hand-calculated values; perhaps it is true that the dumbbell has higher density at its ends. Additionally, as mentioned previously, there is significant error introduced when the object is placed in the robot's grasp by a human. Since the coordinate frame of the object is defined with respect to where it is grasped, having an inexact grasp point will affect the estimates of the center of mass and inertia. Data is also not sampled as frequently on the real system than in simulation, meaning that with noise present, the estimate may not be as accurate. This could be mediated by sampling at a higher frequency. Lastly, a subtle difference in the inertia could be caused by the fact that the fingers of the gripper are open when the unloaded trajectories are collected, but closed enough to grasp the object when collecting the loaded trajectories.

While the results are respectable in terms of absolute error, unfortunately the estimated inertia tensor does not satisfy all of the constraints of a physically feasible answer. This suggests that vanilla least-squares optimization is not sufficient; there are additional semi-definite constraints that are necessary in order to insure that the inertia tensor is both positive-definite and satisfies the triangle equality. However, is a physically feasible inertia tensor necessary for a simulated object? This question is still unanswered, but if we make this a requirement, the results would need to be projected to the nearest feasible solution, even if the estimated parameters perform well dynamically.

Overall, we find that we can obtain reasonable estimates of an objects inertial parameters with a real robotic system, if we disregard physical feasibility. The estimates are not perfect, especially in the inertia, but we do not have a solid ground truth comparison and hypothesize that they are still sufficient for a Real2Sim asset. More engineering will be needed in order to obtain more accurate ground truth comparisons and to remove human error in the estimates. A future step will be to see how this estimated object behaves in a simulation.

5.4 Best Trajectories

Historically, sinusoidal trajectories have been the most widely used as inputs for identifying systems. We wanted to see if the payload identification could also work while performing a pick-and-place task, so that it can fit more seamlessly into a Real2Sim pipeline.

Now that we can obtain reasonable estimates of an object's inertial parameters with a baseline method, we seek to tailor the method more towards the Real2Sim problem. While having the robot perform sinusoidal motions for estimation in an object estimation pipeline is not unrealistic, it would be much more efficient if the estimation occurred while the robot was already moving to place the object on a table or back in a bin. In these experiments we seek to answer whether or not we can still reliably estimate with less exciting data. We hypothesize that for payload

estimation, a pick-and-place trajectory will be sufficient, given that there are not not many parameters to estimate and plenty of data still available.

In this experiment, we compare the mean and standard deviation of the error from the ground truth lumped parameters across 10 trials for both sinusoidal trajectories of varying frequencies and pick-and-place trajectories with varying waypoints. The sinusoidal trajectories each have an amplitude of 0.3 radians, and the frequencies of each joint are sampled uniformly from the interval [1, 5] rad/s. The pick-and-place trajectories start and end positions are sampled uniformly from a $0.3 \times 0.6 \times 0.3\text{m}^3$ box of space both in front of and behind the robot.

During the estimation procedure with the pick-and-place trajectories, we notice that the condition number of the data matrix is much higher, i.e. on the order of 1×10^3. This is to be expected if not all of the axes of rotation are being properly excited. In comparison, the average condition number for a data matrix constructed from a sinusoidal trajectory was about 8.

We notice the effect of this in our results, shown in Figure 5-8. While the error is relatively low for both, the difference in the distribution of error across the parameters is striking. Sinusoidal trajectories result in an error distribution closer to uniform, whereas the pick-and-place data results in high error along certain axes of rotation, In this case it appears there is not enough vertical excitation to estimate the z-component of the center of mass or the moments of inertia about the x or y-axes. This would occur if the robot's hand is moving relatively planar to the ground. This could be avoided by incorporating a motion such as a lift after the object is grasped, which is generally incorporated into a pick-and-place trajectory regardless. In conclusion, when constructing pick-and-place trajectories for object identification, we should keep in mind whether the object is being excited at all axes of rotation, and consider design decisions to ensure this is true.

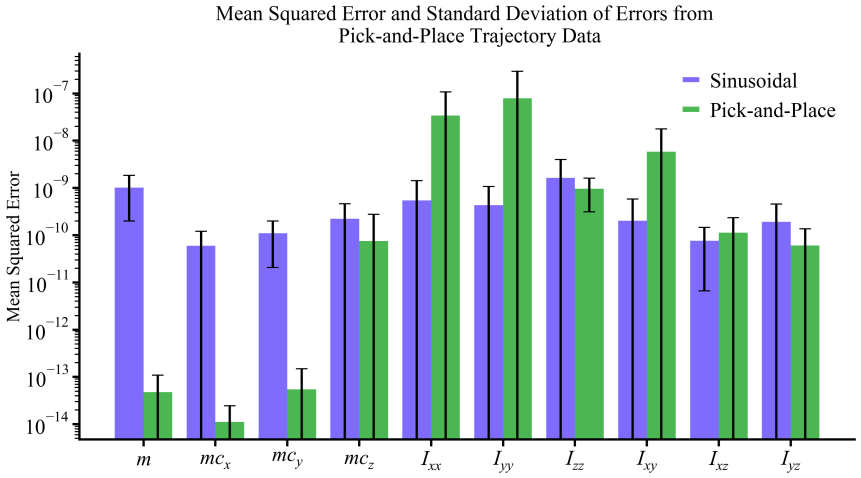

Figure 5-8: The difference in lumped parameter errors from estimation on sinusoidal trajectories vs. pick-and-place trajectories. The pick-and-place trajectories result in much higher error along certain axes of rotation, whereas sinusoidal gives more constant error across the parameters.

Chapter 6

Conclusion

6.1 Summary of Results

We presented two classes of methods that can be used to identify the inertial parameters of a payload: estimating the dynamic parameters of the robot arm first and estimating the payload parameters directly. After attempting to identify the robot's dynamic parameters, we find that this requires much more engineering effort than it may be worth, as we need to find a set of base lumped parameters and address issues with physical feasibility. There is value in having a robust model of the arm, but given that it is bespoke to each robot and must be accurate enough to not propagate error into the object estimate, we decide to focus our efforts on the direct payload identification method.

We find that direct payload identification with torque residuals works sufficiently well, even with real robot data, and can be integrated into a Real2Sim pipeline. For example, we learn that pick-and-place trajectories can be used for data collection, as long as there are motions which excite all three axes of rotation. There likely is no additional trajectory optimization that must be done in order to obtain a good estimate; the addition of extra motions via human design is likely sufficient. Additionally, we find that filtering the input data, specifically the acceleration and torque observations, is shown to be consequential to the estimate quality and must be incorporated into the overall method.

6.2 Challenges

The challenges faced in this project were mainly related to attempts to estimate the parameters of a robot. While this problem has been solved and understood thoroughly in theory, there are many practical issues of implementation on real hardware. Firstly, we would need to find a set of base lumped parameters to estimate rather than our initially proposed lumped parameters, due to issues of identifiability and parameters of some links being linearly dependent on the parameters of other links. Additionally, in order to perform payload identification, the model of the robot that we estimate must be physically feasible and physically realistic. While physical feasibility can be ensured via semi-definite constraints, we cannot guarantee physical reality. The issue of physical feasibility also reappears in the problem of direct payload estimation. We find that it is necessary to impose constraints on the payload parameters in order to ensure that they are possible in nature.

Hardware challenges also manifested themselves. One significant challenge was finding objects that could be easily grasped by the robot's fingers. While we would like to test objects that have non-zero centers of mass, lopsided objects such as these were difficult to grasp and would often slip. In a sterilized data collection scenario, these issues can be fixed by further engineering. However, in a Real2Sim pipeline, we may have significant trouble with ensuring objects are ridigly grasped.

6.3 Future Directions

The future direction of this work mainly involves further integration with the Real2Sim pipeline. The perception problem is the main barrier; we would like to be able to calculate the object's inertial properties with respect to a chosen antipodal grasp point on a point cloud that can be directly mapped to the object mesh. This would provide much better precision of the origin's location with respect to the center of mass, as there is currently human-introduced error.

Adaptive control is another natural direction. We focus on offline estimation in

this work, but online system has also been explored in previous works. With a fast enough estimation procedure, the robot could learn about the object in real-time, or even choose exploratory motions to excite all of the parameters.

Another crucial future step is to examine how the objects perform in simulation with estimated inertial parameters, but it is hard to define good metrics. We eventually expect to out-perform existing Real2Sim methods, especially for objects that are not of constant density. When the full pipeline becomes solidified, we plan to create a dataset that has accurate dynamic properties many types of common objects. We hope that this pipeline can be used to create simulation-ready assets that are more dynamically accurate than what is currently available.

Bibliography

[1] Claudio Gaz, Marco Cognetti, Alexander Oliva, Paolo Robuffo Giordano, and Alessandro De Luca. Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization. *IEEE Robotics and Automation Letters*, 4(4):4147–4154, 2019.

[2] Fernando Díaz Ledezma and Sami Haddadin. Ril: Riemannian incremental learning of the inertial properties of the robot body schema. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9354–9360, 2021.

[3] H. Mayeda, K. Osuka, and A. Kangawa. A new identification method for serial manipulator arms. *IFAC Proceedings Volumes*, 17(2):2429–2434, 1984. 9th IFAC World Congress: A Bridge Between Control Science and Technology, Budapest, Hungary, 2-6 July 1984.

[4] Russ Tedrake. *Underactuated Robotics*. 2023.

[5] Richard M. Murrray, Zexiang Li, and S. Shankar Sastry. *A Mathematical INtroduction to Robotic Manipulation*, chapter 4. CRC Press, Taylor and Francis Group, 1994.

[6] Gregory L. Long. *Fundamentals of Robot Mechanics*. Quintus-Hyperion Press, 2015.

[7] Chirstopher G. Atkeson, Chae H. An, and John M. Hollerbach. Rigid body load identification for manipulators. In *1985 24th IEEE Conference on Decision and Control*, pages 996–1002, 1985.

[8] 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, 1985.

[9] M. Gautier and W. Khalil. On the identification of the inertial parameters of robots. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 2264–2269 vol.3, 1988.

[10] Yiwen Chen, Xue Li, Sheng Guo, Xian Yao Ng, and Marcelo Ang. Real2sim or sim2real: Robotics visual insertion using deep reinforcement learning and real2sim policy adaptation, 2022.

[11] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[12] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery.

[13] Simon Le Cleac'h, Hong-Xing Yu, Michelle Guo, Taylor A. Howell, Ruohan Gao, Jiajun Wu, Zachary Manchester, and Mac Schwager. Differentiable physics simulation of dynamics-augmented neural objects, 2023.

[14] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. PAC-neRF: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. In *The Eleventh International Conference on Learning Representations*, 2023.

[15] Eric Heiden, David Millard, and Gaurav S. Sukhatme. Real2sim transfer using differentiable physics. *R:SS Workshop on Closing the Reality Gap in Sim2real Transfer for Robotic Manipulation*, 2019.

[16] Wisama Khalil. *Modeling, Identification, and Control of Robots*, chapter 12. Elsevier Science, 2004.

[17] Nikos Mavrakis and Rustam Stolkin. Estimation and exploitation of objects' inertial parameters in robotic grasping and manipulation: A survey. *Robotics and Autonomous Systems*, 124:103374, 2020.

[18] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[19] Trevor Standley, Ozan Sener, Dawn Chen, and Silvio Savarese. image2mass: Estimating the mass of an object from its image. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 324–333. PMLR, 13–15 Nov 2017.

[20] Yong Yu, T. Arima, and S. Tsujio. Estimation of object inertia parameters on robot pushing operation. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1657–1662, 2005.

[21] Nikos Mavrakis, Amir M. Ghalamzan E., and Rustam Stolkin. Estimating an object's inertial parameters by robotic pushing: A data-driven approach. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9537–9544, 2020.

[22] Rechard P. Paul. *Robot Manipulators: Mathematics, Programming, and Control*, chapter 8. MIT Press, 1981.

[23] P. Coiffet. Robot technology: Interaction with the environment. volume 2.

[24] Wisama Khalil, Maxime Gautier, and Philippe Lemoine. Identification of the payload inertial parameters of industrial manipulators. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4943–4948, 2007.

[25] Claudio Gaz and Alessandro De Luca. Payload estimation based on identified coefficients of robot dynamics — with an application to collision detection. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3033–3040, 2017.

[26] H. Olsen and G. Bekey. Identification of parameters in models of robots with rotary joints. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 1045–1049, 1985.

[27] A. Mukerjee and D. Ballard. Self-calibration in robot manipulators. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 1050–1057, 1985.

[28] B. Armstrong, O. Khatib, and J. Burdick. The explicit dynamic model and inertial parameters of the puma 560 arm. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 510–518, 1986.

[29] franka_ros. https://frankaemika.github.io/docs/franka_ros.html. Accessed: 2023-04-05.

[30] Yvonne R. Stürz, Lukas M. Affolter, and Roy S. Smith. Parameter identification of the kuka lbr iiwa robot including constraints on physical feasibility. *IFAC-PapersOnLine*, 50(1):6863–6868, 2017. 20th IFAC World Congress.

[31] M. Gautier and W. Khalil. A direct determination of minimum inertial parameters of robots. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 1682–1687 vol.3, 1988.

[32] M. Gautier. Numerical calculation of the base inertial parameters of robots. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 1020–1025 vol.2, 1990.

[33] Wolfgang Rackl, Roberto Lampariello, and Gerd Hirzinger. Robot excitation trajectories for dynamic parameter estimation using optimized b-splines. pages 2042–2047, 05 2012.

[34] Patrick M. Wensing, Sangbae Kim, and Jean-Jacques E. Slotine. Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution. *IEEE Robotics and Automation Letters*, 3(1):60–67, 2018.

[35] Giovanni Sutanto, Austin S. Wang, Yixin Lin, Mustafa Mukadam, Gaurav S. Sukhatme, Akshara Rai, and Franziska Meier. Encoding physical constraints in differentiable newton-euler algorithm, 2020.

[36] Andy Lambert. aalamber/real2sim-payload-id, 2023.

[37] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.

[38] Gilbert Strang. *Linear Algebra and its Applications: Third Edition*. Thomson Learning, 1988.

[39] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[40] Kongming Wang and Theo Gasser. Alignment of curves by dynamic time warping. *The Annals of Statistics*, 25(3):1251 – 1276, 1997.

[41] wannesm, khendrickx, Aras Yurtman, Pieter Robberechts, Dany Vohl, Eric Ma, Gust Verbruggen, Marco Rossi, Mazhar Shaikh, Muhammad Yasirroni, Todd, Wojciech Zieliński, Toon Van Craenendonck, and Sai Wu. wannesm/dtaidistance: v2.3.5, January 2022.

[42] Russ Tedrake. *Robotic Manipulation*. 2022.

[43] Jaime Peraire and Sheila Widnall. Lecture l26 - 3d rigid body dynamics: The inertia tensor, September 2008.

[44] S. Butterworth. On the Theory of Filter Amplifiers. *Experimental Wireless & the Wireless Engineer*, 7:536–541, October 1930.