NORTHWESTERN UNIVERSITY

Information-based Trajectory Optimization for Active Estimation in Mechanical Systems

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

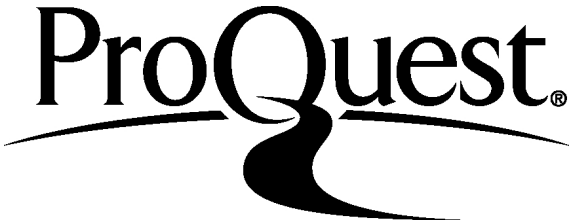
Field of Mechanical Engineering

By

Andrew D. Wilson

EVANSTON, ILLINOIS

December 2015



ABSTRACT

Information-based Trajectory Optimization for Active Estimation in Mechanical Systems

Andrew D. Wilson

In order to control complex and nonlinear systems, models are created to predict physical behavior. These mathematical models contain a variety of parameters such as masses, geometries, and friction coefficients that are specific to each system. In many cases, these parameters may not be certain and must be estimated. However, the control inputs or trajectory used to excite the system for purposes of estimation will affect the ability and precision with which the parameters can be estimated. The primary focus of this work is determining if local optimization approaches can be used to automatically synthesize trajectories for better estimation of parameters in dynamical systems.

The motivation for the techniques used in this thesis stems from a case-study which is presented on the simulation of a biomechanical model of the upper limb. This study motivates the need for estimation of model parameters due to high sensitivities of important parameters such as the muscle wrapping radius. Additionally, the use of structured numerical techniques in the simulation and optimization algorithms is studied and compared to continuous based methods.

Several algorithms are derived and validated using experimental trials. Continuous and discrete methods of information optimization are derived, and results from a cart and double pendulum system are provided. Additionally, optimization of the condition number of the estimator is examined to improve the rate of convergence of the estimation of three parameters in a cart pendulum system. Finally, a real-time implementation of information-based trajectory synthesis and parameter estimation is provided with results using the Baxter Research Robot.

In all of the algorithms, the variational approaches to optimal control are shown to successfully improve the information metrics by orders of magnitude. Least-squares estimators are successfully able to provide values for the parameters using the data collected from the synthesized motions. These algorithms provide functional alternatives to purely sampling-based or periodic excitation trajectories for dynamical systems. Future investigation into managing process noise and unbiased errors will continue to improve automatic estimation techniques for use by robots interacting in the physical world.

Acknowledgements

The road to completing a dissertation is filled with help from numerous people including colleagues, friends, and family. First, I have been very fortunate to have an outstanding advisor, Todd Murphey, who has provided ideas and guidance throughout the entire PhD process. Despite the roadblocks and changes in research directions, he has been supportive of my goals and interests at all times.

I am also grateful for the input of my committee members – Brenna Argall, Ed Colgate, and Wendy Murray. In particular, our collaboration with Wendy's group provided much of the motivation regarding biomechanical models that this thesis is built on. As with most graduate research, a great deal of credit also goes to the fantastic people I've had the opportunity to share the lab with. Thanks to Alex Ansari, Adam Barber, Tim Caldwell, Elliot Johnson, Lauren Miller, David Pekarek, Jarvis Schultz, and Vlad Seghete for all the brainstorming and arguing on optimal control, calculus, Mathematica, ROS, Git, and everything in between.

Finally, I have been incredibly lucky to have the greatest support from my wife Beth every step of the way as we navigated graduate school together. I would also like to thank all of my family and friends for their support while not continuously trying to figure out what I've been working on. Now that it is finished, hopefully this thesis provides the answers to all of those questions.

Nomenclature

Symbol	Description
$\theta \in \mathbb{R}^p$	Parameter vector
$I(\theta)$	Fisher information matrix
β	Least-squares estimator cost
$\rho(x)$	Probability distribution
$\mathcal{L}$	Likelihood function
$x, x_d \in \mathbb{R}^n$	System states and desired states
$\bar{x}$	Extended system states
y, $\tilde{y} \in \mathbb{R}^h$	System output (predicted and measured)
$u \in \mathbb{R}^m$	Control vector
f	Dynamics model
g	Measurement model
w_y	Noise distribution
ϕ	First order parametric state sensitivity, $\frac{dx}{d\theta}$
Ω	Second order parametric state sensitivity, $\frac{d^2x}{d\theta^2}$
Γ_θ	Derivative of measurement model with respect to the parameters, $\frac{dg}{d\theta}$
Σ	Covariance matrix
J	Trajectory optimization cost
ξ, η	Trajectory with states and controls (η specifies a feasible trajectory)
$\zeta = (z, v)$	Variation along linearized trajectory (state and control)
α, μ	Infeasible states and controls
K	Projection matrix

Symbol	Description
A, B	Dynamics linearization matrices
Q, P, R	Weighting matrices (incremental state, terminal state, control)
κ	Condition number
λ_n	n^{th} Eigenvalue of a given matrix
L_c, L_d	Continuous and Discrete Lagrangians
$q_k \in \mathbb{R}^{n/2}$	Discrete configuration states
$p_k \in \mathbb{R}^{n/2}$	Discrete momentum states
$\rho_k \in \mathbb{R}^{n/2}$	Discrete kinematic configuration states
$D_x y(x,t)$	Partial derivative of y w.r.t. x
$D_1 y(x,t)$	Partial derivative of y w.r.t. the 1^{st} argument (i.e., x)

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Whether one realizes it or not, people are constantly creating and refining internal models of objects and themselves while interacting with the physical world. Physical motion at a macroscopic level is constrained by Newtonian mechanics with the classical relationships between forces and momentum. The ability to internalize a model of an object provides the means to predict motion and forces which aids in the interaction and control of physical systems. These models and constraints are referred to as the dynamics of a system. Classical control is a combination of feed-forward and feedback mechanisms with the feed-forward component synthesized directly from these internal models [**1**].

For robots and machines, models have manifested primarily in two ways. The first is the creation of differential equations representing the equations of motion of the system. The mathematical equations can be obtained from a variety of sources including first principle derivations and empirical formulations. This direct approach to mathematical modeling has been used for control of manipulators, motion planning, and countless other areas in robotics [**2**]. The second is the creation of models completely from collected data through the use of traditional machine learning techniques such as neural networks which use regression based approaches [**3**]. While both techniques are useful in many

different situations, completely unstructured machine learning of large-scale dynamic systems is currently unrealistic due to the amount of training that would be required to learn all aspects of the system. For people, this learning takes years of interaction with systems to master. Instead, this thesis focuses on the former technique of modeling with mathematical equations of motion derived from the structure of the system.

Even with a mathematical model, there are still significant sources of error that can occur from the use of feed-forward control techniques. The following are three of the most significant sources of error in feed-forward control:

Parametric Errors

Mathematical models, whether derived from first principles or empirical data, typically contain a set of parameters within the equations. Using parametric formulations allow models to be applied to general classes of systems rather than a single instance. Widely used parameters in physical systems include values such as mass and length, or quantities such as damping and friction coefficients. More abstract quantities can also be considered as parameters such as the mean and variance of probability distributions which could be used as components of a model. These parameters exist in the structure or topology of the model, allowing the model structure to remain the same while capturing the effects of changes in physical quantities such as mass.

In many cases, it may be advantageous to provide a robot with the topology of a model that it may interact with; however, the exact parameters within that model may be uncertain or unknown. This error will cause feed-forward predictions to be inaccurate;

however through interaction with an object, the robot may be able to estimate more accurate values of these parameters and much more quickly than if the model topology itself was unknown. This problem involving the refinement of model parameters encompasses the field of parameter identification [**4**]. Similar techniques of regression used in general machine learning are used in a structured manner leveraging knowledge of the model topology. The field of parameter identification extends far beyond robotics into almost all scientific fields including economics [**5**].

Unmodeled Dynamics and Model Errors

Models are inherently simplified representations of the physical reality; thus it is virtually impossible to completely capture all effects. At the extreme end, a complete model of physics would require the incorporation of principles down to microscopic interactions such as van der Waals forces, which is clearly unreasonable. So as simplifications, the model predictions will always contain some error due to unmodeled effects. Friction is a classic case where a number of empirical and principled models exist, including parametric models, but each has a vastly different structure.

For robots, some of these model errors may be reasonable to recover. An example of such would be automatically realizing the difference between a two and three link robot arm. Other errors may only be realized through the unstructured machine learning techniques previously mentioned. In many cases, the mechanism of feedback is relied on to complete a task despite these types of errors. This leads to the field of robust control which attempts to design controllers which will function with guarantees on quantities such as stability despite unstructured errors in the models [**6**].

Numerical Errors

Models of physical systems attempt to predict behavior over time, which is continuously evolving. Although the underlying mathematical models can be described using continuous differential equations, the solutions to these equations rarely admit analytic solutions. Therefore, numerical methods are required to find approximate solutions to the models over time. These numerical methods discretize time into a series of time steps and solve for the updated model state at each successive time step. Since these solutions are approximations, errors from the integration method itself can appear in the predicted system behavior depending on the time step used; however, too small of a time step can result in high computational load when simulating the system.

A number of different numerical methods exist for solving or simulating these models. Some common integrators include Euler and Runga-Kutta methods which directly discretize the equations of motion from the model. Euler integration is only accurate to the first order linearization of the model, which can result in model error. Runga-Kutta methods can have varying order, though a 4th order Runga-Kutta scheme is most commonly used. For mechanical systems, structured integrators also known as geometric integrators can offer even better results than standard methods. One such structured integrator is the Variational Integrator which derives a discretization of the dynamics from first principles [**7**]. This results in a symplectic method which is one that preserves a conserved quantity called the Hamiltonian over time. The Hamiltonian of a mechanical system corresponds to the total energy of the system. Work by Schultz and Murphey has demonstrated that using these structured integrators can improve the model prediction and performance of the robotic system [**8**, **9**].

In this thesis, parameter error and numerical error will be the focus while leaving the unmodeled dynamics to the feedback controller or other forms of control synthesis.

1.1. Experimental Design in Dynamical Systems

With regard to the parameter estimation problem, an area of extensive focus will be the experimental design of trajectories to estimate uncertain model parameters. In order to refine estimates of model parameters, one must generate an experiment to collect data which can be used to tune the parameter set. For the control of dynamical systems, this experiment involves the design of a trajectory which will be executed. During the trajectory execution, observations are compared against the model prediction and errors may be reduced by refining the parameter estimates. However, it is critical to note that not all trajectories are equal in this process. Certain maneuvers will provide better estimates of parameters than others. Once again, people develop an intuition to generate appropriate trajectories which provide information about uncertain parameters. As an example, when driving in the winter, it is desirable to know if the road is icy. A general response is to test the brakes, quickly decelerating the vehicle to test for slip. In this case, a deceleration trajectory will provide more information about road friction than a constant velocity trajectory.

In the same manner, it is useful to encode the ability to create informative experimental trajectories in robotic systems. Given the model structure, trajectories can be optimized such that they provide the best probability of accurately estimating parameters. This problem has evolved into the field of optimal design which includes a wide range of techniques including Persistent Excitation and Dual Control [**10**–**12**]. Using a

quantity called Fisher Information, an optimization problem can be created to determine desirable trajectories for parameter estimation in dynamical models.

1.2. Contributions and Thesis Outline

The primary contributions of this work include the derivation and experimental validation of several methods of information-based optimization using variational techniques. There has been significant work in both the field of optimal control for trajectory tracking and the estimation community. This work pulls heavily from both areas in an attempt to leverage the algorithmic foundations developed in both communities. The following are brief highlights of the remaining six chapters of this thesis and the specific contributions from each.

Chapter 2 aims to provide a sufficient introduction to the many mathematical concepts brought together throughout the thesis. The chapter begins with a brief overview of information-based metrics, of which Fisher information will be critical in the derivations to follow. An overview of parameter estimation follows, including the relationship to state estimation and filtering. Section 2.3 presents the optimal control theory and algorithms that will serve as the foundation to the algorithms developed in Chapters 4, 5, and 6. Finally, the chapter concludes with an introduction to discrete mechanics which is a fundamental component of the following chapter.

Chapter 3 serves as a case-study in simulation, motivating the focus on numerical methods and parameter estimation in the remainder of the thesis. A biomechanical model of the upper limb is discussed, and results from continuous and structured numerical

methods are presented. Additionally, parametric results are discussed, motivating the need for optimal trajectory synthesis in experiment design.

In Chapter 4, an algorithm is developed to optimize trajectories to improve estimation of model parameters. The results in this chapter are developed using continuous-time theory for two objectives. The first, presented in Section 4.2, optimizes the trajectory to maximize a metric on Fisher information to attempt to provide better precision in the estimation of a set of parameters. A lower bound on the estimator precision is related to Fisher information by the Cramer-Rao bound which is introduced in Chapter 2. The algorithm and subsequent results using a cart-double pendulum system are also published in [**13**, **14**]. The second contribution focuses on improving the convergence rate of the estimator by optimizing the trajectory to provide better conditioning of the estimator. This algorithm is experimentally validated with results from a single cart-pendulum system. The results and algorithm are also published in [**15**, **16**].

Chapter 5 takes the Fisher information maximization algorithm from the previous chapter and provides a parallel derivation using discrete mechanics. The results demonstrate a faster algorithm using a larger simulation time step, supporting the observations from the biomechanical case-study. The same double-cart pendulum system is used as a comparison of continuous and discrete optimization techniques. The algorithm and results are published in [**17**] as well.

Leveraging a recently developed algorithm by Ansari and Murphey [**18**], Chapter 6 takes the concepts used to develop the previous information optimization algorithms and creates an algorithm which is capable of real-time execution. The result is an iterative learning algorithm which simultaneously optimizes the experimental trajectory in a

receding-horizon fashion while providing parameter updates from an on-line least-squares estimator. Experimental trials using a Baxter Research Robot with a suspended payload are presented and also published in [**19**].

The thesis concludes in Chapter 7 with a discussion of the results from this work. Since a variety of algorithms are developed throughout this thesis, the trade-offs and criteria for selecting each algorithm for varying applications are also presented along with future research directions.

CHAPTER 2

Background

Since this thesis focuses on the nexus of estimation theory and trajectory optimization with respect to systems defined both in continuous and discrete mechanics forms, this chapter introduces the mathematical and algorithmic concepts as well as context for the techniques used in the remainder of the thesis. As a complete background on estimation theory and trajectory optimization are works unto themselves, the goal of this chapter is to provide consistent notation while introducing specific aspects of each area that are relavant to the remaining chapters.

The chapter begins with the definitions of important metrics in Information Theory and Statistics, including Fisher information, which will be the basis for developments in the remainder of the thesis. Section 2.2 presents the estimation algorithms used in this thesis and the relationship and context of parameter estimation to state estimation, including a discussion on the Kalman filter. Section 2.3 provides a background on trajectory optimization with a detailed introduction into the projection-based optimization approach used in Chapter 4 and the Sequential Action Control method used in Chapter 6. Lastly, Section 2.4 presents an overview of modeling systems using a discrete mechanics

approach and the framework for the projection-based trajectory optimization in discrete time which is used in Chapter 5.

2.1. Information Theory and Statistics Metrics

In sweeping terms, information theory encompasses a vast array of disciplines including finance, applied mathematics, computer science, and control theory. In robotics, information theory has been fundamental in the development of state estimation and optimal filtering techniques and continues to be a core concept in emerging areas such as symbol and natural language processing. In particular, two notions of information have dominated the field: Shannon information and Fisher information.

2.1.1. Shannon Information

When the term information theory is discussed, it is commonly referring to Shannon information. The field is defined by a single paper written by Claude Shannon in 1948 - *"A Mathematical Theory of Communication"* [**20**]. This work provides the theory to encode any signal into a set of bits which can be reconstructed after transmission. Additionally, quantitative analyses can be performed to minimize the loss of information during transmission and provide mechanisms for error correction to prevent corruption of signals. From this work, two basic measures of information emerged - entropy and mutual information.

Entropy is a means of encoding the expected uncertainty in a signal (represented by a random variable). Given a continuous probably density function (PDF), the entropy can

be written as

$$S(\rho) = -\int_{-\infty}^{\infty} \rho(x) \log \rho(x) dx,$$

where $\rho(x)$ is the probability distribution of random variable x.

On the other hand, mutual information represents uncertainty that is common to two different random variables. Given a joint PDF of the two variables, the mutual information is given by

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \rho(x_1, x_2) \log \frac{\rho(x_1, x_2)}{\rho(x_1)\rho(x_2)},$$

where $\rho(x_1, x_2)$ is the joint distribution of the two random variables (x_1, x_2).

In many applications such as clustering techniques and developing prediction methods, it is useful to maximize the mutual information of two random variable to try to maximize the dependence of the variables [**21**]. This is related to the maximization of information with respect to estimation; however, Fisher information has been the primary metric in terms of estimation and statistics which is presented in the following section.

2.1.2. Fisher Information

Another important metric, particularly used in the estimation and statistical analysis community, is Fisher information. As opposed to information-theoretic quantities like Shannon information which present global measures of uncertainty in a random variable, Fisher information is a local measure which is closely related to the relative entropy between an output variable and underlying parameter.

At the core of the Fisher information quantity lies the likelihood function. The likelihood is defined as the probability of a set of observed outcomes given a set of parameter

values, i.e.

$$\mathcal{L}(\theta|y) \triangleq \rho(y|\theta),$$

where θ is the parameter set and y is the set of observed random variables.

Fisher information is defined as the variance of the score function, a quantity in statistics defined by the first derivative of the log-likelihood function. The score is thus denoted by

$$V(\theta) \triangleq \frac{\partial}{\partial\theta} \log \mathcal{L}(\theta|y) = \frac{\frac{\partial}{\partial\theta}\mathcal{L}(\theta|y)}{\mathcal{L}(\theta|y)}.$$

The first moment of the score can be shown to be zero by

$$\mathbb{E}\left[\frac{\frac{\partial}{\partial\theta}\mathcal{L}(\theta|y)}{\mathcal{L}(\theta|y)}\middle|\theta\right] = \int \frac{\frac{\partial}{\partial\theta}\mathcal{L}(\theta|y)}{\mathcal{L}(\theta|y)}\mathcal{L}(\theta, y)dy = \frac{\partial}{\partial\theta}\int \mathcal{L}(\theta|y)dy = 0$$

since the integral of the likelihood function is always equal to 1. The variance, or second moment, then provides the Fisher information, $\mathcal{I}(\theta)$,

$$I(\theta) = \mathbb{E}\left[\left(\frac{\partial}{\partial\theta}\log \mathcal{L}(\theta|y)\right)^2\middle|\theta\right].$$

Assuming minimal regularity conditions, Fisher information can be written in terms of the second derivative of the log-likelihood function

$$I(\theta) = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta^2}\log \mathcal{L}(\theta|y)\middle|\theta\right].$$

2.1.3. Cramer-Rao Lower Bound

An important proof in estimation theory, which will be central to the results in thesis is the Cramer-Rao bound first derived by C. Rao in 1945 [**22**]. The result shows that the

variance of the parameter estimate provided by an unbiased estimator must be greater than or equal to the inverse of the Fisher information. Despite being only a lower bound on the covariance, in many simplified cases where the noise is assumed to be Gaussian, the bound is strict, and the estimator is called *efficient* [**23**]. The following is a simplified proof adapted from [**24**] for a single parameter and measurement which can be further generalized to multiple parameters and Gaussian measurements.

For an unbiased estimator, $\hat{\theta}$, using the Cauchy-Schwarz inequality and the fact that $\text{cov}(V, \hat{\theta}) = \text{cov}(V\hat{\theta})$ since $\mathbb{E}(V) = 0$,

$$\begin{aligned}
\text{var}(V)\text{var}(\hat{\theta}) &\geq \text{cov}(V, \hat{\theta})^2 \\
\mathbb{E}[(V - \mathbb{E}[V])^2] \cdot \mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2] &\geq \mathbb{E}\left[(V - \mathbb{E}[V])(\hat{\theta} - \mathbb{E}[\hat{\theta}])\right]^2 \\
\mathbb{E}[V^2 - 2V\mathbb{E}[V] + \mathbb{E}[V]^2] \cdot \text{var}(\hat{\theta}) &\geq \mathbb{E}[V\hat{\theta} - \mathbb{E}[V]\hat{\theta} - V\mathbb{E}[\hat{\theta}] + \mathbb{E}[V]\mathbb{E}[\hat{\theta}]]^2.
\end{aligned}$$

Since the expectation of the score, $\mathbb{E}[V] = 0$ as previously noted, the equation simplifies to the following

$$\begin{aligned}
\mathbb{E}[V^2] \cdot \text{var}(\hat{\theta}) &\geq \mathbb{E}[V\hat{\theta} - V\mathbb{E}[\hat{\theta}]]^2 \\
I(\theta) \cdot \text{var}(\hat{\theta}) &\geq \left(\mathbb{E}[V\hat{\theta}] - \mathbb{E}[V]\mathbb{E}[\hat{\theta}]\right)^2 \\
I(\theta) \cdot \text{var}(\hat{\theta}) &\geq \mathbb{E}[V\hat{\theta}]^2.
\end{aligned}$$

Substituting the definition of the expectation into the right hand side results in the following equation

$$I(\theta) \cdot \text{var}(\hat{\theta}) \geq \left(\int_{-\infty}^{\infty} \frac{\frac{\partial}{\partial \theta} \mathcal{L}(\theta|y)}{\mathcal{L}(\theta|y)} g(y) \mathcal{L}(\theta|y) dy \right)^2$$

$$I(\theta) \cdot \text{var}(\hat{\theta}) \geq \left(\frac{\partial}{\partial \theta} \int_{-\infty}^{\infty} \hat{\theta}(y) \mathcal{L}(\theta|y) dy \right)^2$$

$$I(\theta) \cdot \text{var}(\hat{\theta}) \geq \left(\frac{\partial}{\partial \theta} \mathbb{E}[\hat{\theta}] \right)^2.$$

Since the estimator is assumed to be unbiased, $\mathbb{E}(\hat{\theta}) = \theta$, and thus

$$I(\theta)\text{var}(\hat{\theta}) \geq 1.$$

Therefore,

$$\text{var}(\hat{\theta}) \geq \frac{1}{I(\theta)}.$$

In the case of a parameter vector with multiple unknowns, the Fisher information metric can be represented as a matrix quantity. The Cramer-Rao bound then becomes

$$\text{cov}(\hat{\theta}) \geq I(\theta)^{-1}. \tag{2.1}$$

To create a metric from the Fisher information matrix, various optimality metrics can be used, including A-, D-, and E-optimal designs. The use of these metrics are discussed in further detail in Chapter 4.

2.1.4. Summary

The field of information theory continues to progress with recent developments concerning the relationships and connections between various information quantities such as mutual information and Fisher information [**25**, **26**].

Since the scope of this thesis focuses on the development of trajectories in the context of estimation and filtering, Fisher information coupled with maximum likelihood estimation will serve as the primary tool driving the work. The following section continues to expand on the role of information in estimation and filtering with a survey of several classical estimators and filtering techniques.

2.2. Estimation and Filtering Overview

While the contributions in this thesis are limited to fixed parameter estimation problems, this section will give an overview of those estimation methods as well as provide background on extensions and relations to time-varying parameter and state estimation problems, which are natural extensions to the work in this thesis.

2.2.1. Nonlinear Least-Squares Estimation

The goal of a least-squares estimation problem is to fit a vector of observations y with a model containing a vector of unknown parameters θ. A least-squares cost is used in evaluating the model fit to an estimate of the parameter vector. This method is equivalent to maximum likelihood estimation in the case of Gaussian noise [**23**]. Using a set of measurements, nonlinear least-squares estimation can be performed using a gradient descent

method, Iterated Least-Squares quasi-Newton approach, or a full Newton-Raphson search method.

For any of these approaches, the nonlinear least-squares method assumes a system model and observer

$$\begin{aligned} \dot{x}(t) =& f(x(t), u(t), \theta) \\ y(t) =& g(x(t), u(t), \theta) + \mathrm{w}_y \end{aligned} \tag{2.2}$$

where $x \in \mathbb{R}^n$ defines the system states, $y \in \mathbb{R}^h$ defines the measured outputs, $u \in \mathbb{R}^m$ defines the inputs to the system, $\theta \in \mathbb{R}^p$ defines the set of model parameters to be estimated, and w_y is additive output noise where $p(\mathrm{w}_y) = N(0, \Sigma)$.

The least-squares estimator can be written as

$$\hat{\theta} = \arg\min_{\theta} \beta(\theta) \tag{2.3}$$

where

$$\beta(\theta) = \frac{1}{2} \sum_{i}^{t_f/dt} (\tilde{y}(t_i) - y(t_i))^T \cdot \Sigma^{-1} \cdot (\tilde{y}(t_i) - y(t_i)). \tag{2.4}$$

$\tilde{y}(t_i)$ is the observed state at the i^{th} index of t_f/dt measurements, $\Sigma \in \mathbb{R}^{h \times h}$ is the covariance matrix associated with the sensor measurement error, and $\hat{\theta}$ is the least-squares estimate of the parameter set.

2.2.1.1. Gradient Descent Method. Gradient descent techniques are commonly used in optimization problems due to their simplicity and that only the first derivative of the

objective function needs to be calculated. The first derivative can be found by differentiating (2.4) with respect to θ, resulting in the following equation:

$$\frac{d}{d\theta}\beta(\theta) = \sum_{i}^{t_f/dt} (\tilde{y}(t_i) - y(t_i))^T \cdot \Sigma^{-1} \cdot \Gamma_\theta(t_i) \tag{2.5}$$

where Γ_θ represents $\frac{d}{d\theta}g(x, u, \theta)$ given by

$$\begin{aligned} \Gamma_\theta(t_i) =& D_x g(x(t_i), u(t_i), \theta) \cdot \frac{d}{d\theta} x(x(t_i), u(t_i), \theta) \\ &+ D_\theta g(x(t_i), u(t_i), \theta). \end{aligned} \tag{2.6}$$

This equation requires the evaluation of $\frac{d}{d\theta}x(x(t_i), u(t_i), \theta)$ of (2.2), which is computed by the following ordinary differential equation (ODE):

$$\dot{\psi}(t) = D_x f(x(t), u(t), \theta) \cdot \psi(t) + D_\theta f(x(t), u(t), \theta), \tag{2.7}$$

where

$$\psi(t) = \frac{d}{d\theta} x(x(t), u(t), \theta) \in \mathbb{R}^{n \times p} \text{ and } \psi(0) = \{0\}^{n \times p}.$$

The estimation algorithm is applied as shown by Algorithm 2.1. An Armijo backtracking line-search is used at every iteration to ensure that the new parameter estimate sufficiently decreases the least-squares cost [**27**].

With respect to algorithm convergence, gradient descent optimization ensures convergence to a local minimizer; however, the rate of convergence can be extremely slow in practice. Given a strongly convex least-squares problem, it has been shown in prior

Algorithm 2.1 Gradient Descent Parameter Estimation

Choose initial $\theta_0 \in \mathbb{R}^p$, tolerance ϵ
while $\frac{d}{d\theta}\beta_p(\theta_i) > \epsilon$ **do**
  $d_i = -\frac{d}{d\theta}\beta_p(\theta_i)$ from (2.5)
  Compute γ_i using Armijo backtracking search
  $\theta_{i+1} = \theta_i + \gamma_i d_i$
  $i = i + 1$

literature [**28**] that the error of the parameter estimates follows

$$\frac{\|\theta_{i+1} - \theta^*\|}{\|\theta_i - \theta^*\|} \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right),$$

where λ_n and λ_1 are the respective maximum and minimum eigenvalues of the Hessian of β. This result indicates that as the condition number $\kappa = \lambda_n/\lambda_1$ increases, the convergence rate of the gradient descent algorithm degrades.

2.2.1.2. Iterated Least-Squares Method. Since gradient descent-based optimization has slower convergence with increasingly ill-conditioned problems, Hessian-based methods may be used to improve convergence rates. The Iterated Least-Squares method uses an approximation of the Hessian to perform a quasi-Newton type optimization [**23**]. For the Iterated Least Squares method, the parameter estimate update is given by the following,

$$\theta_{i+1} = \theta_i + \left[\sum_{i}^{t_f/dt} \Gamma_\theta(t_i)^T \cdot \Sigma^{-1} \cdot \Gamma_\theta(t_i)\right]^{-1} \cdot \frac{d}{d\theta}\beta(\theta_i) \tag{2.8}$$

To show how one obtains the approximate Hessian used in (2.8), the full Hessian matrix will be derived in a similar manner to the first derivative. Taking the derivative

of (2.5) results in the following:

$$\frac{d^2}{d\theta^2}\beta(\theta) = \sum_i^h \Gamma_\theta(t_i)^T \cdot \Sigma^{-1} \cdot \Gamma_\theta(t_i) + (\tilde{y}(t_i) - y(t_i))^T \cdot \Sigma^{-1} \cdot \left[D_x g(\cdot) \cdot \frac{d^2}{d\theta^2} x(\cdot) + D_x^2 g(\cdot) \cdot \psi(t_i) + D_\theta^2 g(\cdot) \right]. \tag{2.9}$$

To calculate the exact Hessian, a second differential equation for $\frac{d^2}{d\theta^2}x$ must be calculated; however, near the optimal parameter set, $\mathbb{E}[\hat{\theta}]$, $(\tilde{y}(t_i) - y(t_i)) \approx 0$. Therefore, assuming that the estimated parameters are near the optimal set, the Hessian is approximated as

$$\frac{d^2}{d\theta^2}\beta(\theta) \approx \sum_i^h \Gamma_\theta(t_i)^T \cdot \Sigma^{-1} \cdot \Gamma_\theta(t_i). \tag{2.10}$$

Although the Iterated Least-Squares algorithm improves convergence rates near the optimal parameter set compared to the gradient descent method, for highly ill-conditioned problems, undesirable convergence rates may exist.

2.2.1.3. Newton-Raphson Search Method. For exact second-order convergence results, Newton's method with a backtracking line-search can be used to find optimal parameter values. The Hessian is computed from (2.9), and in this case, the formulation for the Hessian requires the evaluation of $\frac{d^2}{d\theta^2}x(\cdot)$.

This term is given by the following ODE:[1]

$$\dot{\Omega}(t) = \Big[\big[D_x^2 f(\cdot)\cdot\psi(t) + D_\theta D_x f(\cdot)\big]^{T_{(1,3,2)}}\cdot\psi(t)\Big]^{T_{(1,3,2)}} + D_x f(\cdot)\cdot\Omega(t) + D_x D_\theta f(\cdot)\cdot\psi(t) + D_\theta^2 f(\cdot),$$

where

$$\Omega(t) = \frac{d^2}{d\theta^2}x(\cdot) \in \mathbb{R}^{n\times p\times p} \text{ and } \Omega(0) = \{0\}^{n\times p\times p}.$$

Given this second order differential equation, the parameter estimate update is given by the following,

$$\theta_{i+1} = \theta_i + \left[\frac{d^2}{d\theta^2}\beta(\theta_i)\right]^{-1}\cdot\frac{d}{d\theta}\beta(\theta_i). \tag{2.11}$$

Using the Hessian results in second-order convergence of the parameter update compared to first-order convergence for gradient descent and between first and second-order convergence for the ILS method.

2.2.2. Minimum Mean-Square Error Estimation

In the least-squares estimation problem, the parameter values are not assumed to be random variables, simply unknowns. The problem of estimating random variables from uncertain measurements is handled by a minimum mean-square error estimator.

Given two random vectors which are jointly Gaussian, the estimate of x, the variable to be estimated, given the measurement vector y is given by the conditional mean $\mathbb{E}[x|y]$.

[1]Since the equation for $\Omega(t)$ involves tensors, we will use the notation $[\cdot]^{T_{(1,3,2)}}$ to indicate a transpose of the 2nd and 3rd tensor dimensions.

In the case of joint Gaussians, the conditional mean of the estimate $\hat{x}$ is given by

$$\hat{x} \triangleq \mathbb{E}[x|y] = \bar{x} + \Sigma_{xy}\Sigma_y^{-1}(y - \bar{y}), \tag{2.12}$$

and the conditional covariance matrix is given by

$$\Sigma_{x|y} \triangleq \mathbb{E}[(x - \hat{x})(x - \hat{x})^T|y] = \Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{yx}, \tag{2.13}$$

where $\hat{x}$ and $\hat{y}$ are the mean values of the estimated random variable and the measurements respectively.

In the case of linear, or linearized systems, the minimum mean-square error estimator (MMSE) for a Gaussian random variable with zero mean is equivalent to the least-squares estimator. This is due to the fact that the Gaussian system provides a linear estimate for the MMSE, which is exactly the same as the constrained linear least-squares estimator.

The relationship of least-squares estimation to the MMSE can be readily seen comparing the gradient descent update (2.5) to the MMSE update (2.12). Assuming a single measurement and full Armijo step, the gradient descent update can be written as

$$\theta_{i+1} = \theta_i + \Gamma_\theta(t_i) \cdot \Sigma^{-1} \cdot (\tilde{y}(t_i) - y(t_i))^T. \tag{2.14}$$

The previous estimate serves as the mean $\bar{x}$ with the derivative of y w.r.t. the parameters encoding the coupling of the variables. In the MMSE, this coupling is given by the cross-covariance Σ_{xy}. The next section takes one more step forward in relating static parameter estimation to the time-varying state estimation problem which can be computed with the Kalman filter.

2.2.3. State Estimation and the Kalman Filter

The Kalman filter is a fundamental algorithm which provides a solution to the state estimation problem for linear systems. Instead of working on static parameters, the filter provides an estimate for the states of the system which can be viewed as time varying random variables. Despite this difference, the concepts of minimum mean-square error estimation are relevant and useful in the derivation of the filter.

Each iteration of the Kalman filter algorithm consists of mapping a state estimate and associated covariance matrix to the next point in time. We focus on the discrete time case for purposes of the comparison to parameter estimation. Given a linear dynamic system, the discrete update equations are

$$x_{k+1} = F_k x_k + G_k u_k + v_k$$

$$y_k = H_k x_k + w_k$$

where v_k and w_k are zero-mean noise signals with covariances Q_k and R_k respectively.

Starting with the mean $\bar{\theta}$ of (2.12) and covariance Σ_x of (2.13), the corresponding terms in the Kalman filter are the predicted state and covariance from the linear system,

$$\bar{x} \to \hat{x}_{k+1|k} = F_k \hat{x}_{k|k} + B_k u_k$$

$$\Sigma_x \to P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k$$

with the measurement mean $\bar{y}$ corresponding to the predicted measurement,

$$\bar{y} \to \hat{y}_{k+1|k}.$$

The measurement covariance Σ_y corresponds to the predicted measurement covariance in the Kalman filter with the following

$$\Sigma_y \to S_{k+1} = H_{k+1} P_{k+1|k} H_{k+1}^T + R_{k+1},$$

and the predicted covariance between the state and measurement is given by

$$\Sigma_{xz} \to P_{k+1|k} H_{k+1}^T.$$

Therefore, the updated state estimate $\hat{x}$ given a new measurement y_{k+1} follows the form of (2.12) with

$$\hat{x} \to \hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + P_{k+1|k} H_{k+1}^T S_{k+1}^{-1} (y_{k+1} - \hat{y}_{k+1|k}). \tag{2.15}$$

The covariance update follows (2.13) with

$$\Sigma_{x|y} \to P_{k+1|k+1} = P_{k+1|k} - (P_{k+1|k} H_{k+1}^T) S_{k+1}^{-1} (P_{k+1|k} H_{k+1}^T)^T.$$

The quantity $P_{k+1|k} H_{k+1}^T S_{k+1}^{-1}$ is referred to as the Kalman gain since it is the optimal weight on the measurement residual in (2.15) to update the state estimate.

The takeaway from the Kalman filter comparison is that state estimation can be viewed as a form of recursive parameter estimation, only the parameters are states which are time-varying. The opposite view is likewise true, that parameter estimation can be performed as state estimation, though static parameters should be treated as static states, i.e. the predicted state update will be equivalent to the previous state.

This notion provides a means of extrapolating methods for static parameter estimation studied in this thesis to time-varying parameters and state-based forms. Though outside the scope of this thesis, the relationship is important and discussed as future work in Chapter 7.

2.3. Trajectory Optimization and Optimal Control

Trajectory optimization is a common problem in the field of optimal control. The goal is determine a time-varying set of controls for a dynamic system that minimize a certain objective function. Typically in a trajectory tracking problem, the cost is represented by a running cost that is a function of state and control input and a terminal cost on the state. This cost function can be represented as

$$J = \int_{t_0}^{t_f} l(x(t), u(t), t)dt + s(x(t_f)) \tag{2.16}$$

with the system dynamics constraining the state evolution as

$$\dot{x}(t) = f(x(t), u(t)).$$

A variety of techniques are used to solve these types of control problems; however, numerical solution methods generally are categorized as either indirect or direct methods.

Direct methods of optimal control involve parameterizing the dynamics and cost function of the problem in a manner that can be formulated as a non-linear programming problem (NLP). NLP problems can be solved using a variety of optimization algorithms including active-set methods such as SQP and BFGS or a trust-region method [**28**]. Direct

methods provide a relatively straightforward optimization implementation with a number of commercially available software tools. The disadvantages of direct methods lie in the parameterization of the problem which can lead to infeasible trajectories if the dynamics are coarsely parameterized or very high dimensional optimization problems which can be unreasonable to solve.

Indirect methods of optimal control attempt to directly solve the optimization problem with respect to the differential equation constraint. Generally, indirect methods are formulated using the calculus of variations, resulting in two-point boundary value problems. Pontryagin's Maximum Principle is a classical result using variational calculus to produce the optimal controls given state and control constraints [**29**]. While the indirect techniques can maintain a low dimensional state, solutions for the boundary value problem are not universal and can be difficult to solve, even numerically, depending on the system.

Given these trade-offs, the results in this thesis rely on two optimal controls methods as the basis upon which the information optimization is built. Section 2.3.1 describes the projection-based optimization approach developed by Hauser [**30**] and extended to discrete mechanical systems by Johnson and Murphey [**31**] which is used in Chapters 4 and 5. This algorithm leverages indirect solutions to linear quadratic control problems in an iterative framework. Section 2.3.2 provides an overview of the Sequential Action Control method used in Chapter 6. This method provides a fast way to compute the current control action based on future dynamics in a receding-horizon fashion.

2.3.1. Projection-based Trajectory Optimization

The projection-based approach for trajectory optimization tackles the problem of providing optimal control solutions using variational calculus while generalizing the methods for any nonlinear system. The premise of the algorithm involves the iterative linearization around a current trajectory followed by an optimal perturbation solution based on LQR techniques with a constraint projection. The original formulation of the algorithm in continuous time is presented in [**32**] and the discrete formulation can be found in [**33**]. This section serves a basic introduction to the algorithm which is used throughout the examples in this thesis.

We define $\mathcal{T}$ to be the set of admissible trajectories for a dynamic system, i.e., trajectories that satisfy the dynamics constraint equations. The space of trajectories is embedded in an inner product space $\mathcal{V}$ such that $\mathcal{T} \subseteq \mathcal{V}$. Therefore $\mathcal{T} = \{\eta = (x(t), u(t)) \in \mathcal{V} : \dot{x}(t) = f(x(t), u(t), t)\}$ where η denotes a feasible trajectory.

Given an initial trajectory $\xi_0 = (x(t), u(t))$, the optimization problem can be defined as

$$\eta^* = \arg\min_{\eta \in \mathcal{T}} J(\eta)$$

where $J(\eta)$ is defined by (2.16).

In the case of trajectory tracking, it is common to have a quadratic norm on the state error using a running and terminal cost. This results in the following terms of $J(\eta)$

$$l(x(t), u(t), t) = (x(t) - x_d(t))^T \cdot Q(t) \cdot (x(t) - x_d(t)) + u(t)^T \cdot R(t) \cdot u(t)$$

$$s(x(t_f)) = (x(t_f) - x_d(t_f))^T \cdot Q(t_f) \cdot (x(t_f) - x_d(t_f))$$

Algorithm 2.2 Trajectory Optimization

Initialize $\eta_0 \in \mathcal{T}$, tolerance ϵ, $k = 0$
while $DJ(\eta_k(t)) \circ \zeta_k > \epsilon$ **do**
  Calculate descent, ζ_k:
    $\zeta_k = \arg\min_{\zeta_k(t)} DJ(\mathcal{P}(\xi_k(t))) \circ \zeta_k + \frac{1}{2}\langle \zeta_k, \zeta_k \rangle$
  Compute γ_k with Armijo backtracking search
  Calculate the infeasible step:
    $\xi_k(t) = \eta_k(t) + \gamma_k \zeta_k$
  Project trajectory onto dynamics constraints:
    $\eta_{k+1}(t) = \mathcal{P}(\xi_k(t))$
  $k = k + 1$

where Q is positive semi-definite matrix, and R is a positive definite matrix which may both be a function of time. This cost will optimally track a desired path even if the path is not dynamically feasible.

Since the constrained optimization problem presents difficulties in direct solutions, it is advantageous to relax the dynamics constraints, solve an unconstrained optimization problem, and then use a projection operator to project the solution to a nearby feasible trajectory. The main steps of the optimization algorithm are shown in Algorithm 2.2.

2.3.1.1. Projection Operator. The projection operator uses a stabilizing feedback law to take a feasible or infeasible trajectory, defined by $\xi(t) = (\alpha(t), \mu(t))$, and maps it to a feasible trajectory, $\eta(t) = (x(t), u(t))$. The local, iterative approach to the algorithm allows the use of projection operators that are only locally stabilizing rather than requiring global stability.

While any stabilizing feedback law will suffice, the work in this thesis uses the solution to an LQR optimal control problem as the feedback regulator. The form of the projection

operator then becomes

$$P(\xi(t)) : \begin{cases} u(t) = \mu(t) + K(t)(\alpha(t) - x(t)) \\ \dot{x}(t) = f(x(t), u(t)). \end{cases}$$

The feedback gain $K(t)$ can be optimized as well by solving an additional linear quadratic regulation problem. Details of the optimal gain problem can be found in [**32**], but any stabilizing feedback may be used.

With the addition of the projection operator, the problem can be reformulated from an optimization over the constrained trajectory $\eta(t)$ to an optimization over the unconstrained trajectory $\xi(t)$. This relation is given by

$$\arg\min_{\eta(t)} \; J(\eta(t)) \quad \longleftrightarrow \quad \arg\min_{\xi(t)} \; J(\mathcal{P}(\xi(t))).$$

The unconstrained formulation allows variations of the trajectory to be calculated free of the constraint of maintaining feasible dynamics; however, the solution is projected to a feasible trajectory at each iteration of the optimization algorithm.

2.3.1.2. Calculating the Descent Direction. In the iterative optimization algorithm shown in Algorithm 2.2, a descent direction ζ must be computed based on the linearizations around the current trajectory ξ_k. This is achieved by minimizing a local quadratic approximation of the cost function. This approximation can be represented as

$$\zeta_k^* = \arg\min_{\zeta \in T_{\xi_k}\mathcal{T}} DJ(\mathcal{P}(\xi_k)) \circ \zeta_k + \frac{1}{2}\langle \zeta_k, \zeta_k \rangle, \tag{2.17}$$

where $\zeta = (z, v)$ is constrained to the linearized dynamics,

$$\dot{z}(t) = A(t)z(t) + B(t)v(t) \tag{2.18}$$

and $\zeta_k(t) \in T_{\eta_k}\mathcal{T}$, i.e., the descent direction for each iteration lies in the tangent space of the trajectory manifold at η_k. The components of the descent direction $\zeta_k = (z(t), v(t))$ are defined by $z(t)$, the perturbation to the extended state, and $v(t)$, the perturbation to the control. The quantity $\langle \zeta_k(t), \zeta_k(t) \rangle$ is a local quadratic model computed as an inner product of $\zeta_k(t)$. Matrices $A(t)$ and $B(t)$ are the linearizations of the dynamics. Since (2.17) is a quadratic function of ζ_k with linear constraints, the descent direction can be solved as a time-varying LQ problem which depends on the linearization of the cost function, $DJ(P(\xi_k(t)))$, and the local quadratic model, $\langle \zeta_k(t), \zeta_k(t) \rangle$.

Applying a quadratic model and expanding the linearizations of the cost function, (2.17) can be rewritten as

$$\begin{aligned} \arg\min_{\zeta_k} = \int_{t_0}^{t_f} & a(t)^T z(t) + b(t)^T v(t) + \frac{1}{2} z(t)^T Q_n z(t) + \frac{1}{2} v(t)^T R_n v(t)\ dt \\ & + r_1^T z(T) + \frac{1}{2} z(T)^T Q_T z(T), \end{aligned} \tag{2.19}$$

where $a(t)$, $b(t)$, and r_1 are the linearizations of the cost function with respect to x and u, and Q_n, Q_T, and R_n are weighting matrices for the local quadratic model approximation. Design of these weighting matrices can lead to faster convergence of the optimal control algorithm depending on the specific problem.

The LQ problem can be solved with three Ricatti equations resulting in an affine feedback law minimizing (2.19).

2.3.1.3. Backtracking Line-Search. Given the descent direction ζ_k, a backtracking line-search of the projection, $P(\eta_k(t) + \gamma_k \zeta_k)$, provides a feasible trajectory assuming the step size γ_k satisfies the Armijo sufficient decrease condition [**27**]. Iterations upon the feasible trajectories continue until a given termination criteria is achieved.

2.3.2. Sequential Action Control

The SAC control synthesis process follows a receding-horizon style format to sequence together separately short optimal control *actions* into a piecewise continuous constrained feedback response to state. In SAC, *actions* are defined by a pair composed of a control vector value and its associated (typ. short) application duration which is computed for each action. The blue shaded region in Fig. 2.1 shows a SAC action for a 1-D control.

The algorithm iterates on the following 4-step process to synthesize each action:

2.3.2.1. Predict. The SAC algorithm predicts system motion from current state feedback, $x(t_0) = x_0$. The process involves simulation of a state and adjoint system (x, ρ) for a fixed time horizon, T, until (receding) final time $t_f = T + t_0$. For the purposes of this paper, the nominal control value used for simulations (x, ρ) is $u = 0$ so that SAC computes optimal actions relative to the free (unforced) system motion.

The cost function for SAC is given by

$$J_{cost} = \int_{t_0}^{t_f} l(x(t))\, dt + s(x(t_f)), \tag{2.20}$$

where

$$l(x(t)) = x(t)^T \cdot Q_\tau \cdot x(t),$$

i.e., the running cost involves minimization of the inverse of the information and an optional trajectory tracking cost to bias the system toward a particular part of the state space.

The adjoint variable $\rho : \mathbb{R} \mapsto \mathbb{R}^n$ provides information about the sensitivity of the cost function to the state, x. The algorithm maps this sensitivity to a control sensitivity provided by an inner product between the adjoint and dynamics (2.2). The process of control synthesis uses this sensitivity, $\frac{dJ_{cost}}{du_\tau}$, to search for least norm actions that optimize the expected change in cost (2.20). Thus SAC actions optimize the rate of trajectory improvement. These optimal actions depend directly on the adjoint, which is determined from open-loop simulation of the following equation,

$$\dot{\rho} = -D_x l(x)^T - D_x f(x,u)^T \rho$$

with a terminal condition $\rho(t_f) = \nabla s(x(t_f))$.

2.3.2.2. The Value of Optimal Action. After simulating the open-loop system (x, ρ) under nominal control, SAC computes the value of actions that it should apply to optimally improve nominal trajectory cost (2.20). Because SAC has not yet decided on a time, τ, specifying *when* to act,[2] it searches for a curve u^* that provides the value of the optimal action it should apply as a function of time. The algorithm will search this curve to determine the best (optimal) time to act. The entire curve that provides the value of the optimal actions that maximize trajectory improvement is provided analytically from

[2]Each cycle of control synthesis assumes SAC will specify / search for an optimal time to act between the current time, t_0, and the end of the prediction horizon, t_f.

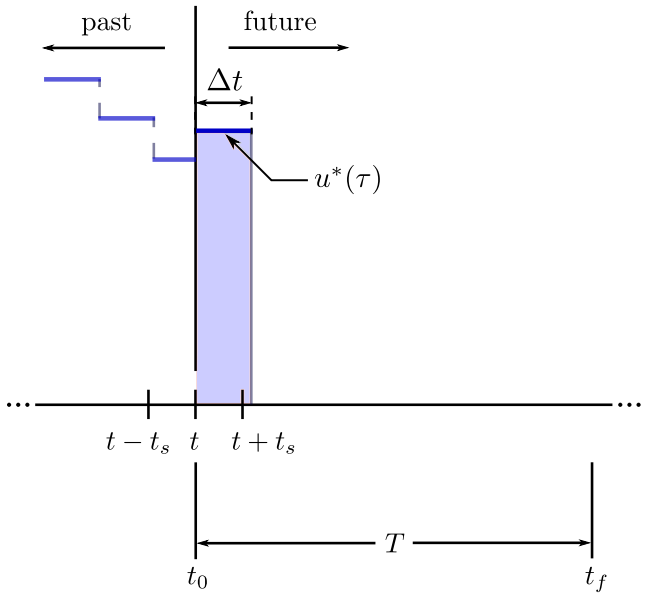


Figure 2.1. SAC actions for a 1-D control are sequenced in receding-horizon fashion.

the expression,

$$u^* = (\Lambda + R^T)^{-1}\, h(x)^T \rho\, \alpha_d \tag{2.21}$$

with $\Lambda \triangleq h(x)^T \rho \rho^T h(x)$ [**18**].

As opposed to traditional receding-horizon and nonlinear trajectory optimization routines, SAC actions optimally *improve* rather than *minimize* a trajectory cost over the current horizon. The process is often less sensitive to local minima and examples in [**18**] show that actions based on (2.21) actually outperform nonlinear trajectory optimization in terms of cost and computational efficiency on a variety of benchmark nonlinear control problems. Furthermore, [**18**] proves min/max input constraints of the form $u_{min,k} < 0 < u_{max,k}\ \forall k \in \{1, \ldots, m\}$, can be applied on-line to actions from (2.21) without added computation.

2.3.2.3. When to Act. In application, SAC can pre-specify an action time $\tau = t_0$ so that actions are always applied from the current time and sequenced into a piecewise continuous, receding-horizon style response to state. However, SAC includes the option to search for an optimal time to act. It can therefore do nothing and choose to wait until a system "drifts"[3] into a more opportune state where control effort would be better spent. In searching for an optimal time to act, SAC optimizes an objective measuring the cost of waiting relative to the expected efficacy of controls (2.21) in improving system performance based on (2.20). It searches for the time, τ that minimizes the objective

$$J_t(\tau) = \|u^*(\tau)\| + \left.\frac{dJ_{cost}}{du_\tau}\right|_\tau + (\tau - t_0)^\gamma$$

with $\gamma = 1.6$ and $\frac{dJ_{cost}}{du_\tau}$ as the measure of expected cost improvement (see [**18**]).

2.3.2.4. How Long to Act. The SAC algorithm computes a duration, Δt, to apply control action values (2.21) using a line search with simple descent condition. The line search process iteratively reduces an initial duration, $\Delta t = \Delta t_0$, and simulates the expected cost (2.20) based on application of the control action around the application time τ. If the current duration results in a minimum improvement in cost (relative to the nominal control), it is selected and the action is fully specified based on the pair $(u^*(\tau), \Delta t)$. If the action does not provide the expected cost improvement, the line search reduces the duration and repeats. The process is guaranteed to find a duration that provides a minimum expected improvement in cost [**34**].

For a more detailed derivation of SAC control synthesis with examples see [**18**, **34**].

[3]Under free dynamics and the nominal control $u = 0$.

2.4. Discrete Mechanics

The results in Chapter 5 use a discrete mechanics framework to derive the information optimization algorithm. The discrete mechanics framework can be particularly useful in trajectory optimization as it provides a way to create sympletic integrator called a variational integrator which has stable long-term energy behavior, even with large time-steps [**35**]. This section provides an introduction into discrete mechanical system representations including variational integrators.

Given a system with a configuration $q \in Q$, where Q is the configuration space, a sequence $\{(t_0, q_0), (t_1, q_1), ..., (t_n, q_n)\}$ can be found that approximates a trajectory in continuous time where $q_k \approx q(t_k)$. Instead of numerically integrating differential equations derived from a continuous Lagrangian, L_c, a discrete Lagrangian, L_d, is chosen such that the action integral is approximated over a discrete time step

$$L_d(q_k, q_{k+1}, \theta) \approx \int_{t_k}^{t_{k+1}} L_c(q(t), \dot{q}(t), \theta) dt.$$

The action integral can then be approximated by a discrete action sum given by

$$\int_{t_0}^{t_f} L_c(q(\tau), \dot{q}(\tau), \theta) d\tau \approx \sum_{k=0}^{k_f} L_d(q_k, q_{k+1}, \theta).$$

Given the discrete action sum, the DEL equations, including forcing, have been derived for the discrete-time system [**7**]. The unconstrained DEL equations are given by[4]

$$D_2 L_d(q_{k-1}, q_k, \theta) + F_d^+(q_{k-1}, q_k, u_{k-1}, \theta) +$$
$$D_1 L_d(q_k, q_{k+1}, \theta) + F_d^-(q_k, q_{k+1}, u_k, \theta) = 0.$$

where $F_d^{\pm}$ are the right and left discrete force approximations of a continuous forcing model, F_c. For clarity, the arguments to the discrete Lagrangian will be dropped and the indices will be referred to using the following subscript notation,

$$L_{k+1} = L_d(q_k, q_{k+1}, \theta)$$
$$F_{k+1}^{\pm} = F_d^{\pm}(q_k, q_{k+1}, u_k, \theta).$$

In order to create a suitable analog to the continuous time framework, the DEL equations can be rewritten in a one-step map using a discrete momentum term, p_k. The first equation is implicit in terms of q_{k+1} and is solved using a root-finding algorithm. The following is the one-step update of $\{q_k, p_{k+1}\}$

$$p_k + D_1 L_{k+1} + F_{k+1}^- = 0$$
$$p_{k+1} = D_2 L_{k+1} + F_{k+1}^+. \tag{2.22}$$

[4]The slot derivative notation, $D_\kappa \alpha(x, y, z)$ represents the partial derivative of α w.r.t the κ^{th} argument.

Therefore the discrete states, x_k are of the form

$$x_k = \begin{bmatrix} q_k \\ \\ p_k \end{bmatrix}.$$

2.4.1. Variational Integrators

In order to solve the DEL equations, the discrete Lagrangian and discrete forcing functions must be specified. A variational integrator is created by specifying the approximation used to convert the continuous-time Lagrangian to a discrete-time analog. For the scope of this paper, the most common form with midpoint approximation is used. The discrete Lagrangian is approximated as

$$L_d(q, q_{k+1}, \theta) = L_c\left(\frac{q_{k+1} + q_k}{2}, \frac{q_{k+1} - q_k}{\Delta t}, \theta\right). \tag{2.23}$$

The discrete forcing function is chosen using the same approximation resulting in the following discrete forces

$$F_d^-(q_k, q_{k+1}, u_k, \theta) = f_c\left(\frac{q_{k+1} + q_k}{2}, \frac{q_{k+1} - q_k}{\Delta t}, u_k\right)\Delta t$$

$$F_d^+(q_k, q_{k-1}, u_k, \theta) = 0.$$

Since the right discrete force is zero using this approximation method, the derivations and equations in this thesis will not include the right discrete force term for simplicity; however, if a different approximation method is used, this term may need to be added to the derivations.

2.4.2. Kinematic Configuration Variables

In the model used in Chapter 5, kinematic configuration variables are used when simulating and optimizing the system. For certain subsets of a robot model, it may be reasonable to assume a kinematic model where the actuators are strong enough to accurately realize any reasonable trajectory in the configuration space. For the cart-pendulum example, the position of the cart x will be treated as kinematic, i.e., the input at any time t_k will be the position of the cart rather than the force upon the cart. While this separation is not required or essential, it tends to simplify implementation in practice by allowing a higher frequency, low-level controller to provide position control for the kinematic states while running a lower frequency controller on the dynamic states. The derivation of the optimal control algorithm with kinematic configurations and the associated derivatives are available in [**33**].

To implement the kinematic configuration variables, new states are added for the kinematic components. Thus, the state becomes

$$x_k = \begin{bmatrix} q_k \\ \rho_k \\ p_k \\ \nu_k \end{bmatrix},$$

where $\rho_k \in \mathbb{R}^{nk}$ is the set of kinematic configuration variables and $\nu_k \in \mathbb{R}^{nk}$ is the set of kinematic velocities. The discrete Lagrangian and discrete forces are expanded to include these kinematic configurations; however, the solution to the equation remains the same since the variables are predefined as inputs. The discrete Lagrangian and discrete forces

can be written in the following manner to include the kinematic configurations

$$L_{k+1} = L_d(q_k, q_{k+1}, \rho_k, \rho_{k+1}, \theta)$$

$$F^{\pm}_{k+1} = F^{\pm}_d(q_k, q_{k+1}, \rho_k, \rho_{k+1}, u_k, \theta).$$

The input vector is then defined as the force inputs u_k at the current time t_k and the kinematic configuration at the next time step ρ_{k+1} which will be notated as

$$\bar{u}_k = \begin{bmatrix} u_k \\ \\ \rho_{k+1} \end{bmatrix}.$$

With the introduction of all of the concepts presented in this chapter, the information-based optimization algorithms can be developed. Before diving into the derivations and experimental results from the algorithms, the following chapter presents a case-study on numerical methods for biomechanical simulation and the need for trajectory optimization for parameter estimation.

CHAPTER 3

Continuous vs. Structured Simulation Methods

Numerical integration methods are the backbone of almost all simulation and optimal control methods. Physical systems evolve in a continuous time domain; however, the continuous differential equations that specify a mathematical predication of that evolution rarely can be analytically solved in the continuous domain. Numerical methods are used to compute approximate solutions to these differential equations for use in optimal control solutions. It is often the case that a simple numerical method is chosen, such as an RK4 scheme presented in Section 3.1.2; however, there are many non-trivial advantages and disadvantages to different numerical methods.

The information-based optimization methods presented in Chapters 4 and 5 are derived for two different classes of numerical methods. Before presenting those methods, this chapter will provide a case study of the differences between continuous and structured simulation methods. Additionally, this study motivates the necessity for exploration into the question of trajectory optimization for the purposes of parameter estimation. In this thesis, continuous simulation methods refer to algorithms that directly discretize the continuous-time dynamics, typically derived using the Euler-Lagrange equations. Structured methods, such as the variational integrator introduced in Chapter 2, provide a

discrete formulation of the dynamics which are derived specifically for the chosen time-discretization. Continuous methods, which include Euler and Runga-Kutta integrators, are typically simple to implement, and have a relatively low computation cost per iteration. Structured methods can involve a more unintuitive derivation of the discrete dynamics with higher computation per iteration but may have better performance at larger time steps.

While several studies have been presented in the literature highlighting these trade-offs, this study focuses on the simulation of flexion and extension of the elbow in musculoskeletal model of the upper limb. Biomechanical modeling is just one area where continuous-time based integration methods are commonly used, and this work presents the first benchmark comparison to the variational integrator, a structured integration method.

3.1. Musculoskeletal Modeling

Rigid-body modeling has been a popular technique for forward simulation of the dynamics of human movement. The skeletal system is treated as a set of rigid bodies connected by joints with muscles applying moments about each joint. Muscle activation signals in the form of electrical impulses are sent to the muscles from the brain through the nervous system. These signals can be recorded as EMG data using surface or intramuscular collection techniques. A challenge unique to the biomechanical modeling community is the creation of appropriate muscle models to simulate the activation and resultant forces produced by the muscles which act on the rigid-body skeletal model [**36**]. One popular choice among biomechanics researchers is the OpenSim simulation platform [**37**]. This

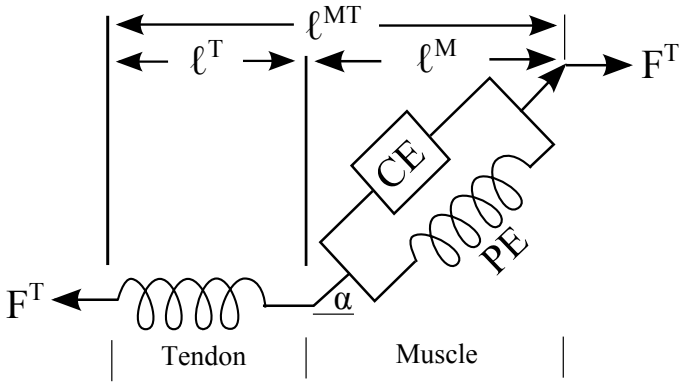


Figure 3.1. Diagram of the Hill muscle model. Adapted from [**36**].

package allows the creation of the skeletal structure as well as choice of muscle model and forcing in the system.

These muscle models are typically derived from the Hill muscle model, shown in Fig. 3.1 which contains three elements: an elastic tendon element, elastic muscle element, and active contractile muscle element. One such model that has been used in the OpenSim platform is Schutte's muscle model based on the three element Hill model [**38**]. In this model, the muscle dynamics are governed by a first order dynamical system given by

$$0 = aF_\ell(\tilde{\ell}^M)F_v(\dot{\tilde{\ell}}^M) + F_{PE}(\tilde{\ell}^M)\cos\alpha - F_T(\tilde{\ell}^T), \tag{3.1}$$

where $\tilde{\ell}^M$ is the normalized muscle length, $\tilde{\ell}^T$ the normalized tendon length, α the pennation angle, F_{PE} the passive force length relation, F_T the tendon force length relation, F_v the active force velocity relation, F_ℓ the active force length relation, a the muscle activation, and F_{max} is the maximum contractile force. The muscle activation signal is also

represented by a first-order dynamical system

$$\dot{a} = \begin{cases} (u-a)/\tau_{deact} & : u \leq a \\ (u-a/\tau_{act} & : u > a \end{cases}$$

where u is the muscle EMG signal and τ are the activation and deactivation constants. The EMG data is preprocessed to provide the inputs for the model. A number of studies have examined EMG processing techniques which will not be discussed in this chapter [**39**–**41**]. Using the Schutte model, the muscle length will serve as a state variable in addition to the generalized coordinate states of the skeletal model. At each step of the simulation, the skeletal and muscle states are simultaneously updated providing new moment predictions from the muscle model.

Muscle pathways are defined by a series of points that are static relative to an assigned bone. Each muscle is fixed at two endpoints and slides freely through additional intermediate points. As the configuration of the skeletal system changes, the muscle length will change based on the sum of the Euclidean distance between each of the points along the pathway. The musculotendons apply forces on the bones at the attachment points, thus affecting the configuration of the skeletal structure. The following section addresses the computation of the muscle moments required for forward simulation of the system dynamics.

3.1.1. Computing Muscle Moments

The muscle dynamics given by (3.1) provide a mapping from activation signals to a change in muscle length. This change in length produces a tension force along the muscle pathway,

producing a configuration dependent moment on the joint. One method of computing the moment is determining the moment arm and direction of the applied force on a bone which will determine the moment on the joint. This method requires a number of geometric calculations which may be computationally expensive.

Alternatively, the method of virtual work provides a cleaner method of calculating the moment. The virtual work of the muscle can be defined by the tension force multiplied by the change in the length of the muscle pathway

$$\partial W_{muscle} = F_T \cdot d\ell^{MT},$$

where ℓ^{MT} is the total muscle-tendon length as defined by the pathway and W_{muscle} is the work done by the muscle. Meanwhile, the work done on a joint, represented by a generalized coordinate, is given by the moment about that joint M_{joint} multiplied by the change in the generalized coordinate

$$\partial W_{joint} = M_{joint} \cdot dq.$$

The work done must be equivalent for either representation, therefore

$$F_T \cdot d\ell^{MT} = M_{joint} \cdot dq$$
$$M_{joint} = F_T \cdot \frac{d\ell^{MT}}{dq}.$$

The derivative of the length with changes in the generalized coordinate can be easily computed from the skeletal geometry. This provides a simpler means of determining the applied torque to the joint at each time step of the simulation. Since the muscloskeletal

system is redundant with several muscle acting on the same joint, different methods can be employed to solve the force sharing problem [**42**]. The simulation techniques provided in this chapter result in the minimum total muscle force solution.

3.1.2. Forward simulation with explicit integration

With the specification of the muscle model, skeletal geometry, and inertias, a forward simulation of the upper-limb can be performed from recorded EMG signals. The equations of motion of the skeletal structure can be solved using the Euler-Lagrange equations,

$$0 = \frac{\partial L_c}{\partial q} - \frac{d}{dt}\left(\frac{\partial L_c}{\partial \dot{q}}\right),$$

where L_c is a standard Lagrangian of kinetic minus potential energies.

The generalized coordinates of the skeletal system q are then added to the muscle length states from (3.1). These equations can be written in state space form

$$\dot{x}(t, q, \dot{q}, \ell^M, a) = f(t, x(t, q, \dot{q}, \ell^M, a)), \tag{3.2}$$

where

$$x(t, q, \dot{q}, \ell^M, a) = \begin{bmatrix} q(t, \ell^M) \\ \dot{q}(t, \ell^M) \\ \ell^M(q, \dot{q}, a) \end{bmatrix}.$$

A standard method of forward simulation is the integration of the equations of motion from (3.2) using an explicit Runga-Kutta scheme. Runga-Kutta integrators provide a method for time-based discretization of the differential equations governing the dynamics. The explicit integrator only relies on the current and past states to compute the future

state while an implicit method is a function of the future state as well. Explicit integrators are commonly used for their ease of implementation and speed. In the simulation results, we will use a RK4 integrator which is a 4th order explicit Runga-Kutta scheme. The RK4 update formula is given by the following,

$$
\begin{aligned}
k_1 &= f\left(t_n, x_n\right) \Delta t \\
k_2 &= f\left(t_n + \frac{1}{2}h, x_n + \frac{1}{2}k_1\right) \Delta t \\
k_3 &= f\left(t_n + \frac{1}{2}h, x_n + \frac{1}{2}k_2\right) \Delta t \\
k_4 &= f\left(t_n + h, x_n + k_3\right) \Delta t \\
x_{n+1} &= x_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4.
\end{aligned}
$$

Using the EMG recordings, the activation inputs can be determined and the forward simulation can be carried performed. The following section describes the particular model for the elbow that is used in the simulation results.

3.1.3. Elbow Model

For the integrator comparison and simulation results to follow, the focus will be on the motion of the elbow. From the complete upper limb model used by Saul et al. [**43**], eight muscles spanning the elbow joint are used. The model also includes the full geometry and inertias of the upper limb. A rendering of the model using the OpenSim software platform can be seen in Fig. 3.2.

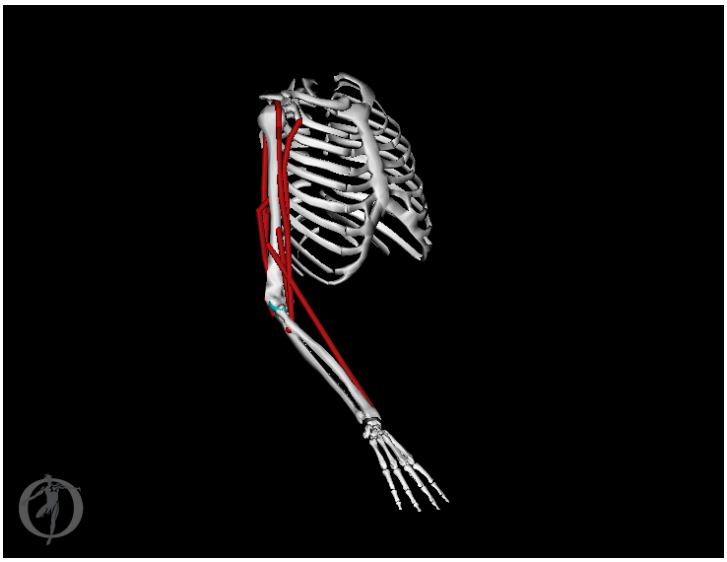


Figure 3.2. An OpenSim rendering of the upper-limb biomechanical model with eight muscles.

To simplify the model for the purposes of this comparison, only a single degree of freedom in the elbow joint is used. The remaining joints are locked to a specified position with all fingers in extension. The parameters required for the simulation of these muscles are shown in Table 3.1 which come from a subset of those presented in [**43**]. Additionally, cylindrical wrapping surfaces are used in the model for the triceps, biceps, and brachioradialis muscles. During extension and flexion of the elbow, as these muscles come in contact with the wrapping surface near the elbow joint, the muscle paths will wrap around the surface. The derivation of the equations representing the wrapping surface are presented in Appendix A.

Default normalized force-length curves for the muscle and force-strain curves for the tendon are used. The Schutte model also defines a damping parameter for the muscle model which is set to 0.017 N/m/s.

Table 3.1. Muscle Parameters for Elbow Simulation

Muscle	Abbreviation	Optimal fiber length (cm)	Peak force (N)	Tendon slack length (cm)	Pennation angle (°)
Triceps					
Long	TRIlong	13.4	771.8	14.3	12
Lateral	TRIlat	11.4	717.5	9.8	9
Medial	TRImed	11.4	717.5	9.1	9
Anconeus	ANC	2.7	283.2	1.8	0
Biceps					
Long	BIClong	11.6	525.1	27.8	0
Short	BICshort	13.2	316.8	20.0	0
Brachialis	BRA	8.6	1177.4	5.4	0
Brachioradialis	BRD	17.3	276.0	13.3	0

3.2. Variational Integrators and the Muscle Model

While the variational integrator has been used in the simulation of a number of rigid body systems, both conservative and forced, it has yet to be applied to a biomechanical model. One factor limiting direct use of the integrator is the construction of the muscle model. Since the muscle model is not an inertial system (i.e., the muscle typically is modeled without mass) a hybrid approach needs to be taken in order to incorporate the variational approach into the biomechanical model. The method that is proposed in this section is the use of the variational integrator natively on the skeletal states, which include inertial elements, while the muscle model is solved simultaneously using an implicit Euler integration scheme.

As introduced in Section 2.4.1, the variational integrator is an implicit integrator which depends on the current and future state to simulate the discrete system. The integrator is derived from the notion of generalized momentum equations which are a natural choice for physical inertial systems; however, the muscle model lacks this variational principle

of generalized momentum in its update equations. Therefore, since an implicit Newton-Raphson solver is already in place to solve the variational integrator system, the natural complement is the implicit Euler form to update the single state muscle model.

The implicit Euler scheme has also shown promise in the simulation of biomechanical systems. A study by van den Bogert et al. used a first-order Rosenbrock method to simulate both the rigid body dynamics and muscle model of both upper and lower limb models [**44**]. The Rosenbrock method is equivalent to a midpoint implicit Euler method, taking only one iteration of the Newton-Raphson solver per step. Results indicate good performance from this method with improvements in the computational speed by orders of magnitude. The addition of the variational integrator should improve the stability of the results while maintaining the computational performance of the simulation.

3.2.1. Setting up the Variational Integrator and Muscle Model

The generalized coordinates for the skeletal system will be given by q with velocities $\dot{q}$, and the muscle states are notated as ℓ^M. The equations for the variational integrator are derived using the discrete Lagrangian as indicated in Section 2.4.1. Using the midpoint VI formulation, the forced DEL equations are given by

$$D_2L_d(q_{k-1}, q_k) + D_1L_d(q_k, q_{k+1}) + F_d^-(q_k, q_{k+1}, u_k) = 0, \tag{3.3}$$

since the right discrete force is equal to zero using this discretization choice. The left discrete force is then given by

$$F_d^-(q_k, q_{k+1}, u_k) = f_c\left(\frac{q_{k+1}+q_k}{2}, \frac{q_{k+1}-q_k}{\Delta t}, u_k\right)\Delta t,$$

Algorithm 3.1 Newton-Raphson Iterator

Initialize $(q_{k+1}, \ell^M_{k+1}) = (q_k, \ell^M_k)$, tolerance ϵ
while $f(q_{k+1}, \ell^M_{k+1}) > \epsilon$ **do**
$(q_{k+1}, \ell^M_{k+1}) = (q_{k+1}, \ell^M_{k+1}) - Df^{-1}(q_{k+1}, \ell^M_{k+1})$
return (q_{k+1}, ℓ^M_{k+1})

where f_c can be computed from the tendon passive force equation. Since the tendon length, ℓ_T is the difference between the total musculotendon path length and muscle length, the force equation can be written as

$$f_c = -F_{max} F_T(\tilde{\ell}^{MT} - \tilde{\ell}^M) \cdot D\ell^{MT}.$$

Taking the Schutte equation and using the midpoint approximation results in the following implicit Euler form of the muscle update,

$$\begin{aligned} 0 =& aF_\ell \left(\frac{\tilde{\ell}^M_{k+1} + \tilde{\ell}^M_k}{2} \right) F_v \left(\frac{\tilde{\ell}^M_{k+1} - \tilde{\ell}^M_k}{\Delta t} \right) \\ &+ F_{PE} \left(\frac{\tilde{\ell}^M_{k+1} + \tilde{\ell}^M_k}{2} \right) \cos\alpha - F_T \left(\frac{\tilde{\ell}^T_{k+1} + \tilde{\ell}^T_k}{2} \right). \end{aligned} \tag{3.4}$$

Taking (3.3) and (3.4) provides a system of update equations that can be simultaneously solved using the Newton-Raphson iterator

$$f\left(\begin{bmatrix} q \\ \ell^M \end{bmatrix} \right) = \begin{bmatrix} D_2 L_d(q_{k-1}, q_k) + D_1 L_d(q_k, q_{k+1}) + F_d^-(q_k, q_{k+1}, u_k) \\ \text{Equation 3.4} \end{bmatrix}.$$

The Newton Raphson iterator is shown in Algorithm 3.1. As shown in the algorithm, the derivative of f is needed at each iteration to converge on the future state. Therefore, derivatives of both the skeletal equations and muscle equations are needed. The derivative

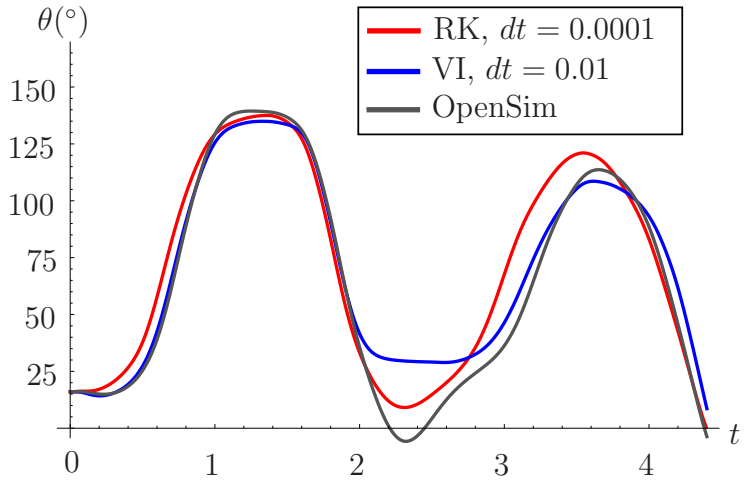


Figure 3.3. Comparison of the Runga-Kutta, Opensim and Variational integrators for a 4.5 s simulation of the elbow motion from EMG activations.

of the muscle equations from (3.4) are

$$a\left[\frac{1}{2\ell^0_{ml}}DF_\ell\left(\tilde{\ell}^M_{k+1},\tilde{\ell}^M_k\right)F_v\left(\tilde{\ell}^M_{k+1},\tilde{\ell}^M_k\right)+\frac{1}{\dot{\ell}^0_M\Delta t}F_\ell\left(\tilde{\ell}^M_{k+1},\tilde{\ell}^M_k\right)DF_v\left(\tilde{\ell}^M_{k+1},\tilde{\ell}^M_k\right)\right]$$

$$+\frac{1}{2\ell^0_{ml}}DF_{PE}\left(\tilde{\ell}^M_{k+1},\tilde{\ell}^M_k\right)\cos\alpha+\frac{\cos\alpha}{2\ell^0_T}DF_T\left(\tilde{\ell}^M_{k+1},\tilde{\ell}^M_k\right), \tag{3.5}$$

with the complete derivative of f given by

$$f\left(\begin{bmatrix} q \\ \ell^M \end{bmatrix}\right)=\begin{bmatrix} D_2D_1L_d(q_k,q_{k+1})+D_2F^-_d(q_k,q_{k+1},u_k) \\ \text{Equation 3.5} \end{bmatrix}.$$

After setting up the iterator with the activation dynamics, a, the muscle and skeletal states can be computed. The following section presents a comparison of this variational integration based simulation technique with an explicit Runga-Kutta scheme.

3.3. Simulation Results

Two algorithms are simulated and compared for the upper elbow model and also compared against the OpenSim predicted trajectory. The first is an explicit Runga-Kutta fourth order integrator, and the second is the variational integrator with a midpoint Euler muscle model. Simulations are compared qualitatively at varying time steps for stability and accuracy of the predicted motion.

Figure 3.3 shows the results of the three simulations at stable time steps. Given the same EMG excitation signals, the motion of the elbow is similar but not exact. This difference is likely due to a combination of integration scheme and recruitment curve specification. For the variational integrator and RK4 integrator, the recruitment curves are identical, leading to the integrator as the primary cause of the curve differences. Due to the implicit nature of the variational integrator, the stiff muscle model is exactly satisfied at every time step. Explicit integrators, such as the Runga-Kutta integrator do not explicitly enforce the length relationship between the muscle and tendon, likely resulting in the difference in simulated motion. Without the use of passive joint limit forces, the OpenSim model also results in non-physiological motion of the elbow with a negative angle. The Runga-Kutta integrator also results in much higher extension which may be attributed to the integrator itself.

While the motion is similar at stable time steps, the two integrators have drastically different properties when varying the time steps. Figure 3.4 shows the RK4 simulation at three different time steps. From the qualitative plot, the integration is stable below time steps of 0.0005 sec; however, at 0.0008 sec, the simulation clearly becomes unstable.

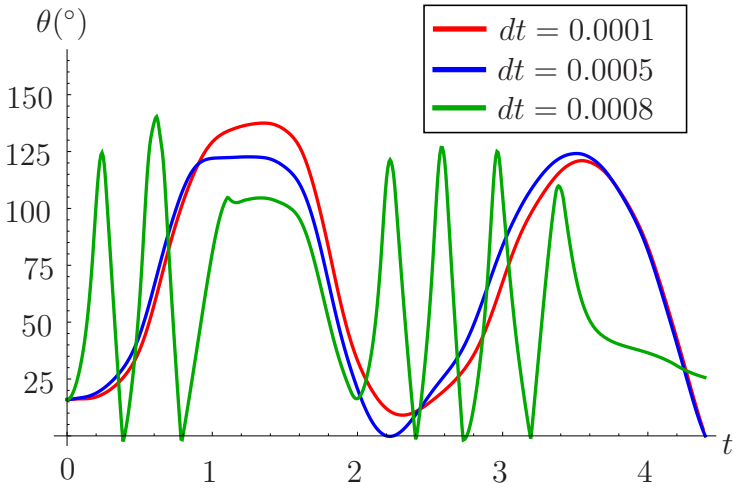


Figure 3.4. Comparison of the simulated elbow trajectories using the Runga-Kutta integrator at different time steps.

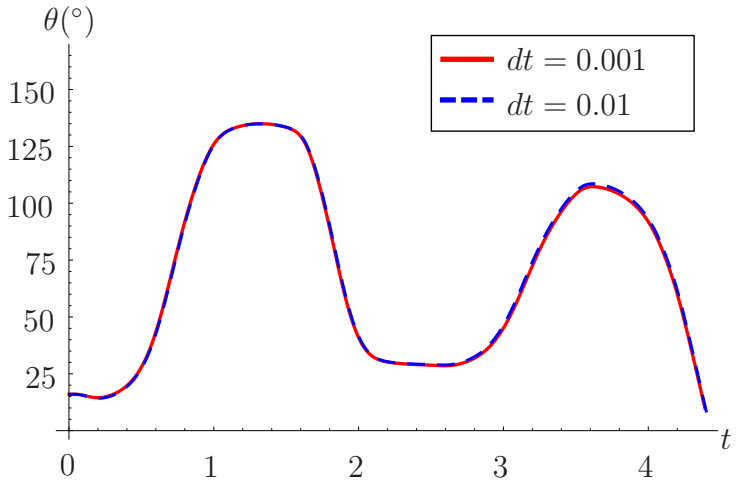


Figure 3.5. Comparison of the simulated elbow trajectories using the Variational integrator at different time steps.

The RK4 results are compared against the variational integrator at two different time steps. Figure 3.5 shows the VI at two time steps, both larger than the 0.0008 sec which proved to cause an instability in the RK4 integrator. At two orders of magnitude longer, 0.01 sec, the variational integrator remains stable. The integrators also fail in two different manners: At longer time steps, the Runga-Kutta integrator becomes unstable. The

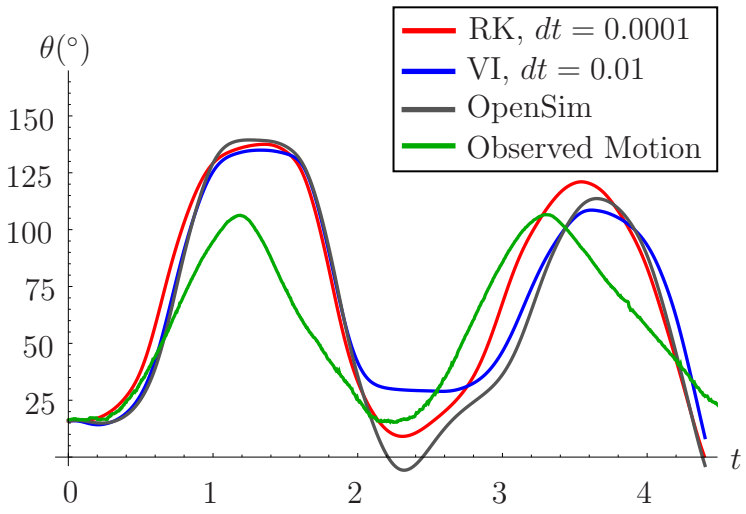


Figure 3.6. Comparison of the simulated trajectories using each integrator to the recorded elbow motion.

variational integrator will instead fail to solve the Newton-Raphson iterations, resulting in no simulation solution. In this respect, the VI provides a better failure mode as the simulation only drastically diverges very close to where no solution exists. Therefore, it is less likely that an errant motion will be returned by the variation integrator compared to the RK4 scheme at large time steps.

3.3.1. Parametric Model Error and Experimental Identification

The EMG data used for the simulation tests was obtained simultaneously with actual motion capture of the elbow trajectory. While the simulation results highlight the differences between numerical methods, when the resulting trajectories are compared against the actual motion of the elbow, it becomes clear that significant differences remain between all of the simulations and the observed trajectory of the elbow. Figure 3.6 shows the same three simulated results with the addition of the actual elbow motion.

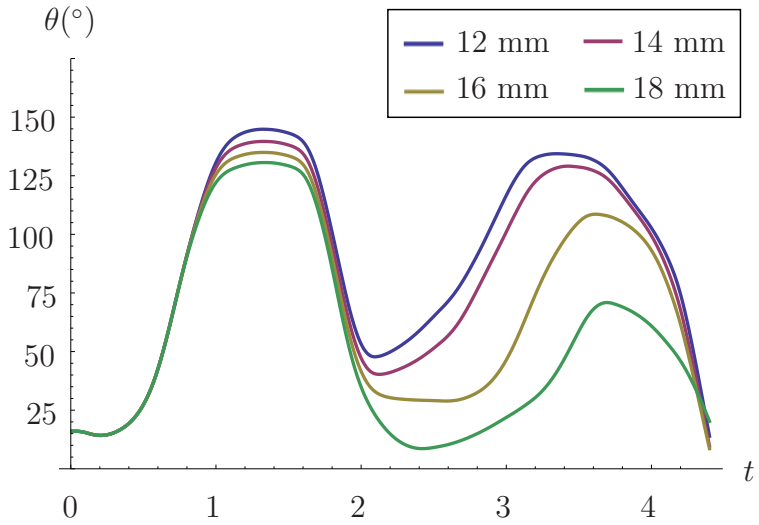


Figure 3.7. Simulated elbow trajectories for four different values of the cylindrical wrapping surface on the triceps muscles.

As highlighted in the introduction of this thesis, three major causes of error can include numerical error, parametric error, and error from unmodeled dynamics. Since the simulation results focused primarily on reducing numerical error in the simulation, there is clearly a large amount of parametric and unmodeled error. It should not be completely surprising to have such large errors due to the nature of biomechanical models. As opposed to machines created to satisfy a set of specifications and a predetermined set of models, the biomechanical models must be obtained primarily using empirical formulations, especially when modeling the forces produced by the muscles.

Figure 3.7 highlights the sensitivity of the elbow trajectory output to changes in one important parameter which is the tricips muscle wrapping radius. This wrapping radius affects the moment that is applied on the elbow joint given the force produced by the muscle-tendon. Even small changes on the order of millimeters have a significant effect

on the elbow trajectory. These sensitivities have been observed in several studies as well as variations in the muscle models used [**45**, **46**]

In order to refine estimates of parameters such as the wrapping radius within the system model, it is desirable to be able to perform experiments *in vivo* in the case of biomechanics or on the functioning machine otherwise. This methodology has been used to attempt identification of geometric parameters as well as general models [**47**–**49**]. However, blindly creating experimental trajectories for estimation in these systems may not result in better estimates of the model parameters if the parameters are not observable, or the sensitivity to the output is very low. The question of how to create these optimal trajectories for physical systems, while managing computational complexity and integrator error serves as the motivating force behind the contributions of this work. The following chapter presents a step toward this goal with a continuous-time algorithm for information and conditioning optimization.

CHAPTER 4

Continuous-time Information Optimization

The trajectory of a dynamical system can significantly impact the experimental identification of parameters both in the precision of an estimate and the ability to estimate multiple parameters. As trajectories for physical systems evolve in continuous time, a natural goal is to compute optimal trajectories in the continuous time domain while satisfying the constraints imposed by the system dynamics. Discretization of the dynamics *a priori* has several problems. First and foremost, an arbitrary choice about the discretization needs to be made. Secondly, adaptive time-stepping methods cannot be used, and a discretization appropriate for the initial trajectory cannot be expected to be appropriate for the final trajectory. Lastly, discretization can lead to high-dimensional constrained optimizations (dimensions of 10^7 to 10^{12} are common in practical problems) that are impractical to solve numerically. This chapter addresses the trajectory optimization problem using a continuous-time algorithm that has been modified to improve upon information metrics derived from Fisher information.

Two approaches in the cost metric are derived and validated in the continuous-time domain. In both cases, the projection-based trajectory optimization method introduced in Section 2.3.1 is used as the optimization mechanism to compute updates to the trajectory.

Section 4.2 presents the first approach which aims to maximize the Fisher information using E-optimality conditions. The experimental setup used to validate the method is discussed in Section 4.3 with the simulated and experimental results presented in Section 4.4.

In the second approach, the conditioning of the estimation problem itself is considered. Section 4.5 presents the conditioning optimization algorithm which improves the condition number of the Hessian matrix of the estimation problem to improve the estimator rate of convergence given multiple unknown parameters. For the Iterated-Least Squares estimator, the Hessian becomes equivalent to the Fisher information matrix given the Gaussian measurement noise assumed in the problem formulation. Therefore, the problem becomes an extension of the Fisher information maximization algorithm using a ratio of the eigenvalues as the cost metric. The algorithm is validated with another experiment with results provided in Section 4.7.

4.1. Related Work

Since the design of an experimental trajectory has a wide range of potential uses, there have been a number of contributions to the area from different fields. A large amount of literature on optimal experimental design exists in the fields of biology [**50**–**52**], chemistry [**53**], aerospace [**54**], and systems [**55**–**58**]. Many of these results focus on particular applications to experiments specific to their respective fields; however, the underlying principles of information theory remain the same.

Metrics on the Fisher information are used as a cost function in many optimization problems including work by Swevers on "exciting" trajectories [**59**]. This work, as well as

related works [**10**, **60**], synthesize trajectories for nonlinear systems that can be recast as linear systems with respect to the parameters. Further research has resulted in optimal design methods for general nonlinear systems. In work by Emery [**61**], least-squares and maximum-likelihood estimation techniques are combined with Fisher information to optimize the experimental trajectories. In this case and a number of others, the dynamics are solved as a discretized, constrained optimization problem [**62**, **63**].

To avoid discretization of continuous dynamics, a class of methods has been developed that relies on sets of basis functions to synthesize optimal controls for the system [**64**–**66**]. These methods allow one to optimize over a cost index, such as the condition number, using a finite dimensional optimization method over a fixed set of basis functions. While this allows the trajectory to be expressed in the continuous time domain, the space of allowable control signals is still finite. One example of the basis function set includes the Fourier basis, which is used create a class of trajectories over which the optimization can be performed [**67**].

A wide variety of work has also been performed in the area of experiment design concerning estimators and the identifiability of parameters [**4**, **68**–**73**]. In the robotics field specifically, trajectory optimization and parameter identification algorithms have been developed for special classes of robotic systems. For serial robot arms and similarly connected systems, chain-based techniques and linear separation of parameters can be used [**74**, **75**]. Techniques have also been adapted for parallel robots and manipulators [**76**–**78**]. While these techniques perform well for the intended class of robots, we seek an algorithm that has the ability to work on general nonlinear systems, only requiring differentiability of the dynamics and some form of control authority.

4.2. Trajectory Optimization Maximizing Fisher Information

The trajectory optimization algorithm used in the results to follow uses the projection-based optimal trajectory tracking algorithm defined in Section 2.3.1 with an extension to include Fisher information as an objective [**30**, **32**]. This algorithm is formulated in continuous-time to allow the time discretization of the dynamics to be easily decoupled from the measurement time points as well as allow for the use of any numerical integration method. As discussed in Chapter 3, the continuous algorithm may not result in the most stable method; however, it provides the theoretical basis for extension into the structured domain in Chapter 5. Therefore for the continuous derivation, the objective including any costs on the Fisher information must be formulated in continuous time.

Section 2.1.2 introduced the quantity of Fisher information, quantifying the amount of information a set of observations contains about a set of unknown parameters [**79**]. For dynamic systems, a set of measurements taken during the execution of a trajectory provides information about model parameters. The effectiveness of the parameter estimation techniques outlined in Section 2.2 is related to the Fisher information matrix (FIM) through the Cramer-Rao bound described in Section 2.1.3. In practice, increasing the Fisher information of the system tends to increase the maximum precision that can be obtained by an estimation algorithm; however this is not explicitly guaranteed by (2.1).

The method presented in this chapter will assume that the measurement noise of the system is normally distributed with zero process noise, and the FIM for the system is given by,

$$I(\theta) = \sum_{i}^{t_f/dt} \Gamma_\theta(t_i)^T \cdot \Sigma^{-1} \cdot \Gamma_\theta(t_i), \tag{4.1}$$

where $\Gamma_\theta(t_i)$ is the parametric sensitivity of the system output as defined by (2.6), and Σ is the known sensor covariance matrix. Note that this expression has been simplified from the general form of Fisher information given the assumptions stated above. Details of the simplification can be found in [**23**].

To use the FIM as a metric for trajectory optimization, $I(\theta) \in \mathbb{R}^{p \times p}$ must be mapped to a scalar value. There is a significant amount of literature on different types of mapping choices [**50**, **61**, **62**]; however, this work will restrict itself to the design choice of E-optimality. The choice of E-optimal design is used to improve the worst-case variances of the parameter set by maximizing the minimum eigenvalue of the FIM. Thus, the algorithm creates a variation on the trajectory to proportionally improve the information acquired. Other experimental design approaches such as A and D-optimality conditions, which provide different goals for the desired information acquisition, can be realized with modifications to the objective function. The following section presents the derivation of the E-optimal objective function and presents the iterative algorithm.

4.2.1. Objective Function

To define an objective function dependent on the FIM on a continuous-time domain, the maximization of the information matrix needs to be cast from a finite set of discrete measurements into an appropriate continuous analogue. To satisfy this condition, the information equation (4.1) will be written as

$$\tilde{I}(\theta) = \int_{t_0}^{t_f} \Gamma_\theta(t)^T \cdot \Sigma^{-1} \cdot \Gamma_\theta(t)\, dt. \tag{4.2}$$

Assuming that observations are taken regularly along the entire trajectory, the optimal trajectory $x^*(t)$ maximizing the eigenvalues of the continuous $\tilde{I}(\theta)$ will approximately optimize the eigenvalues of the sampled $I(\theta)$. As the sampling rate increases, the values of $\tilde{I}(\theta)$ and $I(\theta)$ will converge.

If measurements only occur along certain portions of the trajectory with predetermined times, a weighting function can be added to $\tilde{I}(\theta)$ to ensure that sensitivity is maximized specifically in the sampled areas of the trajectory. However, if the observation time is not predetermined, the sensitivity along the entire trajectory will be maximized using $\tilde{I}(\theta)$.

The optimization objective function will therefore be given by

$$J = \frac{Q_p}{\lambda_{min}} + \frac{1}{2}\int_{t_0}^{t_f} (x(t) - x_d(t))^T \cdot Q_\tau \cdot (x(t) - x_d(t)) + u(t)^T \cdot R_\tau \cdot u(t)\ dt, \tag{4.3}$$

where λ_{min} is the minimum eigenvalue of $\tilde{I}(\theta)$, Q_p is the information weight, $x_d(t)$ is a reference trajectory, Q_τ is a trajectory tracking weighting matrix, and R_τ is a control effort weighting matrix. The weights must be chosen such that $Q_p \geq 0$, Q_τ is positive semi-definite, and R_τ is positive definite. Although the cost function appears to include a terminal condition on the eigenvalues of $\tilde{I}(\theta)$, the eigenvalues themselves are functions of each point along the trajectory. This property will allow the apparent non-Bolza form of the optimal control problem to be cast into a Bolza form for the calculation of the descent direction. This Bolza form of the problem only holds locally for perturbations to the trajectory required for the iterative LQR optimization.

The various weights allow for design choices in the optimal trajectory that is obtained. The requirements of positive definiteness and positive semi-definiteness of the weighting

matrices are necessary to maintain a locally convex optimization problem including the fact that $\lambda \geq 0$ [**30**]. Increasing the control weight will result in less aggressive trajectories, generally decreasing the obtained information. Using a reference trajectory allows for an optimal solution that remains in the neighborhood of a known trajectory.

While the cost function is generally well defined, it does become ill-conditioned and singular in the case of a zero eigenvalue. This will occur if the initially chosen trajectory yields no information about at least one unknown parameter. For a dynamic system, this may occur if a stationary trajectory is chosen in which case, perturbing the initial trajectory of the system using a heuristic method will likely provide a sufficient initial trajectory for the algorithm. If a perturbation fails to provide any information, the system geometry or sensor configuration may prohibit the experiment from resolving the parameters simultaneously for any trajectory. In this case, sensors may need to be added, or unknown parameter removed to create a well-posed experiment.

4.2.2. Extended Dynamic Constraints

The optimal control algorithm was previously formulated for trajectory tracking problems where the objective function is explicitly a function of the system states [**30**]. However, given (4.3), the cost also depends on $\psi(t) = \frac{d}{d\theta}x(t)$. Since the objective is to minimize a norm that includes $\psi(t)$, which depends nonlinearly on $x(t)$, $\psi(t)$ is treated as an additional state. Appending $\psi(t)$ to the state vector as an additional dynamic constraint allows for variations in $\psi(t)$ in the optimization algorithm. This technique has been used in the optimization of aeronautical trajectories through extending the state with similar sensitivity terms to improve the covariance of the planned trajectory [**80**–**82**]. Work

by Ansari [**83**] uses the state extension technique for the related problem of sensitivity minimization. For convenience, the extended state will be defined in this thesis by $\bar{x}(t) = (x(t), \psi(t))$, and $\eta(t) = (\bar{x}(t), u(t))$ defines a curve which satisfies the nonlinear system dynamics.

4.2.3. Projection Operator

While the minimization of (4.3) is subject to the dynamic constraints given by (2.2) and (2.7), the optimization algorithm defined in Section 2.3.1 linearizes the constraints to produce an LQ problem. This problem is then iteratively solved to produce a descent direction followed by a projection that maps the infeasible trajectory, formed by the sum of the current iterate and the descent direction, onto the dynamic constraints as detailed in [**32**].

To incorporate the extended dynamics constraints, the projection operator in (2.17) is reformulated with the sensitivity constraint and extended dynamics

$$P(\xi(t)) : \begin{cases} u(t) = \mu(t) + K(t)(\bar{\alpha}(t) - \bar{x}(t)) \\ \dot{x}(t) = f(x(t), u(t)) \\ \dot{\psi}(t) = D_x f(x, u, \theta)^T \psi(t) + D_\theta f(x, u, \theta)^T. \end{cases}$$

This projection formulation allows unconstrained variations of the trajectory in both the nominal state and sensitivity state to be calculated with the solution then projected to a feasible trajectory at each iteration of the optimization algorithm.

4.2.4. Optimization Routine

Algorithm 2.2 defines the iterative method using a gradient descent approach to solve the optimization problem. Each iteration requires a descent direction $\zeta_k(t) = (\bar{z}, v)$ to be computed from (2.17), only now the linearized dynamics include the extended state. Thus (2.18) becomes

$$\dot{\bar{z}} = \bar{A}(t)\bar{z} + B(t)v,$$

where the descent direction includes the perturbation to the extended state. As with the original form of the trajectory optimization algorithm, the descent direction can be solved solved as a time-varying LQ problem. The detailed calculation of the descent direction for this extension of the algorithm is provided in Appendix B.1.

Given the descent direction ζ_k, a backtracking line-search of the projection, $P(\eta_k(t) + \gamma_k\zeta_k)$, provides a feasible trajectory assuming the step size γ_k satisfies the Armijo sufficient decrease condition [**27**]. Iterations upon the feasible trajectories continue until a given termination criteria is achieved.

4.3. Experiment Setup for Fisher Information Maximization

To demonstrate the optimization of Fisher information for a dynamic system, a simulation and experimental test of a 2-link cart-pendulum system is considered. The system has three configuration variables, $q = (x(t), \phi_1(t), \phi_2(t))$, where $x(t)$ is the horizontal displacement of the cart, and $\phi_i(t)$ is the rotational angle of each link as seen in Fig. 4.1.

The control input to the system is the acceleration of the cart, given by u. The cart can accelerate in either direction with positive acceleration to the right. Rotational friction is modeled at each pendulum joint, but the joints remain unactuated. The goal of the

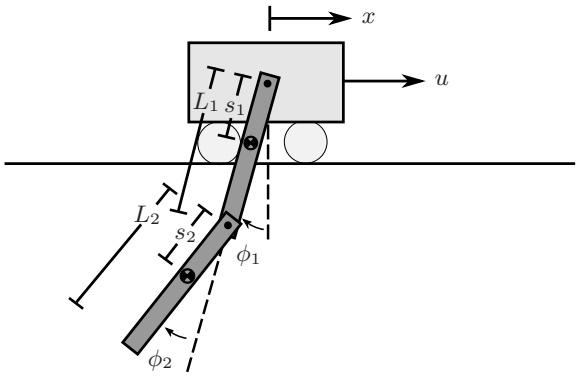


Figure 4.1. Cart-pendulum system

optimization algorithm is to accurately estimate the mass of the top pendulum link and the damping coefficient of the joints.

4.3.1. Experimental Testbed Setup

4.3.1.1. Robotic System. The experimental setup shown in Fig. 4.2 consists of a differential drive mobile robot with magnetic wheels moving in a plane. The pivot point for the first link of the double pendulum is provided by mounts on the robot. A plastic membrane provides the robot's driving surface. The membrane is tensioned in all directions within an aluminum frame mounted parallel to the ground at a height of approximately 2.5 meters. An unpowered, magnetic idler mechanism is placed on the top side of the plastic to provide adherence for the robot. The idler has the same geometric footprint as the robot, but its magnetic wheels have opposite polarity.

Two 24 V DC motors drive the robot's magnetic wheels. PID loops running at 1500 Hz close the loop around motor velocity using optical encoders for feedback. Two 12 V lithium iron phosphate batteries provide all required power. Desired velocity commands are sent to the robot wirelessly using Digi XBee® modules at 50 Hz, and a 32-bit Microchip PIC

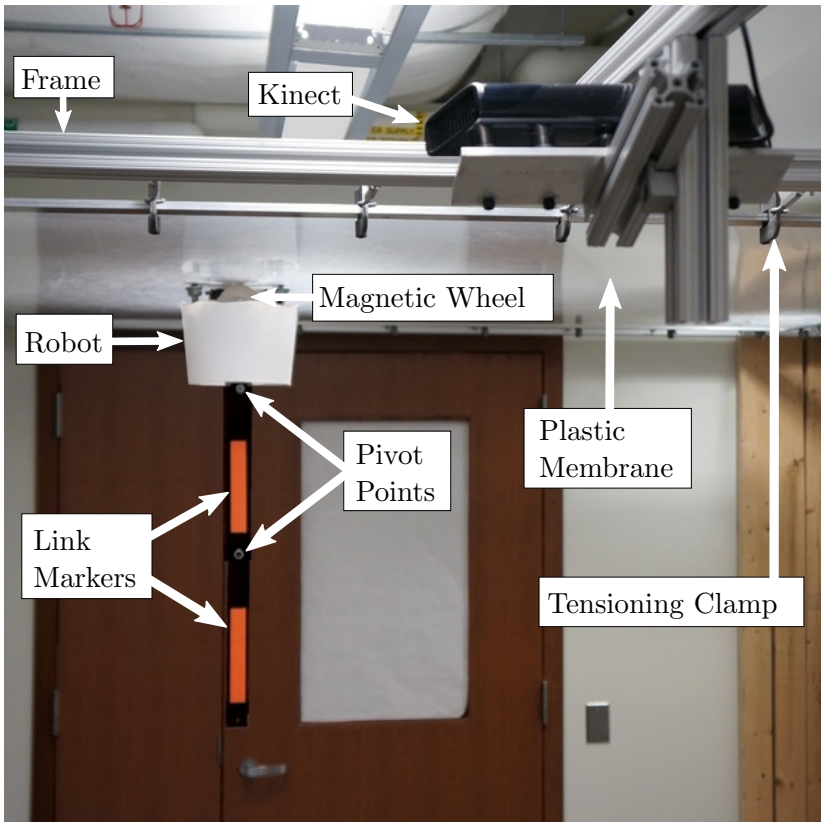


Figure 4.2. Image showing the experimental setup with annotations of the important components provided.

microprocessor handles all on-board processing, motor control, and communication. The motors are powerful relative to the inertias of the robot, the double-pendulum payload, and the wheels. Thus, they are capable of accurately tracking aggressive trajectories up to a maximum rotational velocity. Since the input to the 2-link cart-pendulum system is the direct acceleration of the cart, the PID loops on the motor velocity allow the robot to accurately track a given velocity profile. The Robot Operating System (ROS) is used for visualization, interpreting and transmitting desired trajectories and for processing and recording all experimental data [**84**].

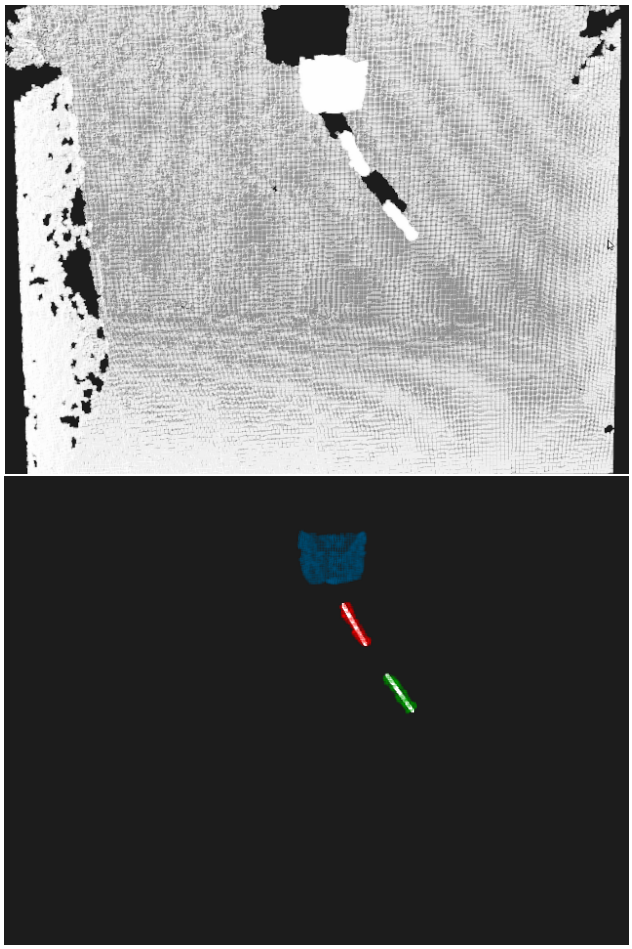


Figure 4.3. Illustration of the image processing performed on the point cloud provided by the Microsoft Kinect®. The top image shows the raw point cloud, and the bottom shows the data after all processing; different colors are assigned to each cluster, and white lines are displayed to show the lines produced by the sample consensus filter.

4.3.1.2. Tracking System. A Microsoft Kinect® is employed to track the system and obtain experimental measurements. The Kinect provides uncolored point clouds at 30 Hz as seen in Fig. 4.3, and the Point Cloud Library (PCL) is used for data processing [**85**]. Each raw point cloud is first downsampled, and then pass-through filters eliminate points that lie outside of a predetermined bounding box of expected system configurations. A Euclidean cluster extraction algorithm is then used to extract three separate clouds –

one for the robot and one for each of the links. The pendulum links are made from an acrylic that is transparent to the Kinect; opaque markers adhered to each of the links provide a visible surface. The markers have a gap between them ensuring that the software uniquely detects the three clusters. The coefficients of the line equations along the axis of each link are then extracted using a sample consensus segmentation filter on the clusters representing the links. Once the coefficients for these lines are known, simple trigonometry allows calculation of the desired link angles, ϕ_1 and ϕ_2.

Since the Kinect will provide measurements of the absolute rotation angle of each pendulum link, it is important to estimate the variance of the measurements for use in the optimization algorithm. To measure the covariance of the angles computed from the point cloud, measurements were recorded for 30 seconds at sensor's fixed frequency of 30 Hz with the double pendulum hanging in a stationary position. Although the variance of the measurements will have some dependence on the link velocity and configuration, we assume independence for the purpose of this experiment. The angles calculated from the point cloud are assumed to be independent for the two links, so the off-diagonal elements are set to zero, yielding a covariance matrix given by

$$\Sigma = \begin{bmatrix} 1.12 \times 10^{-4} & 0 \\ 0 & 4.79 \cdot 10^{-4} \end{bmatrix} \text{rad}^2. \tag{4.4}$$

4.3.2. System Model

A mathematical model of the system is created by deriving standard rigid body dynamics equations. A Lagrangian, $L_c = T - V$ is constructed where the kinetic energy of the system T is given by $T = \frac{1}{2} \sum_{i=1}^{2} m_i \nu_i(x, \phi_1, \phi_2)^2 + \mathcal{I}_i \omega_i(x, \phi_1, \phi_2)^2$, where ν_i is the translational

velocity, and ω_i is the rotational velocity of each link. The potential energy V is a function of the height z_i of each link's center of mass, $V = \sum_{i=1}^{2} m_i g z_i(x, \phi_1, \phi_2)$ given the gravitational constant g. For each link, the moment of inertia $\mathcal{I}_i$ is defined as a function of the mass and measured center of mass location. The center of mass on the link is also used to define the potential energy of the link in the Lagrangian.

Using the Euler-Lagrange equation,

$$0 = \frac{\partial L_c}{\partial q} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right),$$

the full equations of motion can be solved in terms of $\ddot{q}(t)$ (keeping in mind that $\ddot{x} = u$ by assumption).

The Kinect measures the absolute angle y_i of each link i which provides two measurement outputs, $y(t) = [y_1(t), y_2(t)]$. Since the system states define the angle of link 2 relative to link 1, the output function is

$$g(x(t), u(t), \theta) = \begin{bmatrix} \phi_1(t) \\ \\ \phi_1(t) + \phi_2(t) \end{bmatrix}.$$

4.3.3. System Parameter Selection

We have not yet addressed the choice of which parameters to identify for a given system. Choosing the set of parameters will depend on the context of the estimation needs as well as concerns with identifiability. In certain situations, some parameters may not be identifiable regardless of trajectory selection. For example, given the 2-link cart-pendulum system with no damping in the joints, estimating both link masses independently is not possible. To illustrate this, Fig. 4.4 shows a contour plot of β, the parameter estimator

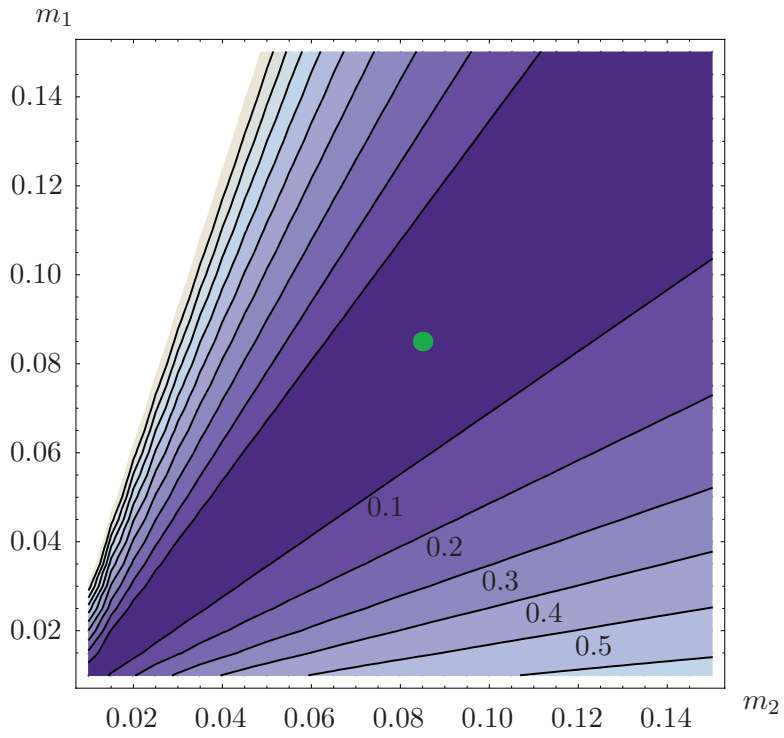


Figure 4.4. Contour plot of the unidentifiable system. The deterministic value of the parameter is shown by the green dot in the contour plot with isolines indicating the estimator cost, $\beta(\theta)$. The straight lines indicate a poorly conditioned estimation problem which results in non-identifiable parameters.

cost, vs. changes in each mass value. The straight isolines in the plot indicate that the parameters are coupled through a constant scaling factor resulting in no unique set of mass parameters that minimize the cost. Rather, the estimator cost can be minimized with an infinite linear combination of the mass parameters, therefore, the specific parameter set is non-identifiable by the estimator.

The non-identifiability condition can be tested for a given trajectory by examining the eigenvalues of the Fisher information matrix. If and only if a zero eigenvalue exists, neither parameter can be uniquely identified. If a zero eigenvalue exists for the trajectory,

Table 4.1. Measured System Parameters

Parameter	Link 1 Value	Link 2 Value
Mass, m	Estimate	0.0847 kg
Link Length, L	0.305 m	0.305 m
Link Width, w	0.0445 m	0.0381 m
Bearing offset from edge	0.0127 m	0.0127 m
Center of mass position, s	0.146 m	0.125 m
Damping coefficient, c	Estimate*	Estimate*

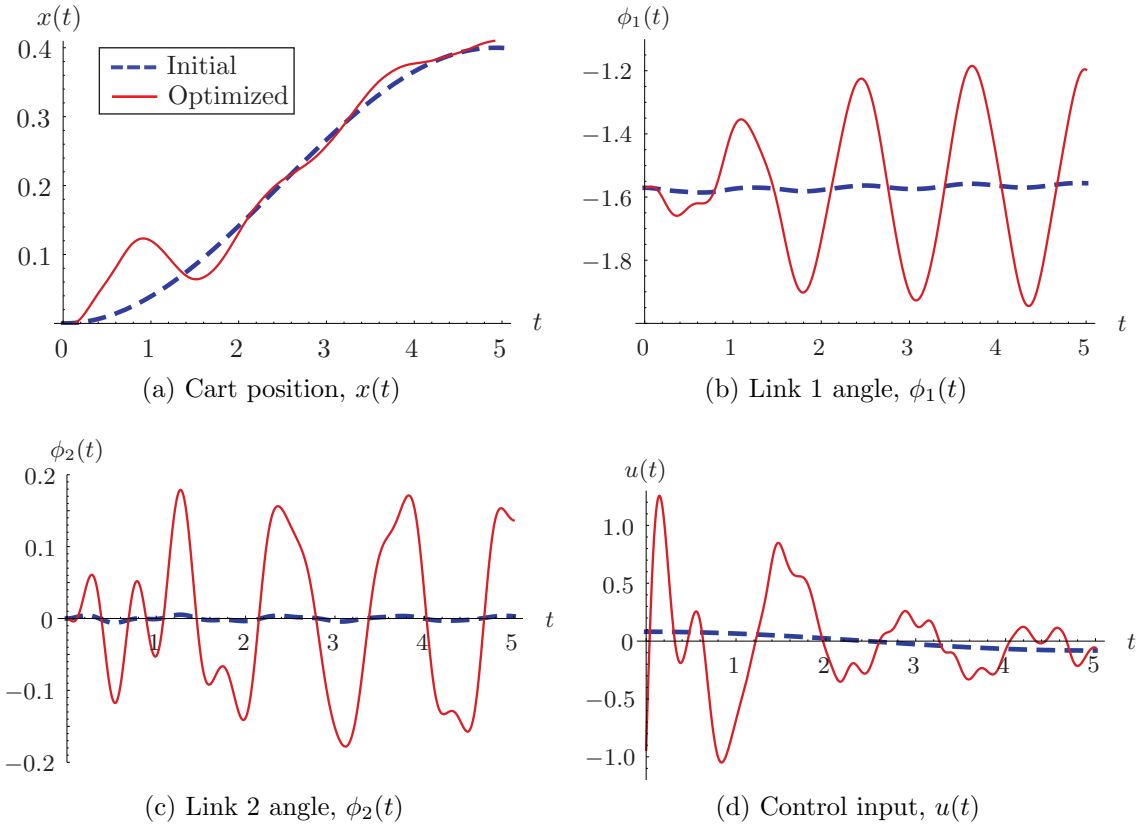
*The values for the damping coefficient of each joint will be assumed to be equal.

another trajectory should be tested for identifiability. If no trajectory can be found, the algorithm cannot be initialized and a different set of parameters should be chosen. The experiment that follows in this paper is performed under the assumption that the two unknown system parameters are the mass of the top link m_1 and the damping coefficient of the top link joint and bottom link joint c. Since both links use the same bearings, we will assume that the damping coefficients of the two joints are equal. The remaining system parameters have been measured and are presented in Table 4.1. Additionally, Table 4.2 lists the eigenvalues of the FIM for the initial trajectory. Since both eigenvalues are non-zero, the parameters can be simultaneously estimated to some precision.

Given that m_1 and c are the uncertain parameters, the following values will be used as initial estimates of the parameters,

$$m_1 = 0.085 \text{ kg} \qquad c = 0.50 \text{ g/sec.} \tag{4.5}$$

Note that the choice of units weights the desired precision between estimated parameters. In this case, the damping coefficient units are scaled to provide more relative precision in the mass estimate. The remaining system parameters which appear in Table 4.1 will be considered the actual known model parameters.



(a) Cart position, $x(t)$

(b) Link 1 angle, $\phi_1(t)$

(c) Link 2 angle, $\phi_2(t)$

(d) Control input, $u(t)$

Figure 4.5. Comparisons of the trajectory before and after Fisher information optimization.

4.4. Fisher Information Experimental Results

4.4.1. Optimization Results

The optimization algorithm was run until a convergence criterion of $|DJ(\xi(t)) \circ \zeta| < 10^{-1}$ was satisfied, starting from an initial descent magnitude around $8 \cdot 10^3$. The comparison of initial and optimized trajectories can be seen in Fig. 4.5.

Table 4.2 shows the initial and optimized eigenvalues of the FIM, $\tilde{I}(\theta)$, and the initial and optimized cost J. The results show that the minimum eigenvalue λ_2 increases by over

Table 4.2. Information Optimization Results

	λ_1	λ_2	J
Initial:	123.3	0.0315	15.9
Optimized:	$1.43 \cdot 10^5$	32.6	0.618
	Fisher Information Matrix		
Initial:	$\begin{bmatrix} 3.72 \cdot 10^3 & -5.14 \\ -5.14 & 9.57 \cdot 10^{-1} \end{bmatrix}$		
Optimized:	$\begin{bmatrix} 4.31 \cdot 10^6 & -3.77 \cdot 10^3 \\ -3.77 \cdot 10^3 & 9.91 \cdot 10^2 \end{bmatrix}$		

a factor of 10^3. Additionally, the other eigenvalue increases, though not included directly in the cost function.

Examining the plots of the optimized trajectory, it is clear that more information is gained by oscillating the pendulum back and forth. In particular, more information about the damping parameter c is gained by increasing the oscillation. This observation leads to a hypothesis that the cost function and optimization may be driven strongly by information concerning the damping parameter, although the result increases the overall amount of information for both parameters.

Using this optimized trajectory, simulation and experimental results of the system as well as the experimental testbed are presented in the following sections to demonstrate the improvement in the estimation of system parameters.

4.4.2. Monte-Carlo Simulation Analysis

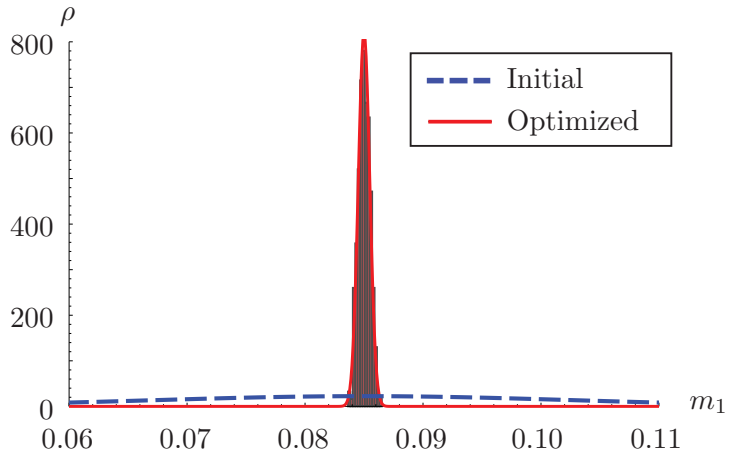
Given the system model and parameters, results are first obtained in simulation to assess the effectiveness of the algorithm given an exact model and pure Gaussian noise

Table 4.3. Monte-Carlo Simulation Results

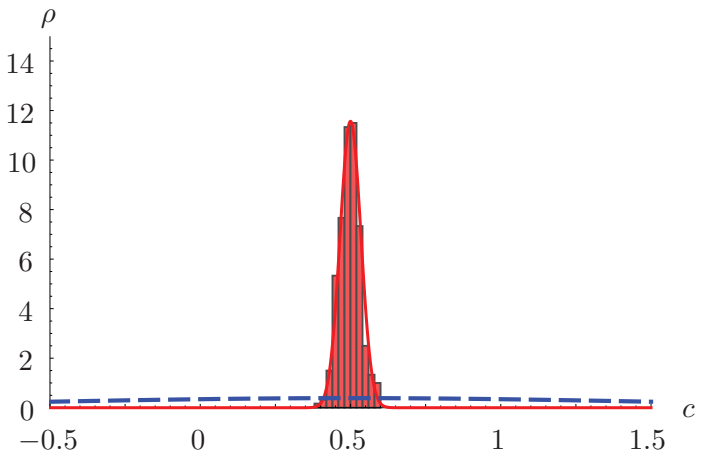
	Monte-Carlo Covariance
Initial:	$\begin{bmatrix} 3.16 \cdot 10^{-4} & 3.45 \cdot 10^{-3} \\ 3.45 \cdot 10^{-3} & 1.06 \end{bmatrix}$
Optimized:	$\begin{bmatrix} 2.37 \cdot 10^{-7} & 1.63 \cdot 10^{-6} \\ 1.63 \cdot 10^{-6} & 1.19 \cdot 10^{-3} \end{bmatrix}$
	Cramer-Rao Lower Bound
Initial:	$\begin{bmatrix} 2.71 \cdot 10^{-4} & 1.46 \cdot 10^{-3} \\ 1.46 \cdot 10^{-3} & 1.05 \end{bmatrix}$
Optimized:	$\begin{bmatrix} 2.32 \cdot 10^{-7} & 8.83 \cdot 10^{-7} \\ 8.83 \cdot 10^{-7} & 1.01 \cdot 10^{-3} \end{bmatrix}$

distributions. An experimental measurement of the trajectory is simulated by generating the trajectory of the deterministic system using our estimate of the parameters and sampling the trajectory at a discrete number of points adding Gaussian additive noise at each sample. This results in a discrete set of noisy measurements that will be used for the subsequent parameter optimization routine. For the simulation example, a 5 second trajectory is simulated at the Kinect's sampling rate of 30 Hz. The sampling rate is set by the frequency of sensors being used. The uncertainty of each state measurement is normally distributed with zero mean and variance equivalent to that measured by the Kinect during the testbed setup given by (4.4).

A Monte-Carlo simulation is performed using 300 trials, resampling the trajectory with new additive Gaussian noise samples for each trial. For the scope of the Monte-Carlo simulation analysis, the parameter estimates from (4.5) are treated as the actual system parameter set used to generate the simulated output and the initial parameter



(a) PDF histograms of the top link mass, m_1.



(b) PDF histograms of the damping coefficient, c.

Figure 4.6. Monte-Carlo histogram approximations of the PDF with respect to the Lebesgue measure of the optimized trajectory with the optimized variance fit in red and the initial trajectory variance in blue.

estimate are uniformly varied in the range of ±100% of each parameter value in (4.5). This provides simulation results that are independent of the initial estimate.

After running the simulation, histogram plots in Fig. 4.6 show the distribution of parameter estimates using the optimized trajectory. The histogram of the optimized trajectory is shown by the red bins with a Gaussian fit to the mean and standard deviation shown by the solid red line. The dashed blue line indicates the Gaussian fit to the histogram results of the initial trajectory; however, to simplify the plot, the initial histogram is not shown. The averaged statistics of the simulation trials can be seen in Table 4.3. Since the expected information has been significantly increased with the optimized trajectory, the results confirm that the covariance of the estimated parameters decreases dramatically. The precision of both parameter estimates with respect to the Lebesgue measure is improved by a factor of 10^3.

4.4.3. Cramer-Rao Lower Bound

The estimator's covariance result should also be compared with the theoretical Cramer-Rao bound. As discussed in Section 2.1.3, the Cramer-Rao bound places an absolute lower bound on the variance of the parameter estimate that can be obtained using the batch least-squares estimator or other unbiased estimator. The bound is given in (2.1).

Table 4.3 lists the Cramer-Rao bounds for the initial and optimized trajectories. The covariance of the initial trajectory is clearly subject to a higher bound than that of the optimized trajectory. Due to round-off and other numerical errors in the algorithms and Monte-Carlo simulations, the covariance of the Monte-Carlo estimates is higher than the lower bound; however, overall remains within a factor of 3 of the predicted best-case variance estimates according to the Cramer-Rao bound.

Table 4.4. Experimental Results

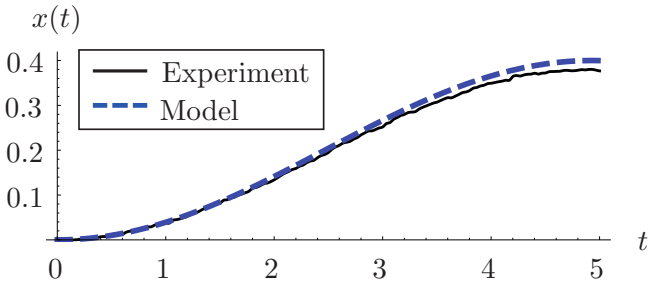
	m_1 (kg)	c (g/sec)
Initial Estimate:	0.085	0.500
Measured Actual Value:	0.110	0.180
Initial Trajectory Estimate:	0.085	0.500
% Error from Actual Value:	22.7%	316.7%
Optimal Trajectory Estimate:	0.107	0.211
% Error from Actual Value:	2.7%	17.2%

4.4.4. Experimental Results

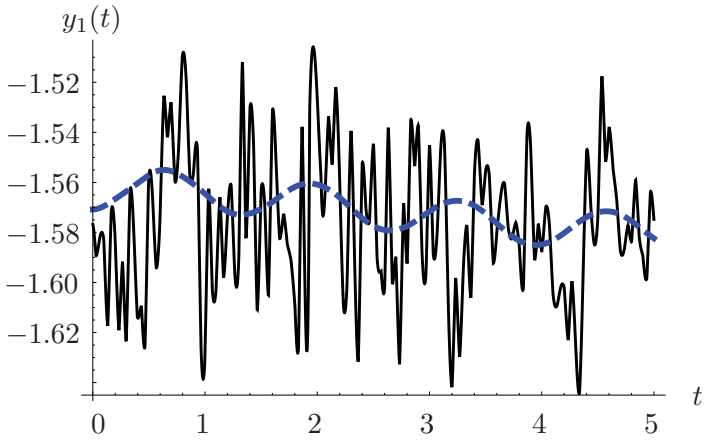
After validating the optimization results in simulation, the trajectories were tested on the experimental testbed to determine their experimental effectiveness. Each trajectory was run on the system with the observed angles and position of the robot recorded by the Kinect tracking system. The recorded data can be seen in Figs. 4.7 and 4.8.

Using the collected data as the reference for the least-squares parameter optimization, the parameter set estimate from (4.5) is used as the initial estimate for the parameter optimization routine. Using the batch-least squares estimation method, the best estimates of the parameters were found based on the data collected. The results can be seen in Table 4.4. Actual values of the parameters for the experimental system were obtained by disassembling the pendulum system. The mass of each link was determined by individually weighing each link, and the damping coefficient was obtained from a batch-least squares estimate of a single link in a free swinging trajectory.

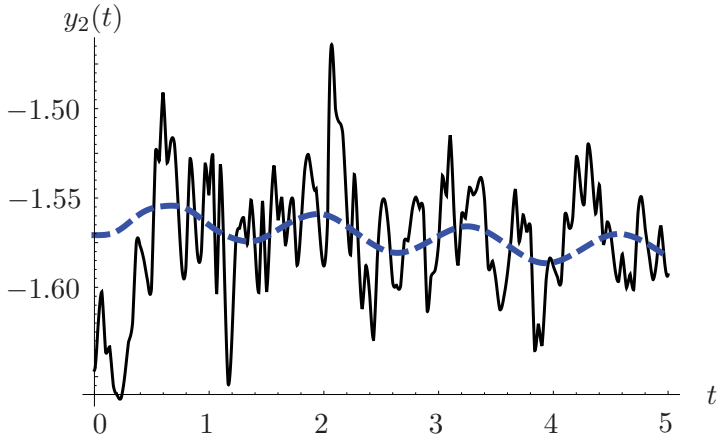
Since the noise present in the initial trajectory is so high relative to the observed angles, the parameter estimation algorithm is not able to make any progress from the initial estimate of the parameters. This results in a 22.7% error in the mass and 316.7% error



(a) Cart position, $x(t)$

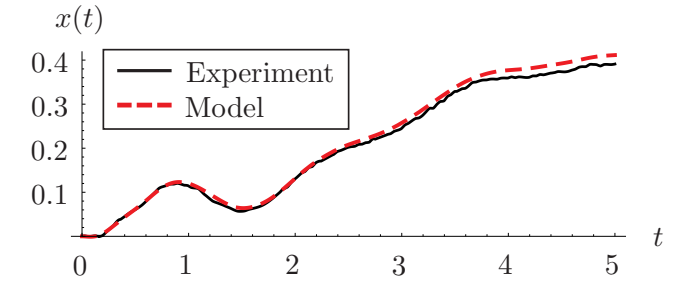


(b) Link 1 angle, $\phi_1(t)$

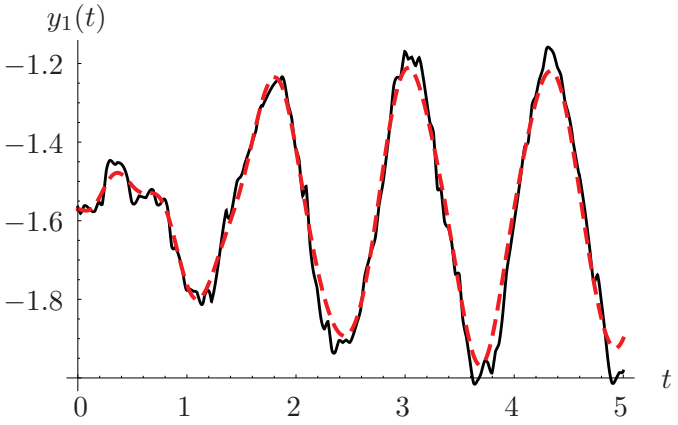


(c) Link 2 angle, $\phi_2(t)$

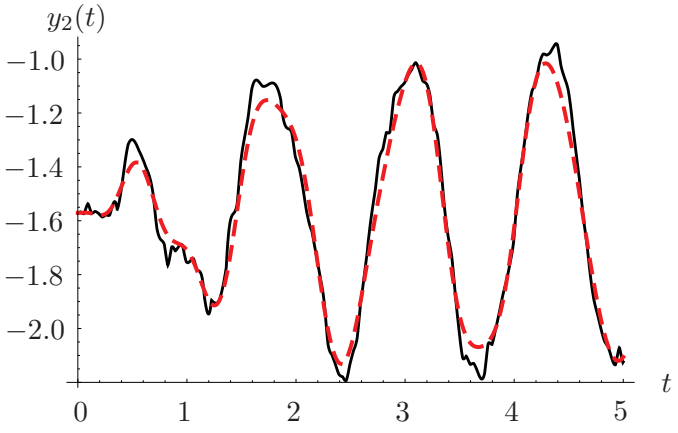
Figure 4.7. Comparisons of the measured trajectories and estimated model trajectory for the initial choice of experimental trajectory.



(a) Cart position, $x(t)$



(b) Link 1 angle, $\phi_1(t)$



(c) Link 2 angle, $\phi_2(t)$

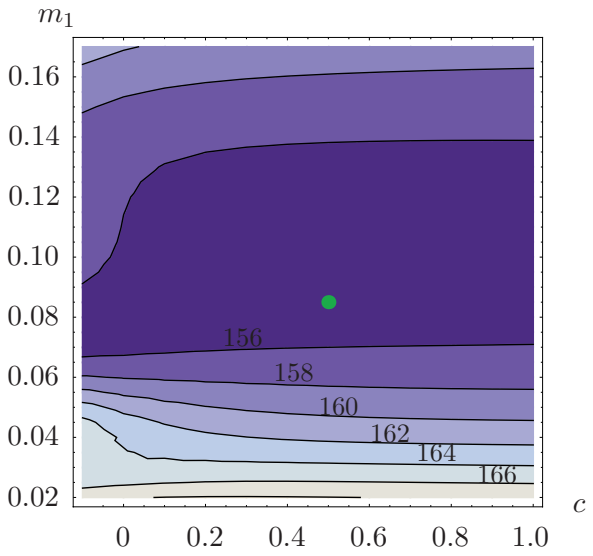
Figure 4.8. Comparisons of the measured trajectories and estimated model for the optimized trajectory based on the Fisher information metric.

in the damping coefficient from the baseline values. The optimized trajectory provides parameter estimates with only 2.7% error in the mass and 17.2% error in the damping coefficient values.

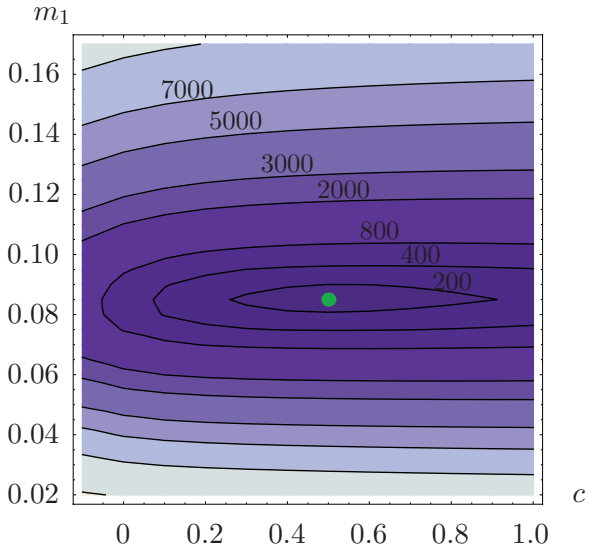
A contour plot of the parameter estimator cost β is shown in Fig. 4.9. Using the experimental trajectory data, the cost was computed for a portion of the parameter space. The figure shows a well-defined optimization basin for the optimized trajectory where the basin of the initial experimental trajectory is far less convex. This optimized contour plot illustrates why the data from the optimized trajectory provides the better estimate of the parameters of interest given the observed measurements. The E-optimality cost function, however, does not explicitly condition the optimized basin as seen in Fig. 4.9b. In some cases, it may be advantageous to use an additional approach to improve the condition number of the optimization problem as well [**16**].

It is important to note that the error in the experimental case is far greater than the predicted Cramer-Rao bound in Table 4.3. This is due to additional disturbances and unmodeled effects such as out of plane swinging motion, error in the control signal, and sensor nonlinearities. These effects will cause bias error in the estimator; however, even with these sources of bias, the optimized trajectory provides a much better estimate of model parameters than the initial system trajectory.

In the case of multiple parameter estimation, one concern not addressed in this cost function is the conditioning or existence of a well-posed estimator. Situations may occur where good converge rates of the estimator are as important as the precision for the estimation. The following sections highlight a modification to the algorithm in the case where a well-conditioned estimator is desired.



(a) Initial trajectory



(b) Optimized trajectory

Figure 4.9. Comparisons of parameter optimization cost contours with isolines indicating the estimator cost, $\beta(\theta)$. The deterministic estimate of the parameter is shown by the green dot in the contour plot. The parametric map of the optimized trajectory is significantly more convex than the initial trajectory.

4.5. Trajectory Optimization for Well-Conditioned Estimation

The goal of Fisher information optimization algorithm is to improve the estimation of the worst performing parameter direction; however, when using a gradient descent or Iterated Least-Squares algorithm with multiple unknown parameters, another important metric is the condition number of the estimator Hessian. As mentioned in Section 2.2.1.1, the conditioning, and thus the rate of convergence is dependent upon the minimum and maximum eigenvalue of the Hessian.

To improve the convergence rates using either gradient descent or Iterated Least-Squares estimation methods, an initial experimental trajectory having a Hessian with a high condition number can be optimized to produce an experimental trajectory with better Hessian conditioning. The theoretical framework for the optimization routine shares many of the same derivations as the Fisher information algorithm but with a modified cost related to the condition number of the Hessian.

4.5.1. Cost Function

The cost function, which is minimized by the trajectory optimization algorithm, directly incorporates the condition number of the parameter estimation Hessian, a control cost, and an optional cost on the trajectory itself. The trajectory optimization algorithm uses the current best estimate of the parameter set. Therefore, it is assumed that the parameter set is close to the true value; thus the Hessian is approximated by (2.10).

Using this formulation, the condition number of the approximated Hessian can be calculated. Assuming that the Euclidean norm is used and the Hessian is symmetric, the

condition number is equal to

$$\kappa(D_\theta^2 \beta(\theta)) = \left| \frac{\lambda_{max}}{\lambda_{min}} \right|,$$

where λ is the set of eigenvalues of the matrix $\frac{d^2}{d\theta^2}\beta(\theta)$.

A well-conditioned estimator has a condition number very close to 1, therefore, the cost function for the trajectory optimization problem is given by

$$J = Q_p \left(\frac{\lambda_{max}}{\lambda_{min}} - 1 \right) + \frac{1}{2} \int_{t_0}^{t_f} (x(t) - x_d(t))^T \cdot Q_\tau \cdot (x(t) - x_d(t)) + u(t)^T \cdot R_\tau \cdot u(t) \; dt, \tag{4.6}$$

where Q_p is a scalar weight on the condition number minimization, Q_τ is a $n \times n$ weighting matrix on the trajectory tracking cost, and R_τ is a $r \times r$ weighing matrix on the control inputs.

It can be clearly seen that the resulting cost function (4.6) is very similar to that of (4.3) in Section 4.2. Instead of using the worst-performing eigenvalue as the metric, the condition number looks at the ratio of the eigenvalue spread. Thus, any direction which is drastically different than another will be optimized to produce an estimator which quickly converges under the Iterated Least-Squares method.

4.5.2. Optimization Routine

The optimal control problem is solved using the same iterative descent technique shown in Algorithm 2.2. As was the case with the Fisher information optimization method, the state is extended with the sensitivity states as described in Section 4.2.2. The projection operator also is equivalent to the projection specified in Section 4.2.3. Given the descent direction, ζ_i, a backtracking line-search of the projection, $P(\eta_i(t) + \gamma_i \zeta_i)$ provides a feasible

trajectory solution assuming that the step size γ_i satisfies the Armijo sufficient decrease condition. This new feasible trajectory then becomes the trajectory for the next iteration of the optimal control algorithm.

At each iteration of the conditioning optimization algorithm, the time-varying LQ problem given by (2.17) must be solved. The descent direction depends on the linearization of the cost function, $DJ(P(\xi_i(t)))$ and the local quadratic model, $\frac{1}{2}\langle\zeta_i(t), \zeta_i(t)\rangle$. Details of the derivation of the descent direction for the conditioning cost are shown in Appendix B.1.

4.6. Experimental Setup for Conditioning Optimization

To illustrate the use of conditioning optimization on a nonlinear, dynamic system, a simulation and experimental test of a two dimensional robot and suspended mass system is analyzed. The system has two configuration variables, $q = (x(t), \phi(t))$, where x is the horizontal displacement of the cart and ϕ is the rotational angle of the link as seen in Fig. 4.10. For this example, the robot's horizontal acceleration will be directly controlled,

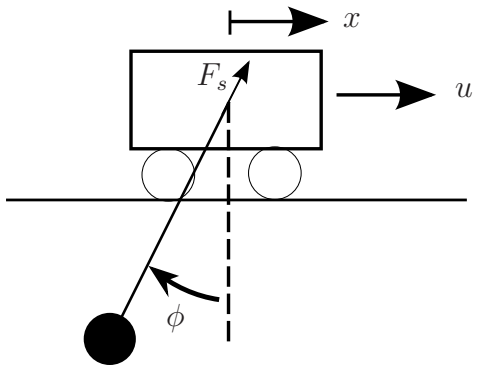


Figure 4.10. Diagram of the robot and suspended mass system.

therefore, $u(t) = \ddot{x}(t)$. A load cell provides the magnitude of string tension force as the sole output of the system.

4.6.1. System Model

The equations of motion for the system are given by the following

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ u \\ \dot{\phi} \\ \frac{u}{\ell}\cos\phi - \frac{g}{\ell}\sin\phi \end{bmatrix},$$

where u is the cart acceleration control input, m is the mass suspended from the robot, and ℓ is the length of the string. Additionally, the equation for the force output F_s is

$$\mathbf{y} = F_s = mg\cos\phi - m\ell\dot{\phi}^2 - u\sin\phi. \tag{4.7}$$

It is assumed that the trajectories will maintain tension in the string; therefore, a fixed distance between the robot and mass can be assumed.

The goal of the experiment is to estimate the value of the mass attached to the string and the length of the string, $\theta = \{m, \ell\}$. The choice of force as the system output highlights the effect of the trajectory on the conditioning of the estimation problem. Static or nearly static trajectories, such as the initial trajectory shown in Fig. 4.12a, provide valid estimates of the mass value; however, the string length cannot be resolved. On the other hand, dynamic trajectories provide fluctuating force measurements which can be

used in conjunction with the predicted model to simultaneously estimate the length and mass values.

4.6.2. Experimental Testbed Setup

The experimental setup shown in Fig. 4.11 consists of a differential drive mobile robot with magnetic wheels moving in a plane. The robot's driving surface is provided by a smooth steel plate mounted above and parallel to the ground. Two 24V DC motors drive the robot's magnetic wheels. PID loops running at 1500 Hz close the loop around motor velocity using optical encoders for feedback. Two 12V lithium iron phosphate batteries provide the required power. Desired velocity commands are sent to the robot wirelessly using Digi XBee® modules at 50 Hz, and a 32-bit Microchip PIC microprocessor handles all on-board processing, motor control, and communication.

The motors are powerful relative to the inertias of the robot, the suspended mass, and the wheels. Thus they are capable of accurately tracking aggressive trajectories up to a maximum rotational velocity. Since the input to the system is the direct acceleration of the cart, PID loops on the motor velocity allow the robot to accurately track a given velocity profile. A load cell is attached to the robot and the mass is suspended from its measurement point to enable collection of force data using an instrumentation amplifier and 10-bit, on-board ADC. The load cell output has been fit to a linear model using six known masses ranging from 0.05kg to 0.30kg. The Robot Operating System (ROS) is used for interpreting and transmitting desired trajectories and for collecting and processing all of the experimental data [**84**].

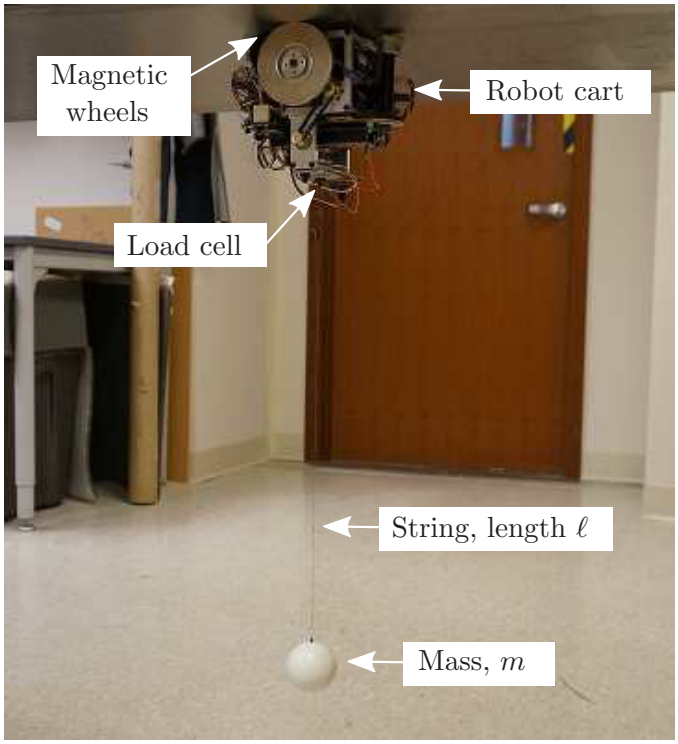


Figure 4.11. Image showing the robot and suspended mass experimental system with a diagram of main components.

4.6.3. Identification Procedure

The procedure for completing the optimizations and taking measurement data from the experimental system is performed in the following order.

(1) Perform the conditioning optimization using the current best estimate of the parameters which yields a new set of control inputs for the system.

(2) Run the experiment using the newly generated inputs and take measurement of the output along the trajectory.

(3) Perform the least-squares parameter optimization using the optimized trajectory to determine a new estimate of the parameter set.

4.7. Conditioning Optimization Experimental Results

4.7.1. Overview

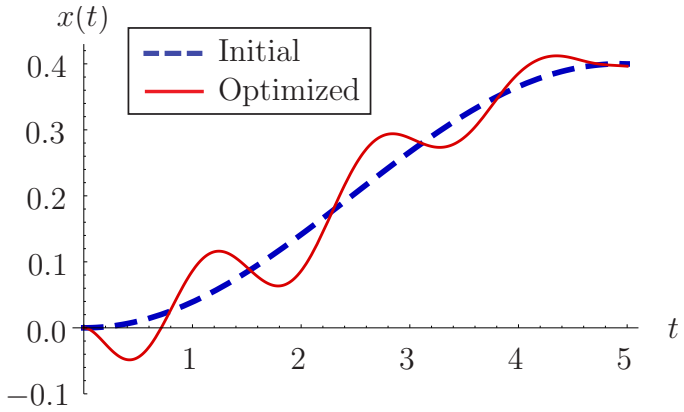
The following section presents the results of the optimization study in both simulation and experimental trials. Given an initial trajectory and set of unknown model parameters, the conditioning algorithm is used to synthesize an optimized trajectory. The key results are as follows:

- The condition number of the parameter optimization Hessian decreases from 1.73×10^8 using measurements from the initial trajectory to 19.3 using optimized trajectory measurements.
- In simulation, 27 gradient descent steps are needed to converge to the actual parameter values using optimized trajectory measurements compared to over 1000 using the initial trajectory.
- Using the experimental system, the error in predicted string length decreases to 2.4% using optimized trajectory measurements compared to 122.% using the initial trajectory data.

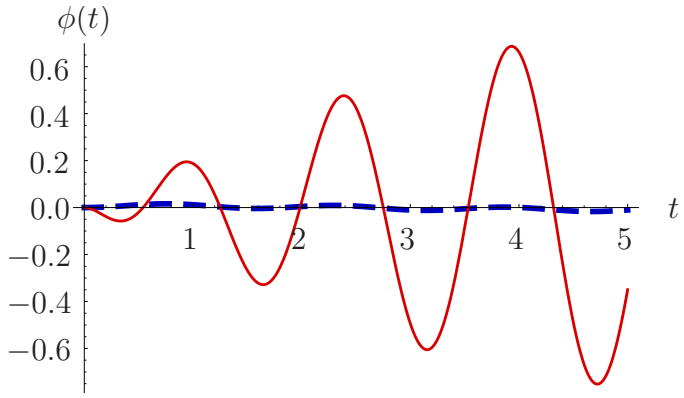
Detailed results and analysis of the data is provided in the sections that follow.

4.7.2. Conditioning Optimization Results

The two parameters that will be estimated are suspended mass value and the string length. An initial, and intentionally incorrect, estimate of the parameters was chosen for both the gradient descent and the Iterated Least-Squares estimation algorithms. The actual values of the parameters were independently measured to provide a benchmark for algorithm



(a) Robot position, $x(t)$



(b) Angle of suspended mass with respect to the robot, $\phi(t)$

Figure 4.12. Plots showing the robot (a) and suspended mass (b) trajectory before and after conditioning optimization.

verification. These values are as follows,

$$\text{Initial estimate} : m = 0.10 \text{ kg}, \quad \ell = 0.60 \text{ m}$$

$$\text{Actual values} : m = 0.12 \text{ kg}, \quad \ell = 0.50 \text{ m}$$

A minimal control trajectory is chosen as the initialization for the conditioning optimization algorithm as shown in Fig. 4.12. Using the initial estimate of the parameter set

Table 4.5. Conditioning Optimization Results

	Initial	Optimal
First Eigenvalue, λ_1:	3.18×10^6	3.63×10^6
Second Eigenvalue, λ_2:	1.84×10^{-2}	1.88×10^5
Condition Number, κ:	1.73×10^8	1.93×10^1
Optimization Cost, J_τ:	8.63×10^4	1.02×10^2

Table 4.6. Number of iterations to converge to $|J_p| < 10^{-6}$

	Initial	Optimal
Gradient Descent:	$> 1000^*$	27
ILS:	5	3

*Estimation was stopped after 1000 iterations with no significant improvement in J_p

and initial trajectory, the optimization algorithm was run until a convergence criterion of $(J(\eta_{i+1}(t)) - J(\eta_i(t))) < 10^{-1}$ was satisfied. The comparison of initial and optimized trajectories can be seen in Fig. 4.12.

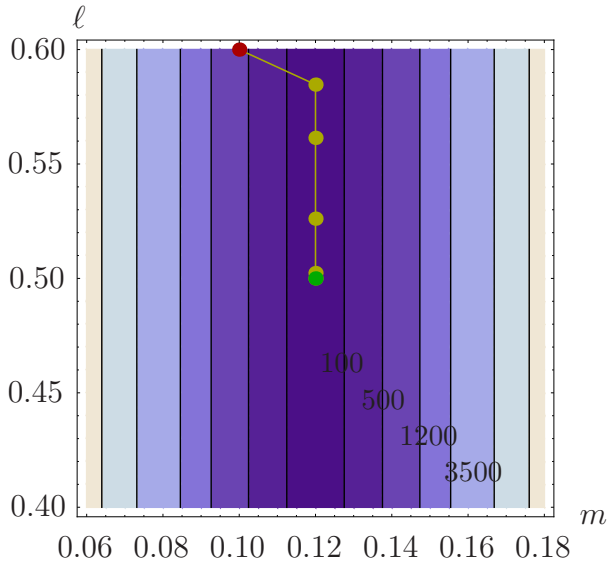
The eigenvalues of $\frac{d^2}{d\theta^2}\beta(\theta)$, the condition number κ, and the cost J for the initial trajectories and the optimized trajectories are listed in Table 4.5. The results show that the condition number, κ, decreases drastically from 1.73×10^8 to 19.3. Thus a significant improvement in the convergence rate of the parameter estimation algorithms, especially the gradient descent method, can be expected.

The plots of the optimized trajectory show that the oscillation of the suspended mass improved the simultaneous estimation conditioning involving the suspended mass and length parameters. While the load cell can correctly estimate the mass given a static or near static system, the length can only be estimated given a dynamic motion.

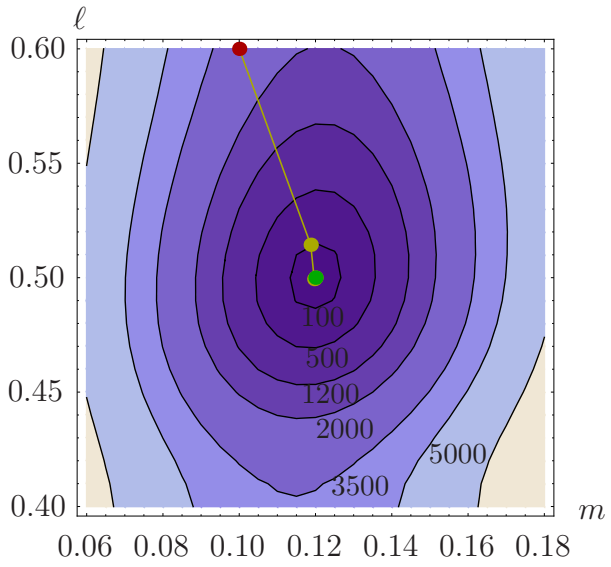
In simulation, the gradient descent and Iterated Least-Squares parameter estimation algorithms were executed using the initial and optimal trajectories. The convergence results appear in Table 4.6. As the results indicate, second-order information in the Iterated Least-Squares method greatly improves convergence over the gradient descent method, even for the initial, ill-conditioned problem. The benefits of conditioning optimization are evident in the significantly improved convergence of the gradient descent algorithm. The optimal trajectory does provide a convergence improvement for the Iterated Least-Squares method; however, it is less significant.

The convergence path of the Iterated Least-Squares algorithm is shown on contour plots of the parametric space in Fig. 4.13. In the initial trajectory, the lack of information about the length parameter leads to the ill-conditioning of the problem. The algorithm quickly converges to the correct mass yet takes many extra steps to converge to both parameters. After optimization, the optimal trajectory yields improved conditioning of the cost basin which allows the algorithm to more directly converge on both parameters simultaneously.

In simulations with no measurement or model uncertainty, the optimal trajectory only provides slight improvement in the Iterated Least-Squares convergence rate; however, in practice, use on experimental systems introduces measurement noise and model uncertainty which greatly impacts the estimates using the two trajectories. The following section provides experimental results illustrating this point.



(a) Contour plot showing convergence in simulation using the initial trajectory.



(b) Contour plot showing convergence in simulation using the optimized trajectory.

Figure 4.13. Contour plots showing the parameter estimation cost J across the parametric space with the convergence path shown by the colored dots. The red dot indicates the initial parameter estimate, the yellow dots show intermediate iteration points, and the green dot indicates the converged point.

Table 4.7. Experimental Results

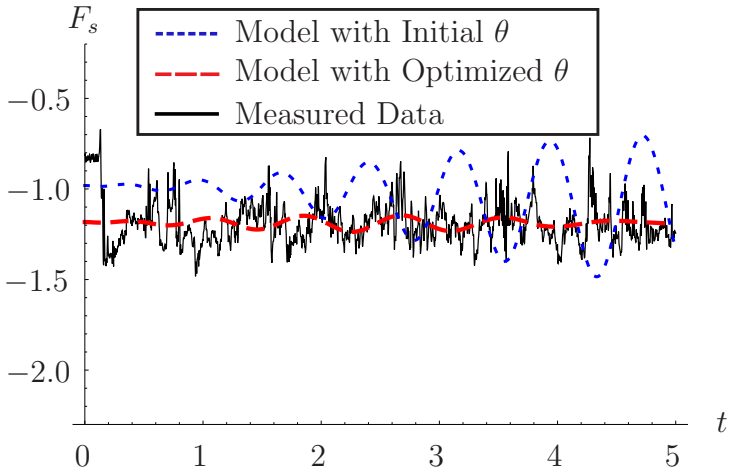
	m (kg)	ℓ (m)
Initial Estimate:	0.100	0.60
Measured Baseline Value:	0.124	0.50
Estimate from Initial Trajectory:	0.121	1.11
% Error from Baseline Value:	2.4	122.
Optimal Trajectory Estimate:	0.121	0.51
% Error from Baseline Value:	2.4	2.0

4.7.3. Experimental Results

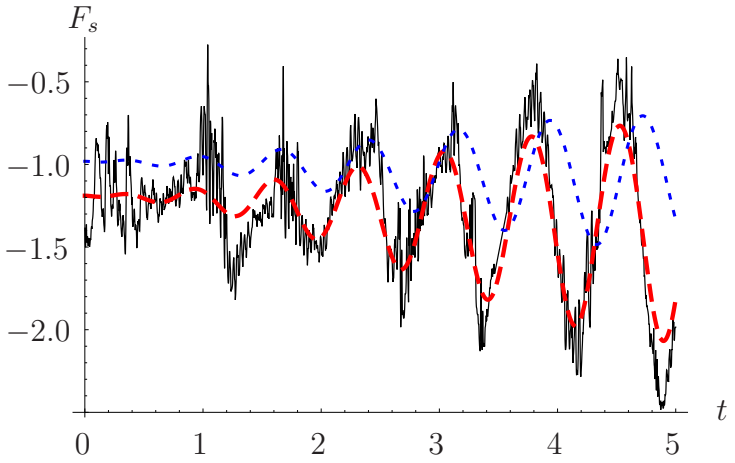
Using the same initial and optimized control inputs shown in Fig. 4.12, each trajectory was run on the experimental platform described in Sec. 4.6.2. During each trial, force data was collected at 300Hz and interpolated to provide continuous data for the estimation algorithm. Following the data collection, the Iterated Least-Squares algorithm was run on each set of data to attempt to find a best estimate of the parameter set using the measured output.

The results of the experiment are shown in Fig. 4.14 and Table 4.7. The impact of an ill-conditioned problem is accentuated in the experimental results. While both trajectories provide estimates that are within 2.4% of the mass value, the length parameter estimate from the initial trajectory has a 122% error compared to only 2.0% with the optimized trajectory.

Fig. 4.14 shows the predicted force output using the initial parameter estimate with the blue dotted line before running the Iterated Least-Squares estimation algorithm. After running the estimation algorithm on both trajectories, the model predicted force output



(a) Model and data from the initial experimental trajectory.



(b) Model and data from the optimized experimental trajectory.

Figure 4.14. Plots showing measured force data collected from the experimental trial and predicted force data using both initial and optimized parameter values.

for the optimized parameters is shown by the red dashed line. The force data collected by the load cell on the robot platform is shown by the solid black line.

Examining this data, the initial trajectory results in small angle displacements of the mass. Since the displacement is small compared to the noise, the model is fit mostly to noise, providing an inaccurate estimate of the length as seen in Table 4.7. As the mass is purely a function of the mean of the signal, the signal from both trajectories is sufficient to provide a close estimate of the suspended mass. Synthesizing a new trajectory with better conditioning results in oscillations of the mass, which provide greater sensitivity in the measurement output with respect to the length parameter, resulting in better estimate convergence.

To summarize the results, using the condition number cost balanced with appropriate weights of control effort and tracking error results in an improved trajectory while allowing users to assign importance to each component. Different scenarios will require varying levels of constraint on control effort and tracking error while attempting to estimate system parameters. One particular area of improvement is reducing the computational complexity due to the choice of continuous numerical method. The goal of faster, more stable performance as shown in the previous chapter motivates the derivation of the algorithm in discrete time which is presented in the following chapter.

CHAPTER 5

Information Optimization Using Discrete Mechanics

This chapter provides a discrete-time formulation of the continuous-time Fisher information maximization algorithm presented in Section 4.2. The algorithm uses a discrete formulation of the gradient-descent projection-based trajectory optimization algorithm [**33**] to find a trajectory that locally maximizes the information of an experimental trajectory to produce the best estimate of the model parameters.

The motivation for designing an estimation algorithm using discrete mechanics is two-fold. First, the continuous-time algorithm requires numerical solutions to several differential equations which provides flexibility in its implementation; however, as demonstrated in Chapter 3, continuous-time algorithms may result in long computation times depending on the numerical method that is used. Using a result of much recent work in the area of discrete mechanics, variational integrators, also known as *structured integrators* [**86**], can be used to provide a time-discretized version of the action integral which results in a symplectic form with stable long-term energy behavior, even with large time-steps [**35**]. These integrators have been applied to the simulation of complex mechanical systems using generalized coordinates with accuracy comparisons to traditional numerical integrators in [**87**].

Second, the continuous-time algorithm requires a continuous cost function which necessitates the approximation of Fisher information, defined as $\tilde{I}(\theta)$ in (4.2) as a continuous measure along the trajectory. This is an approximation which converges to the actual Fisher information as the measurement frequency increases. Formulating the optimization in discrete-time allows for a discrete cost, which includes the exact Fisher information of the trajectory.

5.1. Parameter Estimation for Discrete Mechanical Systems

The formulation of the nonlinear least-squares estimator remains the same as presented in Section 2.2.1; however, the dynamics constraints are now in terms of the discrete mechanical system. Using the discrete mechanics notation, the nonlinear estimator in (2.4) can be written as

$$\beta(\theta) = \frac{1}{2}\sum_{k}^{k_f}(\tilde{y}_k - y_k)^T \cdot \Sigma^{-1} \cdot (\tilde{y_k} - y_k),$$

where $\tilde{y_k}$ is the observed state at the k^{th} index of t_f/dt measurements.

For the Fisher information maximization algorithm used in this chapter, the gradient descent method is used to find the optimal set of parameters, $\hat{\theta}$. Therefore, the first derivative of $\beta(\theta)$ must be calculated w.r.t θ.

$$D_\theta\beta(\theta) = \sum_{k}^{k_f}(\tilde{y}_k - y_k)^T \cdot \Sigma^{-1} \cdot \Gamma_k,$$

where

$$\Gamma_k = D_x g(x_k, \bar{u}_k, \theta) \cdot \frac{dx_k}{d\theta} + D_\theta g(x_k, \bar{u}_k, \theta).$$

Note that the control $\bar{u}$ now includes both dynamic and kinematic inputs as specified in Section 2.4.2.

The gradient requires the evaluation of $\frac{dx_k}{d\theta}$ which can be found using first-order linearizations of x_k from the DEL equations. Since ρ and ν are kinematic configurations, $\frac{d\rho_k}{d\theta}$ and $\frac{d\nu_k}{d\theta}$ are zero for all k. To match notation in the continuous time problem, we note that the sensitivity ψ_k is given by

$$\psi_k = \begin{bmatrix} \frac{dq_k}{d\theta} \\ \\ \frac{dp_k}{d\theta} \end{bmatrix}.$$

The first-order sensitivity is thus given by

$$\psi_{k+1} = A_k \cdot \psi_k + \frac{\partial x_k}{\partial \theta}, \tag{5.1}$$

where

$$\begin{aligned} \frac{\partial q_{k+1}}{\partial \theta} &= -M^{-1} \cdot (D_5 D_1 L_{k+1} + D_6 F^-_{k+1}) \\ \frac{\partial p_{k+1}}{\partial \theta} &= D_2^2 L_{k+1} \cdot \tfrac{\partial q_{k+1}}{\partial \theta} + D_5 D_2 L_{k+1} \end{aligned}$$

$$A_k = \begin{bmatrix} \frac{\partial q_{k+1}}{\partial q_k} & \frac{\partial q_{k+1}}{\partial p_k} \\ \\ \frac{\partial p_{k+1}}{\partial q_k} & \frac{\partial p_{k+1}}{\partial p_k} \end{bmatrix}.$$

We refer readers to [**31**] for the equations representing the components of A_k and M which are derived and presented in full.

5.2. Maximizing Fisher Information using Discrete Mechanics

The projection-based trajectory optimization algorithm is used in a similar manner to the continuous-time case; however, the algorithm itself has been reformulated in [**33**] into a discrete-time framework. This discrete version is extended in the following section to include Fisher information as an additional objective.

5.2.1. Objective Function

The objective function consists of the same three components as the continuous algorithm: the Fisher information cost, a trajectory tracking cost, and a control cost. Since the optimization algorithm is formated in discrete-time, the Fisher information can be directly used in the algorithm versus the continuous-time formulation presented in the previous work. Since the FIM is a matrix quantity, we use E-optimality, which uses the minimum eigenvalue of the FIM as the cost metric.

The exact FIM given by (4.1) is used as the information cost. Therefore, the optimization objective function is defined as

$$J = \frac{Q_p}{\lambda_{min}} + \sum_{k=k_0}^{k_f} (x_k - \hat{x}_k)^T \cdot Q_\tau \cdot (x_k - \hat{x}_k) + (\bar{u}_k - \hat{\bar{u}}_k)^T \cdot R_\tau \cdot (\bar{u}_k - \hat{\bar{u}}_k), \tag{5.2}$$

where λ_{min} is the minimum eigenvalue of $I(\theta)$, Q_p is the information weight, $\hat{x}_k$ is a reference trajectory, $\hat{\bar{u}}_k$ is a reference control signal, Q_τ is a trajectory tracking weighting matrix, and R_τ is a control effort weighting matrix. The weights must be chosen such that $Q_p \geq 0$, Q_τ is positive semi-definite, and R_τ is positive definite.

The various weights allow for design choices in the optimal trajectory that is obtained. The requirements of positive definiteness and positive semi-definiteness of the weighting matrices are necessary to maintain a locally convex optimization problem including the fact that $\lambda_{min} \geq 0$ [**32**]. Increasing the control weight will result in less aggressive trajectories, generally decreasing the obtained information. Using a reference trajectory allows for an optimal solution that remains in the neighborhood of a known trajectory.

5.2.2. Extended Dynamic Constraints

As with the continuous trajectory optimization method, the discrete first-order sensitivity ψ_k must be computed along the trajectory to extend the optimal control algorithm to include the FIM metric in (5.2). Appending the discrete parametric sensitivity to the state vector as an additional dynamic constraint allows for variations in ψ_k in the optimization algorithm. Including the dynamic and kinematic control inputs, the extended state will be defined by $\bar{x}_k = (x_k, \psi_k)$, and $\eta_k = (\bar{x}_k, \bar{u}_k)$ which defines a curve that satisfies the nonlinear system dynamics.

5.2.3. Discrete Projection Operator

A discrete version of the projection operator given in (2.17) can be defined to produce feasible trajectories from the infeasible trajectory produced during the descent calculations. The discrete projection operator uses a discrete-time stabilizing feedback law to take a feasible or infeasible trajectory, defined by $\xi_k = (\bar{\alpha}_k, \bar{\mu}_k)$, and maps it to a feasible trajectory, $\eta_k = (\bar{x}_k, \bar{u}_k)$. Proofs of the discrete projection operator were carried out by Johnson [**33**].

Using the discrete form of the projection operator with the appended sensitivity dynamics results in the following

$$P(\xi_k) : \begin{cases} \bar{u}_k = \bar{\mu}_k - K_k(\bar{x}_k - \bar{\alpha}_k) \\ x_{k+1} = \text{solution to (2.22)} \\ \psi_{k+1} = \text{equation (5.1)}. \end{cases}$$

5.2.4. Optimization Algorithm

Algorithm 5.1 reflects the updated notation of the iterative method using a gradient descent approach to solve the discrete optimization problem. Each iteration requires a descent direction, $\zeta_k = (\bar{z}_k, \bar{v}_k)$ to be computed from the following equation [**32**]:

$$\zeta_k = \arg\min_{\zeta_k} DJ(P(\xi_k)) \circ \zeta_k + \frac{1}{2}\langle \zeta_k, \zeta_k \rangle, \tag{5.3}$$

such that

$$\bar{z}_{k+1} = \bar{A}_k \bar{z}_k + \bar{B}_k \bar{v}_k,$$

where $\zeta_k \in T_{\eta_k}\mathcal{T}$, i.e., the descent direction for each iteration lies in the tangent space of the trajectory manifold at η_k. The components of the descent direction $\zeta_k = (\bar{z}_k, \bar{v}_k)$ are defined by $\bar{z}_k$, the perturbation to the extended state, and $\bar{v}_k$, the perturbation to the control. The quantity $\langle \zeta_k, \zeta_k \rangle$ is a local quadratic model computed as an inner product of ζ_k. Matrices $\bar{A}_k$ and $\bar{B}_k$ are the linearizations of the extended state dynamics. The descent direction (5.3) can be computed by solving an LQ regulation problem which is detailed in Appendix B.2.

Algorithm 5.1 Structured Trajectory Optimization

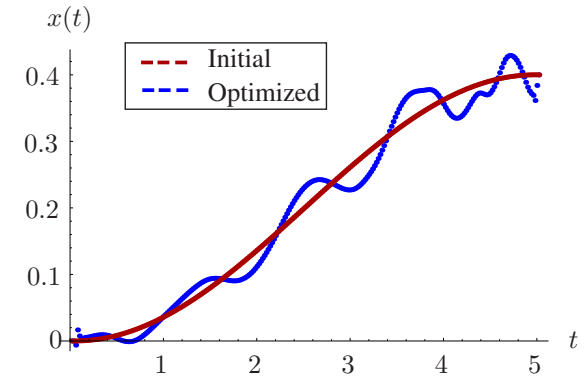
Initialize $\eta_k^0 \in \mathcal{T}$, tolerance ϵ, $i = 0$
while $DJ(\eta_k^i) \circ \zeta^i > \epsilon$ **do**
  Calculate descent, ζ^i:
    $\zeta^i = \arg\min_{\zeta^i} DJ(P(\xi^i)) \circ \zeta^i + \frac{1}{2}\langle \zeta^i, \zeta^i \rangle$
      minimizing (2.19) to compute ζ^i
  Compute γ^i with Armijo backtracking search
  Calculate the infeasible step:
    $\xi_k^{i+1}(t) = \eta_k^i + \gamma^i \zeta_k^i$
  Project trajectory onto dynamics constraints:
    $\eta_k^{i+1} = P(\xi_k^{i+1})$
  $i = i + 1$

Given the descent direction ζ^i, a backtracking line-search of the projection, $P(\eta^i + \gamma^i \zeta^i)$, provides a feasible trajectory assuming the step size γ^i satisfies the Armijo sufficient decrease condition [**27**]. Iterations upon the feasible trajectories continue until a given termination criteria is achieved.

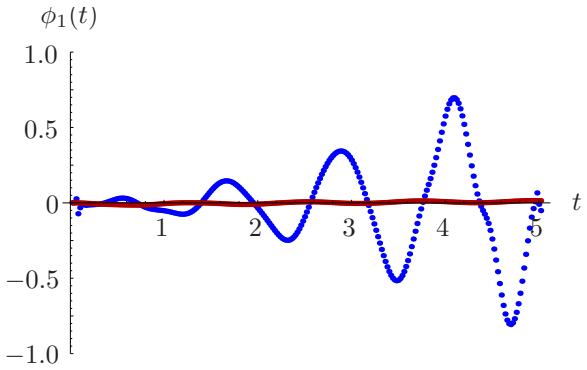
5.3. Discrete Information Optimization Results

To examine the results of the discrete-time formulation, the algorithm is tested in simulation on the 2-link cart-pendulum robot from Section 4.3. The robot has two dynamic configuration variables, $q = (\phi_1(t_k), \phi_2(t_k))$, and one kinematic configuration variable, $\rho = x(t_k)$, where $x(t_k)$ is the horizontal displacement of the robot, and $\phi_i(t_k)$ is the rotational angle of each link as shown in Fig. 4.1.

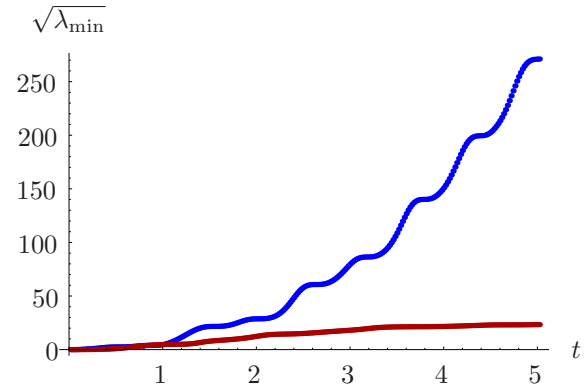
The input to the robot for the discrete-time experiment is the position $x(t_k)$ which is treated as a kinematic input. The robot can move in one dimension with positive motion to the right. Rotational friction is modeled at each pendulum joint, but the joints remain



(a) Robot position, $x(t)$



(b) Link 1 angle, ϕ_1



(c) Square root of the minimum eigenvalue of the Fisher information matrix $I(\theta)$ over time.

Figure 5.1. Comparisons of the trajectory before and after Fisher information optimization.

unactuated. For comparison to the continuous-time results, the goal of the optimization algorithm and the simulated model remains the same as in Section 4.3 - estimating the mass of the top pendulum link m_1 and the damping coefficient c of the joints.

5.3.1. Discrete-time Model Approximation

An approximation of the continuous Lagrangian in discrete time is found using (2.23). The same values for model parameters are used as defined in Section 4.3. We will assume that sensor measurements occur at 50 Hz, which will also be used as the simulation time step, dt. In general, measurements may occur at a lower frequency than the simulation time-step. In that case, Σ_k^{-1} should be set to zero for all time steps k in which measurements do not occur.

5.3.2. Optimization Results

The optimization algorithm was run until a convergence criterion of $|DJ(\xi_k) \circ \zeta| < 10^{-2}$ was satisfied, starting from an initial cost of 23.0. A comparison of initial and optimized trajectories can be seen in Fig. 5.1a-b. The optimized trajectory improves the Fisher information of the measured trajectory by a factor of 10^2. These results can be seen in Table 5.1. Additionally, the Fisher information gained over time is shown in Fig. 5.1c. The optimized trajectory greatly exceeds the rate of information gained compared to the initial trajectory. Using the Cramer-Rao bound [**22**], error estimates in the parameter set will decrease up to a factor of 10^2 due to this increase in Fisher information.

One motivation for creating a discrete-time version of the information maximization algorithm is to reduce computation time compared to the continuous version. Since the

Table 5.1. Optimization Results

	λ_1	λ_2	J
Initial:	5.95×10^3	543.2	23.0
Optimized:	6.91×10^6	7.34×10^4	9.0
- - - - - -	- -		
	Fisher Information Matrix		
Initial:	$\begin{bmatrix} 5.94 \times 10^3 & 98.6 \\ 98.6 & 544.9 \end{bmatrix}$		
Optimized:	$\begin{bmatrix} 6.90 \times 10^6 & 2.79 \times 10^3 \\ 2.79 \times 10^3 & 8.48 \times 10^4 \end{bmatrix}$		

Table 5.2. Algorithm Execution Time

	Discrete-Time	Continuous-Time
Avg. Time per iteration:	39.9 s	762.5 s
Number of Iterations:	2	36
Min. Eigenvalue Improvement:	factor of 135	factor of 110

structured integration allows for stable results at large time steps, such as 0.02 s used for this results, the computation time for each iteration is reduced by a factor of 19 as shown in Table 5.2.

While these results show the discrete-time algorithm converging in fewer iterations, we do not expect this trend to be generally true. In this case, the numerical differences happen to produce a slightly more efficient descent direction. However, even on a per iteration basis, the improvement in computation time is significant. Iterations of the continuous and discrete algorithms were timed in Mathematica on the same 3.0 GHz Intel i7 machine. Both algorithms return similar optimized costs, resulting in expected information improvements of 10^2.

Another concern may be that the resulting trajectory of the discrete algorithm does not precisely track the continuous trajectory. While the example system and model parameters are the same for both sets of results, we have incorporated the notion of kinematic control inputs into the discrete algorithm. This results in a different set of states compared to the continuous algorithm and different weights, Q_τ and R_τ. Additionally, the formulation in discrete time results in discrete momenta states p_k rather than velocity states. This also modifies the local quadratic approximation of the cost function, J_τ. Finally, the discrete algorithm uses the exact Fisher information, which slightly modifies the cost compared to the continuous approximation. Therefore, we expect that the algorithms will both achieve the goal of maximizing Fisher information but under slightly different objectives due to these differences which results in the varying trajectories.

5.3.3. Conclusion

This chapter presented a discrete-time method to automatically maximizing the Fisher information with respect to a set of model parameters by optimizing the robot's trajectory. We compare the method and results to the continuous-time analog from Section 4.2. In both cases, the algorithms significantly result in higher information which improves the estimation of uncertain parameters within the system model. The discrete formulation presented in the paper results in a factor of 19 improvement in the computation speed per iteration over the continuous algorithm while maintaining similar performance on the objective function. The algorithm also incorporates the actual Fisher information given a discrete set of measurements along the robot's trajectory rather than the approximation used in the continuous-time algorithm.

While computational speed has been significantly improved, expanding the dimensionality of the system still poses a formidable problem in terms of computation time. While the algorithm applies to larger systems, the need for extended state representations significantly affects general scalability. In order to allow for use in a learning type algorithm where iterations can be performed on-line as information is gained, a different method of optimal control may be required. The following chapter investigates information optimization using a real-time control algorithm to create an on-line learning method with an experiment using the Baxter Research Robot.

CHAPTER 6

Real-time Information Optimization for Learning

A fundamental goal of artificial intelligence and learning in robotics is to provide the means for a robot to automatically synthesize actions which improve estimates of the robot's internal dynamics and dynamical interactions with real-world objects. From an early age, humans are able to constantly learn and improve upon their motor skills and interactions with objects in the physical world. We aim to provide this form of learning on robots using real-time processing of feedback from active exploration of the environment.

Since the general problem of model synthesis and learning remains a formidable one, we restrict ourselves in this paper to creating a method for real-time active synthesis of dynamic trajectories to estimate a single model parameter in a known dynamical model.

The algorithms in Chapters 4 and 5 have demonstrated trajectory optimization algorithms for information maximization on continuous and discrete systems. The algorithms successfully improve the expected information by perturbing the trajectories using first-order optimization techniques. While effective, these algorithms remains too computationally expensive for real-time application.

Sequential Action Control (SAC), the model-based control approach introduced in Section 2.3.2, has shown promise in simulation as a closed-loop receding-horizon style

controller that can compute optimal actions in real-time for nonlinear systems, similar to a nonlinear model predictive controller [**88**, **89**]. Results have shown that the algorithm can outperform off-line trajectory optimization techniques on a number of canonical examples. The implementation in this chapter encompasses one of the first practical demonstrations of the algorithm on a robotic platform, including coupling of the algorithm with a nonlinear state observer as described in Section 6.2.2.3.

This chapter presents the experimental trial of the SAC algorithm coupled with an on-line estimator for real-time active estimation of a dynamical system using a Baxter Research Robot. The Baxter Robot has been used as a practical platform for a number of studies [**90**–**92**] while also presenting a number of challenges including high compliance and actuator saturation. Despite these potential sources of unmodeled dynamic effects, results show that the estimator successfully converges to the actual parameter value across several trials. Section 6.1 provides an overview of the trajectory synthesis and estimation algorithms. The specific implementation details for Baxter and the suspended load estimation task are provided in Section 6.2, and Section 6.3 provides results from several trials of the real-time estimation task.

6.1. Real-time Control Structure

This section presents a detailed overview of the active trajectory synthesis and estimation problem which is implemented in Section 6.2. As shown in Fig. 6.1, there are two primary modules interacting with the robot hardware: the SAC controller for trajectory synthesis and the nonlinear least-squares estimator. At the highest level, the least-squares estimator requires the control inputs provided by the SAC controller to compute a predicted output

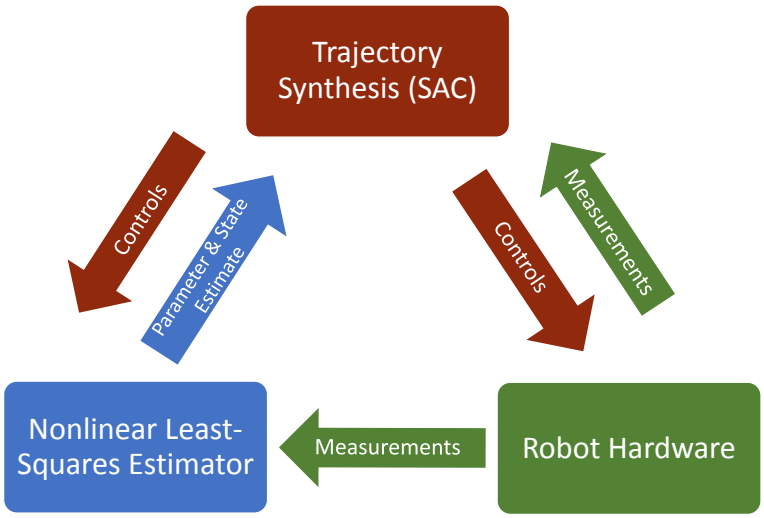


Figure 6.1. Overview of the real-time control structure.

which is compared to the actual measurements provided by the robot. The SAC controller is updated with new parameter estimates and state estimates by the least-squares estimator and optionally can receive state feedback directly from the robot hardware if available. These modules can be run asynchronously and at different rates with the use of a nonlinear state observer model.

For this implementation, it is assumed that one parameter is uncertain with additive noise on observer measurements but negligible process noise. For SAC, the state is usually derived assuming control-affine dynamics as opposed to the general nonlinear dynamics model presented in Section 2.2.1 [**18**]. Thus, the model of the system is defined as

$$\dot{x} = f(x,\theta) + h(x,\theta)u \tag{6.1}$$

$$y = g(x,u,\theta) + \mathrm{w}_y$$

where $x \in \mathbb{R}^n$ defines the system states, $\tilde{y} \in \mathbb{R}^h$ defines the measured outputs, $u \in \mathbb{R}^m$ defines the inputs to the system, $\theta \in \mathbb{R}$ defines the parameter to be estimated, and w_y is additive output noise where $p(\mathrm{w}_y) = N(0, \Sigma)$.

6.1.1. Nonlinear Least-Squares Estimator

While the robot is executing a motion, a nonlinear least-squares estimator is used on-line to update the estimated value of the parameter as well as the robot state. The gradient descent version of the estimator is used as presented in Section 2.2.1.

Since the estimator requires a predicted output, $\tilde{y}(t)$ which is based on current estimates of θ and $x(t)$, a nonlinear state observer is also required for systems without full-state feedback. While any numerical differential equation solver may be sufficient, for this implementation, we are using the `trep` software package available at `http://nxr.northwestern.edu/trep`.

6.1.2. Sequential Action Control

The SAC control process is detailed in Section 2.3.2. The implementation in this chapter requires SAC to include a cost on the Fisher information of the uncertain parameter. For the Baxter experiment, we assume that the measurement noise of the system is normally distributed with zero process noise. Therefore the Fisher information is given by

$$I(\theta) = \sum_{k=k_0}^{k_f} \Gamma_\theta(t_k) \cdot \Sigma^{-1} \cdot \Gamma_\theta(t_k), \tag{6.2}$$

which is equivalent to the information metric (4.1) presented in the previous optimization algorithms. As detailed in Chapter 4, a cost function on Fisher information from (4.1)

also requires the simulation of the gradient of x w.r.t. θ, i.e. $\psi(t) = \frac{d}{d\theta}x$. These additional states are referred to in the paper as extended state dynamics and notated as $\bar{x} = (x, \psi)$.

The SAC cost function in (2.20) is modified to maximize information with the following form

$$J_{cost} = \int_{t_0}^{t_f} l(\bar{x}(t))\, dt + s(x(t_f)), \tag{6.3}$$

where

$$l(\bar{x}(t)) = \left[\Gamma_\theta(t) \cdot \Sigma^{-1} \cdot \Gamma_\theta(t)\right]^{-1} + x(t)^T \cdot Q_\tau \cdot x(t),$$

i.e., the running cost involves minimization of the inverse of the information and an optional trajectory tracking cost to bias the system toward a particular part of the state space. Additionally, an extended adjoint system $\bar{\rho}$ is computed using the extended state $\bar{x}$ in the same manner as presented in Section 2.3.2.

6.2. Baxter Implementation

This section describes both the problem formulation and software specific implementation of the control structure described in the previous section on the Baxter Research Robot created by Rethink Robotics shown in Fig. 6.2 [**93**]. Results from experimental trials are presented in Section 6.3.

6.2.1. Problem Description and Model Formulation

To test this control paradigm in real-time on a practical robotic system, we chose the task of determining the string length of a suspended payload from one of Baxter's grippers using a load cell as the sole output to the estimator. This configuration necessitates

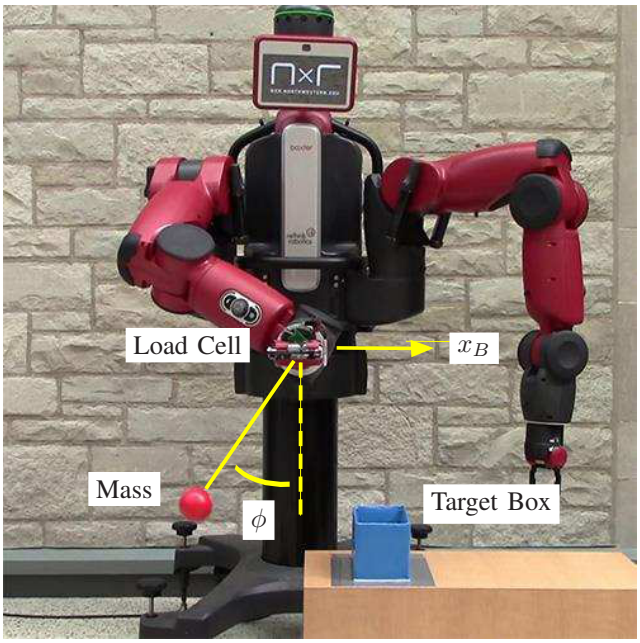


Figure 6.2. Baxter performing real-time active parameter estimation on the length of a suspended payload.

active motion to estimate the string length as the Fisher information is zero when the robot is stationary. To simplify the control of Baxter, we chose to approximate the end effector as kinematic and control motion only in the Cartesian x-axis, x_B. The equations of motion for the system are given by the following,

$$f(x, u, \theta) = \begin{bmatrix} \dot{x}_B \\ u \\ \dot{\phi} \\ \frac{u}{\ell}\cos\phi - \frac{g}{\ell}\sin\phi \end{bmatrix}$$

where u is the x-axis acceleration of the gripper, m is the known mass suspended from the robot, g is the gravitational constant, and ℓ is the length of the string, which will be

estimated. Additionally, the equation for the force output F_s is

$$\tilde{y}(x, u, \theta) = F_s = mg\cos\phi - m\ell\dot{\phi}^2 - u\sin\phi.$$

It is assumed that the trajectories will maintain tension in the string; therefore, the distance between the robot and mass is fixed. Given this system model, the extended state $\bar{x} \in \mathbb{R}^8$.

6.2.2. Experimental Implementation

As shown in Fig. 6.1, there are essentially two separate modules interacting with the robot hardware: the SAC trajectory synthesis module, and the nonlinear least-squares estimator module. In this section we describe how each of these modules is implemented for the Baxter experiments.

The backbone of communication between modules and the robot is provided by the Robot Operating System (ROS) [**84**]. ROS provides the ability to asynchronously run the control and estimation modules, implemented as ROS nodes, and Baxter natively uses ROS as the primary API for motion commands. Three primary nodes are employed: the SAC control node, a measurement receiver node, and the estimator node.

6.2.2.1. Baxter SAC Node. As mentioned in Section 6.2.1, the end effector of the Baxter robot is approximated as a kinematic input to the dynamic suspended mass system. The dynamic model for the suspended mass system assumes only planar motion, and the kinematic input moves perpendicular to gravity in this plane. It follows that the SAC module produces target locations along a one-dimensional line that Baxter's end effector should follow. A joint velocity controller for Baxter's right arm is used to stabilize the

right end effector to these target locations. To avoid issues with kinematic singularities in the manipulator while controlling to these target locations, this one-dimensional line is expanded to a candidate set of closely-spaced end effector targets in $SE(3)$. An off-line computation is done to solve the inverse kinematics problem for each of the targets in the set producing a set of target joint angles for each target. These joint angle targets are then stored to disk as a lookup table between target horizontal positions, like those produced by SAC, and target joint angles. During an experimental run, the desired joint position, velocity, and acceleration data is sent using a Joint Trajectory Publisher to Baxter's internal controller which runs a high frequency real-time loop to control each of the joints.

6.2.2.2. Baxter Measurement Receiver Node. The payload is attached to Baxter's end effector through a one-dimensional load cell. A microcontroller samples the load cell at 100 Hz and transmits the measured forces via a serial link back to a client computer which is communicating with Baxter over an Ethernet connection. These force measurements represent the $\tilde{y}(t_i)$ in (2.4). The load cell has been calibrated prior to the experiment to convert the load cell output to a force (N). The resulting force is timestamped as it is received and published at 100 Hz for use by the estimator node.

6.2.2.3. Baxter Estimator Node. The estimator node subscribes to the actual end effector trajectory which is provided by the Baxter API. These measurements are used to generate the $\tilde{y}(t_i)$ terms in (2.4) through the use of a nonlinear state observer. As mentioned in Section 6.1.1, the `trep` software package is used as the state observer. The trajectory is used as the input and `trep` provides the predicted state evolution of the suspended load. From these states, the predicted force $y(t)$ can be calculated.

At a frequency of 2 Hz, the estimation module solves the optimization problem described by (2.3), updating the estimate of the string length. This frequency was chosen as a conservative rate at which the estimator has enough time to provide an update; however, the rate can be modified depending on the required computational time for the particular system. This parameter estimate and new state estimate are provided as a service to the SAC node which queries the service at the start of each computation. Updates to the string length estimate and expected state are reflected in this service call.

6.3. Experimental Results using Baxter

This section presents the results of real-time experimental trials of the algorithm using the Baxter robot. A total of 9 trials are presented with initial estimates of the length parameter, ℓ that varied from 0.308 m to 0.468 m with the actual string length at 0.368 m. The tests were conducted for a total of 6 seconds for each trial. For each trial, the initial position of the gripper was set to $x_B = -0.25$ m and the suspended load was hanging in a stationary position.

The convergence of the parameter estimates over time for all 9 trials is shown in Fig. 6.3. The estimates clearly converge toward the actual string length which is noted by the dashed horizontal line. Qualitatively, the rate of convergence across all the trials is similar, with estimates beginning to converge around 2 to 4 seconds. This suggests that for this system, the trajectories generated by the SAC algorithm provide relatively similar levels of information despite the initial estimate. Since the estimator cost may not be convex, initial estimates too far from the actual value may not converge to the real value given multiple local minima. The mean estimate of ℓ from the 9 trials after 6

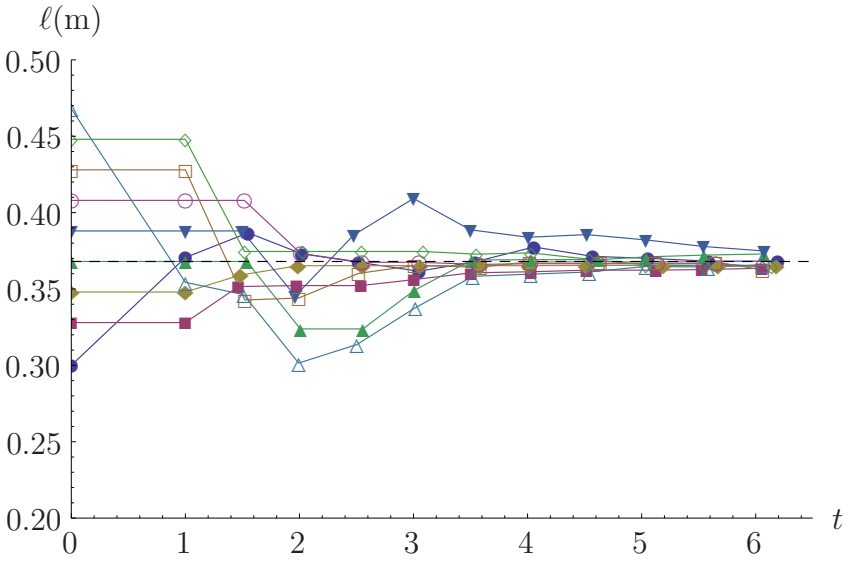


Figure 6.3. Experimental trials on Baxter using different initial estimates of the string length. The dashed line indicates the actual measured length.

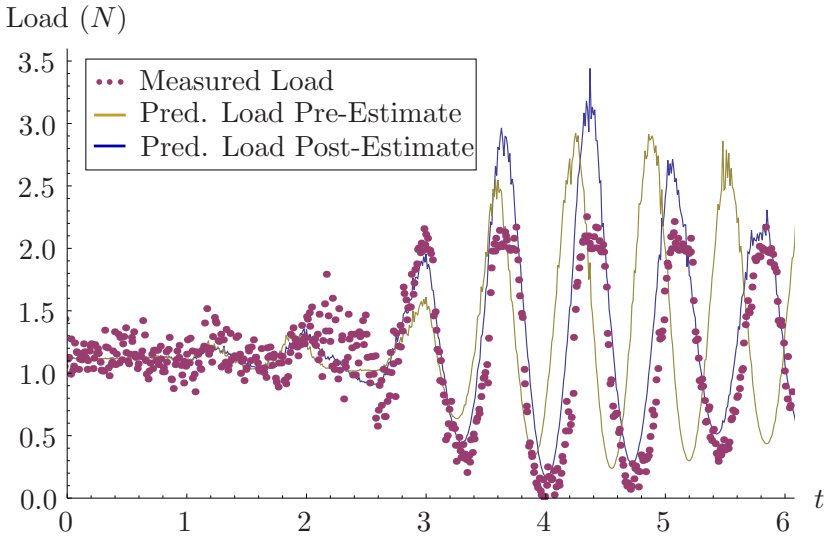


Figure 6.4. Measured and predicted force from one Baxter trial over time for $\ell_0 = 0.35$ m.

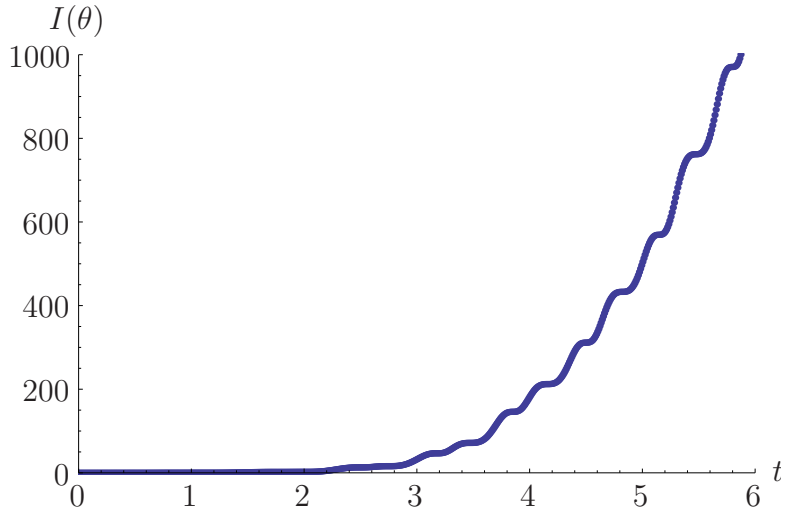
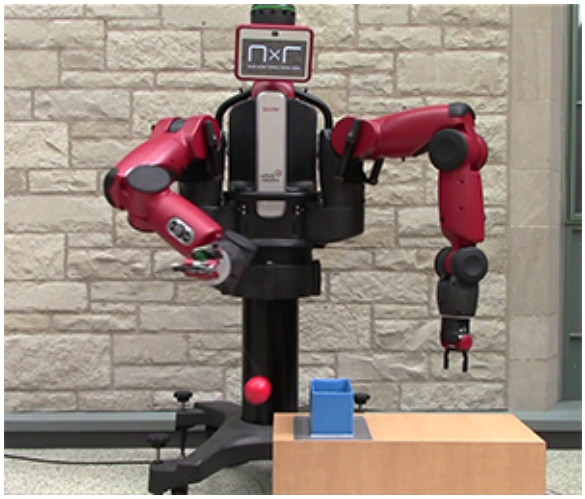


Figure 6.5. Expected information accrued over time during one Baxter trial for $\ell_0 = 0.35$ m.

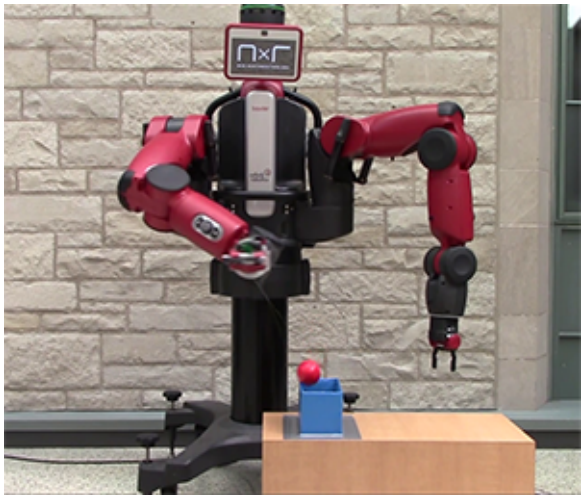
seconds was 0.367 m with the actual string length set to 0.368 m. The standard deviation of the final estimates is 0.0042 m.

The measured and predicted force data is shown from a single trial in Fig. 6.4. The figure shows that as the suspended load begins to react to the control inputs, providing a much better signal for the estimator to track, the estimates begin to converge. It is also noted in this figure that the measurements taken by the load cell begin to saturate the hardware setup around 2.5 N, however, the estimator is still able track the phase of the system despite this saturation.

Additionally, Fig. 6.5 shows the expected information accrued over time from one experimental trial. A large increase in information begins to accrue as the suspended load is excited, leading to the convergence of the parameter estimate. This information provides a best-case bound on the parameter estimate through the Cramer-Rao bound [**22**]. With the estimator now validated, the following section examines the impact of the parameter



(a) Incorrect parameter plan failing to swing the mass into the box.



(b) Correct parameter plan successfully swinging the mass into the box.

Figure 6.6. Baxter near the completion of the dynamic task for both incorrect and correct parameter plans.

estimates on the open-loop execution of a dynamic trajectory involving the Baxter with a suspended load.

6.3.1. Impact on Trajectory Planning and Execution

To demonstrate the effect of incorrect parameter estimates during the execution of an actual task, an experiment is conducted where Baxter must swing a suspended ball into a nearby box. In order to force the dynamics of the system to be exploited, Baxter's range of motion is constrained in software to only provide motion in the horizontal x-axis. As shown in Fig. 6.6, the suspended ball must be swung in a dynamic motion in order to clear the edge of the box.

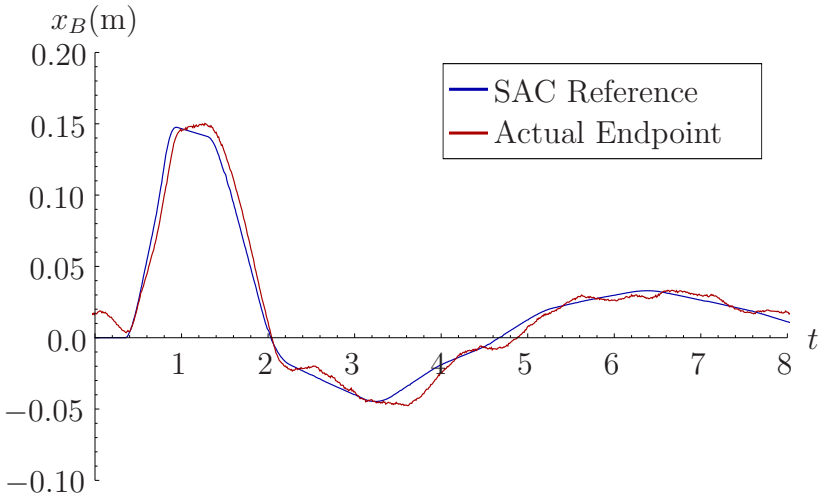


Figure 6.7. Endpoint reference compared to actual endpoint position of the Baxter trial with $\ell_0 = 0.45$ m.

The process for estimation of the string length is completed in an identical fashion to the prior experiment. The SAC controller is executed in real-time to provide a perturbation to the system which then provides length estimates in real-time from the collected force data. This estimation is executed for 8 seconds from an initial length estimate of 0.45 m. The actual string length is 0.368 m. During the trial, the estimator converges on a refined estimate of 0.370 m which is then used to plan the task trajectory.

The generated and executed trajectories for the estimation trial are shown in Fig. 6.7. As discussed in Section 6.2, the motion of the gripper is controlled to move along the Cartesian x-axis to follow the generated reference. The estimator is given the actual endpoint position which allows for accurate prediction of the load, and SAC receives the estimated state from the estimator for trajectory synthesis.

Once the uncertain parameter has been accurately identified using the SAC active estimator, the task trajectory must be synthesized. At the end of the estimation process,

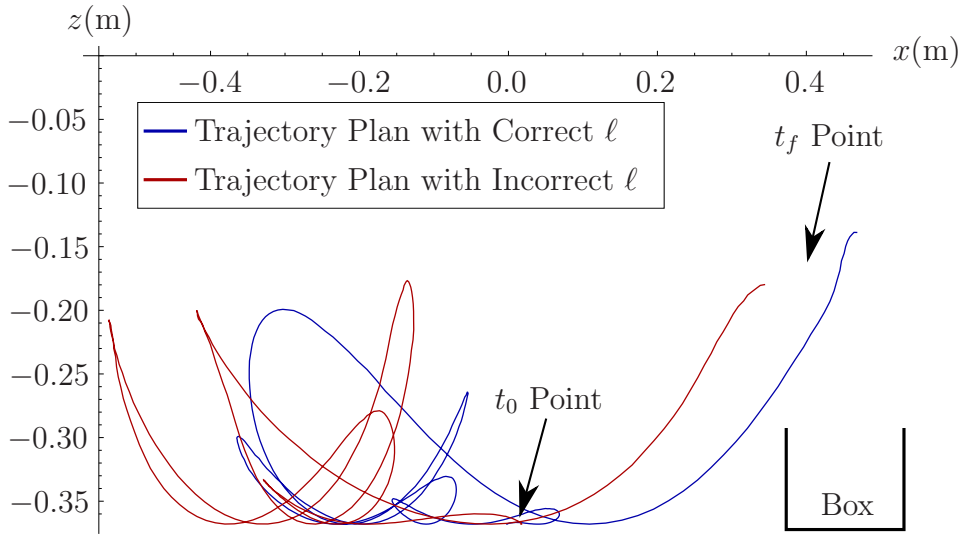


Figure 6.8. Simulated trajectory plans for the suspended mass with the correct length at 0.37 m and incorrect length at 0.45 m.

Baxter's end effector is returned to the zero position and the motion of the mass is allowed to naturally dampen while the task trajectory computation completes.

The selection of an appropriate optimal control method is dependent both on the computational requirements and level of dynamic motion in the task. Sequential Action Control was selected for the on-line estimator since real-time performance was desired. However, the SAC algorithm relies on a predetermined set of dynamics for the future time horizon when computing an optimal action as opposed to optimizing a control curve over the entire horizon. For the estimator, this is acceptable since the model parameters are incorrect, and future controls are largely irrelevant relative to the measurement feedback. Once the parameters are known, a complete trajectory optimization method can then be used.

Table 6.1. Baxter Experimental Task Results

Initial Length Estimate (m):	Without SAC Estimation:	With SAC Estimation:
0.308	Fail	Success
0.328	Fail	Success
0.348	Fail	Success
0.368*	Success	Success
0.388	Success	Success
0.408	Fail	Success
0.428	Fail	Success
0.448	Fail	Success
0.468	Fail	Success

*actual string length

Planning for the task trajectory is completed using the `trep` software package. The software contains the original discrete trajectory optimization algorithm, which the extension in Chapter 5 is based on. For this example, a box is placed 0.45m to the left of the suspended mass. The dynamic task is to swing the mass over the box so that it lands inside the box. Two trajectories are then calculated. The first is using the incorrect initial estimate of the string length at 0.45 m. The second is the improved estimate from the SAC controller at 0.37 m. Figure 6.8 shows the predicted path of the ball using the actual string length for both plans. As shown in the figure, the trajectory using the incorrect estimate falls short of the box opening.

A total of 18 trials were run on the Baxter robot to evaluate the task performance using active estimation. Using the same distribution of 9 initial length estimates from Fig. 6.3, the task was at each value with SAC estimation and directly using the incorrect estimate. If the suspended mass landed inside the box at the end of the trial, the trial is considered a success. Any trial with the mass outside of the box at the end of the trial resulted in failure.

The results of the trials are shown in Table 6.1. Without estimation, the task is successful with only two length values - the actual value of 0.368m and a slightly longer length of 0.388m. With SAC estimation the task is successfully completed from each initial length estimate for all 9 trials.

The results demonstrate that with the open-loop trajectory plans, it is important to have good estimates of the model parameters for model-based control. Using least-squares estimation techniques along with the SAC algorithm can provide an automated real-time learning method for parameter estimation for use in control of dynamical systems.

CHAPTER 7

Conclusion

Models are continuing to become more important as the complexity of the systems to control increases. Better controllers can be developed for these systems given accurate mathematical models of the system dynamics. While people develop the skills to refine models over long periods of time, it is advantageous for a robot to have the ability to quickly synthesize appropriate trajectories for estimation of parameters within these models. To this end, this thesis has demonstrated several algorithms using local optimization techniques to enable robots to better estimate parameters for dynamical systems.

7.1. Summary of Contributions

The focus of this thesis has been motivated by the biomechanical case-study in Chapter 3. The results from the study showed the benefit of using structured numerical techniques for simulation and optimal control to improve stability and computational performance in the biomechanical system. Furthermore, the study showed the impact of variations in important parameters such as the wrapping radius on the simulated trajectory.

Taking these results into account, the continuous version of the projection-based information optimization algorithm was developed in Chapter 4. The goal of the algorithm was

to maximize the minimum eigenvalue of the Fisher information matrix as defined by the E-optimality criterion. An experiment was performed using a cart and double pendulum system with two uncertain parameters. Using the optimization algorithm, the information metric increased by a factor of 10^3. Experimentally, the precision of the parameter estimate improved by an order of magnitude. The conditioning of the least-squares estimator was also studied in Chapter 4. A cost function on the condition number of the estimator Hessian resulted in a similar form of the optimization algorithm. Using the algorithm with a cart-pendulum system, the condition number was optimized and the convergence rate improved by two orders of magnitude.

While the continuous algorithm provided good results, the computational load was high as demonstrated in the biomechanical case-study. To improve the computational performance, the algorithm was derived in Chapter 5 using the structure preserving discrete mechanics techniques. The discrete algorithm was tested in simulation on the same cart and double-pendulum system as the continuous information maximization system. The result was an algorithm which provided a similar increase in the information metric with a factor of 19 reduction in the computation time per iteration.

In order to achieve real-time performance from an information-based algorithm, a different trajectory optimization algorithm was used in Chapter 6. The Sequential Action Control algorithm provided real-time computation of optimal control actions which could be modified to use an information-based cost function. The result was tested on the Baxter Research Robot with a load suspended from the manipulator endpoint. In real-time, a least-squares estimator was run on the uncertain length of the string suspending the load while a trajectory was synthesized. In 9 trials with different initial estimates of

the length, the estimator successfully converged to within 1% of the actual string length. Using this result, a dynamic task was run in which Baxter must swing the suspended load into a nearby box. Using the projection-based optimization algorithm on the initial and estimated values of the length parameter resulted in successful task completion using the estimated length of 0.37 m. The task was not successful with the initial length estimate of 0.45 m.

7.2. Selecting a Method

Throughout this thesis, a variety of cost functions and numerical methods have been implemented. While the motivation for each algorithmic choice is detailed in the chapters, a high-level view of the relationship between algorithms is useful when selecting an appropriate method for a particular application. The common thread through all of the methods is a desire to synthesize trajectories that have some impact on an information metric. The algorithmic design choices can be separated into two parts - the objective and the numerical method.

The first consideration is the objective. If there is only a single uncertain parameter within the model, information maximization as detailed in Chapter 4 is the most desirable metric. Conditioning optimization, presented later in the chapter is only applicable when there are multiple parameters to be estimated. If the application requires estimation of multiple parameters, the choice then depends on the estimation method that will be used. Second-order Hessian based estimators reduce the need for conditioning of the problem since the curvature of the solution is taken into account. However the computational

effort is greater than with a simple gradient descent estimator which would require a well-conditioned problem. The desire for both multiple parameter estimates combined with a gradient descent estimator requires use of a conditioning objective for estimation.

It is also important to note that the two primary objective functions presented in this thesis, information maximization and conditioning, are purely exploitative cost functions. For the scope of this work, an exploitative algorithm refers to one that relies only on the current best estimate of information and parameters to compute future iterations. In each of these algorithms, the current best estimate of the parameter set is used at each iteration. For applications which may have bimodal or more complex parametric belief maps, the use of an explorative cost may also be appropriate. Instead of relying solely on the current best estimate, an explorative cost may use a distribution of current estimates or a sampling method to find a more global solution. These objectives are commonly seen in simultaneous localization and mapping (SLAM) work [**94**–**96**] and ergodic exploration algorithms [**97**,**98**]. Extension of the algorithms presented in this thesis to an exploration application may be an interesting area of future work.

Following the selection of objective, the numerical method must be considered. Using a continuous-time method such as the projection-based trajectory optimization algorithm in Chapter 4 results in a well-defined optimization problem over an entire time-series of controls. This is crucial when a task requires fine control in the state space over time, typically requiring the optimization of the complete control sequence in time. The continuous nature is important when the time-scales may not be readily known and adaptive time-stepping is needed.

If computation time becomes constrained, a solution can first be attempted using discrete mechanics theory, such as the method in Chapter 5 which still preserves the predictive nature of the trajectory optimization. In the case of tight constraints on computation, or real-time requirements, an optimal control method such as the Sequential Action Control algorithm in Chapter 6 may allow for fast computation at the expense of forward-looking control signals. The advantage of Sequential Action Control is that only the control value at the current time is optimized, resulting in much faster computation. This algorithm works well for the information maximization objective as the perturbations do not have strict constraints in the state-space over time. When a task has these strict time and state requirements, such as swinging a suspended payload into a box, optimization of a full time-series of controls may be required.

7.3. Future Research Directions

This work represents only the first step toward improving robot learning on physical systems by exploiting dynamic models. One possible improvement may include the use of Lie groups as the fundamental tool to build the dynamic models. While the models can be more complicated to create, they can facilitate better-posed solutions to the optimization problems, reducing some of the singularities and angle wrapping issues that occur when modeling certain robotic systems, especially non-holonomic systems.

Additionally, the extension to multiple parameters only requires that an optimality metric is set; however, it would be useful for a system to realize which parameters need to be estimated in the first place. This could be achieved through forms of sensor fusion and covariance estimation as well as an exploration-based search algorithms. Eventually,

the addition of a model creation and learning algorithm would enable a robot to develop not only estimates of the parameters, but also internal structure without prior knowledge of the internal dynamic model.

Taking this idea a step further leads to the problem of "*unknown* unknown" parameters. The knowledge of model topology is needed in this thesis to estimate the unknown values of the parameters; however, in some cases, the topology itself may be uncertain. A promising direction for further work is the extension of the estimation to hybrid systems with varying topologies. A tractable problem may include a robot arm with varying numbers of joints. In this case, the estimator would function over all hybrid configurations of the arm to generate a trajectory which excites the system in a way that can best identify the configuration mode.

This manner of estimation will pose a number of problems as well, primarily stemming from the explosion of complexity with larger mode sets. Avoiding the combinatorial complexity that is common in hybrid system optimization is critical to creating an algorithm which can perform well on reasonable time scales. Similar projection based trajectory optimization algorithms have been applied to mode scheduling and may be useful in tackling the mode estimation problem in the future [**99**].

While the formulation of the algorithm allows for general, nonlinear dynamics, limitations on tractable noise models remains a formidable challenge. Since process noise is assumed to be negligible for the purpose of this algorithm, results may only be useful on systems with accurate input models. Although the algorithm should improve prediction on most systems, predicted precision improvements in simulation may be overestimated compared to the experimental results due to unmodeled bias and system modeling errors.

Still, with appropriate selection of weighting matrices on controls and tracking error, improvement in estimator performance is expected compared to arbitrary choices of the experimental trajectory. The continued development of dynamics-based methods for robot learning will allow robots to learn and better interact with real-world objects and tasks in physical environments.

References

[1] D. M. MacKay, "Cerebral Organization and the Conscious Control of Action," in *Brain and Conscious Experience* (J. C. Eccles, ed.), pp. 422–445, Berlin, Heidelberg: Springer Berlin Heidelberg, 1965.

[2] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. New York: Wiley, 2006.

[3] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks.," *IEEE Transactions on Neural Networks*, vol. 1, pp. 4–27, Jan. 1990.

[4] L. Ljung, *System Identification: Theory for the User (2nd Edition)*. Prentice Hall, 1999.

[5] R. S. Pindyck and D. L. Rubinfeld, *Econometric models and economic forecasts*. Boston: Irwin/McGraw-Hill, 1998.

[6] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, pp. 1361–1379, Oct. 1996.

[7] J. E. Marsden and M. West, "Discrete mechanics and variational integrators," *Acta Numerica*, vol. 10, pp. 357–514, 2001.

[8] J. Schultz and T. D. Murphey, "Embedded control synthesis using one-step methods in discrete mechanics," in *2013 American Control Conference*, pp. 5293–5298, June 2013.

[9] J. Schultz and T. D. Murphey, "Extending filter performance through structured integration," in *2014 American Control Conference*, pp. 430–436, June 2014.

[10] B. Armstrong, "On Finding Exciting Trajectories for Identification Experiments Involving Systems with Nonlinear Dynamics," *The International Journal of Robotics Research*, vol. 8, pp. 28–48, Dec. 1989.

[11] A. A. Feldbaum, "Dual control theory. I," *Avtomat. i Telemekh*, vol. 21, no. 9, pp. 1240–1249, 1960.

[12] J. Rathouský and V. Havlena, "MPC-based approximate dual controller by information matrix maximization," *International Journal of Adaptive Control and Signal Processing*, vol. 27, pp. 974–999, Nov. 2013.

[13] A. D. Wilson, J. A. Schultz, and T. D. Murphey, "Trajectory Synthesis for Fisher Information Maximization," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1358–1370, 2014.

[14] A. D. Wilson and T. D. Murphey, "Local E-optimality Conditions for Trajectory Design to Estimate Parameters in Nonlinear Systems," in *2014 American Controls Conference*, 2014.

[15] A. D. Wilson, J. A. Schultz, and T. D. Murphey, "Trajectory Optimization for Well-Conditioned Parameter Estimation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, pp. 28–36, Jan. 2015.

[16] A. D. Wilson and T. D. Murphey, "Optimal Trajectory Design for Well-Conditioned Parameter Estimation," in *2013 IEEE International Conference on Automation Science and Engineering*, pp. 13–19, Aug. 2013.

[17] A. D. Wilson and T. D. Murphey, "Maximizing Fisher Information using Discrete Mechanics and Projection-based Trajectory Optimization," in *2015 IEEE International Conference on Robotics and Automation*, pp. 2403–2409, May 2015.

[18] A. Ansari and T. D. Murphey, "Sequential Action Control: Closed-Form Optimal Control for Nonlinear Systems," *IEEE Transactions on Robotics*.

[19] A. D. Wilson, J. A. Schultz, A. Ansari, and T. D. Murphey, "Real-time Trajectory Synthesis for Information Maximization using Sequential Action Control and Least-Squares Estimation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[20] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, July 1948.

[21] D. M. Magerman and M. P. Marcus, "Parsing a natural language using mutual information statistics," in *Proceedings of the eighth National conference on Artificial intelligence*, pp. 984–989, AAAI Press, July 1990.

[22] C. R. Rao and J. Wishart, "Minimum variance and the estimation of several parameters," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 43, p. 280, Oct. 1947.

[23] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. John Wiley & Sons, 2004.

[24] N. Gershenfeld, *The nature of mathematical modeling*. Cambridge University Press, Jan. 1999.

[25] N. Brunel and J.-P. Nadal, "Mutual Information, Fisher Information, and Population Coding," *Neural Computation*, vol. 10, pp. 1731–1757, Oct. 1998.

[26] T. Asnani, Himanshu and Venkat, Kartik and Weissman, "Relations Between Information and Estimation in the Presence of Feedback," in *Information and Control in Networks*, pp. 157–175, Springer International Publishing, 2014.

[27] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives.," *Pacific Journal of Mathematics*, vol. 16, no. 1, pp. 1–3, 1966.

[28] J. Nocedal and S. Wright, *Numerical Optimization*. Springer, 2006.

[29] D. Liberzon, *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, Dec. 2011.

[30] J. Hauser and D. Meyer, "The trajectory manifold of a nonlinear control system," in *Proceedings of the 37th IEEE Conference on Decision and Control*, vol. 1, pp. 1034–1039, 1998.

[31] E. Johnson, J. Schultz, and T. Murphey, "Structured Linearization of Discrete Mechanical Systems for Analysis and Optimal Control," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 140–152, 2015.

[32] J. Hauser, "A projection operator approach to the optimization of trajectory functionals," in *World Congress*, vol. 15, p. 310, July 2002.

[33] E. Johnson, *Trajectory Optimization and Regulation for Constrained Discrete Mechanical Systems*. PhD thesis, Northwestern University, 2012.

[34] A. Ansari and T. D. Murphey, "Control-On-Request: Short-Burst Assistive Control for Long Time Horizon Improvement," in *American Control Conference*, 2015.

[35] A. Lew, J. E. Marsden, M. Ortiz, and M. West, "Variational time integrators," *International Journal for Numerical Methods in Engineering*, vol. 60, pp. 153–212, May 2004.

[36] F. E. Zajac, "Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control.," *Critical reviews in biomedical engineering*, vol. 17, pp. 359–411, Jan. 1989.

[37] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen, "OpenSim: open-source software to create and analyze dynamic simulations of movement.," *IEEE Transactions on Biomedical Engineering*, vol. 54, pp. 1940–50, Nov. 2007.

[38] L. M. Schutte, *Using Musculoskeletal Models to Explore Strategies for Improving Performance in Electrical Stimulation-Induced Leg Cycle Ergometry*. PhD thesis, Stanford University, 1993.

[39] D. Staudenmann, J. R. Potvin, I. Kingma, D. F. Stegeman, and J. H. van Dieën, "Effects of EMG processing on biomechanical models of muscle joint systems: sensitivity of trunk muscle moments, spinal forces, and stability.," *Journal of biomechanics*, vol. 40, pp. 900–9, Jan. 2007.

[40] H. Garland, R. W. Angel, and R. D. Melen, "A state variable averaging filter for electromyogram processing," *Medical & Biological Engineering*, vol. 10, pp. 559–560, July 1972.

[41] M. Zecca, S. Micera, M. C. Carrozza, and P. Dario, "Control of Multifunctional Prosthetic Hands by Processing the Electromyographic Signal," *Critical Reviews in Biomedical Engineering*, vol. 30, no. 4-6, pp. 459–485, 2002.

[42] M. Ackermann and W. Schiehlen, "Physiological Methods to Solve the Force-Sharing Problem in Biomechanics," in *Multibody Dynamics*, pp. 1–23, Springer, 2009.

[43] K. R. Saul, X. Hu, C. M. Goehler, M. E. Vidt, M. Daly, A. Velisar, and W. M. Murray, "Benchmarking of dynamic simulation predictions in two software platforms using an upper limb musculoskeletal model.," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 18, pp. 1445–58, Jan. 2015.

[44] A. J. van den Bogert, D. Blana, and D. Heinrich, "Implicit methods for efficient musculoskeletal simulation and optimal control.," *Procedia IUTAM*, vol. 2, pp. 297–316, Jan. 2011.

[45] C. Y. Scovil and J. L. Ronsky, "Sensitivity of a Hill-based muscle model to perturbations in model parameters.," *Journal of Biomechanics*, vol. 39, pp. 2055–63, Jan. 2006.

[46] R. T. Raikova and H. T. Aladjov, "Comparison between two muscle models under dynamic conditions.," *Computers in biology and medicine*, vol. 35, pp. 373–87, June 2005.

[47] A. Kirk, J. O'Brien, and D. Forsyth, "Skeletal Parameter Estimation from Optical Motion Capture Data," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 782–788.

[48] D. G. Thelen, A. B. Schultz, S. D. Fassois, and J. A. Ashton-Miller, "Identification of dynamic myoelectric signal-to-force models during isometric lumbar muscle contractions," *Journal of Biomechanics*, vol. 27, pp. 907–919, July 1994.

[49] A. Erdemir, S. McLean, W. Herzog, and A. J. van den Bogert, "Model-based estimation of muscle forces exerted during movements.," *Clinical Biomechanics*, vol. 22, pp. 131–54, Feb. 2007.

[50] D. Faller, U. Klingmuller, and J. Timmer, "Simulation Methods for Optimal Experimental Design in Systems Biology," *SIMULATION*, vol. 79, pp. 717–725, Dec. 2003.

[51] M. Baltes, R. Schneider, C. Sturm, and M. Reuss, "Optimal experimental design for parameter estimation in unstructured growth models," *Biotechnology Progress*, vol. 10, pp. 480–488, Sept. 1994.

[52] P. Felix Oliver Lindner and B. Hitzmann, "Experimental design for optimal parameter estimation of an enzyme kinetic process based on the analysis of the Fisher information matrix," *Journal of Theoretical Biology*, vol. 238, no. 1, pp. 111–123, 2006.

[53] G. Franceschini and S. Macchietto, "Model-based design of experiments for parameter precision: State of the art," *Chemical Engineering Science*, vol. 63, pp. 4846–4872, Oct. 2008.

[54] C. Jauberthie, L. Denis-Vidal, P. Coton, and G. Joly-Blanchard, "An optimal input design procedure," *Automatica*, vol. 42, no. 5, pp. 881–884, 2006.

[55] F. Dietrich, A. Raatz, and J. Hesselbach, "A-priori Fisher information of nonlinear state space models for experiment design," in *2010 IEEE International Conference on Robotics and Automation*, pp. 3698–3702, May 2010.

[56] J. Martensson, C. R. Rojas, and H. Hjalmarsson, "Conditions when minimum variance control is the optimal experiment for identifying a minimum variance controller," *Automatica*, vol. 47, pp. 578–583, Mar. 2011.

[57] S. Joshi and S. Boyd, "Sensor Selection via Convex Optimization," *IEEE Transactions on Signal Processing*, vol. 57, pp. 451–462, Feb. 2009.

[58] M. Gevers, X. Bombois, R. Hildebrand, and G. Solari, "Optimal Experiment Design for Open and Closed-loop System Identification," *Communications in Information and Systems*, vol. 11, no. 3, pp. 197–224, 2011.

[59] J. Swevers, C. Ganseman, D. Tukel, J. de Schutter, and H. Van Brussel, "Optimal robot excitation and identification," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 730–740, 1997.

[60] M. Gautier and W. Khalil, "Exciting Trajectories for the Identification of Base Inertial Parameters of Robots," *The International Journal of Robotics Research*, vol. 11, pp. 362–375, Aug. 1992.

[61] A. F. Emery and A. V. Nenarokomov, "Optimal experiment design," *Measurement Science and Technology*, vol. 9, pp. 864–876, June 1998.

[62] R. Mehra, "Optimal input signals for parameter estimation in dynamic systems–Survey and new results," *IEEE Transactions on Automatic Control*, vol. 19, pp. 753–768, Dec. 1974.

[63] T. L. Vincent, C. Novara, K. Hsu, and K. Poolla, "Input design for structured nonlinear system identification," *Automatica*, vol. 46, pp. 990–998, June 2010.

[64] B. Wahlberg, H. Hjalmarsson, and M. Annergren, "On optimal input design in system identification for control," in *49th IEEE Conference on Decision and Control*, pp. 5548–5553, Dec. 2010.

[65] W. Rackl, R. Lampariello, and G. Hirzinger, "Robot excitation trajectories for dynamic parameter estimation using optimized B-splines," in *2012 IEEE International Conference on Robotics and Automation*, pp. 2042–2047, May 2012.

[66] W. Wu, S. Zhu, X. Wang, and H. Liu, "Closed-loop Dynamic Parameter Identification of Robot Manipulators Using Modified Fourier Series," *International Journal of Advanced Robotic Systems*, May 2012.

[67] K.-J. Park, "Fourier-based optimal excitation trajectories for the dynamic identification of robots," *Robotica*, vol. 24, p. 625, Mar. 2006.

[68] K. Hsu, K. Poolla, and T. L. Vincent, "Identification of Structured Nonlinear Systems," *IEEE Transactions on Automatic Control*, vol. 53, pp. 2497–2513, Dec. 2008.

[69] S. K. Pradhan and B. Subudhi, "Real-Time Adaptive Control of a Flexible Manipulator Using Reinforcement Learning," *IEEE Transactions on Automation Science and Engineering*, vol. 9, pp. 237–249, Apr. 2012.

[70] H. Hjalmarsson and J. Martensson, "Optimal Input Design for Identification of Non-linear Systems: Learning From the Linear Case," in *2007 American Control Conference*, pp. 1572–1576, July 2007.

[71] H. Jansson and H. Hjalmarsson, "Input design via LMIs admitting frequency-wise model specifications in confidence regions," *IEEE Transactions on Automatic Control*, vol. 50, pp. 1534–1549, Oct. 2005.

[72] M. Casini, A. Garulli, and A. Vicino, "Time complexity and input design in worst-case identification using binary sensors," in *2007 IEEE Conference on Decision and Control*, pp. 5528–5533, 2007.

[73] H. Kim, B. Park, J. Lee, Y. J. Park, and W. K. Chung, "A parameter estimation method for the bilateral teleoperation framework for an O¡inf¿2¡/inf¿ lance manipulator," in *2012 IEEE International Conference on Automation Science and Engineering*, pp. 564–568, Aug. 2012.

[74] M. Gautier and W. Khalil, "Direct calculation of minimum set of inertial parameters of serial robots," *IEEE Transactions on Robotics and Automation*, vol. 6, pp. 368–373, June 1990.

[75] K. Radkhah, D. Kulic, and E. Croft, "Dynamic parameter identification for the CRS A460 robot," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3842–3847, Oct. 2007.

[76] A. Vivas, P. Poignet, F. Marquet, F. Pierrot, and M. Gautier, "Experimental dynamic identification of a fully parallel robot," in *2003 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3278–3283, 2003.

[77] P. Kopacek, P. Lutz, A. Pashkevich, J. Wu, J. Wang, L. Wang, and T. Li, "Dynamic formulation of redundant and nonredundant parallel manipulators for dynamic parameter identification," *Mechatronics*, vol. 19, no. 4, pp. 586–590, 2009.

[78] N. Farhat, V. Mata, A. Page, and F. Valero, "Identification of dynamic parameters of a 3-DOF RPS parallel manipulator," *Mechanism and Machine Theory*, vol. 43, no. 1, pp. 1–17, 2008.

[79] E. Lehmann, G. Casella, and E. L Lehmann, "Theory of Point Estimation," *Design*, vol. 31, p. 589, 1998.

[80] T. V. Small, "Optimal trajectory-shaping with sensitivity and covariance techniques," Master's thesis, Massachusetts Institute of Technology, 2010.

[81] B. R. Saunders, "Optimal trajectory design under uncertainty," Master's thesis, Massachusetts Institute of Technology, 2012.

[82] S. Zimmer, C. Ocampo, and R. Bishop, "Reducing Orbit Covariance for Continuous Thrust Spacecraft Transfers," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, pp. 771–791, Apr. 2010.

[83] A. Ansari and T. Murphey, "Minimal parametric sensitivity trajectories for nonlinear systems," in *2013 American Control Conference*, pp. 5011–5016, June 2013.

[84] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, 2009.

[85] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1–4, May 2011.

[86] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration*, vol. 31 of *Springer Series in Computational Mathematics*. Berlin/Heidelberg: Springer-Verlag, 2nd ed., 2006.

[87] E. Johnson and T. Murphey, "Scalable Variational Integrators for Constrained Mechanical Systems in Generalized Coordinates," *IEEE Transactions on Robotics*, vol. 25, pp. 1249–1261, Dec. 2009.

[88] E. F. Camacho and C. B. Alba, *Model predictive control*. London: Springer-Verlag, 2007.

[89] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*. London: Springer-Verlag, 2011.

[90] T. M. Caldwell, D. Coleman, and N. Correll, "Optimal parameter identification for discrete mechanical systems with application to flexible object manipulation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 898–905, Sept. 2014.

[91] R. Mao, Y. Yang, C. Fermuller, Y. Aloimonos, and J. S. Baras, "Learning hand movements from markerless demonstrations for humanoid tasks," in *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 938–943, Nov. 2014.

[92] C. Bowen and R. Alterovitz, "Closed-loop global motion planning for reactive execution of learned tasks," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1754–1760, Sept. 2014.

[93] Rethink Robotics Baxter Research Robot. [Online]. Available: http://www.rethinkrobotics.com/baxter-research-robot/.

[94] R. Sim and N. Roy, "Global a-optimal robot exploration in slam," in *2005 IEEE International Conference on Robotics and Automation*, pp. 661–666, April 2005.

[95] A. Makarenko, S. Williams, F. Bourgault, and H. Durrant-Whyte, "An experiment in integrated exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 534–539, 2002.

[96] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, 2000.

[97] Y. Silverman, L. Miller, M. MacIver, and T. Murphey, "Optimal planning for information acquisition," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5974–5980, Nov 2013.

[98] L. Miller and T. Murphey, "Trajectory optimization for continuous ergodic exploration," in *2013 American Control Conference*, pp. 4196–4201, June 2013.

[99] T. Caldwell and T. Murphey, "Projection-based optimal mode scheduling," in *2013 IEEE 52nd Annual Conference on Decision and Control*, pp. 5307–5314, Dec 2013.

[100] I. W. Charlton and G. R. Johnson, "Application of spherical and cylindrical wrapping algorithms in a musculuskeletal model of the upper limb," *Journal of Biomechanics*, vol. 34, pp. 1209–1216, 2001.

[101] R. B. Nelson, "Simplified calculation of eigenvector derivatives," *AIAA Journal*, vol. 14, pp. 1201–1205, Sept. 1976.

[102] M. I. Friswell, "The Derivatives of Repeated Eigenvalues and Their Associated Eigenvectors," *Journal of Vibration and Acoustics*, vol. 118, p. 390, July 1996.

APPENDIX A

Continuous Cylindrical Wrapping Model

An important part of modeling human biomechanics, particularly the upper limb is the creation of muscle wrapping surfaces. These surfaces simulate the natural wrapping that occurs around the bones and joints in the upper limb. For use with the variational integrator system derived in Chapter 3, the wrapping surface will be assumed to be completely passive and frictionless, performing no work on the muscle or skeletal structure. In order to satisfy this requirement, a continuous model is needed, including derivatives of the wrapped muscle length w.r.t. changes in the generalized joint coordinate. This section presents a derivation of the continuous cylindrical model which is modified from [**100**].

To begin, two tendon attachment points are defined, p_1 and p_2. For this implementation, the wrapping surface must completely lie between these two points as the frames move. The cylinder and point of rotation are both defined in the same reference frame as p_1, so there is no relative movement between p_1 and the wrapping surface. The transformation, T_{12} defines the transformation from p_1 to p_2. For many systems, this will include a translation, followed by a generalized rotation, followed by another translation. The point, p_x defines the point from which the cylinder and axis of rotation are defined. Both

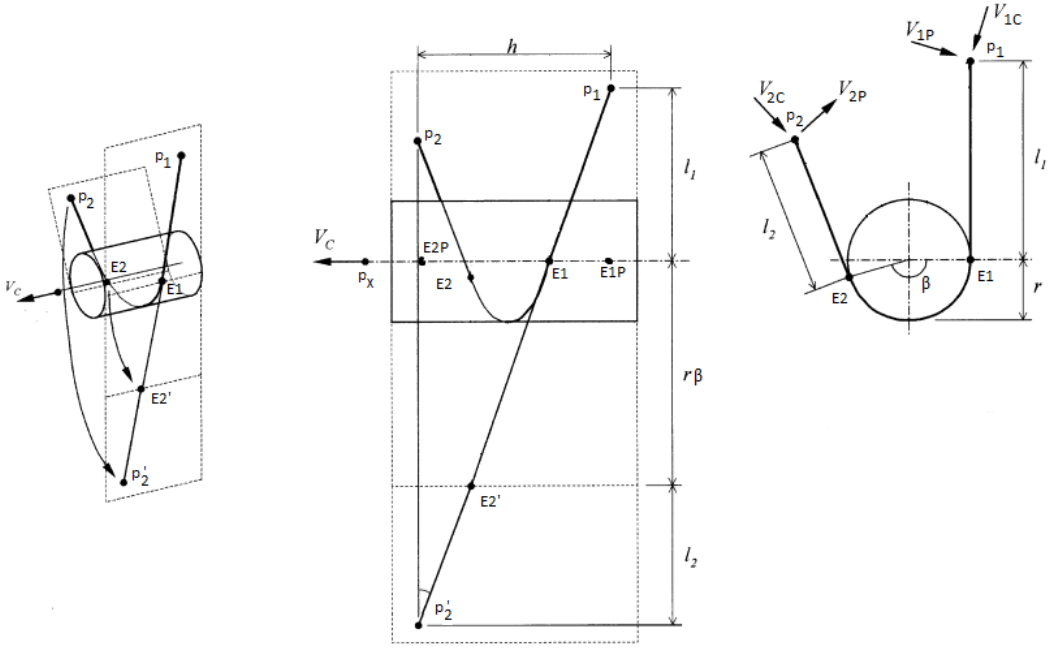


Figure A.1. Coordinates used for unwrapped cylinder model. Graphic modified from [**100**].

the cylinder and axis of rotation can be defined by unit vectors, where V_c denotes the cylinder axis and V_r denotes the rotation axis.

Therefore, it is assumed that p_2 is only a function of one generalized coordinate, which will be the angle of rotation, ϕ around V_r. Additionally, several other unit vectors and quantities are defined to assist in the calculation of the wrapping geometry. Ultimately, the two points that are tangent to the wrapping surface, E_1 and E_2, and the arclength around the cylinder must be computed.

A.1. Calculating the Wrapping Condition

The system is constrained to only wrap around one direction of the cylinder, thus ensuring that it will be wrapped correctly even if the line between p_1 and p_2 does not pass through

the cylinder. It is assumed that the cylinder axis of rotation, V_c is defined such that the tendon from p_1 to p_2 wraps in the positive direction around V_c.

The condition for wrapping is given by the following equation

$$(p_x - p_1) \cdot \frac{(V_c \times (p_2 - p_1))}{|V_c \times (p_2 - p_1)|}, \tag{A.1}$$

where $|\cdot|$ denotes the Euclidean norm. If (A.1) is less than the radius, r, including negative, the tendon wraps. If the value is positively greater than r, the tendon does not wrap. This assumes that the joint cannot rotate more than one revolution, otherwise a discontinuity will occur. However, in musculoskeletal modeling, this should not present a problem.

A.2. Calculating the Wrapped Tendon Length

If the wrapping condition is not satisfied, the length of the tendon is simply

$$|p_2 - p_1| .$$

If the wrapping condition is satisfied, a few intermediate quantities are calculated to make the problem manageable. The first quantity is the shortest distance from each tendon point to the cylinder axis. Note that p_x can lie anywhere on the defined cylinder axis, though will probably be the origin of the definition. For either p_1 or p_2, the equation is given by

$$g_{1,2} = |(p_{1,2} - p_x) \times (p_{1,2} - (p_x + V_c))| .$$

The distance from p_1 and p_2 to the face of the cylinder is also needed, which is just the third side of a right triangle given by

$$\ell_{1,2} = \sqrt{g_{1,2}^2 - r^2}.$$

Next the unit vectors that are parallel and perpendicular to this line are computed from each tendon point to the cylinder axis. The parallel components are denoted V_{1c} and V_{2c} and the perpendicular components are V_{1p} and V_{2p}.

The equations are as follows,

$$V_{1p,2p} = \frac{V_c \times (p_{1,2} - p_x)}{|V_c \times (p_{1,2} - p_x)|}$$
$$V_{1c,2c} = \frac{V_c \times V_{1p,2p}}{|V_c \times V_{1p,2p}|}.$$

Now, points p_1 and p_2 are projected onto the cylinder at the shortest distance. These points will be denoted as E_{1p} and E_{2p}. The equation is

$$E_{1p} = p_1 + (g_1 - \frac{r^2}{g_1})V_{1c} + r\sqrt{1 - \frac{r^2}{g_1^2}}V_{1p}$$
$$E_{2p} = p_2 + (g_2 - \frac{r^2}{g_2})V_{2c} - r\sqrt{1 - \frac{r^2}{g_2^2}}V_{2p}.$$

The angle between E_{1p} and E_{2p} must also be computed to find the arc length of the path around the cylinder. This is the angle of the points projected on a cross section of

the cylinder. The equation for this angle is given by

$$m = (p_x - E_{1p}) \cdot \frac{(E_{1p} - E_{2p}) \times V_c}{|(E_{1p} - E_{2p}) \times V_c|}$$
$$\beta = \pi - 2 \arcsin \left[\frac{m}{r} \right].$$

Note that in this chapter, β represents the wrapping angle as opposed to an estimator from the prior uses in the thesis. Now that the angle has been computed, the last quantity needed is the parallel distance between the tendon points. This is given by

$$h = (p_2 - p_1) \cdot V_c.$$

Using the unwrapped cylinder model, and similar triangles, the total length of the wrapped tendon ℓ^{WT} is given by

$$\ell^{WT} = \sqrt{h^2 + (\ell_1 + \ell_2 + r\beta)^2}.$$

If the locations of the actual tangent points on the cylinder are of interest, the points are given by:

$$E_1 = E_{1p} + h_1 V_c$$
$$E_2 = E_{2p} - h_2 V_c,$$

where

$$h_{1,2} = \frac{h\ell_{1,2}}{r\beta + \ell_1 + \ell_2}.$$

A.2.1. Derivatives of the Muscle Length

In order to use the virtual work principle to solve for the muscle moment on the joint, the derivative of the length w.r.t. the joint ϕ is required. Using the derived equations for the length of the tendon wrapped around the cylinder, the derivative can be readily calculated. Assume that p_1 is fixed with respect to the cylinder and rotation axis, therefore, any derivative of a quantity only involving p_1 is equal to zero. Also assume that p_2 is related to p_1 by a set of fixed translations and one rotation around an arbitrary axis defined at p_x. Given these assumptions, and that $p_2 = T_{12} \cdot p_1$, the derivative of p_2 with respect to ϕ is given by

$$\frac{dp_2}{d\phi} = \frac{dT_{12}}{d\phi} \cdot p_1.$$

The remaining equations will be in terms of the derivative of p_2 with respect to ϕ which will be denoted by Dp_2 for simplicity. If the condition from (A.1) is not satisfied, the tendon is not wrapping and the derivative of the length is given by

$$\frac{dL_t}{d\phi} = \frac{(p_2 - p_1) \cdot Dp_2}{|(p_2 - p_1)|}.$$

If the tendon is wrapping, each of the intermediate quantities are differentiated to compute the derivative of the length. Again the simpler D notation will be used. The equations are derived as follows

$$Dg_2 = -(Dp_2 \times V_c) \cdot \frac{(p_2 - p_x) \times (p_2 - (p_x + V_c))}{|(p_2 - p_x) \times (p_2 - (p_x + V_c))|}$$

$$DV_{2p} = \frac{(V_c \times Dp_2)}{|V_c \times (p_2 - p_x)|} - \frac{(V_c \times (p_2 - p_x)) \cdot (V_c \times Dp_2)}{((V_c \times (p_2 - p_x)) \cdot (V_c \times (p_2 - p_x))^{3/2}} (V_c \times (p_2 - p_x))$$

$$DV_{2c} = \frac{V_c \times DV_{2p}}{|V_c \times DV_{2p}|} - \frac{(V_c \times V_{2p}) \cdot (V_c \times DV_{2p})}{((V_c \times V_{2p}) \cdot (V_c \times V_{2p}))^{3/2}}(V_c \times V_{2p})$$

$$DE_{2p} = Dp_2 + Dg_2(\frac{r^2}{g_2^2} + 1)V_{2c} + (g_2 - \frac{r^2}{g_2})DV_{2c} - \frac{r^3 Dg_2}{g_2^2\sqrt{1 - \frac{r^2}{g_2^2}}}V_{2p} - r\sqrt{1 - \frac{r^2}{g_2^2}}DV_{2p}$$

$$\begin{aligned} Dm = &- \frac{(p_x - E_{1p}) \cdot (DE_{2p} \times V_c)}{|(E_{1p} - E_{2p}) \times V_c|} + \\ &\frac{((E_{1p} - E_{2p}) \times V_c) \cdot (DE_{2p} \times V_c)}{(((E_{1p} - E_{2p}) \times V_c) \cdot ((E_{1p} - E_{2p}) \times V_c))^{3/2}}(p_x - E_{1p}) \cdot ((E_{1p} - E_{2p}) \times V_c). \end{aligned}$$

Using these quantities, the derivative of the wrapping point angle w.r.t. the joint rotation ϕ is given by

$$D\beta = \frac{-2Dm}{r\sqrt{1 - \left(\frac{m}{r}\right)^2}}.$$

Finally, using two intermediate quantities defined by the following

$$Dh = Dp_2 \cdot V_c$$

$$D\ell_2 = \frac{g_2 Dg_2}{\sqrt{g_2^2 - r^2}}.$$

The derivative of the length of the tendon with respect to ϕ can be computed with

$$DL_t = \frac{hDh + (\ell_1 + \ell_2 + r\beta)(D\ell_2 + rD\beta)}{\sqrt{h^2 + (\ell_1 + \ell_2 + r\beta)^2}}.$$

Although there is a discrete check and switch between equations as the tendon goes between contact and no contact with the cylinder, the first derivative is continuous, therefore satisfying a satisfying a no-work condition on the cylinder. This should be the case since the cylinder is assumed to be passive and frictionless.

APPENDIX B

Computing the Descent Directions, ζ

This chapter presents the detailed derivations of the descent directions used in Chapters 4 and 5. Section B.1 presents the continuous version of the derivation while Section B.2 covers the derivation using discrete mechanics. Both formulations are used in the respective optimization algorithms to iteratively compute local directions of descent to optimize the trajectories.

B.1. Continuous-time Descent Direction

To find a descent direction for the continuous optimal control algorithm, (5.3) or (4.6) must be solved. As shown in the cost functions, the descent direction depends on the linearization $DJ(P(\xi_k(t)))$, and the local quadratic model, $\langle \zeta_k(t), \zeta_k(t) \rangle$. Using a quadratic model and expanding the linearizations of the cost function, (5.3) is rewritten as

$$\arg\min_{\zeta_k(t)} = \int_{t_0}^{t_f} a(t)^T \bar{z}(t) + b(t)^T v(t) + \frac{1}{2}\bar{z}(t)^T Q_n \bar{z}(t) + \frac{1}{2} v(t)^T R_n v(t)\ dt, \tag{B.1}$$

such that

$$\dot{\bar{z}} = A\bar{z} + Bv,$$

where $a(t)$ and $b(t)$ are the linearizations of the cost function with respect to $\bar{x}$ and u, and Q_n and R_n are weighting matrices for the local quadratic model approximation. Design of these weighting matrices can lead to faster convergence of the optimal control algorithm depending on the specific problem. The derivations of the cost function linearizations and the dynamics linearizations are now presented.

B.1.1. Cost Function Linearization - Information Maximization

The linearization of the cost function $DJ(P(\xi_k(t)))$ is found by taking the directional derivative of (4.3) with respect to the extended states $\bar{x}(t)$ and the control vector $u(t)$.

The derivative of (4.3) with respect to $\bar{x}$ yields

$$\frac{\partial J}{\partial \bar{x}} = -\frac{Q_p}{\lambda_{min}^2}\frac{\partial \lambda_{min}}{\partial \bar{x}} + \int_{t_0}^{t_f} \left[(x(t) - x_d(t))^T \cdot Q_\tau\right] dt. \tag{B.2}$$

Since the cost function involves eigenvalues of $\tilde{I}(\theta)$, this linearization requires the calculation of the derivatives of eigenvalues. A process for this calculation was formalized by Nelson [**101**]. Given an eigensystem of the form

$$AX = X\Lambda,$$

where Λ is a diagonal matrix of distinct eigenvalues $(\lambda_1, \lambda_2, ...\lambda_n)$, and X is the associated matrix of eigenvectors, the derivative of an eigenvalue λ_m is given by

$$D_x\lambda_m = \omega_m^T \cdot D_x A \cdot \nu_m,$$

where ω_m is the left eigenvector and ν_m is the right eigenvector associated with λ_m.

If the eigenvalues are not distinct, i.e., multiplicity greater than 1, Nelson's method does not hold since the choice of eigenvectors is not unique. However, in the case of repeated eigenvalues and repeated eigenvalue derivatives, a set of eigenvectors can be determined up to a scalar multiplier as detailed in [**102**]. Once the eigenvectors are calculated, each eigenvalue and eigenvector pair can be used to compute the direction of steepest descent for the objective.

Using these methods to compute the eigenvalue derivative, $\frac{\partial \lambda_{min}}{\partial \bar{x}}$ from (B.2) can be calculated. Taking the derivative of the eigenvalue of $\tilde{I}(\theta)$ from (4.2) with respect to the extended state yields

$$\frac{\partial \lambda_s}{\partial \bar{x}} = \omega_s^T \frac{\partial}{\partial \bar{x}} \left(\int_{t_0}^{t_f} \Gamma_\theta(t)^T \cdot \Sigma^{-1} \cdot \Gamma_\theta(t) \, dt \right) \nu_s,$$

where s denotes the index of the minimum eigenvalue and eigenvector. Since the partial derivative and eigenvectors are evaluated only at the final time, the equation can be rewritten to a running cost formulation given by

$$\frac{\partial \lambda_s}{\partial \bar{x}} = \int_{t_0}^{t_f} \omega_s^T \frac{\partial}{\partial \bar{x}} \left(\Gamma_\theta(t)^T \cdot \Sigma^{-1} \cdot \Gamma_\theta(t) \right) \nu_s \, dt. \tag{B.3}$$

Finally, differentiating the inner product of the gradients yields

$$\frac{\partial}{\partial \bar{x}} \left(\Gamma_\theta(t)^T \cdot \Sigma^{-1} \cdot \Gamma_\theta(t) \right) = \begin{bmatrix} 2\,\Gamma_\theta(t)^T \cdot \Sigma^{-1} \cdot (D_x^2 g(\cdot) \cdot \psi(\cdot) + D_x D_\theta g(\cdot)) \\ 2\,\Gamma_\theta(t)^T \cdot \Sigma^{-1} \cdot D_x g(\cdot) \cdot E \end{bmatrix}, \tag{B.4}$$

where E is a tensor of the form

$$E_{ijkl} = \delta_{ik} \delta_{jl}$$

with δ as the Kronecker delta function.

Combining equations (B.2), (B.3), and (B.4), $a(t)$ in (B.1) is given by,

$$a(t) = \begin{bmatrix} (x(t) - x_d(t))^T Q_\tau \\ \{0\}^{1\times n\times p} \end{bmatrix} - \frac{Q_p}{\lambda_{min}^2}\omega_s^T \begin{bmatrix} 2\,\Gamma_\theta(t)^T\Sigma^{-1}\left(D_x^2 g(\cdot)\psi(t) + D_x D_\theta g(\cdot)\right) \\ 2\,\Gamma_\theta(t)^T\Sigma^{-1} D_x g(\cdot) E \end{bmatrix} \nu_s.$$

The linearization $b(t)$ from (B.1), defining the derivative of the cost function with respect to the controls $u(t)$, is given by

$$b(t) = u(t)^T \cdot R_\tau,$$

where R_τ is the weighting matrix from (4.3).

B.1.1.1. Cost Function Linearization - Conditioning Optimization. The first term of (B.1) is computed as the linearization of the cost function with respect to the extended states, $\bar{x}(t)$ and the controls, $u(t)$. To begin, we apply the quotient rule to the cost function given in (4.6). Taking the first derivative results in the following equation for $a(t)$,

$$a(t) = \frac{\partial J}{\partial \bar{x}} = Q_p\left(\frac{1}{\lambda_{min}}\frac{\partial \lambda_{max}}{\partial \overline{x}} - \frac{\lambda_{max}}{\lambda_{min}^2}\frac{\partial \lambda_{min}}{\partial \overline{x}}\right) + \int_{t_0}^{t_f}\left[(x(t) - x_d(t))^T \cdot Q_\tau\right] dt. \tag{B.5}$$

where λ are the eigenvalues of the approximation of the Hessian, $\frac{d^2}{d\theta^2}\beta(\theta)$.

It is clear from this result that the derivative of the eigenvalues, $\frac{\partial \lambda_{max}}{\partial \overline{x}}$ is needed. Fortunately, eigenvalue perturbation theory permits such differentiation, and the resulting form is relatively compact. From [**101**], the derivative of one eigenvalue of some matrix

A is given by

$$D_i\lambda_k = y_k^T \cdot D_i A \cdot x_k, \tag{B.6}$$

where λ_k is the k^{th} eigenvalue of A, x_k is the associated left eigenvector, y_k is the associated right eigenvector of A, and D_iA and $D_i\lambda_k$ are the partial derivatives of A and λ_k with respect to some argument i.

Given (B.6), the derivative of λ_i, for $i \in \{min, max\}$, with respect to $\bar{\mathbf{x}}$ is written as

$$\frac{\partial \lambda_i}{\partial \overline{x}} = \int_{t_0}^{t_f} w_s^T \frac{\partial}{\partial \overline{x}} \left[\Gamma_\theta(t)^T \cdot Q_p \cdot \Gamma_\theta(t)^T\right] w_s \, dt, \tag{B.7}$$

where w_s is the eigenvector associated with λ_i.

Lastly, differentiating the terms of the Hessian yields,

$$\frac{\partial}{\partial \overline{x}} \left(\Gamma_\theta(t)^T \cdot Q_p \cdot \Gamma_\theta(t)\right) = \begin{bmatrix} 2\,\Gamma_\theta(t)^T \cdot Q_p \cdot (D_x^2 g(\cdot) \cdot \psi(\cdot) + D_x D_\theta g(\cdot)) \\ 2\,\Gamma_\theta(t)^T \cdot Q_p \cdot D_x g(\cdot) \cdot E \end{bmatrix}, \tag{B.8}$$

where E is a tensor of the form

$$E_{i,j,k,l} = \delta_{i,k}\delta_{j,l}$$

with $\delta_{\cdot,\cdot}$ as the Kronecker delta function.

Combining equations (B.5), (B.7), (B.8), results in the complete linearization $\mathbf{a}(t)$ in (B.1). The linearization with respect to the controls $\mathbf{u}(t)$ is simply given by

$$b(t) = u(t)^T \cdot R_\tau.$$

B.1.2. Dynamics Linearization

For both the information and conditioning optimizations, the two other quantities needed to compute the descent direction are $A(t)$ and $B(t)$ – the linearizations of the dynamics. These linearizations take the same form for both methods. The descent direction ζ_k will satisfy the linear constraint ODE given by

$$\dot{\bar{z}}_k(t) = A(t)\bar{z}_k(t) + B(t)v_k(t),$$

where $A(t)$ is the linearization of the nonlinear dynamics given by (2.2) and (2.7) with respect to $\bar{x}(t)$, and $B(t)$ is the linearization with respect to $u(t)$. The linearization $A(t)$ of the dynamics with respect to the extended state $\bar{x}(t)$ is given by

$$A(t) = \begin{bmatrix} \frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial \psi} \\ \frac{\partial \dot{\psi}}{\partial x} & \frac{\partial \dot{\psi}}{\partial \psi} \end{bmatrix} = \begin{bmatrix} D_x f(\cdot) & \{0\}^{n\times n\times p} \\ D_x^2 f(\cdot)\cdot\psi(t) + D_x D_\theta f(\cdot) & D_x f(\cdot)\cdot E \end{bmatrix}.$$

In addition to the state linearizations, the control linearizations are required. This linearization matrix $B(t)$ is given by

$$B(t) = \begin{bmatrix} \frac{\partial \dot{x}}{\partial u} \\ \frac{\partial \dot{\psi}}{\partial u} \end{bmatrix} = \begin{bmatrix} D_u f(\cdot) \\ D_u D_x f(\cdot)\cdot\psi(t) + D_u D_\theta f(\cdot) \end{bmatrix}.$$

Given the linearizations $a(t), b(t), A(t)$, and $B(t)$, (B.1) can be used to compute the descent direction $\zeta_k(t)$. At each iteration of the optimization algorithm, the perturbed trajectory $\eta_k(t) + \gamma_k\zeta_k$ must be projected to satisfy the dynamics. As shown in Algorithm 2.2, the process is repeated until a termination criteria is satisfied.

B.2. Descent Direction for Discrete Mechanical Systems

To find a descent direction for the discrete optimal control algorithm, (5.3) must be solved. As shown in (5.3), the descent direction depends on the linearization of the cost function, $DJ(P(\xi_k))$, and the local quadratic model, $\langle \zeta_k, \zeta_k \rangle$. Using a quadratic model and expanding the linearizations of the cost function, (5.3) is rewritten as

$$\arg\min_{\zeta_k} = \sum_{k=k_0}^{k_f} 2a_k^T \bar{z}_k + 2b_k^T \bar{v}_k + \bar{z}_k^T Q_n \bar{z}_k + \bar{v}_k^T R_n \bar{v}_k \; dt,$$

such that

$$\bar{z}_{k+1} = \bar{A}\bar{z}_k + B\bar{v}_k,$$

where a_k and b_k are the linearizations of the cost function with respect to $\bar{x}$ and $\bar{u}$, and Q_n and R_n are weighting matrices for the local quadratic model approximation. Design of these weighting matrices can lead to faster convergence of the optimal control algorithm depending on the specific problem. Since the derivations of the cost function linearizations follows the same steps as the continuous-time formulation, we refer reader to [**13**] for the derivations. We present the equations here in discrete-time for reference.

$$a_k = \begin{bmatrix} (x_k - \hat{x}_k)^T Q_\tau \\ \{0\}^{1\times n\times p} \end{bmatrix} - \frac{Q_p}{\lambda_{min}^2} \omega_s^T \begin{bmatrix} 2\,\Gamma_k^T \Sigma_k^{-1} \left(\frac{\partial^2 g}{\partial x_k^2} \psi_k + \frac{\partial^2 g}{\partial \theta \partial x_k} \right) \\ 2\,\Gamma_k^T \Sigma_k^{-1} \frac{\partial g}{\partial x_k} E \end{bmatrix} \nu_s$$

$$b_k = (\bar{u}_k - \hat{\bar{u}}_k)^T \cdot R_\tau$$

$$\bar{A}_k = \begin{bmatrix} \frac{\partial x_{k+1}}{\partial x_k} & \frac{\partial x_{k+1}}{\partial \psi_k} \\ \frac{\partial \psi_{k+1}}{\partial x_k} & \frac{\partial \psi_{k+1}}{\partial \psi_k} \end{bmatrix} = \begin{bmatrix} A_k & \{0\}^{n\times n\times p} \\ \frac{\partial^2 x_{k+1}}{\partial x_k^2}\cdot\psi_k + \frac{\partial^2 x_{k+1}}{\partial\theta\partial x_k} & A_k\cdot E \end{bmatrix} \tag{B.9}$$

$$\bar{B}_k = \begin{bmatrix} \frac{\partial x_{k+1}}{\partial \bar{u}_k} \\ \frac{\partial \psi_{k+1}}{\partial \bar{u}_k} \end{bmatrix} = \begin{bmatrix} B_k \\ \frac{\partial^2 x_{k+1}}{\partial x_k \partial \bar{u}_k}\cdot\psi_k + \frac{\partial^2 x_{k+1}}{\partial\theta\partial\bar{u}_k} \end{bmatrix} \tag{B.10}$$

where ω_s and ν_s are the left and right eigenvectors of $I(\theta)$ respectively. Definitions of $\frac{\partial^2 x_{k+1}}{\partial\theta\partial x_k}$ and $\frac{\partial^2 x_{k+1}}{\partial\theta\partial\bar{u}_k}$ are presented in the following section. E is a tensor of the form $E_{ijkl} = \delta_{ik}\delta_{jl}$ where δ as the Kronecker delta function.

After computing the linearization of the cost function and dynamics at each time step along the trajectory, (5.3) can be solved using the discrete-time algebraic Riccati equation. The formulation of this LQ problem is detailed in the appendix of [**33**].

B.2.1. Descent direction derivatives

In order to compute the linearization of the extended state dynamics, $\frac{\partial^2 x_{k+1}}{\partial\theta\partial x_k}$ and $\frac{\partial^2 x_{k+1}}{\partial\theta\partial\bar{u}_k}$ are required to compute (B.9) and (B.10).

$\frac{\partial^2 x_{k+1}}{\partial\theta\partial x_k}$ is constructed from the following components:

$$\frac{\partial^2 q_{k+1}}{\partial\theta\partial q_k} = -M^{-1}[(D_2D_5D_1L_{k+1} + D_2D_6F^-_{k+1})\frac{\partial q_{k+1}}{q_k} + D_1D_5D_1L_{k+1} + D_1D_6F^-_{k+1}]$$

$$\frac{\partial^2 q_{k+1}}{\partial\theta\partial p_k} = -M^{-1}[(D_2D_5D_1L_{k+1} + D_2D_6F^-_{k+1})\frac{\partial q_{k+1}}{p_k}]$$

$$\frac{\partial^2 p_{k+1}}{\partial\theta\partial q_k} =(D_2D_2D_2L_{k+1}\frac{\partial q_{k+1}}{\partial q_k} + D_1D_2D_2L_{k+1})\frac{\partial q_{k+1}}{\partial\theta} + D_2D_2L_{k+1}\frac{\partial^2 q_{k+1}}{\partial\theta\partial q_k}$$
$$+ D_2D_5D_2L_{k+1}\frac{\partial q_{k+1}}{\partial q_k} + D_1D_5D_2L_{k+1}$$
$$\frac{\partial^2 p_{k+1}}{\partial\theta\partial p_k} =(D_2D_2D_2L_{k+1}\frac{\partial q_{k+1}}{\partial p_k})\frac{\partial q_{k+1}}{\partial\theta} + D_2D_2L_{k+1}\frac{\partial q_{k+1}^2}{\partial\theta\partial p_k} + D_2D_5D_2L_{k+1}\frac{\partial q_{k+1}}{\partial p_k},$$

and $\frac{\partial^2 x_{k+1}}{\partial\theta\partial\bar{u}_k}$ is constructed from the following components

$$\frac{\partial^2 q_{k+1}}{\partial\theta\partial u_k} = - M^{-1}[(D_2D_5D_1L_{k+1} + D_2D_6F_{k+1}^-)\frac{\partial q_{k+1}}{\partial u_k} + D_5D_6F_{k+1}^-]$$
$$\frac{\partial^2 q_{k+1}}{\partial\theta\partial\rho_{k+1}} = - M^{-1}[(D_2D_5D_1L_{k+1} + D_2D_6F_{k+1}^-)\frac{\partial q_{k+1}}{\partial\rho_{k+1}} + D_4D_5D_1L_{k+1} + D_4D_6F_{k+1}^-]$$
$$\frac{\partial^2 p_{k+1}}{\partial\theta\partial u_k} =(D_2D_2D_2L_{k_1}\frac{\partial q_{k+1}}{\partial u_k})\frac{\partial q_{k+1}}{\partial\theta} + D_2D_2L_{k+1}\frac{\partial q_{k+1}^2}{\partial\theta\partial u_k} + D_2D_5D_2L_{k+1}\frac{\partial q_{k+1}}{\partial u_k}$$
$$\frac{\partial^2 p_{k+1}}{\partial\theta\partial\rho_{k+1}} =(D_2D_2D_2L_{k_1}\frac{\partial q_{k+1}}{\partial\rho_{k+1}})\frac{\partial q_{k+1}}{\partial\theta} + D_2D_2L_{k+1}\frac{\partial q_{k+1}^2}{\partial\theta\partial\rho_{k+1}} + D_4D_5D_2L_{k+1}$$
$$+ D_2D_5D_2L_{k+1}\frac{\partial q_{k+1}}{\partial\rho_{k+1}}.$$

The equations representing the remaining components of A_k, B_k and M are derived and presented in full in [**31**].