NORTHWESTERN UNIVERSITY

Trajectory Optimization and Regulation for Constrained Discrete 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

Elliot Johnson

EVANSTON, ILLINOIS

August 2012

ABSTRACT

Trajectory Optimization and Regulation for Constrained Discrete Mechanical Systems

Elliot Johnson

Projection operator-based trajectory optimization uses unconstrained gradient-descent methods to find desirable trajectories of a mechanical system. The constraint on the admissible trajectories imposed by the system dynamics is handled by a projection operator that uses linear feedback to map arbitrary curves to nearby feasible trajectories. While other optimization algorithms do not guarantee admissible trajectories until an optimum is found, this approach results in valid trajectories at each step of the algorithm. This is a major benefit for applications by allowing a trade-off between computation time and the optimality of the result. Additionally, the computational complexity of an optimization is linear with respect to the time horizon.

Each step of the algorithm involves multiple dynamic simulations as well as calculating the first- (and potentially second-) order linearizations of the dynamics. Algorithms to calculate continuous dynamics of arbitrary system in linear time exist, but linearizations are typically calculated by working directly with symbolic equations of motion. This has been a limiting factor to optimizing high-dimensional systems. Numerical instabilities introduced by numeric integration of continuous dynamics compound the problem by introducing artificial instabilities and calculating linearizations that only approximate the true derivatives of the dynamics.

Variational integrators model the dynamics of a system directly in the discrete time domain, eliminating many of the problems caused by integration. Stability and other fundamental properties of the dynamics are preserved.

This thesis develops an algorithmic approach to implement variational integrators (and their exact first- and second-order linearizations) for arbitrary, constrained mechanical systems in generalized coordinates and adopts the projection operator-based optimization to discrete time. The algorithm is based on a recursive approach similar to those for continuous dynamics which allows it to scale to large dimensional systems. The recursive representation is also compatible with continuous dynamics and, while not as efficient as existing continuous methods, it provides the continuous first- and second-order linearizations as well. Several numerical examples are provided comparing the performance of the variational integrator to continuous dynamics, and a successful trajectory optimization is demonstrated for a string-actuated marionette in discrete time with 40 generalized coordinates and six holonomic constraints.

Table of Contents

List of Figures

CHAPTER 1

Introduction

This work was developed for the Automated Marionette Project. The goal of this project is to create a complete stack of technology that takes a high level description of a theatrical play to a physical performance by robotically-controlled marionettes. The ability to find dynamically feasible marionette trajectories (including the corresponding inputs) that approximate a desired motion is a key component of this project. Marionettes have highly non-linear dynamics and are limited to indirect actuation by strings rather than joint torques. Desired motions may come from arbitrary sources such as motion capture data from human actors or handmade performances by computer animators. They are not required to be dynamically feasible trajectories for the marionette.

Trajectory optimization is a constrained optimization problem that seeks curves that minimize a cost function while satisfying the system dynamics. Algorithms can roughly be divided into two categories. The first group explicitly uses constrained methods, similar to optimization in a finite vector space subject to constraints using Lagrange multipliers. These approaches simultaneously search for curves that minimize the cost and satisfy the dynamic constraint. Dynamically admissible trajectories are not produced in the intermediate stages of the optimization.

Approaches such as the Discrete Mechanics Optimal Control (DMOC) framework [**32**][**26**] discretize the time horizon to treat the problem directly as a finite dimensional constrained optimization that can be solved with conventional non-linear programming methods. These types of approaches lead to extremely high dimensional optimizations.

The second group uses the system dynamics to generate valid trajectories as the optimization progresses, allowing the optimization to be treated as an unconstrained problem. These shooting methods are typically limited to stable systems and short time horizons, as even minor changes to the inputs can lead to a drastically different trajectory.

Projection operator-based trajectory optimization[**16**][**15**] modifies this approach by incorporating a feedback law. A gradient descent method iteratively improves the trajectory by calculating a descent direction and performing a line search to improve the cost along this direction. This is possible because the feedback law introduces stability in the sense that small changes to the system inputs correspond to small changes in the resulting trajectory.

Projection operator-based optimization is well suited for the marionette problem, because it is guided by the system dynamics, and the inherent stability allows for optimizations over large time horizons. However, its applications have been limited to small systems because first- and second-order linearizations of the dynamics are required at each iteration to find the descent direction and, typically, the controller. While there are many fast algorithms to calculate a systems dynamics, linearizations are typically found directly from symbolic equations of the systems dynamics which grow significantly in complexity as the dimensionality of the system increases. Additionally, numeric integration introduces error to both the simulated trajectories, and the linearized trajectories used to find the descent direction. Consequently, the resulting linearization may not reflect the true linearization of either the system or the simulated trajectory.

The strings that control the marionette create another problem in addition to the large dimensionality of the system. Marionette strings are modeled as holonomic constraints. In continuous dynamics, holonomic constraints are implemented as equivalent constraints on the acceleration of the generalized coordinates. The acceleration constraints are enforced exactly at each time step, but error from the numeric integration causes the holonomic constraints to become violated over time.

Variational integrators provide a way to circumvent these problems. They are an alternative to simulating mechanical systems by numerically integrating the continuous ordinary differential equations of motion. Variational integrators often yield superior results, but were once thought to be impossible to scale to large systems without specialized coordinate choices.

The methods developed in this thesis enables scalable variational integrators for arbitrary mechanical systems in generalized coordinates. The approach relies on a tree-based representation of the system that efficiently calculates positions, velocities, and their derivatives (numerically and without approximation). The resulting integrators have excellent long-time behavior, elegant treatment of holonomic constraints, and scalable performance.

Variational integrators are a class of integrators [**14**] that preserve (or nearly preserve, depending on the particular integrator) fundamental properties like conservation of energy and momentum [**29**]. They have traditionally been used to study conservative systems (e.g. celestial mechanics), but modern variational integrators also include forcing and dissipation. In fact, the results with external forcing often have better accuracy than classical methods. Variational integrators also treat holonomic constraints and non-smooth phenomenon like collisions well.

While energy conservation does not guarantee an accurate solution, it is desirable that a simulation respects known properties of a system like energy dissipation and/or conservation of momentum. When these properties are not satisfied, the resulting trajectory is certainly incorrect.

Variational integrators handle holonomic constraints (expressed as $h(q) = 0$ for valid configurations q) better than continuous methods (Force-balance or Euler-Lagrange). Holonomic constraints restrict the relative positions and orientations between bodies. Non-holonomic constraints, on the other hand, restrict relative velocities between bodies rather than relative positions.

In continuous dynamics, holonomic constraints are not implemented directly; they are replaced with equivalent acceleration constraints. Errors introduced during numeric integration cause the system to violate the original constraint, resulting in unrealistic behavior like objects 'sinking' into hard surfaces [**37**]. Restoring forces–resembling damped springs–are added to correct these errors. These add arbitrary parameters (e.g. the Error Reduction Parameter in the Open Dynamics Engine) that must be tuned for each simulation (often at the risk of introducing unstable dynamics for bad choices of parameters), and introduces artificial energy loss. For highly constrained systems, the losses can dominate the resulting dynamics (see the scissor lift example in Sec. 5.1). Variational integrators work with holonomic constraints directly and completely avoid these problems.

The common approach to implementing a variational integration is to choose special coordinate representations that give simple, but scalable, equations. Equations of motion in generalized coordinates typically become unmanageable for large systems; this has been a major obstacle in using variational integrators for complex systems when generalized coordinates are desirable.

Continuous dynamics was once in a similar state. Equations of motion were either written explicitly, or algorithms that depend on specific coordinate choices were used. In particular, there is a long tradition of representing systems as collections of free bodies and adding constraints to impose structure [**42**]. This is still the dominant method when the coordinate choice is irrelevant (e.g. computer graphics, video games). The widely used Open Dynamics Engine (`ODE`) [**38**] takes this approach.

Controls applications, on the other hand, typically require generalized coordinates [**10**] so that important notions like controllability and observability are meaningful. For simple systems, we usually derive the full equations of motion, either by hand or with the help of software like Mathematica or Autolev [**36**]. This approach does not scale to large systems. Instead, we must turn to algorithmic methods of computing the dynamics at each time step.

Featherstone [**12**] developed methods that use a tree[1] representation of a mechanical system to calculate the (continuous) forward dynamics. The key to this approach are

[1]More generally, we may represent a system as a graph with tree descriptions as the subset of directed acyclic graphs [**8**]. However, the applications presented in this paper do not derive any immediate benefits to considering trees from a graph theoretic perspective, so we do not advocate this perspective here.

recursive equations that explicitly reuse information and avoid the repetition in the equations of motion. Several existing software packages (e.g. OpenHRP [**2**]) use this approach to simulate mechanical systems.

The same approach can be used to derive variational integrators in generalized coordinates. The discrete dynamics are calculated algorithmically instead of needing the full equations of motion. We use a different tree representation that is based on homogeneous coordinate transformations. The resulting algorithm is concise, transparent, and extendable. No modifications are required to handle branching, closed kinematic chains, or holonomic constraints. Other tree-based methods [**22**] also handle closed chains, but modify the dynamics to use iterative algorithms that require solving the inverse kinematics.

The tree representation framework has the additional advantage that it does extend to calculate the first- and second-order linearization of the variational integrator discrete dynamics. This allows a discrete adaptation of the projection operator trajectory optimization to work directly with variational integrators, completely eliminating the need for numeric integration. The linearizations are the exact derivatives of the dynamics.

An algorithmic approach provides another significant advantage in trajectory optimization. Numeric optimization algorithms typically converge to local, not global, minima. A good initial guess for the procedure is therefore critical if the optimization is to converge to the desired solution. An algorithmic approach to dynamics allows the system to be easily modified and augmented with additional inputs that lead to a much simpler optimization problem. By solving this problem, a suitable initial trajectory can often be found for the original system. This is particularly useful for a system like the marionette where the inputs are often used to excite dynamic modes that are not directly controllable by the strings.

1.1. Contribution of the Work

The primary contributions of this work are the development of the projection-based trajectory optimization in discrete time, the recursive development of variational integrators (similar to approaches such as Featherstone's algorithm[**12**]) for arbitrary, constrained systems in generalized coordinates, and the first and second derivatives of the variational integrator dynamics. This combination provides an approach that is capable optimizing trajectories of large dimensional systems with arbitrary inputs/actuation.

Other contributions include the mixed kinematic-dynamic modeling technique and algorithms to calculate the continuous time dynamics and linearizations from the same representation used by the variational integrator. The latter provides a unified framework where continuous and discrete approaches can be directly compared and enables applications to easily switch between the continuous and discrete time domains as appropriate.

The approach described by the thesis was specifically motivated by the automated marionette project, but the requirements of the project make this work applicable to the broader field of trajectory optimization. The primary advantages are that the process applies to arbitrary systems and is largely automated.

The ability to handle arbitrary systems is due to the optimization algorithm itself and the underlying approach to calculate the system dynamics. The (continuous and discrete time) dynamics and corresponding derivatives are calculated by a recursive algorithm that operates on a tree-based representation of the mechanical system. The representation uses generalized coordinates and is capable of describing most rigid body systems including those with closed kinematic chains. The optimization algorithm is equally versatile and works with arbitrary dynamics including under-actuated, over-actuated, and constrained systems.

The process is essentially automatic in the sense that the system designer does not need to consider properties like over/under-actuation, does not need to derive equations of motion, and does not need to choose many optimization parameters. The designer provides a mechanical description of the system, a desired trajectory, and the weights that determine the relative importance of each generalized coordinate. The dynamics and necessary derivatives are generated from the mechanical description without any further user input. The optimization itself does have extra parameters that must be tuned to each problem.

In addition to being well suited to systems with arbitrary actuation, the projection operator optimization has the advantage that every iteration results in a valid trajectory of the system. The algorithm can be terminated as soon as an acceptable trajectory is found. Algorithms such as DMOC only guarantee a valid trajectory once the optimization has formally converged and therefore cannot be stopped early. Not only does terminating early save time, but it also allows the user to get useful information even when the optimization cannot fully converge. Examining the non-converged results may provide clues for design changes that will improve the resulting trajectories.

The ability to work directly in discrete time is useful from an implementation perspective. The implementation is simpler by avoiding the need to approximate and interpolate continuous time curves. It avoids numeric integration, relying on root-solving or evaluating matrix equations to “integrate” forward in time. This abstracts the implementation from the optimization results–while different numeric integrators yield different solutions to an ODE, all root-finding algorithms will either find the same solution to a set of equations or fail. Additionally, since holonomic constraints are handled directly in discrete time, there is no need for fictitious restoring forces or damping. Ongoing research will continue to expand this benefit to applications that include impacts, contact, and unilateral constraints.

Finally, the algorithm used here to calculate the dynamics for arbitrary systems is not specific to the optimization algorithm. The algorithm calculates the generic dynamics function (i.e, $\dot{x} = f(x, u, t)$ or $x(k + 1) = f(x(k), u(k), k)$), and corresponding derivatives rather than a specialized representation. While the projection operator approach was chosen for the previously outlined reasons, the same algorithm and implementation can be used for any other trajectory optimization technique. Conversely, neither the continuous or discrete time projection operator optimizations specifically require this approach to the dynamics. The same optimization implementation can easily be used with any future improved algorithms that are capable of providing the dynamics and derivatives.

1.2. Outline of Thesis

Chapter 2 introduces a hierarchical tree representation for mechanical systems that serves as the basis for simulation and analysis in continuous and discrete mechanical systems. Continuous dynamics are discussed in Ch. 3, including constrained systems and the first- and second-order derivatives of the dynamics. Chapter 4 introduces variational integrators and describes an implementation for arbitrary constrained systems in generalized coordinates. This includes creating an explicit state space representation of the implicit variational integrator equations and deriving the discrete dynamic linearizations. The continuous and discrete time dynamics are compared in Ch. 5 with numeric examples. Trajectory optimization with projection operators is discussed in Ch. 6 by reviewing the continuous time optimization algorithm and deriving an equivalent algorithm for discrete time.

CHAPTER 2

Tree Representations of Mechanical Systems

Mechanical systems often have an inherent hierarchical structure–the wrist is affected by the forearm, and the forearm is affected by the upper arm. A tree representation formalizes this structure by attaching coordinate frames to bodies in a system, organizing them into parent-child relationships, and defining rigid body transformations from the parent to each child. Each frame has a single parent. The root of the tree–and only frame without a parent–is the stationary world frame s.

We use six transformation primitives, represented as elements of SE(3): translations along and rotations about the X, Y, and Z axes of the parent. If a frame is fixed relative to the parent, the transformation is parameterized by a constant. If the frame can move, it is parameterized by a unique configuration variable. The set of all configuration variables forms the generalized coordinates for the system. Complex joints are modeled by compositions of the six transformations. More complex joints (e.g. screws, cams, gears) are modeled with compositions and additional constraints.

Fig. 2.1 is an example tree description for a planar human. Note that coordinate frames do not necessarily have to have an associated mass and so we can define as many frames as desired. Consequently, a tree representation is not unique, there are an infinite number of representations for a given system.

Fig. 2.1 also demonstrates the naming convention used throughout this chapter. Constant transformations are indexed with letters (e.g., $g_A(1.0)$). Variable transformations are

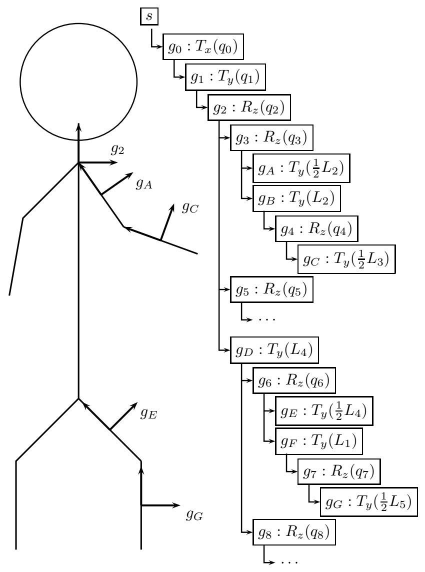


Figure 2.1. A planar human is represented with a tree structure. Although we are restricted to simple transformations, we can represent complex mechanical systems by composing multiple transformations.

indexed by numbers and are parameterized by the corresponding configuration variable (e.g., $g_1(q_1)$).

We impose the following requirements on the tree description:

R1. Frames are related to their parents through six basic transformations: Translations along the parent's X, Y, and Z axes and rotations about the parent's X, Y, and Z axes.

R2. Each configuration variable is used in only one transformation and each transformation only depends on one parameter or configuration variable.

Together, these two requirements establish a one-to-one correspondence between configuration variables and variable transformations. They are not absolutely necessary for this approach, but they significantly simplify the discussion, notation, and equations without restricting the systems that can be represented.

The basic transformations defined by R1 are parameterized by single, real-valued numbers: configuration variables for moving joints and constant numbers for fixed joints. This requirement leads to simplifications and provides uniform generalized coordinates consisting only of real-valued numbers.

Note that R1 disallows direct SO(3) parameterizations for free rotations. Instead, we represent free rotation with multiple single-axis rotations (e.g. Euler angles). This requirement can be removed if one is willing to handle the extra book-keeping involved.[1]

R2 establishes a one-to-one relationship between configuration variables and parameterized transformations to greatly simplify the equations and notation that will be discussed. Using the same configuration variable to drive multiple transformations is useful for modeling systems with parallel linkages. Parallel linkages and closed chains are instead modeled by open chains with holonomic constraints.

The following list defines the notation used throughout the chapter. We will typically drop the explicit dependence on q and q_k for visual clarity (e.g. $g_{s,k}$ rather than $g_{s,k}(q)$). The definitions refer to the example in Fig. 2.2

[1]SO(3) parameterizations require constraints on the nine matrix elements to ensure the SO(3) matrix remains orthonormal [**30**].

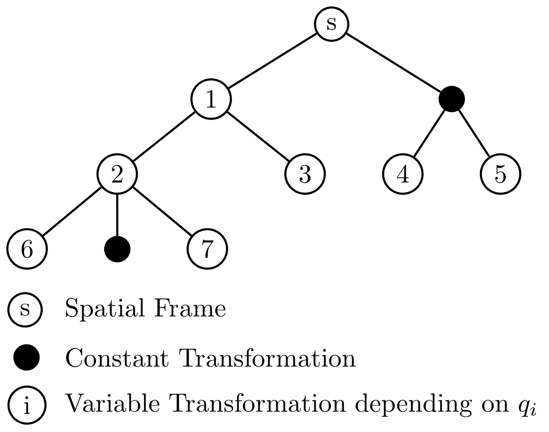


Figure 2.2. Notation used for the tree representation.

$q_i(t) \in \mathbb{R}$**:** Configuration variable of the i-th frame.

$\dot{q}_i(t) \in \mathbb{R} = \frac{d}{dt} q_i(t)$**:** Time derivative of the i-th configuration variable.

$q(t) \in \mathbb{R}^n$**:** Configuration vector comprising q_0, q_1, ... , q_n.

m_i**:** Mass of the i-th frame.

$M_i \in \mathbb{R}^{6x6}$**:** Inertia Tensor of the i-th mass in body-fixed coordinates.

$g_i(q_i) \in \mathrm{SE}(3)$**:** Transformation of the i-th frame to its parent frame.

$g_{s,i}(q) \in \mathrm{SE}(3)$ **:** Transformation from the i-th frame to the spatial reference frame s.

$g_{j,i}(q) \in \mathrm{SE}(3)$ **:** Transformation the i-th frame to the j-th frame.

$v^b_{s,i} \in T_e SE(3)$ **:** Body velocity (i.e., an element of the Lie algebra) of the i-th frame relative to the spatial reference frame s.

$p_{s,i}(q) \in \mathbb{R}^3$**:** Position of the i-th frame relative to the spatial reference frame.

anc(i) **:** The ancestors of the the i-th frame are the frames passed while moving up the tree from the i-th frame to the spatial frame. For example, frame 6 (above) has ancestors $\{2, 1, s\}$.

par(i) **:** Immediate parent frame of the i-th frame, for example, the parent of frame 6 (above) is frame 2.

$g'_k = \dfrac{\partial g_k}{\partial q_k}$**:** The derivative of a transformation of the i-th frame to its parent frame.

$g''_k = \dfrac{\partial^2 g_k}{\partial q_k \partial q_k}$ **:** The second derivative of a transformation of the i-th frame to its parent frame.

$\dot{g}_k = \dfrac{dg_k}{dt}$ **:** The time derivative of a transformation of the i-th frame to its parent frame.

$(\cdot)^\wedge : \mathbb{R}^6 \to T_eSE(3)$ **:** The 'hat' operator maps twists to the $T_eSE(3)$.

$(\cdot)^\vee : T_eSE(3) \to \mathbb{R}^6$ **:** The 'unhat' operator maps elements of $T_eSE(3)$ to twists in $\mathbb{R}^6$

A tree representation by itself is a clean organization of a system's mechanical structure, but it also provides a foundation for efficiently calculating quantities needed for dynamic simulation and analysis. In the following section, we present methods for computing the position, velocities, and their derivatives for all frames in a tree.

Note: The piecewise equations that follow in this chapter have a specific ordering. For cases that can be simultaneously satisfied, the top-most case always takes precedence.

2.1. Frame Positions

The rigid body transformation from any frame in a tree to the spatial reference frame is a straightforward calculation. For the spatial reference frame, it is the identity transformation, I. Otherwise, for frame k, it is the parent frame's transformation $g_{s,\mathrm{par}(k)}$ composed with the local transformation g_k. This is expressed as a piecewise expression for $g_{s,k}(q)$:

$$g_{s,k}(q) = \begin{cases} I & k = s \\ \\ g_{s,\mathrm{par}(k)} g_k & k \neq s \end{cases} \tag{2.1}$$

Eq. (2.1) evaluates the transformation from any frame to the spatial frame using only the local transformations and itself recursively. The recursion is guaranteed to terminate on the $k = s$ condition because the tree is acyclic by definition. Each recursive call moves towards the root node. The same is true for the remaining recursive equations that are derived from the tree.

The compact form of (2.1) is particularly nice because it leads to derivatives with similarly simple forms. The first derivative with respect to a configuration variable q_i is found with the standard derivative:

$$\begin{aligned} \frac{\partial g_{s,k}}{\partial q_i}(q) &= \begin{cases} \dfrac{\partial}{\partial q_i} I & k = s \\ \dfrac{\partial}{\partial q_i}\left(g_{s,\mathrm{par}(k)} g_k\right) & k \neq s \end{cases} \\ &= \begin{cases} 0 & k = s \\ \\ \dfrac{\partial}{\partial q_i}\left(g_{s,\mathrm{par}(k)}\right) g_k + g_{s,\mathrm{par}(k)} \dfrac{\partial}{\partial q_i} g_k & k \neq s \end{cases} \end{aligned}$$

The two requirements, R1 and R2, help to further simplify this expression. R1 guarantees $\frac{\partial}{\partial q_i} g_k = 0$ when $i \neq k$. R2 implies $\frac{\partial}{\partial q_i} g_{s,\mathrm{par}(i)} = 0$ since $g_i(q)$ is the only transformation that depends on q_i. Furthermore, we know that if g_i is not an ancestor of g_k, then the derivative is always zero:

$$\frac{\partial g_{s,k}}{\partial q_i}(q) = \begin{cases} 0 & \begin{matrix} k = s \\ i \notin \mathrm{anc}(k) \end{matrix} \\ \frac{\partial g_{s,\mathrm{par}(k)}}{\partial q_i} g_k & i \neq k \\ g_{s,\mathrm{par}(k)} g'_k & i = k \end{cases} \tag{2.2}$$

where $g'_k = \frac{d}{dq_k} g_k(q_k)$. Using (2.1) and (2.2), we can numerically compute the *exact* derivative of any coordinate frame with respect to any configuration variable in the system. The mixed partial derivative with respect to q_i and q_j is calculated using the same ideas:

$$\frac{\partial^2 g_{s,k}}{\partial q_j \partial q_i}(q) = \begin{cases} 0 & \begin{matrix} k = s \\ i \notin \mathrm{anc}(k) \\ j \notin \mathrm{anc}(k) \end{matrix} \\ g_{s,\mathrm{par}(k)} g''_k & i = k = j \\ \frac{\partial g_{s,\mathrm{par}(k)}}{\partial q_j} g'_k & i = k \neq j \\ \frac{\partial g_{s,\mathrm{par}(k)}}{\partial q_i} g'_k & i \neq k = j \\ \frac{\partial^2 g_{s,\mathrm{par}(k)}}{\partial q_j \partial q_i} g_k & i \neq k, j \neq k \end{cases} . \tag{2.3}$$

Eq. (2.3) uses itself recursively along with (2.1) and (2.2) to evaluate second derivatives of the rigid body transformations. These are the only derivatives needed to calculate. Higher derivatives required for linearizations of the dynamics are presented in Sec. 2.4.

2.1.1. Inverse Derivatives

At times the derivative of a frame position's inverse is also required. For example, the first and second dynamic linearizations of systems with wrench forces (See Sec. 3.1.3.1) require derivatives of the Jacobian. These derivatives are calculated from the position equations. By differentiating the definition of the derivative $g^{-1}g = \mathcal{I}$, the derivative of the inverse is found to be:

$$\frac{\partial g^{-}1}{\partial q_i} = g^{-1}\frac{\partial g}{\partial q_i}g^{-1}.$$

Higher derivatives are also found this way.

2.2. Body Velocity

The body velocity is the relative motion of a coordinate frame with respect to the stationary world frame, but expressed in the frame's local coordinates. It is calculated as the velocity of the parent frame, transformed into local coordinates, plus the velocity of the frame with respect to the parent, $g_k^{-1}\dot{g}_k$:

$$\hat{v}^b_{s,k}(q,\dot{q}) = \begin{cases} 0 & k = s \\ g_k^{-1}\hat{v}^b_{s,\mathrm{par}(k)}g_k + g_k^{-1}\dot{g}_k & k \neq s \end{cases} \tag{2.4}$$

where $\dot{g}_k = \frac{d}{dt}g_k$. For the transformations defined in R1, the local velocity term $g_k^{-1}\dot{g}_k$ reduces to a special form using a twist $\hat{\xi}$ [**30**]. The twist is constant (i.e. it does not depend on the configuration) for each type of transformation:

$$\hat{v}^b_{s,k}(q,\dot{q}) = \begin{cases} 0 & k = s \\ g_k^{-1}\hat{v}^b_{s,\mathrm{par}(k)}g_k + \hat{\xi}_k\dot{q}_k & k \neq s \end{cases}. \tag{2.5}$$

Taking the same approach used earlier, we find derivative expressions:

$$\frac{\partial \hat{v}^b_{s,k}}{\partial q_i}(q,\dot{q}) = \begin{cases} 0 & \begin{matrix} k = s \\ i \notin \mathrm{anc}(k) \end{matrix} \\ g_k^{-1\prime}\hat{v}^b_{s,\mathrm{par}(k)}g_k + g_k^{-1}\hat{v}^b_{s,\mathrm{par}(k)}g_k' & i = k \\ g_k^{-1}\dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_i}g_k & i \neq k \end{cases}. \tag{2.6}$$

Eq. (2.6) is evaluated recursively with itself, the local rigid body transformations, and (2.5):

$$\frac{\partial^2 \hat{v}^b_{s,k}(q,\dot{q})}{\partial q_j \partial q_i} = \begin{cases} 0 & \begin{matrix} k = s \\ i \notin \mathrm{anc}(k) \\ j \notin \mathrm{anc}(k) \end{matrix} \\ \begin{matrix} g_k^{-1\prime\prime}\hat{v}^b_{s,\mathrm{par}(k)}g_k + 2g_k^{-1\prime}\hat{v}^b_{s,\mathrm{par}(k)}g_k' \\ \quad + g_k^{-1}\hat{v}^b_{s,\mathrm{par}(k)}g_k'' \end{matrix} & i = k = j \\ g_k^{-1\prime}\dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_i}g_k + g_k^{-1}\dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_i}g_k' & i \neq k = j \\ g_k^{-1\prime}\dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_j}g_k + g_k^{-1}\dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_j}g_k' & i = k \neq j \\ g_k^{-1}\dfrac{\partial^2 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_j \partial q_i}g_k & i \neq k, j \neq k \end{cases}. \tag{2.7}$$

Eq. (2.7) is evaluated using itself, local rigid body transformations, (2.5), and (2.6). We are also interested in derivatives of body velocities with respect to $\dot{q}$:

$$\frac{\partial \hat{v}^b_{s,k}}{\partial \dot{q}_i}(q, \dot{q}) = \begin{cases} 0 & \begin{array}{l} k = s, \\ i \notin \operatorname{anc}(k) \end{array} \\ \hat{\xi}_k & i = k \\ g_k^{-1} \dfrac{\partial \hat{v}^b_{s,\operatorname{par}(k)}}{\partial \dot{q}_i} g_k & i \neq k \end{cases} \tag{2.8}$$

$$\frac{\partial^2 \hat{v}^b_{s,k}}{\partial \dot{q}_j \partial \dot{q}_i}(q, \dot{q}) = 0. \tag{2.9}$$

We can also find mixed partial derivatives with respect to configuration variables q_i and their time derivatives $\dot{q}_i$:

$$\frac{\partial^2 \hat{v}^b_{s,k}}{\partial \dot{q}_j \partial q_i}(q, \dot{q}) = \begin{cases} 0 & \begin{array}{l} k = s \\ k = j \\ i \notin \operatorname{anc}(k) \\ j \notin \operatorname{anc}(k) \end{array} \\ g_k^{-1\prime} \dfrac{\partial \hat{v}^b_{s,\operatorname{par}(k)}}{\partial \dot{q}_j} g_k + g_k^{-1} \dfrac{\partial \hat{v}^b_{s,\operatorname{par}(k)}}{\partial \dot{q}_j} g_k' & k = i \\ g_k^{-1} \dfrac{\partial^2 \hat{v}^b_{s,\operatorname{par}(k)}}{\partial \dot{q}_j \partial q_i} g_k & k \neq i \end{cases} \tag{2.10}$$

Eq. (2.1) through (2.10) demonstrate how to calculate the forward kinematics and derivatives from a tree description. As will be discussed later, these equations provide all the necessary values for simulating the system dynamics. Other applications, such as trajectory exploration in optimal control or nonlinear controllability analysis, may require higher derivatives. That these are discussed in (2.4.2).

2.3. Primitive Transformations

Eq. (2.1) through (2.10) include terms that we have not explicitly shown how to calculate (i.e. g_k, g'_k, g''_k, g_k^{-1}, $g_k^{-1\prime}$, $g_k^{-1\prime\prime}$, and ξ_k). These are found manually for each of the primitive transforms defined in R1. For example, the values for a rotation about the Z axis are:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R'_z(\theta) = \begin{bmatrix} -\sin\theta & -\cos\theta & 0 & 0 \\ \cos\theta & -\sin\theta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Higher derivatives for the six transformations in R1 can be parameterized easily. The derivatives of translations are zero for the second derivatives and higher. The derivatives of rotations are cyclic (e.g. $R^{(4)} = R$). We can therefore find the expression for the n^{th} derivative arbitrarily.

For the six transformations given in R1, the inverse is always the same transformation but by the opposite amount ($g^{-1}(x) = g(-x)$):

$$\begin{aligned} R_x^{-1}(\theta) &= R_x(-\theta) \\ R_x^{-1\prime}(\theta) &= -R'_x(-\theta) \\ R_x^{-1\prime\prime}(\theta) &= R''_x(-\theta) \end{aligned}$$

The twist ξ_{R_z} takes slightly more work.

$$\begin{aligned}\hat{\xi}_{R_z}\dot{\theta} &= R_z^{-1}(\theta)\dot{R}_z(\theta) = R_z^{-1}(\theta)R_z'(\theta)\dot{\theta} \\ \hat{\xi}_{R_z} &= R_z^{-1}(\theta)R_z'(\theta) \\ \Rightarrow \xi_{R_z} &= \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}^T\end{aligned}$$

Later equations also make use of the position of a coordinate frame's origin relative to the spatial world frame. The position $p_{s,k}$ of a frame is directly obtained by extracting the x, y, and z components from the corresponding SE(3) $g_{s,k}$ transformation in (2.1). Derivatives of $p_{s,k}$ are similarly extracted from the corresponding derivative of $g_{s,k}$ in (2.2) and (2.3).

2.4. Higher Derivatives

The derivatives of the frame positions and body velocities in Sec. 2.1 and Sec. 2.2 are sufficient for calculating continuous or discrete dynamics. Higher derivatives are required for the first and second dynamic linearizations.

2.4.1. Frame Positions

Higher derivatives of the frame positions are required for the linearizations of the dynamics discussed in Sec. 3 and Sec. 4. The third derivative of $g_{s,k}$ is calculated by (2.11). The number of cases tested in the equation has been reduced by taking advantage of the symmetry provided by mixed partials commuting. Each condition of the piecewise equation applies to all permutations of i, j, ℓ. For example, if only one of the variables i, j, k

is equal to k, the equation for $i = k$, $j \neq k$, $\ell \neq k$ is used with a permutation swapping that variable and i:

$$\frac{\partial^3 g_{s,k}}{\partial q_\ell \partial q_j \partial q_i}(q) = \begin{cases} 0 & \begin{array}{l} k = s \\ i \notin \operatorname{anc}(k) \\ j \notin \operatorname{anc}(k) \\ \ell \notin \operatorname{anc}(k) \end{array} \\ \dfrac{\partial^3 g_{s,\mathrm{par}(k)}}{\partial q_\ell \partial q_j \partial q_i} g_k & i \neq k,\, j \neq k,\, \ell \neq k \\ \dfrac{\partial^2 g_{s,\mathrm{par}(k)}}{\partial q_\ell \partial q_j} g'_k & i = k,\, j \neq k,\, \ell \neq k \\ \dfrac{\partial g_{s,\mathrm{par}(k)}}{\partial q_\ell} g''_k & i = k,\, j = k,\, \ell \neq k \\ g_{s,\mathrm{par}(k)} g'''_k & i = k,\, j = k,\, \ell = k \end{cases}. \tag{2.11}$$

The fourth derivative is required for the second linearization of the continuous dynamics[2]. The fourth derivative is calculated using (2.12). This piecewise equation is interpreted in the same way as (2.12):

[2]The discrete dynamics do not require this derivative. This is due to variational integrators working directly with holonomic constraints while the continuous dynamics uses equivalent non-holonomic constraints.

$$\frac{\partial^4 g_{s,k}}{\partial q_m \partial q_\ell \partial q_j \partial q_i}(q) = \begin{cases} 0 & \begin{array}{l} k = s \\ i \notin \operatorname{anc}(k) \\ j \notin \operatorname{anc}(k) \\ \ell \notin \operatorname{anc}(k) \\ m \notin \operatorname{anc}(k) \end{array} \\ \dfrac{\partial^4 g_{s,\mathrm{par}(k)}}{\partial q_m \partial q_\ell \partial q_j \partial q_i} g_k & i \neq k,\, j \neq k,\, \ell \neq k,\, m \neq k \\ \dfrac{\partial^3 g_{s,\mathrm{par}(k)}}{\partial q_m \partial q_\ell \partial q_j} g'_k & i = k,\, j \neq k,\, \ell \neq k,\, m \neq k \\ \dfrac{\partial^2 g_{s,\mathrm{par}(k)}}{\partial q_m \partial q_\ell} g''_k & i = k,\, j = k,\, \ell \neq k,\, m \neq k \\ \dfrac{\partial g_{s,\mathrm{par}(k)}}{\partial q_m} g'''_k & i = k,\, j = k,\, \ell = k,\, m \neq k \\ g_{s,\mathrm{par}(k)} g''''_k & i = k,\, j = k,\, \ell = k,\, m = k \end{cases} . \tag{2.12}$$

Higher derivatives of the frame position are not required for the first or second linearizations of the continuous or discrete dynamics. However, it is worth noting that this procedure could be repeated to get higher and higher derivatives as required for a particular problem.

2.4.2. Body Velocity

The third derivative of the body velocity with respect to configuration variables is calculated with (2.13). This equation has taken advantage of symmetry to reduce the number of cases listed. The piecewise tests apply for any permutation of i, j, and ℓ:

$$\frac{\partial^3 \hat{v}^b_{s,k}}{\partial q_\ell \partial q_j \partial q_i}(q,\dot{q}) =$$

$$\begin{cases} 0 & \begin{array}{l} k = s \\ i \notin \mathrm{anc}(k) \\ j \notin \mathrm{anc}(k) \\ \ell \notin \mathrm{anc}(k) \end{array} \\ g_k^{-1} \dfrac{\partial^3 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_\ell \partial q_j \partial q_i} g_k & i \neq k, j \neq k,\ \ell \neq k \\ g_k^{-1\prime} \dfrac{\partial^2 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_\ell \partial q_j} g_k + g_k^{-1} \dfrac{\partial^2 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_\ell \partial q_j} g_k' & i = k,\ j \neq k, \ell \neq k \\ g_k^{-1\prime\prime} \dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_\ell} g_k + 2 g_k^{-1\prime} \dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_\ell} g_k' + g_k^{-1} \dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_\ell} g_k'' & i = k,\ j = k, \ell \neq k \\ \begin{array}{l} g_k^{-1\prime\prime\prime} \hat{v}^b_{s,\mathrm{par}(k)} g_k + 3 g_k^{-1\prime\prime} \hat{v}^b_{s,\mathrm{par}(k)} g_k' + \\ \qquad\qquad 3 g_k^{-1\prime} \hat{v}^b_{s,\mathrm{par}(k)} g_k'' + g_k^{-1} \hat{v}^b_{s,\mathrm{par}(k)} g_k''' \end{array} & i = k,\ j = k\, \ell = k \end{cases} . \quad (2.13)$$

The third derivative of the body velocity with respect to one configuration velocity and two configuration variables is calculated with (2.14). This equation has taken advantage of symmetry to reduce the number of cases listed. The piecewise tests apply for any permutation of j and ℓ:

$$\frac{\partial^3 \hat{v}^b_{s,k}}{\partial q_\ell \partial q_j \partial \dot{q}_i}(q,\dot{q}) =$$

$$\begin{cases} 0 & \begin{array}{l} k = s, \\ i \notin \mathrm{anc}(k) \\ j \notin \mathrm{anc}(k) \\ \ell \notin \mathrm{anc}(k) \end{array} \\ g_k^{-1} \dfrac{\partial^3 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_\ell \partial q_j \partial \dot{q}_i} g_k & \begin{array}{l} i = k \\ j \neq k,\, \ell \neq k \end{array} \\ g_k^{-1\prime} \dfrac{\partial^2 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_\ell \partial \dot{q}_i} g_k + g_k^{-1} \dfrac{\partial^2 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_\ell \partial \dot{q}_i} g'_k & j = k, \ell \neq k \\ g_k^{-1\prime\prime} \dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial \dot{q}_i} g_k + 2 g_k^{-1\prime} \dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial \dot{q}_i} g'_k + g_k^{-1} \dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial \dot{q}_i} g''_k & j = k, \ell = k \end{cases} . \quad (2.14)$$

The fourth derivative of the body velocity with respect to one configuration velocity and three configuration variables is calculated with (2.15). Symmetry has also been used to reduce the number of cases listed. The piecewise tests apply for any permutation of j, ℓ, and m:

$$\frac{\partial^4 \hat{v}^b_{s,k}}{\partial q_m \partial q_\ell \partial q_j \partial \dot{q}_i}(q,\dot{q}) =$$
$$\begin{cases} 0 & \begin{array}{l} k = s, \\ i \notin \mathrm{anc}(k) \\ j \notin \mathrm{anc}(k) \\ \ell \notin \mathrm{anc}(k) \\ m \notin \mathrm{anc}(k) \\ i = k \end{array} \\ g_k^{-1} \dfrac{\partial^4 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_m \partial q_\ell \partial q_j \partial \dot{q}_i} g_k & j \neq k,\ \ell \neq k, m \neq k \\ g_k^{-1\prime} \dfrac{\partial^3 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_m \partial q_\ell \partial \dot{q}_i} g_k + g_k^{-1} \dfrac{\partial^3 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_m \partial q_\ell \partial \dot{q}_i} g_k' & j = k, \ell \neq k, m \neq k \\ g_k^{-1\prime\prime} \dfrac{\partial^2 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_m \partial \dot{q}_i} g_k + 2 g_k^{-1\prime} \dfrac{\partial^2 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_m \partial \dot{q}_i} g_k' + g_k^{-1} \dfrac{\partial^2 \hat{v}^b_{s,\mathrm{par}(k)}}{\partial q_m \partial \dot{q}_i} g_k'' & j = k,\ \ell = k, m \neq k \\ g_k^{-1\prime\prime\prime} \dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial \dot{q}_i} g_k + 3 g_k^{-1\prime\prime} \dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial \dot{q}_i} g_k' + \\ \qquad 3 g_k^{-1\prime} \dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial \dot{q}_i} g_k'' + g_k^{-1} \dfrac{\partial \hat{v}^b_{s,\mathrm{par}(k)}}{\partial \dot{q}_i} g_k''' & j = k, \ell = k, m = k \end{cases} \cdot \tag{2.15}$$

As with the frame positions, higher derivatives of the body velocity can be calculated using this method if required for a problem.

2.5. Relaxing the Restrictions

Restriction R2 was originally introduced to improve performance by reducing the amount of computation required for the derivatives[3]. R2 can be relaxed to allow a frame

[3]It is also useful for introducing the algorithm as it simplifies notation considerably.

to depend on multiple configuration variables. In this case, the initial equations for position (2.1) and body velocity (2.4) do not change. They are differentiated as before, but more cases must be considered when simplifying the equations. It is possible to do this without significantly impacting performance.

Alternatively, R2 can be relaxed or removed completely to allow multiple frames to use the same configuration variable. Unfortunately this will significantly impact performance by severely limiting the number of terms that are guaranteed to be zero.

The restricted primitives in R1 can be relaxed to allow arbitrary constant transformations or any single parameter transformation generated from a twist. If R2 has been relaxed as discussed, then this restriction can be removed completely.

2.5.1. Caching

Values are frequently reused in the above equations. For example, $g_{s,k}(q)$ may be evaluated once for itself, once for each of its descendants' positions, and then again for derivative values. However, once the value is evaluated, it is constant until a new configuration is written to the tree. By saving and reusing results until a new configuration is written, we avoid recursing all the way to the base of the tree in each calculation, essentially flattening (2.1) through (2.10). This technique, called caching, improves performance well enough to use the tree representation for fast dynamic simulation for large mechanical systems. In the following section, we introduce a variational integrator based on the tree representation.

CHAPTER 3

Continuous Dynamics

There are many existing algorithms for calculating the continuous dynamics of a mechanical system in generalized coordinates, including many with linear complexity with respect to system dimensions. These approaches typically calculate the acceleration of the configuration without directly calculating the Lagrangian or its derivatives. Such algorithms are not suitable platforms for an arbitrary variational integrator in generalized coordinates.

This section develops an alternative dynamics algorithm that calculates the dynamics in terms of the Lagrangian and the Euler-Lagrange equation. This approach scales substantially better than working directly from symbolic equations of motion, but has inferior complexity to existing $O(n)$ algorithms. The primary purpose is to develop necessary ground work for the variational integrator discussed in Ch. 4.

In addition to supporting the variational integrator here, the approach does lead to efficient ways of numerically calculating the exact first- and second-order linearizations of the continuous dynamics. This a benefit over $O(n)$ algorithms which do not provide this capability, and permits optimal control and analysis applications that are typically used with symbolic equations of motion, for more complex systems.

Section 3.1 discusses continuous Euler-Lagrange dynamics beginning with a brief review of the Least Action Principle and Euler-Lagrange equation. The dynamics are extended to include external forcing and constraints in Sec. 3.1.3 and Sec. 3.1.4. Each section describes how the required values are calculated from the tree representation in Ch. 2.

A state space representation for the continuous dynamics is chosen in Sec. 3.2. This representation defines the first- and second-order linearizations, discussed in Sec. 3.3 and Sec. 3.4 for unconstrained systems, respectively. Section 3.5 develops the constrained linearizations. Finally, Section 3.6 introduces an extension to the continuous dynamics to include kinematic reductions.

3.1. Continuous Euler-Lagrange Dynamics

Lagrangian mechanics provide a coordinate-invariant method for generating a system's dynamic equations. Lagrangian mechanics are derived from a variational principle. The Lagrangian of a system is the total kinetic energy minus potential energy in terms of generalized coordinates $q = [q_1, q_2, \ldots, q_n] \in \mathbb{R}^n$ and the coordinates' time derivative $\dot{q} = \left[\frac{\partial q_1}{\partial t}, \frac{\partial q_2}{\partial t}, \ldots, \frac{\partial q_n}{\partial t}\right]$:

$$L\left(q, \dot{q}\right) = KE(q, \dot{q}) - PE(q). \tag{3.1}$$

The Lagrangian is discussed further in Sec. 3.1.1.

The integral of the Lagrangian over a trajectory is called the Action (Fig. 3.1):

$$S(q([t_0, t_f])) = \int_{t_0}^{t_f} L(q(\tau), \dot{q}(\tau))\mathrm{d}\tau. \tag{3.2}$$

Hamilton's Least Action Principle states that a mechanical system will naturally follow the trajectory that extremizes (e.g. minimizes) the action. Extremizing (3.2) with a

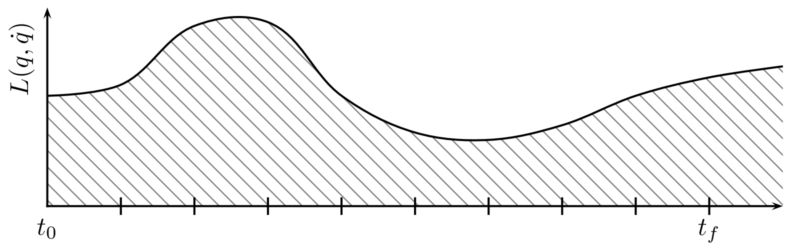


Figure 3.1. Continuous Action Integral

variational principle shows that such trajectories satisfy the Euler-Lagrange equation:

$$\frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{q}}(q,\dot{q}) - \frac{\partial L}{\partial q}(q,\dot{q}) = 0 \tag{3.3}$$

which is a second-order ordinary differential equation (ODE) in q. Given a set of initial conditions $(q(t_0),\dot{q}(t_0))$, we numerically integrate (3.3) to simulate the system dynamics.

The Euler-Lagrange equation (3.3) is often written in the standard form:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + V(q) = 0 \tag{3.4}$$

This form is useful for analysis as important properties can be found from the Coriolis term, $C(q,\dot{q})$. Typically, the terms $M(q)$, $C(q,\dot{q})$, and $V(q)$ are found symbolically and the equation is integrated by evaluating the symbolic expressions at each configuration. For complex systems, the symbolic expressions become prohibitively large. Symbolic algebra software packages are essentially required and yet the sheer size of the equations leads to tremendous memory requirements and extremely slow evaluations.

We avoid these problems by ignoring the special form and making appropriate use of the chain rule. The result is a set of simple equations that apply to arbitrary rigid body systems and evaluate each other numerically rather than working with large system-specific symbolic equations. To begin, we expand (3.3) with the chain rule:

$$\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}} \ddot{q} + \frac{\partial^2 L}{\partial q \partial \dot{q}} \dot{q} - \frac{\partial L}{\partial \dot{q}} = 0 \tag{3.5}$$

Equation (3.5) differs from (3.4) in that each term can be found numerically for an arbitrary rigid body system whereas terms of (3.4) must be found from the system-specific symbolic differential equations. Note that $M(q) = \dfrac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}$ so we are still able to find the numeric value of the inertia matrix at any configuration. Once each term in (3.5) is calculated, we solve for $\ddot{q}$:

$$\ddot{q} = -\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[\frac{\partial^2 L}{\partial q \partial \dot{q}} \dot{q} - \frac{\partial L}{\partial \dot{q}}\right] \tag{3.6}$$

The following section describes how the tree representation from Ch. 2 provides a way to numerically calculate the *exact* derivatives of the Lagrangian in (3.5). We calculate the exact continuous dynamics without resorting to symbolic expressions by using these values in (3.6).

3.1.1. The Continuous Lagrangian

The Lagrangian (3.1) requires the kinetic and potential energies of the system. If we define a coordinate frame at each center of mass and align the axes with the body's principal axes [**9**][**33**], the inertia matrix is a constant 6x6 diagonal matrix and the kinetic energy for the k-th mass is written as $T_k(q, \dot{q}) = \frac{1}{2} v^{bT}_{s,k} M_k v^b_{s,k}$ (recalling that v^b is the body velocity) where the inertia matrix is

$$M_k = \begin{bmatrix} m_k & 0 & 0 & 0 & 0 & 0 \\ 0 & m_k & 0 & 0 & 0 & 0 \\ 0 & 0 & m_k & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx,k} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy,k} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zz,k} \end{bmatrix}$$

The resulting Lagrangian is:

$$L(q, \dot{q}) = \sum_k \tfrac{1}{2} v_{s,k}^{bT} M_k v_{s,k}^b - \sum_\ell V_\ell(q) \tag{3.7}$$

where $V_\ell(q)$ is a potential energy in the system. Common potentials include gravity and springs. Assuming we can evaluate $V_\ell(q)$, (3.7) can be evaluated numerically from the tree representation with (2.5). We continue to assume that $V_\ell(q)$ and its derivatives are known. Their actual computation is discussed in Sec. 3.1.2.

We continue by finding the necessary derivatives of the continuous Lagrangian (3.7):

$$\begin{aligned} \frac{\partial L}{\partial q_i} &= \frac{\partial}{\partial q_i}\left[\sum_k \tfrac{1}{2} v_{s,k}^{bT} M_k v_{s,k}^b - \sum_k V_k(q)\right] \\ &= \sum_k v_{s,k}^{bT} M_k \frac{\partial v_{s,k}^b}{\partial q_i} - \sum_k \frac{\partial V_k}{\partial q_i}(q) \end{aligned} \tag{3.8}$$

Eq. (3.8) is evaluated from the tree structure using (2.5) and (2.6). The remaining derivatives are found similarly.

$$\frac{\partial^2 L}{\partial q_i \partial q_j} = \sum_k \left[\frac{\partial v_{s,k}^{bT}}{\partial q_j} M_k \frac{\partial v_{s,k}^b}{\partial q_i} + v_{s,k}^{bT} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_i \partial q_j} \right] - \sum_k \frac{\partial^2 V_k}{\partial q_i \partial q_j}(q) \tag{3.9}$$

Eq. (3.9) is evaluated from the tree description using (2.5), (2.6), and (2.7).

$$\frac{\partial L}{\partial \dot{q}_i} = \sum_k v_{s,k}^{bT} M_k \frac{\partial v_{s,k}^b}{\partial \dot{q}_i} \tag{3.10}$$

Eq. (3.10) is evaluated from the tree using (2.5) and (2.8).

$$\frac{\partial^2 L}{\partial q_i \partial \dot{q}_j} = \sum_k \left[\frac{\partial v_{s,k}^{bT}}{\partial \dot{q}_j} M_k \frac{\partial v_{s,k}^b}{\partial q_i} + v_{s,k}^{bT} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_i \partial \dot{q}_j} \right] \tag{3.11}$$

Eq. (3.11) is evaluated from the tree structure using (2.5), (2.6), (2.8), and (2.10).

$$\frac{\partial^2 L}{\partial \dot{q}_i \partial \dot{q}_j} = \sum_k \left[\frac{\partial v_{s,k}^{bT}}{\partial \dot{q}_j} M_k \frac{\partial v_{s,k}^b}{\partial \dot{q}_i} + v_{s,k}^{bT} M_k \frac{\partial^2 v_{s,k}^b}{\partial \dot{q}_i \partial \dot{q}_j} \right] \tag{3.12}$$

Eq. (3.12) is evaluated from the tree using (2.5), (2.8), and (2.9). Once (3.8) - (3.12) can be evaluated, we can completely evaluate (3.5). Thus, we can calculate the continuous dynamics for an arbitrary tree description.

3.1.2. Potential Energy

Potential energies are included in the model simulation through the generalized terms $V(q)$ and their derivatives. Each type of potential energy has a different form for $V(q)$. These are implemented manually, but in a way that uses the tree calculations and makes them applicable to arbitrary systems. This technique provides a great deal of flexibility for including potentials.

Common potential energies encountered in mechanical systems are gravity and linear springs. The following sections show how both can be defined in terms of coordinate frames in the tree. This approach allows each potential to be used in arbitrary systems.

3.1.2.1. Gravity. We commonly use the simple model of gravity for mechanical systems: $F = m\vec{g}$ where $\vec{g}$ is the gravity vector, typically $[0\ 0\ \ -9.81]^T$. The potential created by this force as applied to a mass at point $p_{s,k}(q)$ is:

$$V(q) = -m_k\vec{g} \cdot p_{s,k} \tag{3.13}$$

The two derivatives are straightforward:

$$\frac{\partial V}{\partial q_i}(q) = -m_k\vec{g} \cdot \frac{\partial}{\partial q_i} p_{s,k} \tag{3.14}$$

$$\frac{\partial^2 V}{\partial q_i \partial q_j}(q) = -m_k\vec{g} \cdot \frac{\partial^2}{\partial q_i \partial q_j} p_{s,k} \tag{3.15}$$

Eq. (3.13), (3.14), and (3.15) are evaluated using (2.1), (2.2), and (2.3) from a tree description. Typically, we would automatically add a gravity potential for every mass in the system. For more exotic simulations, however, we might selectively add this gravity model for some masses and a different model for others.

This approach can be taken for the nonlinear gravity model ($F = -G\frac{m_1 m_2}{r^2}\hat{r}$).

3.1.2.2. Linear Springs. Suppose we have a linear spring with spring constant k and natural length x_0 connecting two points p_1 and p_2. The potential energy for the spring is $V(q) = \frac{1}{2}k(x - x_0)^2$ where x is the distance between p_1 and p_2. The derivatives are found manually:

$$\frac{\partial V}{\partial q_i}(q) = k\,(x - x_0)\,\frac{\partial x}{\partial q_i}$$
$$\frac{\partial^2 V}{\partial q_i \partial q_j}(q) = k\frac{\partial x}{\partial q_j}\frac{\partial x}{\partial q_i} + k\,(x - x_0)\,\frac{\partial^2 x}{\partial q_i \partial q_j}$$

The spring length x is $x(q) = \sqrt{\vec{v}^T\vec{v}}$ where $\vec{v} = p_1 - p_2$. The derivatives are found:

$$\frac{\partial x}{\partial q_i}(q) = x^{-1}\vec{v}^T\frac{\partial \vec{v}}{\partial q_i}$$
$$\frac{\partial^2 x}{\partial q_i \partial q_j}(q) = -\tfrac{1}{2}x^{-2}\frac{\partial x}{\partial q_j}\vec{v}^T\frac{\partial \vec{v}}{\partial q_i} + x^{-1}\frac{\partial \vec{v}}{\partial q_j}^T\frac{\partial \vec{v}}{\partial q_i} + x^{-1}\vec{v}^T\frac{\partial^2 \vec{v}}{\partial q_i \partial q_j}$$

Finally, the spring vector $\vec{v}$ and its derivatives are found:

$$\vec{v}(q) = p_1 - p_2 \tag{3.16}$$
$$\frac{\partial \vec{v}}{\partial q_i}(q) = \frac{\partial p_1}{\partial q_i} - \frac{\partial p_2}{\partial q_i} \tag{3.17}$$
$$\frac{\partial^2 \vec{v}}{\partial q_i \partial q_j}(q) = \frac{\partial^2 p_1}{\partial q_i \partial q_j} - \frac{\partial^2 p_2}{\partial q_i \partial q_j} \tag{3.18}$$

Eq. (3.16) through (3.18) are evaluated with (2.1), (2.2), and (2.3) from the tree representation. This approach is easily adapted to model non-linear and torsional springs.

3.1.3. Forcing

Robotics applications almost certainly require the ability to include external forcing. Forcing is used to include dissipation (e.g. friction), control inputs (e.g. motor torque), and other effects. The Lagrange d'Alembert principle introduces external forcing to the continuous Euler-Lagrange equation [**28**]. A forcing term is added to the action integral:

$$\delta \int_{t_0}^{t_f} L(q(\tau), \dot{q}(\tau))\ \mathrm{d}\tau + \int_{t_0}^{t_f} f_c(q(\tau), \dot{q}(\tau), u(\tau), \tau) \cdot \delta q\ \mathrm{d}\tau = 0$$

where $f_c(q, \dot{q}, i, t)$ is the total external forcing expressed in the system's generalized coordinates and u are force inputs (e.g, a voltage applied to a motor). This leads to the forced Euler-Lagrange equation:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}}(q, \dot{q}) - \frac{\partial L}{\partial q}(q, \dot{q}) = f_c(q, \dot{q}, u, t). \tag{3.19}$$

Expanding the time derivative and solving for $\ddot{q}$, we find:

$$\ddot{q} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[f_c(q, \dot{q}, u, t) - \frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q} + \frac{\partial L}{\partial \dot{q}}\right] \tag{3.20}$$

The only modification needed to include forcing is the ability to calculate the forcing function $f_c(q, \dot{q}, u, t)$. The following section uses a body wrench as an example of how a forcing function is written in terms of values provided by the tree representation.

3.1.3.1. Body Wrenches. We can include a non-generalized force in an Euler-Lagrange simulation by transforming its wrench representation, F, into generalized coordinates:

$$f_c = \left[J_{s,i}^b\right]^T F$$

where $J_{s,i}^b$ is the body Jacobian [**30**] of the frame to which the force is applied.

The Jacobian can be calculated from values provided by the tree representation (specifically, (2.1) and (2.2)):

$$J^b_{s,i} = \begin{bmatrix} \left(g^{-1}_{s,i}\frac{\partial g_{s,i}}{\partial q_0}\right)^\vee & \left(g^{-1}_{s,i}\frac{\partial g_{s,i}}{\partial q_1}\right)^\vee & \dots & \left(g^{-1}_{s,i}\frac{\partial g_{s,i}}{\partial q_n}\right)^\vee \end{bmatrix}$$

A similar approach is used to include forces applied to a frame but specified in spatial coordinates by using the spatial Jacobian.

3.1.4. Constraints

The tree representation does not change how constraints are included, but can help in calculating the necessary values. Similarly, the constraints do not change the tree representation at all. The constraints depend on the values provided by the tree, but the tree does not depend on the constraints.

In discrete mechanics, we typically deal with holonomic constraints only and therefore do not discuss non-holonomic constraints here. However, [**6**] shows how to do this, and that technique could be implemented using the method presented in this work.

A holonomic constraint restricts the system to a sub-manifold of the configuration. Holonomic constraints are defined as $h(q) = 0 \in \mathbb{R}$ for valid configurations. A system may be subject to many holonomic constraints at once. These are grouped together as a vector of the individual constraints:

$$h(q) = \begin{bmatrix} h_1(q) & \dots & h_m(q) \end{bmatrix}^T \in \mathbb{R}^m$$

The constrained Euler-Lagrange equations are derived by minimizing the action $S(q(\cdot))$ subject to the constraint $h(q(\tau)) = 0 \; \forall \; \tau \in [t_0, t_f]$. The derivation [**28**] yields the constrained Euler-Lagrange equations:

$$\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}} \ddot{q} + \frac{\partial^2 L}{\partial q \partial \dot{q}} \dot{q} - \frac{\partial L}{\partial q} = \frac{\partial h}{\partial q}^T (q) \lambda \tag{3.21a}$$

$$\frac{\partial^2 h}{\partial q \partial q} (q) \circ (\dot{q}, \dot{q}) + \frac{\partial h}{\partial q} (q) \ddot{q} = 0 \tag{3.21b}$$

where the Lagrange multipliers λ are related to the constraint forces and must be solved during integration.[1] Note that the original constraint $h(q)$ doesn't directly appear in (3.21). Instead, the holonomic constraints are enforced in (3.21b) as constraints on the acceleration. This often leads to errors that slowly creep into the simulation during numeric integration. As the error grows, the constraints are increasingly violated.

Special techniques, such as projecting the system into the constraint sub-manifold or introducing damped-spring restoring forces, are used to fix this but these tend to add or remove energy from the system and introduce simulation parameters that have to be adjusted for individual scenarios. Bad choices for these parameters can introduce unstable dynamics.

[1]The notation $X \circ (Y, Z)$ is the bilinear operator X applied to arguments Y and Z

3.1.4.1. Distance between Points. Suppose we have two points $p_1(q), p_2(q) \in \mathbb{R}^3$ to be a fixed distance $L \in \mathbb{R}$ apart. The constraint $h(q)$ is

$$\begin{aligned} h(q) &= 0 \\ &= ||p_1 - p_2||^2 - L^2 \\ &= (p_1 - p_2)^T (p_1 - p_2) - L^2 \end{aligned} \tag{3.22}$$

The above is evaluated using (2.1) from a tree representation. The derivative is found manually

$$\begin{aligned} \frac{\partial h}{\partial q_i}(q) &= \frac{\partial}{\partial q_i}\left((p_1 - p_2)^T (p_1 - p_2) - L^2\right) \\ &= 2\,(p_1 - p_2)^T \left(\frac{\partial p_1}{\partial q_i} - \frac{\partial p_2}{\partial q_i}\right) \end{aligned} \tag{3.23}$$

and is evaluated using (2.1) and (2.2).

Again, by implementing (3.22) and (3.23) and providing a way to define the constraint for two coordinate frames in the tree, the constraint is available to arbitrary systems.

3.1.4.2. Point on Plane. A natural approach to forcing two points to be coincident is to use the distance constraint with $L = 0$. This approach, unfortunately, introduces a singularity to the dynamics because the derivative (3.23) is zero so no constraint force can be applied[2].

Instead, a new constraint is defined to keep a point coincident with a plane. The same point can be constrained against two non-parallel planes to have the point follow a line

[2]In other words, the direction of the force is always undefined when the constraint is satisfied.

along the intersection of the planes, or three planes to keep the point coincident with the point where the planes intersect.

We define the plane by a coordinate frame g_1 and a vector $\vec{n}$, defined relative to g_1, such that the plane intersects the origin of g_1 and is normal to $\vec{n}$. Letting p_2 be the point to constrain, the constraint equation is:

$$h(q) = g_1 \cdot \hat{n} \cdot (p_1 - p_2)$$

where p_1 is the origin of g_1. The values of g_1, p_1, and p_2 are calculated from (2.1).

The derivative:

$$\frac{\partial h}{\partial q_i}(q) = \frac{\partial g_1}{\partial q_i} \cdot \hat{n} \cdot (p_1 - p_2) + g_1 \cdot \hat{n} \cdot \left(\frac{\partial p_1}{\partial q_i} - \frac{\partial p_2}{\partial q_i} \right)$$

is evaluated using (2.1) and (2.2).

3.1.4.3. Screw Constraint. One limitation of the proposed six primitive transformations is they cannot naturally represent screw joints. However, we can build a screw joint by using constraints.

A screw motion is a rotation about an axis by an angle θ followed by a translation along the same axis by a distance d. The ratio of the distance to the angle is called the pitch $p = d/\theta$ [**30**].

We define the constraint equation:

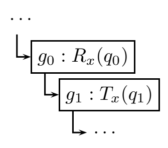
$$h(q) = pq_\theta - q_d \tag{3.24}$$

where q_θ is a configuration variable that parameterizes a rotation and q_d is a configuration variable that parameterizes a translation.

The derivative is:

$$\frac{\partial h}{\partial q_i}(q) = \begin{cases} p & i = \theta \\ -1 & i = d \\ 0 & d \neq i \neq \theta \end{cases} \tag{3.25}$$

To set up a screw joint, we create a frame with a rotation about an axis and add a child that translates along the same axis:



A screw constraint is created using q_0 for q_θ and q_1 for q_d. The resulting system will model a screw joint.

3.2. State Space Form

In practice the Euler-Lagrange dynamics are represented as a first-order state space representation $\dot{x} = f(x, u, t)$ for numeric integration, analysis, and optimal control. The particular choice for this representation determines the form the linearized dynamics.

For the following sections, we use the state space $x = \begin{bmatrix} q & \dot{q} \end{bmatrix}$. The corresponding dynamics for a forced, unconstrained system are:

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = f(x, u, t) = \begin{bmatrix} \dot{q} \\ \left(\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right)^{-1} \left(f_c(q, \dot{q}, u, t) - \frac{\partial^2 L}{\partial q \partial \dot{q}} \dot{q} + \frac{\partial L}{\partial \dot{q}}\right) \end{bmatrix}. \tag{3.26}$$

The first- and second-order linearizations for this system are developed in Sec. 3.3 and Sec. 3.4. Linearizations for constrained systems are discussed in Sec. 3.5. Constraints change the resulting equation, but the state space form remains the same.

3.3. First Derivatives of the Dynamics

In Sec. 3.1, we remarked that the standard form of the Euler-Lagrange equations (3.4) is often desired for analysis of the system. While we lose some analysis tools by avoiding the symbolic expression[3], the numeric approach extends to find linearizations of arbitrary rigid body systems.

Calculating the dynamics of the system allows us to simulate trajectories; linearizations go considerably further by enabling many analysis and optimization techniques. For example, finding the controllability matrix from the linearization at a particular configuration tells us if the system can be locally stabilized with a linear feedback law. Furthermore, we can use LQR methods to calculate a stabilizing feedback law for that configuration automatically.

The computational complexity of calculating linearizations for complex systems is often a severely limiting factor in symbolic approaches [**39**]. The explosive growth of the equations continues to get worse as further derivatives of the Lagrangian are found. The numeric approach still provides the exact derivatives for the linearization and continuous dynamics to be scalable to large systems.

This section derives the linearization for an unconstrained equation to emphasize the approach. Linearizations for constrained systems are considered in Sec. 3.5.

[3]In practice, this is not accurate. The symbolic equations for complex systems like the hand are so large that analysis is impractical or impossible.

The linearization of the continuous state space model is $\delta\dot{x} = \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial u}\delta u$. Expanding the state form defined in (3.26):

$$\begin{bmatrix} \delta\dot{q} \\ \delta\ddot{q} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial\dot{q}}{\partial q} & \dfrac{\partial\dot{q}}{\partial\dot{q}} \\ \dfrac{\partial\ddot{q}}{\partial q} & \dfrac{\partial\ddot{q}}{\partial\dot{q}} \end{bmatrix} \begin{bmatrix} \delta q \\ \delta\dot{q} \end{bmatrix} + \begin{bmatrix} \dfrac{\partial\dot{q}}{\partial u} & \dfrac{\partial\ddot{q}}{\partial u} \end{bmatrix} \delta u = \begin{bmatrix} 0 & \mathcal{I} \\ \dfrac{\partial\ddot{q}}{\partial q} & \dfrac{\partial\ddot{q}}{\partial\dot{q}} \end{bmatrix} \begin{bmatrix} \delta q \\ \delta\dot{q} \end{bmatrix} + \begin{bmatrix} 0 & \dfrac{\partial\ddot{q}}{\partial u} \end{bmatrix} \delta u \qquad (3.27)$$

The terms $\dfrac{\partial\ddot{q}}{\partial q}$ and $\dfrac{\partial\ddot{q}}{\partial\dot{q}}$ are found by differentiating (3.20):

$$\frac{\partial\ddot{q}}{\partial q} = \left(\frac{\partial^2 L}{\partial\dot{q}\partial\dot{q}}\right)^{-1} \left(\frac{\partial f_c}{\partial q} + \frac{\partial L}{\partial q} - \frac{\partial^3 L}{\partial q\partial\dot{q}\partial\dot{q}}\ddot{q} - \frac{\partial^3 L}{\partial q\partial q\partial\dot{q}}\dot{q}\right)$$
$$\frac{\partial\ddot{q}}{\partial\dot{q}} = \left(\frac{\partial^2 L}{\partial\dot{q}\partial\dot{q}}\right)^{-1} \left(\frac{\partial f_c}{\partial\dot{q}} + \frac{\partial L}{\partial\dot{q}} - \frac{\partial^3 L}{\partial\dot{q}\partial q\partial\dot{q}}\dot{q} - \frac{\partial^2 L}{\partial q\partial\dot{q}}\right)$$

where we have taken advantage of the fact that $\dfrac{\partial^3 L}{\partial\dot{q}\partial\dot{q}\partial\dot{q}} = 0$ for Lagrangians of the form (3.7). The derivatives of the input are also found from (3.20):

$$\frac{\partial\ddot{q}}{\partial u} = \left(\frac{\partial^2 L}{\partial\dot{q}\partial\dot{q}}\right)^{-1} \frac{\partial f_c}{\partial u}$$

These three equations are calculated numerically from the Lagrangian derivatives and used in (3.27) to calculate the linearization at a particular state.

The linearization requires third-order derivatives of the Lagrangian. These are found by continuing the chain-rule approach used to find the first and second derivatives. Equations for these derivatives can be found in Sec. A.1.1 of the appendix.

3.4. Second Derivatives of the Dynamics

The approach in Sec. 3.3 extends to the second-order linearization of the dynamics as well. This section derives the equations for the unconstrained, forced equation. Constrained linearizations are discussed in Sec 3.5. The expanded second-order linearization is:

$$\delta^2 \dot{x} = \begin{bmatrix} \delta q \\ \delta \dot{q} \\ \delta u \end{bmatrix}^T \begin{bmatrix} \frac{\partial^2 f}{\partial q \partial q} & \frac{\partial^2 f}{\partial q \partial \dot{q}} & \frac{\partial^2 f}{\partial q \partial u} \\ \frac{\partial^2 f^T}{\partial q \partial \dot{q}} & \frac{\partial^2 f}{\partial \dot{q} \partial \dot{q}} & \frac{\partial^2 f}{\partial \dot{q} \partial u} \\ \frac{\partial^2 f^T}{\partial q \partial u} & \frac{\partial^2 f^T}{\partial \dot{q} \partial u} & \frac{\partial^2 f}{\partial u \partial u} \end{bmatrix} \begin{bmatrix} \delta q \\ \delta \dot{q} \\ \delta u \end{bmatrix}$$

where symmetry was used to reduced the number of unique terms to six. Each derivative of f has two components corresponding to derivatives of $\dot{q}$ and $\ddot{q}$. The second derivatives of $\dot{q}$ are zero, so six equations are required to calculate (3.4).

Second derivatives are found by differentiating the expanded Euler-Lagrange equation. For $\frac{\partial^2 f}{\partial q_i \partial q_j}$, we have:

$$\frac{\partial^2}{\partial q_i \partial q_j} \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}} \ddot{q} + \frac{\partial^2 L}{\partial q \partial \dot{q}} \dot{q} - \frac{\partial L}{\partial q} = f_c \right]$$

$$\frac{\partial}{\partial q_i} \left[\frac{\partial^3 L}{\partial q_j \partial \dot{q} \partial \dot{q}} \ddot{q} + \frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}} \frac{\partial \ddot{q}}{\partial q_j} + \frac{\partial^3 L}{\partial q_j \partial q \partial \dot{q}} \dot{q} - \frac{\partial^2 L}{\partial q_j \partial q} = \frac{\partial f_c}{\partial q_j} \right]$$

$$\frac{\partial^4 L}{\partial q_i \partial q_j \partial \dot{q} \partial \dot{q}} \ddot{q} + \frac{\partial^3 L}{\partial q_j \partial \dot{q} \partial \dot{q}} \frac{\partial \ddot{q}}{\partial q_i} + \frac{\partial^3 L}{\partial q_i \partial \dot{q} \partial \dot{q}} \frac{\partial \ddot{q}}{\partial q_j} + \frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}} \frac{\partial^2 \ddot{q}}{\partial q_i \partial q_j} + \frac{\partial^4 L}{\partial q_i \partial q_j \partial q \partial \dot{q}} \dot{q} - \frac{\partial^3 L}{\partial q_i \partial q_j \partial q} = \frac{\partial^2 f_c}{\partial q_i \partial q_j}$$

$$\frac{\partial^2 \ddot{q}}{\partial q_i \partial q_j} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[\frac{\partial^2 f_c}{\partial q_i \partial q_j} + \frac{\partial^3 L}{\partial q_i \partial q_j \partial q} - \frac{\partial^4 L}{\partial q_i \partial q_j \partial q \partial \dot{q}} \dot{q} \right.$$
$$\left. - \frac{\partial^3 L}{\partial q_i \partial \dot{q} \partial \dot{q}} \frac{\partial \ddot{q}}{\partial q_j} - \frac{\partial^3 L}{\partial q_j \partial \dot{q} \partial \dot{q}} \frac{\partial \ddot{q}}{\partial q_i} - \frac{\partial^4 L}{\partial q_i \partial q_j \partial \dot{q} \partial \dot{q}} \ddot{q} \right].$$

The other two second derivatives with respect to a configuration variable are:

$$\frac{\partial^2 \ddot{q}}{\partial q_i \partial \dot{q}_j} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[\frac{\partial^2 f_c}{\partial q_i \partial \dot{q}_j} + \frac{\partial^3 L}{\partial q_i \partial \dot{q}_j \partial q} - \frac{\partial^3 L}{\partial q_i \partial q_j \partial \dot{q}} \right.$$
$$\left. - \frac{\partial^4 L}{\partial q_i \partial \dot{q}_j \partial q \partial \dot{q}} \dot{q} - \frac{\partial^3 L}{\partial q_i \partial \dot{q} \partial \dot{q}} \frac{\partial \ddot{q}}{\partial \dot{q}_j} - \frac{\partial^3 L}{\partial \dot{q}_j \partial \dot{q} \partial \dot{q}} \frac{\partial \ddot{q}}{\partial q_i} - \frac{\partial^4 L}{\partial q_i \partial \dot{q}_j \partial \dot{q} \partial \dot{q}} \ddot{q} \right]$$

$$\frac{\partial^2 \ddot{q}}{\partial q_i \partial u_j} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[\frac{\partial^2 f_c}{\partial q_i \partial u_j} - \frac{\partial^3 L}{\partial q_i \partial \dot{q} \partial \dot{q}} \frac{\partial \ddot{q}}{\partial u_j}\right].$$

Derivatives of $\ddot{q}$ with respect to configuration velocities $\dot{q}_i$ are:

$$\frac{\partial^2 \ddot{q}}{\partial \dot{q}_i \partial \dot{q}_j} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[\frac{\partial^2 f_c}{\partial \dot{q}_i \partial \dot{q}_j} + \frac{\partial^3 L}{\partial \dot{q}_i \partial \dot{q}_j \partial q} - \frac{\partial^3 L}{\partial \dot{q}_i \partial q_j \partial \dot{q}}\right]$$
$$\frac{\partial^2 \ddot{q}}{\partial \dot{q}_i \partial u_j} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \frac{\partial^2 f_c}{\partial \dot{q}_i \partial u_j}.$$

Finally, the derivative with respect to two force inputs is:

$$\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \frac{\partial^2 f_c}{\partial u_i \partial u_j}.$$

The second derivative equations are calculated numerically using the results from the first derivative in Sec. 3.3 and derivatives of the Lagrangian that are calculated from the tree representation. Derivatives of the Lagrangian are reviewed in Sec. A.1.

3.5. Constrained Linearizations

Section 3.1.4 reviewed the constrained Euler-Lagrange equation. This section discusses extensions to the linearizations Sec. 3.3 and 3.4 to include constraints as well.

The approach for deriving and calculating the constrained equations is the same. The difference is that derivatives of the constraint force λ are also required. The constraints do not change the state form (3.26) so the same derivatives of $\ddot{q}$ are required. The process is demonstrated for the derivative $\frac{\partial \ddot{q}}{\partial q}$.

We start by differentiating (3.21a):

$$\frac{\partial}{\partial q_i}\left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\ddot{q} + \frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q} - \frac{\partial L}{\partial q} = \frac{\partial h}{\partial q}^T \lambda\right]$$

$$\frac{\partial \ddot{q}}{\partial q_i} = \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1}\left[\frac{\partial^2 h}{\partial q_i \partial q}^T \lambda + \frac{\partial h}{\partial q}^T \frac{\partial \lambda}{\partial q_i} + \frac{\partial^2 L}{\partial q_i \partial q} - \frac{\partial^3 L}{\partial q_i \partial q \partial \dot{q}}\dot{q} - \frac{\partial^3 L}{\partial q_i \partial \dot{q} \partial \dot{q}}\ddot{q}\right]. \tag{3.28}$$

Evaluating (3.28) requires $\frac{\partial \lambda}{\partial q_i}$. We find this derivative by first differentiating (3.21b) and substituting in (3.28):

$$\frac{\partial}{\partial q_i}\left[\frac{\partial^2 h}{\partial q \partial q} \circ (\dot{q}, \dot{q}) + \frac{\partial h}{\partial q}\ddot{q} = 0\right]$$

$$\frac{\partial^3 h}{\partial q_i \partial q \partial q} \circ (\dot{q}, \dot{q}) + \frac{\partial^2 h}{\partial q_i \partial q}\ddot{q} + \frac{\partial h}{\partial q}\frac{\partial \ddot{q}}{\partial q_i} = 0$$

$$\frac{\partial^3 h}{\partial q_i \partial q \partial q} \circ (\dot{q}, \dot{q}) + \frac{\partial^2 h}{\partial q_i \partial q}\ddot{q} + \frac{\partial h}{\partial q}\left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1}\left[\frac{\partial^2 h}{\partial q_i \partial q}^T \lambda \right.$$
$$\left. + \frac{\partial h}{\partial q}^T \frac{\partial \lambda}{\partial q_i} + \frac{\partial^2 L}{\partial q_i \partial q} - \frac{\partial^3 L}{\partial q_i \partial q \partial \dot{q}}\dot{q} - \frac{\partial^3 L}{\partial q_i \partial \dot{q} \partial \dot{q}}\ddot{q}\right] = 0.$$

We now solve for the desired derivative, $\dfrac{\partial \lambda}{\partial q_i}$:

$$\frac{\partial \lambda}{\partial q_i} = -\left[\frac{\partial h}{\partial q}\left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1}\frac{\partial h}{\partial q}^T\right]^{-1}\left[\frac{\partial^3 h}{\partial q_i \partial q \partial q}\circ(\dot{q},\dot{q}) + \frac{\partial^2 h}{\partial q_i \partial q}\ddot{q}\right.$$
$$\left. + \frac{\partial h}{\partial q}\left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1}\left(\frac{\partial^2 h}{\partial q_i \partial q}^T\lambda + + \frac{\partial^2 L}{\partial q_i \partial q} - \frac{\partial^3 L}{\partial q_i \partial q \partial \dot{q}}\dot{q} - \frac{\partial^3 L}{\partial q_i \partial \dot{q}\partial \dot{q}}\ddot{q}\right)\right] = 0. \quad (3.29)$$

Equation (3.29) is evaluated numerically using the computed Lagrangian derivatives and the constraint functions provided for each constraint. The result is used in (3.28) to calculate the derivative of the dynamics.

The same approach is used for the remaining first derivatives and the second derivatives. The following section introduces another extension to the continuous Euler-Lagrange equation for kinematic configuration variables. The full derivative equations including constraints, forcing, and the kinematic variables are presented in Sec. A.1.

3.6. Kinematic Configuration Variables

Mechanical systems are often modeled in two ways. In a dynamic model, all of the forces and inertial bodies of the system are included and used to determine the system's motion within a static environment. A subset of these can be reduced to kinematic models in which the actuators are assumed to be strong enough to track any desired configuration path or that the system intrinsically follows kinematic paths [**10**].

Neither of these approaches are appropriate for a marionette. A dynamic model must include the both the marionette and the driving mechanisms (so long as the end of the string actuating the marionette is not inertially fixed). The model is therefore tightly coupled to the particular implementation it was derived from. However, there are certainly characteristics of a marionette's dynamics that are independent of the method used to pull its strings. A kinematic model that captures the full range of motion cannot be derived for an under-actuated system such as the marionette.

A better approach is to consider the marionette as two distinct systems. The marionette is considered an under-actuated, dynamic system. The string-driving mechanism is a fully actuated system that can be reduced to a kinematic model with velocity inputs[4]. The systems are coupled by the strings, which are introduced to the model as non-holonomic constraints.

Such an arrangement allows us to the study the intrinsic dynamics of the system rather than new dynamics introduced by actuator mechanisms. If the dynamics of a particular implementation wish to be studied, the kinematic reduction can easily be replaced with a full dynamic model.

The modified equations of motion for a mixed kinematic-dynamic model are derived in the following section. A mixed kinematic-dynamic model for the marionette that is based on this approach is discussed in detail in [**17**]. This formulation is not restricted to marionette. It applies to arbitrary mechanical systems. The kinematic variables may be "internal" to the system, affecting the Lagrangian, or purely "external," only affecting the system through constraint forces.

[4]In order to guarantee that the kinematic trajectory is C^2, the kinematic acceleration is used for the input. This is required for constrained systems because both holonomic and non-holonomic constraints are reduced to constraints on the acceleration of the generalized coordinates in continuous time.

3.6.1. Mixed Dynamic-Kinematic Modeling

We begin by organizing the configuration variables into a dynamic part, q, and a kinematic part, ρ. The kinematic configuration variables ρ may be parameters from a tree description or new variables that were created to parameterize a specific constraint. For example, the marionette model in [**17**] uses kinematic configuration variables to control the length of the spring. Configuration variables in dynamic models must uniquely parameterize the Lagrangian to avoid singular dynamics[5]. Kinematic configuration variables do not have this restriction because their trajectories are by definition explicitly specified.

The constraint matrix and generalized forces are split into the parts affecting the dynamic and kinematic coordinates: $A(t) = \begin{bmatrix} A_d & A_k \end{bmatrix}$ and $f(\cdot) = \begin{bmatrix} f_d \\ f_k \end{bmatrix}$. Substituting these and the new state $q = \begin{bmatrix} q_d \\ q_k \end{bmatrix}$ into the constrained and forced Euler-Lagrange equation, we have:

$$\begin{bmatrix} \frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}} & \frac{\partial^2 L}{\partial \dot{\rho} \partial \dot{q}} \\ \frac{\partial^2 L}{\partial \dot{q} \partial \dot{\rho}} & \frac{\partial^2 L}{\partial \dot{\rho} \partial \dot{\rho}} \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \ddot{\rho} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 L}{\partial q \partial \dot{q}} & \frac{\partial^2 L}{\partial \rho \partial \dot{q}} \\ \frac{\partial^2 L}{\partial q \partial \dot{\rho}} & \frac{\partial^2 L}{\partial \rho \partial \dot{\rho}} \end{bmatrix} \begin{bmatrix} \dot{q} \\ \dot{\rho} \end{bmatrix} - \begin{bmatrix} \frac{\partial L}{\partial q} \\ \frac{\partial L}{\partial \rho} \end{bmatrix} = \begin{bmatrix} f_d \\ f_k \end{bmatrix} + \begin{bmatrix} A_d^T \lambda \\ A_k^T \lambda \end{bmatrix}$$

[5]In other words, every configuration variable must have some mass associated with it and the inertia matrix $M(q) = \frac{\partial^2 L}{\partial q \partial q}$ must be non-singular.

Expanding the matrix multiplications leads to two equations representing the dynamics of q and ρ[6]:

$$\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}} \ddot{q} + \frac{\partial^2 L}{\partial \dot{\rho} \partial \dot{q}} \ddot{\rho} + \frac{\partial^2 L}{\partial q \partial \dot{q}} \dot{q} + \frac{\partial^2 L}{\partial \rho \partial \dot{q}} \dot{\rho} - \frac{\partial L}{\partial q} = f_d + A_d^T \lambda \tag{3.30a}$$

$$\frac{\partial^2 L}{\partial \dot{q} \partial \dot{\rho}} \ddot{q} + \frac{\partial^2 L}{\partial \dot{\rho} \partial \dot{\rho}} \ddot{\rho} + \frac{\partial^2 L}{\partial q \partial \dot{\rho}} \dot{q} + \frac{\partial^2 L}{\partial \rho \partial \dot{\rho}} \dot{\rho} - \frac{\partial L}{\partial \rho} = f_k + A_k^T \lambda \tag{3.30b}$$

Equation (3.30b) can be satisfied for any values of ρ, $\dot{\rho}$, $\ddot{\rho}$, q, and $\dot{q}$ by the appropriate choice of f_k (provided the constraints are satisfied). This is the kinematic assumption. We are not concerned with calculating the kinematic force inputs, and so we drop (3.30b) and let $\ddot{\rho}$ become an input to the model. Solving (3.30a) for $\ddot{q}$, we find:

$$\ddot{q} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[A_d^T \lambda + f_c + \frac{\partial L}{\partial q} - \frac{\partial^2 L}{\partial \dot{\rho} \partial \dot{q}} \ddot{\rho} - \frac{\partial^2 L}{\partial q \partial \dot{q}} \dot{q} - \frac{\partial^2 L}{\partial \rho \partial \dot{q}} \dot{\rho}\right].$$

This is the mixed dynamic-kinematic model of the system's equations of motion. The full model, including the modified equations for λ and the extended linearizations are discussed in Sec. A.1 of the appendix.

[6]The coordinates ρ are still being treated as dynamic.

CHAPTER 4

Discrete Dynamics

An alternative to deriving the continuous equations of motion is to introduce the discrete approximation earlier in the derivation. This leads to integration methods that respect the fundamental symmetries in dynamics.[1] This approach is called discrete mechanics and the resulting algorithms are called Variational Integrators [**23**]. Variational integrators conserve (or nearly conserve, depending on the particular integrator) fundamental quantities like momentum and energy [**24**]. They are also well suited for problems involving holonomic constraints, impacts, and non-smooth phenomenon [**13**].

This section builds on the approaches introduced in Ch. 2 and Ch. 3 to develop variational integrators for arbitrary mechanical systems in generalized coordinates including those with external forcing, holonomic constraints, and kinematic configuration variables.

Section. 4.1 introduces variational integrators for simulating mechanical systems. The algorithm for implementing a variational integrator based on a tree representation is described in Sec. 4.2. Variational integrators describe the system dynamics as an implicit difference equation. An abstraction to treat variational integrators as explicit first-order discrete systems in a state space is introduced in Sec. 4.3. The state space representation allows the first- and second-order linearizations to be calculated as discussed in Sec. 4.4 and Sec. 4.5, respectively. Section 4.6 discusses linearizations of constrained variational

[1]There are also specially designed numeric integrators for continuous dynamics that preserve the same properties (e.g. the Newmark scheme). It has been shown that these special integrators can often be derived from a variational integrator by choosing a particular discrete Lagrangian [**41**].

integrators. Finally, kinematic configuration variables are added to the variational integrator in Sec. 4.7.

4.1. Discrete Dynamics

In discrete mechanics, we find a sequence $\{(t_0, q_0), (t_1, q_1), \ldots, (t_n, q_n)\}$ that approximates the continuous trajectory of a mechanical system ($q_k \approx q(t_k)$). This chapter assumes a constant time step ($t_{k+1} - t_k = \Delta t \; \forall \; k$) for simplicity, but in general the time step can be varied to use adaptive time stepping algorithms. For example, [**29**] describes a method that adapts the time step to maintain perfect energy conservation.

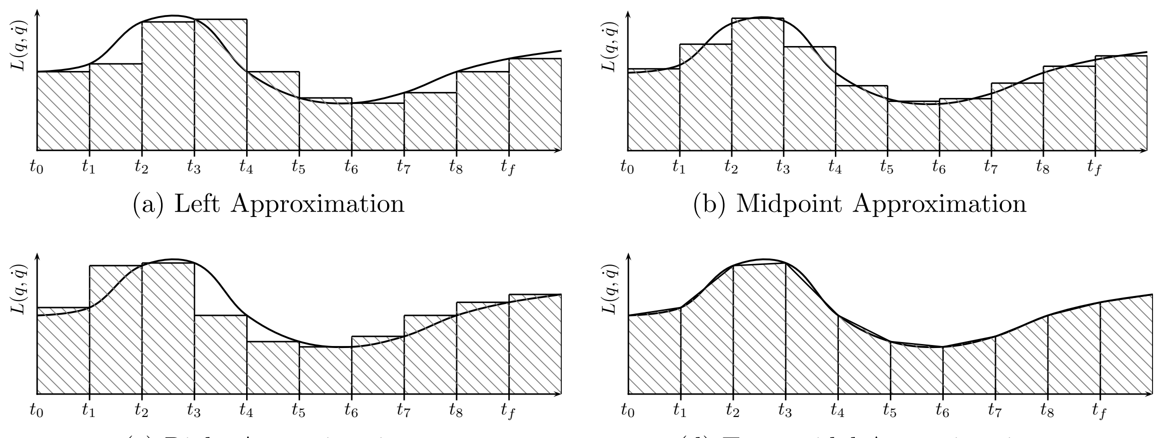
To derive a variational integrator, we define a discrete Lagrangian that approximates the action integral over a short interval:

$$L_d\left(q_k, q_{k+1}\right) \approx \int_{t_k}^{t_{k+1}} L(q(\tau), \dot{q}(\tau)) \mathrm{d}\tau.$$

Figure 4.1 shows several choices of approximations used to define a discrete Lagrangian. The order of accuracy for the approximation is directly related to the order of accuracy for the resulting trajectory [**29**]. An example of a discrete Langrian is discussed in Sec. 4.2.

The discrete Lagrangian replaces the system's action integral with an action sum:

$$S(q([t_0, t_f])) = \int_{t_0}^{t_f} L(q(\tau), \dot{q}(\tau)) \mathrm{d}\tau \approx \sum_{k=0}^{n-1} L_d\left(q_k, q_{k+1}\right). \tag{4.1}$$



(a) Left Approximation (b) Midpoint Approximation

(c) Right Approximation (d) Trapezoidal Approximation

Figure 4.1. The discrete Lagrangian approximates segments of the continuous action integral. The area of each shaded region represents a value of the discrete Lagrangian.

The action sum is minimized with a variational principle to get an implicit difference equation–analogous to the Euler-Lagrange equations in (3.3)–called the Discrete Euler-Lagrange (DEL) equation:

$$D_1 L_d\left(q_k, q_{k+1}\right) + D_2 L_d\left(q_{k-1}, q_k\right) = 0. \tag{4.2}$$

where $D_n f(\dots)$ is the derivative of $f(\dots)$ with respect to its n-th argument. The derivation of (4.2) is analogous to the approach used to derive the continuous dynamics equation (3.3).

Whereas the continuous Euler-Lagrange equation is an ordinary differential equation that is numerically integrated, the discrete Euler-Lagrange equation (4.2) presents a root-finding problem to get the next configuration. Given two initial configurations q_0 and q_1, we solve:

$$f(q_{k+1}) = D_1 L_d\left(q_k, q_{k+1}\right) + D_2 L_d\left(q_{k-1}, q_k\right) = 0. \tag{4.3}$$

to determine q_2. We then move forward and iterate to find $q_3, q_4, \ldots, q_N$. The sequence $\{q_0, q_1, \ldots, q_N\}$ is the discrete trajectory.

At each time step, we must solve $f(q_{k+1}) = 0$. The Newton-Raphson root finding algorithm [**35**] performs well for this problem but requires the derivative of $f(q_{k+1})$:

$$Df(q_{k+1}) = D_2 D_1 L_d\left(q_k, q_{k+1}\right) \tag{4.4}$$

Section 4.2 describes how these equations are implemented from the tree representation. First, the following subsections extend the DEL equations to handle forcing and constraints.

4.1.1. Forcing

In discrete mechanics, we approximate the continuous force $f_c(q, \dot{q}, u, t)$ with left and right discrete forces $f_d^-(q_k, q_{k+1}, u_k, t_k, t_{k+1})$ and $f_d^+(q_k, q_{k+1}, u_k, t_k, t_{k+1})$ such that:

$$\begin{aligned} f_d^-\left(q_k, q_{k+1}, u_k, t_k, t_{k+1}\right) \cdot \delta q_k + f_d^+\left(q_k, q_{k+1}, u_k, t_k, t_{k+1}\right) \cdot \delta q_{k+1} \\ \approx \int_{t_k}^{t_{k+1}} f_c\left(q(\tau), \dot{q}(\tau), u(\tau), \tau\right) \cdot \delta q \; \mathrm{d}\tau. \end{aligned} \tag{4.5}$$

The discrete input u_k approximates the continuous input ($u_k \approx u(t_k)$). The forcing approximation is similar to the discrete Lagrangian in Fig. 4.1. A particular approximation is defined and discussed in Sec. 4.2.

In addition to choosing different approximations for (4.5), there are other approaches to discrete forcing that change how the force inputs u_k are introduced[**31**]. In particular, the force might be defined to accept the force inputs at two points: $f_d^\pm\left(q_k, q_{k+1}, u_k, u_{k-1}, t_k, t_{k+1}\right)$.

The discrete analog to the Lagrange d'Alembert principle is:

$$\delta \sum_{k=0}^{N-1} L_d\left(q_k, q_{k+1}\right) + \sum_{k=0}^{N-1} \Big(f_d^-\left(q_k, q_{k+1}, t_k, t_{k+1}\right) \cdot \partial q_k + \\ f_d^+\left(q_k, q_{k+1}, t_k, t_{k+1}\right) \cdot \partial q_{k+1} \Big) = 0 \quad (4.6)$$

Solving (4.6) leads to the forced discrete Euler-Lagrange equation:

$$D_2 L_d\left(q_{k-1}, q_k\right) + f_d^+\left(q_{k-1}, q_k, u_{k-1}, t_{k-1}, t_k\right) + \\ D_1 L_d\left(q_k, q_{k+1}\right) + f_d^-\left(q_k, q_{k+1}, u_k, t_k, t_{k+1}\right) = 0 \quad (4.7)$$

As with the unforced variational integrator (4.3), a system is integrated forward by specifying initial conditions q_0, u_0, q_1, u_1 and numerically solving (4.7) to determine q_2. This integration is continued by specifying the next force input, u_2, and solving for q_3. The process is repeated to simulate the system for as long as desired.

4.1.2. Constraints

The constrained variational integrator is derived by minimizing the discrete action sum (4.1) subject to $h(q_k) = 0 \; \forall \; k = 0 \dots N$. This leads to the constrained DEL equations[**29**]:

$$D_2 L_d(q_{k-1}, q_k) + D_1 L_d(q_k, q_{k+1}) = Dh^T(q_k)\lambda_k$$

$$h(q_{k+1}) = 0$$

We now have $n + m$ non-linear equations to solve in terms of q_{k+1} and λ_k. We define a new equation for the root solver:

$$g\left(\begin{bmatrix} q_{k+1} \\ \lambda_k \end{bmatrix}\right) = \begin{bmatrix} D_2L_d(q_{k-1}, q_k) + D_1L_d(q_k, q_{k+1}) - Dh^T(q_k)\lambda_k \\ h(q_{k+1}) \end{bmatrix}$$

and find the necessary derivative:

$$Dg\left(\begin{bmatrix} q_{k+1} \\ \lambda_k \end{bmatrix}\right) = \begin{bmatrix} D_2D_1L_d(q_k, q_{k+1}) & -Dh^T(q_k) \\ Dh(q_{k+1}) & 0 \end{bmatrix}$$

The discrete integrator enforces $h(q_k) = 0$ directly at every time step. This eliminates the aforementioned error creep, and results in trajectories that always satisfy the constraint.

The discrete Lagrange multipliers λ_k and the term $Dh^T(q_k)$ have the same interpretation as the continuous case. The term $Dh^T(q_k)\lambda_k$ represents the forcing applied by the constraints with λ_k being proportional to the magnitude of each constraint force.

From the above, constraints are included in the simulation by providing $h(q)$ and $Dh(q)$ for each type of constraint. Examples of constraints using the tree structure are discussed in Sec. 3.1.4.

4.2. Creating a Variational Integrator

There are generally two approaches to implement a variational integrator.[2] In the first, one explicitly finds the equations of motion (the discrete Euler-Lagrange equation) manually or with symbolic algebra software. For large systems the complexity essentially

[2]Though we focus on variational integrators, this discussion largely applies to continuous Lagrangian mechanics.

requires a symbolic algebra package such as *Mathematica*, but such tools only make the task possible in a formal sense. Realistically, the equations become too large and complex.

Instead, the system can be described using special coordinate choices that result in special Lagrangian forms [**21**]. The most common examples are treating everything as a point mass ($L(q, \dot{q}) = \dot{q}^T M \dot{q} + V(q)$) [**25**] or treating each body as being free in space and imposing the mechanical structure through constraints [**7**]. Integrators based on these forms have excellent performance because of their simplicity, but lose the benefits and convenience of generalized coordinates.

With the tree description, we achieve comparable performance and still work in generalized coordinates. The integrator works for arbitrary systems, is not dependent on symbolic algebra software, and by taking advantage of caching, performance scales well.

We must choose a discrete Lagrangian to implement (4.3) and (4.4). A common choice is the midpoint approximation:

$$L_d(q_k, q_{k+1}) = L\left(\frac{q_k + q_{k+1}}{2}, \frac{q_{k+1} - q_k}{\Delta t}\right)\Delta t. \tag{4.8}$$

This choice corresponds to Fig. 4.1b and has second order accuracy[**40**].

We find derivatives of (4.8) using the chain rule:

$$D_1 L_d(q_k, q_{k+1}) = \frac{\Delta t}{2}\frac{\partial L}{\partial q} - \frac{\partial L}{\partial \dot{q}} \tag{4.9}$$

$$D_2 L_d(q_{k-1}, q_k) = \frac{\Delta t}{2}\frac{\partial L}{\partial q} + \frac{\partial L}{\partial \dot{q}} \tag{4.10}$$

$$D_2 D_1 L_d(q_k, q_{k+1}) = \frac{1}{4}\frac{\partial^2 L}{\partial q \partial q} + \frac{1}{2}\frac{\partial^2 L}{\partial \dot{q} \partial q} - \frac{1}{2}\frac{\partial^2 L}{\partial q \partial \dot{q}} - \frac{1}{\Delta t}\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}} \tag{4.11}$$

Eq. (4.9), (4.10), and (4.11) allow us to calculate (4.3) and (4.4) in terms of the continuous Lagrangian and its derivatives, which are calculated numerically from the tree representation discussed in Sec. 3.1.1.

A similar approximation is used for the discrete forcing:

$$f_d^{\alpha-}(q_k, q_{k+1}, u_k, t_k, t_{k+1}) = f\left(\frac{q_k + q_{k+1}}{2}, \frac{q_{k+1} - q_k}{\Delta t}, u_k, \frac{t_k + t_{k+1}}{2}\right)\Delta t \tag{4.12a}$$

$$f_d^{\alpha+}(q_k, q_{k+1}, t_k, t_{k+1}) = 0 \tag{4.12b}$$

Equation (4.12) is evaluated numerically from the various continuous force equations as described in Sec. 3.1.3.

The combined forced, constrained integration equations are:

$$g\left(\begin{bmatrix} q_{k+1} \\ \lambda_k \end{bmatrix}\right) = \begin{bmatrix} D_2L_d(q_{k-1}, q_k) + f_d^+(q_{k-1}, q_k, u_{k-1}, t_{k-1}, t_k) + \\ D_1L_d(q_k, q_{k+1}) + f_d^-(q_k, q_{k+1}, u_k, t_k, t_{k+1}) - Dh^T(q_k)\lambda_k \\ h(q_{k+1}) \end{bmatrix}. \tag{4.13}$$

We can improve the performance of (4.13) by acknowledging that several terms are constant with respect to parameters q_{k+1} and λ_k. We define the terms:

$$p_k = D_2L_d(q_{k-1}, q_k) + f_d^+(q_{k-1}, q_k, u_{k-1}, t_{k-1}, t_k)$$

$$\pi_k = Dh(q_k).$$

In the absence of forcing, p_k is the conserved momentum of the integrator[**40**]. In general, however, it is defined for computation convenience. The integrator equation becomes:

$$g\left(\begin{bmatrix} q_{k+1} \\ \lambda_k \end{bmatrix}\right) = \begin{bmatrix} p_k + D_1 L_d(q_k, q_{k+1}) + f_d^-(q_k, q_{k+1}, t_k, t_{k+1}) - \pi_k^T \lambda_k \\ h\left(q_{k+1}\right) \end{bmatrix} \tag{4.14}$$

The derivative is:

$$Dg\left(\begin{bmatrix} q_{k+1} \\ \lambda_k \end{bmatrix}\right) = \begin{bmatrix} D_2 D_1 L_d(q_k, q_{k+1}) + D_2 f_d^-(q_k, q_{k+1}, t_k, t_{k+1}) & -\pi_k^T \\ \pi_{k+1}\left(= Dh(q_{k+1})\right) & 0 \end{bmatrix} \tag{4.15}$$

This avoids calculating $D_2 L_d(\cdot)$ during each root solver iteration. Note that introducing the p_k term removes the explicit dependence on q_{k-1} and the integrator becomes a one step mapping $(q_k, p_k) \rightarrow (q_{k+1}, p_{k+1})$.

We now need an initialization procedure for the integrator:

```
**Require:** $q_0$, $q_1$
    Set tree: $q = (1-\alpha)q_0 + \alpha q_1$, $\dot{q} = \frac{q_1 - q_0}{\Delta t}$
    $p_1 = D_2 L_d(q_0, q_1) + f_d^+(q_0, q_1)$
    Set tree: $q = q_1$, $\dot{q} = \frac{q_1 - q_0}{\Delta t}$
    $\pi_1 = Dh(q_1)$
    $\lambda_0 = 0$
    **return** $q_1, p_1, \pi_1, \lambda_0$
```

4.3. State Space Form

To use variational integrators with existing discrete dynamic theory, we must represent the dynamics as an explicit first-order dynamic system in state space: $x_{k+1} = f(x_k, u_k, t_k)$. It is useful to introduce abbreviated notation for the discrete Lagrangian and discrete force: $L_k = L_d(q_{k-1}, q_k, t_{k-1}, t_k)$ and $f_k^{\pm} = f_d(q_{k-1}, q_k, u_{k-1}, t_{k-1}, t_k)$. When used with slot derivative notation, the abbreviated forms are considered to have the same arguments as the unabbreviated ones.

In continuous time the configuration and velocity of an Euler-Lagrange system are often concatenated into a single state $x = \begin{bmatrix} q & \dot{q} \end{bmatrix}$ to create a first-order representation of the system. For the variational integrator in discrete time, a convenient choice[**14**] for the state is:

$$x_{k+1} = \begin{bmatrix} q_{k+1} \\ p_{k+1} \end{bmatrix} = f(x_k, u_k).$$

Unlike the continuous case, we cannot generally write this explicit form of the dynamics $f(x_k, u_k)$, because they are described by the implicit DEL equation (4.14). Nevertheless, the Implicit Function Theorem guarantees that such a function exists provided that the derivative:

$$M_{k+1} = D_2 D_1 L_{k+1} + D_2 F^-_{k+1}. \tag{4.16}$$

is non-singular at q_k, p_k, and u_k. This justifies abstracting the discrete dynamics of the variational integrator this way even though the underlying implementation still calculates the update q_{k+1} by numerically solving (4.14). The purpose of this abstraction is to define the form for the linearization of the discrete dynamics. In the next section, we derive this linearization and find that the derivatives of the abstract $f(x_k, u_k, t_k)$ representation are calculated explicitly.

4.4. First Derivatives of Dynamics

Analysis and optimal control methods typically rely on the first-order linearization of system dynamics about a trajectory. This section will develop the full linearization for the unconstrained variational integrator. Constrained linearizations are discussed in Sec. 4.6. The first-order linearization of the discrete dynamics for the state form in Sec. 4.3 is:

$$\delta x_{k+1} = \frac{\partial f}{\partial x_k}\delta x_k + \frac{\partial f}{\partial u_k}\delta u_k$$

$$\begin{bmatrix} \delta q_{k+1} \\ \delta p_{k+1} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial q_{k+1}}{\partial q_k} & \dfrac{\partial q_{k+1}}{\partial p_k} \\ \dfrac{\partial p_{k+1}}{\partial q_k} & \dfrac{\partial p_{k+1}}{\partial p_k} \end{bmatrix} \begin{bmatrix} \delta q_k \\ \delta p_k \end{bmatrix} + \begin{bmatrix} \dfrac{\partial q_{k+1}}{\partial u_k} \\ \dfrac{\partial p_{k+1}}{\partial u_k} \end{bmatrix} \delta u_k. \tag{4.17}$$

Six components are required to calculate this linearization. These derivatives are found directly from the variational integrator equations (4.7), and all are calculated from explicit equations.

Derivatives of q_{k+1} are found by implicitly differentiating (4.7), and solving for the desired derivative. We start by finding $\frac{\partial q_{k+1}}{\partial q_k}$:

$$\frac{\partial}{\partial q_k}\left[p_k + D_1 L_{k+1} + F^-_{k+1} = 0\right]$$

$$0 + D_1 D_1 L_{k+1} + D_2 D_1 L_{k+1} \frac{\partial q_{k+1}}{\partial q_k} + D_1 F^-_{k+1} + D_2 F^-_{k+1} \frac{\partial q_{k+1}}{\partial q_k} = 0$$

$$\left[D_2 D_1 L_{k+1} + D_2 F^-_{k+1}\right] \frac{\partial q_{k+1}}{\partial q_k} = -\left[D_1 D_1 L_{k+1} + D_1 F^-_{k+1}\right]$$

$$\frac{\partial q_{k+1}}{\partial q_k} = -M^{-1}_{k+1}\left[D_1 D_1 L_{k+1} + D_1 F^-_{k+1}\right] \tag{4.18}$$

where M_{k+1} is as defined by (4.16) and is assumed to be non-singular. Otherwise the Implicit Function Theorem would not apply, making the state representation invalid.

The process is repeated to calculate $\frac{\partial q_{k+1}}{\partial p_k}$ and $\frac{\partial q_{k+1}}{\partial u_k}$:

$$\frac{\partial q_{k+1}}{\partial p_k} = -M_{k+1}^{-1} \tag{4.19}$$

$$\frac{\partial q_{k+1}}{\partial u_k} = -M_{k+1}^{-1} \cdot D_3 F_{k+1}^{-}. \tag{4.20}$$

Notice that each of these derivatives depends on the new configuration q_{k+1} (e.g, $D_1D_1L_{k+1} = D_1D_1L_d\left(q_k, q_{k+1}\right)$). Before evaluating the derivatives, q_{k+1} must be found by solving the DEL.

Derivatives of p_{k+1} are found by direct differentiation:

$$\frac{\partial p_{k+1}}{\partial q_k} = \left[D_2D_2L_{k+1} + D_2F_{k+1}^{+}\right] \frac{\partial q_{k+1}}{\partial q_k} + D_1D_2L_{k+1} + D_1F_{k+1}^{+} \tag{4.21}$$

$$\frac{\partial p_{k+1}}{\partial p_k} = \left[D_2D_2L_{k+1} + D_2F_{k+1}^{+}\right] \frac{\partial q_{k+1}}{\partial p_k} \tag{4.22}$$

$$\frac{\partial p_{k+1}}{\partial u_k} = \left[D_2D_2L_{k+1} + D_2F_{k+1}^{+}\right] \frac{\partial q_{k+1}}{\partial u_k} + D_3F_{k+1}^{+}. \tag{4.23}$$

These derivatives depend on (4.18)–(4.20), thus they must be evaluated first. Once calculated, their values are used in (4.21)–(4.23) along with the known value of q_{k+1} to find the derivatives of p_{k+1}. Once all six derivatives are calculated, they are organized into the two matrices in (4.17) to get the complete first-order linearization about the current state.

4.5. Second Derivatives of Dynamics

Optimal control applications frequently make use of the second-order linearization of the dynamics to improve their convergence rate. The approach used in Sec. 4.4 extends to the second derivative as well. The expanded second-order linearization of the discrete dynamics is:

$$\delta^2 x_{k+1} = \begin{bmatrix} \delta q_k \\ \delta p_k \\ \delta u_k \end{bmatrix}^T \begin{bmatrix} \dfrac{\partial^2 f}{\partial q_k \partial q_k} & \dfrac{\partial^2 f}{\partial q_k \partial p_k} & \dfrac{\partial^2 f}{\partial q_k \partial u_k} \\ \dfrac{\partial^2 f^T}{\partial q_k \partial p_k} & \dfrac{\partial^2 f}{\partial p_k \partial p_k} & \dfrac{\partial^2 f}{\partial p_k \partial u_k} \\ \dfrac{\partial^2 f^T}{\partial q_k \partial u_k} & \dfrac{\partial^2 f^T}{\partial p_k \partial u_k} & \dfrac{\partial^2 f}{\partial u_k \partial u_k} \end{bmatrix} \begin{bmatrix} \delta q_k \\ \delta p_k \\ \delta u_k \end{bmatrix}.$$

where symmetry was used to reduce the number of unique terms to six. Each derivative of f has two components, q_{k+1} and p_{k+1}, so 12 values are needed for the second-order linearization.

4.5.1. Derivatives of q_{k+1}

As with the first derivatives, the second derivatives of q_{k+1} are found by differentiating the DEL equations twice and solving for the desired derivative. We find $\frac{\partial^2 q_{k+1}}{\partial q_k \partial q_k}$ as an example:

$$\frac{\partial^2}{\partial q_k \partial q_k} \left[p_k + D_1 L_{k+1} + F^-_{k+1} = 0\right]$$

$$\frac{\partial}{\partial q_k} \left(\left[D_2 D_1 L_{k+1} + D_2 F^-_{k+1}\right] \frac{\partial q_{k+1}}{\partial q_k} = - \left[D_1 D_1 L_{k+1} + D_1 F^-_{k+1}\right] \right)$$

$$\begin{aligned} \left[D_2 D_1 L_{k+1} + D_2 F^-_{k+1}\right] \frac{\partial^2 q_{k+1}}{\partial q_k \partial q_k} + \left[D_1 D_2 D_1 L_{k+1} + D_1 D_2 F^-_{k+1}\right] \frac{\partial q_{k+1}}{\partial q_k} \\ + \left[D_2 D_2 D_1 L_{k+1} + D_2 D_2 F^-_{k+1}\right] \circ \left(\frac{\partial q_{k+1}}{\partial q_k}, \frac{\partial q_{k+1}}{\partial q_k} \right) \\ = - \left[D_1 D_1 D_1 L_{k+1} + D_1 D_1 F^-_{k+1}\right] - \left[D_2 D_1 D_1 L_{k+1} + D_2 D_1 F^-_{k+1}\right] \frac{\partial q_{k+1}}{\partial q_k} \end{aligned}$$

$$\frac{\partial^2 q_{k+1}}{\partial q_k \partial q_k} = -M_{k+1}^{-1}\Big(\left[D_1 D_1 D_1 L_{k+1} + D_1 D_1 F_{k+1}^{-}\right] + \left[D_1 D_2 D_1 L_{k+1} + D_1 D_2 F_{k+1}^{-} + D_2 D_1 D_1 L_{k+1} + D_2 D_1 F_{k+1}^{-}\right] \frac{\partial q_{k+1}}{\partial q_k} + \left[D_2 D_2 D_1 L_{k+1} + D_2 D_2 F_{k+1}^{-}\right] \circ \left(\frac{\partial q_{k+1}}{\partial q_k}, \frac{\partial q_{k+1}}{\partial q_k}\right)\Big) \quad (4.24)$$

where the notation $M \cdot (X, Y)$ represents a bilinear operator[3] M operating on X and Y.

We previously metioned that the next state x_{k+1} had to be found in order to calculate the first derivatives. Here, we see that that second derivative requires the first derivative as well. This establishes the required order for these calculations: The next state is found by the variational integrator, that is used to calculate the first derivative, and then both results are used to calculate the second derivative.

The remaining second derivatives of q_{k+1} are found by the same procedure. Derivatives with respect to state variables are:

$$\frac{\partial^2 q_{k+1}}{\partial q_k \partial p_k} = -M_{k+1}^{-1}\Big(\left[D_2 D_1 D_1 L_{k+1} + D_2 D_1 F_{k+1}^{-}\right] \frac{\partial q_{k+1}}{\partial p_k} + \left[D_2 D_2 D_1 L_{k+1} + D_2 D_2 F_{k+1}^{-}\right] \cdot \left(\frac{\partial q_{k+1}}{\partial q_k}, \frac{\partial q_{k+1}}{\partial p_k}\right)\Big) \quad (4.25)$$

$$\frac{\partial^2 q_{k+1}}{\partial p_k \partial p_k} = -M_{k+1}^{-1} \left[D_2 D_2 D_1 L_{k+1} + D_2 D_2 F_{k+1}^{-}\right] \cdot \left(\frac{\partial q_{k+1}}{\partial p_k}, \frac{\partial q_{k+1}}{\partial p_k}\right). \quad (4.26)$$

[3]This is equivalent to the matrix representation $X^T M Y$ in finite dimensions, but this notation extends to infinite dimensional spaces such as those encountered in continuous trajectory optimization.

The derivatives with respect to state and input variables are:

$$\frac{\partial^2 q_{k+1}}{\partial q_k \partial u_k} = -M_{k+1}^{-1}\Bigg(D_3 D_1 F_{k+1}^{-} + D_3 D_2 F_{k+1}^{-} \frac{\partial q_{k+1}}{\partial q_k} + \Big[D_2 D_1 D_1 L_{k+1} + D_2 D_1 F_{k+1}^{-} \Big] \frac{\partial q_{k+1}}{\partial u_k} + \big[D_2 D_2 D_1 L_{k+1} + D_2 D_2 F_{k+1}^{-} \big] \cdot \left(\frac{\partial q_{k+1}}{\partial q_k}, \frac{\partial q_{k+1}}{\partial u_k} \right) \Bigg) \tag{4.27}$$

$$\frac{\partial^2 q_{k+1}}{\partial p_k \partial u_k} = -M_{k+1}^{-1}\Bigg(D_3 D_2 F_{k+1}^{-} \frac{\partial q_{k+1}}{\partial p_k} + \big[D_2 D_2 D_1 L_{k+1} + D_2 D_2 F_{k+1}^{-} \big] \cdot \left(\frac{\partial q_{k+1}}{\partial p_k}, \frac{\partial q_{k+1}}{\partial u_k} \right) \Bigg). \tag{4.28}$$

The second derivative of the next configuration with respect to the inputs is:

$$\frac{\partial^2 q_{k+1}}{\partial u_k \partial u_k} = -M_{k+1}^{-1}\Bigg(D_3 D_3 F_{k+1}^{-} + \big[D_3 D_2 F_{k+1}^{-} + D_2 D_3 F_{k+1}^{-} \big] \frac{\partial q_{k+1}}{\partial u_k} + \big[D_2 D_2 D_1 L_{k+1} + D_2 D_2 F_{k+1}^{-} \big] \cdot \left(\frac{\partial q_{k+1}}{\partial u_k}, \frac{\partial q_{k+1}}{\partial u_k} \right) \Bigg). \tag{4.29}$$

These six derivatives make up the complete second linearization of the first half of the state.

4.5.2. Derivatives of p_{k+1}

The six derivatives of the momentum component of the state, p_{k+1} are calculated directly from the definition of the discrete momentum. The derivatives with respect to state variables are:

$$\frac{\partial^2 p_{k+1}}{\partial q_k \partial q_k} = D_1 D_1 D_2 L_{k+1} + D_1 D_1 F^+_{k+1} + \left[D_2 D_2 L_{k+1} + D_2 F^+_{k+1}\right] \frac{\partial^2 q_{k+1}}{\partial q_k \partial q_k}$$
$$+ \left[D_2 D_1 D_2 L_{k+1} + D_1 D_2 D_2 L_{k+1} + D_2 D_1 F^+_{k+1} + D_1 D_2 F^+_{k+1}\right] \frac{\partial q_{k+1}}{\partial q_k}$$
$$+ \left[D_2 D_2 D_2 L_{k+1} + D_2 F^+_{k+1}\right] \cdot \left(\frac{\partial q_{k+1}}{\partial q_k}, \frac{\partial q_{k+1}}{\partial q_k}\right) \quad (4.30)$$

$$\frac{\partial^2 p_{k+1}}{\partial q_k \partial p_k} = \left[D_2 D_1 D_2 L_{k+1} + D_2 D_1 F^+_{k+1}\right] \frac{\partial q_{k+1}}{\partial p_k} + \left[D_2 D_2 L_{k+1} + D_2 F^+_{k+1}\right] \frac{\partial^2 q_{k+1}}{\partial q_k \partial p_k}$$
$$+ \left[D_2 D_2 D_2 L_{k+1} + D_2 D_2 F^+_{k+1}\right] \cdot \left(\frac{\partial q_{k+1}}{\partial q_k}, \frac{\partial q_{k+1}}{\partial p_k}\right) \quad (4.31)$$

$$\frac{\partial^2 p_{k+1}}{\partial p_k \partial p_k} = \left[D_2 D_2 L_{k+1} + D_2 F^+_{k+1}\right] \frac{\partial^2 q_{k+1}}{\partial p_k \partial p_k}$$
$$+ \left[D_2 D_2 D_2 L_{k+1} + D_2 D_2 F^+_{k+1}\right] \cdot \left(\frac{\partial q_{k+1}}{\partial p_k}, \frac{\partial q_{k+1}}{\partial p_k}\right). \quad (4.32)$$

The derivatives with respect to state and input variables are:

$$\frac{\partial^2 p_{k+1}}{\partial q_k \partial u_k} = D_3 D_1 F^+_{k+1} + D_3 D_2 F^+_{k+1} \frac{\partial q_{k+1}}{\partial q_k} + \left[D_2 D_1 D_2 L_{k+1} + D_2 D_1 F^+_{k+1}\right] \frac{\partial q_{k+1}}{\partial u_k}$$
$$+ \left[D_2 D_2 L_{k+1} + D_2 F^+_{k+1}\right] \frac{\partial^2 q_{k+1}}{\partial q_k \partial u_k} + \left[D_2 D_2 D_2 L_{k+1} + D_2 D_2 F^+_{k+1}\right] \cdot \left(\frac{\partial q_{k+1}}{\partial q_k}, \frac{\partial q_{k+1}}{\partial u_k}\right) \quad (4.33)$$

$$\frac{\partial^2 p_{k+1}}{\partial p_k \partial u_k} = D_3 D_2 F^+_{k+1} \frac{\partial q_{k+1}}{\partial p_k} + \left[D_2 D_2 L_{k+1} + D_2 F^+_{k+1}\right] \frac{\partial^2 q_{k+1}}{\partial p_k \partial u_k}$$
$$+ \left[D_2 D_2 D_2 L_{k+1} + D_2 D_2 F^+_{k+1}\right] \cdot \left(\frac{\partial q_{k+1}}{\partial p_k}, \frac{\partial q_{k+1}}{\partial u_k}\right). \quad (4.34)$$

Finally, the second derivative with respect to the input variables is:

$$\frac{\partial^2 p_{k+1}}{\partial u_k \partial u_k} = D_3 D_3 F^+_{k+1} + \left[D_2 D_2 D_2 L_{k+1} + D_2 D_2 F^+_{k+1}\right] \cdot \left(\frac{\partial q_{k+1}}{\partial u_k}, \frac{\partial q_{k+1}}{\partial u_k}\right)$$
$$+ \left[D_3 D_2 F^+_{k+1} + D_2 D_3 F^+_{k+1}\right] \frac{\partial q_{k+1}}{\partial u_k} + \left[D_2 D_2 L_{k+1} + D_2 F^+_{k+1}\right] \frac{\partial^2 q_{k+1}}{\partial u_k \partial u_k}. \quad (4.35)$$

As with the first derivatives, the second derivatives of p_{k+1} depend on those of q_{k+1}. We handle this dependency by first evaluating (4.24)–(4.29) to obtain their numerical values and then plug those values into (4.30)–(4.35).

By evaluating each of the twelve equations above we explicitly calculate the second-order linearization for a forced system in generalized coordinates. Section 4.6 describes how this approach is extended to systems with holonomic constraints.

4.6. Constrained Systems

The linearizations in Sec. 4.4 and 4.5 must be modified for constrained mechanical systems.

Variational integrators are excellent for systems with holonomic constraints. In continuous time holonomic constraints are replaced with equivalent non-holonomic constraints (i.e, velocity constraints). As a simulation proceeds, small numeric errors often accumulate and cause the holonomic constraints to become violated. Variational integrators, on the other hand, enforce the original holonomic constraint at every time step while still preserving the symplectic form and conserving momentum.

Variational integrators for constrained systems[**29**] are derived using the same Lagrange-multiplier method used in the continuous case[**30**]. The DEL equations for a forced and constrained variational integrator are:

$$p_k + D_1 L_{k+1} + F^-_{k+1} - Dh^T(q_k)\lambda_k = 0 \tag{4.36a}$$

$$h(q_{k+1}) = 0 \tag{4.36b}$$

$$p_{k+1} = D_2 L_{k+1} + F^+_{k+1} \tag{4.36c}$$

where $h(q_k)$ is the holonomic constraint function and λ_k are the Lagrange multipliers. In this case, given p_k and q_k, a root-finding algorithm solves (4.36a) and (4.36b) to find q_{k+1} and λ_k. The updated momentum p_{k+1} is then explicitly calculated from (4.36c).

The Lagrange multipliers are completely determined by q_k, p_k, and u_k, so the state representation is unchanged. Accordingly, the same derivatives are needed to find the linearizations. Rather than derive every equation we demonstrate the process with one component of the first and second derivatives.

For the first-order linearization, we find $\frac{\partial q_{k+1}}{\partial q_k}$. We start by differentiating (4.36a):

$$\begin{aligned} &\tfrac{\partial}{\partial q_k}\left[p_k + D_1 L_{k+1} + F^-_{k+1} - Dh^T(q_k)\lambda_k = 0\right] \\ &\tfrac{\partial q_{k+1}}{\partial q_k} = -M_k^{-1}\left[C_{q_k} - Dh^T(q_k)\tfrac{\partial \lambda_k}{\partial q_k}\right] \end{aligned} \tag{4.37}$$

where

$$C_{q_k} = D_1 D_1 L_{k+1} + D_1 F^-_{k+1} - D^2 h^T(q_k)\lambda_k.$$

To evaluate this derivative, we must calculate $\frac{\partial \lambda_k}{\partial q_k}$. This is found by differentiating (4.36b), substituting in (4.37), and solving for $\frac{\partial \lambda_k}{\partial q_k}$:

$$\begin{aligned}
\tfrac{\partial}{\partial q_k}\left[h(q_{k+1}) = 0\right] \\
Dh(q_{k+1})\tfrac{\partial q_{k+1}}{\partial q_k} = 0 \\
Dh(q_{k+1})M_k^{-1}\Big[C_{q_k} - Dh^T(q_k)\tfrac{\partial \lambda_k}{\partial q_k}\Big] = 0 \\
Dh(q_{k+1})M_k^{-1}C_{q_k} - Dh(q_{k+1})M_k^{-1}Dh^T(q_k)\tfrac{\partial \lambda_k}{\partial q_k} = 0 \\
\tfrac{\partial \lambda_k}{\partial q_k} = \left[Dh(q_{k+1})M_k^{-1}Dh^T(q_k)\right]^{-1} Dh(q_{k+1})M_k^{-1}C_{q_k}. \qquad (4.38)
\end{aligned}$$

To calculate $\frac{\partial q_{k+1}}{\partial q_k}$, the the constrained DEL equation (4.36) is solved numerically to find q_{k+1} and λ_k. These values are used in (4.38) to find $\frac{\partial \lambda_k}{\partial q_k}$. Finally, $\frac{\partial q_{k+1}}{\partial q_k}$ is calculated with (4.37).

The same approach is used to find the remaining components of the first derivative. We do not repeat the derivation here. We continue onto the second derivative by calculating $\frac{\partial^2 q_{k+1}}{\partial q_k \partial q_k}$:

$$\begin{aligned}
\tfrac{\partial^2}{\partial q_k \partial q_k}\left[p_k + D_1 L_{k+1} + F_{k+1}^{-} - Dh^T(q_k)\lambda_k = 0\right] \\
\tfrac{\partial^2 q_{k+1}}{\partial q_k \partial q_k} = -M_{k+1}^{-1}\Big(C_{q_k q_k} - Dh^T(q_k)\tfrac{\partial^2 \lambda_k}{\partial q_k \partial q_k}\Big) \qquad (4.39)
\end{aligned}$$

where

$$C_{q_k q_k} = D_1 D_1 D_1 L_{k+1} + D_1 D_1 F^-_{k+1} + \left[D_2 D_2 D_1 L_{k+1} + D_2 D_2 F^-_{k+1}\right] \cdot \left(\tfrac{\partial q_{k+1}}{\partial q_k}, \tfrac{\partial q_{k+1}}{\partial q_k}\right)$$
$$+ \left[D_2 D_1 D_1 L_{k+1} + D_2 D_1 F^-_{k+1} + D_1 D_2 D_1 L_{k+1} + D_1 D_2 F^-_{k+1}\right] \tfrac{\partial q_{k+1}}{\partial q_k}$$
$$- D^3 h^T(q_k)\lambda_k - 2D^2 h^T(q_k) \tfrac{\partial \lambda_k}{\partial q_k}.$$

Again, we find the corresponding second derivative of λ_k by differentiating (4.36b) twice:

$$\tfrac{\partial^2}{\partial q_k \partial q_k}\left[h(q_{k+1}) = 0\right]$$
$$D^2 h(q_{k+1}) \cdot \left(\tfrac{\partial q_{k+1}}{\partial q_k}, \tfrac{\partial q_{k+1}}{\partial q_k}\right) + Dh(q_{k+1}) \tfrac{\partial^2 q_{k+1}}{\partial q_k \partial q_k} = 0.$$

We substitute in (4.39) and solve for $\frac{\partial^2 \lambda_k}{\partial q_k \partial q_k}$:

$$\tfrac{\partial^2 \lambda_k}{\partial q_k \partial q_k} = \left[Dh(q_{k+1}) M^{-1}_{k+1} Dh^T(q_k)\right] \cdot$$
$$\left[Dh(q_{k+1}) M^{-1}_{k+1} C_{q_k q_k} - D^2 h(q_{k+1}) \cdot \left(\tfrac{\partial q_{k+1}}{\partial q_k}, \tfrac{\partial q_{k+1}}{\partial q_k}\right)\right]. \quad (4.40)$$

To calculate this $\frac{\partial^2 q_{k+1}}{\partial q_k \partial q_k}$, we solve for the next state, calculate the first derivatives, evaluate (4.40) to find $\frac{\partial^2 \lambda_k}{\partial q_k \partial q_k}$, and finally evaluate (4.39) to find the second derivative. This same procedure is used to calculate the other components of the constrained second derivative.

Note that the constrained momentum update (4.36c) is identical to the unconstrained case, so the first- and second-order linearizations are identical.

4.7. Kinematic Configuration Variables

Section 3.6 introduced kinematic configuration variables for continuous systems. The discrete Euler Lagrange equations are easily modified to handle kinematic variables.

If we split the generalized coordinates into a dynamic part q and kinematic part ρ, but still consider both to be dynamic, the DEL equations can written as:

$$\begin{aligned}&D_2L_d(q_{k-1}, q_k, \rho_{k-1}, \rho_k) + f_d^{d+}(q_{k-1}, q_k, \rho_{k-1}, \rho_k, \mu_{k-1}, t_{k-1}, t_k)\\ &\quad + D_1L_d(q_k, q_{k+1}, \rho_k, \rho_{k+1}) + f_d^{d-}(q_k, q_{k+1}, \rho_k, \rho_{k+1}, \mu_k, t_k, t_{k+1}) - D_1h^T(q_k, \rho_k)\lambda_k = 0\end{aligned} \tag{4.41}$$

$$\begin{aligned}&D_4L_d(q_{k-1}, q_k, \rho_{k-1}, \rho_k, t_{k-1}, t_k) + f_d^{k+}(q_{k-1}, q_k, \rho_{k-1}, \rho_k, \mu_{k-1}, t_{k-1}, t_k)\\ &+D_3L_d(q_k, q_{k+1}, \rho_k, \rho_{k+1}, t_k, t_{k+1}) + f_d^{k-}(q_k, q_{k+1}, \rho_k, \rho_{k+1}, \mu_k, t_k, t_{k+1}) - D_2h^T(q_k, \rho_k)\lambda_k = 0\end{aligned} \tag{4.42}$$

$$h(q_{k+1}, \rho_{k+1}) = 0 \tag{4.43}$$

where $f_d^{d\pm}$ and $f_d^{k\pm}$ are the generalized forces on the dynamic and discrete variables, respectively. These first two equations represent the dynamics of q and ρ, respectively.

As with the continuous case, the kinematic assumption implies that at each step, a u_k can be found that satisfies (4.42) for any values of q_{k+1} and ρ_{k+1}. Therefore, we drop this equation and make ρ_{k+1} a direct input to the dynamics. The new variational integrator for mixed kinematic-dynamic models is defined by (4.41) and (4.43). The dynamic equation (4.41) is unchanged other than the additional inputs to the discrete functions. Due to this, adding support for mixed models is trivial with variational integrators. It only requires removing the kinematic variables from the set of variables found by the root solver.

Appendix A.2 presents the equations for a full mixed dynamic-kinematic model with forcing and constraints, including the modified state representation and the entire first and second derivative equations.

CHAPTER 5

Examples

The ideas presented in this thesis have been implemented in a software package named `trep`. The software implements the complete dynamics, first derivative, and second derivative for continuous and discrete systems based on the tree representation. It is easily extendable to include new types of constraints, forces, or potential energies. *Trep* also provides facilities for the trajectory optimization technique introduced in the following chapter.

Trep is implemented as a Python package with a C back-end for performance critical sections. This arrangement makes it convenient to use without sacrificing speed. `trep` is open source and is freely at `http://trep.googlecode.com`.

This chapter provides several numerical example applications of the discrete system representation and comparisons to the continuous domain. Section 5.1 discusses a scissor lift system and includes comparisons to the continuous time domain. Simulations of an actuated two-dimension arm are presented in Sec. 5.2. Section 5.3 uses a humanoid maroinette to demonstrate a stabilizing feedback law that was automatically generated using the discrete linearization.

5.1. The Scissor Lift

The mechanism shown in Fig. 5.1 is commonly found in industrial lifts and is an excellent example for comparing simulations with constraints. We consider the mechanism to be hanging rather than lifting to avoid introducing actuation.

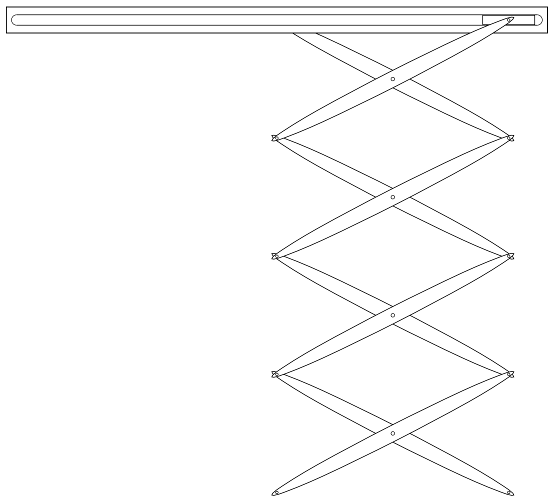


Figure 5.1. The scissor lift has many bodies and closed kinematic chains but only one degree of freedom.

The scissor lift has many links and many closed kinematic chains, but can be reduced to a single degree of freedom (DOF). Its complexity is parameterized by varying the number of segments. We can write the Lagrangian for the equivalent one DOF system and use an accurate numeric integrator to generate a benchmark trajectory for comparison.

A schematic of the device is shown in Fig. 5.2. The top segment is pinned in the upper left. The upper right joint is pinned to a mass m_S that slides horizontally without friction. Each link has mass m_L at its center and rotational inertia I.

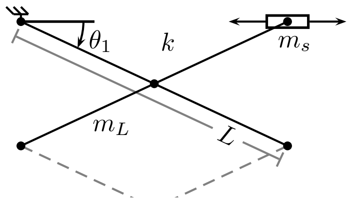


Figure 5.2. A schematic for the top of the scissor lift. Each link has mass m_L at the center and rotational inertia I.

The Lagrangian for a lift is

$$L(\theta,\dot{\theta}) = \sum_{n=1}^{N} \left(m_L L^2 \left(\left(\tfrac{1}{2} - n\right)^2 \cos^2\theta_1 + \tfrac{1}{2}\sin^2\theta_1 + I \right) \dot{\theta}_1^2 \right.$$
$$\left. + 2 m_L g (n - \tfrac{1}{2}) L \sin\theta_1 \right) + \tfrac{1}{2} m_S L^2 \sin^2\theta_1 \theta_1^2$$

where N is the number of segments.

5.1.1. Trajectory Accuracy

A benchmark solution was generated for a 5 segment lift from the above Lagrangian using Mathematica's `NDSolve` function. The system was simulated with `trep` for 10 seconds with a time step of $0.01s$ and took 1.74 seconds to compute. The system was also simulated in `ODE` with time steps of $0.01s$, $0.001s$, and $0.0001s$ and took 0.19, 1.85, and 18.47 seconds to compute, respectively. The simulated trajectories are plotted in Fig. 5.3.

The variational integrator tracks the benchmark solution almost perfectly while the `ODE` simulation is clearly wrong. The $0.01s$ trajectory dissipates most of the energy immediately. Small time steps dissipate energy more slowly but still depart from the true trajectory quickly. All three `ODE` solutions result in an incorrect period of oscillation.

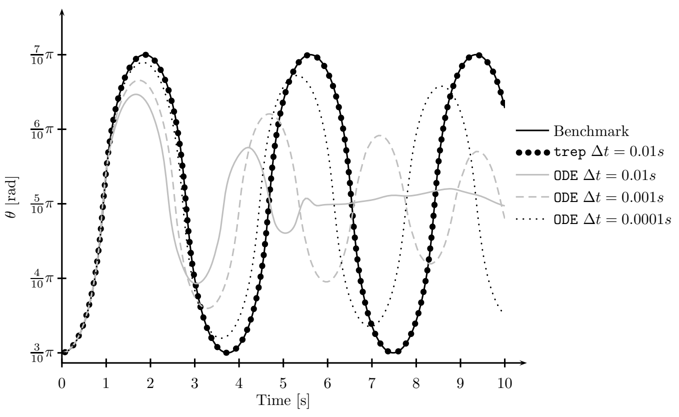


Figure 5.3. Simulated trajectories for the scissor lift with 5 segments.

The variational integrator continues to perform well for long time scales. Even after 1000 seconds, there is only a slight phase shift from accumulated error while the amplitude, shape, and frequency are still close to the benchmark trajectory.

5.1.2. Complexity Analysis

Dynamics algorithms are also compared by complexity. That is, the number of operations needed to calculate $f(\cdot)$ in $\dot{x} = f(x, u)$. However, this is not always the relevant notion of complexity; it assumes that we are only interested in the complexity of evaluating a single step of an integrator and ignored other factors, like the quality of the solution and post-operations performed to fix constraints. As seen above in Fig. 5.3, results from the two methods can differ significantly in quality for a given time step. A meaningful comparison must take into account both the computational effort and the quality of the solution.

Our suggested approach is to compare the computational effort to generate a trajectory that is within a specified error of a benchmark trajectory. Given a benchmark trajectory

$\theta_b(t)$ and a simulation result $\theta_s(t)$, the simulation error is defined as

$$e = \int_{t_0}^{t_f} \left(\theta_b(\tau) - \theta_s(\tau)\right)^2 \mathrm{d}\tau \tag{5.1}$$

First, we choose a desired error and simulation time. An N-segment lift is simulated until the error exceeds the desired error. If the simulation time is longer than the desired time, we increase the step size and re-run the simulation. If the simulation time is shorter, we decrease the step size. We iterate until the achieved simulation time is approximately equal to the desired time.

Figure 5.4 shows the results of this analysis when the desired time was 15 seconds and the desired error was 0.1. In this case `ODE` scales so poorly that the results must be plotted logarithmically. The results show that despite `ODE` having linear dynamics, it does not perform as well as the variational integrator because the integrator time step must be reduced to maintain an acceptable error.

The results of this analysis are obviously limited to the scissor lift. There may be other examples that favor `ODE` over `trep`. The main point is that the result of traditional complexity analysis that `ODE` should always scale better can be misleading in practice because of error.

5.1.3. System Identification

A common argument for the artificial energy loss seen with holonomic constraints in continuous systems is that all real systems have friction and damping, so energy conservation is irrelevant. There are two strong points against this argument. First, friction is not random noise in the dynamics; it affects the system in a structured way that we can and

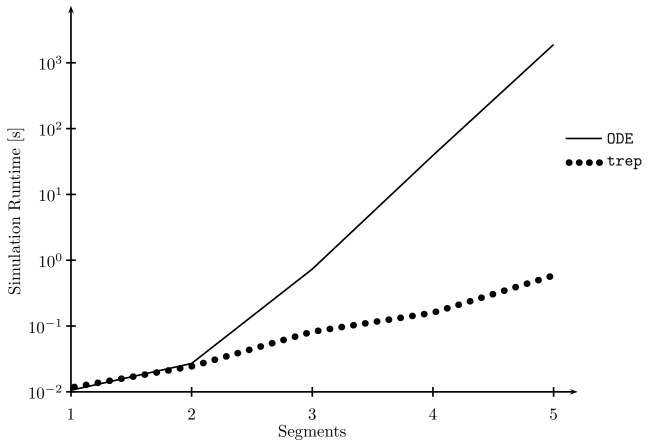


Figure 5.4. Simulation runtime vs. Number of Segments in a scissor Lift. The simulation runtime was found by setting the time step so that the simulation time was $15s \pm 1\%$ when accumulated error exceeded 0.1. The y-axis is logarithmic.

should model. Second, the rate of artificial energy loss will not equal the true energy loss. If more real loss is occurring, damping will still have to be added. If less real loss is occurring, energy *must be added to the system!*

To illustrate this point, consider a system identification experiment for the scissor lift. A linear friction model (i.e. $\tau = -c\dot{\theta}$) dissipates energy at each joint at the ends of the links. A sinusoidal input force acts on the sliding block to establish a non-trivial steady state solution.

Mathematica's `NDSolve` generates a benchmark solution from the one-dimensional Lagrangian model (augmented to include the dissipation and forcing). The damping constant for each model was varied until the model produced a steady state solution with an amplitude within 1% of the benchmark solution's amplitude.

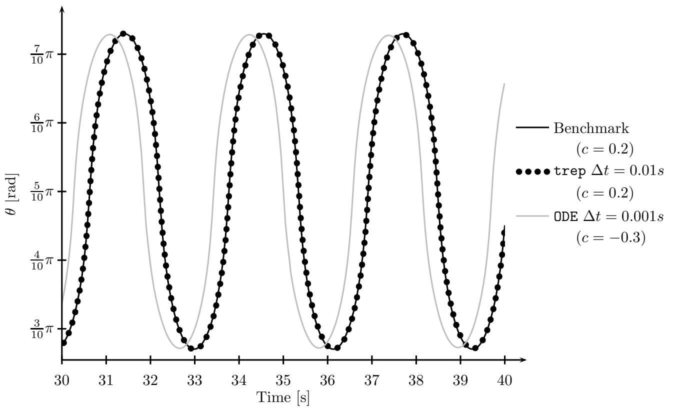


Figure 5.5. The scissor lift is simulated with damping and a sinusoidal input force until it reaches a steady state periodic solution. `ODE` and `trep` were used to determine the scissor lift's damping constant from benchmark data. `trep` found the correct damping constant while `ODE` predicts an unstable system. The phase offset indicates differences between `ODE` and the benchmark during the transient response.

The benchmark system was simulated with a damping constant of $c = 0.2$. `trep` used a time step of 0.01s and found the correct damping constant $c = 0.2$ with an amplitude error of -0.11%. `ODE` used a time step of 0.0013s[1] and found a damping constant of $c = -0.3$ with an amplitude error of -0.97%. The resulting trajectories are shown in Fig. 5.5.

Note that the model found using `ODE` is *unstable* because of the artificial energy dissipation. The model is useless for further analysis.

It is a common misconception that variational integrators cannot model dissipative systems. This example emphasizes that variational integrators are able to handle both dissipation and forcing.

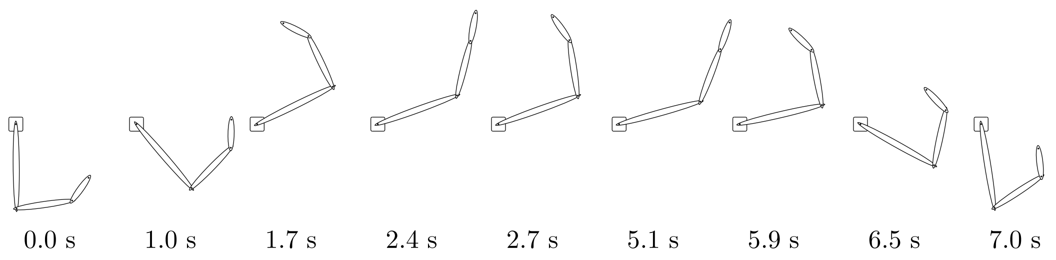


Figure 5.6. The actuated 3-DOF arm was simulated with the variational integrator.

5.2. Actuated Robotic Arm

Variational integrators are well suited to the forced mechanical systems found in robotics. We demonstrate a three degree of freedom arm. Each joint of the arm is actuated.

We begin with a desired trajectory, $q_d(t)$ and $\dot{q}_d(t)$ (in this case, a waving motion). The inverse dynamics [**30**] of this controllable, non-degenerate system are used to find the corresponding inputs, $u_d(t)$. The arm is unstable in the raised position, so a stabilizing time-varying feedback law, $K(t)$ is found from the Riccati equation with the linearization of the arm dynamics along the desired trajectory. The combined control law takes is

$$f_c(t, q, \dot{q}) = u_d(t) - K(t) \begin{bmatrix} q - q_d(t) \\ \dot{q} - \dot{q}_d(t) \end{bmatrix} \tag{5.2}$$

The forced arm was simulated in trep using (4.12) to obtain the discrete forcing terms from (5.2). Snapshots from the simulation are shown in Fig. 5.6.

A comparison trajectory was generated from the continuous Euler-Lagrange equations with Mathematica's `NDSolve`. The configuration variable trajectories for both simulations are shown in Fig. 5.7. The simulations clearly agree.

[1]`ODE`'s time step was chosen so that the simulation took approximately as long `trep`

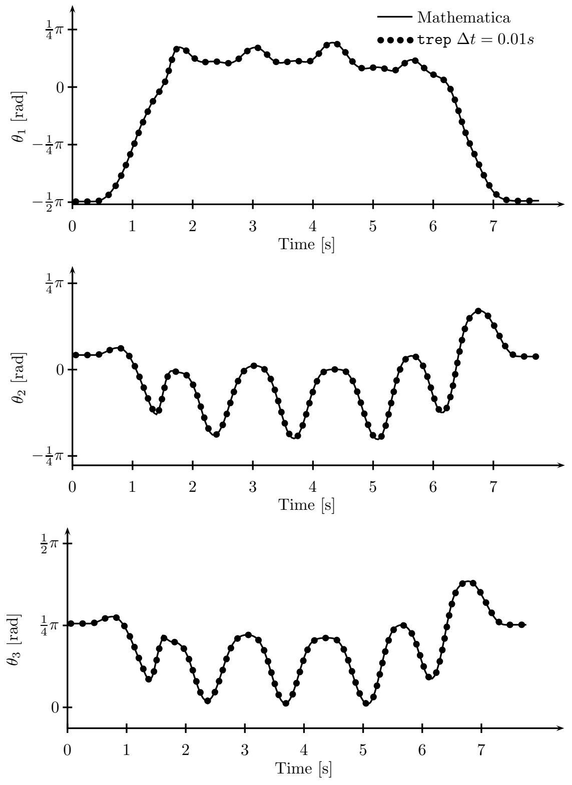


Figure 5.7. The continuous Euler-Lagrange and discrete variational integrator simulations of the actuated arm give equivalent results.

We emphasize that the inputs and feedback law, which were obtained using classical techniques, were used without any modification for the variational integrator.

5.3. Marionette

As an example, we use the Linear Quadratic Regulator (LQR) method to generate a stabilizing feedback controller for the marionette in Fig. 5.8. The marionette has 22 dynamic configuration variables, 18 kinematic configuration variables[**17**], and 6 holonomic constraints. The corresponding state-space model has 80 state and 18 input variables.

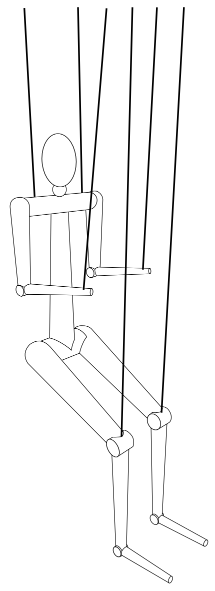


Figure 5.8. The marionette model has 40 configuration variables and 6 holonomic constraints.

For a non-linear system like the marionette, the dynamics can be linearized about a known trajectory. The LQR solution for the linearization yields a feedback law that stabilizes the system near the known trajectory[].

The discrete LQR problem finds an optimal feedback law for a discrete linear system[**3**]:

$$z_{k+1} = A_k z_k + B_k \mu_k.$$

For the linearized dynamics about a non-linear system's trajectory, this corresponds to $A_k = \frac{\partial f(x_k,u_k)}{\partial x_k}$ and $B_k = \frac{\partial f(x_k,u_k)}{\partial u_k}$. The solution to the discrete LQR problem is found by iteratively solving the discrete Ricatti equation:

$$P_k = Q_k + A_k^T P_{k+1} A_k - B_k P_{k+1} B_k^T \left[R_k + B_k^T P_{k+1} B_k \right]^{-1} B_k^T P_{k+1} B_k \tag{5.3a}$$

$$P_{k_f} = Q_{k_f} \tag{5.3b}$$

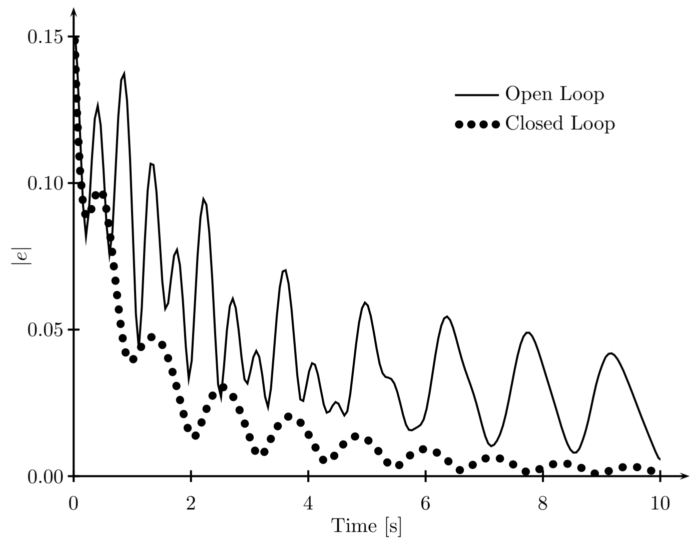


Figure 5.9. The discrete LQR feedback law significantly improves the error response of the marionette compared to the open-loop simulation.

where Q_k and R_k correspond to the cost of the linearized state and inputs, and Q_{k_f} determines the terminal cost of the linearized state. The Ricatti equation is solved to find P_k by recursively evaluating (5.3a) backwards in time from the boundary condition (5.3b). The solution is used to calculate a stabilizing feedback law:

$$\mathcal{K}_k = \left[R_k + B_k^T P_{k+1} B_k\right]^{-1} B_k^T P_{k+1} B_k.$$

The marionette was simulated and linearized about a 10.0 second trajectory using the midpoint variational integrator in `trep`. The linearization was used to create a locally stabilizing controller by solving the discrete LQR problem with identity matrices for each cost matrix. A perturbation was then added to the initial condition and the simulation was performed with and without the added stabilizing feedback controller.

The resulting error between the perturbed and original trajectories is shown in Fig. 5.9. The closed-loop trajectory converges significantly faster compared to the open-loop trajectory as expected. The ability to generate locally stabilizing feedback laws for complex systems that are simulated with variational integrators is a useful application of the methods described here.

The optimization was performed on an Intel i7-2760QM CPU at 2.40GHz. The simulation takes approximately 2.11ms per step, the linearization takes approximately 1.12ms per step. The second-order linearization takes approximately 22.15ms per step, though it was not required for this example.

5.4. Conclusion

These exampels have highlighted some applications where discrete variational integrators provide signficant improvements compared to models in the continuous time domain. The projection operator-based trajectory optimization is introduced next in Ch. 6. Further examples are found afterwards in Sec. 6.7.

CHAPTER 6

Trajectory Optimization

This chapter discusses projection operator-based trajectory optimization for continuous and discrete dynamic systems. This strategy iteratively improves a known trajectory until a local minimum of the cost is found. The optimization works for systems with constraints and with under-actuated systems.

The continuous time projection-operator trajectory optimization is a previously known result[**15**]. It is presented here for reference and comparison with the discrete time equivalent. The discrete time formulation is a contribution of this thesis. The algorithm applies to arbitrary discrete time dynamic systems, but was motivated by the discrete time models based on variational integrators that were described in Ch. 4.

The trajectory optimization problem is stated for both discrete and continuous systems in Sec. 6.1. The projection operator is discussed in Sec. 6.2. The details of the continuous operator are given in Sec. 6.2.1 and the discrete operator in Sec. 6.2.2.

Section 6.3 introduces the optimization algorithm. Each iteration of the algorithm can be broken into three steps. The first is calculating the projection operator, discussed in Sec. 6.3.1. The second step, described in Sec. 6.3.2, is calculating the descent direction. Three different descent directions are described: Newton's method in Sec. 6.3.2.2, steepest descent in Sec. 6.3.2.3, and a quasi-Newton method in Sec. 6.3.2.4. The final step of each iteration is the line search along the descent direction, discussed in Sec. 6.3.3.

Those sections apply to both the continuous and discrete time domains. Details specific to continuous time are discussed in Sec. 6.4, and discrete time in Sec. 6.5.

Section 6.6 discusses several practical methods to find improved initial trajectories for optimizations.

Results of several example applications are presented in Sec. 6.7. The example systems are a pendulum on a cart (Sec. 6.7.1), a string-actuated, two-dimensional marionette arm (Sec. 6.7.2), a string-actuated, three-dimensional humanoid marionette (Sec. 6.7.3), and a tendon-actuated human hand (Sec. 6.7.4).

6.1. Problem Definition

We seek trajectories for a continuous or discrete dynamic system that minimize a cost function. Let $\mathcal{T}$ be the set of dynamics admissible trajectories for a dynamic system. The trajectory space is embedded in a inner product space V so that $\mathcal{T} \subseteq V$. In continuous time, $\mathcal{T} = \{\xi = (x, u) \in V : \dot{x}(t) = f(x(t), u(t), t)\}$. In discrete time, $\mathcal{T} = \{\xi = (x, u) \in V : x(k+1) = f(x(k), u(k), k)\}$.

Throughout this chapter, $\xi = (x, u)$ is always an element of $\mathcal{T}$ and $\delta\xi = (\delta x, \delta u)$ is always an element of the tangent trajectory space $T\mathcal{T}$. Elements of V use the symbols $\bar{\xi} = (\bar{x}, \bar{u})$ while elements in the tangent space TV use the notation $\delta\bar{\xi} = (\delta\bar{x}, \delta\bar{u})$. Elements $\bar{\xi} \in V$ will be referred to as trajectories even though they may not satisfy the system dynamics.

The discrete trajectory space is formally finite-dimensional. However, in this derivation it is treated as infinite-dimensional like the continuous trajectory space. This approach leads to an algorithm that explicitly takes advantage of the discrete dynamics, and avoids directly optimizing over the entire (potentially very large) dimensionality of the discrete space. For the remainder of the chapter, we refer to both spaces as infinite-dimensional.

Continuous Problem Statement: Given an initial trajectory $\xi_0 = (x, u) \in \mathcal{T}$, find

$$\begin{aligned} \xi^* &= \arg\min_{\xi \in \mathcal{T}} h(\xi) \\ \text{where } h(\xi) &= \int_{t_0}^{t_f} \ell(x(t), u(t), t)\mathrm{d}t + m(x(t_f)). \end{aligned}$$

Discrete Problem Statement: Given an initial trajectory $\xi_0 = (x, u) \in \mathcal{T}$, find

$$\begin{aligned} \xi^* &= \arg\min_{\xi \in \mathcal{T}} h(\xi) \\ \text{where } h(\xi) &= \sum_{k=k_0}^{k_f - 1} \ell(x(k), u(k), k) + m(x(k_f)). \end{aligned}$$

In both cases, the initial condition $x(0)$ of the trajectory may be included in the optimization or considered fixed.

For example, suppose we want the system to track a desired state path x_d as closely as possible. We can treat this as an optimization problem with the costs

$$\begin{aligned} \ell(x, u, k) &= (x - x_d(k))^T Q (x - x_d(k)) + u^T R u \\ m(x) &= (x - x_d(k_f))^T Q (x - x_d(k_f)) \end{aligned}$$

where Q and R are positive definite matrices. By solving this optimization, we obtain a trajectory that approximates the desired path even if the path is not dynamically feasible or the system is under-actuated.

These are *constrained* optimizations because solutions must satisfy the dynamics of the system. The approach described here introduces a projection operator that takes arbitrary curves in V and projects them to nearby trajectories. The projection operator effectively removes the dynamics constraint. The unconstrained optimization is solved using an iterative approach that improves the cost at each iteration by calculating a descent direction, and by performing a one dimensional line search along that direction.

The next section introduces the continuous and discrete projection operators and their derivatives.

6.2. Projection Operators

The optimization algorithm is based on a projection operator $\xi = \mathcal{P}(\bar{\xi})$ that maps curves $\bar{\xi} \in V$ to nearby trajectories of the system $\xi \in \mathcal{T}$. The implementation details of the continuous and discrete projection operators are discussed in Sec. 6.2.1 and Sec. 6.2.2. First we introduce some common properties of the projection operators.

The continuous and discrete projection operators are both created by integrating the system dynamics while tracking the $\bar{\xi}$ with a linear feedback law. To be projections, both operators must have the property that if $\xi \in \mathcal{T}$, then $\mathcal{P}(\xi) = \xi$. A consequence of this is that the projections are idempotent:

$$\mathcal{P}(\mathcal{P}(\xi)) = \mathcal{P}(\xi). \tag{6.1}$$

Both projection operators are twice differentiable[1]. The first derivative $\delta\xi = D\mathcal{P}(\bar{\xi})\circ\delta\bar{\xi}$ is also a projection:

$$D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi} = D\mathcal{P}(\bar{\xi}) \circ D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi}. \tag{6.2}$$

The derivative maps elements of $\delta\bar{\xi} \in T_{\bar{\xi}}V$ to tangent trajectories $\delta\xi \in T_{\mathcal{P}(\bar{\xi})}\mathcal{T}$. The derivative has the useful invariance property that it is the same whether evaluated at $\bar{\xi}$ or the projected trajectory $\mathcal{P}(\bar{\xi})$:

$$D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi} = D\mathcal{P}(\mathcal{P}(\bar{\xi})) \circ \delta\bar{\xi}. \tag{6.3}$$

The second derivative of the projection $\delta^2\xi = D^2\mathcal{P}(\bar{\xi}) \circ (\delta\bar{\xi}, \delta\bar{\xi})$ has two similar invariance properties:

$$D^2\mathcal{P}(\bar{\xi}) \circ (\delta\bar{\xi}, \delta\bar{\xi}) = D^2\mathcal{P}(\mathcal{P}(\xi)) \circ (\delta\bar{\xi}, \delta\bar{\xi}) \tag{6.4}$$

$$D^2\mathcal{P}(\bar{\xi}) \circ (\delta\bar{\xi}, \delta\bar{\xi}) = D^2\mathcal{P}(\xi) \circ (D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi}, D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi}) \tag{6.5}$$

The properties are important for calculating the descent direction in Sec. 6.3.2.

[1]We presume that the system dynamics are twice differentiable.

6.2.1. Continuous Projection Operators

A suitable projection operator for a continuous system $\frac{\partial}{\partial t}x(t) = f(x(t), u(t), t)$ can be created by tracking the curve $\bar{\xi} \in V$ with a stabilizing linear feedback law $\mathcal{K}(t)$:

$$\xi = \mathcal{P}(\bar{\xi}) \tag{6.6a}$$

$$x(t_0) = \bar{x}(t_0) \tag{6.6b}$$

$$\frac{\partial}{\partial t}x(t) = f(x(t), u(t), t) \tag{6.6c}$$

$$u(t) = \bar{u}(t) - \mathcal{K}(t)(x(t) - \bar{x}(t)). \tag{6.6d}$$

A suitable feedback law can be found, for example, by solving the LQR problem about a known trajectory of the system, reviewed in Sec. B.1.2.

Proposition 1. *The projection operator defined by* (6.6) *satisfies idempotent projection property* (6.1)

Proof. Let $\xi_1 = (x_1, u_1) \in \mathcal{T}$ and $\xi_2 = (x_2, u_2) = \mathcal{P}(\xi_1)$. The definition $\xi_1 \in \mathcal{T}$ implies $\dot{x}_1(t) = f(x_1(t), u_1(t), t)$. By (6.6), we have:

$$x_2(t_0) = x_1(t_0)$$

$$\dot{x}_2(t) = f(x_2(t), u_1(t) - \mathcal{K}(t)\left[x_2(t) - x_1(t)\right], t).$$

Let $z(t)$ be the difference between the two trajectories:

$$\begin{aligned}\dot{z}(t) &= x_2(t) - x_1(t)\\ &= f(x_2(t), u_1(t) - \mathcal{K}(t)\left[x_2(t) - x_1(t)\right], t) - f(x_1(t), u_1(t), t)\\ &= f(x_1(t) + z(t), u_1(t) - \mathcal{K}(t)z(t), t) - f(x_1(t), u_1(t), t).\end{aligned}$$

When $z(t) = 0$, we have:

$$\dot{z}(t) = f(x_1(t) + 0, u_1(t) - 0, t) - f(x_1(t), u_1(t), t) = 0$$

Since $z(t_0) = x_2(t_0) - x_1(t_0) = x_1(t_0) - x_1(t_0) = 0$ and $z(t) = 0$ implies $\dot{z}(t) = 0$, then $z(t) = 0 \;\forall\; t >= t_0$. □

6.2.1.1. First Derivative of $\mathcal{P}(\bar{\xi})$. The derivative of the projection operator $\delta\xi = D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi}$ is found by differentiating (6.6) along a direction $\delta\bar{\xi} = (\delta\bar{x}, \delta\bar{u})$:

$$\delta\xi = (\delta x, \delta u) = D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi}$$

$$\xi = \mathcal{P}(\bar{\xi}) \tag{6.7a}$$

$$\delta x(t_0) = \delta\bar{x}(t_0) \tag{6.7b}$$

$$\begin{aligned}\tfrac{\partial}{\partial t}\delta x(t) &= \frac{\partial f}{\partial x}(t)\delta x(t) + \frac{\partial f}{\partial u}(t)\delta u(t) \qquad (6.7\text{c})\\ &= Df(t) \circ \delta\xi(t)\end{aligned}$$

$$\delta u(t) = \delta\bar{u}(t) - \mathcal{K}(t)(\delta x(t) - \delta\bar{x}(t)) \tag{6.7d}$$

where $\frac{\partial f}{\partial x}(t)$ is shorthand for $\frac{\partial f}{\partial x}(x(t), u(t), t)$. (The same applies for $\frac{\partial f}{\partial u}(t)$ and $Df(t)$.)

Comparing (6.7) with the projection (6.6) shows that $D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi}$ is a projection itself that maps $\delta\bar{\xi}$ to trajectories $\delta\xi$ of the time-varying linear system $\frac{\partial}{\partial t}\delta x(t) = A(t)\delta x(t) + B(t)\delta u(t)$ with $A(t) = \frac{\partial f}{\partial x}(t)$ and $B(t) = \frac{\partial f}{\partial u}(t)$. Therefore, $D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi}$ satisfies the idempotent projection property (6.2).

The invariance property (6.3) follows directly from (6.7a).

6.2.1.2. Second Derivative of $\mathcal{P}(\bar{\xi})$. The second derivative of the projection operator is found directly from (6.11). We only consider the second derivative with respect to the same directions.

$$\delta^2\xi = D^2\mathcal{P}(\bar{\xi}) \circ (\delta\bar{\xi}, \delta\bar{\xi})$$

$$\xi = \mathcal{P}(\bar{\xi}) \tag{6.8a}$$

$$\delta\xi = D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi} \tag{6.8b}$$

$$\delta^2 x(t_0) = 0 \tag{6.8c}$$

$$\begin{aligned} \tfrac{\partial}{\partial t}\delta^2 x(t) &= D^2 f(t) \circ (\delta\xi(t), \delta\xi(t)) + Df(t) \circ \delta^2\xi(t) \qquad (6.8\text{d}) \\ &= \frac{\partial f}{\partial x}(t)\delta^2 x(t) + \frac{\partial f}{\partial u}(t)\delta^2 u(t) + D^2 f(t) \circ (\delta\xi(t), \delta\xi(t)) \end{aligned}$$

$$\delta^2 u(t) = -\mathcal{K}(t)\delta^2 x(t) \tag{6.8e}$$

The invariance properties (6.4) and (6.5) follow directly from (6.8a) and (6.8b), respectively.

The second derivative is not a projection, but solutions can be expressed as the dynamics of a linear system. We see this by substituting the feedback (6.8e) into the dynamics equation (6.8d):

$$\tfrac{\partial}{\partial t}\delta^2 x(t) = \left[\frac{\partial f}{\partial x}(t) - \frac{\partial f}{\partial u}(t)\mathcal{K}(t)\right]\delta^2 x(t) + D^2 f(t) \circ (\delta\xi(t), \delta\xi(t))$$

This is an affine system, equivalent to a linear system with an input:

$$\begin{aligned}\tfrac{\partial}{\partial t}\delta^2 x(t) &= \hat{A}(t)\delta^2 x(t) + \hat{B}(t)\\ \hat{A}(t) &= \left[\frac{\partial f}{\partial x}(t) - \frac{\partial f}{\partial u}(t)\mathcal{K}(t)\right]\\ \hat{B}(t) &= D^2 f(t) \circ (\delta\xi(t), \delta\xi(t))\end{aligned}$$

Solutions to the system dynamics can be expressed in closed form in terms of the state transition matrix (Sec. B.1.1):

$$\delta^2 x(t) = \Phi(t, t_0)\delta^2 x(t_0) + \int_{t_0}^{t} \Phi(t, s)\hat{B}(s)\mathrm{d}s \tag{6.9}$$

This leads to an elegant calculation of the second derivative of the projected cost, discussed in Sec. 6.4.2.

6.2.2. Discrete Projection Operators

The projection operator for a discrete dynamic system $x(k+1) = f(x(k), u(k), k)$ is analogous to the continuous time case:

$$\xi = (x, u) = \mathcal{P}\left(\bar{\xi} = (\bar{x}, \bar{u})\right):$$

$$x(k_0) = \bar{x}(k_0) \tag{6.10a}$$

$$x(k+1) = f(x(k), u(k), k) \tag{6.10b}$$

$$u(k) = \bar{u}(k) - \mathcal{K}(k)(x(k) - \bar{x}(k)) \tag{6.10c}$$

where $\mathcal{K}(k)$ is a stabilizing feedback law for the dynamic system $f(x, u, k)$. $\mathcal{K}(k)$ is typically found by solving the discrete LQR[**3**][**27**] problem for linearization of the nonlinear system about the current trajectory. The discrete LQR problem is reviewed in Appendix B.2.2.

Proposition 2. *The discrete projection operator defined by* (6.10) *satisfies the idempotent projection property* (6.1)*.*

Proof. Let $(x_1, u_1) = \mathcal{P}(\bar{x}, \bar{u})$, then:

k	$x_1(k)$	$u_1(k)$
0	$\bar{x}(0)$	$\bar{u}(0) + \mathcal{K}(0)(x_1(0) - \bar{x}(0)) = \bar{u}(0)$
1	$f(\bar{x}(0), \bar{u}(0))$	$\bar{u}(1) + \mathcal{K}(1)(x_1(1) - \bar{x}(1))$
2	$f(x_1(1), u_1(1))$	$\bar{u}(2) + \mathcal{K}(2)(x_1(2) - \bar{x}(2))$
3	$f(x_1(2), u_1(2))$	$\bar{u}(3) + \mathcal{K}(3)(x_1(3) - \bar{x}(3))$

$$\vdots$$

Applying the projection again, let $(x_2, u_2) = \mathcal{P}(x_1, u_1)$:

k	$x_2(k)$	$u_2(k)$
0	$x_1(0) = \bar{x}(0)$	$u_1(0) + \mathcal{K}(0)(x_2(0) - x_1(0)) = u_1(0)$
1	$f(\bar{x}(0), \bar{u}(0)) = x_1(1)$	$u_1(1) + \mathcal{K}(1)(x_2(1) - x_1(1)) = u_1(1)$
2	$f(x_1(1), u_1(1)) = x_1(2)$	$u_1(2) + \mathcal{K}(2)(x_2(2) - x_1(2)) = u_1(2)$
3	$f(x_1(2), u_2(2)) = x_1(3)$	$u_1(3) + \mathcal{K}(3)(x_2(3) - x_1(3)) = u_1(3)$

$$\vdots$$

By induction, the trajectories are equal. $\square$

6.2.2.1. First Derivative of $\mathcal{P}(\bar{\xi})$. The derivative of the projection operator $\delta\xi = D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi}$ is found by differentiating (6.10) along a direction $\delta\bar{\xi} = (\delta\bar{x}, \delta\bar{u})$:

$$\delta\xi = (\delta x, \delta u) = D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi} :$$

$$\xi = \mathcal{P}(\bar{\xi}) \tag{6.11a}$$

$$\delta x(k_0) = \delta\bar{x}(k_0) \tag{6.11b}$$

$$\begin{aligned} \delta x(k+1) &= \frac{\partial f}{\partial x}(k)\delta x(k) + \frac{\partial f}{\partial u}(k)\delta u(k) \qquad (6.11c) \\ &= Df(k) \circ \delta\xi(k) \end{aligned}$$

$$\delta u(k) = \delta\bar{u}(k) - \mathcal{K}(k)(\delta x(k) - \delta\bar{x}(k)) \tag{6.11d}$$

where the notation $\frac{\partial f}{\partial x}(k)$ represents $\frac{\partial f}{\partial x}(x(k), u(k), k)$, and also applies to $\frac{\partial f}{\partial u}(k)$ and $Df(k)$.

Like the continuous case, the derivative of the discrete projection operator is a projection itself that maps $\bar{\xi}$ to trajectories of the discrete linear system $\delta x(k+1) = A(k)\delta x(k) + B(k)\delta u(k)$ with $A(k) = \frac{\partial f}{\partial x}(k)$ and $B(k) = \frac{\partial f}{\partial u}(k)$. Therefore, the discrete $D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi}$ satisfies the idempotent projection property (6.2).

The invariance property (6.3) follows directly from (6.11a).

6.2.2.2. Second Derivative of $\mathcal{P}(\bar{\xi})$. The second derivative of the projection operator is found directly from (6.11). The second derivative is considered only with respect to the same perturbation.

$\delta^2\xi = D^2\mathcal{P}(\bar{\xi}) \circ (\delta\bar{\xi}, \delta\bar{\xi})$:

$$\xi = \mathcal{P}(\bar{\xi}) \tag{6.12a}$$

$$\delta\xi = D\mathcal{P}(\bar{\xi}) \circ \delta\bar{\xi} \tag{6.12b}$$

$$\delta^2 x(k_0) = 0 \tag{6.12c}$$

$$\begin{aligned} \delta^2 x(k+1) &= Df(k) \circ \delta^2\xi(k) + D^2 f(k) \circ (\delta\xi(k), \delta\xi(k)) \qquad (6.12\text{d}) \\ &= \frac{\partial f}{\partial x}(k)\delta^2 x(k) + \frac{\partial f}{\partial u}(k)\delta^2 u(k) + D^2 f(k) \circ (\delta\xi(k), \delta\xi(k)) \end{aligned}$$

$$\delta^2 u(k) = -\mathcal{K}(k)\delta^2 x(k) \tag{6.12e}$$

The invariance properties (6.4) and (6.5) follow directly from (6.12a) and (6.12b), respectively.

The second derivative of the discrete projection, while not a projection itself, can be expressed as the dynamics of a discrete linear system as with the continuous projection. Substituting (6.12e) into (6.12d):

$$\delta^2 x(k+1) = \left[\frac{\partial f}{\partial x}(k) - \frac{\partial f}{\partial u}(k)\mathcal{K}(k)\right]\delta^2 x(k) + D^2 f(k) \circ (\delta\xi(k), \delta\xi(k)). \tag{6.13}$$

This is an affine linear system, equivalent to a linear system with an input:

$$\begin{aligned} \delta^2 x(k+1) &= \hat{A}(k)\delta^2 x(k) + \hat{B}(k) \\ \hat{A}(k) &= \frac{\partial f}{\partial x}(k) - \frac{\partial f}{\partial u}(k)\mathcal{K}(k) \\ \hat{B}(k) &= D^2 f(k) \circ (\delta\xi(k), \delta\xi(k)). \end{aligned}$$

The solutions for the linear system can be written in terms of the discrete state transition matrix using (B.3) and the zero initial condition (6.12c):

$$\delta^2 x(k) = \sum_{j=k_0}^{k-1} \Phi(k, j+1)\hat{B}(j). \tag{6.14}$$

This result plays an important role in deriving the second-order Newton's method descent direction in Sec. 6.5.2.

6.2.3. Summary

There are limitations to the projection operators defined here. In general, it is difficult to find globally stabilizing linear feedback laws for non-linear systems. We typically settle for a locally stabilizing controller about a known trajectory ξ_0 and work with curves $\bar{\xi}$ sufficiently close to ξ_0. This is particularly well suited for optimization where each iteration improves a known trajectory by a small perturbation.

The continuous and discrete projection operators have different implementation details, but otherwise they have identical properties and are represented by identical notation. The next section takes advantage of this notation to describe the overall optimization algorithm for both the continuous and discrete domains simultaneously.

6.3. Optimization Algorithm

The trajectory optimization problems in Sec. 6.1 are constrained to solutions that satisfy the system dynamics. The constraints are problematic for an iterative gradient descent algorithm. Even if the current iterate is a valid trajectory $\xi_i \in \mathcal{T}$, and the descent direction is guaranteed to be in the tangent space of ξ_i: $\delta\xi \in T_{\xi_i}\mathcal{T}$, linear combinations of the two $\xi_0 + \delta\xi$ will not, in general, be an admissible trajectory. The projection operator provides a means to remove the constraint, allowing the optimization to take place in the unconstrained inner product space V by defining a new cost $g(\bar{\xi}) = h(\mathcal{P}(\bar{\xi}))$ to optimize.

Proposition 3. *The optimizations* $\min_{\bar{\xi}} g(\bar{\xi})$ *and* $\min_{\xi \in \mathcal{T}} h(\xi)$ *are equivalent in the sense that if* $\bar{\xi}^*$ *minimizes* $g(\cdot)$*, then* $\xi^* = \mathcal{P}(\bar{\xi}^*)$ *minimizes* $h(\cdot)$*, and if* ξ^* *minimizes* $h(\cdot)$*, then* ξ^* *also minimizes* $g(\cdot)$

Proof. Suppose ξ^* minimizes $h(\cdot)$ but not $g(\cdot)$. Then there exists $\delta\bar{\xi} \in V$ such that:

$$Dg(\xi^*) \circ \delta\bar{\xi} \neq 0$$

$$Dh(\xi^*) \circ D\mathcal{P}(\xi^*) \circ \delta\bar{\xi} \neq 0.$$

Letting $\delta\xi = D\mathcal{P}(\xi^*) \circ \delta\bar{\xi} \in T_{\xi^*}\mathcal{T}$:

$$Dh(\xi^*) \circ \delta\xi \neq 0,$$

but this contradicts ξ^* being a minimizer of $h(\cdot)$.

On the other hand, suppose $\bar{\xi}^*$ minimizes $g(\cdot)$ but $\xi = \mathcal{P}(\bar{\xi}^*)$ does not minimize $h(\cdot)$, then there exists $\delta\xi \in T_{\xi^*}\mathcal{T}$ such that:

$$Dh(\xi) \circ \delta\xi \neq 0$$

$$Dh(\mathcal{P}(\bar{\xi}^*)) \circ D\mathcal{P}(\mathcal{P}(\bar{\xi}^*)) \circ \delta\xi \neq 0$$

$$Dg(\bar{\xi}^*) \circ \delta\xi \neq 0,$$

but this contradicts $\bar{\xi}^*$ being a minimizer of $g(\cdot)$. $\square$

This unconstrained optimization can be solved using iterative numeric methods. The overall algorithm is shown in Alg. 1. Each iteration involves designing a projection operator, choosing a descent direction, and performing a search along the descent direction. These steps are described in detail in the following subsections.

Algorithm 1 Trajectory Optimization

```
Require: $\xi_0 \in \mathcal{T}$
Ensure: $\xi_i = \arg\min_{\xi \in \mathcal{T}} h(\xi)$
 1: $i \leftarrow 0$
 2: loop
 3:     $\mathcal{K} \leftarrow CreateProjection(\xi_i)$
 4:     $\delta\xi \leftarrow FindDescentDirection(\xi_i, \mathcal{K})$
 5:     if $|\delta\xi| \approx 0$ then
 6:         return $\xi_i$
 7:     end if
 8:     $\lambda \leftarrow PerformLineSearch(\xi_i, \delta\xi)$
 9:     $\xi_{i+1} \leftarrow \mathcal{P}(\xi_i + \lambda\delta\xi)$
10: end loop
```

6.3.1. Creating the Projection Operator

The continuous and discrete projection operators rely on a stabilizing linear feedback law. In the optimization algorithm, each iteration starts with a known trajectory and searches for new trajectories along a descent direction from there. This local nature allows us to work with projection operators that are only locally stabilizing rather than requiring global stability. A updated stabilizing feedback law is required for each new trajectory.

While any algorithm that provides a stabilizing linear feedback law will work, the LQR optimal control problem is a convenient and robust method. It only requires linearizations of the dynamics, which are already necessary for the trajectory optimization, and it handles arbitrary trajectories without any user intervention. The choice of the cost matrices Q and R in the LQR problem can have a significant impact on performance, but as of yet there is no clear way to choose them. Experience has shown that the choice $Q = \mathcal{I}$, $R = \mathcal{I}$ generally provides acceptable feedback laws.

6.3.2. Finding the Descent Direction

In gradient-descent optimization, the descent direction is calculated by minimizing a quadratic approximation of the cost near each iterate. For finite-dimensional problems, the resulting descent direction is $z = -M^{-1}Dg(x_i)$, where M is the quadratic term of the approximation. Common choices are $M = \mathcal{I}$, which leads to the steepest descent algorithm, and $M = D^2g(x_i)$, which leads to Newton's method[**20**]. This algorithm is simple, but it relies on a matrix representation of the model M.

In trajectory optimization, the vector space is infinite dimensional, so no matrix representation exists. Instead, the approximation of the cost is minimized directly at each iteration. This section develops the quadratic approximation of the cost, and shows that the required minimization is equivalent to a Linear Quadratic (LQ) optimization problem for the linearization of the system about the current trajectory.

The cost is approximated near the current trajectory by the Taylor series:

$$g(\xi_i + \delta\bar{\xi}) \approx g(\xi_i) + Dg(\xi_i) \circ \delta\bar{\xi} + \tfrac{1}{2} D^2 g(\xi_i) \circ (\delta\bar{\xi}, \delta\bar{\xi})$$

$$= h(\mathcal{P}(\xi_i)) + Dh(\mathcal{P}(\xi_i)) \circ D\mathcal{P}(\xi_i) \circ \delta\bar{\xi} + \tfrac{1}{2} Dh(\mathcal{P}(\xi_i)) \circ D^2\mathcal{P}(\xi_i) \circ (\delta\bar{\xi}, \delta\bar{\xi})$$
$$+ \tfrac{1}{2} D^2 h(\mathcal{P}(\xi_i)) \circ (D\mathcal{P}(\xi_i) \circ \delta\bar{\xi}, D\mathcal{P}(\xi_i) \circ \delta\bar{\xi})$$

where ξ_i is the current trajectory and $\bar{\xi}$ is a small perturbation from the trajectory.

The curve ξ_i is guaranteed to be an admissible trajectory (i.e, $\xi_i \in \mathcal{T}$) by the initial condition of the algorithm and the projection step at the end of each iteration (Line 9 in Alg. 1). Applying the identity $\xi = \mathcal{P}(\xi)\ \forall\ \xi \in \mathcal{T}$, and applying the invariance property (6.5) to $D^2\mathcal{P}(\cdot)$, the model becomes

$$= h(\xi_i) + Dh(\xi_i) \circ D\mathcal{P}(\xi_i) \circ \delta\bar{\xi} + \tfrac{1}{2} Dh(\xi_i) \circ D^2\mathcal{P}(\xi_i) \circ (D\mathcal{P}(\xi_i) \circ \delta\bar{\xi}, D\mathcal{P}(\xi_i) \circ \delta\bar{\xi})$$
$$+ \tfrac{1}{2} D^2 h(\xi_i) \circ (D\mathcal{P}(\xi_i) \circ \delta\bar{\xi}, D\mathcal{P}(\xi_i) \circ \delta\bar{\xi})$$

$$= h(\xi_i) + Dh(\xi_i) \circ D\mathcal{P}(\xi_i) \circ \delta\bar{\xi} + \tfrac{1}{2} q(\xi) \circ (D\mathcal{P}(\xi_i) \circ \delta\bar{\xi}, D\mathcal{P}(\xi_i) \circ \delta\bar{\xi}) \tag{6.15}$$

where the second order terms have been collected into the *quadratic model*:

$$q(\xi) \circ (\delta\xi, \delta\xi) = +Dh(\xi_i) \circ D^2\mathcal{P}(\xi_i) \circ (\delta\xi, \delta\xi) + D^2 h(\xi_i) \circ (\delta\xi, \delta\xi). \tag{6.16}$$

The descent direction is found by minimizing (6.15) over all directions in TV:

$$\delta\bar{\xi}^* = \arg\min_{\delta\bar{\xi}} h(\xi_i) + Dh(\xi_i) \circ D\mathcal{P}(\xi_i) \circ \delta\bar{\xi} + \tfrac{1}{2} q(\xi) \circ (D\mathcal{P}(\xi_i) \circ \delta\bar{\xi}, D\mathcal{P}(\xi_i) \circ \delta\bar{\xi}).$$

The earlier approach is now reversed. Noting that $\delta\bar{\xi}$ is always projected into $T_{\xi_i}\mathcal{T}$ by $D\mathcal{P}(\xi_i)$ in this unconstrained optimization, the search is restricted to descent directions in this space:

$$\delta\xi^* = \arg\min_{\delta\xi \in T_{\xi_i}\mathcal{T}} 2Dh(\xi_i) \circ \delta\xi + q(\xi) \circ (\delta\xi, \delta\xi) \tag{6.17}$$

where we have dropped the constant term $h(\xi_i)$ and multiplied the cost by 2 without affecting the optimal direction.

In this case, the constrained optimization problem (6.17) is easier to solve than the unconstrained case because every element $\delta\xi \in T_{\xi_i}\mathcal{T}$ is a trajectory of a (continuous or discrete) linear system. It is a linear optimal control problem with a cost comprising a linear and quadratic term, better known as a Linear Quadratic (LQ) problem.

Solutions to the LQ problem are found by solving a set of (continuous or discrete) Ricatti equations. The solution provides an affine control law $\delta u = -K\delta x + C$ where K and C are (continuous or discrete) time-varying values found from the Ricatti equations. Thus, (6.17) is solved by finding the optimal feedback law from the LQ solution, choosing an initial condition for the descent direction, and integrating the system forward with the feedback input. The trajectory found by the forward simulation is the descent direction that locally minimizes the cost approximation. The choice for the initial conditions of the descent direction depends on whether or not the initial conditions of the non-linear trajectory are to be optimized, discussed in Sec. 6.3.2.1. Details of solving (6.17) for the continuous and discrete cases are discussed in Sec. 6.4.1 and Sec. 6.5.1.

In order for (6.17) to have a well defined minimum, the quadratic model must be positive definite:

$$q(\xi) \circ (\delta\xi, \delta\xi) > 0 \quad \forall\ \delta\xi \in T_\xi\mathcal{T}.$$

The quadratic model (6.16) is not guaranteed to be positive definite. Additionally, even if it is positive definite, it is a well-known result in optimization that descent directions calculated from the second derivative often perform poorly when the current iteration is not near the optimal solution[**20**]. The quadratic model q is therefore generalized, as in finite dimensional optimization, to an arbitrary bilinear model to avoid these problems. The specific form of this bilinear model depends on the time domain. The continuous form is discussed in Sec. 6.4 and the discrete form in Sec. 6.5. Three choices for the bilinear model are discussed in Sec. 6.3.2.2, Sec. 6.3.2.3, and Sec. 6.3.2.4.

The completed algorithm for calculating the descent direction is shown in Alg. 2. The following section discusses the initial conditions for the descent direction.

Algorithm 2 Calculating the Descent Direction

```
Require: $\xi_i \in \mathcal{T}$ and $q(\xi) \circ (\delta\xi, \delta\xi) > 0$
1: $\mathcal{K}, C \leftarrow SolveLQProblem()$
2: $\delta x_0 = ChooseInitialCondition()$
3: $\delta\xi = SimulateLinearizedSystem(dx_0, \delta u = -K\delta x - C)$
4: return $\delta\xi$
```

6.3.2.1. Initial Conditions. The initial conditions for the descent direction determine whether the initial conditions of the original optimization in Sec. 6.1 are fixed or included in the optimization. The material in this section applies to both the continuous and discrete case, so the argument 0 is used to represent t_0 or k_0.

At each iteration, the initial condition of the new trajectory is $x_{i+1}(0) = x_i(0)+\lambda\delta x(0)$, where $\lambda \in (0, 1]$ is the line search parameter discussed in Sec. 6.3.3. If the initial condition

$x(0)$ of the optimization is fixed, then the initial condition $\delta x(0) = 0$ is chosen so that $x_{i+1}(0) = x_i(0) + \lambda 0 = x_i(0)$.

Alternatively, the initial condition may be optimized as well. In this case, we are still guided by the optimization (6.17), but must include the initial condition in addition to the optimal input found by the LQ solution.

The solutions to continuous and discrete LQ problems provide a way to calculate the optimal cost of (6.17):

$$V^*(\delta x(0), 0) = \delta x^T(0)P(0)\delta x(0) + 2b^T(0)\delta x(0) + c(0) \tag{6.18}$$

where $P(0)$, $b(0)$, and $c(0)$ are the solutions to the continuous or discrete Ricatti equation.

Equation (6.18) is minimized with respect to $\delta x(0)$ by setting its derivative to zero and solving for $\delta x(0)$:

$$\delta x(0) = -P^{-1}(0)b(0).$$

This approach has a caveat that it only applies to unconstrained systems. If there are any holonomic or non-holonomic constraints, the new initial condition $x_{i+1}(0) = x_i(0) + \lambda \delta x(0)$ for $\lambda \in (0, 1]$ may be inadmissible.

One approach to circumvent this is to numerically optimize (6.18) using a constrained optimization algorithm. However, since λ is varied in the line search, it would require each iteration of the line search to perform the numeric optimization, and then recalculate the descent direction. This is likely to be computationally prohibitive.

A more sophisticated solution might be to calculate the initial condition without considering the constraint, and then modify the projection operator to project its initial condition into an admissible state. The primary difficulty with this potential approach is that the modified projection operator must still be differentiable.

6.3.2.2. Model for Newton's Method. The original quadratic model (6.16) from the second derivative of the projected cost $g(\bar{\xi})$ leads to a Newton's method descent direction[**15**]. Newton's method is a second-order optimization strategy with quadratic convergence rates near the optimal solution[**20**][**5**]. At trajectories far from the solution, however, Newton steps usually performs poorly and may not even converge. In trajectory optimization, Newton steps are also costly to calculate for complex systems. For these reasons it is standard practice to use a steepest descent or quasi-Newton method first, before switching to Newton's method when the optimization approaches a solution.

The details of calculating the model for Newton's method are presented in Sec. 6.4.2 for continuous time and Sec. 6.5.2 for discrete time.

6.3.2.3. Model for Steepest Descent. Defining $q(\xi) \circ (\delta\xi, \delta\xi) = \langle \delta\xi, \delta\xi \rangle$, the descent direction optimization (6.17) becomes

$$\begin{aligned} \delta\xi^* &= \arg \min_{\delta\xi \in T_{\xi_i}\mathcal{T}} 2Dh(\xi) \circ \delta\xi + \langle \delta\xi, \delta\xi \rangle \\ &= \arg \min_{\delta\xi \in T_{\xi_i}\mathcal{T}} 2Dh(\xi) \circ \delta\xi + ||\delta\xi||^2. \end{aligned}$$

The quadratic term of this cost depends solely on the magnitude of $\bar{\xi}$, so the optimization finds the direction that minimizes the first-order linear term. This is the steepest descent method. It generally has slower convergence rates than Newton's method near the optimum (linear vs. quadratic convergence), but is globally convergent and much less costly to compute.

6.3.2.4. Model for Quasi-Newton Method. A third model is a compromise between the steepest descent and Newton's method models. The quasi-Newton model is derived from the derivative of the cost (6.16), but the second derivative of the dynamics is removed:

$$q(\xi) \circ (\delta\xi, \delta\xi) = D^2 h(\xi_i) \circ (\delta\xi, \delta\xi). \tag{6.19}$$

Unlike the complete second derivative, this model can be guaranteed to be positive definite if an appropriate cost function $h(\cdot)$ is used. The example optimization in Sec. 6.1 has this property.

This model was discovered experimentally and often outperforms the steepest descent algorithm. It is as inexpensive to calculate but often converges quadratically. Several optimizations using this method are presented with the examples in Sec. 6.7.

6.3.3. Performing the Line Search

The line search is required in iterative optimization to guarantee a sufficient decrease in the cost of each new iterate[**19**]. In trajectory optimization, the line search plays an additional role by reducing the size of the descent step until the new trajectory is in the stabilizable subspace of the current projection operator. Otherwise, the line search is identical to the finite dimensional case, and any number of existing line search algorithms can be used.

A modified Armijo line search[**4**] is shown in Alg. 3. Note the additional check on line 5 to guarantee that the perturbed trajectory is successfully stabilized by the projection operator.

Algorithm 3 Armijo Line Search

```
Require: $\xi \in \mathcal{T}$, $\delta\xi \in T_\xi\mathcal{T}$, $\alpha \in [0,1]$ and $\beta \in (0,1)$
Ensure: $g(\xi + \lambda\delta\xi) < h(\xi) + \alpha\lambda Dh(\xi) \circ \delta\xi$
 1: $m \leftarrow 0$
 2: loop
 3:     $\lambda = \beta^m$
 4:     $\xi_n = \mathcal{P}(\xi + \lambda\delta\xi)$
 5:     if the projection succeeded then
 6:         if $g(\xi + \lambda\delta\xi) < h(\xi) + \alpha\lambda Dh(\xi) \circ \delta\xi$ then
 7:             return $\lambda$
 8:         end if
 9:     end if
10:     $m \leftarrow m + 1$
11: end loop
```

6.3.4. Summary

The previous discussion outlines the overall optimization algorithm for both the continuous and discrete time domains. The descent direction is found by solving a linear optimal control problem at each iteration in both cases. The details of calculating the cost approximation and solving the LQ problem are discussed in the following sections (Sec. 6.4 for continuous time and Sec. 6.5 for discrete time).

6.4. Continuous Optimization

6.4.1. Descent Direction

The generalized quadratic model for the continuous time domain is:

$$q(\xi) \circ (\delta\xi, \delta\xi) = \int_{t_0}^{t_f} \begin{bmatrix} \delta x(t) \\ \delta u(t) \end{bmatrix}^T \begin{bmatrix} Q(t) & S(t) \\ S^T(t) & R(t) \end{bmatrix} \begin{bmatrix} \delta x(t) \\ \delta u(t) \end{bmatrix} \mathrm{d}t + \delta x^T(t_f) Q_f \delta x(t_f) \tag{6.20}$$

The descent direction optimization (6.17) is expanded with the quadratic model and the continuous time equation for $Dh(\xi_i) \circ \delta\xi$:

$$\begin{aligned} \delta\xi^* &= \arg \min_{\delta\xi \in T_{\xi_i}\mathcal{T}} 2Dh(\xi) \circ \delta\xi + q(\xi) \circ (\delta\xi, \delta\xi) \\ &= \arg \min_{\delta\xi \in T_{\xi_i}\mathcal{T}} 2 \int_{t_0}^{t_f} \begin{bmatrix} \frac{\partial \ell}{\partial x}(t) & \frac{\partial \ell}{\partial u}(t) \end{bmatrix} \begin{bmatrix} \delta x(t) \\ \delta u(t) \end{bmatrix} + \begin{bmatrix} \delta x(t) \\ \delta u(t) \end{bmatrix}^T \begin{bmatrix} Q(t) & S(t) \\ S^T(t) & R(t) \end{bmatrix} \begin{bmatrix} \delta x(t) \\ \delta u(t) \end{bmatrix} \mathrm{d}t \end{aligned} \tag{6.21}$$

$$+ 2\frac{\partial m}{\partial x} \delta x(t_f) + \delta x^T(t_f) Q_f \delta x(t_f) \tag{6.22}$$

This is equivalent to the continuous time LQ problem for the linear system defined by $A(t) = \frac{\partial f}{\partial x}(t)$ and $B(t) = \frac{\partial f}{\partial u}(t)$. The LQ problem and solution are provided in Sec. B.1.3 for reference. The LQ problem is solved by integrating three Ricatti equations backwards in time to find an affine feedback law $\delta u(t) = -K(t)\delta u(t) + C(t)$ that minimizes Equation 6.22. An initial condition for $\delta x(t_0)$ is chosen as described in Sec. 6.3.2.1 and the system is simulated forward in time with the optimal feedback law to numerically calculate the descent direction $\delta\xi$.

The LQ problem cannot be solved until a specific model is chosen for the quadratic model $q(\xi)$. Section 6.4.2 describes the model for Newton's method. A model for a steepest descent step is given in Sec. 6.4.3, and a quasi-Newton's method is described in Sec. 6.4.4.

6.4.2. Newton's Method Model

For Newton's method in continuous time, the model function is the second derivative of the actual cost function as defined in (6.16), restated here:

$$q(\xi) \circ (\delta\xi, \delta\xi) = D^2 h(\xi) \circ (\delta\xi, \delta\xi) + Dh(\xi) \circ D^2\mathcal{P}(\xi) \circ (\delta\xi, \delta\xi) \tag{6.23}$$

where the model function $q(\xi)$ has the form defined in (6.20).

This model is derived in two parts. The values for the terms in $q(\xi)$ are given by Prop. 4. These are written in terms of an adjoint trajectory $z(t)$, defined in the proposition, that is calculated separately. Sec. 6.4.2.1 continues with a derivation to calculate the adjoint trajectory efficiently by solving a backwards difference equation.

Proposition 4. *The model function that satisfies* (6.23) *is*

$$\begin{aligned}
Q(t_f) &= D^2 m(x(t_f)) \\
Q(t) &= \frac{\partial^2 \ell}{\partial x \partial x}(t) + z^T(t) \frac{\partial^2 f}{\partial x \partial x}(t) \\
S(t) &= \frac{\partial^2 \ell}{\partial x \partial u}(t) + z^T(t) \frac{\partial^2 f}{\partial x \partial u}(t) \\
R(t) &= \frac{\partial^2 \ell}{\partial u \partial u}(t) + z^T(t) \frac{\partial^2 f}{\partial u \partial u}(t)
\end{aligned}$$

where

$$z^T(s) = \int_s^{t_f} \left[\frac{\partial \ell}{\partial x}(t) - \frac{\partial \ell}{\partial u}(t) \circ \mathcal{K}(t)\right] \circ \Phi(t,s)\mathrm{d}t + \frac{\partial m}{\partial x} \circ \Phi(t_f, s)$$

and $\Phi(t,s)$ *is the state transition matrix for the linear system* $A(t) = \frac{\partial f}{\partial x}(t)$*.*

Proof. The second derivative of the cost $h(\xi)$ is:

$$D^2 h(\xi) \circ (\delta\xi, \delta\xi) = \int_{t_0}^{t_f} D^2 \ell(t) \circ (\delta\xi(t), \delta\xi(t))\mathrm{d}t + D^2 m(t_f) \circ (\delta x(t_f), \delta x(t_f))\,. \tag{6.24}$$

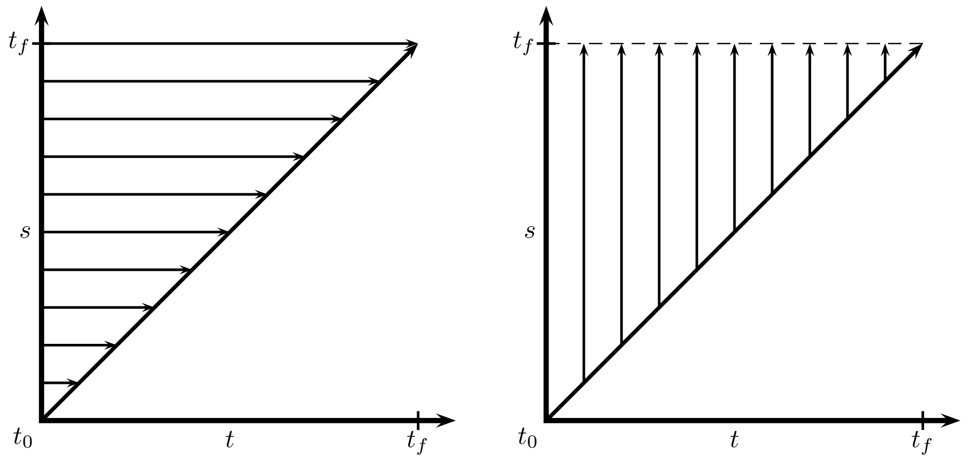
This corresponds to the form of the continuous quadratic model (6.20), so we continue to the second term of (6.23). Letting $(\delta^2 x, \delta^2 u) = D^2\mathcal{P}(\xi) \circ (\delta\xi, \delta\xi)$, we expand $Dh(\xi)$ and apply (6.8e).

$$\begin{aligned} Dh(\xi) \circ D^2\mathcal{P}(\xi) \circ (\delta\xi, \delta\xi) &= \int_{t_0}^{t_f} \left[\frac{\partial \ell}{\partial x}(t) \circ \delta^2 x(t) + \frac{\partial \ell}{\partial u}(t) \circ \delta^2 u(t)\right] \mathrm{d}t + \frac{\partial m}{\partial x}(t_f) \circ \delta^2 x(t_f) \\ &= \int_{t_0}^{t_f} \left[\frac{\partial \ell}{\partial x}(t) - \frac{\partial \ell}{\partial u}(t) \circ \mathcal{K}(t)\right] \circ \delta^2 x(t)\mathrm{d}t + \frac{\partial m}{\partial x}(t_f) \circ \delta^2 x(t_f) \end{aligned}$$

Section 6.2.1.2 derived a closed form solution (6.9) for $\delta^2 x(t)$ as the trajectory of a linear system. Substituting this definition, the following is obtained:

$$= \int_{t_0}^{t_f} \int_{t_0}^{t} \left[\frac{\partial \ell}{\partial x}(t) - \frac{\partial \ell}{\partial u}(t) \circ \mathcal{K}(t)\right] \circ \Phi(t,s) \circ \hat{B}(s)\mathrm{d}s\mathrm{d}t + \int_{t_0}^{t_f} \frac{\partial m}{\partial x}(t_f) \circ \Phi(t_f, s) \circ \hat{B}(s)\mathrm{d}s. \tag{6.25}$$

The two terms in (6.25) are combined by switching the order of integration in the first term. The original integration is depicted in Fig. 6.1a. Each horizontal line depicts an integration over s for a single value of t. Fig. 6.1b illustrates the switched order that preserves the integration area.



(a) Original Integration Order (b) Modified Integration Order

Figure 6.1. Switching the order of integration in (6.25).

The order of integration in (6.25) is switched and simplified:

$$
\begin{aligned}
&= \int_{t_0}^{t_f} \int_{s}^{t_f} \left[\frac{\partial \ell}{\partial x}(t) - \frac{\partial \ell}{\partial u}(t) \circ \mathcal{K}(t)\right] \circ \Phi(t,s) \circ \hat{B}(s)\mathrm{d}t\mathrm{d}s + \int_{t_0}^{t_f} \frac{\partial m}{\partial x}(t_f) \circ \Phi(t_f,s) \circ \hat{B}(s)\mathrm{d}s \\
&= \int_{t_0}^{t_f} \left[\int_{s}^{t_f} \left[\frac{\partial \ell}{\partial x}(t) - \frac{\partial \ell}{\partial u}(t) \circ \mathcal{K}(t)\right] \circ \Phi(t,s)\mathrm{d}t + \frac{\partial m}{\partial x}(t_f) \circ \Phi(t_f,s)\right] \circ \hat{B}(s)\mathrm{d}s \\
&= \int_{t_0}^{t_f} z^T(s) \circ \hat{B}(s)\mathrm{d}s = \int_{t_0}^{t_f} z^T(s) \circ D^2 f(s) \circ (\delta\xi(s), \delta\xi(s))\mathrm{d}s
\end{aligned} \tag{6.26}
$$

where

$$
z^T(s) = \int_{s}^{t_f} \left[\frac{\partial \ell}{\partial x}(t) - \frac{\partial \ell}{\partial u}(t) \circ \mathcal{K}(t)\right] \circ \Phi(t,s)\mathrm{d}t + \frac{\partial m}{\partial x}(t_f) \circ \Phi(t_f,s). \tag{6.27}
$$

We assume that $z(\cdot)$ can be calculated and return to the original problem of finding the model function $q(\xi) \circ (\delta\xi, \delta\xi)$. The derivation of $z(\cdot)$ continues in Sec. 6.4.2.1.

Expanding (6.23) with $D^2h(\xi)$ from (6.24) and $Dh(\xi) \circ D^2\mathcal{P}(\xi) \circ (\delta\xi, \delta\xi)$ from (6.26), we have:

$$q(\xi) \circ (\delta\xi, \delta\xi) = \int_{t_0}^{t_f} D^2\ell(t) \circ (\delta\xi(t), \delta\xi(t))\mathrm{d}t + D^2m(t_f) \circ (\delta x(t_f), \delta x(t_f)) + \int_{t_0}^{t_f} z^T(t) \circ D^2f(t) \circ (\delta\xi(t), \delta\xi(t))\mathrm{d}t$$

$$q(\xi) \circ (\delta\xi, \delta\xi) = \int_{t_0}^{t_f} D^2\ell(t) \circ (\delta\xi(t), \delta\xi(t)) + z^T(t) \circ D^2f(t) \circ (\delta\xi(t), \delta\xi(t))\mathrm{d}t + D^2m(t_f) \circ (\delta x(t_f), \delta x(t_f))\,. \quad (6.28)$$

The components of the $q(\xi)$ are found by comparing (6.28) with the definition of $q(\xi)$ in (6.20):

$$\begin{aligned} Q(t_f) &= D^2m(x(t_f)) \\ Q(t) &= \frac{\partial^2\ell}{\partial x\partial x}(t) + z^T(t)\frac{\partial^2 f}{\partial x\partial x}(t) \\ S(t) &= \frac{\partial^2\ell}{\partial x\partial u}(t) + z^T(t)\frac{\partial^2 f}{\partial x\partial u}(t) \\ R(t) &= \frac{\partial^2\ell}{\partial u\partial u}(t) + z^T(t)\frac{\partial^2 f}{\partial u\partial u}(t). \end{aligned} \qquad \square$$

This model is the exact second derivative of the projected cost function. Equation (6.27) provides a concrete but inefficient means to calculate $z(t)$. The following subsection continues the derivation with an improved method to compute $z(t)$ as a backwards difference equation.

6.4.2.1. Calculating $z(t)$. Equation (6.27) provides a concrete but inefficient means to calculate $z(t)$ because it requires $\Phi(t, s)$ for every combination[2] of t and s. Proposition 5 provides an alternative way to calculate $z(k)$ by solving a backwards difference equation that does not require the state transition matrix at all. Furthermore, the model $q(\xi)$ is evaluated by the LQ solution, which is also solved by a backwards differential equation. This allows $z(t)$ to be calculated simultaneously and used without being stored, reducing memory requirements of the optimization.

Proposition 5. *The continuous adjoint trajectory $z(t)$ is the solution to a backwards differential equation:*

$$\begin{aligned} z^T(t_f) &= Dm(x(t_f)) \\ -\frac{\partial}{\partial t} z^T(t) &= z^T(t)\left[\frac{\partial f}{\partial x}(t) - \frac{\partial f}{\partial u}(t)\mathcal{K}(t)\right] + \frac{\partial \ell}{\partial x}(t) - \frac{\partial \ell}{\partial u}(t)\mathcal{K}(t). \end{aligned}$$

Proof. The backwards differential equation is derived by differentiating (6.27) with respect to $s = t$ and rewriting the resulting differential equation in terms of $z(t)$.

$$\frac{\partial z^T}{\partial t}(t) = \frac{\partial}{\partial t}\left[\int_t^{t_f}\left[\frac{\partial \ell}{\partial x}(s) - \frac{\partial \ell}{\partial u}(s)\circ\mathcal{K}(s)\right]\circ\Phi(s,t)\mathrm{d}s + \frac{\partial m}{\partial x}(t_f)\circ\Phi(t_f,t)\right]$$

$$\begin{aligned} = \left[\int_t^{t_f} - \left[\frac{\partial \ell}{\partial x}(s) - \frac{\partial \ell}{\partial u}(s)\circ\mathcal{K}(s)\right]\circ\Phi(s,t)\hat{A}(t)\mathrm{d}s\right] \\ - \left[\frac{\partial \ell}{\partial x}(t) - \frac{\partial \ell}{\partial u}(t)\circ\mathcal{K}(t)\right]\circ\Phi(t,t) - \frac{\partial m}{\partial x}(t_f)\circ\Phi(t_f,t)\hat{A}(t) \end{aligned}$$

[2]This can be done by numerically calculating $\Phi(t_0, s)\ \forall\ s \in [t_0, t_f]$ and using the identity $\Phi(t, s) = \Phi(t_0, t)^{-1}\Phi(t_0, s)$.

$$
\begin{aligned}
&= -\left[\int_{t}^{t_f} \left[\frac{\partial \ell}{\partial x}(s) - \frac{\partial \ell}{\partial u}(s) \circ \mathcal{K}(s)\right] \circ \Phi(s,t)\mathrm{d}s + \frac{\partial m}{\partial x}(t_f) \circ \Phi(t_f,t)\right] \hat{A}(t) - \frac{\partial \ell}{\partial x}(t) + \frac{\partial \ell}{\partial u}(t)\mathcal{K}(t) \\
&= -z^T(t)\hat{A}(t) - \frac{\partial \ell}{\partial x}(t) + \frac{\partial \ell}{\partial u}(t) \circ \mathcal{K}(t) \\
&= -z^T(t)\left[\frac{\partial f}{\partial x}(t) - \frac{\partial f}{\partial u}(t)\mathcal{K}(t)\right] - \frac{\partial \ell}{\partial x}(t) + \frac{\partial \ell}{\partial u}(t)\mathcal{K}(t)
\end{aligned}
$$

Finally, we find a boundary condition for $z^T(t)$ by evaluating (6.27) with $s = t_f$:

$$
\begin{aligned}
z^T(t_f) &= \int_{t_f}^{t_f} \left[\frac{\partial \ell}{\partial x}(t) - \frac{\partial \ell}{\partial u}(t) \circ \mathcal{K}(t)\right] \circ \Phi(t,t_f)\mathrm{d}t + \frac{\partial m}{\partial x}(t_f) \circ \Phi(t_f,t_f) \\
&= \frac{\partial m}{\partial x}(t_f) = Dm(x(t_f)).
\end{aligned}
$$

□

6.4.3. Steepest Descent Model

As discussed in Sec. 6.3.2.3, the steepest descent algorithm uses the model $q(\xi)\circ(\delta\xi,\delta\xi) = \langle\delta\xi,\delta\xi\rangle$. In continuous time, the inner product of V is:

$$
q(\xi) \circ (\delta\xi,\delta\xi) = \int_{t_0}^{t_f} \left[\delta x^T(t)\delta x(t) + \delta u^T(t)\delta u(t)\right] \mathrm{d}t + \delta x^T(t)\delta x(t).
$$

This is equivalent to the model (6.20) with

$$
\begin{aligned}
Q(t) &= \mathcal{I} \\
R(t) &= \mathcal{I} \\
S(t) &= 0 \\
Q_f &= \mathcal{I}.
\end{aligned}
$$

6.4.4. Quasi-Newton Model

The quasi-Newton step from Sec. 6.3.2.4 uses the model $q(\xi)\circ(\delta\xi,\delta\xi) = D^2h(\xi_i)\circ(\delta\xi,\delta\xi)$. The derivative of the cost in continuous time is:

$$D^2h(\xi)\circ(\delta\xi,\delta\xi) = \int_{t_0}^{t_f} D^2\ell(t)\circ(\delta\xi(t),\delta\xi(t))\mathrm{d}t + D^2m(t_f)\circ(\delta x(t_f),\delta x(t_f))\,.$$

This is equivalent to the model (6.20) with

$$\begin{aligned} Q(t_f) &= D^2m(x(t_f)) \\ Q(t) &= \frac{\partial^2\ell}{\partial x\partial x}(t) \\ S(t) &= \frac{\partial^2\ell}{\partial x\partial u}(t) \\ R(t) &= \frac{\partial^2\ell}{\partial u\partial u}(t). \end{aligned}$$

6.5. Discrete Optimization

The details of the discrete optimization correspond almost exactly to the continuous optimization in Sec. 6.4.

6.5.1. Descent Direction

The generalized quadratic term in discrete time is:

$$q(\xi)\circ(\delta\xi,\delta\xi) = \left(\sum_{k=k_0}^{k_f-1}\begin{bmatrix}\delta x(k)\\ \delta u(k)\end{bmatrix}^T\begin{bmatrix}Q(k) & S(k)\\ S^T(k) & R(k)\end{bmatrix}\begin{bmatrix}\delta x(k)\\ \delta u(k)\end{bmatrix}\right) + \delta x^T(k_f)Q(k_f)\delta x(k_f). \quad (6.29)$$

Expanding (6.17) for the discrete case:

$$\delta\xi^* = \arg\min_{\delta\xi} 2Dh(\xi) \circ \delta\xi + q(\xi) \circ (\delta\xi, \delta\xi)$$

$$= \arg\min_{\delta\xi} \sum_{k=k_0}^{k_f-1} \left[2 \begin{bmatrix} \frac{\partial \ell}{\partial x}(k) & \frac{\partial \ell}{\partial u}(k) \end{bmatrix} \begin{bmatrix} \delta x(k) \\ \delta u(k) \end{bmatrix} + \begin{bmatrix} \delta x(k) \\ \delta u(k) \end{bmatrix}^T \begin{bmatrix} Q(k) & S(k) \\ S^T(k) & R(k) \end{bmatrix} \begin{bmatrix} \delta x(k) \\ \delta u(k) \end{bmatrix} \right]$$
$$+ 2Dm(x(k_f))\delta x(k_0) + \delta x^T(k_f) Q(k_f) \delta x(k_f). \quad (6.30)$$

This is the discrete LQ problem. The LQ problem definition and solution are provided in Sec. B.2.3. The LQ problem is solved by three discrete Ricatti equations for $P(k)$, $b(k)$, and $c(k)$ that calculate the optimal input to minimize this cost. The optimal input is an affine feedback law of the form $\delta u(k) = -\mathcal{K}(k)\delta x(k) - C(k)$. Once the optimal input is found, the linear system is simulated forward in time using the input, and the simulated trajectory becomes the numeric descent direction $\delta\xi$.

6.5.2. Newton's Method Model

Newton's method uses the second derivative of the cost for the quadratic term:

$$q(\xi) \circ (\delta\xi, \delta\xi) = D^2 h(\xi) \circ (\delta\xi, \delta\xi) + Dh(\xi) \circ D^2\mathcal{P}(\xi) \circ (\delta\xi, \delta\xi) \quad (6.31)$$

where the form of the model function $q(\xi)$ was defined in (6.29).

This model is derived in two parts. The values for the terms in $q(\xi)$ are given by Prop. 6 in terms of an adjoint trajectory $z(k)$ that must be calculated, which is defined in the proposition. Sec. 6.5.2.1 continues the derivation by providing a way to calculate the adjoint trajectory efficiently by solving a backwards difference equation.

Proposition 6. *The model function with*

$$\begin{aligned}
Q(k_f) &= D^2 m(x(k_f)) \\
Q(k) &= \frac{\partial^2 \ell}{\partial x \partial x}(k) + z^T(k+1)\frac{\partial^2 f}{\partial x \partial x}(k) \\
S(k) &= \frac{\partial^2 \ell}{\partial x \partial u}(k) + z^T(k+1)\frac{\partial^2 f}{\partial x \partial u}(k) \\
R(k) &= \frac{\partial^2 \ell}{\partial u \partial u}(k) + z^T(k+1)\frac{\partial^2 f}{\partial u \partial u}(k)
\end{aligned}$$

where

$$z^T(j) = \sum_{k=j}^{k_f-1} \left[\frac{\partial \ell}{\partial x}(k) - \frac{\partial \ell}{\partial u}(k)\mathcal{K}(k)\right] \circ \Phi(k,j) + Dm(x(k_f)) \circ \Phi(k_f, j).$$

satisfies (6.31)*.*

Proof. The second derivative of the cost $h(\xi)$ is:

$$D^2 h(\xi) \circ (\delta\xi, \delta\xi) = \sum_{k=k_0}^{k_f-1} D^2 \ell(k) \circ (\delta\xi(k), \delta\xi(k)) + D^2 m(k_f) \circ (\delta x(k_f), \delta x(k_f))\,. \quad (6.32)$$

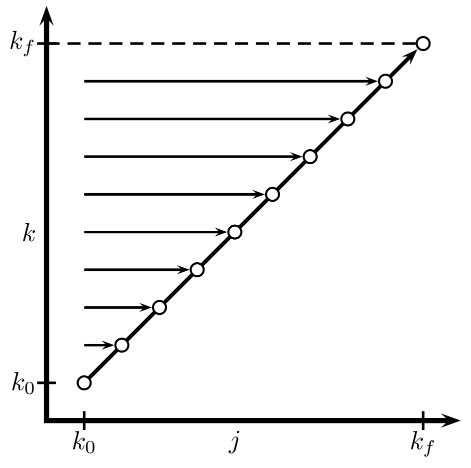
This corresponds to the form of (6.29), so we focus on the second term of (6.31). Letting $(\delta^2 x, \delta^2 u) = D^2\mathcal{P}(\xi) \circ (\delta\xi, \delta\xi)$, we expand $Dh(\xi)$ and apply (6.12e).

$$\begin{aligned} Dh(\xi)\circ & D^2\mathcal{P}(\xi) \circ (\delta\xi, \delta\xi) \\ &= \sum_{k=k_0}^{k_f-1} \left[\frac{\partial \ell}{\partial x}(k) \circ \delta^2 x(k) + \frac{\partial \ell}{\partial u}(k) \circ \delta^2 u(k)\right] + Dm(x(k_f)) \circ \delta^2 x(k_f) \\ &= \sum_{k=k_0}^{k_f-1} \left[\frac{\partial \ell}{\partial x}(k) - \frac{\partial \ell}{\partial u}(k)\mathcal{K}(k)\right] \circ \delta^2 x(k) + Dm(x(k_f)) \circ \delta^2 x(k_f) \end{aligned}$$

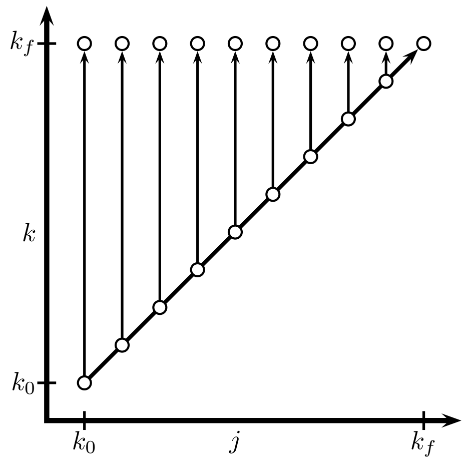
In Sec. 6.2.2.2, a closed-form solution for $\delta^2 x$ is derived. Substituting this result (6.14) into this equation:

$$\begin{aligned} = \left(\sum_{k=k_0}^{k_f-1} \sum_{j=k_0}^{k-1} \left[\frac{\partial \ell}{\partial x}(k) - \frac{\partial \ell}{\partial u}(k)\mathcal{K}(k)\right] \circ \Phi(k, j+1) \circ B(j)\right) & \\ + \left[\sum_{i=k_0}^{k_f-1} Dm(x(k_f)) \circ \Phi(k_f, j+1) \circ B(j)\right]. & \qquad (6.33) \end{aligned}$$

The two terms in (6.33) can be combined by switching the order of summation in the first term. The original summation is depicted in Fig. 6.2a. Each horizontal line depicts iterations over j for a single value of k. Fig. 6.2b illustrates the switched order and shows that the same combinations of k and j are included in the summation.



(a) Original Summation Order



(b) Switched Summation Order

Figure 6.2. Switching the order of summation in (6.33). Empty circles and dotted line indicate combinations of j and k that are not included in the summation.

We apply the switched order of summation and simplify (6.33):

$$Dh(\xi) \circ D^2\mathcal{P}(\xi) \circ (\delta\xi, \delta\xi) = \left(\sum_{j=k_0}^{k_f-1} \sum_{k=j+1}^{k_f-1} \left[\frac{\partial \ell}{\partial x}(k) - \frac{\partial \ell}{\partial u}(k)\mathcal{K}(k) \right] \circ \Phi(k, j+1) \circ B(i) \right) + \left[\sum_{j=k_0}^{k_f-1} Dm(x(k_f)) \circ \Phi(k_f, j+1) \circ B(i) \right]$$

$$= \sum_{j=k_0}^{k_f-1} \left(\sum_{k=j+1}^{k_f-1} \left[\frac{\partial \ell}{\partial x}(k) - \frac{\partial \ell}{\partial u}(k)\mathcal{K}(k) \right] \circ \Phi(k, j+1) + Dm(x(k_f)) \circ \Phi(k_f, j+1) \right) \circ B(j)$$

$$= \sum_{j=k_0}^{k_f-1} z^T(j+1) \circ B(j) = \sum_{j=k_0}^{k_f-1} z^T(j+1) \circ D^2 f(j) \circ (\delta\xi(j), \delta\xi(j)) \tag{6.34}$$

where

$$z^T(j) = \sum_{k=j}^{k_f-1} \left[\frac{\partial \ell}{\partial x}(k) - \frac{\partial \ell}{\partial u}(k)\mathcal{K}(k) \right] \circ \Phi(k,j) + Dm(x(k_f)) \circ \Phi(k_f, j). \tag{6.35}$$

We assume that $z(\cdot)$ can be calculated and return to the original problem of finding the model function $q(\xi) \circ (\delta\xi, \delta\xi)$. We further discuss $z(\cdot)$ in Sec. 6.5.2.1.

Expanding (6.31) with $D^2 h(\xi)$ from (6.32) and $Dh(\xi) \circ D^2 \mathcal{P}(\xi) \circ (\delta\xi, \delta\xi)$ from (6.34), the following is found:

$$q(\xi) \circ (\delta\xi, \delta\xi) = \sum_{k=k_0}^{k_f-1} D^2 \ell(k) \circ (\delta\xi, \delta\xi) + D^2 m(k) \circ (\delta x(k_f), \delta x(k_f)) \\ + \sum_{k=k_0}^{k_f-1} z^T(k+1) D^2 f(k) \circ (\delta\xi(k), \delta\xi(k))$$

$$q(\xi) \circ (\delta\xi, \delta\xi) = \sum_{k=k_0}^{k_f-1} \left[D^2 \ell(k) + z^T(k+1) D^2 f(k) \right] \circ (\delta\xi(k), \delta\xi(k)) \\ + D^2 m(k) \circ (\delta x(k_f), \delta x(k_f)). \tag{6.36}$$

Matching the components of (6.36) with the definition of $q(\xi)$ in (6.29) results in:

$$\begin{aligned}
Q(k_f) &= D^2 m(x(k_f)) \\
Q(k) &= \frac{\partial^2 \ell}{\partial x \partial x}(k) + z^T(k+1)\frac{\partial^2 f}{\partial x \partial x}(k) \\
S(k) &= \frac{\partial^2 \ell}{\partial x \partial u}(k) + z^T(k+1)\frac{\partial^2 f}{\partial x \partial u}(k) \\
R(k) &= \frac{\partial^2 \ell}{\partial u \partial u}(k) + z^T(k+1)\frac{\partial^2 f}{\partial u \partial u}(k). \qquad \square
\end{aligned}$$

This model is the exact second derivative of the projected cost function. The following subsection continues the derivation with an improved method to compute $z(k)$ as a backwards difference equation.

6.5.2.1. Calculating $z(k)$. Equation (6.35) provides a concrete but inefficient means to calculate $z(k)$ because it requires $\Phi(k, i)$ for every combination of k and i. Proposition 7 provides an alternative way to calculate $z(k)$ by solving a backwards difference equation that does not require the state transition matrix at all. Furthermore, since the model $q(\xi)$ is evaluated by the LQ solution which is also solved by a backwards difference equation, $z(k)$ can be calculated in tandem with the LQ solution and does not need to be stored, reducing memory requirements of the optimization.

Proposition 7. *The adjoint trajectory $z(k)$ is the solution to a backwards difference equation:*

$$\begin{aligned}
z^T(k_f) &= Dm(x(k_f)) \\
z^T(k) &= \left[\frac{\partial \ell}{\partial x}(k) - \frac{\partial \ell}{\partial u}(k)\mathcal{K}(k)\right] + z^T(k+1)\left[\frac{\partial f}{\partial x}(k) - \frac{\partial f}{\partial u}(k)\mathcal{K}(k)\right]
\end{aligned}$$

Proof. The backwards difference equation is derived by extracting the first element of the summation in (6.35) and rewriting the equation in terms $z(k+1)$:

$$z^T(j) = \left[\frac{\partial \ell}{\partial x}(j) - \frac{\partial \ell}{\partial u}(j)\mathcal{K}(j)\right] \circ \Phi(j,j) \\ + \sum_{k=i+1}^{k_f-1} \left[\frac{\partial \ell}{\partial x}(j) - \frac{\partial \ell}{\partial u}(j)\mathcal{K}(j)\right] \circ \Phi(k,j) + Dm(x(k_f)) \circ \Phi(k_f,j).$$

Applying the linear system identities (B.2a) and (B.2c):

$$= \left[\frac{\partial \ell}{\partial x}(j) - \frac{\partial \ell}{\partial u}(j)\mathcal{K}(j)\right] + \sum_{k=i+1}^{k_f-1} \left[\frac{\partial \ell}{\partial x}(k) - \frac{\partial \ell}{\partial u}(k)\mathcal{K}(k)\right] \circ \Phi(k,j+1)A(j) \\ + Dm(x(k_f)) \circ \Phi(k_f,j+1)A(j)$$

$$= \left[\frac{\partial \ell}{\partial x}(j) - \frac{\partial \ell}{\partial u}(j)\mathcal{K}(j)\right] + \left(\sum_{k=j+1}^{k_f-1} \left[\frac{\partial \ell}{\partial x}(k) - \frac{\partial \ell}{\partial u}(k)\mathcal{K}(k)\right] \circ \Phi(k,j+1) \\ + Dm(x(k_f)) \circ \Phi(k_f,j+1)\right)A(j)$$

The term in parentheses is the definition of $z(j+1)$:

$$z(j) = \left[\frac{\partial \ell}{\partial x}(j) - \frac{\partial \ell}{\partial u}(j)\mathcal{K}(j)\right] + z^T(j+1)A(j)$$
$$z(j) = \left[\frac{\partial \ell}{\partial x}(j) - \frac{\partial \ell}{\partial u}(j)\mathcal{K}(j)\right] + z^T(j+1)\left[\frac{\partial f}{\partial x}(j) - \frac{\partial f}{\partial u}(j)\mathcal{K}(j)\right].$$

This is the backwards difference equation in the proposition.

Finally, the initial condition is found by evaluating the original definition (6.35) at $i = k_f$:

$$\begin{aligned} z^T(k_f) &= \sum_{k=k_f}^{k_f-1} \left[\frac{\partial \ell}{\partial x}(k) - \frac{\partial \ell}{\partial u}(k)\mathcal{K}(k) \right] \circ \Phi(k, k_f) + Dm(x(k_f)) \circ \Phi(k_f, k_f) \\ &= Dm(x(k_f)). \end{aligned} \qquad \square$$

6.5.3. Steepest Descent Model

As discussed in Sec. 6.3.2.3, the steepest descent algorithm uses the model $q(\xi) \circ (\delta\xi, \delta\xi) = \langle \delta\xi, \delta\xi \rangle$. In discrete time, the inner product of V is:

$$q(\xi) \circ (\delta\xi, \delta\xi) = \sum_{k=k_0+1}^{k_f} \left[x^T(k)x(k) + u^T(k)u(k) \right] + x^T(k_f)x(k_f)$$

This is equivalent to the model (6.29) with

$$\begin{aligned} Q(k_f) &= \mathcal{I} \\ Q(k) &= \mathcal{I} \\ S(k) &= 0 \\ R(k) &= \mathcal{I}. \end{aligned}$$

6.5.4. Quasi-Newton Model

The quasi-Newton step from Sec. 6.3.2.4 uses the model $q(\xi) \circ (\delta\xi, \delta\xi) = D^2 h(\xi_i) \circ (\delta\xi, \delta\xi)$. The derivative of the cost in discrete time is:

$$D^2 h(\xi) \circ (\delta\xi, \delta\xi) = \sum_{k=k_0}^{k_f - 1} D^2 \ell(k) \circ (\delta\xi(k), \delta\xi(k)) + D^2 m(k_f) \circ (\delta x(k_f), \delta x(k_f)) \,.$$

This is equivalent to the model (6.29) with

$$\begin{aligned} Q(k_f) &= D^2 m(x(k_f)) \\ Q(k) &= \frac{\partial^2 \ell}{\partial x \partial x}(k) \\ S(k) &= \frac{\partial^2 \ell}{\partial x \partial u}(k) \\ R(k) &= \frac{\partial^2 \ell}{\partial u \partial u}(k). \end{aligned}$$

6.6. Generating Initial Trajectories

Gradient descent optimizations terminate in local minimums of the cost. The initial condition for the optimization largely influences the resulting solution. Thus it is imperative to start the optimization from a point (or in this case, a trajectory) that is as close to the desired solution as possible. This section describes two practical approaches to generating initial trajectories. In both cases, we consider optimizations where the goal is to track a desired trajectory[3] $\xi_d = (x_d, u_d)$ as in the example cost function in Sec. 6.1.

[3]The desired trajectory may not be a true admissible trajectory of the system; it is an arbitrary curve $\xi_d \in V$.

6.6.1. Dynamic Embedding

If a system is fully actuated by forces applied directly to each configuration variable, tracking a desired trajectory is trivial in both the continuous and discrete domains. This is the motivation for a technique called *dynamic embedding.*

In dynamic embedding, the system is augmented with forces on each configuration variable. The augmented system is optimized with very little cost associated with these fictitious inputs. This optimization typically converges to the desired solution.

The optimization is repeated using the previous solution as the initial condition, but the cost of the fictitious inputs are increased. The algorithm will naturally reduce the fictitious force and rely more on the other inputs in the system. This process is repeated multiple times with the solution from the preceding optimization, becoming the initial condition for the next, while increasing the cost of the fictitious inputs. The process is stopped when the augmented inputs are virtually unused. At this point, the fictitious inputs are removed from the system, and the original cost is optimized using the last trajectory as the initial trajectory.

This procedure is notably effective for systems with singularities, like the pendulum on a cart example in Sec. 6.7.1. The augmented inputs effectively remove large cost barriers near the singularities, allowing the optimization to reach the desired region of the trajectory space. The barriers are reintroduced as the inputs become more expensive, but at this point the optimization is already working in the desired region and less likely to converge to an undesirable local minimum.

6.6.2. Optimizations with Constraints

Projection operator trajectory optimization is capable of optimizing systems with holonomic constraints without modification. However, convergence rates are often improved by special treatment of the constraints. Two approaches are described here. The type of treatment depends on the nature of the constraint, but the motivation behind both approaches is to replace the strict constraint with something more flexible so that the optimization has more freedom to explore the trajectory space.

For systems like the marionette (Sec. 6.7.3) which are actuated by kinematically-controlled holonomic constraints, the purpose of each string constraint is to model an externally applied force. Each constraint is removed and replaced with a wrench force that acts on the attachment point of the string. This forced system is optimized to track the desired trajectory. Finally, the forced trajectory is transformed back to the constrained system. In the case of the string constraints, the control points of each string is chosen to make the string collinear with the wrench force in $\mathbb{R}^3$ and the length of the string is measured from the puppet's configuration.

Other constraints model internal structures of a system, such as a closed kinematic chain. In these cases, the constraint $h(q)$ is replaced by an incremental cost term $h^2(q)$ to penalize configurations that violate the constraint. As the optimization proceeds, the penalty is increased until the constraint error is negligible. This resulting trajectory is projected into the trajectory space of the constrained system and serves as the initial trajectory of the final optimization.

The dynamic embedding technique can be used in addition to either of these constraint modifications. For example, Sec. 6.7.3 describes a marionette optimization that uses dynamic embedding to develop a trajectory using wrenches in place of the strings. That result is transformed back into a constrained trajectory to become the initial trajectory for the final optimization. The initial trajectory is so close to the minimum that the optimization can immediately use Newton's method and terminates within a few iterations.

The drawback of both techniques is that they are difficult to automate for arbitrary systems. The designer must identify which constraints to remove, decide how to replace them, how to increase the cost terms, and design a procedure to transform the result back into a trajectory of the original system.

6.7. Examples

This section describes several example applications of the optimization algorithm to different systems. Several of the examples include comparisons between continuous and discrete time optimizations. Sec. 6.7.1 discusses an inverted pendulum on a cart. An optimization for a two-dimensional marionette arm is considered in Sec. 6.7.2 and includes a hardware experiment. A humanoid marionette is optimized in Sec. 6.7.3. Finally, a human hand is briefly considered in Sec. 6.7.4.

`Trep` provides tools for continuous and discrete time simulation and dynamics. The discrete time code has been extensively optimized for execution speed while the continuous time has not. Accordingly, some examples provide timing benchmarks for discrete time optimizations but not continuous time.

The continuous time examples use `Scipy`'s built-in routine `odeint`[**18**] for numeric integration. This function is based on the LSODA numeric integrator which selects between BDF and Adams methods for stiff and non-stiff problems respectively[**34**].

6.7.1. Pendulum on a Cart

This section deals with the pendulum on a cart shown in Fig. 6.3. The pendulum is optimized in continuous and discrete time to find a trajectory that moves the pendulum from a hanging position to an inverted one. This example illustrates the dynamic embedding technique described in Sec. 6.6.1. The source code for this example is distributed with `trep` in the file `/examples/pend-on-cart-optimization.py`.

The pendulum on a cart is actuated by moving the cart horizontally. There is no joint torque applied directly to the pendulum. This creates a singularity in the dynamics.

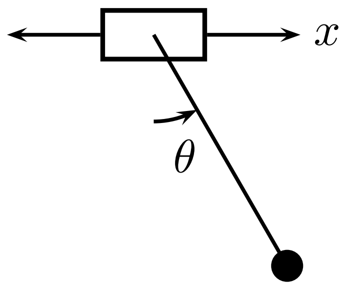


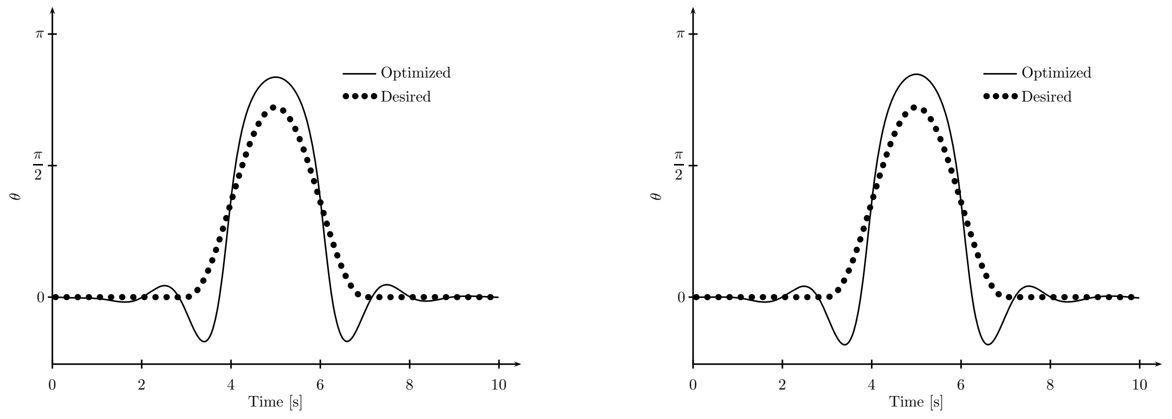
Figure 6.3. The pendulum on a cart is controlled by moving the cart horizontally.

When the pendulum is horizontal, accelerating the cart no longer applies a torque to the pendulum.

The desired trajectory moves the pendulum from a hanging position to the inverted position and back again, so the system must pass through this singularity. However, if we use an initial trajectory for the optimization of the pendulum hanging in the stable position, the optimization will not cross the singularity. It will converge to a trajectory that swings the pendulum to just below the horizontal position.

Instead, the technique described in Sec. 6.6.1 was used to generate a new initial trajectory for the system. A fictitious input was added to directly apply a torque to the pendulum. This removes the singularity and allows the optimization to converge to the desired trajectory.

This augmented optimization was run three times. Initially, a low cost was associated with the fictitious torque. In the following two optimizations, the cost was drastically increased until. After the last optimization, the applied torque was essentially unused



(a) Continuous Time (b) Discrete Time

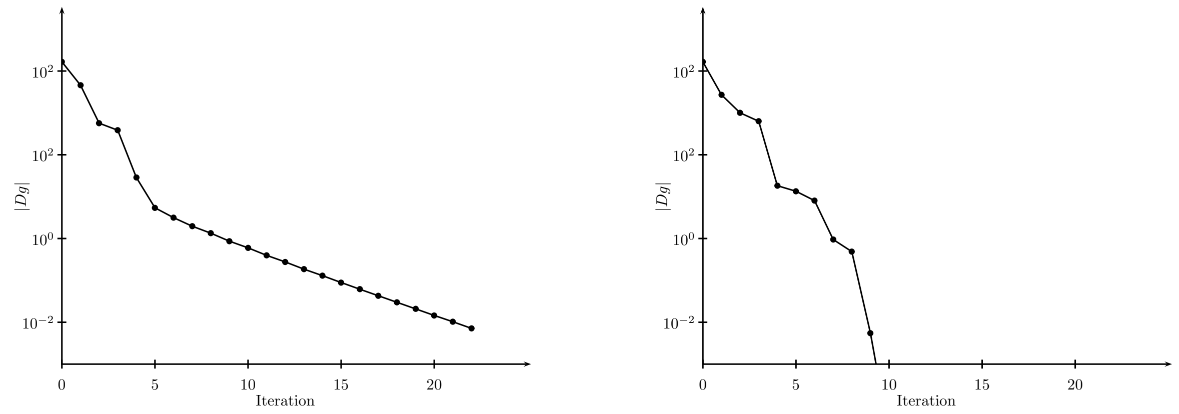
Figure 6.4. Both optimizations converge to the same trajectory for the angle of the pendulum.

because of the associated cost. That trajectory was then used as the initial trajectory for the original problem.

This optimization was performed in both continuous and discrete time. The final results are shown in Fig. 6.4. Both optimizations converge to essentially the same trajectory. As can be seen, the final trajectory approximates the desired path. The path cannot be tracked directly because of the limited dynamics of the system.

The discrete time optimization completes in 112s on a 2.2Ghz processor.. Each iteration takes between 2 and 3 seconds depending on the number of Armijo steps taken.

Figure 6.5 shows convergence plots for the continuous and discrete optimizations during one of the intermediate optimizations. The discrete optimization converges in considerably fewer steps than the continuous one. The discrete time optimization was also more numerically robust. It tolerates larger ratios between the maximum and minimum costs of the individual states and reliably converges to stricter terminal conditions.



(a) Continuous Time (b) Discrete Time

Figure 6.5. Convergence plots for continuous and discrete time optimizations from a simulated desired trajectory.

The continuous optimization, on the other hand, was much more sensitive to the optimization parameters. Large cost ratios create stiffness that cause the descent direction calculation to fail when the Ricatti equation is numerically integrated. The discrete optimization is able to converge for ratios as high as $10^3 : 10^{-3}$ whereas the ratio had to be reduced to $100 : 1$ for the continuous optimization to proceed. In practice, large ratios allow the optimization to ignore components of the desired trajectory so that other components can be tracked more closely.

The terminal conditions also had to be relaxed for the continuous optimization to formally converge near the optimal solution. This is largely due to the error introduced by numeric integration being larger than the descent tolerance. The discrete optimization reliably converges with a terminal condition of $|Dg(\xi) \circ \delta\xi < 10^{-6}$. The condition was relaxed to $|Dg(\xi) \circ \delta\xi < 10^{-2}$ for the continuous optimization.

6.7.2. 2D Marionette Arm

This example considers the two-dimensional marionette arm shown in Fig. 6.6. The arm is actuated by a single string. The string is modeled as a holonomic constraint. The length of the string and the horizontal position of top end-point are controlled by kinematic configuration variables.

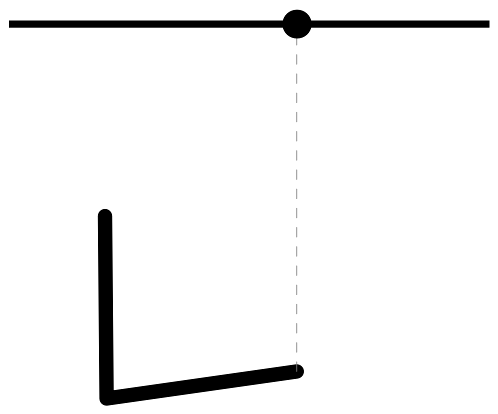


Figure 6.6. The 2D marionette arm is actuated by a single string.

A desired waving motion for the arm was designed by hand using computer animation software[**1**]. The software was used to create an animation skeleton for a two-link two dimensional arm (without a string). The skeleton was animated using key-frames and interpolation without regard to the dynamics of the marionette or the actuator limitations. An eight second waving motion was created to raise the arm from a lowered position, wave three times, and then lower back to the starting position.

The animated joint angles were used directly as the desired angles for the marionette arm. The position and length of the string, which are undefined by the animation, were uniformly set to zero. Similarly, the additional state components (i.e, configuration velocities) were set to zero. To set up the optimization, the angles were given relatively large weights while the other state components were weighted lightly.

An initial trajectory was generated by taking the initial conditions from the desired trajectory (modified to satisfy the closed kinematic chain) and simulating the system with constant inputs. This results in a valid trajectory to seed the optimization. The optimization was performed in continuous time and converged in 11 iterations.

The cost is plotted against iterations in Fig. 6.7. The optimization converges towards the minimum very quickly at first as the steepest-descent direction moves toward the minimum. After a few steps, the trajectory is near the optimum trajectory. The Newton-descent direction takes over after the 5th iteration and quickly converges on the final trajectory.

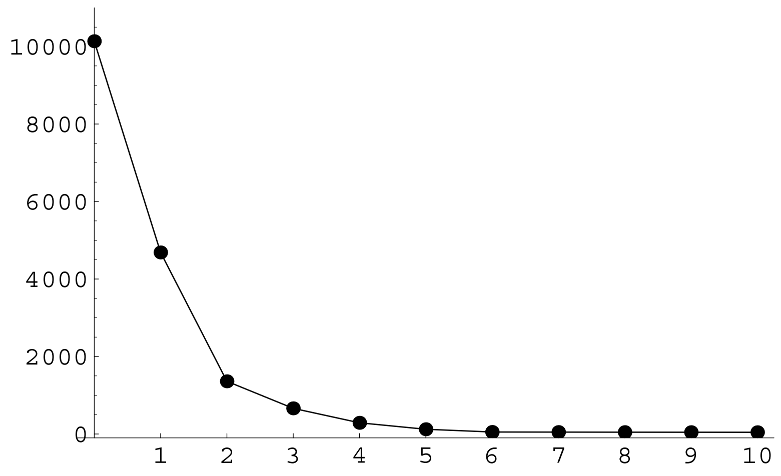


Figure 6.7. The trajectory cost as a function of the iteration.

Fig. 6.8 shows several snapshots of the optimized trajectory compared to the desired motion. The optimized trajectory tracks the desired animation well.

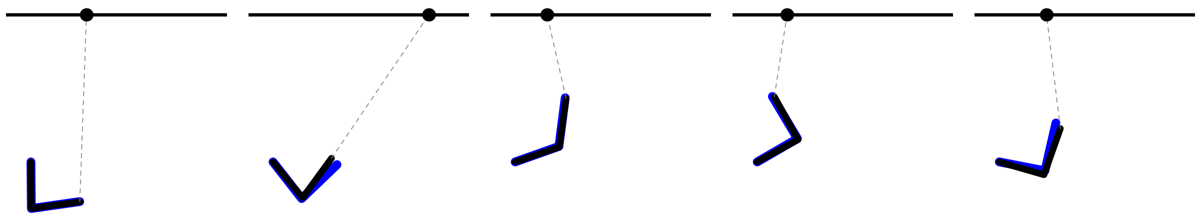


Figure 6.8. Several frames from the resulting trajectory. The marionette arm and animated input are drawn in black and blue, respectively.

The hardware platform shown in Fig. 6.9 was built to verify the results of the optimization. The hardware directly measures both arm angles, the horizontal position of the string carriage, and the length of the string. This allows the controller to estimate the entire state and implement the feedback control law when running a trajectory.

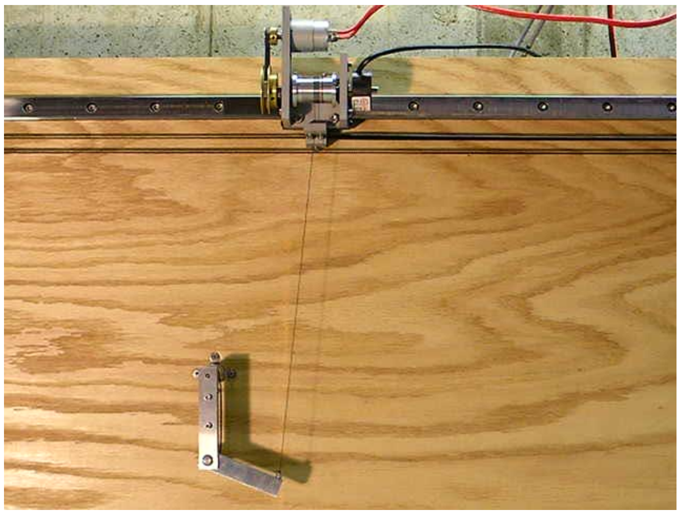
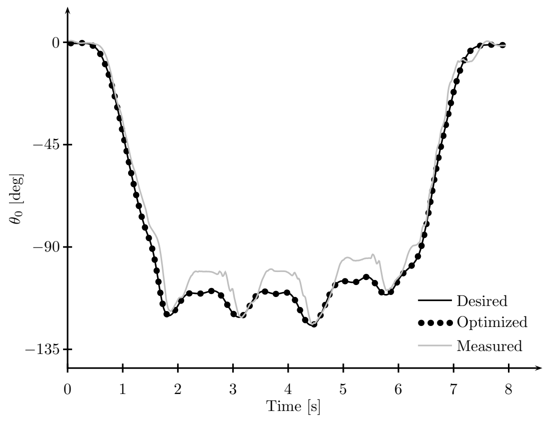
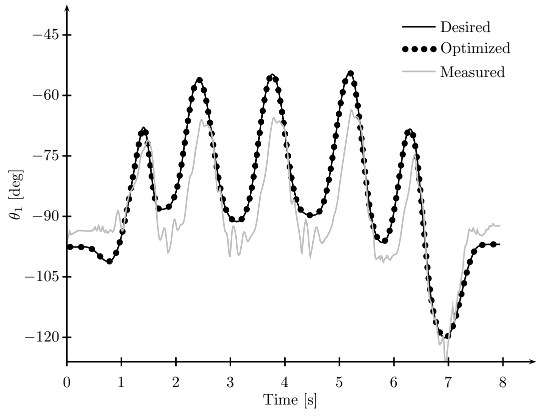


Figure 6.9. The optimization results were verified on a hardware platform.

The results of the optimization and hardware experiment are shown in Fig. 6.10 and Fig. 6.11. Trajectories for the shoulder and elbow joint angles are shown in Fig. 6.10a and Fig. 6.10b, respectively. The trajectory found by the optimization algorithm corresponds



(a) Shoulder Angle



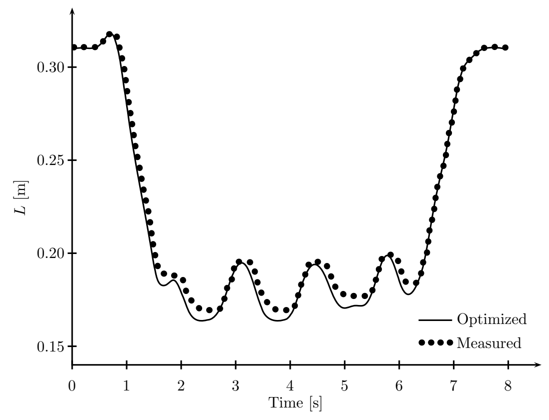
(b) Elbow Angle

Figure 6.10. Plots of the desired, optimized, and measured arm angles. The optimized trajectories match the desired trajectories very well.

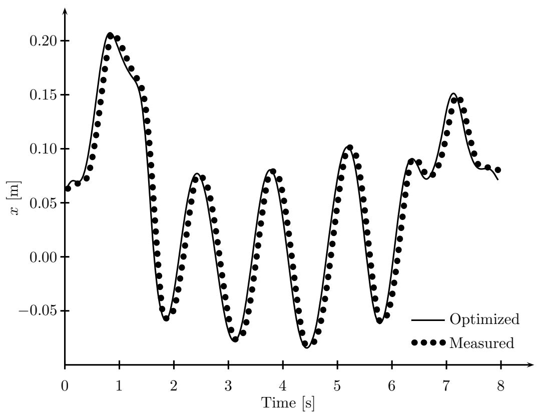
very closely to the desired trajectory despite the limitations imposed by the dynamics. The experimental results also show good agreement with the optimal trajectory.

The length of the string and horizontal position of the carriage are plotted in Fig. 6.11a and Fig. 6.11b. These trajectories were determined completely by the optimization algorithm. They are not tracking a pre-specified desired path.

The hardware experiment confirmed that the trajectories found by the optimization algorithm are valid and approximate the desired motion. The following section continues by applying the optimization to a model of the full marionette.



(a) String Length



(b) String Position

Figure 6.11. Plots of the optimized and measured system inputs. The inputs were synthesized from scratch by the trajectory optimization.

6.7.3. 3D Marionette

We now consider the model of a humanoid marionette shown in Fig. 6.12. The marionette has 40 configuration variables and is actuated by 6 strings. The strings are modeled as holonomic constraints. Kinematic configuration variables control the two-dimensional position of the end-point of each string as well as the string length. There are no joint torques and only slight damping.

Two optimizations are discussed in the following subsections. The optimization in Sec. 6.7.3.1 uses a desired trajectory that was generated separately by simulating the system. Section 6.7.3.2 discusses an optimization from motion capture data.

6.7.3.1. Desired Motion: Simulated Trajectory. A desired trajectory was generated by simulating the system forward in time. The lengths of the arm and leg strings were varied sinusoidally to create a walking motion. The configuration trajectory was saved. The rest of the state (e.g, configuration velocity or discrete momentum) and the

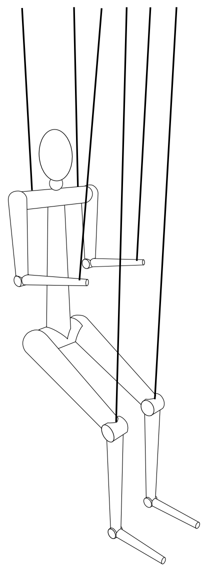
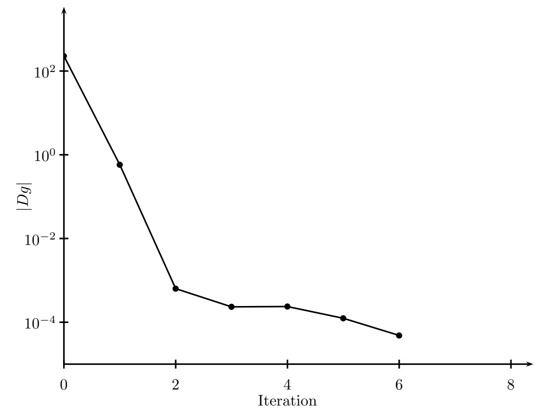


Figure 6.12. The 3D Marionette is actuated by six kinematic strings.

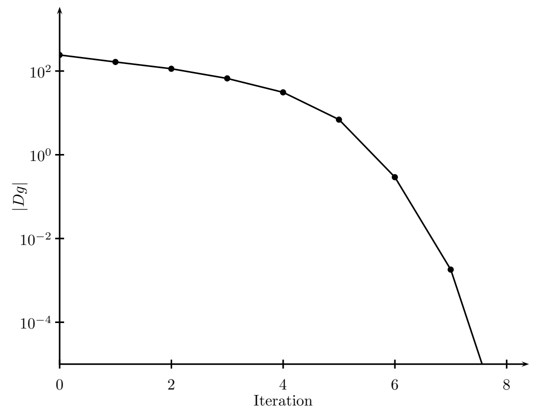
simulation inputs were discarded and replaced with uniformly zero trajectories. This results in a smooth desired trajectory that we expect the puppet to be able to track, but is still an in-feasible trajectory.

The marionette was optimized to the desired trajectory in both continuous and discrete time. Both optimizations successfully converged to solutions that track the desired configuration very well. Convergence plots for both optimizations are shown in Fig. 6.13. The source code for the discrete optimization is distributed with `trep` in the file `/examples/puppet-optimization.py`.

In this case, the continuous optimization initially converges drastically faster than the discrete one. It tracks the desired trajectory almost perfectly after a single step. The discrete optimization makes slow progress initially but converges quickly after about five iterations. The discrete optimization takes 5:50s to finish. Each iteration takes between 15 and 60 seconds depending on the descent direction type and number of Armijo steps.



(a) Continuous Time



(b) Discrete Time

Figure 6.13. Convergence plots for continuous and discrete time optimizations from a simulated desired trajectory.

Although the convergence plot is flattering for the continuous optimization, there were numerous problems. As with the pendulum example, the continuous optimization was sensitive to the optimization parameters. Large ratios between the maximum and minimum state cost cause the optimization to fail. The terminal conditions had to be relaxed again as well. The discrete time optimization suffers from neither of these problems.

6.7.3.2. Desired Motion: Motion Capture Data. A more practical application of the trajectory optimization is finding trajectories to track data acquired from a motion capture system. In this example, a desired trajectory was generated using a Microsoft Kinect to record a student walking in place. This process is illustrated in Fig. 6.14.

In this case, the continuous optimization was unable to converge. The discrete optimization converged successfully and found a trajectory that closely approximates the student's movement. Figure 6.15 plots the desired trajectory and optimization result for the angle of the right elbow as an example. The trajectory found by the optimization

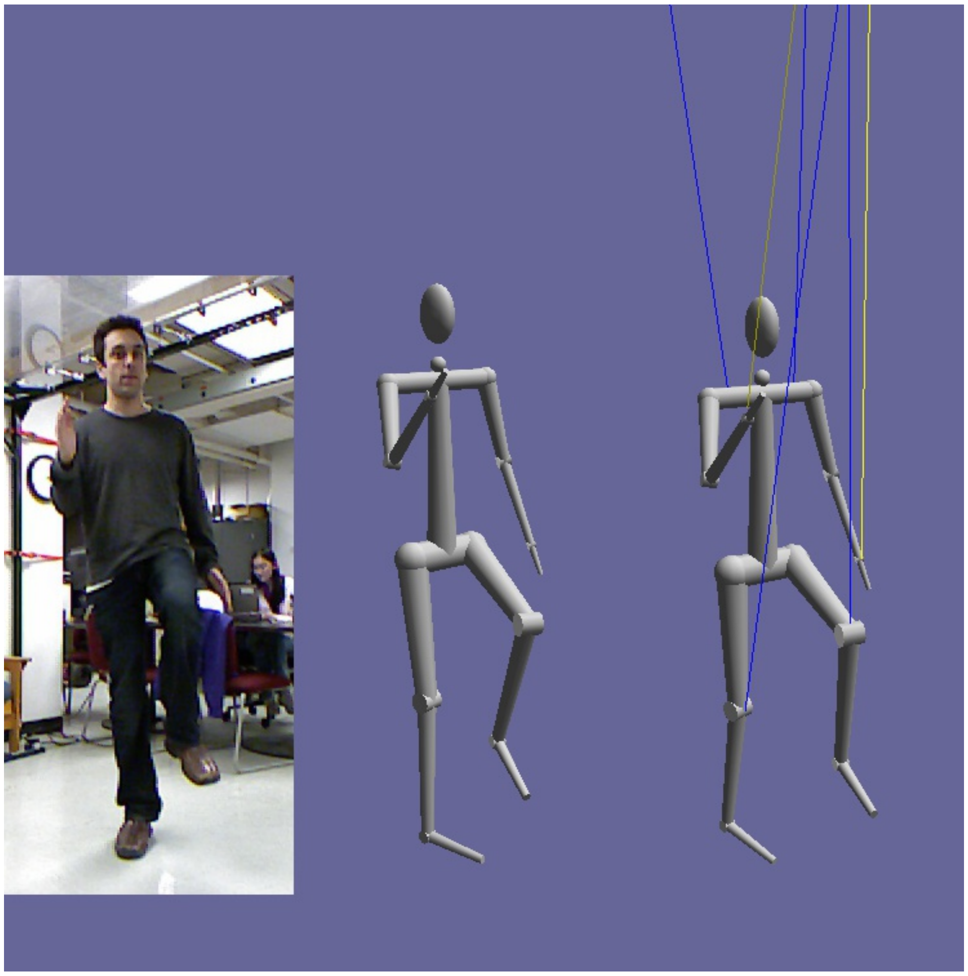


Figure 6.14. These three images show a single frame from the motion-capture optimization. The left-most picture shows images recorded by a Microsoft Kinect. The middle figure is the motion capture data found from the image. The right-most figure is the optimized trajectory.

tracks the desired trajectory very well. However, a large amount of noise was introduced. This is most likely caused by too large of a ratio between the weight of the configuration portion of the state compared the discrete momentum portions and the cost of the inputs.

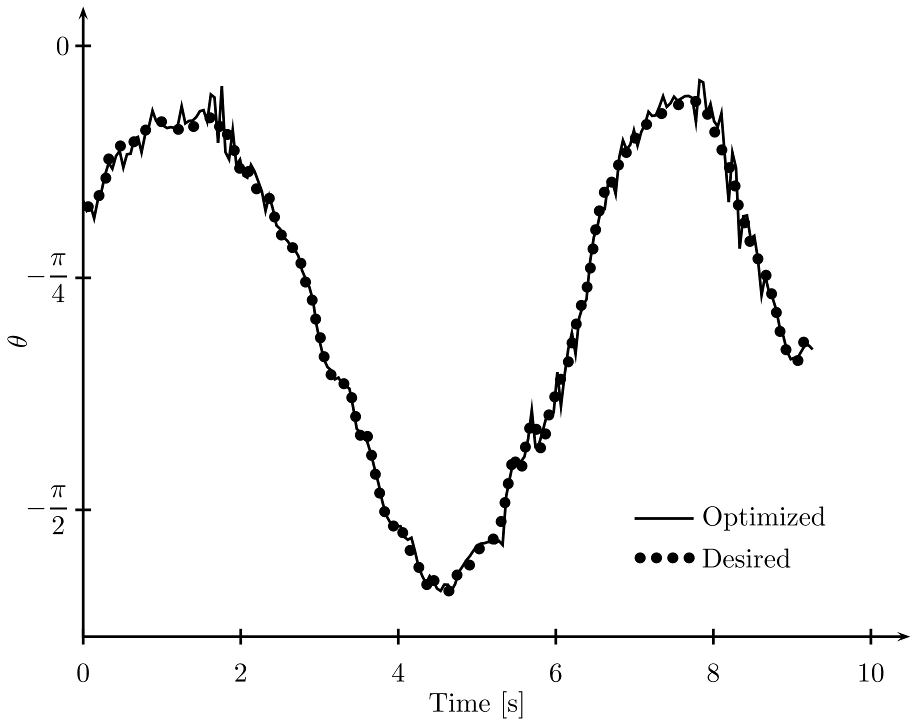


Figure 6.15. The optimal trajectory for the right arm elbow tracks the desired trajectory well.

6.7.4. Human Hand

For the final example, the trajectory optimization was applied to a simplified model of the human hand. The hand has 16 degrees of freedom. Each bone is a full rigid body with mass and inertia. The hand is actuated by 25 tendons. The inputs to the system are the tensions in each tendon, modified by a force-length curve.

A desired motion for the hand was created in a computer animation package. Starting from an open posture, the hand closes into a fist and then returns to the starting position. A frame of the optimization result is shown in Fig. 6.16.

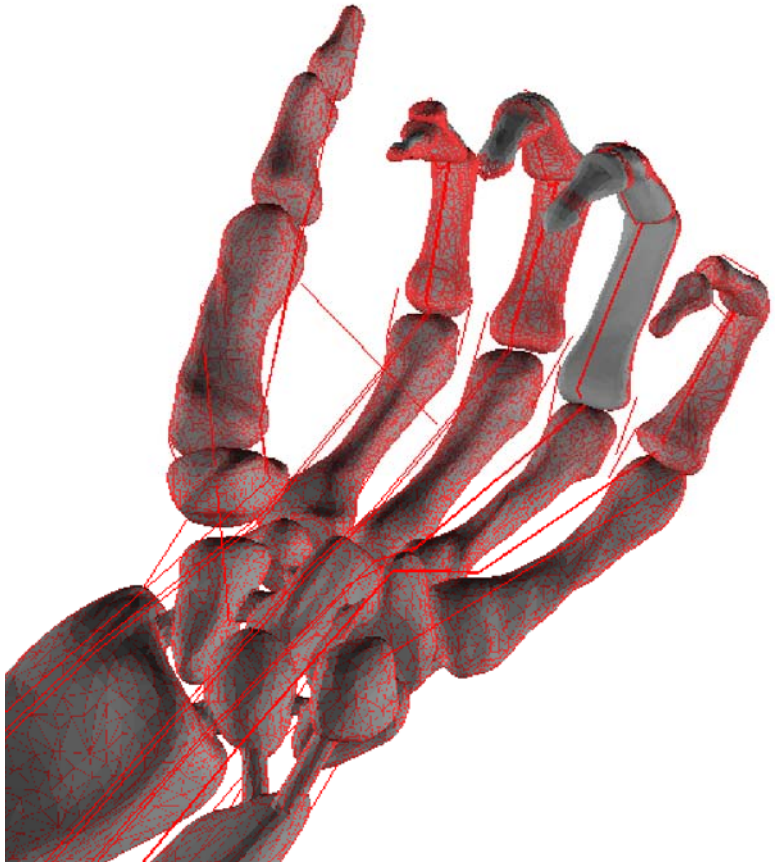


Figure 6.16. An optimization of the human hand model. The red mesh represents the desired motion for the hand.

The trajectory of the angle of the third distal phalanx is plotted in Fig. 6.17 along with the desired angle. The resulting trajectory can be seen to approximate the desired path. This plot is representative of the entire configuration. As the plot suggests, the trajectory found by the optimization tracks the desired path. However, the limited actuation of the model prevent the hand from tracking the desired path perfectly.

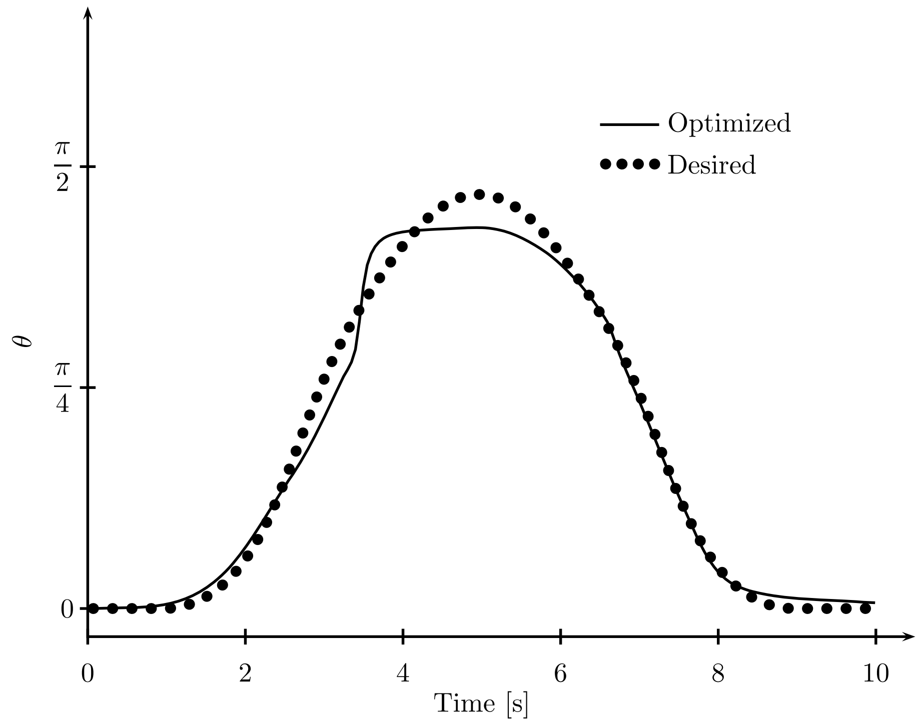


Figure 6.17. The optimized joint trajectory for the third distal phalanx tracks the desired trajectory.

References

[1] Blender. *http://www.blender.org*, 2007.

[2] AIST, Univ. of Tokyo, , and MSTC. Open architecture humanoid robotics platform, 2008. http://www.is.aist.go.jp/humanoid/openhrp/.

[3] B.D.O. Anderson and J.B. Moore. *Optimal Control: Linear Quadratic Methods.* Prentice Hall, Inc, 1990.

[4] L. Armijo. Minimization of functions having lipschitz continuous first-partial derivatives. *Pacific Journal of Mathematics*, Vol. 16, 1966.

[5] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1995.

[6] P. Betsch. A unified approach to the energy-consistent numerical integration of nonholonomic mechanical systems and flexible multibody dynamics. *GAMM Mitteilungen*, 27:66–87, 2004.

[7] P. Betsch and S. Leyendecker. The discrete null space method for the energy consistent integration of constrained mechanical systems. part ii: Multibody dynamics. *Int. J. Numer. Meth. Engng*, 67(499-552), 2006.

[8] B. Bollobas. *Modern Graph Theory.* Springer, 1998.

[9] R.W. Brockett, A. Stokes, and F. Park. A geometrical formulation of the dynamical equations describing kinematic chains. In *Proc. IEEE Conf. on Robotics and Automation*, pages 637–641, Atlanta, GA, 1993.

[10] F. Bullo and A.D. Lewis. *Geometric Control of Mechanical Systems.* Number 49 in Texts in Applied Mathematics. Springer-Verlag, 2004.

[11] C.T. Chen. *Linear System Theory and Design.* Saunders College Publishing, 1984.

[12] R. Featherstone. *Robot Dynamics Algorithms.* Kluwer Academic Publishers, 1987.

[13] R.C. Fetecau, J.E. Marsden, M. Ortiz, and M. West. Nonsmooth Lagrangian mechanics and variational collision integrators. *SIAM Journal on Applied Dynamical Systems*, 2003.

[14] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration*. Springer Series in Computational Mathematics; 31. Springer-Verlag, 2004.

[15] J. Hauser. A projection operator approach to the optimization of trajectory functionals. In *IFAC World Congress*, Barcelona, Spain, 2002.

[16] J. Hauser and A. Saccon. A barrier function method for the optimization of trajectory functionals with constraints. In *45th IEEE Conference on Decision and Control*, 2006.

[17] E.R. Johnson and T.D. Murphey. Dynamic modeling and motion planning for marionettes: Rigid bodies articulated by massless strings. In *International Conference on Robotics and Automation*, Rome, Italy, 2007.

[18] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[19] C.T. Kelly. *Iterative Methods of Linear and Nonlinear Equations*. SIAM, 1995.

[20] C.T. Kelly. *Iterative Methods for Optimization*. SIAM, 1999.

[21] L. Kharevych, W. Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schroder, and M. Desbrun. Geometric, variational integrators for computer animation. *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2006.

[22] K.Yamane. *Simulating and Generating Motions of Human Figures*. Springer-Verlag, 2004.

[23] A. Lew, J.E. Marsden, M. Ortiz, and M. West. An overview of variational integrators. *Finite Element Methods: 1970's and Beyond*, 2003.

[24] A. Lew, J.E. Marsden, M. Ortiz, and M. West. Variational time integrators. *International Journal for Numerical Methods in Engineering*, pages 153–212, 2004.

[25] S. Leyendecker, P. Betsch, and P. Steinmann. The discrete null space method for the energy consistent integration of constrained mechanical systems: Part iii: Flexible multibody dynamics. *Multibody System Dynamics*, 2007.

[26] S. Leyendecker, S. Ober-Blbaum, J. E. Marsden, and M. Ortiz. Discrete mechanics and optimal control for constrained systems. *Optimal Control Applications and Methods*, 31(6):505–528, 2010.

[27] A. Locatelli. *Optimal Control: An Introduction*. Birkhauser, 2001.

[28] J. E. Marsden and T. S. Ratiu. *Introduction to Mechanics and Symmetry*. Springer, 1999.

[29] J. E. Marsen and M. West. Discrete mechanics and variational integrators. *Acta Numerica*, 10:357–514, 2001.

[30] R.M. Murray, Z. Li, and S.S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[31] S. Ober-Blbaum. Discrete mechanics and optimal control. *University of Paderborn, Germany*, 2008.

[32] S. Ober-Blbaum, O. Junge, and J.E. Marsden. Discrete mechanics and optimal control: An analysis. *ESAIM: Control, Optimisation and Calculus of Variations*, 17(2):322–352, 2011.

[33] F.C. Park and J.E. Bobrow. A recursive algorithm for robot dynamics using lie groups. *International Conference on Robotics and Automation*, 1994.

[34] Linda Petzold. Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations. *SIAM Journal on Scientific and Statistical Computing*, 4(1):136–148, 1983.

[35] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C Second Edition*. Cambridge University Press, 1992.

[36] D. Schaechter, T. Kane, D. Levinson, and P. Mitiguy. Autolev, 2008. http://www.autolev.com/.

[37] R. Smith. Dynamics Simulation: A whirlwind tour (current state, and new frontiers), 2004. http://ode.org/slides/parc/dynamics.pdf.

[38] R. Smith. Open Dynamics Engine, 2008. http://www.ode.org.

[39] E. Todorov and Y. Tassa. Iterative local dynamic programming. *IEEE Adaptive Dynamic Programming and Reinforcement Learning*, 2009.

[40] M. West. Variational integrators. *California Institute of Technology Thesis*, 2004.

[41] M. West, C. Kane, J. E. Marsden, and M. Ortiz. Variational integrators, the Newmark scheme, and dissipative systems. In *Int. Conference on Differential Equations*, 1999.

[42] A. Witkin, M. Gleicher, and W. Welch. Interactive dynamics. *ACM SIGGRAPH Computer Graphics*, 24(2):11–21, 1990.

APPENDIX A

Derivatives of the Dynamics

This section describes the state representation and lists the full equations for the first and second dynamic linearizations for continuous and discrete systems. The systems include forcing, holonomic constraints, and kinematic configuration variables. These equations were derived with the same method used in Sec. 3.3 and Sec. 4.4. The continuous time equations are discussed in Sec. A.1. The discrete time equations are discussed in Sec. A.2.

A.1. Continuous Dynamics

We consider a continuous time mechanical system with a set of dynamic configuration variables q and kinematic configuration variables ρ. It is convenient to combine the configuration variables into a single vector $Q = \begin{bmatrix} q \\ \rho \end{bmatrix}$. This leads to an obvious choice for a state representation that is used throughout this derivation:

$$x = \begin{bmatrix} q \\ \rho \\ \dot{q} \\ \dot{\rho} \end{bmatrix} \qquad \text{and} \qquad \bar{u} = \begin{bmatrix} u \\ \ddot{\rho} \end{bmatrix}.$$

where $\bar{u}$ is used to distinguish the state input vector $\bar{u}$ from the force input vector u. The kinematic input in this case is the second time derivative (acceleration) of the kinematic configuration.

Section A.1.1 provides equations for the derivatives of the continuous Lagrangian. Definitions and derivatives related to the constraints are discussed in Sec. A.1.2. The continuous Euler-Lagrange equations are provided in Sec. A.1.3, their first linearization in Sec. A.1.4, and their second linearization in Sec. A.1.5.

A.1.1. Continuous Lagrangian and Derivatives

The continuous Lagrangian for the tree representation is

$$L(Q, \dot{Q}) = \sum_k \tfrac{1}{2} v_{s,k}^{bT} M_k v_{s,k}^b - \sum_\ell V_\ell(Q)$$

where the body velocities $v_{s,(}^{bT}k)$ are calculated by the tree equations and a potential function $V_\ell(Q)$ is provided for each type of potential energy in the system.

The following derivatives of the Lagrangian are required to calculate the dynamics.

$$\frac{\partial L}{\partial q_i} = \sum_k v_{s,k}^{bT} M_k \frac{\partial v_{s,k}^b}{\partial q_i} - \sum_\ell \frac{\partial V_\ell}{\partial q_i}$$

$$\frac{\partial^2 L}{\partial \dot{q}_j \partial \dot{q}_i} = \sum_k \frac{\partial v_{s,k}^{bT}}{\partial q_j} M_k \frac{\partial v_{s,k}^b}{\partial q_i}$$

$$\frac{\partial^2 L}{\partial q_j \partial \dot{q}_i} = \sum_k \frac{\partial v_{s,k}^{bT}}{\partial q_j} M_k \frac{\partial v_{s,k}^b}{\partial \dot{q}_i} + v_{s,k}^{bT} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_j \partial \dot{q}_i}$$

$$\frac{\partial^2 L}{\partial q_j \partial q_i} = \sum_k \frac{\partial v_{s,k}^{bT}}{\partial q_j} M_k \frac{\partial v_{s,k}^b}{\partial q_i} + v_{s,k}^{bT} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_j \partial q_i} - \sum_\ell \frac{\partial^2 V_\ell}{\partial q_j \partial q_i}$$

Linearizations of the dynamics require higher derivatives as well. These are simply found by the chain rule. Each equation is still evaluated numerically by the tree representation and provided potential functions.

$$\frac{\partial^3 L}{\partial q_k \partial \dot{q}_j \partial \dot{q}_i} = \sum_k \frac{\partial v_{s,k}^{bT}}{\partial \dot{q}_j} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_k \partial \dot{q}_i} + \frac{\partial v_{s,k}^{bT}}{\partial \dot{q}_i} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_k \partial \dot{q}_j}$$

$$\begin{aligned} \frac{\partial^3 L}{\partial q_k \partial q_j \partial \dot{q}_i} = \sum_k & \frac{\partial v_{s,k}^{bT}}{\partial q_k} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_j \partial \dot{q}_i} + \frac{\partial v_{s,k}^{bT}}{\partial q_j} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_k \partial \dot{q}_i} \\ & + \frac{\partial v_{s,k}^{bT}}{\partial \dot{q}_i} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_k \partial q_j} + v_{s,k}^{bT} M_k \frac{\partial^3 v_{s,k}^b}{\partial q_k \partial q_j \partial \dot{q}_i} \end{aligned}$$

$$\begin{aligned} \frac{\partial^3 L}{\partial q_k \partial q_j \partial q_i} = \sum_k & \frac{\partial v_{s,k}^{bT}}{\partial q_k} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_j \partial q_i} + \frac{\partial v_{s,k}^{bT}}{\partial q_j} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_k \partial q_i} \\ & + \frac{\partial v_{s,k}^{bT}}{\partial q_i} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_k \partial q_j} + v_{s,k}^{bT} M_k \frac{\partial^3 v_{s,k}^b}{\partial q_k \partial q_j \partial q_i} - \sum_\ell \frac{\partial^3 V_\ell}{\partial q_k \partial q_j \partial q_i} \end{aligned}$$

The fourth derivatives can be particularly expensive to calculate. This cost can be significantly improved by taking advantage of the symmetry provided by mixed partials commuting to reduce the number of times these equations must be evaluated.

$$\begin{aligned}\frac{\partial^4 L}{\partial q_\ell \partial q_k \partial q_j \partial \dot{q}_i} = \sum_k \frac{\partial v_{s,k}^{bT}}{\partial q_j} M_k \frac{\partial^3 v_{s,k}^b}{\partial q_\ell \partial q_k \partial \dot{q}_i} + \frac{\partial v_{s,k}^{bT}}{\partial q_k} M_k \frac{\partial^3 v_{s,k}^b}{\partial q_\ell \partial q_j \partial \dot{q}_i} \\ + \frac{\partial v_{s,k}^{bT}}{\partial \dot{q}_i} M_k \frac{\partial^3 v_{s,k}^b}{\partial q_\ell \partial q_k \partial q_j} + \frac{\partial v_{s,k}^{bT}}{\partial q_\ell} M_k \frac{\partial^3 v_{s,k}^b}{\partial q_k \partial q_j \partial \dot{q}_i} + \frac{\partial^2 v_{s,k}^{bT}}{\partial q_\ell \partial q_k} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_j \partial \dot{q}_i} \\ + \frac{\partial^2 v_{s,k}^{bT}}{\partial q_\ell \partial \dot{q}_i} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_k \partial q_j} + \frac{\partial^2 v_{s,k}^{bT}}{\partial q_\ell \partial q_j} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_k \partial \dot{q}_i} v_{s,k}^{bT} M_k \frac{\partial^4 v_{s,k}^b}{\partial q_\ell \partial q_k \partial q_j \partial \dot{q}_i}\end{aligned}$$

$$\begin{aligned}\frac{\partial^4 L}{\partial q_\ell \partial q_k \partial \dot{q}_j \partial \dot{q}_i} = \sum_k \frac{\partial v_{s,k}^{bT}}{\partial \dot{q}_j} M_k \frac{\partial^3 v_{s,k}^b}{\partial q_\ell \partial q_k \partial \dot{q}_i} + \frac{\partial v_{s,k}^{bT}}{\partial \dot{q}_i} M_k \frac{\partial^3 v_{s,k}^b}{\partial q_\ell \partial q_k \partial \dot{q}_j} \\ + \frac{\partial^2 v_{s,k}^{bT}}{\partial q_\ell \partial \dot{q}_i} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_k \partial \dot{q}_j} + \frac{\partial^2 v_{s,k}^{bT}}{\partial q_\ell \partial \dot{q}_j} M_k \frac{\partial^2 v_{s,k}^b}{\partial q_k \partial \dot{q}_i}\end{aligned}$$

A.1.2. Constraints

The constrained continuous Euler-Lagrange equation uses non-holonomic constraints of the form $A(Q)\dot{Q} = 0$. Every holonomic constraint $h(Q)$ has an equivalent non-holonomic constraint which is found by differentiating $h(Q)$ with respect to time:

$$0 = h(Q) = \frac{\partial}{\partial t} h(Q) = \frac{\partial h}{\partial Q} \dot{Q} = A_d \dot{q} + A_k \dot{\rho}$$

where $A_d = \frac{\partial h}{\partial q}$ and $A_k = \frac{\partial h}{\partial \rho}$. Time derivatives of A have the following properties:

$$\begin{aligned}
\frac{dA}{dt} &= \sum_k \frac{\partial A}{\partial Q_k}\dot{Q}_k \\
\frac{\partial}{\partial Q_i}\frac{dA}{dt} &= \sum_k \frac{\partial^2 A}{\partial Q_i \partial Q_k}\dot{Q}_k = \frac{d}{dt}\frac{\partial A}{\partial Q_i} \\
\frac{\partial^2}{\partial Q_j \partial Q_i}\frac{dA}{dt} &= \sum_k \frac{\partial^3 A}{\partial Q_j \partial Q_i \partial Q_k}\dot{Q}_k = \frac{d}{dt}\frac{\partial^2 A}{\partial Q_j \partial Q_i} \\
\frac{\partial}{\partial \dot{Q}_i}\frac{dA}{dt} &= \frac{\partial}{\partial \dot{Q}_i}\sum_k \frac{\partial A}{\partial Q_k}\dot{Q}_k = \frac{\partial A}{\partial Q_i}.
\end{aligned}$$

Derivatives of the constraint matrix $A = \dfrac{\partial h}{\partial Q}$ are calculated from the chain rule.

$$\begin{aligned}
\frac{\partial A}{\partial q_i} &= \frac{\partial^2 h}{\partial q_i \partial Q} & \frac{dA}{dt} &= \textstyle\sum_k \frac{\partial^2 h}{\partial q_k \partial Q}\dot{q}_k \\
\frac{\partial^2 A}{\partial q_j \partial q_i} &= \frac{\partial^3 h}{\partial q_j \partial q_i \partial Q} & \frac{d}{dt}\frac{\partial A}{\partial q_i} &= \textstyle\sum_k \frac{\partial^3 h}{\partial q_k \partial q_i \partial Q}\dot{q}_k \\
& & \frac{d}{dt}\frac{\partial^2 A}{\partial q_j \partial q_i} &= \textstyle\sum_k \frac{\partial^4 h}{\partial q_k \partial q_j \partial q_i \partial Q}\dot{q}_k
\end{aligned}$$

The derivatives of $h(\cdot)$ must be provided by each type of constraint.

A.1.3. Dynamics

The constrained Euler-Lagrange equation with kinematic inputs is given by (A.1). The constraint forces λ are calculated by (A.2). The common terms in these two equations are factored out into the quantity D defined by (A.3).

$$\ddot{q} = \left(\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right)^{-1} D \tag{A.1}$$

$$\lambda = -\left(A_d \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} A_d^T\right)^{-1} \left(\frac{dA}{dt}\dot{Q} + A_k \ddot{\rho} + A_d \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} D\right) \tag{A.2}$$

$$D = F + \frac{\partial L}{\partial q} - \frac{\partial^2 L}{\partial \dot{\rho} \partial \dot{q}}\ddot{\rho} - \frac{\partial^2 L}{\partial Q \partial \dot{q}}\dot{Q} \tag{A.3}$$

A.1.4. First Derivative

The first derivatives are calculated using the approach described in Sec. 3.3. In particular, the derivatives of λ are found by implicit differentiation of the acceleration constraint equation, substituting in the equation for the derivative of f, and solving for the desired term. This avoids differentiating the inverse matrices and leads to simpler expressions.

The first linearization of the state space form is:

$$\begin{bmatrix} \delta \dot{q} \\ \delta \dot{\rho} \\ \delta \ddot{q} \\ \delta \ddot{\rho} \end{bmatrix} = \begin{bmatrix} 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ \frac{\partial f}{\partial q} & \frac{\partial f}{\partial \rho} & \frac{\partial f}{\partial \dot{q}} & \frac{\partial f}{\partial \dot{\rho}} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta q \\ \delta \rho \\ \delta \dot{q} \\ \delta \dot{\rho} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\partial f}{\partial \ddot{\rho}} & \frac{\partial f}{\partial u} \\ I & 0 \end{bmatrix} \begin{bmatrix} \delta \ddot{\rho} \\ \delta u \end{bmatrix}.$$

Since q and ρ are both configuration variables, the derivatives in the first matrix may be consolidated to reduce the number of equations to derive (i.e, $\frac{\partial f}{\partial Q} = \left[\frac{\partial f}{\partial q}\ \frac{\partial f}{\partial \rho}\right]$ and $\frac{\partial f}{\partial \dot{Q}} = \left[\frac{\partial f}{\partial \dot{q}}\frac{\partial f}{\partial \dot{\rho}}\right]$. The derivatives with respect to a configuration variable Q_i are:

$$\frac{\partial f}{\partial Q_i} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[\frac{\partial A_d}{\partial Q_i}^T \lambda + A_d^T \frac{\partial \lambda}{\partial Q_i} + \frac{\partial D}{\partial Q_i} - \frac{\partial^3 L}{\partial Q_i \partial \dot{q} \partial \dot{q}} f\right]$$

$$\frac{\partial \lambda}{\partial Q_i} = -\left(A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} A_d^T\right)^{-1} \Bigg(\left[\frac{d}{dt}\frac{\partial A}{\partial Q_i}\right]\dot{Q} + \frac{\partial A_k}{\partial Q_i}\ddot{\rho} + \frac{\partial A_d}{\partial Q_i} f + A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \left[\frac{\partial A_d}{\partial Q_i}^T \lambda + \frac{\partial D}{\partial Q_i} - \frac{\partial^3 L}{\partial Q_i \partial \dot{q}\partial \dot{q}} f\right]\Bigg)$$

$$\frac{\partial D}{\partial Q_i} = \frac{\partial F}{\partial Q_i} + \frac{\partial^2 L}{\partial Q_i \partial q} - \frac{\partial^3 L}{\partial Q_i \partial \dot{\rho}\partial \dot{q}}\ddot{\rho} - \frac{\partial^3 L}{\partial Q_i \partial Q \partial \dot{q}}\dot{Q}.$$

The derivatives with respect the configuration velocity are calculated by the following three equations.

$$\frac{\partial f}{\partial \dot{Q}_i} = \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \left[A_d^T \frac{\partial \lambda}{\partial \dot{Q}_i} + \frac{\partial D}{\partial \dot{Q}_i}\right]$$

$$\frac{\partial \lambda}{\partial \dot{Q}_i} = -\left(A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} A_d^T\right)^{-1} \left(\frac{\partial A}{\partial Q_i}\dot{Q} + \frac{dA}{dt}\frac{\partial \dot{Q}}{\partial \dot{Q}_i} + A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \frac{\partial D}{\partial \dot{Q}_i}\right)$$

$$\frac{\partial D}{\partial \dot{Q}_i} = \frac{\partial F}{\partial \dot{Q}_i} + \frac{\partial^2 L}{\partial \dot{Q}_i \partial q} - \frac{\partial^3 L}{\partial \dot{Q}_i \partial Q \partial \dot{q}}\dot{Q} - \frac{\partial^2 L}{\partial Q_i \partial \dot{q}}$$

Derivatives with respect to the kinematic inputs are:

$$\frac{\partial f}{\partial \ddot{\rho}_i} = \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \left[A_d^T \frac{\partial \lambda}{\partial \ddot{\rho}_i} - \frac{\partial^2 L}{\partial \dot{\rho}_i \partial \dot{q}}\right]$$

$$\frac{\partial \lambda}{\partial \ddot{\rho}_i} = -\left(A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} A_d^T\right)^{-1} \left(A_k \frac{\partial \ddot{\rho}}{\partial \ddot{\rho}_i} - A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \frac{\partial^2 L}{\partial \dot{\rho}_i \partial \dot{q}}\right).$$

The derivatives with respect to the force inputs are:

$$\frac{\partial f}{\partial u_i} = \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \left[A_d^T \frac{\partial \lambda}{\partial u_i} + \frac{\partial F}{\partial u_i}\right]$$

$$\frac{\partial \lambda}{\partial u_i} = -\left(A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} A_d^T\right)^{-1} \left(A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \frac{\partial F}{\partial u_i} u_i\right)$$

A.1.5. Second Derivative

Second derivatives are found by the same procedure as the first derivatives. The second derivative of the $\ddot{q}$ with respect to configuration variables are calculated from the following three equations.

$$\frac{\partial^2 f}{\partial Q_j \partial Q_i} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[\frac{\partial^2 A_d}{\partial Q_j \partial Q_i}^T \lambda + \frac{\partial A_d}{\partial Q_i}^T \frac{\partial \lambda}{\partial Q_j} + \frac{\partial A_d}{\partial Q_j}^T \frac{\partial \lambda}{\partial Q_i} + A_d^T \frac{\partial^2 \lambda}{\partial Q_j \partial Q_i} + \right.$$
$$\left. \frac{\partial^2 D}{\partial Q_j \partial Q_i} - \frac{\partial^4 L}{\partial Q_j \partial Q_i \partial \dot{q} \partial \dot{q}} f - \frac{\partial^3 L}{\partial Q_i \partial \dot{q} \partial \dot{q}} \frac{\partial f}{\partial Q_j} - \frac{\partial^3 L}{\partial Q_j \partial \dot{q} \partial \dot{q}} \frac{\partial f}{\partial Q_i}\right]$$

$$\frac{\partial^2 \lambda}{\partial Q_j \partial Q_i} = -\left(A_d \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} A_d^T\right)^{-1} \left(\left[\frac{d}{dt} \frac{\partial^2 A}{\partial Q_j \partial Q_i}\right] \dot{Q} + \frac{\partial^2 A_k}{\partial Q_j \partial Q_i} \ddot{\rho} + \frac{\partial^2 A_d}{\partial Q_j \partial Q_i} f + \right.$$
$$\frac{\partial A_d}{\partial Q_i} \frac{\partial f}{\partial Q_j} + \frac{\partial A_d}{\partial Q_j} \frac{\partial f}{\partial Q_i} + A_d \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[\frac{\partial^2 A_d}{\partial Q_j \partial Q_i}^T \lambda + \frac{\partial A_d}{\partial Q_i}^T \frac{\partial \lambda}{\partial Q_j} + \right.$$
$$\left.\left. \frac{\partial A_d}{\partial Q_j}^T \frac{\partial \lambda}{\partial Q_i} + \frac{\partial^2 D}{\partial Q_j \partial Q_i} - \frac{\partial^4 L}{\partial Q_j \partial Q_i \partial \dot{q} \partial \dot{q}} f - \frac{\partial^3 L}{\partial Q_i \partial \dot{q} \partial \dot{q}} \frac{\partial f}{\partial Q_j} - \frac{\partial^3 L}{\partial Q_j \partial \dot{q} \partial \dot{q}} \frac{\partial f}{\partial Q_i}\right]\right)$$

$$\frac{\partial^2 D}{\partial Q_j \partial Q_i} = \frac{\partial^2 F}{\partial Q_j \partial Q_i} + \frac{\partial^3 L}{\partial Q_j \partial Q_i \partial q} - \frac{\partial^4 L}{\partial Q_j \partial Q_i \partial \dot{\rho} \partial \dot{q}} \dot{\rho} - \frac{\partial^4 L}{\partial Q_j \partial Q_i \partial Q \partial \dot{q}} \dot{Q}$$

The second derivative with respect to a configuration variable and a configuration velocity is:

$$\frac{\partial^2 f}{\partial Q_j \partial \dot{Q}_i} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[\frac{\partial A_d}{\partial Q_j}^T \frac{\partial \lambda}{\partial \dot{Q}_i} + A_d^T \frac{\partial^2 \lambda}{\partial Q_j \partial \dot{Q}_i} + \frac{\partial^2 D}{\partial Q_j \partial \dot{Q}_i} - \frac{\partial^3 L}{\partial Q_j \partial \dot{q} \partial \dot{q}} \frac{\partial f}{\partial \dot{Q}_i}\right]$$

$$\frac{\partial^2 \lambda}{\partial Q_j \partial \dot{Q}_i} = -\left(A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} A_d^T\right)^{-1} \left(\frac{\partial^2 A}{\partial Q_j \partial Q_i}\dot{Q} + \left[\frac{d}{dt}\frac{\partial A}{\partial Q_j}\right]\frac{\partial \dot{Q}}{\partial \dot{Q}_i}\right.$$
$$\left. + \frac{\partial A_d}{\partial Q_j}\frac{\partial f}{\partial \dot{Q}_i} + A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \left[\frac{\partial A_d}{\partial Q_j}^T \frac{\partial \lambda}{\partial \dot{Q}_i} + \frac{\partial^2 D}{\partial Q_j \partial \dot{Q}_i} - \frac{\partial^3 L}{\partial Q_j \partial \dot{q}\partial \dot{q}}\frac{\partial f}{\partial \dot{Q}_i}\right]\right)$$

$$\frac{\partial^2 D}{\partial Q_j \partial \dot{Q}_i} = \frac{\partial^2 F}{\partial Q_j \partial \dot{Q}_i} + \frac{\partial^3 L}{\partial Q_j \partial \dot{Q}_i \partial q} - \frac{\partial^4 L}{\partial Q_j \partial \dot{Q}_i \partial Q \partial \dot{q}}\dot{Q} - \frac{\partial^3 L}{\partial Q_j \partial Q_i \partial \dot{q}}.$$

The second derivative with respect to two configuration velocities is calculated from:

$$\frac{\partial^2 f}{\partial \dot{Q}_j \partial \dot{Q}_i} = \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \left[A_d^T \frac{\partial^2 \lambda}{\partial \dot{Q}_j \partial \dot{Q}_i} + \frac{\partial^2 D}{\partial \dot{Q}_j \partial \dot{Q}_i}\right]$$

$$\frac{\partial^2 \lambda}{\partial \dot{Q}_j \partial \dot{Q}_i} = -\left(A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} A_d^T\right)^{-1} \left(\frac{\partial A}{\partial Q_i}\frac{\partial \dot{Q}}{\partial \dot{Q}_j} + \frac{\partial A}{\partial Q_j}\frac{\partial \dot{Q}}{\partial \dot{Q}_i} + A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \frac{\partial^2 D}{\partial \dot{Q}_j \partial \dot{Q}_i}\right)$$
$$\frac{\partial^2 D}{\partial \dot{Q}_j \partial \dot{Q}_i} = \frac{\partial^2 F}{\partial \dot{Q}_j \partial \dot{Q}_i} + \frac{\partial^3 L}{\partial \dot{Q}_j \partial \dot{Q}_i \partial q} - \frac{\partial^3 L}{\partial \dot{Q}_i \partial Q_j \partial \dot{q}} - \frac{\partial^3 L}{\partial \dot{Q}_j \partial Q_i \partial \dot{q}}$$

The second derivative with respect to a configuration variable and a kinematic input is:

$$\frac{\partial^2 f}{\partial Q_j \partial \ddot{\rho}_i} = \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \left[\frac{\partial A_d}{\partial Q_j}^T \frac{\partial \lambda}{\partial \ddot{\rho}_i} + A_d^T \frac{\partial^2 \lambda}{\partial Q_j \partial \ddot{\rho}_i} + \frac{\partial^2 D}{\partial Q_j \partial \ddot{\rho}_i} - \frac{\partial^3 L}{\partial Q_j \partial \dot{q}\partial \dot{q}}\frac{\partial f}{\partial \ddot{\rho}_i}\right]$$

$$\frac{\partial^2 \lambda}{\partial Q_j \partial \ddot{\rho}_i} = -\left(A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} A_d^T\right)^{-1} \left(\frac{\partial A_k}{\partial Q_j}\frac{\partial \ddot{\rho}}{\partial \ddot{\rho}_i} + \frac{\partial A_d}{\partial Q_j}\frac{\partial f}{\partial \ddot{\rho}_i}\right.$$
$$\left. + A_d \left[\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \left[\frac{\partial A_d}{\partial Q_j}^T \frac{\partial \lambda}{\partial \ddot{\rho}_i} + \frac{\partial^2 D}{\partial Q_j \partial \ddot{\rho}_i} - \frac{\partial^3 L}{\partial Q_j \partial \dot{q}\partial \dot{q}}\frac{\partial f}{\partial \ddot{\rho}_i}\right]\right)$$

$$\frac{\partial^2 D}{\partial Q_j \partial \ddot{\rho}_i} = -\frac{\partial^3 L}{\partial Q_j \partial \dot{\rho}_i \partial \dot{q}}$$

The second derivatives with respect to a configuration variable and a force input are:

$$\frac{\partial^2 f}{\partial Q_j \partial u_i} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[\frac{\partial A_d}{\partial Q_j}^T \frac{\partial \lambda}{\partial u_i} + A_d^T \frac{\partial^2 \lambda}{\partial Q_j \partial u_i} + \frac{\partial^2 D}{\partial Q_j \partial u_i} - \frac{\partial^3 L}{\partial Q_j \partial \dot{q} \partial \dot{q}} \frac{\partial f}{\partial u_i}\right]$$

$$\begin{aligned}\frac{\partial^2 \lambda}{\partial Q_j \partial u_i} = -\left(A_d \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} A_d^T\right)^{-1} \Bigg(&\frac{\partial A_d}{\partial Q_j} \frac{\partial f}{\partial u_i} \\ &+ A_d \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[\frac{\partial A_d}{\partial Q_j}^T \frac{\partial \lambda}{\partial u_i} + \frac{\partial^2 D}{\partial Q_j \partial u_i} - \frac{\partial^3 L}{\partial Q_j \partial \dot{q} \partial \dot{q}} \frac{\partial f}{\partial u_i}\right]\Bigg)\end{aligned}$$

$$\frac{\partial^2 D}{\partial Q_j \partial u_i} = \frac{\partial^2 F}{\partial Q_j \partial u_i}.$$

The second derivatives with respect to a configuration velocity and a force input are:

$$\frac{\partial^2 f}{\partial \dot{Q}_j \partial u_i} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[A_d^T \frac{\partial^2 \lambda}{\partial \dot{Q}_j \partial u_i} + \frac{\partial^2 F}{\partial \dot{Q}_j \partial u_i}\right]$$

$$\frac{\partial^2 \lambda}{\partial \dot{Q}_j \partial u_i} = -\left(A_d \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} A_d^T\right)^{-1} \left(A_d \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \frac{\partial^2 F}{\partial \dot{Q}_j \partial u_i}\right).$$

The second derivatives with respect to two inputs are calculated by the following three equations.

$$\frac{\partial^2 f}{\partial u_j \partial u_i} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \left[A_d^T \frac{\partial^2 \lambda}{\partial u_j \partial u_i} + \frac{\partial^2 F}{\partial u_j \partial u_i}\right]$$

$$\frac{\partial^2 \lambda}{\partial u_j \partial u_i} = -\left(A_d \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} A_d^T\right)^{-1} \left(A_d \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1} \frac{\partial^2 F}{\partial u_j \partial u_i}\right)$$

The remaining second derivatives are zero.

$$\frac{\partial^2 f}{\partial \dot{Q}_j \partial \ddot{\rho}_i} = \frac{\partial^2 \lambda}{\partial \dot{Q}_j \partial \ddot{\rho}_i} = 0$$

$$\frac{\partial^2 f}{\partial \ddot{\rho}_j \partial \ddot{\rho}_i} = \frac{\partial^2 \lambda}{\partial \ddot{\rho}_j \partial \ddot{\rho}_i} = 0$$

$$\frac{\partial^2 f}{\partial \ddot{\rho}_j \partial u_i} = \frac{\partial^2 \lambda}{\partial \ddot{\rho}_j \partial u_i} = 0$$

A.2. Discrete Dynamics

We consider a discrete time mechanical system with a set of dynamic configuration variables q and kinematic configuration variables ρ. In Sec. 4.3, the state representation was $x_k = \begin{bmatrix} q_k \\ p_k \end{bmatrix}$. With the addition of the kinematic variables, the state must be increased. However, there is associated momentum with the kinematic configuration variables. Instead we define a new kinematic velocity $v_k = \frac{q_k - q_{k-1}}{t_k - t_{k-1}}$. The state space representation is then:

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

Adding the kinematic velocity to the state makes it easy to penalize large velocities in optimal control problems.

The input is augmented with the new kinematic input:

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

where $\bar{u}_k$ has been used to distinguish the state input vector $\bar{u}_k$ from the force input vector u_k. Note that in discrete time, the kinematic input is the value of the kinematic configuration variables at the next time step.

This section uses a modified slot derivative notation. In addition to the standard form of $D_n f(x, y, z)$, a superscript is added so that $D_n^i f(x, y, z)$ indicates the derivative of $f(x, y, z)$ with respect to the i-th component of the n-th argument. In many equations, the slot derivative will not use a superscript even though we only mean the derivative with respect to the dynamic part of the configuration variable rather than the entire configuration vector. This is ambigiuous but usually obvious in context.

We continue to use the convention from Sec. A.1 of using Q to represent the combined dynamic and kinematic state (i.e, $Q_k = \begin{bmatrix} q_k \\ \rho_k \end{bmatrix}$

Section A.2.1 provides equations for the derivatives of the discrete Lagrangian. The discrete force approximation and derivatives are discussed in Sec. A.2.2. The discrete Euler-Lagrange equations for a constrained, mixed dynamic-kinematic variational integrator are provided in Sec. A.2.4. The first- and second-order linearizations of the variational integrator are given in Sec. A.2.5 and Sec. A.2.6.

A.2.1. Discrete Lagrangian

The midpoint approximation (A.2.1) leads to a second-order variational integrator. It is convenient to abbreviate the discrete Lagrangian as $L_k = L_d(q_{k-1}, q_k, t_{k-1}, t_k)$ where k is the discrete time. When used with slot derivative notation, L_k is considered to have the same arguments as L_d.

$$L_d(q_{k-1}, q_k, t_{k-1}, t_k) = (t_k - t_{k-1}) L\left(\frac{q_{k-1} + q_k}{2}, \frac{q_k - q_{k-1}}{t_k - t_{k-1}}\right)$$

Since derivatives of the discrete Lagrangian are the same whether a configuration variable is dynamic or kinematic, we do not differentiate the two here.

The first derivatives of the discrete Lagrangian are:

$$\begin{aligned} D_1^\ell L_k &= \tfrac{1}{2}(t_k - t_{k-1})\frac{\partial L}{\partial q_\ell} - \frac{\partial L}{\partial \dot{q}_\ell} \\ D_2^\ell L_k &= \tfrac{1}{2}(t_k - t_{k-1})\frac{\partial L}{\partial q_\ell} + \frac{\partial L}{\partial \dot{q}_\ell}. \end{aligned}$$

The second derivatives are:

$$\begin{aligned} D_1^i D_1^\ell L_k &= \tfrac{1}{4}(t_k - t_{k-1})\frac{\partial^2 L}{\partial q_i \partial q_\ell} - \tfrac{1}{2}\left(\frac{\partial^2 L}{\partial q_\ell \partial \dot{q}_i} + \frac{\partial^2 L}{\partial q_i \partial \dot{q}_\ell}\right) + \frac{1}{t_k - t_{k-1}}\frac{\partial^2 L}{\partial \dot{q}_i \partial \dot{q}_\ell} \\ D_1^i D_2^\ell L_k &= \tfrac{1}{4}(t_k - t_{k-1})\frac{\partial^2 L}{\partial q_i \partial q_\ell} - \tfrac{1}{2}\frac{\partial^2 L}{\partial q_\ell \partial \dot{q}_i} + \tfrac{1}{2}\frac{\partial^2 L}{\partial q_i \partial \dot{q}_\ell} - \frac{1}{t_k - t_{k-1}}\frac{\partial^2 L}{\partial \dot{q}_i \partial \dot{q}_\ell} \\ D_2^i D_2^\ell L_k &= \tfrac{1}{4}(t_k - t_{k-1})\frac{\partial^2 L}{\partial q_i \partial q_\ell} + \tfrac{1}{2}\left(\frac{\partial^2 L}{\partial \dot{q}_i \partial q_\ell} + \frac{\partial^2 L}{\partial q_i \partial \dot{q}_\ell}\right) + \frac{1}{t_k - t_{k-1}}\frac{\partial^2 L}{\partial \dot{q}_i \partial \dot{q}_\ell} \end{aligned}$$

The third derivatives are:

$$
\begin{aligned}
D_1^j D_1^i D_1^\ell L_k &= \tfrac{1}{8}(t_k - t_{k-1})\frac{\partial^3 L}{\partial q_j \partial q_i \partial q_\ell} - \tfrac{1}{4}\left(\frac{\partial^3 L}{\partial q_i \partial q_\ell \partial \dot{q}_j} + \frac{\partial^3 L}{\partial q_j \partial q_\ell \partial \dot{q}_i} + \frac{\partial^3 L}{\partial q_j \partial q_i \partial \dot{q}_\ell}\right) \\
&\qquad + \frac{1}{2(t_k - t_{k-1})}\left(\frac{\partial^3 L}{\partial q_\ell \partial \dot{q}_j \partial \dot{q}_i} + \frac{\partial^3 L}{\partial q_i \partial \dot{q}_j \partial \dot{q}_\ell} + \frac{\partial^3 L}{\partial q_j \partial \dot{q}_i \partial \dot{q}_\ell}\right) \\
D_1^j D_1^i D_2^\ell L_k &= \tfrac{1}{8}(t_k - t_{k-1})\frac{\partial^3 L}{\partial q_j \partial q_i \partial q_\ell} + \tfrac{1}{4}\left(\frac{\partial^3 L}{\partial q_j \partial q_i \partial \dot{q}_\ell} - \frac{\partial^3 L}{\partial q_i \partial q_\ell \partial \dot{q}_j} - \frac{\partial^3 L}{\partial q_j \partial q_\ell \partial \dot{q}_i}\right) \\
&\qquad + \frac{1}{2(t_k - t_{k-1})}\left(\frac{\partial^3 L}{\partial q_\ell \partial \dot{q}_j \partial \dot{q}_i} - \frac{\partial^3 L}{\partial q_i \partial \dot{q}_j \partial \dot{q}_\ell} - \frac{\partial^3 L}{\partial q_j \partial \dot{q}_i \partial \dot{q}_\ell}\right) \\
D_1^j D_2^i D_2^\ell L_k &= \tfrac{1}{8}(t_k - t_{k-1})\frac{\partial^3 L}{\partial q_j \partial q_i \partial q_\ell} + \tfrac{1}{4}\left(\frac{\partial^3 L}{\partial q_j \partial q_\ell \partial \dot{q}_i} + \frac{\partial^3 L}{\partial q_j \partial q_i \partial \dot{q}_\ell} - \frac{\partial^3 L}{\partial q_i \partial q_\ell \partial \dot{q}_j}\right) \\
&\qquad + \frac{1}{2(t_k - t_{k-1})}\left(\frac{\partial^3 L}{\partial q_j \partial \dot{q}_i \partial \dot{q}_\ell} - \frac{\partial^3 L}{\partial q_\ell \partial \dot{q}_j \partial \dot{q}_i} - \frac{\partial^3 L}{\partial q_i \partial \dot{q}_j \partial \dot{q}_\ell}\right) \\
D_2^j D_2^i D_2^\ell L_k &= \tfrac{1}{8}(t_k - t_{k-1})\frac{\partial^3 L}{\partial q_j \partial q_i \partial q_\ell} + \tfrac{1}{4}\left(\frac{\partial^3 L}{\partial q_i \partial q_\ell \partial \dot{q}_j} + \frac{\partial^3 L}{\partial q_j \partial q_\ell \partial \dot{q}_i} + \frac{\partial^3 L}{\partial q_j \partial q_i \partial \dot{q}_\ell}\right) \\
&\qquad + \frac{1}{2(t_k - t_{k-1})}\left(\frac{\partial^3 L}{\partial q_\ell \partial \dot{q}_j \partial \dot{q}_i} + \frac{\partial^3 L}{\partial q_i \partial \dot{q}_j \partial \dot{q}_\ell} + \frac{\partial^3 L}{\partial q_j \partial \dot{q}_i \partial \dot{q}_\ell}\right)
\end{aligned}
$$

Each of the above equations is evaluated numerically from calculated values of the continuous Lagrangian. In turn, those are calculated from the tree representation.

A.2.2. Discrete Forcing

A similar midpoint approximation is used for the discrete forcing in (A.2.1). The discrete forcing is abbreviated as $f_k^\pm = f_d(q_{k-1}, q_k, u_{k-1}, t_{k-1}, t_k)$ where k is the discrete time. When used with slot derivative notation, f_k is considered to have the same arguments as f_d.

$$f_d^-(q_{k-1}, q_k, u_{k-1}, t_{k-1}, t_k) = \tfrac{1}{2}(t_k - t_{k-1}) f\left(\frac{q_{k-1}+q_k}{2}, \frac{q_k - q_{k-1}}{t_k - t_{k-1}}, u_{k-1}, \frac{t_{k-1}+t_k}{2}\right) \tag{A.4}$$

$$f_d^+(q_{k-1}, q_k, u_{k-1}, t_{k-1}, t_k) = 0 \tag{A.5}$$

The first and second linearization equations are indifferent to the choice of the discrete approximation. If a different approximation is used, only the following derivatives in this section need to be updated.

The first derivatives of the discrete forces are

$$\begin{aligned}
D_1^i f_k^- &= (t_k - t_{k-1}) \frac{1}{4}\frac{\partial f}{\partial q_i} - \frac{1}{2}\frac{\partial f}{\partial \dot{q}_i} \\
D_2^i f_k^- &= (t_k - t_{k-1}) \frac{1}{4}\frac{\partial f}{\partial q_i} + \frac{1}{2}\frac{\partial f}{\partial \dot{q}_i} \\
D_3^i f_k^- &= (t_k - t_{k-1}) \frac{1}{2}\frac{\partial f}{\partial u_i}
\end{aligned}$$

The second derivatives are:

$$\begin{aligned}
D_1^j D_1^i f_k^- &= (t_k - t_{k-1}) \frac{1}{8}\frac{\partial^2 f}{\partial q_j \partial q_i} - \frac{1}{4}\left(\frac{\partial^2 f}{\partial q_i \partial \dot{q}_j} + \frac{\partial^2 f}{\partial q_j \partial \dot{q}_i}\right) + \frac{1}{2(t_k - t_{k-1})}\frac{\partial^2 f}{\partial \dot{q}_j \partial \dot{q}_i} \\
D_1^j D_2^i f_k^- &= (t_k - t_{k-1}) \frac{1}{8}\frac{\partial^2 f}{\partial q_j \partial q_i} + \frac{1}{4}\left(\frac{\partial^2 f}{\partial q_j \partial \dot{q}_i} - \frac{\partial^2 f}{\partial q_i \partial \dot{q}_j}\right) - \frac{1}{2(t_k - t_{k-1})}\frac{\partial^2 f}{\partial \dot{q}_j \partial \dot{q}_i} \\
D_2^j D_2^i f_k^- &= (t_k - t_{k-1}) \frac{1}{8}\frac{\partial^2 f}{\partial q_j \partial q_i} + \frac{1}{4}\left(\frac{\partial^2 f}{\partial q_i \partial \dot{q}_j} + \frac{\partial^2 f}{\partial q_j \partial \dot{q}_i}\right) + \frac{1}{2(t_k - t_{k-1})}\frac{\partial^2 f}{\partial \dot{q}_j \partial \dot{q}_i} \\
D_1^j D_3^i f_k^- &= (t_k - t_{k-1}) \frac{1}{4}\frac{\partial^2 f}{\partial q_j \partial u_i} - \frac{1}{2}\frac{\partial^2 f}{\partial \dot{q}_j \partial u_i} \\
D_2^j D_3^i f_k^- &= (t_k - t_{k-1}) \frac{1}{4}\frac{\partial^2 f}{\partial q_j \partial u_i} + \frac{1}{2}\frac{\partial^2 f}{\partial \dot{q}_j \partial u_i}
\end{aligned}$$

$$D_3^j D_3^i f_k^- = (t_k - t_{k-1})\frac{1}{2}\frac{\partial^2 f}{\partial u_j \partial u_i}$$

These derivatives are all calculated numerically from the derivatives of the continuous forcing.

A.2.3. Discrete Constraints

The variational integrator works directly with holonomic constraints. They are not approximated or replaced with non-holonomic constraints. Equations for the derivatives of $h(q)$ are provided by each individual type of constraint. In the following sections, we use the abbreviation $h_k = h(q_k)$.

A.2.4. Discrete Dynamics

The discrete Euler-Lagrange equations for the qconstrained, mixed kinematic-dynamic variational integrator are:

$$p_k + D_1 L_{k+1} + f_{k+1}^- - Dh_k^T \cdot \lambda_k = 0 \tag{A.6a}$$

$$h_{k+1} = 0 \tag{A.6b}$$

$$p_{k+1} = D_2 L_{k+1} + f_{k+1}^+. \tag{A.6c}$$

The next state q_{k+1} and p_{k+1} are found in two steps. A numeric root-finding algorithm solves (A.6a) and (A.6b) to find q_{k+1} and λ_k. We define the function to be solved as:

$$f(q_{k+1}, \lambda_k) = \begin{bmatrix} p_k + D_1 L_{k+1} + f_{k+1}^- - Dh_k^T \cdot \lambda_k \\ h_{k+1} \end{bmatrix} = 0. \tag{A.7}$$

Non-linear root finding algorithms typically use the derivative of equation to find roots. The derivative of (A.7) is:

$$Df(q_{k+1}, \lambda_k) = \begin{bmatrix} M_{k+1} & -Dh_k^T \\ Dh_{k+1} & 0 \end{bmatrix}$$

where M_{k+1} is defined as:

$$M_k = \left[D_2 D_1 L_k + D_2 f_k^-\right].$$

Once q_{k+1} is known, p_{k+1} is calculated explictly from (A.6c).

A.2.5. First Derivative

The first linearization of the state form in Sec. A.2 is:

$$\begin{bmatrix} \delta q_{k+1} \\ \delta \rho_{k+1} \\ \delta p_{k+1} \\ \delta v_{k+1} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial q_{k+1}}{\partial q_k} & \dfrac{\partial q_{k+1}}{\partial \rho_k} & \dfrac{\partial q_{k+1}}{\partial p_k} & 0 \\ 0 & 0 & 0 & 0 \\ \dfrac{\partial p_{k+1}}{\partial q_k} & \dfrac{\partial p_{k+1}}{\partial \rho_k} & \dfrac{\partial p_{k+1}}{\partial p_k} & 0 \\ 0 & -\dfrac{1}{t_{k+1} - t_k} & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta q_k \\ \delta \rho_k \\ \delta p_k \\ \delta v_k \end{bmatrix} + \begin{bmatrix} \dfrac{\partial q_{k+1}}{\partial u_k} & \dfrac{\partial q_{k+1}}{\partial \rho_{k+1}} \\ 0 & \mathcal{I} \\ \dfrac{\partial p_{k+1}}{\partial u_k} & \dfrac{\partial p_{k+1}}{\partial \rho_{k+1}} \\ 0 & \dfrac{1}{t_{k+1} - t_k} \end{bmatrix} \begin{bmatrix} \delta u_k \\ \delta \rho_{k+1} \end{bmatrix}.$$

The first derivatives are calculated using the approach described in Sec. 4.4. We consider the derivatives of the future state q_2 and the constraint forces λ_k first.

For the remainder of this section, we let $k = 1$ and $k + 1 = 2$ to simplify these already complex equations. However the equations still apply to any value of k.

The derivatives with respect to the combined current configuration are:

$$\frac{\partial q_2}{\partial Q_1^i} = M_2^{-1}\left[Dh_1^T \cdot \frac{\partial \lambda_1}{\partial Q_1^i} + C_{Q_1^i}\right]$$
$$\frac{\partial \lambda_1}{\partial Q_1^i} = -\left[Dh_2 M_2^{-1} Dh_1^T\right]^{-1} Dh_2 M_2^{-1} C_{Q_1^i}$$
$$C_{Q_1^i} = D^i Dh_1^T \cdot \lambda_1 - D_1^i D_1 L_2 - D_1^i f_2^-$$

The first derivatives with respect to the current discrete momentum are:

$$\frac{\partial q_2}{\partial p_1^i} = M_2^{-1}\left[Dh_1^T \frac{\partial \lambda_1}{\partial p_1^i} + C_{p_1^i}\right]$$
$$\frac{\partial \lambda_1}{\partial p_1^i} = -\left[Dh_2 M_2^{-1} Dh_1^T\right]^{-1} Dh_2 M_2^{-1} C_{p_1^i}$$
$$C_{p_1^i} = -\frac{\partial p_1}{\partial p_1^i}$$

Derivatives with respect to the force inputs are calculated by the following equations.

$$\frac{\partial q_2}{\partial u_1^i} = M_2^{-1}\left[Dh_1^T \frac{\partial \lambda_1}{\partial u_1^i} + C_{u_1^i}\right]$$
$$\frac{\partial \lambda_1}{\partial u_1^i} = -\left[Dh_2 M_2^{-1} Dh_1^T\right]^{-1} Dh_2 M_2^{-1} C_{u_1^i}$$
$$C_{u_1^i} = -D_3^i f_2^-$$

The first derivatives with respect to the kinematic inputs are:

$$\begin{aligned}
\frac{\partial q_2}{\partial \rho_2^i} &= M_2^{-1}\left[Dh_1^T \frac{\partial \lambda_1}{\partial \rho_2^i} + C_{\rho_2^i}\right] \\
\frac{\partial \lambda_1}{\partial \rho_2^i} &= -\left[Dh_2 M_2^{-1} Dh_1^T\right]^{-1}\left[Dh_2 M_2^{-1} C_{\rho_2^i} + D^i h_2\right] \\
C_{\rho_2^i} &= -D_2^i D_1 L_2 - D_2^i f_2^-
\end{aligned}$$

A.2.5.1. First Derivatives of p_2. First derivatives of the future discrete momentum are found by directly differentiating (A.6c).

$$\begin{aligned}
\frac{\partial p_2}{\partial Q_1^i} &= D_1^i D_2 L_2 + D_1^i f_2^+ + \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial q_2}{\partial Q_1^i} \\
\frac{\partial p_2}{\partial p_1^i} &= \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial q_2}{\partial p_1^i} \\
\frac{\partial p_2}{\partial u_1^i} &= D_3^i f_2^+ + \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial q_2}{\partial u_1^i} \\
\frac{\partial p_2}{\partial \rho_2^i} &= D_2^i D_2 L_2 + D_2^i f_2^+ + \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial q_2}{\partial \rho_2^i}
\end{aligned}$$

Each of these requires that the derivatives of the future configuration are calculated first.

A.2.6. Second Derivative

This section contains the second derivatives of the variational integrator (A.6). These equations are calculated using the method described in Sec. 4.4 and apply to any discrete Lagrangian and discrete forcing approximations. The derivatives with respect to the next configuration are given here. Derivatives with respect to the next discrete momentum are listed in Sec. A.2.6.1.

The second derivative with respect to two (discrete or kinematic) configuration variables is:

$$\frac{\partial^2 q_2}{\partial Q_1^j \partial Q_1^i} = M_2^{-1}\left[D^2 h_1^T \frac{\partial^2 \lambda_1}{\partial Q_1^j \partial Q_1^i} + C_{Q_1^i Q_1^j}\right]$$

$$\frac{\partial^2 \lambda_1}{\partial Q_1^j \partial Q_1^i} = -\left[Dh_2 M_2^{-1} D^2 h_1^T\right]^{-1}\left[D^2 h_2 \circ \left(\frac{\partial q_2}{\partial Q_1^j}, \frac{\partial q_2}{\partial Q_1^i}\right) + Dh_2 M_2^{-1} C_{Q_1^i Q_1^j}\right]$$

$$\begin{aligned} C_{Q_1^i Q_1^j} = {} & D^j D^i D h_1^T \cdot \lambda_1 + D^i D h_1^T \frac{\partial \lambda_1}{\partial Q_1^j} + D^j D h_1^T \frac{\partial \lambda_1}{\partial Q_1^i} \\ & - D_1^j D_1^i D_1 L_2 - D_1^j D_1^i f_2^- - \left[D_1^i D_2 D_1 L_2 + D_1^i D_2 f_2^-\right] \frac{\partial q_2}{\partial Q_1^j} \\ & - \left[D_1^j D_2 D_1 L_2 + D_1^j D_2 f_2^-\right] \frac{\partial q_2}{\partial Q_1^i} - \left[D_2 D_2 D_1 L_2 + D_2 D_2 f_2^-\right] \circ \left(\frac{\partial q_2}{\partial Q_1^j}, \frac{\partial q_2}{\partial Q_1^i}\right). \end{aligned}$$

The second derivative is found by calculating $C_{Q_1^i Q_1^j}$, using that result to calculate $\frac{\partial^2 \lambda_1}{\partial Q_1^j \partial Q_1^i}$, and using both values to find the second derivative $\frac{\partial^2 q_2}{\partial Q_1^j \partial Q_1^i}$. The same process is used for the remaining derivatives as well.

The second derivative with respect to a configuration variable and discrete momentum variable is:

$$\frac{\partial^2 q_2}{\partial Q_1^j \partial p_1^i} = M_2^{-1}\left[D h_1^T \frac{\partial^2 \lambda_1}{\partial Q_1^j \partial p_1^i} + C_{Q_1^j p_1^i}\right]$$

$$\frac{\partial^2 \lambda_1}{\partial Q_1^j \partial p_1^i} = -\left[Dh_2 M_2^{-1} D h_1^T\right]^{-1}\left[D^2 h_2 \circ \left(\frac{\partial q_2}{\partial Q_1^j}, \frac{\partial q_2}{\partial p_1^i}\right) + Dh_2 M_2^{-1} C_{Q_1^j p_1^i}\right]$$

$$C_{Q_1^j p_1^i} = D^j D h_1^T \frac{\partial \lambda_1}{\partial p_1^i} - \left[D_1^j D_2 D_1 L_2 + D_1^j D_2 f_2^-\right] \frac{\partial q_2}{\partial p_1^i} - \left[D_2 D_2 D_1 L_2 + D_2 D_2 f_2^-\right] \circ \left(\frac{\partial q_2}{\partial Q_1^j}, \frac{\partial q_2}{\partial p_1^i}\right).$$

The second derivative with respect to two discrete momentum variables is:

$$\frac{\partial^2 q_2}{\partial p_1^j \partial p_1^i} = M_2^{-1}\left[Dh_1^T \frac{\partial^2 \lambda_1}{\partial p_1^j \partial p_1^i} + C_{p_1^j p_1^i}\right]$$

$$\frac{\partial^2 \lambda_1}{\partial p_1^j \partial p_1^i} = -\left[Dh_2 M_2^{-1} Dh_1^T\right]^{-1}\left[DDh_2 \circ \left(\frac{\partial q_2}{\partial p_1^j}, \frac{\partial q_2}{\partial p_1^i}\right) + Dh_2 M_2^{-1} C_{p_1^j p_1^i}\right]$$

$$C_{p_1^j p_1^i} = -\left[D_2 D_2 D_1 L_2 + D_2 D_2 f_2^-\right] \circ \left(\frac{\partial q_2}{\partial p_1^j}, \frac{\partial q_2}{\partial p_1^i}\right)$$

The second derivative with respect to a configuration variable and a force input is:

$$\frac{\partial^2 q_2}{\partial Q_1^j \partial u_1^i} = M_2^{-1}\left[Dh_1^T \frac{\partial^2 \lambda_1}{\partial Q_1^j \partial u_1^i} + C_{Q_1^j u_1^i}\right]$$

$$\frac{\partial^2 \lambda_1}{\partial Q_1^j \partial u_1^i} = -\left[Dh_2 M_2^{-1} Dh_1^T\right]^{-1}\left[DDh_2 \circ \left(\frac{\partial q_2}{\partial Q_1^j}, \frac{\partial q_2}{\partial u_1^i}\right) + Dh_2 M_2^{-1} C_{Q_1^j u_1^i}\right]$$

$$\begin{aligned} C_{Q_1^j u_1^i} &= D^j Dh_1^T \frac{\partial \lambda_1}{\partial u_1^i} - D_1^j D_3^i f_2^- - D_2 D_3^i f_2^- \frac{\partial q_2}{\partial Q_1^j} \\ &\quad - \left[D_1^j D_2 D_1 L_2 + D_1^j D_2 f_2^-\right] \frac{\partial q_2}{\partial u_1^i} - \left[D_2 D_2 D_1 L_2 + D_2 D_2 f_2^-\right] \circ \left(\frac{\partial q_2}{\partial Q_1^j}, \frac{\partial q_2}{\partial u_1^i}\right) \end{aligned}$$

The second derivative with respect to a discrete momentum and a force input is:

$$\frac{\partial^2 q_2}{\partial p_1^j \partial u_1^i} = M_2^{-1}\left[Dh_1^T \frac{\partial^2 \lambda_1}{\partial p_1^j \partial u_1^i} + C_{p_1^j u_1^i}\right]$$

$$\frac{\partial^2 \lambda_1}{\partial p_1^j \partial u_1^i} = -\left[Dh_2 M_2^{-1} Dh_1^T\right]^{-1}\left[DDh_2 \circ \left(\frac{\partial q_2}{\partial p_1^j}, \frac{\partial q_2}{\partial u_1^i}\right) + Dh_2 M_2^{-1} C_{p_1^j u_1^i}\right]$$

$$C_{p_1^j u_1^i} = -D_2 D_3^i f_2^- \frac{\partial q_2}{\partial p_1^j} - \left[D_2 D_2 D_1 L_2 + D_2 D_2 f_2^-\right] \circ \left(\frac{\partial q_2}{\partial p_1^j}, \frac{\partial q_2}{\partial u_1^i}\right)$$

The second derivative with respect to two force inputs is:

$$\frac{\partial^2 q_2}{\partial u_1^j \partial u_1^i} = M_2^{-1}\left[Dh_1^T \frac{\partial^2 \lambda_1}{\partial u_1^j \partial u_1^i} + C_{u_1^j u_1^i}\right]$$

$$\frac{\partial^2 \lambda_1}{\partial u_1^j \partial u_1^i} = -\left[Dh_2 M_2^{-1} Dh_1^T\right]^{-1}\left[DDh_2 \circ \left(\frac{\partial q_2}{\partial u_1^j}, \frac{\partial q_2}{\partial u_1^i}\right) + Dh_2 M_2^{-1} C_{u_1^j u_1^i}\right]$$

$$C_{u_1^j u_1^i} = -D_3^j D_3^i f_2^- - D_3^i D_2 f_2^- \frac{\partial q_2}{\partial u_1^j} - D_3^j D_2 f_2^- \frac{\partial q_2}{\partial u_1^i} - \left[D_2 D_2 D_1 L_2 + D_2 D_2 f_2^-\right] \circ \left(\frac{\partial q_2}{\partial u_1^j}, \frac{\partial q_2}{\partial u_1^i}\right)$$

The second derivative with respect to a configuration variable and a kinematic input is:

$$\frac{\partial^2 q_2}{\partial Q_1^j \partial \rho_2^i} = M_2^{-1}\left[Dh_1^T \frac{\partial^2 \lambda_1}{\partial Q_1^j \partial \rho_2^i} + C_{Q_1^j \rho_2^i}\right]$$

$$\frac{\partial^2 \lambda_1}{\partial Q_1^j \partial \rho_2^i} = -\left[Dh_2 M_2^{-1} Dh_1^T\right]^{-1}\left[D^2 h_2 \circ \left(\frac{\partial q_2}{\partial Q_1^j}, \frac{\partial q_2}{\partial \rho_2^i}\right) + D^i Dh_2 \frac{\partial q_2}{\partial Q_1^j} + Dh_2 M_2^{-1} C_{Q_1^j \rho_2^i}\right]$$

$$\begin{aligned} C_{Q_1^j \rho_2^i} = {} & D^j Dh_1^T \frac{\partial \lambda_1}{\partial \rho_2^i} - D_1^j D_2^i D_1 L_2 - D_1^j D_2^i f_2^- - \left[D_2^i D_2 D_1 L_2 + D_2^i D_2 f_2^-\right] \frac{\partial q_2}{\partial Q_1^j} \\ & - \left[D_1^j D_2 D_1 L_2 + D_1^j D_2 f_2^-\right] \frac{\partial q_2}{\partial \rho_2^i} - \left[D_2 D_2 D_1 L_2 + D_2 D_2 f_2^-\right] \circ \left(\frac{\partial q_2}{\partial Q_1^j}, \frac{\partial q_2}{\partial \rho_2^i}\right) \end{aligned}$$

The second derivative with respect to a discrete momentum and kinematic input is:

$$\frac{\partial^2 q_2}{\partial p_1^j \partial \rho_2^i} = M_2^{-1}\left[Dh_1^T \frac{\partial^2 \lambda_1}{\partial p_1^j \partial \rho_2^i} + C_{p_1^j \rho_2^i}\right]$$

$$\frac{\partial^2 \lambda_1}{\partial p_1^j \partial \rho_2^i} = -\left[Dh_2 M_2^{-1} Dh_1^T\right]^{-1}\left[D^i Dh_2 \frac{\partial q_2}{\partial p_1^j} + D^2 h_2 \circ \left(\frac{\partial q_2}{\partial p_1^j}, \frac{\partial q_2}{\partial \rho_2^i}\right) + Dh_2 M_2^{-1} C_{p_1^j \rho_2^i}\right]$$

$$C_{p_1^j \rho_2^i} = -\left[D_2^i D_2 D_1 L_2 + D_2^i D_2 f_2^-\right] \frac{\partial q_2}{\partial p_1^j} - \left[D_2 D_2 D_1 L_2 + D_2 D_2 f_2^-\right] \circ \left(\frac{\partial q_2}{\partial p_1^j}, \frac{\partial q_2}{\partial \rho_2^i}\right)$$

The second derivative with respect to a force input and kinematic input is:

$$\frac{\partial^2 q_2}{\partial u_1^j \partial \rho_2^i} = M_2^{-1}\left[Dh_1^T \frac{\partial^2 \lambda_1}{\partial u_1^j \partial \rho_2^i} + C_{u_1^j \rho_2^i}\right]$$

$$\frac{\partial^2 \lambda_1}{\partial u_1^j \partial \rho_2^i} = -\left[Dh_2 M_2^{-1} Dh_1^T\right]^{-1}\left[D^i Dh_2 \frac{\partial q_2}{\partial u_1^j} + D^2 h_2 \circ \left(\frac{\partial q_2}{\partial u_1^j}, \frac{\partial q_2}{\partial \rho_2^i}\right) + Dh_2 M_2^{-1} C_{u_1^j \rho_2^i}\right]$$

$$\begin{aligned} C_{u_1^j \rho_2^i} = {} & -D_3^j D_2^i f_2^- - D_2 D_3^j f_2^- \frac{\partial q_2}{\partial \rho_2^i} - \left[D_2^i D_2 D_1 L_2 + D_2^i D_2 f_2^-\right] \frac{\partial q_2}{\partial u_1^j} \\ & - \left[D_2 D_2 D_1 L_2 + D_2 D_2 f_2^-\right] \circ \left(\frac{\partial q_2}{\partial u_1^j}, \frac{\partial q_2}{\partial \rho_2^i}\right) \end{aligned}$$

The second derivative with respect to two kinematic inputs are:

$$\frac{\partial^2 q_2}{\partial \rho_2^j \partial \rho_2^i} = M_2^{-1}\left[Dh_1^T \frac{\partial^2 \lambda_1}{\partial \rho_2^j \partial \rho_2^i} + C_{\rho_2^j \rho_2^i}\right]$$

$$\begin{aligned} \frac{\partial^2 \lambda_1}{\partial \rho_2^j \partial \rho_2^i} = {} & -\left[Dh_2 M_2^{-1} Dh_1^T\right]^{-1}\Bigg[D^j D^i h_2 + D^i Dh_2 \frac{\partial q_2}{\partial \rho_2^j} + D^j Dh_2 \frac{\partial q_2}{\partial \rho_2^i} \\ & + D^2 h_2 \circ \left(\frac{\partial q_2}{\partial \rho_2^j}, \frac{\partial q_2}{\partial \rho_2^i}\right) + Dh_2 M_2^{-1} C_{\rho_2^j \rho_2^i}\Bigg] \end{aligned}$$

$$\begin{aligned} C_{\rho_2^j \rho_2^i} = {} & -D_2^j D_2^i D_1 L_2 - D_2^j D_2^i f_2^- - \left[D_2 D_2 D_1 L_2 + D_2 D_2 f_2^-\right] \circ \left(\frac{\partial q_2}{\partial \rho_2^j}, \frac{\partial q_2}{\partial \rho_2^i}\right) \\ & - \left[D_2^j D_2 D_1 L_2 + D_2^j D_2 f_2^-\right] \frac{\partial q_2}{\partial \rho_2^i} - \left[D_2^i D_2 D_1 L_2 + D_2^i D_2 f_2^-\right] \frac{\partial q_2}{\partial \rho_2^j} \end{aligned}$$

Once the second derivatives of the updated configuration variables are known, the derivatives of the updated discrete momentum are calculated.

A.2.6.1. Second Derivatives of Discrete Momentum. Second derivatives of the discrete momentum are found directly from (A.6c). For the discrete force approximation given in Sec. A.2.2, the derivatives of f_2^+ are always zero and can be removed from the equation.

The following equation calculate the second derivative of the updated momentum with respect to a configuration variable and either another configuration variable, a discrete momentum, a force input, or a kinematic input, in that order.

$$\begin{aligned}\frac{\partial^2 p_2}{\partial Q_1^j \partial Q_1^i} &= D_1^j D_1^i D_2 L_2 + D_1^j D_1^i f_2^+ \\ &\quad + \left[D_1^i D_2 D_2 L_2 + D_1^i D_2 f_2^+\right] \frac{\partial q_2}{\partial Q_1^j} + \left[D_1^j D_2 D_2 L_2 + D_1^j D_2 f_2^+\right] \frac{\partial q_2}{\partial Q_1^i} \\ &\quad + \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial^2 q_2}{\partial Q_1^j \partial Q_1^i} + \left[D_2 D_2 D_2 L_2 + D_2 D_2 f_2^+\right] \circ \left(\frac{\partial q_2}{\partial Q_1^j}, \frac{\partial q_2}{\partial Q_1^i}\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 p_2}{\partial Q_1^j \partial p_1^i} &= \left[D_1^j D_2 D_2 L_2 + D_1^j D_2 f_2^+\right] \frac{\partial q_2}{\partial p_1^i} + \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial^2 q_2}{\partial Q_1^j \partial p_1^i} \\ &\quad + \left[D_2 D_2 D_2 L_2 + D_2 D_2 f_2^+\right] \circ \left(\frac{\partial q_2}{\partial Q_1^j}, \frac{\partial q_2}{\partial p_1^i}\right).\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 p_2}{\partial Q_1^j \partial u_1^i} &= D_1^j D_3^i f_2^+ + D_2 D_3^i f_2^+ \frac{\partial q_2}{\partial Q_1^j} + \left[D_2 D_2 D_2 L_2 + D_2 D_2 f_2^+\right] \circ \left(\frac{\partial q_2}{\partial Q_1^j}, \frac{\partial q_2}{\partial u_1^i}\right) \\ &\quad + \left[D_1^j D_2 D_2 L_2 + D_1^j D_2 f_2^+\right] \frac{\partial q_2}{\partial u_1^i} + \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial^2 q_2}{\partial Q_1^j \partial u_1^i}\end{aligned}$$

$$\frac{\partial^2 p_2}{\partial Q_1^j \partial \rho_2^i} = D_1^j D_2^i D_2 L_2 + D_1^j D_2^i f_2^+$$
$$+ \left[D_2^i D_2 D_2 L_2 + D_2^i D_2 f_2^+\right] \frac{\partial q_2}{\partial Q_1^j} + \left[D_1^j D_2 D_2 L_2 + D_1^j D_2 f_2^+\right] \frac{\partial q_2}{\partial \rho_2^i}$$
$$+ \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial^2 q_2}{\partial Q_1^j \partial \rho_2^i} + \left[D_2 D_2 D_2 L_2 + D_2 D_2 f_2^+\right] \circ \left(\frac{\partial q_2}{\partial Q_1^j}, \frac{\partial q_2}{\partial \rho_2^i}\right)$$

Second derivatives with respect to the discrete momentum are calculated by the following equations. This includes the second derivative with respect to another discrete momentum, a force input, and a kinematic input.

$$\frac{\partial^2 p_2}{\partial p_1^j \partial p_1^i} = \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial^2 q_2}{\partial p_1^j \partial p_1^i} + \left[D_2 D_2 D_2 L_2 + D_2 D_2 f_2^+\right] \circ \left(\frac{\partial q_2}{\partial p_1^j}, \frac{\partial q_2}{\partial p_1^i}\right).$$

$$\frac{\partial^2 p_2}{\partial p_1^j \partial u_1^i} = D_2 D_3^i f_2^+ \frac{\partial q_2}{\partial p_1^j} + \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial^2 q_2}{\partial p_1^j \partial u_1^i} + \left[D_2 D_2 D_2 L_2 + D_2 D_2 f_2^+\right] \circ \left(\frac{\partial q_2}{\partial p_1^j}, \frac{\partial q_2}{\partial u_1^i}\right)$$

$$\frac{\partial^2 p_2}{\partial p_1^j \partial \rho_2^i} = \left[D_2 D_2^i D_2 L_2 + D_2 D_2^i f_2^+\right] \frac{\partial q_2}{\partial p_1^j} + \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial^2 q_2}{\partial p_1^j \partial \rho_2^i}$$
$$+ \left[D_2 D_2 D_2 L_2 + D_2 D_2 f_2^+\right] \circ \left(\frac{\partial q_2}{\partial p_1^j}, \frac{\partial q_2}{\partial \rho_2^i}\right)$$

Finally, the remaining second derivatives with respect to force inputs and kinematic inputs are calculated by the following.

$$\frac{\partial^2 p_2}{\partial u_1^j \partial u_1^i} = D_3^j D_3^i f_2^+ + D_2 D_3^i f_2^+ \frac{\partial q_2}{\partial u_1^j} + D_3^j D_2 f_2^+ \frac{\partial q_2}{\partial u_1^i}$$
$$+ \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial^2 q_2}{\partial u_1^j \partial u_1^i} + \left[D_2 D_2 D_2 L_2 + D_2 D_2 f_2^+\right] \circ \left(\frac{\partial q_2}{\partial u_1^j}, \frac{\partial q_2}{\partial u_1^i}\right)$$

$$\frac{\partial^2 p_2}{\partial u_1^j \partial \rho_2^i} = D_3^j D_2^i f_2^+ + D_3^j D_2 f_2^+ \frac{\partial q_2}{\partial \rho_2^i} + \left[D_2 D_2^i D_2 L_2 + D_2 D_2^i f_2^+\right] \frac{\partial q_2}{\partial u_1^j}$$
$$+ \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial^2 q_2}{\partial u_1^j \partial \rho_2^i} + \left[D_2 D_2 D_2 L_2 + D_2 D_2 f_2^+\right] \circ \left(\frac{\partial q_2}{\partial u_1^j}, \frac{\partial q_2}{\partial \rho_2^i}\right)$$

$$\frac{\partial^2 p_2}{\partial \rho_2^j \partial \rho_2^i} = D_2^j D_2^i D_2 L_2 + D_2^j D_2^i f_2^+$$
$$+ \left[D_2^i D_2 D_2 L_2 + D_2^i D_2 f_2^+\right] \frac{\partial q_2}{\partial \rho_2^j} + \left[D_2^j D_2 D_2 L_2 + D_2^j D_2 f_2^+\right] \frac{\partial q_2}{\partial \rho_2^i}$$
$$+ \left[D_2 D_2 L_2 + D_2 f_2^+\right] \frac{\partial^2 q_2}{\partial \rho_2^j \partial \rho_2^i} + \left[D_2 D_2 D_2 L_2 + D_2 D_2 f_2^+\right] \left(\frac{\partial q_2}{\partial \rho_2^j}, \frac{\partial q_2}{\partial \rho_2^i}\right)$$

APPENDIX B

Linear Systems

This appendix reviews important linear system theory required for the derivation of the trajectory optimization problem. The appendix is divided into continuous (Sec. B.1) and discrete (Sec. B.2) sections. Solutions to the dynamics of a continuous linear system are described in Sec. B.1.1. Solutions to the continuous LQR and LQ problems are given in Sec. B.1.2 and Sec. B.1.3, respectively.

The trajectories of a discrete linear system are discussed in Sec. B.2.1. The discrete LQR and LQ problems are described in Sec. B.2.2 and Sec. B.2.3.

B.1. Continuous Linear Systems

This section reviews results for time-varying continuous linear systems of the form:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t).$$

B.1.1. Continuous Linear System Trajectories

B.1.2. Continuous LQR Problem

The Linear Quadratic Regulator (LQR) problem is used to find a stabilizing feedback controller to implement a projection operator (6.10). In this sense, it is a tool to automatically calculate a feedback law for arbitrary trajectories of a system. The LQR solution is also the basis for solutions to the LQ problem, discussed in the next section.

Problem Statement: Find the control input $u(t)\ \forall\ t \in [t_0, t_f]$ that minimizes the cost:

$$V(x(t_0), u(\cdot), t_0) = \int_{t_0}^{t_f} \left[u^T(t)R(t)u(t) + x^T(t)Q(t)x(t)\right] \mathrm{d}t + x^T(t_f)Q_f x(t_f)$$

where

$$R(t) = R^T(t) > 0\ \forall\ t \in [t_0, t_f]$$

$$Q(t) = Q^T(t) \geq 0\ \forall\ t \in [t_0, t_f]$$

$$Q_f = Q_f^T \geq 0$$

$x(t_0)$ is known.

$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$

Solution: The optimal control $u^*(t)$ and optimal cost $V^*(x(t_0), t_0)$ are

$$u^*(t) = -\mathcal{K}(t)x(t)$$

$$V^*(x(t_0), t_0) = x^T(t_0)P(t_0)x(t_0)$$

where $P(t)$ is a symmetric time varying matrix satisfying a Ricatti equation:

$$P(t_f) = Q_f$$

$$-\dot{P}(t) = P(t)A(t) + A^T(t)P(t) - \mathcal{K}^T(t)R(t)\mathcal{K}(t) + Q(t)$$

and

$$\mathcal{K}(t) = R^{-1}(t)B^T(t)P(t)$$

Detailed discussions of the continuous LQR problem can be found in [**3**].

B.1.3. Continuous LQ Problem

In Sec. 6.4.1, it was shown that the descent direction can be calculated by solving an equivalent LQ problem. The LQ problem is a generalized form of the LQR problem. The solution is found by augmenting the system with extra states and applying the LQR solution. A detailed derivation is given in [**3**].

Problem Statement: Find the control input $u(t)\ \forall\ t \in [t_0, t_f]$ that minimizes the cost:

$$V(x(t_0), u(\cdot), t_0) = \int_{t_0}^{t_f} 2 \begin{bmatrix} q(t) \\ r(t) \end{bmatrix}^T \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} + \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T \begin{bmatrix} Q(t) & S(t) \\ S^T(t) & R(t) \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \mathrm{d}t$$
$$+ 2q_f^T x(t_f) + x^T(t_f) Q_f x(t_f)$$

where

$$R(t) = R^T(t) > 0\ \forall\ t \in [t_0, t_f]$$
$$Q(t) = Q^T(t) \geq 0\ \forall\ t \in [t_0, t_f]$$
$$Q_f = Q_f^T \geq 0$$
$x(t_0)$ is known.
$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$

Solution: The optimal control $u^*(t)$ and optimal cost $V^*(x(t_0), t_0)$ are

$$u^*(t) = -\mathcal{K}(T)x(t) - C(t)$$

$$V^*(x(t_0), t_0) = x^T(t_0)P(t_0)x(t_0) + 2b^T(t_0)x(t_0) + c(t_0)$$

where:

$$\mathcal{K}(t) = R^{-1}(t)\left(B^T(t)P(t) + S^T(t)\right)$$

$$C(t) = R^{-1}(t)\left(B^T(t)b(t) + r(t)\right)$$

$$P(t_f) = Q_f$$

$$-\dot{P}(t) = P(t)A(t) + A^T(t)P(t) - \mathcal{K}(t)R(t)\mathcal{K}(t) + Q(t)$$

$$b(t_f) = q_f$$

$$-\dot{b}(t) = [A(t) - B(t)\mathcal{K}(t)]^T b(t) + q(t) - \mathcal{K}^T(t)r(t)$$

$$c(t_f) = 0$$

$$\dot{c} = C^T(t)R(t)C(t)$$

B.2. Discrete Linear Systems

In this section, we briefly review relevant discrete time linear system theory. The discrete time-varying system has dynamics of the form:

$$x(k+1) = A(k)x(k) + B(k)u(k). \tag{B.1}$$

B.2.1. Discrete Linear System Trajectories

The discrete linear system (B.1) has an associated *state transition matrix* $\Phi(i,j)$ with the following properties.

$$\Phi(k,k) = \mathcal{I} \tag{B.2a}$$

$$k > j: \quad \Phi(k,j) = A(k-1)\Phi(k-1,j) \tag{B.2b}$$

$$k > j: \quad \Phi(k,j) = \Phi(k,j+1)A(j) \tag{B.2c}$$

As with the continuous state transition matrix, these are useful identities and as well as a means to calculate $\Phi(k,j)$ numerically. In particular, $\Phi(k,j)$ can be calculated using (B.2a) as an initial condition and either (B.2b) or (B.2c) to advance forward or backward in time, respectively.

Solutions to (B.1) can be written in terms of $\Phi(k,j)$:

$$x(k) = \Phi(k,k_0)x(k_0) + \sum_{j=k}^{k-1} \Phi(k,j+1)B(j)u(j) \tag{B.3}$$

where $x(k_0)$ are the system's initial conditions. Discrete time linear systems are discussed in detail in [**11**].

B.2.2. Discrete LQR Problem

As in the continuous case, the discrete LQR problem is used to automatically generate feedback laws along arbitrary trajectories for the discrete projection operator (6.10). The discrete LQR problem is discussed in [**3**].

Problem Statement: Find the control input $u(k)\ \forall\ k \in \{k_0 \ldots (k_f - 1)\}$ that minimizes the cost:

$$V(x(k_0), u(\cdot), k_0) = \sum_{k=k_0}^{k_f-1} \left[x^T(k)Q(k)x(k) + u^T(k)R(k)u(k)\right] + x^T(k_f)Q(k_f)x(k_f)$$

where

$$R(k) = R^T(k) \geq 0\ \forall\ k \in \{k_0 \ldots (k_f - 1)\}$$

$$Q(k) = Q^T(k) \geq 0\ \forall\ k \in \{k_0 \ldots k_f\}$$

$$x(k_0) \text{ is known.}$$

$$x(k+1) = A(k)x(k) + B(k)u(k)$$

Solution: The optimal control $u^*(k)$ and optimal cost $V^*(x(k_0), k_0)$ are

$$u^*(k) = -\mathcal{K}(k)x(k)$$

$$V^*(x(k_0), k_0) = x^T(k_0)P(k_0)x(k_0)$$

where

$$\mathcal{K}(k) = \Gamma^{-1}(k)B^T(k)P(k+1)A(k)$$

$$\Gamma(k) = R(k) + B^T(k)P(k+1)B(k)$$

and $P(k+1)$ is a symmetric time varying matrix satisfying a discrete Ricatti-like equation:

$$P(k_f) = Q(k_f)$$

$$P(k) = Q(k) + A^T(k)P(k+1)A(k) - \mathcal{K}^T(k)\Gamma(k)\mathcal{K}(k)$$

B.2.3. Discrete LQ Problem

The Linear Quadratic (LQ) problem is a common optimal control problem for linear systems. The discrete LQ problem is to find the control input $u(k)$ that minimizes the cost

$$V(x(k_0), u(\cdot), k_0) = \sum_{k=k_0}^{k_f-1} \left(\begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q(k) & S(k) \\ S^T(k) & R(k) \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \right.$$
$$\left. + 2 \begin{bmatrix} q(k) \\ r(k) \end{bmatrix}^T \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \right) + 2q^T(k_f)x(k_f) + x^T(k_f)Q(k_f)x(k_f)$$

where

$$R(k) = R^T(k) > 0 \ \forall \ k$$

$$Q(k) = Q^T(k) \geq 0 \ \forall \ k$$

$$x(k_0) \text{ is known.}$$

$$x(k+1) = A(k)x(k) + B(k)u(k)$$

The LQ problem is solved as a generalization of the discrete LQR problem[**3**]. The discrete system is augmented with a unity state (i.e, $\hat{x}^T(k) = \begin{bmatrix} x^T(k) & 1 \end{bmatrix}$) and the input is augmented with feedback and affine terms (i.e, $\hat{u}(k) = u(k) + M_1(k)x(k) + M_2(k)$). The LQR cost for the augmented system with $\hat{x}(k)$ and $\hat{u}(k)$ is expanded and set equal to the desired LQ cost to find equations for M_1, M_2, and the LQR cost terms that make the problems equivalent. The solution to the LQR problem is then expanded using these values to find the solution to the LQ problem[**3**].

The resulting optimal control $u^*(k)$ and optimal cost $V^*(x(k_0), k_0)$ are:

$$u^*(k) = -\mathcal{K}(k)x(k) - C(k)$$

$$V^*(x(k_0), k_0) = x^T(k_0)P(k_0)x(k_0) + 2b^T(k_0)x(k_0) + c(k_0)$$

where

$$\mathcal{K}(k) = \Gamma^{-1}(k)\left[B^T(k)P(k+1)A(k) + S^T(k)\right]$$

$$C(k) = \Gamma^{-1}(k)\left[B^T(k)b(k+1) + r(k)\right]$$

$$\Gamma(k) = \left[R(k) + B^T(k)P(k+1)B(k)\right]$$

and $P(k)$, $b(k)$, and $c(k)$ are calculated from backwards difference equations:

$$\begin{aligned}
P(k_f) &= Q(k_f) \\
P(k) &= Q(k) + A^T(k)P(k+1)A(k) - \mathcal{K}^T(k)\Gamma(k)\mathcal{K}(k) \qquad \text{(B.4)} \\
b(k_f) &= q(k_f) \\
b(k) &= \left[A^T(k) - \mathcal{K}^T(k)B^T(k)\right]b(k+1) + q(k) - \mathcal{K}^T(k)r(k) \\
c(k_f) &= 0 \\
c(k) &= c(k+1) - C^T(k)\Gamma(k)C(k).
\end{aligned}$$

Implementations should expand the last term in (B.4) and explicitly cancel the $\Gamma(k)$ terms to avoid numerical instability. Additionally, $P(k)$ should be numerically projected into a symmetric matrix (i.e, $P(k) = \frac{1}{2}(P(k) + P^T(k)))$ after each iteration.