1-1-2004

Applications of the local state-space form of constrained mechanical systems in multibody dynamics and robotics

Erik Lee Spencer
Iowa State University

Applications of the local state-space form of constrained mechanical systems in multibody dynamics and robotics

by

Erik Lee Spencer

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
Atul G. Kelkar, Major Professor
Degang Chen
Greg R. Luecke

Iowa State University

Ames, Iowa

2004

Graduate College
Iowa State University

This is to certify that the master's thesis of

Erik Lee Spencer

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

The simulation of mechanical systems often involves the solution of differential algebraic equations (DAEs). DAEs occur in every mechanism containing kinematic loops. Such systems can be found in a wide range of areas including the aerospace, automotive, construction, and farm equipment industries. The numerical treatment of DAEs is a topic which is relatively recent and continues to be studied.

One can regard DAEs as ordinary differential equations (ODEs) on certain invariant manifolds after index reduction. Thus, the numerical solutions of the DAEs can be obtained through integration of their underlying ODEs. In certain circumstances, difficulties may occur since the numerical solutions of the underlying ODE can drift away from the invariant manifold. In this thesis, the underlying ODEs are locally transformed into ODEs of minimal dimension via local parameterizations of the invariant manifold. By their nature, such ODEs are local and implicit but their solutions do not suffer from the drift phenomenon. Since the states of these minimal ODEs are independent, they are known as a *local state-space form* of the equations of motion. This work focuses on generalizing the application of the local state-space form and applying it towards problem areas in multibody dynamics and robotics.

Several researchers have proposed standardized methods of parameterizing the invariant manifold; the two popular of which are known as Generalized Coordinate Partitioning [1] and the Tangent Space Method [2, 3]. One objective of this work is to appeal to concepts from differential geometry to generalize the different approaches and algorithms. With such a framework in place, one can search for more stable and/or reliable methods to better handle certain situations. For instance, a certain method of parameterizing the manifold may lead to a computational approach which is better numerically suited near kinematic singularities.

In this work, the first application of the local state-space form is in deriving a formulation of dynamics called the Singularity Robust Null Space Formulation. This formulation utilizes several aspects of the singular value decomposition for an approach which is efficient, does not fail at singularities, and is better suited than most near singularities.

When studying the behavior of mechanical systems described by DAEs, it is natural to want to use the tools normally associated with ODEs such as linearization. However, several questions might arise such as, "Should we look at the linearized underlying ODE or the linearized local state-space form? What is the relationship between the two and how do they relate to the linearized DAE?" As it turns out, local parameterizations of the invariant manifold are crucial in the study of the linearized local state-space and underlying ODE forms. Hence, the second application area in this work is the study of the linearized mechanical system. Since the linearized model is also useful in optimization and implicit integration problems, an efficient recursive algorithm for its construction is derived. The algorithm appeals to a formulation of the dynamics found in robotics to ease in a coherent derivation.

1.1 Literature Review

The concept of a local state-space form first emerged when a formulation of the dynamics known as Generalized Coordinate Partitioning (GCP) [1] was introduced. GCP was introduced as an integration method which aimed at eliminating the problem of drift. By partitioning the coordinates into an independent and dependent set and integrating only the independent coordinates, the dependent coordinates could be found iteratively as the nearest point on the invariant manifold. Similar methods have been introduced which differ in how the independent coordinates are specified; the following list summarizes the most important of them:

1. *Generalized Coordinate Partitioning:* This procedure uses a Gram-Schmidt orthogonalization or QR decomposition of the system constraint jacobian to identify an independent subset of coordinates [1, 4].

2. *Tangent Space Method:* This procedure uses a QR or SVD decomposition to find the null space of the constraint jacobian. The null space basis identifies an independent set

of velocities lying on the tangent plane of the invariant manifold [5, 6].

3. *Differentiable Null Space Method:* This procedure uses Gram-Schmidt orthogonalization or QR decomposition to obtain a basis for the null space of the constraint jacobian. In addition, the Gram-Schmidt orthogonalization or QR decomposition algorithm is differentiated to provide a derivative of the null space basis. Together, the basis and its derivative define a local state-space form of the equations of motion [7].

4. *Globally Independent Coordinates:* Using an intuitive approach which often involves highly nonlinear relationships, a set of coordinates which remain independent over the entire invariant manifold are identified. Coordinates which are not truly global are also acceptable as long as the system is guaranteed not to travel through the configurations where they fail [8].

Various methods for integrating the local state-space from have been proposed [9] including multi-step [2, 10, 6, 11] and implicit [12] integration schemes. The main problem that these integration schemes have to deal with is how to best iteratively solve for the dependent coordinates.

What the literature on the local state-space form lacked was a well-developed general representation. Yen had developed a general representation for Tangent Space Methods [13] and it seemed natural to try to extend his representation to the fully nonlinear case [8]. Under such a representation all other methods, current and future, could be viewed as subsets. In addition, much of the associated theory could be expressed in the general framework. The reformulation of the underlying ODE using the local parameterizations is one of these applications. In this work, these reformulations are termed *Null Space Methods* and there generalized representations are partly inspired from the work found in [14] and [15].

Applied mathematicians have been studying DAEs in recent years and often use multibody systems as examples. For example, some recent software tools have benefited from the work of Ascher and Chin [16, 17] who made improvements on the classical Baumgarte stabilization [18]. Their work and others often use a form of stabilization generally known as Projection

[19, 20, 21] (very similar to the local state-space form approach from a numeric standpoint). Others have proposed a form of stabilization known as Regularization [22, 23, 24, 25] which has had some success dealing with rank-deficient constraints. Other researchers have concentrated on specialized integration routines that only apply to mechanical systems [26]. While learning all of the various integration methods can be time consuming, they make the difference between a good dynamics package and a very bad one; anyone studying in the field needs to understand them. However, only the integration methods that deal with the local state-space form have a bearing on this thesis.

In another line or work, researchers have studied the efficient integration of mechanical systems. Which approach is best has often been a topic of debate [26, 27, 28, 29]. It has often been the case that such formulations have looked towards algorithms from robotics [28] for increased efficiency. The robotics algorithms use a minimal number of coordinates for open chain systems. In systems with kinematic loops, the coordinates are no longer independent since they are subject to loop-closure constraints. Literature regarding the application of recursive robotics algorithms to mechanisms with kinematic loops almost always uses a non-minimal set of coordinates [30, 31]. Each loop is usually removed by cutting a joint and introducing Lagrange multipliers to mimic the forces that would exist at the joint. An efficient recursive scheme is then used for the open chain system.

One of the most recursive popular algorithms from robotics is the Articulated-Body Algorithm [32] by Featherstone which solves the forward dynamics problem of open chain systems with linear complexity. Featherstone invented a notation to express his algorithms which uses 6-dimensional vectors to represent generalized forces and velocities in the system. Very similar notations were invented which do little more than express the ideas originated by Featherstone [33, 34]. One of these notations, the Lie algebraic formulation [34, 35], has been used by its authors on optimization problems which require partial derivatives of the dynamics of open chain systems [36, 37, 38].

While doing some cut-joint based simulation work during the summer of 2002, a need arose for the generation of multiple linearized models. Unimpressed with the accuracy and speed of

finite-difference based linearization, work on a recursive linearization algorithm steeped in the cut-joint approach was undergone. The approach was to express the open chain and cut-joint relations in a common, Lie algebraic, notation [34]. The derivation of the linearized models would hopefully then take the elegant form found in similar research [37, 38]. All prior known research on the linearization of constrained mechanical systems concentrated on augmented formulations [39, 40].

1.2 Contributions

The following list summarizes the research contributions described in this thesis:

- Used concepts from topology to generalize and unify the application of the local state-space form in multibody dynamics and robotics.
- Studied in detail the numeric stability of a formulation of the dynamics near singularities.
- Generalized the concept of the linearized underlying ODE.
- Formulated the dynamics of closed-chain robots in a way which is consistent with the usual treatment of open chain robots including the uniform way in which joint constraints are represented.
- Derived an efficient recursive formulation of the linearized local state-space form with application towards optimization and implicit integration schemes.

1.3 Outline

Chapter 2 begins with an introduction of basic mechanical modelling concepts. A standard model of the equations of motion is introduced which considers scleronomic constraints. The conversion of the equations of motion to the underlying ODE and its interpretation as an ODE with invariants is covered in Section 2.1.1. Local parameterizations of the manifold are discussed in Section 2.2. The selection of the state variables of the local state-space form is covered in Section 2.3 and the special case when these states are a linear combination of the

generalized coordinates is discussed in Section 2.3.3. Finally, Section 2.4 reformulates the underlying ODE using the local parameterizations. The reformulation has certain computational advantages for highly constrained systems.

Chapter 3 develops a particular formulation of the dynamics which utilizes the singular value decomposition. The numerical integration near kinematic singularities is studied in Sections 3.1.1 and 3.1.2. A suggested form of stabilization known as Projection is studied in Section 3.1.3. Finally, numeric examples illustrating the performance of the formulation are given in Section 3.2 followed by some final remarks in Section 3.3.

In Chapter 4, an efficient recursive formulation of the linearized local state-space form is given. The chapter begins with a basic discussion of the linearized underlying ODE. As the representation of the linearized model is not unique, theorems regarding its eigenvalue relationships are developed. The structure of the linearized local state-space form is introduced in Section 4.2. The core numeric algorithms used for the construction of the linearized model are covered in Sections 4.3 and 4.4. An example application of the linearization algorithm is carried out in Section 4.5 followed by some final remarks in Section 4.6.

CHAPTER 2. LOCAL STATE-SPACE FORMS OF THE EQUATIONS OF MOTION

2.1 Mechanical Modeling Concepts

The Euler-Lagrange formulation of the equations of motion of constrained multibody systems can be written in the following general form:

$$\dot{q} = v \tag{2.1a}$$

$$M(q)\dot{v} = F(t,q,v) - G(q)^T\lambda \tag{2.1b}$$

$$0 = g(q) \tag{2.1c}$$

where

- q is an n-dimensional vector of generalized positions.
- v is an n-dimensional vector of generalized velocities.
- $g(q)$ is a twice continuously differentiable m-dimensional vector ($\mathrm{m} \leq \mathrm{n}$) of independent position level scleronomic constraints (closed loops).
- $G(q) \in \mathbb{R}^{m\times \mathrm{n}}$ is the constraint jacobian matrix defined as $G := Dg(q)$. The constraint jacobian G is assumed to have full row rank for all physically realizable positions q.
- $M(q) \in \mathbb{R}^{\mathrm{n}\times \mathrm{n}}$ is the mass-inertia matrix. $M(q)$ is symmetric positive definite for all q.
- $F(t,q,v)$ is the n-dimensional vector representing the contribution of centrifugal, Coriolis, and external forcing terms.
- λ is the m-dimensional vector of Lagrange multipliers.

The algebraic constraints, g, arise from mainly two sources:

1. *Augmented Formulation.* In an augmented formulation [41], the configuration of each body is identified using full cartesian coordinates to indicate each body's position and orientation. If some body is constrained to another by a mechanical joint, then algebraic constraints are used to remove the degrees of freedom that the joint restricts.

2. *Closed Chains.* In a recursive formulation of the dynamics [32], relative joint coordinates are used which result in a minimal set of coordinates for systems with tree-structured topologies. However, when the constrained relationships amongst the bodies is cyclic, the loops formed by these constraints must be *virtually cut* open to regain a tree-structured topology. The kinematic relationships removed in the cutting process are modelled as algebraic constraints.

When the differential equations are coupled with the algebraic constraints the equations of motion (2.1) are known index-3 differential-algebraic equations (DAEs). Traditionally, the numerical solutions of (2.1) are solved by first reducing it to an ordinary differential equation (ODE) by a technique called *index reduction.* In index reduction, the equations of motion are reduced to an index-0 DAE (or, ODE) by differentiating the kinematic constraints with respect to time (for the definition of index, see [42, 43]). For mechanical systems, two differentiations of (2.1c) are required to yield the reduction. The first differentiation of (2.1c) gives constraints on velocity level

$$0 = \dot{g} = G(q)v \tag{2.2}$$

and differentiating again gives constraints on acceleration level

$$0 = \ddot{g} = G(q)\dot{v} + \dot{G}v. \tag{2.3}$$

Solving (2.1b) for $\dot{v}$ and substituting into (2.3), one can solve for the Lagrange multipliers

$$\lambda = \Lambda(q, v) := (GM^{-1}G^T)^{-1}(GM^{-1}F + \dot{G}v). \tag{2.4}$$

Combining (2.1) and (2.4) yields the equations of motion in ODE form

$$\dot{q} = v \tag{2.5a}$$

$$\dot{v} = M^{-1}(q)(F(t,q,v) - G(q)^T\Lambda(q,v)). \tag{2.5b}$$

The analytical solutions of the DAE (2.1) and it's *underlying ODE* (2.5) are identical. That is, under appropriate initial conditions and perfect integration the evolution of the states $q(t)$ and $v(t)$ exactly satisfy the constraints (2.1c) and (2.2).

2.1.1 Equations of Motion as ODEs with Invariants

Let the N := 2n dimensional state of the mechanical system be given by $x = (q^T v^T)^T$ then the ODE (2.5) can be written in the concise form

$$\dot{x} = f(t,x) \tag{2.6}$$

with the M := 2m dimensional vector of invariants

$$h(x) = \begin{pmatrix} g(q) \\ G(q)v \end{pmatrix} = 0. \tag{2.7}$$

We call (2.7) an invariant of (2.6) since the solutions on any interval $[t_o\,,\,t_f]$ satisfy $h(x(t)) = 0$ for all $t \in [t_o\,,\,t_f]$ such that $h(x(t_o)) = 0$ (c.f. [17]). All solutions of (2.6) therefore lie on the invariant set or manifold

$$\mathcal{M} = \{x : h(x) = 0\}. \tag{2.8}$$

However, numerically integrating (2.6) will inherently lead to small perturbations off the invariant manifold $\mathcal{M}$ since the invariants are not explicitly involved in the numerical integration solution procedure. Methods which use knowledge of the invariant (2.7) are often used to correct the solutions. In the reduction to a *local state-space form* method, knowledge of the invariant is used to reduce the underlying ODE (2.6) to a minimal ODE whose solutions

automatically lie on the invariant manifold $\mathcal{M}$. The local state-space form is introduced next in Section 2.2.

2.2 Local State-Space Form of the Equations of Motion

It is possible to locally reduce the equations of motion (2.1) to a minimal ODE whose solutions automatically satisfy the constraints. Such a representation of the equations of motion is known as a *local state-space form.* In this section the existence of the local state-space form is established.

2.2.1 The Local Parametrization

Let $x_o \in \mathcal{M}$ be some point on the invariant manifold. At this point one has

$$h(x_o) = 0 \tag{2.9}$$

and

$$\text{rank}\,[Dh(x_o)] = \text{rank}\begin{bmatrix} G & 0 \\ \dot{G} & G \end{bmatrix} = \text{M} \tag{2.10}$$

since by assumption G is full rank. Hence h is a submersion at each point on $\mathcal{M}$, which then is a submanifold of $\mathbb{R}^{\text{N}}$ of dimension $\text{P} := \text{N} - \text{M}$ [44]. In other words, the dimension of the invariant manifold $\mathcal{M}$ equals the total degrees of freedom on the position and velocity levels.

Since $\mathcal{M}$ is a P-dimensional manifold it is locally diffeomorphic to $\mathbb{R}^{\text{P}}$; therefore, at the arbitrary point x_o there is an open submanifold $\mathcal{X} \subset \mathcal{M}$ which is diffeomorphic to an open set $\mathcal{Y}$ in $\mathbb{R}^{\text{P}}$. The diffeomorphism $\Psi : \mathcal{Y} \to \mathcal{X}$ is continuously differentiable, one-to-one onto, and the inverse map $\Psi^{-1} : \mathcal{X} \to \mathcal{Y}$ is also continuously differential (c.f. [45]). The implicitly defined function Ψ will be referred to as a *local parametrization* [6]. To put it another way, there exists a set of coordinates, y, representing the degrees of freedom of the mechanical system which uniquely determine the generalized positions and velocities and viceversa (at least locally). In addition, the analytic functions of these relationships, Ψ and Ψ^{-1}, are continuous with continuous derivatives.

2.2.2 The Local State-Space Form

Since Ψ is one to one onto then for any $x \in \mathcal{X}$ there is a $y \in \mathcal{Y}$ such that

$$x = \Psi(y) \tag{2.11}$$

and differentiating with respect to time and substituting (2.6) for $\dot{x}$ it holds that

$$D\Psi(y)\,\dot{y} = f(t, \Psi(y)) \quad , \quad y \in \mathcal{Y}. \tag{2.12}$$

Now since Ψ is a diffeomorphism the jacobian $D\Psi$ has full column rank and equation (2.12) can be viewed as an overdetermined system in $\dot{y}$; therefore, a solution is given by

$$\dot{y} = D\Psi(y)^{\dagger}\, f(t, \Psi(y)) \quad , \quad y \in \mathcal{Y} \tag{2.13}$$

where $\dagger$ denotes the Moore-Penrose generalized inverse. This ODE represents the equations of motion reduced on a local parameter space of the invariant manifold and is called the *local state-space form* of the equations of motion.

Similarly, the local state-space form could have been derived by noting that for any $y \in \mathcal{Y}$ there is an $x \in \mathcal{X}$ such that

$$y = \Psi^{-1}(x) \tag{2.14}$$

and differentiating (2.14) with respect to time it holds that

$$\dot{y} = D\Psi^{-1}(\Psi(y))\, f(t, \Psi(y)) \quad , \quad y \in \mathcal{Y} \tag{2.15}$$

(note that this does not imply $(D\Psi)^{\dagger} = D\Psi^{-1}$). In any case, it is unique and a general representation may be given by

$$\dot{y} = \hat{f}(t, y) \quad , \quad y \in \mathcal{Y} \tag{2.16}$$

where $\hat{f}$ can be thought of as any solution of (2.12) for $\dot{y}$.

The Pth-order ODE, $\hat{f}$ (2.16), and the Nth-order the underlying ODE, f (2.6), represent the same mechanical system. However, there are some important differences in how the two are integrated. While the solutions of the underlying ODE f may drift off the invariant manifold $\mathcal{M}$, solutions of the local state-space form $\hat{f}$ always satisfy the constraints since for any discreet solution $y_n \in \mathcal{Y}$ one has

$$h(\Psi(y_n)) = 0. \tag{2.17}$$

Therefore, the implicit nature of $\hat{f}$ gives solutions whose numerical error is restricted to the manifold. On the other hand, solutions of the underlying ODE may drift from the manifold since the position and velocity constraints are not explicitly involved in its solution.

2.2.3 The Tangent Space

Differentiating equation (2.17) gives,

$$H(\Psi(y)) \cdot D\Psi(y) = 0 \quad , \quad y \in \mathcal{Y} \tag{2.18}$$

where $H(x) := Dh(x)$ is the jacobian of the invariants. Therefore, the columns of $D\Psi$ form a basis for the P-dimensional null-space of H at each $y \in \mathcal{Y}$. Similarly, they also represent a basis for the tangent space of $\mathcal{M}$ at $x = \Psi(y)$ which is defined as

$$T_x(\mathcal{M}) = \{z \in \mathbb{R}^{\mathrm{N}} : H(x)z = 0\}. \tag{2.19}$$

When considered as a vector space, elements of $T_x(\mathcal{M})$ may be thought to represent tangents to all possible trajectories passing through the point $x \in \mathcal{M}$.

For any local parametrization Ψ, the derivative $D\Psi$ at $y = \Psi^{-1}(x)$ is a linear transformation from $\mathbb{R}^{\mathrm{P}}$ to $T_x(\mathcal{M})$. Correspondingly, the derivative of Ψ^{-1} at x is the linear transformation $D\Psi^{-1}(x) : T_x(\mathcal{M}) \to \mathbb{R}^{\mathrm{P}}$. From the inverse function theorem, these linear mappings are one-to-one onto (and hence have full rank).

A commutative diagram showing the relationship between the underlying ODE (2.6) and

the local state-space form (2.16) at each $y_o \in \mathcal{Y}$ and $x_o = \Psi(y_o) \in \mathcal{X}$ is shown in Fig.2.1. The commutative diagram illustrates how the local parametrization Ψ relates the equivalent

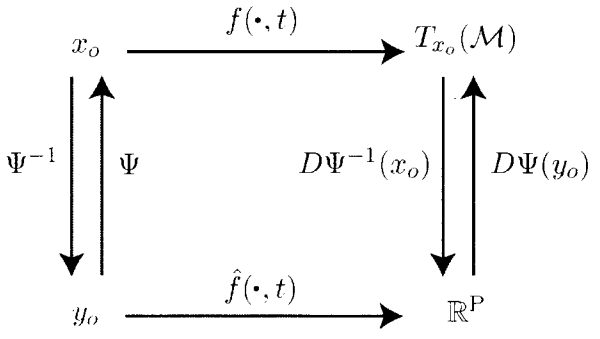


Figure 2.1 Commutative diagram illustrating the equivalence of f and $\hat{f}$

solutions of f and of $\hat{f}$.

The local parameterizations and their derivatives are not necessarily unique and are implicitly defined; appropriate constructions are outlined in Section 2.3.

2.3 Change of Coordinates

Each element of the parameters $y \in \mathcal{Y}$ correspond to a unique degree of freedom in the position and velocity space of the mechanical system. Accordingly, y is often referred to as a *minimal* or *independent* state of the mechanical system. Often physically insightful and nearly global choices of independent coordinates exist; other times automatic selection techniques are used which may only hold locally. In either case, the relationship between a desired independent state y and the full state x can almost always be written as a P-dimensional vector of constraints

$$R(x, y) = 0 \tag{2.20}$$

for some continuously differentiable function $R : \bar{\mathcal{X}} \times \bar{\mathcal{Y}} \to \mathbb{R}^{\mathrm{P}}$ with open subsets $\bar{\mathcal{X}} \subset \mathbb{R}^{\mathrm{N}}$, $\bar{\mathcal{Y}} \subset \mathbb{R}^{\mathrm{P}}$. If the coordinates y are truly *independent* then R must possess certain local attributes which are summarized in the following theorem.

Theorem 2.3.1: Consider some point $x_o \in \bar{\mathcal{X}}$ such that $h(x_o) = 0$ and let $y_o \in \bar{\mathcal{Y}}$ be such that $R(x_o, y_o) = 0$. If R satisfies

$$|D_2 R(x_o, y_o)| \neq 0 \tag{2.21}$$

and

$$\left| \begin{pmatrix} D_1 R(x_o, y_o) \\ H(x_o) \end{pmatrix} \right| \neq 0 \tag{2.22}$$

then there exists a unique continuously differentiable diffeomorphism Ψ between some open neighborhoods $\mathcal{Y} \subset \bar{\mathcal{Y}}$ of y_o and $\mathcal{X} \subset \bar{\mathcal{X}}$ of x_o such that $R(\Psi(y), y) = R(x, \Psi^{-1}(x)) = 0$ for all $y \in \mathcal{Y}$ and $x \in \mathcal{X}$.

Proof: Consider the continuously differentiable function $\varphi : \bar{\mathcal{X}} \times \bar{\mathcal{Y}} \to \mathbb{R}^N$ given by

$$\varphi(x, y) = \begin{pmatrix} R(x, y) \\ h(x) \end{pmatrix} \tag{2.23}$$

where at (x_o, y_o)

$$\varphi(x_o, y_o) = \begin{pmatrix} R(x_o, y_o) \\ h(x_o) \end{pmatrix} = 0$$

and from (2.22) one has

$$|D_1 \varphi(x_o, y_o)| = \left| \begin{pmatrix} D_1 R(x_o, y_o) \\ H(x_o) \end{pmatrix} \right| \neq 0.$$

Then from the implicit function theorem [46] there exists open subsets $\mathcal{X} \subset \bar{\mathcal{X}}$, $\mathcal{Y} \subset \bar{\mathcal{Y}}$ and a unique continuously differentiable function $\Psi : \mathcal{Y} \to \mathcal{X}$ such that

$$\varphi(\Psi(y), y) = \begin{pmatrix} R(\Psi(y), y) \\ h(\Psi(y)) \end{pmatrix} = 0 \quad , \quad y \in \mathcal{Y}. \tag{2.24}$$

Differentiating (2.24) at (x_o, y_o) gives

$$\begin{pmatrix} D_1 R(x_o, y_o) \\ H(x_o) \end{pmatrix} D\Psi(y_o) + \begin{pmatrix} D_2 R(x_o, y_o) \\ 0 \end{pmatrix} = 0$$

and solving for $D\Psi(y_o)$,

$$D\Psi(y_o) = \begin{pmatrix} D_1 R(x_o, y_o) \\ H(x_o) \end{pmatrix}^{-1} \begin{pmatrix} -D_2 R(x_o, y_o) \\ 0 \end{pmatrix}.$$

Note that the jacobian $D\Psi(y_o)$ has full column rank precisely when the first condition of the theorem (2.21) is satisfied. This derivative of $\Psi : \mathcal{Y} \rightarrow \mathcal{X}$ at $y_o \in \mathcal{Y}$ is a linear transformation $D\Psi(y_o) : \mathbb{R}^{\mathrm{P}} \rightarrow T_{x_o}(\mathcal{M})$ where from (2.19) $T_{x_o}(\mathcal{M}) = \mathrm{Null}(H(x_o))$. To explicitly verify that $D\Psi(y_o)$ maps into the null-space of $H(x_o)$ observe that

$$\begin{aligned} HD\Psi &= H \begin{pmatrix} D_1 R \\ H \end{pmatrix}^{-1} \begin{pmatrix} -D_2 R \\ 0 \end{pmatrix} \\ &= \left[\begin{pmatrix} 0 & I_{\mathrm{M}} \end{pmatrix} \begin{pmatrix} D_1 R \\ H \end{pmatrix} \right] \begin{pmatrix} D_1 R \\ H \end{pmatrix}^{-1} \begin{pmatrix} -D_2 R \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & I_{\mathrm{M}} \end{pmatrix} \begin{pmatrix} -D_2 R \\ 0 \end{pmatrix} \\ &= 0 \end{aligned}$$

To show that $D\Psi(y_o)$ is one-to-one onto consider some $v \in T_{x_o}(\mathcal{M})$ and $w \in \mathbb{R}^{\mathrm{P}}$ such that $v = D\Psi(y_o)w$. Since $D\Psi(y_o)$ has full column rank there exists a set of P linearly independent rows and without loss of generality, suppose these are the first P rows such that

$$v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} D\Psi_1(y_o) \\ D\Psi_2(y_o) \end{pmatrix} w$$

with $|D\Psi_1(y_o)| \neq 0$. Since $D\Psi_1(y_o)$ is invertible, the mapping $v_1 = D\Psi(y_o)w$ is one-to-

one onto between $\mathbb{R}^{\mathrm{P}}$ and $\mathbb{R}^{\mathrm{P}}$. But note that v_2 is uniquely determined from v_1 since $v_2 = D\Psi_2(y_o)w = D\Psi_2(y_o)(D\Psi_2(y_o))^{-1}v_1$ and therefore $D\Psi(y_o)$ is a one-to-one onto mapping from $\mathbb{R}^{\mathrm{P}}$ to $T_{x_o}(\mathcal{M})$. From the inverse function theorem, the fact that $D\Psi(y_o)$ is one-to-one onto implies that Ψ is a local diffeomorphism. Taking the neighborhoods $\mathcal{X}$ of x_o and $\mathcal{Y}$ of y_o sufficiently small, $\Psi : \mathcal{Y} \rightarrow \mathcal{X}$ is such that $R(\Psi(y), y) = R(x, \Psi^{-1}(x)) = 0$ for all $y \in \mathcal{Y}$ and $x \in \mathcal{X}$. $\square$

Constraints Defined on the Position Manifold

In practice, it is common to define the constraint on the independent position state as a $\mathrm{p} := \mathrm{n} - \mathrm{m}$ dimensional vector valued function

$$R_q(q, y_q) = 0 \tag{2.25}$$

and take the independent velocity state as $y_v = \dot{y}_q$. In this case the constraint on the independent velocity state may be defined as

$$R_v(x, y) = \frac{d}{dt}R_q(q, y_q) = D_1R_q(q, y_q)v + D_2R_q(q, y_q)y_v = 0 \tag{2.26}$$

and letting $R = (R_q, R_v)$ it is easily verified that conditions (2.21) and (2.22) at $x_o = (q_o, v_o)$ and $y_o = (y_{q_o}, y_{v_o})$ may be reduced to

$$|D_2R_q(q_o, y_{q_o})| \neq 0 \tag{2.27}$$

and

$$\left| \begin{pmatrix} D_1R_q(q_o, y_{q_o}) \\ G(q_o) \end{pmatrix} \right| \neq 0. \tag{2.28}$$

Similar to Theorem 2.3.1 conditions (2.27) and (2.28) imply that there exists a Ψ_q which

is itself a diffeomorphism such that $R_q(\Psi_q(y_q), y_q) = R_q(q, \Psi_q^{-1}(q)) = 0$ for all y_q and q in the neighborhoods $\mathcal{Y}$ and $\mathcal{X}$. Therefore, the local parametrization is of the form $\Psi = (\Psi_q, \dot{\Psi}_q)$.

Example: Consider the simple pendulum of unit length in Fig.2.2 whose position is described by the n = 2 cartesian coordinates $q = (q_1 \ q_2)^T$.

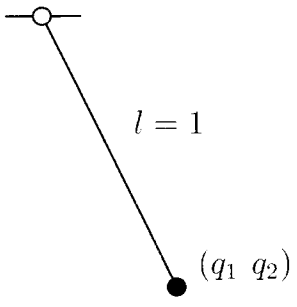


Figure 2.2 Simple pendulum in cartesian coordinates

The constraint that the length of the pendulum remain constant is given by the single equation (m = 1)

$$g(q) = q_1^2 + q_2^2 - 1 = 0$$

and therefore the dimension of the independent position vector y_q is p = n − m = 1 (one degree-of-freedom).

Suppose one chooses the local position coordinate y_q as q_1, i.e.,

$$R_q(q, y_q) = q_1 - y_q = 0;$$

then condition (2.27) is trivially satisfied and the second condition (2.28) is

$$\left| \begin{pmatrix} D_1 R_q(q, y_q) \\ G(q) \end{pmatrix} \right| = \left| \begin{pmatrix} 1 & 0 \\ 2q_1 & 2q_2 \end{pmatrix} \right| \neq 0$$

which clearly holds as long as $q_2 \neq 0$. The two points where $q_2 = 0$ are shown on the graph of the manifold Fig.2.3. It should be clear from Fig.2.3 why choosing $y_q = q_1$ fails at these two

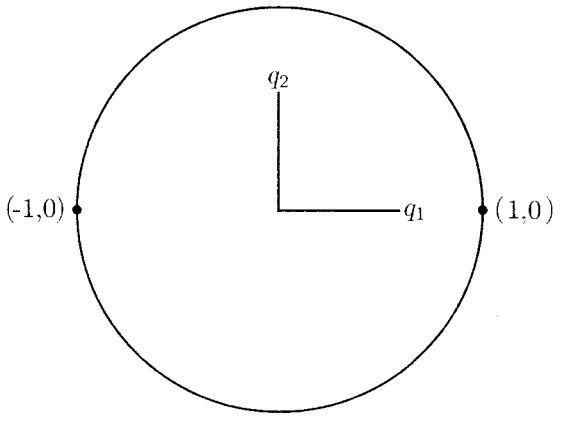


Figure 2.3 The invariant position manifold of the pendulum

points, i.e., q_1 cannot uniquely determine q_2 in any open neighborhood of these points (cannot be one-to-one). □

2.3.1 Iterative Solution of the Local Parametrization

Recall that the local parametrization Ψ is implicitly defined and so iterative methods are usually applied to solve the nonlinear inverse problem of finding an $x \in \mathcal{X}$ such that $x = \Psi(y)$ for a given $y \in \mathcal{Y}$. One usually begins by assuming they have an initial estimate of the solution of the full state denoted by $x^{(0)}$. An iterative method is then applied which starts with $x^{(0)}$ and is considered convergent when the n^{th} estimate of the full state denoted by $x^{(n)}$ is such that $\|x^{(n)} - x\| \leq \epsilon$ where $\epsilon > 0$ is a chosen tolerance [47].

For example, Newton-Raphson iteration may be applied to (2.23) to solve for the local parametrization. On the j^{th} step of the iteration the $(j+1)$ estimate of the full state is the solution of

$$D_1\varphi(x^{(j)}, y)\Delta x^{(j)} = -\varphi(x^{(j)}, y)$$

$$x^{(j+1)} = x^{(j)} + \Delta x^{(j)}.$$

The convergence criterion at the j^{th} step of the iteration may be measured using the inequality

$$\|x^{(j+1)} - x\| \leq \frac{\rho}{1-\rho}\|x^{(j+1)} - x^{(j)}\| \leq \epsilon$$

where ρ is the contraction rate of the iteration which may be estimated by

$$\rho = \left(\frac{\|x^{(j+1)} - x^{(j)}\|}{\|x^{(1)} - x^{(0)}\|} \right)^{(1/(j+1))} .$$

The Newton-Raphson iteration is convergent as long as the matrix $D_1\varphi$ is well conditioned and the initial estimate $x^{(0)}$ is sufficiently close to $x = \Psi(y)$.

In an alternative approach, the Newton-Rhapson iteration may be considered convergent when $\|\varphi(x^{(j+1)}, y)\|$ is sufficiently close to zero. This norm should be weighted to account for the different units of the distance measures.

2.3.2 Derivative of the Local Parametrization

A simple method for finding the partial derivative of the local parametrization is given in this section. The solution requires the inversion of an N × N matrix. For special types of constraints (2.20), an alternative method which requires the inversion of a smaller $M \times M$ matrix is given in Section 2.3.3.1.

In the proof of theorem 2.3.1 it was observed that the derivative of the local parametrization is given by

$$D\Psi(y) = \begin{pmatrix} D_1 R(\Psi(y), y) \\ H(\Psi(y)) \end{pmatrix}^{-1} \begin{pmatrix} -D_2 R(\Psi(y), y) \\ 0 \end{pmatrix} \quad , \quad y \in \mathcal{Y}. \tag{2.29}$$

A slight variation of (2.29) can be derived by noting that the derivative of $\Psi^{-1} \circ \Psi : \mathcal{Y} \to \mathcal{Y}$ is the identity transformation on $\mathbb{R}^{\mathrm{P}}$ which may be expressed as

$$D\Psi^{-1} \cdot D\Psi = I_{\mathrm{P}} \tag{2.30}$$

where I_{P} denotes the usual P×P identity matrix. Combining (2.18) and (2.30) one can solve for the derivative of the local parametrization as

$$D\Psi(y) = \begin{pmatrix} D\Psi^{-1}(\Psi(y)) \\ H(\Psi(y)) \end{pmatrix}^{-1} \begin{pmatrix} I_{\mathrm{P}} \\ 0 \end{pmatrix} \quad , \quad y \in \mathcal{Y}. \tag{2.31}$$

To see that the inverse in (2.31) exists, recall that $D\Psi^{-1}(x)$ is a one-to-one onto mapping from $\mathrm{Null}(H(x))$ to $\mathbb{R}^{\mathrm{P}}$; hence, there does not exist a vector $v \in \mathrm{Null}(H(x))$ such that $D\Psi^{-1}(x)v = 0$ other than the arbitrary solution $D\Psi^{-1}(x) \cdot 0 = 0$. Consequently, there does not exist a nonzero vector $v \in \mathbb{R}^{\mathrm{N}}$ such that

$$\begin{pmatrix} D\Psi^{-1}(x) \\ H(x) \end{pmatrix} v = 0$$

and therefore the inverse exists. The derivative of the inverse of the local parametrization can be found as $D\Psi^{-1} = -(D_2 R)^{-1} D_1 R$ in (2.31).

2.3.3 Linear Change of Coordinates

A common simplifying assumption is that the independent state y is a linear function of x, i.e., the constraint (2.20) may be written in the form

$$R(x, y) = R \cdot (x - \bar{x}) - y = 0 \tag{2.32}$$

or

$$y = R \cdot (x - \bar{x}) \tag{2.33}$$

where $\bar{x} \in \mathbb{R}^{\mathrm{N}}$ is some arbitrary offset and in accordance with condition (2.22) the matrix $R \in \mathbb{R}^{\mathrm{P} \times \mathrm{N}}$ is chosen such that $(R^T H^T)^T$ is nonsingular.

The simple form of the constraint (2.32) makes it a popular choice for methods which automate the selection of independent states. For example, in *Tangent Space Methods* [2, 3] the rows of R form a standard basis for the null space of H so that $(R^T H^T)^T$ is always

nonsingular. In *Generalized Coordinate Partitioning* [1], the independent state y is a subset of the elements of x and consequently the rows of R are simple unit vectors.

2.3.3.1 Derivative of the Local Parametrization: Linear Case

Suppose one defines a set of dependent coordinates similar to (2.33),

$$w = S \cdot (x - \bar{x}) \tag{2.34}$$

where $S \in \mathbb{R}^{\mathrm{M} \times \mathrm{N}}$ has full row rank and is chosen such that $(R^T \; S^T)^T$ is nonsingular. Then the change in coordinates (2.32)-(2.34) defines a structure of the local parametrization of the manifold Ψ,

$$x = \Psi(y) = \begin{pmatrix} R \\ S \end{pmatrix}^{-1} \begin{pmatrix} y \\ \Omega(y) \end{pmatrix} + \bar{x} \tag{2.35}$$

where the continuously differentiable function $w = \Omega(y)$ exists by the implicit function theorem. Equation (2.35) may alternatively be expressed as

$$x = \Psi(y) = \bar{R}y + \bar{S}\Omega(y) + \bar{x} \tag{2.36}$$

where

$$\bar{R} := \begin{pmatrix} R \\ S \end{pmatrix}^{-1} \begin{pmatrix} I_{\mathrm{P}} \\ 0 \end{pmatrix} \quad \text{and} \quad \bar{S} := \begin{pmatrix} R \\ S \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ I_{\mathrm{M}} \end{pmatrix}.$$

The matrices $\bar{R}$ and $\bar{S}$ form bases for the space spanned by the independent and dependent coordinates respectively. It is advantageous to choose R and S such that $(R^T \; S^T)^T$ is unitary and consequently trivially invertible.

Theorem 2.3.3.1.1: If the constraint (2.20) is of the linear form (2.32) then the derivative

of the local parametrization is

$$D\Psi(y) = \bar{R} - \bar{S}\left(H(\Psi(y))\bar{S}\right)^{-1} H(\Psi(y))\bar{R} \quad , \quad y \in \mathcal{Y}. \tag{2.37}$$

Proof: Differentiating (2.36) gives

$$D\Psi = \bar{R} + \bar{S}\, D\Omega \tag{2.38}$$

and substituting $D\Psi$ into (2.18) gives

$$H\, D\Psi = H\bar{R} + H\bar{S}D\Omega = 0. \tag{2.39}$$

Since $(R^T\ H^T)^T \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ is nonsingular and $\bar{S} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ has full column rank one has that

$$\operatorname{rank}\left[\begin{pmatrix} R \\ H \end{pmatrix}\bar{S}\right] = \mathrm{M}.$$

However, note that

$$\begin{aligned} R\bar{S} &= \left[\begin{pmatrix} I_{\mathrm{P}} & 0 \end{pmatrix}\begin{pmatrix} R \\ S \end{pmatrix}\right]\left[\begin{pmatrix} R \\ S \end{pmatrix}^{-1}\begin{pmatrix} 0 \\ I_{\mathrm{M}} \end{pmatrix}\right] \\ &= \begin{pmatrix} I_{\mathrm{P}} & 0 \end{pmatrix}\begin{pmatrix} 0 \\ I_{\mathrm{M}} \end{pmatrix} \\ &= 0 \end{aligned}$$

and consequently

$$\mathrm{M} = \operatorname{rank}\left[\begin{pmatrix} R \\ H \end{pmatrix}\bar{S}\right] = \operatorname{rank}\begin{pmatrix} 0 \\ H\bar{S} \end{pmatrix} = \operatorname{rank}(H\bar{S})$$

which implies $H\bar{S} \in \mathbb{R}^{M\times M}$ is nonsingular. Therefore, (2.39) can be solved for $D\Omega$,

$$D\Omega = -(H\bar{S})^{-1}H\bar{R},$$

and substituting $D\Omega$ back into (2.38) gives the desired result (2.37). □

Remark 1: Since $H\bar{S}$ is invertible only when $(R^T\ H^T)^T$ is as well, the condition number may be monitored during inversion to decide whether a new change of coordinates is necessary.

Remark 2: Suppose a matrix decomposition is applied to reveal the linearly independent columns of H. A valid change of coordinates can be defined by choosing $\bar{S}$ such that $H\bar{S}$ extracts the independent columns of H which ensures it is nonsingular (the columns of $\bar{S}$ are unit vectors). The matrix $\bar{R}$ may consist of the remaining unit vectors (in which case $R = \bar{R}^T$). This general procedure is known as *Generalized Coordinate Partitioning* [1], although, the idea behind it is more easily derived by applying the implicit function theorem to the invariant h.

Remark 3: Note that in the case of linearly independent coordinates the local state-space ODE takes the relatively simple form

$$\dot{y} = R\,f(t, \Psi(y)) \quad , \quad y \in \mathcal{Y}$$

since $D\Psi^{-1} = R$.

2.3.4 Global Change of Coordinates

An independent set of coordinates which satisfy conditions (2.21) and (2.22) over the entire invariant manifold $\mathcal{M}$ are known as *globally independent coordinates*. In practice, globally independent coordinates are very difficult to find and may not exist [8]. In the case that such coordinates do exist, their relationship to the full state x is likely complicated and is most easily formulated as a set of nonlinear constraints, e.g., as in (2.20).

The main advantage of using globally independent coordinates, when they exist, is the absence of computationally expensive automatic independent state selection methods which

usually involve matrix decompositions of the constraint jacobian. Globally independent coordinates are most easily identified using intuition and may be verified by checking conditions (2.21) and (2.22) over the entire invariant manifold.

2.4 Null Space Methods

In this section an alternative derivation of the underlying ODE which utilizes local parameterizations is given. The derivation has computational advantages when p $\ll$ n, i.e., a large collection of bodies with few overall degrees of freedom.

In the following it will be assumed that the independent state is defined as in equations (2.27) and (2.28) where $y = (y_q^T \, \dot{y}_q^T)^T$. The primary consequence of this assumption is that $D\Psi$ has some level of symmetry and takes the following form,

$$D\Psi(y) = \begin{pmatrix} K(y_q) & 0 \\ \dot{K}(y) & K(y_q) \end{pmatrix}. \tag{2.40}$$

Comparing to (2.18) one has that K is a matrix that satisfies

$$GK = 0,$$

i.e., K is in the *null space* of the constraint matrix G. Also, given the structure of $\Psi = (\Psi_q \, \dot{\Psi}_q)$ one has that $K(y_q) := \partial \Psi_q(y_q)/\partial y_q$.

The fact that the matrix K is in the null space of G plays a big role in the alternative derivation of the underlying ODE; for suppose equation (2.1b) is left multiplied by K^T,

$$K^T M \dot{v} = K^T F, \tag{2.41}$$

which consequently annihilates G^T along with the Lagrange multipliers. However, (2.41) represents an underdetermined system in the accelerations, $\dot{v}$. Looking at the relationship $\dot{x} = D\Psi \, \dot{y}$

in terms of the partitions (2.40) one has

$$\dot{q} = K\dot{y}_q \tag{2.42a}$$

$$\dot{v} = K\ddot{y}_q + \dot{K}\dot{y}_q \tag{2.42b}$$

and substituting (2.42b) for $\dot{v}$ in (2.41) gives a system of equations which is solvable in the acceleration $\ddot{y}_q$,

$$K^T M K \ddot{y}_q = K^T F - K^T M \dot{K}\dot{y}_q.$$

Note that the solution requires the inversion of a p×p matrix rather than an a larger n×n matrix as in (2.5b) (this gives a computational advantage when $\mathrm{p} \ll \mathrm{n}$). Substituting the solution of $\ddot{y}_q$ back into (2.42b) gives the alternative underlying ODE,

$$\dot{q} = v \tag{2.43a}$$

$$\dot{v} = K(K^T M K)^{-1} K^T (F - M\dot{K}\dot{y}_q) + \dot{K}\dot{y}_q. \tag{2.43b}$$

The ODE (2.43) does not appear proper due to the apparent dependence of K and $\dot{K}\dot{y}_q$ on y_q and $\dot{y}_q$; however, it can be expressed explicitly in terms of the state variables q and v through some choice of coordinates. Independent of a particular change of coordinates, it suffices to choose K and $\dot{K}\dot{y}_q$ at each instant such that (2.42b) parameterizes the set of solutions of the acceleration constraint (2.3) for $\dot{v}$, i.e.,

- $K \in \mathbb{R}^{\mathrm{n}\times\mathrm{p}}$ is any matrix with full column rank such that $GK = 0$
- $\dot{K}\dot{y}_q$ is any solution of $G\left(\dot{K}\dot{y}_q\right) = -\dot{G}v$ [28].

In a technique known as the *Differentiable Null-Space Method* [7], Gram-Schmidt or QR decomposition is used to find a basis for the null-space of G (this is K) and the decomposition algorithm is further differentiated to find $\dot{K}\dot{y}_q$.

For a specific choice of coordinates, however, K and $\dot{K}\dot{y}_q$ are uniquely determined. Similar

to (2.29) K can be calculated from the equation

$$K = \begin{pmatrix} D_1 R_q \\ G \end{pmatrix}^{-1} \begin{pmatrix} -D_2 R_q \\ 0 \end{pmatrix} \tag{2.44}$$

and the term $\dot{K}\dot{y}_q$ can be computed as

$$\dot{K}\dot{y}_q = - \begin{pmatrix} D_1 R_q \\ G \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \dot{G}v \end{pmatrix}. \tag{2.45}$$

Assuming a linear change of coordinates similar to (2.34)-(2.36) such that the local parametrization on the position level takes the form

$$\Psi_q(y_q) = \bar{R}_q y_q + \bar{S}_q \Omega(y_q) + \bar{q},$$

then similar to (2.37) one has that K is given by

$$K = \bar{R}_q - \bar{S}_q (G\bar{S}_q)^{-1} G\bar{R}_q \tag{2.46}$$

and $\dot{K}\dot{y}_q$ is given by

$$\dot{K}\dot{y}_q = -\bar{S}_q (G\bar{S}_q)^{-1} \dot{G}v. \tag{2.47}$$

Note that the calculation of K in (2.46) requires the inversion of an m×m matrix as does the evaluation of the Lagrange multipliers in equation (2.4); therefore, the computational cost of the two formulations of the underlying ODE are roughly the same in this respect. Also, note that both (2.45) and (2.45) allow for the computation of the product $\dot{G}v$ rather than just $\dot{G}$ alone. This is very significant from a numerical point of view as the calculation of $\dot{G}$ alone can be computationally expensive.

2.4.1 Example

Consider the 2dof mechanism in Fig.2.4. The system has four bodies with the four generalized coordinates chosen as $q = (q_1, q_2, q_3, q_4)^T$. Suppose the independent position

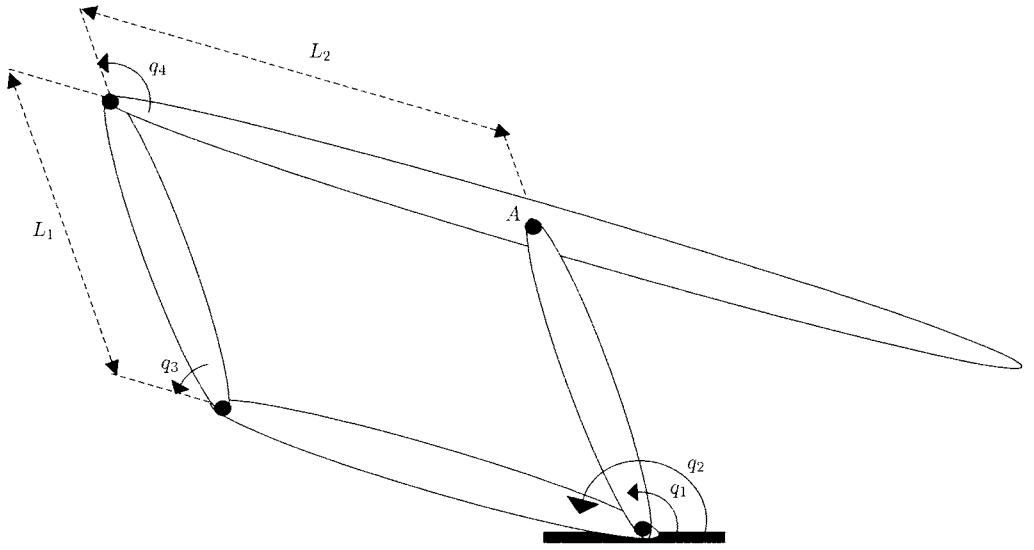


Figure 2.4 Parallel 4-bar robot

coordinates are chosen as $y_q = (q_1, q_2)^T$. Since the mechanism resembles a simple parallelogram, the analytic relationship of the generalized coordinates in terms of q_1 and q_2 is easily derived. From Fig.2.4 it is clear that $q_3 = q_1 - q_2$ and $q_4 = q_2 - q_1 - \pi$ and therefore the local parametrization on the position level is given by

$$\Psi_q(y_q) = \begin{pmatrix} q_1 \\ q_2 \\ q_1 - q_2 \\ q_2 - q_1 - \pi \end{pmatrix}. \tag{2.48}$$

From (2.40), an appropriate definition of K is $K := \partial\Psi_q/\partial y_q$ and so differentiating the analytic relationship (2.48) gives

$$K = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix}.$$

Assuming Ψ_q is not known explicitly (as is almost always the case), the derivation of K using the formulas given in this section begins with defining a set of constraints. Suppose the constraints are chosen to ensure that the first and forth bodies are coincident at point A,

$$g(q) = \begin{pmatrix} L_2\cos(q_2) + L_1\cos(q_2+q_3) + L_2\cos(q_2+q_3+q_4) - L_1\cos(q_1) \\ L_2\sin(q_2) + L_1\sin(q_2+q_3) + L_2\sin(q_2+q_3+q_4) - L_1\sin(q_1) \end{pmatrix} = 0.$$

With q_1 and q_2 still chosen as the independent position coordinates one has that $R_q(y_q) = (q_1, q_2)^T - y_q$ and from (2.44) K is given by

$$\begin{aligned} K &= \begin{pmatrix} \frac{\partial R_q}{\partial q} \\ \frac{\partial g}{\partial q} \end{pmatrix}^{-1} \begin{pmatrix} -\frac{\partial R_q}{\partial y_q} \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\partial g}{\partial q_1} & \frac{\partial g}{\partial q_2} & \frac{\partial g}{\partial q_3} & \frac{\partial g}{\partial q_4} \end{pmatrix}^{-1} \begin{pmatrix} I_2 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \frac{-\sin(q_1-q_2-q_3-q_4)}{\sin(q_4)} & -\frac{L_1\sin(q_4)+L_2\sin(q_3+q_4)}{L_1\sin(q_4)} \\ \frac{L_1\sin(q_1-q_2-q_3)+L_2\sin(q_1-q_2-q_3-q_4)}{L_2\sin(q_4)} & \frac{L_1\sin(q_3)+L_2\sin(q_3+q_4)}{L_1\sin(q_4)} \end{pmatrix}. \end{aligned}$$

Upon substitution of the analytic relationships $q_3 = q_1 - q_2$ and $q_4 = q_2 - q_1 - \pi$ this simplifies

to the relationship found before,

$$K = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix}.$$

Remark: The fact that K is constant (at all points on the invariant manifold) is a consequence of the linear relationship between the generalized and independent position coordinates. While prior knowledge of a constant matrix K represents a large computational advantage, one does not exist in general.

2.5 Final Remarks

In this chapter, a strong theoretical foundation for the local state-space form was established. When it is desirable for a local state-space ODE to be valid over a large subset of the manifold, the nonlinear setting of the change of coordinates (2.20) can incorporate an intuitive selection of valid independent coordinates. On the other hand, every current automated technique for selecting independent coordinates via matrix decompositions can be interpreted as subsets of the linear change or coordinates (2.32). Therefore, the theoretical foundation given here can help in the search for new automated techniques such as the one derived in Chapter 3.

CHAPTER 3. SINGULARITY ROBUST NULL SPACE FORMULATION

A common assumption in multibody dynamics is that the constraints are always independent, i.e., $G(q)$ in (2.1b) has full row rank for all q. In practice, however, there often exist finite positions known as kinematic singularities where $G(q)$ temporarily looses rank and numerical integration at or near these singularities becomes difficult. The difficulties largely stem from numerical ill-conditioning at positions near the kinematic singularities. For example, if G is nearly singular then the computation of the Lagrange multipliers in (2.4) given by

$$\lambda = (GM^{-1}G^T)^{-1}(GM^{-1}F + \dot{G}v).$$

becomes very sensitive to numerical errors in the vector $GM^{-1}F+\dot{G}v$ or to constraint violation errors. These numerical errors cause very large variations in the Lagrange multipliers (or, the *constraint forces*, $G^T\lambda$) which introduces an artificial stiffness in the numerical integration [48].

In this section a formulation based on the null space method of Section 2.4 which performs well near kinematic singularities and also has the ability to detect when the system constraints are redundant is given.

3.1 Using the Singular Value Decomposition (SVD) in the Null Space Formulation

The singular value decomposition of the constraint matrix G may be written as

$$G = USV^T$$

$$= U \left(\begin{array}{ccc|c} \sigma_1 & & 0 & \\ & \ddots & & 0_{\mathrm{n\times p}} \\ 0 & & \sigma_m & \end{array} \right) \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}$$

where $U \in \mathbb{R}^{\mathrm{m\times m}}$ and $V \in \mathbb{R}^{\mathrm{n\times n}}$ are unitary and the singular values $\sigma_1 \geq \ldots \geq \sigma_m \geq 0$. Since V is unitary one has that $V_1^T V_2 = 0$ and therefore right multiplying G by $V_2 \in \mathbb{R}^{\mathrm{n\times p}}$ leads to

$$\begin{aligned} GV_2 &= U \left(\begin{array}{ccc|c} \sigma_1 & & 0 & \\ & \ddots & & 0_{\mathrm{n\times p}} \\ 0 & & \sigma_m & \end{array} \right) \begin{pmatrix} V_1^T V_2 \\ V_2^T V_2 \end{pmatrix} \\ &= U \left(\begin{array}{ccc|c} \sigma_1 & & 0 & \\ & \ddots & & 0_{\mathrm{n\times p}} \\ 0 & & \sigma_m & \end{array} \right) \begin{pmatrix} 0 \\ I_{\mathrm{p}} \end{pmatrix} \\ &= 0. \end{aligned}$$

Therefore the singular value decomposition provides an orthonormal basis for the null space of G.

The fact that the columns of V_2 form a basis for the null space of G also implies that the matrix

$$\begin{pmatrix} V_2^T \\ G \end{pmatrix}$$

is nonsingular (assuming G has full row rank). Comparing to (2.28) this implies that a valid set of independent position coordinates may be defined as

$$R_q(q, y_q) = V_2^T q - y_q = 0 \tag{3.1}$$

or

$$y_q = V_2^T q.$$

With an appropriate set of coordinates defined in (3.1), the null space formulation of the

dynamics will now be examined. The matrix K using formula (2.44) is given by

$$K = \begin{pmatrix} V_2^T \\ G \end{pmatrix}^{-1} \begin{pmatrix} I_{\mathrm{p}} \\ 0 \end{pmatrix}.$$

Since $GV_2 = 0$ and the columns of V_2 are orthonormal, the inverse takes the particular form (see [49])

$$\begin{pmatrix} V_2^T \\ G \end{pmatrix}^{-1} = \begin{pmatrix} V_2^T \\ G \end{pmatrix}^{\dagger} = \begin{pmatrix} V_2 & G^{\dagger} \end{pmatrix}$$

where $\dagger$ denotes the Moore-Penrose generalized inverse and therefore

$$\begin{aligned} K &= \begin{pmatrix} V_2 & G^{\dagger} \end{pmatrix} \begin{pmatrix} I_{\mathrm{p}} \\ 0 \end{pmatrix} \\ &= V_2. \end{aligned} \tag{3.2}$$

Similarly, using formula (2.45) for $\dot{K}\dot{y}_q$ one has

$$\begin{aligned} \dot{K}\dot{y}_q &= -\begin{pmatrix} V_2^T \\ G \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \dot{G}v \end{pmatrix} \\ &= -\begin{pmatrix} V_2 & G^{\dagger} \end{pmatrix} \begin{pmatrix} 0 \\ \dot{G}v \end{pmatrix} \\ &= -G^{\dagger}\dot{G}v. \end{aligned} \tag{3.3}$$

It turns out that the generalized inverse of G may be given in terms of the previously calculated singular value decomposition

$$G^{\dagger} = VS^{\dagger}U^T \tag{3.4a}$$

$$= V \left(\begin{array}{ccc} 1/\sigma_1 & & 0 \\ & \ddots & \\ 0 & & 1/\sigma_m \\ \hline & 0_{\mathrm{p\times m}} & \end{array} \right) U^T \tag{3.4b}$$

$$= V_1 \begin{pmatrix} 1/\sigma_1 & & 0 \\ & \ddots & \\ 0 & & 1/\sigma_m \end{pmatrix} U^T. \tag{3.4c}$$

So the SVD serves a dual purpose, it identifies a set of independent coordinates (3.1) and also directly leads to expressions for K and $\dot{K}\dot{y}_q$. Finally, with both K and $\dot{K}\dot{y}_q$ defined in (3.2) and (3.3) the null space formulation of the underlying ODE (2.43) is complete and is given by

$$\dot{q} = v \tag{3.5a}$$

$$\dot{v} = V_2(V_2^T M V_2)^{-1} V_2^T (F + M G^\dagger \dot{G} v) - G^\dagger \dot{G} v. \tag{3.5b}$$

The advantage of this formulation is that the SVD becomes very useful at singular configurations. To begin with, it provides a reliable measure of how near the system is to a singular configuration, i.e., the ratio $\kappa = \sigma_1/\sigma_m$ is near 1 when G is well-conditioned and is very large when near a singular configuration. Additionally, when the system is precisely at a singular configuration one has that $\sigma_m = 0$, but, even in this situation the Moore-Penrose generalized inverse in (3.4) can still be computed by setting $1/\sigma_m = 0$. Consequently, an evaluation of the underlying ODE (3.5) does not fail when G temporarily looses rank.

3.1.1 Additional Comments on the Generalized Inverse

Although $\dot{K}\dot{y}_q$ in (3.3) can be interpreted as the least squares solution of $G\dot{K}\dot{y}_q = -\dot{G}v$, there may still exist large variations or *spikes* in its solution near singularities. This increased sensitivity should be clearly evident from (3.4), as the system nears a singularity the minimum singular value, σ_m, tends to zero and hence the term $1/\sigma_m$ in $G^\dagger$ becomes very large. More

specifically, using (3.4) to write $\dot{K}\dot{y}_q = -G^{\dagger}\dot{G}v$ as

$$\begin{aligned}
\dot{K}\dot{y}_q &= -V_1 \begin{pmatrix} 1/\sigma_1 & & 0 \\ & \ddots & \\ 0 & & 1/\sigma_m \end{pmatrix} U^T \dot{G}v \\
&= -\begin{pmatrix} v_1 & \cdots & v_m \end{pmatrix} \begin{pmatrix} 1/\sigma_1 & & 0 \\ & \ddots & \\ 0 & & 1/\sigma_m \end{pmatrix} \begin{pmatrix} u_1 & \cdots & u_m \end{pmatrix}^T \dot{G}v \\
&= -\sum_{i=1}^{m} \frac{1}{\sigma_i} v_i u_i^T \dot{G}v \qquad (3.6) \\
&= -\sum_{i=1}^{m-1} \frac{1}{\sigma_i} v_i u_i^T \dot{G}v - \frac{1}{\sigma_m} v_m u_m^T \dot{G}v \qquad (3.7)
\end{aligned}$$

illustrates that unless $v_m u_m^T \dot{G}v$ approaches zero at approximately the same rate as $1/\sigma_m$ approaches infinity, then $\dot{K}\dot{y}_q$ will be very large (implies large accelerations) and the calculation will be sensitive to numerical errors.

Of course, large accelerations can be expected near singularities. The near removal of a degree of degree of freedom may appear visually as a so-called *lock-up* of the mechanical system. Instead of moving through the singularity, a sudden change in direction away from it is observed.

If large accelerations are not observed, then the system likely drove through the singularity in a way complaint to the temporarily removed degree of freedom. This is the case where $v_m u_m^T \dot{G}v$ in (3.7) is approaching zero at approximately the same rate as $1/\sigma_m$ is growing ($\dot{G}v$ is either zero or in the null-space of $v_m u_m^T$).

3.1.2 Stabilization Artifacts

The form of stabilization used can introduce an additional set of difficulties near singularities. For example, suppose Baumgarte stabilization [18] is being used where the acceleration

constraint (2.3) is replaced by

$$0 = \ddot{g} + 2\alpha\dot{g} + \beta^2 g \qquad \alpha, \beta > 0 \tag{3.8}$$

which makes $\mathcal{M}$ an *attracting* manifold. The result of using this type of stabilization is that the formula for $\dot{K}\dot{y}_q$ is now given by

$$\dot{K}\dot{y}_q = -G^{\dagger}(\dot{G}v + 2\alpha\dot{g} + \beta^2 g). \tag{3.9}$$

If the constraints $g = \dot{g} = 0$ are not precisely satisfied then comparing to (3.7) reveals that these terms may be amplified near a singularity which can introduce stiffness in the solution of the underlying ODE. These *remnants* or *artifacts* of the form of stabilization used (and any other numerical errors, in general) may lead to meaningless results. Even if the simulation appeared to have been stable when travelling through a singularity, such sensitivities to error may lead to solutions which violate laws such as conservation of energy.

3.1.3 Projection

One way of alleviating some of the aforementioned problems is the use of *projection* [21]. In projection, the solution at each time step is projected onto the nearest point on the invariant manifold. The method mainly consists of two steps for each timestep:

1. The discrete numerical solution $\tilde{x}_n = (\tilde{q}_n\, \tilde{v}_n)$ of the underlying ODE is computed from an integration routine using projected values obtained at past timesteps.

2. The solution $\tilde{x}_n$ is then projected orthogonally back onto the manifold given by invariants, i.e., the projected solution is the solution of

$$\|x_n - \tilde{x}_n\|_2 = \min_{x_n} \tag{3.10}$$

$$h(x_n) = 0. \tag{3.11}$$

These projected values are used to advance the solution.

An implementation of (3.10) and (3.11) using a Newton-type method fits very nicely into this formulation since it can be formulated using the Moore-Penrose generalized inverse of G. Suppose the projected solution on the position level is written as $q_n = \tilde{q}_n + \delta q_n$, and consider the approximation

$$0 = g(q_n) = g(\tilde{q}_n + \delta q_n) \approx g(\tilde{q}_n) + G\delta q_n.$$

The difference from the projected solution, δq_n, can be minimized in a least squares sense using the Moore-Penrose generalized inverse of G in the solution of the previous equation,

$$\delta q_n = -G^{\dagger} g(\tilde{q}_n).$$

The current projected solution is updated as $q_n = \tilde{q}_n + \delta q_n$ and this process may be repeated until some measure of convergence is obtained. Finally, using the projected solution q_n, the projection onto the velocity invariants (2.2) may be computed in a single step as

$$v_n = \tilde{v}_n - G^{\dagger}(q_n) G(q_n) \tilde{v}_n.$$

As illustrated in (3.9), unwanted remnants of the stabilization have a greater tendency to appear near singularities. Using projection, the positions and velocities are always physically feasible and any stiffness caused by drift from the manifold (*artificial lock-up*) is minimized. Using Baumgarte stabilization, one can only say that $h(x(t)) \to 0$ as $t \to \infty$ and the magnitude of the constraint violation at the specific times near singularities cannot be controlled. Also, using projection one can project onto additional invariants such as conservation of energy which has a tendency to be violated due to the numerical errors near singularities.

3.2 Examples

In this section the five-bar mechanism in Fig.3.1 is considered (from [50]). The parameters are given in Table 3.1. The generalized position coordinates are chosen as $q = (q_1, q_2, q_3, q_4)^T$

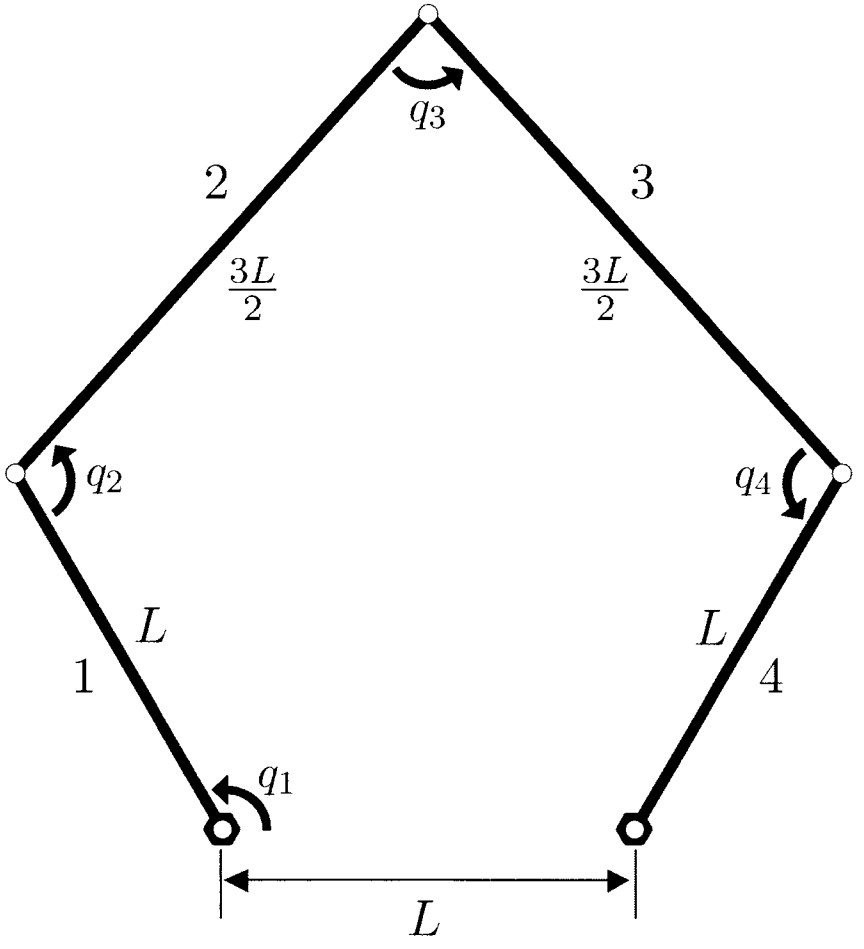


Figure 3.1 Five-bar mechanism

and two closed loop constraints prohibit translational motion at the lower right support. The constraint jacobian is singular at $q = (\pi, 0, 0, \pi)^T$ (see Fig.3.2). A fourth order Runge-Kutta integrator with a timestep of $h = 0.001$s was used in all simulations.

In the first test, the basic non-failure of the formulation at singularities is illustrated. No form of stabilization will be used. The initial condition of the mechanism is $q_o = (\pi, 0, 0, \pi)^T$ (the singular configuration) and $\dot{q} = 0$. The mechanism is simulated for 5 seconds and the resulting joint angle trajectories can be seen in Fig.3.3. Basically, the mechanism falls, bobs up and down once, then returns to the original singular configuration (at 2.82s) to repeat the motion over again. The condition number of G is plotted in Fig.3.4. The very high peaks

Table 3.1 Five-bar mechanism physical parameters.

Link	Length (m)	Mass (kg)	Inertia ($kg \cdot m^2$)
1 & 4	$L = 1.0$	2.0	1.0
2 & 3	$3L/2 = 1.5$	3.0	2.0

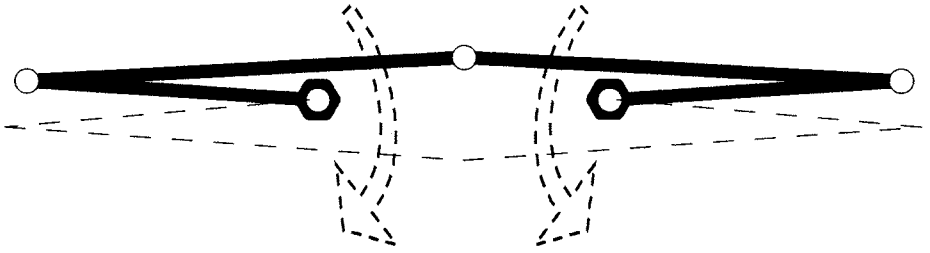


Figure 3.2 Illustration of five-bar mechanism going through singular point

at 0s and 2.82s illustrate the ill-conditioning of G near the singular configuration. Figure 3.5 shows that the change in energy over time is very small (the system is conservative). Along with a plot of the constraint violation Fig.3.6, these are a good indication of the accuracy of the simulation.

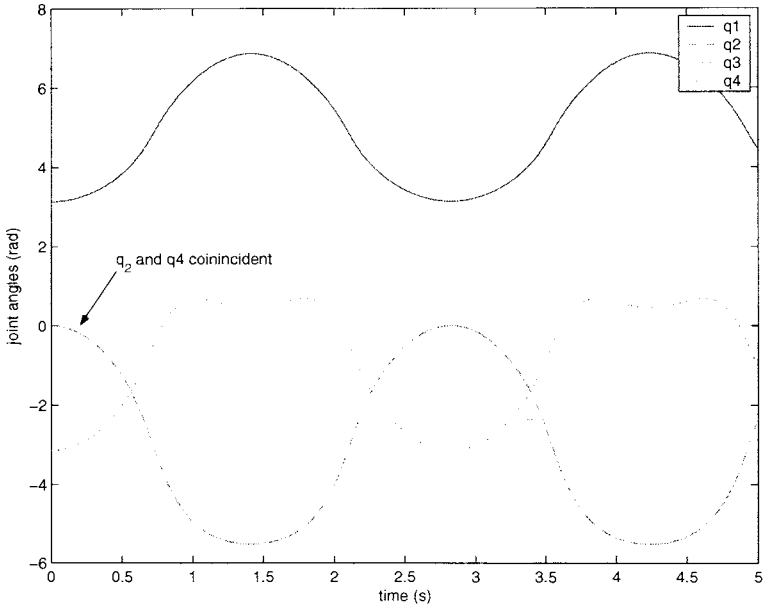


Figure 3.3 Joint trajectories of five-bar mechanism (ex.1)

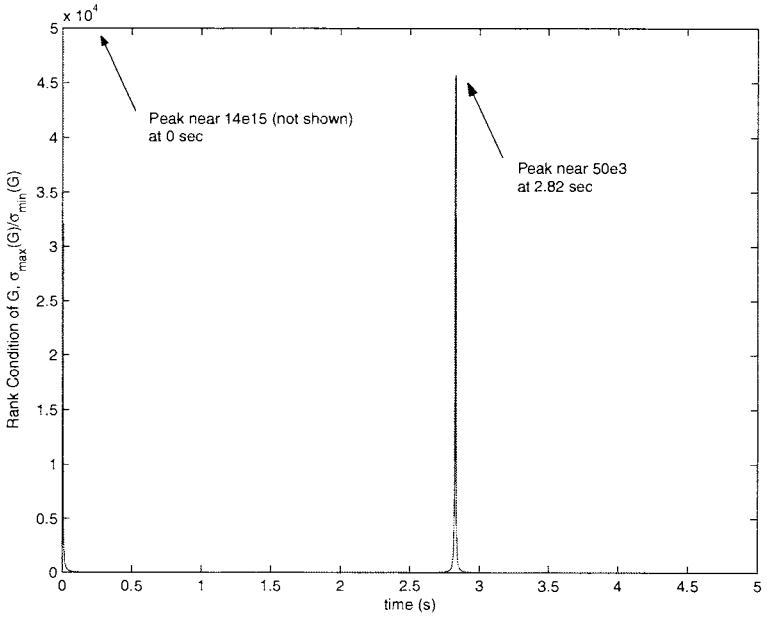
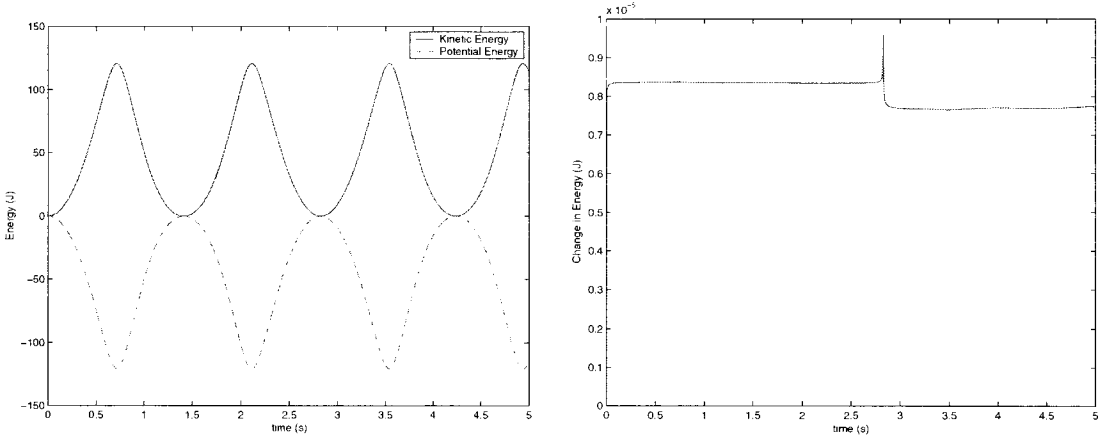


Figure 3.4 Constraint matrix condition number of five-bar mechanism (ex. 1)



(a) (b)

Figure 3.5 (a) Kinetic, potential, and (b) change in energy of five-bar mechanism (ex.1)

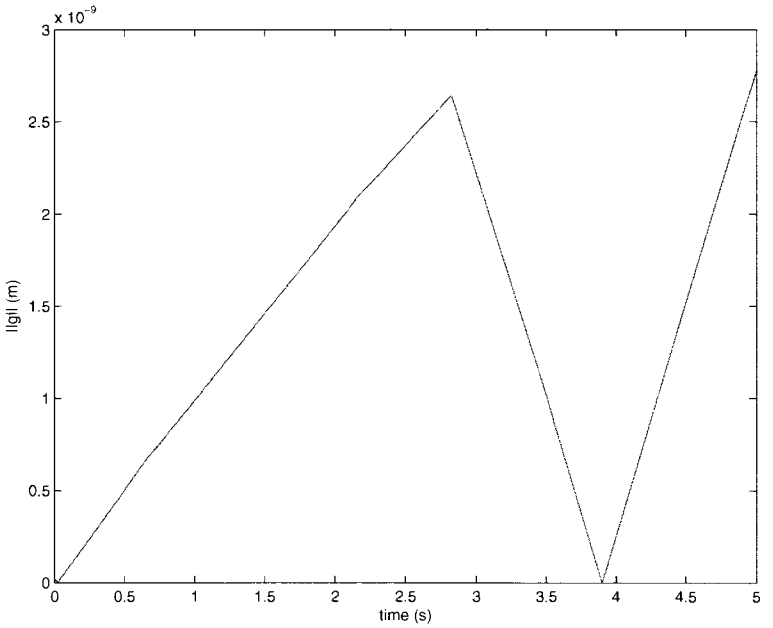


Figure 3.6 Constraint violation of five-bar mechanism (ex.1)

While the previous simulation illustrated that the formulation does not fail at singularities, it was not a worst case scenario since the velocity was more or less zero near the singular points. In this second example, the initial condition is as in Fig.3.1 with $q_o = (2\pi/3,\ \pi/3+\cos^{-1}(2/3),\ 2\sin^{-1}(2/3),\ \pi/3+\cos^{-1}(2/3))^T$ (symmetric about vertical axis) and $\dot{q}_o = 0$. During the 5 second simulation the mechanism may go through the singularity as many as three times. It is worth noting that the singularity represents a bifurcation point and therefore the response is very sensitive to any numerical perturbations in the solution near it. Also keep in mind that the solution is very sensitive to numerical tolerances in the integration and constraint stabilization.

Two different approaches will be compared. In the first approach the classical Lagrange multiplier method (2.5) is used with Baumgarte stabilization (3.8) where $\alpha = \beta = 10$. In the second approach the SVD-based null space formulation (3.5) and projection stabilization method of Section 3.1.3 are used. The solution is also projected onto the conservation of energy invariant (all projection steps are done simultaneously). The projection step is considered convergent when the norm of the invariants is less than 10^{-3}.

In both approaches, the joint trajectories are similar to Fig.3.7. The constraint drift

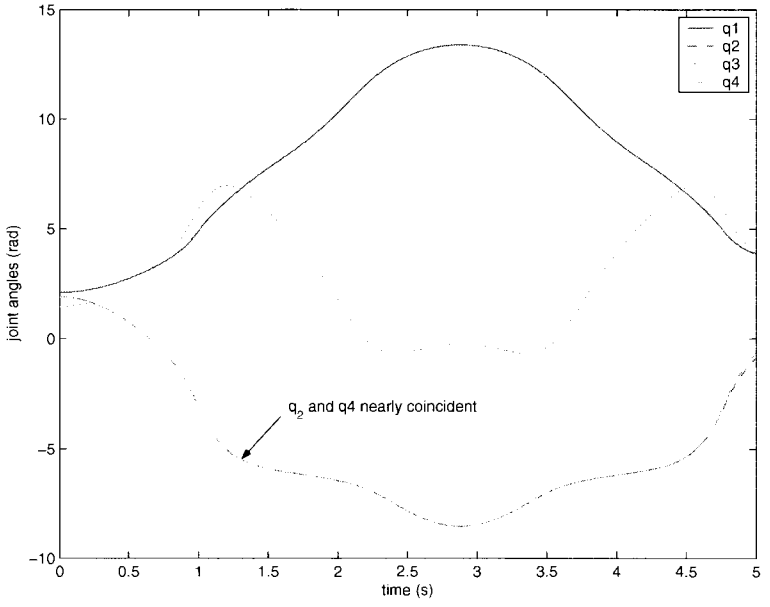
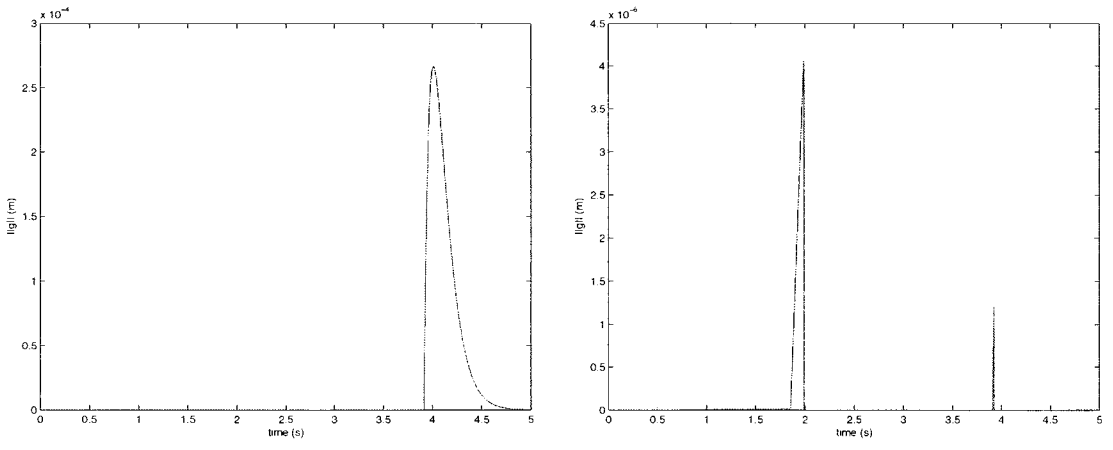


Figure 3.7 Joint trajectories of five-bar mechanism (ex.2)

can be seen if Fig.3.8. The significant increase in the constraint drift just before 4s was induced by the large accelerations encountered when passing through the singularity. The large accelerations can be partly attributed to ill-conditioning in the computation of the Lagrange multipliers (2.4) at the singularity. The Lagrange multipliers can be seen in Fig.3.9a, notice the large spike just before 4s.

In Fig.3.9b the conditioning of the SVD-based null space formulation is illustrated. The plot shows that $v_m u_m^T \dot{G} v$ (3.7) approaches zero near the singularities and indicates that no significantly large spikes in the acceleration occurred.

In Fig.3.10 the main difference between the two results is illustrated. While the Lagrange multiplier method with Baumgarte stabilization violates conservation of energy, the SVD-base null space formulation with projection fairs much better. Even without projection onto the conservation of energy invariant, the results for the SVD-based null space formulation are much better.



(a) (b)

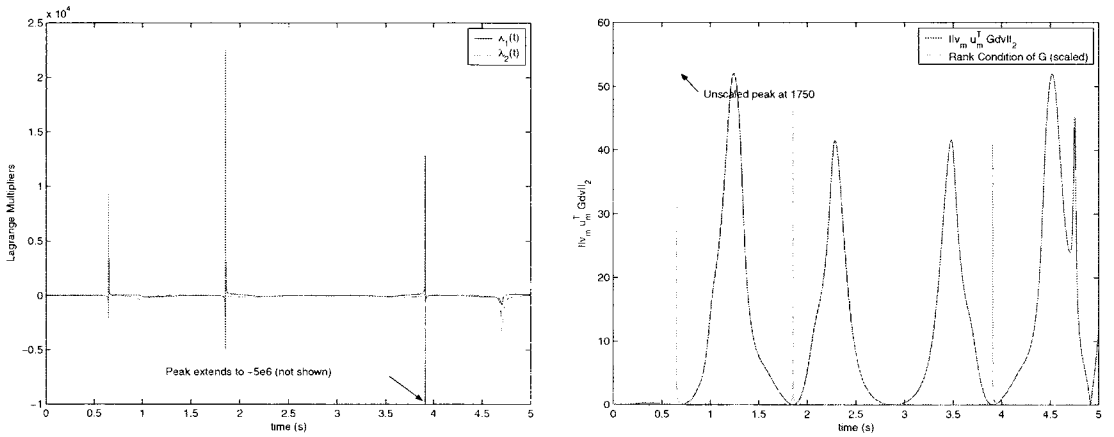
Figure 3.8 Constraint drift: (a) Using Baumgarte (b) Using projection

3.3 Final Remarks

The previous examples have illustrated that the SVD-based null space formulation can handle many simulations where singularities occur. In particular, the formulation performs very well in simulations where large variations in the acceleration do not occur near the singularities. When large accelerations do occur, equations of motion become stiff and the integration method is the most important factor.

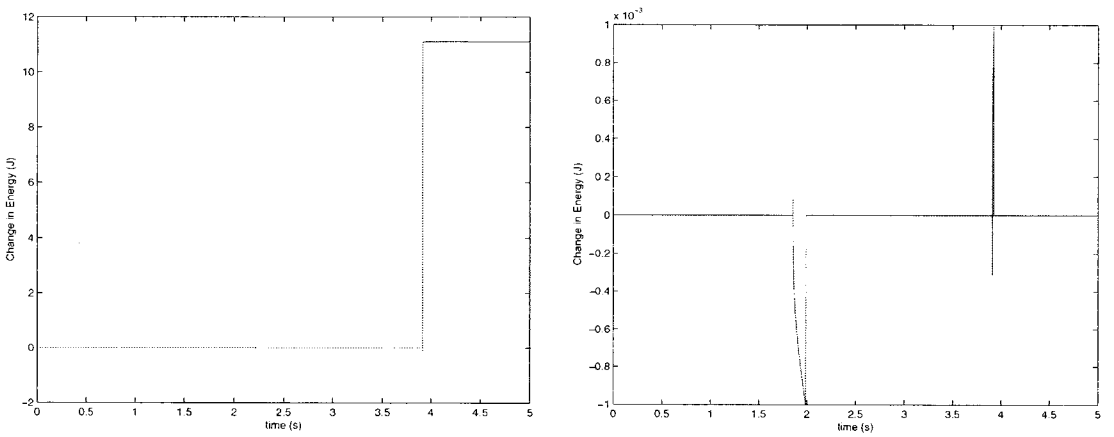
It has been suggested that a simple way of guaranteeing some stability in the presence of singularities is to *soften* (or, *ignore*) the constraints [29]. For example, changing how small σ_m must be before G is considered singular (when computing $G^\dagger$) can be interpreted as a softening of the constraints. A closely related and popular way of softening the constraints is to use the damped least-squares inverse [51] instead of the Moore-Penrose generalized inverse. While such modifications may produce stable results, their accuracy cannot be guaranteed.

It is important to note that this method does not explicitely deal with the singularities; it simply provides a convenient numerically well-conditioned approach via the singular value decomposition. For methods which deal directly with the singularity see [52] and the references therein.



(a) (b)

Figure 3.9 Indicators of large accelerations: (a) Lagrange multiplier method (b) SVD null space formulation



(a) (b)

Figure 3.10 Conservation of energy: (a) Lagrange multiplier method (b) SVD null space formulation

CHAPTER 4. RECURSIVE LINEARIZATION OF ROBOTS CONTAINING CLOSED CHAINS

The following chapter develops a recursive algorithm for linearization of the underlying ODE of general mechanical systems containing closed chains. Recursive algorithms from the field of robotics are chosen for the numeric formulation of the dynamics and hence become the basis of the recursive linearization algorithm. A Lie algebraic formulation of the algorithms is undertaken to aid in a systematic differentiation of the dynamics.

The chapter begins with a general introduction to linearized mechanical models. The development of a recursive algorithm begins in Section 4.2. At some point additional background material dealing with modern robotics algorithms may need to be consulted and can be found in Appendix A.

4.1 The Linearized Underlying ODE

In this section the linearized underlying ODE is introduced and some fundamental properties are discussed.

Recall that the underlying ODE of a constrained mechanical system (2.6) can be written as

$$\dot{x} = f(x) \tag{4.1}$$

where $x = (q^T \, v^T)^T \in \mathbb{R}^N$ and for simplicity time invariance has been assumed (implies that there are no external forces). The underlying ODE also satisfies an M-dimensional vector of invariants

$$h(x) = \begin{pmatrix} g(q) \\ G(q)v \end{pmatrix} = 0$$

where $H(x) := \partial h/\partial x \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$ has full rank for all $x \in \mathcal{M}$. The linearization of (4.1) at some point $x_o \in \mathcal{M}$ is the linear ODE

$$\dot{x} = f(x_o) + A(x - x_o) \tag{4.2}$$

with $A \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ such that (4.2) is a first-order approximation of (4.1) at x_o.

First, let's start with how *not* to find the linearized underlying ODE. At first glance it might be tempting to say that $A \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ is simply the jacobian of f whose i^{th} column is defined as

$$[A]_{col,i} = \lim_{\delta\to 0} \frac{f(x_o + \delta e_i) - f(x_o)}{\delta}$$

where e_i denotes the i^{th} unit vector. The problem with this approach is that $x_o + \delta e_i$ does not necessarily satisfy $h(x_o + \delta e_i) = 0$ and therefore ignores the constraints. The thing to keep in mind is that the coordinates are not independent; a more correct numerical approach would perturb *only* an independent set of coordinates. This quickly leads to the involvement of *local parameterizations* of Section 2.2.

A numerical approach to finding the linearized model which utilizes a local parametrization of the manifold is given now. It will be assumed that the coordinates of the local parametrization, Ψ, are defined as

$$y = Rx$$

(the linear form of (2.20)) where $R \in \mathbb{R}^{\mathrm{P}\times\mathrm{N}}$ ($\mathrm{P} := \mathrm{N} - \mathrm{M}$). Let Ψ be a local parametrization at $x_o = \Psi(y_o) \in \mathcal{M}$ and note that $\Psi(Rx) \in \mathcal{M}$ for all x in some neighborhood of x_o; therefore, applying the chain rule of differentiation to $f(\Psi(Rx))$ at x_o gives an appropriate definition of A at x_o

$$A = [Df]\frac{\partial\Psi}{\partial y}R \tag{4.3}$$

where $[Df] \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ denotes the usual jacobian of f. As an alternative to numerically finding $[Df]$, one may choose to find $\partial f/\partial y = [Df]\partial\Psi/\partial y \in \mathbb{R}^{\mathrm{N}\times\mathrm{P}}$ whose i^{th} column may be found as

a finite difference approximation of

$$\left[\frac{\partial f}{\partial y}\right]_{col,i} = \lim_{\delta \to 0} \frac{f(\Psi(y_o + \delta e_i)) - f(x_o)}{\delta}.$$

This approach to finding A requires less evaluations of the underlying ODE (but also involves evaluations of Ψ).

The representation of the linearized model (4.3) depends on a particular local parametrization Ψ. A coordinate invariant version can be derived as follows. Since $R\frac{\partial \Psi}{\partial y} = I_{\mathrm{P}}$ and $H\frac{\partial \Psi}{\partial y} = 0$ for any local parametrization, (4.3) can be generalized as

$$A = [Df]P \tag{4.4}$$

for any matrix $P \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ such that:

- $HP = 0$
- $P \cdot P = P$
- $\mathrm{rank}(P) = \mathrm{N\text{-}M}$

The matrix P is a *projection* onto the null space of H (the *tangent space*) and therefore any perturbation δx such that $H(x_o)\delta x = 0$ satisfies $\delta x = P\delta x$. Each matrix P can be interpreted as an identity transformation on the tangent space at x_o. Note that choosing $P = I - H^T(HH^T)^{-1}H$ satisfies all of the conditions. This choice for P also provides a continuous representation of the sensitivity, A, over all points on the invariant manifold (it is relative to always choosing R such that $HR^T = 0$).

One property which is invariant under a particular change of coordinates are the eigenvalues of A at an equilibrium.

Theorem 4.1.1: At a point $x_o \in \mathcal{M}$ such that $f(x_o) = 0$ (equilibrium point), the eigenvalues of any representation of A are invariant.

Proof: Let Ψ be a local parametrization at x_o. Differentiating $h(\Psi(y)) = 0$ with respect to

time gives $H(\Psi(y))\frac{\partial \Psi}{\partial y}(y)\dot{y} = 0$ or

$$H(\Psi(y))f(\Psi(y)) = 0 \qquad , \qquad y \in \mathcal{Y}. \tag{4.5}$$

For all x in some neighborhood of x_o it holds that $H(\Psi(Rx))f(\Psi(Rx)) = 0$ and differentiating with respect to x at x_o leads to (uses fact that $f(x_o) = 0$)

$$H(x_o)[Df(x_o)]P = 0$$

where $P = (\partial \Psi(Rx_o)/\partial y)R$.

Let $x \in \mathbb{R}^{\mathrm{N}}$ be an eigenvector of $[Df]P$ for some nonzero eigenvalue then

$$H[Df]Px = \lambda Hx = 0$$

which implies $x \in \mathrm{Null}(H)$. Since every P is a projection onto the null space of H then $x = Px$. Therefore,

$$[Df]Px = [Df]x = \lambda x.$$

Now, suppose P_1 and P_2 correspond to any two choices of coordinates then

$$[Df]P_1x = [Df]P_2x = [Df]x = \lambda x.$$

Therefore, if λ is a nonzero eigenvalue of $A_1 = [Df]P_1$ then it is also an eigenvalue of $A_2 = [Df]P_2$ and any other linearization as well. □

The result of the previous theorem should not be surprising. Clearly the stability and dynamic properties implied by the eigenvalues should be coordinate invariant.

Recall that the local state-space form (2.16) of Section 2.2 is a minimal representation of the underlying ODE. Since it represents the same dynamical system, it too should have the same (nonzero) eigenvalues at equilibrium.

Theorem 4.1.2: The eigenvalues of A are the union of the eigenvalues of the linearized local state-space form, $\hat{A} = \partial \hat{f}/\partial y$, and $\{0\}$.

Proof: Differentiating (2.12) written as $f(\Psi(Rx)) = \partial\Psi/\partial y(Rx)\hat{f}(Rx)$ gives

$$A = \frac{\partial \Psi}{\partial y}\hat{A}R + \frac{\partial \dot{\Psi}}{\partial y}R \tag{4.6}$$

where at equilibrium $\partial\dot{\Psi}/\partial y = 0$ and so

$$A = \frac{\partial \Psi}{\partial y}\hat{A}R. \tag{4.7}$$

Right multiply (4.7) by $\partial\Psi/\partial y$ then since $R(\partial\Psi/\partial y) = I_{\mathrm{P}}$,

$$A\frac{\partial \Psi}{\partial y} = \frac{\partial \Psi}{\partial y}\hat{A}.$$

Let $y \in \mathbb{R}^{\mathrm{P}}$ be such that $\hat{A}y = \lambda y$ with $\lambda \neq 0$, then

$$A\left(\frac{\partial \Psi}{\partial y}y\right) = \lambda\left(\frac{\partial \Psi}{\partial y}y\right)$$

which implies λ is an eigenvalue of A and therefore all nonzero eigenvalues of $\hat{A}$ are eigenvalues of A. Similarly, by left multiplying by R it can be shown that all nonzero eigenvalues of A are eigenvalues of $\hat{A}$. Finally, from (4.7) it is clear that $\mathrm{rank}(A) = \mathrm{rank}(\hat{A}) \leq \mathrm{P}$ and therefore $A \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ must have at least $\mathrm{M} = \mathrm{N} - \mathrm{P}$ zero eigenvalues. □

From (4.3) and (4.4) it is clear that the matrix $A \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ has at most rank P and therefore has at least M zero eigenvalues. The additional zero eigenvalues are a result of the differentiation of the constraints (2.2-2.3) in forming the underlying ODE. If the solution of $\dot{x} = f(x)$ drifts away from the manifold, they can easily become nonzero and can lead to instability. This highlights the need for stabilization. Some forms of stabilization can be interpreted as replacing the M zero eigenvalues with negative real roots. This stabilizes the constraint drift but may also introduce unnecessary stiffness [53].

This section has discussed methods for linearizing the underlying ODE of a mechanical system. It was shown that the standard numerical techniques of linearizing a model had to be modified since the underlying ODE lies on a manifold. These modifications result in a linearized model which is not unique but does possess certain attributes at equilibrium. Finally, the inherent instability of the underlying ODE was highlighted by the set of zero eigenvalues seen in the linearized model.

In this section it was assumed that the limit operators can be approximated by standard numerical approaches. For example, the i^{th} column of $[Df]$ can be approximated by

$$\frac{\partial f}{\partial x_i} \approx \frac{1}{2\epsilon}(f(x_o + \epsilon\, e_i) - f(x_o - \epsilon\, e_i))$$

where ϵ is a small positive parameter [17]. Such an approximation of $[Df]$ would require at least 2N evaluations of the underlying ODE. If the jacobian of the underlying ODE is used in an implicit integration or optimization problem, the forming of $[Df]$ would likely account for the bulk of the computational burden.

The remainder of this chapter deals with an analytic derivation of the linearized model. Besides increased accuracy, if the numerical expense of the analytic derivation is relative to perhaps only 2-4 evaluations of the underlying ODE, then compared to 2N evaluations it would represent a significant computational advantage.

4.2 Structure of the Linearized Local State-Space Form

From Section 2.4 the null space formulation of the local state-space form, $\dot{y} = \hat{f}(y)$, is

$$\dot{y}_q = y_v \tag{4.8a}$$

$$\dot{y}_v = (K(y_q)^T M(q) K(y_q))^{-1} K(y_q)^T (F(q,v) - M(q)\dot{K}(y) y_v) \tag{4.8b}$$

where to simplify things the equations have been assumed autonomous and without external forces. It is assumed that K is locally defined by a linear change of coordinates (see (2.27-2.28)),

$$\begin{pmatrix} y_q \\ y_v \end{pmatrix} = \begin{pmatrix} R_q & 0 \\ 0 & R_q \end{pmatrix} \begin{pmatrix} q \\ v \end{pmatrix}.$$

These also define a local parametrization Ψ_q such that $q = \Psi_q(y_q)$ and $v = \partial\Psi_q(y_q)/\partial y\, \dot{y_q} = K(y_q)y_v$. At times q and v may appear as in (4.8) where it is understood that they are implicitly defined by y_q and y_v (and hence the ODE (4.8) is implicit).

The dynamic equations of the ODE (4.8) will be formulated using the Newton-Euler algorithm from robotics (see Appendix A) which can be viewed as the function

$$\tau_{NE}(q, v, \dot{v}) = M(q)\dot{v} - F(q, v). \tag{4.9}$$

If the equations of motion are written with this algorithm in mind they take the form

$$\dot{y}_q = y_v \tag{4.10a}$$

$$\dot{y}_v = -(K^T M K)^{-1} K^T \tau_{NE}(q, v, \dot{K} y_v) \tag{4.10b}$$

where the i^{th} column of $K^T M K \in \mathbb{R}^{\mathrm{p}\times\mathrm{p}}$ can be solved as

$$\left[K^T M K\right]_{col,i} = K^T \left(\tau_{NE}(q, v, Ke_i) - \tau_{NE}(q, v, 0)\right).$$

With the equations of motion written completely in terms of τ_{NE} and K, they will be linearized to see which derivatives need to be derived. The first order approximation of the local state-space form at y_o is

$$\dot{y} = \hat{f}(y_o) + \hat{A}\,(y - y_o)$$

where the $\mathrm{P} \times \mathrm{P}$ matrix $\hat{A} := \partial\hat{f}(y_o)/\partial y$ is obviously the object of primary interest. It is easily

shown to be of the general form

$$\hat{A} = \begin{pmatrix} 0 & I_{\mathrm{p}} \\ \hat{A}_{21} & \hat{A}_{22} \end{pmatrix}.$$

Here $\hat{A}_{21}$ represents the sensitivity of (4.10b) with respect to y_q and $\hat{A}_{22}$ represents the sensitivity of (4.10b) to y_v. They are

$$\hat{A}_{21} = -(K^T M K)^{-1} \left(\begin{bmatrix} \frac{\partial K^T}{\partial y_{q_1}} \tau_{NE} & \cdots & \frac{\partial K^T}{\partial y_{q_{\mathrm{p}}}} \tau_{NE} \end{bmatrix} + K^T \left(\frac{\partial \tau_{NE}}{\partial q} K + \frac{\partial \tau_{NE}}{\partial v} \dot{K} + M \ddot{K} \right) \right)$$

and

$$\hat{A}_{22} = -(K^T M K)^{-1} K^T \left(\frac{\partial \tau_{NE}}{\partial v} K + 2 M \dot{K} \right)$$

where in all cases τ_{NE} is computed at $(q, v, \dot{v})$. A derivation of $\hat{A}_{21}$ and $\hat{A}_{22}$ can be found in Appendix B. The case when external forces are present can also be found in the derivations in Appendix B.

The remainder of this chapter deals with the analytic derivation of the derivatives found in the expressions $\hat{A}_{21}$ and $\hat{A}_{22}$. The partial derivatives of τ_{NE} are covered next in Section 4.3. The partial derivatives and time derivatives of K are covered in Section 4.4. Both derivations are straightforward using the Lie algebraic formulation of the kinematics and dynamics.

4.3 Partial Derivatives of the Newton-Euler Algorithm

As stated previously, the Newton-Euler algorithm can be viewed as the function $\tau_{NE}(q, v, \dot{v}) = M(q)\dot{v} - F(q, v)$ but the actual algorithm is implemented numerically as (see Appendix A):

Forward Recursion

$$V_i = \mathrm{Ad}_{f_{P_i,i}^{-1}} V_{P_i} + S_i v_i$$
$$\dot{V}_i = \mathrm{Ad}_{f_{P_i,i}^{-1}} \dot{V}_{P_i} + S_i \dot{v}_i - \mathrm{ad}_{S_i v_i} V_i$$

Backward Recursion

$$F_i = \sum_{k \in C_i} \mathrm{Ad}^*_{f^{-1}_{i,k}} F_k + I_i \dot{V}_i - \mathrm{ad}^*_{V_i} I_i V_i$$
$$\tau_i = S_i^T F_i$$

In this algorithm, τ_i represents the i^{th} element of the vector function τ_{NE}. The additional terms such as S_i and I_i are related to the physical description of the robot and are defined in Appendix A.

In the previous algorithm, all joints are assumed to have only one degree of freedom. This assumption greatly simplifies the analytic derivatives and allows identities (A.1)-(A.4) in Appendix A to be directly applied to find the analytic partial derivatives of τ_i with respect to the joint positions and velocities. Common identities used in deriving and simplifying the derivatives were:

- $\frac{\partial}{\partial q_i}\left(Ad_{f^{-1}_{P_i,i}}\right) = -\mathrm{ad}_{S_i} Ad_{f^{-1}_{P_i,i}}$ — This identity accounts for the derivative of the local coordinate transformation, $f_{P_i,i}(q_i)$, in the adjoint operator.
- $\frac{\partial}{\partial q_i}\left(Ad^*_{f^{-1}_{P_i,i}}\right) = -Ad^*_{f^{-1}_{P_i,i}} \mathrm{ad}^*_{S_i}$ — This identity accounts for the derivative of the local coordinate transformation, $f_{P_i,i}(q_i)$, in the dual adjoint operator.
- $\frac{\partial}{\partial q_j}(V_i) = \frac{\partial}{\partial q_j}(\dot{V}_i) = 0$ if $j \in T_i$ — i.e., the instantaneous velocity of a body is not affected by the positions (or velocities) of its outer branches.
- $\mathrm{ad}_{S_i}(S_i\alpha) = 0$ — where $\alpha \in \mathbb{R}$. This identity allows the simplification: $\mathrm{ad}_{S_i} Ad_{f^{-1}_{P_i,i}} V_{P_i} = \mathrm{ad}_{S_i} V_i$.

The following algorithm uses these identities to find the partial derivative of the i^{th} element of τ_{NE} with respect to the j^{th} joint position coordinate:

Forward Recursion

$$\frac{\partial V_i}{\partial q_j} = \begin{cases} -\mathrm{ad}_{S_i} V_i & i = j \\ \mathrm{Ad}_{f_{P_i,i}^{-1}} \frac{\partial V_{P_i}}{\partial q_j} & i \in T_j \backslash \{j\} \\ 0 & i \notin T_j \end{cases}$$

$$\frac{\partial \dot{V}_i}{\partial q_j} = \begin{cases} -\mathrm{ad}_{S_i} \dot{V}_i & i = j \\ \mathrm{Ad}_{f_{P_i,i}^{-1}} \frac{\partial \dot{V}_{P_i}}{\partial q_j} - \mathrm{ad}_{S_i v_i} \frac{\partial V_i}{\partial q_j} & i \in T_j \backslash \{j\} \\ 0 & i \notin T_j \end{cases}$$

Backward Recursion

$$\frac{\partial F_i}{\partial q_j} = \begin{cases} \sum_{k \in C_i} \mathrm{Ad}^*_{f_{i,k}^{-1}} \frac{\partial F_k}{\partial q_j} + I_i \frac{\partial \dot{V}_i}{\partial q_j} - \mathrm{ad}^*_{\frac{\partial V_i}{\partial q_j}} I_i V_i - \mathrm{ad}^*_{V_i} I_i \frac{\partial V_i}{\partial q_j} & i \in T_j \\ \sum_{k \in C_i} \mathrm{Ad}^*_{f_{i,k}^{-1}} \frac{\partial F_k}{\partial q_j} - \left\langle \mathrm{Ad}^*_{f_{i,j}^{-1}} \mathrm{ad}^*_{S_j} F_j \right\rangle_{\text{if } j \in C_i, \text{ else zero}} & i \notin T_j \end{cases}$$

$$\frac{\partial \tau_i}{\partial q_j} = S_i^T \frac{\partial F_i}{\partial q_j}$$

The algorithm can be applied $(\mathrm{N}/2)^2$ times to find all partial derivatives of the elements of τ_{NE} and the matrix $\partial \tau_{NE} / \partial q$ can be formed as

$$\frac{\partial \tau_{NE}}{\partial q} = \begin{pmatrix} \frac{\partial \tau_1}{\partial q_1} & \frac{\partial \tau_1}{\partial q_2} & \cdots & \frac{\partial \tau_1}{\partial q_n} \\ \frac{\partial \tau_2}{\partial q_1} & & & \\ \vdots & & \ddots & \\ \frac{\partial \tau_n}{\partial q_1} & & & \frac{\partial \tau_n}{\partial q_n} \end{pmatrix}.$$

Similarly, the algorithm giving the partial derivatives with respect to the joint velocities was found as:

Forward Recursion

$$\frac{\partial V_i}{\partial v_j} = \begin{cases} S_i & i = j \\ \mathrm{Ad}_{f_{P_i,i}^{-1}} \frac{\partial V_{P_i}}{\partial v_j} & i \in T_j \backslash \{j\} \\ 0 & i \notin T_j \end{cases}$$

$$\frac{\partial \dot{V}_i}{\partial v_j} = \begin{cases} \mathrm{ad}_{V_i} S_i - \mathrm{ad}_{S_i v_i} \frac{\partial V_i}{\partial v_j} & i = j \\ \mathrm{Ad}_{f_{P_i,i}^{-1}} \frac{\partial \dot{V}_{P_i}}{\partial v_j} - \mathrm{ad}_{S_i v_i} \frac{\partial V_i}{\partial v_j} & i \in T_j \backslash \{j\} \\ 0 & i \notin T_j \end{cases}$$

Backward Recursion

$$\frac{\partial F_i}{\partial v_j} = \begin{cases} \sum_{k \in C_i} \mathrm{Ad}^*_{f_{i,k}^{-1}} \frac{\partial F_k}{\partial v_j} + I_i \frac{\partial \dot{V}_i}{\partial v_j} - \mathrm{ad}^*_{\frac{\partial V_i}{\partial v_j}} I_i V_i - \mathrm{ad}^*_{V_i} I_i \frac{\partial V_i}{\partial v_j} & i \in T_j \\ \sum_{k \in C_i} \mathrm{Ad}^*_{f_{i,k}^{-1}} \frac{\partial F_k}{\partial v_j} & i \notin T_j \end{cases}$$

$$\frac{\partial \tau_i}{\partial v_j} = S_i^T \frac{\partial F_i}{\partial v_j}$$

On their own, the previous two algorithms provide gradients of the inverse dynamics of tree-structured mechanisms. Similar algorithms were applied in [36] to find optimal weight lifting motions of serial robotic arms.

4.4 Derivatives of K

Much like the derivations in the previous section, the derivatives of K are found by differentiating the recursive algorithms used in the formulation of K. Such algorithms were outlined

in Section A.3.3 where the matrix K took the form

$$K = E \begin{pmatrix} R_q \\ A\,J \end{pmatrix}^{-1} \begin{pmatrix} I_p \\ 0 \end{pmatrix} \tag{4.11}$$

where A and E are constant matrices and $J = (J_1^T\, J_2^T\, \cdots)^T$ is a matrix comprising the set of body-fixed jacobians at each frame in the multibody system.

Since all matrices in (4.11) are constant other than J, the derivatives of K are directly related to those of J. The relationships are not difficult to derive and are found as:

- $\dfrac{\partial K}{\partial y_{q_i}} = \displaystyle\sum_j \frac{\partial K}{\partial q_j} K_{j,i}$ where $\dfrac{\partial K}{\partial q_j} = -E \begin{pmatrix} R_q \\ A\,J \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ A\frac{\partial J}{\partial q_j} \end{pmatrix} K$
- $\dot{K} = -E \begin{pmatrix} R_q \\ A\,J \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ A\dot{J} \end{pmatrix} K$
- $\ddot{K} = -E \begin{pmatrix} R_q \\ A\,J \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ A\ddot{J} \end{pmatrix} K - 2E \begin{pmatrix} R_q \\ A\,J \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ A\ddot{J} \end{pmatrix} \dot{K}$

Based on a physical description of the robot an algorithm for the body-fixed jacobians can be derived and can be differentiated using identities (A.1)-(A.4) to solve for $\frac{\partial J}{\partial q_j}$, $\dot{J}$, and $\ddot{J}$. These algorithms are described in the next section.

4.4.1 Derivatives of the Body-Fixed Jacobians

From Appendix A, the body-fixed jacobians are found recursively as:

Forward Recursion

$$J_i = \mathrm{Ad}_{f_{P_i,i}^{-1}} J_{P_i} + \begin{bmatrix} & | & \\ 0 \cdots 0 & S_i & 0 \cdots 0 \\ & | & \\ & i & \end{bmatrix}$$

The i^{th} jacobian is a mapping from the generalized velocity coordinates to the instantaneous velocity of the i^{th} frame expressed in frame $\{i\}$ coordinates (the *body-fixed* velocity representation). The algorithms for the derivatives of each J_i were derived and put into the following simplified forms:

Forward Recursion

$$\frac{\partial J_i}{\partial q_j} = \begin{cases} -\mathrm{ad}_{S_i} J_i & i = j \\ \mathrm{Ad}_{f_{P_i,i}^{-1}} \dfrac{\partial J_{P_i}}{\partial q_j} & i \in T_j^v \backslash \{j\} \\ 0 & i \notin T_j^v \end{cases}$$

Forward Recursion

$$\dot{J}_i = \mathrm{Ad}_{f_{P_i,i}^{-1}} \dot{J}_{P_i} - \mathrm{ad}_{S_i v_i} J_i$$

Forward Recursion

$$\ddot{J}_i = \mathrm{Ad}_{f_{P_i,i}^{-1}} \ddot{J}_{P_i} - \mathrm{ad}_{S_i v_i} \dot{J}_i - \mathrm{ad}_{S_i v_i} \mathrm{Ad}_{f_{P_i,i}^{-1}} \dot{J}_{P_i} - \mathrm{ad}_{S_i \dot{v}_i} J_i$$

4.5 Example

All of the previous algorithms were implemented in a personal dynamics library named RCF (Reduced Coordinate Formulations). A straightforward application of the algorithm is undergone in this example which essentially verifies that the algorithm is correctly implemented.

Under consideration is the John Deere 744 Loader linkage illustrated in Fig.4.1. The

Figure 4.1 John Deere 744 Loader

linkage contains 8 bodies, has 3 kinematic loops, and has 2 overall degrees of freedom. The model was input into the RCF library and the plot seen in Fig.4.2 was generated.

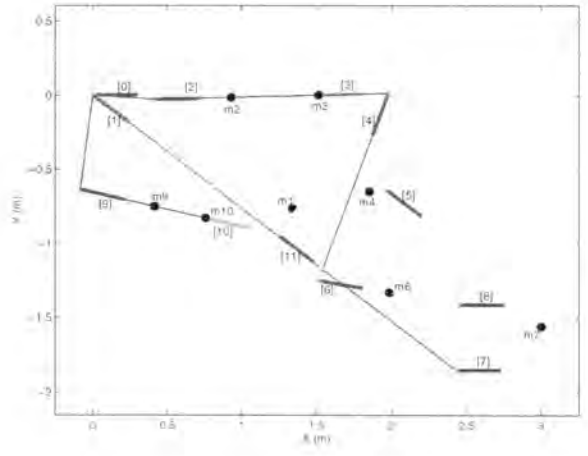
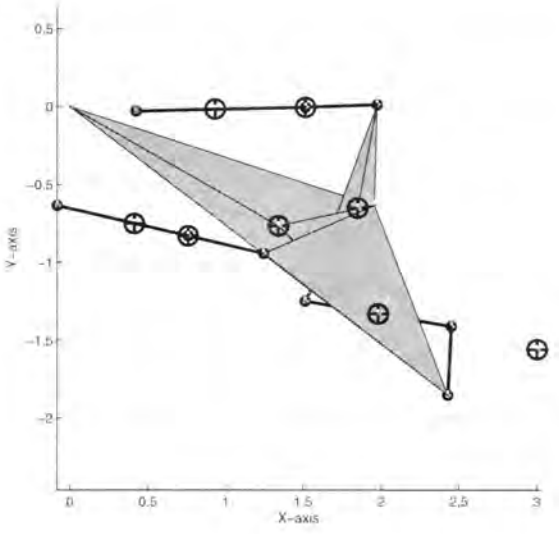
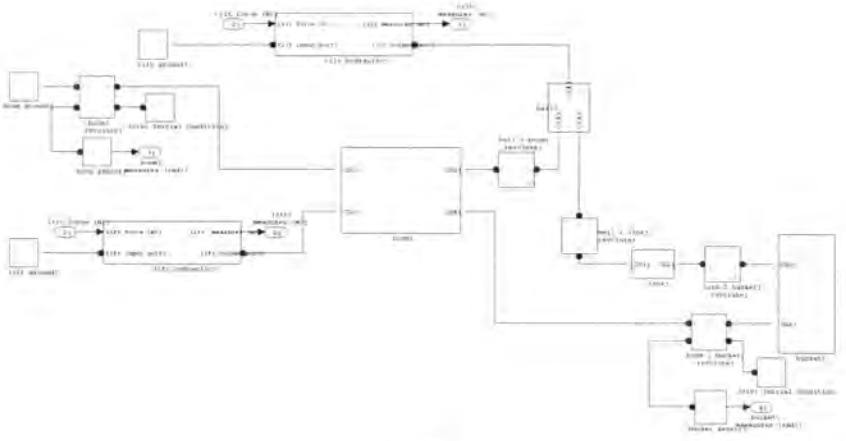


Figure 4.2 Plot of loader generated by RCF

For a comparison, the linkage was also modelled in the commercial package SimMechanics whose graphical plot and block diagram can be seen in Fig.4.3.



(a)



(b)

Figure 4.3 Loader representation in SimMechanics: (a) Graph, (b) Block Diagram

With the linkage in the position seen in Figures 4.1-4.3 it will be linearized using two different choices of independent coordinates. In both cases, the system has zero initial velocity and no external forces exist at the 2 cylinders (this is not an equilibrium point due to gravity). The first choice of coordinates are the angles of the boom and the bucket (rotation about frames

1 and 7 in Fig.4.2). The linearized local state-space form corresponding to these coordinates is found using the RCF library as

```
Ah = rcfLinLSSF(q,v,[0,0],'1 7')
Ah =

         0         0    1.0000         0
         0         0         0    1.0000
   -2.3763   -0.3816         0         0
    7.8530   -1.9778         0         0
```

which has eigenvalues $0.54643 \pm 1.5734i$ and $-0.54643 \pm 1.5734i$. If the independent position coordinates are chosen as the displacements of the cylinders (displacement of frames 3 and 10 in Fig.4.2) then the linearized local state-space form is found once again as

```
Ah = rcfLinLSSF(q,v,[0,0],'3 10')
Ah =

         0         0    1.0000         0
         0         0         0    1.0000
   -1.9649    2.5565         0         0
   -0.5873   -2.6457         0         0
```

which has eigenvalues $-0.37624 \pm 1.5642i$ and $-0.37624 \pm 1.5642i$

The commercial software SimMechanics uses a finite-difference technique to approximate the linearized model and allows the user to control which coordinates are perturbed. Under the first choice of coordinates the eigenvalues were found to be $0.5475 \pm 1.5729i$ and $-0.5475 \pm 1.5729i$. Under the second set of coordinates they were $0.37688 \pm 1.563i$ and $-0.37688 \pm 1.563i$. These values are very close to those obtained using RCF and are a good indication that it is correct.

4.6 Final Remarks

In the background material of Appendix A, algorithms from robotics were modified to cope with robots with kinematic loops. The algorithms still managed to maintain the highly

developed and simple notation first found in Featherstone [32] and later expressed as the Lie algebraic formulation [35, 54]. With all algorithms written in this context, the derivaties of the dynamics would always eventually boil down to applying identities (A.1)-(A.4). The original work which inspired this careful formulation of the dynamics can be found in [55].

The major applications of this work were never fully explored here. The first application is in the area of implicit integration where the efficient computation of the iteration jacobian is of great importance. For example, the iteration jacobian for the local state-space ODE with timestep h has the form

$$I_{\mathrm{P}} - h\hat{A}$$

which can be computed using the algorithms given here without the need for numerically expensive finite-difference approximations of the jacobian. The topic of implicit integration of multibody systems can be found in [12, 56]

The second area of application is in dynamic motion planning [38, 37] where the gradients provided here can be used in the optimization process. The primary focus of the previous work in this area was with open-chain robots. The work done here would allow robots with kinematic loops to be treated within the same context.

CHAPTER 5. CONCLUSIONS

Using local parameterizations of the invariant manifold, several applied areas in dynamics were discussed and expressed in a general framework. In Chapter 2, concepts from differential geometry were used to express the local state-space form in a very generic manner. All of the current applied techniques can be viewed as subsets of this general representation. These applied techniques were all discussed and a new one was proposed in Chapter 3 which was called the Singularity Robust Null Space Formulation. Using the concept of local parameterizations, the linearization of the underlying ODE and the local state-space form was discussed in Chapter 4. In addition, an efficient recursive algorithm for the construction of an exact linearization of the local state-space form was derived.

Chapter 3 uses the SVD as the basis of a null space formulation of the dynamics. It turns out that the SVD of the constraint jacobian directly led to an expression of the underlying/state-space ODE. The formulation also had the unique property of not failing at kinematic singularities. Simulations of a five-bar mechanism which travels through a singularity were performed to illustrate this property. These simulations also showed favorable results compared to traditional approaches which use Lagrange multipliers and Baumgarte stabilization.

Most formulations in dynamics ignore the topic of singularities even though they occur quite often in practice and in other DAE applications. The work done here was an approach which accepts that singularities will occur and does its best to "simulate through" them. At the same time, it had the ability to detect a singularity and could warn the user that the system may have gone through a bifurcation. The method showed some success due in part to the inherent numerical stability of the SVD. However, it was also shown how the simulation can still be prone to numerical errors in certain situations near singularities.

Chapter 4 covered the topic of linearization of ODE representations of constrained mechanical systems. The chapter utilized local parameterizations to derive an expression for the first-order approximation of the underlying ODE which lies on the invariant manifold. Next, a recursive algorithm for the linearized local state-space form was developed. Such an algorithm clearly depends on the particular formulation of the dynamics. A formulation from robotics known as the Lie algebraic formulation was chosen for its highly developed and easily manipulated notation. Using this, the derivation is fairly straightforward and is without the need for ad hoc notation. The algorithm showed application towards optimization and implicit integration problems where computational efficiency is of great importance.

One of the assumptions in the formulation is that all constraints can be described by simple joints. Such limitations are inherent to any approach which restricts itself to a particular formulation of the dynamics. On the other hand, most every dynamics package is restricted to a particular formulation and so it is surprising that the vast majority still rely on finite difference techniques to generate linearized models. Hence, the real contribution of this work is that it illustrates how linearization algorithms are not as complicated as they might have seemed when approached from the proper standpoint.

5.1 Future Work

There are several directions in which this work could be built upon:

- The formulations of the dynamics in Chapter 2 could be generalized to incorporate alternative basis vectors for the velocity space. These modifications would allow representations such as the canonical momenta and could lead to new approaches to forming the equations of motion.

- A numerical efficiency study of the linearization algorithm of Chapter 4 could be undergone to see when it is advantageous to use compared with finite difference techniques. Such a study would require a careful a careful implementation to minimize the number of additions and multiplications.

- The algorithms in Chapter 4 could be extended to handle multiple degree-of-freedom joints. These modifications would require that the partial derivative of the multiple degree-of-freedom joint screws be known.
- The algorithms in Appendix A could be further generalized to handle nonholonomic constraints.

APPENDIX A. ROBOTICS BACKGROUND MATERIAL

In this appendix, some necessary topics in the field of robot dynamics are introduced. For a complete understanding, the reader should be familiar with the work of Featherstone [32]. In Section A.1, the chosen mathematical language of robot dynamics, the *Lie algebraic formulation*, is briefly outlined. In Section A.2, the notation for a proper physical description of constrained robotic systems is described. Finally, Section A.3 introduces modern dynamic algorithms in robotics.

A.1 The Lie Algebraic Formulation

In the following the Lie algebraic formulation in robot kinematic and dynamics is introduced. The reader is referred to [57] and [58] for a more comprehensive treatment. The introduction here is very similar to those in [34] and [59].

A.1.1 The Special Euclidean Group $SE(3)$

The group of rigid body transformations on $\mathbb{R}^3$, denoted $SE(3)$, cast in a 4×4 homogeneous form, is

$$\begin{pmatrix} R & p \\ 0 & 1 \end{pmatrix}$$

where $p \in \mathbb{R}^3$ and R is a 3×3 rotation matrix such that $R^T R = I$ and $\det(R) = +1$.

A.1.2 The Lie Algebra $se(3)$

The Lie algebra of $SE(3)$ is called $se(3)$, and can be represented by the following 4×4 matrix form

$$\begin{pmatrix} [w] & v \\ 0 & 0 \end{pmatrix}$$

where $w, v \in \mathbb{R}^3$, and $[w] := \begin{pmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{pmatrix}$ is a 3×3 skew-symmetric matrix. Note that $se(3)$ can be regarded as 6-dimensional vector space, $(w, v) \in \mathbb{R}^6$. Define the $\vee$ (vee) operator as the extraction to the 6-dimensional representation as

$$\begin{pmatrix} [w] & v \\ 0 & 0 \end{pmatrix}^{\vee} = \begin{pmatrix} w \\ v \end{pmatrix} \in \mathbb{R}^6.$$

A.1.3 The Matrix Exponential

The principle relation between $SE(3)$ and $se(3)$ is the matrix exponential, the exponential map $\exp : se(3) \to SE(3)$ is defined as the following relation

$$\exp g = \begin{pmatrix} e^{[w]} & Av \\ 0 & 1 \end{pmatrix}$$

where the following closed-form expressions exist,

$$e^{[w]} = I + \frac{[w]}{\|w\|} \sin \|w\| + \frac{[w]^2}{\|w\|^2} (1 - \cos \|w\|)$$

$$A = I + \frac{[w]}{\|w\|^2} (1 - \cos \|w\|) + \frac{[w]^2}{\|w\|^2} (\|w\| - \sin \|w\|) .$$

A.1.4 Adjoint Operators

The *adjoint mapping* $\mathrm{Ad} : SE(3) \times se(3) \to se(3)$ is defined as

$$\mathrm{Ad}_G g = GgG^{-1}$$

where $G \in SE(3)$ and $g \in se(3)$. The adjoint map is a coordinate transformation on $se(3)$.

The *Lie bracket* represented by the bilinear mapping $\text{ad} : se(3) \times se(3) \rightarrow se(3)$ is defined as

$$\text{ad}_g\, h = [g, h] = gh - hg$$

where $g, h \in se(3)$. The lie bracket generalizes the standard cross-product operator on $se(3)$. The Lie bracket satisfies the following two properties:

1. Skew-Symmetry: $[x, y] = -[y, x]$
2. Jacobi identity: $[x, [y, z]] + [z, [x, y]] + [y, [z, x]] = 0$

for every $x, y, z \in se(3)$.

When $se(3)$ is regarded as $\mathbb{R}^6$, the adjoint mappings when acting on elements of $\mathbb{R}^6$ have the 6×6 matrix representations

$$\text{Ad}_G\, g = \begin{pmatrix} R & 0 \\ [p]\, R & R \end{pmatrix} \begin{pmatrix} w_g \\ v_g \end{pmatrix}$$

$$\text{ad}_g\, h = \begin{pmatrix} [w_g] & 0 \\ [v_g] & [w_g] \end{pmatrix} \begin{pmatrix} w_h \\ v_h \end{pmatrix}$$

where $(R, p) \in SE(3)$ and both $g = (w_g, v_g)$ and $h = (w_h, v_h)$ are in $se(3)$. It is easily verified from these representations that $\text{Ad}_G^{-1} = \text{Ad}_{G^{-1}}$ and $\text{Ad}_G \text{Ad}_H = \text{Ad}_{GH}$ for any $G, H \in SE(3)$. It can also be verified that $\text{Ad}_G(\text{ad}_g h) = \text{ad}_{\text{Ad}_G g}(\text{Ad}_G h)$ for any $G \in SE(3)$ and $g, h \in se(3)$.

The *dual adjoint* operators, $\text{Ad}^* : SE(3) \times se(3) \rightarrow se(3)$ and $\text{ad}^* : se(3) \times se(3) \rightarrow se(3)$, have matrix representations defined as the usual transpose of the previous:

$$\begin{aligned} \text{Ad}_G^*\, g &= \begin{pmatrix} R & 0 \\ [p]\, R & R^T \end{pmatrix}^T \begin{pmatrix} w_g \\ v_g \end{pmatrix} \\ &= \begin{pmatrix} R^T & R^T\, [p]^T \\ 0 & R^T \end{pmatrix} \begin{pmatrix} w_g \\ v_g \end{pmatrix} \end{aligned}$$

and

$$\begin{aligned} \mathrm{ad}_g^* h &= \begin{pmatrix} [w_g] & 0 \\ [v_g] & [w_g] \end{pmatrix}^T \begin{pmatrix} w_h \\ g_h \end{pmatrix} \\ &= \begin{pmatrix} -[w_g] & -[v_g] \\ 0 & -[w_g] \end{pmatrix} \begin{pmatrix} w_h \\ g_h \end{pmatrix}. \end{aligned}$$

A.1.5 Generalized Velocities and Forces

Let $G(t) = (R(t), p(t)) \in SE(3)$ represent the trajectory of a body relative to some fixed reference frame. It can be shown that the two representations of the velocity of the body, $\dot{G}G^{-1}$ and $G^{-1}\dot{G}$, are elements of $se(3)$. The representation $\dot{G}G^{-1}$ is known as the *inertial velocity* and has the form

$$\dot{G}G^{-1} = \begin{pmatrix} \dot{R} & \dot{p} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} R^T & -R^T p \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \dot{R}R^T & -\dot{R}R^T p + \dot{p} \\ 0 & 0 \end{pmatrix} \in se(3).$$

The representation $G^{-1}\dot{G}$ is known as the *body-fixed velocity* and has the form

$$G^{-1}\dot{G} = \begin{pmatrix} R^T & -R^T p \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{R} & \dot{p} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} R^T\dot{R} & R^T\dot{p} \\ 0 & 0 \end{pmatrix} \in se(3)$$

(observe that both $\dot{R}R^T$ and $R^T\dot{R}$ are skew-symmetric).

If the trajectory undergoes a right transformation of the form $G(t) \to GT(t)$ where T is constant then the body-fixed velocity transforms as $(GT)^{-1}\dot{G}T = T^{-1}G^{-1}\dot{G}T = \mathrm{Ad}_{T^{-1}}(G^{-1}\dot{G})$. The inertial velocity representation is invariant under a right transformation.

Let (m, f) be a moment-force pair (a *generalized force*) at some frame on a rigid body and let $G = (R, p) \in SE(3)$ be a constant transformation from this frame to another frame on the

body. The moment-force pair is equivalently expressed in the new frame as

$$\begin{aligned}\begin{pmatrix} m' \\ f' \end{pmatrix} &= \begin{pmatrix} R^T m - R^T(p \times f) \\ R^T f \end{pmatrix} \\ &= \begin{pmatrix} R^T & R^T [p]^T \\ 0 & R^T \end{pmatrix} \begin{pmatrix} m \\ f \end{pmatrix} \\ &= Ad^*_G \begin{pmatrix} m \\ f \end{pmatrix}.\end{aligned}$$

Therefore, the dual adjoint map allows the transformation of generalized forces from one frame to another via the $SE(3)$ map between the two frames.

A.1.6 Derivatives of Adjoint Operators

Let $f \in SE(3)$ be a transformation of the form $f = Me^{Sq}$ (a fixed transformation $M \in SE(3)$ followed by a rotation $q \in \mathbb{R}$ about the axis $S \in se(3)$) and let $x, y \in se(3)$. The partial derivatives of the adjoint operators with respect to some variable p are:

Identity 1

$$\frac{\partial}{\partial p}\left(\mathrm{Ad}_{f^{-1}}\, x\right) = -\frac{\partial q}{\partial p}\mathrm{ad}_S \mathrm{Ad}_{f^{-1}}\, x + \mathrm{Ad}_{f^{-1}} \frac{\partial x}{\partial p} \tag{A.1}$$

Identity 2

$$\frac{\partial}{\partial p}\left(\mathrm{Ad}^*_{f^{-1}}\, x\right) = -\frac{\partial q}{\partial p}\mathrm{Ad}^*_{f^{-1}} \mathrm{ad}^*_S x + \mathrm{Ad}^*_{f^{-1}} \frac{\partial x}{\partial p} \tag{A.2}$$

Identity 3

$$\frac{\partial}{\partial p}\left(\mathrm{ad}_x y\right) = \mathrm{ad}_{\frac{\partial x}{\partial p}} y + \mathrm{ad}_x \frac{\partial y}{\partial p} \tag{A.3}$$

Identity 4

$$\frac{\partial}{\partial p}\left(\mathrm{ad}^*_x y\right) = \mathrm{ad}^*_{\frac{\partial x}{\partial p}} y + \mathrm{ad}^*_x \frac{\partial y}{\partial p} \tag{A.4}$$

Identities 1-4 provide a straightforward method of differentiating the kinematics and dynamics of robot mechanisms.

A.2 Physical Description of Constrained Robots

The following sets of parameters are used to describe robots with closed chains:

Link Parameters

- $f_{j,i}$ - Homogeneous transformation from frame $\{j\}$ to frame $\{i\}$.
- I_i - Generalized inertia tensor of link i expressed at the i^{th} frame
- S_i - Joint screw of link i
- q_i - Position coordinate(s) of link i
- v_i - Velocity coordinate(s) of link i, $v_i := \dot{q}_i$

Link Pointers

- P_i - Parent of link i
- C_i - Children of link i (excludes virtual links)
- T_i - Links in subtree of link i (excludes virtual links, includes the i^{th} link)
- T_i^v - Links in subtree of link i including virtual links
- R_i - Loop closure link reference of the i^{th} virtual link

Coordinate Sets

- q - Set of position coordinates of open chain
- q_v - Set of position coordinates of virtual links
- $\hat{q}$ - Full set of open chain and virtual position coordinates
- v - Set of velocity coordinates of open chain
- v_v - Set of velocity coordinates of virtual links
- $\hat{v}$ - Full set of open chain and virtual velocity coordinates

Numbering

- N - Number of links in open chain (number of physical bodies)
- n - Number of position/velocity coordinates in open chain
- N_v - Number of virtual links (number of kinematic loops)
- n_v - Number of position/velocity coordinates in virtual links

Physical interpretations of the parameters are given largely by example.

A.2.1 Removal of Kinematic Loops to Form an Open Tree-Structure

Consider the mechanism shown in Fig.A.1a. Most algorithms in robotics require that

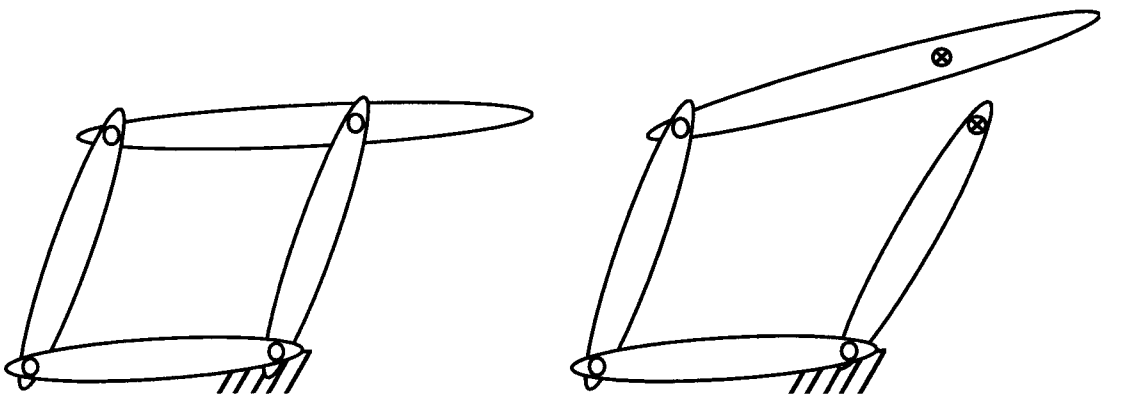


Figure A.1 (a) Closed chain mechanism and (b) equivalent open chain mechanism

the mechanism is free of kinematic loops; therefore the mechanism is first converted to an equivalent tree-structured robot by virtually removing a joint Fig.A.1b. When the dynamics of the robot are formulated, the removed joint is replaced with constraints that the two links are coincident at the point where the joint was cut.

A.2.2 Assignment of Link Frames

So that the links may be identified, they are indexed as links 1-4 in Fig.A.2. Coordinate frames {1}-{4} are attached to each respective link. There are additional frames at ground (labelled {0}) and on each half of the virtually cut joint. Frame {5} at the cut joint is allowed to move just as the actual joint would such that it is always coincident with the frame fixed to the other half. Joints 1-4 make up the set of joints in the *open chain* and joint 5 is the only *virtual link* (a joint without a physical body associated with it). To summarize, every body has a single frame and a single joint associated with it and there are additional frames associated with the virtually cut joints.

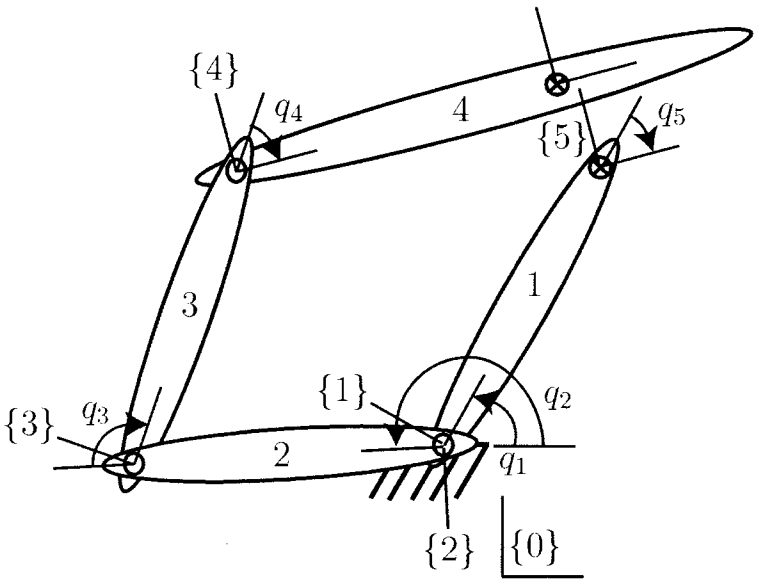


Figure A.2 Example closed chain mechanism with link/frame labels

The positions of bodies in a tree-structured mechanism can be parameterized by the relative motion of one body to the next/previous. In the convention used here, the displacement of each joint, q_i, in the transformation to a link relative to its *parent* (or, the neighboring link closest to the root of the tree), $f_{P_i,i}$, is used to parameterize the position.

To make the dynamics recursions more efficient, there are some imposed restrictions on the general form of $f_{P_i,i}$ which affect the placement of the coordinate frames. All frames must be attached in a way such that the transformation $f_{P_i,i} \in SE(3)$ can be written as a fixed transformation followed by a transformation associated with the movement of the joint. In the case of single degree-of-freedom joints, this implies they can be written as $f_{P_i,i} = M_i e^{S_i q_i}$ where $M_i \in SE(3)$ is a fixed transformation (to the *zero position* of the joint) and $e^{S_i q_i} \in SE(3)$ accounts for the joint displacement.

A.2.3 The Joint Screws

In the Lie algebraic formulation, the joint screws S_i are members of $se(3)$ and represent the axis of rotation/translation of single degree-of-freedom joints. For instance, the $\mathbb{R}^6$ repre-

sentations of S_i for revolute and prismatic joints in local coordinates are:

$$S = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad S = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Revolute, z−axis Prismatic, z−axis

Generalizations of the joint screws to multiple degree-of-freedom joints are discussed in Section A.2.7

A.2.4 The Generalized Inertia Tensor

The generalized inertia tensor at the center of mass of link i is of the form

$$\begin{pmatrix} J_i & 0 \\ 0 & m_i I_3 \end{pmatrix}$$

where m_i is the mass of link i and $J_i \in \mathbb{R}^{3\times 3}$ is the inertia matrix of link i at the center of mass. For efficient storage, the generalized inertia tensor of link i is transformed to the i^{th} frame as

$$I_i = Ad^*_{f^{-1}} \begin{pmatrix} J_i & 0 \\ 0 & m_i I_3 \end{pmatrix} Ad_{f^{-1}} = \begin{pmatrix} RJ_iR^T - m_i\,[p]^2 & m_i\,[p] \\ -m_i\,[p] & m_i I_3 \end{pmatrix}$$

where $f = (R, p)$ is a fixed transformation from frame $\{i\}$ to the frame in which J_i is expressed.

A.2.5 Link Pointers

The local topological information of each link is stored via the pointers: P_i (parent), C_i (children), T_i (subtree), and R_i (loop closure reference). These definitions should be largely

self-explanatory and are illustrated for the example in Table A.1 (∅ denotes the empty set).

Table A.1 Pointer parameters for closed chain example

i	P_i	C_i	T_i	T_i^v	R_i
1	0	∅	1	1,5	∅
2	0	3	2-4	2-4	∅
3	2	4	3,4	3,4	∅
4	3	∅	4	4	∅
5	1	∅	∅	5	4

A.2.6 Link Objects

If the robot was described in an object-oriented computing environment, the following information should be stored in each *Link* and *Virtual Link* object:

$$i^{th}\text{ Link}: \begin{cases} \text{Parameters}: \begin{cases} f_{P_i,\,i} \\ I_i \\ q_i, v_i, \dot{v}_i \\ S_i \end{cases} \\ \text{Pointers}: \begin{cases} P_i \\ C_i \\ T_i \end{cases} \end{cases}$$

$$i^{th}\text{ Virtual Link}: \begin{cases} \text{Parameters}: \begin{cases} f_{P_i,\,i} \\ f_{R_i,i} \\ q_i, v_i, \dot{v}_i \\ S_i \end{cases} \\ \text{Pointers}: \begin{cases} P_i \\ R_i \end{cases} \end{cases}$$

Notice that the *Virtual Link* object is very similar to the *Link* object. The obvious differences are that the *Virtual Link* object does not have a generalized inertia tensor since it is massless and does not have any children or subtree pointers since it is always located at the leaf of a branch.

The *Virtual Link* object must also store information pertaining to the other half of the cut joint. The index of the link on the other half of the cut joint is stored in the pointer R_i. The fixed transformation from the frame of link R_i to the location of the cut is recorded in the homogeneous transformation $f_{R_i,i}$. These two pieces of information uniquely describe the frame to which the i^{th} virtual joint must be aligned to maintain the constraint.

A.2.7 Multiple Degree-of-Freedom Joints

Multiple degree-of-freedom joints do not fit as nicely into the Lie algebraic formulation but can usually be incorporated through simple modifications. For multiple degree-of-freedom joints, S_i is derived such that $S_i v_i$ is equal to the $\mathbb{R}^6$ representation of the relative velocity across the joint, $f_{P_i,i}^{-1}\dot{f}_{P_i,i}$. For example, a cylindric joint usually consists of a rotation about the z-axis followed by a translation about the z-axis; therefore, the local link transformation is of the general form

$$f_{P_i,i}(q_i) = M_i \begin{pmatrix} \cos(q_{i1}) & -\sin(q_{i1}) & 0 & 0 \\ \sin(q_{i1}) & \cos(q_{i1}) & 0 & 0 \\ 0 & 0 & 1 & q_{i2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where $M_i \in SE(3)$ is a fixed transformation and the latter part accounts for the relative motion of the joint. Calculating the generalized velocity across the joint as $f_{P_i,i}^{-1}\dot{f}_{P_i,i}$ gives

$$
\begin{aligned}
f_{P_i,i}^{-1}\dot{f}_{P_i,i} &= \begin{pmatrix} 0 & -v_{i1} & 0 & 0 \\ v_{i1} & 0 & 0 & 0 \\ 0 & 0 & 0 & v_{i2} \\ 0 & 0 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} v_{i1} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} v_{i2}
\end{aligned}
$$

or in $\mathbb{R}^6$,

$$
\begin{aligned}
\left(f_{P_i,i}^{-1}\dot{f}_{P_i,i}\right)^{\vee} &= \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v_{i1} \\ v_{i2} \end{pmatrix} \\
&=: S_i v_i.
\end{aligned}
$$

Hence, S_i is fairly simple for a cylindric joint. For other types of joints, S_i may depend on the orientation of the joint and is thus a function of q_i.

The algorithms in Section A.3 have been modified to cope with multiple degree-of-freedom joints. The single degree-of-freedom versions of these algorithms can be recovered by setting $\dot{S}_i = 0$.

A.3 Dynamics Algorithms in Robotics

A dynamics algorithm known as the Newton-Euler algorithm [60, 61] applies to tree-structured open chain mechanisms and is given by:

Given: $V_0, \dot{V}_0, q, v, \dot{v}$

Initialize: For all i: $f_{P_i,i}(q_i)$

$S_i(q_i)$

$\dot{S}_i(q_i,\ v_i)$

Forward Recursion (open chain)

$$V_i = \mathrm{Ad}_{f_{P_i,i}^{-1}} V_{P_i} + S_i v_i$$
$$\dot{V}_i = \mathrm{Ad}_{f_{P_i,i}^{-1}} \dot{V}_{P_i} + S_i \dot{v}_i + \dot{S}_i v_i - \mathrm{ad}_{S_i v_i} V_i$$

■

Backward recursion (open chain)

$$F_i = \sum_{k \in C_i} \mathrm{Ad}^*_{f_{i,k}^{-1}} F_k + I_i \dot{V}_i - \mathrm{ad}^*_{V_i} I_i V_i$$
$$\tau_i = S_i^T F_i$$

■

The algorithm consists of two recursions. The first recursion starts at the root link and outwardly computes the body-fixed velocity, $V_i := f_{0,i}^{-1} \dot{f}_{0,i}$, and its time derivatives, $\dot{V}_i = \frac{d}{dt}(f_{0,i}^{-1} \dot{f}_{0,i})$, at the frame of each body. The second recursion starts at the tips of the branches and inwardly computes the generalized force at the joint of each body, F_i, and transforms it into a generalized actuation force of each joint, τ_i. So the Newton-Euler algorithm solves the inverse dynamics problem of tree-structured mechanisms, i.e., given a position, velocity, and acceleration, it solves what generalized actuation forces are required to produce such a motion.

The equations of motion of a tree-structured mechanism can be written in the general form

$$\dot{q} = v \tag{A.5a}$$

$$M(q)\dot{v} = \tau(t) + F(q,v) \tag{A.5b}$$

where $M(q)$ is the inertia-mass matrix, $F(q,v)$ accounts for the centrifugal and Coriolis terms, and $\tau(t)$ is the set of generalized actuation forces (no other external forces are assumed). Since the Newton-Euler algorithm solves for the external forces, $\tau(t)$, it can therefore be thought of as the function

$$\tau_{NE}(q,v,\dot{v}) := M(q)\dot{v} - F(q,v). \tag{A.6}$$

Viewing the Newton-Euler algorithm as a functional mapping offers some nice insights. For example, looking at (A.6) it is clear that the i^{th} column of M can be solved by letting

$$[M]_{col,i} = \tau_{NE}(q,v,e_i) - \tau_{NE}(q,v,0)$$

where e_i denotes the i^{th} unit vector. Supposing M is found in this way, the accelerations can be solved from (A.5) as

$$\begin{aligned} \dot{v} &= M^{-1}(\tau(t) + F(q,v)) \\ &= M^{-1}(\tau(t) - \tau_{NE}(q,v,0)) \end{aligned} \tag{A.7}$$

where $\tau(t)$ represents the set of generalized actuation forces [62].

A.3.1 Extending to Closed Chain Robots

The application to closed chain mechanisms is fairly straightforward. First, the kinematic loops of these mechanisms can always be virtually cut to create a tree-structured mechanism. The virtually cut joints are replaced with constraints such that the equations of motion take the form

$$\dot{q} = v \tag{A.8a}$$

$$M(q)\dot{v} = \tau(t) + F(q,v) - G^T(q)\lambda \tag{A.8b}$$

$$0 = g(q) \tag{A.8c}$$

where $g(q) = 0$ is an m-dimensional vector of cut joints constraints and $G^T(q)\lambda$ mimics the forces that would exist at the cut joints. Here $G := \partial g/\partial q$ and λ is a vector of Lagrange multipliers which can be solved as in equation (2.4).

Next, the constraint forces are interpreted as additional external forces and the accelerations can be solved similar to (A.7) as

$$\dot{v} = M^{-1}(\hat{\tau} - \tau_{NE}(q, v, 0)).$$

where $\hat{\tau} := \tau(t) - G^T(q)\lambda$.

A.3.2 Formulating the Constraints

One topic which has not yet been covered is the formulation of the kinematic loop constraints. Such constraints were previously assumed available as in equation (A.8). In this section, an algorithmic derivation of the kinematic loop constraints is introduced [15] which utilizes the previously introduced *virtual* links/joints.

The Constraint Jacobian

The condition that the relative velocity between the i^{th} virtual joint and its reference frame $\{i'\}$ should be zero may be expressed as

$$f_{0,i}^{-1}\dot{f}_{0,i} - f_{0,i'}^{-1}\dot{f}_{0,i'} = 0.$$

This condition is equivalently expressed as

$$J_i\hat{v} - J_{i'}\hat{v} = 0$$

or

$$(J_i - J_{i'})\hat{v} = 0$$

where $J_i \in R^{6\times n+n_v}$ is the body-fixed jacobian at the i^{th} frame. The body-fixed jacobians at each frame can be derived recursively as:

Forward Recursion (all joints)

$$J_i = \mathrm{Ad}_{f_{P_i,i}^{-1}} J_{P_i} + \begin{bmatrix} & | & \\ 0 \cdots 0 & S_i & 0 \cdots 0 \\ & | & \\ & i & \end{bmatrix}$$

■

Suppose the indices of the N_L cut joints are denoted as c_1, c_2,...,c_{N_L}, then the constraints at all of the virtual joints can be combined into one as

$$\begin{pmatrix} J_{c_1} - J_{c_1'} \\ \vdots \\ J_{c_{N_L}} - J_{c_{N_L}'} \end{pmatrix} \hat{v} = 0$$

where $J_{c_i'} = \mathrm{Ad}_{f_{R_{c_i},c_i}^{-1}} J_{R_{c_i}}$ (transforms the link reference jacobian, $J_{R_{c_i}}$, to the cut joint location). Finally, let $\hat{G}$ be the above matrix with a full set of linearly dependent rows removed. The matrix $\hat{G}(q, q_v) \in \mathbb{R}^{m+n_v\times n+n_v}$ represents the jacobian of a complete set of independent constraints.

From this development it is clear that $\hat{G}$ may be expressed as

$$\hat{G} = A \begin{pmatrix} J_1 \\ \vdots \\ J_{N+N_v} \end{pmatrix} \tag{A.9}$$

where $J_1 \ldots J_{N+N_v}$ are the jacobians at each joint and $A \in \mathbb{R}^{m+n_v\times 6(N+N_v)}$ is a constant matrix which accounts for coordinate transformations (to transform the jacobians at the reference links $R_i^{'s}$ to the cut joints) and the removal of dependent rows. Hence, the use of virtual links has led to a constraint jacobian which is linear in the body-fixed jacobians.

The Usual Constraint Jacobian

The only problem with this formulation is that $\hat{G}(q, q_v) \in \mathbb{R}^{m+n_v \times n+n_v}$ is not directly equivalent to the usual constraint jacobian $G(q) \in \mathbb{R}^{m \times n}$. The usual constraint jacobian can be derived by noting that the virtual coordinates, $q_v \in \mathbb{R}^{n_v}$, are implicitly defined by the open chain coordinates, $q \in \mathbb{R}^n$. This derivation follows.

There are n_v constraints associated with the additional redundant coordinates at the virtual links. Suppose these constraints are given by

$$g_v(q, q_v) = 0.$$

Let $g \in \mathbb{R}^m$ denote the remaining constraints associated with the closed loops and let $\hat{g}$ denote the full set of constraints,

$$\hat{g}(q, q_v) = \begin{pmatrix} g(q, q_v) \\ g_v(q, q_v) \end{pmatrix}.$$

and hence it follows that $\hat{G}$ is defined as $\hat{G} := \partial\hat{g}/\partial(q, q_v)$ (of course, a swapping of rows and columns might be necessary to account for the ordering of constraints and coordinates). The jacobian $\hat{G}$ takes the partitioned form

$$\frac{\partial \hat{g}}{\partial(q, q_v)} = \left(\begin{array}{c|c} \frac{\partial g}{\partial q} & \frac{\partial g}{\partial q_v} \\ \hline \frac{\partial g_v}{\partial q} & \frac{\partial g_v}{\partial q_v} \end{array}\right).$$

In view of the partitioned matrix the fact that $\hat{G}\hat{v} = 0$ implies

$$\left(\frac{\partial g}{\partial q}\right) v = -\left(\frac{\partial g}{\partial q_v}\right) v_v \tag{A.10}$$

and

$$\left(\frac{\partial g_v}{\partial q}\right) v = -\left(\frac{\partial g_v}{\partial q_v}\right) v_v. \tag{A.11}$$

Since the orientations at the virtual joints are uniquely determined by the positions of the

bodies then $|\partial g_v/\partial q_v| \neq 0$ and hence (A.11) can be solved for v_v as

$$v_v = -\left(\frac{\partial g_v}{\partial q_v}\right)^{-1}\left(\frac{\partial g_v}{\partial q}\right)v.$$

Substituting this result into (A.10) leads to

$$\left[\left(\frac{\partial g}{\partial q}\right) - \left(\frac{\partial g}{\partial q_v}\right)\left(\frac{\partial g_v}{\partial q_v}\right)^{-1}\left(\frac{\partial g_v}{\partial q}\right)\right]v = 0 \tag{A.12}$$

which makes this matrix an appropriate definition of the usual constraint jacobian G. While this derivation may seem unnecessarily complicated, it does have the advantage of being independent of the type of cut joints. In addition, the inverse of $\partial g_v/\partial q_v$ can be formulated with linear complexity and in most cases, $\partial g/\partial q_v = 0$.

A.3.3 The Null Space Formulation of the Dynamics

The null space formulation of Section 2.4 mainly relies on the derivation of two terms: the basis for the null space of the constraints, K, and one of its derivatives, $\dot{K}y_v$. In this section, a derivation of these two terms which is closely tied to the derivation of the constraints of the previous section is presented.

Let $\hat{q} := (q, q_v)$ denote the full set of position coordinates and suppose an independent set of position coordinates is defined as

$$y_q = R_q\hat{q}$$

where $R_q \in \mathbb{R}^{p\times n+n_v}$ is chosen such that $(R_q^T\hat{G}^T)^T$ is nonsingular. From Chapter 2 the independent position coordinates locally parameterize the full set $\hat{q}$ and the jacobian of this relationship is given by

$$\frac{\partial\hat{q}}{\partial y_q} = \begin{pmatrix} R_q \\ \hat{G} \end{pmatrix}^{-1}\begin{pmatrix} I_p \\ 0 \end{pmatrix}.$$

Notice that the rows of $\partial\hat{q}/\partial y_q$ corresponding to the open chain coordinates q define the basis for the null space $K := \partial q/\partial y_q$. Suppose these rows are extracted through left multiplication

of a matrix E such that

$$K = E \begin{pmatrix} R_q \\ \hat{G} \end{pmatrix}^{-1} \begin{pmatrix} I_p \\ 0 \end{pmatrix}. \tag{A.13}$$

Notice that this derivation of K is somewhat convenient in the sense that it does not require the conversion of $\hat{G}$ to G. Also, recalling that $\hat{G} = AJ$ (A.9) where J denotes the set of body-fixed jacobians it is clear that the various derivatives of K required in Chapter 4 can be conveniently derived using derivatives of J.

From Chapter 2, it follows that $\dot{K}y_v$ is given by

$$\dot{K}y_v = E \begin{pmatrix} R_q \\ \hat{G} \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \dot{\hat{G}}\hat{v} \end{pmatrix}$$

which leaves $\dot{\hat{G}}\hat{v}$ as the only remaining unknown. By differentiating $\hat{G} = AJ$ one has that the vector $\dot{\hat{G}}\hat{v}$ is equal to $A\dot{J}\hat{v}$ or,

$$\dot{\hat{G}}\hat{v} = A \begin{pmatrix} \dot{J}_1\hat{v} \\ \vdots \\ \dot{J}_{N+N_v}\hat{v} \end{pmatrix}.$$

Each $\dot{J}_i\hat{v}$ can be derived similar to the calculation of $\dot{V}_i$ in the Newton-Euler algorithm since by definition, each $\dot{V}_i$ is equal to $J_i\dot{\hat{v}} + \dot{J}_i\hat{v}$. Hence, by setting the accelerations to zero each $\dot{J}_i\hat{v}$ can be found recursively as:

Forward Recursion (all joints)

$$V_i = \mathrm{Ad}_{f_{P_i,i}^{-1}} V_{P_i} + S_i\hat{v}_i$$
$$\left(\dot{J}_i\hat{v}\right) = \mathrm{Ad}_{f_{P_i,i}^{-1}} \left(\dot{J}_{P_i}\hat{v}\right) + \dot{S}_i\hat{v}_i - \mathrm{ad}_{S_i\hat{v}_i} V_i$$

■

Some computational effort can be saved by reusing these terms if the Newton-Euler algorithm is subsequently needed.

Once again, an equivalent to the usual $\dot{G}v$ can be easily derived. In fact, the use of virtual coordinates can be eliminated altogether if global coordinate transformations are used to solve

for the local transformations, $f_{P_i,i}$, at each virtual joint.

A.3.4 Other Algorithms in Robot Dynamics

Depending on the topology of the system and the total number of bodies, algorithms other than the Newton-Euler should be used for increased efficiency. For example, an alternative derivation of the inertia-mass matrix is given by the Composite-Rigid-Body algorithm [32]:

Given: q

Initialize: $M = 0$

For all i: $I_i^c = I_i$

$f_{P_i,i}(q_i)$

$S_i(q_i)$

Backward Recursion (open chain)

$f_{i,i}^c = I_i^c S_i$

for $j \in T_i$

$M_{ij} = S_i^T f_{j,i}^c$

$M_{ji} = M_{ij}^T$

end

if $P_i \neq 0$

$I_{P_i}^c = I_{P_i}^c + \mathrm{Ad}^*_{f_{P_i,i}^{-1}} I_i^c \mathrm{Ad}_{f_{P_i,i}^{-1}}$

for $j \in T_i$

$f_{j,P_i}^c = \mathrm{Ad}^*_{f_{P_i,i}^{-1}} f_{j,i}^c$

end

end

■

Also, the accelerations can be solved without inverting the mass-inertia matrix using the Articulated-Body Algorithm [32] which has linear complexity (traditional inversion has cubic complexity):

Given: V_0, $\dot{V}_0$, τ (external forces), q, v

Initialize: For all i: $f_{P_i,i}(q_i)$

$S_i(q_i)$

$\dot{S}_i(q_i,\, v_i)$

Forward recursion (open chain)

$$V_i = \mathrm{Ad}_{f_{P_i,i}^{-1}} V_{P_i} + S_i v_i$$
$$a_i = \dot{S}_i v_i - \mathrm{ad}_{S_i v_i} V_i$$

■

Backward recursion (open chain)

$$\hat{I}_i = I_i + \sum_{j \,\epsilon\, C_i} \mathrm{Ad}^*_{f_{P_j,j}^{-1}} \left(\hat{I}_j - \hat{I}_j S_j (S_j^T \hat{I}_j S_j)^{-1} S_j^T \hat{I}_j\right) \mathrm{Ad}_{f_{P_j,j}^{-1}}$$
$$b_i = -\mathrm{ad}^*_{V_i} I_i V_i + \sum_{j \,\epsilon\, C_i} \mathrm{Ad}^*_{f_{P_j,j}^{-1}} \left(z_j + \hat{I}_j S_j (S_j^T \hat{I}_j S_j)^{-1} \left(\tau_j - S_j^T z_j\right)\right)$$
$$z_i = \hat{I}_i a_i + b_i$$

■

Forward recursion (open chain)

$$\dot{v}_i = (S_i^T \hat{I}_i S_i)^{-1} \left(\tau_i - S_i^T \left[\hat{I}_i \mathrm{Ad}_{f_{P_i,i}^{-1}} \dot{V}_{P_i} + z_i\right]\right)$$
$$\dot{V}_i = \mathrm{Ad}_{f_{P_i,i}^{-1}} \dot{V}_{P_i} + S_i \dot{v}_i + a_i$$

■

For simple robots, the number of computations involved in the Articulated-Body Algorithm is considerably higher than approaches which invert the mass-inertia matrix. The linear complexity of the Articulated-Body Algorithm begins to win out when the number of bodies in the system exceeds approximately 11 [26].

APPENDIX B. DERIVATION OF LINEARIZED MODELS

In this appendix, derivations of the terms $\hat{A}_{21}$ and $\hat{A}_{22}$ from Section 4.2 are given. The $\hat{A}_{21}$ term is defined implicitly where the system accelerations are assumed to be known. The derivation of $\hat{A}_{22}$ is explicit and more straightforward.

B.1 Derivation of $\hat{\mathbf{A}}_{21}$

Consider the functions

$$\bar{\tau}(y_q, y_v, \dot{y}_v, t) := (K^T M K)\dot{y}_v + K^T(M\dot{K}y_v - F(q,v) - \tau(t)) \tag{B.1}$$

and

$$\bar{f}(y,t) := (K^T M K)^{-1} K^T (F(q,v) - M\dot{K}y_v + \tau(t)). \tag{B.2}$$

The function $\bar{f}$ is an expression for the independent accelerations in (4.8b) where $\tau(t)$ is a set of external forces. Substituting $\bar{f}$ for $\dot{y}_v$ in $\bar{\tau}$ it can be shown that

$$\bar{\tau}(y_q, y_v, \bar{f}(y,t), t) = 0. \tag{B.3}$$

Differentiating (B.3) with respect to y_q gives

$$\begin{aligned} \frac{\partial}{\partial y_q}\left(\bar{\tau}(y_q, y_v, \bar{f}(y,t), t)\right) &= [D_1\bar{\tau}] + [D_3\bar{\tau}]\,\hat{A}_{21} \\ &= 0 \end{aligned}$$

where $\hat{A}_{21}$ has been substituted for $\partial \bar{f}/\partial y_q$ (this is the definition of $\hat{A}_{21}$). Solving for $\hat{A}_{21}$ gives

$$\hat{A}_{21} = -(K^T M K)^{-1} [D_1 \bar{\tau}]$$

where the substitution $[D_3 \bar{\tau}] = K^T M K$ which is clear from (B.1) has been used.

One can verify that

$$\begin{aligned} \bar{\tau}(y_q, y_v, \dot{y}_v, t) &= K^T (M\dot{v} - F(q,v) - \tau(t)) \\ &= K^T (\tau_{NE}(q, v, \dot{v}) - \tau(t)). \end{aligned}$$

Therefore, the derivative $[D_1 \bar{\tau}]$ is equal to

$$\begin{aligned} D_1 \bar{\tau}(y_q, y_v, \dot{y}_v, t) &= \frac{\partial}{\partial y_q} \left(K^T \left(\tau_{NE}(\Psi_q, \dot{\Psi}_q, \ddot{\Psi}_q) - \tau(t) \right) \right) \\ &= \begin{bmatrix} \frac{\partial K^T}{\partial y_{q_1}} (\tau_{NE} - \tau(t)) & \cdots & \frac{\partial K^T}{\partial y_{q_p}} (\tau_{NE} - \tau(t)) \end{bmatrix} \\ &\quad + K^T \frac{\partial}{\partial y_q} \left(\tau_{NE}(\Psi_q, \dot{\Psi}_q, \ddot{\Psi}_q) - \tau(t) \right) \\ &= \begin{bmatrix} \frac{\partial K^T}{\partial y_{q_1}} (\tau_{NE} - \tau(t)) & \cdots & \frac{\partial K^T}{\partial y_{q_p}} (\tau_{NE} - \tau(t)) \end{bmatrix} \\ &\quad + K^T \left(\frac{\partial \tau_{NE}}{\partial q} K + \frac{\partial \tau_{NE}}{\partial v} \dot{K} + M\ddot{K} \right) \end{aligned}$$

where the substitutions $K = \partial \Psi_q / \partial y_q$, $\dot{K} = \partial \dot{\Psi}_q / \partial y_q$, $\ddot{K} = \partial \ddot{\Psi}_q / \partial y_q$, and $M = \partial \tau_{NE} / \partial \dot{v}$ have been used. This is the form of the derivative used in Section 4.2 (note the slight difference when external forces are present).

B.2 Derivation of $\hat{A}_{22}$

Just as in the derivation of A_{21}, the function

$$\bar{f}(y,t) := (K^T M K)^{-1} K^T \left(\tau(t) - \tau_{NE}(q, v, \dot{K} y_v) \right) \tag{B.4}$$

is defined which is an expression for the independent accelerations in (4.10b) and whose partial derivative with respect to y_v is the definition of $\hat{A}_{22}$. Therefore, differentiating (B.4) gives

$$\begin{aligned}\hat{A}_{22} &= \frac{\partial}{\partial y_v}\left((K^T M K)^{-1} K^T \left(\tau(t) - \tau_{NE}(\Psi_q, \dot{\Psi}_q, \dot{K} y_v)\right)\right) \\ &= -(K^T M K)^{-1}\left(\frac{\partial \tau_{NE}}{\partial v} K + M \frac{\partial \dot{K}(y) y_v}{\partial y_v}\right) \end{aligned} \tag{B.5}$$

where the substitutions $K = \partial \dot{\Psi}_q / \partial y_v$ and $M = \partial \tau_{NE} / \partial \dot{v}$ have been used.

The last term in the previous equation can be simplified as

$$\begin{aligned}\frac{\partial(\dot{K}(y) y_v)}{\partial y_v} &= \dot{K} + \left[\frac{\partial \dot{K}}{\partial y_{v_1}} y_v \cdots \frac{\partial \dot{K}}{\partial y_{v_\mathrm{p}}} y_v\right] \\ &= \dot{K} + \left[\frac{\partial}{\partial y_{v_1}}\left(\sum_i \frac{\partial K}{\partial y_{q_i}} y_{v_i}\right) y_v \cdots \frac{\partial}{\partial y_{v_\mathrm{p}}}\left(\sum_i \frac{\partial K}{\partial y_{q_i}} y_{v_i}\right) y_v\right] \\ &= \dot{K} + \left[\frac{\partial K}{\partial y_{q_1}} y_v \cdots \frac{\partial K}{\partial y_{q_\mathrm{p}}} y_v\right] \end{aligned}$$

where observe that

$$\frac{\partial K}{\partial y_{q_i}} y_v = \frac{\partial^2 \Psi_q}{\partial y_{q_i} \partial y_q} y_v = \frac{\partial}{\partial y_q}\left(\frac{\partial \Psi_q}{\partial y_{q_i}}\right) y_v = \frac{\partial}{\partial y_q}\left(K_{col,i}\right) y_v = \dot{K}_{col,i}$$

and hence the last term in (B.5) simplifies to

$$\frac{\partial(\dot{K}(y) y_v)}{\partial y_v} = 2\dot{K}.$$

BIBLIOGRAPHY

[1] R. A. Wehage and E. J. Haug. Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems. *Journal of Mechanical Design*, 104:247–255, 1982.

[2] F. A. Potra and W. C. Rheinboldt. On the numerical solution of the Euler-Lagrange equations. *Mechanics of Structures and Machines*, 19:1–18, 1991.

[3] F. A. Potra and J. Yen. Implicit integration for Euler-Lagrange equations via tangent space parameterization. *Mechanics of Structures and Machines*, 19:77–98, 1991.

[4] E. J. Haug and J. Yen. Implicit numerical integration of constrained equations of motion via generalized coordinate partitioning. In E. Haug and R. Deyo, editors, *NATO Advanced Research Workshop on Real-Time Integration Methods for Mechanical System Simulation.* Springer-Verlag, Heidelberg, 1990.

[5] S. S. Kim and M. J. Vanderploeg. QR decomposition for state space representation of constrained mechanical dynamic systems. *Journal of Mechanisms, Transmissions, and Automation in Design*, 108:183–188, 1986.

[6] J. Yen. Constrained equations of motion in multibody dynamics as ODEs on manifolds. *SIAM Journal on Numerical Analysis*, 30:553–568, 1993.

[7] M. G. Ianeu. *On the Differentiable Null Space Method for Differential Algebraic Equations.* PhD thesis, University of Iowa, 2000.

[8] R. Serban and E. J. Haug. Globally independent coordinates for real-time vehicle simulation. *Journal of Mechanical Design*, 122:575–582, 2000.

[9] J. Yen and L. R. Petzold. An efficient newton-type iteration for the numerical solution of highly oscillatory constrained multibody dynamic systems. *SIAM Journal on Scientific Computing*, 19:1513–1534, 1998.

[10] F. A. Potra. Implementation of linear multistep methods for solving constrained equations of motion. *SIAM Journal on Numerical Analysis*, 30:774–789, 1993.

[11] E. Eich, C. Fhrer, and J. Yen. On the error control for multistep methods applied to odes with invariants and daes in multibody dynamics. *Mechanics of Structures and Machines*, 23:159–179, 1995.

[12] D. Negrut, E. J. Haug, and H. C. German. An implicit rungekutta method for integration of differential algebraic equations of multibody dynamics. *Multibody System Dynamics*, 9:121–142, 2003.

[13] J. Yen. Constrained equations of motion in multibody dynamics as ODEs on manifolds. *SIAM Journal on Numerical Analysis*, 30:553–568, 1993.

[14] Y. Nakamura and M. Ghodoussi. Dynamics computation of closed-link robot mechanisms with nonredundant and redundant actuators. *IEEE Transactions on Robotics and Automation*, 5:294–302, 1989.

[15] Y. Nakamura and K. Yamane. Dynamics computation of structure-varying kinematic chains and its application to human figures. *IEEE Transactions on Robotics and Automation*, 16:124–134, 2000.

[16] H. S. Chin. *Stabilization Methods for Simulations of Constrained Multibody Dynamics.* PhD thesis, The University of British Columbia, 1995.

[17] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Diffential-Algebraic Equations.* SIAM, 1998.

[18] J. Baumgarte. Stabilization of constraints and integrals of motion. *Conceptual Methods of Applied Mechanics and Engineering*, 1:1–66, 1972.

[19] U. M. Ascher and H. Chin. Stabilization of DAEs and invariant manifolds. *Numerische Mathematik*, 67:131–149, 1994.

[20] U. M. Ascher, H. Chin, L. R. Petzold, and S. Reich. Stabilization of constrained mechanical systems with DAEs and invariant manifolds. *Mechanics of Structures and Machines*, 23:135–158, 1995.

[21] E. Eich. Convergence results for a coordinate projection method applied to mechanical systems with algebraic constraints. *SIAM Journal on Numerical Analysis*, 30:1467–1482, 1993.

[22] U. Ascher and P. Lin. Sequential regularization methods for higher index DAEs with constraint singularities: The linear index-2 case. *SIAM Journal on Numerical Analysis*, 33:1921–1940, 1996.

[23] U. Ascher and P. Lin. Sequential regularization methods for nonlinear higher index differential-algebraic equations. *SIAM Journal on Scientific Computing*, 18:160–181, 1997.

[24] L. R. Petzold, Y. Ren, and T. Maly. Regularization of higher-index differential-algebraic equations with rank-deficient constraints. *SIAM Journal on Scientific Computing*, 18:753–774, 1997.

[25] C. B. Macdonald. The predicted sequential regularization method for differential-algebraic equations. B.S. Honours thesis, Arcadia University, 2001.

[26] T. Andrzejewski, H. G. Bock, E. Eich, and R. Von Schwerin. Recent advances in the numerical integration of multibody systems. In W. Schiehlen, editor, *Advanced Multibody System Dynamics - Simulation and Software Tools*. Kluwer Academic Publishers, Dordrecht, 1993.

[27] D. Baraff. Linear-time dynamics using Lagrange multipliers. In *Computer Graphics Proceedings*, pages 137–146, 1996.

[28] R. Featherstone and D. Orin. Robot dynamics: Equations and algorithms. In *IEEE International Conference on Robotics and Automation*, pages 826–834, 2000.

[29] Wood G. D. Simulating mechanical systems in Simulink with SimMechanics. www.mathworks.com, 2002.

[30] D. Bae and E. Haug. A recursive formulation for constrained mechanical system dynamics: Part 2. closed loop systems. *Mechanics of Structures and Machines*, 15:481–506, 1987.

[31] R. Featherstone. A divide-and-conquer atriculated-body algorithm for parallel $O(\log(n))$ calculation of rigid-body dynamics. part 2: Trees, loops and accuracy. *The International Journal of Robotics Research*, 18:876–892, 1999.

[32] R. Featherstone. *Robot Dynamics Algorithms.* Kluwer Academic Publishers, 1987.

[33] G. Rodriguez, A. Jain, and K. Kreutz-Delgado. A spatial operator algebra for manipulator modeling and control. *The International Journal of Robotics Research*, 10:371–381, 1991.

[34] F. C. Park and J. E. Bobrow. A recursive algorithm for robot dynamics using Lie groups. In *IEEE International Conference on Robotics and Automation*, pages 1535–1540, 1994.

[35] F. C. Park, J. E. Bobrow, and S. R. Ploen. A Lie group formulation of robot dynamics. *The International Journal of Robotics Research*, 14:609–618, 1995.

[36] C-Y. E. Wang, W. K. Timoszyk, and J. E. Bobrow. Weightlifting motion planning for a Puma 762 robot. In *IEEE International Conference on Robotics and Automation*, pages 480–485, 1999.

[37] B. J. Martin and J. E. Bobrow. Minimum-effort motions for open-chain manipulators with task-dependent end-effector constraints. *The International Journal of Robotics Research*, 18:213–224, 1999.

[38] G. A. Sohl and J. E. Bobrow. A recursive multibody dynamics and sensitivity algorithm for branched kinematic chains. *ASME Journal of Dynamic Systems, Measurement, and Control*, 123:391–399, 2001.

[39] R. Serban and E. J. Haug. Kinematic and kinetic derivatives in multibody system analysis. *Mechanics of Structures and Machines*, 26:145–173, 1998.

[40] J. Garca de Jaln and E. Bayo. *Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge.* Springer-Verlag, 1993.

[41] E. J. Haug. *Computer Aided Kinematics and Dynamics of Mechanical Systems.* Allyn and Bacon, 1989.

[42] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *The Numerical Solution of Initial Value Problems in Ordinary Differential-Algebraic Equations.* North Holland Publishing Co., 1989.

[43] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems.* Springer-Verlag, 1991.

[44] M. Spivak. *Calculus on Manifolds.* W. A. Benjamin, Inc., 1965.

[45] V. Guillemin and A. Pollack. *Differential Topology.* Prentice-Hall, 1974.

[46] M. Rosenlicht. *Introduction to Analysis.* Dover Publications, Inc., 1986.

[47] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, 1983.

[48] R. E. Ellis and S. L. Ricker. Two numerical issues in simulating constrained robot dynamics. *IEEE Transactions on Systems, Man, and Cybernetics*, 24:19–27, 1994.

[49] S. L. Campbell and Jr. C. P. Meyer. *Generalized Inverses of Linear Transformations.* Dover Publications, 1991.

[50] F. Aghili and Piedboeuf J.-C. Simulation of motion of constrained multibody systems based on projection operator. *Multibody System Dynamics*, 10:3–16, 2003.

[51] A. A. Maciejewski. Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Computer Graphics and Applications*, 10:6371, 1990.

[52] L. R. Petzold, Y. Ren, and T. Maly. Regularization of higher-index differential-algebraic equations with rank-deficient constraints. *SIAM Journal on Scientific Computing*, 18:753–774, 1997.

[53] B. Simeon, C. Fuhrer, and P. Rentrop. The Drazin inverse in multibody system dynamics. *Numerishe Mathematik*, 64:521–539, 1993.

[54] S. R. Ploen. *Geometric Algorithms for the Dynamics and Control of Mulitibody Systems.* PhD thesis, University of California, Irvine, 1997.

[55] K. Yamane. *Realtime Interactive Dynamics Computation of Structure-Varying Kinematic Chains and Its Application to Motion Generation of Human Figures.* PhD thesis, University of Tokyo, 2001.

[56] C. Fuhrer and B. J. Leimkuhler. Numerical solution of differential-algebraic equations for constrained mechanical motion. *Numerische Mathematik*, 59:55–69, 1991.

[57] R. M. Murray, Z. Li, and S. Sastry. *A Mathematical Introduction to Robotics.* CRC Press, 1994.

[58] J. M. Selig. *Geometrical Methods in Robotics.* Springer-Verlag, 1996.

[59] J. Kim, J. Baek, and F. C. Park. Newton-type algorithms for robot motion optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1842–1847, 1999.

[60] D. E. Orin, R. B. McGhee, M. Vukobratovic, and G. Hartoch. Kinematic and kinetic analysis of open-chain linkages utilizing Newton-Euler methods. *Mathematical Biosciences*, 43:107–130, 1979.

[61] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul. On-line computational scheme for mechanical manipulators. *ASME Journal of Dynamic Systems, Measurement, and Control*, 102:69–76, 1980.

[62] M. W. Walker and D. E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *ASME Journal of Dynamic Systems, Measurement, and Control*, 104:205–211, 1982.