JOINT-SPACE RECIPES FOR MANIPULATOR ROBOTS PERFORMING COMPLIANT MOTION TASKS: TRAJECTORY-OPTIMIZATION, INTERPOLATION, AND CONTROL

By

Yueshi Shen

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
THE AUSTRALIAN NATIONAL UNIVERSITY
CANBERRA, AUSTRALIA
FEBRUARY 2006

THE AUSTRALIAN NATIONAL UNIVERSITY
DEPARTMENT OF
INFORMATION ENGINEERING, RSISE

The undersigned hereby certify that they have read and recommend to the Research School of Information Sciences and Engineering for acceptance a thesis entitled "**Joint-Space Recipes for Manipulator Robots Performing Compliant Motion Tasks: Trajectory-Optimization, Interpolation, and Control**" by **Yueshi Shen** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Dated: February 2006

Research Supervisor: _____
Dr. Knut Hüper

Examing Committee: _____
Prof. Fátima Silva Leite

_____
Prof. Martin Buss

THE AUSTRALIAN NATIONAL UNIVERSITY

Date: **February 2006**

Author: **Yueshi Shen**

Title: **Joint-Space Recipes for Manipulator Robots Performing Compliant Motion Tasks: Trajectory-Optimization, Interpolation, and Control**

Department: **Information Engineering, RSISE**

Degree: **Ph.D.**

Signature of Author

To My Parents and Jane.

Table of Contents

List of Figures

List of Symbols

$\hat{\mathbf{a}}$	6-dim rigid body acceleration
C	Constraint function
$\mathbf{C}$	Coriolis and gravitational force of robot manipulator
$\mathcal{E}$	Configuration space, or embedding Euclidean space
$\hat{\mathbf{f}}$	6-dim wrench (generalized force)
$\mathcal{F}^6$	Operational space, force
$\mathcal{F}^q$	Joint space, force
h	Time step size
$\mathbf{H}$	Joint-space inertia matrix of robot manipulator
$\mathbf{H}_o$	Operational-space inertia matrix
i	Index of constraint functions
$\mathbf{I}$	Identity matrix
j	Index of robot joints, or of Cartesian coordinate's dimensions
J	Cost function
$\mathbf{J}$	Jacobian matrix of robot manipulator
k	Index of discrete time
l	Number of inequality constraints
L	Lagrangian function
m	Number of equality constraints
M	Manifold
$\mathcal{M}^6$	Operational space, motion

$\mathcal{M}^q$	Joint space, motion
n	Number of time partitions
$\mathcal{N}, \mathcal{N}'$	Subspaces of $\mathcal{F}^6$
$\mathcal{N}_j, \mathcal{N}'_j$	Subspaces of $\mathcal{F}^q$
$\mathbf{O}$	Zero matrix
p	Number of Euclidean space's dimensions
$\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}}$	Element of $\mathcal{E}$, with its first and second derivative; in particular, rigid body's linear position, velocity, and acceleration
$\hat{\mathbf{p}}$	Configuration of a rigid body in 3-dim Euclidean space
$\mathbf{P}_h$	Discretization operator with time step size h
q	Number of robot manipulator's joints
$\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$	Joint position, velocity, and acceleration of a robot manipulator
r	Rolling map
$\mathbf{r}$	Projected rotation matrix
$\mathbf{R}$	Rotation matrix of a rigid body
$T_{\mathbf{p}}M$	Tangent space of M at $\mathbf{p} \in M$
$T^{\text{aff}}_{\mathbf{p}}M$	Affine tangent space of M at $\mathbf{p} \in M$
t_0, t_n	Initial and final time
$\mathcal{T}, \mathcal{T}'$	Subspaces of $\mathcal{M}^6$
$\mathcal{T}_j, \mathcal{T}'_j$	Subspaces of $\mathcal{M}^q$
$\hat{\mathbf{v}}$	6-dim rigid body velocity
V	Affine tangent space, e.g., $T^{\text{aff}}_{\mathbf{p}}M$
y	1-dim resulting curve of some variational problem
$\mathbf{y}$	p-dim resulting curve of some variational problem
α	Rolling curve
α_{dev}	Development of α
β	Interpolation curve in V
γ	Interpolation curve on M
η	Projected initial or final velocity

κ	Forward kinematic map
μ	Lagrange multiplier
ξ	Initial or final velocity
τ	Joint torque of a robot manipulator
ϕ	Local diffeomorphism
Φ	Dynamic projector
Ψ	Kinematic projector
ω	Angular velocity of a rigid body
Ω	Element of $\mathfrak{so}_3$

Statement of Originality

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement is made in the next.

Most of the technical discussions in this thesis are based on the following publications:

- Y. Shen, K. Hüper, and F. Silva Leite, *Smooth interpolation of orientation by rolling and wrapping for robot motion planning*, Accepted by the 2006 IEEE International Conference on Robotics and Automation (ICRA), Orlando, USA, May 2006.

- Y. Shen and K. Hüper, *A joint space formulation for compliant motion control of robot manipulators*, Proceedings of The 2005 IEEE International Conference on Mechatronics and Automation (ICMA), Niagara Falls, Canada, Jul 2005. *(the Best Student Conference Paper Award of the IEEE ICMA 2005.)*

- Y. Shen and K. Hüper, *Optimal trajectory planning of manipulators subject to motion constraints*, Proceedings of The 12th International Conference on Advanced Robotics (ICAR), Seattle, USA, Jul 2005.

- Y. Shen and K. Hüper, *Optimal joint trajectory planning for manipulator robot performing constrained motion tasks*, Proceedings of The 2004 Australasian

Conference on Robotics and Automation (ACRA), Canberra, Australia, Dec 2004.

- Y. Shen and R. Featherstone, *The effect of ill-conditioned inertial matrix on controlling robot manipulator*, Proceedings of The 2003 Australasian Conference on Robotics and Automation (ACRA), Brisbane, Australia, Dec 2003.

- Y. Shen and R. Featherstone, *Computer simulation of robot closed-loop dynamics for force control study*, Proceedings of The 2002 Australasian Conference on Robotics and Automation (ACRA), Auckland, New Zealand, Nov 2002.

Acknowledgements

Dr. Knut Hüper has been my supervisor and very good friend. His rigorous mathematical reasoning greatly influences my attitude on research and life. The work described in this thesis was mostly born from the lively meetings and intense discussions with Knut.

I am greatly indebted to Prof. John Moore, the head of our department, for his kind and timely helps saving me from several difficulties. In my mind, the model of a great scientist has been established from his preeminent research achievements and respectable personality.

My sweetheart, Jane, dedicated tremendous love, support and consolation. All of the calorie letting me think about hard questions came from the delicious food that she prepared for my daily meals.

My family gave me endless care and encouragement, and it's time for me to contribute something back. My parents put a great deal of effect to let me receive the best available education throughout the years I grew up. They provided me the financial support and bore the pain of their only child being away, so that I can come to Australia for the postgraduate study that I was dreaming of.

Special thanks to my Ph.D. advisors: Dr. Alex Lanzon and Dr. Robert Mahony whose advices on control or robotics were very inspirational.

Also, I would like to express my deep gratitude to Dr. Rui Cortesão who hosted and supervised me at the Institute of Systems and Robotics of the University of Coimbra, Portugal, in June 2005. Rui has introduced me his practical experience of hybrid motion/force control, which is of great use to guide me for future research on robotic manipulation.

The Department of Information Engineering is a big family with a lot of warm-hearted members, and I feel so lucky to work with you during my Ph.D. study. Here

I would like to thank Jochen Trumpf (for your clever brain helping me solving those tough math problems); Danchi Jiang and Robert Orsi (for your suggestions on control and optimization); Hongdong Li (for our cooperation in research and our friendship among you, your wife, Jane and me); Roy Featherstone (for your close guidance in the previous two years); David Austin, Nick Barnes, and Lars Petersson (for your helps on various robotics things and kind words after my first mid-term review); Chris Webers (for your help on VxWorks, and those interesting chats on either technique or politics); James Ashton and Shaun Press (for your helps on resolving all those tricky computer and network troubles); Jason Chen and Luke Cole (for your hours of skillfully making the hardware that I need for my experiment); Rosemary Shepherd (for your kind help on a lot of things that are too many to be listed here); and a lot more.

I have had countless associations within and outside ANU who are not named here but whose contribution are of great importance to my thesis. Thank you so much.

Finally, I wish to thank the following: Kaiyang Yang and Pei-Yean Lee (so good to have females who can speak Chinese with in our department); Felix Schill (so nice to be your cubic mate for so long, and to have lots of interesting chats during these several years); Shahab Kalantar (thanks for your help on Active Contour Models); Chanop Silpa-Anan (sorry I still haven't joined your Latin dance club yet); Jochen Heinzmann (thanks for your very useful tips on fighting with WAM); Grant Grubb and Leanne Matuszyk (so nice to go to Sweden and go skiing with you); Rowel Atienza (how are you in Manila? WAM is not broken yet); Luis Machado (thanks a lot for your host in Coimbra); Meiting Yue and Mark Killow (look forward to our reunion in US or Canada next year); Lin Pan and Chie-Ann Lim (very nice to travel to Alice Spring with you couple); Amy Wu, Jenny Sun, Oakio Teerapong, and Mark Tao (for our friendship started in Sydney).

Abstract

This thesis reports research results on three different topics under the theme of the automatic execution of compliant motion tasks for robot manipulators: numerical trajectory optimization, smooth interpolation of orientation, and control algorithm design. The major purpose of this research is to present a comprehensive joint-space solution for robot's hybrid motion/force control, which we believe is more general and robust than the conventional operational-space ones.

Firstly, a two-step motion planning scheme for robots subject to motion constraints is proposed. The underlying mathematical foundation of our path optimization method is calculus of variations, for which an iterative algorithm is designed to calculate the intermediate points of the resulting curve without solving a corresponding Euler-Lagrange equation.

In the following chapter, we move to the next topic of smooth interpolation of orientation, i.e., the Special Orthogonal Group SO_3, which is a smooth submanifold embedded in $\mathbb{R}^{3\times3}$. The novelty of our approach lies in its combination of rolling and wrapping with the existing pull back/push forward technique, which further yields several remarkable features appropriate for real-time applications.

The dynamics and control aspects of the compliant motion task execution are studied lastly. We will raise a brand-new joint-space formulation of hybrid motion/force control, motivated from difficulties experienced in practical experimentations on a real robotic system. Issues such as contact geometry, closed-loop dynamics, and control algorithm will all be derived in the joint-space. We believe such a melioration

gives two major advantages over the conventional operational-space approach: better applicability on general robots; and more robust performance against joint-level disturbances.

Chapter 1

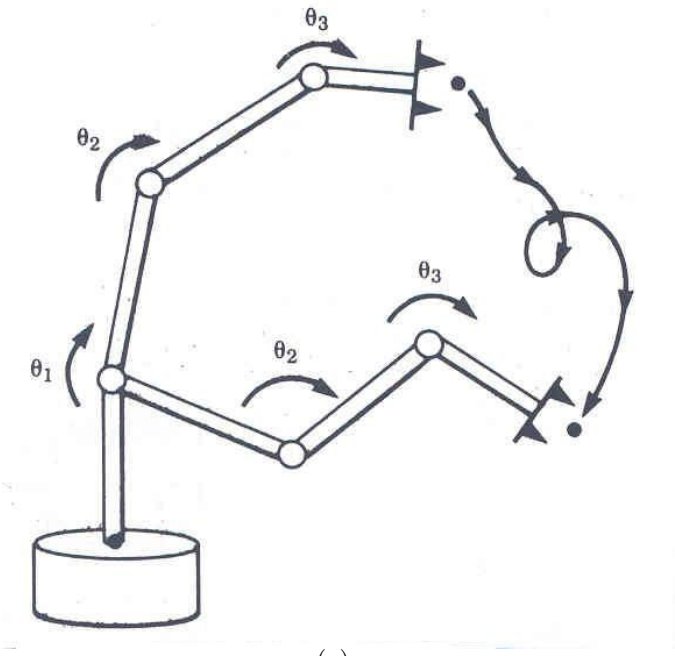
Introduction

The first industrial robot Unimate was online in a General Motors automobile factory in New Jersey in 1961, since then numerous robot manipulators have been widely installed in various manufacturing industries. Nowadays, robotic techniques have spread into many areas across manufacturing, defence, medical service, bio-industry, entertaining, and a lot more.

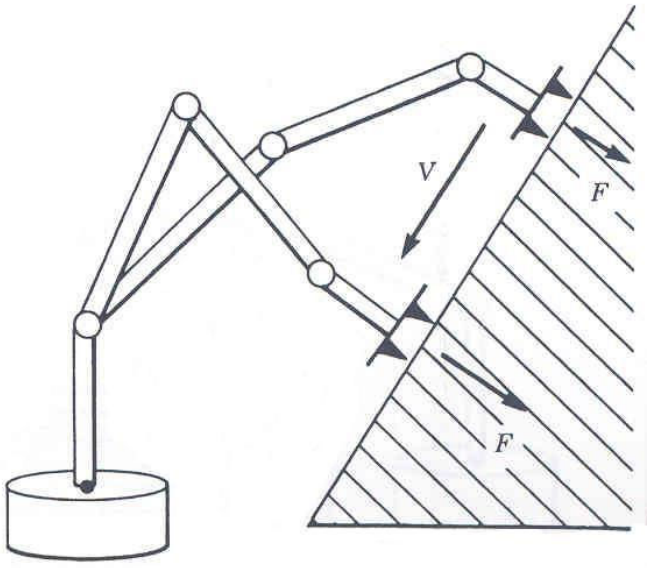
On the other hand, the study of robotics can be traced back to long time before its actual commercialization which took place in 1956 when the world's first robot company Unimation was formed. Ideas of automated machine have been recorded in many sources around the world, e.g., the first known design for a robot is commonly believed to be "the Leonardo's robot", which is a mechanical knight designed by Leonardo da Vinci around 1495.

1.1 Compliant Motion Tasks of Robot Manipulators

Industrial robots have been broadly used in many manufacturing processes of repetitively transferring objects from one location to another. Typical applications of



(a)



(b)

Figure 1.1: Different types of motion tasks for robot manipulators: a) free motion task; b) compliant motion task (courtesy of J.J. Craig)

this kind include packaging, warehouse loading and unloading, assembly of electronic boards, etc. In general, these tasks can be described as "pick and place", i.e., robots pick up an object, move it in space along some pre-defined path, then release it. Since robots can in principle move freely within its work space during the transfers, we classify this category of tasks as "free motion tasks" (Fig.1.1a).

On the other hand, robot making contact with parts, tools, or work surfaces is of great importance for many other real-world applications, e.g., screwing, deburring, grinding, etc. These type of tasks are called "complaint motion tasks" and will be mainly focused on in this thesis. In contrast with free motion tasks, robot's end-effectors are now restrained to some work surface, e.g., a wall in Fig.1.1b. Also, because robots interact with environments, the dynamics analysis in this situation becomes significantly different as the conventional rigid-body one.

Obviously, special concern on motion planning should be made under the circumstance of robots performing compliant motion tasks because of the extra motion constraints. Moreover, a force feedback is usually introduced to actively maintain the contact, because the environment's kinematic model in practice cannot be calibrated 100% accurate, which further results pure position control schemes can cause excessive forces to build up at the contact or contact to be lost with the surface.

1.2 Research Motivations and Contributions

We are aware that the majority of existing research results on robot's compliant motion control are formulated in the operational space, i.e., the 3D Cartesian task space [25, 37, 44, 48, 54, 65, 66, 69]. The major reason accounting for such a choice is that the environmental objects that robots interact with are naturally easier described

in the operational space. In fact, most of papers regard a robot to be a 6-degree-of-freedom rigid body by assuming the mapping from the operational space to the joint space is straightforward and invertible. Apparently, the reference trajectories for such operational-space complaint motion control laws should also be provided in the operational space.

Operational-space control schemes already have quite a few successful implementations reported in the literature. Nevertheless, we reckon there exist weaknesses of this approach with respect to the following several aspects

1. Applicability to general robots;

   If a robot does not have 6 degrees of freedom, the implementation of operational space analysis becomes a bit awkward. More specifically, the operational-space inertia does not exist for constrained robots; while for redundant robots, the pre-optimized information in the Jacobian's null space will be wiped out by the operational-space filtering process.

2. Optimal trajectory planning;

   Ideally, trajectory optimization should aim at some comprehensive optimality with regard to motions of both robot's joints and its end-effector. However, considering robot's kinematic model in the operational-space brings a lot of inconveniences for robot motion planning.

3. Robustness against joint-level disturbances.

   Most of disturbances in robot's control are originated at the joint level, e.g., motor torque ripples, joint friction, etc. From our experimental experience,

operational-space control algorithms are not so capable to overcome those disturbances in general, especially when robot's inertia matrix is ill-conditioned [59].

The major goal of this thesis is to bring a new methodology of robot manipulator's compliant motion control with respect to both planning and dynamics/control. In short, a joint-space solution will be presented in contrast with the conventional operational-space ones. The detailed scientific contributions of this thesis are

1. A two-step optimal joint trajectory planning scheme for robot manipulators subject to general motion constraints;

2. A smooth interpolation algorithm for rigid-body's orientation, i.e., the rotation group SO_3;

3. A joint-space formulation for robot manipulator's hybrid motion/force control.

1.3 Organization of This Thesis

Apart from the introduction and conclusion parts (Chapter 1 and 5), the main contributions of this thesis are presented in Chapter 2, 3, and 4. Although all of them are under the basic scheme of joint-space complaint motion control, each chapter is self-contained and deals with an individual research topic. Readers can focus on whatever topics that most interest you and need not worry about an unexpected dependence of one chapter on another.

Briefs of chapter 2, 3, and 4 are provided as follows:

Chapter 2 In this chapter we first present an iterative algorithm solving for variational problems with holonomic constraints whose Lagrangian function may contain derivatives up to second order. The novelty is to directly calculate the intermediate points of the boundary value path, instead of solving a corresponding nonlinear implicit Euler-Lagrange equation.

Furthermore, a novel optimal joint-trajectory planning algorithm for manipulator robots performing compliant motion tasks is proposed based on our theoretical discoveries. In general, a two-step scheme is deployed to find the optimal robot joint curve. Some numerical examples are shown at the end of this chapter, including a motion planning exercise for our 4-degree-of-freedom experimental robot WAM.

Chapter 3 This chapter investigates a novel procedure to calculate smooth interpolation curves of the rotation group SO_3, which is commonly considered as the standard representation of rigid-body's orientations. The algorithm is a combination of rolling and wrapping with the pull back/push forward technique, yielding a few desirable features appropriate for real-time applications. Also, a numerical example along with some visualization results are presented at the end of this chapter.

Chapter 4 This chapter presents a joint space formulation for robot manipulator's hybrid motion/force control. Contact geometry and closed-loop dynamics will be derived in this chapter, also a joint-space hybrid control scheme will be proposed. Some simulation results are shown to verify the applicability of our theory on a constrained (4-degree-of-freedom) robot WAM. At the end of this

chapter, we suggest a compliant motion control experiment of WAM performing a 2-DOF motion and 2-DOF force task.

Chapter 2

Newton's Method for Constrained Variational Problems with Applications to Robot Path Planning

In this chapter we first present an iterative algorithm solving for variational problems with holonomic constraints whose Lagrangian function may contain derivatives up to second order. This research is an extension of the recent work of Levin *et al.* [40]. The novelty is to directly calculate the intermediate points of the boundary value path, instead of solving a corresponding nonlinear implicit Euler-Lagrange equation. In appendix A, we give a proof that the result of our iterative calculation scheme will converge to the result of the Euler-Lagrange equation as the time difference h inclines to 0.

Furthermore, a novel optimal joint-trajectory planning algorithm for manipulator robots performing compliant motion tasks is proposed based on our theoretical discoveries. In general, a two-step scheme is deployed to find the optimal robot joint curve. Firstly, we approximate the functional and use Newton's iteration to numerically calculate the joint trajectory's intermediate discretized points. Secondly, we

interpolate these points to get the final joint curve in a way such that the motion constraints will always be sustained throughout the movement.

Some numerical examples are shown at the end of this chapter, including a motion planning exercise for our 4-degree-of-freedom experimental robot WAM.

2.1 Background and Literature Review

2.1.1 Calculus of Variations

The classical calculus of variations was originated about 300 years ago in connection with mechanical problems. The so-called brachistochrone problem,

$$\min_{y(x)} \int_{x_0}^{x_n} \frac{\sqrt{1+\dot{y}(x)^2}}{\sqrt{y(x)}}\, \mathrm{d}x, \tag{2.1.1}$$

is probably one of the earliest variational problem in the history of mathematics. It was firstly formulated and solved by Johann Bernoulli with cycloid in 1696. "Brachistochrone" is Greek for "shortest time", and the brachistochrone problem is to find the curve that will yield the shortest possible time for the descent of an object from rest and accelerated by gravity without friction [61].

Generally speaking, calculus of variations looks for a curve for which a given integral attains a stationary value (for physical systems, it is often a minimum or maximum).

E.g., the so-called "simplest problem of variational calculus" is defined as

P2.1 Minimize the integral

$$J(y) = \int_{t_0}^{t_n} L\big(t, y(t), \dot{y}(t)\big)\, \mathrm{d}t \tag{2.1.2}$$

with respect to the class of C^2-smooth curves $y : [t_0, t_n] \to \mathbb{R}$, $t \mapsto y(t)$, which satisfies the boundary conditions (for boundary problems)

$$y(t_0) = y_0,\ y(t_n) = y_n, \tag{2.1.3}$$

or the initial conditions (for initial problems)

$$y(t_0) = y_0,\ \dot{y}(t_0) = \xi_0, \tag{2.1.4}$$

and the integrand L (often called the Lagrangian function) here is defined as $L : [t_0, t_n] \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $\big(t, y(t), \dot{y}(t)\big) \mapsto L(t, y, \dot{y})$.

The solution of **P2.1** satisfies a differential equation, namely the so-called Euler-Lagrange equation,

$$\frac{\partial L}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\partial L}{\partial \dot{y}}\Big) = 0. \tag{2.1.5}$$

Showing Eq. 2.1.5 is the necessary condition for the minimal curve $y(t)$ involves using a mathematical theory named calculus of variations. Here we only briefly show a few key steps of the derivation procedure. For more details on variational calculus, please refer to [61, 62, 67].

Suppose we have found a C^2-curve $y^*(t)$ which renders $J(y)$ a minimum value. Now perturb $y^*(t)$ by another smooth curve $z : [t_0, t_n] \to \mathbb{R}$, $t \mapsto z(t)$ with $z(t_0) = z(t_n) = 0$. Namely let $y(t) = y^*(t) + \varepsilon z(t)$, where $\varepsilon \in \mathbb{R}$ is a small parameter. Having written $y(t)$ in this way, we can define $\tilde{J} : \mathbb{R} \to \mathbb{R}, \varepsilon \mapsto J(y^* + \varepsilon z)$, and the real function $\tilde{J}(\varepsilon)$ has a minimum value at $\varepsilon = 0$. By using elementary calculus, we know that $\mathrm{d}\tilde{J}/\mathrm{d}\varepsilon = 0$ when $\varepsilon = 0$.

We will see $\mathrm{d}\tilde{J}/\mathrm{d}\varepsilon = 0$ is equivalent to Eq. 2.1.5 from the expansion below. The chain rule is used to go from the second equation to the third, whose middle term

will disappear because $z(t_0) = z(t_n) = 0$.

$$\begin{aligned}
\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} &= \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\tilde{J}\big(t, y+\varepsilon z, \dot{y}+\varepsilon\dot{z}\big)\Big|_{\varepsilon=0} \\
&= \int_{t_0}^{t_n} \frac{\partial L}{\partial y}\, z\,\mathrm{d}t + \int_{t_0}^{t_n} \frac{\partial L}{\partial \dot{y}}\,\dot{z}\,\mathrm{d}t \\
&= \int_{t_0}^{t_n} \frac{\partial L}{\partial y}\, z\,\mathrm{d}t + \frac{\partial L}{\partial \dot{y}}\, z\Big|_{t=t_0}^{t=t_n} - \int_{t_0}^{t_n} z\,\frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\partial L}{\partial \dot{y}}\Big)\,\mathrm{d}t \\
&= \int_{t_0}^{t_n} z\left(\frac{\partial L}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\partial L}{\partial \dot{y}}\Big)\right)\mathrm{d}t = 0. \qquad (2.1.6)
\end{aligned}$$

In Eq. 2.1.6, as z can be any arbitrary smooth curve as long as it satisfies the boundary conditions, so the only possibility for the whole equation to be zero is that the terms inside the big parentheses equal 0, which is exactly Eq. 2.1.5.

Ideally, we can get a closed form of $y(t)$ from the Euler-Lagrange equation Eq. 2.1.5 together with conditions Eq. 2.1.3 or Eq. 2.1.4. However, it is in general very difficult to solve a differential equation symbolically. For **P2.1**, nonlinearity of the Lagrangian function L will merge into the Euler-Lagrange equation, which makes the differential equation very tricky to handle. Miele and Pritchard [50] proposed an iterative approach which keeps updating $y(t)$'s symbolic expression with a tiny displacement (the scaled solution of a differential equation) until the overall cost function J reaches some acceptable small value. With sufficiently small step size at each iteration, the integral of the Lagrangian function is forced to decrease. However, the actual implementation is expected to be very tough as $y(t)$'s form will get more and more complicated as the iteration goes on. Elnagar *et al.* [22, 23] approximated the resulting curve as a multi-order Lagrange polynomial, then applied some Newton-type methods to determine the coefficients of such prior-structured model. Comparing with the first method, this approach is more realistic, but there are still a quite big number of parameters for

the optimizer to calculate, which may make such an algorithm fall into "the curse of dimension".

In some cases, it's good enough to know the value of the optimal curve only at a bunch of discrete time instants, for example, if we are asked to design reference tracking trajectories for discrete control systems. Under this circumstance, we can try numerically solving the differential equation as an alternative way to tackle variational problems. Leok [39] and Lew *et al.* [41, 42] presented the variational time integration algorithms which can numerically solve the initial problem (Eq. 2.1.4) by propagating the discretized configuration curve $\mathbf{y}(t_k)$ along the time axis according to the discretized Euler-Lagrange equation. Such techniques are very useful in studying dynamics of mechanical systems.

Comparatively, boundary problem is much more complicated. Gregory and Yang [28] showed the discretized Euler-Lagrange equation for computing the resulting curve's intermediate discretized points, and the equation is derived through picking a special perturbation function $z(t)$ whose integral and derivative's integral is known. Levin *et al.* [40] obtained similar results as what Gregory and Yang have got, but their analysis is done by representing the whole cost function with the discretized $y(t)$ (with which $\dot{y}(t)$ is approximated), then apply the Gauss-Newton algorithm to perform the optimization process (they somehow convert the overall cost function into a summation of plenty of quadratical forms).

Considering more general situations, we can extend **P2.1** in two directions. Firstly, the Lagrangian function may contain second order derivatives (i.e., L upgrades to $[t_0, t_n] \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $\big(t, y(t), \dot{y}(t), \ddot{y}(t)\big) \mapsto L(t, y, \dot{y}, \ddot{y})$).

P2.2 Minimize the integral

$$J(y) = \int_{t_0}^{t_n} L\big(t, y(t), \dot{y}(t), \ddot{y}(t)\big)\ \mathrm{d}t \tag{2.1.7}$$

with respect to the class of C^4-smooth curves $y(t)$ which satisfies the boundary conditions

$$y(t_0) = y_0,\ y(t_n) = y_n,\ \dot{y}(t_0) = \xi_0,\ \dot{y}(t_n) = \xi_n. \tag{2.1.8}$$

Similar as getting Eq.2.1.5, we can derive the corresponding Euler-Lagrange equation for **P2.2** by using variational calculus as below

$$\frac{\partial L}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\partial L}{\partial \dot{y}}\Big) + \frac{\mathrm{d}^2}{\mathrm{d}t^2}\Big(\frac{\partial L}{\partial \ddot{y}}\Big) = 0. \tag{2.1.9}$$

Secondly, we may restrain the resulting curve with some holonomic constraints (referring to no occurrence of $y(t)$'s derivatives). Often such constraints can be expressed explicitly.

P2.3 Similar to **P2.2**, find a smooth curve $\mathbf{y}(t)$ minimizing Eq.2.1.7 under the boundary conditions Eq.2.1.8, besides $\mathbf{y}(t)$ is further subject to

$$C\big(\mathbf{y}(t)\big) = 0,\ \forall t \in [t_0, t_n]. \tag{2.1.10}$$

E.g., if the constraint is a unit sphere in a 3D Cartesian coordinate system: $S^2 = \{\mathbf{y}_1^2 + \mathbf{y}_2^2 + \mathbf{y}_3^2 = 1\}$, then

$$C\big(\mathbf{y}(t)\big) = \mathbf{y}_1^2(t) + \mathbf{y}_2^2(t) + \mathbf{y}_3^2(t) - 1 = 0,\ \forall t \in [t_0, t_n]. \tag{2.1.11}$$

To solve constrained variational problems, one possible approach is to introduce a new coordinate system, sometimes called "Lagrange coordinates of the second kind" [67], so that the side conditions, such as $C(\mathbf{y})$ in Eq.2.1.11, are automatically fulfilled.

E.g., seek for an optimal curve on a 2D plane with regard to

$$\min_{\mathbf{y}(t)} \int_{t_0}^{t_n} \sqrt{\dot{\mathbf{y}}_1^2(t) + \dot{\mathbf{y}}_2^2(t)}\ \mathrm{d}t, \tag{2.1.12}$$

$$\mathbf{y}_1(t_0) = \mathbf{y}_{10},\ \mathbf{y}_1(t_n) = \mathbf{y}_{1n},\ \mathbf{y}_2(t_0) = \mathbf{y}_{20},\ \mathbf{y}_2(t_n) = \mathbf{y}_{2n}, \tag{2.1.13}$$

with the side condition

$$\mathbf{y}_1^2(t) + \mathbf{y}_2^2(t) = 1,\ \forall t \in [t_0, t_n]. \tag{2.1.14}$$

If we let

$$\mathbf{y}_1 = \cos(\theta),\ \mathbf{y}_2 = \sin(\theta), \tag{2.1.15}$$

we can obtain a 1-D variational problem without the side constraint,

$$\min_{\theta(t)} \int_{t_0}^{t_n} \sqrt{\sin^2\big(\theta(t)\big) + \cos^2\big(\theta(t)\big)}\ \mathrm{d}t, \tag{2.1.16}$$

$$\begin{aligned} \cos\big(\theta(t_0)\big) = \mathbf{y}_{10},\ \sin\big(\theta(t_0)\big) = \mathbf{y}_{20}, \\ \cos\big(\theta(t_n)\big) = \mathbf{y}_{1n},\ \sin\big(\theta(t_n)\big) = \mathbf{y}_{2n}. \end{aligned} \tag{2.1.17}$$

The above approach makes it possible to employ analytical results of general manifolds in differential geometry. However, in the meantime, the optimization process utilizes abstract concepts and the Lagrange coordinates are highly dependent on the geometrical properties of the manifold described by $C(\mathbf{y})$, therefore the implementation may become tricky sometimes. Comparatively, the Lagrange multiplier technique is a more straightforward and widely used method for solving constrained variational problems. For **P2.3**, we can introduce a function (i.e., the Lagrange multiplier) $\mu : [t_0, t_n] \to \mathbb{R}$, and the original variational problem is equivalent to

P2.4

$$\min_{\mathbf{y}(t),\mu(t)} \int_{t_0}^{t_n} \Big(L\big(t, \mathbf{y}(t), \dot{\mathbf{y}}(t), \ddot{\mathbf{y}}(t)\big) + \mu(t) C\big((\mathbf{y}(t)\big) \Big)\ \mathrm{d}t \tag{2.1.18}$$

with $\mathbf{y}(t)$ satisfying the boundary conditions

$$\mathbf{y}(t_0) = \mathbf{y}_0,\ \mathbf{y}(t_n) = \mathbf{y}_n,\ \dot{\mathbf{y}}(t_0) = \xi_0,\ \dot{\mathbf{y}}(t_n) = \xi_n. \tag{2.1.19}$$

The corresponding Euler-Lagrange equation system for **P2.4** is

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{y}} - \frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\partial L}{\partial \dot{\mathbf{y}}}\Big) + \frac{\mathrm{d}^2}{\mathrm{d}t^2}\Big(\frac{\partial L}{\partial \ddot{\mathbf{y}}}\Big) + \mu\frac{\mathrm{d}C}{\mathrm{d}\mathbf{y}} &= 0, \\ C\big(\mathbf{y}(t)\big) &= 0. \end{aligned} \tag{2.1.20}$$

Ahlbrandt *et al.* [4] and Crouch *et al.* [17] looked at some particular constrained variational problems. They both succeeded in eliminating the Lagrange multiplier function and establishing the Euler-Lagrange equation explicitly for their specified cost function and constraints, on which numerical approaches can further be applied. However, in more general cases, such an Euler-Lagrange equation in $\mathbf{y}(t)$ turns out to be an implicit differential equation, e.g., the example on robot path planning shown in Section 2.5.

2.1.2 Robot Path Planning

Generally speaking, the task for a motion planner is to specify a motion to be executed by actuators. Properly planned motion can have advantages with respect to different aspects, e.g., obstacle avoidance, work efficiency optimization, better tracking performance, etc. More specifically, for multi-link robotic systems, the reference trajectory generation can be divided into the following two subproblems.

Pr2.1 For a given robot and task, plan a path for the end-effector connecting two or more specified configurations (may include both position and orientation). Such a path should satisfy either equality (e.g., robot's end-tip is required to

move on a working surface) or inequality (e.g., obstacle avoidance) constraints; in the meantime, it may optimize a performance index.

Pr2.2 For a given end-effector path expressed in the operational space (which usually coincides with the Cartesian space), find its corresponding joint trajectory through inverse kinematics. Similarly, some performance index can be optimized in case of a redundant robot, namely, the robot has more degrees of freedom (DOFs) than necessary to perform the given task.

The conventional strategy for robot path planning often requires **Pr2.1** and **Pr2.2** to be resolved separately. The necessity of individually solving **Pr2.1** comes from the facts that:

1. the geometrical shape of end-effector path is very important for the automatic task execution,

2. some optimization criterions or constraint conditions are naturally easier described in the operational space, e.g., the presence of obstacles [56].

For **Pr2.1**, it is a widely researched topic to design a collision-free configuration path for a single rigid object travelling in a crowded environment, and most of such algorithms can be found in Latombe's work [38].

However, the drawback of the above approach is its computational difficulties or inefficiencies artificially introduced by the isolated processing in the operational and joint spaces. Firstly, it may be cumbersome to generalize the end-effector's path planning algorithm described in [38] on a multi-rigid-body manipulator due to the robot's feasible configuration space constrained by its nonlinear kinematics and

joints' mechanical stops. Secondly, as the robot's kinematic model is usually ignored at the end-effector path planning stage, the resulting joint motion may contain some unpredictable behaviors, e.g., when the robot is in the neighborhood of a singularity [15].

Recently, authors have adopted the methodology of solving **Pr2.1** and **Pr2.2** in one attempt by casting a robot path planning problem as an optimal control problem [60, 64], some even seek for the optimal time history of joint torques [12, 45]. Most of these papers are aiming towards the minimum execution time under the constraint of drive torque limit (some consider obstacle avoidance as well). Since the path planning is directly performed in the joint space, such an approach will automatically eliminate the necessity of calculating the feasible configuration space for the manipulator. However, path optimization incorporating robot dynamics will often end up with a suboptimal result. The reason is that apart from the multi-rigid-body model, some dynamics factors which are either difficult to model (e.g., motor torque ripple) or not necessarily smooth functions (e.g., Coulomb friction) are usually beyond the scope of optimization.

Based on our iterative algorithm for constrained varitational problems to be presented in the next section, here we study the optimization of a robot path in the joint space (also solving **Pr2.1** and **Pr2.2** at the same time) with regard to some synthetical geometric performance index for the motions of both robot's joint and end-effector as well. The idea is to optimize not only the joint trajectory but also its resulting end-effector path in the eye of the joint space, since the motion of robot's end-effector is ultimately driven by that of its joints through a surjective forward kinematic map, for which a fairly accurate model is usually possible to obtain. Our

'lump-sum' path planning approach will have the advantage on achieving such a synthetical optimality, as the anterior step of the conventional strategy is usually blind to the final joint trajectory's optimality.

Furthermore, rather than the free space motion, we consider the situation that the robot's end-effector is subject to some motion constraints. A typical example is that the manipulator is performing a compliant motion task, i.e., its end-effector is only allowed to move on some working surface. The extreme case in this category is that the end-effector path is totally determined before we carry out the path planning. Then our trajectory-optimization algorithm will downgrade to a solver for only **Pr2.2** as in [3, 47], except that we allow the cost function to contain second order derivatives and we avoid solving the Euler-Lagrange differential equation.

The rest of the chapter is organized as follows: Section 2 gives the formulation of the constrained variational problem we consider for robot path planning applications; Section 3 and 4 present the mathematical foundations as well as the two-step trajectory-optimization algorithm; Section 5 shows several numerical examples of applying the algorithm on both theoretical variational problems and practical robot-motion-planning exercises. The attached CD contains some videos of both animations and experiments of the 4-DOF robot manipulator WAM.

2.2 Problem Description

This section gives the mathematical formulation of our optimal robot path planning problem, which can be classified as a variational problem presented in Section 2.1.1.

First of all, we are considering the situation that a robot is performing some

compliant motion task, i.e., the manipulator is subject to l end-effector constraints (with respect to linear position and orientation)

$$C_i\big(\mathbf{p}(t), \mathbf{R}(t)\big) = C_i\Big(\kappa_{\mathbf{p}}\big(\mathbf{q}(t)\big), \kappa_{\mathbf{R}}\big(\mathbf{q}(t)\big)\Big) = 0, \;\; i = 1, \cdots, l, \tag{2.2.1}$$

where $\mathbf{q} : \mathbb{R} \to \mathbb{R}^q$, $t \mapsto \mathbf{q}(t)$ is the q-dim joint curve, $\mathbf{p} : \mathbb{R} \to \mathbb{R}^3$, $t \mapsto \mathbf{p}(t)$ is the path of end-effector's linear position, and $\mathbf{R} : \mathbb{R} \to SO_3$, $t \mapsto \mathbf{R}(t)$ is the path of end-effector's orientation. Moreover, $\kappa_{\mathbf{p}} : \mathbb{R}^q \to \mathbb{R}^3$, $\mathbf{q}(t) \mapsto \mathbf{p}\big(\mathbf{q}(t)\big)$ and $\kappa_{\mathbf{R}} : \mathbb{R}^q \to SO_3$, $\mathbf{q}(t) \mapsto \mathbf{R}\big(\mathbf{q}(t)\big)$ are the manipulator's forward position and orientation kinematic maps, respectively.

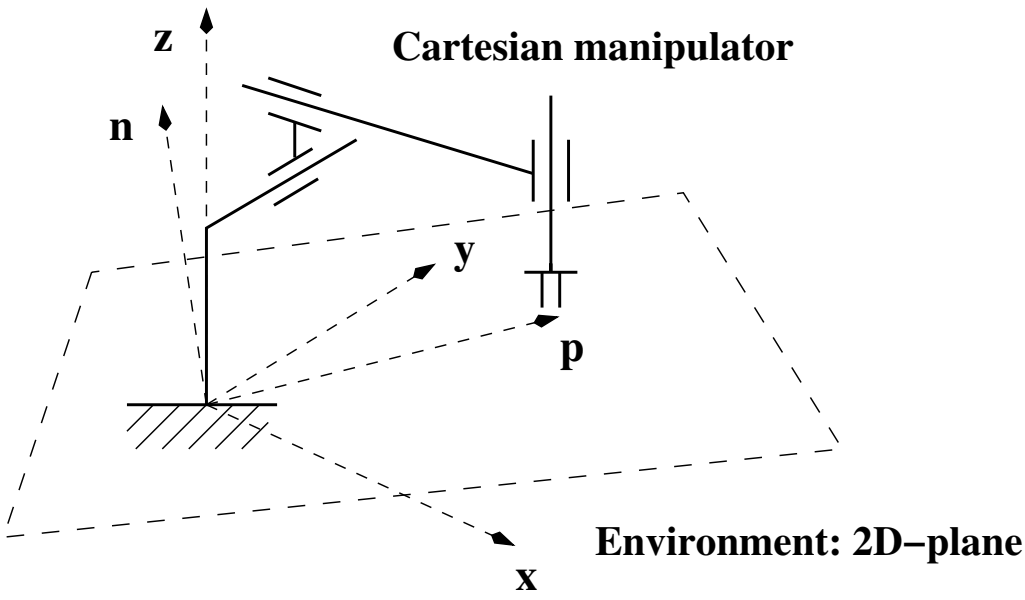


Figure 2.1: Compliant motion task example

For example, suppose the manipulator's end-effector is required to move on a 2D plane in $\mathbb{R}^3$ (Fig.2.1), e.g., the manipulator is wiping a flat window. Then the

corresponding motion constraint can be expressed as

$$\begin{aligned} C\big(\mathbf{q}(t)\big) &= \kappa_{\mathbf{p}}\big(\mathbf{q}(t)\big)^{\top}\mathbf{n} \\ &= \mathbf{p}(t)^{\top}\mathbf{n} = 0 \end{aligned} \tag{2.2.2}$$

where $\mathbf{n} \in \mathbb{R}^3$ is the plane normal.

Another situation is that the path of the end-effector is directly determined by the given task itself, e.g., the robot performs cutting, welding, and so on. Then the motion constraints may be given as

$$\begin{cases} C_1\big(\mathbf{q}(t)\big) = \kappa_{\mathbf{p}}\big(\mathbf{q}(t)\big) - \mathbf{p}_d(t) = 0 \\ C_2\big(\mathbf{q}(t)\big) = \kappa_{\mathbf{R}}\big(\mathbf{q}(t)\big) - \mathbf{R}_d(t) = 0 \end{cases} \tag{2.2.3}$$

where $\mathbf{p}_d : \mathbb{R} \to \mathbb{R}^3$, $t \mapsto \mathbf{p}_d(t)$ and $\mathbf{R}_d : \mathbb{R} \to SO_3$, $t \mapsto \mathbf{R}_d(t)$ are the desired path of end-effector's linear position and orientation, respectively.

With the end-effector constraints, our robot path optimization problem is given as

Pr2.3 (A combination of **Pr2.1** and **Pr2.2**)

Over the set (named $\mathcal{S}_{\mathcal{C}}$) of sufficiently smooth curves subject to the motion constraints and boundary conditions:

$$C_i\big(\mathbf{q}(t)\big) = 0,\ \forall t \in [t_0, t_n],\ i = 1, \cdots, l, \tag{2.2.4}$$

$$\begin{aligned} \mathbf{q}(t_0) = \mathbf{q}_0,\ \mathbf{q}(t_n) = \mathbf{q}_n, \\ \dot{\mathbf{q}}(t_0) = \xi_0,\ \dot{\mathbf{q}}(t_n) = \xi_n, \end{aligned} \tag{2.2.5}$$

find a joint trajectory $\mathbf{q} : [t_0, t_n] \to \mathbb{R}^q$ minimizing the cost function $J : \mathcal{S}_{\mathcal{C}} \to \mathbb{R}$, $\mathbf{q} \mapsto J(\mathbf{q})$:

$$J(\mathbf{q}) = \int_{t_0}^{t_n} \Big(L\big(\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t)\big) \Big)\, \mathrm{d}t, \tag{2.2.6}$$

where

$$L\big(t,\mathbf{q}(t),\dot{\mathbf{q}}(t),\ddot{\mathbf{q}}(t)\big) = L_1\big(t,\mathbf{q}(t),\dot{\mathbf{q}}(t)\big) + L_2\big(t,\mathbf{q}(t),\ddot{\mathbf{q}}(t)\big), \tag{2.2.7}$$

$$L_1 : [t_0,t_n]\times\mathbb{R}^q\times\mathbb{R}^q\to\mathbb{R},\ \big(t,\mathbf{q}(t),\dot{\mathbf{q}}(t)\big)\mapsto L_1(t,\mathbf{q},\dot{\mathbf{q}}),$$

$$L_2 : [t_0,t_n]\times\mathbb{R}^q\times\mathbb{R}^q\to\mathbb{R},\ \big(t,\mathbf{q}(t),\ddot{\mathbf{q}}(t)\big)\mapsto L_2(t,\mathbf{q},\ddot{\mathbf{q}}).$$

The reason for splitting the integrand of J into the sum of L_1+L_2 will be explained in later sections. Nevertheless, besides $\mathbf{q}(t)$ and its first or second order derivatives, the above optimization objective stereotype can also represent the end-effector's static linear position $\mathbf{p}(t)\in\mathbb{R}^3$ or orientation $\mathbf{R}(t)\in SO_3$ and its linear $\dot{\mathbf{p}}(t)\in\mathbb{R}^3$ or angular $\omega(t)\in\mathbb{R}^3$ velocity by using Eqs.2.2.8,2.2.9.

$$\mathbf{p}(t) = \kappa_{\mathbf{p}}\big(\mathbf{q}(t)\big),\ \mathbf{R}(t) = \kappa_{\mathbf{R}}\big(\mathbf{q}(t)\big), \tag{2.2.8}$$

$$\begin{bmatrix}\omega(t)\\ \dot{\mathbf{p}}(t)\end{bmatrix} = \mathbf{J}\big(\mathbf{q}(t)\big)\,\dot{\mathbf{q}}(t) + \begin{bmatrix}0\\ \omega(t)\times\mathbf{p}(t)\end{bmatrix}, \tag{2.2.9}$$

where $\mathbf{J}(t):\mathbb{R}^q\to\mathbb{R}^6$, $\dot{\mathbf{q}}(t)\mapsto\hat{\mathbf{v}}(t)$ is the Jacobian matrix whose symbolic expression in $\mathbf{q}(t)$ is easy to derive [24], and $\hat{\mathbf{v}}(t)$ is the rigid body velocity [51] of robot's end-effector expressed in the inertial frame.

Again, if we introduce Lagrange multiplier functions $\mu_i : [t_0,t_n]\to\mathbb{R}$, $i = 1\cdots l$, with which **Pr2.3** can be established equivalently as

Pr2.4 Find a sufficiently smooth curve $\mathbf{q}$ satisfying the boundary conditions (Eq.2.2.5) and a set of curves μ_i, such that the following function $\tilde{J}$ attains a minimal value.

$$\tilde{J}(\mathbf{q},\mu_i) = \int_{t_0}^{t_n}\Big(L_1(\mathbf{q},\dot{\mathbf{q}}) + L_2(\mathbf{q},\ddot{\mathbf{q}}) + \sum_{i=1}^{l}\mu_i C_i(\mathbf{q})\Big)\,\mathrm{d}t \tag{2.2.10}$$

Using calculus of variations, we know that the optimal joint curve for the above path planning problem satisfies the Euler-Lagrange equation system

$$\begin{cases} \dfrac{\partial L_1}{\partial \mathbf{q}} - \dfrac{\mathrm{d}}{\mathrm{d}t}\Big(\dfrac{\partial L_1}{\partial \dot{\mathbf{q}}}\Big) + \dfrac{\partial L_2}{\partial \mathbf{q}} + \dfrac{\mathrm{d}^2}{\mathrm{d}t^2}\Big(\dfrac{\partial L_2}{\partial \ddot{\mathbf{q}}}\Big) + \displaystyle\sum_{i=1}^{l} \mu_i \frac{\mathrm{d}C_i}{\mathrm{d}\mathbf{q}} = 0 \\ C_i = 0,\ i = 1, \cdots, l. \end{cases} \tag{2.2.11}$$

Eq.2.2.11 is in general quite difficult to handle. The problem is that the resulting differential equation in $\mathbf{q}$ may be implicit, even if we are able to eliminate the Lagrange multipliers μ_i. This happens for example if the motion of the end-effector is also within the scope of our trajectory optimization, meaning L_1 incorporates robot's nonlinear kinematic model. In face of the difficulties of solving (even numerically) an implicit differential equation, we propose a two-step path planning scheme which firstly computes $\mathbf{q}$'s discretization and then interpolates the corresponding points to obtain an approximate solution curve of Eq.2.2.11. The next section will present the mathematical foundations of the first step's calculation, and the overall path planning algorithm will be given in Section 2.4.

2.3 Discretization, Approximation, and Integration Scheme

First of all, we repeat the variational problem studied in this thesis.

P2.5

$$\min_{\mathbf{y}} \ J(\mathbf{y}) = \int_{t_0}^{t_n} L_1\big(t, \mathbf{y}(t), \dot{\mathbf{y}}(t)\big) + L_2\big(t, \mathbf{y}(t), \ddot{\mathbf{y}}(t)\big)\ \mathrm{d}t \tag{2.3.1}$$

with respect to the class of sufficiently smooth functions $\mathbf{y} : [t_0, t_n] \to \mathbb{R}^p$, $t \mapsto \mathbf{y}(t)$, which satisfies the boundary conditions

$$\mathbf{y}(t_0) = \mathbf{y}_0,\ \mathbf{y}(t_n) = \mathbf{y}_n,\ \dot{\mathbf{y}}(t_0) = \xi_0,\ \dot{\mathbf{y}}(t_n) = \xi_n, \tag{2.3.2}$$

and the holonomic constraint (for simplicity, we let $l = 1$)

$$C\big(\mathbf{y}(t)\big) = 0,\ \forall t \in [t_0, t_n]. \tag{2.3.3}$$

As we know, if a Lagrange multiplier $\mu(t)$ is introduced, **P2.5** is equivalent to

$$\min_{\mathbf{y}(t),\mu(t)} \tilde{J}(\mathbf{y}, \mu) = \int_{t_0}^{t_n} \Big(L_1\big(t, \mathbf{y}(t), \dot{\mathbf{y}}(t)\big) + L_2\big(t, \mathbf{y}(t), \ddot{\mathbf{y}}(t)\big) + \mu(t) C\big(\mathbf{y}(t)\big) \Big)\, \mathrm{d}t, \tag{2.3.4}$$

whose solution satisfies the Euler-Lagrange equation system

$$\begin{aligned} \frac{\partial L_1}{\partial \mathbf{y}} - \frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\partial L_1}{\partial \dot{\mathbf{y}}}\Big) + \frac{\partial L_2}{\partial \mathbf{y}} + \frac{\mathrm{d}^2}{\mathrm{d}t^2}\Big(\frac{\partial L_2}{\partial \ddot{\mathbf{y}}}\Big) + \mu \frac{\mathrm{d}C}{\mathrm{d}\mathbf{y}} = 0, \\ C\big(\mathbf{y}(t)\big) = 0. \end{aligned} \tag{2.3.5}$$

To solve for **P2.5**, we will propose a Newton-like algorithm, which can be summarized into the following four steps and the overall data-flow structure is illustrated in Fig.2.2.

1. $\mathbf{y}(t)$ and $\mu(t)$ are discretized with the discretization results respectively denoted by $\mathbf{y}_k$, and μ_k, as at this stage we are only interested in knowing the values of the resulting optimal curve at discrete time instants.

2. Use the neighbor points to approximate the first and second derivatives of $\mathbf{y}$. Note that $\dot{\mathbf{y}}$ and $\ddot{\mathbf{y}}$ will be treated under different time schemes, which accounts for separating L as L_1 and L_2.

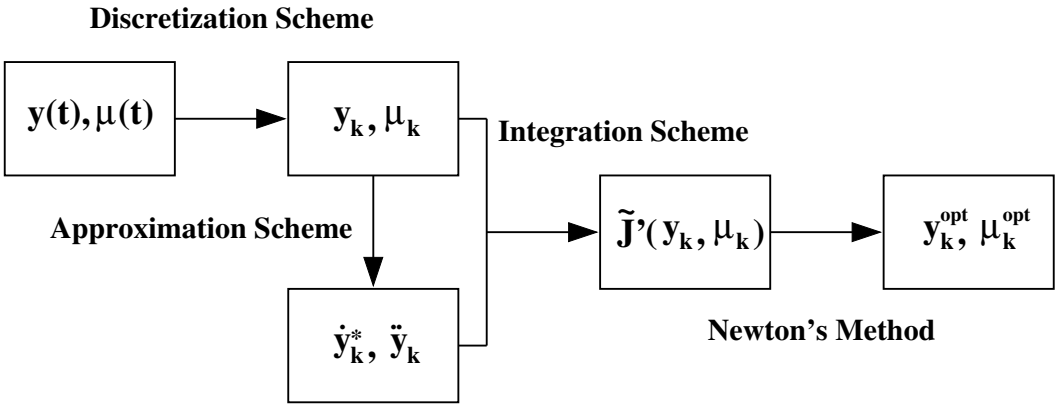


Figure 2.2: Overview of Newton's method for constrained variational Problems

3. The cost function $\tilde{J}$ can further be represented by $\mathbf{y}_k$, $\dot{\mathbf{y}}_k^*$ ($\dot{\mathbf{y}}_k^*$ will be defined in the next page), $\ddot{\mathbf{y}}_k$, and μ_k. Therefore, the infinite-dimensional variational problem has finally been converted into a finite-dimensional optimization problem. The dimension of which is directly governed by the time step size.

4. Apply some developed technique (e.g., the Newton's method) to solve for the optimization problem.

It is non-trivial that the characteristics of the original variational problem get preserved through this processing, in other words, as we discretize $\mathbf{y}(t)$ and $\mu(t)$ with an infinite number of points, the result from applying Newton's method converges to the solution of the Euler-Lagrange equation. Technical details for each step will be revealed gradually next, and the convergence proof is given in Appendix A.

Consider a regular partition of the time interval $[t_0, t_n]$,

$$t_k = kh \ , \ k \in \{0, 1, \cdots, n-1, n\}, \tag{2.3.6}$$

where $h = (t_n - t_0)/n$ is the step size.

Let

$$\mathbf{y}_k := \mathbf{y}(t_k)\ ,\ k \in \{0, 1, \cdots, n-1, n\}, \tag{2.3.7}$$

and

$$\mu_k := \mu(t_k)\ ,\ k \in \{0, 1, \cdots, n-1, n\}. \tag{2.3.8}$$

In this thesis, $\mathbf{y}_k$ or μ_k is called "the discretized point" of $\mathbf{y}(t)$ or $\mu(t)$. Also, we define two discretization operators $\mathbf{P}_h^{n+1}$ and $\mathbf{P}_h^{n-1}$. Feeding a p-dim vector function $\mathbf{y}(t)$ into $\mathbf{P}_h^{n+1}$ or $\mathbf{P}_h^{n-1}$, the output is respectively an $(n+1)\times p$ or $(n-1)\times p$ matrix, i.e.,

$$\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big) = \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{n-1} \\ \mathbf{y}_n \end{bmatrix}, \tag{2.3.9}$$

$$\mathbf{P}_h^{n-1}\big(\mathbf{y}(t)\big) = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{n-1} \end{bmatrix}. \tag{2.3.10}$$

Meanwhile, we define t_k^* as

$$t_k^* := kh + \frac{h}{2}\ ,\ k \in \{0, 1, \cdots, n-1\}. \tag{2.3.11}$$

Similar as above, we can have

$$\mathbf{y}_k^* := \mathbf{y}(t_k^*)\ ,\ k \in \{0, 1, \cdots, n-1\}, \tag{2.3.12}$$

whose corresponding discretization operator ${\mathbf{P}^*}_h^n$ maps p-dim $\mathbf{y}(t)$ into an $n\times p$ matrix

$${\mathbf{P}^*}_h^n\big(\mathbf{y}(t)\big) = \begin{bmatrix} \mathbf{y}_0^* \\ \mathbf{y}_1^* \\ \vdots \\ \mathbf{y}_{n-1}^* \end{bmatrix}. \tag{2.3.13}$$

Now consider representing $\mathbf{y}_k^*$ and $\dot{\mathbf{y}}_k^*$ by $\mathbf{y}_k$,

$$\begin{aligned} \mathbf{y}_k^* &\approx \frac{\mathbf{y}_k + \mathbf{y}_{k+1}}{2}, && (2.3.14)\\ \dot{\mathbf{y}}_k^* &\approx \frac{\mathbf{y}_{k+1} - \mathbf{y}_k}{h}, && (2.3.15) \end{aligned}$$

and $\ddot{\mathbf{y}}_k$ by $\mathbf{y}_k$, ξ_0 and ξ_n,

$$\begin{aligned} \ddot{\mathbf{y}}_0 &\approx \frac{2\mathbf{y}_1 - 2\mathbf{y}_0}{h^2} - \frac{2}{h}\xi_0,\\ \ddot{\mathbf{y}}_k &\approx \frac{\mathbf{y}_{k-1} - 2\mathbf{y}_k + \mathbf{y}_{k+1}}{h^2}, && (2.3.16)\\ \ddot{\mathbf{y}}_n &\approx \frac{2\mathbf{y}_{n-1} - 2\mathbf{y}_n}{h^2} + \frac{2}{h}\xi_n. \end{aligned}$$

Rewrite Eqs. 2.3.14, 2.3.15, 2.3.16 in matrix forms, we get

$$\begin{aligned} \mathbf{P}_h^{*n}\big(\mathbf{y}(t)\big) &\approx \mathcal{A}\,\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big), && (2.3.17)\\ \mathbf{P}_h^{*n}\big(\dot{\mathbf{y}}(t)\big) &\approx \mathcal{B}\,\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big), && (2.3.18)\\ \mathbf{P}_h^{n+1}\big(\ddot{\mathbf{y}}(t)\big) &\approx \mathcal{C}\,\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big) + \mathcal{D}, && (2.3.19) \end{aligned}$$

where

$$\mathcal{A} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & & & \\ & \frac{1}{2} & \frac{1}{2} & & \\ & & \ddots & \ddots & \\ & & & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \in \mathbb{R}^{n\times(n+1)}, \qquad (2.3.20)$$

$$\mathcal{B} = \begin{bmatrix} -\frac{1}{h} & \frac{1}{h} & & & \\ & -\frac{1}{h} & \frac{1}{h} & & \\ & & \ddots & \ddots & \\ & & & -\frac{1}{h} & \frac{1}{h} \end{bmatrix} \in \mathbb{R}^{n\times(n+1)}, \qquad (2.3.21)$$

$$\mathcal{C} = \begin{bmatrix} -\frac{2}{h^2} & \frac{2}{h^2} & & & \\ \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \\ & & & \frac{2}{h^2} & -\frac{2}{h^2} \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}, \tag{2.3.22}$$

$$\mathcal{D} = \begin{bmatrix} -\frac{2}{h}\xi_0 \\ 0 \\ \vdots \\ 0 \\ \frac{2}{h}\xi_n \end{bmatrix} \in \mathbb{R}^{(n+1)\times p}. \tag{2.3.23}$$

The approximation schemes Eqs. 2.3.17, 2.3.18, 2.3.19 satisfy several interesting properties which are critical for the numerical stability. For details, please refer to the convergence proof in appendix A.

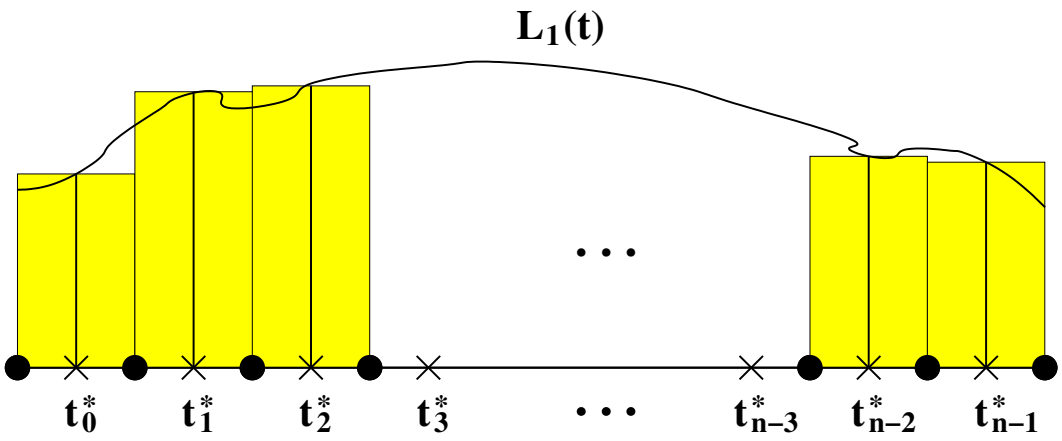


Figure 2.3: Midpoint rule

With the help of $\mathbf{P}^{*n}_{h}\big(\mathbf{y}(t)\big)$ and $\mathbf{P}^{*n}_{h}\big(\dot{\mathbf{y}}(t)\big)$, we can further approximate the integral of L_1 by using the Midpoint Rule [29] (Fig.2.3),

$$\int_{t_0}^{t_n} L_1\big(t, \mathbf{y}(t), \dot{\mathbf{y}}(t)\big)\ \mathrm{d}t \ \approx \ \sum_{k=0}^{n-1} L_1(t^*_k, \mathbf{y}^*_k, \dot{\mathbf{y}}^*_k)\, h \tag{2.3.24}$$

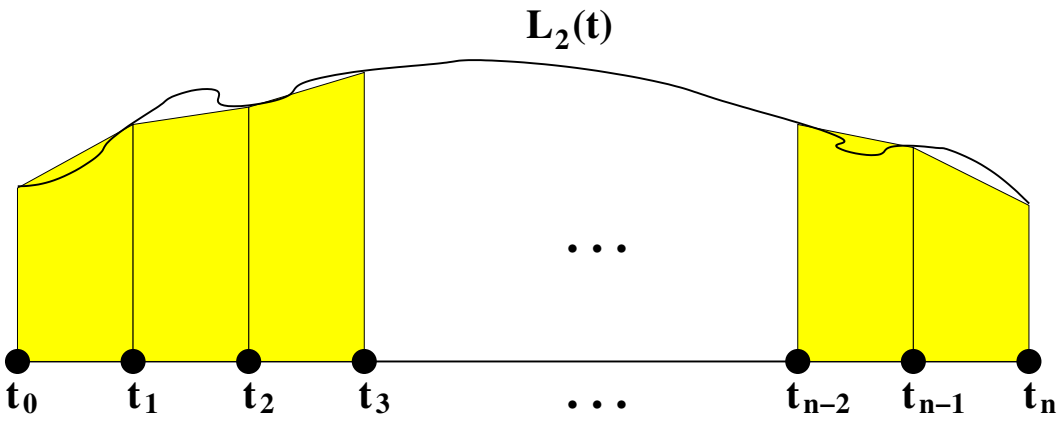


Figure 2.4: Trapezoidal rule

Also, we can apply the Trapezoidal Rule [29] (Fig.2.4) to approximate the integral of $(L_2 + \mu C)$ with $\mathbf{P}^{n+1}_{h}\big(\mathbf{y}(t)\big)$, $\mathbf{P}^{n+1}_{h}\big(\ddot{\mathbf{y}}(t)\big)$ and $\mathbf{P}^{n+1}_{h}\big(\mu(t)\big)$,

$$\begin{aligned}
&\int_{t_0}^{t_n} \Big(L_2\big(t, \mathbf{y}(t), \ddot{\mathbf{y}}(t)\big) + \mu(t) C\big(\mathbf{y}(t)\big)\Big)\ \mathrm{d}t \\
\approx\ & L_2(t_0, \mathbf{y}_0, \ddot{\mathbf{y}}_0)\frac{h}{2} + \sum_{k=1}^{n-1} \big(L_2(t_k, \mathbf{y}_k, \ddot{\mathbf{y}}_k)h\big) + L_2(t_n, \mathbf{y}_n, \ddot{\mathbf{y}}_n)\frac{h}{2} \\
& + \mu_0 C(\mathbf{y}_0)\frac{h}{2} + \sum_{k=1}^{n-1} \big(\mu_k C(\mathbf{y}_k)h\big) + \mu_n C(\mathbf{y}_n)\frac{h}{2}
\end{aligned} \tag{2.3.25}$$

By plugging the approximated first and second derivative Eqs. 2.3.17, 2.3.18, 2.3.19 into Eqs. 2.3.24 and 2.3.25, the total cost function $\tilde{J}(\mathbf{y}, \mu)$ can be expressed as a function of the discretized positions and Lagrange multipliers. Here we assume both

$\mathbf{y}_0$ and $\mathbf{y}_n$ comply with the constraint C.

$$\begin{aligned}
\tilde{J}(\mathbf{y},\mu) &= \int_{t_0}^{t_n} \Big(L_1\big(t,\mathbf{y}(t),\dot{\mathbf{y}}(t)\big) + L_2\big(t,\mathbf{y}(t),\ddot{\mathbf{y}}(t)\big) + \mu(t) C\big(\mathbf{y}(t)\big) \Big)\, \mathrm{d}t \\
&\approx \sum_{k=0}^{n-1} L_1(t_k^*,\mathbf{y}_k^*,\dot{\mathbf{y}}_k^*)\, h \\
&\quad + L_2(t_0,\mathbf{y}_0,\ddot{\mathbf{y}}_0)\,\frac{h}{2} + \sum_{k=1}^{n-1} \big(L_2(t_k,\mathbf{y}_k,\ddot{\mathbf{y}}_k)\, h \big) + L_2(t_n,\mathbf{y}_n,\ddot{\mathbf{y}}_n)\,\frac{h}{2} \\
&\quad + \mu_0\, C(\mathbf{y}_0)\,\frac{h}{2} + \sum_{k=1}^{n-1} \big(\mu_k\, C(\mathbf{y}_k)\, h \big) + \mu_n\, C(\mathbf{y}_n)\,\frac{h}{2} \\
&\approx \mathbf{L}_1\Big(\mathbf{P}^{*n}_{\ h}(t), \mathcal{A}\,\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big), \mathcal{B}\,\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big) \Big)\, h \\
&\quad + \mathbf{L}_2\Big(\mathbf{P}_h^{n+1}(t), \mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big), \mathcal{C}\,\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big) + \mathcal{D} \Big)\, h \\
&\quad + \mathbf{L}_3\Big(\mathbf{P}_h^{n-1}\big(\mathbf{y}(t)\big), \mathbf{P}_h^{n-1}\big(\mu(t)\big) \Big)\, h \\
&= \tilde{J}'(\mathbf{y}_k,\mu_k), \qquad (2.3.26)
\end{aligned}$$

where

$$\mathbf{L}_1\Big(\mathbf{P}^{*n}_{\ h}(t), \mathbf{P}^{*n}_{\ h}\big(\mathbf{y}(t)\big), \mathbf{P}^{*n}_{\ h}\big(\dot{\mathbf{y}}(t)\big) \Big) = \sum_{k=0}^{n-1} L_1(t_k^*,\mathbf{y}_k^*,\dot{\mathbf{y}}_k^*), \qquad (2.3.27)$$

$$\begin{aligned}
&\mathbf{L}_2\Big(\mathbf{P}_h^{n+1}(t), \mathbf{P}_h^{n+1}\big(\dot{\mathbf{y}}(t)\big), \mathbf{P}_h^{n+1}\big(\ddot{\mathbf{y}}(t)\big) \Big) = \\
&\frac{1}{2}\, L_2(t_0,\mathbf{y}_0,\ddot{\mathbf{y}}_0) + \sum_{k=1}^{n-1} \big(L_2(t_k,\mathbf{y}_k,\ddot{\mathbf{y}}_k) \big) + \frac{1}{2}\, L_2(t_n,\mathbf{y}_n,\ddot{\mathbf{y}}_n),
\end{aligned} \qquad (2.3.28)$$

$$\mathbf{L}_3\Big(\mathbf{P}_h^{n-1}\big(\mathbf{y}(t)\big), \mathbf{P}_h^{n-1}\big(\mu(t)\big) \Big) = \sum_{k=1}^{n-1} \mu_k C(\mathbf{y}_k). \qquad (2.3.29)$$

Based on Eq. 2.3.26, the original infinite-dimensional constrained variational problem with boundary conditions (**P2.5**) is eventually converted into a finite-dimensional optimization problem in free space (**P2.6**). The unknowns to be computed are the intermediate discretized points of the optimal curve and the Lagrange multiplier function, i.e., $\{\mathbf{y}_k, \mu_k\}$, $k = \{1, \cdots, n-1\}$.

If we let

$$\mathbf{Q} = \left[y_{1,1}\,,\,\cdots\,,\,y_{n-1,p}\,,\,\mu_1\,,\,\cdots\,,\,\mu_{n-1} \right]^\top \in \mathbb{R}^{(n-1)(p+1)}, \tag{2.3.30}$$

$$\mathbf{Q}_{\mathbf{y}} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,p} \\ \vdots & \ddots & \vdots \\ y_{n-1,1} & \cdots & y_{n-1,p} \end{bmatrix} \in \mathbb{R}^{(n-1)\times p}, \tag{2.3.31}$$

$$\mathbf{Q}_{\mu} = \begin{bmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_{n-1} \end{bmatrix} \in \mathbb{R}^{(n-1)\times(n-1)}, \tag{2.3.32}$$

where $y_{k,j}$ is the jth element of $\mathbf{y}_k$, then the optimization problem can be formulated as

P2.6 Find a vector $\mathbf{Q}$ minimizing the function $\mathbf{L}(\mathbf{Q})$, where

$$\mathbf{L}(\mathbf{Q}) := \mathbf{L}_1(\mathbf{Q}) + \mathbf{L}_2(\mathbf{Q}) + \mathbf{L}_3(\mathbf{Q}). \tag{2.3.33}$$

To solve **P2.6**, one of the most common methods is to apply Newton's iteration (**A2.1**).

A2.1 (Newton's method)

1. Pick a reasonable guess of $\mathbf{Q}$.

2. Update $\mathbf{Q}$ by the following law:

$$\mathbf{Q}_{i+1} = \mathbf{Q}_i - \mathbf{H}^{-1}_{\tilde{J}'}(\mathbf{Q}_i)\nabla\tilde{J}'(\mathbf{Q}_i), \tag{2.3.34}$$

where $\mathbf{Q}_i$ is the ith iterate of $\mathbf{Q}$, $\nabla\tilde{J}'$ is the gradient of $\tilde{J}'$ with respect to $\mathbf{Q}$, and $\mathbf{H}_{\tilde{J}'}$ is the square matrix (Hessian) of second-order partial derivatives of $\tilde{J}'$ with respect to $\mathbf{Q}$.

Keep applying step 2 until the norm of $\nabla\tilde{J}'$ gets small enough.

The result of the discretized system ($\mathbf{Q}$ such that $\nabla \tilde{J}'(\mathbf{Q}) = 0$) will converge to the result of the Euler-Lagrange boundary problem (Eq.2.3.5), as the time step size h goes to 0. Levin *et al.* [40] give the proof of the above proposition for "the simplest problem of calculus of variations". It can be shown that our approximation (Eqs.2.3.17,2.3.18,2.3.19) and integration (Eqs.2.3.24,2.3.25) schemes are 'consistent' in the sense of numerical analysis [68], therefore the proof in [40] can be extended to the constrained variational problem containing second order derivatives (i.e., **P2.5**). Please refer to Appendix A for the detailed proof.

At the moment, we require $\dot{\mathbf{y}}$ and $\ddot{\mathbf{y}}$ to be discretized under two time partition schemes (Eqs.2.3.6,2.3.11) and such a 'discrimination' on $\dot{\mathbf{y}}$ and $\ddot{\mathbf{y}}$ results in decomposing the integrand of J into $L_1\big(t, \mathbf{y}(t), \dot{\mathbf{y}}(t)\big)$ and $L_2\big(t, \mathbf{y}(t), \ddot{\mathbf{y}}(t)\big)$.

2.4 Optimal Trajectory Planning under Motion Constraints

2.4.1 Numerical Trajectory Optimization

To accomplish path optimization numerically, an usual approach in the robotics literature is to approximately represent the resulting curve by a finite number of parameters. One choice is to use the discrete joint positions, velocities, and accelerations (or drive torques), so that the overall problem can be cast as an optimal control problem [60, 64]. However, as redundancies exist among the above three physical properties (which are in fact correlated by the robot dynamic model), this representation scheme requires the optimizer to handle an extra nonlinear equality constraint and more variables.

Alternatively, we can assume the resulting curve matches a particular pattern, e.g. a cubic or B-spline [63]. Such a scheme contains much less unknowns, and the value of the curve function and its derivatives at any time instants can be calculated. However, to make the whole pre-modelled curve comply with motion constraints will be a quite difficult (or impossible) job.

Based on the thought described in the previous section, we instead represent the curve by its discrete positions only, from which reasonable approximations of velocity and acceleration can be obtained as long as we've got enough points. Namely, for the cost function $\tilde{J}(\mathbf{q}, \mu_i)$ in **Pr2.4**, we let

$$\int_{t_0}^{t_n} L_1\big(\mathbf{q}(t), \dot{\mathbf{q}}(t)\big)\,\mathrm{d}t \approx \sum_{k=0}^{n-1} L_1\Big(\frac{\mathbf{q}_k + \mathbf{q}_{k+1}}{2}, \frac{\mathbf{q}_{k+1} - \mathbf{q}_k}{h}\Big)\,h, \tag{2.4.1}$$

$$\begin{aligned}
&\int_{t_0}^{t_n} \Big(L_2\big(\mathbf{q}(t), \ddot{\mathbf{q}}(t)\big) + \sum_{i=1}^{l} \mu_i(t) C_i\big(\mathbf{q}(t)\big)\Big)\,\mathrm{d}t \\
\approx\ & L_2\Big(\mathbf{q}_0, \frac{2\mathbf{q}_1 - 2\mathbf{q}_0}{h^2} - \frac{2}{h}\xi_0\Big)\frac{h}{2} + \sum_{k=1}^{n-1} L_2\Big(\mathbf{q}_k, \frac{\mathbf{q}_{k-1} - 2\mathbf{q}_k + \mathbf{q}_{k+1}}{h^2}\Big)h \\
&+ L_2\Big(\mathbf{q}_n, \frac{2\mathbf{q}_{n-1} - 2\mathbf{q}_n}{h^2} + \frac{2}{h}\xi_n\Big)\frac{h}{2} + \sum_{i=1}^{l}\sum_{k=1}^{n-1} \mu_{i,k} C_i(\mathbf{q}_k)h.
\end{aligned} \tag{2.4.2}$$

Therefore,

$$\tilde{J}(\mathbf{q}, \mu_i) \approx \tilde{J}'(\mathbf{Q}), \tag{2.4.3}$$

in this case,

$$\tilde{J}' := \text{R.H.S. of Eq.2.4.1 + R.H.S. of Eq.2.4.2}, \tag{2.4.4}$$

$$\mathbf{Q} := \Big[\, q_{1,1}\,,\ \cdots\,,\ q_{p,n-1}\,,\ \mu_{1,1}\,,\ \cdots\,,\ \mu_{l,n-1} \,\Big]^\top. \tag{2.4.5}$$

We may try our luck to solve for $\min_{\mathbf{Q}} \tilde{J}'(\mathbf{Q})$ by applying **A2.1**. However, the classical Newton's iteration usually only enjoys local convergence properties, so a good

choice of $\mathbf{Q}$'s initial value is necessary. To make such a 'guess' more efficient, we can gradually increase the number of time partitions n, then the previously converged result can be utilized to form the initial value for the next round of the Newton's iteration.

A2.2 (Newton's method, modified)

1. Pick a small n which makes $\mathbf{Q}$ easy to converge, and pick an initial guess of $\mathbf{Q}$.

2. (*Inner loop*) Update $\mathbf{Q}$ by the following law until it converges.

$$\mathbf{Q}_{i+1} = \mathbf{Q}_i - \delta\, \mathbf{H}^{-1}_{\tilde{J}'}(\mathbf{Q}_i) \nabla \tilde{J}'(\mathbf{Q}_i) \tag{2.4.6}$$

   The step size $\delta \in (0, 1]$ has been introduced here to improve the numerical stability. δ should satisfy the Armijo condition [8], and can be calculated by the backtracking line search [9].

3. (*Outer loop*) Double n, the value of newly introduced elements is initially set as the mean of their calculated left and right neighbors. With the expanded vector $\mathbf{Q}$, go to step 2.

Extension: Besides equality constraints such as Eq.2.2.1, robots are usually subject to inequality constraints as well, e.g., joint position/velocity limit or obstacle avoidance, which can eventually be converted into a bunch of constraints on the discrete joint positions $C'_i(\mathbf{Q}) < 0, i = l + 1 \cdots m$ by using approximation schemes.

To deal with the inequalities, we propose to use the interior point (IP) method [9], which relaxes the constraints $C'_i < 0$ by inserting a convex, smooth 'barrier' function ϕ in the cost function $\tilde{J}'$. An example for ϕ is given as below. In practical

implementations, the initial value of $\mathbf{Q}$ should be picked to satisfy all of the inequality constraints.

$$\phi(\mathbf{Q}) = \begin{cases} -\sum_{i=l+1}^{m} \log\big(-C_i'(\mathbf{Q})\big) & \text{all } C_i'(\mathbf{Q}) < 0 \\ +\infty & \text{otherwise} \end{cases} \tag{2.4.7}$$

Apart from applying the above IP method, an easier way to avoid violating the joint velocity constraint is to linearly scale the execution time. Similar time scaling methods can sometimes also be employed for drive torque limit (e.g., the dynamic scaling technique [56]).

2.4.2 Interpolation

After having computed the intermediate discretized points, the next step is to find a sufficiently smooth joint trajectory $\mathbf{q}(t)$ such that

$$\mathbf{q}(t_k) = \mathbf{q}_k, \tag{2.4.8}$$

$$C_i\big(\mathbf{q}(t)\big) = 0, \ \ \forall t \in [t_0, t_n]. \tag{2.4.9}$$

To ensure the resulting joint curve satisfies both two conditions (Eqs.2.4.8,2.4.9), we design a 3-step calculation scheme (**A2.3**) as follows:

A2.3 (Interpolation)

1. Interpolate the previously computed discretized points $\{\mathbf{q}_0, \mathbf{q}_1 \cdots \mathbf{q}_{n-1}, \mathbf{q}_n\}$ by a cubic spline called $\mathbf{q}_{org}$.

2. Interpolate $\mathbf{p}_k = \kappa_{\mathbf{p}}(\mathbf{q}_k)$ and $\mathbf{R}_k = \kappa_{\mathbf{R}}(\mathbf{q}_k)$ by curves $\mathbf{p}$ and $\mathbf{R}$ with the motion constraints C_i satisfied, or directly use the pre-determined $\mathbf{p}_d, \mathbf{R}_d$ in Eq.2.2.3 if available.

3. Adjust the joint trajectory $\mathbf{q}_{org}$ to fit the end-effector path $\mathbf{p}, \mathbf{R}$ by repeatedly applying the following law. Set $\mathbf{q}_{old}(t) = \mathbf{q}_{org}(t)$ initially.

$$\mathbf{q}_{new}(t) = \mathbf{q}_{old}(t) + \mathbf{J}'\big(\mathbf{q}_{old}(t)\big)^{\dagger} \begin{bmatrix} \Big(\log\Big(\kappa_{\mathbf{R}}\big(\mathbf{q}_{old}(t)\big)^{\top}\mathbf{R}(t)\Big)\Big)^{\wedge -1} \\ \mathbf{p}(t) - \kappa_{\mathbf{p}}\big(\mathbf{q}_{old}(t)\big) \end{bmatrix}, \tag{2.4.10}$$

where

   (a) $\dagger$ denotes the pseudo-inverse operation;

   (b) log denotes the matrix logarithm, uniquely defined in an open neighborhood of the identity matrix;

   (c) $\mathbf{J}'(t) : \mathbb{R}^p \to \mathbb{R}^6$, $\dot{\mathbf{q}}(t) \mapsto \begin{bmatrix} \omega(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix}$ can be regarded as $\mathbf{J}(t)$ (the Jacobian) transformed into another coordinate system, here $\mathbf{I}_3$ is the 3×3 identity matrix;

$$\mathbf{J}'(t) = \begin{bmatrix} \mathbf{I}_3 & 0 \\ \widehat{\mathbf{p}}(t)^{\top} & \mathbf{I}_3 \end{bmatrix} \mathbf{J}(t) \tag{2.4.11}$$

   (d) $\widehat{\ }: \mathbb{R}^3 \to \mathfrak{so}_3$, $\mathbf{x} \mapsto \widehat{\mathbf{x}}$ is as

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \mapsto \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{2.4.12}$$

   with obvious inverse $^{\wedge -1} : \mathfrak{so}_3 \to \mathbb{R}^3$, $\widehat{\mathbf{x}} \mapsto \big(\widehat{\mathbf{x}}\big)^{\wedge -1} = \mathbf{x}$.

   (e) the upper element in the square bracket is the rotation vector in *canonical* coordinates extracted from two rotation matrices, whose closed form has been revealed by *Rodrigues' Formula* [51].

Sometimes only the tracking of $\mathbf{p}$ is feasible (or necessary) due to the robot's kinematic constraints (or the demand of motion task). A pull back/push forward technique with

rolling and wrapping for smooth interpolating curves on a manifold was proposed recently [31] and it can be used to compute $\mathbf{p}$. One remarkable feature of this technique is that it is a coordinate-free approach. Extension of this particular smooth interpolation approach to orientation (namely SO_3) has been done, and the next section will present the details of that work.

Eq.2.4.10 is a Newton-Raphson-like root-finding algorithm. Using the pseudo-inverse of the Jacobian is to fine-tune the robot pose so that the end-effector configuration will get closer to the desired configuration $\{\mathbf{p}(t), \mathbf{R}(t)\}$ with minimum change in joint position, i.e., $\triangle\mathbf{q}(t) = \mathbf{q}_{new}(t) - \mathbf{q}_{old}(t)$. Here we assume we've already have enough discretized points, in other words, $\{\mathbf{p}(t), \mathbf{R}(t)\}$ and $\{\kappa_{\mathbf{p}}\big(\mathbf{q}_{org}(t)\big), \kappa_{\mathbf{R}}\big(\mathbf{q}_{org}(t)\big)\}$ are sufficiently close to each other.

2.5 Examples

Example 2.1 (Cubic spline)

Find a C^4-smooth curve $y : [0, 1] \to \mathbb{R}$, such that the integral of the square of its acceleration $\mathrm{d}^2y(t)/\mathrm{d}t^2$ is at stationary value,

$$\begin{aligned} &\min_{y(t)} \int_0^1 \ddot{y}(t)^2 \, \mathrm{d}t, \\ s.t. \quad &y(0) = y_0,\, y(1) = y_1,\, \dot{y}(0) = \xi_0,\, \dot{y}(1) = \xi_1. \end{aligned} \tag{2.5.1}$$

Now derive the Euler-Lagrange equation for the above problem, here ε, z are defined as in Section 2.1.1 on page 10.

$$\begin{aligned} & \frac{\mathrm{d}}{\mathrm{d}\varepsilon} \int_0^1 \big(\ddot{y}(t) + \varepsilon\ddot{z}(t)\big)\big(\ddot{y}(t) + \varepsilon\ddot{z}(t)\big) \, \mathrm{d}t \Big|_{\varepsilon=0} \\ = & \int_0^1 2\big(\ddot{y}(t) + \varepsilon\ddot{z}(t)\big)\ddot{z}(t) \, \mathrm{d}t \Big|_{\varepsilon=0} \\ = & \, 2\int_0^1 \ddot{y}(t)\ddot{z}(t) \, \mathrm{d}t \end{aligned}$$

$$
\begin{aligned}
&= 2\Big(\ddot{y}(t)\dot{z}(t)\Big|_{t=0}^{t=1} - \int_0^1 \dot{z}(t)\dddot{y}(t)\,\mathrm{d}t\Big) \\
&= 2\Big(\ddot{y}(t)\dot{z}(t)\Big|_{t=0}^{t=1} - \dddot{y}(t)z(t)\Big|_{t=0}^{t=1} + \int_0^1 z(t)\ddddot{y}(t)\,\mathrm{d}t\Big) \\
&= 0. \qquad (2.5.2)
\end{aligned}
$$

The first and second term in the bracket will disappear because $z(t_0) = z(t_n) = 0$, $\dot{z}(t_0) = \dot{z}(t_n) = 0$. Also, as z can be any arbitrary smooth curve as long as it satisfies the boundary conditions, the only possibility for the whole equation to be zero is that $\ddddot{y}(t)$ equals 0, which leads to the Euler-Lagrange equation (Eq. 2.5.3),

$$\ddddot{y}(t) = 0 \qquad (2.5.3)$$

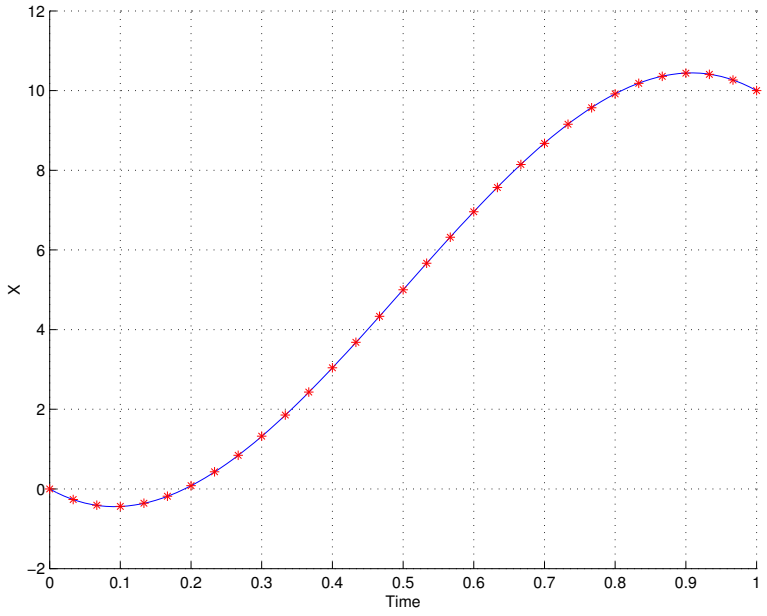


Figure 2.5: Blue: cubic spline; Red: computed discrete points

Therefore, the solution of the above problem is a cubic spline $y(t) = at^3 + bt^2 + ct + d$, whose coefficients are uniquely determined by the boundary conditions, i.e.,

by solving a linear system.

Fig. 2.5 shows the points computed through our algorithm in contrast with the real curve (here $y_0 = 0$, $y_1 = 10$, $\xi_0 = -10$, $\xi_1 = -10$). Their difference is not obvious, as in this example the error of y_k is within 0.1% of the real cubic spline function $y(t)$.

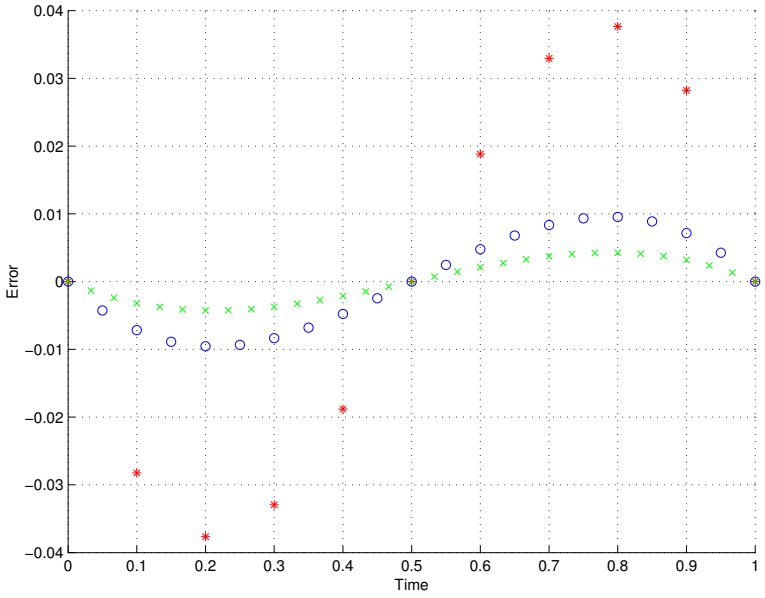


Figure 2.6: Numerical error for situations with different numbers of points: Red, 9; Blue, 19; Green: 29

Fig. 2.6 shows the numerical error for different numbers of intermediate discretized points. We can see that more points will give rise to better accuracy, such a phenomenon complies with our convergence result proved in Appendix A.

Example 2.2 (Curve with minimum tangential acceleration on 2-sphere)

Find a curve $\mathbf{y} : [0, 1] \to S^2$, such that the integral over the squared norm of its

acceleration projected to the tangent plane attains a minimum,

$$\begin{aligned} &\min_{\mathbf{y}(t)} \int_0^1 \ddot{\mathbf{y}}(t)^T \left(I - \frac{\mathbf{y}\mathbf{y}^T}{\mathbf{y}^T\mathbf{y}}\right) \ddot{\mathbf{y}}(t)\ \mathrm{d}t, \\ &s.t.\ \mathbf{y}(t)^T\mathbf{y}(t) = 1\ \forall t \in [0,1],\ \mathbf{y}(0) = \mathbf{y}_0,\ \mathbf{y}(1) = \mathbf{y}_1,\ \dot{\mathbf{y}}(0) = \xi_0,\ \dot{\mathbf{y}}(1) = \xi_1. \end{aligned} \tag{2.5.4}$$

The Euler-Lagrange equation of the above problem is implicit (for which an explicit formula has been derived in [17]) but no closed-form solution has been derived yet. We apply our algorithm with initial value of equal-distance points along the geodesics, the result is illustrated in Fig. 2.7 (here $\mathbf{y}_0$ = [0 0 1], $\mathbf{y}_1$ = [0 1 0], ξ_0 = [2 -2 0], ξ_1 = [1 0 -1]). This is the similar problem that Crouch *et al.* considered in [17]. Their approach is different from ours and the comparison between both are not yet considered.

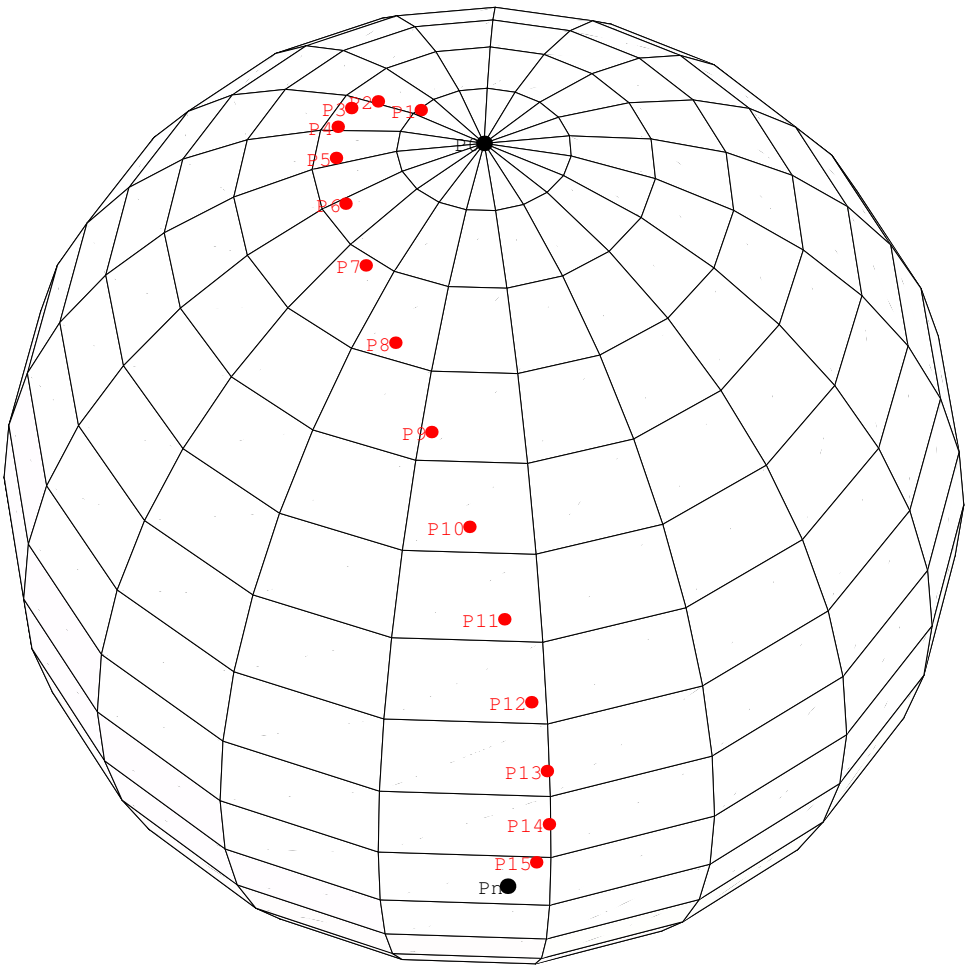


Figure 2.7: Computed intermediate discretized points on the sphere.

Example 2.3 (Trajectory optimization for WAM performing a compliant motion task)

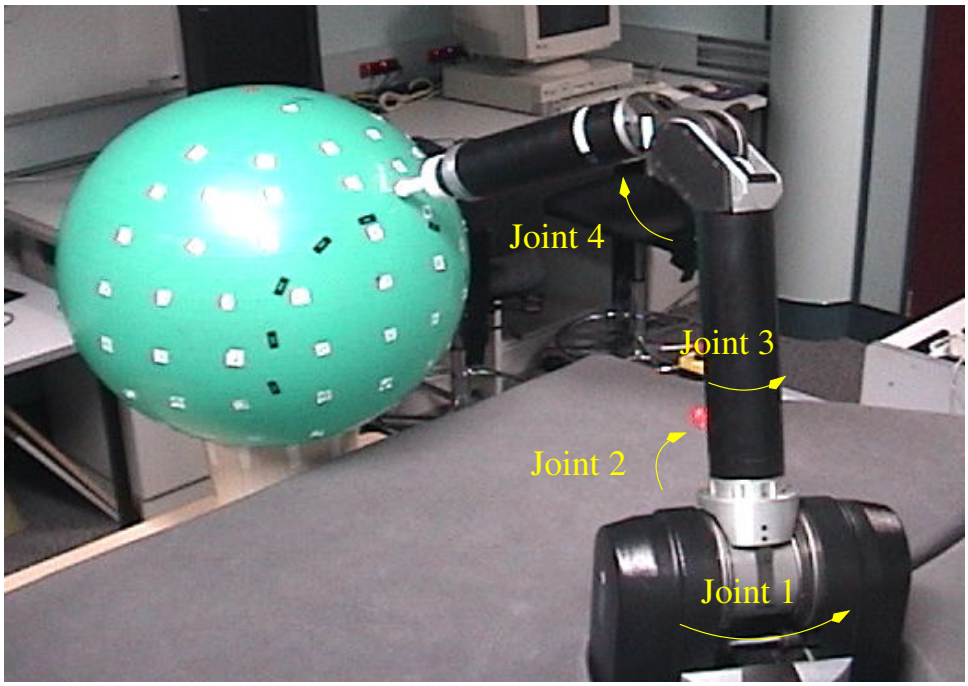


Figure 2.8: 4-DOF robot manipulator WAM

WAM is a 4-joint robot manipulator with human-like kinematics (see Fig.2.8). Viewed in the Cartesian space, WAM's 4-DOFs correspond to its end-effector moving to an arbitrary 3D position within its reachable space and rotating about vector $\mathbf{p}(t)$ (1-DOF in orientation).

Now suppose the robot is required to move its end-effector on some working surface, for instance, a sphere (see Fig.2.9). Given the initial and final joint positions, now the challenge is to find a joint trajectory yielding a comprehensive optimality with respect to the following four aspects:

1. WAM's last link is as perpendicular to the spherical surface as possible, i.e., θ

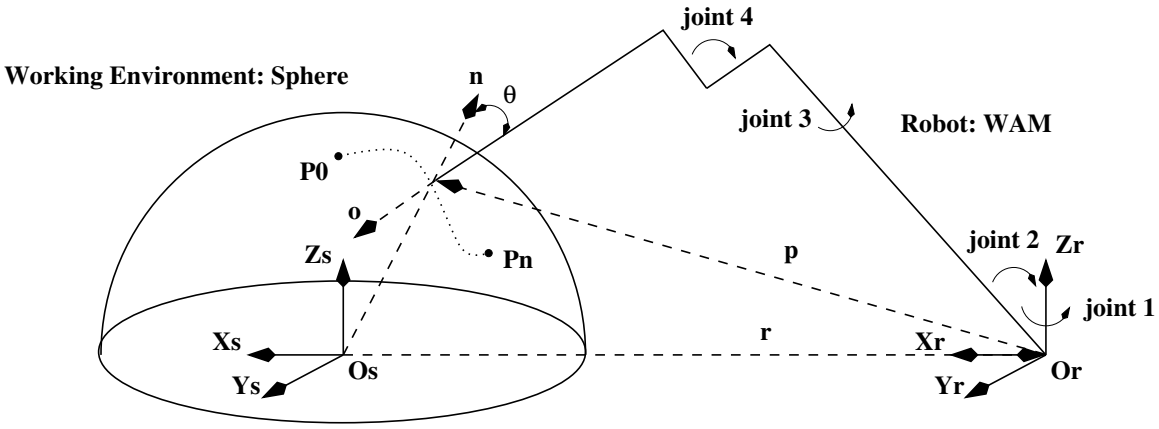


Figure 2.9: WAM moves its end-effector on a sphere

should be minimal;

2. The end-effector traverses minimum distance;

3. The joints traverse minimum distance;

4. The joints yield minimum curvature.

In addition, the joint position limit must be satisfied throughout the movement.

Summarily, the overall path planning problem for WAM can be described as follows.

Pr2.6 Find a sufficiently smooth 4-dim joint curve $\mathbf{q}$ subject to

$$\mathbf{q}_{min} \leq \mathbf{q}(t) \leq \mathbf{q}_{max},\ \forall t \in [t_0, t_n], \tag{2.5.5}$$

$$\|\mathbf{r} + \mathbf{p}(t)\| = R,\ \forall t \in [t_0, t_n], \tag{2.5.6}$$

$$\begin{aligned} \mathbf{q}(t_0) = \mathbf{q}_0,\ \mathbf{q}(t_n) = \mathbf{q}_n, \\ \dot{\mathbf{q}}(t_0) = \xi_0,\ \dot{\mathbf{q}}(t_n) = \xi_n, \end{aligned} \tag{2.5.7}$$

such that the following cost function J has a minimal value,

$$J = \int_{t_0}^{t_n} \Big(\cos(\pi - \theta) + \alpha\, \dot{\mathbf{p}}^\top \dot{\mathbf{p}} + \beta\, \dot{\mathbf{q}}^\top \dot{\mathbf{q}} + \gamma\, \ddot{\mathbf{q}}^\top \ddot{\mathbf{q}} \Big)\, \mathrm{d}t, \tag{2.5.8}$$

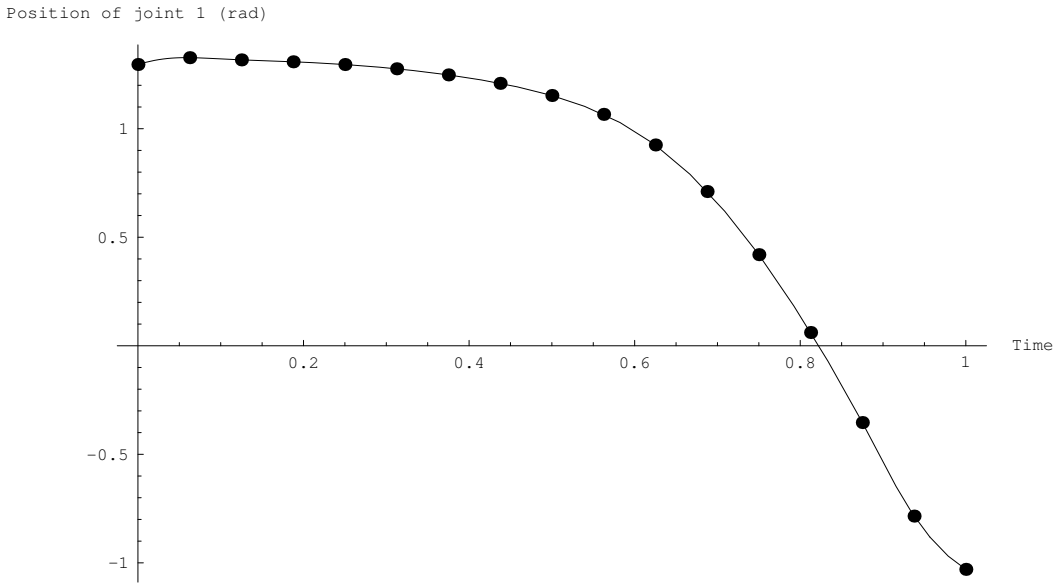
$$\cos\big(\pi - \theta(t)\big) = \frac{\mathbf{o}(t)^\top (\mathbf{r} + \mathbf{p}(t))}{\|\mathbf{o}(t)\| \|(\mathbf{r} + \mathbf{p}(t))\|}, \tag{2.5.9}$$

where $\mathbf{o}(t) \in \mathbb{R}^3$ is the z-axis of link 4's local coordinate viewed in the inertial frame $\{\mathbf{o_r}, \mathbf{x_r}, \mathbf{y_r}, \mathbf{z_r}\}$.

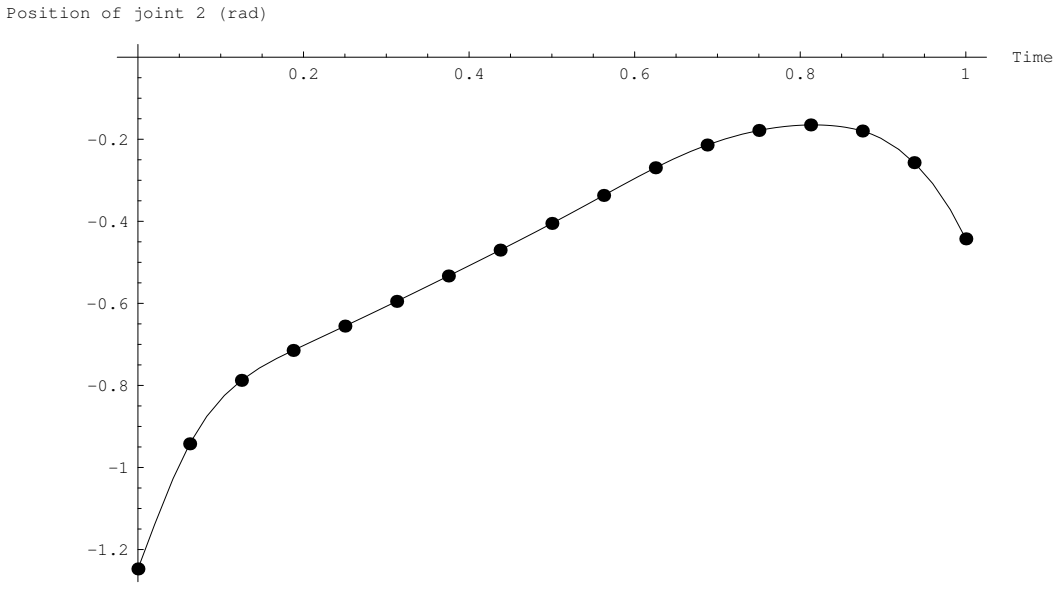
$$\mathbf{o}(t) = \mathbf{R}(t) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \kappa_{\mathbf{R}}\big(\mathbf{q}(t)\big) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.5.10}$$

$\mathbf{r} \in \mathbb{R}^3$ is the vector from the center of sphere to the base of WAM, and $\mathbf{o}(t)$, $\mathbf{p}(t)$, $\mathbf{r}$ are all as Fig.2.9 illustrates. α, β and γ are the weights of their corresponding terms, their values can be chosen according to customer's preference. R is the radius of the sphere, and $\|\mathbf{x}\| = \sqrt{\mathbf{x}^\top \mathbf{x}}$.

Figs.2.10,2.11 show the result of implementing our path planning method for P6, the second step of interpolation is only carried out for $\mathbf{p}$ by using the pull back/push forward technique with rolling and wrapping. Here $\alpha = 0.5$, $\beta = 2.5 \times 10^{-4}$, $\gamma = 5 \times 10^{-7}$, $R = 0.3$, $\mathbf{r} = [-0.6, 0, 0]$, all data about WAM's kinematic model including $\mathbf{q}_{min,max}$ are as in [6]. WAM's initial and final poses are arbitrarily chosen, but they both satisfy the motion constraints. Fig.2.12 illustrates the resulting end-effector path, from which we can see the equality constraint (Eq.2.5.6) has been fulfilled throughout the entire movement. The black and red straight lines in Fig.2.12 are respectively the sphere normal and direction of WAM's last link, and the lower curve are the only places where WAM's last link can be perpendicular to the spherical surface, so the path of end-effector looks like to be 'attracted' towards that region.

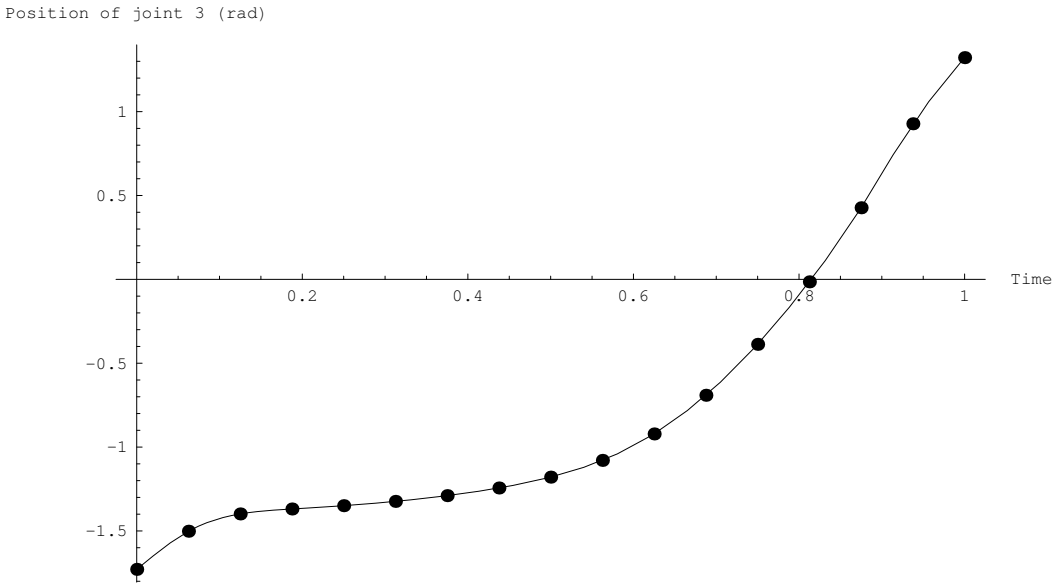


(a)

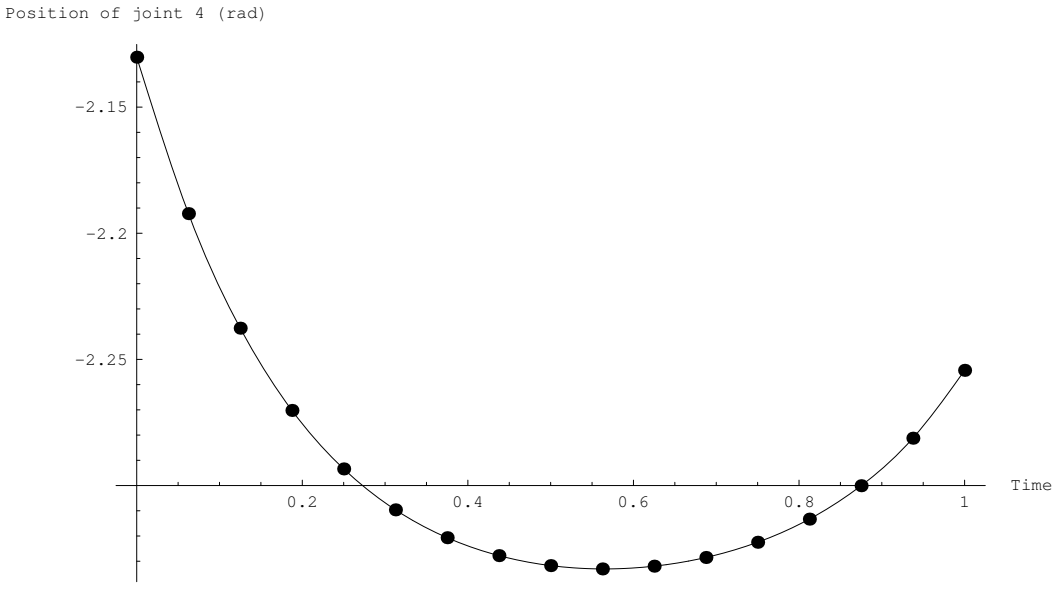


(b)

Figure 2.10: Discretized points and interpolating joint trajectory: a) joint 1; b) joint 2



(a)



(b)

Figure 2.11: Discretized points and interpolating joint trajectory: a) joint 3; b) joint 4

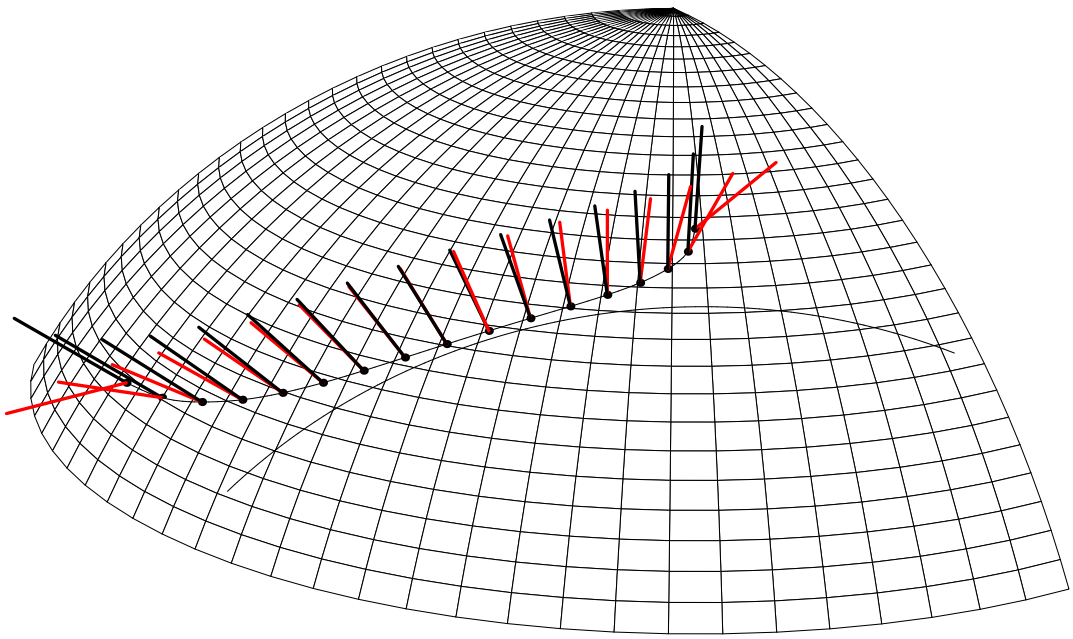


Figure 2.12: End-effector path calculated from joint trajectory in Figs.2.10, 2.11. Black: sphere normal, red: WAM's last link

The attached CD contains a Java 3D visualization demo (*3dAnimation.mpg*) and a motion control experiment video (*Experiment.mpg*), implementing our path planning algorithm on the real robot manipulator WAM. The experiment's setup is exactly as in Fig.2.8.

2.6 Summary and Future Work

In this chapter, we present a two-step path planning approach for numerically calculating a joint trajectory for a manipulator subject to motion constraints yielding some synthetical geometrical optimality. This calculation scheme is based on an iterative algorithm solving for variational problems. A convergence proof is given in Appendix A which ensures that the curve calculated from our algorithm will converge to the resulting curve of the Euler-Lagrange equation as we let the time step in the discretization scheme go to zero. Some examples including a motion planning exercise for a 4-DOF robot WAM performing a compliant motion task has been given at the end of this chapter. It shows our method can generate quite satisfactory results for an actual robotic system with a fairly complicated optimization objective.

The current time partition schemes (Eqs.2.3.6,2.3.11) for $\ddot{\mathbf{q}}$ and $\dot{\mathbf{q}}$ result in the integrand of J under the form of $L_1\big(\mathbf{q}(t), \dot{\mathbf{q}}(t)\big) + L_2\big(\mathbf{q}(t), \ddot{\mathbf{q}}(t)\big)$. The limitation of such a decomposition is its inability of representing any optimization objective simultaneously relevant to $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$. Therefore, it will be worthwhile to find 'consistent' [68] approximation schemes for $\dot{\mathbf{q}}(t_k)$ and $\ddot{\mathbf{q}}(t_k)$ discretized under a unique time partition scheme. Then the scope of our path optimization can be extended to some dynamic properties such as joint drive torque or end-effector acceleration.

Moreover, we are interested in extending our trajectory-optimization algorithm to robotic systems subject to non-holomonic constraints (i.e., those constraints contain first-order derivatives and are not integrable). Some typical applications are for example, docking of mobile robots, path planning for aircrafts, and so on.

Chapter 3

Smooth Interpolation of Orientation by Rolling and Wrapping for Robot Motion Planning

This chapter investigates a novel procedure to calculate smooth interpolation curves of the rotation group SO_3, which is commonly considered as the standard representation of rigid-body's orientations. The algorithm is a combination of rolling and wrapping with the pull back/push forward technique. Remarkable advantages of this approach include 1) it is coordinate-free; and 2) interpolation curves will be given in closed forms, which brings convenience for implementations on real-time control systems. A numerical example along with some visualization results is presented as well at the end.

3.1 Background and Literature Review

Smooth interpolation in Euclidean spaces has many applications in robot motion planning (e.g., interpolation of end-effector's linear position, or of joint traversing

points, for free motion tasks), and has been well studied ever since the outset of robotics research. The usual approach for solving this category of problems is to apply cubic splines or even higher order polynomials, whose coefficients can be calculated through a linear system [15, 56].

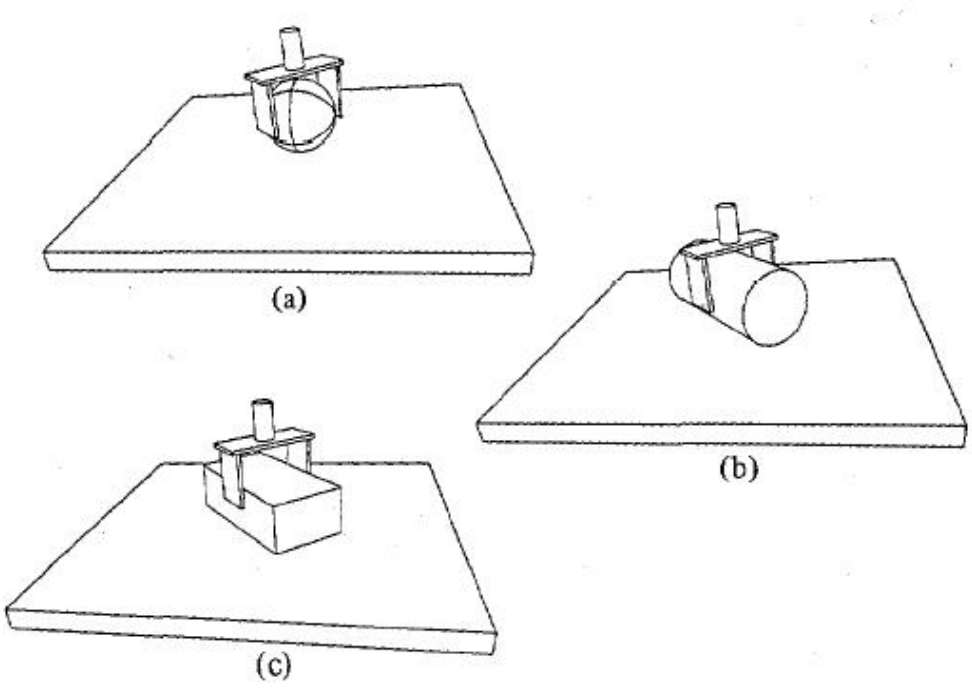


Figure 3.1: Constrained motion of robot's end-effector: a) 5-DOFs, b) 4-DOFs, c) 3-DOFs (courtesy of O. Khatib)

On the other hand, smooth interpolation in non-Euclidean spaces has comparatively attracted less roboticists' attentions, although it is in fact an interesting theoretical problem with applications to path planning for mechanical systems whose configuration spaces contain components which are Lie groups or symmetric spaces. For instance, robot manipulators' end-effectors are subject to motion constraints due

to given tasks as illustrated in Fig.3.1. In contrast to the free motion situation (i.e., $\mathcal{E} = \mathbb{R}^3 \times SO_3$), the configuration spaces of the gripper in Fig.3.1a, Fig.3.1b, Fig.3.1c, and of WAM's 'pawn' end-effector in example 2.3, are $\mathbb{R}^2 \times SO_3$, $\mathbb{R}^2 \times S^2$, $\mathbb{R}^2 \times S^1$, and $S^2 \times SO_3$ respectively.

Recall the robot motion planning scheme in the previous chapter, **A2.3** involves a preliminary step of interpolating both $\mathbf{p}$ (linear position) and $\mathbf{R}$ (orientation), which hasn't been talked too much about at that time. Actually, we only performed the interpolation of $\mathbf{p}$ on S^2 in the example 2.3, since WAM has less than 6 degrees of freedom anyway. Nevertheless, smooth interpolation of end-effector's orientation is worthwhile for general robots. With respect to those orientations subject to motion constrains as in Fig.3.1b and Fig.3.1c, their corresponding configuration spaces are just 1- or 2-sphere whose interpolation can be treated similarly as with the linear position in the example 2.3.

However, interpolation on the entire rotation group SO_3 is much more difficult, on the other hand it is particularly useful in robotics (e.g., motion planning of rigid body); computer graphics (e.g., animation of 3D objects); satellite attitude control; and so on. Although being a smooth submanifold embedded in $\mathbb{R}^{3\times3}$, the Special Orthogonal group SO_3 is not a simple geometric object. It brings difficulties in directly engrafting smooth interpolation algorithms on orientation, and this chapter will deal with extending the interpolation technique used in the example 2.3 to SO_3.

As far as we notice, most of the existing literature on SO_3 curve design can be classified into two major methodologies:

1. extension of the De Casteljau algorithm;

2. coordinate parametrization.

The first class of work focuses on generalization of Bézier curves for Lie groups. Park and Ravani show how the De Casteljau algorithm can be extended to Riemannian manifolds, and the mathematical elegance of Lie groups is utilized for constructing Bézier curves efficiently [52]. Later Crouch *et al.* suggest a modified De Casteljau algorithm to generate cubic splines on connected and compact Lie groups. Details for SO_3 appear in [16].

However, De Casteljau-like algorithms are in general computationally expensive and quite cumbersome for interpolation applications [36]. More recent research has adopted a different approach for SO_3's interpolation, which first re-parameterizes rotation matrices (by rotation axes and angles, for instance), then performs cubic spline interpolation based on such representations. Kang and Park's paper collects a few popular SO_3 interpolation algorithms of this class, also the trajectory distortion caused by various coordinate parametrization schemes has been comparatively studied and discussed [36].

To eliminate the distortion of interpolation curves introduced by local diffeomorphisms (i.e., by the process of simple pull back/push forward [46]), Hüper and Silva Leite recently proposed a novel interpolation method combining pull back/push forward with rolling and wrapping on smooth manifolds, in particular on S^2 [31]. Although many of the ideas in [31] can be directly applied here, contrary to the two-sphere situation, as we said, the geometric intuition of SO_3 is less obvious. Building on [32], the scientific contribution of the work described in this chapter is to specialize and examplify the previous research to SO_3.

The rest of the chapter is organized as follows: Section 2 gives the mathematical

formulation of our SO_3 interpolation problem; Section 3 introduces a few relevant differential geometry concepts and then presents the major algorithm which calculates an interpolation curve of SO_3, given in an explicit form; Section 4 shows a numerical example of interpolating three orientations, and some visualization results are attached at the end of this chapter.

3.2 Problem Description

As we know, SO_3 can be considered as a smooth 3-dim submanifold of $\mathbb{R}^{3\times3}$. Therefore, for all $\mathbf{R} \in SO_3$, the affine tangent space $T^{\text{aff}}_{\mathbf{R}}SO_3$ can be considered as an affine subspace of $\mathbb{R}^{3\times3}$ (see Fig.3.2).

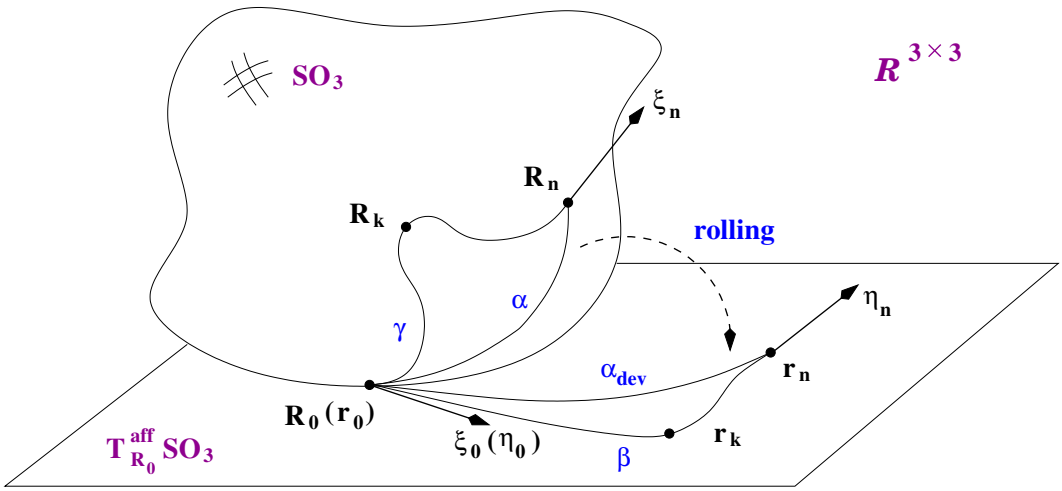


Figure 3.2: Geometry of manifold SO_3, affine tangent space $T^{\text{aff}}_{\mathbf{R}_0}SO_3$ at $\mathbf{R}_0$, and the embedding Euclidean space $\mathbb{R}^{3\times3}$

P3.1 Now, we wish to find a C^2-smooth curve $\gamma : [0, T] \to SO_3$ interpolating a set

of given intermediate points $\mathbf{R}_k \in SO_3$, such that

$$\gamma(t_k) = \mathbf{R}_k,\ 1 \leq k \leq n-1, \tag{3.2.1}$$

where

$$0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = T, \tag{3.2.2}$$

in addition, subject to the boundary conditions

$$\begin{aligned} \gamma(0) &= \mathbf{R}_0 \in SO_3, \\ \gamma(T) &= \mathbf{R}_n \in SO_3, \\ \dot{\gamma}(0) &= \xi_0 \in T_{\mathbf{R}_0} SO_3, \\ \dot{\gamma}(T) &= \xi_n \in T_{\mathbf{R}_n} SO_3. \end{aligned} \tag{3.2.3}$$

3.3 Interpolation of SO_3 by Rolling and Wrapping

This section starts with a brief review of the pull back/push forward technique, for the particular manifold SO_3. Next, the key component in our calculation scheme, the rolling map, will be introduced and explained in detail. In the last part of this section, the major smooth interpolation algorithm is presented in steps and the closed form of the resulting C^2-smooth curve on SO_3 is given at the end.

3.3.1 Local Diffeomorphism

Pull back/push forward is one possible approach for smooth interpolation on manifolds [46]. The basic idea is to project data points from a manifold M onto its affine tangent space (at a point $\mathbf{p}_0 \in M$), hereafter denoted by V; then perform smooth

interpolation there; finally project the interpolation curve back onto M. Most of the algorithms recorded in [36] can be classified into this category, although the concept of the affine tangent space has not been explicitly raised in that paper.

Several pull back (i.e., projection from M to V) and push forward maps (i.e., inverse of pull back) have been established for SO_3 already. The most noticeable one is probably the exponential map, for simplicity, considered here only at the identity. Let $\mathcal{U} \subset SO_3$ such that

$$\mathcal{U} := \{\mathbf{R} \in SO_3 | \mathbf{R} \notin \{ \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \}\} \tag{3.3.1}$$

then

$$\begin{aligned} \phi : \mathcal{U} &\to T^{\text{aff}}_{\mathbf{I}_3} SO_3, \\ \mathbf{R} &\mapsto \mathbf{I}_3 + \log(\mathbf{R}), \\ \phi^{-1} : T^{\text{aff}}_{\mathbf{I}_3} SO_3 &\to \mathcal{U}, \\ \mathbf{r} &\mapsto \mathrm{e}^{(\mathbf{r}-\mathbf{I}_3)}, \end{aligned} \tag{3.3.2}$$

defines a diffeomorphism. Here for all $\mathbf{R} \in \mathcal{U}$,

$$\log(\mathbf{R}) = \begin{cases} \frac{\cos^{-1}\left(\frac{\mathrm{tr}(\mathbf{R})-1}{2}\right)}{2\sin\left(\cos^{-1}\left(\frac{\mathrm{tr}(\mathbf{R})-1}{2}\right)\right)} (\mathbf{R}-\mathbf{R}^\top), & \mathbf{R} \neq \mathbf{I}_3 \\ \mathbf{O}_3, & \mathbf{R} = \mathbf{I}_3, \end{cases}$$

where $\mathbf{I}_3$ is the 3×3 identity matrix, $\mathbf{O}_3$ is the 3×3 zero matrix, and $\mathrm{tr}(\mathbf{R}) := \sum_{i=1}^{3} R_{ii}$; and for all $\Omega = -\Omega^\top$,

$$\mathrm{e}^{\Omega} = \begin{cases} \mathbf{I}_3 + \frac{\Omega}{\|\Omega\|}\sin(\|\Omega\|) + \frac{\Omega^2}{\|\Omega\|^2}\big(1-\cos(\|\Omega\|)\big), & \|\Omega\| \neq 0 \\ \mathbf{I}_3, & \|\Omega\| = 0, \end{cases}$$

where $\|\Omega\| = \| \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \| := \sqrt{x^2+y^2+z^2}$.

An alternative diffeomorphism, being attractive due to its simple implementation, is related to the QR-decomposition or the Gram-Schmidt(GS) orthonormalization process of a square matrix. If the matrix to be QR-decomposed has positive determinant, the decomposition into a product of a special orthogonal matrix and an upper-triangular matrix can be made unique by requiring the diagonal entries of the upper-triangular matrix to be positive. Moreover, in this case, this decomposition can be shown to be a local diffeomorphism as well. Again, for simplicity, we only present the formulas at the identity. Let

$$\begin{aligned} SO_3 \supset \mathcal{Q} := \{\mathbf{R} \in SO_3 \,|\, \mathbf{R} = \mathrm{e}^{t\Omega},\, t \in [0, \frac{\pi}{2}),\\ \Omega = -\Omega^\top,\, \|\Omega\| = 1\}, \end{aligned} \tag{3.3.3}$$

then

$$\begin{aligned} {\phi_{GS}}^{-1} : T^{\mathrm{aff}}_{\mathbf{I}_3} SO_3 \to \mathcal{Q},\\ \mathbf{I}_3 + \Omega \mapsto (\mathbf{I}_3 + \Omega)\mathbf{U}^{-1}, \end{aligned} \tag{3.3.4}$$

where $\mathbf{I}_3 + \Omega = \mathbf{QU}$ is the unique decomposition of $\mathbf{I}_3 + \Omega$ into a special orthogonal matrix $\mathbf{Q}$ and an upper-triangular matrix $\mathbf{U}$ with positive diagonal entries. Note that $\det(\mathbf{I}_3 + \Omega) > 0$ holds for all $\Omega = -\Omega^\top$ and therefore $\det(\mathbf{Q}) = 1$ always. Moreover,

$$\begin{aligned} \phi_{GS} : \mathcal{Q} \to T^{\mathrm{aff}}_{\mathbf{I}_3} SO_3,\\ \mathbf{R} \mapsto \phi_{GS}(\mathbf{R}) = \mathbf{I}_3 + \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}, \end{aligned} \tag{3.3.5}$$

where x, y, z can be uniquely computed by

$$\begin{aligned} x &= \frac{R_{32}(R_{11}^2 + R_{21}^2) - R_{31}(R_{11}R_{12} + R_{21}R_{22})}{R_{11}(R_{11}R_{22} - R_{12}R_{21})} \\ &= \frac{R_{32}(1 - R_{31}^2) + R_{31}^2 R_{32}}{R_{11}(R_{11}R_{22} - R_{12}R_{21})} \\ &= \frac{R_{32}}{R_{11}(R_{11}R_{22} - R_{12}R_{21})}, \\ y &= -\frac{R_{31}}{R_{11}}, \\ z &= \frac{R_{21}}{R_{11}}. \end{aligned} \tag{3.3.6}$$

The derivation of x utilizes the properties of special orthogonal matrices, i.e., $R_{11}^2 + R_{21}^2 + R_{31}^2 = 1$ and $R_{11}R_{12} + R_{21}R_{22} + R_{31}R_{32} = 0$.

As we know, the QR-decomposition is less computationally intensive and numerically more stable than the exponential map. It can be shown that for any matrix $\mathbf{R} \in \mathcal{Q}$, $R_{11} > 0$ and $R_{11}R_{22} - R_{12}R_{21} \neq 0$ hold always.

3.3.2 Rolling Map

Compared with De Casteljau-like algorithms, interpolation schemes based on local diffeomorphisms are easier to implement. One important reason is due to the fact that the former requires the solution of nonlinear implicit matrix equations. In order to make interpolation processing coordinate-free, a new thought of combining a rolling map with the simple pull back/push forward technique was recently proposed [31]. The novelty is that the manifold will be firstly rolled (without slipping or twisting) as a rigid body, then the given data including positions and velocities are unwrapped onto the affine tangent space.

Next we will introduce the concept of rolling maps and present the formulation

for rolling SO_3 in detail.

As we know, rigid body motion in $\mathbb{R}^n$ can generally be described as the usual action of the Euclidean group

$$SE_n = SO_n \ltimes \mathbb{R}^n \tag{3.3.7}$$

acting on $\mathbb{R}^n$. Here, as usual, the symbol $\ltimes$ denotes the semi-direct product of the groups $(SO_n, (\,\cdot\,))$ and $(\mathbb{R}^n, (+))$. Therefore, since the embedding Euclidean space for SO_3 is $\mathbb{R}^{3\times 3} \cong \mathbb{R}^9$, any rigid body motion of SO_3 is given by a curve in SE_9, i.e., by

$$\begin{aligned} r : [0,T] &\to SE_9 = SO_9 \ltimes \mathbb{R}^9, \\ t &\mapsto r(t) = \big(R(t), s(t)\big), \end{aligned} \tag{3.3.8}$$

with

$$\begin{aligned} R &: [0,T] \to SO_9, \\ s &: [0,T] \to \mathbb{R}^9. \end{aligned} \tag{3.3.9}$$

The group SE_9 acts on any point $\mathbf{p} \in \mathbb{R}^9$ in the conventional way

$$r(t) \circ \mathbf{p} = R(t)\,\mathbf{p} + s(t). \tag{3.3.10}$$

Rolling maps describe how a manifold M rolls along a given smooth curve $\alpha : [0,T] \to M$ on its affine tangent space V at $\mathbf{p}_0 \in M$, such that $\alpha(0) = \mathbf{p}_0$. If r above is a rolling map, then the following conditions need to be satisfied.

C1 *(The rolling condition)*

For all $t \in [0,T]$ it holds

(i) $r(t) \circ \alpha(t) \in V$,

(ii) $T_{r(t)\circ\alpha(t)}\big(r(t) \circ M\big) = T_{r(t)\circ\alpha(t)}V$.

The curve $\alpha_{dev} : [0, T] \to V$, $t \mapsto r(t) \circ \alpha(t)$ is called the development of α on V.

C2 *(The no-slipping condition)*

For all $t \in [0, T]$ it holds

$\dot{r}(t) \circ r(t)^{-1} \circ \alpha_{dev}(t) = 0,$

C3 *(The no-twisting condition)*

For all $t \in [0, T]$ it holds

(i) (Tangential part)

$$\dot{r}(t) \circ r(t)^{-1} \circ T_{\alpha_{dev}(t)}V = R(t)\, R(t)^{\top} \circ T_{\alpha_{dev}(t)}V \subset (T_{\alpha_{dev}(t)}V)^{\perp},$$

(ii) (Normal part)

$$\dot{r}(t) \circ r(t)^{-1} \circ (T_{\alpha_{dev}(t)}V)^{\perp} = R(t)\, R(t)^{\top} \circ (T_{\alpha_{dev}(t)}V)^{\perp} \subset T_{\alpha_{dev}(t)}V.$$

It can be proven that *C1-C3* are the necessary and sufficient conditions for the existence and uniqueness of such a rolling map (without slipping or twisting), with given rolling curve [58].

In [32], it has been shown that for rolling the compact manifold SO_3 as a rigid body in the space $\mathbb{R}^{3\times3}$, the rotational and translational components of the original Euclidean group SE_9 turn out to be $SO_3 \times SO_3$ and $\mathbb{R}^{3\times3}$, respectively. Therefore, the rolling map r in this case can be simplified as

$$\begin{aligned} r : [0, T] &\to SO_3 \times SO_3 \ltimes \mathbb{R}^{3\times3}, \\ t &\mapsto r(t) = \big(U(t), V(t), X(t)\big), \end{aligned} \tag{3.3.11}$$

with

$$\begin{aligned} U &: [0, T] \to SO_3, \\ V &: [0, T] \to SO_3, \\ X &: [0, T] \to \mathbb{R}^{3\times3}. \end{aligned} \tag{3.3.12}$$

The group action on $\mathbf{p} \in \mathbb{R}^{3\times 3}$ is defined as

$$r(t) \circ \mathbf{p} = U(t)\, \mathbf{p}\, V(t)^{\top} + X(t). \tag{3.3.13}$$

As we are only interested in the rolling map without slipping or twisting, r must satisfy both *C2* and *C3*. Consequently, we will get a system of ordinary differential equations (the *kinematic equations*), whose solution is the rolling map:

$$\begin{aligned} \dot{X}(t) &= \Omega(t)\, \mathbf{R}_0, \\ \dot{U}(t) &= -\frac{1}{2}\, U(t)\, \Omega(t), \\ \dot{V}(t) &= \frac{1}{2}\, V(t)\, \mathbf{R}_0^{\top}\, \Omega(t)\, \mathbf{R}_0, \end{aligned} \tag{3.3.14}$$

where $\Omega : [0, T] \rightarrow \mathfrak{so}_3 = \{\Omega \in \mathbb{R}^{3\times 3} \,|\, \Omega = -\Omega^{\top}\}$ describes the rolling curve α.

If we choose $\Omega(t) = \Omega$, i.e., the manifold SO_3 rolls at a constant angular velocity, then the solution of the kinematic equations becomes

$$\begin{aligned} X(t) &= t\Omega\, \mathbf{R}_0, \\ U(t) &= \mathrm{e}^{-t\frac{\Omega}{2}}, \\ V(t) &= {\mathbf{R}_0}^{\top}\, \mathrm{e}^{t\frac{\Omega}{2}}\, \mathbf{R}_0. \end{aligned} \tag{3.3.15}$$

The derivation of Eq.3.3.14 and the proof of Eq.3.3.15 to be the rolling map without slipping or twisting for SO_3 are provided in Appendix B.

3.3.3 Interpolation Algorithm

This subsection presents a calculation scheme (**A3.1**) for smooth interpolation of orientation, i.e., our proposed solution for the question described in section 2. Most of the symbols below (denoting spaces, curves, points, and so on) are illustrated in Fig.3.2.

The calculation of **A3.1** proceeds as follows:

1. Set up the local diffeomorphism between SO_3 and the affine tangent space $T^{\text{aff}}_{\mathbf{R}_0}SO_3$, e.g., by using the exponential map,

$$\begin{aligned} \phi : \mathcal{U} &\to T^{\text{aff}}_{\mathbf{I}_3}SO_3, \\ \mathbf{R} &\mapsto \mathbf{I}_3 + \log(\mathbf{R}), \\ \phi^{-1} : T^{\text{aff}}_{\mathbf{I}_3}SO_3 &\to \mathcal{U}, \\ \mathbf{r} &\mapsto \mathrm{e}^{(\mathbf{r}-\mathbf{I}_3)}; \end{aligned} \tag{3.3.16}$$

   or by using the Gram-Schmidt orthonormalization,

$$\begin{aligned} {\phi_{GS}}^{-1} : T^{\text{aff}}_{\mathbf{I}_3}SO_3 &\to \mathcal{Q}, \\ \mathbf{I}_3 + \Omega &\mapsto (\mathbf{I}_3 + \Omega)\mathbf{U}^{-1}, \\ \phi_{GS} : \mathcal{Q} &\to T^{\text{aff}}_{\mathbf{I}_3}SO_3, \\ \mathbf{R} &\mapsto \phi_{GS}(\mathbf{R}). \end{aligned} \tag{3.3.17}$$

   Certainly, both diffeomorphisms, ϕ_{GS} as well as ϕ, have to be adapted from $\mathbf{I}_3$ to $\mathbf{R}_0$.

2. Calculate a rolling curve α, e.g., the geodesic on SO_3 connecting $\mathbf{R}_0$ and $\mathbf{R}_n$,

$$\begin{aligned} \alpha : [0, T] &\to SO_3, \\ t &\mapsto \mathrm{e}^{t\Omega}\mathbf{R}_0, \end{aligned} \tag{3.3.18}$$

   where

$$\Omega = \frac{\log(\mathbf{R}_n\mathbf{R}_0^\top)}{T}. \tag{3.3.19}$$

3. Set up the rolling map with the above defined rolling curve,

$$\begin{aligned} r : [0, T] &\to SO_3 \times SO_3 \ltimes \mathbb{R}^{3\times 3}, \\ t &\mapsto r(t) = \big(U(t), V(t), X(t)\big), \end{aligned} \tag{3.3.20}$$

where

$$\begin{aligned} U : [0,T] &\to SO_3, \\ t &\mapsto \mathrm{e}^{-t\frac{\Omega}{2}}, \\ V : [0,T] &\to SO_3, \\ t &\mapsto \mathbf{R}_0{}^{\top}\mathrm{e}^{t\frac{\Omega}{2}}\,\mathbf{R}_0, \\ X : [0,T] &\to \mathbb{R}^{3\times 3}, \\ t &\mapsto t\Omega\,\mathbf{R}_0. \end{aligned} \tag{3.3.21}$$

The group $SO_3 \times SO_3 \ltimes \mathbb{R}^{3\times 3}$ acts on any $\mathbf{p} \in \mathbb{R}^{3\times 3}$ as

$$\begin{aligned} r(t) \circ \mathbf{p} &:= U(t)\,\mathbf{p}V(t)^{\top} + X(t), \\ r(t)^{-1} \circ \mathbf{p} &:= U(t)^{\top}\big(\mathbf{p} - X(t)\big)V(t). \end{aligned} \tag{3.3.22}$$

4. Then the development of α on the affine tangent space $T^{\mathrm{aff}}_{\mathbf{R}_0}SO_3$ can be given as

$$\begin{aligned} \alpha_{dev} : [0,T] &\to T^{\mathrm{aff}}_{\mathbf{R}_0}SO_3, \\ t &\mapsto \mathbf{R}_0 + t\Omega\,\mathbf{R}_0. \end{aligned} \tag{3.3.23}$$

5. Unwrap the initial/intermediate/final orientations and initial/final angular velocities into the affine tangent space $T^{\mathrm{aff}}_{\mathbf{R}_0}SO_3$, by combining the rolling map with pull back/push forward (i.e., the local diffeomorphism),

$$\begin{aligned} \mathbf{r}_0 &= \phi\big(r(0)\circ\mathbf{R}_0 - \alpha_{dev}(0) + \mathbf{R}_0\big) + \alpha_{dev}(0) - \mathbf{R}_0 \\ &= \mathbf{R}_0, \\ \mathbf{r}_k &= \phi\big(r(t_k)\circ\mathbf{R}_k - \alpha_{dev}(t_k) + \mathbf{R}_0\big) + \alpha_{dev}(t_k) - \mathbf{R}_0, \\ \mathbf{r}_n &= \phi\big(r(T)\circ\mathbf{R}_n - \alpha_{dev}(T) + \mathbf{R}_0\big) + \alpha_{dev}(T) - \mathbf{R}_0 \\ &= \alpha_{dev}(T), \end{aligned} \tag{3.3.24}$$

$$\begin{aligned}\eta_0 &= r(0) \circ (\xi_0 + \mathbf{R}_0) \\ &= \xi_0 + \mathbf{R}_0, \\ \eta_n &= r(T) \circ (\xi_n + \mathbf{R}_n).\end{aligned} \tag{3.3.25}$$

6. Interpolate the mapped data $\mathbf{r}_0$, $\mathbf{r}_k$, $\mathbf{r}_n$, η_0, η_n with a C^2-smooth curve, e.g., a cubic spline $\beta : [0, T] \to T^{\text{aff}}_{\mathbf{R}_0} SO_3$, such that

$$\begin{aligned}\beta(0) &= \mathbf{r}_0, \\ \beta(t_k) &= \mathbf{r}_k, \\ \beta(T) &= \mathbf{r}_n, \\ \dot{\beta}(0) &= \eta_0, \\ \dot{\beta}(T) &= \eta_n.\end{aligned} \tag{3.3.26}$$

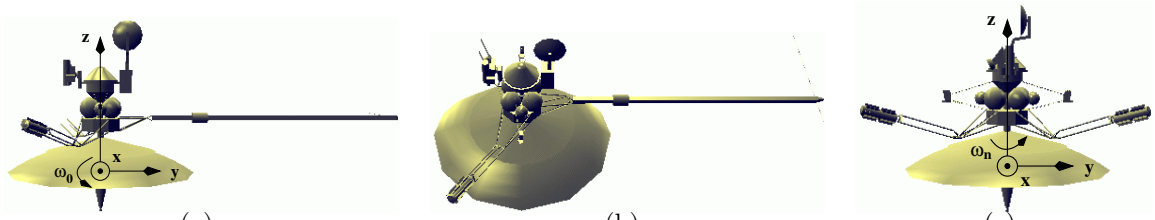
7. Wrap β back onto the manifold, the closed form of the resulting C^2-smooth curve on SO_3 is given as

$$\gamma(t) = r(t)^{-1} \circ \Big(\phi^{-1}\big(\beta(t) - \alpha_{dev}(t) + \mathbf{R}_0\big) + \alpha_{dev}(t) - \mathbf{R}_0\Big).$$

3.4 Example

This section presents a numerical example of implementing the above interpolation algorithm for designing a rotation curve of a rigid body (e.g., the Galileo Satellite here).

Suppose the satellite's initial/final orientations, as well as its initial/final angular



(a) (b) (c)

Figure 3.3: a) initial orientation $\mathbf{R}_0$, angular velocity ω_0, b) intermediate orientation $\mathbf{R}_1$, c) final orientation $\mathbf{R}_n$, angular velocity ω_n

velocities are given as below (see Fig.3.3a,3.3c)

$$\begin{aligned} \mathbf{R}_0 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \omega_0 = \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix}, \\ \xi_0 &= \widehat{\omega}_0 \mathbf{R}_0, \end{aligned} \tag{3.4.1}$$

$$\begin{aligned} \mathbf{R}_n &= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \omega_n = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}, \\ \xi_n &= \widehat{\omega}_n \mathbf{R}_n, \end{aligned}$$

where $\mathbb{R}^3 \to \mathfrak{so}_3$ is defined same as in Eq.2.4.12.

Moreover, the satellite is required to traverse an intermediate configuration $\mathbf{R}_1$ (in our case at $T/2$, see Fig.3.3b), where

$$\mathbf{R}_1 = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ 0 & 1 & 0 \\ -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{bmatrix}. \tag{3.4.2}$$

The smooth interpolation algorithm works out as follows:

1. A neat way to produce the rolling curve $\alpha : [0, T] \to SO_3$ connecting $\mathbf{R}_0$ and $\mathbf{R}_n$ can be calculated as

$$\alpha(t) = \mathrm{e}^{\Omega t}, \tag{3.4.3}$$

where

$$\Omega = \begin{bmatrix} 0 & -\frac{\pi}{2} & 0 \\ \frac{\pi}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.4.4}$$

2. The rolling map is constructed with

$$\begin{aligned} U(t) &= \mathrm{e}^{-t\Omega}, \\ V(t) &= \mathrm{e}^{t\Omega}, \\ X(t) &= t\Omega. \end{aligned} \tag{3.4.5}$$

3. α's development $\alpha_{dev} : [0, T] \to T^{\mathrm{aff}}_{\mathbf{I}_3} SO_3$ is given by

$$\alpha_{dev}(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + t\Omega. \tag{3.4.6}$$

4. The projected initial/intermediate/final orientations and initial/final angular velocities by using the exponential map are

$$\begin{aligned} \mathbf{r}_0 &= \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}, \\ \mathbf{r}_1 &= \begin{bmatrix} 1.0 & -0.0416 & 0.8051 \\ 0.0416 & 1.0 & 0.0 \\ -0.8051 & 0.0 & 1.0 \end{bmatrix}, \\ \mathbf{r}_n &= \begin{bmatrix} 1.0 & -1.5708 & 0.0 \\ 1.5708 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}, \end{aligned} \tag{3.4.7}$$

$$\eta_0 = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & -5.0 \\ 0.0 & 5.0 & 1.0 \end{bmatrix},$$
$$\eta_n = \begin{bmatrix} 1.0 & -6.5708 & 0.0 \\ 6.5708 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{3.4.8}$$

While the corresponding values by using the Gram-Schimdt orthonormalization are

$$\mathbf{r}_0 = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix},$$
$$\mathbf{r}_1 = \begin{bmatrix} 1.0 & 0.5350 & 1.4292 \\ -0.5350 & 1.0 & 0.8372 \\ -1.4292 & -0.8372 & 1.0 \end{bmatrix},$$
$$\mathbf{r}_n = \begin{bmatrix} 1.0 & -1.5708 & 0.0 \\ 1.5708 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}, \tag{3.4.9}$$
$$\eta_0 = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & -5.0 \\ 0.0 & 5.0 & 1.0 \end{bmatrix},$$
$$\eta_n = \begin{bmatrix} 1.0 & -6.5708 & 0.0 \\ 6.5708 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}.$$

5. The interpolation curve $\beta : [0, T] \to T^{\text{aff}}_{\mathbf{I}_3} SO_3$ by using the exponential map ends up with

$$\beta(t) = \begin{cases} \beta_1(t), & t \in [0, \frac{T}{2}] \\ \beta_2(t), & t \in [\frac{T}{2}, T] \end{cases}, \tag{3.4.10}$$

where

$$\beta_1(t) = \begin{bmatrix} 1.0 & -2.1881 & -12.8810 \\ 2.1881 & 1.0 & -15.0 \\ 12.8810 & 15.0 & 1.0 \end{bmatrix} t^3 + \begin{bmatrix} 1.0 & 0.9276 & 9.6608 \\ -0.9276 & 1.0 & 17.5 \\ -9.6608 & -17.5 & 1.0 \end{bmatrix} t^2$$

$$+ \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & -5.0 \\ 0.0 & 5.0 & 1.0 \end{bmatrix} t + \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix},$$

$$\beta_2(t) = \begin{bmatrix} 1.0 & -4.6703 & 12.8810 \\ 4.6703 & 1.0 & 5.0 \\ -12.8810 & -5.0 & 1.0 \end{bmatrix} t^3 + \begin{bmatrix} 1.0 & 4.6508 & -28.9823 \\ -4.6508 & 1.0 & -12.5 \\ 28.9823 & 12.5 & 1.0 \end{bmatrix} t^2$$

$$+ \begin{bmatrix} 1.0 & -1.8616 & 19.3215 \\ 1.8616 & 1.0 & 10.0 \\ -19.3215 & -10.0 & 1.0 \end{bmatrix} t + \begin{bmatrix} 1.0 & 0.3103 & -3.2202 \\ -0.3103 & 1.0 & -2.5 \\ 3.2203 & 2.5 & 1.0 \end{bmatrix}.$$

While the interpolation curve $\beta : [0, T] \to T^{\text{aff}}_{\mathbf{I}_3} SO_3$ by using the exponential map ends up with

$$\beta(t) = \begin{cases} \beta_1(t), & t \in [0, \frac{T}{2}] \\ \beta_2(t), & t \in [\frac{T}{2}, T] \end{cases}, \tag{3.4.11}$$

where

$$\beta_1(t) = \begin{bmatrix} 1.0 & -11.4136 & -22.8667 \\ 11.4136 & 1.0 & -28.3950 \\ 22.8667 & 28.3950 & 1.0 \end{bmatrix} t^3 + \begin{bmatrix} 1.0 & 7.8467 & 17.15 \\ -7.8467 & 1.0 & 27.5462 \\ -17.15 & -27.5462 & 1.0 \end{bmatrix} t^2$$

$$+ \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & -5.0 \\ 0.0 & 5.0 & 1.0 \end{bmatrix} t + \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix},$$

$$\beta_2(t) = \begin{bmatrix} 1.0 & 4.5552 & 22.8667 \\ -4.5552 & 1.0 & 18.3950 \\ -22.8667 & -18.3950 & 1.0 \end{bmatrix} t^3 + \begin{bmatrix} 1.0 & -16.1066 & -51.45 \\ 16.1066 & 1.0 & -42.6387 \\ 51.45 & 42.6387 & 1.0 \end{bmatrix} t^2$$

$$+ \begin{bmatrix} 1.0 & 11.9767 & 34.3 \\ -11.9767 & 1.0 & 30.0925 \\ -34.3 & -30.0925 & 1.0 \end{bmatrix} t + \begin{bmatrix} 1.0 & -1.9961 & -5.7167 \\ 1.9961 & 1.0 & -5.8487 \\ 5.7167 & 5.8487 & 1.0 \end{bmatrix}.$$

Several snapshots of the resulting interpolation curves are contained in Fig.3.4-3.6, in which all of the orientations are shown at equally spaced time intervals. We can see from Fig.3.4-3.6 that, being slightly different, both trajectories (left: ϕ used, right: ϕ_{GS} used) achieve the boundary conditions as well as the intermediate orientation. An M-PEG video, demonstrating the smooth interpolation result of the above Galileo Satellite example (*GalileoSatellite.mpg*) is contained in the attached CD.

3.5 Summary and Future Work

This chapter investigates a novel algorithm to calculate smooth interpolation curves of rigid-body's orientations in $\mathbb{R}^3$, i.e.,interpolation curves on the rotation group SO_3. The contribution is to combine rolling and wrapping with the simple pull back/push forward technique, so as to reduce the distortion of interpolation curves introduced by local diffeomorphisms. Remarkable features of our interpolation scheme include:

1. the approach is coordinate-free;

2. resulting curves are given in closed form.

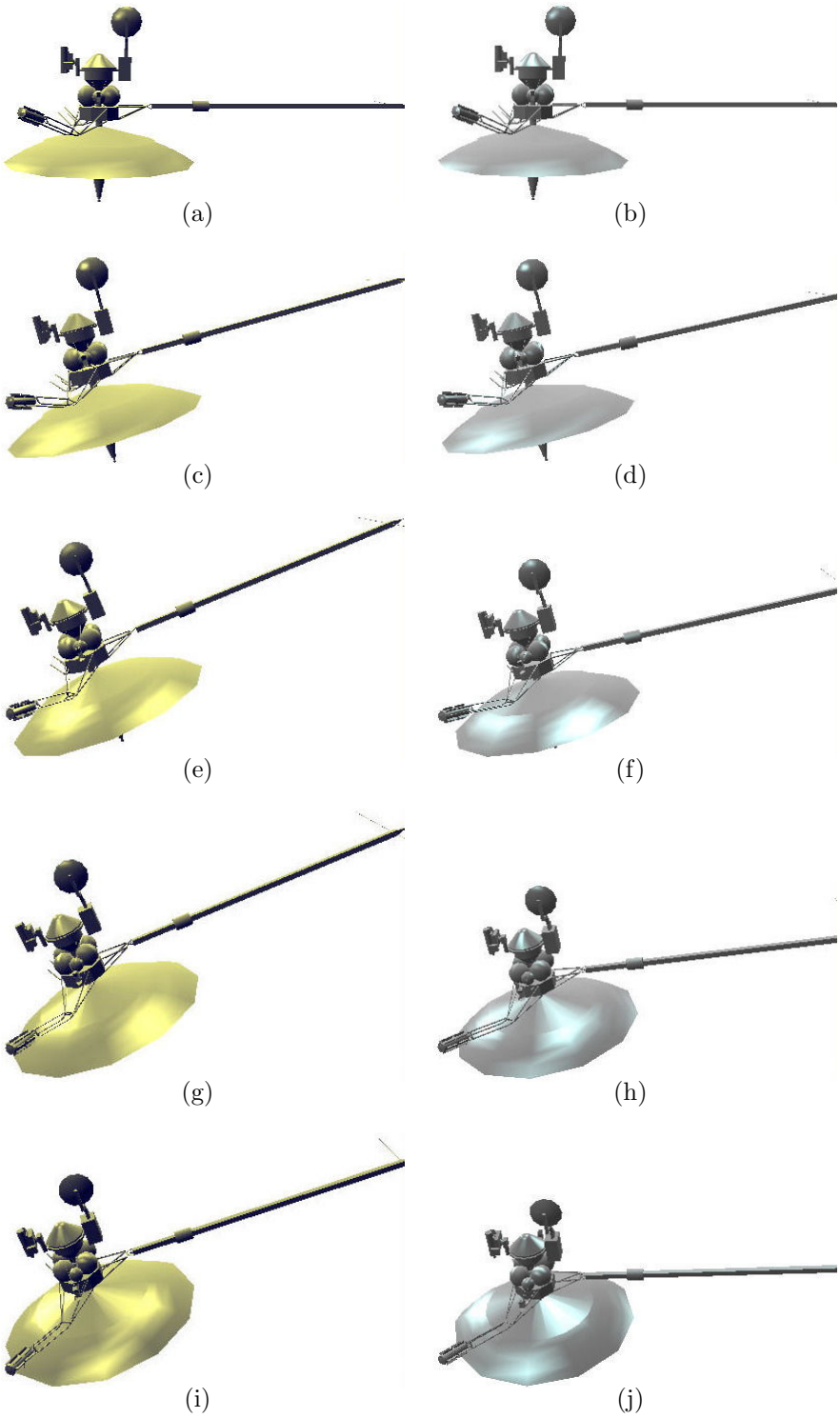


Figure 3.4: Snapshots of the interpolation curves: $\gamma(0)$, $\gamma(\frac{T}{14})$, $\gamma(\frac{T}{7})$, $\gamma(\frac{3T}{14})$, $\gamma(\frac{2T}{7})$. Left: ϕ used, Right: ϕ_{GS} used

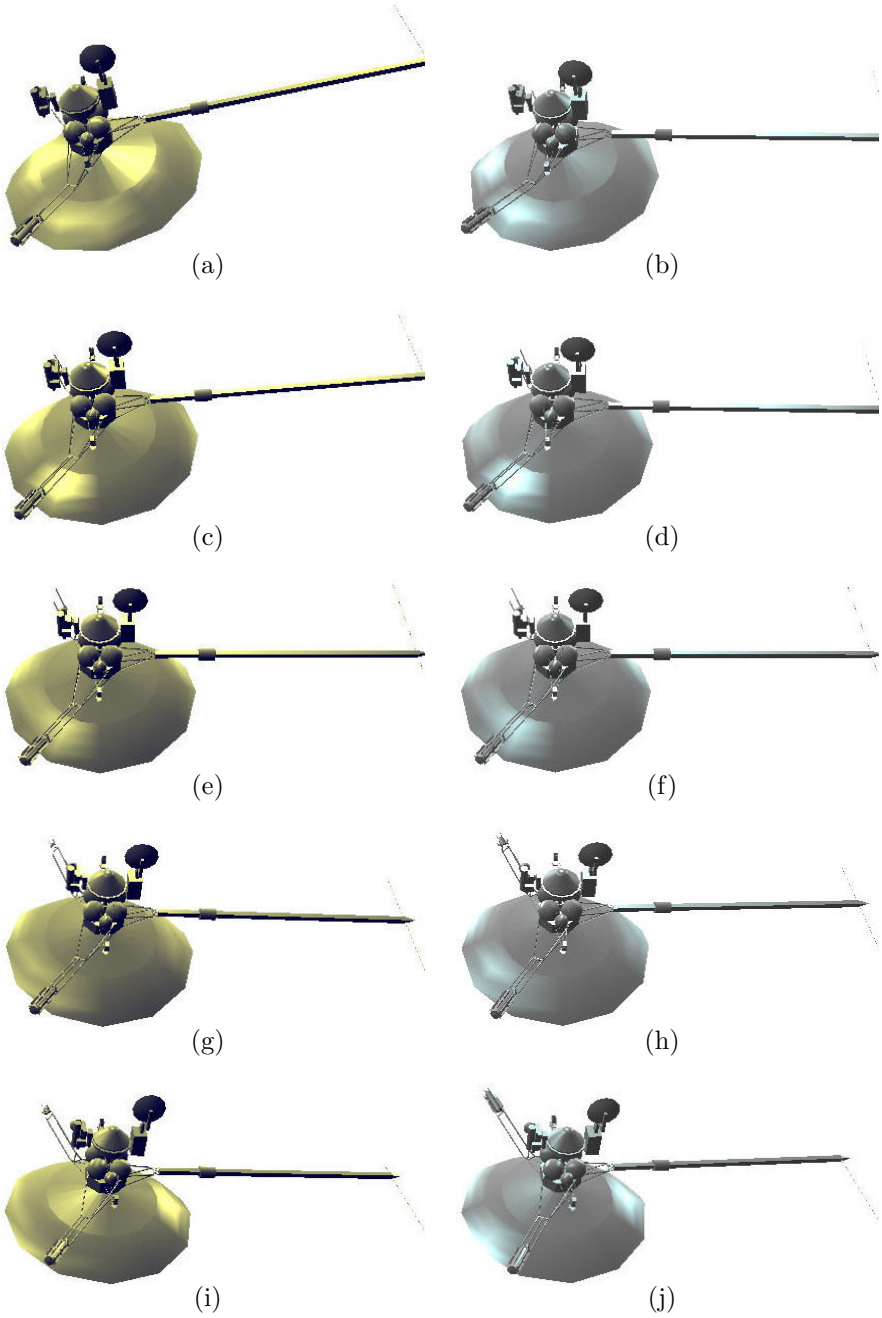


Figure 3.5: Snapshots of the interpolation curves: $\gamma(\frac{5T}{14})$, $\gamma(\frac{2T}{7})$, $\gamma(\frac{T}{2})$, $\gamma(\frac{4T}{7})$, $\gamma(\frac{9T}{14})$. Left: ϕ used, Right: ϕ_{GS} used

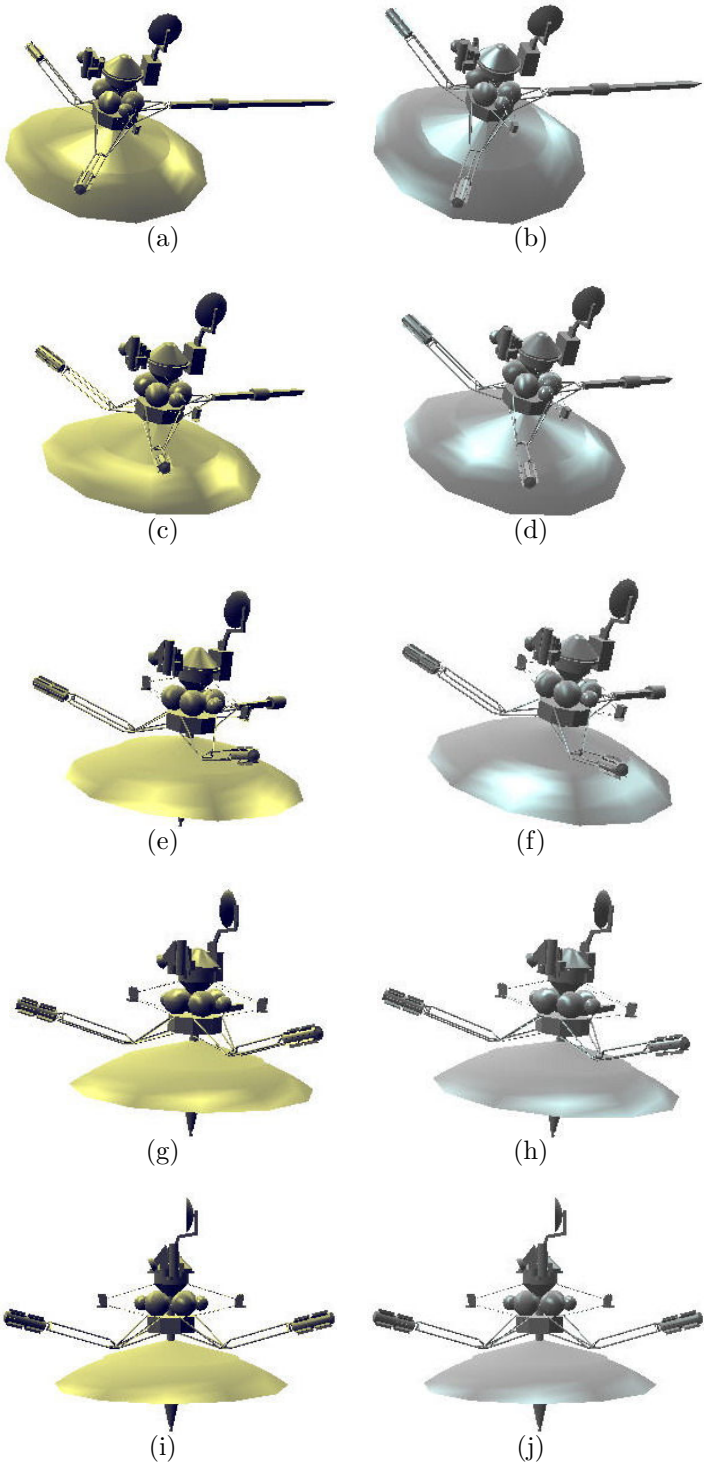


Figure 3.6: Snapshots of the interpolation curves: $\gamma(\frac{5T}{7})$, $\gamma(\frac{11T}{14})$, $\gamma(\frac{6T}{7})$, $\gamma(\frac{13T}{14})$, $\gamma(T)$. Left: ϕ used, Right: ϕ_{GS} used

As a verification, a numerical experiment along with visualization results are presented at the end of this chapter.

In the future, we are interested in investigating the influences on interpolation results and computational efficiency due to applying various pull back/push forward maps. Moreover, a thorough comparison between different interpolation approaches on SO_3 with an optimal solution approximated by solving a Euler-Lagrange equation numerically is under consideration.

In more detail, we intend to compare with respect to the performance index of resulting curve's angular acceleration

$$\min \int_0^T \|\mathrm{d}\dot{\gamma}/\mathrm{d}t\| \, \mathrm{d}t, \tag{3.5.1}$$

and some candidate interpolation methods that we can think of at the moment are as follows

1. interpolation with rolling and wrapping by using the local diffeomorphism of the exponential map as described in this chapter;

2. interpolation with rolling and wrapping by using the local diffeomorphism of the Gram-Schmidt orthonormalization (QR-decomposition) as described in this chapter;

3. the Newton's method on manifold [33] minimizing the following cost function

$$\int_0^T \|\mathrm{d}\dot{\mathbf{R}}/\mathrm{d}t\| \, \mathrm{d}t, \tag{3.5.2}$$

   where $\mathbf{R} \in SO_3$, $\mathrm{d}\dot{\mathbf{R}}/\mathrm{d}t \in \mathfrak{so}_3$ by right translation, and $\|\|$ is as defined in page 56;

4. the numerical algorithm on constrained variational problem as presented in Chapter 2, i.e.,

$$\min_{\mathbf{p}} \int_0^T \operatorname{tr}(\ddot{\mathbf{p}}^\top \ddot{\mathbf{p}})\,\mathrm{d}t + \int_0^T \operatorname{tr}\big(\mu(\mathbf{p}^\top \mathbf{p} - \mathbf{I}_3)\big)\,\mathrm{d}t, \tag{3.5.3}$$

where $\mathbf{p} \in \mathbb{R}^{3\times 3}$ and μ is the Lagrange multiplier.

Chapter 4

A Joint Space Formulation for Compliant Motion Control of Robot Manipulators

Continuing from numerical trajectory optimization and smooth interpolation, this chapter studies the dynamics and control aspects of the automatic execution of compliant motion tasks. More specifically, this chapter presents a joint space formulation for robot manipulator's hybrid motion/force control.

The motivations of this research come from 1) extending the previous work to general (either constrained or redundant) robots; and 2) improving the robustness against disturbances originated at the joint level. Contact geometry and closed-loop dynamics will be derived in this chapter, also a joint space hybrid control scheme will be proposed. Some simulation results are shown to verify the applicability of our theory on a constrained (4-degree-of-freedom) robot WAM. At the end, we suggest a compliant motion control experiment of WAM performing a 2-DOF motion and 2-DOF force task.

4.1 Background and Literature Review

As we mentioned already, there are many applications of robot manipulators requiring the end-effector to keep contact with some external environment, e.g., polishing, grasping etc. The general compliant motion control refers to those control schemes that actively maintain the contact through an extra force feedback loop. Among them, the hybrid motion/force control approach, which is firstly proposed by Raibert and Craig [54], aims to simultaneously control not only the position in the unconstrained degrees of freedom but the contact force in the constrained degrees of freedom as well. In a conventional hybrid control system, the workspace is decomposed into purely motion controlled directions and purely force controlled directions, and two parallel feedback loops are accordingly constructed for the separate control of motion and force.

The hybrid motion/force control has attracted roboticists a great amount of interests since early 1980s and there is already a rich body of literature on this topic referring to various issues including control algorithm [25, 37, 44, 48, 54, 65, 66, 69], contact modelling [10, 11], kinematic consistency [2, 18, 21, 43, 57], kinematic stability [5, 20, 27], dynamical decoupling [19, 26] etc. Fig.4.1 illustrates the generic structure for most of the existing hybrid motion/force control schemes, which we think can further be roughly divided into the following four categories.

1. *Joint space servoing without inverse dynamics*

   This stereotype appears in the pioneer work of hybrid control [54, 69] (*Block* B/C and E/F are swapped here). The desired motion/force trajectories are

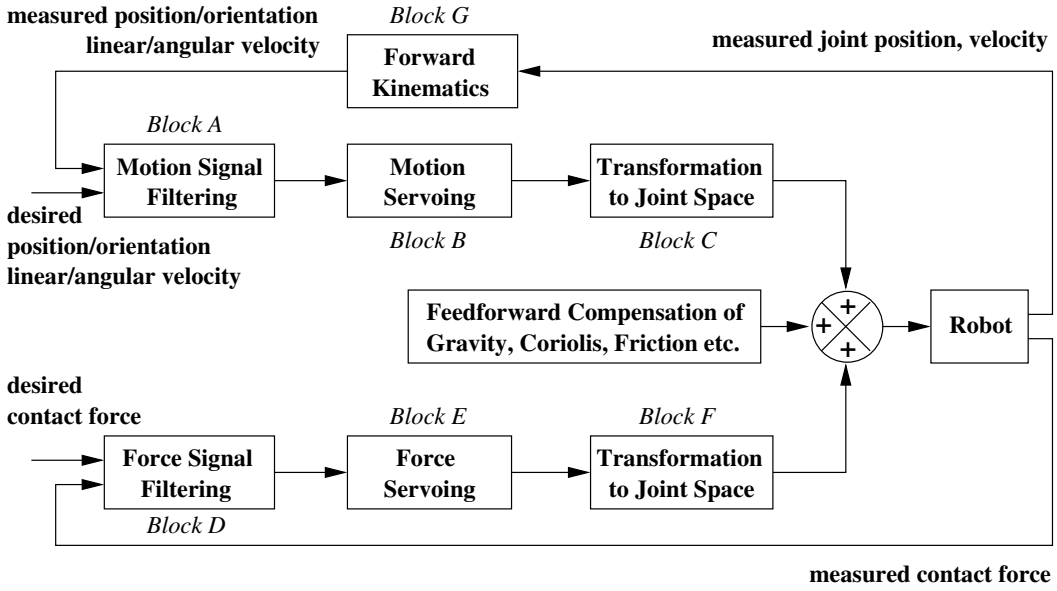


Figure 4.1: Block diagram of generic hybrid motion/force control scheme

compared with the actual measurements, then further projected onto their respective controlled subspaces through the so-called "selection matrices" (*Block A,D*). The filtered signals in the operational space will be transformed into the joint space (*Block C,F*), where some PID-like controllers are implemented (*Block B,E*), by using $\mathbf{J}^{-1}$ or $\mathbf{J}^{\top}$ ($\mathbf{J}$ is the manipulator Jacobian [51]).

2. *Operational space servoing without inverse dynamics*

   2) is similar to 1) except that the servoing units operate before the transformations, thus are formulated in the operational space, i.e., the outputs of the PID-like controllers are 6-dim wrenches, which will later be converted into the joint drive torques [65].

3. *Operational space servoing with inverse dynamics*

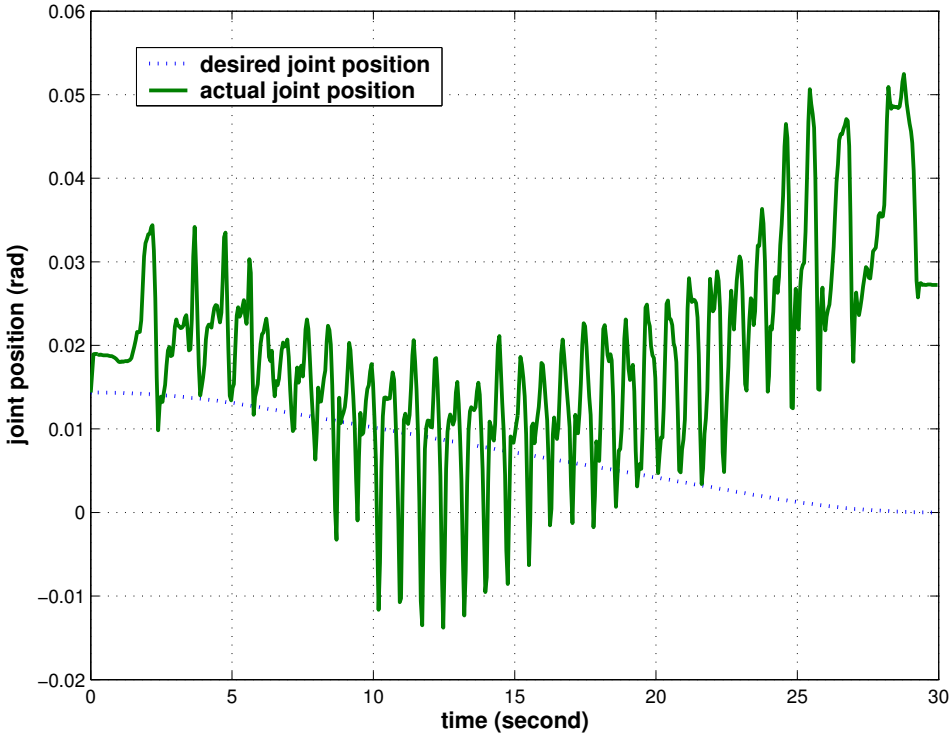
3) is similar to 2) apart from *Block B*, in which the PID-like controller produces a 6-dim acceleration instead, in other words, its parameters become time constants rather than stiffness coefficients in 2). The calculated acceleration will be multiplied by the robot's operational space inertia matrix to ultimately get the 6-dim drive wrench [37].

4. *Constraint space servoing with inverse dynamics*

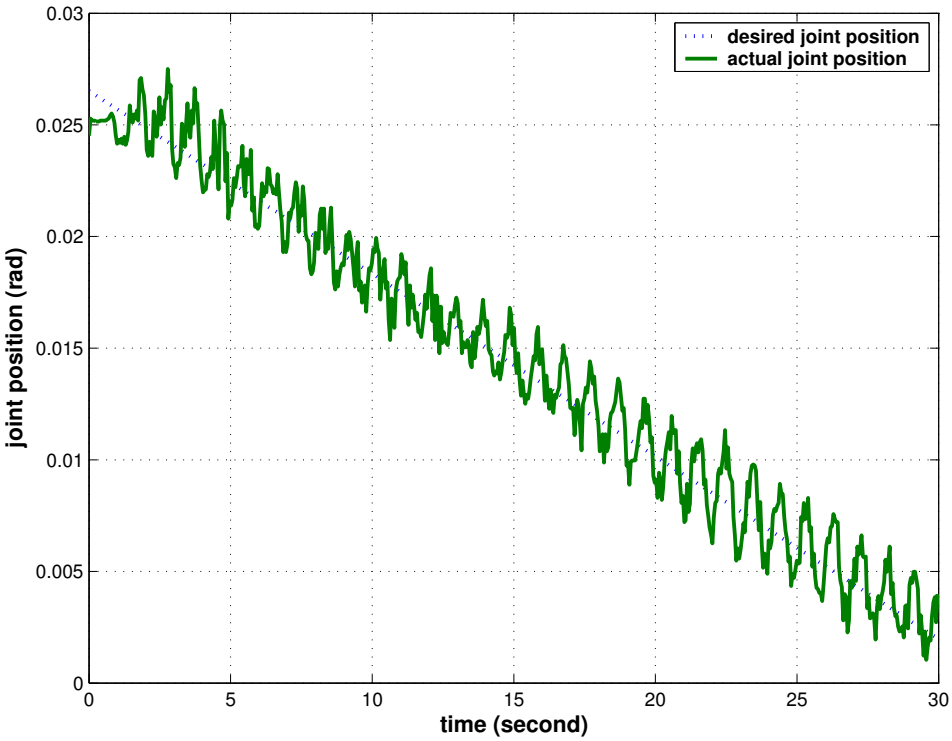
   4) deals with the filtering and servoing in a slightly different way as 1), 2) and 3). The motion/force signals are mapped into their corresponding lower-dimensional constraint spaces (also called the local spaces), and this process in fact has the same effect as the projection step of other approaches. Accordingly, the servoing is performed at the downgraded level, and the results will later be transformed up to the operational space and eventually to the joint space [44, 48, 66].

We find that most of the hybrid motion/force control algorithms in the literature have implemented both the path planning and the signal filtering in the operational space, which is clearly a reasonable choice because the contact is naturally easier described there. However, if the robot does not have exactly 6 DOFs, the implementation of the operational space oriented analysis will become a bit difficult or awkward, e.g., the operational space inertia doesn't exist for constrained robots; or for redundant robots, a natural question is how the pre-optimized information in the Jacobian's null space can be preserved through the filtering process?

Also, as we discovered from our experimentation experience, the operational-space controller is not so capable to overcome those disturbances originated at the joint level,



(a)



(b)

Figure 4.2: Tracking performance of WAM's shoulder joint: a) operational-space control law, b) joint-space control law

e.g., motor torque ripples, joint frictions etc, especially when the robot's inertia matrix is ill-conditioned [59]. Fig.4.2 compares the operational- and joint-space controllers' position tracking performances of WAM's joint 3 (see Fig.2.8) for a typical motion control task, which is to compensate some tiny discrepancies in the shoulder joint when the robot arm is fully stretching out.

Therefore, this chapter proposes to reformulate the hybrid motion/force control law entirely in the joint space. As a result, *Block C,F,G* in Fig.4.1 will be abolished in our new scheme, and the motion planning for compliant motion tasks is required to be performed at the joint level so as to fully comply with the contact task and the robot kinematics. The numerical optimal path planning method we developed in Chapter 2 for robots subject to motion constraints can be used to calculate the desired joint trajectories in this scenario.

The rest of this chapter is organized as follows: Section 2 formulates the contact model and derives the robot closed-loop dynamics; Section 3 presents our joint space hybrid control algorithm and discusses a few issues such as dynamical decoupling, asymptotic stability etc; Section 4 shows the simulation results of applying our control law on the 4-DOF experimental robot manipulator WAM; and Section 5 proposes an dual-contact compliant motion experiment.

4.2 Contact Model and Robot Closed-loop Dynamics

This section first reviews the concept of the dual vector spaces $\mathcal{M}^6$ ($\mathcal{M}^q$) and $\mathcal{F}^6$ ($\mathcal{F}^q$), which respectively represent velocities/accelerations and forces in the rigid body

(robot) dynamics. Next we introduce the contact model, first formulated in the operational space, then migrated to the joint space, and derive the robot manipulator's closed-loop dynamics. Our discoveries and the known results will be compared at the end of this section.

4.2.1 Geometry of Constrained Rigid Body (Robot) Systems

Consider a single rigid body which can translate and rotate freely in a 3-dim workspace. The configuration space [38] $\mathcal{E}$ for such an object is then the Euclidian group SE_3. By right translation, the tangent space to $\mathcal{E}$ at any $\hat{\mathbf{p}} \in \mathcal{E}$ can be identified with the Lie algebra $\mathfrak{se}_3$, which is a 6-dim real vector space, denoted here by $\mathcal{M}^6$. The space $\mathcal{M}^6$ consists of all possible rigid body velocity vectors $\hat{\mathbf{v}}$ of the free-flying object [51] for a given configuration.

Meanwhile, we can introduce the dual space of $\mathcal{M}^6$, denoted by $\mathcal{F}^6$, which can be identified with the set of linear functionals $F : \mathcal{M}^6 \to \mathbb{R}$. If we represent F as a 6-dim column vector $\hat{\mathbf{f}}$, then the canonical evaluation map $F(\hat{\mathbf{v}})$ is given as

$$F : \mathcal{M}^6 \to \mathbb{R},\ \hat{\mathbf{v}} \mapsto \hat{\mathbf{f}}^\top \hat{\mathbf{v}}, \tag{4.2.1}$$

where $\hat{\mathbf{f}}$ can be regarded as any 6-dim wrench (generalized force) [51] applied on or by the rigid body object [44, 57].

Now suppose we have a q-joint robot manipulator constrained by mechanical stops, i.e., the joint position $\mathbf{q}$ is an element of the direct product of q open intervals. Ideally, the set of joint velocities $\dot{\mathbf{q}}$ can be identified with the q-dim real vector space $\mathcal{M}^q$, and any joint acceleration $\ddot{\mathbf{q}} \in \mathcal{M}^q$ as well. Similar to the analysis above, $\mathcal{M}^q$'s dual space $\mathcal{F}^q$ can be considered as the set of all joint torques τ.

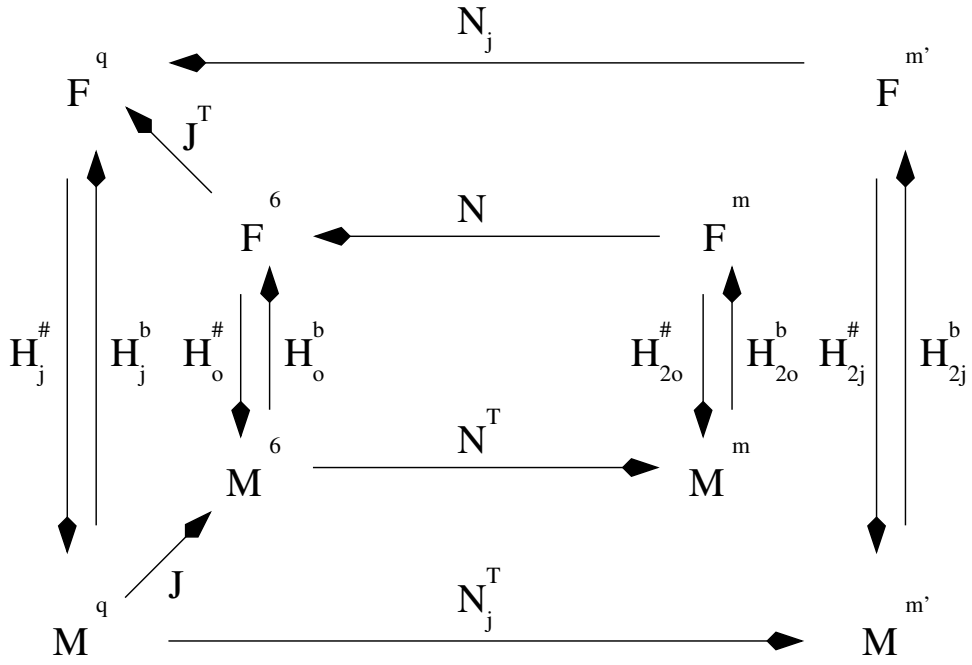


Figure 4.3: Commutative maps connecting operational, constraint, joint, and joint-constraint Spaces

Between the corresponding dual spaces, we can study a linear map, denoted by $\mathbf{H}_j^\sharp : \mathcal{F}^q \to \mathcal{M}^q$ (the subscript j or o in this chapter stands for the joint or operational space); and if it is invertible, we call the inverse $\mathbf{H}_j^\flat$. For example, we can choose the robot inertia $\mathbf{H}$ as $\mathbf{H}_j^\flat$

$$\mathbf{H} : \mathcal{M}^q \to \mathcal{F}^q, \ddot{\mathbf{q}} \mapsto \tau. \tag{4.2.2}$$

The linear map $\mathbf{H}$ is well known to be symmetric and positive definite, in particular, it is invertible. Likewise, we can define maps in the operational space, $\mathbf{H}_o^\sharp : \mathcal{F}^6 \to \mathcal{M}^6$ or $\mathbf{H}_o^\flat : \mathcal{M}^6 \to \mathcal{F}^6$.

To relate these properties in the joint and operational space, we need to employ the manipulator Jacobian, which is usually defined as the map from the joint to the

end-effector velocities. More specifically, if the above mentioned rigid body is the end-effector of the q-joint robot manipulator, we have the following relationship

$$\hat{\mathbf{v}} = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}\,, \tag{4.2.3}$$

where $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{6\times q} : \mathcal{M}^q \to \mathcal{M}^6, \dot{\mathbf{q}} \mapsto \hat{\mathbf{v}}$. By duality, the transpose of $\mathbf{J}$ can serve as the map between the corresponding dual spaces, i.e.,

$$\tau = \mathbf{J}(\mathbf{q})^\top \hat{\mathbf{f}}. \tag{4.2.4}$$

Combining Eqs.4.2.3 and 4.2.4 with 4.2.2, we get $\mathbf{H}_o^\sharp = \mathbf{J}\mathbf{H}^{-1}\mathbf{J}^\top$; and if $\mathbf{J}$ is invertible, $\mathbf{H}_o^\flat = \mathbf{J}^{-\top}\mathbf{H}\mathbf{J}^{-1}$ holds as well. However, we need to be aware that $\mathbf{H}_o^\flat$ does not always exist, e.g., if the robot has less than 6 DOFs or is at a singularity.

Next we model the force or motion constraints of the robot's end-effector due to the contact as

$$\hat{\mathbf{f}} = \mathbf{N}\alpha,\ \mathbf{N}^\top\hat{\mathbf{v}} = 0, \tag{4.2.5}$$

where $\alpha \in \mathcal{F}^m$, and $\mathcal{F}^m$ is the so-called constraint space (or local space). The matrix $\mathbf{N} \in \mathbb{R}^{6\times m}$ has full column rank, representing the linear map from $\mathcal{F}^m$ to $\mathcal{F}^6$. Moreover, it can be considered as a basis matrix for the m-dim subspace for all of the achievable contact forces, denoted by $\mathcal{N}$, in $\mathcal{F}^6$. Note that the constraint $\mathbf{N}^\top\hat{\mathbf{v}} = 0$ defines the $(6-m)$-dim subspace $\mathcal{T}$ for all of the allowable velocities in $\mathcal{M}^6$ at a given configuration. If $\mathbf{T}$ is a matrix representation of $\mathcal{T}$, then $\mathbf{N}^\top\mathbf{T} = 0$ (called reciprocity).

For example, the compliant motion task raised in Chapter 2 (Fig.2.1) requires the manipulator's end-effector to move on and maintain a single-point contact with a 2-dim plane. Again, $\mathbf{n} \in \mathbb{R}^3$ is the plane normal, $\mathbf{p} \in \mathbb{R}^3$ is the linear position of

the end-effector, both expressed in the inertial frame. Using the cross product in $\mathbb{R}^3$, in this situation,

$$\mathbf{N} = \begin{bmatrix} \mathbf{n} \\ \mathbf{p} \times \mathbf{n} \end{bmatrix}, \alpha \in \mathcal{F}^1 = \mathbb{R}\,. \tag{4.2.6}$$

Combining Eq.4.2.5 with Eqs.4.2.3 and 4.2.4, we get

$$\tau = \mathbf{J}^\top \mathbf{N}\alpha = \mathbf{N}_j \alpha',\ \mathbf{N}^\top \mathbf{J}\dot{\mathbf{q}} = \mathbf{N}_j^\top \dot{\mathbf{q}} = 0\,, \tag{4.2.7}$$

where $\alpha' \in \mathcal{F}^{m'}$, $m' \leq m$, $\mathbf{N}_j \in \mathbb{R}^{q \times m'}$ also has full column rank. It happens that $m' < m$ if $\mathbf{J}^\top \mathbf{N}$ is singular, i.e., $\mathrm{Ker}(\mathbf{J}^\top) \cap \mathcal{N} \neq \{0\}$. In this case, $\mathbf{N}_j$ can be obtained by performing a singular value decomposition on $\mathbf{J}^\top \mathbf{N}$

$$\mathbf{J}^\top \mathbf{N} = \mathbf{U}\Sigma\mathbf{V}^\top, \tag{4.2.8}$$

and extracting those column vectors, m' in number, in $\mathbf{U}$ corresponding to the m' non-zero singular values. In the same way, $\mathbf{N}_j$ defines the subspaces $\mathcal{N}_j$ and $\mathcal{T}_j$ in $\mathcal{F}^q$ and $\mathcal{M}^q$ respectively.

Remember $\mathbf{H}$ being symmetric positive definite. The $\mathbf{H}^{-1}$-orthogonal subspace of $\mathcal{N}_j$, denoted by $\mathcal{N}_j'$, or the $\mathbf{H}$-orthogonal subspace of $\mathcal{T}_j$, denoted by $\mathcal{T}_j'$, can be given as

$$\begin{aligned} \mathcal{N}_j' &= \{\tau_2 \,|\, \tau_1^\top \mathbf{H}^{-1} \tau_2 = 0,\ \tau_1 \in \mathcal{N}_j\}, \\ \mathcal{T}_j' &= \{\ddot{\mathbf{q}}_2 \,|\, \ddot{\mathbf{q}}_1^\top \mathbf{H} \ddot{\mathbf{q}}_2 = 0,\ \ddot{\mathbf{q}}_1 \in \mathcal{T}_j\}. \end{aligned} \tag{4.2.9}$$

The connections among those spaces and constraint sub-spaces are illustrated in Fig.4.3.

According to Fig.4.3, we can build up the "dynamical projectors" Φ_{fj}, Φ'_{fj}, Φ'_{mj} and Φ_{mj} as follows (apart from j and o, the subscript m or f stands for motion or

force; and $\mathbf{I}_n$ denotes the $n \times n$ identity matrix). The meaning of this name will become clearer in the next subsection.

$$\Phi_{fj} : \mathcal{F}^q \to \mathcal{N}_j,\ \Phi_{fj} = \mathbf{N}_j(\mathbf{N}_j^\top \mathbf{H}^{-1}\mathbf{N}_j)^{-1}\mathbf{N}_j^\top \mathbf{H}^{-1}, \tag{4.2.10}$$

$$\Phi'_{fj} : \mathcal{F}^q \to \mathcal{N}'_j,\ \Phi'_{fj} = \mathbf{I}_q - \Phi_{fj}, \tag{4.2.11}$$

$$\Phi'_{mj} : \mathcal{M}^q \to \mathcal{T}'_j,\ \Phi'_{mj} = \mathbf{H}^{-1}\mathbf{N}_j(\mathbf{N}_j^\top \mathbf{H}^{-1}\mathbf{N}_j)^{-1}\mathbf{N}_j^\top, \tag{4.2.12}$$

$$\Phi_{mj} : \mathcal{M}^q \to \mathcal{T}_j,\ \Phi_{mj} = \mathbf{I}_q - \Phi'_{mj}, \tag{4.2.13}$$

$$\Phi_{fj} = \mathbf{H}\Phi'_{mj}\mathbf{H}^{-1},\ \Phi'_{fj} = \mathbf{H}\Phi_{mj}\mathbf{H}^{-1}. \tag{4.2.14}$$

Because $\mathbf{H}$ is symmetric positive definite, it is easily seen that

$$\Phi_{fj}^2 = \Phi_{fj}\,, \tag{4.2.15}$$

$$\langle \tau_1, \Phi_{fj}\tau_2 \rangle_{\mathbf{H}^{-1}} = \langle \Phi_{fj}\tau_1, \tau_2 \rangle_{\mathbf{H}^{-1}}\,, \tag{4.2.16}$$

where

$$\langle\ ,\ \rangle_{\mathbf{H}^{-1}} : \mathcal{F}^q \times \mathcal{F}^q \to \mathbb{R},\ \langle \tau_1, \tau_2 \rangle_{\mathbf{H}^{-1}} \mapsto \tau_1 \mathbf{H}^{-1}\tau_2\,. \tag{4.2.17}$$

Similar properties as Eqs.4.2.15,4.2.16 hold for three other dynamical projectors as well, e.g., Φ_{mj} is an orthogonal projector being self-adjoint but with respect to $\langle\ ,\ \rangle_{\mathbf{H}}$.

4.2.2 Equations of Motion

Recall the conventional robot dynamics equation together with the motion constraints Eq.4.2.7, the robot closed-loop dynamics can be expressed as the system

$$\begin{aligned} \mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \tau_c &= \tau\,, \\ \mathbf{N}_j^\top \ddot{\mathbf{q}} + \dot{\mathbf{N}}_j^\top \dot{\mathbf{q}} &= 0\,, \end{aligned} \tag{4.2.18}$$

where $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the gravity combined with the Coriolis term. Moreover, $\tau_c = \mathbf{N}_j \alpha'$ is the corresponding joint torque to the contact force applied from the robot to the environment, and τ is the entire joint drive torque.

From Eq.4.2.18, we can solve for α' and $\ddot{\mathbf{q}}$ as

$$\begin{aligned} \alpha' &= (\mathbf{N}_j^\top \mathbf{H}^{-1} \mathbf{N}_j)^{-1} \Big(\dot{\mathbf{N}}_j^\top \dot{\mathbf{q}} + \mathbf{N}_j^\top \mathbf{H}^{-1} (\tau - \mathbf{C}) \Big), \\ \ddot{\mathbf{q}} &= \mathbf{H}^{-1} \Big(\big(\mathbf{I}_q - \mathbf{N}_j (\mathbf{N}_j^\top \mathbf{H}^{-1} \mathbf{N}_j)^{-1} \mathbf{N}_j^\top \mathbf{H}^{-1} \big) (\tau - \mathbf{C}) \\ &\qquad - \mathbf{N}_j (\mathbf{N}_j^\top \mathbf{H}^{-1} \mathbf{N}_j)^{-1} \dot{\mathbf{N}}_j^\top \dot{\mathbf{q}} \Big). \end{aligned} \tag{4.2.19}$$

After some algebraic manipulation, Eq.4.2.19 can be rewritten as

$$\begin{aligned} \mathbf{H} \Phi_{mj} \ddot{\mathbf{q}} &= \Phi'_{fj} (\tau - \mathbf{C}) \,, \\ \mathbf{H} \Phi'_{mj} \ddot{\mathbf{q}} &= \Phi_{fj} (\tau - \mathbf{C}) - \tau_c \,. \end{aligned} \tag{4.2.20}$$

From Eq.4.2.20, we can see that by using the dynamical projectors Φ_{fj}, Φ'_{fj}, Φ'_{mj} and Φ_{mj}, the closed-loop dynamics can be analyzed as two decoupled subsystems.

When $\mathbf{J}^\top \mathbf{N}$ has full rank, $\mathbf{N}_j$ can be substituted by $\mathbf{J}^\top \mathbf{N}$, as a result, Eq.4.2.20 will transform into Yoshikawa's formulation [66].

Furthermore, as we already mentioned before, if the robot manipulator has 6 DOFs and is free from singularities, i.e., $\mathbf{J}$ is invertible, there exist the following transformations

$$\mathbf{H}_o^\flat = \mathbf{J}^{-\top} \mathbf{H} \mathbf{J}^{-1} = \mathbf{H}_o, \;\; \mathbf{H}_o^\sharp = \mathbf{J} \mathbf{H}^{-1} \mathbf{J}^\top = \mathbf{H}_o^{-1}, \tag{4.2.21}$$

where $\mathbf{H}_o$ is called the operational space inertia matrix, and

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J}^\top \mathbf{C}_o(\hat{\mathbf{p}}, \hat{\mathbf{v}}) + \mathbf{H} \mathbf{J}^{-1} \dot{\mathbf{J}} \dot{\mathbf{q}} \,, \tag{4.2.22}$$

where $\mathbf{C}_o$ combines the operational space gravity and Coriolis term [37].

Plug Eqs.4.2.21, 4.2.22, $\dot{\mathbf{N}}_j^\top = \mathbf{N}^\top\dot{\mathbf{J}}+\dot{\mathbf{N}}^\top\mathbf{J}$ and $\hat{\mathbf{a}} = \mathbf{J}\ddot{\mathbf{q}}+\dot{\mathbf{J}}\dot{\mathbf{q}}$ ($\hat{\mathbf{a}}$ being the derivative of $\hat{\mathbf{v}}$ with respect to time) into Eq.4.2.18, we then get a similar formulation of the robot closed-loop dynamics but now formulated in operational space:

$$\begin{aligned}\mathbf{H}_o(\hat{\mathbf{p}})\hat{\mathbf{a}}+\mathbf{C}_o(\hat{\mathbf{p}},\hat{\mathbf{v}})+\hat{\mathbf{f}}_c &= \hat{\mathbf{f}},\\ \mathbf{N}^\top\hat{\mathbf{a}}+\dot{\mathbf{N}}^\top\hat{\mathbf{v}} &= 0,\end{aligned} \tag{4.2.23}$$

where $\hat{\mathbf{f}}_c$ is the contact force and $\hat{\mathbf{f}}$ is the 6-dim drive wrench.

The system response of Eq.4.2.23 can be brought to a similar form as Eq.4.2.20, which is consistent with the Lagrangian formulation in [44]:

$$\begin{aligned}\mathbf{H}_o\Phi_{mo}\hat{\mathbf{a}} &= \Phi'_{fo}(\hat{\mathbf{f}}-\mathbf{C}_o)\,,\\ \mathbf{H}_o\Phi'_{mo}\hat{\mathbf{a}} &= \Phi_{fo}(\hat{\mathbf{f}}-\mathbf{C}_o)-\hat{\mathbf{f}}_c\,,\end{aligned} \tag{4.2.24}$$

where

$$\Phi_{fo}:\mathcal{F}^6\rightarrow\mathcal{N},\ \Phi_{fo}=\mathbf{N}(\mathbf{N}^\top\mathbf{H}_o^{-1}\mathbf{N})^{-1}\mathbf{N}^\top\mathbf{H}_o^{-1}, \tag{4.2.25}$$

$$\Phi'_{fo}:\mathcal{F}^6\rightarrow\mathcal{N}',\ \Phi'_{fo}=\mathbf{I}_6-\Phi_{fo}, \tag{4.2.26}$$

$$\Phi'_{mo}:\mathcal{M}^6\rightarrow\mathcal{T}',\ \Phi'_{mo}=\mathbf{H}_o^{-1}\mathbf{N}(\mathbf{N}^\top\mathbf{H}_o^{-1}\mathbf{N})^{-1}\mathbf{N}^\top, \tag{4.2.27}$$

$$\Phi_{mo}:\mathcal{M}^6\rightarrow\mathcal{T},\ \Phi_{mo}=\mathbf{I}_6-\Phi'_{mo}, \tag{4.2.28}$$

$$\Phi_{fo}=\mathbf{H}_o\Phi'_{mo}\mathbf{H}_o^{-1},\ \Phi'_{fo}=\mathbf{H}_o\Phi_{mo}\mathbf{H}_o^{-1}. \tag{4.2.29}$$

Again, $\mathcal{N}'$ and $\mathcal{T}'$ are defined by $\mathbf{H}_o$ and $\mathbf{H}_o^{-1}$ in the same manner as in Eq.4.2.9.

Finally, if comparing Eqs.4.2.25-4.2.28 with Eqs.4.2.10-4.2.13, we will find

$$\Phi_{fj}=\mathbf{J}^\top\Phi_{fo}\mathbf{J}^{-\top},\ \Phi'_{fj}=\mathbf{J}^\top\Phi'_{fo}\mathbf{J}^{-\top}, \tag{4.2.30}$$

$$\Phi'_{mj}=\mathbf{J}^{-1}\Phi'_{mo}\mathbf{J},\ \Phi_{mj}=\mathbf{J}^{-1}\Phi_{mo}\mathbf{J}\,. \tag{4.2.31}$$

4.3 Hybrid Motion/Force Control Scheme Formulated in Joint Space

Based on the robot closed-loop dynamics (Eq.4.2.20), we move on to the joint space hybrid motion/force control scheme design in this section. The open-loop and the closed-loop control laws will be proposed one after another, and a key component called "kinematic projector" will be introduced and explained in detail. Also, during the development of our control algorithm, we will discuss two important issues of the system response: dynamical decoupling and asymptotic stability.

4.3.1 Open-loop Control Law Design and Dynamical Decoupling

Assuming the robot dynamics model is available, now consider the following open-loop control law

$$\tau = \mathbf{H}\ddot{\mathbf{q}}_d + \mathbf{C} + \mathbf{J}^\top \hat{\mathbf{f}}_{cd}\,, \tag{4.3.1}$$

where $\ddot{\mathbf{q}}_d$ and $\hat{\mathbf{f}}_{cd}$ are the desired joint acceleration and contact force, respectively.

If $\mathbf{J}^\top \hat{\mathbf{f}}_{cd} \in \mathcal{N}_j$, inserting Eq.4.3.1 into Eq.4.2.20, we get

$$\begin{aligned} \mathbf{H}\Phi_{mj}\ddot{\mathbf{q}} &= \mathbf{H}\Phi_{mj}\ddot{\mathbf{q}}_d\,, \\ \mathbf{J}^\top \hat{\mathbf{f}}_c + \mathbf{H}\Phi'_{mj}\ddot{\mathbf{q}} &= \mathbf{H}\Phi'_{mj}\ddot{\mathbf{q}}_d + \mathbf{J}^\top \hat{\mathbf{f}}_{cd}\,. \end{aligned} \tag{4.3.2}$$

Eq.4.3.2 can be regarded as the joint space version of the existing result of dynamical decoupling analysis [19, 26], which shows the independence between the component of $\hat{\mathbf{a}}$ in $\mathcal{T}$ and $\hat{\mathbf{f}}_c$ in $\mathcal{N}$ attributed to an operational space hybrid control law equivalent to Eq.4.3.1.

Now if we make $\mathbf{J}^\top \hat{\mathbf{f}}_{cd}$ and $\ddot{\mathbf{q}}_d$ both comply with the contact model, i.e., $\mathbf{J}^\top \hat{\mathbf{f}}_{cd} \in \mathcal{N}_j$ and $\mathbf{N}_j^\top \ddot{\mathbf{q}}_d - \dot{\mathbf{N}}_j^\top \dot{\mathbf{q}} = 0$ as well, then

$$\mathbf{N}_j^\top \ddot{\mathbf{q}}_d = \dot{\mathbf{N}}_j^\top \dot{\mathbf{q}} = \mathbf{N}_j^\top \ddot{\mathbf{q}}\,, \tag{4.3.3}$$

which leads to

$$\Phi'_{mj} \ddot{\mathbf{q}}_d = \Phi'_{mj} \ddot{\mathbf{q}}\,. \tag{4.3.4}$$

Finally, from Eqs.4.3.2,4.3.4, a "genuine" dynamical decoupling form between motion and force can be shown as

$$\begin{aligned} \ddot{\mathbf{q}} &= \ddot{\mathbf{q}}_d\,, \\ \mathbf{J}^\top \hat{\mathbf{f}}_c &= \mathbf{J}^\top \hat{\mathbf{f}}_{cd}\,. \end{aligned} \tag{4.3.5}$$

4.3.2 Closed-loop Control Law Design and Stability Analysis

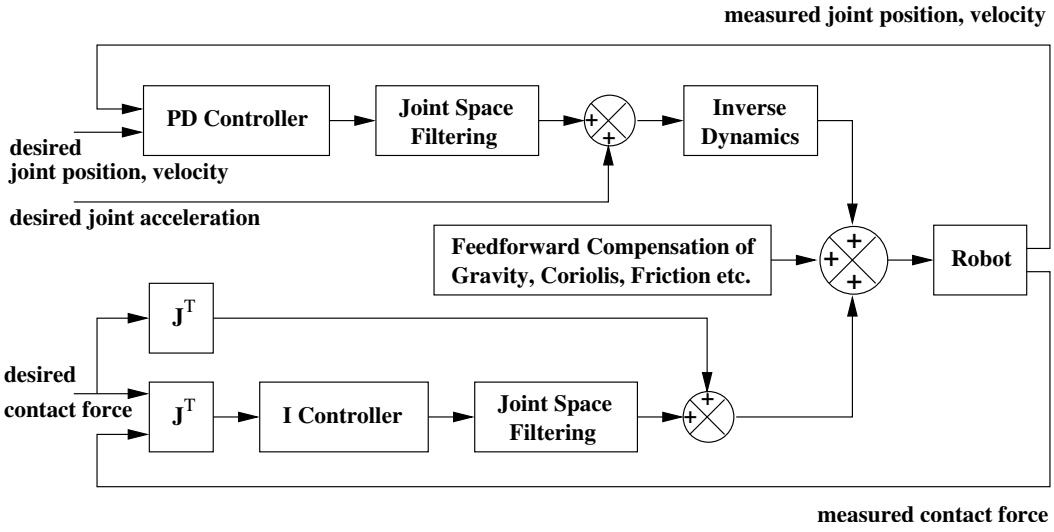


Figure 4.4: Block diagram of joint-space hybrid motion/force control scheme

As disturbances are usually inevitable in real life robotic systems, it is always essential to implement some servoing mechanisms to improve a control law's robustness.

Consider combining a motion PD-controller and a force I-controller with Eq.4.3.1, our closed-loop control algorithm is finally designed as

$$\begin{aligned}\tau = \mathbf{H}\big(\ddot{\mathbf{q}}_d + \Psi_{mj}\mathbf{K}_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \Psi_{mj}\mathbf{K}_p(\mathbf{q}_d - \mathbf{q})\big) \\ + \Big(\mathbf{J}^\top \hat{\mathbf{f}}_{cd} + \Psi_{fj}\,\mathbf{K}_i \int \mathbf{J}^\top(\hat{\mathbf{f}}_{cd} - \hat{\mathbf{f}}_c)\,\mathrm{d}t\Big) + \mathbf{C}\,,\end{aligned} \tag{4.3.6}$$

where $\mathbf{K}_p$, $\mathbf{K}_v$ and $\mathbf{K}_i$ are the $(q\times q)$-dim coefficient matrices for the joint space PD/I-controllers. Ψ_{fj} and Ψ_{mj} are named kinematic projectors (in contrast to dynamical projectors, Eqs.4.2.10-4.2.13) which are used to filter out the noise incompatible with the contact model from the original force or motion signal. Details will be presented next. Fig.4.4 illustrates the general data flow of Eq.4.3.6. Comparing with other hybrid motion/force control schemes, our design may be summarized as "joint space servoing, filtering with inverse dynamics".

Recall how the dynamical projectors are derived in the last section, e.g., $\Phi_{mj} : \mathcal{M}^q \to \mathcal{T}_j$ (Eq.4.2.13), where $\mathcal{T}_j$ directly comes from the contact and robot kinematic models (Eq.4.2.7), and its complement $\mathcal{T}_j'$ is defined from $\mathcal{T}_j$ together with the robot inertia $\mathbf{H}$ (Eq.4.2.9). The joint space dynamical projector can be transformed into the operational space (Eqs.4.2.30,4.2.31) when the Jacobian $\mathbf{J}$ is invertible, in other words, $\mathbf{H}_o$ exists.

The kinematic projectors on the contrary are sourced from kinematic models only. Ψ_{fj} and Ψ_{mj} in Eq.4.3.6 are actually derived *from* the operational space projectors, e.g., "selection matrices" [54] or "kinestatic filters" [43], and function in a similar way as them but in the joint space.

Let's repeat the original analysis of the motion constraints (Eq.4.2.5): when the end-effector is in contact with the environment, $\mathcal{M}^6$ or $\mathcal{F}^6$ consequently splits into $\mathcal{T}$ and $\mathcal{T}'$ or $\mathcal{N}$ and $\mathcal{N}'$, whose matrix forms are $\mathbf{T}$, $\mathbf{T}'$, $\mathbf{N}$ and $\mathbf{N}'$ respectively. Different

from Eq.4.2.9, $\mathcal{T}'$ or $\mathcal{N}'$ here are not the dual subspaces of $\mathcal{N}$ or $\mathcal{T}$ in $\mathcal{M}^6$ or $\mathcal{F}^6$ respectively with regard to the linear map of $\mathbf{H}_o^\sharp$ or $\mathbf{H}_o^\flat$, but are the constrained subspaces of $\hat{\mathbf{v}}$ or $\hat{\mathbf{f}}$ due to the contact. $\mathbf{T}$, $\mathbf{T}'$, $\mathbf{N}$ and $\mathbf{N}'$ will satisfy [19]

$$\mathbf{N}^\top\mathbf{T} = 0,\ \mathbf{N}'^\top\mathbf{T}' = 0\,, \tag{4.3.7}$$

$$\mathbf{N}'^\top\mathbf{T} = \mathbf{I}_{6-m},\ \mathbf{N}^\top\mathbf{T}' = \mathbf{I}_m\,. \tag{4.3.8}$$

Since $\mathbf{T}$ and $\mathbf{T}'$ are not necessarily $\mathbf{I}_n$-orthogonal, neither are $\mathbf{N}$ and $\mathbf{N}'$ [21], the operational space kinematic projector $\Psi_{mo} : \mathcal{M}^6 \to \mathcal{T}$ need to be given as [49]

$$\Psi_{mo} = [\,\mathbf{T}\,|\,\mathbf{T}'\,] \begin{bmatrix} \mathbf{I}_{6-m} & 0 \\ 0 & 0 \end{bmatrix} [\,\mathbf{T}\,|\,\mathbf{T}'\,]^{-1}. \tag{4.3.9}$$

Now we will try to find Ψ_{mo}'s equivalent Ψ_{mj} in the joint space. Firstly, we define $\tilde{\mathbf{J}} : \mathrm{Ker}(\mathbf{J})^\perp \to \mathrm{Image}(\mathbf{J})$, such that

$$\tilde{\mathbf{J}}\dot{\mathbf{q}} = \mathbf{J}\dot{\mathbf{q}},\ \forall\,\dot{\mathbf{q}} \in \mathrm{Ker}(\mathbf{J})^\perp, \tag{4.3.10}$$

and $\tilde{\mathbf{J}}$ is a bijection. Also, we name

$$\mathcal{M}_1 := \mathcal{T} \cap \mathrm{Image}(\mathbf{J}),\ \mathcal{M}_2 := \mathcal{T}' \cap \mathrm{Image}(\mathbf{J}), \tag{4.3.11}$$

$$\mathcal{Q}_1 := \tilde{\mathbf{J}}^{-1}(\mathcal{M}_1),\ \mathcal{Q}_2 := \tilde{\mathbf{J}}^{-1}(\mathcal{M}_2). \tag{4.3.12}$$

$\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are vector spaces, and $\mathrm{Image}(\mathbf{J}) = \mathcal{M}_1 \oplus \mathcal{M}_2$, $\mathrm{Ker}(\mathbf{J})^\perp = \mathcal{Q}_1 \oplus \mathcal{Q}_2$ (see Fig.4.5).

Theorem: The projector Ψ_{mj} mapping $\mathcal{M}^q$ onto $\mathcal{Q}_1 \oplus \mathrm{Ker}(\mathbf{J})$ along $\mathcal{Q}_2$ and its complementary projector Ψ'_{mj} with the property that

1. $\Psi_{mj}(\mathbf{q}_1 + \mathbf{q}_0) = \mathbf{q}_1 + \mathbf{q}_0,\ \forall\,\mathbf{q}_1 \in \mathcal{Q}_1, \forall\,\mathbf{q}_0 \in \mathrm{Ker}(\mathbf{J})$;

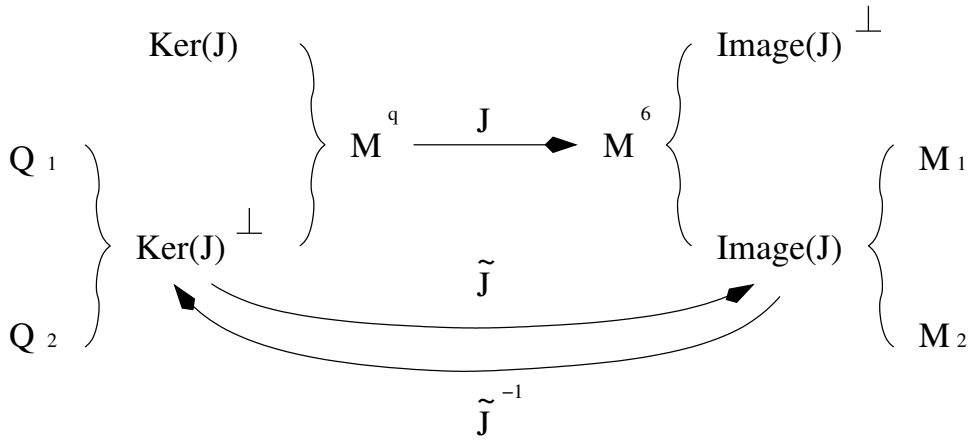


Figure 4.5: Division of $\mathcal{M}^q$ and $\mathcal{M}^6$ for kinematic projector setup

2. $\Psi_{mj}\mathbf{q}_2 = 0, \forall\, \mathbf{q}_2 \in \mathcal{Q}_2$;

3. $\mathbf{I}_q - \Psi_{mj} = \Psi'_{mj}$.

can be given as

$$\Psi_{mj} = \mathbf{J}^\dagger \Psi_{mo} \mathbf{J} + (\mathbf{I}_q - \mathbf{J}^\dagger \mathbf{J}), \tag{4.3.13}$$

$$\Psi'_{mj} = \mathbf{J}^\dagger \Psi'_{mo} \mathbf{J}, \tag{4.3.14}$$

where $\Psi'_{mo} = \mathbf{I}_6 - \Psi_{mo}$ is Ψ_{mo}'s complementary projector, and † stands for the Moore-Penrose pseudo-inverse.

Proof:

$$\begin{aligned} 1)\ \Psi_{mj}(\mathbf{q}_1 + \mathbf{q}_0) &= (\mathbf{J}^\dagger \Psi_{mo} \mathbf{J})\mathbf{q}_1 + (\mathbf{I}_q - \mathbf{J}^\dagger \mathbf{J})\mathbf{q}_1 + \mathbf{q}_0 \\ &= \mathbf{J}^\dagger \mathbf{J}\mathbf{q}_1 + (\mathbf{I}_q - \mathbf{J}^\dagger \mathbf{J})\mathbf{q}_1 + \mathbf{q}_0 = \mathbf{q}_1 + \mathbf{q}_0. \end{aligned}$$

The first step is because $\mathbf{q}_0 \in \text{Ker}(\mathbf{J})$, i.e., $\mathbf{J}\mathbf{q}_0 = 0$. The second step comes from $\mathbf{q}_1 \in \mathcal{Q}_1$, so that $\mathbf{J}\mathbf{q}_1 = \tilde{\mathbf{J}}\mathbf{q}_1 \in \mathcal{M}_1 \in \mathcal{T}$, and $\Psi_{mo}(\mathbf{J}\mathbf{q}_1) = \mathbf{J}\mathbf{q}_1$.

$$\begin{aligned} 2) \qquad \Psi_{mj}\mathbf{q}_2 &= (\mathbf{I}_q - \mathbf{J}^\dagger \mathbf{J})\mathbf{q}_2 \\ &= 0\,. \end{aligned}$$

The first step is the result of $\mathbf{q}_2 \in \mathcal{Q}_2$, then $\Psi_{mo}(\mathbf{J}\mathbf{q}_2) = 0$. The second step is because $\mathbf{q}_2 \in \mathrm{Ker}(\mathbf{J})^{\perp}$ and $(\mathbf{I}_q - \mathbf{J}^{\dagger}\mathbf{J})$ projects $\mathcal{M}_q$ onto $\mathrm{Ker}(\mathbf{J})$ along $\mathrm{Ker}(\mathbf{J})^{\perp}$.

3) $$\begin{aligned} \mathbf{I}_q - \Psi_{mj} &= \mathbf{I}_q - \mathbf{J}^{\dagger}\Psi_{mo}\mathbf{J} - (\mathbf{I}_q - \mathbf{J}^{\dagger}\mathbf{J}) \\ &= \mathbf{J}^{\dagger}(\mathbf{I}_6 - \Psi_{mo})\mathbf{J} \\ &= \mathbf{J}^{\dagger}\Psi'_{mo}\mathbf{J} = \Psi'_{mj}\,. \end{aligned}$$ □

The force kinematic projectors can be constructed in the same way as Ψ_{mj}, Ψ'_{mj}. Here we present their expressions straightaway without proof.

$$\Psi_{fj} = \mathbf{J}^{\top}\Psi_{fo}(\mathbf{J}^{\top})^{\dagger} + \big(\mathbf{I}_q - \mathbf{J}^{\top}(\mathbf{J}^{\top})^{\dagger}\big), \tag{4.3.15}$$

$$\Psi'_{fj} = \mathbf{J}^{\top}\Psi'_{fo}(\mathbf{J}^{\top})^{\dagger}, \tag{4.3.16}$$

where $\Psi_{fo} = [\mathbf{N} \,|\, \mathbf{N}'] \begin{bmatrix} \mathbf{I}_m & 0 \\ 0 & 0 \end{bmatrix} [\mathbf{N} \,|\, \mathbf{N}']^{-1}$, and $\Psi'_{fo} = \mathbf{I}_6 - \Psi_{fo}$.

An and Hollerbach [5] discovered that the traditional hybrid control scheme [54] without the robot dynamics compensation will become unstable in certain cases. In a later paper, Fisher and Mujtaba [27] imputed the instability to the joint space projectors, and proposed a substitutional formulation as

$$\Psi_{mj} = (\mathbf{J}\Psi_{mo})^{\dagger}(\mathbf{J}\Psi_{mo}). \tag{4.3.17}$$

This idea was followed by Doulgeri *et al.* [20].

Obviously, Eq.4.3.17 is not equivalent to Eq.4.3.13, and we suggest to employ the following simple example to make a comparison between those two expressions.

Suppose we have a Cartesian robot as in Fig.2.1 with a spherical wrist installed at its end-tip, then the Jacobian is simply a 6×6 identity matrix. Obviously, in this situation, Ψ_{mo} and Ψ_{mj} should be identical. Plug $\mathbf{J} = \mathbf{I}_6$ into Eqs.4.3.13,4.3.17, we

get

$$\Psi_{mj\,(Ours)} = \mathbf{I}_6^{\dagger}\Psi_{mo}\mathbf{I}_6 + (\mathbf{I}_6 - \mathbf{I}_6^{\dagger}\mathbf{I}_6) = \Psi_{mo}\,, \tag{4.3.18}$$

$$\begin{aligned}\Psi_{mj\,(Fisher's)} &= (\mathbf{I}_6\Psi_{mo})^{\dagger}(\mathbf{I}_6\Psi_{mo}) = \Psi_{mo}^{\dagger}\Psi_{mo} \\ &= \mathbf{V}\Sigma^{\dagger}\mathbf{U}^{\top}\mathbf{U}\Sigma\mathbf{V}^{\top} \\ &= \mathbf{V}\begin{bmatrix} \mathbf{I}_{6-m} & 0 \\ 0 & 0 \end{bmatrix}\mathbf{V}^{\top}.\end{aligned} \tag{4.3.19}$$

The second step in deriving Fisher's projector utilizes a singular value decomposition where $\mathbf{U}$,$\mathbf{V}$ are orthogonal matrices. As we can see, Fisher's formulation drifts away from the original Ψ_{mo} (Eq.4.3.9) and ends up with an artificial orthogonal projector.

Now back to our hybrid control law (Eq.4.3.6). Ψ_{mj} or Ψ_{fj} is included so as to project the position displacement/velocity or force onto its own allowable degrees of freedom. Consequently, we will have

$$\mathbf{N}_j^{\top}\left(\Psi_{mj}\mathbf{K}_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \Psi_{mj}\mathbf{K}_p(\mathbf{q}_d - \mathbf{q})\right) = 0, \tag{4.3.20}$$

$$\left(\Psi_{fj}\,\mathbf{K}_i \int \mathbf{J}^{\top}(\hat{\mathbf{f}}_{cd} - \hat{\mathbf{f}}_c)\,\mathrm{d}t\right) \in \mathcal{N}_j\,. \tag{4.3.21}$$

Moreover, if the desired motion and force trajectories are planned wisely in a sense of fitting the contact model, then

$$\mathbf{N}_j^{\top}\ddot{\mathbf{q}}_d = \dot{\mathbf{N}}_j^{\top}\dot{\mathbf{q}}_d \simeq \dot{\mathbf{N}}_j^{\top}\dot{\mathbf{q}}\,, \tag{4.3.22}$$

$$\mathbf{J}^{\top}\hat{\mathbf{f}}_{cd} \in \mathcal{N}_j\,. \tag{4.3.23}$$

As a result, the overall feedback control input meets the requirement of realizing the dynamical decoupling (Eq.4.3.5), and the system response will ultimately become

$$\begin{aligned}\ddot{\mathbf{q}} &= \ddot{\mathbf{q}}_d + \Psi_{mj}\mathbf{K}_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \Psi_{mj}\mathbf{K}_p(\mathbf{q}_d - \mathbf{q}), \\ \mathbf{J}^{\top}\hat{\mathbf{f}}_c &= \mathbf{J}^{\top}\hat{\mathbf{f}}_{cd} + \Psi_{fj}\,\mathbf{K}_i \int \mathbf{J}^{\top}(\hat{\mathbf{f}}_{cd} - \hat{\mathbf{f}}_c)\,\mathrm{d}t\,.\end{aligned} \tag{4.3.24}$$

Define $\mathbf{e}_m := \mathbf{q}_d - \mathbf{q}$ and $\mathbf{e}_f := \mathbf{J}^\top \hat{\mathbf{f}}_{cd} - \mathbf{J}^\top \hat{\mathbf{f}}_c$, the above equation can further be rewritten as

$$\begin{aligned} \ddot{\mathbf{e}}_m + \Psi_{mj}\mathbf{K}_v\dot{\mathbf{e}}_m + \Psi_{mj}\mathbf{K}_p\mathbf{e}_m &= 0, \\ \mathbf{e}_f + \Psi_{fj}\mathbf{K}_i \int \mathbf{e}_f \, \mathrm{d}t &= 0. \end{aligned} \tag{4.3.25}$$

As long as $\Psi_{mj}\mathbf{K}_p$, $\Psi_{mj}\mathbf{K}_v$ and $\Psi_{fj}\mathbf{K}_i$ are chosen to be positive definite, both motion and force subsystems will enjoy asymptotic stability.

4.4 Simulation and Future Research Directions

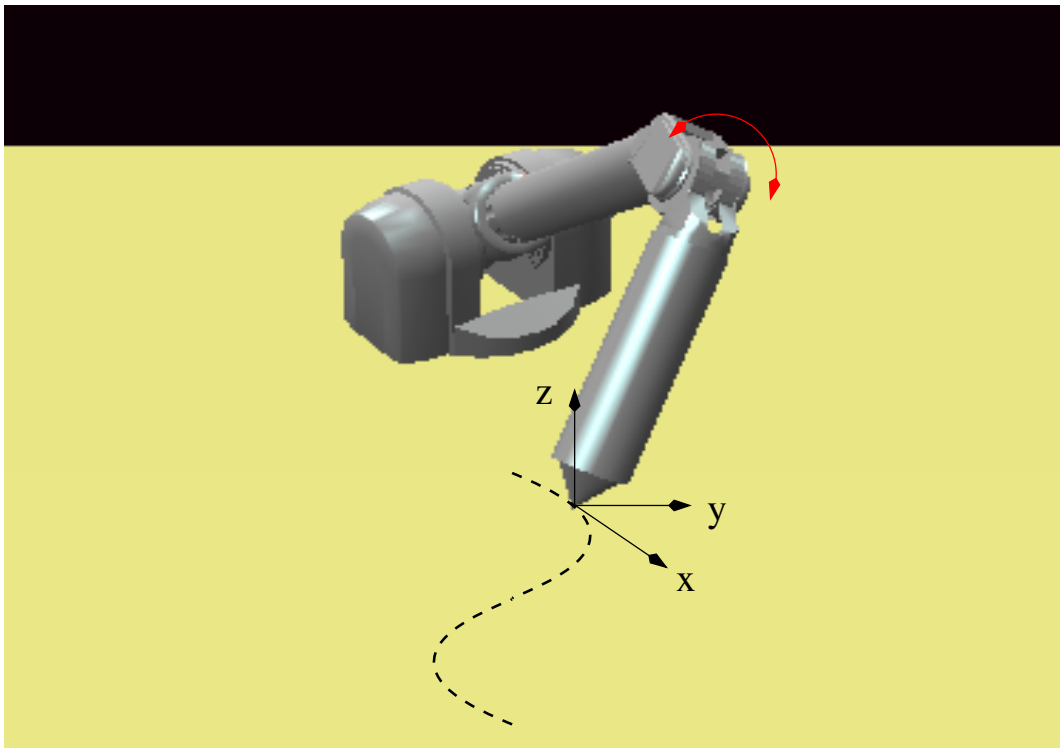


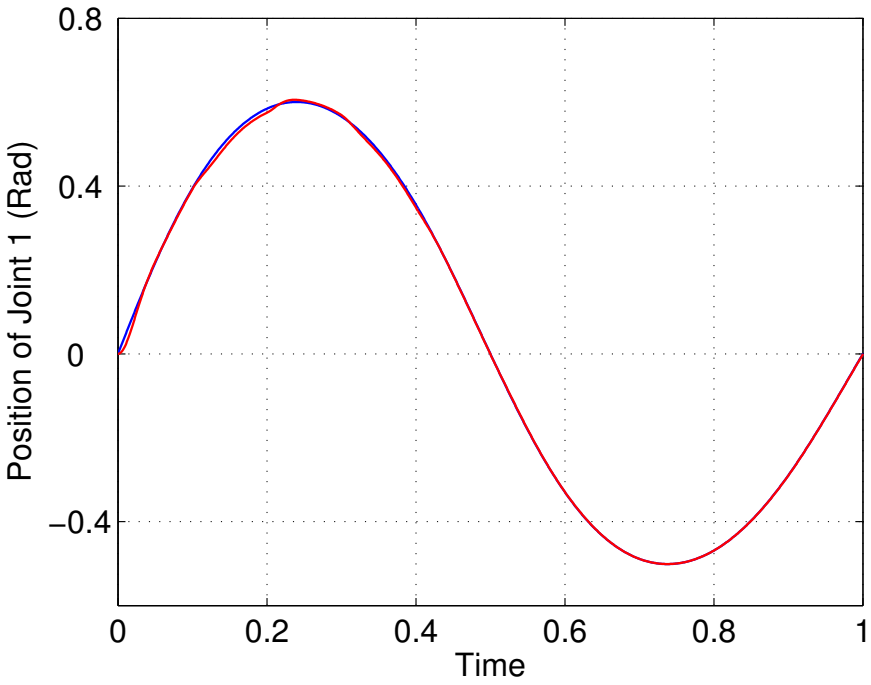
Figure 4.6: Compliant motion task for WAM's simulation

WAM is a 4-joint robot manipulator with human-like kinematics (see Fig.2.8).

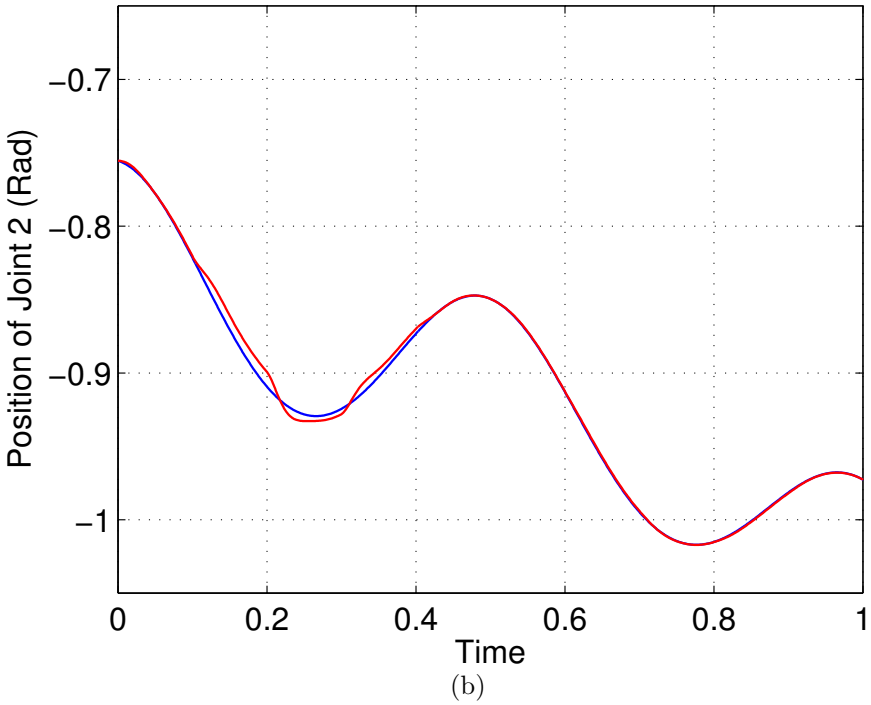
Now suppose WAM is required to move its end-effector on a horizontal plane (see Fig.4.6), then such a single-point contact will divide WAM's workspace into a combination of 3-DOF motion and 1-DOF force, i.e., in the compliance (task) frame [11] $\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$, 2-DOF translation in the x-y plane plus 1-DOF rotation; and a linear contact force along z-axis. For example, the compliant motion task illustrated in Fig.4.6 makes WAM's end-tip to traverse an S-shape path with its body swinging back and forth; in the meantime, a fluctuant magnitude contact force can possibly be applied against the environmental surface as well. A similar task of WAM drawing the NICTA logo is demonstrated by a short video clip *NictaLogo.mpg* contained in the attached CD.

A simulation on WAM of applying our joint space hybrid control algorithm for the proposed single-point contact task has been performed by using *Matlab 7.0* (The MathWorks, Inc. 2004), whose results are shown in Figs.4.7-4.9. The relatively poor tracking performance of the force controller as we see in Figs.4.7,4.8 and 4.9 is intrinsically due to the varied characteristics of the motion and force subsystems, i.e., disturbances of joint drive torques (simulated as white noises here) have more direct destructive effect on the eventual output of the contact force than that of the joint positions.

Furthermore, another reason accounting for the jagged force response is the environment has been assumed as fixed in the closed-loop dynamics simulation and hybrid control algorithm here. In the future, we wish to incorporate the contact dynamics as it is a critical factor for the controller's capacity to reject the force disturbance [55]. Also, we will be interested in looking at the stability issue due to the transition between free and constrained motion, e.g., to reduce the influence of the impact force

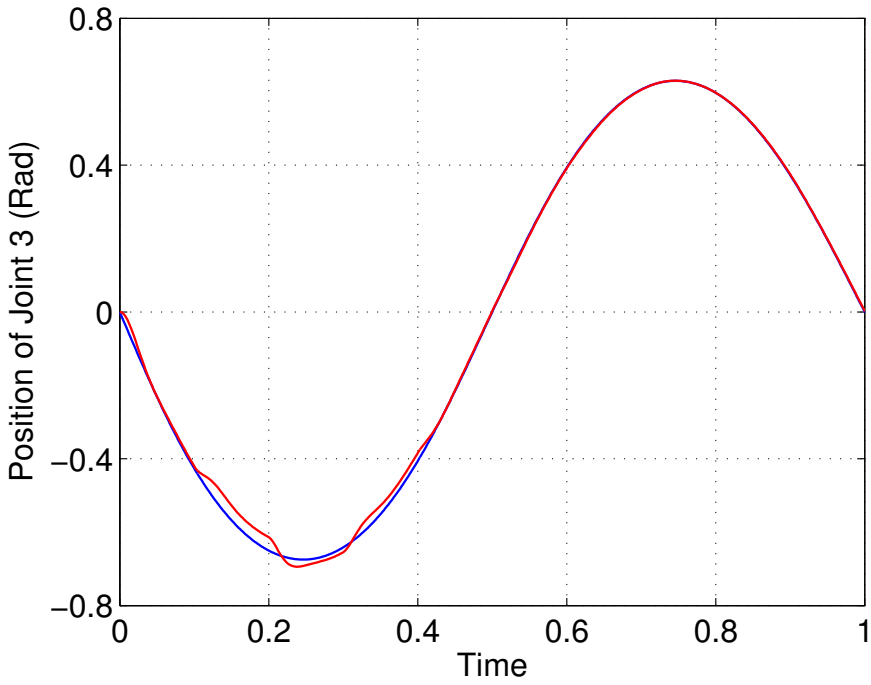


(a)

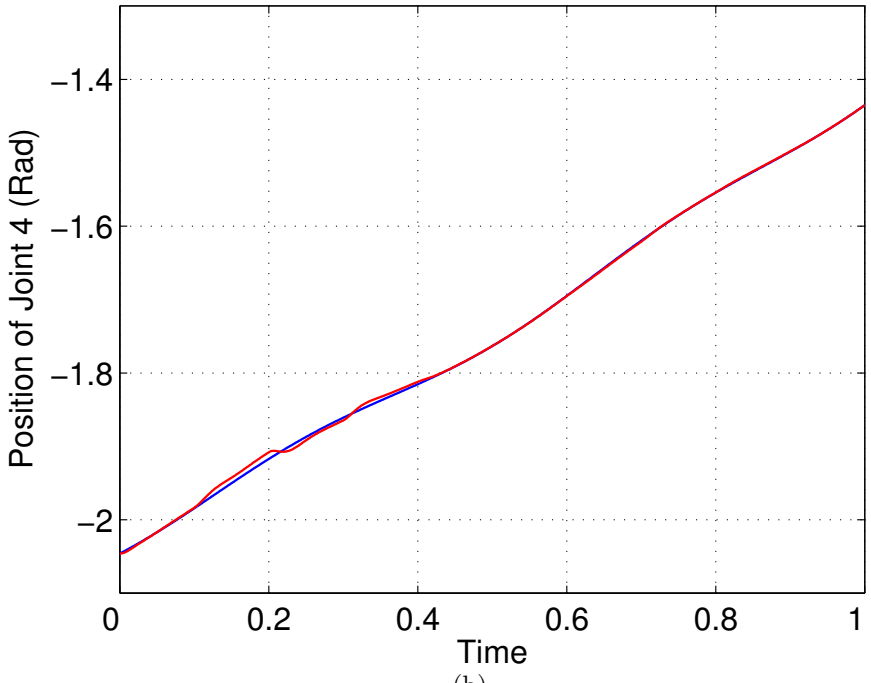


(b)

Figure 4.7: Trajectory of joint 1: a), and 2: b). Blue: desired; Red: actual



(a)



(b)

Figure 4.8: Trajectory of joint 3: a), and 4: b). Blue: desired; Red: actual

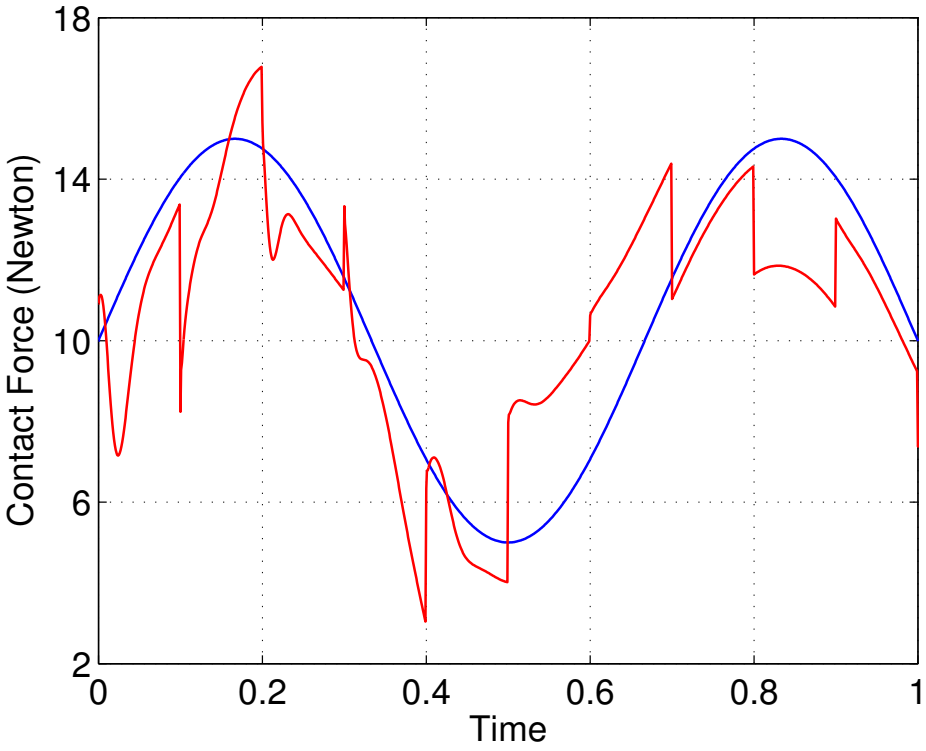


Figure 4.9: Trajectory of contact force. Blue: desired; Red: actual

on the dynamical instability etc.

4.5 Proposed Hybrid Motion/Force Control Experiment on WAM

4.5.1 RSISE Experimental Robot Manipulator: WAM

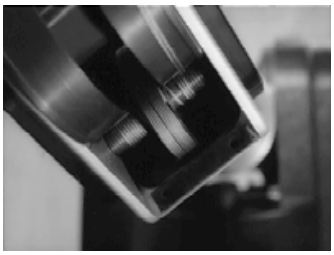
As mentioned in Section 2.5 and Section 4.4, WAM (**W**hole **A**rm **M**anipulator) is a 4-DOF experimental robot manipulator (Fig.2.8), which was purchased in 1997 from the Barrett Technology Inc, USA. One distinguished design of the robot is its advanced cable-drive system (Fig.4.10) which yields several remarkable characteristics such as zero backlash, low friction, high bandwidth, high force fidelity, and so on. With these advantages, WAM can be considered as a suitable platform for research on advanced force control [6].

Two end-effectors, the "Dumbbell" and "Pawn" probes (Fig.4.10), have been made for single or dual contact force control experiments on WAM. The main body of the "dumbbell" is a Nylon beam which is designed for absorbing a majority of impact force when a hard contacts takes place. Its dimensions (length and radius) are optimized with regard to small deflection and good distinguishablity of the two contact forces, by using *MDSolids 1.5* (Timothy A. Philpot, Murray State University. 1995). The end-tip spheres are chosen to be made of *Teflon* (DuPont), since this material has very good rigidity and exceptionally low frictional coefficient.

The robotic system is equipped with several JR3 6-DOF force sensors (the blue metal block with JR3 logo in Fig.4.10), which are capable of measuring force and torque on three orthogonal directions. By the electronic system contained within

"Dumbbell" Probe

Cabled Transmission



"Pawn" Probe

Figure 4.10: WAM's cable-drive transmission system, and its custom-made end-effectors

the sensor, the strain gage signal is amplified and converted to digital data, which can further be accessible from computers via the JR3 communication boards with interfaces of ISA or PCI. The drivers for both JR3 ISA-bus and PCI-bus adaptor boards have been written under VxWorks, a real-time operating system which will be introduced next. The drivers' source code in C++ is available on request.

WAM is controlled by a 500MHz Pentium III PC named *Nozomi* at a sampling rate of 1000Hz. *Nozomi* runs a real-time operating system, VxWorks 5.5 (Wind River Systems, Inc. 2001). One distinguishing feature of VxWorks is that its development is done on a "host" machine running UNIX or Windows, while cross-compiling target software to run on various "target" CPU architectures.

Fig.4.11 shows the overall structure of the WAM robotic system and the general

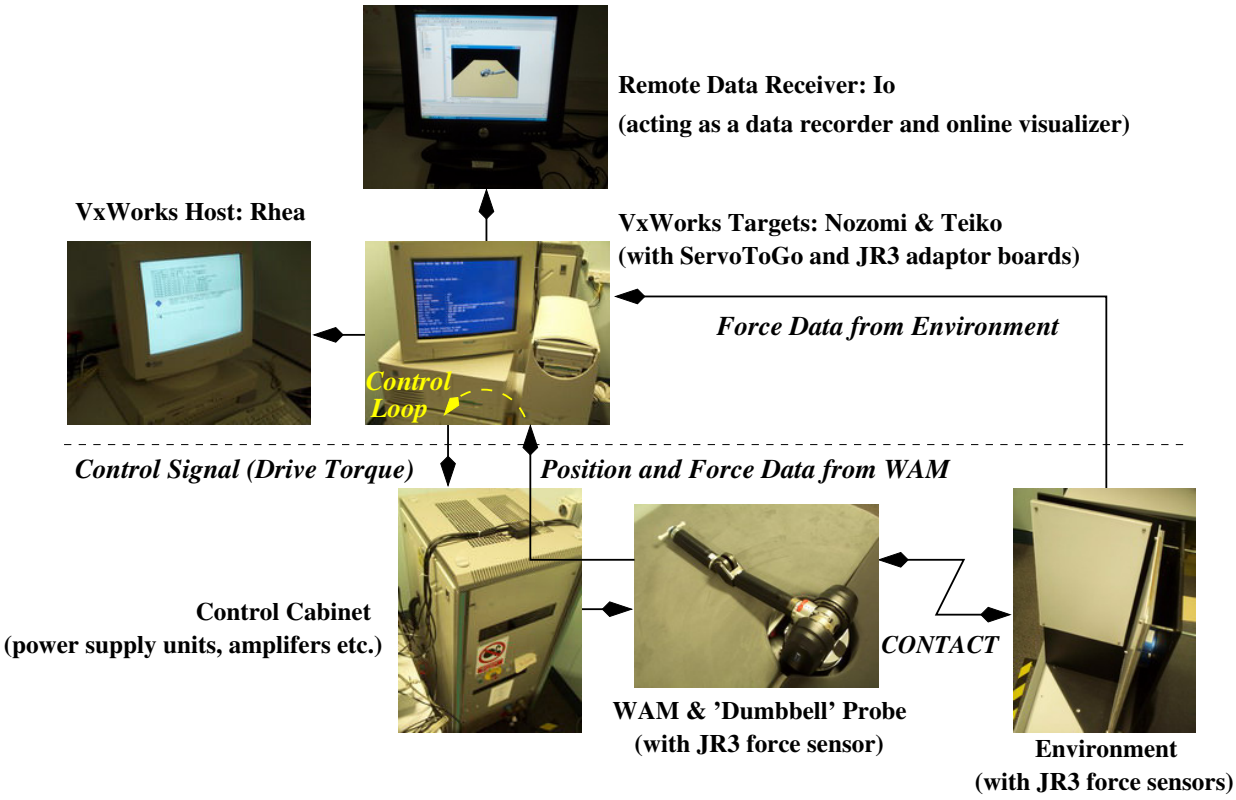


Figure 4.11: Overall hardward/software structure of WAM and peripheral devices for the proposed dual-contact experiment

dataflow for the proposed dual-contact force control experiment. The VxWorks target *Nozomi* runs the major control software (developed in C++), and is supported by the host computer *Rhea*, a Sun-Solaris desktop where VxWorks binaries and software IDE Tornado reside. A motion control board (*ServoToGo 8-axis ISA-bus Servo IO card*, Servo To Go, Inc.) and a JR3 ISA-bus communication board are installed in *Nozomi*, interfacing the incoming data of WAM's motion and applied force, and the motor-torque control signals calculated by the control loop. For verification purposes, two extra JR3 6-DOF force sensors are mounted in the "V-shape" environment, and the

contact force data goes into two JR3 PCI-bus boards installed in another VxWorks Target PC *Teiko*. Furthermore, an online Java 3D visualizer demonstrating WAM's motion operates on a Windows PC *Io*, which is connected with *Nozomi* via the local ethernet.

4.5.2 Purposes and Procedure of the Suggested Experiment

The purposes of the experiment is to further verify the claims made in the theory sections of closed-loop dynamics analysis and joint-space control law design. These claims can be summarized as follows.

1. Applicability on general robots;

   As explained in the very beginning of this chapter, for robot manipulators already subject to kinematic constrains (e.g., the 4-DOF WAM), our joint-space control algorithm, even may not be the only available choice, has several advantages over the conventional operational-space ones. The trajectory optimization scheme in Chapter 2 can be applied here for performing the corresponding joint-level motion planning.

2. Dynamic decoupling of force and motion spaces.

   We have derived in Section 4.3 the decoupling between the system responses ($\ddot{\mathbf{q}}$ and $\mathbf{J}^\top \hat{f}_c$), provided that 1) the desired trajectories ($\ddot{\mathbf{q}}_d$ and $\mathbf{J}^\top \hat{f}_{cd}$) are planned carefully in terms of fully complying with the contact task, which can be guaranteed if our two-step path planning scheme is applied; and 2) the robot's dynamics is sufficiently compensated in the control law.

We intend to apply our joint-space control algorithm on WAM to perform a multi-contact 4-DOF compliant motion task (Fig.4.12). The dumbbell end-effector will be making contact with two perpendicular walls, also moving vertically along with an in/outward twist. The contact can be considered as frictionless, since both the spheres and the plates are made of *Teflon* (some lubricant may also be applied during the experimentation if necessary).

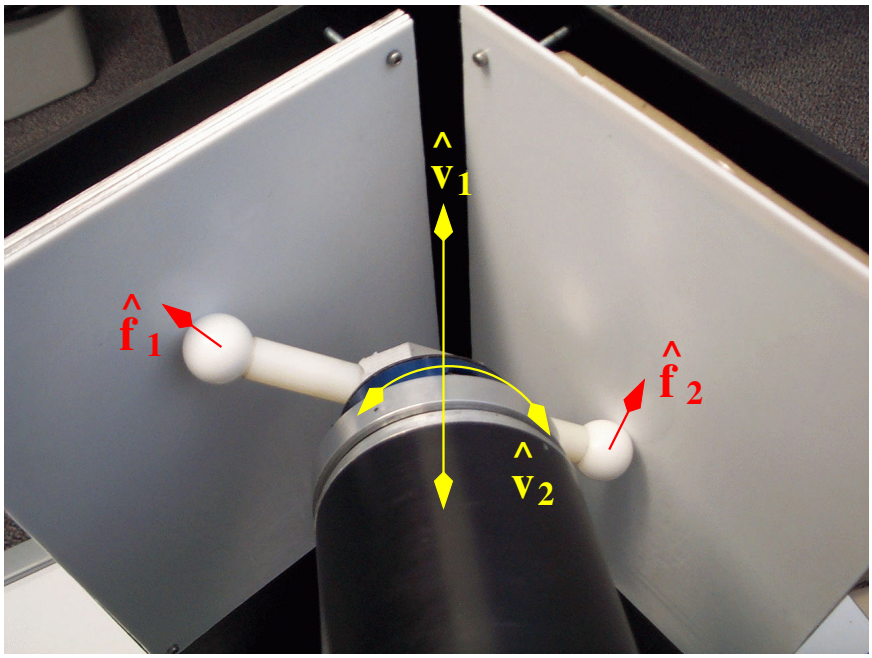


Figure 4.12: Proposed dual-point contact hybrid motion/force control experiment for WAM

The subtlety of modelling the dual-contact action is its time-variance, namely it can not be formulated by the conventional "Task Frame Formalism" [11]. The kinematic projectors Ψ_{mj} and Ψ_{fj} based on dual motion/force spaces are delibrately designed on a different theoretical basis as the well-known "Selection Matrices" [54],

and are able to cope with general contact modelling such as the situation here. Moreover, the kinematic stability issue arising from joint-space filtering will be tested in the experiment as well.

Values of the hybrid motion/force variables can be chosen as

$$\begin{aligned} z &= z_{\text{offset}} + z_{\text{mag}} \sin(z_{\text{freq}}\, t), \\ \theta &= \theta_{\text{offset}} + \theta_{\text{mag}} \sin(\theta_{\text{freq}}\, t), \\ f_1 &= f_{1\text{offset}} + f_{1\text{mag}} \sin(f_{1\text{freq}}\, t), \\ f_2 &= f_{2\text{offset}}. \end{aligned} \tag{4.5.1}$$

where z is the vertical position of the center of rod, θ indicates the rotation angle of the in/outward twist, and $z_{\text{freq}} = \theta_{\text{freq}} = 1.25\, f_{1\text{freq}}$. Note that the force control performance of f_1 and f_2 will be independently measured by two JR3 force sensors installed in the "V-shape" environment. Also, the purpose for setting z/θ and f_1 under different frequencies is to later analyze the dynamical decoupling of motion/force sub-systems by looking at the cross-coupling between the force/motion reference trajectories and the actual tracking errors.

Due to several hardware problems of the robot, the suggested experiment has not yet been implemented on WAM. Some completed experimental work is recorded in Appendix C.

4.6 Summary

This chapter presents a new scheme for the hybrid motion/force control of robot manipulators. Our contribution is to redo the formulation in the joint space so as to make the closed-loop dynamics analysis and the control algorithm design applicable

to general robots. Also, the joint space control law allows us to implement different suitable control gains for each robot joints, which we think is a more practical and robust approach for real robotic systems whose disturbances are mainly originated at the joint level. An example of the compliant motion control for a 4-DOF robot WAM along with some simulation results are presented, and a hybrid motion/force control experiment is proposed at the end of the chapter.

Chapter 5

Conclusion and Outlook

5.1 Conclusion

The major goal of this thesis is to bring a new thought of robot manipulator's compliant motion control concerning both planning and control aspects. Due to the natural easiness of describing motion constraints and the simplified treatment of robot's kinematic/dynamic models, operational-space is taken for granted as the default coordinate frame of robot's hybrid motion/force control formulation, especially after Khatib's historic work [37]. However, we believe joint-space is in fact a better choice in terms of global optimization, robust performance, and general applicability. Apart from these desirable benefits, the corresponding price of introducing joint-space processing is its computational complicity, as the robot manipulator's kinematic or dynamic features are now considered in reference trajectory and control algorithm designs. To cope with such technical difficulties, we have proposed numerical trajectory-optimization, interpolation, and control algorithms, which have been presented in the previous chapters.

The mathematical foundations of robotics research results recorded in this thesis can be traced back to our two major theoretical contributions in variational calculus

and differential geometry. More specifically, they are

1. A convergence-proved iterative algorithm solving for variational problems subject to boundary conditions with Lagrangian functions containing up to second-order derivatives;

2. A smooth interpolation algorithm on manifold combining rolling and wrapping with the pull back/push forward technique.

Based on these mathematical research achievements, the contributions on the engineering aspect of this thesis mainly focus on three topics of robot manipulator's compliant motion control, which are

1. **Numerical trajectory optimization**;

   A two-step calculation scheme planning optimal joint trajectory for robot subject to equality and inequality constraints has been proposed. The motion planner aims to some comprehensive optimization objective with regard to both joint trajectory and its resulting end-effector path as well.

2. **Smooth interpolation of orientation**;

   A novel procedure to calculate smooth interpolation curves of the rotation group SO_3 has been investigated. The suggested approach has two remarkable features appropriate for real-time applications: 1) coordinate-free; 2) resulting curves given in closed form.

3. **Joint-space formulation of dynamics and control**.

The general contact geometry and robot's closed-loop dynamics have been reformulated in joint space, moreover, a corresponding joint-space hybrid motion/force control algorithm has been proposed. Some theoretical discoveries with regard to system responses include dynamical decoupling and asymptotic stability.

5.2 Future Research Directions

In this thesis, a new methodology of robot's complaint motion control is presented, and our research results on several aspects of this theme are shown. Future research projects are being considered either to eliminate limitations of the algorithms we proposed or to apply our theoretically idealized research outcomes on practical robotics applications.

On the mathematical part, firstly, we intend to expand the scope of the optimization objective of our Newton-like method for constrained variational problems. In more detail, the Lagrangian function of the improved iterative algorithm now can be any general function of position, velocity, and acceleration. Moreover, the resulting trajectory may be subject to non-holonomic motion constraints. Therefore, instead of **P2.5**, we are further interested to solve

P5.1

$$\min_{\mathbf{y}} \; J(\mathbf{y}) = \int_{t_0}^{t_n} L\big(t, \mathbf{y}(t), \dot{\mathbf{y}}(t), \ddot{\mathbf{y}}(t)\big) \; \mathrm{d}t \tag{5.2.1}$$

with respect to the class of sufficiently smooth functions $\mathbf{y} : [t_0, t_n] \to \mathbb{R}^p$, $t \mapsto \mathbf{y}(t)$ satisfying the boundary conditions

$$\mathbf{y}(t_0) = \mathbf{y}_0, \; \mathbf{y}(t_n) = \mathbf{y}_n, \; \dot{\mathbf{y}}(t_0) = \xi_0, \; \dot{\mathbf{y}}(t_n) = \xi_n, \tag{5.2.2}$$

and the holonomic constraint

$$C\big(\mathbf{y}(t)\big) = 0,\ \forall t \in [t_0, t_n], \tag{5.2.3}$$

or the non-holonomic constraint

$$C\big(\dot{\mathbf{y}}(t)\big) = 0,\ \forall t \in [t_0, t_n]. \tag{5.2.4}$$

As we mentioned before, the key technical difficulty is to find a unified and numerical consistent approximation scheme for $\dot{\mathbf{y}}$ and $\ddot{\mathbf{y}}$.

Secondly, a thorough study will be performed to investigate the influences on SO_3 curve design due to applying various local diffeomorphisms in smooth interpolation by rolling and wrapping, and to compare different interpolation approaches with respect to the performance index of minimum angular acceleration, i.e., $\int_{t_0}^{t_n} \ddot{\gamma}(t)^\top \ddot{\gamma}(t)\ \mathrm{d}t$. Furthermore, motivated by practical applications such as optimal control of satellite's attitude, we also plan to modify our SO_3 interpolation algorithm incorporating dynamics consideration. Based on **P3.1**, the new interpolation task can be formulated as

P5.2 Find a C^2-smooth curve $\gamma : [0, T] \to SO_3$ interpolating a set of given intermediate points $\mathbf{R}_k \in SO_3$, such that

$$\gamma(t_k) = \mathbf{R}_k,\ 1 \le k \le n-1, \tag{5.2.5}$$

where

$$0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = T, \tag{5.2.6}$$

subject to the boundary conditions

$$\begin{aligned} \gamma(0) &= \mathbf{R}_0 \in SO_3, \\ \gamma(T) &= \mathbf{R}_n \in SO_3, \\ \dot{\gamma}(0) &= \xi_0 \in T_{\mathbf{R}_0} SO_3, \\ \dot{\gamma}(T) &= \xi_n \in T_{\mathbf{R}_n} SO_3. \end{aligned} \tag{5.2.7}$$

Now we suggest the revised interpolation process aims to design the resulting curve towards the following optimization objective

$$\min_{\gamma} \int_{t_0}^{t_n} \ddot{\gamma}(t)^\top \mathbf{I}\, \ddot{\gamma}(t)\, \mathrm{d}t, \tag{5.2.8}$$

where $\mathbf{I}$ is the moment of inertia.

Since the metric has been redefined, the rolling map as well as rolling curve (usually the geodesics) need to be modified correspondingly.

Back to the robotics aspect, the very first thing to consider for our future research is definitely the hybrid motion/force control experiment, whose procedure and purposes have been proposed in Section 4.5 already. Apart from those completed work recorded in Appendix C, other critical issues that will probably be encountered in the practical experimentation are

1. Torque ripple compensation;

   Further attempts should be made to properly identify the motor torque ripples of WAM. If the feed-forward compensation method again turns out to be unsuccessful eventually, we probably have to think about introducing some inner loop as an alternative way to discard the motor torque ripple disturbance. Usually,

this lower-level loop is set to be 3 to 4 times faster than the main control loop and it will involve installation of motor torque sensing devices.

2. Control law design;

   We have presented the asymptotic stability analysis of our joint-space hybrid motion/force control algorithm in Section 4.3.2. Nevertheless, proper PID coefficients need to be chosen to achieve satisfactory tracking performance. From control theory's point of view, tuning PID gains here is exactly a state-feedback pole-placement problem. We will consider techniques which are reported to be successfully implemented on robot force control applications in the literature, such as "Ackermann's formula" [14],[53].

3. Force sensing.

   As robots will be making contact as well as moving in typical hybrid motion/force control procedures, noisy force data becomes another very critical component for system's stability in practice. Therefore, implementing some online filter is very necessary for rejecting the high-frequency noises. The observer design we are considering is the AOB (active observer), which was invented by Cortesão [13] and has been verified to be very effective in robot compliant motion control experimentations [14],[53].

Appendix A

Proof of Convergence

The proof of convergence here shows the correspondence between **P2.5** and **P2.6**, in other words, as h goes to 0, the solution of the discretized system of **P2.6** will converge to the solution of the Euler-Lagrange equation system of **P2.5**.

A.1 Solution of the Discretized System

Firstly, we will make some analysis on the solution of the discretized system. As we know, the necessary condition of the solution of **P2.6** is its cost function $\mathbf{L}$ stays at some stationary value, namely $\nabla\mathbf{L}(\mathbf{Q}) = 0$. Also, this is the result that the Newton's method (**A2.1** or **A2.2**) will eventually converge to (if it can converge).

Expand $\nabla\mathbf{L}(\mathbf{Q}) = 0$, we will get

$$\begin{aligned}
&\tilde{\mathcal{A}}\,\frac{\partial L_1}{\partial \mathbf{y}_j}\big(\mathbf{P}^{*n}_h(t), \mathcal{A}\tilde{\mathbf{Q}}_\mathbf{y}, \mathcal{B}\tilde{\mathbf{Q}}_\mathbf{y}\big) + \tilde{\mathcal{B}}\,\frac{\partial L_1}{\partial \dot{\mathbf{y}}_j}\big(\mathbf{P}^{*n}_h(t), \mathcal{A}\tilde{\mathbf{Q}}_\mathbf{y}, \mathcal{B}\tilde{\mathbf{Q}}_\mathbf{y}\big)\\
&+\frac{\partial L_2}{\partial \mathbf{y}_j}\big(\mathbf{P}^{n-1}_h(t), \mathbf{Q}_\mathbf{y}, \tilde{\mathcal{C}}\mathbf{Q}_\mathbf{y}\big) + \tilde{\mathcal{C}}\,\frac{\partial L_2}{\partial \ddot{\mathbf{y}}_j}\big(\mathbf{P}^{n+1}_h(t), \tilde{\mathbf{Q}}_\mathbf{y}, \mathcal{C}\tilde{\mathbf{Q}}_\mathbf{y} + \mathcal{D}\big)\\
&\qquad + \mathbf{Q}_\mu\,\frac{\partial C}{\partial \mathbf{y}_j}(\mathbf{Q}_\mathbf{y}) \quad = \quad 0 \in \mathbb{R}^{n-1},\ j \in \{1, \cdots, p\}, \qquad \text{(A.1.1)}\\
&C(\mathbf{Q}_\mathbf{y}) \quad = \quad 0 \in \mathbb{R}^{n-1}, \qquad \text{(A.1.2)}
\end{aligned}$$

where

$$\tilde{\mathbf{Q}}_{\mathbf{y}} = \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{Q}_y \\ \mathbf{y}_n \end{bmatrix} \in \mathbb{R}^{(n+1)\times p}, \tag{A.1.3}$$

$$\tilde{\mathcal{A}} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & & & \\ & \frac{1}{2} & \frac{1}{2} & & \\ & & \ddots & \ddots & \\ & & & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \in \mathbb{R}^{(n-1)\times n}, \tag{A.1.4}$$

$$\tilde{\mathcal{B}} = \begin{bmatrix} \frac{1}{h} & -\frac{1}{h} & & & \\ & \frac{1}{h} & -\frac{1}{h} & & \\ & & \ddots & \ddots & \\ & & & \frac{1}{h} & -\frac{1}{h} \end{bmatrix} \in \mathbb{R}^{(n-1)\times n}, \tag{A.1.5}$$

$$\tilde{\mathcal{C}} = \begin{bmatrix} \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \end{bmatrix} \in \mathbb{R}^{(n-1)\times(n+1)}. \tag{A.1.6}$$

A.2 Convergence Proof

In this section, we will prove that as h goes to 0, the solution of the discretized system (Eqs. A.1.1, A.1.2) will converge to the solution of the Euler-Lagrange equation system (Eq. 2.3.5). The proof here follows the general principle of proving convergence for difference methods (Page 195-199, [68]).

Firstly, we make two assumptions, which will be respectively used to prove lemma A.2.1 and theorem A.2.2.

1. Existence and uniqueness: Eq. 2.3.5 has a unique solution $\{\mathbf{y}(t), \mu(t)\}$ and Eqs. A.1.2, A.1.1 have a unique solution $\{\mathbf{Y}_h \in \mathbb{R}^{(n-1)\times p}, \mathbf{M}_h \in \mathbb{R}^{n-1}\}$.

2. Stability: there exist positive constants s and γ such that

$$\| [\mathbf{V} - \mathbf{U}] \|^s \leq \gamma \|\mathbf{E}_h(\mathbf{V}) - \mathbf{E}_h(\mathbf{U})\|, \tag{A.2.1}$$

where $\mathbf{V}, \mathbf{U} \in \mathbb{R}^{(n-1)\times(p+1)}$, and the operator $\mathbf{E}_h$ is defined as

$$\begin{aligned} \mathbf{E}_h : \quad & \mathbb{R}^{(n-1)\times(p+1)} && \to \mathbb{R}^{n-1}, \\ & \begin{bmatrix} y_{1,1} & \cdots & y_{1,p} & \mu_1 \\ \vdots & \ddots & \vdots & \vdots \\ y_{n-1,1} & \cdots & y_{n-1,p} & \mu_{n-1} \end{bmatrix} && \mapsto \text{LHS of Eq. A.1.1.} \end{aligned} \tag{A.2.2}$$

Next, we will prove lemma A.2.1 which is the key step to approach the finial convergence result.

Lemma A.2.1. *With the approximation schemes Eqs. 2.3.17, 2.3.18, 2.3.19 and assumption 1, there exists a positive constant α such that*

$$\left\| \boldsymbol{E}_h\Big(\begin{bmatrix} \boldsymbol{P}_h^{n-1}\big(\boldsymbol{y}(t)\big) & \boldsymbol{P}_h^{n-1}\big(\mu(t)\big) \end{bmatrix} \Big) \right\| \leq \alpha h \tag{A.2.3}$$

for sufficiently small h.

Proof.

Step 1: Show the consistency of the approximation schemes.

$$\mathcal{A}\,\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big) = \mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big) + \phi(h)h^2, \tag{A.2.4}$$

$$\tilde{\mathcal{A}}\,\mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big) = \mathbf{P}_h^{n-1}\big(\mathbf{y}(t)\big) + \tilde{\phi}(h)h^2, \tag{A.2.5}$$

$$\mathcal{B}\,\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big) = \mathbf{P}^{*n}_h\big(\dot{\mathbf{y}}(t)\big) + \varphi(h)h^2, \tag{A.2.6}$$

$$-\tilde{\mathcal{B}}\,\mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big) = \mathbf{P}_h^{n-1}\big(\dot{\mathbf{y}}(t)\big) + \tilde{\varphi}(h)h^2, \tag{A.2.7}$$

$$\mathcal{C}\,\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big) + \mathcal{D} = \mathbf{P}_h^{n+1}\big(\ddot{\mathbf{y}}(t)\big) + \psi_1(h)h + \psi_2(h)h^2, \tag{A.2.8}$$

$$\tilde{\mathcal{C}}\,\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big) = \mathbf{P}_h^{n-1}\big(\ddot{\mathbf{y}}(t)\big) + \tilde{\psi}_2(h)h^2, \tag{A.2.9}$$

where

$$\phi(h) = \begin{bmatrix} \cdots & \ddot{\mathbf{y}}_j\big((0+\theta_{1j1\frac{1}{2}})h\big) + \ddot{\mathbf{y}}_j\big((\frac{1}{2}+\theta_{1j2\frac{1}{2}})h\big) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \ddot{\mathbf{y}}_j\big((k-1+\theta_{kj1\frac{1}{2}})h\big) + \ddot{\mathbf{y}}_j\big((k-\frac{1}{2}+\theta_{kj2\frac{1}{2}})h\big) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \ddot{\mathbf{y}}_j\big((n-1+\theta_{nj1\frac{1}{2}})h\big) + \ddot{\mathbf{y}}_j\big((n-\frac{1}{2}+\theta_{nj2\frac{1}{2}})h\big) & \cdots \end{bmatrix} /16 \in \mathbb{R}^{n\times p},$$

$$\tilde{\phi}(h) = \begin{bmatrix} \cdots & \ddot{\mathbf{y}}_j\big((\frac{1}{2}+\tilde{\theta}_{1j1\frac{1}{2}})h\big) + \ddot{\mathbf{y}}_j\big((1+\tilde{\theta}_{1j2\frac{1}{2}})h\big) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \ddot{\mathbf{y}}_j\big((k-\frac{1}{2}+\tilde{\theta}_{kj1\frac{1}{2}})h\big) + \ddot{\mathbf{y}}_j\big((k+\tilde{\theta}_{kj2\frac{1}{2}})h\big) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \ddot{\mathbf{y}}_j\big((n-1-\frac{1}{2}+\tilde{\theta}_{nj1\frac{1}{2}})h\big) + \ddot{\mathbf{y}}_j\big((n-1+\tilde{\theta}_{nj2\frac{1}{2}})h\big) & \cdots \end{bmatrix} /16 \in \mathbb{R}^{(n-1)\times p},$$

$$\varphi(h) = \begin{bmatrix} \cdots & \dddot{\mathbf{y}}_j\big((0+\vartheta_{1j1\frac{1}{2}})h\big) + \dddot{\mathbf{y}}_j\big((\frac{1}{2}+\vartheta_{1j2\frac{1}{2}})h\big) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \dddot{\mathbf{y}}_j\big((k-1+\vartheta_{kj1\frac{1}{2}})h\big) + \dddot{\mathbf{y}}_j\big((k-\frac{1}{2}+\vartheta_{kj2\frac{1}{2}})h\big) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \dddot{\mathbf{y}}_j\big((n-1+\vartheta_{nj1\frac{1}{2}})h\big) + \dddot{\mathbf{y}}_j\big((n-\frac{1}{2}+\vartheta_{nj2\frac{1}{2}})h\big) & \cdots \end{bmatrix} /48 \in \mathbb{R}^{n\times p},$$

$$\tilde{\varphi}(h) = \begin{bmatrix} \cdots & \dddot{\mathbf{y}}_j\big((\frac{1}{2}+\tilde{\vartheta}_{1j1\frac{1}{2}})h\big) + \dddot{\mathbf{y}}_j\big((1+\tilde{\vartheta}_{1j2\frac{1}{2}})h\big) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \dddot{\mathbf{y}}_j\big((k-\frac{1}{2}+\tilde{\vartheta}_{kj1\frac{1}{2}})h\big) + \dddot{\mathbf{y}}_j\big((k+\tilde{\vartheta}_{kj2\frac{1}{2}})h\big) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \dddot{\mathbf{y}}_j\big((n-1-\frac{1}{2}+\tilde{\vartheta}_{nj1\frac{1}{2}})h\big) + \dddot{\mathbf{y}}_j\big((n-1+\tilde{\vartheta}_{nj2\frac{1}{2}})h\big) & \cdots \end{bmatrix} /48 \in \mathbb{R}^{(n-1)\times p},$$

$$\psi_1(h) = \begin{bmatrix} \cdots & \dddot{\mathbf{y}}_j\big((0+\iota_{1j})h\big) & \cdots \\ \cdots & 0 & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & 0 & \cdots \\ \cdots & \dddot{\mathbf{y}}_j\big((n-1+\iota_{(n-1)j})h\big) & \cdots \end{bmatrix} /3 \in \mathbb{R}^{(n+1)\times p},$$

$$\psi_2(h) = \begin{bmatrix} \cdots & 0 & \cdots \\ \cdots & \ddddot{\mathbf{y}}_j\big((0+\lambda_{2j1})h\big) + \ddddot{\mathbf{y}}_j\big((1+\lambda_{2j2})h\big) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \ddddot{\mathbf{y}}_j\big((n-2+\lambda_{(n-1)j1})h\big) + \ddddot{\mathbf{y}}_j\big((n-1+\lambda_{(n-2)j2})h\big) & \cdots \\ \cdots & 0 & \cdots \end{bmatrix} /24 \in \mathbb{R}^{(n+1)\times p},$$

$$\tilde{\psi}_2(h) = \begin{bmatrix} \cdots & \ddddot{\mathbf{y}}_j\big((0+\tilde{\lambda}_{2j1})h\big) + \ddddot{\mathbf{y}}_j\big((1+\tilde{\lambda}_{2j2})h\big) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \ddddot{\mathbf{y}}_j\big((n-2+\tilde{\lambda}_{(n-1)j1})h\big) + \ddddot{\mathbf{y}}_j\big((n-1+\tilde{\lambda}_{(n-2)j2})h\big) & \cdots \end{bmatrix} /24 \in \mathbb{R}^{(n-1)\times p},$$

$$\theta_{kj1}, \theta_{kj2}, \tilde{\theta}_{kj1}, \tilde{\theta}_{kj2}, \vartheta_{jk1}, \vartheta_{kj2}, \tilde{\vartheta}_{kj1}, \tilde{\vartheta}_{kj2}, \iota_{1j}, \iota_{(n-1)j}, \lambda_{kj1}, \lambda_{kj2}, \tilde{\lambda}_{kj1}, \tilde{\lambda}_{kj2} \in (0,1).$$

Eqs. A.2.4 − A.2.9 are derived through Taylor expansion. $\phi(h)$, $\tilde{\phi}(h)$, $\varphi(h)$, $\tilde{\varphi}(h)$, $\psi(h)$, $\tilde{\psi}(h)$ are their Lagrange remainders, they are all bounded in some neighborhood of $\{h : |h| < \varepsilon\}$ of zero.

The reason for us to restrain the Lagrangian function $L(t, \mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}})$ into the summation of $L_1(t, \mathbf{y}, \dot{\mathbf{y}})$ and $L_2(t, \mathbf{y}, \ddot{\mathbf{y}})$, which is obviously less general, is that we are

at the moment unable to find a consistent approximation scheme for $\dot{\mathbf{y}}$ and $\ddot{\mathbf{y}}$ under one single discretization scheme (currently $\dot{\mathbf{y}}(t_k^*)$ and $\ddot{\mathbf{y}}(t_k)$ are discretized differently). As a conjecture, we guess the cross-coupling between velocity and acceleration will lead to some particular pattern in the differential equation of the variational problem, which may be beyond the reach of the conventional discretization approaches we have used.

Step 2: Expand $\mathbf{E}_h\Big(\Big[\ \mathbf{P}_h^{n-1}(\mathbf{y}(t))\quad \mathbf{P}_h^{n-1}(\mu(t))\ \Big]\Big)$.

$$\mathbf{E}_h\Big(\Big[\ \mathbf{P}_h^{n-1}(\mathbf{y}(t))\quad \mathbf{P}_h^{n-1}(\mu(t))\ \Big]\Big)$$

$$= \tilde{\mathcal{A}}\,\frac{\partial L_1}{\partial \mathbf{y}_j}\Big(\mathbf{P}^{*n}_h(t), \mathcal{A}\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big), \mathcal{B}\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big)\Big)$$

$$+\tilde{\mathcal{B}}\,\frac{\partial L_1}{\partial \dot{\mathbf{y}}_j}\Big(\mathbf{P}^{*n}_h(t), \mathcal{A}\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big), \mathcal{B}\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big)\Big)$$

$$+\frac{\partial L_2}{\partial \mathbf{y}_j}\Big(\mathbf{P}_h^{n-1}(t), \mathbf{P}_h^{n-1}\big(\mathbf{y}(t)\big), \tilde{\mathcal{C}}\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big)\Big)$$

$$+\tilde{\mathcal{C}}\,\frac{\partial L_2}{\partial \ddot{\mathbf{y}}_j}\Big(\mathbf{P}_h^{n+1}(t), \mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big), \mathcal{C}\mathbf{P}_h^{n+1}\big(\mathbf{y}(t)\big) + \mathcal{D}\Big)$$

$$+diag\Big(\mathbf{P}_h^{n-1}\big(\mu(t)\big)\Big)\,\frac{\partial C}{\partial \mathbf{y}_j}\Big(\mathbf{P}_h^{n-1}\big(\mathbf{y}(t)\big)\Big)$$

$$=$$ continue at next page ...

$$
\begin{aligned}
= \quad & \tilde{\mathcal{A}}\,\frac{\partial L_1}{\partial \mathbf{y}_j}\Big(\mathbf{P}^{*n}_h(t), \mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big) + \phi(h)h^2, \mathbf{P}^{*n}_h\big(\dot{\mathbf{y}}(t)\big) + \varphi(h)h^2\Big) \\
& + \tilde{\mathcal{B}}\,\frac{\partial L_1}{\partial \dot{\mathbf{y}}_j}\Big(\mathbf{P}^{*n}_h(t), \mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big) + \phi(h)h^2, \mathbf{P}^{*n}_h\big(\dot{\mathbf{y}}(t)\big) + \varphi(h)h^2\Big) \\
& + \frac{\partial L_2}{\partial \mathbf{y}_j}\Big(\mathbf{P}^{n-1}_h(t), \mathbf{P}^{n-1}_h\big(\mathbf{y}(t)\big), \mathbf{P}^{n-1}_h\big(\ddot{\mathbf{y}}(t)\big) + \tilde{\psi}_2(h)h^2\Big) \\
& + \tilde{\mathcal{C}}\,\frac{\partial L_2}{\partial \ddot{\mathbf{y}}_j}\Big(\mathbf{P}^{n+1}_h(t), \mathbf{P}^{n+1}_h\big(\mathbf{y}(t)\big), \mathbf{P}^{n+1}_h\big(\ddot{\mathbf{y}}(t)\big) + \psi_1(h)h + \psi_2(h)h^2\Big) \\
& + diag\Big(\mathbf{P}^{n-1}_h\big(\mu(t)\big)\Big)\,\frac{\partial C}{\partial \mathbf{y}_j}\Big(\mathbf{P}^{n-1}_h\big(\mathbf{y}(t)\big)\Big) \\
= \quad & \tilde{\mathcal{A}}\,\mathbf{P}^{*n}_h\Big(\frac{\partial L_1}{\partial \mathbf{y}_j}\big(t, \mathbf{y}(t), \dot{\mathbf{y}}(t)\big)\Big) \\
& + \tilde{\mathcal{B}}\,\mathbf{P}^{*n}_h\Big(\frac{\partial L_1}{\partial \dot{\mathbf{y}}_j}\big(t, \mathbf{y}(t), \dot{\mathbf{y}}(t)\big)\Big) \\
& + \mathbf{P}^{n-1}_h\Big(\frac{\partial L_2}{\partial \mathbf{y}_j}\big(t, \mathbf{y}(t), \ddot{\mathbf{y}}(t)\big)\Big) \\
& + \tilde{\mathcal{C}}\,\mathbf{P}^{n+1}_h\Big(\frac{\partial L_2}{\partial \ddot{\mathbf{y}}_j}\big(t, \mathbf{y}(t), \ddot{\mathbf{y}}(t)\big)\Big) \\
& + \mathbf{P}^{n-1}_h\Big(\mu(t)\,\frac{\partial C}{\partial \mathbf{y}_j}\big(\mu(t), \mathbf{y}(t)\big)\Big) \\
& + \tilde{\mathcal{A}}\,\Bigg(\frac{\partial^2 L_1}{\partial \mathbf{y}_j \partial \dot{\mathbf{y}}_j}\Big(\mathbf{P}^{*n}_h(t), \mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big) + \epsilon \cdot \phi(h)h^2, \mathbf{P}^{*n}_h\big(\dot{\mathbf{y}}(t)\big)\Big)\,\phi(h)h^2 \\
& \qquad + \frac{\partial^2 L_1}{\partial \mathbf{y}_j \partial \ddot{\mathbf{y}}_j}\Big(\mathbf{P}^{*n}_h(t), \mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big), \mathbf{P}^{*n}_h\big(\dot{\mathbf{y}}(t)\big) + \varepsilon \cdot \varphi(h)h^2\Big)\,\varphi(h)h^2\Bigg) \\
& + \tilde{\mathcal{B}}\,\Bigg(\frac{\partial^2 L_1}{\partial \mathbf{y}_j \partial \dot{\mathbf{y}}_j}\Big(\mathbf{P}^{*n}_h(t), \mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big) + \epsilon \cdot \phi(h)h^2, \mathbf{P}^{*n}_h\big(\dot{\mathbf{y}}(t)\big)\Big)\,\phi(h)h^2 \\
& \qquad + \frac{\partial^2 L_1}{\partial \mathbf{y}_j \partial \ddot{\mathbf{y}}_j}\Big(\mathbf{P}^{*n}_h(t), \mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big), \mathbf{P}^{*n}_h\big(\dot{\mathbf{y}}(t)\big) + \varepsilon \cdot \varphi(h)h^2\Big)\,\varphi(h)h^2\Bigg) \\
& + \frac{\partial^2 L_2}{\partial \ddot{\mathbf{y}}_j^2}\Big(\mathbf{P}^{n+1}_h(t), \mathbf{P}^{n+1}_h\big(\mathbf{y}(t)\big), \mathbf{P}^{n+1}_h\big(\ddot{\mathbf{y}}(t)\big) + \delta \cdot \big(\psi_1(h)h + \psi_2(h)h^2\big)\Big) \\
& \quad \Big(\psi_1(h)h + \psi_2(h)h^2\Big) \\
& + \tilde{\mathcal{C}}\,\frac{\partial^2 L_2}{\partial \ddot{\mathbf{y}}_j^2}\Big(\mathbf{P}^{n+1}_h(t), \mathbf{P}^{n+1}_h\big(\mathbf{y}(t)\big), \mathbf{P}^{n+1}_h\big(\ddot{\mathbf{y}}(t)\big) + \delta \cdot \big(\psi_1(h)h + \psi_2(h)h^2\big)\Big) \\
& \quad \Big(\psi_1(h)h + \psi_2(h)h^2\Big)
\end{aligned}
$$

where $\epsilon, \varepsilon \in \mathbb{R}^{n\times p}$ and $\delta \in \mathbb{R}^{(n+1)\times p}$ whose elements $\epsilon_{ij}, \varepsilon_{ij}, \delta_{ij} \in (0,1)$, and $\cdot$ is the elementwise (Hadamard) matrix multiplication. The first step is just repeating the definition A.2.2, and the next one is derived by plugging Eqs. A.2.4, A.2.6, A.2.8 in. The third step uses Taylor expansion with Lagrange remainders in the same way as getting Eqs. A.2.4 $-$ A.2.9.

Let $\mathcal{X}$ be the last five terms in $\mathbf{E}_h()$, we can see that all terms in $\mathcal{X}$ has the common factor h, so that $\mathcal{X}$ can be expressed as $\mathcal{Y}\,h$, where $\mathcal{Y}$ is bounded.

$$
\begin{aligned}
\mathcal{X} :=\ & \tilde{\mathcal{A}} \left(\frac{\partial^2 L_1}{\partial \mathbf{y}_j \partial \dot{\mathbf{y}}_j} \Big(\mathbf{P}^{*n}_h(t), \mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big) + \epsilon \cdot \phi(h) h^2, \mathbf{P}^{*n}_h\big(\dot{\mathbf{y}}(t)\big) \Big) \phi(h) h^2 \right. \\
& \left. + \frac{\partial^2 L_1}{\partial \mathbf{y}_j \partial \ddot{\mathbf{y}}_j} \Big(\mathbf{P}^{*n}_h(t), \mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big), \mathbf{P}^{*n}_h\big(\dot{\mathbf{y}}(t)\big) + \varepsilon \cdot \varphi(h) h^2 \Big) \varphi(h) h^2 \right) \\
& + \tilde{\mathcal{B}} \left(\frac{\partial^2 L_1}{\partial \mathbf{y}_j \partial \dot{\mathbf{y}}_j} \Big(\mathbf{P}^{*n}_h(t), \mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big) + \epsilon \cdot \phi(h) h^2, \mathbf{P}^{*n}_h\big(\dot{\mathbf{y}}(t)\big) \Big) \phi(h) h^2 \right. \\
& \left. + \frac{\partial^2 L_1}{\partial \mathbf{y}_j \partial \ddot{\mathbf{y}}_j} \Big(\mathbf{P}^{*n}_h(t), \mathbf{P}^{*n}_h\big(\mathbf{y}(t)\big), \mathbf{P}^{*n}_h\big(\dot{\mathbf{y}}(t)\big) + \varepsilon \cdot \varphi(h) h^2 \Big) \varphi(h) h^2 \right) \\
& + \frac{\partial^2 L_2}{\partial \ddot{\mathbf{y}}_j^2} \Big(\mathbf{P}^{n+1}_h(t), \mathbf{P}^{n+1}_h\big(\mathbf{y}(t)\big), \mathbf{P}^{n+1}_h\big(\ddot{\mathbf{y}}(t)\big) + \delta \cdot \big(\psi_1(h) h + \psi_2(h) h^2 \big) \Big) \\
& \quad \Big(\psi_1(h) h + \psi_2(h) h^2 \Big) \\
& + \tilde{\mathcal{C}} \frac{\partial^2 L_2}{\partial \ddot{\mathbf{y}}_j^2} \Big(\mathbf{P}^{n+1}_h(t), \mathbf{P}^{n+1}_h\big(\mathbf{y}(t)\big), \mathbf{P}^{n+1}_h\big(\ddot{\mathbf{y}}(t)\big) + \delta \cdot \big(\psi_1(h) h + \psi_2(h) h^2 \big) \Big) \\
& \quad \Big(\psi_1(h) h + \psi_2(h) h^2 \Big) \\
=\ & \mathcal{Y}\, h
\end{aligned}
$$

Continue our journey of expanding $\mathbf{E}_h()$.

$$
\begin{aligned}
& \mathbf{E}_h\Big(\Big[\ \mathbf{P}_h^{n-1}(\mathbf{y}(t)) \quad \mathbf{P}_h^{n-1}(\mu(t))\ \Big]\Big) \\
= \ & \mathbf{P}_h^{n-1}\Big(\frac{\partial L_1}{\partial \mathbf{y}_j}\big(t, \mathbf{y}(t), \dot{\mathbf{y}}(t)\big)\Big) + \tilde{\phi}(h)h^2 \\
& - \mathbf{P}_h^{n-1}\Big(\frac{\partial L_1}{\partial \dot{\mathbf{y}}_j}\big(t, \mathbf{y}(t), \dot{\mathbf{y}}(t)\big)\Big) - \tilde{\varphi}(h)h^2 \\
& + \mathbf{P}_h^{n-1}\Big(\frac{\partial L_2}{\partial \mathbf{y}_j}\big(t, \mathbf{y}(t), \ddot{\mathbf{y}}(t)\big)\Big) \\
& + \mathbf{P}_h^{n-1}\Big(\frac{\partial L_2}{\partial \ddot{\mathbf{y}}_j}\big(t, \mathbf{y}(t), \ddot{\mathbf{y}}(t)\big)\Big) + \tilde{\psi}(h)h^2 \\
& + \mathbf{P}_h^{n-1}\Big(\mu(t)\,\frac{\partial C}{\partial \mathbf{y}_j}\big(\mu(t), \mathbf{y}(t)\big)\Big) \\
& + \mathcal{X} \\
= \ & 0 + \tilde{\phi}(h)h^2 - \tilde{\varphi}(h)h^2 + \tilde{\psi}(h)h^2 + \mathcal{X} \\
= \ & \Big(\tilde{\phi}(h)h - \tilde{\varphi}(h)h + \tilde{\psi}(h)h + \mathcal{Y}\Big)\, h \\
\leq \ & \alpha\, h
\end{aligned}
$$

Step 1 comes from one property of our approximation schemes Eqs. A.2.5, A.2.7, A.2.9, and step 2 is because $\{\mathbf{y}(t),\, \mu(t)\}$ is the only solution of Eq. 2.3.5.

□

Lastly, we present the main convergence proposition and its proof.

Theorem A.2.2. *With the approximation schemes Eqs. 2.3.17, 2.3.18, 2.3.19 and assumptions 1,2, there exists positive constants α, s and γ such that*

$$
\Big\|\Big[\ \boldsymbol{Y}_h \quad \boldsymbol{M}_h\ \Big] - \Big[\ \boldsymbol{P}_h^{n-1}\big(\boldsymbol{y}(t)\big) \quad \boldsymbol{P}_h^{n-1}\big(\mu(t)\big)\ \Big]\Big\|^s \leq \alpha\gamma h \tag{A.2.10}
$$

Proof.

$$
\begin{aligned}
& \left\| \begin{bmatrix} \mathbf{Y}_h & \mathbf{M}_h \end{bmatrix} - \begin{bmatrix} \mathbf{P}_h^{n-1}(\mathbf{y}(t)) & \mathbf{P}_h^{n-1}(\mu(t)) \end{bmatrix} \right\|^s \\
\leq \quad & \gamma \left\| \mathbf{E}_h\Big(\begin{bmatrix} \mathbf{Y}_h & \mathbf{M}_h \end{bmatrix} \Big) - \mathbf{E}_h\Big(\begin{bmatrix} \mathbf{P}_h^{n-1}(\mathbf{y}(t)) & \mathbf{P}_h^{n-1}(\mu(t)) \end{bmatrix} \Big) \right\| \\
= \quad & \gamma \left\| \mathbf{E}_h\Big(\begin{bmatrix} \mathbf{P}_h^{n-1}(\mathbf{y}(t)) & \mathbf{P}_h^{n-1}(\mu(t)) \end{bmatrix} \Big) \right\| \\
\leq \quad & \alpha\gamma h
\end{aligned}
$$

□

The first step uses the stability assumption, then the next step is because $\{\mathbf{Y}_h, \mathbf{M}_h\}$ is the unique solution of Eqs. A.1.2, A.1.1. Lastly, lemma A.2.1 is employed at step 3 to finalize the whole proof.

Of course, $\mathbf{Y}_h$ fulfills the constraint C (Eq. A.1.2). From theorem A.2.2, we will see as $h \to 0$, the computed result from algorithm **A1.1** converges to the discretized value of the Euler-Lagrange equation's solution curve.

Appendix B

Derivation of Rolling Map for SO_3

The section derives the *kinematic equations* (Eq.3.3.14) from the no-slipping and no-twisting conditions (***C2, C3*** in page 59). Contrary to the sphere case, we now lose geometric intuition. Therefore, instead of having an idea of manifold's rigid-body motion initially (as for S^2 in [31]), the derivation of the SO_3' rolling map involves quite a bit of abstract mathematical reasoning.

B.1 Group Action, Rolling Curve, and Its Development

As mentioned in Section 3.3.2, we define the group $G = SO_3 \times SO_3 \ltimes \mathbb{R}^{3\times3}$ acts on $\mathbb{R}^{3\times3}$ via

$$\begin{aligned} G \times \mathbb{R}^{3\times3} &\to \mathbb{R}^{3\times3} \\ \big((U,V,X),\mathbf{p}\big) &\mapsto U\mathbf{p}V^\top + X, \end{aligned} \tag{B.1.1}$$

where G acts on itself via

$$(U_2,V_2,X_2)\circ(U_1,V_1,X_1) := \big(U_2U_1, V_2V_1, U_2X_1V_2^\top + X_2\big). \tag{B.1.2}$$

Now let $\mathbf{R}_0$ be an arbitary point in SO_3, and $[0,T] \to \alpha(t) = U(t)^\top\mathbf{R}_0V(t)$ a curve on SO_3 starting at $\mathbf{R}_0$ at $t = 0$. We will show that, under some restrictions,

the map

$$\begin{aligned} r : [0,T] &\to G = SO_3 \times SO_3 \ltimes \mathbb{R}^{3\times 3} \\ t &\mapsto r(t) = \big(U(t), V(t), X(t)\big) \end{aligned} \tag{B.1.3}$$

is a rolling map along

$$\alpha(t) = U^\top(t)\, \mathbf{R}_0\, V(t), \tag{B.1.4}$$

with development

$$\begin{aligned} \alpha_{dev} &= r(t) \circ \alpha(t) \\ &= U(t)\, \alpha(t)\, V^\top(t) + X(t) \\ &= \mathbf{R}_0 + X(t). \end{aligned} \tag{B.1.5}$$

B.2 Rolling Map without Slipping or Twisting

As we know, if r is a rolling map, the following two conditions need to be satisfied

C2 *(The no-slipping condition)*

For all $t \in [0,T]$ it holds

$\dot{r}(t) \circ r(t)^{-1} \circ \alpha_{dev}(t) = 0,$

C3 *(The no-twisting condition)*

For all $t \in [0,T]$ it holds

(i) (Tangential part)

$\dot{r}(t) \circ r(t)^{-1} \circ T_{\alpha_{dev}(t)}V \subset (T_{\alpha_{dev}(t)}V)^\perp,$

(ii) (Normal part)

$\dot{r}(t) \circ r(t)^{-1} \circ (T_{\alpha_{dev}(t)}V)^\perp \subset T_{\alpha_{dev}(t)}V.$

For the rolling map defined in Eq.B.1.3, the no-slipping condition can be written as

$$\begin{aligned} \dot{r} \circ r^{-1} \circ \alpha_{dev} = 0 &\Leftrightarrow \dot{r} \circ \alpha_{dev} = 0 \\ &\Leftrightarrow \dot{U}\alpha V^\top + U\alpha \dot{V}^\top + \dot{X} = 0 \\ &\Leftrightarrow \dot{U}U^\top \mathbf{R}_0 V V^\top + U U^\top \mathbf{R}_0 V \dot{V}^\top + \dot{X} = 0 \\ &\Leftrightarrow \dot{U}U^\top \mathbf{R}_0 + \mathbf{R}_0 V \dot{V}^\top + \dot{X} = 0. \end{aligned} \tag{B.2.1}$$

If we assign

$$\begin{aligned} \dot{U}U^\top &=: -\Omega_U/2 \in \mathfrak{so}_3, \\ V\dot{V}^\top &=: -\Omega_V/2 \in \mathfrak{so}_3, \end{aligned} \tag{B.2.2}$$

the no-slipping condition ***C2*** becomes

$$\dot{X} = \frac{\Omega_U}{2}\mathbf{R}_0 + \mathbf{R}_0 \frac{\Omega_V}{2}. \tag{B.2.3}$$

Now let's deal with the no-twisting condition. From ***C3***, we know that if we let $\forall \rho \subset T_{\alpha_{dev}} V$ and $\forall \varrho \subset (T_{\alpha_{dev}} V)^\perp$, the followings must hold

$$\begin{aligned} \left(\dot{U}, \dot{V}, \dot{X}\right) \circ \left(U^\top, V^\top, -U^\top X V\right) \circ \rho &= \left(\dot{U}, \dot{V}, \dot{X}\right) \circ \left(U^\top \rho V\right) \\ &= \dot{U}U^\top \rho V V^\top + U U^\top \rho V \dot{V}^\top \\ &= \dot{U}U^\top \rho + \rho V \dot{V}^\top \subset (T_{\alpha_{dev}} V)^\perp, \end{aligned} \tag{B.2.4}$$

and, similarly,

$$\left(\dot{U}, \dot{V}, \dot{X}\right) \circ \left(U^\top, V^\top, -U^\top X V\right) \circ \varrho = \dot{U}U^\top \varrho + \varrho V \dot{V}^\top \subset T_{\alpha_{dev}} V. \tag{B.2.5}$$

Moreover, $\forall \rho \subset T_{\alpha_{dev}} V$ is of the form $\rho = \mathbf{R}_0 \Omega_\rho$ for some $\Omega_\rho \in \mathfrak{so}_3$. Therefore, Eq.B.2.4 is equivalent to requiring the matrix $\mathbf{R}_0^\top \left(\dot{U}U^\top \mathbf{R}_0 \Omega_\rho + \mathbf{R}_0 \Omega_\rho V \dot{V}^\top\right)$ symmetric,

$\forall \Omega_\rho \in \mathfrak{so}_3$. Namely,

$$\begin{aligned}
&\mathbf{R}_0^\top\left(\dot{U}U^\top\mathbf{R}_0\Omega_\rho + \mathbf{R}_0\Omega_\rho V\dot{V}^\top\right) = \left(\mathbf{R}_0^\top\left(\dot{U}U^\top\mathbf{R}_0\Omega_\rho + \mathbf{R}_0\Omega_\rho V\dot{V}^\top\right)\right)^\top \\
\Leftrightarrow\ &\mathbf{R}_0^\top\Omega_U\mathbf{R}_0\Omega_\rho + \Omega_\rho\Omega_V = \left(\mathbf{R}_0^\top\Omega_U\mathbf{R}_0\Omega_\rho + \Omega_\rho\Omega_V\right)^\top \\
\Leftrightarrow\ &\mathbf{R}_0^\top\Omega_U\mathbf{R}_0\Omega_\rho + \Omega_\rho\Omega_V = \Omega_\rho^\top\mathbf{R}_0^\top\Omega_U^\top\mathbf{R}_0 + \Omega_V^\top\Omega_\rho^\top \\
\Leftrightarrow\ &\mathbf{R}_0^\top\Omega_U\mathbf{R}_0\Omega_\rho - \Omega_V^\top\Omega_\rho^\top = \Omega_\rho^\top\mathbf{R}_0^\top\Omega_U^\top\mathbf{R}_0 - \Omega_\rho\Omega_V \\
\Leftrightarrow\ &\mathbf{R}_0^\top\Omega_U\mathbf{R}_0\Omega_\rho - \Omega_V\Omega_\rho = \Omega_\rho\mathbf{R}_0^\top\Omega_U\mathbf{R}_0 - \Omega_\rho\Omega_V \\
\Leftrightarrow\ &\left(\mathbf{R}_0^\top\Omega_U\mathbf{R}_0 - \Omega_V\right)\Omega_\rho = \Omega_\rho\left(\mathbf{R}_0^\top\Omega_U\mathbf{R}_0 - \Omega_V\right).
\end{aligned} \tag{B.2.6}$$

We can see that $\mathbf{R}_0^\top\Omega_U\mathbf{R}_0 - \Omega_V$ commutes with Ω_ρ, $\forall\Omega_\rho \in \mathfrak{so}_3$, and the only possibility for such circumstance is that $\mathbf{R}_0^\top\Omega_U\mathbf{R}_0 - \Omega_V$ is a zero matrix, i.e.,

$$\Omega_V = \mathbf{R}_0^\top\Omega_U\mathbf{R}_0. \tag{B.2.7}$$

On the other hand, $\forall\varrho \subset (T_{\alpha_{dev}}V)^\perp$ is of the form $\varrho = \mathbf{R}_0 S_\varrho$ for some symmetric matrix S_ϱ, and Eq.B.2.5 is equivalent to requiring the matrix $\mathbf{R}_0^\top\left(\dot{U}U^\top\mathbf{R}_0 S + \mathbf{R}_0 S V\dot{V}^\top\right)$ is skew-symmetric, $\forall S = S^\top$. After some similar derivation, we can also end up with Eq.B.2.7.

Plug Eq.B.2.7 into Eq.B.2.3, now we can finally derive the *kinematic equations* for the rolling motion of SO_3, where $\Omega(t) := \Omega_U(t)$,

$$\begin{aligned}
\dot{X}(t) = \frac{\Omega_U(t)}{2}\mathbf{R}_0 + \mathbf{R}_0\frac{\Omega_V(t)}{2} &\Leftrightarrow \dot{X}(t) = \Omega(t)\,\mathbf{R}_0, \\
\dot{U}(t)U^\top(t) = -\Omega_U(t)/2 &\Leftrightarrow \dot{U}(t) = -\frac{1}{2}\,U(t)\,\Omega(t), \\
V(t)\dot{V}^\top(t) = -\Omega_V(t)/2 &\Leftrightarrow V(t)\dot{V}^\top(t) = -\frac{1}{2}\,\mathbf{R}_0^\top\,\Omega(t)\,\mathbf{R}_0 \\
&\Leftrightarrow \dot{V}(t)V(t)^\top = -\frac{1}{2}\,\mathbf{R}_0^\top\,\Omega^\top(t)\,\mathbf{R}_0 \\
&\Leftrightarrow \dot{V}(t)V(t)^\top = \frac{1}{2}\,\mathbf{R}_0^\top\,\Omega(t)\,\mathbf{R}_0 \\
&\Leftrightarrow \dot{V}(t) = \frac{1}{2}\,V(t)\,\mathbf{R}_0^\top\,\Omega(t)\,\mathbf{R}_0.
\end{aligned} \tag{B.2.8}$$

As mentioned in Section 3.3.2, if we choose $\Omega(t) = \Omega$, the answer for the *kinematic equations* become

$$\begin{aligned} X(t) &= t\Omega\,\mathbf{R}_0, \\ U(t) &= \mathrm{e}^{-t\frac{\Omega}{2}}, \\ V(t) &= {\mathbf{R}_0}^{\top}\,\mathrm{e}^{t\frac{\Omega}{2}}\,\mathbf{R}_0. \end{aligned} \tag{B.2.9}$$

As the result, the rolling curve is given as

$$\begin{aligned} \alpha(t) &= U^{\top}(t)\,\mathbf{R}_0\,V(t) \\ &= \mathrm{e}^{t\frac{\Omega}{2}}\,\mathbf{R}_0\,{\mathbf{R}_0}^{\top}\mathrm{e}^{t\frac{\Omega}{2}}\mathbf{R}_0 \\ &= \mathrm{e}^{t\Omega}\mathbf{R}_0, \end{aligned} \tag{B.2.10}$$

which is used in the second step of **A3.1** in Section 3.3.3.

Appendix C

Experimental Work Performed on WAM and Peripheral Devices

This section records our endeavors to achieve a good position tracking performance on WAM for the future hybrid motion/force control experiment. Some analytical and experimental work has been performed to identify dynamics properties of the WAM robotic system and its peripheral devices. Although some of our attempts turn out to be unsuccessful eventually, they can still be considered as useful experience for later experimentations on general real-time-controlled electro-mechanical systems.

C.1 Identification of Motor Torque Ripple

Precise position tracking is critical in many robotic applications, such as arc welding, laser cutting, NC machining, etc. Moreover, it is also the necessary foundation for a successful implementation of any hybrid motion/force control algorithm. For cable-drive robots like WAM, the advanced transmission system on one hand yields desirable features such as zero-backlash, near-zero friction, backdrivability, etc. On the other hand, high-frequency disturbances originated from motors will also be noticeable at the output, therefore an adequate compensation of motor torque ripple is critical for

achieving satisfactory tracking performances [6].

General speaking, motor torque ripple of brushless DC motor is caused by

1. the electromagnetic torque fluctuation, which is a function of motor current and rotor position;

2. the reluctance torque, i.e., the cogging torque, which is a function of rotor position only. [30, 34, 35]

Namely, motor torque ripple can be roughly modelled as

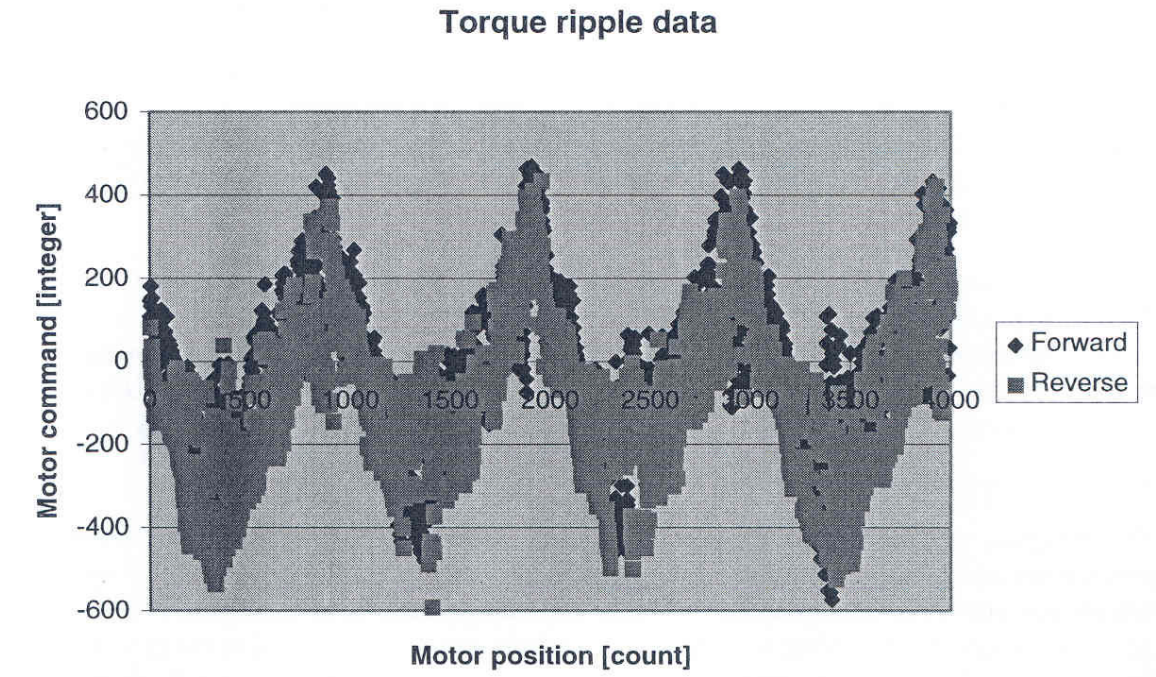
$$\begin{aligned} \tau_{\text{ripple}} &= \tau_{\text{fluctuation}} + \tau_{\text{cogging}} \\ &= f_1(\theta)\, I + f_2(\theta), \end{aligned} \tag{C.1.1}$$

where θ is the rotor position, and I is the drive current (i.e., the load).

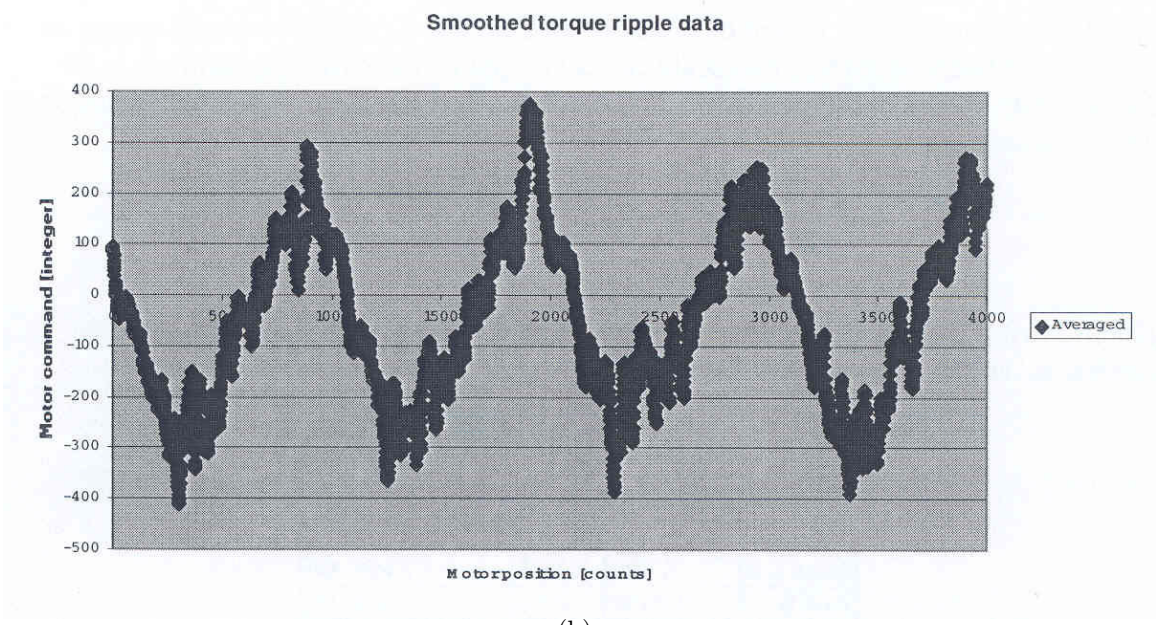
We have noticed from our experimentation experience that the motor torque ripple in WAM can sometimes be as large as 10 per cent of commanding torque. As a result, the motor torque ripple becomes as the second biggest disturbance besides the gravity for motion control of WAM, especially when the robot moves in a slow speed.

A dynamical calibration and a gravity compensation experiment have been successfully performed on WAM, whose video clip (ZeroGravity.mpg) can be found in the attached CD. The result of our gravity compensation is satisfactory: WAM can stay still in most of situations, except for a little bit drift of the shoulder joints 2 and 3 when the robot is in some particular poses.

However, during the course of the zero-gravity experiment, strong torque fluctuation can be obviously felt at the end-effector side, which we reckon is exactly the effect of motor torque ripple. More specifically, when the amplifier is switched off,



(a)



(b)

Figure C.1: The original motor torque identification: a) raw data; b) averaged and filtered data (courtesy of Barrett Technology Inc.)

almost no cogging torque is noticeable. Nevertheless, quite large torque ripple immediately appears, as long as the amplifier is powered on, even if zero torque has been commanded.

The existing torque ripple compensation is based on an identification procedure suggested in WAM's user manual [6]. The idea is to measure the steady state torque required to hold the unloaded motor splindle at each of its 4096 discretely measurable position (in practice, move the shaft at a very slow speed). The ripple data needs to be collected in both the forward and the reverse motor directions (Fig.C.1a), then averaged and filtered to finally form the look-up table (Fig.C.1b).

The effect of the torque ripple compensation under the above described scheme is in practice not good. Two obvious weaknesses of such an approach are:

1. A large proportion of the recorded commanding torque (i.e., the output of the high-gain closed-loop PID controller) is actually consumed for overcoming the static friction, which can not be eliminated by the off-line averaging and filtering processing;

2. A unified look-up table regardless of motor's load is applied for the later ripple compensation, however, the drive current is in fact a critical factor determining the magnitude of the motor torque ripple.

Instead, we choose to use an open-loop method for identifying the motor torque ripple. The experiment proceeds as follows:

1. Get a PID controller with fairly good performance on motor position tracking.

2. Drive only one motor with a constant commanding torque, let it move for a few shaft revolutions then stop the controller. Meanwhile, the positions of other three motors are forced to be kept fixed. Record the motor position for later off-line analysis.

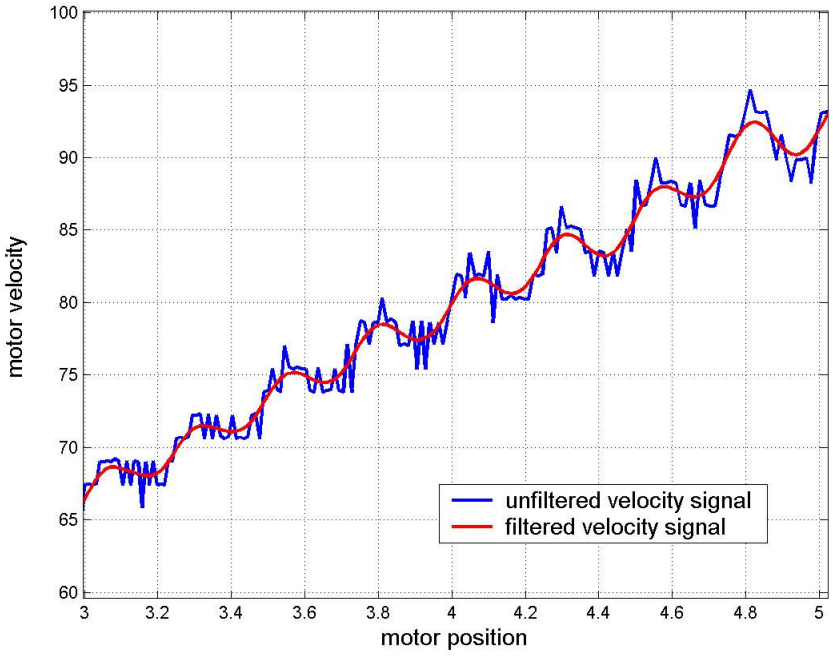
3. Differentiate the motor position to get the unfiltered motor velocity. Apply Gaussian filter to discard the high frequency quantization noise (Fig.C.2a). The velocity ripple is assumed to be caused by the motor torque ripple.

4. Further differentiate the smooth motor velocity signal once more to get the motor acceleration, multiply by the nominal motor inertia to get the final identified motor torque ripple.

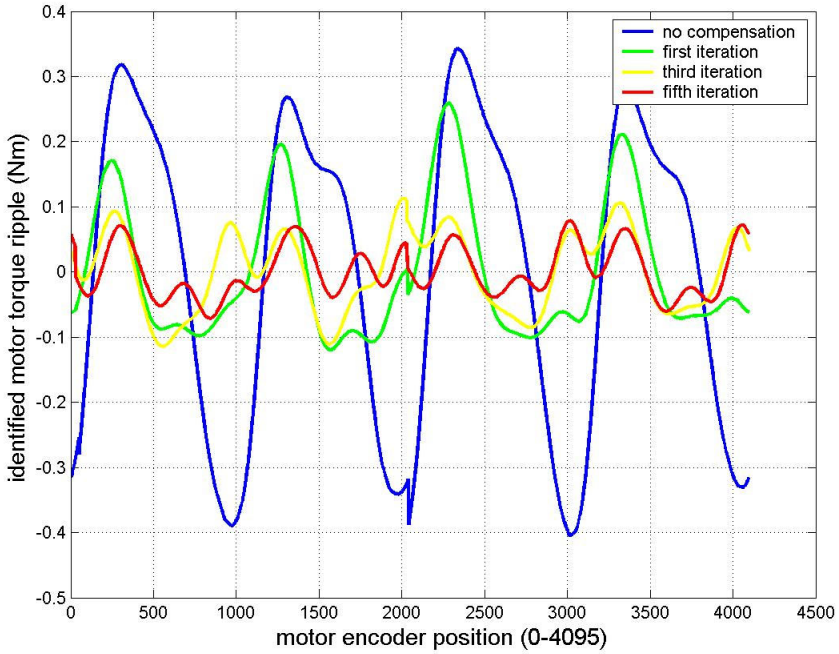
5. Apply a small fraction of such a ripple model as a feed-forward compensation term. Repeat from 2 until the identified motor torque ripple converges to a reasonable low magnitude.

Fig.C.2b shows one result of the iterative process. We can see that the magnitude of the identified torque ripple has been exaggerated, which may be caused by inaccurate estimation of the motor acceleration, or some dynamics model error. Therefore, in the experiment, we decide to feed only a small fraction of the identified torque ripple into the next iteration. As shown in Fig.C.2b, the compensation seems to be working, as the motor torque ripple has been suppressed by almost 80%. Nevertheless, the improvement is not significant when we introduce our torque ripple compensation into the feed-forward zero-gravity control.

We consider our effort on this issue to be a failure and abandon the feed-forward compensation method. In later experiments, the motor torque ripple is treated as a



(a)



(b)

Figure C.2: The modified motor torque identification: a) raw and filtered data of motor velocity; b) convergence of torque ripple by iterative feed-forward compensation

disturbance and is supposed to be rejected through the output feedback. It will be interesting to investigate why the iteration approach works well in one configuration, but fails under other situations. One problem that we can think of is with regard to our incomplete understanding of the motor amplifiers' mechanism. We assume them as current amplifiers according to the manual, however, that doesn't explain why torque ripple can be felt when the amplifiers are switched on while zero motor torque is commanded.

C.2 Kalman Filter for Motor Velocity Estimation

Another attempt to achieve a good position tracking performance that we made is to introduce a velocity feedback into the control loop. The major snag is that WAM is not equipped with any tachometer or accelerometer, and the only motion sensing is through the simulated shaft encoder measurement from resolver. Before, the motor velocity was obtained by directly differentiating the position signal, so that the resulting velocity data became quite noisy and the derivative-control was of very little use.

The high-frequency quantization noise of the velocity data is due to the limited encoder resolution and high sampling frequency. More specifically, suppose a motor is equipped with an encoder system with an interpulse angle of θ_m, and T_s is the sampling interval of control system, then the motor velocity estimate by finite encoder-driven shaft-position difference will be θ_m/T_s. For WAM, $\theta_m = 0.09\,\text{deg}$ (as the encoder resolution is 4096 per revolution), the sampling time interval T_s is 0.955 millisecond, and the motor velocity resolution becomes 94.24 deg/sec, which is unacceptably low.

Conventional digital low-pass filter can be implemented to obtain smoother velocity data. For example, Gaussian filters have been used in the off-line motor torque identification as described in the previous section, and the filtering effects are quite satisfactory. However, digital low-pass filters will inevitably introduce some time delay, which can bring instability to real-time controlled system therefore are unappropriate for WAM.

An alternative and better solution is to apply Kalman filters, which are deliberately invented for online filtering, and computationally cheap as well. Quite a few Kalman filter designs for estimating angular velocity from only encoder measurement are available in literature, and we have chosen Belanger *et al.*'s work [7] because of its easy implementation.

This particular filter design models the acceleration (if only velocity is supposed to be estimated) as being driven by a white noise (later will be considered as the process noise in the Kalman filtering formulation); and the quantization error is considered as the measurement noise, which is further assumed to be uniform distributed. In short, the canonical state description of system model can be shown as in follows

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w, \tag{C.2.1}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + e, \tag{C.2.2}$$

where w is a white noise (whose covariance Q is chosen empirically, indicating the intensiveness of motor acceleration's change), and e is the quantization error whose covariance is $\frac{1}{3}\theta_m$.

The corresponding discrete Kalman iteration is given as

$$\mathbf{P}(t+1|t) = \mathbf{A}_d\,\mathbf{P}(t|t)\,\mathbf{A}_d^\top + \mathbf{B}_d, \tag{C.2.3}$$

$$\mathbf{k}_d = \mathbf{P}(t|t-1)\begin{bmatrix} 0 \\ 1 \end{bmatrix}\Big(\begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{P}(t|t-1)\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \frac{1}{3}\,\theta_m\Big)^{-1}, \tag{C.2.4}$$

$$\mathbf{P}(t|t) = \Big(\mathbf{I}_2 - \mathbf{k}_d\begin{bmatrix} 1 & 0 \end{bmatrix}\Big)\,\mathbf{P}(t|t-1)\,\Big(\mathbf{I}_2 - \mathbf{k}_d\begin{bmatrix} 1 & 0 \end{bmatrix}\Big)^\top + \mathbf{k}_d\,(\frac{1}{3}\,\theta_m)\,\mathbf{k}_d^\top, \tag{C.2.5}$$

where

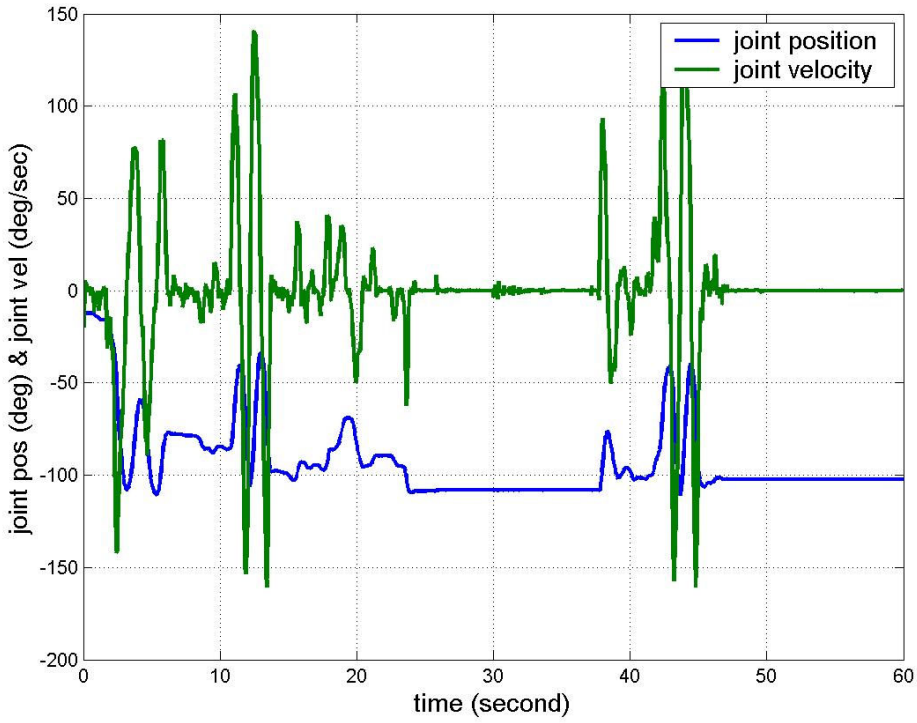


Figure C.3: Joint velocity estimation from encoder measurement

$$\mathbf{A}_d \approx \mathbf{I}_2 + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} 0.000955\,, \tag{C.2.6}$$

$$\mathbf{B}_d \approx 0.000955\,Q \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{C.2.7}$$

Fig.C.3 shows the velocity estimation of WAM's 4th joint (which is driven by a DC motor through a straight transmission) against its position. The result seems to be good as almost no phase shift is noticeable and the velocity configuration reasonably fits with the position trajectory. However, no matter how we tune the parameters in the Kalman filter, there are still some amount of high-frequency ripples in the estimated velocity curve. To ultimately solve this problem, we probably need to install either higher-resolution encoders or some velocity sensors.

C.3 Dynamics Calibration of JR3 6-DOF Force Sensor

For the proposed compliant motion control experiment, a JR3 6-DOF force sensor and a custom-made dumbbell end-effector are mounted at the end-tip of WAM's link 4. The JR3 6-DOF force sensors are capable of measuring force and torque on three orthogonal directions, and the reference point is the geometric center of the sensor (Fig.C.4).

Feedback of contact force will go into the control loop when the experiment runs. However, the total force reading is polluted by the gravity of the dumbbell and the sensor's working surface. To cancel the effect of gravity, a few parameters with regard to the dynamics properties of the force sensor and the end-effector need to be identified through experimentation.

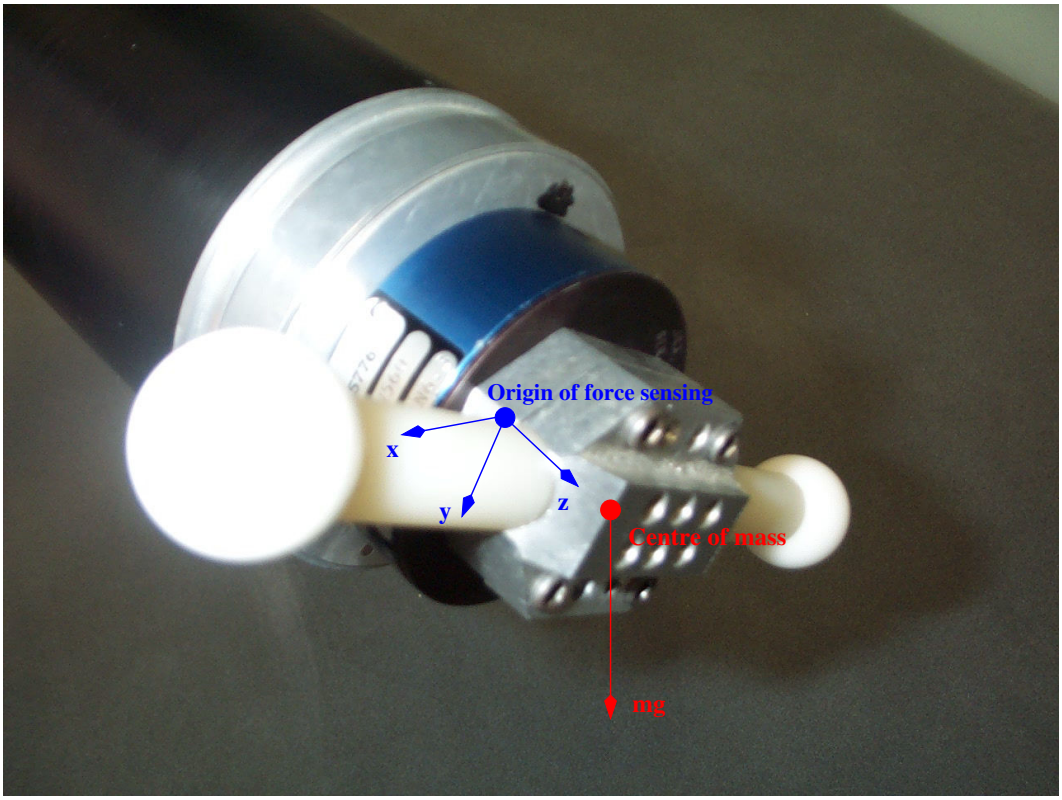


Figure C.4: JR3 force sensor with dumbbell end-effector

The generalized 6-D force of gravity is given as

$$\hat{\mathbf{f}} = \begin{bmatrix} \mathbf{E}\begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} \\ \mathbf{CoM} \times \mathbf{E}\begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} \end{bmatrix} = \begin{bmatrix} m\,\mathbf{E}\begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} \\ \begin{pmatrix} 0 & -m\mathbf{CoM}_z & m\mathbf{CoM}_y \\ m\mathbf{CoM}_z & 0 & -m\mathbf{CoM}_x \\ -m\mathbf{CoM}_y & m\mathbf{CoM}_x & 0 \end{pmatrix} \mathbf{E}\begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} \end{bmatrix} \tag{C.3.1}$$

where m is the mass, g is the gravitational acceleration, $\mathbf{CoM} \in \mathbb{R}^3$ is the center of mass, and $\mathbf{E} \in \mathbb{R}^{3\times3}$ is the rotational matrix representing the orientation of the force

sensor.

Obviously, the unknowns for the identification is m and the 3×3 matrix (denoted H) in the second row of $\hat{\mathbf{f}}$, which further include four parameters of m, $\mathbf{CoM}_x$, $\mathbf{CoM}_y$, and $\mathbf{CoM}_z$. As $\mathbf{E}$ can be calculated through the forward kinematics of WAM given robot's joint position, we can apply least-square algorithm to estimate the values of the unknowns from force readings at various robot poses. During the experiment, WAM moves very slowly, so that the effect of Coriolis force can be neglected.

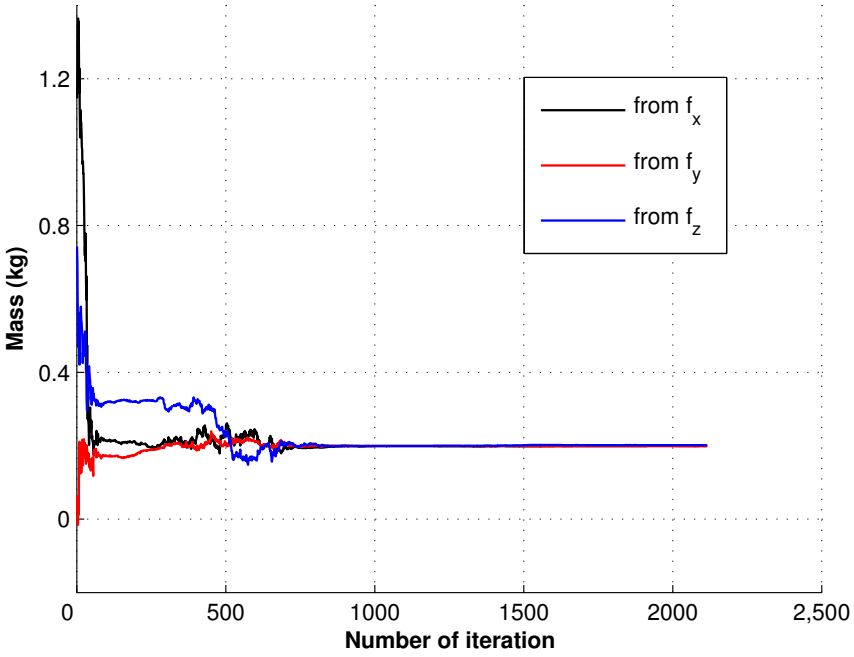
We have used the recursive least square estimation algorithm [1], and the convergence of the dynamics parameters is shown in Figs.C.5,C.6. Finally, these values are identified as

$$m = 0.200 \tag{C.3.2}$$

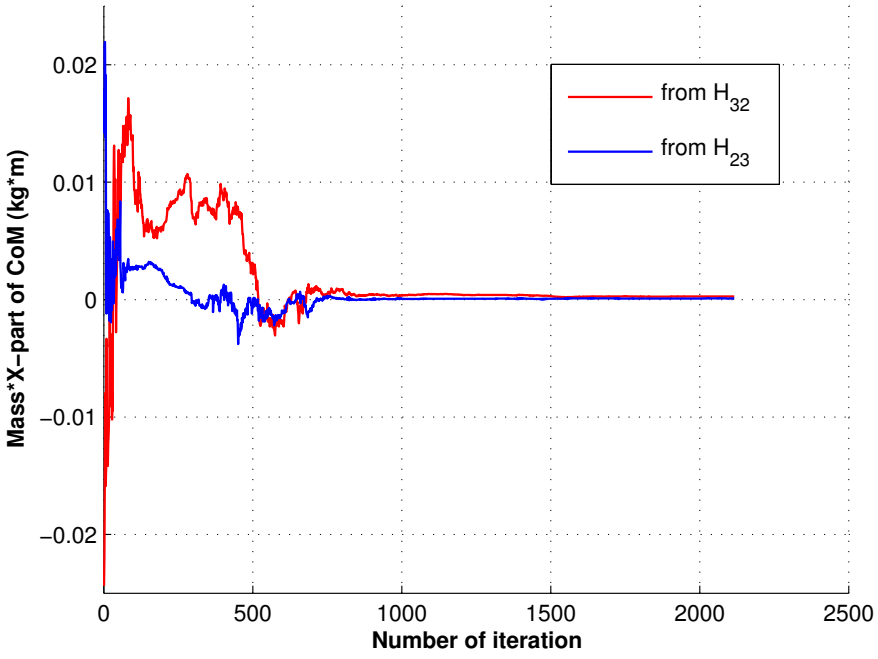
$$m\mathbf{CoM}_x = 0.000 \tag{C.3.3}$$

$$m\mathbf{CoM}_y = 0.000 \tag{C.3.4}$$

$$m\mathbf{CoM}_z = 0.00405 \tag{C.3.5}$$

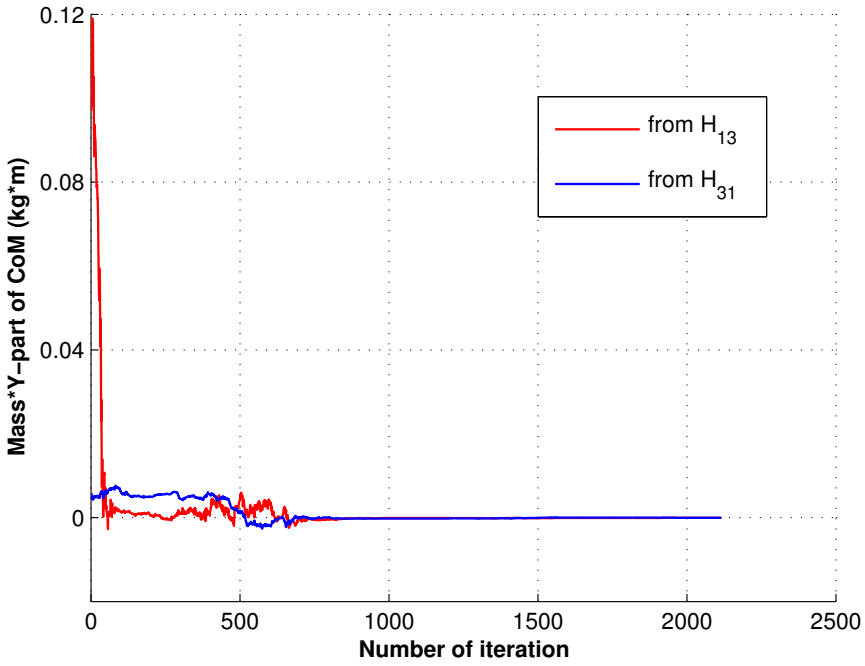


(a)

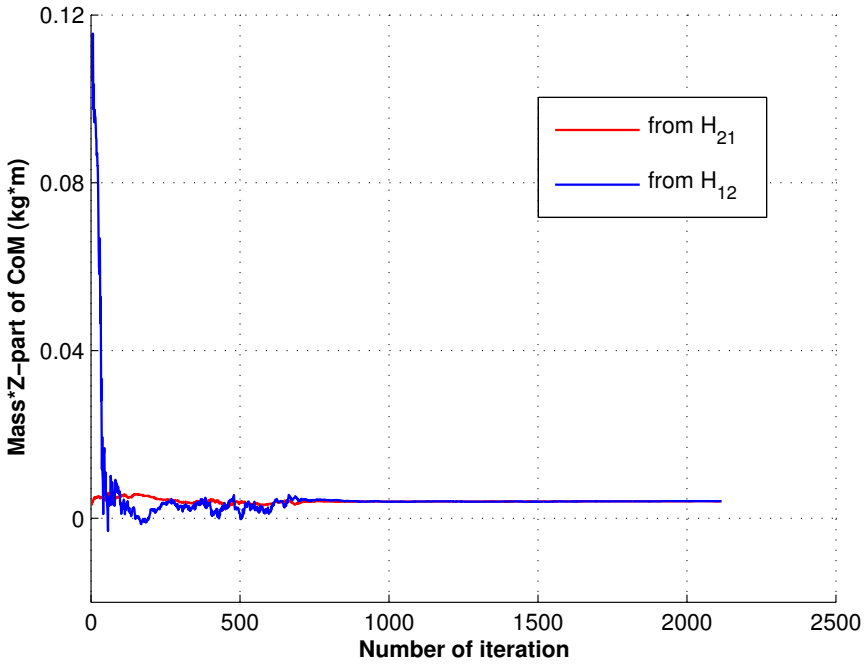


(b)

Figure C.5: Recursive least square estimation result: a) mass; b) mass $\times$ $\mathrm{CoM}_{\text{x-part}}$



(a)



(b)

Figure C.6: Recursive least square estimation result: a) mass × $\mathrm{CoM}_{\text{y-part}}$; b) mass × $\mathrm{CoM}_{\text{z-part}}$

Bibliography

[1] K.J. Åström and B. Wittenmark, *Computer-controlled systems: Theory and design*, Prentice Hall, Inc., 1990.

[2] A. Abbati-Marescotti, C. Bonivento, and C. Melchiorri, *On the invariance of the hybrid position force control*, Journal of Intelligent and Robotic Systems **3** (1990), no. 3, 233–250.

[3] O.P. Agrawal and Y.S. Xu, *On the global optimum path planning for redundant space manipulators*, IEEE Transactions on Systems Man and Cybernetics **24** (Sep 1994), no. 9, 1306–1316.

[4] C. Ahlbrandt, G. Benson, and W. Casey, *Minimal entropy probability paths between genome families*, J. Math. Biol. **48** (2004), no. 5, 563–590.

[5] C.H. An and J. Hollerbach, *The role of dynamic models in cartesian force control of manipulators*, International Journal of Robotics Research **8** (Aug 1989), no. 4, 51–72.

[6] Barrett Technology Inc., *Barrett arm BA4-300 user manual, version 1.0*, 1998.

[7] P.R. Belanger, P. Dobrovolny, A. Helmy, and X. Zhang, *Estimation of angular velocity and acceleration from shaft-encoder measurements*, International Journal of Robotics Research **17** (Nov 1998), no. 11, 1225–1233.

[8] J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C.A. Sagastizábal, *Numerical optimization*, Springer-Verlag, 2003.

[9] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge University Press, 2004.

[10] H. Bruyninckx, S. Demey, S. Dutre, and J. De Schutter, *Kinematic models for model-based compliant motion in the presence of uncertainty*, International Journal of Robotics Research **14** (Oct 1995), no. 5, 465–482.

[11] H. Bruyninckx and J. De Schutter, *Specification of force-controlled actions in the "Task Frame Formalism" - a synthesis*, IEEE Transactions on Robotics and Automation **12** (Aug 1996), no. 4, 581–589.

[12] A.J. Cahill, M.R. James, J.C. Kieffer, and D. Williamson, *Remarks on the application of dynamic programming to the optimal path timing of robot manipulators*, International Journal of Robust and Nonlinear Control **8** (May 1998), no. 6, 463–482.

[13] R. Cortesão, *Kalman techniques for intelligent control systems: Theory and robotic experiments*, Ph.D. thesis, University of Coimbra, Portugal, 2003.

[14] R. Cortesão, J. Park, and O. Khatib, *Real-time adaptive control for haptic manipulation with active observers*, Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2003, pp. 2938–2943.

[15] J.J. Craig, *Introduction to robotics: Mechanics and control*, second ed., Addison-Wesley Publishing Company, 1989.

[16] P. Crouch, G. Kun, and F. Silva Leite, *De Casteljau algorithm for cubic polynomials on the rotation group*, Proceedings of CONTROLO'96, 2nd Portuguese Conference on Automatic Control (Porto, Portugal), Sep 1996, pp. 547–552.

[17] ———, *Generalization of spline curves on the sphere: A numerical comparison*, Proceedings of CONTROLO'98, 3rd Portuguese Conference on Automatic Control (Coimbra, Portugal), Sep 1998, pp. 447–451.

[18] K.L. Doty, C. Melchiorri, and C. Bonivento, *A theory of generalized inverses applied to robotics*, International Journal of Robotics Research **12** (Feb 1993), no. 1, 1–19.

[19] Z. Doulgeri, N. Fahantidis, and A. Konstantinidis, *On the decoupling of position and force controllers in constrained robotic tasks*, Journal of Robotic Systems **15** (Jun 1998), no. 6, 323–340.

[20] Z. Doulgeri, N. Fahantidis, and R.P. Paul, *Nonlinear stability of hybrid control*, International Journal of Robotics Research **17** (Jul 1998), no. 7, 792–806.

[21] J. Duffy, *The fallacy of modern hybrid control theory that is based on "Orthogonal Complements" of twist and wrench spaces*, Journal of Robotic Systems **7** (Apr 1990), no. 2, 139–144.

[22] G.N. Elnagar and M.A. Kazemi, *Numerical solvability of nonlinear problems of the calculus of variations*, Numer. Func. Anal. Opt. **17** (1996), no. 5-6, 563–575.

[23] G.N. Elnagar and M. Razzaghi, *An alternative method for a classical problem in the calculus of variations*, Math Method Appl. Sci. **19** (1996), no. 13, 1091–1097.

[24] R. Featherstone, *Robot dynamics algorithms*, Kluwer Academic Publishers, 1987.

[25] ———, *Modeling and control of contact between constrained rigid bodies*, IEEE Transactions on Robotics and Automation **20** (Feb 2004), no. 1, 82–92.

[26] R. Featherstone, S.S. Thiebaut, and O. Khatib, *A general contact model for dynamically-decoupled force/motion control*, Proceedings of the 1999 IEEE International Conference on Robotics and Automation, May 1999, pp. 3281–3286.

[27] W.D. Fisher and M.S. Mujtaba, *Hybrid position force control: A correct formulation*, International Journal of Robotics Research **11** (Aug 1992), no. 4, 299–311.

[28] J. Gregory and G. Yang, *A c-infinity numerical method for the basic problem in the calculus of variations*, Utilitas Mathematica **56** (1999), 79–95.

[29] G. Hämmerlin and K. Hoffmann, *Numerical mathematics*, Springer-Verlag, 1991.

[30] D.C. Hanselman, *Minimum torque ripple, maximum efficiency excitation of brushless permanent-magnet motors*, IEEE Transactions on Industrial Electronics **41** (Jun 1994), no. 3, 292–300.

[31] K. Hüper and F. Silva Leite, *Smooth interpolating curves with applications to path planning*, Proceedings of the 10th Mediterranean Conference on Control and Automation (Lisbon, Portugal), Jul 2002.

[32] K. Hüper and F. Silva Leite, *On the geometry of rolling and interpolation curves on S^n, SO_n and Graßmann manifolds*, Tech. Report SISSA 56/2005/M, International School for Advanced Studies, Trieste, Italy, 2005.

[33] K. Hüper and J. Trumpf, *Newton-like methods for numerical optimization on manifolds.*, Proceedings of the 38th Asilomar Conference on Signals, Systems and Computers (California, USA), Nov 2004.

[34] S.M. Hwang and D.K. Lieu, *Design techniques for reduction of reluctance torque in brushless permanent-magnet motors*, IEEE Transactions on Magnetics **30** (Nov 1994), no. 6, 4287–4289.

[35] ———, *Reduction of torque ripple in brushless dc motors*, IEEE Transactions on Magnetics **31** (Nov 1995), no. 6, 3737–3739.

[36] I.G. Kang and F.C. Park, *Cubic spline algorithms for orientation interpolation*, International Journal for Numerical Methods in Engineering **46** (Sep 1999), no. 1, 45–64.

[37] O. Khatib, *A unified approach for motion and force control of robot manipulators - the operational space formulation*, IEEE Journal of Robotics and Automation **3** (Feb 1987), no. 1, 43–55.

[38] J.-C. Latombe, *Robot motion planning*, Kluwer, 1991.

[39] M. Leok, *Foundations of computational geometric mechanics*, Ph.D. thesis, California Institute of Technology, 2004.

[40] Y. Levin, M. Nediak, and A. Ben-Israel, *A direct newton method for calculus of variations*, Journal of Computational and Applied Mathematics **139** (2002), no. 2, 197–213.

[41] A. Lew, J.E. Marsden, M. Ortiz, and M. West, *An overview of variational integrators*, Finite Element Methods: 1970's and Beyond (2003).

[42] ———, *Variational time integrators*, Int. J. Numer. Meth. Engng. **60** (2004), 153–212.

[43] H. Lipkin and J. Duffy, *Hybrid twist and wrench control for a robotic manipulator*, Transactions of ASME, Journal of Mechanisms, Transmissions, and Automation in Design **110** (Jun 1988), no. 2, 138–144.

[44] G.F. Liu and Z.X. Li, *A unified geometric approach to modeling and control of constrained mechanical systems*, IEEE Transactions on Robotics and Automation **18** (Aug 2002), no. 4, 574–587.

[45] C.G. Lo Bianco and A. Piazzi, *Minimum-time trajectory planning of mechanical manipulators under dynamic constraints*, International Journal of Control **75** (Sep 2002), no. 13, 967–980.

[46] A. Marthinsen, *Interpolation in Lie groups*, SIAM Journal of Numerical Analysis **37** (1999), no. 1, 269–285.

[47] D.P. Martin, J. Baillieul, and J.M. Hollerbach, *Resolution of kinematic redundancy using optimization techniques*, IEEE Transactions on Robotics and Automation **5** (Aug 1989), no. 4, 529–533.

[48] N.H. McClamroch and D. Wang, *Feedback stabilization and tracking of constrained robots*, IEEE Transactions on Automatic Control **33** (May 1988), no. 5, 419–426.

[49] C.D. Meyer, *Matrix analysis and applied linear algebra*, SIAM, 2000.

[50] A. Miele and R.E. Pritchard, *Numerical solutions in the simplest problem of the calculus of variations*, SIAM Rev. **14** (1972), no. 3, 385–398.

[51] R.M. Murray, Z. Li, and S.S. Sastry, *A mathematical introduction to robotic manipulation*, CRC Press, 1994.

[52] F.C. Park and B. Ravani, *Bézier curves on Riemannian manifolds and Lie groups with kinematics applications*, ASME Journal of Mechanical Design **117** (1995), no. 1, 36–40.

[53] J. Park, R. Cortesão, and O. Khatib, *Multi-contact compliant motion control for robotic manipulators*, Proceedings of the 2004 IEEE International Conference on Robotics and Automation, April 2004, pp. 4789–4794.

[54] M.H. Raibert and J.J. Craig, *Hybrid position-force control of manipulators*, Transactions of ASME, Journal of Dynamic Systems, Measurement, and Control **103** (1981), no. 2, 126–133.

[55] J. De Schutter, H. Bruyninckx, W.H. Zhu, and M.W. Spong, *Force control: A bird's eye view*, Proceedings of IEEE CSS/RAS International Workshop on "Control Problems in Robotics and Automation: Future Directions" (San Diego, CA), Dec 1997.

[56] L. Sciavicco and B. Siciliano, *Modeling and control of robot manipulators*, The McGraw-Hill Companies, Inc., 1996.

[57] J.M. Selig and P.R. McAree, *A simple approach to invariant hybrid control*, Proceedings of the 1996 IEEE International Conference on Robotics and Automation, Apr 1996, pp. 2238–2245.

[58] R.W. Sharpe, *Differential geometry*, Springer-Verlag, New York, 1996.

[59] Y. Shen and R. Featherstone, *The effect of ill-conditioned inertial matrix on controlling robot manipulator*, Proceedings of Australasian Conference on Robotics and Automation (Brisbane, Australia), Dec 2003.

[60] S.K. Singh and M.C. Leu, *Manipulator motion planning in the presence of obstacles and dynamic constraints*, International Journal of Robotics Research **10** (Apr 1991), no. 2, 171–187.

[61] B. van Brunt, *The calculus of variations*, Springer-Verlag, 2004.

[62] F.Y.M. Wan, *Introduction to the calculus of variations and its applications*, Chapman and Hall, 1995.

[63] C-Y.E. Wang, W.K. Timoszyk, and J.E. Bobrow, *Payload maximization for open chained manipulators: Finding weightlifting motions for a puma 762 robot*, IEEE Transactions on Robotics and Automation **17** (Apr 2001), no. 2, 218–224.

[64] D. Wang and Y. Hamam, *Optimal trajectory planning of manipulators with collision detection and avoidance*, International Journal of Robotics Research **11** (Oct 1992), no. 5, 460–468.

[65] T. Yabuta, *Nonlinear basic stability concept of the hybrid position/force control scheme for robot manipulators*, IEEE Transactions on Robotics and Automation **8** (Oct 1992), no. 5, 663–670.

[66] T. Yoshikawa, *Dynamic hybrid position force control of robot manipulators - description of hand constraints and calculation of joint driving force*, IEEE Journal of Robotics and Automation **3** (Oct 1987), no. 5, 386–392.

[67] E. Zeidler, *Nonlinear functional analysis and its applications*, vol. 3, Springer-Verlag, 1985.

[68] ———, *Nonlinear functional analysis and its applications*, vol. 2a, Springer-Verlag, 1985.

[69] H. Zhang and R.P. Paul, *Hybrid control of robot manipulators*, Proceedings of the 1985 IEEE Conference on Decision and Control, Mar 1985, pp. 251–259.