Intelligent Control and Planning for Industrial Robots

by

Yu Zhao

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Summer 2018

Intelligent Control and Planning for Industrial Robots

Abstract

Intelligent Control and Planning for Industrial Robots

by

Yu Zhao

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

Industrial robots are widely used in a variety of applications in manufacturing. Today, most industrial robots have been pushed to work near their hardware design limits. Therefore, it is essential to develop advanced control techniques to further improve the performance of industrial robots. This dissertation focuses on efficient motion planning and effective trajectory tracking control for flexible robots. The difficulties of this work arise from the facts that 1) due to the complicated nonlinear mapping between robot configuration space and workspace, the constraints applied in one space are difficult to transfer to another space, 2) due to the inherent mechanical flexibility, static and dynamic deflections between actuators and robot end-effector are frequently observed, and they degrade the overall trajectory tracking performance. In regards to these issues, this dissertation proposes several methods to improve motion control performance of industrial robots.

Regarding motion planning, an optimal control based approach is presented. Because of the inherent complexity of the motion planning problem for articulated robots with multiple joints, most existing solutions decompose motion planning as path planning and trajectory planning problems. Because of the implementation of manual or random sampling approaches in path planning, the resulting solution is in general suboptimal. This dissertation proposes to formulate motion planning as a general nonlinear optimal control problem. A practical numerical method is investigated for trajectory optimization as one solution to the underlying optimal control problem. Intelligent discretization and automatic differentiation techniques are introduced to make the proposed approach highly efficient.

Regarding trajectory tracking of robots with compliant components, two kinds of flexibility are considered. One kind of flexibility comes from the compliant transmission elements, i.e., joint flexibility. For robot with joint flexibility, back-stepping control is designed to achieve high performance of trajectory tracking. To address model uncertainties in the system, a radial basis function network is introduced for online adaptive compensation. Lyapunov stability theory is used to prove the stability of the proposed adaptive controller. A data-driven approach for the structural design of a radial basis function network is also presented to effectively reduce the computation load.

Another kind of flexibility comes from the compliant links, which is known as link flexibility. Industrial robots equipped with large articulated structures as end-effectors are good examples of robots with link flexibility. One popular and promising approach for vibration suppression involving flexible links is input shaping. However the time delay introduced by input shaping is not permissible for time stringent applications. In this dissertation two modified input shaping approaches are presented to suppress residual vibration of end-effectors without introducing time delay to the entire motion. Considering the control signal generated from input shaping does not necessarily seem to be the best for robotic application, optimal vibration suppression is also discussed based on the proposed efficient numerical method for trajectory optimization.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

I am indebted to many people who made it possible for me to conclude my Ph.D. studies at Berkeley. I would first like to thank my thesis advisor Professor Masayoshi Tomizuka for admitting me to join the research group and always being supportive. This dissertation would not have been possible without his mentoring and guidance. I also want to express my gratitude to him and his wife Mrs. Miwako Tomizuka for being supportive to my newborn daughter and my family.

I would like to express my gratitude to my dissertation and qualifying exam committee members: Professor Oliver O'Reilly, Professor Ruzena Bajcsy, Professor Pieter Abbeel, and Professor J. Karl Hedrick for their insightful and invaluable suggestions and professional guidance. I could not imagine having better mentors than them for learning dynamics, robotics, and nonlinear control.

I am grateful to FANUC's support for my doctoral research on industrial robots. I would like to thank Dr. Wenjie Chen, Mr. Kaimeng Wang, Mr. Satoshi Inagaki, Ms. Weijia Li, Mr. Hiroshi Nakagawa and Mr. Tetsuaki Katou for their help and suggestions on the research project. I would also like to thank all members working in the robotics research group: Cong Wang, Chung-Yen Lin, Changliu Liu, Te Tang, Hsien-Chung Lin, Yongxiang Fan, Yujiao Cheng, and Shiyu Jin, for all the suggestions, discussions, and collaboration.

To all my colleagues and friends in Mechanical Systems Control Laboratory, I appreciate all of your support, discussion, conversation, advice, and friendship over the past years: Wenjie Chen, Xu Chen, Michael Chan, Kan Kanjanapas, Chi-Shen Tsai, Pedro Reynoso, Wenlong Zhang, Yizhou Wang, Raechel Tan, Minghui Zheng, Junkai Lu, Chung-Yen Lin, Cong Wang, Chen-Yu Chan, Changliu Liu, Yaoqiong Du, Xiaowen Yu, Kevin Haninger, Shiying Zhou, Dennis Wai, Shuyang Li, Te Tang, Hsien-Chung Lin, Yongxiang Fan, Wei Zhan, Cheng Peng, Daisuke Kaneishi, Zining Wang, Kiwoo Shin, Liting Sun, Yu-Chu Huang, Jiachen Li, Chen Tang, Zhuo Xu, Jianyu Chen, Yujiao Cheng, Yeping Hu, Jessica Leu, Hengbo Ma, Shiyu Jin, Taohan Wang, Richard Lee, and all others. The lab is always a welcoming place because of you.

Last and foremost, I would like to thank my family. I would like to thank my parents for their unconditional love and support through my love. I am deeply indebted to my beloved wife, Xiaowen Yu. I can not imagine a life without her love, support, understanding, and dedication. Finally, I want to express my love to my daughter Natalie for being a part of my life. Your coming means so much to me.

Chapter 1

Introduction

1.1 Background

An industrial robot is an automatically controlled, re-programmable multipurpose manipulator, which is programmable in three or more axes [1]. Today, industrial robots are widely used in a variety of applications in manufacturing, including but not limited to welding, painting, assembly, palletizing, inspection, and testing. Since the development of industrial robots has been mainly dictated by the automotive industry, high cost efficiency, reliability, and productivity have been the focus of the research and development work of robot manufacturers. In order to maximize productivity, most robots have been pushed to work near their hardware design limits. To further improve the performance of industrial robots in terms of speed and accuracy, advanced control approaches become the key technology.

In regards to controlling the motion of an industrial robot, there are two important topics for cost reduction and performance improvement [2, 3]:

- motion planning: given a task, generating a feasible motion reference that optimally drives the robot from the initial configuration to the target configuration
- motion control: given the optimal motion reference, performing the movement on a physical robot exactly as intended

As an important topic in the field of robotics, motion planning has been studied for more than three decades. However due to the complicated nonlinear mapping between robot configuration space and the workspace, as well as complicated robot dynamics constraints, it remains challenging to find a feasible and optimal trajectory efficiently. One practical solution to the problem of motion planning is generating time optimal trajectories that accommodate kinematic and dynamic constraints with manually selected key points This solution is simple and effective in a mass production system where the manufacturing schedule only changes occasionally. In such a production process which requires low flexibility, it is allowed to obtain a good predefined motion for robots using time consuming manual tuning. However, in the current era of automation and data exchange in manufacturing, existing simple solutions do not meet the requirement of mass customization. In order to achieve highly flexible production, robot motion planning should be performed in an efficient and automated way.

Motion control is another important topic to achieve high performance of the robot's controller. The most of current motion controllers utilize a standard assumption that robot manipulation involves only rigid bodies and ideal transmission components. However this assumption can be considered valid only for slow motion and small interaction forces [4]. In practice, static and dynamic deflections introduced by mechanical flexibility are non-negligible when a robot is performing high acceleration motion with heavy payload. The mechanical flexibility comes from two sources: compliant transmission components and slender structural design. Today, high volume production in modern manufacturing systems is pushing robots to operate at their speed and payload limits, which is expensive to address by replacing robot hardware. In order to meet the stringent performance requirement, mechanical flexibility should also be taken into account in the design of the motion controller. These flexibilities may also be considered in the motion planning stage.

1.2 Motivation and Contribution

Neuroadaptive Control for Trajectory Tracking of Flexible Joint Robots

Most industrial robots utilize the indirect drive mechanism. The indirect drive design adopts actuators with high speed and low torque output, and transmission units with large gear ratios. Though this design effectively reduces the cost of industrial robots, non-negligible dynamic deflection is introduced when a robot is performing high speed motion. Furthermore, mismatched sensing and control makes it difficult to estimate and compensate for model uncertainties. In this dissertation, a neuroadaptive control, which is essentially a neural network based adaptive backstepping control approach, is proposed to deal with the joint flexibility and model uncertainty. The stability of the proposed approach is analyzed using Lyapunov stability theory. A data-driven approach is also proposed for the training of the neural network, which is used to compensate for model uncertainty. The effectiveness of the proposed controller is verified by simulation of a 6-axis industrial robot.

Zero Time Delay Input Shaping for Smooth Settling of Industrial Robots

Precise motion control is desired in a variety of industrial robot applications. In order to achieve precise and rapid rest-to-rest motion, overshoot and residual vibrations should be minimized. In this dissertation, a modified input shaping approach is developed to address these problems. The time delay introduced by conventional input shaping technique is fully compensated in the proposed approach. Experimental results on an industrial robot show the effectiveness of the proposed approach.

Modified Zero Time Delay Input Shaping for Industrial Robot with Flexible Link End-Effector

Input shaping is an effective approach for vibration suppression in a variety of applications. However the time delay introduced by input shaping is not desired in some applications. Current techniques that try to reduce the time delay do not guarantee zero time delay or may cause non-smooth motion, which is harmful for the service life of the actuators. In order to guarantee zero time delay, as well as the smoothness of motion, a modified zero time delay input shaping is proposed in this work. Experimental results are used to show the advantage of the proposed approach.

Robot Motion Planning Based on Efficient Trajectory Optimization

Robot motion planning, including path planning and trajectory planning, can be formulated and solved as a general optimal control problem. However since no analytical solutions for nonlinear optimal control problems can be found easily, existing motion planning techniques decompose path planning and trajectory planning as two independent problems and typically result in suboptimal solutions. In this dissertation, an efficient numerical method for trajectory optimization is investigated as one practical solution to solve the nonlinear optimal control problem for motion planning. The effectiveness of the proposed approach is demonstrated using examples of planar and spatial robots.

Optimal Vibration Suppression for Industrial Robot with Flexible End-Effector

Industrial robots with flexibility are inherently under-actuated mechanical systems. When robot end-effector shows flexibility due to the slender structural design, no external sensor is available for feedback control. Model based feedforward control is one practical choice for vibration suppression of robots with flexible end-effector. Though input shaping is proven to be a simple and effective approach, there is no guarantee about optimality and feasibility under constraints of actuator limitations. This dissertation proposes to deal with vibration suppression as an optimal control problem. Using the proposed efficient trajectory optimization technique, the optimal control problem for vibration suppression can be solved effectively. Simulations and experiments on an industrial robot have been performed to demonstrate the effectiveness of the proposed approach.

1.3 Dissertation Organization

The rest of this dissertation is organized in two parts. In Part I, a series of works related to intelligent control techniques are presented. Chapter 2 presents the design of trajectory tracking controller for robots with flexible joints. The design and stability analysis of a backstepping control is presented to compensate the flexibility. The design and training method of a neural network to compensate model uncertainties in the mismatched system is also presented. Chapter 3 presents a zero time delay input shaping approach for smooth settling of industrial robots with flexibility. Experimental results on a heavy load industrial robot are presented. Chapter 4 presents a modified zero time delay input shaping approach for residual vibration suppression of industrial robots with flexible end-effector. The modification focuses on the improvement of smoothness of the control signal generated by the zero time delay input shaping.

Part II discusses intelligent planning technique. Chapter 5 presents an efficient numerical method for trajectory optimization. The proposed trajectory optimization is implemented to solve robot path planning and trajectory planning problems simultaneously under both kinematic and dynamic constraints. Chapter 6 presents optimal vibration suppression based on the efficient numerical method used for motion planning.

Chapter 7 concludes this dissertation. Possible extensions of this dissertation research are discussed as well as future directions of this research.

Part I

Intelligent Control

Chapter 2

Neuroadaptive Control for Trajectory Tracking of Flexible Joint Robots

2.1 Introduction

Today, most industrial robots have indirect drives for high power/weight ratio and low cost. However, it is hard for indirect drive robots to achieve high trajectory tracking accuracy because of the flexibilities in transmission units in each robot joint [5]. Such flexibilities introduce time-varying mismatches between the positions of actuators and the driven links, which will result in degradation of the tracking performance [4].

Several control approaches have been proposed for robots with flexible transmission units. Some of these approaches require an accurate robot dynamic model: e.g., feedback linearization [6], singular perturbation based approach [7], and the model based feedforward control [8]. Other approaches are model free approaches: e.g., iterative learning control [9] and nonparametric learning control based on Neural Network (NN) techniques [10].

The model based controller relies on a good model. If the model is either too simple to accommodate all complex characteristics in the robots, or too complicated to identify actual dynamics and parameters, the performance of the controller will be poor due to modeling errors. On the other hand, model free approaches can provide reasonable performance most of time because no analytic model is required except that the tuning of the controller may be time consuming or data inefficient. For example, iterative learning control may require many iterations before a good control input can be learned for a specific trajectory while NN always requires large data sets for training the controller.

To address such problems, in this chapter, a neuroadaptive controller, which is essentially a neural network based adaptive backstepping control, is proposed. Instead of only using a dynamic model of the robot, an auxiliary model parameterized by a neural network is used to approximate the modeling error. The controller is designed in two stages: an off-line training stage and an online adaptive training stage. To train for a specific trajectory, it takes only at most two iterations until convergence and only requires the data from the first iteration.

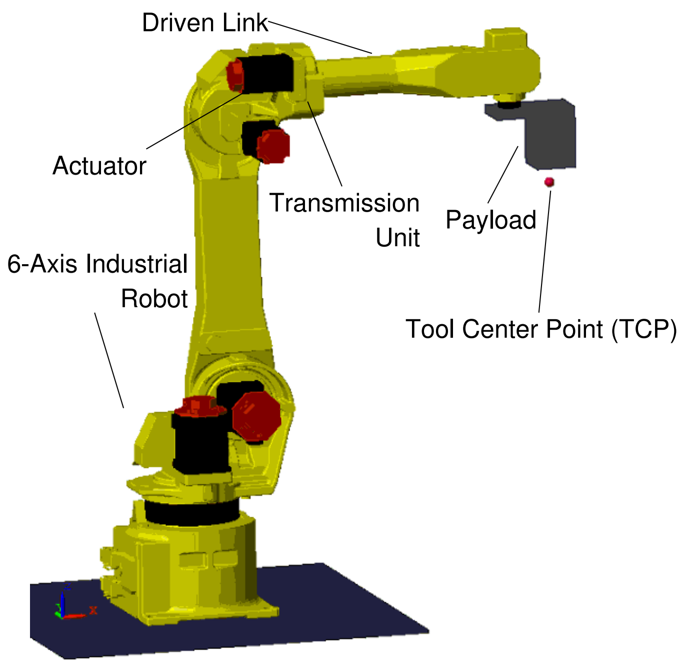


Figure 2.1: 6-axis indirect drive robot

2.2 Neuroadaptive Control of Indirect Drive Robots

Dynamical System Model

For a N DOF industrial robot, there are $2N$ generalized coordinates [11, 12, 4]:

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{q}^T & \boldsymbol{\theta}^T \end{bmatrix}^T \in \mathbb{R}^{2N}$$

where $\boldsymbol{q} \in \mathbb{R}^N$ refers to the link positions, and $\boldsymbol{\theta} \in \mathbb{R}^N$ refers to the actuator positions.

The dynamic model of the system is

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{d}_\ell + \underbrace{\boldsymbol{K}(\boldsymbol{R}^{-1}\boldsymbol{\theta} - \boldsymbol{q}) + \boldsymbol{D}(\boldsymbol{R}^{-1}\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{q}})}_{\boldsymbol{y}} \tag{2.1a}$$

$$\boldsymbol{J}_m\ddot{\boldsymbol{\theta}} = \boldsymbol{\tau} + \boldsymbol{d}_m + \boldsymbol{R}^{-1}\left[\boldsymbol{K}(\boldsymbol{q} - \boldsymbol{R}^{-1}\boldsymbol{\theta}) + \boldsymbol{D}(\dot{\boldsymbol{q}} - \boldsymbol{R}^{-1}\dot{\boldsymbol{\theta}})\right] \tag{2.1b}$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{N\times N}$, $\boldsymbol{J}_m \in \mathbb{R}^{N\times N}$ are the inertia matrices for link motion and actuator motion respectively; $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \in \mathbb{R}^N$ is Coriolis and centrifugal force; $\boldsymbol{G}(\boldsymbol{q}) \in \mathbb{R}^N$ is the gravity term; $\boldsymbol{y} = \boldsymbol{K}(\boldsymbol{R}^{-1}\boldsymbol{\theta} - \boldsymbol{q}) + \boldsymbol{D}(\boldsymbol{R}^{-1}\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{q}})$ is the transmission torque between the actuator and the robot link; $\boldsymbol{\tau} \in \mathbb{R}^N$ is torque generated by actuators; $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{D} \in \mathbb{R}^{N\times N}$ are diagonal matrices representing stiffness and damping of the transmission units respectively; $\boldsymbol{R} \in \mathbb{R}^{N\times N}$ is diagonal matrix representing gear ratio; $\boldsymbol{d}_\ell \in \mathbb{R}^N$ and $\boldsymbol{d}_m \in \mathbb{R}^N$ are disturbances applied the links and actuators respectively. The input to the system is the actuator torque $\boldsymbol{\tau}$, and output of the system is the link position $\boldsymbol{q}$.

The disturbances $\boldsymbol{d}_\ell$ and $\boldsymbol{d}_m$ include complex friction, transmission error mentioned in [13, 14], and actuator-link interaction mentioned in [4]. It is difficult to build a parametric model for all the disturbances, but in general, they can be modelled as $\boldsymbol{d}_\ell \approx \boldsymbol{d}_\ell(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$, $\boldsymbol{d}_m \approx \boldsymbol{d}_m(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$.

Backstepping Control

Backstepping control, as discussed in [15], is a control method that is developed for nonlinear dynamical systems with recursive structure. The design of backstepping control involves designing controllers that progressively stabilize a series of subsystems.

An indirect drive robot has such a kind of recursive structure. The first subsystem is expressed as (2.1a), and the second subsystem is expressed as (2.1b). Design of backstepping control of indirect drive robot consists of two steps. The first step designs the control law for (2.1a) with the transmission torque $\boldsymbol{y}$ as input, and the second step designs the control law for (2.1b) with motor torque $\boldsymbol{\tau}$ as input. In the first step, the designed controller stabilizes the trajectory tracking error to 0. In the second step, the designed controller stabilizes the transmission torque error to 0.

First step: Letting the reference trajectory of an indirect drive robot be $\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d, \ddot{\boldsymbol{q}}_d$, the trajectory tracking error is defined as

$$\boldsymbol{e} = \boldsymbol{q}_d - \boldsymbol{q} \tag{2.2}$$

According to [16], a filtered-error term $\boldsymbol{r}$ can be defined as

$$\boldsymbol{r} = \dot{\boldsymbol{e}} + \boldsymbol{K_p}\boldsymbol{e} \tag{2.3}$$

where $\boldsymbol{K}_p \in \mathbb{R}^{N\times N}$ is a positive definite gain matrix with the minimum singular value $\sigma_{\min}(\boldsymbol{K}_p) > 0$. Since (2.3) can be considered as a stable system with $\boldsymbol{r}$ as input, and $\boldsymbol{e}$ as output, therefore stabilizing tracking error $\boldsymbol{e}$ is equivalent to stabilizing filtered error $\boldsymbol{r}$.

$$\lim_{t\to\infty} \boldsymbol{r} = 0 \Rightarrow \lim_{t\to\infty} \boldsymbol{e} = 0 \tag{2.4}$$

In order to stabilize the tracking error $\boldsymbol{e}$, a desired transmission torque $\boldsymbol{y}_d$ can be designed as [16]

$$\boldsymbol{y}_d = \boldsymbol{K}_r\boldsymbol{r} + \boldsymbol{M}(\boldsymbol{q})(\ddot{\boldsymbol{q}}_d + \boldsymbol{K}_p\dot{\boldsymbol{e}}) + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})(\dot{\boldsymbol{q}}_d + \boldsymbol{K}_p\boldsymbol{e}) + \boldsymbol{G}(\boldsymbol{q}) - \boldsymbol{d}_\ell$$

where $\boldsymbol{K}_r \in \mathbb{R}^{N\times N}$ is a positive definite gain matrix.

Second step: The error between desired transmission torque $\boldsymbol{y}_d$ and the actual interaction torque through flexible transmission unit $\boldsymbol{y}$ is

$$\boldsymbol{s} = \boldsymbol{y}_d - \boldsymbol{y} \tag{2.5}$$

In order to design a control law that stabilize the transmission torque error $\boldsymbol{s}$ to 0, we first construct a Lyapunov function L,

$$L = \frac{1}{2}\boldsymbol{r}^T\boldsymbol{M}(\boldsymbol{q})\boldsymbol{r} + \frac{1}{2}\boldsymbol{s}^T\boldsymbol{A}\boldsymbol{s}$$

where $\boldsymbol{A} = \boldsymbol{J}_m\boldsymbol{R}\boldsymbol{D}^{-1}$. Since $\boldsymbol{J}_m$, $\boldsymbol{R}$, $\boldsymbol{D}$ are diagonal, positive definite matrices, $\boldsymbol{A}$ is positive definite.

The time derivative of the filtered error in the first step can be derived as:

$$\begin{aligned}\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{r}} &= \boldsymbol{M}(\boldsymbol{q})[\ddot{\boldsymbol{e}} + \boldsymbol{K}_p\dot{\boldsymbol{e}}] \\ &= \boldsymbol{M}(\boldsymbol{q})[\ddot{\boldsymbol{q}}_d - \ddot{\boldsymbol{q}} + \boldsymbol{K}_p\dot{\boldsymbol{e}}] \\ &= \boldsymbol{M}(\boldsymbol{q})(\ddot{\boldsymbol{q}}_d + \boldsymbol{K}_p\dot{\boldsymbol{e}}) - \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}\end{aligned}$$

Substituting the dynamic model (2.1a), the formulation can be further derived as:

$$\begin{aligned}\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{r}} &= \boldsymbol{M}(\boldsymbol{q})(\ddot{\boldsymbol{q}}_d + \boldsymbol{K}_p\dot{\boldsymbol{e}}) - [\boldsymbol{d}_\ell + \boldsymbol{K}(\boldsymbol{R}^{-1}\boldsymbol{\theta} - \boldsymbol{q}) + \boldsymbol{D}(\boldsymbol{R}^{-1}\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{q}}) \\ &\quad -\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{G}(\boldsymbol{q})] \\ &= \boldsymbol{M}(\boldsymbol{q})(\ddot{\boldsymbol{q}}_d + \boldsymbol{K}_p\dot{\boldsymbol{e}}) + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) - \boldsymbol{d}_\ell - \boldsymbol{y} \\ &= \boldsymbol{M}(\boldsymbol{q})(\ddot{\boldsymbol{q}}_d + \boldsymbol{K}_p\dot{\boldsymbol{e}}) + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})[\dot{\boldsymbol{q}}_d + \boldsymbol{K}_p\boldsymbol{e} - \boldsymbol{r}] + \boldsymbol{G}(\boldsymbol{q}) - \boldsymbol{d}_\ell - \boldsymbol{y} \\ &= \underbrace{\boldsymbol{M}(\boldsymbol{q})(\ddot{\boldsymbol{q}}_d + \boldsymbol{K}_p\dot{\boldsymbol{e}}) + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})(\dot{\boldsymbol{q}}_d + \boldsymbol{K}_p\boldsymbol{e}) + \boldsymbol{G}(\boldsymbol{q}) - \boldsymbol{d}_\ell}_{\boldsymbol{f}} \\ &\quad -\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{r} + \underbrace{\boldsymbol{y}_d - \boldsymbol{y}}_{\boldsymbol{s}} - \boldsymbol{y}_d\end{aligned}$$

The time derivative of the transmission torque error can be derived as:

$$\begin{aligned}\boldsymbol{A}\dot{\boldsymbol{s}} &= \boldsymbol{A}(\dot{\boldsymbol{y}}_d - \dot{\boldsymbol{y}}) \\ &= \boldsymbol{A}[\dot{\boldsymbol{y}}_d - \boldsymbol{K}(\boldsymbol{R}^{-1}\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{q}}) + \boldsymbol{D}(\boldsymbol{R}^{-1}\ddot{\boldsymbol{\theta}} - \ddot{\boldsymbol{q}})] \\ &= \boldsymbol{A}[\dot{\boldsymbol{y}}_d - \boldsymbol{K}(\boldsymbol{R}^{-1}\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{q}})] - \boldsymbol{A}\boldsymbol{D}(\boldsymbol{R}^{-1}\ddot{\boldsymbol{\theta}} - \ddot{\boldsymbol{q}}) \\ &= \boldsymbol{A}[\dot{\boldsymbol{y}}_d - \boldsymbol{K}(\boldsymbol{R}^{-1}\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{q}})] - \boldsymbol{J}_m\boldsymbol{R}(\boldsymbol{R}^{-1}\ddot{\boldsymbol{\theta}} - \ddot{\boldsymbol{q}}) \\ &= \boldsymbol{A}[\dot{\boldsymbol{y}}_d - \boldsymbol{K}(\boldsymbol{R}^{-1}\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{q}})] + \boldsymbol{J}_m\boldsymbol{R}\ddot{\boldsymbol{q}} - \boldsymbol{J}_m\ddot{\boldsymbol{\theta}}\end{aligned}$$

Substituting the dynamic model (2.1b), the formulation can be further derived as:

$$\begin{aligned}\boldsymbol{A}\dot{\boldsymbol{s}} &= \boldsymbol{A}[\dot{\boldsymbol{y}}_d - \boldsymbol{K}(\boldsymbol{R}^{-1}\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{q}})] + \boldsymbol{J}_m\boldsymbol{R}\ddot{\boldsymbol{q}} - [\boldsymbol{\tau} + \boldsymbol{d}_m - \boldsymbol{R}^{-1}\boldsymbol{y}] \\ &= \underbrace{\boldsymbol{A}\left[\dot{\boldsymbol{y}}_d - \boldsymbol{K}(\boldsymbol{R}^{-1}\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{q}})\right] + \boldsymbol{J}_m\boldsymbol{R}\ddot{\boldsymbol{q}} + \boldsymbol{R}^{-1}\boldsymbol{y} - \boldsymbol{d}_m}_{\boldsymbol{h}} - \boldsymbol{\tau}\end{aligned}$$

After substituting $\boldsymbol{y}_d$ in the first step, and the time derivatives above, the time derivative of L is:

$$\begin{aligned}\dot{L} &= \tfrac{1}{2}\boldsymbol{r}^T\dot{\boldsymbol{M}}(\boldsymbol{q})\boldsymbol{r} + \boldsymbol{r}^T\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{r}} + \boldsymbol{s}^T\boldsymbol{A}\dot{\boldsymbol{s}} \\ &= \tfrac{1}{2}\boldsymbol{r}^T\dot{\boldsymbol{M}}(\boldsymbol{q})\boldsymbol{r} + \boldsymbol{r}^T[\boldsymbol{f} - \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{r} + \boldsymbol{s} - \boldsymbol{y}_d] + \boldsymbol{s}^T(\boldsymbol{h} - \boldsymbol{\tau}) \\ &= \tfrac{1}{2}\boldsymbol{r}^T\left(\dot{\boldsymbol{M}}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\right)\boldsymbol{r} + \boldsymbol{r}^T\boldsymbol{s} - \boldsymbol{s}^T\boldsymbol{r} + \boldsymbol{r}^T(\boldsymbol{f} - \boldsymbol{y}_d) + \boldsymbol{s}^T(\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{\tau}) \\ &= \tfrac{1}{2}\boldsymbol{r}^T\left(\dot{\boldsymbol{M}}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\right)\boldsymbol{r} + \boldsymbol{r}^T(\boldsymbol{f} - \boldsymbol{y}_d) + \boldsymbol{s}^T(\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{\tau})\end{aligned}$$

With the skew symmetric property of $\dot{\boldsymbol{M}}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ [12], the time derivative of L can be further simplified:

$$\begin{aligned}\dot{L} &= \boldsymbol{r}^T(\boldsymbol{f} - \boldsymbol{y}_d) + \boldsymbol{s}^T(\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{\tau}) \\ &= -\boldsymbol{r}^T\boldsymbol{K}_r\boldsymbol{r} + \boldsymbol{s}^T(\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{\tau})\end{aligned}$$

Letting $\boldsymbol{\tau} = \boldsymbol{h} + \boldsymbol{r} + \boldsymbol{K}_s \boldsymbol{s}$, where $\boldsymbol{K}_s \in \mathbb{R}^{N \times N}$ is a positive definite gain matrix. The time derivative of L is then negative definite.

$$\dot{L} = -\boldsymbol{r}^T \boldsymbol{K}_r \boldsymbol{r} - \boldsymbol{s}^T \boldsymbol{K}_s \boldsymbol{s} < 0$$

According to Lyapunov stability theory, the dynamical system is asymptotically stable, thus $\lim_{t\to\infty} \boldsymbol{r} = 0$, $\lim_{t\to\infty} \boldsymbol{s} = 0$. According to (2.4), $\lim_{t\to\infty} \boldsymbol{e} = 0$.

To sum up, backstepping controller can be designed based on an accurate model of the system as

$$\boldsymbol{y}_d = \boldsymbol{f} + \boldsymbol{K}_r \boldsymbol{r} \tag{2.6a}$$

$$\boldsymbol{\tau} = \boldsymbol{h} + \boldsymbol{r} + \boldsymbol{K}_s \boldsymbol{s} \tag{2.6b}$$

In this approach, two nonlinear functions $\boldsymbol{f}$ and $\boldsymbol{h}$ are derived to represent the physical model of the dynamic system.

NN Based Adaptive Backstepping Control

The ideal backstepping controller (2.6a) and (2.6b) is impractical since the exact $\boldsymbol{f}$ and $\boldsymbol{h}$ terms are not available due to the complexity of any real physical system. Moreover, though $\dot{\boldsymbol{y}}_d$ and $\ddot{\boldsymbol{q}}$ required in the calculation of $\boldsymbol{h}$ may be estimated using the system model or by finite difference, the estimation could be difficult due to noise. One way to accommodate the uncertainty in $\boldsymbol{f}$ and $\boldsymbol{h}$ is to add a robust feature to the controller, but this approach may not be able to make tracking error small if large uncertainty exists. Another way is to use an auxiliary model that approximates the modeling error by an artificial NN. The backstepping controller can then be designed as

$$\boldsymbol{y}_d = \boldsymbol{f}_n + \hat{\boldsymbol{f}} + \boldsymbol{K}_r \boldsymbol{r} \tag{2.7a}$$

$$\boldsymbol{\tau} = \boldsymbol{h}_n + \hat{\boldsymbol{h}} + \boldsymbol{r} + \boldsymbol{K}_s \boldsymbol{s} \tag{2.7b}$$

where $\boldsymbol{f}_n$ and $\boldsymbol{h}_n$ are the nominal system model terms obtained using computer aided design software. $\hat{\boldsymbol{f}}$ and $\hat{\boldsymbol{h}}$ are the auxiliary model terms that approximate the difference between the actual system and the nominal model. The difference includes estimation errors in the inertia parameters of the robot, estimation error in the transmission units stiffness and damping parameters, unmodeled complex frictions, and transmission errors.

Radial basis function (RBF) network, also known as the functional-link neural network (FLNN) in [16], is chosen to build this auxiliary model. The reason to use RBF network is that RBF neural network has the ability to approximate an arbitrary nonlinear function with very simple structure (only one hidden layer), as shown in [17].

The terms $\hat{\boldsymbol{f}}$ and $\hat{\boldsymbol{h}}$ can be written as functions of $\boldsymbol{X}$, where $\boldsymbol{X} \equiv \left[\boldsymbol{q}_d^T, \dot{\boldsymbol{q}}_d^T, \ddot{\boldsymbol{q}}_d^T, \boldsymbol{q}^T, \boldsymbol{\theta}^T, \dot{\boldsymbol{q}}^T, \dot{\boldsymbol{\theta}}^T\right]^T$ is the augmented state that includes the reference trajectory $\{\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d, \ddot{\boldsymbol{q}}_d\}$. The auxiliary model can then be formulated as

$$\begin{aligned} \hat{\boldsymbol{f}}(\boldsymbol{X}) &= \kappa_1 \sum_{i=1}^{U} \boldsymbol{w}_1^i \phi_i(\boldsymbol{X}) = \kappa_1 \boldsymbol{W}_1^T \boldsymbol{\Phi}(\boldsymbol{X}) \\ \hat{\boldsymbol{h}}(\boldsymbol{X}) &= \kappa_2 \sum_{i=1}^{U} \boldsymbol{w}_2^i \phi_i(\boldsymbol{X}) = \kappa_2 \boldsymbol{W}_2^T \boldsymbol{\Phi}(\boldsymbol{X}) \end{aligned} \tag{2.8}$$

where $\kappa_1 \in \mathbb{R}$ and $\kappa_2 \in \mathbb{R}$ are two constant parameters that scale the neural network weights. U is the number of neurons used in the RBF network, and $\boldsymbol{\Phi}(\boldsymbol{X}) = \left[\phi_1(\boldsymbol{X}), \cdots, \phi_U(\boldsymbol{X})\right]^T$ is the vector of activation functions. $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{U\times N}$ are scaled weights of the neural networks, where the i^{th} column of the transposed weight matrices $\boldsymbol{W}_1^T, \boldsymbol{W}_2^T \in \mathbb{R}^{N\times U}$ are $\boldsymbol{w}_1^i$ and $\boldsymbol{w}_2^i$.

Gaussian radial basis function is one common choice for the activation function in the RBF network. A Gaussian radial basis function used in the i^{th} neuron can be formulated as

$$\phi_i(\boldsymbol{x}) = \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Lambda}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right\} \tag{2.9}$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^n$ is the center of the Gaussian radial basis function $\phi_i(\boldsymbol{x})$, and $\boldsymbol{\Lambda}_i \in \mathbb{R}^{n\times n}$ can be called the width parameter. Choosing the center and width of the Gaussian radial basis function $\phi_i(\boldsymbol{x})$ will be introduced in Section 2.3 as the initial training stage. After the center and width parameters are determined, the weights of RBF network can be trained using adaptive control. The adaptation law is designed as

$$\dot{\boldsymbol{W}}_1 = \boldsymbol{F}_1\kappa_1\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{r}^T - \gamma_1\boldsymbol{F}_1\boldsymbol{W}_1 \tag{2.10a}$$

$$\dot{\boldsymbol{W}}_2 = \boldsymbol{F}_2\kappa_2\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{s}^T - \gamma_2\boldsymbol{F}_2\boldsymbol{W}_2 \tag{2.10b}$$

where $\boldsymbol{F}_1 \in \mathbb{R}^{U\times U}$ and $\boldsymbol{F}_2 \in \mathrm{R}^{U\times U}$ are positive definite gain matrices, $\gamma_1 \in \mathbb{R}$, and $\gamma_2 \in \mathbb{R}$ are two extra gains. The uniform ultimate boundedness, which has once been introduced by [16], will be proved in section 2.4 for both tracking error and neural network weights estimation error.

2.3 Two Stage Training Approach for Neural Network

The training of a RBF network using Gaussian radial basis functions can be divided into two stages [18]. The first stage determines the placements of the localized units, i.e. Gaussian units in input space. The second stage then determines the weights of a RBF network. In this chapter, these two stages are called initial training stage and online training stage.

Initial Training Stage

The centers of the Gaussian radial basis functions of a RBF network should be uniformly and densely distributed in the domain of the function to guarantee a small approximation error [16]. The width can then be chosen to be the maximum distance between adjacent centers. However, this is hard to realize by simple discretization if the domain of function has high dimensionality because too many neurons/radial basis functions are required to cover the function domain. For example, in section 2.2, the input to the RBF network X could be a 42 dimensional vector if $N = 6$. Even only 2 levels are used for the discretization of each dimension, the required neuron number should be $2^{42} \approx 4.3980 \times 10^{12}$, which is even larger than the number of neurons in a human brain. To avoid this problem, an alternative data-driven approach is proposed in this section.

Since any trajectory of a robot can be parametrized by time as $\boldsymbol{X} = \boldsymbol{X}(t)$, the domain of function can be limited in a one-dimensional manifold in the high dimensional function domain. Instead of choosing centers in the high dimensional function domain, the centers can be determined in the low dimensional manifold, as shown in Fig.2.2. The required number of neurons can then be reduced.The center and width parameters can be first determined in the low dimensional manifold using clustering approaches like k-means, as shown in [19], then transfer back to the high dimensional space.

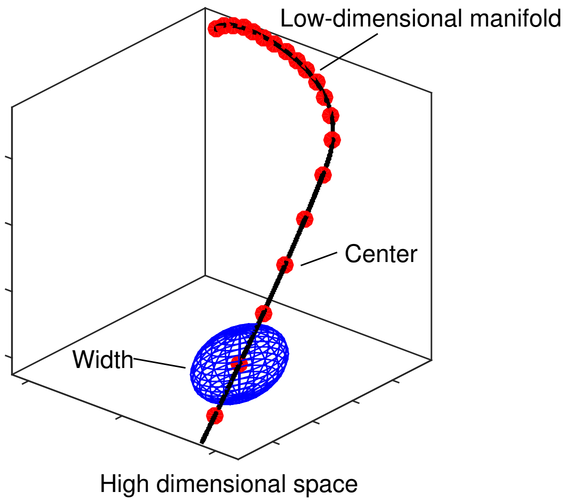


Figure 2.2: Distributing neurons / radial basis functions in the low-dimensional manifold. Line: low-dimensional manifold. Points: centers of radial basis functions. Ellipsoid: width of radial basis functions

Suppose the dimension of the augmented state $\boldsymbol{X}$ is n, then an experiment data set that contains H data points can be denoted as a $H \times n$ matrix as $\boldsymbol{X}_S \equiv [\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_H]^T$. Principle component analysis (PCA) can be implemented for dimension reduction [20]. Let the singular value decomposition of $\boldsymbol{X}_S$ be

$$\boldsymbol{X}_S = \boldsymbol{P}\boldsymbol{S}\boldsymbol{Q}^T = \sum_{i=1}^{n} \boldsymbol{p}_i \rho_i \boldsymbol{\nu}_i^T \tag{2.11}$$

where $\boldsymbol{P} = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_H]$ is an $H \times H$ orthogonal matrix, $\boldsymbol{Q} = [\boldsymbol{\nu}_1, \cdots, \boldsymbol{\nu}_n]$ is an $n \times n$ orthogonal matrix, and $\boldsymbol{S}$ is an $H \times n$ diagonal matrix with $\boldsymbol{S}[i,i] = \rho_i$, where ρ_i is the i-th singular value of $\boldsymbol{X}_S$. Since $\boldsymbol{X}_S$ is actually representing a low-dimensional manifold, the first $k (k < n)$ singular values will dominate, and $\forall i > k, \rho_i \approx 0$. Thus the data set can be well approximated by a low-rank approximation $\hat{\boldsymbol{X}}_{Sk}$, as in [21],

$$\hat{\boldsymbol{X}}_{Sk} = \sum_{i=1}^{k} \boldsymbol{p}_i \rho_i \boldsymbol{\nu}_i^T \tag{2.12}$$

Though this approximation is only linear, k could still be much smaller than n. This approximation projects all data points approximately to a hyperplane spanned by $\{\boldsymbol{\nu}_1, \cdots, \boldsymbol{\nu}_k\}$. The center and width of the Gaussian radial basis functions can be designed on the hyper plane then. The centers can be designed to be uniformly distributed along the projection of the low dimensional manifold on the hyperplane, and the widths can be designed to be constants. Suppose there are U neurons in the neural network. Let the $\{\boldsymbol{\mu}_{p1}, \boldsymbol{\mu}_{p2}, \cdots, \boldsymbol{\mu}_{pU}\}$ be the coordinates of the centers of the Gaussian radial basis functions on the hyper plane, and the corresponding width parameters on the hyper plane be $\{\boldsymbol{\Lambda}_{p1}, \boldsymbol{\Lambda}_{p2}, \cdots, \boldsymbol{\Lambda}_{pU}\}$. Let $\boldsymbol{Q}_p \equiv [\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \cdots, \boldsymbol{\nu}_k]$. The centers $\{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_U\}$ and the widths $\{\boldsymbol{\Lambda}_1, \cdots, \boldsymbol{\Lambda}_U\}$ of the Gaussian radial basis functions in the original n-dimensional space are calculated as

$$\begin{array}{rcl} \{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_U\} & = & \{\boldsymbol{Q}_p\boldsymbol{\mu}_{p1}, \cdots, \boldsymbol{Q}_p\boldsymbol{\mu}_{pU}\} \\ \{\boldsymbol{\Lambda}_1, \cdots, \boldsymbol{\Lambda}_U\} & = & \{\boldsymbol{Q}_p\boldsymbol{\Lambda}_{p1}\boldsymbol{Q}_p^T, \cdots, \boldsymbol{Q}_p\boldsymbol{\Lambda}_{pU}\boldsymbol{Q}_p^T\} \end{array} \tag{2.13}$$

Online Training Stage

After the initial training stage, the vector of activation functions $\boldsymbol{\Phi}(\boldsymbol{X})$ is determined. The weights $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ can then be learned to minimize the difference between the nominal model and the actual system.

The optimal RBF network weights $\boldsymbol{W}_1^*$ and $\boldsymbol{W}_2^*$, which minimize the model difference can be defined as

$$\begin{array}{l} \boldsymbol{W}_1^* = \arg\min\limits_{\boldsymbol{W}_1}(\sup\limits_{\boldsymbol{X}}\|\boldsymbol{f}(\boldsymbol{X}) - \boldsymbol{f}_n(\boldsymbol{X}) - \kappa_1\boldsymbol{W}_1^T\boldsymbol{\Phi}(\boldsymbol{X})\|) \\ \boldsymbol{W}_2^* = \arg\min\limits_{\boldsymbol{W}_2}(\sup\limits_{\boldsymbol{X}}\|\boldsymbol{h}(\boldsymbol{X}) - \boldsymbol{h}_n(\boldsymbol{X}) - \kappa_2\boldsymbol{W}_2^T\boldsymbol{\Phi}(\boldsymbol{X})\|) \end{array} \tag{2.14}$$

Since the actual dynamic system model $\boldsymbol{f}$ and $\boldsymbol{h}$ are not available, no supervised learning technique can be used to train this neural network. Instead of using supervised learning techniques, the RBF network weights $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are trained using adaptive control approach as (2.10a) and (2.10b) in this chapter. It will be proved in section 2.4 that the network weights are uniformly ultimately bounded.

2.4 Stability Analysis

This section shows the uniform ultimate boundedness of both trajectory tracking error and the neural network weights. This can be proved by showing the uniform ultimate boundedness of the filtered error $\boldsymbol{r}$ and the weight difference $\tilde{\boldsymbol{W}}_1 = \boldsymbol{W}_1^* - \boldsymbol{W}_1, \tilde{\boldsymbol{W}}_2 = \boldsymbol{W}_2^* - \boldsymbol{W}_2$.

We introduce three assumptions. a) The domain of $\boldsymbol{f}$ is compact and simply connected; b) the domain of $\boldsymbol{h}$ is compact and simply connected; and c) $\boldsymbol{f}$ and $\boldsymbol{h}$ are continuous functions. According to [16, 17], the universal approximation property of radial basis function networks holds. This suggests that the optimal approximation error should be bounded within the domains of the functions, as

$$\begin{array}{rclcl} \|\boldsymbol{\epsilon}_1^*\| & = & \|\boldsymbol{f} - \boldsymbol{f}_n - \kappa_1\boldsymbol{W}_1^{*T}\boldsymbol{\Phi}(\boldsymbol{X})\| & \leq & \epsilon_{N1} \\ \|\boldsymbol{\epsilon}_2^*\| & = & \|\boldsymbol{h} - \boldsymbol{h}_n - \kappa_2\boldsymbol{W}_2^{*T}\boldsymbol{\Phi}(\boldsymbol{X})\| & \leq & \epsilon_{N2} \end{array} \tag{2.15}$$

where $\boldsymbol{\epsilon}_1^*$ and $\boldsymbol{\epsilon}_2^*$ are the optimal approximation errors, and ϵ_{N1}, ϵ_{N2} are the upper bounds of $\|\boldsymbol{\epsilon}_1^*\|$ and $\|\boldsymbol{\epsilon}_2^*\|$. Furthermore, $\kappa_1 \boldsymbol{W}_1^*$ and $\kappa_2 \boldsymbol{W}_2^*$ can be chosen to be constant and bounded matrices, as

$$\begin{array}{rcl} \kappa_1 \|\boldsymbol{W}_1^*\|_F & \leq & W_{B1} \\ \kappa_2 \|\boldsymbol{W}_2^*\|_F & \leq & W_{B2} \end{array} \tag{2.16}$$

where $\|\cdot\|_F$ is the Frobenius norm; W_{B1} and W_{B2} are upper bounds of the norm of neural network weights.

To analyze the stability, a Lyapunov function candidate can be chosen as

$$\begin{array}{rcl} V & = & \frac{1}{2}\boldsymbol{r}^T \boldsymbol{M}(\boldsymbol{q})\boldsymbol{r} + \frac{1}{2}\boldsymbol{s}^T \boldsymbol{A}\boldsymbol{s} + \frac{1}{2}\mathrm{tr}\{\tilde{\boldsymbol{W}}_1^T \boldsymbol{F}_1^{-1} \tilde{\boldsymbol{W}}_1\} \\ & & + \frac{1}{2}\mathrm{tr}\{\tilde{\boldsymbol{W}}_2^T \boldsymbol{F}_2^{-1} \tilde{\boldsymbol{W}}_2\} \end{array} \tag{2.17}$$

where $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix, $\boldsymbol{r}$, $\boldsymbol{s}$, and $\boldsymbol{A}$ are defined in section 2.2, $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ are defined in the adaptation law (2.10a), (2.10b). Since the optimal neural network weights $\boldsymbol{W}_1^*$ and $\boldsymbol{W}_2^*$ are constant matrices,

$$\dot{\tilde{\boldsymbol{W}}}_1 = -\dot{\boldsymbol{W}}_1 \tag{2.18a}$$

$$\dot{\tilde{\boldsymbol{W}}}_2 = -\dot{\boldsymbol{W}}_2 \tag{2.18b}$$

With the proposed controller (2.7a), (2.7b), the proposed adaptation law (2.10a), (2.10b), and (2.18a), (2.18b), the time derivative of the Lyapunov function candidate is

$$\begin{array}{rcl} \dot{V} & = & \frac{1}{2}\boldsymbol{r}^T \dot{\boldsymbol{M}}(\boldsymbol{q})\boldsymbol{r} + \boldsymbol{r}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{r}} + \boldsymbol{s}^T \boldsymbol{A}\dot{\boldsymbol{s}} \\ & & + \mathrm{tr}\{\tilde{\boldsymbol{W}}_1^T \boldsymbol{F}_1^{-1} \dot{\tilde{\boldsymbol{W}}}_1\} + \mathrm{tr}\{\tilde{\boldsymbol{W}}_2^T \boldsymbol{F}_2^{-1} \dot{\tilde{\boldsymbol{W}}}_2\} \\ & = & \frac{1}{2}\boldsymbol{r}^T [\dot{\boldsymbol{M}}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})]\boldsymbol{r} \\ & & -\boldsymbol{r}^T \boldsymbol{K}_r \boldsymbol{r} + \boldsymbol{r}^T \boldsymbol{\epsilon}_1^* + \kappa_1 \boldsymbol{r}^T \tilde{\boldsymbol{W}}_1^T \boldsymbol{\Phi}(\boldsymbol{X}) \\ & & -\boldsymbol{s}^T \boldsymbol{K}_s \boldsymbol{s} + \boldsymbol{s}^T \boldsymbol{\epsilon}_2^* + \kappa_2 \boldsymbol{s}^T \tilde{\boldsymbol{W}}_2^T \boldsymbol{\Phi}(\boldsymbol{X}) \\ & & -\mathrm{tr}\{\kappa_1 \tilde{\boldsymbol{W}}_1^T \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{r}^T - \gamma_1 \tilde{\boldsymbol{W}}_1^T \boldsymbol{W}_1\} \\ & & -\mathrm{tr}\{\kappa_2 \tilde{\boldsymbol{W}}_2^T \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{r}^T - \gamma_2 \tilde{\boldsymbol{W}}_2^T \boldsymbol{W}_2\} \end{array}$$

Using the skew symmetric property of $\dot{\boldsymbol{M}}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, the linearity of trace, and the property $\mathrm{tr}(\boldsymbol{AB}) = \mathrm{tr}(\boldsymbol{BA})$, the time derivative of V can be further manipulated as

$$\begin{array}{rcl} \dot{V} & = & -\boldsymbol{r}^T \boldsymbol{K}_r \boldsymbol{r} - \boldsymbol{s}^T \boldsymbol{K}_s \boldsymbol{s} + \boldsymbol{r}^T \boldsymbol{\epsilon}_1^* + \boldsymbol{s}^T \boldsymbol{\epsilon}_2^* - \gamma_1 \mathrm{tr}\{\tilde{\boldsymbol{W}}_1^T \tilde{\boldsymbol{W}}_1\} \\ & & + \gamma_1 \mathrm{tr}\{\boldsymbol{W}_1^{*T} \tilde{\boldsymbol{W}}_1\} - \gamma_2 \mathrm{tr}\{\tilde{\boldsymbol{W}}_2^T \tilde{\boldsymbol{W}}_2\} + \gamma_2 \mathrm{tr}\{\boldsymbol{W}_2^{*T} \tilde{\boldsymbol{W}}_2\} \\ & = & -\begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{s} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{K}_r & 0 \\ 0 & \boldsymbol{K}_s \end{bmatrix} \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{s} \end{bmatrix} + \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{s} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\epsilon}_1^* \\ \boldsymbol{\epsilon}_2^* \end{bmatrix} - \gamma_1 \mathrm{tr}\{\tilde{\boldsymbol{W}}_1^T \tilde{\boldsymbol{W}}_1\} \\ & & + \gamma_1 \mathrm{tr}\{\boldsymbol{W}_1^{*T} \tilde{\boldsymbol{W}}_1\} - \gamma_2 \mathrm{tr}\{\tilde{\boldsymbol{W}}_2^T \tilde{\boldsymbol{W}}_2\} + \gamma_2 \mathrm{tr}\{\boldsymbol{W}_2^{*T} \tilde{\boldsymbol{W}}_2\} \end{array}$$

Let $\boldsymbol{W}_i^v = [\boldsymbol{w}_i^{1T}, \cdots, \boldsymbol{w}_i^{UT}]^T$ be the vectorized $\boldsymbol{W}_i (i = 1, 2)$, where $\boldsymbol{W}_i^T \in \mathbb{R}^{N\times U}$, $\boldsymbol{w}_i^j$ is the j^{th} column of $\boldsymbol{W}_i^T$. The time derivative of the V can be written as

$$\dot{V} = \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{s} \\ \tilde{\boldsymbol{W}}_1^v \\ \tilde{\boldsymbol{W}}_2^v \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\epsilon}_1^* \\ \boldsymbol{\epsilon}_2^* \\ \gamma_1 \boldsymbol{W}_1^{*v} \\ \gamma_2 \boldsymbol{W}_2^{*v} \end{bmatrix} - \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{s} \\ \tilde{\boldsymbol{W}}_1^v \\ \tilde{\boldsymbol{W}}_2^v \end{bmatrix}^T \begin{bmatrix} \boldsymbol{K}_r & 0 & 0 & 0 \\ 0 & \boldsymbol{K}_s & 0 & 0 \\ 0 & 0 & \gamma_1 \boldsymbol{I} & 0 \\ 0 & 0 & 0 & \gamma_2 \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{s} \\ \tilde{\boldsymbol{W}}_1^v \\ \tilde{\boldsymbol{W}}_2^v \end{bmatrix} \tag{2.19}$$

According to (2.15), the optimal neural network approximation error is bounded. According to (2.16), $\boldsymbol{W}_i^{*v}$ are bounded since $\|\boldsymbol{W}_i\|_F = \sqrt{\boldsymbol{W}_i^{vT}\boldsymbol{W}_i^v} = \|\boldsymbol{W}_i^v\| (i = 1, 2)$. Then

$$\begin{aligned} & \left\| [\boldsymbol{\epsilon}_1^{*T}, \boldsymbol{\epsilon}_2^{*T}, \gamma_1 \boldsymbol{W}_1^{*vT}, \gamma_2 \boldsymbol{W}_2^{*vT}]^T \right\| \\ = \; & \sqrt{\|\boldsymbol{\epsilon}_1^*\|^2 + \|\boldsymbol{\epsilon}_2^*\|^2 + \gamma_1^2 \|\boldsymbol{W}_1^{*v}\|^2 + \gamma_2^2 \|\boldsymbol{W}_2^{*v}\|^2} \\ \le \; & \sqrt{\epsilon_{N1}^2 + \epsilon_{N2}^2 + \gamma_1^2 W_{B1}^2/\kappa_1^2 + \gamma_2^2 W_{B2}^2/\kappa_2^2} \\ \triangleq \; & b_\epsilon \end{aligned}$$

Let σ_l be

$$\sigma_l = \min_{\|\boldsymbol{x}\|=1} \boldsymbol{x}^T \begin{bmatrix} \boldsymbol{K}_r & 0 & 0 & 0 \\ 0 & \boldsymbol{K}_s & 0 & 0 \\ 0 & 0 & \gamma_1 \boldsymbol{I} & 0 \\ 0 & 0 & 0 & \gamma_2 \boldsymbol{I} \end{bmatrix} \boldsymbol{x} > 0$$

Let $\boldsymbol{\eta} \triangleq [\boldsymbol{r}^T, \boldsymbol{s}^T, \tilde{\boldsymbol{W}}_1^{vT}, \tilde{\boldsymbol{W}}_2^{vT}]^T$. Then from (2.19),

$$\begin{aligned} \dot{V} & \le -\sigma_l \|\boldsymbol{\eta}\|^2 + b_\epsilon \|\boldsymbol{\eta}\| \\ & = \|\boldsymbol{\eta}\| (b_\epsilon - \sigma_l \|\boldsymbol{\eta}\|) \end{aligned}$$

Therefore

$$\dot{V} \le -\delta \|\boldsymbol{\eta}\| < 0, \forall \|\boldsymbol{\eta}\| \ge (b_\epsilon + \delta)/\sigma_l > 0 \tag{2.20}$$

where $\delta > 0$ can be any small number.

In addition, according to [16], $\boldsymbol{M}(\boldsymbol{q})$ is positive definite and bounded, thus the Lyapunov function candidate V can be bounded by quadratic functions:

$$0 < \sigma_1 \|\boldsymbol{\eta}\|^2 \le V \le \sigma_2 \|\boldsymbol{\eta}\|^2 \tag{2.21}$$

where σ_1 is a positive number smaller than one half of the minimum singular values of $\boldsymbol{M}(\boldsymbol{q})$, $\boldsymbol{A}$, $\boldsymbol{F}_1^{-1}$, $\boldsymbol{F}_2^{-1}$, and σ_2 is a positive number larger than one half of the maximum singular values of the four gain matrices.

Referring to [16] and [22], uniform ultimate boundedness of $\boldsymbol{\eta}$ can be guaranteed by (2.20) and (2.21). Thus there exists t_0, such that $\forall t \ge t_0$, $\|\boldsymbol{\eta}\| \le \sqrt{\frac{\sigma_2}{\sigma_1}} \frac{b_\epsilon + \delta}{\sigma_l}$. Since δ can be any small number, this inequality is reduced to $\|\boldsymbol{\eta}\| \le \sqrt{\frac{\sigma_2}{\sigma_1}} \frac{b_\epsilon}{\sigma_l}$ eventually. This upper bound of $\|\boldsymbol{\eta}\|$ can be made arbitrarily small by increasing $\boldsymbol{K}_r$, $\boldsymbol{K}_s$, κ_1, κ_2, and decreasing γ_1, γ_2. Thus the trajectory tracking error $\boldsymbol{e}$ and neural network weight estimation error $\tilde{\boldsymbol{W}}_1, \tilde{\boldsymbol{W}}_2$ are uniformly ultimately bounded as $(i = 1, 2)$

$$\|\boldsymbol{e}\| \leq \frac{\|\boldsymbol{r}\|}{\sigma_{\min}(\boldsymbol{K}_p)} \leq \frac{\|\boldsymbol{\eta}\|}{\sigma_{\min}(\boldsymbol{K}_p)} \leq \frac{1}{\sigma_{\min}(\boldsymbol{K}_p)}\sqrt{\frac{\sigma_2}{\sigma_1}\frac{b_\epsilon}{\sigma_l}} \tag{2.22a}$$

$$\|\tilde{\boldsymbol{W}}_i\|_F \leq \|\boldsymbol{\eta}\| \leq \sqrt{\frac{\sigma_2}{\sigma_1}\frac{b_\epsilon}{\sigma_l}} \tag{2.22b}$$

2.5 Simulation Results

Joint flexibility is non-negligible when industrial robot is performing high speed motion with heavy payload. Such kind of motion is dangerous in laboratory environment without reliable perimeter safeguarding. Moreover, the proposed controller requires direct sensing of the robot link motion, which is not currently available in the Mechanical Systems Control laboratory at the University of California, Berkeley. Thus the proposed approach is verified by simulation study. The proposed controller is implemented to control a 6-axis robot in a high fidelity simulation. In the simulation, rigid body dynamics of the robot, joint flexibility, motor dynamics, complex friction are all taken into account. The reference trajectory in the simulation is designed to have high velocity and

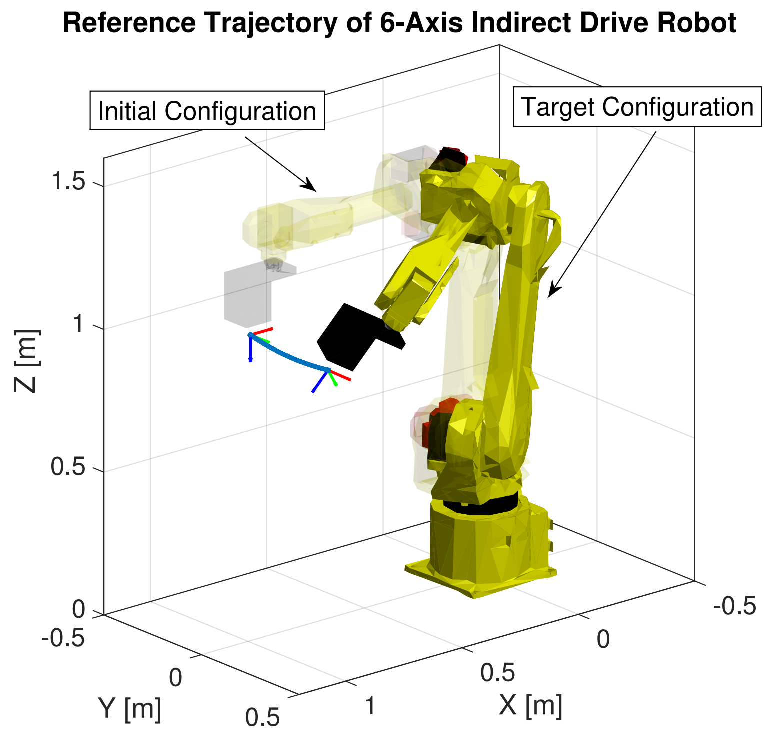


Figure 2.3: Reference trajectory of 6-axis indirect drive robot

acceleration. The reference trajectory is illustrated in Fig. 2.3, and the acceleration is shown in Fig. 2.4.

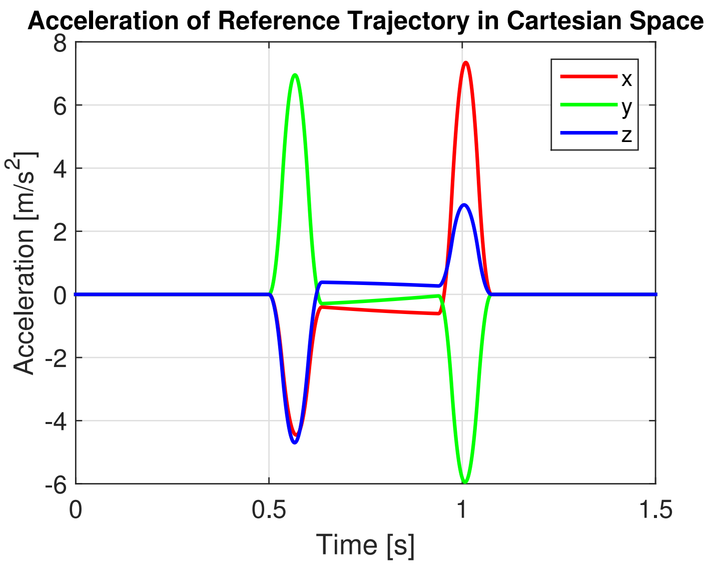


Figure 2.4: Acceleration of the reference trajectory in Cartesian space

A benchmark controller for industrial robot and the proposed controller are implemented in the simulation. Nominal dynamical model are used in both controllers. The modelling error includes: a) link inertia and center-of-gravity; b) friction; c) stiffness and damping of transmission units.

The benchmark controller consists of two parts: torque feedforward control part and feedback control part. The feedforward part utilizes a nominal rigid body dynamics model of the robot to compensate the nonlinear dynamics of the 6-axis robot. The feedback control part utilizes a well tuned proportional−integral−derivative (PID) controller. The benchmark controller has the form

$$\begin{aligned}\boldsymbol{\tau} = \boldsymbol{R}^{-1}\left[(\boldsymbol{M}_n(\boldsymbol{q}_d) + \boldsymbol{J}_{mn}\boldsymbol{R}^2)\ddot{\boldsymbol{q}}_d + \boldsymbol{C}_n(\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d)\dot{\boldsymbol{q}}_d + \boldsymbol{G}_n(\boldsymbol{q}_d)\right] \\ + \boldsymbol{K}_P(\boldsymbol{R}\boldsymbol{q}_d - \boldsymbol{\theta}) + \boldsymbol{K}_D(\boldsymbol{R}\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{\theta}}) + \boldsymbol{K}_I\int_0^t (\boldsymbol{R}\boldsymbol{q}_d - \boldsymbol{\theta})\end{aligned}$$

where the subscript $_n$ denotes the nominal model. $\boldsymbol{K}_P$, $\boldsymbol{K}_D$, and $\boldsymbol{K}_I$ are the PID gains.

The proposed neuroadaptive controller is designed as (2.7a), (2.7b). Two iterations are required for this approach. The first iteration is mainly used to collect data for designing center and width of each Gaussian basis function in the RBF network. In the first iteration, only the nominal model is used and there is no auxiliary model available, i.e., in the first iteration,

$$\begin{aligned}\boldsymbol{y}_d &= \boldsymbol{f}_n + \boldsymbol{K}_r\boldsymbol{r} \\ \boldsymbol{\tau} &= \boldsymbol{h}_n + \boldsymbol{r} + \boldsymbol{K}_s\boldsymbol{s}\end{aligned}$$

For the second iteration of the proposed controller, RBF network is used to build the auxiliary model as (2.8). In the second iteration, both nominal model and auxiliary model are used in the

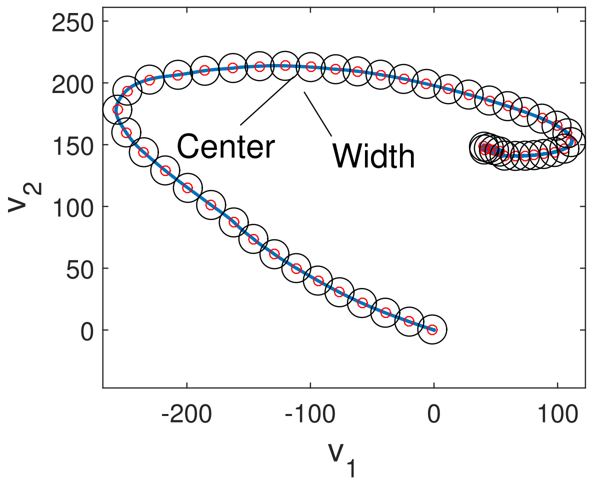


Figure 2.5: Low dimensional manifold from experiment data with designed center and width of radial basis functions

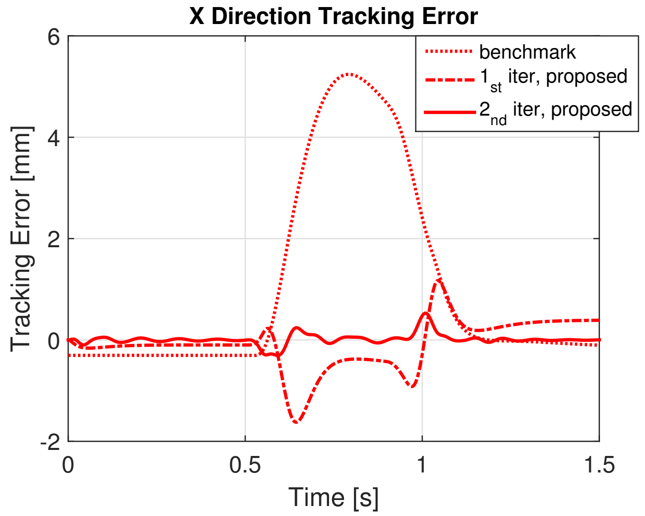


Figure 2.6: Cartesian space tracking error in X direction

controller,

$$\begin{aligned} \boldsymbol{y}_d &= \boldsymbol{f}_n + \kappa_1 \boldsymbol{W}_1^T \boldsymbol{\Phi}(\boldsymbol{X}) + \boldsymbol{K}_r \boldsymbol{r} \\ \boldsymbol{\tau} &= \boldsymbol{h}_n + \kappa_2 \boldsymbol{W}_2^T \boldsymbol{\Phi}(\boldsymbol{X}) + \boldsymbol{r} + \boldsymbol{K}_s \boldsymbol{s} \end{aligned}$$

The data from the first iteration is used in the initial training stage before running the second iteration, i.e. determining the center and width parameters for the RBF network. 50 neurons are used in this neural network. The 2-D projection of center and width of the radial basis functions

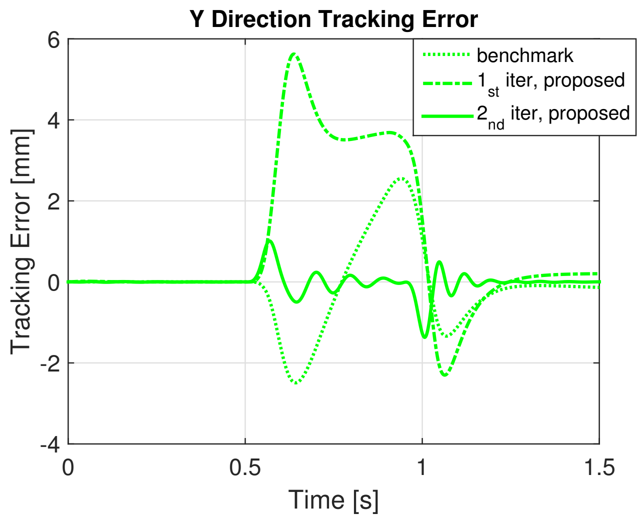


Figure 2.7: Cartesian space tracking error in Y direction

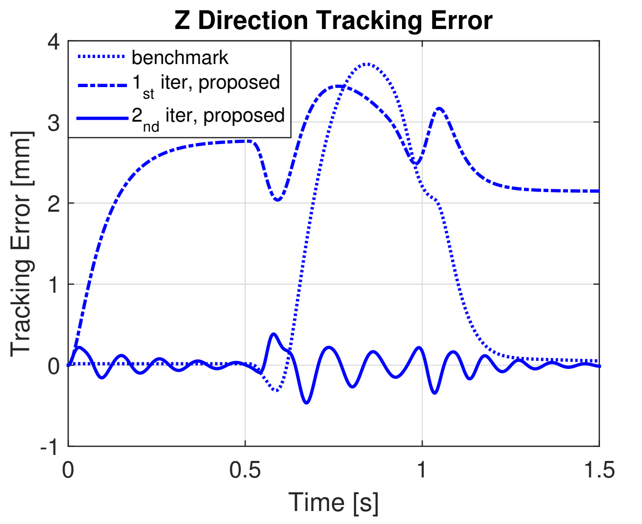


Figure 2.8: Cartesian space tracking error in Z direction

are shown in Fig. 2.5. The online training stage takes place during the second iteration. The neural network weights are trained adaptively as designed in (2.10a) and (2.10b).

The trajectory tracking for the benchmark controller and the proposed controller are shown in Fig. 2.6, 2.7, and 2.8. Due to modelling error, the trajectory tracking error is large for the benchmark controller and the first iteration of the proposed controller. But the error is effectively reduced in the second iteration.

The future work of this research is experimental validation. Before experimental study can be performed, installation of safeguards for high speed experiments should be finished. The development of direct sensing or accurate estimation of robot link motion is also necessary for the experimental study.

2.6 Chapter Summary

In this chapter, a neural network based adaptive backstepping control approach is proposed to improve trajectory tracking accuracy of indirect drive robots. An artificial neural network is used to approximate the difference of actual system and the physical model used for control. A two stage training approach, which consists of an offline data-driven initial training stage and an online training stage, was proposed to train the radial basis function network used in the controller. In the initial training stage, a model based backstepping controller is first implemented for data collection. The center and width parameters of neurons are then designed based on the motion data. In the second stage, the same trajectory is used and the weights of neural network are tuned online to improve the controller performance. Compared to other learning control techniques such as iterative learning control, the approach proposed in this chapter requires only at most two iterations for a specific trajectory, which is more efficient. It is proved that the trajectory tracking error and the neural network weight estimation error are uniform ultimate bounded. The effectiveness of the proposed controller is demonstrated using simulation on a six axis indirect drive robot. In order to perform experimental validation, the future work of this research shall focus on the sensing of robot link motions and safeguards installation for high speed robotic experiments.

Chapter 3

Zero Time Delay Input Shaping for Smooth Settling of Industrial Robots

3.1 Introduction

Industrial robots are widely used in manufacturing. In order to guarantee high product quality, as well as high productivity, precision position control and rapid rest-to-rest motion are desired in a variety of applications. However, serious overshoots and residual vibrations are widely and frequently observed when industrial robots are conducting fast motions [23]. Flexibility introduced by transmission units is the major cause of these unwanted motions [5, 12, 4]. In order to improve trajectory tracking performance, the overshoot and the residual vibration must be minimized, and the flexibility of industrial robots must be taken into account in the controller design .

To address these problems, several approaches are proposed, including singular perturbation [24, 25, 26], optimal trajectory planning [27, 28, 29], input shaping [30, 31, 32, 33], nonlinear feedback control [34, 35, 6, 36, 37], and iterative learning control [38, 9]. With a sophisticated system model, optimal trajectory planning can be implemented to generate an optimal motion reference to minimize the overshoot and the residual vibration. If such kind of model is not available, but the states related to the elastic vibration can be measured or observed, singular perturbation and nonlinear feedback control may be implemented to accommodate unmodeled dynamics and disturbances. Iterative learning control is another choice to address these problems by learning an optimal feedforward control when robot performs the same task repeatedly.

Comparing to other approaches, input shaping, which is also known as command shaping, may be implemented to effectively minimize the overshoot and the residual vibration a) without a sophisticated dynamical model; b) without directly measuring elastic vibrations for online feedback; c) without requirement of repeated tasks. Input shaping was first proposed for smoothing or shaping the inputs of linear second order systems. Later, modifications and extensions were introduced to handle multiple modes and changing natural frequencies of the system [39]. The robustness of input shaping was also considered to accommodate parameter uncertainty and disturbances [40].

As one of the easiest and successfully applied feedforward control techniques, input shaping has been implemented in a variety of applications ranging from nano-positioning devices to large industrial cranes [41, 42, 43]. However, there are certain drawbacks of input shaping for industrial robot applications. One drawback is the time delay introduced by input shaping. For industrial robots performing rapid rest-to-rest motions, the time delay could slow down the entire task, which is not desired in industrial applications. Another drawback is that input shaping may change the original motion reference. In certain applications, like spot welding, an industrial robot is required to move along a pre-specified trajectory to avoid colliding with work-pieces. If the shaped motion command does not preserve the path of the original trajectory, there could be a collision between the robot and the work-pieces. In order to compensate for the time delay, this chapter proposes a zero time delay input shaping approach. The time delay introduced by input shaping can be fully compensated. Path constraint is also considered in the design of the proposed input shaping approach. Experimental results on a 6-axis industrial robot have verified the effectiveness of the proposed approach.

3.2 Conventional Input Shaping

A review of conventional input shaping techniques is provided in this section[40, 32, 33]. Input shaping is implemented by convolving a sequence of impulses with the original system input. Each impulse can excite an oscillatory response. When the amplitudes and time delays are well tuned such that the oscillatory responses cancel each other, there will be no residual vibration. This convolutional approach is equivalent to separating the input into different parts with time delay, and then getting these parts together, as illustrated in Fig. 3.1 [33].

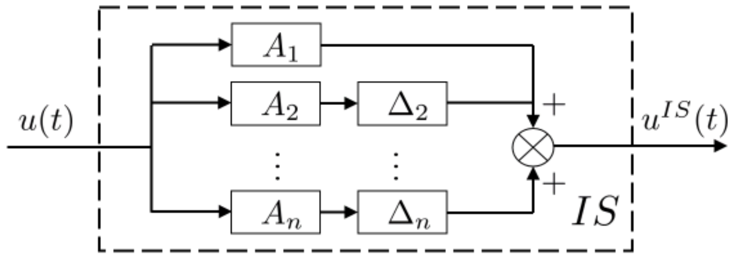


Figure 3.1: Time-delay blocks representing input shaping

The idea of input shaping was first introduced for linear second order systems. Consider a linear second order system with the transfer function G,

$$G(s) = \frac{K\omega_0^2}{s^2 + 2D\omega_0 s + \omega_0^2} \tag{3.1}$$

where ω_0 is the natural frequency, D is the damping ratio, and K is the static gain. The unit impulse response $y(t)$ of this linear second order system (3.1) is

$$y(t) = K\frac{\omega_0}{\sqrt{1-D^2}}e^{-\omega_0 Dt}\sin(\omega_d t) \tag{3.2}$$

where $\omega_d = \omega_0\sqrt{1-D^2}$ is the damped natural frequency.

Let f_{IS} be a sequence of n impulses

$$f_{IS}(t) = \sum_{i=1}^{n} A_i\delta(t-t_i) \tag{3.3}$$

where A_i is the amplitude of the i^{th} impulse, t_i is the time delay of the i^{th} impulse. Typically, it is assumed that

$$\begin{array}{rcl} t_{i+1} & > & t_i \\ A_i & > & 0 \end{array} \tag{3.4}$$

Convolving this sequence with the original unit impulse, the resulting response $Y_{IS}(t)$ for $t \geq t_n$ is

$$\begin{array}{rcl} Y_{IS}(t) & = & \sum_{i=1}^{n} A_i y(t-t_i) \\ & = & K\frac{\omega_0}{\sqrt{1-D^2}}e^{-\omega_0 Dt}\left[A(\omega_0,D)\sin(\omega_d t)\right. \\ & & \left.-B(\omega_0,D)\cos(\omega_d t)\right] \\ & = & K\frac{\omega_0}{\sqrt{1-D^2}}e^{-\omega_0 Dt}I(\omega_0,D)\sin(\omega_d t+\phi) \end{array} \tag{3.5}$$

where

$$\begin{array}{rcl} I(\omega_0,D) & = & \sqrt{A(\omega_0,D)^2+B(\omega_0,D)^2} \\ A(\omega_0,D) & = & \sum_{i=1}^{n} A_i e^{\omega_0 Dt_i}\cos(\omega_d t_i) \\ B(\omega_0,D) & = & \sum_{i=1}^{n} A_i e^{\omega_0 Dt_i}\sin(\omega_d t_i) \\ \cos(\phi) & = & \frac{A(\omega_0,D)}{I(\omega_0,D)} \\ \sin(\phi) & = & -\frac{B(\omega_0,D)}{I(\omega_0,D)} \end{array} \tag{3.6}$$

The amplitude ratio between the shaped impulse response (3.5) and unshaped impulse response (3.2) after t_n is typically used as the performance index of input shaping[40]. This ratio is also known as percentage of residual vibration, which is defined as

$$\begin{array}{rcl} V(\omega_0,D) & := & e^{-\omega_0 Dt_n}\sqrt{A(\omega_0,D)^2+B(\omega_0,D)^2} \\ & = & e^{-\omega_0 Dt_n}I(\omega_0,D) \end{array} \tag{3.7}$$

The term $e^{-\omega_0 Dt_n}$ implies that a time delay t_n is introduced in the shaped response. This ratio reflects the effect of the residual vibration suppression. The design objective of input shaping is to make $V \approx 0$.

For a given system, V depends only on the amplitudes and time delays of the sequence of impulses f_{IS}. Therefore the design of input shaping is equivalent to the design of the sequence f_{IS}, which is also known as an input shaper.

One design of the input shaper is called zero vibration (ZV) shaper. There are only two impulses in a ZV shaper. The design of a ZV shaper involves solving a set of equations with constraints (3.4)

$$\begin{array}{rcl} A(\omega_0, D) & = & 0 \\ B(\omega_0, D) & = & 0 \\ \sum_{i=1}^{2} A_i & = & 1 \end{array} \tag{3.8}$$

where the first two equations are derived from that the percentage of residual vibration $V = 0$, and the third equation is derived from the requirement that the input shaper has an unity static gain for avoiding overshoot. The design of a ZV shaper can be chosen as the solution of (3.8) with the minimum t_2. The residual vibrations caused by the two impulses cancel out each other after the second impulse is applied, as illustrated in Fig (3.2) [33].

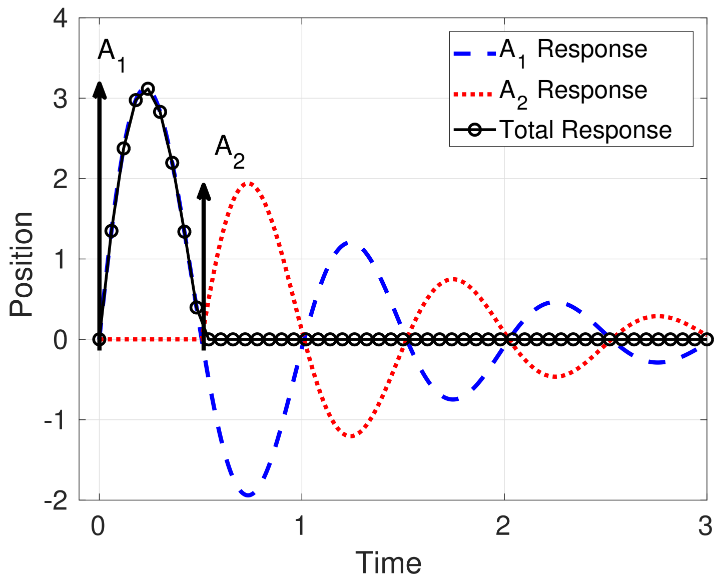


Figure 3.2: Vibration from two impulses cancel each other

Theoretically, the ZV shaper could completely eliminate the residual vibration since $V = 0$ for accurately known ω_0, D. However, the ZV shaper can be sensitive to modeling errors in practice. Thus robust design of input shaping was considered. Zero vibration and derivative (ZVD) shaper, extra-insensitivity (EI) shaper, and specified insensitivity (SI) shaper are commonly used robust input shapers [44]. Only the specified insensitivity shaper is reviewed here since it provides the most robust performance in these approaches.

The design of SI shaper can be stated as an optimization problem. The objective is to minimize the total time delay of the input shaping. On one hand, the constraints of this optimization problem come from (3.4). On the other hand, the constrains of this optimization problem come from the requirement of SI that the percentage of residual vibration is below a given level within a range of frequencies. It is difficult to derive the analytical form of the percentage of residual vibration constraint. Instead, an approximate approach called frequency sampling approach is typically implemented in the design of SI shaper.

In the frequency sampling approach, it is assumed that the natural frequency satisfies $\omega_0 \in [\omega_{inf}, \omega_{sup}]$, and the resulting percentage of residual vibration is required to be below a given level V_0. A set of frequencies are sampled from the frequency range as $\{\omega_0^1, \omega_0^2, \cdots, \omega_0^m\}$, where m is the number of samples, and ω_0^i is the i^{th} frequency sample. Suppose n impulses are used in the shaper, the design of SI shaper can be formulated as

$$\begin{array}{rl} \min\limits_{A_1,\cdots,A_n,t_1,\cdots,t_n} & t_n \\ s.t. & t_{i+1} > t_i, i = 1, \cdots, n \\ & A_i > 0, i = 1, \cdots, n \\ & \sum_{i=1}^n A_i = 1 \\ & V(\omega_0^j, D) \leq V_0, j = 1, \cdots, m \end{array} \tag{3.9}$$

When the sample set is large enough to cover the frequency range, the frequency range $[\omega_{inf}, \omega_{sup}]$ can be well approximated by the samples. In actual application, as long as the estimated natural frequency is in the frequency range, SI shaper guarantees good residual vibration suppression performance. The cost of such robust input shaper is longer time delay. Usually more than two impulses should be implemented, and the overall time delay is longer than ZV shaper.

3.3 Zero Time Delay Input Shaping

The proposed zero time delay input shaping is introduced in this section. A path constraint issue of implementing input shaping on industrial robot is firstly addressed. The zero time delay input shaping is then developed based on the path constraint design.

Input Shaping with Path Constraint

In many works, input shaping is applied on single-input single-output systems. In this chapter, input shaping is implemented on a 6-axis industrial robot, which is a multiple-input multiple-output system. One natural choice is to implement input shaping on each axis independently. However, as mentioned in the introduction, this may change the original task space motion reference, which could result in undesired behaviour of the robot.

For the motion command given in Cartesian space, another intuitive approach is to apply input shaping to the Cartesian space motion command of industrial robots. The corresponding joint space motion command can be obtained through the solution of inverse kinematics problem. However, input shaping is "smoothing" the motion command in each direction of the Cartesian space, thus the shaped motion path can still be different from the original motion path.

In this chapter, a third approach is developed. Let the Cartesian space motion command be $\{x(t), y(t), z(t)\}$, where the motion time $t \in [0, T]$. The motion command can be parametrized with the normalized arc length s, defined as

$$s(t) = \frac{\int_{\tau=0}^{t} \sqrt{\dot{x}(\tau)^2 + \dot{y}(\tau)^2 + \dot{z}(\tau)^2}\mathrm{d}\tau}{\int_{\tau=0}^{T} \sqrt{\dot{x}(\tau)^2 + \dot{y}(\tau)^2 + \dot{z}(\tau)^2}\mathrm{d}\tau} \tag{3.10}$$

where $s \in [0, 1]$. The motion command can be parametrized as $\{x(s), y(s), z(s)\}$. Input shaping is then implemented on the normalized arc length $s(t)$. The corresponding joint space motion command is then obtained through the solution of an inverse kinematics problem.

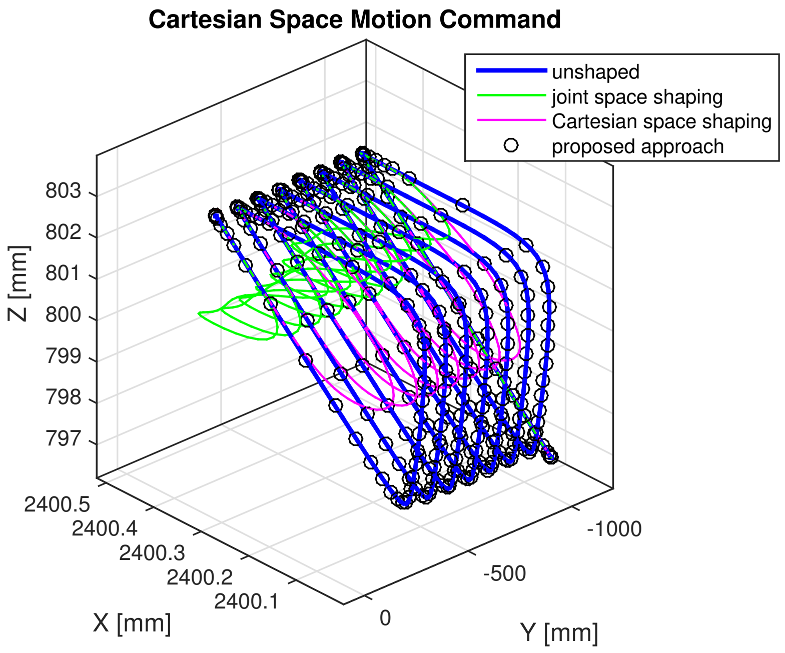


Figure 3.3: Comparison of input shaping on joint space motion command, Cartesian space motion command, and the proposed approach

A comparison of input shaping on joint space motion command, Cartesian space motion command, proposed approach, and the unshaped motion command is given in Fig.3.3. As shown in the figure, directly implementing input shaping to joint space motion command results in a large deviation. Implementing input shaping in Cartesian space makes the shaped motion command closer to the unshaped motion command, but the deviation still exists. The proposed approach preserves the path of the unshaped motion command.

Zero Time Delay Shaping

In order to preserve the path of unshaped motion command, the proposed approach in 3.3 is implemented. The input to the system can be chosen as the normalized arc length $s(t), t \in [0, T]$. Input shaping is then implemented on the normalized arc length. According to the existing literature, time delay will inevitably be introduced by traditional input shaping. If robustness is considered in the design, the time delay could be even longer. In order to eliminate the undesired time delay, and keep the robust design of input shaping at the same time, the following design procedure is proposed:

1. Design an input shaping using any approach introduced in the literature. The input shaper $f_{IS} = \sum_{i=1}^{n} A_i \delta(t - t_i)$ is obtained. The time delay introduced by the input shaping is t_n.

2. Accelerate the unshaped motion command $s(t)$ to $s_{acc}(\tau)$, where $t \in [0, T]$, $\tau \in [0, T - t_{acc}]$, and $t_n < t_{acc} < T$.

3. Apply the input shaping designed in the first step to the accelerated motion command $s(\tau)$. The resulting shaped input is $S_{IS} = f_{IS} * s_{acc}$.

For the second step, let a time scale parameter be $k = \frac{T - t_{acc}}{T} < 1$, the accelerated normalized arc length is

$$s_{acc}(\tau) = s_{acc}(kt) = s(t), t \in [0, T] \tag{3.11}$$

Suppose there are n impulses in the input shaper (3.3). The resulting shaped motion command is

$$S_{IS}(t') = \sum_{i=1}^{n} A_i s'_{acc}(t' - t_i) \cdot u(t' - t_i) \tag{3.12}$$

where the time variable $t' \in [0, T - t_{acc} + t_n]$; $s'_{acc}(t')$ is an extension of s_{acc} such that

$$s'_{acc}(t') = \begin{cases} s_{acc}(t'), & t' \in [0, T - t_{acc}] \\ s_{acc}(T - t_{acc}), & t' \geq T - t_{acc} \end{cases}$$

and $u(t')$ is the Heaviside step function that

$$u(t') = \begin{cases} 0, & t' < 0 \\ 1, & t' \geq 0 \end{cases}$$

Comparing to the unshaped motion command $s(t), t \in [0, T]$, the shaped motion command ends at $T - t_{acc} + t_n$. Since $t_{acc} > t_n$, the end time of the shaped motion command satisfies $T - t_{acc} + t_n < T$.

The proposed input shaping approach is sketched in Fig.3.4. As shown in the figure, the unshaped motion command is first accelerated. After input shaping is applied, time delay is introduced to the accelerated motion command, but there is no time delay between the unshaped motion command and the shaped motion command.

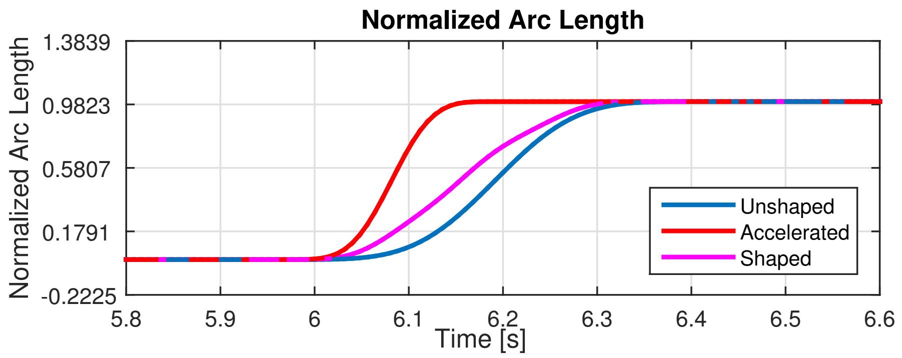


Figure 3.4: Shape of arc length

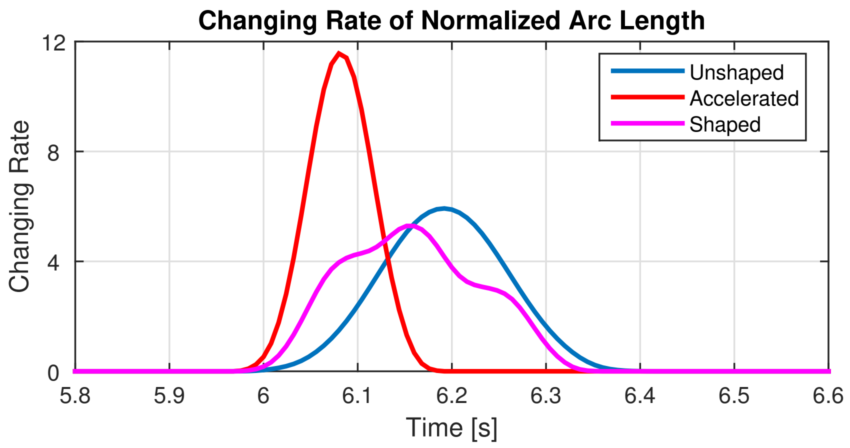


Figure 3.5: Shape of arc velocity

The velocity or changing rate of the motion commands are compared in Fig.3.5. As shown in the figure, it is clear that the shaped motion command ends earlier than the unshaped motion command, which means that there is no time delay when applying this approach.

3.4 Implementation and Experimental Results

The proposed approach is implemented on a 6-axis industrial robot shown in Fig.3.6. The robot is performing a rapid rest-to-rest motion (e.g., a typical spot welding motion). The motion of the endeffector of the robot is measured by a laser tracker. An optical reflector of the laser tracker is attached to the desired tool center point of the robot. The Cartesian space motion command and the position measurement from the laser tracker is shown in Fig.3.7. As shown in the figure, there exists obvious overshoot and residual vibration.

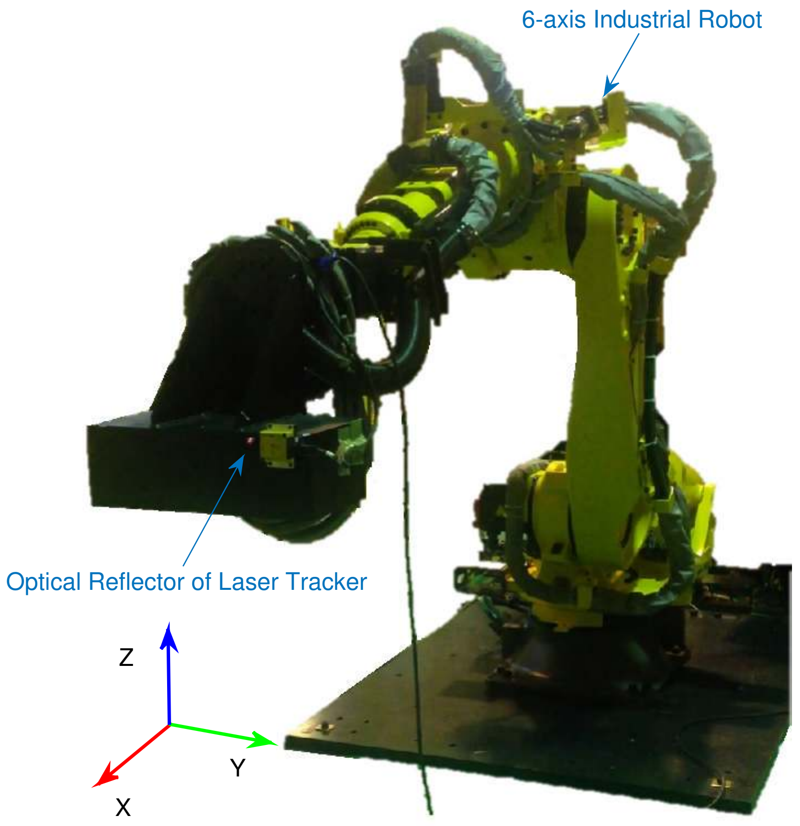


Figure 3.6: 6-axis industrial robot

Frequency and Damping Ratio Estimation

The natural frequency and damping ratio are required for the design of input shaping. The motion data measured by the laser tracker is used to estimate these parameters. The residual vibration of the robot is considered to be caused by the flexibility at each joint of the robot. It is assumed that the residual vibration of each joint can be fitted into a free vibration of a mass-spring-damper system, which is a linear second order system. No angular velocity in Cartesian space is measured in the experiment, thus the joint space velocity is assumed to be calculated using

$$\dot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q})^{-1} \begin{bmatrix} \dot{\boldsymbol{p}} \\ \boldsymbol{0} \end{bmatrix} \tag{3.13}$$

where $\boldsymbol{q} = [q_1, \cdots, q_6]^T$ is the set of joint positions of the robot; $\boldsymbol{J}(\boldsymbol{q})$ is the Jacobian matrix of the robot; $\boldsymbol{p} = [x, y, z]^T$ is measured Cartesian space position of the robot, in which x, y, z are the

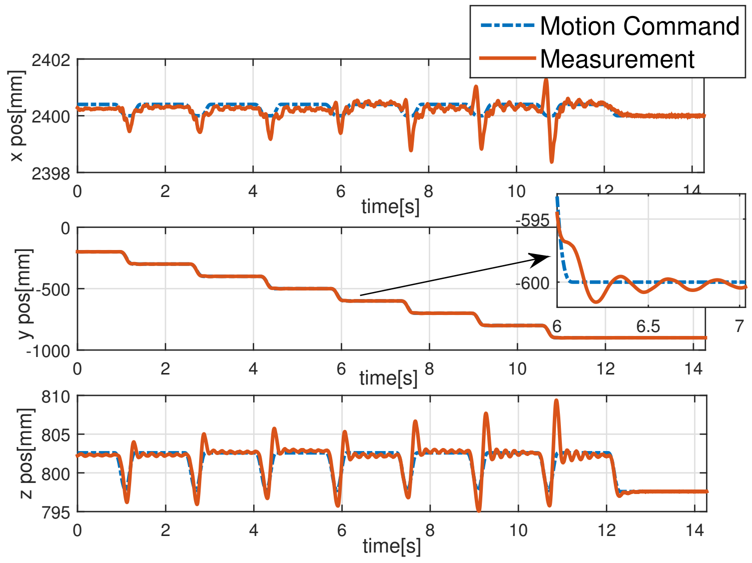


Figure 3.7: Cartesian space motion command and position measurements

Cartesian space position in x, y, z direction. The calculated joint space velocities during the one of the residual vibrations in the entire motion are shown in Fig.3.8.

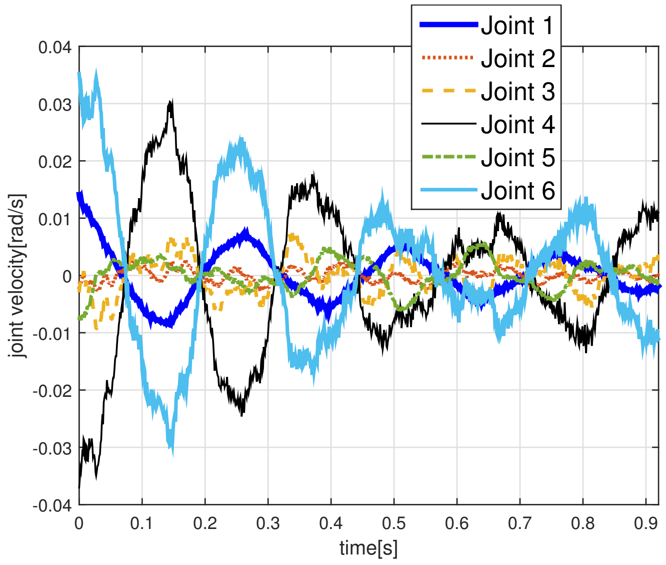


Figure 3.8: Estimated joint velocity during one of the residual vibrations

Let the time domain response of the free vibration of the linear second order system (3.1) be $\eta(t)$:

$$\eta(t) = e^{-D\omega_0 t}\left(C_1 \cos(\omega_d t) + C_2 \sin(\omega_d t)\right) \tag{3.14}$$

where

$$\begin{array}{rcl} C_1 & = & \eta(0) \\ C_2 & = & \frac{\dot{\eta}(0)+D\omega_0\eta(0)}{\omega_d} \end{array} \tag{3.15}$$

The velocity of the free vibration of a linear second order system (3.1) is

$$\begin{array}{rcl} \dot{\eta}(t) & = & -D\omega_0 e^{-D\omega_0 t}\left(C_1\cos(\omega_d t)+C_2\sin(\omega_d t)\right) \\ & & \omega_d e^{-D\omega_0 t}\left(C_2\cos(\omega_d t)-C_1\sin(\omega_d t)\right) \end{array} \tag{3.16}$$

The frequency and damping parameters of each joint is first roughly estimated using fast Fourier transformation (FFT). These parameters are further tuned to fit (3.16) using least squares method. The fitting result is illustrated by one of the results shown in Fig.3.9.

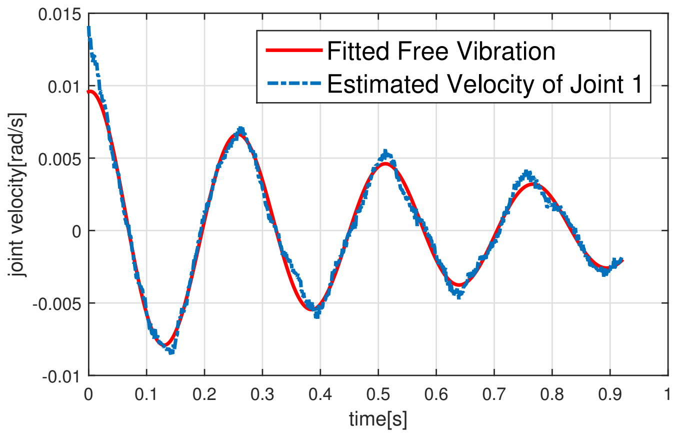


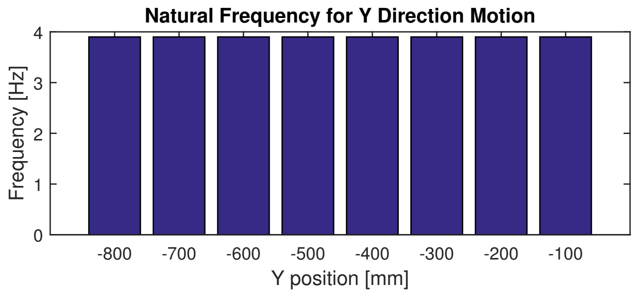
Figure 3.9: Estimated joint velocity and fitted free vibration

The estimated frequency and damping ratio parameters of each joint are shown in Table.3.1. (where Jnt is short for Joint)

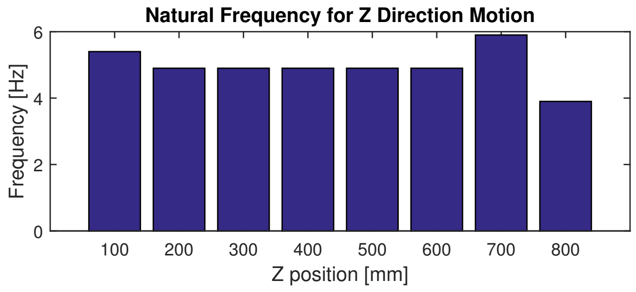
Table 3.1: Estimated frequency and damping ratio

Parameter	Jnt 1	Jnt 2	Jnt 3	Jnt 4	Jnt 5	Jnt 6
Frequency [Hz]	3.7	4.7	5.9	4.3	4.4	4.4
Damping Ratio	0.11	0.12	0.07	0.14	0.23	0.14

The natural frequencies are around 4 Hz. The robot is performing rapid rest-to-rest motion within its workspace. The natural frequencies at different positions may be different. The same estimation is repeated at different positions, and the distribution of the mean natural frequencies of all the joints is shown in Fig.3.10. As shown in the figure, the frequencies are very close.



(a) Y direction



(b) Z direction

Figure 3.10: Distribution of natural frequencies at different positions

Robust Design

In practical application, the design of input shaping must consider robustness since there will be estimation error of the natural frequency and damping ratio parameters of any practical system. Furthermore, only one pair of frequency and damping ratio parameters can be used when the proposed approach is implemented on an industrial robot. According to the parameter estimation results in 3.4, the frequency and damping ratio parameters are different a) for different joint; b) at different positions. In order to avoid poor performance due to a bad choice of parameters, robust design should be considered.

The specified insensitivity design of input shaping is implemented in the experiment. Three impulses are included in the input shaper. The natural frequency is chosen to be 4.5 Hz in the final design, and the damping ratio is chosen to be 0.05. The constraint of the SI shaper design is that the residual vibration level should not exceed 15% of the unshaped motion for the actual natural frequency $3.6\text{Hz} < \omega_0 < 5.4\text{Hz}$, which is a $\pm 20\%$ range. The parameters of the input shaper are: $A_1 = 0.3369$, $A_2 = 0.4069$, $A_3 = 0.2542$, $t_1 = 0$, $t_2 = 0.0891$, and $t_3 = 0.1766$. The SI shaper design is compared with the standard ZV shaper design in Fig.3.11. The x and y axis indicates the distribution of frequency and damping ratio parameters, and the z axis indicates the level of residual vibration. As shown in the figure, the vibration suppression performance is not sensitive to damping ratio. Comparing to ZV shaper, SI shaper is less sensitive to the change of frequency.

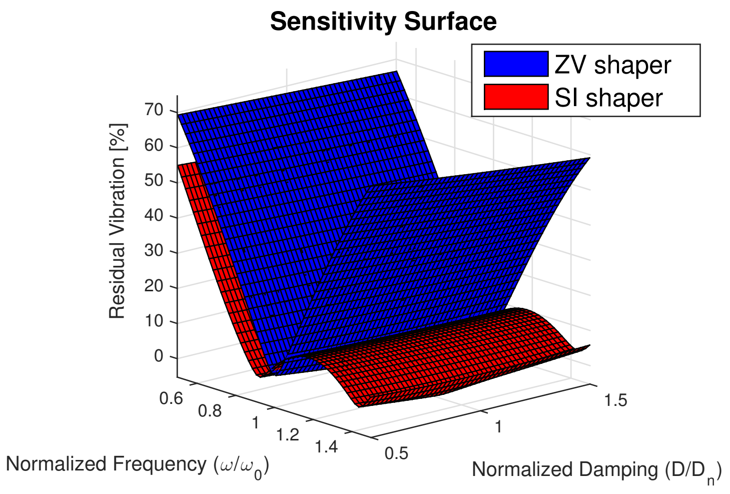


Figure 3.11: Sensitivity surface

Experimental Result

The proposed input shaping is implemented on the 6-axis industrial robot. The original motion command is accelerated by $t_{acc} = 0.2$. The input shaping is performed on normalized arc length as shown in Fig.3.4 in section 3.3. The Cartesian space motion command, unshaped motion command, and shaped motion command are compared in Fig.3.12. Comparing to unshaped motion,

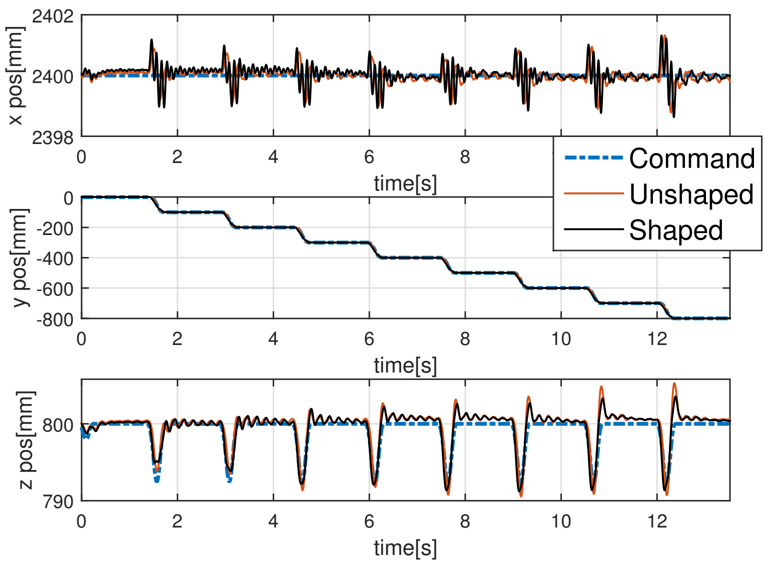


Figure 3.12: Comparison of motion command, unshaped motion, and shaped motion

input shaping has effectively reduced the overshoot in the z direction. This result can be observed more clearly in Fig.3.13. As shown in the figure, the overshoot has been reduced by 1.7 mm, which is about one third of the unshaped motion.

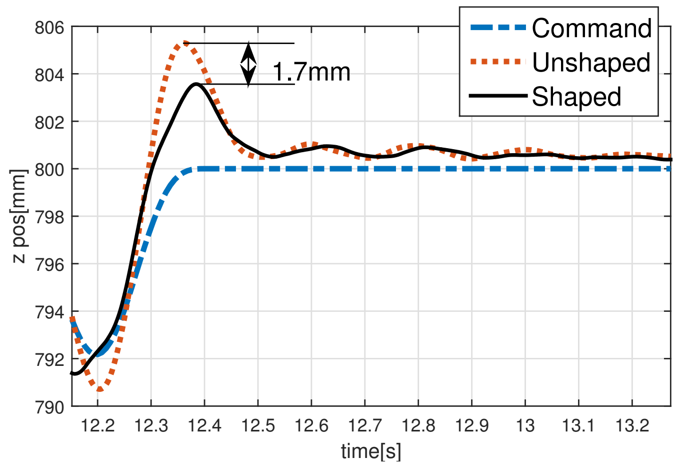
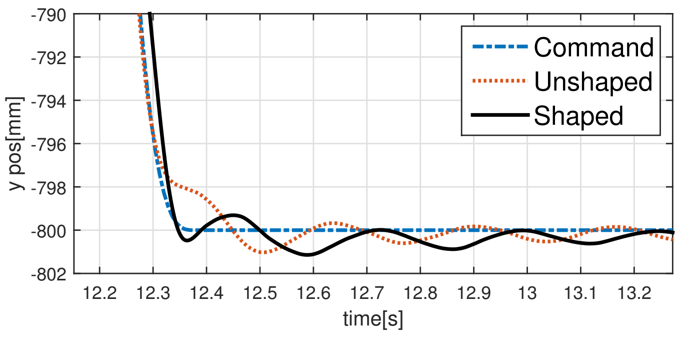
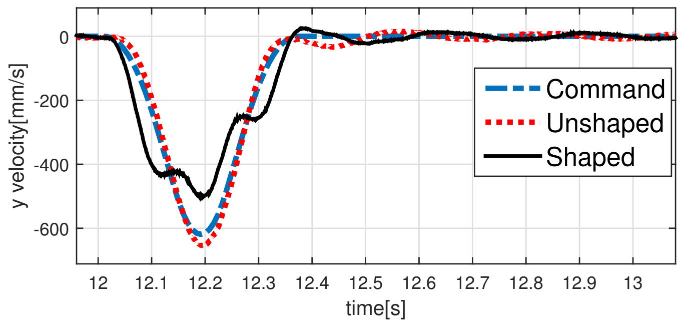


Figure 3.13: Effect of the the proposed approach in Z direction

From a practical point of view, the robot is considered to be settled when the residual vibration is less than 1mm and the robot can start its work (e.g., spot welding). Since the amplitude of the residual vibration observed is very small, this level of vibration is tolerable for industrial robot applications. In this case the reduction of the overshoot becomes important as it can help avoid the collision due to path deviation. The effect of input shaping in the y direction is shown more clearly in Fig.3.14. While not included in the figure, the conventional SI shaper would have introduced a delayed response with a delay time of about 0.2 sec. The proposed method does not introduce such a delay.



(a) Position



(b) Velocity

Figure 3.14: Effect of the proposed approach in Y direction

3.5 Chapter Summary

In this chapter, a zero time delay input shaping approach was proposed for smooth settling of an industrial robot. The proposed approach could fully compensate for the time delay introduced by the conventional input shaping techniques. Another feature of the proposed approach was the ability to preserve the path of the unshaped motion command, which made it practical when applied to multi axis industrial robots. The proposed approach was implemented on a 6-axis industrial robot. The experimental results showed that the proposed approach could effectively improve the performance of the robot by reducing the overshoot and no time delay was introduced.

Chapter 4

Modified Zero Time Delay Input Shaping for Industrial Robot with Flexible End-Effector

4.1 Introduction

Fast and precise motion control is required in a variety of industrial applications, such as spot welding using industrial robot shown in Fig.4.1. Due to the flexibility in the drive train and end-effector, nonnegligible residual vibration will appear when the robot is performing a motion at high speed and/or high acceleration. Residual vibration is undesired because a) large vibration may cause collision between the robot and the workpiece, b) a robot can not proceed to the next task until the residual vibration settles, which reduces the production rate.

Figure 4.1: Industrial robots for spot welding

In order to suppress the residual vibrations without modifying the mechanical structure, advanced control techniques should be investigated. Input shaping (IS) [33, 31] is one of the most promising techniques for vibration suppression. Though input shaping is effective, easy to use, and robust to modelling errors [40], the time delay introduced is not desired in applications with stringent requirements on operation times.

In order to overcome the time delay problem, techniques including predictive approach [45], smith predictor [46], equal length shaper [47] and zero time delay input shaping [48] have been proposed. Some of these techniques such as predictive approach or smith predictor can only partially eliminate the time delay. Equal length shaper and zero time delay input shaping, which utilize similar ideas, can achieve zero time delay by accelerating the original input. However, the shaped motion may be non-smooth.The non-smooth motion is undesired since it causes more wear and decrease the robot service life [49, 50].

In this chapter, a modified zero time delay input shaping approach is proposed to address the non-smoothness issue. The proposed approach is developed based on the zero time delay input shaping approach, and thus no time delay is introduced. Comparing to the equal length shaper or zero time delay input shaping, the proposed modification generates more smooth motion.

4.2 Review of Input Shaping

This section reviews input shaping and zero time delay input shaping. In order to reveal the connection between these two approaches, a convolution representation of input shaping is introduced.

Input Shaping

The idea of input shaping is: separating the input into different parts with time delay, such that the residual vibration caused by each parts would cancel each other. The structure of input shaping is represented in Fig. 4.2, where the 'IS' block represents input shaping. This block divides the input

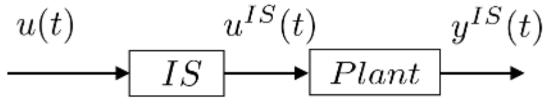


Figure 4.2: Input shaping

$u(t)$ into several parts with different time delay as illustrated in Chapter 3.

Taking second order linear system as an example. Let the transfer function be

$$G(s) = \frac{K\omega_0^2}{s^2 + 2D\omega_0 s + \omega_0^2} \tag{4.1}$$

where ω_0 is the natural frequency, D is the damping ratio, and K is the static gain. Let the input be an unit impulse signal, as $u_\delta(t) = \delta(t)$. Then the output of this system, i.e. the unit impulse response is:

$$h(t) = K\frac{\omega_0}{\sqrt{1-D^2}}e^{-D\omega_0 t}\sin(\sqrt{1-D^2}\ \omega_0 t) \tag{4.2}$$

Refer to [31], let $k = \exp\left(\frac{-D\pi}{\sqrt{1-D^2}}\right)$, $A_1 = \frac{1}{1+k}$, $A_2 = \frac{k}{1+k}$, $\Delta_2 = \frac{\pi}{\omega_0\sqrt{1-D^2}}$, then the shaped input $u^{IS}(t) = A_1\delta(t) + A_2\delta(t-\Delta_2)$ can remove residual vibration after Δ_2 since the responses of $A_1\delta(t)$ and $A_2\delta(t-\Delta_2)$ are cancelling out each other after Δ_2.

From the aspect of convolution, $u_\delta^{IS}(t) = (u_\delta * IS)(t)$, where $*$ is the convolution operator and $IS(t)$ is a sequence of impulses, known as input shaper:

$$IS(t) = \sum_{i=1}^{n} A_i\delta(t-\Delta_i) \tag{4.3}$$

It turns out that for any input with finite length, such kind of input shaper is able to effectively suppress residual vibration after delaying the original input by Δ_n, which is the time delay of the last impulse. If the length of input $u(t)$ is T, then the length after input shaping is $T + \Delta_n$. When more impulses are added, the input shaping is more robust to modelling error, but the time delay is longer [40].

Zero Time Delay Input Shaping

The structure of zero time delay input shaping is summarized as shown in Fig. 4.3. Comparing to the conventional input shaping, an 'accelerate' block is used to shorten the length of the original control input. This block makes it possible to totally eliminate the time delay.

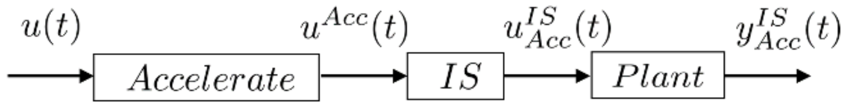


Figure 4.3: Zero time delay input shaping

Let T be the length of the input $u(t)$, and Δ_n be the time delay introduced by input shaping. Let a time scale parameter be $\alpha = \frac{T-\Delta_n}{T} < 1$, then the accelerated input is

$$u^{Acc}(t) = u\left(\frac{t}{\alpha}\right), t \in [0, \alpha T] \tag{4.4}$$

The length of the accelerated input is αT. Input shaping is applied to the accelerated input $u^{Acc}(t)$, and time delay Δ_n is added to the accelerated input. The length of the shaped input $u^{IS}_{Acc}(t)$ is T since

$$\alpha T + \Delta_n = \frac{T-\Delta_n}{T}T + \Delta_n = T \tag{4.5}$$

As a result, there is no time delay compared to the original input $u(t)$.

Convolution Representation of Input Shaping

Let $h(t)$ be the impulse response of a linear system. Then the output of input $u(t)$ is

$$y(t) = \int_0^t h(\tau)u(t-\tau)\mathrm{d}\tau = (h * u)(t) \tag{4.6}$$

For conventional input shaping, the output $y^{IS}(t)$ is

$$y^{IS}(t) = u(t) * IS(t) * h(t) = u(t) * (IS * h)\,(t) := u(t) * h^{IS}(t) \tag{4.7}$$

According to Eqn. (4.7), input shaping can be interpreted as modifying the impulse response of dynamical system. This modification adjust impulse response from $h(t)$ to $h^{IS}(t) = (IS * h)(t)$. The residual vibration can be suppressed for any input as long as the impulse response is $h^{IS}(t)$.

For zero time delay input shaping, the input has been accelerated to $u^{Acc}(t)$, and the output $y^{IS}_{Acc}(t)$ is

$$y^{IS}_{Acc}(t) = u^{Acc}(t) * IS(t) * h(t) = u^{Acc}(t) * h^{IS}(t) \tag{4.8}$$

which verifies the point that the residual vibration can be suppressed if $h^{IS}(t)$ is used.

4.3 Modified Zero Time Delay Input Shaping

Comparing to the conventional input shaping, zero time delay input shaping totally eliminates the time delay. However, this approach shows drawbacks in some applications because of the accelerated nature of the input signal. Thus a modification of zero time delay input shaping is developed in this section.

Drawback of Zero Time Delay Input Shaping

A feature of zero time delay input shaping is to accelerate the original input $u(t)$. When the length of the time delay and the control input are close, the accelerating action results in non-smooth motion as shown in Fig. 4.4.

In Fig. 4.4, the length of $u(t)$ is close to Δ_n, thus the time scale α is close to 0 and a severe accelerating action is performing on the input $u(t)$. After input shaping, the accelerated input decomposes, resulting in several peaks in the shaped input. The shaped input is less smooth comparing to the original input. If the non-smooth input is used, the changing rate of input could exceed the actuator's limit and cause more wear in the actuator's mechanical parts.

Modified Zero Time Delay Input Shaping

In order to overcome the non-smoothness issue, a modified zero time delay input shaping is proposed. The structure of the proposed approach is summarized in Fig. 4.5.

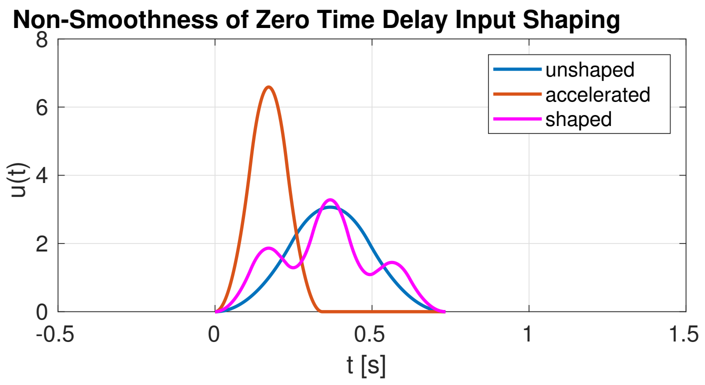


Figure 4.4: Drawback of zero time delay input shaping: close time length and time delay result in non-smoothness

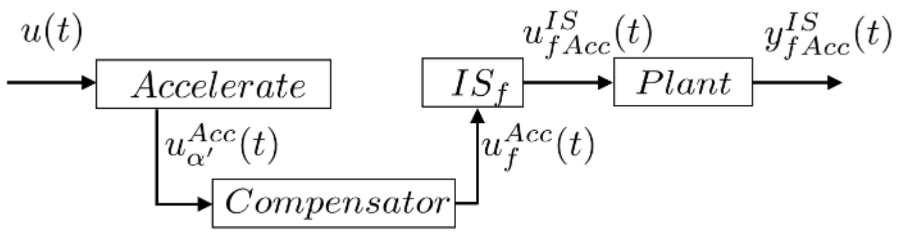


Figure 4.5: Modified zero time delay input shaping

Comparing to zero time delay input shaping, the modification here is a compensator block. The function of this block is to reduce time delay required for input shaping, thus no severe accelerating action is required for the input $u(t)$.

This section presents the design of this compensator and corresponding input shaper IS_f from the viewpoint of convolution product.

Design of Modified Zero Time Delay Input Shaping In zero time delay input shaping, $u(t)$ is accelerated by time scale α, but $h^{IS}(t) = (IS * h)(t)$ is not scaled at all. In this chapter, we consider the idea that accelerating $u(t)$ and $h^{IS}(t)$ by the same time scale α', such that

$$u_{\alpha'}^{Acc}(t) = u\left(\frac{t}{\alpha'}\right) \tag{4.9}$$

and $h_{\alpha'}^{IS}(t) = h^{IS}\left(\frac{t}{\alpha'}\right)$. According to the time scaling property of convolution [51], letting $y_{\alpha'}^{IS}(t) = \frac{1}{\alpha'}h_{\alpha'}^{IS}(t) * u_{\alpha'}^{Acc}(t), \forall t \in [0, \alpha' T]$

$$y_{\alpha'}^{IS}(t) = \frac{1}{\alpha'}\int_0^t u\left(\frac{\tau}{\alpha'}\right) h^{IS}\left(\frac{t}{\alpha'} - \frac{\tau}{\alpha'}\right) \mathrm{d}\tau = y^{IS}\left(\frac{t}{\alpha'}\right) \tag{4.10}$$

Let T be the length of input, and Δ_n be the time delay of input shaping. In order to guarantee zero time delay, $\alpha'(T+\Delta_n)=T$ should be satisfied. Thus

$$\alpha' = \frac{T}{T+\Delta_n} > \frac{T-\Delta_n}{T} = \alpha \tag{4.11}$$

where α is the time scale of zero time delay input shaping from Eqn. (4.5). Since α' is larger, the accelerating action is less severe than that of the zero time delay input shaping. The function of the compensator and corresponding input shaper is to adjust $h^{IS}(t)$ to $\frac{1}{\alpha'}h^{IS}_{\alpha'}(t)$.

Compensator The design of the compensator then became a problem to find a signal $f(t)$, such that the impulse response $h(t)$ can be accelerated and scaled to $\frac{1}{\alpha'}h_{\alpha'}(t)$,

$$h_{\alpha'}(t) = f(t) * h(t) = \frac{1}{\alpha'}h\left(\frac{t}{\alpha'}\right) \tag{4.12}$$

The compensator can be designed in either frequency domain or time domain, as long as the impulse response of the designed filter is $f(t)$. A frequency domain design example will be given later.

Input Shaper The input shaper can be designed as:

$$IS_f(t) = \sum_{i=1}^{n} A_i\delta(t-\alpha'\Delta_i) \tag{4.13}$$

where A_i and Δ_i are the same as in Eqn. (4.3).

Since

$$h^{IS}(t) = (IS * h)(t) = \sum_{i=1}^{n} A_i h(t-\Delta_i), \tag{4.14}$$

$$\left(h_{\alpha'} * IS_f\right)(t) = \sum_{i=1}^{n} A_i \frac{h\left(\frac{t}{\alpha'}-\Delta_i\right)}{\alpha'} = \frac{h^{IS}\left(\frac{t}{\alpha'}\right)}{\alpha'} = \frac{1}{\alpha'}h^{IS}_{\alpha'}(t) \tag{4.15}$$

Thus the designed input shaper and compensator can adjust $h^{IS}(t)$ to $\frac{1}{\alpha'}h^{IS}_{\alpha'}(t)$.

Example on a Second Order Linear System Suppose the plant is a second order linear system with transfer function G in Eqn. (4.1). The impulse response of the system is shown in Eqn. (4.2). The transfer function of another system which has impulse response $\frac{1}{\alpha'}h\left(\frac{t}{\alpha'}\right)$ is:

$$G^{Acc}(s) = \frac{K\omega_0^2}{(\alpha's)^2 + 2\alpha'D\omega_0 s + \omega_0^2} \tag{4.16}$$

Then the transfer function $F(s)$ of the compensator can be designed as

$$F(s) = \frac{G^{Acc}(s)}{G(s)} = \frac{s^2 + 2D\omega_0 s + \omega_0^2}{(\alpha's)^2 + 2\alpha'D\omega_0 s + \omega_0^2} \tag{4.17}$$

which is causal. According to the property of Laplace transformation [52],

$$G^{Acc}(s) = F(s)G(s) \Rightarrow \frac{1}{\alpha'} h\left(\frac{t}{\alpha'}\right) = f(t) * h(t) \tag{4.18}$$

Thus the impulse response of this compensator $f(t)$ agrees with Eqn. (4.12).

The input shaper can be firstly designed for $G(s)$ using any input shaping design technique. Then the input shaper for modified zero time delay input shaping can be designed as Eqn. (4.13), i.e. scale time delay for each impulse.

4.4 Experimental Result

The proposed approach has been tested on a FANUC M-16iB industrial robot with an experimental flexible payload as shown in Fig. 4.6.

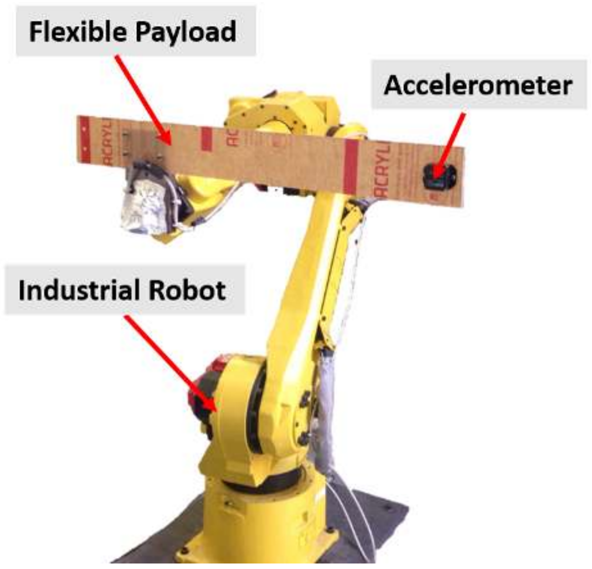


Figure 4.6: Robot with flexible payload

The flexible payload is designed to have a natural frequency similar to the one of a large end-effector of industrial robot. A wireless accelerometer is attached at the end tip of the payload for monitoring residual vibration. In the experiment, the robot is performing a rapid rest-to-rest motion along X direction in the workspace. The motion path is illustrated in Fig. 4.7. The position, velocity, and acceleration reference along X direction are shown in Fig. 4.9.

From the acceleration measured by the wireless accelerometer, the flexible payload can be approximately identified as a second order linear system. Figure 4.8 shows the measured acceleration and the estimated acceleration from the identified model. The identified natural frequency of the system is 2.55Hz, and the damping ratio is 0.04.

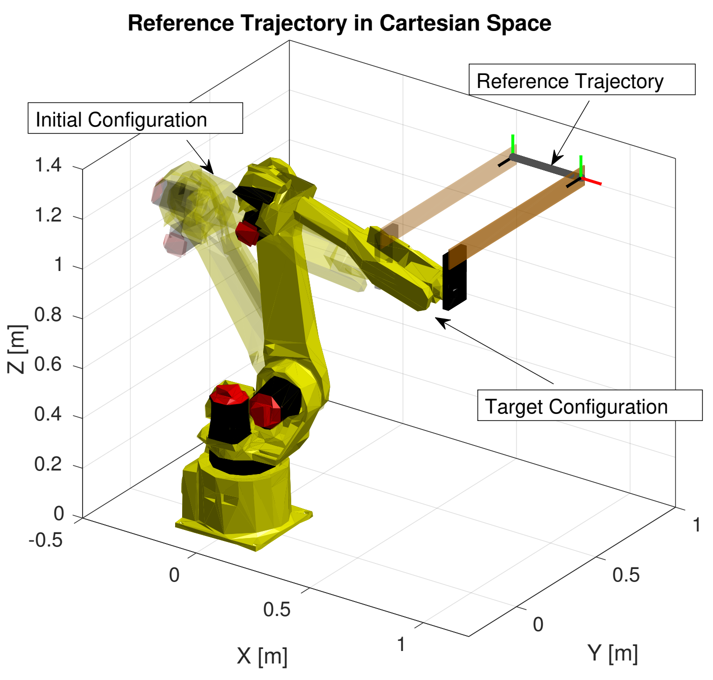


Figure 4.7: Reference trajectory of robot in Cartesian space

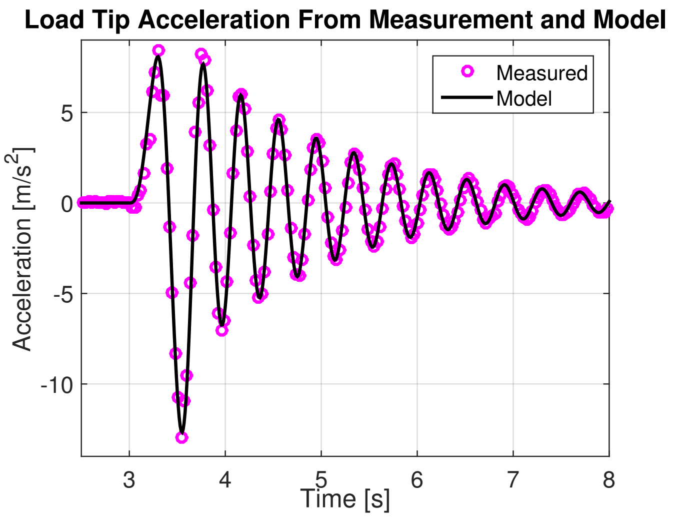


Figure 4.8: Measured and estimated acceleration at the end tip of payload

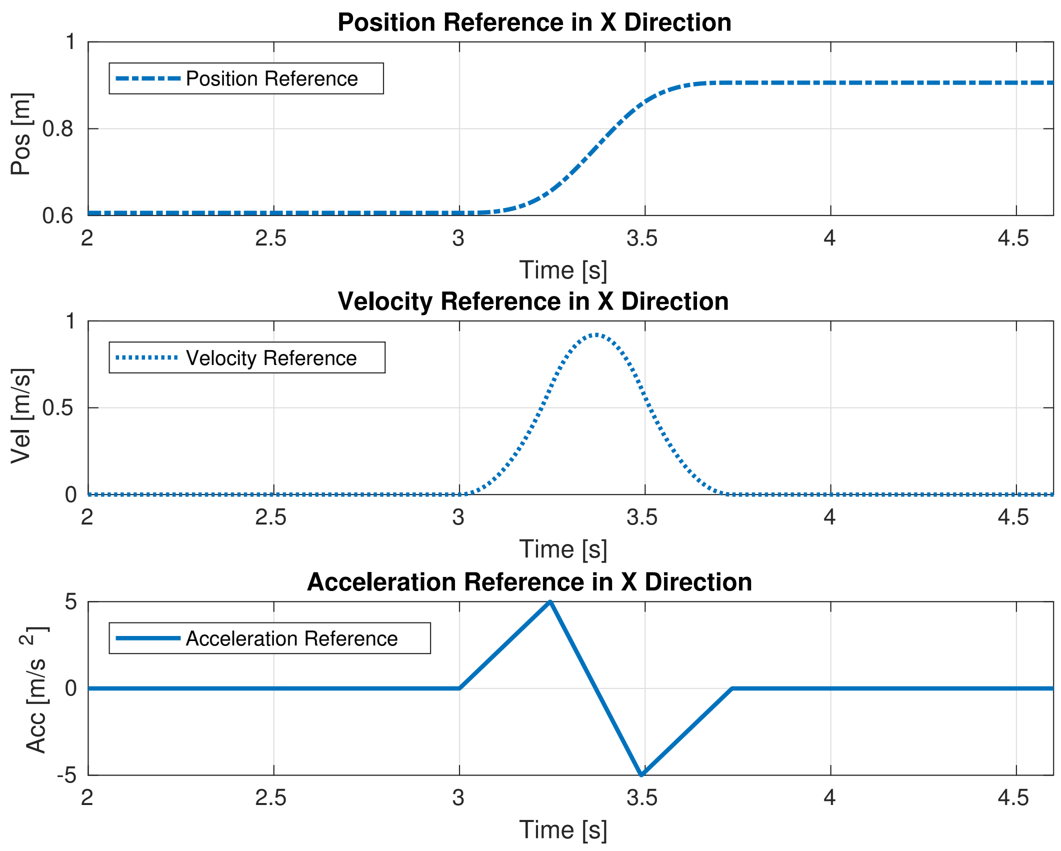


Figure 4.9: Position, velocity, and acceleration reference along X direction

The measured payload tip acceleration of unshaped motion is shown in Fig. 4.10. The tip acceleration of residual vibration reaches as large as about 6 m/s^2. Moreover, the residual vibration lasts more than one seconds, which is longer than the duration of the desired motion.

Input shaping, zero time delay input shaping, and modified zero time delay input shaping are tested in the experiment. All of the three approaches use Zero Vibration and Derivative (ZVD) shaper as $IS(t)$ for robustness consideration [31]. Figure 4.11 shows the measured payload tip acceleration for the three approaches. The desired acceleration is also shown as reference.

As shown in Fig. 4.11, the residual vibration can be effectively suppressed by all of the approaches. The proposed approach and zero time delay input shaping introduce no time delay while input shaping introduces a time delay which adds about 40% time to the original motion. The proposed approach shows smoother acceleration than zero time delay input shaping.

In the experiment, $\alpha' = 0.6516$, and $\alpha = 0.4653$. Such time scaling factors mean that in zero time delay input shaping, the motion reference has been shortened to about 47% of the original

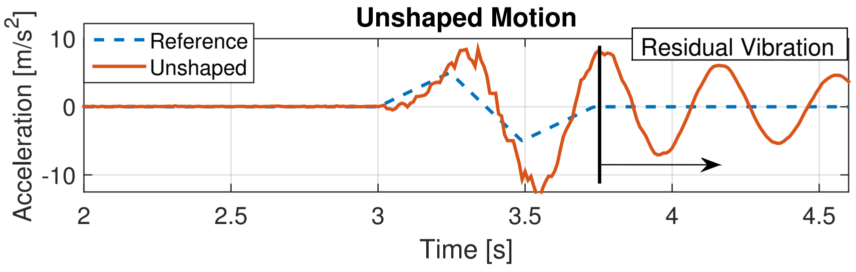


Figure 4.10: Experiment result of unshaped response

length, while it is only to 65% for the proposed approach. Since the modified zero time delay input

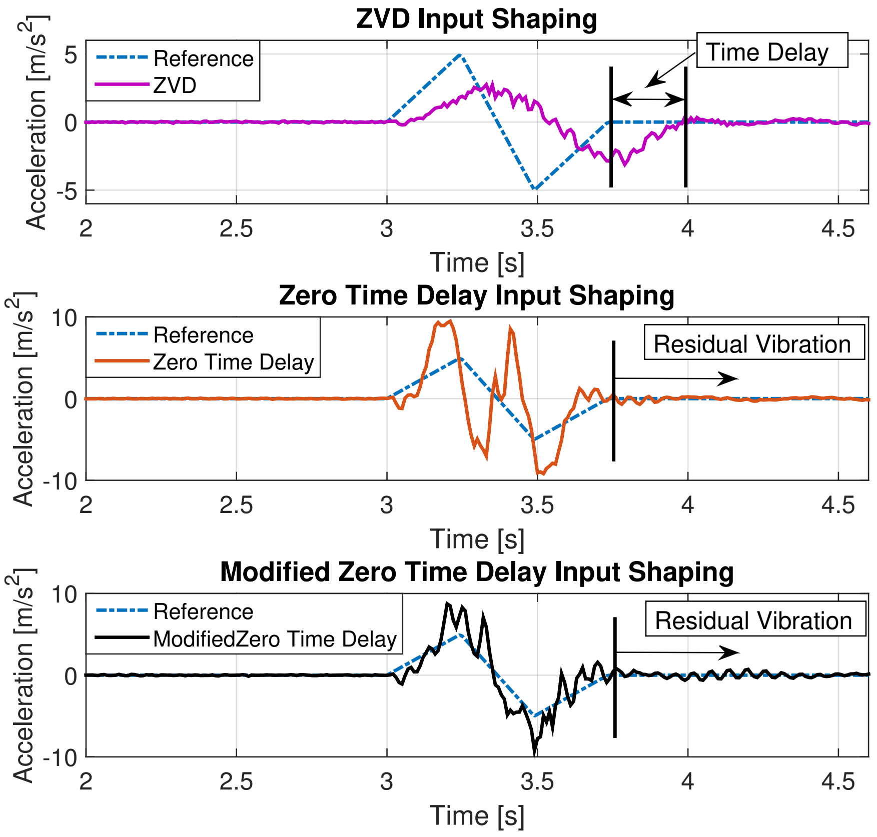


Figure 4.11: Experiment results of three approaches

shaping avoids severe accelerating action, a smoother robot motion can be used to suppress the vibration. Figure 4.12 plots the velocity reference in X direction of the proposed approach and zero time delay input shaping to show the smoothness comparison result more clearly.

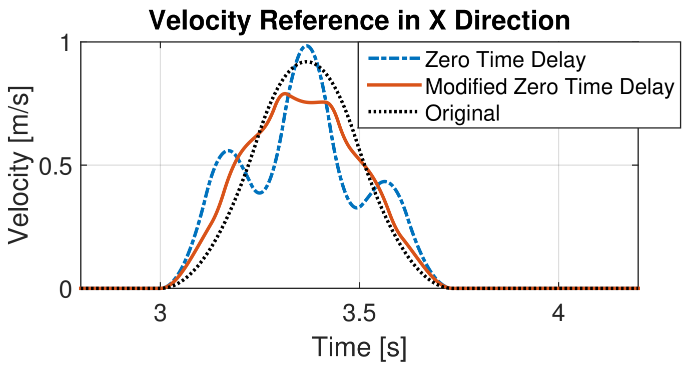


Figure 4.12: Comparison of velocity reference in X direction

4.5 Chapter Summary

In this chapter, a modified zero time delay input shaping approach has been proposed for residual vibration suppression of industrial robot with flexibility. Comparing to existing input shaping techniques, the proposed approach can fully compensate the time delay introduced by conventional input shaping. Furthermore, the proposed approach produces smoother motion than existing techniques for avoiding time delay. Experimental results have been presented to show that the proposed approach outperforms conventional input shaping and zero time delay input shaping in terms of time delay and motion smoothness.

Part II

Intelligent Planning

Chapter 5

Robot Motion Planning Based on Efficient Trajectory Optimization

5.1 Introduction

Motion planning problem is important in robotic applications. A motion planning algorithm generates a continuous motion that connects the initial configuration and target configuration, while avoiding collision with known obstacles. Though algorithmic motion planning has been studied for more than three decades, it is still challenging in industrial robot area. The difficulties come from the complicated geometric structure and highly nonlinear dynamical model of robots. To address these difficulties, the so-called path-velocity decomposition [53] is implemented in most existing solutions. The path-velocity decomposition separated into two subtopics in most existing solutions: path planning and trajectory planning. The path planning problem generates collision free geometric path in configuration, which focuses the complicated geometric model issue. The trajectory planning generates time optimal velocity and acceleration profile along the path, which focuses on the kinodynamic constraints (velocity, acceleration, torque/force bounds). Path planning can be considered as a mature topic since there is a rich collection of path planning algorithms [54, 55]. However, motion planning under kinodynamic constraints which involves complicated robot dynamics still presents a major challenge [56, 57]. Furthermore, though optimality can be achieved for both path planning and trajectory planning, the path-velocity decomposition can still lead to sub-optimal solution for the overall motion planning task. This chapter proposes to address these challenges by modeling motion planning as a general nonlinear optimal control problem. An efficient numerical method for trajectory optimization is then proposed to solve the underlying optimal control problem.

This chapter is organized as follows: section 5.2 presents optimal control formulation for robot motion planning problems, section 5.3 presents an efficient numerical method for trajectory optimization, section 5.4 presents working examples of the proposed approach, and section 5.5 summarizes this chapter.

5.2 Problem Formulation

Formulation of General Optimal Control Problem

The general optimal control problem is posed formally as follows [58]. Determine the state $\boldsymbol{x}(t) \in \mathbb{R}^n$, the control $\boldsymbol{u}(t) \in \mathbb{R}^m$, the vector of static parameters $\boldsymbol{p} \in \mathbb{R}^q$, the initial time $t_0 \in \mathbb{R}$, and the terminal time $t_f \in \mathbb{R}$ that optimizes the performance index (also known as the *cost function*)

$$J = \Phi[\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f; \boldsymbol{p}] + \int_{t_0}^{t_f} \mathcal{L}[\boldsymbol{x}(t), \boldsymbol{u}(t), t; \boldsymbol{p}] \mathrm{d}t \tag{5.1}$$

subject to the *dynamic constraints*

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}[\boldsymbol{x}(t), \boldsymbol{u}(t), t; \boldsymbol{p}] \tag{5.2}$$

the *path constraints* (including bounds of these variables)

$$\boldsymbol{C}_{min} \leq \boldsymbol{C}[\boldsymbol{x}(t), \boldsymbol{u}(t), t; \boldsymbol{p}] \leq \boldsymbol{C}_{max} \tag{5.3}$$

and the *boundary conditions*

$$\boldsymbol{\phi}_{min} \leq \boldsymbol{\phi}[\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f; \boldsymbol{p}] \leq \boldsymbol{\phi}_{max} \tag{5.4}$$

The state, control, and static parameter can each be written in component form

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}; \boldsymbol{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix}; \boldsymbol{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_q \end{bmatrix} \tag{5.5}$$

The term Φ is called endpoint cost or *terminal cost*, and $\mathcal{L}$ is called *running cost* or Lagrangian [59].

In most robotic applications, parameters $\boldsymbol{p}$, initial time t_0, and initial state $\boldsymbol{x}(t_0)$ are pre-defined and thus will be omitted in this chapter. The objective is to find state $\boldsymbol{x}(t)$ and control $\boldsymbol{u}(t)$ as functions of independent variable $t \in [t_0, t_f]$ that are feasible and optimizing the performance index. Since functional space is far more complex than the space of real numbers, solving optimal control problems is more challenging than optimization problems of real numbers.

Formulation of Motion Planning as Optimal Control Problem

The robot motion planning task can be formulated as an optimal control problem by defining the cost function, dynamic constraints, path constraints, and boundary conditions. In industrial robot motion planning, time-optimality is the ultimate goal under physical constraints including collision avoidance, velocity bounds, acceleration bounds, torque/force bounds, or even jerk bounds and torque-rate bounds. The state trajectory $\boldsymbol{x}(t)$ and control trajectory $\boldsymbol{u}(t)$ must satisfy the constraints of robot dynamics. Following most robot motion planning literatures, this chapter only includes rigid body dynamics as it is already challenging enough. The state of a robot motion planning problem can be chosen as joint position and velocity, and the control can be chosen as actuator torque (for articulated robot with n joints):

$$\boldsymbol{x}(t) = \begin{bmatrix} q_1(t) \\ \vdots \\ q_n(t) \\ \dot{q}_1(t) \\ \vdots \\ \dot{q}_n(t) \end{bmatrix}; \boldsymbol{u}(t) = \begin{bmatrix} \tau_1(t) \\ \vdots \\ \tau_n(t) \end{bmatrix}; \tag{5.6}$$

The cost function of a robot motion planning problem can be formulated as:

$$J = t_f + \int_{t_0}^{t_f} \mathcal{L}[\boldsymbol{x}(t), \boldsymbol{u}(t), t]\mathrm{d}t \tag{5.7}$$

where the terminal cost $\Phi = t_f$. The running cost can be chosen in regards to the problem as an optional regulation term that penalize large effort or motion.

The dynamic constraints is the robot dynamics. The equation of motion can be derived using Lagrangian's equations or Newton-Euler approach.

$$\boldsymbol{M}(\boldsymbol{q}(t))\ddot{\boldsymbol{q}}(t) + \boldsymbol{C}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))\dot{\boldsymbol{q}(t)} + \boldsymbol{G}(\boldsymbol{q}(t)) + \boldsymbol{f}^{fric}(t) = \boldsymbol{\tau}(t)$$

The equation of motion can be formulated into a state-space form to fit Equation (5.2).

$$\frac{d}{dt}\begin{bmatrix} \boldsymbol{q}(t) \\ \dot{\boldsymbol{q}}(t) \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}}(t) \\ \boldsymbol{M}(\boldsymbol{q}(t))^{-1}\left[\boldsymbol{\tau}(t) - \boldsymbol{C}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))\dot{\boldsymbol{q}(t)} - \boldsymbol{G}(\boldsymbol{q}(t)) - \boldsymbol{f}^{fric}(t)\right] \end{bmatrix} \tag{5.8}$$

A set of path constraints can be formulated to accommodate various physical limitations of a robot. Typical path constraints include but are not limited to

Joint position bounds:	$q_i^{min}(t) \leq q_i(t) \leq q_i^{max}(t), i = 1, \cdots, n$	(5.9a)
Joint velocity bounds:	$\dot{q}_i^{min}(t) \leq \dot{q}_i(t) \leq \dot{q}_i^{max}(t), i = 1, \cdots, n$	(5.9b)
Joint acceleration bounds:	$\ddot{q}_i^{min}(t) \leq \ddot{q}_i(t) \leq \ddot{q}_i^{max}(t), i = 1, \cdots, n$	(5.9c)
Joint jerk bounds:	$\dddot{q}_i^{min}(t) \leq \dddot{q}_i(t) \leq \dddot{q}_i^{max}(t), i = 1, \cdots, n$	(5.9d)
Joint torque bounds:	$\tau_i^{min}(t) \leq \tau_i(t) \leq \tau_i^{max}(t), i = 1, \cdots, n$	(5.9e)
Joint torque rate bounds:	$\dot{\tau}_i^{min}(t) \leq \dot{\tau}_i(t) \leq \dot{\tau}_i^{max}(t), i = 1, \cdots, n$	(5.9f)

where joint velocity and acceleration bounds are considered to guarantee smoothness of the generated motion, joint position bounds and torque bounds are considered for safety and feasibility, joint jerk bounds and torque rate bounds are considered to address the joint oscillation problem caused by discontinuous actuator torques [60].

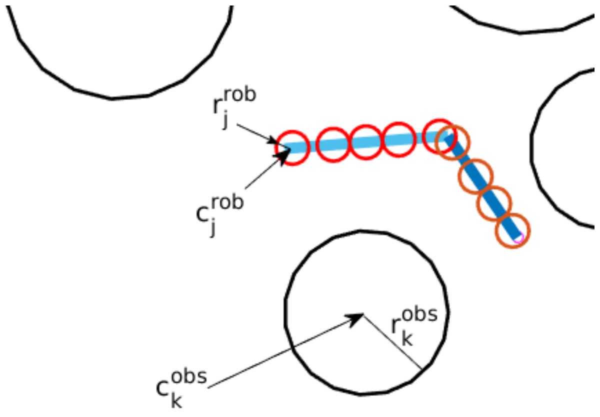


Figure 5.1: Sphere approximation of robot and obstacles

Besides the physical limitations, collision avoidance can be formulated as additional path constraints. Dues to the complicated geometric mapping between robot workspace and configuration space, it is difficult to represent collision free condition analytically. To simplify the formulation, a finite number of spheres can be used to approximate the known obstacles and robot links as shown in Fig. 5.1. The approximation can be performed either manually or automatically using sphere-trees construction algorithms [61].

Suppose M spheres are used to approximate robot links, and S spheres are used to approximate obstacles. Let the center and radius of each sphere that representing robot links be $\boldsymbol{c}_j^{rob}(\boldsymbol{q}(t)), r_j^{rob}, j = 1, \cdots, M$, the center and radius of each sphere that representing obstacles be $\boldsymbol{c}_k^{obs}, r_k^{obs}, k = 1, \cdots, S$. Robot forward kinematics problem can be solved to determine the functional relationship between joint positions $\boldsymbol{q}(t)$ and the location of sphere centers $\boldsymbol{c}_j^{rob}(\boldsymbol{q}(t))$. The collision free constraint can then be formulated as

$$\|\boldsymbol{c}_j^{rob}(\boldsymbol{q}(t)) - \boldsymbol{c}_k^{obs}\|_2 \geq r_j^{rob} + r_k^{obs} + \epsilon, \forall t \in [t_0, t_f], j = 1, \cdots, M, k = 1, \cdots, S \tag{5.10}$$

where $\epsilon > 0$ is a threshold that guarantee strict collision free condition.

Typical boundary conditions for robot motion planning problem include

$$\text{Initial state:} \qquad q_i(t_0) = q_i^0, \dot{q}_i(t_0) = \dot{q}_i^0, i = 1, \cdots, n \tag{5.11a}$$

$$\text{Target state:} \qquad q_i(t_f) = q_i^f, \dot{q}_i(t_f) = \dot{q}_i^f, i = 1, \cdots, n \tag{5.11b}$$

$$\text{Terminal time bounds :} \qquad t_f^{min} \leq t_f \leq t_f^{max} \tag{5.11c}$$

where q_i^0 and q_i^f are initial and target joint positions of robot joint i, and $\dot{q}_i^0, \dot{q}_i^f$ are the respective initial and target joint velocities. $t_f^{min} > 0$ is the minimum allowed terminal time, and t_f^{max} is the maximum allowed terminal time.

The formulation of robot motion planning is a general nonlinear optimal control problem. Since there is no analytical solution found for the general problem, numerical methods are investigated in the next section.

5.3 Numerical Methods for Trajectory Optimization

Trajectory optimization is the process of designing state and control trajectories that optimize some measure of performance while satisfying a set of constraints [62]. Generally speaking, trajectory optimization is a technique for computing an open-loop solution to an optimal control problem. Since optimal control is a general control strategy, trajectory optimization can be implemented in a variety of control applications, ranging from aerospaces, transportation, to industrial processes, robotics, and so on.

In the early years of trajectory optimization research, efforts have been focusing on analytical solutions based on calculus of variations. Though important results including Hamilton-Jacobi-Bellman equation and Pontryagin's maximum principle are proposed, no general analytical solution can be found (except for the simple linear-quadratic regulator (LQR) problem). As a result, it is necessary to implement numerical methods for most real-world optimal control problems. A variety of numerical approaches have been developed during the last several decades.

Review of Numerical Method

Numerical methods have been developed for trajectory optimization[58, 63]. The numerical methods can be classified into two types: indirect methods and direct methods. Due to the requirements of complicated derivation of optimality condition and the solution of challenging boundary-value problems, indirect methods are difficult to be implemented practically. Direct methods have gained popularity during the last several years. The focus of this chapter is therefore direct methods.

In regards to the direct method, the control trajectory(and state trajectory) is discretized first, which is known as the transcription procedure [64]. The transcription converts a continuous optimal control problem into a nonlinear optimization problem as follows:

$$\begin{aligned}
\text{minimize} \quad & \Phi[\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_f] + \sum_{i=0}^{N} w_i \mathcal{L}[\boldsymbol{x}(T_i), \boldsymbol{u}(T_i), T_i] \\
\text{subject to} \quad & \dot{\boldsymbol{x}}(T_i) = \boldsymbol{f}[\boldsymbol{x}(T_i), \boldsymbol{u}(T_i), T_i] \\
& \boldsymbol{C}_{min} \leq \boldsymbol{C}[\boldsymbol{x}(T_i), \boldsymbol{u}(T_i), T_i] \leq \boldsymbol{C}_{max} \\
& \boldsymbol{\phi}_{min} \leq \boldsymbol{\phi}[\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_f] \leq \boldsymbol{\phi}_{max}
\end{aligned} \tag{5.12}$$

where the integration of running cost is approximated using quadrature with weights w_i. In the transcribed problem, $\{T_i,\ i = 0, \cdots N\}$ are called knots. As the solution of the transcribed problem, the optimal control at knots $\{\boldsymbol{u}(T_i),\ i = 0, \cdots N\}$ (and optimal state at knots $\{\boldsymbol{x}(T_i),\ i = 0, \cdots N\}$, and t_f) can be returned by any nonlinear optimization solver. Polynomial interpolation is then used to approximate the continuous time optimal control trajectory (and state trajectory) based on the value at knots. The type of polynomial interpolation and quadrature approximation depends on the transcription method. The schematic of numerical method for trajectory optimization is shown in Fig. 5.2.

There are two kinds of transcription approaches [58, 64, 65]: shooting and collocation. The shooting approach is illustrated in Fig. 5.3a. The decision variables of the transcribed optimization

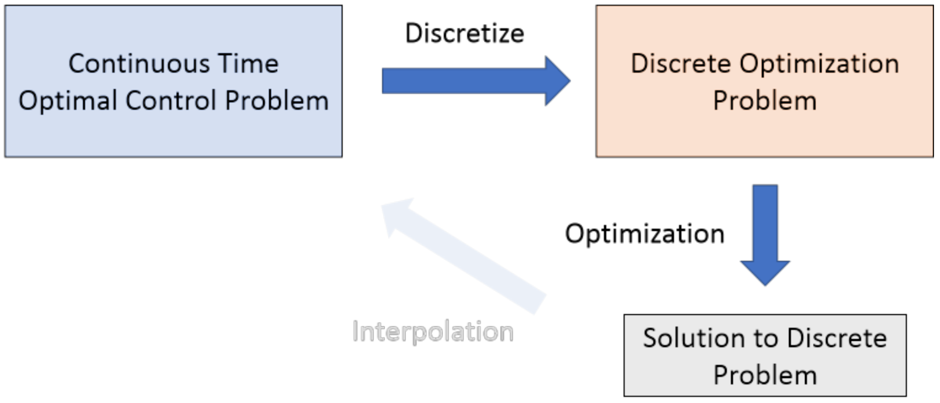
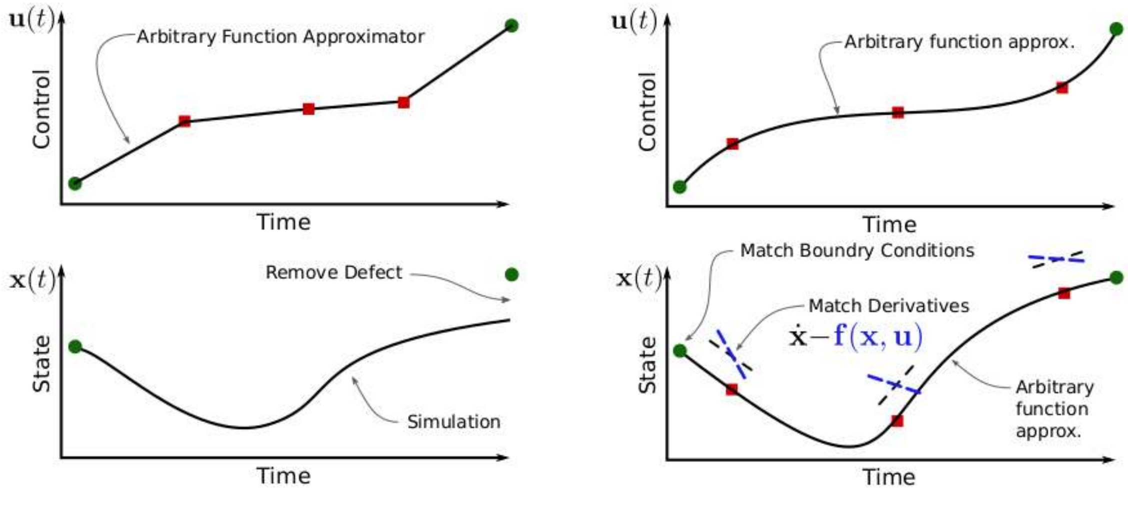

Continuous Time Optimal Control Problem
Discretize
Discrete Optimization Problem
Optimization
Solution to Discrete Problem
Interpolation

Figure 5.2: Schematic of numerical method for trajectory optimization

problem are the $\{\boldsymbol{u}(T_i),\ i=0,\cdots N\}$, which are marked as red dots in the figure. The control $\boldsymbol{u}(t)$ is approximated by arbitrary function approximater (e.g. piece-wise linear function that passes all red dots in the figure). The cost function J is evaluated by quadrature approximation, in which the state values $\boldsymbol{x}(T_i)$ are calculated using numerical integration of the differential equations that describe the dynamical system (e.g. Equation (5.2), (5.8)). The optimization solver tries to adjust the values of $\{\boldsymbol{u}(T_i),\ i=0,\cdots N\}$ such that J is minimized under the condition that the all constraints are satisfied (e.g. the final state $\boldsymbol{x}(t_f)$ reaches a certain goal).



(a) Shooting (b) Collocation

Figure 5.3: Difference between shooting and collocation

The collocation approach is illustrated in Fig. 5.3b. Similar to the shooting approach, $\{\boldsymbol{u}(T_i),\ i=0,\cdots N\}$ are decision variables. But unlike the shooting approach, the decision variables also include $\{\boldsymbol{x}(T_i),\ i=0,\cdots N\}$, which are marked as red dots in the lower part of Fig. 5.3b. The state $\boldsymbol{x}(t)$ is approximated by arbitrary function approximater. In the transcribed optimization problem, the dynamic constraint $\dot{\boldsymbol{x}}(T_i) = \boldsymbol{f}[\boldsymbol{x}(T_i), \boldsymbol{u}(T_i), T_i]$ is applied to guarantee feasibility of the solution. The optimization solver tries to adjust the values of $\{\boldsymbol{u}(T_i),\, i = 0, \cdots N\}$ and $\{\boldsymbol{x}(T_i),\, i = 0, \cdots N\}$ such that the cost function J is minimized under a set of constraints. In general, the collocation approach requires more decision variables than the shooting approach, but the transcribed optimization problem is easier to solve.

Any nonlinear optimization algorithm can be implemented to solve the transcribed problem. The most commonly used methods are sequential quadratic programming and interior-point methods. Extensive research has been performed in the past several decades. The research has lead to extremely versatile and robust softwares for the numerical solution of nonlinear optimization problems. The details of these two methods are omitted in this dissertation. It is worth noting that most of optimization solvers are based on gradient descent algorithm, thus the calculation of Jacobian or Hessian matrix are necessary.

Efficient Numerical Method for Trajectory Optimization

In regards to the schematic of numerical method for trajectory optimization, two parts are playing the key role: discretization and optimization. An efficient implementation can be designed by choosing these two components smartly. In this dissertation, the pseudospectral method is chosen to transcribe the continuous time optimal control problem, and the interior point method with the support of automatic differentiation is chosen to solve the transcribed discrete optimization problem. The schematic of the efficient numerical method is illustrated in Fig. 5.4.

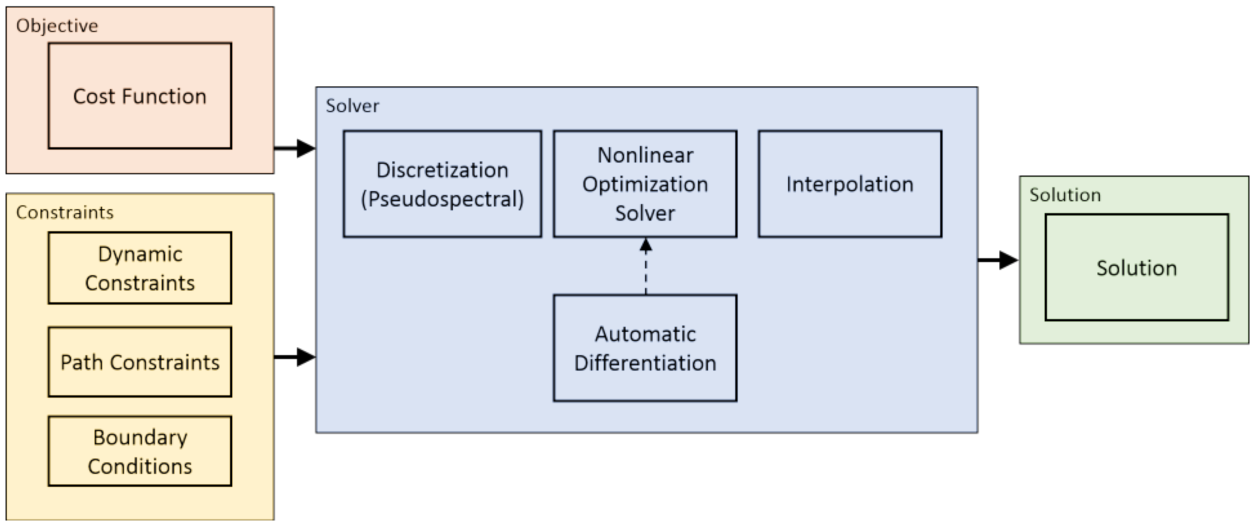


Figure 5.4: Frame work of the proposed efficient numerical method

Intelligent Discretization

It is argued that there exists complicated nonlinear mapping between decision variables and cost / constraint functions in shooting approaches [64]. The resulting optimization problem can not be well approximated by local linearized models adopted by most solvers. Collocation approaches are considered the most powerful methods for solving general optimal control problems.

In collocation approaches, state and control trajectories are approximately parametrized by polynomials. Runge's phenomenon is well known as a problem of polynomial interpolation that oscillation occurs when interpolation is performed over a set of equispaced nodes using high order polynomials [66]. Two solutions exist to mitigate this issue: using piecewise low order polynomials or using special interpolation nodes. The two mitigation solutions lead to two common forms of collocations: *local collocation* and *global collocation*. A local collocation parametrizes state and control trajectories using piecewise low order polynomials and a global collocation uses high order polynomials with interpolation nodes that are specially designed. The global collocation is also known as *pseudospectral methods*. Pseudospectral methods have increased in popularity in recent years. The advantage of pseudospectral methods is that the polynomial approximation converges exponentially fast with the increase of interpolation nodes [58, 67, 68]. If efficiency is the major concern, then using pseudospectral methods is a good choice since less interpolation nodes can be chosen to reduce the scale of the transcribed optimization problem.

1. **Choice of knots**:

   In pseudospectral methods, it is assumed that the functions we would like to approximate have been mapped to the interval $\zeta \in [-1, 1]$. Suppose state and control trajectories are initially defined in $T \in [t_0, t_f]$, and $t_0 = 0$, then the mapping can be achieved by

$$\zeta = 2\frac{T - t_0}{t_f - t_0} - 1 = 2\frac{T}{t_f} - 1 \tag{5.13}$$

   The knots $\{\zeta_i, i = 0, \cdots, N\}$ in pseudospectral theory are typically chosen to be the root of orthogonal polynomials. In this chapter, Chebyshev orthogonal polynomials are chosen for the calculation of knots since it is easy to compute. The Chebyshev-Lobatto points (or Chebyshev points) are chosen to be

$$\zeta_i = \cos\left(\frac{i\pi}{N}\right), i = 0, \cdots, N \tag{5.14}$$

   and thus the knots can be chosen as

$$T_i = \frac{t_f}{2}\left[\cos\left(\frac{i\pi}{N}\right) + 1\right], i = 0, \cdots, N \tag{5.15}$$

2. **Interpolation**

   The polynomial interpolation in pseudospectral methods can be performed by *barycentric interpolation* [68, 69].

The barycentric interpolation can be formulated as a linear combination of Lagrangian polynomials. For state trajectory and control trajectory, the form is

$$\begin{aligned} x_i(T) &= \sum_{j=0}^{N} x_i(T_j)\ell_j(T), i = 1, \cdots, n \\ u_i(T) &= \sum_{j=0}^{N} u_i(T_j)\ell_j(T), i = 1, \cdots, m \end{aligned} \tag{5.16}$$

where $\ell_j(T)$ is the jth Lagrange polynomial. In barycentric interpolation, a special form of Lagrange polynomial is implemented. This form can be derived as follows:

$$\begin{aligned} \ell_j(T) &= \prod_{k\neq j}(T - T_k) \Big/ \prod_{k\neq j}(T_j - T_k) \\ &= \prod_{k=0}^{N}(T - T_k) \Big/ (T - T_j)\prod_{k\neq j}(T_j - T_k) \end{aligned} \tag{5.17}$$

In a special case $\ell_j(T_j) = 1$ if $T = T_j$. Otherwise let the interpolation weights v_j be

$$v_j = \frac{1}{\prod_{k\neq j}(T_j - T_k)}, j = 0, \cdots, N \tag{5.18}$$

Let $\ell(T) = \prod_{k=0}^{N}(T - T_k)$, then

$$\ell_j(T) = \frac{\ell(T)v_j}{(T - T_j)} \tag{5.19}$$

Since $\sum_{j=0}^{N} \ell_j(T) = 1$, the special form of jth Lagrange polynomial can be formulated as

$$\ell_j(t) = \frac{v_j}{T - T_j} \Bigg/ \sum_{k=0}^{N} \frac{v_k}{T - T_k}, j = 0, \cdots, N \tag{5.20}$$

Interestingly, if interpolation weights v_j is scaled by an arbitrary constant gain $\alpha \neq 0$, $v'_j = \alpha v_j$ does not affect ℓ_j. When Chebyshev points are chosen, the interpolation weights v_j and Lagrange polynomials can be simplified as

$$v_j = (-1)^j \tag{5.21a}$$

$$\ell_j(T) = \frac{(-1)^j}{T - T_j} \Bigg/ \left[\sum_{k=1}^{N-1} \frac{(-1)^k}{T - T_k} + \frac{1}{2}\left(\frac{1}{T - T_0} + \frac{(-1)^N}{T - T_N}\right)\right] \tag{5.21b}$$

for $j = 1, \cdots, N - 1$, and half of the values for $j = 0$ and N.

The interpolation procedure is not used in the transcription step. Once the decision variables $x_i(T_j)$, $u_i(T_j)$, and t_f are obtained, the approximate continuous solution can be evaluated using the interpolation method.

3. **Quadrature**

   Quadrature is the standard term for the numerical calculation of integrals. The running cost in integration form can be approximately evaluated by quadrature rules in pseudospectral methods. Let $t_0 = 0$, the quadrature rule to evaluate the integration of running cost is

$$\begin{aligned}\int_0^{t_f} \mathcal{L}[\boldsymbol{x}(T), \boldsymbol{u}(T), T]\mathrm{d}T &\approx \int_0^{t_f} \left[\sum_{j=0}^{N} \ell_j(T)\mathcal{L}[\boldsymbol{x}(T), \boldsymbol{u}(T), T]\right] \mathrm{d}T \\ &= \sum_{j=0}^{N} w_j \mathcal{L}[\boldsymbol{x}(T_j), \boldsymbol{u}(T_j), T_j]\end{aligned} \tag{5.22}$$

   where $\{w_j, j = 0, \cdots, N\}$ is a set of quadrature weights. The quadrature weights can be explicitly defined to be

$$w_j = \int_0^{t_f} \ell_j(T)\mathrm{d}T, j = 0, \cdots, N \tag{5.23}$$

   Referring to Equations (5.13) and (5.21b), the jth Lagrange polynomial can be written as function of $\zeta \in [-1, 1]$, and Equation (5.23) can be rewritten as

$$\begin{aligned} w_j &= \int_0^{t_f} \ell_j(T)\mathrm{d}T \\ &= \frac{t_f}{2} \int_{-1}^{1} \ell_j^s(\zeta)\mathrm{d}\zeta \end{aligned} \tag{5.24}$$

   where

$$\ell_j^s(\zeta) = \frac{(-1)^j}{\zeta - \zeta_j} \Bigg/ \left[\sum_{k=1}^{N-1} \frac{(-1)^k}{\zeta - \zeta_k} + \frac{1}{2}\left(\frac{1}{\zeta - \zeta_0} + \frac{(-1)^N}{\zeta - \zeta_N}\right)\right] \tag{5.25}$$

   When Chebyshev points are chosen, the corresponding quadrature rule is called ClenshawCurtis quadrature. Explicit expression of the quadrature weights can be found in [70]:

$$\int_{-1}^{1} \ell_j^s(\zeta)\mathrm{d}\zeta = \frac{c_j}{N}\left(1 - \sum_{k=1}^{\lfloor N/2 \rfloor} \frac{b_k}{4k^2 - 1}\cos(2k\zeta_j)\right), j = 0, \cdots, N \tag{5.26}$$

   where the coefficients b_k, c_j are defined as

$$b_k = \begin{cases} 1, & k = N/2 \\ 2, & k < N/2 \end{cases}, \quad c_j = \begin{cases} 1, & j = 0, N \\ 2, & j = 1, \cdots, N-1 \end{cases} \tag{5.27}$$

4. **Differentiation matrix**

   One advantage of polynomial interpolation is that it is easy to approximate derivative easily.

Let the stacked state and control at knots be

$$\boldsymbol{X}_i = \begin{bmatrix} x_i(T_0) \\ \vdots \\ x_i(T_N) \end{bmatrix}, \quad \boldsymbol{U}_j = \begin{bmatrix} u_j(T_0) \\ \vdots \\ u_j(T_N) \end{bmatrix} \tag{5.28}$$

where $i = 1, \cdots, n$, $j = 1, \cdots, m$. The derivatives of the state and control at knots can be calculated from the polynomial approximation

$$\frac{\mathrm{d}}{\mathrm{d}T}\boldsymbol{X}_i = \frac{2}{t_f}\boldsymbol{D}\boldsymbol{X}_i, \quad \frac{\mathrm{d}}{\mathrm{d}T}\boldsymbol{U}_j = \frac{2}{t_f}\boldsymbol{D}\boldsymbol{U}_j \tag{5.29}$$

where $\boldsymbol{D}$ is the differential matrix that is used to compute the derivative of the $\boldsymbol{X}_i$ and $\boldsymbol{U}_j$ with respect to $\zeta \in [-1, 1]$. The derivation of $\boldsymbol{D}$ can be found in [69]. Each element in the ith row and jth column of the differential matrix can be computed as shown in [67]

$$D_{ij} = \begin{cases} \dfrac{v_j/v_i}{\zeta_i - \zeta_j} & i \neq j \\ -\displaystyle\sum_{i \neq j} D_{ij} & i = j \end{cases} \tag{5.30}$$

Let

$$\boldsymbol{F}_i = \begin{bmatrix} f_i\left[\boldsymbol{x}(T_0), \boldsymbol{u}(T_0), T_0\right] \\ \vdots \\ f_i\left[\boldsymbol{x}(T_N), \boldsymbol{u}(T_N), T_N\right] \end{bmatrix}, \quad i = 1, \cdots, n \tag{5.31}$$

The discretized dynamic constraints can be formulated using differentiation matrix as

$$\boldsymbol{D}\boldsymbol{X}_i = \frac{t_f}{2}\boldsymbol{F}_i, \quad i = 1, \cdots, n \tag{5.32}$$

The intelligent discretization utilizes pseudospectral method to transcribe continuous time optimal control problem into optimization problem with decision variables $\boldsymbol{x}(T_j)$, $\boldsymbol{u}(T_j)$, and t_f (for time-optimal planning problems). The solution to the continuous time optimal control problem can be well approximated by polynomial interpolation with relatively small amount of discretization points. Another advantage is that the smoothness of the returned solution can be guaranteed by the global polynomial approximation.

Automatic Differentiation

In order to further improve the efficiency of the numerical method, automatic differentiation is introduced to accelerate the solution procedure for the optimization problem.

Lots of optimization solvers are based on gradient descent algorithm. Derivative of objective function and constraints are frequently evaluated by numerical differentiation approaches, which perturbs input to the function in each dimension to obtain an approximation of the derivative using finite differences. However, numerical differentiation approaches are computationally expensive for functions with high dimensional input, and inevitably introduces round-off errors. Symbolic differentiation is one way to avoid round-off errors, however it frequently leads to inefficient code. Both numerical differentiation and symbolic differentiation are problematic in the calculation of higher order derivatives like Hessian.

To address the problems in numerical differentiation and symbolic differentiation, automatic differentiation is introduced. Automatic differentiation is a set of techniques to evaluate derivative of a function [71]. Automatic differentiation is typically implemented as programming approaches. The implementation is based on dual number arithmetics. One simple example is to calculate the derivative of a given function $y = F(x)$. Replace input x with the number $x + x'\epsilon$, where x' is a real number and ϵ is an abstract number with the property $\epsilon^2 = 0$. Using Taylor expansion, the function value can be calculated as

$$y + y'\epsilon = F(x + x'\epsilon) = F(x) + \sum_{i=1}^{\infty} \frac{F^{(i)}(x)}{i!}(x'\epsilon)^i = F(x) + F^{(1)}(x)x'\epsilon$$

where $F^{(i)}(x)$ is the ith derivative of function F. Define a dual number

$$\langle x, x' \rangle = x + x'\epsilon$$

then the function F is ‘overloaded’ as

$$\langle y, y' \rangle = F(\langle x, x' \rangle) = \langle F(x), F^{(1)}x' \rangle$$

When $x' = 1$ is used, the function value $F(x)$ and derivative $F^{(1)}(x)$ of function F is directly obtained by one evaluation as the two components in the returned dual number. More dual components can be introduced to handle function with multiple inputs. It is obvious that non round-off errors are introduced by automatic differentiation. The computational cost of automatic differentiation is lower than numerical differentiation or symbolic differentiation. Efficient implementation of automatic differentiation includes Adol-C [72], CppAD [73], and FADBAD++ [74]. In this dissertation, CasADi[75] is chosen for its good usability in MATLAB environment.

5.4 Numerical Examples and Experimental Results

Planar Robot Example

Time-optimal motion planning of a planar wafer-handling robot with kinematic constraints and workspace collision avoidance requirement is considered. The wafer-handling robot is shown in Fig. 5.5. This type of wafer handling robot is used to transfer silicon wafers from one point to another automatically inside a semiconductor manufacturing machine.

The wafer handling robot is designed to have three degrees of freedom for arbitrary planar positioning purpose. Since wafer transferring is only one small amount work in a semiconductor

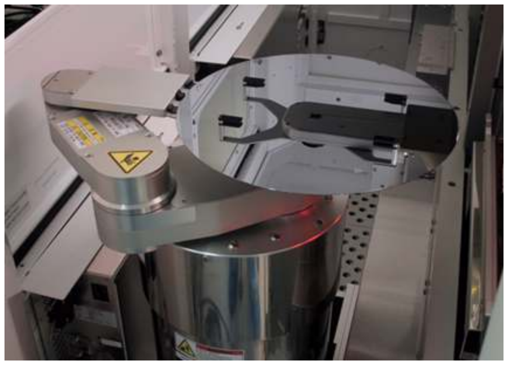


Figure 5.5: Wafer handling robot

manufacturing machine, the workspace of wafer handling robot is designed to be compact. The geometric model of the wafer handling robot and its workspace is shown in Fig. 5.6. The task of the planar robot is moving wafer from loading port to unloading port. Three steps can be used for the motion: picking wafer from loading port to a location near the port, moving to a location near unloading port, and placing wafer to unloading port. Picking and placing in the first and last step are easy. Though the design of wafer handling robot has prevented interference among links, the second step is challenging due to the limited workspace.

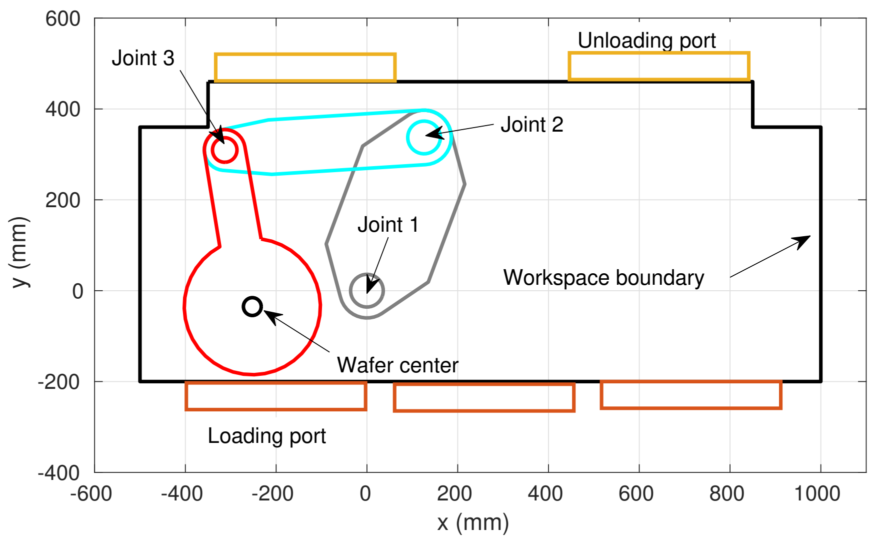


Figure 5.6: Geometric model of wafer handling robot in workspace

Let the three joint positions be $q_1^{wr}(t)$, $q_2^{wr}(t)$, and $q_3^{wr}(t)$, where $t \in [0, t_f]$. The planning objective is to minimize cycle time for the motion, thus the cost function can be formulated as

$$J^{wr} = t_f \tag{5.33}$$

Wafer handling robots typically come with high torque actuators, thus torque saturation is not the concern of the planning task. Therefore the 'robot dynamics' can be formulated by differential kinematics. The state and control of this dynamical system can be chosen to be

$$\boldsymbol{x}^{wr}(t) = \begin{bmatrix} \boldsymbol{q}^{wr}(t) \\ \dot{\boldsymbol{q}}^{wr}(t) \\ \ddot{\boldsymbol{q}}^{wr}(t) \end{bmatrix}, \quad \boldsymbol{u}^{wr}(t) = \dddot{\boldsymbol{q}}^{wr}(t) \tag{5.34}$$

where $\boldsymbol{q}^{wr}(t) = [q_1^{wr}(t), q_2^{wr}(t), q_3^{wr}(t)]^T$ is the vector for joint positions. There are 9 states and 3 controls.

The dynamic constraint of this task can be formulated as

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \boldsymbol{q}^{wr}(t) \\ \dot{\boldsymbol{q}}^{wr}(t) \\ \ddot{\boldsymbol{q}}^{wr}(t) \end{bmatrix} = \begin{bmatrix} 0 & \boldsymbol{I} & 0 \\ 0 & 0 & \boldsymbol{I} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{q}^{wr}(t) \\ \dot{\boldsymbol{q}}^{wr}(t) \\ \ddot{\boldsymbol{q}}^{wr}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \boldsymbol{I} \end{bmatrix} \dddot{\boldsymbol{q}}^{wr}(t) \tag{5.35}$$

The path constraints of this planning task include bounded joint velocity, acceleration, and jerk determined by actuator capability. The linear acceleration at wafer center should also be bounded to avoid sliding of wafers, which particle contamination or even tip-over of wafer [76]. In addition, collision between the robot and the workspace boundary should be avoided. Sphere approximation of robot links and the workspace boundary is shown in Fig. 5.7. The sphere centers are denoted

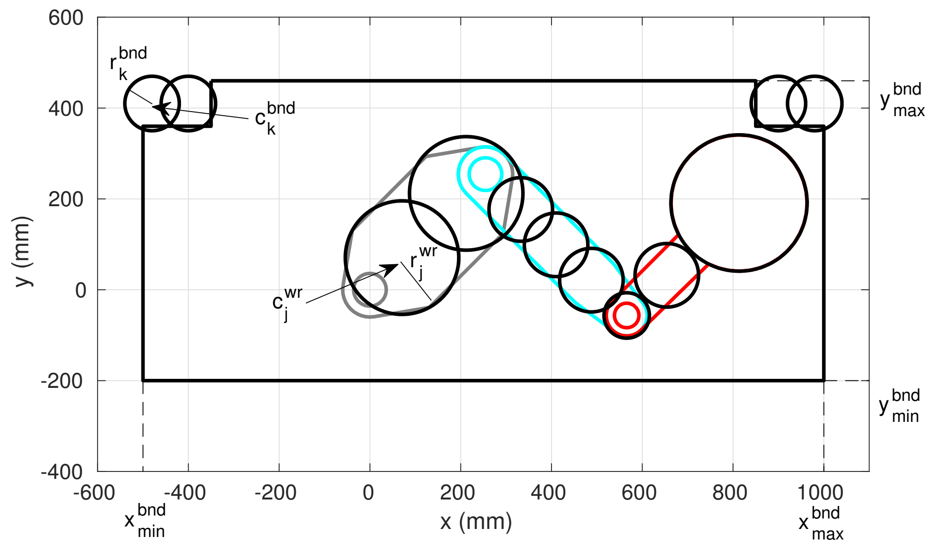


Figure 5.7: Sphere approximation of planar robot and workspace boundary

by $\boldsymbol{c}_j^{wr}$ $(j = 1, \cdots, 8)$ for robot, and $\boldsymbol{c}_k^{bnd}$ $(k = 1, \cdots, 4)$ for workspace boundary. The radius are denoted by r_j^{wr} for robot, and r_k^{wr} for workspace boundary. The workspace is also bounded by x_{min}^{bnd}, x_{max}^{bnd} in X direction and y_{min}^{bnd}, y_{max}^{bnd} in Y direction.

The path constraints of this problem can be formulated as

Joint velocity bounds: $\dot{\boldsymbol{q}}^{wr}_{min} \leq \dot{\boldsymbol{q}}^{wr}(t) \leq \dot{\boldsymbol{q}}^{wr}_{max}$ (5.36a)

Joint acceleration bounds: $\ddot{\boldsymbol{q}}^{wr}_{min} \leq \ddot{\boldsymbol{q}}^{wr}(t) \leq \ddot{\boldsymbol{q}}^{wr}_{max}$ (5.36b)

Joint jerk bounds: $\dddot{\boldsymbol{q}}^{wr}_{min} \leq \dddot{\boldsymbol{q}}^{wr}(t) \leq \dddot{\boldsymbol{q}}^{wr}_{max}$ (5.36c)

Workspace acceleration bounds: $\boldsymbol{a}^{wr}_{min} \leq \boldsymbol{f}^{wr}\left(\boldsymbol{q}^{wr}(t), \dot{\boldsymbol{q}}^{wr}(t), \ddot{\boldsymbol{q}}^{wr}(t)\right) \leq \boldsymbol{a}^{wr}_{max}$ (5.36d)

Workspace bounds: $\left[x^{bnd}_{min}, y^{bnd}_{min}\right]^T \leq \boldsymbol{c}^{wr}_j(t) \leq \left[x^{bnd}_{max}, y^{bnd}_{max}\right]^T$ (5.36e)

Corner collision avoidance: $\|\boldsymbol{c}^{wr}_j(\boldsymbol{q}(t)) - \boldsymbol{c}^{bnd}_k\|_2 \geq r^{wr}_j + r^{bnd}_k + \epsilon$ (5.36f)

where $\boldsymbol{c}^{wr}_j$ can be calculated using forward kinematics of this planar robot. The function $\boldsymbol{f}^{wr}$ represents differential forward kinematics for the calculation of workspace acceleration at wafer center. The kinematic limitations of this robot is listed in Table 5.1.

Table 5.1: Kinematic limits of wafer handling robot

	Kinematic limits in joint space		
Joint #	Velocity(rad/s)	Acceleration (rad/s^2)	Jerk (rad/s^3)
1	4.2	8.4	10
2	8.4	16.8	10
3	6.3	12.6	10
	Kinematic limits in workspace		
Axis	Acceleration (mg)		
x	100		
y	100		

The boundary condition can be formulated as Equation (5.11) by specifying initial and target position, and the bounds for t_f. The cost function, dynamic constraints, path constraints, and boundary conditions are then provided to an optimal control solver. $N = 19$ is chosen to obtain 20 knots in this task, resulting an optimization problem with $9 \times 20 + 3 \times 20 + 1 = 241$ decision variables. One initial guess of these variables are obtained by choosing all zero velocities, accelerations, and jerks. The position variables are initialized to linearly connecting the initial and the target position. Optimal path and trajectory are simultaneously returned by the optimal control solver.

Test case 1 The target position is on the right portion in the workspace. The initial guess of the position trajectory is shown in Fig. 5.8. t_f is initialized to be 10 seconds. It is clear that the initial guess does not provide a feasible trajectory.

The optimization problem is solved using Ipopt solver [77]. All the computation is performed on a laptop with 2.1 GHz Intel® Core™ processor. The optimal control solver takes around 2.6 s for a one-time initialization of automatic differentiation by CasADi. The optimization solver takes

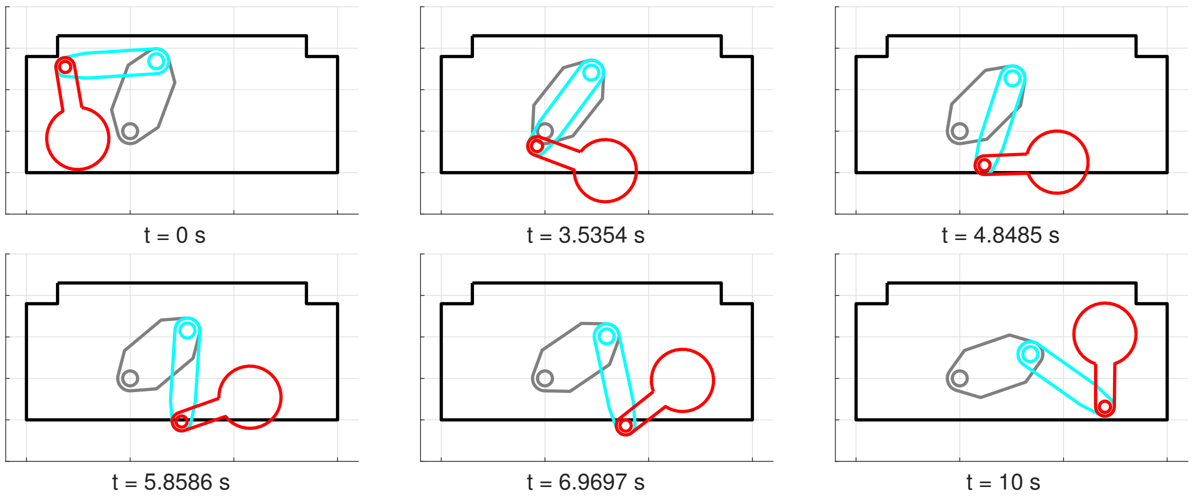


Figure 5.8: Initial trajectory for motion planning: planar test case 1

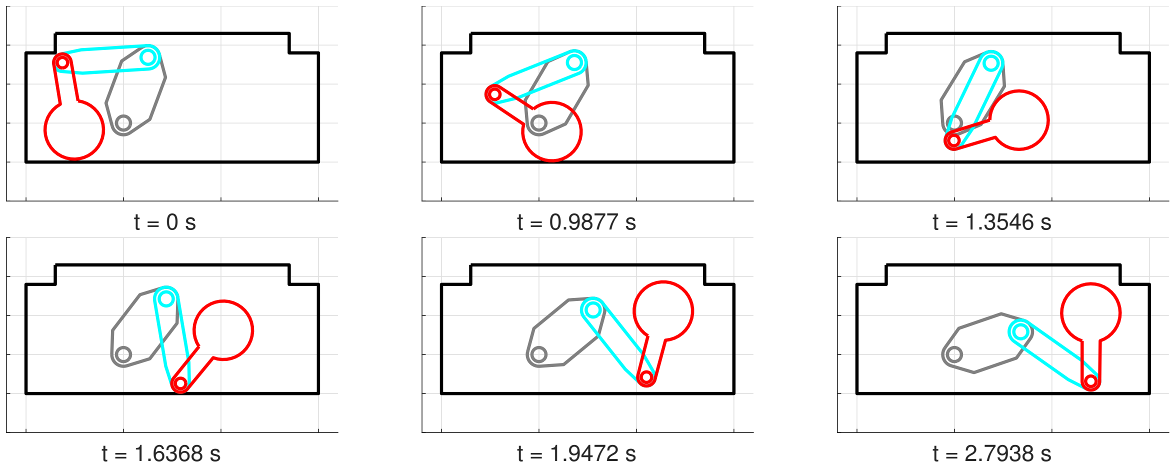


Figure 5.9: Optimal trajectory: planar test case 1

about 0.6 s to find the optimal trajectory. The returned path is shown in Fig. 5.9. The motion path has been adjusted automatically such that the there is no collision between the robot and the workspace boundary. The cycle time of robot motion has been reduced from 10 s to around 2.8 s. The planned joint velocity, acceleration, and workspace acceleration profiles are shown in Fig. 5.10, where $\ddot{x}^{wf}(t)$ and $\ddot{y}^{wf}(t)$ are the workspace acceleration of wafer center. The variables which are denoted as functions T_i are the variable values at discretized knots. The discretized values are interpolated to approximate the optimal solution for the continuous time problem. It is shown in the figures that joint velocity and acceleration are not getting close to the limits. The joint jerks are well bounded by the limits. The maximum allowed magnitude of workspace acceleration is 10 mg $\approx$ 1000 mm/s^2, which is not exceeded by the planned motion.

It is worth noting that the planned motion is highly smooth because high order polynomial

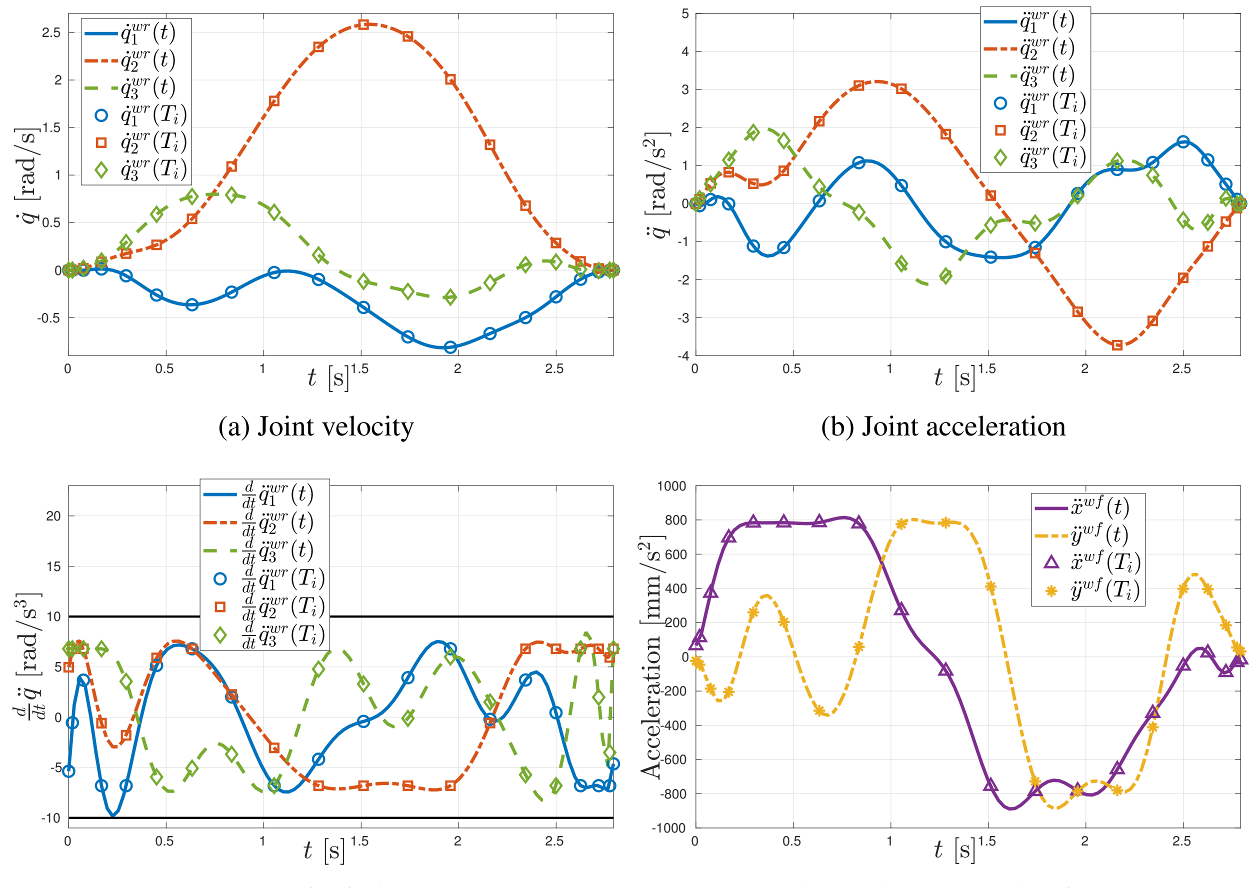


Figure 5.10: Optimized motion profiles for joint velocity, acceleration, jerk, and workspace acceleration, test case 1

interpolation is implemented. No obvious oscillation is observed in the interpolation. It is also worth noting that the polynomial interpolation does not necessarily satisfy constraints in between knots. This issue can be addressed by using more conservative constraints or more knots.

Test case 2 The target position is on the left portion in the workspace, while the initial position is the same as test case 1.

This is the same planning task as that in [76]. The optimization is initialized with $t_f = 10$. It takes around 2.6 s for a one-time initialization, and about 0.9 s to find the optimal trajectory. The returned path is shown in Fig. 5.11. The motion path has been adjusted automatically such that the there is no collision between the robot and the workspace boundary. The cycle time of robot motion has been reduced from 10 s to about 2.1 s. The planned joint velocity, acceleration, and workspace acceleration profiles are shown in Fig. 5.12, where $\ddot{x}^{wf}(t)$ and $\ddot{y}^{wf}(t)$ are the workspace acceleration of wafer center. The variables which are denoted as functions T_i are the variable values at discretized knots.

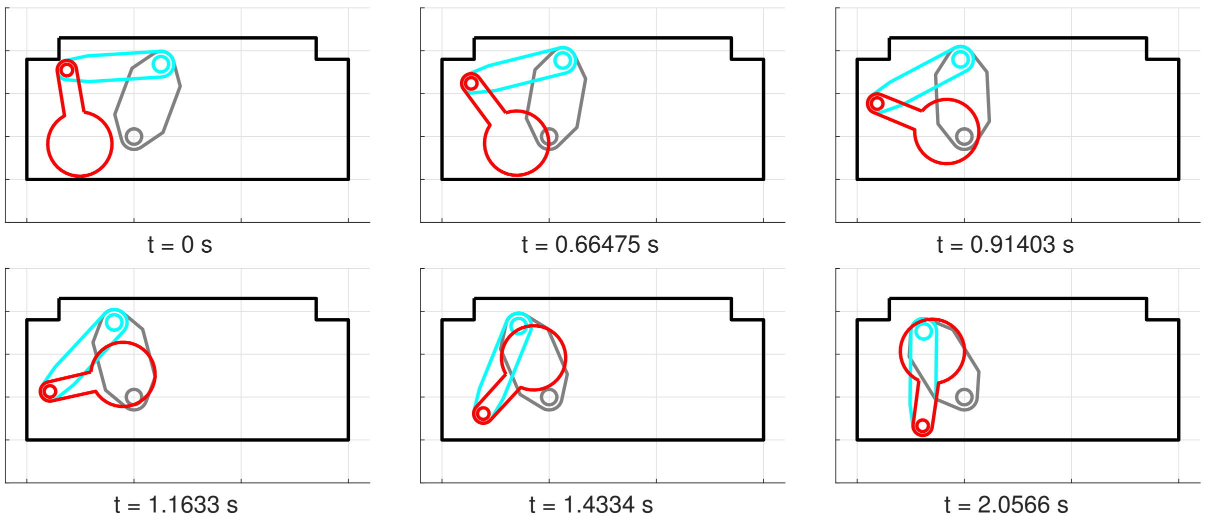
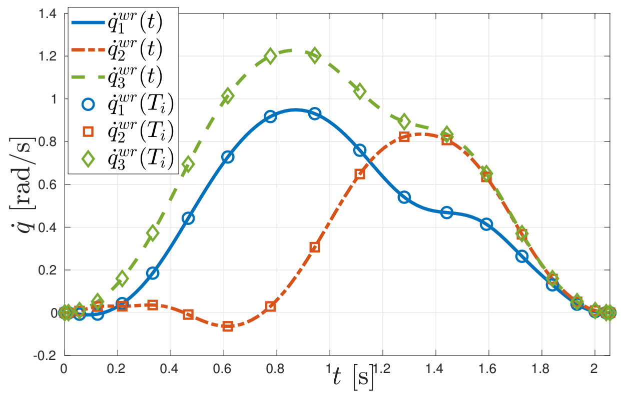


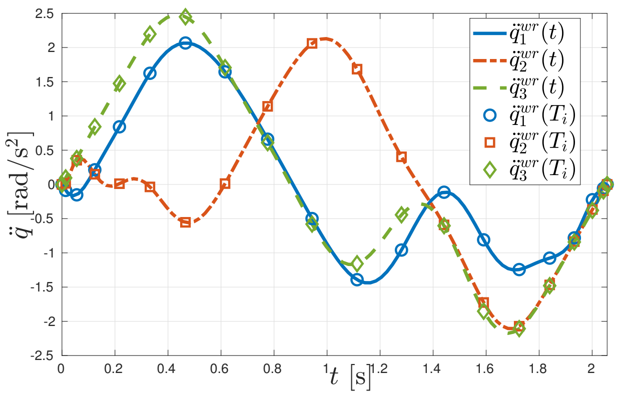
Figure 5.11: Optimal trajectory: planar test case 2

As clearly shown in the figures, the planned joint velocity, acceleration, jerk, and workspace acceleration are bounded by the design limits. In order to guarantee the interpolated motion does not exceed the actuator limit, the bounds for joint motion have been designed to be conservative.

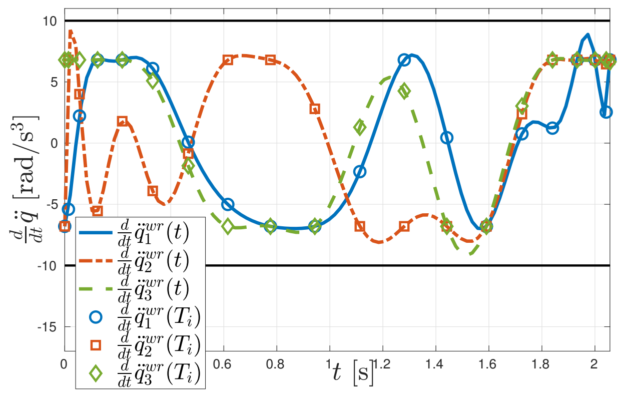
It is worth noting that the planned motion is actually faster than that obtained using path-velocity decomposition approach reported in [76], where the planned motion consists of several segments. Unlike the path-velocity decomposition approach, in which a sampling based planner could return non-smooth path that is hard to follow, trajectory optimization based approach is able to optimize the overall performance of path planning and trajectory planning. Therefore, the trajectory optimization based approach is able to find a better result than the path-velocity decomposition based approach.



(a) Joint velocity



(b) Joint acceleration


(c) Joint jerk

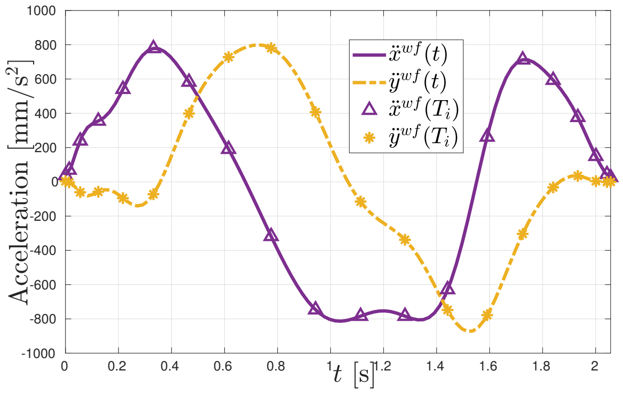

(d) Workspace acceleration

Figure 5.12: Optimized motion profiles for joint velocity, acceleration, jerk, and workspace acceleration, test case 2

Multiple Joint Industrial Robot Example

Time-optimal motion planning of a 6-axis industrial robot with dynamic constraints is considered in this example. The industrial robot is supposed to work in a constrained workspace for pick-and-place tasks. The geometric model of the industrial robot is shown in Fig. 5.13.

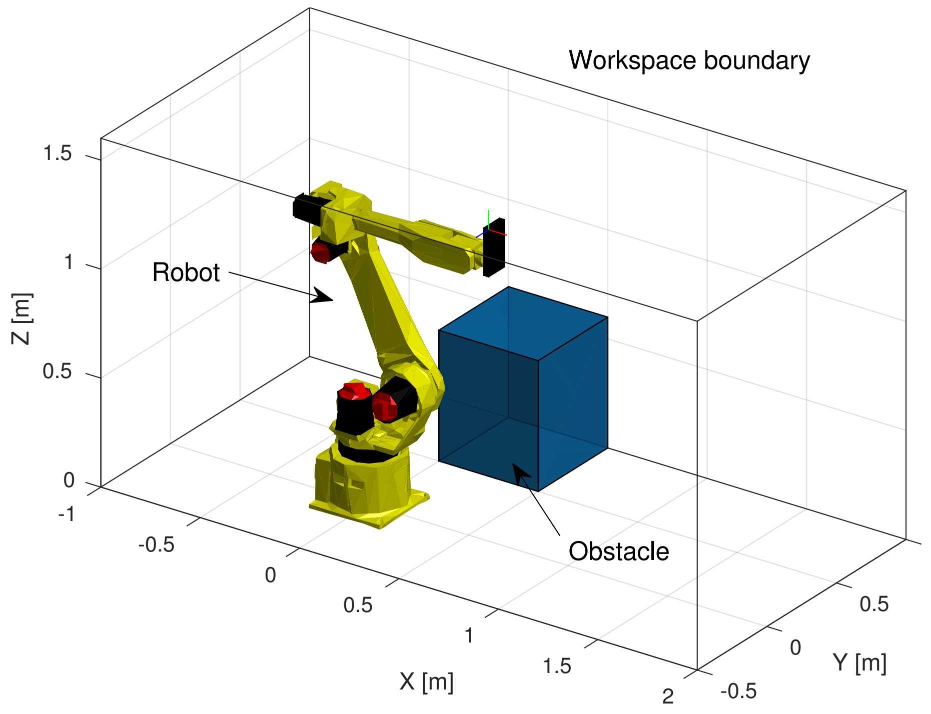


Figure 5.13: Geometric model of a 6 joint industrial robot

Unlike wafer handling robot, this type of robot does not come with strong actuators to support high speed motion. Therefore constraints coming from actuator including torque limits and even torque rate limits should be considered in the planning. Furthermore, kinematics constraints including joint position, velocity, workspace boundary and obstacle avoidance also apply to this industrial robot.

Let the joint positions be $q_1(t), \cdots, q_6(t)$, the joint torques be $\tau_1(t) \cdots, \tau_6(t)$, where $t \in [0, t_f]$ is time. The state and control of this dynamical system can be chosen to be

$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{q}(t) \\ \dot{\boldsymbol{q}}(t) \end{bmatrix}, \quad \boldsymbol{u}(t) = \boldsymbol{\tau}(t) \tag{5.37}$$

where $\boldsymbol{q}(t) = [q_1(t), \cdots, q_6(t)]^T$ is the vector for joint positions, and $\boldsymbol{\tau}(t) = [\tau_1(t), \cdots, \tau_6(t)]^T$ is the vector for joint torques. There are 12 states and 6 controls. In the motion planning work, the robot dynamics is simplified by supposing that 1) frictions are negligible, and 2) transmission is ideal. The simplified robot dynamics can be formulated as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{5.38a}$$

$$\boldsymbol{J}_m \boldsymbol{R} \ddot{\boldsymbol{q}} = \boldsymbol{\tau}_m - \boldsymbol{R}^{-1}\boldsymbol{\tau} \tag{5.38b}$$

where $\boldsymbol{J}_m$ is a diagonal matrix with $\boldsymbol{J}_m(i,i)$ the inertia of i the motor of the robot. $\boldsymbol{R}$ is a diagonal matrix with $\boldsymbol{R}(i,i)$ the gear ratio of the ith joint, and $\boldsymbol{R}(i,j) = 0$ if $i \neq j$. $\boldsymbol{\tau}_m$ is the motor torque.

The dynamic constraint of this task can be formulated as Equation (5.8) with $\boldsymbol{f}^{fric} = 0$.

The planning objective is to minimize motion time t_f. Considering that the motion of human arm can be promisingly modeled by minimum-jerk model [78], a regulation term of jerk integration can be included in the cost function for generating robot motion naturally. The cost function can be formulated as

$$J = t_f + \mu \int_0^{t_f} \dddot{\boldsymbol{q}}(t)^T \boldsymbol{Q} \dddot{\boldsymbol{q}}(t) \mathrm{d}t \tag{5.39}$$

where $\dddot{\boldsymbol{q}}(t)$ is the jerk of joint motion, μ is the weighting coefficient for regulation term, and $\boldsymbol{Q}$ is a weight matrix designed to penalize large motion of joints with high gear ratios. The weight matrix $\boldsymbol{Q}$ is defined as

$$\boldsymbol{Q}(i,j) = \begin{cases} 0, & j \neq i \\ 1/\boldsymbol{R}(i,i)^2, & j = i \end{cases} \tag{5.40}$$

Collision free condition is included in the path constraints. The robot links and obstacles are approximated by spheres for simple and differentiable collision detection for trajectory optimization. The sphere approximation of robot links, obstacles, and workspace boundary is shown in Fig. 5.14, where $\boldsymbol{c}_j$, $\boldsymbol{c}_k$, r_j, and r_k are centers and radius of the jth and kth spheres that approximate the robot links, and $\boldsymbol{c}_l^{obs}$ and r_l^{obs} present center and radius of the kth sphere that approximate the obstacle. The workspace boundary is represented by $[x_{min}^{bnd}, x_{max}^{bnd}]$, $[y_{min}^{bnd}, y_{max}^{bnd}]$, $[z_{min}^{bnd}, z_{max}^{bnd}]$ in X, Y, and Z direction respectively.

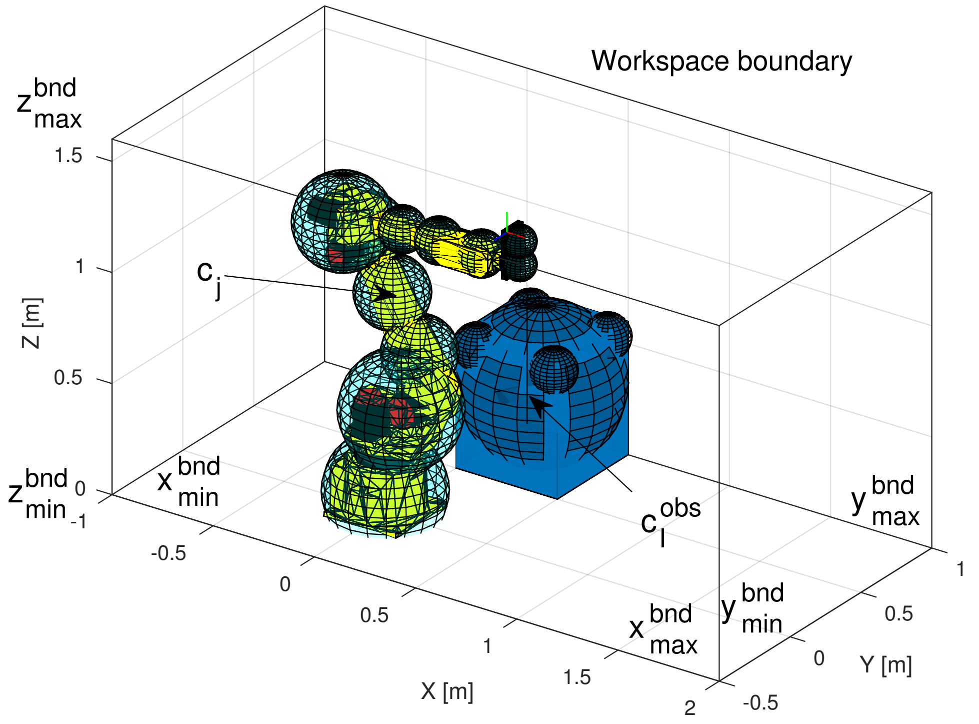


Figure 5.14: Sphere approximation of 6-axis robot and obstacle

The path constraints of this problem includes:

Joint position bounds:	$\boldsymbol{q}_{min} \leq \boldsymbol{q}(t) \leq \boldsymbol{q}_{max}$	(5.41a)
Joint velocity bounds:	$\dot{\boldsymbol{q}}_{min} \leq \dot{\boldsymbol{q}}(t) \leq \dot{\boldsymbol{q}}_{max}$	(5.41b)
Workspace bounds:	$\left[x^{bnd}_{min}, y^{bnd}_{min}, z^{bnd}_{min}\right]^T \leq \boldsymbol{c}_j(t) \leq \left[x^{bnd}_{max}, y^{bnd}_{max}, z^{bnd}_{max}\right]^T$	(5.41c)
Self collision avoidance:	$\lVert \boldsymbol{c}_j(\boldsymbol{q}(t)) - \boldsymbol{c}_k(\boldsymbol{q}(t)) \rVert_2 \geq r_j + r_k + \epsilon,\ [j,k] \in I$	(5.41d)
Obstacle collision avoidance:	$\lVert \boldsymbol{c}_j(\boldsymbol{q}(t)) - \boldsymbol{c}^{obs}_l \rVert_2 \geq r_j + r^{obs}_l + \epsilon$	(5.41e)

where I is a set of indices indicating possible collision between two balls that approximate robot links. Motor torque and torque rates should also be included in path constraints. According to Equation. (5.38b), torque and torque rate constraints can be formulated as

Motor torque bounds:	$\boldsymbol{R}\boldsymbol{\tau}_{m\,\mathrm{min}} \leq \boldsymbol{R}\boldsymbol{J}_m\boldsymbol{R}\ddot{\boldsymbol{q}}(t) + \boldsymbol{\tau}(t) \leq \boldsymbol{R}\boldsymbol{\tau}_{m\,\mathrm{max}}$	(5.42a)
Motor torque rate bounds:	$\boldsymbol{R}\dot{\boldsymbol{\tau}}_{m\,\mathrm{min}} \leq \boldsymbol{R}\boldsymbol{J}_m\boldsymbol{R}\dddot{\boldsymbol{q}}(t) + \dot{\boldsymbol{\tau}}(t) \leq \boldsymbol{R}\dot{\boldsymbol{\tau}}_{m\,\mathrm{max}}$	(5.42b)

The actuator limit of the industrial robot in this example is listed in Table 5.2, where joint positions have been manually tightened for safety reason. The scaled motor torque and torque rate limits are restricted to be 80% of the original value in the planning algorithm.

Table 5.2: Actuator limits of 6-joint industrial robot

Limits	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Joint position min $\boldsymbol{q}_{\min}$ (deg)	-160	-90	-120	-180	-120	-180
Joint position max $\boldsymbol{q}_{\max}$ (deg)	170	90	230	180	100	180
Joint speed $\dot{\boldsymbol{q}}$ (deg/s)	±165	±165	±175	±350	±340	±520
Scaled motor torque $\boldsymbol{R}\boldsymbol{\tau}_m$ (Nm)	±1396.5	±1402.3	±382.7	±45.2	±44.6	±32.5
Scaled torque rate $\boldsymbol{R}\dot{\boldsymbol{\tau}}_m$ (Nm/s)	±20947.5	±21034.5	±5740.5	±678	±669	±487.5

The boundary condition include initial and final position, velocity, acceleration, and terminal time bound.

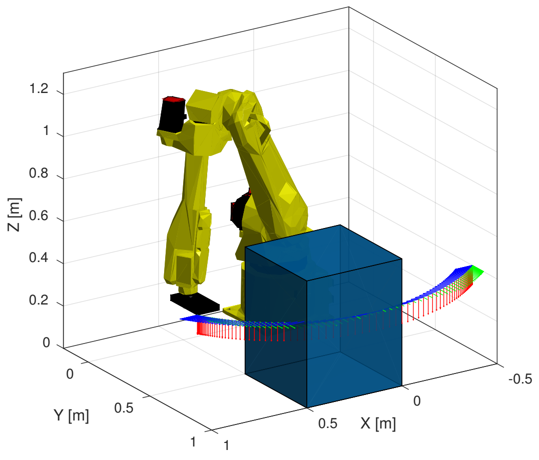
$$\text{Initial position, velocity, acceleration:} \quad \boldsymbol{q}(0) = \boldsymbol{q}^0, \dot{\boldsymbol{q}}(0) = 0, \ddot{\boldsymbol{q}}(0) = 0 \tag{5.43a}$$

$$\text{Target position, velocity, acceleration:} \quad \boldsymbol{q}(t_f) = \boldsymbol{q}^f, \dot{\boldsymbol{q}}(t_f) = 0, \ddot{\boldsymbol{q}}(t_f) = 0 \tag{5.43b}$$

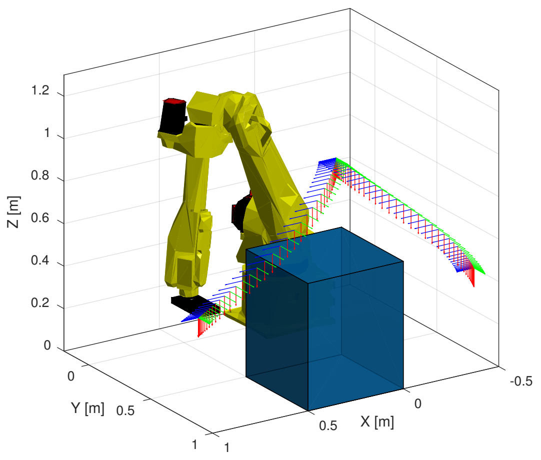
$$\text{Terminal time bounds :} \quad t_f^{min} \leq t_f \leq t_f^{max} \tag{5.43c}$$

The cost function, dynamic constraints, path constraints, and boundary conditions can be discretized $N+1$ knots, resulting an optimization problem with $12 \times (N+1) + 6 \times (N+1) + 1 = 18N + 19$ decision variables. Three factors will affect the solution to the optimization problems: N itself, the regulation weight μ, and the initialization of decision variables.

In general, the solution accuracy will be better if larger N is used, but the optimization will take longer time. The cycle time for robot motion t_f will be shorter if smaller μ is used, but the optimization will also take longer time. In regards to the initialization, two choices as shown in Fig. 5.15 can be designed: 1) a naive initialization: designing an straight line initial trajectory in



(a) Naive initialization



(b) Intuitive initialization

Figure 5.15: Initial trajectory for motion planning: multiple joint industrial robot

joint space without considering feasibility, 2) an intuitive initialization: adding one middle point to guide the robot off the obstacle, and composing the initial trajectory by combining two trajectories that connect initial-middle point, and middle-target point respectively.

The naive initialization requires more iterations to solve the optimization problem, and thus takes longer time. When the initial and target position is the same as Fig. 5.15, $N + 1 = 8$, $\mu = 0.3$, and intuitive initialization is used, the planned optimal motion is shown in Fig. 5.16. All

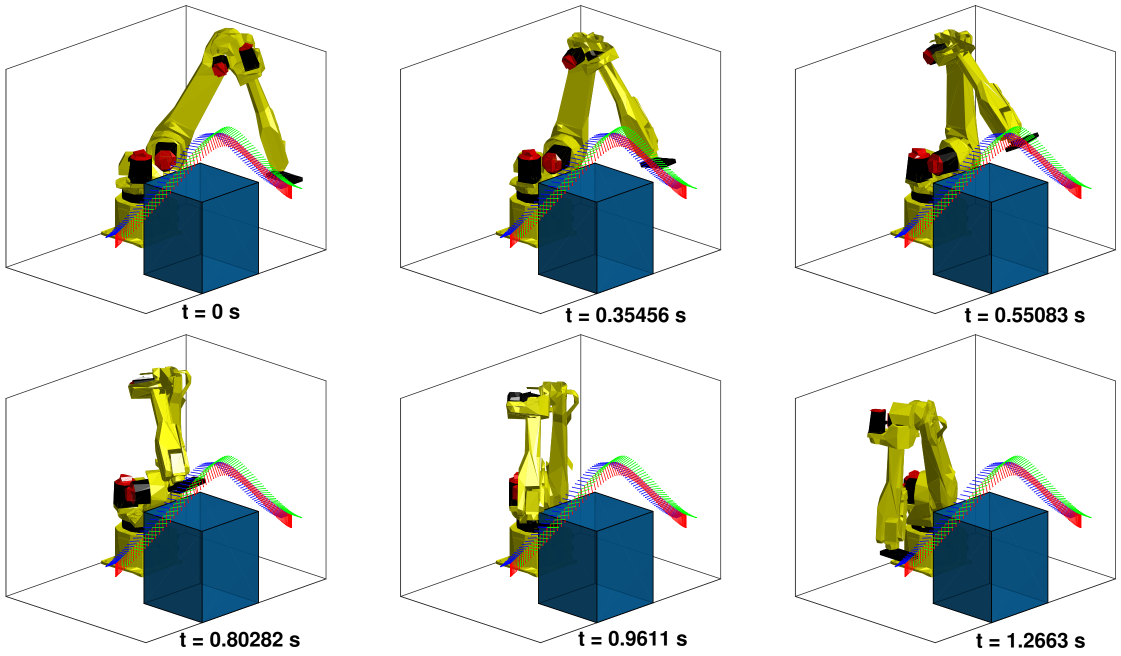
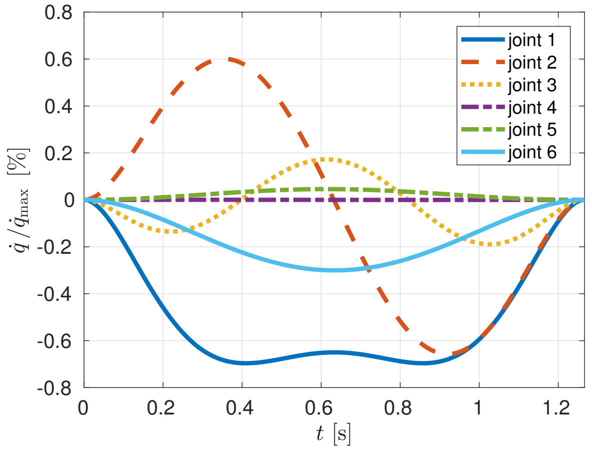


Figure 5.16: Planned optimal trajectory of multiple joint industrial robot, $N + 1 = 8$, $\mu = 0.3$

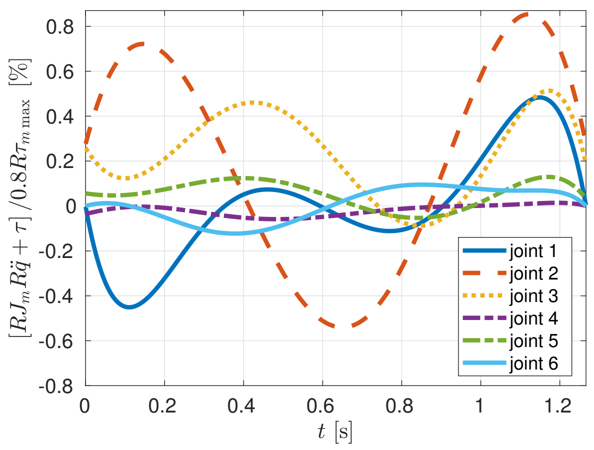
the computation is performed on the same hardware as that for planar robot example, i.e., a laptop with 2.1 GHz Intel® Core$^{\text{TM}}$ processor. The solver takes around 5.9 s for a one-time initialization for automatic differentiation by CasADi. The optimization takes around 1.2 s to find the optimal trajectory. The motion path has been automatically adjusted to guarantee smoothness, as well as avoiding self collision and collisions with workspace boundary or obstacle.

The planned motion takes around 1.27 s to finish while the initial trajectory takes 10 s. The percentage of the joint velocity and scaled motor torque are shown in Fig. 5.17. The actual optimal joint velocity and scaled motor torque are computed using larger N ($N + 1 = 35$) and longer time (computation time 415.8 s). The difference of the planned motion time t_f are close, which is 1.266 s for $N + 1 = 8$ and 1.251 s for $N + 1 = 35$. Both of the planning results indicate that the active limitation of the robot motion is the velocity limit of joint 1 and torque limit of joint 2. Comparing to the optimal result shown in Fig. 5.18, the planning result using $N + 1 = 8$ is a good approximation with low computational load.

Several tests have been performed to evaluate the effect of different μ, N, and initialization methods. The computation time for planning and planned motion time t_f are shown in Table 5.3,

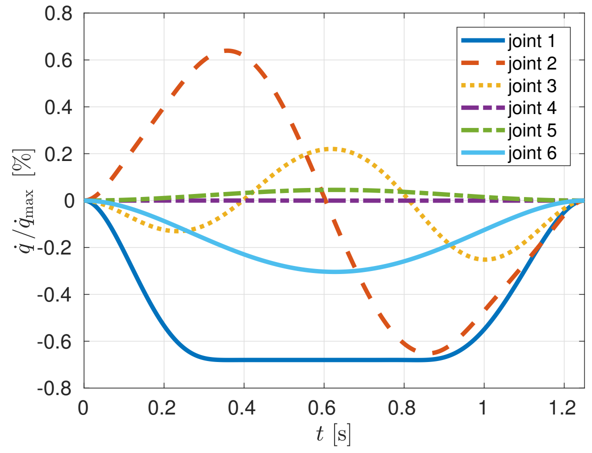


(a) Joint velocity

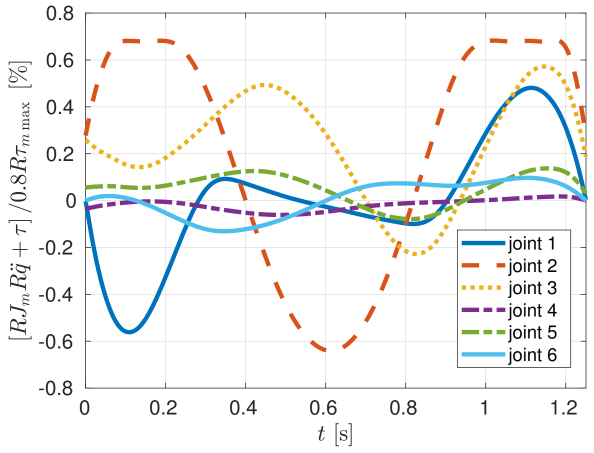


(b) Joint torque

Figure 5.17: Optimal motion profiles for multiple joint robot, $N + 1 = 8$, $\mu = 0.3$



(a) Joint velocity



(b) Joint torque

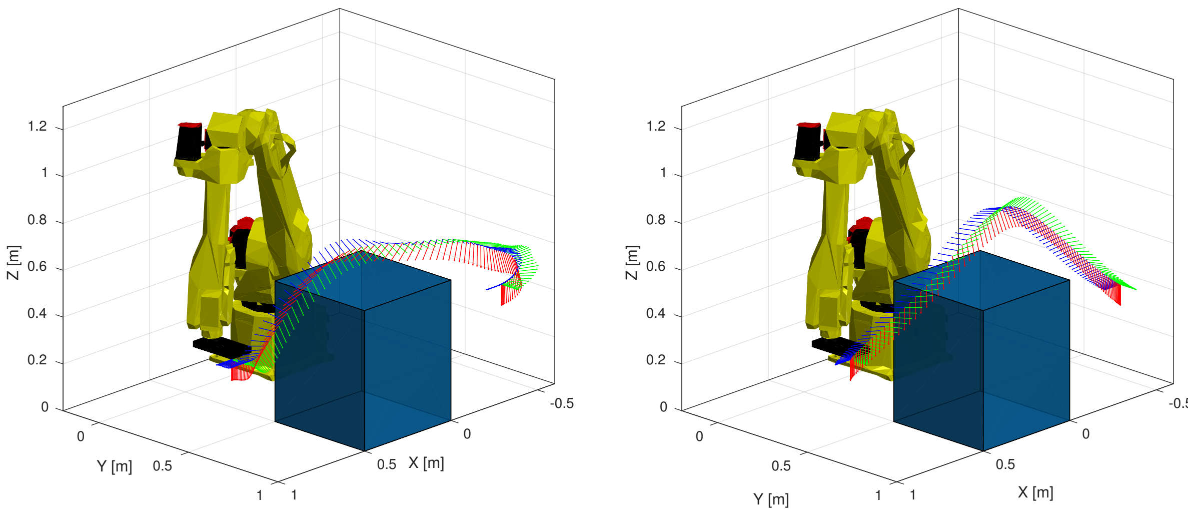
Figure 5.18: Optimal motion profiles for multiple joint robot, $N + 1 = 35$, $\mu = 0.3$

where torque and torque rate bounds are restricted to be 80% of the actual limits. It is clearly shown that: 1) increasing μ is helpful for shortening the computation time, but will result in slower planned motion, 2) shorter motion time t_f can be obtained by using larger N, but the computation takes longer time, and 3) naive initialization will result in longer computation time, and sometimes no solution can be returned. According to the result, $N + 1 = 12$ is a good choice since the planned motion time is reasonably close to the possible minimum value, and the computation time is acceptable. If even faster planning is desired, $N + 1 = 8$ can be considered. The intuitive initialization should always be used instead of naive initialization. It is also observed that shorter motion time can be obtained when $\mu = 0$, but the planned motion is unnatural, for example, the planned motions for $N+1=12, \mu=0$ and $N+1=12, \mu=0.3$ as shown in Fig. 5.19. In order to generate natural motion of robot arm, as well as reducing computation time for planning, a small regulation weight like $\mu=0.3$ can be chosen for the planning task.

Table 5.3: Computation time (s) / motion time t_f (s), with different parameters

	$\mu=0$		$\mu=0.3$		$\mu=10$	
N+1	NI[a]	II[b]	NI	II	NI	II
8	2.1 / 1.24	1.9 / 1.25	1.6 / 1.27	1.2 / 1.27	1.8 / 2.02	1.5 / 2.02
10	4.1 / 1.23	4.6 / 1.23	3.3 / 1.34	2.1 / 1.34	2.5 / 2.03	1.5 / 2.03
11	8.0 / 1.25	2.7 / 1.25	5.3 / 1.28	2.9 / 1.28	4.4 / 2.02	2.4 / 2.02
12	7.1 / 1.19	5.5 / 1.19	6.6 / 1.26	3.3 / 1.26	4.1 / 2.03	1.9 / 2.03
14	8.3 / 1.18	6.1 / 1.18	6.3 / 1.25	3.5 / 1.25	8.2 / 2.03	2.6 / 2.03
15	9.8 / 1.18	5.8 / 1.18	11.8 / 1.25	3.5 / 1.25	- / -	2.6 / 2.03
16	9.9 / 1.18	9.2 / 1.18	9.3 / 1.25	3.7 / 1.25	10.5 / 2.03	4.0 / 2.03
18	20.8 / 1.18	7.8 / 1.18	12.6 / 1.25	4.1 / 1.25	7.9 / 2.03	6.5 / 2.03
19	14.9 / 1.18	13.1 / 1.18	15.9 / 1.25	5.2 / 1.25	14.3 / 2.03	4.3 / 2.03

[a] Naive initialization.
[b] Intuitive initialization.

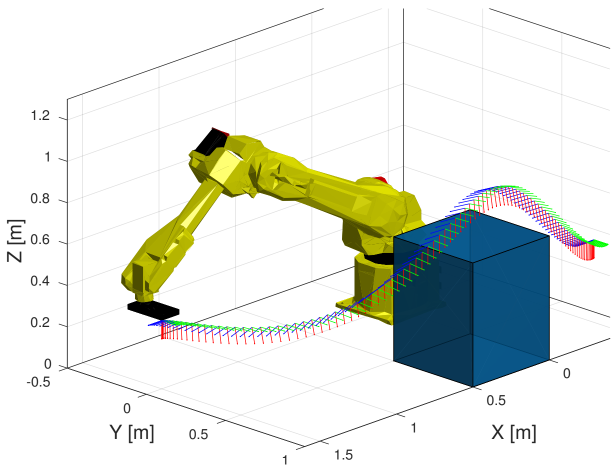


(a) $N+1=12, \mu=0$ (b) $N+1=12, \mu=0.3$

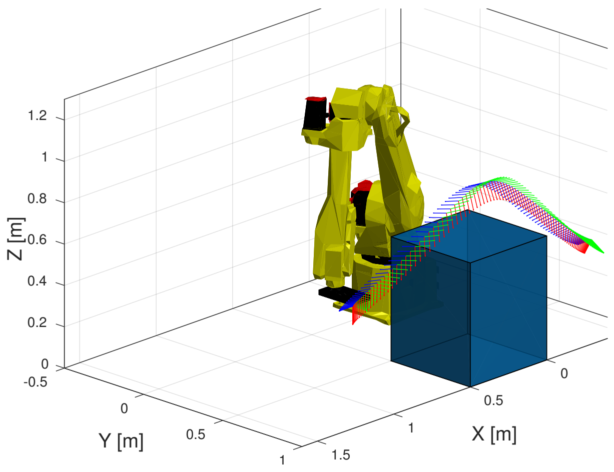
Figure 5.19: Optimal multiple joint robot trajectory with different planning parameters

The proposed planning algorithm has been tested using a group of different initial and target positions. The parameters are chosen to be $N+1=12$, and $\mu=0.3$, and intuitive initialization is implemented. Fig. 5.20 shows the planning results of optimal trajectories. In all the test cases, good approximations of the time optimal trajectories are returned in about 3 seconds by the proposed planning algorithm. It is also observed that the regulation term has prevented unnatural arm motion. Comparing to existing results for motion planning involving only robot dynamics [79, 65], which requires from 20 seconds to several minutes, the proposed approach is highly efficient.

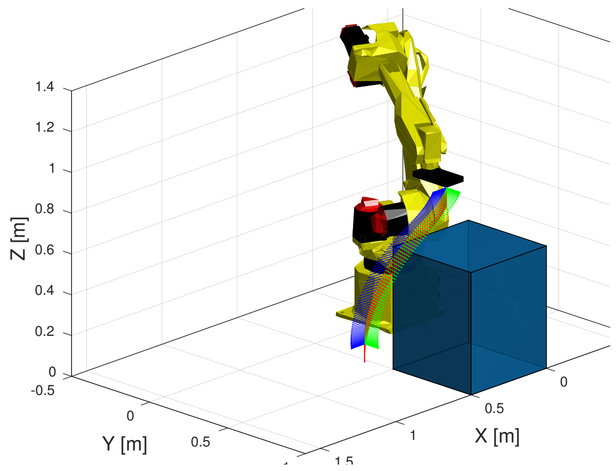
It is worth noting that the planned motion is a good approximation to the true optimal and feasible trajectory since there is no guarantee that no limitation is exceeded using only interpolation. A more conservative constraints or more knots can be used to guarantee safety. The planned trajectory should be checked before being executed. If the planned motion is infeasible, either tuning the constraints or adding more knots will be helpful for planning a feasible trajectory.



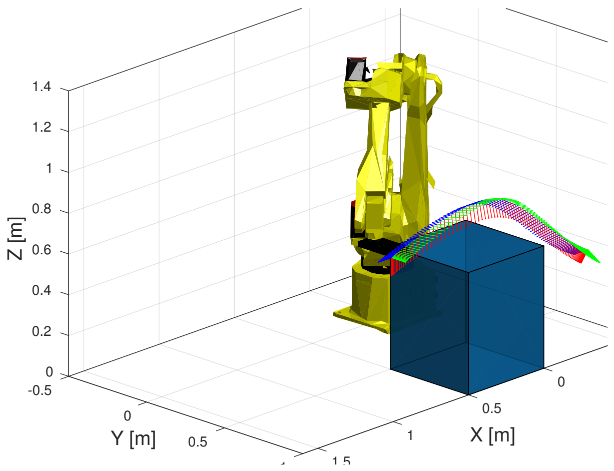
(a) Test case 1



(b) Test case 2



(c) Test case 3



(d) Test case 4

Figure 5.20: Optimal multiple joint robot trajectory with different initial and target positions, $N + 1 = 12$, $\mu = 0.3$

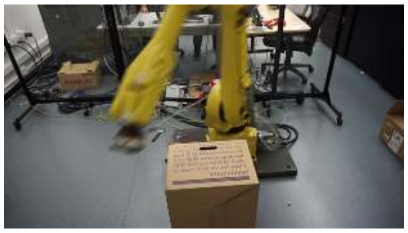
The planned optimal trajectory of test case 2 in Fig. 5.20 has been used as motion reference in experiment at the Mechanical Systems Control laboratory at the University of California, Berkeley.

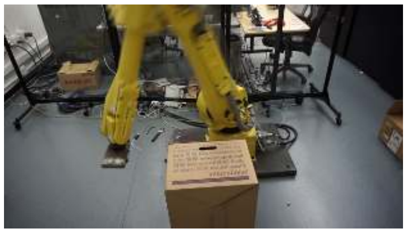
The actual robot motions are captured from video record as shown in Fig. 5.21. As shown in the figure, there is no collision between the robot and the obstacle in the workspace.

(a) t = 0 s

(b) t = 0.33 s

(c) t = 0.56 s


(d) t = 0.8 s


(e) t = 0.96 s

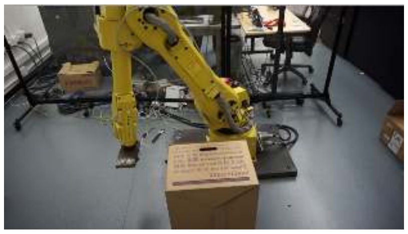
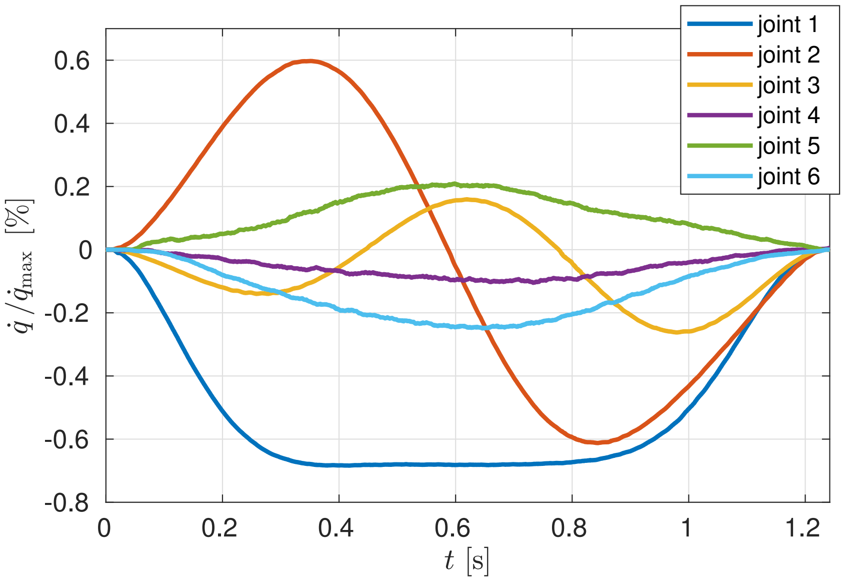
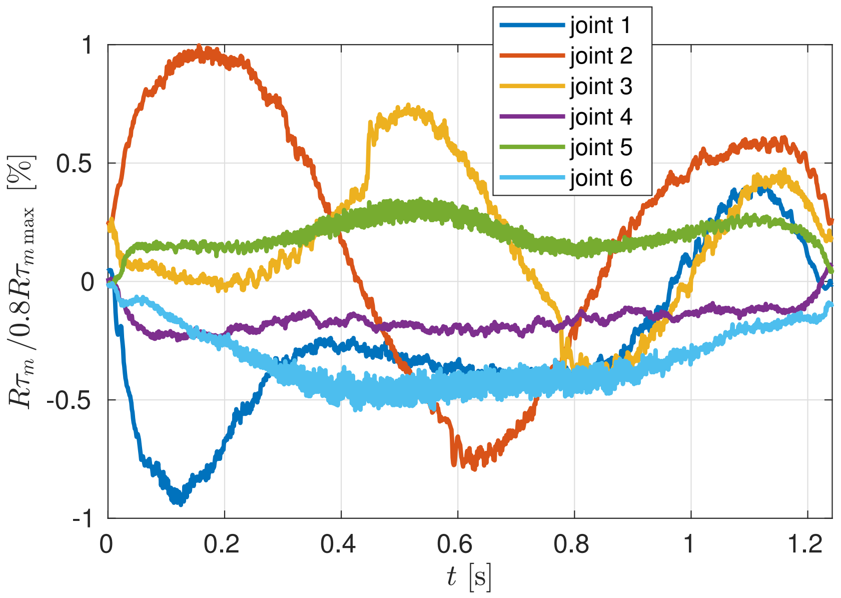

(f) t = 1.2 s

Figure 5.21: Actual robot motion in experiment

It is observed in the experiment data that the joint velocity and joint torque are all bounded by the actuator limit. The measured velocity and torque are scaled by the corresponding actuator limits, as shown in Fig. 5.22.



(a) Joint velocity



(b) Joint torque

Figure 5.22: Measured joint velocity and torque of optimal robot trajectory in experiment

5.5 Chapter Summary

Robot motion planning which involves complicated robot dynamics and geometric constraints is challenging. Since most approaches decompose motion planning to two subtopics and deal with them separately, only suboptimal solution can be found. In this chapter, motion planning is modeled as a general nonlinear optimal control problem to address the path planning and trajectory planning topics simultaneously. An efficient numerical method for trajectory optimization is proposed as one practical solution for the nonlinear optimal control problem. Results on a planar wafer handling robot and a 6-axis industrial robot have suggested that the proposed approach is able to find better solution than the current path-velocity decomposition approach. It is also found that the proposed approach can return a reasonably good solution within a short time. It is worth investigating improvements to this approach and exploring possibilities to implement it in different robotic applications.

Chapter 6

Optimal Vibration Suppression for an Industrial Robot with Flexible End-Effector

6.1 Introduction

Residual vibration suppression is important to achieve fast and precise motion in real-world industrial robot applications. However, it is challenging to guarantee low level residual vibration when robots are equipped with large articulated structures as end-effector for specific tasks. One practical example is robot equipped with large spot welding gun as shown in Fig. 6.1. The overall robotic

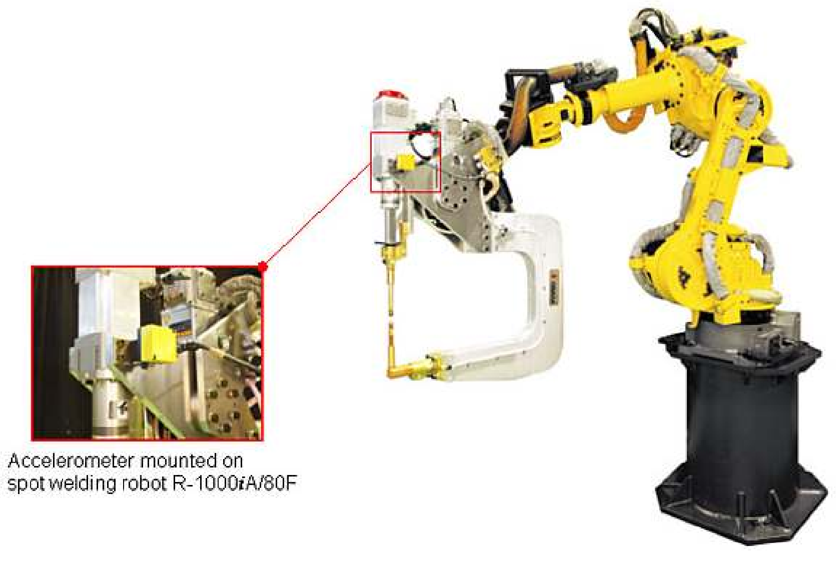


Figure 6.1: Industrial robot with large spot welding gun.

system becomes under-actuated because of the link flexibility introduced by slender design of end-effector. Since there is no external sensor available for the measurement of actual motion of robot end-effector, the residual vibration can not be simply controlled by feedback control using only joint actuators of industrial robot. One popular and promising model based feedforward control approach for this problem is input shaping. However the time delay introduced by input shaping is not allowed for time stringent applications unless intelligent modification can be implemented to guarantee zero time delay. Inspired by the success of input shaping, this chapter focuses on improving the performance of vibration suppression by fully taking advantage of the model information. Considering input shaping does not necessarily to be the best choice for robotic vibration suppression applications, an optimal control based vibration suppression is proposed based on the efficient trajectory optimization technique introduced in Chapter 5. Simulation and experiment results are presented to demonstrate the effectiveness of the proposed approach.

This chapter is organized as following: section 6.2 introduces a dynamical model of industrial robot with flexible end-effector, section 6.3 presents formulation of vibration suppression as optimal control problem, section 6.4 implements the proposed approach to working examples, and shows experimental results on an industrial robot equipped with experimental flexible end-effector, and section 6.5 summarizes this chapter.

6.2 Dynamical System Model

Industrial robots on the factory floor such as the one in Fig. 6.1 are not available in the Mechanical Systems Control laboratory at the University of California, Berkeley. A smaller industrial robot equipped with an experimental flexible payload in Fig. 6.2 is used for this dissertation research. The flexible payload is designed to have similar natural frequency and damping as the flexible end-effector of a larger industrial robot.

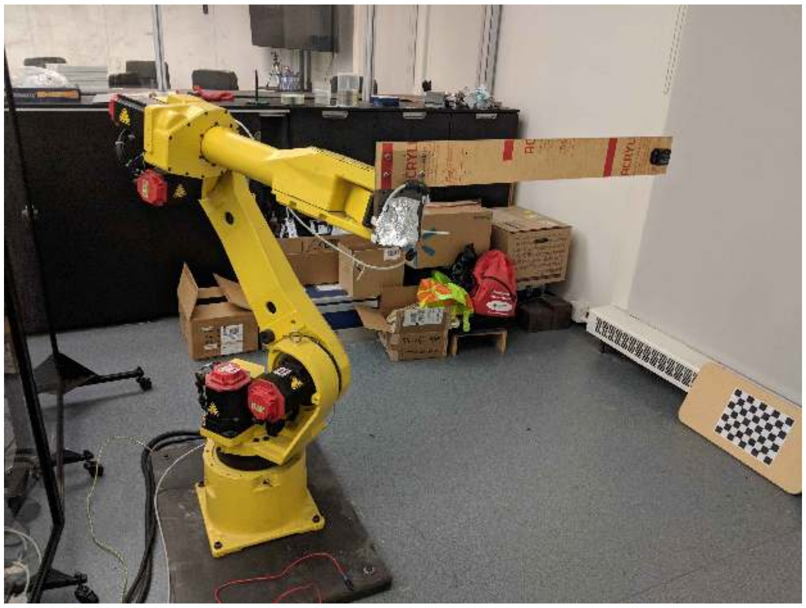


Figure 6.2: Industrial robot with experimental flexible payload.

The dynamical model of the system can be decomposed into two components: robot dynamics and end-effector dynamics.

Robot Dynamics

Though mechanical flexibility exists in the transmission units of industrial robot, the joint elasticity is negligible comparing to the link flexibility introduced by slender end-effectors. In this chapter, is is supposed that joint frictions are negligible and ideal joint transmission is applied. The robot dynamics can be simplified as rigid body dynamics. Letting the joint positions of a 6-joint robot be $\boldsymbol{q} = [q_1,\, q_2,\, q_3,\, q_4,\, q_5,\, q_6]^T$, the joint torque through transmission unit be $\boldsymbol{\tau} = [\tau_1,\, \tau_2,\, \tau_3,\, \tau_4,\, \tau_5,\, \tau_6]^T$, and the actuator torque be $\boldsymbol{\tau}_m = [\tau_{m1},\, \tau_{m2},\, \tau_{m3},\, \tau_{m4},\, \tau_{m5},\, \tau_{m6}]^T$, the simplified robot dynamics can be formulated as

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) &= \boldsymbol{\tau} \\ \boldsymbol{J}_m\boldsymbol{R}\ddot{\boldsymbol{q}} &= \boldsymbol{\tau}_m - \boldsymbol{R}^{-1}\boldsymbol{\tau} \end{aligned}$$

where $\boldsymbol{J}_m$ is a diagonal matrix with $\boldsymbol{J}_m(i,i)$ the inertia of i the motor of the robot. $\boldsymbol{R}$ is a diagonal matrix with $\boldsymbol{R}(i,i)$ the gear ratio of the ith joint, and $\boldsymbol{R}(i,j) = 0$ if $i \neq j$. The simplified robot dynamics can be formulated as one equation

$$[\boldsymbol{M}(q) + \boldsymbol{R}\boldsymbol{J}_m\boldsymbol{R}]\,\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{R}\boldsymbol{\tau}_m \tag{6.2}$$

Dynamics of Flexible End-Effector

The flexible end-effector is assumed to be simplified as a single mass-spring-damper system as shown in Fig. 6.3. Let

$$\boldsymbol{x} = \boldsymbol{f}_{fw}(\boldsymbol{q}) \tag{6.3}$$

be the Cartesian space position of the tool center point (the origin of the tool frame), where $\boldsymbol{f}_{fw}$ is the forward kinematics function of a robot with only rigid body parts. Let $\boldsymbol{r}$ be the tip elastic deformation of the end-effector, and $\boldsymbol{y}$ be the tip position of the end-effector, then $\boldsymbol{y}$ can be formulated as

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{r} \tag{6.4}$$

The mass-spring-damper system can be modeled as a second order system. The equation of motion can be formulated as

$$M_p\ddot{\boldsymbol{y}} = C_p(\dot{\boldsymbol{x}} - \dot{\boldsymbol{y}}) + K_p(\boldsymbol{x} - \boldsymbol{y}) + M_p\boldsymbol{g} + \boldsymbol{N} \tag{6.5}$$

where $\boldsymbol{g}$ is the gravitational acceleration, $\boldsymbol{N}$ is constraint force, M_p is the mass of payload, K_p is the spring stiffness, and C_p is the damping coefficient. Let $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ be unit vectors along X, Y, and Z directions in the tool frame, the following assumption is made in our experiment setup

$$\boldsymbol{r} = r_1\boldsymbol{i} + r_2\boldsymbol{j} + r_3\boldsymbol{k}, \quad r_2 = \boldsymbol{r}\cdot\boldsymbol{j} = 0, \quad r_3 = \boldsymbol{r}\cdot\boldsymbol{k} = 0 \tag{6.6}$$

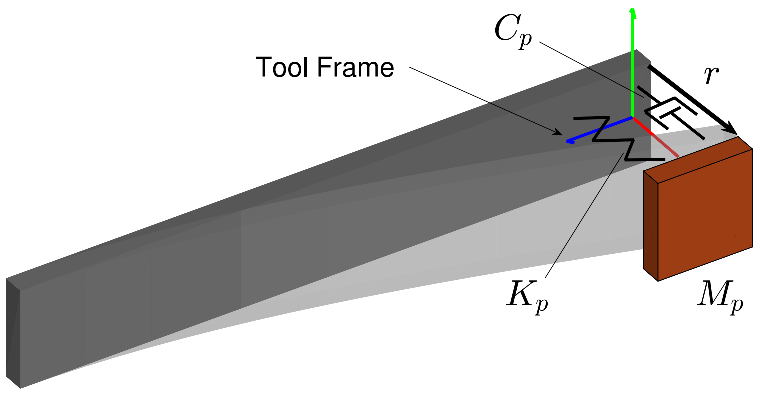


Figure 6.3: Simplification of flexible payload as mass-spring-damper system.

This assumption implies that the residual vibration only occurs in the X direction in the tool frame, while the deformation along Y and Z direction can be ignored due to the structural design. Projecting the equation of motion to the X direction in the tool frame, the dynamics of the flexible end-effector can be formulated as

$$M_p\ddot{\boldsymbol{y}}\cdot\boldsymbol{i} = C_p(\dot{\boldsymbol{x}}-\dot{\boldsymbol{y}})\cdot\boldsymbol{i} + K_p(\boldsymbol{x}-\boldsymbol{y})\cdot\boldsymbol{i} + M_p\boldsymbol{g}\cdot\boldsymbol{i} \tag{6.7}$$

where constraint force $\boldsymbol{N}$ is eliminated because no constraint force is applied in the motion direction. The time derivative of flexible deformation can be derived as

$$\boldsymbol{r} = \boldsymbol{y}-\boldsymbol{x} = r_1\boldsymbol{i} \tag{6.8a}$$
$$\dot{\boldsymbol{r}} = \dot{\boldsymbol{y}}-\dot{\boldsymbol{x}} = \dot{r}_1\boldsymbol{i} + r_1\boldsymbol{\omega}\times\boldsymbol{i} \tag{6.8b}$$
$$\ddot{\boldsymbol{r}} = \ddot{\boldsymbol{y}}-\ddot{\boldsymbol{x}} = \ddot{r}_1\boldsymbol{i} + 2\dot{r}_1\boldsymbol{\omega}\times\boldsymbol{i} + r_1\boldsymbol{\epsilon}\times\boldsymbol{i} + r_1\boldsymbol{\omega}\times(\boldsymbol{\omega}\times\boldsymbol{i}) \tag{6.8c}$$

where $\boldsymbol{\omega} = \omega_1\boldsymbol{i} + \omega_2\boldsymbol{j} + \omega_3\boldsymbol{k}$ is the angular velocity of the tool frame, and $\boldsymbol{\epsilon} = \dot{\boldsymbol{\omega}}$ is the angular acceleration of the tool frame.

Referring to Equation (6.8c), the tip acceleration is

$$\ddot{\boldsymbol{y}} = \ddot{\boldsymbol{x}} + \ddot{r}_1\boldsymbol{i} + 2\dot{r}_1\boldsymbol{\omega}\times\boldsymbol{i} + r_1\boldsymbol{\epsilon}\times\boldsymbol{i} + r_1\boldsymbol{\omega}\times(\boldsymbol{\omega}\times\boldsymbol{i}) \tag{6.9}$$

Using the properties of cross product, the X-axis component of the tip acceleration can be derived as

$$\begin{aligned}\ddot{\boldsymbol{y}}\cdot\boldsymbol{i} &= \ddot{\boldsymbol{x}}\cdot\boldsymbol{i} + \ddot{r}_1\boldsymbol{i}\cdot\boldsymbol{i} + 2\dot{r}_1(\boldsymbol{\omega}\times\boldsymbol{i})\cdot\boldsymbol{i}\\ &\quad + r_1(\boldsymbol{\epsilon}\times\boldsymbol{i})\cdot\boldsymbol{i} + r_1\left[\boldsymbol{\omega}\times(\boldsymbol{\omega}\times\boldsymbol{i})\right]\cdot\boldsymbol{i}\\ &= \ddot{\boldsymbol{x}}\cdot\boldsymbol{i} + \ddot{r}_1 + r_1\left[\boldsymbol{\omega}\times(\boldsymbol{\omega}\times\boldsymbol{i})\right]\cdot\boldsymbol{i}\\ &= \ddot{\boldsymbol{x}}\cdot\boldsymbol{i} + \ddot{r}_1 - r_1(\boldsymbol{\omega}\times\boldsymbol{i})\cdot(\boldsymbol{\omega}\times\boldsymbol{i})\end{aligned} \tag{6.10}$$

Since

$$\boldsymbol{\omega}\times\boldsymbol{i} = (\omega_1\boldsymbol{i} + \omega_2\boldsymbol{j} + \omega_3\boldsymbol{k})\times\boldsymbol{i} = -\omega_2\boldsymbol{k} + \omega_3\boldsymbol{j} \tag{6.11}$$

Letting $\ddot{\boldsymbol{x}} = \ddot{x}_1\boldsymbol{i} + \ddot{x}_2\boldsymbol{j} + \ddot{x}_3\boldsymbol{k}$, the X-axis component of the tip acceleration can be further derived as

$$\ddot{\boldsymbol{y}} \cdot \boldsymbol{i} = \ddot{x}_1 + \ddot{r}_1 - r_1(\omega_2^2 + \omega_3^2) \tag{6.12}$$

Similarly, there is

$$\begin{aligned} (\dot{\boldsymbol{x}} - \dot{\boldsymbol{y}}) \cdot \boldsymbol{i} &= -\dot{\boldsymbol{r}} \cdot \boldsymbol{i} \\ &= -(\dot{r}_1\boldsymbol{i} + r_1\boldsymbol{\omega} \times \boldsymbol{i}) \cdot \boldsymbol{i} \\ &= -\dot{r}_1 \end{aligned} \tag{6.13}$$

and

$$(\boldsymbol{x} - \boldsymbol{y}) \cdot \boldsymbol{i} = -\boldsymbol{r} \cdot \boldsymbol{i} = -r_1 \tag{6.14}$$

Substituting (6.12), (6.13), and (6.14) into Equation (6.7), the equation of motion of the flexible end-effector can be formulated as

$$M_p\ddot{x}_1 + M_p\ddot{r}_1 - M_p r_1\left(\omega_2^2 + \omega_3^2\right) = -C_p\dot{r}_1 - K_p r_1 + M_p\boldsymbol{g} \cdot i \tag{6.15}$$

or

$$\ddot{x}_1 + \ddot{r}_1 - r_1\left(\omega_2^2 + \omega_3^2\right) = -\frac{C_p}{M_p}\dot{r}_1 - \frac{K_p}{M_p}r_1 + \boldsymbol{g} \cdot i \tag{6.16}$$

Let

$$v_1 = \dot{r}_1$$

the dynamics of flexible end-effector can then be formulated as the state space model

$$\dot{r}_1 = v_1 \tag{6.17a}$$

$$\dot{v}_1 = -\frac{C_p}{M_p}v_1 - \frac{K_p}{M_p}r_1 + \left(\omega_2^2 + \omega_3^2\right)r_1 - \ddot{x}_1 + \boldsymbol{g} \cdot i \tag{6.17b}$$

6.3 Problem Formulation

The vibration suppression of flexible end-effector can be formulated as an optimal control problem and solved using trajectory optimization technique. One fundamental difference between optimal vibration suppression and optimal trajectory planning is that no time optimality is required in vibration suppression applications. Therefore the motion time t_f is fixed in the optimal control problem for vibration suppression.

In this work, it is assumed that $\boldsymbol{g} \cdot \boldsymbol{i} = 0$ at the beginning and end of the reference trajectory. Refer to Equation (6.17b), there will be no residual vibration if 1) $r_1(t_f) = 0$, 2) $\dot{r}_1(t_f) = 0$, and 3) $\ddot{x}_1(t_f) = 0$ are satisfied. However, the residual vibration is not guaranteed to be 100% eliminated by planning on a simplified dynamical model. In order to reduce residual vibration as much as possible, the residual elastic energy stored in the flexible end-effector should be minimized. Inspired by the integral of time-weighted absolute error (ITAE) and integral of time-weighted squared error (ITSE) criteria in feedback controller gain tuning works [80, 81, 82], the integral of time-weighted elastic energy (ITEE) can be used as the cost function of optimal control. The design of ITEE is expected to reduce residual elastic energy at the end of the robot motion, and thus reduce residual vibration level. The objective of the optimal control problem is

$$\min \qquad J = \int_0^{t_f} t\frac{1}{2}K_p r_1^2 \mathrm{d}t \tag{6.18}$$

The dynamic constraint of this problem can be formulated as the combination of robot dynamics Equation (6.2) and flexible end-effector dynamics Equation (6.17b)

$$\ddot{\boldsymbol{q}} = \left[\boldsymbol{M}(\boldsymbol{q}) + \boldsymbol{R}\boldsymbol{J}_m\boldsymbol{R}\right]^{-1}\left[\boldsymbol{R}\boldsymbol{\tau}_m - \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{G}(\boldsymbol{q})\right] \tag{6.19a}$$

$$\ddot{r}_1 = -\frac{C_p}{M_p}\dot{r}_1 - \frac{K_p}{M_p}r_1 + \left(\omega_2^2 + \omega_3^2\right)r_1 - \ddot{x}_1 + \boldsymbol{g}\cdot i \tag{6.19b}$$

with forward kinematics $\boldsymbol{x} = \boldsymbol{f}_{fw}(\boldsymbol{q})$ and $\ddot{x}_1 = \ddot{\boldsymbol{x}}\cdot\boldsymbol{i}$.

The path constraints of this problem include:

Joint position bounds:	$\boldsymbol{q}_{\min} \le \boldsymbol{q}(t) \le \boldsymbol{q}_{\max}$	(6.20a)
Joint velocity bounds:	$\dot{\boldsymbol{q}}_{\min} \le \dot{\boldsymbol{q}}(t) \le \dot{\boldsymbol{q}}_{\max}$	(6.20b)
Motor torque bounds:	$\boldsymbol{R}\boldsymbol{\tau}_{m\min} \le \boldsymbol{R}\boldsymbol{\tau}_m(t) \le \boldsymbol{R}\boldsymbol{\tau}_{m\max}$	(6.20c)
Motor torque rate bounds:	$\boldsymbol{R}\dot{\boldsymbol{\tau}}_{m\min} \le \boldsymbol{R}\dot{\boldsymbol{\tau}}_m(t) \le \boldsymbol{R}\dot{\boldsymbol{\tau}}_{m\max}$	(6.20d)

In addition, the planned robot motion should not deviate from the original motion too much. In this work, the desired robot motion is designed to be a straight line. The deviation constraint is formulated as angular velocity bounds and translational velocity bounds off the motion direction.

Angular velocity bounds:	$\|\boldsymbol{\omega}\|_{\min} \le \|\boldsymbol{\omega}(t)\| \le \|\boldsymbol{\omega}\|_{\max}$	(6.21a)
Translational velocity bounds:	$\dot{x}_{i\min} \le \dot{x}_i(t) \le \dot{x}_{i\max}, \quad i \in I_{off}$	(6.21b)

where I_{off} is the index set for directions other than motion direction (in this work the Y and Z directions, $I_{off} = \{2,3\}$).

The boundary condition include fixed initial and final position, velocity, acceleration, jerk, elastic deformation, and deformation rate. In addition, the final acceleration and jerk of the elastic deformation are also fixed to guarantee minimum residual vibration.

Initial position, velocity, acceleration, jerk:	$\boldsymbol{q}(0) = \boldsymbol{q}^0, \dot{\boldsymbol{q}}(0) = 0, \ddot{\boldsymbol{q}}(0) = 0, \dddot{\boldsymbol{q}}(0) = 0$	(6.22a)
Final position, velocity, acceleration, jerk:	$\boldsymbol{q}(t_f) = \boldsymbol{q}^f, \dot{\boldsymbol{q}}(t_f) = 0, \ddot{\boldsymbol{q}}(t_f) = 0, \dddot{\boldsymbol{q}}(t_f) = 0$	(6.22b)
Initial, final elastic deformation and rate:	$r_1(0) = 0, \dot{r}_1(0) = 0, r_1(t_f) = 0, \dot{r}_1(t_f) = 0$	(6.22c)
Final deformation acceleration and jerk:	$\ddot{r}_1(t_f) = 0, \dddot{r}_1(t_f) = 0$	(6.22d)

6.4 Numerical Examples and Experimental Results

In this work, the proposed optimal control based vibration suppression is implemented in the 6-joint industrial robot with flexible payload shown in Fig. 6.2. The actuator limits are shown in Table 5.2. As shown in Fig. 6.4, the reference trajectory is designed to be a straight line with fixed orientation.

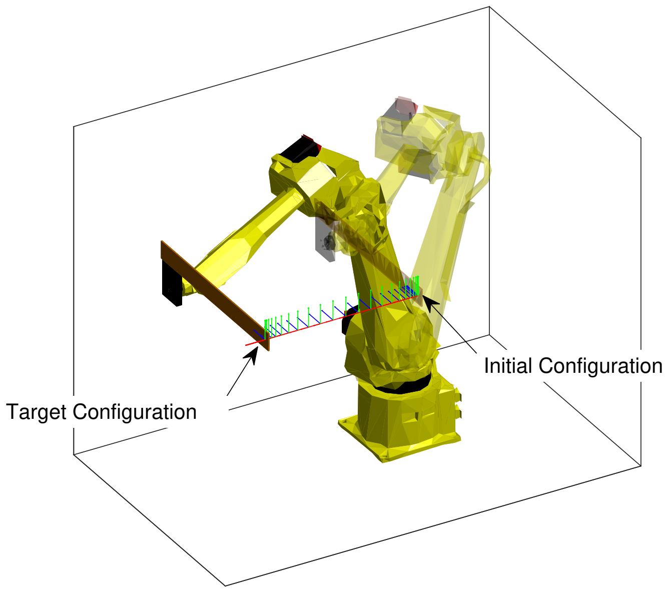


Figure 6.4: Reference trajectory of robot with flexible end-effector.

The reference trajectory has bounded velocity, acceleration, and jerk. The Cartesian space acceleration profile of the reference trajectory is shown in Fig. 6.5. The maximum acceleration is around 10 m/s^2, which is relatively high for the flexible payload. If no time delay is desired, input shaping based technique is guaranteed to generate non smooth motion. The proposed approach

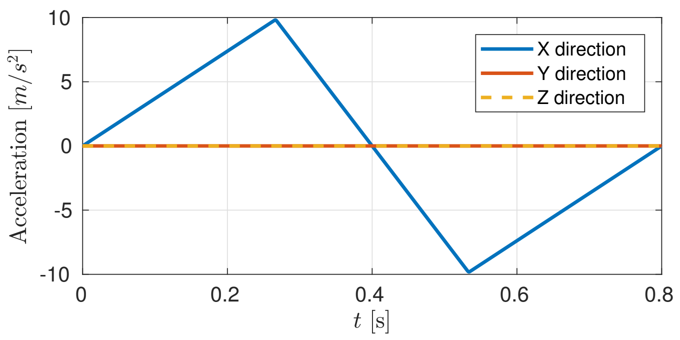


Figure 6.5: Cartesian space acceleration profile of reference trajectory.

approach is implemented for residual vibration suppression of this reference trajectory. $N+1 = 15$

knots are chosen to discretize the optimal control problem formulated in Sec. 6.3. The efficient trajectory optimization approach proposed in Chapter 5 takes 1.2 s to solve the problem. The optimized Cartesian space acceleration profile is show in Fig. 6.6. The maximum acceleration is reduced to around 7 m/s^2. The optimized trajectory is not a straight line anymore, but the deviation is negligible. The comparison of X direction acceleration from the original reference trajectory, trajectory adjusted by input shaping based approach, and optimized trajectory by the proposed approach is shown in Fig. 6.7.

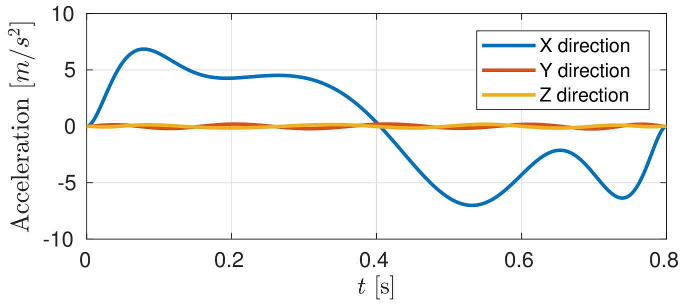


Figure 6.6: Optimized Cartesian space acceleration profile.

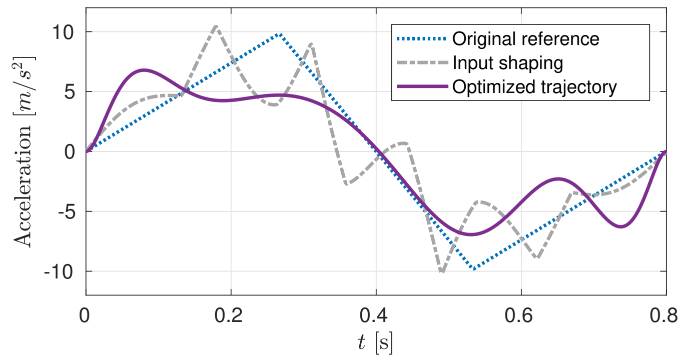


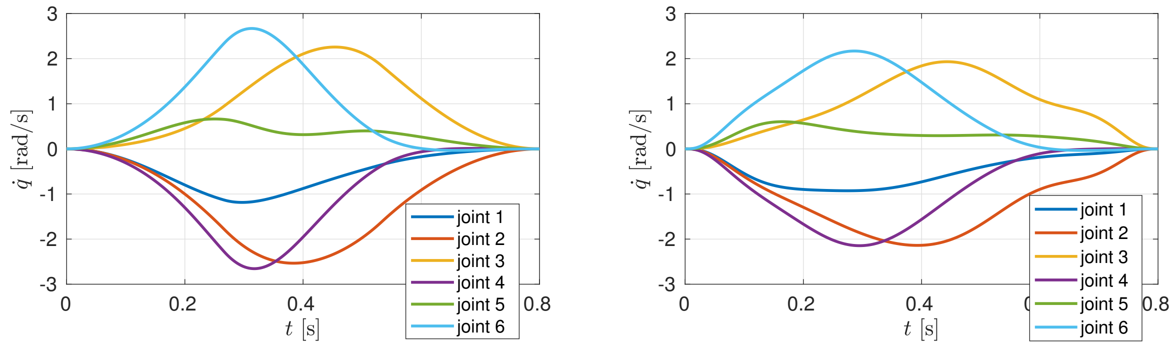
Figure 6.7: Comparison of X direction acceleration from different motion reference.

The joint velocity profile of reference trajectory and the optimized trajectory are shown in Fig. 6.8. Comparing to the original reference trajectory, the joint velocity distributes more even in the optimized trajectory, which reduces the peak magnitude of the velocity.

The optimized trajectory has been used as motion reference in both simulation and experiment.

Simulation

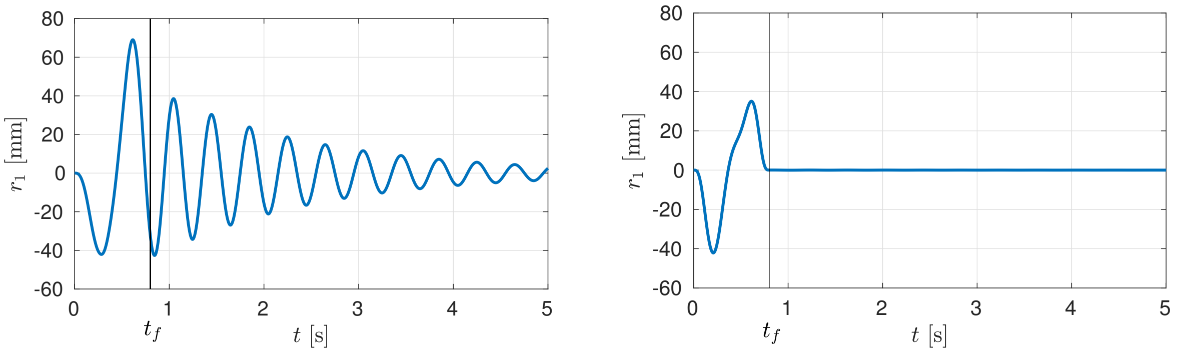
The response of the dynamical system is simulated using a high fidelity robot simulator which takes robot dynamics, flexible end-effector dynamics, and controller into consideration. When the original reference trajectory has been used as motion reference, the simulation shows long settling



(a) Reference trajectory (b) Optimized trajectory

Figure 6.8: Joint velocity profile of reference trajectory and optimized trajectory.

time before residual vibration disappears. There is almost no residual vibration observed when the optimized trajectory is used. The simulated elastic deformations using reference trajectory and optimized trajectory have been shown in Fig. 6.9. The simulated tip accelerations using reference trajectory and optimized trajectory are shown in Fig. 6.10.



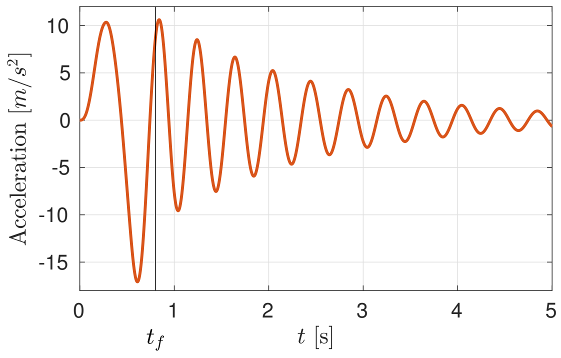
(a) Reference trajectory (b) Optimized trajectory

Figure 6.9: Simulated elastic deformation using different motion reference.

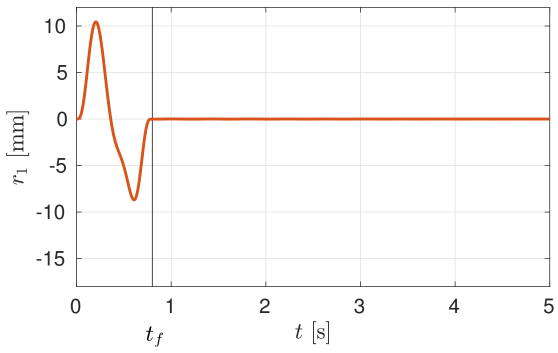
As shown in the figures, both the elastic deformation and the tip acceleration indicate that if no model uncertainty exists, the tip motion can be perfectly stopped at t_f.

Experimental Results

The optimized trajectory has been experimented on a 6-joint industrial robot shown in Fig. 6.2. A wireless accelerometer is mounted on the tip of the flexible payload to monitor the residual vibration. Obvious residual vibration is observed when the original reference trajectory is used as motion reference. The tip acceleration of residual vibration reached as large as about 10 m/s^2. The optimized trajectory has successfully reduced the residual tip acceleration to around 1 m/s^2. The



(a) Reference trajectory



(b) Optimized trajectory

Figure 6.10: Simulated tip acceleration using different motion reference.

desired tip acceleration, tip acceleration using original reference trajectory as motion reference, and tip acceleration using optimized trajectory as motion reference are shown in Fig. 6.11.

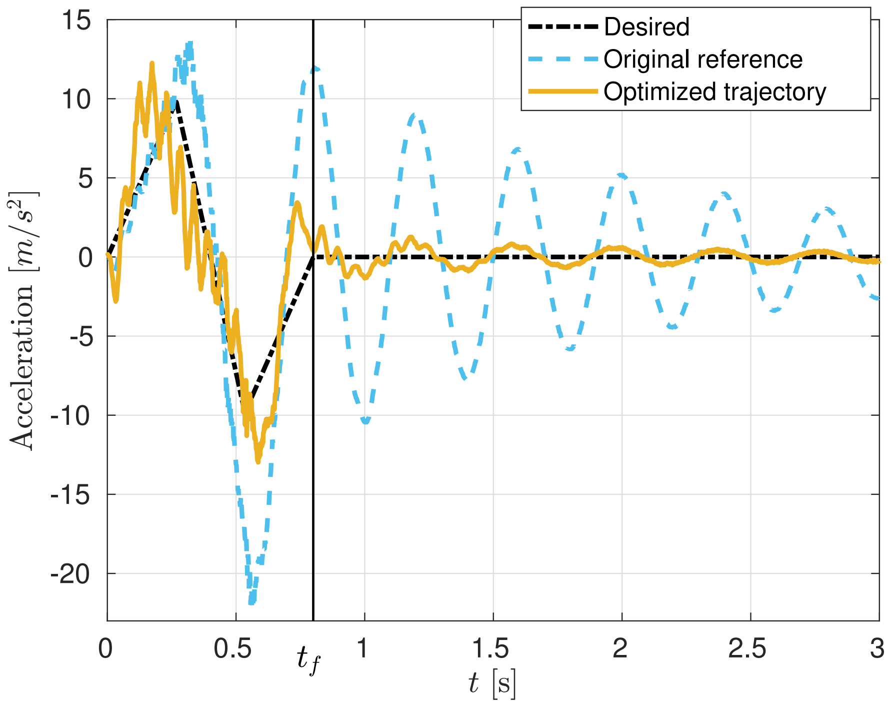
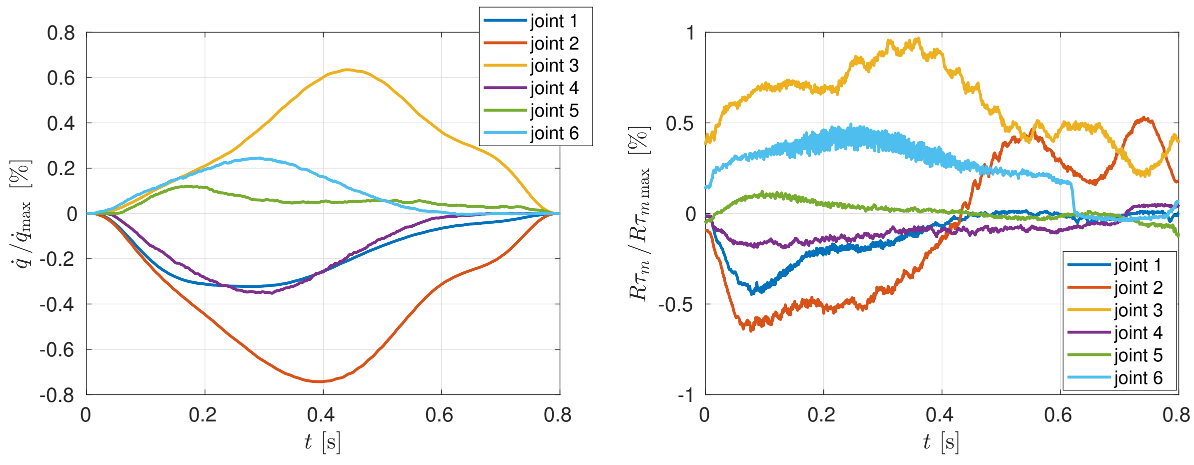


Figure 6.11: Measured tip acceleration from accelerometer.

It is observed in the experiment data that the joint velocity and joint torque are all bounded by the actuator limit. The measured velocity and torque are scaled by the corresponding actuator limits, as shown in Fig. 6.12.



(a) Joint velocity

(b) Joint torque

Figure 6.12: Measured joint velocity and torque in experiment.

It is also observed visually that the residual vibration is suppressed effectively. However due to model uncertainty and imperfect servo control, the residual vibration is not eliminated perfectly. Careful system identification and controller gain tuning would be helpful to further improve the performance of the proposed vibration suppression approach.

6.5 Chapter Summary

Residual vibration suppression is important to achieve fast and precise motion for robotic applications. In this chapter, an optimal control based approach for residual vibration suppression is proposed. The robot dynamics, a simplified flexible end-effector model, and the actuator limits are utilized to adjust the robot motion reference to guarantee smoothness, feasibility, and optimal vibration suppression performance. Trajectory optimization technique proposed in Chapter 5 is implemented to solve the optimal control problem for residual vibration suppression efficiently. The proposed approach has been applied in both simulation and experiment on a 6-joint industrial robot equipped with flexible payload. The results have demonstrated the effectiveness of the proposed approach. Future work of this topic includes improving model accuracy, necessary modifications for complicated trajectories with orientation changes, and issues related to practical deployment in robot controllers.

Chapter 7

Conclusion and Future Work

7.1 Concluding Remarks

This dissertation presented a series of works on performance improvement of existing industrial robots. These works are summarized as follows:

Intelligent Control

Chapter 2 has presented a neural network based adaptive backstepping controller for trajectory tracking of industrial robots with non-negligible transmission flexibility and model uncertainties. Theoretical analysis has guaranteed the stability of the backstepping controller for flexible joint robot without model uncertainty. To address model uncertainties in the system, a neural network was introduced for on-line adaptive compensation. The stability of the proposed controller was established using Lyapunov stability theory. Simulation work on a 6-joint industrial robot demonstrated the effectiveness of the proposed approach.

Chapter 3 presented a zero time delay input shaping approach for smooth settling of industrial robot with joint flexibility. The time delay issue in conventional input shaping technique is addressed by the proposed approach. Path constraint was taken into consideration for implementation on multi-joint industrial robot. Experimental results on a 6-joint industrial robot demonstrated the performance improvement in terms of overshoot reduction.

Chapter 4 presented a modified zero time delay input shaping approach for residual vibration suppression of industrial robot with flexible end-effector. The proposed approach focused on the improvement of zero time delay input shaping technique by increasing motion smoothness. Based on an analysis of conventional input shaping and zero time delay input shaping using convolution representation, a model based compensator was introduced to reduce the non-smoothness of zero time delay input shaping. Experimental results on an industrial robot equipped with flexible payload showed the effectiveness of the proposed approach.

Intelligent Planning

Chapter 5 presented an optimal control based approach for robot motion planning. The proposed approach solved path planning and trajectory planning problems simultaneously. As one practical solution for the general nonlinear optimal control problem introduced for motion planning, an efficient numerical method for trajectory optimization is presented. Simulations and experimental results showed that the proposed approach can return a reasonably good solution within a short time.

Chapter 6 presented an optimal control based approach for residual vibration suppression of industrial robot with flexible end-effector. The proposed approach takes full advantage of existing dynamical model of industrial robot and flexible end-effector. A smooth motion reference was generated by the efficient trajectory optimization technique presented in Chapter 5. Actuator limitations were taken into consideration to guarantee feasibility of the generated motion reference. The effectiveness of the proposed approach was shown by simulations and experimental results on a 6-joint industrial robot equipped with flexible payload.

7.2 Future Work

Several open issues arise during the course of this research. Addressing these issues may be future directions of this research.

Full State Sensing

Joint flexibility is a practical issue in industrial robots with a heavy payload. The research work in Chapter 2 has assumed full access to the robot's states, including actuator states and robot link states. However most existing industrial robots do not have sensors for robot link position, which is essential for high accuracy trajectory tracking control. In order to address this issue, the development of robot link state sensing is necessary. One possible direction is to investigate hardware design like the DLR lightweight robot [83] and integrated robot joint encoder mentioned in [2], another possible direction is to utilize extra low cost sensors and investigate model based state observer [84].

Efficient Collision Checking

The efficient trajectory optimization approach in Chapter 5 is highly efficient in dealing with motion planning problems which only involve robot dynamics. It is found that the most time consuming part in the robot motion planning work is collision checking using a sphere approximation. The reason is that the number of collision-free constraints grows quadratically as the number of spheres used for approximation increases. Efficient collision checking which uses less geometric elements (e.g. [85]) can be investigated to improve the efficiency of motion planning.

System Identification

Most existing industrial robots are not equipped with sensors for the deformation measurement of flexible end-effectors, and thus no feedback control can be used for the residual vibration suppression. In order to guarantee the optimal control based vibration suppression in Chapter 6 works well for arbitrary trajectory, it is necessary to fit a good dynamical model of the mechanical system. The identification task is challenging because of limited observations, noisy measurement data, and unmodeled dynamics. Identification approaches based on Bayesian theory is one possible direction to achieve good estimation of physical parameters. It is also interesting to consider on-line identification approaches based on dual-estimation.

Bibliography

[1] *Robots and robotic devices —Vocabulary*. ISO 8373:2012(en). International Organization for Standardization, Geneva, Switzerland, second edition, 2012.

[2] Torgny Brogårdh. Robot control overview: An industrial perspective. *Modeling, Identification and Control*, 30(3):167, 2009.

[3] Erik Wernholt. *Multivariable frequency-domain identification of industrial robots*. PhD thesis, Institutionen för systemteknik, 2007.

[4] Alessandro De Luca and Wayne Book. Robots with flexible elements. In *Springer Handbook of Robotics*, pages 287–319. Springer, 2008.

[5] Grzegorz Litak and Michael I Friswell. Vibration in gear systems. *Chaos, Solitons & Fractals*, 16(5):795–800, 2003.

[6] Alessandro De Luca and Pasquale Lucibello. A general algorithm for dynamic feedback linearization of robots with elastic joints. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 1, pages 504–510. IEEE, 1998.

[7] Mark W. Spong, Khashayar Khorasani, and Petar V. Kokotovic. An integral manifold approach to the feedback control of flexible joint robots. *IEEE Journal on Robotics and Automation*, 3(4):291–300, 1987.

[8] Alessandro De Luca. Feedforward/feedback laws for the control of flexible robots. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 233–240. IEEE, 2000.

[9] Wenjie Chen and Masayoshi Tomizuka. Dual-stage iterative learning control for mimo mismatched system with application to robots with joint elasticity. *Control Systems Technology, IEEE Transactions on*, 22(4):1350–1361, 2014.

[10] Cong Wang, Yu Zhao, Yubei Chen, and Masayoshi Tomizuka. Nonparametric statistical learning control of robot manipulators for trajectory or contour tracking. *Robotics and Computer-Integrated Manufacturing*, 35:96–103, 2015.

[11] Mark W. Spong. Modeling and control of elastic joint robots. *Journal of dynamic systems, measurement, and control*, 109(4):310–318, 1987.

[12] Carlos Canudas de Wit, Bruno Siciliano, and Georges Bastin. *Theory of robot control*. Springer Science & Business Media, 2012.

[13] Prasanna Subhash Gandhi. *Modeling and control of nonlinear transmission attributes in harmonic drive systems*. PhD thesis, Rice University, 2001.

[14] Cheng-Huei Han, Chun-Chih Wang, and Masayoshi Tomizuka. Suppression of vibration due to transmission error of harmonic drives using peak filter with acceleration feedback. In *Advanced Motion Control, 2008. AMC'08. 10th IEEE International Workshop on*, pages 182–187, 2008.

[15] Petar V. Kokotovic. The joy of feedback: nonlinear and adaptive. *IEEE Control systems*, 12(3):7–17, 1992.

[16] Frank L. Lewis, Suresh Jagannathan, and Aydin Yesildirak. *Neural network control of robot manipulators and nonlinear systems*. CRC Press, 1998.

[17] Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

[18] Bernd Fritzke. Fast learning with incremental rbf networks. *Neural processing letters*, 1(1):2–5, 1994.

[19] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[20] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[21] Calafiore Giuseppe and El Ghaoui Laurent. *Optimization Models*. Control systems and optimization series. Cambridge University Press, October 2014.

[22] H.K. Khalil. *Nonlinear Systems*, pages 168–174. Prentice Hall, 2002.

[23] Santosha Kumar Dwivedy and Peter Eberhard. Dynamic analysis of flexible manipulators, a literature review. *Mechanism and machine theory*, 41(7):749–777, 2006.

[24] Bruno Siciliano and Wayne J. Book. A singular perturbation approach to control of lightweight flexible manipulators. *The International Journal of Robotics Research*, 7(4):79–90, 1988.

[25] Bidyadhar Subudhi and Alan S. Morris. Singular perturbation approach to trajectory tracking of flexible robot with joint elasticity. *International Journal of Systems Science*, 34(3):167–179, 2003.

[26] Atsushi Konno, Liu Deman, and Masaru Uchiyama. A singularly perturbed method for pole assignment control of a flexible manipulator. *Robotica*, 20(6):637–651, 2002.

[27] Miguel A Serna and Eduardo Bayo. Trajectory planning for flexible manipulators. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 910–915. IEEE, 1990.

[28] Akira Mohri, Pritam Kumar Sarkar, and Motoji Yamamoto. An efficient motion planning of flexible manipulator along specified path. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1104–1109. IEEE, 1998.

[29] Mouhacine Benosman, Georges Le Vey, Leonardo Lanari, and Alessandro De Luca. Rest-to-rest motion for planar multi-link flexible manipulator through backward recursion. *Journal of dynamic systems, measurement, and control*, 126(1):115–123, 2004.

[30] Neil C Singer and Warren P Seering. Preshaping command inputs to reduce system vibration. *Journal of Dynamic Systems, Measurement, and Control*, 112(1):76–82, 1990.

[31] Tarunraj Singh and William Singhose. Input shaping/time delay control of maneuvering flexible structures. In *American Control Conference, 2002. Proceedings of the 2002*, volume 3, pages 1717–1731. IEEE, 2002.

[32] Amine Kamel, Friedrich Lange, and Gerd Hirzinger. New aspects of input shaping control to damp oscillations of a compliant force sensor. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2629–2635. IEEE, 2008.

[33] William Singhose. Command shaping for flexible systems: A review of the first 50 years. *International Journal of Precision Engineering and Manufacturing*, 10(4):153–168, 2009.

[34] Riccardo Marino and Mark W. Spong. Nonlinear control techniques for flexible joint manipulators: a single link case study. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1030–1036. IEEE, 1986.

[35] Alessandro De Luca and Bruno Siciliano. Inversion-based nonlinear control of robot arms with flexible links. *Journal of guidance, control, and dynamics*, 16(6):1169–1176, 1993.

[36] Fathi Ghorbel, John Y Hung, and Mark W Spong. Adaptive control of flexible-joint manipulators. *Control Systems Magazine, IEEE*, 9(7):9–13, 1989.

[37] Jung Hua Yang, Feng Li Lian, and Li Chen Fu. Nonlinear adaptive control for flexible-link manipulators. *Robotics and Automation, IEEE Transactions on*, 13(1):140–148, 1997.

[38] Alessandro De Luca and Giovanni Ulivi. Iterative learning control of robots with elastic joints. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 1920–1926. IEEE, 1992.

[39] Wayne J Book. Controlled motion in an elastic world. *Journal of dynamic systems, measurement, and control*, 115(2B):252–261, 1993.

[40] William E Singhose, Warren P Seering, and Neil C Singer. Input shaping for vibration reduction with specified insensitivity to modeling errors. *Japan-USA Sym. on Flexible Automation*, 1:307–13, 1996.

[41] Youmin Hu, Bo Wu, John Vaughan, and William Singhose. Oscillation suppressing for an energy efficient bridge crane using input shaping. In *Control Conference (ASCC), 2013 9th Asian*, pages 1–5. IEEE, 2013.

[42] Frank Boeren, Dennis Bruijnen, Niels van Dijk, and Tom Oomen. Joint input shaping and feedforward for point-to-point motion: Automated tuning for an industrial nanopositioning system. *Mechatronics*, 24(6):572–581, 2014.

[43] John Vaughan, Jieun Yoo, Nicholas Knight, and William Singhose. Multi-input shaping control for multi-hoist cranes. In *American Control Conference (ACC), 2013*, pages 3449–3454. IEEE, 2013.

[44] Joshua Vaughan, Aika Yano, and William Singhose. Comparison of robust input shapers. *Journal of Sound and Vibration*, 315(4):797–815, 2008.

[45] Amine Kamel, Friedrich Lange, and Gerd Hirzinger. An industrial-robots suited input shaping control scheme. *Motion and Vibration Control*, pages 177–188, 2009.

[46] Withit Chatlatanagulchai, Puwadon Poedaeng, Nitirong Pongpanich, Piyasan Praserthdam, EJ Editor, Ekatet Intakan, Editorial Office, and Yan Zhao. Improving closed-loop signal shaping of flexible systems with smith predictor and quantitative feedback. *Engineering Journal*, 20(5), 2016.

[47] Craig F Cutforth and Lucy Y Pao. Control using equal length shaped commands to reduce vibration. *IEEE transactions on control systems technology*, 11(1):62–72, 2003.

[48] Yu Zhao, Wenjie Chen, Te Tang, and Masayoshi Tomizuka. Zero time delay input shaping for smooth settling of industrial robots. In *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*, pages 620–625. IEEE, 2016.

[49] Sonja Macfarlane and Elizabeth A Croft. Jerk-bounded manipulator trajectory planning: design for real-time applications. *IEEE Transactions on Robotics and Automation*, 19(1):42–52, 2003.

[50] Matthias Oberherber, Hubert Gattringer, and Andreas Müller. Successive dynamic programming and subsequent spline optimization for smooth time optimal robot path tracking. *Mechanical Sciences*, 6(2):245–254, 2015.

[51] R. K. Rao Yarlagadda. *Analog and digital signals and systems*, volume 1. Springer, 2010.

[52] Richard C. Dorf and Robert H. Bishop. *Modern control systems*. Pearson (Addison-Wesley), 1998.

[53] Quang-Cuong Pham, Stéphane Caron, Puttichai Lertkultanon, and Yoshihiko Nakamura. Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots. *The International Journal of Robotics Research*, 36(1):44–67, 2017.

[54] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[55] Ioan A Sucan, Mark Moll, and Lydia E Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

[56] Steven M La Valle. Motion planning. *IEEE Robotics & Automation Magazine*, 18(2):108–118, 2011.

[57] Quang-Cuong Pham. A general, fast, and robust implementation of the time-optimal path parameterization algorithm. *IEEE Transactions on Robotics*, 30(6):1533–1540, 2014.

[58] Anil V Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

[59] Daniel Liberzon. *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2011.

[60] Daniela Constantinescu and Elizabeth A Croft. Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. *Journal of robotic systems*, 17(5):233–249, 2000.

[61] Gareth Bradshaw and Carol O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Transactions on Graphics (TOG)*, 23(1):1–26, 2004.

[62] Wikipedia. Trajectory optimization — wikipedia, the free encyclopedia, 2017. [Online; accessed 1-December-2017].

[63] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming*, volume 19. Siam, 2010.

[64] Matthew P Kelly. Transcription methods for trajectory optimization. *Tutorial, Cornell University, Feb*, 2015.

[65] Moritz Diehl, Hans Georg Bock, Holger Diedam, and P-B Wieber. Fast direct multiple shooting algorithms for optimal robot control. In *Fast motions in biomechanics and robotics*, pages 65–93. Springer, 2006.

[66] Elliott Ward Cheney and William Allan Light. *A course in approximation theory*, volume 101. American Mathematical Soc., 2009.

[67] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[68] Lloyd N Trefethen. *Approximation theory and approximation practice*, volume 128. Siam, 2013.

[69] Jean-Paul Berrut and Lloyd N Trefethen. Barycentric lagrange interpolation. *SIAM review*, 46(3):501–517, 2004.

[70] Jörg Waldvogel. Fast construction of the fejér and clenshaw–curtis quadrature rules. *BIT Numerical Mathematics*, 46(1):195–202, 2006.

[71] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*, volume 105. Siam, 2008.

[72] Andrea Walther and Andreas Griewank. Getting started with adol-c. In *Combinatorial scientific computing*, pages 181–202, 2009.

[73] Bradley M Bell. Cppad: a package for c++ algorithmic differentiation. *Computational Infrastructure for Operations Research*, 57, 2012.

[74] Claus Bendtsen and Ole Stauning. Fadbad, a flexible c++ package for automatic differentiation. Technical report, Technical Report IMM–REP–1996–17, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1996.

[75] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.

[76] Xiaowen Yu, Yu Zhao, Cong Wang, and Masayoshi Tomizuka. Trajectory planning for robot manipulators considering kinematic constraints using probabilistic roadmap approach. *Journal of Dynamic Systems, Measurement, and Control*, 139(2):021001, 2017.

[77] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[78] Tamar Flash and Neville Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience*, 5(7):1688–1703, 1985.

[79] T Chettibi, HE Lehtihet, M Haddad, and S Hanchi. Minimum cost trajectory planning for industrial robots. *European Journal of Mechanics-A/Solids*, 23(4):703–715, 2004.

[80] Mang Zhuang and DP Atherton. Automatic tuning of optimum pid controllers. In *IEE Proceedings D (Control Theory and Applications)*, volume 140, pages 216–224. IET, 1993.

[81] Weng Khuen Ho, OP Gan, Ee Beng Tay, and EL Ang. Performance and gain and phase margins of well-known pid tuning formulas. *IEEE Transactions on Control Systems Technology*, 4(4):473–477, 1996.

[82] Zwe-Lee Gaing. A particle swarm optimization approach for optimum design of pid controller in avr system. *IEEE transactions on energy conversion*, 19(2):384–391, 2004.

[83] Alin Albu-Schäffer, Sami Haddadin, Ch Ott, Andreas Stemmer, Thomas Wimböck, and Gerhard Hirzinger. The dlr lightweight robot: design and control concepts for robots in human environments. *Industrial Robot: an international journal*, 34(5):376–385, 2007.

[84] Alessandro De Luca, Dierk Schroder, and Michael Thummel. An acceleration-based state observer for robot manipulators with elastic joints. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3817–3823. IEEE, 2007.

[85] Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.