Title
Controller Synthesis and Vibration Suppression Techniques for Industrial Robotic Manipulators with Joint Flexibilities

Author
Chan, Michael

Publication Date
2013

Peer reviewed|Thesis/dissertation

Controller Synthesis and Vibration Suppression Techniques for Industrial Robotic Manipulators with Joint Flexibilities

by

Michael L Chan

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

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor J. Karl Hedrick
Professor Ming Gu

Fall 2013

Controller Synthesis and Vibration Suppression Techniques for Industrial Robotic Manipulators with Joint Flexibilities

by

Michael L Chan

Abstract

Controller Synthesis and Vibration Suppression Techniques for Industrial Robotic Manipulators with Joint Flexibilities

by

Michael L Chan

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

This dissertation focuses on the design of feedback and feedforward controllers for direct application to industrial manipulators. In this dissertation, an iterative online controller tuning algorithm based on nonlinear programming concepts and extremum seeking control is introduced and applied to a 6 degree of freedom FANUC M16*i*B industrial robot. Details regarding stepsize selection and gradient estimation for the proposed controller tuning is also discussed. Experimental results show that the proposed controller tuning method is able to improve robot performance by successively reducing a cost function. Additionally, a controller tuning framework based off the disturbance observer is also introduced for stable controller tuning. The framework is shown to be robustly stable under most practical tuning applications. The assumptions and constraints of the proposed framework is also detailed. The framework is experimentally verified by sweeping through a variety of controller gains that satisfy the framework conditions. Finally, an input shaping technique for application to industrial robots with elastic joints is also proposed in this dissertation. The approach is simple to apply and can be easily integrated to existing trajectory generation techniques for industrial robots. Experimental results show substantial joint vibration suppression during transient motions.

To my parents

Contents

List of Figures

List of Tables

Acknowledgments

This dissertation is a culmination of 5 years of hard work at the University of California, Berkeley. But needless to say, this dissertation would probably not have been written if it were not for the people in my life supporting me these past years. First and foremost, I would like thank Professor Tomizuka for initially taking me in as his student back in 2009. Under his guidance, I have learned much in the area of control theory and mechatronics. But more importantly, Professor Tomizuka gave me an opportunity to learn how complicated graduate level research can be. It was definitely a humbling experience to realize how little I knew even after completing college. Additionally, I would also like to thank Professor Tomizuka's wife, Miwako sensei, for always being such a cheerful host whenever Professor Tomizuka would throw a party at his home.

I would also like to thank FANUC Corporation for supporting my research these past 5 years. In particular, I would like to personally thank Dr. Seiuemon Inaba, Dr. Yoshiharu Inaba, and Dr. Kiyonori Inaba for their continued support and confidence in the Mechanical Systems Controls (MSC) Laboratory at the University of California, Berkeley. I hope that the continued collaboration between FANUC and Berkeley will continue to produce invaluable technologies for FANUC Corporation.

At this point I would also like to thank my family for their continued support these past few years. In particular, I would like to thank my dad and mom, Pak and Ming, for postponing their retirement for a few years as I toiled away in graduate school. My grandparents for always making home a welcoming place whenever I returned for the holidays. And my brother, Alan, for taking care of things at home while I was away for graduate school.

I would also like to thank the many individuals I met while studying at Berkeley. I would to thank Shih-Yuan, Chi-Shen, Oak, Max, Selina, and Raechel for your friendship and for being an excellent source of entertainment and amusement during both the good times and the bad. In particular, a special shout out to Shih-Yuan and Chi-Shen for inviting me out to the BATS events early during my graduate studies, thus giving me somewhat of a resemblance of a social life in graduate school. I would also like to give my thanks to the other members of the robotics research group, namely Nora, Wenjie, Pedro, Cong, and Chung-Yen. It was a pleasure exchanging intellectual thoughts with you guys (and girl) these past years. I would also like to give my thanks to the remaining individuals in Professor Tomizuka's research lab: (Current members) Chen-Yu, Yaoqiong, Kevin, ChangLiu, Junkai, Robert, Dennis, Yizhou, Xiaowen, Wenlong, (Alumni) Evan, Sanggyum, Joonbum, Nancy, Emma, Hoday, Haifei, Lucy, Steve, KC, Takashi, Steve, Ben, Hiroshi, Omar, Kengo, James, and William. I was able to learn a lot about doing research in fields other than my own due to my interactions with you all and listening to your weekly research seminars. In addition, I would also thank Sugi-san for his technical guidance and friendship while he was still here in Berkeley. I hope to keep in touch and I wish you all the best in your future endeavors.

Chapter 1

Introduction

1.1 Background

Feedback control is often an under appreciated concept. Most human beings, given their sensitive, yet robust, senses of vision, hearing, touch are capable of doing many miraculous feats. Yet even the simplest of feats, such as walking or catching a ball, would not be possible if it was not for the constant visual, audio, and tactile feedback the human body provides to the brain. The brain can process this information which then allows the human to react and, in some cases, anticipate events. This latter concept is called feedforward control by people in the Controls community. Yet despite all the advantages that feedback and feedforward control provides, humanity rarely acknowledges these concepts outside of the Controls community.

Similarly, industrial automation is also an under appreciated field in robotics. Robotics has always stemmed from humanity's desire to create life. Even the term 'robot' was first coined by the Russian science fiction writer Isaac Asimov when he was envisioning a world with robots that were human-like in appearance. As a result, the general public tends to associate robots with androids and other human-like robots. But in reality, the vast majority of the robots today are fixed industrial machines that are not human-like in appearance at all. But more importantly, these industrial machines play a crucial role in the advancement of a nation's manufacturing capabilities.

This dissertation in return focuses on these two under appreciated concepts. More specifically, this dissertation will talk about how fundamental principles such as feedback and feedforward control can be further refined to improve the performance of industrial robots.

1.2 Motivation and Contribution

In today's competitive manufacturing environment, industrial robots are pushed to operate near their designed hardware and software limitations. Even so, manufacturers still demand

more performance from their machinery. Rather than redesigning the hardware for these industrial robots, which is a costly strategy, an alternative is to design better software algorithms. This dissertation focuses primarily on the software aspect by introducing intelligent control algorithms that can improve the performance of industrial manipulators.

This dissertation can be roughly divided into two sections. The first two-thirds of the dissertation focuses on feedback techniques while the last third of the dissertation will emphasize feedforward techniques. In the feedback portion, the dissertation presents a novel application of extremum seeking control (ESC) for use with industrial robotics as well as present a novel platform based off the disturbance observer (DOB) which can be applied for gain tuning applications. In the feedforward portion, a simple input shaping method is demonstrated to work efficiently at reducing transient vibrations in flexible joint robots. While the feedback and feedforward techniques in this dissertation can be used in unison, they are implemented individually in this dissertation for the purpose of demonstrating the effectiveness of each approach.

Sensor-based Feedback Controller Tuning of Robot Manipulators by Nonlinear Programming

In current practice, whenever a new robot model is introduced, experienced engineers have to spend a long period of time to tune and validate the feedback and feedforward controller gains for the robot. This process is performed on a single robot and then applied to all other robots of the same model. As a result, these tuned gains have to be robust to both manufacturing uncertainties and different robot applications. Furthermore, robot dynamics are highly nonlinear and can vary substantially from one configuration to another. While it is possible to use a fixed set of gains to stabilize a robot for its entire workspace, it is highly unlikely that these controller gains can guarantee good performance for any given trajectory in the workspace. Hence in an industrial setting, the ability to quickly optimize the robot controller for any particular task can likely improve robot performance. Manually retuning robots can be a time consuming and expensive process. Hence it is desirable to have an algorithm that can automatically tune robot controllers based on a user specified trajectory. Chapter 3 focuses on the development of an automated gain tuning algorithm for industrial manipulators.

A Disturbance Observer Framework for Stable Feedback Controller Tuning

Feedback control is essential for good performance in almost any electro mechanical system. Consequently, a great deal of effort is put into properly designing feedback controllers. The most practical controller used in industrial robots today is still the proportional plus integral plus derivative (PID) controller. The PID controller has many practical properties that make

them easy to tune and use, namely, the controller parameters have very intuitive physical implications which makes manual tuning feasible. As computational technologies improve, however, it may be possible to take advantage of more advanced higher order controllers to further improve robot performance. Tuning these higher order controllers, however, may not be as intuitive as tuning a PID controller. An alternative method is to empirically tune these higher order controllers using a nonlinear programming technique. This approach, however, may run into stability issues as higher order controllers are more likely to excite a system's higher order dynamics. Chapter 4 focuses on developing a gain tuning framework that is robustly closed loop stable for controller tuning applications.

An Input Shaping Method to Suppress Transient Vibrations In Flexible Joint Robotic Manipulators

Actuators found in mechanical systems have to satisfy a variety of seemingly conflicting requirements. Often these actuators are required to have high positioning accuracy, good repeatability, and high torque capacity while simultaneously required to be compact, light, and competitively priced. To meet these demands, engineers decide to introduce various gear reduction mechanisms (also called transmission) between the motor output and the actual output shaft of the actuator. This way, a light and high speed motor with low torque capacity and moderate positioning accuracy can be used to transmit large amounts of torque with high positioning accuracy. Systems that utilize this actuator and transmission setup are called indirect drive mechanisms. Although they have many benefits, indirect drive mechanisms also create interesting problems for control engineers. Namely, the transmission mechanism has its own dynamic properties. For a variety of reasons, the sensors used for motor feedback are usually placed prior to the transmission mechanism, hence a good feedback controller that provides excellent motor performance cannot guarantee good performance of the output shaft of the actuator. If the transmission dynamics are known, however, feedforward techniques can be used to shape the desired motor reference trajectory to pre-compensate for the transmission dynamics. Chapter 5 focuses on developing an input shaping approach to compensate for the dynamics of a particular family of transmission mechanisms, namely strain wave gearing mechanisms.

1.3 Dissertation Outline

The remainder of this dissertation is organized as follows: Chapter 2 will introduce basic robotics and system modeling ideas used throughout this dissertation. Additionally, the chapter will also detail the experimental setup used to verify the developed algorithms in this dissertation. Chapter 3 will introduce an automatic controller tuning algorithm based on nonlinear programming. This chapter will also provide methods of selecting important parameters such as initial controller gains and stepsizes. Chapter 4 will introduce a frame-

work based on the disturbance observer (DOB) concept that can be used for tuning higher order controllers. The chapter will also prove that the framework is robustly stable given a few mild constraints. Chapter 5 will introduce an input shaping technique to compensate for transmission dynamics. A simple procedure for empirically identifying the transmission parameters will also be introduced. And finally, the main results and contributions of this dissertation will be highlighted in Chapter 6. Additionally, this chapter will also discuss possible extensions for the work presented in this dissertation.

Chapter 2

System Modeling, Hardware Description, and System Identification

2.1 Introduction

This chapter summarizes all the modeling, hardware, and system identification results necessary to understand the remainder of this dissertation. Section 2.2 motivates and introduces the two mass model used for studying indirect drive mechanisms. In particular, this section will also highlight the modeling simplifications used in this dissertation due to the nature of industrial robotic manipulators. Section 2.3 briefly introduces the dynamics for a 6 degree of freedom (DOF) robotic manipulator. Section 2.4 talks about both the physical hardware and simulation software that is used to verify the theory presented in the later chapters of this dissertation. Section 2.5 highlights the system identification process used to obtain empirical frequency response data for the physical hardware. And finally, the contents of this chapter will be summarized in section 2.6.

2.2 Two Mass Model

Given the physical nature of indirect drive mechanisms, many researchers chose to use a two inertia model to physically model the behavior of the indirect drive mechanism [39, 16, 23]. This two inertial model is shown in Fig. 2.1. J_* and θ_* denote the inertia and displacement. The subscripts m and l denote the motor side and load side parameters respectively. In the proposed model, the torque input, u, is applied on the motor side. Furthermore, motor side viscous damping is captured by the viscous damping coefficient, d_m, whereas the other motor side nonlinear forces and damping effects are denoted generally as $f_{nl,m}$. The transmission is modeled as a combination of a linear spring and viscous damper, whose coefficients are denoted by k_j and d_j respectively, as well as nonlinear joint friction forces and joint transmission error denoted by $f_{nl,j}$ and $\tilde{\theta}$ respectively. Also note that the

transmission mechanism also reduces the displacement from the motor side to load side by a factor of N. The transmission error, $\tilde{\theta}$ is defined as the deviation between the expected output position and the actual output position. More specifically, it is given as:

$$\tilde{\theta} = \frac{\theta_m}{N} - \theta_l \tag{2.1}$$

The behavior of the transmission error depends heavily on the transmission mechanism itself. The transmission error will be addressed more specifically later in this chapter. Looking back at the two inertial model, performing a torque balance on the two inertia elements in Fig. 2.1 yields:

$$\begin{aligned} J_m\ddot{\theta}_m &= -d_m\dot{\theta}_m - \tfrac{k_j}{N}\left(\tfrac{\theta_m}{N} - \theta_l - \tilde{\theta}\right) - \tfrac{d_j}{N}\left(\tfrac{\dot{\theta}_m}{N} - \dot{\theta}_l - \dot{\tilde{\theta}}\right) - f_{nl,m} - f_{nl,j} + u \\ J_l\ddot{\theta}_l &= -k_j\left(\theta_l - \tfrac{\theta_m}{N} + \tilde{\theta}\right) - d_j\left(\dot{\theta}_l - \tfrac{\dot{\theta}_m}{N} + \dot{\tilde{\theta}}\right) \end{aligned} \tag{2.2}$$

Note that the motor side and load side displacements, velocities, and accelerations are coupled together in Eq. (2.2). Furthermore, if the nonlinear friction effects and transmission error are ignored, the ideal relationship between the motor input and motor position as well as load position in the Laplace domain is given by:

$$G_{mu}(s) = \frac{\theta_m(s)}{u(s)} = \frac{J_l s^2 + d_j s + k_j}{J_m J_l s^4 + J_d s^3 + J_k s^2 + k_j d_m s} \tag{2.3}$$

$$G_{lu}(s) = \frac{\theta_l(s)}{u(s)} = \frac{d_j s + k_j}{N\left(J_m J_l s^4 + J_d s^3 + J_k s^2 + k_j d_m s\right)} \tag{2.4}$$

where:

$$J_d = J_m d_j + J_l\left(\frac{d_j}{N^2} + d_m\right) \tag{2.5}$$

$$J_k = J_m k_j + \frac{J_l k_j}{N^2} + d_j d_m \tag{2.6}$$

Other relevant transfer function variants for the two inertia system are:

$$G_{dmu}(s) = \frac{\dot{\theta}_m(s)}{u(s)} = \frac{J_l s^2 + d_j s + k_j}{J_m J_l s^3 + J_d s^2 + J_k s + k_j d_m} \tag{2.7}$$

$$G_{ddlu}(s) = \frac{\ddot{\theta}_l(s)}{u(s)} = \frac{d_j s^2 + k_j s}{N\left(J_m J_l s^3 + J_d s^2 + J_k s + k_j d_m\right)} \tag{2.8}$$

Transmissions for Industrial Manipulators

Flexible gear reducers are commonly used in industrial robots due to their high gear reduction ratios [33]. Among the different types of flexible gear reducers, the discussion in this

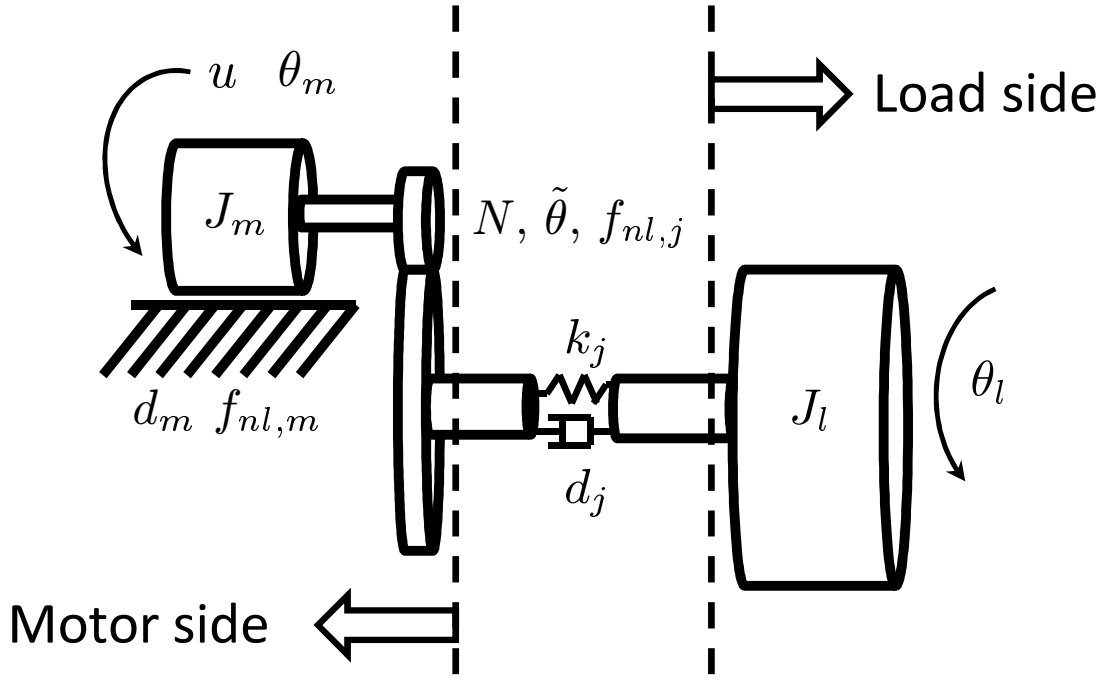


Figure 2.1: Two inertial model of an indirect drive mechanism

subsection will focus harmonic drives since variants of their transmission properties can be found in many of the other reducers used in industrial robots. The main components of a harmonic drive are identified in Fig. 2.2 [21]. The wave generator is a rigid core having an slightly elliptical shape. The wave generator fits inside the flexspline, which is usually a thin-walled hollow cup. The teeth of the flexspline are on the outside surface of the hollow cup and the output of the harmonic drive is also attached to the flexspline. The flexspline fits inside the circular spline. The circular spline is a rigid and fixed component that has teeth machined along the inside surface. Usually, there are two fewer teeth along the outside of the flexspline as there are along the inside of the circular spline. This difference in teeth causes the flexspline to essentially rotate by the difference in teeth for every full rotation of the wave generator. This design allow harmonic drives to efficiently achieve large gear reduction ratios. Additionally, the inherent multiple-tooth contact design in harmonic drives also allow them to withstand high amounts of torque while simultaneously eliminating all transmission backlash [38]. Additional advantageous properties of harmonic drives include: lightweight and compact design, high efficiency, and backdrivability.

The harmonic drive, however, is not without its disadvantages. The use of the flexspline in the harmonic drive instills a large amount of flexibility in the drive. Additionally the concentric nature of the harmonic drive assembly tend to give rise to kinematic errors due to manufacturing and alignment inaccuracies. Studies show that these kinematic errors

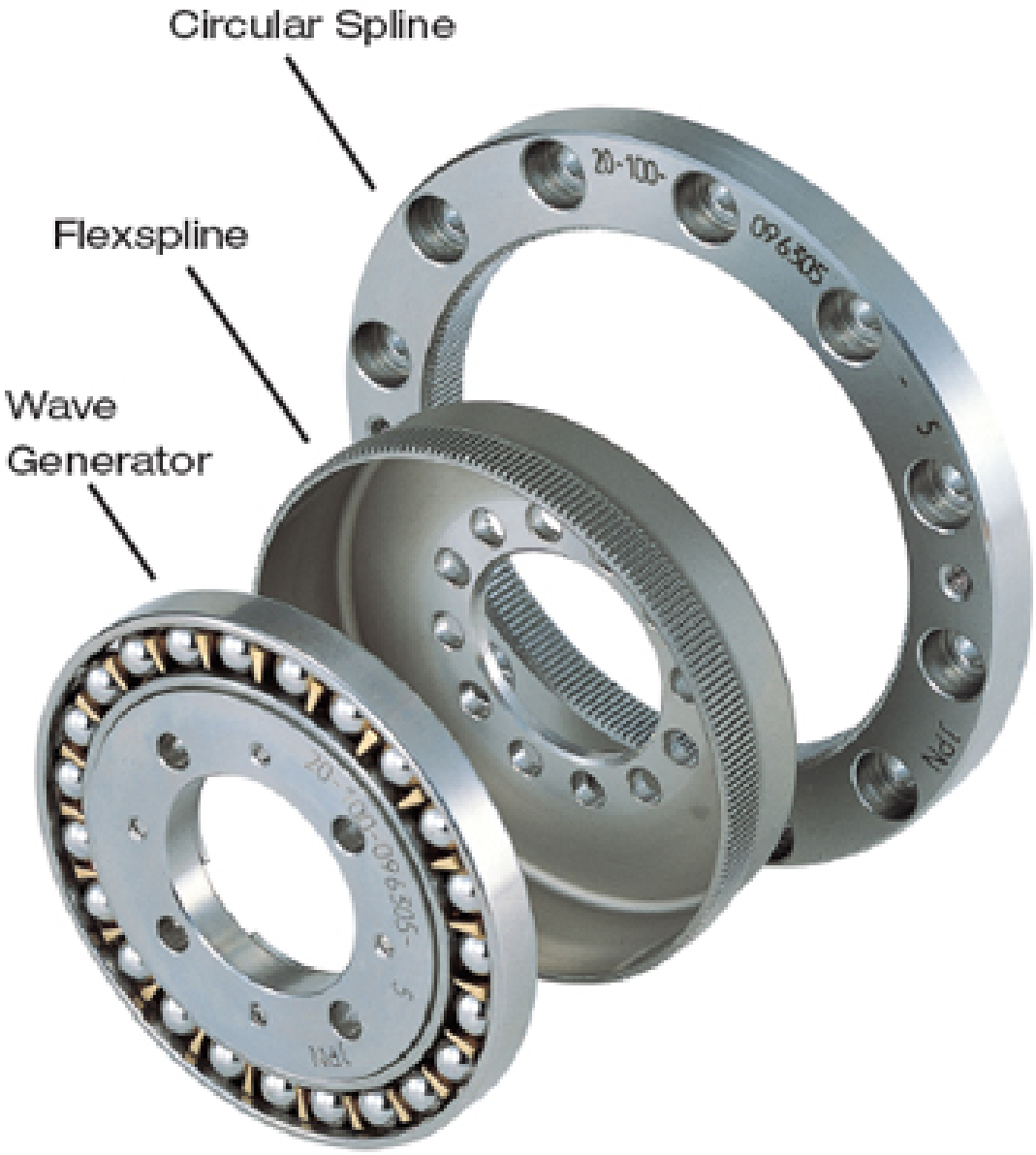


Figure 2.2: Harmonic drive components

occur dominantly at a frequency of twice the wave generator rotation velocity. The root cause of these kinematic errors are caused by manufacturing and assembly imperfections in the harmonic drive. More specifically, the tooth placement errors along the flexspline and circular spline as well as misalignment of the three major harmonic drive components [38]. In this dissertation, the kinematic errors caused by component misalignment, manufacturing

imperfections, and nonlinear flexspline flexibilities are denoted as the transmission error $\tilde{\theta}$. Additionally the nonlinear friction effect caused by the meshing of the flexspline and circular spline is denoted by $f_{nl,j}$ whereas the linear flexibility and damping effects are characterized by the linear joint stiffness and damping coefficients k_j and d_j respectively.

Much work has been done on trying to compensate for the transmission error and nonlinear friction effects found in harmonic drives [20, 17, 8]. As a result, the work in this dissertation will not focus on the transmission error or nonlinear friction effects, but instead will focus primarily on compensating for the linear flexibilities and damping. For all intensive purposes, the nonlinear friction effects and transmission error will be treated as process noise or disturbances in this dissertation.

2.3 Multiple Degree of Freedom Robot Model

This dissertation will only consider serial industrial manipulators. The dynamics of a n DOF serial industrial manipulator with joint flexibilities can be derived through Lagrangian dynamics and can be expressed generally as:

$$\begin{aligned} &M_l(q_l)\ddot{q}_l + C(q_l,\dot{q}_l)\dot{q}_l + G(q_l) + D_l\dot{q}_l + F_{lc}\text{sgn}(\dot{q}_l) + J(q_l)^T f_{\text{ext}} \\ &= K_j\left(N^{-1}q_m - q_l - \tilde{q}\right) + D_j\left(N^{-1}\dot{q}_m - \dot{q}_l - \dot{\tilde{q}}\right) \end{aligned} \tag{2.9}$$

$$\begin{aligned} &M_m\ddot{q}_m + D_m\dot{q}_m + F_{mc}\text{sgn}(\dot{q}_m) = \\ &\tau_m - N^{-1}\left[K_j\left(N^{-1}q_m - q_l - \tilde{q}\right) + D_j\left(N^{-1}\dot{q}_m - \dot{q}_l - \tilde{q}\right)\right] \end{aligned} \tag{2.10}$$

where $q_l \in \mathbb{R}^n$ and $q_m \in \mathbb{R}^n$ denote the load side and motor side position vectors where the i^{th} element denote the position at the i^{th} joint. $\tilde{q} \in \mathbb{R}^n$ is the vector of transmission errors caused by flexible joint reducers. $M_* \in \mathbb{R}^{n\times n}$, $D_* \in \mathbb{R}^{n\times n}$, and $F_{*c} \in \mathbb{R}^{n\times n}$ are the inertia, viscous damping, and coulomb friction matrices respectively. Again the subscripts l and m denote load side and motor side respectively. $C(q_l,\dot{q}_l) \in \mathbb{R}^{n\times n}$ is the Coriolis and centrifugal force matrix, $G(q_l) \in \mathbb{R}^n$ is the gravity torque vector, $N \in \mathbb{R}^{n\times n}$ is the matrix containing the gear reduction ratios of each reducer, and $J(q_l) \in \mathbb{R}^{6\times n}$ is the Jacobian matrix which maps the load side joint space to the end-effector Cartesian space. $K_j \in \mathbb{R}^{n\times n}$ and $D_j \in \mathbb{R}^{n\times n}$ are the joint linear stiffness and viscous damping matrices respectively. Vectors $\tau \in \mathbb{R}^n$ and $f_{\text{ext}} \in \mathbb{R}^6$ denote the motor input torques and external forces/torques acting on the robot end-effector in the Cartesian coordinate system. Note that M_m, K_j, D_j, D_l, D_m, F_{lc}, F_{mc}, and N are all diagonal matrices.

In the case where the joints are rigid, Eqs. (2.9-2.10) can be combined and rewritten as:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + D\dot{q} + F_c\text{sgn}(\dot{q}) + J(q)^T f_{\text{ext}} = \tau \tag{2.11}$$

where $q = q_l$, $\tau = N\tau_m$, $M(q) = M_l(q_l) + N^2M_m$, $C(q,\dot{q}) = C(q_l,\dot{q}_l)$, $D = D_l + N^2D_m$, $F_c = F_{lc} + NF_{mc}$, and $J(q) = J(q_l)$.

Decentralized Analysis

In general, it is difficult to analyze and design control algorithms for the coupled multiple-input-multiple-output (MIMO) system described by Eq. (2.9-2.10). More specifically, the inertia, gravity, external, Coriolis, and centrifugal forces cause the behavior at each individual joint to be coupled with each other. This coupling is characterized by the off diagonal terms in the corresponding matrices. In many cases, however, the diagonal terms in these matrices are substantially larger than those in the off diagonal terms. While the coupling effects are non-negligible, this dissertation will mainly treat the analysis of each joint in a multiple degree of freedom robot as essentially decoupled. The coupling effects will be regarded as regular process noise. Although this is a nontrivial assumption, the developed algorithms in this dissertation will be experimentally validated. The experimental results will shed light on the limitations caused by this assumption.

2.4 Hardware and Software Setup

As mentioned in the previous section, all the developed algorithms in this dissertation are experimentally verified. The FANUC M-16*i*B [14] industrial manipulator, provided to the University of California, Berkeley, by FANUC Corporation, is used for all the experimental verifications in this dissertation. This setup is shown in Fig. 2.3. The FANUC M-16*i*B is a standard 6-DOF serial industrial manipulator with a 20 kg payload capacity. The naming and positive rotation conventions for each joint of the FANUC M-16*i*B robot is depicted in Fig. 2.4. Each joint of the robot uses a gear reduction mechanism. Due to the high inertia loads on the first three joints (J1-J3), the joint flexibilities in these joints are dominant compared to those on the last three joints (J4-J6). As a result, this dissertation will focus on the joint flexibilities of J1-J3 only whereas J4-J6 are assumed to be rigid. The first three joints of the robot uses Rotor-Vector (RV) reducers [34]. RV reducers contains elements of both a planetary gearbox and a harmonic drive. The RV reducer design allows it to keep most of the strengths and weakness found in a harmonic drive. One noticeable difference is that the dominant kinematic error in RV reducers occurs at a frequency that is eight times that of the wave generator rotation frequency.

For sensing, the FANUC robot is equipped with motor side encoders at each joint. These encoders are standard on the commercially available M-16*i*B and provide the motor position and velocity information for feedback. In addition to the motor encoders, an inertia sensor (Analog Devices, ADIS16400) [1] has also been attached to the end-effector to provide three dimensional end-effector angular velocity and translational acceleration measurements. The three dimensional end-effector position can also be measured with the CompuGauge 3D [12]. If direct contact to the robot end-effector is not desirable, then a Position Sensitive Detector (PSD) [11], also referred to as the PSD camera, can be utilized to obtain the end-effector position measurements. In this dissertation, only the motor encoders are used for real-time

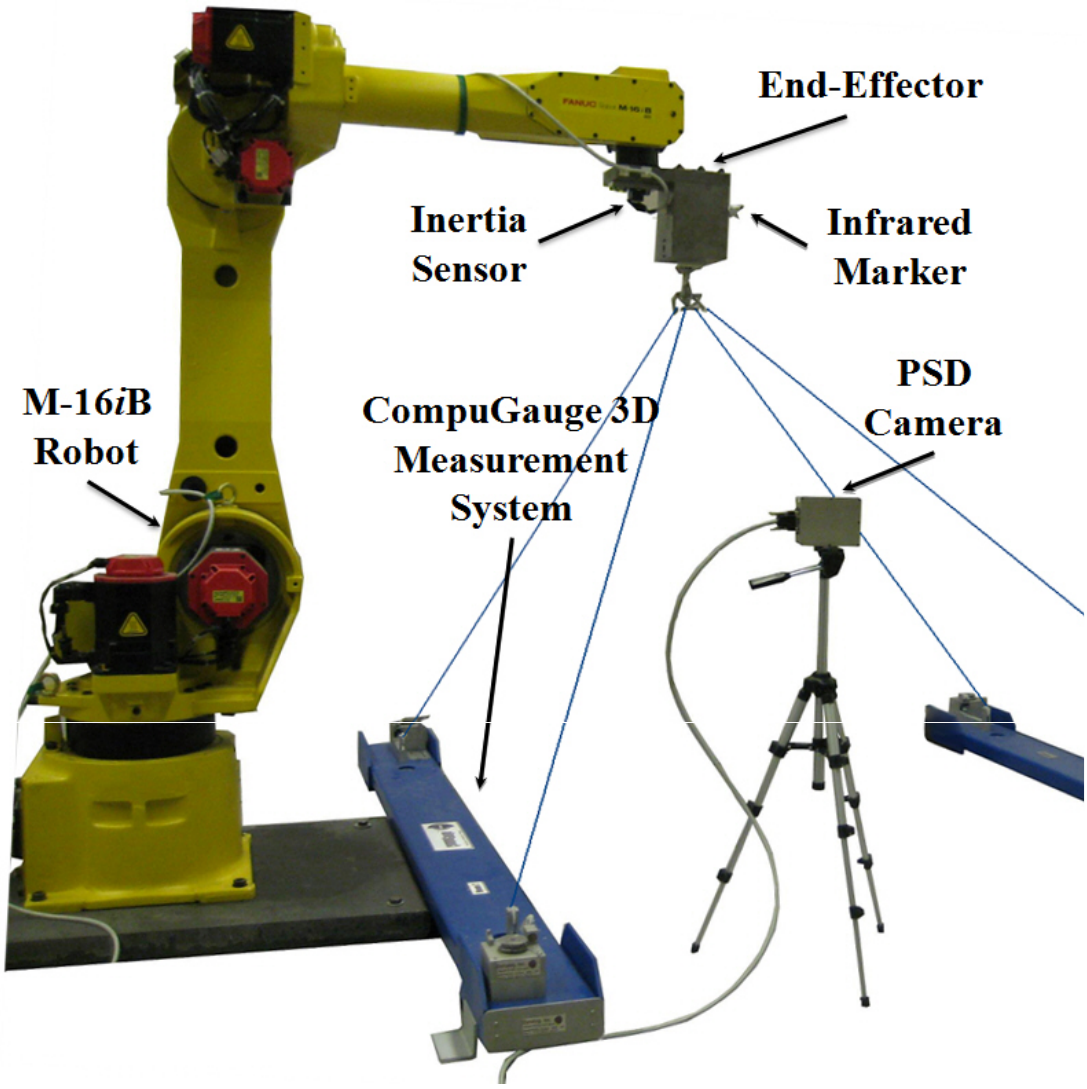


Figure 2.3: FANUC M16iB industrial manipulator

feedback. The other load side sensors (i.e. inertial sensor, CompuGauge 3D, PSD camera) are only used for performance evaluation of the algorithms proposed in this dissertation.

The hardware configuration for the FANUC robot is depicted in Fig. 2.5. While the commercial M-16iB robot controller is capable of moving the robot along any desired trajectory, it does not allow for flexibilities such as adjusting the controller parameters or algorithm. As a result, MATLAB [28] is used to design the control algorithms and a digital to analog servo adaptor (DSA) is used to transfer the information to the robot. By using the DSA, the control signal (i.e. motor torque command) is generated by the target computer and is then converted to output current by the robot controller. Although the robot controller is

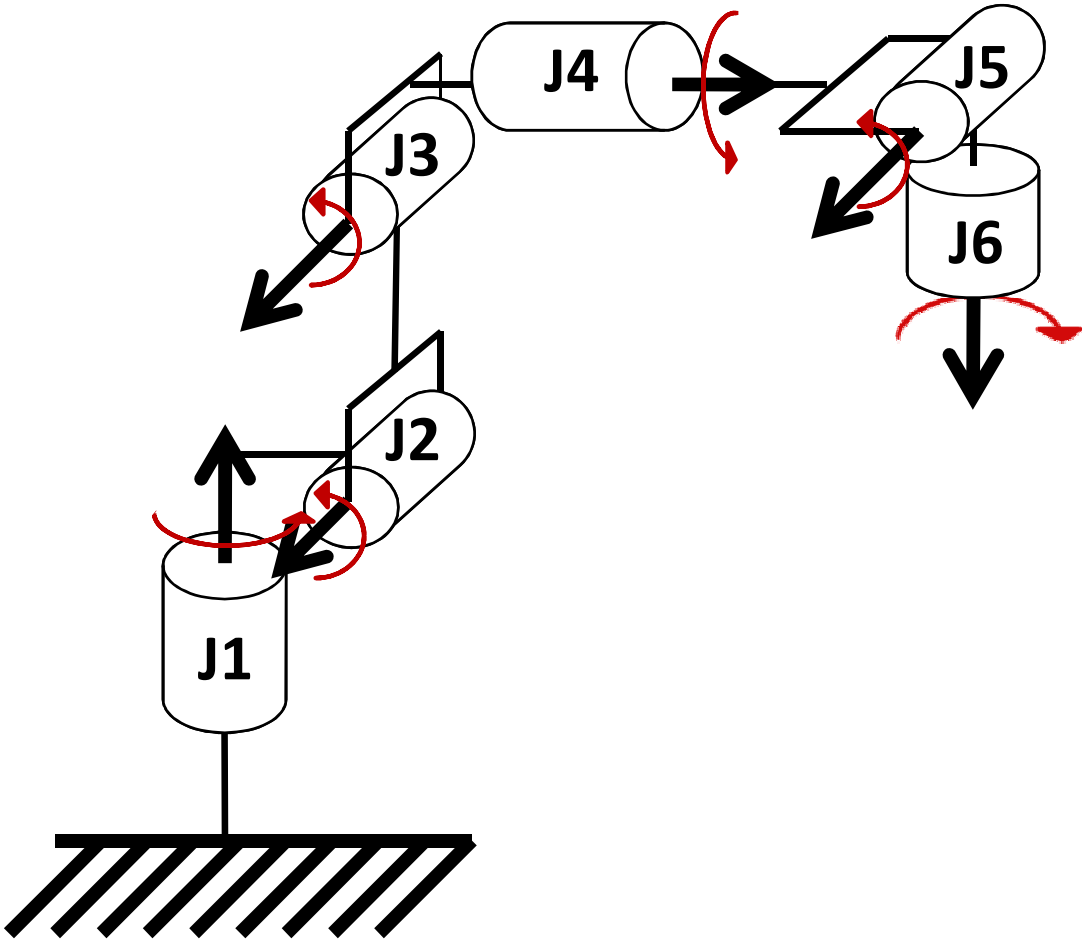


Figure 2.4: Joint naming and rotating conventions for the FANUC M16*i*B robot

not directly generating signals to control the robot, it is still used to amplify the output from the target computer. In addition, the robot controller is also used to activate the mechanical motor brakes in the robot. Note that the brake function is turned on and off directly with a digital input/output (DIO) board installed on the target computer.

In operating real-time systems such as the FANUC robot using MATLAB, it is critical that computational hardware performance runs as quickly and as consistently as possible. More specifically, it is important that the hardware performance of the robot is not limited by computational limitations from the computer used for control. Since MATLAB runs on a Windows platform, it is not guaranteed that the real-time computation will not be hindered by virus scanning software and other event logging processes. To overcome this problem, two computers are used for the real-time control system setup. A host computer running Windows and MATLAB is used to design and implement the algorithms in this disserta-

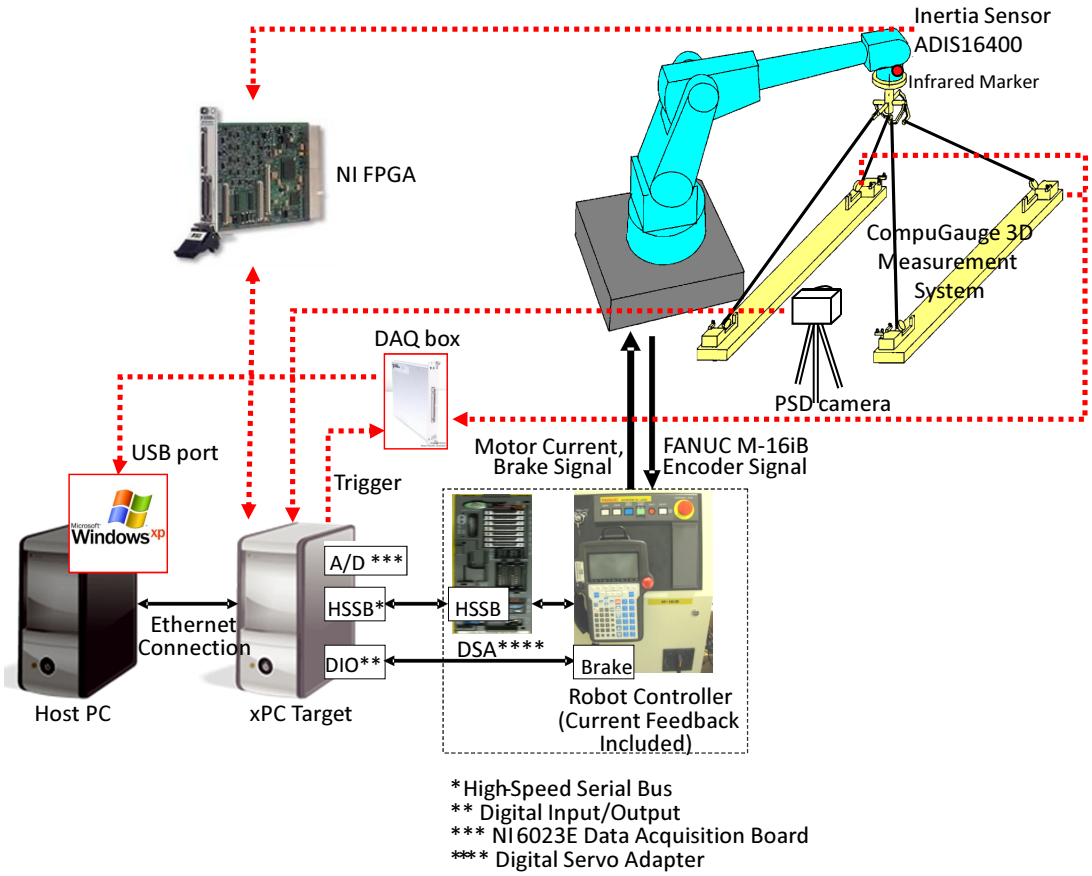


Figure 2.5: FANUC M16*i*B robot system setup scheme

tion. Once the design and coding for the algorithms are complete on the host computer, the information is then compiled and loaded onto the second computer via an ethernet connection. This second computer, also called the target computer, runs the MATLAB XPCtarget environment. The XPCtarget environment allows the target computer to execute the compiled information from the host computer. Once the target computer begins running the control algorithm, the connection between the two computers is automatically disengaged, thus allowing the target computer to run without any interferences from the host computer. The current minimum sampling time for the XPCtarget is 0.5 msec. Additionally, the robot encoder information is accessible by the target computer through a high-speed serial bus (HSSB) interface while the inertia sensor information is acquired through an National Instruments (NI) FPGA board.

In addition to the aforementioned hardware a simulator is also available for the FANUC M-16*i*B robot. The simulator is mainly constructed in the MATLAB Simulink environment and utilizes both the SimMechanics and Robotics toolbox [13]. This simulator is capable of

simulating the dynamic behaviors of the robot. Preliminary studies show that the behavior of the physical robot system can be captured by the robot simulator model. As a result, the robot simulator is used to test the proposed algorithms in this dissertation prior to actual implementation on the FANUC M-16*i*B robot.

Controller Structure

By default, the FANUC M-16*i*B robot uses a decentralized PID feedback controller scheme. The block diagram for a single joint of the robot is depicted in Fig. 2.6. The controller consists of two loops, an inner velocity feedback loop and an outer position feedback loop. The inner loop is controlled with a Proportional plus Integral (PI) controller with parameters K_v and K_i respectively whereas the outer loop is controlled with a simple Proportional controller with parameter K_p [15]. Combining the two loops together, the resulting feedback controller is a Proportional-Integral-Derivative (PID) controller whose transfer function from the motor position error, $e = \theta_{m,\text{ref}} - \theta_m$, to the feedback torque, u_{fb}, is given by:

$$C(s) = \frac{u_{fb}(s)}{e(s)} = \frac{K_v s^2 + (K_p K_v + K_i)\, s + K_p K_i}{s} \tag{2.12}$$

The feedforward torque, u_{ff} is computed using a recursive algorithm that solves the Newton-Euler equations for the robot [35] and is represented simply as just F_2 block in Fig. 2.6. Note that the Newton-Euler equations are solved in a centralized manner, hence all the reference joint trajectories are needed. This aspect is not depicted in Fig. 2.6. The algorithm is implemented via the robotics toolbox mentioned earlier. The feedforward torque calculation requires the desired robot joint trajectory to be entirely known in advance. Although the algorithm for computing the feedforward torque is recursive, it is extremely computationally intensive and cannot be computed in real time. The computed feedforward torque using this approach, however, is extremely accurate and provides the majority of the performance for the robot manipulator. The primary role of the feedback controller then is to compensate for any residual errors that is not captured by the recursive Newton-Euler (RNE) feedforward torque. Since most of the time, it is more convenient to define the robot's trajectory from the load side perspective, the feedforward block F_1 is a conversion between the desired load side trajectory, $\theta_{l,ref}$, to the desired motor side trajectory, $\theta_{m,ref}$. In many cases, F_2 is simply a multiplication by the respective gear ratio N.

2.5 System Identification

As mentioned in Section 2.3, each joint of the 6 DOF industrial robot will be analyzed in a decoupled manner. In this decoupled analysis, a single two inertia model as described in Fig. 2.1 is used to represent the dynamic behavior of each joint. As with all models, the two inertia model is a simplified model that cannot capture all the dynamics of the physical

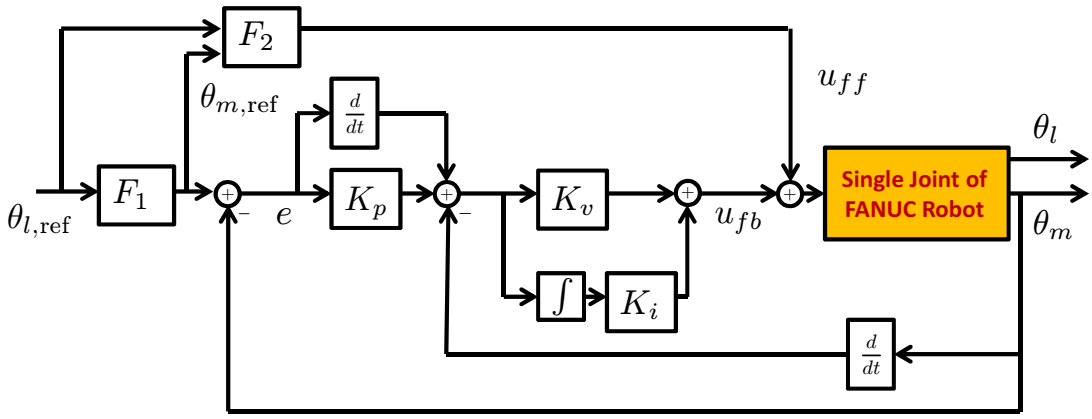


Figure 2.6: Decentralized feedback controller scheme

system. Hence a system identification on the physical FANUC M-16iB robot is performed to measure the actual dynamic behavior of the robot. This data is useful for both identifying some of the model parameters and also learning the bandwidth limitations of the two inertia model.

The system identification was performed one joint at a time. Two different system identification experiments were performed for each joint. The first is a sine sweep experiment which harmonically excites the joint from 1 Hz to 100 Hz over a 180 second span. The objective of the sine sweep experiment is to capture an initial estimate of the joint frequency response. The second experiment is a sine by sine test. In the sine by sine test, the joint is excited at a particular frequency for a fixed amount of time, stops, and then proceeds to the next excitation frequency. The sine by sine experiment provide a more accurate frequency response than the sine sweep experiment since the excitation frequency is stationary at each frequency. In addition, the sine by sine gives the user more flexibility in setting testing parameters such as the experimental time duration spent exciting each frequency. This is important as lower excitation frequencies require a longer excitation duration to acquire meaningful data. For each system identification experiment, three measurements: motor position, θ_m, motor velocity, $\dot{\theta}_m$, and load acceleration, $\ddot{\theta}_l$ are recorded. The control structure for the system identification process is shown in Fig. 2.7. During the system identification process, the reference trajectory for each joint is set to zero and the feedforward torque, u_{ff}, is used for gravity compensation only. The sinusoidal torque, u_{ref} is injected at the same point as the feedforward torque. Since the reference trajectory is set to zero at every joint, the feedback controller, $C(s)$, may produce conflicting torque commands at the joint that is undergoing system identification. As such, the gains of $C(s)$ for the joint being identified is set to be substantially lower to reduce the feedback controller effects. More specifically, the

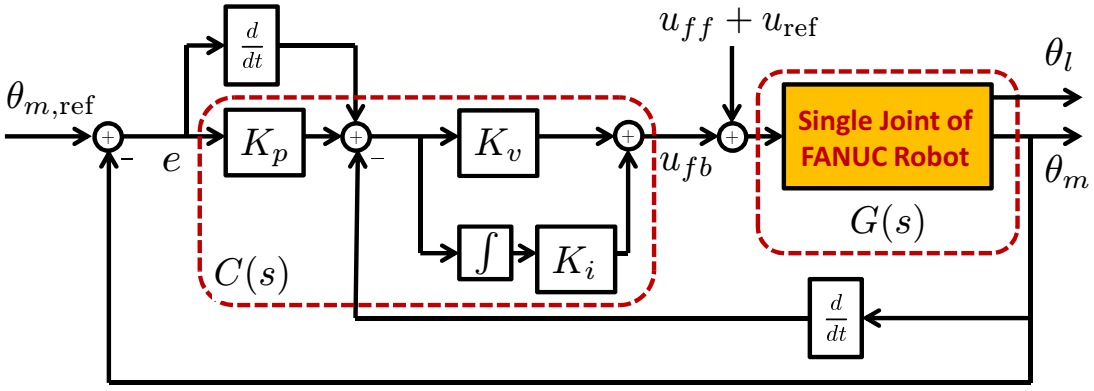


Figure 2.7: Controller for system identification

closed loop transfer function in Fig. 2.7 can be expressed as:

$$\frac{\theta_m}{u_{\text{ref}}} = \frac{G(s)}{1 + G(s)C(s)} \tag{2.13}$$

Hence by selecting the gains, K_p, K_v, and K_i to be very small, $G(s)C(s) \ll 1$ and the effect of the feedback controller can be neglected. For joints that are not heavily influenced by gravity effects, the controller gains can be set exactly to zero. But non-zero feedback controller gains are necessary to maintain safe robot motions during the system identification process otherwise. In all experiments and for all joints, the integrator gain, K_i, is set to zero. The remaining controller controller gains used for each joint during the system identification process for each joint are tabulated in Table 2.1. Additionally, the posture of the FANUC M-16iB robot during the system identification process for each joint is shown in Fig. 2.8.

Parameter Identification

Much of the postprocessing of the measured data is done through the MATLAB's System Identification Toolbox. The processing accepts the input torque command and measured output data as an input and returns the estimated magnitude and phase for this input-output relationship at each measured frequency. Ideally the system identification should be performed in an open loop manner. But as mentioned in the previous section, small feedback gains are used to keep the robot system stable during the experiment due to gravity effects. But since the structure of the feedback controller and the gain values are known, it is possible to calculate the open loop transfer function from the estimated closed loop data. The first step to doing so is to convert the magnitude and phase estimates into the real and imaginary parts of the closed loop transfer function. Consider a closed loop transfer function:

$$G_{cl}(j\omega) = \alpha(\omega) + j\beta(\omega) \tag{2.14}$$

Table 2.1: Feedback parameters for system identification

Joint ID		Joint Feedback Gains					
		J_1	J_2	J_3	J_4	J_5	J_6
J_1	K_p	0	10	10	20	20	20
	K_v	0	0.2280	0.05072	0.08592	0.0046	0.003986
J_2	K_p	10	5	10	20	20	20
	K_v	0.1533	0.1	0.05072	0.08592	0.0046	0.003986
J_3	K_p	10	10	2	20	20	20
	K_v	0.1533	0.2280	0.01	0.08592	0.0046	0.003986
J_4	K_p	10	10	10	0	20	20
	K_v	0.1533	0.2280	0.05072	0	0.0046	0.003986
J_5	K_p	10	10	10	20	0	20
	K_v	0.1533	0.2280	0.05072	0.08592	0	0.003986
J_6	K_p	10	10	10	20	20	0
	K_v	0.1533	0.2280	0.05072	0.08592	0.0046	0

where $\alpha(\omega) \in \mathbb{R}$ and $\beta(\omega) \in \mathbb{R}$ represents the coefficients of the real and imaginary part of the closed loop transfer function, G_{cl}, evaluated at frequency ω. The magnitude and phase of the closed loop transfer function can be represented as:

$$|G_{cl}(\omega)| = \sqrt{\alpha^2(\omega) + \beta^2(\omega)} \tag{2.15}$$

$$\angle G_{cl}(\omega) = \tan^{-1}(\frac{\beta(w)}{\alpha(w)}) \tag{2.16}$$

With some algebraic manipulation, the coefficients $\alpha(\omega)$ and $\beta(\omega)$ can be expressed as:

$$\beta(\omega) = \alpha(\omega)\tan(\angle G_{cl}(\omega)) \tag{2.17}$$

$$\alpha(\omega) = \sqrt{\frac{|G_{cl}(\omega)|^2}{1 + \tan^2(\angle G_{cl}(\omega))}} \tag{2.18}$$

Due to the inherent nature of $\tan(\cdot)$, one needs to be careful when assigning the signs for $\alpha(\omega)$ and $\beta(\omega)$ using Eqs. (2.17-2.18). Once the coefficients are calculated, one can deduce the real and imaginary parts of $G(j\omega)$ by equating the real and imaginary parts of $G_{cl}(j\omega)$ and utilizing Eq. (2.13). More specifically, one can define the following:

$$C(j\omega) = \sigma(\omega) + j\gamma(\omega) \tag{2.19}$$

$$G(j\omega) = x(\omega) + jy(\omega) \tag{2.20}$$

Combining Eq. (2.14), (2.19), and (2.20) with Eq. (2.13) and equating the real and imaginary parts on both sides of the equations yield two equations and two unknowns, namely $x(\omega)$ and

$y(\omega)$. Note that since the controller structure and gains are known, $\sigma(\omega)$ and $\gamma(\omega)$ can be computed. The bode plots of the system identification results from the motor input torque, $u_{/rmref}$ to motor velocity, $\dot{\theta}_m$, are shown in Figs 2.9-2.14. The blue circles indicate the sine sweep results whereas the red crosses indicate the sine by sine results. Note that at low frequencies, both the sine sweep and sine by sine results are closely matched. Furthermore, the system identification results suggest that the first three joints (J1-J3) have much simpler than dynamic behavior than the three wrist joints (J4-J6) of the FANUC robot. Also note that the first three joints have a clear antiresonance and resonance behavior at about the 10 Hz range. This dynamic behavior can be captured by the two mass model relationship given in Eq. (2.7). The parameter identification and transfer function fitting process is detailed in later relevant chapters.

2.6 Chapter Summary

This chapter introduced the two inertia model used to model indirect drive mechanisms. The chapter also talked about typical transmission used in industrial robots. In particular, the harmonic drive was discussed in detail in Section 2.2. Then the dynamic equations of a multi DOF robot was briefly introduced. A few modeling assumptions and their justifications were also discussed. The chapter then talked about the hardware and software configuration used when producing the results in this paper. Finally, the chapter closed by discussing the system identification process used to identify and characterize the actual hardware behavior. The material in this chapter is a necessary prerequisite to understanding the later chapters of this dissertation.

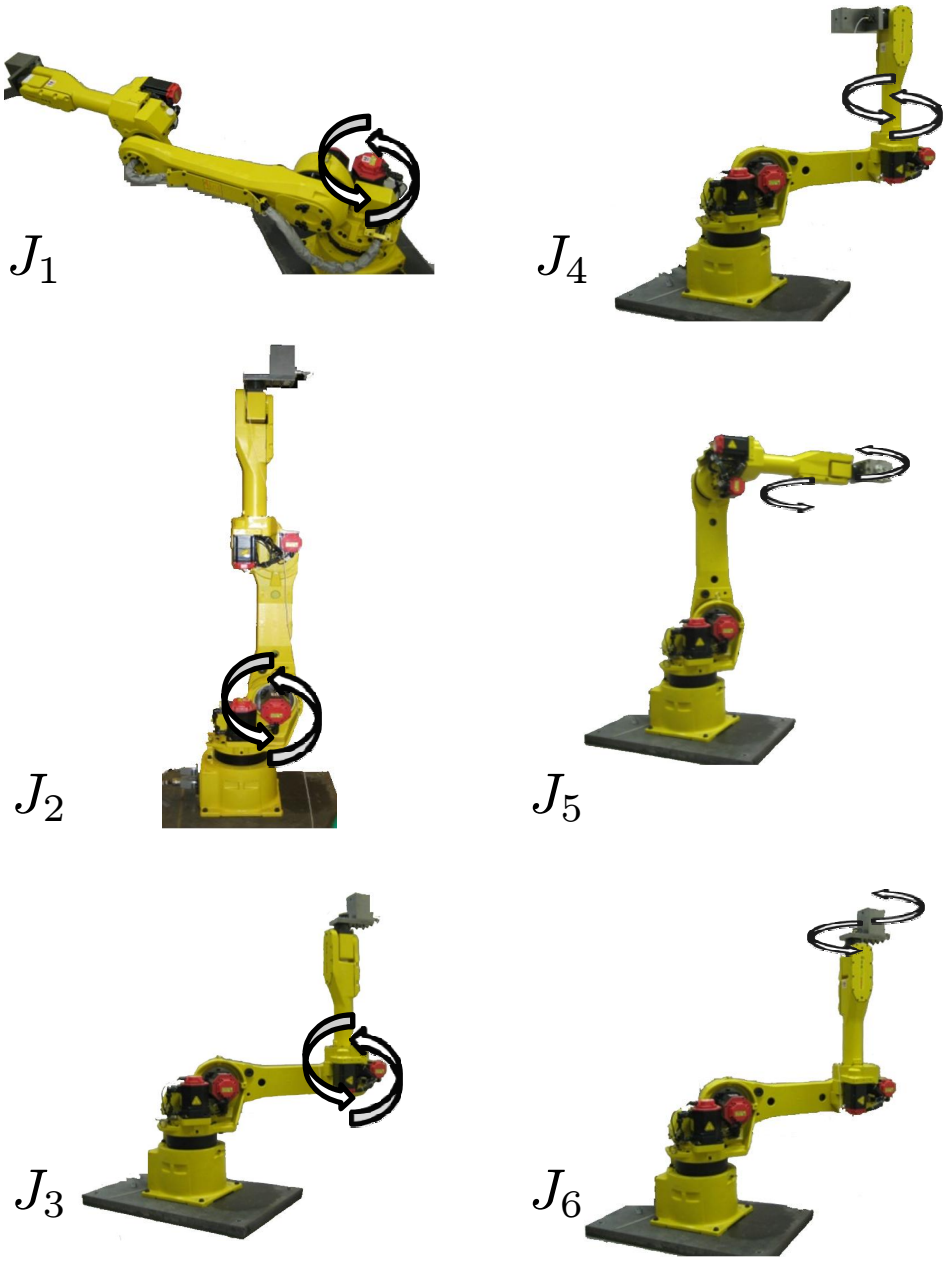


Figure 2.8: Robot postures for system identification

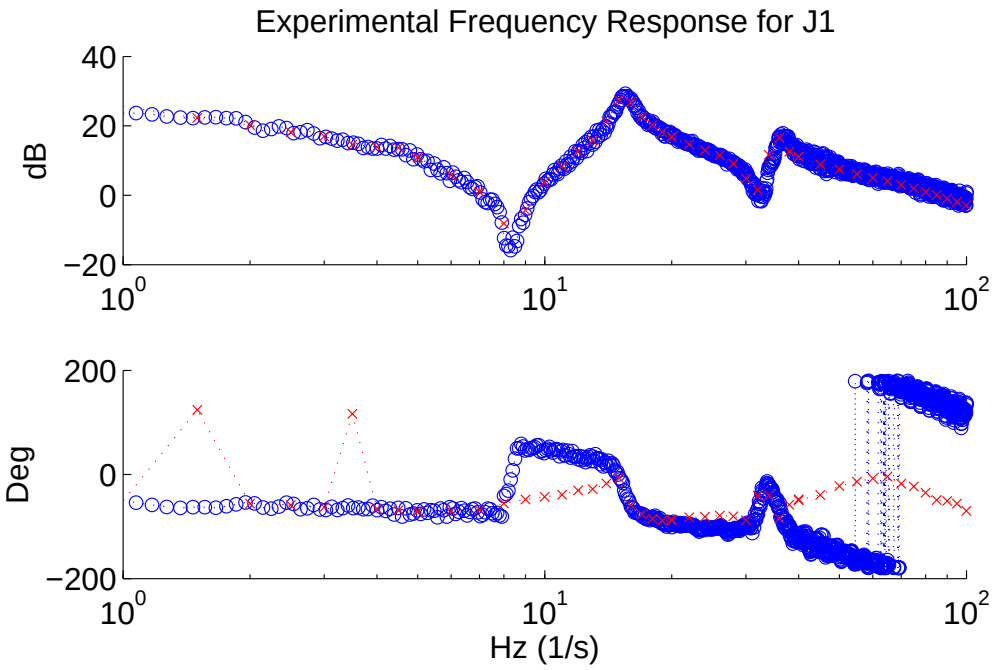


Figure 2.9: System identification result for joint 1: $u_{\text{ref}} \Rightarrow \dot{\theta}_m$ (o: sine sweep results, x: sine by sine results)

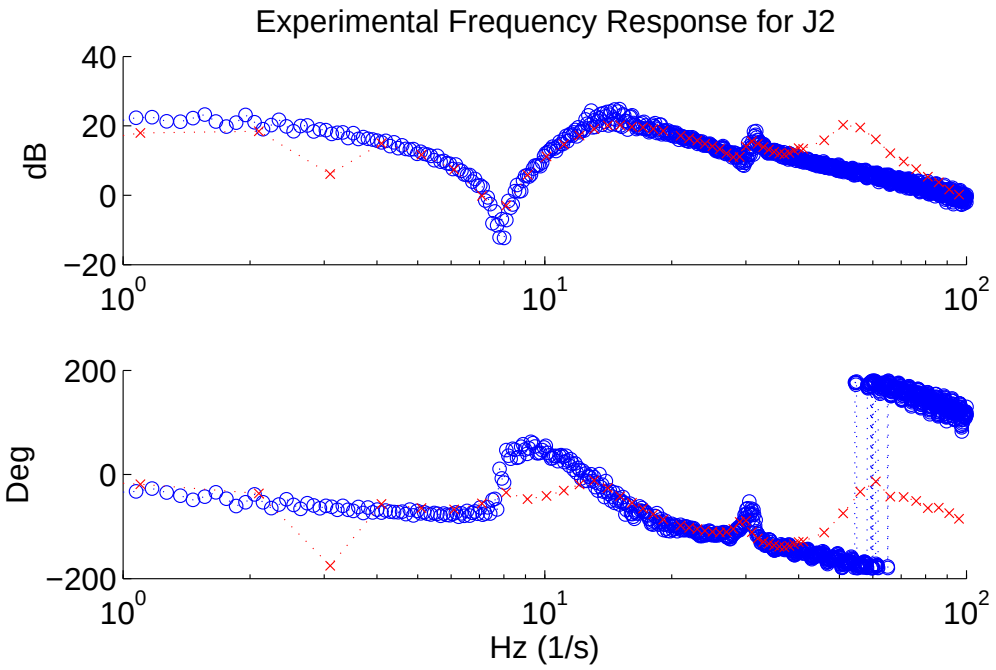


Figure 2.10: System identification result for joint 2: $u_{\text{ref}} \Rightarrow \dot{\theta}_m$ (o: sine sweep results, x: sine by sine results)

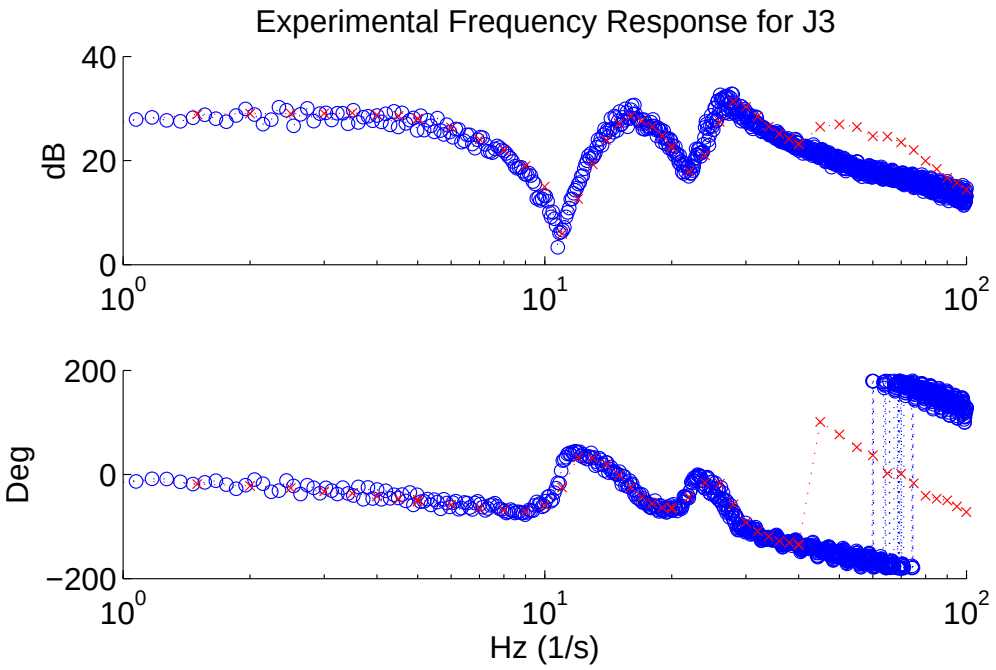


Figure 2.11: System identification result for joint 3: $u_{\text{ref}} \Rightarrow \dot{\theta}_m$ (o: sine sweep results, x: sine by sine results)

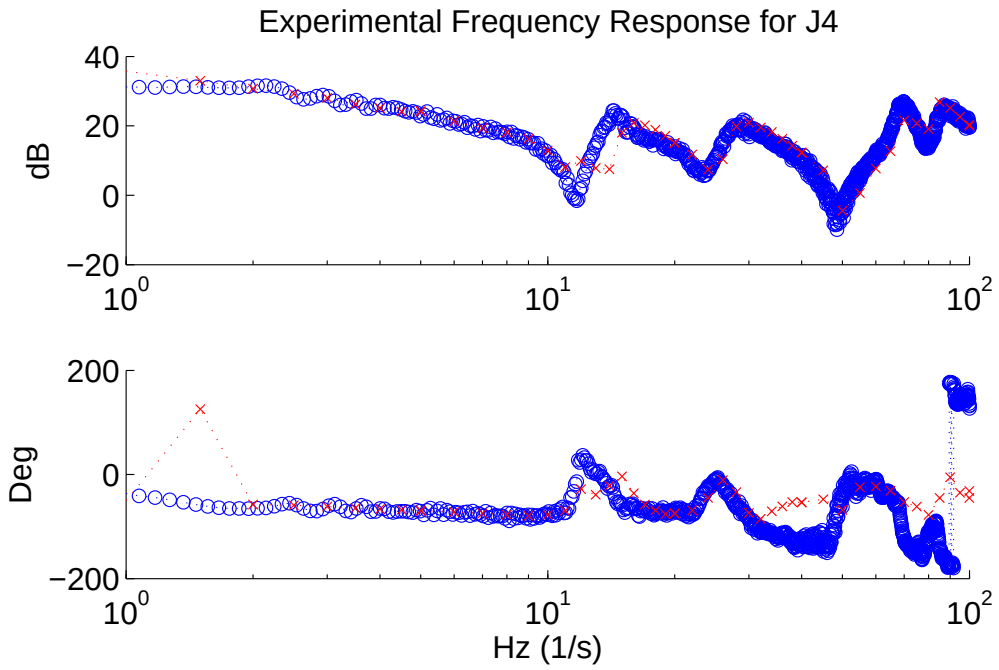


Figure 2.12: System identification result for joint 4: $u_{\text{ref}} \Rightarrow \dot{\theta}_m$ (o: sine sweep results, x: sine by sine results)

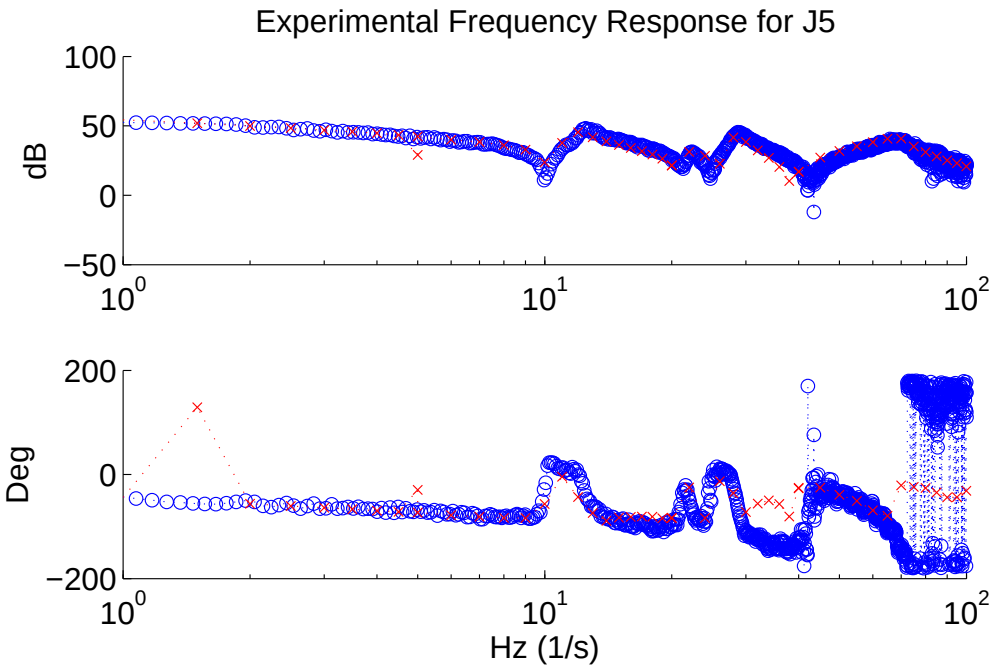


Figure 2.13: System identification result for joint 5: $u_{\text{ref}} \Rightarrow \dot{\theta}_m$ (o: sine sweep results, x: sine by sine results)

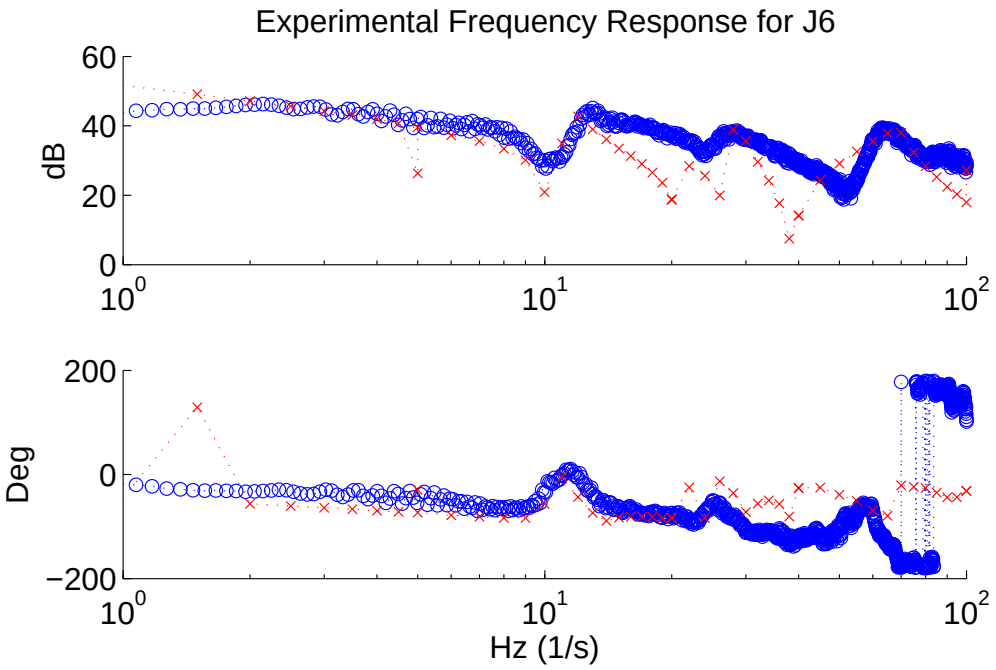


Figure 2.14: System identification result for joint 6: $u_{\text{ref}} \Rightarrow \dot{\theta}_m$ (o: sine sweep results, x: sine by sine results)

Chapter 3

Sensor-based Feedback Controller Tuning of Robot Manipulators by Nonlinear Programming

3.1 Introduction

This chapter presents a method to automatically tune the controller gains for industrial robots. Section 3.2 briefly introduces basic nonlinear programming (NLP) concepts and highlights the math behind the proposed gain tuning algorithm. Section 3.2 outlines how nonlinear programming can be used to tune the controller gain tuning. The chapter then proceeds to introduce barrier functions in section 3.4, which is essential for preserving system stability during the gain tuning process. Stepsize and initial gain selection are then highlighted in section 3.5 and section 3.6 respectively. Section 3.7 presents experimental results demonstrating the utility of the proposed algorithm. The chapter is summarized in section 3.8.

3.2 Nonlinear Programming

NLP is a field in mathematics that focuses on developing algorithms to minimize arbitrary cost functions. The solutions to these optimization processes can be subjected to both equality and inequality constraints. Due to such flexibilities, NLP has already been applied to various applications in control systems; for example, computing the controller gains to minimize the H_∞ norm or identifying unknown parameters through transfer function fitting. Most NLP algorithms minimize cost functions in an iterative manner. More specifically, NLP algorithms adjusts the free parameters at every iteration such that the cost function decreases. There are many different methods for solving NLP problems, such as the subgradient methods, Lagrangian multiplier methods, quasi Newton methods, etc [4, 5, 26].

The majority of these methods, however, rely on a family of descent algorithms known as gradient methods. There are many variations of gradient methods: e.g., the steepest descent method and the Newton method. More specifically, for an arbitrary cost function $J(x_k)$, where subscript k denotes the iteration, it is desirable to update the free variable x_{k+1} such that:

$$J(x_{k+1}) < J(x_k) \quad \forall \ k \tag{3.1}$$

In particular, gradient methods suggests that the free parameter x can be iteratively updated as follows:

$$x_{k+1} = x_k - \alpha_k D_k \nabla J(x_k) \tag{3.2}$$

where α_k and $\nabla J(x_k)$ denote the stepsize and the gradient of the cost function at the k^{th} iterative respectively. D_k is a positive definite matrix that can be used to scale the descent direction to improve the convergence rate of the algorithm. While $\nabla J(x_k)$ depends on the cost function and x_k, α_k and D_k are design parameters for the descent algorithm. Methods such as Newton's method or sequential quadratic programming (SQP) put emphasis on designing α_k and D_k. This dissertation will revisit α_k selection later in this chapter but will redirect interested readers to the cited text above for additional details on how to choose D_k. Unless otherwise specified, D_k will be assumed to be the identity matrix for the remainder of this dissertation.

In this dissertation, NLP will be used to iteratively update the robot feedback controller gains. As a result, the free parameters x_k will be the robot feedback controller gains. For most industrial robotic applications, robots are required to repetitively perform a single task. This makes it intuitive to select each NLP iteration to coincide with a single completion of the robot's desired task. With this said, it is important to choose the cost function such that minimizing such a quantity will also desirably improve the robot performance for the desired task. For example, if tracking performance is only of interest, the cost function can be designed to be a weighted sum of the tracking error norms along the trajectory the robot moves as it performs its task. If vibration suppression is desirable, then the cost function can be designed to prioritize penalizing the acceleration error instead. In many cases, the cost function at each iteration is a function of the robot tracking performance or behavior during that iteration. As such, the cost function is inherently a highly nonlinear function of the free parameters x_k and analytically evaluating the gradient of such a function when utilizing Eq. (3.2) may not be a feasible option. Instead an experimental method can be used to evaluate the cost function gradient.

Gradient Estimation

The method used to experimentally estimate the cost function gradient is a variant of extremum-seeking control originally proposed by [2]. In short, extremum-seeking control is a real-time perturbation technique used for online optimization. Variations of extremum-seeking control has been applied to simple robot systems for online controller tuning [24, 19].

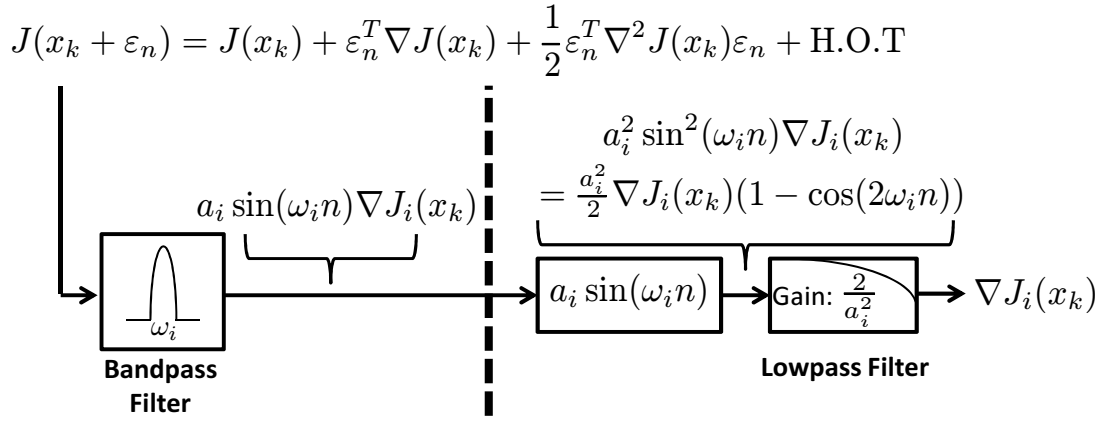


Figure 3.1: Block diagram of procedure to obtain $\nabla J(x_k)$

These applications, however, involved tuning simple robot systems where perturbations in the time domain was possible. In many cases, however, standard robot controllers do not support the ability to perturb the controller gains in real-time. Hence it is more practical and also safer to perturb and update the controller gains in between each robot task iteration instead.

If the cost function $J(x_k)$ is perturbed by the vector ε_k, the resulting cost function can be expressed by a Taylor series expansion as:

$$J(x_k + \varepsilon_n) = J(x_k) + \varepsilon_n^T \nabla J(x_k) + \frac{1}{2}\varepsilon_n^T \nabla^2 J(x_k)\varepsilon_n + \text{ H.O.T} \tag{3.3}$$

where $\nabla^2 J(x_k)$ is the Hessian of the cost function and H.O.T. refers to higher order terms of the Taylor expansion. Suppose the perturbation vector is chosen to be:

$$\varepsilon_n = [a_1 \sin(\omega_1 n) \cdots a_{3J} \sin(\omega_{3J} n)]^T \ \in \ \mathbb{R}^{3J} \tag{3.4}$$

where a_i and ω_i represents the perturbation amplitude and frequency respectively. J denotes the number of joints being tuned. Note that while the indices k and n are not necessarily the same, the perturbations vector, ε_n, is updated in Eq. (3.3) in the iteration domain, not in the time domain. If the perturbation frequencies are chosen to be linearly independent of each other (e.g. $\omega_e \neq b\omega_i + c\omega_j \ \forall \ b, c, e, i, j$) then the perturbation frequencies in Eq. (3.3) are isolated to the first order term, $\varepsilon_n^T \nabla J(x_k)$. Recall that all the terms beyond the gradient contain cross frequency terms, $\sin(\omega_i)\sin(\omega_j)$ or $\sin(\omega_i)\sin(\omega_i)$), which have frequency content, $|\omega_i \pm \omega_j|$ and $|\omega_i \pm \omega_i|$ respectively. This allows the i^{th} element of the gradient to be partially isolated by passing the sequence of $\{J(x_k + \varepsilon_n)\}$ through a bandpass filter with a narrow pass band centered about ω_i. This process is depicted in the left half of Fig. 3.1.

The bandpass filter used in this dissertation have transfer functions in the form:

$$p_i(z) = \frac{b_{0,i}(z - 1)}{z^2 + a_{1,i}z + a_{0,i}} \tag{3.5}$$

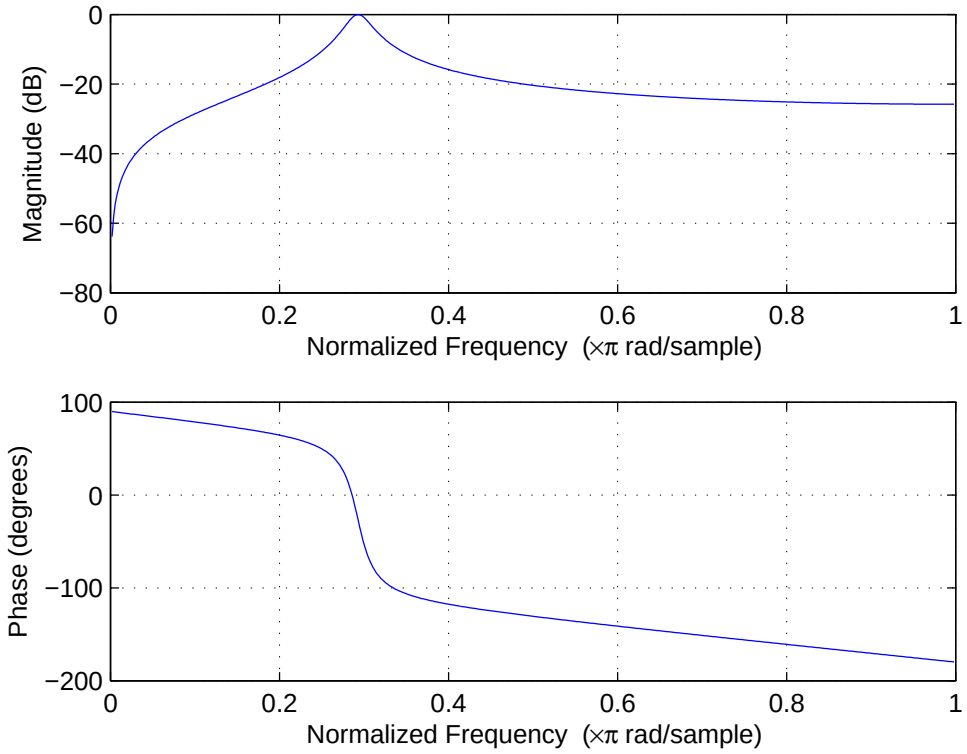


Figure 3.2: Frequency response of bandpass filter centered at 0.29π with $b = 0.1$

where

$$\begin{aligned} a_{0,i} &= e^{-b\omega_i} \\ a_{1,i} &= -2e^{-0.5b\omega_i}\cos\left(\sqrt{1-\left(\tfrac{b}{2}\right)^2}\omega_i\right) \\ b_{0,i} &= \left|\frac{e^{j2\omega_i}+a_{1,i}e^{j\omega_i}+a_{0,i}}{e^{j\omega_i}-1}\right| \end{aligned}$$

Note that for a given ω_i, b is the only design variable for the bandpass filters. The magnitude of b is inversely proportional to the width of the filter pass band. This dissertation selects $b = 0.1$. The frequency response of such a filter is show in Fig. 3.2. Note that the filter design provide zero phase shift at the pass band.

Assuming one of the perturbation frequencies occur at ω_i, the output of Eq. (3.3) after passing through a bandpass filter centered at ω_i is nominally:

$$a_i \sin(\omega_i n)\nabla J_i(x_k) \tag{3.6}$$

Note that in actual implementation, the bandpass filter will only attenuate the content at other frequency ranges and not completely eliminate them. The nominal expression given

by Eq. (3.6), however, is reasonably accurate if the perturbation frequencies are sufficiently spaced apart. At this point, if Eq. (3.6) is modulated by its perturbation value, that is:

$$\begin{aligned} a_i \sin(\omega_i n)\nabla J_i(x_k) \cdot a_i \sin(\omega_i n) &= a_i^2 \sin(\omega_i n)^2 \nabla J_i(x_k) \\ &= a_i^2 \nabla J_i(x_k) \tfrac{1}{2}\left(1 - \cos(2\omega_i n)\right) \end{aligned} \tag{3.7}$$

then the i^{th} term of the gradient can be isolated by passing Eq. (3.7) through a lowpass filter whose DC gain is $\frac{2}{a_i^2}$ and cutoff frequency is lower than two times the lowest perturbation frequency. Note that there is a one step delay induced by the bandpass filter given by Eq. (3.5). As a result, this delay should be taken into consideration prior to modulating Eq. (3.6). This modulating and filtering process is depicted by the right half of Fig. 3.1. Once this process is done for each perturbation frequency, the cost function gradient can then be reassembled by concatenating the processed results together. As a note, one inherent assumption made in this section is that the gain tuning environment is time-invariant or slowly time-varying (i.e. the dominant factor that influences the value of the cost function should be the controller gain perturbations). Consequently, this assumption also mandates that the nonlinearities of the system, such as Coulomb friction in the gear train, and other disturbances remain relatively consistent between iterations. Otherwise, the time varying nature of the system will invalidate Eq. (3.3). However, this condition is easily satisfied during gain-tuning processes of robots when they repeat the same tasks. Furthermore, if each tuning iteration is selected to be a single completion of the robot's task, then the nonlinearities experienced by the robot for each iteration should remain relatively constant as well.

3.3 Controller Tuning Process

The controller tuning process is relatively simple. Once the desired robot trajectory, cost function, and initial gains are selected, the robot can then be commanded to iteratively follow the desired robot trajectory. The desired controller gains can be perturbed according to Eq. (3.4) after the completion of every robot trajectory while simultaneously computing and storing the cost function value at every iteration. More specifically, the index n is the number of iterations that the robot has traversed its desired trajectory. Once enough iterations have been performed to capture a few perturbation periods, the sequence of cost function values can be processed using the process depicted in Fig. 3.1 to obtain the cost function gradient about the set of initial gains. More specially, if the robot performs N iterations of its desired trajectory, the calculated cost function sequence can be represented as:

$$\bar{J}(x_k, \varepsilon_n)_N = \{ \; J(x_k + \varepsilon_{n+1}) \quad J(x_k + \varepsilon_{n+2}) \quad \cdots \quad J(x_k + \varepsilon_{n+N}) \; \} \; \in \; \mathbb{R}^N \tag{3.8}$$

Once $\bar{J}(x_k, \varepsilon_n)_N$ is obtained, the filtering process outlined in Fig. 3.1 can be used to obtain $\nabla J(x_k)$. At this point, any gradient descent method, whose form is given by Eq. (3.2),

can be used to update the controller gains x_k. Once the desired gains have been updated, this process can be repeated to obtain the next cost function gradient to further update the controller gains.

Note that the controller gain index, k, is incremented once after every N iterations of the robot performing its task. This can lead to a time consuming gain tuning process, especially if the robot training trajectory is chosen to be fairly long. One method to expedite this tuning process is to approximate $\bar{J}(x_{k+1}, \varepsilon_n)_N$ with:

$$\bar{J}(x_{k+1}, \varepsilon_n)_N \approx \hat{J}(x_{k+1}, \varepsilon_n)_N = \begin{bmatrix} J(x_k + \varepsilon_{n+2}) & \cdots & J(x_k + \varepsilon_{n+N}) & J(x_{k+1} + \varepsilon_{n+N+1}) \end{bmatrix} \tag{3.9}$$

Essentially the last $N-1$ element of $\bar{J}(x_k, \varepsilon_n)_N$ is used to approximate the first $N-1$ elements of $\bar{J}(x_{k+1}, \varepsilon_n)_N$. This approximation is remarkably accurate as long as the stepsize, α_k, used for updating the gains is sufficiently small. This stepsize assumption does not hinder the gain tuning process since in practice, the controller gains should be updated in small increments anyways to preserve system stability. Additionally, the first order parameter update law given in Eq. (3.2) also requires the parameter updates to be sufficiently small such that the cost function values decreases after each parameter update. In this dissertation, the experimental results will utilize the approximation in Eq. (3.9) to expedite the gain tuning process.

3.4 Barrier Functions

An inherent weakness of using NLP methods for tuning physical systems is that NLP methods are strictly numerical in nature and does not take into consideration physical limitations of these system. More specifically, when using NLP algorithms for gain tuning, it is intuitive that increasing these gains will make the performance better. The NLP algorithm, however, does not consider the possibility that increasing the gains can also make the system unstable. For feedback gain tuning, it is absolutely critical that system stability is preserved in the process. Because stability is a binary condition, the robot seldom exhibit measurable symptoms of instability during tuning prior to becoming instable. Hence some modifications needs to be made to the NLP formulation to account for this issue.

Up until now, the gain tuning problem, when phrased as an optimization problem, is simply an unconstrained optimization problem of the form:

$$\min \;\; J(x_k)$$

Whereas to preserve stability, the problem should be rephrased as a constrained optimization problem of form:

$$\begin{aligned} \min \;\; & J(x_k) \\ \text{s.t.} \;\; & x_k \;\in\; X_s \end{aligned}$$

where X_s denotes the subspace of all stabilizing controller gains for the robot. Since it is normally very difficult to produce an analytical expression for X_s, In this dissertation, the two mass model given by Eq. (2.7) is used to assess the closed loop poles of each joint to account for system stability. By doing this, the optimization problem can be reformulated as:

$$\begin{array}{c} \min \ \ J(x_k) \\ \text{s.t. } \mathbb{R}\{c_{j,n}\} \ < \ 0 \ \ \forall j,n \end{array} \tag{3.10}$$

where $\mathbb{R}\{c_{j,n}\}$ is the real part of the n^{th} pole of the closed loop transfer function for the j^{th} joint. Rather than trying to solve a constrained optimization problem, the constraints can be incorporated into the cost function as barrier functions. More specifically, rather than solving Eq. (3.10), one can solve:

$$\min \ \ J(x_k) + B(x_k) \tag{3.11}$$

$B(x_k)$ is called a barrier function and is constructed such that these functions become arbitrarily large if any one of the inequality constraints (i.e. $\mathbb{R}\{c_{j,n}\} < 0$) is about to violated. Common choices for $B(x_k)$ in Eq. (3.11) are:

$$\begin{aligned} B_1(x_k) &= -\sum_j \sum_n \ln\{-\zeta_{j,n}\mathbb{R}\{c_{j,n}\}\} \\ B_2(x_k) &= -\sum_j \sum_n \frac{-\zeta_{j,n}}{\mathbb{R}\{c_{j,n}\}} \end{aligned} \tag{3.12}$$

where $\zeta_{j,n}$ serves as a scaling factor. Both function in Eq. (3.12) are plotted in Fig. 3.3. Note that both examples become arbitrarily large as $\mathbb{R}\{c_{j,n}\}$ approach 0 from below. A fundamental assumption when using barrier methods is that the barrier functions are never violated when updating the free parameters x_k. This criteria can be satisfied by properly choosing the stepsize. Stepsize selection criteria will be discussed in a later subsection of this chapter.

On a technical note, in order for the solution of Eq. (3.11) to be exactly the same as the original formulation in Eq. (3.10), one would have to iteratively solve Eq. (3.11) with the coefficients $\zeta_{j,n}$ decreasing in each iteration. The optimal solution $x_k^*(\zeta_{j,n})$ as each $\zeta_{j,n}$ is iteratively reduced to zero follows a trajectory known as the *centralpath* along the solution space. The limit point of these solutions is the solution to Eq. (3.10) [4]. For the purpose of gain tuning, however, Eq. (3.11) is not solved iteratively, but instead is treating as its own optimization problem. Hence the optimal solution for the feedback controller gains using the barrier method may not be same as the one obtained by solving Eq. (3.10). This is not a problem in this application as long as the gain tuning algorithm can continue to improve the robot performance. This can be guaranteed by initially setting each $x_k^*(\zeta_{j,n})$ such that $J(x_1) \gg B(x_1)$ based on the initial values for x_1. This way, any substantial decrease of the combined cost function is caused by improved performance and not by the barrier functions.

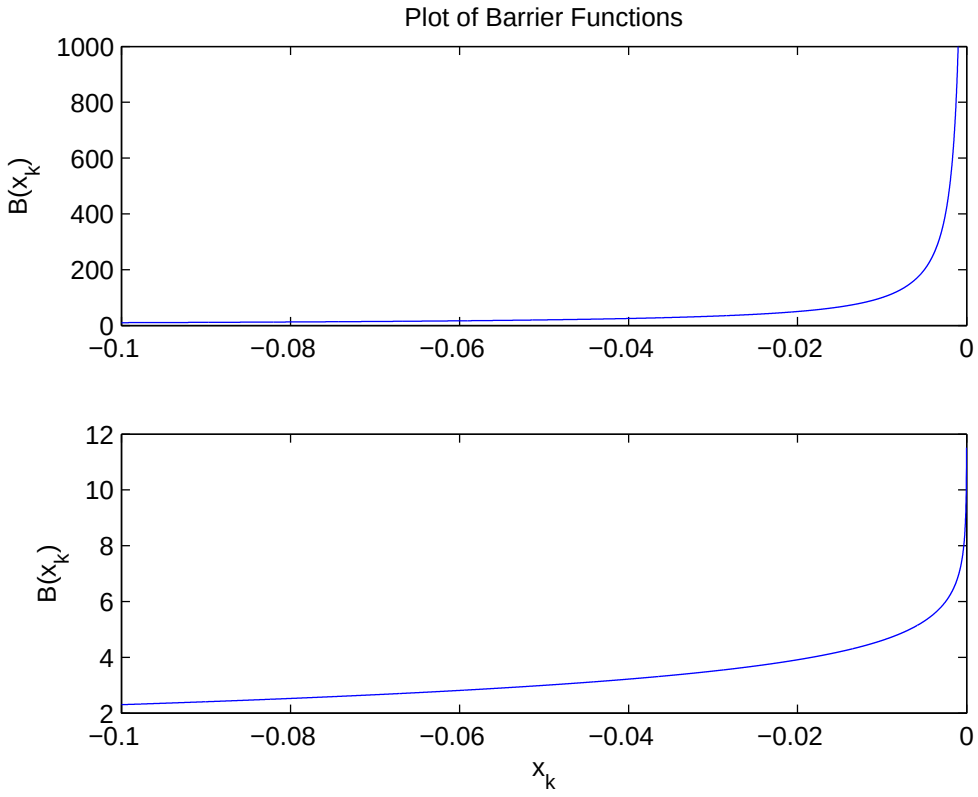


Figure 3.3: Plots of $B_1(x_k)$ and $B_2(x_k)$ on the top and bottom respectively

Barrier Functions Gradients

Using the gradient estimation procedure outlined in section 3.3, one inherent assumption is that the cost function changes slowly from one iteration to another. Barrier functions, however, do not satisfy this condition as they can grow exponentially. Hence, the proposed gradient estimation technique may not be accurate if it is used to empirically estimate the gradient of Eq. (3.11). This is especially problematic since it implies that the gradient estimate becomes inaccurate when the system is on the verge of instability. One approach to avoid the aforementioned problem is to numerically calculate the barrier function gradients separately from the gradient of the cost function, $J(x_k)$. To simplify the gradient calculation, the barrier function for each robot joint in Eq. (3.12) can be reformulated as

$$\begin{array}{l} \max \\ s = -\sigma_o + j\omega \\ \forall\, \omega \in (-\infty, \infty) \end{array} \quad \frac{\gamma}{||1 + L(s)||_2^2} \tag{3.13}$$

where $L(s)$ is the open loop transfer function and σ_o is a positive tolerance term. Eq. (3.13) essentially performs a search along a vertical line in the complex plane whose real part is given by $-\sigma_o$. Eq. (3.13) will grow rapidly as the closed loop poles approach $-\sigma_o$. Like all barrier methods, this approach assumes that initially all the real components of the closed loop poles are less than $-\sigma_o$ and updating the controller gains do not cause the poles to cross $-\sigma_o$. The first assumption can be checked by properly selecting the initial controller gains. The second assumption can be satisfied through careful stepsize selection. The value of s in Eq. (3.13) can be found through a numerical search using commercial software such as MATLAB. Assuming $s = \sigma_o + j\omega_o$ is known, the gradient of the barrier function can be computed . To simplify the algebra, consider the following notation

$$\begin{aligned}
s &= \sigma_o + \omega_o j \\
s^2 &= (\sigma_o^2 - \omega_o^2) + (2\omega_o\sigma_o)j \\
&= \phi_1 + \phi_2 j \\
s^3 &= (\sigma_o\phi_1 - \omega_o\phi_2) + (\omega_o\phi_1 + \sigma_o\phi_2)j
\end{aligned}$$

$$\begin{aligned}
G_{dmu}(s) &= \tfrac{\dot{\theta}_m(s)}{u(s)} = \tfrac{J_l s^2 + d_j s + k_j}{J_m J_l s^3 + J_d s^2 + J_k s + k_j d_m} \\
&= \tfrac{\beta_2 2 s^2 + \beta_1 s + \beta_o}{\alpha_3 s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_o} \\
&= \tfrac{(\beta_2\phi_1 + \beta_1\sigma_o + \beta_o) + (\beta_2\phi_2 + \beta_1\omega_o)j}{(\alpha_3(\sigma_o\phi_1 - \omega_o\phi_2) + \alpha_2\phi_1 + \alpha_1\sigma_o + \alpha_o) + (\alpha_3(\omega_o\phi_1 + \sigma_o\phi_2) + \alpha_2\phi_2 + \alpha_1\omega_o)j} \\
&= \tfrac{\gamma_1 + \gamma_2 j}{\delta_1 + \delta_2 j}
\end{aligned}$$

$$L(s) = \tfrac{\gamma_1 + \gamma_2 j}{\delta_1 + \delta_2 j} \tfrac{K_v s^2 + (K_v K_p + K_i)s + K_p K_i}{s^2}$$

where $j = \sqrt{-1}$. The expressions above breaks the complex values into their corresponding real and imaginary parts. This is convenient since it allows Eq. (3.13) to be expressed as

$$\tfrac{1}{||1 + L(\sigma_o + j\omega_o)||^2} = \tfrac{1}{w^T w} \tag{3.14}$$

where $w = [\Re\mathrm{e}\{1 + L(\sigma_o + j\omega_o)\} \quad \Im\mathrm{m}\{1 + L(\sigma_o + j\omega_o)\}]^T$. From (3.14), the gradient of the barrier function can be expressed as

$$\frac{\partial}{\partial x_k} \frac{1}{||1 + L(\sigma_o + j\omega_o)||^2} = \frac{\partial w}{\partial x_k} \frac{-2w}{(w^T w)^2} \tag{3.15}$$

Since the current application of the gain tuning algorithm focuses on tuning the feedback controller gains, the free variables for the NLP are the feedback controller parameters. More

specifically, $x_k = [K_p \quad K_v \quad K_i]^T$. To evaluate $\frac{\partial w}{\partial x_k} \in \mathbb{R}^{3\times 2}$, consider the following notation

$$\begin{aligned}
\tfrac{\gamma_1+\gamma_2 j}{(\delta_1+\delta_2 j)s^2} &= \tfrac{\{\phi_1(\gamma_1\delta_1+\gamma_2\delta_2)+\phi_2(\gamma_2\delta_1-\gamma_1\delta_2)\}+\{\phi_1(\gamma_2\delta_1-\gamma_1\delta_2)-\phi_2(\gamma_1\delta_1+\gamma_2\delta_2)\}j}{||\delta_1+\delta_2 j||^2\,||\phi_1+\phi_2 j||^2} \\
&= \tfrac{\theta_1+\theta_2 j}{||\delta_1+\delta_2 j||^2\,||\phi_1+\phi_2 j||^2} \\
L(\sigma_o+j\omega_o) &= \tfrac{\theta_1+\theta_2 j}{||\delta_1+\delta_2 j||^2\,||\phi_1+\phi_2 j||^2}\{K_v s^2+(K_vK_p+K_i)s+K_pK_i\} \\
\Re\mathrm{e}\{1+L(\sigma_o+j\omega_o)\} &= 1+\tfrac{\theta_1(K_v\phi_1+(K_vK_p+K_i)\sigma_o+K_pK_i)-\theta_2(K_v\phi_2+(K_vK_p+K_i)\omega_o)}{||\delta_1+\delta_2 j||^2\,||\phi_1+\phi_2 j||^2} \\
\Im\mathrm{m}\{1+L(\sigma_o+j\omega_o)\} &= \tfrac{\theta_2(K_v\phi_1+(K_vK_p+K_i)\sigma_o+K_pK_i)+\theta_1(K_v\phi_2+(K_vK_p+K_i)\omega_o)}{||\delta_1+\delta_2 j||^2\,||\phi_1+\phi_2 j||^2}
\end{aligned}$$

The first and second columns of $\frac{\partial w}{\partial x_k}$ are $\frac{\partial}{\partial x_k}\Re\mathrm{e}\{1+L(\sigma_o+j\omega_o)\}$ and $\frac{\partial}{\partial x_k}\Im\mathrm{m}\{1+L(\sigma_o+j\omega_o)\}$ respectively. Evaluating these partial derivatives yield

$$\begin{aligned}
\tfrac{\partial}{\partial x_k}\Re\mathrm{e}\{1+L(\sigma_o+j\omega_o)\} &= \tfrac{1}{||\delta_1+\delta_2 j||^2\,||\phi_1+\phi_2 j||^2}\begin{bmatrix} \theta_1(\sigma_oK_v+K_i)-\theta_2\omega_oK_v \\ \theta_1(\phi_1+\sigma_oK_p)-\theta_2(\phi_2+K_p\omega_o) \\ \theta_1(\sigma_o+K_p)-\theta_2\omega_o \end{bmatrix} \\
\tfrac{\partial}{\partial x_k}\Im\mathrm{m}\{1+L(\sigma_o+j\omega_o)\} &= \tfrac{1}{||\delta_1+\delta_2 j||^2\,||\phi_1+\phi_2 j||^2}\begin{bmatrix} \theta_2(\sigma_oK_v+K_i)+\theta_1\omega_oK_v \\ \theta_2(\phi_1+\sigma_oK_p)+\theta_1(\phi_2+K_p\omega_o) \\ \theta_2(\sigma_o+K_p)+\theta_1\omega_o \end{bmatrix}
\end{aligned}$$

Hence, $\frac{\partial}{\partial x_k}\frac{1}{||1+L(\sigma_o+j\omega_o)||^2} = \frac{\partial w}{\partial x_k}\frac{-2w}{(w^Tw)^2}$ can be evaluated using the equations listed above.

3.5 Stepsize Selection

As mentioned earlier, the selection of the stepsize, α_k, needs to ensure that updating the controller gains will not violate the barrier constraints. If the barrier functions are accurately defined, then enforcing the barrier functions will also guarantee system stability. Many traditional stepsize selection techniques require knowledge of how the stepsize affects the updated cost function, $J(x_{k+1})$. Examples of these techniques include the Goldstein rule and the Armijo rule. In this particular application, the cost function $J(x_k)$ does not have a closed form expression and can only be obtained through experimental measurements. Hence it is not possible to utilize many of these existing stepsize selection techniques. While the cost function itself cannot be computed without performing an experiment on the robot, the barrier functions can be. Hence a successive stepsize reduction algorithm is used to continually reduce the stepsize until the updated gains satisfy the barrier functions.

In general, the stepsize, α_k, can be represented more generally as

$$\alpha_k = \alpha_{0,k}\alpha_r^{n_k} \tag{3.16}$$

where $\alpha_{0,k}$ is the initial stepsize at iteration k, $\alpha_r < 1$ is a stepsize reduction ratio, and n_k is the stepsize reduction factor at iteration k. Intuitively, if the initial stepsize violates the barrier constraints at a particular iteration, n_k can be increased to decrease the stepsize. For the purpose of automatic gain tuning, it is important that the stepsize is selected to both improve system performance while simultaneously enforcing system stability. In this dissertation the selection of $\alpha_{0,k}$ is used to improve performance while the selection of n_k is used to satisfy the second requirement. Given that the gains are being updated using steepest descent, a first order approximation shows that any positive value for $\alpha_{0,k}$ is guaranteed to decrease the cost function. The caveat to this statement, however, is that it is only true for $\alpha_{0,k}$ small enough such that the first order effects are dominant. Since "small" also depends heavily on the nonlinearity of the cost function, it is usually very difficult to quantify a good value for $\alpha_{0,k}$ if the cost function is unknown. This dissertation uses a fixed value for $\alpha_{0,k}$ for every iteration. The selected value is manually tuned based on experimental data. Additional refinement of $\alpha_{0,k}$ is a topic of future work. Calculating n_k, however, is relatively straight forward since the barrier functions can be numerically evaluated prior to running the experiment. Hence one can check whether or not updating the gains will violate the barrier constraints. In this dissertation, n_k is selected using the logic tree presented in Fig. 3.4.

3.6 Initial Condition Selection

Due to the nonlinear nature of the robot system, and inherently of the cost function $J(x_k)$, NLP techniques can only arrive at locally optimal solutions rather than globally optimal solutions. Hence selecting the appropriate initial feedback controller gains prior to tuning can strongly affect the performance of the gain tuning algorithm. In particular for an industrial robot, tuning the three wrist joints (J4-J6) is particularly difficult as these joints tend to be very sensitive to controller parameter variations. Hence it is especially important to correctly select the initial gains for these values. This subsection will provide a theoretical way for selecting initial gains to satisfy certain performance criterion. For practicality, the work in this subsection will focus on selecting gains for the three wrist joints of the FANUC M-16iB robot. Two gain selection approaches will be given. The first is a standard approach that is commonly used in industry, whereas the second is a slightly more complicated approach which gives consideration to system stability and unmodeled dynamics.

Standard Initial Gain Selection Method

The feedback controller structure for a single wrist joint of the FANUC M-16iB is depicted in Fig. 3.5. Note that Fig. 3.5 uses motor velocity, $\dot{\theta}_m$, for motor side feedback. Note that

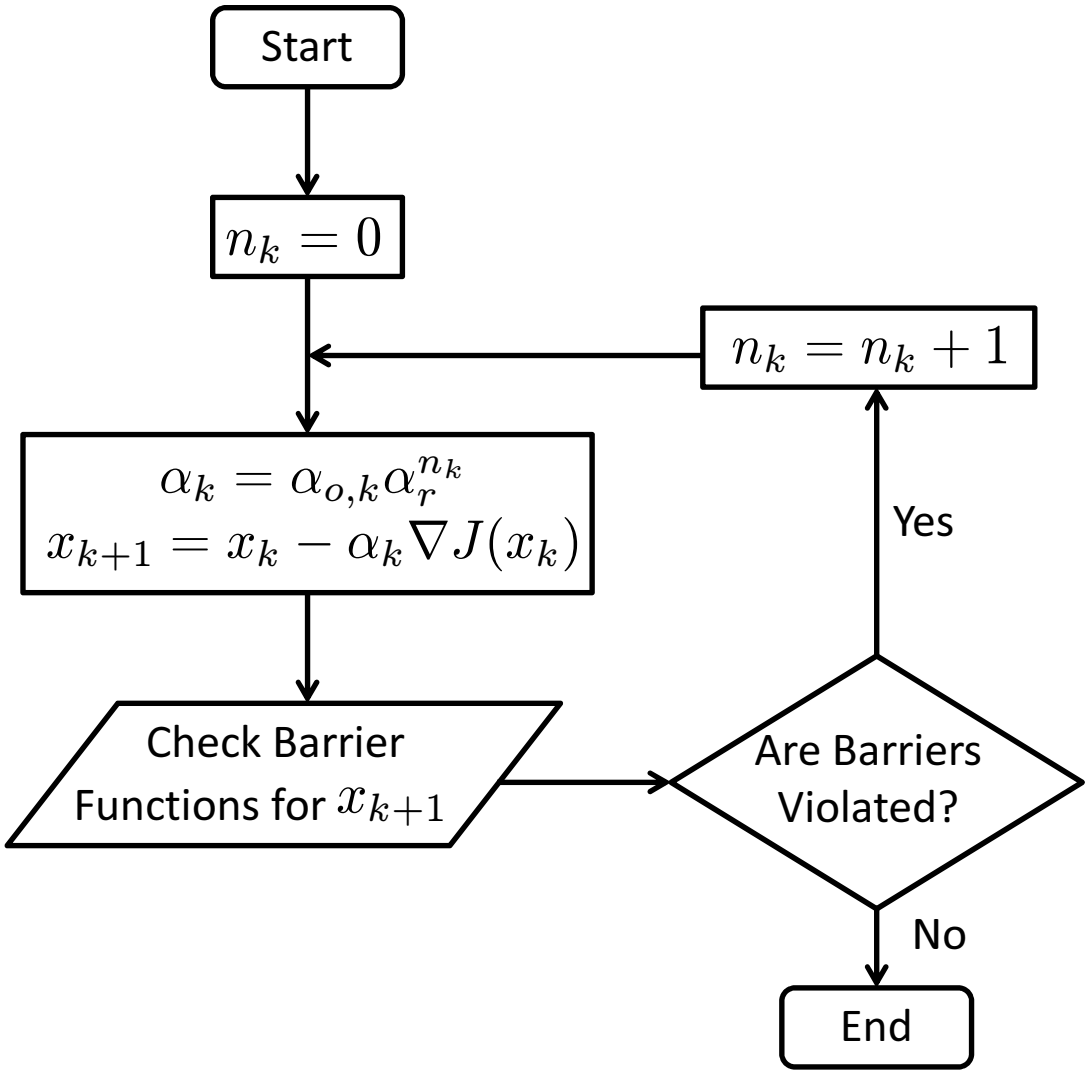


Figure 3.4: Logic tree for selecting n_k

since the load side inertias of the wrist joints are relatively small, the reducer dynamics for these three joints can be effectively ignored. Using this assumption, the wrist joint can be approximated as an inertia system with viscous damping. This allows Eq. (2.7) to be simplified as a first order system of form:

$$G_{dmu}(s) = \frac{\dot{\theta}_m(s)}{u(s)} \approx \hat{G}_{dmu}(s) = \frac{1}{Js + D} \tag{3.17}$$

where the lumped inertia and damping are given by $J = J_m + \frac{J_l}{N^2}$ and $D = d_m$ respectively.

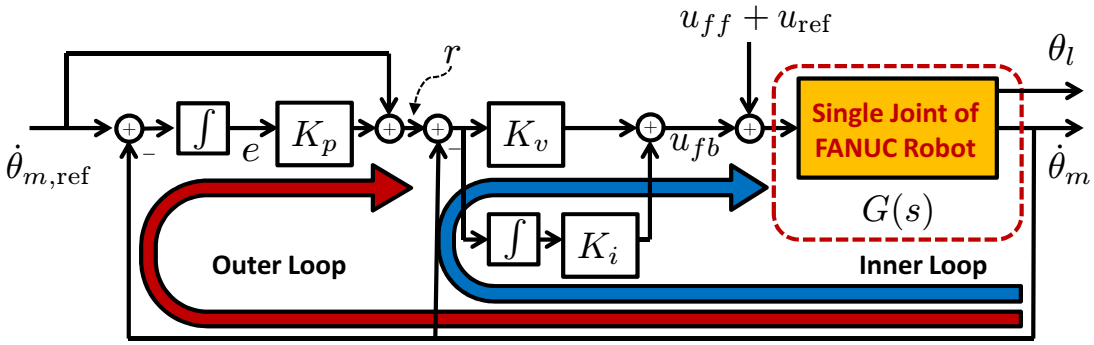


Figure 3.5: Feedback controller structure for an individual joint

Additionally, also note that the controller structure can be broken down into two separate loops: an inner velocity loop and an outer position loop. This structure can simplify the tuning process as the bandwidth of the inner and outer loop can be tuned separately. Performing a closed loop analysis of the inner loop from r to $\dot{\theta}_m$ gives the following transfer function:

$$\frac{\dot{\theta}_m}{r} = \frac{K_v s + K_i}{Js^2 + (K_v + D)\, s + K_i} \tag{3.18}$$

Based on this observation, it would be ideal if the inner loop behaved like a first order closed loop system of form:

$$\frac{\dot{\theta}_m}{r} = \frac{1}{\delta s + 1} \tag{3.19}$$

This can be achieved by setting $K_v = J/\delta$ and $K_i = D/\delta$. Once this is done, tuning of the outer loop is performed while assuming that the inner loop behaves like Eq. (3.19). The proportional gain, K_p, is tuned such that the outer loop has the desired bandwidth. Note that this method works if the outer loop bandwidth is significantly less than that of the inner loop bandwidth. While this method is simple to apply, it does not give consideration to any unmodeled system dynamics. Furthermore, there is no guarantee that these initial gains can even stabilize the system.

Alternate Initial Gain Selection Method

The bandwidth of a control loop is often used in practice as a measure of performance. The alternate method for initial gain selection also aims at satisfying bandwidth requirements, but also give considerations to stability and unmodeled dynamics. In the aforementioned tuning method, the inner loop and outer loop are tuned separately. When tuning the inner loop, however, it is inherently assumed that the plant behaves strictly like a first order damped inertia system. More specifically, this process assumes the plant to behave like this ideal first order system up to the bandwidth specified by the time constant δ, which can in practice be

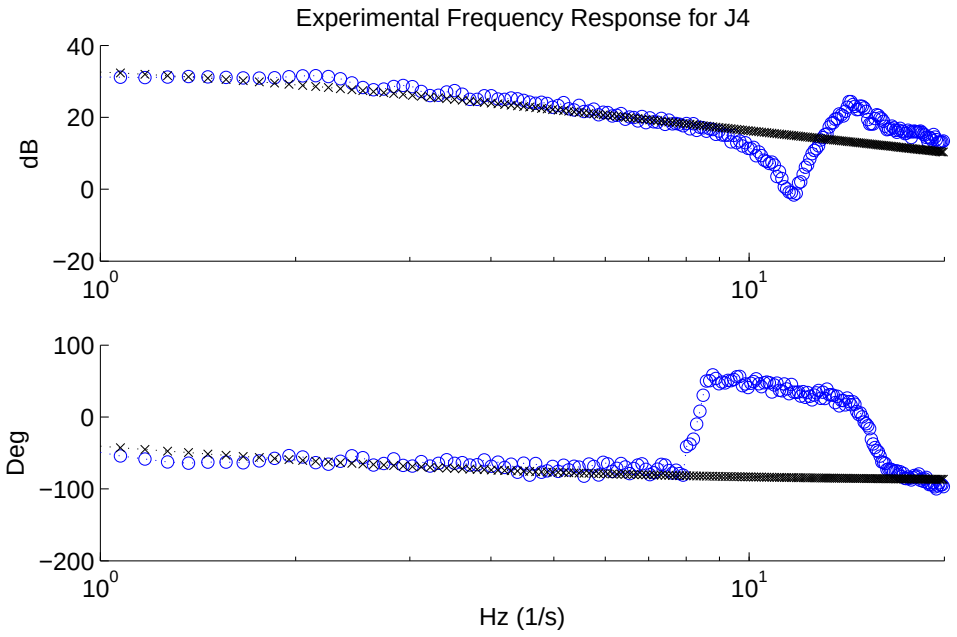


Figure 3.6: System identification result for joint 4 (Blue circles indicate the measured frequency response from a sine sweep, black crosses indicates the response of an ideal first order model)

as high as 20 Hz. In actuality, however, this assumption may not hold, and depending on the actual dynamics of the plant, setting the values of K_v and K_i carelessly can cause the inner velocity loop to be unstable. Rather than making assumptions about the behavior of the plant, another approach is to use the system identification data to better characterize the actual dynamics of the plant. The frequency response of J4-J6 are shown in Figs. 3.6-3.8. The blue circles indicate the measured system response from a sine sweep identification experiment whereas the black crosses indicate the frequency response of an ideal first order damped inertia model given by Eq. (3.17). The inertia parameters, J_m and J_l, used for the calculations of the damped inertia model was provided by FANUC Corporation. The motor side viscous damping parameter, d_m, was experimentally verified by Chen in [8]. Note that Figs. 3.6-3.8 suggest that the first order approximation is accurate at low frequencies, but fails to capture the behavior of the system at higher frequencies. Notably, the first order model is accurate up to about 10 Hz.

When tuning the bandwidth of the two feedback loops in Fig. 3.5, one thing to note is that the only bandwidth that ultimately matters for actual performance is the outer loop bandwidth. As a result, the inner loop bandwidth does not necessarily matter as long as the outer loop maintains its desired bandwidth.

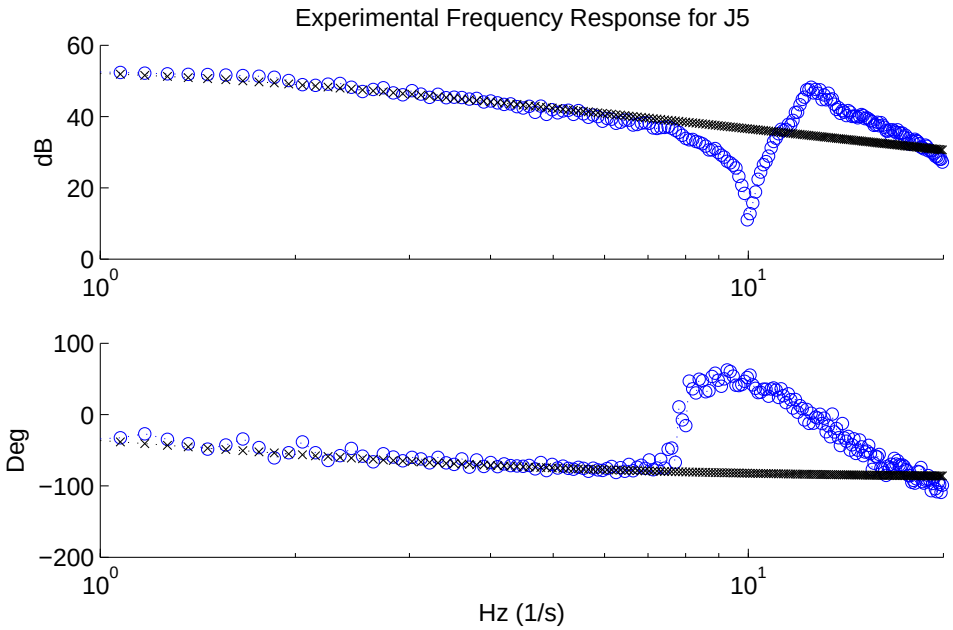


Figure 3.7: System identification result for joint 5 (Blue circles indicate the measured frequency response from a sine sweep, black crosses indicates the response of an ideal first order model)

Inner Loop Bandwidth Analysis

In the standard approach, the inner loop is tuned to behave like the first order system given by Eq. (3.19). Eq. (3.18) shows that the inner loop is naturally second order. While forcing it to behave like a first order makes it simple for tuning the outer loop, this approach may not be very robust to modeling uncertainties. A more straightforward and robust approach is to directly analyze the bandwidth of the second order system. This section will detail the bandwidth analysis for a second order system in the form given by Eq. (3.18). The bandwidth of Eq. (3.18) is defined as the lowest frequency, ω in which:

$$\left|\frac{\dot{\theta}_m}{r}\right|_{s=j\omega} = \left|\frac{\frac{K_i}{J}}{s^2 + \frac{(K_v+D)}{J}s + \frac{K_i}{J}} \cdot \frac{K_v s + K_i}{K_i}\right|_{s=j\omega} < \frac{1}{\sqrt{2}} \tag{3.20}$$

Rewriting the second order term in Eq. (3.20) as $\frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$ and then squaring both sides of Eq. (3.20) yields:

$$\frac{\omega_n^4}{K_i^2} \cdot \frac{K_v^2\omega^2 + K_i^2}{(\omega_n^2 - \omega^2)^2 + (2\zeta\omega_n\omega)^2} < \frac{1}{2} \tag{3.21}$$

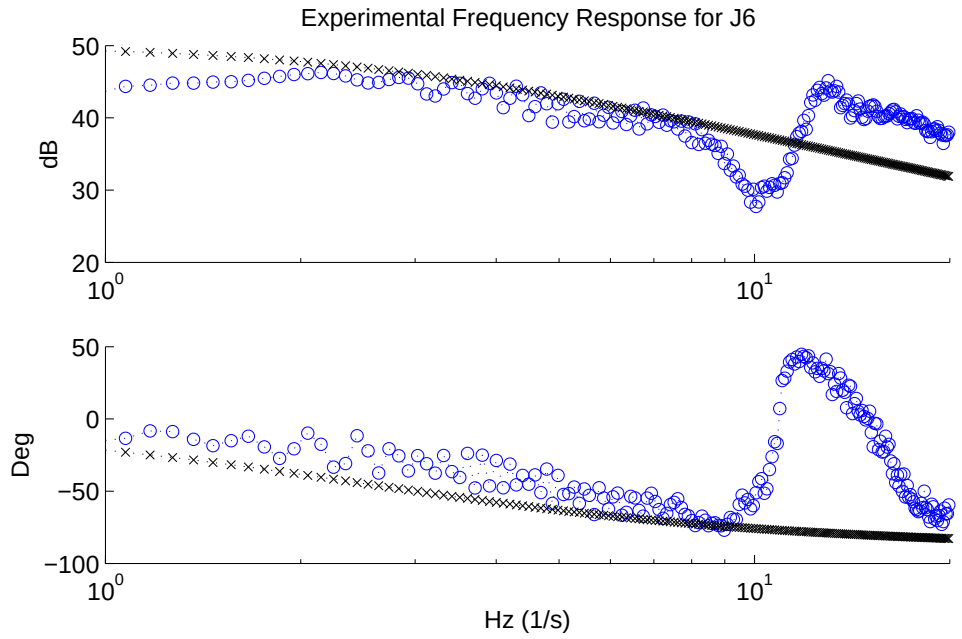


Figure 3.8: System identification result for joint 6 (Blue circles indicate the measured frequency response from a sine sweep, black crosses indicates the response of an ideal first order model)

where $\zeta = \frac{(K_v+D)}{2\sqrt{K_i J}}$ and $\omega_n^2 = K_i/J$. After some algebraic manipulation and also setting the inequality to an equality, the solution to ω in Eq. (3.21) can be written as:

$$\omega^2 = \frac{K_v^2+JK_i-2JK_i\zeta^2}{J^2} \pm \frac{\left[4\zeta^4K_i^2J^2-4\zeta^2K_i^2J^2-4\zeta^2K_v^2K_iJ+2JK_v^2K_i+K_v^4+2J^2K_i^2\right]^{1/2}}{J^2} \tag{3.22}$$

Solving Eq. (3.22) for the smallest positive value of ω will yield the bandwidth of the inner loop for given controller gains K_i and K_v. Upon careful observation of Eq. (3.22) one can show that the term under the square root is larger than the leading term, hence the solution of interest can be expressed as:

$$\omega^2 = \frac{K_v^2+JK_i-2JK_i\zeta^2}{J^2} + \frac{\left[4\zeta^4K_i^2J^2-4\zeta^2K_i^2J^2-4\zeta^2K_v^2K_iJ+2JK_v^2K_i+K_v^4+2J^2K_i^2\right]^{1/2}}{J^2} \tag{3.23}$$

To prevent overshoot in the system response, it is desirable to set the value $\zeta = 1$. By doing so, K_i becomes a function of K_v through the following relationship:

$$K_i = \frac{(K_v + D)^2}{4J} \tag{3.24}$$

Substituting this relationship into Eq. (3.23) yields the following expression:

$$\begin{gathered}\frac{1}{16}K_v^4+\frac{D}{4}K_v^3+\left(\frac{3D^2}{8}+\frac{3J^2\omega^2}{2}\right)K_v^2+\left(\frac{D^3}{4}-DJ^2\omega^2\right)K_v+\\ \left(\frac{D^4}{16}-J^4\omega^4-\frac{J^2\omega^2D^2}{2}\right)=0\end{gathered} \tag{3.25}$$

Once the desired bandwidth, ω, is fixed by the designer, a numeric solver can be used to solve Eq. (3.25) for K_v. Once K_v is known, Eq. (3.24) can be used to solve for K_i.

Outer Loop Bandwidth Analysis

Once the inner loop is tuned, as long as the bandwidth of the inner loop is substantially higher than the desired bandwidth of the outer loop, the the inner loop can be treated as an unit gain when doing the outer loop analysis. At low frequencies, the outer loop transfer function effectively becomes:

$$\frac{\dot{\theta}_m}{\dot{\theta}_{md}}=\frac{K_p}{s+K_p} \tag{3.26}$$

Hence K_p can be designed to shape the outer loop bandwidth.

Final Initial Controller Gain Adjustments

Once the initial controller gains are calculated using the analysis approach outlined in the previous two sections, the final step is to lower K_v as much as possible while still maintaining the desired outer loop bandwidth. Recall that decreasing K_v will decrease the inner loop bandwidth but it may not affect the outer loop bandwidth. This final step is to ensure that the initial gains do not cause the mechanical system to become unstable.

3.7 Experimental Results

In this section, the M-16iB is used to verify the proposed gain tuning algorithm. In this dissertation, a quadratic cost function that penalizes the tracking position error and velocity error is used. Since the built in robot motor encoders can readily provide motor position and velocity error information, the motor side error is used when computing the cost function. Using the motor side error is sufficient for demonstrating the feasibility of the proposed approach, but in practice, the cost function should penalize the load side errors. More specifically, the cost function used in the dissertation has form:

$$J(x_k)=\sum_{t=0}^{T}\{e_{m,k}^T(x_k,t)W_1(t)e_{m,k}(x_k,t)+\dot{e}_{m,k}^T(x_k,t)W_2(t)\dot{e}_{m,k}(x_k,t)\} \tag{3.27}$$

where t is a time index, $e_{m,k}(t)$ and $\dot{e}_{m,k}(t)$ is the motor side position error and velocity error vector respectively. $W_1(t)$ and $W_2(t)$ are the weighting matrices for the motor side position and velocity errors respectively. Additionally, the barrier function parameters σ_o and γ were manually tuned for each joint so that the initial contributions from the barrier functions to the entire cost function were an order of magnitude less than the contributions from the tracking errors $e_m(t)$ and $\dot{e}_m(t)$. The initial stepsize, $\alpha_{o,k}$, and step size reduction ratio, α_r, were both selected to be 0.5. The perturbation amplitudes for each controller gain was selected to be 10% of their initial value. And finally, 65 iterations were used for the gain tuning process.

Single Joint Tuning Experiments

This section presents the results from the single joint tuning experiments on the M-16iB robot. The perturbation frequencies used for K_p, K_v, and K_i were 0.293π, 0.467π, and 0.313π respectively for each joint. Since each joint is tuned individually, the weighting matrices W_1 and W_2 reduce to scalars. Values of 1 and 0.3 were used for W_1 and W_2 respectively. In these tuning experiments, each joint of the robot was instructed to sweep a 22.5 degree arc over one second and then sweep back. The initial and final controller gains for each joint is tabulated in Table 3.1.

Figures. 3.9-3.14 plot the cost function versus iteration for each joint. Note that in some cases, it requires up to 30 iterations before the cost function begins to substantially decrease. This can be caused by many factors, such as the initial gains starting off in a local plateau in the map between the cost function and controller gains. Since the gains are perturbed from iteration to iteration, the effects of these perturbations can been in the cost function. Since the sensitivity of the cost function to these perturbations vary from one set of gains to another, it may cause the cost function to decrease nonmonotonicly. Also note that for $J1$, $J2$, $J3$ and $J5$, the contributions of the barrier functions to the overall cost function seem minimal. This is to be expected if the tuning process does not encounter any gains that may destabilize the system. For some joints, in particular $J4$ and $J6$, the contributions from the barrier functions are not negligible. Both $J4$ and $J6$ were found to be extremely sensitive to their velocity gain K_v. While increasing K_v improves performance, increasing it beyond a certain value would abruptly cause the joint to chatter significantly. As a result, the barrier function parameter γ was set to be more conservative for these two joints to preserve overall system stability. This effect can be seen in Fig. 3.14 after 30 iterations and to a lesser extent in Fig. 3.12. Even with these nuances, however, the gain tuning method still decreased the performance-based portion of the cost function (dashed lines), thus resulting in improved tracking performance of the robot manipulator.

Table 3.1: Parameters for gain tuning

	Parameters					
Joint	$K_{p,1}$	$K_{v,1}$	$K_{i,1}$	$K_{p,65}$	$K_{v,65}$	$K_{i,65}$
$J1$	10	0.2300	0.1000	10.293	0.3493	0.1257
$J2$	10	0.2000	0.1000	10.030	0.2250	0.1310
$J3$	10	0.0507	0.0100	9.991	0.1010	0.0954
$J4$	20	0.0560	0.2000	19.998	0.0622	0.2010
$J5$	15	0.0020	0.0020	15.000	0.0025	0.0024
$J6$	20	0.0040	0.0040	19.999	0.0069	0.0035

3.8 Chapter Summary

This chapter introduced an automatic gain tuning algorithm using nonlinear programming. The proposed gain tuning procedure was presented and barrier functions were introduced to help preserve system stability during tuning. Additionally, this chapter also introduced techniques for selecting the stepsize and initial gains when utilizing this tuning algorithm on industrial robots. And finally, experimental results were presented to verify the utility of the proposed approach.

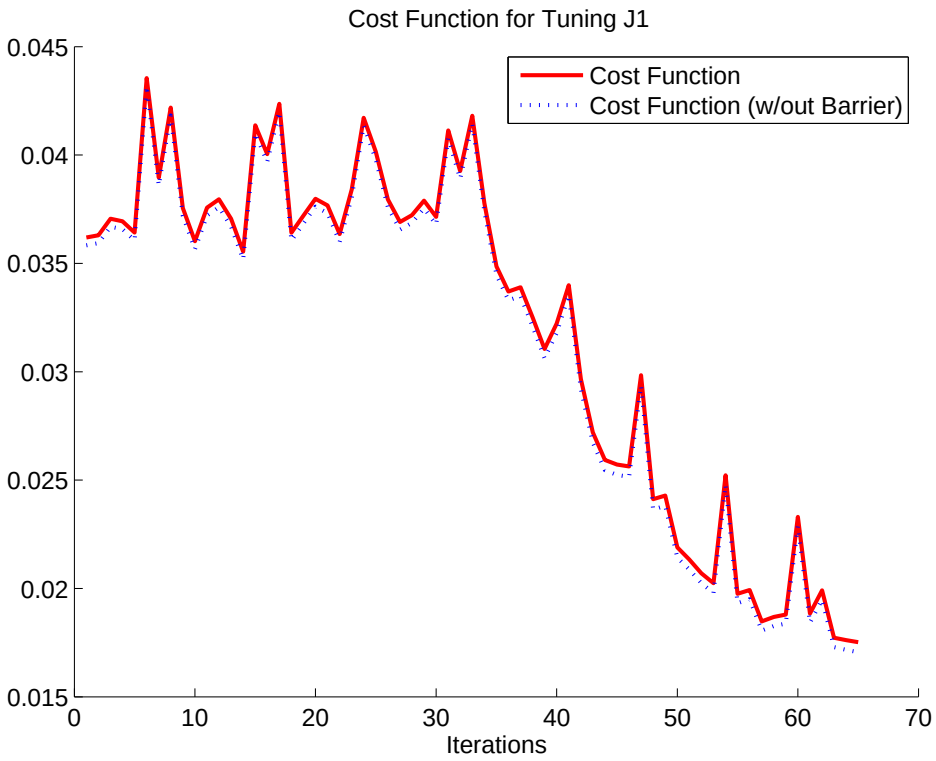


Figure 3.9: Cost function versus iteration while tuning $J1$ of the FANUC M-16iB robot

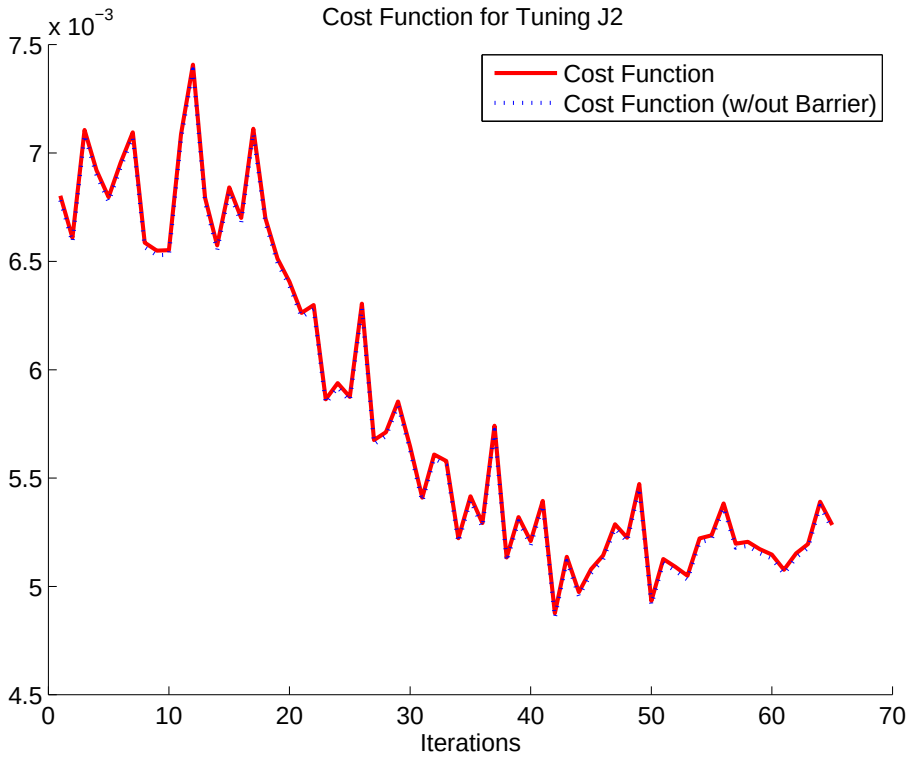


Figure 3.10: Cost function versus iteration while tuning $J2$ of the FANUC M-16iB robot

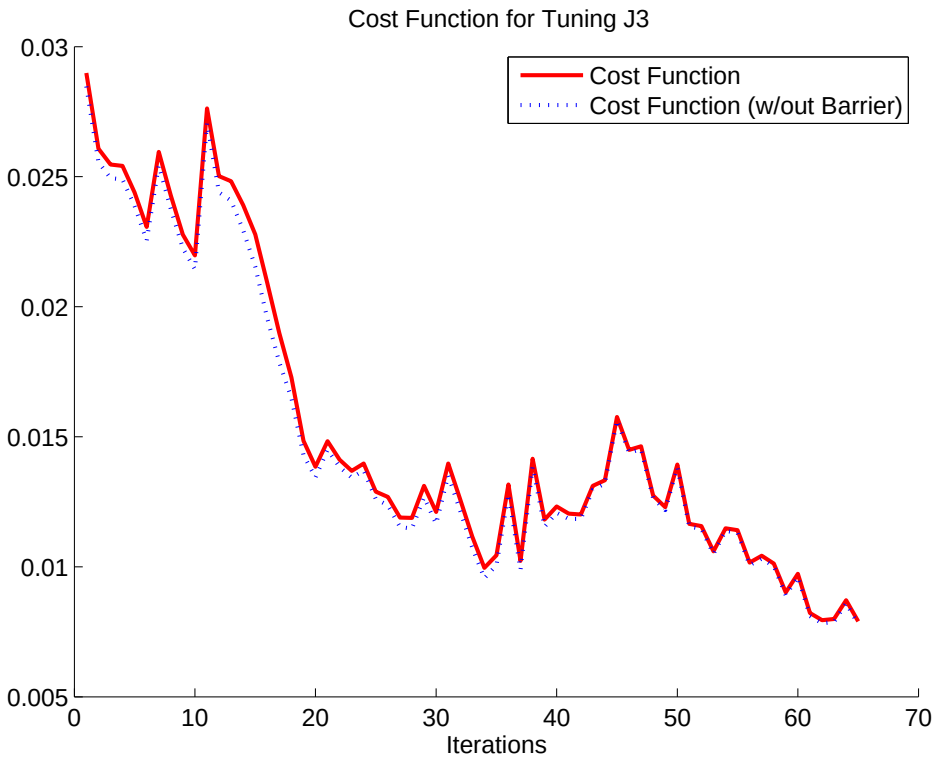


Figure 3.11: Cost function versus iteration while tuning $J3$ of the FANUC M-16iB robot

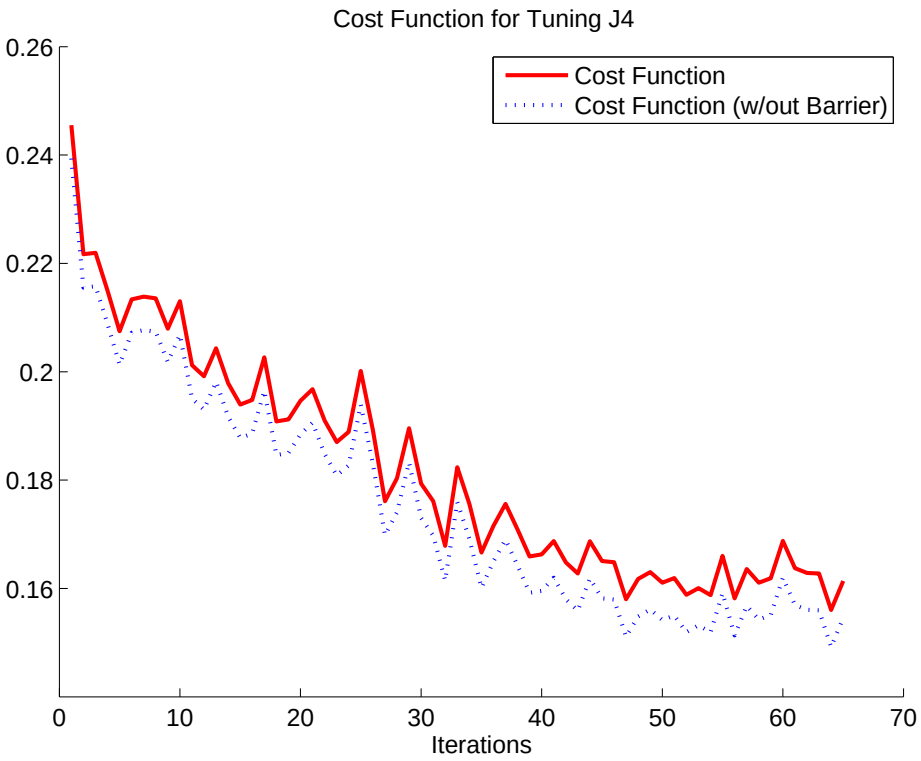


Figure 3.12: Cost function versus iteration while tuning $J4$ of the FANUC M-16iB robot

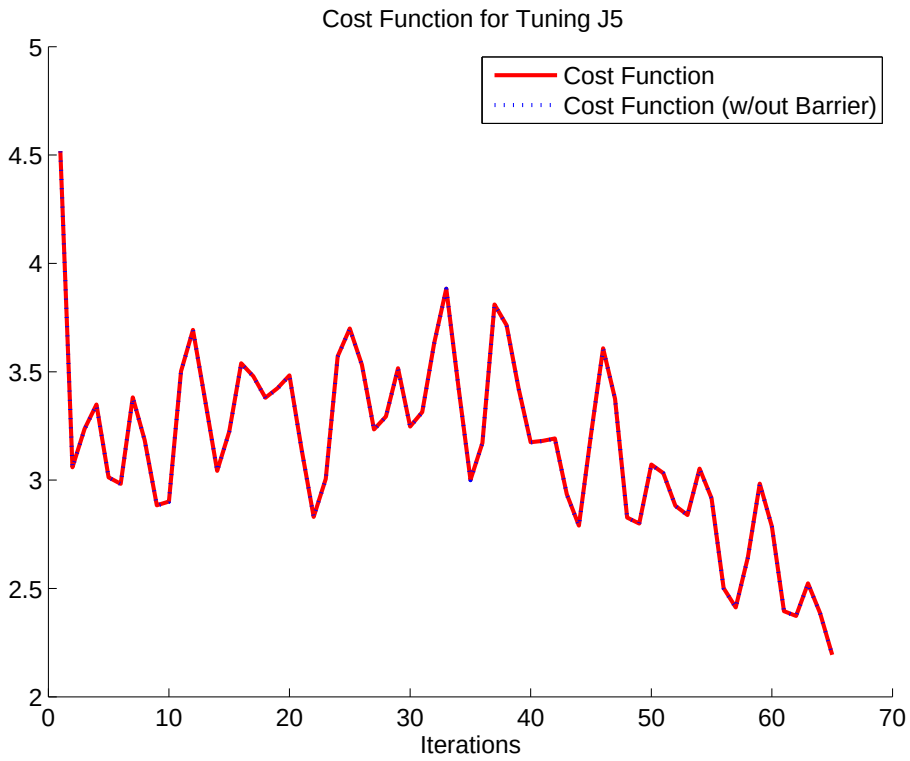


Figure 3.13: Cost function versus iteration while tuning $J5$ of the FANUC M-16iB robot

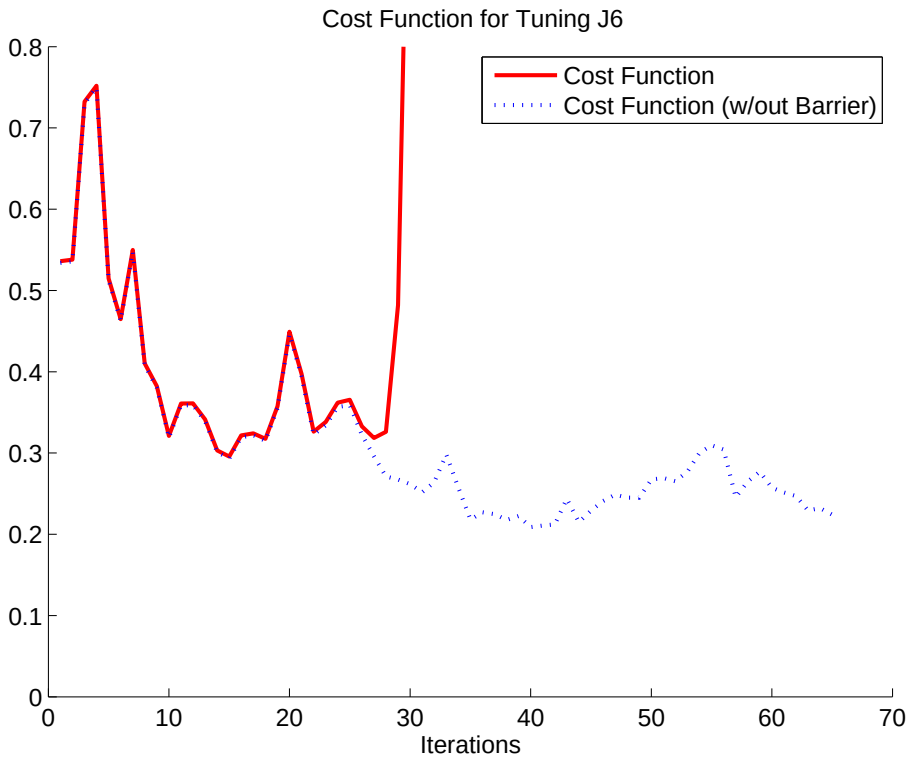


Figure 3.14: Cost function versus iteration while tuning $J6$ of the FANUC M-16iB robot

Chapter 4

A Disturbance Observer Framework for Stable Feedback Controller Tuning

4.1 Introduction

This chapter introduces a disturbance observer based framework for stable feedback controller tuning. When tuning high order controllers with a plant whose behavior is not fully known, feedback controller tuning always run the risk of destabilizing the system. Barring a few mild restrictions, this chapter presents a tuning framework that allows one to arbitrarily tune a subset of the feedback controller parameters while simultaneously preserving system stability. This chapter begins by briefly reviewing the traditional disturbance observer in section 4.2 before introducing the framework in section 4.3. The motivation and proof for the framework is also presented. Both simulation and experimental verification of the framework are organized in section 4.4. Finally, the contents of the chapter is summarized in section 4.5.

4.2 Traditional Disturbance Observer

The structure of a typical disturbance observer (DOB) is show in Fig. 4.1. In the figure, $G_a(s)$ is the phyiscal plant, $G_n^{-1}(s)$ is the stable nominal inverse of the plant, and $Q(s)$ is called the Q-filter and is to be designed. $U^*(s)$, $D(s)$, $Y(s)$, and $N(s)$ are the command input, disturbance input, plant output, and the measurement noise respectively. The disturbance observer has several beneficial properties, namely the transfer function $U^*(s)$ to $Y(s)$ is given by:

$$G_{yu^*}(s) = \frac{G_a(s)G_n(s)}{G_n(s) + Q(s)\left(G_a(s) - G_n(s)\right)} \tag{4.1}$$

If $Q(s) = 1$, then $G_{yu^*}(s) = G_n(s)$. Similarly, if $Q(s) = 0$, then $G_{yu^*}(s) = G_a(s)$. This implies that the DOB structure will force the input/output relationship from $U^*(s)$ to $Y(s)$ to behave like the nominal model within the pass band of the Q-filter, and to behave like

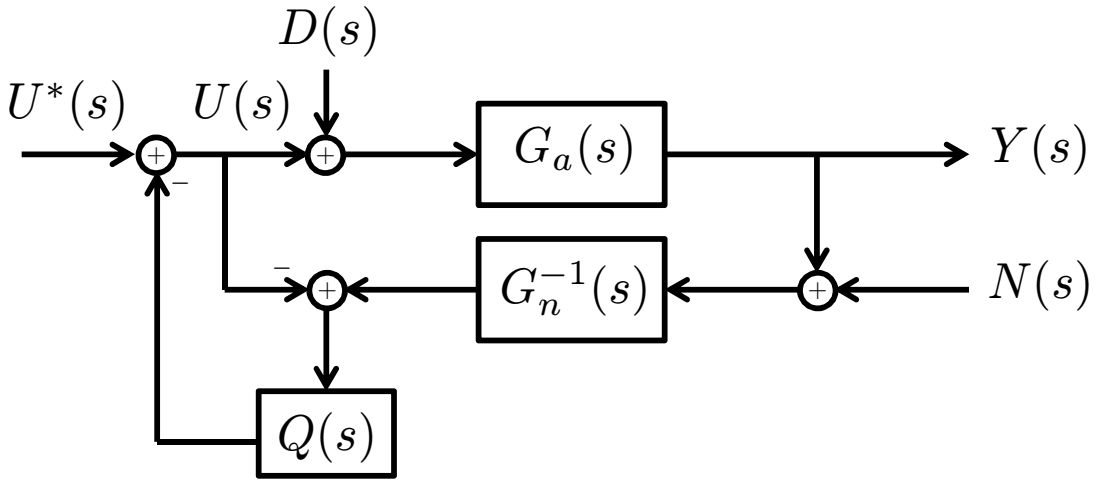


Figure 4.1: Typical DOB structure

$G_a(s)$ in the stop band of the Q-filter. Additionally, the DOB structure has modest stability robustness properties, especially if $G_n(s)$ closely matches $G_a(s)$ within the pass band of the Q-filter. Traditionally, the Q-filter is designed as a lowpass filter whose bandwidth is limited by the error between $G_n(s)$ and $G_a(s)$.

4.3 Disturbance Observer Tuning Framework

In most controller applications applications, the controllers are tuned to provide good system performance up to a predesignated bandwidth. While increasing the controller gains will improve the system performance within the desired bandwidth, recklessly increasing the gains may also cause the controller to excite resonances and other high frequency dynamics of the physical system. This in effect will deteriorate the overall system performance and in the worst case destabilize the system. Many established controller tuning methods such as the Ziegler-Nichols method operate by effectively giving practical guidelines for balancing the trade off between good low frequency performance and overall system stability.

So if the desired system bandwidth is known in advance, it would be ideal if tuning the feedback controller would just affect the closed loop system performance within this desired closed-loop bandwidth. A naive implementation of this idea can be done with the parallel controller structure shown in Fig. 4.2. $C(s)$ is a fixed stabilizing controller for plant $G(s)$. $C_t(s)$ is the controller being tuned and $\hat{Q}(s)$ is a lowpass filter whose cutoff frequency dictates the desired bandwidth of the tuned controller. $R(s)$, $E(s)$, $U(s)$, and $Y(s)$ are the reference signal, error, plant input, and plant output respectively. The equivalent controller in Fig. 4.2 can be expressed as:

$$C_{\text{eq}}(s) = \left(1 - \hat{Q}(s)\right) C(s) + \hat{Q}(s) C_t(s) \tag{4.2}$$

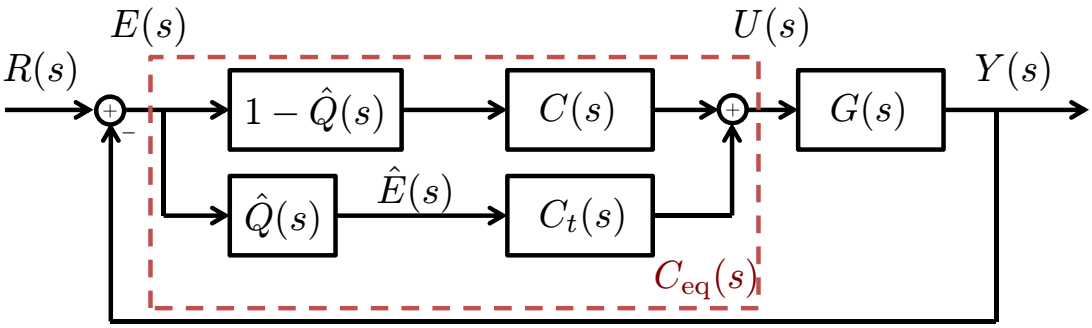


Figure 4.2: Parallel controller structure

Roughly speaking, $C_{\text{eq}}(s)$ behaves like $C_t(s)$ in the passband of $\hat{Q}(s)$ and behaves like $C(s)$ otherwise. This implementation has several key advantages when it comes to gain tuning. Normally, engineers can use a nominal model, $\hat{G}(s)$, to model the physical plant $G(s)$. Due to both computational and modeling limitations, these nominal models capture the behavior of $G(s)$ well at low frequencies, but are unable to capture the higher frequency dynamics. Additionally, most physical plants tend to exhibit nonlinear amplitude dependent behavior at higher frequencies. For these reasons, typical feedback controller gains are limited. By using a framework similar to that shown in Fig. 4.2, the controller gains in $C_t(s)$ can be tuned without experiencing the aforementioned limitations. While it is easy to draw intuitive statements about the performance of the structure in Fig. 4.2, rigorously proving the internal system stability for such a structure, however, turns out to be very difficult.

Motivation

Recalling the input/output relationship of the DOB in Eq. (4.1), one may note that this transfer function has similar characteristics as Eq. (4.2). Namely, both structures allow the combined transfer function to behave like one transfer function within a certain bandwidth and behave like another independent transfer function outside of that bandwidth. Most importantly, these two transfer functions are completely separated in the block diagram, hence one can be adjusted without affecting the other. Using Figs. 4.1-4.2 as inspiration, the DOB tuning framework is shown in Fig. 4.3.

Like before, $C(s)$ is a fixed and stabilizing controller for the plant $G(s)$. $C_t^{-1}(s)$ is the inverse of the controller being tuned and $Q(s)$ is the Q-filter that is to be designed by the user. Using this framework, the relationship from the error $E^*(s)$ to $U(s)$ is given by:

$$C_{ue^*}(s) = \frac{C(s)C_t(s)}{C_t(s) + Q(s)\left(C(s) - C_t(s)\right)} \tag{4.3}$$

Similar to the behavior in Eq. (4.1), $C_{ue^*}(s)$ behaves like $C_t(s)$ when $Q(s) = 1$ and like $C(s)$ when $Q(s) = 0$. This is important, since it allows $C_t(s)$ to only affect particular frequency

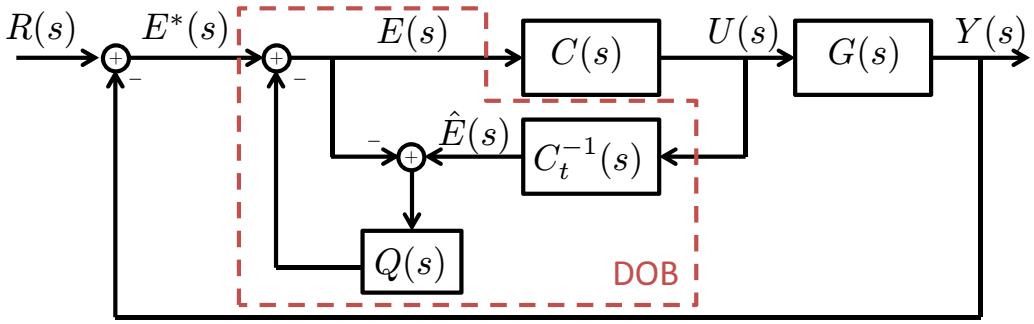


Figure 4.3: Proposed DOB framework for controller tuning

bands of the closed loop behavior. By designing $Q(s)$ as a lowpass filter for example, will limit the gain tuning to only affect low frequency behavior. This keeps the $C_t(s)$ from exciting high frequency dynamics in $G(s)$, which is the primary cause of instability during feedback controller tuning.

Stability Analysis

Given the proposed DOB framework, it is important to analyze the internal stability of the entire closed loop system when varying the parameters in $C_t(s)$. Again, since $C_t^{-1}(s)$ is the only transfer function that varies during the gain tuning process, it is reasonable to begin the stability analysis by isolating $C_t^{-1}(s)$ from the rest of the system. Doing this results in the block diagram shown in Fig. 4.3.

Once in this form, the small gain theorem can be used to analyze the internal stability of the structure. The specific form of the small gain theorem used in this dissertation is

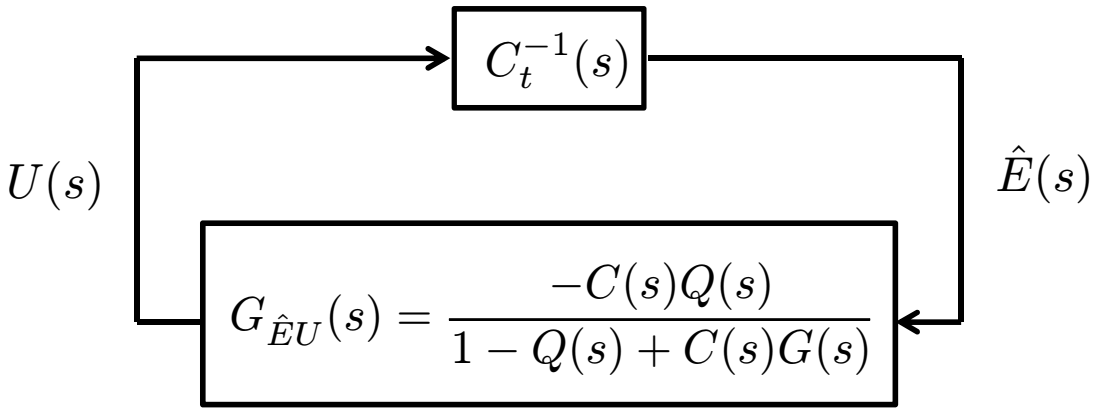


Figure 4.4: Equivalent representation of Fig. 4.3 for stability analysis

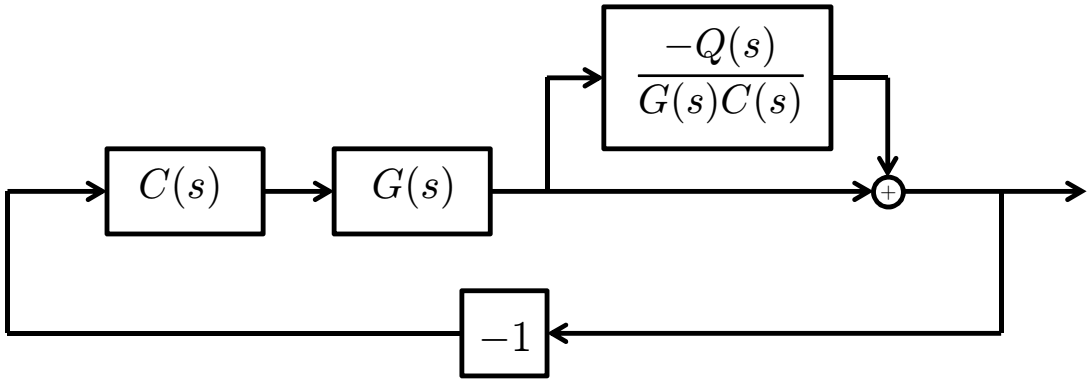


Figure 4.5: Block diagram representation of $\hat{G}_{\hat{E}U}$ as a structured uncertainty problem

formally stated as follows:

Theorem 1. *If two systems $S_1(s)$ and $S_2(s)$ are connected in a positive feedback loop, then the closed loop system is internally stable if both $S_1(s)$ and $S_2(s)$ are internally stable and if $\|S_1(s)S_2(s)\|_\infty < 1$.*

Remark The general small gain theorem was derived by Zames and the proof will not be repeated here [41]. The specific form of the small gain theorem used in this dissertation is the form commonly used to analyze stability of systems with bounded uncertainty and often used in conjunction with H_∞ controller synthesis.

Based on the small gain theorem, the first two conditions to check are the internal stability conditions of $C_t^{-1}(s)$ and $G_{\hat{E}U}(s)$. Since $C_t^{-1}(s)$ is a controller that is actively being tuned, its stability can be easily enforced while tuning. The internal stability of $G_{\hat{E}U}(s)$ on the other hand, is harder to analyze. Rather than directly analyzing the stability of $G_{\hat{E}U}(s)$, it is more convenient to look at the stability of $\hat{G}_{\hat{E}U}(s)$, which is defined as:

$$\hat{G}_{\hat{E}U}(s) = \frac{C(s)G(s) - Q(s)}{1 + C(s)G(s) - Q(s)} \tag{4.4}$$

Note that $\hat{G}_{\hat{E}U}(s)$ and $G_{\hat{E}U}(s)$ both have the same characteristic equation, hence showing that $\hat{G}_{\hat{E}U}(s)$ is internally stable is the same as showing that $G_{\hat{E}U}(s)$ is internally stable. It is convenient to consider $\hat{G}_{\hat{E}U}(s)$ because it can be interpreted as a closed loop system with controller $C(s)$ and plant $G(s)$ where the plant has a structured multiplicative uncertainty given by $\dfrac{-Q(s)}{G(s)C(s)}$. This feedback system is depicted in Fig. 4.5. The following theorem is helpful for interpreting the stability conditions of such a system:

Theorem 2. *Given a plant with multiplicative structured uncertainty defined as* $G_{\text{unc}}(s) = G(s)\,(1 + W_u(s)\Delta(s))$ *where* $G(s)$ *is the nominal plant representation as a rational transfer function,* $W_u(s)$ *is the structured uncertainty modeled as a stable transfer function, and* $\Delta(s)$ *is any transfer function such that* $\|\Delta(s)\|_\infty \leq 1$*. Furthermore, it is assumed that the number of unstable roots of* $G_{\text{unc}}(s)$ *remains the same and is equivalent to the number of unstable roots of* $G(s)$ *under all possible parameter variations characterized by the structured uncertainty model. If these conditions hold, then a controller* $C(s)$ *stabilizes all variations of* $G_{unc}(s)$ *under negative feedback if and only:*

1. $C(s)$ *stabilizes* $G(s)$
2. $\left|\left|\dfrac{W_u(s)G(s)C(s)}{1+G(s)C(s)}\right|\right|_\infty < 1$

Proof. The proof for Theorem 2 can be constructed graphically. The dashed line in Fig. 4.6 is the Nyquist plot of the open loop transfer function $G(s)C(s)$. The structured uncertainty of $G_{\text{unc}}(s)$ implies that at any point $s = j\bar{\omega}$ along the Nyquist plot, $G_{\text{unc}}(j\bar{\omega})C(j\bar{\omega})$ can take any value within a spherical region centered around $G(j\bar{\omega})C(j\bar{\omega})$ with radius $r(\bar{\omega}) = \|W_u(s)G(s)C(s)\|_{s=j\bar{\omega}}$. The first condition implies that the Nyquist plot of $G(s)C(s)$ has the correct number of encirclements of -1 to guarantee closed loop system stability. Since it is assumed that the number of unstable roots of $G_{\text{unc}}(s)$ is the same as the number of unstable roots of $G(s)$, the Nyquist criteria states that $G_{\text{unc}}(s)C(s)$ will also be closed loop stable under all possible outcomes of the proposed structural uncertainty if none of these uncertainties can change the number of encirclements of $G_{\text{unc}}(s)C(s)$ about -1. Another way to write this condition is for the radius of uncertainty, $r(\bar{\omega}) = \|W_u(s)G(s)C(s)\|_{s=j\bar{\omega}}$, to be less than $\|1+G(j\bar{\omega})C(j\bar{\omega})\|$ for all frequencies. This is exactly the second condition of the theorem. Additional discussion regarding the derivation and implications of this theorem can be found in [37]. □

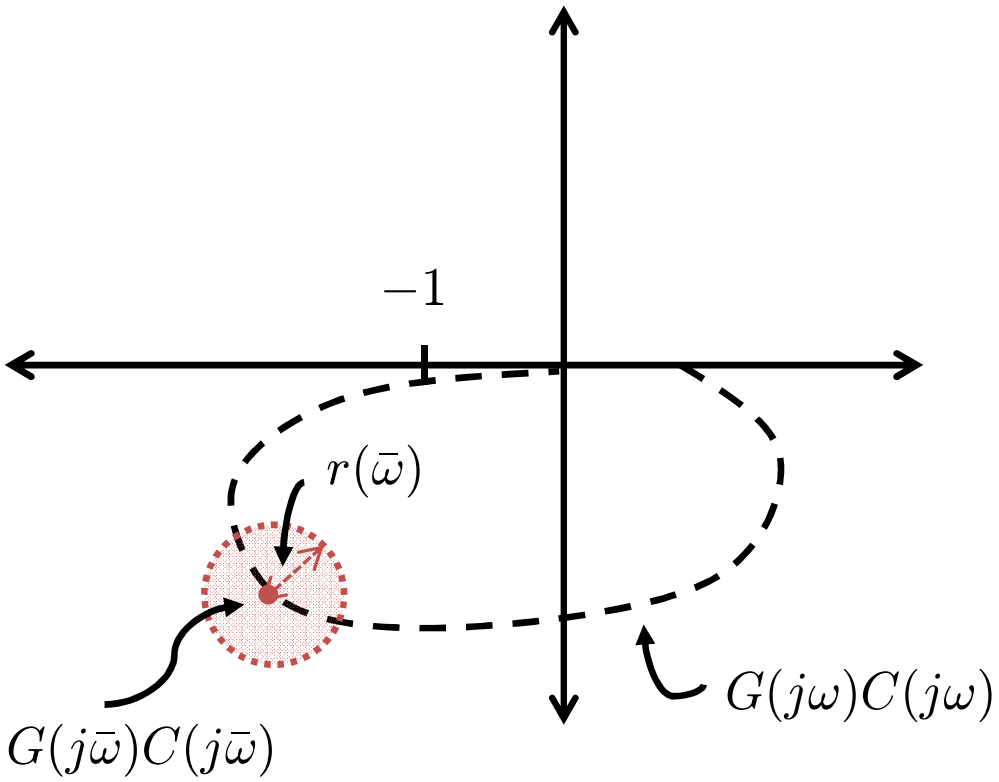
It is relatively straight forward to apply Theorem 2 to show the stability of $\hat{G}_{\hat{E}U}(s)$. One possible application of the theorem is to set:

$$\begin{aligned} W_u(s) &= \tfrac{-Q(s)}{G(s)C(s)} \\ \Delta(s) &= 1 \end{aligned}$$

One can quickly confirm that the assumptions in Theorem 2 can be easily satisfied for the system at hand. Going through the checklist:

1) $G(s)$ needs to be represented as a rational transfer function: this requirement can be satisfied through preliminary models of the system.

2) $W_u(s)$ must be a stable transfer function: since in this particular application, $W_u(s) = \frac{-Q(s)}{G(s)C(s)}$, stability can be guaranteed if $G(s)$ and $C(s)$ are minimum phase



$$r(\bar{\omega}) = ||W_u(j\bar{\omega})G(j\bar{\omega})C(j\bar{\omega})||$$

Figure 4.6: Nyquist plot of $C(s)G(s)$

and if $Q(s)$ is stable. These requirements are generally not problematic as most industrial robots are minimum phase systems. Furthermore non-minimum phase controllers and unstable filters are usually avoided in feedback loops, hence this requirement can be easily satisfied.

3) The number of unstable roots of $\hat{G}_{\hat{E}U}(s)$ must be the same as those in $G(s)$. This constraint can be violated in two ways:

i. Introduction of an unstable pole by the uncertainty term.

ii. Elimination of an unstable pole by the uncertainty term.

Since $\hat{G}_{\hat{E}U}(s)$ can also be rewritten as:

$$\hat{G}_{\hat{E}U}(s) = G(s)\left(1 - \frac{Q(s)}{G(s)C(s)}\right) \tag{4.5}$$

An additional unstable pole can be inserted only if $\frac{Q(s)}{G(s)C(s)}$ is unstable, which violates the previous constraint. An existing unstable pole can be removed if a unstable pole zero cancellation occurs between the $G(s)$ and $\left(1 - \frac{Q(s)}{G(s)C(s)}\right)$. Again, since $C(s)$ and $Q(s)$ are designed by the user, this scenario can be easily avoided.

Now to apply Theorem 2. The first condition states that $\hat{G}_{\hat{E}U}(s)$ is stable if and only if $C(s)$ is a stabilizing controller for $G(s)$, which is trivially satisfied. More interestingly, the second condition states that:

$$\left|\left|\frac{-Q(s)}{1 + G(s)C(s)}\right|\right|_{\infty} < 1 \tag{4.6}$$

Eq. 4.6 suggests that magnitude the product of $Q(s)$ and the sensitivity function of the nominal closed loop plant with controller $C(s)$ must be upper bounded by 1. This naturally leads to $Q(s)$ to be designed as a low pass filter. And $\hat{G}_{\hat{E}U}$ satisfies the theorem, which shows that the transfer function is stable, and that consequently also shows that $G_{\hat{E}U}$ is stable. At this point, the small gain theorem says that the DOB framework (Fig. 4.4) is stable if:

$$\left|\left|\frac{-C_t^{-1}(s)C(s)Q(s)}{1 - Q(s) + C(s)G(s)}\right|\right|_{\infty} < 1 \tag{4.7}$$

Eq. (4.7) is only interesting within the pass band of $Q(s)$. Setting $Q(s) = 1$ suggests that:

$$\left|\left|\frac{-1}{C_t(s)G(s)}\right|\right|_{\infty} < 1 \tag{4.8}$$

Eq. (4.8) suggests that there is a lower bound on the controller gains for $C_t(s)$ but not necessarily an upper bound. This condition is useful for gain tuning applications since in most gain tuning scenarios, the gains are increases for better performance rather than decreased.

Limitations

There are a few limitations with the DOB framework, many of these are mild limitations and should not affect the overall effectiveness of such a framework. A summary of these limitations are as follows:

1) $C_t^{-1}(s)$ is required to be stable: This condition is introduced so that the small gain theorem can be applied to the system shown in Fig. 4.4. This condition implies that the tuned block $C_t(s)$ should not introduce unstable zeros to the equivalent controller given by Eq. 4.3. This constrain is reasonable since Bode's Integral Theorem suggests that right half plane zeros can put limitations on the system's performance.

2) $G(s)$ and $C(s)$ cannot contain any unstable zeros: This constraint is notably problematic if the physical plant has non-minimum phase characteristics. Although for industrial robot applications, most robots are indeed minimum phase, hence the plant constraint should not be an issue. Additionally, since $C(s)$ is designed to be any stabilizing controller for $G(s)$, it should not be difficult to introduce a stabilizing controller with minimum phase properties.

3) $Q(s)$ needs to be stable and designed in a way such that Eq. (4.6) is satisfied: The stability condition of $Q(s)$ is implied by the structure of the DOB framework to ensure internal stability. As previously mentioned, Eq. (4.6) suggests that $Q(s)$ can be naturally designed as a lowpass filter. This coincides well with the initial motivation for the DOB structure. The only limitation set by Eq. (4.6) is that the bandwidth of $Q(s)$ is limited by the closed loop performance of the stabilizing controller $C(s)$.

As a result, while a few constraints are introduced to enforce system stability, closer analysis show that these constraints do not hinder the practical application of the DOB framework.

4.4 Results

This section attempts to verify the stability properties of the proposed DOB framework through both simulations and experiments. The nominal plant model used for each joint of the FANUC M16*i*B robot is a fifth order transfer function that is designed to closely match the robot system identification data up until 20 Hz. Both $C(s)$ and $C_t(s)$ are chosen to be PID controllers with form:

$$C(s) = \frac{K_v s^2 + (K_v K_p + K_i)\, s + K_p K_i}{s^2} \tag{4.9}$$

$$C_t(s) = \frac{K_{vt} s^2 + (K_{vt} K_{pt} + K_{it})\, s + K_{pt} K_{it}}{s^2} \tag{4.10}$$

The gains of $C(s)$ are conservatively selected such that $C(s)$ is a minimum phase controller that can stabilize the robot. $C(s)$ and $C_t(s)$ are chosen to be PID controllers so that the DOB framework can be directly compared with the M16*i*B's existing PID structure. $Q(s)$ is designed to be a lowpass filter with a 20 Hz bandwidth of form:

$$Q(s) = \frac{\omega_n^2}{s^2 + 2\omega_n s + \omega_n^2} \tag{4.11}$$

where $\omega_n = 40\pi$. The selection of $Q(s)$ and $C(s)$ satisfy the conditions introduced in the previous section to ensure stability of the DOB framework.

While it is difficult to prove that a particular system is stable for an unbounded set of controller gains, the following table attempts to show that the system is stable for a wide

	K_{pt}	K_{vt}	K_{it}
Gains in DOB Framework	$0.06 - 305$	$0.03 - 1.02$	$0.00 - 10.4$
Stable Gains in Default PID Structure	$10 - 20$	< 0.25	< 1

Table 4.1: Verified stable gain variations in DOB framework through simulation

range of gains. In particular, the following data is collected from the fourth joint of the FANUC M16*i*B robot. Table 4.1 details the gain variations for $C_t(s)$ which is still able to stabilize the system. Note that typical values for the PID gains using the default controller structure are given in the second row. The DOB framework is able to stabilize the system for a much higher range of gain variations. Table 4.2 provides a range of gains that are

	K_{pt}	K_{vt}	K_{it}
Gains in DOB Framework	$16.8 - 168$	$0.035 - 0.35$	$0.29 - 2.9$
Stable Gains in Default PID Structure	$10 - 30$	< 0.1	< 1

Table 4.2: Verified stable gain variations in DOB framework through experimentation

able stable through the DOB structure for the M16*i*B robot. These gains are verified and obtained through experimentation. Again, the typical values that can be used to stabilize the standard PID controller are given in the second row. Note the much wider range of stabilizing controller parameters.

With this said, there are several limitations with the DOB framework. Looking at Fig. 4.3, the entire feedback controller can be thought as a higher order controller whose order is upper bounded by the sum of the orders of $C(s)$, $C_t(s)$, and $Q(s)$. While it's straightforward to design $C(s)$ and $Q(s)$ and subsequently tune $C_t(s)$ to obtain the entire feedback controller, it is difficult to guarantee in advance the structure of the tuned controller. Hence the DOB framework may not be applicable if the hardware/software configuration does not allow for flexible feedback controller structures. A consequence of this limitation is that the framework cannot be applied to tune typical PID controllers found in many of today's mechanical systems.

4.5 Chapter Summary

This chapter presented a framework based on the disturbance observer for stable feedback controller tuning. Both simulation and experimental results are provided to demonstrate the robust stability conditions of the DOB framework. A current limitations of this approach, however, is that it cannot be used to PID controllers. While the user can specify the controller being tuned to be of any structure, the complete controller is given by Eq. (4.3). The framework essentially allows the user to tune a subset of the parameters of a higher

order controller. While this increases the computational complexity of the entire controller, in return, it is able to provide very robust stability properties for the controller. At this point of development, the DOB framework cannot be utilized for tuning PID controllers but it may play a significant role in tuning higher order controllers in the future.

Chapter 5

An Input Shaping Method to Suppress Transient Vibrations In Flexible Joint Robotic Manipulators

5.1 Introduction

This chapter proposes an input shaping approach for suppressing the fundamental vibratory mode in flexible joint robots. In many robot manipulator applications, such as spot welding, it is desirable for the robot to travel quickly from one point to another. In these situations, the robot user is primarily concerned with suppressing the robot end effector vibrations as it approaches each point. As a result, it is important to develop methods that can suppress transient vibrations. Section 5.2 briefly introduces the input shaping concept. Development of the proposed algorithm takes place in section 5.3. More specifically, section 5.3 discusses about both the system identification aspect as well as the actual input shaping procedure. Both simulation and experimental results are given in section 5.4. Finally, section 5.5 summarizes the contents of this chapter.

5.2 Input Shaping

Input shaping was first proposed by the authors in [36]. As the name suggests, the method involves modifying the desired system's input such that the actual output from the closed loop system matches the initial desired trajectory. The initial theory assumes that the system contains damped vibratory modes that can be accurately modeled by linear vibration analysis. Using this assumption, impulses can be injected into the system at a $\pi/2$ phase offset to effectively cancel out the vibratory response of the system. While the fundamental theory lies heavily on linear vibration analysis, the effectiveness of the input shaping approach was validated through experimentation on a more complex mechanism. Following

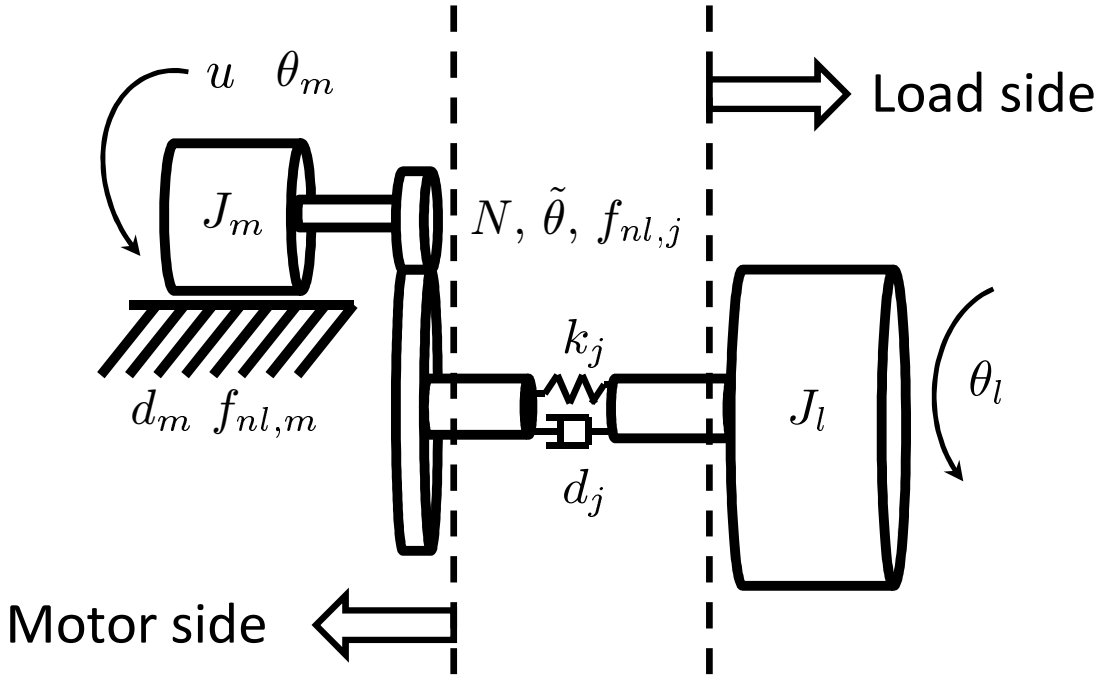


Figure 5.1: Two inertial model of an indirect drive mechanism

this example, this chapter develops the underlying theory for the input shaping approach based on linear vibration theory. But ultimately, the utility of the approach is demonstrated by both simulation and experimentation on the M-16iB industrial robot.

5.3 Theory

An individual joint of a robot manipulator can be approximated a two inertia model. For reading convenience, the two inertia model originally presented in Fig. 2.1 is shown again here in Fig. 5.1. In industrial robotic joints, both linear and non-linear joint dynamics exist. The linear portion is modeled in the two inertia model by the linear spring and damper which have constants k_j and d_j respectively. The nonlinear portion is modeled by the transmission error, $\tilde{\theta}$, and the generic variable $f_{nl,j}$. Unfortunately in most systems, nonlinear dynamics is difficult to model and predict. Unless more information about these nonlinear effects is known, it is not possible to compensate for them without the use of load side sensors. Since this nonlinear behavior is difficult to learn and generalize for any given input, the focus of the input shaping technique presented in this chapter will focus on learning and compensating for the linear component of the transmission dynamics, namely the fundamental vibratory mode.

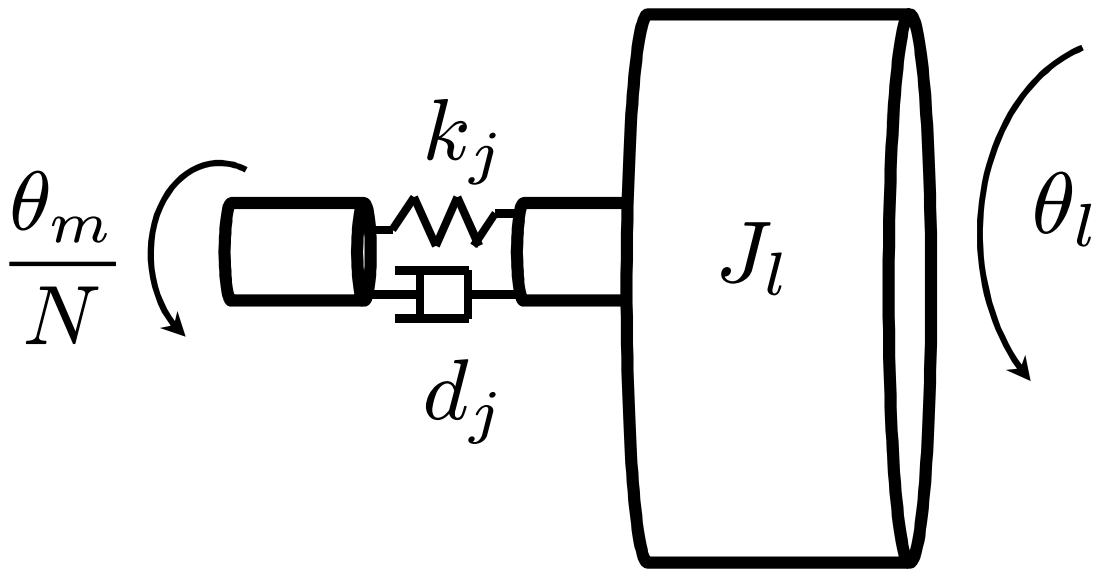


Figure 5.2: Single mass simplification

System Identification

In many applications, J_m and J_l can be calculated from the mechanical schematics and/or orientation of the mechanical system. Furthermore, if the motor side controller can provide tight control, then the motor side dynamics, namely the effects of motor side damping, d_m, and motor side friction, $f_{nl,m}$, can be neglected. The gear reduction ratio N is usually specified by the manufacturer; hence the only parameters to identify are k_j and d_j. If the motor side feedback controller has tight control performance, the two inertia system can be simplified to a single spring-mass-damper system shown in Fig. 5.2. Using this assumption, the load side time response subjected to a unit motor side step input is given by [29]:

$$\theta_l(t) = \frac{1}{Nk_j}\left[1 - e^{-\zeta\omega_n t}\left(\cos(\omega_d t) + \frac{\zeta\omega_n}{\omega_d}\sin(\omega_d t)\right)\right]1(t) \tag{5.1}$$

where $\zeta = \frac{1}{2}\frac{d_j}{\sqrt{k_j J_l}}$, $\omega_n = \sqrt{\frac{k_j}{J_l}}$, $\omega_d = \omega_n\sqrt{1-\zeta^2}$, and $1(t)$ represents the unit step response. The nominal step response of a second order system is shown in Fig. 5.3. Let x_1 and x_2 denote the overshoots of two consecutive vibratory responses of the step function, while t_1 and t_2 denote the time instances corresponding to x_1 and x_2 respectively. Taking the ratio

of x_1 and x_2 yields:

$$\begin{aligned} \frac{x_1}{x_2} &= \frac{e^{-\zeta\omega_n t_1}\left(\cos\omega_d t_1 + \frac{\zeta\omega_n}{\omega_d}\sin\omega_d t_1\right)}{e^{-\zeta\omega_n(t_1+T_d)}\left(\cos\omega_d(t_1+T_d) + \frac{\zeta\omega_n}{\omega_d}\sin\omega_d(t_1+T_d)\right)} \\ &= e^{\zeta\omega_n T_d} \\ &= e^{2\pi\zeta\omega_n/\omega_d} \\ &= e^{2\pi\zeta/\sqrt{1-\zeta^2}} \end{aligned} \tag{5.2}$$

where $T_d = \frac{2\pi}{\omega_d}$ is the period of the damped natural frequency ω_d. Then, solving for ζ yields:

$$\zeta = \frac{\delta}{\sqrt{(2\pi)^2 + \delta^2}} \tag{5.3}$$

where $\delta = \ln\frac{x_1}{x_2}$. Noting that $\zeta = \frac{1}{2}\frac{d_j}{\sqrt{k_j J_l}}$, Eq. (5.3) provides a relationship between the system parameter d_j and k_j based on the system's step response. ω_d can also be empirically measured from the step response data by the following formula:

$$w_d = \frac{2\pi}{T_d} = \frac{2\pi}{t_2 - t_1} \tag{5.4}$$

Given empirical measurements ω_d and δ, the joint parameters k_j and d_j can be determined with the following relationships:

$$k_j = \frac{\omega_d^2 J_l \sqrt{(2\pi)^2 + \delta^2}}{2\pi} \tag{5.5}$$

$$d_j = \frac{2\delta\sqrt{k_j J_l}}{\sqrt{(2\pi)^2 + \delta^2}} \tag{5.6}$$

In order to perform this empirical test to obtain the model parameters, a load side sensor to estimate the load side behavior is required. Additionally, the actuator on the motor side needs to be able to cleanly produce a step displacement, which requires both feedback and feedforward control. In some cases, it is not always possible to get good motor side performance with just a feedback controller, especially when the feedforward controller performance is poor. In these scenarios, techniques such as iterative learning control (ILC) [6] can be used to tune the feedforward torques such that the motor side actuator can produce the best possible step input for system identification.

Trajectory Modification

Once the system parameters have been identified, the procedure in this section can be used to reshape the system input to improve load side behavior. In many complex systems

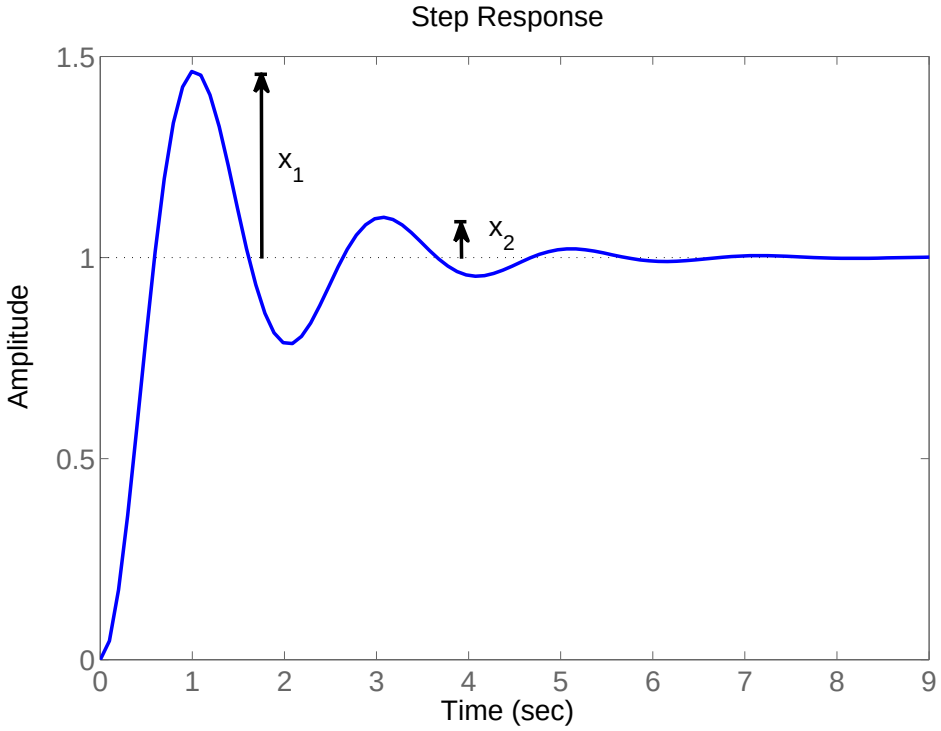


Figure 5.3: Nominal step response of a second order system

that utilize harmonic drives, such as industrial robots, a closed-form relationship between the input torque and load side output is not readily available due to the complexity of the mechanism. Instead, a complex dynamic model derived from Lagrangian mechanics is used to establish the relationship between the desired motor side trajectory and motor side torque. Ideally, if the transmission dynamics are known, it may be possible to calculate the corresponding motor side trajectory based on the desired load side trajectory. Since this is rarely the case, the motor side reference trajectory is usually calculated by neglecting any transmission dynamics. This desired motor side reference is then used by the dynamic model to compute the required input torque. While these complicated dynamic models provide superior motor side performance, they do not consider joint flexibilities. Hence the motor side reference and resulting feedforward torque may not always produce the desired load side performance, especially during transients. Rather than directly changing the feedforward torque or dynamic model, the proposed method focuses on reshaping the desired motor side trajectory to pre-compensate for the joint dynamics such that the resulting feedforward torques will produce the desired load side behavior. This approach has the advantage of being easy to implement since it does not require any substantial modification to existing

procedures.

Let $\bar{\theta}_{d,l}$ be the desired load side trajectory vector. Neglecting joint dynamics, the corresponding motor side trajectory vector can be written as:

$$\bar{\theta}_{d,m} = N * \bar{\theta}_{d,l} \tag{5.7}$$

Eq. (5.7) is often used as the feedforward controller F_1 in Fig. 2.6 whereas the Lagrangian model is used for F_2. If these feedforward controllers and motor side feedback controller are good, then the actual motor side output will be nearly identical to $\bar{\theta}_{d,m}$. Using this assumption, $\bar{\theta}_{d,m}$ is used as the reference base input to the single inertia system given in Fig. 5.2. The resulting load side output vector from this base input can be calculated by numerically solving the following:

$$J_l\ddot{\theta}_l + d_j\dot{\theta}_l + k_j\theta_l = \frac{d_j}{N}\dot{\theta}_m + \frac{k_j}{N}\theta_m \tag{5.8}$$

where θ_m is the known input which is set to be $\bar{\theta}_{d,m}$ and the output, θ_l, yields the estimated load side output, $\bar{\theta}_{l,\text{est}}$. This relationship is obtained from a torque balance on J_l in the two inertia model. Once $\bar{\theta}_{l,\text{est}}$ is known, the load side trajectory error can be computed as follows:

$$\delta\bar{\theta}_{d,l} = \bar{\theta}_{d,l} - \bar{\theta}_{l,\text{est}} \tag{5.9}$$

Using this error trajectory, the motor side compensation, $\delta\bar{\theta}_{d,m}$ is computed by solving Eq. (5.8) again. This time $\theta_l = \delta\bar{\theta}_{d,l}$ is the input and the output is $\theta_m = \delta\bar{\theta}_{d,m}$. Once this computation is complete, the new desired motor side trajectory is defined as:

$$\bar{\theta}^*_{d,m} = \bar{\theta}_{d,m} + \delta\bar{\theta}_{d,m} \tag{5.10}$$

Note that in the single mass model representation, the majority of the system uncertainty is eliminated as J_l can be accurately calculated and the remaining system parameters k_j and d_j are identified empirically. As a result, the load side output estimate from the single mass model can accurately capture the actual system behavior. From first glance, the process outlined above may seem like a roundabout way to calculate the compensation torque. Intuitively, it would be simpler to let $\theta_l = \bar{\theta}_{d,l}$ and solve for the motor side trajectory using Eq. (eq:GELoadside—ch:IS). Empirical data from simulation and experimentation, however, suggest that this method produces substantially worse performance than the initial outlined approach. The cause for this discrepancy is still unknown and may be a topic for future investigation.

5.4 Results

In this section, the proposed input shaping algorithm will be verified by application on both the robot simulator and the M-16*i*B industrial robot.

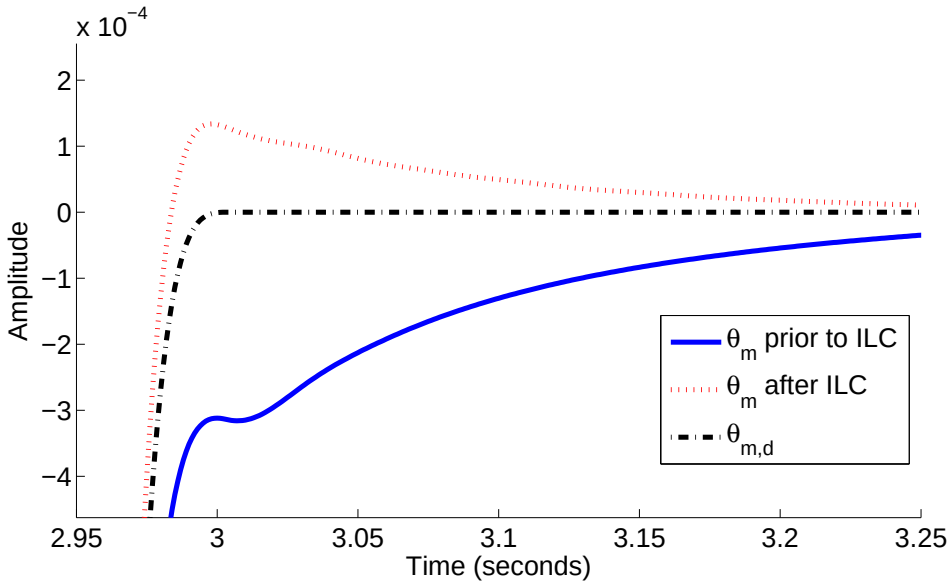


Figure 5.4: Motor side step response tracking

Simulation Results

This section contains the simulation results for the input shaping technique. For simplicity, in this section the algorithm is tested only on the base joint of the robot.

System Identification Results

In the system identification process, the actuator of the first axis of the robot (J_1) was instructed to produce a step response indicated by the dashed-dotted line in Fig. 5.4. The robot, however, was not able to track the step response well and produced the response shown by the solid line in Fig. 5.4. As a result, the measured load side response, shown in Fig. 5.5 as the solid line, was unable to produce the desired load side response necessary for system identification. As suggested earlier in the chapter, motor side ILC can be used to improve the motor side response of the robot. The motor side and load side responses after applying ILC are plotted as the dotted line in Figs. 5.4-5.5 respectively. 20 ILC iterations were used to obtain the aforementioned results.The joint stiffness and viscous damping parameters used in the remainder of this section are taken from the post-ILC load side response shown in Fig. 5.5.

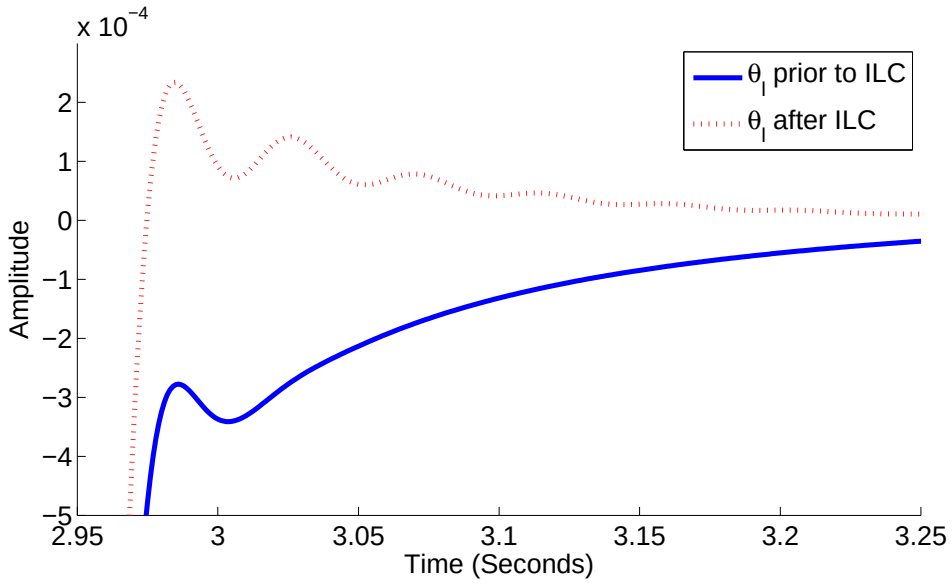


Figure 5.5: Load side step response

Trajectory Compensation Results

An assumption used in this approach is that the joint stiffness and damping coefficients are independent of the robot posture. Before testing this assumption, it is important to first verify the accuracy of the identified joint stiffness and damping parameters. If the identified parameters are accurate, the proposed method should be able to effectively compensate for the vibratory motions caused by motions along the J_1 axis when the robot is in its home posture. In the first simulation, the robot end effector is instructed to perform a step displacement about the J_1 axis in its home configuration and then return back to its home position. Fig. 5.6 shows the uncompensated load side step response (dashed-dot line). Note that the response exhibit mild oscillatory behavior. For comparison purposes, the desired load side reference is shown as the dotted line and the computed load side response based on the single inertia model is shown as the solid line. Note that while the predicted response of the single inertia system is unable to capture the offset of the simulated response, it is able to capture the oscillatory behavior (dashed dot line). This is not surprising since the single inertia model used to compute the trajectory compensation does not take into account static friction effects that is present in the robot simulator. The load side trajectory compensation is then calculated using the proposed approach. Fig. 5.7 plots the original load side desired trajectory in the top subplot and the calculated trajectory correction, $\delta\bar{\theta}_{d,l}$, in the bottom subplot. Note that the magnitude of the trajectory correction is relatively small compared to the original trajectory. The load side response after the proposed trajectory compensation

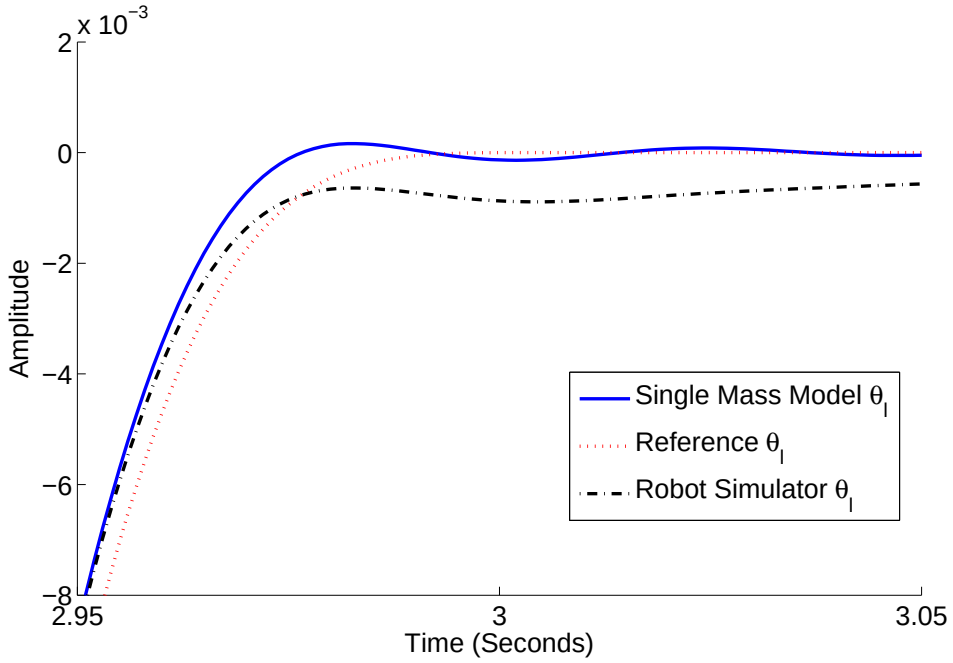


Figure 5.6: Load side step response comparison

is shown in Fig. 5.8. The top subplot of Fig. 5.8 compares the load side position response before and after the trajectory compensation. While the proposed trajectory was unable to compensate for the undershoot behavior, the compensated load side response exhibit less oscillatory behavior when compared to the uncompensated load side response. The bottom subplot of Fig. 5.8 compares the load side position error before and after the proposed trajectory compensation. From a quantitative standpoint, the compensated result showed a 35% decrease in the $2-$norm of the load side position error vector along the entire trajectory.

As mentioned earlier, an assumption of the proposed approach is that the joint parameters k_j and d_j do not vary significantly over the entire workspace. To validate this assumption, several simulations were performed on the robot simulator under different robot configurations. As mentioned earlier, in many applications of industrial robots, especially spot welding, the robots are desired to travel quickly from one point to another. In many cases, the user is primarily interested in suppressing robot vibrations as they approach these designated points. A step response can be used to mimic these conditions. Hence, the following simulations will test the step response of the robot simulator. Note that under all of these simulations, the same k_j and d_j obtained through system identification of the robot in its home posture are used. Each simulation will be done at a different robot configuration.

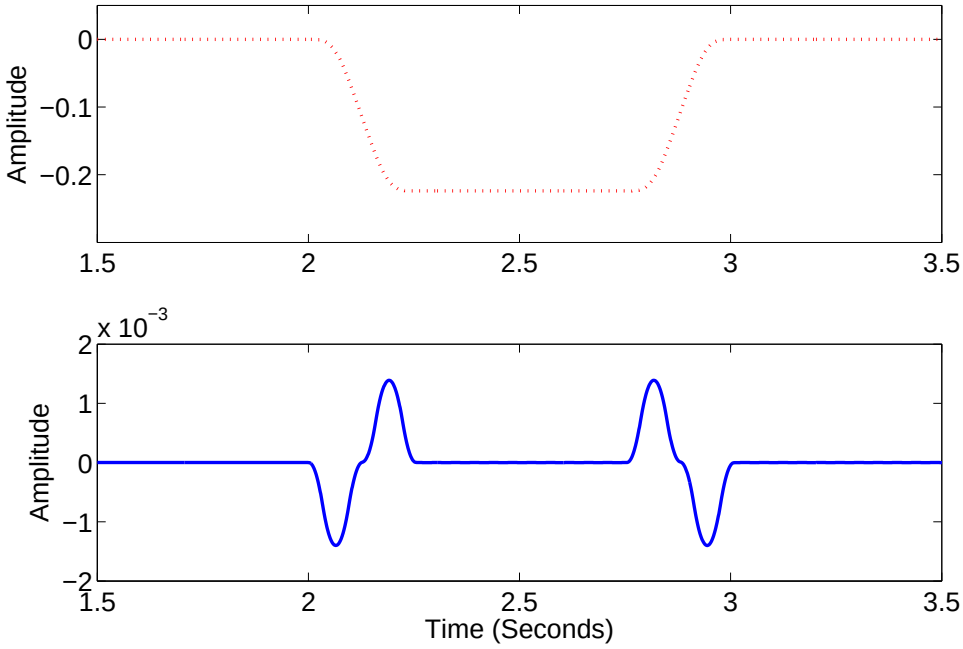


Figure 5.7: Trajectory correction: top subplot shows the original desired load side trajectory, bottom subplot shows the trajectory correction, $\delta\bar{\theta}_{d,l}$

Each configuration provides different load side inertias to the robot. For ease of terminology, the load side inertia of the robot in its home position will be referred to as the default inertia. The top subplot in Fig. 5.9 shows the load side position step response. In this simulation, a posture which has approximately 90% of the default inertia is used. The load side position error comparison can be found in the bottom subplot of Fig. 5.9. Note that the proposed compensation approach is still able to suppress a large amount of the load side vibrations. Fig. 5.10 shows the same load side response when the load side inertia is reduced to roughly 50% of the default inertia. Note that the compensated result is still able to greatly reduce the load side vibrations. Finally, Fig. 5.11 shows the robot's load side output response when the robot is in a pose that maximizes its load side inertia. The load side inertia in this case is roughly 200% of the default inertia. In this worst case scenario, the performance of the proposed algorithm seems to significantly degrade based on the time history plots in Fig. 5.11. A quantitative comparison of the $2-$norm of the error vector, however, still shows that the compensated response was able to reduce the error by 20%.

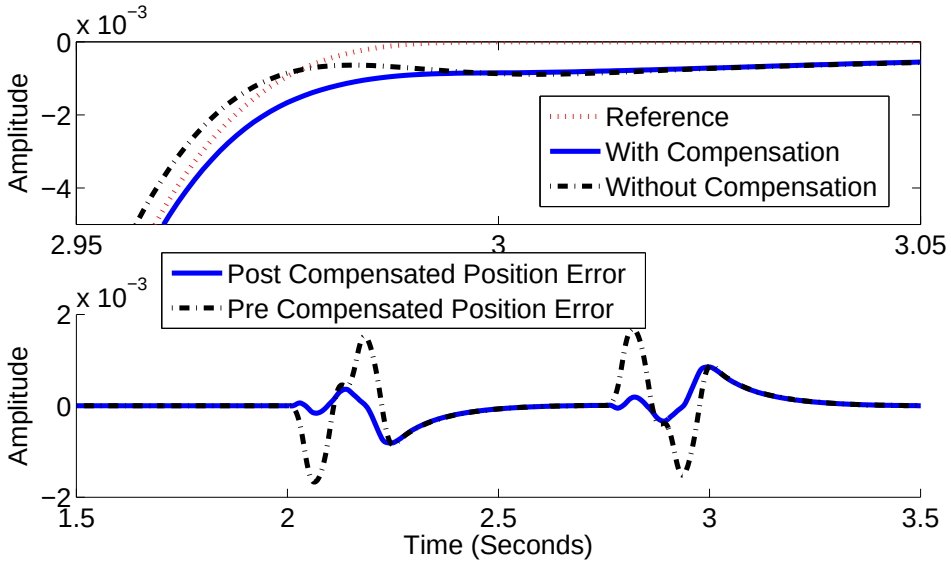


Figure 5.8: Compensated response: top subplot shows the load side tracking performance, bottom subplot shows load side error

Experimental Results

In this section, the M-16*i*B industrial robot is used to verify the proposed input shaping algorithm. The load side sensors such as inertia sensor and CompuGauge3D system will be used during the system identification process and also later on for performance evaluation. Due to the high inertia loads on the base three joints (J1-J3) of the FANUC robot, joint flexibilities are predominantly concentrated in these three joints. As such, the input shaping technique is applied to these three joints only.

The system posture used for parameter identification of these three joints is shown in Fig. 5.12. The acceleration measurements from the inertia sensor is used for evaluating the load side vibratory behavior during the system identification process. The load side acceleration measurements for the first three joints during the step response experiment are shown in Figs. 5.13-5.15. The motor encoder position during the measurement is also included as a reference. The expected linear step response behavior can be clearly seen in Fig. 5.13 and Fig. 5.15. As such, it confirms the assumption that the reducer behavior can be captured by a simple spring and damper model. This vibratory behavior, however, is not seen as clearly in Fig. 5.14. In particular, J2 is a much stiffer joint than J1 and J3. Hence, the natural frequency is much higher and is much harder to decouple the desired joint vibratory content from transient noise and other joint coupling effects. Due to this, the input shaping will be applied to J1 and J3 only in the following experimental results.

Similar to the simulation example, the robot is instructed to quickly turn 0.2 radians

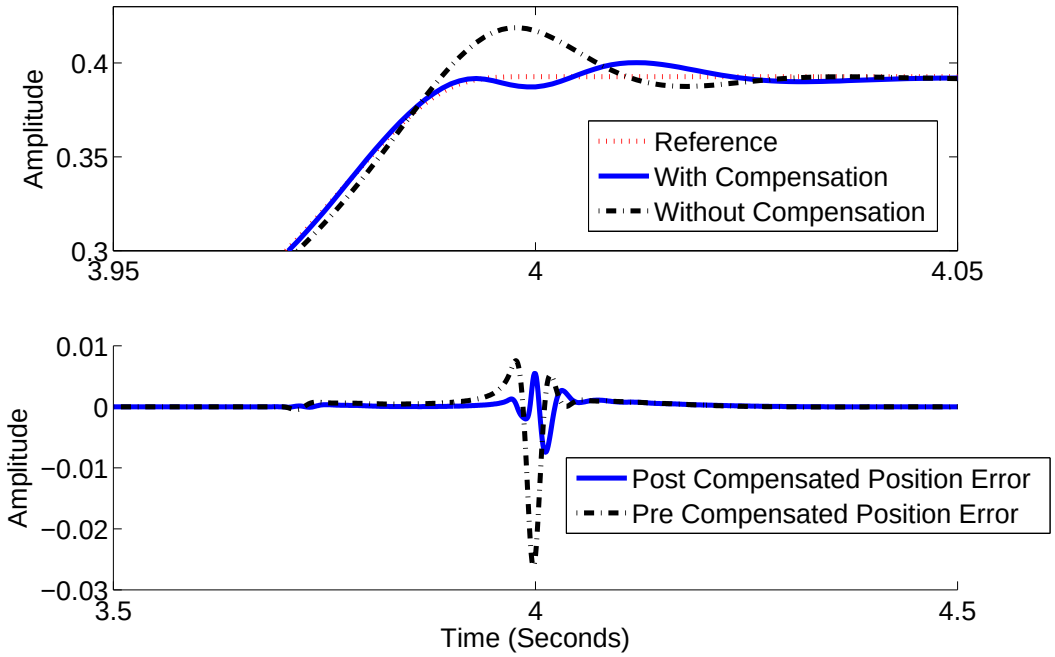


Figure 5.9: Compensated response: top subplot shows the load side tracking performance, bottom subplot shows load side error (90 % of default inertia)

about the J1 axis, hold for a second, and then swing back. In this experiment, the CompuGauge3D is used to track the robot end-effector position. The load side performance is shown in Fig. 5.16. The input shaping algorithm reduced the load side tracking error by 40% over the entire trajectory. In a second experiment, the end effector is maping out a planar square, stopping for 2 seconds at each corner. Again the load side tracking error is measured using the CompuGauge3D system. Figs. 5.17-5.19 show a portion of the load side position error as the robot traverses the trajectory. Note that the load side response after applying the input shaping technique shows superior vibration suppression characteristics. While this level of vibration suppression is not uniform across the entire trajectory, nowhere along the trajectory is the vibration actually increased after applying the input shaping technique. Overall in this multi-joint experiment, the proposed input shaping technique is able to reduce the load side position tracking error by 15%. One point to note is that the proposed input shaping approach is limited by the motor side performance. In reality, the feedforward controller is never perfect and considerations for stability and robustness put a limit on the feedback controller gains. If the motor side performance is poor, then the two inertia model in Fig. 5.1 cannot be reasonably approximated by Fig. 5.2 and thus the

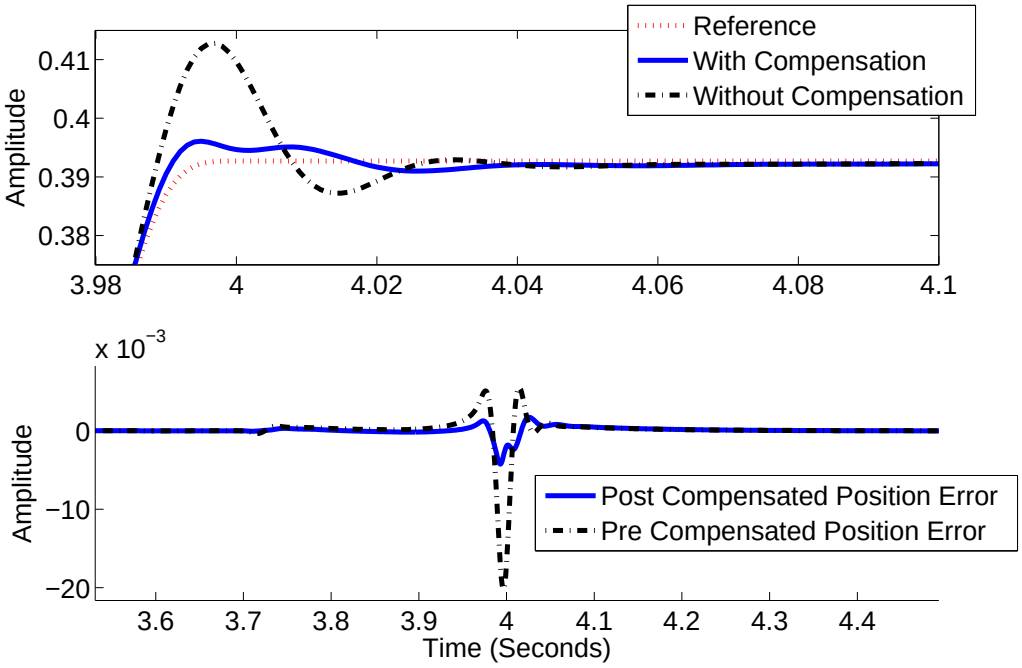


Figure 5.10: Compensated response: top subplot shows the load side tracking performance, bottom subplot shows load side error (50 % of default inertia)

identified system coefficients k_j and d_j cannot be used to approximate the actual system's behavior. This limitation may be overcome by extending the system identification procedure to identifying the motor side parameters as well. This is a topic of future work.

5.5 Chapter Summary

An input shaping approach for compensating for transient transmission dynamics for industrial robots with flexible joints was proposed in this chapter. The necessary system identification process required by the approach was also presented in detail. The proposed approach was evaluated through both simulations on the robot simulator as well as experimentation on the M-16iB industrial robot. Both the simulation and experimental results have shown that the approach is able to reduce load side transient vibrations if the base joint of the robot is required to follow a simple step trajectory. The proposed approach was also experimentally shown to be effective at suppressing transient vibrations along a more complicated trajectory.

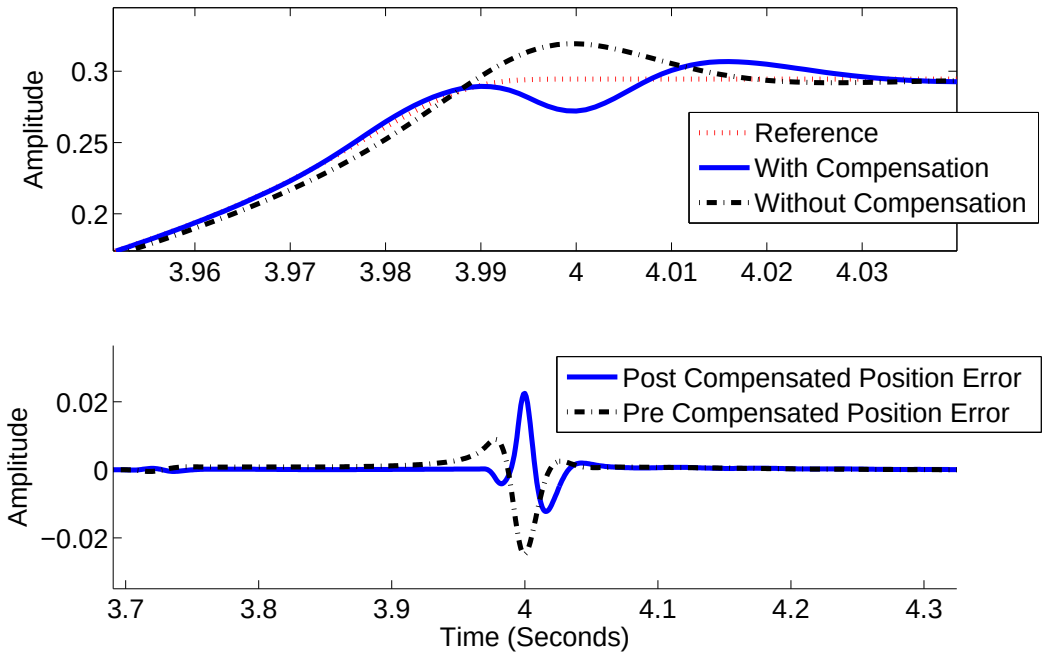


Figure 5.11: Compensated response: top subplot shows the load side tracking performance, bottom subplot shows load side error (200 % of default inertia)

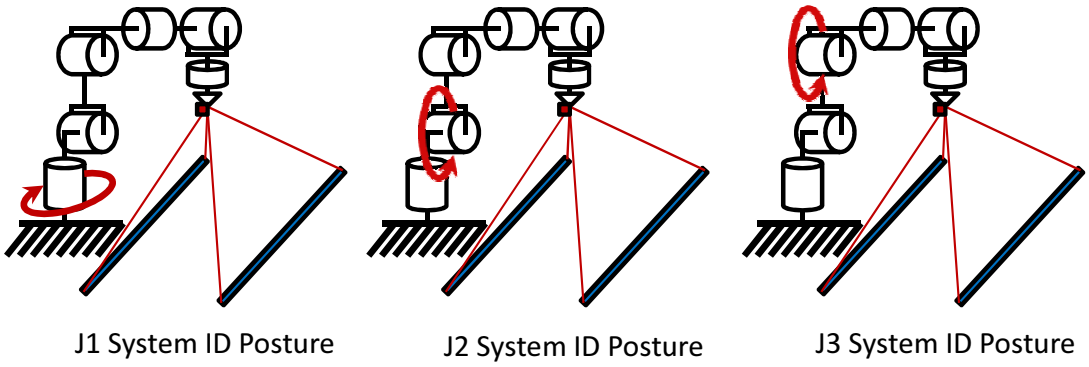


Figure 5.12: System identification posture for first 3 joints of the FANUC M-16iB robot

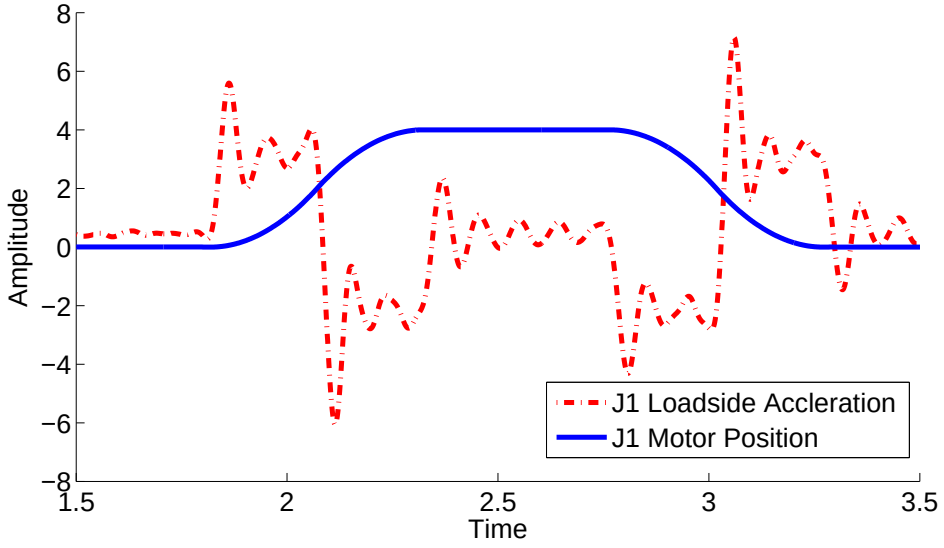


Figure 5.13: Load side acceleration response used for system identification of joint 1

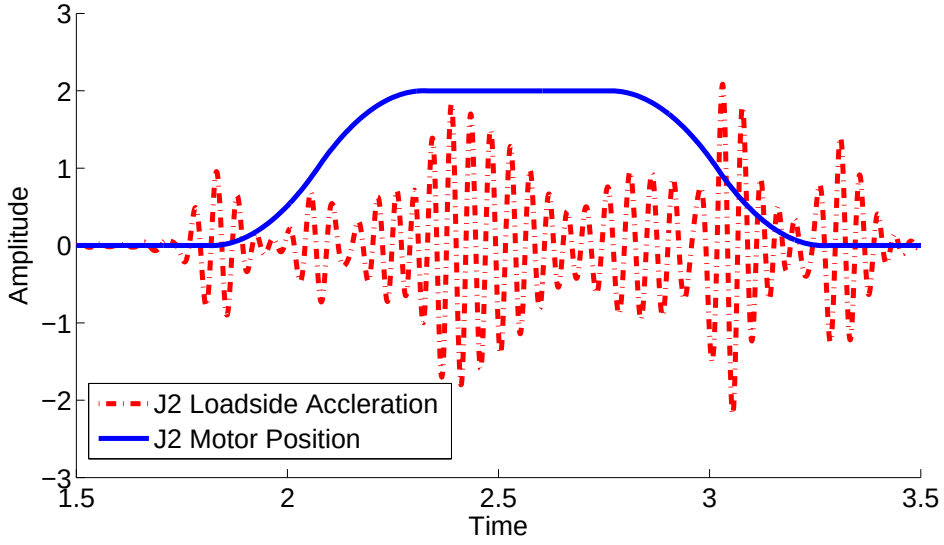


Figure 5.14: Load side acceleration response used for system identification of joint 2

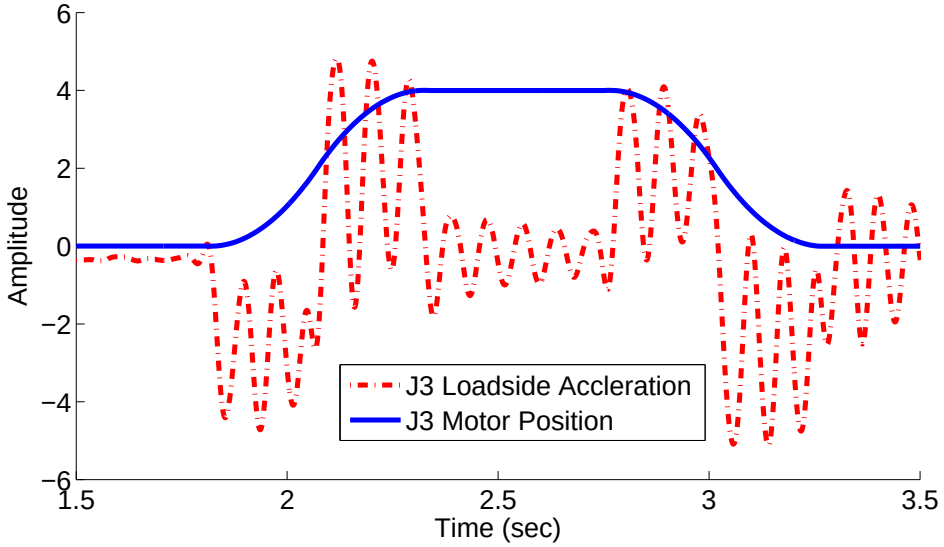


Figure 5.15: Load side acceleration response used for system identification of joint 3

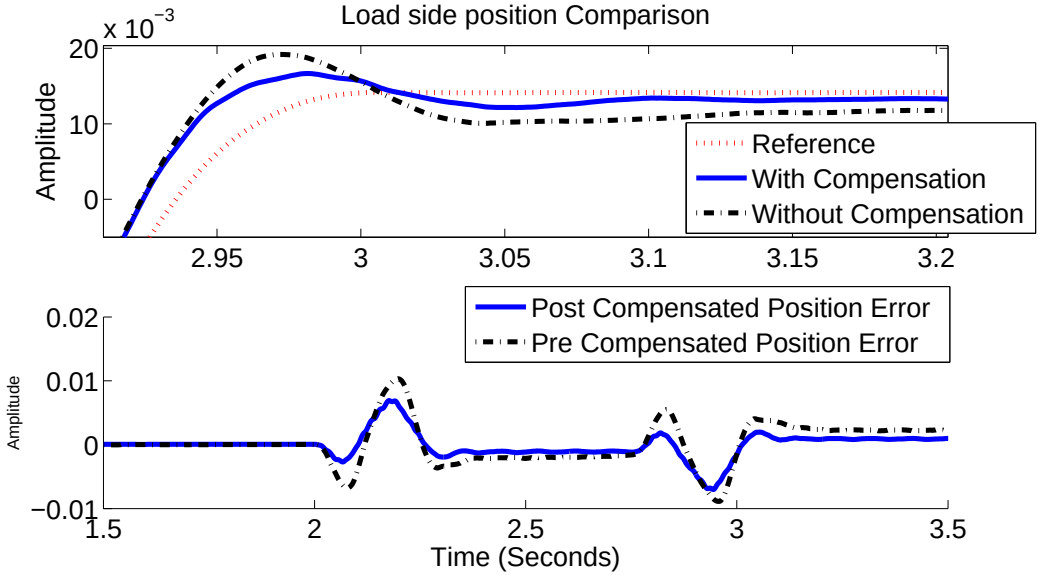


Figure 5.16: Compensated response: top subplot shows the load side position tracking performance, bottom subplot shows load side error

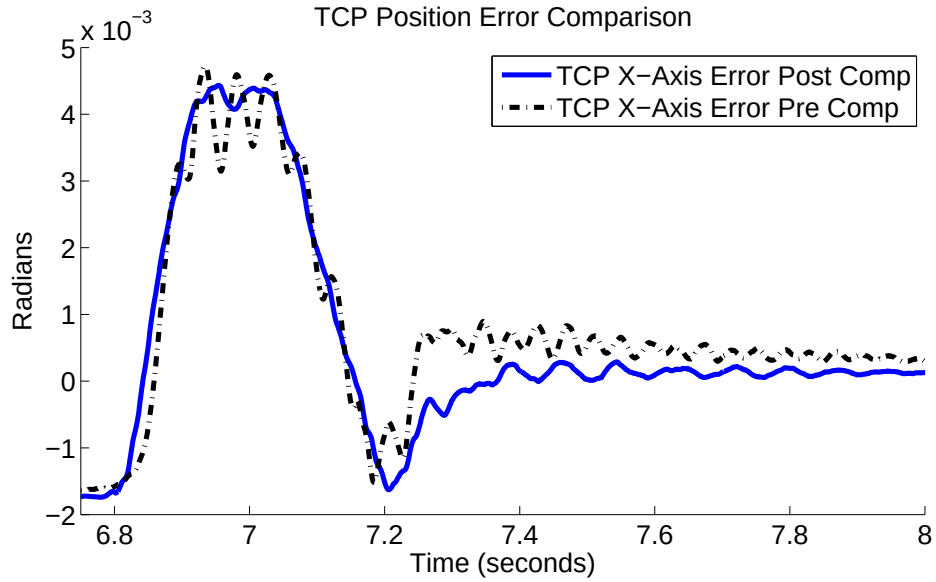


Figure 5.17: Compensated response: load side position tracking error along X axis

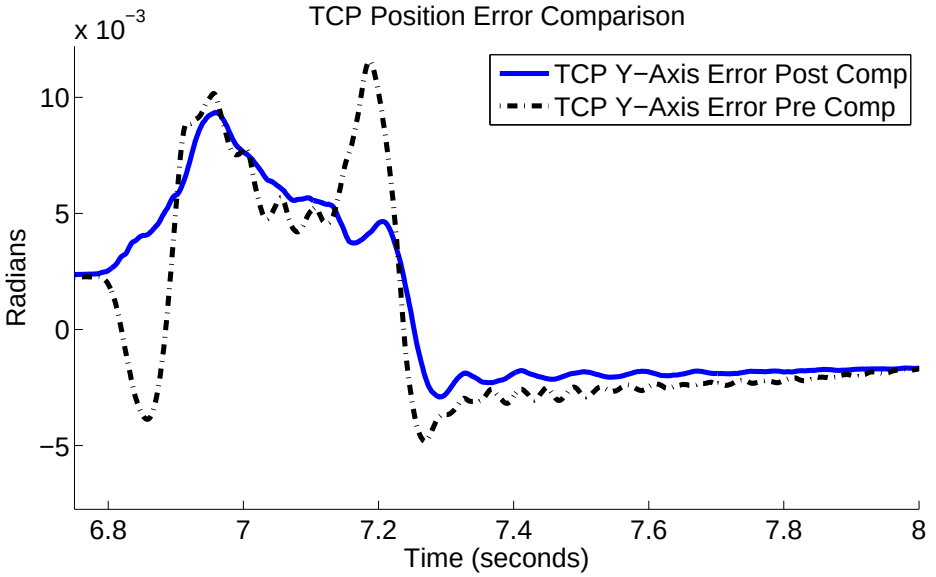


Figure 5.18: Compensated response: load side position tracking error along Y axis

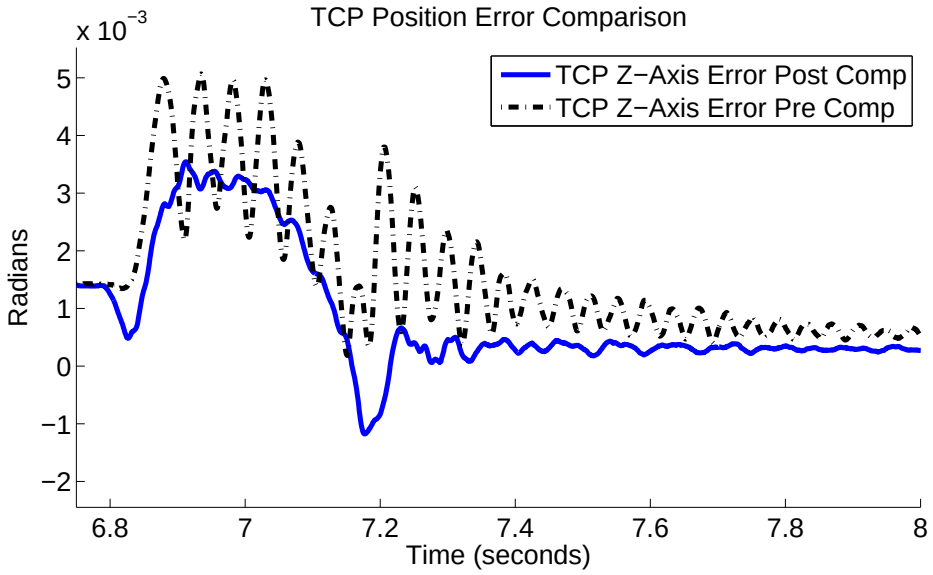


Figure 5.19: Compensated response: load side position tracking error along Z axis

Chapter 6

Conclusions

6.1 Chapter Summary

This dissertation presented feedback and feedforward controller design techniques for direct application to industrial robots. Chapter 2 introduced the background information necessary to understand this dissertation. This information includes single joint modeling techniques, multi joint robot dynamics, and controller structure. This chapter also introduced the software and hardware used for experimental verification in this dissertation as well as relevant system identification results.

Chapter 3 presented an algorithm for automatic gain tuning of robot manipulators. The proposed approach is an iterative method that uses nonlinear programming to improve robot performance by successively reducing a cost function. This chapter also discusses barrier methods, stepsize selection, as well as initial gain selection. Experimental results are included to verify the effectiveness of the proposed approach.

Chapter 4 presents a controller framework based on the disturbance observer. This controller framework is designed for gain tuning purposes and is shown to be robustly stable under a few mild assumptions. These assumptions are addressed and do not hinder the proposed framework's viability for most gain tuning scenarios. The framework is experimentally shown to be stable under a large range of controller gain variations.

Chapter 5 introduces an input shaping approach for reducing residual vibrations in industrial robots with joint flexibilities. This approach models the joint dynamics as a simple second order system and utilizes a simple system identification process to identify the model parameters. Experimental results show that the proposed approach is effective at reducing residual vibrations in robot manipulators, especially when the link inertia is large.

6.2 Future Work

Additional research can be done to further improve the topics presented in this dissertation. Chapter 3 presented an algorithm for automatically tuning the feedback controller gain for robot manipulators. Further work can be done to improve the step size selection process. Additionally, the proposed gain tuning algorithm improves robot performance by iteratively reducing a cost function. More work can be done to refine the cost function design process to see how different cost function parameters correlate to actual robot performance. Chapter 4 presents a controller tuning framework for stable gain tuning. The proposed framework can use further experimental verification. In particular, it would be interesting to combine the proposed gain tuning algorithm from chapter 3 with the framework proposed in chapter 4. More specifically, it would be interesting to tune the free controller $C_t(s)$ in the DOB framework using the proposed gain tuning approach presented in chapter 3. And finally, chapter 5 presented an input shaping method for suppressing residual joint vibrations. The current implementation assumes good motor side performance, which is something that is not always readily available. More work can be done to extend the input shaping approach to situations where the motor feedback controller has not been optimized.

Bibliography

[1] Analog Devices Inc. www.analog.com/en/mems-sensors/imu/adis16400/products/product.html.

[2] Kartik B. Ariyur and Miroslav Krstic. *Real-Time Optimization by Extremum-Seeking Control*. John Wiley & Sons, 2003.

[3] K. Astrom and T. Hagglund. *PID Controllers: Theory, Design, and Tuning*. The International Society for Measurement and Control, 2nd edition, 1995.

[4] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 2008.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2008.

[6] D. Bristow, M. Tharayil, and A. Alleyne. A survey of iterative learning control. *IEEE Control Systems*, 26:96–114, 2006.

[7] Michael Chan, Kyoungchul Kong, and Masayoshi Tomizuka. Automatic controller gain tuning of a multiple joint robot based on modified extremum seeking control. *Proceedings of the 18th IFAC World Congress*, pages 4131–4136, 2011.

[8] Wenjie Chen. *Intelligent Control of Robots with Midmatched Dynamics and Mismatched Sensing*. PhD thesis, 2012.

[9] Wenjie Chen, Pedro Reynoso Mora, Mike Chan, Cheng-Huei Han, and Masayoshi Tomizuka. UCB-FANUC Project: Activity Report June 2009 May 2010. Technical report, Mechanical Engineering Department, University of California, Berkeley.

[10] Wenjie Chen, Pedro Reynoso Mora, Mike Chan, Cong Wang, Chung-Yen Lin, and Masayoshi Tomizuka. UCB-FANUC Project: Activity Report June 2011 May 2012. Technical report, Mechanical Engineering Department, University of California, Berkeley.

[11] Wenjie Chen, Pedro Reynoso Mora, Mike Chan, Cong Wang, and Masayoshi Tomizuka. UCB-FANUC Project: Activity Report June 2010 May 2011. Technical report, Mechanical Engineering Department, University of California, Berkeley.

[12] CompuGauge 3D. www.dynalog-us.com/solutions/.

[13] P. Corke. A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, 3:24–32, 1996.

[14] FANUC Corp. www.micromech.co.uk/dir_products/pdf/fanuc/m_16ib_20_10l.pdf.

[15] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*. Pearson Prentice Hall, 5th edition, 2006.

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

[17] I. Godler, K. Ohnishi, and T. Yamashita. Repetitive control to reduce spped ripple caused by strain wave gearing. *Internation Conference on Industrial Electronics, Control and Instrumentations*, 2:1034–1038, 1994.

[18] Cheng-Huei Han, Wenjie Chen, Pedro Reynoso Mora, Mike Chan, and Masayoshi Tomizuka. UCB-FANUC Project: Activity Report June 2008 May 2009. Technical report, Mechanical Engineering Department, University of California, Berkeley.

[19] Cheng-Huei Han, Kyoungchul Kong, and Masayoshi Tomizuka. Sensor-based controller tuning of robot manipulators by real-time optimization. *Proc. of 9th International Symposium on Robot Control (SYROCO)*, pages 715–720, 2009.

[20] 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. *IEEE International Workshop on Advanced Motion Control*, pages 182–187, 2008.

[21] HarmonicDrive. http://www.harmonicdrive.net/.

[22] Monson H. Hayes. *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, 1996.

[23] S. Kemal Ider and Ozan Korkmaz. Trajectory tracking control of parallel robots in the presence of joint drive flexibility. *Journal of Sound and Vibration*, 319:77–90, 2009.

[24] Kyoungchul Kong, Kiyonori Inaba, and Masayoshi Tomizuka. Real-time nonlinear programming by amplitude modulation. *Proc. of ASME Dynamics Systems and Control Conference*, pages 417–424, 2008.

[25] Lennart Ljung. *System Identification: Theory for the User*. PTR Prentice Hall Information and System Sciences Series. Prentice Hall PTR, 2nd edition, 1999.

[26] David G. Luenberger. *Linear and Nonlinear Programming*. International Series in Operations Research and Management Science. Springer Science+Business Media, 2nd edition, 2008.

[27] T. Marilier and J. A. Richard. Non-linear mechanic and electric behavior of a robot axis with a "harmonic-drive" gear. *Robotics and Computer-Integrated Manufacturing*, 5:129–136.

[28] MATLAB. www.mathworks.com/products/index.html.

[29] Leonard Meirovitch. *Fundamental of Vibrations*. McGRAW-HILL INTERNATION EDITIONS: Mechanical Engineering Series. McGraw-Hill Higher Education, 2001.

[30] C. Walton Musser. Strain Wave Gear-Species in Which Only One of the Gears is Input.

[31] National Instruments. www.ni.com/labview/.

[32] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Signal Processing Series. Prentice Hall, 3rd edition, 2010.

[33] Neil Sclater. *Mechanisms and Mechanical Devices Sourcebook*. McGraw-Hill, 5th edition, 2011.

[34] Nabesco RV Series Spec Sheet. www.nabtescomotioncontrol.com/pdfs/rvseries.pdf.

[35] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer, 2009.

[36] N. Singer and W. Seering. Preshaping command inputs to reduce system vibration. *Journal of Dynamic Systems, Measurement, and Control*, 112:76–82, 1997.

[37] S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control, Analysis, and Design*. John Wiley & Sons, 2005.

[38] Timothy D. Tuttle. Understanding and Modeling the Behavior of a Harmonic Drive Gear Transmission. Technical report, MIT Artificial Intelligence Laboratory.

[39] Chun-Chih Wang. *Motion control of Indirect-drive Robots: Model Based Controller Design and Performance Enhancement Based on Load-side Sensors*. PhD thesis, 2008.

[40] Chun-Chih Wang and Masayoshi Tomizuka. Sensor-based controller tuning of indirect drive trains. *Advanced Motion Control*, pages 188–193, 2008.

[41] G. Zames. On the input-output stability of time-varying nonlinear feedback systems part one: Conditions derived using concepts of loop gain, conicity, and positivity. *IEEE Transactions on Automatic Control*, 11:228–238, 1966.