2013

On the Analysis and Design of Disturbance Rejecter

Jason Tatsumi
Cleveland State University

Follow this and additional works at: https://engagedscholarship.csuohio.edu/etdarchive

Part of the Electrical and Computer Engineering Commons

How does access to this work benefit you? Let us know!

Recommended Citation
https://engagedscholarship.csuohio.edu/etdarchive/830

This Thesis is brought to you for free and open access by EngagedScholarship@CSU. It has been accepted for inclusion in ETD Archive by an authorized administrator of EngagedScholarship@CSU. For more information, please contact library.es@csuohio.edu.
ON THE ANALYSIS AND DESIGN OF DISTURBANCE REJECTER

JASON TATSUMI

Bachelor of Electrical Engineering

Cleveland State University

December 2011

Submitted in partial fulfillment of requirements for the degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

at the

CLEVELAND STATE UNIVERSITY

December 2013
We hereby approve this thesis for

Jason Tatsumi

Candidate for the Master of Science in Electrical Engineering degree for the

Department of Electrical and Computer Engineering

and the CLEVELAND STATE UNIVERSITY

College of Graduate Studies

________________________

Thesis Chairperson, Zhiqiang Gao

Electrical and Computer Engineering Department & Date

________________________

Thesis Committee Member, Lili Dong

Electrical and Computer Engineering Department & Date

________________________

Thesis Committee Member, F. Eugenio Villaseca

Electrical and Computer Engineering Department & Date

Student’s Date of Defense: 12-9-2013
PREFACE

When I was first introduced to active disturbance rejection control (ADRC) by Prof. Zhiqiang Gao, I had a very difficult time trying to understand how it works. I could visualize sine waves and impulses flowing through transfer functions, but I could not imagine what happened to signals as they passed through state-space matrices. If it were true that state-space and frequency domain were simply two languages used to describe the same thing, why is it that classical control tools are left behind when the discussion is about a state-space controller? Are modern devices so well behaved that it is unthinkable that their response should fluctuate in gain or phase? These questions would continue to frustrate me until I was finally able to visualize ADRC in an alternate fashion.

During the final year of my undergraduate study, several students and I studied ADRC with Prof. Gao’s doctoral student Shen Zhao, who did a commendable job answering my barrage of questions about how it works. We were presenting an ADRC solution to a steam boiler control company with hopes of being able to get the technology into the PID dominated field of industrial control. Zhao proposed a modification to ADRC for industrial machines with long time delays that allows the observer bandwidth to be increased significantly without instability. Things were looking very promising until a sales person asked, “Didn’t you promise one parameter setup for ADRC?” You see, this company’s current software automatically sets up the PID control gains and they are expecting a similar, if not simpler, setup for ADRC. Yet for every implementation, we ask the engineers to determine the gain and time delay of the plant and the bandwidth for the controller, all of which were outside of their experience. With all of the research that has been done with ADRC, why don’t we give these people a single pushbutton solution?
I began studying the relay methods that could give me the information I need to set up ADRC. I found a method on my own but later discovered that there were better solutions that had been published. In addition, I needed to figure out how to automatically assign control parameters that would work well. The same rules for selecting PID gains would obviously not work for ADRC since its architecture is very different.

Moreover, I thought, why was it that Zhao’s modification allowed higher bandwidths? If the controller is designed to place the closed-loop poles at a specific location, how come in some cases the actual response doesn’t agree with it? This should not happen if, through disturbance rejection, the plant is reduced to chained integrators. But if this isn’t the case, what is it the plant is reduced to?

In order to completely automate the tuning process for ADRC, I had to know.
ACKNOWLEDGEMENTS

Thank you to Dr. Zhiqiang Gao for identifying my curiosity in control systems, mentoring me through my graduate studies, and providing me with the opportunity to make a contribution to active disturbance rejection control. Thanks to my dearest Qinling for encouraging me to discover new ideas and helping me to stay on course. And thanks to my father James for answering my childhood questions about how things work and nurturing my desire to learn.
ON THE ANALYSIS AND DESIGN OF DISTURBANCE REJECTER

JASON TATSUMI

ABSTRACT

In this thesis, the impact of the disturbance rejecter concept and the enforced plant has been explored. In order for the active disturbance rejection controller (ADRC) to provide a reasonable alternative to the industry standard PID controller, it is necessary to develop tuning procedures capable of providing adequate performance with reasonable stability margins. A focus should be placed upon the disturbance rejecter, as it is the heart and soul of ADRC. In this thesis, transfer function analysis of the enforced plant has been performed to connect ADRC with the tools from classical control. The relationship between the gain parameter and observer bandwidth is studied to understand why higher bandwidths are attainable with smaller gains. A root locus technique demonstrates how the enforced plant poles change with observer bandwidth. The Nyquist stability criterion is used to offer tuning methods that satisfy gain and phase margins and ensures a transient that satisfies a given damping requirement. A technique is offered to display the infinite radius encirclements of the Nyquist plot within a finite graph. Analysis is performed on why the controlled response is typically slower than desired and how to correct it.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>VI</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>X</td>
</tr>
<tr>
<td>NOMENCLATURE</td>
<td>XIII</td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Problem Statement</td>
<td>5</td>
</tr>
<tr>
<td>1.2 Structure</td>
<td>6</td>
</tr>
<tr>
<td>II. UNDERSTANDING THE DISTURBANCE REJECTER</td>
<td>8</td>
</tr>
<tr>
<td>2.1 The Cascaded-Integral Response</td>
<td>9</td>
</tr>
<tr>
<td>2.2 A Frequency Domain Perspective of the Enforced Plant</td>
<td>10</td>
</tr>
<tr>
<td>2.3 The Effect of Observer Bandwidth upon the Enforced Plant</td>
<td>13</td>
</tr>
<tr>
<td>III. FINDING ROOT LOCATIONS OF THE ENFORCED PLANT</td>
<td>16</td>
</tr>
<tr>
<td>3.1 Guidelines for the Enforced Plant Roots</td>
<td>17</td>
</tr>
<tr>
<td>3.2 Root Analysis of the First-Order System</td>
<td>20</td>
</tr>
<tr>
<td>3.3 Root Analysis of the Second-Order System</td>
<td>24</td>
</tr>
<tr>
<td>3.4 Summary of Enforced Plant Root Locations</td>
<td>26</td>
</tr>
<tr>
<td>IV. GRAPHICAL DESIGN OF THE DISTURBANCE REJECTER</td>
<td>27</td>
</tr>
<tr>
<td>4.1 Finding an Open-Loop Gain for Nyquist Stability Analysis</td>
<td>28</td>
</tr>
</tbody>
</table>
4.2 Designing StabilityMargins ................................................................. 31

4.3 Satisfying a Damping Factor of Closed-loop Poles .......................... 33

4.4 Solving for a Minimum Bandwidth to Guarantee Stability ............... 37

V. NYQUIST PLOTS WITH INFINITE RADIUS ........................................ 41

5.1 Measuring the Angle of an Infinite Radius Arc ............................... 42

5.2 Mapping Infinite Scales to Finite Axes ........................................... 44

5.3 Generating the Infinite Radius Nyquist Plot .................................. 51

VI. RECOVERING LOST GAIN IN THE ENFORCED PLANT .................. 53

6.1 First-order Gain Correction .............................................................. 54

6.2 Second-order Gain Correction ......................................................... 56

6.3 Using the Inverse of the Enforced Plant .......................................... 59

VII. PLANT IDENTIFICATION ................................................................. 60

7.1 Relay Tuning Method ........................................................................ 61

7.2 Solving for the Unknown Parameters .............................................. 62

VIII. FREQUENCY-BASED DESIGN EXAMPLE ........................................ 64

8.1 Finding Model Parameters .............................................................. 65

8.2 Root Locus Analysis .......................................................................... 66

8.3 Finding Observer Parameter Values ................................................. 67

8.4 Analyzing the Results ...................................................................... 69
8.5 Improving Observer Bandwidth ................................................................. 70
8.6 Simulation Results .................................................................................. 73

IX. CONCLUSIONS AND FUTURE RESEARCH ......................................... 77

REFERENCES ............................................................................................... 79

APPENDICES ................................................................................................. 84

A. MATLAB Function for ADRC Root Locus ............................................. 84
B. MATLAB Function for Damping Angle Nyquist Contour ......................... 85
C. MATLAB Functions for Infinite Radius Nyquist Plot .............................. 85
D. MATLAB Function for Minimum $b_0$ Calculation .................................... 89
E. MATLAB S-Function for FOPDT Identification ....................................... 93
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1: Block diagram of enforced plant</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2: Effect of increasing observer bandwidth</td>
<td>14</td>
</tr>
<tr>
<td>Figure 3: First-order pole movement: $a=1$</td>
<td>21</td>
</tr>
<tr>
<td>Figure 4: First-order pole movement with various $b/b_0$ ratios: $a=1$</td>
<td>23</td>
</tr>
<tr>
<td>Figure 5: First-order pole movement with various $b/b_0$ ratios: $a=-1$</td>
<td>23</td>
</tr>
<tr>
<td>Figure 6: Second-order pole movement for various $\zeta$, $n=2$, $b_0=b$, $\omega_n=1$</td>
<td>25</td>
</tr>
<tr>
<td>Figure 7: Second-order pole movement for various $b/b_0$ ratios: $\zeta=1$</td>
<td>25</td>
</tr>
<tr>
<td>Figure 8: Simplifying block diagram</td>
<td>28</td>
</tr>
<tr>
<td>Figure 9: Determining $b_0$ to satisfy stability margins</td>
<td>32</td>
</tr>
<tr>
<td>Figure 10: Traditional Nyquist contour</td>
<td>34</td>
</tr>
<tr>
<td>Figure 11: Modified Nyquist contour</td>
<td>34</td>
</tr>
<tr>
<td>Figure 12: Nyquist plots evaluated at various damping ratio angle contours</td>
<td>36</td>
</tr>
<tr>
<td>Figure 13: Finding bandwidth limitation</td>
<td>39</td>
</tr>
<tr>
<td>Figure 14: Phase discontinuities over positive and negative frequencies</td>
<td>43</td>
</tr>
<tr>
<td>Figure 15: Piecewise reciprocal scale</td>
<td>46</td>
</tr>
<tr>
<td>Figure 16: Piecewise reciprocal scale with parameter $x_1$</td>
<td>47</td>
</tr>
</tbody>
</table>
Figure 17: Arctangent scale ................................................................. 47
Figure 18: Arctangent scale with parameters $x_1$ and $x_2$ ......................... 48
Figure 19: Logarithmic scale .................................................................. 49
Figure 20: Logarithmic scale with parameter $x_1$ ........................................ 50
Figure 21: Radii representing $x_1$, $x_2$, and infinity ..................................... 52
Figure 22: Comparison of conversion scales .................................................. 52
Figure 23: Uncorrected enforced plant ........................................................ 54
Figure 24: Gain corrected enforced plant ..................................................... 54
Figure 25: Gain correction with filter .......................................................... 55
Figure 26: Modified enforced plant: $b=3$, $a_1=7$, $a_0=0$ ......................... 58
Figure 27: Modified enforced plant: $b=3$, $a_1=7$, $a_0=11$ ....................... 58
Figure 28: Modified enforced plant: $b=0$, $a_1=7$, $a_0=11$ ....................... 58
Figure 29: Modified enforced plant: $b=3$, $a_1=0$, $a_0=0$ ......................... 58
Figure 30: Nyquist response of various model identifications ......................... 65
Figure 31: Nyquist response of model derivatives ......................................... 65
Figure 32: Approximation of time delay response ......................................... 66
Figure 33: Pole paths of approximate model .............................................. 66
Figure 34: Minimum $b_0$ for various GM .................................................... 69
Figure 35: Minimum $b_0$ for various PM ..................................................... 70
Figure 36: Minimum $b_0$ for various damping angles ........................................... 70

Figure 37: Nyquist of $G_p s/b_0$ ........................................................................ 71

Figure 38: Shaping plot of $G_p s/b_0$ ................................................................. 71

Figure 39: Modified block diagram for delay system ......................................... 72

Figure 40: Various control signal delay values .................................................. 72

Figure 41: Testing gain margin ........................................................................... 74

Figure 42: Testing phase margin ......................................................................... 74

Figure 43: Applying inverse correction of enforced plant ................................... 75

Figure 44: Inverse correction with higher plant gains ........................................ 75

Figure 45: Effect of inverse enforced plant correction on controlled response .... 76
## NOMENCLATURE

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UIO</td>
<td>Unknown input-output observer</td>
</tr>
<tr>
<td>DOB</td>
<td>Disturbance observer</td>
</tr>
<tr>
<td>ESO</td>
<td>Extended state observer</td>
</tr>
<tr>
<td>GPI</td>
<td>Generalized proportional-integral</td>
</tr>
<tr>
<td>ADRC</td>
<td>Active disturbance rejection control</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional-integral-derivative</td>
</tr>
<tr>
<td>GM</td>
<td>Gain margin</td>
</tr>
<tr>
<td>PM</td>
<td>Phase margin</td>
</tr>
<tr>
<td>FOPDT</td>
<td>First-order plus dead time</td>
</tr>
<tr>
<td>RMS</td>
<td>Root-mean-square</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

The field of automatic control owes its conception to the dreamers who wished for devices capable of enhancing or replacing the actions of human beings as they operate machines. These controlling devices cannot possibly be expected to be aware of all of the changing elements that might cause a machine to react differently, nor how the machine will respond given measurements of these elements. Since perfect knowledge of a machine’s response at all times may not be achievable or may be far too complex to understand, anything that causes the machine to behave differently from what might be expected can be considered as disturbance.

The concept of rejecting disturbance began at a time before the magnetic compass when navigation needed a tool to keep track of direction. The south-pointing chariot, used during the Han Dynasty in China, accomplished this by the use of mechanical systems between a pointer and the two wheels of the chariot [1]. Once the pointer was manually set towards south, any change in direction by the chariot would be considered disturbance, measured by the wheels, and the pointer would spin toward its original direction.
Disturbance rejection established itself in modern control theory with the invention of the unknown input observer (UIO) [2]. Other developments were later proposed that took advantage of additional information. Adaptive plant-noise cancelling uses an adaptive modeling technique to sense the output noise of the machine and uses an inverse model to cancel the noise at the input [3]. The disturbance observer (DOB) functions very similar to this by using linear, time-invariant (LTI) models and low-pass filters [4, 5]. The extended state observer (ESO) offers the state estimation from the Luenberger observer [6], but includes an additional disturbance estimation state without the requirement of a model [7, 8]. Similarly, the generalized proportional-integral (GPI) observer estimates disturbance with a series of integration states to remove the effects of random noise [9, 10].

Traditionally, a focus is placed upon making changes to a controller in order to satisfy each different machine’s response. With a disturbance rejection control system, changes are instead made to the response of the machine in order to satisfy a predetermined controller. A recent proposal has labeled this a “Copernican moment” which radically changes how automatic control systems are seen by combining the machine with a disturbance rejecter to make an enforced plant [11]. As long as the machine has been changed to operate in a predictable fashion, control of the machine becomes much simpler.

Active disturbance rejection control is designed with the ESO to provide real-time estimation and cancellation of the disturbance and unknown internal dynamics [7, 8, 12]. Implementation of the ESO was simplified by linearization and by making the observer gains functions of a single tuning parameter representing observer bandwidth [13]. A
discrete-time version of this linearized ESO was proposed for digital implementations [14]. Studies have been performed on ADRC to test its stability [15], prove the convergence of state and disturbance estimation [16, 17, 18], and show that estimated states are able to track in real-time [19]. Analysis of ADRC has been performed in the frequency domain [20, 21]. It has been applied to the temperature control systems of a hose extrusion process with a 50 percent reduction in energy consumption [22] and has attracted private sector development and implementation [23-25].

Recent developments in disturbance rejection include research from across the globe. In China, a comparative study between ADRC and proportional-integral-derivative (PID) control has been conducted on the five-input, four-output ALSTOM power plant to negate the disturbance effects of varying coal quality [26]. The ESO has been used with a model predictive controller to cancel disturbance and modeling errors [27]. The closed-loop and sensitivity transfer functions have been studied for frequency response analysis [28].

In the United States, ADRC has been compared with PID and H-Infinity to control the NASA developed HiTECC jet engine model simulation software [29]. Smith’s predictor has been combined with ADRC to provide a disturbance rejection control method for systems with large time delay [30]. A disturbance decoupling technique was studied to control a mobile robotic manipulator arm while the vehicle changes acceleration and drives over uneven terrain [31]. A cost minimizing adaptive disturbance rejection technique was applied to aircraft models to cancel the jet actuator nonlinearity [32].
A study in Mexico of disturbance rejection has been done on differentially flat systems [33]. The GPI observer has been developed to control a railway train suspension system [34]. Smith’s predictor has also been applied to the GPI observer for nonlinear delay systems [35].

In Germany, a method for reducing computation latency in discrete-time implementations of the ESO has been discovered [36]. A discrete-time control of electrical drives has been performed using a disturbance estimation technique very similar to ESO [37]. ADRC has been used in France to decouple mass flow and pressure from a centrifugal compressor in a fuel cell application [38]. In Italy, ADRC has been applied as a temperature control for lasers in optical cavities [39]. In Turkey, ADRC has been compared with a fractional-order PID to control a two-mass drive system [40]. Rotor and grid voltages of a wind energy generation system in Morocco have been controlled by ADRC [41]. In India, a survey of other important ADRC applications has been conducted [42].

Although analysis and stability proofs have been derived [15-17, 19], they are complex with very strict assumptions. Since ADRC assumes very little information about the process, it isn’t easy to state whether a system with proven stability is allowed a stability margin at a set parameter tuning. Likewise, the tuning process has been more of a trial-and-error process of guessing the right parameters, albeit possibly more intuitive than PID tuning. Unlike PID control, there isn’t an available solution that can automatically assign a starting point for tuning parameters. Rather than analyzing ADRC in the time-domain, it may be more practical for engineers to understand stability and tuning using frequency-domain techniques.
The enforced plant is studied because of its importance to the design of ADRC. When the plant response has been altered to act as a desired enforced plant, a controller feedback loop changes this response into a desired relationship between the plant output and the reference. ADRC design assumes that the controller has been tuned based upon a design requirement and that the disturbance rejecter is to alter the plant enough so that the controller can meet it. Therefore, the effort of control design should be placed upon tuning the plant rather than the controller. After all, shouldn’t the primary means of rejecting disturbance receive greater attention in a disturbance rejection control system?

1.1 Problem Statement

The purpose of this master’s thesis is to develop a tuning method for ADRC so that the enforced plant satisfies a bandwidth requirement while meeting a predetermined stability margin requirement. After the process model has been automatically obtained by means of relay tuning identification [43, 44] or using an adaptive filter [45, 46], the model information could be used to find the acceptable tuning parameters or a relationship between them. A computer algorithm is made possible to discover a means of eliminating the need for manual parameter tuning by the operator.
1.2 Structure

Chapter 2 introduces ADRC from an alternate perspective so that it can be better understood how it can reject disturbances and reduce a complex system into one that approximates a cascaded-integral.

The traditional root-locus technique demonstrates how a response changes as a proportional gain value is adjusted within a feedback loop [47]. Chapter 3 presents a modification to root-locus to show how the enforced plant will change as an observer bandwidth parameter is adjusted.

The Nyquist stability criterion can be used to measure the stability of the enforced plant and to design the disturbance rejecter [48-50]. Chapter 4 presents how predetermined stability margins can be used in order to decide on tuning parameters that allow for a range of different responses to result in a stable enforced plant. The chapter also introduces a Nyquist contour with an altered path in order to ensure that the enforced plant will have a desired amount of damping so as not to result in excessive oscillation. Chapter 5 presents techniques for displaying the infinite plane of points on the Nyquist plot within a finite window for computer displays and printouts.

Since the enforced plant may only approximate a cascaded-integral, there may be a range of frequencies where the gain is too low, resulting in slower performance when the controller is tuned for a desired response time. Chapter 6 discusses techniques to correct the gain in order to recover lost performance.
Relay tuning offers a method of identifying plant parameters with little [51, 52] to no information [43, 44] about the process and has been traditionally used to automatically set PID gains [53, 54].

Chapter 8 uses the techniques presented in order to discover the tuning parameters in order to control an industrial process model and satisfy stability margin requirements.

In Chapter 9, a first-order plus dead time (FOPDT) model example is used to demonstrate the techniques presented.
CHAPTER II
UNDERSTANDING THE DISTURBANCE REJECTER

The ADRC system can be viewed as a cascaded loop. The inside loop is the enforced plant that consists of the actual plant and the disturbance rejecter. The outside loop consists of the controller itself. Since the form of the controller is somewhat predetermined, consider what is sometimes being asked of it. If the enforced plant alone is unstable as the result of observer tuning, the controller must work to stabilize it. However, this controller hasn’t been intentionally designed to alter stability. Unstable poles in the enforced plant may place a minimum limit on the proportional gain of the controller in order to move them into a stable region. Likewise the controller is not designed to intentionally suppress oscillation caused by complex or imaginary poles in the enforced plant. If the predetermined controller is to correct for issues such as these, it is more accidental than intentional.

As a result, the goal of tuning the disturbance rejecter is to ensure that the enforced plant behaves well enough so that the controller can do its job without being designed with any plant information.
2.1 The Cascaded-Integral Response

The enforced plant proposed by Han [7, 8] is a cascaded-integral. This is simply a series of integrating functions between the input and output with all poles located at the origin. In the differential equation of a process model, a gain exists between the control signal and the \( n^{th} \) derivative of the process output when all other terms are ignored. A linear time-invariant model will have a constant gain denoted as \( b \). Since the plant is not expected to always behave like the process model, a time varying disturbance is said to be present. The goal of this disturbance rejecter is to treat the undesired terms of the process model and any external disturbance as total disturbance \( (f) \) and to estimate and cancel it in order for the enforced plant to behave like a cascaded-integral. Equation (2.1) shows the relationship between the output derivative, gain \( b \), control signal, and total disturbance. A controller would then be designed based on the cascaded-integral response, which is reasonable since it can be easily controlled by proportional and derivative gains.

\[
y^{(n)} = bu + f
\]

(2.1)

ADRC incorporates the Luenburger state-observer in order to estimate a relatively noiseless output and several \((n-1)\) derivatives of it [6, 12]. In addition, an extended state is added for the estimation of total disturbance. This extended state observer (ESO) provides the information necessary for the disturbance rejecter and for state-feedback control. When originally proposed, the ESO contained several non-linear gain parameters to configure, resulting in a time consuming implementation. However, this has been simplified into a linear design with a single tuning parameter that dictates the bandwidth.
of frequencies that will pass through the state observer [13]. This has been presented in a
generalized form in both continuous and discrete forms [14].

The bandwidth of the ESO determines how quickly the disturbance estimation can
track these states [19]. As bandwidth increases, the states will approach the actual values
resulting in the enforced plant response of a cascaded-integral. However, what becomes
of this response when bandwidth is quite limited? It would be reasonable to assume that
the enforced plant would not be cascaded-integral resulting in a controller that is
designed to control the wrong response.

2.2 A Frequency Domain Perspective of the Enforced Plant

If it is known how the actual response of the enforced plant differs from the
cascaded-integral, perhaps there are means of compensating for it. Frequency domain
analysis can be used in order to better understand this difference and to identify
bandwidth limitations that would result in an oscillating or unstable response. This was
previously used to study the entire response of the enforced plant and controller
combined [20, 21]. Frequency analysis can also be used to make adjustments to the
rejecter that would result in additional damping and satisfy stability margins. For these
techniques to be useful, a reasonable process model would have to be known which can
be obtained by relay-tuning identification techniques.

A diagram of the forced plant can be seen in Figure 1. The actual plant is shown
as $G_p$ with its output $y$ and its input as the control signal $u$. All other parts in the diagram
belong to the disturbance rejecter. The gain of the plant is reduced by a parameter $b_0$, which is typically an estimate of the $b$ value from (2.1). The total disturbance estimation ($\hat{f}$) of the ESO is shown as two separate single-input, single-output (SISO) blocks representing transfer functions from $b_0u$ to $\hat{f}$ and from $y$ to $\hat{f}$. The combined signal is subtracted from the input of the enforced plant ($u_0$).

![Block diagram of enforced plant](image)

Figure 1: Block diagram of enforced plant

When the state-observer matrices are converted into complex-frequency domain, transfer functions can be realized that convert signals from $b_0u$ and $y$ into each of the estimated states. Equations (2.2) and (2.3) show the general form transfer functions that generate each of the states, where $M_i$ is a $(1, n+1)$ row vector containing all zeros except for the $i^{th}$ column. The estimated disturbance $\hat{f}$ is found in the extended state when $i=n+1$.

$$\frac{\dot{z}_{i}}{b_0u} = M_i C (sI - A)^{-1} B \left[ \begin{array}{c} 1 \\ 0 \end{array} \right]^T$$  \hspace{1cm} (2.2)

$$\frac{\dot{z}_y}{y} = M_i C (sI - A)^{-1} B \left[ \begin{array}{c} 0 \\ 1 \end{array} \right]^T$$  \hspace{1cm} (2.3)

When parameterized ADRC [13] is used, $C$ becomes the identity matrix in an $n+1$ square. The $B$ matrix contains two columns where the left contains all zeros except for
the $n^{th}$ position where it is equal to one. This value is typically $b_0$, but it is factored out here to use with the diagram in Figure 1. The right column is the observer gain vector $L$ created by the $n+1$ row of Pascal’s triangle (without the first element) with ascending orders of the observer bandwidth parameter $\omega_o$ as shown in (2.4). The $A$ matrix is an $(n,n)$ identity matrix with an additional row of zeros at the bottom and $-L$ used as an additional column to the left. The $A$, $B$, and $C$ matrices can be seen in (2.5), where $\beta_i$ are the elements of $L$.

$$L = \begin{cases} \begin{bmatrix} 2\omega_o & \omega_o^2 \end{bmatrix}^T, & n = 1 \\ \begin{bmatrix} 3\omega_o & 3\omega_o^2 & \omega_o^3 \end{bmatrix}^T, & n = 2 \\ \begin{bmatrix} 4\omega_o & 6\omega_o^2 & 4\omega_o^3 & \omega_o^4 \end{bmatrix}^T, & n = 3 \\ \begin{bmatrix} 5\omega_o & 10\omega_o^2 & 10\omega_o^3 & 5\omega_o^4 & \omega_o^5 \end{bmatrix}^T, & n = 4 \end{cases}$$

$$A = \begin{bmatrix} -\beta_1 & 1 \\ \vdots & \ddots \end{bmatrix}, B = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}, C = I$$

Parameterized ADRC is designed to place all observer poles at $-\omega_o$. This allows the transfer functions that produce the estimated total disturbance to reduce to the equations shown in (2.6) and (2.7).

$$\frac{\hat{f}}{b_0 u} = \frac{-1}{(s/\omega_o + 1)^{n+1}}$$

$$\frac{\hat{f}}{y} = \frac{s^n}{(s/\omega_o + 1)^{n+1}}$$
The diagram in Figure 1 can then be written as the transfer function of the enforced plant $G_p$ from input $u_0$ to output $y$ as shown in (2.8). A direct path can be seen as $G_p(s)b_0^{-1}$. The feedback path contains the sum of $f/b_0u$ and $(f/y)G_p(s)b_0^{-1}$. Note that this feedback path contains an $n+1$ order low-pass filter with its corner frequency at $\omega_o$.

$$
\frac{G_p(s)}{u_0(s)} = \frac{G_p(s)b_0^{-1}}{1 + \frac{G_p(s)b_0^{-1}s^n}{(s/\omega_o + 1)^m}}
$$

(2.8)

### 2.3 The Effect of Observer Bandwidth upon the Enforced Plant

Since a low-pass filter has approximately unity gain for frequencies much smaller than the corner frequency, and has nearly zero gain for frequencies much higher, (2.8) can be rewritten as the approximation in (2.9). From this, it can be seen that the enforced plant should respond like a cascaded-integral for frequencies considerably below the observer bandwidth. Ideally, if there were no limitations upon this bandwidth allowing an infinite bandwidth to be chosen, the enforced plant would be cascaded integral for all frequencies regardless of what the actual plant response. Obviously, this condition is not possible due to sampling time, computing power, and stability concerns.

$$
\bar{G}_p(j\omega) = \begin{cases} 
1 & \omega \ll \omega_o \\
G_p(j\omega)b_0^{-1} & \omega \gg \omega_o 
\end{cases}
$$

(2.9)
When observer bandwidth approaches zero, (2.9) would indicate that the enforced plant should behave like the actual plant multiplied by a constant gain. What the equation does not show is what the response will be for the bandwidths in-between. Equation (2.8) can be rewritten as (2.10) to discover how the response changes from a cascaded-integral at low frequencies to the plant response at high frequencies. Here, the $n^{th}$ order derivative in the denominator has only a low-pass filter on it, whereas the inverse plant is going through something very similar to a first-order high-pass filter. So all finite frequencies will contain some of the cascaded-integral and some of the plant. As increasing frequencies pass through the enforced plant, the response would have an increasing response from the plant and a decreasing response from the cascaded-integral. When the corner frequency of the filters increases, a wider frequency range the enforced plant will respond more like the cascaded-integral and less like the original plant.

$$\tilde{G}_p(s) = \frac{1}{s^n \left( \frac{s}{\omega_o} + 1 \right)^{n+1} + G^{-1}_p(s) b_0 \left[ \frac{(s/\omega_o + 1)^{n+1} - 1}{(s/\omega_o + 1)^{n+1}} \right]}$$

(2.10)

Figure 2: Effect of increasing observer bandwidth
In the example shown in Figure 2, a second-order plant is under the effect of the disturbance rejecter with various observer bandwidths. The parameter $b_0$ was selected so that the high frequency response of the enforced plant will match that of the cascaded-integral regardless of bandwidth chosen. As bandwidth increases, the response of the enforced plant behaves more like the cascaded-integral.

Equation (2.8) and Figure 2 both show that higher observer bandwidth will result in an enforced plant that is closer to a cascaded-integral response. However, all finite bandwidths will allow the original plant to have some contribution to the response typically resulting in less than desirable control. If a reasonable model can be discovered for the plant, frequency domain techniques could be incorporated into the design of the disturbance rejecter in order to improve how well the enforced plant mimics a cascaded-integral.
CHAPTER III

FINDING ROOT LOCATIONS OF THE ENFORCED PLANT

It may be beneficial to the understanding of ADRC to visualize how the roots of the enforced plant move about the s-plane as the tuning parameters vary. The root locus technique is a graphical method for sketching the paths that the roots will move in as a single parameter is changing. This method was introduced by Evans in 1948 [47] and has been developed and utilized extensively in control engineering practice [55-57]. This provides the engineer with a measure of the sensitivity of the system roots to a change in a tuning parameter. By understanding of how the root paths change with parameters, it can usually be determined what tuning changes are necessary in order to achieve the desired results [58].

If the enforced plant has been made to have the response of an $n^{\text{th}}$ order cascaded-integral, then frequency analysis of the roots should show only $n$ poles at the origin and no zeros. Equations (2.9) and (2.10) would give the impression that an infinite bandwidth should satisfy this, whereas a zero bandwidth would have roots that match the plant itself. It would seem likely that these roots move gradually with bandwidth in order to satisfy both extremes. Perhaps knowing how these roots move will provide some insight as to how to tune the observer in order to reduce oscillation.
3.1 Guidelines for the Enforced Plant Roots

If when bandwidth is infinite, the response is allowed to have only \( n \) number of poles, and the same number of poles arrived at the origin from their original plant location, then the plant would only be allowed to have \( n \) number of poles. However, the disturbance rejecter is able to work in configurations where this is not true which would indicate that there must be something else happening to the roots of the enforced plant as bandwidth is changing. Either there are additional poles that at one extreme are having their effects cancelled by zeros, or are approaching negative infinity to have their effect reduced to nothing. The contribution of a pole that is approaching negative infinity would have the same effect as an impulse having no contribution to the response which can be seen by taking the limit of the time-domain response of a single pole as shown in (3.1), where \(-a\) is the pole location and \( \delta(t) \) is the impulse function. Since the convolution of a function with an impulse is the same as the function, (3.1) would have no effect on the response.

\[
\mathcal{L}^{-1}\left(\frac{a}{s+a}\right) = \lim_{a \to \infty} e^{-at} = \begin{cases} 
1, & t = 0 \\
0, & t > 0
\end{cases} = \delta(t) \tag{3.1}
\]

In order to figure out where the roots will be, equations must be created that solve for zeros and poles. Say that functions \( N \) and \( D \) represent the numerator and denominator of the \( s \)-domain transfer function model of the gain adjusted plant such that \( \frac{G_p b_0^{-1}}{s} = \frac{N}{D} \) assuming that this plant is a linear, time-invariant function. Equation (2.8) can be rewritten as (3.2), where \( \omega_0 \) is the parameterized observer bandwidth and \( \widetilde{G}_p \) is the enforced plant.
\[ \bar{G}_p = \frac{N(s/\omega_o + 1)^{n+1}}{Ns^n + D(s/\omega_o + 1)^{n+1} - 1} \] (3.2)

From (3.2), zeros or poles are found when the numerator or denominator is set to zero, respectively. There will always be \( n+1 \) zeros located at \(-\omega_o\). In addition, any zeros from the plant will also be present in the enforced plant regardless of observer bandwidth. The denominator’s highest order term will be \( n+1 \) plus the highest order of \( D \). If \( q \) is the order of the plant denominator \( D \) and \( p \) is the total number of poles in the enforced plant, then the relationship \( p = q + n + 1 \) can be used to find the poles of (3.2). In cases where \( n \) is selected as \( q \), there will be \( 2n+1 \) poles.

Equation (3.2) may have poles that remain at the origin regardless of bandwidth. When the factor \([(s/\omega_o + 1)^{n+1} - 1] \) is expanded, the ones cancel and leave a single that can be factored out. If \( m \) is the number of poles constantly at the origin, \( G_i \) and \( G_d \) are the number of integrators and derivatives in the plant, respectively, then (3.3) will solve for \( m \). Although there will be \( m \) poles at the origin at all times, there will be \( n \) poles that approach the origin as bandwidth approaches infinity and the two are not necessarily equal to each other. The value \( m \) could easily be smaller than \( n \) since \( n \) could easily be selected as a value greater than \( 1+G_i \) and (3.3) would still force \( m \) to be smaller. Both \( m \) and \( n \) can be equal when there are \( n+1 \) integrators in the plant. However, \( m \) cannot be larger than \( n \). When the transfer function of the plant has been simplified, non-zero value cannot exist for \( G_i \) and \( G_d \) at the same time. Say that \( n > 0 \) otherwise there would be no cascaded-integral to approximate. The minimum of (3.3) would be \( 1+G_i \) unless \( G_i > n > 0 \) in which case \( G_d = 0 \) and \( m \) would be equal to \( n \). When \( n > G_i \), \( G_i = m-1 \), therefore \( n > m-1 \) and since \( n \) and \( m \) must be positive integers, \( m \) cannot be larger than \( n \). As a result, when \( n \)
and $m$ are not equal, there must be additional poles that are moving towards the origin and approach it as bandwidth approaches infinity. The number of poles moving to the origin must be $n-m$.

$$m = \min\left(1 + G_i, n + G_d\right)$$

(3.3)

Since the total number of poles $p$ is $2n+1$, but only $n$ poles are to be at the origin at infinite bandwidth, then there should be $n+1$ poles that are not moving toward the origin. These poles would either need to move toward negative infinity, or to approach zeros in order to cancel their effect. As bandwidth increases, there are also $n+1$ zeros located at $-\omega_o$ that approach negative infinity on the real axis. It may seem reasonable for these additional poles to follow them.

To summarize the discussion of this section, there are several observations about the enforced plant using parameterized ADRC on a linear time-invariant plant:

- Zeros of the plant become zeroes of the enforced plant: $z$
- Additional $n+1$ zeroes located at $-\omega_o$
- Total number of poles: $p=2n+1$
- Number of poles at origin: $m = \min\left(1 + G_i, n + G_d\right)$
- Number of poles moving toward origin: $p_1=n-m$
- Number of poles moving toward negative infinity: $n+1-z$
3.2 Root Analysis of the First-Order System

If a simple, first-order model is used such that \( G_p = \frac{b}{s+a} \), \( n \) is selected as 1, and \( b_0 \) is selected as \( b \), solving for the poles using the denominator of (3.2) can be done with (3.4), where \(-a\) is the pole location. The equation shows that there will be three poles total with one pole constantly at the origin and two poles moving as bandwidth changes.

\[
s + \left[ \frac{1}{\omega_o^2} s^2 + \left( \frac{2}{\omega_o} + \frac{a}{\omega_o^2} \right) s + \left( \frac{2a}{\omega_o} + 1 \right) \right] = 0 \tag{3.4}
\]

Solving for the two moving poles in (3.5) shows that two poles will generally be moving left as bandwidth increases. If the original plant pole is not unstable (i.e. \( a \geq 0 \)), then the two moving poles will converge until the bandwidth reaches a specific value \( (\omega_o=0.25a) \) at which point both poles will be located at \(-0.75a\). As bandwidth further increases, the poles would continue moving leftward as complex conjugates with increasing imaginary components that push them away from the real axis.

\[
s_{2,3} = -\frac{a}{2} - \omega_o \pm \sqrt{\frac{a^2}{4} - a\omega_o} \tag{3.5}
\]

An example of the pole movement is shown in Figure 3 where the plant pole is at -1. There should be \( 2n+1 \) total poles with one constantly at the origin, leaving two to move toward negative infinity. One of these poles must begin at -1 and the other begins at the origin so that at zero bandwidth, the two poles at origin may cancel the two zeros that follow \(-\omega_o\) in order to match the transient of the original plant. The two moving poles converge at \(-0.75\) when bandwidth is 0.25 rad/s. The shape of the pole movement is the same for any stable plant \((a>0)\), and the values scale multiplicatively with the value of \( a \).
Figure 3: First-order pole movement: $a=1$

In cases where $b_0$ does not match the actual $b$ gain from the plant, perhaps due to gain fluctuations or inaccurate model identification, (3.4) would instead become (3.6).

$$s \left[ \frac{1}{\omega_o^2}s^2 + \left( \frac{2}{\omega_o} + \frac{a}{\omega_o^2} \right)s + \left( \frac{2a}{\omega_o} + \frac{b}{b_0} \right) \right] = 0$$  (3.6)

Solving for $s$ in (3.7) shows the pole locations as a function of $\omega_o$ and the ratio of $b/b_0$. As $b$ becomes larger than $b_0$, the $\omega_o^2$ term will become more negative, increasing the imaginary component of the two moving poles. When $b$ is smaller than $b_0$, the $\omega_o^2$ term becomes positive and will eventually overcome the negative value from the $\omega_o$ term at higher bandwidths causing the complex conjugate paths will return to the real axis. However, since the $\omega_o^2$ term is inside the square root and its coefficient is less than one, it will not overcome the -$\omega_o$ term outside the root. Therefore, the rightmost moving pole will not continuously move to the right as bandwidth approaches infinity, although it should move leftward at a far slower rate than the leftmost pole. At lower ratios of $b/b_0$, this seem like one pole is getting stuck near the second convergence point, which will likely slow down performance even at high observer bandwidth tunings.
\[ s_{2,3} = -\frac{a}{2} - \omega_o \pm \frac{a^2}{4} - a\omega_o + \left(1 - \frac{b}{b_0}\right)\omega_o^2 \]  \hspace{1cm} (3.7)

Solving for \( \omega_o \) when the square root becomes zero gives the bandwidths where the pole paths will converge, provided that the solution is positive. This is shown in (3.8). If the plant was stable \((a>0)\) with a gain lower than the estimate \((b<b_0)\), then there will be two positive solutions. If instead the plant gain were higher than the estimate \((b>b_0)\), there would only be one positive solution. These bandwidth values can be substituted into (3.7) to determine the frequency \(\omega_{o,\text{conv}}\) where the poles will converge.

\[ \omega_{o,\text{conv}} = \frac{a}{2} \left( \frac{1 \pm \sqrt{b / b_0}}{1 - b / b_0} \right) \]  \hspace{1cm} (3.8)

The example in Figure 4 shows how the pole paths change for various \(b/b_0\) ratios. Higher ratios appear to bend the paths more toward the imaginary axis. Values less than one will converge at a point, then move apart from each other. As the ratio is reduced, the frequency required in order to bring the roots back to the real axis is also reduced leaving one pole near this convergence point that moves leftward very slowly with increasing bandwidth.

Figure 5 shows an example where the plant pole is unstable and located to the right of the origin. As the ratio between \(b\) and \(b_0\) increases beyond one, the complex conjugate paths again bend toward the imaginary axis. As can be seen from (3.7), the rightmost pole will cross the origin into the left half plane when \(\omega_o=1-2a-b/b_0\). Therefore, observer bandwidth would need to be greater in order for the enforced plant itself to be stable.
Figure 4: First-order pole movement with various $b/b_0$ ratios: $a=1$

Figure 5: First-order pole movement with various $b/b_0$ ratios: $a=-1$

When the plant pole is instead unstable ($a<0$) and $b_0$ matches $b$, a positive value of $\omega_o$ does not exist that would cause the inside of the square root in (3.7) to be less than $a^2/4$. Therefore, the moving poles will both be located on the real axis and the distance between them will increase with bandwidth. When (3.5) is solved for $s=0$ it can be seen that a pole will cross the origin when $\omega_o=-2a$. Bandwidths higher than this will have both poles left of the imaginary axis. If the ratio of $b/b_0$ is less than one with an unstable plant pole, the square root still cannot be negative so the two moving poles would also remain on the real axis. However, the point where the rightmost moving pole would cross the origin would instead be when $\omega_o=-2ab_0/b$. A stable enforced plant would be found at bandwidths higher than this.
3.3 Root Analysis of the Second-Order System

If a third order ESO is used to enforce a plant of second-order, the enforced plant would have five total poles making the equations much more difficult to solve. A computer program can instead be used in order to visualize how these poles are moving with increasing bandwidth. Out of these five poles, two must start where the original plant poles are located and the other three must start at the origin. Only one pole will remain at the origin (unless the plant contains pure integrator) and a second pole should approach it from somewhere. Three poles should try to move towards negative infinity. Since the plots scale with frequency, plants with various damping factor $\zeta$ and scaleable natural frequency $\omega_n$ (3.9) will be shown as examples.

$$G_p = \frac{b\omega_n^2}{s^3 + 2\zeta\omega_n s + \omega_n^2}$$ (3.9)

Figure 6 shows the enforced plant poles for a plant where $\omega_n = 1$ and $0.25 \leq \zeta \leq 1.5$ as $\omega_o$ increases. Underdamped systems will have two poles start at a complex-conjugate position that move left towards negative infinity. This leaves two poles leaving the origin to converge at another point on the real axis where one will move left towards negative infinity and the other will move towards the origin. This point becomes further away from the origin with lower $\zeta$ values requiring higher bandwidth for the slowest non-zero pole to become close to the origin. Since in overdamped systems the original plant poles are already on the real axis, and the one closest to the origin will move towards the origin, higher values of $\zeta$ will have the slowest non-zero pole closer to the origin at the
same bandwidth. The closer that this pole is to the origin, the easier it should be for the disturbance rejecter to approximate a second-order cascaded-integral.

![Figure 6: Second-order pole movement for various $\zeta$, $n=2$, $b_0=b$, $\omega_n=1$](image)

![Figure 7: Second-order pole movement for various $b/b_0$ ratios: $\zeta=1$](image)

Considering only the example where the plant is critically damped, higher ratios of $b/b_0$ will bend the pole paths toward the right as shown in Figure 7. Unlike the first-order example given in Figure 4, ratios higher than 10 may cross over the imaginary axis leading to an unstable enforced plant. Lower ratios cause the paths to bend left toward the negative real axis, so more conservative tunings of $b_0$ should lead to less oscillation.

Similar to the traditional root locus, poles appear to repel each other. Two poles would rather be a complex-conjugate pair than to have the same value. As additional poles are added, the paths of the poles will follow different angles to allow more space between them. If several high-frequency poles are added to simulate the effect of a time-delay system, it would seem reasonable to assume that the effect would push the paths of the two slower poles toward or even past the imaginary axis thereby increasing oscillation at a given bandwidth.
3.4 Summary of Enforced Plant Root Locations

Active disturbance rejection control is designed with the assumption that the response of the enforced plant can be approximated as a cascaded-integral. Although this may be reasonable in some cases, it may not be a safe assumption for situations where high observer bandwidth is not attainable. It is for these conditions where a root locus style plot can be used in order to determine a stable bandwidth range may satisfy a predetermined damping requirement. Likewise, the information about the pole locations can be used to determine a reasonable high limit for controller bandwidth.

The plots shown in this chapter provide the reader with an understanding of the effects of gain fluctuation. The different pole paths generated from various $b/b_0$ ratios demonstrate how higher plant gain will lead to less damping, more oscillation, and potentially instability if the $b_0$ estimate is too low by comparison. Likewise, lower gain ratios can leave an extra pole on the real axis that does not increase as quickly with observer bandwidth thereby hindering performance.

This chapter also provided some basic rules for what the enforced plant poles will be doing based upon the number of plant poles and zeros and the order of the cascaded-integral to approximate. This may also present an interesting dilemma to the reader. If the number of enforced plant poles at the origin is only one unless there are integrators in the plant, then there are likely to be many situations where low frequencies are approximating only a single integral rather than $n^{th}$ order resulting in a significantly lower gain.
CHAPTER IV

GRAPHICAL DESIGN OF THE DISTURBANCE REJECTER

Since no mathematical model can describe the dynamics of a physical system with complete accuracy, it may not be enough simply to determine that a control solution is stable. Rather, it may be more beneficial to ensure that the solution is stable as long as the physical process does not deviate from its estimated model by a predetermined amount. Frequency analysis provides the ability to determine and visualize the relative stability of a controlled system to know deviation is allowed before encountering instability. A stability criterion was introduced by Nyquist to visualize how to adjust tuning parameters in order to improve the controlled system relative stability [59]. This had become a very popular method to determine the stability of the controlled system [60].

This chapter will focus on using the Nyquist plot to measure the stability of the approximate, cascaded integral formed by the disturbance rejection loop.
4.1 Finding an Open-Loop Gain for Nyquist Stability Analysis

The Nyquist plot requires the complex outputs of an open-loop function as the result of every point of the Nyquist contour is used as inputs. This open-loop function $G_{ol}$ must satisfy the condition such that the closed-loop $G_{cl}$ is formed by (4.1). The closed-loop equations for the disturbance rejection loop given thus far do not fit this form. As a result, an equivalent equation must be given that does.

$$G_{cl} = \frac{G_{ol}}{1 + G_{ol}}$$  \hspace{1cm} (4.1)

The disturbance estimation can be seen as the sum of two signals generated by the ESO. When the input factor $[b_0\mu 0]^T+[0 y]^T$ in (4.2) is distributed, the extended state $z_{n+1}$ will be sum of each signal just as the superposition theorem would state. A block diagram can be shown (Figure 8) such that the two signals are subtracted from $u_0$ separately. The signal from input $b_0\mu$ does not pass through the physical process before it enters the ESO and can be simplified into a closed loop.

$$z_{n+1} = \begin{bmatrix} 0 & \cdots & 1 \end{bmatrix} (sl-A)^{-1} B \begin{bmatrix} b_y u & 0 \end{bmatrix}^T + \begin{bmatrix} 0 & y \end{bmatrix}^T$$  \hspace{1cm} (4.2)

Figure 8: Simplifying block diagram
Let $F_u$ be the transfer function from $b_0u$ to the disturbance estimation $f$, $F_y$ be the function from $y$ to $f$, and $F_1$ be the simplified closed-loop. The function $F_u$ will have a negative sign when $A$ and $B$ are built as a parameterized, linear ESO. The output of $F_u$ is added with the output of $F_y$ and the sum is subtracted from the modified control signal ($u_0$). Therefore, the closed-loop can be written as a positive feedback loop with $-F_u$ in the feedback path. Simplifying the closed-loop $F_1$ using the parameterized ESO will give (4.3).

$$F_1 = \frac{1}{1+(-F_u)} = \frac{F_u^{-1}}{F_u^{-1} - 1} = \frac{\left(\frac{s}{\omega_o} + 1\right)^{n+1}}{\left(\frac{s}{\omega_o} + 1\right)^{n+1} - 1} \quad (4.3)$$

With the simplification of $F_1$, only $F_y$ remains in the feedback path of the closed-loop from $u_0$ to $y$. A form that satisfies (4.1) can be obtained if $F_y^{-1}$ is factored out of this loop. This can be seen in (4.4). Since $F_y^{-1}$ contains only $n$ poles at the origin and no right-half plane poles regardless of observer bandwidth selection or ESO order, this factor cannot cause the $y/u_0$ function to become unstable and therefore does not need to be analyzed. Rather, the Nyquist plot of $G_{p_0}b_0^{-1}F_1F_y$ would be able to show stability measurements.

$$\frac{y}{u_0} = \frac{F_1 G_{p_0} b_0^{-1}}{1 + F_1 G_{p_0} b_0^{-1} F_y} = \frac{1}{F_y} \frac{F_1 G_{p_0} F_y^{-1}}{1 + F_1 G_{p_0} b_0^{-1} F_y} \quad (4.4)$$

The multiplication of $F_1F_y$ will simplify and contain $n$-1 derivatives due to a single $s$ factoring out of the denominator as seen in (4.5). As a result, (4.5) can be substituted into (4.4) to solve for $y/u_0$ in (4.6) which contains the right factor in the form of (4.1).
\[
F_y F = \frac{(s / \omega_o + 1)^{n+1}}{(s / \omega_o + 1)^{n+1} - 1} s^n = \frac{s^n}{(s / \omega_o + 1)^{n+1} - 1}
\]  
(4.5)

\[
y = \frac{(s / \omega_o + 1)^{n+1}}{s^n} \left[ \frac{G_p b_0^{-1} s^n}{(s / \omega_o + 1)^{n+1} - 1} \right] + \frac{G_p b_0^{-1} s^n}{(s / \omega_o + 1)^{n+1} - 1}
\]
(4.6)

For brevity, the open-loop gain of the right factor in (4.6) will be referred to as \( G_1 \). Rather than having \( n+1 \) poles located at \(-\omega_o\) in addition to those contributed by \( G_p \), this open-loop function instead has \( n+1 \) poles located on a circle centered at \(-\omega_o\) with a radius of \( \omega_o \) such that it intersects the origin where one of the poles are found. This pole at origin will cancel one of the zeros at the origin, leaving a net derivative order in \( G_1 \) of \( n-1 \). This can also be seen since \( s / \omega_o \) can be factored out of the denominator of \( G_1 \) once it is expanded and the ones cancel. The open-loop gain can be rewritten as (4.7) to better understand how observer bandwidth will change the loop shape.

\[
G_1 = \frac{G_p b_0^{-1} s^n}{(s / \omega_o + 1)^{n+1} - 1} = G_p s^{n-1} \left[ \frac{\omega_o}{b_0 (n+1)} \right] \frac{\omega_o^n (n+1)}{\prod_{k=1}^{n} \left[ s + \omega_o \left( 1 - e^{i2\pi k/(n+1)} \right) \right]}
\]  
(4.7)

At low frequency, the plot of \( G_p s^{n-1} \) is multiplied by a gain of \( \omega_o b_0^{-1} (n+1)^{-1} \). The result is shaped by a low-pass filter with unity gain at low frequency. The filter contains complex poles located on the circle radius \( \omega_o \) centered at \(-\omega_o\) with the exception of the origin since it has been factored out. It can be seen from (4.7) that the parameter \( b_0 \) only changes the size of the plot, whereas \( \omega_o \) changes the size and the loop shape. Improved
stability can be found with larger values of $b_0$ and smaller values of $\omega_o$ although this would obviously sacrifice performance.

If $n$ is selected as one, the low-pass filter becomes first order with a corner frequency of $2\omega_o$ and the value of the low-frequency gain becomes $0.5\omega_o/b_0$. When $n$ is two, low-frequency gain is $(1/3)\omega_o/b_0$ with complex poles at $-\omega_o(1.5\pm j0.866)$.

4.2 Designing Stability Margins

Generating the Nyquist plot of the open-loop function $G_1$ from (4.7) can be used to find the gain margin (GM) and phase margin (PM) of the function from $u_0$ to $y$ using traditional methods. Tuning $b_0$ alone is enough to adjust the GM since it changes the size of $G_1$ without changing its shape. Both $b_0$ and $\omega_o$ should be tuned in order to adjust the PM.

Perhaps there is a desired observer bandwidth that can be decided upon based on how rapidly the ESO should respond to a changing disturbance. Equation (4.7) can be plotted without $b_0$ being decided upon to find $b_0$ values that adhere to predetermined stability margin requirements. By generating the Nyquist plot for (4.8), minimum values for $b_0$ can be found at radius $\rho$ that satisfy various phase margins at angle $\theta$. If a ray is drawn from the origin at an angle equal to the phase margin requirement, it may intersect the plot at multiple locations. The largest radius that intersects the plot is the value of $b_0$ necessary in order to provide the phase margin. When a ray is drawn at $0^\circ$, the maximum radius that intersects is the value of $b_0$ that would cause the enforced plant to be critically
stable, so multiplying this by the required gain margin will give the value necessary to adhere to it. The larger of the two discovered values of $b_0$ should be used to satisfy both stability margin requirements. The outermost path of (4.8) is the set of points equal to the value of $b_0$ required in order to satisfy phase margins of the angles of each point.

$$b_0 = \rho e^{j\theta} = -\frac{G_p s^n}{(s / \omega_o + 1)^{n+1}} - 1$$  \hspace{1cm} (4.8)$$

Figure 9: Determining $b_0$ to satisfy stability margins

As an example, say that the response of $G_p$ is best approximated by a second-order response with small time delay. Let $G_p = \exp(-0.1s) / (s^2+2s+2)$, $\omega_o = 100$, and $n=2$. Figure 9 shows the plot of equation (4.8) with rays shown at 30, 45, 60, and 75 degrees. Points are labeled that intersect the plot. Note that the labels are only on the points furthest from the origin on each ray. In order to determine the minimum value of $b_0$ that will satisfy a gain margin of 30, 40, 60, or 75 degrees, the radius is measured between the origin and the point of intersection. In this example, $b_0$ must be at least 3 in order for the disturbance rejection loop to satisfy a phase margin of 30 degrees. It would need to be at least 3.6, 4.6, or 5.9 to satisfy phase margins of 45, 60, and 75 degrees, respectively.
4.3 Satisfying a Damping Factor of Closed-loop Poles

As discovered by root locus, the enforced plant is only an approximate, cascaded integral. Additional poles exist other than at the origin resulting in slower response and likely oscillation. Higher bandwidths typically improve response time but may not improve an oscillatory condition. In some cases, increasing bandwidth may make oscillation worse. A condition may be required to ensure that as the parameters are being tuned, the poles of the enforced plant are within a specified angle of the negative real axis to support a predetermined damping ratio.

When the Nyquist plot is generated, it tests if the closed-loop will contain unstable poles by defining the input as a contour that enclose all points in the right-half plane. If the input is changed such that it encloses all points that violate a damping ratio angle, then the Nyquist plot could be used to verify that the closed-loop adheres to this damping ratio over a range of gain and phase fluctuations. If the original plant contained a number of poles that did not adhere to the damping requirement, then the modified Nyquist plot would need to show a counterclockwise encirclement count to match.

By taking a LaPlace domain transfer function and replacing complex frequency $s$ with $\omega \cdot \exp(j\theta)$, the traditional Nyquist plot is obtained when the angle $\theta$ is $\pi/2$ placing the input to the imaginary axis and evaluating all values for $\omega$. A stable result means that all poles exist at an angle that is less than or equal to $\pi/2$ from the negative real axis with respect to the origin. To instead satisfy a damping ratio angle, $\theta$ would instead be the difference between $\pi$ and this angle. This can be seen in (4.9) with $\theta_d$ as the damping ratio angle.
\[ s \rightarrow \omega e^{j(\pi - \theta_d)}, \text{ where } 0 < \theta_d \leq \pi / 2 \]  \hspace{1cm} (4.9)

The traditional Nyquist contour is shown in Figure 10 as a path that surrounds every finite value with a positive real component as radius \( \varepsilon \) approaches zero. The modified contour is shown in Figure 11 additionally including values that are not within angle \( \theta_d \) of the negative real axis. The two contours are identical when the damping ratio angle is \( \pi / 2 \) or 90 degrees.

![Figure 10: Traditional Nyquist contour](image1)
![Figure 11: Modified Nyquist contour](image2)

With the open-loop transfer function reduced to a polynomial numerator and denominator, a conversion can be made by changing the coefficients so that a typical Nyquist plot program can evaluate on angle \( \pi / 2 \) but produce the results of the original function evaluated at \( \theta \) assuming that the plotting program can handle a function that contains complex numbers that are not necessarily conjugates. Since \( s \) will be replaced by \( \omega \exp(j\pi / 2) \) but it is desired to generate a plot as if \( s \) were replaced by \( \omega \exp[j(\pi - \theta_d)] \), the ratio of the two replacements in (4.10) will be used to multiply the coefficients of the transfer function, where \( s_c \) is what \( s \) would be replaced with before it is evaluated by software.
\[ s_c = \frac{\omega e^{j(\pi-a_j)}}{\omega e^{j(\pi/2)}} = e^{j(\pi/2-a_j)} \] (4.10)

If \( P \) is a column vector of numerator or denominator coefficients \( a_i \), the converted vector of coefficients \( P_c \) can be obtained by multiplying by a row vector of descending order of the ratio of replacements as in (4.11).

\[
P_c = \left[ a_m \ a_{m-1} \ \cdots \ a_1 \ a_0 \right] \left[ s_c^m \ s_c^{m-1} \ \cdots \ s_c \ 1 \right]^T \] (4.11)

Once each polynomial is converted, the traditional Nyquist plot software can be used to generate the plot. Something to note is that the number of encirclements alone do not give the number of poles that violate the damping ratio angle in the closed-loop. Rather, it is the difference between these encirclements and the number of violations in the open-loop function. Just as traditional Nyquist plot stability [59], (4.12) still applies except that \( N \) is the number of counterclockwise encirclements, \( P \) is the number of open-loop violations to the damping ratio angle, and \( Z \) is the number of closed-loop violations.

\[ Z = P - N \] (4.12)

As an example, say that an open-loop function of \( 1/(s+1)^2 \) is to be evaluated at to see if the closed-loop poles are within a damping ratio angle. The actual closed-loop poles have two located at -1 and a complex conjugate pair at \(-1 \pm j\), which are 45 degrees from the negative real axis. If the angle is chosen as 60 degrees (\(\pi/3\) radians) and solving for the exponent needed of \(\pi/6\), the coefficients of the denominator would be multiplied by \([\exp(2\pi/6), \exp(\pi/6), 1]^T\) yielding the converted function seen in (4.13), where \(G_{60}\) is the converted transfer function allowing evaluation of a 60 degree damping angle.
\[
G_{60} = \frac{1}{e^{j2\pi/6}s^2 + 2e^{j\pi/6}s + 1}
\]

(4.13)

If the same process is followed for 50 and 45 degrees, the plots in Figure 12 can be generated. The plot for 90 degrees is simply the traditional Nyquist plot of the open-loop function. As the damping ratio angle is reduced from 90 to 45 degrees, the plot grows with an additional twist on the negative real axis until it intersects the (-1, 0) stability point which is expected since the angle of the closed-loop poles are known. Choosing an angle less than 45 degrees will generate a plot that encircles the stability point twice as it should since there would be two closed-loop poles with a damping ratio angle that is greater than what was selected for the plot.

![Figure 12: Nyquist plots evaluated at various damping ratio angle contours](image)

Margins can also be seen for this example by viewing Figure 12. In the case where 45 degrees was the target damping ratio angle, the plot intersects the (-1, 0) point indicating that there is no additional gain or phase lag tolerable in the plant that would meet the target. When the target is 50 degrees, the plot intersects the real axis at about -0.7 so the plant gain can fluctuate up to 1.42 before violating the target damping ratio angle. Likewise, the plot intersects the unit circle at about 160 degrees so the plant may
encounter additional phase lag of 20 degrees before violating the target angle. The plot for 60 degrees has increased margins. The plant gain can increase by 3 and the phase may lag by 60 degrees.

4.4 Solving for a Minimum Bandwidth to Guarantee Stability

Using the Nyquist plot, it is fairly straightforward to find a value of $b_0$ that satisfies margin requirements if the observer bandwidth has already been selected. It is more complicated to find a bandwidth that will still provide adequate low-frequency gain while adhering to these margins. The high-frequency gain $b_0$ can be identified within the process model and used for $b_0$ as a starting point. On the contrary, a bandwidth may be infinitely large or small in continuous time making it difficult to know where to start tuning it. Bandwidth may be limited by sampling time assuming that hardware has already been selected for sensing output measurements and for processing the information. If a minimum safe bandwidth can be determined, it can be used as a starting point for tuning and testing can be made within a range between maximum and minimum observer bandwidths.

Nyquist plots test stability by counting encirclements of -1 of an open-loop function, but when this function is instead shifted right by one, the encirclements may be counted at the origin. Both one and the open-loop function are terms of a characteristic equation when set to zero. There is a relationship between the two terms that can be used as a conservative method to guarantee stability if the open-loop function is already stable.
Imagine that there is an old lady walking her dog with an adjustable leash in some path around a tree. The dog wanders around somewhat randomly in whatever direction that the grass carries the most interesting smell. When the leash is short and the lady keeps her distance from the tree, the two of them make a successful stroll. If at any point the leash becomes longer than the distance between the lady and the tree, and the dog decides there is something interesting on the other side, then the possibility exists that the dog will wrap its leash around the tree. Should this happen, the dog has encircled the tree a different number of times than the lady has.

Say that the tree is the origin of the complex plane. The path that the lady walks is the output of a function in which the number of encirclements is known. The dog’s position relative to the lady is from another function. The dog’s actual position would be the vector sum of these two functions. When the distance of function #2 (length of the leash) is kept shorter than the distance of function #1 (distance between the lady and the tree), then the vector sum must have the same number of encirclements as function #1.

Say that a plot of $g(s)$ is made with may have $n$ number of encirclements around the origin. Another function $h(s)$ is to be added to $g(s)$ and it is desired to know if the number of encirclements of $g+h$ is also $n$. As each point of complex $s$ is evaluated, solutions for $g(s)$ and $h(s)$ are added together with vector addition. If the distance between the solution of $g(s)$ and the origin is larger than the distance between $h(s)$ and the origin, then $h(s)$ is unable to bring the $g(s)$ point to the other side of the origin by the addition. Therefore, if the magnitude of $g(s)$ is greater than the magnitude of $h(s)$ for all $s$ being evaluated, the number of encirclements of $g+h$ must be $n$. 
The characteristic equation $q(s)$ of (4.6) consists of two terms that can be rewritten to separate the tuning parameters of $b_0$ and $\omega_o$ as shown in (4.14). The left term $(s/\omega_o+1)^{n+1}$ will have a net encirclement count of zero since it does not contain a discontinuity. If the magnitude of this term is greater at all frequencies than the right terms $\left(G_p b_0^{-1} s^n - 1 \right)$, then the net encirclement of the characteristic equation is zero.

$$q(s) = \left(\frac{s}{\omega_o} + 1\right)^{n+1} + \left(G_p b_0^{-1} s^n - 1 \right) = 0 \quad (4.14)$$

An example of the magnitude comparison between these two parts is shown in Figure 13 using a two-inertia plant model [61, 62] and a 5th order ESO [63]. Since $b_0$ has been derived from the plant model, the right terms are plotted and considered fixed in place. The left term is shown at various bandwidths. As bandwidth increases, the plot of the left term shifts to the right in the figure. The bandwidth of 225 rad/s appears to be the highest value that does not allow the plot of the right terms to exceed its magnitude at any frequency. As a result, all bandwidths less than 225 rad/s are stable. However, this does not mean that higher bandwidths are unstable. Rather, the critically stable bandwidth would be found somewhere above this value making the value a starting point for tuning.

![Figure 13: Finding bandwidth limitation](image-url)
As observer bandwidth increases, the plot of the left term of (4.14) moves to the right bringing more of the plot closer to a gain of unity. As the bandwidth approaches infinity, the magnitude of the right terms would need to be less than unity as shown in (4.15) in order to guarantee that the encirclement count of the characteristic equation is zero. This indicates that its Nyquist plot would need to be contained within the unit circle. Therefore, if the plot of $G_p b_0^{-1} s^n$ is contained within a circle of radius 1 and centered at (1,0) then the characteristic equation (4.14) will not contain encirclements about the origin at an infinite bandwidth. Since a conservative stability can be found within this circle, then as long as the plot of $G_p s^n$ is contained to finite points within quadrant I and IV, then a finite, non-zero value of $b_0^{-1}$ exists that will contain it within the circle.

\[
\left| G_p b_0^{-1} s^n - 1 \right| < 1 \tag{4.15}
\]

Knowing that a bandwidth stability limitation is avoidable if $G_p s^n$ is contained within quadrants I and IV and a high enough value of $b_0$ is used, this presents a target to selecting the cascaded integral order $n$. If the phase of $G_p$ can be contained within a total of 180 degrees, then a value of $n$ would allow for $G_p s^n$ to be kept within the first and fourth quadrant. When $n$ is selected too high, the plot of $G_p s^n$ will enter the second quadrant in the lowest frequencies. When $n$ is selected too low, the high-frequency phase lag will push the plot into the third quadrant.
CHAPTER V

NYQUIST PLOTS WITH INFINITE RADIUS

Computer software does well to represent the Nyquist plot for models where all magnitudes are finite values. Plots are typically generated with a linear scale with finite axes that show only a range of possible values. Not all system models can be represented by only finite magnitudes leaving some parts of the Nyquist plot that cannot be displayed regardless of how much the user attempts to zoom out of the graph.

If encirclements could only be created by finite magnitudes, the current methods of plotting would be sufficient in order to determine stability. When encirclements are created by an infinite magnitude, the encirclement cannot be seen on a Nyquist plot that uses linear axes, thereby losing the advantage that it holds over the traditional Bode plot of magnitude and phase. This chapter will discuss how computer software can deal with these types of encirclements and display them in order for stability to be properly analyzed.
5.1 Measuring the Angle of an Infinite Radius Arc

When the magnitude of a frequency response becomes infinite, the phase response becomes discontinuous [48]. It is because of this discontinuity that the Nyquist plot must contain an arc at an infinite radius to connect the endpoints. The discontinuity may be caused by imaginary poles (with no real component), which can be seen on the Bode plot since the discontinuity occurs at the location of the imaginary frequency. Discontinuities may also be caused by either poles or zeros at the origin, which are invisible on a typical Bode phase plot. By first making a connection between the Bode phase plot discontinuity and the Nyquist plot infinite radius arc of a pair of imaginary poles, an argument can be made for displaying Bode for the entire infinite range of frequencies in order to make the same connection with other systems where this discontinuity is not normally visible. Figure 14 demonstrates this by showing the magnitude and phase response over positive and negative frequencies. The discontinuity for the response with imaginary poles can be seen on the plots. However, the other two responses have discontinuous phase in the middle of the two plots and again when connecting the outsides of the two plots. The phase discontinuities that coincide with infinite magnitudes should be considered in order to determine the angular distance of the infinite radius arc on the Nyquist plot.

A single imaginary pole will cause a semicircle of 180° around it on the Nyquist contour, which will create a 180° semicircle with infinite radius on the Nyquist plot. When looking at the phase plot of positive frequencies, the limit of the phase is +90° as frequency approaches from the left and -90° as it approaches from the right. This can be seen since the equation of an imaginary pole with \( s \) replaced by \( j \omega \) will change signs.
depending on whether $\omega$ is greater than or less than the imaginary value of the pole as shown in (5.1), where $a$ is the imaginary value of the pole location, $\omega$ is frequency as it approaches $a$ from larger and smaller values, and “arg” is the argument function that provides the angle of the complex value. Measuring the change in phase across the discontinuity of $-180^\circ$ as frequency is increasing matches the arc angle and clockwise direction on the Nyquist plot.

Figure 14: Phase discontinuities over positive and negative frequencies

$$
\theta = \arg \left[ \lim_{\omega \rightarrow a} \frac{1}{j(\omega - a)} \right] = \begin{cases} 
\pi / 2, & \omega \rightarrow a^- \\
-\pi / 2, & \omega \rightarrow a^+
\end{cases} \quad (5.1)
$$

If the pole happens to be at the origin, this discontinuity occurs when frequency is zero. The Nyquist plot uses both positive and negative frequencies so both sides must be evaluated. Positive frequency shows a phase at $-90^\circ$ so negative frequencies must have a phase of $+90^\circ$ as shown in (5.2). The difference between the two limits obtained as frequency approaches zero is $-180^\circ$ just as it was for an imaginary pole.

$$
\theta = \arg \left[ \lim_{\omega \rightarrow 0} \frac{1}{j\omega} \right] = \begin{cases} 
\pi / 2, & \omega \rightarrow 0^- \\
-\pi / 2, & \omega \rightarrow 0^+
\end{cases} \quad (5.2)
$$
A zero at the origin will have similar behavior except that a discontinuity occurs as frequency approaches infinity and the sign of the difference is opposite that of the pole. Since the set of inputs for the Nyquist plot is a continuous contour, it can be viewed as the imaginary axis bent into a circle of infinite radius so that the two opposite ends are joined together where the order of inputs can only travel in one direction. Therefore, if infinity is viewed as a single point, a measurement of this discontinuity can be obtained by taking the limit at this point from both directions. Equation (5.3) explains this more properly by taking the limit at zero from both directions of the reciprocal of frequency showing that it is discontinuous. Since the difference is +180°, this would generate a counterclockwise arc at infinite radius.

\[
\theta = \arg \left[ \lim_{\omega^{-1} \to 0} j\omega^{-1} \right] = \begin{cases} 
-\pi / 2, & \omega^{-1} \to 0^- \\
\pi / 2, & \omega^{-1} \to 0^+ 
\end{cases} \quad (5.3)
\]

5.2 Mapping Infinite Scales to Finite Axes

Since the angle of the infinite arc can be determined simply by obtaining the instantaneous change in phase as the frequency travels in the proper direction around the circle, what remains is a method to allow this information to be put on a flat, two-dimensional screen or page. Any number of functions \( f(\bullet) \) can be created that have a discontinuity over a finite set of real numbers and plenty of these will have solutions that are infinite as the input approaches the discontinuous point. Of these functions, some may map an input value of zero to a solution of zero such that: \( f(0) = 0 \). Likewise, the
derivative of the function must be positive within the input domain between zero and the discontinuous point to maintain a one-to-one relationship between input and output points. Assuming that there are no other discontinuities within this domain, the input can be seen as the values of the finite axis whereas the output will represent an infinite scale.

In some cases, an axis would also need to represent a negative infinite point. These axes would require a function that contains two discontinuities at input values that are opposite signs. Likewise, approaching the negative input from zero must yield an output that approaches negative infinity. The input domain would then range between the two discontinuities with zero at its center and the function must remain positive over this domain. This case will be the focus of this chapter since these functions will work whether negative values are needed or not. Likewise, the aforementioned functions will be discussed as their inverses to discover how to change an infinite scale into a finite axis.

If preserving linearity over a given range is critical, a piecewise function can be constructed that satisfies the previously mentioned conditions. Say that half of the finite axis is to be used for the range that will be linearly mapped leaving the other half for a nonlinear relationship. The functions \( x \) and \( 1/x \) can be used to satisfy both halves of the axis where \( x \) will refer to the values on the infinite scale. Half of the axis is represented linearly when \( x \) is between -1 and 1 with a simple function of \( f(x) = x \). When \( x \) is larger than 1, the finite axis value can be represented by \( f(x) = 2-1/x \) intersecting the other function when \( x=1 \) and maintaining the positive derivative \( f'(1) = 1/x^2 \). When \( x \) is smaller than -1, the function \(-2-1/x\) can be used to intersect -1 and have the same
derivative as its positive counterpart. This places the value of 1 as the midpoint on the axis between zero and infinity.

The resulting finite axis will have all values identical between -1 and 1. Half the distance between a number and zero will represent half of that number. Outside of this range, half the distance between a number and the closest terminus will be twice that number. This places each number and its reciprocal equidistant to the value of one with the same sign as the number. This is shown in Figure 15 where x is shown as the values of infinite range and f(x) is the value that it is scaled to.

![Figure 15: Piecewise reciprocal scale](image)

The user probably shouldn’t be expected to convert the axis values into the actual ones, so the markers should instead indicate the value from the infinite scale. These markers could be placed at the halfway points mentioned due to their relationship, though their positions obviously are not critical as long as the values listed match the location.

The functions mentioned only maintain a linear range between -1 and 1. This range can be selected as whatever is appropriate. If the finite axis is kept between -2 and 2, then the values between -1 and 1 can represent a different linear range if the function is scaled by a constant. This range can be represented by the function \( f(x) = \frac{x}{x_1} \) where \( x_1 \) is the value on the infinite scale that maps to the point 1 on the finite axis. The functions containing the reciprocal \( 1/x \) can have this replaced by \( x_1/x \). The piecewise function would instead be divided up at the points \( \pm x_1 \) as shown in (5.4). Figure 16 shows the relationship using the \( x_1 \) parameter.
If it is not critical to have a given range that preserves linearity, then a single function can be used that represents the entire range. When plotted, the reciprocal method given in (5.4) mildly resembles the shape of an arctangent curve. The arctangent function retains the characteristic where an input and its reciprocal have outputs that are equidistant to an output value, although the value is $\pi/4$ in the case of the arctangent so multiplying this value by $4/\pi$ will bring this point to one. Using the function shown in (5.5), the values -1, 0, and 1 map to themselves and the infinite values map to -2 and 2 matching the results of (5.4) when $x_1$ was selected as one. Figure 17 shows how $x$ and $f(x)$ relate to each other.

$$f(x) = \frac{4}{\pi} \arctan x$$  \hspace{1cm} (5.5)
The halfway point between zero and infinity on the fixed axis can represent a different value by scaling the inside of the arctangent. When the function becomes (5.6), this value becomes $x_1$ such that $f(x_1) = 1$. Although the relationship is different than when using (5.4), the markers are the same as shown in Figure 16.

$$f(x) = \frac{4}{\pi} \arctan \frac{x}{x_1}, \text{ where } x_1 > 0$$ (5.6)

This can be further modified in order to assign a value $x_2$ on the infinite scale that will appear on the axis as the halfway point between $x_1$ and infinity. Within the arctangent, the fraction $x/x_1$ is to be raised by a constant $\beta$. The equation then is to be evaluated at $x_2$ and must be equal to $3/2$ as shown in (5.7).

$$f(x_2) = \frac{4}{\pi} \arctan \left( \frac{x_2}{x_1} \right)^{\beta} = \frac{3}{2}$$ (5.7)

Manipulating the equation and solving for $\beta$ allows for it to be rewritten as (5.8). Negative values of $x$ will need to have their signs preserved in the solution, but removed from within the arctangent. The relationship between $x$ and $f(x)$ with respect to parameters $x_1$ and $x_2$ are shown in Figure 18.

$$f(x) = \text{sgn} x \left[ \frac{4}{\pi} \arctan \left( \frac{\ln|1+x_2|}{\ln(x_2/x_1)} \right) \right], \text{ where } 0 < x_1 < x_2$$ (5.8)

Figure 18: Arctangent scale with parameters $x_1$ and $x_2$
The fixed axis can also be modified so that each half represents a logarithmic scale. The middle would still represent the actual number zero. The right half would be for the entire range of numbers from 0 to infinity but spaced in such a way so that neighboring powers of 10 would be about the same distance apart. The left half would represent negative numbers in the same fashion. Equation (5.5) is first used in order to allow an infinite range of exponents to fit into the right half of the fixed axis by dividing (5.5) in half and adding one. Then, $\log_{10}|x|$ is substituted into $x$ and the sign of $x$ is moved out of the logarithm as shown in (5.9) to prevent it from generating complex results.

$$f(x) = \text{sgn} x \left( 1 + \frac{2}{\pi} \arctan \left[ \log_{10}|x| \right] \right)$$

As shown in Figure 19, this scale places ten halfway between one and infinity and one tenth is halfway between zero and one. Due to the arctangent in the equation, one hundredth would not be located halfway between one tenth and zero, but it would be equidistant to one as one hundred is.

As (5.5), this places $10^0$ at the halfway point between 0 and infinity. The halfway point can represent a different value if $x$ is divided by that value inside the logarithm as in (5.10). This makes the halfway point represent the value $x_1$. Figure 20 shows the relationship between $x$ and $f(x)$ with respect to the $x_1$ parameter for this equation.
The point between $x_1$ and infinity using (5.10) is $10x_1$. The equation can be altered so that a value $x_2$ is instead represented. The logarithm inside the arctangent must be divided by a constant $\beta$ such that $f(x_2) = 3/2$.

$$f(x_2) = 1 + \frac{2}{\pi} \arctan \left( \frac{\log_{10} \left( \frac{x_2}{x_1} \right)}{\beta} \right) = 3/2$$

(5.11)

Manipulating the equation shows that the arctangent is equal to $\pi/4$ meaning that $\beta$ must be equal to $\log_{10}(x_2/x_1)$. Using this to divide the logarithm in (5.10) gives the equation in (5.12). The markers for this relationship are identical to Figure 18 although the values in between will be different.

$$f(x) = \text{sgn} \left( 1 + \frac{2}{\pi} \arctan \left( \frac{\log_{10} \left( \frac{|x|}{x_1} \right)}{\log_{10} \left( \frac{x_2}{x_1} \right)} \right) \right), \text{ where } 0 < x_1 < x_2$$

(5.12)

Using the equations presented in (5.4), (5.8), and (5.12), a finite range of values can represent an infinite range. All three equations include a parameter to control the value that is considered to be the midpoint between zero and infinity. Two of the equations include a parameter to adjust the value that is three quarters of that distance.
5.3 Generating the Infinite Radius Nyquist Plot

The Nyquist plot can either be considered to be in Cartesian or polar coordinates. These appear identical if no equations are used to confine an infinite range to a finite distance. These may also appear identical within the radius of the parameter $x_1$ when (5.4) is used. Otherwise, the equations will warp the plot of the equation in order to squeeze a potentially infinite radius arc into a finite range. If the Nyquist plot is warped on Cartesian coordinates, a constant angle or radius may instead show as a bent curve. Likewise, if the plot is warped on polar coordinates, a constant real or imaginary component may bend. Since LTI functions have phase that typically approaches a constant value asymptote it may be more reasonable to avoid using Cartesian coordinates. In addition, the domain of converted Cartesian coordinates will become a square so the representation of the infinite radius arc would instead appear as the border of a square, which may seem peculiar.

When polar coordinates are used to generate the Nyquist plot, the angle would be preserved and only the magnitude would use an equation to squeeze an infinite range into a finite radius. Since a magnitude would range between zero and infinity, only half of the output range of (5.4), (5.8), or (5.12) would be used, which is reasonable since it needs only to represent the radius of the finished plot. The circle at radius two centered at the origin would represent infinite magnitudes. The circle at radius one would represent the value chosen as $x_1$. If (5.8) or (5.12) is chosen then the parameter $x_2$ would be the values that exist on the circle radius 1.5. This can be seen in Figure 21.
Figure 22 shows a comparison between the different scaling methods. Using the piecewise-reciprocal method in (5.4), all points within the assigned value of $x_1$ will be a scalar multiple of its actual value. In the example, $x_1$ is selected as one, so the plots match each other within the unit circle. The arctangent method (5.8) would have been reasonably close to the piecewise-reciprocal method if $x_2$ were selected as two. Instead, $x_2$ was set to ten to demonstrate how it can be tuned so that a wider range of magnitudes that can be interpreted.

Once the points of the plot (with positive and negative frequencies) have been converted and plotted, the infinite radius arcs will need to be drawn. Using the information from section 5.1, if a phase discontinuity does exist, the difference between phase angles must be determined and used to plot an arc at radius two. A discontinuity may exist at zero or infinite frequency for poles or zeros at the origin whereas imaginary poles may cause one at a finite frequency.
CHAPTER VI
RECOVERING LOST GAIN IN THE ENFORCED PLANT

The controller from linear ADRC attempts to place all closed-loop poles at $-\omega_c$ (the controller bandwidth parameter) and does so under the assumption that the enforced plant is a perfect cascaded-integral. As discussed in Chapter II, this may only be true if the observer bandwidth were infinite so there will likely be some expected differences between what the controller expects and what it actually sees. When low-frequency gain is below that of the cascaded-integral, the closed-loop response will be slower than intended, thereby disconnecting the selected controller bandwidth from the actual response. If this gain can be corrected, the connection can be restored allowing the response time to be selected directly.

Figure 2 showed an example of how the low frequency magnitudes would decrease with lower observer bandwidth values. Likewise, the low frequency mimicked the slope of a single integral even though it was designed to approximate a double integral. The single integral response was expected since only a single pole would have existed at the origin regardless of the bandwidth. This can be seen in (3.2) as a single $s$ may be factored out of the bottom of the equation. Perhaps a gain that is a function of observer bandwidth may also be factored out.
6.1 First-order Gain Correction

For a first-order system where \( b_0 \) is selected as \( b \), equation (3.2) can be used to extract the low-frequency gain of the enforced plant \( \bar{G}_p \) in (6.1). It is known that the response will be single integral, therefore multiplying (3.2) by \( s \) and taking the limit as \( s \) approaches 0 should result in a constant. If the reciprocal of this value \( k_c = 1 + 2a/\omega_o \) is used as a gain correction to multiply the enforced plant, then this gain would effectively cancel leaving a low-frequency response with the same magnitudes as a single integral.

\[
\lim_{s \to 0} \left( s \bar{G}_p \right) = \frac{(s/\omega_o + 1)^2}{\frac{1}{\omega_o^2} s^2 + \left( \frac{2}{\omega_o} + \frac{a}{\omega_o^2} \right) s + \left( 1 + \frac{2a}{\omega_o} \right)} = \left( 1 + \frac{2a}{\omega_o} \right)^{-1} \tag{6.1}
\]

Figure 23: Uncorrected enforced plant  
Figure 24: Gain corrected enforced plant

For example, should a first-order plant be modeled such that \( G_p b_0^{-1} = 1/(s+7) \), Figure 23 shows the Bode diagram of the enforced plant for bandwidths of 0.1, 1, and 10. As bandwidth increases, the magnitudes below the bandwidth frequency approach a single integral. Magnitudes below the bandwidth frequency matched the single integral when each of these responses was multiplied by the gain correction \( k_c \) (Figure 24).
Above the bandwidth, the gain corrected response actually has a higher gain than a single integral. This is reasonable due to the numerator of (6.1). This numerator consists only of the zeros placed at $\omega_o$, and could probably just be cancelled by poles. Figure 25 shows the gain corrected response with a second-order low-pass filter at $\omega_o$.

![Figure 25: Gain correction with filter](image)

By using the filtered and gain correction $M_c$ in (6.2), the enforced plant has a stronger low-frequency response without unnecessarily increasing magnitudes at frequencies above the observer bandwidth.

$$M_c = \left(1 + \frac{2a}{\omega_o}\right)^2 \left(\frac{s}{\omega_o + 1}\right)^2$$  \hspace{1cm} (6.2)
6.2 Second-order Gain Correction

When a second-order plant is being modified into a double integral by the disturbance rejecter, a gain reduction is also present albeit in a slightly different fashion. Say that the plant $G_p$ contains one zero and two poles such that $G_p = \frac{s+b}{s^2+a_1s+a_0}$ and $b_0$ is selected as $b$. Substituting this information into (3.2) will result in one of four possibilities. If $b$ and $a_1$ are nonzero values, but $a_0$ is zero, then the enforced plant will have two $s$ factors in the denominator. When this is true, multiplying the enforced plant by $s^2$ and taking the limit as $s$ approaches 0 will result in the low-frequency gain of the enforced plant compared with a double integral as shown in equation (6.3). If the reciprocal $(1+3a_1b^{-1}\omega_o^{-1})$ of this is multiplied with the enforced plant, the low-frequency response will match that of a double integral within the bandwidth of the observer.

$$\lim_{s \to 0} (s^2 G_p) = \frac{s^2(s+b)(s/\omega_o+1)^3}{(s+b)s^2+(s^2+a_1s+a_0)(s/\omega_o+1)^3-1} = b\left(\frac{3a_1}{\omega_o} + b\right)^{-1} \quad (6.3)$$

If $b$, $a_1$, and $a_0$ are all nonzero values, then there will be one $s$ in the denominator of the enforced plant. Equation (6.4) shows the low-frequency gain compared with a single integral. The enforced plant is multiplied by only one $s$ before taking the limit. The low frequency of the double integral can be obtained by multiplying the enforced plant by the reciprocal $(3a_0b^{-1}\omega_o^{-1})$ and by another integral.

$$\lim_{s \to 0} (s\tilde{G}_p) = \frac{s(s+b)(s/\omega_o+1)^3}{(s+b)s^2+(s^2+a_1s+a_0)(s/\omega_o+1)^3-1} = b\left(\frac{3a_0}{\omega_o}\right)^{-1} \quad (6.4)$$
If $a_1$ and $a_0$ are nonzero but $b$ is zero, then the enforced plant will not have an integral response in the low frequency range. It would therefore need to be multiplied by a double integrator in order to get the desired response. The low frequency gain can be found by taking the limit of the enforced plant as $s$ approaches 0 as shown in (6.5). The reciprocal of this gain would need to be multiplied by the double integrator.

$$\lim_{s \to 0} \left( \tilde{G}_p \right) = \frac{(s + 0)\left(\frac{s}{\omega_o} + 1\right)^3}{(s + 0)s^2 + \left(s^2 + a_1s + a_0\right)\left(\frac{s}{\omega_o} + 1\right)^3 - 1} = \frac{\omega_o}{3a_0} \quad (6.5)$$

In the fourth possibility, $a_1$ and $a_0$ are zero but $b$ is nonzero. In this case, equation (6.3) can still be used. However, when $a_1$ is substituted into the equation, the gain reduces to 1. In this case, no gain correction would be required to match the low frequency double integral response.

Each of these techniques can be modified to cancel the roots in the numerators of the enforced plant. Equations (6.6), (6.7), and (6.8) would be used when the original plant contained a single integral, no integral, or single derivative, respectively.

$$M_c = \frac{1 + \frac{3a_1}{b\omega_o}}{\left(\frac{s}{\omega_o} + 1\right)^3} \quad (6.6)$$

$$M_c = \frac{3a_0}{b\omega_o s\left(\frac{s}{\omega_o} + 1\right)^3} \quad (6.7)$$

$$M_c = \frac{3a_0}{\omega_o s^2\left(\frac{s}{\omega_o} + 1\right)^3} \quad (6.8)$$
Figure 26 shows an example where $a_0$ is zero, $b$ and $a_1$ are nonzero, and the enforced plant is modified by equation (6.6). In figure 27, $b$, $a_1$, and $a_0$ are all nonzero and the enforced plant is modified by (6.7). Figure 28 shows the enforced plant modified by (6.8) where $a_1$ and $a_0$ are nonzero, but $b$ is zero. A plant where $a_1$ and $a_0$ are zero but $b$ is nonzero has its enforced plant unmodified in Figure 29.

**Figure 26:** Modified enforced plant: $b=3$, $a_1=7$, $a_0=0$

**Figure 27:** Modified enforced plant: $b=3$, $a_1=7$, $a_0=11$

**Figure 28:** Modified enforced plant: $b=0$, $a_1=7$, $a_0=11$

**Figure 29:** Modified enforced plant: $b=3$, $a_1=0$, $a_0=0$
6.3 Using the Inverse of the Enforced Plant

In the previous two sections, methods were proposed in order to correct for what typically appears as a gain reduction of the enforced plant compared with that of a cascaded-integral within the bandwidth of the observer. Correction may also be done by creating the inverse of the enforced plant to adjust the gain and cancel the poles and zeros provided that the model is a reasonable approximation. The cascade of the enforced plant and its inverse would be a unity gain, which may even be sufficient enough to control even without adding an additional feedback loop for a controller. However, this would require the model to always be correct and for the inverse of the enforced plant to be proper. Equation (6.9) shows the inverse of the enforced plant from (2.10). This is proper only when the inverse of the plant is proper.

\[
\bar{G}_p^{-1}(s) = s^n \left[ \frac{1}{(s/\omega_o + 1)^{n+1}} \right] + G_p^{-1}(s) b_0 \left[ \frac{(s/\omega_o + 1)^{n+1} - 1}{(s/\omega_o + 1)^{n+1}} \right]
\]  

(6.9)

The inverse can be made into a proper form that can better approximate the cascaded-integral when cascaded with the enforced plant by integrating both terms of (6.9) \(n\) times. This is provided that the function \(G_p^{-1} s^n\) is proper and should work for any plant that can be represented as an inverse. The correction function is shown in (6.10).

\[
M_s = \frac{1}{(s/\omega_o + 1)^{n+1}} + \frac{b_0}{G_p s^n} \left[ 1 - \frac{1}{(s/\omega_o + 1)^{n+1}} \right]
\]  

(6.10)
CHAPTER VII

PLANT IDENTIFICATION

The transfer function analysis of the enforced plant is possible only with a reasonable model of the plant. Although there are several methods of obtaining a model, this chapter will provide an example of obtaining model parameters for a system with a first-order plus dead time (FOPDT) response. The results need only to be accurate enough so that the actual plant response remains within the gain and phase margins of the model as the plant changes over time.
7.1 Relay Tuning Method

Model parameters for a FOPDT response can be obtained by using a relay tuning method. This is accomplished by placing the plant in a negative feedback loop with a relay as its controller. This forces the feedback loop to oscillate at the lowest frequency where the plant output phase is lagging 180° from the relay [54]. The period of oscillation can be easily obtained by timing the pulse wave seen at the output of the relay. Equation (7.1) shows the phase relationship between the oscillation frequency ($\omega_1$), dead time ($T_d$), and a time-constant ($\tau$). With only $\omega_1$ known, a curve exists with all possible combinations of $T_d$ and $\tau$ that satisfies the equation. Since there isn’t enough information, additional points would be needed in order to identify the other model parameters.

$$T_d = \frac{\pi - \tan^{-1}(\omega_1 \tau)}{\omega_1}$$ (7.1)

By adding a known amount of delay ($d$) to the feedback loop, an additional test can be run yielding a second oscillation frequency ($\omega_2$), which should be lower than the first. Equation (7.2) provides a second curve of possible parameter values. Combining the equations by substituting $T_d$ provides (7.3) which leaves only to solve for a $\tau$ that makes the equation true. Rather than solving for 0, it can be solved for the difference between the dead time variables in the other two equations ($\Delta T_d$), which should be as close to 0 as possible.

$$T_d + d = \frac{\pi - \tan^{-1}(\omega_2 \tau)}{\omega_2}$$ (7.2)

$$\Delta T_d = \frac{\pi - \tan^{-1}(\omega_1 \tau)}{\omega_1} - \left(\frac{\pi - \tan^{-1}(\omega_2 \tau)}{\omega_2} - d\right) = 0$$ (7.3)
7.2 Solving for the Unknown Parameters

Equation (7.3) appears to be too complex to solve arithmetically. To find the solution, a bounded box of realistic solutions must be established so the answer may be found within. An iterative process can make this box smaller in each step until the solution is close enough to zero to make the equation approximately true.

The total phase delay at the oscillation frequency must be \( \pi \). The amount of phase lag from the first-order time constant at the oscillation frequencies must be between 0 and \( \pi/2 \). Therefore, the amount of phase lag from the delay must be between \( \pi/2 \) and \( \pi \). Likewise, since the derivative of the arctangent function is always positive, and the function is always increasing, the amount of phase lag at \( \omega_1 \) must be larger than \( \omega_2 \). At the smallest possible time-constant values, where its phase contribution is nearly zero, the contribution from dead time must be nearly \( \pi \). When the time-constant contribution is nearly \( \pi/2 \), the dead time contribution would also be nearly \( \pi/2 \). As a result, the value for dead time should be bounded within \( \pi/\omega_1 \) and \( \pi/(2 \omega_2) \).

This boundary assumes that a zero or infinite time-constant is possible. A boundary for \( \tau \) should be found to make a more reasonable boundary for \( T_d \). By solving for \( \tau \) in (7.2) and substituting the previously mentioned limits for \( T_d \), the boundary \( \tau \) for can be obtained as in (7.4), where \( \tau_{\text{min}} \leq \tau \leq \tau_{\text{max}} \).

\[
\begin{align*}
\tau_{\text{min}} &= -\tan \left( \frac{\pi}{\omega_1} + \omega_2 d \right) - \frac{\omega_2}{\omega_1} \\
\tau_{\text{max}} &= -\tan \left( \frac{\pi}{2 \omega_1} + \omega_2 d \right) - \frac{\omega_2}{\omega_1}
\end{align*}
\]
When these boundary limits are substituted into (7.3), a range of $\Delta T_d$ can be established as shown in (7.5).

$$\begin{align}
\Delta T_{d,\text{min}} &= \frac{\pi - \tan^{-1}(\omega_1 \tau_{\text{min}})}{\omega_1} - \left(\frac{\pi - \tan^{-1}(\omega_2 \tau_{\text{min}})}{\omega_2} - d\right) \\
\Delta T_{d,\text{max}} &= \frac{\pi - \tan^{-1}(\omega_1 \tau_{\text{max}})}{\omega_1} - \left(\frac{\pi - \tan^{-1}(\omega_2 \tau_{\text{max}})}{\omega_2} - d\right)
\end{align}$$

(7.5)

Consider a two-axis plot with a vertical axis of $\Delta T_d$ and $\tau$ as the horizontal. The solution exists inside the rectangle bounded by $\Delta T_{d,\text{min}} \leq \Delta T_d \leq \Delta T_{d,\text{max}}$ and $\tau_{\text{min}} \leq \tau \leq \tau_{\text{max}}$. During each iteration, a diagonal line is drawn between opposite corners to find the value of $\tau$ that allows $\Delta T_d$ to be 0. Equation (7.3) is then evaluated for this value of $\tau$. If the sign of the solution matches that of $\Delta T_{d,\text{min}}$ then $\tau_{\text{min}}$ is changed to $\tau$ and a new value for $\Delta T_{d,\text{min}}$ is obtained from (7.5). Otherwise, $\tau_{\text{max}}$ is changed to it $\tau$ and $\Delta T_{d,\text{min}}$ is updated. This process repeats until the boundary is small enough where a reasonable $\tau$ can be obtained so that it can be substituted into (7.1) to solve for $T_d$.

With solutions for $\tau$ and $T_d$, a gain ($k$) would need to be estimated to build a FOPDT model. Assuming a starting gain of 1, the output of the model can be compared with that of the actual plant to determine the value of $k$ that makes the two nearly equal. Since a cycling response is periodic, the root-mean-square (RMS) values of both can be generated and compared. The $k$ value would be the value that the unity-gain model RMS must be multiplied by to match that of the plant RMS. Since the plant has already been put through oscillations, the RMS value could have been measured when the dead time and time-constant tests were being conducted. By storing the time and amplitude information of the relay output, the same information can later be passed into the unity-gain model in order to measure its RMS so that the two measurements may be compared.
The tools provided in this thesis offer the ability to find tunings for the parameterized ESO that will place the poles of the enforced plant within a predetermined angle of the negative real axis while providing the plant with desired gain and phase margins. This chapter will give an example of using these tools to tune a FOPDT plant.
8.1 Finding Model Parameters

To verify the relay tuning method for plant identification from Chapter VIII, a program was developed and tested. Equation (8.1) shows a FOPDT plant model $G_p$ of a steam boiler firing system [64].

$$G_p = \frac{0.003e^{-60s}}{145s+1}$$  \hspace{1cm} (8.1)

The amount of additional delay ($d$) was varied between 1 and 5 seconds. Each test provided different estimations of $k$, $T_d$, and $\tau$. Estimated models were created with their responses compared with the original plant model from (8.1). Although the low-frequency gain varies between these models (Figure 30), their accuracy improves considerably near the negative real axis. Considering that the stability analysis will be done on the first derivative of the plant model (4.8), the Nyquist responses of the model derivative were compared (Figure 31) showing responses that nearly overlap each other.

Figure 30: Nyquist response of various model identifications

Figure 31: Nyquist response of model derivatives
8.2 Root Locus Analysis

To have an idea as to where the enforced plant poles will be located, the root locus method can be used. However, the delay would need to be approximated by actual poles. Equation (8.2) shows an approximate model to (8.1) that does not contain a delay.

\[
G_p = \frac{0.003 e^{-60s}}{154s + 1} = \frac{0.003}{145s + 1} \left( \frac{1}{s / 0.07 + 1} \right)^4
\]

This approximation should do a decent job at mimicking the low frequency response of the plant, but may have a difficult time with the higher frequencies. A step response is shown in Figure 32 to demonstrate the approximation.

![Figure 32: Approximation of time delay response](image)

Figure 32: Approximation of time delay response

![Figure 33: Pole paths of approximate model](image)

Figure 33: Pole paths of approximate model

Root locus plots were generated (Figure 33) using several different values for \( b_0 \). In all cases, two poles approach the imaginary axis rather quickly which would lead to increasing oscillation as observer bandwidth increases. This would indicate that only a low bandwidth would be obtainable.
Further analysis will be done using the original FOPDT model rather than the approximation, as the approximation was only required in order to generate continuous-time roots. This could actually have been done by searching for discrete-time roots instead as long as the delay was an integer multiple of a sample time. Continuous-time seems to be more intuitive for most readers so it is used instead.

### 8.3 Finding Observer Parameter Values

A program was written for MATLAB to automatically find the minimum values of $b_0$ that satisfy stability margins. The program needs to be provided with a transfer function system that may include a time delay. When it is given the order of the approximate cascaded-integral and the requirements of gain margin, phase margin, and damping angle, it provides the minimum $b_0$ value needed for a series of observer bandwidths. The program uses the Nyquist contour modification in order to satisfy the damping angle to satisfy the damping angle requirement. It first extracts the polynomial coefficients of the transfer function without the delay and multiplies these by values as shown in (4.10) and (4.11) so that when frequency results are processed from imaginary axis inputs, the results will be as if they were generated from rays that are the damping angle from the negative real axis. A new transfer function is assembled with the modified information and frequency results are processed.

A reasonable list of observer bandwidths is determined based on where the original plant roots are located. A finite scale is created using the logarithmic equation in
(5.12). This scale is then broken into an arbitrary number of equidistant steps (35 was used for the later results) where no value could represent zero or infinity. The logarithmic center of the plant’s roots was chosen as the midpoint on the scale whereas the maximum root value was set to be at 75% of the scale. The inverse of (5.12) was used to convert these values into actual bandwidth frequencies.

At each of these observer bandwidth frequencies, a function is created using (4.8) in order to find the minimum values of $b_0$ that satisfy the stability margins. Each time this function crosses the angle dictated by the phase margin requirement, the magnitudes are compared to determine which one is largest (same as the example in Figure 9). As the function crosses the positive real axis, magnitudes are compared to get the maximum, which is then multiplied by the gain margin requirement. The largest of these conditions is stored as the minimum value of $b_0$ for the observer bandwidth being evaluated.

As shown in (4.7) and explained in section 4.1, the low-frequency gain has a factor of $\omega_o b_0^{-1}(n+1)$. As $\omega_o$ increases, $b_0$ would need to increase by the same amount so as not to dramatically change the low-frequency gain. Since $\omega_o$ also shapes the response, the relationship is not necessarily constant across all frequencies, though it should be expected that this relationship might approach a diagonal asymptote.
8.4 Analyzing the Results

The program was run to test how the relationship between $b_0$ and $\omega_0$ changes as a gain margin requirement varies between 1 and 100. Since the high frequency gain of the plant model is slightly larger than $2 \times 10^{-5}$, a plot was generated showing where higher bandwidths cross this value. Figure 34 shows several curves that satisfy several gain margin requirements while satisfying a phase margin of 45 degrees. These curves appear to approach a diagonal asymptote but are not linear at lower frequencies. If higher bandwidth is required, higher values of $b_0$ must be selected to satisfy the stability margins. When a larger gain margin must be provided, $b_0$ value selection must be considerably higher at a given bandwidth.

![Figure 34: Minimum $b_0$ for various GM](image)

Results were then obtained by varying the phase margin requirement between 0 and 60 degrees. Tests were done while keeping gain margin at 1. Figure 35 shows that similar to gain margin, a larger value of $b_0$ is needed in order to provide a larger phase margin. Another test was performed by varying the damping angle requirement between 90 and 50 degrees. When the angle is at 90 degrees, the results are simply checking for...
the critical stability point where poles would exist in the right-half plane if the minimum $b_0$ were not selected. Figure 36 shows that higher values of $b_0$ for a given observer bandwidth will satisfy a lower damping angle requirement.

![Figure 35: Minimum $b_0$ for various PM](image1)

![Figure 36: Minimum $b_0$ for various damping angles](image2)

### 8.5 Improving Observer Bandwidth

If the plant is altered in such a way as to make (4.15) true, then there should be no limit to the observer bandwidth once the $b_0$ values is large enough. To make this happen, the $n^{th}$ derivative of the plant must be contained within a circle of radius 1, centered at (+1, 0). Figure 37 shows a response that is radius 1 but instead encircles the origin caused by the 60-second plant delay taking all Nyquist points and spinning them clockwise around the origin by a phase equal to 60 times the frequency. There are no values of $b_0$ that would satisfy (4.15) for the delayed plant. However, the non-delayed plant can easily be made to satisfy this condition.
Since it isn’t possible to have a perfect 60-second prediction to cancel the time delay, whatever modification is done cannot contain a plant without the delay. Likewise, the modified response must begin from the origin otherwise it wouldn’t be a derivative function anymore. Let \( G_{p, s} b_{0}^{-1} = MD \) where \( M = 0.003 s b_{0}^{-1} / (145 s + 1) \) and \( D = e^{-60 s} \). Say that the plot of \( M \) is shifted left by one, and then multiplied by \( D \) causing the high frequencies near the origin of \( M-1 \) to spiral around it. If this plot is then pushed to the right by one, it will again begin at the origin, be contained in the right half plane, and can satisfy (4.15). These plot modifications can be seen in Figure 38.

The modification of \( D(M-1)+1 \) would then need to replace the function \( G_{p, s} b_{0}^{-1} \) with \( G_{p, s} b_{0}^{-1} - D + 1 \) which would require additional paths to be drawn in the block diagram of the disturbance rejecter. By splitting (2.7) such that one part is \( s^{m} \) and the other is the negative of (2.6), the additional signal paths can be added or subtracted to the signal node between them. Both paths must originate from the node \( b_{0} u \), but one of them...
will be delayed by 60 seconds to match the amount of delay in the FOPDT model. Figure 39 shows how these signal paths are connected.

This diagram can be simplified. Since the new unity gain signal paths is flowing into a transfer function that is the negative of (2.6) and there is already a unity gain path entering the function of (2.6), the net effect cancels. Likewise, the new delayed signal can instead flow directly into the (2.6) function allowing for (2.7) to be recombined resulting in something similar to Figure 1 except that the $b_0u$ signal into the observer is delayed. The result is actually identical to another proposed solution [65] showing that it does indeed work with reasoning that can be understood because of equation (4.15).

In order to find the new $b_0$ values for various bandwidths, the program must be modified to support the delayed control signal into the observer. MATLAB does not allow for addition terms to contain systems with different delay amounts, so equation (4.8) cannot be constructed in the program and an alternative means must be done in order to find the magnitudes when phase crosses over. The equation can be broken into
three parts, evaluated as complex values at a reasonable number of predetermined frequencies, then have the solutions combined back together as if (4.8) were just evaluated directly. This makes it far more difficult to find where the phase crosses the positive real axis of the phase margin angle. The program’s current state does generate some false positives for phase crossing, resulting in some $b_0$ value calculations to be higher than they should be. Regardless, this still provides some intuition as to where the relationship between bandwidth and $b_0$ should be.

Figure 40 shows how the minimum $b_0$ to $\omega_o$ relationship changes as delay is added to the control signal input to the observer. As the additional delay increases, the minimum $b_0$ value is reduced for each given observer bandwidth. A predetermined value for $b_0$ may have a significantly larger maximum bandwidth while satisfying the stability margins. These plots curve to the right possibly towards a horizontal or diagonal asymptote.

### 8.6 Simulation Results

In order to validate the information presented in this chapter, applying the results in simulation should demonstrate a predetermined amount of damping while satisfying stability margins. In Figure 40, a minimum $b_0$ value of approximately $1.24 \times 10^{-4}$ is necessary for a bandwidth of 0.02 rad/s as long as a control delay of 60 seconds is used. This supports a gain margin of 10 times, a phase margin of 45 degrees, and ensures that the enforced plant poles are within 45 degrees of the negative real axis.
Since the enforced plant should be integrating, an input step will yield an output that is constantly increasing making it more difficult to see any oscillation. Since this is in simulation, the derivative of the output can be taken without ill effects from noise. In an ideal case where the enforced plant is a perfect integral, the derivative of the output should appear as a non-delayed step with unity gain.

However, the output results in a delayed step with an initial impulse spike at 60 seconds followed by a falling transient that settles at a steady value. As shown in Figure 41, when plant gain was increased greater than 10 times the original value, the simulation showed noticeable oscillation, albeit still damped. At exactly 10 times the gain, the result shows the transient falling from the initial impulse just slightly past the settling value then returning to it with an overshoot that is reminiscent of poles with 45 degrees of damping.

Additional plant delay was added while keeping the control signal delay at 60 seconds in order to test phase margin. Results are shown in Figure 42. Since the \( b_0 \) selection was taken to also support a gain margin of 10, increasing the plant delay from
60 to 660 seconds does slow down the amount of time that the derivative of the output will settle. Values above 160 seconds do show a damped oscillation that would appear to have less damping than 45 degrees.

When the inverse of the enforced plant from (6.9) is constructed and placed inline with the enforced plant, the derivative of the response is closer to that of a delayed step. The delay in the plant cannot be inverted, and neglecting it prevents a perfect inversion from being possible. Nonetheless, the low frequency gain appears to be six to seven times as strong compared to the enforced plant without the correction (Figure 43). The correction does not appear to cause issue should the plant gain fluctuate up to the selected gain margin value, although the total corrected gain does exceed that of the delayed step (Figure 44).

Figure 43: Applying inverse correction of enforced plant
Figure 44: Inverse correction with higher plant gains

Much of the difference between the corrected and uncorrected responses could be obtained by a proportional gain and therefore could be obtained by using a simple proportional controller. The parameterized controller for ADRC will attempt to place all poles at \(-\omega_c\), dictated by a controller bandwidth. With a proportional controller and a
single integral for the enforced plant, a single pole at $-\omega_c$ should be fairly close to steady state between 4 and 5 time constants. When the enforced plant is not corrected and has magnitudes significantly less than that of the integral, the controller will not be able to make the plant settle in time and would therefore require a higher controller gain to do so.

The proportional control was added to the simulation in order to validate this assumption. The feedback needed to be taken from the plant output measurement rather than the estimated output state due to the low observer bandwidth selected. When the inverse enforced plant was used inline with the enforced plant, a controller bandwidth selection of 0.0125 could cause the plant to settle in about 400 seconds which is reasonable since 4 to 5 time constants plus 60 seconds gives a window between 380 and 460 seconds. Without the inverse correction, the controller bandwidth needed to be 0.1 to settle in about the same amount of time. If this bandwidth were selected based upon a performance requirement, it would be far shy of the 100-110 seconds.

Figure 45 shows the controlled responses with and without using the inverse correction compared with the first-order response that is expected by a controller bandwidth selection of 0.0125.

![Figure 45: Effect of inverse enforced plant correction on controlled response](image_url)
CHAPTER IX

CONCLUSIONS AND FUTURE RESEARCH

The work presented in this master’s thesis demonstrates that if a plant model is obtainable, parameters for the observer can be obtained providing predetermined stability margins while allowing the disturbance rejecter to create an enforced plant that approximates a cascaded-integral with poles that are contained within a desired damping angle. The disturbance rejecter is studied in frequency domain using a general form that works for any order of the state observer. A root locus technique has been presented for understanding of how the enforced plant poles change as observer bandwidth increases, allowing the user to determine whether reasonable enforced plant poles are obtainable by tuning the observer parameters. A procedure is given for a graphical means of obtaining the gains necessary to provide the plant with stability margins. A modification to the traditional Nyquist plot has been proposed that determines whether closed-loop poles will exist within a given damping angle by changing the Nyquist contour. A method is provided to make an infinite radius Nyquist plot visible on a computer display by altering the magnitude scale. A procedure is given to solve for the observer tuning parameters given a plant model, approximate integral-order, stability margins, and enforced plant
damping angle requirement, simplifying the tuning process by showing the relationship between $b_0$ and $\omega_o$.

The techniques presented here should offer new avenues for future research. New strategies may be developed to improve observer bandwidth by changing the plant response statically or actively in order to make it easier for the disturbance rejecter to change its shape into the approximate cascaded-integral. The shape of the frequency-domain plots may be able to answer questions as to the best selection of $b_0$ and integral order for more complex model forms. With the ability to show the relationship between $b_0$ and the observer bandwidth, perhaps there is a means to connect this relationship with the controller bandwidth in such a way as to provide optimal performance. Research could be done to study design techniques that ensure stability margins and damping for the entire controlled system rather than the enforced plant alone. Software can to be developed in order to combine plant identification and automatic parameter tuning to finally provide the one-button procedure that is desired for implementation. Studies may be performed to see how these techniques can be applied to various nonlinear plant model forms. More work can be performed to determine whether it is best to have $b_0$ as a fixed constant based on plant parameters or to link to $\omega_o$ and $\omega_c$ by a proportional constant and have $b_0$ automatically selected based upon bandwidth.
REFERENCES


A. MATLAB Function for ADRC Root Locus

```matlab
function [] = rladrc( Gp, b0, n, wo )
% Plot of ADRC root path from varying observer bandwidth
% % Author: Jason Tatsumi
% % Master’s Thesis at Cleveland State University
% % 10/25/2013
% % Gp must be SISO system
% % b0 is value chosen for gain in disturbance rejecter
% % n is order of cascaded-integral to approximate
% % wo (optional) evaluates roots only at a specific bandwidth
% %
% $\text{[np, dp]} = \text{tfdata}(Gp/b0);$;
% Np = tf(np, 1);
% Dp = tf(dp, 1);
% s = tf(’s’);
% hold on;
% step = 0.001;  % Lower values take longer to process
% maxradius = 10; % Program loops until one pole exceeds this distance
% outofbounds = 0;
% unstable = 0;
% index = 0;
% if exist(’wo’)
% w = wo;
% else
% w = eps;
% end
% Q = Np*s^n + Dp*((s/w+1)^(n+1)-1);
% [poles, -, -] = zpkdata(Q);
% pi = imag(poles{1});
% pr = real(poles{1});
% scatter(pr, pi, 200, [0, 0, 1], ’X’);
% if ~exist(’wo’)
% while (outofbounds == 0)
% exp = (index - 0.5) * 100;
% w = 2^exp;
% Q = Np*s^n + Dp*((s/w+1)^(n+1)-1);
% [poles, -, -] = zpkdata(Q);
% pi = imag(poles{1});
% pr = real(poles{1});
% scatter(pr, pi, 40, [0, 0, 1], ’*’);
% if ((max(abs(poles{1})) > maxradius) || step >= 1)
% outofbounds = 1;
% end
% if ((max(real(poles{1})) > 0) && (unstable == 0))
% unstable = 1;
% end
% index = index + step;
% end
```
B. MATLAB Function for Damping Angle Nyquist Contour

```matlab
function [ ] = nyqdamping( Gp, theta )
% Nyquist Plot for a maximum damping requirement
if (~exist('theta'))
    theta=pi/2;
end
M = exp(1i*(pi/2-theta));
[n, d] = tfdata(Gp);
Num = n{1} .* M.^(length(n{1})-1:-1:0);
Den = d{1} .* M.^(length(d{1})-1:-1:0);
G = tf(Num, Den);
nyquist(G);
end
```

C. MATLAB Functions for Infinite Radius Nyquist Plot

```matlab
function [y] = scaleinf(x, type, center, scale)
% SCALEINF Maps infinite scale to finite one between -2 and 2
% %
% % Author: Jason Tatsumi
% % Master’s Thesis at Cleveland State University
% % 10/25/2013
% %
% % type 0: reciprocal
% % 1: arctangent
% % 2: logarithmic arctangent
% % center is the input point that maps to 1
% % scale is the value that maps to 1.5
% % 0 maps to 0
% % Infinity maps to 2
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (type == 0)    % Piecewise Reciprocal Scale
    x1 = abs(x);
    x1(~isfinite(x1)) = 0;  % Prevent NaN caused by 0*Inf
    x2 = 1./abs(x);
    x2(~isfinite(x2)) = 0;  % Prevent NaN caused by 0*Inf
    y = sign(x).*((2-abs(center).*x2).*((abs(x) > abs(center)) +
    (x1./abs(center))).*(abs(x) <= abs(center)));
end
if (type == 1)    % Arctangent Scale
    alpha = abs(center);
    beta = log(1+sqrt(2)) / log(scale/center);
    y = sign(x).*((2-abs(center)).*x2).*((abs(x) > abs(center)) +
    (x1./abs(center))).*(abs(x) <= abs(center)))
    y = sign(x).*((2-abs(center)).*x2).*((abs(x) > abs(center)) +
    (x1./abs(center))).*(abs(x) <= abs(center)))
end
end
```
if (type == 2) % Logarithmic Scale
    alpha = log10(abs(center));
    beta = log10(abs(scale)) - alpha;
    y = sign(x).*(1 + (2/pi).*atan((log10(abs(x)) - alpha) ./ beta));
end

function [ ] = infnyquist3( G )
% Infinite Radius Nyquist plot
% % Author: Jason Tatsumi
% % Master's Thesis at Cleveland State University
% % 10/25/2013
% % G must be continuous-time transfer function

% Use frequencies of root locations to determine reasonable scaling parameters. Arrows will be located based on same frequencies.
[z, p, ~] = zpkdata(G); % Find roots
freqs = [abs(z{1})' abs(p{1})'];
freqs = unique(freqs(freqs~=0)); % Remove zeros and duplicates

% Discover parameters for infinite magnitude scale
[mag, phase, ~] = bode(G, freqs);
mag = mag(:)';
phase = phase(:)';
maxmag = max(mag);
minmag = min(mag);
magcenter = 10^(0.5*(log10(minmag) + log10(maxmag))); % logarithmic center of magnitudes

if (x2 <= 1.1*x1) % if x2 isn't large enough, take defaults
    if (type == 1)
        x2 = x1*(1+sqrt(2)); % no scaling for arctangent
    elseif (type == 2)
        x2 = 10*x1; % default
    end
end

% Discover parameters for infinite frequency scale
maxfreq = 10*max(freqs);
minfreq = 0.1*min(freqs);
freqcenter = 10^(0.5*(log10(minfreq) + log10(maxfreq))); % logarithmic center of roots
freqx1 = freqcenter;
freqx2 = maxfreq;
if (freqx2 <= 1.1*freqx1) % if x2 isn't large enough, take defaults
    if (freqx1 == 1)
        freqx2 = freqx1*(1+sqrt(2)); % no scaling for arctangent
    elseif (freqx1 == 2)
        freqx2 = 10*freqx1; % default
    end
end

% Get Nyquist plot information and scale to finite coordinates
[rho, theta, W] = bode(G); % Store magnitude and phase
rho = rho(:,(1:length(rho)));
theta = theta(:,(1:length(theta)));

end
\[ \rho_0, \theta_0, -] = \text{bode}(G, \text{eps}); \\
\rho_{\text{Inf}}, \theta_{\text{Inf}}, -] = \text{bode}(G, \text{Inf}); \\
\rho = [\rho_0, \rho, \rho_{\text{Inf}}]; \\
\theta = [\theta_0, \theta, \theta_{\text{Inf}}]; \\
W = [0; \text{W}; \text{Inf}]; \\
\rho = \text{scaleinf}(\rho, \text{type}, x_1, x_2); \quad \% \text{Convert into finite range} \\
\theta = \theta \cdot \pi / 180; \\
W = W'; \\
W_1 = \text{scaleinf}(W, \text{freqtype}, f_{\text{freq}1}, f_{\text{freq}2}); \\
\text{starttheta} = \theta(1); \\
\text{endtheta} = \theta(\text{length}(\theta)); \\
\text{theta}_2 = (\text{endtheta} : -(\pi/180) \cdot \text{sign(\text{endtheta})} : -\text{endtheta}); \\
\text{theta}_4 = (-\text{starttheta} : (\pi/180) \cdot \text{sign(\text{starttheta})} : \text{starttheta}); \\
W_2 = (2 : -4 \cdot (\text{length}(\text{theta}_2) - 1) : -2); \\
W_4 = \text{zeros}(1, \text{length}(\text{theta}_4)); \\
X = \rho \cdot \cos(\theta); \\
Y = \rho \cdot \sin(\theta); \\
X_2 = \rho(\text{length}(\rho)) \cdot \cos(\text{theta}_2); \\
Y_2 = \rho(\text{length}(\rho)) \cdot \sin(\text{theta}_2); \\
X_4 = \rho(1) \cdot \cos(\text{theta}_4); \\
Y_4 = \rho(1) \cdot \sin(\text{theta}_4); \\
U_t = (\pi/180) \cdot [0 : 6 : 360]; \\
U_x = \cos(U_t); \\
U_y = \sin(U_t); \\
U_w = 1 \cdot \text{ones}(1, \text{length}(U_x)); \\
\% \text{Initialize plot} \\
\text{figure}(1); \\
\text{clf}(1); \\
\text{set}(1, \text{'Color'}, [1, 1, 1]); \\
\text{hold on}; \\
\% \text{Draw Nyquist plot} \\
\text{plot3}(X, Y, W_1, \text{'b'}, X_2, Y_2, W_2, \text{'b'}, X, -Y, -W_1, \text{'b'}, X_4, Y_4, W_4, \text{'b'}, \text{'LineWidth'}, 2); \\
\% \text{Create arrows for plot} \\
\text{for index} = (1 : \text{length(freqs)}) \\
\quad \text{mag}_2 = \text{scaleinf}(\text{mag(index)}, \text{type}, x_1, x_2); \\
\quad \text{freq}_2 = \text{scaleinf}(\text{freqs(index)}, \text{freqtype}, f_{\text{freq}1}, f_{\text{freq}2}); \\
\quad X = \text{mag}_2 \cdot \cos(\text{phase(index)} \cdot \pi/180); \\
\quad Y = \text{mag}_2 \cdot \sin(\text{phase(index)} \cdot \pi/180); \\
\quad z = \text{freq}_2; \\
\quad w_{\text{index}} = \text{length}(W(W < \text{freqs(index)})); \\
\quad \text{direction} = \text{atan2}(Y(w_{\text{index}}+1) - Y(w_{\text{index}}), X(w_{\text{index}}+1) - X(w_{\text{index}})); \\
\quad \text{trisize} = 0.1; \\
\quad \text{trix} = [\text{trisize} \cdot \cos(\text{direction}) \text{trisize} \cdot \cos(\text{direction} + 2\pi/3)] + X; \\
\quad \text{triy} = [\text{trisize} \cdot \sin(\text{direction}) \text{trisize} \cdot \sin(\text{direction} + 2\pi/3)] + Y; \\
\quad \text{triz} = [0 0 0] + z; \\
\quad \text{fill3}(\text{trix}, \text{triy}, \text{triz}, \text{'b'});
trix = [trisize*cos(-direction-pi) trisize*cos(-direction+pi/3) trisize*cos(-direction+pi/3)] + x;
triy = [trisize*sin(-direction-pi) trisize*sin(-direction+pi/3) trisize*sin(-direction+pi/3)] - y;
triz = [0 0 0] - z;
fill3(trix, triy, triz, 'b');
end

% Is part 2 an infinite radius arc?
if (rho(length(rho)) == 2)
  for infarcangle = (endtheta : -(pi/2)*sign(endtheta) : endtheta)
    if (abs(infarncangle) == abs(endtheta))
      x = 2*cos(infarncangle);
y = 2*sin(infarncangle);
direction = infarcangle + (pi/4)*sign(endtheta);
trisize = 0.1;
trix = [trisize*cos(direction) trisize*cos(direction+2*pi/3) trisize*cos(direction-2*pi/3)] + x;
triy = [trisize*sin(direction) trisize*sin(direction+2*pi/3) trisize*sin(direction-2*pi/3)] + y;
triz = [0 0 0];
fill3(trix, triy, triz, 'b');
  end
end
end

% Is part 4 an infinite radius arc?
if (rho(1) == 2)
  for infarcangle = (-starttheta : (pi/2)*sign(starttheta) : starttheta)
    if (abs(infarncangle) == abs(starttheta))
      x = 2*cos(infarncangle);
y = 2*sin(infarncangle);
direction = infarcangle - (pi/4)*sign(starttheta);
trisize = 0.1;
trix = [trisize*cos(direction) trisize*cos(direction+2*pi/3) trisize*cos(direction-2*pi/3)] + x;
triy = [trisize*sin(direction) trisize*sin(direction+2*pi/3) trisize*sin(direction-2*pi/3)] + y;
triz = [0 0 0];
fill3(trix, triy, triz, 'b');
  end
end
end

% Draw red unit circle and -1,0 point
radius = scaleinf(1, type, x1, x2);
plot3(radius*Ux, radius*Uy, Uw, 'r', radius*Ux, radius*Uy, -Uw, 'r');
scatter3(-radius, 0, 200, '+', 'r');
lowdb = 10*floor(2*log10(x1^2/x2));
highdb = 10*ceil(2*log10(x2));
dbscale = (highdb-lowdb)/4;
dbrange = (lowdb - dbscale : dbscale : highdb + dbscale);
for db = [dbrange Inf]
  radius = scaleinf(10^(db/20), type, x1, x2);
circlecolor = [0.6 0.6 0.6];
  if (db ~= 0)
    plot3(radius*Ux, radius*Uy, Uw, 'k:', radius*Ux, radius*Uy, -Uw, 'k:', 'Color', circlecolor)
    text(0.7071*radius, 0.7071*radius, 2, [num2str(db) ' dB'], 'HorizontalAlignment', 'center')
  end
end

% Angle marks
for angle = [0 30 60 90 120 150]
c = 2*cos(angle*pi/180);
s = 2*sin(angle*pi/180);
plot3([c -c -c c],[-s -s s s],[-2 -2 2 -2], ':', 'Color', [0.6 0.6 0.6])
text(1.1*c, 1.1*s, 2, [num2str(angle) '^o'], 'HorizontalAlignment', 'center')
text(-1.1*c, -1.1*s, 2, [num2str(angle+180) '^o'], 'HorizontalAlignment', 'center')
end

% Finish plot
axis([-2, 2, -2, 2])
axis square;
axis off;
hold off;
end

D. MATLAB Function for Minimum $b_0$ Calculation

function [ bmin, frequency ] = b0( g, n, gm, pm, theta )
% ADRC Observer Bandwidth Limit

% % Author: Jason Tatsumi
% % Master’s Thesis at Cleveland State University
% % 10/25/2013
% % g must be SISO system
% % n is order of cascaded-integral to approximate
% % gm is required gain margin (not in dB!)
% % pm is required phase margin (radians)
% % theta is damping ratio angle of enforced plant poles (radians)
% % 0 < theta ? pi/2
% % controldelay is number of seconds that control signal is delayed
% %
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (~exist('n'))
n=1;
end
if (~exist('gm'))
gm=10;
end
if (~exist('pm'))
  pm=pi/4;
end
if (~exist('theta'))
  theta=pi/2;
end
s = tf('s');
M = exp(li*(pi/2-theta));
delay=g.InputDelay;
g = minreal(g);  % Cancel duplicate pole/zero pairs
[num, den] = tfdata(g);
NUM = num{1} .* M.^(length(num{1})-1:-1:0);
DEN = den{1} .* M.^(length(den{1})-1:-1:0);
G = tf(NUM, DEN);  % Modified transfer function
delay = delay * (pi - theta);  % Modified delay for damping requirement
G.InputDelay = delay;
[z, p, -] = zpkdata(g);  % Find roots
freqs = [abs(z{1})' abs(p{1})'];
freqs = unique(freqs(freqs==0));  % Remove zeros and duplicates
maxfreq = 10*max(freqs);
minfreq = 0.1*min(freqs);
freqcenter = 10^(0.5*(log10(minfreq) + log10(maxfreq)));  % logarithmic center of roots

% finite frequency will scale from 0 to 2 using arclog scale
alpha = log10(abs(freqcenter));
beta = log10(abs(maxfreq)) - alpha;
y = sign(x).*((1 + (2/pi)*atan((log10(abs(x)) - alpha) ./ beta));
steps = 35;
frequency = zeros(steps, 1);
bmin = zeros(steps, 1);
for step = (1 : steps)
scaledwo = step*2/(steps+1);
% convert frequency from finite scale to infinite scale
wo = sign(scaledwo).*10.*((alpha+beta).*tan((pi/2).*((stopphase - pm) && (gm*stopmag < b0gm)))
N = -G*s^n/((s/wo+1)^n+1);
[maq, phase, w] = bode(N);  % get Nyquist information
mag = mag(:)';
phase = phase(:)';
% phase2 is to be truncated between +/-pi
phase2 = mod(phase, 360);
if (startphase == pm) && (startmag > b0pm)
b0pm = startmag;
elseif (stopphase == pm) && (stopmag > b0pm)
b0pm = stopmag;
elseif (startphase == 0) && (gm*startmag > b0gm)
b0gm = gm*startmag;
elseif (stopphase == 0) && (gm*stopmag > b0gm)
b0gm = gm*stopmag;
else
% Ensure that this does not cross over pi!!!!!
if (sign(stopphase - startphase) == sign(phase(index+1) - phase(index)))  % phase changes in same direction
if ((startphase > pm) && (stopphase < pm)) || ((startphase > 0) && (stopphase < 0))
    startphase = phase2(index+1);
    stopphase = phase2(index);
    startmag = mag(index);
    stopmag = mag(index+1);
else
    % interpolate
    if ((startphase < pm) && (stopphase > pm)) || ((startphase < 0) && (stopphase > 0))
        slope = (stopmag - startmag) / (stopphase - startphase);
        intercept = startmag - slope*startphase;
        if straddled_pm, and larger than b0pm, then set
        if (startphase < pm) && (stopphase > pm)
            interpolated_mag = slope*pm+intercept;
            if (interpolated_mag > b0pm)
                b0pm = interpolated_mag;
        end
    end
end
end
end
% if straddled 0, and larger than \( b_0 \), then set
if (startphase < 0) && (stopphase > 0)
    interpolated_mag = intercept;
    if (gm*interpolated_mag > b0gm)
        b0gm = gm*interpolated_mag;
end
end

frequency(step) = wo;
bmin(step) = b0gm;
bmin(step) = max(b0pm, b0gm);
end

function [ bmin, frequency ] = b0( g, n, gm, pm, theta, controldelay )
% ADRC Observer Bandwidth Limit
% % Author: Jason Tatsumi
% % Master's Thesis at Cleveland State University
% % 10/25/2013
% % g must be SISO system
% % n is order of cascaded-integral to approximate
% % gm is required gain margin (not in dB!)
% % pm is required phase margin (radians)
% % theta is damping ratio angle of enforced plant poles (radians)
% % 0 < theta < \( \pi/2 \)
% % controldelay is number of seconds that control signal is delayed
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (~exist('n'))
    n=1;
end
if (~exist('gm'))
    gm=10;
end
if (~exist('pm'))
    pm=pi/4;
end
if (~exist('theta'))
    theta=pi/2;
end
if (~exist('controldelay'))
    controldelay=0;
end
s = tf('s');
M = exp(1i*(pi/2-theta));
delay=g.InputDelay + g.OutputDelay;
g = minreal(g);  % Cancel duplicate pole/zero pairs
[num, den] = tfdata(g);
NUM = num{1} .* M.^(length(num{1})-1:-1:0);
DEN = den{1} .* M.^(length(den{1})-1:-1:0);
G = tf(NUM, DEN);  % Modified transfer function
delay = delay * (pi - theta);  % Modified delay for damping requirement
G.InputDelay = delay;
[z, p, ~] = zpkdata(g);  % Find roots
frels = [abs(z{1}) abs(p{1})]';
frels = unique(frels(frels==0));  % Remove zeros and duplicates
maxfreq = 10*max(frels);
minfreq = 0.1*min(freqs);
freqcenter = 10^((0.5*(log10(minfreq) + log10(maxfreq)))); % logarithmic center of roots
% finite frequency will scale from 0 to 2 using arclog scale
alpha = log10(abs(freqcenter));
beta = log10(abs(maxfreq)) - alpha;
steps = 35;
frequency = zeros(steps, 1);
bmin = zeros(steps, 1);
for step = (1 : steps) % step through various observer bandwidths
    scaledwo = step*2/(steps+1);
    % convert frequency from finite scale to infinite scale
    wo = sign(scaledwo).*10.^((alpha+beta.*tan((pi/2).*(abs(scaledwo)-1))));
    G1 = -G*s^n;
    G2 = (s/wo+1)'*(n+1);
    G3 = tf(1);
    G3.InputDelay = controldelay * (pi - theta);
    % MATLAB cannot create LTI system using different delay terms
    G0 = G1/(G2-1); % Frequency response without control signal delay
    [-, ~, w] = bode(G0); % get relevant frequencies
    vec1 = freqresp(G1, w);
    vec2 = freqresp(G2, w);
    vec3 = freqresp(G3, w);
    vec1 = conj(vec1(:'));
    vec2 = conj(vec2(:'));
    vec3 = conj(vec3(:'));
    vec = vec1./(vec2 - vec3);
    mag = abs(vec);
    phase = atan2(imag(vec), real(vec)); % -pi <= phase <= pi
    b0pm = 0;
    b0gm = 0;
    for index = (1 : length(w)-1)
        startfreq = w(index);
        stopfreq = w(index+1);
        startphase = phase(index);
        stopphase = phase(index+1);
        startmag = mag(index);
        stopmag = mag(index+1);
        if (startphase == pm) && (startmag > b0pm)
            b0pm = startmag;
        elseif (stopphase == pm) && (stopmag > b0pm)
            b0pm = stopmag;
        elseif (startphase == 0) && (gm*startmag > b0gm)
            b0gm = gm*startmag;
        elseif (stopphase == 0) && (gm*stopmag > b0gm)
            b0gm = gm*stopmag;
        else
            % Does phase cross 0 or pm?
            if ((sign(startphase) == sign(stopphase)) || (sign(startphase-pm) == sign(stopphase-pm)))
                phasespan = stopphase - startphase;
                % Evaluate point at logarithmic middle frequency
                w_half = 10^((log10(startfreq) + log10(stopfreq))/2);
                half1 = freqresp(G1, w_half);
                half2 = freqresp(G2, w_half);
                half3 = freqresp(G3, w_half);
                half = half1./((half2-half3);
                half = conj(half(:'));
                mag_half = abs(half);
                phase_half = atan2(imag(half), real(half));
                % If point is actually between and curve between
                % endpoints, then sum of angles must be total angle.
                span1 = phase_half - startphase;
                span2 = stopphase - phase_half;
                if (abs(span1 + span2 - phasespan) < 0.001 ) %
within 0.1% tolerance

% Interpolation to get magnitude
if ((sign(startphase) == sign(phase_half)) ||
(sign(startphase-pm) == sign(phase_half-pm))
  % Use section between start point and midpoint
  slope = (mag_half - startmag) / (phase_half - startphase);
  intercept = startmag - slope*startphase;
  if (sign(startphase) == sign(phase_half)) % phase crosses 0
    interpolated_mag = intercept;
    if (gm*interpolated_mag > b0gm)
      b0gm = gm*interpolated_mag;
    end
  end
  if (sign(startphase-pm) == sign(phase_half-pm)) %
    phase crosses pm
    interpolated_mag = slope*pm+intercept;
    if (interpolated_mag > b0pm)
      b0pm = interpolated_mag;
    end
  end
else
  % Use section between midpoint and endpoint
  slope = (stopmag - mag_half) / (stopphase - phase_half);
  intercept = mag_half - slope*phase_half;
  if (sign(phase_half) == sign(stopphase)) % phase crosses 0
    interpolated_mag = intercept;
    if (gm*interpolated_mag > b0gm)
      b0gm = gm*interpolated_mag;
    end
  end
  if (sign(phase_half-pm) == sign(stopphase-pm)) %
    phase crosses pm
    interpolated_mag = slope*pm+intercept;
    if (interpolated_mag > b0pm)
      b0pm = interpolated_mag;
    end
  end
end
end
end
end
end
end
frequency(step) = wo;

E. MATLAB S-Function for FOPDT Identification

function [sys,x0,str,ts,simStateCompliance] = relay(t,x,u,flag)
% Relay system identification
% Author: Jason Tatsumi
% Master's Thesis at Cleveland State University
% 10/25/2013
% x1: discovery mode no delay/delay/off/controller (1/2/0/-1)
% x2: output positive/negative (1/-1)
% x3: k, area under response
% x4: tau
% x5: td
% x6: (reserved)
% x7: period1
% x8: period2
% x9: new y value
% x10: y value change time
% x11: timestamp of previous change
% x12: previous x11
% x13: previous x12
% x14: previous x13
% ...
% x30: previous x29

switch flag,
case 0,
    [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 2,
    sys=mdlUpdate(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
case 9,
    sys=mdlTerminate(t,x,u);
otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
end
end

function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes
sizess = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 30;
sizes.NumOutputs     = 30;
sizes.NumInputs      = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = zeros(30,1);
x0(1) = 1;
x0(2) = 1;
str = [ ];
ts  = [0 0];
simStateCompliance = 'UnknownSimState';
end

function sys=mdlDerivatives(t,x,u)
sys = [];
end

function sys=mdlUpdate(t,x,u)
    td = [0 10];  % two test time delays in seconds; (0 0.01)
xnew = x;
\[ e = u(1) - u(2); \]
\[ \text{if } x(1) > 0 \]
\[ x_{new}(3) = x_{new}(3) + (u(2)^2) \times 0.001; \quad \% \text{ multiply by sample time} \]
\[ \text{elseif } (t \leq x(11) + x(5)) \land (x(11) > 0) \land (x(5) > 0) \]
\[ x_{new}(3) = x_{new}(3) + (u(2)^2) \times 0.001; \quad \% \text{ multiply by sample time} \]
\[ \text{end} \]
\[ \text{if } x(1) > 0 \]
\[ y = x_{new}(2); \]
\[ \% \text{ output should change based on time delay} \]
\[ \% \text{ first calculate output without delay} \]
\[ \% \text{ if calculation differs from output,} \]
\[ \text{if } (x_{new}(9) < 1) \land (e > \text{eps}) \land (y < 1) \]
\[ x_{new}(9) = 1; \]
\[ x_{new}(10) = t + td(x(1)); \]
\[ \text{end} \]
\[ \text{if } (x_{new}(9) > -1) \land (e < -\text{eps}) \land (y > -1) \]
\[ x_{new}(9) = -1; \]
\[ x_{new}(10) = t + td(x(1)); \]
\[ \text{end} \]
\[ \text{if } (x_{new}(9) \neq 0) \land (t \geq x_{new}(10)) \]
\[ y = x_{new}(9); \]
\[ x_{new}(9) = 0; \]
\[ x_{new}(10) = 0; \]
\[ \text{end} \]
\[ \text{if } \text{abs}(\text{sign}(y) - \text{sign}(x(2))) > 0 \]
\[ \text{for } \text{index} = (30 : -1 : 12) \]
\[ x_{new}(\text{index}) = x_{new}(\text{index} - 1); \]
\[ \text{end} \]
\[ x_{new}(11) = t; \]
\[ \text{DELTA} = \text{eye}(20) - [\text{zeros}(19,1) \text{ eye}(19);\text{zeros}(1,19) \text{ 1}]; \]
\[ x_{new}(11:30); \]
\[ \text{period} = \text{DELTA} \times x_{new}(11:30); \]
\[ \% \text{ Are there positive values for periods 1 - 3?} \]
\[ \text{if } (\text{period}(1) > 0) \land (\text{period}(2) > 0) \land (\text{period}(3) > 0) \]
\[ \% \text{ Is (p1/p2) close enough to 1? ratio of 1.8 - 10\% average} \]
\[ \% \text{ This could be higher with adjusted average!} \]
\[ \text{ratio1} = (\text{period}(1)/\text{period}(2))^\text{sign}(\text{period}(1)-\text{period}(2)); \]
\[ \text{ratio2} = (\text{period}(2)/\text{period}(3))^\text{sign}(\text{period}(2)-\text{period}(3)); \]
\[ \text{avg1} = 0.5*(\text{period}(1)+\text{period}(2)); \]
\[ \text{avg2} = 0.5*(\text{period}(2)+\text{period}(3)); \]
\[ \% \text{ alpha is the adjustment coefficient} \]
\[ \text{alpha} = 0.145; \]
\[ \text{adjavg1} = \text{avg1} \times \exp(-\text{alpha}*(\text{ratio1}-1)); \]
\[ \text{adjavg2} = \text{avg2} \times \exp(-\text{alpha}*(\text{ratio2}-1)); \]
\[ \text{if } (\text{ratio1} < 1.1) \land (\text{ratio2} < 1.1) \]
\[ \text{adjavgratio} = (\text{adjavg1}/\text{adjavg2})^\text{sign(adjavg1-adjavg2)}; \]
\[ \% \text{ save period!} \]
\[ \text{if } x(1) == 1 \]
\[ x_{new}(7) = (\text{avg1}+\text{avg2}); \quad \% \text{ use adjustment if needed} \]
\[ x_{new}(1) = 2; \]
\[ \text{for } \text{index} = (30 : -1 : 12) \]
\[ x_{new}(\text{index}) = x_{new}(\text{index} - 1); \]
\[ \text{end} \]
\[ x_{new}(11) = 0; \]
\[ \text{end} \]
\[ \text{if } x(1) == 2 \]
\[ x_{new}(8) = (\text{avg1}+\text{avg2}); \quad \% \text{ use adjustment if needed} \]
\[ x_{new}(1) = 0; \]
\[ x_{new}(3) = x(3); \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{end} \]
xnew(2) = y;

end

if (xnew(1) == 0) && (x(1) == 0) && (xnew(7) > 0) && (xnew(8) > 0)
    tries = 0;
    A = 2*pi/xnew(7);
    B = 2*pi/xnew(8);
    max(td);
    mintau = -tan(B*(pi/A+max(td)))/B;
    maxtau = -tan(B*((2*A)+max(td)))/B;
    if (maxtau > mintau)
        maxdelay = ((pi-atan(A*maxtau))/A - td(1)) - ((pi-atan(B*maxtau))/B - td(2));
        mindelay = ((pi-atan(A*mintau))/A - td(1)) - ((pi-atan(B*mintau))/B - td(2));
        evalmaxtau = maxdelay;
        evalmintau = mindelay;
        if (sign(evalmaxtau) == sign(evalmintau)) % plot must cross 0!
            crossover = 0;
            crossovereval = Inf;
            while (abs(crossovereval) > 0.001*(maxdelay-mindelay)) && (tries < 10) % pick 2% of maxdelay-mindelay 0.001 and 10
                tries = tries + 1;
                m = (evalmaxtau - evalmintau) / (maxtau - mintau); % slope
                crossover = -evalmintau / m + mintau;
                crossovereval = ((pi-atan(A*crossover))/A - td(1)) - ((pi-atan(B*crossover))/B - td(2));
                if sign(crossovereval) == sign(maxdelay)
                    evalmaxtau = crossovereval;
                    maxtau = crossover;
                else
                    if sign(crossovereval) == sign(mindelay)
                        evalmintau = crossovereval;
                        mintau = crossover;
                    end
                end
            end
    end
end

xnew(4) = crossover;

if (xnew(4) > 0) && (xnew(5) > 0) && (xnew(3) > 0)
    tau = xnew(4);
    td = xnew(5);
    Kp = xnew(3);
    t2 = 0;
    p2 = 0;
    F2 = 0;
    k2 = -eps;
    A = 0;
    n = 0;
    for tindex = (0 : 19)
        eindex = 30 - tindex;
        t1 = xnew(eindex);
        if t1 > 0
            p1 = t1 - t2;
            k1 = ((-1)^n) - F2;
            if k2 == -eps
                k1 = 1;
                k2 = 0;
            end
            A = A + 0.5*tau*k1^2 + (F2^2 + 2*F2*k1 + k1^2) - 2*k1*tau*(F2 + k1) - 0.5*tau*k1^2*exp(-2*p1/tau) + 2*tau*k1*(F2 + k1)*exp(-p1/tau);
        end
        F1 = k1*(1-exp(-p1/tau)) + F2;
        % save current values as previous ones
        t2 = t1;
        p2 = p1;
F2 = F1;
k2 = k1;
n = n + 1;
end
end
xnew(6) = A;
end
else
xnew(1) = 1;
end
end
if (xnew(1) == 0) && (t > xnew(11) + xnew(5)) && (xnew(6) > 0)
t
w1 = 2 * pi / xnew(7)
w2 = 2 * pi / xnew(8)
k = sqrt(xnew(3) / xnew(6))
tau = xnew(4)
td = xnew(5)
xnew(3) = k;
xnew(1) = -1;
end
sys = xnew;
end
function sys=mdlOutputs(t,x,u)
y = x;
if (x(1) < 0)
y(2) = u(3);
end
sys = y;
end
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = .001;
sys = t + sampleTime;
end
function sys=mdlTerminate(t,x,u)
sys = [];
end