ML for Loop Gain Identification of DC/DC Converters

by

Cecelia Chu

S.B., Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2021)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2022

© Massachusetts Institute of Technology 2022. All rights reserved.

Author ................................................................

Department of Electrical Engineering and Computer Science

January 14, 2022

Certified by ............................................................

David J. Perreault
Professor of Electrical Engineering
Thesis Supervisor

Certified by ............................................................

Wenjie Lu
Research Scientist, Analog Devices, Inc.
Thesis Supervisor

Accepted by ...........................................................

Katrina LaCurts
Chair, Master of Engineering Thesis Committee
ML for Loop Gain Identification of DC/DC Converters

by

Cecelia Chu

Submitted to the Department of Electrical Engineering and Computer Science on January 14, 2022, in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

Control loop identification is necessary for evaluating the stability of switched power supplies and is therefore an important step during design and verification. Analytical models of power supplies often yield inaccurate predictions of the loop gain; therefore, power engineers traditionally must conduct slow, invasive loop gain measurements on physical hardware. This thesis presents an alternate approach to loop gain identification in which a machine learning model infers the frequency-domain loop response from the quick and convenient time-domain measurement of a load step transient. We show that we can train a neural network to accurately infer the loop gain of a current-mode buck converter over a generalized set of configurations and illustrate the disruptive potential of such a model with example applications such as live Bode plot monitoring and automatic loop compensation.

Thesis Supervisor: David J. Perreault
Title: Professor of Electrical Engineering

Thesis Supervisor: Wenjie Lu
Title: Research Scientist, Analog Devices, Inc.
Acknowledgments

I was fortunate to complete this work with the support and resources of Analog Devices, made available through the company’s partnership with the MIT 6-A program. In particular, I would like to thank Wenjie Lu, who supervised my thesis work at the Analog Garage and without whose exceptional guidance my research would not have been possible. I would also like to thank Mike Holloway, Henry Zhang, and Mark Hartman of Analog Devices for their generous contributions of time and expertise, as well as my manager, Shyam Nambiar, for his advocacy and mentorship.

I am grateful to Professor Perreault for his valuable feedback, and for inspiring my interest in power electronics through his teaching.

Thank you to all the professors, teachers, and mentors (at MIT and beyond) who have contributed to my ongoing education, and to my beloved family, friends, and felines for being the spice in my life.

Finally, a special thank you to my parents, Yonqing Lu and Yi Chu, for supporting me through it all and making me the person I am today.
### 3.4 Results

- 3.4.1 Tier 0 Results
- 3.4.2 Tier 1 Results, Cout Generalization
- 3.4.3 Tier 2 Results, Compensation Generalization
- 3.4.4 Tier 3 Results, 1-Phase and 4-Phase
- 3.4.5 Comparison to LTPowerCAD
- 3.4.6 Summary

### 4 Applications

- 4.1 Live Bode Plot Monitoring
- 4.2 Automatic Loop Compensation
  - 4.2.1 Motivation
  - 4.2.2 Implementation
  - 4.2.3 Discussion

### 5 Conclusion

- 5.1 Future Extensions

### A Daughterboard PCB Details

### B Preprocessing Code
List of Figures

1-1 Schematic of a basic buck converter [19]. .............................. 14
1-2 Simplified block diagram for a buck converter control loop. ......... 15
1-3 Sample loop gains and corresponding transient responses. .......... 16
1-4 Connection diagram for performing frequency analyzer measurements [16]. ................................................................. 18

2-1 Image of a DC2702A-B demo board. See [1] for complete schematic and component values. .................................................. 23
2-2 Schematic of LTM4700 compensation network with external capacitors. 24
2-3 Render of custom daughterboard for varying Cth capacitance. ...... 25
2-4 Schematic of dynamic load step circuit on the DC2702A-B. See [1] for component details. ..................................................... 27
2-5 Photograph of data collection hardware setup. .......................... 28
2-6 Effect of changing Rth on loop gain Bode plots. ....................... 31
2-7 Example data points from final dataset. ................................. 33
2-8 Plot of collected data points. .............................................. 34
3-1 High-level diagram of BodeNet structure. .............................. 38
3-2 Illustrated example of input preprocessing. ............................. 42
3-3 Examples of unstable configurations. ................................. 44
3-4 Plot of dataset cleaning results. ......................................... 45
3-5 Training history and learning rate ..................................... 47
3-6 Example BodeNet inferences. .......................................... 49
3-7 Histogram of test results for Tier 0 partition. ......................... 50
3-8 Plot of all gain and phase errors for Tier 0 test partition. . . . . . 51
3-9 Plots of phase margin (left) and bandwidth (right) errors for Tier 0
test partition. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
3-10 Histogram of Tier 1 test partition results. . . . . . . . . . . . . . 52
3-11 Plots of phase margin (left) and bandwidth (right) errors for Tier 1
test partition. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
3-12 Comparing results for standard (a) and non-standard (b) compensation
configurations, showing similar results. . . . . . . . . . . . . . . . . 54
3-13 Histogram for experimental 1-phase test results. . . . . . . . . . . 55
3-14 Histogram for experimental 4-phase test results . . . . . . . . . . 55
3-15 Comparing BodeNet inferences (blue) to analytical LTpowerCAD pre-
dictions (red), with ground truth Bode plots for reference (orange). . 57
4-1 User interface for live Bode plot monitoring demonstration. . . . . 60
4-2 Flowchart for automated compensation tuning with BodeNet. . . . 62
4-3 Example space of configurations for compensation tuning, with optimal
configuration marked. . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
4-4 Example result of predicting loop gain with \((gm, Rth, Cth) = (3.69,
1.5, 3300)\) (orange) from measured loop gain with \((gm, Rth, Cth) =
(2.35, 6, 22000)\) (blue) with \(Cout\) and \(Fsw\) fixed. . . . . . . . . 65
4-5 Results of simulating optimization algorithm on a dataset with \(Cout = 1980uF\) ceramic + 4x2R5TPE470M7 and \(Fsw = 425\) kHz. . . . . 66
4-6 Example results of auto-tuning demo running on live hardware. . . 67
A-1 Complete schematic of daughterboard. . . . . . . . . . . . . . . . . 71
A-2 Top and bottom renders of daughterboard PCB. . . . . . . . . . . 72
A-3 Top and bottom daughterboard PCB layers. . . . . . . . . . . . . . 72
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>List of parameters varied during standard data collection.</td>
<td>26</td>
</tr>
<tr>
<td>2.2</td>
<td>List of Tantalum capacitors used in data collection.</td>
<td>30</td>
</tr>
<tr>
<td>2.3</td>
<td>List of output capacitance configurations included in the dataset.</td>
<td>31</td>
</tr>
<tr>
<td>2.4</td>
<td>Values for parameters varied in standard data collection procedure.</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>Detailed encoder structure.</td>
<td>39</td>
</tr>
<tr>
<td>3.2</td>
<td>Detailed decoder structure.</td>
<td>40</td>
</tr>
<tr>
<td>A.1</td>
<td>Bill of materials for daughterboard components.</td>
<td>73</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Power supply design is becoming a ubiquitous task in the electronics industry as system demands on power solutions grow increasingly complex and customer-specific. Instead of using commercial off-the-shelf supply bricks, the need for multiple power rails and strict constraints on area, performance, and cost mean that engineers often must design their own supplies to meet specifications [19]. This section will provide background on power converters in order to introduce the challenge of loop gain identification during power supply design and motivate the need for a novel Machine Learning tool to assist in this task.

1.1 Background on Power Converters

Switched power supplies such as buck converters are a popular choice due to their high efficiency. A well-designed buck converter can achieve power efficiencies of above 90% in most applications [18].

A simplified model of a buck converter’s power stage is drawn in Figure 1-1. The converter takes a DC input voltage $V_{in}$ and produces an output voltage $V_{out} = D \times V_{in}$, where $D$ is the duty cycle of the switch.

In practice, feedback is necessary in order to regulate the duty cycle and maintain a constant output voltage. The design of this control loop is nontrivial and an active area of research. Many different control schemes for buck converters exist, including
voltage mode control, current mode control, and $V^2$ constant on-time control [18]. The purpose of the control loop is to maintain the stability of the output (avoid oscillations) and ensure fast responses to changes in conditions. Power supplies often face rapid transients in their load currents, also known as load steps; the loop characteristics determine how much the output voltage deviates from the operating point after the perturbation as well as how quickly it settles back to steady-state.

Taking feedback into account, we can represent the buck converter according to the block diagram shown in Figure 1-2.

From this block diagram, we can define the open loop gain as follows:

$$G(j\omega) = G_{pwm}(j\omega) \times G_{filt}(j\omega) \times G_{EA}(j\omega) \times G_{comp}(j\omega), \quad (1.1)$$

where $G_{pwm}$ is the gain of the modulator, $G_{filt}$ is the gain of the output filter, $G_{EA}$ is the gain of the feedback divider and error amplifier, and $G_{comp}$ is the gain of the compensation network.

The open loop gain determines the performance and stability of the system and is commonly represented using a Bode plot. The speed of the transient response generally corresponds to the loop gain bandwidth, which is defined as the crossover
frequency where the gain magnitude reaches unity (0 dB). Many different metrics for stability margin exist, all which essentially measure how far the loop gain transfer function is from the (0 dB, -180 degrees) point of positive feedback. The most commonly used metric in industry is phase margin, which measures the phase at the crossover frequency. Typically, a phase margin of at least 45-60 degrees from -180 indicates a sufficiently stable system [19] [7].

Figure 1-3 illustrates three different examples of loop configurations. 1-3a shows a loop gain with high phase margin (very stable) but low bandwidth, causing slow transient load step response with large undershoot. 1-3b has high bandwidth but is close to becoming unstable, as indicated by low phase margin and oscillations in the transient. This would be an unacceptable solution, as small changes in operating condition and part variation may cause the output to start ringing. 1-3c is an optimal solution displaying both high bandwidth and sufficient stability margin.

Characterizing a power supply’s stability margin is essential for ensuring safety and reliability. However, as the examples above illustrate, determining stability is a nontrivial task which is usually undertaken through frequency-domain analysis of the loop gain. The next section will outline approaches to conducting this analysis.

Figure 1-2: Simplified block diagram for a buck converter control loop.
(a) High phase margin, low bandwidth.

PM = 71.02 deg; GBW = 11.33 kHz

(b) Low phase margin, high bandwidth.

PM = 14.46 deg; GBW = 55.66 kHz

(c) High phase margin, high bandwidth.

PM = 63.37 deg; GBW = 40.90 kHz

Figure 1-3: Sample loop gains and corresponding transient responses.
1.2 Motivation for ML Approach

Current approaches to power supply and control loop design are based on analytical models. Since switched mode power supplies are inherently time-variant nonlinear systems, the conventional practice is to create a small-signal linearized model, treating the switching elements as averaged sources and ignoring higher-order AC terms. In particular, the development of accurate analytical models for current-mode controlled converters, in which an inner loop regulates the inductor current to a level set by the compensator output, is an active area of research. First-order equivalent circuit models derived using the averaging method, which approximate the inductor as a controlled current source, as well as more accurate extended models which account for higher-order effects of the inductor current ripple and artificial ramp, can provide useful insights about low-frequency behaviour [8] [6]. More recent works have attempted to develop more complex linear time-invariant models which capture small-signal behaviour across the full frequency range, though the usefulness and accuracy of such models in regions approaching half the switching is questionable due to time-varying effects [14] [10] [13].

In addition to being difficult and time-consuming to derive, especially as new control scheme variations are constantly being developed, analytical models incur significant error compared to physical hardware due to mismatch in parameter values and the inability to account for board layout effects and other parasitics. As a result, even for known topologies, analytical models can only be used as guidelines during simulation. Once the power supply is implemented in hardware, the loop gain must be verified by measurement.

Traditional measurement of the open loop gain requires breaking the loop and injecting a sinusoidal perturbation; the resulting disturbance at the other end of the loop is used to determine the gain and phase shift. Typically, this is done by inserting a small resistor in the feedback divider. A frequency analyzer instrument is required to perform the signal injection and measurement. In order to characterize the response over the full frequency range, multiple measurements at different frequencies must
be made with sufficient resolution. Additionally, the amplitude of the injected signal must be properly adjusted to avoid saturation for large gains and over-attenuation for small gains. This means that different amplitudes must be chosen for different frequencies, and improper choices can yield erroneous measurement results [7] [16].

A diagram of the frequency analysis setup is shown in Figure 1-4. Importantly, the breaking of the control loop with an additional resistor makes this an invasive measurement that requires modifying hardware. Due to the sensitive nature of the measurement, wire leads must be directly soldered to the board. In cases where the feedback loop is not conveniently accessible, this measurement becomes infeasible or extremely inconvenient. Furthermore, the measurement is slow as it requires sweeping a large range of frequencies. Additionally, it requires specialized equipment and an operator with sufficient expertise to adjust the settings. For these reasons, verification and tuning of the loop gain is an exceedingly manual procedure requiring time from an experienced power engineer, and is often a bottleneck in the critical path to delivery [16] [7].

There also exist several non-invasive methods for indirectly measuring the loop
gain which are based on output impedance measurements. The relationship between output impedance ($Z_{\text{out}}$) and loop gain is as follows:

$$Z_{\text{out\_closed}}(s) = \frac{Z_{\text{out\_open}}(s)}{1 + G_{\text{loop}}(s)}$$  \hspace{1cm} (1.2)

Closed-loop output impedance can be measured using a hardware setup that only requires access to the output terminal of the converter. A frequency analyzer instrument injects sinusoidal current waveforms of varied frequencies at the output and measures the corresponding output voltage response to obtain the gain-phase plot of the output impedance. However, in order to solve for the loop gain, the open-loop output impedance is also needed. In situations where modifying the loop to obtain the open-loop impedance is inconvenient, this measurement can be approximated as the un-powered output impedance of the converter, though in practice it is quite sensitive to the DC voltage applied to the output due to bias dependence of bulk capacitors [7].

An alternate method which uses only the closed-loop output impedance method is known as Non-Invasive Stability Measurement (NISM). It assumes that the output filter is a second-order LC filter and calculates the phase margin from the group delay and the peaking of the closed-loop impedance. The major drawback to this method is that it only provides the phase margin number and not the rest of the loop gain Bode plot which can provide valuable context for diagnosis [7].

While the output impedance based methods for approximating loop gain have the benefit of being non-invasive and working even in situations where the feedback loop is inaccessible, they still have the drawbacks of being slow, requiring specialized equipment, and being sensitive to measurement location and noise. Clearly, there is much room for improvement where a fast, accurate, and data driven ML model could prove advantageous.

In particular, we propose an ML model which can infer the Bode plot of the loop gain from a load step transient measurement alone. Such an approach would have the advantage of being based on physical hardware and therefore would avoid
the pitfalls of idealization faced by analytical models, while replacing the invasive frequency analyzer measurement with a quick and simple measurement of the load step transient response.

1.3 Thesis Outline

The goal of this thesis is to explore an ML approach to extracting loop gain from load step transient response. We would like to provide a proof-of-concept for a ML pipeline which can be used to develop generalized models for power supplies in an efficient and accurate manner. This thesis will present the work in three main parts. We will begin with data collection, which will be discussed in Chapter 2. Chapter 3 will discuss the model structure, training procedure, and inference results. Chapter 4 will explore sample applications of the resulting model. Finally, Chapter 5 will conclude with a summary of the novel contributions and discuss opportunities for future extensions of this work.
Chapter 2

Data Collection

In order to train an accurate and useful ML model, it is necessary to obtain a labeled dataset. Not only does the quality and composition of the dataset have a heavy influence on the resulting model, but the effort of collecting training data dictates how feasible it is to deploy the training pipeline in practice. Thus, our goals were to collect quality data representing a broad variety of configurations and operating conditions, and to do so in an automated, efficient way. This chapter discusses how we defined and implemented a data collection procedure to achieve these goals.

2.1 Scope and Requirements

Since we wish to train a model that infers loop gain frequency response from the load step transient response, our dataset must consist of pairs of transient and frequency responses corresponding to different power supply configurations. To make the dataset, and by extension the trained model, as generalized as possible, we want to collect data for a wide variety of configurations.

We define a configuration of a power supply as a set of settings and operating conditions which yields a specific loop gain. Since the loop gain can be viewed as a fixed property of the configuration, each unique frequency response in the dataset corresponds to one configuration. However, one configuration can produce many different load step transient responses by varying the characteristics of the load current.
step. Thus, the mapping of transients to frequency responses in our dataset is many-to-one.

For the sake of tenability, in this thesis we restrict our problem space to one specific part and board: the LTM4700 with the DC2702A-B demo board [2] [1]. Thus, our dataset collection procedure must be able to measure samples which cover the range of configurations for this supply.

The LTM4700 is a buck converter from the Analog Devices µModule family of power products which uses peak current-mode control. The power stage, controller, and output inductor are all integrated into the same package, making it a convenient choice for engineers who face time and space constraints and require a reliable, efficient solution. The converter supports input voltages from 4.5 to 16V and is capable of producing output voltages between 0.5 and 1.8V. It has two output rails which can operate in single phase to provide up to 50A each, or operate in parallel to provide up to 100A [2]. In this project, we focus on the parallel, 2-phase configuration.

In addition to the variable operating point, the LTM4700 has many other parameters which are either programmable over the PMBus I2C interface or controlled by externally populated components [2].

Even for a single part and board, the space of possible configurations is still very high-dimensional. The following section will discuss the implementation of the setup used to collect data points from this space.

### 2.2 Hardware Setup

The hardware setup for data collection can be roughly divided into two functional blocks. The first block varies the configuration of the power supply, and the second block collects the inputs and outputs by performing the frequency analyzer loop gain measurement and load step transient measurements.

We will first describe how the power supply is set up and configured, then describe the process for performing measurements and the automation.
2.2.1 Power Supply Configuration

For data collection, we use a standard demo board for the LTM4700, the DC2702A-B, pictured in Figure 2-1. The board contains one µModule configured for 2-phase parallel operation and has an on-board dynamic load step circuit [1]. The LTM4700 has an I2C-based PMBus programming interface which is used to read and write registers. We connect a lab PC to the part through a DC1613A USB dongle, so that a Python script can perform read/write operations to the LTM4700 using the pyserial library. The LTM4700 has many configurable registers, but for the purposes of our data collection we chose to only vary the registers which control the compensation values \( gm \) and \( R_{th} \), switching frequency, and output voltage.

Additionally, some other important parameters, the bulk output capacitance and the Type II compensation capacitors \( C_{th} \) and \( C_{thp} \) are not programmable and must be configured by changing external components.

The main role of the output capacitance is to provide filtering for the switching
ripple; it forms a second order low-pass LC filter with the power stage inductor and contributes a low-frequency pole as well as an ESR zero [18]. Changing either the total amount of output capacitance or switching to capacitors with different material, size, or ESR, can have significant impacts on the performance and stability of the power supply. In fact, mismatch between nominal and physical capacitance and ESR values due to manufacturing tolerance and operating conditions is one major source of error between analytical models of power supplies and their hardware counterparts. Therefore, it was important to vary output capacitance between different configurations during data collection. However, power supplies are quite sensitive to parasitic resistances and layout/placement of the bulk capacitors due to the high current requirements so we had to manually solder and desolder components to vary the output capacitance [17].

Cth and Cthp are two capacitors in the LTM4700’s Type II compensation network, shown in Figure 2-2. While the compensation resistor Rth is internally programmable, Cth and Cthp must be configured by changing external discrete components [2].

Cth controls the location of a low frequency pole and a zero. The placement of the zero at $\frac{1}{R_{th}C_{th}}$ has a strong impact on the loop bandwidth and phase margin and therefore on the transient response as well, so Cth was an important value to be able to vary during data collection. Automatic variation of the Cth value without having to manually solder and desolder capacitors was achieved through a custom-
designed daughterboard, which consists of a parallel bank of 8 SPST analog switches (ADG841), each in series with a discrete 0603 ceramic capacitor (see Appendix A for details).

The switches are independently opened or closed to control the total capacitance value seen at the output port (C+ and C- terminals in Figure 2-3), which is flywired to the Cth footprint on the demo board. Since the \( \frac{1}{R_{th}C_{th}} \) zero is relatively low frequency, the additional series resistance (0.28 \( \Omega \), nominal) of the analog switches and the flywire is assumed to be negligible relative to \( R_{th} \). This assumption was confirmed as no noticeable difference was found between measured Bode plots with the same capacitance value achieved using a discrete component soldered directly on the demo board versus the daughterboard.

During data collection, the daughterboard switches were controlled using the digital output pins of an ADALM2000 Active Learning module (M2K) which was controlled by a Python script running on the lab PC [3].

As seen in Figure 2-2, the LTM4700 contains an internal 22pF capacitor for Cthp which is typically sufficient for designs as this sets the Cthp pole above the crossover frequency of the loop but below the switching frequency to attenuate noise [19]. Since the pole is beyond the loop bandwidth, we do not expect it to impact the transient response significantly and therefore do not vary Cthp during data collection.

Finally, the input rails of the converter are supplied by a HP6654A power supply unit, which can be programmed to vary the Vin voltage level and supplies a maximum 9A current.
Table 2.1: List of parameters varied during standard data collection.

Table 2.1 summarizes the different parameters we vary in the hardware setup to achieve different power supply configurations.

### 2.2.2 Measurements

The second block of the data collection hardware was concerned with measuring the frequency domain loop response and the transient load step responses.

The frequency domain loop gain measurement is conducted using a frequency analyzer. As discussed in Chapter 1, this is an invasive measurement which involves injecting a varying frequency sinusoidal signal of known amplitude into the loop and measuring the resulting gain and phase. The DC2702 demo board contains a small 10 Ω resistor which can be used for signal injection; however, the board still had to be modified for the measurement by soldering on twisted pairs of wire. These formed a 4-wire Kelvin sense connection from which the frequency analyzer can conduct the bode plot measurement. The specific frequency analyzer we used to measure the Bode plots was the LTpowerAnalyzer board, a compact instrument which works with the ADALM2000 (M2K) oscilloscope. We sweep from 100Hz to 1MHz with 100 points.

The load step transient measurement is conducted via the on-board dynamic load step circuit shown in Figure 2-4. We use the waveform generator of an M2K oscilloscope to apply a 0.3 ms square pulse to the gate of the MOSFET, resulting in a current step on the output rail. This current is sensed via a 10 mΩ sense resistor; both output current and voltage waveforms are measured by the M2K at a 10 MHz
Figure 2-4: Schematic of dynamic load step circuit on the DC2702A-B. See [1] for component details.

sampling rate. We record a total of 8192 points (0.8192 ms) for each waveform, consisting of approximately 0.2 ms before the rising edge of the current step, 0.3 ms points during the pulse, and 0.3 ms after the falling edge. We vary the amplitude of the square wave to vary the size of the applied load step; due to part variation we have to empirically determine what voltages correspond to the desired step amplitudes.

A photo of the final hardware setup is shown in Figure 2-5.

The programmable power supply, as well as the three M2Ks which perform transient measurement, frequency response measurement, and control the Cth daughter-board, are connected to a lab PC via USB. The LTM4700 itself is connected to the PC via the DC1613a PMBus dongle. The PC runs a master Python script which configures the parameters, performs the frequency analyzer measurement, then takes the transient measurements.

The frequency measurement is by far the slowest step in the process. Overall, we are able to collect data for one power supply configuration in around 30 seconds.

Since all configuration parameters other than output capacitance are programmable by the PC, the script can run in a loop and collect data for a list of different configurations for a fixed output capacitance without requiring manual intervention.
2.2.3 Output Format

The collection script saves data points as pickled Python objects which contain a dictionary organized as follows:

1. Transient response

2. Frequency response

3. Metadata

The transient response is represented as 3 vectors of length 8192: \( t \), representing the time of each index, \( v \), the measured output voltage where \( v_i = v(t_i) \), and \( I \), the measured output current where \( I_i = I(t_i) \). Since the sample rate of the ADC is 10MHz, we have that \( t_{i+1} = t_i + 100\text{ns} \). The frequency response is likewise represented as 3 vectors of length 100: \( \omega \), representing the frequency points in Hz, \( G \), representing the gain of the loop gain response in dB, and \( \Phi \), representing the phase shift of the loop gain response in degrees.
The metadata is a table containing all of the loop gain configuration settings, operating point information, and load step amplitude size which uniquely identify each data point.

2.3 Methodology

The hardware setup described above enabled us to collect measurements for any desired configuration. The next task was then to determine what subset of possible configurations we wanted to include in the dataset. This can be seen as a tradeoff between covering more variety in loop gain behaviour versus limiting total time and effort required for data collection. We define the optimal point as the smallest set that enables consistent model performance across the range of configurations, such that adding additional data points does not lead to noticeable improvement.

The space of configurations that we aim to cover is all the permutations of the variable parameters listed in Table 2.1. Some of the parameters (such as \( gm \), \( R_{th} \)) take on discrete values, but others (\( V_{in} \), \( V_{out} \)) are continuous or unbounded (\( C_{out} \)). Therefore, covering the space with a limited number of data requires strategic sampling.

We first reduce the dimension of the configuration space by using the observation that two of the parameters, input and output voltage, have a much smaller, less direct impact on loop gain and transient response. We choose to exclude these parameters when generating permutations, and instead assign a randomly chosen input and output voltage to every configuration. We choose input voltages between 8V and 12V with 1V resolution and output voltages between 0.7V and 1.8V with 100mV resolution.

Next, we must determine what values of output capacitance to include in our dataset. In the most general sense, the output capacitance configuration is a combination of many different individual capacitors in parallel, each of which has its own part number. The relevant characteristics of different part numbers are the capacitance value, material/type, and parasitics (ESR). In practice, power supplies usually
<table>
<thead>
<tr>
<th>Part Number</th>
<th>Manufacturer</th>
<th>Capacitance (uF)</th>
<th>Nominal ESR (mΩ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6TPE330MAA</td>
<td>Panasonic</td>
<td>330</td>
<td>10</td>
</tr>
<tr>
<td>ETPF470M5H</td>
<td>Panasonic</td>
<td>470</td>
<td>5</td>
</tr>
<tr>
<td>2R5TPE470M7</td>
<td>Panasonic</td>
<td>470</td>
<td>7</td>
</tr>
<tr>
<td>2R5TPF680ML</td>
<td>Panasonic</td>
<td>680</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 2.2: List of Tantalum capacitors used in data collection.

have some number of ceramic capacitors, all of the same part number, optionally in addition to some number of tantalum capacitors, all of the same part number [18]. Additionally, we apply the assumption that ceramic capacitors all have very low ESR and therefore we can approximate ceramic capacitors as a single bulk capacitance value, regardless of the part number. Thus, we are able to reduce our description of the output capacitance to 3 dimensions: total ceramic capacitance, part number of tantalum capacitor, and number of tantalum capacitors. Due to the fixed number of capacitor footprints on the demo board, we are further subject to the restriction of having at most 18 total capacitors, including up to 4 tantalum capacitors.

Taking these factors into account, we chose 4 different types of tantalum capacitors with varying capacitance values and nominal ESRs, shown in Table 2.2. For ceramic capacitors, we used the Taiyo Yuden AMK325AC6337MM-P, a 330uF capacitor which is the default part for the demo board.

We mixed and matched different combinations of ceramic and tantalum capacitors during our dataset collection with the goal of covering variations in total bulk capacitance, ESR, and ceramic to tantalum ratios. Table 2.3 lists the 12 output capacitance configurations included in our final dataset.

We further reduce the size of the dataset by reducing the resolution with which gm, Rth, and Cth are swept when generating permutations. The hardware supports 8, 32, and 8 different values of gm, Rth, and Cth respectively, which allows for very fine-grained control over the loop gain phase margin and bandwidth. Figure 2-6 contains Bode plots and transient responses of 32 different configurations where Rth is swept from the minimum to maximum programmable value and all other configuration parameters are held constant. The effect of changing Rth is consistent enough that
<table>
<thead>
<tr>
<th>#</th>
<th>Ceramic</th>
<th>Tantalum</th>
<th>Tantalum PN</th>
<th>Total Capacitance (uF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>0</td>
<td>N/A</td>
<td>2640</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>2</td>
<td>6TPE330MAA</td>
<td>3300</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>2</td>
<td>2R5TPF680ML</td>
<td>3340</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>4</td>
<td>6TPE330MAA</td>
<td>3960</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>4</td>
<td>ETPF470M5H</td>
<td>4520</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>0</td>
<td>N/A</td>
<td>4620</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>4</td>
<td>2R5TPE470M7</td>
<td>5180</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>4</td>
<td>2R5TPF680ML</td>
<td>5360</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>4</td>
<td>ETPF470M5H</td>
<td>5840</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td>4</td>
<td>6TPE330MAA</td>
<td>5940</td>
</tr>
<tr>
<td>11</td>
<td>14</td>
<td>4</td>
<td>2R5TPE470M7</td>
<td>6500</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>4</td>
<td>2R5TPF680ML</td>
<td>7340</td>
</tr>
</tbody>
</table>

Table 2.3: List of output capacitance configurations included in the dataset.

intuitively, the model should be able to interpolate intermediate curves even if only trained with a downsampled set of values.

Thus, in our final standardized dataset collection procedure, we sweep only every other $R_{th}$, $gm$, and $C_{th}$ value, reducing the required dataset size and collection time by a factor of 8. We verify after training that this is sufficient for the model to interpolate successfully such that it performs no worse on the intermediate parameter values (See Section 3.4.3).

Finally, for each power supply configuration, we excite the circuit with 3 different sizes of load current steps: 6A, 10A, and 20A. Each of these three transient responses is mapped to the same Bode plot measurement.
<table>
<thead>
<tr>
<th>Parameter</th>
<th># Swept Values</th>
<th>Swept Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rth</td>
<td>16</td>
<td>0, 0.5, 1, 1.5, 2.0, 3.0, 4.0, 5, 6, 8, 11, 15, 20, 28, 38, 54 (kΩ)</td>
</tr>
<tr>
<td>gm</td>
<td>4</td>
<td>1, 2.35, 3.69, 5.04</td>
</tr>
<tr>
<td>Cth</td>
<td>4</td>
<td>1000, 3300, 10000, 22000 (pF)</td>
</tr>
<tr>
<td>Fsw</td>
<td>7</td>
<td>250, 350, 425, 500, 575, 650, 750 (kHz)</td>
</tr>
<tr>
<td>Cout</td>
<td>12</td>
<td>See Table 2.3</td>
</tr>
<tr>
<td>Vin</td>
<td>Randomly assigned</td>
<td>8-12V</td>
</tr>
<tr>
<td>Vout</td>
<td>Randomly assigned</td>
<td>0.7-1.8V</td>
</tr>
<tr>
<td>Load Step Amplitude</td>
<td>3</td>
<td>6, 10, 20 (A)</td>
</tr>
</tbody>
</table>

Table 2.4: Values for parameters varied in standard data collection procedure.

The structure of our final dataset, constructed to be a minimal representation of the configuration space, is shown in Table 2.4.

## 2.4 Results

To summarize, the primary contributions of this part of the thesis were first to generate a generalized set of loop gain Bode plots and transient responses for different configurations of the LTM4700, and second to create a automated system for performing efficient data collection.

First, we will present the collected dataset. Using the methods described in the previous section, we obtained a dataset consisting of over 70,000 data points corresponding to more than 20,000 unique AC configurations. Two examples of these data points are shown in Figure 2-7.

Figure 2-8 illustrates the breadth of the dataset by plotting the configurations by the bandwidth and phase margins of their loop gains. For comparison, points resulting from varying only gm and Rth with all other parameters fixed are overlaid. We can see that the variation of additional parameters drastically increases the range of frequency responses that we are able to cover.

Second, let us summarize the data collection workflow that was developed. We built a hardware setup based on compact M2K oscilloscopes and standard power...
Gm = 5.04 mS; Rth = 28.00 kOhm; Cth = 1000pF
Cout = 14x330uF; FSW = 750 kHz; Vin = 8.0V;

PM = 53.92 deg; GBW = 44.38 kHz

Gm = 3.69 mS; Rth = 2.00 kOhm; Cth = 3300pF
Cout = 8x330uF_4x680uF10m; FSW = 250 kHz; Vin = 9.0V;

PM = 28.26 deg; GBW = 10.95 kHz

Figure 2-7: Example data points from final dataset.
supply demo boards, which is capable of automatically collecting hundreds of high-quality training data with a single Python script. Far from being an arduous or costly process, the data collection can be run mostly unsupervised and only rarely requires manual intervention to replace output capacitors. The system collects data for one unique loop gain configuration approximately every 30 seconds, so a full training set can be collected in a handful of days with very little active labor. Thus, we have achieved an important proof of concept that suggests the logistical feasibility of scaling this approach to other power supplies.

We must, however, acknowledge some limitations of our data collection approach. One parameter which we did not vary during data collection was temperature; power supplies often operate in extreme temperature conditions which can cause significant shifts in behaviour [11]. Future work could address this by adding a thermal chamber to the hardware setup so that temperature could be varied as an additional configuration parameter. Another parameter that was not varied was the slew rate of the load step itself. While we did vary the amplitude of the load step from 6 to 20A, these
pulses were all essentially square waves. Adding some method to control the slew rate of the load step current would be a necessary step in the future to investigate the ability of the model to generalize across different slew rates. A final limitation is that the data collection procedure assumes the existence of a PMBus interface for configuring the part, which is not universal to all power product families. However, the procedure should be easily adaptable to different programming interfaces. In the scenario that compensation networks are completely external, the daughterboard approach could be extended to vary resistance values as well.
Chapter 3

Modelling and Training

Having established a procedure for generating labeled training data, the next stage of the project was to create a model that could be applied to the learning task. This chapter describes in detail the development of the model structure (which we call BodeNet), data preprocessing and training procedures, and the performance of the final trained model.

3.1 BodeNet Model

3.1.1 Input and Output Features

BodeNet is constructed to consume an input of 1D vectors of fixed length $N_{\text{in}}$. These represent the transient voltage waveform. Since there is no separate time input, it is implied that the time scale is linear and predetermined.

The output is the Bode plot of the loop gain, represented as a length $2^*N_{\text{out}}$ vector, $Y$. The first $N_{\text{out}}$ points are the gains $G_i$ in decibels, and the second $N_{\text{out}}$ points are the phases $\Phi_i$ in degrees (-180 to +180), such that we have $Y_i = G_i$ and $Y_{i+N_{\text{out}}} = \Phi_i$ for $0 \leq i < N_{\text{out}}$. Furthermore, $G_i = G(\omega_i)$ and $\Phi_i = \Phi(\omega_i)$, where the $\omega_i$’s are the predetermined target frequencies, in Hz, chosen such that the $\omega_i$’s are evenly spaced between some $\omega_{\text{MIN}}$ and $\omega_{\text{MAX}}$ on a log scale. In our modeling experiments, we chose $N_{\text{out}} = 128$, $\omega_{\text{MIN}} = 100$ Hz, and $\omega_{\text{MAX}} = 10^{5.3}$ Hz.
3.1.2 Model Structure

BodeNet is a neural network with an encoder-decoder structure inspired by autoencoders, a family of unsupervised machine learning models that encode high-dimensional inputs (such as images) into a low-dimensional latent space and then reconstruct the input from the latent representation [15]. When adapting the autoencoder approach to our supervised learning task, we preserve the encoder, which compresses the input transient response into a low-dimensional latent vector. Instead of recreating the input, the decoder is trained to generate the Bode plot of the frequency response that corresponds to the given input. The intuition behind this structure is that the latent vector is essentially a compact representation of the control loop itself, analogous to an analytical model, from which both time domain and frequency domain responses can be derived. A high-level diagram of the model architecture is shown in Figure 3-1.

Next, we will elaborate on the details of the encoder and decoder.

The encoder consists of a head layer, followed by a sequence of convolution layers and then a fully connected layer. The head, which directly consumes the input, is a 1D convolution (Conv) followed by a ReLu activation (Act) layer. The subsequent convolution layers consist of 1D convolutions, batch normalizations (BN), activations, and dropout (DO) with $p=0.1$, where every other convolutional layer halves the size of its input with a stride of 2. The final convolutional layer is a 1x1 convolution.
<table>
<thead>
<tr>
<th>Layer Types</th>
<th>Kernel Size</th>
<th># Channels</th>
<th>Output Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv/Act</td>
<td>7</td>
<td>32</td>
<td>193</td>
</tr>
<tr>
<td>Conv/BN/Act/DO</td>
<td>3</td>
<td>32</td>
<td>97</td>
</tr>
<tr>
<td>Conv/BN/Act/DO</td>
<td>3</td>
<td>32</td>
<td>97</td>
</tr>
<tr>
<td>Conv/BN/Act/DO</td>
<td>3</td>
<td>32</td>
<td>49</td>
</tr>
<tr>
<td>Conv/BN/Act/DO</td>
<td>3</td>
<td>32</td>
<td>49</td>
</tr>
<tr>
<td>Conv/BN/Act/DO</td>
<td>3</td>
<td>32</td>
<td>25</td>
</tr>
<tr>
<td>Conv/BN/Act/DO</td>
<td>3</td>
<td>32</td>
<td>25</td>
</tr>
<tr>
<td>Conv/BN/Act/DO</td>
<td>3</td>
<td>32</td>
<td>13</td>
</tr>
<tr>
<td>Conv/BN/Act/DO</td>
<td>3</td>
<td>32</td>
<td>13</td>
</tr>
<tr>
<td>Conv/BN/Act/DO</td>
<td>3</td>
<td>32</td>
<td>7</td>
</tr>
<tr>
<td>Conv/BN/Act/DO</td>
<td>3</td>
<td>32</td>
<td>7</td>
</tr>
<tr>
<td>Conv/Act</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>FC/Act/BN</td>
<td></td>
<td></td>
<td>16</td>
</tr>
</tbody>
</table>

Table 3.1: Detailed encoder structure.

Finally, the output of the convolution layers is flattened and a fully connected layer is applied to produce the latent vector of the desired size. The structure and key metrics of the encoder are summarized in Table 3.1.

The decoder network begins with a fully connected layer which consumes the latent vector. This is followed by a sequence of 1D transposed convolutions (TConv), batch normalization, ReLu activation, and dropout layers. Each transposed convolution doubles the size of the output due to the stride of 2. Similarly to the encoder, we end with a 1x1 convolutional layer, a flattening layer, and a fully connected layer.

Finally, we have two matching, parallel output heads which are each a single fully connected layer. Both consume the output of the decoder; one outputs the gain plot and the other outputs the phase plot.

Table 3.2 details the structure of the decoder.

The model has 90,578 trainable parameters in total and occupies around 350 kBytes when stored in ONNX format [12]. BodeNet’s relatively lightweight size makes it potentially deployable on embedded hardware.
### Table 3.2: Detailed decoder structure.

<table>
<thead>
<tr>
<th>Layer Types</th>
<th>Kernel Size</th>
<th># Channels</th>
<th>Output Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC/Act</td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>TConv/BN/Act/DO</td>
<td>4</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>TConv/BN/Act/DO</td>
<td>4</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>TConv/BN/Act/DO</td>
<td>4</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>TConv/BN/Act/DO</td>
<td>4</td>
<td>32</td>
<td>128</td>
</tr>
<tr>
<td>Conv/Act</td>
<td>1</td>
<td>1</td>
<td>128</td>
</tr>
<tr>
<td>FC/Act</td>
<td></td>
<td>128</td>
<td></td>
</tr>
</tbody>
</table>

#### 3.1.3 Loss Function

We define the loss function, which is minimized during training, to be a weighted mean-squared error (MSE) between the ground truth gain and phase plots and the model outputs.

\[
\text{Loss} = \sum_{i=1}^{N_{\text{out}}} W(\omega_i)(G_{\text{pred}}(\omega_i) - G_{\text{truth}}(\omega_i))^2 + W(\omega_i)(\Phi_{\text{pred}}(\omega_i) - \Phi_{\text{truth}}(\omega_i))^2
\] (3.1)

In Equation 3.1, \( G \) represents the gain (in dB) and \( \Phi \) represents the phase (in degrees), and \( \omega_i \) is the frequency corresponding to the i’th index of the output gain or phase vector, as in Section 3.1.1. \( W \) is a predefined frequency-dependent "weighting" function. For example, \( W(\omega) = 1 \) would indicate uniform weighting across all frequency points. During training, we defined \( W(\omega) \) such that the weight was 2 near the crossover frequency of the loop gain, was 1 elsewhere, and decayed to zero just before half the switching frequency. This was designed to avoid penalizing for errors above half the switching frequency, where the Bode plot inherently becomes noisy and meaningless due to non-linearity, and to emphasize accuracy in the crossover region which determines the phase margin and bandwidth.

#### 3.2 Data Preprocessing

Our collected data points must be preprocessed to convert them from the raw hardware output into the form consumed by the model. Input transient waveforms and
The input preprocessing, which must be performed for both training and inference and therefore must not rely on any frequency domain information, consists of the following stages:

1. Decimation
2. Cropping
3. Random crop (training only)
4. Standardization
5. Random vertical shift (training only)

The goal of decimation is to downsample the output voltage waveform measured by the oscilloscope. As mentioned in Section 2, the sample rate of the raw measurement is 10MHz, which means that the transient pulse contains 8192 points over 0.8192ms. Such high resolution is unnecessary from a theoretical standpoint since LTM4700 loop bandwidth typically does not extend past 100 kHz. Therefore, downsampling helps reduce the size of the model and improves robustness by decreasing the likelihood of overfitting. We set our downsampling factor to be 8 such that the effective sampling rate is 1.25 MHz. We implement our decimation by first applying an anti-aliasing FIR filter (forwards and backwards) and then simply keeping every 8th point.

Next, we crop our signal to only include the response to the rising edge of the current step. This has the benefit of halving the input size and removing dependence on load step width. We found through experimentation that doing so did not detract from model performance; this matches theoretical intuition.

During training only we perform an additional preprocessing step for data augmentation. The cropping window is randomly shifted by up to 25% of the final length.

We then standardize the output to center it around zero by subtracting the mean of the vector.
Again during training only, there is a final step for data augmentation where we randomly shift the waveform vertically by up to $\pm 5$ mV. Combined with the random horizontal cropping, this data augmentation makes the model more robust to small horizontal and vertical shifts in the input, forcing it to base inference on the shape of the waveforms rather than overfit to offsets.

The final preprocessed input has a length $N_{in} = 386$ points. An illustrated example of the input preprocessing is shown in Figure 3-2.

When preparing the ground truth outputs for training, we apply two preprocessing steps: interpolation and smoothing. As discussed in Section 3.1.1, we require the gain and phase outputs of the network to correspond to $N_{out} = 128$ fixed frequencies $\omega_i$, which range from 100 Hz to $10^{5.3}$ Hz and are evenly spaced on a log scale. Since the raw output of the frequency analyzer gives the gain and phase for a different set of frequencies, we use linear interpolation on the measured Bode plots to generate the gain and phase points for each frequency on the target grid. We then apply a Savgol filter with window length 11 to smooth out measurement noise.

See Appendix B for Python implementations of the preprocessing described above.

3.3 Partitioning and Experiment Procedure

3.3.1 Dataset Cleaning

Prior to generating our partitions, we needed to perform some dataset cleaning to remove highly oscillatory/unstable points. These examples, characterized by steady-state ringing or non-settling transients in the time domain and very low phase margin
in the frequency domain, often had noisy Bode plot measurements. Furthermore, the large-amplitude oscillations in the time-domain waveforms made them quite different from the majority of stable or marginally stable configurations that we were most interested in analyzing. A few examples of unstable configuration with both time and frequency plots are shown in Figure 3-3.

We exclude these data points by using a simple algorithm which detects oscillations in the voltage waveform and rejects if oscillations fail to settle by the falling edge of the current step. We also add some quality checks to remove cases where the oscilloscope triggered incorrectly. Since this filtering can be done using the input time domain waveform alone without frequency-domain knowledge, and such obviously ringing cases would need to be stabilized before undergoing further loop analysis or optimization in the real world, reducing the breadth of our dataset in this manner is justified.

Figure 3-4 the results of filtering out oscillating points on a phase margin vs bandwidth plot. Only a specific region of low phase margin points are excluded, while we still maintain complete coverage of the region of interest.

3.3.2 Partitioning

In order to evaluate the model, we had to divide the dataset into training and testing partitions. During training, the model has access to the ground truth Bode plots, and "learns" through backpropagation to produce outputs which match the ground truth in a way that minimizes the total loss defined in Section 3.1.3. During testing, the model performs inference on the inputs, and the resulting outputs are compared against the ground truth to obtain the testing loss. Achieving low testing loss indicates that the model is accurate and well-generalized.

For our modeling problem, we have several different definitions of generalization that we wish to test. In addition to the most basic sense of generalization, which tests that the model did not overfit to the training set, we also wish to confirm that the model is particularly generalized over variations in output capacitance; in other words that it is able to provide accurate inferences for configurations with output
Figure 3-3: Examples of unstable configurations.
capacitances not seen by the model during training. This is due to the fact that the difficulty of varying output capacitance makes it a bottleneck for data collection. We also wish to confirm that the model can infer accurately for configurations with \((gm, Rth, Cth)\) combinations that were skipped during data collection (see Section 2.3). Finally, we wish to characterize the performance of a model trained on only 2-phase configurations on data from single phase or 4-phase configurations. Therefore, we have several different testing partitions which correspond to these different "tiers" of generalization.

From the dataset we collect using the standard methodology described in Section 2.3, we divide the points by output capacitance such that one group contains 7 and the other contains 5 of the 12 capacitances. From the first group, we randomly select 50% of the configurations to use as training data for the model. The remaining 50% forms our Tier 0 testing partition, which is drawn from the same space as the training data. The second group is designated as the Tier 1 partition, which tests whether the model can infer accurately on configurations with unfamiliar output capacitances.
These partitions only contain data points from combinations of 16 Rth values, 4 gm values, and 4 Cth values according to the standard data collection methodology.

The Tier 2 testing partition was used to validate whether the model can provide good inferences for configurations with intermediate compensation values that were skipped in the standard methodology for efficiency reasons. It is a specially collected dataset covering a full sweep of permutations of 32 Rth values, 8 gm values, and 8 Cth values. Switching frequency and output capacitance are not varied, but load step, input voltage, and output voltage are varied in the same manner as in the standard methodology.

The Tier 3 test partition contains two small datasets collected from a 1-phase LTM4700 board and a 4-phase LTM4700 board. The single-phase board was the DC2702A-A, which has a layout identical to the DC2702A-B but is configured to provide two separate output rails. The 4-phase board was the DC2784A-A, which contains two LTM4700 modules with all output rails combined in parallel. For these boards, we modify the standard data collection procedure to just generate a small sample of points, sweeping only 4 values of switching frequency and not varying output capacitance or Cth.

Note that our dataset is partitioned by power supply configurations which have a one-to-one mapping with loop gain Bode plots rather than by transient responses, since one configuration maps to multiple transient responses corresponding to different sizes of load current steps. This removes any overlap between our training and testing partitions, ensuring that they are disjoint in the most conservative sense.

### 3.4 Results

We train the BodeNet model on the 13046 training points for 2000 epochs with batch size of 128, where each epoch iterates over 15 batches. 10% of the training dataset is used for validation. We use an Adam optimizer and a multistep learning rate scheduler with a 50 epoch warmup [9]. The total training time is about 3 hours on a single GTX1070 GPU.
Figure 3-5: Training history and learning rate
Figure 3-5 shows the training history of the model along with the learning rate. We see that in the final few hundred epochs, both loss and validation loss remain flat. This indicates that overfitting is not present, since otherwise one would expect loss to continue decreasing while validation loss stays constant or increases.

During testing, we quantify the accuracy of the model using several different metrics. For a single inference, we can compare the output of the model to the ground truth using the root mean-squared error (RMSE) of the gain and phase as well as comparing the bandwidth and phase margin error. While the RMSE gives a more holistic representation of the match between prediction and ground truth, phase margin and bandwidth error are useful metrics since loop gain characteristics around the crossover frequency are the primary drivers of design decisions in practice.

To evaluate the model performance on a given test partition, we calculate the gain RMSE, phase RMSE, phase margin error, and bandwidth error for each point in the partition and then look at the means and standard deviations across the partition. We desire a mean close to zero and a low standard deviation, which would indicate that the model outputs can be used with a high level of confidence.

### 3.4.1 Tier 0 Results

Figure 3-6 shows several examples of inferences from the test dataset. The model predictions closely match the ground truth outputs for a variety of loop configurations.

The model shows strong performance across the entire partition of 13088 data points. Figure 3-7 shows histograms for the accuracy metrics. In particular, we note the bell-shaped distributions centered near zero for the phase margin and bandwidth error metrics. The phase margin error has a mean of 1.01 degrees and a standard deviation of only 6.48 degrees, such that the vast majority of inferred phase margins fall within 10 degrees of the ground truth.

To further visualize our results, we plot the gain and phase errors vs frequency for every point in the dataset as shown in Figure 3-8. With the exception of some outliers, we can see that the curves match each other closely.

Additionally, we can visualize the distribution of phase margin and bandwidth
Figure 3-6: Example BodeNet inferences.
error as a function of the actual loop gain phase margin and bandwidths, as shown in Figure 3-9. Both phase margin and bandwidth error are the largest for regions with extremely high and extremely low phase margin, where the phase margins were under- and over-estimated, and the bandwidths were over- and under-estimated, respectively. However, the performance is quite consistent for majority of points.

3.4.2 Tier 1 Results, Cout Generalization

Having established strong performance on the Tier 0 testing dataset, we continue to evaluate the model for the Tier 1 dataset, which contains 17701 data points corresponding to different output capacitance configurations. The results for this dataset are shown in Figures 3-10 and 3-11.

The performance of the model on this dataset is remarkably similar to the performance on the Tier 0 dataset despite being composed of output capacitance configurations that are completely unfamiliar to the model. This indicates that the model has successfully generalized to a range of variation in output capacitance with just
Figure 3-8: Plot of all gain and phase errors for Tier 0 test partition.

Figure 3-9: Plots of phase margin (left) and bandwidth (right) errors for Tier 0 test partition.
Figure 3-10: Histogram of Tier 1 test partition results.

Figure 3-11: Plots of phase margin (left) and bandwidth (right) errors for Tier 1 test partition.
7 different configurations in its training data. Thus, we are optimistic that scaling our modeling approach to new parts would not require an unreasonable amount of manual output capacitance variation during data collection.

### 3.4.3 Tier 2 Results, Compensation Generalization

We evaluate the performance of the model on this test partition of 3954 points, which contains all 32 values of \( R_{th} \), 8 values of \( gm \), and 8 values of \( C_{th} \) in the sweep, to verify generalization of the model over the compensation network parameters. For consistency, we compare the performance of the model on the full partition versus the subset of points in the partition containing only the seen compensation network combinations (16 \( R_{th} \) values, 4 \( gm \) values, 4 \( C_{th} \) values). The results in Figure 3-12 shows no degradation in performance after introducing unseen compensation network configurations, thus we conclude that the downsampled sweep during data collection provides sufficient granularity for the model to be able to generalize.

### 3.4.4 Tier 3 Results, 1-Phase and 4-Phase

As an additional experiment, we test the performance of our model, which is trained on data from 2-phase configurations of the LTM4700 on the DC2702A-B demo board only, on small datasets collected using single phase and 4-phase operation.

The single phase dataset contained 480 points. The phase margin error had a mean of 4.04 and a standard deviation of 5.63.

The 4-phase dataset had 302 points, which not only had a different number of phases but also a very different demo board and output capacitance. The performance of the model on this dataset is summarized in Figure 3-14.

While the accuracy of the model is noticeably poorer compared to the previous 2-phase partitions, we still achieve reasonable performance. This leads us to believe that future iterations of our model would be able to provide accurate inferences for configurations with different numbers of phases if we included such variations in the training data.
Figure 3-12: Comparing results for standard (a) and non-standard (b) compensation configurations, showing similar results.
Figure 3-13: Histogram for experimental 1-phase test results.

Figure 3-14: Histogram for experimental 4-phase test results
3.4.5 Comparison to LTpowerCAD

We can use the loop gain Bode plots predicted by LTpowerCAD, Analog Devices’ software tool for power supply design, as a useful benchmark for comparison with the BodeNet inferences. It allows users to specify configurations for a power supply built using an ADI power product and outputs the expected characteristics, including the loop gain Bode plot and load step transient response [4]. LTpowerCAD is powered by complex analytical models based on the work of Jian Li and others [19]; these models have many parameters which must be manually fitted for every new released part to match physical measurements as closely as possible.

Despite the amount of engineering effort that goes into tuning the analytical model parameters, LTpowerCAD predictions still have significant error for reasons discussed in Section 1.2. Figure 3-15 shows ground truth and inferred Bode plots compared with the results from the LTpowerCAD analytical mode for four randomly chosen samples. The LTpowerCAD predictions have significant mismatches compared to the measured frequency response, whereas the BodeNet inferences, which are derived from measurements on physical hardware, are able to achieve much better fidelity.

3.4.6 Summary

In this section, we have shown that the BodeNet model, which is trained on 13046 points representing 7 output capacitance values, is able to infer loop gain Bode plots on test datasets with impressive accuracy. It achieves phase margin standard deviations of under 6.5 degrees, with consistent performance on power supply configurations with output capacitances or compensation values that are not contained in the training data. This indicates the model’s ability to generalize across a wide range of configurations and operating points despite a training dataset that is relatively small compared to the entire space of possible configurations. The model also provides impressively decent predictions for 1- and 4-phase configurations which are not at all represented in the training set. BodeNet is trainable within a few hours and is much more accurate than the LTpowerCAD simulator.
Figure 3-15: Comparing BodeNet inferences (blue) to analytical LTpowerCAD predictions (red), with ground truth Bode plots for reference (orange).
Chapter 4

Applications

The final part of this thesis work explores potential applications of the BodeNet model. The two example applications described in this chapter demonstrate the advantages of the novel BodeNet approach to loop gain identification compared to conventional methods.

4.1 Live Bode Plot Monitoring

One of the primary benefits of inferring loop gain from transient load step response is the noninvasive nature and rapid speed of the measurement. This is contrasted with frequency analyzer measurements like the one we used to collect ground truth, which not only require modifying the hardware to allow for signal injection but are also slow to run.

The Live Bode Plot Monitoring application illustrates these two advantages of the BodeNet approach, showcasing both its low-overhead ease-of-use and the benefits of real-time feedback. When connected to hardware, the demo application provides the user with live plots of the power supply’s load transient response and the Bode plot of the loop gain (plus derived phase margin and bandwidth) as inferred by the neural network (See Figure 4-1). The application runs as an extension of LTpowerPlay, ADI’s power supply debug tool [5]. Since LTpowerPlay allows users to interact with the hardware by reading and writing registers over the PMBus interface, users can
Figure 4-1: User interface for live Bode plot monitoring demonstration.

adjust programmable compensation values and immediately see the impact on the loop gain, all through the same graphical interface.

Apart from an oscilloscope to excite on-board load step circuitry and read resulting waveforms, no additional hardware is required; notably a frequency analyzer is not needed. Thus, this demo application shows how BodeNet enables loop gain plots to become much more conveniently accessible in a way that drastically streamlines the debug process for power supplies.

This application also had the secondary benefit of demonstrating the platform-agnostic and easily deployable nature of our model. We generated and trained BodeNet in a Python environment using the PyTorch library, but were able to deploy it in the C# environment of LTPowerPlay. This was done by exporting the model to ONNX format [12]. Since the input preprocessing requires only standard mathematical operations, these were easily translated into the new programming language. Thus, we can envision a deployment workflow where trained models for different parts are maintained in a cloud-based "model zoo," such that applications wishing to use the BodeNet would be able to generate inference results in their own native languages.
4.2 Automatic Loop Compensation

The Live Bode Plot Monitoring application allows users to view loop gain information in real-time; however, adjusting the compensation values to optimize the loop gain would still require an engineer in the loop. The application discussed in this section shows how BodeNet enables us to close the loop and optimize the compensation network automatically.

4.2.1 Motivation

Tuning the compensation network of the control loop by appropriately choosing a set of component values is a crucial step in power supply design. The aim of compensation tuning is to maximize the performance of the loop once all other parameters, such as output capacitance, switching frequency, input and output voltage, and board layout, are fixed. Typically, the requirement for an acceptable design is to meet some given transient specification requiring the overshoot and undershoot amplitudes to be below some threshold while maintaining phase margin above some threshold (usually 60 degrees) to ensure stability. Overshoot and undershoot are defined as maximum deviation from the steady state value during a step in the load, and may be evaluated by direct observation of the load step response. However, evaluating the stability criteria requires knowing the Bode plot of the loop gain, which is conventionally obtained via an invasive and complex measurement.

Depending on the system, it may be the case that adjusting the compensation network alone is insufficient for meeting the transient performance specification while maintaining sufficient phase margin. This occurs often in practice when one is trying to reach the performance target with the lowest area or lowest cost solution, for example by minimizing the output capacitance in the design. Therefore, it is common for power engineers to have to repeatedly optimize the compensation network each time they change any other parameter in the design.

Normally, compensation tuning is achieved through incremental adjustments to the component values, with transient response and Bode plot measurements in be-
between changes guiding the direction of adjustment. Due to the slowness of Bode plot measurements, the repeated frequency measurements become a bottleneck in the procedure. Furthermore, determining which parameters to change and in what direction based on the transient response and Bode plot is a nontrivial task which requires understanding of the system’s underlying control theory. Thus, automating the compensation tuning and cutting out the repeated frequency analyzer measurements is desirable since this would simplify and speed up the applications process for power supplies, reducing both the delay of the critical path and the demand on the expertise of the engineer.

4.2.2 Implementation

Our goal in introducing BodeNet to the loop optimization task is to achieve the simplified workflow depicted in Figure 4-2, in which the compensation network is automatically tuned without requiring frequency measurements or user input.

This section will detail our approach to creating a proof-of-concept demonstration for this new procedure using the LTM4700. We showed in Chapter 3 that we achieved a model which could accurately infer the loop gain frequency response from the transient response. Here, we will describe an algorithm which uses the inferred loop gain Bode plot to determine how the compensation parameters should be adjusted.
First, we define the optimization problem which this algorithm attempts to solve as the following:

\[ G_{opt} = \arg \max_{g \in G} \left[ g.\text{bandwidth} \times 1(g.\text{phase\_margin} > 60) \right], \quad (4.1) \]

where \( G \) is the space of loop gains with varied compensation parameters and \( g.\text{bandwidth} \) and \( g.\text{phase\_margin} \) are the bandwidth and phase margin of some loop gain \( g \) in \( G \). Since higher bandwidth generally corresponds to faster transient response and therefore lower undershoot or overshoot, this definition of optimal is consistent with the requirements described in Section 4.2.1.

For the LTM4700, which uses a Type II compensation network, the compensation parameters are \( g_m, R_{th}, C_{th}, \) and \( C_{thp} \). For the purposes of our demo, we disregard \( C_{thp} \), so the space we must optimize over is the set of valid \( g_m, R_{th}, \) and \( C_{th} \) values.

Figure 4-3 shows a plot of an example optimization space on phase margin and bandwidth axes. Each point represents a unique combination of compensation parameter values. The optimal configuration, which has the highest bandwidth among configuration with phase margin of at least 60 degrees, is marked with a star. A successful optimization algorithm would be able to start from any of the non-optimal points and end at or near the optimal point.

From a given starting loop gain, we can predict the loop gain that would result from changing to a different set of compensation parameters by using the analytical model of the Type II compensation network. The compensation network has the following gain (See Figure 2-2 for a circuit diagram):

\[ G_{\text{comp}}(s) = g_m R_0 \frac{1 + sR_{th}C_{th}}{(1 + sR_0C_{th})(1 + sR_{th}C_{thp})}; C_{thp} << C_{th}, \quad (4.2) \]

where \( R_0 \) is the output resistance of the current source \( g_m \) amplifier [19]. We have a zero at \( \frac{1}{R_0C_{th}} \) and two poles at \( \frac{1}{R_0C_{th}} \) and \( \frac{1}{R_{th}C_{thp}} \). Recalling from Equation 1.1 that the loop gain contains \( G_{\text{comp}} \) as a product term, this yields that the impact of changing compensation parameters is to shift the locations of a zero and two poles in the overall loop gain function.
Thus, we can write the following formula for predicting a loop gain with compensation values \((g_{m1}, R_{th1}, C_{th1})\) from a known loop gain with compensation values \((g_{m0}, R_{th0}, C_{th0})\):

\[
G_1(s) = G_0(s) \frac{g_{m1}}{g_{m0}} \frac{1 + sR_{th1}C_{th1}}{(1 + sR_{th0}C_{th0})(1 + sR_{th1}C_{thp})} \frac{(1 + sR_{th0}C_{th0})(1 + sR_{th0}C_{thp})}{1 + sR_{th0}C_{th0}},
\]

(4.3)

where \(C_{thp}\) is 22 pF and \(R_0\) is estimated to be 10 MΩ. Experimentation showed that results were not sensitive to reasonable variation in \(R_0\).

An example of this prediction is shown in Figure 4-4. The loop gain measured in blue is used as a starting point, and Equation 4.3 is applied to yield the predicted curve in orange. The actual measured loop gain is included for comparison. While the prediction is close to the actual gain, there is still some error, most likely due to mismatches in parameter values and the first-order nature of the analytical model.

A naive approach to performing optimization would be to apply Equation 4.3 to the initial loop gain for every possible combination of compensation parameters, and
then select the compensation with the most optimal loop gain from the predicted set. However, due to prediction error, actual loop gain of this first guess may not match the analytical prediction and the point may not actually be optimal. A natural improvement from this observation is to check the actual loop gain at this predicted optimal point and re-run the search if the measured loop gain is much different from the analytical prediction. Therefore, it is advantageous to perform multiple iterations, measuring the transient response after setting parameters to the predicted optimal values and using using the inferred loop gain to re-predict the optimal until convergence is reached. We define the configurations "visited" by the algorithm as the intermediate predicted optimal points; the hardware is only physically adjusted to this subset configurations during the search.

The algorithm for optimization is summarized below:

```python
visited = list()

while True:
    hardware.program_compensation(start_config)
    transient = hardware.measure_transient()
    loop_gain = bodenet_model.infer(transient)
    predicted_bandwidths = dict()
    for every other_config not in visited:
        # Code to predict bandwidths...
```

Figure 4-4: Example result of predicting loop gain with \((\text{gm}, \text{Rth}, \text{Cth}) = (3.69, 1.5, 3300)\) (orange) from measured loop gain with \((\text{gm}, \text{Rth}, \text{Cth}) = (2.35, 6, 22000)\) (blue) with \(\text{Cout}\) and \(\text{Fsw}\) fixed.
(a) Plot of optimization paths for 10 randomly chosen starting configurations.

(b) Transient responses of the two optimized endpoints.

Figure 4-5: Results of simulating optimization algorithm on a dataset with $C_{out} = 1980uF$ ceramic + 4x2R5TPE470M7 and $F_{sw} = 425 \text{ kHz}$.

```python
    g = predict_gain(loop_gain, from=start_config, to=other_config)
    if g.phase_margin > 60:
        predicted_bandwidths[other_config] = g.bandwidth
    best_config = argmax(predicted_bandwidths)
    if best_config == start_config:
        return
    start_config = best_config
```

The auto-tuning algorithm was simulated using a pre-collected dataset to provide a ground truth reference for the optimization results. The dataset was collected using the same standard procedure as the Tier 1 testing dataset except that input and output voltages were fixed at 12V and 1V. When running the algorithm, the only difference is that transient responses are looked up from the dataset each time a hardware measurement would be performed. Since we already had ground truth Bode plots for this dataset, we can track the configurations visited by the algorithm accord-
Figure 4-6: Example results of auto-tuning demo running on live hardware.

We also implemented a demonstration of the algorithm on live hardware. Since the algorithm only requires fast transient measurements, the total optimization requires less than a minute. Similarly to in simulation, the algorithm was able converge to a stable, high-bandwidth solution within a few iterations from any random starting point. Figure 4-6 illustrates one example of the demonstration, where a configuration with a oscillatory transient is automatically tuned to become fast and stable.

4.2.3 Discussion

In this section, we have shown how BodeNet can be leveraged in two example applications which provide user direct benefits. First, we described a live Bode plot monitoring application, which enables continuous and responsive frequency-domain insights to aid power engineers during design and troubleshooting. Second, we de-
scribed our implementation of an algorithm which quickly and automatically optimizes loop compensation parameters. Our simple algorithm demonstrates how this ML-powered auto-tuning feature has the potential to significantly streamline what is otherwise a painstaking and manual procedure in power supply implementation, not only saving time but making power design much more accessible as well. Future work on automatic compensation tuning may investigate more sophisticated algorithms for predicting optimal values, especially for control topologies where analytical models of the compensation may be less well-known.
Chapter 5

Conclusion

The aim of this thesis was to develop a novel method for identifying power supply loop gain from a measured load step transient response. We presented a encoder/decoder-based neural network structure and an efficient workflow for data collection and training which result in accurate inferences across a wide variety of configurations for the LTM4700 buck converter. The ML approach bypasses the cumbersome and invasive frequency analyzer measurements normally required to characterize loop gains in physical hardware while drastically surpassing the accuracy of analytical models. Finally, we showed how this innovation enables disruptive automation and streamlining of important tasks, such as loop compensation tuning, in the power industry.

5.1 Future Extensions

This promising proof-of-concept work on the LTM4700 μModule lay a strong foundation for extending the ML approach to other power products. Our results also suggest that it would be worthwhile to pursue further experimentation on the model’s ability to generalize over multi-phase converters, different board layouts, and transients with varying slew rates. An interesting direction for future research would be to apply the methods developed in this thesis to modeling converters with newer control loop schemes such as V^2 control. For such architectures where the literature of analytical models is less well-developed, a data-driven approach would be even more valuable.
Appendix A

Daughterboard PCB Details

The daughterboard is a 2-layer PCB designed to provide a variable capacitance at the C+ and C- terminals. It is populated with 8 ceramic chip capacitors (C1-C8) of various values which are connected or disconnected from the output by SPST analog switches (U1-U8). The control signals for the switches (IN1-IN8), along with 3.3V power and ground, are supplied from a 10 pin connector (J1).

The schematic, renders, PCB layout, and bill of materials for the daughterboard are included below.

![Complete schematic of daughterboard.](image)

Figure A-1: Complete schematic of daughterboard.
Figure A-2: Top and bottom renders of daughterboard PCB.

Figure A-3: Top and bottom daughterboard PCB layers.
<table>
<thead>
<tr>
<th>Designator</th>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>06035C102JAT2A</td>
<td>0603 1000pF</td>
</tr>
<tr>
<td>C2</td>
<td>06035C222JAT2A</td>
<td>0603 2200pF</td>
</tr>
<tr>
<td>C3</td>
<td>06035C332JAT2A</td>
<td>0603 3300pF</td>
</tr>
<tr>
<td>C4</td>
<td>06035C472JAT2A</td>
<td>0603 4700pF</td>
</tr>
<tr>
<td>C5</td>
<td>06035C682JAT2A</td>
<td>0603 6800pF</td>
</tr>
<tr>
<td>C6</td>
<td>06035C103JAT2A</td>
<td>0603 10000pF</td>
</tr>
<tr>
<td>C7</td>
<td>06035C153JAT2A</td>
<td>0603 15000pF</td>
</tr>
<tr>
<td>C8</td>
<td>06035C223JAT2A</td>
<td>0603 22000pF</td>
</tr>
<tr>
<td>C9-C15</td>
<td>06035C104KAT2A</td>
<td>0603 0.1uF</td>
</tr>
<tr>
<td>J1</td>
<td></td>
<td>RA Header 10Pos 2.54mm</td>
</tr>
<tr>
<td>U1-U8</td>
<td>ADG841YKS</td>
<td>0.28 Ohm CMOS SPST Switch</td>
</tr>
</tbody>
</table>

Table A.1: Bill of materials for daughterboard components.
Appendix B

Preprocessing Code

This appendix contains Python implementations of the data preprocessing described in Section 3.2. Two functions, along with the imported libraries used, are presented: `process_transient` processes input transients, while `process_bode` processes ground truth gain and phase plots.

```python
import math
import scipy.signal as sp
from scipy.interpolate import interp1d
import numpy as np
import torch

def process_transient(x, train=True, decimate_factor=8, random_crop_frac=0.2, random_vshift=0.005):
    """Process transient waveform into BodeNet input
    Args:
    x (torch.Tensor): Tensor of measured transient voltage waveforms, shape (N, 8192)
    train (bool): True to randomize cropping and shifting during training
    decimate_factor (int): Downsampling factor for decimation
    random_crop_frac (float): fraction of cropping to randomize during training
    random_vshift (float): if train, randomly shifts voltage by +/ random_vshift"
```
# Decimate

decimate_filter = sp.firwin(
    21, 1 / (2 * decimate_factor), window="hamming", pass_zero=True
)

x = sp.filtfilt(decimate_filter, [1.0], x.numpy())
x = x[:, ::decimate_factor]
x = torch.from_numpy(x.copy()).float()

# Crop to include only rising edge of transient
x = x[:, 100 : x.shape[-1] // 2 + 70]  # hard-coded crop window
L_crop = math.floor(x.shape[-1] * random_crop_frac)  #=0.2
L_stay = x.shape[-1] - L_crop
if train:
    # Randomly shift window during training
    i_crop = np.random.randint(L_crop)
    x = x[:, i_crop : i_crop + L_stay]
else:
    # Crop from center during testing
    L_crop_left = L_crop // 2
    L_crop_right = L_crop - L_crop_left
    x = x[:, L_crop_left: -L_crop_right]

# Standardize
x = x - torch.mean(x)

if train:
    # Add random vertical shift during training
    vshift = np.random.random() * 2 * random_vshift - random_vshift
    x = x + vshift

return x
def process_bode(freq, y, freq_targ=np.linspace(2, 5.3, 128, endpoint=True), savgol_win=11):
    
    """Preprocess measured gain/phase plots for use as ground truth BodeNet output
    
    Args:
        freq (torch.Tensor): Original measured frequencies, in Hz
        y (torch.Tensor): Original measured gain or phase
        freq_targ (np.ndarray): Target frequency array in log scale
        savgol_win (int): Savgol filter window size"
    
    freq = freq.squeeze()
    y = y.squeeze()

    # Convert freq to log scale
    freq = np.log10(freq)

    # Interpolate values for target frequency points
    f_intp = interp1d(freq, y, fill_value="extrapolate")
    y = f_intp(freq_targ)

    # Smooth
    if savgol_win > 0:
        y = sp.savgol_filter(y, savgol_win, 3)

    y = torch.tensor(y, dtype=torch.float32).unsqueeze(0)

    return y
Bibliography


