Alexander Kropiunig, Bsc

Intelligent post-processing of quasi-stationary recorder data

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dr.techn. MSc Eduardo Enrique Veas

Co- Supervisor

Dipl.-Ing. Dr.techn. Granit Luzhnica

Institute of Interactive Systems and Data Science

Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Frank Kappe

Graz, Aug 2024

Abstract

Time delay compensation is a critical challenge in powertrain calibration optimization using slow dynamic slope (SDS) data. SDS data involves recording ramps between design of experiment (DoE) points, providing more detailed system behavior than steady-state measurements. However, measured signals can exhibit time delays from sources like gas transport times, thermal inertia, and measurement system latency. Properly aligning the time-shifted signals is essential for building accurate models from the SDS data. This thesis explores different techniques for automatically aligning and synchronizing time-delayed SDS data for powertrain calibration. An automated script is developed to facilitate efficient post-processing of the data.

Contents

List of Figures

List of Tables

1. Introduction

Design of Experiments (DOE) is a widely used method for optimizing high dimensional systems in various fields, including powertrain calibration. DOE involves the pre-planning of a list of experiments that are distributed in the n-dimensional variation space by a certain criterion, such as D-optimal or V-optimal. These experiments are then carried out on a test bed, and mathematical models are developed based on the resulting steady-state measurements. However, the conventional DoE approach has several limitations, including the need for a large number of experiments to adequately cover the variation space and the assumption of steady-state conditions, which may not always be feasible or accurate.

To overcome these limitations, Slow Dynamic Slope (SDS) DOE is used. In this approach, each target DOE point is approached in slow ramps, and the ramps between the points are recorded as training data for the models. These recorded ramps are known as slow dynamic slopes (SDS), and they provide a more detailed representation of the system behavior than steady-state measurements alone. The use of SDS data enables the development of more accurate and efficient models for powertrain calibration optimization, as it allows the reduction of experimental runs.

However, the use of SDS data also brings up several challenges that need to be addressed. One of the main challenges is time delay compensation. Many real measurement signals for performance and emissions, such as those used in powertrain calibration, are measured with a certain delay. Typical delay times for emission measurement devices are in the range of a few seconds. In addition, it is often necessary to use different systems to record the data, which can lead to the need for recorder synchronization. To effectively align and synchronize the recorded SDS data, different techniques must be investigated and evaluated in terms of efficiency, practicability, and feasibility.

In this master's thesis, we will explore several approaches to time alignment for SDS data, including cross-correlation, phase-based alignment, and Kalman filtering. We will develop an automated script to facilitate the post-processing of SDS data and evaluate the performance of the different techniques using experimental data. The results of our work will provide insights into the most effective approaches for intelligent post-processing of SDS data in the context of powertrain calibration optimization.

2. Problem statement and research question

2.1. Background

Design of Experiments (DOE) is a widely used method for optimizing the steady state verification and calibration of powertrains. In conventional DOE, a pre-planned list of experiments is carried out, and the results are used to build mathematical models that describe the relationships between different input variables and output responses. The experiments are typically distributed in the n-dimensional variation space using a certain criterion, such as D-optimal, V-optimal, or space-filling.

While DoE has been an established method for decades, it has some limitations when it comes to optimizing powertrain calibration. One of these limitations is that it relies on steady state measurements, which may not fully capture the behavior of the system. This can lead to models that are less accurate or less representative of the system's behavior under different operating conditions.

To address this issue, a new approach called slow dynamic slopes (SDS) has been developed. In SDS, each target DoE point is approached in slow ramps, and the ramps between the points are recorded. The recorded data is then used as training data for the models. This approach has the potential to generate more accurate and representative models for optimizing powertrain configuration and calibration compared to conventional DOE.

However, using SDS data for model building also brings up several challenges that need to be addressed. One of these challenges is time delay compensation. Many real measurement signals for performance and emissions are measured with a certain delay, which can be a few seconds or more. Additionally, it is common to use different systems to record the data, such as test bed system measurement devices or onboard diagnostics (OBD) systems. This leads to the need for synchronization of the recorded data from different systems.

Time alignment and synchronization is essential for building accurate and representative models from SDS data. Without proper time alignment and synchronization, the recorded data may not be properly correlated, leading to incorrect or biased models. Therefore, finding efficient, practical and feasible techniques for aligning and synchronizing recorded data is crucial for optimizing powertrain calibration using SDS data.

2.2. Problem statement

The problem that this research aims to address is how to efficiently, practically and feasibly align and synchronize recorded data from different systems in order to build models for optimizing powertrain calibration using SDS data.

2.3. Research question

The research question for this study is: How can we align and synchronize recorded data from different systems in order to build models for optimizing powertrain calibration using SDS data, in a way that is efficient, practical and feasible?

2.4. Objectives

The specific objectives of this study are:

- Objective 1: To review the existing techniques for time alignment and synchronization of recorded data and identify their advantages and disadvantages.
- Objective 2: To evaluate a range of techniques and algorithms for time alignment and synchronization of recorded data using defined metrics for efficiency, practicability, and feasibility.
- Objective 3: To compare the results of the different techniques and algorithms to identify the most promising ones for aligning and synchronizing recorded data from different systems.
- Objective 4: To develop an automated script for post-processing of recorder data, which includes time alignment and synchronization as well as any other necessary processing steps.
- Objective 5: To apply the selected time alignment and synchronization technique(s) to real recorded data from different systems and assess the accuracy and reliability of the resulting models for optimizing powertrain calibration.

2.5. Motivation

The optimization of powertrain systems is a complex task that requires the consideration of multiple variables and performance criteria. Design of Experiments (DOE) has been widely used as a method for optimizing powertrain systems, particularly for steady-state conditions. However, the conventional DOE approach has several limitations, including the need for a large number of experiments to adequately cover the variation space and the assumption of steady-state conditions, which may not always be feasible or accurate.

To overcome these limitations, a new approach has been developed in recent years. This approach involves the use of slow dynamic slopes (SDS) as training data for the development of mathematical models for powertrain calibration optimization. SDS data provides a more detailed representation of the system behavior than steady state measurements alone, is captures way much more data in the DOE process. The use of SDS data also enables the reduction of experimental runs and the development of more accurate and efficient models for powertrain calibration optimization.

This study has been has been developed and evaluated at AVL List GmbH.

2.6. AVL List GmbH

AVL List GmbH is a world-renowned company that specialized in the development of advanced powertrain systems for the automotive industry. The company was founded in 1948 by Dr. Hans List and has since become a global leader in the development of internal combustion engines, electric drives, and fuel cell systems.

AVL List GmbH employs over 9,000 people worldwide and has a network of more than 100 locations across the globe. The company's headquarters is located in Graz, Austria, and it has a strong presence in North America, Europe, and Asia.

AVL List GmbH's research and development efforts are focused on the development of advanced powertrain technologies that are more efficient, cleaner, and safer than traditional systems. The company works closely with automakers and other industry partners to develop and commercialize these technologies, which include hybrid electric vehicles, plug-in hybrid electric vehicles, and fuel cell vehicles.

In addition to its research and development activities, AVL List GmbH provides a range of services to its customers, including engineering consulting, testing and validation, and project management. The company also has a strong focus on sustainability and is committed to reducing its environmental impact through the use of renewable energy and other eco-friendly initiatives.

Overall, AVL List GmbH is a leading provider of advanced powertrain systems for the automotive industry, with a strong commitment to research and development, sustainability, and customer service. The company's expertise and innovative technologies are helping to drive the transition to a more sustainable future for the automotive industry (List, 2021).

2.7. DOE

The use of DOE in the automotive industry has a long history, with a number of studies demonstrating the effectiveness of this approach in optimizing various aspects of vehicle design and manufacturing. For example, DOE has been used to optimize the design of engine components such as combustion chambers and valve trains. In addition, DOE has been used to identify the optimal combination of process parameters for various manufacturing processes in the automotive industry, such as casting and forging.

One of the main advantages of DOE in the automotive industry is its ability to efficiently identify the factors that have the greatest impact on a particular response, such as fuel efficiency or emission levels. By carefully controlling the levels of these factors in a structured experimental design, it is possible to determine the optimal combination of factors for a desired outcome, such as maximum fuel efficiency or minimum emissions. In addition, DOE allows for the identification of interactions between factors, which can be important in understanding the underlying mechanisms of a process or system.

There are several different DOE approaches that have been applied in the automotive industry, including full factorial design, fractional factorial design, and response surface methodology (RSM). Each of these approaches has its own strengths and limitations, and the choice of approach will depend on the specific goals and constraints of the study.

It is worth noting that the use of DOE in the automotive industry is not without challenges. For example, the number of factors that can potentially influence a response in the automotive industry is often large, making it difficult to include all relevant factors in a single study. In addition, the complexity of automotive systems can make it difficult to accurately model the relationships between factors and responses. Despite these challenges, DOE remains a valuable tool for optimizing various aspects of vehicle design and manufacturing in the automotive industry.

2.8. SDS-DOE for Powertrain Calibration

Design of Experiments (DOE) is a widely used method for optimizing high dimensional systems in various fields, including powertrain calibration. DOE involves the pre-planning of a list of experiments that are distributed in the n-dimensional variation space by a certain criterion, such as D-optimal or V-optimal. These experiments are then carried out on a test bed or in simulation, and mathematical models are developed based on the resulting steady-state measurements. However, the conventional DOE approach has several limitations, including the need for a large number of experiments to adequately cover the variation space and the assumption of steady-state conditions, which may not always be feasible or accurate.

2.8.1. SDS-DOE for Calibration

To overcome these limitations, a new approach called SDS has been developed in recent years. In this approach, each target DOE point is approached in slow ramps, and the ramps between the points are recorded as training data for the models. These recorded ramps are known as slow dynamic slopes (SDS), and they provide a more detailed representation of the system behavior than steady-state measurements alone. The use of SDS data enables the development of more accurate and efficient models for powertrain calibration optimization.

However, the use of SDS data also brings up several challenges that need to be addressed. One of the main challenges is time delay compensation. Many real measurement signals for performance and emissions, such as those used in powertrain calibration, are measured with a certain delay. Typical delay times for emission measurement devices, for example, can range from a few seconds up to a minute. In addition, it is often necessary to use different systems to record the data, which can lead to the need for recorder synchronization. To effectively align and synchronize the recorded SDS data, different techniques must be investigated and evaluated in terms of efficiency, practicability, and feasibility.

2.8.2. SDS-DOE in the Automotive Sector

The use of SDS-DOE for powertrain calibration is particularly relevant in the automotive sector, where the optimization of fuel efficiency and emissions is a critical concern. The development of accurate and efficient models for powertrain calibration using SDS data can help to reduce fuel consumption and emissions, leading to both environmental and economic benefits. In addition, the use of SDS data can enable more rapid and efficient development of new powertrain technologies, such as electric and hybrid systems, which are becoming increasingly important in the automotive industry.

2.9. Sources of Time Delay in Signals on Test Beds: Causes and Characterization

Time delay in signals on automotive test beds can have a significant impact on the accuracy and reliability of the measured data. In this report, we will consider potential sources of delay and discuss appropriate mitigation strategies.

Time delays in signals on automotive test beds can occur for a number of reasons.

2.9.1. Gas Transport

When operating an internal combustion engine, combustion occurs within the cylinder, producing exhaust gases. These exhaust gases, including NOX and CO emissions, are expelled from the cylinder and travel through the exhaust system. Along this path, an extraction point is designated where a measuring device collects a sample of the exhaust gases. This sample is then analyzed to determine the concentration of NOX and CO emissions.

Direct measurement of emissions at the point of origin within the engine cylinder is hardly feasible since a hole has to be drilled and the sensing system has to be made tight and immune to extremely high temperatures. Instead, the sampling occurs a few centimeters downstream in the exhaust system. The exhaust system typically includes the turbocharger, exhaust manifold, and pipes leading to the catalytic converter. The extraction point for sampling is usually located between the turbocharger and the catalytic converter.

The NOX emissions must travel from the engine's interior to the extraction point, and from there, they continue through a sampling tube to the analyzer, which can be up to 10 meters away. This journey causes a time delay, averaging 4-5 seconds, in the NOX measurement signal due to the distance the gas must travel. Although the volumetric flow rate within the sampling tube is constant, ensuring consistent transport time from the extraction point to the analyzer, the travel time from the engine to the extraction point can vary. This variability depends on engine speed, air throughput, and load conditions, potentially causing delays ranging from 0.1 to 0.2 seconds.

2.9.2. Thermal Inertia of the System

When driving a vehicle, especially under conditions such as climbing a hill at full throttle, the exhaust system does not instantly reach high temperatures. The thermal inertia of the exhaust system, including the catalytic converter, results in a delay before the system stabilizes at the operational temperature necessary for accurate emission measurements. This thermal inertia causes a time delay in the response of the catalytic converter and other components in the exhaust system to changes in engine load and operating conditions. As the system gradually heats up, the time delay decreases, but during initial periods of heavy load, the delay can significantly impact the accuracy of emission measurements.

2.9.3. Measurement System Latency

One common cause of time delay in signals on automotive test beds is the time it takes for a signal to travel through the various sensors, wiring, and electronic components that make up the test bed's measurement system. This type of delay, known as measurement system latency, can be particularly significant in high-speed or high-frequency measurements, as well as in systems with long signal paths or many intermediate signal processing steps.

2.9.4. Software-based Signal Processing

Another potential cause of time delay in signals on automotive test beds is the use of software-based signal processing, such as filtering or averaging. These types of operations can introduce additional delay, as the computer or processor performing the calculations must complete them before the final processed signal can be output.

2.9.5. Design of the Test Bed

Time delay can also be introduced due to the way the test bed is designed. For example, a test bed that uses a wired connection between the vehicle and the measurement system may introduce additional delay, as the signals must travel through the cable. Similarly, a test bed that uses wireless transmission of signals may introduce delay due to the inherent limitations of wireless communication.

2.9.6. Multiplexers and Signal Routing Components

Another possible cause of time delay is the use of multiplexers and other signal-routing components in the measurement system. These devices allow multiple signals to be routed through a single channel, but they can introduce additional delay as the signals are switched between channels.

2.9.7. Multiple Layers of Signal Conditioning or Amplification

Another potential cause of time delay is the use of multiple layers of signal conditioning or amplification, such as amplifiers and filters. Each layer of signal conditioning can introduce additional delay, which can accumulate as the signal travels through the measurement system.

2.9.8. External Triggering

Another cause could be the use of external triggering, which can introduce time delay because the signal of interest is not acquired until a specific trigger event occurs. For example, an engine RPM (revolutions per minute) signal would not be acquired until the engine reaches the RPM set by a trigger.

2.9.9. Resolution of the Measurement System

The resolution of the measurement system can also affect time delay. A system with lower resolution may require more processing time to acquire and digitize the signal, thus introducing additional delay.

2.9.10. Environmental Factors

Environmental factors such as temperature, humidity, and electromagnetic interference can also affect the time delay of signals on automotive test beds. For example, temperature changes can cause changes in the electrical properties of wires and other components, resulting in changes in the time delay of signals.

2.9.11. Mitigation Strategies

To minimize the time delay in signals on automotive test beds, it is crucial to use high-quality, low-latency components and to design the test bed with attention to signal path and signal routing. Additionally, proper calibration of the measurement system and proper usage of the test bed can also help to minimize time delay.

It is also important to note that even with all these efforts the time delay cannot be completely eliminated and the test bed should be designed to handle the time delay and allow for its measurement and compensation.

2.10. Modeling

Modeling is an essential aspect of many industries, including the automotive industry. It involves the creation of a representation or simulation of a system or process, which can be used to predict the behavior of the system or process under various conditions. The use of modeling in the automotive industry can be traced back to the early days of the industry when engineers used physical models to test and refine their designs. Today, the use of computer-based modeling has become widespread, enabling engineers to quickly and accurately test and optimize their designs.

2.11. Types of Modeling in the Automotive Industry

There are various types of modeling that are commonly used in the automotive industry, including structural modeling, kinematic modeling, and dynamic modeling.

2.11.1. Structural Modeling

Structural modeling involves the creation of a model that represents the physical structure of a vehicle or component. This can include the dimensions, materials, and geometry of the structure. Structural models are used to analyze the strength and stiffness of a design, as well as to identify potential failure points.

2.11.2. Kinematic Modeling

Kinematic modeling involves the creation of a model that represents the movement of a vehicle or component. This can include the position, velocity, and acceleration of the moving parts. Kinematic models are used to analyze the performance of a design, such as the handling and ride comfort of a vehicle.

2.11.3. Dynamic Modeling

Dynamic modeling involves the creation of a model that represents the interaction of a vehicle or component with its environment. This can include the forces acting on the vehicle, such as gravity, aerodynamics, tire forces and road conditions. Dynamic models are used to analyze the performance of a design, such as the fuel efficiency and acceleration of a vehicle.

2.12. Modeling in the Context of Automotive Calibration

Automotive calibration refers to the process of adjusting the parameters of a vehicle's control systems to optimize its performance. This can include adjusting the fuel-to-air ratio of an engine, the damping characteristics of a suspension, or the steering ratio of a steering system. The use of modeling can be especially useful in the context of automotive calibration, as it enables engineers to analyze and optimize the performance of the control systems under various conditions.

One example of the use of modeling in automotive calibration is the optimization of the fuel-to-air ratio in an internal combustion engine. By creating a model of the engine, engineers can analyze the effects of different fuel-to-air ratios on the performance of the engine, such as the power output, fuel efficiency, and emissions. This can enable them to identify the optimal fuel-to-air ratio for a given set of operating conditions, improving the performance and efficiency of the engine.

Another example of the use of modeling in automotive calibration is the optimization of the suspension characteristics of a vehicle. By creating a model of the suspension, engineers can analyze the effects of different damping coefficients on the ride comfort and handling of the vehicle. This can enable them to identify the optimal damping coefficients for a given set of operating conditions, improving the ride comfort and handling of the vehicle.

Overall, the use of modeling in the context of automotive calibration offers a powerful tool for optimizing the performance of a vehicle's control systems. By analyzing and optimizing the parameters of the control systems, engineers can improve the performance, efficiency, and comfort of a vehicle.

2.13. Benefits of Modeling in the Automotive Industry

The use of modeling in the automotive industry has numerous benefits, including:

- **Reduced Development Time and Costs:** By using computer-based modeling, engineers can quickly and accurately test and optimize their designs, reducing the time and cost required to bring a new product to market.
- **Improved Performance:** By using modeling to analyze and optimize the performance of a design, engineers can improve the fuel efficiency, acceleration, handling, and other performance characteristics of a vehicle.
- **Increased Safety:** By using modeling to analyze the structural and dynamic performance of a design, engineers can identify and address potential failure points, improving the safety of the vehicle.
- **Reduced Prototyping:** By using modeling to test and refine a design, engineers can reduce the number of physical prototypes required, saving

time and resources.

2.14. Challenges of Modeling in the Automotive Industry

While modeling has many benefits in the automotive industry, it also has its challenges. Some of the key challenges include:

- **Complexity:** The automotive industry involves a wide range of systems and processes, which can be complex to model accurately.
- **Accuracy:** Modeling results can be affected by various factors, such as the quality of the data used, the assumptions made, and the limitations of the model itself. As a result, it is important to carefully validate and verify the accuracy of the model.
- **Cost:** The development of accurate and sophisticated models can be time-consuming and costly, requiring specialized software and expertise.

Despite these challenges, the use of modeling in the automotive industry continues to grow, as it offers a powerful tool for improving the design, performance, and safety of vehicles.

2.15. Calibration

The automotive industry is highly regulated, with strict standards for vehicle performance, safety, and emissions. To meet these standards, manufacturers must carefully design and test their vehicles, ensuring that all components function properly. Calibration plays a critical role in this process, particularly in optimizing vehicle components such as engines and suspension systems.

2.16. Role of Calibration in the Automotive Industry

Calibration is used throughout the automotive industry, from the design and testing phase to the production and maintenance of vehicles. During the design and testing phase, calibration involves tuning specific parts of the vehicle to achieve optimal performance. For example:

- **Engine Calibration**: Adjusting the engine parameters (e.g., fuel injection timing, air-fuel mixture, ignition timing) to achieve the best performance, fuel efficiency, and emissions control.
- **Suspension Calibration**: Fine-tuning the suspension system to improve handling, ride comfort, and stability. This can involve adjusting the shock absorbers, springs, and other components to suit specific driving conditions or preferences.
- **Transmission Calibration**: Optimizing gear shift patterns and clutch engagement for smooth and efficient power delivery.
- **Brake System Calibration**: Ensuring that brake components function correctly and that the braking force distribution is optimized for safety and performance.

In production, calibration ensures that manufactured components meet specified tolerances and function optimally. Proper calibration of the engine control systems and suspension components is essential for the vehicle's performance and compliance with regulatory standards. Calibration remains crucial in vehicle maintenance, ensuring that performance tuning and adjustments are accurate and reliable.

2.17. Current Calibration Methods

There are several methods currently used for calibrating vehicle components in the automotive industry. These include:

- **Software Algorithms**: For engine control systems, calibration often involves using software algorithms to adjust performance parameters based on collected data to ensure optimal operation.
- **Mechanical Adjustments**: For suspension systems, calibration involves mechanical adjustments and testing under various conditions to fine-tune the system.
- **Diagnostic Tools and Equipment**: Advanced diagnostic tools and equipment are employed to assist in the calibration process, ensuring precision and adherence to standards.
- **Real-world Testing**: Performing tests in real driving conditions to calibrate components like the transmission and brake systems, ensuring they perform optimally in diverse scenarios.

This comprehensive approach to calibration helps maintain vehicle safety, performance, and regulatory compliance.

2.18. Current State of SDS DOE data Calibration

Currently, the process of data calibration involves manually shifting data to identify the most accurate state for calibration. This method is not only time-

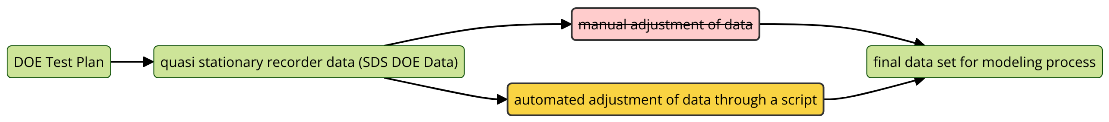
DOE Test Plan
quasi stationary recorder data (SDS DOE Data)
~~manual adjustment of data~~
automated adjustment of data through a script
final data set for modeling process

Figure 2.1.: The Current State vs. How It Should Be in the Future: Solutions Proposed in This Work.

consuming but also prone to inaccuracies, which are detrimental to the efficiency and reliability of the calibration process. The labor-intensive nature of this manual intervention adds significant delays to projects, increasing overall timelines and potentially leading to higher costs and reduced effectiveness.

In response to these challenges, this work proposes an innovative solution designed to automate the calibration process, thereby enhancing both its speed and accuracy. The core of this solution involves the development of an algorithm capable of automatically finding the optimal data delay. This algorithm leverages advanced computational techniques to rapidly and precisely adjust data, ensuring that the calibration is performed using the best possible data state without human intervention.

By automating the calibration process, the proposed solution aims to significantly reduce the time required for data preparation, while simultaneously increasing the accuracy of the calibration. This dual benefit not only streamlines the overall process but also improves the quality of the data output, which is crucial for subsequent analyses and applications.

3. Background and Related Work

3.1. Introduction

In the dynamic landscape of modern engineering and technology, the challenge of time delays in measurement signals emerges as a critical obstacle, affecting the stability and reliability of numerous systems. This chapter delves into the theoretical background and practical implications of these delays across various engineering fields. It begins by elucidating the inherent sources and types of time delays—from signal propagation and communication latency to sensor response times and computational delays. These delays not only compromise the accuracy of data analysis but also impact system operations in control plants, telecommunications, and more, potentially leading to significant financial and operational setbacks.

3.1.1. Theoretical Background

In modern systems, the accurate synchronization of measurement signals is critical for guaranteeing the stability and reliability of processes in various engineering domains. However, the emergence of inherent time delays in the measurement signals poses a substantial challenge (Fridman, 2014; Sipahi et al., 2012). These delays often originate from various sources such as signal propagation, communication latency, computational time, or the limitations of responses of sensors and actuators (Lakshmanan & Senthilkumar, 2011).

Time delays present challenges in various modern technological areas, including the reliability of control systems, signal processing, and real-time cyber-physical systems (Kawabata et al., 2017; Kopetz & Steiner, 2022). For instance, in the field of control plants, inherent time delays can lead to oscillation and even threaten system stability. In data analysis, the accuracy of results and predictions are posed critical risks when the data points are not properly determined, leading to unreliable outcomes, and financial losses. In the telecommunications sector, transmission delays in video conferencing and online games, lead to communication disruption and user dissatisfaction. These time delays introduce risks and safety concerns in various fields, including manufacturing processes, healthcare, and transformation systems. Addressing these challenges calls for the development of sophisticated compensation strategies to effectively reduce and manage the impact of these delays. Consequently, it is essential to address and remove these time delays for better functioning of current interconnected processes.

3.1.2. Time delays in measurement signals

The presence of inherent time delays in measurement signals is accompanied by many problems and various consequences across critical fields. In measurement signals, the time delay is characterized by the latency between the time a signal is measured and the moment that the measurement data is received (X. Li & Li, 2021; Zhong, 2006). The precise analysis of these delays in many domains such as control systems, signal processing, synchronization, and scientific issues is essential. Since time delays can occur due to various factors, analyzing and compensating for these delays is critical for enabling better risk management and effective decision-making.

The main cause of time delays in measurement signals is associated with the response time of actuators and measurement units (sensors). Sensors usually require a certain amount of time to detect and convert the signal into a physical phenomenon (e.g., temperature or pressure) embedded in an electrical signal. Consequently, this response time imposes complexities in system control design or analysis, especially where fast response is required in real-time applications.

Additionally, significant time delays arise from gas transport in the exhaust system of internal combustion engines. The NOx emissions must travel from the engine to the extraction point and then through a sampling tube to the analyzer, which can cause a delay of up to 4-5 seconds due to the distance and variable travel time depending on engine conditions.

Another major source of time delays is thermal inertia in the exhaust system. When the engine load changes, such as during heavy acceleration, the exhaust system, including the catalytic converter, takes time to reach the necessary operational temperature for accurate emission measurements. This thermal inertia can cause delays during the initial periods of heavy load until the system stabilizes at high temperatures.

Furthermore, time delays also occur during the signal transformation and data acquisition procedures. In modern technologies, the set of data is often captured through remote infrastructures and then transferred to a processing system. In this application, the transmitting time of data from the source to the receiver, including parameters such as transformation speed and data computation, impacts the overall time delay included in measurement signals. This transformation delay is essential in domains such as telecommunication, distributed energy systems, and remote measurements (Shangguan et al., 2020; Yan et al., 2019).

Moreover, time delays also occur during the sampling and data processing of measurements. There are some applications, such as data processing, communication systems, and control plant delays, that introduce interval delays between the samples. It is crucial to preserve the reliability of measurements to obtain accurate timing and synchronization for such processes. Under these circumstances, compensation for time delays provides the possibility to ensure the accuracy of decision-making and control systems.

3.1.3. The impacts of time delays in various applications

Time delays in measurement signals are a definite challenge that can bring a wide range of problems in various engineering applications. Control mechanisms, which have a critical role in the industry sector, are highly susceptible to the destructive effects of time delays. Such delays can result in instability and can even damage the equipment included in control systems (Shen et al., 2019). For instance, time delays in feedback measurements can lead to changes in the system operation and can reduce the system's safety. Moreover, time delays can decrease the efficiency of chemical procedures, which adversely affects quality and resource utilization (Bento, 2020; Niu et al., 2019).

Besides, cyber-physical networks have a high level of sensitivity to time delays in communication infrastructures. Any time delay in data transmission can lead to packet loss, which will cause re-transformation and network congestion. In telecommunications and the Internet of Things (IoT), accurate scheduling for data timing is required to guarantee reliable data exchange (Jafari & Rezvani, 2023; L. Zhao et al., 2019). The presence of time delays can disrupt the scalability of such systems, affecting user satisfaction and operational efficiency, e.g., throughput, workflow disruption, and many more.

The safety of healthcare systems depends on the timely and accurate knowledge of information. In remote monitoring, the timely transformation of patient information is necessary to provide reliable telemedicine services. The quality of healthcare treatment and diagnostic accuracy can be highly affected by time delays (Murni et al., 2021). Time lag can also affect the receipt and retrieval of information in the electronic health record (EHR), which disrupts information integrity. For disaster management, real-time detection is required to quickly respond to any changes and hazards in the environment. For example, timely processing is necessary to monitor and detect floods, windstorms, and wildfires to mitigate the destructive effects of natural disasters. The emergence of any time delays in the computational procedure and data transformation can disrupt the warning elements and efficiency of quick response to disasters (Khan et al., 2020).

Therefore, the deployment of interdisciplinary methodologies such as control theory, data science, signal processing, networked control, and telecommunication is required to address the issues associated with time delays. A wide class of solutions can be utilized such as model predictive techniques, network infrastructure optimization, parallel processing, real-time data coercion, and so on. The minimization and management of time delays in measurement signals is necessary to ensure the safety and reliability of processes across a wide spectrum of applications.

4. Literature Review

4.1. Strategies for Time Delay Mitigation

Extensive efforts have been made by contemporary researchers to eliminate the problems created by time delays to improve the system's performance. In the automotive sector, these delays are addressed to enhance the safety of autonomous vehicles due to minor time differences that can lead to accidents (Petrillo et al., 2018; Xu et al., 2020). Manufacturing industries depend on precise timing for quality control and efficiency, and compensating for time delays can lead to fewer defects and increased productivity (Abdellatif & Alshibani, 2019). Telecommunication and data centers require real-time synchronization to avoid network congestion and ensure uninterrupted data transfer (Sheykhi et al., 2022). These challenges underline the importance of developing effective time delay compensation methods to alleviate the problems inherent in measurement signals' time delays.

Multiple time delays can appear in complex manufacturing processes, where data is transformed from sensors to control systems, causing production inefficiencies. The multifaceted nature of the challenge posed by multiple time delays is evident in the cross-disciplinary applications of time delay compensation. The importance of addressing inherent time delays in measurement signals transcends individual domains and disciplines, impacting the core of modern technology, industry, and daily life. The complications posed by these delays elicit solutions that intersect the realms of science, engineering, and innovation, manifesting in substantial implications.

To address multiple time delays, researchers and engineers have explored an array of approaches. Up to now, various efforts have been made by contemporary researchers to mitigate the effect of multiple time delays using Generalized Predictive Control (GPC) (Pawlowski et al., 2016), model predictive control (MPC) (Lu et al., 2013; J. Zhang et al., 2019), linear matrix inequalities (LMI) (Fridman & Shaikhet, 2017; L. Xiong et al., 2018), event-triggered controllers (Borri & Pepe, 2020; A. Wang et al., 2017; Y. Wang et al., 2021), Smith predictor (Santos et al., 2014, 2016), sliding mode control (SMC) (Gao et al., 2020), AutoRegressive eXogenous (ARX) (Y. Zhao et al., 2016), Polynomial Regression etc.

Mobayen et al. (Mobayen et al., 2020) formulated an LMI-based global SMC scheme for uncertain discrete-time descriptor systems with multiple time-varying delays using Lyapunov-Krasovskii theory. One of the key outcomes of this research is the development of sufficient conditions for the asymptotic stability of the sliding mode dynamics, which is crucial for ensuring the system's robust and stable performance. The proposed global SMC approach not only addresses time delays but also deals with parametric uncertainties in the system. In (Azmi & Yazdizadeh, 2023), Azmi and Yazdizadeh proposed a fault-tolerant technique based on online adaptive tuning for nonlinear plants with multiple input and state delays. The research aims to bridge this gap by designing a control strategy that can handle the complexities arising from multiple delays and nonlinear dynamics.

The authors of (Zeng et al., 2019) focused on developing a novel mathematical tool, the Generalized Free-Matrix-Based Integral Inequality (GFMBII), to tackle time-varying delay systems in complex problems. The objective is not only to introduce a novel tool but also to show its practical applicability and superior performance in analyzing and ensuring the stability of time-varying delay systems.

In (Qin et al., 2018), the H-infinity synchronization dynamics in complex networks characterized by multiple time delays are explored. This paper employs advanced mathematical techniques, such as inequalities, and constructs appropriate Lyapunov functionals, enabling a rigorous analysis of synchronization stability.

The authors of (Wu et al., 2019) aimed to address the optimization of a specific class of nonlinear optimal control problems with multiple time delays, particularly when the control system is subject to equality/inequality constraints.

The research of (Fatehi & Huang, 2017) focuses on the application of a fusion Kalman filter to improve state estimation accuracy in scenarios where measurements from lab data are obtained at a slower rate and subject to variable delays and irregular sampling times. This work provides a robust methodology for state estimation in plants with delayed measurements, with a particular emphasis on leveraging lab data to enhance accuracy.

4.2. Challenges and advancements of state-of-the-art methodologies in time delay mitigation

Several shortcomings are associated with prevalent schemes designed to address systems with multiple time delays.

1. Limited Generalizability: Prevalent schemes are often tailored to specific system models or delay structures, making them less adaptable to a wide range of practical applications.

2. Complex Analysis: Some prevalent schemes involve complex mathematical analysis and intricate modeling of the system, which can make them challenging to implement, particularly in real-time applications. The complexity can also lead to difficulties in understanding and interpreting the results.

3. Intense computing: Prevalent schemes may require extensive computational resources, particularly for systems with multiple delays. This computational intensity can be a significant drawback, as it may not be feasible for systems that require real-time or resource-constrained operation.

4. Inadequate Consideration of Practical Constraints: Prevalent schemes may not adequately consider practical constraints that exist in real-world time-delayed systems which can result in sub-optimal or infeasible control strategies.

5. Complex Tuning: Tuning parameters for some schemes can be a time-consuming and challenging task. Achieving optimal or stable performance may require extensive parameter tuning, which can be impractical in some scenarios.

6. Lack of training: The conventional approaches suffer from a lack of training capability which limits their application to handle complex problems with large time delays.

To address the above problems, 'black box' methodologies have been recognized as powerful tools for training prediction models from big databases. Unlike traditional modeling techniques that rely on understanding the underlying physical phenomena, black box schemes operate by constructing a function solely from interconnected sample data, effectively describing the behavior of a specific system. With the input-output (I/O) data of a specific system, all the required information can be obtained without the need for model identification. In the context of time-delayed systems, many black-box methods are used, including Multiple Linear Regression (MLR) (Meulenbroek & Pichardo, 2020; Plonis et al., 2020), meta-heuristic approaches (e.g., genetic algorithms, harmony search, grey wolf optimizer, etc.) (Shakarami & Davoudkhani, 2016), deep neural networks (DNNs) (J. Han & Hu, 2021; Snyder et al., 2015), Support Vector Machines (SVM) (Lin et al., 2006; Z. Zhang et al., 2019), cross-correlation-based methods (X. Liu et al., 2022), and Long Short-Term Memory (LSTM) (Huang et al., 2022; Tian et al., 2021; Yin et al., 2022), among others. These algorithms can find the behavior of time delays using the input and output information of systems. For example, these algorithms can aid in determining the time delay when it is state-dependent or under multiple conditions. In such situations, DNNs or model comparison models can be adopted for estimating time delays under stochastic conditions (Albrecht & Taylor, 2022).

In (Y. Zhao et al., 2016), a combination of AutoRegressive eXogenous (ARX) and Markov chain model have been introduced to model discrete time delay in measurement signals of industrial processes, considering both time-invariant and time-variant signals. The outcomes of self-validation and cross-validation revealed that the accuracy of the suggested Markov chain (more than 90 percent) is higher than both independent delay and fixed delay while its Root Mean Square Error (RMSE) is smaller than two other schemes. Furthermore, it was verified by numerical simulation that the recursive form of expectation-maximization (EM), which is an expansion of Cappe's technique with an iterative estimation, provides a superior level of identification performance than its recursive version without iteration.

Meulenbroek and Pichardo (Meulenbroek & Pichardo, 2020) developed a multivariable linear regression and three cross-correlation methods to estimate the onset time delay (OTD) in measurement signals. Comparative analysis revealed that the multivariable linear regression method offers a significant improvement by being 80.4 percent more accurate than cross-correlation-based methods for estimating the onset time delay between two measured signals at similar spatial positions. Additionally, time-delay techniques based on correlation were found to consistently underestimate OTD, potentially leading to misinterpretation of results.

A black box scheme, based on a pair of regression equations, has been developed in (Albrecht & Taylor, 2022) that does not require any dynamic model of time delay variability. This algorithm autonomously navigates through parts of an open-loop experiment, accounting for variations in the output using a set of regression equations, thereby providing an estimation of the time delay. Unlike State-Dependent Parameter (SDP) models, the proposed scheme eliminates the need for assumptions of a fixed delay. Furthermore, it is not constrained by the requirements for linear modeling and initial conditions. Compared to neural networks, the suggested technique requires less computational time and is not dependent on a dynamic model. Comparative analysis using the Sum of Squared Errors (SSE) metric demonstrated the superiority of the proposed scheme over previous works.

In (X. Liu et al., 2022), Liu et al. developed a new polynomial inversion to find the relationship between true time delay and time-delay estimation (TDE) in channels in multiantenna by employing least squares (LS) approaches. According to the numerical analysis of (X. Liu et al., 2022), the Mean Square Error (MSE) of the suggested technique is lower than the Cramer–Rao lower bound while incorporating only a limited number of additional multipliers than the convex parabolic method.

Chen et al. (J. Chen et al., 2018) developed a variational version of the Bayesian technique to identify ARX models subjected to communication-varying time delays. In this study, the unknown observations employed in Bayesian were estimated by the Kalman filter. The simulation outcomes in the study clearly confirmed the effectiveness of the proposed Bayesian model in estimating the time delays.

In (Pan et al., 2018), the time delays of a radar system were estimated by integrating the theory of forward-backward linear prediction (FBLP) and Support Vector Regression (SVR). This research work focused on the advancement of processing techniques, providing a reliable and robust time delay estimation in many challenging scenarios. The numerical and experimental verifications demonstrated a higher level of accuracy in delay estimation using the hybrid technique compared to the conventional FBLP and SVR methodologies.

In (Le Bastard et al., 2013) the application of Support Vector Regression for the estimation of time delays was explored using the nondestructive testing and evaluation (NDTE). The outcomes of this research work confirmed the feasibility and usefulness of the SVR scheme-based predictor in all contexts of the system under study. Furthermore, the results obtained by various scenarios of simulation demonstrated a high level of estimation accuracy with low computational time.

The authors of (H. Chen et al., 2020) proposed a modified version of multivariate linear regressive (MLR) to precisely predict time delays involved in three parts including communication, transmission, and processing. The comparative outcomes and detailed analysis revealed that a higher level of prediction accuracy can be obtained by the suggested MLR technique over auto-regressive (AR), NNs, and cubic polynomial model-based (CBMB) approaches, especially in the presence of time delay in vast jitters. Apart from this, the experiment results of this study confirmed that modifications made to the MLR scheme have resulted in improved performance and stable predictive capabilities.

Xiong et. al (W. Xiong et al., 2017) introduced a novel time delay reconstruction based on Gaussian process regression to address the challenges of nonlinearity and time-varying features of industrial processes in the context of soft sensor modeling. A fuzzy curve analysis was incorporated into local time-delay coefficient extraction to capture the time-varying dynamics. The reliability examination and RMSE analysis of different time delay models demonstrated that the proposed scheme is able to improve prediction accuracy by extracting local time delay.

By training neural networks, machine learning is frequently utilized to handle time delays in various fields. Linear regression, as one of the most important machine learning algorithms, has been widely applied to various fields for modeling relationships between variables, but its adaptation to time delay scenarios presents unique challenges. The incorporation of time delays into linear regression models has attracted significant attention in recent years, as it can provide valuable insights into dynamic systems and temporal dependencies. The traditional linear regression model assumes instantaneous relationships between variables, making it unsuitable for dynamic systems with time delays. Researchers have extended linear regression by incorporating lagged variables, allowing for the capture of delayed effects.

The advantages and disadvantages of various techniques have been illustrated in Table 1. Fig. 1 illustrates evaluation methodologies for linear regression models spanning from traditional statistical methods to contemporary approaches.

Table 4.1.: Comparison of Time Delay Mitigation Methodologies (Part 1)

Methodology	**Advantages**	**Disadvantages**
Cross Correlation (Benesty et al., 2004; Podobnik & Stanley, 2008)	• Simple to calculate. • Provides a measure of similarity between two signals.	• Assumes stationary signals. • Sensitive to noise and outliers.
Linear Regression (H. Chen et al., 2020; Ebrahimi & Rajaee, 2017)	• Simple to implement. • Operates well irrespective of dataset size. • Gives information about the relevance of features.	• Prone to underfitting. • Boundaries are linear. • Assumes the information is independent.
AutoRegressive eXogenous (ARX) (X. Chen et al., 2020; Y. Zhao et al., 2017)	• High predictive accuracy. • Estimation of model parameters based on observed data.	• The statistical properties of the data do not change over time. • Limited to linear relationships between variables.
Polynomial Regression (Desai & Shah, 2020; Fan et al., 2020)	• Works on any size of the dataset. • High flexibility and can capture complex relationships between variables.	• Need to select the right polynomial degree for good bias. • Small changes in the input data can result in significant changes in the model.

Table 4.2.: Comparison of Time Delay Mitigation Methodologies (Part 2)

Methodology	**Advantages**	**Disadvantages**
Long Short-Term Memory (LSTM) (Lim et al., 2022; Sherstinsky, 2020)	• Capturing and modeling long-term dependencies. • Flexibility to various configurations and can be combined with other neural network architectures.	• High complexity. • Sensitivity to hyperparameters. • Requires substantial amounts of training data.
Bayesian Models (J. Chen et al., 2018; Shang et al., 2013; Y. Zhao et al., 2017)	• Utilizing a natural way to estimate and quantify uncertainty. • Interaction data and parameters.	• Require substantial computational resources. • Dependencies between model parameters. • High sensitivity to selecting hyperparameters.
Support Vector Machines (SVM) (Ebrahimi & Rajaee, 2017; W. Xiong et al., 2017; Z. Zhang et al., 2019)	• Robust to noisy data. • Capable of modeling nonlinear relationships.	• Sensitivity to outliers in the dataset. • Training SVMs can be computationally intensive. • Prone to overfitting.

4.3. Conclusion

The comparison of various time delay mitigation methodologies highlights the strengths and limitations of each approach, guiding the selection of appropriate techniques for specific scenarios. Traditional methods like Cross-Correlation and Linear Regression offer simplicity and ease of implementation but are limited by their assumptions of stationarity and linearity. ARX models provide high predictive accuracy but are constrained to linear relationships and stationary data. Polynomial Regression offers flexibility and the ability to capture complex relationships but requires careful selection of polynomial degree and can be sensitive to input changes.

Modern approaches such as LSTM and Bayesian Models demonstrate significant advantages in handling complex, non-linear relationships and quantifying uncertainties, respectively. However, these methods demand substantial computational resources and are sensitive to hyperparameters. Support Vector Machines strike a balance with robustness to noisy data and non-linear modeling capabilities but face challenges with computational intensity and potential overfitting.

Given these insights, continuing with methods such as Cross Correlation, Linear Regression, ARX, Polynomial Regression, and LSTM is justified. These methods cover a spectrum from simplicity to advanced modeling capabilities, offering a comprehensive toolkit for addressing various time delay mitigation challenges.

4.4. Selected Methods

Cross Correlation:

- Provides a straightforward measure of similarity between signals.
- Useful as a preliminary analysis tool to identify time delay.

Linear Regression:

- Simple to implement and interpret.
- Effective for understanding feature relevance despite potential underfitting in complex scenarios.

AutoRegressive eXogenous (ARX):

- High predictive accuracy with parameter estimation based on observed data.
- Suitable for scenarios where linear relationships are predominant and data properties are stable over time.

Polynomial Regression:

- High flexibility to model complex relationships.
- Effective for datasets where the relationship between variables is non-linear and requires more than a simple linear approach.

Long Short-Term Memory (LSTM):

- Captures long-term dependencies effectively.
- Versatile for various configurations and can be integrated with other neural network architectures for improved performance.

Each of these methods brings distinct advantages to the table, making them valuable tools for a comprehensive approach to time delay mitigation.

4.5. Mathematical Formulation of the Problem

In real-world problems, there are some relationships between variables in complex systems which can be formulated using multiple variables. In order to mathematically formulate the problem, let us assume X is the input time series with n data points:

$$X(t) = [x_1(t), x_2(t), \ldots, x_n(t)] \text{ for } t \geq 0. \tag{4.1}$$

and the corresponding output time Y series with n data points :

$$Y(t) = [y_1(t), y_2(t), \ldots, y_n(t)] \text{ for } t \geq 0. \tag{4.2}$$

with time delay d

$$d = [d_1, d_2, \ldots, d_n] \tag{4.3}$$

where d_n represents the time delay for the n-th output. These equations may be set appropriately depending on the method to solve for the delay. To address the problem, we need to find dynamics that map the input variables to their corresponding outputs under time delay. Various methodologies can be utilized to address the formulation of input/output problems according to the system's specifications. The following techniques are prevalent schemes to solve the input/output problems in the literature.

- Linear Regression
- Polynomial Regression
- AutoRegressive with eXogenous input (ARX)
- Cross-Correlation

- Support vector machines
- Bayesian methods

4.6. Linear Regression

4.6.1. Regression Analysis

Regression analysis is a statistical technique that may be applied to investigate relationships between two variables where a linear relationship between two variables is estimated. This scheme aims to find the best-fit line in such a way that minimizes the variance between the predicted and observed points. Regression analysis requires that we evaluate data on every kind independently, as opposed to ordination and clustering, which allows us to examine data on every type at once. It can be extended to multiple linear regression comprising of several independent variables (James et al., 2013; Uyanık & Güler, 2013).

Regression analysis emphasizes addressing questions like "Is there a connection between dependent and independent (explanatory) variables? If so, what is the magnitude of this connection? How robust is this association? It can be described as the representation of the connection between dependent and independent variables in the guise of a mathematical function. Many studies make the assumption of a linear relationship between independent and dependent variables. The parameters of these regression models are typically unknown but can be estimated through various techniques. One of the most commonly employed methods for prediction is the least squares approach, which will be applied in this research. The correlation coefficient and coefficient of determination serve as indicators of the strength of the estimated relationships, and the sign of the correlation coefficient signifies the direction of these relationships. Regression analysis is a readily understandable method with extensive applications today, facilitated by statistical software packages such as SPSS, Minitab, Matlab, SAS, and Stata.

4.6.2. Definitions and Fundamentals of Linear Regression Models

-Regression model: A simple model of regression referees to the model that utilizes only one independent parameter for estimating a single dependent parameter.

-Linear regression: In this scheme, a linear scheme is adopted to model a predictive behavior between a response and several dependent or independent parameters. In the following, the various models of linear regression are explained.

• Simple Linear Regression: In this model, we examine the association between a single dependent parameter and one independent parameter.

• Multiple Linear Regression: This scheme utilizes several independent parameters to estimate a single dependent parameter.

-Correlation analysis: Correlation analysis is a statistical technique used to assess and quantify the degree of association or relationship between two or more variables. It helps in understanding how changes in one variable relate to changes in another. The result of a correlation analysis is expressed as a correlation coefficient, which indicates the strength and direction of the relationship. A positive correlation suggests that when one variable increases, the other tends to increase as well, while a negative correlation implies that as one variable increases, the other tends to decrease. Correlation analysis is valuable in various fields, including statistics, economics, and social sciences, for identifying and understanding patterns and dependencies in data. The relationship between correlation analysis and linear regression lies in the fact that correlation coefficients can be used to provide insights into the potential strength and direction of a relationship between variables (Pal et al., 2019).

-Correlation coefficient: often denoted as "r," is a numerical measure that quantifies the strength and direction of the linear relationship between two variables. It falls within the range of -1 to 1, with specific interpretations:

1. Positive Correlation: When the correlation coefficient is greater than zero ($r > 0$), it signifies a positive or direct relationship between the two variables. When one parameter is enhanced, then the other will tend to enhance accordingly.

2. Negative Correlation: When the correlation coefficient is less than zero ($r < 0$), an inverse relationship between the parameters is created. In this correlation, other parameters tend to be reduced by enhancing a parameter.

3. No Correlation: A correlation with the value of zero ($r = 0$) emphasizes that there is no linear relationship between the considered parameters. In fact, any variations of a parameter don't lead to any change in the other parameter.

Statistical techniques (like Pearson's correlation) are often adopted for linear relationships.

4.6.3. Basic equations of simple linear regression

In statistics, a formula for regression is used to determine whether or not there is a relationship between sets of data. Finding out if the information can be fitted to an equation is possible with the use of regression equations. There are actually several kinds of formulas for regression. Simple linear and exponential regression are two of the more used methods (to match the information to an exponential equation or the linear formula). The regression formula that you are likely to encounter in introductory statistics is the linear version (Seber & Lee, 2012).

The regression aims to estimate a term dependent variable (Y) according to the one independent variable (X). In other words, the regression focuses on finding a relationship between the terms Y and X using a line or curve. The simple linear regression is formulated as (Chatterjee & Hadi, 1988; Florea et al., 2016):

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{4.4}$$

where β_0 is the intercept term; β_1 denotes the slope. ϵ denotes the error or deviation term. The schematic of linear regression is depicted in Fig. 2. In multiple types of regression, multiple independent parameters (X_1, X_2,..., X_k) are introduced and thus the regression equation is extended as follows (Chatterjee & Hadi, 1988).

$$Y = \beta_0 + \beta_1 X_1 + \beta_1 X_2 + ... + \beta_k X_k + \epsilon \tag{4.5}$$

In the above equation, the main objective of regression is to predict the terms of β_0, β_1, β_2 in such a way that the best description for the equation is obtained.

4.6.4. Linear Regression Assumptions

Conventional linear regression methods using conventional estimate techniques (e.g., ordinary least squares) make the following essential assumptions (Seber &

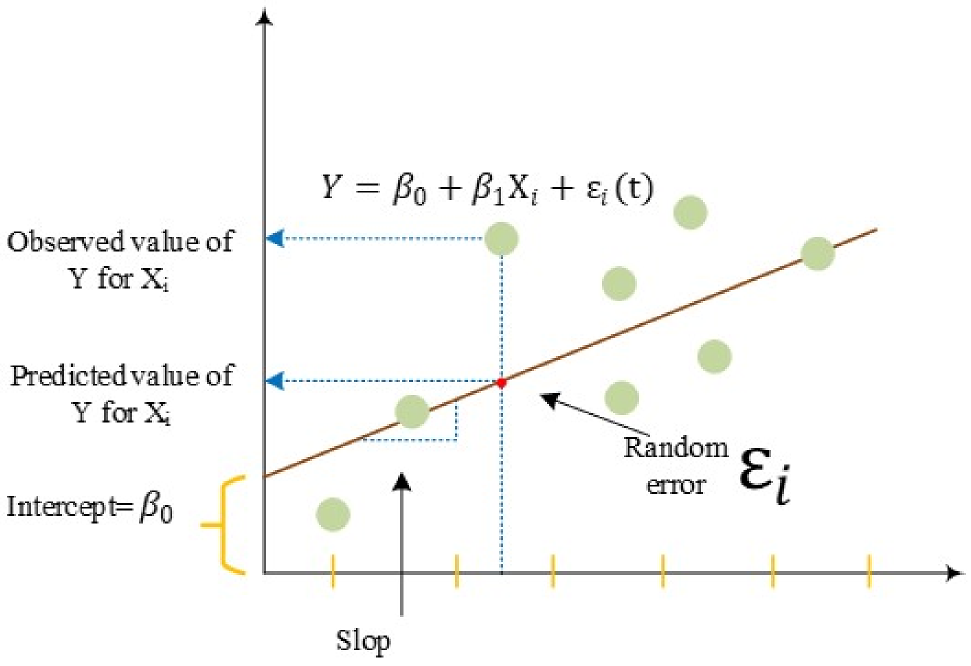
$Y = \beta_0 + \beta_1 X_i + \varepsilon_i (t)$
Observed value of Y for X_i
Predicted value of Y for X_i
Intercept= β_0
Random error
ε_i
Slop

Figure 4.1.: Illustration of Linear Regression

Lee, 2012):

- There is statistical independence among error values. This presupposes that there is no correlation between outcomes and parameter errors. Certain techniques can deal with correlated mistakes, but they usually need a lot more data, unless the model is biased to assume uncorrelated errors by some kind of regularisation. A generic approach to addressing this problem is Bayesian linear regression.
- The probability distribution of the errors is normal with a mean zero.
- The probability distribution of the errors has constant variance.
- There is no measurement error for the independent variables. It is assumed that the observed values of x precise without any inaccuracy.

- Linearity: There is a linear connection at the root between the variables x and y. More broadly, the parameters (regression coefficients) and the predictor variables are linear combinations that result in the mean of the response variable. It is merely a limitation on the variables since linearity is regarded as if the predictor variables were fixed values.

4.6.5. Formulation of linear regression with time delay

The formulation of time-delayed regression is introduced by considering the impact of delay using delayed values of independent variables. The general form of linear regression with time delay is given as (Sinha, 2013):

$$Y_t = \beta_0 + \beta_1 X_t + \beta_2 X_{t-1} + \beta_3 X_{t-3} + ... + \beta_d X_{t-d} + \epsilon \tag{4.6}$$

In the context of our problem, the goal of least squares regression is to find the coefficients β and the time delays d that minimize the sum of squared errors between the predicted output and the actual output.

$$\min_{\beta,d} \sum_{i=1}^{n} (Y_t - \beta_0 + \beta_1 X_t + \beta_2 X_{t-1} + \beta_3 X_{t-3} + ... + \beta_d X_{t-d} + \epsilon)^2 \tag{4.7}$$

which can be solved using techniques like normal equations, gradient descent, or matrix factorization methods to find the values of β and d that minimize this sum of squared errors. The solution for β will give us the weights, and from the values of d_i in your model, we can extract the time delays.

4.6.6. Ridge Regression

Ridge regression is a regularized form of linear regression that adds a penalty term to the least squares loss function to prevent overfitting. Ridge regression is particularly useful when we have many input variables and want to prevent overfitting while still finding the optimal values for β and d (Saunders et al., 1998). In this context, we can set up the ridge regression problem as follows:

$$\min_{\beta,d} \sum_{i=1}^{n} \sum_{t} \left(y_i(t) - \sum_{j=1}^{n} \beta_{ij} x_j(t - d_i) \right)^2 + \alpha \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij}^2 \tag{4.8}$$

where α is the regularization parameter, the term $L2$ norm is adopted to prevent overfitting. The ridge regression problem can be solved by the least squares regression.

4.6.7. Evaluation Metrics for Linear Regression

Several evaluation metrics can be defined to assess the performance of linear regression. These evaluation criteria offer a quantification of how accurately the model generates the observed outputs. In the following, the most popular evaluation metrics including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Coefficient of Determination (R-squared) in the context of linear regression are defined (Chicco et al., 2021; Qi et al., 2020).

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{4.9}$$

$$RMSE = \sqrt{\text{MSE}} \tag{4.10}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{4.11}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.12}$$

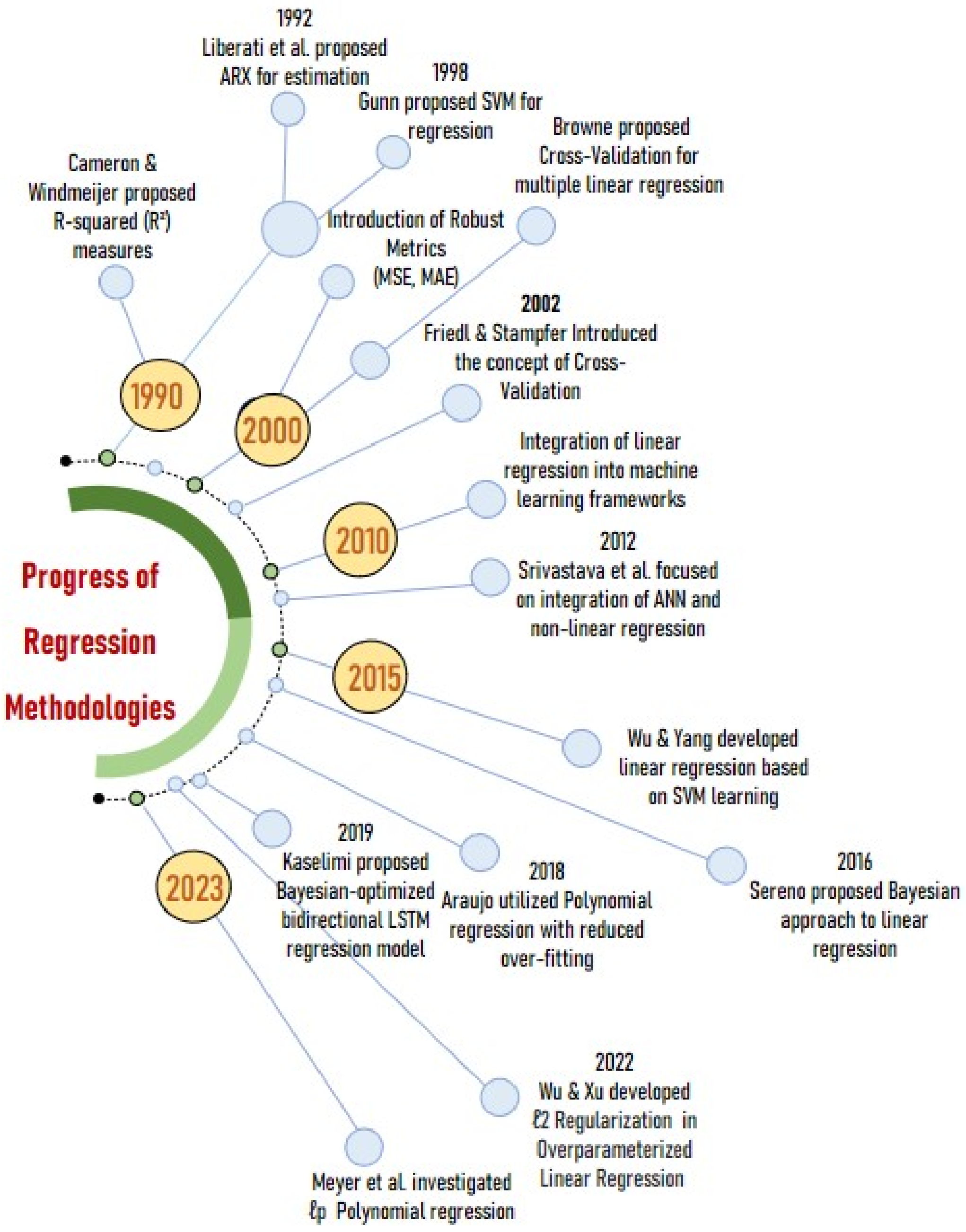
1992
Liberati et al. proposed ARX for estimation
1998
Gunn proposed SVM for regression
Browne proposed Cross-Validation for multiple linear regression
Cameron & Windmeijer proposed R-squared (R²) measures
Introduction of Robust Metrics (MSE, MAE)
2002
Friedl & Stampfer Introduced the concept of Cross-Validation
1990
2000
Integration of linear regression into machine learning frameworks
2010
2012
Srivastava et al. focused on integration of ANN and non-linear regression
Progress of
Regression
Methodologies
2015
Wu & Yang developed linear regression based on SVM learning
2019
Kaselimi proposed Bayesian-optimized bidirectional LSTM regression model
2018
Araujo utilized Polynomial regression with reduced over-fitting
2016
Sereno proposed Bayesian approach to linear regression
2023
2022
Wu & Xu developed ℓ2 Regularization in Overparameterized Linear Regression
Meyer et al. investigated ℓp Polynomial regression

Figure 4.2.: Illustration of Linear Regression

4.7. Formulation of ARX

The AutoRegressive with eXogenous (ARX) is an autoregressive model that has been adopted to identify the block box systems (Duran-Hernandez et al., 2020; Xie et al., 2021). Despite the one-stage estimation can be accomplished in a straightforward manner, the ARX faces some serious challenges in solving the multi-step estimation. For a time-delayed system, the formulation of ARX with time delay is described as:

$$\begin{aligned} y(t) = {} & \alpha_1 y(t-d_1) + \alpha_2 y(t-d_2) + \cdots + \alpha_n y(t-d_n) \\ & + \beta_0 u(t-d_0) + \beta_1 u(t-d_1) + \cdots + \beta_m u(t-d_m) + e(t) \end{aligned} \tag{4.13}$$

4.8. Polynomial Regression

In polynomial regression, the relationship between input X and output Y is defined by an hth-degree polynomial, given as (Sinha, 2013):

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + ... + \beta_h X^h + \epsilon \tag{4.14}$$

In practice, the values of h greater than 3 or 4 are less adopted. This is because large values of h are too flexible, which creates strange forms. Despite the non-linear relationship between the independent variable X and the dependent variable Y can be provided by the above model, the polynomial regression is recognized as a form of linear regression. This is because of linearity between the polynomial regression parameters, i.e., $\beta_1, \beta_2, ..., \beta_h$. Therefore, the linear feature of polynomial regression is oriented from its parameters, not its predictors. Even if its model has the power of X, this model is linear in the variables of $\beta_1, \beta_2, ..., \beta_h$. This linearity of variables provides the possibility to estimate different models using regression methodologies. For the estimation of polynomial regression, the values of the response parameter (Y) and the predictor parameter (X) are required.

4.8.1. Precautions in Polynomial Regression

Polynomial regression is well-known as a specific class of regression analysis that can be employed to model nonlinear problems. In order to verify the model's accuracy, some strategies such as cross-validation are often implemented. This type of analysis is adopted to solve problems in which there is no clear scheme to model the relationship between the input X and output Y data.

Degree of the Polynomial n

In the polynomial analysis, the number of x is determined by the degree of n. In spite of high degrees of polynomials that can handle complex relationships, their accuracy is threatened by overfitting.

Model Fitting

Least squares are a mathematical scheme and are widely used to fit a polynomial regression to any data. In this strategy, it is aimed to explore the values of the coefficients $(\beta_0, \beta_1, \beta_2, \ldots, \beta_n)$ in such a way that minimizes the sum of squared differences between the estimated values and actual terms.

Over Fitting & Regularization

Overfitting refers to a phenomenon that appears when the training of models is completely well while it is not appropriate for testing data. The overfitting problem often happens in cases where noises adversely affect the behavior of training data. This indicates that noises or random deviations are trained as a part of the system model. The main issue is that the new data will not be created by these concepts and will fail the capability of the model to generalize (Vasicek, 2019).

Overfitting is usually included in non-linear models with a high level of flexibility to learn the target data. When the degree of the polynomial is enhanced, a more flexible model can be realized to fit the noise, and this leads to overfitting. In this case, regularization strategies (e.g., Ridge and Lasso regression) can be adopted to add penalties on the size of the variables.

By considering four models for regression, one can express (Sinha, 2013):

$$y(X;\theta) = \omega_0 + \omega_1 X + \omega_2 X^2 + \omega_3 X^3 + \omega_4 X^4 \tag{4.15}$$

Assume $\theta = (\omega_0, \omega_1, \omega_2, \omega_3, \omega_4)$, the order less than two for this polynomial is referred to as high order due to the overfitting may occur for high order polynomial.

The basic regularization scheme aims to decrease the coefficient's values, particularly for high-order parameters. By utilizing these parameters in the objective function, the high-order terms can be suppressed by minimizing the objective function. Based on the concept of regularization, the objective function can be defined as:

$$J(\theta) = 1/2(\sum(h(x^i;\theta) - y^i)^2 + \lambda \sum \omega_j^2) \tag{4.16}$$

where λ is the regularization parameter.

Choosing the Right Degree n

It is special to choose the accurate degree of the polynomial. The underfitting is created by a low level of degree, where the model is not able to capture the related relationship while overfitting is appeared by a high level of degree. Cross-validation and model verification methodologies are good options to choose a proper degree.

4.9. Formulation of Cross Correlation

Cross-correlation is used for analyzing the choice of a known element in an unknown complex (B. Li et al., 2017). This scheme has been developed for a wide range of applications such as telecommunications and image processing. It is adopted as a measurement criterion to investigate and compare the similarity between two or more sets of samples.

$$C_{xy}(\tau) = \sum x(t) * y(t - \tau) \tag{4.17}$$

4.10. Fuzzy Logic Control Method

The Fuzzy Logic Control Method (FLCM) is a powerful control technique that can effectively handle complex and uncertain systems, making it suitable for compensating for time delays in measurement signals. By employing the fuzzy logic into the control system, FLC can accommodate imprecise and uncertain information, and allows for robust and adaptive control in the presence of time delays (Hagras, 2007). A key advantage of the FLC is its ability to handle non linearity and approximate human-like decision-making. Unlike traditional control methods that rely on precise mathematical models, which may not accurately capture the dynamics of the system affected by time delays, the FLC on the other hand, can operate based on linguistic rules and fuzzy sets, which provide a more flexible and intuitive framework for control. FLC offers several benefits when it comes to compensating for time delays in measurement signals such as; Adaptability, FLC allows for the dynamical adjustment of its control actions based on the current state of the system and the measurement delays. The linguistic rules in an FLC system can be designed to explicitly consider the time delays and their impact on the control process. This adaptability allows FLC to compensate for delays and maintain the stability of the system and its performance even with the presence of change in the delay characteristics. The tolerance of uncertainty is another benefit provided by the FLC, it deals with uncertainties associated with time delays, such as variations in delay duration or change in the system dynamics. FLC allows representation and manipulation of imprecise and uncertain information, by using linguistic variables and fuzzy sets, thereby enabling the control system to adapt to varying delay conditions (Nangrani et al., 2018). Nonlinear control capability is one benefit of the FLC that cannot be overlooked, as time delays in measurement signals can introduce non linearity in the control system. FLC is well suited for handling nonlinear systems and can effectively address the complexities introduced by time delays. The linguistics rule in the FLC system can capture the nonlinear relationships between the delayed measurements and the desired control actions. To provide a mathematical representation of fuzzy logic control (FLC) for compensating time delays in measurement signals, some key concepts and equations which is a generalization of the representation should be taken into consideration and may need to be customized based on the specific FLC implementation and system requirements. Firstly, define a fuzzy set B over a universe of discourse X, where each element $x \in X$ has a degree of membership $\phi B(x)$. The membership function $\phi B(x)$ map each element x to a grade of membership between 0 and 1, indicating the degree to which $x \in B$. The membership function can then be expressed using various mathematical functions depending on the shape and characteristics of the fuzzy set. Linguistics variables and fuzzy rules are also key concepts used in FLC, linguistic variables represent the input and output variables in the FLC system. The next concept is the Fuzzy rules which defines the relationship between the linguistic variables and control actions, where each rule consists of an antecedent (IF condition) and a consequent (THEN action). The antecedent of a fuzzy rule is typically expressed as a combination of linguistic variables using logical operators example AND, OR (Zadeh, 2008), (Czogala & Leski, 2012). The next concept is fuzzification, this is the process of converting cripes (numerical) inputs into fuzzy sets using membership functions. After fuzzification, the rule evaluation comes into play as it involves determining the degree to which each fuzzy rule is satisfied based on the membership values that are passed in the linguistic variable. Aggregation is an important concept because it combines the output of individual fuzzy rules to obtain an overall fuzzy output; common aggregation methods that can be used to achieve this include the maximum operator, which selects the highest membership value for each linguistic term, or the product (MIN) operator, which takes the minimum membership value. Lastly, the defuzzification concept converts the aggregated fuzzy output into a cripes value, representing the final control action. The various defuzzification methods that can be used to determine the cripes output value are centroid, mean of maximum (MOM), or weighted average(Mahdavian et al., 2012). Consider n input signals denoted as $x_1, x_2, \ldots, x_n$, and one output signal, denoted as y, we can define the linguistic variable for each input and output signal, and their associated membership functions, let X_i represents the linguistic variable for input signal x_i, and Y for the linguistic variables of the output signal y. The membership functions can be denoted as $\phi X_{ij}(x)$ and $\phi Y_k(y)$, where j and k represent the linguistic terms for the respective variables. We can set up a rule base that maps combinations of linguistics variables from the inputs to the output. Let R_i represents the fuzzy rule i, which can be expressed as: IF x_1 is X_{j1} AND x_2 is X_{j2} AND . . . AND x_n is X_{jn} THEN y is Y_k. Employing the membership function $\phi X_{ij}(x)$ and $\phi Y_k(y)$, we can convert the crips values of the delayed input signals to fuzzy values. The fuzzy rule can then be evaluated using the computed fuzzy values. Each rule will have a firing strength, denoted as

$$w_i = MIN(\phi X_{j1}(x_1), \phi X_{j2}(x_2), \ldots, \phi X_{jn}(x_n)) \tag{4.18}$$

4.11. Neural Network (RNNs) Method

The use of Recurrent Neural Networks (RNNs) can be deployed as an effective method for compensating time delays in measurement signals. RNNs are a special type of neural network that can be used to capture temporal dependencies in sequential data, making them well-suited for handling time-delayed signals. In the process of estimating and compensating for time delays, an RNN model can learn to model the relationship between the current measurement and previous measurements, allowing it to make predictions and estimate the true value of the current measurement despite the delay. An RNN can effectively learn the temporal patterns and compensate for the delay in real time, by leveraging the sequential nature of the data. The RNNs networks are designed to process inputs of variable length, providing the flexibility for handling time series data with varying time delays. They achieve this by utilizing recurrent connections, which allows information to be propagated through time steps. This enables the network to maintain a hidden state that will capture and store temporal information from previous inputs. There are several approaches to recurrent Neural networks (RNNs) that can be used to compensate for time delays in measurement signals. Some commonly used RNN architectures will be studied and discussed.

4.11.1. Simple RNNs Method

Simple RNNs are the basic form of RNNs and have a simple recurrent connection. They suffer from the vanishing gradient problem, which makes it difficult for them to capture long-term dependencies effectively (Su & Kuo, 2019). As a result, they may struggle to compensate for larger time delays. The simple RNNs have a straightforward architecture and can be easier to train and implement compared to more complex variants like LSTMs or GRUs (Alom et al., 2019). The RNNs are best seen to capture delays that are relatively small. In simple RRN, each neuron has a recurrent connection that self feeds the output of the neuron back to itself at the next time step (Murugan, 2018). This allows the network to maintain a hidden state that will capture information from past inputs. The hidden state act as a memory that retains the information about the previous measurements and helps in compensating for the time delayed (Yue-Hei Ng et al., 2015). To employ the simple RNN for compensating time delays in measurement signals, the first thing to do is to preprocess the data; the input data should be organized into a sequence, where each sequence consists of a set of measurements along with their corresponding delayed measurements. The delayed measurement can then be obtained by adjusting the original signals in time. The input data should appropriately be normalized to facilitate training. After pre-processing, a simple RNN network architecture that suits the problem should be designed, the network architecture typically includes an input layer, a recurrent layer which contains recurrent connections, and an output layer. The number of neurons in the recurrent layer can be adjusted based on complexity of the problem and the expected time delays to be determined. The pre-processed data should then be splitted into training and validation sets. The training set should contain sequences of input measurements along with the corresponding target values, which can be the original measurements without the time delays. The validation set is used to monitor and to determine the models performance during the training process and to prevent overfitting the model (Ajiboye et al., 2015). After the data-set is splitted into training and validation sets, the simple RNN can then be trained using the data, the network learns to capture the temporal relationships between the input measurements and its corresponding target values. The weight of the network can be adjusted using optimization algorithms such as stochastic gradient descent (SGD) (Mostafa, 2017). When employing the simple RNN network, one should be mindful of the vanishing gradient problem, which can affect the network's ability to capture long-term dependencies. The simple RNN model should then be evaluated using the validation set after it is trained. The performance of the model should then be calculated using appropriate metrics such as mean squared error (MSE) or root mean squared error (RMSE) to assess the model's ability to compensate for time delays. To increase the model performance, the network architecture or training parameters should be adjusted. While the simple RNNs may struggle with capturing long-term dependencies, they can still be effective in compensating for smaller time delays in measurement signals (Tsiouris et al., 2018). However, in order to capture more complex time delays, more advanced RNN architectures like LSTMs or GRUs are generally preferred.

4.12. State Estimation Method

State estimation is one common method for compensating time delays in measurement signals, it is also known as the state observer method. State estimations is a technique used to estimate the current state of a dynamic system based on available measurements. It is particularly useful when there are time delays in the measurement signals (Schmelzeisen-Redeker et al., 2015). The idea behind the state estimation is to build a mathematical model of the system and use this model to predict the current state, based on previous measurements. The estimated state can then be used in control algorithms of further analysis of the system. One notable approach used in state estimators is the Kalman filter. The Kalman filter is an optimal recursive estimator that can handle noisy measurements and time delays. It operates in two steps, the prediction step and the update step (De Souza et al., 2021). In the prediction step, the filter uses the system model to predict the current state based on the previous state estimate and the system dynamics. An estimate of the measurement delay is also included in the predicted state. In the update step, the filter utilizes the actual measurements and adjusts the predicted state estimate to improve accuracy. The filter uses a weighted combination of the predicted state and the measurement to obtain an updated state estimate. The weights are determined based on the uncertainty of the measurements and the predicted state. The Kalman filter can compensate for time delays in the measurement signals by iteratively performing the prediction and update steps, and providing an accurate estimate of the system state. It is important to note that the Kalman filter assumes linear dynamics and Gaussian noise distributions. If the system's dynamics are nonlinear or the noise happens to be non-Gaussian, then more advanced techniques like the extended Kalman filter (EFK) of the unscented filter (UKF) may be employed. Kalman filtering has numerous technological applications. One notable application is it usage in navigating, controlling and providing guidance for vehicles, particularly in aircrafts, spacecraft and ships that are positioned dynamically (Selimović et al., 2020).

To utilize the Kalman filter for estimating the internal state of a system based on a sequence of noisy observations, the process must be structured according to a specific framework. This framework involves defining the following elements for each time step i:

- T_i: the state-transition model,
- O_i: the observation model,
- σ^p: the covariance of the process noise,

- σ_{ob}: the covariance of the observation noise.

In some cases, the model may also include B_i, the control-input model, depending on the nature of the system. If B_i is present, then the control vector μ_i is also included, representing the external control input applied to the system. According to the Kalman filter model, the actual state at time i evolves from the state at time $(i-1)$ as follows (J. Zhang et al., 2013):

$$x_i = T_i x_{i-1} + B_i \mu_i + w_i \tag{4.19}$$

- T_i represents the state transition model that is applied to the previous state x_{i-1}.
- B_i is the control-input model, which is applied to the control vector μ_i.
- w_i denotes the process noise, assumed to follow a multivariate normal distribution with zero mean and covariance σ^p:

$$w_i \sim \mathcal{N}(0, \sigma^p)$$

This framework allows the Kalman filter to update the estimate of the system's state at each time step, accounting for both the inherent process dynamics and the influence of external controls and noise.

It is important to point out the consequences of time delays on state estimation accuracy. Time delays in measurement signals can have significant implications on the accuracy and reliability of state estimation in the systems. It is essential that this consequences are understood in other to design robust state estimation algorithms that can effectively compensate for the challenges posed by delays (Y. Wang et al., 2020). Observability degradation is a key aspect in highlighting the impact of time delays using state estimation, time delays introduce a lag between the occurrence of a state change and the corresponding measurement, leading to a reduced ability to observe and track the systems actual state. This observability degradation can result in incomplete or delayed information, hindering the accurate reconstruction of the system's internal dynamics. Aside from the Kalman filter method, one notable approach employed in state estimation is the predictor-corrector method. This method involves predicting the future state using available measurements and then correcting the prediction based on the actual measurements when they become available. The aim of this method is to reduce the impact of time delays on state estimation accuracy. (R. Han et al., 2020). A common predictor-corrector approaches include the Smith predictor, more advanced techniques include the moving horizon estimation (MHE) and recursive predictor-corrector methods. It is important to note that each state estimation method has its strengths and also its limitations, identifying these limitations will be pivotal when considering the implementation of the state estimation method. As seen the discussions above, the Kalman filtering methods are effective in linear systems with Gaussian noise but may likely struggle in highly non-linear scenarios. In this case, particle filtering technique provides a more general framework but can be computationally intensive. Extended Kalman filtering (EKF) and Unscented Kalman filtering (UKF) strikes a balance between linear systems and non-linear systems, but may require accurate models of the signal dynamics. Future research in this area could focus on developing hybrid approaches that combine the strengths of different methods or incorporating machine learning techniques to enhance time delay compensation. Additionally, the integration of state estimation methods with adaptive algorithms and optimization techniques could further improve the accuracy and efficiency of time delay compensation in various applications. Overall, state estimation methods provide valuable tools for time delay compensation, enabling precise synchronization and alignment of signals in diverse fields. Their effectiveness and versatility make them an essential component in systems requiring accurate temporal information.

4.13. Optimization techniques for time delay enhancement

In the context of time delay estimation, optimization is essential to achieve the best possible results. Techniques like Gradient Descent, Newton's method, and meta-heuristic algorithms systematically refine model parameters, leading to accurate and efficient performance. By leveraging these methods, we can ensure precise and robust time delay estimations for each result.

In the context of statistical models, optimization techniques play a crucial role in the procedure of model training and predicting coefficients. These schemes offer the possibility to optimize the statistical models of various engineering problems such as robotics, networked systems, and energy management in a systematic manner. Their ability to solve optimization problems is to explore the optimal solution within a pre-defined constraint. In the related literature, numerous optimization techniques have been developed to improve the efficiency and accuracy of parameter estimation such as Gradient descent (GD) (Barbano et al., 2021; Franzese et al., 2021; Z. Zhang et al., 2023), Newton's scheme (El Anes et al., 2016; Najeh et al., 2017) , quasi-Newton method (H.

Chen et al., 2019; Gaffke & Schwabe, 2019; Hu et al., 2019), Regularization (El-Koka et al., 2013; Wu & Xu, 2020), meta-heuristic algorithms (particle swarm optimization, grey wolf optimizer (GWO)) (F. Zhang et al., 2016), so on.

In (Cao & Su, 2023), two distinct versions of GD based on fractional (FGDs) schemes have been introduced which provide a faster convergence and more accuracy of estimation than the conventional identification models. Additionally, the fractional-based Gradient Descent (FGD) ensures unbiased predictions of parameters and guarantees the robust performance of the information matrix.

Chen et al. (J. Chen et al., 2023) introduced a second-order model of nature GD (NGD) for time-delay ARX identification using a redundant rule-based technique. In comparison with the traditional identification methodologies, the proposed scheme improves the computational efficiency and dynamically updates each component in the coefficient vector.

In (Jing, 2023), Jing focused on identifying the effect of time delay by employing a normalized Gaussian approach and cross-correlation function (CCF) in Hammerstein systems. In the identification scheme, a multi-error stochastic information gradient (SIG) is adopted to identify system coefficients using loss function descent criteria. In this study, the efficiency and robust performance of the proposed CCF were verified by comprehensive simulations.

Guo and Ma (Guo & Ma, 2023) introduced a recursive expectation–maximization (REM) to solve the optimization problem of parameter identification in Markov jump ARX (MJARX) mehtod with unknown time delays. In this work, the transition probability matrix and variance were computed by minimizing the mean squared error (MSE). The extensive numerical analysis emphasized superior identification outcomes in comparison with the traditional REM algorithm.

The authors of (J. Chen et al., 2021) proposed a robust and accelerative stochastic gradient descent (RA-SGD) algorithm for ARX models where the convergence rate is enhanced by transferring from the linear to at least quadratic. In the large-scale domain, the proposed RA-SGD scheme takes fewer computational efforts with no restrictions on the step size. By comparing the results of numerical and practical systems, it was verified that more accurate parameter estimation can be reached by RA-SGD than the standard SGD algorithm.

In (Coelho & Neto, 2017), the deployment of the meta-heuristic algorithm (Genetic algorithm) has led to the discovery of polynomial models with the extension of evolutionary polynomial regression (EPR). In the study of (Coelho & Neto, 2017), many experimental scenarios have been performed to confirm the superiority of the EPR scheme in comparison with LR, regression trees, Bayesian estimation (BE), and SVR methodologies. The numerical analysis in terms of 25 percent percentile, error mean and 75 percentile have been made. The outcomes of these scenarios revealed that the hybridization of polynomial regression and regularization scheme can provide superior fitting while needing less computational time than the basic version of EPR.

A stochastic gradient descent was developed in (Franzese et al., 2021) for sampling the Markov chain to address the large-scale approximation challenges of Bayesian modeling arising from the mini-batches operation. In this study, the RMSE and the mean negative log were adopted as metrics of uncertainty measurement. The comparative analysis of the proposed scheme to other types of Bayesian sampling schemes like Stochastic Gradient Hamiltonian Monte Carlo and Stochastic Gradient Langevin Dynamics demonstrated indicated 1) superior performance of the proposed scheme to tackle the measurement uncertainty, 2) straightforward tuning of the proposed scheme than alternatives, and 3) competitiveness of proposed modeling in the realm of deep Bayesian modeling, even when benchmarked against established methods found in the existing literature.

4.14. Optimization Techniques for Solving Regression Problem

In this subsection, various optimization strategies to solve the regression problems are presented.

Ordinary Least Square

Ordinary least square (OLS) is one of the well-known regression techniques that can be developed for estimating coefficients in a straightforward manner. The OLS tries to explore the parameters of the regression model By minimizing the sum of squared differences, i.e., minimizing the error between predicted parameters and actual terms.

The following considerations are given for the OLS regression. 1) The mean of the population is zero. The error term refers to any deviation from the output 'Y' or the dependent/independent parameters that can not be shown. In ideal conditions, the error can be characterized by random probability.

2) Observations are independent of each other and there is no correlation between the error terms. In the cases where there is a correlation, the error term can be estimated by independent parameters.

3) The variance of the error term is constant. This demonstrates that vari-

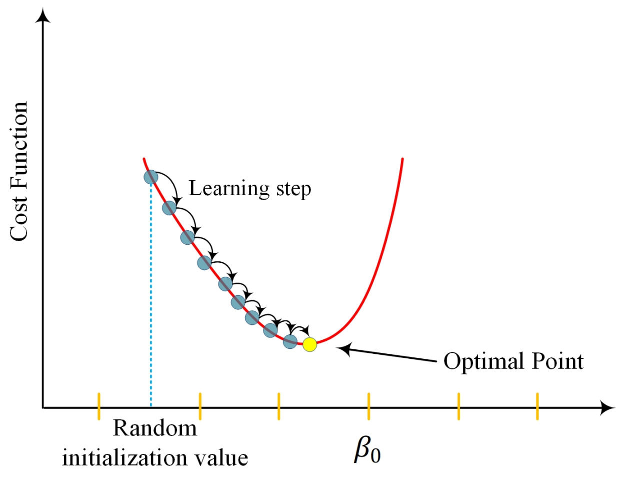

Figure 4.3.: Schematic of optimization procedure of gradient descent

ance will be kept the same for a single or a wide range of observations. This assumption can be confirmed by drawing the curve of the true values versus the residuals.

4)

Gradient Descent

Gradient descent (GD) is an iterative technique to find the minimum local point which is adopted for various regression strategies such as multiple linear regression. The GD is initialized by an arbitrary point which will be updated in an iterative process towards a negative gradient to minimize a defined loss function. Fig. 3 illustrates the procedure learning of GD to obtain the optimal point (Parvathy & Devi, 2014; X. Wang et al., 2021).

Stochastic Gradient Descent

Stochastic GD, entitled SGD, is a modified version of the SG technique which updates the coefficients of a single point at each time. This algorithm can be utilized for complex problems with large datasets where quick convergence is required (Ighalo et al., 2020).

Mini-Batch Gradient Descent

Mini-batch GD (MBGD) is formed by a combination of SGD and full-batch GD techniques. In the MBGD, the coefficients of the problem are updated using a small (or mini-batch) part of the dataset (Lizhen et al., 2022).

4.14.1. Regularization and Overfitting

In this subsection, the basic concepts of regularization are elaborated.

Regularization

Regularization is a common technique to solve overfitting in regression problems. In the regularization, a penalty function is defined that reduces the complexity of a specific model (Kolluri et al., 2020).

$L1$ Regularization (Lasso)

In the L1 regularization, an "absolute value of magnitude" is added to the loss function. Lasso aims to reduce the parameters with less important properties to zero; as a result, it can be adopted for feature selection problems where a large number of features is involved.

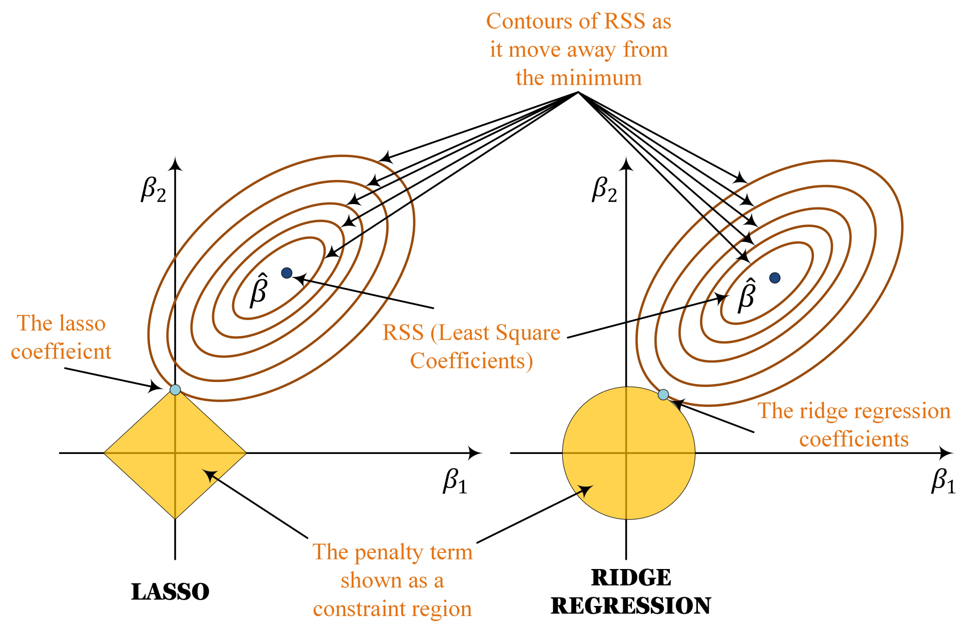
Contours of RSS as it move away from the minimum
β_2
$\hat{\beta}$
The lasso coeffieicnt
RSS (Least Square Coefficients)
The ridge regression coefficients
β_1
The penalty term shown as a constraint region
LASSO
RIDGE REGRESSION

Figure 4.4.: Illustration of L1 (left side) and L2 (right side) regularisation

$L2$ Regularization (Ridge)

L2 regularization (Ridge) introduces a new loss function by adding the squares of the parameters. In this strategy, large values of the coefficient are penalized while tending to be sufficiently small.

The schematic of $L1$ and $L2$ Regularization is depicted in Fig. 4.

Overfitting

Overfitting often occurs in the cases where training data is accurately learned but it is not appropriate for new data, i.e., lack of generalization. This problem is rooted in many reasons such as the high complexity of the problem and the presence of noises in data sets. To address this issue, $L1$ and $L2$ Regularization techniques can be adopted to handle the overfitting issue.

Hyperparameter Tuning

It is essential to accurately adjust the term of α for the effective performance of $L1$ and $L2$ Regularization. To do this, many tuner mechanisms such as cross-validation can be adopted for optimal tuning of hyperparameters.

In summary, regularization is a prominent scheme to handle overfitting, where the right balance between complexity and generalization behavior is preserved.

5. Advanced Methodologies for Time Delay Estimation

In this chapter, we delve deeper into the methodologies we have identified as the most effective for addressing the time delay estimation problem. Building on the foundational concepts discussed in previous sections, we focus on advanced techniques and their practical applications. Our exploration includes a thorough analysis of each method's theoretical underpinnings and implementation strategies. This chapter aims to equip the reader with a comprehensive understanding of the state-of-the-art approaches.

5.1. Cross Correlation Method

The cross-correlation method is a common technique used to determine time delays in measurement signals by aligning the signals in time (Khyam et al., 2016). It measures the similarity between n input and output signals as a function of the time lag between them. By identifying the lag at which the signals appear to be more correlated, one can estimate the time delay (Ianniello, 1982). The cross-correlation method is particularly useful in applications where multiple sensors or measurement devices are involved, and the signals obtained from these devices may not be perfectly synchronized as a result of inherent delays in the system (Bertrand, 2011). The cross-correlation method is also widely used in various fields, including communication systems and image registration, to compensate for time delays and align signals for further analysis or synchronization (Chinaev et al., 2021). Mathematically, in order to determine the cross-correlation of two continuous signals $x(t)$ and $y(t)$, we employ

$$(x \cdot y)\pi = \int_{-\infty}^{\infty} x(t) \cdot y(t+\pi)dt, \tag{5.1}$$

where $(x \cdot y)\pi$ denotes the cross correlation of $x(t)$ and $y(t)$ at time delay π, also $x(t)$ and $y(t+\pi)$ represents the two signals being correlated, with $y(t+\pi)$ being the delayed version of $y(t)$ by a time delay π. Finally, the integral $\int_{-\infty}^{\infty}$ is the sum of the product of the two signals overall time. It is worth noting that the result of the cross-correlation is a function of π, which shows the similarity between the two signal changes with varying time offsets. The peak of this function signifies the time delay at which the two signals $x(t)$ and $y(t)$ are most aligned or correlated (DiBiase et al., 2001; Savorani et al., 2010). A generalized outline of the cross-correlation method for compensating time delays is by obtaining the two measurement signals that you want to compare, after which we determine the time range in which to calculate the cross-correlation. This range should cover the expected maximum time delay between the signals. After that, the cross-correlation function between signals should be computed. The cross-correlation function $(x \cdot y)\pi$ will have a peak value at the time lag corresponding to the time delay between the signals. This peak value is identified to determine the corresponding time lag which represents the time delay between the two signals. After the peak value is identified, the time delay compensation is applied by shifting one of the signals by the identified time lag. The signal shift can be done by interpolating or re-sampling it to align with the other signal. Further analysis can be performed or comparison between the signals after compensating for the time delay. It is of utmost importance that the cross-correlation method assumes that the signals being compared are similar but with a time shift. The method works best when the peak value of the signals is clear in the cross-correlation function, which indicates a strong correlation at a specific time lag. However, cross-correlation can be very sensitive to noise and other sources of interference in the signal. It may be pertinent to pre-process the signals or apply filtering techniques in order to improve the accuracy of the time delay estimation. Also, cross-correlation assumes that the signals are linearly related and that the delay is constant over time (Boker et al., 2002). In cases where the signals are non-linearly related, or if the delay changes over time, then methods such as a phase-based method or dynamic time warping may be appropriate (Dinov, 2017). Moreover, the cross-correlation method is an extensively used technique for the compensation of inherent time delays in measurement signals. Specifically, it can be particularly useful in applications such as signal synchronization, signal alignment and time delay estimation (J. A. Zhang et al., 2021). In practical applications, signals are often discrete rather than continuous. The cross-correlation function for discrete signals is calculated using the formula

$$(x \cdot y)\pi = \sum_{-\infty}^{\infty} x(t) \cdot y(t+\pi), \tag{5.2}$$

where the sum is taken over the discrete samples. The fast Fourier transform (FFT) is employed to efficiently compute cross-correlations in the frequency domain, and this is especially useful for large data sets. In many applications, it's common to use normalized cross-correlation, which accounts for differences in signal amplitudes; the normalized cross-correlation function is represented as

$$N_{xy}(\pi) = \frac{(x \cdot y)\pi}{\sqrt{(x \cdot x)\pi * (y \cdot y)\pi}}. \tag{5.3}$$

Here $(x \cdot x)\pi$ and $(y \cdot y)\pi$ are the auto-correlation functions of $x(t)$ and $y(t)$ respectively. The cross-correlation method is susceptible to noise, and in some use cases, the presence of multiple peaks in the cross-correlation function may lead to ambiguities in determining the correct time delay. Techniques such as windowing and filtering can be applied to mitigate the effects of noise. Additionally, using advanced algorithms, like matched filtering or adaptive filtering, can improve performance in noisy environments (Ahmadi et al., 2021).

5.2. Linear Regression Method

The linear regression method is among the methods that can be applied to estimate and compensate for the time delays in measurement signals. This is done by modeling the linear relationship between the delayed signals and the corresponding actual values (Carrion & Spencer, 2006). The linear regression method can be incorporated to compensate for the time delays in measurement signals by employing the time-shifted predictors into the model (Khorram et al., 2019). This method can also be seen as the time lag regression or the dynamic regression method. The basic idea of the regression model is to introduce lagged values of the input signals into the regression model in order to account for the delays. Generally, the model can be represented mathematically as;

$$y_t = \beta_0 + \beta_1 x_{t-\pi} + \epsilon_t, \tag{5.4}$$

where y_t denotes the independent signal at time t, $x_{t-\pi}$ is the lagged value of the independent variable with time delay π, β_0 and β_1 represents the regression coefficients, and finally ϵ_t denotes the error term with respect to time t. It is important to note that linear regression provides a simple and interpretable approach to compensating time delays in measurement signals. Its effectiveness depends majorly on the linearity assumption and the quality of the data (Yu & Horng, 2019). A step-by-step approach to using the linear regression method for compensating time delays in measurement signals can be outlined as follows. Data collection is a crucial part of signal processing, especially when compensating for time delays in measurement signals (Lai et al., 2013). Collection of data sets consisting of paired measurements of the original signal and the delayed signal is of utmost importance (C. Li, 2013). The original signal will represent the independent variable x, while the delayed signal will be the dependent variable y. After the collection of the data, it is necessary for the data to be pre-processed in order to remove outliers or noise that could affect the accuracy of the linear regression model (Gibert et al., 2016). In linear regression modeling, the process of feature engineering for extracting relevant features from the original signals is important as it could help with predicting the delayed signal. The feature engineering process could include past values of the signal, time derivatives, or any other relevant information. Upon completion of preprocessing the data, it is pertinent to train the model using the data set and splitting it into a training set and a test set. The training set can be 70 percent of the total data set of size N, while the test set comprises the other 30 percent. The model when fitted, will learn the relationship between the original signal and the delayed signal (Xiao et al., 2014). After model development, the need for model evaluation is important to evaluate the performance of the trained model using the test set. Appropriate evaluation metrics can be used to evaluate the model. Metrics such as mean squared error (MSE), or coefficient of determination (R-squared) are used to assess how well the model estimates the delays in the signals (Osah et al., 2021). The next step after model evaluation is to validate the linear regression model. If the model performs well on the test set, you can further validate it by applying it to new and unseen data. This process ensures that the model's performance is not specific to the training and test data set and helps to avoid model overfitting and under-fitting (Deng et al., 2015). Once the model is well-trained and validated, it can then be used to estimate and compensate for the time delay in real-time measurement signals. Given a new measurement of the original signal, the model will predict the corresponding delayed signal while effectively compensating for the delay. Once again, it is important to note that the success of this approach depends on the assumption that there is a linear relationship between the original signal and the delayed signal. If no such linear relationship exists, or if the relationship is non-linear, then alternative methods such as nonlinear regression or machine learning algorithms like neural networks may seem more appropriate (Murray-Smith, 1994).

5.3. ARX Modeling

Auto-regressive with exogenous inputs models are a popular approach that is used in the identification of systems and controlling the dynamics of a system based on input-output signals (Y. Zhao et al., 2017). When considering time delays in measurement signals, ARX modeling can be augmented to accommodate these delays (Anderson et al., 1975). In order to estimate time delays in the ARX model, the concept of delayed inputs and outputs can be utilized. Since time delays are often encountered in real-world systems, it is important to model and compensate for them when analyzing or controlling dynamic processes (Mulder et al., 2017). The modified form of an ARX model that includes time delays can be represented mathematically as follows:

$$Y(t) = \sum_{i=1}^{n_a} a_i Y(t-i) + \sum_{j=1}^{n_b} b_j V(t-j) + e(t-d), \tag{5.5}$$

where $Y(t)$ is the output signal at time t, $V(t)$ is the input signal at time t, a_i and b_j are the model parameters for the auto-regressive and exogenous input components n_a denotes the auto-regressive outputs of the model n_b denote the exogenous input part of the model while $e(t-d)$ represents the delayed noise term. Finally, d represents the time delay between the input and output signals. To evaluate the value of d adequately, knowledge of the system has to be considered through the modeling process (Mould & Upton, 2012). The basic idea behind ARX modeling is to try to model the relationship between the current output signal of a system and its past outputs and input signals by using historical data. The ARX model estimates the system dynamics and can predict the current output based on the past inputs and outputs (Privara et al., 2013). This prediction can then be used to compensate for the time delay in the measurement signal. To effectively apply the ARX modeling to compensate for time delays, the following steps should be employed. Data collection, the process of gathering data sets that include the input and output signals with corresponding time stamps is important. Additionally, it is essential to have a sufficient amount of data that covers a range of operating conditions. After the data has been collected, it is important to determine the time delay, and analyze the time delay in the measurement signal by examining the cross-correlation between the input and output signals. Processing the collected data is important in order to remove outliers or noise factors from the data that could affect the modeling process. The data can also be re-sampled if the time stamps are unevenly spaced. Model identification is then employed to estimate the parameters of the ARX model. The model typically takes the form of a ratio of polynomials, where the numerator represents the past inputs and the denominator represents the past outputs. Validating the ARX model should then follow by comparing its predictions to the actual output data. This helps to ensure that the model accurately captures the system dynamics and compensates for the time delays. Upon validation of the ARX model, it can be used to compensate for time delay in real-time measurement. In order to incorporate time delays in ARX models, it is essential to accurately represent controlling systems with such dynamics (Y. Liu et al., 2021). However, estimating time delays using the ARX model can be a challenging task and thus requires prior knowledge or additional analysis techniques to determine appropriate delay values (Kasture et al., 2015). Once the model parameters are estimated, the ARX model can, therefore, be employed for simulation, prediction, or control purposes depending on the specific application(Peng et al., 2006). Finally, it is worth noting that ARX modeling assumes linearity and time in variance in the system dynamics, which may not always hold true in practice. Therefore, it is necessary to be careful when analyzing the system and evaluating the model's performance by ensuring its effectiveness in estimating and compensating for time delays in measurement signals.

5.4. Polynomial Regression Method

Polynomial regression method is a technique used for time delay estimation and compensation in control system's and signal processing. It is a model-based approach that involves approximating the system's time delay using a finite number of polynomial functions. The method is particularly useful when the time delay is known to be constant or can accurately be approximated. The polynomial regression method employs the use of multiple independent signals denoted X_i for $i = 1, 2, 3, ..., n$ and one dependent variable denoted as y. The method can also be referred to as the multiple polynomial regression method (Benesty et al., 2004). The polynomial regression method provides a flexible and efficient approach for time delay estimations and compensation in signal processing and control systems. However, it is important to note that the method's effectiveness depends on the accuracy of the time delay estimation and a fitting selection of polynomial functions and weights. The general second-order polynomial regression equation with two dependent variables is represented as:

$$E(y) = \psi_0 + \psi_1 X_1 + \psi_2 X_2 + \psi_q X_1^2 + \psi_r X_2^2 + \psi_e X_1 X_2 + \epsilon, \tag{5.6}$$

where the parameters ψ_1 and ψ_2 show the linear effect, the parameters ψ_q and ψ_r denote the quadratic effect, ψ_e represents the interaction effect, while ϵ shows the tolerance level or noise effect (Candon et al., 2022). The model can be rewritten in matrix form as:

$$P = QB + \epsilon, \tag{5.7}$$

where $P = (y_1, y_2, y_3, \dots, y_n)$ is the output signal, Q is the matrix of multiple independent signals X. We can obtain the solution of equation (2) by applying the least square method. The polynomial regression approach is identified as an adequate machine learning regression model in determining the inherent time delays in measurement signals, this is in line with (Comte-Bellot & Corrsin, 1971), (Duriez et al., 2017) and (Gibbons & Ringdal, 2006). Consider a uniform random sample of input size N. We take 70 percent of the data set consisting of signal pairs as the training data and the remaining 30 percent as our test set. The regression fit will be determined by minimizing the residual on the training set by applying the normal equation (Hanus, 2019), (Jain et al., 2004). One notable reason why the Polynomial regression was considered in this study is to enable addressing the problem of under-fitting, i.e., avoiding incomplete generalization of the signal's time delay estimation. In applying the polynomial regression method, it is important to note that the degree of the polynomial is an important factor. A higher degree polynomial can capture more complex relationships between the input and output signals, but on the other hand, it may also lead to over-fitting if the data is small or noisy (Kaiser et al., 2018). A lower-degree polynomial may not capture the full complexity of the time delay and can further lead to an underfitting of the problem. It is therefore important to strike a balance and to select a degree that adequately captures the behavior while avoiding over-fitting the data (Meiri & Zahavi, 2006). Prior to performing polynomial regression, it is important for the data to be processed. This process may involve handling missing values and normalizing the input and output signals. Normalizing the data helps to remove the impact of scale and puts all features on the same scale and can lead to faster training and better performance of the regression model. Accurately estimating the time delay between the input and output signals is also crucial for aligning the data points before applying the polynomial regression method (Nelles & Nelles, 2020). The polynomial regression model can then be assessed to validate how well it performs on unseen data. It is important to split the data set into training and testing subsets to evaluate the accuracy of the model prediction on the testing data. These steps help to ensure that the model generalizes well and is not over-fitting or under-fitting the training data. Polynomial regression assumes a linear relationship between the input and output signals; if the relationship is highly nonlinear, polynomial regression may not be a suitable method or applicable to compensate for time delays. In such cases, other techniques, such as neural networks, support vector regression, or Gaussian processes, can be explored to accurately model and compensate for time delays. The concept of model refinement in polynomial regression models can not be overemphasized. It is important to note that if the initial polynomial regression model does not provide satisfactory results, further refinement may be necessary. This process may include adjusting the polynomial degree, considering interaction terms between signals, or exploring different regression algorithms. Continuous experimentation and proper refinement may be required to achieve the desired estimation and compensation accuracy.

5.4.1. Long Short-Term Memory Networks (LSTMs)

Long Short-Term Memory (LSTM) networks are a popular type of RNN that addresses vanishing gradient problems that are associated with traditional RNNs. The LSTM is one of the promising models that tends to solve the problem of preserving long-term information and skipping short-term input (An et al., 2020). Apart from the typical input and output, LSTMs incorporate a memory cell and three gating mechanisms (input gate, forget gate, and output gate) that regulate the flow of information within the network. This architecture enables LSTMs to capture long-term dependencies and effectively learn and compensate for time delays in measurement signals. The working dynamics of RNNs can be described by a set of differential equations. If we consider a simple RNN with a single recurrent unit, the state of the recurrent unit at time step t, denoted as R(t), evolves according to the equation:

$$\frac{dR(t)}{dt} = f(w_{hh} * R(t-1) + w_{hx} * x(t) + b_h), \tag{5.8}$$

where $R(t)$ denotes the state (or hidden state) of the recurrent unit at time step t, $x(t)$ is the input at time step t, w_{hh} is the weight matrix connecting the recurrent unit to itself, w_{hx} denotes the weight matrix connecting the input to the recurrent unit, while b_h denotes the bias vector. Long Short-Term Memory Networks offer a powerful approach for compensating time delays in measurement signals. This is by leveraging their ability to model temporal dependencies and retain long-term information. LSTM networks can effectively estimate and compensate for the delays, leading to improved accuracy and real-time performance in various applications. However, one needs to pay careful consideration to data processing, network architecture design, and computational requirements when employing the LSTMs for time delay compensation (X. Wang et al., 2020).

6. Code Implementation

In this chapter, we explore the practical aspects of implementing the selected time delay estimation methods. We provide a detailed account of the algorithms utilized, outlining their step-by-step processes. This section covers the integration of theoretical concepts into practical applications, highlighting the coding strategies, software tools, and computational techniques employed.

6.1. Code Implementation for Time Delay Estimation and Compensation

In various signal processing applications, it is often necessary to estimate and compensate for time delays between input and output signals. Time delays occur due to physical phenomena, system characteristics, or transmission delays. Accurate estimation and compensation of these delays are crucial for tasks such as synchronization, system identification, control, and analysis. This section seeks to present a detailed implementation of different time delay estimation and compensation methods that have been discussed, and to evaluate their performance using a real-world dataset containing input and output signals. The goal is to accurately determine the time delay between input and output signals in processing system, with a specific interest in achieving an optimized time delay. The implemented methods include cross-correlation, polynomial regression, linear regression, ARX modeling, and Long Short Term Memory (LSTM) networks. The performance of these methods is evaluated, and an optimization approach is employed to determine the best-performing method for estimating the optimal delay. The solution implementation was done with python version 3.12.1 (George & Sokolovsky, 2014). Python libraries such as Pandas for data frame manipulations, Numpy for arrays related operations and Scipy, Scikit learn and Keras for optimization and signal processing manipulations.

6.1.1. Code Analysis

In our implementation of the solution there are two approaches, the first approach considers the maximum or peak time delay of the five methods implemented, while in the second approach, the solution considers all input and returns the average time delay, using appropriate score parameters. The solution implementation began with importing the required Python libraries for data manipulation, numerical operations, signal processing, and machine learning model development. Overall, the solution provides a modular and extensible framework for time delay estimation and compensation. It allows flexibility in choosing different methods and evaluating their performance. The sampling rate which is determined by the frequency at which the data points are collected or recorded was defined, and is expressed in samples per unit of time (sec). The sample rate was dynamically determined with respect to each dataset passed during the program runtime. A simple Pseudocode to illustrate how the sampling rate was calculated is given below.

Algorithm 1 Estimate Sample Rate

```
1: procedure ESTIMATE SAMPLE RATE(dataset)
2:     if dataset is not empty then
3:         // Assuming dataset has at least two data points
4:         // Calculate the average index difference between consecutive samples
5:         index_diff ← index of second data point - index of first data point
6:         total_data_points ← number of data points
7:         if index_diff > 0 and total_data_points > 1 then
8:             // Estimate sampling rate
9:             sample_rate ← 1 / index_diff
10:            // Output the estimated sample rate
11:            Output "Estimated Sample Rate:", sample_rate, "samples per unit index"
12:        else
13:            Output "Unable to estimate sample rate. Index difference or total data points are insufficient."
14:        end if
15:    else
16:        Output "Dataset is empty. Unable to estimate sample rate."
17:    end if
18: end procedure
```

Cross Correlation Implementation Analysis

Before performing cross-correlation, it is important to pre-process the signals appropriately; this may involve filtering and removing outliers and normalizing the signals to ensure accurate time delay estimation. The Cross-Correlation function is defined as `cross_correlation`, and takes the input signal and the output signal as parameters given as $Signal1$ and $Signal2$ respectively, also the sampling rate which is the time difference between data points measured illustrated by $algorithm5$is passed as a parameter in the defined function. The time lag obtained from the cross-correlation calculation was converted into a time delay value. This conversion depends on the sampling rate of the signals. By knowing the sampling rate, the time delay can be expressed in seconds. The cross-correlation function returns the maximum time delay across all input and output channels for Implementation One and the exact time delay by averaging delays across all input channels for Implementation Two whenever an instance of the function is created. The cross-correlation function was calculated using K = 1730 samples, having a sampling rate of Fs=1000Hz. the optimal delays obtained from the solution implementation, shows that the cross correlation generalizes better time delay estimate on the test data set. This implies a proper working of the method, utilizing the fast Fourier transform (FFT) which efficiently computes the cross correlation in the frequency domain. This is especially useful in large data sets. The accuracy of the time delay estimation obtained depends on various factors, which include the quality and characteristics of the input and output signals in the data set, the signal-to-noise ratio, and the presence of any distortions. The limitations of this method has also been discussed, these limitations involves inaccuracy to sampling rate and the assumption of a linear relationship between the signals. The results, the estimated time delay and the correlation coefficient at the peak, provides valuable information for understanding and compensating for time delays in signal processing applications. These results can be vital in several task such as signal alignment, synchronization, and latency compensation, enabling improved analysis, control, and decision-making processes. Overall, the implementation of cross correlation for time delay estimation in our solution offers a valuable approach for various applications requiring accurate time synchronization and compensation. The versatility and effectiveness of the cross-correlation method make it a fundamental tool in signal processing, supporting advancements in fields such as audio and speech processing, communication systems, and image registration. The Cross-Correlation method implements the Fast Fourier transform (FFT) for convolution. A simple pseudocode to illustrate the cross-correlation method implementation is given below.

Algorithm 2 Cross Correlation

```
1: function CROSS_CORRELATION(input_signals, output_signal, sampling_rate)
2:     $input\_signals\_array \leftarrow input\_signals.to\_numpy()$
3:     $output\_signal\_array \leftarrow output\_signal.to\_numpy()$
4:     $N \leftarrow len(input\_signals\_array[1])$
5:     $num\_rows \leftarrow len(input\_signals\_array)$
6:     $cross\_corr \leftarrow zeros((num\_rows, N))$
7:     for $i = 0$ do $num\_rows - 1$
8:         for $k = 0\ N - 1$ do
9:             $cross\_corr[i,k] \leftarrow \sum(input\_signals\_array[i,k :] * output\_signal\_array[0 : N - k])$
10:        end for
11:    end for
12:    $time\_delay \leftarrow argmax(cross\_corr, axis = 1)/sampling\_rate$
13:    return $argmax(time\_delay)$
14: end function
```

Mathematically, the implementation of the cross-correlation method of two signals say f and g at lag k is given by

$$C(f,g)[k] = \sum_{n=-\infty}^{\infty} f[n].g[n-k] \tag{6.1}$$

The time delay (k) is found by locating the index of the peak or maximum cross-correlation along all the considered channels.

Linear Regression Method Implementation

Linear regression is a popular statistical technique used for estimating the relationship between a dependent variable and one or more independent variables. In this context of time delay estimation, linear regression can be applied to model the relationship between input and output signals to estimate the time delay between the signals. This section outlines the detailed implementation of linear regression for time delay estimation using Python scripts. The Linear Regression function was declared as `linear_regression_time_delay` taking `input_signals` and `output_signal` as function parameters. The input and output signals were converted to numpy arrays in order to handle the dimensionality of the pandas data frame. An empty array was initialized to store the time delays for each channel in the data set. Afterward, the linear regression was performed for each channel using a for loop that iterates through the channels one after the other until all input signals are considered. The solution employs the numpy.cov and numpy.var functions to calculate the covariance and variance between the input channel and the output signal. The covariance measures the linear relationship between the two variables, while the variance quantifies the variability of the input signal. The linear regression process involves calculating the slope of the linear relationship between the input and output signals. The slope represents the rate of change in the output signal per unit change in the input signal, it is then used as an estimate of the time delay. A positive slope suggests a delay between the input and output signals, while a negative slope indicates an advance in the input signal relative to the output signal. Overall, the implementation of linear regression for time delay estimation in the implementation, contributes to the field of signal processing and offers a valuable tool for various applications requiring accurate time synchronization and compensation. The linear regression function was implemented and tested with a test data set with a sample size of K=1730 samples and a linear degree of 2. The visualizations of the distribution of time delays in the results chapter obtained above show how well the linear regression model captures the relationship between the input and output signals and measures the linearity in implementing the method. It suggests that a less linear distribution of time delays would indicate a better fit of the linear regression model to the data. The accuracy of the time delay estimation depends on various factors, the appropriateness of feature extraction techniques, and the performance of the linear regression model. The model's performance has been evaluated using suitable metrics to ensure reliable time delay estimates, and in this implementation, we employed the r-squared metrics to determine how well the linear regression model generalizes on the data set containing input and output signals. The results obtained from the linear regression model, such as the estimated time delay and the performance metrics, provide valuable information for understanding and compensating for time delays in signal processing applications. These results can assist in tasks such as synchronization, alignment, and latency compensation, enabling improved analysis, control, and decision-making processes. A pseudocode illustrating the linear regression method implementation is given as follows:

Algorithm 3 Linear Regression Time Delay

```
1: function LINEAR_REGRESSION_TIME_DELAY($input\_signals, output\_signal$)
2:     $input\_signals\_array \leftarrow input\_signals.to\_numpy()$
3:     $output\_signal\_array \leftarrow output\_signal.to\_numpy().flatten()$
4:     $num\_channels \leftarrow \mathrm{shape}(input\_signals\_array)[1]$
5:     $time\_delays \leftarrow \mathrm{zeros}(num\_channels)$
6:     for $channel = 0\ num\_channels - 1$ do
7:         $input\_channel \leftarrow input\_signals\_array[:, channel]$
8:         $cov \leftarrow \mathrm{cov}(input\_channel, output\_signal\_array)[0, 1]$
9:         $var\_x \leftarrow \mathrm{var}(input\_channel)$
10:        $slope \leftarrow cov/var\_x$
11:        $intercept \leftarrow \mathrm{mean}(output\_signal\_array) - slope * \mathrm{mean}(input\_channel)$
12:        $time\_delays[channel] \leftarrow -intercept/slope$
13:    end for
14:    return $\mathrm{argmax}(time\_delays)$
15: end function
```

Polynomial Regression Method Implementation

The solution employs the numpy.polyfit function to fit a polynomial function to the input and output signals. The input and output signals were converted into numpy arrays to address the dimensionality of the data. The degree of the polynomial is specified as a parameter. The function estimates the coefficients of the polynomial that best fits the data using the least squares method. The number of channels was stored as a variable to determine the total number of channel in the data set. Furthermore, an array was initialized to store time delays for each channel. The polynomial regression was performed for each channel using a for loop. The solution fitted a polynomial regression model, thereafter calculating the time delay, which is proportional to the coefficient of the highest-degree term. Finally, the function returned the maximum time delay across all the channels. Once the polynomial coefficients are estimated, the solution extracts the coefficient of the highest-degree term in the polynomial. This coefficient is used as an estimate of the time delay. The rationale behind this is that the highest-degree term captures the time-delayed relationship between the input and output signals. A positive coefficient indicates a delay, while a negative coefficient suggests an advance in the input signal with respect to the output signal. The polynomial regression function was implemented on a test data set with sample size of K=1730, with a polynomial degree of 2.

The distribution of the time delays obtained from the calculation of input and output signals of the test data sets. The method performed slightly better as compared to the cross correlation method. However, an increased degree of the polynomial regression method can give a better and more precise accuracy. The accuracy of the time delay estimation depends on several factors, including the quality and characteristics of the input signals, the appropriate selection of the polynomial degree, and the performance of the polynomial regression model. The model's performance has been evaluated using suitable metrics to ensure reliable time delay estimates. The results obtained from the polynomial regression model provide valuable information for understanding and compensating for time delays in signal processing applications. These results can assist in tasks such as synchronization, alignment, and latency compensation, enabling improved analysis, control, and decision-making processes. Overall, the implementation of polynomial regression for time delay estimation offers a valuable approach for various applications requiring accurate time synchronization and compensation. The flexibility and adaptability of polynomial regression make it a useful tool in signal processing, enabling the estimation of complex time delays and supporting advancements in signal processing. The pseudocode illustrating the polynomial regression method implementation is given as follows:

Algorithm 4 Polynomial_regression_time_delay

```
1: function POLYNOMIAL_REGRESSION_TIME_DELAY($input\_signals, output\_signal, degree$)
2:     $input\_signals\_array \leftarrow input\_signals.to\_numpy()$
3:     $output\_signal\_array \leftarrow output\_signal.to\_numpy().flatten()$
4:     $num\_channels \leftarrow \text{shape}(input\_signals\_array)[1]$
5:     $time\_delays \leftarrow \text{zeros}(num\_channels)$
6:     for $channel = 0\ num\_channels - 1$ do
7:         $input\_channel \leftarrow input\_signals\_array[:, channel]$
8:         $coeffs \leftarrow \text{polyfit}(input\_channel, output\_signal\_array, degree)$
9:         $time\_delays[channel] \leftarrow -coeffs[-2]/(degree * coeffs[-1])$
10:    end for
11:    return $\text{argmax}(time\_delays)$
12: end function
```

The *np.polyfit* function from numpy is a powerful tool for polynomial regression, fitting a polynomial of degree 2 to the set of data points. This can be particularly useful in signal processing and communications by supplying the np.polyfit function with time points ($input_channel$) as the x-values and the corresponding signal values ($output_signal_array$) as the y-values. This step aims to find a polynomial that best represents how the input signal transforms into the output signal over time.

AutoRegressive eXogenous Method Implementation

The ARX model is a widely used approach for modeling and estimating the relationship between input and output signals. It is particularly useful for estimating time delay between two signals. This section gives a detailed explanation to the solution of the ARX model for time delay estimation using Python. The ARX modeling time delay function was defined to take input signals, the output signal and the order as parameters. The signals were converted to numpy arrays in order to handle its dimensionality when fitting the data. The number of channels across the data was also determined, and an empty numpy array of zeros was initialized to store the time delays. The solution uses the numpy.linalg.lstsq function to estimate the coefficients of the ARX model. This function performs a least squares estimation, finding the coefficients that minimize the sum of squared residuals between the predicted output and the actual output. The estimated coefficients represent the weights assigned to the input signals in the ARX model. In the ARX model, the time delay is estimated based on the coefficient of the first input signal. Since the coefficients represent the relationship between the inputs and the output, the coefficient of the first input provides an estimate of the time delay between the input and output signals. In the process of calculating the time delay estimation using auto-Regressive with exogenous inputs (ARX) modeling, the construction of a Hankel matrix is a crucial step as implemented in the solution. In our implemented solution, a Hankel matrix was constructed using the scipy.linalg.hankel function. A Hankel matrix is a specific type of matrix where each anti-diagonal from top-right to bottom-left contains constant values. In the context of ARX modeling, the Hankel matrix is created by sliding a window over the input signal, with the window size determined by the order of the auto-regressive model. The Hankel matrix is utilized to capture the temporal dependencies in the output signal and create a mathematical model that represents the system's dynamics. The ARX modeling approach aims to relate the current output of a system to its past outputs and inputs. This is achieved by creating a mathematical model that includes terms representing the past values of the output signal. The construction of the Hankel matrix is essential for organizing these past values in a structured form.

The accuracy of the time delay estimation depends on several factors, including the quality and characteristics of the input signals, as well as the performance of the ARX model. The model's performance has been evaluated using suitable validation techniques to ensure reliable time delay estimates. The results obtained from the ARX model implemented in the solution, such as the estimated time delay and the model coefficients, provide valuable information for understanding and compensating for time delays in signal processing applications or systems. Mathematically, the implementation of the auto-regressive exogenous method is characterized by the equation:

$$\begin{aligned} y(t) + a_1 y(t-1) + a_2 y(t-2) + \cdots + a_p y(t-p) = \\ b_0 u(t) + b_1 u(t-1) + b_2 u(t-2) + \cdots + b_q u(t-q) + e(t), \quad (6.2) \end{aligned}$$

where $y(t)$ is the output at time t, $u(t)$ is the input at time t, $a_1, a_2, ..., a_p$ are the auto-regressive coefficients, $b_0, b_1, ..., b_q$ are the exogenous input coefficients and $e(t)$ is the models error term. The time delay in the ARX model can be obtained by examining the auto-regressive part. the time delay is related to the auto-regressive coefficients, for first-order ARX model ARX(1,1), the time delay can be calculated as $\textit{Time delay} = -\frac{b_1}{b_0}$.

The pseudocode illustrating the Auto-Regressive exogenous Method Implementation is outlined as follows:

Algorithm 5 ARX_Modeling_time_delay

```
1:  function ARXModelingTimeDelay(input_signals, output_signal, order)
2:      input_signals_array ← Convert to NumPy array(input_signals)
3:      output_signal_array ← Convert to NumPy array and flatten(output_signal)
4:      num_channels ← Number of channels in input_signals_array
5:      time_delays ← Array of zeros for time delays
6:      for channel ← 0 to num_channels − 1 do
7:          input_channel ← Extract channel(input_signals_array, channel)
8:          hankel_matrix ← Construct Hankel matrix(output_signal_array, order)
9:          input_matrix ← Construct input matrix(input_channel, order)
10:         augmented_matrix ← Concat input matrix with Hankel matrix (input_matrix, hankel_matrix)
11:         coefficients ← Solve for coefficients (augmented_matrix, output_signal_array, order)
12:         time_delays[channel] ← $-\frac{\text{coefficients}[0]}{\text{coefficients}[1]}$
13:     end for
14:     return Index of the minimum time delay across all channels
15: end function
```

Long Short Term Memory Method Implementation

The LSTM (Long Short-Term Memory) method is implemented to estimate and compensate time delay in signals using a recurrent neural network (RNN) architecture with LSTM units. These LSTM units are designed to capture long-term dependencies and to retain information over longer sequences, making them well-suited for time delay estimations. The LSTM network has been implemented with python, utilizing the TensorFlow and keras libraries with the goal of determining the time delay between input and output signals. The LSTM approach is compared with traditional methods, such as Cross-Correlation, Polynomial regression, Linear regression as well as the Auto-regressive with exogenous input model. Our implementation begins with the solution reading several datasets presented as comma-separated variable (CSV) files using pandas. These datasets are split into input and output signals based on the number of input and output defined as a list. The solution dynamically takes each input and output from all the CSV files passed in the list, utilizing proper list indexing. The input signals were normalized before model training, before an optimization process was then incorporated using the Python scipy library to refine the time delay estimates obtained from the LSTM implementation. The Scipy optimization was employed to minimize the defined objective function considering the squared differences between estimated and actual LSTM delays. The Long Short-Term Memory network has been tested on a test data set with a sample size of 1730 samples using 2 epochs to train the model. It shows that the method detected a linear relationship on the test data set between the input and output signals. This could possibly be due to the lack of sufficient data in training the model or the selection of the appropriate activation function during the training phase. The result from the test shows that the accuracy of the long short term memory is way lower compared to the cross correlation, linear regression, polynomial regression and ARX methods. Possible reasons could be the lack of substantial amount of data to effectively learn temporal patterns, or the inadequacy of the data to represent the underlying dynamics. This may lead to LSTMs struggling to generalize well. Additionally, LSTM models are complex and require many parameters. In situations where the relationship between the input signal and time delays is relatively simple, this complexity might lead to overfitting or inefficiency. The Pseudocode for the implementation of the Long short term memory is given below as follows:

Algorithm 6 LSTM Time Delay

```
1: function LSTM_TIME_DELAY($input\_signals$, $output\_signal$, $epochs = 100$,
   $batch\_size = batchSize$)
2:     $scaler\_input \leftarrow$ MinMaxScaler($feature\_range = (0, 1)$)
3:     $scaler\_output \leftarrow$ MinMaxScaler($feature\_range = (0, 1)$)
4:     $input\_signals\_scaled \leftarrow scaler\_input.fit\_transform(input\_signals)$
5:     $output\_signal\_scaled \leftarrow$
         $scaler\_output.fit\_transform(output\_signal.values.reshape(-1, 1))$
6:     $input\_signals\_reshaped \leftarrow$
         reshape($input\_signals\_scaled,$
         $(input\_signals\_scaled.shape[0], 1, input\_signals\_scaled.shape[1])$)
7:     $model \leftarrow$ Sequential()
8:     $model.add$(LSTM($units = 100, activation =' relu',$
         $input\_shape = (1, input\_signals\_scaled.shape[1])$)))
9:     $model.add$(Dense($units = 1$))
10:    $model.compile(optimizer =' adam',$
         $loss =' mean\_squared\_error')$
11:    $model.fit(input\_signals\_reshaped, output\_signal\_scaled,$
         $epochs = 4, batch\_size = batch\_size, verbose = 0)$
12:    $predicted\_output\_scaled \leftarrow$
         $model.predict(input\_signals\_reshaped)$
13:    $predicted\_output \leftarrow$
         $scaler\_output.inverse\_transform(predicted\_output\_scaled)$
14:    $time\_delay \leftarrow$
         cross_correlation(DataFrame($predicted\_output$),
         $output\_signal, sampling\_rate$)
15:    return $time\_delay$
16: end function
```

6.1.2. Computational Complexity

Time delay estimation algorithms may vary in computational complexity. Some methods may be computationally intense, especially when dealing with large datasets or real-time applications. In our study, we have considered the complexity of each method in assessing their accuracy and precision of time delay estimations. The computational complexity of the cross-correlation method depends on the length of the input and output signals. Calculating the cross-correlation for each input signal requires computing the dot product at different time lags, which has a complexity of $O(N)$, where N is the length of the signals. Therefore, the overall complexity of the cross-correlation method is $O(N * M)$, where M is the number of input signals. The computational complexity of polynomial regression depends on the degree of the polynomial and the number of input samples. The $numpy.polyfit$ function used for polynomial regression has a complexity of $O(d * N^2)$, where d is the degree of the polynomial and N is the number of input samples. The complexity can increase for higher-degree polynomials. Linear regression involves calculating the covariance and variance between the input and output signals. The computational complexity of calculating the covariance matrix using $numpy.cov$ is approximately $O(N^2)$, where N is the number of input samples. The complexity of calculating the variance using $numpy.var$ is $O(N)$. Therefore, the overall complexity of linear regression is $O(N^2)$. The complexity of ARX modeling depends on the number of input samples, the order of the autoregressive model, and the number of input channels. Constructing the Hankel matrix using scipy.linalg.hankel has a complexity of $O(nm)$, where n is the number of input samples and m is the order of the auto-regressive model. Concatenating the Hankel matrix and the input matrix has a complexity of $O(nm + nM)$, where M is the number of input channels. Solving for the coefficients using *numpy.linalg.lstsq* has a complexity of $O(nm^2)$. Therefore, the overall complexity of ARX modeling is $O(nm^2 + nM)$. The computational complexity of training LSTM neural networks depends on the number of input samples, the number of LSTM layers, the number of units in each layer, and the number of training epochs. Constructing the LSTM model has a complexity of $O(NLU)$, where N is the number of input samples, L is the number of LSTM layers, and U is the number of units in each layer. Training the model involves forward and backward propagation for each epoch, resulting in a complexity of approximately $O(N * E)$, where E is the number of training epochs. The overall complexity of LSTM neural networks can be high, especially for large datasets and complex network architectures.

6.1.3. Optimization Results

The solution outputs detailed information about the estimated time delays for each method. During execution of the test bench using several test dataset, the objective function calculates a score for each method based on the difference between the estimated time delay and a target delay, the target delay represents the desired or known time delay that needs to be compensated from the simulated test data. The solution utilizes the scipy.optimize.minimize function to minimize the total score and find the best optimized time delays for each method. This function implements various optimization algorithms, such as Nelder-Mead, Powell, CG, BFGS. The optimization process involves iteratively adjusting the time delays for each method and evaluating the objective function. The goal is to find the combination of time delays that minimizes the total score. The optimization algorithm explores different combinations of time delays and updates them based on the objective function's feedback. By comparing the estimated delays with the optimized delays, the effectiveness of the optimization can be assessed. Based on the optimization results, the solution provides conclusions about the method with the best time delay estimation and compensation performance. The method with the lowest total score, indicating the closest approximation to the target delay, is considered the best-optimized method. Conclusively, the optimization results are crucial in selecting the most accurate and reliable method for time delay estimation and compensation by systematic evaluation and comparison with different methods and fine-tuning the time delays through optimization. The solution helps in identifying the best approach that yields the best results for a specific dataset. It is also important to note that the optimization results may vary depending on factors such as the dataset, the optimization algorithm used, and the specific implementation details. The mathematical formulation of the objective function is given as:

$$\phi(\tau_{crosscor}, \tau_{poly}, \tau_{linear}, \tau_{ARX}, \tau_{lstm}) = \sum_{i=1}^{N} (\tau_i - \widehat{\tau_i})^2, \tag{6.3}$$

Wwhere N is the number of methods, τ_i is the actual time delay for the $i-th$ method and $\widehat{\tau_i}$ is the estimated time delay. The pseudocode outlining the objective function and scipy optimization usage is given as follows:

Algorithm 7 Objective Function

```
1:  function ObjectiveFunction(params)
2:      # Extract parameters for optimization
3:      delay_CC, delay_poly, delay_lin, delay_ARX ← params
4:      # Calculate squared differences between estimated and actual delays
5:      squared_diff ←
          [ (delay_CC − find_time_delay(input_signals, output_signal, rate))^2,
            (delay_poly − poly_reg_time_delay(input_signals, output_signal, deg))^2,
            (delay_lin − lin_reg_time_delay(input_signals, output_signal))^2,
            (delay_ARX − arx_time_delay(input_signals, output_signal, ord))^2 ]
6:      # Sum of scores, aiming to minimize the total score
7:      return sum(squared_diff)
8:  end function
9:  # Initial guesses for the time delays
10: init_guesses ← [0, 0, 0, 0]
11: # Minimize the objective function using SciPy
12: result ← minimize(objective_function, init_guesses)
13: # Get the optimized time delays
14: opt_delay_CC ← result.x[0]
15: opt_delay_poly ← result.x[1]
16: opt_delay_lin ← result.x[2]
17: opt_delay_ARX ← result.x[3]
```

7. Data used for Validation

Real-world datasets, especially in areas like NOX measurements, are often fraught with complexities and inconsistencies. Factors such as equipment flushing, warm-up periods, changes in operation points, and other unforeseen variables can introduce uncertainties. In gas transport systems and those affected by thermal inertia, these factors cause significant delays between input and output data. Accurately determining these delays is crucial for system performance and interpreting its behavior effectively.

7.1. Simulated Data for Evaluation

To tackle the challenges posed by real-world data, we employed a simulated dataset. The primary advantage of simulated data lies in its controlled environment. Unlike real-world data, where numerous variables can introduce inconsistencies, simulated data is crafted to represent specific scenarios with precision. This controlled representation ensures that the delay between input and output is both definitive and consistent, allowing for more reliable analyses and methodological testing.

To validate our methods' accuracy, we generated simulated output data from real input data using a Recurrent Neural Network (RNN). For this purpose, we utilized 10 different real measurement runs. These datasets ranged from 1740 to 8540 columns, with 4 to 8 inputs and outputs ranging from 2 to 8. Outputs analyzed included:

- PC
- T_41
- T_31
- NOX
- THC
- CO2
- CO
- IMEPC
- mf_fuel

For each simulated output, we tested the originally simulated data and also added noise and filters to the signal to ensure the robustness of the methods. We then manually shifted the signals in the range of 1 to 10 seconds, generating over 10,000 different output variations to test robustness.

Data Augmentation

Data augmentation was performed on the data used for testing. Artificially increasing the size and diversity of the data set through various transformations helps to increase the robustness and generalizability of the methods. Time shifts introduce variations in the temporal alignment of signals, helping the model to be more robust to timing discrepancies in real-world data. The data used for this study contains different time shifts, such as 2, 4, 6, 8, and 10-second shifts. Noise was added to each of the time-shifted signals in order to simulate real-world data measurements. Additive noise helps the model become more robust to real-world data, leading to better generalization. Finally, each time-shifted and noise-added signal was further filtered to smooth out noise and highlight important features. The reason for filtering is to help in capturing the true underlying patterns of the signals by eliminating unwanted noise, enhancing the model's ability to learn from clean data. After shifting the data 2 seconds apart and applying the aforementioned filters and noise, the number of test cases was significantly increased. Initially, each dataset contained N test cases; data augmentation expanded the dataset to include multiple variations of each original test case. More specifically, each original test case was augmented by applying each time shift (2 sec, 4 sec, 6 sec, 8 sec, 10 sec) to generate a new test time series which in turn was then augmented with either noise or filtering or both. Thus, for each original test case, we generated: The original signal, the 2-sec shift, 2-sec shift with noise, 2-sec shift with filter, 2-sec shift with noise and filter, the 4-sec shift, 4-sec shift with noise, 4-sec shift with filter, 4-sec shift with noise and filter, the 6-sec shift, 6-sec shift with noise, 6-sec shift with filter, 6-sec shift with noise and filter, the 8-sec shift, 8-sec shift with noise, 8-sec shift with filter, 8-sec shift with noise and filter, the 10-sec shift, 10-sec shift with noise, 10-sec shift with filter, 10-sec shift with noise and filter. Therefore, the total number of test cases generated after augmentation with respect to each data set will be

$$\#test\_case = N + N * 20, \tag{7.1}$$

where N is the number of original outputs before augmentation. The table below gives a summary of every data set used in the implementation.

Table 7.1.: Data structure and description

dataset	number of rows	number of columns	number of input	number of outputs N	test cases
csv 1	1730	46	4	2	42
csv 2	1730	46	4	2	42
csv 3	1736	46	4	2	42
csv 4	7398	176	8	8	168
csv 5	5744	176	8	8	168
csv 6	8530	152	5	7	147
csv 7	5740	174	6	8	168
csv 8	3805	174	6	8	168
csv 9	6791	133	7	6	126
csv 10	5686	174	6	8	168

7.2. Graphical Representations

Visual analyses were conducted on the simulated datasets, providing insights into the various NOX data variations. Graphs depicting original NOX data alongside its variations, such as noise-added data, delayed data, and filtered data, showcased the intricacies of each modification. These visualizations emphasized the consistency of the known delay across different data scenarios and highlighted the precision with which the data was simulated.

7.2.1. NOX Variations

Lastly, the NOX data variations are visualized over a concise timeframe, providing a granular comparison between the original and its simulated variations.

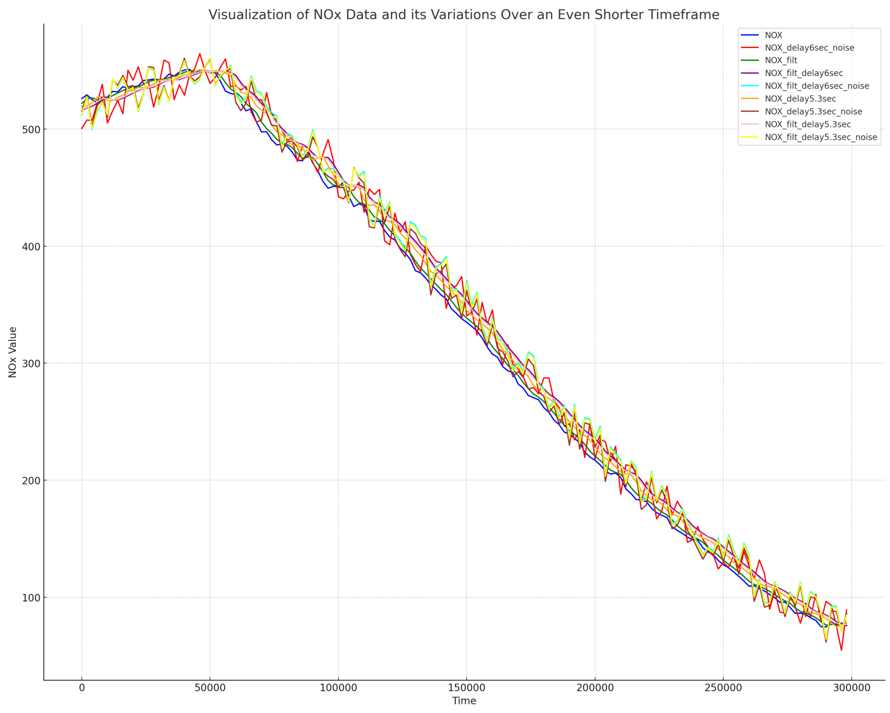

Figure 7.1.: Variations of NOX Data for Testing

7.3. Output Data Analysis

The following figure illustrates the variations in the NOX output data. Specifically, it includes the original NOX data, the NOX data with added noise, and the filtered NOX data. This comparison allows us to observe the effects of noise and filtering on the data and assess the robustness of the methods used.

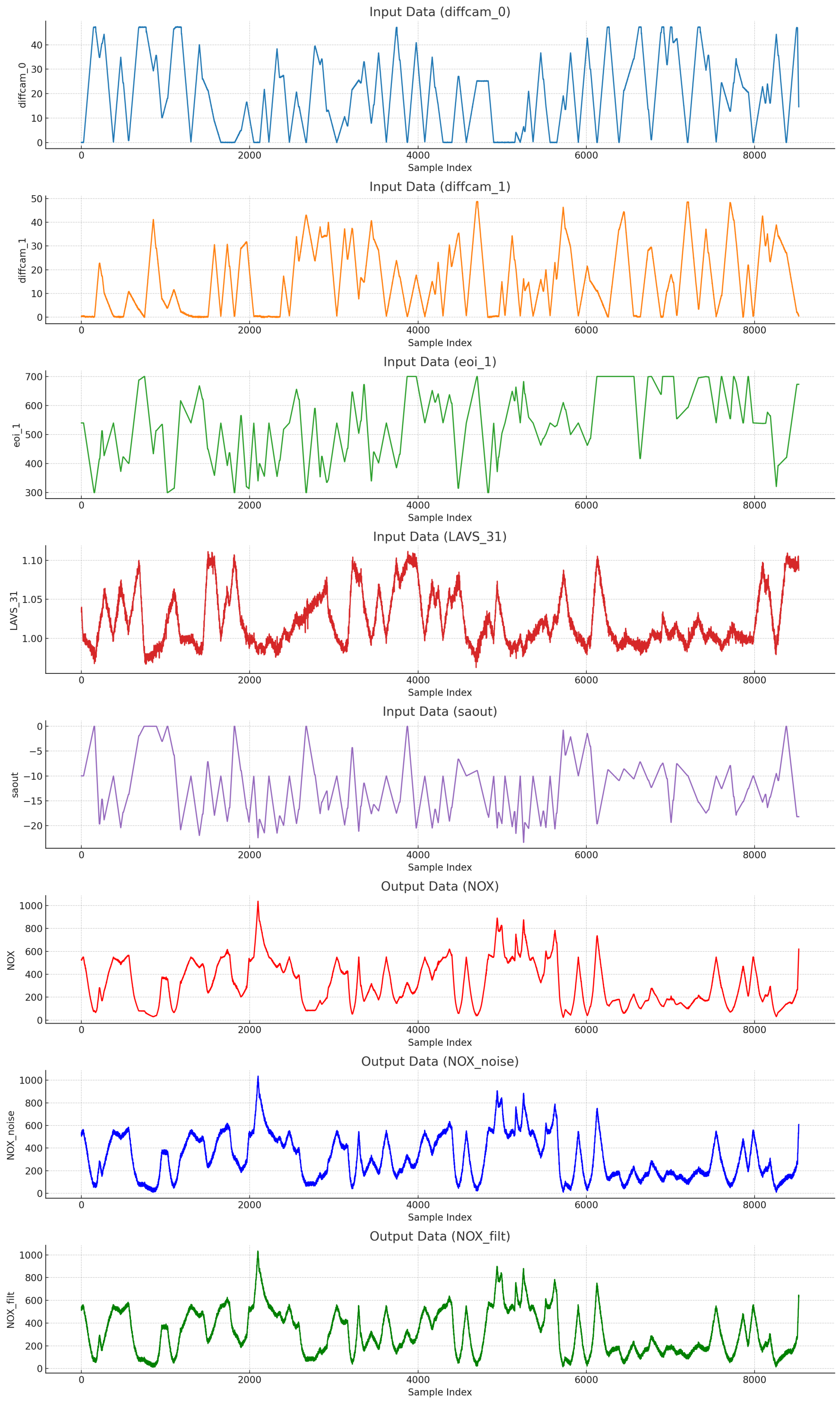

Figure 7.2.: Output Data Variations: Original NOX, NOX with Noise, and Filtered NOX

The NOX output data provides a clear depiction of the signal variations under different conditions. The original NOX data represents the raw measurements, while the NOX with noise includes random fluctuations to simulate real-world imperfections. The filtered NOX data demonstrates the effectiveness of the applied filters in smoothing out these fluctuations.

7.4. Conclusion

The intricacies of real-world systems, particularly those involving delays due to gas transport and thermal inertia, necessitate methodologies that can accurately determine these delays. A simulated dataset, complemented with graphical representations and crafted with precision, becomes an invaluable tool in this endeavor. It offers a controlled environment for testing and ensures the methodologies developed are robust and reliable.

8. Evaluation and Results

The aim of implementing delay estimation both before and after optimization is to compare the performance of different methods and to demonstrate the crucial role of the optimization process in time delay estimation and compensation. Additionally, we analyze the effect of noise and filtering on the performance of the methodology. Furthermore, we examine whether certain output signals perform better and if there are significant performance differences between different datasets.

The description and rationale of the implemented methods have been presented in Table 8.1, evaluating the strengths and weaknesses of each method and their performance in estimating and compensating time delay. After the development of the solutions for each method, they were tested using a series of simulated data presented in CSV files and executed through a generated test bench. The solution execution commences by reading each dataset sequentially, taking all input and output signals into consideration with respect to each dataset. For testing, ten different datasets have been utilized, and the data structure, which was previously discussed in the previous Chapter.

Method	**Description**	**Rationale**
Cross Correlation	Measures the similarity between two or more signals as a function of the time-lag applied to the signals.	Ideal for identifying time delays and synchronization between input and output signals.
Polynomial Regression	Fits a polynomial equation to the data points.	Useful for modeling non-linear relationships between the input and output signals.
Linear Regression	Fits a linear equation to the data points.	Simple and effective for modeling linear relationships between the input and output signals.
ARX (AutoRegressive with eXogenous inputs)	A time-series model that uses past values of the output and current/past values of the input to predict the time delay.	Captures the dynamic relationship between input and output signals, taking into account past values and external inputs.
Long Short Term Memory (LSTM)	A type of recurrent neural network (RNN) that can learn long-term dependencies and patterns in sequential data.	Excellent for handling time-series data and capturing complex temporal patterns that may span over long sequences.

Table 8.1.: Method description and Rationale

8.1. Evaluation and Results

The implemented methods were tested using a series of simulated data. The test bench processes each dataset sequentially, using a loop operation that considers all input and output signals for each data frame. The methods analyze the input signals and perform time delay analysis for the corresponding output signal, as represented in the pseudo-code given in Algorithm 7.

First, we examine the results by analyzing the percentage of times each method, both optimized and non-optimized, achieved a correct result, defined as a deviation of 0. The results are summarized in Table 8.2 and visualized in Figure 8.1.

In the first approach, we evaluate the frequency, in percentage terms, with which the methodology returns exactly the correct results. Here, we consider the unoptimized delay returned by the methodology, showing that the Linear Regression method returned 11.3% correct identification, followed by the Polynomial Regression method with 9.6%, the ARX Modeling method with 6.5%, the Cross-Correlation method with 6.7%, and the LSTM method with 4.8%.

We will also review the outcomes of the optimized approach. The Linear Regression Optimized method returned 27.2% correct identification of the optimized delay. This was followed by the Polynomial Regression Optimized method with 23.5%, the ARX Modeling Optimized method with 22.7%, the Cross-Correlation Optimized method with 22.0%, and the LSTM Optimized method with 4.8%.

The analysis reveals that optimized methods generally perform better in achieving zero deviation compared to non-optimized counterparts. For instance, the Cross-Correlation Optimized method has 32.97% correct results, compared to 6.70% for its non-optimized version. Similarly, Polynomial Regression Optimized and Linear Regression Optimized methods show substantial improvements with 27.87% and 22.54% correct results, respectively.

Non-optimized methods like Linear Regression and Polynomial Regression achieve 11.31% and 9.61% correct results, respectively. However, the optimized versions demonstrate that optimization can significantly enhance performance and reliability. The box plot analysis further indicates that optimized methods have lower mean deviations and less spread, indicating more consistent performance.

	Cross Correlation	Polynomial Regression	Linear Regression	ARX Modeling	LSTM
not optimized	6.70 ± 2.79	9.61 ± 2.36	11.31 ± 2.76	6.46 ± 2.62	4.77 ± 3.04
optimized	21.97 ± 2.86	23.51 ± 2.31	27.22 ± 2.84	22.70 ± 2.70	4.77 ± 3.04

Table 8.2.: Percentage of Zero Deviation and Standard Deviation for Optimized and Non-Optimized Methods

Graphically, the percentages of the correct identification of the optimal delays obtained from the solutions can be represented as follows:

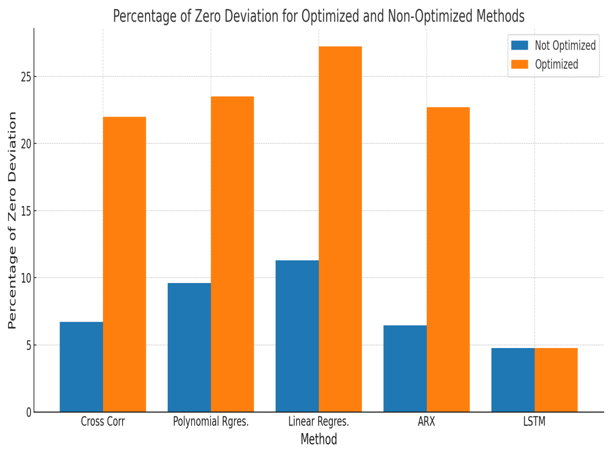

Figure 8.1.: Percentage of accurate identification of optimal delay by each method

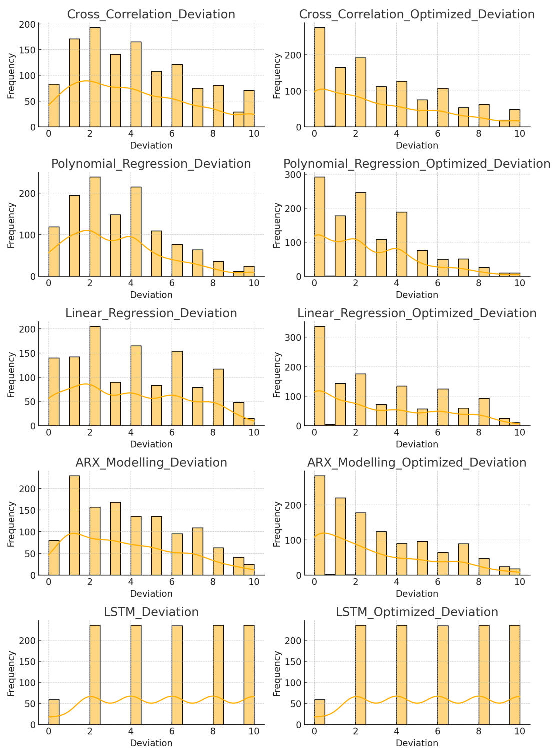

Figure 8.2.: Absolute Delay Distribution for each Method

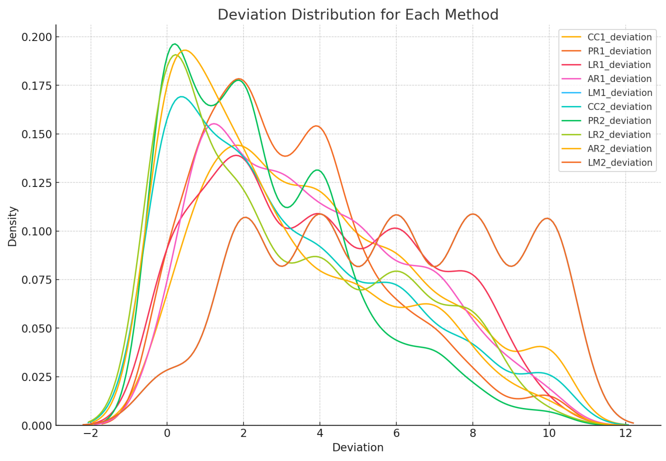

Figure 8.3.: Visual Comparison of Delay Density for each Method

Figure 8.2 displays the distribution of delay deviations between the predicted values obtained from various methodologies and the actual values. Each subfigure represents the deviation distribution before and after optimization for the respective methods.

Figure 8.3 provides a visual comparison of these deviation distributions across all methods, illustrating the density of deviations to evaluate the performance and accuracy of each methodology. The comparison highlights how the optimization process affects the distribution of deviations across different techniques.

Overall, the choice of method and application of optimization techniques significantly impact the accuracy and reliability of results. These findings highlight the importance of selecting appropriate methods and applying optimization techniques to achieve better performance in data analysis tasks. The calculated time delays from the two solutions were compared, the cross-correlation method gave higher absolute results but polynomial regression gave better results overall showing to be the more robust method.

8.1.1. Deviation Distribution

In this section, we analyze the deviation distribution for each method, considering both optimized and non-optimized approaches, using box plots. The key statistics for these box plots—such as count, mean, standard deviation, minimum, quartiles, interquartile range (IQR), and whiskers—are summarized in Table 8.3. Figure 8.4 provides a visual representation of these statistics, allowing for a clearer comparison of the deviation patterns across different methods.

While we previously examined the percentage of instances where each method accurately detected the exact result, it became evident that optimization played a crucial role in enhancing the methodology. However, this data alone does not provide a complete picture of the reliability and robustness of each method. Therefore, we extend our analysis by examining how far each method deviates from the correct result. In practical applications, particularly in our use case, a method that consistently produces results close to the correct value, even with a reasonable deviation, remains valuable. This further analysis allows us to assess not only the precision of each method but also its overall stability and dependability in varied scenarios.

Table 8.3.: Statistics for Deviation Distributions in seconds

Method	Count	Mean	Std	MAE	MSE	RMSE	Min	25%	50%	75%	Max	IQR	Lower Whisker	Upper Whisker
Cross Correlation	1238	3.21	3.76	4.08	24.41	4.94	0.0	2.0	4.0	6.0	10.0	4.0	-4.0	12.0
Polynomial Regression	1238	1.82	3.61	3.29	16.34	4.04	0.0	1.0	3.0	5.0	10.0	4.0	-5.0	11.0
Linear Regression	1238	3.55	3.26	3.95	23.25	4.82	0.0	2.0	4.0	6.0	10.0	4.0	-4.0	12.0
ARX Modelling	1238	3.09	3.45	3.81	21.43	4.63	0.0	2.0	3.0	6.0	10.0	4.0	-4.0	12.0
LSTM	1238	5.71	3.04	5.71	41.91	6.47	0.0	4.0	6.0	8.0	10.0	4.0	-2.0	14.0
Cross Correlation Optimized	1238	2.63	2.04	2.13	24.41	4.94	0.0	0.0	2.0	4.0	6.0	4.0	-6.0	10.0
Polynomial Regression Optimized	1238	1.52	2.28	2.44	16.34	4.04	0.0	0.0	2.0	4.0	8.0	4.0	-6.0	10.0
Linear Regression Optimized	1238	2.78	2.33	2.92	23.25	4.82	0.0	1.0	3.0	5.0	8.0	4.0	-5.0	9.0
ARX Modelling Optimized	1238	2.39	2.71	3.12	21.43	4.63	0.0	1.0	3.0	5.0	10.0	4.0	-4.0	11.0
LSTM Optimized	1238	5.71	2.87	4.05	41.91	6.47	0.0	2.0	4.0	6.0	10.0	4.0	-2.0	12.0

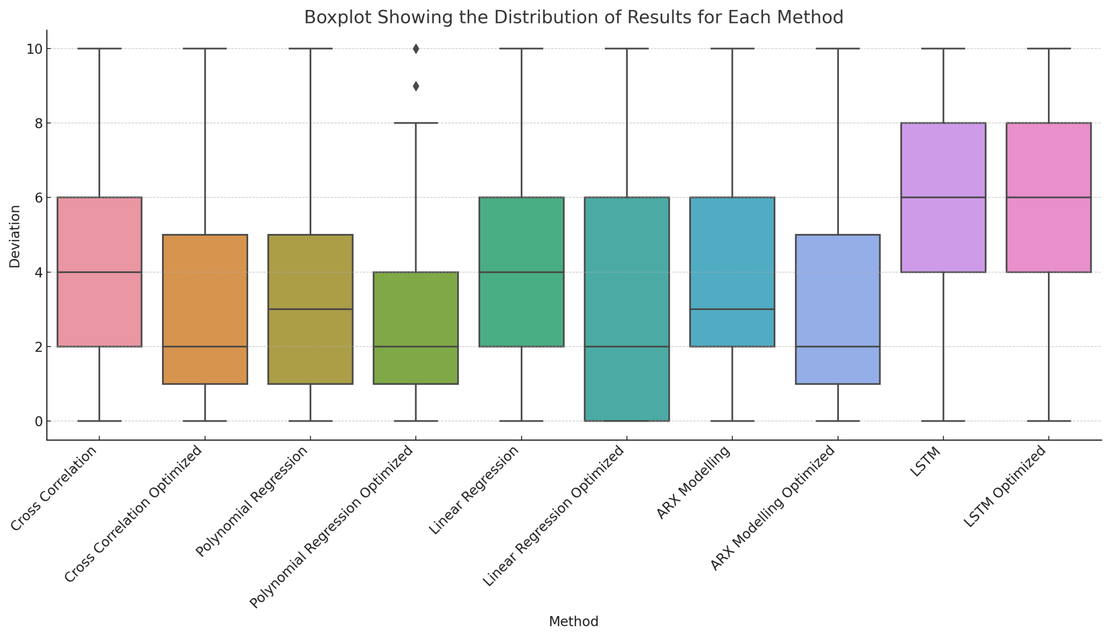

Figure 8.4.: Deviation Distribution for Each Method in seconds

The Table 8.3 and Figure 8.4 provide an overview of the statistical metrics for different methods, including both standard and optimized versions, applied to a dataset of 1,238 observations.

8.1.2. General Observations

Optimized vs. Non-Optimized Methods: Across most methods, optimization results in a notable reduction in the Mean, although the mean does not become zero in all optimized cases anymore. This suggests that while optimization reduces systematic bias, some residual bias remains in certain models.

8.1.3. Error Metrics (MAE, MSE, RMSE)

In general, optimized methods continue to demonstrate improved performance compared to their non-optimized versions. However, the degree of improvement varies by method.

- **MAE**: The method with the lowest MAE is now the *Polynomial Regression Optimized* at 2.44, surpassing the previous leader, *Cross Correlation Optimized*. The *LSTM Optimized* method continues to have the highest MAE (4.05) among the optimized models, indicating ongoing challenges in minimizing absolute errors.
- **MSE and RMSE**: The MSE and RMSE patterns remain similar, with optimized methods outperforming non-optimized ones. However, the *LSTM* method, both standard and optimized, maintains the highest variability in error (MSE of 41.91 and RMSE of 6.47), signaling that this method still struggles with error consistency.

8.1.4. Spread and Dispersion (Standard Deviation and IQR)

Standard Deviation (Std): Post-optimization, the standard deviation decreases for most methods, reflecting reduced error variability. Notably, the *LSTM*

Optimized method's standard deviation is 2.87, still higher than other methods, indicating less consistency.

Interquartile Range (IQR): The IQR remains consistent at 4.0 across nearly all methods, showing that the spread of the middle 50% of the data is unaffected by optimization, suggesting that improvements are mostly in error reduction rather than data spread.

8.1.5. Distribution Extremes (Min, Max, Lower Whisker, Upper Whisker)

Minimum and Maximum: The minimum value remains at 0.0 across all methods, demonstrating that perfect predictions are occasionally achieved. The maximum values have generally decreased after optimization, with a significant reduction observed in methods like *Cross Correlation Optimized* and *Polynomial Regression Optimized*, suggesting fewer extreme errors.

Whiskers: The optimized models, particularly *Linear Regression* and *ARX Modelling*, show less extreme whisker values, indicating fewer outliers after optimization. However, the *LSTM* method continues to produce wider whiskers, reflecting its propensity for generating significant outliers.

8.1.6. Specific Method Observations

Cross Correlation: This method, especially in its optimized form, continues to exhibit strong performance with low MAE and MSE. The updated table shows that the *Cross Correlation Optimized* still outperforms many other methods in error metrics, underlining its effectiveness.

Polynomial and Linear Regression: These methods benefit notably from optimization, with significant reductions in MAE, MSE, and RMSE. The performance of these models is now more closely aligned, indicating that both approaches similarly capitalize on optimization techniques.

ARX Modelling: Post-optimization, this method remains competitive, particularly in MAE and RMSE metrics, where it surpasses *Polynomial* and *Linear Regression*.

LSTM: Despite optimization, *LSTM* continues to have the highest errors. The updates suggest that while optimization helps, the LSTM model may require further tuning or alternative strategies to reduce error and variability to levels seen in simpler models.

Conclusion

Optimization remains a crucial step in enhancing model performance, as evidenced by improvements in key metrics like Mean, MAE, MSE, and RMSE. However, the extent of these improvements varies by method, with some like *Cross Correlation* and *ARX Modelling* benefiting more than others. The *LSTM* method, in particular, still lags behind despite optimization, indicating a need for further refinement. Overall, the updated table highlights the significant impact of optimization on predictive accuracy and the importance of model selection in achieving robust predictions. This updated LaTeX text incorporates the changes in data values while maintaining the original

8.1.7. Effect of Noise and Filter on Methods

The presence of noise generally increases the mean deviation for most methods. Figure 8.5 shows the mean deviations with and without noise for each method. From the chart, it is evident that noise has a significant impact on the performance of the methods. For example, the mean deviation for Polynomial Regression increases from 3.31 to 3.26 when noise is introduced.

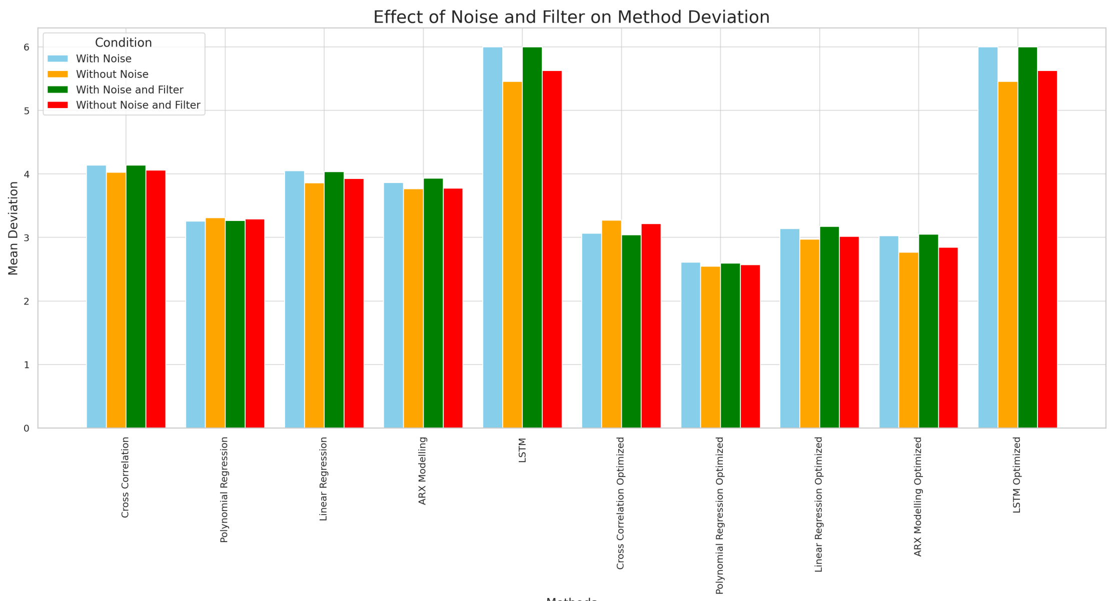

Figure 8.5.: Effect of Noise and Filter on Method Deviation Barcharts in seconds

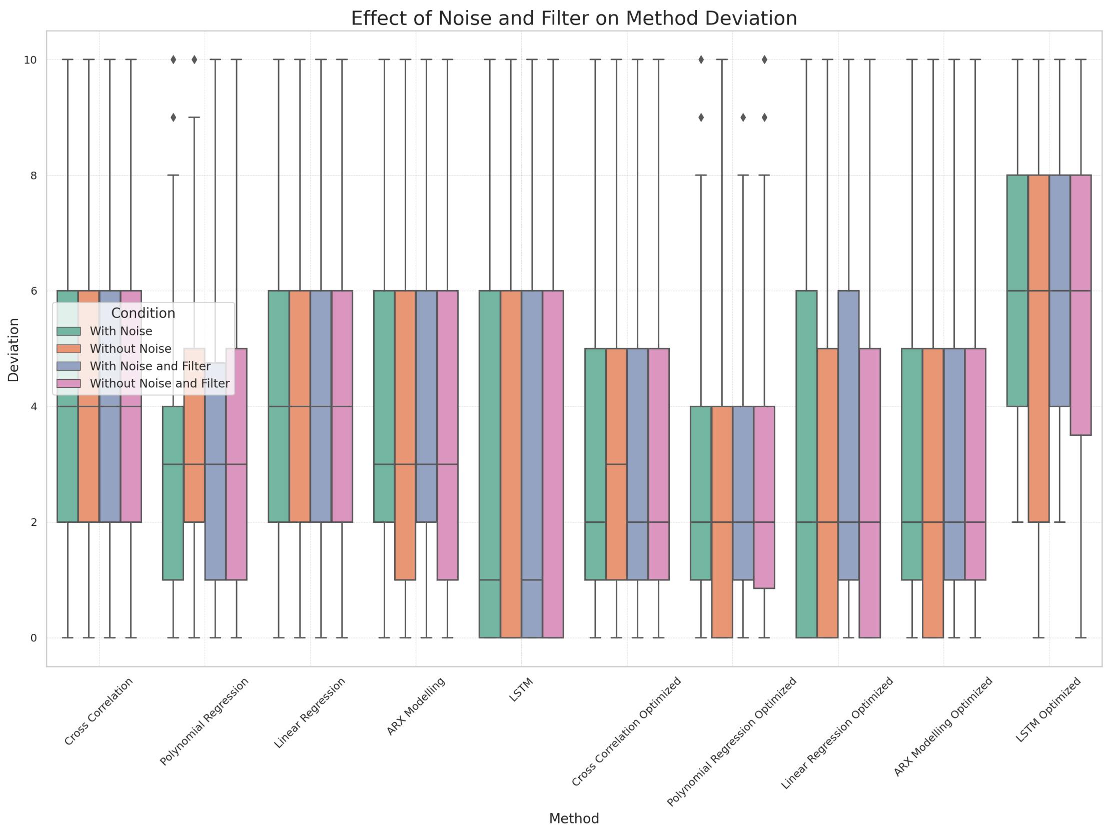

Figure 8.6.: Effect of Noise and Filter on Method Deviation Boxplots in seconds

Applying a filter tends to reduce the deviation for most methods, as shown in Figure 8.6. The presence of both noise and filter still results in lower deviations compared to having noise alone. For instance, the LSTM method's deviation increases from 5.45 without noise to 6.00 with noise and filter.

The combined effect of noise and filter on method deviation is shown in Table 8.4. The table presents the mean deviations for each method under four conditions: with noise, without noise, with both noise and filter, and without both noise and filter. These results demonstrate that optimized methods show smaller increases in deviation with noise and benefit more from filtering compared to non-optimized methods.

Table 8.4.: Effect of Noise and Filter on Method Deviation in seconds (Mean ± Standard Deviation)

Method	Vanilla (Mean ± Std)	With Filter (Mean ± Std)	With Noise (Mean ± Std)	With Both (Mean ± Std)
Cross Correlation	3.92 ± 2.76	4.15 ± 2.78	4.14 ± 2.80	4.14 ± 2.78
Polynomial Regression	3.36 ± 2.32	3.26 ± 2.37	3.26 ± 2.37	3.27 ± 2.38
Linear Regression	3.72 ± 2.67	4.03 ± 2.79	4.05 ± 2.80	4.03 ± 2.80
ARX Modeling	3.60 ± 2.59	3.95 ± 2.63	3.87 ± 2.65	3.93 ± 2.65
LSTM	5.00 ± 3.42	6.00 ± 2.83	6.00 ± 2.83	6.00 ± 2.84
Cross Correlation Optimized	3.23 ± 2.93	3.18 ± 2.84	3.07 ± 2.83	3.04 ± 2.84
Polynomial Regression Optimized	2.53 ± 2.45	2.58 ± 2.23	2.61 ± 2.23	2.59 ± 2.14
Linear Regression Optimized	2.99 ± 2.86	3.06 ± 2.81	3.14 ± 2.85	3.17 ± 2.82
ARX Modeling Optimized	2.66 ± 2.73	2.98 ± 2.71	3.03 ± 2.70	3.05 ± 2.76
LSTM Optimized	5.00 ± 3.42	6.00 ± 2.83	6.00 ± 2.83	6.00 ± 2.84

Based on the analysis, the Polynomial Regression Optimized method performs the best. This method consistently shows the lowest mean deviation across all conditions—whether noise and filters are present or not. The effectiveness of this method can be attributed to its ability to effectively handle variations and optimize the correlation, resulting in minimal deviation.

The analysis indicates that noise increases the mean deviation for most methods, but applying a filter can mitigate this effect to some extent. Optimized methods generally perform better and show more consistent results. The choice of method and the application of optimization and filtering techniques can significantly impact the accuracy and reliability of the results.

8.1.8. Analysis of Deviation Across different Signals and Methods

The mean deviations for each variable, representing different types of measurement signals that sometimes appear in multiple datasets, are presented in Table 8.6. Knowing what kinds of signals are easier to detect and with which method is very important for the company. While the results have been anonymized to avoid revealing which signals are easier to detect, the data clearly suggests that some signals are indeed easier to identify than others. Figure 8.7 visually compares the mean deviations across different methods and variables, helping to determine whether certain types of signals, due to their inherent characteristics, are more or less easily detected by different methods.

Table 8.5.: Mean ± Standard Deviation for Each Variable and Method in seconds

Variable	Cross Correlation (Mean ± Std)	Cross Correlation Optimized (Mean ± Std)	Polynomial Regression (Mean ± Std)	Polynomial Regression Optimized (Mean ± Std)	Linear Regression (Mean ± Std)	Linear Regression Optimized (Mean ± Std)	ARX Modeling (Mean ± Std)	ARX Modeling Optimized (Mean ± Std)	LSTM (Mean ± Std)	LSTM Optimized (Mean ± Std)	Overall (Mean ± Std)
variable1	4.58 ± 3.76	2.82 ± 2.10	3.14 ± 3.50	2.43 ± 2.25	4.03 ± 3.30	2.98 ± 2.45	3.93 ± 3.60	2.88 ± 2.80	5.71 ± 3.10	5.71 ± 2.95	4.12 ± 3.28
variable2	4.45 ± 3.80	3.41 ± 2.05	3.28 ± 3.55	2.82 ± 2.30	4.12 ± 3.25	3.51 ± 2.40	3.85 ± 3.55	3.04 ± 2.85	5.71 ± 3.05	5.71 ± 2.90	4.17 ± 3.24
variable3	4.00 ± 3.70	3.20 ± 2.15	3.50 ± 3.60	2.90 ± 2.20	4.10 ± 3.35	3.60 ± 2.35	3.90 ± 3.50	2.85 ± 2.75	5.70 ± 3.00	5.70 ± 2.85	4.15 ± 3.24
variable4	3.90 ± 3.65	3.15 ± 2.10	3.45 ± 3.65	2.85 ± 2.25	4.00 ± 3.40	3.55 ± 2.50	3.85 ± 3.65	2.80 ± 2.90	5.60 ± 3.15	5.60 ± 2.90	4.14 ± 3.27
variable5	3.83 ± 3.85	3.20 ± 2.05	3.49 ± 3.50	3.14 ± 2.20	4.21 ± 3.30	3.15 ± 2.35	3.52 ± 3.55	2.93 ± 2.70	5.71 ± 3.10	5.71 ± 2.95	4.29 ± 3.26
variable6	3.22 ± 3.70	3.75 ± 2.10	3.07 ± 3.40	2.36 ± 2.35	3.59 ± 3.25	3.02 ± 2.50	3.85 ± 3.45	2.81 ± 2.85	5.71 ± 3.05	5.71 ± 2.85	3.91 ± 3.25
variable7	2.94 ± 3.75	3.22 ± 2.15	2.78 ± 3.45	2.02 ± 2.40	3.50 ± 3.20	2.97 ± 2.45	3.99 ± 3.50	2.86 ± 2.95	5.71 ± 3.00	5.71 ± 2.80	3.92 ± 3.23

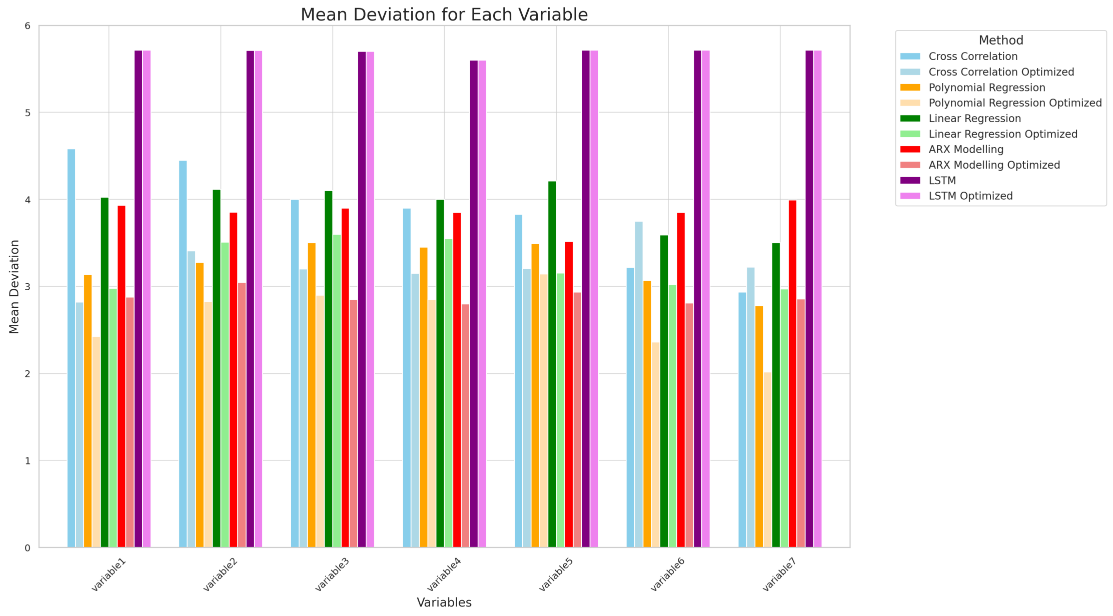

Figure 8.7.: Mean Deviation for Each Variable and Method in seconds

The deviation distribution for each variable and method is illustrated in Figure 8.10. This box plot highlights the spread and central tendency of deviations, providing insights into the variability of each method's performance.

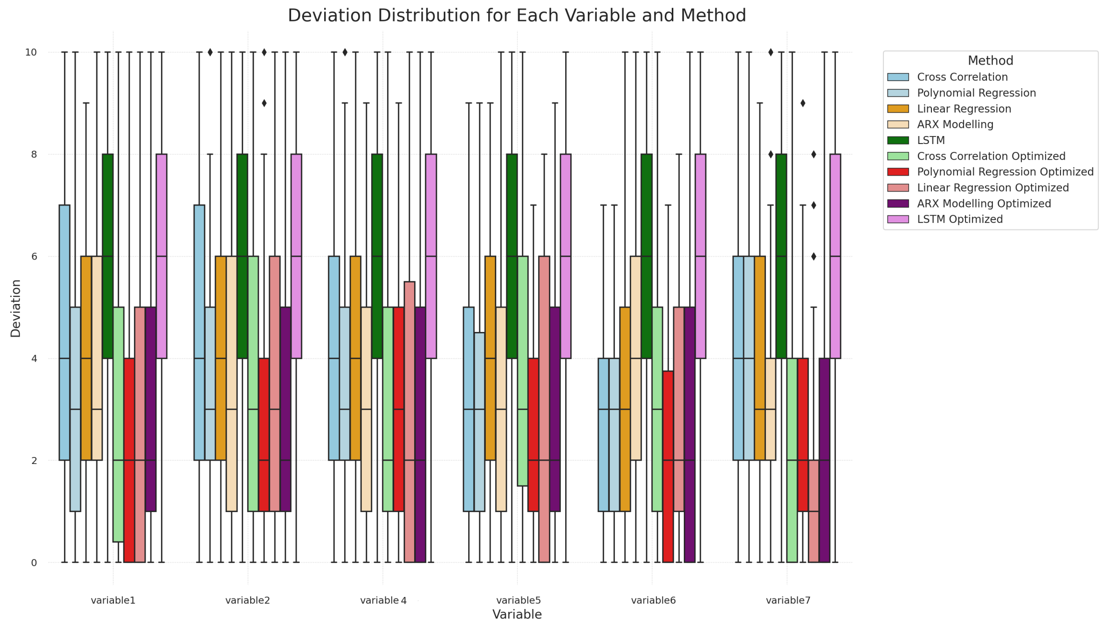

Figure 8.8.: Deviation Distribution for Each Variable and Method in seconds

The analysis reveals several key insights:

- **Best Performing Variables:**
    - *variable7* generally shows the lowest deviations across most methods, indicating more stable performance.
    - Optimized methods such as *Cross Correlation Optimized* and *Polynomial Regression Optimized* perform particularly well with *variable7*, demonstrating minimal deviation.
- **Impact of Optimization:**
    - Optimization significantly reduces deviations across all variables, highlighting the importance of optimization techniques in improving method performance.
- **Method Variability:**
    - The box plot in Figure 8.10 shows that optimized methods not only have lower mean deviations but also exhibit less variability, suggesting more consistent performance.
    - Non-optimized methods, while sometimes achieving lower deviations, often show greater variability, which may affect reliability.

Overall, these findings emphasize the importance of selecting the appropriate method and applying optimization techniques to achieve better accuracy and reliability in data analysis tasks. Even though some signals seem to be easier to detect, there appears to be no significant difference between methods; however, some inherently result in lower deviations and more stable outcomes.

8.1.9. Performance Analysis of Datasets

The overall mean deviations for each file are presented in Table 8.6. Figure 8.9 visually compares the overall mean deviations across different files.

Table 8.6.: Overall Mean ± Standard Deviation for Each File in seconds

Filename	Cross Correlation (Mean ± Std)	Polynomial Regression (Mean ± Std)	Linear Regression (Mean ± Std)	ARX Modeling (Mean ± Std)	LSTM (Mean ± Std)	Cross Correlation Optimized (Mean ± Std)	Polynomial Regression Optimized (Mean ± Std)	Linear Regression Optimized (Mean ± Std)	ARX Modeling Optimized (Mean ± Std)	LSTM Optimized (Mean ± Std)	Overall Mean (Mean ± Std)
ds5.csv	3.851 ± 3.70	2.851 ± 3.55	3.565 ± 3.20	3.482 ± 3.40	5.714 ± 2.90	2.147 ± 2.10	2.264 ± 2.20	2.321 ± 2.30	2.086 ± 2.60	5.714 ± 2.80	3.400 ± 3.10
ds9.csv	3.754 ± 3.80	2.976 ± 3.60	4.024 ± 3.25	3.492 ± 3.45	5.714 ± 3.00	2.797 ± 2.15	2.857 ± 2.25	2.720 ± 2.35	2.741 ± 2.65	5.714 ± 2.85	3.679 ± 3.15
ds4.csv	3.363 ± 3.65	2.470 ± 3.50	4.310 ± 3.15	3.940 ± 3.35	5.714 ± 3.10	2.905 ± 2.05	2.333 ± 2.15	3.060 ± 2.25	3.024 ± 2.55	5.714 ± 2.75	3.679 ± 3.05
ds8.csv	4.071 ± 3.85	3.290 ± 3.70	3.833 ± 3.30	3.633 ± 3.50	5.714 ± 3.20	2.786 ± 2.25	2.702 ± 2.35	3.321 ± 2.45	2.548 ± 2.75	5.714 ± 2.95	3.761 ± 3.25
ds7.csv	3.971 ± 3.75	3.651 ± 3.65	3.929 ± 3.20	3.893 ± 3.40	5.714 ± 3.05	3.727 ± 2.10	2.548 ± 2.20	2.893 ± 2.30	2.705 ± 2.50	5.714 ± 2.70	3.875 ± 3.15
ds6.csv	4.218 ± 3.90	3.374 ± 3.75	4.082 ± 3.40	3.810 ± 3.55	5.714 ± 3.30	2.912 ± 2.35	2.735 ± 2.45	3.102 ± 2.55	3.299 ± 2.85	5.714 ± 3.05	3.896 ± 3.35
ds1.csv	5.561 ± 3.85	3.390 ± 3.55	4.610 ± 3.30	4.780 ± 3.45	5.707 ± 3.20	2.344 ± 2.10	1.537 ± 2.20	2.561 ± 2.30	3.317 ± 2.65	5.707 ± 2.85	3.951 ± 3.15
ds3.csv	5.714 ± 4.00	3.381 ± 3.70	3.690 ± 3.35	4.643 ± 3.50	5.714 ± 3.40	2.905 ± 2.20	2.333 ± 2.30	4.762 ± 2.40	3.167 ± 2.70	5.714 ± 2.90	4.202 ± 3.35
ds10.csv	4.089 ± 3.80	4.250 ± 3.65	3.857 ± 3.25	3.750 ± 3.45	5.714 ± 3.10	4.907 ± 2.15	3.080 ± 2.25	3.315 ± 2.35	3.530 ± 2.55	5.714 ± 2.75	4.221 ± 3.20
ds2.csv	5.714 ± 3.95	3.381 ± 3.60	3.952 ± 3.20	4.548 ± 3.35	5.714 ± 3.00	3.905 ± 2.15	2.214 ± 2.25	4.071 ± 2.35	3.500 ± 2.55	5.714 ± 2.70	4.271 ± 3.15

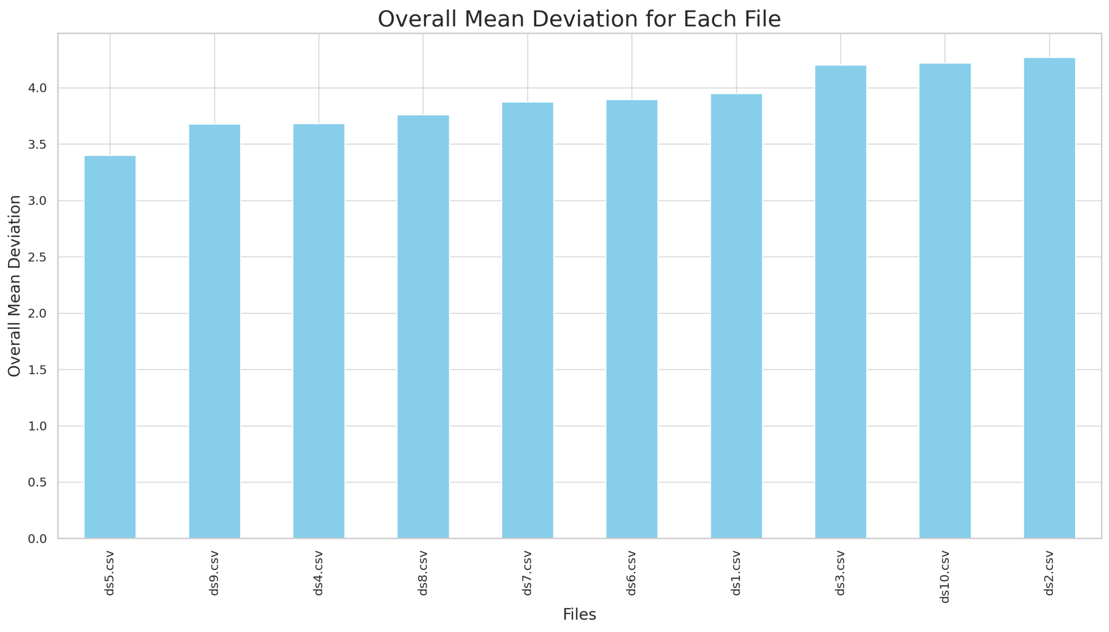

Figure 8.9.: Overall Mean Deviation for Each File in seconds

The deviation distribution for each file is illustrated in Figure 8.10. This box plot highlights the spread and central tendency of deviations, providing insights into the variability of each file's performance.

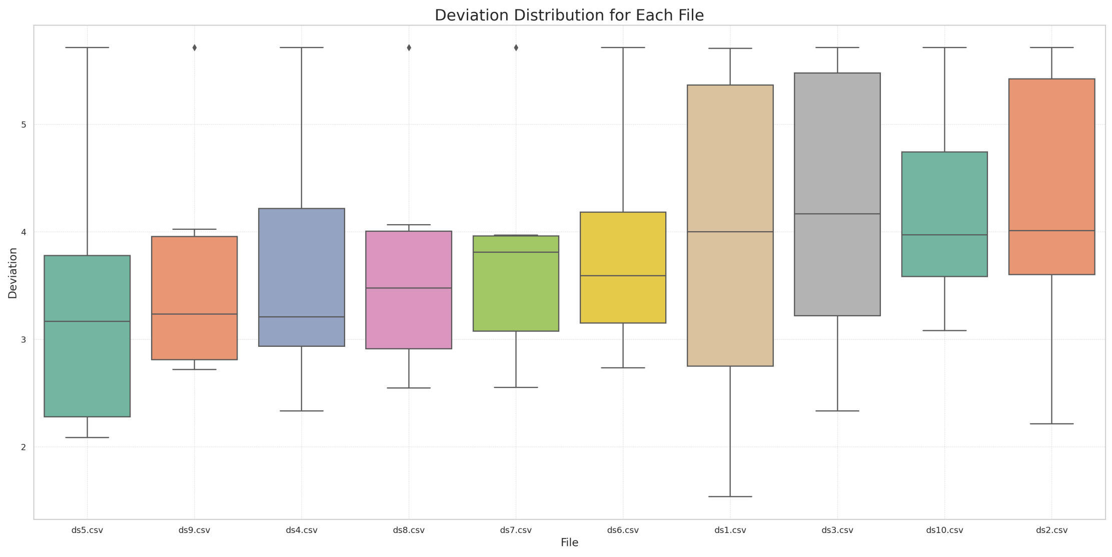

Figure 8.10.: Deviation Distribution for Each File in seconds

8.1.10. Discussion and Conclusions

The analysis reveals several key insights:

- **Best Performing Files:**
  - File *ds5.csv* shows the lowest overall mean deviation and a relatively narrow spread of deviations, indicating the best performance across all methods and variables.
  - Files *ds9.csv* and *ds4.csv* also show relatively low overall mean deviations and consistent performance.
- **Overall Performance:**
  - The overall mean deviations provide a summary of performance across all methods for each file. Lower values indicate better overall performance.
  - Files with higher overall mean deviations, such as *ds7.csv*, suggest poorer performance across the methods analyzed.

- **Method Variability:**
  - The box plot in Figure 8.10 shows that some files exhibit greater variability in their performance across different methods.
  - Files with less variability in their deviations indicate more consistent performance across methods.

Conclusion

This analysis highlights which files have better overall performance, providing insights into data quality and method effectiveness. Identifying files with lower overall mean deviations and less variability can help focus efforts on improving methods for files with higher deviations.

8.1.11. Method Correlations

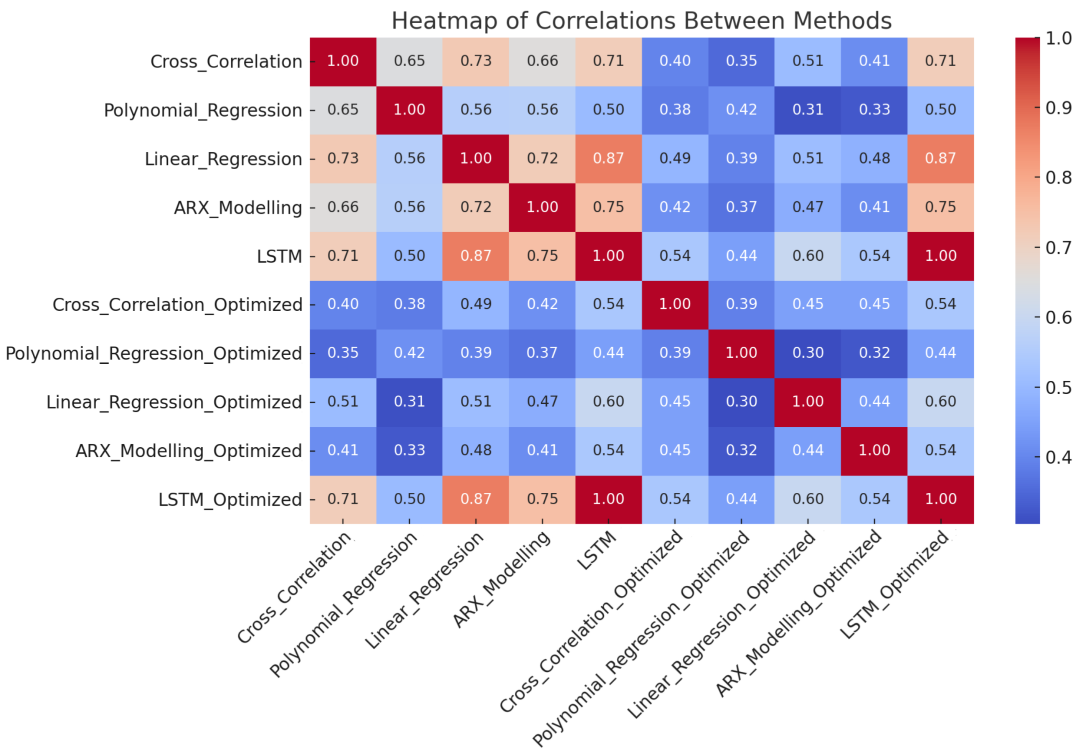

Figure 8.11.: Heatmap of Method Correlations

For our use case, the performance of the methodology in terms of speed is crucial for its scalability, especially when analyzing large datasets. The time variable is very significant in this context. The heat map of correlation was created to determine if some methodologies yield consistently similar results across the board and are most efficient in terms of time. Additionally, it was used to correlate and identify which methods produce results similar to other methods. This allows us to pinpoint methodologies that are not only efficient in terms of computation time but also comparable in outcome, thereby selecting the most effective approaches for calculating delays. Among the methodologies evaluated, polynomial regression stands out as one of the best options due to its reliability and speed. It performs overall the fastest, along with linear regression, making it a point worth considering.

8.1.12. Discussion of Results

The cross-correlation function calculates the dot product of the two signals at different time lags, normalized by the product of their standard deviations. The results from the cross-correlation method show that it works well to a certain extent because the signals have a strong linear relationship and a consistent time delay. However, the method struggles when it needs to capture non-linear relationships between the input and output signals or when the time delays vary. This limitation is especially noticeable when noise filters are applied to the output signals, making it harder for the method to detect the exact delays in the simulated data accurately. Additionally, inconsistencies in the sampling rate for the cross-correlation method significantly influence the results, leading the method to fail in generalizing the exact delays.

The Polynomial Regression method fits a polynomial curve to the data, estimating the time delay by finding the minimum of the regression error. Based on the results obtained from our implementation, it can be observed that Polynomial Regression effectively captures non-linear relationships and is robust to noise. However, the results deviate when the choice of polynomial degree is inappropriate, leading to either overfitting or underfitting, as seen in our case with the simulated data. Notably, the research also revealed that the quality of the datasets and the conditions under which data is recorded are of utmost importance. The nature of different measurement signals varies significantly, making some easier to detect and align with certain methods. Despite these variations, Polynomial Regression still proved to be the best overall solution.

The Linear Regression model dynamically represents the relationship between the input signals and a single output signal using a linear equation. Linear Regression fits a straight line to the data, assuming a linear relationship between the signals. This method works effectively due to its simplicity, interpretability, and suitability for modeling linear relationships between signals. However, a significant drawback is that Linear Regression assumes linearity, which prevents it from capturing non-linear relationships or complex interactions between multiple input signals.

The ARX (Auto-Regressive Exogenous) method models the relationship between the output signals and its past values and external input signals. In our implementation, the ARX method performed better at estimating delays under both noise and filter factors, showing promising results under filtered signals in the output data. The method's effectiveness is tied to the amount of auto-regressive and exogenous relationships captured in the data. However, the ARX method fails to estimate accurate delays when it assumes linearity, missing non-linear relationships or complex interactions between signals.

Finally, the LSTM (Long Short-Term Memory), a type of recurrent neural network, was designed in this work to handle and estimate delayed signals with long-term dependencies. It uses memory cells and gates to selectively retain or forget information, capturing complex patterns and relationships. However, in our implementation, the method did not yield accurate estimates of the delays. This was due to the technique requiring significant training data and its computational expense.

In general, each method has its strengths and weaknesses. Cross-correlation is useful for analyzing relationships between signals, while Polynomial and linear regression are effective for modeling non-linear and linear relationships, respectively. ARX models time series data with external influences and LSTM excels at capturing complex patterns and long-term dependencies. The quality of the datasets and the recording conditions are critical factors influencing the success of these methods.

Table 8.2 presents the values of Cross-Correlation, Polynomial Regression, Linear Regression, Auto-Regressive Exogenous method, and Long Short-Term Memory methods, both before and after optimization. The results clearly demonstrate the crucial role of the optimization process in time delay estimation and compensation. Furthermore, most methods exhibit a wide range of deviations, with some showing more concentrated distributions. The method with the smallest spread is Polynomial Regression, which indicates that it has the most consistent deviations. In contrast, methods with larger spreads indicate more variability in their performance.

The presence of noise has a noticeable impact on the deviations of various methods. As illustrated in Figure 8.4, methods generally show higher deviations when noise is present. This indicates that noise introduces variability and reduces the accuracy of the methods. The table summarizing the effect of noise on method deviations supports this observation, showing increased mean deviations for methods in noisy conditions.

When examining the performance of methods for each variable, it is evident that some methods perform consistently well across different signals, while others show more variability. The mean and standard deviation of deviations for each variable indicate that certain methods are more robust and adaptable to different types of data. Among these, the Polynomial Regression method performs the best overall.

Among the methods analyzed, the LSTM method consistently showed higher deviations, particularly under noisy conditions, suggesting it might be less robust to noise compared to other methods. On the other hand, methods like Polynomial Regression and Cross-Correlation (both standard and optimized) often exhibited lower deviations, indicating better performance. This could be due to their ability to fit the data more accurately and their relative simplicity, making them less prone to overfitting than more complex models like LSTM.

In conclusion, the analysis provides a comprehensive understanding of the performance of various methods under different conditions. Noise and filtering significantly affect deviations. The performance analysis by file underscores the impact of dataset characteristics on method effectiveness. After optimization, the Polynomial Regression method performs best overall due to its simplicity and robustness to noise. It consistently shows more accurate results than other methods, performing well with both noise and filter across all types of signals and the entire dataset. This makes it a reliable and robust choice for minimizing deviations for the given kind of data.

9. Conclusion

The research aimed to address the challenge of efficiently, practically, and feasibly aligning and synchronizing recorded data from different systems to optimize powertrain calibration using SDS data. By systematically exploring and evaluating various methods, the study sought to answer the research question: *How can recorded data from different systems be aligned and synchronized to build models for optimizing powertrain calibration in a manner that is efficient, practical, and feasible?*

9.0.1. Summary of Research Objectives and Outcomes

Objective 1: Review Existing Techniques

The review identified several techniques, including Cross-Correlation, Polynomial Regression, Linear Regression, Auto-Regressive Exogenous (ARX) methods, and Long Short-Term Memory (LSTM). Each method has distinct advantages and disadvantages, particularly in handling linear versus non-linear relationships, noise, and varying time delays.

Objective 2: Evaluate Techniques Using Defined Metrics

Evaluation based on efficiency, practicability, and feasibility revealed that simpler methods like Polynomial Regression and Cross-Correlation are more efficient and practical in most scenarios, particularly where noise is present. Complex methods like LSTM, while theoretically powerful, were less practical due to their high computational requirements and sensitivity to noise.

Objective 3: Compare Techniques and Algorithms

Comparative analysis showed that Polynomial Regression consistently outperformed other methods in aligning and synchronizing data, particularly in noisy conditions. Its ability to handle non-linear relationships without overfitting makes it a robust and reliable choice. Additionally, the research revealed that the quality of the datasets and the conditions under which data is recorded are of utmost importance. The nature of different measurement signals varies significantly, making some easier to detect and align with certain methods. Despite these variations, Polynomial Regression still proved to be the best overall solution.

Objective 4: Develop Automated Script for Post-Processing

An automated script was successfully developed, incorporating the Polynomial Regression technique for time alignment and synchronization. The script also includes other necessary processing steps, streamlining the post-processing of recorded data.

Objective 5: Apply Selected Techniques to Real Data

The selected Polynomial Regression method was applied to real recorded data from different systems, demonstrating its accuracy and reliability in optimizing powertrain calibration. The results affirmed the method's effectiveness in practical scenarios, with minimal deviations observed across different datasets.

9.0.2. Final Conclusion

The research concluded that **Polynomial Regression** is the most efficient, practical, and feasible method for aligning and synchronizing recorded data from different systems to optimize powertrain calibration using SDS data. It consistently demonstrated superior performance across various metrics, particularly in handling non-linear relationships and maintaining robustness in noisy conditions. Moreover, the study highlighted that the quality of the datasets and the conditions under which data is recorded are crucial to the success of any alignment and synchronization efforts. The diverse nature of measurement signals also influences the effectiveness of different methods, with Polynomial Regression emerging as the best overall fit.

The development of an automated post-processing script further enhances the practical application of this method, ensuring that it can be efficiently integrated into real-world powertrain calibration workflows. In answering the research question, the study found that a method's simplicity, robustness to noise, and ability to handle non-linear relationships are critical factors in achieving effective data synchronization for powertrain optimization. This research provides a solid foundation for future work, suggesting that continued refinement and optimization of Polynomial Regression, coupled with real-world application testing, will further enhance its effectiveness in the field.

Bibliography

Abdellatif, H., & Alshibani, A. (2019). Major factors causing delay in the delivery of manufacturing and building projects in saudi arabia. buildings, 2 (9): 1–15. (Cit. on p. 29).

Ahmadi, M., Sharifi, A., Hassantabar, S., Enayati, S., et al. (2021). Qais-dsnn: Tumor area segmentation of mri image with optimized quantum matched-filter technique and deep spiking neural network. *BioMed Research International*, *2021* (cit. on p. 76).

Ajiboye, A., Abdullah-Arshah, R., & Hongwu, Q. (2015). Evaluating the effect of dataset size on predictive model using supervised learning technique (cit. on p. 60).

Albrecht, O., & Taylor, C. (2022). A linear regression variable time delay estimation algorithm for the analysis of hydraulic manipulators. *2022 UKACC 13th International Conference on Control (CONTROL)*, 148–153 (cit. on pp. 33, 34).

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A., & Asari, V. K. (2019). A state-of-the-art survey on deep learning theory and architectures. *electronics*, *8*(3), 292 (cit. on p. 59).

An, Q., Tao, Z., Xu, X., El Mansori, M., & Chen, M. (2020). A data-driven model for milling tool remaining useful life prediction with convolutional and stacked lstm network. *Measurement*, *154*, 107461 (cit. on p. 84).

Anderson, J. D., Esposito, P. B., Martin, W., Thornton, C. L., & Muhleman, D. O. (1975). Experimental test of general relativity using time-delay data from mariner 6 and mariner 7. *Astrophysical Journal, vol. 200, Aug. 15, 1975, pt. 1, p. 221-233.*, *200*, 221–233 (cit. on p. 79).

Azmi, H., & Yazdizadeh, A. (2023). Fault-tolerant controller design for nonlinear systems with multiple input and state delays based on sliding mode algorithm. *International Journal of Robust and Nonlinear Control*, *33*(15), 9128–9149 (cit. on p. 30).

Barbano, R., Zhang, C., Arridge, S., & Jin, B. (2021). Quantifying model uncertainty in inverse problems via bayesian deep gradient descent. *2020 25th International Conference on Pattern Recognition (ICPR)*, 1392–1399 (cit. on p. 65).

Benesty, J., Chen, J., & Huang, Y. (2004). Time-delay estimation via linear interpolation and cross correlation. *IEEE Transactions on speech and audio processing*, *12*(5), 509–519 (cit. on pp. 38, 81).

Bento, M. E. (2020). Fixed low-order wide-area damping controller considering time delays and power system operation uncertainties. *IEEE Transactions on Power Systems*, *35*(5), 3918–3926 (cit. on p. 26).

Bertrand, A. (2011). Applications and trends in wireless acoustic sensor networks: A signal processing perspective. *2011 18th IEEE symposium on communications and vehicular technology in the Benelux (SCVT)*, 1–6 (cit. on p. 74).

Boker, S. M., Rotondo, J. L., Xu, M., & King, K. (2002). Windowed cross-correlation and peak picking for the analysis of variability in the association between behavioral time series. *Psychological methods*, *7*(3), 338 (cit. on p. 75).

Borri, A., & Pepe, P. (2020). Event-triggered control of nonlinear systems with time-varying state delays. *IEEE Transactions on Automatic Control*, *66*(6), 2846–2853 (cit. on p. 30).

Candon, M., Esposito, M., Fayek, H., Levinski, O., Koschel, S., Joseph, N., Carrese, R., & Marzocca, P. (2022). Advanced multi-input system identification for next generation aircraft loads monitoring using linear regression, neural networks and deep learning. *Mechanical Systems and Signal Processing*, *171*, 108809 (cit. on p. 82).

Cao, Y., & Su, S. (2023). Fractional gradient descent algorithms for systems with outliers: A matrix fractional derivative or a scalar fractional derivative. *Chaos, Solitons & Fractals*, *174*, 113881 (cit. on p. 66).

Carrion, J. E., & Spencer, B. (2006). Real-time hybrid testing using model-based delay compensation. *Proceedings of the 4th international conference on earthquake engineering*, *299* (cit. on p. 76).

Chatterjee, S., & Hadi, A. S. (1988). Impact of simultaneous omission of a variable and an observation on a linear regression equation. *Computational Statistics & Data Analysis*, *6*(2), 129–144 (cit. on p. 46).

Chen, H., Huang, P., Liu, Z., & Ma, Z. (2020). Time delay prediction for space telerobot system with a modified sparse multivariate linear regression method. *Acta Astronautica*, *166*, 330–341 (cit. on pp. 36, 38).

Chen, H., Wu, H.-C., Chan, S.-C., & Lam, W.-H. (2019). A stochastic quasi-newton method for large-scale nonconvex optimization with applications. *IEEE transactions on neural networks and learning systems*, *31*(11), 4776–4790 (cit. on p. 65).

Chen, J., Gan, M., Zhu, Q., Narayan, P., & Liu, Y. (2021). Robust standard gradient descent algorithm for arx models using aitken acceleration technique. *IEEE transactions on cybernetics*, *52*(9), 9646–9655 (cit. on p. 67).

Chen, J., Huang, B., Ding, F., & Gu, Y. (2018). Variational bayesian approach for arx systems with missing observations and varying time-delays. *Automatica*, *94*, 194–204 (cit. on pp. 35, 39).

Chen, J., Pu, Y., Guo, L., Cao, J., & Zhu, Q. (2023). Second-order optimization methods for time-delay autoregressive exogenous models: Nature gradient descent method and its two modified methods. *International Journal of Adaptive Control and Signal Processing*, *37*(1), 211–223 (cit. on p. 66).

Chen, X., Zhao, S., & Liu, F. (2020). Online identification of time-delay jump markov autoregressive exogenous systems with recursive expectation-maximization algorithm. *International Journal of Adaptive Control and Signal Processing*, *34*(3), 407–426 (cit. on p. 38).

Chicco, D., Warrens, M. J., & Jurman, G. (2021). The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *Peerj computer science*, *7*, e623 (cit. on p. 49).

Chinaev, A., Thüne, P., & Enzner, G. (2021). Double-cross-correlation processing for blind sampling-rate and time-offset estimation. *IEEE/ACM Transac-*

tions on Audio, Speech, and Language Processing, *29*, 1881–1896 (cit. on p. 74).

Coelho, F., & Neto, J. P. (2017). A method for regularization of evolutionary polynomial regression. *Applied Soft Computing*, *59*, 223–228 (cit. on p. 67).

Comte-Bellot, G., & Corrsin, S. (1971). Simple eulerian time correlation of full- and narrow-band velocity signals in grid-generated,'isotropic'turbulence. *Journal of fluid mechanics*, *48*(2), 273–337 (cit. on p. 82).

Czogala, E., & Leski, J. (2012). *Fuzzy and neuro-fuzzy intelligent systems* (Vol. 47). Physica. (Cit. on p. 57).

De Souza, D. A., Batista, J. G., Vasconcelos, F. J., Dos Reis, L. L., Machado, G. F., Costa, J. R., Junior, J. N., Silva, J. L., Rios, C. S., & Júnior, A. B. (2021). Identification by recursive least squares with kalman filter (rls-kf) applied to a robotic manipulator. *IEEE Access*, *9*, 63779–63789 (cit. on p. 62).

Deng, B.-C., Yun, Y.-H., Liang, Y.-Z., Cao, D.-S., Xu, Q.-S., Yi, L.-Z., & Huang, X. (2015). A new strategy to prevent over-fitting in partial least squares models based on model population analysis. *Analytica chimica acta*, *880*, 32–41 (cit. on p. 78).

Desai, M. V., & Shah, S. N. (2020). A local multivariate polynomial regression approach for ionospheric delay estimation of single-frequency navic receiver. *SN Applied Sciences*, *2*(9), 1503 (cit. on p. 38).

DiBiase, J. H., Silverman, H. F., & Brandstein, M. S. (2001). Robust localization in reverberant rooms. In *Microphone arrays: Signal processing techniques and applications* (pp. 157–180). Springer. (Cit. on p. 74).

Dinov, M. (2017). The design and implementation of novel computational and machine learning approaches for modelling brain dynamics: Towards more interpretable and real-time brain analysis (cit. on p. 75).

Duran-Hernandez, C., Ledesma-Alonso, R., & Etcheverry, G. (2020). Using autoregressive with exogenous input models to study pulsatile flows. *Applied Sciences*, *10*(22), 8228 (cit. on p. 52).

Duriez, T., Brunton, S. L., & Noack, B. R. (2017). *Machine learning control-taming nonlinear dynamics and turbulence* (Vol. 116). Springer. (Cit. on p. 82).

Ebrahimi, H., & Rajaee, T. (2017). Simulation of groundwater level variations using wavelet combined with neural network, linear regression and support vector machine. *Global and Planetary Change*, *148*, 181–191 (cit. on pp. 38, 39).

El Anes, A., Maraoui, S., & Bouzrara, K. (2016). Optimal expansions of multivariable arx processes on laguerre bases via the newton–raphson method. *International Journal of Adaptive Control and Signal Processing*, *30*(4), 578–598 (cit. on p. 65).

El-Koka, A., Cha, K.-H., & Kang, D.-K. (2013). Regularization parameter tuning optimization approach in logistic regression. *2013 15th International Conference on Advanced Communications Technology (ICACT)*, 13–18 (cit. on p. 66).

Fan, X.-j., Liu, J.-f., Luo, M.-m., Zeng, X.-y., Lu, J., Liu, J., & Yang, W.-r. (2020). A method for nonlinearity compensation of ofdr based on polynomial regression algorithm. *Optoelectronics letters*, *16*(2), 108–111 (cit. on p. 38).

Fatehi, A., & Huang, B. (2017). Kalman filtering approach to multi-rate information fusion in the presence of irregular sampling rate and variable measurement delay. *Journal of Process Control*, *53*, 15–25 (cit. on p. 31).

Florea, N. M., Meghisan, G. M., & Nistor, C. (2016). Multiple linear regression equation for economic dimension of standard of living. *Finante-provocarile viitorului (Finance-Challenges of the Future)*, *1*(18), 103–108 (cit. on p. 46).

Franzese, G., Milios, D., Filippone, M., & Michiardi, P. (2021). A scalable bayesian sampling method based on stochastic gradient descent isotropization. *Entropy*, *23*(11), 1426 (cit. on pp. 65, 67).

Fridman, E. (2014). *Introduction to time-delay systems: Analysis and control*. Springer. (Cit. on p. 23).

Fridman, E., & Shaikhet, L. (2017). Stabilization by using artificial delays: An lmi approach. *Automatica*, *81*, 429–437 (cit. on p. 30).

Gaffke, N., & Schwabe, R. (2019). Quasi-newton algorithm for optimal approximate linear regression design: Optimization in matrix space. *Journal of Statistical Planning and Inference*, *198*, 62–78 (cit. on p. 66).

Gao, M., Zhang, L., Qi, W., Cao, J., Cheng, J., Kao, Y., Wei, Y., & Yan, X. (2020). Smc for semi-markov jump ts fuzzy systems with time delay. *Applied Mathematics and Computation*, *374*, 125001 (cit. on p. 30).

George, D. P., & Sokolovsky, P. (2014). Micro python documentation. (Cit. on p. 88).

Gibbons, S. J., & Ringdal, F. (2006). The detection of low magnitude seismic events using array-based waveform correlation. *Geophysical Journal International*, *165*(1), 149–166 (cit. on p. 82).

Gibert, K., Sànchez–Marrè, M., & Izquierdo, J. (2016). A survey on pre-processing techniques: Relevant issues in the context of environmental data mining. *AI Communications*, *29*(6), 627–663 (cit. on p. 77).

Guo, Y., & Ma, C. (2023). Identification of time-delay markov jumps autoregressive system with recursive expectation maximum and convex optimization algorithm. *Transactions of the Institute of Measurement and Control*, *45*(9), 1607–1618 (cit. on p. 66).

Hagras, H. (2007). Type-2 flcs: A new generation of fuzzy controllers. *IEEE Computational Intelligence Magazine*, *2*(1), 30–43 (cit. on p. 56).

Han, J., & Hu, R. (2021). Recurrent neural networks for stochastic control problems with delay. *Mathematics of Control, Signals, and Systems*, *33*(4), 775–795 (cit. on p. 33).

Han, R., Salehi, Y., Huang, B., & Prasad, V. (2020). State estimation for multirate measurements in the presence of integral term and variable delay. *IEEE Transactions on Control Systems Technology*, *29*(6), 2416–2426 (cit. on p. 64).

Hanus, R. (2019). Time delay estimation of random signals using cross-correlation with hilbert transform. *Measurement*, *146*, 792–799 (cit. on p. 82).

Hu, J., Jiang, B., Lin, L., Wen, Z., & Yuan, Y.-x. (2019). Structured quasi-newton methods for optimization with orthogonality constraints. *SIAM Journal on Scientific Computing*, *41*(4), A2239–A2269 (cit. on p. 66).

Huang, T., Zhang, Q., Tang, X., Zhao, S., & Lu, X. (2022). A novel fault diagnosis method based on cnn and lstm and its application in fault diagnosis for complex systems. *Artificial Intelligence Review*, *55*(2), 1289–1315 (cit. on p. 33).

Ianniello, J. (1982). Time delay estimation via cross-correlation in the presence of large estimation errors. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *30*(6), 998–1003 (cit. on p. 73).

Ighalo, J. O., Adeniyi, A. G., & Marques, G. (2020). Application of linear regression algorithm and stochastic gradient descent in a machine-learning environment for predicting biomass higher heating value. *Biofuels, Bioproducts and Biorefining*, *14*(6), 1286–1295 (cit. on p. 70).

Jafari, V., & Rezvani, M. H. (2023). Joint optimization of energy consumption and time delay in iot-fog-cloud computing environments using nsga-ii metaheuristic algorithm. *Journal of Ambient Intelligence and Humanized Computing*, *14*(3), 1675–1698 (cit. on p. 27).

Jain, S., Fall, K., & Patra, R. (2004). Routing in a delay tolerant network. *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, 145–158 (cit. on p. 82).

James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An introduction to statistical learning* (Vol. 112). Springer. (Cit. on p. 43).

Jing, S. (2023). Time-delay hammerstein system identification using modified cross-correlation method and variable stacking length multi-error algorithm. *Mathematics and Computers in Simulation*, *207*, 288–300 (cit. on p. 66).

Kaiser, E., Kutz, J. N., & Brunton, S. L. (2018). Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, *474*(2219), 20180335 (cit. on p. 82).

Kasture, H., Bartolini, D. B., Beckmann, N., & Sanchez, D. (2015). Rubik: Fast analytical power management for latency-critical systems. *Proceedings* *of the 48th International Symposium on Microarchitecture*, 598–610 (cit. on p. 80).

Kawabata, A., Chatterjee, B. C., Ba, S., & Oki, E. (2017). A real-time delay-sensitive communication approach based on distributed processing. *IEEE Access*, *5*, 20235–20248 (cit. on p. 24).

Khan, A., Gupta, S., & Gupta, S. K. (2020). Multi-hazard disaster studies: Monitoring, detection, recovery, and management, based on emerging technologies and optimal techniques. *International journal of disaster risk reduction*, *47*, 101642 (cit. on p. 27).

Khorram, S., McInnis, M. G., & Provost, E. M. (2019). Jointly aligning and predicting continuous emotion annotations. *IEEE Transactions on Affective Computing*, *12*(4), 1069–1083 (cit. on p. 76).

Khyam, M. O., Ge, S. S., Li, X., & Pickering, M. R. (2016). Highly accurate time-of-flight measurement technique based on phase-correlation for ultrasonic ranging. *IEEE sensors journal*, *17*(2), 434–443 (cit. on p. 73).

Kolluri, J., Kotte, V. K., Phridviraj, M., & Razia, S. (2020). Reducing overfitting problem in machine learning using novel l1/4 regularization method. *2020 4th international conference on trends in electronics and informatics (ICOEI)(48184)*, 934–938 (cit. on p. 70).

Kopetz, H., & Steiner, W. (2022). Real-time communication. In *Real-time systems: Design principles for distributed embedded applications* (pp. 177–200). Springer. (Cit. on p. 24).

Lai, X., Liu, Q., Wei, X., Wang, W., Zhou, G., & Han, G. (2013). A survey of body sensor networks. *Sensors*, *13*(5), 5406–5447 (cit. on p. 77).

Lakshmanan, M., & Senthilkumar, D. V. (2011). *Dynamics of nonlinear time-delay systems*. Springer Science & Business Media. (Cit. on p. 24).

Le Bastard, C., Wang, Y., Baltazart, V., & Derobert, X. (2013). Time delay and permittivity estimation by ground-penetrating radar with support vector regression. *IEEE Geoscience and Remote Sensing Letters*, *11*(4), 873–877 (cit. on p. 35).

Li, B., Wang, M., & Yang, Y. (2017). Multiple linear regression with correlated explanatory variables and responses. *Survey Review*, *49*(352), 1–8 (cit. on p. 55).

Li, C. (2013). *Complexity analysis of physiological time series with applications to neonatal sleep electroencephalogram signals* [Doctoral dissertation, Case Western Reserve University]. (Cit. on p. 77).

Li, X., & Li, P. (2021). Stability of time-delay systems with impulsive control involving stabilizing delays. *Automatica*, *124*, 109336 (cit. on p. 24).

Lim, J.-Y., Kim, S., Kim, H.-K., & Kim, Y.-K. (2022). Long short-term memory (lstm)-based wind speed prediction during a typhoon for bridge traffic control. *Journal of Wind Engineering and Industrial Aerodynamics*, *220*, 104788 (cit. on p. 39).

Lin, J.-Y., Cheng, C.-T., & Chau, K.-W. (2006). Using support vector machines for long-term discharge prediction. *Hydrological sciences journal*, *51*(4), 599–612 (cit. on p. 33).

List, A. (2021). About avl [Retrieved from https://www.avl.com/about-avl]. https://www.avl.com/about-avl (cit. on p. 7).

Liu, X., Ren, G., Yin, X., Zhang, B., & Wang, Y. (2022). A polynomial inversion-based fast time-delay estimation method for wideband large-scale antenna systems. *Applied Sciences*, *12*(7), 3378 (cit. on pp. 33, 35).

Liu, Y., Singh, A. K., Zhao, J., Meliopoulos, A. S., Pal, B., bin Mohd Ariff, M. A., Van Cutsem, T., Glavic, M., Huang, Z., Kamwa, I., et al. (2021). Dynamic state estimation for power system control and protection. *IEEE Transactions on Power Systems*, *36*(6), 5909–5921 (cit. on p. 80).

Lizhen, W., Yifan, Z., Gang, W., & Xiaohong, H. (2022). A novel short-term load forecasting method based on mini-batch stochastic gradient descent regression model. *Electric Power Systems Research*, *211*, 108226 (cit. on p. 70).

Lu, J., Li, D., & Xi, Y. (2013). Probability-based constrained mpc for structured uncertain systems with state and random input delays. *International Journal of Systems Science*, *44*(7), 1354–1365 (cit. on p. 30).

Mahdavian, A., Banakar, A., Beigi, M., Tavakoli, M., & Zareiforoush, H. (2012). Modeling of some important mechanical properties of barley straw using fuzzy logic. *Int. J. Agric. Food Sci*, *2*, 21–29 (cit. on p. 58).

Meiri, R., & Zahavi, J. (2006). Using simulated annealing to optimize the feature selection problem in marketing applications. *European journal of operational research*, *171*(3), 842–858 (cit. on p. 83).

Meulenbroek, N. E., & Pichardo, S. (2020). Multiple linear regression estimation of onset time delay for experimental transcranial narrowband ultrasound signals. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, *68*(4), 1032–1039 (cit. on pp. 33, 34).

Mobayen, S., Bayat, F., Omidvar, H., & Fekih, A. (2020). Robust global controller design for discrete-time descriptor systems with multiple time-varying delays. *International Journal of Robust and Nonlinear Control*, *30*(7), 2809–2831 (cit. on p. 30).

Mostafa, H. (2017). Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, *29*(7), 3227–3235 (cit. on p. 60).

Mould, D. R., & Upton, R. (2012). Basic concepts in population modeling, simulation, and model-based drug development. *CPT: pharmacometrics & systems pharmacology*, *1*(9), 1–14 (cit. on p. 79).

Mulder, M., Pool, D. M., Abbink, D. A., Boer, E. R., Zaal, P. M., Drop, F. M., van der El, K., & van Paassen, M. M. (2017). Manual control cybernetics: State-of-the-art and current trends. *IEEE Transactions on Human-Machine Systems*, *48*(5), 468–485 (cit. on p. 79).

Murni, I. K., Wirawan, M. T., Patmasari, L., Sativa, E. R., Arafuri, N., Nugroho, S., & Noormanto. (2021). Delayed diagnosis in children with congenital heart disease: A mixed-method study. *BMC pediatrics*, *21*, 1–7 (cit. on p. 27).

Murray-Smith, R. (1994). *A local model network approach to nonlinear modelling* [Doctoral dissertation, University of Strathclyde]. (Cit. on p. 78).

Murugan, P. (2018). Learning the sequential temporal information with recurrent neural networks. *arXiv preprint arXiv:1807.02857* (cit. on p. 60).

Najeh, T., Mbarek, A., Bouzrara, K., Nabli, L., & Messaoud, H. (2017). New methods of laguerre pole optimization for the arx model expansion on laguerre bases. *ISA transactions*, *70*, 93–103 (cit. on p. 65).

Nangrani, S., Singh, A. R., & Chandan, A. (2018). Chaos driven instability control using interval type-2 fuzzy logic controller for better performance. *Journal of Intelligent & Fuzzy Systems*, *34*(3), 1491–1501 (cit. on p. 56).

Nelles, O., & Nelles, O. (2020). *Nonlinear dynamic system identification*. Springer. (Cit. on p. 83).

Niu, B., Wang, D., Liu, M., Song, X., Wang, H., & Duan, P. (2019). Adaptive neural output-feedback controller design of switched nonlower triangular nonlinear systems with time delays. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(10), 4084–4093 (cit. on p. 26).

Osah, S., Acheampong, A. A., Fosu, C., & Dadzie, I. (2021). Deep learning model for predicting daily igs zenith tropospheric delays in west africa using tensorflow and keras. *Advances in Space Research*, *68*(3), 1243–1262 (cit. on p. 78).

Pal, M., Bharati, P., Pal, M., & Bharati, P. (2019). Introduction to correlation and linear regression analysis. *Applications of regression techniques*, 1–18 (cit. on p. 45).

Pan, J., Le Bastard, C., Wang, Y., & Sun, M. (2018). Time-delay estimation using ground-penetrating radar with a support vector regression-based linear prediction method. *IEEE Transactions on Geoscience and Remote Sensing*, *56*(5), 2833–2840 (cit. on p. 35).

Parvathy, R., & Devi, R. R. (2014). Gradient descent based linear regression approach for modeling pid parameters. *2014 International conference on power signals control and computations (EPSCICON)*, 1–4 (cit. on p. 69).

Pawlowski, A., Guzmán, J. L., Berenguel, M., Normey-Rico, J. E., & Dormido, S. (2016). Multivariable gpc for processes with multiple time delays:

Implementation issues. *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–6 (cit. on p. 30).

Peng, H., Nakano, K., & Shioya, H. (2006). Nonlinear predictive control using neural nets-based local linearization arx model—stability and industrial application. *IEEE Transactions on Control Systems Technology*, *15*(1), 130–143 (cit. on p. 81).

Petrillo, A., Salvi, A., Santini, S., & Valente, A. S. (2018). Adaptive multi-agents synchronization for collaborative driving of autonomous vehicles with multiple communication delays. *Transportation research part C: emerging technologies*, *86*, 372–392 (cit. on p. 29).

Plonis, D., Katkevičius, A., Gurskas, A., Urbanavičius, V., Maskeliūnas, R., & Damaševičius, R. (2020). Prediction of meander delay system parameters for internet-of-things devices using pareto-optimal artificial neural network and multiple linear regression. *IEEE access*, *8*, 39525–39535 (cit. on p. 33).

Podobnik, B., & Stanley, H. E. (2008). Detrended cross-correlation analysis: A new method for analyzing two nonstationary time series. *Physical Review Letters*, *100*(8), 084102 (cit. on p. 38).

Privara, S., Cigler, J., Váňa, Z., Oldewurtel, F., Sagerschnig, C., & Žáčeková, E. (2013). Building modeling as a crucial part for building predictive control. *Energy and Buildings*, *56*, 8–22 (cit. on p. 79).

Qi, J., Du, J., Siniscalchi, S. M., Ma, X., & Lee, C.-H. (2020). On mean absolute error for deep neural network based vector-to-vector regression. *IEEE Signal Processing Letters*, *27*, 1485–1489 (cit. on p. 49).

Qin, Z., Wang, J.-L., Huang, Y.-L., & Ren, S.-Y. (2018). Analysis and adaptive control for robust synchronization and h synchronization of complex dynamical networks with multiple time-delays. *Neurocomputing*, *289*, 241–251 (cit. on p. 31).

Santos, T. L., Flesch, R. C., & Normey-Rico, J. E. (2014). On the filtered smith predictor for mimo processes with multiple time delays. *Journal of Process Control*, *24*(4), 383–400 (cit. on p. 30).

Santos, T. L., Torrico, B. C., & Normey-Rico, J. E. (2016). Simplified filtered smith predictor for mimo processes with multiple time delays. *ISA transactions*, *65*, 339–349 (cit. on p. 30).

Saunders, C., Gammerman, A., & Vovk, V. (1998). Ridge regression learning algorithm in dual variables (cit. on p. 49).

Savorani, F., Tomasi, G., & Engelsen, S. B. (2010). Icoshift: A versatile tool for the rapid alignment of 1d nmr spectra. *Journal of magnetic resonance*, *202*(2), 190–202 (cit. on p. 74).

Schmelzeisen-Redeker, G., Schoemaker, M., Kirchsteiger, H., Freckmann, G., Heinemann, L., & Del Re, L. (2015). Time delay of cgm sensors: Relevance, causes, and countermeasures. *Journal of diabetes science and technology*, *9*(5), 1006–1015 (cit. on p. 61).

Seber, G. A., & Lee, A. J. (2012). *Linear regression analysis*. John Wiley & Sons. (Cit. on p. 46).

Selimović, D., Lerga, J., Prpić-Oršić, J., & Kenji, S. (2020). Improving the performance of dynamic ship positioning systems: A review of filtering and estimation techniques. *Journal of Marine Science and Engineering*, *8*(4), 234 (cit. on p. 62).

Shakarami, M. R., & Davoudkhani, I. F. (2016). Wide-area power system stabilizer design based on grey wolf optimization algorithm considering the time delay. *Electric Power Systems Research*, *133*, 149–159 (cit. on p. 33).

Shang, C., Gao, X., Yang, F., & Huang, D. (2013). Novel bayesian framework for dynamic soft sensor based on support vector machine with finite impulse response. *IEEE Transactions on Control Systems Technology*, *22*(4), 1550–1557 (cit. on p. 39).

Shangguan, X.-C., Zhang, C.-K., He, Y., Jin, L., Jiang, L., Spencer, J. W., & Wu, M. (2020). Robust load frequency control for power system considering transmission delay and sampling period. *IEEE Transactions on Industrial Informatics*, *17*(8), 5292–5303 (cit. on p. 26).

Shen, Y., Wu, Z.-G., Shi, P., Shu, Z., & Karimi, H. R. (2019). H control of markov jump time-delay systems under asynchronous controller and quantizer. *Automatica*, *99*, 352–360 (cit. on p. 26).

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, *404*, 132306 (cit. on p. 39).

Sheykhi, N., Salami, A., Guerrero, J. M., Agundis-Tinajero, G. D., & Faghihi, T. (2022). A comprehensive review on telecommunication challenges of microgrids secondary control. *International Journal of Electrical Power & Energy Systems*, *140*, 108081 (cit. on p. 29).

Sinha, P. (2013). Multivariate polynomial regression in data mining: Methodology, problems and solutions. *Int. J. Sci. Eng. Res*, *4*(12), 962–965 (cit. on pp. 48, 52, 54).

Sipahi, R., Vyhlídal, T., Niculescu, S.-I., Pepe, P., et al. (2012). *Time delay systems: Methods, applications and new trends*. Springer. (Cit. on p. 23).

Snyder, D., Garcia-Romero, D., & Povey, D. (2015). Time delay deep neural network-based universal background models for speaker recognition. *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 92–97 (cit. on p. 33).

Su, Y., & Kuo, C.-C. J. (2019). On extended long short-term memory and dependent bidirectional recurrent neural network. *Neurocomputing*, *356*, 151–161 (cit. on p. 59).

Tian, B., Wang, G., Xu, Z., Zhang, Y., & Zhao, X. (2021). Communication delay compensation for string stability of cacc system using lstm prediction. *Vehicular Communications*, *29*, 100333 (cit. on p. 33).

Tsiouris, K. M., Pezoulas, V. C., Zervakis, M., Konitsiotis, S., Koutsouris, D. D., & Fotiadis, D. I. (2018). A long short-term memory deep learning network for the prediction of epileptic seizures using eeg signals. *Computers in biology and medicine*, *99*, 24–37 (cit. on p. 61).

Uyanık, G. K., & Güler, N. (2013). A study on multiple linear regression analysis. *Procedia-Social and Behavioral Sciences*, *106*, 234–240 (cit. on p. 43).

Vasicek, D. (2019). Artificial intelligence and machine learning: Practical aspects of overfitting and regularization. *Information Services & Use*, *39*(4), 281–289 (cit. on p. 54).

Wang, A., Mu, B., & Shi, Y. (2017). Event-triggered consensus control for multiagent systems with time-varying communication and event-detecting delays. *IEEE Transactions on Control Systems Technology*, *27*(2), 507–515 (cit. on p. 30).

Wang, X., Han, Y., Leung, V. C., Niyato, D., Yan, X., & Chen, X. (2020). Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, *22*(2), 869–904 (cit. on p. 85).

Wang, X., Yan, L., & Zhang, Q. (2021). Research on the application of gradient descent algorithm in machine learning. *2021 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, 11–15 (cit. on p. 69).

Wang, Y., Cao, J., & Wang, H. (2021). State estimation for markovian coupled neural networks with multiple time delays via event-triggered mechanism. *Neural Processing Letters*, *53*, 893–906 (cit. on p. 30).

Wang, Y., Tian, J., Sun, Z., Wang, L., Xu, R., Li, M., & Chen, Z. (2020). A comprehensive review of battery modeling and state estimation approaches for advanced battery management systems. *Renewable and Sustainable Energy Reviews*, *131*, 110015 (cit. on p. 63).

Wu, D., & Xu, J. (2020). On the optimal weighted ℓ_2 regularization in overparameterized linear regression. *Advances in Neural Information Processing Systems*, *33*, 10112–10123 (cit. on p. 66).

Wu, D., Bai, Y., & Yu, C. (2019). A new computational approach for optimal control problems with multiple time-delay. *Automatica*, *101*, 388–395 (cit. on p. 31).

Xiao, Z., Wen, H., Markham, A., Trigoni, N., Blunsom, P., & Frolik, J. (2014). Non-line-of-sight identification and mitigation using received signal strength. *IEEE Transactions on Wireless Communications*, *14*(3), 1689–1702 (cit. on p. 78).

Xie, J., Li, C., Li, N., Li, P., Wang, X., Gao, D., Yao, D., Xu, P., Yin, G., & Li, F. (2021). Robust autoregression with exogenous input model for system identification and predicting. *Electronics*, *10*(6), 755 (cit. on p. 52).

Xiong, L., Li, H., & Wang, J. (2018). Lmi based robust load frequency control for time delayed power system via delay margin estimation. *International Journal of Electrical Power & Energy Systems*, *100*, 91–103 (cit. on p. 30).

Xiong, W., Li, Y., Zhao, Y., & Huang, B. (2017). Adaptive soft sensor based on time difference gaussian process regression with local time-delay reconstruction. *Chemical Engineering Research and Design*, *117*, 670–680 (cit. on pp. 36, 39).

Xu, S., Peng, H., & Tang, Y. (2020). Preview path tracking control with delay compensation for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *22*(5), 2979–2989 (cit. on p. 29).

Yan, S., Shen, M., Nguang, S. K., & Zhang, G. (2019). Event-triggered H_∞ control of networked control systems with distributed transmission delay. *IEEE Transactions on Automatic Control*, *65*(10), 4295–4301 (cit. on p. 26).

Yin, J., Ning, C., & Tang, T. (2022). Data-driven models for train control dynamics in high-speed railways: Lag-lstm for train trajectory prediction. *Information Sciences*, *600*, 377–400 (cit. on p. 33).

Yu, S.-H., & Horng, T.-S. (2019). Highly linear phase-canceling self-injection-locked ultrasonic radar for non-contact monitoring of respiration and heartbeat. *IEEE Transactions on Biomedical Circuits and Systems*, *14*(1), 75–90 (cit. on p. 77).

Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., & Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4694–4702 (cit. on p. 60).

Zadeh, L. A. (2008). Is there a need for fuzzy logic? *Information sciences*, *178*(13), 2751–2779 (cit. on p. 57).

Zeng, H.-B., Liu, X.-G., & Wang, W. (2019). A generalized free-matrix-based integral inequality for stability analysis of time-varying delay systems. *Applied Mathematics and Computation*, *354*, 1–8 (cit. on p. 31).

Zhang, F., Deb, C., Lee, S. E., Yang, J., & Shah, K. W. (2016). Time series forecasting for building energy consumption using weighted support vector regression with differential evolution optimization technique. *Energy and Buildings*, *126*, 94–103 (cit. on p. 66).

Zhang, J. A., Liu, F., Masouros, C., Heath, R. W., Feng, Z., Zheng, L., & Petropulu, A. (2021). An overview of signal processing techniques for joint communication and radar sensing. *IEEE Journal of Selected Topics in Signal Processing*, *15*(6), 1295–1315 (cit. on p. 75).

Zhang, J., Welch, G., Bishop, G., & Huang, Z. (2013). A two-stage kalman filter approach for robust and real-time power system state estimation. *IEEE Transactions on Sustainable Energy*, *5*(2), 629–636 (cit. on p. 63).

Zhang, J., Yang, H., Li, M., & Wang, Q. (2019). Robust model predictive control for uncertain positive time-delay systems. *International journal of control, automation and systems*, *17*(2), 307–318 (cit. on p. 30).

Zhang, Z., Pan, S., Gao, C., Zhao, T., & Gao, W. (2019). Support vector machine for regional ionospheric delay modeling. *Sensors*, *19*(13), 2947 (cit. on pp. 33, 39).

Zhang, Z., Li, X., & Wang, S. (2023). Amortized bayesian meta-learning with accelerated gradient descent steps. *Applied Sciences*, *13*(15), 8653 (cit. on p. 65).

Zhao, L., Wu, W., & Li, S. (2019). Design and implementation of an iot-based indoor air quality detector with multiple communication interfaces. *IEEE Internet of Things Journal*, *6*(6), 9621–9632 (cit. on p. 27).

Zhao, Y., Fatehi, A., & Huang, B. (2016). A data-driven hybrid arx and markov chain modeling approach to process identification with time-varying time delays. *IEEE Transactions on Industrial Electronics*, *64*(5), 4226–4236 (cit. on pp. 30, 33).

Zhao, Y., Fatehi, A., & Huang, B. (2017). Robust estimation of arx models with time varying time delays using variational bayesian approach. *IEEE transactions on cybernetics*, *48*(2), 532–542 (cit. on pp. 38, 39, 79).

Zhong, Q.-C. (2006). *Robust control of time-delay systems*. Springer Science & Business Media. (Cit. on p. 24).

Appendix

Appendix A.

Appendix

A.1. All Test Data results

The table presents the measured delays for each output signal before and after optimization across different methods: Cross Correlation (CC), Polynomial Regression (PR), ARX Modeling (AR), and Long Short-Term Memory (LSTM). Each method is compared in two stages: before optimization (indicated by '1') and after optimization (indicated by '2'). The filenames and output signals are listed in the first two columns, while the calculated delay values in seconds for each method, pre- and post-optimization, are presented in the subsequent columns.

Table A.1.: Presentation of Results Before and After Optimization for each Method

Filename	**Output signal**	**CC1**	**PR1**	**LR1**	**AR1**	**LM1**	**CC2**	**PR2**	**LR2**	**AR2**	**LM2**
ds1.csv	out1_0sec	0.0	3.0	2.0	2.0	0.0	0.0.0	0.0.0	4.0.0	0.0.0	0.0
ds1.csv	out2_0sec	0.0	3.0	0.0	3.0	0.0	0.0.0	0.0.0	4.0.0	2.0.0	0.0
ds1.csv	out1_2sec_shift_noise	0.0	3.0	2.0	1.0	0.0	2.0.0	2.0.0	1.0.0	1.0.0	0.0
ds1.csv	out1_2sec_shift	0.0	3.0	2.0	1.0	0.0	2.0.0	6.0.0	2.0.0	7.0.0	0.0

.0 Continued on next page

.o **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds1.csv	out1_2sec_shift_filt	0.0	3.0	3.0	3.0	0.0	2.0.0	2.0.0	4.0.0	1.0.0	0.0
ds1.csv	out1_2sec_shift_noise_filt	0.0	3.0	3.0	3.0	0.0	1.0.0	2.0.0	1.0.0	3.0.0	0.0
ds1.csv	out1_4sec_shift	3.0	3.0	2.0	1.0	0.0	4.0.0	4.0.0	3.0.0	4.0.0	0.0
ds1.csv	out1_4sec_shift_noise	3.0	3.0	2.0	1.0	0.0	4.0.0	5.0.0	1.0.0	5.0.0	0.0
ds1.csv	out1_4sec_shift_filt	0.0	3.0	2.0	0.0	0.0	3.0.0	4.0.0	4.0.0	3.0.0	0.0
ds1.csv	out1_4sec_shift_noise_filt	0.0	3.0	2.0	0.0	0.0	3.0.0	6.0.0	4.0.0	4.0.0	0.0
ds1.csv	out1_6sec_shift	0.0	3.0	2.0	0.0	0.0	6.4.0	6.0.0	4.0.0	3.0.0	0.0
ds1.csv	out1_6sec_shift_noise	0.0	3.0	2.0	0.0	0.0	7.0.0	6.0.0	4.0.0	3.0.0	0.0
ds1.csv	out1_6sec_shift_filt	0.0	3.0	2.0	3.0	0.0	7.0.0	6.0.0	4.0.0	0.0.0	0.0
ds1.csv	out1_6sec_shift_noise_filt	0.0	3.0	2.0	3.0	0.0	7.0.0	6.0.0	4.0.0	0.0.0	0.0
ds1.csv	out1_8sec_shift	0.0	3.0	2.0	1.0	0.0	7.5.0	6.0.0	8.0.0	4.0.0	0.0
ds1.csv	out1_8sec_shift_noise	0.0	3.0	2.0	1.0	0.0	7.0.0	6.0.0	4.0.0	4.0.0	0.0
ds1.csv	out1_8sec_shift_filt	0.0	3.0	2.0	1.0	0.0	8.2.0	6.0.0	4.0.0	1.0.0	0.0
ds1.csv	out1_8sec_shift_noise_filt	0.0	3.0	2.0	1.0	0.0	7.0.0	6.0.0	4.0.0	1.0.0	0.0
ds1.csv	out1_10sec_shift	0.0	3.0	2.0	3.0	0.0	7.0.0	6.0.0	4.0.0	8.0.0	0.0
ds1.csv	out1_10sec_shift_noise	0.0	3.0	2.0	3.0	0.0	7.0.0	6.0.0	10.0.0	7.0.0	0.0
ds1.csv	out1_10sec_shift_filt	0.0	3.0	2.0	0.0	0.0	7.0.0	6.0.0	10.0.0	3.0.0	0.0
ds1.csv	out1_10sec_shift_noise_filt	0.0	3.0	2.0	0.0	0.0	10.0.0	6.0.0	10.0.0	3.0.0	0.0
ds1.csv	out2_2sec_shift	0.0	3.0	0.0	0.0	0.0	2.0.0	2.0.0	4.0.0	4.0.0	0.0
ds1.csv	out2_2sec_shift_noise	0.0	3.0	0.0	0.0	0.0	2.0.0	2.0.0	2.0.0	4.0.0	0.0
ds1.csv	out2_2sec_shift_filt	0.0	3.0	0.0	3.0	0.0	2.0.0	2.0.0	3.0.0	2.0.0	0.0
ds1.csv	out2_2sec_shift_noise_filt	0.0	3.0	0.0	3.0	0.0	2.0.0	2.0.0	2.0.0	4.0.0	0.0
ds1.csv	out2_4sec_shift	0.0	3.0	0.0	1.0	0.0	7.0.0	6.0.0	3.0.0	3.0.0	0.0
ds1.csv	out2_4sec_shift_noise	0.0	3.0	0.0	1.0	0.0	7.0.0	6.0.0	3.0.0	3.0.0	0.0
ds1.csv	out2_4sec_shift_filt	0.0	3.0	3.0	0.0	0.0	2.0.0	6.0.0	4.0.0	3.0.0	0.0
ds1.csv	out2_4sec_shift_noise_filt	0.0	3.0	3.0	3.0	0.0	2.0.0	6.0.0	4.0.0	3.0.0	0.0
ds1.csv	out2_6sec_shift	0.0	3.0	0.0	1.0	0.0	2.0.0	6.0.0	4.0.0	7.0.0	0.0
ds1.csv	out2_6sec_shift_noise	0.0	3.0	0.0	0.0	0.0	2.0.0	6.0.0	4.0.0	4.0.0	0.0
ds1.csv	out2_6sec_shift_filt	0.0	3.0	0.0	3.0	0.0	2.0.0	6.0.0	4.0.0	2.0.0	0.0
ds1.csv	out2_8sec_shift	0.0	3.0	0.0	2.0	0.0	2.0.0	6.0.0	4.0.0	1.0.0	0.0
ds1.csv	out2_8sec_shift_noise	0.0	3.0	0.0	0.0	0.0	2.0.0	6.0.0	4.0.0	1.0.0	0.0
ds1.csv	out2_8sec_shift_filt	0.0	3.0	0.0	0.0	0.0	2.0.0	6.0.0	2.0.0	7.0.0	0.0
ds1.csv	out2_8sec_shift_noise_filt	0.0	3.0	0.0	0.0	0.0	2.0.0	6.0.0	2.0.0	7.0.0	0.0

.o Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds1.csv	out2_10sec_shift	0.0	3.0	3.0	3.0	0.0	2.0.0	6.0.0	2.0.0	1.0.0	0.0
ds1.csv	out2_10sec_shift_noise	0.0	3.0	0.0	1.0	0.0	2.0.0	6.0.0	2.0.0	1.0.0	0.0
ds1.csv	out2_10sec_shift_filt	0.0	3.0	0.0	2.0	0.0	2.0.0	6.0.0	2.0.0	3.0.0	0.0
ds1.csv	out2_10sec_shift_noise_filt	0.0	3.0	0.0	2.0	0.0	2.0.0	6.0.0	2.0.0	3.0.0	0.0
ds2.csv	out1_0sec	0.0	3.0	2.0	1.0	0.0	2.0.0	0.0.0	2.0.0	0.0.0	0.0
ds2.csv	out2_0sec	0.0	3.0	2.0	3.0	0.0	2.0.0	0.0.0	2.0.0	0.0.0	0.0
ds2.csv	out1_2sec_shift	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	2.0.0	1.0.0	0.0
ds2.csv	out1_2sec_shift_noise	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	2.0.0	2.0.0	0.0
ds2.csv	out1_2sec_shift_filt	0.0	3.0	2.0	3.0	0.0	2.0.0	6.0.0	2.0.0	2.0.0	0.0
ds2.csv	out1_2sec_shift_noise_filt	0.0	3.0	2.0	3.0	0.0	2.0.0	1.0.0	2.0.0	2.0.0	0.0
ds2.csv	out1_4sec_shift	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	2.0.0	3.0.0	0.0
ds2.csv	out1_4sec_shift_noise	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	2.0.0	3.0.0	0.0
ds2.csv	out1_4sec_shift_filt	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	2.0.0	7.0.0	0.0
ds2.csv	out1_4sec_shift_noise_filt	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	2.0.0	7.0.0	0.0
ds2.csv	out1_6sec_shift	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	2.0.0	3.0.0	0.0
ds2.csv	out1_6sec_shift_noise	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	2.0.0	3.0.0	0.0
ds2.csv	out1_6sec_shift_filt	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	2.0.0	4.0.0	0.0
ds2.csv	out1_6sec_shift_noise_filt	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	2.0.0	4.0.0	0.0
ds2.csv	out1_8sec_shift	0.0	3.0	2.0	1.0	0.0	2.0.0	6.0.0	2.0.0	2.0.0	0.0
ds2.csv	out1_8sec_shift_noise	0.0	3.0	2.0	1.0	0.0	2.0.0	6.0.0	2.0.0	7.0.0	0.0
ds2.csv	out1_8sec_shift_filt	0.0	3.0	2.0	1.0	0.0	2.0.0	6.0.0	2.0.0	1.0.0	0.0
ds2.csv	out1_8sec_shift_noise_filt	0.0	3.0	2.0	1.0	0.0	2.0.0	6.0.0	2.0.0	1.0.0	0.0
ds2.csv	out1_10sec_shift	0.0	3.0	2.0	3.0	0.0	2.0.0	6.0.0	2.0.0	3.0.0	0.0
ds2.csv	out1_10sec_shift_noise	0.0	3.0	2.0	3.0	0.0	2.0.0	6.0.0	2.0.0	3.0.0	0.0
ds2.csv	out1_10sec_shift_filt	0.0	3.0	2.0	3.0	0.0	2.0.0	6.0.0	2.0.0	4.0.0	0.0
ds2.csv	out1_10sec_shift_noise_filt	0.0	3.0	2.0	3.0	0.0	2.0.0	6.0.0	2.0.0	2.0.0	0.0
ds2.csv	out2_2sec_shift	0.0	3.0	3.0	3.0	0.0	2.0.0	6.0.0	2.0.0	1.0.0	0.0
ds2.csv	out2_2sec_shift_noise	0.0	3.0	3.0	3.0	0.0	2.0.0	6.0.0	2.0.0	1.0.0	0.0
ds2.csv	out2_2sec_shift_filt	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	2.0.0	7.0.0	0.0
ds2.csv	out2_2sec_shift_noise_filt	0.0	3.0	2.0	2.0	0.0	2.0.0	6.0.0	2.0.0	4.0.0	0.0
ds2.csv	out2_4sec_shift	0.0	3.0	2.0	3.0	0.0	2.0.0	1.0.0	1.0.0	2.0.0	0.0
ds2.csv	out2_4sec_shift_noise	0.0	3.0	2.0	3.0	0.0	2.0.0	1.0.0	1.0.0	2.0.0	0.0
ds2.csv	out2_4sec_shift_filt	0.0	3.0	2.0	3.0	0.0	2.0.0	1.0.0	1.0.0	7.0.0	0.0

.0 Continued on next page

.0 Table A.1 Presentation of Results Before and After Optimization

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds2.csv	out2_4sec_shift_noise_filt	0.0	3.0	2.0	3.0	0.0	2.0.0	1.0.0	1.0.0	7.0.0	0.0
ds2.csv	out2_6sec_shift	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	1.0.0	3.0.0	0.0
ds2.csv	out2_6sec_shift_noise	0.0	3.0	2.0	2.0	0.0	2.0.0	6.0.0	1.0.0	3.0.0	0.0
ds2.csv	out2_6sec_shift_filt	0.0	3.0	2.0	2.0	0.0	2.0.0	6.0.0	1.0.0	6.0.0	0.0
ds2.csv	out2_6sec_shift_noise_filt	0.0	3.0	2.0	2.0	0.0	2.0.0	6.0.0	1.0.0	6.0.0	0.0
ds2.csv	out2_8sec_shift	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	1.0.0	0.0.0	0.0
ds2.csv	out2_8sec_shift_noise	0.0	3.0	2.0	1.0	0.0	2.0.0	6.0.0	1.0.0	6.0.0	0.0
ds2.csv	out2_8sec_shift_filt	0.0	3.0	2.0	1.0	0.0	2.0.0	6.0.0	1.0.0	3.0.0	0.0
ds2.csv	out2_8sec_shift_noise_filt	0.0	3.0	2.0	1.0	0.0	2.0.0	6.0.0	1.0.0	3.0.0	0.0
ds2.csv	out2_10sec_shift	0.0	3.0	2.0	3.0	0.0	2.0.0	6.0.0	1.0.0	3.0.0	0.0
ds2.csv	out2_10sec_shift_noise	0.0	3.0	2.0	1.0	0.0	2.0.0	6.0.0	1.0.0	3.0.0	0.0
ds2.csv	out2_10sec_shift_filt	0.0	3.0	2.0	3.0	0.0	2.0.0	6.0.0	1.0.0	0.0.0	0.0
ds2.csv	out2_10sec_shift_noise_filt	0.0	3.0	2.0	3.0	0.0	2.0.0	10.0.0	10.0.0	0.0.0	0.0
ds3.csv	out1_0sec	0.0	3.0	2.0	1.0	0.0	0.0.0	2.0.0	0.0.0	0.0.0	0.0
ds3.csv	out2_0sec	0.0	3.0	2.0	3.0	0.0	0.0.0	0.0.0	1.0.0	0.0.0	0.0
ds3.csv	out1_2sec_shift	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	1.0.0	2.0.0	0.0
ds3.csv	out1_2sec_shift_noise	0.0	3.0	2.0	0.0	0.0	2.0.0	6.0.0	1.0.0	2.0.0	0.0
ds3.csv	out1_2sec_shift_filt	0.0	3.0	3.0	0.0	0.0	5.0.0	6.0.0	1.0.0	0.0.0	0.0
ds3.csv	out1_2sec_shift_noise_filt	0.0	3.0	3.0	3.0	0.0	5.0.0	6.0.0	1.0.0	0.0.0	0.0
ds3.csv	out1_4sec_shift	0.0	3.0	3.0	1.0	0.0	5.0.0	6.0.0	1.0.0	0.0.0	0.0
ds3.csv	out1_4sec_shift_noise	0.0	3.0	3.0	1.0	0.0	5.0.0	6.0.0	1.0.0	0.0.0	0.0
ds3.csv	out1_4sec_shift_filt	0.0	3.0	3.0	1.0	0.0	4.0.0	6.0.0	1.0.0	0.0.0	0.0
ds3.csv	out1_4sec_shift_noise_filt	0.0	3.0	3.0	1.0	0.0	4.0.0	6.0.0	1.0.0	0.0.0	0.0
ds3.csv	out1_6sec_shift	0.0	3.0	2.0	1.0	0.0	4.0.0	6.0.0	1.0.0	7.0.0	0.0
ds3.csv	out1_6sec_shift_noise	0.0	3.0	2.0	1.0	0.0	4.0.0	6.0.0	1.0.0	7.0.0	0.0
ds3.csv	out1_6sec_shift_filt	0.0	3.0	2.0	0.0	0.0	4.0.0	6.0.0	1.0.0	3.0.0	0.0
ds3.csv	out1_6sec_shift_noise_filt	0.0	3.0	2.0	3.0	0.0	4.0.0	6.0.0	1.0.0	3.0.0	0.0
ds3.csv	out1_8sec_shift	0.0	3.0	2.0	0.0	0.0	4.0.0	6.0.0	1.0.0	0.0.0	0.0
ds3.csv	out1_8sec_shift_noise	0.0	3.0	2.0	0.0	0.0	4.0.0	6.0.0	1.0.0	0.0.0	0.0
ds3.csv	out1_8sec_shift_filt	0.0	3.0	2.0	2.0	0.0	4.0.0	6.0.0	1.0.0	3.0.0	0.0
ds3.csv	out1_8sec_shift_noise_filt	0.0	3.0	2.0	1.0	0.0	4.0.0	6.0.0	1.0.0	3.0.0	0.0
ds3.csv	out1_10sec_shift	0.0	3.0	2.0	1.0	0.0	4.0.0	6.0.0	1.0.0	2.0.0	0.0
ds3.csv	out1_10sec_shift_noise	0.0	3.0	2.0	1.0	0.0	4.0.0	6.0.0	1.0.0	2.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds3.csv	out1_10sec_shift_filt	0.0	3.0	2.0	3.0	0.0	4.0.0	6.0.0	1.0.0	3.0.0	0.0
ds3.csv	out1_10sec_shift_noise_filt	0.0	3.0	2.0	3.0	0.0	4.0.0	6.0.0	1.0.0	3.0.0	0.0
ds3.csv	out2_2sec_shift	0.0	3.0	2.0	0.0	0.0	4.0.0	6.0.0	1.0.0	1.0.0	0.0
ds3.csv	out2_2sec_shift_noise	0.0	3.0	2.0	3.0	0.0	4.0.0	6.0.0	1.0.0	1.0.0	0.0
ds3.csv	out2_2sec_shift_filt	0.0	3.0	2.0	2.0	0.0	1.0.0	6.0.0	1.0.0	6.0.0	0.0
ds3.csv	out2_2sec_shift_noise_filt	0.0	3.0	2.0	0.0	0.0	2.0.0	4.0.0	0.0.0	5.0.0	0.0
ds3.csv	out2_4sec_shift	0.0	3.0	2.0	1.0	0.0	1.0.0	6.0.0	1.0.0	6.0.0	0.0
ds3.csv	out2_4sec_shift_noise	0.0	3.0	2.0	1.0	0.0	3.0.0	6.0.0	0.0.0	6.0.0	0.0
ds3.csv	out2_4sec_shift_filt	0.0	3.0	2.0	3.0	0.0	3.0.0	6.0.0	0.0.0	5.0.0	0.0
ds3.csv	out2_4sec_shift_noise_filt	0.0	3.0	2.0	3.0	0.0	7.0.0	6.0.0	0.0.0	3.0.0	0.0
ds3.csv	out2_6sec_shift	0.0	3.0	3.0	1.0	0.0	1.0.0	6.0.0	0.0.0	3.0.0	0.0
ds3.csv	out2_6sec_shift_noise	0.0	3.0	3.0	1.0	0.0	2.0.0	4.0.0	1.0.0	5.0.0	0.0
ds3.csv	out2_6sec_shift_filt	0.0	3.0	2.0	1.0	0.0	1.0.0	6.0.0	0.0.0	2.0.0	0.0
ds3.csv	out2_6sec_shift_noise_filt	0.0	3.0	2.0	0.0	0.0	3.0.0	6.0.0	1.0.0	1.0.0	0.0
ds3.csv	out2_8sec_shift	0.0	3.0	3.0	1.0	0.0	1.0.0	6.0.0	0.0.0	7.0.0	0.0
ds3.csv	out2_8sec_shift_noise	0.0	3.0	3.0	1.0	0.0	3.0.0	4.0.0	1.0.0	3.0.0	0.0
ds3.csv	out2_8sec_shift_filt	0.0	3.0	3.0	1.0	0.0	1.0.0	6.0.0	1.0.0	6.0.0	0.0
ds3.csv	out2_8sec_shift_noise_filt	0.0	3.0	2.0	1.0	0.0	3.0.0	6.0.0	0.0.0	6.0.0	0.0
ds3.csv	out2_10sec_shift	0.0	3.0	2.0	3.0	0.0	1.0.0	6.0.0	1.0.0	3.0.0	0.0
ds3.csv	out2_10sec_shift_noise	0.0	3.0	2.0	3.0	0.0	10.0.0	4.0.0	10.0.0	10.0.0	0.0
ds3.csv	out2_10sec_shift_filt	0.0	3.0	3.0	2.0	0.0	10.0.0	10.0.0	1.0.0	10.0.0	0.0
ds3.csv	out2_10sec_shift_noise_filt	0.0	3.0	3.0	2.0	0.0	7.0.0	6.0.0	1.0.0	6.0.0	0.0
ds4.csv	out1_0sec	7.0	6.0	4.0	7.0	0.0	1.0.0	0.0.0	1.0.0	5.0.0	0.0
ds4.csv	out2_0sec	3.0	6.0	4.0	2.0	0.0	3.0.0	3.0.0	0.0.0	4.0.0	0.0
ds4.csv	out3_0sec	3.0	6.0	2.0	7.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds4.csv	out4_0sec	3.0	1.0	1.0	1.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds4.csv	out5_0sec	5.0	6.0	4.0	1.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds4.csv	out6_0sec	1.0	6.0	1.0	3.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds4.csv	out7_0sec	6.0	6.0	3.0	7.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds4.csv	out8_0sec	5.0	6.0	1.0	5.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds4.csv	out1_2sec_shift	7.0	6.0	4.0	3.0	0.0	2.0.0	2.0.0	3.0.0	2.0.0	0.0
ds4.csv	out1_2sec_shift_noise	7.0	6.0	4.0	3.0	0.0	2.0.0	6.0.0	3.0.0	2.0.0	0.0
ds4.csv	out1_2sec_shift_filt	7.0	6.0	4.0	3.0	0.0	2.0.0	6.0.0	3.0.0	2.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds4.csv	out1_2sec_shift_noise_filt	7.0	6.0	4.0	3.0	0.0	2.0.0	6.0.0	3.0.0	2.0.0	0.0
ds4.csv	out1_4sec_shift	7.0	6.0	4.0	0.0	0.0	4.0.0	6.0.0	3.0.0	4.0.0	0.0
ds4.csv	out1_4sec_shift_noise	7.0	6.0	4.0	0.0	0.0	6.0.0	6.0.0	3.0.0	0.0.0	0.0
ds4.csv	out1_4sec_shift_filt	7.0	6.0	4.0	4.0	0.0	6.0.0	6.0.0	3.0.0	3.0.0	0.0
ds4.csv	out1_4sec_shift_noise_filt	7.0	6.0	4.0	4.0	0.0	6.0.0	6.0.0	3.0.0	3.0.0	0.0
ds4.csv	out1_6sec_shift	7.0	6.0	4.0	1.0	0.0	6.0.0	6.0.0	3.0.0	1.0.0	0.0
ds4.csv	out1_6sec_shift_noise	7.0	6.0	4.0	1.0	0.0	6.0.0	6.0.0	3.0.0	1.0.0	0.0
ds4.csv	out1_6sec_shift_filt	7.0	6.0	4.0	7.0	0.0	6.0.0	6.0.0	3.0.0	0.0.0	0.0
ds4.csv	out1_6sec_shift_noise_filt	7.0	6.0	4.0	7.0	0.0	6.0.0	6.0.0	3.0.0	0.0.0	0.0
ds4.csv	out1_8sec_shift	7.0	6.0	4.0	3.0	0.0	6.0.0	6.0.0	3.0.0	6.0.0	0.0
ds4.csv	out1_8sec_shift_noise	7.0	6.0	4.0	3.0	0.0	6.0.0	6.0.0	3.0.0	6.0.0	0.0
ds4.csv	out1_8sec_shift_filt	7.0	6.0	4.0	4.0	0.0	5.0.0	6.0.0	1.0.0	6.0.0	0.0
ds4.csv	out1_8sec_shift_noise_filt	7.0	6.0	4.0	4.0	0.0	5.0.0	6.0.0	1.0.0	6.0.0	0.0
ds4.csv	out1_10sec_shift	7.0	6.0	3.0	2.0	0.0	5.0.0	6.0.0	1.0.0	6.0.0	0.0
ds4.csv	out1_10sec_shift_noise	7.0	6.0	3.0	2.0	0.0	5.0.0	6.0.0	1.0.0	5.0.0	0.0
ds4.csv	out1_10sec_shift_filt	7.0	6.0	3.0	3.0	0.0	5.0.0	6.0.0	1.0.0	1.0.0	0.0
ds4.csv	out1_10sec_shift_noise_filt	7.0	6.0	3.0	3.0	0.0	5.0.0	6.0.0	1.0.0	1.0.0	0.0
ds4.csv	out2_2sec_shift	2.0	6.0	4.0	3.0	0.0	5.0.0	6.0.0	1.0.0	1.0.0	0.0
ds4.csv	out2_2sec_shift_noise	2.0	6.0	4.0	3.0	0.0	5.0.0	6.0.0	1.0.0	1.0.0	0.0
ds4.csv	out2_2sec_shift_filt	2.0	6.0	4.0	7.0	0.0	5.0.0	6.0.0	0.0.0	4.0.0	0.0
ds4.csv	out2_2sec_shift_noise_filt	2.0	6.0	4.0	4.0	0.0	5.0.0	6.0.0	0.0.0	4.0.0	0.0
ds4.csv	out2_4sec_shift	2.0	6.0	4.0	2.0	0.0	5.0.0	6.0.0	0.0.0	4.0.0	0.0
ds4.csv	out2_4sec_shift_noise	2.0	6.0	4.0	2.0	0.0	5.0.0	6.0.0	0.0.0	4.0.0	0.0
ds4.csv	out2_4sec_shift_filt	2.0	6.0	4.0	1.0	0.0	5.0.0	6.0.0	0.0.0	5.0.0	0.0
ds4.csv	out2_4sec_shift_noise_filt	2.0	6.0	4.0	1.0	0.0	5.0.0	6.0.0	0.0.0	5.0.0	0.0
ds4.csv	out2_6sec_shift	2.0	6.0	2.0	7.0	0.0	5.0.0	6.0.0	0.0.0	1.0.0	0.0
ds4.csv	out2_6sec_shift_noise	2.0	6.0	2.0	7.0	0.0	5.0.0	6.0.0	0.0.0	1.0.0	0.0
ds4.csv	out2_6sec_shift_filt	2.0	6.0	2.0	1.0	0.0	5.0.0	6.0.0	1.0.0	6.0.0	0.0
ds4.csv	out2_6sec_shift_noise_filt	2.0	6.0	2.0	1.0	0.0	5.0.0	6.0.0	1.0.0	4.0.0	0.0
ds4.csv	out2_8sec_shift	2.0	6.0	2.0	3.0	0.0	5.0.0	6.0.0	1.0.0	6.0.0	0.0
ds4.csv	out2_8sec_shift_noise	2.0	6.0	2.0	3.0	0.0	5.0.0	6.0.0	1.0.0	6.0.0	0.0
ds4.csv	out2_8sec_shift_filt	2.0	6.0	2.0	5.0	0.0	6.0.0	6.0.0	1.0.0	4.0.0	0.0
ds4.csv	out2_8sec_shift_noise_filt	2.0	6.0	2.0	5.0	0.0	1.0.0	6.0.0	2.0.0	0.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds4.csv	out2_10sec_shift	2.0	6.0	2.0	1.0	0.0	1.0.0	6.0.0	2.0.0	0.0.0	0.0
ds4.csv	out2_10sec_shift_noise	2.0	6.0	2.0	7.0	0.0	6.0.0	6.0.0	3.0.0	2.0.0	0.0
ds4.csv	out2_10sec_shift_filt	2.0	6.0	2.0	2.0	0.0	4.0.0	6.0.0	3.0.0	1.0.0	0.0
ds4.csv	out2_10sec_shift_noise_filt	2.0	6.0	2.0	2.0	0.0	3.0.0	1.0.0	4.0.0	0.0.0	0.0
ds4.csv	out3_2sec_shift	2.0	6.0	2.0	3.0	0.0	3.0.0	6.0.0	2.0.0	7.0.0	0.0
ds4.csv	out3_2sec_shift_noise	2.0	6.0	2.0	3.0	0.0	4.0.0	6.0.0	2.0.0	4.0.0	0.0
ds4.csv	out3_2sec_shift_filt	2.0	6.0	2.0	7.0	0.0	6.0.0	6.0.0	1.0.0	2.0.0	0.0
ds4.csv	out3_2sec_shift_noise_filt	2.0	6.0	2.0	7.0	0.0	6.0.0	6.0.0	1.0.0	2.0.0	0.0
ds4.csv	out3_4sec_shift	2.0	6.0	2.0	3.0	0.0	6.0.0	6.0.0	1.0.0	4.0.0	0.0
ds4.csv	out3_4sec_shift_noise	2.0	6.0	2.0	3.0	0.0	6.0.0	6.0.0	1.0.0	4.0.0	0.0
ds4.csv	out3_4sec_shift_filt	2.0	6.0	2.0	4.0	0.0	6.0.0	6.0.0	1.0.0	0.0.0	0.0
ds4.csv	out3_4sec_shift_noise_filt	2.0	6.0	2.0	4.0	0.0	6.0.0	6.0.0	1.0.0	0.0.0	0.0
ds4.csv	out3_6sec_shift	2.0	6.0	2.0	2.0	0.0	6.0.0	6.0.0	1.0.0	1.0.0	0.0
ds4.csv	out3_6sec_shift_noise	2.0	6.0	2.0	7.0	0.0	6.0.0	6.0.0	1.0.0	1.0.0	0.0
ds4.csv	out3_6sec_shift_filt	2.0	6.0	2.0	1.0	0.0	6.0.0	6.0.0	1.0.0	7.0.0	0.0
ds4.csv	out3_6sec_shift_noise_filt	2.0	6.0	2.0	1.0	0.0	6.0.0	6.0.0	1.0.0	7.0.0	0.0
ds4.csv	out3_8sec_shift	2.0	6.0	2.0	3.0	0.0	6.0.0	6.0.0	1.0.0	0.0.0	0.0
ds4.csv	out3_8sec_shift_noise	2.0	6.0	2.0	3.0	0.0	6.0.0	6.0.0	1.0.0	0.0.0	0.0
ds4.csv	out3_8sec_shift_filt	2.0	6.0	2.0	4.0	0.0	6.0.0	6.0.0	1.0.0	2.0.0	0.0
ds4.csv	out3_8sec_shift_noise_filt	2.0	6.0	2.0	2.0	0.0	6.0.0	6.0.0	1.0.0	2.0.0	0.0
ds4.csv	out3_10sec_shift	2.0	6.0	2.0	1.0	0.0	6.0.0	6.0.0	1.0.0	0.0.0	0.0
ds4.csv	out3_10sec_shift_noise	2.0	6.0	2.0	1.0	0.0	6.0.0	6.0.0	1.0.0	0.0.0	0.0
ds4.csv	out3_10sec_shift_filt	2.0	6.0	2.0	7.0	0.0	6.0.0	6.0.0	1.0.0	0.0.0	0.0
ds4.csv	out3_10sec_shift_noise_filt	2.0	6.0	2.0	4.0	0.0	6.0.0	6.0.0	1.0.0	0.0.0	0.0
ds4.csv	out4_2sec_shift	2.0	1.0	1.0	2.0	0.0	6.0.0	6.0.0	1.0.0	1.0.0	0.0
ds4.csv	out4_2sec_shift_noise	2.0	1.0	1.0	2.0	0.0	6.0.0	6.0.0	1.0.0	1.0.0	0.0
ds4.csv	out4_2sec_shift_filt	2.0	1.0	1.0	7.0	0.0	1.0.0	6.0.0	2.0.0	3.0.0	0.0
ds4.csv	out4_2sec_shift_noise_filt	2.0	1.0	1.0	7.0	0.0	1.0.0	6.0.0	2.0.0	3.0.0	0.0
ds4.csv	out4_4sec_shift	2.0	6.0	1.0	3.0	0.0	1.0.0	6.0.0	2.0.0	1.0.0	0.0
ds4.csv	out4_4sec_shift_noise	2.0	6.0	1.0	3.0	0.0	1.0.0	6.0.0	2.0.0	1.0.0	0.0
ds4.csv	out4_4sec_shift_filt	2.0	6.0	1.0	6.0	0.0	1.0.0	6.0.0	2.0.0	5.0.0	0.0
ds4.csv	out4_4sec_shift_noise_filt	2.0	6.0	1.0	6.0	0.0	1.0.0	6.0.0	2.0.0	5.0.0	0.0
ds4.csv	out4_6sec_shift	2.0	6.0	1.0	0.0	0.0	1.0.0	6.0.0	2.0.0	1.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds4.csv	out4_6sec_shift_noise	2.0	6.0	1.0	6.0	0.0	1.0.0	6.0.0	2.0.0	0.0.0	0.0
ds4.csv	out4_6sec_shift_filt	2.0	6.0	1.0	3.0	0.0	1.0.0	6.0.0	2.0.0	1.0.0	0.0
ds4.csv	out4_6sec_shift_noise_filt	2.0	6.0	1.0	3.0	0.0	1.0.0	6.0.0	2.0.0	7.0.0	0.0
ds4.csv	out4_8sec_shift	2.0	6.0	1.0	3.0	0.0	1.0.0	6.0.0	8.0.0	0.0.0	0.0
ds4.csv	out4_8sec_shift_noise	2.0	6.0	1.0	3.0	0.0	1.0.0	6.0.0	8.0.0	2.0.0	0.0
ds4.csv	out4_8sec_shift_filt	2.0	6.0	1.0	0.0	0.0	1.0.0	8.0.0	8.0.0	7.0.0	0.0
ds4.csv	out4_8sec_shift_noise_filt	2.0	6.0	1.0	0.0	0.0	1.0.0	6.0.0	2.0.0	3.0.0	0.0
ds4.csv	out4_10sec_shift	2.0	6.0	1.0	0.0	0.0	1.0.0	6.0.0	10.0.0	10.0.0	0.0
ds4.csv	out4_10sec_shift_noise	2.0	6.0	1.0	7.0	0.0	10.0.0	6.0.0	10.0.0	2.0.0	0.0
ds4.csv	out4_10sec_shift_filt	2.0	6.0	1.0	2.0	0.0	10.0.0	6.0.0	8.0.0	10.0.0	0.0
ds4.csv	out4_10sec_shift_noise_filt	2.0	6.0	1.0	2.0	0.0	10.0.0	6.0.0	8.0.0	10.0.0	0.0
ds4.csv	out5_2sec_shift	5.0	6.0	1.0	0.0	0.0	1.0.0	6.0.0	2.0.0	3.0.0	0.0
ds4.csv	out5_2sec_shift_noise	5.0	6.0	1.0	0.0	0.0	1.0.0	6.0.0	2.0.0	3.0.0	0.0
ds4.csv	out5_2sec_shift_filt	5.0	6.0	1.0	0.0	0.0	1.0.0	6.0.0	2.0.0	2.0.0	0.0
ds4.csv	out5_2sec_shift_noise_filt	5.0	6.0	1.0	0.0	0.0	1.0.0	6.0.0	2.0.0	2.0.0	0.0
ds4.csv	out5_4sec_shift	4.0	6.0	1.0	0.0	0.0	1.0.0	6.0.0	2.0.0	1.0.0	0.0
ds4.csv	out5_4sec_shift_noise	4.0	6.0	1.0	0.0	0.0	1.0.0	6.0.0	2.0.0	0.0.0	0.0
ds4.csv	out5_4sec_shift_filt	4.0	6.0	1.0	7.0	0.0	1.0.0	6.0.0	2.0.0	2.0.0	0.0
ds4.csv	out5_4sec_shift_noise_filt	4.0	6.0	1.0	7.0	0.0	1.0.0	6.0.0	2.0.0	2.0.0	0.0
ds4.csv	out5_6sec_shift	4.0	6.0	1.0	3.0	0.0	1.0.0	6.0.0	2.0.0	3.0.0	0.0
ds4.csv	out5_6sec_shift_noise	4.0	6.0	1.0	3.0	0.0	1.0.0	6.0.0	2.0.0	1.0.0	0.0
ds4.csv	out5_6sec_shift_filt	4.0	6.0	1.0	0.0	0.0	1.0.0	6.0.0	2.0.0	0.0.0	0.0
ds4.csv	out5_6sec_shift_noise_filt	4.0	6.0	1.0	0.0	0.0	1.0.0	6.0.0	2.0.0	3.0.0	0.0
ds4.csv	out5_8sec_shift	4.0	6.0	1.0	3.0	0.0	1.0.0	6.0.0	2.0.0	7.0.0	0.0
ds4.csv	out5_8sec_shift_noise	4.0	6.0	1.0	3.0	0.0	8.0.0	6.0.0	2.0.0	7.0.0	0.0
ds4.csv	out5_8sec_shift_filt	4.0	6.0	1.0	2.0	0.0	8.0.0	6.0.0	2.0.0	7.0.0	0.0
ds4.csv	out5_8sec_shift_noise_filt	4.0	6.0	1.0	2.0	0.0	1.0.0	6.0.0	8.0.0	7.0.0	0.0
ds4.csv	out5_10sec_shift	4.0	6.0	1.0	3.0	0.0	1.0.0	6.0.0	2.0.0	7.0.0	0.0
ds4.csv	out5_10sec_shift_noise	4.0	6.0	1.0	3.0	0.0	1.0.0	10.0.0	2.0.0	7.0.0	0.0
ds4.csv	out5_10sec_shift_filt	4.0	6.0	1.0	1.0	0.0	10.0.0	6.0.0	10.0.0	7.0.0	0.0
ds4.csv	out5_10sec_shift_noise_filt	4.0	6.0	1.0	1.0	0.0	10.0.0	6.0.0	10.0.0	6.0.0	0.0
ds4.csv	out6_2sec_shift	1.0	6.0	1.0	6.0	0.0	1.0.0	6.0.0	2.0.0	3.0.0	0.0
ds4.csv	out6_2sec_shift_noise	2.0	4.0	0.0	5.0	0.0	1.0.0	6.0.0	2.0.0	3.0.0	0.0

.0 Continued on next page

.o **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds4.csv	out6_2sec_shift_filt	1.0	6.0	1.0	6.0	0.0	6.0.0	6.0.0	3.0.0	2.0.0	0.0
ds4.csv	out6_2sec_shift_noise_filt	3.0	6.0	0.0	6.0	0.0	6.0.0	6.0.0	3.0.0	0.0.0	0.0
ds4.csv	out6_4sec_shift	3.0	6.0	0.0	5.0	0.0	6.0.0	6.0.0	3.0.0	6.0.0	0.0
ds4.csv	out6_4sec_shift_noise	7.0	6.0	0.0	3.0	0.0	6.0.0	6.0.0	3.0.0	6.0.0	0.0
ds4.csv	out6_4sec_shift_filt	1.0	6.0	0.0	3.0	0.0	7.0.0	6.0.0	3.0.0	7.0.0	0.0
ds4.csv	out6_4sec_shift_noise_filt	2.0	4.0	1.0	5.0	0.0	7.0.0	6.0.0	3.0.0	2.0.0	0.0
ds4.csv	out6_6sec_shift	1.0	6.0	0.0	2.0	0.0	3.0.0	6.0.0	3.0.0	0.0.0	0.0
ds4.csv	out6_6sec_shift_noise	3.0	6.0	1.0	1.0	0.0	3.0.0	6.0.0	3.0.0	1.0.0	0.0
ds4.csv	out6_6sec_shift_filt	1.0	6.0	0.0	7.0	0.0	3.0.0	6.0.0	3.0.0	7.0.0	0.0
ds4.csv	out6_6sec_shift_noise_filt	3.0	4.0	1.0	3.0	0.0	7.0.0	6.0.0	3.0.0	7.0.0	0.0
ds4.csv	out6_8sec_shift	1.0	6.0	1.0	6.0	0.0	3.0.0	6.0.0	3.0.0	6.0.0	0.0
ds4.csv	out6_8sec_shift_noise	3.0	6.0	0.0	6.0	0.0	3.0.0	6.0.0	3.0.0	6.0.0	0.0
ds4.csv	out6_8sec_shift_filt	1.0	6.0	1.0	3.0	0.0	3.0.0	6.0.0	3.0.0	0.0.0	0.0
ds4.csv	out6_8sec_shift_noise_filt	3.0	4.0	1.0	1.0	0.0	3.0.0	6.0.0	3.0.0	8.0.0	0.0
ds4.csv	out6_10sec_shift	1.0	6.0	1.0	5.0	0.0	3.0.0	6.0.0	10.0.0	7.0.0	0.0
ds4.csv	out6_10sec_shift_noise	7.0	6.0	1.0	6.0	0.0	10.0.0	6.0.0	10.0.0	7.0.0	0.0
ds4.csv	out6_10sec_shift_filt	1.0	6.0	1.0	5.0	0.0	3.0.0	10.0.0	3.0.0	10.0.0	0.0
ds4.csv	out6_10sec_shift_noise_filt	3.0	6.0	1.0	4.0	0.0	3.0.0	6.0.0	3.0.0	7.0.0	0.0
ds4.csv	out7_2sec_shift	6.0	6.0	3.0	0.0	0.0	3.0.0	6.0.0	3.0.0	6.0.0	0.0
ds4.csv	out7_2sec_shift_noise	6.0	6.0	3.0	0.0	0.0	3.0.0	6.0.0	3.0.0	6.0.0	0.0
ds4.csv	out7_2sec_shift_filt	6.0	6.0	3.0	0.0	0.0	3.0.0	6.0.0	2.0.0	3.0.0	0.0
ds4.csv	out7_2sec_shift_noise_filt	6.0	6.0	3.0	0.0	0.0	3.0.0	6.0.0	2.0.0	3.0.0	0.0
ds4.csv	out7_4sec_shift	6.0	6.0	3.0	0.0	0.0	3.0.0	6.0.0	2.0.0	4.0.0	0.0
ds4.csv	out7_4sec_shift_noise	6.0	6.0	3.0	0.0	0.0	3.0.0	6.0.0	2.0.0	4.0.0	0.0
ds4.csv	out7_4sec_shift_filt	6.0	6.0	3.0	7.0	0.0	3.0.0	6.0.0	2.0.0	6.0.0	0.0
ds4.csv	out7_4sec_shift_noise_filt	6.0	6.0	3.0	7.0	0.0	3.0.0	6.0.0	2.0.0	6.0.0	0.0
ds4.csv	out7_6sec_shift	6.0	6.0	3.0	7.0	0.0	3.0.0	6.0.0	2.0.0	7.0.0	0.0
ds4.csv	out7_6sec_shift_noise	6.0	6.0	3.0	7.0	0.0	3.0.0	6.0.0	2.0.0	7.0.0	0.0
ds4.csv	out7_6sec_shift_filt	6.0	6.0	3.0	0.0	0.0	3.0.0	6.0.0	2.0.0	7.0.0	0.0
ds4.csv	out7_6sec_shift_noise_filt	6.0	6.0	3.0	0.0	0.0	3.0.0	6.0.0	2.0.0	7.0.0	0.0
ds4.csv	out7_8sec_shift	6.0	6.0	3.0	3.0	0.0	3.0.0	6.0.0	2.0.0	1.0.0	0.0
ds4.csv	out7_8sec_shift_noise	6.0	6.0	3.0	3.0	0.0	3.0.0	6.0.0	2.0.0	1.0.0	0.0
ds4.csv	out7_8sec_shift_filt	6.0	6.0	3.0	1.0	0.0	3.0.0	6.0.0	3.0.0	2.0.0	0.0

.o Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds4.csv	out7_8sec_shift_noise_filt	6.0	6.0	3.0	1.0	0.0	3.0.0	6.0.0	3.0.0	2.0.0	0.0
ds4.csv	out7_10sec_shift	6.0	6.0	3.0	0.0	0.0	3.0.0	6.0.0	3.0.0	4.0.0	0.0
ds4.csv	out7_10sec_shift_noise	6.0	6.0	3.0	0.0	0.0	3.0.0	6.0.0	3.0.0	4.0.0	0.0
ds4.csv	out7_10sec_shift_filt	6.0	6.0	3.0	6.0	0.0	3.0.0	6.0.0	4.0.0	2.0.0	0.0
ds4.csv	out7_10sec_shift_noise_filt	6.0	6.0	3.0	6.0	0.0	3.0.0	6.0.0	4.0.0	3.0.0	0.0
ds4.csv	out8_2sec_shift	5.0	6.0	1.0	6.0	0.0	3.0.0	6.0.0	4.0.0	7.0.0	0.0
ds4.csv	out8_2sec_shift_noise	5.0	6.0	1.0	6.0	0.0	3.0.0	6.0.0	4.0.0	7.0.0	0.0
ds4.csv	out8_2sec_shift_filt	5.0	6.0	1.0	6.0	0.0	3.0.0	1.0.0	4.0.0	1.0.0	0.0
ds4.csv	out8_2sec_shift_noise_filt	5.0	6.0	1.0	5.0	0.0	0.0.0	1.0.0	4.0.0	3.0.0	0.0
ds4.csv	out8_4sec_shift	5.0	6.0	1.0	1.0	0.0	3.0.0	1.0.0	4.0.0	2.0.0	0.0
ds4.csv	out8_4sec_shift_noise	5.0	6.0	1.0	1.0	0.0	3.0.0	1.0.0	4.0.0	2.0.0	0.0
ds4.csv	out8_4sec_shift_filt	5.0	6.0	1.0	1.0	0.0	4.0.0	1.0.0	4.0.0	0.0.0	0.0
ds4.csv	out8_4sec_shift_noise_filt	5.0	6.0	1.0	1.0	0.0	2.0.0	1.0.0	4.0.0	6.0.0	0.0
ds4.csv	out8_6sec_shift	5.0	6.0	0.0	4.0	0.0	1.0.0	1.0.0	4.0.0	6.0.0	0.0
ds4.csv	out8_6sec_shift_noise	5.0	6.0	0.0	4.0	0.0	6.0.0	6.0.0	4.0.0	6.0.0	0.0
ds4.csv	out8_6sec_shift_filt	5.0	6.0	0.0	4.0	0.0	1.0.0	1.0.0	4.0.0	6.0.0	0.0
ds4.csv	out8_6sec_shift_noise_filt	5.0	6.0	0.0	4.0	0.0	1.0.0	1.0.0	7.0.0	6.0.0	0.0
ds4.csv	out8_8sec_shift	5.0	6.0	0.0	5.0	0.0	1.0.0	1.0.0	8.0.0	8.0.0	0.0
ds4.csv	out8_8sec_shift_noise	5.0	6.0	0.0	5.0	0.0	2.0.0	8.0.0	8.0.0	4.0.0	0.0
ds4.csv	out8_8sec_shift_filt	5.0	6.0	0.0	1.0	0.0	8.0.0	8.0.0	8.0.0	4.0.0	0.0
ds4.csv	out8_8sec_shift_noise_filt	5.0	6.0	0.0	1.0	0.0	7.0.0	8.0.0	8.0.0	3.0.0	0.0
ds4.csv	out8_10sec_shift	5.0	6.0	1.0	6.0	0.0	10.0.0	1.0.0	10.0.0	10.0.0	0.0
ds4.csv	out8_10sec_shift_noise	5.0	6.0	1.0	4.0	0.0	1.0.0	10.0.0	10.0.0	8.0.0	0.0
ds4.csv	out8_10sec_shift_filt	5.0	6.0	1.0	6.0	0.0	1.0.0	1.0.0	9.5.0	10.0.0	0.0
ds4.csv	out8_10sec_shift_noise_filt	5.0	6.0	1.0	6.0	0.0	8.0.0	10.0.0	10.0.0	1.0.0	0.0
ds5.csv	out1_0sec	6.0	6.0	1.0	4.0	0.0	1.0.0	1.0.0	0.0.0	0.0.0	0.0
ds5.csv	out2_0sec	1.0	6.0	2.0	0.0	0.0	3.0.0	0.0.0	0.0.0	0.0.0	0.0
ds5.csv	out3_0sec	1.0	6.0	2.0	0.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds5.csv	out4_0sec	6.0	6.0	3.0	2.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds5.csv	out5_0sec	4.0	6.0	3.0	1.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds5.csv	out6_0sec	3.0	1.0	4.0	0.0	0.0	0.0.0	0.0.0	2.0.0	0.0.0	0.0
ds5.csv	out7_0sec	3.0	6.0	2.0	7.0	0.0	0.0.0	0.0.0	0.0.0	3.0.0	0.0
ds5.csv	out8_0sec	4.0	6.0	2.0	4.0	0.0	0.0.0	2.0.0	2.0.0	3.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds5.csv	out1_2sec_shift	6.0	6.0	1.0	2.0	0.0	3.0.0	6.0.0	2.0.0	2.0.0	0.0
ds5.csv	out1_2sec_shift_noise	6.0	6.0	1.0	2.0	0.0	3.0.0	6.0.0	2.0.0	2.0.0	0.0
ds5.csv	out1_2sec_shift_filt	6.0	6.0	1.0	4.0	0.0	3.0.0	6.0.0	2.0.0	2.0.0	0.0
ds5.csv	out1_2sec_shift_noise_filt	6.0	6.0	1.0	4.0	0.0	3.0.0	6.0.0	2.0.0	2.0.0	0.0
ds5.csv	out1_4sec_shift	6.0	6.0	1.0	0.0	0.0	3.0.0	6.0.0	2.0.0	3.0.0	0.0
ds5.csv	out1_4sec_shift_noise	6.0	6.0	1.0	0.0	0.0	3.0.0	6.0.0	4.0.0	3.0.0	0.0
ds5.csv	out1_4sec_shift_filt	6.0	6.0	1.0	1.0	0.0	3.0.0	6.0.0	4.0.0	2.0.0	0.0
ds5.csv	out1_4sec_shift_noise_filt	6.0	6.0	1.0	1.0	0.0	3.0.0	6.0.0	4.0.0	2.0.0	0.0
ds5.csv	out1_6sec_shift	6.0	6.0	1.0	7.0	0.0	3.0.0	6.0.0	6.0.0	2.0.0	0.0
ds5.csv	out1_6sec_shift_noise	6.0	6.0	1.0	7.0	0.0	3.0.0	6.0.0	6.0.0	2.0.0	0.0
ds5.csv	out1_6sec_shift_filt	6.0	6.0	1.0	0.0	0.0	3.0.0	6.0.0	6.0.0	1.0.0	0.0
ds5.csv	out1_6sec_shift_noise_filt	6.0	6.0	1.0	0.0	0.0	3.0.0	6.0.0	6.0.0	1.0.0	0.0
ds5.csv	out1_8sec_shift	6.0	6.0	1.0	2.0	0.0	3.0.0	8.0.0	2.0.0	7.0.0	0.0
ds5.csv	out1_8sec_shift_noise	6.0	6.0	1.0	2.0	0.0	8.0.0	8.0.0	8.0.0	7.0.0	0.0
ds5.csv	out1_8sec_shift_filt	6.0	6.0	1.0	0.0	0.0	4.0.0	6.0.0	8.0.0	8.0.0	0.0
ds5.csv	out1_8sec_shift_noise_filt	6.0	6.0	1.0	0.0	0.0	4.0.0	6.0.0	8.0.0	8.0.0	0.0
ds5.csv	out1_10sec_shift	6.0	6.0	1.0	0.0	0.0	4.0.0	6.0.0	10.0.0	5.0.0	0.0
ds5.csv	out1_10sec_shift_noise	6.0	6.0	1.0	0.0	0.0	10.0.0	10.0.0	2.0.0	10.0.0	0.0
ds5.csv	out1_10sec_shift_filt	6.0	6.0	1.0	1.0	0.0	9.3.0	8.0.0	10.0.0	3.0.0	0.0
ds5.csv	out1_10sec_shift_noise_filt	6.0	6.0	1.0	1.0	0.0	10.0.0	6.0.0	2.0.0	10.0.0	0.0
ds5.csv	out2_2sec_shift	1.0	6.0	2.0	3.0	0.0	4.0.0	6.0.0	2.0.0	3.0.0	0.0
ds5.csv	out2_2sec_shift_noise	1.0	6.0	2.0	3.0	0.0	4.0.0	2.0.0	2.0.0	2.0.0	0.0
ds5.csv	out2_2sec_shift_filt	1.0	6.0	2.0	1.0	0.0	4.0.0	6.0.0	5.0.0	6.0.0	0.0
ds5.csv	out2_2sec_shift_noise_filt	1.0	6.0	2.0	1.0	0.0	4.0.0	6.0.0	2.0.0	2.0.0	0.0
ds5.csv	out2_4sec_shift	1.0	6.0	2.0	5.0	0.0	4.0.0	6.0.0	5.0.0	0.0.0	0.0
ds5.csv	out2_4sec_shift_noise	1.0	6.0	2.0	5.0	0.0	4.0.0	6.0.0	5.0.0	0.0.0	0.0
ds5.csv	out2_4sec_shift_filt	1.0	6.0	2.0	1.0	0.0	3.0.0	6.0.0	5.0.0	6.0.0	0.0
ds5.csv	out2_4sec_shift_noise_filt	1.0	6.0	2.0	0.0	0.0	4.0.0	6.0.0	5.0.0	6.0.0	0.0
ds5.csv	out2_6sec_shift	1.0	6.0	2.0	1.0	0.0	3.0.0	6.0.0	5.0.0	4.0.0	0.0
ds5.csv	out2_6sec_shift_noise	1.0	6.0	2.0	7.0	0.0	3.0.0	6.0.0	5.0.0	4.0.0	0.0
ds5.csv	out2_6sec_shift_filt	1.0	6.0	2.0	0.0	0.0	3.0.0	6.0.0	5.0.0	6.0.0	0.0
ds5.csv	out2_6sec_shift_noise_filt	1.0	6.0	2.0	2.0	0.0	3.0.0	6.0.0	5.0.0	6.0.0	0.0
ds5.csv	out2_8sec_shift	1.0	6.0	2.0	7.0	0.0	3.0.0	6.0.0	8.0.0	6.0.0	0.0

.0 Continued on next page

.o **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds5.csv	out2_8sec_shift_noise	1.0	6.0	2.0	3.0	0.0	3.0.0	6.0.0	5.0.0	6.0.0	0.0
ds5.csv	out2_8sec_shift_filt	1.0	6.0	2.0	1.0	0.0	0.0.0	3.0.0	8.0.0	1.0.0	0.0
ds5.csv	out2_8sec_shift_noise_filt	1.0	6.0	2.0	2.0	0.0	8.0.0	3.0.0	1.0.0	2.0.0	0.0
ds5.csv	out2_10sec_shift	1.0	6.0	2.0	3.0	0.0	4.0.0	3.0.0	2.0.0	1.0.0	0.0
ds5.csv	out2_10sec_shift_noise	1.0	6.0	2.0	3.0	0.0	10.0.0	3.0.0	2.0.0	3.0.0	0.0
ds5.csv	out2_10sec_shift_filt	1.0	6.0	2.0	3.0	0.0	1.0.0	10.0.0	2.0.0	10.0.0	0.0
ds5.csv	out2_10sec_shift_noise_filt	1.0	6.0	2.0	3.0	0.0	9.0.0	3.0.0	10.0.0	0.0.0	0.0
ds5.csv	out3_2sec_shift	1.0	6.0	2.0	2.0	0.0	4.0.0	3.0.0	2.0.0	0.0.0	0.0
ds5.csv	out3_2sec_shift_noise	1.0	6.0	2.0	2.0	0.0	0.0.0	3.0.0	1.0.0	3.0.0	0.0
ds5.csv	out3_2sec_shift_filt	1.0	6.0	2.0	1.0	0.0	0.0.0	3.0.0	2.0.0	2.0.0	0.0
ds5.csv	out3_2sec_shift_noise_filt	1.0	6.0	2.0	0.0	0.0	0.0.0	3.0.0	1.0.0	1.0.0	0.0
ds5.csv	out3_4sec_shift	1.0	6.0	2.0	2.0	0.0	0.0.0	3.0.0	4.0.0	4.0.0	0.0
ds5.csv	out3_4sec_shift_noise	1.0	6.0	2.0	2.0	0.0	0.0.0	4.0.0	1.0.0	4.0.0	0.0
ds5.csv	out3_4sec_shift_filt	1.0	6.0	2.0	3.0	0.0	0.0.0	3.0.0	1.0.0	1.0.0	0.0
ds5.csv	out3_4sec_shift_noise_filt	1.0	6.0	2.0	1.0	0.0	0.0.0	3.0.0	1.0.0	3.0.0	0.0
ds5.csv	out3_6sec_shift	1.0	6.0	2.0	0.0	0.0	0.0.0	3.0.0	1.0.0	6.0.0	0.0
ds5.csv	out3_6sec_shift_noise	1.0	6.0	2.0	3.0	0.0	0.0.0	3.0.0	6.0.0	1.0.0	0.0
ds5.csv	out3_6sec_shift_filt	1.0	6.0	2.0	7.0	0.0	0.0.0	3.0.0	1.0.0	1.0.0	0.0
ds5.csv	out3_6sec_shift_noise_filt	1.0	6.0	2.0	7.0	0.0	6.0.0	3.0.0	1.0.0	1.0.0	0.0
ds5.csv	out3_8sec_shift	1.0	6.0	2.0	7.0	0.0	0.0.0	3.0.0	1.0.0	8.0.0	0.0
ds5.csv	out3_8sec_shift_noise	1.0	6.0	2.0	7.0	0.0	0.0.0	3.0.0	1.0.0	1.0.0	0.0
ds5.csv	out3_8sec_shift_filt	1.0	6.0	2.0	7.0	0.0	8.0.0	8.0.0	1.0.0	1.0.0	0.0
ds5.csv	out3_8sec_shift_noise_filt	1.0	6.0	2.0	7.0	0.0	8.0.0	3.0.0	1.0.0	1.0.0	0.0
ds5.csv	out3_10sec_shift	1.0	6.0	2.0	7.0	0.0	0.0.0	3.0.0	1.0.0	8.5.0	0.0
ds5.csv	out3_10sec_shift_noise	1.0	6.0	2.0	6.0	0.0	0.0.0	10.0.0	1.0.0	10.0.0	0.0
ds5.csv	out3_10sec_shift_filt	1.0	6.0	2.0	3.0	0.0	10.0.0	3.0.0	2.0.0	10.0.0	0.0
ds5.csv	out3_10sec_shift_noise_filt	1.0	6.0	2.0	3.0	0.0	10.0.0	3.0.0	10.0.0	9.3.0	0.0
ds5.csv	out4_2sec_shift	6.0	6.0	3.0	2.0	0.0	0.0.0	3.0.0	1.0.0	2.0.0	0.0
ds5.csv	out4_2sec_shift_noise	6.0	6.0	3.0	0.0	0.0	2.0.0	2.0.0	1.0.0	2.0.0	0.0
ds5.csv	out4_2sec_shift_filt	6.0	6.0	3.0	6.0	0.0	2.0.0	3.0.0	0.0.0	3.0.0	0.0
ds5.csv	out4_2sec_shift_noise_filt	6.0	6.0	3.0	6.0	0.0	2.0.0	3.0.0	1.0.0	2.0.0	0.0
ds5.csv	out4_4sec_shift	7.0	6.0	3.0	7.0	0.0	1.0.0	3.0.0	1.0.0	4.0.0	0.0
ds5.csv	out4_4sec_shift_noise	7.0	6.0	3.0	2.0	0.0	4.0.0	3.0.0	1.0.0	3.0.0	0.0

.o Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds5.csv	out4_4sec_shift_filt	3.0	6.0	3.0	0.0	0.0	1.0.0	3.0.0	1.0.0	4.0.0	0.0
ds5.csv	out4_4sec_shift_noise_filt	3.0	6.0	3.0	1.0	0.0	4.0.0	4.0.0	1.0.0	4.0.0	0.0
ds5.csv	out4_6sec_shift	3.0	6.0	3.0	7.0	0.0	1.0.0	3.0.0	1.0.0	3.0.0	0.0
ds5.csv	out4_6sec_shift_noise	7.0	6.0	3.0	7.0	0.0	1.0.0	3.0.0	1.0.0	2.0.0	0.0
ds5.csv	out4_6sec_shift_filt	3.0	6.0	3.0	6.0	0.0	0.0.0	6.0.0	0.0.0	6.0.0	0.0
ds5.csv	out4_6sec_shift_noise_filt	3.0	6.0	3.0	6.0	0.0	1.0.0	3.0.0	1.0.0	6.0.0	0.0
ds5.csv	out4_8sec_shift	3.0	6.0	3.0	0.0	0.0	0.0.0	8.0.0	8.0.0	1.0.0	0.0
ds5.csv	out4_8sec_shift_noise	3.0	6.0	3.0	3.0	0.0	8.0.0	8.0.0	8.0.0	2.0.0	0.0
ds5.csv	out4_8sec_shift_filt	3.0	6.0	3.0	7.0	0.0	8.0.0	8.0.0	8.0.0	4.0.0	0.0
ds5.csv	out4_8sec_shift_noise_filt	3.0	6.0	3.0	7.0	0.0	8.0.0	3.0.0	8.0.0	1.0.0	0.0
ds5.csv	out4_10sec_shift	3.0	6.0	3.0	2.0	0.0	10.0.0	3.0.0	1.0.0	10.0.0	0.0
ds5.csv	out4_10sec_shift_noise	3.0	6.0	3.0	7.0	0.0	10.0.0	3.0.0	10.0.0	2.0.0	0.0
ds5.csv	out4_10sec_shift_filt	3.0	6.0	3.0	6.0	0.0	10.0.0	3.0.0	10.0.0	10.0.0	0.0
ds5.csv	out4_10sec_shift_noise_filt	3.0	6.0	3.0	6.0	0.0	0.0.0	10.0.0	1.0.0	1.0.0	0.0
ds5.csv	out5_2sec_shift	3.0	6.0	2.0	3.0	0.0	0.0.0	3.0.0	1.0.0	0.0.0	0.0
ds5.csv	out5_2sec_shift_noise	3.0	6.0	2.0	3.0	0.0	4.0.0	3.0.0	2.0.0	1.0.0	0.0
ds5.csv	out5_2sec_shift_filt	3.0	6.0	2.0	4.0	0.0	4.0.0	3.0.0	2.0.0	1.0.0	0.0
ds5.csv	out5_2sec_shift_noise_filt	3.0	6.0	2.0	4.0	0.0	4.0.0	3.0.0	2.0.0	4.0.0	0.0
ds5.csv	out5_4sec_shift	3.0	6.0	2.0	6.0	0.0	4.0.0	2.0.0	2.0.0	3.0.0	0.0
ds5.csv	out5_4sec_shift_noise	3.0	6.0	2.0	6.0	0.0	4.0.0	3.0.0	2.0.0	1.0.0	0.0
ds5.csv	out5_4sec_shift_filt	3.0	6.0	2.0	7.0	0.0	4.0.0	3.0.0	2.0.0	1.0.0	0.0
ds5.csv	out5_4sec_shift_noise_filt	3.0	6.0	2.0	7.0	0.0	4.0.0	3.0.0	2.0.0	4.0.0	0.0
ds5.csv	out5_6sec_shift	3.0	6.0	2.0	7.0	0.0	6.0.0	3.0.0	2.0.0	6.0.0	0.0
ds5.csv	out5_6sec_shift_noise	3.0	6.0	2.0	7.0	0.0	4.0.0	6.0.0	6.0.0	3.0.0	0.0
ds5.csv	out5_6sec_shift_filt	3.0	6.0	2.0	1.0	0.0	4.0.0	6.0.0	2.0.0	3.0.0	0.0
ds5.csv	out5_6sec_shift_noise_filt	3.0	6.0	2.0	1.0	0.0	4.0.0	6.0.0	2.0.0	3.0.0	0.0
ds5.csv	out5_8sec_shift	3.0	6.0	3.0	2.0	0.0	4.0.0	3.0.0	2.0.0	3.0.0	0.0
ds5.csv	out5_8sec_shift_noise	3.0	6.0	3.0	2.0	0.0	8.0.0	3.0.0	2.0.0	8.0.0	0.0
ds5.csv	out5_8sec_shift_filt	3.0	6.0	3.0	4.0	0.0	4.0.0	3.0.0	2.0.0	3.0.0	0.0
ds5.csv	out5_8sec_shift_noise_filt	3.0	6.0	3.0	4.0	0.0	4.0.0	8.0.0	2.0.0	6.0.0	0.0
ds5.csv	out5_10sec_shift	3.0	6.0	4.0	2.0	0.0	10.0.0	3.0.0	2.0.0	10.0.0	0.0
ds5.csv	out5_10sec_shift_noise	3.0	6.0	4.0	3.0	0.0	10.0.0	3.0.0	10.0.0	8.0.0	0.0
ds5.csv	out5_10sec_shift_filt	3.0	6.0	4.0	7.0	0.0	4.0.0	10.0.0	2.0.0	10.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds5.csv	out5_10sec_shift_noise_filt	3.0	6.0	4.0	7.0	0.0	4.0.0	3.0.0	2.0.0	1.0.0	0.0
ds5.csv	out6_2sec_shift	3.0	1.0	4.0	1.0	0.0	4.0.0	3.0.0	2.0.0	3.0.0	0.0
ds5.csv	out6_2sec_shift_noise	0.0	1.0	4.0	3.0	0.0	2.0.0	3.0.0	2.0.0	1.0.0	0.0
ds5.csv	out6_2sec_shift_filt	3.0	1.0	4.0	2.0	0.0	0.0.0	3.0.0	2.0.0	2.0.0	0.0
ds5.csv	out6_2sec_shift_noise_filt	3.0	1.0	4.0	2.0	0.0	2.0.0	2.0.0	2.0.0	3.0.0	0.0
ds5.csv	out6_4sec_shift	1.0	1.0	4.0	0.0	0.0	0.0.0	3.0.0	4.0.0	3.0.0	0.0
ds5.csv	out6_4sec_shift_noise	2.0	1.0	4.0	6.0	0.0	0.0.0	3.0.0	4.0.0	4.0.0	0.0
ds5.csv	out6_4sec_shift_filt	1.0	1.0	4.0	2.0	0.0	0.0.0	3.0.0	4.0.0	3.0.0	0.0
ds5.csv	out6_4sec_shift_noise_filt	1.0	1.0	4.0	5.0	0.0	0.0.0	3.0.0	4.0.0	3.0.0	0.0
ds5.csv	out6_6sec_shift	1.0	1.0	4.0	1.0	0.0	0.0.0	3.0.0	6.0.0	3.0.0	0.0
ds5.csv	out6_6sec_shift_noise	1.0	1.0	7.0	3.0	0.0	6.0.0	3.0.0	2.0.0	6.0.0	0.0
ds5.csv	out6_6sec_shift_filt	1.0	1.0	4.0	4.0	0.0	0.0.0	3.0.0	2.0.0	6.0.0	0.0
ds5.csv	out6_6sec_shift_noise_filt	2.0	1.0	4.0	4.0	0.0	6.0.0	3.0.0	6.0.0	4.0.0	0.0
ds5.csv	out6_8sec_shift	1.0	1.0	4.0	4.0	0.0	6.0.0	3.0.0	2.0.0	8.0.0	0.0
ds5.csv	out6_8sec_shift_noise	7.0	1.0	4.0	3.0	0.0	0.0.0	3.0.0	8.0.0	1.0.0	0.0
ds5.csv	out6_8sec_shift_filt	1.0	1.0	4.0	1.0	0.0	0.0.0	3.0.0	8.0.0	1.0.0	0.0
ds5.csv	out6_8sec_shift_noise_filt	1.0	1.0	1.0	0.0	0.0	8.0.0	3.0.0	2.0.0	8.0.0	0.0
ds5.csv	out6_10sec_shift	1.0	1.0	4.0	2.0	0.0	10.0.0	10.0.0	2.0.0	4.0.0	0.0
ds5.csv	out6_10sec_shift_noise	2.0	1.0	2.0	1.0	0.0	10.0.0	3.0.0	10.0.0	1.0.0	0.0
ds5.csv	out6_10sec_shift_filt	1.0	1.0	4.0	0.0	0.0	10.0.0	3.0.0	10.0.0	10.0.0	0.0
ds5.csv	out6_10sec_shift_noise_filt	3.0	1.0	4.0	0.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds5.csv	out7_2sec_shift	3.0	6.0	2.0	0.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds5.csv	out7_2sec_shift_noise	3.0	6.0	2.0	0.0	0.0	1.0.0	3.0.0	2.0.0	2.0.0	0.0
ds5.csv	out7_2sec_shift_filt	3.0	6.0	2.0	0.0	0.0	1.0.0	3.0.0	2.0.0	2.0.0	0.0
ds5.csv	out7_2sec_shift_noise_filt	3.0	6.0	2.0	0.0	0.0	1.0.0	3.0.0	2.0.0	2.0.0	0.0
ds5.csv	out7_4sec_shift	3.0	6.0	2.0	3.0	0.0	4.0.0	3.0.0	2.0.0	3.0.0	0.0
ds5.csv	out7_4sec_shift_noise	3.0	6.0	2.0	3.0	0.0	4.0.0	3.0.0	4.0.0	4.0.0	0.0
ds5.csv	out7_4sec_shift_filt	3.0	6.0	2.0	7.0	0.0	1.0.0	4.0.0	2.0.0	4.0.0	0.0
ds5.csv	out7_4sec_shift_noise_filt	3.0	6.0	2.0	7.0	0.0	4.0.0	3.0.0	2.0.0	4.0.0	0.0
ds5.csv	out7_6sec_shift	3.0	6.0	2.0	7.0	0.0	6.0.0	6.0.0	6.0.0	6.0.0	0.0
ds5.csv	out7_6sec_shift_noise	3.0	6.0	2.0	7.0	0.0	5.0.0	3.0.0	6.0.0	5.0.0	0.0
ds5.csv	out7_6sec_shift_filt	3.0	6.0	2.0	1.0	0.0	4.0.0	6.0.0	6.0.0	5.0.0	0.0
ds5.csv	out7_6sec_shift_noise_filt	3.0	6.0	2.0	1.0	0.0	6.0.0	3.0.0	2.0.0	6.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds5.csv	out7_8sec_shift	3.0	6.0	2.0	2.0	0.0	8.0.0	3.0.0	2.0.0	8.0.0	0.0
ds5.csv	out7_8sec_shift_noise	3.0	6.0	2.0	2.0	0.0	8.0.0	8.0.0	6.0.0	8.0.0	0.0
ds5.csv	out7_8sec_shift_filt	3.0	6.0	2.0	2.0	0.0	2.0.0	8.0.0	2.0.0	2.0.0	0.0
ds5.csv	out7_8sec_shift_noise_filt	3.0	6.0	2.0	2.0	0.0	8.0.0	6.0.0	2.0.0	8.0.0	0.0
ds5.csv	out7_10sec_shift	3.0	6.0	2.0	1.0	0.0	8.0.0	10.0.0	8.0.0	10.0.0	0.0
ds5.csv	out7_10sec_shift_noise	3.0	6.0	2.0	1.0	0.0	10.0.0	3.0.0	10.0.0	4.0.0	0.0
ds5.csv	out7_10sec_shift_filt	3.0	6.0	2.0	7.0	0.0	10.0.0	3.0.0	10.0.0	10.0.0	0.0
ds5.csv	out7_10sec_shift_noise_filt	3.0	6.0	2.0	7.0	0.0	10.0.0	10.0.0	2.0.0	10.0.0	0.0
ds5.csv	out8_2sec_shift	4.0	6.0	2.0	6.0	0.0	2.0.0	3.0.0	2.0.0	2.0.0	0.0
ds5.csv	out8_2sec_shift_noise	4.0	6.0	2.0	6.0	0.0	4.0.0	3.0.0	4.0.0	2.0.0	0.0
ds5.csv	out8_2sec_shift_filt	4.0	6.0	2.0	5.0	0.0	4.0.0	3.0.0	4.0.0	2.0.0	0.0
ds5.csv	out8_2sec_shift_noise_filt	4.0	6.0	2.0	5.0	0.0	4.0.0	3.0.0	2.0.0	2.0.0	0.0
ds5.csv	out8_4sec_shift	4.0	6.0	2.0	3.0	0.0	4.0.0	3.0.0	4.0.0	3.0.0	0.0
ds5.csv	out8_4sec_shift_noise	4.0	6.0	2.0	6.0	0.0	4.0.0	3.0.0	4.0.0	4.0.0	0.0
ds5.csv	out8_4sec_shift_filt	4.0	6.0	2.0	3.0	0.0	4.0.0	3.0.0	4.0.0	0.0.0	0.0
ds5.csv	out8_4sec_shift_noise_filt	4.0	6.0	2.0	3.0	0.0	4.0.0	3.0.0	4.0.0	0.0.0	0.0
ds5.csv	out8_6sec_shift	4.0	6.0	5.0	6.0	0.0	4.0.0	6.0.0	6.0.0	0.0.0	0.0
ds5.csv	out8_6sec_shift_noise	4.0	6.0	5.0	6.0	0.0	4.0.0	6.0.0	4.0.0	5.0.0	0.0
ds5.csv	out8_6sec_shift_filt	4.0	6.0	5.0	0.0	0.0	6.0.0	6.0.0	4.0.0	6.0.0	0.0
ds5.csv	out8_6sec_shift_noise_filt	4.0	6.0	5.0	0.0	0.0	6.0.0	3.0.0	4.0.0	3.0.0	0.0
ds5.csv	out8_8sec_shift	3.0	6.0	5.0	6.0	0.0	4.0.0	6.0.0	4.0.0	8.0.0	0.0
ds5.csv	out8_8sec_shift_noise	4.0	6.0	5.0	6.0	0.0	8.0.0	3.0.0	8.0.0	3.0.0	0.0
ds5.csv	out8_8sec_shift_filt	3.0	6.0	5.0	4.0	0.0	8.0.0	6.0.0	8.0.0	8.0.0	0.0
ds5.csv	out8_8sec_shift_noise_filt	3.0	6.0	5.0	4.0	0.0	8.0.0	3.0.0	4.0.0	8.0.0	0.0
ds5.csv	out8_10sec_shift	3.0	6.0	5.0	6.0	0.0	4.0.0	3.0.0	4.0.0	3.0.0	0.0
ds5.csv	out8_10sec_shift_noise	3.0	6.0	5.0	6.0	0.0	10.0.0	9.6.0	10.0.0	10.2.0	0.0
ds5.csv	out8_10sec_shift_filt	3.0	6.0	5.0	6.0	0.0	10.0.0	3.0.0	4.0.0	3.0.0	0.0
ds5.csv	out8_10sec_shift_noise_filt	3.0	6.0	5.0	6.0	0.0	10.0.0	3.0.0	4.0.0	10.0.0	0.0
ds6.csv	out1_0sec	0.0	3.0	1.0	1.0	0.0	0.0.0	0.0.0	4.0.0	0.0.0	0.0
ds6.csv	out2_0sec	1.0	3.0	1.0	2.0	0.0	0.0.0	0.0.0	2.0.0	0.0.0	0.0
ds6.csv	out3_0sec	4.0	3.0	2.0	1.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds6.csv	out4_0sec	0.0	3.0	2.0	3.0	0.0	0.0.0	3.0.0	0.0.0	3.0.0	0.0
ds6.csv	out5_0sec	1.0	3.0	2.0	1.0	0.0	0.0.0	0.0.0	0.0.0	3.0.0	0.0

.0 Continued on next page

.o **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds6.csv	out6_0sec	4.0	3.0	4.0	0.0	0.0	4.0.0	0.0.0	0.0.0	0.0.0	0.0
ds6.csv	out7_0sec	4.0	3.0	2.0	0.0	0.0	4.0.0	0.0.0	0.0.0	0.0.0	0.0
ds6.csv	out1_2sec_shift	0.0	3.0	1.0	3.0	0.0	4.0.0	3.0.0	2.0.0	0.0.0	0.0
ds6.csv	out1_2sec_shift_noise	0.0	3.0	1.0	3.0	0.0	4.0.0	3.0.0	2.0.0	0.0.0	0.0
ds6.csv	out1_2sec_shift_filt	0.0	3.0	1.0	1.0	0.0	4.0.0	3.0.0	2.0.0	3.0.0	0.0
ds6.csv	out1_2sec_shift_noise_filt	0.0	3.0	1.0	1.0	0.0	4.0.0	3.0.0	2.0.0	3.0.0	0.0
ds6.csv	out1_4sec_shift	0.0	3.0	1.0	1.0	0.0	4.0.0	3.0.0	2.0.0	3.0.0	0.0
ds6.csv	out1_4sec_shift_noise	0.0	3.0	1.0	1.0	0.0	4.0.0	3.0.0	2.0.0	3.0.0	0.0
ds6.csv	out1_4sec_shift_filt	0.0	3.0	1.0	3.0	0.0	4.0.0	3.0.0	2.0.0	3.0.0	0.0
ds6.csv	out1_4sec_shift_noise_filt	0.0	3.0	1.0	3.0	0.0	4.0.0	3.0.0	2.0.0	3.0.0	0.0
ds6.csv	out1_6sec_shift	0.0	3.0	1.0	1.0	0.0	4.0.0	3.0.0	2.0.0	3.0.0	0.0
ds6.csv	out1_6sec_shift_noise	0.0	3.0	1.0	1.0	0.0	4.0.0	3.0.0	2.0.0	3.0.0	0.0
ds6.csv	out1_6sec_shift_filt	0.0	3.0	1.0	1.0	0.0	4.0.0	3.0.0	0.0.0	3.0.0	0.0
ds6.csv	out1_6sec_shift_noise_filt	0.0	3.0	1.0	1.0	0.0	4.0.0	3.0.0	0.0.0	3.0.0	0.0
ds6.csv	out1_8sec_shift	0.0	3.0	1.0	1.0	0.0	4.0.0	3.0.0	0.0.0	3.0.0	0.0
ds6.csv	out1_8sec_shift_noise	0.0	3.0	1.0	1.0	0.0	4.0.0	3.0.0	0.0.0	3.0.0	0.0
ds6.csv	out1_8sec_shift_filt	0.0	3.0	1.0	1.0	0.0	1.0.0	4.0.0	1.0.0	1.0.0	0.0
ds6.csv	out1_8sec_shift_noise_filt	0.0	3.0	1.0	1.0	0.0	3.0.0	2.0.0	3.0.0	3.0.0	0.0
ds6.csv	out1_10sec_shift	0.0	3.0	1.0	4.0	0.0	3.0.0	10.0.0	10.0.0	2.0.0	0.0
ds6.csv	out1_10sec_shift_noise	0.0	3.0	1.0	4.0	0.0	8.0.0	8.0.0	10.0.0	4.0.0	0.0
ds6.csv	out1_10sec_shift_filt	0.0	3.0	1.0	3.0	0.0	8.0.0	8.0.0	10.0.0	0.0.0	0.0
ds6.csv	out1_10sec_shift_noise_filt	0.0	3.0	1.0	3.0	0.0	10.0.0	8.0.0	10.0.0	3.0.0	0.0
ds6.csv	out2_2sec_shift	1.0	3.0	1.0	2.0	0.0	5.0.0	5.0.0	3.0.0	3.0.0	0.0
ds6.csv	out2_2sec_shift_noise	1.0	3.0	0.0	3.0	0.0	0.0.0	0.0.0	3.0.0	3.0.0	0.0
ds6.csv	out2_2sec_shift_filt	1.0	3.0	1.0	2.0	0.0	1.0.0	4.0.0	1.0.0	0.0.0	0.0
ds6.csv	out2_2sec_shift_noise_filt	1.0	3.0	1.0	2.0	0.0	1.0.0	4.0.0	1.0.0	0.0.0	0.0
ds6.csv	out2_4sec_shift	1.0	3.0	1.0	3.0	0.0	1.0.0	4.0.0	1.0.0	0.0.0	0.0
ds6.csv	out2_4sec_shift_noise	1.0	3.0	1.0	2.0	0.0	1.0.0	4.0.0	1.0.0	0.0.0	0.0
ds6.csv	out2_4sec_shift_filt	1.0	3.0	1.0	1.0	0.0	1.0.0	4.0.0	0.0.0	5.0.0	0.0
ds6.csv	out2_4sec_shift_noise_filt	1.0	3.0	1.0	3.0	0.0	1.0.0	4.0.0	0.0.0	5.0.0	0.0
ds6.csv	out2_6sec_shift	1.0	3.0	1.0	2.0	0.0	1.0.0	4.0.0	0.0.0	2.0.0	0.0
ds6.csv	out2_6sec_shift_noise	0.0	3.0	0.0	4.0	0.0	1.0.0	4.0.0	0.0.0	2.0.0	0.0
ds6.csv	out2_6sec_shift_filt	1.0	3.0	1.0	2.0	0.0	1.0.0	4.0.0	0.0.0	3.0.0	0.0

.o Continued on next page

.o **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds6.csv	out2_6sec_shift_noise_filt	0.0	3.0	1.0	1.0	0.0	1.0.0	4.0.0	0.0.0	3.0.0	0.0
ds6.csv	out2_8sec_shift	1.0	3.0	1.0	2.0	0.0	1.0.0	4.0.0	0.0.0	3.0.0	0.0
ds6.csv	out2_8sec_shift_noise	0.0	3.0	1.0	4.0	0.0	1.0.0	4.0.0	0.0.0	3.0.0	0.0
ds6.csv	out2_8sec_shift_filt	1.0	3.0	1.0	1.0	0.0	1.0.0	4.0.0	0.0.0	3.0.0	0.0
ds6.csv	out2_8sec_shift_noise_filt	1.0	3.0	1.0	3.0	0.0	1.0.0	4.0.0	0.0.0	3.0.0	0.0
ds6.csv	out2_10sec_shift	0.0	3.0	1.0	2.0	0.0	10.0.0	8.0.0	10.0.0	8.0.0	0.0
ds6.csv	out2_10sec_shift_noise	1.0	3.0	1.0	4.0	0.0	10.0.0	8.0.0	0.0.0	8.0.0	0.0
ds6.csv	out2_10sec_shift_filt	0.0	3.0	1.0	1.0	0.0	10.0.0	8.0.0	10.0.0	8.0.0	0.0
ds6.csv	out2_10sec_shift_noise_filt	0.0	3.0	1.0	0.0	0.0	10.0.0	4.0.0	10.0.0	8.0.0	0.0
ds6.csv	out3_2sec_shift	4.0	3.0	2.0	1.0	0.0	1.0.0	4.0.0	1.0.0	0.0.0	0.0
ds6.csv	out3_2sec_shift_noise	4.0	3.0	2.0	1.0	0.0	1.0.0	4.0.0	1.0.0	0.0.0	0.0
ds6.csv	out3_2sec_shift_filt	4.0	3.0	2.0	4.0	0.0	3.0.0	2.0.0	3.0.0	3.0.0	0.0
ds6.csv	out3_2sec_shift_noise_filt	4.0	2.0	2.0	3.0	0.0	4.0.0	2.0.0	3.0.0	2.0.0	0.0
ds6.csv	out3_4sec_shift	4.0	3.0	2.0	1.0	0.0	3.0.0	2.0.0	3.0.0	0.0.0	0.0
ds6.csv	out3_4sec_shift_noise	4.0	3.0	2.0	1.0	0.0	3.0.0	2.0.0	3.0.0	2.0.0	0.0
ds6.csv	out3_4sec_shift_filt	4.0	3.0	2.0	4.0	0.0	3.0.0	2.0.0	3.0.0	4.0.0	0.0
ds6.csv	out3_4sec_shift_noise_filt	4.0	3.0	2.0	1.0	0.0	4.0.0	2.0.0	3.0.0	3.0.0	0.0
ds6.csv	out3_6sec_shift	4.0	3.0	2.0	3.0	0.0	3.0.0	2.0.0	3.0.0	3.0.0	0.0
ds6.csv	out3_6sec_shift_noise	4.0	3.0	2.0	3.0	0.0	0.0.0	2.0.0	6.0.0	6.0.0	0.0
ds6.csv	out3_6sec_shift_filt	4.0	3.0	2.0	3.0	0.0	3.0.0	6.0.0	7.0.0	6.0.0	0.0
ds6.csv	out3_6sec_shift_noise_filt	4.0	3.0	2.0	3.0	0.0	6.0.0	6.0.0	7.0.0	8.0.0	0.0
ds6.csv	out3_8sec_shift	4.0	3.0	2.0	3.0	0.0	3.0.0	2.0.0	8.0.0	7.0.0	0.0
ds6.csv	out3_8sec_shift_noise	4.0	3.0	2.0	3.0	0.0	3.0.0	2.0.0	8.0.0	8.0.0	0.0
ds6.csv	out3_8sec_shift_filt	4.0	3.0	2.0	3.0	0.0	3.0.0	8.0.0	8.0.0	0.0.0	0.0
ds6.csv	out3_8sec_shift_noise_filt	4.0	3.0	2.0	3.0	0.0	8.0.0	2.0.0	1.0.0	1.0.0	0.0
ds6.csv	out3_10sec_shift	4.0	3.0	2.0	3.0	0.0	3.0.0	2.0.0	10.0.0	10.0.0	0.0
ds6.csv	out3_10sec_shift_noise	4.0	3.0	2.0	1.0	0.0	3.0.0	2.0.0	3.0.0	3.0.0	0.0
ds6.csv	out3_10sec_shift_filt	4.0	3.0	2.0	1.0	0.0	3.0.0	2.0.0	10.0.0	9.0.0	0.0
ds6.csv	out3_10sec_shift_noise_filt	4.0	3.0	2.0	3.0	0.0	10.0.0	2.0.0	10.0.0	9.0.0	0.0
ds6.csv	out4_2sec_shift	0.0	3.0	2.0	1.0	0.0	3.0.0	2.0.0	3.0.0	2.0.0	0.0
ds6.csv	out4_2sec_shift_noise	0.0	3.0	2.0	1.0	0.0	0.0.0	2.0.0	2.0.0	4.0.0	0.0
ds6.csv	out4_2sec_shift_filt	0.0	3.0	2.0	3.0	0.0	3.0.0	0.0.0	1.0.0	2.0.0	0.0
ds6.csv	out4_2sec_shift_noise_filt	0.0	3.0	2.0	3.0	0.0	3.0.0	0.0.0	1.0.0	2.0.0	0.0

.o Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds6.csv	out4_4sec_shift	0.0	3.0	2.0	3.0	0.0	0.0.0	4.0.0	4.0.0	4.0.0	0.0
ds6.csv	out4_4sec_shift_noise	0.0	3.0	2.0	3.0	0.0	4.0.0	4.0.0	1.0.0	2.0.0	0.0
ds6.csv	out4_4sec_shift_filt	0.0	3.0	2.0	3.0	0.0	3.0.0	4.0.0	4.0.0	3.0.0	0.0
ds6.csv	out4_4sec_shift_noise_filt	0.0	3.0	2.0	3.0	0.0	4.0.0	4.0.0	3.0.0	4.0.0	0.0
ds6.csv	out4_6sec_shift	0.0	3.0	2.0	1.0	0.0	0.0.0	0.0.0	1.0.0	2.0.0	0.0
ds6.csv	out4_6sec_shift_noise	0.0	3.0	2.0	1.0	0.0	0.0.0	0.0.0	1.0.0	2.0.0	0.0
ds6.csv	out4_6sec_shift_filt	0.0	3.0	2.0	4.0	0.0	0.0.0	0.0.0	1.0.0	2.0.0	0.0
ds6.csv	out4_6sec_shift_noise_filt	0.0	3.0	2.0	4.0	0.0	0.0.0	0.0.0	1.0.0	2.0.0	0.0
ds6.csv	out4_8sec_shift	0.0	3.0	2.0	1.0	0.0	0.0.0	0.0.0	1.0.0	3.0.0	0.0
ds6.csv	out4_8sec_shift_noise	0.0	3.0	2.0	1.0	0.0	8.0.0	0.0.0	1.0.0	3.0.0	0.0
ds6.csv	out4_8sec_shift_filt	0.0	3.0	2.0	4.0	0.0	8.0.0	0.0.0	1.0.0	2.0.0	0.0
ds6.csv	out4_8sec_shift_noise_filt	0.0	3.0	2.0	4.0	0.0	8.0.0	8.0.0	8.0.0	2.0.0	0.0
ds6.csv	out4_10sec_shift	0.0	3.0	2.0	1.0	0.0	0.0.0	0.0.0	1.0.0	2.0.0	0.0
ds6.csv	out4_10sec_shift_noise	0.0	3.0	2.0	1.0	0.0	0.0.0	0.0.0	1.0.0	2.0.0	0.0
ds6.csv	out4_10sec_shift_filt	0.0	3.0	2.0	3.0	0.0	0.0.0	0.0.0	10.0.0	10.0.0	0.0
ds6.csv	out4_10sec_shift_noise_filt	0.0	3.0	2.0	3.0	0.0	0.0.0	10.0.0	10.0.0	1.0.0	0.0
ds6.csv	out5_2sec_shift	1.0	3.0	2.0	2.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds6.csv	out5_2sec_shift_noise	1.0	3.0	2.0	2.0	0.0	2.0.0	2.0.0	1.0.0	2.0.0	0.0
ds6.csv	out5_2sec_shift_filt	1.0	3.0	2.0	3.0	0.0	4.0.0	5.0.0	2.0.0	4.0.0	0.0
ds6.csv	out5_2sec_shift_noise_filt	1.0	3.0	2.0	3.0	0.0	4.0.0	5.0.0	2.0.0	4.0.0	0.0
ds6.csv	out5_4sec_shift	1.0	3.0	2.0	2.0	0.0	4.0.0	5.0.0	2.0.0	4.0.0	0.0
ds6.csv	out5_4sec_shift_noise	1.0	3.0	2.0	2.0	0.0	4.0.0	5.0.0	2.0.0	4.0.0	0.0
ds6.csv	out5_4sec_shift_filt	1.0	3.0	2.0	4.0	0.0	4.0.0	5.0.0	2.0.0	3.0.0	0.0
ds6.csv	out5_4sec_shift_noise_filt	1.0	3.0	2.0	4.0	0.0	4.0.0	5.0.0	2.0.0	3.0.0	0.0
ds6.csv	out5_6sec_shift	1.0	3.0	2.0	2.0	0.0	4.0.0	5.0.0	2.0.0	1.0.0	0.0
ds6.csv	out5_6sec_shift_noise	1.0	3.0	2.0	2.0	0.0	4.0.0	5.0.0	2.0.0	1.0.0	0.0
ds6.csv	out5_6sec_shift_filt	1.0	3.0	2.0	3.0	0.0	4.0.0	5.0.0	2.0.0	3.0.0	0.0
ds6.csv	out5_6sec_shift_noise_filt	1.0	3.0	2.0	3.0	0.0	4.0.0	5.0.0	2.0.0	3.0.0	0.0
ds6.csv	out5_8sec_shift	2.0	3.0	2.0	2.0	0.0	4.0.0	5.0.0	2.0.0	0.0.0	0.0
ds6.csv	out5_8sec_shift_noise	2.0	3.0	2.0	2.0	0.0	4.0.0	5.0.0	2.0.0	0.0.0	0.0
ds6.csv	out5_8sec_shift_filt	2.0	3.0	2.0	4.0	0.0	4.0.0	5.0.0	2.0.0	4.0.0	0.0
ds6.csv	out5_8sec_shift_noise_filt	2.0	3.0	2.0	4.0	0.0	4.0.0	5.0.0	2.0.0	4.0.0	0.0
ds6.csv	out5_10sec_shift	2.0	3.0	2.0	4.0	0.0	4.0.0	5.0.0	2.0.0	3.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds6.csv	out5_10sec_shift_noise	2.0	3.0	2.0	4.0	0.0	4.0.0	5.0.0	2.0.0	3.0.0	0.0
ds6.csv	out5_10sec_shift_filt	2.0	3.0	2.0	4.0	0.0	4.0.0	5.0.0	2.0.0	5.0.0	0.0
ds6.csv	out5_10sec_shift_noise_filt	2.0	3.0	2.0	4.0	0.0	4.0.0	5.0.0	2.0.0	5.0.0	0.0
ds6.csv	out6_2sec_shift	4.0	3.0	4.0	0.0	0.0	4.0.0	5.0.0	2.0.0	5.0.0	0.0
ds6.csv	out6_2sec_shift_noise	4.0	3.0	4.0	0.0	0.0	4.0.0	5.0.0	2.0.0	5.0.0	0.0
ds6.csv	out6_2sec_shift_filt	4.0	3.0	4.0	3.0	0.0	4.0.0	4.0.0	2.0.0	1.0.0	0.0
ds6.csv	out6_2sec_shift_noise_filt	4.0	3.0	4.0	3.0	0.0	4.0.0	4.0.0	2.0.0	1.0.0	0.0
ds6.csv	out6_4sec_shift	4.0	3.0	4.0	0.0	0.0	4.0.0	4.0.0	2.0.0	1.0.0	0.0
ds6.csv	out6_4sec_shift_noise	4.0	3.0	4.0	0.0	0.0	4.0.0	4.0.0	2.0.0	1.0.0	0.0
ds6.csv	out6_4sec_shift_filt	4.0	3.0	4.0	0.0	0.0	4.0.0	4.0.0	2.0.0	3.0.0	0.0
ds6.csv	out6_4sec_shift_noise_filt	4.0	3.0	4.0	0.0	0.0	4.0.0	4.0.0	2.0.0	3.0.0	0.0
ds6.csv	out6_6sec_shift	4.0	3.0	4.0	3.0	0.0	4.0.0	4.0.0	2.0.0	4.0.0	0.0
ds6.csv	out6_6sec_shift_noise	4.0	3.0	4.0	3.0	0.0	4.0.0	4.0.0	2.0.0	4.0.0	0.0
ds6.csv	out6_6sec_shift_filt	4.0	3.0	4.0	3.0	0.0	4.0.0	4.0.0	2.0.0	3.0.0	0.0
ds6.csv	out6_6sec_shift_noise_filt	4.0	3.0	4.0	3.0	0.0	4.0.0	4.0.0	2.0.0	3.0.0	0.0
ds6.csv	out6_8sec_shift	4.0	3.0	4.0	3.0	0.0	4.0.0	4.0.0	2.0.0	1.0.0	0.0
ds6.csv	out6_8sec_shift_noise	4.0	3.0	4.0	3.0	0.0	4.0.0	4.0.0	2.0.0	1.0.0	0.0
ds6.csv	out6_8sec_shift_filt	4.0	3.0	4.0	3.0	0.0	4.0.0	4.0.0	2.0.0	1.0.0	0.0
ds6.csv	out6_8sec_shift_noise_filt	4.0	3.0	4.0	3.0	0.0	4.0.0	4.0.0	2.0.0	1.0.0	0.0
ds6.csv	out6_10sec_shift	4.0	3.0	4.0	3.0	0.0	4.0.0	4.0.0	2.0.0	3.0.0	0.0
ds6.csv	out6_10sec_shift_noise	4.0	3.0	4.0	3.0	0.0	4.0.0	4.0.0	2.0.0	3.0.0	0.0
ds6.csv	out6_10sec_shift_filt	4.0	3.0	4.0	0.0	0.0	0.0.0	4.0.0	2.0.0	1.0.0	0.0
ds6.csv	out6_10sec_shift_noise_filt	4.0	3.0	4.0	0.0	0.0	0.0.0	4.0.0	2.0.0	1.0.0	0.0
ds6.csv	out7_2sec_shift	4.0	3.0	2.0	0.0	0.0	0.0.0	4.0.0	2.0.0	1.0.0	0.0
ds6.csv	out7_2sec_shift_noise	4.0	3.0	2.0	0.0	0.0	0.0.0	4.0.0	2.0.0	1.0.0	0.0
ds6.csv	out7_2sec_shift_filt	4.0	3.0	2.0	3.0	0.0	5.0.0	5.0.0	1.0.0	0.0.0	0.0
ds6.csv	out7_2sec_shift_noise_filt	4.0	3.0	2.0	3.0	0.0	5.0.0	5.0.0	1.0.0	0.0.0	0.0
ds6.csv	out7_4sec_shift	4.0	3.0	2.0	0.0	0.0	5.0.0	5.0.0	1.0.0	2.0.0	0.0
ds6.csv	out7_4sec_shift_noise	4.0	3.0	2.0	0.0	0.0	5.0.0	5.0.0	1.0.0	2.0.0	0.0
ds6.csv	out7_4sec_shift_filt	4.0	3.0	2.0	0.0	0.0	5.0.0	5.0.0	4.0.0	5.0.0	0.0
ds6.csv	out7_4sec_shift_noise_filt	4.0	3.0	2.0	0.0	0.0	5.0.0	5.0.0	4.0.0	4.0.0	0.0
ds6.csv	out7_6sec_shift	4.0	3.0	2.0	3.0	0.0	5.0.0	5.0.0	4.0.0	0.0.0	0.0
ds6.csv	out7_6sec_shift_noise	4.0	3.0	2.0	3.0	0.0	5.0.0	5.0.0	4.0.0	0.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds6.csv	out7_6sec_shift_filt	4.0	3.0	2.0	3.0	0.0	5.0.0	5.0.0	4.0.0	3.0.0	0.0
ds6.csv	out7_6sec_shift_noise_filt	4.0	3.0	2.0	3.0	0.0	5.0.0	5.0.0	4.0.0	4.0.0	0.0
ds6.csv	out7_8sec_shift	4.0	3.0	2.0	3.0	0.0	5.0.0	5.0.0	4.0.0	0.0.0	0.0
ds6.csv	out7_8sec_shift_noise	4.0	3.0	2.0	3.0	0.0	5.0.0	5.0.0	4.0.0	4.0.0	0.0
ds6.csv	out7_8sec_shift_filt	4.0	3.0	2.0	3.0	0.0	5.0.0	5.0.0	4.0.0	5.0.0	0.0
ds6.csv	out7_8sec_shift_noise_filt	4.0	3.0	2.0	3.0	0.0	5.0.0	5.0.0	4.0.0	0.0.0	0.0
ds6.csv	out7_10sec_shift	4.0	3.0	0.0	3.0	0.0	5.0.0	5.0.0	4.0.0	0.0.0	0.0
ds6.csv	out7_10sec_shift_noise	4.0	3.0	0.0	3.0	0.0	5.0.0	5.0.0	4.0.0	5.0.0	0.0
ds6.csv	out7_10sec_shift_filt	4.0	3.0	0.0	3.0	0.0	5.0.0	5.0.0	4.0.0	3.0.0	0.0
ds6.csv	out7_10sec_shift_noise_filt	4.0	3.0	0.0	3.0	0.0	5.0.0	5.0.0	4.0.0	3.0.0	0.0
ds7.csv	out1_0sec	1.0	4.0	1.0	1.0	0.0	5.0.0	0.0.0	4.0.0	0.0.0	0.0
ds7.csv	out2_0sec	3.0	2.0	3.0	3.0	0.0	0.0.0	0.0.0	4.0.0	0.0.0	0.0
ds7.csv	out3_0sec	3.0	0.0	1.0	2.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds7.csv	out4_0sec	4.0	5.0	2.0	4.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds7.csv	out5_0sec	4.0	4.0	2.0	0.0	0.0	0.0.0	3.0.0	0.0.0	0.0.0	0.0
ds7.csv	out6_0sec	5.0	5.0	1.0	3.0	0.0	0.0.0	0.0.0	3.0.0	0.0.0	0.0
ds7.csv	out7_0sec	5.0	5.0	3.0	3.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds7.csv	out8_0sec	0.0	0.0	3.0	3.0	0.0	0.0.0	0.0.0	0.0.0	4.0.0	0.0
ds7.csv	out1_2sec_shift	1.0	4.0	1.0	0.0	0.0	2.0.0	2.0.0	3.0.0	0.0.0	0.0
ds7.csv	out1_2sec_shift_noise	1.0	4.0	1.0	0.0	0.0	2.0.0	2.0.0	3.0.0	0.0.0	0.0
ds7.csv	out1_2sec_shift_filt	1.0	4.0	1.0	0.0	0.0	5.0.0	5.0.0	2.0.0	2.0.0	0.0
ds7.csv	out1_2sec_shift_noise_filt	1.0	4.0	1.0	0.0	0.0	5.0.0	5.0.0	2.0.0	2.0.0	0.0
ds7.csv	out1_4sec_shift	1.0	4.0	0.0	5.0	0.0	4.0.0	4.0.0	3.0.0	2.0.0	0.0
ds7.csv	out1_4sec_shift_noise	1.0	4.0	0.0	5.0	0.0	5.0.0	4.0.0	4.0.0	4.0.0	0.0
ds7.csv	out1_4sec_shift_filt	1.0	4.0	0.0	2.0	0.0	5.0.0	5.0.0	3.0.0	1.0.0	0.0
ds7.csv	out1_4sec_shift_noise_filt	1.0	4.0	0.0	2.0	0.0	5.0.0	5.0.0	3.0.0	1.0.0	0.0
ds7.csv	out1_6sec_shift	1.0	4.0	0.0	3.0	0.0	6.0.0	6.0.0	3.0.0	6.0.0	0.0
ds7.csv	out1_6sec_shift_noise	1.0	4.0	0.0	3.0	0.0	6.0.0	6.0.0	3.0.0	6.0.0	0.0
ds7.csv	out1_6sec_shift_filt	1.0	4.0	0.0	3.0	0.0	5.0.0	5.0.0	3.0.0	4.0.0	0.0
ds7.csv	out1_6sec_shift_noise_filt	1.0	4.0	0.0	3.0	0.0	5.0.0	5.0.0	3.0.0	4.0.0	0.0
ds7.csv	out1_8sec_shift	1.0	4.0	0.0	3.0	0.0	8.0.0	8.0.0	3.0.0	8.0.0	0.0
ds7.csv	out1_8sec_shift_noise	1.0	4.0	0.0	3.0	0.0	8.0.0	8.0.0	6.0.0	8.0.0	0.0
ds7.csv	out1_8sec_shift_filt	1.0	4.0	0.0	2.0	0.0	0.0.0	0.0.0	8.0.0	5.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds7.csv	out1_8sec_shift_noise_filt	1.0	4.0	0.0	2.0	0.0	0.0.0	8.0.0	2.0.0	5.0.0	0.0
ds7.csv	out1_10sec_shift	1.0	4.0	1.0	3.0	0.0	0.0.0	10.0.0	2.0.0	9.0.0	0.0
ds7.csv	out1_10sec_shift_noise	1.0	4.0	1.0	3.0	0.0	10.0.0	0.0.0	8.0.0	9.0.0	0.0
ds7.csv	out1_10sec_shift_filt	1.0	4.0	1.0	0.0	0.0	0.0.0	10.0.0	8.0.0	8.0.0	0.0
ds7.csv	out1_10sec_shift_noise_filt	1.0	4.0	1.0	0.0	0.0	0.0.0	10.0.0	10.0.0	5.0.0	0.0
ds7.csv	out2_2sec_shift	3.0	2.0	3.0	3.0	0.0	0.0.0	0.0.0	2.0.0	1.0.0	0.0
ds7.csv	out2_2sec_shift_noise	4.0	2.0	3.0	2.0	0.0	0.0.0	0.0.0	2.0.0	1.0.0	0.0
ds7.csv	out2_2sec_shift_filt	3.0	2.0	3.0	0.0	0.0	0.0.0	0.0.0	2.0.0	5.0.0	0.0
ds7.csv	out2_2sec_shift_noise_filt	3.0	2.0	3.0	2.0	0.0	0.0.0	0.0.0	2.0.0	5.0.0	0.0
ds7.csv	out2_4sec_shift	3.0	2.0	3.0	4.0	0.0	0.0.0	0.0.0	2.0.0	3.0.0	0.0
ds7.csv	out2_4sec_shift_noise	4.0	2.0	3.0	3.0	0.0	0.0.0	0.0.0	2.0.0	3.0.0	0.0
ds7.csv	out2_4sec_shift_filt	3.0	2.0	3.0	3.0	0.0	0.0.0	0.0.0	2.0.0	4.0.0	0.0
ds7.csv	out2_4sec_shift_noise_filt	0.0	2.0	3.0	2.0	0.0	0.0.0	0.0.0	2.0.0	0.0.0	0.0
ds7.csv	out2_6sec_shift	3.0	2.0	3.0	3.0	0.0	0.0.0	0.0.0	2.0.0	1.0.0	0.0
ds7.csv	out2_6sec_shift_noise	4.0	2.0	1.0	2.0	0.0	0.0.0	0.0.0	2.0.0	1.0.0	0.0
ds7.csv	out2_6sec_shift_filt	3.0	2.0	3.0	3.0	0.0	0.0.0	0.0.0	2.0.0	3.0.0	0.0
ds7.csv	out2_6sec_shift_noise_filt	3.0	2.0	3.0	3.0	0.0	0.0.0	0.0.0	2.0.0	3.0.0	0.0
ds7.csv	out2_8sec_shift	3.0	2.0	3.0	0.0	0.0	0.0.0	0.0.0	2.0.0	0.0.0	0.0
ds7.csv	out2_8sec_shift_noise	0.0	2.0	1.0	1.0	0.0	0.0.0	0.0.0	2.0.0	0.0.0	0.0
ds7.csv	out2_8sec_shift_filt	3.0	2.0	3.0	3.0	0.0	1.0.0	4.0.0	0.0.0	3.0.0	0.0
ds7.csv	out2_8sec_shift_noise_filt	3.0	2.0	3.0	3.0	0.0	3.0.0	2.0.0	3.0.0	1.0.0	0.0
ds7.csv	out2_10sec_shift	3.0	2.0	3.0	2.0	0.0	4.0.0	4.0.0	2.0.0	2.0.0	0.0
ds7.csv	out2_10sec_shift_noise	4.0	2.0	2.0	2.0	0.0	0.0.0	5.0.0	4.0.0	3.0.0	0.0
ds7.csv	out2_10sec_shift_filt	3.0	2.0	3.0	4.0	0.0	1.0.0	4.0.0	2.0.0	3.0.0	0.0
ds7.csv	out2_10sec_shift_noise_filt	0.0	2.0	3.0	4.0	0.0	4.0.0	5.0.0	1.0.0	2.0.0	0.0
ds7.csv	out3_2sec_shift	3.0	0.0	1.0	2.0	0.0	3.0.0	5.0.0	3.0.0	0.0.0	0.0
ds7.csv	out3_2sec_shift_noise	3.0	0.0	1.0	2.0	0.0	4.0.0	0.0.0	2.0.0	1.0.0	0.0
ds7.csv	out3_2sec_shift_filt	0.0	0.0	1.0	2.0	0.0	1.0.0	4.0.0	0.0.0	4.0.0	0.0
ds7.csv	out3_2sec_shift_noise_filt	0.0	0.0	1.0	2.0	0.0	1.0.0	4.0.0	0.0.0	4.0.0	0.0
ds7.csv	out3_4sec_shift	3.0	0.0	1.0	3.0	0.0	1.0.0	4.0.0	0.0.0	4.0.0	0.0
ds7.csv	out3_4sec_shift_noise	0.0	0.0	1.0	3.0	0.0	1.0.0	4.0.0	0.0.0	4.0.0	0.0
ds7.csv	out3_4sec_shift_filt	0.0	0.0	1.0	2.0	0.0	1.0.0	4.0.0	0.0.0	3.0.0	0.0
ds7.csv	out3_4sec_shift_noise_filt	0.0	0.0	1.0	2.0	0.0	1.0.0	4.0.0	0.0.0	3.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds7.csv	out3_6sec_shift	0.0	0.0	1.0	2.0	0.0	1.0.0	4.0.0	0.0.0	3.0.0	0.0
ds7.csv	out3_6sec_shift_noise	0.0	0.0	1.0	2.0	0.0	1.0.0	4.0.0	0.0.0	3.0.0	0.0
ds7.csv	out3_6sec_shift_filt	0.0	0.0	1.0	3.0	0.0	1.0.0	4.0.0	0.0.0	4.0.0	0.0
ds7.csv	out3_6sec_shift_noise_filt	0.0	0.0	1.0	3.0	0.0	1.0.0	4.0.0	0.0.0	4.0.0	0.0
ds7.csv	out3_8sec_shift	0.0	0.0	1.0	2.0	0.0	1.0.0	4.0.0	0.0.0	4.0.0	0.0
ds7.csv	out3_8sec_shift_noise	0.0	0.0	1.0	2.0	0.0	7.0.0	4.0.0	0.0.0	8.0.0	0.0
ds7.csv	out3_8sec_shift_filt	0.0	0.0	1.0	2.0	0.0	8.0.0	4.0.0	0.0.0	8.0.0	0.0
ds7.csv	out3_8sec_shift_noise_filt	0.0	0.0	1.0	2.0	0.0	1.0.0	4.0.0	8.0.0	4.0.0	0.0
ds7.csv	out3_10sec_shift	0.0	0.0	1.0	1.0	0.0	1.0.0	4.0.0	0.0.0	10.0.0	0.0
ds7.csv	out3_10sec_shift_noise	0.0	0.0	1.0	1.0	0.0	9.0.0	8.0.0	10.0.0	10.0.0	0.0
ds7.csv	out3_10sec_shift_filt	0.0	0.0	1.0	1.0	0.0	9.0.0	8.0.0	10.0.0	10.0.0	0.0
ds7.csv	out3_10sec_shift_noise_filt	0.0	0.0	1.0	1.0	0.0	10.0.0	4.0.0	10.0.0	10.0.0	0.0
ds7.csv	out4_2sec_shift	4.0	5.0	2.0	4.0	0.0	1.0.0	4.0.0	1.0.0	5.0.0	0.0
ds7.csv	out4_2sec_shift_noise	4.0	5.0	2.0	4.0	0.0	1.0.0	4.0.0	1.0.0	5.0.0	0.0
ds7.csv	out4_2sec_shift_filt	4.0	5.0	2.0	4.0	0.0	3.0.0	2.0.0	3.0.0	3.0.0	0.0
ds7.csv	out4_2sec_shift_noise_filt	4.0	5.0	2.0	4.0	0.0	3.0.0	2.0.0	2.0.0	0.0.0	0.0
ds7.csv	out4_4sec_shift	4.0	5.0	2.0	3.0	0.0	3.0.0	2.0.0	3.0.0	3.0.0	0.0
ds7.csv	out4_4sec_shift_noise	4.0	5.0	2.0	3.0	0.0	4.0.0	2.0.0	2.0.0	1.0.0	0.0
ds7.csv	out4_4sec_shift_filt	4.0	5.0	2.0	1.0	0.0	3.0.0	2.0.0	4.0.0	0.0.0	0.0
ds7.csv	out4_4sec_shift_noise_filt	4.0	5.0	2.0	1.0	0.0	3.0.0	2.0.0	5.0.0	2.0.0	0.0
ds7.csv	out4_6sec_shift	4.0	5.0	2.0	3.0	0.0	3.0.0	2.0.0	2.0.0	1.0.0	0.0
ds7.csv	out4_6sec_shift_noise	4.0	5.0	2.0	3.0	0.0	4.0.0	2.0.0	4.0.0	5.0.0	0.0
ds7.csv	out4_6sec_shift_filt	4.0	5.0	2.0	0.0	0.0	3.0.0	2.0.0	4.0.0	3.0.0	0.0
ds7.csv	out4_6sec_shift_noise_filt	4.0	5.0	2.0	0.0	0.0	0.0.0	2.0.0	2.0.0	1.0.0	0.0
ds7.csv	out4_8sec_shift	4.0	5.0	2.0	4.0	0.0	3.0.0	2.0.0	4.0.0	4.0.0	0.0
ds7.csv	out4_8sec_shift_noise	4.0	5.0	2.0	4.0	0.0	4.0.0	2.0.0	2.0.0	3.0.0	0.0
ds7.csv	out4_8sec_shift_filt	4.0	5.0	2.0	3.0	0.0	3.0.0	2.0.0	4.0.0	5.0.0	0.0
ds7.csv	out4_8sec_shift_noise_filt	4.0	5.0	2.0	3.0	0.0	0.0.0	2.0.0	10.0.0	8.0.0	0.0
ds7.csv	out4_10sec_shift	4.0	5.0	2.0	5.0	0.0	3.0.0	2.0.0	10.0.0	2.0.0	0.0
ds7.csv	out4_10sec_shift_noise	4.0	5.0	2.0	5.0	0.0	4.0.0	2.0.0	10.0.0	10.0.0	0.0
ds7.csv	out4_10sec_shift_filt	4.0	5.0	2.0	5.0	0.0	3.0.0	2.0.0	8.0.0	10.0.0	0.0
ds7.csv	out4_10sec_shift_noise_filt	4.0	5.0	2.0	5.0	0.0	4.0.0	10.0.0	9.0.0	10.0.0	0.0
ds7.csv	out5_2sec_shift	4.0	4.0	2.0	1.0	0.0	3.0.0	2.0.0	4.0.0	0.0.0	0.0

.0 Continued on next page

.o **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds7.csv	out5_2sec_shift_noise	4.0	4.0	2.0	1.0	0.0	3.0.0	2.0.0	4.0.0	0.0.0	0.0
ds7.csv	out5_2sec_shift_filt	4.0	4.0	2.0	1.0	0.0	4.0.0	4.0.0	5.0.0	3.0.0	0.0
ds7.csv	out5_2sec_shift_noise_filt	4.0	4.0	2.0	1.0	0.0	4.0.0	4.0.0	5.0.0	3.0.0	0.0
ds7.csv	out5_4sec_shift	4.0	4.0	2.0	3.0	0.0	4.0.0	4.0.0	2.0.0	2.0.0	0.0
ds7.csv	out5_4sec_shift_noise	4.0	4.0	2.0	3.0	0.0	4.0.0	4.0.0	2.0.0	2.0.0	0.0
ds7.csv	out5_4sec_shift_filt	4.0	4.0	2.0	4.0	0.0	0.0.0	4.0.0	2.0.0	1.0.0	0.0
ds7.csv	out5_4sec_shift_noise_filt	4.0	4.0	2.0	4.0	0.0	3.0.0	4.0.0	2.0.0	1.0.0	0.0
ds7.csv	out5_6sec_shift	4.0	4.0	2.0	3.0	0.0	0.0.0	4.0.0	5.0.0	6.0.0	0.0
ds7.csv	out5_6sec_shift_noise	4.0	4.0	2.0	3.0	0.0	0.0.0	4.0.0	2.0.0	6.0.0	0.0
ds7.csv	out5_6sec_shift_filt	4.0	4.0	2.0	1.0	0.0	4.0.0	4.0.0	5.0.0	6.0.0	0.0
ds7.csv	out5_6sec_shift_noise_filt	4.0	4.0	2.0	1.0	0.0	6.0.0	4.0.0	5.0.0	6.0.0	0.0
ds7.csv	out5_8sec_shift	4.0	4.0	2.0	1.0	0.0	4.0.0	4.0.0	2.0.0	2.0.0	0.0
ds7.csv	out5_8sec_shift_noise	4.0	4.0	2.0	1.0	0.0	4.0.0	4.0.0	2.0.0	2.0.0	0.0
ds7.csv	out5_8sec_shift_filt	4.0	4.0	2.0	3.0	0.0	4.0.0	4.0.0	2.0.0	4.0.0	0.0
ds7.csv	out5_8sec_shift_noise_filt	4.0	4.0	2.0	3.0	0.0	4.0.0	4.0.0	2.0.0	4.0.0	0.0
ds7.csv	out5_10sec_shift	0.0	4.0	2.0	1.0	0.0	4.0.0	4.0.0	10.0.0	3.0.0	0.0
ds7.csv	out5_10sec_shift_noise	0.0	4.0	2.0	1.0	0.0	4.0.0	4.0.0	2.0.0	3.0.0	0.0
ds7.csv	out5_10sec_shift_filt	0.0	4.0	2.0	1.0	0.0	0.0.0	4.0.0	5.0.0	10.0.0	0.0
ds7.csv	out5_10sec_shift_noise_filt	0.0	4.0	2.0	1.0	0.0	10.0.0	4.0.0	8.0.0	10.0.0	0.0
ds7.csv	out6_2sec_shift	5.0	5.0	1.0	0.0	0.0	4.0.0	4.0.0	8.0.0	2.0.0	0.0
ds7.csv	out6_2sec_shift_noise	5.0	5.0	1.0	0.0	0.0	4.0.0	4.0.0	5.0.0	2.0.0	0.0
ds7.csv	out6_2sec_shift_filt	5.0	5.0	1.0	2.0	0.0	0.0.0	5.0.0	4.0.0	3.0.0	0.0
ds7.csv	out6_2sec_shift_noise_filt	5.0	5.0	1.0	2.0	0.0	0.0.0	5.0.0	4.0.0	3.0.0	0.0
ds7.csv	out6_4sec_shift	5.0	5.0	4.0	5.0	0.0	0.0.0	5.0.0	4.0.0	5.0.0	0.0
ds7.csv	out6_4sec_shift_noise	5.0	5.0	4.0	4.0	0.0	0.0.0	5.0.0	4.0.0	5.0.0	0.0
ds7.csv	out6_4sec_shift_filt	5.0	5.0	4.0	0.0	0.0	0.0.0	5.0.0	4.0.0	0.0.0	0.0
ds7.csv	out6_4sec_shift_noise_filt	5.0	5.0	4.0	0.0	0.0	0.0.0	5.0.0	4.0.0	0.0.0	0.0
ds7.csv	out6_6sec_shift	5.0	5.0	4.0	3.0	0.0	0.0.0	5.0.0	4.0.0	1.0.0	0.0
ds7.csv	out6_6sec_shift_noise	5.0	5.0	4.0	4.0	0.0	0.0.0	5.0.0	4.0.0	1.0.0	0.0
ds7.csv	out6_6sec_shift_filt	5.0	5.0	4.0	0.0	0.0	0.0.0	5.0.0	4.0.0	0.0.0	0.0
ds7.csv	out6_6sec_shift_noise_filt	5.0	5.0	4.0	4.0	0.0	0.0.0	5.0.0	4.0.0	5.0.0	0.0
ds7.csv	out6_8sec_shift	5.0	5.0	4.0	5.0	0.0	0.0.0	5.0.0	4.0.0	3.0.0	0.0
ds7.csv	out6_8sec_shift_noise	5.0	5.0	4.0	0.0	0.0	0.0.0	5.0.0	4.0.0	3.0.0	0.0

.o Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds7.csv	out6_8sec_shift_filt	5.0	5.0	4.0	0.0	0.0	0.0.0	5.0.0	4.0.0	0.0.0	0.0
ds7.csv	out6_8sec_shift_noise_filt	5.0	5.0	4.0	5.0	0.0	0.0.0	5.0.0	4.0.0	0.0.0	0.0
ds7.csv	out6_10sec_shift	5.0	5.0	4.0	3.0	0.0	0.0.0	5.0.0	4.0.0	3.0.0	0.0
ds7.csv	out6_10sec_shift_noise	5.0	5.0	4.0	3.0	0.0	0.0.0	5.0.0	4.0.0	3.0.0	0.0
ds7.csv	out6_10sec_shift_filt	5.0	5.0	4.0	3.0	0.0	0.0.0	5.0.0	4.0.0	5.0.0	0.0
ds7.csv	out6_10sec_shift_noise_filt	5.0	5.0	4.0	3.0	0.0	0.0.0	5.0.0	4.0.0	5.0.0	0.0
ds7.csv	out7_2sec_shift	5.0	5.0	3.0	5.0	0.0	0.0.0	5.0.0	4.0.0	1.0.0	0.0
ds7.csv	out7_2sec_shift_noise	5.0	5.0	3.0	3.0	0.0	0.0.0	5.0.0	4.0.0	5.0.0	0.0
ds7.csv	out7_2sec_shift_filt	5.0	5.0	3.0	0.0	0.0	1.0.0	4.0.0	2.0.0	4.0.0	0.0
ds7.csv	out7_2sec_shift_noise_filt	5.0	5.0	3.0	0.0	0.0	1.0.0	4.0.0	2.0.0	4.0.0	0.0
ds7.csv	out7_4sec_shift	5.0	5.0	3.0	0.0	0.0	1.0.0	4.0.0	2.0.0	4.0.0	0.0
ds7.csv	out7_4sec_shift_noise	5.0	5.0	3.0	4.0	0.0	1.0.0	4.0.0	2.0.0	4.0.0	0.0
ds7.csv	out7_4sec_shift_filt	5.0	5.0	3.0	0.0	0.0	1.0.0	4.0.0	2.0.0	0.0.0	0.0
ds7.csv	out7_4sec_shift_noise_filt	5.0	5.0	3.0	0.0	0.0	1.0.0	4.0.0	2.0.0	0.0.0	0.0
ds7.csv	out7_6sec_shift	5.0	5.0	3.0	3.0	0.0	1.0.0	6.0.0	2.0.0	3.0.0	0.0
ds7.csv	out7_6sec_shift_noise	5.0	5.0	3.0	0.0	0.0	1.0.0	6.0.0	2.0.0	3.0.0	0.0
ds7.csv	out7_6sec_shift_filt	5.0	5.0	3.0	2.0	0.0	1.0.0	6.0.0	2.0.0	4.0.0	0.0
ds7.csv	out7_6sec_shift_noise_filt	5.0	5.0	3.0	2.0	0.0	6.0.0	4.0.0	2.0.0	4.0.0	0.0
ds7.csv	out7_8sec_shift	5.0	5.0	3.0	1.0	0.0	1.0.0	4.0.0	8.0.0	5.0.0	0.0
ds7.csv	out7_8sec_shift_noise	5.0	5.0	3.0	1.0	0.0	1.0.0	4.0.0	2.0.0	5.0.0	0.0
ds7.csv	out7_8sec_shift_filt	5.0	5.0	3.0	5.0	0.0	1.0.0	8.0.0	2.0.0	3.0.0	0.0
ds7.csv	out7_8sec_shift_noise_filt	5.0	5.0	3.0	5.0	0.0	1.0.0	4.0.0	2.0.0	3.0.0	0.0
ds7.csv	out7_10sec_shift	5.0	5.0	3.0	4.0	0.0	1.0.0	4.0.0	2.0.0	3.0.0	0.0
ds7.csv	out7_10sec_shift_noise	5.0	5.0	3.0	4.0	0.0	1.0.0	10.0.0	10.0.0	3.0.0	0.0
ds7.csv	out7_10sec_shift_filt	5.0	5.0	3.0	4.0	0.0	1.0.0	10.0.0	2.0.0	0.0.0	0.0
ds7.csv	out7_10sec_shift_noise_filt	5.0	5.0	3.0	0.0	0.0	1.0.0	4.0.0	2.0.0	0.0.0	0.0
ds7.csv	out8_2sec_shift	0.0	0.0	2.0	5.0	0.0	1.0.0	4.0.0	2.0.0	5.0.0	0.0
ds7.csv	out8_2sec_shift_noise	0.0	0.0	2.0	5.0	0.0	1.0.0	4.0.0	2.0.0	5.0.0	0.0
ds7.csv	out8_2sec_shift_filt	0.0	0.0	2.0	1.0	0.0	3.0.0	5.0.0	1.0.0	3.0.0	0.0
ds7.csv	out8_2sec_shift_noise_filt	0.0	0.0	2.0	1.0	0.0	3.0.0	5.0.0	1.0.0	3.0.0	0.0
ds7.csv	out8_4sec_shift	0.0	0.0	2.0	5.0	0.0	3.0.0	5.0.0	1.0.0	3.0.0	0.0
ds7.csv	out8_4sec_shift_noise	0.0	0.0	2.0	5.0	0.0	3.0.0	5.0.0	1.0.0	3.0.0	0.0
ds7.csv	out8_4sec_shift_filt	0.0	0.0	2.0	1.0	0.0	3.0.0	5.0.0	1.0.0	0.0.0	0.0

.0 Continued on next page

.o **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds7.csv	out8_4sec_shift_noise_filt	0.0	0.0	2.0	1.0	0.0	3.0.0	5.0.0	4.0.0	4.0.0	0.0
ds7.csv	out8_6sec_shift	0.0	0.0	2.0	5.0	0.0	3.0.0	5.0.0	1.0.0	5.0.0	0.0
ds7.csv	out8_6sec_shift_noise	0.0	0.0	2.0	5.0	0.0	3.0.0	5.0.0	1.0.0	3.0.0	0.0
ds7.csv	out8_6sec_shift_filt	0.0	0.0	2.0	3.0	0.0	3.0.0	5.0.0	1.0.0	1.0.0	0.0
ds7.csv	out8_6sec_shift_noise_filt	0.0	0.0	2.0	3.0	0.0	3.0.0	5.0.0	1.0.0	1.0.0	0.0
ds7.csv	out8_8sec_shift	0.0	0.0	2.0	4.0	0.0	3.0.0	5.0.0	8.0.0	3.0.0	0.0
ds7.csv	out8_8sec_shift_noise	0.0	0.0	2.0	0.0	0.0	8.0.0	7.0.0	8.0.0	3.0.0	0.0
ds7.csv	out8_8sec_shift_filt	0.0	0.0	2.0	1.0	0.0	8.0.0	7.0.0	6.0.0	4.0.0	0.0
ds7.csv	out8_8sec_shift_noise_filt	0.0	0.0	2.0	1.0	0.0	8.0.0	7.0.0	1.0.0	6.0.0	0.0
ds7.csv	out8_10sec_shift	0.0	0.0	2.0	3.0	0.0	3.0.0	5.0.0	10.0.0	10.0.0	0.0
ds7.csv	out8_10sec_shift_noise	0.0	0.0	2.0	3.0	0.0	10.0.0	10.0.0	10.0.0	9.5.0	0.0
ds7.csv	out8_10sec_shift_filt	0.0	0.0	2.0	0.0	0.0	3.0.0	5.0.0	10.0.0	8.0.0	0.0
ds7.csv	out8_10sec_shift_noise_filt	0.0	0.0	2.0	0.0	0.0	3.0.0	5.0.0	10.0.0	10.0.0	0.0
ds8.csv	out1_0sec	1.0	4.0	0.0	3.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds8.csv	out2_0sec	3.0	2.0	3.0	1.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds8.csv	out3_0sec	4.0	4.0	2.0	2.0	0.0	0.0.0	0.0.0	0.0.0	3.0.0	0.0
ds8.csv	out4_0sec	0.0	5.0	4.0	3.0	0.0	3.0.0	5.0.0	0.0.0	0.0.0	0.0
ds8.csv	out5_0sec	1.0	4.0	2.0	3.0	0.0	3.0.0	3.0.0	0.0.0	0.0.0	0.0
ds8.csv	out6_0sec	4.0	5.0	1.0	2.0	0.0	0.0.0	0.0.0	0.0.0	3.0.0	0.0
ds8.csv	out7_0sec	3.0	5.0	3.0	0.0	0.0	0.0.0	0.0.0	3.0.0	0.0.0	0.0
ds8.csv	out8_0sec	4.0	0.0	2.0	1.0	0.0	0.0.0	0.0.0	1.0.0	0.0.0	0.0
ds8.csv	out1_2sec_shift	1.0	4.0	0.0	4.0	0.0	3.0.0	5.0.0	3.0.0	0.0.0	0.0
ds8.csv	out1_2sec_shift_noise	1.0	4.0	0.0	4.0	0.0	3.0.0	5.0.0	3.0.0	5.0.0	0.0
ds8.csv	out1_2sec_shift_filt	1.0	4.0	0.0	4.0	0.0	3.0.0	5.0.0	3.0.0	1.0.0	0.0
ds8.csv	out1_2sec_shift_noise_filt	1.0	4.0	0.0	4.0	0.0	3.0.0	5.0.0	3.0.0	1.0.0	0.0
ds8.csv	out1_4sec_shift	1.0	4.0	0.0	3.0	0.0	4.0.0	4.0.0	3.0.0	4.0.0	0.0
ds8.csv	out1_4sec_shift_noise	1.0	4.0	0.0	3.0	0.0	3.0.0	5.0.0	3.0.0	3.0.0	0.0
ds8.csv	out1_4sec_shift_filt	1.0	4.0	0.0	3.0	0.0	3.0.0	5.0.0	3.0.0	4.0.0	0.0
ds8.csv	out1_4sec_shift_noise_filt	1.0	4.0	0.0	3.0	0.0	3.0.0	5.0.0	3.0.0	1.0.0	0.0
ds8.csv	out1_6sec_shift	1.0	4.0	0.0	4.0	0.0	3.0.0	6.0.0	3.0.0	2.0.0	0.0
ds8.csv	out1_6sec_shift_noise	1.0	4.0	0.0	4.0	0.0	3.0.0	6.0.0	6.0.0	6.0.0	0.0
ds8.csv	out1_6sec_shift_filt	1.0	4.0	0.0	4.0	0.0	3.0.0	6.0.0	6.0.0	6.0.0	0.0
ds8.csv	out1_6sec_shift_noise_filt	1.0	4.0	0.0	4.0	0.0	3.0.0	5.0.0	3.0.0	6.0.0	0.0

.o Continued on next page

.0 Table A.1 Presentation of Results Before and After Optimization

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds8.csv	out1_8sec_shift	1.0	4.0	0.0	4.0	0.0	3.0.0	5.0.0	3.0.0	5.0.0	0.0
ds8.csv	out1_8sec_shift_noise	1.0	4.0	0.0	4.0	0.0	3.0.0	5.0.0	3.0.0	5.0.0	0.0
ds8.csv	out1_8sec_shift_filt	1.0	4.0	0.0	2.0	0.0	4.0.0	8.0.0	2.0.0	10.0.0	0.0
ds8.csv	out1_8sec_shift_noise_filt	1.0	4.0	0.0	2.0	0.0	4.0.0	0.0.0	2.0.0	10.0.0	0.0
ds8.csv	out1_10sec_shift	1.0	4.0	1.0	2.0	0.0	4.0.0	0.0.0	2.0.0	10.0.0	0.0
ds8.csv	out1_10sec_shift_noise	1.0	4.0	1.0	2.0	0.0	10.0.0	10.0.0	2.0.0	10.0.0	0.0
ds8.csv	out1_10sec_shift_filt	1.0	4.0	1.0	5.0	0.0	4.0.0	10.0.0	2.0.0	3.0.0	0.0
ds8.csv	out1_10sec_shift_noise_filt	1.0	4.0	1.0	5.0	0.0	4.0.0	10.0.0	2.0.0	3.0.0	0.0
ds8.csv	out2_2sec_shift	3.0	2.0	3.0	3.0	0.0	4.0.0	0.0.0	2.0.0	2.0.0	0.0
ds8.csv	out2_2sec_shift_noise	3.0	2.0	2.0	0.0	0.0	4.0.0	0.0.0	2.0.0	4.0.0	0.0
ds8.csv	out2_2sec_shift_filt	3.0	2.0	3.0	3.0	0.0	4.0.0	0.0.0	2.0.0	4.0.0	0.0
ds8.csv	out2_2sec_shift_noise_filt	4.0	2.0	2.0	1.0	0.0	4.0.0	0.0.0	2.0.0	0.0.0	0.0
ds8.csv	out2_4sec_shift	3.0	2.0	4.0	0.0	0.0	4.0.0	0.0.0	2.0.0	5.0.0	0.0
ds8.csv	out2_4sec_shift_noise	3.0	2.0	5.0	2.0	0.0	4.0.0	0.0.0	2.0.0	5.0.0	0.0
ds8.csv	out2_4sec_shift_filt	3.0	2.0	2.0	1.0	0.0	4.0.0	0.0.0	2.0.0	3.0.0	0.0
ds8.csv	out2_4sec_shift_noise_filt	4.0	2.0	4.0	5.0	0.0	4.0.0	0.0.0	2.0.0	3.0.0	0.0
ds8.csv	out2_6sec_shift	3.0	2.0	4.0	3.0	0.0	4.0.0	0.0.0	2.0.0	4.0.0	0.0
ds8.csv	out2_6sec_shift_noise	0.0	2.0	2.0	1.0	0.0	4.0.0	0.0.0	2.0.0	4.0.0	0.0
ds8.csv	out2_6sec_shift_filt	3.0	2.0	4.0	4.0	0.0	4.0.0	0.0.0	2.0.0	5.0.0	0.0
ds8.csv	out2_6sec_shift_noise_filt	4.0	2.0	2.0	3.0	0.0	4.0.0	0.0.0	2.0.0	6.0.0	0.0
ds8.csv	out2_8sec_shift	3.0	2.0	4.0	5.0	0.0	4.0.0	8.0.0	2.0.0	5.0.0	0.0
ds8.csv	out2_8sec_shift_noise	0.0	2.0	4.0	0.0	0.0	8.0.0	8.0.0	2.0.0	8.0.0	0.0
ds8.csv	out2_8sec_shift_filt	3.0	2.0	4.0	2.0	0.0	8.0.0	1.0.0	3.0.0	8.0.0	0.0
ds8.csv	out2_8sec_shift_noise_filt	4.0	2.0	3.0	1.0	0.0	6.0.0	4.0.0	0.0.0	8.0.0	0.0
ds8.csv	out2_10sec_shift	3.0	2.0	4.0	4.0	0.0	4.0.0	6.0.0	9.0.0	1.0.0	0.0
ds8.csv	out2_10sec_shift_noise	4.0	2.0	4.0	4.0	0.0	6.0.0	9.0.0	10.0.0	10.0.0	0.0
ds8.csv	out2_10sec_shift_filt	3.0	2.0	4.0	0.0	0.0	4.0.0	6.0.0	10.0.0	9.0.0	0.0
ds8.csv	out2_10sec_shift_noise_filt	3.0	2.0	4.0	0.0	0.0	10.0.0	6.0.0	4.0.0	9.0.0	0.0
ds8.csv	out3_2sec_shift	4.0	4.0	5.0	3.0	0.0	3.0.0	1.0.0	3.0.0	2.0.0	0.0
ds8.csv	out3_2sec_shift_noise	4.0	4.0	5.0	3.0	0.0	3.0.0	1.0.0	3.0.0	2.0.0	0.0
ds8.csv	out3_2sec_shift_filt	4.0	4.0	2.0	2.0	0.0	3.0.0	1.0.0	3.0.0	0.0.0	0.0
ds8.csv	out3_2sec_shift_noise_filt	4.0	4.0	2.0	2.0	0.0	3.0.0	1.0.0	3.0.0	0.0.0	0.0
ds8.csv	out3_4sec_shift	0.0	4.0	2.0	1.0	0.0	3.0.0	1.0.0	3.0.0	2.0.0	0.0

.0 Continued on next page

.o Table A.1 Presentation of Results Before and After Optimization

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds8.csv	out3_4sec_shift_noise	0.0	4.0	2.0	1.0	0.0	3.0.0	1.0.0	3.0.0	2.0.0	0.0
ds8.csv	out3_4sec_shift_filt	0.0	4.0	2.0	1.0	0.0	3.0.0	1.0.0	3.0.0	3.0.0	0.0
ds8.csv	out3_4sec_shift_noise_filt	0.0	4.0	2.0	1.0	0.0	3.0.0	1.0.0	3.0.0	3.0.0	0.0
ds8.csv	out3_6sec_shift	4.0	4.0	5.0	4.0	0.0	3.0.0	1.0.0	3.0.0	3.0.0	0.0
ds8.csv	out3_6sec_shift_noise	4.0	4.0	5.0	4.0	0.0	3.0.0	1.0.0	3.0.0	3.0.0	0.0
ds8.csv	out3_6sec_shift_filt	4.0	4.0	2.0	2.0	0.0	3.0.0	1.0.0	3.0.0	1.0.0	0.0
ds8.csv	out3_6sec_shift_noise_filt	4.0	4.0	2.0	2.0	0.0	3.0.0	1.0.0	3.0.0	1.0.0	0.0
ds8.csv	out3_8sec_shift	4.0	4.0	2.0	4.0	0.0	3.0.0	1.0.0	3.0.0	1.0.0	0.0
ds8.csv	out3_8sec_shift_noise	4.0	4.0	2.0	4.0	0.0	3.0.0	1.0.0	3.0.0	1.0.0	0.0
ds8.csv	out3_8sec_shift_filt	4.0	4.0	2.0	3.0	0.0	3.0.0	1.0.0	3.0.0	6.0.0	0.0
ds8.csv	out3_8sec_shift_noise_filt	4.0	4.0	2.0	3.0	0.0	3.0.0	1.0.0	3.0.0	6.0.0	0.0
ds8.csv	out3_10sec_shift	0.0	4.0	5.0	1.0	0.0	3.0.0	1.0.0	3.0.0	4.0.0	0.0
ds8.csv	out3_10sec_shift_noise	0.0	4.0	5.0	1.0	0.0	3.0.0	1.0.0	3.0.0	4.0.0	0.0
ds8.csv	out3_10sec_shift_filt	4.0	4.0	5.0	2.0	0.0	3.0.0	1.0.0	3.0.0	2.0.0	0.0
ds8.csv	out3_10sec_shift_noise_filt	4.0	4.0	5.0	2.0	0.0	3.0.0	1.0.0	3.0.0	2.0.0	0.0
ds8.csv	out4_2sec_shift	0.0	5.0	4.0	3.0	0.0	6.0.0	4.0.0	0.0.0	2.0.0	0.0
ds8.csv	out4_2sec_shift_noise	0.0	5.0	4.0	3.0	0.0	6.0.0	4.0.0	0.0.0	5.0.0	0.0
ds8.csv	out4_2sec_shift_filt	0.0	5.0	4.0	5.0	0.0	6.0.0	4.0.0	0.0.0	5.0.0	0.0
ds8.csv	out4_2sec_shift_noise_filt	0.0	5.0	4.0	5.0	0.0	6.0.0	4.0.0	0.0.0	4.0.0	0.0
ds8.csv	out4_4sec_shift	0.0	5.0	4.0	0.0	0.0	6.0.0	4.0.0	0.0.0	3.0.0	0.0
ds8.csv	out4_4sec_shift_noise	0.0	5.0	4.0	0.0	0.0	6.0.0	4.0.0	0.0.0	1.0.0	0.0
ds8.csv	out4_4sec_shift_filt	0.0	5.0	4.0	1.0	0.0	6.0.0	4.0.0	0.0.0	5.0.0	0.0
ds8.csv	out4_4sec_shift_noise_filt	0.0	5.0	4.0	1.0	0.0	6.0.0	4.0.0	0.0.0	3.0.0	0.0
ds8.csv	out4_6sec_shift	0.0	5.0	4.0	0.0	0.0	6.0.0	4.0.0	0.0.0	4.0.0	0.0
ds8.csv	out4_6sec_shift_noise	0.0	5.0	4.0	5.0	0.0	6.0.0	4.0.0	6.0.0	3.0.0	0.0
ds8.csv	out4_6sec_shift_filt	0.0	5.0	4.0	3.0	0.0	6.0.0	4.0.0	6.0.0	3.0.0	0.0
ds8.csv	out4_6sec_shift_noise_filt	0.0	5.0	4.0	3.0	0.0	6.0.0	7.0.0	0.0.0	6.0.0	0.0
ds8.csv	out4_8sec_shift	0.0	5.0	4.0	0.0	0.0	6.0.0	4.0.0	0.0.0	7.0.0	0.0
ds8.csv	out4_8sec_shift_noise	0.0	5.0	4.0	0.0	0.0	6.0.0	4.0.0	0.0.0	7.0.0	0.0
ds8.csv	out4_8sec_shift_filt	0.0	5.0	4.0	3.0	0.0	6.0.0	4.0.0	0.0.0	3.0.0	0.0
ds8.csv	out4_8sec_shift_noise_filt	0.0	5.0	4.0	3.0	0.0	6.0.0	4.0.0	0.0.0	5.0.0	0.0
ds8.csv	out4_10sec_shift	0.0	5.0	4.0	5.0	0.0	6.0.0	4.0.0	0.0.0	5.0.0	0.0
ds8.csv	out4_10sec_shift_noise	0.0	5.0	4.0	5.0	0.0	6.0.0	6.0.0	0.0.0	1.0.0	0.0

.o Continued on next page

.o Table A.1 Presentation of Results Before and After Optimization

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds8.csv	out4_10sec_shift_filt	0.0	5.0	4.0	1.0	0.0	6.0.0	4.0.0	0.0.0	2.0.0	0.0
ds8.csv	out4_10sec_shift_noise_filt	0.0	5.0	4.0	5.0	0.0	6.0.0	4.0.0	0.0.0	4.0.0	0.0
ds8.csv	out5_2sec_shift	1.0	4.0	2.0	4.0	0.0	4.0.0	6.0.0	0.0.0	3.0.0	0.0
ds8.csv	out5_2sec_shift_noise	1.0	4.0	2.0	4.0	0.0	5.0.0	6.0.0	0.0.0	1.0.0	0.0
ds8.csv	out5_2sec_shift_filt	1.0	4.0	2.0	4.0	0.0	4.0.0	6.0.0	0.0.0	4.0.0	0.0
ds8.csv	out5_2sec_shift_noise_filt	1.0	4.0	2.0	4.0	0.0	4.0.0	6.0.0	0.0.0	5.0.0	0.0
ds8.csv	out5_4sec_shift	1.0	4.0	2.0	0.0	0.0	4.0.0	6.0.0	0.0.0	5.0.0	0.0
ds8.csv	out5_4sec_shift_noise	1.0	4.0	2.0	0.0	0.0	5.0.0	6.0.0	0.0.0	3.0.0	0.0
ds8.csv	out5_4sec_shift_filt	1.0	4.0	2.0	3.0	0.0	4.0.0	6.0.0	0.0.0	2.0.0	0.0
ds8.csv	out5_4sec_shift_noise_filt	1.0	4.0	2.0	3.0	0.0	4.0.0	6.0.0	0.0.0	3.0.0	0.0
ds8.csv	out5_6sec_shift	1.0	4.0	2.0	4.0	0.0	4.0.0	6.0.0	0.0.0	3.0.0	0.0
ds8.csv	out5_6sec_shift_noise	1.0	4.0	2.0	4.0	0.0	5.0.0	6.0.0	0.0.0	4.0.0	0.0
ds8.csv	out5_6sec_shift_filt	1.0	4.0	2.0	5.0	0.0	4.0.0	6.0.0	0.0.0	3.0.0	0.0
ds8.csv	out5_6sec_shift_noise_filt	1.0	4.0	2.0	5.0	0.0	4.0.0	6.0.0	0.0.0	4.0.0	0.0
ds8.csv	out5_8sec_shift	1.0	4.0	2.0	3.0	0.0	4.0.0	6.0.0	0.0.0	4.0.0	0.0
ds8.csv	out5_8sec_shift_noise	1.0	4.0	2.0	3.0	0.0	5.0.0	6.0.0	0.0.0	3.0.0	0.0
ds8.csv	out5_8sec_shift_filt	1.0	4.0	2.0	3.0	0.0	4.0.0	6.0.0	0.0.0	2.0.0	0.0
ds8.csv	out5_8sec_shift_noise_filt	1.0	4.0	2.0	3.0	0.0	5.0.0	6.0.0	0.0.0	4.0.0	0.0
ds8.csv	out5_10sec_shift	1.0	4.0	2.0	0.0	0.0	4.0.0	6.0.0	10.0.0	4.0.0	0.0
ds8.csv	out5_10sec_shift_noise	1.0	4.0	2.0	0.0	0.0	9.0.0	6.0.0	8.0.0	5.0.0	0.0
ds8.csv	out5_10sec_shift_filt	1.0	4.0	2.0	5.0	0.0	10.0.0	6.0.0	0.0.0	3.0.0	0.0
ds8.csv	out5_10sec_shift_noise_filt	1.0	4.0	2.0	5.0	0.0	5.0.0	6.0.0	0.0.0	6.0.0	0.0
ds8.csv	out6_2sec_shift	3.0	5.0	1.0	3.0	0.0	6.0.0	6.0.0	5.0.0	2.0.0	0.0
ds8.csv	out6_2sec_shift_noise	3.0	5.0	1.0	3.0	0.0	2.0.0	6.0.0	2.0.0	2.0.0	0.0
ds8.csv	out6_2sec_shift_filt	3.0	5.0	1.0	3.0	0.0	6.0.0	6.0.0	2.0.0	3.0.0	0.0
ds8.csv	out6_2sec_shift_noise_filt	3.0	5.0	1.0	3.0	0.0	2.0.0	2.0.0	5.0.0	3.0.0	0.0
ds8.csv	out6_4sec_shift	3.0	5.0	1.0	0.0	0.0	6.0.0	6.0.0	5.0.0	4.0.0	0.0
ds8.csv	out6_4sec_shift_noise	3.0	5.0	1.0	4.0	0.0	6.0.0	6.0.0	5.0.0	4.0.0	0.0
ds8.csv	out6_4sec_shift_filt	3.0	5.0	1.0	5.0	0.0	6.0.0	6.0.0	4.0.0	2.0.0	0.0
ds8.csv	out6_4sec_shift_noise_filt	3.0	5.0	1.0	3.0	0.0	6.0.0	6.0.0	4.0.0	2.0.0	0.0
ds8.csv	out6_6sec_shift	3.0	5.0	1.0	1.0	0.0	1.0.0	6.0.0	5.0.0	4.0.0	0.0
ds8.csv	out6_6sec_shift_noise	3.0	5.0	1.0	1.0	0.0	1.0.0	6.0.0	5.0.0	4.0.0	0.0
ds8.csv	out6_6sec_shift_filt	3.0	5.0	1.0	3.0	0.0	1.0.0	6.0.0	5.0.0	6.0.0	0.0

.o Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds8.csv	out6_6sec_shift_noise_filt	3.0	5.0	1.0	3.0	0.0	1.0.0	6.0.0	5.0.0	6.0.0	0.0
ds8.csv	out6_8sec_shift	3.0	5.0	1.0	4.0	0.0	1.0.0	6.0.0	5.0.0	4.0.0	0.0
ds8.csv	out6_8sec_shift_noise	3.0	5.0	1.0	4.0	0.0	1.0.0	6.0.0	5.0.0	4.0.0	0.0
ds8.csv	out6_8sec_shift_filt	3.0	5.0	1.0	0.0	0.0	1.0.0	6.0.0	5.0.0	2.0.0	0.0
ds8.csv	out6_8sec_shift_noise_filt	3.0	5.0	1.0	0.0	0.0	1.0.0	6.0.0	5.0.0	2.0.0	0.0
ds8.csv	out6_10sec_shift	3.0	5.0	1.0	3.0	0.0	1.0.0	6.0.0	3.0.0	4.0.0	0.0
ds8.csv	out6_10sec_shift_noise	3.0	5.0	1.0	3.0	0.0	1.0.0	6.0.0	3.0.0	4.0.0	0.0
ds8.csv	out6_10sec_shift_filt	3.0	5.0	1.0	5.0	0.0	1.0.0	6.0.0	3.0.0	4.0.0	0.0
ds8.csv	out6_10sec_shift_noise_filt	3.0	5.0	1.0	5.0	0.0	1.0.0	6.0.0	3.0.0	4.0.0	0.0
ds8.csv	out7_2sec_shift	3.0	5.0	3.0	3.0	0.0	0.0.0	6.0.0	3.0.0	1.0.0	0.0
ds8.csv	out7_2sec_shift_noise	3.0	5.0	3.0	3.0	0.0	2.0.0	6.0.0	3.0.0	1.0.0	0.0
ds8.csv	out7_2sec_shift_filt	3.0	5.0	3.0	3.0	0.0	2.0.0	6.0.0	3.0.0	1.0.0	0.0
ds8.csv	out7_2sec_shift_noise_filt	3.0	5.0	3.0	3.0	0.0	2.0.0	6.0.0	3.0.0	1.0.0	0.0
ds8.csv	out7_4sec_shift	3.0	5.0	3.0	3.0	0.0	0.0.0	6.0.0	3.0.0	6.0.0	0.0
ds8.csv	out7_4sec_shift_noise	3.0	5.0	3.0	3.0	0.0	0.0.0	6.0.0	3.0.0	6.0.0	0.0
ds8.csv	out7_4sec_shift_filt	3.0	5.0	3.0	0.0	0.0	0.0.0	4.0.0	3.0.0	2.0.0	0.0
ds8.csv	out7_4sec_shift_noise_filt	3.0	5.0	3.0	5.0	0.0	0.0.0	6.0.0	3.0.0	3.0.0	0.0
ds8.csv	out7_6sec_shift	3.0	5.0	3.0	1.0	0.0	0.0.0	6.0.0	3.0.0	8.0.0	0.0
ds8.csv	out7_6sec_shift_noise	3.0	5.0	3.0	1.0	0.0	0.0.0	6.0.0	3.0.0	8.0.0	0.0
ds8.csv	out7_6sec_shift_filt	3.0	5.0	3.0	2.0	0.0	0.0.0	6.0.0	3.0.0	8.0.0	0.0
ds8.csv	out7_6sec_shift_noise_filt	3.0	5.0	3.0	3.0	0.0	0.0.0	6.0.0	3.0.0	2.0.0	0.0
ds8.csv	out7_8sec_shift	3.0	5.0	3.0	4.0	0.0	0.0.0	6.0.0	3.0.0	4.0.0	0.0
ds8.csv	out7_8sec_shift_noise	3.0	5.0	3.0	1.0	0.0	8.0.0	6.0.0	3.0.0	4.0.0	0.0
ds8.csv	out7_8sec_shift_filt	3.0	5.0	3.0	2.0	0.0	8.0.0	6.0.0	3.0.0	6.0.0	0.0
ds8.csv	out7_8sec_shift_noise_filt	3.0	5.0	3.0	0.0	0.0	8.0.0	6.0.0	3.0.0	5.0.0	0.0
ds8.csv	out7_10sec_shift	3.0	5.0	3.0	4.0	0.0	8.0.0	6.0.0	3.0.0	10.0.0	0.0
ds8.csv	out7_10sec_shift_noise	3.0	5.0	3.0	4.0	0.0	0.0.0	8.0.0	3.0.0	10.0.0	0.0
ds8.csv	out7_10sec_shift_filt	3.0	5.0	3.0	5.0	0.0	0.0.0	9.0.0	3.0.0	10.0.0	0.0
ds8.csv	out7_10sec_shift_noise_filt	3.0	5.0	3.0	5.0	0.0	10.0.0	6.0.0	3.0.0	1.0.0	0.0
ds8.csv	out8_2sec_shift	4.0	0.0	2.0	1.0	0.0	1.0.0	6.0.0	4.0.0	1.0.0	0.0
ds8.csv	out8_2sec_shift_noise	4.0	0.0	2.0	1.0	0.0	1.0.0	6.0.0	4.0.0	1.0.0	0.0
ds8.csv	out8_2sec_shift_filt	4.0	0.0	2.0	5.0	0.0	1.0.0	6.0.0	4.0.0	1.0.0	0.0
ds8.csv	out8_2sec_shift_noise_filt	4.0	0.0	2.0	5.0	0.0	1.0.0	6.0.0	4.0.0	1.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds8.csv	out8_4sec_shift	4.0	0.0	2.0	3.0	0.0	4.0.0	6.0.0	4.0.0	1.0.0	0.0
ds8.csv	out8_4sec_shift_noise	4.0	0.0	2.0	3.0	0.0	4.0.0	6.0.0	4.0.0	1.0.0	0.0
ds8.csv	out8_4sec_shift_filt	4.0	0.0	2.0	3.0	0.0	4.0.0	6.0.0	4.0.0	1.0.0	0.0
ds8.csv	out8_4sec_shift_noise_filt	4.0	0.0	2.0	4.0	0.0	4.0.0	6.0.0	4.0.0	1.0.0	0.0
ds8.csv	out8_6sec_shift	4.0	0.0	2.0	4.0	0.0	4.0.0	6.0.0	4.0.0	1.0.0	0.0
ds8.csv	out8_6sec_shift_noise	4.0	0.0	2.0	0.0	0.0	4.0.0	6.0.0	4.0.0	1.0.0	0.0
ds8.csv	out8_6sec_shift_filt	4.0	0.0	2.0	5.0	0.0	4.0.0	6.0.0	4.0.0	4.0.0	0.0
ds8.csv	out8_6sec_shift_noise_filt	4.0	0.0	2.0	5.0	0.0	4.0.0	6.0.0	4.0.0	1.0.0	0.0
ds8.csv	out8_8sec_shift	4.0	0.0	2.0	3.0	0.0	4.0.0	6.0.0	4.0.0	8.0.0	0.0
ds8.csv	out8_8sec_shift_noise	4.0	0.0	2.0	3.0	0.0	4.0.0	6.0.0	8.0.0	2.0.0	0.0
ds8.csv	out8_8sec_shift_filt	4.0	0.0	2.0	4.0	0.0	4.0.0	6.0.0	8.0.0	6.0.0	0.0
ds8.csv	out8_8sec_shift_noise_filt	4.0	0.0	2.0	4.0	0.0	8.0.0	6.0.0	8.0.0	8.0.0	0.0
ds8.csv	out8_10sec_shift	4.0	0.0	2.0	5.0	0.0	8.0.0	6.0.0	4.0.0	1.0.0	0.0
ds8.csv	out8_10sec_shift_noise	4.0	0.0	2.0	5.0	0.0	4.0.0	6.0.0	4.0.0	10.0.0	0.0
ds8.csv	out8_10sec_shift_filt	4.0	0.0	2.0	5.0	0.0	4.0.0	6.0.0	10.0.0	8.0.0	0.0
ds8.csv	out8_10sec_shift_noise_filt	4.0	0.0	2.0	5.0	0.0	10.0.0	6.0.0	10.0.0	0.0.0	0.0
ds9.csv	out1_0sec	3.0	1.0	3.0	4.0	0.0	0.0.0	0.0.0	2.0.0	0.0.0	0.0
ds9.csv	out2_0sec	6.0	4.0	0.0	5.0	0.0	0.0.0	2.0.0	0.0.0	0.0.0	0.0
ds9.csv	out3_0sec	4.0	6.0	0.0	1.0	0.0	0.0.0	0.0.0	1.0.0	0.0.0	0.0
ds9.csv	out4_0sec	6.0	6.0	5.0	0.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds9.csv	out5_0sec	4.0	6.0	3.0	4.0	0.0	0.0.0	0.0.0	2.0.0	0.0.0	0.0
ds9.csv	out6_0sec	1.0	6.0	4.0	1.0	0.0	0.0.0	0.0.0	0.0.0	0.0.0	0.0
ds9.csv	out1_2sec_shift	3.0	1.0	3.0	2.0	0.0	2.0.0	2.0.0	2.0.0	0.0.0	0.0
ds9.csv	out1_2sec_shift_noise	3.0	1.0	3.0	2.0	0.0	0.0.0	0.0.0	2.0.0	3.0.0	0.0
ds9.csv	out1_2sec_shift_filt	3.0	1.0	3.0	0.0	0.0	2.0.0	0.0.0	2.0.0	4.0.0	0.0
ds9.csv	out1_2sec_shift_noise_filt	3.0	1.0	3.0	0.0	0.0	2.0.0	0.0.0	2.0.0	4.0.0	0.0
ds9.csv	out1_4sec_shift	3.0	1.0	3.0	2.0	0.0	4.0.0	0.0.0	4.0.0	1.0.0	0.0
ds9.csv	out1_4sec_shift_noise	3.0	1.0	3.0	2.0	0.0	4.0.0	0.0.0	4.0.0	1.0.0	0.0
ds9.csv	out1_4sec_shift_filt	3.0	1.0	3.0	3.0	0.0	4.0.0	0.0.0	4.0.0	3.0.0	0.0
ds9.csv	out1_4sec_shift_noise_filt	3.0	1.0	3.0	3.0	0.0	4.0.0	0.0.0	2.0.0	3.0.0	0.0
ds9.csv	out1_6sec_shift	3.0	1.0	3.0	3.0	0.0	0.0.0	0.0.0	2.0.0	6.0.0	0.0
ds9.csv	out1_6sec_shift_noise	3.0	1.0	3.0	3.0	0.0	0.0.0	0.0.0	2.0.0	6.0.0	0.0
ds9.csv	out1_6sec_shift_filt	3.0	1.0	3.0	1.0	0.0	0.0.0	0.0.0	2.0.0	6.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds9.csv	out1_6sec_shift_noise_filt	3.0	1.0	3.0	1.0	0.0	0.0.0	0.0.0	2.0.0	5.0.0	0.0
ds9.csv	out1_8sec_shift	3.0	1.0	3.0	1.0	0.0	0.0.0	0.0.0	2.0.0	1.0.0	0.0
ds9.csv	out1_8sec_shift_noise	3.0	1.0	3.0	1.0	0.0	0.0.0	0.0.0	2.0.0	8.0.0	0.0
ds9.csv	out1_8sec_shift_filt	3.0	1.0	3.0	6.0	0.0	0.0.0	0.0.0	2.0.0	8.0.0	0.0
ds9.csv	out1_8sec_shift_noise_filt	3.0	1.0	3.0	6.0	0.0	6.0.0	7.0.0	6.0.0	0.0.0	0.0
ds9.csv	out1_10sec_shift	3.0	1.0	3.0	4.0	0.0	0.0.0	0.0.0	2.0.0	1.0.0	0.0
ds9.csv	out1_10sec_shift_noise	3.0	1.0	3.0	4.0	0.0	8.0.0	10.0.0	9.0.0	1.0.0	0.0
ds9.csv	out1_10sec_shift_filt	3.0	1.0	3.0	2.0	0.0	8.0.0	10.0.0	10.0.0	5.0.0	0.0
ds9.csv	out1_10sec_shift_noise_filt	3.0	1.0	3.0	2.0	0.0	6.0.0	10.0.0	10.0.0	5.0.0	0.0
ds9.csv	out2_2sec_shift	6.0	4.0	0.0	2.0	0.0	0.0.0	0.0.0	2.0.0	3.0.0	0.0
ds9.csv	out2_2sec_shift_noise	6.0	4.0	0.0	5.0	0.0	0.0.0	0.0.0	2.0.0	3.0.0	0.0
ds9.csv	out2_2sec_shift_filt	6.0	4.0	0.0	5.0	0.0	3.0.0	2.0.0	1.0.0	4.0.0	0.0
ds9.csv	out2_2sec_shift_noise_filt	6.0	4.0	0.0	4.0	0.0	0.0.0	2.0.0	2.0.0	5.0.0	0.0
ds9.csv	out2_4sec_shift	6.0	4.0	0.0	3.0	0.0	3.0.0	2.0.0	1.0.0	2.0.0	0.0
ds9.csv	out2_4sec_shift_noise	6.0	4.0	0.0	1.0	0.0	0.0.0	2.0.0	4.0.0	3.0.0	0.0
ds9.csv	out2_4sec_shift_filt	6.0	4.0	0.0	5.0	0.0	3.0.0	2.0.0	4.0.0	3.0.0	0.0
ds9.csv	out2_4sec_shift_noise_filt	6.0	4.0	0.0	3.0	0.0	4.0.0	2.0.0	1.0.0	3.0.0	0.0
ds9.csv	out2_6sec_shift	6.0	4.0	0.0	4.0	0.0	3.0.0	2.0.0	1.0.0	4.0.0	0.0
ds9.csv	out2_6sec_shift_noise	6.0	4.0	0.0	3.0	0.0	6.0.0	2.0.0	6.0.0	2.0.0	0.0
ds9.csv	out2_6sec_shift_filt	6.0	4.0	0.0	3.0	0.0	3.0.0	2.0.0	6.0.0	0.0.0	0.0
ds9.csv	out2_6sec_shift_noise_filt	6.0	4.0	0.0	6.0	0.0	3.0.0	2.0.0	6.0.0	4.0.0	0.0
ds9.csv	out2_8sec_shift	6.0	4.0	0.0	3.0	0.0	3.0.0	2.0.0	1.0.0	8.0.0	0.0
ds9.csv	out2_8sec_shift_noise	6.0	4.0	0.0	1.0	0.0	3.0.0	8.0.0	1.0.0	8.0.0	0.0
ds9.csv	out2_8sec_shift_filt	6.0	4.0	0.0	3.0	0.0	3.0.0	2.0.0	1.0.0	4.0.0	0.0
ds9.csv	out2_8sec_shift_noise_filt	6.0	4.0	0.0	5.0	0.0	8.0.0	8.0.0	1.0.0	0.0.0	0.0
ds9.csv	out2_10sec_shift	6.0	4.0	0.0	5.0	0.0	3.0.0	2.0.0	10.0.0	9.0.0	0.0
ds9.csv	out2_10sec_shift_noise	6.0	6.0	0.0	1.0	0.0	3.0.0	2.0.0	10.0.0	10.0.0	0.0
ds9.csv	out2_10sec_shift_filt	6.0	4.0	0.0	2.0	0.0	3.0.0	10.0.0	9.3.0	3.0.0	0.0
ds9.csv	out2_10sec_shift_noise_filt	6.0	4.0	0.0	4.0	0.0	10.0.0	8.0.0	10.0.0	3.0.0	0.0
ds9.csv	out3_2sec_shift	4.0	6.0	0.0	3.0	0.0	3.0.0	2.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_2sec_shift_noise	5.0	6.0	0.0	1.0	0.0	3.0.0	2.0.0	2.0.0	3.0.0	0.0
ds9.csv	out3_2sec_shift_filt	4.0	6.0	0.0	4.0	0.0	4.0.0	1.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_2sec_shift_noise_filt	4.0	6.0	0.0	5.0	0.0	4.0.0	1.0.0	2.0.0	2.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds9.csv	out3_4sec_shift	4.0	6.0	0.0	5.0	0.0	4.0.0	1.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_4sec_shift_noise	5.0	6.0	0.0	3.0	0.0	4.0.0	1.0.0	4.0.0	2.0.0	0.0
ds9.csv	out3_4sec_shift_filt	4.0	6.0	0.0	2.0	0.0	4.0.0	1.0.0	4.0.0	2.0.0	0.0
ds9.csv	out3_4sec_shift_noise_filt	4.0	6.0	0.0	3.0	0.0	4.0.0	1.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_6sec_shift	4.0	6.0	0.0	3.0	0.0	4.0.0	1.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_6sec_shift_noise	5.0	6.0	0.0	4.0	0.0	5.0.0	1.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_6sec_shift_filt	4.0	6.0	0.0	3.0	0.0	6.0.0	6.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_6sec_shift_noise_filt	4.0	6.0	0.0	4.0	0.0	4.0.0	1.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_8sec_shift	4.0	6.0	0.0	4.0	0.0	4.0.0	1.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_8sec_shift_noise	5.0	6.0	0.0	3.0	0.0	8.0.0	1.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_8sec_shift_filt	4.0	6.0	0.0	2.0	0.0	8.0.0	8.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_8sec_shift_noise_filt	5.0	6.0	0.0	4.0	0.0	4.0.0	6.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_10sec_shift	4.0	6.0	0.0	4.0	0.0	4.0.0	1.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_10sec_shift_noise	5.0	6.0	0.0	5.0	0.0	4.0.0	10.0.0	2.0.0	2.0.0	0.0
ds9.csv	out3_10sec_shift_filt	4.0	6.0	0.0	3.0	0.0	4.0.0	1.0.0	10.0.0	2.0.0	0.0
ds9.csv	out3_10sec_shift_noise_filt	5.0	6.0	0.0	6.0	0.0	4.0.0	1.0.0	10.0.0	2.0.0	0.0
ds9.csv	out4_2sec_shift	6.0	6.0	5.0	2.0	0.0	4.0.0	1.0.0	2.0.0	3.0.0	0.0
ds9.csv	out4_2sec_shift_noise	6.0	6.0	5.0	2.0	0.0	4.0.0	1.0.0	2.0.0	2.0.0	0.0
ds9.csv	out4_2sec_shift_filt	6.0	6.0	5.0	3.0	0.0	3.0.0	5.0.0	2.0.0	4.0.0	0.0
ds9.csv	out4_2sec_shift_noise_filt	6.0	6.0	5.0	3.0	0.0	3.0.0	5.0.0	2.0.0	4.0.0	0.0
ds9.csv	out4_4sec_shift	6.0	6.0	5.0	4.0	0.0	3.0.0	5.0.0	2.0.0	5.0.0	0.0
ds9.csv	out4_4sec_shift_noise	6.0	6.0	5.0	4.0	0.0	3.0.0	5.0.0	2.0.0	5.0.0	0.0
ds9.csv	out4_4sec_shift_filt	6.0	6.0	5.0	2.0	0.0	3.0.0	5.0.0	2.0.0	0.0.0	0.0
ds9.csv	out4_4sec_shift_noise_filt	6.0	6.0	5.0	2.0	0.0	3.0.0	5.0.0	2.0.0	0.0.0	0.0
ds9.csv	out4_6sec_shift	1.0	6.0	5.0	4.0	0.0	3.0.0	5.0.0	2.0.0	3.0.0	0.0
ds9.csv	out4_6sec_shift_noise	1.0	6.0	5.0	4.0	0.0	3.0.0	5.0.0	2.0.0	3.0.0	0.0
ds9.csv	out4_6sec_shift_filt	1.0	6.0	5.0	6.0	0.0	3.0.0	5.0.0	2.0.0	5.0.0	0.0
ds9.csv	out4_6sec_shift_noise_filt	1.0	6.0	5.0	6.0	0.0	3.0.0	5.0.0	2.0.0	5.0.0	0.0
ds9.csv	out4_8sec_shift	1.0	6.0	5.0	4.0	0.0	3.0.0	5.0.0	2.0.0	5.0.0	0.0
ds9.csv	out4_8sec_shift_noise	1.0	6.0	5.0	4.0	0.0	3.0.0	5.0.0	2.0.0	5.0.0	0.0
ds9.csv	out4_8sec_shift_filt	1.0	6.0	5.0	2.0	0.0	3.0.0	5.0.0	2.0.0	5.0.0	0.0
ds9.csv	out4_8sec_shift_noise_filt	1.0	6.0	5.0	2.0	0.0	3.0.0	5.0.0	2.0.0	5.0.0	0.0
ds9.csv	out4_10sec_shift	1.0	6.0	3.0	4.0	0.0	3.0.0	5.0.0	8.0.0	10.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds9.csv	out4_10sec_shift_noise	1.0	6.0	3.0	4.0	0.0	3.0.0	10.0.0	2.0.0	9.6.0	0.0
ds9.csv	out4_10sec_shift_filt	1.0	6.0	3.0	4.0	0.0	3.0.0	5.0.0	9.0.0	0.0.0	0.0
ds9.csv	out4_10sec_shift_noise_filt	1.0	6.0	3.0	4.0	0.0	8.0.0	10.0.0	2.0.0	1.0.0	0.0
ds9.csv	out5_2sec_shift	0.0	6.0	3.0	1.0	0.0	3.0.0	5.0.0	2.0.0	1.0.0	0.0
ds9.csv	out5_2sec_shift_noise	0.0	6.0	3.0	1.0	0.0	3.0.0	5.0.0	2.0.0	1.0.0	0.0
ds9.csv	out5_2sec_shift_filt	0.0	6.0	3.0	1.0	0.0	0.0.0	0.0.0	2.0.0	3.0.0	0.0
ds9.csv	out5_2sec_shift_noise_filt	0.0	6.0	3.0	1.0	0.0	0.0.0	0.0.0	2.0.0	3.0.0	0.0
ds9.csv	out5_4sec_shift	0.0	6.0	3.0	6.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds9.csv	out5_4sec_shift_noise	0.0	6.0	3.0	6.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds9.csv	out5_4sec_shift_filt	0.0	6.0	3.0	2.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds9.csv	out5_4sec_shift_noise_filt	0.0	6.0	3.0	3.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds9.csv	out5_6sec_shift	0.0	6.0	3.0	4.0	0.0	0.0.0	6.0.0	2.0.0	6.0.0	0.0
ds9.csv	out5_6sec_shift_noise	0.0	6.0	3.0	4.0	0.0	0.0.0	4.0.0	6.0.0	2.0.0	0.0
ds9.csv	out5_6sec_shift_filt	0.0	6.0	3.0	2.0	0.0	6.0.0	4.0.0	6.0.0	2.0.0	0.0
ds9.csv	out5_6sec_shift_noise_filt	0.0	6.0	3.0	2.0	0.0	6.0.0	6.0.0	6.0.0	2.0.0	0.0
ds9.csv	out5_8sec_shift	0.0	6.0	3.0	4.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds9.csv	out5_8sec_shift_noise	0.0	6.0	3.0	4.0	0.0	0.0.0	0.0.0	7.0.0	2.0.0	0.0
ds9.csv	out5_8sec_shift_filt	0.0	6.0	3.0	6.0	0.0	0.0.0	0.0.0	7.0.0	0.0.0	0.0
ds9.csv	out5_8sec_shift_noise_filt	0.0	6.0	3.0	5.0	0.0	0.0.0	0.0.0	7.0.0	0.0.0	0.0
ds9.csv	out5_10sec_shift	0.0	6.0	3.0	2.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds9.csv	out5_10sec_shift_noise	0.0	6.0	3.0	3.0	0.0	10.0.0	10.0.0	2.0.0	2.0.0	0.0
ds9.csv	out5_10sec_shift_filt	0.0	6.0	3.0	1.0	0.0	9.6.0	10.0.0	10.0.0	9.0.0	0.0
ds9.csv	out5_10sec_shift_noise_filt	0.0	6.0	3.0	1.0	0.0	0.0.0	10.0.0	2.0.0	7.0.0	0.0
ds9.csv	out6_2sec_shift	1.0	6.0	4.0	1.0	0.0	0.0.0	0.0.0	2.0.0	3.0.0	0.0
ds9.csv	out6_2sec_shift_noise	1.0	6.0	4.0	1.0	0.0	0.0.0	2.0.0	2.0.0	3.0.0	0.0
ds9.csv	out6_2sec_shift_filt	1.0	6.0	4.0	1.0	0.0	5.0.0	2.0.0	2.0.0	3.0.0	0.0
ds9.csv	out6_2sec_shift_noise_filt	1.0	6.0	4.0	1.0	0.0	5.0.0	5.0.0	2.0.0	0.0.0	0.0
ds9.csv	out6_4sec_shift	4.0	6.0	4.0	1.0	0.0	5.0.0	5.0.0	2.0.0	3.0.0	0.0
ds9.csv	out6_4sec_shift_noise	4.0	6.0	4.0	1.0	0.0	5.0.0	5.0.0	2.0.0	4.0.0	0.0
ds9.csv	out6_4sec_shift_filt	4.0	6.0	4.0	1.0	0.0	5.0.0	5.0.0	2.0.0	4.0.0	0.0
ds9.csv	out6_4sec_shift_noise_filt	4.0	6.0	4.0	1.0	0.0	5.0.0	5.0.0	2.0.0	4.0.0	0.0
ds9.csv	out6_6sec_shift	4.0	6.0	4.0	1.0	0.0	5.0.0	5.0.0	2.0.0	4.0.0	0.0
ds9.csv	out6_6sec_shift_noise	4.0	6.0	4.0	1.0	0.0	5.0.0	6.0.0	2.0.0	0.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds9.csv	out6_6sec_shift_filt	4.0	6.0	4.0	4.0	0.0	5.0.0	6.0.0	2.0.0	5.0.0	0.0
ds9.csv	out6_6sec_shift_noise_filt	4.0	6.0	4.0	1.0	0.0	5.0.0	5.0.0	2.0.0	5.0.0	0.0
ds9.csv	out6_8sec_shift	4.0	6.0	4.0	2.0	0.0	5.0.0	5.0.0	2.0.0	3.0.0	0.0
ds9.csv	out6_8sec_shift_noise	4.0	6.0	4.0	2.0	0.0	5.0.0	8.0.0	2.0.0	6.0.0	0.0
ds9.csv	out6_8sec_shift_filt	4.0	6.0	4.0	1.0	0.0	5.0.0	8.0.0	2.0.0	7.0.0	0.0
ds9.csv	out6_8sec_shift_noise_filt	4.0	6.0	4.0	1.0	0.0	5.0.0	5.0.0	2.0.0	8.0.0	0.0
ds9.csv	out6_10sec_shift	4.0	6.0	4.0	1.0	0.0	10.0.0	5.0.0	2.0.0	10.0.0	0.0
ds9.csv	out6_10sec_shift_noise	4.0	6.0	4.0	1.0	0.0	10.0.0	5.0.0	2.0.0	9.0.0	0.0
ds9.csv	out6_10sec_shift_filt	4.0	6.0	4.0	0.0	0.0	10.0.0	5.0.0	3.0.0	9.0.0	0.0
ds9.csv	out6_10sec_shift_noise_filt	4.0	6.0	4.0	0.0	0.0	10.0.0	5.0.0	3.0.0	10.0.0	0.0
ds10.csv	out1_0sec	0.0	0.0	2.0	0.0	0.0	0.0	0.0	1.0.0	0.0	0.0
ds10.csv	out2_0sec	3.0	2.0	1.0	5.0	0.0	0.0	5.0.0	3.0.0	0.0	0.0
ds10.csv	out3_0sec	4.0	1.0	2.0	3.0	0.0	0.0	0.0	4.0.0	0.0	0.0
ds10.csv	out4_0sec	3.0	5.0	2.0	3.0	0.0	0.0	0.0	4.0.0	0.0	0.0
ds10.csv	out5_0sec	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ds10.csv	out6_0sec	5.0	5.0	5.0	3.0	0.0	3.0.0	0.0	0.0	0.0	0.0
ds10.csv	out7_0sec	5.0	5.0	4.0	0.0	0.0	2.0.0	0.0	0.0	2.0.0	0.0
ds10.csv	out8_0sec	0.0	0.0	2.0	3.0	0.0	0.0	0.0	0.0	2.0.0	0.0
ds10.csv	out1_2sec_shift	0.0	0.0	2.0	4.0	0.0	5.0.0	2.0.0	2.0.0	2.0.0	0.0
ds10.csv	out1_2sec_shift_noise	0.0	0.0	2.0	4.0	0.0	5.0.0	5.0.0	4.0.0	4.0.0	0.0
ds10.csv	out1_2sec_shift_filt	0.0	0.0	2.0	1.0	0.0	5.0.0	5.0.0	4.0.0	5.0.0	0.0
ds10.csv	out1_2sec_shift_noise_filt	0.0	0.0	2.0	1.0	0.0	5.0.0	5.0.0	4.0.0	2.0.0	0.0
ds10.csv	out1_4sec_shift	0.0	0.0	2.0	3.0	0.0	5.0.0	5.0.0	4.0.0	1.0.0	0.0
ds10.csv	out1_4sec_shift_noise	0.0	0.0	2.0	3.0	0.0	5.0.0	5.0.0	4.0.0	0.0.0	0.0
ds10.csv	out1_4sec_shift_filt	0.0	0.0	2.0	1.0	0.0	5.0.0	5.0.0	4.0.0	0.0.0	0.0
ds10.csv	out1_4sec_shift_noise_filt	0.0	0.0	2.0	1.0	0.0	5.0.0	5.0.0	4.0.0	3.0.0	0.0
ds10.csv	out1_6sec_shift	0.0	0.0	2.0	5.0	0.0	5.0.0	5.0.0	4.0.0	4.0.0	0.0
ds10.csv	out1_6sec_shift_noise	0.0	0.0	2.0	5.0	0.0	5.0.0	5.0.0	4.0.0	4.0.0	0.0
ds10.csv	out1_6sec_shift_filt	0.0	0.0	2.0	1.0	0.0	5.0.0	5.0.0	5.0.0	5.0.0	0.0
ds10.csv	out1_6sec_shift_noise_filt	0.0	0.0	2.0	1.0	0.0	5.0.0	5.0.0	5.0.0	1.0.0	0.0
ds10.csv	out1_8sec_shift	0.0	0.0	2.0	0.0	0.0	5.0.0	5.0.0	5.0.0	1.0.0	0.0
ds10.csv	out1_8sec_shift_noise	0.0	0.0	2.0	0.0	0.0	5.0.0	8.0.0	5.0.0	1.0.0	0.0
ds10.csv	out1_8sec_shift_filt	0.0	0.0	2.0	1.0	0.0	5.0.0	8.0.0	2.0.0	5.0.0	0.0

.0 Continued on next page

.o **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds10.csv	out1_8sec_shift_noise_filt	0.0	0.0	2.0	1.0	0.0	6.0.0	0.0.0	2.0.0	3.0.0	0.0
ds10.csv	out1_10sec_shift	0.0	0.0	2.0	5.0	0.0	0.0.0	0.0.0	2.0.0	3.0.0	0.0
ds10.csv	out1_10sec_shift_noise	0.0	0.0	2.0	5.0	0.0	10.0.0	0.0.0	2.0.0	3.0.0	0.0
ds10.csv	out1_10sec_shift_filt	0.0	0.0	2.0	3.0	0.0	10.0.0	0.0.0	2.0.0	4.0.0	0.0
ds10.csv	out1_10sec_shift_noise_filt	0.0	0.0	2.0	3.0	0.0	9.0.0	8.0.0	2.0.0	4.0.0	0.0
ds10.csv	out2_2sec_shift	3.0	2.0	1.0	4.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds10.csv	out2_2sec_shift_noise	0.0	2.0	1.0	5.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds10.csv	out2_2sec_shift_filt	3.0	2.0	1.0	1.0	0.0	0.0.0	0.0.0	2.0.0	4.0.0	0.0
ds10.csv	out2_2sec_shift_noise_filt	0.0	2.0	1.0	3.0	0.0	0.0.0	0.0.0	2.0.0	4.0.0	0.0
ds10.csv	out2_4sec_shift	3.0	2.0	1.0	3.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds10.csv	out2_4sec_shift_noise	5.0	2.0	1.0	3.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds10.csv	out2_4sec_shift_filt	3.0	2.0	1.0	4.0	0.0	0.0.0	0.0.0	2.0.0	3.0.0	0.0
ds10.csv	out2_4sec_shift_noise_filt	0.0	2.0	1.0	2.0	0.0	0.0.0	0.0.0	2.0.0	4.0.0	0.0
ds10.csv	out2_6sec_shift	3.0	2.0	1.0	0.0	0.0	0.0.0	0.0.0	2.0.0	5.0.0	0.0
ds10.csv	out2_6sec_shift_noise	3.0	2.0	1.0	4.0	0.0	0.0.0	0.0.0	2.0.0	5.0.0	0.0
ds10.csv	out2_6sec_shift_filt	3.0	2.0	1.0	5.0	0.0	0.0.0	0.0.0	2.0.0	0.0.0	0.0
ds10.csv	out2_6sec_shift_noise_filt	3.0	2.0	1.0	1.0	0.0	0.0.0	0.0.0	2.0.0	4.0.0	0.0
ds10.csv	out2_8sec_shift	3.0	2.0	1.0	4.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds10.csv	out2_8sec_shift_noise	0.0	2.0	1.0	0.0	0.0	0.0.0	0.0.0	2.0.0	2.0.0	0.0
ds10.csv	out2_8sec_shift_filt	3.0	2.0	1.0	3.0	0.0	0.0.0	3.0.0	2.0.0	2.0.0	0.0
ds10.csv	out2_8sec_shift_noise_filt	3.0	2.0	1.0	0.0	0.0	0.0.0	3.0.0	0.0.0	3.0.0	0.0
ds10.csv	out2_10sec_shift	3.0	2.0	1.0	3.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out2_10sec_shift_noise	0.0	2.0	1.0	3.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out2_10sec_shift_filt	3.0	2.0	1.0	2.0	0.0	0.0.0	3.0.0	3.0.0	3.0.0	0.0
ds10.csv	out2_10sec_shift_noise_filt	3.0	2.0	1.0	3.0	0.0	0.0.0	3.0.0	3.0.0	3.0.0	0.0
ds10.csv	out3_2sec_shift	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out3_2sec_shift_noise	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out3_2sec_shift_filt	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out3_2sec_shift_noise_filt	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out3_4sec_shift	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out3_4sec_shift_noise	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out3_4sec_shift_filt	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out3_4sec_shift_noise_filt	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0

.o Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	**Output signal**	**CC1**	**PR1**	**LR1**	**AR1**	**LM1**	**CC2**	**PR2**	**LR2**	**AR2**	**LM2**
ds10.csv	out3_6sec_shift	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out3_6sec_shift_noise	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out3_6sec_shift_filt	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out3_6sec_shift_noise_filt	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out3_8sec_shift	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	8.0.0	8.0.0	0.0
ds10.csv	out3_8sec_shift_noise	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	7.5.0	6.0.0	0.0
ds10.csv	out3_8sec_shift_filt	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	8.0.0	0.0.0	0.0
ds10.csv	out3_8sec_shift_noise_filt	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	8.0.0	6.0.0	0.0
ds10.csv	out3_10sec_shift	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	0.0.0	0.0.0	0.0
ds10.csv	out3_10sec_shift_noise	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	0.0.0	10.0.0	0.0
ds10.csv	out3_10sec_shift_filt	4.0	1.0	2.0	3.0	0.0	0.0.0	3.0.0	11.0.0	10.0.0	0.0
ds10.csv	out3_10sec_shift_noise_filt	4.0	1.0	2.0	2.0	0.0	0.0.0	3.0.0	11.0.0	3.0.0	0.0
ds10.csv	out4_2sec_shift	3.0	5.0	2.0	4.0	0.0	0.0.0	3.0.0	0.0.0	1.0.0	0.0
ds10.csv	out4_2sec_shift_noise	3.0	5.0	2.0	4.0	0.0	0.0.0	3.0.0	0.0.0	1.0.0	0.0
ds10.csv	out4_2sec_shift_filt	3.0	5.0	2.0	5.0	0.0	0.0.0	3.0.0	3.0.0	0.0.0	0.0
ds10.csv	out4_2sec_shift_noise_filt	3.0	5.0	2.0	5.0	0.0	0.0.0	4.0.0	3.0.0	4.0.0	0.0
ds10.csv	out4_4sec_shift	3.0	5.0	2.0	0.0	0.0	0.0.0	3.0.0	0.0.0	1.0.0	0.0
ds10.csv	out4_4sec_shift_noise	3.0	5.0	2.0	0.0	0.0	0.0.0	3.0.0	0.0.0	0.0.0	0.0
ds10.csv	out4_4sec_shift_filt	3.0	5.0	2.0	3.0	0.0	0.0.0	3.0.0	0.0.0	3.0.0	0.0
ds10.csv	out4_4sec_shift_noise_filt	3.0	5.0	2.0	3.0	0.0	0.0.0	3.0.0	0.0.0	3.0.0	0.0
ds10.csv	out4_6sec_shift	3.0	5.0	2.0	5.0	0.0	0.0.0	3.0.0	0.0.0	2.0.0	0.0
ds10.csv	out4_6sec_shift_noise	3.0	5.0	2.0	5.0	0.0	0.0.0	3.0.0	0.0.0	0.0.0	0.0
ds10.csv	out4_6sec_shift_filt	3.0	5.0	2.0	5.0	0.0	0.0.0	3.0.0	0.0.0	0.0.0	0.0
ds10.csv	out4_6sec_shift_noise_filt	3.0	5.0	2.0	5.0	0.0	0.0.0	3.0.0	0.0.0	0.0.0	0.0
ds10.csv	out4_8sec_shift	3.0	5.0	2.0	5.0	0.0	0.0.0	3.0.0	3.0.0	3.0.0	0.0
ds10.csv	out4_8sec_shift_noise	3.0	5.0	2.0	5.0	0.0	8.0.0	6.0.0	8.0.0	1.0.0	0.0
ds10.csv	out4_8sec_shift_filt	3.0	5.0	2.0	1.0	0.0	8.3.0	6.0.0	8.0.0	8.0.0	0.0
ds10.csv	out4_8sec_shift_noise_filt	3.0	5.0	2.0	1.0	0.0	0.0.0	8.0.0	6.0.0	8.0.0	0.0
ds10.csv	out4_10sec_shift	3.0	5.0	2.0	0.0	0.0	0.0.0	3.0.0	10.0.0	10.0.0	0.0
ds10.csv	out4_10sec_shift_noise	3.0	5.0	2.0	1.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out4_10sec_shift_filt	3.0	5.0	2.0	1.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out4_10sec_shift_noise_filt	3.0	5.0	2.0	1.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out5_2sec_shift	0.0	0.0	2.0	3.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds10.csv	out5_2sec_shift_noise	0.0	0.0	2.0	3.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out5_2sec_shift_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out5_2sec_shift_noise_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out5_4sec_shift	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out5_4sec_shift_noise	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out5_4sec_shift_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out5_4sec_shift_noise_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out5_6sec_shift	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out5_6sec_shift_noise	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out5_6sec_shift_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out5_6sec_shift_noise_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out5_8sec_shift	0.0	0.0	2.0	0.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out5_8sec_shift_noise	0.0	0.0	2.0	0.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out5_8sec_shift_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out5_8sec_shift_noise_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out5_10sec_shift	0.0	0.0	2.0	5.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out5_10sec_shift_noise	0.0	0.0	2.0	5.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out5_10sec_shift_filt	0.0	0.0	2.0	3.0	0.0	0.0.0	3.0.0	3.0.0	3.0.0	0.0
ds10.csv	out5_10sec_shift_noise_filt	0.0	0.0	2.0	3.0	0.0	0.0.0	3.0.0	3.0.0	3.0.0	0.0
ds10.csv	out6_2sec_shift	5.0	5.0	2.0	3.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out6_2sec_shift_noise	5.0	5.0	2.0	0.0	0.0	0.0.0	3.0.0	2.0.0	2.0.0	0.0
ds10.csv	out6_2sec_shift_filt	5.0	5.0	2.0	3.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out6_2sec_shift_noise_filt	5.0	5.0	2.0	1.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out6_4sec_shift	5.0	5.0	2.0	4.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out6_4sec_shift_noise	5.0	5.0	2.0	4.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out6_4sec_shift_filt	5.0	5.0	2.0	4.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out6_4sec_shift_noise_filt	5.0	5.0	2.0	0.0	0.0	0.0.0	3.0.0	2.0.0	2.0.0	0.0
ds10.csv	out6_6sec_shift	5.0	5.0	2.0	5.0	0.0	0.0.0	3.0.0	2.0.0	2.0.0	0.0
ds10.csv	out6_6sec_shift_noise	5.0	5.0	2.0	5.0	0.0	0.0.0	3.0.0	2.0.0	2.0.0	0.0
ds10.csv	out6_6sec_shift_filt	5.0	5.0	2.0	3.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out6_6sec_shift_noise_filt	5.0	5.0	2.0	5.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out6_8sec_shift	5.0	5.0	2.0	0.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out6_8sec_shift_noise	5.0	5.0	2.0	3.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds10.csv	out6_8sec_shift_filt	5.0	5.0	2.0	3.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out6_8sec_shift_noise_filt	5.0	5.0	2.0	3.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out6_10sec_shift	5.0	5.0	3.0	5.0	0.0	0.0.0	10.0.0	2.0.0	3.0.0	0.0
ds10.csv	out6_10sec_shift_noise	5.0	5.0	3.0	2.0	0.0	10.0.0	3.0.0	2.0.0	10.0.0	0.0
ds10.csv	out6_10sec_shift_filt	5.0	5.0	3.0	2.0	0.0	10.0.0	3.0.0	2.0.0	10.0.0	0.0
ds10.csv	out6_10sec_shift_noise_filt	5.0	5.0	3.0	2.0	0.0	1.0.0	10.0.0	2.0.0	3.0.0	0.0
ds10.csv	out7_2sec_shift	5.0	5.0	4.0	3.0	0.0	0.0.0	3.0.0	2.0.0	2.0.0	0.0
ds10.csv	out7_2sec_shift_noise	5.0	5.0	4.0	0.0	0.0	0.0.0	3.0.0	2.0.0	2.0.0	0.0
ds10.csv	out7_2sec_shift_filt	5.0	5.0	4.0	0.0	0.0	0.0.0	3.0.0	3.0.0	2.0.0	0.0
ds10.csv	out7_2sec_shift_noise_filt	5.0	5.0	4.0	0.0	0.0	0.0.0	3.0.0	3.0.0	3.0.0	0.0
ds10.csv	out7_4sec_shift	5.0	5.0	4.0	2.0	0.0	0.0.0	3.0.0	3.0.0	1.0.0	0.0
ds10.csv	out7_4sec_shift_noise	5.0	5.0	4.0	2.0	0.0	0.0.0	3.0.0	3.0.0	4.0.0	0.0
ds10.csv	out7_4sec_shift_filt	5.0	5.0	4.0	4.0	0.0	0.0.0	3.0.0	3.0.0	4.0.0	0.0
ds10.csv	out7_4sec_shift_noise_filt	5.0	5.0	4.0	4.0	0.0	0.0.0	3.0.0	3.0.0	1.0.0	0.0
ds10.csv	out7_6sec_shift	5.0	5.0	4.0	5.0	0.0	0.0.0	6.0.0	2.0.0	6.0.0	0.0
ds10.csv	out7_6sec_shift_noise	5.0	5.0	4.0	2.0	0.0	0.0.0	6.0.0	2.0.0	1.0.0	0.0
ds10.csv	out7_6sec_shift_filt	5.0	5.0	4.0	1.0	0.0	0.0.0	6.0.0	2.0.0	0.0.0	0.0
ds10.csv	out7_6sec_shift_noise_filt	5.0	5.0	4.0	0.0	0.0	6.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out7_8sec_shift	5.0	5.0	4.0	0.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out7_8sec_shift_noise	5.0	5.0	4.0	3.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out7_8sec_shift_filt	5.0	5.0	4.0	4.0	0.0	0.0.0	3.0.0	2.0.0	2.0.0	0.0
ds10.csv	out7_8sec_shift_noise_filt	5.0	5.0	4.0	4.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out7_10sec_shift	5.0	5.0	5.0	5.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out7_10sec_shift_noise	5.0	5.0	5.0	1.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out7_10sec_shift_filt	5.0	5.0	5.0	1.0	0.0	10.0.0	10.0.0	9.0.0	10.0.0	0.0
ds10.csv	out7_10sec_shift_noise_filt	5.0	5.0	5.0	1.0	0.0	10.0.0	10.0.0	9.5.0	9.0.0	0.0
ds10.csv	out8_2sec_shift	0.0	0.0	2.0	5.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out8_2sec_shift_noise	0.0	0.0	2.0	3.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out8_2sec_shift_filt	0.0	0.0	2.0	3.0	0.0	0.0.0	3.0.0	2.0.0	2.0.0	0.0
ds10.csv	out8_2sec_shift_noise_filt	0.0	0.0	2.0	3.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out8_4sec_shift	0.0	0.0	2.0	4.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out8_4sec_shift_noise	0.0	0.0	2.0	4.0	0.0	0.0.0	3.0.0	2.0.0	1.0.0	0.0
ds10.csv	out8_4sec_shift_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	4.0.0	2.0.0	4.0.0	0.0

.0 Continued on next page

.0 **Table A.1 Presentation of Results Before and After Optimization**

Filename	Output signal	CC1	PR1	LR1	AR1	LM1	CC2	PR2	LR2	AR2	LM2
ds10.csv	out8_4sec_shift_noise_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	4.0.0	2.0.0	4.0.0	0.0
ds10.csv	out8_6sec_shift	0.0	0.0	2.0	4.0	0.0	0.0.0	3.0.0	3.0.0	1.0.0	0.0
ds10.csv	out8_6sec_shift_noise	0.0	0.0	2.0	4.0	0.0	0.0.0	3.0.0	3.0.0	1.0.0	0.0
ds10.csv	out8_6sec_shift_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	5.0.0	6.0.0	6.0.0	0.0
ds10.csv	out8_6sec_shift_noise_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	2.0.0	0.0.0	0.0
ds10.csv	out8_8sec_shift	0.0	0.0	2.0	3.0	0.0	0.0.0	3.0.0	3.0.0	1.0.0	0.0
ds10.csv	out8_8sec_shift_noise	0.0	0.0	2.0	4.0	0.0	0.0.0	3.0.0	3.0.0	1.0.0	0.0
ds10.csv	out8_8sec_shift_filt	0.0	0.0	2.0	5.0	0.0	0.0.0	3.0.0	7.0.0	8.0.0	0.0
ds10.csv	out8_8sec_shift_noise_filt	0.0	0.0	2.0	5.0	0.0	0.0.0	3.0.0	2.0.0	8.0.0	0.0
ds10.csv	out8_10sec_shift	0.0	0.0	2.0	0.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out8_10sec_shift_noise	0.0	0.0	2.0	4.0	0.0	0.0.0	3.0.0	2.0.0	3.0.0	0.0
ds10.csv	out8_10sec_shift_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	3.0.0	10.0.0	2.0.0	0.0
ds10.csv	out8_10sec_shift_noise_filt	0.0	0.0	2.0	2.0	0.0	0.0.0	9.5.0	3.0.0	10.0.0	0.0