Comparison of Parametric Identification Techniques

Opperman, J.M

CSC 411
2019-06-16

Comparison of Parametric Identification Techniques

Opperman, J.M
15067077

Department of Chemical Engineering
University of Pretoria

CSC 411
2019-06-16

Comparison of Parametric Identification Techniques

Executive summary

In this study, an analysis of certain identification algorithms was performed. They were linear least squares regression, sequential quadratic programming (SQL) and differential evolution. These techniques were compared in terms of their robustness and speed when subjected to measurement noise and process input variations. The experimental section consisted of a python program that identified the coefficients of an Auto-regressive eXogenous model based on process input and output data. These models were then simulated and compared to the original responses. All three techniques were capable of dealing with low measurement noise and two input changes. It was found that the algorithms did not succeed to simulate the process response when multiple input changes were made, regardless of the measurement noise. Least squares and SQL showed similar accuracy in prediction, but SQL was faster in determining the coefficients. Differential evolution was the slowest algorithm, but superior to the other techniques in terms of precision when estimating the process output.

Contents

List of Figures

List of Tables

1 Introduction

System identification involves the determination of parameters in a mathematical model that aims to describe a dynamic system. This insight allows the prediction of how a physical process might respond to certain conditions or changes in its state. Parameters include physical properties or an estimation of the process dynamics.

Model-based control allows the controller to have insight of the process dynamics (Emami-Naeini, 2018). The model provides an estimated relationship between the physical properties of an industrial process and predictions in the behaviour of processes can be made (Ljung, 2008). The development of accurate empirical models that describe the process aid in process control (J.R Seborg, 2011). Therefore, the investigation on identification techniques to formulate process models may improve model-based control.

Before identification, it is necessary to specify what is known and unknown of a mathematical model. A white box model represents a system that can be described by differential equations derived from the physical relationship between inputs and outputs, otherwise known as transfer functions. In contrast, in a black box model, there is no information about the transfer function. Identification techniques are divided into two categories termed parametric and non-parametric identification. Parametric identification may utilize a white box model, whereas non-parametric techniques are applied to black box models.

In the past, identification was mostly dominated by prediction error methods and subspace identification (Gevers, 2006). Exploration has since expanded to incorporate optimization techniques. There is the need to reflect on some of these techniques and compare them in terms of robustness and speed. The following chapters will explain in more detail about the current state of the art. In order to evaluate identification techniques, a few optimization algorithms were explored in the context of parametric identification and with the aid of the Autoregressive with exogenous input (ARX) model. Elaboration on the terminology and methods are in the chapters to come.

2 Theory

2.1 Process, system and data types

In this section, some context and definitions are given to clarify terminology and concepts used later on.

A process encapsulates the physical equipment, operations and properties that involve converting raw feed material into products. Processes undergo changes over time and control systems were developed to predict and deal with these changes. In terms of identification, the system may be an empirical Equation such as a transfer function (J.R Seborg, 2011). Changes to a process can be deliberate such as changes in the production quantity. These changes are termed "inputs". Sudden and sustained changes may be applied to the inputs of an industrial process. Figure 1 shows some common types: The step input implements a sudden change in magnitude and is held indefinitely; the rectangular pulse input (RPI) subjects the system to a time limited step, after which the input returns to zero (J.R Seborg, 2011); a doublet input is initiated by first subjecting the system to a step input and letting the output reach steady state, then a positive rectangular pulse is implemented followed by an identical, but negative, rectangular pulse and returning the input to the original stepped value.

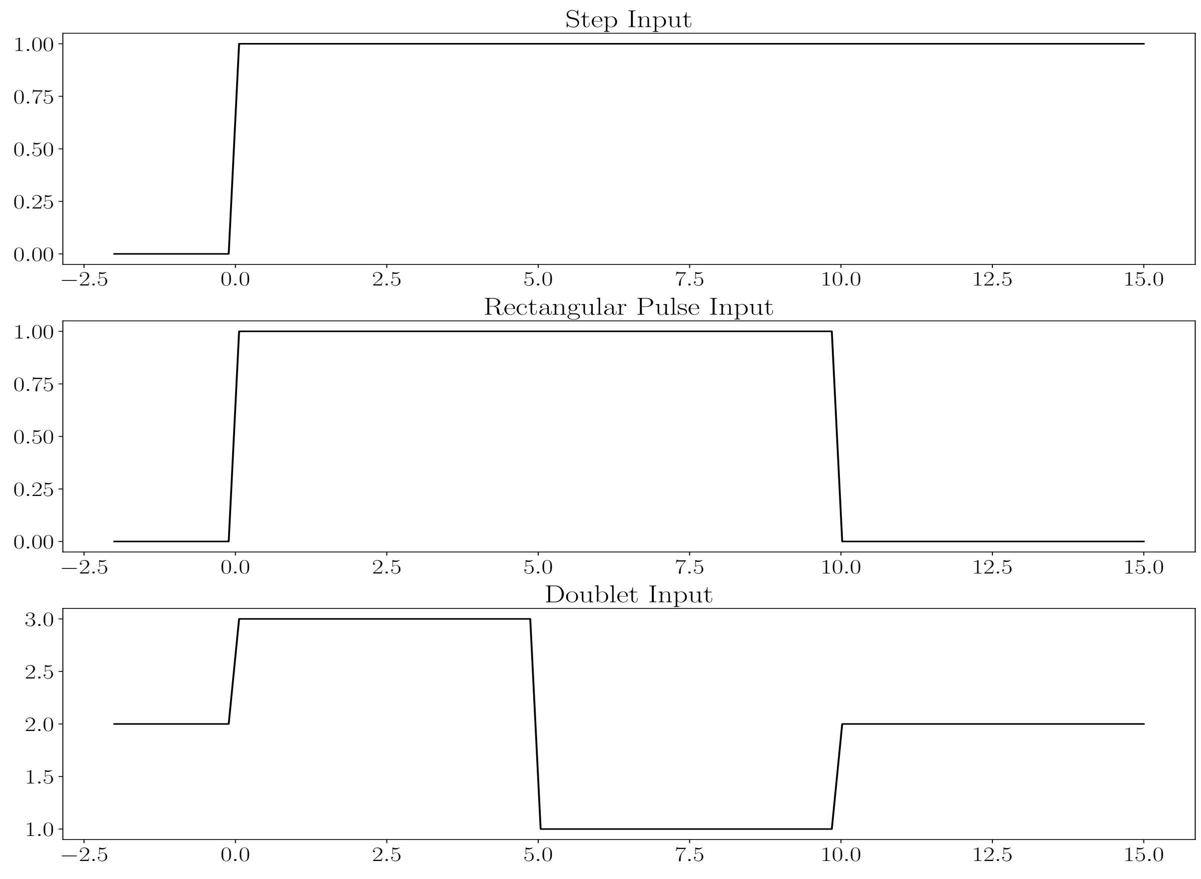

Figure 1: Inputs to the TCLab actuator.

In certain methods of system identification, models are fitted to process data - elaboration is contained in the following sections. This investigation utilized numerical data that were collected from a temperature sensor. Sensors are often installed in environments that produce significant noise and these vibrations may interfere with the quantity that is being measured (F Bordoni, 1990). Therefore the effect of measurement noise should be investigated.

2.2 Parametric identification

Estimating the parameters of a mathematical model is termed parametric identification.

2.2.1 Prediction error methods (PEM)

Prediction error methods aim to minimize the difference between the measured and estimated outputs (Guidi, 2008). There are cases where the system does not have a transfer function, but may be predicted by a standard predefined model. According to Ljung (1996), the most common model structures used in prediction error are Auto-regressive eXogenous (ARX), Auto-regressive Moving Average eXogenous (ARMAX), Box-Jenkins (BJ) and Output Error (OE) models. The model structures are chosen based on the relationship between the system's transfer functions. Figure 2 demonstrates these relationships and the results are discussed henceforth.

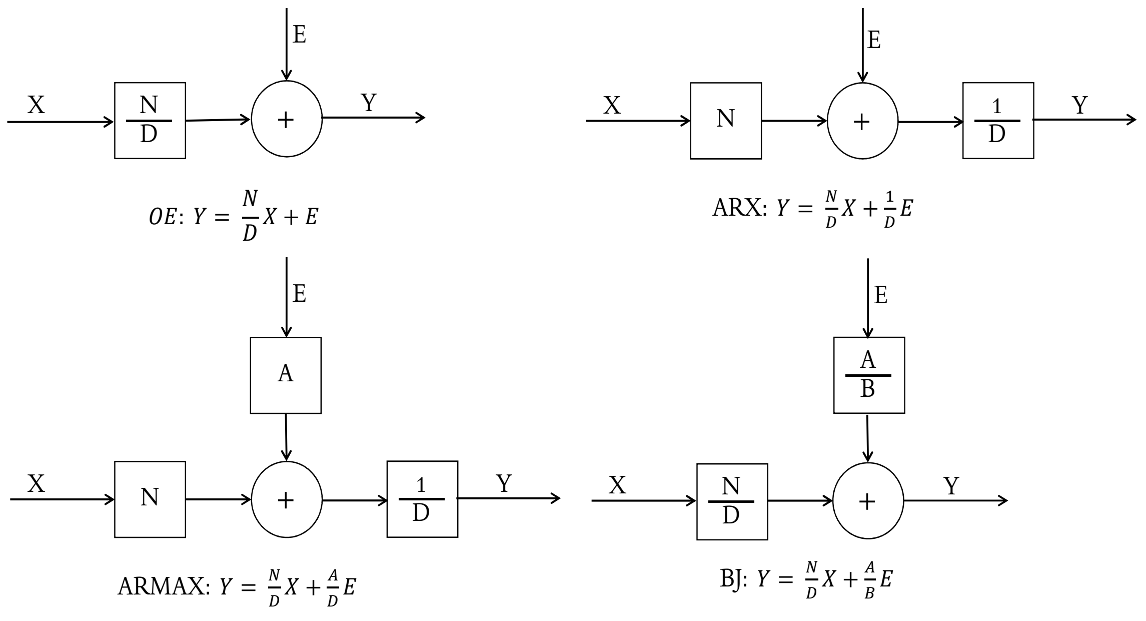
E
X
N/D
+
Y
OE: $Y = \frac{N}{D}X + E$
E
X
N
+
1/D
Y
ARX: $Y = \frac{N}{D}X + \frac{1}{D}E$
E
A
X
N
+
1/D
Y
ARMAX: $Y = \frac{N}{D}X + \frac{A}{D}E$
E
A/B
X
N/D
+
Y
BJ: $Y = \frac{N}{D}X + \frac{A}{B}E$

Figure 2: Model Structures of various systems (Chinarro, 2014).

The procedure entails fitting the ready-made models to input and output data. The term "exogenous" refers to the input having an external cause or origin and is therefore not affected by the output. Essentially, the next output is derived based on previous findings (Ljung, 1996). Therefore, upon defining the parameters of these models, one would be able to predict the system outputs when it is subjected to various inputs.

2.2.2 Linear regression

The least squares method is worth mentioning because it remains popular today. The basic principle will be explained at the hand of an example. Consider the well known second order ARX model below. In Equation 1, a_i and b_i are the coefficients that need to be identified and y_p is the predicted process output. Identification is done with the aid of the system's input (u) and output (y) data. The suitable coefficients for Equation 1 are obtained when the minimum value of the least squares Equation (Equation 2) is achieved. To clarify, the sum of the absolute difference between the real output and the predicted output is to be minimized.

$$y_p(t) = a_1y(t-1) + a_2y(t-2) + b_1u(t-1) + b_2u(t-2) \tag{1}$$

$$\frac{1}{N}\sum_{t=1}^{N}(y(t) - y_p(t))^2 \tag{2}$$

According to Lennart Ljung, this simple model and least squares method is versatile and can be used to perform identification on a variety of different system inputs (Ljung, 1996).

The advantage of least squares regression is that no knowledge of the system is required to identify the parameters. Studies have shown that the least squares algorithm, applied to identify the coefficients of the autoregressive model with exogenous input (ARMAX), produced satisfactory process identification (S Bedoui, 2015). The following figures were generated based on ARMAX systems identification studies with the least squares approach. In figure 3, the algorithm received data that contained very little measurement noise. Therefore, an investigation may be conducted to determine the robustness of this algorithm in the presence of noisy data.

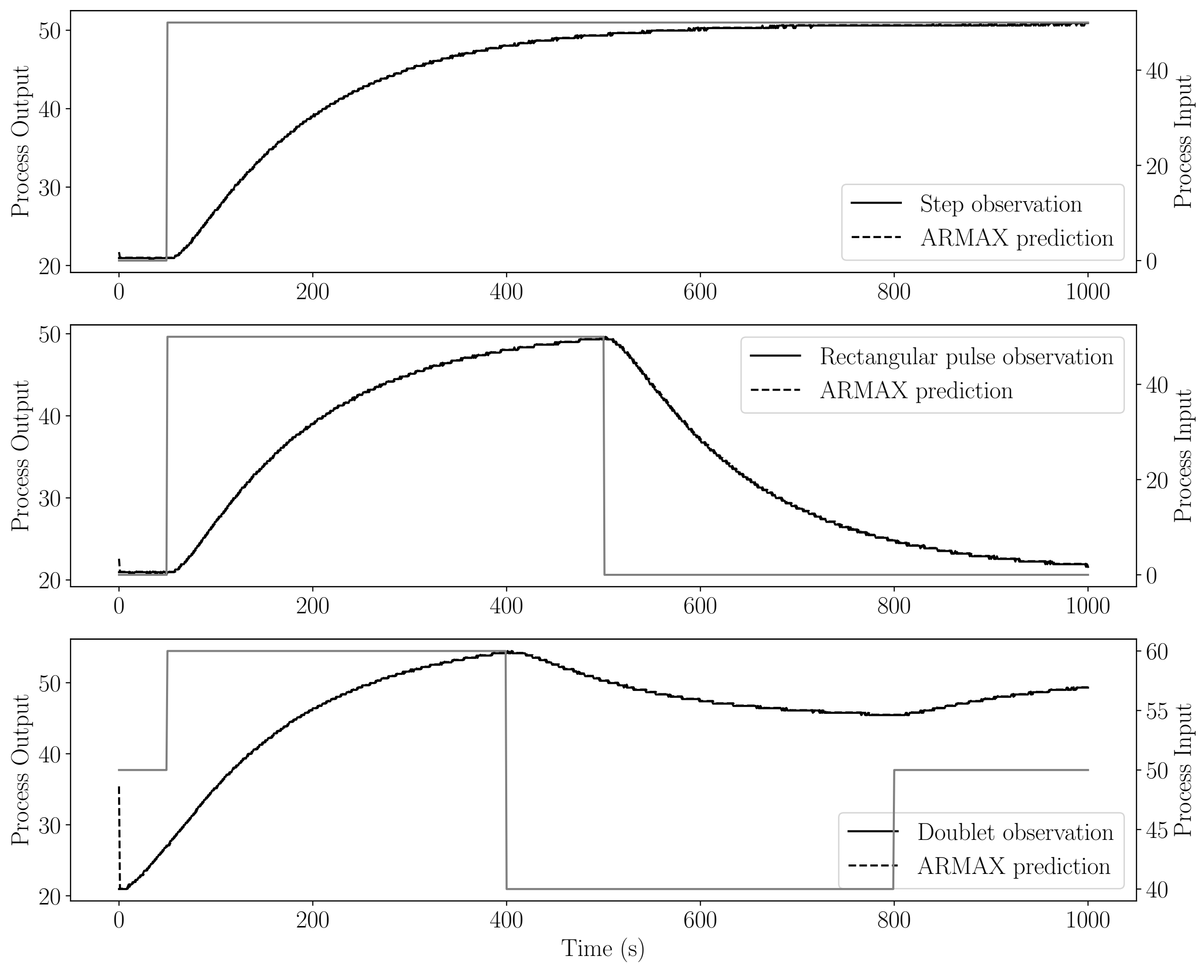

Figure 3: Responses of the TCL and ARMAX model with least squares prediction

2.2.3 Optimization algorithms

Optimization involves finding the most perfect solution for a given situation. Essentially, an objective function is to be minimized or maximized (Wright, 2016). More detail on the objective function is in section 3.2. In system identification a process model may be utilized as an objective function and the optimization will aim to identify parameters that succeed in describing the system. Consider the ARX model mentioned earlier. The algorithm computes a set of outputs by supplying possible parameters to the ARX Equation. The optimization will be complete when a minimum difference between the calculated and measured output has been obtained.

Through an iterative process, an optimization algorithm generates multiple possible solutions and weans out the non-compliant results. General optimization techniques may be classified as local or global methods. Local methods find the optimal point in one possible solution. Global optimization searches through all candidate solutions to find the overall perfect answer (M Saeid, 2014). It is however not always possible to find a global solution. The case often exists where a local solution is mistaken for a global solution. Depending on the starting point, the result from global optimization may differ if multiple local solutions exist (T.F Edgar, 2001).

2.2.4 Optimization algorithms tested in this project

Among the various local optimization methods, Minimization is available in the Python `Scipy` module and was chosen because it requires little computer memory (Venter, 2010). In the module `scipy.minimize`, the default optimization algorithm is called sequential quadratic programming (SQP). Newton's method utilizes the first and second derivatives (Hessian in a multivariate case) of a function to find the minima or maxima of it. SQP uses Newton's method and builds an approximation to a set of linear Equations. The first derivative locates a minimum or maximum in the function and the second derivative indicates whether the function is convex or concave.

Differential Evolution (DE) is a global method that optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple formulae. Then it keeps whichever candidate solution has the best score or fitness on the optimization problem at hand. In this way the optimization problem is treated as a black box that merely provides a measure of quality given a candidate solution and the gradient is therefore not needed (H Yousefi, 2008).

In past studies, differential evolution (DE) was applied to the identification of nonlinear systems. This algorithm was attractive because boundary limits could be applied to speed up the location of variables that describe the system. It was suggested that DE guarantees accurate solutions despite inaccurate initial conditions (H Yousefi, 2008).

2.3 Methods to determine the robustness of an identification algorithm

The robustness of an identification technique is its capacity to remain unaffected by small, deliberate variations in system inputs or disturbances (Y Vander Heyden, 2001). In this investigation, robustness was examined for three categories within identification. The section on Prediction Error methods (2.2.1) described that identification may be conducted by fitting a predetermined model to data. In this case the robustness of the model is described by its ability to accommodate variations in process behaviour. In other words, the model is judged on producing suitable predictions for a spectrum of systems and various inputs.

The second examination determines the robustness for a specific instance of a predetermined model. Continuing with Equation 1 as an example, an instance is created when the parameters of an ARX model have been determined. Furthermore for ARX, changing the order of the model produces a new instance. One measure of robustness for a model-instance is examined by applying it to the data at a different time interval.

Simply put, give this model instance a new set of inputs to predict the outputs. The test is to critique the model-instance accuracy in predicting the system outputs. Critiquing involves judging the goodness of fit of the prediction on the data and what discrepancies there were (J.A Royle, 2016).

The third category involves the identification algorithm. The parameters of an ARX model are easily determined through linear regression. A robust identification algorithm should recover the same parameters for an ARX model if a disturbance is experienced, or process noise is added. The robustness here provides an indication of the algorithm's reliability during normal usage. In a study that compared nonlinear system identification techniques, the error in coefficients was used to determine model performance (J Mengshi, 2019). Refer to Equation 3: y and y_1 are the current and previous system outputs respectively, u is the current input value and a and b are coefficients. In Equation 4, a_1 and b_1 are coefficients that perfectly describe the system. If the system experienced measurement noise and coefficients a_2 and b_2 were then obtained, the absolute difference in coefficients would present insight to how well the identification algorithm dealt with noise.

$$y = ay_1 + bu \tag{3}$$

$$error = \mid (a_1 - a_2) \mid + \mid (b_1 - b_2) \mid \tag{4}$$

To draw conclusions about the performance of an identification algorithm in this regard, it should be subjected to data with varying degrees of noise.

3 Experimental

3.1 Test system: Temperature Control Lab

It was decided to use the temperature control lab (TCL) in order to generate the data on which identification methods were tested. The TCL is a portable micro-controller unit that is accessed via a computer with the use of Python. The TCL consists of a heating element and temperature sensor. The data were generated by sending specified inputs to the heater and recording the associated changes in temperature. Figure 4 depicts the signal path between the computer and the TCL.

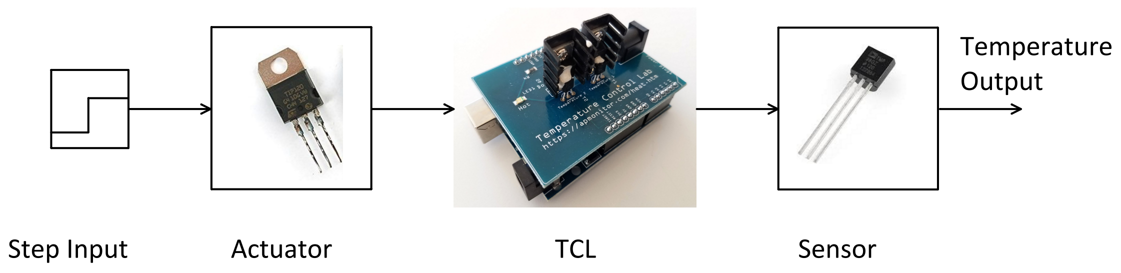

Figure 4: Schematic of the TCL and input to output loop.

3.2 Experimental method, computer programs and techniques

This section describes the experimental procedure and computer programs that were used.

3.2.1 Performing Identification

Python programs were written to build a tool that performs an identification-technique investigation. The program that identifies coefficients of an ARX equation was called `ARX_Optimization`. Various data files generated with the use of the TCL were stored in a directory. The analysis program utilized `ARX_Optimization` and it iterated through every file in the directory. In the subsequent steps, multiple identification methods pinpoint possible coefficients for the ARX equation in an attempt to optimally describe the system. The duration of each optimizer was recorded, along with the standard deviation in noise added to the test system. In order to identify trends, this analysis was repeated ten times. This quantity provided sufficient results and was also the maximum number of iterations that could be processed in a reasonable amount of time on the computer used.

The system data files were in the comma-separated values (csv) format and contained input and output data of the TCL. These inputs were step, rectangular-pulse and doublet. At the start of the analysis, `pandas.read_csv` (McKinney, 2018) was used to extract the TCL data. Ten degrees of Gaussian white noise with a range of zero to five standard

Table 1: Computer used to perform system identification

Atribute	Detail
Model	Dell inspiron 13-5378
Intel Core	i7-7500 CPU @ 2.70GHz
Installed memory	8Gb
System Type	64-bit operating system
Microsoft Windows Version	10

deviations were then added to the output data. For every newly generated noisy data set, `scipy.optimize` was used to implement three identification techniques to obtain possible coefficients of the ARX model, namely least squares, Sequential quadratic programming and Differential evolution (E Jones, 2001). Section 2.2.3 briefly mentioned that an objective function was required on which to perform optimization. The objective function (OF) in this project was generated as follows: The OF received guessed coefficients of the ARX equation (reprinted in Equation 5 for convenience) and calculated a list of output values y_p. As shown in Equation 6, the absolute difference between each real and calculated output was then summed to obtain a value that needed to be minimized.

$$y_p(t) = a_1 y(t-1) + a_2 y(t-2) + b_1 u(t-1) + b_2 u(t-2) \tag{5}$$

$$\sum_{k=1}^{t} \mid (ydev[k] - y_p[k]) \mid \tag{6}$$

Once the lowest possible value of Equation 6 was achieved, the corresponding coefficients were saved along with the time it took for the algorithm to reach this result. The input and noisy output data, coefficients and runtime were written into excel files using `XlsXwriter` (McNamara, n.d.).

In order to draw definitive conclusions about trends in the data, a substantial number of runs needed to be completed to collect enough samples. It was discovered that each identification procedure often took more than a minute and this made the collection of sufficient data tedious. To speed up the testing, parallel processing was utilized. The module `joblib.Parallel` enabled the Python program to be executed on multiple CPUs. In this way, it was possible to generate approximately 200 excel files in under an hour.

4 Results and Discussion

Note that the figures in this section all follow the following format: “Clean” refers to the original data (with no measurement noise added); the term “data” is the TCL output data; “approximation” indicates the simulated results obtained from identification.

4.1 Linear least squares regression

In figure 5.a, the line “Clean Approximation” closely follows the trend of line ”Clean Data”. This depicts the step response of the TCL without any added measurement noise and the simulation of the ARX equation for this data set. It appeared as though an increase in measurement noise slowed down the identification procedure since, in figure 5.b, the slowest time occurred at the largest degree of noise. Figure 5.d shows little variation in runtime and the slowest time corresponds to the largest coefficient error. The approximation became less accurate as measurement noise increased, as can be expected.

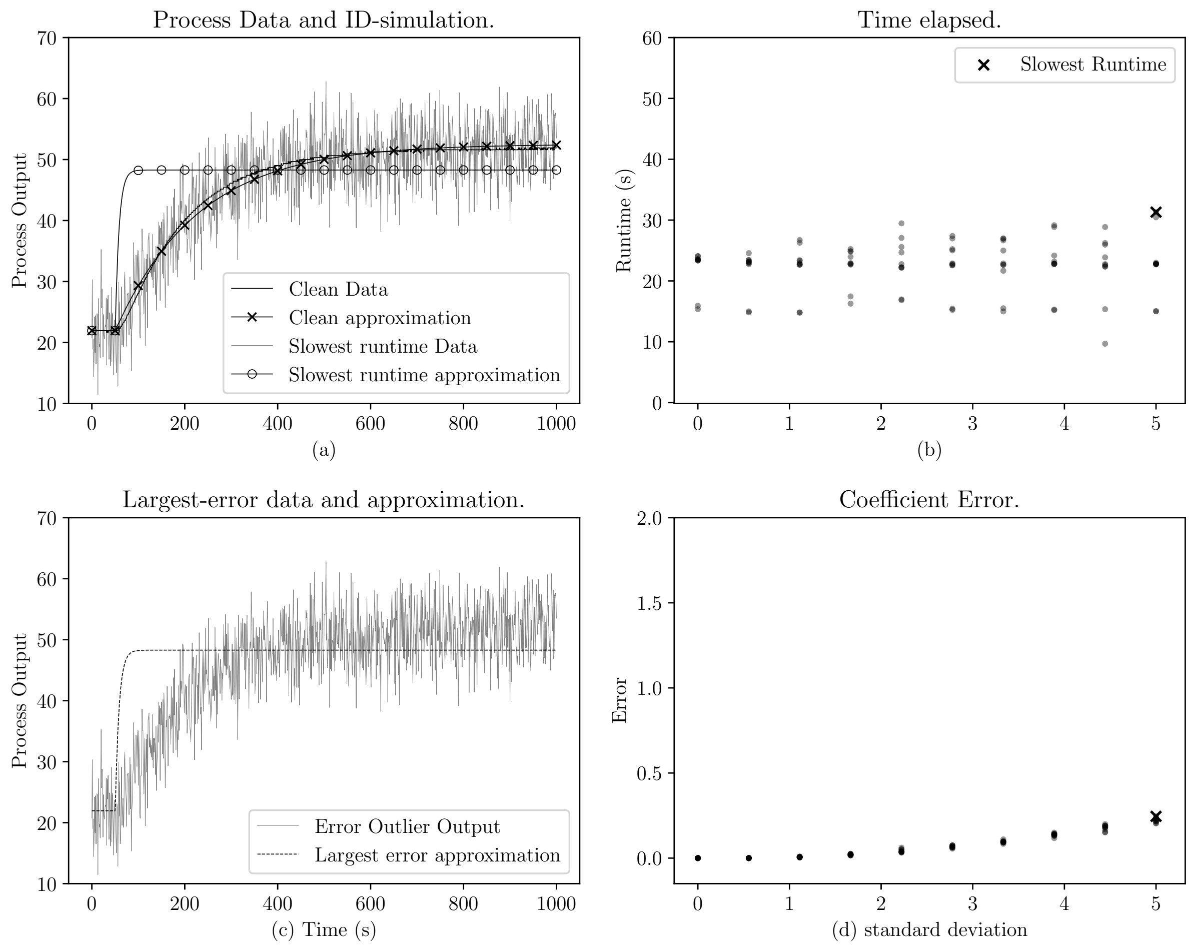

Figure 5: (a) TCL step response data and second order ARX model with least squares approximation. (b) Runtimes of each optimization versus the standard deviation in added noise. (c) Data and simulation of the data set that produced the largest coefficient error (marker “×” in d). (d) Coefficient error in the ARX model plotted against the standard deviation in added noise.

Figures 6.b and 7.b show that identifying a first order ARX model was approximately twice as fast as that of second order. Furthermore, contradictory to the findings in step response run-time, the slowest identification procedure occurred in the regions with low measurement noise.

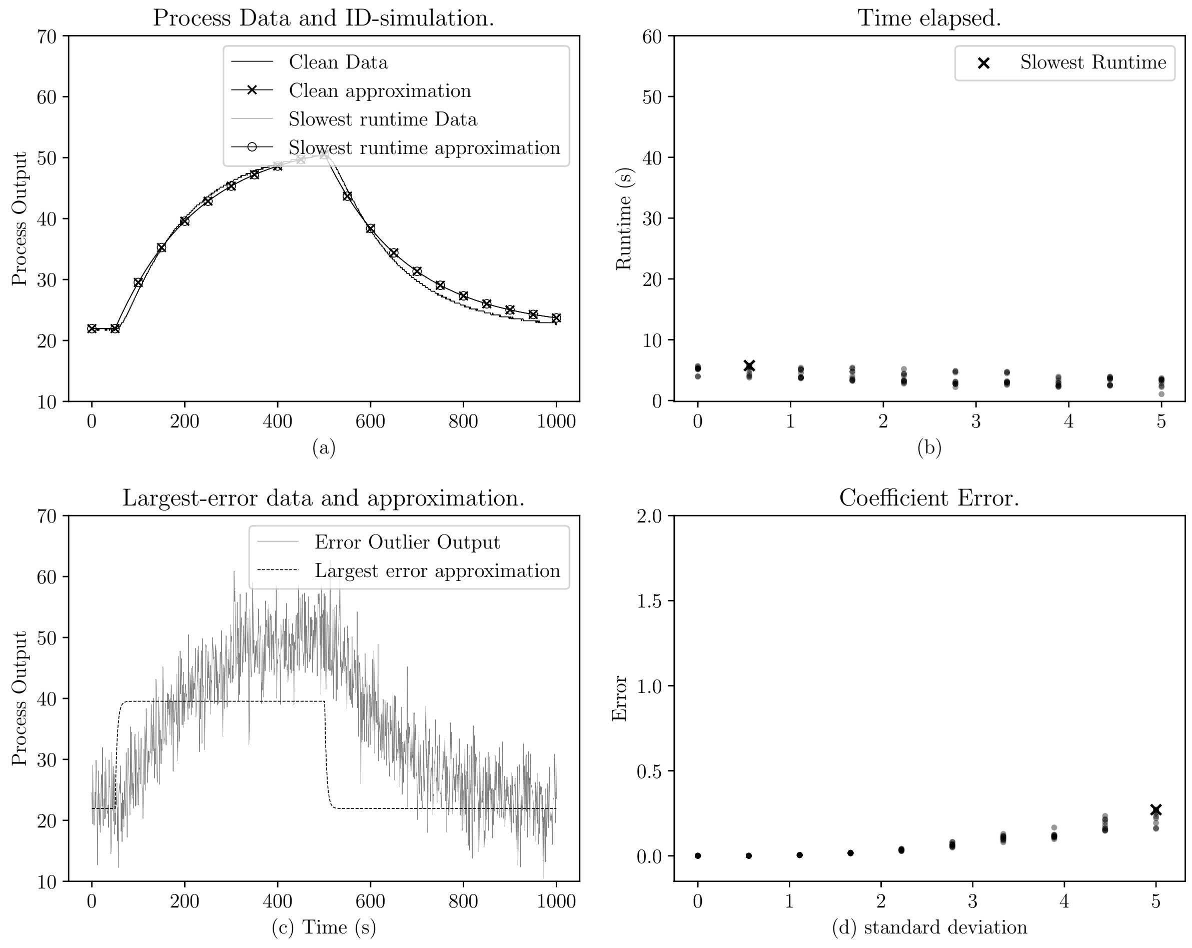

Figure 6: (a) TCL RPI response data and first order ARX model with least squares approximation. (b) Runtimes of each optimization versus the standard deviation. (c) Data and simulation of the data set that produced the largest coefficient error (marker "×" in d). (d) Coefficient error in the ARX model plotted against the standard deviation in added noise.

The least squares algorithm sufficiently identifies the clean rectangular pulse response. The algorithm also shows little variation in runtime as depicted in figure 7.b. Figure 7.c shows that the quality of the prediction deteriorates with the addition of measurement noise.

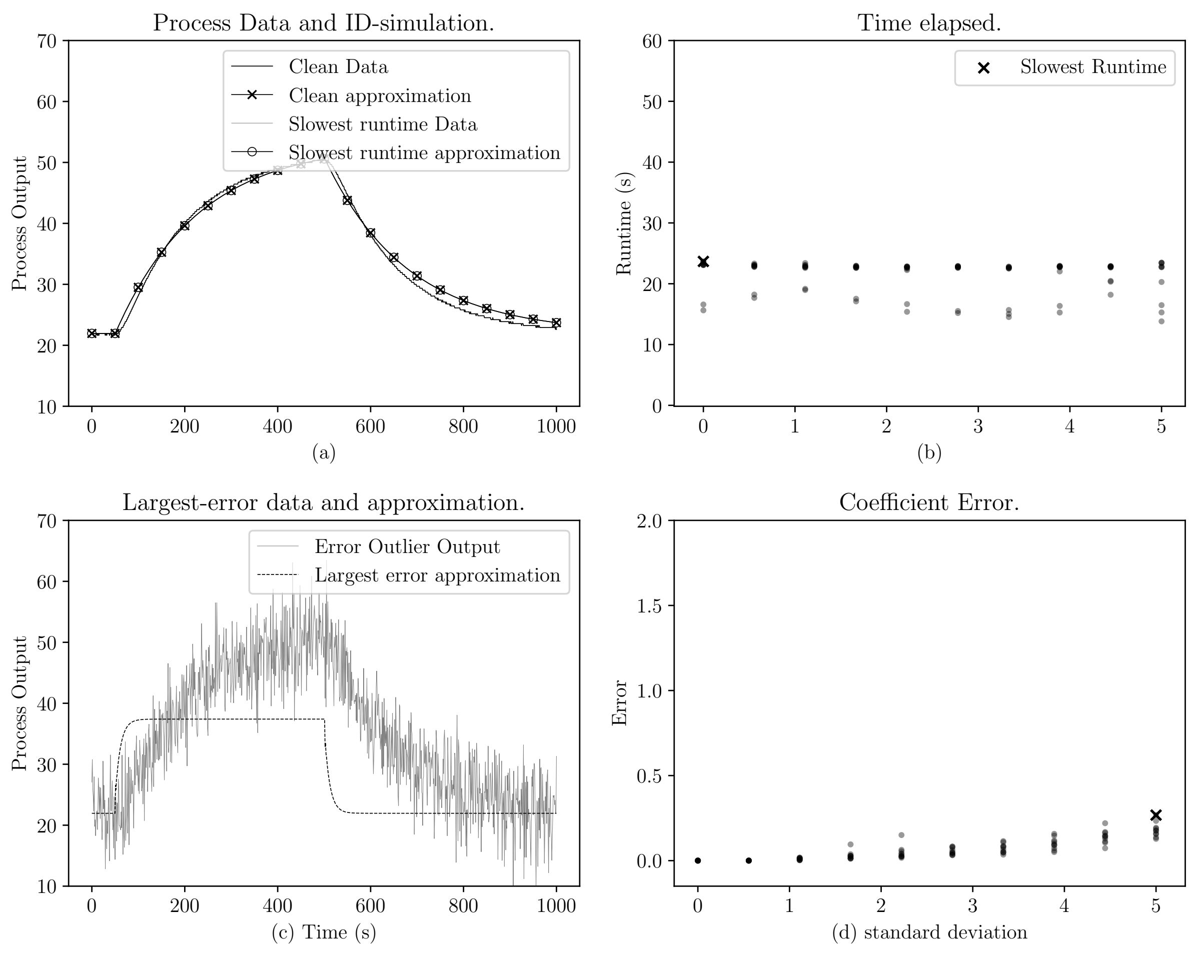

Figure 7: (a) TCL RPI response data and second order ARX model with least squares approximation. (b) Runtimes of each optimization versus the standard deviation. (c) Data and simulation of the data set that produced the largest coefficient error (marker "$\times$" in d). (d) Coefficient error in the ARX model plotted against the standard deviation in added noise.

The least squares fit to a doublet input is shown in figures 8 and 9. The former depicts first order ARX identification. Once again, for low measurement noise, a similar shape to the real response was identified. Least squares struggled to deal with more variation in inputs. The prediction undershoots and incorrectly predicts the final steady state. Since the second order ARX form considers a larger range of input-output data (two time-steps), one might expect better performance.

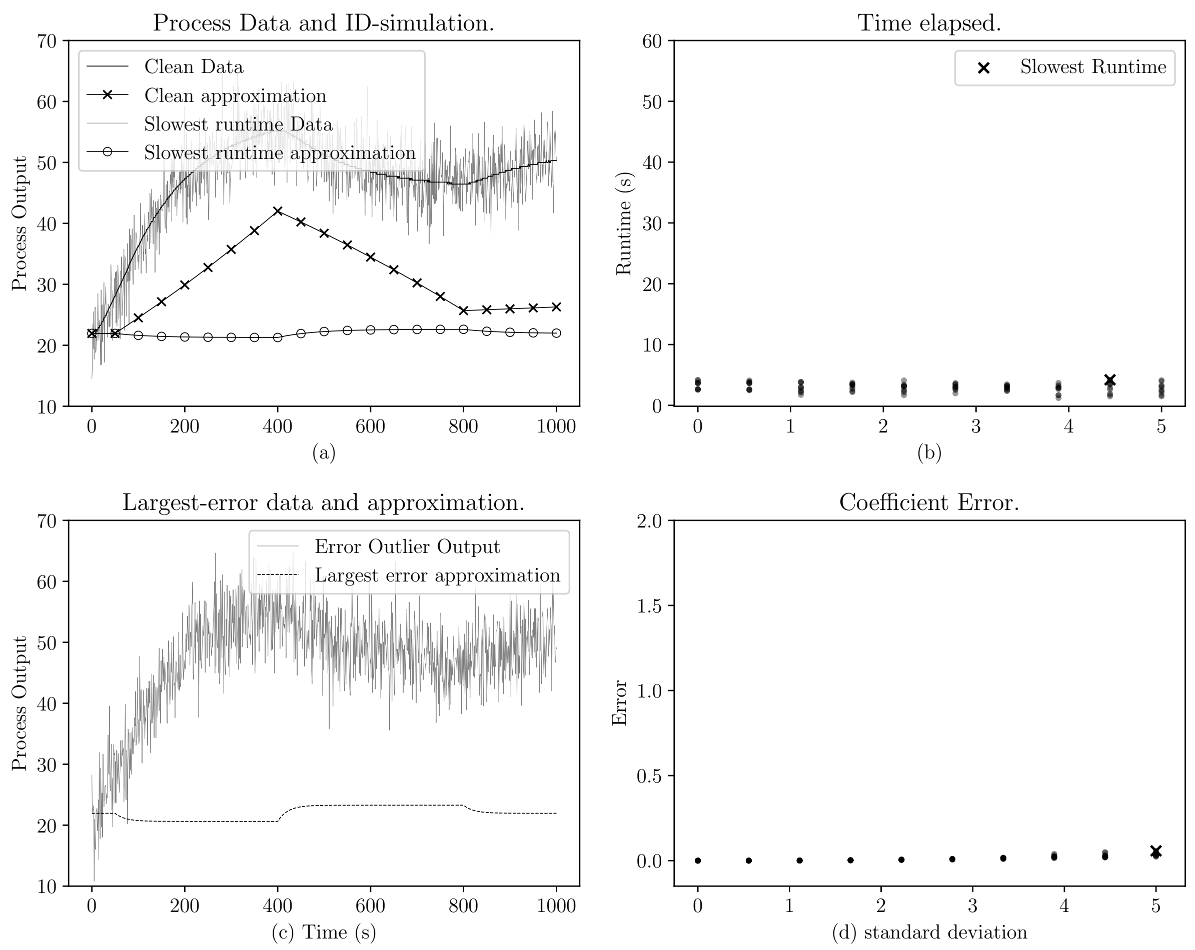

Figure 8: (a) TCL doublet response data and first order ARX model with least squares approximation. (b) Runtimes of each optimization versus the standard deviation. (c) Data and simulation of the data set that produced the largest coefficient error (marker "×" in d). (d) Coefficient error in the ARX model plotted against the standard deviation.

Unfortunately, figure 9.a showed no significant improvement. Similar to the previously mentioned RPI case, the second order form delayed the identification procedure. Furthermore as seen in figure 9.d, this model instance caused larger coefficient errors.

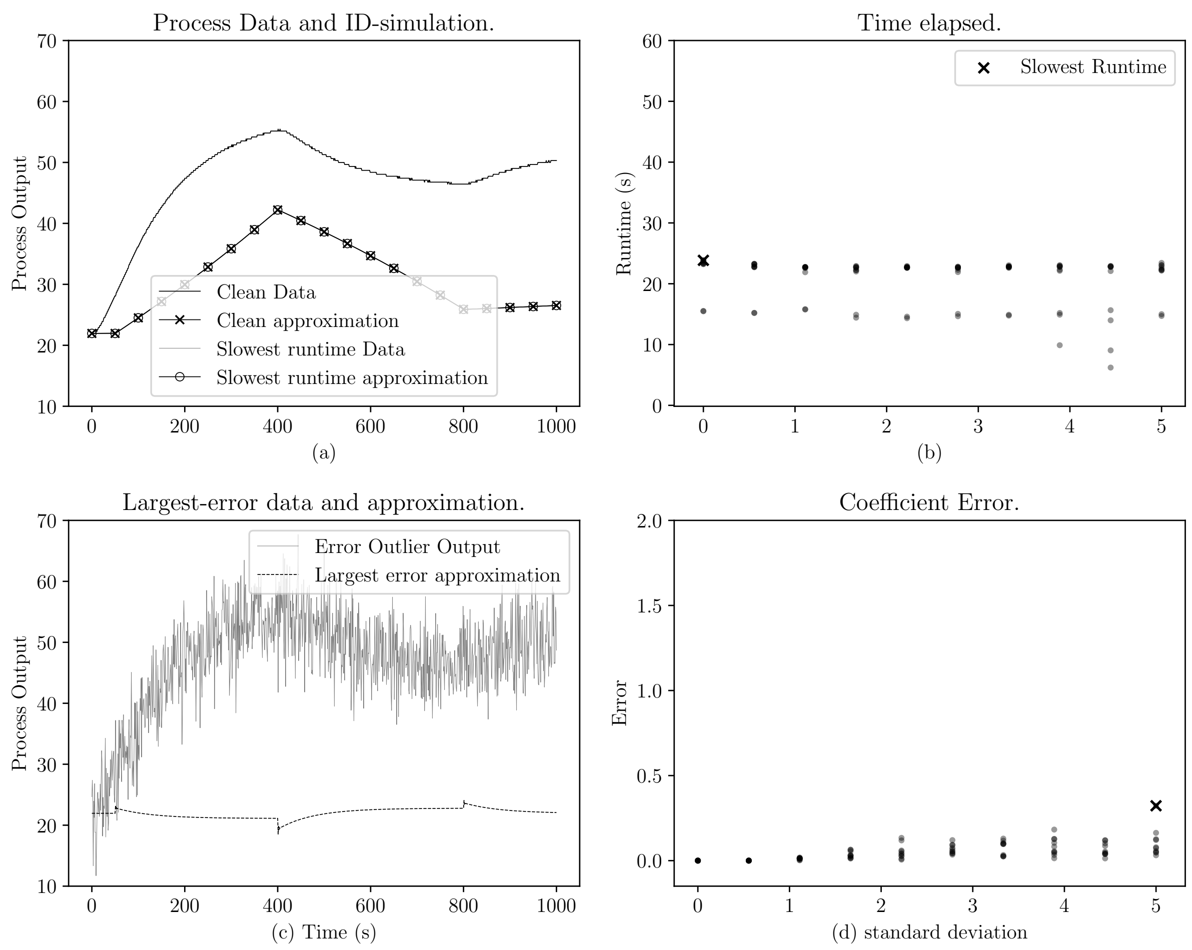

Figure 9: (a) TCL doublet response data and second order ARX model with least squares approximation. (b) Runtimes of each optimization versus the standard deviation. (c) Data and simulation of the data set that produced the largest coefficient error (marker “×” in d). (d) Coefficient error in the ARX model versus standard deviation.

When considering the least squares results collectively, it may be noted that for corresponding model instances the algorithm was consistent in runtime. Furthermore, with the addition of noise, the largest error was less than 0.5. To quantify this, consider that the maximum error per degree of noise did not exceed $\frac{0.5}{5} = 10\%$. This algorithm in combination with the ARX model therefore does not allow large margins for error and relies on the accuracy of the predicted coefficients.

4.2 Optimization Results

4.2.1 Minimization

The graphs in this section depict identification results for the second order ARX model. For this model instance, minimization produced faster run-times than the least squares algorithm. Refer to figures 10, 11 and 12; the run-time range for minimization was on average 10 seconds. This algorithm was almost twice as fast a the least squares runs. Similar to the least squares results, run-times did not vary much with an increase in measurement noise. When little to no measurement noise was present, the minimization algorithm predicted the TCL step response with satisfactory precision as shown in figure 10.a. The clean approximation rarely deviated from the clean data and arrived at the correct steady state value.

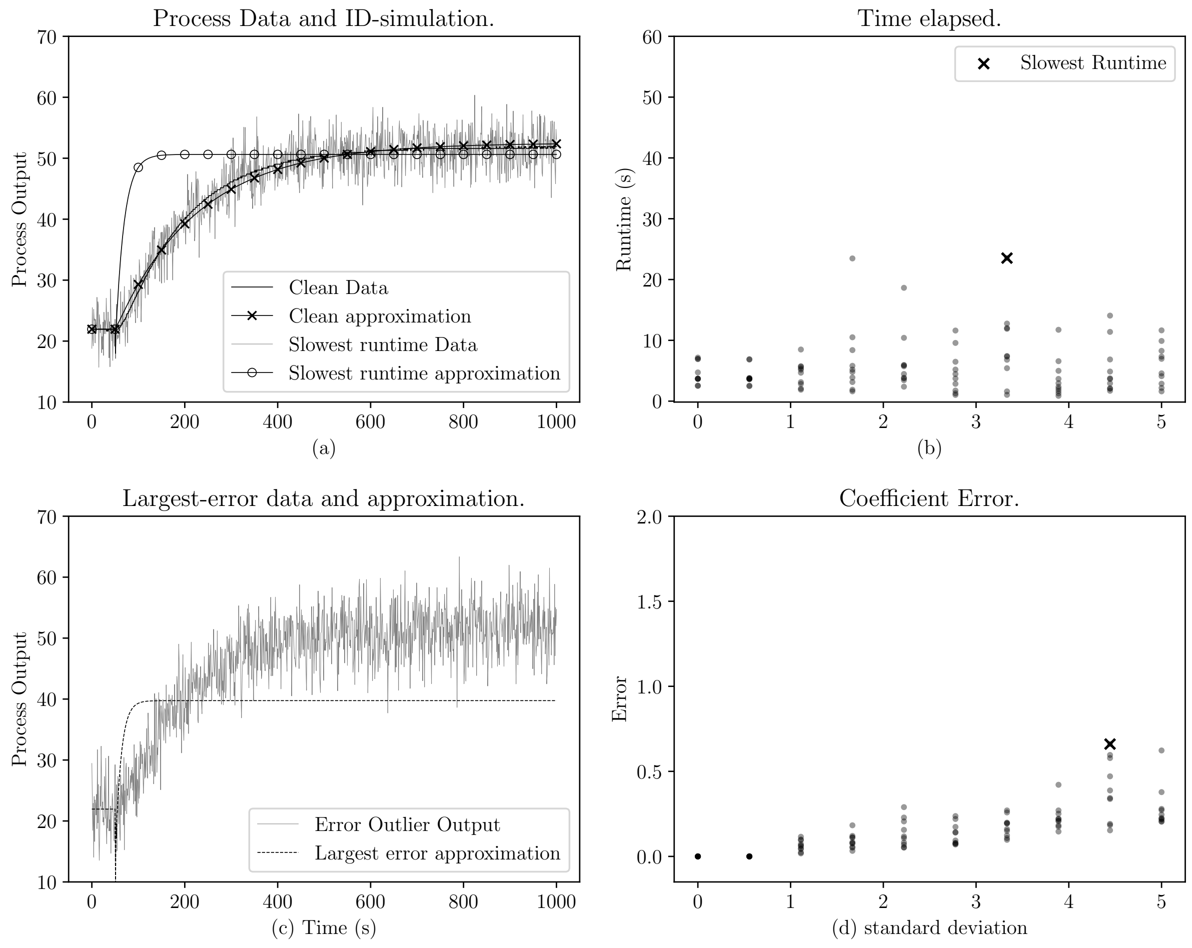

Figure 10: (a) TCL step response data and minimization approximation. (b) Runtimes of each optimization versus the standard deviation. (c) Data and simulation of the data set that produced the largest coefficient error (marker "×" in d). (d) Coefficient error in the ARX model plotted against the standard deviation in added noise.

In figure 11.a, minimization was capable of predicting the RPI response when no measurement noise was present. Similarly to the least squares algorithm, the quality of

RPI-prediction deteriorated rapidly with an increase of measurement noise.

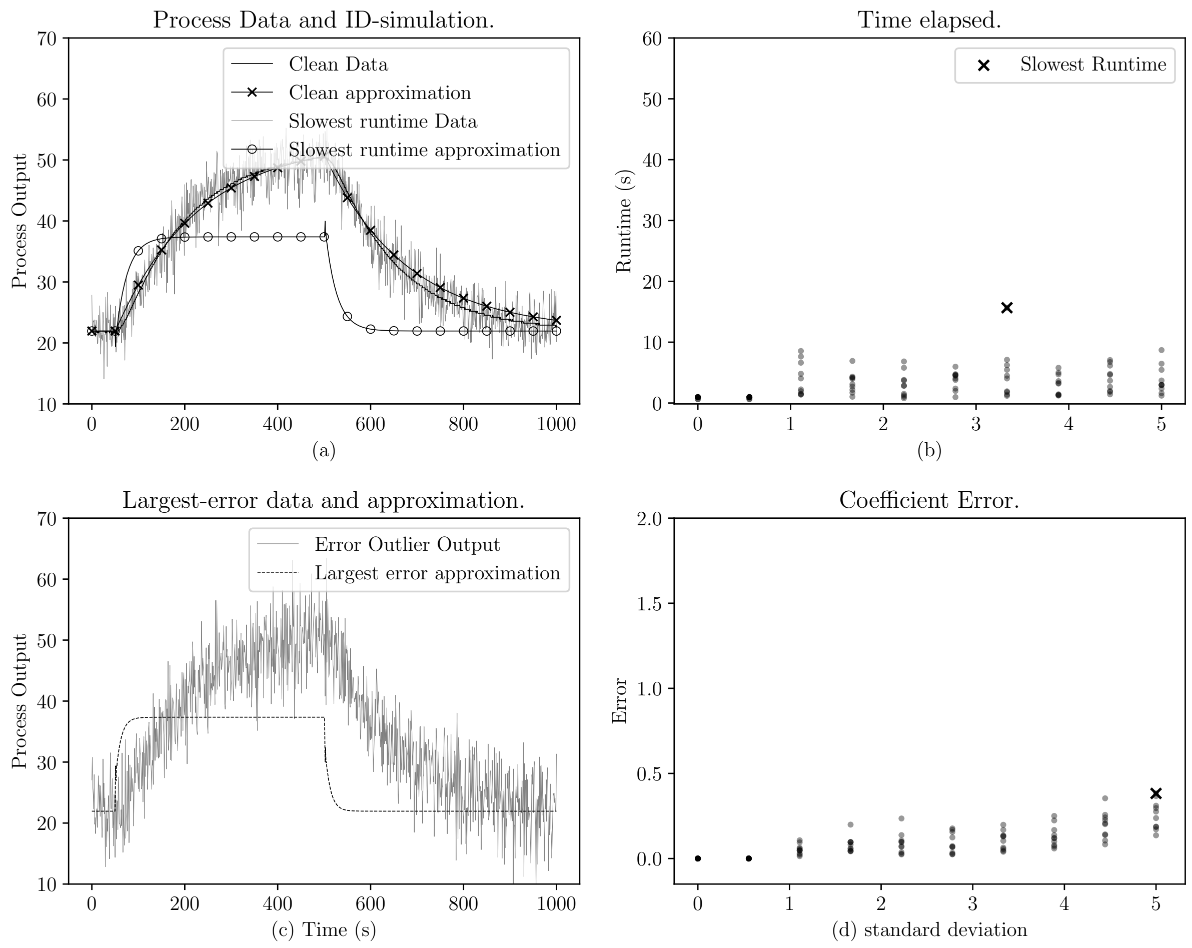

Figure 11: (a) TCL RPI response data and minimization approximation. (b) Runtimes of each optimization versus the standard deviation. (c) Data and simulation of the data set that produced the largest coefficient error (marker “×” in d). (d) Coefficient error in the ARX model plotted against the standard deviation in added noise.

When more variation in process input was experienced, minimization struggled to predict the response. Refer to the clean data approximation in figure 12.a, credit should be given to the fact that, albeit with considerable offset, a similar shape as the data was predicted.

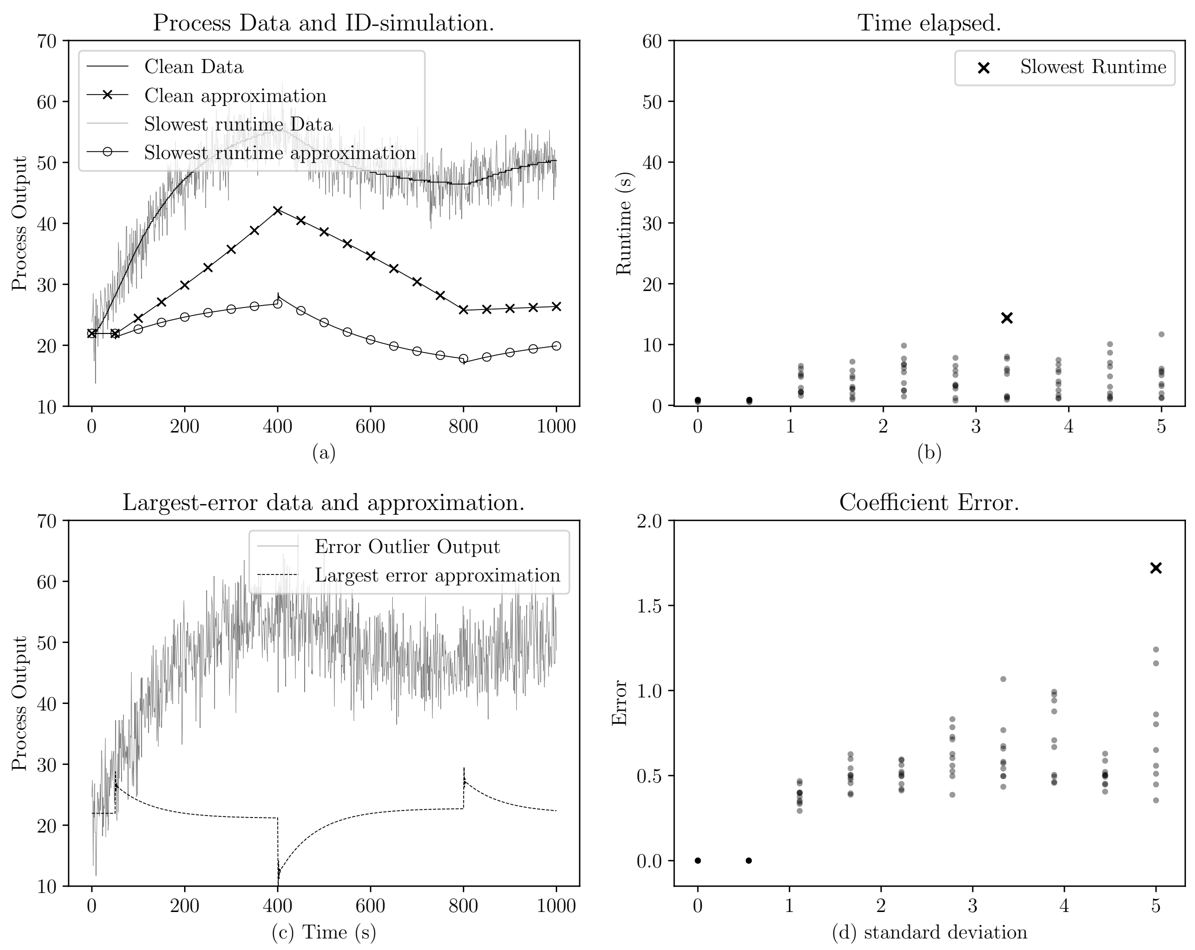

Figure 12: (a) TCL doublet response data and minimization approximation. (b) Runtimes of each optimization versus the standard deviation. (c) Data and simulation of the data set that produced the largest coefficient error (marker “×” in d). (d) Coefficient error in the ARX model plotted against the standard deviation in added noise.

Figure 12.c shows that the approximation failed to predict the correct output. Therefore, when subjected to the combination of input variation and measurement noise, minimization does not prove to be very robust.

The addition of measurement noise had a larger effect on minimization than least squares. In figure 12.d, the maximum coefficient error per standard deviation was approximately 35%. Recall that this value was 10% for the least squares case and notice that the approximation results here closely resemble that of the previous section. This proves that minimization allows larger margins for coefficient error.

4.2.2 Differential Evolution

As depicted in figure 13.a, differential evolution presents ample capability in predicting the step response of the TCL. This algorithm was proven superior to least squares and minimization. Figure 13.c shows that the step approximation came closest to predicting the output in spite of measurement noise since the final steady state offset is smaller. It is interesting to note the decrease in run-time shown in figure 13.b because this indicates that the optimization algorithm terminates if it is evident that the solution may not converge any time soon. According to Y Wu (2011), unstable convergence and the tendency to stop at a local solution are the main drawbacks of differential evolution. The data suggests that an inverse proportionality exists in the relationship between runtime and coefficient error. Accordingly, the slowest runtime may be associated with more accurate predictions.

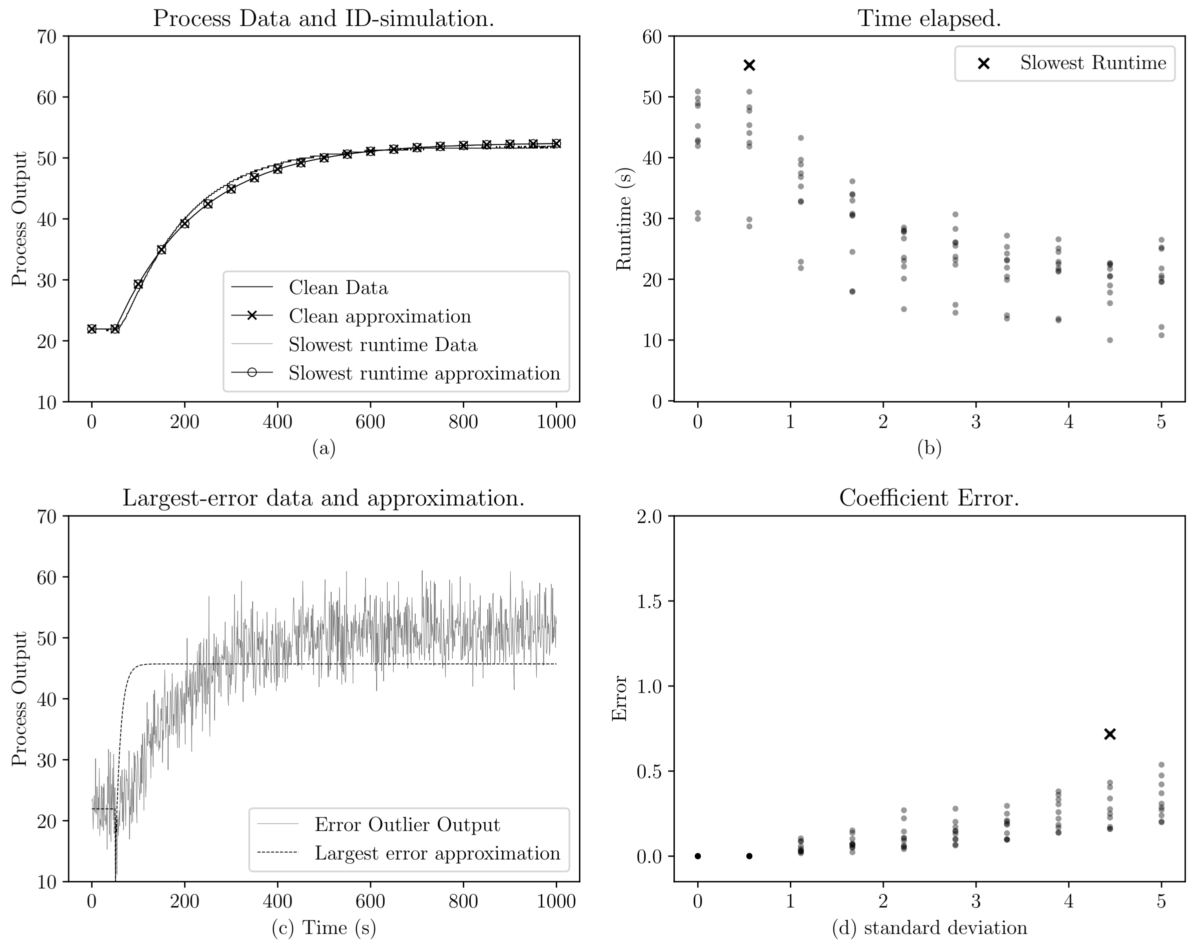

Figure 13: (a) TCL step response data and differential evolution approximation. (b) Runtimes of each optimization versus the standard deviation. (c) Data and simulation of the data set that produced the largest coefficient error (marker "×" in d). (d) Coefficient error in the ARX model plotted against the standard deviation in added noise.

Figures 14.c and 15.c show that this algorithm is sensitive to the model instance.

When the model order was increased, the approximation to noisy data improved significantly.

Once again, this algorithm outperformed the previously tested methods. Compare figures 11.c and 15.c: The DE approximation was more accurate and came closer to predicting the right steady state value.

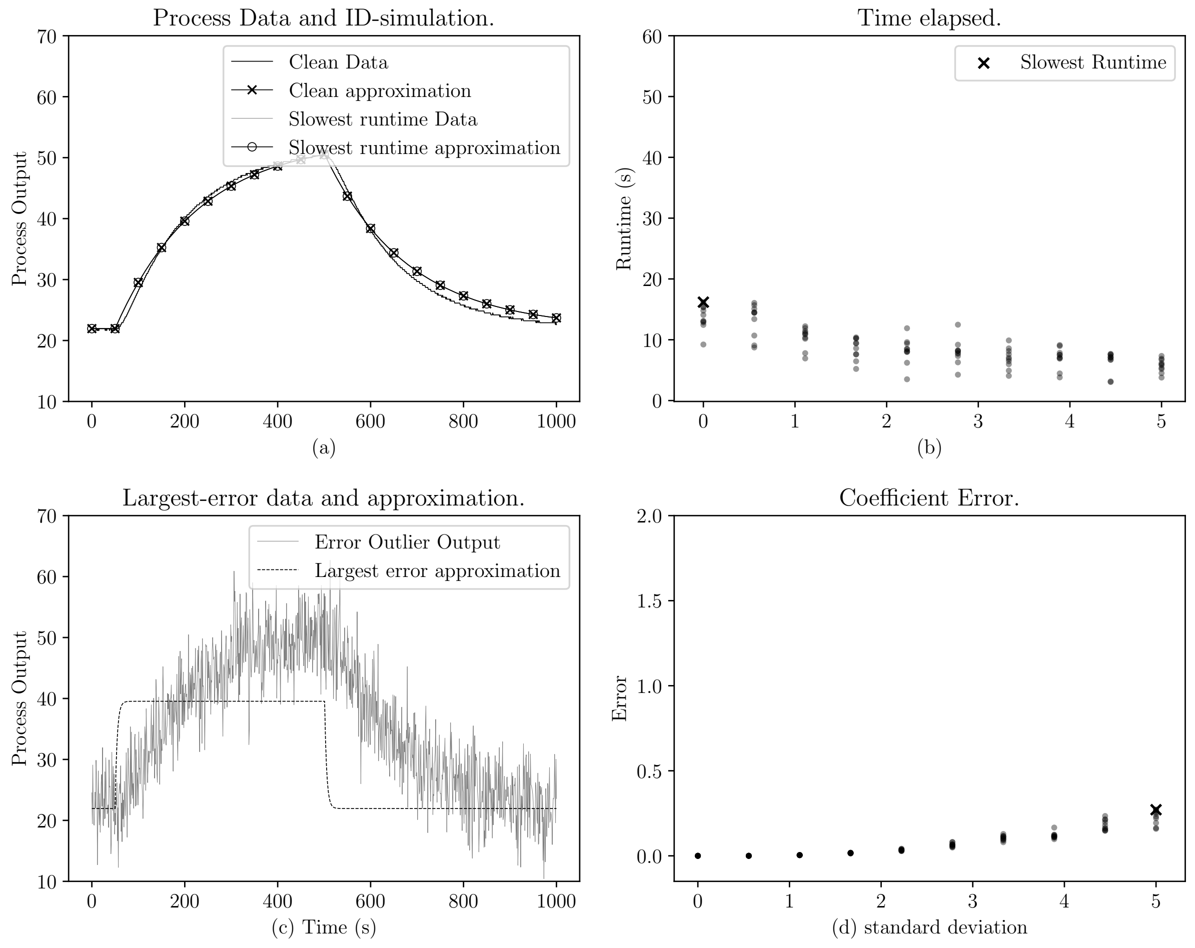

Figure 14: (a) TCL RPI response data and first order ARX model with differential evolution approximation. (b) Runtimes of each optimization versus the standard deviation. (c) Data and simulation of the data set that produced the largest coefficient error (marker "×" in d). (d) Coefficient error in the ARX model plotted against the standard deviation in added noise.

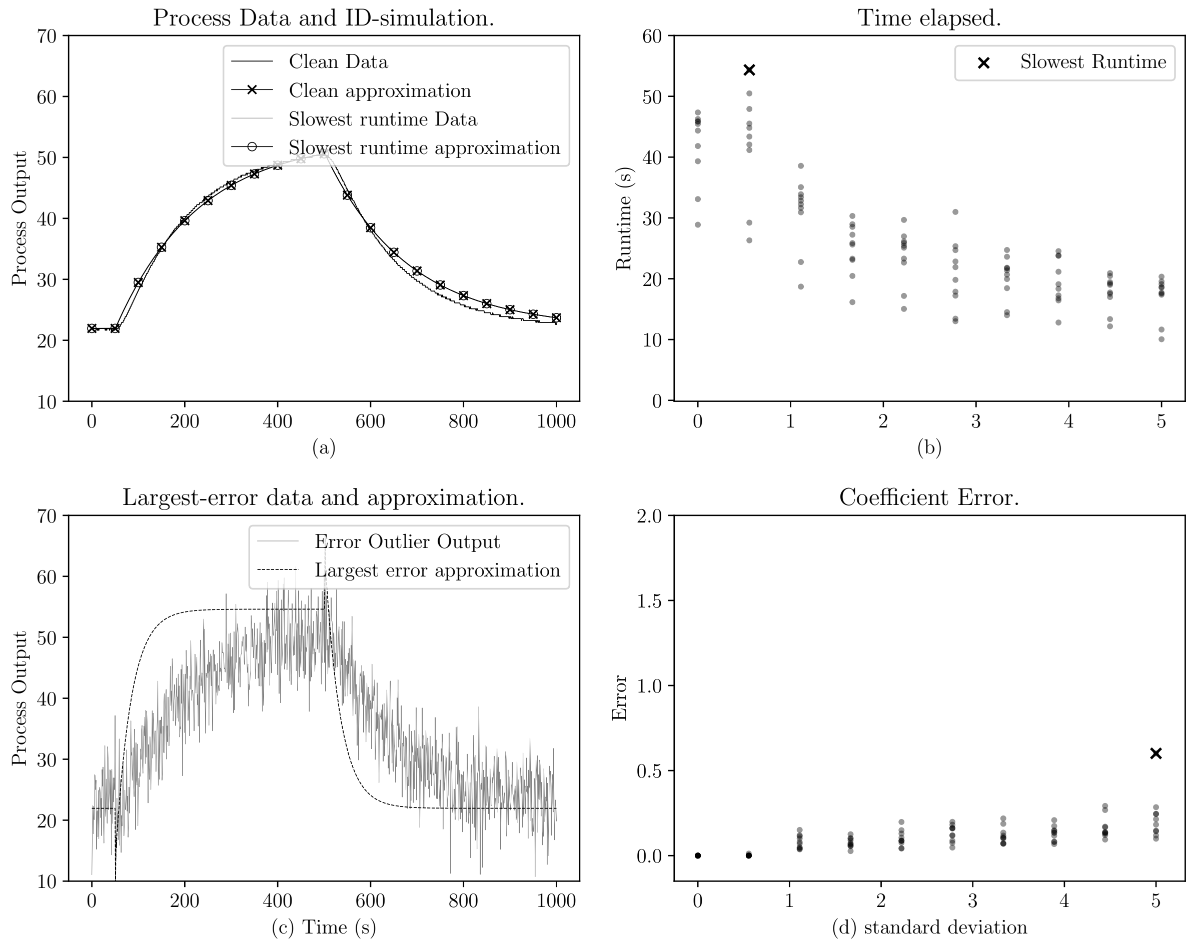

Figure 15: (a) TCL RPI response data and second order ARX model with differential evolution approximation. (b) Runtimes of each optimization versus the standard deviation. (c) Data and simulation of the data set that produced the largest coefficient error (marker "×" in d). (d) Coefficient error in the ARX model plotted against the standard deviation in added noise.

This algorithm performed poorly when predicting the process response to multiple input changes. In figure 16.a, in the absence of noise, the simulation shows more time delay than the TCL and under-predicts the magnitude of the response. Upon the addition of noise, again it may be noted that differential evolution somewhat identifies the shape of the process response as seen in figure 16.c.

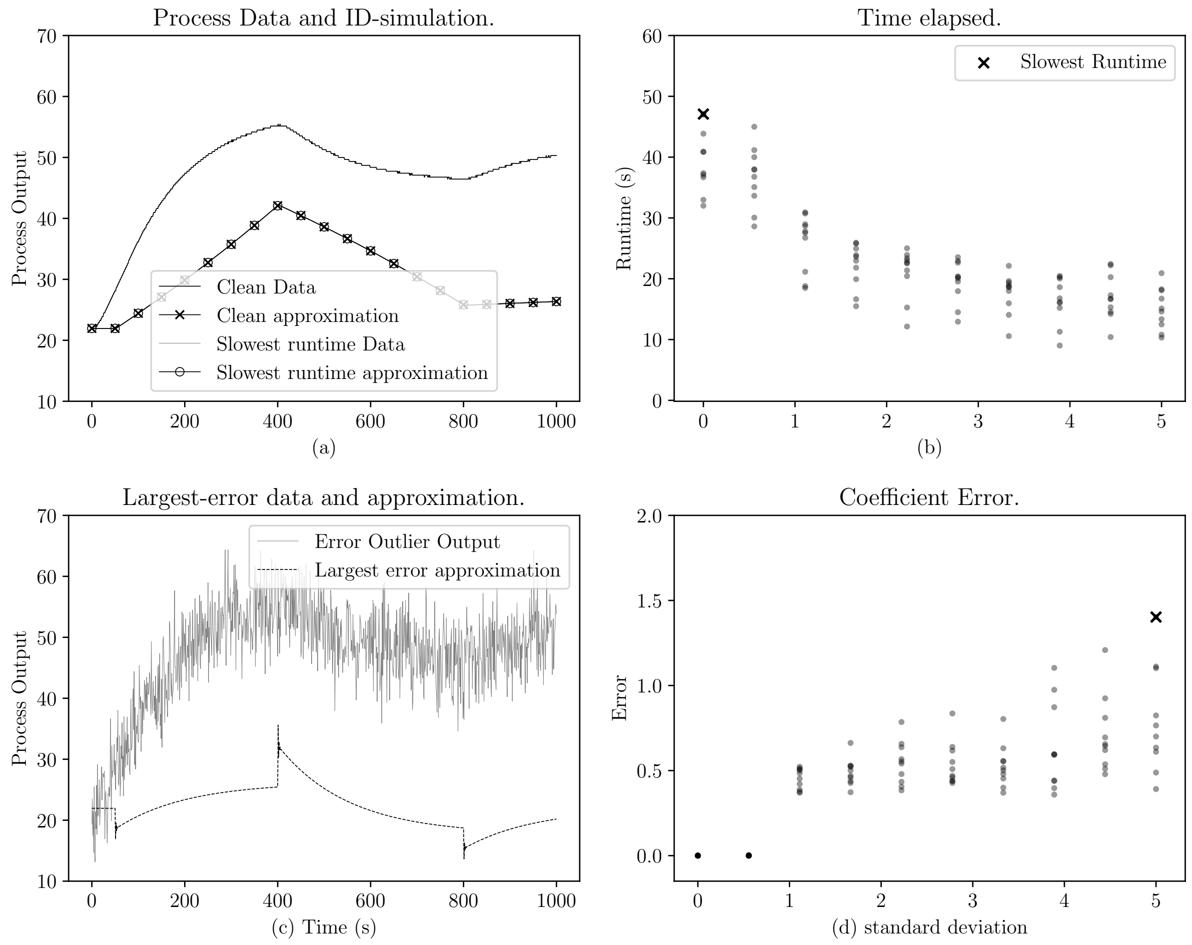

Figure 16: (a) TCL doublet response data and differential evolution approximation. (b) Runtimes of each optimization versus the standard deviation. (c) Data and simulation of the data set that produced the largest coefficient error (marker "×" in d). (d) Coefficient error in the ARX model plotted against the standard deviation in added noise.

Similar trends in coefficient error and runtime exist in the above figures for differential evolution. As the degree of measurement noise increased, the coefficient error increased and the runtime decreased. The range in error per degree of noise was less than 30% and yet, the DE results were better than the least squares results. Therefore when comparing DE to least squares, it may be concluded that this algorithm is less affected by the coefficient error.

5 Future work

Subspace Identification offers an alternative to regression techniques. The parameters of linear time invariant (LTI) state space models are identified with the use of input and output data. LTI dynamic systems may be subjected to deterministic or stochastic inputs B.L.R. De Moor, 1996. Geometric tools are used to project subspaces of the data to find an estimate of its states. Refer to the state space representation below; the data matrices are geometrically manipulated to eliminate the influence of the input matrix u to get a system that is only proportional to the state x.

$$x = Ax + Bu \tag{7}$$

$$y = Cx + Du \tag{8}$$

Python Active-subspaces Utility Library contains some useful tools to implement subspace identification (P Constantine, 2016). Active subspace simplifies a multidimensional process model to one dimension. This application was examined in a study of a photovoltaic solar cell with five input parameters. The analysis provided a one-dimensional model that enabled simpler parametrization of the system. Furthermore, insight to the system's input-output relationships were gained in terms of how sensitive the outputs were to certain inputs.

6 Conclusion

The linear regression algorithm slowed down as measurement noise was increased and prediction quality deteriorated rapidly with a small increase in coefficient error. Increasing the ARX model order caused larger coefficient error, slowed down the identification procedure, and did not improve the fit. Minimization was the fastest technique, but not superior to least squares in terms of predicting the correct process response. However, this technique was more forgiving when larger coefficient errors occurred. The maximum coefficient error per noise degree was larger than that of least squares and yet the process responses were similar in accuracy. In the presence of measurement noise, differential evolution (DE) was the most robust overall. It was discovered that this algorithm was sensitive to model instance since the prediction quality improved for the second order ARX model. Contrary to the findings for least squares and minimization, the DE runtime decreased with an increase in measurement noise. This peculiarity may be attributed to one of the pitfalls in DE which is that it settles on a local solution as noise and input variation is increased. Similarly to minimization, larger coefficient error was tolerated since the predicted process responses remained similar and sometimes better than that of least squares. Optimization was superior to linear regression, but at the cost of more computational power. None of the techniques were able to deal with the doublet input. Therefore research into finding a technique that is more robust in the presence of input variation is recommended.

References

B.L.R. De Moor, PVO (1996). *Subspace Identification For Linear Systems.* Belgium: Kluwer Academic Publishers.

Chinarro, D (2014). *System Engineering Applied to Fuenmayor Karst Aquifer.* Springer International Publishing Switzerland.

E Jones T Oliphant, PP (2001). *SciPy: Open source scientific tools for Python.* URL: `http://www.scipy.org/`.

Emami-Naeini, A (2018). "Model-Based versus PID Control". In: *SC Solutions.*

F Bordoni, AD (1990). "Noise in sensors". In: *Sensors and Actuators* A21-A23, pp. 17–24.

Gevers, M (2006). "A Personal View of the Development of System Identification; A 30-year Journey Through an Exciting Field". In: *IEEE Control Systems Magazine.*

Guidi, H (2008). "Open and Closed-loop Model Identification and Validation". MA thesis. University of Pretoria.

H Yousefi H Handroos, AS (2008). "Application of Differential Evolution in system identification of a servo-hydraulic system with a flexible load". In: *Mechatronics* 18, pp. 513–528.

J Mengshi M.R.W Brake, HS (2019). "Comparison of nonlinear system identification methods for free decay measurements with application to jointed structures". In: *Journal of Sound and Vibration* 453, pp. 268–293.

J.A Royle R.B Chandler, RS (2016). *Goodness of Fit: An Overview.* Elsevier Inc.

J.R Seborg T.F Edgar, DM (2011). *Process Dynamics and Control.* John Wiley and Sons, Inc.

Ljung, L (1996). *System Identification.* CRC Press Inc.

Ljung, L (2008). "Perspectives on System Identification". In: *Division of Automatic Control.*

M Saeid J.Y Mak, JV (2014). *Handbook of Liquefied Natural Gasr*. Elsevier.

McKinney, W (2018). *Pandas: Data Structures for Statistical Computing in Python*. URL: `https://pandas.pydata.org/`.

McNamara, J (n.d.). *NumPy: A Python module for creating Excel XLSX files*. URL: `https://github.com/jmcnamara/XlsxWriter/`.

P Constantine P.M Diaz, LF (2016). *Python Active-subspaces Utility Library*. URL: `https://github.com/paulcon/active_subspaces/`.

S Bedoui, KA (2015). "ARMAX Time Delay Systems Identification Based on Least Square Approach". In: *IFAC-PapersOnline* 48-28, pp. 1100–1105.

T.F Edgar, DH (2001). *Optimization of chemical processes*. New York: McGraw-Hill.

Venter, G (2010). "Review of Optimization Techniques". In: *Encyclopedia of Aerospace Engineering*.

Wright, S (2016). "Optimization". In: *Encyclopaedia Britannica*.

Y Vander Heyden A Nijhuis, JSV (2001). *Guidance for Robustness/Ruggedness Tests in Method Validation*. Laarbeeklaan 103 1090 Brussel Belgium: Vrije Universiteit Brussel.

Y Wu W Lee, CC (2011). "Modified the Performance of Differential Evolution Algorithm with Dual Evolution Strategy". In: *International Conference on Machine Learning and Computing* 3.