Finite Time Joint Parameter and State Estimation Using Python

Manoj Krishna Venkatesan

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

May 2022

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Science. © 2022 Manoj Krishna Venkatesan

Abstract

Parameter and state estimation of linear systems play a significant role in the realm of control systems. More often than not, the parameters and the state values are not readily available and must be deduced from the applied input and the measured output information. Algebraic estimators have been introduced to combat the parameter identification problem, under the major assumption that an accurate measurement of the output signal is available. If the measured output is not noise-free, estimating the output states and their derivatives is not possible. This thesis focuses on the algebraic estimation and filtering of the parameters and the states of linear systems using a forward-backward kernel representation of differential invariants of a system, that will be able to handle measured output signals with high noise values. The kernels, working in tandem with the Kalman filter and Raunch-Tung-Striebel (RTS) smoother, provide accurate parameters and state values even for high noise levels in the measured output. Further, this thesis explains in detail, the construction of a python based estimation library that modularises the algorithms used for state and parameter estimation. This would facilitate applications like implementing model based controllers, and also help in further analysis of a system. This library provides a flexible infrastructure to test the performance of different algorithms against the benchmark results from the Unscented Kalman Filter(UKF). It is a collaborative effort to drive the research forward for further advancements and handling more complex non-homogeneous and non-linear systems.

Résumé

L'estimation de paramètres et états des systèmes linéaires joue un rôle important dans le domaine des systèmes de contrôle. Presque toujours, les paramètres et les valeurs d'état ne sont pas facilement disponibles et ils doivent être déduits par informations d'entrée appliquées et de sortie mesurées. Des estimateurs algébriques ont été introduits pour combattre le problème de l'identification des paramètres, en partant de l'hypothèse principale, qu'une mesure précise du signal de sortie est disponible. Si la sortie mesurée n'est pas exonérée de bruit, l'estimation des états de sortie et de leurs dérivés n'est pas possible. Si la sortie mesurée n'est pas exempte de bruit, l'estimation des états de sortie et de leurs dérivés n'est pas possible. Cette thèse se concentre sur l'estimation algébrique et le filtrage des paramètres et des états des systèmes linéaires en utilisant une représentation par noyau avant-arrière des invariants différentiels d'un système, qui sera capable de traiter des signaux de sortie mesurés avec des valeurs de bruit élevées. Les noyaux fonctionnant en conjonction avec le filtre Kalman et un adoucisseur Rauch-Tung-Striebel (RTS), fournissent des paramètres précis et des valeurs d'état même pour des niveaux de bruit élevés dans la sortie mesurée. En outre, cette thèse explique en détail la construction d'une bibliothèque d'estimation basée sur Python qui modularise les algorithmes utilisés pour l'estimation de l'état et des paramètres. Cela faciliterait applications telles que la mise en œuvre de contrôleurs basés sur des modèles, et aiderait également à une analyse plus approfondie d'un système. Cette bibliothèque fournit une infrastructure flexible pour tester les performances de différents algorithmes contre les résultats de référence du filtre de Kalman sans parfum (UKF). Il s'agit d'un effort de collaboration visant à faire avancer la recherche en vue de nouvelles avancées et de la manipulation de systèmes non-homogènes et non-linéaires plus complexes.

Acknowledgments

It is my pleasure to begin this thesis by thanking my professor and mentor, Dr Hannah Michalska, who was an absolute joy to work with. Her wisdom and guidance proved substantial to the work carried out in this thesis. I will be beholden to her for being bestowed with the opportunity to make a significant contribution to science at such a prestigious university.

I wish to extend my gratitude to my fellow research partner Nithilasaravanan Kuppan for his collaboration and relentless drive, constantly pushing through the horizon of our research. Additionally, I would like to sincerely thank Israel Hernandez Quinto and Adharsh Mahesh Kumaar for their incredible support throughout this arduous journey. This work would not have been possible without the research work done by Luis Medrano del Rosal, Shantanil Bagchi, Nikhil Jayam, Debarshi Patanjali Ghoshal and fellow students of the Systems and Control group under Dr Michalska.

Here's a toast to all my friends - Nithila, Shreya, Vishal, Adharsh, Sai, Mitasha and Ashish for filling my life with laughter and rapture, making my Master's journey memorable. A special thanks to Rhythm and Momo for their constant support. Finally, I wish to dedicate this thesis to my parents for believing in me and supporting me through my ups and downs.

Preface

This section is used to declare that the work presented in this document was completed and carried out by Manoj Krishna Venkatesan and is part of the collaborative work of a three-member team guided by Professor Dr Hannah Michalska. The team includes Nithilasaravanan Kuppan, Israel Hernandez Quinto and myself. The theory of representing an n^{th} order system using kernels in Reproducing Kernel Hilbert Space (RKHS) was derived by Debarshi Patanjali Ghoshal, a PhD scholar in the research group. The theoretical background for the RKHS approaches for optimization is based on the research notes by Professor Hannah Michalska, which is duly acknowledged. The preliminary work for the estimation of a fourth-order system was done by Luis Medrano del Rosal and Adharsh Mahesh Kumaar, while the integration of the Kalman filter and RTS was developed by Nithilasaravanan Kuppan and myself. Subsequently, the development of the Python Estimation Toolkit (PETs) library was the result of collaboration between Nithilasaravanan Kuppan and myself. The extension of this algorithm to handle infinite horizons is developed individually by myself.

Contents

List of Figures

List of Tables

List of Acronyms

RKHS	Reproducing Kernel Hilbert Space
LTI	Linear Time Invariant
LTV	Linear Time Variant
PETs	Python Estimation Toolkits
RMSE	Root Mean Square Error
MAD	Maximum Absolute Difference
MAE	Maximum Absolute Error
SISO	Single Input Single Output
BLUE	Best Linear Unbiased Estimator
OLS	Ordinary Least Squares
GLS	Generalized Least Squares
IV	Instrument Variable
RRLS	Regularized Recursive Least Squares
RTS	Raunch-Tung-Striebel
UKF	Unscented Kalman Filter
EKF	Extended Kalman Filter
BMFLS	Biswas-Mahalanabis Fixed-Lag Smoother
AWGN	Additive Gaussian White Noise
SNR	Signal to Noise Ratio
URTS	Unscented Rauch-Tung-Striebel
JSON	JavaScript Object Notation
CLI	Command Line Interface

Chapter 1

Introduction

1.1 Background and Motivation

Mathematical representation of a system is crucial to understanding the underlying dynamics of a system, as well as establishing efficient control. This system, mathematically known as a **model**, represents the essential characteristics of the system. However, the immediate challenge to this mathematical representation is the identification of system parameters, which in most cases, remain unknown. The process of obtaining the parameters that govern the dynamics of the system using the applied input and the observed output data is known as system identification. The research on system identification is vast, and various proposals were made that depend on the model characteristics. With the advancements in computational prowess, the system identification techniques improved to provide results faster and with better accuracy. One such method, algebraic estimation and filtering of the parameters and the states of linear systems using a forward-backward kernel representation of differential invariants of a system, is analyzed extensively in this thesis.

This solves only a part of the problem. It is also necessary to have knowledge of the state of the system. The state of the system is the minimum set of variables whose present values together with the values of input signals in the future, completely determine the future behaviour of the system[1]. The accurate estimation of system states is crucial to driving the system to the desired future state. This poses a myriad of challenges, specifically because of the underlying process and measurement noise in the system. It is not feasible to obtain accurate parameters and output values in most applications. Hence it is vital for the state estimation algorithm to filter the noise and provide an accurate state estimate. In this thesis,

various techniques involved in state estimation are covered and their performance is analyzed with examples.

1.2 State and Parameter estimation

A model is a mathematical representation of the essential characteristics of an existing system. When a system model can be defined by a finite number of variables and parameters, it is known as a parametric model. To implement a parametric model-based controller, it is necessary to know precisely the structure of the model of the system and its associated parameters[1]. This makes parameter identification one of the most crucial processes involved in the control system design. In most often circumstances, the measured output signal consists of a high level of measurement noise with unknown characteristics. Hence, it is pivotal to have a robust algorithm that efficiently filters the noise in a short period and provides an accurate description of the system parameters.

The least squares algorithm was the earliest known method for solving estimation problems, which predates 1809 when Carl Friedrich Gauss[2] published his method of calculating the orbits of celestial bodies. This method is based on the minimization of the sum of squares of residuals (known as the error function or cost function). It was followed by the introduction of maximum likelihood by Sir Ronald Aylmer Fisher in 1922. The inception of this method and its evolution by R.A. Fischer is explained thoroughly in [3]. Later, linear estimation for stochastic systems was introduced in the works of Wiener (1949) and Kolmogorov (1941)[1], [4],[5]. There are recent developments in algebraic methods for parameter and state estimation where time differentiations of expression, multiplications by positive powers of the time variable, and iterated integrations are sufficient to obtain linear expressions in the parameters without the knowledge of the initial condition of the system[1].

The joint parameter and state estimation of linear systems based on Hilbert-Schmidt forward-backward double-sided kernel has the potential to handle high noise in output measurement[6]. This algebraic kernel estimator works over a finite time interval under the assumption that the system is observable. A system is stated observable if its state-space can be reconstructed at any finite instant, in terms of the input, the output, and their respective time derivatives of finite order. One major challenge of this method is its sensitivity to measurement noise. To address this issue, the integral regression equation is rewritten point-wise as a matrix equation using finite, distinct time instances, known as *knots*[6]. This matrix equation is then treated as a least squares problem. The performance of parameter estimation is further improved by using multiple regression. The multiple regression algorithm can be extended

to a computationally feasible Regularized Recursive Least Squares (RRLS) algorithm where the covariance matrix, parameters and state are updated recursively from the previously estimated parameters and state values. This method is powerful and efficient since it doesn't require the complete information of the output in the time interval for parameter estimation. The reconstruction of the state is instrumental for this algorithm and has to be performed at every recursive step.

In addition to the parameters, the states of the system must be estimated simultaneously. In the algebraic parameter estimation discussed above, the states of the system must be reconstructed at each recursive step. Luenberger observer[7], proposed by D.Luenberger in 1964 is based on a simple feedback loop aimed at minimising the error in the estimated states. But this observer ensures convergence of states under a major assumption that the system is observable and deterministic. This limitation is addressed in the groundbreaking work of Rudolf E.Kalman, which allowed estimation of states of stochastic systems using the Kalman Filter[8]. This thesis proves the dexterity of the Kalman filter in its robustness to noise, computational simplicity and ease of implementation. The major advantage that the Kalman filter has over its contemporaries is that it accommodates the possibility of process and measurement uncertainties and uses a stochastic optimization technique to filter the noises. The Kalman filter is improved further by using a fixed interval smoother, namely Raunch-Tung-Striebel smoother[9] developed by Raunch, Tung and Striebel.

1.3 Python based estimation library

The parameter and state estimation strategies for SISO LTI systems are discussed and analyzed in this thesis to a great extent. However, to have a practical implication of this research, it is essential to ensure that the results are reproducible and the deployed algorithms are user-friendly. It involves making the algorithms generic with configurable parameters and the system of interest, order-agnostic. It is also important to keep in mind that to reap maximum benefit from this research, the algorithms must be accessible to the students, universities, companies and aspirants alike. Hence, developing this library in a community-driven, open-source framework seemed beneficial. Python is one of the most commonly used open-source programming languages with enormous support in terms of scientific libraries like NumPy, SciPy, Pandas and Matplotlib, which facilitates both easy and stable implementation[10]. Owing to this record, a python based estimation library is meticulously constructed, taking into consideration the factors of reliability, scalability, and ease of implementation. It ensures that the developed algorithms are modular with clearly defined assumptions and adaptable

to different applications with proper configuration. In addition, this library is accommodating to improvisation and extension, like support for Linear Time Variant (LTV) systems and non-linear systems[11],[12],[13] in future. This python library is developed through the combined effort of my fellow lab partner Nithilasaravanan Kuppan[14] and myself, standing testimonial to the years of painstaking research carried out by Dr Hannah Michalska and the graduate students under her supervision. A comprehensive explanation of the construction of this library and the method of usage is present in chapter 6.

1.4 Thesis objectives and organization

The primary objective of this thesis is to extend the parameter estimation algorithm developed using Hilbert-Schmidt forward-backward double sided kernel[6] and Recursive Regularized Least Squares algorithm to perform accurate state estimation using Kalman filter and RTS smoother for higher-order SISO LTI systems with various noise levels. It is followed by the construction of the estimation library - python library, detailing the library architecture, dependencies, usage, examples, and scope for future works.

The organization of this thesis can be summarized as follows:
Chapter 1 provides a background to the estimation problem, followed by a brief introduction to the research works which has already been carried out in that area. It also outlines the scope and objective of this thesis.

Chapter 2 introduces the double sided kernel approach, which is detailed in [6] and focuses on understanding the kernel equations and supporting theorems for SISO LTI systems.

Chapter 3 discusses the kernel-based multiple regression techniques, and the regression equations are derived for a 4^{th} order LTI system along with the Recursive Regularized Least Squares (RRLS) algorithm. A case study is carried out with a 4^{th} order LTI system and the results are presented.

Chapter 4 provides a detailed analysis of a state estimation problem, indulging the working of the Kalman filter and the Rauch–Tung–Striebel (RTS) smoother. It is followed by the integration of the Kalman filter and the RTS smoother to the RRLS algorithm. Finally, the

results are presented where the performance of the algorithm is discussed with a couple of examples.

Chapter 5 discusses the working of the Unscented Kalman Filter (UKF) algorithm to simultaneously estimate the parameters and the states of the SISO LTI systems. The performance of this algorithm is compared with the Kernel-based RRLS algorithm using a case study.

Chapter 6 elaborates on the necessity for a python based estimation library, followed by the assumptions and requirements. This chapter explains in easy to follow procedures - the steps involved in using this library like installation, input preparation, algorithm selection and algorithm execution, with use cases and examples.

Chapter 7 extends the current research presented in the thesis to cover a particular use case - state estimation of a SISO LTI system in the infinite horizon, the use case which is of paramount importance for a control system design. The preliminary results and strategies for future work follow.

Chapter 2

Parameter and State Estimation of LTI systems - Double sided Kernel Approach

The general problem of system identification in Linear Time Invariant (LTI) Systems is defined in this chapter. It is followed by the double-sided kernel approach [6] for parameter estimation, which does not require any knowledge of the underlying dynamics of the system. This method employs a forward-backward integration to convert a high order differential equation into an integral form with no singularities at the boundaries of the finite time interval.

2.1 Finite interval estimation problem for LTI systems[11]

Consider a general n^{th} order, strictly proper and minimal Single Input Single Output (SISO) LTI system in state space form evolving on a given finite time interval $[a, b] \subset \mathbb{R}$:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{2.1}$$

with $x \in \mathbb{R}^n$, the state vector; $x(0) = x_0$, the initial state vector; $\dot{x} = dx/dt$; $y \in \mathbb{R}$ is the measured output; $u \in \mathbb{R}$ is the applied input. The matrices $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times 1}, C \in \mathbb{R}^{1\times n}$ are given by,

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix}, \tag{2.2}$$

(2.3)

$$B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{2.4}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \end{bmatrix} \tag{2.5}$$

with matching dimensions of the system matrices and the characteristic equation

$$y^{(n)}(t) + a_{n-1}y^{(n-1)}(t) + \cdots + a_1 y^{(1)}(t) + a_0 y(t) = 0 \tag{2.6}$$

The input-output equation for system (2.1) becomes

$$y^{(n)}(t) + a_{n-1}y^{(n-1)}(t) + ... + a_1 y^{(1)}(t) + a_0 y(t) = -b_{n-1}u^{(n-1)}(t) - ... - b_0 u(t) \tag{2.7}$$

where b_i for $i = 0, ..., n-1$ are the coefficients of the polynomial in the numerator of the rational transfer function for (2.1). The unknown parameters a_i and b_i for $i = 0, .., n-1$ need to be estimated from noisy observations of the system's output, $y_M(t)$ for $t \in [a, b]$.

The estimation problem is stated as follows. Given an arbitrary finite interval of time $[a, b]$, suppose that:

(1) The dimension of the state vector of the LTI system is known a priori.

(2) The system input function $u(t)$ is equal to zero (No input).

(3) The output of the system is observed as a single realization of a 'continuous' measurement process $y_M(t) := y(t) + \eta(t), t \in [a, b]$ in which η denotes additive white Gaussian noise with unknown intensity (variance) σ^2.

An implementable version of assumption (3) simply requires availability of an unrestricted number of output measurements over the observation horizon $[a, b]$.

2.2 Kernel representation of a system differential invariance[6]

The kernel representation of the nth order SISO LTI system was defined in [6]. The key to finite interval estimation approach is the integral representation of the controlled differential invariance of the system 2.7. The parameter estimation of a homogeneous system can be viewed as the identification of a differential invariant $\mathcal{I}$ ($\mathcal{I} \equiv 0$, $\equiv$ is 'equivalent to') which remains constant in the absence of external input:

$$\begin{aligned} &\mathcal{I}(t, y(t), y^{(1)}(t), \cdots, y^{(n)}(t)) \\ &\quad = y^{(n)}(t) + a_{n-1} y^{(n-1)}(t) + \cdots + a_0 y(t) \equiv const. = 0 \ ; \ t \in [a, b] \end{aligned} \tag{2.8}$$

Equation 2.8 delivers additional measurement-noise independent information about the behaviour of the system in addition to the noisy signal y_M. The **zero-input response** characterization (2.8) has to be put in a form, which does not depend on the initial or boundary conditions of the system, and that does not involve any time derivatives of the output as they cannot be measured directly. Such a system characterization is given in terms of the theorems below.

Definition 2.1.

A pair of smooth (class C^∞) functions (α_a, α_b), $\alpha_s : [a, b] \longrightarrow \mathbb{R}, s = a$ or b, is an annihilator of the boundary conditions for system (2.1) if the functions α_s are non-negative, monotonic, vanish with their derivatives up to order $n - 1$ at the respective ends of the interval $[a, b]$; i.e.

$$\alpha_s^{(i)}(s) = 0; \quad i = 0, \ldots, n-1; \quad s = a, b; \quad \alpha_s^{(0)} \equiv \alpha_s \tag{2.9}$$

and such that their sum is strictly positive, i.e. for some constant $c > 0$

$$\alpha_{ab}(t) := \alpha_a(t) + \alpha_b(t) > c\,;\; t \in [a, b] \tag{2.10}$$

A simplest example of such an annihilator for system (2.1) is the pair,

$$\begin{aligned} &\alpha_a(t) := (t-a)^n; \quad \alpha_b(t) := (b-t)^n; \quad t \in [a, b] \\ &\alpha_{ab}(t) := \alpha_a(t) + \alpha_b(t) > 0 \\ &\alpha_{ab}(s) = (b-a)^n; \quad s = a, b \end{aligned} \tag{2.11}$$

Indeed, it is easy to see that (2.11) holds for all $n \geq 1$ because $\alpha_{ab}(a) = \alpha_{ab}(b) = (b-a)^n > 0$ and, for $n \geq 2$, α_{ab} has a unique stationary point $t^* = 0.5(a+b) \in [a, b]$ at which $\alpha_{ab}(t^*) = (0.5)^{n-1}(b-a)^n$.

Employing this particular annihilator the integral representation for system (2.1) is rendered by the following:

Theorem 2.2.1 *There exist Hilbert-Schmidt kernels $K_{DS,y}, K_{DS,u}$, such that input and output functions u and y of system* (2.1) *satisfy*

$$y(t) = \alpha_{ab}^{-1}(t, n)\left[\int_a^b K_{DS,y}(n, t, \tau)y(\tau)\ \mathrm{d}\tau + \int_a^b K_{DS,u}(n, t, \tau)u(\tau)\ \mathrm{d}\tau\right] \tag{2.12}$$

with,

$$\alpha_{ab}^{-1}(t, n) = \frac{1}{(t-a)^n + (b-t)^n} \tag{2.13}$$

Hilbert-Schmidt double-sided kernels of equation (2.12) *are square integrable functions on $L^2[a, b] \times L^2[a, b]$ and are expressed in terms of the forward and backward kernels given*

below:

$$K_{DS,y}(n,t,\tau) \triangleq \begin{cases} K_{F,y}(n,t,\tau), & \text{for} \quad \tau \leq t \\ K_{B,y}(n,t,\tau), & \text{for} \quad \tau > t \end{cases} \tag{2.14}$$

$$K_{DS,u}(n,t,\tau) \triangleq \begin{cases} K_{F,u}(n,t,\tau), & \text{for} \quad \tau \leq t \\ K_{B,u}(n,t,\tau), & \text{for} \quad \tau > t \end{cases} \tag{2.15}$$

The kernel functions $K_{DS,y}, K_{DS,u}$ are n - 1 times differentiable functions of t. The forward kernels $K_{F,y}(n,t,\tau), K_{F,u}(n,t,\tau)$ and backward kernels $K_{B,y}(n,t,\tau), K_{B,u}(n,t,\tau)$ in equation (2.14) *and* (2.15) *are given below:*

$$\begin{aligned} K_{F,y}(n,t,\tau) = &\sum_{j=1}^{n}(-1)^{j+1}\binom{n}{j}\frac{n!(t-\tau)^{j-1}(\tau-a)^{n-j}}{(n-j)!(j-1)!} \\ &+\sum_{i=0}^{n-1}a_i\sum_{j=0}^{i}(-1)^{j+1}\binom{i}{j}\frac{n!(t-\tau)^{n-i+j-1}(\tau-a)^{n-j}}{(n-j)!(n-i+j-1)!} \end{aligned} \tag{2.16}$$

$$\begin{aligned} K_{B,y}(n,t,\tau) = &\sum_{j=1}^{n}\binom{n}{j}\frac{n!(t-\tau)^{j-1}(b-\tau)^{n-j}}{(n-j)!(j-1)!} \\ &+\sum_{i=0}^{n-1}a_i\sum_{j=0}^{i}\binom{i}{j}\frac{n!(t-\tau)^{n-i+j-1}(b-\tau)^{n-j}}{(n-j)!(n-i+j-1)!} \end{aligned} \tag{2.17}$$

$$K_{F,u}(n,t,\tau) = \sum_{i=0}^{n-1}b_i\sum_{j=0}^{i}(-1)^{j+1}\binom{i}{j}\frac{n!(t-\tau)^{n-i+j-1}(\tau-a)^{n-j}}{(n-j)!(n-i+j-1)!} \tag{2.18}$$

$$K_{B,u}(n,t,\tau) = \sum_{i=0}^{n-1}b_i\sum_{j=0}^{i}\binom{i}{j}\frac{n!(t-\tau)^{n-i+j-1}(b-\tau)^{n-j}}{(n-j)!(n-i+j-1)!} \tag{2.19}$$

The proof for Theorem 2.2.1, can be found in [6] (pp. 153-157). The integral kernel representation of the system in Theorem 2.2.1 eliminates the need for boundary conditions as they are annihilated during every integration operation by the presence of the annihilating factors α_a and α_b. The following conjecture is then quite obvious.

Corollary 2.2.2 *For any given input function u, the output function $y : [a, b] \to \mathbb{R}$ satisfies the system input-output equation* (2.7) *on the interval $[a, b]$ if and only if it satisfies the integral equation* (2.12) *regardless of any boundary conditions that may be imposed. The kernel representation of the system invariance provides a unique criterion whose reproducing property unambiguously characterizes all zero input solutions of the SISO LTI system. In particular all the fundamental solutions of the LTI system share the reproducing property* (2.12) *as they span a subspace of the RKHS of dimension n.*

□

Explicit kernel expressions for the derivatives of the output function:

The time derivatives of the system output $y^{(k)}, k = 1, \ldots, n-1$ is obtained using the following Theorem 2.2.3.

Theorem 2.2.3 *There exist Hilbert-Schmidt kernels $K_{F,k,y}, K_{F,k,u}, K_{B,k,y}, K_{B,k,y}, k = 1, \ldots, n-1$ such that the derivatives of the output function in* (2.1) *can be computed recursively as follows:*

$$
\begin{aligned}
y^{(k)}(t) =& \frac{1}{(t-a)^n + (b-t)^n} \Bigg[\sum_{i=1}^{k} (-1)^{i+1} \binom{p+i-1}{i} \frac{n!(t-a)^{n-i} y^{(k-i)}(t)}{(n-i)!} \\
&+ \sum_{i=p}^{n-1} a_i \sum_{j=0}^{i-p} (-1)^{j+1} \binom{p+j-1}{j} \frac{n!(t-a)^{n-j} y^{(i-j-p)}(t)}{(n-j)!} \\
&+ \int_a^t K_{F,k,y}(n,p,t,\tau) y(\tau) d\tau + \int_a^t K_{F,k,u}(n,p,t,\tau) u(\tau) d\tau \\
&+ \sum_{i=p}^{n-1} b_i \sum_{j=0}^{i-p} (-1)^{j+1} \binom{p+j-1}{j} \frac{n!(t-a)^{n-j} u^{(i-j-p)}(t)}{(n-j)!}
\end{aligned}
$$

$$
\begin{aligned}
&- \sum_{i=1}^{k} \begin{pmatrix} p+i-1 \\ i \end{pmatrix} \frac{n!(b-t)^{n-i} y^{(k-i)}(t)}{(n-i)!} \\
&- \sum_{i=p}^{n-1} a_i \sum_{j=0}^{i-p} \begin{pmatrix} p+j-1 \\ j \end{pmatrix} \frac{n!(b-t)^{n-j} y^{(i-j-p)}(t)}{(n-j)!} \\
&+ \int_t^b K_{B,k,y}(n,p,t,\tau) y(\tau) d\tau + \int_t^b K_{B,k,u}(n,p,t,\tau) u(\tau) d\tau \\
&- \sum_{i=p}^{n-1} b_i \sum_{j=0}^{i-p} \begin{pmatrix} p+j-1 \\ j \end{pmatrix} \frac{n!(b-t)^{n-j} u^{(i-j-p)}(t)}{(n-j)!} \Bigg]
\end{aligned}
$$

where $p = n - k$ *and:*

$$
\begin{aligned}
K_{F,k,y}(n,p,t,\tau) = &\sum_{j=1}^{p} (-1)^{j+n-p+1} \begin{pmatrix} n \\ n-p+j \end{pmatrix} \frac{n!(t-\tau)^{j-1}(\tau-a)^{p-j}}{(p-j)!(j-1)!} \\
&+ \sum_{i=0}^{p-1} a_i \sum_{j=0}^{i} (-1)^{j+1} \begin{pmatrix} i \\ j \end{pmatrix} \frac{n!(t-\tau)^{p-i+j-1}(\tau-a)^{n-j}}{(n-j)!(p-i+j-1)!} \\
&+ \sum_{i=p}^{n-1} a_i \sum_{j=1}^{p} (-1)^{j+i-p+1} \begin{pmatrix} i \\ i-p+j \end{pmatrix} \frac{n!(t-\tau)^{j-1}(\tau-a)^{n-i+p-j}}{(n-i+p-j)!(j-1)!}
\end{aligned} \tag{2.20}
$$

$$
\begin{aligned}
K_{F,k,u}(n,p,t,\tau) = &\sum_{i=0}^{p-1} b_i \sum_{j=0}^{i} (-1)^{j+1} \begin{pmatrix} i \\ j \end{pmatrix} \frac{n!(t-\tau)^{p-i+j-1}(\tau-a)^{n-j}}{(n-j)!(p-i+j-1)!} \\
&+ \sum_{i+p}^{n-1} b_i \sum_{j=1}^{p} (-1)^{j+i-p+1} \begin{pmatrix} i \\ i-p+j \end{pmatrix} \frac{n!(t-\tau)^{j-1}(\tau-a)^{n-i+p-j}}{(n-i+p-j)!(j-1)!}
\end{aligned} \tag{2.21}
$$

$$
\begin{aligned}
K_{B,k,y}(n,p,t,\tau) = &\sum_{j=1}^{p} \begin{pmatrix} n \\ n-p+j \end{pmatrix} \frac{n!(t-\tau)^{j-1}(\tau-a)^{p-j}}{(p-j)!(j-1)!} \\
&+ \sum_{i=0}^{p-1} a_i \sum_{j=0}^{i} \begin{pmatrix} i \\ j \end{pmatrix} \frac{n!(t-\tau)^{p-i+j-1}(b-\tau)^{n-j}}{(n-j)!(p-i+j-1)!} \\
&+ \sum_{i=p}^{n-1} a_i \sum_{j=1}^{p} \begin{pmatrix} i \\ i-p+j \end{pmatrix} \frac{n!(t-\tau)^{j-1}(b-\tau)^{n-i+p-j}}{(n-i+p-j)!(j-1)!}
\end{aligned} \tag{2.22}
$$

$$
\begin{aligned}
K_{B,k,u}(n,p,t,\tau) = &\sum_{i=0}^{p-1} b_i \sum_{j=0}^{i} \begin{pmatrix} i \\ j \end{pmatrix} \frac{n!(t-\tau)^{p-i+j-1}(b-\tau)^{n-j}}{(n-j)!(p-i+j-1)!} \\
&+ \sum_{i+p}^{n-1} b_i \sum_{j=1}^{p} \begin{pmatrix} i \\ i-p+j \end{pmatrix} \frac{n!(t-\tau)^{j-1}(b-\tau)^{n-i+p-j}}{(n-i+p-j)!(j-1)!}
\end{aligned}
\tag{2.23}
$$

The system parameters $a_i, b_i, i = 0, \ldots, n-1$ in Theorem 2.2.3 can be obtained using algebraic methods discussed in Chapter 3.

Chapter 3

Finite interval estimation using multiple regression

Chapter 2 provided the necessary background into the development of the double-sided kernels. This chapter discusses the usage of these double-sided kernels for the joint estimation of both the parameters and the states over a finite interval $[a, b]$. A specific case of 4^{th} order LTI system is considered in this chapter. However, this strategy can be extended to any finite n^{th} order system.

3.1 Kernel representation of a 4^{th} order system [11]

Consider a fourth order homogeneous LTI system described by the characteristic equation (3.1) consisting of four unknown parameters a_0, a_1, a_2, a_3.

$$y^{(4)}(t) + a_3 y^{(3)}(t) + a_2 y^{(2)}(t) + a_1 y^{(1)}(t) + a_0 y(t) = 0, \quad \forall t \in [a, b] \tag{3.1}$$

The general idea of the Theorem 2.2.1 and Theorem 2.2.3 was to reduce the order of the output derivatives in (3.1) until no derivatives appeared. The influence of any pre-existing initial conditions were removed by pre-multiplying the *annihilator functions* that vanished together with their derivatives at the endpoints of $[a, b]$. Thus, we pre-multiply (3.1) by

$(\epsilon - a)^4$ and $(b - \zeta)^4$ (annihilator at $\epsilon = a$ and $\zeta = b$) to get

$$(\varepsilon - a)^4 y^{(4)}(t) + a_3(\varepsilon - a)^4 y^{(3)}(t) + a_2(\varepsilon - a)^4 y^{(2)}(t) + a_1(\varepsilon - a)^4 y^{(1)}(t) + a_0(\varepsilon - a)^4 y(t) = 0 \tag{3.2}$$

$$(b - \zeta)^4 y^{(4)}(t) + a_3(b - \zeta)^4 y^{(3)}(t) + a_2(b - \zeta)^4 y^{(2)}(t) + a_1(b - \zeta)^4 y^{(1)}(t) + a_0(b - \zeta)^4 y(t) = 0 \tag{3.3}$$

The equations (3.2) and (3.3) are integrated four times on the intervals $[a, a+\tau]$ and $[b-\sigma, b]$. It means, the equation (3.1) is integrated the forward direction for the interval $[a, a+\tau]$ and in the backward direction for $[b-\sigma, b]$. The complete derivation is detailed in [15] and the resulting forward and backward kernel is given below.

$$\begin{aligned}
\tau^4 y(a+\tau) = & \int_a^{a+\tau} \left[16\left(\varepsilon''' - a\right)^3 - a_3\left(\varepsilon''' - a\right)^4\right] y\left(\varepsilon'''\right) \mathrm{d}\varepsilon''' \\
& + \int_a^{a+\tau} \int_a^{\varepsilon'''} \left[-72\left(\varepsilon'' - a\right)^2 + 12a_3\left(\varepsilon'' - a\right)^3 - a_2\left(\varepsilon'' - a\right)^4\right] y\left(\varepsilon''\right) \mathrm{d}\varepsilon'' \mathrm{d}\varepsilon''' \\
& + \int_a^{a+\tau} \int_a^{\varepsilon'''} \int_a^{\varepsilon''} \left[96\left(\varepsilon' - a\right) - 36a_3\left(\varepsilon' - a\right)^2 + 8a_2\left(\varepsilon' - a\right)^3 - a_1\left(\varepsilon' - a\right)^4\right] y\left(\varepsilon'\right) \mathrm{d}\varepsilon' \mathrm{d}\varepsilon'' \mathrm{d}\varepsilon''' \\
& + \int_a^{a+\tau} \int_a^{\varepsilon'''} \int_a^{\varepsilon''} \int_a^{\varepsilon'} \left[-24 + 24a_3(\varepsilon - a) - 12a_2(\varepsilon - a)^2 + 4a_1(\varepsilon - a)^3 - a_0(\varepsilon - a)^4\right] y(\varepsilon) \mathrm{d}\varepsilon \mathrm{d}\varepsilon' \mathrm{d}\varepsilon'' \mathrm{d}\varepsilon'''
\end{aligned} \tag{3.4}$$

$$\begin{aligned}
\sigma^4 y(b-\sigma) = & \int_b^{b-\sigma} \left[-16\left(b - \zeta'''\right)^3 - a_3\left(b - \zeta'''\right)^4\right] y\left(\zeta'''\right) \mathrm{d}\zeta''' \\
& + \int_b^{b-\sigma} \int_b^{\zeta'''} \left[-72\left(b - \zeta''\right)^2 - 12a_3\left(b - \zeta''\right)^3 - a_2\left(b - \zeta''\right)^4\right] y\left(\zeta''\right) \zeta'' \zeta''' \\
& + \int_b^{b-\sigma} \int_b^{\zeta'''} \int_b^{\zeta''} \left[-96\left(b - \zeta'\right) - 36a_3\left(b - \zeta'\right)^2 - 8a_2\left(b - \zeta'\right)^3 - a_1\left(b - \zeta'\right)^4\right] y\left(\zeta'\right) \mathrm{d}\zeta' \mathrm{d}\zeta'' \mathrm{d}\zeta''' \\
& + \int_b^{b-\sigma} \int_b^{\zeta'''} \int_b^{\zeta''} \int_b^{\zeta'} \left[-24 - 24a_3(b - \zeta) - 12a_2(b - \zeta)^2 - 4a_1(b - \zeta)^3 - a_0(b - \zeta)^4\right] y(\zeta) \mathrm{d}\zeta \mathrm{d}\zeta' \mathrm{d}\zeta'' \mathrm{d}\zeta'''
\end{aligned} \tag{3.5}$$

Cauchy's formula for repeated integration [16] is applied to equations (3.4) and (3.5) for simplifying the repeated integration.

Let f be a continuous function on the real line, then the n^{th} repeated integral of f at a is given by,

$$f^{(-n)}(t) = \int_a^t \int_a^{\sigma_1} \cdots \int_a^{\sigma_{n-1}} f(\sigma_n)\, \mathrm{d}\sigma_n \cdots \mathrm{d}\sigma_2\, \mathrm{d}\sigma_1 \tag{3.6}$$

which is equivalent to a single integration.

$$f^{(-n)}(t) = \frac{1}{(n-1)!} \int_a^t (t-s)^{n-1} f(s) \mathrm{d}s \tag{3.7}$$

Assuming $a + \tau = t$ in (3.4), $b - \sigma = t$ in (3.5) and by applying Cauchy's formula in the forward direction for the interval $[a, a+\tau]$ and backward direction for the interval $[b, b-\sigma]$ we arrive at,

$$\alpha_a(t) y(t) \triangleq \int_a^t K_{F,y}(t,s) y(s) ds; \qquad \text{where } \alpha_a(t) = (t-a)^4 \tag{3.8}$$

$$\alpha_b(t) y(t) \triangleq \int_t^b K_{B,y}(t,s) y(s) ds; \qquad \text{where } \alpha_b(t) = (b-t)^4 \tag{3.9}$$

with $K_{F,y}(t,s)$ given by,

$$\begin{aligned} K_{F,y}(t,s) = &\left[16(s-a)^3 - a_3(s-a)^4\right] + \frac{(t-s)^1}{1!}\left[-72(s-a)^2 + 12a_3(s-a)^3 - a_2(s-a)^4\right] \\ &+ \frac{(t-s)^2}{2!}\left[96(s-a) - 36a_3(s-a)^2 + 8a_2(s-a)^3 - a_1(s-a)^4\right] \\ &+ \frac{(t-s)^3}{3!}\left[-24 + 24a_3(s-a) - 12a_2(s-a)^2 + 4a_1(s-a)^3 - a_0(s-a)^4\right] \end{aligned} \tag{3.10}$$

and $K_{B,y}(t,s)$ as,

$$\begin{aligned}K_{B,y}(t,s) = &\left[16(b-s)^3 + a_3(b-s)^4\right] + \frac{(t-s)}{1!}\left[72(b-s)^2 + 12a_3(b-s)^3 + a_2(b-s)^4\right] \\ &+ \frac{(t-s)^2}{2!}\left[96(b-s) + 36a_3(b-s)^2 + 8a_2(b-s)^3 + a_1(b-s)^4\right] \\ &+ \frac{(t-s)^3}{3!}\left[24 + 24a_3(b-s) + 12a_2(b-s)^2 + 4a_1(b-s)^3 + a_0(b-s)^4\right]\end{aligned} \tag{3.11}$$

3.2 Multiple regression equations for 4$^{\text{th}}$ order system [11]

By integrating (3.4) and (3.5) multiple times using the Cauchy's formula for repeated integration[16], we arrive at the multiple regression equations. The complete derivation for the forward kernel is shown below. Similar derivation can be carried out for the backward kernel. This derivation is a taken from [11].

By simultaneously integrating both sides of (3.4) over the interval $[a,t]$ and applying Cauchy formula for repeated integration,

$$\begin{aligned}\int_a^t (s-a)^4 y(s)ds = &\int_a^t \int_a^s \left[16\left(\varepsilon''' - a\right)^3 - a_3\left(\varepsilon''' - a\right)^4\right] y\left(\varepsilon'''\right) \mathrm{d}\varepsilon''' ds \\ &+ \int_a^t \int_a^s \int_a^{\varepsilon'''} \left[-72\left(\varepsilon'' - a\right)^2 + 12a_3\left(\varepsilon'' - a\right)^3 - a_2\left(\varepsilon'' - a\right)^4\right] y\left(\varepsilon''\right) \mathrm{d}\varepsilon'' \mathrm{d}\varepsilon''' ds \\ &+ \int_a^t \int_a^s \int_a^{\varepsilon'''} \int_a^{\varepsilon''} \left[96\left(\varepsilon' - a\right) - 36a_3\left(\varepsilon' - a\right)^2 + 8a_2\left(\varepsilon' - a\right)^3 - a_1\left(\varepsilon' - a\right)^4\right] y\left(\varepsilon'\right) \mathrm{d}\varepsilon' \mathrm{d}\varepsilon'' \mathrm{d}\varepsilon''' ds \\ &+ \int_a^t \int_a^s \int_a^{\prime\prime\prime} \int_a^{\prime\prime} \int_a^{\varepsilon'} \left[-24 + 24a_3(\varepsilon - a) - 12a_2(\varepsilon - a)^2 + 4a_1(\varepsilon - a)^3 - a_0(\varepsilon - a)^4\right] y(\varepsilon) \mathrm{d}\varepsilon \mathrm{d}\varepsilon' \mathrm{d}\varepsilon'' \mathrm{d}\varepsilon''' ds\end{aligned} \tag{3.12}$$

$$
\begin{aligned}
&= \int_a^t \frac{(t-s)^1}{1!}\left[16(s-a)^3 - a_3(s-a)^4\right] y(s)ds \\
&+ \int_a^t \frac{(t-s)^2}{2!}\left[-72(s-a)^2 + 12a_3(s-a)^3 - a_2(s-a)^4\right] y(s)ds \\
&+ \int_a^t \frac{(t-s)^3}{3!}\left[96(s-a) - 36a_3(s-a)^2 + 8a_2(s-a)^3 - a_1(s-a)^4\right] y(s)ds \\
&+ \int_a^t \frac{(t-s)^4}{4!}\left[-24 + 24a_3(s-a) - 12a_2(s-a)^2 + 4a_1(s-a)^3 - a_0(s-a)^4\right] y(s)ds
\end{aligned} \tag{3.13}
$$

Similarly, integrating both sides of (3.4) twice over the interval $[a, t]$ gives,

$$
\begin{aligned}
&\int_a^t (s-a)^4(t-s)y(s)ds = \int_a^t \frac{(t-s)^2}{2!}\left[16(s-a)^3 - a_3(s-a)^4\right] y(s)ds \\
&+ \int_a^t \frac{(t-s)^3}{3!}\left[-72(s-a)^2 + 12a_3(s-a)^3 - a_2(s-a)^4\right] y(s)ds \\
&+ \int_a^t \frac{(t-s)^4}{4!}\left[96(s-a) - 36a_3(s-a)^2 + 8a_2(s-a)^3 - a_1(s-a)^4\right] y(s)ds \\
&+ \int_a^t \frac{(t-s)^5}{5!}\left[-24 + 24a_3(s-a) - 12a_2(s-a)^2 + 4a_1(s-a)^3 - a_0(s-a)^4\right] y(s)ds
\end{aligned} \tag{3.14}
$$

By further integrating (3.4) multiple times in a similar way as above yields the following formula of the forward kernel of a 4^{th} order system for k^{th} order of integration

$$
\begin{aligned}
&\frac{1}{(k-1)!}\int_a^t \alpha_a(t,s)(t-s)^{k-1}y(s)ds \\
&= \int_a^t \frac{(t-s)^k}{k!}\Big[16(s-a)^3 - a_3(s-a)^4\Big] y(s)ds \\
&+ \int_a^t \frac{(t-s)^{k+1}}{(k+1)!}\Big[-72(s-a)^2 + 12a_3(s-a)^3 - a_2(s-a)^4\Big] y(s)ds \\
&+ \int_a^t \frac{(t-s)^{k+2}}{(k+2)!}\Big[96(s-a) - 36a_3(s-a)^2 + 8a_2(s-a)^3 - a_1(s-a)^4\Big] y(s)ds \\
&+ \int_a^t \frac{(t-s)^{k+3}}{(k+3)!}\Big[-24 + 24a_3(s-a) - 12a_2(s-a)^2 + 4a_1(s-a)^3 - a_0(s-a)^4\Big] y(s)ds
\end{aligned} \tag{3.15}
$$

Equation (3.15) can be rewritten as

$$\frac{1}{(k-1)!}\int_a^t \alpha_a(t,s)(t-s)^{k-1}y(s)ds = \int_a^t K_{F_k,y}(t,s)y(s)ds \quad \text{for } k=1,..,m \tag{3.16}$$

By similar fashion, integrating both sides of (3.5) simultaneously k times over the interval $[t,b]$ and applying Cauchy formula for repeated integration, we get

$$\begin{aligned}
&\frac{1}{(k-1)!}\int_t^b \alpha_b(t,s)(t-s)^{k-1}y(s)ds \\
&= -\int_t^b \frac{(t-s)^k}{k!}\Big[16(b-s)^3 + a_3(b-s)^4\Big]y(s)ds \\
&\quad -\int_t^b \frac{(t-s)^{k+1}}{k+1!}\Big[72(b-s)^2 + 12a_3(b-s)^3 + a_2(b-s)^4\Big]y(s)ds \\
&\quad -\int_t^b \frac{(t-s)^{k+2}}{k+2!}\Big[96(b-s) + 36a_3(b-s)^2 + 8a_2(b-s)^3 + a_1(b-s)^4\Big]y(s)ds \\
&\quad -\int_t^b \frac{(t-s)^{k+3}}{k+3!}\Big[24 + 24a_3(b-s) + 12a_2(b-s)^2 + 4a_1(b-s)^3 + a_0(b-s)^4\Big]y(s)ds
\end{aligned} \tag{3.17}$$

And (3.17) can be rewritten as

$$\frac{1}{(k-1)!}\int_t^b \alpha_b(t,s)(t-s)^{k-1}y(s)ds = \int_t^b K_{B_k,y}(t,s)y(s)ds \quad \text{for } k=1,..,m \tag{3.18}$$

Adding (3.16) and (3.18) gives,

$$\frac{1}{(k-1)!}\int_a^b \alpha_{ab}(t,s)(t-s)^{k-1}y(s)ds = \int_a^b K_{DS_k,y}(t,s)y(s)ds \quad \text{for } k=1,..,m \tag{3.19}$$

where,

$$K_{DS_k,y}(t,s) \triangleq \begin{cases} K_{F_k,y}(t,s) : s \le t \\ K_{B_k,y}(t,s) : s > t \end{cases} ; \alpha_{ab}(t,s) \triangleq \begin{cases} (t-a)^4 : s \le t \\ (b-t)^4 : s > t \end{cases} \tag{3.20}$$

3.3 Parameter estimation using multiple regression for n$^{\text{th}}$ order system [11]

The multiple regression equation for a 4^{th} order system can be generalized for n^{th} order systems, only difference being the kernel definitions from Theorem 2.2.1. Therefore, we can write the multiple regression equations for $k = 1, ..., m(m \geq n)$ in terms of component kernels as follows

$$\int_a^b \alpha_{ab}(s)y(s)ds = \sum_{i=0}^{n} \beta_i \int_a^b K_{DS_1(i),y}(t,s)y(s)ds \tag{3.21}$$

$$\int_a^b \alpha_{ab}(s)(t-s)y(s)ds = \sum_{i=0}^{n} \beta_i \int_a^b K_{DS_2(i),y}(t,s)y(s)ds \tag{3.22}$$

$$\frac{1}{2}\int_a^b \alpha_{ab}(s)(t-s)^2y(s)ds = \sum_{i=0}^{n} \beta_i \int_a^b K_{DS_3(i),y}(t,s)y(s)ds \tag{3.23}$$

$$\vdots$$

$$\frac{1}{(m-1)!}\int_a^b \alpha_{ab}(s)(t-s)^{m-1}y(s)ds = \sum_{i=0}^{n} \beta_i \int_a^b K_{DS_m(i),y}(t,s)y(s)ds \tag{3.24}$$

In a noise-free deterministic setting, the output variable y becomes the measured output coinciding with the nominal output trajectory y_T. With $\bar{a} := [a_0; \cdots ; a_{n-1}]$ and $\beta := [a_0; \cdots ; a_{n-1}; a_n] = [\bar{a}; 1]$, let $K_{DS_k(\bar{a})}(t, y_T)$ be row vectors with integral components

$$K_{DS_k(\bar{a})}(t, y_T) := \left[\int_a^b K_{DS_k(0),y}(t,s)y_T(s)ds, \cdots , \int_a^b K_{DS_k(n-1),y}(t,s)y_T(s)ds\right] \tag{3.25}$$

and $K_{DS_k(a_n)}(t, y_T)$ be scalars

$$K_{DS_k(a_n)}(t, y_T) := \int_a^b K_{DS_k(n),y}(t,s)y_T(s)ds \tag{3.26}$$

corresponding to $\beta_n := a_n = 1$.

Rearranging (3.21) - (3.24) for $k = 1, \ldots, m (m \geq n)$

$$\begin{bmatrix} \int_a^b \alpha_{ab}(s) y(s) ds - K_{DS_1(a_n)}(t, y_T) \\ \int_a^b \alpha_{ab}(s)(t-s) y(s) ds - K_{DS_2(a_n)}(t, y_T) \\ \vdots \\ \frac{1}{(m-1)!} \int_a^b \alpha_{ab}(s)(t-s)^{m-1} y(s) ds - K_{DS_m(a_n)}(t, y_T) \end{bmatrix} = \begin{bmatrix} K_{DS_1(\bar{a})}(t, y_T) \\ K_{DS_2(\bar{a})}(t, y_T) \\ \vdots \\ K_{DS_m(\bar{a})}(t, y_T) \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix} \tag{3.27}$$

Using the distinct time instants $t_j = t_1, \ldots, t_N \in (a, b]$, known as *knots*, we define the following

$$q^k(t_j, y_T) = \frac{1}{(k-1)!} \int_a^b \alpha_{ab}(t_j, s) (t_j - s)^{k-1} y_T(s) ds - K_{DS_k(a_n)}(t_j, y_T); \tag{3.28}$$

$$\begin{aligned} p^k(t_j, y_T) = K_{DS_k(\bar{a})}(t_j, y_T) &:= \begin{bmatrix} \int_a^b K_{DS_k(0),y}(t_j, s) y_T(s) ds & \cdots & \int_a^b K_{DS_k(n-1),y}(t_j, s) y_T(s) ds \end{bmatrix} \\ &= \begin{bmatrix} p_0^k(t_j, y_T) & \cdots & p_{n-1}^k(t_j, y_T) \end{bmatrix} \end{aligned} \tag{3.29}$$

Thus, the equation (3.27) can be re-written in point-wise for discrete values $k = 1, \ldots, m$ in the form of another matrix equation as shown

$$\begin{bmatrix} q^1(t_1, y_T) \\ \vdots \\ q^1(t_N, y_T) \\ \vdots \\ q^m(t_1, y_T) \\ \vdots \\ q^m(t_N, y_T) \end{bmatrix}_{Nm \times 1} = \begin{bmatrix} p_0^1(t_1, y_T) & p_1^1(t_1, y_T) & \cdots & p_{n-1}^1(t_1, y_T) \\ & & \ddots & \\ p_0^1(t_N, y_T) & p_1^1(t_N, y_T) & \cdots & p_{n-1}^1(t_N, y_T) \\ & & \vdots & \\ p_0^m(t_1, y_T) & p_1^m(t_1, y_T) & \cdots & p_{n-1}^m(t_1, y_T) \\ & & \ddots & \\ p_0^m(t_N, y_T) & p_1^m(t_N, y_T) & \cdots & p_{n-1}^m(t_N, y_T) \end{bmatrix}_{Nm \times n} \begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix}_{n \times 1} \tag{3.30}$$

which can be represent in a concise way as follows

$$Q(y_T) = P(y_T) \bar{a} \tag{3.31}$$

where $Q(y_T) \in \mathbb{R}^{Nk}, P(y_T) \in \mathbb{R}^{Nk} \times \mathbb{R}^n, \bar{a} \in \mathbb{R}^n$ and $k = 1, \ldots, m$. Thus, the matrix equation (3.30) can be solved exactly using least squares error minimization with respect to the parameter vector $\bar{a}$ provided adequate identifiability assumptions are met and the output is measured without error.

Identifiability of homogeneous LTI systems from a single realization of a measured output [17] [11]

A homogeneous LTI system such as

$$\dot{x}(t) = Ax(t); \quad y = Cx; \quad x \in \mathbb{R}^n; \quad x(0) = b \tag{3.32}$$

is identifiable from a single noise-free realization of its output trajectory y on the interval $[0, \infty)$. The identifiability condition is stated in its equivalent form:

Definition: System (3.32) *is globally identifiable from b if and only if the functional mapping* $b \mapsto y(\cdot; A, b)$ *is injective on* $\mathbb{R}^n$ *where* $y(\cdot; A, b)$ *denotes the output orbit of* (3.32) *through b.*

Theorem 3.3.1 *[18] System* (3.32) *is globally identifiable from b if and only if the evolution of the output orbit of* (3.32) *is not confined to a proper subspace of* $\mathbb{R}^n$.

The above criterion has limited use for reasons of practicality: it is difficult to verify computationally, pertains to infinite time horizons $[0, \infty)$ and, most importantly, requires the output trajectory to be known exactly. For the purpose of the present exposition it hence suffices to invoke a *practical version of identifiability* as defined below.

Definition 2: Practical linear identifiability
The homogeneous system (3.32) is practically linearly identifiable on $[a, b]$ with respect to a particular noisy discrete realization of the output measurement process, $y_M(t), t \in [a, b]$, if and only if there exist distinct knots $t_1, \cdots, t_N \in (a, b]$ which render rank of $P(y) = n$. Any such output realization is then called *persistent*.

In practical applications the N distinct time instants needed can be placed equidistantly over the interval $(a, b]$ or else generated randomly. Since no assumptions are made about system perturbations or measurement noise, the estimation equation (3.31) is solved in terms of a pseudo-inverse P^+ of P:

$$\overline{a} = P^{+}(y_M)Q(y_M) \tag{3.33}$$

It is worth noting that, parameter estimation can be conducted simultaneously with state estimation. Under the assumption of system flatness, the system states are immediately recovered as functions of the time derivatives of the output. Following parametric estimation, the output derivatives can be computed using the recursive kernels in Theorem (2.2.3).

3.4 Calculation of the error covariance matrix [17] [11]

In the Section 3.3, the regression equations were derived under the assumption that the measured signal is free from external disturbances (noise). In the presence of measurement noise, the reproducing property fails to hold along an inexact output trajectory. To address this issue, it is assumed that the measured signal is ingrained with additive white Gaussian noise (AWGN), resulting in a stochastic regression problem. The stochastic output measurement process, $y_M(t)$ adapted to the natural filtration of the standard Wiener process W on $[a, b]$ is

$$y_M(t, \omega) = y_T(t) + \sigma\dot{W}(t, \omega) \; ; \quad t \in [a, b] \tag{3.34}$$

where $\sigma\dot{W}$ represents the generalized derivative of the standard Wiener process; see [19] i.e. $\sigma\dot{W}$ is identified with the white noise process having a constant variance σ^2 and where y_T is the true system output.

The generalized expectation and covariance functions of white noise are given by:

$$E[\dot{W}(t)] = 0 \tag{3.35}$$

$$Cov[\dot{W}(t)\dot{W}(s)] = E[\dot{W}(t)\dot{W}(s)] = \delta(t - s) \tag{3.36}$$

$$Var[\dot{W}(t)] = E(\dot{W}(t))^2 = 1 \quad t, s \in [a, b] \tag{3.37}$$

where δ is the Dirac delta distribution but acting on a square integrable functions as an evaluation functional:

$$\int_a^b g(s)\delta(t - s)ds = g(t) \tag{3.38}$$

Using the noisy realization of the signal(3.34), the kernel expression for $k = 1, \ldots, m$ is given by

$$\int_a^b K_{DS_k,y}(t,s) y_M(s) ds = \int_a^b K_{DS_k,y}(t,s) y_T(s)\ ds + \int_a^b K_{DS_k,y}(t,s) \sigma \dot{W}(s)\ ds \tag{3.39}$$

The stochastic regression equation is given by

$$\frac{1}{(k-1)!} \int_a^b \alpha_{ab}(t,s)(t-s)^{k-1} y_M(s) ds = \int_a^b K_{DS_k,y}(t,s) y_M(s)\ ds + e(t) \tag{3.40}$$

which has the random regressor vector

$$\left[\int_a^b K_{DS_k(a_0),y}(t,s) y_M(s) ds, \cdots, \int_a^b K_{DS_k(a_n),y}(t,s) y_M(s) ds \right]^T \tag{3.41}$$

The assumptions of the Gauss-Markov Theorem are violated in the linear regression problem (3.40) because the random regressor is correlated with a regression error, which additionally fails to be homoskedastic i.e. the vector of random variables don't have the same finite variance. The above regression is thus a typical 'error-in-the-variable' problem with heteroskedastic noise which has been tackled using the instrumental variable (IV) approach adopted in [20]. The IV approach resulted in minor improvement in the results but at the cost of high computational burden. Hence, the multiple regression equation approach was considered to be the better choice.

The standard way to deal with unknown heteroskedasticity is to employ Generalized Least Squares (GLS) approach, which is considered to be the BLUE (Best Linear Unbiased Estimator). The GLS utilizes inverse covariance weighting in the regression error minimization problem. Let $Q(y_M)$ and $P(y_M)$ be the matrices corresponding to N samples of the measurement process realization y_M at a batch of knots $t_1, t_2, ..., t_N$. The matrix regression equation (3.31) can be adapted for $k = 1, \ldots, m$ as

$$Q(y_M) = P(y_M)\overline{a} + e \tag{3.42}$$

where

$$e := \begin{bmatrix} e^1(t_1) \\ \vdots \\ e^1(t_N) \\ \vdots \\ e^m(t_1) \\ \vdots \\ e^m(t_N) \end{bmatrix}_{Nm\times 1} \tag{3.43}$$

with

$$e^k(t_j) := \frac{\sigma}{(k-1)!}\int_a^b \alpha_{ab}(t_j,s)(t_j-s)^{k-1}\dot{W}(s)ds - \sigma\int_a^b K_{DS_k,y}(t_j,s)\dot{W}(s)ds; k=1,\ldots,m \tag{3.44}$$

The regression error minimization problem associated with (3.42) is solved using a Regularized Least Squares (RLS). The standard regression error minimization of the parameter vector $\overline{a}$ is

$$\min_{\bar{a}}\left((\bar{a}-\bar{a}_0)^T W_0^{-1}(\bar{a}-\bar{a}_0) + (Q(y_M)-P(y_M)\bar{a})^T S(Q(y_M)-P(y_M)\bar{a})\right) \tag{3.45}$$

where W_0 is a given penalty matrix, which is a positive-definite matrix (initial guess), $\overline{a_0}$ is a given parameter vector (also an initial guess) and $S \in \mathbb{R}^{N_k\times N_k}$ is the weighing matrix defined as $S := \text{diag}(S_1,...,S_k)$ for $k = 1,\ldots,m$ and $S_k \in \mathbb{R}^{N\times N}$ are the inverses of the corresponding error covariance matrices, as defined below:

$$[S_k]^{-1} := \begin{bmatrix} \text{Cov}[e^k(t_1),e^k(t_1)]\cdots\text{Cov}[e^k(t_1),e^k(t_N)] \\ \ddots \\ \text{Cov}[e^k(t_N),e^k(t_1)]\cdots\text{Cov}[e^k(t_N),e^k(t_N)] \end{bmatrix}; \quad k=1,\ldots,m \tag{3.46}$$

As the kernel functions are Hilbert-Schmidt and hence are square integrable, using the properties (3.35) - (3.37) of white noise, the covariance matrix calculation for $Cov[e^k(t_i),e^k(t_j)]$

for the general n^{th} order system with $k = 1, \ldots, m(m \geq n)$ is given by

$$\begin{aligned}
&\mathrm{Cov}[e^k(t_i), e^k(t_j)] = E[e^k(t_i)e^k(t_j)] \\
&= \sigma^2 E\Big[\Big[\int_a^b \frac{1}{(k-1)!}\alpha_{ab}(\tau)(t_i-\tau)^{k-1}\dot{W}(\tau)d\tau - \int_a^b K_{DS_k,y}(t_i,\tau)\dot{W}(\tau)d\tau\Big] \\
&\qquad \Big[\int_a^b \frac{1}{(k-1)!}\alpha_{ab}(s)(t_j-s)^{k-1}\dot{W}(s)ds - \int_a^b K_{DS_k,y}(t_j,s)\dot{W}(s)ds\Big]\Big] \\
&= \sigma^2 E\Big[\frac{1}{((k-1)!)^2}\int_a^b\int_a^b \alpha_{ab}(\tau)\alpha_{ab}(s)(t_i-\tau)^{k-1}(t_j-s)^{k-1}\dot{W}(\tau)\dot{W}(s)d\tau ds\Big] \\
&\qquad - E\Big[\frac{1}{(k-1)!}\int_a^b\int_a^b \alpha_{ab}(\tau)(t_i-\tau)^{k-1}\dot{W}(\tau)K_{DS_k,y}(t_j,s)\dot{W}(s)d\tau ds\Big] \\
&\qquad - E\Big[\frac{1}{(k-1)!}\int_a^b\int_a^b \alpha_{ab}(s)(t_j-s)^{k-1}\dot{W}(s)K_{DS_k,y}(t_i,\tau)\dot{W}(\tau)ds d\tau\Big] \\
&\qquad + E\Big[\int_a^b\int_a^b K_{DS_k,y}(t_i,\tau)K_{DS_k,y}(t_j,s)\dot{W}(\tau)\dot{W}(s)d\tau ds\Big] \\
&= \frac{\sigma^2}{((k-1)!)^2}\int_a^b\int_a^b \alpha_{ab}(\tau)\alpha_{ab}(s)(t_i-\tau)^{k-1}(t_j-s)^{k-1}E\Big[\dot{W}(\tau)\dot{W}(s)\Big]d\tau ds \\
&\qquad - \frac{\sigma^2}{(k-1)!}\int_a^b\int_a^b \alpha_{ab}(\tau)(t_i-\tau)^{k-1}K_{DS_k,y}(t_j,s)E\Big[\dot{W}(\tau)\dot{W}(s)\Big]d\tau ds \\
&\qquad - \frac{\sigma^2}{(k-1)!}\int_a^b\int_a^b \alpha_{ab}(s)(t_j-s)^{k-1}K_{DS_k,y}(t_i,\tau)E\Big[\dot{W}(s)\dot{W}(\tau)\Big]ds d\tau \\
&\qquad + \sigma^2\int_a^b\int_a^b K_{DS_k,y}(t_i,\tau)K_{DS_k,y}(t_j,s)E\Big[\dot{W}(\tau)\dot{W}(s)\Big]d\tau ds
\end{aligned}$$

$$\begin{aligned}
&= \frac{\sigma^2}{((k-1)!)^2}\int_a^b \alpha_{ab}(s)\alpha_{ab}(s)(t_i-s)^{k-1}(t_j-s)^{k-1}ds \\
&\qquad - \frac{\sigma^2}{(k-1)!}\int_a^b \alpha_{ab}(s)(t_i-s)^{k-1}K_{DS_k,y}(t_j,s)ds \\
&\qquad - \frac{\sigma^2}{(k-1)!}\int_a^b \alpha_{ab}(s)(t_j-s)^{k-1}K_{DS_k,y}(t_i,s)ds + \sigma^2\int_a^b K_{DS_k,y}(t_i,s)K_{DS_k,y}(t_j,s)ds
\end{aligned}$$

3.5 Modified Regularized Least Squares [21] [11]

The covariance matrix derived in section 3.4 depends on the unknown variance σ^2 and the unknown parameter vector $\bar{a}$ in $K_{DS_k,y}$ kernels. Hence we use a modified version of the least

squares algorithm in which the covariance matrix is estimated progressively as more data about the multiple regression residuals are available. This is performed in a recursively where the consecutive batches of samples are acquired from the realization of y_M. The quadratic cost function of the unknown parameter vector $\bar{a}$ is given by

$$J(\bar{a}) = (\bar{a} - \bar{a}_0)^T W_0^{-1} (\bar{a} - \bar{a}_0) + \|Q - P\bar{a}\|_S^2 \tag{3.47}$$

The major advantage of using (3.47) is that it ensures a unique solution to the least squares problem, even when the matrix P is not full rank. When P is full rank, including $(\bar{a}-\bar{a}_0)^T W_0^{-1}(\bar{a}-\bar{a}_0)$ can improve the condition number of the matrix resulting in better numerical behaviour.

The solution to (3.47) is of the form

$$\bar{a} = (W_0^{-1} + P^T S P)^{-1} P^T S Q \tag{3.48}$$

Ideally, equation (3.48) is most suitable when all the measurements are available beforehand. Unfortunately, in practice this becomes computationally expensive and even unfeasible as the measurements are obtained sequentially and we need to update our estimate $\bar{a}$ with every new batch of measurement. Hence, we adapt the recursive form of least squares to address this issue.

Assuming that $\bar{a_0}$ is 0 (for simplicity), at iteration j, the minimization function is given by

$$\min_{\bar{a}} \left[\bar{a}^T W_0^{-1} \bar{a} + \|\bar{Q}_j - \bar{P}_j \bar{a}\|_{S_j}^2 \right] \tag{3.49}$$

where for $k = 1, \ldots, m$

$$\bar{Q}_j = \begin{bmatrix} Q_0 \\ Q_1 \\ \vdots \\ Q_j \end{bmatrix}; \quad \bar{P}_j = \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_j \end{bmatrix}; \quad \text{with } Q_j = \begin{bmatrix} q_j^1(y_M) \\ \vdots \\ q_j^k(y_M) \end{bmatrix} \text{ and } P_j = \begin{bmatrix} p_j^1(y_M) \\ \vdots \\ p_j^k(y_M) \end{bmatrix} \tag{3.50}$$

and

$$\bar{S}_j = \text{diag}(S_0, S_1, \ldots, S_j); \quad \text{with } S_j = \text{diag}(S_{1_j}, \ldots, S_{k_j}) \tag{3.51}$$

The detailed derivation for the resulting RLS solution can be referred in [21] and [22].

3.5.1 Recursive Regularized Least Squares Algorithm (RRLS)

- Initialize the estimator:

$$\bar{a}_0 = 0$$
$$W_0 = \delta I$$

  In case of no prior knowledge about parameters, simply let $W_0 \approx \infty I$. In the case of perfect prior knowledge, $W_0 = 0$.

- Iterate the following two steps.
  (a) Obtain a new batch of knot points (measurements) and calculate the Q_j, P_j and S_j matrices
  (b) Update the estimate $\hat{a}$ and the covariance of the estimation error as per the following equations

$$K_j = W_{j-1}P_j^T(P_jW_{j-1}P_j^T + S_j^{-1})^{-1} \tag{3.52}$$
$$W_j = (I - K_jP_j)W_{j-1} \tag{3.53}$$
$$\hat{a}_j = \hat{a}_{j-1} + K_j(Q_j - P_j\hat{a}_{j-1}) \tag{3.54}$$

  The initial estimate of S_j^{-1} is calculated as the *empirical* variance of $(y_M - y_E)$ where y_E is the estimated output corresponding to the parameter values obtained in iteration $j = 0$. This is updated at each consecutive iteration by the same empirical method until the difference in parameter values converges below a set threshold value.

3.6 Reconstruction of Output trajectory and its derivatives [13]

Once the parameters of the system are estimated, the system output can be reconstructed as follows:
From Theorem 2.2.1

$$y = \alpha_{ab}^{-1}(t,n)\int_a^b K_{DS,y}(n,t,\tau)y(\tau)\,d\tau \tag{3.55}$$

However, the system output can be reconstructed from a noisy measurement more precisely by orthogonal projection onto the finite dimensional subspace of RKHS spanned by the fundamental solutions of the characteristic equation of the system 2.1, here denoted by $\xi_1,\ldots,\xi_n$. Every solution of the characteristic equation along with the estimated parameter vector $\hat{a}$ satisfies the reproducing property of Theorem 1. So, the projection on to the space of fundamental solutions will be the noise free trajectory of the system.

The fundamental solutions for the characteristic equation of the system (2.6) are obtained by integrating characteristic equation of the system using n independent vectors as initial conditions for the homogeneous LTI system. The n-independent vectors are considered to be the canonical basis vectors in $\mathbb{R}^n$.

$$\begin{aligned} e_1 &= [1,0,\ldots,0] \\ e_2 &= [0,1,\ldots,0] \\ &\vdots \\ e_n &= [0,0,\ldots,1] \end{aligned} \tag{3.56}$$

After integrating the characteristic equation with sets of n-initial conditions, the fundamental solutions set $\xi_k, k=1,\ldots,n$ is orthonormalized into $\zeta_k, k=1,\ldots,n$, using Gram-Schmidt ortho-normalization procedure in L^2 for computational efficiency. Since, the ortho-normalizing transformation is linear we can write

$$\begin{aligned} &\text{span}\{\xi_k, k=1,\ldots,n\} = \text{span}\{\zeta_k, k=1,\ldots,n\} \\ &\langle\zeta_i|\zeta_j\rangle = 0,\ i\neq j\ ;\ \langle\zeta_i|\zeta_i\rangle_2 = 1 \end{aligned} \tag{3.57}$$

where $\langle\cdot\mid\cdot\rangle_2$ is the inner product in L^2. The estimated noise free ouput is a linear combination

of the transformed fundamental solutions ζ_k.

$$y_T = \sum_{i=1}^{n} c_i \zeta_i \ ; \ \text{with } c_i \ = \ \langle y_T | \zeta_i \rangle_2, \quad i = 1, \ldots, n \tag{3.58}$$

Similarly, we can write an expression for the estimator $\hat{y}_T$ with linear estimators $\hat{c}_i$ for the coefficients c_i in the form

$$\hat{c}_i := \langle y_M | \zeta_i \rangle_2 \ = \ \int_a^b y_M(\tau) \zeta_i(\tau) \, d\tau, \quad i = 1, 2, \ldots, n \tag{3.59}$$

Hence, given a measurement process realization y_M in the finite interval $[a,b]$, the output trajectory can be reconstructed using

$$y_E(t) = \sum_{i=1}^{n} \langle y_M | \zeta_i \rangle_2 \zeta_i(t) \ ; \ t \in [a, b] \tag{3.60}$$

3.6.1 Reconstruction of output derivatives

From Theorem 2.2, the estimated output $y_E(t)$ can be used to reconstruct the derivatives using the following equation.

$$y^{(i)}(t) = \int_a^b K_{DS}^i(t, \tau) y_E(\tau) \, d\tau \quad i = 1, \ldots, n-1 \tag{3.61}$$

where $K_{DS}^i, i = 1, \ldots, n-1$ are the kernel representation for the derivatives.

3.7 Results

3.7.1 Example: Fourth Order System

Consider a fourth order Single Input Single Output (SISO) LTI system as given below.

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & -5 & -5 & 0 \end{bmatrix} x \quad ; y = x_1 \quad ; \quad x(0) = [0, 0, 0, 1] \tag{3.62}$$

with its corresponding characteristic equation

$$y^{(4)}(t) + 0y^{(3)}(t) + 5y^{(2)}(t) + 5y^{(1)}(t) + 1y(t) = 0 \tag{3.63}$$

The poles of this system are $0.4562 \pm 2.334i$, -0.633, -0.279. This makes the system unstable. A case study of the analysis and the parameter estimation for this system is presented, followed by the reconstruction of the state and its derivatives. The performance of the RRLS algorithm for various noise levels is discussed below.

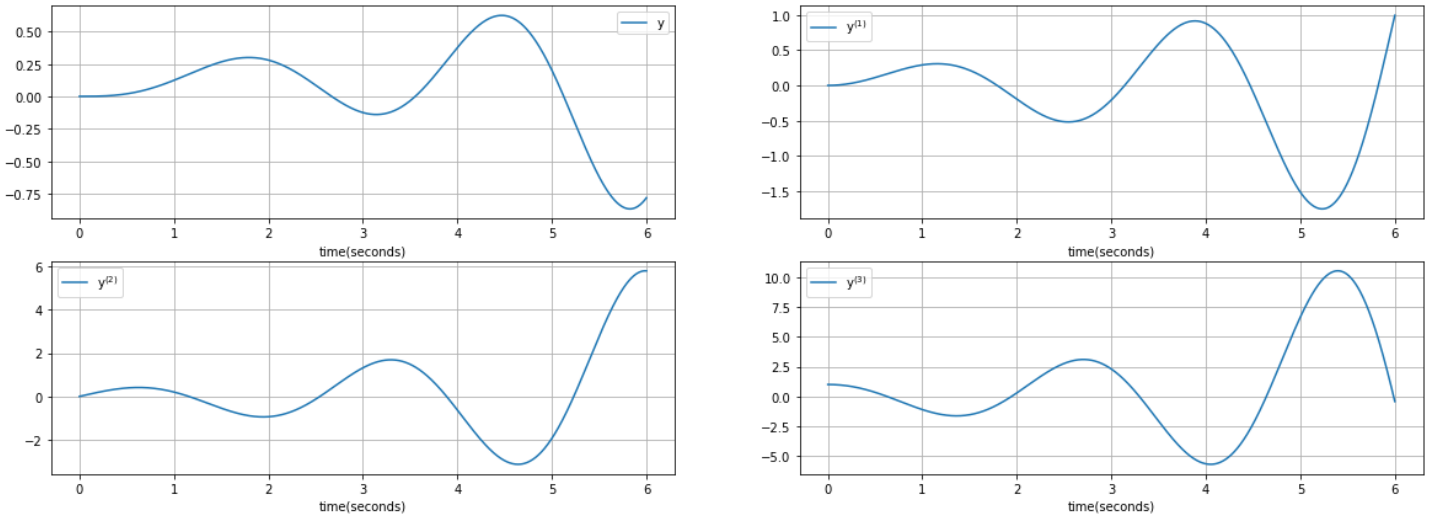

Fig. 3.1 Response of the System (3.63)

In order to validate the robustness of the RRLS algorithm, Additive White Gaussian Noise (AWGN) is superimposed on the signal to simulate real-world noisy signals obtained from measuring devices such as sensors.

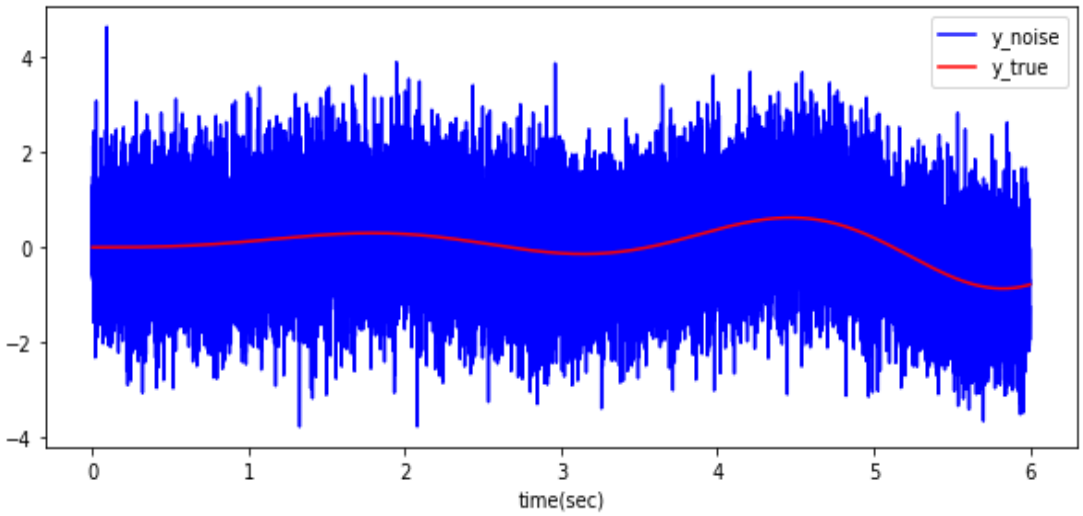

Fig. 3.2 Comparing true signal and measured signal (SNR of -9.5dB) of (3.63)

The following table summarizes the estimated parameter values for different noise levels.

STD (σ)	**SNR (dB)**	a_0	a_1	a_2	a_3	**RMSE**	**MAD**	**MAE**
-	-	1	5	5	0	-	-	-
0	0.00	1.00	4.99	5.00	0	0.00	0.00	0.00
0.5	-4.75	1.02	4.42	5.02	0.08	0.004	0.013	0.003
1	-9.5	0.62	5.17	5.05	0.05	0.016	0.042	0.014
3	-18.45	0.14	4.44	5.21	-0.12	0.025	0.082	0.022

Table 3.1: Estimated parameter values of (3.63)

3.7.2 Output and derivatives reconstruction:

The reconstructed output signal and their derivatives are compared with the true signal.

Noise level $\sigma = 0$, SNR = 0dB

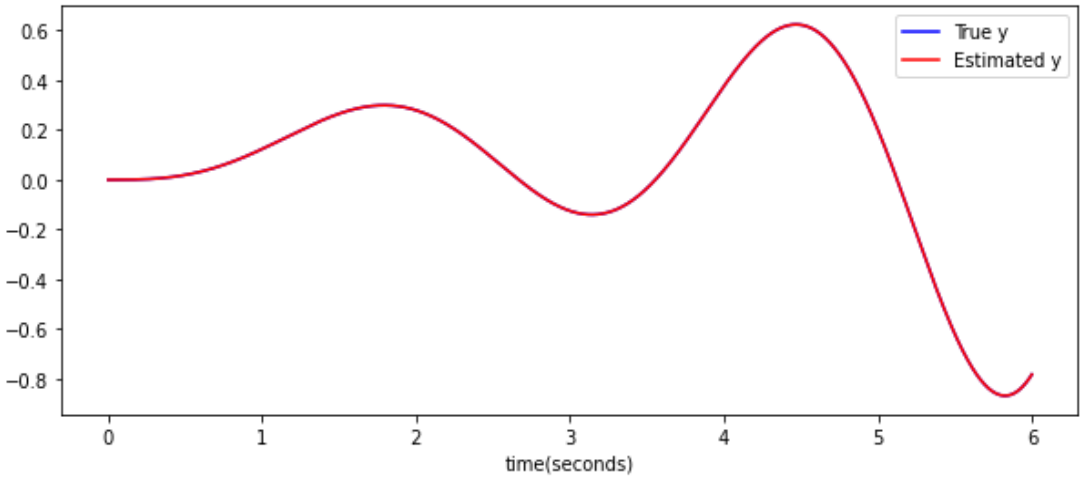

Fig. 3.3 Estimated Output $y(t)$

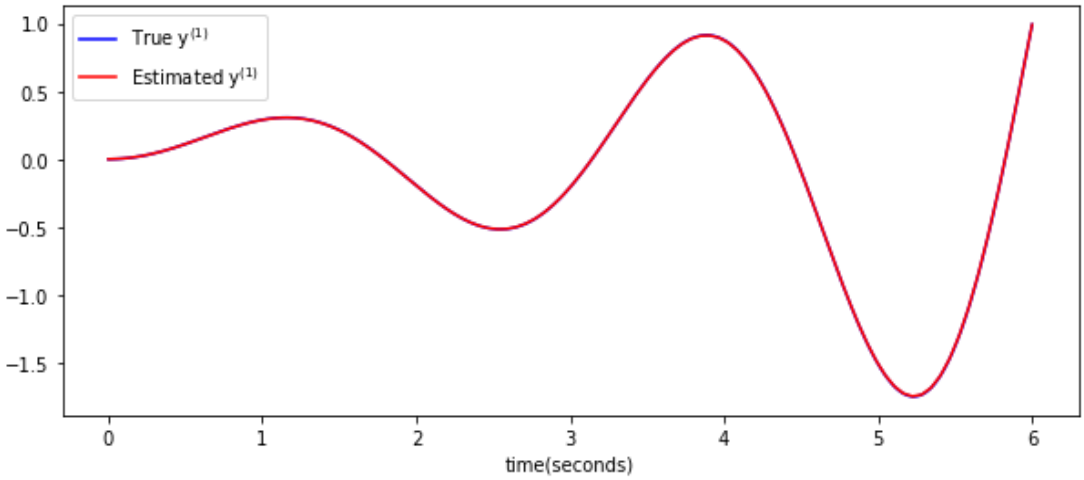

Fig. 3.4 Estimated Output $y^1(t)$

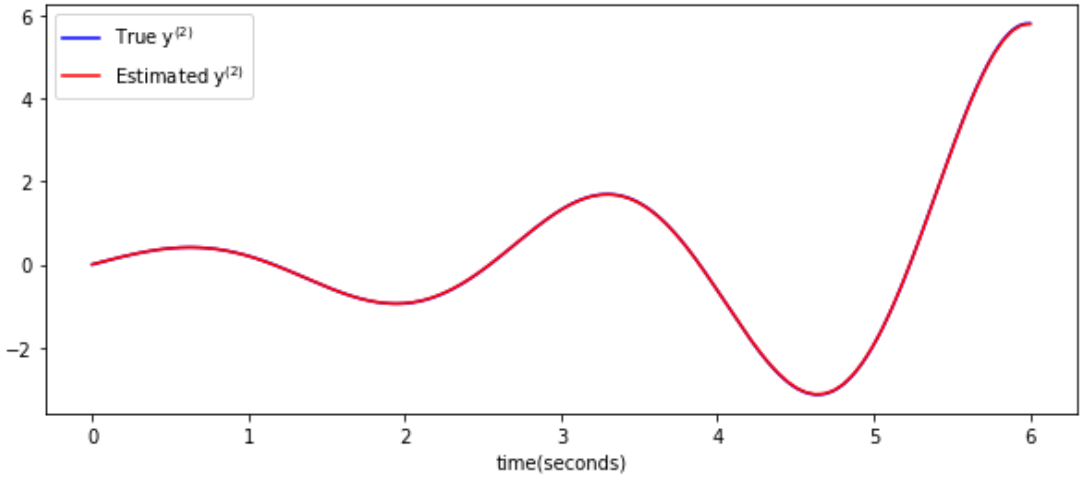

Fig. 3.5 Estimated Output $y^2(t)$

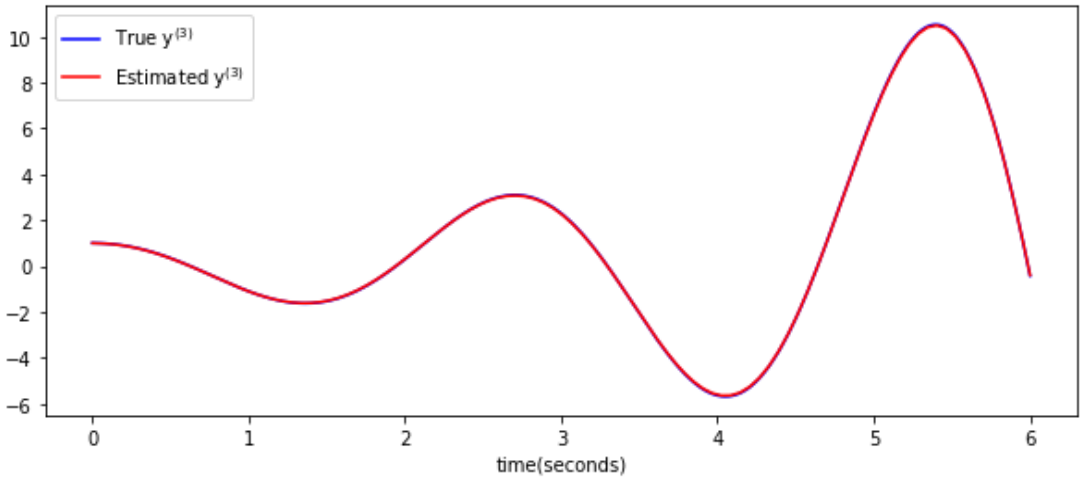

Fig. 3.6 Estimated Output $y^3(t)$

Noise level $\sigma = 0.5$, SNR = -4.75dB

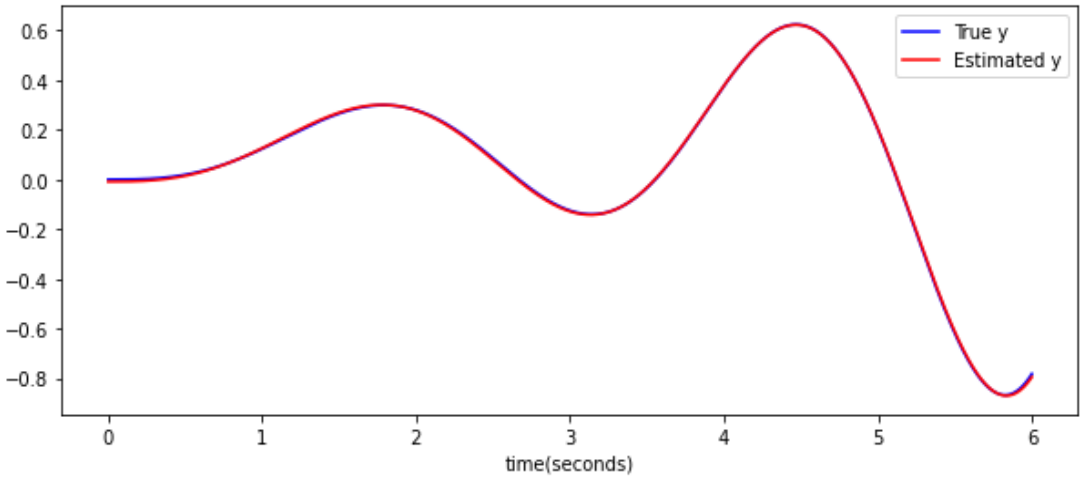

Fig. 3.7 Estimated Output $y(t)$

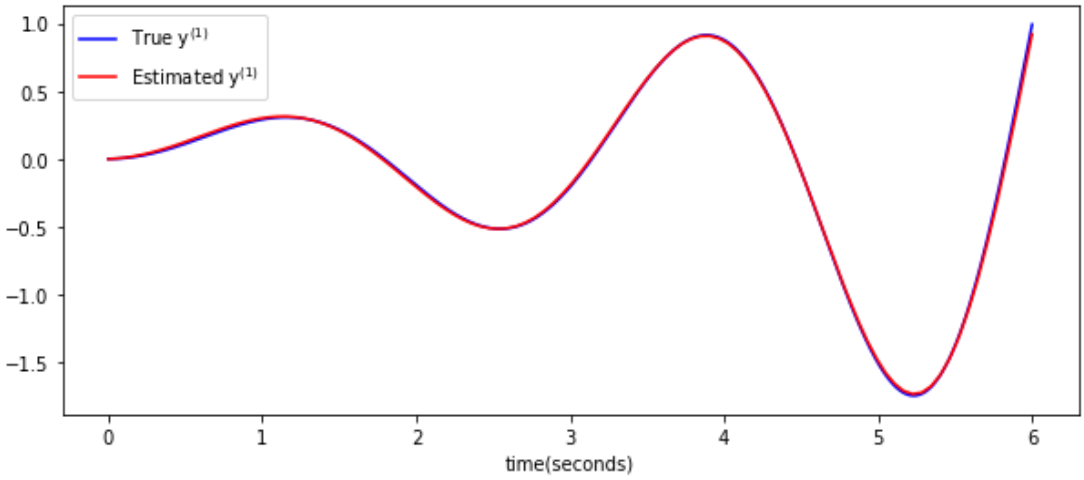

Fig. 3.8 Estimated Output $y^1(t)$

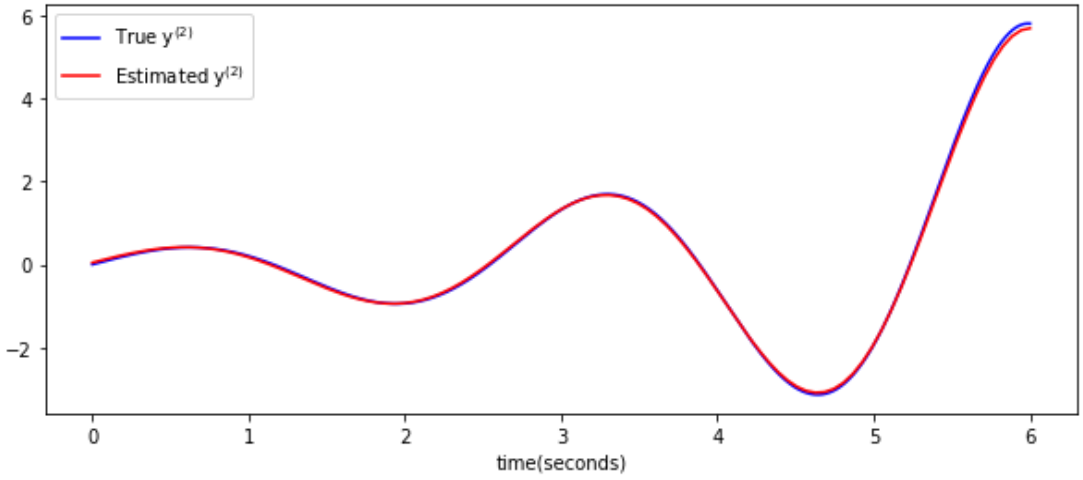

Fig. 3.9 Estimated Output $y^2(t)$

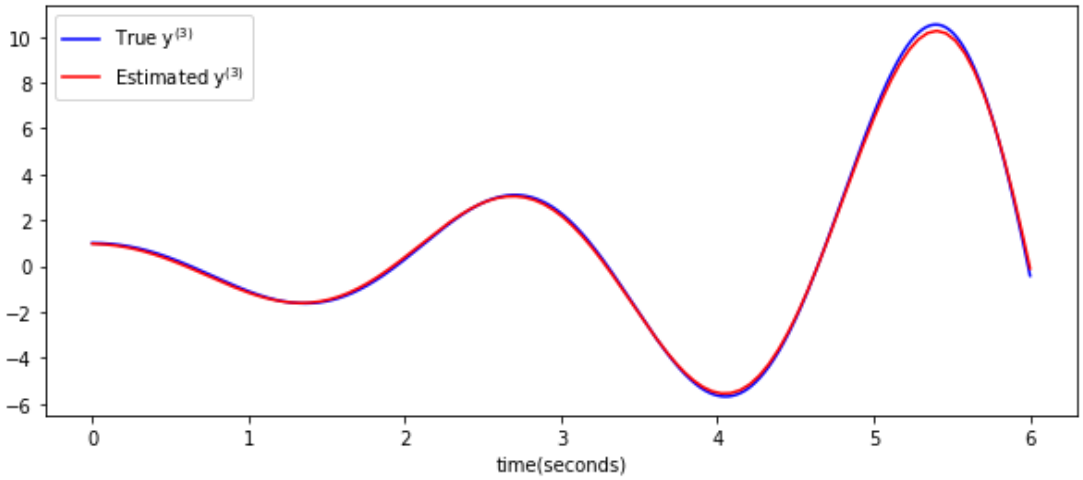

Fig. 3.10 Estimated Output $y^3(t)$

Noise level $\sigma = 1$, SNR = -9.5dB

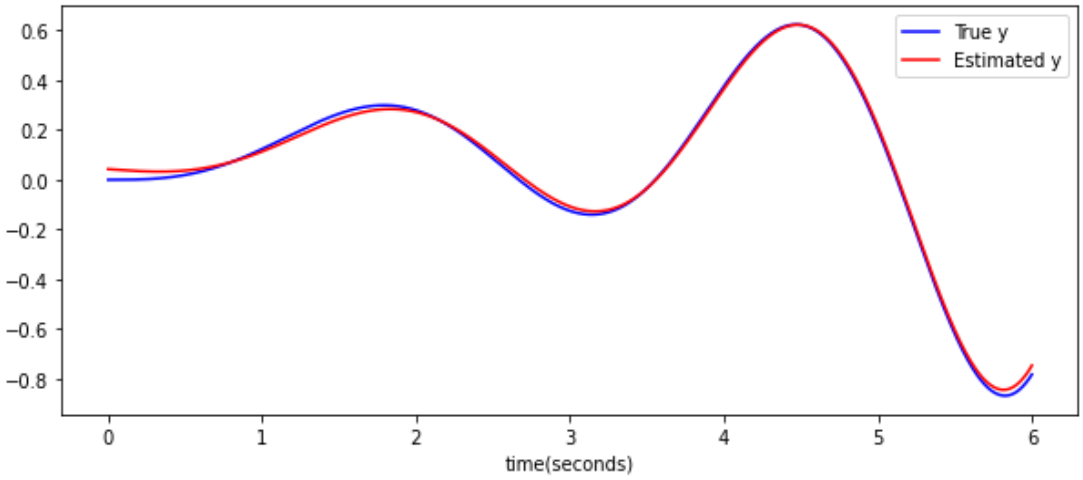

Fig. 3.11 Estimated Output $y(t)$

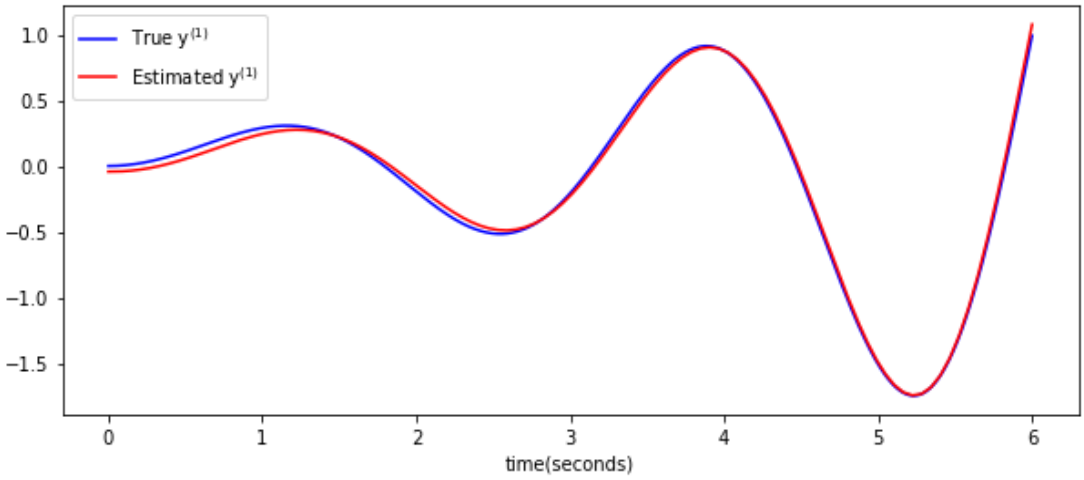

Fig. 3.12 Estimated Output $y^1(t)$

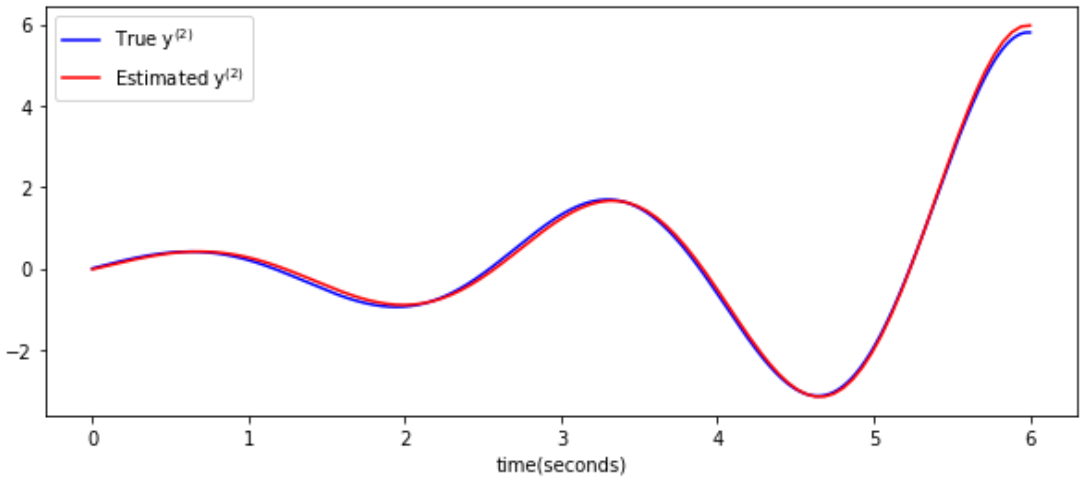

Fig. 3.13 Estimated Output $y^2(t)$

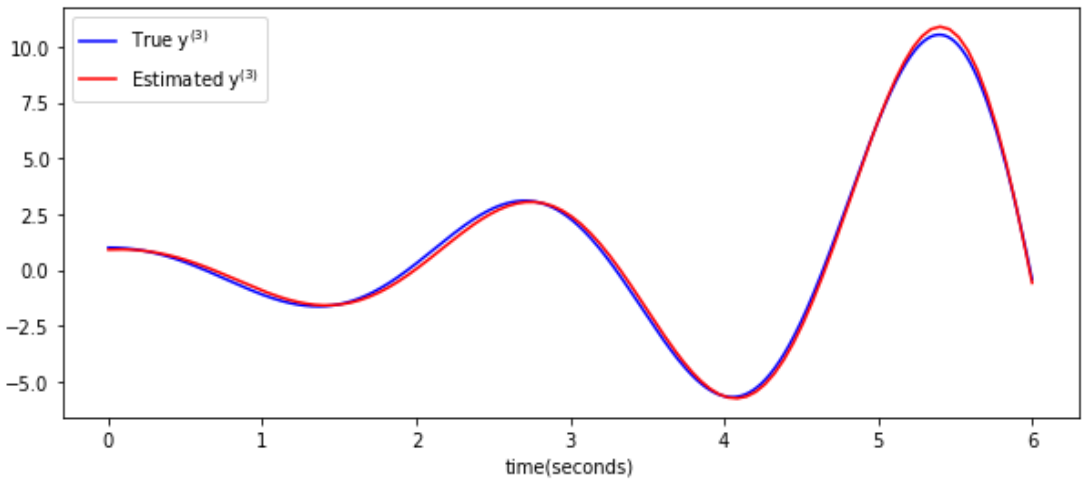

Fig. 3.14 Estimated Output $y^3(t)$

Noise level $\sigma = 3$, SNR = -18.45dB

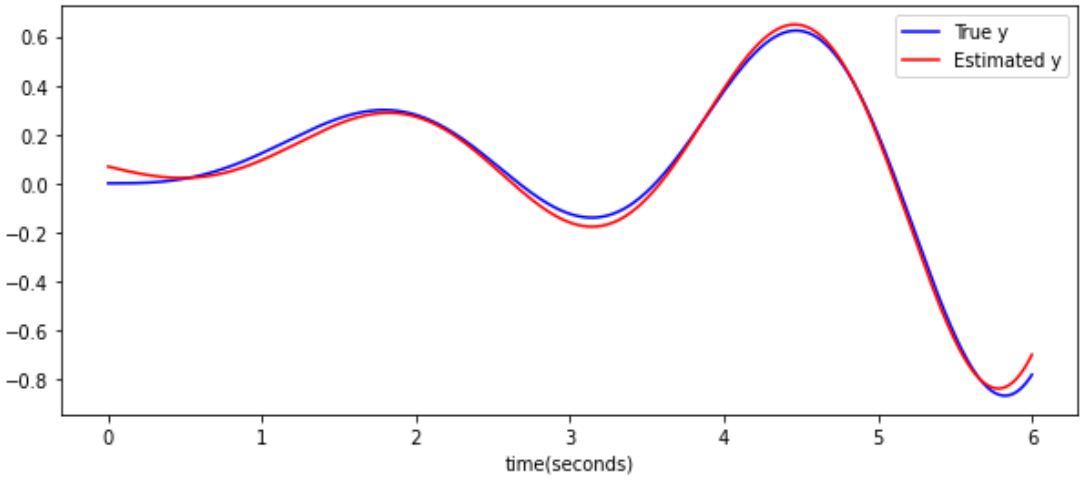

Fig. 3.15 Estimated Output $y(t)$

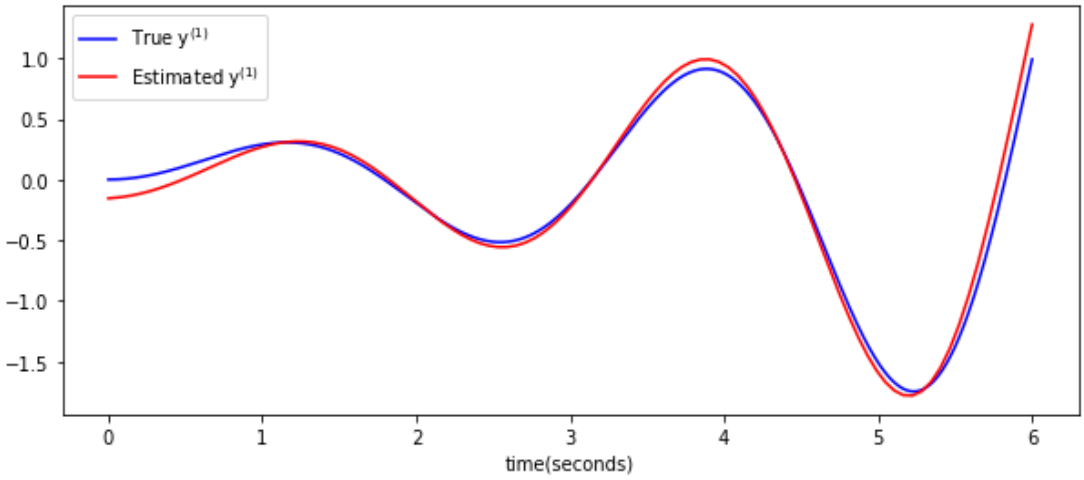

Fig. 3.16 Estimated Output $y^1(t)$

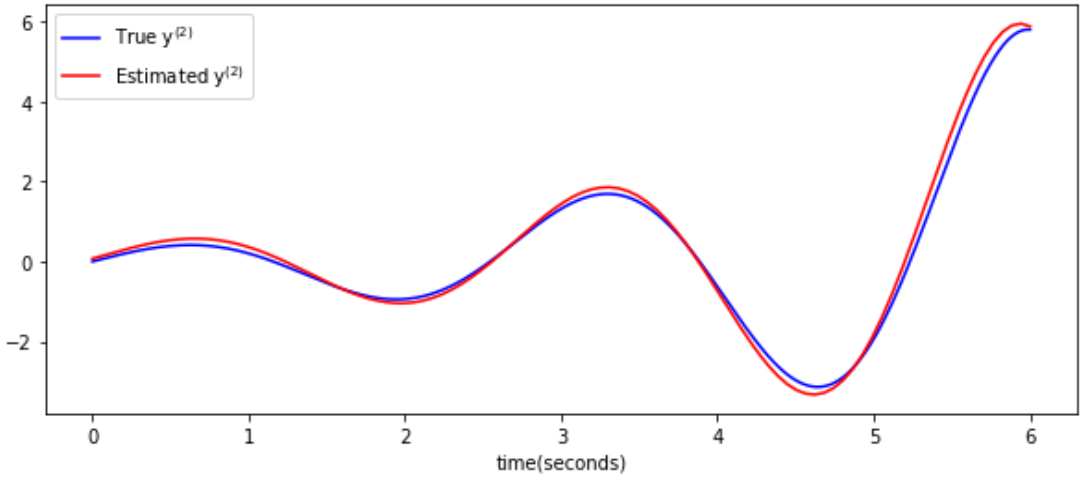

Fig. 3.17 Estimated Output $y^2(t)$

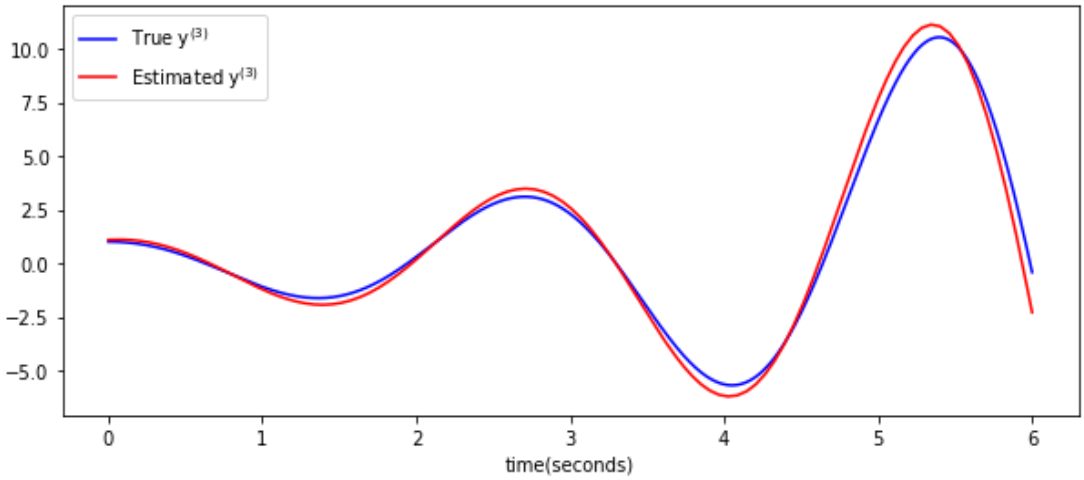

Fig. 3.18 Estimated Output $y^3(t)$

Chapter 4

State Estimation for LTI Systems with Known Parameters

This chapter discusses the state estimation of a SISO LTI system utilizing the parameters estimated in chapter 3. The focus is primarily on the Kalman filter, which is discussed in the initial sections followed by the optimal smoothing using the Rauch-Tung-Striebel (RTS) algorithm. The performance of this algorithm for various systems in analyzed in the results. This work was derived together with my lab partner Nithilasaravanan Kuppan[14].

4.1 An overview of state estimation techniques

Under the constraints of process and measurement noise, state estimation poses a significant challenge in generating accurate state values. Unmodeled dynamics, modelling approximations, and parameter uncertainties all contribute to process noise in the system. The physical characteristics of the sensor and the measurement process itself are the primary determinants of measurement noise.

The direct way to achieve convergence of a state irrespective of its initial condition is by using a Luenberger observer[7]. The Luenberger observer provides feedback to the model on the error caused by the discrepancy between the predicted and measured output. If the system is observable, the state values converge to the actual state of the system. The major limitation of this method is that it is suited only for deterministic systems. Unfortunately, most real-world systems are stochastic due to the induced process and measurement noise.

To circumvent this issue, Rudolph Kalman came up with the Kalman filter[8], which uses the stochastic properties of the noise to perform accurate state estimation. In his paper, Kalman proposed a recursive solution to the discrete-data linear filtering problem, assuming a normal distribution of process and measurement noise. The Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) are subsequent adaptations of this method to a nonlinear system.

4.2 Kalman filters for state estimation

Rudolf E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem in 1960. The Kalman filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator that is optimal in the sense that it minimizes the estimated error covariance when some presumed conditions are met[23].

4.2.1 Kalman filter algorithm - an overview[24]

This section provides the Kalman filter algorithm which is used for accurate estimation of system states $x \in \mathbb{R}^n$, given a noisy signal. The rigorous derivation for the algorithm can be found in [25].

Consider a discrete-time controlled system governed by the stochastic difference equation

$$x_k = A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + w_{k-1} \tag{4.1}$$

And the measurement $z \in \mathbb{R}^m$

$$z_k = H_k x_k + v_k \tag{4.2}$$

Here, u_k is the control input and the initial state x_0 is a random vector with known mean $\mu_0 = E[x_0]$ and covariance $P_0 = E[(x_0 - \mu_0)(x_0 - \mu_0)^T]$.

The random vector w_k represents the process noise and v_k is the measurement noise. Both the random vectors are temporally uncorrelated (white noise), zero mean random sequences with known covariances $E[w_k w_k^T] = Q_k$, $E[v_k v_k^T] = R_k$ where Q_k is process noise covariance matrix and R_k is measurement noise covariance matrix. In reality, the process and measurement noise covariance matrices change with each time step and measurement. In this thesis, it is assumed that they are constant.

The initial state $\hat{x}_0$ and the process noise covariance P_0 is given by

$$\hat{x}_0 = \mu_0 = E[x_0] \tag{4.3}$$
$$P_0 = E[(x_0 - \hat{x}_0))(x_0 - \hat{x}_0))^T] \tag{4.4}$$

The time update equations (predictor step) from time step $k-1$ to step k are given by

$$\hat{x}_k^- = A_{k-1}\hat{x}_{k-1} + B_{k-1}u_{k-1} \tag{4.5}$$
$$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + Q_{k-1} \tag{4.6}$$

The measurement update equations are given by

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_K)^{-1} \tag{4.7}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H_k\hat{x}_k^-) \tag{4.8}$$
$$P_k = (I - K_k H_k)P_k^- \tag{4.9}$$

Here, first the Kalman gain K_k is computed with the values from predictor equations. Further, the measurement z_k is obtained and a *posteriori* state estimate is generated. Eventually, a *posteriori* error covariance estimate is generated.

After each *predict* and *update* step, the entire process is repeated with a previous *a posteriori* estimates to predict a new *a priori* estimates. This recursive nature of Kalman filter makes the practical implementation of the algorithm feasible. Figure 4.1 shows the high level diagram of this algorithm along with the equations.

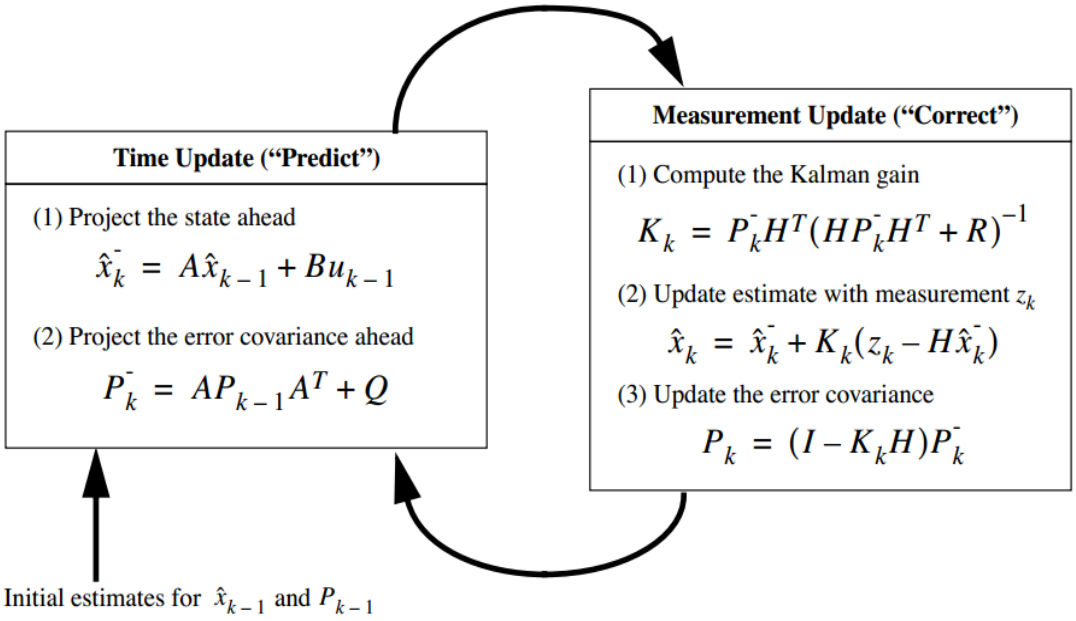

Fig. 4.1 Operation of a Kalman filter - A summary[24]

4.3 Optimal smoothing

By incorporating a suitable smoothing algorithm, the Kalman Filter's accuracy can be further improved. Smoothing algorithms are divided into three categories, which are summarised below[22, 26, 27].

Fixed-interval smoothers

Fixed-interval smoothers use all the measurements made at times t_{meas} over a fixed interval $t_{start} \leq t_{meas} \leq t_{end}$ to produce an estimated state vector $\hat{x}(t_{est})$ at time $t_{start} \leq t_{est} \leq t_{end}$ in the same fixed interval.

Fixed-interval smoothing is usually performed after taking all the measurements prior. As a result, it's usually used for post-processing the data collected during a test procedure. Fixed-interval smoothing is typically not a real-time process because the data processing occurs after the end of the measurement.

Fixed-lag smoothers

Fixed-lag smoothers use all measurements made over a time interval $t_{start} \leq t_{meas} \leq t_{est} + \Delta t_{lag}$ for the estimate $\hat{x}(t_{est})$ at time t_{est}. It means that the generated estimate at time t is for the value of x at time $t - \Delta t_{lag}$, where Δt_{lag} is a fixed time. Fixed-lag smoothers are commonly used in communications to improve signal estimation, however this method introduces some signal delay.

Fixed-point smoothers

Fixed-point smoothers generate an estimate $\hat{x}(t_{fixed})$ of x at a fixed time t_{fixed} based on all measurements $z(t_{meas})$ up to the current time $t, (t_{start} \leq t_{meas} \leq t)$. Fixed-point smoothers function as a predictor when $t < t_{fixed}$, as filters when $t = t_{fixed}$ and as smoothers when $t > t_{fixed}$.

Fixed-point smoothing is useful for estimation problems in which the system state is only of interest at some specific event time t_{fixed}, which is often the initial state.

4.3.1 Rauch-Tung-Striebel(RTS) smoother

Rauch-Tung-Striebel smoother [9, 21, 26, 27] is a fixed interval smoother, based on a *two-filter* model

(1) A **forward filter** running forward in time. At each instant of time, the estimate from the forward filter is based on all the measurements made up to that time, and the associated estimation uncertainty covariance characterizes estimation uncertainty based on all those measurements

(2) A **backward filter** running backward in time. At each instant of time, the estimate from the backward filter based on all the measurements made after that time, and the associated estimation uncertainty covariance characterizes estimation uncertainty based on all those measurements.

At each time t, the forward filter generates the covariance matrix $P_{[f]}(t)$ representing the mean-squared uncertainty in the estimate $\hat{x}_{[f]}(t)$ using all measurements $z(s)$ for $s \leq t$. Similarly, the backward filter generates the covariance matrix $P_{[b]}(t)$ representing the mean-squared uncertainty in the estimate $\hat{x}_{[b]}(t)$ using all measurements $z(s)$ for $s \geq t$. The optimal smoother combines $\hat{x}_{[f]}(t)$ and $\hat{x}_{[b]}(t)$, using $P_{[f]}(t)$ and $P_{[b]}(t)$ in a Kalman filter to minimize

the resulting covariance matrix $P_{[s]}(t)$ of smoother uncertainty. $P_{[s]}(t)$ will tell us how well the smoother performs [26].

4.3.2 RTS Algorithm

The following section provides a quick summary of the Kalman algorithm in terms of the equations used, without any of the derivations from [9, 21, 26, 27].

- **Forward pass**: Based on the equations given in the algorithm for Kalman filter, the standard filtered quantities, i.e., the smoothed means and corresponding covariances $\hat{x}_{k|k-1}, \hat{x}_{k|k}, P_{k|k-1}, P_{k|k}$ for $k = 0, ..., n$ are obtained and stored in memory

- **Backward pass**: To compute $\hat{x}_{k|n}$, the following equation is used

$$\hat{x}_{k|n} = \hat{x}_{k|k} + A_k \left(\hat{x}_{k+1|n} - \hat{x}_{k+1|k} \right), \quad k = n-1, \ldots, 0 \tag{4.10}$$

  where,

$$A_k = P_{k|k-1} \bar{F}_k^T P_{k+1|k}^{-1} \tag{4.11}$$

  and

$$P_{k+1|k} = \bar{F}_k P_{k|k-1} \bar{F}_k^T \tag{4.12}$$

  The error covariance can be found by

$$P_{k|n} = P_{k|k} + A_k \left(P_{k+1|n} - P_{k+1|k} \right) A_k^T \tag{4.13}$$

In the above equations, A_k is the *smoother gain matrix*, n is the final time step, $P_{k|n}$ is the corresponding *state error covariance* matrix, $\bar{F}_k$ is the state transition matrix and $\hat{x}_{k|n}$ is the *smoothed state* at time step k.

4.3.3 Improvement over Kalman outputs [26]

Theoretical limits on the asymptotic improvement of smoothing over filtering were shown in [28] for asymptotically exponentially stable dynamic systems. A factor of 2 in the mean-squared estimation uncertainty was found to be the limit but there is a possibility of greater improvement in unstable systems.

For multidimensional problems, if $P_{[s]}$ is the covariance matrix of smoothing uncertainty and $P_{[f]}$ is the covariance matrix of filtering uncertainty then for smoothing to be an improvement over filtering

$$P_{[s]} < P_{[f]}, or [P_{[f]} - P_{[s]} \text{ is positive - definite }] \tag{4.14}$$

Practically, this is done by comparing the covariance matrices after the implementation of both the filter and smoother algorithms.

4.4 State estimation algorithm

The pseudo-code for leveraging Kalman filter + RTS smoother is described below in Algorithm 1. This algorithm works under the assumption that the signal to be filtered Y_M of known order n is induced with Additive Gaussian White Noise (AWGN). Also, this algorithm utilizes the system parameters estimated in Chapter 3 using RLS algorithm.

Algorithm 1 State estimation using Kalman + RTS

Initialize: $a_0 = a_{k_n}$ where a_{k_n} are the parameters of the system of order n
Initialize: $x_{initial} = [y_0;\ y_0^{(1)};\ y_0^{(2)}; ...;\ y_0^{(n)}]$
Procedure Kalman Filter
Initialize: A_k, $P_{initial}, Q, R$ and H matrices
Compute the state transition matrix $F = e^{A_k t}$
for $k = 0, 1, ..., N$ **do**
  while $Y_{m_k} \in Y_m$ **do**
    Predict I: $x_k^f = F_{k-1} x_{k-1}$ {State Extrapolation [f for forecast]}
    Predict II: $P_k^f = F_{k-1} P_{k-1} F_{k-1}^T + Q$ {Covariance Extrapolation}
    Set $z_k = Y_{m_k}$
    Update I: $K_k = P_k^f H^T (H P_k^f H^T + R_k^{-1})$ {Kalman Gain}
    Update II: $x_k = x_n^f + K_k(z_k - H x_k)$ {State Update}
    Update III: $P_k = (I - K_n H) P_k^f$ {Covariance Update}
    return x_k, P_k
  end while
end for
Append x_k to X and P_k to P {Stored as a batch}
Procedure Rauch-Tung-Striebel {Fixed Interval Smoothing}
Initialize $\hat{x_{k|n}} = X[-1]$ and $P_{k|n} = P[-1]$ {Backward filter}
for $k = N-1, ..., 1, 0$ **do**
  while $x_k \in X$ **do**
    Calculate $A_k = P_{k|k-1} \bar{F}_k^T P_{k+1|k}^{-1}$ {Smoother gain matrix}
    Calculate $P_{k+1|k} = \bar{F}_k P_{k|k-1} \bar{F}_k^T$ using 4.13 {Error covariance}
    Update: $\hat{x}_{k|n} = \hat{x}_{k|k} + A_k \left(\hat{x}_{k+1|n} - \hat{x}_{k+1|k}\right)$
    return x_k
  end while
end for
Append x_k to x_E

4.5 Results

4.5.1 Example 1 :

Let us consider the same fourth order system used as an example in Chapter 3. The LTI system is given by

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & -5 & -5 & 0 \end{bmatrix} x \quad ; y = x_1 \quad ; \quad x(0) = [0, 0, 0, 1] \tag{4.15}$$

with its corresponding characteristic equation

$$y^{(4)}(t) + 0y^{(3)}(t) + 5y^{(2)}(t) + 5y^{(1)}(t) + 1y(t) = 0 \tag{4.16}$$

The poles of this system are $0.4562 \pm 2.334i, -0.633, -0.279$. This makes the system unstable. The output of the system is superimposed with Additive White Gaussian Noise (AWGN) with different Signal to Noise Ratio (SNR) and the accuracy of state estimation is compared.

The following graphs show the reconstruction of the actual signal and its derivatives using parameters estimated from the techniques described in Chapter 2 and Chapter 3 and with the help of noisy signal. It can be noted that kalman filter will converge to the true state irrespective of the initial state condition. This is tested by assuming different initial condition during the state estimation process.

4.5.2 State estimation for various noise levels:

The estimated output signal and their derivatives are compared with the true signal.

Noise level $\sigma = 0$, SNR = 0dB

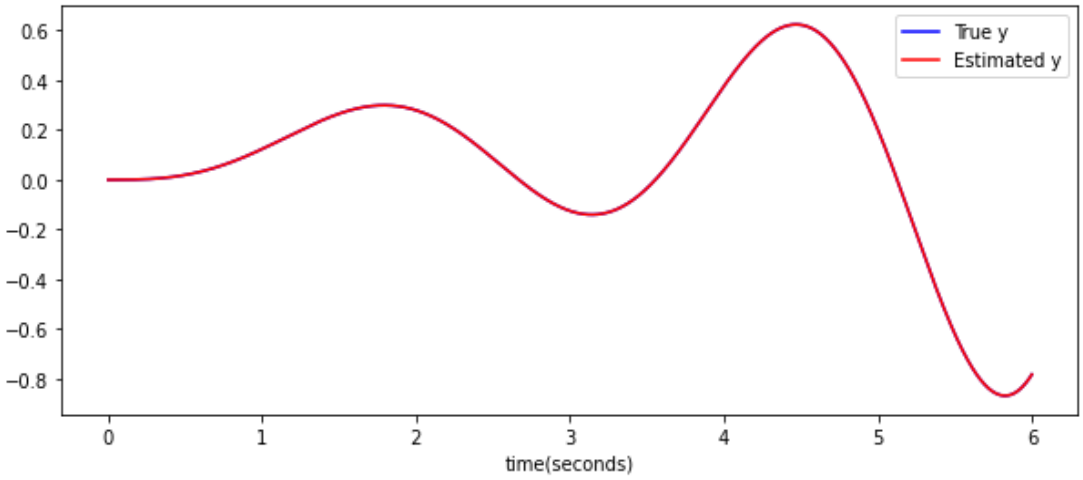

Fig. 4.2 Estimated Output $y(t)$

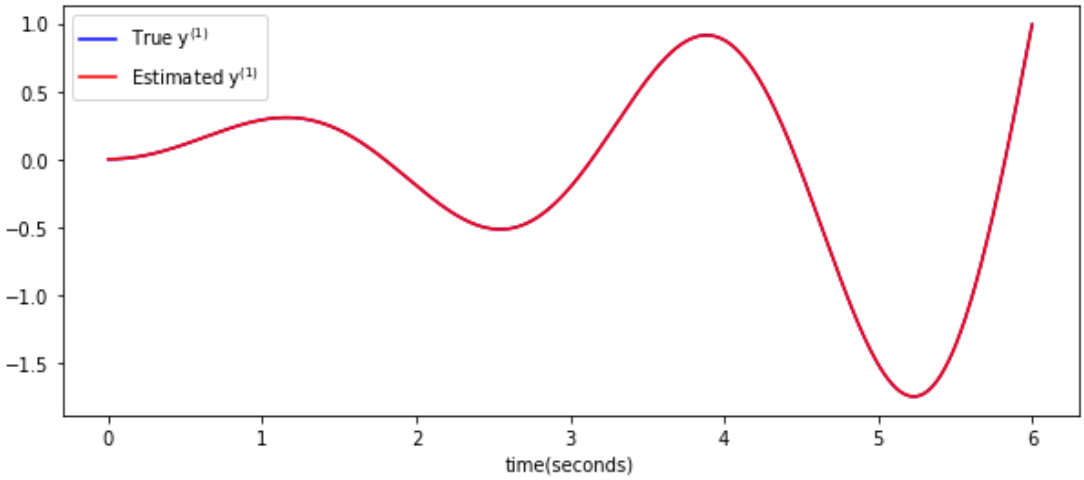

Fig. 4.3 Estimated Output $y^1(t)$

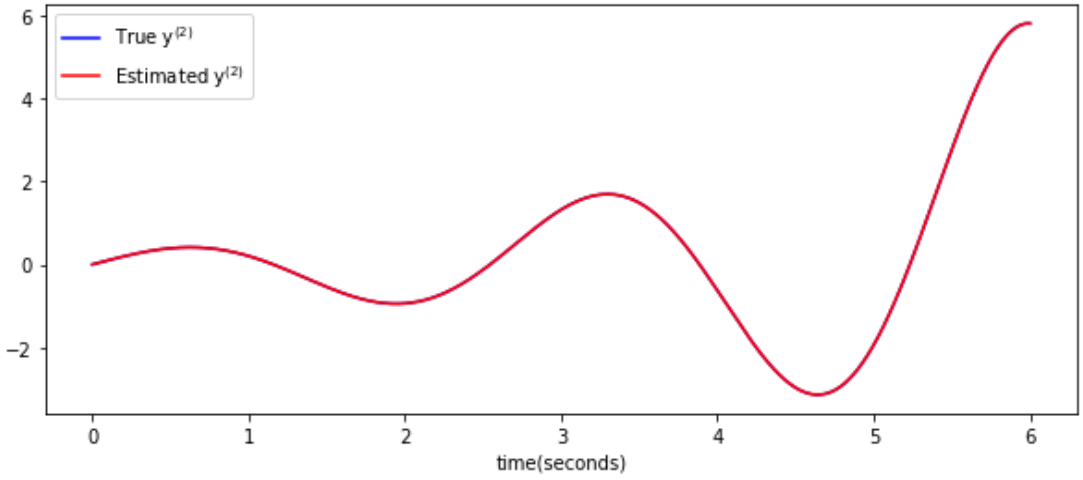

Fig. 4.4 Estimated Output $y^2(t)$

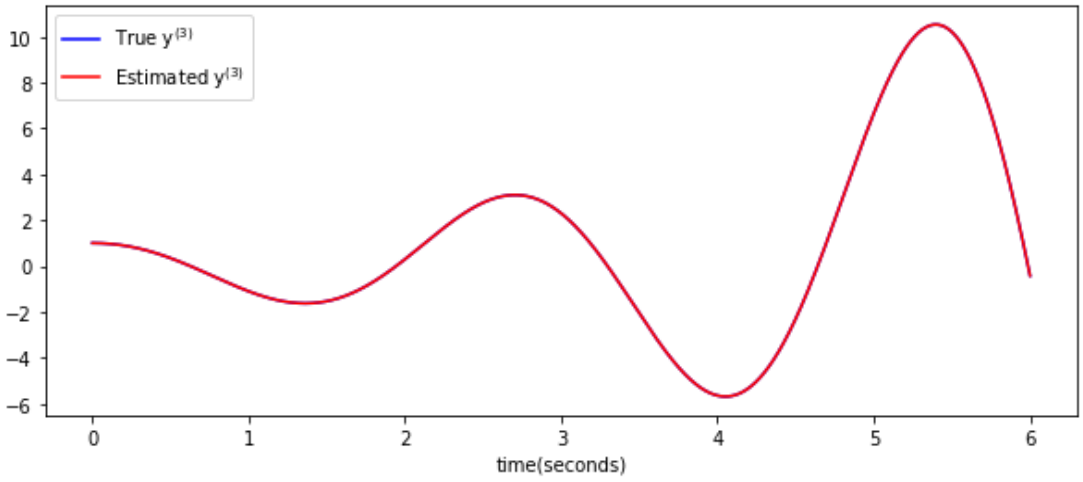

Fig. 4.5 Estimated Output $y^3(t)$

Noise level $\sigma = 0.5$, SNR = -4.75dB

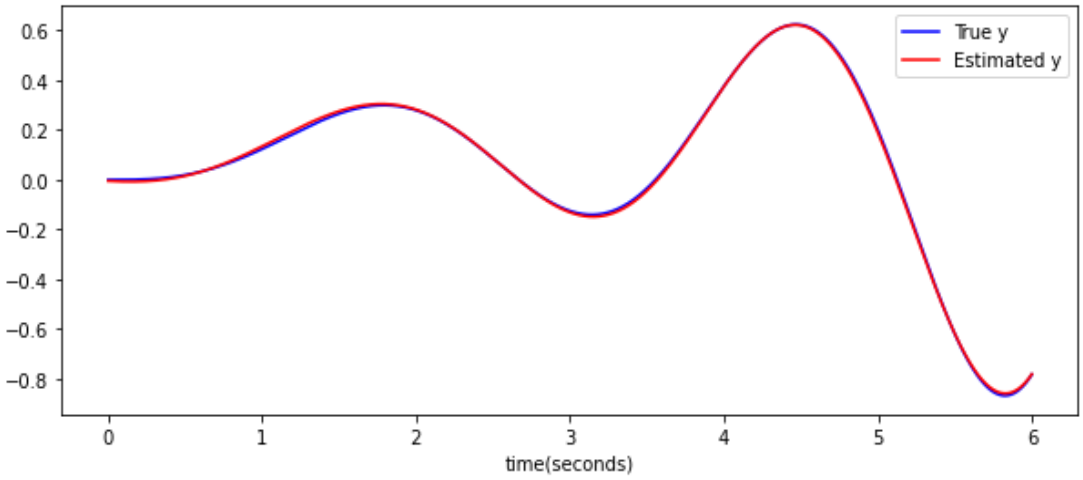

Fig. 4.6 Estimated Output $y(t)$

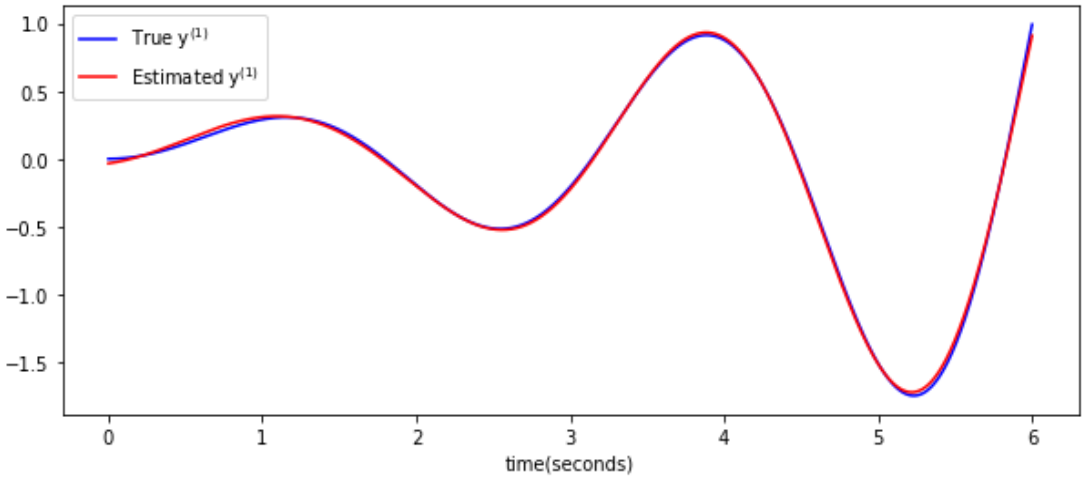

Fig. 4.7 Estimated Output $y^1(t)$

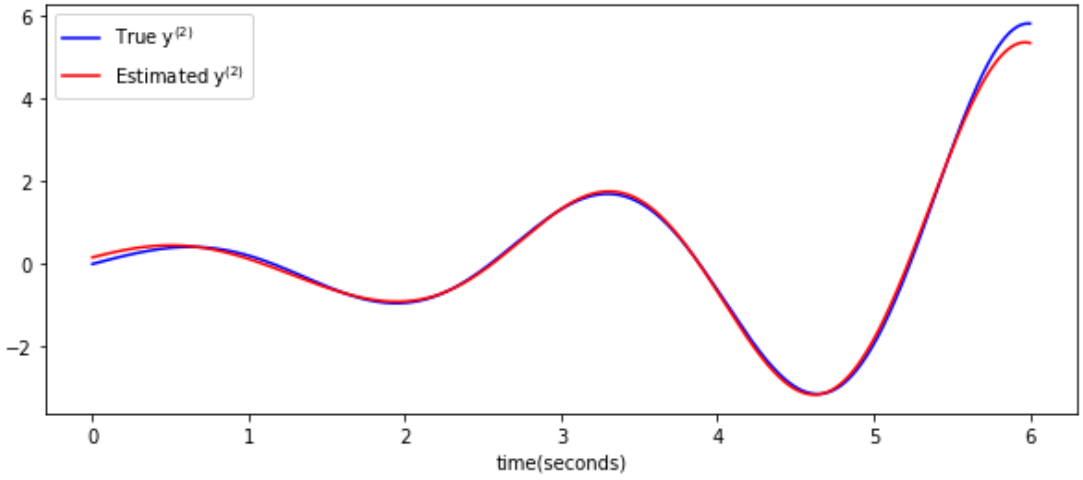

Fig. 4.8 Estimated Output $y^2(t)$

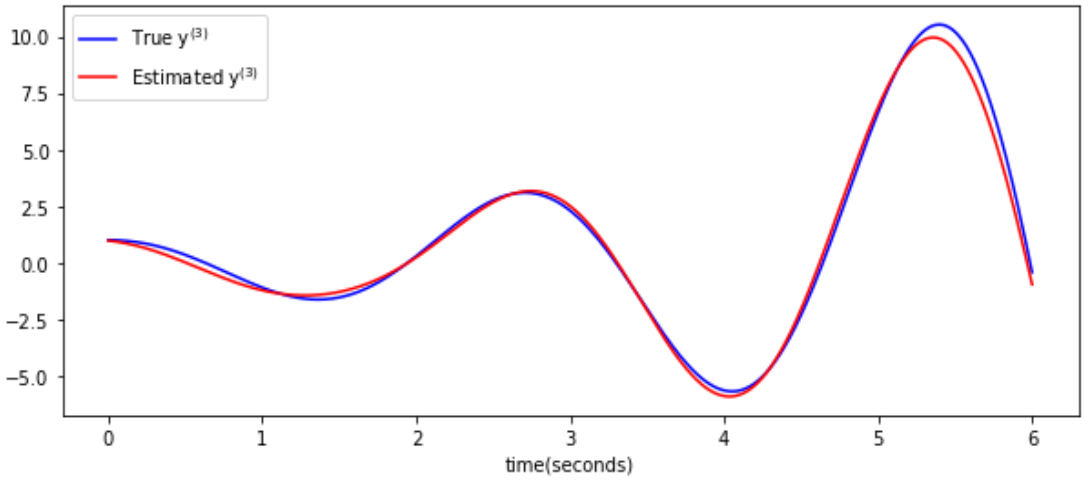

Fig. 4.9 Estimated Output $y^3(t)$

Noise level $\sigma = 1$, SNR = -9.5dB

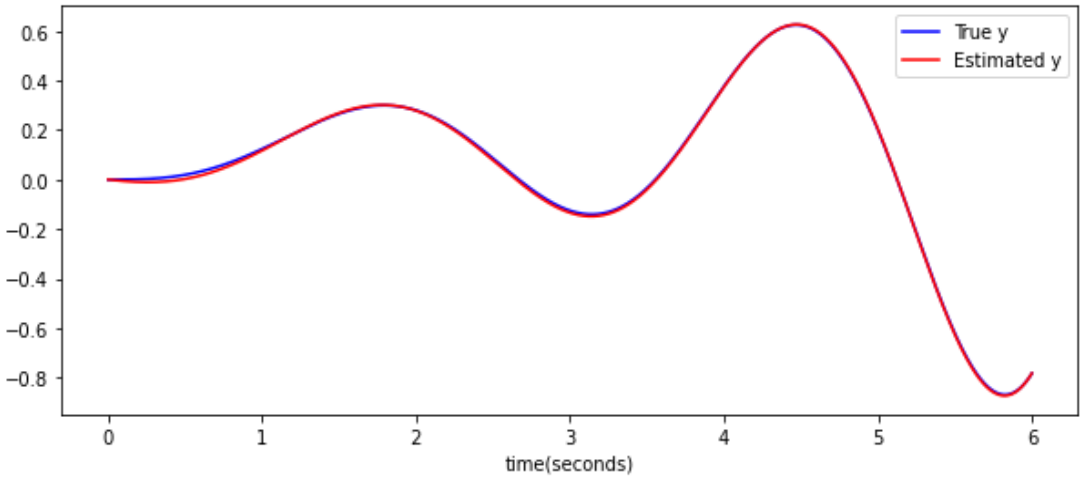

Fig. 4.10 Estimated Output $y(t)$

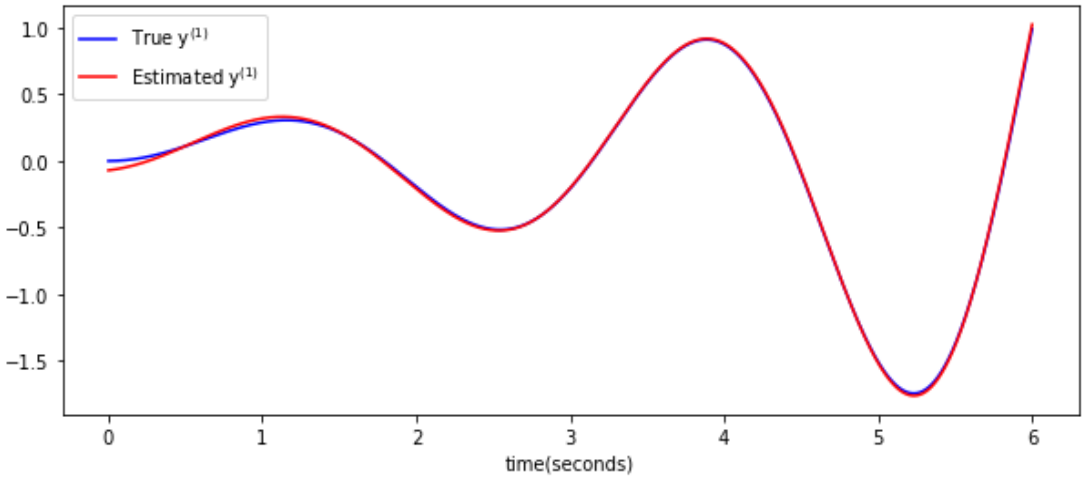

Fig. 4.11 Estimated Output $y^1(t)$

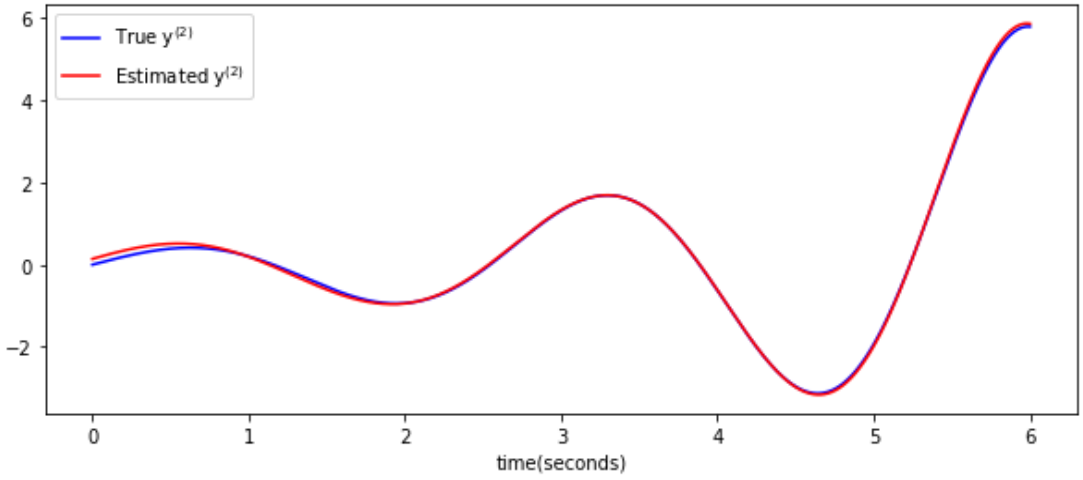

Fig. 4.12 Estimated Output $y^2(t)$

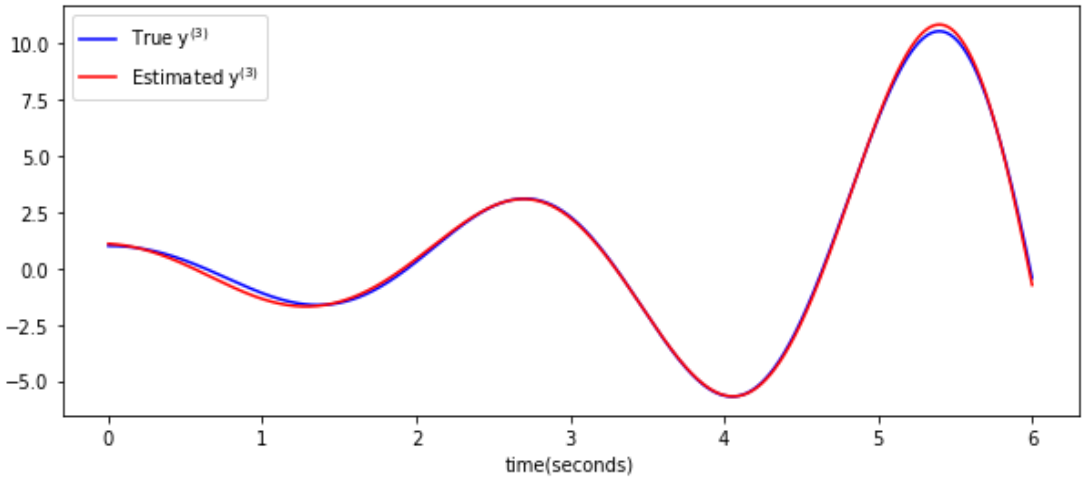

Fig. 4.13 Estimated Output $y^3(t)$

Noise level $\sigma = 3$, SNR = -18.45dB

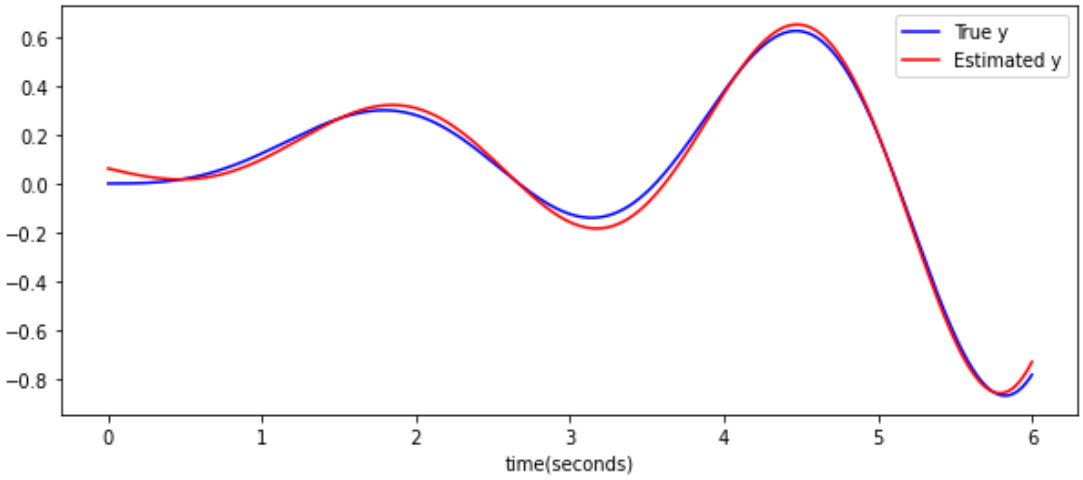

Fig. 4.14 Estimated Output $y(t)$

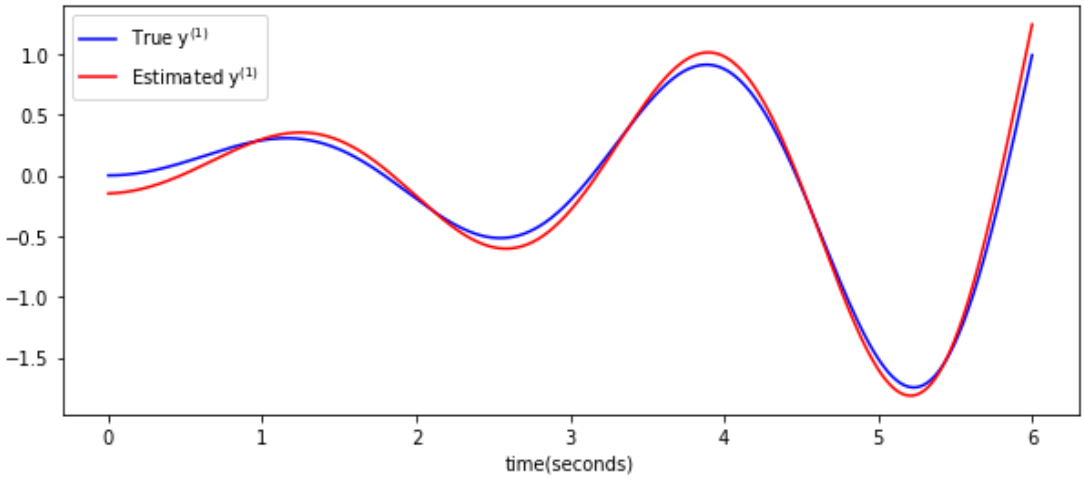

Fig. 4.15 Estimated Output $y^1(t)$

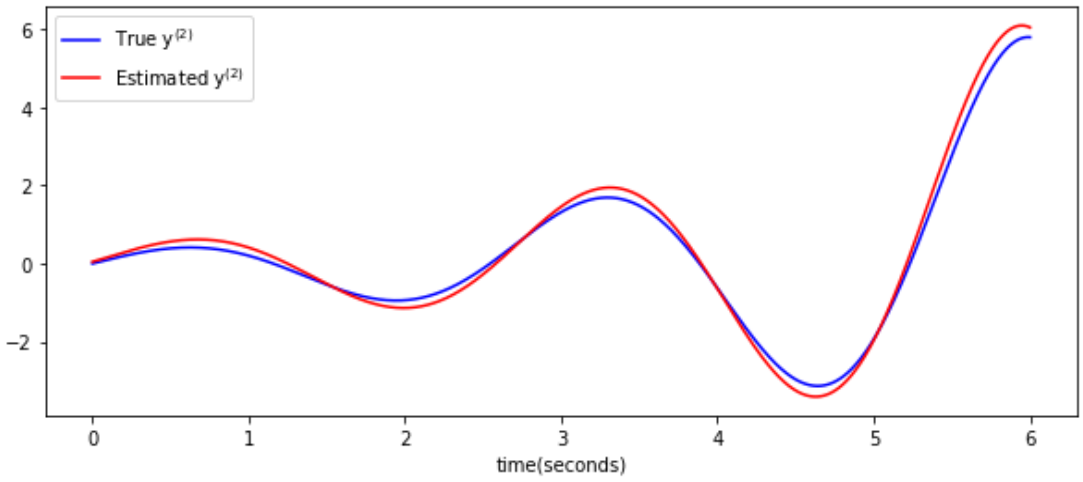

Fig. 4.16 Estimated Output $y^2(t)$

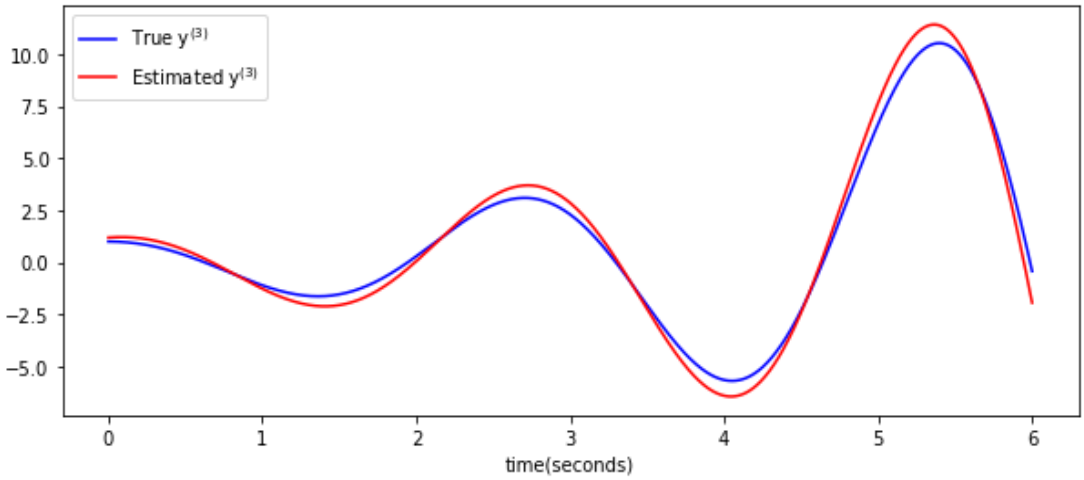

Fig. 4.17 Estimated Output $y^3(t)$

Table 4.1: Calculated error metrics for the estimated output and its derivatives for fourth-order LTI system using Kalman filter with Raunch-Tung-Striebel algorithm

Standard deviation (σ)	**SNR(dB)**	**Output**	**MAD**	**RMSE**	**MAE**
0	0	y	2.36×10^{-5}	1.14×10^{-5}	9.74×10^{-6}
		$y^{(1)}$	1.12×10^{-4}	3.27×10^{-5}	2.47×10^{-5}
		$y^{(2)}$	1.17×10^{-4}	6.80×10^{-5}	5.88×10^{-5}
		$y^{(3)}$	6.49×10^{-4}	1.18×10^{-4}	1.43×10^{-4}
0.5	-4.75	y	0.013	0.007	0.006
		$y^{(1)}$	0.082	0.021	0.017
		$y^{(2)}$	0.467	0.105	0.072
		$y^{(3)}$	1.018	0.346	0.255
1	-9.5	y	0.017	0.007	0.005
		$y^{(1)}$	0.070	0.017	0.013
		$y^{(2)}$	0.149	0.060	0.042
		$y^{(3)}$	0.316	0.136	0.103
3	-18.45	y	0.061	0.025	0.022
		$y^{(1)}$	0.252	0.084	0.070
		$y^{(2)}$	0.510	0.221	0.189
		$y^{(3)}$	1.513	0.541	0.451

4.5.3 Example 2 :

Let us consider another example. The LTI system is given by

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -150 & -125 & -31 & -5 \end{bmatrix} x \quad ; y = x_1 \quad ; \quad x(0) = [0, 0, 0, 1] \tag{4.17}$$

with its corresponding characteristic equation

$$y^{(4)}(t) + 5y^{(3)}(t) + 31y^{(2)}(t) + 125y^{(1)}(t) + 150y(t) = 0 \tag{4.18}$$

The poles of this system are $-0.0194 \pm 4.9744i, -2.7828, -2.1782$. This makes the system critically stable. The output of the system is superimposed with Additive White Gaussian Noise (AWGN) with different Signal to Noise Ratio (SNR) and the accuracy of state estimation is compared.

4.5.4 State estimation for various noise levels:

The estimated output signal and their derivatives are compared with the true signal.

Noise level $\sigma = 0$, SNR = 0dB

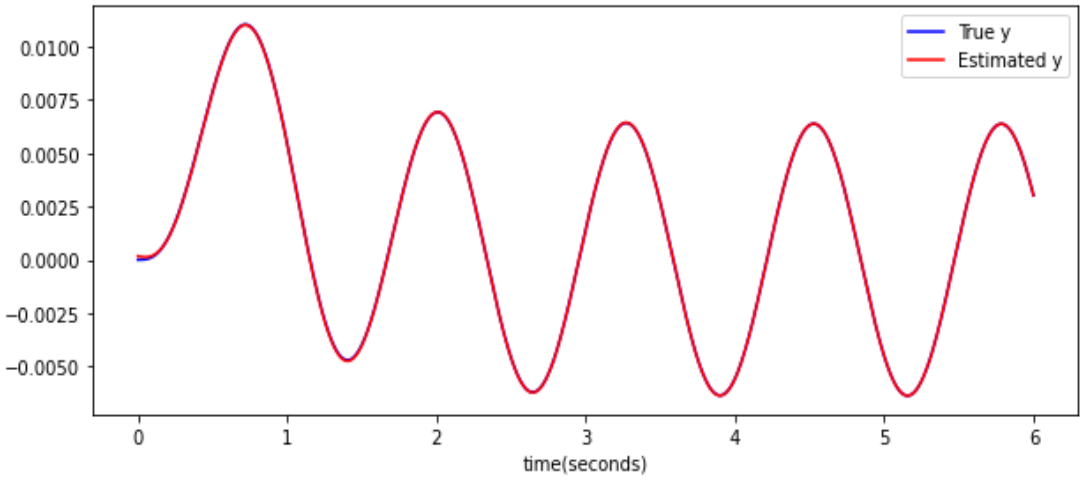

Fig. 4.18 Estimated Output $y(t)$

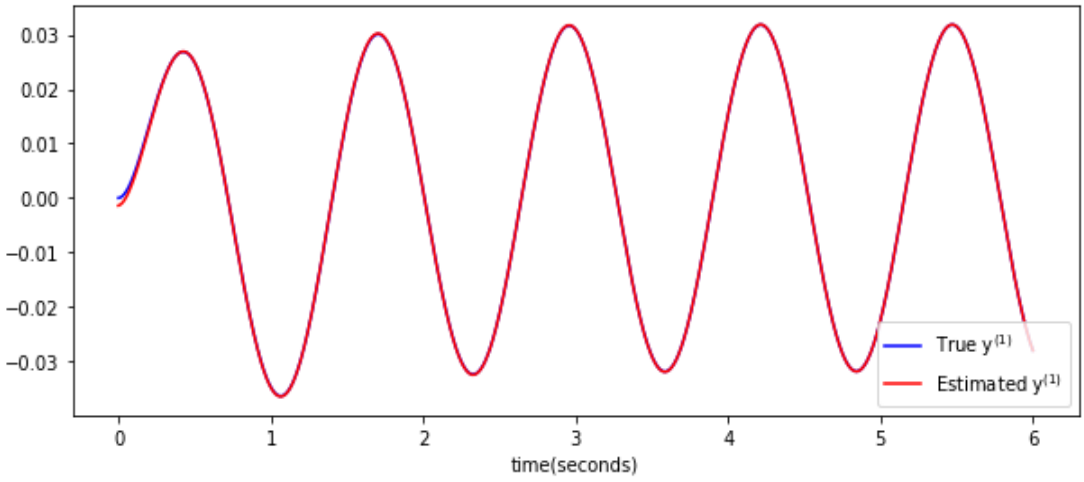

Fig. 4.19 Estimated Output $y^1(t)$

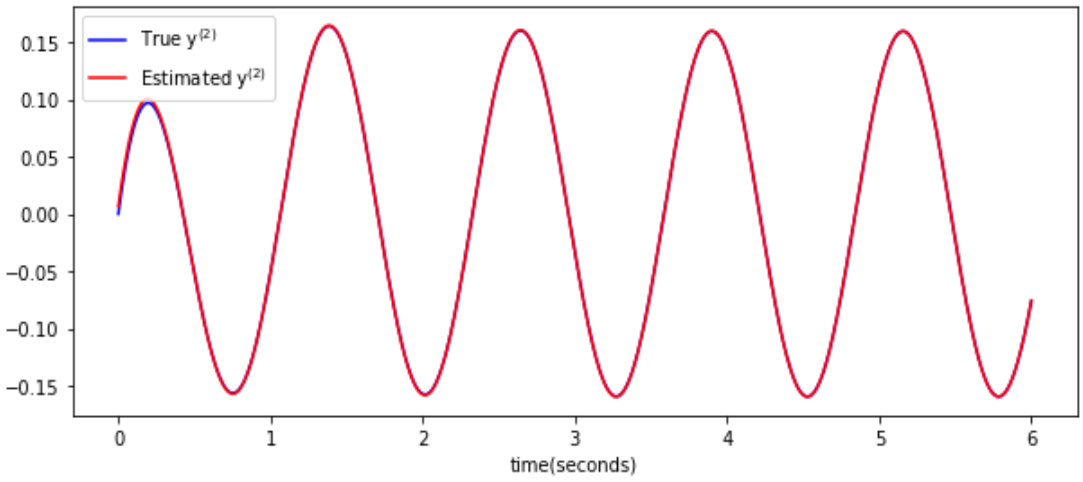

Fig. 4.20 Estimated Output $y^2(t)$

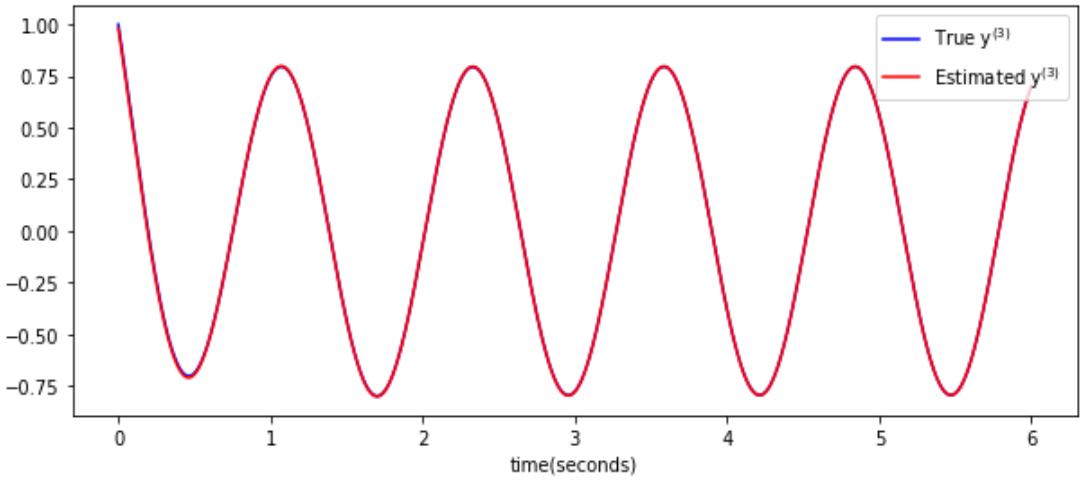

Fig. 4.21 Estimated Output $y^3(t)$

Noise level $\sigma = 0.007$, SNR = -4.5dB

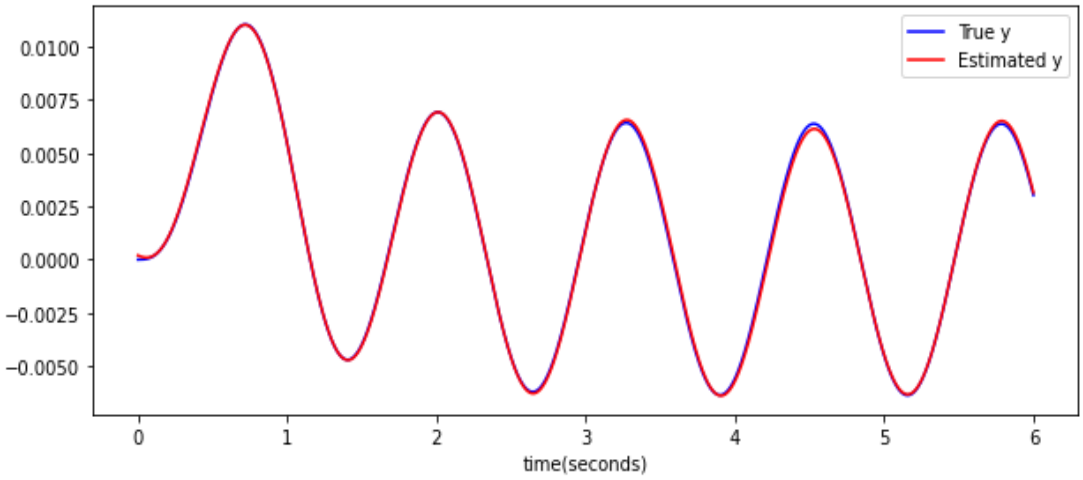

Fig. 4.22 Estimated Output $y(t)$

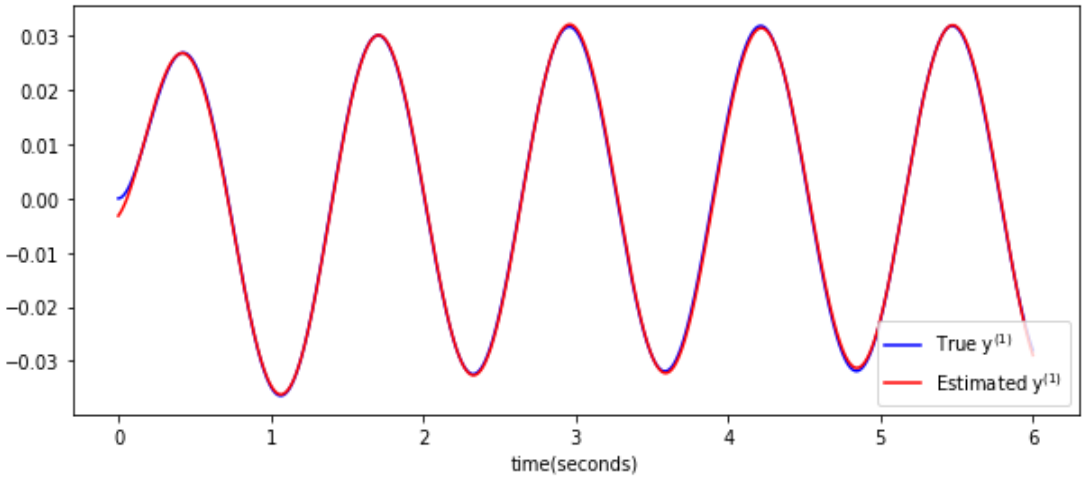

Fig. 4.23 Estimated Output $y^1(t)$

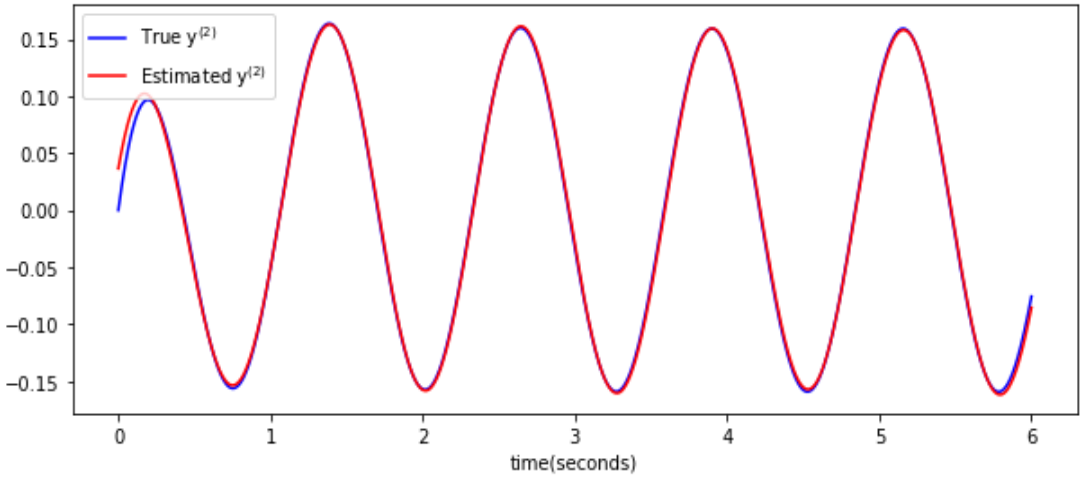

Fig. 4.24 Estimated Output $y^2(t)$

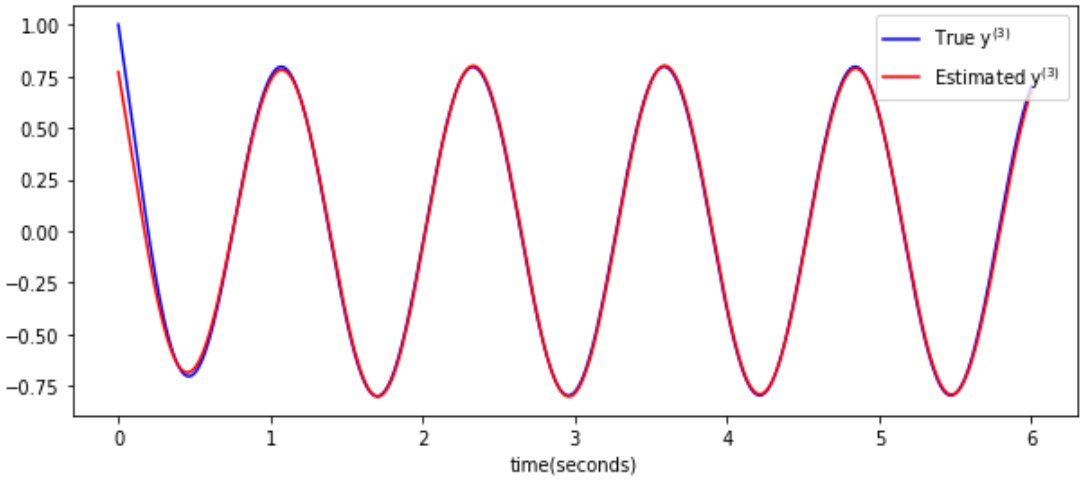

Fig. 4.25 Estimated Output $y^3(t)$

Noise level $\sigma = 0.01$, SNR = -7dB

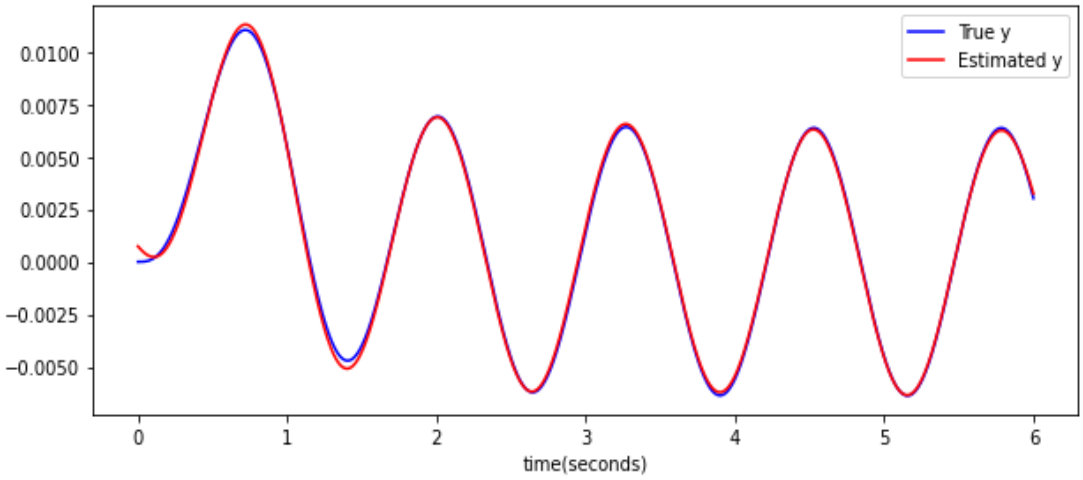

Fig. 4.26 Estimated Output $y(t)$

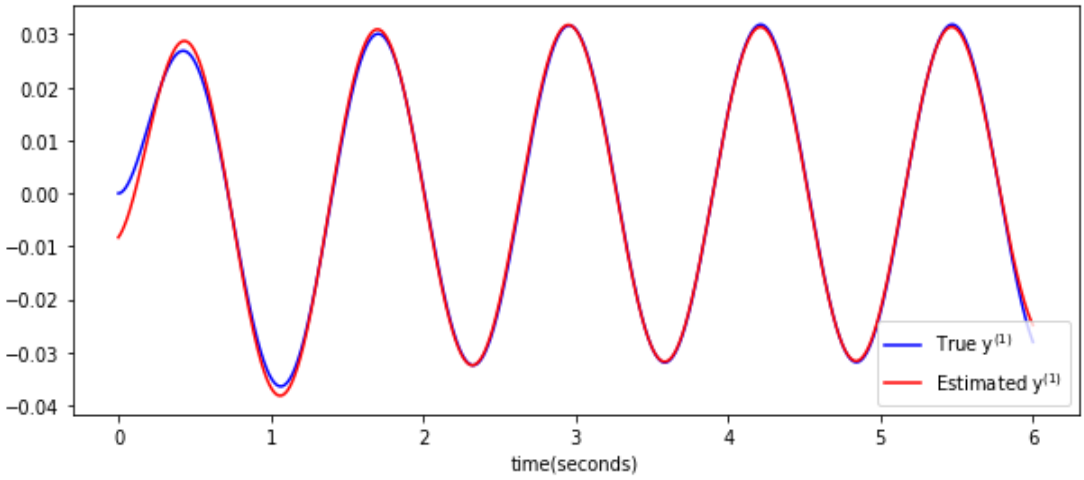

Fig. 4.27 Estimated Output $y^1(t)$

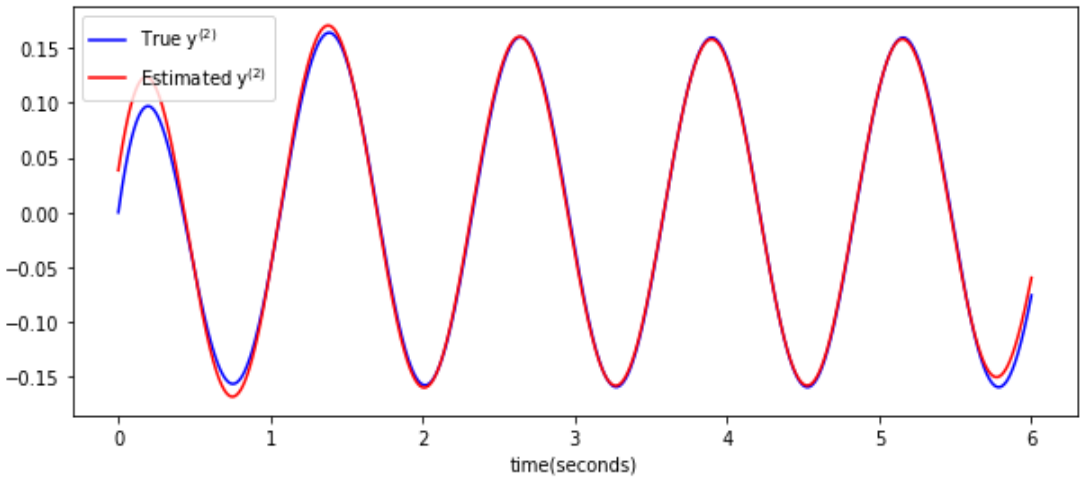

Fig. 4.28 Estimated Output $y^2(t)$

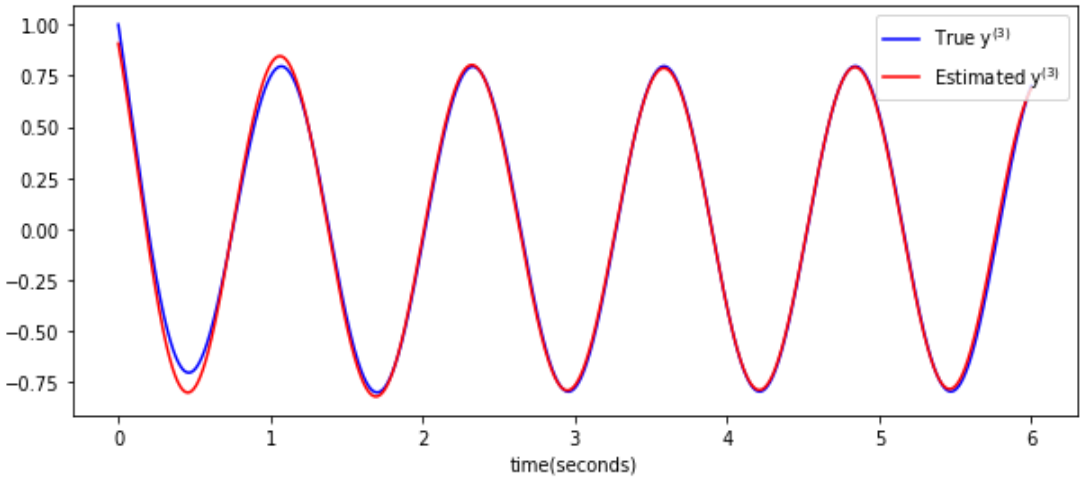

Fig. 4.29 Estimated Output $y^3(t)$

Noise level $\sigma = 0.017$, SNR = -11dB

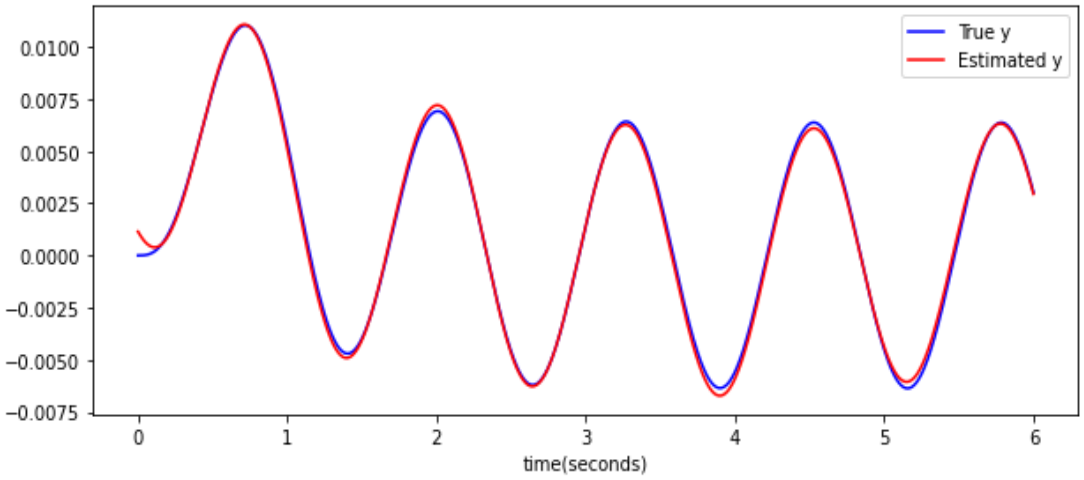

Fig. 4.30 Estimated Output $y(t)$

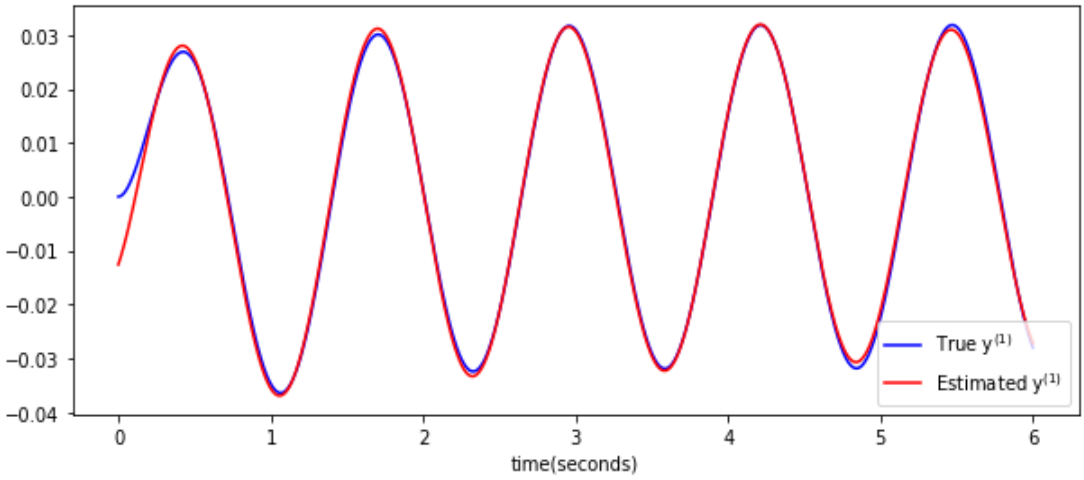

Fig. 4.31 Estimated Output $y^1(t)$

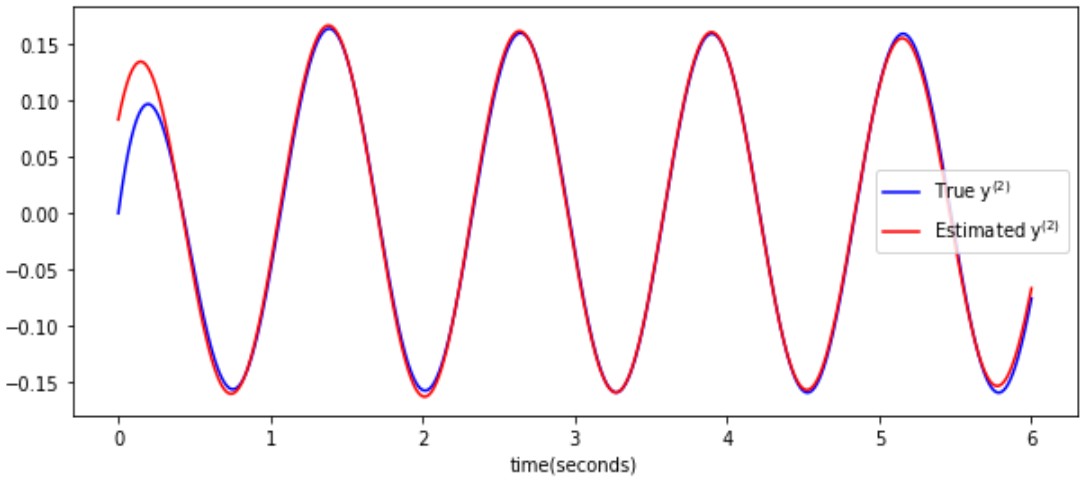

Fig. 4.32 Estimated Output $y^2(t)$

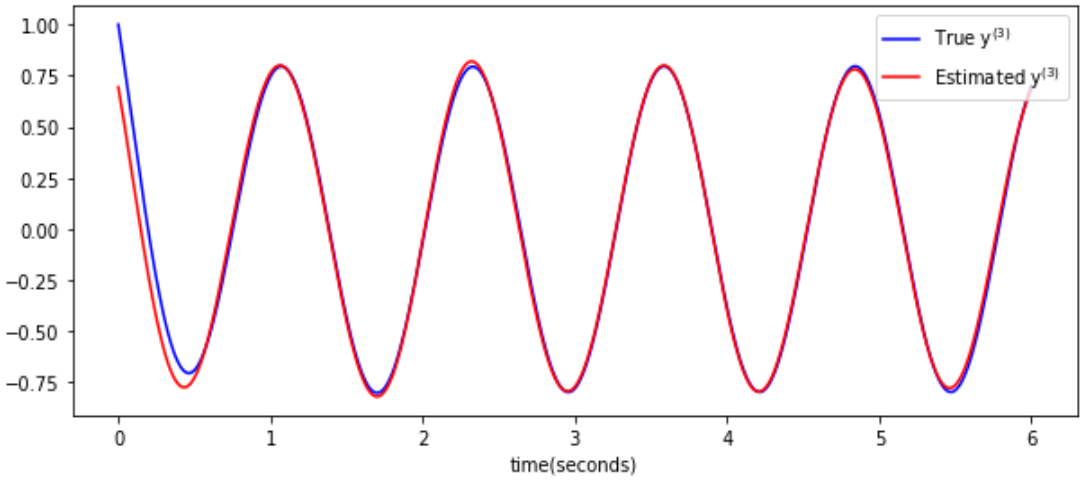

Fig. 4.33 Estimated Output $y^3(t)$

Table 4.2: Calculated error metrics for the estimated output and its derivatives for fourth-order LTI system using Kalman filter with Raunch-Tung-Striebel algorithm

Standard deviation (σ)	**SNR(dB)**	**Output**	**MAD**	**RMSE**	**MAE**
0	0	y	1.6×10^{-4}	2.47×10^{-5}	1.61×10^{-5}
		$y^{(1)}$	1.4×10^{-3}	1.78×10^{-4}	8.16×10^{-5}
		$y^{(2)}$	6.72×10^{-3}	1.06×10^{-3}	4.9×10^{-4}
		$y^{(3)}$	0.019	0.004	0.002
0.007	-4.5	y	3.08×10^{-4}	1.17×10^{-4}	8.56×10^{-5}
		$y^{(1)}$	3.19×10^{-3}	5.08×10^{-4}	3.94×10^{-4}
		$y^{(2)}$	0.037	0.004	0.003
		$y^{(3)}$	0.229	0.35	0.018
0.01	-7	y	7.37×10^{-4}	1.71×10^{-4}	1.37×10^{-4}
		$y^{(1)}$	8.32×10^{-4}	1.25×10^{-4}	7.57×10^{-4}
		$y^{(2)}$	0.038	0.008	0.005
		$y^{(3)}$	0.101	0.033	0.024
0.017	-11	y	1.15×10^{-3}	2.42×10^{-4}	1.98×10^{-4}
		$y^{(1)}$	1.26×10^{-2}	1.48×10^{-3}	8.30×10^{-4}
		$y^{(2)}$	0.083	0.011	0.005
		$y^{(3)}$	0.306	0.060	0.030

Chapter 5

Unscented Kalman Filter Method for Joint Parameter and State Estimation

As discussed in chapter 4, for the Kalman filters to accurately estimate the states at any given time, they must have complete knowledge of the underlying system dynamics. This chapter indulges in one of the most prevalent and frequently used algorithms for the parameter and the state estimation of the nonlinear systems - The Unscented Kalman Filter (UKF). This chapter discusses a method that utilizes UKF to simultaneously estimate the parameters and the states of the LTI system. The final section compares the results from the UKF algorithm with the Kernel-based RRLS algorithm.

5.1 Unscented Kalman filter

The Kalman filter is an optimal tool for predicting and updating the future behaviour in a linear system. When we gravitate toward nonlinear systems, it is necessary to address the assumptions of the Kalman filter. The Kalman always works with linear systems. So, we use the Taylor series to get a linear approximation of a nonlinear system. This approach is known as Extended Kalman Filter (EKF). While the EKF is a standard technique employed widely for estimation of the states and parameters of the system, there are some drawbacks to this method. In EKF, the states of the system are approximated using a Gaussian Random Variable (GRV) and propagated during the first-order Taylor series linearization of the nonlinear systems. Since the EKF omits the higher-order terms during the linearization process, the estimation accuracy suffers in highly nonlinear systems. Instead, Unscented Transformation

(UT) is used as an alternative to the first-order linearization of the nonlinear systems used by EKF. The Unscented Transformation method propagates mean and covariance information through nonlinear transformations. It is more accurate, easier to implement, and uses the same order of calculations as linearization[29].
The UT is founded on the intuition that it is easier to approximate a probability distribution than it is to approximate an arbitrary nonlinear function or transformation. Instead of linearizing at a single point, multitude points, known as *sigma points* are considered and when the nonlinear function is applied to each point, it yields a cloud of transformed points. The statistics of the transformed points can then be calculated to form an estimate of the nonlinearly transformed mean and covariance[30]. This process is called the Unscented Transform and the working of UKF is explained in the next section.

5.1.1 The UKF Algorithm [11, 31, 32]

Unscented Transform

1. **Selection of sigma points**

   The state vector, $\mathrm{x_k}$ (dimension of state space n) is propagated through the nonlinear process model $f()$, having a mean $\bar{x}_k$ and covariance P_k.

   Let M_k be a matrix of $2n+1$ sigma vectors $\mathrm{m_{i,k}}$ (with corresponding weights $w^m_{i,k}$ (mean) and $w^c_{i,k}$ (covariance)). The subscript k is dropped for readability.

$$w^m = \begin{bmatrix} w^m_0 & w^m_1 & \dots & w^m_{2n} \end{bmatrix} \tag{5.1}$$

$$w^c = \begin{bmatrix} w^c_0 & w^c_1 & \dots & w^c_{2n} \end{bmatrix} \tag{5.2}$$

$$M = \begin{bmatrix} m_{0,0} & m_{0,1} & \dots & m_{0,n-1} \\ m_{1,0} & m_{1,1} & \dots & m_{1,n-1} \\ & & \vdots & \\ m_{2n,0} & m_{2n,1} & \dots & m_{2n,n-1} \end{bmatrix} \tag{5.3}$$

2. **Sigma point computation**

Let the first sigma point be the mean of the input, (m_0). We have,

$$m_0 = \bar{x}_k \tag{5.4}$$

For notational convenience, we define $\lambda = \alpha^2(n + \kappa) - n$ where λ is a scaling factor, α determines the spread of the sigma points and κ is a secondary scaling parameter. The remaining sigma points are computed as follows:

$$m_i = \begin{cases} \bar{x}_k + [\sqrt{(n+\lambda)P_k}]_j, & \text{for } j = 1, ..., n \\ \bar{x}_k - [\sqrt{(n+\lambda)P_k}]_{j-n}, & \text{for } j = n+1, ..., 2n \end{cases} \tag{5.5}$$

The j subscript selects the j^{th} row or column of the matrix, i.e. we scale the covariance matrix by a constant, take its square rooted and ensure its symmetry by adding and subtracting it from the mean.

3. **Square root of matrix**

The square root matrix of the posterior covariance matrix ($P_k = S_k S_k^T$) is needed to construct a fresh set of sigma points. This definition is favored because S_k is computed using the *Cholesky decomposition*. It decomposes a Hermitian, positive definite matrix into a triangular matrix and its conjugate transpose.

4. **Weight computation**

The formulation uses one set of weights for the means and another set for the covariances. The weight for the mean and covariance of m_0 is

$$w_0^m = \frac{\lambda}{n+\lambda} \tag{5.6}$$

$$w_0^c = \frac{\lambda}{n+\lambda} + 1 - \alpha^2 + \beta \tag{5.7}$$

where β is used to incorporate prior knowledge of distribution and is set to 2 for Gaussian distribution. The weights for the rest of the sigma points m_i are the same for the mean and covariance:

$$w_i^m = w_i^c = \frac{1}{2(n+\lambda)}, \text{ for } i = 1, ..., 2n \tag{5.8}$$

Now, consider the following nonlinear system, described by the difference equation and the observation model with additive noise:

$$x_{k+1} = f(x_k, u_k) + w_k \tag{5.9}$$

$$y_k = h(x_k) + v_k \tag{5.10}$$

The initial state x_0 is a random vector with known mean $\bar{x}_0 = E[x_0]$ and covariance $P_0 = E[(x_0 - \bar{x}_0)(x_0 - \bar{x}_0)^T]$.

Predict Step

The UKF's predict step computes the prior using the process model $f()$, which is assumed to be nonlinear. Sigma points M_{k-1} and their corresponding weights w^m, w^c are generated and each sigma point is passed through $f(x, \Delta t)$. This projects the sigma points forward in time according to the process model, forming the new prior, which is a set of sigma points.

For k $\in$ [1,2,...,∞), the sigma points are:

$$M_{k-1} = [\bar{x}_{k-1} \quad \bar{x}_{k-1} \pm \sqrt{(n+\lambda)P_{k-1}}] \tag{5.11}$$

$$M_k = f(M_{k-1}) \tag{5.12}$$

The transformed points are used to compute the mean and covariance of the prior/forecast value.

$$\bar{x}_k^- = \sum_{i=0}^{2n} w_i^m m_{i,k} \tag{5.13}$$

$$\bar{P}_k^- = \sum_{i=0}^{2n} w_i^c (m_{i,k} - \bar{x}_k^-)(m_{i,k} - \bar{x}_k^-)^T + Q_{k-1} \tag{5.14}$$

Update Step

Kalman filters perform the update in the measurement space. So, we must convert the sigma points of the prior into measurements using observation model:

$$y_{i,k-1} = h(m_{i,k-1}) \tag{5.15}$$

With the resulted transformed observations, we compute the mean and covariances for these points. The **y** subscript denotes that these are the mean and covariance of the measurement sigma points.

$$\bar{y_{k-1}^{-}} = \sum_{i=0}^{2n} w_i^m y_{k-1} \tag{5.16}$$

$$\bar{P}_{y_{k-1}}^{-} = \sum_{i=0}^{2n} w_i^c (y_{i,k-1} - \bar{y_{k-1}^{-}})(y_{i,k-1} - \bar{y_{k-1}^{-}})^T + R_k \tag{5.17}$$

To compute the Kalman gain, the cross covariance of the state and the measurements are computed first:

$$P_{x_k,y_{k-1}} = \sum_{i=0}^{2n} w_i^c (m_{i,k} - \bar{x_k^{-}})(y_{i,k-1} - \bar{y_{k-1}^{-}})^T \tag{5.18}$$

Next, the residual and Kalman gain can be computed.

$$K_k = P_{x_k,y_{k-1}} (\bar{P}_{y_{k-1}}^{-})^{-1} \tag{5.19}$$

Eventually, the new state state estimate can be computed from the residual Kalman gain as follows:

$$\bar{x}_k = \bar{x_k^{-}} + K_k(\bar{y}_k - \bar{y_{k-1}^{-}}) \tag{5.20}$$

and the posterior covariance is computed as

$$P_k = \bar{P}_k^{-} - K_k \bar{P}_{y_{k-1}}^{-} K_k^T \tag{5.21}$$

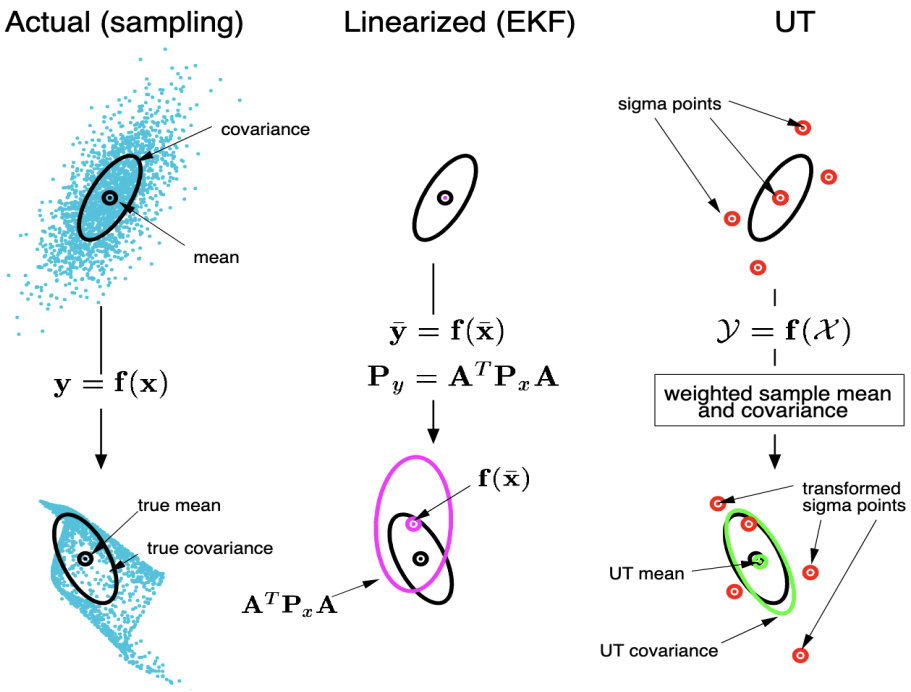

Fig. 5.1 Operation of an Unscented Kalman filter[33]

5.2 Joint parameter and state estimation [11]

The Unscented Kalman Filter(UKF) cannot be used directly for the state estimation when knowledge of the system is not available a priori. To estimate the parameters and the states of the system simultaneously, the parameters and the state vector of a n^{th} order linear system is augmented to form a new aggregated state vector. The general form of this state vector is given by the following matrix equation:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{a}_0 \\ \dot{a}_1 \\ \dot{a}_2 \\ \dot{a}_3 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ -a_0x_1 - a_1x_2 - a_2x_3 - a_3x_4 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}; \qquad y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{5.22}$$

This augmented parameter and state matrix must be treated as an extended n^{th} order nonlinear system. To estimate the states of this augmented nonlinear system, we employ UKF along with the RTS smoother, the Unscented RTS smoother(URTS)[34, 35]. Both the states and the parameters of the original LTI system could be then extracted from the output of the URTS algorithm.
The following section illustrates the results from the URTS algorithm.

5.3 Results

In this section, we will see the performance of the same fourth order linear system we used in the chapter 3 and chapter 4.

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & -5 & -5 & 0 \end{bmatrix} x \quad ; y = x_1 \quad ; \quad x(0) = [0, 0, 0, 1] \tag{5.23}$$

with its corresponding characteristic equation

$$y^{(4)}(t) + 0y^{(3)}(t) + 5y^{(2)}(t) + 5y^{(1)}(t) + 1y(t) = 0 \tag{5.24}$$

The poles of this system are $0.4562 \pm 2.334i, -0.633, -0.279$. This makes the system unstable. The output of the system is superimposed with Additive White Gaussian Noise (AWGN) for different Signal to Noise Ratio (SNR) and the accuracy of state estimation is compared.

5.3.1 Parameter estimation for various noise levels

The parameter estimation for various noise levels are compared with the results deduced using RRLS algorithm and Unscented Kalman Filter. Three noise levels, 0dB, 4.75dB and -9.5dB are considered. From this table, we could infer that the performance of parameter estimation using the RRLS algorithm is superior to that of Unscented Kalman Filter.

STD (σ)	**SNR (dB)**	**Method**	a_0	a_1	a_2	a_3
-	-	True Value	1	5	5	0
0	0.00	RRLS	1.00	4.99	5.00	0
		UKF	1.36	7.04	4.73	0.42
0.5	-4.75	RRLS	1.02	4.42	5.02	0.08
		UKF	-4.11	16.44	2.18	1.52
1	-9.5	RRLS	0.62	5.17	5.05	0.05
		UKF	6.48	5.63	6.04	0.61

Table 5.1: Comparison of the estimated parameter values values using Unscented Kalman Filter and RRLS algorithm for various noise levels (5.23)

5.3.2 State estimation for various noise levels

The estimated output signal and their derivatives are compared with the true signal. From the graphs, it could be inferred that the performance of Unscented Kalman Filter is on par with that of the results obtained from the RRLS algorithm, when there is no noise involved. But the true significance of the RRLS algorithm is clearly visible in the presence of noise. From the graphs, it is evident that the states and their derivatives of the system fall astray as the noise in the output signal increases.

Noise level $\sigma = 0$, SNR = 0dB

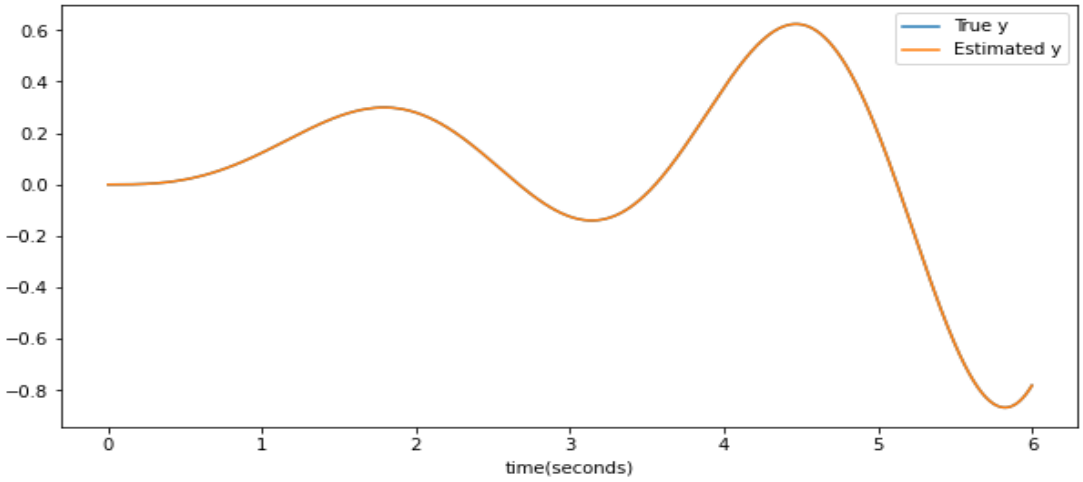

Fig. 5.2 Estimated Output $y(t)$

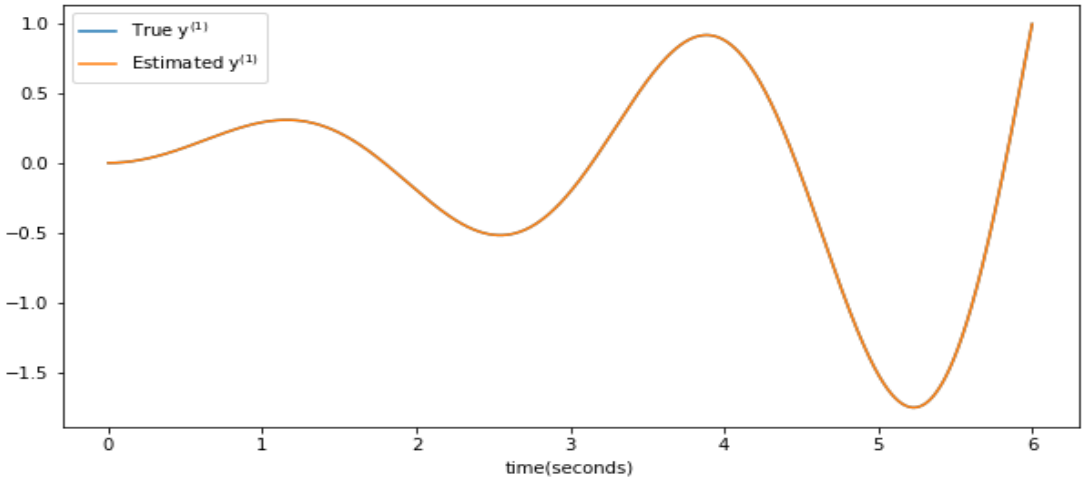

Fig. 5.3 Estimated Output $y^1(t)$

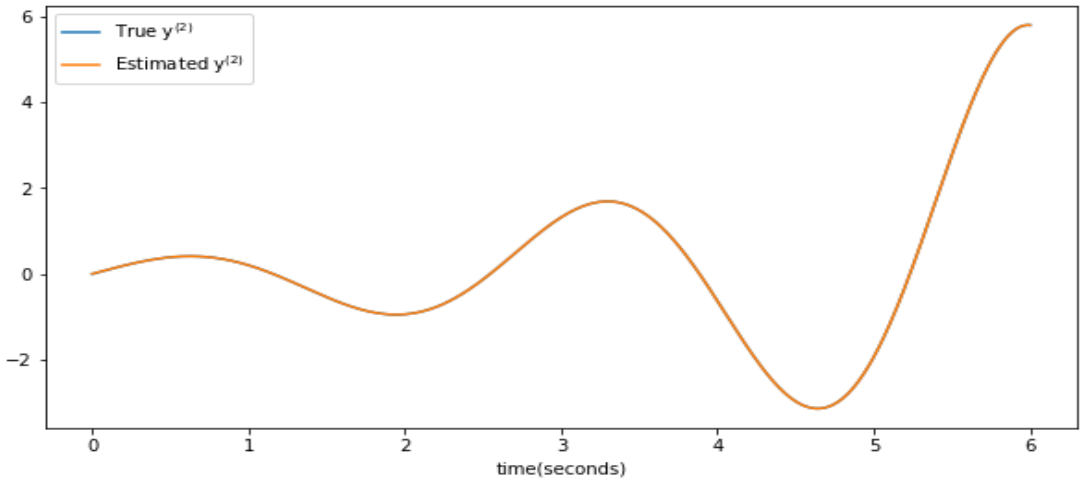

Fig. 5.4 Estimated Output $y^2(t)$

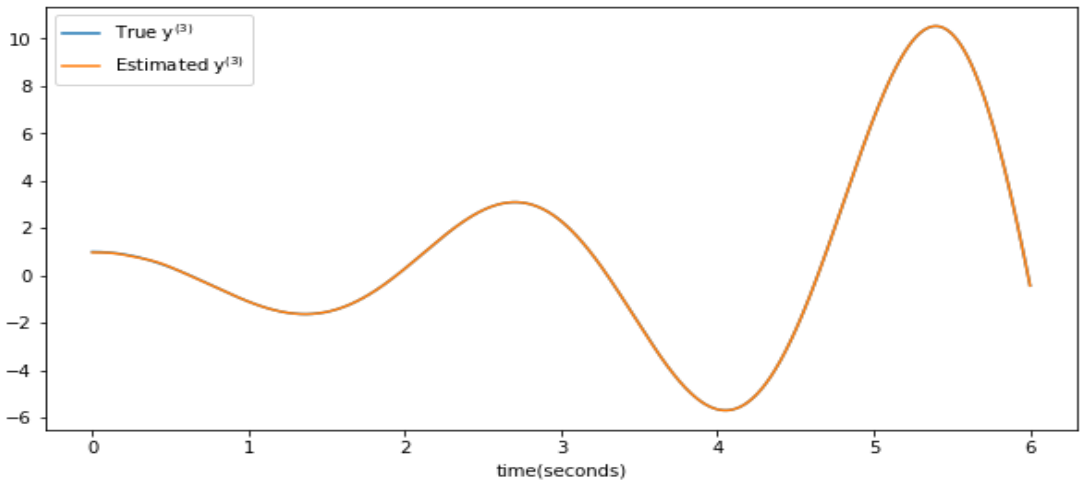

Fig. 5.5 Estimated Output $y^3(t)$

Noise level $\sigma = 0.5$, SNR = -4.75dB

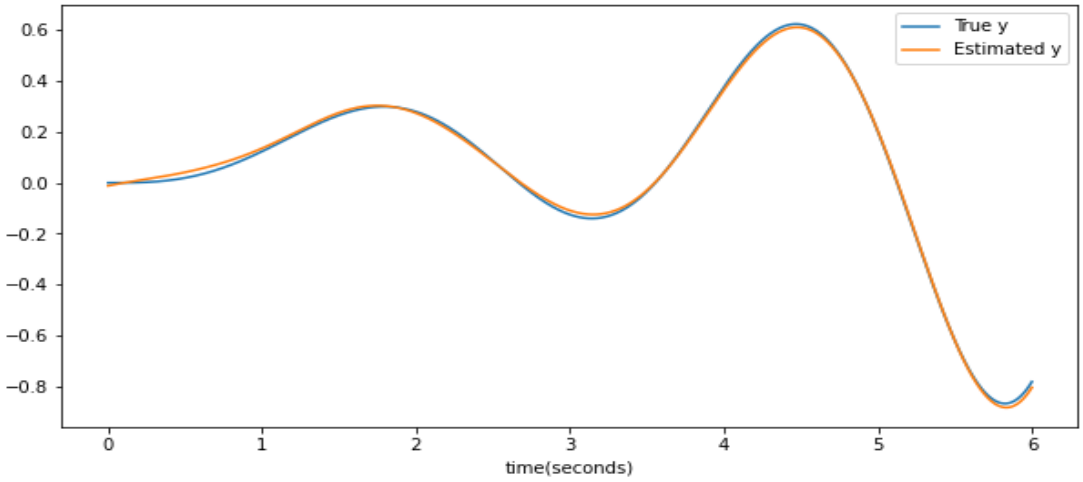

Fig. 5.6 Estimated Output $y(t)$

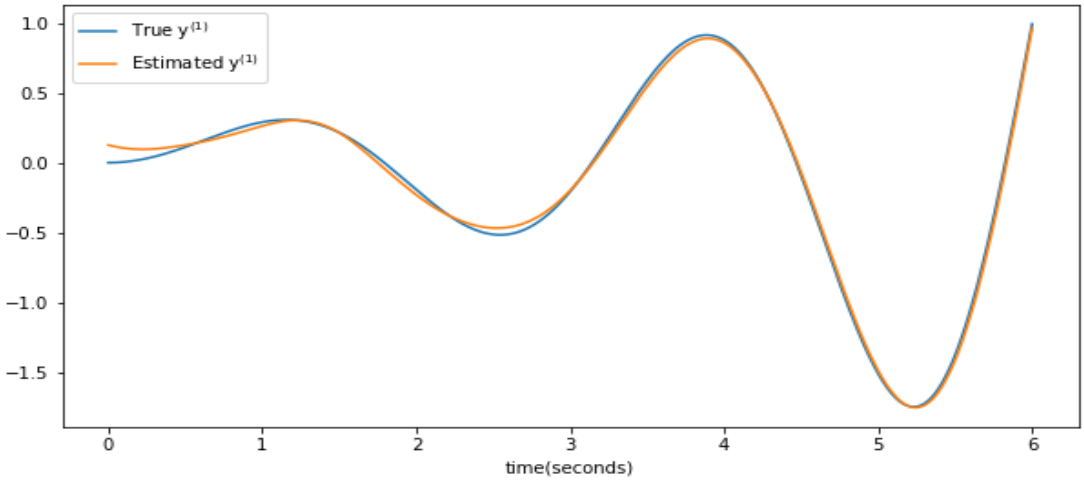

Fig. 5.7 Estimated Output $y^1(t)$

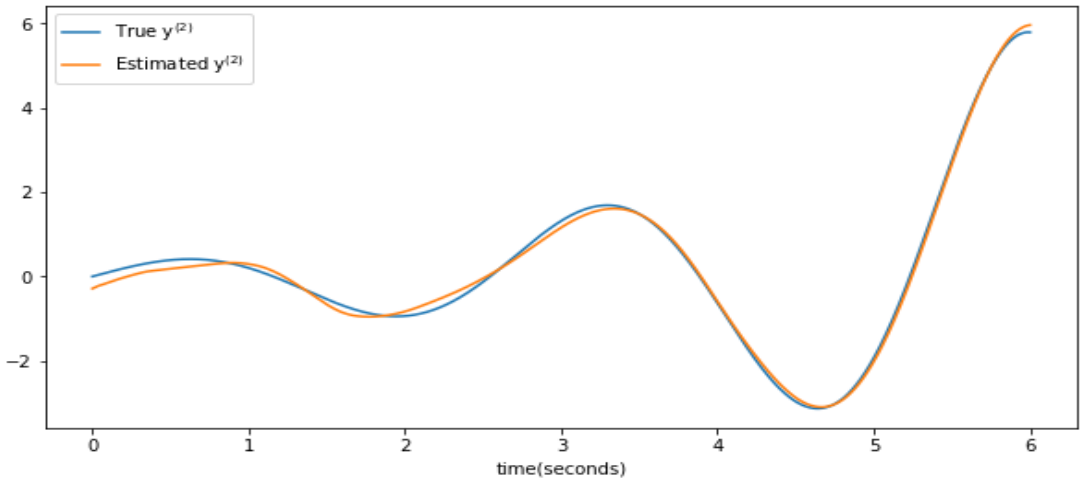

Fig. 5.8 Estimated Output $y^2(t)$

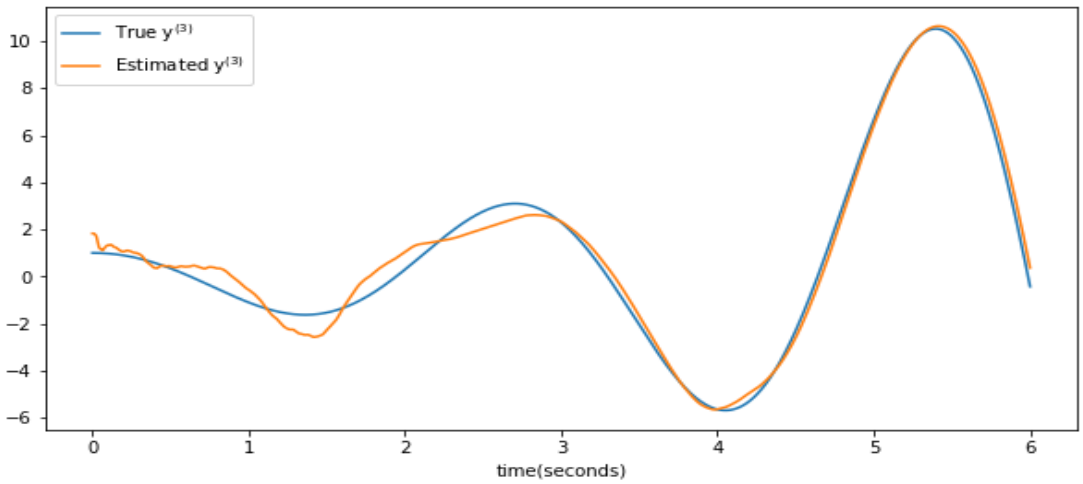

Fig. 5.9 Estimated Output $y^3(t)$

Noise level $\sigma = 1$, SNR = -9.5dB

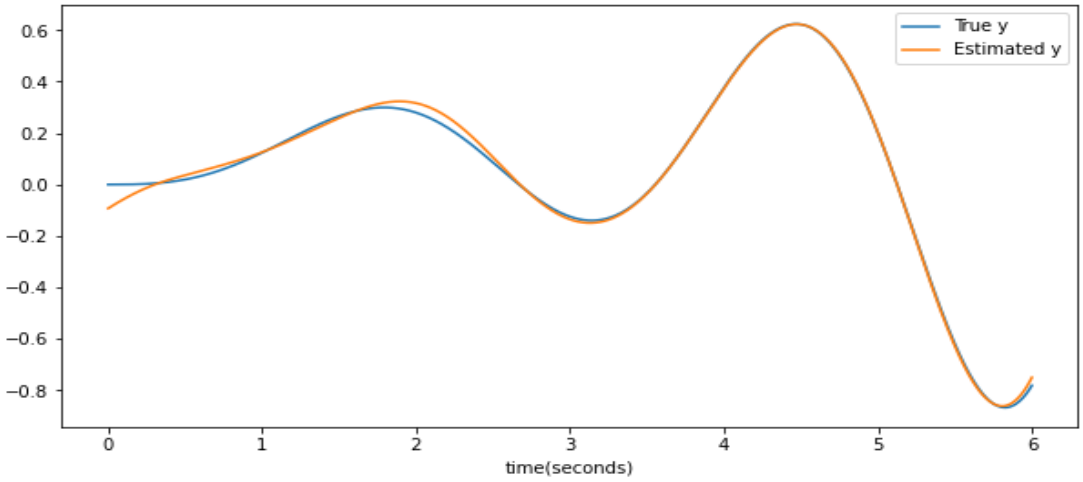

Fig. 5.10 Estimated Output $y(t)$

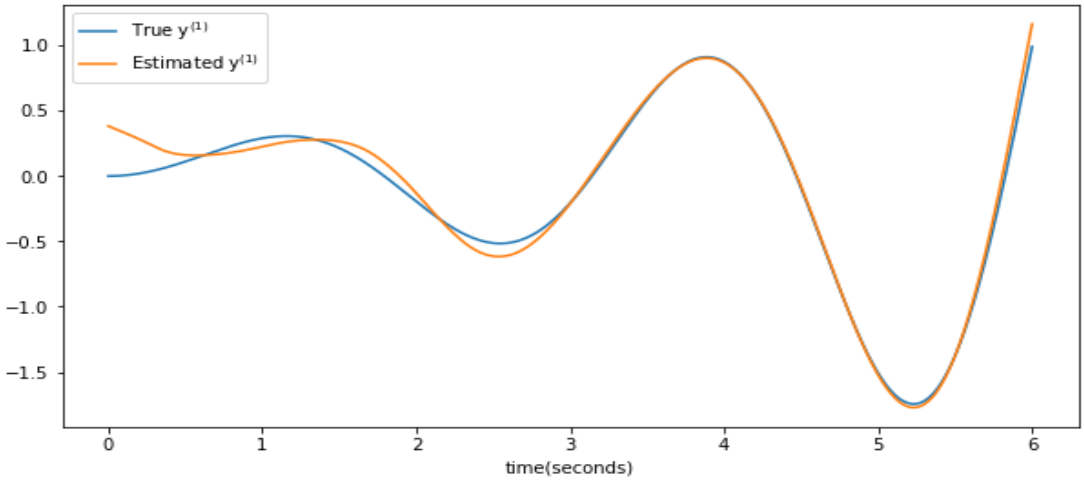

Fig. 5.11 Estimated Output $y^1(t)$

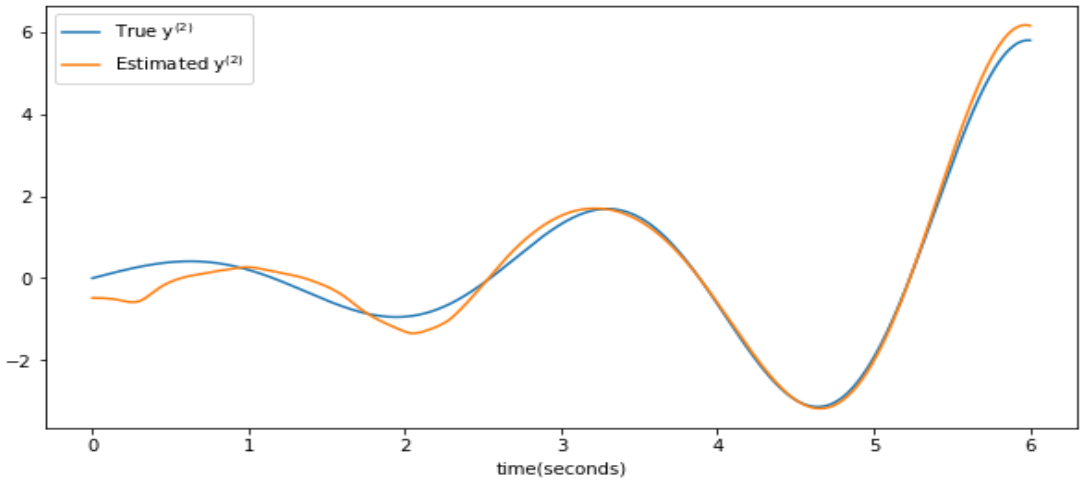

Fig. 5.12 Estimated Output $y^2(t)$

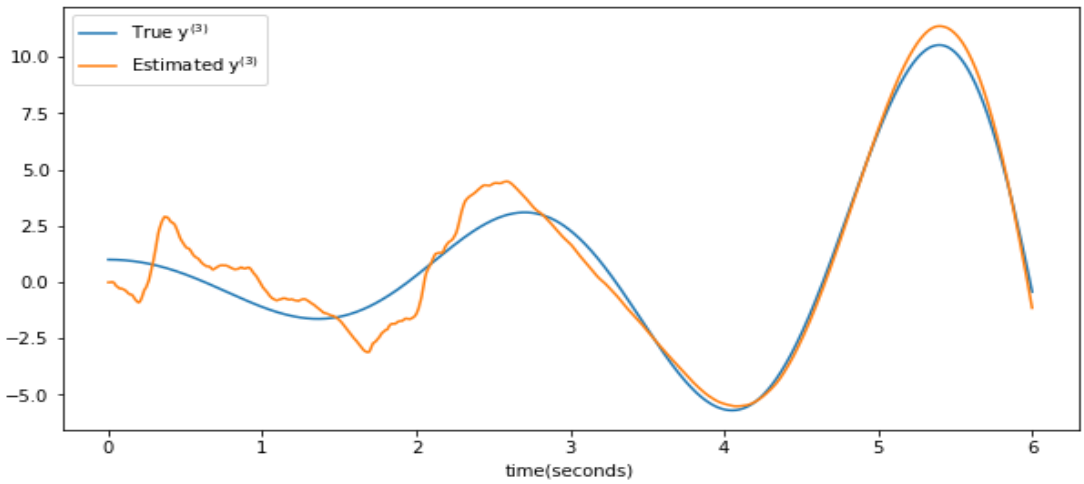

Fig. 5.13 Estimated Output $y^3(t)$

Chapter 6

Kernel based Parameter and State Estimation Toolkits

The significance of parameter and state estimation has been emphasized thoroughly in this thesis. The kernel-based and the Kalman-based strategies for the system estimation are discussed in fine detail from chapter 2 to chapter 5. As discussed in chapters 3 and 4, the Kernel-based RRLS algorithm exhibit superior performance in parameter estimation, when the output signal consists of a high Signal to Noise Ratio (SNR). The success of this algorithm is a tribute to the years of conscientious research carried out by professor Dr Hannah Michalska and the graduate students under her supervision. To propel the research forward and also make this research useful for the general public, it is imperative to standardize the algorithms. For this purpose, the implementation of these algorithms must be cohesive and in a modular fashion. It ensures reproducibility, improvisation, and easy understanding of the algorithms. The conception of the Python-based Estimation Toolkits (PETs) library[36] is one such attempt by my research partner Nithilasaravanan Kuppan and myself, with valuable insights from Dr Hannah Michalska. PETs is a library hosted on GitHub, containing modules that execute these algorithms and provide a user-friendly interface. The repository, in conjunction with this thesis, would prove to be an effective way to understand the algorithm, the codes, the logic, and the implementation of a novel as well as a traditional method of the parameter and the state estimation. This chapter was co-authored by Nithilasaravanan Kuppan; see [14]

The following sections illustrate the organization of the library, the working of each module, and the method for executing the algorithms. This library is thoroughly documented, with

its implementation explained with an example.

6.1 Requirements and Assumptions

The following assumptions must be adhered to to ensure the proper function of this library.

- The parameter matrix of the system is assumed to be represented in a controllable canonical form. It is an $n \times n$ square matrix with the last row containing the coefficients of the characteristic equation of the homogeneous SISO LTI system.
- The noise present in the signal is assumed to be an Additive White Gaussian Noise (AWGN).

In addition to the assumptions stated above, the following requirements must comply.

- This library requires both the noisy signal (which emulate the practical behaviour of a measuring device) and the true signal, in order to compare the performance with different algorithms and hyper-parameters and to plot all the graphs. The following sections discuss the exact functions associated with these tasks in detail.
- This library is developed completely in python and a python version of 3.8 or above is recommended for the proper function of all the modules and their dependent libraries. All the dependent external libraries are completely open-source and readily available for download. All the dependent external libraries is listed in the GitHub repository under `/PETS/requirements.txt` which then could be installed using the console, using the popular pip command

  ```
  pip install requirements.txt
  ```

6.2 Process flow

This section describes the process flow associated with this library. It provides the complete information necessary for the successful execution of the library. A summary of this section is illustrated in the image 6.1.

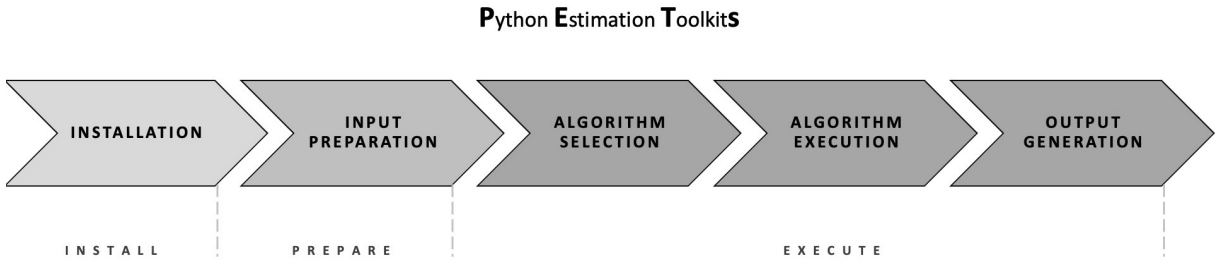

Fig. 6.1 Process flow for the PETs repository - Install, Prepare and Execute

Installation

After cloning the repository to the local system, it is necessary to ensure that all the supporting libraries mentioned in *requirements.txt* are installed. The packages like Pandas, Numpy, Matplotlib, Sklearn, and Scipy provide useful tools to run user-friendly and efficient code. After installation, based on how the user decides to run the library, it is necessary to add the environment variables for python, and the dependent packages to the working environment.

Input preparation

The library requires two primary inputs from the user. One is the measured output of a system, a potential noisy signal which is used to estimate the parameters and state of the system. The information must be provided to the function *noisy_signal* which will return a 1-dimensional array containing the data. For research purposes, like testing a new algorithm, the user can simulate noisy data from within the function. Otherwise, the data can be imported from a file. In addition to this, the user must provide the clean (True) signal, which will be used to calculate the error metrics and plot comparison graphs. It is assumed that both the information are returned as a 1-dimensional NumPy array. This file is located at

```
/PETS/src/pets/noisy_input.py
```

The second file which needs user attention is the configuration file for the intended algorithm. Each algorithm consists of several hyper-parameters that must be tuned depending on the signals used and intended results. This consolidated information is in a config file in JSON format. It makes the configuration user friendly, and the user could directly open the file in a text editor and make the intended changes. This file is located at

```
/PETS/configs/config_<algorithm>.json
```

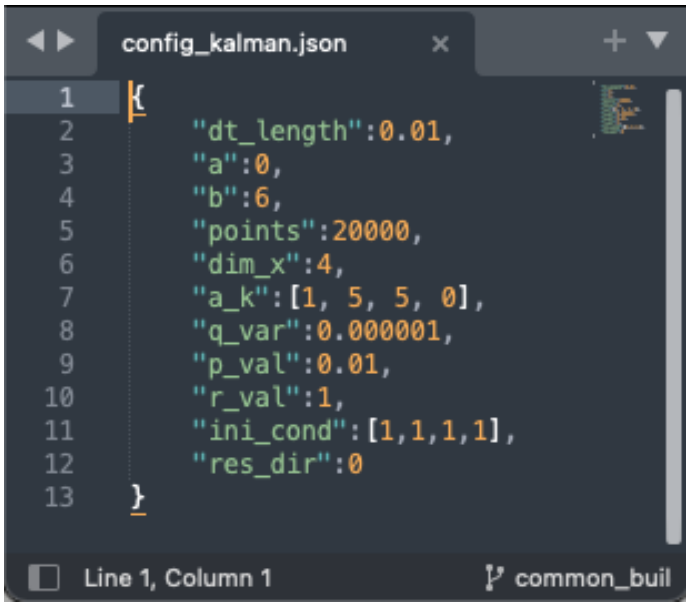

Fig. 6.2 An example for a config file

The above image depicts what a typical config file looks like. Every key in this JSON dictionary represents a hyper-parameter that can be configured by the user. Every algorithm will have one such key associated with it.

Algorithm selection

After the input preparation is complete, the user is ready to run the library with one of the supported algorithms. The list of algorithms supported is four while writing this thesis. The list of algorithms supported can be retrieved using the following command from the Command Line Interface (CLI).

```
python3 /PETS/scripts/run_estimation.py -h
```

This command will list all the algorithms supported with that particular version of the

downloaded library. The developer is expected to update the list while including a new algorithm or modifying an already existing one.

The desired algorithm can be executed using the command

```
python3 PETS/scripts/run_estimation.py -m <algorithm>
```

This command executes the script and invokes the relevant modules based on the selected algorithm. The duration of execution depends on the algorithm and configured hyper-parameters.

Algorithm execution

A brief summary of the supported algorithms is provided below.

kernel_projection: This algorithm utilizes the forward-backward double-sided kernels and the RRLS algorithm for estimating the parameters of SISO LTI systems to estimate the system parameters and uses the projection method to reconstruct the system states and their derivatives.

kernel_kalman: This algorithm also utilizes the forward-backward double-sided kernels and the RRLS algorithm for estimating the parameters of SISO LTI systems and the Kalman filter with the RTS smoother for reconstructing the state and its derivatives.

kalman_statesonly: This algorithm is used when the parameters of the system are already available. This algorithm utilizes the Kalman filter with RTS smoother to estimate the system states and their derivatives. The parameters of the system is taken from the `config_kalman_statesonly.json` file.

kalman_ukf: This algorithm utilizes Unscented Kalman Filter (UKF) with RTS smoother to estimate both the system parameters and the states (along with its derivatives) of SISO LTI systems.

The primary script `run_estimation.py` can be found under the `PETS/scripts/` folder. All the source files necessary to run the algorithm are present in `PETS/src/pets` folder. The folder structure for the PETS library is illustrated in the figure 6.3.

```
PETS/
├── configs/
│   ├── config_<algorithm>.JSON
│   ├── README.md
├── data/
├── resources/
│   ├── reference thesis/
│   └── other resources/
├── scripts/
│   ├── run_estimation.py
│   └── README.md
├── src/
│   ├── pets/
│   │   ├── noisy_input.py
│   │   ├── gen_results.py
├── requirements.txt
├── README.md
```

Fig. 6.3 Folder structure - PETs repository

Output generation

Output for the parameter and the state estimation algorithms is generated automatically when the `run_estimation.py` script is executed. The reconstructed states are generated as a graph, superimposed on the *true signal* for comparison. In addition, the script generates three default error metrics, Root Mean Squared Error(RMSE), Mean Absolute Error(MAE), and Mean Absolute Difference(MAD). The location where these results must be saved is a configurable parameter, present in the `config` file of that particular algorithm.

The following are the default output generated by the library.

1. `*_rmse_mad.txt` This file contains the Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and the Maximum Absolute Difference (MAD) that compares the true states and the estimated states and their derivatives.

2. `*.png` These are multiple images containing the graphs, comparing the true and the

estimated states and their derivatives.

3. `state_estimates.tsv` This file contains the log of the complete values of the estimated states and their derivatives. It could be used for further analysis and processing.

4. `parameter_estimates.tsv` It is an output file generated for algorithms that estimate the parameters with the states. The coefficients of the system characteristic equation are present in this file as a 1-dimensional array.

In addition to this, the desired output can be manipulated from the obtained results using the `gen_results.py` script present in `/PETS/scripts/` folder.

6.3 Estimation algorithms

While the previous chapters explained the analytical aspects and results of the algorithms, this section focuses on the way these algorithms are embedded in the library. A summary of the implementation and functioning of the estimation algorithms `kernel_kalman` and `kernel_projection` is given below. The remainder of the two algorithms were developed and implemented by my research partner Nithilasaravanan Kuppan and their detailed explanation can be found in [14].

6.3.1 Augmented Kalman and Kernel for parameter and state estimation

```
python run_estimation.py -m kernel_kalman
```

This algorithm utilizes forward-backward double-sided kernels and the RRLS algorithm to compute the parameters of a SISO LTI system of orders up to 4. Ideally, the algorithm could support a linear system of any finite order. But the time restrictions and computational complexity of a typical commercial computer limit the order to 4. The algorithm fetches the information necessary for estimation from `config_kernel_kalman.json` and executes the algorithm. The source code for the estimation algorithm is present in `PETS/src/pets/kernels.py`. Followed by the estimation of parameters, the states of the system are reconstructed using the Kalman filter with an option to include RTS smoother. The hyper-parameters for this filter could be configured in the same configuration file `config_kernel_kalman.json`. The estimated parameters are stored in `*parameter_estimates.txt` file and the reconstructed states, with their derivatives, are stored in `*state_estimates.tsv` file. In addition to this, these states are plotted as a graph superimposed on the actual states. These graphs are stored as image files.

A summary of the configuration parameters are given below.
`knots` - This parameter specifies the knots (data-points) upto which the forward and backward integration happens.
`S_type` - Covariance mattix algorithm. `full` and `diagonal` are the supported algorithms right now.
`tolerance` - This parameter allows the user to assign the tolerance for the convergence of system parameters.
`w_delta` - Initial weight factor for measurement in prediction vs measurement trade-off.
`a,b,points` - `a` and `b` represent the boundaries of the finite time interval within which the estimation takes place. `points` represents the total number of samples within the finite time interval.
`order` - The order of the SISO LTI system.
`q_var,p_var,r_var` - Process noise variance, Covariance multiple(scalar number multiplied with the identity matrix of specified order) and Measurement noise variance.
`init_cond` - Initial state values for Kalman filter.
`res_dir` - Destination folder to save all the results.

6.3.2 Kernel and projection based parameter and state estimation

```
python run_estimation.py -m kernel_projection
```

This algorithm utilizes the same forward-backward double-sided kernel to compute the parameters of a SISO LTI system. This algorithm fetches the information necessary for estimation from `config_kernel_projection.json` and executes the algorithm. The source code for the estimation algorithm can be found in `PETS/src/pets/kernels.py` file. Instead of the popularly available Kalman filter, this algorithm uses a projection to reconstruct the state and their derivatives. The output from this algorithm is similar to the previous one, where the estimated parameters are stored in `*_parameter_estimate.txt` file and the reconstructed states and their derivatives are stored in `*_state_estimate.tsv` file. The graphs containing the estimated states and the actual states are stored as images.
This algorithm has the same configuration parameters listed in Augmented Kernel and Kalman filter algorithm. The users can refer to 6.3.1 for the information regarding each configuration parameter. The state estimation is carried out by directly projecting the noisy signal onto the finite-dimensional subspace of RKHS spanned by the fundamental solutions of the characteristic equation of the system. Hence, there are no additional parameters in-

volved in state estimation. Once the states are reconstructed, the output derivatives are reconstructed using 3.61.

6.4 Application and further improvements

The architecture for this library is designed in such a way to ensure versatility. This library can be used for research purposes where the performance of various algorithms is compared against a benchmark system for analysis purposes. Also, this library could serve as a precursor to a control problem since the noise-free realization of the system is pivotal to ensuring efficient control. In addition to these applications, the library allows easy integration of additional algorithms. For instance, this library can be extended to support non-homogeneous systems and non-linear systems in future. By adding the implementation of these algorithms to `PETS/src/pets/` and modifying `PETS/scripts/run_estimation.py`, this library can be expanded without affecting any of the existing functionality. Further, this estimation library can be extended to support control frameworks like PID controller, MPC controller, and LQR controller. This controller could be developed separately and invoked using a script `run_control.py` file. This control implementation could leverage the suitable estimation algorithms depending on the application.
All the supporting documents and previous works which helped develop this library are provided in the `PETS/resources/` folder for reference purposes.

Chapter 7

Future works with state estimation in infinite horizon

The parameter and state estimation techniques for finite horizon SISO LTI systems are discussed extensively in the thesis. Chapter 6 focused on the development of the PETs library, which focused on building a foundation to aid further research and analysis. This chapter addresses one such possible extension for the current library. The finite horizon state estimation and filtering of LTI systems are extended to handle the evolution of a system in the infinite horizon. To control a system, the parameter and the state estimation algorithms must be extended to handle non-homogeneous systems, as the system is subjected to a control input. Further, the state estimation algorithms for a stationary signal are inadequate as estimation should take place online, where there is a continuous stream of measured output signal that has to be filtered. The derivation of kernels and regression equations for non-homogeneous systems, although possible, is beyond the scope of this thesis. This chapter is dedicated to real-time state estimation of linear homogeneous systems by continuous filtering of the system states evolving through an infinite horizon without the influence of control input. Another assumption is that an adequate number of samples are available throughout the horizon.

7.1 Moving window estimation for SISO LTI systems

To perform real-time state estimation and filtering, the state estimation algorithm must be modified to accommodate a continuous stream of the measured output signal. The algorithm

must continuously filter the states at discrete windows that move forward in time. The filtering must occur in this window with minimal latency, yield the state values, and the window must move on to the next available set of input samples. It ensures that real-time states are available to further control applications. In addition, it accounts for any sudden perturbations of the system or subjecting the system to a control input.
Let us assume a fixed moving window of length W_L is shifted forward in time. At any window n, the measured signal Y_n in the window is filtered by the Kalman Filter and the RTS smoother.

Then, the complete output signal of the system 2.6 is given by

$$Y_M = \sum_{n=0}^{\infty} Y_n \tag{7.1}$$

where Y_n is the output signal in a single window n and the output signal is assimilated into Y_M such that,

$$Y_M = [Y_0, Y_1, Y_2, \ldots, Y_\infty]$$

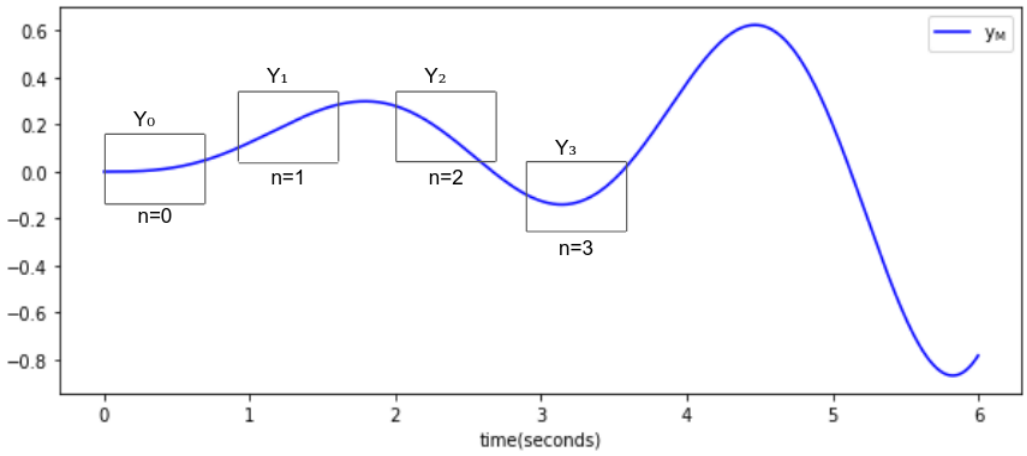

Fig. 7.1 Windows breakdown for moving window state estimation of y_M

The following algorithm illustrates the working of moving window state estimation of LTI systems over infinite horizon.

Algorithm 2 Moving window estimation of states in infinite horizon

Initialize: $a_0 = a_{k_n}$ where a_{k_n} are the parameters of the system of order n
Initialize: $x_{initial} = [y_0;\ y_0^{(1)};\ y_0^{(2)}; ...;\ y_0^{(n)}]$
Procedure Moving Window Kalman Filter + RTS
Initialize: A_k, $P_{initial}, Q, R$ and H matrices
Compute the state transition matrix $F = e^{A_k t}$
for $n = 0, 1, 2, 3, ...,$ **do**
  for $k = 0, 1, ..., N$ **do**
    while $Y_{n_k} \in Y_n$ **do**
      Predict I: $x_k^f = F_{k-1} x_{k-1}$ {State Extrapolation [f for forecast]}
      Predict II: $P_k^f = F_{k-1} P_{k-1} F_{k-1}^T + Q$ {Covariance Extrapolation}
      Set $z_k = Y_{n_k}$
      Update I: $K_k = P_k^f H^T (H P_k^f H^T + R_k^{-1})$ {Kalman Gain}
      Update II: $x_k = x_n^f + K_k (z_k - H x_k)$ {State Update}
      Update III: $P_k = (I - K_n H) P_k^f$ {Covariance Update}
      return x_k, P_k
    end while
  end for
  Append x_k to X and P_k to P {Stored as a batch}
  Procedure Rauch-Tung-Striebel {Fixed Interval Smoothing}
  Initialize $x_{\hat{k}|n} = X[-1]$ and $P_{k|n} = P[-1]$ {Backward filter}
  for $k = N - 1, ..., 1, 0$ **do**
    while $x_k \in X$ **do**
      Calculate $A_k = P_{k|k-1} \bar{F}_k^T P_{k+1|k}^{-1}$ {Smoother gain matrix}
      Calculate $P_{k+1|k} = \bar{F}_k P_{k|k-1} \bar{F}_k^T$ using 4.13 {Error covariance}
      Update: $\hat{x}_{k|n} = \hat{x}_{k|k} + A_k \left(\hat{x}_{k+1|n} - \hat{x}_{k+1|k} \right)$
      return x_k
    end while
  end for
  Append x_k to x_E
end for

7.2 Results

In this section, we will see the performance of the same fourth order linear system used in chapter 3 - 5.

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & -5 & -5 & 0 \end{bmatrix} x \quad ; y = x_1 \quad ; \quad x(0) = [0, 0, 0, 1] \tag{7.2}$$

with its corresponding characteristic equation

$$y^{(4)}(t) + 0y^{(3)}(t) + 5y^{(2)}(t) + 5y^{(1)}(t) + 1y(t) = 0 \tag{7.3}$$

Here, the complete information about the output is not available as a stationary signal. Instead, the output is divided into discrete windows of size $100ms$ and 2000 samples are available in each window. The windows of the output signal arrive sequentially as we move forward in time. In addition, we assume that the complete parameters of the systems are known beforehand. Hence, only the states of the system are estimated using the Kalman filter and RTS smoother.

State estimation for various noise levels

Noise level $\sigma = 0$, SNR = 0dB

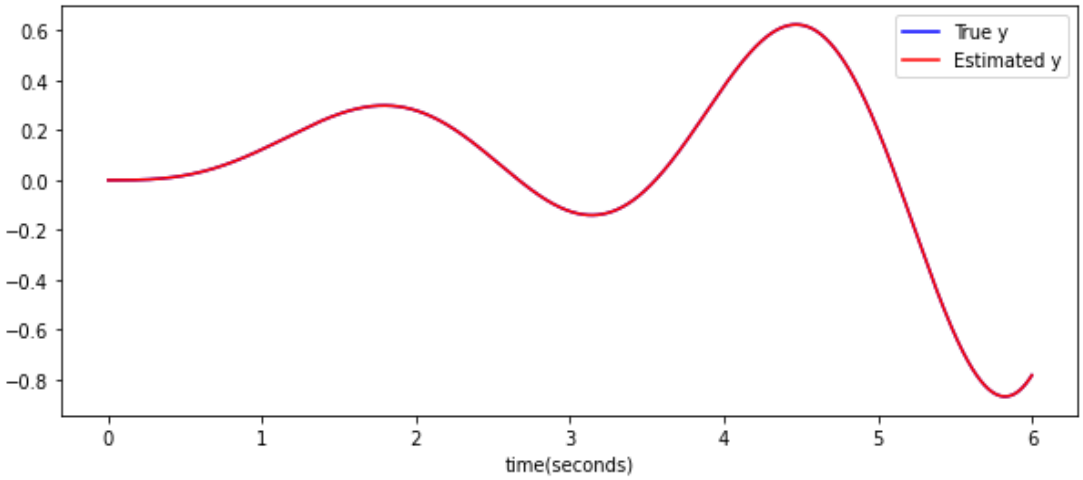

Fig. 7.2 Estimated Output $y(t)$

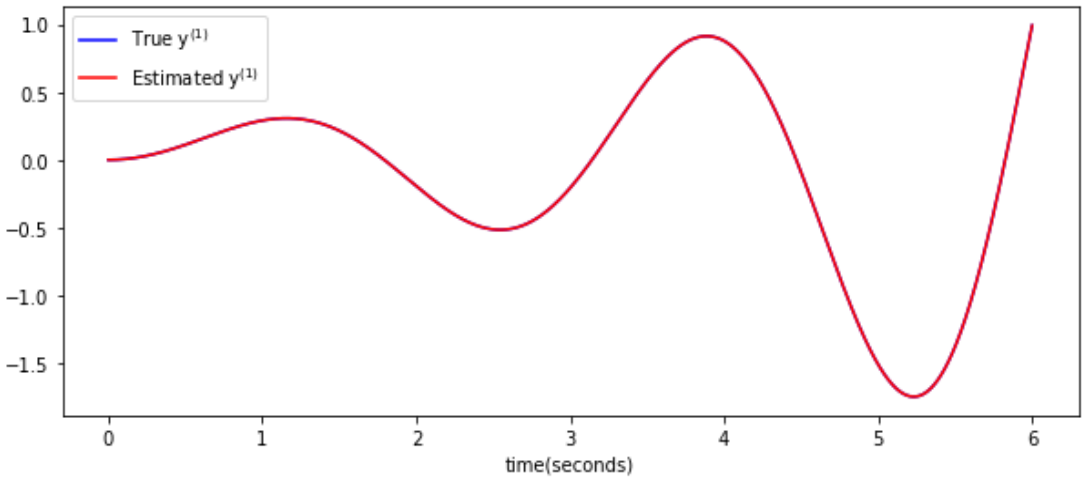

Fig. 7.3 Estimated Output $y^1(t)$

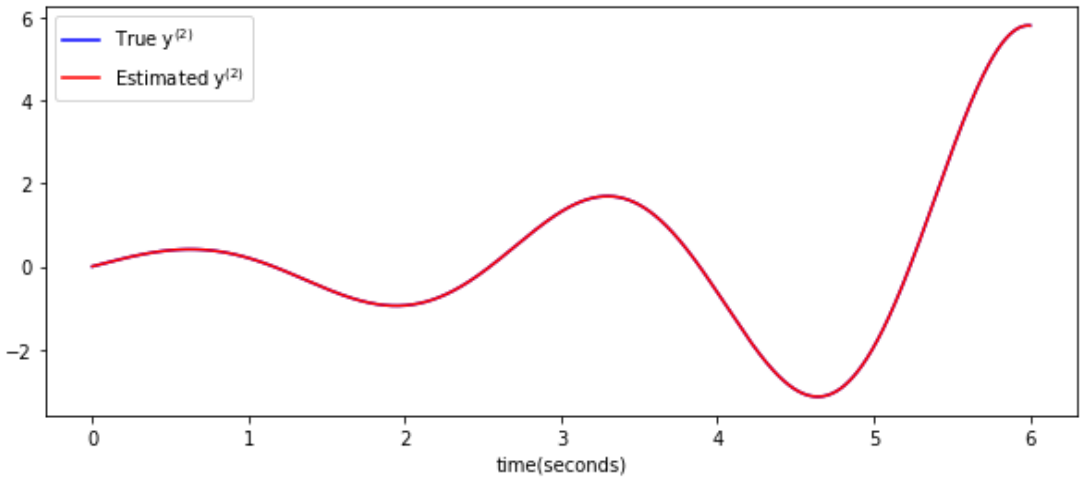

Fig. 7.4 Estimated Output $y^2(t)$

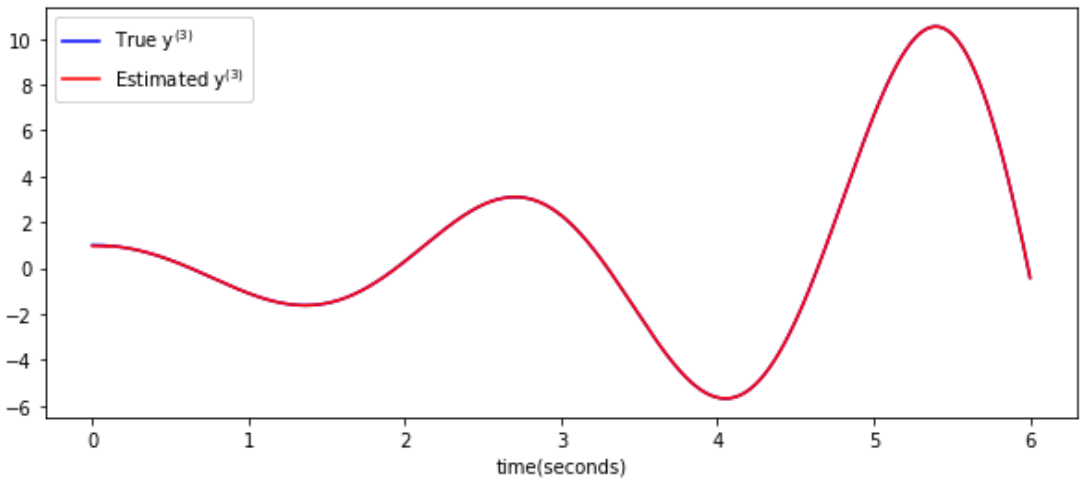

Fig. 7.5 Estimated Output $y^3(t)$

Noise level $\sigma = 0.5$, SNR = -4.75dB

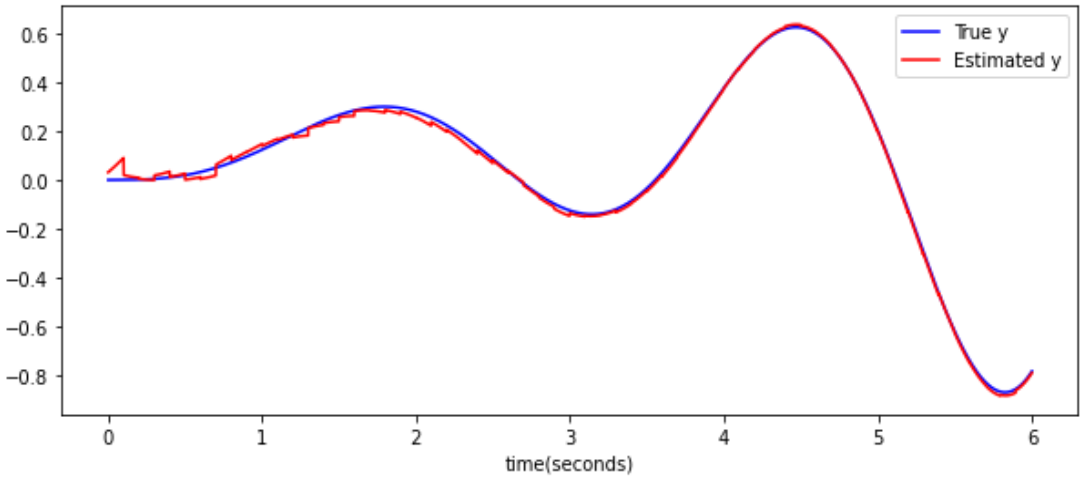

Fig. 7.6 Estimated Output $y(t)$

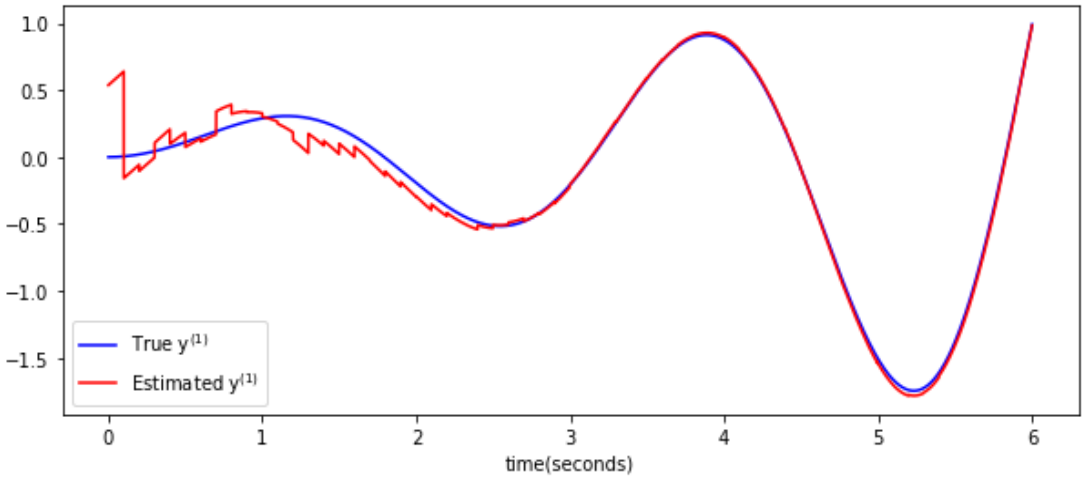

Fig. 7.7 Estimated Output $y^1(t)$

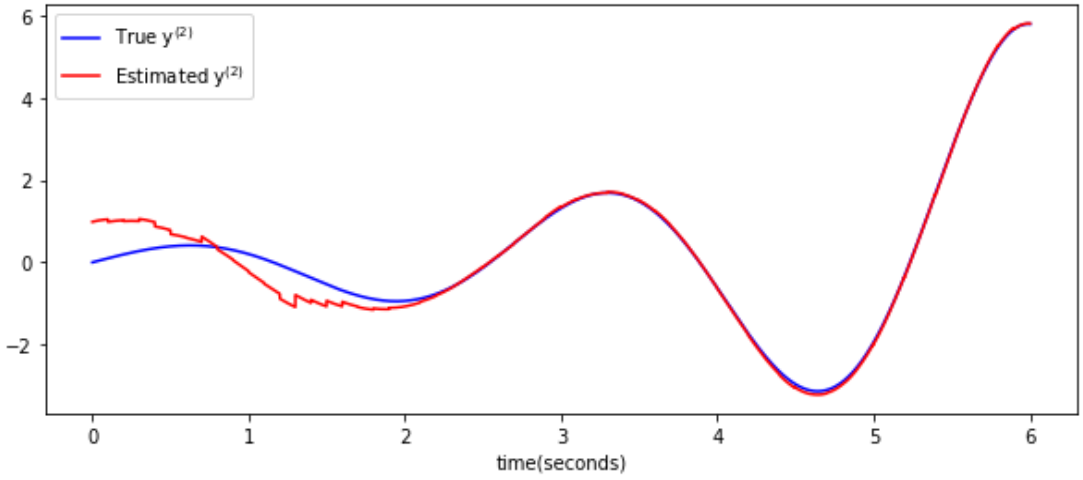

Fig. 7.8 Estimated Output $y^2(t)$

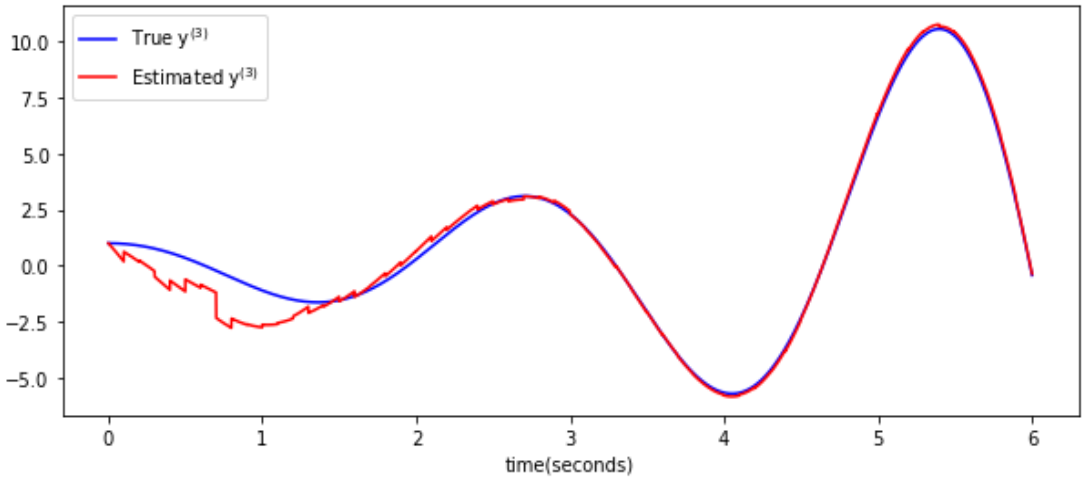

Fig. 7.9 Estimated Output $y^3(t)$

Noise level $\sigma = 1$, SNR = -9.5dB

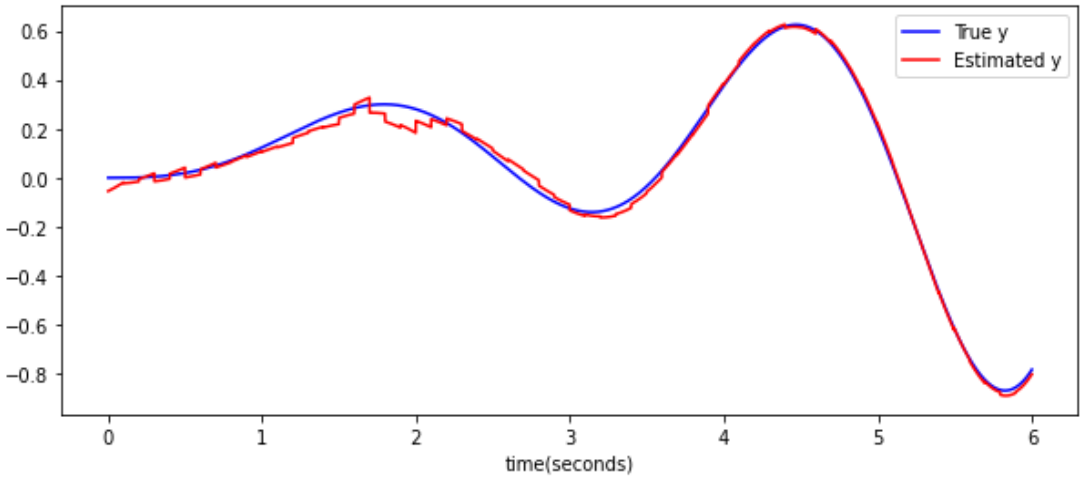

Fig. 7.10 Estimated Output $y(t)$

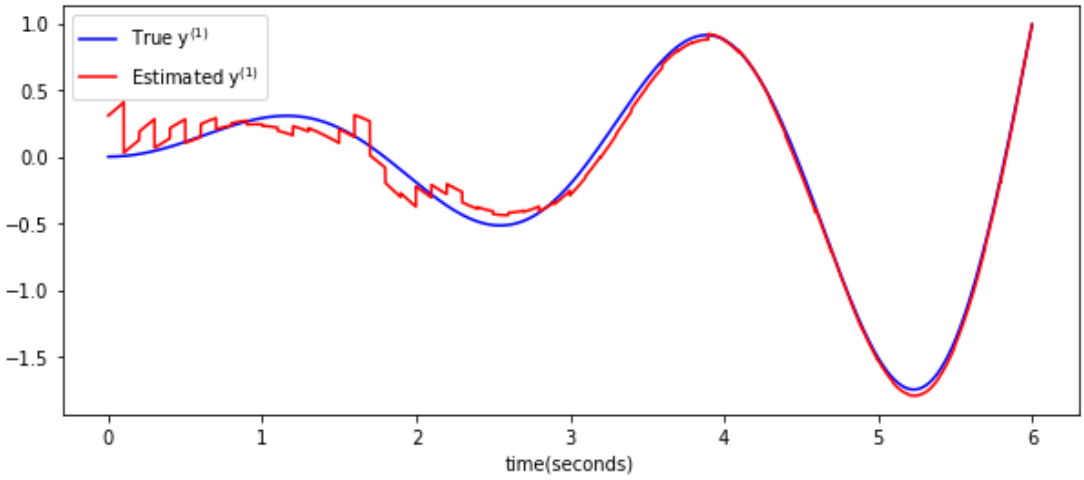

Fig. 7.11 Estimated Output $y^1(t)$

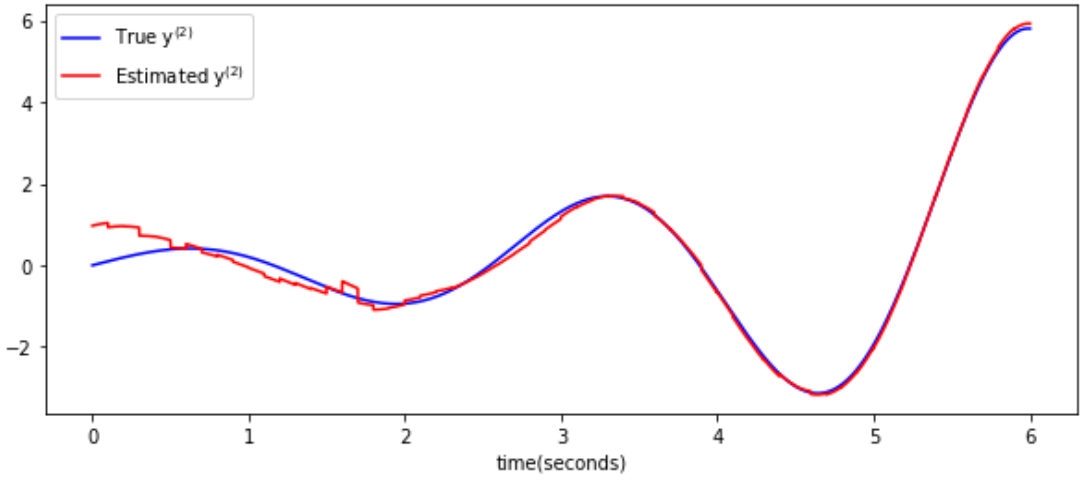

Fig. 7.12 Estimated Output $y^2(t)$

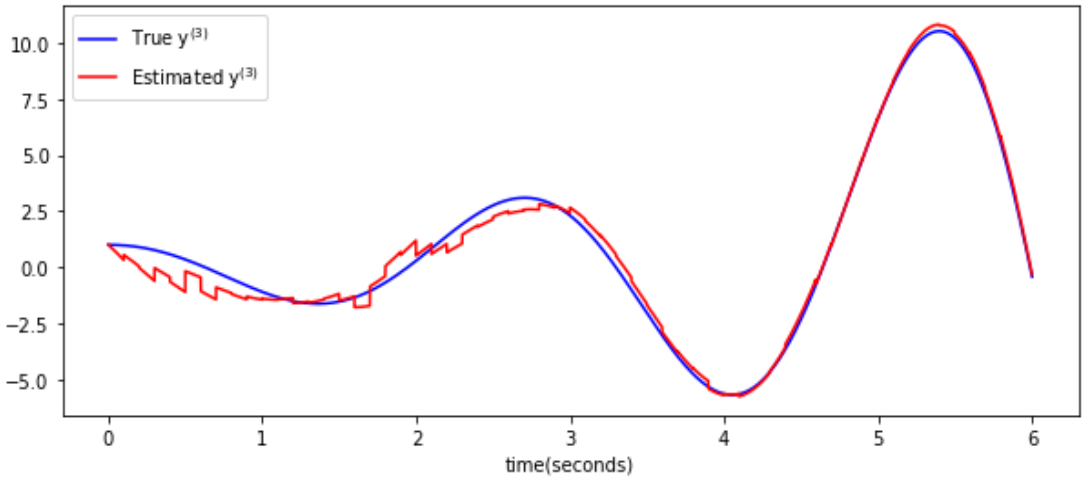

Fig. 7.13 Estimated Output $y^3(t)$

Inference

From the results, it is evident that the performance of the Kalman filter and RTS smoother for state estimation and filtering in the infinite horizon takes a longer time to converge to its true signal. The deviation becomes prominent as the noise in the signal increases. This behaviour is understandable because the filtering algorithm works with a limited window initially, having no information about the future evolution of the system. Also, since the system works under the assumption that the system parameters are accurately known beforehand, we could see a satisfactory performance in filtering. If the estimated parameters deviate from the true parameters in a real-world system, it induces additional errors in estimated states. In future works on online state estimation and filtering, we must ensure to mitigate the error due to inaccurately estimated parameters.

Conclusion

1. The research on the parameter and the state estimation of the SISO LTI system has remained relevant in control theory for a long time. With the recent advancements in the computing technology allowing us to work with data-intensive applications, kernel-based parameter and state estimation algorithms have a huge potential.

2. The construction of the PETs library leverages the research work pursued by Professor Dr Hannah Michalska and the team. This library provides a foundation to progress the research. The library being open-sourced, it is available for the general public to aid numerous applications like analyzing a system or building a control system.

3. The advantage of the kernel-based RRLS algorithm is its robustness to high noise levels in the measured output. From the results shown in this thesis, it is evident that the performance of the RRLS algorithm is superior to the conventional UKF based parameter estimation and filtering for high noise levels.

4. The major drawback of this method is the assumption that infinite samples of measured outputs are available. In the real world, this might not be possible due to the quality of the sensors, economics and logistics. We should address the compensation for the RRLS algorithm to work with fewer samples and missing windows in future works.

5. Another drawback is that the kernels expect the parameters of the characteristic equation in controllable canonical form. It is relevant while the objective is to control the systems. But there are systems for which the objective of the parameter and state estimation is not always to control the system, but just to study and analyze the systems. And it is not possible to represent all systems in their controllable and observable canonical form.

References

[1] Hebertt Sira-Ramírez. *Algebraic Identification and Estimation Methods in Feedback Control Systems*. John Wiley & Sons, 2014.

[2] Carl Friedrich Gauss and Charles Henry Davis. *Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections: A Translation of Gauss's Theoria Motus with an Appendix*. Little, Brown, 1857.

[3] John Aldrich. R.A. Fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12(3):162 – 176, 1997.

[4] N. Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series with Engineering Application*. MIT Press: Cambridge, MA, USA,, 1964.

[5] A.N. Kolmogorov. Interpolation and extrapolation of stationary sequences. *Ser Math*, 5:3–14, 1940.

[6] Debarshi Patanjali Ghoshal. *Finite-interval estimation using double-sided kernels and differential invariants*. PhD thesis, McGill University, 2021.

[7] David G Luenberger. Observing the state of a linear system. *IEEE transactions on military electronics*, 8(2):74–80, 1964.

[8] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[9] Herbert E Rauch, F Tung, and Charlotte T Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

[10] Guido van Rossum. Python programming language. `https://www.python.org/`, 2001.

[11] Shantanil Bagchi. Finite-time identification and estimation of non-linear system dynamics using kernel-based sliding window. Master's thesis, McGill University, 2021.

[12] Nikhil Jayam. Moving-horizon and kernel-based extended kalman filter for non-linear estimation. Master's thesis, McGill University, 2021.

[13] Adharsh Mahesh Kumaar. Comparison of different online kernel and kalman filter based estimation and filtering methods for siso and mimo non-linear systems. Master's thesis, McGill University, 2021.

[14] Nithilasaravanan Kuppan. A python toolkit for kernel estimation for siso linear systems. Master's thesis, McGill University, 2022.

[15] Anju John. Estimation for siso lti systems using differential invariance. Master's thesis, McGill University, 2019.

[16] Augustin-Louis Cauchy. *Résumé des leçons données à l'école royale polytechnique sur le calcul infinitésimal*. Imprimerie royale, 1823.

[17] Debarshi Patanjali Ghoshal and Hannah Michalska. Forward-backward kernel-based state and parameter estimation for linear systems of arbitrary order. *arXiv preprint arXiv:2012.09033*, 2020.

[18] S Stanhope, Jonathan E Rubin, and David Swigon. Identifiability of linear and linear-in-parameters dynamical systems from a single trajectory. *SIAM Journal on Applied Dynamical Systems*, 13(4):1792–1815, 2014.

[19] Stefan Schäffler. *Generalized Stochastic Processes*. Springer, 2018.

[20] Debarshi Patanjali Ghoshal and Hannah Michalska. Finite interval estimation of lti systems using differential invariance, instrumental variables, and covariance weighting*. In *2020 American Control Conference (ACC)*, pages 731–736, 2020.

[21] Thomas Kailath, Ali H Sayed, and Babak Hassibi. *Linear estimation*, volume BOOK. Prentice Hall, 2000.

[22] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[23] Peter S Maybeck. The kalman filter: An introduction to concepts. In *Autonomous robot vehicles*, pages 194–204. Springer, 1990.

[24] An Introduction to the Kalman Filter, 1997.

[25] Peter S Maybeck. *Stochastic models, estimation, and control*. Academic press, 1982.

[26] Mohinder S. Grewal and Angus P. Andrews. *Optimal Smoothers*, pages 183–223. Wiley–Blackwell, 2008.

[27] Roger R. Labbe Jr. Filterpy. `https://github.com/rlabbe/filterpy`, 2015.

[28] B Anderson. Properties of optimal linear smoothing. *IEEE Transactions on Automatic Control*, 14(1):114–115, 1969.

[29] Simon J Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[30] Jeffrey K Uhlmann. Simultaneous map building and localization for real time applications. Master's thesis, University of Oxford, 1994.

[31] E Wan and R Van Der Merwe. The unscented kalman filter, ch. 7: Kalman filtering and neural networks. edited by s. haykin, 2001.

[32] Rudolph Van Der Merwe and Eric A Wan. The square-root unscented kalman filter for state and parameter-estimation. In *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, volume 6, pages 3461–3464. IEEE, 2001.

[33] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[34] Simo Särkkä. Unscented rauch–tung–striebel smoother. *IEEE transactions on automatic control*, 53(3):845–849, 2008.

[35] Simo Särkkä. Continuous-time and continuous–discrete-time unscented rauch–tung–striebel smoothers. *Signal Processing*, 90(1):225–235, 2010.

[36] Nithilasaravanan Kuppan and Manoj Venkatesan. Python estimation toolkits. `https://github.com/NithilaSaravanan/PETS/`, 2022.