Latent Variable Modeling with Slowness, Monotonicity, and Impulsivity Features

by

Ranjith Ravi Kumar Chiplunkar

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

PROCESS CONTROL

Department of Chemical and Materials Engineering

University of Alberta

Abstract

Data-driven modeling has been finding increasing prominence in process systems engineering in both academia and industries. Latent variable modeling forms an important component of data-driven modeling. Through latent variable modeling, not only can we deal with issues such as collinearity, noise, and high data dimensionality, but we can also impart the "notions" that we have regarding the process into the model. Imparting such notions makes the latent variables process-relevant and enhances the accuracy of the models. This thesis explores such ways of making the latent variables process-relevant by incorporating aspects such as slowness, monotonicity, and impulsivity that are commonly observed in various industrial processes.

Many chemical engineering processes are typically characterized primarily by slow variations. The latent variables of such processes will be characterized by high temporal correlation or low velocities. This aspect is considered in slow feature analysis which is a latent variable method that aims to extract slowly varying features. Since the basic version of slow feature analysis is unsupervised, the first contribution of the thesis explores supervised learning of slow features through a linear model. In this case, the objective function of the slow feature analysis method is modified by adding a term that maximizes the correlation of the slow features with the output variables. Two such formulations are proposed to achieve the required objective and corresponding algorithms to achieve each objective are proposed.

The second contribution extends the supervised slow feature extraction problem to a nonlinear case. The nonlinearity is achieved through the usage of Siamese neural networks. Siamese neural networks contain two identical networks that give them

the ability to handle two samples at a time. Since the objective of slow feature analysis is to reduce the velocity of the latent variables, it needs to handle two samples simultaneously. Hence, this work uses the Siamese networks to perform supervised slow feature analysis.

The third contribution of the thesis considers the monotonicity aspect in latent variable modeling for degrading processes. Processes that have degradation in either equipment or the quality of the process are overall non-stationarity in nature. Since degradation or damage usually evolves monotonically, latent variable modeling of such processes needs to include the monotonicity condition. This work proposes a state-space model to characterize such systems where the latent variable corresponding to the degrading component is modeled using a closed skew-normal random walk model, and other stationary variations are modeled by a Gaussian dynamic model. Here, the objective is to separate the monotonically degrading component of the data from other stationary variations for effective monitoring of the process. The resulting simultaneous state-and-parameter estimation problem is solved using the expectation-maximization approach and the smoothing algorithm is rigorously derived for a system defined by a closed skew-normal distribution random walk model.

In the fourth contribution of the thesis, processes that are characterized by sudden or impulsive changes are studied. To characterize such behaviors the system is modeled using a state-space model with the dynamics of one of the states being defined by a Cauchy distribution. Since the Cauchy distribution has a fat tail, it can model the sudden jumps in a process. Hence, the resulting model has a mix of Cauchy and Gaussian latent variables, where the Cauchy latent variable models the sudden jumps and the Gaussian latent variables model other variations. The states and parameters of the resulting model are identified in a Bayesian manner using the variational Bayesian inference framework. The efficacy of all the contributions is verified through both numerical and relevant industrial case studies.

Preface

This thesis is an original work conducted by Ranjith Chiplunkar under the supervision of Dr. Biao Huang and is funded in part by Natural Sciences and Engineering Research Council (NSERC) of Canada. Portions of the thesis have been published in peer-reviewed journals.

1. Chapter 3 of this thesis has been published as: **R. Chiplunkar** and B. Huang, "Output relevant slow feature extraction using partial least squares," *Chemometrics and Intelligent Laboratory Systems*, vol. 191, pp. 148–157, 2019.

2. Chapter 4 of this thesis has been published as **R. Chiplunkar** and B. Huang, "Siamese Neural Network-Based Supervised Slow Feature Extraction for Soft Sensor Application," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 9, pp. 8953-8962, 2021.

3. Chapter 5 of this thesis has been published as **R. Chiplunkar** and B. Huang, "Latent variable modeling and state estimation of non-stationary processes driven by monotonic trends", *Journal of Process Control*, vol. 108, pp. 40-54, 2021

4. Chapter 6 of this thesis will be submitted as **R. Chiplunkar** and B. Huang, "Modeling and Bayesian Inference for processes characterized by impulsive changes".

   The data for the experimental case studies presented in chapters 3 and 4 were provided by Dr. Lei Fan, Senior Engineer at Amgen who at the time was a graduate student at the University of Alberta.

Dedicated to

My loving parents Ravi Kumar and Rashmi, and all my teachers.

Acknowledgements

This thesis would not have been possible without the efforts and contributions of many, the foremost of whom is my supervisor Dr. Biao Huang. I express my sincerest gratitude to Dr. Biao Huang for giving me the opportunity to work under his supervision. Working in his group has been a greatly rewarding experience to me both in terms of technical and personal aspects. I appreciate his patience and support throughout my Ph.D. His insightful suggestions have greatly helped me in resolving many challenges I faced in research as a graduate research student. I am also grateful to him for exposing me to various real-world process systems engineering problems through industrial projects.

I want to express my gratitude to the supervisory committee members Dr. Vinay Prasad and Dr. Jinfeng Liu for their valuable feedback during my candidacy exam, which has helped me improve my work.

Working in a vast research group has been an enriching experience due to the diverse areas of research conducted in the group. I have enjoyed working with Oguzhan Dogru on applications of state estimation in reinforcement learning, Dr. Jayaram Valluru on PSFA, Dr. Xunyuan Yin on the fouling monitoring project, Vamsi Krishna on the heat exchanger modeling project, and Dr. Yanjun Ma on the steam quality soft sensor project. I also want to acknowledge the insightful discussions with Dr. Rahul Raveendran that have helped me in my research. I want to thank the current and past members of the research group Rui, Yashas, Arun, Dr. Nabil, Alireza, Dr. Lei, Aswathi, David, and many others for being wonderful colleagues. Special thanks to Dr. Fadi Ibrahim and Mrs. Terry Runyon for all the support.

I want to acknowledge the financial support from Natural Sciences and Engineering Research Council (NSERC) of Canada. I also want to express my gratitude to the University of Alberta, and the Department of Chemical and Materials Engineering

for providing financial support and other resources to conduct research.

I want to acknowledge all my friends Bharath, Bala, Venu, Damayantee, Sourayon, especially Noureen who have made my graduate studies memorable. The lockdown periods due to the pandemic have been tougher times and I want to thank all my friends for making my stay easier. I want to thank my sister Rajani and all the family members for their continuous love and support. Finally, I am greatly indebted to my parents for their love and sacrifices without which this journey would not have been possible.

Contents

List of Tables

List of Figures

List of Algorithms

Glossary

Abbreviations

ANN	Artificial neural networks
ARMA	Autoregressive moving average
CA	Cointegration analysis
cdf	Cumulative distribution function
CSN	Closed skew-normal distribution
DLM	Dynamic linear model
DPCA	Dynamic principal component analysis
DPLS	Dynamic partial least squares
ELBO	Evidence lower bound
EM	Expectation-maximization
E-step	Expectation step
FNN	Feedforward neural network
HELM	Hierarchical extreme learning machine
HLS	Hot lime softener
KL	Kullback–Leibler
LMT	Latent monotonic trend

LST	Latent stationary trend
LSTM	Long-short term memory network
LV	Latent variable
MAP	Maximum a posteriori
MLE	Maximum likelihood estimation
M-step	Maximization step
MSV	Mean of the squared velocities
NIPALS	Nonlinear iterative partial least squares
PCA	Principal component analysis
PCR	Principal component regression
pdf	Probability distribution function
PLS	Partial least squares
prop	Proposed approach
PSFA	Probabilistic slow feature analysis
RBM	Restricted Boltzmann machine
ReLU	Rectified linear unit
RMSE	Root mean squared error
RTS	Rauch-Tung-Striebel
SAE	Stacked autoencoder
SAGD	Steam-assisted gravity drainage
SF	Slow features
SFA	Slow feature analysis

SFR	Slow feature regression
SIMPLS	Statistically inspired modification of partial least squares
SSA	Stationary subspace analysis
SSE	Sum of squared errors
SSFASN	Supervised slow feature analysis by Siamese networks
V#	Valve number #
VB	Variational Bayesian
VW-SAE	Variable-wise weighted stacked autoencoder

Notations

$\langle \cdot \rangle$	Expectation operator
A	State evolution matrix
a_i	ith diagonal entry of the matrix A
α	Objective function term weight
$\left(\alpha_{(\cdot)}^{(\cdot)}, \beta_{(\cdot)}^{(\cdot)}\right)$	Parameters of Gamma and Beta distributions
b	Impulsive random variable evolution coefficient
$\mathcal{B}(\cdot)$	Beta distribution
B_{PLS}	Regression coefficient of partial least squares
b_a	ath regression coefficient
β	Objective function term weight
C	Output weight matrix

$\mathcal{C}(\cdot)$	Cauchy distribution
c_a	ath output weight
C_t	Gain matrix in the CSN smoothing step
c_t	Cauchy (Impulsive) latent variable
$cov(\cdot)$	Covariance matrix
D	Input velocity covariance matrix
Δ	Fifth parameter of the CSN
δ	Fifth parameter of the one-dimensional CSN
ΔP	Pressure drop
$det(\cdot)$	Determinant
$diag(\cdot)$	Diagonal elements of $(\cdot)$ or Diagonal matrix form of vector $(\cdot)$
$D_{KL}(q(S)\|\|p(S)$	Kullback–Leibler divergence between $q(S)$ and $p(S)$
e_t	White noise (state evolution noise for m_t and c_t)
$\mathbb{E}[\cdot]$	Expectation operation
$\exp(\cdot)$	Exponential operation
F	Flow-rate
$f(\cdot)$	Encoder network function
$f_{(\cdot)}$, $f_{(\cdot),(\cdot)}$	First and second order derivative of f
F_A, F_B	Fluctuation terms
$g(\cdot)$	Output network function
Γ	Skewness parameter of the CSN
$\Gamma(\cdot)$	Gamma distribution

H, M	Latent space to observed data mapping matrix
J_t	Gain matrix in the CSN filtering (prediction) step
K_t	Kalman gain matrix
$\mathcal{L}(q, \theta)$	Evidence lower bound
$\ln(\cdot)$	Natural logarithm
m_t	Latent monotonic variable at time t
μ	Mean of a Gaussian distribution or the first parameter of the CSN
N	Total number of samples
$n_{(\cdot)}$	Dimension of the variable $(\cdot)$
$\mathcal{N}(\cdot)$	Normal (Gaussian) distribution
ν	Fourth parameter of CSN
P	Input loading matrix
$p(\cdot)$	Probability distribution
p_a	ath input loading
$\Phi(\cdot)$	Cumulative probability distribution of a Gaussian variable
Q_a	SIMPLS objective function matrix at ath iteration
$q(\cdot)$	Sampling distribution or Proposal distribution
ρ	Pearson correlation coefficient (Skewness parameter of $p(m_t\|m_{t-1})$)
ρ_c	Concordance correlation coefficient
r_t	White noise
$\Sigma_{(\cdot)}$	Covariance matrix of $(\cdot)$
$\sigma_{(\cdot)}$	Standard deviation of $(\cdot)$

S	Set of all the states
$S(\cdot)$	Sphering operation
S_a	ath input-output covariance matrix
$\mathcal{S}_t$	Augmented state at time t
$s_t^{(i)}$	ith slow feature (or a state) at time t
T	Final time index (Total number of samples)
$\mathbf{T}$	Input scores matrix
θ	Parameters
θ_{pr}	Parameters of the prior distribution
t	Time index
t_a	ath input score
τ	Precision
u_a	ath Output score
u_t, w_t	Observation model noise at time t
V	Orthogonal loading matrix
v_a	ath orthogonalized loading
Var$[\cdot]$	Variance of $\cdot$
v_t	State evolution noise
W	Input weight matrix
w_a	ath input weight vector
$w(s_t^{(i)})$	Weight of the ith particle of state s at time t
$\tilde{w}(s_t^{(i)})$	Normalized weight of the ith particle of state s at time t

X	Observed input dataset
x_i	ith input sample
Y	Observed output dataset
y_i	ith Output sample
Z	A general random variable in the variational Bayesian scheme
$\tilde{Z}$	Random variables other than Z in the variational Bayesian scheme

Chapter 1

Introduction

The process industry has been continuously exploring various avenues to enhance the operation of industrial processes to meet the standards of green, safe and sustainable production. With improvements in the technology to store and handle vast quantities of data, the industry is looking to leverage such capabilities to achieve these standards. Data-driven modeling thus presents an attractive way to model industrial processes for efficient operation [1]. This chapter outlines the motivation for the proposed data-driven approaches to model industrial processes and provides an overview of the relevant existing literature.

1.1 Motivation

For the efficient and smooth running of an industrial process, the accurate knowledge of key variables and indicators through tools such as soft sensors [2] or process monitoring statistics [3] is of primal importance. Data-driven modeling aims at estimating these based on the observed historical data. Raw industrial data has issues of collinearity, outliers, high dimensionality, missing values, etc. Hence, the raw data is usually projected onto a latent space and these latent variables (LV) are then used to develop models. But, a pure data-based approach may not be the most effective way of latent variable modeling and hence it is important to make the LVs process-relevant. Hence, it becomes pertinent to perform LV modeling in such a way that it allows one to impart the "notions" that one may have regarding the nature of the process. These notions could be either based on the knowledge of the physics of the process or the observed data.

Every process possesses its own unique characteristic such as stationarity or non-stationarity, temporal slowness, impulsivity, multi-modal behavior, oscillatory behavior, etc. Hence, the LVs developed must be tailored to each process such that it reflects the nature of the process intended to be captured by the model. This thesis mainly focuses on three key characters of the industrial processes: slowness, monotonicity, and impulsivity. The following sections will discuss each behavior in detail.

1.1.1 Slowness

Many chemical engineering processes are primarily slow in nature as the process conditions vary slowly. Hence, the LVs that characterize such processes must primarily be slow in nature. Slow feature analysis (SFA) is a linear LV extracting method that is based on this notion. SFA [4] extracts the latent space by minimizing the velocity of the LVs. The slowest LVs are usually used for further modeling as they are considered to be the most informative if the process in consideration is believed to be primarily slowly driven. The faster ones are usually attributed to noise or other disturbances that are not representative of the dynamics of the process. Fig. 1.1 depicts the result after SFA is performed on the observed data. It can be observed that the extracted SFs are segregated according to their velocities and usually the slowest ones are used for modeling.

The vanilla version of SFA is unsupervised in nature [4], meaning the slow features (SF) are extracted without consideration of the output variable. Hence, if supervised learning is the task at hand, such as the development of soft sensors, this is not the most effective way of extracting the SFs. This hence prompts the exploration of supervised learning of SFs such that the extracted SFs are more relevant to the outputs they predict and hence improve the accuracy of the model. This argument is similar to the one used typically in the case of partial least squares (PLS) and principal component regression (PCR) case where the supervised nature of PLS makes it more suited for such cases. Motivated by this, this thesis presents a method of performing supervised slow feature extraction which combines the aspects of PLS with SFA. The resulting formulation improves the performance of the model on the datasets of processes where slowness is the key factor.

Deep learning in recent years has seen a big boom owing to the emergence of

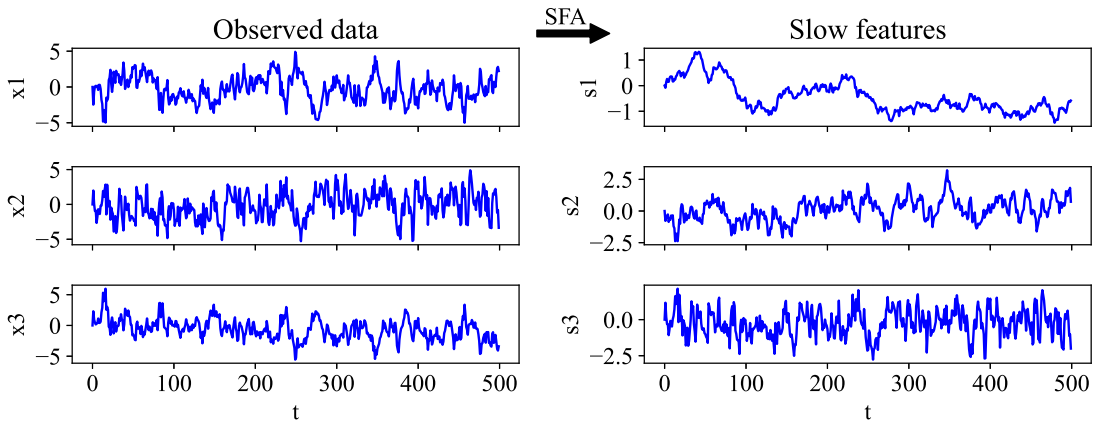


Figure 1.1: Depiction of SFA where the observed data x is mapped to the SFs s. The observed data is mapped to SFs which are segregated according to the velocities. The slowest features are more representative of the true nature of the dynamics of the process.

enhanced computational ability and importantly due to the development of sophisticated algorithms. As a result, artificial neural networks and their variants have been prominently used in machine learning research and applications [5]. Process systems engineering too has seen increasing usage of deep learning techniques including soft sensors [6]. Hence, to model nonlinear processes characterized by slowness, the powerful tools of deep learning can be used. This thesis presents a method for achieving the aforementioned objective of supervised SFA for nonlinear systems through Siamese neural networks, a neural network architecture containing two identical coupled networks [7].

1.1.2 Monotonicity

Processes involving either degradation of process quality (catalyst deactivation, fouling, etc) or damage in physical equipment (wear and tear, corrosion, etc) usually are non-stationary in nature. In particular, this non-stationarity is characterized by a monotonically evolving factor. Monotonicity refers to the non-stationary behavior where a signal is either strictly increasing or strictly decreasing. Tracking signals which exhibit such behaviors is vital for effective monitoring of such processes.

But in many processes, this health factor is masked by observation noise as shown in Fig. 1.2. This makes it difficult to monitor the process and hence filtering tech-

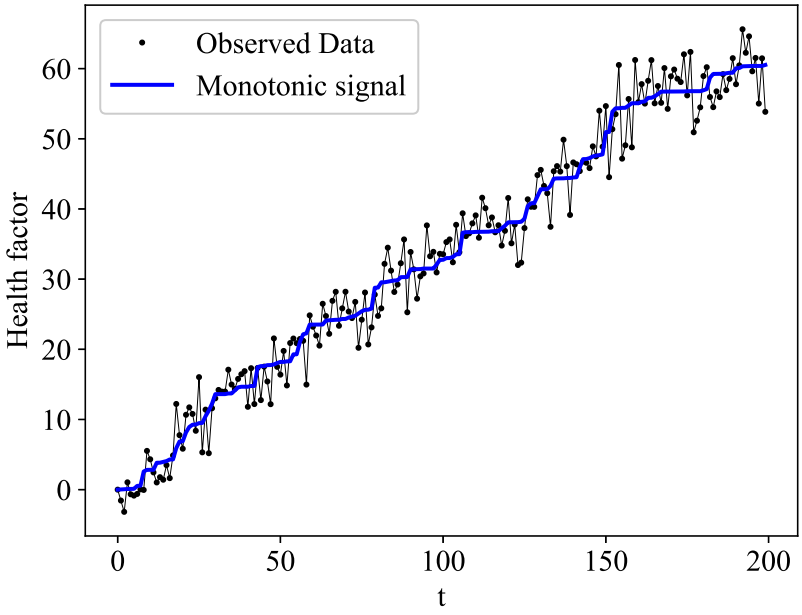
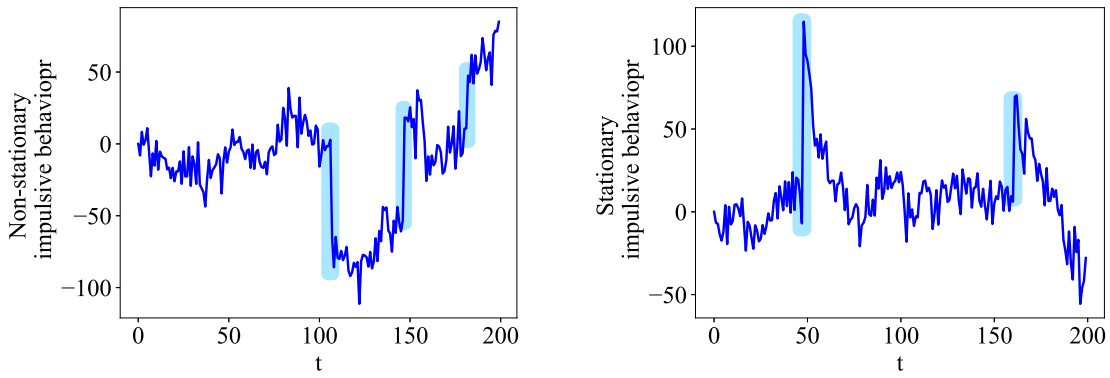


Figure 1.2: An example of a monotonically evolving health factor masked by observation noise. The objective is to extract this latent monotonic trend for effective process monitoring.

niques may be used to remove the observed noise. But this filtering cannot be a straightforward unconstrained filtering scheme as the monotonic nature of the LV must be respected. This thesis proposes a method for modeling and estimating the LV of such a system where the monotonicity constraint is implemented through a closed skew-normal distribution.

The problem depicted in Fig. 1.2 is that of a single variable case, i.e when the health factor is directly observed, but subject to noise. This thesis also considers an extended case where the health factor is not directly observed but needs to be extracted from a multivariate dataset. In such cases, the health factor will not only be masked by noise but also additionally will be masked by signals of stationary nature. This is because, in spite of the monotonic degradation nature of the process, there will also exist some stationary relationships between different process variables. Hence, the observed dataset of such systems will overall be non-stationary but will be a mixture of stationary and non-stationary (monotonic in this case) sources. Hence, for effective monitoring, one needs to separate the signals that correspond to the



(a) Non-stationary impulsive changes (b) Stationary impulsive changes

Figure 1.3: Example depiction of impulsive behavior in dynamic processes

monotonic, and stationary sources. In other words, the objective is to estimate and separate the latent monotonic trend (LMT) and latent stationary trends (LST) from the observed dataset. This thesis proposes a method of modeling and estimation of such systems where the monotonic component is modeled using a closed skew-normal distribution and the stationary components are modeled using Gaussian distribution. The resulting model is estimated using the expectation-maximization algorithm.

1.1.3 Impulsivity

Another commonly occurring type of dynamic behavior in industrial datasets is that of impulsive behavior. This is characterized by sudden jumps or abrupt changes in the variable. An example of such a behavior is depicted in Fig. 1.3a and 1.3b. Such behaviors can be observed when the process condition itself is moved from one regime to another quickly, or when there is a sudden injection of a disturbance in some either observed or unobserved variables. Hence, based on the types, one can have either a non-stationary or a stationary impulsive behavior respectively. Fig. 1.3a and 1.3b depict the non-stationary and stationary impulsive behaviors respectively. It needs to be noted that the abrupt jumps seen in the stationary impulsive behavior case are not measurement outliers, but true changes that happen due to a sudden injection of disturbance.

This thesis presents a method to model such systems using a state-space model where one of the latent variables evolves according to a Cauchy distribution. Cauchy

distribution is a fat-tailed distribution and hence can easily have realizations that are farther from the center when compared with a Gaussian distribution. Hence, the Cauchy distribution can capture such abrupt jumps better than the Gaussian. For a single variable case, it may be sufficient to just filter out the available signal. But in a multivariate case, there will also exist some slower variations mixed with the impulsive ones which can be characterized by Gaussian state variables. This thus results in a model containing a mixture of Cauchy and Gaussian LVs. The resulting model is estimated in a Bayesian manner using the variational Bayesian inference approach. The Bayesian scheme provides a convenient way of imposing the beliefs about states, and parameters. Since the objective is to separate impulsive and slower variations, the preferences regarding slowness can be implemented in a Bayesian scheme conveniently [8].

While multimodal modeling is a solution to such problems, particularly for the non-stationary case depicted in Fig. 1.3a, it may not be the most suitable option always. This is because modeling with such a perspective might need one to model for a huge number of modes which may not feasible. So a feasible solution in such a scenario is to separate the abrupt jumps from the slower stationary variations for a better understanding of the operation.

1.2 Background Literature

This section presents the background literature related to the various algorithms developed in the thesis. Since three characteristics of the LVs are considered, the literature overview is divided into three subsections each covering the literature related to slowness, monotonicity, and impulsivity.

1.2.1 Slowness

Among the various LV methods principal component analysis (PCA) [9] and PLS-based are still among the most widely used ones [1]. In both of these methods, LVs are extracted based on the notions of variances; preserving the variance in the case of PCR and maximizing the covariance with the outputs in the case of PLS. These are widely used in unsupervised learning applications such as process monitoring [10–13]

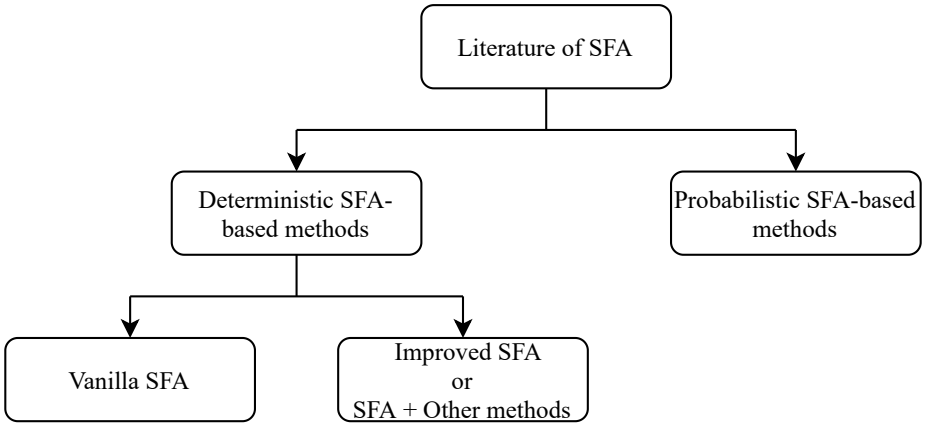


Figure 1.4: The SFA literature classification

and supervised learning applications such as soft sensors [14, 15]. For soft sensor applications, given the supervised nature of LV extraction in PLS, it is preferred over PCR. PLS as such does not consider the dynamic nature of the process and hence various versions of PLS are developed where the dynamic nature of the process is considered. These include but not limited to dynamic PLS [16–18], dynamic inner PLS [19, 20], recursive PLS [21], adaptive PLS [22], ensemble PLS [23, 24], locally weighted PLS [25, 26]. Except for the dynamic inner PLS method, other versions of PLS do not have an explicit model for the dynamics of the process. As discussed earlier many chemical processes are characterized primarily by slower dynamics and hence the LV model that explicitly considers the slowness principle is more desirable.

The aspect of slowness is considered in the framework of SFA. The vanilla version of SFA was proposed by Wiskott and Sejnowski [4] with the notion that slowly varying representation of faster appearing sensory signals is of a higher abstraction level. The idea of the slowness principle is based on a theory in neuroscience according to which, the processing of imagery in the brain happens based on slowly moving features. In computer science, this idea is used as a basis to perform SFA for the analysis of videos [27–30]. SFA has been increasingly used in chemical engineering applications such as process monitoring, soft sensor design, etc. The literature of SFA can be broadly categorized into two classes of deterministic SFA and probabilistic slow feature analysis(PSFA)-based approaches (Fig. 1.4).

1.2.1.1 Deterministic SFA

Among the unsupervised applications, SFA has been used in many process monitoring applications such as control loop performance assessment [31], detection of plant-wide oscillations [32], operation condition monitoring and anomaly detection [33], batch process monitoring [34]. Shang *et al.* [35] demonstrated the effectiveness of vanilla SFA on quality prediction problems using slow feature regression (SFR) where the SFs extracted in an unsupervised manner are used for regression. The recent research in SFA has been either to modify SFA or to use SFA in conjunction with other algorithms so as to improve the performance and widen the horizon of applicability of the vanilla SFA (Fig. 1.4).

The natural extension of SFA to dynamic SFA [36,37], recursive SFA [38], ensemble SFA [39], locally weighted SFR [40] etc have also been explored. Besides these, many improvements to the vanilla SFA have been proposed. A multi-lag SFA (and multi-lag dynamic SFA) framework has been proposed to consider multi-lag correlations to improve the detection and isolation of oscillations [41]. Another such improvement is the recursive exponential SFA where instead of the covariance matrices of the variables and their velocities, the exponential of these matrices is taken. This results in SFs that are further slower than the vanilla SFs [42]. An improvement over dynamic SFA in the form of dynamic inner SFA is proposed. Similar to its PLS counterpart, an autoregressive moving average model (ARMA) is assumed for the latent SFs [43, 44] thus incorporating dynamics into the model explicitly.

Another approach is to use SFA along with other algorithms to cater to a wider range of problems. Since, SFA does not explicitly consider non-stationarity, for non-stationary processes SFA can be combined with non-stationary analysis methods to improve the performance. Zhao and Huang [45] used cointegration analysis (CA) to monitor the non-stationary components and SFA for the stationary ones. An extension of this approach for the nonlinear case also has been explored [46]. Another method where multiple SFA models are developed for different time slices which are defined based on the process condition is developed to suit the SFA framework for non-stationary processes [47]. Besides these SFA has been used to complement numerous methods such as canonical correlation analysis to obtain features that have the quality

of slowness and are relevant to the performance variables [48]. The usage of SFA in various frameworks in literature all point to the effectiveness of extracting a latent space that considers the slowness aspect explicitly for analyzing chemical engineering process datasets. And considering the fact that in the existing literature of SFA where SFA is used for regression, the simultaneous considerations of the input space and the output space for extracting the SFs have not been done, thus prompting further exploration of ways to achieve this objective.

1.2.1.2 Probabilistic SFA

Probabilistic modeling has many advantages such as the handling of missing data, handling outliers, etc because the latent space is defined in terms of probability distributions rather than deterministic quantities [49]. The probabilistic counterparts of PCA [50] and PLS [51, 52] certainly have these advantages over their deterministic counterparts. In consideration of these advantages the probabilistic counterpart of SFA, PSFA also has been proposed [53]. In this formulation, the slow features are assumed to evolve as dictated by a Markov chain. The slow features are mapped to the observed dataset, through a linear transformation. Essentially, PSFA is a state-space model with special assumptions regarding the parameters of the state transition equations to make the states the SFs with conditions of decorrelation and unit-variance. Such a model can be solved for the states and the parameters through the expectation-maximization (EM) algorithm [54, 55]. This basic PSFA model has been used to perform process monitoring [56], develop soft sensors [55], model gross errors [57]. Due to the probabilistic framework of PSFA, it has been shown to be effective in the cases of missing data, multi-rate measurements, etc.

Various extensions of PSFA have been studied in the literature which combine various aspects of state-space models, probability theory, etc to tackle common issues observed in industrial datasets. The PSFA model structure could be tweaked in multiple ways to attain the objective at hand, thus providing a more convenient way than the SFA. If the observed dataset has outliers, these could influence the estimation of the SFs and the model parameters. To make the model robust to these outliers, a fat-tailed distribution such as the student's t-distribution can be used instead of the Gaussian distribution in the measurement equation of the PSFA

model [58]. PSFA can be easily extended to systems with non-stationary behavior by including additional states modeled by a random-walk model [59]. If the system has oscillations, instead of a diagonal matrix for state transition, a block diagonal matrix that results in complex poles for the system can be assumed [60]. Such a structure would result in a better extraction of the oscillating SFs than the conventional PSFA. For the supervised learning case, one can have an augmented output model combining both inputs and outputs. Given the probabilistic nature, this framework enables a convenient way of performing semi-supervised learning [61]. Deep learning is another avenue that PSFA can utilize to extend its capabilities to nonlinear systems. Jiang *et al* [62] proposed a deep Bayesian SFA using the gated recurrent unit.

The probabilistic approach also allows one to not only model the states (SFs) and observations as random variables but also the parameters (mapping matrices and variances) as random variables. Such problems can be solved using the variational Bayesian (VB) inference framework which is an extension of the EM algorithm for the Bayesian inference of the parameters [63, 64]. Ma and Huang [65] proposed a VB inference scheme for the PSFA model by considering the parameters of the PSFA model as random variables. The constraints to certain parameters were enforced by using particular distributions as priors to those parameters (e.g. Gamma distribution to model the inverse of the variance, also known as precision). An extension of this method for the multimodal operation is proposed where multiple output models are trained and then switched appropriately to suit the current operating mode [66]. A more general extension of this idea is proposed using the concept of transfer learning where the SFs learned from multiple PSFA models are adaptively weighted to predict the outputs of the new target domain [67]. In spite of the vast literature, there is still scope to use PSFA in conjunction with other types of latent models that deal with various types of processes behaviors such as impulsivity, monotonicity, etc.

1.2.2 Monotonicity

Process quality degradation or damage usually evolves monotonically because once the damage sets in, it cannot be reversed during the course of the operation. Hence, modeling of such processes must incorporate this aspect into the LVs. Such problems may be formulated as regression problems and solved under a constrained optimiza-

tion framework. This is commonly referred to as isotonic regression [68]. However, for online monitoring application of processes driven by LMT, formulating the problem as that of state estimation of a dynamic linear model (DLM) is more desirable [69]. In these cases, a monotonic random-walk model is commonly assumed for the LMT and a distribution with support of $[0,\infty]$ (for a monotonically increasing trend) is used. For the output equation, the Gaussian noise is used. In 2004, Gorinevsky [69] used an exponential distribution to define the evolution of the LMT and proposed a batch optimization problem to obtain the maximum a posteriori probability (MAP) estimates of the state. The same method was further extended under a moving horizon estimation framework [70, 71]. Gamma distribution, another distribution with a positive support, has been more widely used in predictive maintenance to model the evolution of the LMT [72]. Schirru et al. [73] and Susto et al. [74] modeled the deteriorating health factor using a hidden gamma process. Since the calculation of posteriors becomes intractable, these works propose an approximate inference method using particle filtering schemes to extract the LMT. Besides the gamma distribution, distributions such as inverse gamma distribution [75] and inverse Gaussian distribution [76] have been used to model the incremental changes in a degradation process. These result in analytical solutions to the state estimates, but have been used under the assumption that the degrading factor is observed without any noise.

The choice of exponential or gamma distributions to model the LMT leads to the adoption of either numerical optimization approaches or approximate solutions through particle filtering to obtain the distribution of the monotonic variable. To get analytical solutions, the distribution of choice should lead to tractable distributions in the prediction, update, and smoothing steps of a state estimation procedure. Although the Gaussian distribution has these properties, it cannot be used to model the LMT as its support spans $(-\infty, \infty)$. This thesis explores the usage of the closed skew-normal distribution to model the evolution of the LMT. Closed skew-normal distribution (CSN) as the name suggests is a skewed distribution and can be considered as a generalized version of a Gaussian distribution. It is closed under linear transformation and Bayesian rule which makes it an attractive alternative to the Gaussian distribution for the cases that desire skewness. In this work, the modeling of LMT as a CSN process is explored and a simultaneous state and parameter estimation method

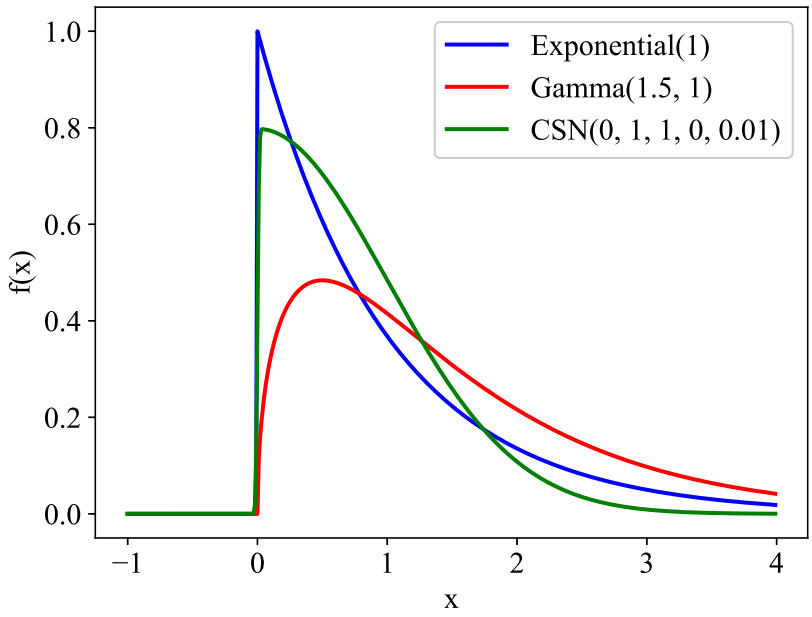


Figure 1.5: Depiction of commonly used probability distributions to model the monotonic trend. CSN although not strictly truncated ar zero, can be made to look so through appropriate parameter settings.

is proposed for such processes. Fig. 1.5 depicts a few commonly used distributions to model the monotonic trend along with the CSN.

The CSN generalizes the skew-normal distributions discussed in Azzalini and Dalla [77] and the distribution initially was proposed by Copas and Li [78]. It was used to model the distribution of a variable with another associated variable with missing data. This can be generalized to define a multivariate CSN distribution [79]. Gonzalez-Farias et al. [79] showed the closedness of the CSN under the operations of a linear transformation, conditional inference, marginalization, etc. These properties are vital for developing recursive schemes for filtering and smoothing. The development of such methods has been explored in the literature [80]. Karimi et al. [81] show that the posterior is a CSN when the prior and the likelihood are CSN and derive equations for the posterior estimate of the CSN. Rezaie and Eidsvik [82] provide an overview of the prediction and update equations for CSN-Kalman filtering. For nonlinear systems, ensemble filter [82] and unscented Kalman filter [83] in the CSN framework have been explored. Arellano-Valle et al. [84] derive the filtering

and smoothing scheme of a CSN where the process evolves with a Gaussian noise and the observations are corrupted by a CSN. The CSN filtering schemes have been implemented on cases like petroleum reservoir simulations [82], event-based state estimation [85], [86], etc.

Although CSN has been used in solving predictive maintenance problems, the modeling in these cases is not of a DLM with CSN noises. Peng and Tseng [87] proposed a skew-Wiener degradation model where the observations are directly modeled as a CSN. Huang et al. [88] take the initial value of the state as a CSN, but the increments in the state are taken as Gaussian. These formulations are not for monotonically deteriorating faults. In this thesis, a model that models the deteriorating process conditions as a monotonic fault and also separates the LST is proposed. As discussed earlier, the notion is that in a degrading process, the degrading variable need not be measured directly. This monotonic trend will be latent in the observed data which will be a mixture of stationary and non-stationary trends. Our objective is to separate these two latent trends for effective monitoring and visualization of faults.

1.2.3 Impulsivity

Certain processes are characterized by sudden jumps as observed through their datasets. One way of modeling such systems, particularly of the type depicted in Fig. 1.3a is by modeling them as piecewise constant signals [89, 90]. These kinds of problems are usually solved by optimization-based techniques [91, 92]. If one is to look at the aspects such as online predictions, randomness or uncertainties in the predictions, etc, a stochastic time series modeling of such behavior is useful. Moreover, in multivariate industrial datasets such a behavior might be observed in multiple variables and also will be mixed with other variations. Hence, rather than a signal filtering approach, an LV modeling approach is more useful.

If probabilistic modeling of such systems is to be performed, then the dynamics of the systems needs to be modeled by a distribution that has heavy tails. Hence, the Gaussian distribution cannot be used. α-stable distribution is a family of mostly heavy-tailed distributions that can be used for this purpose. In this family, the Gaussian and the Cauchy distributions are the only symmetric distributions that

have a closed-form expression for the probability distribution function (pdf), and the rest of them can only be expressed in terms of the characteristic function. Hence, they cannot be conveniently used to model the dynamics as the inference of the parameters can be complicated [93]. Cauchy distribution is a fat-tailed distribution that has a closed-form expression for the pdf and thus can be used to model the impulsive noises [94]. For a state-space model that is characterized by Cauchy distributions in both the state transition and output equations, recursive state estimation schemes can be derived using characteristic function-based filtering schemes [95–97]. Cauchy noise-based state space models have been used in robust state estimation [98] and control [99].

Heavier-tailed distributions are commonly used in such problems to achieve robustness w.r.t the measurement outliers. The most commonly used ones are the student's t-distribution [58, 100] and the Cauchy distribution [101]. The latter in fact is a special case of the student's t-distribution. While such models consider the jumps as the outliers, when the process itself has a behavior characterized by abrupt jumps, one would want to capture those jumps by modeling it in the state transition equations. The key issue is that, such a LV extraction problem from datasets with impulsive behavior leads to a simultaneous state and parameter estimation problem. In this case, one of the LVs is modeled according to a Cauchy distribution and the remaining follow Gaussian distribution. Such research is scarce in the literature, particularly in process systems engineering and this thesis aims to bridge this gap by solving this problem in a VB framework.

1.3 Thesis Outline

With the presented motivations and the background literature reviewed in Chapter 1, the thesis proceeds to detail each of the contributions in the forthcoming chapters. The rest of the thesis is organized as follows.

In Chapter 2, the mathematical background relevant to the proposed methods is presented. The first and the second contributions use the principle of slowness and hence the basic version of the deterministic and probabilistic SFA are revisited. Since the first contribution also uses PLS to make the SFs output-relevant, the formulation

of PLS and algorithms to extract LVs in PLS are also discussed. The third and the fourth contributions are modeled in a probabilistic framework. These are solved using the EM algorithm and the VB framework respectively. Hence, these two algorithms are discussed in detail particularly from the perspective of estimation of linear dynamic models. Finally, particle filtering and smoothing algorithms are also revisited as the fourth contribution relies on these state estimation techniques to extract the LVs.

The contributions of this thesis are discussed in detail from Chapter 3 to Chapter 6. Each chapter discusses the proposed model, algorithms to solve the proposed model, and demonstrations of the proposed approaches on relevant numerical and industrial case studies.

Chapter 3 discusses the first contribution of the thesis in detail which is the output-relevant slow feature extraction using partial least squares. The proposed modifications to the objective function of the vanilla SFA are presented. Two such modifications are proposed and two algorithms to solve the proposed objective are presented. These algorithms are modifications of the existing well-known algorithm for PLS. The proposed methods combine the advantages of SFA and PLS and result in a fewer number of the LVs required to define the system. These advantages are demonstrated through a numerical case study, an open-source industrial dataset, and an experimental dataset.

The second contribution of the thesis is discussed in chapter 4 which extends the same objective as that of Chapter 1 to a nonlinear case. This is achieved through the power of deep learning using a special type of neural network known as the Siamese neural network. The slowness aspect is related to the velocity of the LVs which needs handling two adjacent samples in time simultaneously. Siamese neural networks have such a provision and thus are employed in the proposed method. Two architectures based on the Siamese networks are proposed and the efficacy of these is shown through relevant case studies.

Chapter 5 presents the third contribution of the thesis that deals with the monotonicity in the LV. A state-space formulation of the system is proposed containing two types of LVs: LMT and LST. The LMT is modeled according to a CSN random-walk model and the LST is modeled according to a Gaussian distribution. The resulting

simultaneous state and parameter estimation problem is solved by the EM algorithm. The resulting state estimation problem now involved CSN distribution. A rigorous derivation of the analytical expressions of the filtering and smoothing steps for such a system is given. The effectiveness of the method is demonstrated through a numerical case study and an industrial fouling monitoring problem.

Chapter 6 presents the final contribution of the thesis which is the modeling of a process that shows impulsive behavior. This behavior can be caused by the sudden injection of a disturbance or a sudden change in process condition. Similar to the previous monotonic case, a state-space formulation of the system is proposed with the latent space containing Gaussian variables that model the slower variations and a Cauchy variable that models the abrupt changes. The resulting model is identified in a VB framework. The VB framework is adopted here because the modeling preferences for the slower variations modeled as SFs can be conveniently incorporated. Relevant prior distributions are assumed for all the parameters considering various constraints for each of the parameters. The states are estimated using a particle smoother algorithm as the system contains a mix of Gaussian and Cauchy LVs. The efficacy of the proposed method is demonstrated through a numerical case study and an industrial dataset obtained from a steam-assisted gravity drainage (SAGD) process.

Chapter 7 is the final chapter of the thesis and it summarizes the conclusions drawn from the various developed models and algorithms. The possible future work is also outlined in this chapter.

1.4 Main Contributions

The main contributions of the thesis are outlined in the following points.

1. A method of performing supervised extraction of SFs is proposed that combines SFA and PLS to result in a more efficient way of extracting meaningful features.

2. Deep learning is used to extract SFs in a supervised manner for nonlinear systems through the usage of Siamese neural networks.

3. An approach towards the formulation and estimation of systems that have both monotonical and stationary variations in the latent space of the observed data is

proposed. A state estimation procedure for such a system is rigorously derived.

4. An approach towards the formulation and estimation of systems that have both impulsive and slow variations in the latent space is proposed. The resulting problem is solved in a VB framework.

Chapter 2

Mathematical Background

This chapter provides a review of the various algorithms employed in the methods proposed in this thesis. As discussed, all the proposed methods are LV modeling methods. Of the four contributions, the first two are deterministic LV modeling methods and the next two are probabilistic ones. The first contribution is based on SFA and PLS and hence mathematical foundations of each of these are presented in detail. The second contribution involves Siamese neural networks and the relevant discussion regarding these is presented in Chapter 4. The third and fourth contributions involve probabilistic modeling of latent variables and involve the adoption of the variational inference framework. The VB method and its special case of EM algorithm both are presented in detail in this chapter. The final contribution also involves particle filtering and smoothing and thus description of these algorithms is also presented here.

2.1 PLS

The PLS method emerged as an alternative to multivariate linear regression and PCR, and continues to be one of the most widely used methods in process data analysis [1, 102]. PLS is essentially an optimization algorithm where the objective is to extract latent features which are most correlated with the outputs. Two of the popular algorithms for solving PLS objective function are nonlinear iterative partial least squares (NIPALS) [103] and statistically inspired modification of partial least squares (SIMPLS) [104]. These algorithms provide a way of systematically solving the objective function by extracting latent features in a stage-wise manner. Each

method slightly differs in terms of the objective function they solve, but the overall model structure is similar in both cases. The PLS model is given by the following equations [103]

$$X = \mathbf{T}P' + E \tag{2.1}$$

$$Y = UQ' + G \tag{2.2}$$

Here X is the $n \times n_x$ dimensional input data matrix containing n samples of input vectors x arranged as $X = [x_1, x_2, ..., x_n]'$. Similarly, Y is the $n \times n_y$ dimensional output data matrix. $\mathbf{T}$ and U are input and output score matrices with dimensions $n \times n_l$ with n_l being the number of latent features. P and Q are the input and output loading matrices. The input and output scores are extracted such that they have maximum covariance between them. Equations (2.1) and (2.2) represent how the scores result in the respective observations.

2.1.1 NIPALS

As the name suggests NIPALS is an iterative method and at each iteration, although not explicitly mentioned, the algorithm essentially maximizes the following objective function.

$$\begin{aligned} &\underset{w_a}{\text{maximize}} \quad w_a' X_a' Y_a Y_a' X_a w_a \\ &\text{subject to} \quad w_a' w_a = 1, \quad t_b' t_a = 0 \quad \text{for} \quad a > b \end{aligned} \tag{2.3}$$

Here a represents the iteration number, and w_a is the weight of X_a. Matrices X_a and Y_a represent the original input and output data matrices if $a = 1$ or the depreciated input and output data matrices if $a > 1$. The NIPALS algorithm is given in Algorithm. 2.1. Here, t_a and u_a represent the scores of X_a and Y_a respectively, and c_a represents the weight of Y_a. B_{PLS} represents the final regression coefficient, and W, P, and C are matrices whose columns are w_a, p_a and c_a respectively. Höskuldsson [105] has shown that if we substitute expression for u_a in (2.5) using (3.10), then substitute c_a by (3.8) and substitute t_a using (2.7), we will reach the following result

$$w_a \propto X_a' Y_a Y_a' X_a \; w_a \tag{2.4}$$

In the above equation, we can see that w_a is nothing but the eigenvector of the matrix $X_a' Y_a Y_a' X_a$. This hence is the solution of the optimization problem shown in (2.3).

Algorithm 2.1 The NIPALS algorithm

1. Take a column of Y_a as an initial guess for u_a.
2. Perform a regression of X_a on u_a to get the coefficients w_a. The vector w_a contains information about covariance between X_a and u_a.

$$w_a = \frac{1}{u_a' u_a} X_a' \; u_a \tag{2.5}$$

3. Normalize w_a.

$$w_a = \frac{w_a}{\sqrt{w_a' w_a}} \tag{2.6}$$

4. Project X_a onto w_a to get scores of X_a.

$$t_a = X_a \; w_a \tag{2.7}$$

5. Regress Y_a on t_a to get c_a. The regression coefficients c_a contain information about covariance between Y_a and scores of X_a.

$$c_a = \frac{1}{t_a' t_a} Y_a' \; t_a \tag{2.8}$$

6. Normalize c_a.

$$c_a = \frac{c_a}{\sqrt{c_a' c_a}} \tag{2.9}$$

7. Project Y_a onto c_a to get scores of Y_a.

$$u_a = Y_a \; c_a \tag{2.10}$$

8. Iterate from step 2 to 7 till convergence.
9. Regress u_a on t_a. Deflate X_a and Y_a and continue with step 1.

$$b_a = \frac{1}{t_a' t_a} u_a' t_a \tag{2.11}$$

$$p_a = \frac{1}{t_a' t_a} X_a' \; t_a \tag{2.12}$$

$$X_{a+1} = X_a - t_a \; t_a' \; X_a \frac{1}{t_a' t_a} \tag{2.13}$$

$$Y_{a+1} = Y_a - t_a \; t_a' \; Y_a \frac{1}{t_a' t_a} \tag{2.14}$$

10. Iterate from steps 1 to 9 till the required number of features are extracted.
11. Calculate the final regression coefficient matrix as

$$B_{PLS} = W(P'W)^{-1} BC' \tag{2.15}$$

Algorithm 2.2 The SIMPLS algorithm

1. Define the covariance matrix between X and Y as

$$S = X'Y \tag{2.16}$$

2. Find the dominant eigenvector of $S_a'S_a$. This gives c_a, the weights of Y.
3. Obtain the weights of X as

$$w_a = Sc_a \tag{2.17}$$

4. Project X onto w_a to get the scores of X.

$$t_a = Xw_a \tag{2.18}$$

5. Normalize t_a and scale w_a using the norm of t_a.

$$w_a = \frac{w_a}{\sqrt{t_a't_a}} \tag{2.19}$$

$$t_a = \frac{t_a}{\sqrt{t_a't_a}} \tag{2.20}$$

6. Find the X and Y loadings, p_a and z_a.

$$p_a = X't_a \tag{2.21}$$

$$z_a = Y't_a \tag{2.22}$$

7. Project Y onto z_a to get the scores of Y

$$u_a = Yz_a \tag{2.23}$$

8. Make the current loading orthogonal to the previous loadings.

$$v_a = p_a \tag{2.24}$$

$$v = v_a - VV'p_a \tag{2.25}$$

9. Normalize v_a.

$$v_a = \frac{v_a}{\sqrt{v_a'v_a}} \tag{2.26}$$

10. Deflate S by projecting it onto a subspace orthogonal to the current loading.

$$S_a = (I - v_av_a')S_a \tag{2.27}$$

11. Repeat steps 2 through 10 till the required number of features are extracted.
12. Calculate the final regression coefficient matrix as

$$B_{PLS} = WW'S_0 = W\mathbf{T}'Y \tag{2.28}$$

2.1.2 SIMPLS

Unlike NIPALS, the SIMPLS approach is formulated such that first an objective function is specified, and then the algorithm solves the formulated objective function [104]. The SIMPLS method is a computationally effective and more intuitive version (in terms of the calculation of the weights) of the original version of the PLS. The objective of the SIMPLS algorithm is to find weights, w_a and c_a, such that they maximize the covariance between input and output scores. The objective function is given as

$$\begin{aligned} &\underset{p_a, c_a}{\text{maximize}} \quad w_a' X' Y c_a \\ &\text{subject to} \quad w_a' w_a = 1, \quad c_a' c_a = 1, \quad t_b' t_a = 0 \quad \text{for} \quad a > b \end{aligned} \tag{2.29}$$

The solution to the above objective function is again in the form of eigenvector extraction. The SIMPLS algorithm extracts eigenvectors from $S'S$, where $S = X'Y$, in a stage-wise manner solving the above objective. The SIMPLS algorithm is shown in Algorithm. 2.2.

Both the SIMPLS and NIPALS algorithms extract latent features from the input space and the output spaces such that they are maximally correlated. In both cases, in each step, there is deflation of certain matrices to remove the variance explained by the extracted LVs and to make the subsequent LVs independent of the current ones. In NIPALS, X and Y matrices are deflated and hence the scores and loadings are calculated based on the deflated matrices. The SIMPLS method is designed such that the matrix S is directly deflated by projecting it onto a subspace orthogonal to the subspace spanned by the previous loadings. In SIMPLS, the weights are directly calculated based on original matrices and hence the SIMPLS algorithm involves fewer computational steps.

2.2 SFA

SFA was proposed as an unsupervised method of extracting features based on temporal slowness [4]. Since the feature extraction is based on the slowness principle, SFA provides a way of extracting and segregating features based on their 'velocities'. For processes driven by slower variations, the slower features contain more relevant information about the process than the faster ones. In general, if the observed data

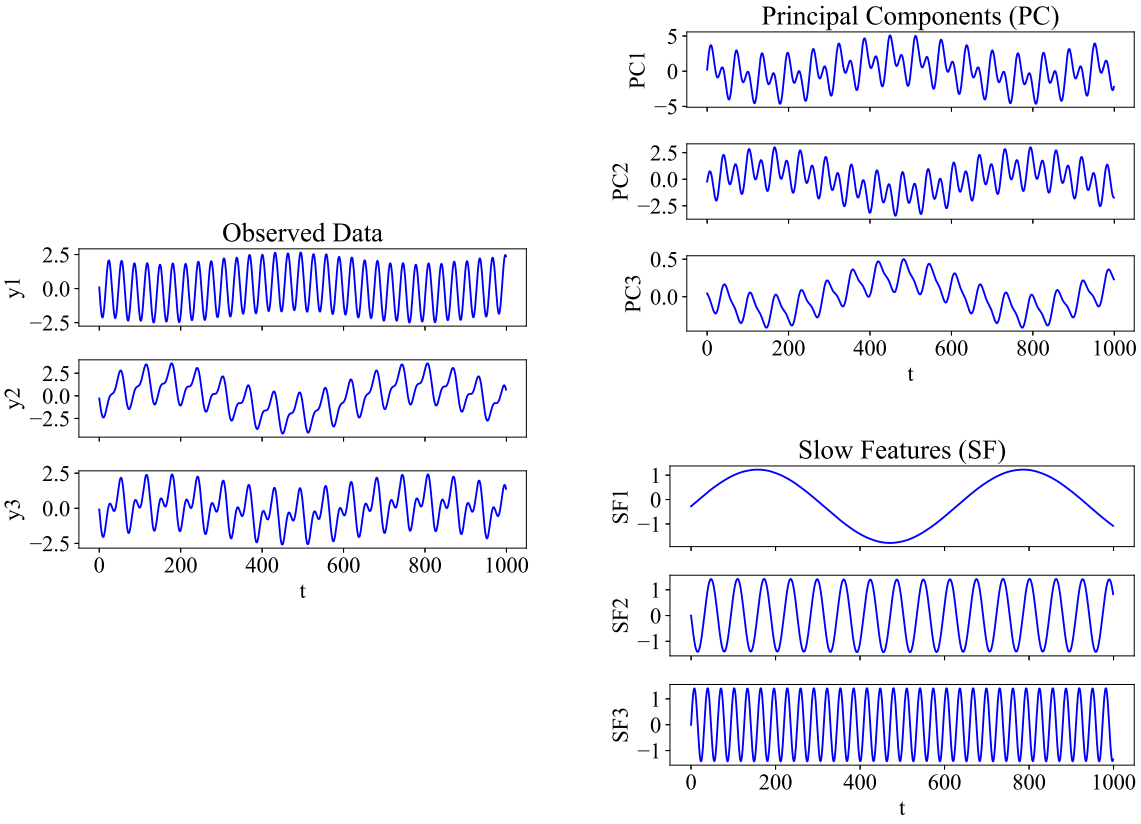


Figure 2.1: A comparison of SFA and PCR for the case where the observed data is a mixture of sine waves. Three sine waves with different frequencies (velocities) are mixed to obtain the observed data. PCA does not recover the original sine waves, whereas SFA does.

is a mixture of multiple sources, each characterized by a unique velocity, then SFA is a more apt LV modeling approach. A classical example for this case is when the observed data is a mixture of sine waves with different frequencies. In this case, each source is characterized by a unique velocity, and hence an analysis in terms of the velocities of the LVs is more appropriate. As a result, SFA recovers the sources (LVs) more accurately than PCA as shown in Fig. 2.1.

2.2.1 Deterministic SFA

In the conventional slow feature analysis [4], the aim is to find a function $g(y) = [g_1(y)\ g_2(y)\ ...\ g_k(y)]$ that maps the data y to its features, s. The following optimiza-

tion problem is solved to obtain the slow features.

$$\min_{g_j} \quad \langle \dot{s}_j^2(t) \rangle \quad or \quad \langle \Delta s_j^2(t) \rangle \tag{2.30}$$

subject to the following constraints

$$\langle s_j(t) \rangle = 0 \tag{2.31}$$

$$\langle s_j^2(t) \rangle = 1 \tag{2.32}$$

$$\langle s_{j'}(t)\ s_j(t) \rangle = 0 \quad \forall j' < j \tag{2.33}$$

Here $\dot{s}_j(t)$ is the velocity of the jth feature. For sampled systems, this can be replaced by $\Delta s_j(t) = s_j(t+1) - s_j(t)$. The angled brackets $\langle \cdot \rangle$ in the above expressions indicate the expectation over time.

$$\langle f \rangle = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} f(t) dt \tag{2.34}$$

Constraints (2.31) and (2.32) are imposed to make the features have zero mean and unit covariance. This is to avoid trivial solutions of the features being mapped to a constant signal ($s_j(t) = k$). Constraint (2.33) ensures that the obtained features are not correlated so that there is no duplication of the information that each feature contains.

For the linear SFA, the solution to the problem is similar to that of PCA. First, the data is transformed to have zero mean and identity matrix as its covariance matrix. This process is called sphering. It can be achieved through a PCA on the centered dataset.

$$z = S(\tilde{y} - \langle \tilde{y} \rangle) \tag{2.35}$$

After this, the 'velocities' of the variables are calculated, and PCA is performed on $\Delta z(t)$. The covariance of the 'velocity' data is $\langle \Delta z(t) \Delta z(t)' \rangle$. The features corresponding to the lowest eigenvalues represent the slowest features. Since we want to extract latent features with slow temporal variations, low $\langle \Delta s_j^2(t) \rangle$ is preferred. Hence features with low eigenvalues for $\langle \Delta z(t) \Delta z(t)' \rangle$ are retained.

2.2.2 Probabilistic SFA

The PSFA [53] model is a DLM with a special structure to ensure slowness along with the constraints in (2.31), (2.32), and (2.33). The PSFA model is represented by the

following equations.

$$s_t = As_{t-1} + v_t, \qquad v_t \sim \mathcal{N}(v_t; \mathbf{0}, \Sigma_v) \tag{2.36}$$

$$y_t = Hs_t + u_t, \qquad u_t \sim \mathcal{N}(u_t; \mathbf{0}, \Sigma_u) \tag{2.37}$$

Here, s_t represents the LV which is the SF at time instant t, and v_t represents the process noise. The observed data y_t is generated by the LVs s_t as dictated by the matrix H and observation noise u_t. Both noises v_t and u_t are taken as Gaussian distributions with zero mean and covariance matrix given by Σ_v and Σ_u respectively. The above equations represent a general state-space model. For PSFA, the state evolution equation has a particular structure given by the following equations.

$$A = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n_s} \end{bmatrix}; \quad 0 < a_i < 1; \quad \Sigma_v = \begin{bmatrix} 1-a_1^2 & 0 & \cdots & 0 \\ 0 & 1-a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1-a_{n_s} \end{bmatrix} \tag{2.38}$$

The matrix A is assumed to be a diagonal matrix to ensure that the SFs are uncorrelated with each other. Each diagonal entry of A is restricted to be between 0 and 1. An a_i close to 1 indicates a slow feature. The farther it is from 1, the faster it is. This is because, the expected value of the squared velocity is given as $\langle (s_t^{(i)} - s_{t-1}^{(i)})^2 \rangle = 2\,(1-a_i)$, where $s^{(i)}$ represents ith SF. This phenomenon is depicted in Fig. 2.2. The noise covariance structure assumed for Σ_v in the above equation is to ensure that the SFs have unit variance.

The estimation of the SFs in the probabilistic formulation involves the estimation of the LVs s_t and the parameters H, A, Σ_u, and Σ_v. This can be viewed as a simultaneous state and parameter estimation problem. Given the probabilistic framework of the problem, depending upon the estimation preference, the problem can be approached either from the point of view of maximizing the likelihood of the observed data or as a maximum a-posteriori estimation problem of estimating the distributions of the random variables. The next section discusses these approaches from the perspective of the estimation of the DLM expressed in (2.36) and (2.37).

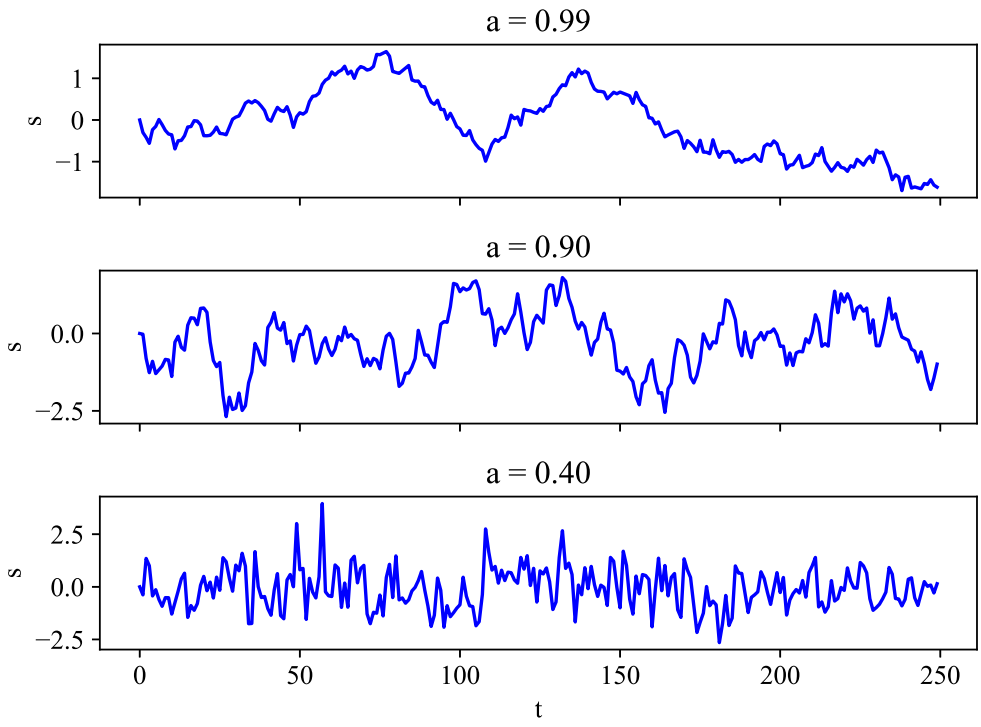


Figure 2.2: Depiction of the SFs in PSFA. The closer a is to 1, the slower is the feature.

2.3 Probabilistic Model Estimation Methods

In probabilistic approaches, the estimation of the models is done in a maximum likelihood estimation (MLE) framework which is based on the observed data, or a Bayesian framework where the prior beliefs about the parameters are used along with the observed data. For models involving LVs, these cannot be implemented in a straightforward manner because of the unobserved variables and hence other efficient techniques need to be adopted. This section presents two such methods which are the EM algorithm and the VB inference approach that are used for MLE and MAP problems respectively.

2.3.1 EM algorithm

The EM algorithm is a technique that can be used for maximizing the likelihood function of a system when the data for some of the variables are missing [106, 107]. It is one of the standard techniques used for latent variable modeling. In these cases,

maximizing the likelihood directly is infeasible. Let Y be the observed dataset and S be the latent variables. Let the parameters of the model be represented by θ. For the DLM shown in (2.36) and (2.37), $S = \{s_1, s_2, \ldots, s_T\}$ represents the set of all latent variables s_t, and $Y = \{y_1, y_2, \ldots, y_T\}$ represents the set of all observed variables y_t. θ represents the set of all parameters to be identified in the model, i.e,θ = [A, Σ_v, H, Σ_u]. It can be noted that θ includes Σ_v separately for a general DLM and for the PSFA model it will not be so as Σ_v is a function of A. Since S is not observed, one cannot maximize the total data likelihood of $p(Y, S|\theta)$. Hence the likelihood that needs to maximize is

$$\ln\, p(Y|\theta) = \ln\left(\int p(Y, S|\theta)\; dS\right) \tag{2.39}$$

Since S is not observable, let $q(S)$ be an arbitrary distribution of S through which S can be observed. This is also known as the proposal distribution. The likelihood can be written as

$$\ln\, p(Y|\theta) = \ln\left(\int \frac{p(Y, S|\theta)}{q(S)} q(S)\; dS\right) \tag{2.40}$$

One can consider the above equation as an expectation of $\frac{p(Y,S|\theta)}{q(S)}$ w.r.t $q(S)$ evaluated inside a logarithm. Maximizing such quantities is not feasible as there is an integration inside a logarithm. An easier quantity to maximize would be a logarithm inside an expectation. This can be arrived at using the Jensen's inequality according to which for a concave function such as a logarithm, the following inequality holds.

$$f(\mathbb{E}[x]) \geq \mathbb{E}[f(x)] \tag{2.41}$$

Hence, the log-likelihood can be written as

$$\ln\left(\int q(S)\frac{p(Y, S|\theta)}{q(S)}\; dS\right) \geq \int q(S) \ln\left(\frac{p(Y, S|\theta)}{q(S)}\right) dS = \mathcal{L}(q, \theta) \tag{2.42}$$

Here, $\mathcal{L}(q, \theta)$ is called the evidence lower bound (ELBO) as it is a lower bound on the log-evidence (marginal log-likelihood). Hence, maximizing this lower bound would be an alternative solution to maximizing the observed data likelihood. A closer look at the lower bound reveals its relationship with the actual log-likelihood and the

inequality gap in the above equation.

$$\begin{aligned}\mathcal{L}(q,\theta) &= \int q(S)\ln\left(\frac{p(Y,S|\theta)}{q(S)}\right)dS \\ &= \int q(S)\ln\left(\frac{p(S|Y,\theta)\ p(Y|\theta)}{q(S)}\right)dS \\ &= \int q(S)\ln\left(\frac{p(S|Y,\theta)}{q(S)}\right)dS + \int q(S)\ln\big(p(Y|\theta)\big)dS \\ &= -D_{KL}(q(S)||p(S|Y,\theta) + \ln\big(p(Y|\theta)\big) \end{aligned} \tag{2.43}$$

Here, $D_{KL}(q(S)||p(S|Y,\theta)$ represents the Kullback–Leibler (KL) divergence between the distributions $q(S)$ and $p(S|Y,\theta)$ which is a measure of the distance between two distributions. This measure is always a positive quantity and it can be observed that the inequality in (2.42) is plugged by this KL divergence. Finally, we have

$$\mathcal{L}(q,\theta) = -D_{KL}(q(S)||p(S|Y,\theta)) + \ln\big(p(Y|\theta)\big) \tag{2.44}$$

To arrive at the final steps of the EM algorithm, one needs to further examine the ELBO. We can simplify ELBO as

$$\mathcal{L}(q,\theta) = \int q(S)\ln\left(\frac{p(Y,S|\theta)}{q(S)}\right)dS = \int q(S)\ln\big(p(Y,S|\theta)\big)dS - \int q(S)\ln\big(q(S)\big)dS \tag{2.45}$$

The objective is to maximize the ELBO w.r.t the missing/hidden data (states) S, and the parameters θ. It can be observed that $q(S)$ appears only in the first term of ELBO in (2.44), and θ appears only in the first term of ELBO in (2.45). Hence, the optimization problem can be split into two steps, one where θ is optimized, and the other where S, or $q(S)$ is optimized. These two steps are iterated till convergence.

1. The first step is finding the q that maximizes the ELBO. For this we consider (2.44) where $q(S)$ appears in the KL divergence term. As mentioned earlier, KL divergence is a measure of the distance between two distributions and is a positive quantity. Since KL divergence is subtracted from $\ln\big(p(Y|\theta)\big)$, maximizing ELBO is equivalent to minimizing the KL divergence. The minimum value of KL divergence is zero when the two distributions are equal. Hence,

$$q^* = \arg\max_q \mathcal{L}(q,\theta^{old}) = p(S|Y,\theta^{old}) \tag{2.46}$$

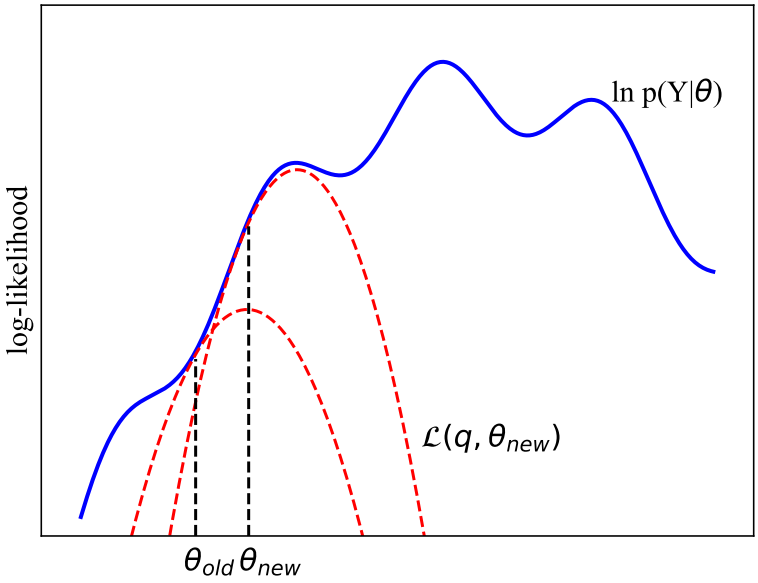


Figure 2.3: Sample depiction of the EM algorithm. The ELBO evaluated at θ_{old} is maximized to get the updated parameters to θ_{new}.

2. The second step is finding the θ^{new} that maximizes the ELBO. Since θ appears on only one term in (2.45), this equation could be used to maximize ELBO w.r.t θ. Hence, we have

$$\theta^{new} = \arg\max_{\theta} \mathcal{L}(q^*, \theta^{old}) = \int q^*(S) \ln \big(p(Y, S|\theta)\big) dS$$
$$= \mathbb{E}_{S \sim p(S|Y,\theta^{old})} \big[\ln \big(p(Y, S|\theta)\big)\big] \tag{2.47}$$

3. Iterate the two steps till convergence.

The first step where the distribution of the hidden states is estimated is nothing but the E-step. The expectation step or the E-step involves finding the expected values of various quantities which are a function of the LVs. This needs the knowledge of the distribution of the LVs which is obtained in the first step. The second step updates the parameters by maximizing the expected value of the complete data log-likelihood, with the expectation being taken according to the conditional distribution of the latent variables $p(S|Y, \theta^{old})$. This step is called the maximization step or the M-step. Fig. 2.3 shows the evolution of the ELBO during the iterations of the EM algorithm. The following points include some important aspects of the EM algorithm.

1. **Maximization aspects:** The EM algorithm does not explicitly maximize the likelihood, rather maximizes a lower bound on it. It can be shown that in each iteration, maximizing the lower bound also maximizes the observed data likelihood. Moreover, in each iteration, the likelihood always increases.

2. **Local optimum:** Even though the likelihood increases in every iteration, there is no guarantee that the algorithm converges to a global optimum. The case depicted in Fig. 2.3 corresponds to such a scenario where the EM algorithm may get stuck in a local minimum.

3. **Initial guess:** The final result of the EM algorithm depends on the initial guess of the parameters θ due to the presence of multiple local minima. One way is to have an initial guess based on some a-priori knowledge of the system (e.g. for PSFA, the deterministic SFA model can be used to get a good initial guess [55]). But in the absence of such knowledge, a random initialization is the best option. The iterations can be started from multiple initial guesses selected randomly, and the one resulting in the highest likelihood after convergence can be selected.

The DLM shown in (2.36) and (2.37) can be estimated through the EM algorithm. The M-step involves the estimation of the parameters $\theta = [A,\, \Sigma_v,\, H,\, \Sigma_u]$. The LVs $\{s_{1:T}\}$ (SFs in this case) are estimated in the estimation step which involves estimating the first and the second moments of the LVs. This can be achieved through the Kalman filter and Rauch–Tung–Striebel (RTS) smoother algorithms [55, 106]. If any of the modeling choices results in the case where the analytical realization of the E-step is not possible (the posterior $p(S|Y, \theta^{old}$ is not tractable), then other filtering and smoothing algorithms need to be used including the particle-based algorithms.

2.3.2 VB Inference

The MAP inference is a Bayesian inference method where the prior notions about the parameters are combined with the observed data to obtain the updated parameters. In this case, the parameters are also considered as random variables. This approach is particularly useful in the case where we have certain beliefs or notions about that

process and want to integrate them into the modeling process. Such modeling preferences can be realized by assuming the appropriate prior structure to the parameters. Besides imparting notions, one can also handle constraints on the parameters such as the ones in (2.38) through appropriate choice of the prior distribution. The VB inference framework is suitable for this case which works on similar notions as that of the EM algorithm, but extends to the case of Bayesian learning of the parameters [63,64].

In the EM algorithm for a DLM, the total data likelihood was $p(Y, S|\theta)$. Now, since the parameters θ are also considered as random variables, the total data likelihood is $p(Y, S, \theta|\theta_{pr})$ where θ_{pr} represents the parameters of the prior distribution of θ which are the hyper-parameters of the model. These result in the meshing of modeling preferences into the model. Hence, the relationship between the observed data likelihood and the total likelihood is represented as

$$\ln\, p(Y|\theta_{pr}) = \ln\left(\int \frac{p(Y, S, \theta|\theta_{pr})}{q(S, \theta)} q(S, \theta)\; dS\; d\theta\right) \tag{2.48}$$

Proceeding similar to the EM algorithm case, the ELBO can be expressed as

$$\begin{aligned}\mathcal{L}(q(S, \theta)) &= \int q(S, \theta) \ln\left(\frac{p(Y, S, \theta|\theta_{pr})}{q(S, \theta)}\right) dS\; d\theta \\ &= \int q(S, \theta) \ln\big(p(Y, S, \theta|\theta_{pr})\big) dS\; d\theta - \int q(S, \theta) \ln\big(q(S, \theta)\big) dS\; d\theta \end{aligned} \tag{2.49}$$

The objective is now to predict the distribution $q(S, \theta)$. Although the ELBO can be expressed as

$$\mathcal{L}(q(S, \theta)) = -D_{KL}(q(S, \theta)||p(S, \theta|Y, \theta_{pr}) + \ln\big(p(Y|\theta_{pr})\big), \tag{2.50}$$

this form only suggests that the best $q(S, \theta)$ is the posterior $p(S, \theta|Y, \theta_{pr})$ and it may not be feasible to calculate this posterior as it would be a complicated, intractable distribution of all states and parameters. Hence, an assumption is made regarding the structure of $q(S, \theta)$ that results in an efficient iterative solution to the problem. This is called the mean-field approximation [64]. According to this, the variational distribution over all the random variables in the model is factorized as

$$q(S, \theta) = q(S) \cdot q(\theta) \tag{2.51}$$

Since $\theta = [A,\, \Sigma_v,\, H,\, \Sigma_u]$, the assumption is also extended to each of the parameters. Hence,

$$q(S, \theta) = q(S) \cdot q(A) \cdot q(\Sigma_v) \cdot q(H) \cdot q(\Sigma_u) \tag{2.52}$$

Minimizing the ELBO in (2.49) w.r.t $q(S, \theta)$ as such is not feasible. The above factorization is thus substituted into (2.49) to further simplify it. Let Z represent a general random variable of the problem and $\tilde{Z}$ be all the remaining ones. Hence we have $\{Z, \tilde{Z}\} = \{S,\ A,\ \Sigma_v,\ H,\ \Sigma_u\}$. The first term of (2.49) can be viewed as an expectation of the total data likelihood w.r.t the proposal distribution. Due to the factorization, one can have the following result.

$$\int \underbrace{q(Z, \tilde{Z})}_{q(Z)\ q(\tilde{Z})} \ln\big(p(Y, Z, \tilde{Z}|\theta_{pr})\big) dZ\ d\tilde{Z} = \int q(Z) \mathbb{E}_{\tilde{Z}\sim q(\tilde{Z})}\Big[\ln\ p(Y, Z, \tilde{Z}|\theta_{pr})\Big]\ dZ \quad (2.53)$$

Similarly the second term in the ELBO can be expressed as

$$\int q(Z, \tilde{Z}) \ln\big(q(Z, \tilde{Z})\big) dZ\ d\tilde{Z} = \int q(Z) \ln q(Z)\ dZ + \int q(\tilde{Z}) \ln\big(q(\tilde{Z})\big) d\tilde{Z} \quad (2.54)$$

Hence, the ELBO can be written as follows.

$$\mathcal{L}(q(Z, \tilde{Z})) = \int q(Z) \mathbb{E}_{\tilde{Z}\sim q(\tilde{Z})}\Big[\ln\ p(Y, Z, \tilde{Z}|\theta_{pr})\Big]\ dZ - \int q(Z) \ln q(Z)\ dZ - \int q(\tilde{Z}) \ln\big(q(\tilde{Z})\big) d\tilde{Z} \quad (2.55)$$

One can make the following modification to the first term in the RHS of the above equation.

$$\mathcal{L}(q(Z, \tilde{Z})) = \underbrace{\int q(Z) \ln\Big(\exp\Big(\mathbb{E}_{\tilde{Z}\sim q(\tilde{Z})}\Big[\ln\ p(Y, Z, \tilde{Z}|\theta_{pr})\Big]\Big)\Big)\ dZ - \int q(Z) \ln q(Z)\ dZ}_{D_{KL}\ \text{form}} - \int q(\tilde{Z}) \ln\big(q(\tilde{Z})\big) d\tilde{Z} \quad (2.56)$$

Using the definition of KL divergence, the ELBO can be expressed as follows.

$$\mathcal{L}(q(Z, \tilde{Z})) = -D_{KL}\left(q(Z)||\exp\left(\mathbb{E}_{\tilde{Z}\sim q(\tilde{Z})}\left[\ln\ p(Y, Z, \tilde{Z}|\theta_{pr})\right]\right)\right) - \int q(\tilde{Z}) \ln\big(q(\tilde{Z})\big) d\tilde{Z} \quad (2.57)$$

The objective is to maximize the ELBO as expressed in the above equation. It can be observed that the term Z appears only in the KL divergence term. Hence, to maximize the ELBO w.r.t Z, only the KL divergence term needs to be considered. Since the expression in (2.57) has a negative KL divergence term, the maximum is reached when the KL divergence is zero. Hence,

$$q(Z) \propto \exp\left(\mathbb{E}_{\tilde{Z}\sim q(\tilde{Z})}\left[\ln\ p(Y, Z, \tilde{Z}|\theta_{pr})\right]\right) \quad (2.58)$$

The distribution of each random variable of the model is expressed according to the above expression. Since each variable mostly appears once each in the likelihood term and the prior term, usually the above expression boils down to

$$q(Z) \propto \exp\left(\mathbb{E}_{\tilde{Z}\sim q(\tilde{Z})}\left[\ln\ (\text{likelihood} \times \text{prior})\right]\right) \tag{2.59}$$

For example, the total data likelihood for the DLM model can be written as

$$p(S, Y, A, \Sigma_v, H, \Sigma_u) = p(S|A, \Sigma_v) \cdot p(Y|H, \Sigma_u) \cdot p(A) \cdot p(\Sigma_v) \cdot p(H) \cdot p(\Sigma_u) \tag{2.60}$$

And, if one is to write an expression for $q(A)$, only the pdf terms that have A in them need to be considered and rest can be clubbed in the constant term of (2.57). Thus, the expression for $q(A)$ can be written as

$$q(A) \propto \exp\left(\mathbb{E}_{\{S,\Sigma_u\}\sim q(S,\tilde{\Sigma}_u)}\left[\ln\ p(S|A, \Sigma_v) \cdot p(A)\right]\right) \tag{2.61}$$

which has the form given in (2.59). While estimating the posterior as in (2.59), if the prior and the likelihood form a conjugate pair, then $q(Z)$ will have an analytical expression. If that is not the case, sampling-based methods will need to be followed to estimate $q(Z)$. For the model parameters, usually importance sampling is employed for the non-conjugate pair case. For the states, since they are part of a dynamic equation, particle filtering and smoothing methods need to be used. The fourth contribution of this thesis uses the Cauchy distribution to model the state transition of one of the states. Since this results in no tractable expression for the posterior, particle smoothing is used. The next section presents the sampling-based algorithms such as importance sampling, particle filtering, and particle smoothing which are used in this thesis.

2.4 Sampling-based algorithms

It is common to encounter situations where one cannot have a tractable distribution. The common example is the case of Bayesian learning of a parameter when the prior and the likelihood do not form a conjugate pair. In such cases, one may not have a closed-form expression for the posterior distribution. In such cases, it is not possible to get an analytical expression for the moments of the distribution. In several other

cases even with an analytical expression for the distribution, it may not be possible to calculate the moments analytically. In such cases, one resorts to sampling-based algorithms which as the name suggests involves generating samples from a distribution. These samples represent the distribution and are used to calculate the moments of the distribution. Let $\{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ represent a set of N samples drawn from a distribution $p(x)$. The estimate of the distribution is represented as follows [108].

$$\hat{p}(x) = \frac{1}{N} \sum_{i=1}^{N} \delta(x - x^{(i)}) \tag{2.62}$$

Here, $\delta(x - x^{(i)})$ is the Dirac delta function.

$$\delta(x - a) = \begin{cases} +\infty, & \text{if } x = a \\ 0, & \text{otherwise} \end{cases}; \quad \int_{-\infty}^{+\infty} \delta(x - a)\ dx = 1 \tag{2.63}$$

With such a definition, it can be observed that any expectation of a function $f(x)$ evaluated according to $p(x)$ can be approximated as follows.

$$\begin{aligned} \mathbb{E}_p[f(x)] = \int_{-\infty}^{+\infty} f(x)\ p(x)\ dx &\approx \int_{-\infty}^{+\infty} f(x) \frac{1}{N} \left(\sum_{i=1}^{N} \delta(x - x^{(i)}) \right) dx \\ &= \frac{1}{N} \sum_{i=1}^{N} \int_{-\infty}^{+\infty} f(x) \delta(x - x^{(i)})\ dx = \frac{1}{N} \sum_{i=1}^{N} f(x^{(i)}) \end{aligned} \tag{2.64}$$

Hence, any moment of $p(x)$ can be calculated according to the above expression. It must be noted that in the above case each sample is generated from $p(x)$, i.e., $x^{(i)} \sim p(x)$. For the case where $p(x)$ is not tractable, one cannot generate samples from $p(x)$, and the approach to solving this problem is discussed in the next section.

2.4.1 Importance sampling

As mentioned earlier, if $p(x)$ is intractable then it is not possible to sample from it. In such a scenario one can generate samples from an easy-to-sample distribution $q(x)$, also called the importance distribution, and use these samples to estimate the distribution. In this case, instead of using the samples directly, an importance weight is assigned to each sample calculated such that it compensates for the fact that the sample belongs to a different distribution. The notion behind importance sampling

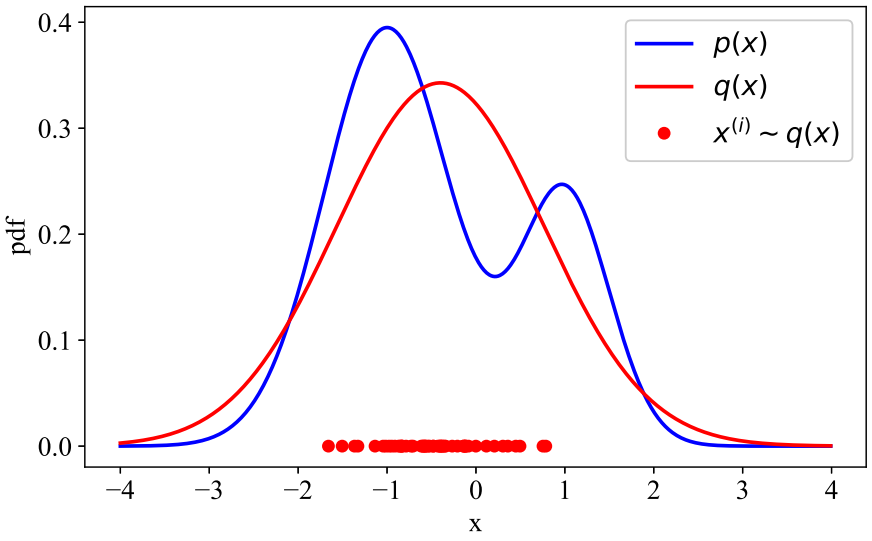


Figure 2.4: Depiction of importance sampling

and importance weights is shown in the following equations. Say the objective is to evaluate $\mathbb{E}[f(x)]$ with $x \sim p(x)$. We have

$$\mathbb{E}[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = \int w(x)f(x)q(x)\ d(x) \tag{2.65}$$

Now, $f(x)w(x)$ can be considered as a function and $q(x)$ as the distribution according to which the expectation is to be evaluated. Hence, one can generate N samples from $q(x)$ such that the samples $\{x^{(i)}\}$ represent $q(x)$ and can be represented similar to (2.62). Substituting the approximate form $\hat{q}(x)$ into (2.65) we get the following result.

$$\mathbb{E}[f(x)] = \frac{1}{N}\sum_{i=1}^{N} w(x^{(i)})f(x^{(i)}); \quad w(x^{(i)}) = \frac{p(x^{(i)})}{q(x^{(i)})}; \quad x^{(i)} \sim q(x). \tag{2.66}$$

Consequently, the distribution can be represented in terms of the importance weights.

$$\hat{p}(x) = \frac{1}{N}\sum_{i=1}^{N} w^{(i)}\delta(x - x^{(i)}) \tag{2.67}$$

Importance sampling results in a biased estimate of the expected value which can be easily observed for an extreme case when $N = 1$. Nevertheless, it gives a consistent estimate, meaning the bias approaches zero as N approaches a large number and the bias is $\mathcal{O}\ (1/N)$ [109]. Fig. 2.4 depicts the process of importance sampling. In this

case, $p(x)$ is a complex distribution that has no tractable posterior. Hence, a Gaussian distribution is selected as the importance distribution $q(x)$ from which samples can be generated conveniently.

2.4.2 Particle-based state estimation

The state estimation procedure for a DLM comprises of two main steps: the filtering and the smoothing step. For the DLM represented in (2.36) and (2.37), the filtering step involves estimating the pdf $p(s_t|y_{1:t})$ and the smoothing step involves estimating the pdf $p(s_t|y_{1:T})$, where $T > t$. These steps for a DLM with Gaussian process and observation noises are rather straightforward due to the "closedness" property of the Gaussian which is, the resultant distribution after a Bayesian inversion and a convolution operation is always a Gaussian if all the involved distributions are Gaussian. This results in a recursive estimation procedure which is realized by the Kalman filter and RTS smoother algorithms. In most of the cases where non-Gaussian distributions are involved, or if the system model is nonlinear, then such recursive derivations of the filtering and smoothing steps are not possible because of the tractability issue of estimating $p(s_t|y_{1:t})$ and $p(s_t|y_{1:T})$. In such cases, particle (sample)-based algorithms are used which are essentially performing sequential importance sampling, which is an extension of the importance sampling discussed in the previous section to the dynamic model case [109, 110].

2.4.2.1 Particle filtering

A hidden Markov model is characterized by two probability distributions: the state transition distribution $p(s_t|s_{t-1})$ and the observation noise distribution $p(y_t|s_t)$. Particle filtering aims at inferring the distribution of the states given the observations. The popular versions of the particle filtering algorithms focus on the estimation of the joint density $p(s_{1:t}|y_{1:t})$. With the Markovian assumption and the assumption of independence between observation noises, we have the following result.

$$p(s_{1:t}) = p(s_t|s_{t-1}) \cdot p(s_{1:t-1}) = \prod_{i=2}^{t} p(s_i|s_{i-1}) \cdot p(s_1) \tag{2.68}$$

$$p(y_{1:t}|s_{1:t}) = p(y_t|s_t) \cdot p(y_{1:t-1}|s_{1:t-1}) = \prod_{i=1}^{t} p(y_i|s_i) \tag{2.69}$$

With the above results, the recursive relation between $p(s_{1:t}|y_{1:t})$ and $p(s_{1:t-1}|y_{1:t-1})$ can be arrived at, as follows.

$$p(s_{1:t}|y_{1:t}) = \frac{p(y_{1:t}|s_{1:t}) \cdot p(s_{1:t})}{p(y_{1:t})} = \frac{p(y_t|s_t)p(y_{1:t-1}|s_{1:t-1}) \cdot p(s_t|s_{t-1})p(s_{1:t-1})}{p(y_{1:t})}$$
$$= \frac{p(y_t|s_t) \cdot p(s_t|s_{t-1}) \cdot p(s_{1:t-1}|y_{1:t-1})}{p(y_t|y_{1:t-1})} \tag{2.70}$$

The above relation forms the basis of the particle filtering algorithms which relates $p(s_{1:t-1}|y_{1:t-1})$ to its updated form for the next time step $p(s_{1:t}|y_{1:t})$. Since any of the posteriors $p(s_{1:i}|y_{1:i})$ are not tractable, these are represented in terms of weights as discussed in the previous section of importance sampling. With the help of (2.70), the weights that correspond to $p(s_{1:t-1}|y_{1:t-1})$ can be recursively updated using the state transition pdf and the observation noise pdf to get the weights for $p(s_{1:t}|y_{1:t})$. A general particle filtering algorithm is represented in Algorithm. 2.3.

Algorithm 2.3 A generic particle filtering algorithm

At $t = 1$

1. Sample $s_1^{(i)} \sim q(s_1|y_1)$
2. Compute the weights and normalize them.

$$w(s_1^{(i)}) = \frac{p(s_1^{(i)})\ p(y_1|s_1^{(i)})}{q(s_1^{(i)}|y_1)}; \quad \tilde{w}(s_1^{(i)}) \propto w(s_1^{(i)})$$

3. Resample (if required) $\{\tilde{w}(s_1^{(i)}), s_1^{(i)}\}$ to get $\{1/N, \bar{s}_1^{(i)}\}$

For $t \geq 2$

1. Sample $s_t^{(i)} \sim q(s_t|\bar{s}_{t-1}^{(i)}, y_t)$ and set $s_{1:t}^{(i)} \leftarrow \{s_t^{(i)}, \bar{s}_{1:t-1}^{(i)}\}$
2. Compute the weights and normalize them.

$$w(s_{1:t}^{(i)}) = \frac{p(y_t|s_t^{(i)})\ p(s_t^{(i)}|\bar{s}_{t-1}^{(i)})}{q(s_t^{(i)}|\bar{s}_{t-1}^{(i)}, y_t)} \tilde{w}(s_{1:t-1}^{(i)}); \quad \tilde{w}(s_{1:t}^{(i)}) \propto w(s_{1:t}^{(i)})$$

3. Resample (if required) $\{\tilde{w}(s_{1:t}^{(i)}), s_{1:t}^{(i)}\}$ to get $\{1/N, \bar{s}_{1:t}^{(i)}\}$

Step 3 in Algorithm. 2.3 is called the resampling step. In sequential importance sampling, the variance of the estimate increases over time and it may also result in

a scenario where many weights are close to zero. In this case, many particles will be rendered ineffective as they do not represent the distribution. An effective way to solve this issue is the resampling technique. As the name suggests, this step involves resampling the current particles with each particle assigned a probability equal to its normalized weight. This results in particles that are equally weighted and hence the weight of the new set of particles is $1/N$ for each particle. It must be noted that resampling need not be done in every step and can be performed only if the effective number of samples calculated as $\frac{1}{\sum_{i=1}^{N}(\tilde{w}(s_{1:t}^{(i)}))^2}$ falls below a certain threshold (typically $N/2$). Here $\tilde{w}(s_{1:t}^{(i)})$ represents the normalized weight associated with the augmented sample $s_{1:t}^{(i)}$.

An important aspect to be noted about (2.70) and Algorithm. 2.3 is that the distribution being learned is a joint distribution of the states over all the instants of past time i.e., $p(s_{1:t}|y_{1:t})$. In many cases, the marginal distribution of $p(s_t|y_{1:t})$ is of interest particularly for the case of smoothing which is based on marginal distribution weights. In such a case, a particle filter that recursively estimates the weights of the marginal distribution is of interest. One way of achieving this is to extract $\{s_t^{(i)}\}$ from $\{s_{1:t}^{(i)}\}$ and associate the weight $\{\tilde{w}(s_{1:t}^{(i)})\}$ with it. This results in a depletion problem as only one particular path of $s_{1:t-1}$ is considered [111]. The proper way of developing a marginal particle filter should be based on how the general marginal filtering framework is derived. Thus, the marginal particle filter is based on the following equation [112].

$$p(s_t|y_{1:t}) = \frac{p(y_t|s_t) \cdot p(s_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} = \frac{p(y_t|s_t) \cdot \int p(s_t|s_{t-1}) p(s_{t-1}|y_{1:t-1}) ds_{t-1}}{p(y_t|y_{1:t-1})} \quad (2.71)$$

In this case, the recursion step is different from the one generic particle filter algorithm as the previous step's posterior appears inside an integral. The procedure of the marginal particle filter [112] is provided in Algorithm. 2.4.

2.4.2.2 Particle smoothing

Smoothing refers to the estimation of the pdf $p(s_t|y_{1:T})$ where T is in the future of t. In this step, the past states are re-estimated based on the future observations. This procedure is particularly useful for the case where the estimated states are used to learn the parameters as in the case of learning the DLM through EM algorithm or

Algorithm 2.4 Marginal particle filtering algorithm

At $t = 1$

1. Sample $s_1^{(i)} \sim q(s_1|y_1)$
2. Compute the weights and normalize them.

$$w(s_1^{(i)}) = \frac{p(s_1^{(i)})\ p(y_1|s_1^{(i)})}{q(s_1^{(i)}|y_1)}; \quad \tilde{w}(s_1^{(i)}) \propto w(s_1^{(i)})$$

3. Resample (if required) $\{\tilde{w}(s_1^{(i)}), s_1^{(i)}\}$ to get $\{1/N, \bar{s}_1^{(i)}\}$

For $t \geq 2$

1. Sample $s_t^{(i)} \sim \sum_{j=1}^{N} \tilde{w}_{t-1}^{(j)}\ q(s_t|\bar{s}_{t-1}^{(j)}, y_t)$
2. Compute the weights and normalize them.

$$w(s_t^{(i)}) = \frac{p(y_t|s_t^{(i)})\ \sum_{j=1}^{N} \tilde{w}_{t-1}^{(j)}\ p(s_t^{(i)}|\bar{s}_{t-1}^{(j)})}{\sum_{j=1}^{N} \tilde{w}_{t-1}^{(j)}\ q(s_t^{(i)}|\bar{s}_{t-1}^{(j)}, y_t)}; \quad \tilde{w}(s_t^{(i)}) \propto w(s_t^{(i)})$$

3. Resample (if required) $\{\tilde{w}(s_t^{(i)}), s_t^{(i)}\}$ to get $\{1/N, \bar{s}_t^{(i)}\}$

VB inference. The equation for smoothing is expressed as follows.

$$p(s_t|y_{1:T}) = \int \frac{p(s_{t+1}|s_t)p(s_t|y_{1:t})}{\int p(s_{t+1}|s_t)p(s_t|y_{1:t})\ ds_t} p(s_{t+1}|y_{1:T})\ ds_{t+1} \tag{2.72}$$

As seen from the above equation, in this case, one moves backward in time i.e., $p(s_t|y_{1:T})$ is estimated based on $p(s_{t+1}|y_{1:T})$. The particle version of the above equation can be expressed as follows.

$$w(s_{t|T}^{(i)}) = w(s_t^{(i)}) \left[\sum_{j=1}^{N} w(s_{t+1|T}^{(j)}) \frac{p(s_{t+1}^{(j)}|s_t^{(i)})}{\sum_{k=1}^{N} w(s_t^{(k)})\ p(s_{t+1}^{(j)}|s_t^{(k)})} \right] \tag{2.73}$$

It is to be noted that in the above equation all the weights involved represent the marginal distributions thus requiring a marginal particle filter in the forward pass. The computational complexities of the marginal particle filter and particle smoother are high ($\mathcal{O}(N^2T)$). Although lesser computationally intensive alternatives exist, they are not accurate because the marginal distribution weight calculation is not accurate in those cases. Since the particle filtering and smoothing algorithms are run together only offline, such high computational complexity may be tolerated.

Chapter 3

Output-Relevant Slow Feature Extraction Using Partial Least Squares *

This chapter presents the first contribution of the thesis which is based on SFA. As discussed earlier, SFA is an approach that extracts features from a time series dataset in an unsupervised manner based on the temporal slowness principle. These SFs contain relevant information about the dynamics of the process and hence are useful for developing models. For supervised learning objectives, these SFs need to be relevant to the outputs. Partial least squares (PLS) is a method used to perform supervised feature extraction and build models. For time series datasets this approach cannot be used directly as it assumes each sample to be sequentially independent of each other. This work proposes an approach to perform feature extraction which combines the temporal slowness element of SFA and the output-relevance element of PLS. The proposed approach extracts temporally SFs that are relevant to the outputs, which is an essential aspect of supervised learning. The proposed formulations can be solved using the existing PLS algorithms like NIPALS and SIMPLS with proposed modifications. The proposed methods are applied to three case studies: simulated, industrial, and experimental case studies to demonstrate their efficacy.

*This chapter has been published as: **R. Chiplunkar** and B. Huang, "Output-relevant slow feature extraction using partial least squares," *Chemometrics and Intelligent Laboratory Systems*, vol. 191, pp. 148–157, 2019.

3.1 Introduction

For soft sensor development, among the linear regression methods, PCR and PLS are the commonly used approaches. The conventional PCR and PLS methods have limitations when it comes to time series models. These methods assume independence between samples which is not the case for time series data. In time series data successive samples are correlated with each other. This relationship between successive samples needs to be considered while developing models for time series datasets. Methods like dynamic principal component analysis (DPCA) and dynamic partial least squares (DPLS) (section 1.2.1) augment current observations with past ones to incorporate system dynamics. These result in increased dimensionality of the data. Process datasets generally contain a huge number of variables and these approaches blow up the dimensionality of the problem and are not suitable for big data problems. Although various versions of DPLS have been proposed that do not have this issue of augmentation, they do not consider the aspect of temporal correlation or slowness.

Temporal correlation is an important aspect of time series data. Hence, the latent variables that underlie the data set, have strong temporal correlations or in other words, must be smoother or slowly varying. Slow feature analysis (SFA) is an unsupervised method of extracting features subject to temporal slowness from a time series dataset [4]. The objective is to obtain slowly varying features that contain useful information about the process. This approach is particularly suitable for chemical processes whose dynamics is slow by nature and hence the primary latent variables of the data must be slow in nature. Hence the slower latent variables contain key information about the data and faster ones are usually associated with noise. As discussed in section 1.2.1, SFA and its variants have been increasingly studied for industrial applications. As pointed in that section, the SFA-based algorithms where the objective is to perform supervised learning using the extracted FSs, do not extract the SFs in a supervised manner. This thesis hence proposes to achieve this goal by combining the temporal slowness aspect of SFA with the output-relevance aspect of PLS.

Shang *et al* [114] have proposed an approach in which the temporal smoothness aspect is brought into DPLS as an L2 regularization term, regularizing the differ-

ence between the weights of successive samples in the DPLS objective function. The method resulted in an enhanced performance when compared against the unconstrained DPLS method. This approach is applicable to DPLS, but cannot be applied to PLS as it does not consider temporal slowness directly, rather achieves it by regularizing weights. This work proposes an approach that extracts SFs in a supervised manner resulting in SFs that explain the output data well. The objective of the proposed formulation is a combination of PLS and SFA methods. The combined objective results in features with the properties of high covariance with outputs while retaining temporal slowness. This work also proposes solutions to the PLS-SFA formulation which are similar to the conventional PLS algorithms but with modifications. The formulations of the conventional PLS methods result in objective functions whose solutions are given by extracting eigenvectors from a certain matrix. Two of the popular algorithms for solving PLS objective function are nonlinear iterative partial least squares (NIPALS) [103] and statistically inspired modification of partial least squares (SIMPLS) [104]. These algorithms provide a way of systematically solving the objective function by extracting latent features in a stage-wise manner. It will be shown that the proposed output-relevant SF extraction formulation too results in an objective function whose solution will be given by successive extraction of eigenvectors from a certain matrix. It will be shown that the solutions to the proposed formulations will proceed similarly to the NIPALS and SIMPLS algorithms with proposed modifications to result in the temporal slowness of the extracted features.

The proposed approaches are implemented in three case studies. The first of these is a simulated example. The second one is the data from a debutanizer column. This is an open-source industrial dataset [115]. The final case study has an experimental dataset obtained from a hybrid tank system. The obtained results from these case studies are compared with conventional linear models PCR, PLS, and SFR. For time-varying systems, adaptive modeling strategies like recursive PLS [21] and just-in-time [116] have been proven to be more effective. But since the proposed method models a static model and is not adaptive in nature, the comparisons are limited to linear soft sensor models which are not adaptive in nature. The contributions of this work are as follows.

1. Formulation of the problem to extract output-relevant SFs, which combines the aspects of both PLS and SFA.

2. Algorithms for solving the proposed formulation.

The rest of this chapter is organized as follows. In section 3.2 the conventional SFA problem is revisited. In section 3.3 the PLS model is discussed. In section 3.4 the proposed formulations of output-relevant SF extraction and modifications to NIPALS and SIMPLS are presented. Results from the three case studies are presented in section 3.5. Finally, section 3.6 summarizes the proposed approaches and the conclusions drawn from the results.

3.2 SFA

The conventional SFA was introduced in section 2.2.1 as proposed by the original authors [4]. This section revisits it briefly to introduce the objective function in the concise notation that will be used in this chapter. The SFA extracts LVs by minimizing the velocity of the LVs. For a linear model, the objective is to obtain a weight matrix w that map the data matrix $X = [x_1, x_2, ..., x_n]'$ such that the velocity of the scores Xw is minimized. This is same as minimizing the signal ΔXw where ΔX is the temporal difference data matrix calculated as $\Delta X = [x_2 - x_1, x_3 - x_2, \ldots, x_n - x_{n-1}]$. This results in the following objective function.

$$\begin{aligned} \underset{w_a}{\text{minimize}} \quad & w_a' \Delta X' \Delta X w \\ \text{subject to} \quad & w' X' X w = I \end{aligned} \tag{3.1}$$

Here, I is an identity matrix. The constraint in the above equation realizes the constraints in (2.31), (2.32), and (2.33). The above problem can be solved as a generalized eigenvalue problem [55].

3.3 PLS

As discussed in section 2.1, the PLS algorithms are of many types, the primary ones are the NIPALS and SIMPLS. The objective function for the NIPALS is as follows.

$$\begin{aligned} \underset{w_a}{\text{maximize}} \quad & w_a' X_a' Y_a Y_a' X_a w_a \\ \text{subject to} \quad & w_a' w_a = 1, \quad t_b' t_a = 0 \quad \text{for} \quad a > b \end{aligned} \tag{3.2}$$

And the SIMPLS objective function is as follows.

$$\begin{aligned} &\underset{p_a, c_a}{\text{maximize}} \quad w_a' X' Y c_a \\ &\text{subject to} \quad w_a' w_a = 1, \quad c_a' c_a = 1, \quad t_b' t_a = 0 \ \text{ for } \ a > b \end{aligned} \tag{3.3}$$

3.4 Output-relevant SFA

In PLS, feature extraction is performed using both input and output data so that the extracted features aid in improved prediction ability of the model. The feature extraction in PLS is performed considering the covariance between the extracted features from the input and output spaces. For these reasons, in the proposed approach SFA is combined with PLS to obtain output-relevant SFs. NIPALS and SIMPLS algorithms are among the popular algorithms used to solve PLS problems. In this section, the proposed formulation of the optimization problem of these two algorithms, and the solutions to obtain output-relevant SFs are presented.

3.4.1 NIPALS for slow feature extraction

The NIPALS objective function for PLS as expressed in (3.2) maximizes the squared covariance of the scores of X with the output. Since the objective in the proposed case is to obtain slower features with the output-relevance aspect, the proposed objective function combines the objective of SFA in (3.1) and the objective of the NIPALS method. The combined objective function is expressed as follows.

$$\begin{aligned} &\underset{w_a}{\text{maximize}} \quad w_a' X_a' Y_a Y_a' X_a w_a - \alpha \; w_a' \Delta X_a' \Delta X_a w_a \\ &s.t \qquad\qquad\quad w_a' w_a = 1 \end{aligned} \tag{3.4}$$

This formulation will achieve two objectives. The first term is the PLS objective function while the second one is the SFA objective function. The two objectives can be appropriately weighed using the weight α. This is a hyper-parameter and the procedure for tuning α is presented in section 3.4.3. In the conventional SFA (sec. 2.2.1) sphering is performed on the original data followed by PCA on the velocity of the sphered data. So it is similar to solving an optimization problem minimizing the quantity given by $w_a' \Delta X_a' \Delta X_a w_a$. Incorporating this term into the PLS objective function will bring the temporal slowness aspect into PLS. The new objective function in (3.4) can be viewed as a PLS objective function that penalizes the velocities of the

input features, or as an SFA objective function that extracts SFs which are correlated with outputs.

The iterative procedure in Algorithm. 3.1 essentially converges in the following result.

$$w_a \propto X_a' Y_a Y_a' X_a \; w_a - \alpha \; \Delta X_a' \Delta X_a w_a \tag{3.16}$$

This means that w_a is the eigenvector of $X_a' Y_a Y_a' X_a - \alpha \; \Delta X_a' \Delta X_a$. Hence, such an iterative procedure is not required. Nevertheless, it is presented for the sake of completeness as the original NIPALS algorithm followed such an iterative procedure. Algorithm. 3.1 can be simplified as represented in Algorithm. 3.2.

Properties of weights,loadings and scores The weights, scores, and loadings in the NIPALS algorithm have special properties. The four key properties are

1. The weights are orthogonal to each other i.e., $w_i' w_j = 0, \; \forall \; i \neq j$
2. The scores are orthogonal to each other i.e., $t_i' t_j = 0, \; \forall \; i \neq j$
3. The weights are orthogonal to subsequent loadings i.e., $w_i' p_j = 0, \; \forall \; i < j$
4. The loadings are orthogonal in the kernel space of X i.e., $p_i' (X'X)^{-1} p_j = 0, \; \forall \; i \neq j$

The second property particularly implies that the scores are independent of each other which is essential because one would want every latent variable to give new information. The above properties hold for the proposed approach as well. The proofs for the above properties for the NIPALS method are given in Höskuldsson [105]. The proof for the first property for the proposed method is presented here. The matrix deflation equation can be written as

$$X_i = X_{i-1} - t_{i-1} p_{i-1}' = X_{i-1} - t_{i-1} \frac{t_{i-1}' X_{i-1}}{t_{i-1}' t_{i-1}} \tag{3.25}$$

Multiplying both sides by w_{i-1}

$$X_i w_{i-1} = X_{i-1} w_{i-1} - t_{i-1} \frac{t_{i-1}' X_{i-1} w_{i-1}}{t_{i-1}' t_{i-1}} = 0 \tag{3.26}$$

Algorithm 3.1 The modified NIPALS algorithm for output-relevant SFA

1. Take a column of Y_a as an initial guess for u_a. Take a column of ΔX_a as an initial guess for r_a.
2. Calculate w_a as
$$w_a = \frac{1}{u_a' u_a} X_a' \; u_a - \frac{\alpha}{r_a' r_a} \Delta X_a' \; r_a \tag{3.5}$$
3. Normalize w_a.
$$w_a = \frac{w_a}{\sqrt{w_a' w_a}} \tag{3.6}$$
4. Project X_a onto w_a to get scores of X_a. Similarly, project ΔX_a onto w_a to get scores of ΔX_a.
$$t_a = X_a \; w_a; \quad r_a = \Delta X_a \; w_a \tag{3.7}$$
5. Regress Y_a on t_a to get c_a. The regression coefficients c_a contain information about covariance between Y_a and scores of X_a.
$$c_a = \frac{1}{t_a' t_a} Y_a' \; t_a \tag{3.8}$$
6. Normalize c_a.
$$c_a = \frac{c_a}{\sqrt{c_a' c_a}} \tag{3.9}$$
7. Project Y_a onto c_a to get scores of Y_a.
$$u_a = Y_a \; c_a \tag{3.10}$$
8. Iterate from step 2 to 7 till convergence.
9. Regress u_a on t_a. Deflate X_a and Y_a and continue with step 1.
$$b_a = \frac{1}{t_a' t_a} u_a' t_a \tag{3.11}$$
$$p_a = \frac{1}{t_a' t_a} X_a' \; t_a \tag{3.12}$$
$$X_{a+1} = X_a - t_a \; t_a' \; X_a \frac{1}{t_a' t_a} \tag{3.13}$$
$$Y_{a+1} = Y_a - t_a \; t_a' \; Y_a \frac{1}{t_a' t_a} \tag{3.14}$$
10. Iterate from steps 1 to 9 till the required number of features are extracted.
11. Calculate the final regression coefficient matrix as
$$B_{PLS} = W(P'W)^{-1} BC' \tag{3.15}$$

Algorithm 3.2 The simplified version of the modified NIPALS algorithm for output-relevant SFA

1. Find w_a as the largest eigenvector of the matrix $X_a' Y_a Y_a' X_a - \alpha\ \Delta X_a' \Delta X_a$.
2. Project X_a onto w_a to get the scores t_a.

$$t_a = X_a\ w_a \tag{3.17}$$

3. Regress Y_a onto t_a to get c_a.

$$c_a = \frac{1}{t_a' t_a} Y_a'\ t_a \tag{3.18}$$

4. Normalize c_a.

$$c_a = \frac{c_a}{\sqrt{c_a' c_a}} \tag{3.19}$$

5. Find scores of Y_a.

$$u_a = Y_a c_a \tag{3.20}$$

6. Perform regression between output and input scores and deflate the data matrices.

$$b_a = \frac{1}{t_a' t_a} u_a' t_a \tag{3.21}$$

$$p_a = \frac{1}{t_a' t_a} X_a'\ t_a \tag{3.22}$$

$$X_{a+1} = X_a - t_a\ t_a'\ X_a \frac{1}{t_a' t_a} \tag{3.23}$$

$$Y_{a+1} = Y_a - t_a\ t_a'\ Y_a \frac{1}{t_a' t_a} \tag{3.24}$$

7. Continue till the required number of features are obtained.

The above property can be proved for any pair of X_j and w_k with $j > k$. From this result, for the NIPALS method, it can be observed that

$$w_i' w_{i-1} \propto w_i X_i' Y_i Y_i' X_i w_{i-1} = 0 \tag{3.27}$$

For the proposed method, (3.27) becomes

$$\begin{aligned} w_i' w_{i-1} &\propto w_i (X_i' Y_i Y_i' X_i - \alpha \Delta X_i' \Delta X_i) w_{i-1} && (3.28)\\ &= -\alpha w_i (X_i^{t+1} - X_i^t)' (X_i^{t+1} - X_i^t) w_{i-1} && (3.29)\\ &= 0 && (3.30) \end{aligned}$$

Here, X_i^{t+1} is nothing but the X_i matrix without the first sample and X_i^t is the X_i matrix without the last sample. The proofs for the remaining properties are similar to the ones for the NIPALS method and are given in Höskuldsson [105].

3.4.2 SIMPLS for slow feature extraction

Similar to the previous case, the objective function of the SIMPLS algorithm too can be extended to include the slowness aspect into the LVs. To incorporate temporal slowness into the objective function of PLS, the new objective function is formed by adding the temporal slowness term similar to the previous case ((3.4)). The resulting objective function is expressed as

$$\begin{aligned} \underset{w_a, c_a}{\text{maximize}} \quad & w_a' X' Y c_a - \alpha \; w_a' \Delta X' \Delta X w_a \\ \text{subject to} \quad & w_a' w_a = 1, c_a' c_a = 1 \;\text{ and }\; t_b' t_a = 0 \;\text{ for }\; a > b \end{aligned} \tag{3.31}$$

The above formulation in this form does not have a solution in terms of eigenvectors. It can be modified to convert it into a form such that the solutions can be expressed in terms of eigenvectors. The decision variables, w_a and c_a can be concatenated to get the decision variable z_a.

$$z_a = \begin{bmatrix} w_a \\ c_a \end{bmatrix} \tag{3.32}$$

The objective function can now be expressed in terms of z_a

$$\underset{z_a}{\text{maximize}} \quad z_a' Q_a z_a \tag{3.33}$$

$$Q = \begin{bmatrix} -\alpha \Delta X' \Delta X & \frac{1}{2} X' Y \\ \frac{1}{2} Y' X & 0 \end{bmatrix} \tag{3.34}$$

The solution of the above objective is a simple eigenvalue-eigenvector solution with z_a being the decision variable. The Q matrix of the proposed formulation can be written as

$$Q = \begin{bmatrix} D & \frac{1}{2}S \\ \frac{1}{2}S' & 0 \end{bmatrix} \tag{3.35}$$

where $D = -\alpha \Delta X' \Delta X$ is the velocity covariance matrix of inputs and $S = X'Y$ is the covariance between inputs and outputs. Looking at the Q matrix and (3.35), in the modified approach the matrices to be deflated are S and D. The deflation of S is performed similarly to SIMPLS as the constraints for the SIMPLS-SFA objective are the same as that of SIMPLS (since the matrix deflation equations arise from the constraints). The matrix D is deflated as given by the (3.45). The modified SIMPLS algorithm for supervised SF extraction is given as in Algorithm. 3.3. Although SIMPLS is computationally more effective than NIPALS, with today's computational advances, both methods can be easily implemented. Given the more intuitive definition of scores and weights in SIMPLS, it could be preferred over NIPALS in general.

Properties of weights,loadings and scores In the SIMPLS method, the input scores are orthogonal to each other. Also, the input weights are orthogonal to the loadings [104]. This is because in every step the input weights are projected onto a subspace orthogonal to the previous loadings. This is achieved by projecting S repeatedly onto a subspace orthogonal to the loadings. Since the weights are eigenvectors of SS', the subsequent weights also will be orthogonal to the previous loadings. This aspect is crucial in achieving independent scores. Since a similar approach is followed in the proposed method, the proposed method also has similar properties, i.e., the scores are orthogonal to each other and weights are orthogonal to loadings. In the proposed method, in addition to S, D is also deflated. If one looks at the objective function (equation (3.31)), in the slowness term, since w_a appears on both sides of the matrix, to achieve the orthogonal subspace projection of w_a, the projection matrix must appear on both sides of the velocity covariance matrix. Hence, the D matrix must be deflated as in (3.45). These considerations lead to orthogonality between weights and loadings and hence orthogonality between the scores in the proposed method.

Algorithm 3.3 The modified SIMPLS algorithm for output-relevant SFA

1. Calculate the matrices D_a and S_a

$$D_a = -\alpha \Delta X_a' \Delta X_a; \quad S_a = X_a' Y_a \tag{3.36}$$

2. Form the Q_a matrix as

$$Q_a = \begin{bmatrix} D_a & \frac{1}{2} S_a \\ \frac{1}{2} S_a' & 0 \end{bmatrix} \tag{3.37}$$

3. Find the largest eigenvector of the matrix Q_a which yields the weight vectors, w_a and c_a.
4. Project X onto w_a to get the scores of X.

$$t_a = X w_a \tag{3.38}$$

5. Normalize t_a and scale w_a using the norm of t_a.

$$w_a = \frac{w_a}{\sqrt{t_a' t_a}}; \quad t_a = \frac{t_a}{\sqrt{t_a' t_a}} \tag{3.39}$$

6. Find the X and Y loadings, p_a and z_a.

$$p_a = X' t_a; \quad z_a = Y' t_a \tag{3.40}$$

7. Project Y onto z_a to get the scores of Y

$$u_a = Y z_a \tag{3.41}$$

8. Make the current loading orthogonal to the previous loadings.

$$v_a = p_a; \quad v = v_a - V V' p_a \tag{3.42}$$

9. Normalize v_a.

$$v_a = \frac{v_a}{\sqrt{v_a' v_a}} \tag{3.43}$$

10. Deflate the matrices S_a and D_a

$$S_{a+1} = (I - v_a v_a') S_a \tag{3.44}$$
$$D_{a+1} = (I - v_a v_a') D_a (I - v_a v_a') \tag{3.45}$$

Or, the Q_a matrix can be directly deflated as

$$Q_{a+1} = \begin{bmatrix} I - v_a v_a' & 0 \\ 0 & I \end{bmatrix} Q_a \begin{bmatrix} I - v_a v_a' & 0 \\ 0 & I \end{bmatrix} \tag{3.46}$$

11. Repeat the above steps till the required number of features are extracted.

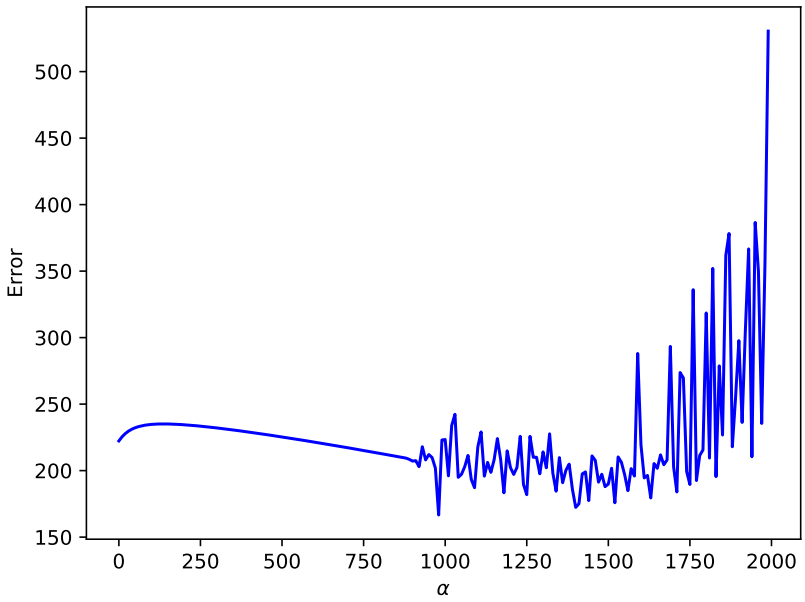


Figure 3.1: The behavior of error with α for the modified NIPALS method

3.4.3 Tuning α and the definiteness of the matrices in the objective functions

In both of the proposed methods, the objective function contains two parts. One is the covariance between the output and input matrices, and the other is the velocity covariance matrix whose relative importance is controlled by the hyper-parameter α.

In the case of the NIPALS approach shown in (3.4), the matrix that appears in the objective function is the difference between the square of the covariance between input and output, and the variance of the velocities of the input scores (weighted with α). Eigenvector extraction hence is performed on the matrix resulting from the difference between the two matrices. Both of these matrices, $X'YY'X$ and $\Delta X'\Delta X$ are positive definite matrices. But since a difference of these is taken, the resulting matrix need not be positive definite. As the value of α increases, at some point, the matrix loses its positive definiteness and becomes an indefinite matrix and then may become a negative semi-definite or negative definite matrix. In the results, it has been found that as α is increased, the prediction errors on training and validation sets decrease steadily. But beyond a certain value of α, the prediction errors increase

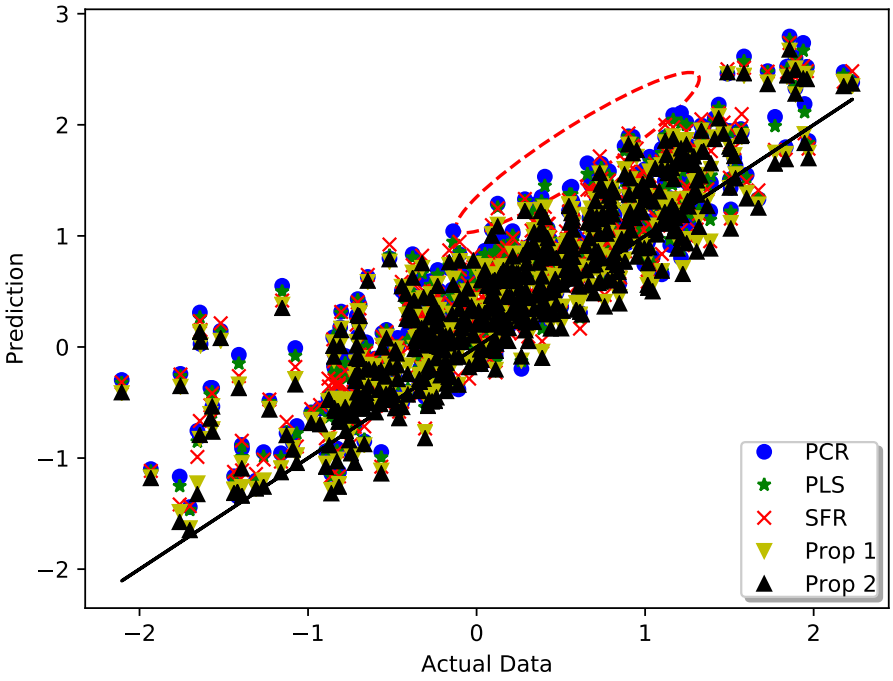


Figure 3.2: Plot of predicted response against the actual data for the test data of the simulated dataset

and start to oscillate. It has been observed that this behavior starts when the highest eigenvalue of the matrix $X'YY'X - \alpha\ \Delta X'\Delta X$ becomes zero. Hence one needs to stop at a point before the value of α beyond which the errors become unstable. Fig. 3.1 depicts such behavior for the Debutanizer column case study. In this case, an α value of 850 is chosen. For the SIMPLS case too, similar behavior is observed when the highest eigenvalue of the Q matrix approaches zero with an increase in α. In this case also, α cannot be increased beyond this point.

3.5 Results

The proposed methods have been applied to three case studies. This section presents the results obtained.

3.5.1 Simulated case study

The data for this case study is created using the probabilistic slow feature analysis (PSFA) model. The PSFA model is a state-space model described by the following equations.

$$s_t = As_{t-1} + v_t; \quad v_t \sim \mathcal{N}(0, \Sigma_s) \tag{3.47}$$

$$x_t = Hs_t + w_t; \quad w_t \sim \mathcal{N}(0, \Sigma_x) \tag{3.48}$$

Here s represents the SFs and x represents the observed inputs. The matrix A defines how SFs evolve, and is diagonal to ensure that the SFs are independent of each other. Matrix H describes how the SFs generate the observed data. v and w are Gaussian white noise. Using the PSFA model, eight features with different velocities are generated. Five inputs are generated using a linear transformation of all eight features through matrix H. The elements of the matrix H are sampled from a discrete uniform distribution ranging from -10 to 10 (although some changes are made after sampling). Two SFs are used to construct the output (equation 3.51). This is according to the rationale that the observed input data that has underlying SFs and these SFs result in the outputs. The following parameters are used to construct the simulated dataset.

$$A = diag(0.999, 0.99, 0.97, 0.95, 0.94, 0.93, 0.1, 0.05) \tag{3.49}$$

$$H' = \begin{bmatrix} -1 & 0 & 0 & 1 & 5 \\ -1 & 1 & -5 & 5 & 7 \\ -10 & 6 & -9 & 2 & 9 \\ -3 & 3 & -4 & 8 & 6 \\ -5 & 8 & 1 & 0 & -2 \\ 4 & 8 & -6 & -1 & -2 \\ 7 & -10 & 3 & -1 & 4 \\ -1 & -3 & -9 & -10 & 0 \end{bmatrix} \tag{3.50}$$

$$y(t) = Bs(t) + e_y(t) \tag{3.51}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \end{bmatrix} \tag{3.52}$$

A total of 2000 samples are generated using (3.47), (3.48) and (3.51). The dataset is divided into three parts named training, validation, and test sets. The training dataset is the one used to train the model to get the model parameters. This is the

set that is used while solving the objective function of the optimization problems. The validation dataset is used to tune the hyper-parameters. In this case, the hyper-parameters are the number of features to be extracted and the weighting factor α in the objective functions of proposed approaches ahown in (3.4) and (3.31). The combination of the number of latent variables and α that gives a low error on the validation dataset are chosen. This is illustrated in the second case study. The test dataset is the set unseen by the model during training. Hence, the performance of the model on the test dataset is the true indication of the generalization ability or the model prediction. For this case study, the data split is done in the ratio of 50:25:25. The test data comes after the validation data which comes after the training data. The proposed approaches are compared against conventional linear models, PCR, PLS and SFR w.r.t root mean squared error (RMSE) values and concordance correlation coefficient. The Concordance correlation coefficient is a measure of the agreement between two variables. The concordance correlation coefficient ρ_c, between two variables x and y is expressed as

$$\rho_c = \frac{2\rho\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2} \tag{3.53}$$

Table 3.1: Comparing the RMSE values for training, validation and test sets of conventional linear regression methods and proposed methods for the simulated dataset

Method	n	$RMSE_{train}$	$RMSE_{val}$	$RMSE_{test}$
PCR	5	0.4395	0.4730	0.5448
PLS	4	0.4412	0.4717	0.5269
SFR	3	0.4605	0.4912	0.5275
Prop - 1	2	0.4580	0.4773	0.4596
Prop - 2	2	0.4682	0.4895	0.4510

In the tables and figures, proposed method-1 and proposed method-2 refer to the proposed modified NIPALS and modified SIMPLS methods respectively. A plot of predicted response versus the actual data for the test dataset is presented in Fig. 3.2. The black line in this figure is the 45° line (when observed and predicted values are equal) and the points are expected to lie close to this line. It can be observed from the figure that the points corresponding to the proposed methods, particularly from the

Table 3.2: Comparing the Concordance correlation coefficient values for training, validation and test sets of conventional linear regression methods and proposed methods for the simulated dataset

Method	n	$\rho_{c_{train}}$	$\rho_{c_{val}}$	$\rho_{c_{test}}$
PCR	5	0.9123	0.8718	0.7851
PLS	4	0.9116	0.8718	0.7975
SFR	3	0.9030	0.8587	0.7977
Prop - 1	2	0.9041	0.8682	0.8405
Prop - 2	2	0.8993	0.8620	0.8457

SIMPLS based method, lie closer to the 45° line. The points corresponding to PCR, PLS, and SFR are mostly on one side of the line resulting in a higher RMSE. The obtained results are quantified in the Table. 3.1 and 3.2. The slowness term on the proposed objective functions can be seen as a regularization term. So the error on the training dataset for the proposed approaches is more than that for the conventional approaches as expected since by regularization one sacrifices the performance of the model on the training set in terms of fitting so that the model has better generalization ability. It can be observed that the proposed approaches have improved the RMSE. To test if the RMSE of the proposed methods is significantly smaller than the RMSE of the conventional methods, a t-test can be performed. The null hypothesis is that the RMSE values are not significantly different and the alternate hypothesis being RMSE of conventional methods is larger than that of the proposed methods. Since PCR and PLS are the most common conventional methods, a t-test is performed here with the best of the RMSEs among PCR and PLS, which is PLS in this case. Different estimates of the RMSE are obtained by randomly sampling from the test data without replacement such that all the samples in the test dataset are used. The t statistic for the two-sample test is calculated as

$$t = \frac{(\overline{x}_1 - \overline{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{3.54}$$

Here $\overline{x}$, μ and s^2 are the sample mean, population mean, and sample variance respectively. Since, the null hypothesis is that the RMSE values are same, $\mu_1 - \mu_2 = 0$. Hence the t-statistic is

$$t = \frac{(RMSE_{PLS} - RMSE_{prop})}{\sqrt{\frac{s_{PLS}^2}{n} + \frac{s_{prop}^2}{n}}} \tag{3.55}$$

The degrees of freedom is calculated using the following equation.

$$dof = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{1}{(n_1-1)}\left(\frac{s_1^2}{n_1}\right)^2 + \frac{1}{(n_2-1)}\left(\frac{s_2^2}{n_2}\right)^2} \tag{3.56}$$

Twenty-five sets of samples are obtained from the test data and the mean of RMSE for PLS is 0.5207 and for the proposed methods is 0.4523 and 0.4435 respectively (The values in Table. 3.1 are for the whole test data). The critical value of the statistic for a one-sided, 95% confidence interval test is $t_{0.95,48} = 1.6772$. The t values for the two proposed methods are 2.9180 and 3.2869 respectively. Hence it can be concluded that the proposed methods have reduced the RMSE. The proposed approaches have also improved the concordance between the predicted output and the actual output. In the table, the column labeled n represents the number of latent variables required by the corresponding method. It can be seen that the proposed methods require a lesser number of latent variables than the conventional methods in order to explain the outputs. It can be inferred that the proposed approaches are efficient in extracting the hidden features. The lesser number of latent variables can be advantageous as they help in reducing the variance of the predictions as well as reliability for practical applications.

3.5.2 Debutanizer column

The debutanizer column dataset is a benchmark industrial dataset for soft sensor designing [115]. The debutanizer column is used to remove butane and other lighter hydrocarbons from naphtha. The objective is to design a soft sensor to estimate the butane content in the bottoms stream of the debutanizer column. There are 7 input variables: top temperature, top pressure, reflux flow, flow to the next process, the temperature of the sixth tray in the column, and the bottom temperature measured at two points. The schematic of the process is depicted in Fig. 3.3, and the description of the variables is shown in Tab. 3.3 [115]. There are a total of 2393 samples in the data set. The first 2200 are used in this case study. The data is split into three parts such that the training data accounts for a good amount of variations in the variables. In this case, the points are split as 1200-400-600 is used for training, validation, and testing purposes.

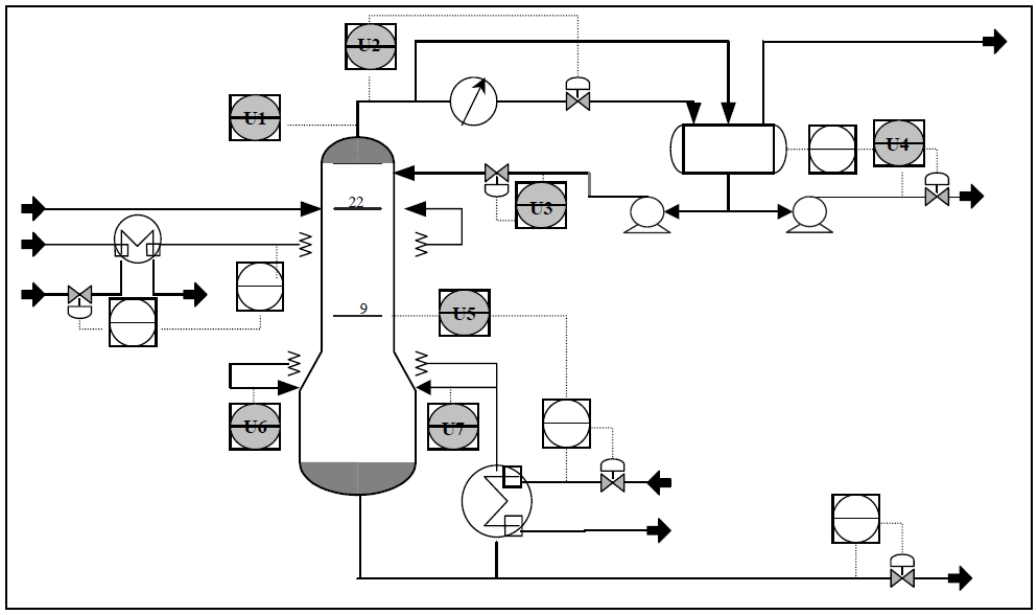


Figure 3.3: A schematic disgram of the debutanizer column [115]

Table 3.3: Description of the variables in the debutanizer column process dataset

Variable	Description
$x^{(1)}$	Top temperature
$x^{(2)}$	Top pressure
$x^{(3)}$	Reflux flow
$x^{(4)}$	Flow to next process
$x^{(5)}$	6^{th} tray temperature
$x^{(6)}$	Bottom temperature
$x^{(7)}$	Bottom pressure
y	C4 content in the debutanizer bottoms

Similar to the previous case, proposed methods are implemented on the dataset and the performance is compared with PCR, PLS, and SFR. For this case, it is found that taking just the seven regressors gives poor results. Hence for each input variable, a lagged variable is also taken to enhance the performance of the soft sensor. The number of lags is decided based on the correlation of the lagged input variable with the output. For each variable, a lag of up to 10 samples is considered and the lag that gives the highest correlation with the outputs is selected. Augmenting lagged measurements does increase the dimensionality of the problem. But for this case, a static model gave a poor performance, and hence a dynamic model is used. Hence in actuality, the comparison here would be made between the dynamic version of

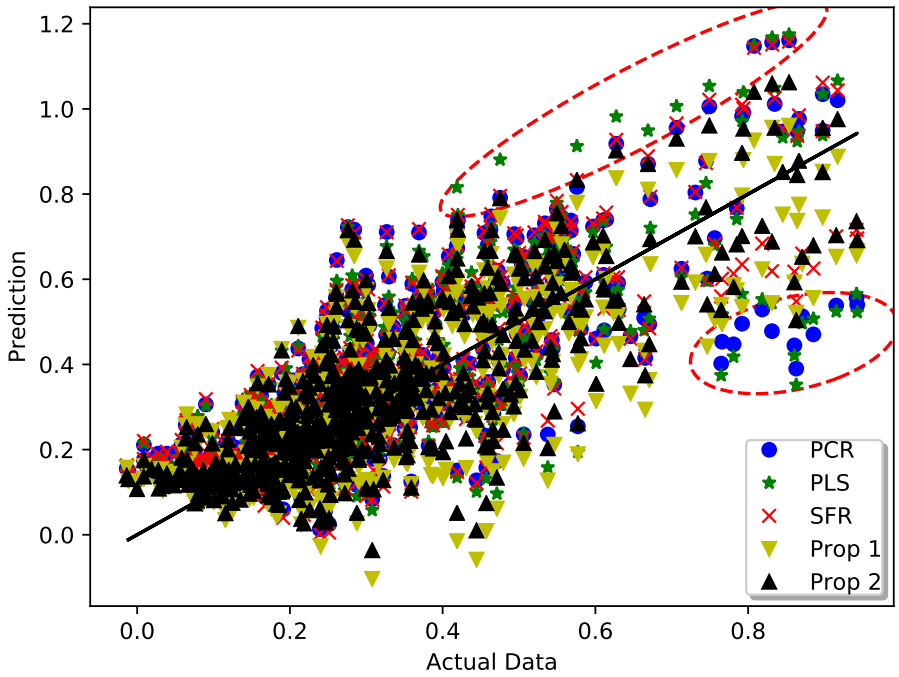


Figure 3.4: Comparison of the output predicted by PCR, PLS, SFR and proposed methods on the test data of the debutanizer column dataset

the proposed approach and DPCR, DPLS, and DSFR. Due to the augmentation of the lagged variables, the velocity of the features will appear twice in the velocity covariance matrix for most of the samples (in the second term of the objective in (3.4) and (3.31)). But this will be taken care of by the weighting factor α which is tuned using the validation dataset similar to the previous case.

The plot of predictions versus the observations for the five methods is depicted in Fig. 3.4. The proposed methods result in a tighter spread around the 45° line. This can be seen particularly when the response is above 0.75. The comparison results w.r.t RMSE and ρ_c are tabulated in Table. 3.4 and 3.5. Similar to the previous case, the proposed methods have resulted in reduced RMSE and increased concordance, particularly in the case of SIMPLS based method. To test the significance of the results the t-test is done similar to the previous case study. A t-test is conducted between the proposed methods and PCR (in this case PCR is better than PLS in terms of RMSE) to test the significance in RMSE reduction. The t-statistic values

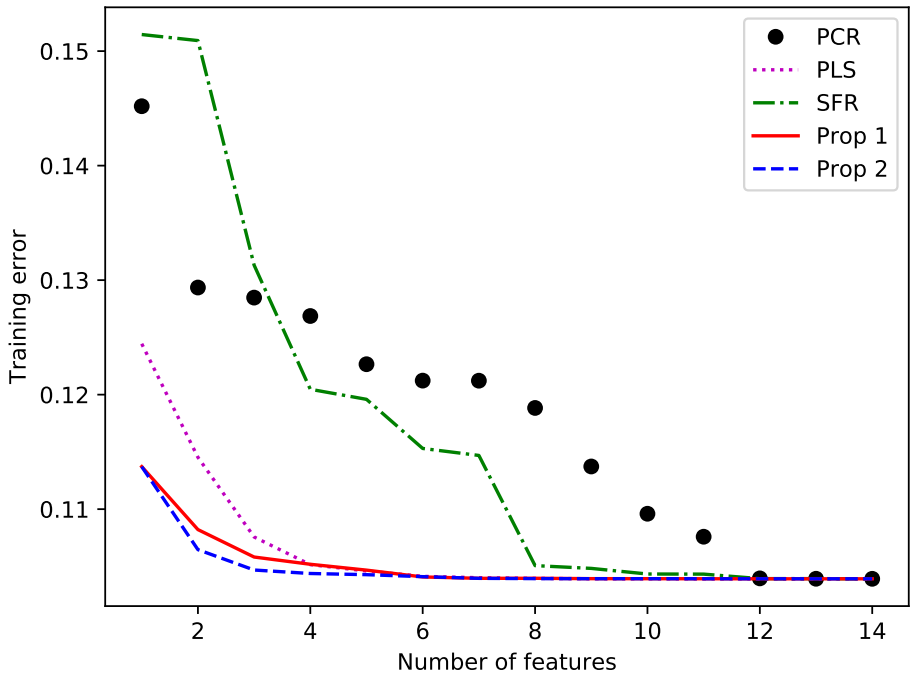


Figure 3.5: Comparison of the variation of training error with the number of features

for the proposed methods are 1.2929 and 2.4031 while the critical values are $t_{0.05,48}$ = 1.6722 and $t_{0.05,47}$ = 1.6780 respectively. The second proposed method passes this test while the first one does not. The p-value for the first proposed method is 0.1011. Nevertheless, the proposed method uses a significantly less number of latent variables than the PCR. The proposed methods require the least number of features to describe the dataset. Due to the augmentation of lagged samples, there can be a maximum of 14 latent variables. While PCR, PLS, and SFR required 12, 4, and 10 latent variables respectively to explain the observed outputs, the proposed methods give enhanced results with just two latent variables. The proposed approaches hence are efficient at the extraction of relevant features for the outputs. This can be seen in Fig. 3.5 and 3.6. It can be seen that the training error monotonously decreases with added features and the error for the proposed methods is lower than the conventional methods. The validation dataset is used to tune the hyper-parameter α and also select the number of features. It can be observed that the validation error for the proposed methods drops quickly to low values when compared with the conventional methods. The number of

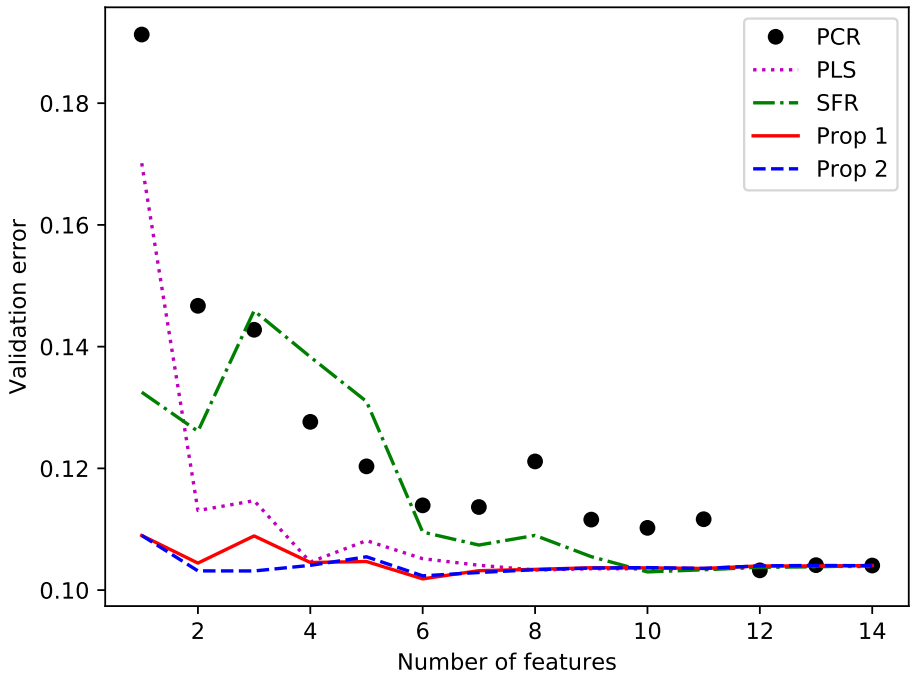


Figure 3.6: Comparison of the variation of validation error with the number of features

features required to describe the data is selected when a low value of validation error is obtained.

3.5.3 Hybrid tank system

The data for this case study is taken from a hybrid tank system containing three tanks. The experimental setup is located in the process control laboratory at the University of Alberta. The soft sensor is developed to predict the level of water in the middle tank. The experimental setup is shown in Fig. 3.7.

The system consists of three vertical tanks at the top and a storage tank at the bottom. There are two pumps each on the left and right side to pump water into the left and right tanks respectively. There are nine valves indicated as V1 to V9 in the figure. Valves V1 to V4, V6, and V8 facilitate water flow between middle and side tanks. Valves V5, V7, and V9 connect the three vertical tanks and the storage tank. Except for valves V1 and V2, all other valves were kept open during the course of the experiment. The liquid level in the middle tank is kept around the halfway

Table 3.4: Comparing the RMSE values for training, validation and test sets of conventional linear regression methods and proposed methods for the debutanizer column dataset

Method	n	$RMSE_{train}$	$RMSE_{val}$	$RMSE_{test}$
PCR	12	0.1040	0.1032	0.1348
PLS	4	0.1052	0.1046	0.1373
SFR	10	0.1043	0.1030	0.1288
Prop - 1	2	0.1082	0.1044	0.1273
Prop - 2	2	0.1065	0.1032	0.1217

Table 3.5: Comparing the Concordance correlation coefficient values for training, validation and test sets of conventional linear regression methods and proposed methods for the debutanizer column dataset

Method	n	$\rho_{c_{train}}$	$\rho_{c_{val}}$	$\rho_{c_{test}}$
PCR	12	0.6923	0.6888	0.7457
PLS	4	0.6829	0.6802	0.7350
SFR	10	0.6893	0.6897	0.7793
Prop - 1	2	0.6576	0.6850	0.7589
Prop - 2	2	0.6721	0.7033	0.7940

mark of the total height, i.e., between the valves V2 and V4. Ten input variables are chosen. These are the liquid level, pump outlet flow rate, pump speed, primary and slave controller outputs that set the level inside the tank and the pump speed. Two sets of each of these five variables are taken, one for the left side and the other for the right side.

Similar to the previous case studies, the proposed approaches are implemented on the dataset and compared against PCR, PLS, and SFR and the results are as given in Table. 3.6 and 3.7.

The RMSE on the test data of the proposed methods is lower than that of the existing linear regression methods. Similar to the previous case studies, a t-test is performed to check for if the reduction in RMSE is significant or not as compared with PCR. The t values are 3.6473 and 8.8987 while the critical value are 1.6722 and 1.6780 respectively. Both the proposed methods pass the test implying a significant reduction in the RMSE of the proposed methods. The concordance correlation coefficient between data and predictions by the proposed approaches is higher than the conventional methods. It must also be noted that the number of latent features

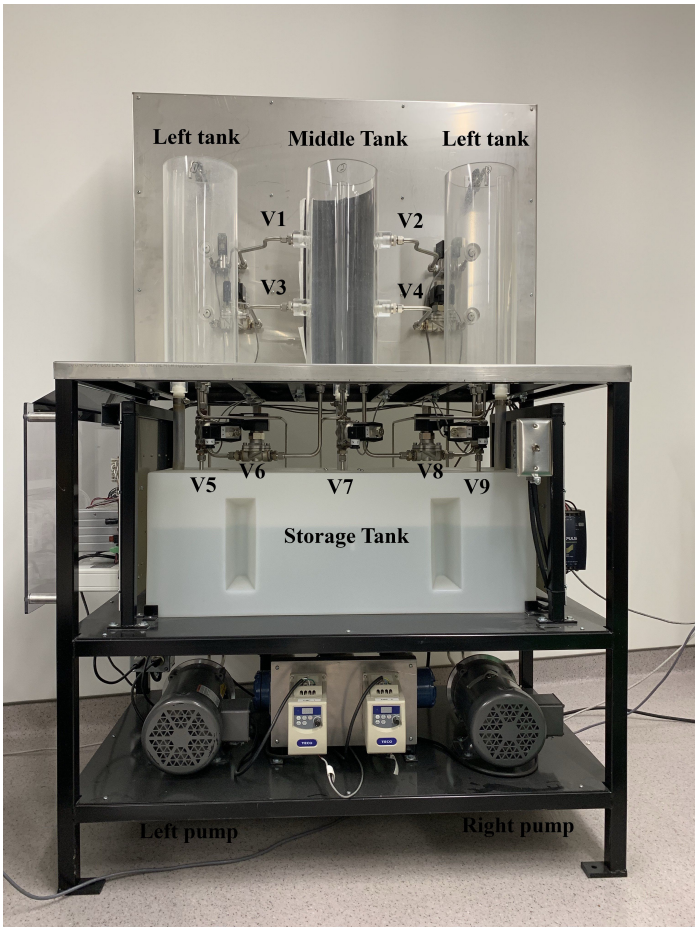


Figure 3.7: Experimental setup of the hybrid tank system

required for the proposed method is lower than that of the conventional methods. All these observations follow a trend similar to the previous case studies. In Fig. 3.8, the predictions by the five methods are plotted against the test data. To avoid cluttering of the points, the data is down-sampled and then plotted in this figure. The SIMPLS based method results in a spread on both sides of the 45° line and the spread is tighter than the PCR, PLS, and SFR methods. For the NIPALS based method, although one-sided, the points are closer to the 45° than the conventional methods.

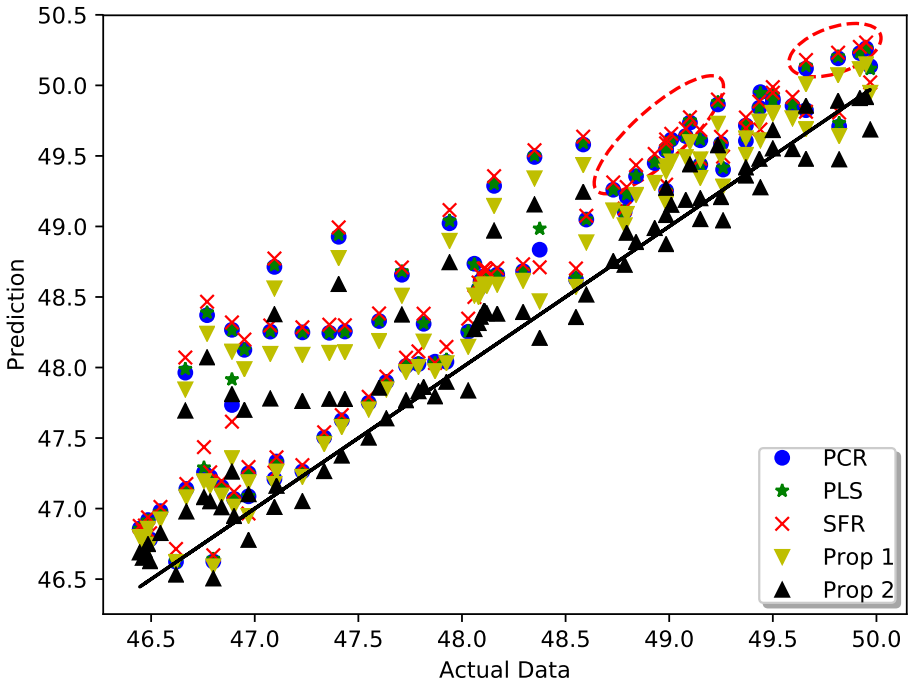


Figure 3.8: Comparison of the liquid level in the middle tank predicted by PCR, PLS, SFR and proposed methods on the test dataset

Table 3.6: Comparing the RMSE values for training, validation and test sets of conventional linear regression methods and proposed methods for the hybrid tank system dataset

Method	n	$RMSE_{train}$	$RMSE_{val}$	$RMSE_{test}$
PCR	6	0.4351	0.4963	0.6326
PLS	4	0.4347	0.4983	0.6385
SFR	6	0.4413	0.5140	0.6673
Prop - 1	3	0.4391	0.4314	0.5354
Prop - 2	3	0.4493	0.4038	0.4034

Table 3.7: Comparing the Concordance correlation coefficient values for training, validation and test sets of conventional linear regression methods and proposed methods for the hybrid tank system dataset

Method	n	$\rho_{c_{train}}$	$\rho_{c_{val}}$	$\rho_{c_{test}}$
PCR	6	0.8510	0.7040	0.9280
PLS	4	0.8514	0.7069	0.9285
SFR	6	0.8453	0.7040	0.9144
Prop - 1	3	0.8474	0.7066	0.9486
Prop - 2	3	0.8379	0.6854	0.9672

3.6 Conclusion

A new approach for extracting output-relevant SFs is presented in this article. The method combines the elements of output-relevant feature extraction from PLS and feature extraction with temporal slowness from SFA. The proposed method helps to bring the temporal correlation aspect into PLS, which is essential for time-series data. Similar to PLS, the proposed problem also results in a solution that can be expressed in terms of eigenvectors of certain matrices. Algorithms to solve the proposed problem have also been presented which are modified versions of popular PLS algorithms NIPALS and SIMPLS and convenient to use. The proposed methods are applied to three case studies for evaluating the effectiveness of the approaches. It is found that the proposed approaches not only result in smaller RMSE of predictions and larger concordance correlation values as compared with PCR, PLS, and SFR, but also require a lesser number of latent features to explain the relationship between the inputs and the outputs. This observation implies that the proposed approach extracts more meaningful features than the conventional methods.

Chapter 4

Siamese Neural Network-Based Supervised Slow Feature Extraction *

In this chapter, the idea of supervised SF extraction is extended to nonlinear systems. Although various nonlinear modeling techniques exist, this thesis explores the usage of deep learning techniques due to their rich representational ability. Artificial neural networks, owing to their ability to model complex nonlinearities, can be used to extract slow features (SFs) from a dataset obtained from a complicated process. A special type of network called the Siamese neural network can be used for this purpose. Siamese neural networks have a provision of handling two samples at a time and this aspect helps in extracting SFs. In this chapter, we present two approaches that extract SFs using Siamese neural networks in a supervised manner. The output-relevance aspect is brought into feature extraction as a regularization term in the objective function of the Siamese neural network. Such regularization improves the performance of the neural network model. The proposed approaches are implemented on three datasets to demonstrate their effectiveness.

4.1 Introduction

Machine learning algorithms can be categorized into unsupervised, supervised and reinforcement learning categories and soft sensors fall under the category of supervised

*This chapter has been published as **R. Chiplunkar** and B. Huang, "Siamese Neural Network-Based Supervised Slow Feature Extraction for Soft Sensor Application," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 9, pp. 8953-8962, 2021.

learning. Soft sensor models can be broadly classified as linear and nonlinear methods. Linear methods for soft sensors are mostly based on principal component regression (PCR) and partial least squares (PLS). Since most industrial processes are nonlinear, methods like kernel PLS [118], support vector regression [119], and artificial neural networks (ANN) [6] are used.

ANNs are a class of nonlinear models inspired by the network of neurons in the brain. The multi-layered nature of the network results in a good representation power. Algorithms such as greedy layer-wise training [120, 121] enabled the usage of deeper network architectures. Deep learning has a rich representational ability as it can extract low-level features from the data. Deep learning has revolutionized fields such as computer vision, speech recognition, etc [122]. Neural networks have generated a lot of interest in process systems engineering too. Multilayer perceptrons or vanilla feedforward neural networks (FNN) have been used to develop soft sensors for various applications such as the estimation of the concentration of top and bottom products in a debutanizer column [123], estimation of viscosity in a polymerization process [124], monitoring NO_x emissions in industrial boilers [125], and more.

The recent implementations of deep learning techniques have been increasingly focusing on extracting latent features with a certain rationale and can be broadly classified as unsupervised and supervised latent feature extraction methods. Unsupervised layer-wise pre-training methods like deep belief networks [126, 127], extreme learning machine-based stacked autoencoders called hierarchical extreme learning machine (HELM) [128], stacked denoising autoencoders [129] have been used to extract nonlinear latent structures which are then used to perform supervised learning. Recently, the focus has been more towards extracting latent features in a supervised and semi-supervised manner. A technique called the variable-wise weighted stacked autoencoder (VW-SAE) algorithm to perform deep feature extraction in a supervised manner has been developed that trains the autoencoders in a supervised manner [130, 131]. The method is based on SAE, but solves a weighted least squares problem, with weights calculated based on a correlation measure of latent variables with the outputs. A variant of the same algorithm named stacked quality relevant autoencoder in which the latent features extracted using augmented input-output data has also been proposed [132]. The problem of supervised latent variable extrac-

tion has also been explored in the frameworks of variational autoencoders [133, 134]. For processes with long-term dynamic dependencies, soft sensors based on long-short term memory (LSTM) networks have been used. Sun and Ge [135] proposed a method where a restricted Boltzmann machine (RBM) was trained in an unsupervised manner to obtain the latent space which was used to predict the outputs using LSTM. Similar to the supervised latent feature extraction extensions to autoencoder-based methods, supervised LSTM schemes have been proposed in recent times. LSTM architectures that incorporate attributes such as adaptive quality relevant feature selection [136], supervised learning of the latent space by input-output augmentation [137] have been studied. The supervised nature of training has been found to enhance the performance in all these cases.

In this chapter, we combine the two aspects of supervised SF extraction and deep learning, and present two methods in which SFs are extracted using neural networks in a supervised manner. Although nonlinear SFA can be achieved using a nonlinear transformation of the data [4] or by kernel SFA [138], deep learning is more powerful than these approaches as discussed earlier. We have used the Siamese neural networks for this purpose [7]. These networks contain two identical neural networks in parallel. This enables us to handle two samples at a time, a useful feature when considering temporal relations between samples. Siamese networks can also be viewed as manifold embedding algorithms [139]. Siamese networks have been used to perform unsupervised SF extraction in computer vision [140–142]. The focus in computer vision is more on unsupervised feature extraction to obtain meaningful low-level features based on the idea that the human brain processes visual information in an unsupervised manner. In the case of soft sensors, it is more important to generate features that help in predicting the output accurately. Supervised feature extraction results in features that contain information relevant to both inputs and outputs, which leads to better prediction of the outputs; an argument similar to PLS vs PCR. Hence, the proposed methods here focus on the supervised extraction of SFs. In both the proposed methods, this is achieved by adding an extra term in the objective function of the Siamese network. These terms are chosen such that they regularize the training of the Siamese network in a way that results in supervised extraction of SFs. The proposed methods are implemented in three case studies. The

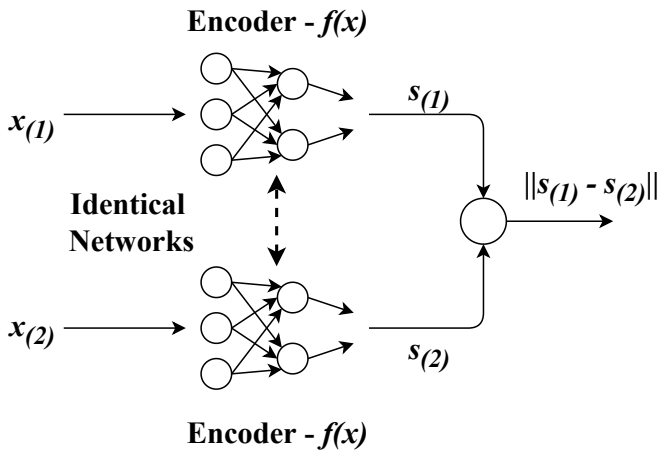


Figure 4.1: Configuration of a Siamese neural network

first one is a simulated dataset. The second one is an open-source industrial dataset of a debutanizer column [115]. The final case study is on an experimental setup of a hybrid tank system.

The rest of the chapter is organized as follows. Section 4.2 gives an overview of the Siamese neural networks. The proposed approaches are presented in section 4.3. The two case studies are presented in section 4.4. Section 4.5 discusses the conclusions.

4.2 Siamese neural networks

The proposed methods are based on SFA and Siamese neural networks. As the SFA is already discussed in section 2.2.1, this section provides an overview of the Siamese neural networks.

Siamese neural networks are a class of neural networks that contain two identical subnetworks. The subnetworks are identical to each other both in structure and parameters. Each of the subnetworks hence provides an identical mapping of inputs to the latent features. The output of the network is a difference metric between the two latent features. Since the two networks are identical, if two very similar inputs are given to the network, then the resulting features also should lie very close to each other in the feature space. Siamese neural networks are used when we need to learn about the similarity or dissimilarity between two inputs. The configuration of the Siamese neural network is depicted in Fig. 4.1. The block named 'Encoder' in

the figure refers to a feedforward network that encodes the input x into the hidden feature s. Both the encoder networks are identical to each other providing the same nonlinear function $f(x)$. The output of the network is the difference metric between the extracted features $s_{(1)}$ and $s_{(2)}$ corresponding to the input samples $x_{(1)}$ and $x_{(2)}$ respectively. This metric indicates if the samples are similar or not. The input to a Siamese network should always be in pairs. The objective function for Siamese networks is given as [143]

$$\mathcal{L} = \begin{cases} d\{f(x_{(1)}), f(x_{(2)})\}, & \text{if } x_{(1)} \text{ and } x_{(2)} \text{ are similar} \\ \max(0, \delta - d\{f(x_{(1)}), f(x_{(2)})\}), & \text{otherwise} \end{cases} . \tag{4.1}$$

Here, $d\{f(x_{(1)}), f(x_{(2)})\}$ is a distance metric between the hidden features $f(x_1)$ and $f(x_2)$. If Euclidean distance is taken as the distance metric then this term can be written as $\left\|f(x_{(1)}) - f(x_{(2)})\right\|_2^2$. The hyper-parameter δ is the minimum distance between the hidden features of the dissimilar samples. Hence the objective during the training of a Siamese neural network is to minimize the distance between the hidden features of two samples if they are similar and to have a minimum distance of δ between the features of dissimilar samples. In the proposed approaches, the notion of similarity between samples is based on the time proximity of the samples. While training the network, the parameters of both networks are updated simultaneously and in a similar way so that the networks represent the same nonlinear function.

4.3 Proposed methods

In SFA, feature extraction is performed such that the obtained features vary slowly or have slow velocities. This means that $s(t)$ will be very close to $s(t-1)$. In other words, the difference between $s(t)$ and $s(t-1)$ will be small. In the case of deep learning, such mappings from x to s can be obtained using Siamese neural networks. Thus, $x_{(1)}$ and $x_{(2)}$ become $x(t)$ and $x(t-1)$ and the distance between $s(t)$ and $s(t-1)$ is minimized. Hence, the objective function for the unsupervised SF extraction using Siamese networks is given by

$$\mathcal{L} = \begin{cases} \|f(x(t_1)) - f(x(t_2))\|_2^2, & \text{if } |t_1 - t_2| = 1 \\ \max(0, \delta - \|f(x(t_1)) - f(x(t_2))\|_2^2), & \text{otherwise.} \end{cases} \tag{4.2}$$

Even though neural networks are a powerful class of nonlinear models, solving SFA using the above formulations as such may not lead to good results. This is because, for different configurations of the network, different sets of SFs will be obtained. This will make it difficult to determine the relevance of these extracted SFs, particularly for supervised learning cases. As stated earlier, for supervised learning applications, the extracted features must be able to explain the observed outputs well. Hence, we propose two methods to perform Supervised Slow Feature Analysis by Siamese Networks. Here onwards, the proposed methods will be referred to as SSFASN1 and SSFASN2 respectively.

4.3.1 SSFASN1

The configuration of the network in this approach is shown in Fig. 4.2. A two-step approach is proposed to train this configuration. In the first step, the network as shown in the figure is used to extract SFs which are decorrelated with each other and these SFs are used to construct the outputs. This results in a supervised SF extraction since only the SFs that can explain the output are extracted. The block named 'Encoder' is a feedforward network that maps the inputs x to the hidden features s. These hidden features are then mapped to the outputs y by the 'Output network' which again is a neural network. The loss function to be minimized for this step is given as

$$\min_{f,g} \underbrace{\sum \|f(x_t) - f(x_{t-1})\|_2)^2}_{\textbf{Slowness of features}} + \underbrace{\alpha \sum (y - g(s))^2}_{\textbf{Output error}} + \underbrace{\beta||cov(f(X)) - I||}_{\textbf{Independence and unit variance}} . \tag{4.3}$$

The above objective has three terms in it

- The first term minimizes the Euclidean distance between the features of two samples that are adjacent to each other in the time series, reflecting the velocity aspect. This term ensures the extraction of hidden features that vary slowly.
- The second term minimizes the error between the observed outputs and the outputs predicted using the extracted SFs. Training the encoder network such

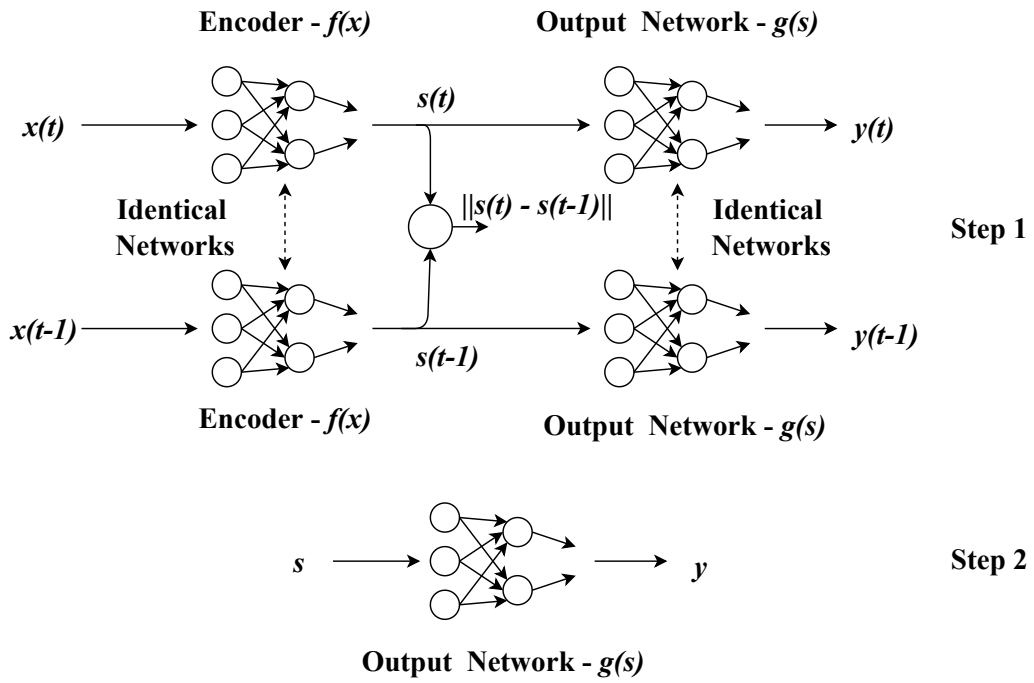


Figure 4.2: Configuration of the network for SSNSFA1. Step 1 and Step 2 architectures correspond to the objective functions in (4.3) and (4.5) respectively.

that the extracted SFs predict the output makes the SF extraction supervised. This term is weighted by the hyperparameter $\alpha > 0$.

- The third term is added to obtain SFs that are decorrelated and have unit variance. Making the covariance matrix of the extracted features an identity matrix ensures this. This term is weighted by the hyperparameter $\beta > 0$.

It is important to note that it is the covariance matrix that needs to be identity and not the correlation matrix. The slowness objective tries to make the variations in features very small and since the neural networks are powerful functions, the features may be mapped to constant signals. This is avoided by the output error term in this approach as the variations in outputs influence the feature space and hence constant signals as features will not be obtained. But, the slowness term will make the variance in features very small. The obtained features in a true sense will not be slow in such a scenario (this is because multiplying a very fast signal by a very small number does not make it a slow signal). Hence the third term is added to achieve this and to incorporate the constraints in the formulation of conventional SFA ((2.31) and

(2.32)).

We can represent each term in (4.3) as inner products. So the slowness term becomes $\Delta f(X)^T \Delta f(X)$, the output error term becomes $(Y - g(f(x)))^T (Y - g(f(x)))$, and the last term is $tr((f(X)^T f(X) - I)^T (f(X)^T f(X) - I))$. If we calculate the gradient of the above objective function with respect to any of the weights in the encoder network, we get

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_f} &= 2\Delta f(X)^T \frac{\partial \Delta f(X)}{\partial w_f} + 2\alpha(Y - g(f(X))) \frac{-\partial g(f(X))}{\partial w_f} \\ &+ 2\beta \sum \left((f(X_i)^T f(X_i) - 1) f(X_i)^T \frac{\partial f(X_i)}{\partial w_f} \right) \\ &+ 2\beta \sum \left((f(X_i)^T f(X_j)) f(X_i)^T \frac{\partial f(X_j)}{\partial w_f} \right). \end{aligned} \tag{4.4}$$

Here, $\mathcal{L}$ is the overall loss function of (4.3) and $f(X)$ is the vector containing latent features of all the samples in the mini-batch. The parameter w_f is any parameter in the encoder network. From (4.4), we can see that the gradients of the loss function w.r.t the encoder network parameters are affected by the output error. Therefore, the updates of the parameters in the encoder network and hence the function realized by the encoder network will depend on the output error. As a result of this, the extracted SFs will be performed in a supervised manner making them relevant to the outputs.

In the second step, the SFs obtained in the first step are used to construct the output. The objective function for this step is

$$\min_g \quad \sum (y - g(s))^2. \tag{4.5}$$

Only the output network is trained in this step. This is because the features obtained from the previous steps are used as inputs. The parameters obtained from the first step are taken as the initial guess and the parameters are tuned according to the objective function in (4.5).

Both steps can be solved using stochastic gradient descent. For (4.3), to ensure that samples are generated in pairs for stochastic gradient descent, the sampling can be done as follows. A random subset of the samples is generated which is to be passed into one half of the network. For every sample taken, its adjacent sample (preceding

or succeeding) is passed through the other half of the network. The same samples are then used to evaluate the loss functions. During the testing phase, only the upper half of the network shown in Fig. 4.2 is used. Once a test sample $\tilde{x}$ is obtained, it is passed through the encoder network to get $\tilde{s} = f(\tilde{x})$ which is then passed through the output network to get the estimated value $\hat{\tilde{y}} = g(\tilde{s})$.

Tuning the weights: It can be noted from the objective function in 4.3 that apart from the weights and biases of the encoder and output networks, there are two hyper-parameters α and β to be tuned. Hence, the data used for learning the model is split into two parts, the training and the validation datasets. The weights and biases are learned directly through the stochastic gradient descent as discussed earlier using the training dataset. Hyperparameters such as α and β, activation functions and their parameters, etc are learned using the validation dataset. The validation dataset is also used for early-stopping of the training procedure so as to avoid over-fitting. All three weights must be carefully tuned such that all the objectives are achieved to a satisfactory level. For example, correlation between $f(X)$ and Y, and $cov(f(X))$ can be calculated for the validation dataset. If a good correlation with outputs and a diagonally dominant covariance matrix for $cov(f(X))$ is obtained, it can be inferred that the objectives have been met regarding output-relevance and independence. Regarding slowness, a good way of judging if the objective is achieved or not is by calculating the velocities of the latent features and comparing them with those of the input variables w.r.t the validation dataset. If few of the latent features are slower than all of the inputs, the nonlinear map has successfully projected the data to a slower latent surface, thus achieving the objective. Finally, a third set called the test dataset is used to test the performance of the model and this dataset is not 'seen' by the model during the training phase.

Deep learning enables the extraction of deeper features with deeper levels of abstraction. So the encoder network can be typically configured to have more hidden layers than the output network. Once the feature representation is obtained, the output layer will function as a mapping of the features to the outputs. Hence a shallow network can be used here.

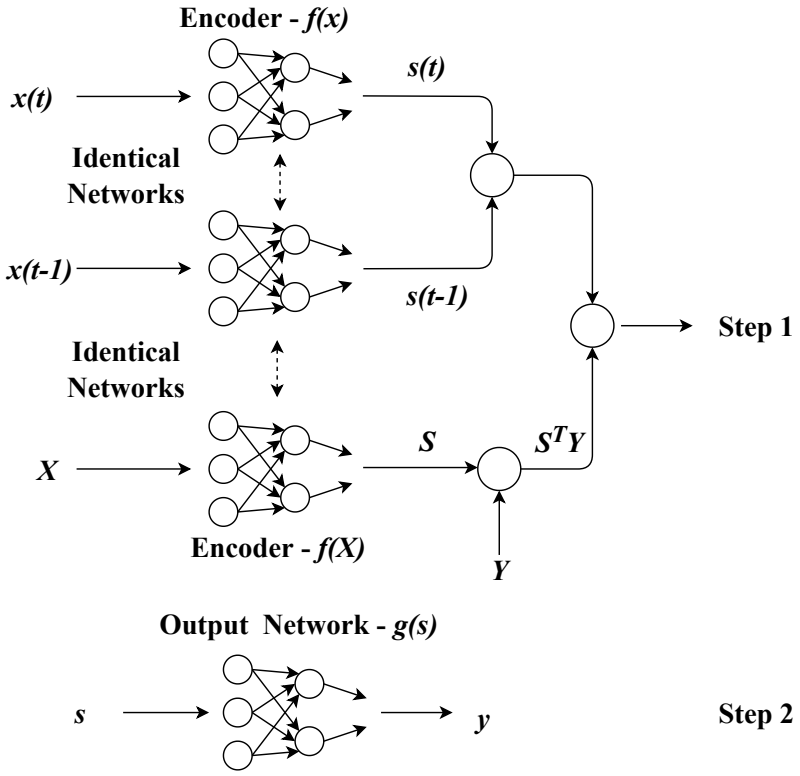


Figure 4.3: Configuration of the network for SSNSFA2. Step 1 architecture corresponds to the objective in (4.6). Step 2 is same as the Step 2 in SSNSFA1

4.3.2 SSFASN2

The configuration of the network for SSFASN2 is shown in Fig. 4.3. Similar to SSFASN1, SSFASN2 is also a two-step procedure. In the first step, the objective is to obtain SFs via the encoder network that are correlated with outputs. The objective function of SSFASN2 is as follows.

$$\min_f \underbrace{\sum \|f(x_t) - f(x_{t-1})\|_2)^2}_{\textbf{Slowness of features}} - \underbrace{\alpha \|cov(f(X), Y)\|}_{\textbf{Output covariance}} + \underbrace{\beta ||cov(f(X)) - I||}_{\textbf{Independence and unit variance}} \tag{4.6}$$

The above objective has three terms.

- The first term is similar to SSFASN1. It minimizes the distance between extracted features so that it results in the extracted features being slow.
- The second term tries to maximize the covariance between the extracted feature and the output. In PLS, feature extraction is performed such the features

extracted from input and output spaces have the maximum covariance. This term is similar to the objective function of PLS [102]. This term is weighted by a positive α as the negative sign is already included in the loss function.

- The third term is again similar to that in SSFASN1 which results in decorrelated features.

In the first step of the approach, only the encoder network is trained. Incorporation of the output covariance terms results in a high covariance between the extracted SFs and the outputs. This makes the features extracted relevant to the outputs. If we expand the gradient of (4.6) similar to (4.4), we can observe that the gradient w.r.t any parameter in the encoder section is affected by the outputs through the output covariance term making the latent variable learning supervised. Once the training of the encoder network is done, this can be used to extract the latent features and these can be used as inputs to the output network which is trained to minimize the output error with the objective function the same as the one in (4.5).

The training and testing are done similarly to the SSFASN1. The difference between SSFASN1 and SSFASN2 is that in the former, both encoder and output networks are trained together in the first step and the second step acts as a tuning step for the output network. For SSFASN2, the encoder and output networks are trained separately. The whole network can be tuned again using the parameters obtained in steps 1 and 2 as initial guesses. But it is possible that tuning the whole network again may lead to losing the slowness characteristics of latent features which may lead to noisier predictions of the output. This can be avoided to a certain extent using smaller steps and keeping the slowness penalty term during the training of the whole network.

The training times for SSFASN1 and SSFASN2 are higher than those for an unregularized neural network training. This is because the added regularizing terms in the objective functions mean more computational complexity to evaluate the objective function and its gradients. Besides this, the stagewise training nature of the proposed algorithms and the number of hyperparameters to be tuned, also mean an increased computation time. But, since the training is performed offline, the added computation time will not be a limiting factor.

4.4 Results

The proposed methods are implemented in three case studies: a simulated, an industrial, and an experimental dataset. The neural network simulations are performed using Python's TensorFlow library. Networks are trained by the stochastic gradient descent using the Adam algorithm implemented by TensorFlow's AdamOptimizer. In the simulations, as discussed earlier, the total data is split into training, validation, and test datasets. Training and validation sets are used to train the parameters and tune the hyperparameters respectively. The hyperparameters are the structure of the network, activation function, weights α and β in the proposed approaches. The proposed methods are aimed at developing static models by incorporating the temporal correlation (slowness) aspect. Hence, in the comparative analysis, we have selected two of the recently developed deep learning models HELM [128] and VW-SAE [130] which are static in nature. Along with these, we have also included results from FNN and slow feature regression(SFR). In SFR, the SFs are extracted using SFA, and the features are ordered according to their correlation with output. Regression is performed between the SFs and outputs and the number of SFs is decided based on the performance on the validation dataset. A comparison of these methods with the proposed ones is made in terms of root-mean-squared error (RMSE) and concordance correlation coefficient (ρ_c). RMSE is expressed as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}}. \tag{4.7}$$

Here, $\hat{y}$ and y are the estimated and the actual value of the output. ρ_c is a measure of agreement between a set of paired values. It is a measure of how the data spreads around the 45° line when $\hat{y}$ is plotted against y. ρ_c is expressed as

$$\rho_c = \frac{2\rho\sigma_{\hat{y}}\sigma_y}{\sigma_{\hat{y}}^2 + \sigma_y^2 + (\mu_{\hat{y}} - \mu_y)^2}. \tag{4.8}$$

Here, ρ is the correlation coefficient between y and $\hat{y}$. $\sigma_\bullet$ and $\mu_\bullet$ are the standard deviations and means of the corresponding variables. The value of ρ_c ranges from -1 to 1, with 1 implying perfect agreement between the two sets.

4.4.1 Simulated case study

In this case study, SFs are generated using the PSFA model. These are then nonlinearly mapped to get the input and the output signals. This is according to the rationale that the SFs are primarily responsible for the observed outputs for processes that are primarily slow in nature. The PSFA model is given as follows.

$$s_t = As_t + v_t; \quad v_t \sim \mathcal{N}(0, \Sigma_s) \tag{4.9}$$

$$x_t = Hs_t + w_t; \quad w_t \sim \mathcal{N}(0, \Sigma_x) \tag{4.10}$$

The matrix A is a diagonal matrix with diagonal entries a_i lying between 0 and 1. The closer a_i is to 1, the slower the feature is. The noise covariance matrix Σ_s is diagonal with the covariance being $1 - a_i^2$ to make the SFs have unit variance. Equation (4.10) describes a linear relationship between the SFs and the inputs. In this case study, the SFs are mapped nonlinearly to get the input data as well as the output data.

In this case study, three SFs are generated using (4.9). The matrix A is taken as diag(0.999,0.997,0.995). These SFs are mapped using nonlinear functions to get the input dataset x. The nonlinear functions used in this case study are of the form s^k, $s \cdot exp(k \cdot s)$, $exp(k \cdot s^k)$, $exp(\frac{k}{c+s})$, $log(k \cdot s)$, $tanh(k \cdot s)$. These functions in various combinations are used to generate eight input signals from the three SFs. To generate the outputs, a quadratic polynomial function of the three SFs is used. Using this formulation, 8000 samples are generated. If we normalize the generated signals and calculate the mean of the squared velocities (MSV), we get the following results.

$$\begin{aligned} \langle(\Delta x)^2\rangle = [&0.0165,\ 0.0534,\ 0.0185,\ 0.0676,\\ &0.0252,\ 0.0318,\ 0.0313,\ 0.0325] \end{aligned} \tag{4.11}$$

The same value for the latent variables and the output signals are [0.0057 0.0067 0.0075] and 0.0106 respectively. Hence, the output is slower than all of the input signals. Hence it would be pertinent to do an analysis in terms of the velocities or the slowness of the features that result in the output which is the aspect on which the proposed approaches are based.

The dataset is split in the ratio of 50:25:25 between training, validation, and test sets. The split is made such that the temporal sequence of points is maintained, i.e, the test set comes after the validation set, which comes after the training set in the

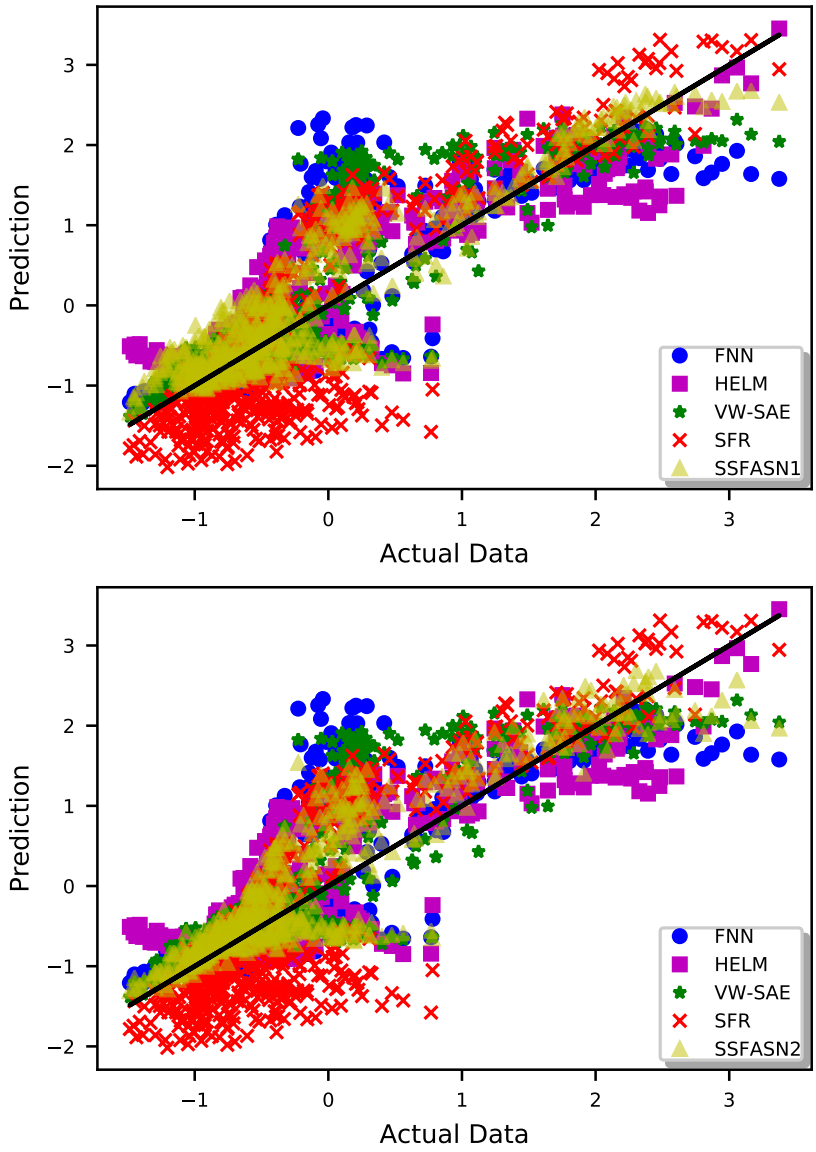


Figure 4.4: Scatter plot of predictions vs. actual data for the simulated dataset

time series. The network has the following structure [8-6-4-**3**-2-1] and the number of SFs is taken to be three. The activation function used is the leaky rectified linear unit (ReLU). ReLU activation function is expressed as $f(x)$ =max(x, ax) and the hyperparameter a is tuned using the validation dataset. For this case, a is 0.05 for

Table 4.1: Comparison of RMSE and ρ_c obtained for FNN, HELM, VW-SAE, SFR and proposed methods for the simulated dataset

Method	*RMSE*	ρ_c
FNN	0.5902	0.8199
HELM	0.5336	0.8427
VW-SAE	0.5171	0.8608
SFR	0.6848	0.8226
SSFASN1	0.4466	0.8953
SSFASN2	0.4377	0.9001

VW-SAE and the proposed methods, 0.1 for FNN, and 0.2 for HELM. The last layer is set to be a linear layer. The same structure of the overall network is chosen for FNN, VW-SAE and the two proposed methods.

The proposed approaches are implemented on the dataset and the performance is compared with that of FNN, VW-SAE, and SFR. Fig. 4.4 shows the plot of predicted output against the actual output for the test dataset. The top figure compares SSFASN1 with FNN, VW-SAE, and SFR and the bottom figure is for SSFASN2. The closer the points lie to the 45° line, the better is the approach. We can observe that the points are spread more tightly around the 45° line for the proposed methods than that for FNN, VW-SAE, and SFR. The RMSE and ρ_c values obtained on the test data for the four methods are presented in Table. 4.1. It can be seen that the proposed methods have a smaller RMSE and larger ρ_c than FNN and VW-SAE. The average reduction in RMSE for the proposed methods is 25.08% w.r.t FNN, 17.14% w.r.t HELM, 14.49% w.r.t VW-SAE, and 35.45% w.r.t SFR. SFR gives the largest RMSE as it is a linear method (but results in a better ρ_c than FNN as the data has slow latent features). But, the main reason is the supervised nature of latent feature extraction which can be observed if we consider the correlation of SFs with outputs. For SFR, the top three correlations are $[0.3844, 0.4101, 0.7023]$. While for the SSFASN1 and SSFASN2 are $[-0.5373, 0.3334, 0.8229]$ and $[0.2628, 0.7153, 0.6015]$ respectively. This higher correlation leads to better predictions. Among the neural networks, FNN gives the largest RMSE because in training an FNN, the objective is only to build an input-output model and no attention is paid towards the latent

representations of the data. A lack of such considerations may lead to a model with a poor generalization ability. HELM and VW-SAE extract the latent features from the inputs in stages, in unsupervised and supervised manners respectively and this results in a better representation of the latent structure of the data. This leads to a better generalization ability than FNN. But, in these methods (and also in FNN), the temporal nature of the data is not considered. Proposed SSFASN1 and SSFASN2 learn the latent structures considering the temporal nature of the data and latent features are learned in a supervised manner. This results in a better performance on datasets with the characteristics considered in this work.

Both the proposed approaches have three terms in the objective function and as a result, there are two weights α and β to be tuned. It is very likely that all the objectives may not be achieved while training the network. Hence, careful considerations must be given while tuning the weights such that all the objectives are achieved to a certain degree. In SSFASN1, the objective is to obtain SFs that will construct the output well (4.3). If α is low, the SFs may not be relevant to the outputs and if β is low, it is possible that all three features are highly correlated. If the weights are too high, the obtained features may not be slow at all. Because, with the slowness objective, it is easy to map inputs such that the features overall have a very low variance satisfying the slowness objective. But such signals are not necessarily slow and it must be ensured that they have unit variance. A similar analysis follows for SSFASN2 also. In this study, these facts are considered and weights are tuned such that all the objectives are achieved. For SSFASN1, the covariance matrix of the obtained features is

$$Cov(f(X)) = \begin{bmatrix} 0.8859 & 0.0107 & -0.1915 \\ 0.0107 & 0.8564 & 0.0017 \\ -0.1915 & 0.0017 & 0.7841 \end{bmatrix}. \tag{4.12}$$

Although not exactly an identity matrix, the variances are close to one and the matrix is diagonally dominant. When it comes to the slowness aspect, the obtained features have the following MSV: [0.0136, 0.0216, 0.0242]. Overall, these values are lower than the same for the input signals (4.11). This shows that the trained network has resulted in SFs. Hence, it can be concluded that all the objectives in the objective function are achieved. For SSFASN2, a diagonally dominant structure was obtained

in step-wise training. But when the whole network was trained (encoder and output networks trained together), such a structure was lost.

4.4.2 Debutanizer column

The second case study is performed on the debutanizer column dataset [115]. This is an open-source industrial dataset. Debutanizer column is used in a refinery to remove the lighter components of propane and butane from the naphtha. One of the objectives in the operation is to minimize the concentration of butane in the bottoms for which real-time measurement of the same is needed. This can be achieved using a soft sensor. The dataset contains 2394 samples with seven input variables. These variables are top temperature (x_1), top pressure (x_2), reflux flow (x_3), flow to the next process (x_4), sixth tray temperature (x_5) and bottom temperatures (x_6, x_7). The schematic diagram of the process and the description of the variables of the dataset are given in Fig. 3.3 and Table. 3.3 respectively [115]. Similar to the previous case, if we normalize the dataset and calculate the MSV of inputs, we get the following result.

$$\begin{aligned}\langle(\Delta x)^2\rangle =&[0.0400, 0.2802, 0.0438, 0.0293,\\ &0.0436, 0.0493, 0.0419]\end{aligned} \tag{4.13}$$

This value for the output is 0.0067 implying that the output is slower than all the inputs. The first 2200 samples in the dataset are used for the analysis. Similar to the previous case, the data is split into three portions, one each for training, validation, and testing in the ratio of 50:25:25. To enhance the performance of the soft sensors, delayed inputs are augmented. For every variable, a delayed sample is augmented based on its correlation with the output. Variables x_6 and x_7 are combined to get $\tilde{x_6} = \frac{x_6+x_7}{2}$. Variable x_5 overall has a high correlation with the outputs. Hence three delayed samples are taken. For every variable, up to 10 delays are considered. In total there are 14 inputs and one output.

$$\begin{aligned}\tilde{x}(t) =&[x_1(t),\ x_2(t),\ x_3(t),\ x_4(t),\ x_5(t),\ \tilde{x_6}(t),\\ &x_1(t-9),\ x_2(t-6),\ x_3(t-9),\ x_4(t-9),\\ &x_5(t-9),\ \tilde{x_6}(t-9),\ x_5(t-3),\ x_3(t-6)]\end{aligned} \tag{4.14}$$

After augmentation, $\tilde{x}(t)$ and $\tilde{x}(t-1)$ are still adjacent in time. Hence the argument that latent features of two adjacent samples must be close to each other still holds

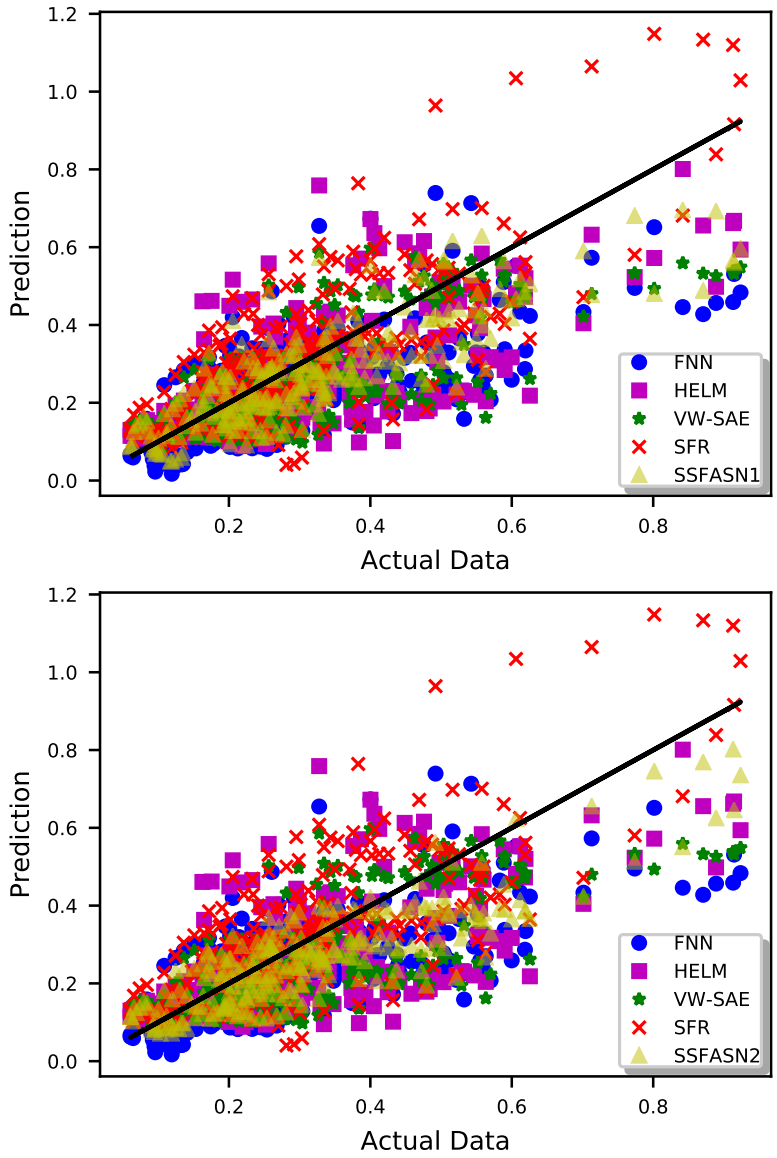


Figure 4.5: Scatter plot of predictions vs. actual data for the debutanizer column dataset

here. The structure of the neural network is [14-10-6-**3**-2-1] with the number of SFs taken as three. The ReLU parameter a is taken as 0.05 for FNN and 0.1 for VW-SAE. For the proposed methods, it is taken as {0.2, 0.2, 0.5, 0.2} after tuning. The latent

Table 4.2: Comparison of RMSE and ρ_c obtained for FNN, HELM, VW-SAE, SFR and proposed methods for debutanizer column dataset

Method	*RMSE*	ρ_c
FNN	0.1430	0.5669
HELM	0.1365	0.6234
VW-SAE	0.1262	0.6685
SFR	0.1260	0.7507
SSFASN1	0.1004	0.7783
SSFASN2	0.1018	0.7673

variable layer required a higher value of a to meet the objectives of decorrelation and slowness.

Fig. 4.5 shows the scatter plot of predictions vs. actual data for the test dataset. It can be observed that the proposed methods result in a tighter bound around the 45° line indicating better predictive ability. The results are quantified in Table. 4.2 which compares the RMSE and ρ_c values of the proposed methods against those of FNN, HELM, VW-SAE and SFR. The average reduction in RMSE in comparison with FNN, HELM, VW-SAE, and SFR is 29.30%, 25.93% 19.89%, and 19.76% respectively. It can be observed that SFR performs better than FNN indicating that the process is primarily driven by slow latent features and incorporating slowness into neural networks improves the results. The debutanizer column dataset has been used in [130], [132] and [137]. But it must be noted that in these cases y_{t-1} is used to predict y_t. We have developed models to predict outputs solely based on the inputs which are suitable for a scenario when y measurements are not available.

The weights of all the terms in the objective are carefully tuned so as to achieve each objective. For SSFASN1, the covariance matrix of the hidden features is

$$Cov(f(X)) = \begin{bmatrix} 0.8999 & 0.0194 & -0.0005 \\ 0.0194 & 0.8607 & 0.1198 \\ -0.0005 & 0.1198 & 0.9440 \end{bmatrix}. \tag{4.15}$$

The diagonal entries are close to one and the matrix is diagonally dominant. The MSV of the hidden features are 0.0205, 0.0222 and 0.0300 which are overall smaller

than the MSV of input signals (4.13). For SSFASN2, we get

$$Cov(f(X)) = \begin{bmatrix} 1.0406 & 0.1858 & 0.2222 \\ 0.1858 & 1.0202 & 0.1769 \\ 0.2222 & 0.1769 & 1.0344 \end{bmatrix}. \tag{4.16}$$

The variances are close to one and the matrix is diagonally dominant. The MSV of the features are 0.0361, 0.0347, and 0.0489 which indicate that the extracted features are overall slower than the inputs. These indicate that the weights are properly tuned in order to achieve all the objectives.

4.4.3 Hybrid tank system

This is an experimental case study and the data is obtained from a hybrid tank system. The experimental setup consists of three tanks connected as shown in Fig. 3.7, two pumps, and nine valves. Valves V1-V4 connect the middle tank and the side tanks, and valves V5, V7, and V9 connect the tanks and the storage tank. The flow of water from the tanks to the storage tank is due to gravity making the system nonlinear and the interactions between the tanks increase the nonlinear nature of the system. The objective is to develop a soft sensor to predict the height of water in the middle tank. Eight input variables are selected: flowrate, side tank liquid level, pump-speed, slave controller output for both sides of the system. Heights usually vary slower than the input variables. Hence it would be advantageous to impart the slowness constraints on the latent variables. This can be seen through MSV for inputs and outputs. The MSV for inputs is

$$\begin{aligned} \langle(\Delta x)^2\rangle = &[0.4061, 0.0006, 1.0438, 0.0019, \\ &0.4072, 0.0006, 1.0562, 0.0018] \end{aligned} \tag{4.17}$$

The MSV for output is 0.0006. We can see that the heights are slower than the remaining inputs. A total of 10,000 samples are collected and are split as 50:25:25 into training, validation, and test sets. The network structures are taken as [8-6-5-**4**-2-1] with the latent structure taken as four. The ReLU parameter for the hidden layers is taken as {0.2, 0.2, 0.5, 0.2} for the proposed methods and 0.2 and 0.05 for all layers for FNN and VW-SAE respectively. The RMSE and ρ_c values are given in Table. 4.3 and the scatter plot of the predictions against actual data for test data is shown in Fig. 4.6. The proposed methods gave a tighter spread around the 45° line

Table 4.3: Comparison of RMSE and ρ_c obtained for FNN, HELM, VW-SAE, SFR and proposed methods for the hybrid tank system dataset

Method	*RMSE*	ρ_c
FNN	0.6643	0.7650
HELM	0.8042	0.5861
VW-SAE	0.6165	0.8101
SFR	1.4731	0.5158
SSFASN1	0.5220	0.8552
SSFASN2	0.4823	0.8766

than FNN, HELM, VW-SAE, and SFR. Conventional SFR gave poor results when compared with nonlinear methods while incorporating slowness into neural networks by the proposed methods gave the best results both in terms of RMSE and ρ_c. Similar to the previous two case studies near-identity covariance matrices are obtained for the latent space for both methods.

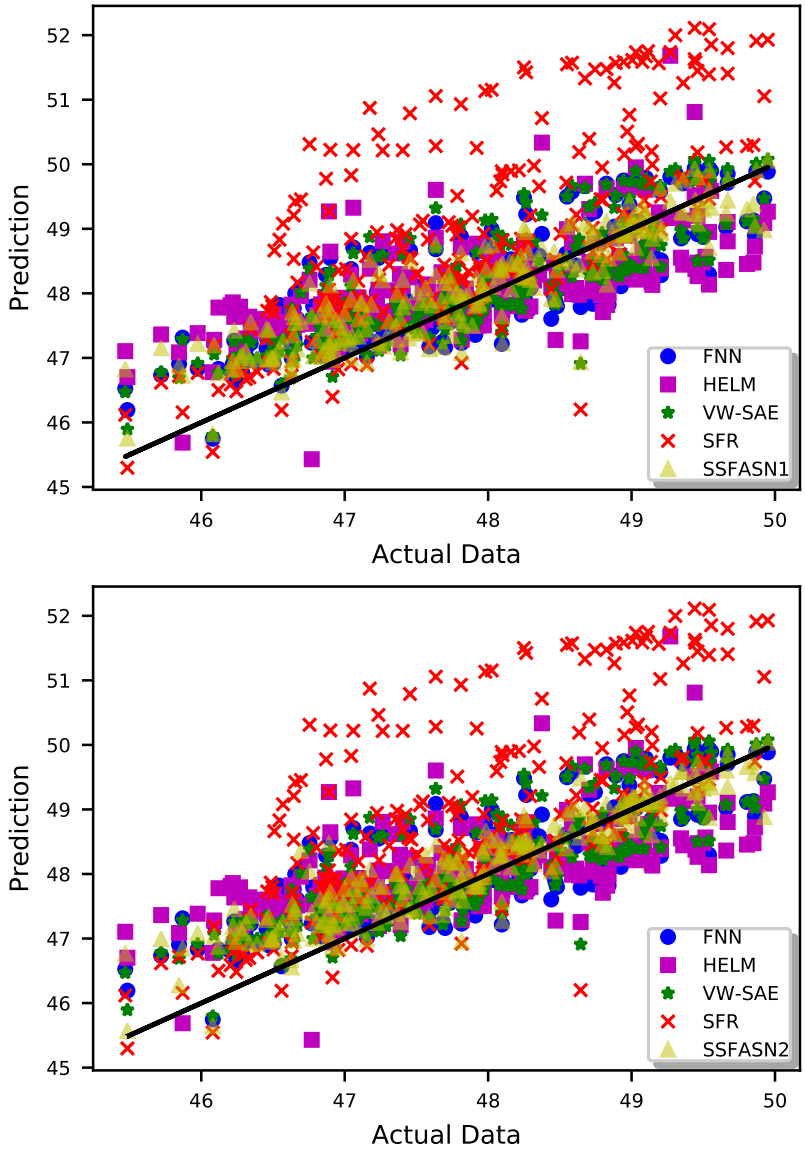


Figure 4.6: Scatter plot of predictions vs. actual data for the hybrid tank system dataset

4.5 Conclusion

In this chapter, two approaches are presented that perform output-relevant slow feature extraction. The proposed methods try to learn the latent space of time-series data considering two aspects: temporal relations and relevance of latent features to the outputs. Both these aspects are important to consider in the supervised learning of time series datasets. The proposed methods are implemented using Siamese neural networks. In neural networks, it is important that training is performed in a structured way to get better results. The proposed approaches provide such a structure from a perspective of temporal relation and output relevance. These methods can also be seen as a way of regularizing the deep learning process to give a certain structure to the latent space. The simulation results have shown that it is indeed beneficial to consider the temporal relations and the output relevance aspect for time-series data. The performance of the proposed SSFASN1 and SSFASN2 is better than FNN, HELM, and VW-SAE where such aspects are not considered.

Chapter 5

Latent Variable Modeling and State Estimation of Non-stationary Processes Driven by Monotonic Trends *

This chapter presents the latent variable modeling approach considering the monotonicity aspect of the latent variables. In certain non-stationary processes, the non-stationary dynamics is caused by degradation or wearing of certain process components. Such dynamics can be characterized by a latent monotonic signal. Meanwhile, there also exist stationary dynamics characterizing the regular process variables. It hence becomes pertinent to distinguish these two sets of latent variables for the monitoring of the process from both the stationary and non-stationary aspects. In this regard, we propose a methodology to achieve such a goal by modeling the latent monotonic trend as a closed skew-normal random walk model. The other stationary relations are characterized by a state-space model with Gaussian noises. The problem is solved as a simultaneous state and parameter estimation problem using the expectation-maximization algorithm. As a result of the closed skew-normal random walk model for the monotonic trend, the state estimation problem becomes a skew-normal filtering and smoothing problem. The effectiveness of the proposed method is verified through a numerical simulation, and the algorithm is applied to solve a Hot Lime Softener fouling predictive monitoring problem.

*This chapter has been published as **R. Chiplunkar** and B. Huang, "Latent variable modeling and state estimation of non-stationary processes driven by monotonic trends", *Journal of Process Control*, vol. 108, pp. 40-54, 2021.

5.1 Introduction

Processes accompanied by deteriorating "health factors" are often driven by a latent non-stationary variable which is monotonic in nature. Catalyst activity, fouling deposition, equipment damage, etc would be a few typical examples for such processes where the process progresses in a monotonic way. Yet there will be many relationships between the variables that would remain the same through time which could be characterized by a stationary nature. The observed multivariate dataset of such processes would still have an overall non-stationary character as it is a mixture of both stationary and non-stationary data trends. For effective monitoring of such processes, one needs to isolate the latent monotonic trend (LMT) from the latent stationery trends (LST).

Although such problems may be formulated as constrained optimization problems, modeling such systems with a DLM and approaching it with a state and parameter estimation perspective is more convenient for dynamic systems. As discussed in section 1.2.2, the process noise for a monotonically evolving signal should be a distribution with a positive support (for a monotonically increasing signal). In the literature, various distributions such as the exponential distribution, gamma distribution, etc have been used to model the monotonic signal. With these distributions, the monotonic signal's distribution cannot be exactly inferred because of non-conjugacy issue in the state estimation procedure. This thesis hence explores the usage of the closed skew-normal distribution (CSN) to model the evolution of the monotonic signal. the CSN distribution hase the properties of closedness, meaning the resulting distribution is always a CSN upon a convolution or Bayesian inversion operation as long as the distributions involved in these operations are CSN. This results in a recursive derivation of the state estimation procedure for a DLM modeled using CSN.

In deteriorating processes, the monotonic nature will not be apparent as it may be mixed with other stationary sources. As discussed earlier, the objective in this case is to model and separate the LMT and LST sources from the observed dataset for efficient monitoring of the processes. Monotonicity is a special case of non-stationary behavior. Separation of the stationary and non-stationary trends has been researched previously. Co-integration analysis (CA) [146–148] is an approach where the non-

stationary system, modeled typically as a vector autoregressive model, is linearly projected onto a stationary subspace. Stationary subspace analysis (SSA) [149–151] is another approach towards this where the observed data is assumed to be a linear mixture of stationary and non-stationary sources and an optimization problem is solved to learn a matrix that maps the data to its stationary subspace. The main limitation of these methods is that they do not consider the monotonicity aspect. Since the faults considered in this work are assumed to evolve monotonically, CA and SSA cannot be used for such cases. Secondly, these approaches rely on linear projections of the available data onto the stationary and non-stationary subspaces. So, if the system has a mixture of n_m non-stationary components and n_s stationary components, and the number of observed signals n_y is such that $n_y < n_m + n_s$, these approaches cannot be used. Another approach to separate the stationary and non-stationary sources is based on state-space modeling. Scott et al. [59] approached the problem by having two independent state-space models, one for stationary and another random-walk model for non-stationary characteristics, and identifying the state-space models. In this work, the current problem is approached via the state-space model approach as it provides two main advantages. The first one is that it is convenient to incorporate the monotonicity constraint through the state-space model approach as the LMT is modeled using a CSN. The second one is that the number of observed signals can be less than the mixed stationary and non-stationary signals. The combined state and parameter estimation problem is then solved using the expectation-maximization (EM) algorithm [106, 107]. The state estimation portion of the problem then reduces to a Kalman filtering and Rauch–Tung–Striebel (RTS) smoothing problem with CSN distribution. In this work, a complete analytical derivation of the recursive equations of the filtering and smoothing steps is provided. Finally, the efficacy of the proposed methods is verified on two case studies. The first one is a simulated case study. The second one is on an industrial fouling dataset where the proposed method is used to monitor the fouling buildup in a hot lime softener process. The contributions of this chapter are as follows.

1. Separation of stationary and non-stationary sources of a monotonically deteriorating process by formulating the system as a CSN state-space model

2. Develop recursive smoothing equations analytically for a CSN model when the process noise is CSN and observation noise is Gaussian.

The rest of the chapter is organized as follows. In section 5.2, the proposed model is presented. In section 5.3, the solution for the proposed model by the EM algorithm is given. Section 5.4 presents the results of the case studies and section 5.5 concludes the chapter. The derivations involved in this Chapter for the E-step are setailed in Appendix A.

5.2 Latent variable model with a hidden monotonic trend

The section presents the proposed model to model the LMT and LST. Before that, the CSN distribution and its notion is visited briefly.

5.2.1 Closed skew-normal distribution - Revisit

The CSN, as proposed in [78] and [79], can be described using two variables, one of which has truncated observations. Let us consider the following equations.

$$l = \alpha k + e_l, \quad e_l \sim \mathcal{N}(e_l;\ 0, \sigma_l^2) \tag{5.1}$$

$$m = \beta k + e_m, \quad e_m \sim \mathcal{N}(e_m;\ 0, \sigma_m^2) \tag{5.2}$$

Let the covariance between the two noises be σ_{lm}. In the remainder of this chapter, the following notations are used to denote the Gaussian probability distribution function (pdf) and cumulative distribution function (cdf) respectively: $\mathcal{N}_{dimension}$(variable; mean, variance) for the pdf and $\Phi_{dimension}$ (limit; mean, variance) for the cdf. The joint distribution of y and z now can be expressed as

$$\begin{bmatrix} l \\ m \end{bmatrix} \sim \mathcal{N}_{1+1}\left(\begin{bmatrix} l \\ m \end{bmatrix}; \begin{bmatrix} \alpha k \\ \beta k \end{bmatrix}, \begin{bmatrix} \sigma_l^2 & \sigma_{lm} \\ \sigma_{ml} & \sigma_m^2 \end{bmatrix} \right). \tag{5.3}$$

Let us look at the distribution $p(l|k, m \geq 0)$.

$$\begin{aligned} p(l|k, m \geq 0) =& \frac{p(m \geq 0|l, k)\ p(l|k)}{p(m \geq 0|k)} \\ =& \frac{(1 - \Phi_1(0;\ \mu_{m|l,k}, \sigma_{m|l,k}))\ \mathcal{N}_1(l;\ \alpha k, \sigma_l^2)}{(1 - \Phi_1(0;\ \beta k, \sigma_m^2))} \end{aligned} \tag{5.4}$$

Here,

$$\mu_{m|l,k} = \beta k + \frac{\sigma_{ml}}{\sigma_l^2}(l - \alpha k);\ \sigma_{m|l,k} = \sigma_m^2 - \frac{\sigma_{ml}^2}{\sigma_l^2} \tag{5.5}$$

Equation (5.4) can be simplified as

$$p(l|k, m \geq 0) = \frac{\mathcal{N}_1(l;\ \alpha k, \sigma_l^2)\ \Phi_1(\frac{\sigma_{ml}}{\sigma_l^2}(l - \alpha k);\ -\beta k, \sigma_{m|l,k})}{\Phi_1(0;\ -\beta k, \sigma_m^2)} \tag{5.6}$$

From the above equation, one can see that the distribution has two terms which are a function of l. The Gaussian pdf and the Gaussian cdf. The cdf term is the one that causes skewness in the distribution. The denominator is a normalization constant. Equation (5.6) can be generalized to define the pdf of CSN. The pdf of x distributed as a CSN is given as

$$p(x) = \frac{\mathcal{N}_n(x;\ \mu, \Sigma)\ \Phi_q(\Gamma(x - \mu);\ \nu, \Delta)}{\Phi_q(0;\ \nu, \Delta + \Gamma\Sigma\Gamma')} \tag{5.7}$$

which can be represented in a compact form as

$$x \sim CSN_{n,q}(x;\ \mu, \Sigma, \Gamma, \nu, \Delta) \tag{5.8}$$

Here n is the dimension of x and hence the dimension of the Gaussian pdf term. q is the dimension of the cdf term. The CSN is parameterized by five parameters. $\mu(n \times 1)$ and $\Sigma(n \times n)$ are the mean and covariance of the pdf term, while $\nu(q \times 1)$ and $\Delta(q \times q)$ are the mean and covariances of the cdf term. $\Gamma(q \times n)$ is called the skewness parameter and when $\Gamma = 0$, CSN reduces to a Gaussian distribution.

5.2.2 Model formulation

The objective is to extract LMT($\{m_t\}$) and LST($\{s_t\}$), given the observations ($\{y_t\}$). Here $\{\ .\ \}$ represents the set of all variables from time $t = 1$ to $t = T$. The proposed model is defined in the following equations.

$$s_t = A\ s_{t-1} + v_t, \qquad v_t \sim \mathcal{N}_{n_s}(v_t;\ \mathbf{0}, \Sigma_v) \tag{5.9}$$

$$m_t = m_{t-1} + e_t \tag{5.10}$$

$$r_t = w_t \tag{5.11}$$

$$y_t = H_1\ m_t + H_2\ s_t + u_t, \qquad u_t \sim \mathcal{N}_{n_y}(u_t;\ \mathbf{0}, \Sigma_u) \tag{5.12}$$

It is assumed that the overall non-stationarity is caused by a single source of degradation which is taken to be the LMT in this case. The stationary part of the system has no such assumption and hence is represented by a higher-order system as given in (5.9) of the proposed formulation. Here, the evolution of the stationary component of the process s is represented by A which is an $n_s \times n_s$ matrix. The matrix A must have all the eigenvalues within the unit circle for stability, but mainly to ensure that (5.9) does not model the non-stationary dynamics. v_t is the process noise for s_t with a mean of a $n_s \times 1$ vector of zeros represented by $\mathbf{0}$ and a covariance matrix of Σ_v which is assumed to be diagonal. Here onwards whenever the LST modeled by (5.9) is referred to as being stationary, we imply weak stationary dynamics, i.e the case where the mean is constant throughout time, but the covariance need not be. One could assume special structures in Σ_v to make the (5.9) strictly stationary. However, a weak-stationary model is assumed in this work for a more general case. The observed data is defined as a linear mixture of the LMT and LST and hence is represented as in (5.12). Here, u_t is the observation noise with a zero mean and a covariance matrix Σ_u which is taken as a diagonal matrix. Equations (5.10) (a random walk model) and (5.11) together represent the LMT. Here, m_t represents the LMT and r_t is a latent variable that is introduced just to make m_t monotonic (similar to m in (5.2)). Noises e_t and w_t are correlated and the Pearson correlation coefficient is $\rho > 0$. All other noises are uncorrelated with each other. The dimensionality of m_t (and also r_t) is taken to be one because of the assumption that the overall degradation in the process can be represented by a single LMT. With these definitions, the joint distribution of m_t and r_t can be expressed as

$$\begin{bmatrix} m_t \\ r_t \end{bmatrix} \sim \mathcal{N}_2\left(\begin{bmatrix} m_{t-1} \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_e^2 & \rho\sigma_e^2 \\ \rho\sigma_e^2 & \sigma_e^2 \end{bmatrix} \right). \tag{5.13}$$

The variance of both e_t and w_t is taken to be σ_e^2. Here onwards, the model and the procedure presented is to extract a monotonically increasing trend. The extension for a decreasing one is straightforward. To ensure that x is monotonically increasing, one must define the transition probability for x such that $m_t \geq m_{t-1}$. Hence one would want to find the pdf $p(m_t|m_{t-1}, r_t \geq 0)$. The reason to have the condition that r_t is greater than zero is as follows: the signal r_t is nothing but a white noise w_t and if w_t is correlated with e_t, and ρ is set close to 1, $r_t \geq 0$ or $w_t \geq 0$ would result in $e_t \geq 0$

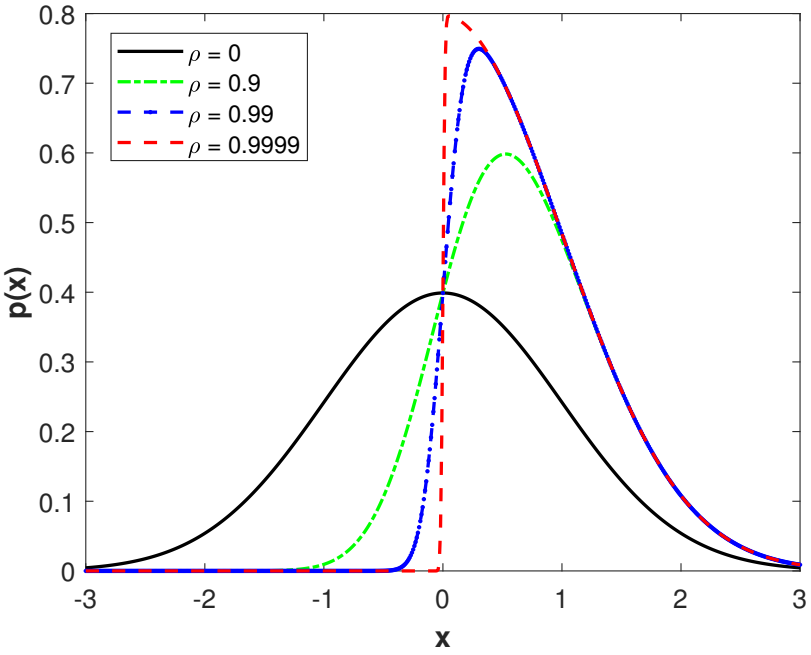


Figure 5.1: The pdf of CSN at different values of ρ.

with a greater probability. $e_t \geq 0$ ensures that $m_t \geq m_{t-1}$. The pdf $p(m_t|m_{t-1} \geq 0)$ can be written as

$$p(m_t|m_{t-1}, r_t \geq 0) = \frac{p(r_t \geq 0|m_{t-1}, m_t)\ p(m_t|m_{t-1})}{p(r_t \geq 0|m_{t-1})}. \tag{5.14}$$

It can be observed that this has a form similar to the pdf $p(p(l|k, m \geq 0))$ defined in section 5.2.1. One can now substitute each term in the above equation as follows.

$$p(m_t|m_{t-1}) \quad = \mathcal{N}_1(m_t;\ m_{t-1}, \sigma_e^2), \tag{5.15}$$

$$p(r_t \geq 0|m_{t-1}) = p(r_t \geq 0) = 0.5 = \Phi_1(0;\ 0, \sigma_e^2). \tag{5.16}$$

The skewness term in the numerator is

$$\begin{aligned} p(r_t \geq 0|m_{t-1}, m_t) =& 1 - \Phi_1(0;\ \rho(m_t - m_{t-1}), \sigma_e^2(1-\rho^2)) \\ =& \Phi_1(\rho(m_t - m_{t-1});\ 0, \sigma_e^2(1-\rho^2)). \end{aligned} \tag{5.17}$$

Substituting these terms in (5.14), the following expression is obtained.

$$p(m_t|m_{t-1}, r_t \geq 0) = CSN_{1,1}(m_t;\ m_{t-1}, \sigma_e^2, \rho, 0, \sigma_e^2(1-\rho^2)). \tag{5.18}$$

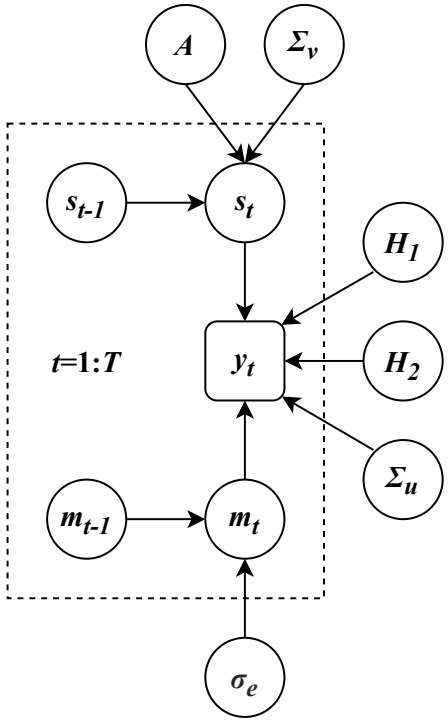


Figure 5.2: The hierarchical probabilistic graphical model of the proposed approach to extract the monotonic signal

Ideally, one would want a truncated Gaussian distribution to ensure that m_t is always greater than m_{t-1}. We still define it in terms of a CSN as even a half-truncated Gaussian distribution would make the states follow CSN in the E-step (discussed later) due to the fact that the convolution of a half-truncated Gaussian with a Gaussian leads to CSN, and also a CSN process has a nice recursive structure during filtering. CSN moreover generalizes a half-truncated Gaussian distribution thus also providing an extra parameter ρ for tuning. This gives one more flexibility as not all datasets need a very high ρ to ensure the extracted signal to be monotonic in nature as shown in the case studies. Fig. 5.1 shows the pdf of $CSN(x;\ 0, 1, \rho, 0, (1-\rho^2))$ at different values of ρ. One can observe that as ρ approaches 1, the skewness of the distribution increases and the pdf resembles a truncated Gaussian distribution when ρ is close to 1. At $\rho = 1$, the pdf in fact becomes a truncated Gaussian distribution. Although this parameter adds some flexibility in tuning the LMT, particularly the rate of the LMT's evolution, it must be tuned carefully so as to ensure a monotonic trend.

Finally, the proposed model can be summarized in terms of the distributions as

follows.

$$p(s_t|s_{t-1}) = \mathcal{N}_{n_s}(s_t;\ As_{t-1}, \Sigma_v) \tag{5.19}$$

$$p(m_t|m_{t-1}, r_t \geq 0) = CSN_{1,1}(m_t;\ m_{t-1}, \sigma_e^2, \rho, 0, \sigma_e^2(1-\rho^2)) \tag{5.20}$$

$$p(y_t|m_t, s_t) = \mathcal{N}_{n_y}(y_t;\ H_1 m_t + H_2 s_t, \Sigma_u) \tag{5.21}$$

The states to be estimated are $\{[m_t'\ s_t']'\}$ and the parameters are $\{A, H_1, H_2, \Sigma_v, \sigma_e^2, \Sigma_u\}$. This state and parameter estimation problem can be solved using the EM algorithm. Note that in the remainder of this chapter, mentioning of the condition $r_t \geq 0$ is left out for the convenience of presentation. The hierarchical probabilistic graphical model for the model represented in (5.19), (5.20), and (5.21) is shown in Fig. 5.2

5.3 Maximum likelihood estimation

Given the distributions defined in (5.19), (5.20) and (5.21), the joint likelihood of the whole dataset conditioned on $\{r_t > 0\}$ can be formulated as follows.

$$\begin{aligned}
&p(\{m_t\},\{y_t\},\{s_t\}|\{r_t \geq 0\}) \\
&\quad = \prod_{t=1}^{T} p(y_t|m_t, s_t, r_t \geq 0) \times \prod_{t=2}^{T} p(m_t|m_{t-1}, r_t \geq 0) p(s_t|s_{t-1}, r_t \geq 0) \\
&\qquad \times p(m_1|r_1 \geq 0) p(s_1|r_1 \geq 0) \\
&\quad = \prod_{t=1}^{T} p(y_t|m_t, s_t) \times \prod_{t=2}^{T} p(m_t|m_{t-1}, r_t \geq 0) p(s_t|s_{t-1}) \times p(m_1|r_1 \geq 0) p(s_1)
\end{aligned} \tag{5.22}$$

Maximizing this likelihood directly is infeasible and hence the EM algorithm is used.

5.3.1 EM algorithm - Revisit

The EM algorithm is a technique for maximizing the likelihood function of a system when the data for some of the variables are missing. It is one of the standard techniques used for latent variable modeling. In these cases, maximizing the likelihood directly is infeasible. EM algorithm which is an iterative procedure can be followed to iteratively estimate the latent variables and the parameters. As the name suggests, the algorithm has two steps: the expectation step (E-step) and the maximization step (M-step). The details of the algorithm can be found in Bishop [106] and we briefly

mention the two steps. Suppose we have observed data Y and latent data S, the joint likelihood of $p(Y, S|\theta)$ can be maximized in the following steps.

1. Initialize the parameters θ as θ_{old}

2. *E-step*: Calculate the expected value of the joint log likelihood as a function of θ, where the expectation is taken w.r.t the conditional distribution of $p(S|Y, \theta_{\text{old}})$. This is called the Q function.

$$Q(\theta|\theta_{\text{old}}) = \mathbb{E}_{S\sim p(S|Y,\theta_{\text{old}})}[log\ p(Y, S|\theta)] \tag{5.23}$$

3. *M-step*: Maximize the Q function w.r.t θ to update the parameters.

$$\begin{aligned} \theta_{\text{new}} &= \arg\max_{\theta} Q(\theta|\theta_{\text{old}}) \\ \theta_{\text{old}} &= \theta_{\text{new}} \end{aligned} \tag{5.24}$$

4. Iterate till convergence.

5.3.2 M-Step

We first present the M-step of the algorithm. Substituting the distributions defined in (5.19), (5.20) and (5.21) into the likelihood in (5.22) and taking the logarithm, the log-likelihood function can be obtained as follows. The constant terms can be ignored as they will not affect the following derivations and the log-likelihood function is expressed in terms of the states and parameters.

$$\begin{aligned} &log\ p(\{m_t\}, \{y_t\}, \{s_t\}|\{r_t \geq 0\}) = \\ &-\sum_{t=1}^{T} \frac{1}{2}(y_t - H_1 m_t - H_2 s_t)' \Sigma_u^{-1} (y_t - H_1 m_t - H_2 s_t) \\ &-\frac{T}{2} log(det(\Sigma_u)) \\ &-\sum_{t=2}^{T} \frac{1}{2}(s_t - A s_{t-1})' \Sigma_v^{-1} (s_t - A s_{t-1}) - \frac{T-1}{2} log(det(\Sigma_v)) \\ &-\sum_{t=2}^{T} \frac{1}{2} \frac{(m_t - m_{t-1})^2}{\sigma_e^2} - \frac{T-1}{2} log(\sigma_e^2) \\ &+\sum_{t=2}^{T} log(\Phi_1(\rho(m_t - m_{t-1});\ 0, \sigma_e^2(1-\rho^2))) \end{aligned} \tag{5.25}$$

The Q function can be calculated by taking the expectation of the above expression w.r.t the distribution of the latent variables given all the observations. To obtain the expression for certain parameters, the Q function is differentiated w.r.t the parameter and equated to zero. The whole procedure is straightforward and can be accomplished using some well-known matrix differentiation lemmas. The following equations for the mapping matrices of the model are obtained [106].

$$A = \left\{ \sum_{t=2}^{T} \mathbb{E}[s_t s_{t-1}'] \right\} \left\{ \sum_{t=2}^{T} \mathbb{E}[s_{t-1} s_{t-1}'] \right\}^{-1} \tag{5.26}$$

$$H = \left\{ \sum_{t=1}^{T} y_t \mathbb{E}[[m_t' \ s_t']] \right\} \left\{ \sum_{t=1}^{T} \mathbb{E}\left[\begin{bmatrix} m_t \\ s_t \end{bmatrix} [m_t' \ s_t'] \right] \right\}^{-1} \tag{5.27}$$

Here $H = [H_1 \ H_2]$ and the two matrices are estimated together as H by augmenting the LMT and LST. The two covariance matrices can be estimated as

$$\Sigma_u = \frac{1}{T} \sum_{t=1}^{T} \left\{ y_t y_t' - y_t \ \mathbb{E}[[m_t' \ s_t']] H' - H\mathbb{E}\left[\begin{bmatrix} m_t \\ s_t \end{bmatrix} \right] y_t' + H\mathbb{E}\left[\begin{bmatrix} m_t \\ s_t \end{bmatrix} [m_t' \ s_t'] \right] H' \right\} \tag{5.28}$$

$$\Sigma_v = \frac{1}{T-1} \sum_{t=2}^{T} \left\{ \mathbb{E}[s_t s_t'] - \mathbb{E}[s_t s_{t-1}'] A' - A\mathbb{E}[s_{t-1} s_t'] + \quad A\mathbb{E}[s_{t-1} s_{t-1}'] A' \right\} \tag{5.29}$$

Since we have assumed a diagonal structure for the covariance matrices, only the diagonal elements of the above matrices are needed to form the updated covariance matrix.

$$\Sigma_u = diag(diag(\Sigma_u)); \qquad \Sigma_v = diag(diag(\Sigma_v)) \tag{5.30}$$

To obtain the skewness parameter ρ of the LMT, the following term has to be differentiated w.r.t ρ.

$$\mathbb{E}\left[\sum_{t=2}^{T} log(\Phi_1(\rho(m_t - m_{t-1}); \ 0, \sigma_e^2(1-\rho^2)) \right] \tag{5.31}$$

With the states being inside a nonlinear function of $log(\Phi())$, it is difficult to obtain closed-form expressions for ρ. Referring back to Fig. 5.1 and the discussion in section 5.2.2, the CSN considered here needs to resemble a half-truncated Gaussian. For that, ρ needs to be set close to one. Hence ρ is taken as a tuning parameter and

manually tuned in a range close to 1 such that monotonic realization of the signal m_t is guaranteed. With CSN resembling a half-normal distribution, the $\Phi(.)$ term will disappear from the distribution and the expected log-likelihood of the LMT will be as follows.

$$\mathbb{E}\left[-\sum_{t=2}^{T} \frac{1}{2} \frac{(m_t - m_{t-1})^2}{\sigma_e^2} - \frac{T-1}{2} log(\sigma_e^2) \right] \tag{5.32}$$

Differentiating the above term w.r.t σ_e, gives us

$$\sigma_e^2 = \frac{1}{T-1} \sum_{t=2}^{T} \left\{ \mathbb{E}[m_t^2] - 2\mathbb{E}[m_t m_{t-1}] + \mathbb{E}[m_{t-1}^2] \right\} \tag{5.33}$$

It can be observed that all the equations for the parameter updates contain the expected values of the states. These will be estimated in the E-step of the EM algorithm.

5.3.3 E-step

In the E-step the expected values required to update the parameters are estimated. As discussed in section 5.3.1, the distribution w.r.t which the expectation of the quantities are evaluated is $p(m_t, s_t|y_t, \theta_{\text{old}})$. This is now a state estimation problem where this probability distribution is estimated at each time instant. We briefly visit the state estimation procedure in the next section. Here onwards, we will denote the set of measurements from time $t = 1$ to any time t as $Y_t = \{m_1, m_2, \ldots, m_t\}$. Hence Y_T represents the set of all measurements available and the distributions of our interest to be estimated are $p(m_t, s_t|Y_T)$.

5.3.3.1 State estimation - Revisit

The state estimation procedure of a Markov process defined by a state transition pdf $p(s_t|s_{t-1})$ and a observation pdf $p(y_t|s_t)$ has two steps. In the first step called filtering, one moves forwards from $t = 1$ to $t = T$ to estimate the pdf $p(s_t|Y_t)$ at each time instant. The filtering step can be further divided into prediction and update steps. In the second step called smoothing one moves backwards from $t = T$ to $t = 1$ estimating the pdf $p(s_t|Y_T)$ [152]. The state s defined in this revisit denotes a general hidden state. The whole procedure is summarized in the following equations.

Prediction:

$$p(s_t|Y_{t-1}) = \int p(s_t|s_{t-1})\ p(s_{t-1}|Y_{t-1})\ \ ds_{t-1} \tag{5.34}$$

Update:

$$p(s_t|Y_t) = \frac{p(y_t|s_t)\ p(s_t|Y_{t-1})}{\int p(y_t|s_t)\ p(s_t|Y_{t-1})\ \ ds_t} \tag{5.35}$$

Smoothing:

$$p(s_t|Y_T) = \int \frac{p(s_{t+1}|s_t)p(s_t|Y_t)}{\int p(s_{t+1}|s_t)p(s_t|Y_t)\ ds_t} p(s_{t+1}|Y_T)\ ds_{t+1} \tag{5.36}$$

5.3.3.2 Forward pass - Filtering

In this section we discuss the filtering part of the algorithm which involves solving (5.34) and (5.35).

Prediction: The prediction step involves calculating the pdf $p(m_t, s_t|Y_{t-1})$ as in (5.34). Say at time t, the following distribution is available from $t-1$ for the states, both the LMT and LST.

$$\begin{aligned} &p(m_{t-1}, s_{t-1}|Y_{t-1}) = \\ &CSN_{n_s+1,t-1}\left(\begin{bmatrix} m_{t-1} \\ s_{t-1} \end{bmatrix};\ \begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix}, \Sigma_{t-1|t-1}, \Gamma_{t-1|t-1}, \nu_{t-1|t-1}, \Delta_{t-1|t-1}\right) \end{aligned} \tag{5.37}$$

The subscripts of the parameters indicate the values of the corresponding parameters at $t-1$ given Y_{t-1}. The transition probability of the states is

$$\begin{aligned} p(m_t, s_t|m_{t-1}, s_{t-1}) &= CSN_{1,1}(m_t;\ m_{t-1}, \sigma_e^2, \rho, 0, \sigma_e^2(1-\rho^2)) \times \mathcal{N}_{n_s}(s_t;\ As_{t-1}, \Sigma_v) \\ &= \mathcal{N}_{n_s+1}\left(\begin{bmatrix} m_t \\ s_t \end{bmatrix}; \begin{bmatrix} m_{t-1} \\ As_{t-1} \end{bmatrix}, \begin{bmatrix} \sigma_e^2 & \mathbf{0} \\ \mathbf{0} & \Sigma_v \end{bmatrix}\right) \\ &\quad \times \frac{\Phi_1(\rho(m_t - m_{t-1});\ 0, \sigma_e^2(1-\rho^2))}{\Phi_1(0;\ 0, \sigma_e^2)} \end{aligned} \tag{5.38}$$

Here onwards, we use $\mathbf{0}$ to represent the vector or matrix of zeros of appropriate size. The cdf term in the numerator of the above equation can be rewritten as follows

$$\Phi_1(\rho(m_t - m_{t-1});\ 0, \sigma_e^2(1-\rho^2)) = \Phi_1\left(\gamma \begin{bmatrix} m_t - m_{t-1} \\ s_t - s_{t-1} \end{bmatrix};\ 0, \delta\right) \tag{5.39}$$

with $\gamma = [1\ \mathbf{0}]$ and $\delta = \sigma_e^2(1-\rho^2)$. Finally, the transition pdf can be expressed as

$$p(m_t, s_t|m_{t-1}, s_{t-1}) = CSN_{n_s+1,1}\left(\begin{bmatrix} m_t \\ s_t \end{bmatrix};\ \begin{bmatrix} m_{t-1} \\ As_{t-1} \end{bmatrix}, \begin{bmatrix} \sigma_e^2 & \mathbf{0} \\ \mathbf{0} & \Sigma_v \end{bmatrix}, \gamma, 0, \delta\right) \tag{5.40}$$

We also define the two following matrices.

$$\tilde{A} = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & A \end{bmatrix}; \quad \Sigma_S = \begin{bmatrix} \sigma_e^2 & \mathbf{0} \\ \mathbf{0} & \Sigma_v \end{bmatrix} \tag{5.41}$$

We can now substitute (5.40) and (5.37) into (5.34) to calculate $p(m_t, s_t|Y_{t-1})$. The resulting pdf is also a CSN and the expressions can be found in Rezaie and Eidsvik [82]. We show the probabilistic approach of the derivation for our case in A.1. Looking at the integral in the derivation, had we selected a truncated Gaussian to model the LMT, we would still get a CSN for the predicted distribution. This is because instead of integrating between the limits of $(-\infty,\infty)$ one would have integrated between $(-\infty, m_t)$ which would again result in a CSN (Lemma 5.1 in He et al. [85]). After the derivation, the final expressions for the parameters of the CSN are as follows.

$$\begin{aligned}
&\begin{bmatrix} m_{t|t-1} \\ s_{t|t-1} \end{bmatrix} = \tilde{A} \begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix}; \\
&\Sigma_{t|t-1} = \tilde{A}\Sigma_{t-1|t-1}\tilde{A}' + \Sigma_S; \\
&\Gamma_{t|t-1} = \begin{bmatrix} \Gamma_{t-1|t-1}\Sigma_{t-1|t-1}\tilde{A}'\Sigma_{t|t-1}^{-1} \\ \gamma\Sigma_S\Sigma_{t|t-1}^{-1} \end{bmatrix}; \\
&\nu_{t|t-1} = \begin{bmatrix} \nu_{t-1|t-1} \\ 0 \end{bmatrix}; \\
&\Delta_{t|t-1} = \begin{bmatrix} \Delta_{t-1|t-1} & \mathbf{0} \\ \mathbf{0} & \delta \end{bmatrix} + \begin{bmatrix} \Gamma_{t-1|t-1} \\ -\gamma\tilde{A} \end{bmatrix} (I - J_t\tilde{A})\Sigma_{t-1|t-1} \begin{bmatrix} \Gamma'_{t-1|t-1} & -\tilde{A}'\gamma' \end{bmatrix}
\end{aligned} \tag{5.42}$$

with $J_t = \Sigma_{t-1|t-1}\tilde{A}'(\tilde{A}\Sigma_{t-1|t-1}\tilde{A}' + \Sigma_S)^{-1}$. It can be observed that the cdf parameters of the CSN involve augmentation of prior and noise cdf parameters. This leads to an increase in the dimensionality of the skewness term of the resulting CSN from $t-1$ to t. Finally, at the end of the prediction step, we have a CSN expressed as follows.

$$p(m_t, s_t|Y_{t-1}) = CSN_{n_s+1,t}\left(\begin{bmatrix} m_t \\ s_t \end{bmatrix}; \begin{bmatrix} m_{t|t-1} \\ s_{t|t-1} \end{bmatrix}, \Sigma_{t|t-1}, \Gamma_{t|t-1}, \nu_{t|t-1}, \Delta_{t|t-1}\right) \tag{5.43}$$

Update step: On observing the measurement at t, the update step updates the prior pdf of $p(m_t, s_t|Y_{t-1})$ to the posterior $p(m_t, s_t|Y_t)$ using the Bayesian rule as given in (5.35). As the measurement noise is of a Gaussian distribution, the derivation of the update step is simpler than the prediction step and the details of the derivation are provided in A.2. After this implementation of the Bayesian rule, the following

expressions to update the parameters are obtained.

$$\begin{aligned}
&\begin{bmatrix} m_{t|t} \\ s_{t|t} \end{bmatrix} = \begin{bmatrix} m_{t|t-1} \\ s_{t|t-1} \end{bmatrix} + K_t(y_t - H_1 m_{t|t-1} - H_2 s_{t|t-1}); \\
&\Sigma_{t|t} = (I - K_t H)\Sigma_{t|t-1}; \\
&\Gamma_{t|t} = \Gamma_{t|t-1}; \\
&\nu_{t|t} = \nu_{t|t-1} - \Gamma_{t|t-1} \begin{bmatrix} m_{t|t} - m_{t|t-1} \\ s_{t|t} - s_{t|t-1} \end{bmatrix}; \\
&\Delta_{t|t} = \Delta_{t|t-1}
\end{aligned} \tag{5.44}$$

where $K_t = \Sigma_{t|t-1}H'(H\Sigma_{t|t-1}H' + \Sigma_u)^{-1}$. We finally have a CSN given as follows.

$$p(m_t, s_t|Y_t) = CSN_{n_s+1,t}\left(\begin{bmatrix} m_t \\ s_t \end{bmatrix}; \begin{bmatrix} m_{t|t} \\ s_{t|t} \end{bmatrix}, \Sigma_{t|t}, \Gamma_{t|t}, \nu_{t|t}, \Delta_{t|t}\right) \tag{5.45}$$

One can see that the dimensionality of the skewness term of the CSN has remained the same in the update step. This is because of the Gaussian noise of the observation pdf which has no skewness in it. So at every time instant, there is an increase in the dimensionality by 1 due to the prediction step. Finally, at the end of these recursions in the forward pass, we would have a CSN with a T dimensional skewness term.

5.3.3.3 Backward pass - Smoothing

From the update equations for the parameters, it can be observed that the distributions $p(m_t, s_t|Y_T)$ are needed. In the forward pass, only $p(m_t, s_t|Y_t)$ are estimated and hence one needs to move backward recursively updating all the parameters when all the observations are observed. This is a smoothing procedure. The recursive estimates for the smoothened parameters can be derived using (5.36). The detailed derivation is presented in A.3. As evident from (5.36) and A.3, the smoothing step is the most difficult to derive. To the best of our knowledge, such equations for the case when the process noise is CSN or truncated Gaussian, have not been derived. The recursive estimates of the parameters are provided in this section.

At the end of the forward pass, we would have the distribution

$$p(m_T, s_T|Y_T) = CSN_{n_s+1,T}\left(\begin{bmatrix} m_T \\ s_T \end{bmatrix}; \begin{bmatrix} m_{T|T} \\ s_{T|T} \end{bmatrix}, \Sigma_{T|T}, \Gamma_{T|T}, \nu_{T|T}, \Delta_{T|T}\right) \tag{5.46}$$

From this, we start to move backwards by estimating $p(m_{T-1}, s_{T-1}|Y_T)$, and so on till $t = 1$ is reached. At any time t, the equations for the recursive estimates of the

smoothened distributions are as follows.

$$
\begin{aligned}
C_t &= \Sigma_{t+1|T} J'_{t+1} (J_{t+1} \Sigma_{t+1|T} J'_{t+1} + (I - J_{t+1}\tilde{A})\Sigma_{t|t})^{-1} \\
\Sigma^*_t &= (I - C_t J_{t+1})\Sigma_{t+1|T} \\
\Gamma^*_t &= \begin{bmatrix} \gamma(C_t - \tilde{A}) \\ \Gamma^*_{t+1} C_t \end{bmatrix} \\
\nu^*_t &= \begin{bmatrix} -\gamma \begin{bmatrix} m_{t+1|T} - m_{t|T} \\ s_{t+1|T} - A s_{t|T} \end{bmatrix} \\ \nu^*_{t+1} \end{bmatrix} \\
\delta^*_t &= \begin{bmatrix} \delta & \mathbf{0} \\ \mathbf{0} & \delta^*_{t+1} \end{bmatrix} + \begin{bmatrix} \gamma \\ \Gamma^*_{t+1} \end{bmatrix} \Sigma^*_t \begin{bmatrix} \gamma' & \Gamma^*_{t+1}{}' \end{bmatrix} \\
\begin{bmatrix} m_{t|T} \\ s_{t|T} \end{bmatrix} &= \begin{bmatrix} m_{t|t} \\ s_{t|t} \end{bmatrix} + J_{t+1} \begin{bmatrix} m_{t+1|T} - m_{t+1|t} \\ s_{t+1|T} - s_{t+1|t} \end{bmatrix} \\
\Sigma_{t|T} &= J_{t+1} \Sigma_{t+1|T} J'_{t+1} + (I - J_{t+1}\tilde{A})\Sigma_{t|t} \\
\Gamma_{t|T} &= \begin{bmatrix} \Gamma_{t|t} \\ \Gamma^*_t \end{bmatrix} \\
\nu_{t|T} &= \begin{bmatrix} \nu_{t|t} - \Gamma_{t|t} \begin{bmatrix} m_{t|T} - m_{t|t} \\ s_{t|T} - s_{t|t} \end{bmatrix} \\ \nu^*_t \end{bmatrix} \\
\Delta_{t|T} &= \begin{bmatrix} \Delta_{t|t} & \mathbf{0} \\ \mathbf{0} & \delta^*_t \end{bmatrix}
\end{aligned} \tag{5.47}
$$

At every time instant, the distribution is

$$
p(m_t, s_t | Y_T) = CSN_{n_s+1,T} \left(\begin{bmatrix} m_t \\ s_t \end{bmatrix} ; \begin{bmatrix} m_{t|T} \\ s_{t|T} \end{bmatrix}, \Sigma_{t|T}, \Gamma_{t|T}, \nu_{t|T}, \Delta_{t|T} \right) \tag{5.48}
$$

It can be observe from the equations that in the backward pass the dimension of the skewness term of the CSN stays the same. With these parameters, we have the distributions of $m_t, s_t|Y_T$. From these, the necessary first-order moments of $\mathbb{E}[m_t]$, $\mathbb{E}[s_t]$, $\mathbb{E}[[m'_t\ s'_t]]$ and the second-order moments of $\mathbb{E}[m_t^2]$, $\mathbb{E}[s_t s'_t]$, $\mathbb{E}\left[\begin{bmatrix} m_t \\ s_t \end{bmatrix} [m'_t\ s'_t] \right]$ can be calculated. First-order moments are nothing but the means of the corresponding distributions and the second-order moments are the covariance + mean $\times$ mean' of the corresponding distributions.

5.3.3.4 The cross-time joint distribution

Another important required term is the cross covariance term, which can be seen in the equation for updating the parameters corresponding to the transition pdf which

include A, Σ_v and σ_e^2. The required expectations are $\mathbb{E}[s_t s'_{t-1}]$ and $\mathbb{E}[m_t m_{t-1}]$. To estimate them, we first need to form the distribution of $p(m_{t-1}, s_{t-1}, m_t, s_t|Y_T)$. For CSN, if the conditional and the marginal distributions are CSN, then the joint distribution is also a CSN. Hence, $p(m_{t-1}, s_{t-1}\ , m_t, s_t|Y_T)$ is also a CSN. The detailed derivation of the parameters is given in A.4.

$$\begin{aligned} & p(m_{t-1}, s_{t-1}, m_t, s_t|Y_T) \\ =& CSN_{2n_s+2,T}\left(\begin{bmatrix} m_{t-1} \\ s_{t-1} \\ m_t \\ s_t \end{bmatrix};\ \mu_{t-1,t|T}, \Sigma_{t-1,t|T}, \Gamma_{t-1,t|T}, \nu_{t-1,t|T}, \Delta_{t-1,t|T}\right) \end{aligned} \tag{5.49}$$

The parameters of the above CSN are calculated according to the following equations.

$$\begin{aligned} \mu_{t-1,t|T} &= \begin{bmatrix} \begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix} + J_t \begin{bmatrix} m_{t|T} - m_{t|t-1} \\ s_{t|T} - s_{t|t-1} \end{bmatrix} \\ \begin{bmatrix} m_{t|T} \\ s_{t|T} \end{bmatrix} \end{bmatrix}; \\ \Sigma_{t-1,t|T} &= \begin{bmatrix} J_t\Sigma_{t|T}J'_t + (I - J_t\tilde{A})\Sigma_{t-1|t-1} & J_t\Sigma_{t|T} \\ \Sigma_{t|T}J'_t & \Sigma_{t|T} \end{bmatrix}; \\ \Gamma_{t-1,t|T} &= \begin{bmatrix} \gamma(I_2 - \tilde{A}I_1) \\ \Gamma_{t-1|t-1}I_1 \\ \Gamma_t^* I_2 \end{bmatrix}; \\ \nu_{t-1,t|T} &= \begin{bmatrix} -\gamma(I_2 - \tilde{A}I_1)\mu_{t-1,t|T} \\ \nu_{t-1|t-1} - \Gamma_{t-1|t-1}\left(I_1\mu_{t-1,t|T} - \begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix}\right) \\ \nu_t^* - \Gamma_t^*\left(I_2\mu_{t-1,t|T} - \begin{bmatrix} m_{t|T} \\ s_{t|T} \end{bmatrix}\right) \end{bmatrix}; \\ \Delta_{t-1,t|T} &= \begin{bmatrix} \delta & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Delta_{t-1|t-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \delta_t^* \end{bmatrix} \end{aligned} \tag{5.50}$$

with I_1 and I_2 defined as in (A.31) and (A.32). One can observe that the dimensionality of the cdf part of the above CSN is T. For example, the dimensionality of δ, $\Delta_{t-1|t-1}$, and δ_t^* are 1×1, $(t-1) \times (t-1)$, and $(T-t) \times (T-t)$ respectively which combine to give a total of $T \times T$.

This completes the recursive equations of the parameters to estimate the distributions $p(m_t, s_t|Y_T)$. Now to estimate the expected value of the quantities appearing in the parameter estimation equations, one needs to estimate the moments of each of the pdfs. For Gaussian cases it is easy to do this as the parameters of a Gaussian

distribution are the mean and the covariance. But for a CSN this is not the case and we discuss how to evaluate the mean and the covariance matrix of a CSN in the next section.

5.3.3.5 Evaluating the expectations

As discussed earlier the μ and the Σ parameter are not the true mean and covariance of the CSN distribution. The expressions for the moments of a CSN have analytical expressions only for the case where the dimensionality of the skewness term is 1 [153]. For higher dimensionality, there are no analytical expressions. We hence resort to a sampling based-method. For a CSN described in (5.8), we can calculate the mean and variance as

$$\begin{aligned} \mathbb{E}[x] &= \mu - K\nu + K\mathbb{E}[p]; \\ \mathrm{Var}[x] &= \Sigma + K\Gamma\Sigma + K(\mathrm{Var}[p])K' \end{aligned} \tag{5.51}$$

with $K = -\Sigma\Gamma'(\Delta + \Gamma\Sigma\Gamma')^{-1}$. Here, p is a dummy variable whose mean and variance are calculated from the N samples drawn from the following truncated Gaussian distribution.

$$\begin{aligned} p &\sim \frac{\mathcal{N}_q(p;\ \nu, \Delta + \Gamma\Sigma\Gamma')}{\Phi_q(0;\ \nu, \Delta + \Gamma\Sigma\Gamma')}\mathbf{1}(p \leq 0) \\ \mathbb{E}[p] &= \frac{1}{N}\sum_{i=1}^{N} p_i; \quad \mathrm{Var}[p] = \frac{1}{N}\sum_{i=1}^{N}(p_i - \mathbb{E}[p])(p_i - \mathbb{E}[p])' \end{aligned} \tag{5.52}$$

The details of these expressions are discussed in A.5. With this, all the necessary equations to calculate the expectations appearing in the parameter update equations of the M-step are derived, and hence the E-step is concluded.

The issue of dimensionality: It is to be noted that the dimension of p is nothing but the dimension of the skewness term of the CSN. For T samples, the dimension of p is T and hence one would need to sample the variable p from a T-dimensional truncated Gaussian distribution. This could result in some computational burden. One possible way of avoiding this is to keep the dimension of the skewness term of the CSN at a constant number $k << T$. To achieve this, one has to first approximate the high dimensional CSN to a lower one at every point in the forward pass. For example,

$k = 1$ requires the CSN to be approximated as a Gaussian at every time instant at the end of the update step. This would not only decrease the accuracy, but also make the derivation of recursive estimates for smoothing not possible, necessitating the usage of particle filters to estimate the distributions of $p(m_t, s_t|Y_T)$. Given this, and also particularly when the application of this model is done off-line, the computational burden caused by the dimensionality of the skewness term may be tolerated, and the original solution can be applied.

Online implementation: The proposed approach results in a simultaneous state and parameter estimation solution. The framework of the implementation of the algorithm for monitoring deterioration and other stationary variations depends on the timescale of the deterioration. For processes that degrade over smaller timescales (of the order of weeks to a few months), it would be preferable to visualize the trends very frequently. In such cases, the iterative estimation procedure which is computationally expensive needs to be performed offline based on historical datasets of the same process (previous cycles of operation). With a reasonable model available, only the state estimation portion may be done online. Although, it needs to be noted that the parameters may differ between different cycles of degradation and one may need to intermittently perform the EM algorithm iterations to adjust the parameters to the current cycle. For degrading processes with larger time scales (years), the intervals at which the monitoring of degradation is required are larger. For such cases, the model is run mostly offline, and the EM algorithm iterations can be performed whenever there is a need to visualize the deterioration.

5.4 Case studies

In this section, we present the results obtained from the implementation of the proposed approach on two case studies. The first one is a simulated case study where the data is artificially generated from a linear system with LMT and LST. The second one is an industrial case study where the proposed approach is applied on a hot lime softener (HLS) process dataset to monitor the fouling buildup in the process.

5.4.1 Simulation case study

Given our objective of separating the LMT and LST from a non-stationery process, we mimic such behavior in this simulated case study. The system under consideration has two observed variables. These are assumed to be a mixture of one LMT and two LSTs. A sigmoid function has been used in our case to generate the LMT. The sigmoid function is a monotonically increasing function and the rate of increase varies from near zero at the extreme ends to very high in the middle, hence giving a good variation in the function as far as monotonically increasing ones are concerned. The following sigmoid function is used in the case study.

$$m_t = \frac{10}{1 + exp(-0.04 * (t - 125))} \tag{5.53}$$

The stationary signals are generated using a state-space model with the absolute value of the eigenvalues being less than 1. For this case study, the following equation is used to generate the LST.

$$s_t = \begin{bmatrix} 0.95 & 0.5 \\ -0.4 & 0.25 \end{bmatrix} s_{t-1} + \Sigma_v \tag{5.54}$$

The eigenvalues of the A matrix are $0.6000 \pm 0.2784i$. Hence the system shows some oscillatory behavior also. For simplicity, the noise covariances Σ_u nd Σ_v are selected to be identity matrices. The inputs generated for this case study are depicted in Fig. 5.3. The outputs are generated according to the following equations.

$$y_t = \begin{bmatrix} 1 \\ -1 \end{bmatrix} m_t + \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} s_t + \Sigma_u \tag{5.55}$$

The coefficients are selected to be one in this case as LMT and LST both have a 'similar' variation and hence similar contributions of each signal are selected to generate the output. A total of 250 samples are generated. The output data generated for this system is depicted in Fig. 5.4. From the figure, it can be observed that the overall data has a non-stationary characteristic. But this is not apparent because of the stationary trends that are mixed with it.

Although co-integration analysis and stationary subspace analysis can be used to separate the stationary and non-stationary components, in this case, one would only be able to get two latent variables as there are two outputs. This would not result in

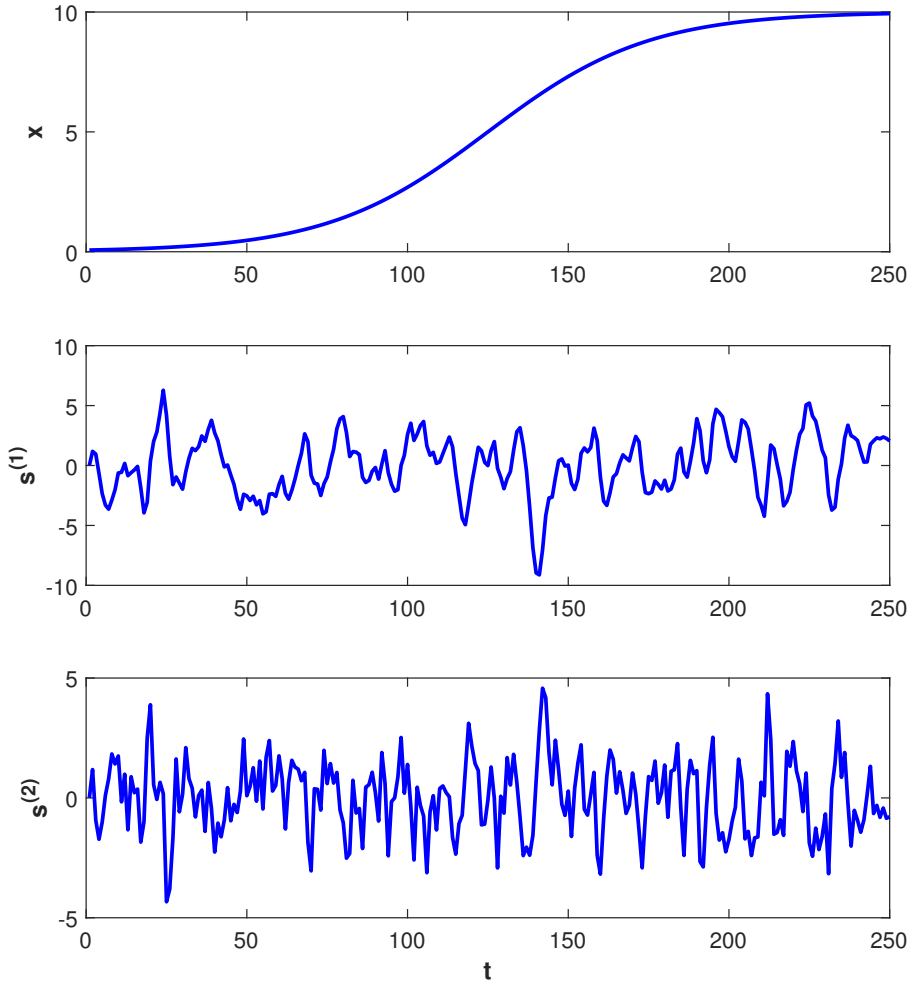


Figure 5.3: The latent variables generated for the simulation case study. x is the sigmoidal monotonic function, and $s^{(1)}$ and $s^{(2)}$ are the two stationary signals

a proper separation of the stationary and non-stationary components. The stationary space will have a mixture of the two stationary signals and the non-stationary space would have a mixture of the stationary and non-stationary signals. Hence the results obtained from the proposed method are compared with an algorithm in which a Gaussian distribution is assumed. For the Gaussian case, the M-step turns out to be similar to the M-step we derived for the CSN. The E-step is nothing but the original Kalman filter followed by an RTS smoother. These two steps are alternatively performed till convergence. For the CSN algorithm, since there are 250 samples, the

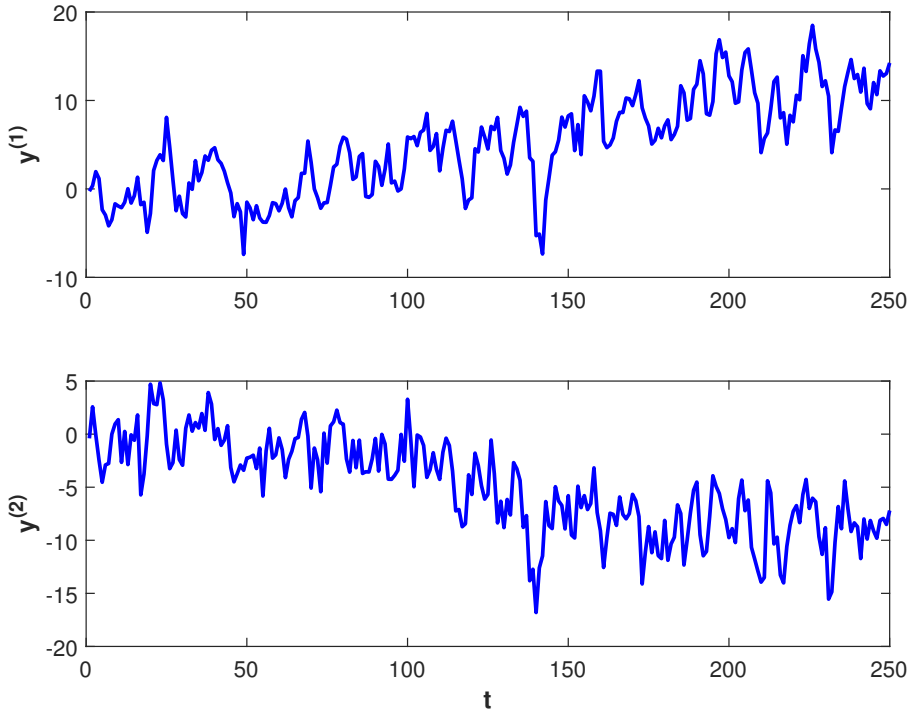


Figure 5.4: The outputs generated for the simulation case study

size of the CSN at every instant is 250. This requires a large number of samples to estimate the moments of the CSN at each step. It increases the computation time significantly.

This problem can be overcome to some extent by using a good initial guess for the EM algorithm. The EM algorithm can result in a local minimum solution and the initial guess may influence the final results. In our case, we first trained the Gaussian model on the data and used the final values of the parameters as initial guesses for the CSN model. This is equivalent to first training the CSN model with a ρ being equal to zero and then subsequently increasing ρ. This reduced the number of iterations of the EM algorithm for the CSN model, hence saving some time. For this case study, a ρ of 0.8 was able to ensure that the extracted x is monotonic. The results of the simulations are presented in Table. 5.1. The Gaussian model is compared with the CSN model in terms of the sum of the squared errors (SSE). The SSE of the state estimation is calculated w.r.t to the generated data. The SSE is calculated according

to the following equation.

$$SSE_m = \sum_{t=1}^{T}(m_t - \hat{m}_t)^2; \quad SSE_{s^{(i)}} = \sum_{t=1}^{T}(s_t^{(i)} - \hat{s}_t^{(i)})^2 \tag{5.56}$$

Here, m_t and $s_t^{(i)}$ represent the states generated according to (5.53) and (5.54). Variables $\hat{m}_t$ and $\hat{s}_t^{(i)}$ represent the estimated states. It can be observed from Table. 5.1 that the SSE for the sigmoid signal is larger for the CSN model case. However, for the stationary states, the CSN model outperforms the Gaussian model. These phenomena can be explained as follows. The state estimation for the CSN model essentially is a constrained estimation, with the constraint ensuring a monotonic trend. The Gaussian model allows the non-stationary trend to vary freely hence reducing the SSE. But this would violate the monotonic trend nature of the signal. Since $y_t = H_1 m_t + H_2 s_t$, the violation of the constraint seen in x, induces an error in the estimated stationary state s. Since the estimated non-stationary state preserves the monotonic nature of the signal, it practically improves the estimation of the stationary state. Hence it is observed that the proposed method increases the accuracy of the extracted stationary signals. This can also be observed in the scatter plots of the estimated vs. actual values of the signals depicted in Fig. 5.5, 5.6 and 5.7.

The non-stationary signal extracted by the Gaussian model is not monotonic in nature. This can be observed in Fig. 5.8 which shows the rates of change of the non-stationary signals extracted using a Gaussian model and a CSN model. The one from Gaussian is not monotonic as the rate attains negative values at certain points and the one from the CSN model is monotonic. It can also be observed that the CSN one is smoother than the Gaussian one resembling a trend more similar to the original sigmoid function.

Table 5.1: Comparing the SSE obtained for the Gaussian model and the CSN model

Method	SSE_m	$SSE_{s^{(1)}}$	$SSE_{s^{(2)}}$	Total SSE of stationary features
Gaussian model	1.1714	183.7478	88.1470	273.0662
Proposed CSN model	6.9707	142.2412	35.6361	184.8480

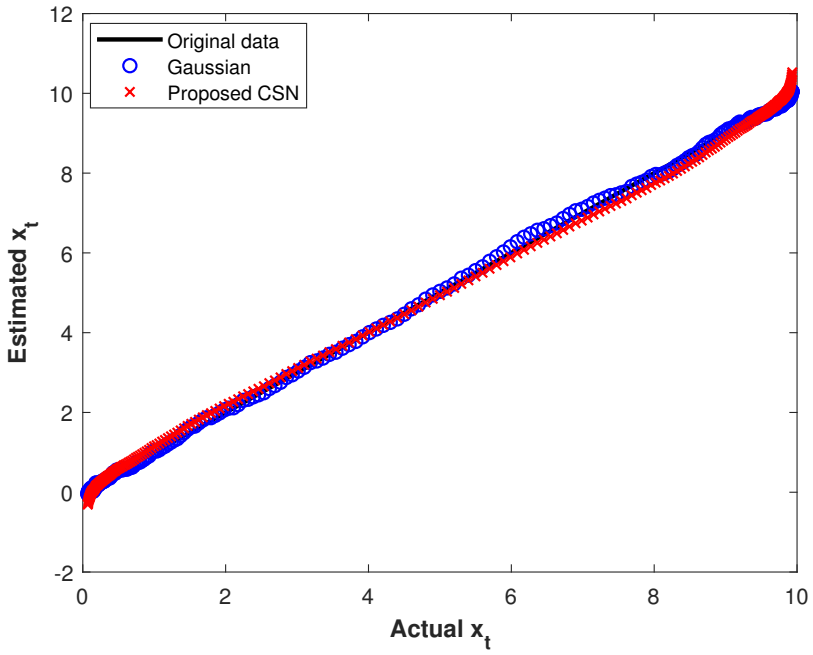


Figure 5.5: Comparison of the scatter plot of the monotonic signal extracted from the Gaussian and CSN models

5.4.2 Fouling monitoring in a Hot Lime Softener

Hot lime softener (HLS) is an industrial process used to remove the hardness, silica, etc from water. The hard water is mixed with chemicals such as lime, soda ash, etc in a tank. The resulting mixture contains suspended solids and is sent for filtering through a piping section to remove the solids. This piping section suffers from fouling due to these suspended solids, which increases the cost of operation. On the other hand, frequent cleaning would need the process to be shut down, hence reducing the in-operation time. Monitoring the fouling buildup hence becomes essential for the efficient operation of the process. The data used in this case study is obtained from an industrial HLS used to soften the boiler feed water in a steam-assisted gravity drainage (SAGD) process. The fouling indicator is based on the Darcy–Weisbach equation, according to which

$$\left(\frac{F}{\sqrt{\Delta P}}\right)^{0.8} \propto A. \tag{5.57}$$

Here, F is the flowrate through the piping section, ΔP is the pressure drop and A is the cross-section area. As the fouling material deposits during the course of the

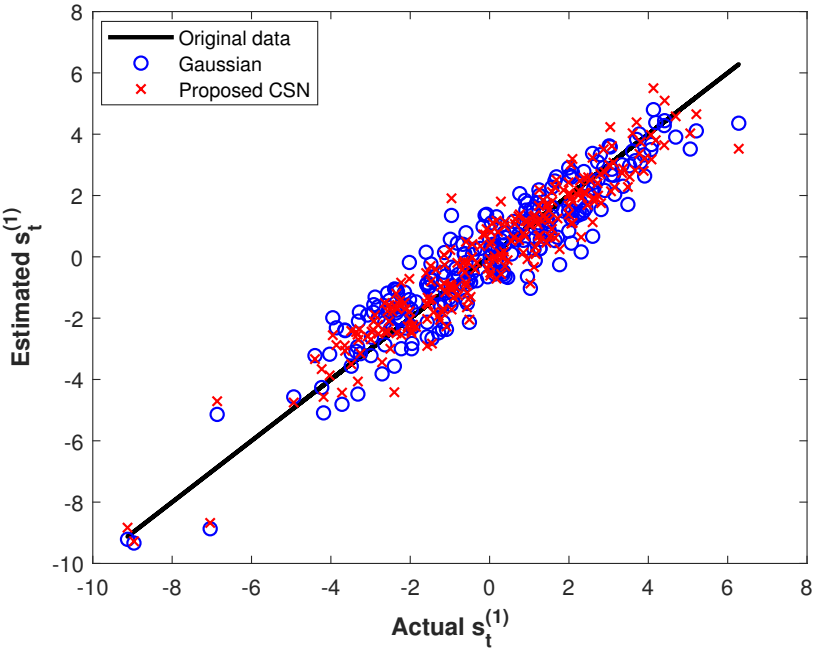


Figure 5.6: Comparison of the scatter plot of the first stationary signal extracted from the Gaussian and CSN models

operation, the area A decreases gradually. Hence monitoring $(F/\sqrt{\Delta P})^{0.8}$, which we will refer to as the flow coefficient, will help us to monitor the fouling. Alsadaie and Mujtaba [154] categorize fouling into different categories based on the interaction between fouling deposition and removal rates. These categories are the linear rate, falling rate, asymptotic rate, and sawtooth behavior. In our case, the saw-tooth nature is seen where there is an initial increase in fouling followed by an oscillatory behavior. This oscillatory behavior is due to the intermittent flushing away of the deposited fouling substance. The flow coefficient calculated for the data is depicted in Fig. 5.9. These oscillations increase over time and one could use this as an indication of fouling buildup. But these oscillations are mixed with the non-stationary trend and for the monitoring purpose, we need to separate them. Hence we model the system

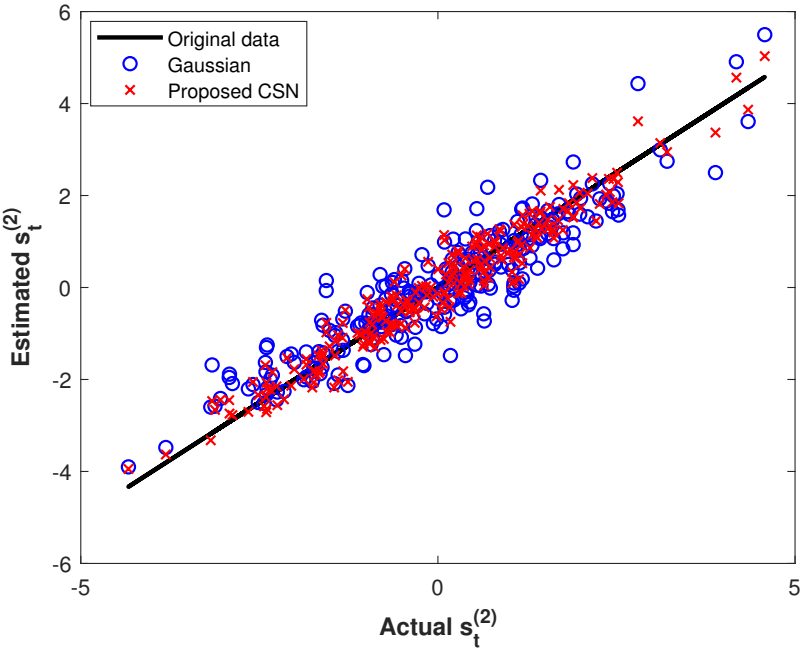


Figure 5.7: Comparison of the scatter plot of the second stationary signal extracted from the Gaussian and CSN models

as follows.

$$
\begin{aligned}
s_t &= A\ s_{t-1} + v_t \\
m_t &= m_{t-1} + e_t \\
r_t &= w_t \\
\left(\frac{F}{\sqrt{\Delta P}}\right)^{0.8} &= y_t = H_1\ m_t + H_2\ s_t + u_t
\end{aligned}
\tag{5.58}
$$

Here the dimensionality of x and s is 1. The monitoring strategy for this case is summarized below.

1. Calculate the flow coefficient according to (5.57).

2. Implement the proposed algorithm to separate the LMT and LST.

3. Fig. 5.9 indicates an LMT superimposed by an oscillating variable. Use the LMT to infer the rate of fouling buildup. A non-monotonic signal extracted by a Gaussian model cannot be used to monitor the rate of fouling.

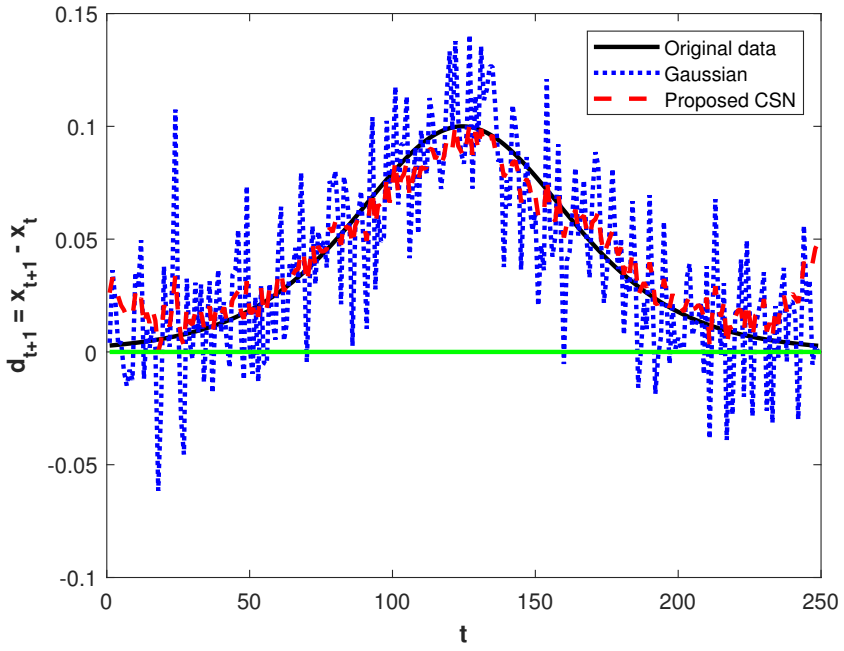


Figure 5.8: Comparison of the rate of change of the monotonic signals obtained from the Gaussian and CSN models

4. Use the stationary trend to monitor the oscillations caused by flushing away of the fouling material. The decision about cleaning can be made when the magnitude of the oscillations crosses a certain threshold.

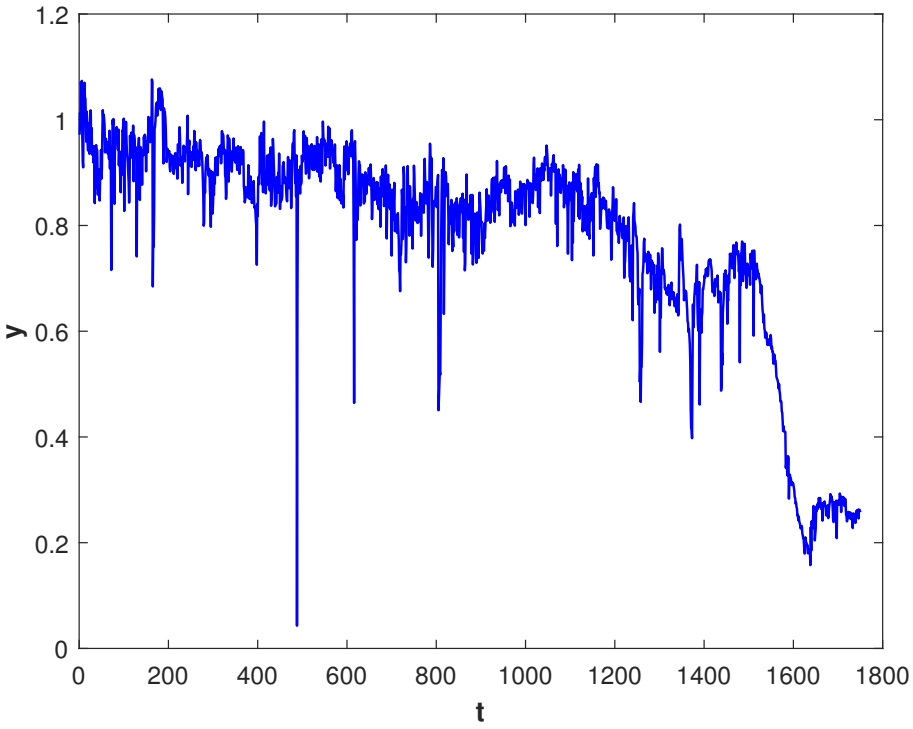


Figure 5.9: The flow coefficient calculated for dataset 1

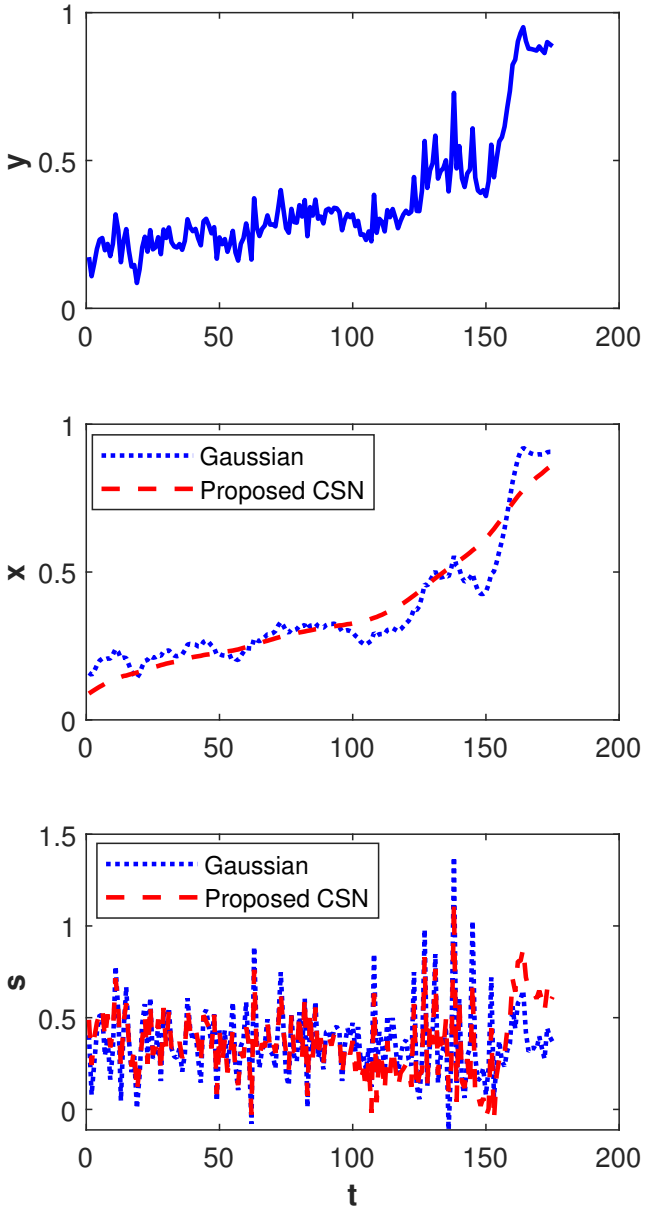


Figure 5.10: Summary of the fouling monitoring results for the first dataset. The top figure shows the calculated flow coefficient. The other two depict the LMT and LST respectively.

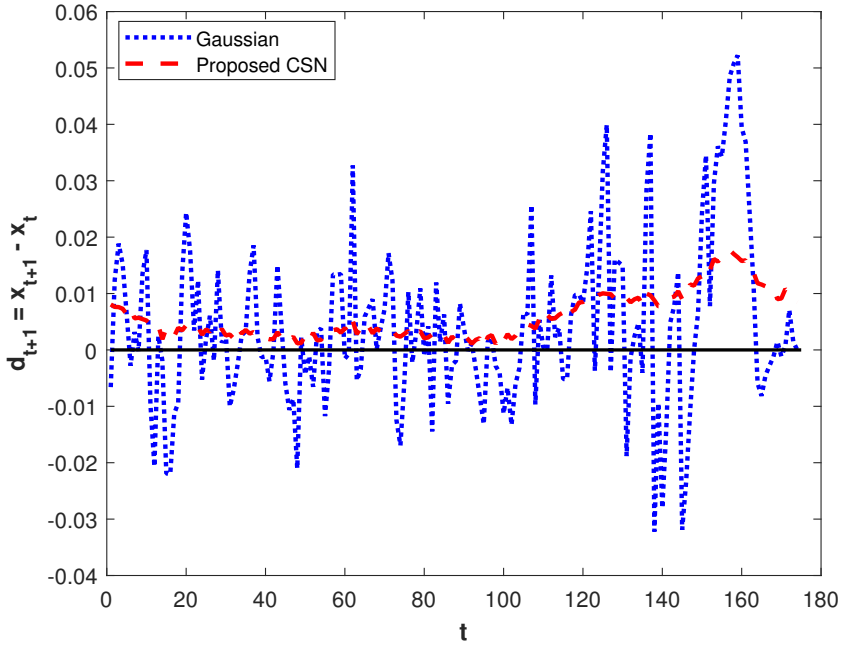


Figure 5.11: Rate of change of the LMT obtained from the first fouling dataset

The proposed method is implemented on the fouling dataset and the results are shown in Fig. 5.10. We have down-sampled the dataset for both cases so that the dimensionality of the CSN does not go too high. Also, the flow coefficient is a decreasing quantity and we have converted it to an increasing one by only subtracting the trend from its maximum value to facilitate visualization. This does not change the trend in any way as all the temporal relations and covariances are still intact. From the results in Fig. 5.10, it can be observed that the Gaussian model does not extract a signal even close to the monotonic nature. The LST extracted shows a trend of increasing oscillations for both the Gaussian and CSN models. But this nature is more apparent for LST obtained from the proposed method. The rate of change of the LMT is depicted in Fig. 5.11. As expected, the rate of change is oscillatory and takes values on both sides of the zero line for the Gaussian model. As a result of this, no clear conclusions can be made about the rate of fouling by looking at this trend. For the CSN model, the trend is smoother and will be suitable for monitoring purposes. This shows that the proposed approach is more efficient in extracting the two sets of latent features.

5.5 Conclusion

We propose an approach to perform the simultaneous state and parameter estimation of a process driven by both latent monotonic and stationary trends. The observed data is assumed to be a linear mixture of these two trends and we propose an EM algorithm-based approach to separate them. The problem is modeled as a closed skew-normal distribution process and we have derived recursive estimation procedures for the E-step (state estimation) of the algorithm. The method is suited for both off-line modeling and on-line monitoring applications for processes where there is a slow monotonic drift in the process operation due to degrading health factors of the process. The results from the simulated and industrial case studies show a successful demonstration of the proposed method. These promising results prompt further exploration of process-relevant probabilistic modeling of industrial processes using distributions beyond the standard Gaussian distribution.

Chapter 6

Modeling and Bayesian Inference for Processes Characterized by Impulsive Changes

This chapter presents the fourth contribution of the thesis which proposes a method of modeling and estimating systems characterized by abrupt (impulsive) changes. These impulsive changes may be due to multiple reasons such as disturbances, capacity change, etc. All these cases result in signals that appear to have sudden jumps in the process. But mixed along with these jumps will be the other dynamic variations characterizing the regular dynamics of the process. Hence, for effective modeling of such processes, it is important to model both the jumps and the regular dynamic variations. In chapters 3 and 4 it was argued that temporal slowness is the main characteristic of most of the chemical processes. Hence, we model the other dynamic variations using the probabilistic slow feature analysis (PSFA) model. The resulting model has two types of latent variables (LVs) each characterizing the abrupt jumps and the slower variations. The inference of the states and parameters is done in the variational Bayesian (VB) framework. The efficacy of the proposed approach is demonstrated as a soft sensor application in both a simulated case study and industrial case study.

6.1 Introduction

Signals with impulsive changes or abrupt jumps are a common occurrence in many industrial datasets. There could be multiple reasons for such behavior. One such

reason is the abrupt injection of a disturbance into the system. This will cause an abrupt change not only in the variables in the immediate vicinity of the injected disturbance but might also travel to many other variables due to the coupled nature of the processes. Another way that an abrupt change is observed is when there is a sudden capacity change either due to the changes in the supply/demand side or due to the availability/unavailability of certain units. In other cases, the process itself could be of impulsive nature. The source of these impulsive jumps will be a common one and the source itself may not be measured. In such cases, an LV model that models these jumps is more desirable. Also, these jumps will be mixed with other sources of dynamic variations which characterize the dynamic relations of various process variables. The separation of these two sources thus becomes important for effective modeling of the process. Hence, modeling of such processes must consider these two different sources of process behavior, i.e., the abrupt jumps and the regular dynamics in order to capture them effectively.

The literature related to processes with impulsive behaviors has explored various methods such as the optimization-based approaches to fit piecewise constant signals, state space estimation using fat-tailed distribution, etc as discussed in section 1.2.3. The state space approach is more appropriate because the stochastic time series nature of the model provides more flexibility in modeling through the incorporation of various types of distributions, special structures in the model matrices, etc. Moreover, for online implementation, a dynamic model is always better suited. In the state space model, if a dynamic variable trajectory is to be generated with abrupt jumps, then a Gaussian distribution cannot be used as the process noise. Rather, a heavy-tailed distribution must be used because the abrupt jumps can be realized only if there is a significant probability of obtaining realizations of extreme values. The Cauchy distribution is a heavy-tailed distribution and has a closed-form expression for its pdf unlike many heavy-tailed distributions in the α-stable distribution family. Hence, it can be used to model the dynamics of the LV that corresponds to the impulsive sources of variations. Fig. 6.1 compares the pdfs and random-walks generated by Gaussian and Cauchy distributions.

As discussed earlier, in addition to the abrupt jumps, there will be also other regular variations. These can be modeled according to the Gaussian distribution. As

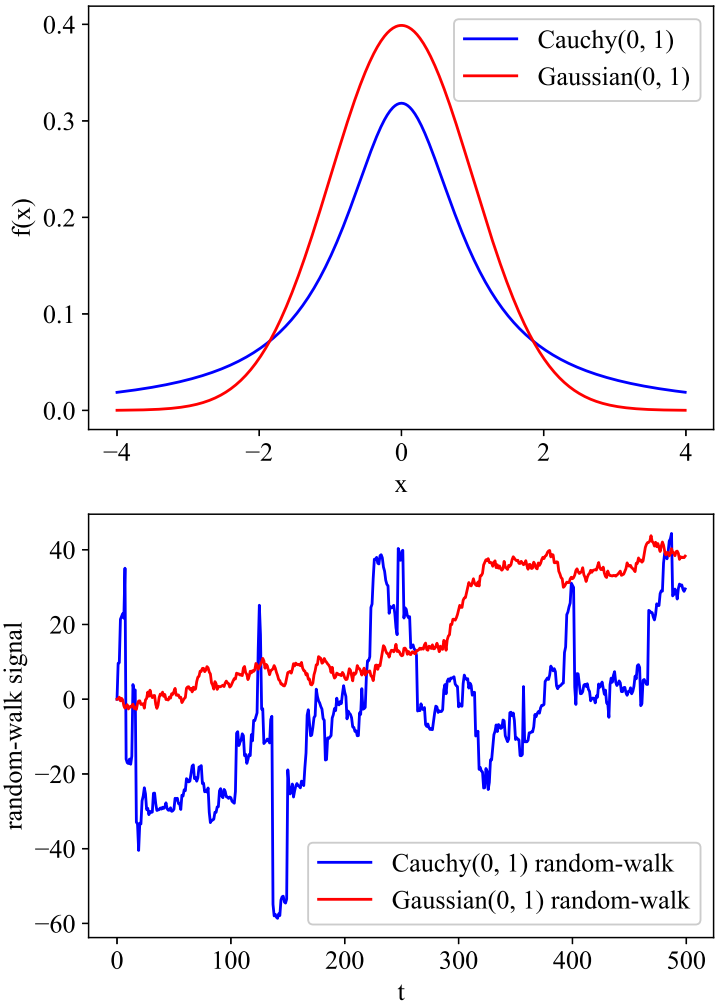


Figure 6.1: Comparison of the Cauchy and the Gaussian distributions. The top figure compares the pdfs of the two distributions and the bottom one compares the realizations of a random-walk model of the form $s_t = s_{t-1} + v_t$ with v_t modeled as the Cauchy and Gaussian distributions.

discussed in chapters 3 and 4, temporal slowness is an important characteristic of chemical processes. Hence the latent variables related to the process variations are slow. Hence, in this case, the Gaussian latent variables are modeled according to the PSFA model to account for the temporal slowness of the sources driving the process.

This hence results in a model with two types of LVs: Cauchy LV to account for the abrupt changes and Gaussian LVs to account for the slower variations. This results in a state (LV) and parameter identification problem of a state space model with the states (LVs) having Cauchy and Gaussian distributions. As discussed in section 1.2.3, the literature on impulsive processes mostly focuses on the state estimation portion where it is assumed that the system model is available. Research to perform the simultaneous state and parameter estimation is scarce. Moreover, modeling the system in the aforementioned way to describe both the impulsive sources and the slower sources of variations has not been done.

This work proposes a method to model systems that have both impulsive and slower sources of variations. The identification of the resulting model is performed in the VB inference framework [63, 64]. In this case, the model parameters too are assumed to be random variables and are inferred in a Bayesian framework combining the prior beliefs and the observed data. The VB framework is advantageous in the cases where one has certain modeling preferences, such as the one in this case related to the Gaussian LVs. The Gaussian LVs are modeled according to the PSFA model which has a particular structure. In this case, the notions of slowness are better implemented in a Bayesian framework [65–67]. To accommodate the constraints on the parameters of the model, relevant distributions are assumed as the priors. Bayesian estimation of the model is performed which involves estimating the posterior distribution of the states and the parameters. The main challenge to the posterior estimation is posed by the states and parameters involved in the Cauchy distribution. Since the Cauchy distribution does not have an exponential form, the estimation of the posteriors becomes difficult. This issue is circumvented through an approximation based on the Taylor series expansion. Hence, the parameters that have a tractable posterior are estimated through analytical update equations while the ones that do not have a tractable posterior are evaluated through importance sampling. The state estimation is conducted through particle filtering and smoothing [109, 110] as the Cauchy distribution makes the analytical state estimation procedure intractable.

The effectiveness of the proposed algorithm is demonstrated through two case studies. The proposed approach can be used for both process monitoring and soft sensor applications as both applications may need an accurate model that considers

the abrupt jumps and the slower variations. In this work, the main focus is on the soft sensor application. The first case study is a numerical example where the data is generated according to the assumed notions through an appropriate state space model. The second one is a real-world industrial case study. The data is obtained from a steam-assisted gravity drainage (SAGD) process and the objective is to predict the emulsion flow-rate.

The rest of the chapter is organized as follows. The proposed model formulation is presented in section 6.2. Section 6.3 presents the VB inference framework applied to the proposed model. Derivation of the posterior distribution of each of the parameters and states is provided. Section 6.4 presents the results from the two case studies. Finally, the conclusions drawn from the work are summarized in section 6.5.

6.2 Proposed model

The proposed model aims at modeling and estimating the impulsive behavior and the slower variations. Let $\{x_t\}$ be the set of observed input data with $x_t \in \mathbb{R}^{n_x \times 1}$, and $\{y_t\}$ be the set of the observed output data with $y_t \in \mathbb{R}^{n_y \times 1}$. The objective is to build a soft sensor model to predict y_t from x_t. Since both x_t and y_t come from a process with the aforementioned latent spaces, it would be more effective to have an LV model. Hence, two LVs are defined: $c_t \in \mathbb{R}$, whose dynamics evolves according to a Cauchy distribution, and $s_t \in \mathbb{R}^{n_s \times 1}$, whose dynamics evolve according to a Gaussian distribution. These two LVs model the abrupt jumps and the slower variations respectively. The proposed model is expressed in the following set of equations.

$$s_t = A\ s_{t-1} + v_t, \qquad v_t \sim \mathcal{N}(v_t;\ \mathbf{0}, \Sigma_v) \tag{6.1}$$

$$c_t = b\ c_{t-1} + e_t, \qquad e_t \sim \mathcal{C}(e_t;\ \mathbf{0}, \sigma_e) \tag{6.2}$$

$$x_t = M_1\ c_t + M_2\ s_t + w_t, \qquad w_t \sim \mathcal{N}(w_t;\ \mathbf{0}, \Sigma_w) \tag{6.3}$$

$$y_t = H_1\ c_t + H_2\ s_t + u_t, \qquad u_t \sim \mathcal{N}(u_t;\ \mathbf{0}, \Sigma_u) \tag{6.4}$$

Here, $\mathcal{N}$ and $\mathcal{C}$ represent Gaussian and Cauchy distributions respectively. Equation (6.1) models the evolution of the slower variations which are defined by the matrix A and the state transition noise covariance Σ_v. Since these Gaussian LVs are modeled according to the PSFA model (section 2.2.2), A and Σ_v have a special structure as

shown in the following equations.

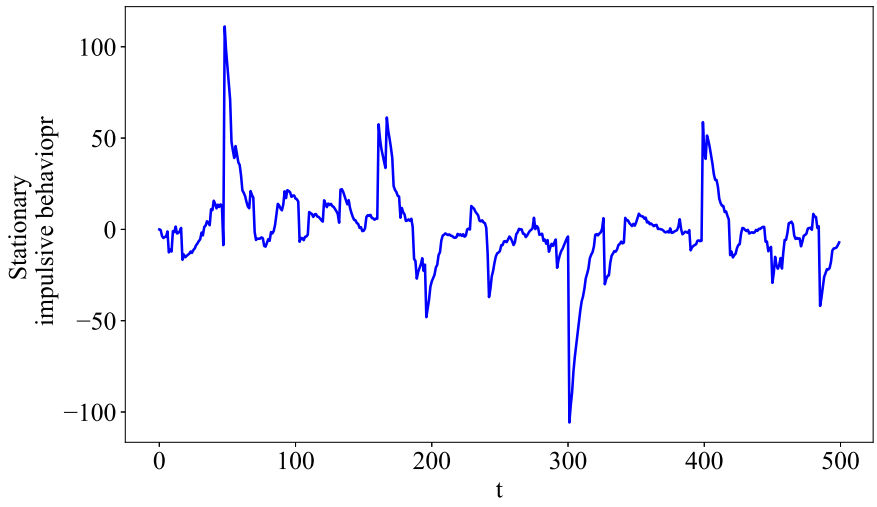
$$A = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n_s} \end{bmatrix}; \quad 0 < a_i < 1; \quad \Sigma_v = \begin{bmatrix} 1-a_1^2 & 0 & \cdots & 0 \\ 0 & 1-a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1-a_{n_s} \end{bmatrix} \tag{6.5}$$

Equation (6.2) shows the evolution of the Cauchy LV determined by the coefficient b and the state transition noise scaling parameter σ_e. The coefficient b has following constraint $0 < b \leq 1$. If b is less than 1, this results in a stationary Cauchy trajectory. This case is more suitable for the situation when the abrupt changes in the process are caused by a disturbance. Usually, such disturbances are rejected and the system is returned to the normal operation condition by a controller. Nevertheless, these jumps cannot be termed as outliers as they are not measurement errors as the abrupt jump actually happens. If a soft sensor or a monitoring model is to be built, one would either want accurate predictions in these jumps and not term them as faults or outliers. Fig. 6.2a shows such a stationary Cauchy trajectory. If $b = 1$, we get a non-stationary trajectory as this will be a random walk process. Such a trajectory may be observed in the case when the abrupt jump is caused either by a capacity change or changes in the demand or the supply side of the process. Fig. 6.2b shows such a non-stationary Cauchy trajectory. For simplicity, in this work, the abrupt jumps are modeled using a single variable and the extension to multiple sources is straightforward.

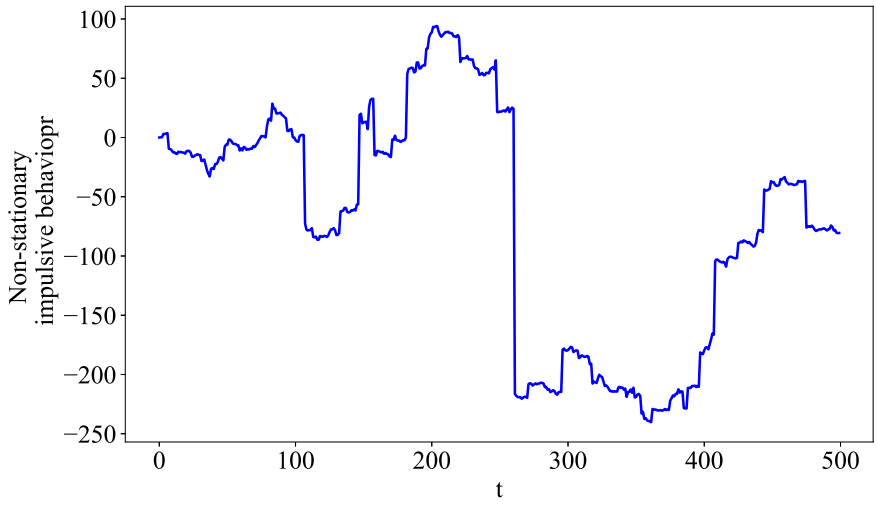
Equations (6.3) and (6.4) show how the observed input and output data are generated from the two LVs respectively. x_t is generated through the two matrices M_1 and M_2 and has a measurement noise w_t, which is white noise with a covariance matrix Σ_w. Similarly, y_t is generated through the two matrices H_1 and H_2 and has a white measurement noise of u_t with a covariance matrix Σ_u.

6.2.1 Model formulation in the probabilistic framework

In this work, the inference of the model in (6.1)-(6.4) is done in a Bayesian framework, where all the states and parameters are assumed as random variables. As discussed previously, the evolution of the states s_t and c_t are modeled according to Gaussian



(a) A stationary Cauchy dynamic process (b = 0.9)



(b) A non-stationary Cauchy dynamic process (b = 1)

Figure 6.2: Stationary and non-stationary Cauchy processes

and Cauchy distributions respectively.

$$p(s_t|s_{t-1}, A) = \mathcal{N}(s_t;\ As_{t-1}, I - AA') \tag{6.6}$$

$$p(c_t|c_{t-1}, b, \sigma_e) = \mathcal{C}(c_t;\ bc_{t-1}, \sigma_e) \tag{6.7}$$

For the output model, since the observation noise is modeled as a Gaussian distribution, the conditional distribution of the observations is represented as

$$p(x_t|c_t, s_t, M_1.M_2, \Sigma_w) = \mathcal{N}(x_t;\ M_1 c_t + M_2 s_t, \Sigma_w) \tag{6.8}$$

$$p(y_t|c_t, s_t, H_1.H_2, \Sigma_u) = \mathcal{N}(y_t;\ H_1 c_t + H_2 s_t, \Sigma_u) \tag{6.9}$$

For the parameters, the prior distribution assumed depends on the specific constraints the parameters have. The following points define the prior distribution assumed for each of the parameters.

1. Since matrix A is diagonal, the distribution for each of the diagonal elements a_i can be defined separately. Since a_i is constrained to be between 0 and 1, the prior for a_i must have distribution with a support of [0,1]. In this work, the beta distribution is used as a prior for a_i as the support of the beta distribution is between 0 and 1. The beta distribution is parameterized by two parameters α and β and a prior beta distribution can be specified for each a_i. The beta distribution for an a_i henceforth will be represented as $\mathcal{B}(a_i;\ \alpha_a^{(i)}, \beta_a^{(i)})$.

2. Due to the adoption of the PSFA model, we have $\Sigma_v = I - AA'$. In a general case, Σ_v would be an independent parameter and a suitable prior can be assumed.

3. Like a_i, b is also constrained to be between 0 and 1 and hence a beta distribution of $\mathcal{B}(b;\ \alpha_b, \beta_b)$ is assumed as the prior distribution of b.

4. The parameter σ_e is the scaling parameter of the Cauchy distribution and is a positive quantity. Hence, a gamma distribution is assumed as the prior. This is represented as $\Gamma(\sigma_e;\ \alpha_e, \beta_e)$.

5. Since, during the training phase, x and y are not distinguishable in terms of the role they play in estimating the distributions, they can be augmented to get an overall observed data vector. This results in the following expression.

$$\begin{bmatrix} y_t \\ x_t \end{bmatrix} = \boldsymbol{y}_t = \underbrace{\begin{bmatrix} H_1 & H_2 \\ M_1 & M_2 \end{bmatrix}}_{H} \begin{bmatrix} c_t \\ s_t \end{bmatrix} + \boldsymbol{u}_t; \quad \boldsymbol{u}_t \sim \mathcal{N}\left(\begin{bmatrix} u_t \\ w_t \end{bmatrix};\ \mathbf{0},\ \underbrace{\begin{bmatrix} \Sigma_u & \mathbf{0} \\ \mathbf{0} & \Sigma_w \end{bmatrix}}_{\Sigma_{\boldsymbol{u}}} \right) \tag{6.10}$$

$$\boldsymbol{y}_t = H \begin{bmatrix} c_t \\ s_t \end{bmatrix} + \boldsymbol{u}_t; \quad \boldsymbol{u}_t \sim \mathcal{N}(\boldsymbol{u}_t;\ \mathbf{0}, \Sigma_{\boldsymbol{u}}) \tag{6.11}$$

Now, one can define the priors for the augmented matrices H and $\Sigma_{\boldsymbol{u}}$.

6. The elements of H do not have any specific constraints. Also, they appear in a Gaussian likelihood for which a Gaussian prior serves as a conjugate prior. Hence, a multivariate Gaussian distribution is assumed as a prior for each row of H. The prior for a row H_i' is represented as $\mathcal{N}(H_i;\ \mu_H^{(i)}, \Sigma_H^{(i)})$.

7. Since the covariance matrix $\Sigma_{\boldsymbol{u}}$ is diagonal, a prior for each element can be defined separately. It one looks at the estimation of variance in terms of precision $\tau_{\boldsymbol{u}}^{(i)}$ defined as the inverse of the variance, i.e., $\tau_{\boldsymbol{u}}^{(i)} = 1/(\sigma_{\boldsymbol{u}}^{(i)})^2$, then it is known that conjugate prior for a Gaussian likelihood is a gamma distribution. Hence, the prior distribution for the precision is taken as a gamma distribution represented as $\Gamma(\tau_{\boldsymbol{u}}^{(i)};\ \alpha_{\boldsymbol{u}}^{(i)}, \beta_{\boldsymbol{u}}^{(i)})$.

One can write the joint distribution of all the states and parameters to be estimated as follows.

$$\begin{aligned} p(S, A, C, b, \sigma_e, \boldsymbol{Y}, H, \Sigma_{\boldsymbol{u}}) =& p(\boldsymbol{Y}|H, C, S, \Sigma_{\boldsymbol{u}}) \cdot p(C|b, \sigma_e) \cdot p(S|A) \\ & \cdot p(\Sigma_u) \cdot p(H) \cdot p(b) \cdot p(\sigma_e) \cdot p(A) \end{aligned} \tag{6.12}$$

Here, $\boldsymbol{Y} = \{\boldsymbol{y}_t\}$, $S = \{s_t\}$, and $C = \{c_t\}$.

Incorporating all the aforementioned distributions, the following result is obtained.

$$\begin{aligned} & p(S, A, C, b, \sigma_e, \boldsymbol{Y}, H, \Sigma_{\boldsymbol{u}}) \\ & \quad = \prod_{t=1}^{T} \mathcal{N}\left(\boldsymbol{y}_t;\ H \begin{bmatrix} c_t \\ s_t \end{bmatrix}, \Sigma_{\boldsymbol{u}}\right) \cdot \prod_{t=2}^{T} \mathcal{C}(c_t;\ bc_{t-1}, \sigma_e)\ \mathcal{C}(c_1; \mu_{c_1}, \sigma_{c_1}) \\ & \quad \cdot \prod_{t=2}^{T} \mathcal{N}(s_t;\ As_{t-1}, I - AA')\ \mathcal{N}(s_1;\ \mu_{s_1}, \Sigma_{s_1}) \\ & \quad \cdot \prod_{i=1}^{n_y+n_x} \Gamma(\tau_{\boldsymbol{u}}^{(i)};\ \alpha_{\boldsymbol{u}}^{(i)}, \beta_{\boldsymbol{u}}^{(i)}) \cdot \prod_{i=1}^{n_y+n_x} \mathcal{N}(H_i;\ \mu_H^{(i)}, \Sigma_H^{(i)}) \cdot \prod_{i=1}^{n_s} \mathcal{B}(a_i;\ \alpha_a^{(i)}, \beta_a^{(i)}) \\ & \quad \cdot \mathcal{B}(b;\ \alpha_b, \beta_b) \cdot \Gamma(\sigma_e;\ \alpha_e, \beta_e) \end{aligned} \tag{6.13}$$

The probabilistic graphical representation of the model is depicted in Fig. 6.3. Here, the variables inside a circle represent the random variables whose posteriors need to be estimated. The quantities in the square boxes represent the observed data and prior distribution parameters which are fixed.

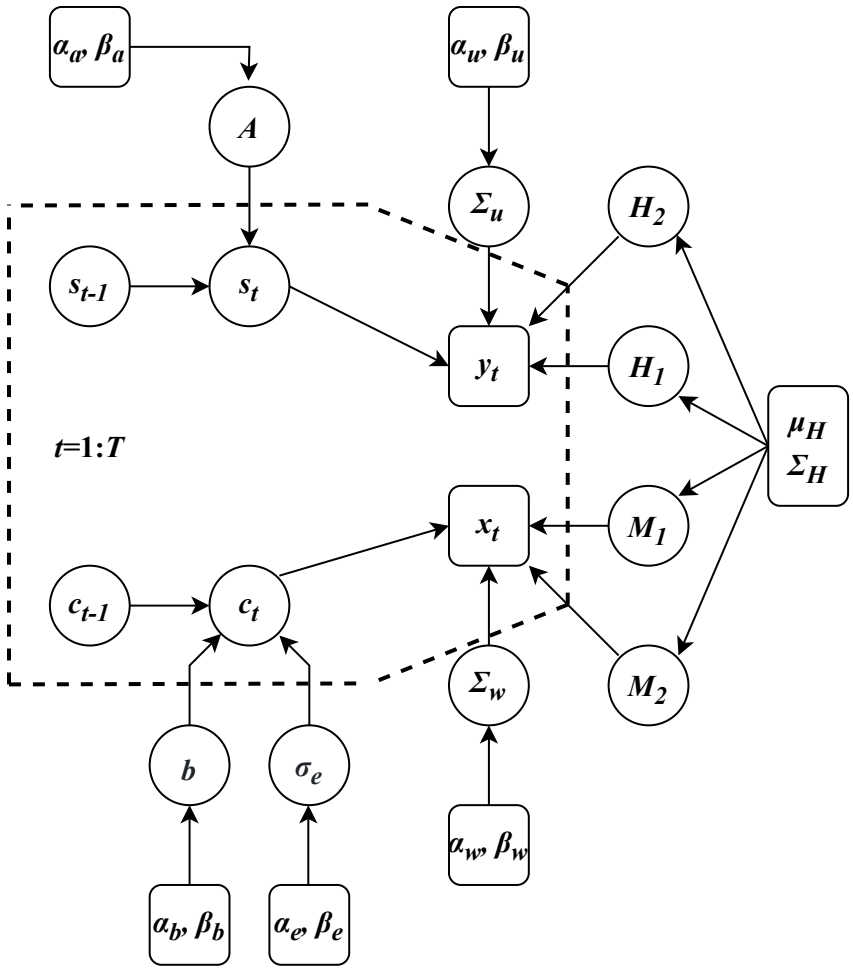


Figure 6.3: The hierarchical probabilistic graphical model of the proposed approach

6.3 Variational Bayesian inference of the model

The VB inference framework is used to estimate the posterior distributions of variables in LV models. The details of the VB method and the process of arriving at the expression for the posterior distribution in this approach are presented in section 2.3.2. The posterior distribution of a variable Z is expressed as follows.

$$q(Z) \propto \exp\left(\mathbb{E}_{\tilde{Z}\sim q(\tilde{Z})}\left[\ln\, p(Y, Z, \tilde{Z}|\theta_{pr})\right]\right) \tag{6.14}$$

Here, Y represents the observed data, $\tilde{Z}$ represents all the other latent variables of the model, and $q(\tilde{Z})$ is the distribution of all other LVs at the current iteration. The distribution $p(Y, Z, \tilde{Z}|\theta_{pr})$ represents the total data likelihood which in this case is given in (6.13). Only the distributions that have Z in them need to be retained and

the rest become the normalization constants. In this section, the detailed derivation of inferring the posterior distribution is provided for all the involved variables.

6.3.1 Inference of H

H is a $(n_y + n_x) \times n_s + 1$ dimensional matrix, and since the inference is done for each row H_i', there are $(n_y + n_x)$ number of H_i' to be inferred. In the total data likelihood, the distributions where H appears are $p(\boldsymbol{Y}|H, C, S, \Sigma_{\boldsymbol{u}})$ and $p(H)$. Thus to infer H_i', the distributions are $p(\boldsymbol{Y_i}|H_i, C, S, \tau_{\mathbf{u}}^{(i)})$ and $p(H_i)$. Hence, the following expression is obtained for $q(H_i)$

$$q(H_i) \propto \exp\left(\langle \ln\, p(\boldsymbol{Y_i}|H_i, C, S, \tau_{\mathbf{u}}^{(i)}) p(H_i)\rangle\right) \tag{6.15}$$

Here, $\langle\cdot\rangle$ represents the expectation. Henceforth, the same notation is used for expectation. For convenience, explicit mentioning of the variable w.r.t which the expectation is taken has been left out. The expectation taken in each case is w.r.t all other variables except the one for which the posterior expression is being derived. Since both the pdfs in the above equation are Gaussian, the following result is obtained.

$$q(H_i) \propto \exp\left(\sum_{t=1}^{T} \left\langle -\frac{1}{2}\left(\boldsymbol{y}_t^{(i)} - H_i'\mathcal{S}_t\right)' \tau_{\boldsymbol{u}}^{(i)} \left(\boldsymbol{y}_t^{(i)} - H_i'\mathcal{S}_t\right) \right\rangle - \frac{1}{2}(H_i - \mu_H^{(i)})'(\Sigma_H^{(i)})^{-1}(H_i - \mu_H^{(i)})\right) \tag{6.16}$$

Note that the prior does not contain any other random variables and hence it comes out of the expectation. Here, $\mathcal{S}_t = \begin{bmatrix} c_t \\ s_t \end{bmatrix}$ represents the augmented states. The above equation has a summation of two quadratic terms of H_i. Hence, it can always be written as a single quadratic of H_i. The objective thus is to express the sum of two above quadratics of H_i as

$$\left(H_i - \hat{\mu}_H^{(i)}\right)' (\hat{\Sigma}_H^{(i)})^{-1} \left(H_i - \hat{\mu}_H^{(i)}\right) = H_i(\hat{\Sigma}_H^{(i)})^{-1}H_i' - 2H_i'(\hat{\Sigma}_H^{(i)})^{-1}\hat{\mu}_H^{(i)} + \hat{\mu}_H^{(i)\prime}(\hat{\Sigma}_H^{(i)})^{-1}\hat{\mu}_H^{(i)} \tag{6.17}$$

Here, $\hat{\mu}_H^{(i)}$ and $\hat{\Sigma}_H^{(i)}$ represent the mean and covariance matrix of the posterior $q(H_i)$ which is a Gaussian. The expressions for these can be obtaining by comparing the second-order and first-order coefficients of H_i in (6.16) and (6.17). The second-order

term of H_i in (6.16) is

$$\left\langle \sum_{t=1}^{T} \mathcal{S}_t' H_i \tau_{\boldsymbol{u}}^{(i)} H_i' \mathcal{S}_t \right\rangle + H_i'(\Sigma_H^{(i)})^{-1} H_i = H_i' \left\langle \sum_{t=1}^{T} \mathcal{S}_t \tau_{\boldsymbol{u}}^{(i)} \mathcal{S}_t' \right\rangle H_i + H_i'(\Sigma_H^{(i)})^{-1} H_i$$
$$= H_i' \left((\Sigma_H^{(i)})^{-1} + \sum_{t=1}^{T} \langle \mathcal{S}_t \mathcal{S}_t' \rangle \langle \tau_{\boldsymbol{u}}^{(i)} \rangle \right) H_i \quad (6.18)$$

The simplification of $\langle \mathcal{S}_t \tau_{\boldsymbol{u}}^{(i)} \mathcal{S}_t' \rangle = \langle \mathcal{S}_t \mathcal{S}_t' \rangle \langle \tau_{\boldsymbol{u}}^{(i)} \rangle$ can be done because of the mean field approximation. The new quadratic matrix in the above equation is the posterior variance of H_i. Similarly, the linear term in of H_i in (6.16) is

$$-2H_i'(\Sigma_H^{(i)})^{-1}\mu_H^{(i)} - 2H_i' \sum_{t=1}^{T} \langle \mathcal{S}_t \tau_{\boldsymbol{u}}^{(i)} \boldsymbol{y}_t^{(i)} \rangle = -2H_i \left((\Sigma_H^{(i)})^{-1}\mu_H^{(i)} + \langle \tau_{\boldsymbol{u}}^{(i)} \rangle \sum_{t=1}^{T} \boldsymbol{y}_t^{(i)} \langle \mathcal{S} \rangle \right) \quad (6.19)$$

From (6.18), (6.19), and (6.17), the following results are obtained. These parameters are the posterior mean and variance of H_i which follows a Gaussian distribution.

$$\hat{\Sigma}_H^{(i)} = \left\{ (\Sigma_H^{(i)})^{-1} + \langle \tau_{\boldsymbol{u}}^{(i)} \rangle \sum_{t=1}^{T} \langle \mathcal{S}_t \mathcal{S}_t' \rangle \right\}^{-1}$$
$$\hat{\mu}_H^{(i)} = \hat{\Sigma}_H^{(i)} \left\{ (\Sigma_H^{(i)})^{-1}\mu_H^{(i)} + \langle \tau_u^{(i)} \rangle \sum_{t=1}^{T} \boldsymbol{y}_t^{(i)} \langle \mathcal{S}_t \rangle \right\} \quad (6.20)$$

6.3.2 Inference of Σ_u

The inference of each element of $\Sigma_{\boldsymbol{u}}$ i.e., $\left(\sigma_{\boldsymbol{u}}^{(i)}\right)^2$ is done through the precision $\tau_{\boldsymbol{u}}^{(i)}$ for which a gamma distributed prior is assumed. From the total data likelihood in (6.13) and the VB inference equation in (6.14), one can write the following.

$$q(\tau_{\boldsymbol{u}}^{(i)}) \propto \exp\left(\sum_{t=1}^{T} \left\langle -\frac{1}{2} (\boldsymbol{y}_t^{(i)} - H_i' \mathcal{S}_t)' \tau_{\boldsymbol{u}}^{(i)} (\boldsymbol{y}_t^{(i)} - H_i' \mathcal{S}_t) \right\rangle \right) (\tau_{\boldsymbol{u}}^{(i)})^{T/2}$$
$$\times (\tau_{\boldsymbol{u}}^{(i)})^{\alpha_{\boldsymbol{u}}^{(i)} - 1} \exp\left(-\beta_{\boldsymbol{u}}^{(i)} \tau_{\boldsymbol{u}}^{(i)}\right) \quad (6.21)$$

The above expression has the form of a gamma distribution which can be observed by looking at the $\tau_{\boldsymbol{u}}^{(i)}$ and the exponential terms in the likelihood and prior.

$$\hat{\alpha}_u^{(i)} = \alpha_u^{(i)} + \frac{T}{2} \quad (6.22)$$

$$\hat{\beta}_u^{(i)} = \beta_u^{(i)} + \left(\sum_{t=1}^{T} \left\langle \frac{1}{2} (\boldsymbol{y}_t^{(i)} - H_i' \mathcal{S}_t)' (\boldsymbol{y}_t^{(i)} - H_i' \mathcal{S}_t) \right\rangle \right) \quad (6.23)$$

The expectation in the equation for the posterior β can be simplified as follows

$$\hat{\beta}_u^{(i)} = \beta_u^{(i)} + \frac{1}{2}\sum_{t=1}^{T}\Big((\boldsymbol{y}_t^{(i)})^2 - 2\boldsymbol{y}_t^{(i)}\langle H_i\rangle'\langle\mathcal{S}_t\rangle + \mathrm{tr}\{\langle\mathcal{S}_t\mathcal{S}_t'\rangle\cdot\langle H_i H_i'\rangle\}\Big) \tag{6.24}$$

6.3.3 Inference of A

For a_i, due to the constraint $0 < a_i < 1$, a beta distribution is used as a prior. Substituting the total data likelihood in (6.13) into the VB equation in (6.14), $q(a_i)$ can be expressed as

$$q(a_i) \propto \exp\Bigg(-\frac{T-1}{2}\ln(1-a_i^2) + \sum_{t=2}^{T}\frac{-\langle(s_t^{(i)})^2\rangle - a_i^2\langle(s_{t-1}^{(i)})^2\rangle + 2a_i\langle s_t^{(i)}s_{t-1}^{(i)}\rangle}{2(1-a_i^2)} + (\alpha_a^{(i)}-1)\ln a_i + (\beta_a^{(i)}-1)\ln(1-a_i)\Bigg) \tag{6.25}$$

The above equation does not follow any standard distribution, and hence a closed-form expression for the pdf and its moments is not possible. Hence, importance sampling is used to estimate the pdf and its moments. Importance sampling is discussed in detail in section (2.4.1). In this case, the sampling distribution can be selected as the prior distribution which is $\mathcal{B}(a_i;\ \alpha_a^{(i)}, \beta_a^{(i)})$. The importance weights can now be expressed as

$$w_a^{(j)} \propto \exp\left(-\frac{T-1}{2}\ln(1-(a_i^{(j)})^2) + \sum_{t=2}^{T}\frac{-\langle(s_t^{(i)})^2\rangle - (a_i^{(j)})^2\langle(s_{t-1}^{(i)})^2\rangle + 2a_i^{(j)}\langle s_t^{(i)}s_{t-1}^{(i)}\rangle}{2(1-(a_i^{(j)})^2)}\right)$$
$$a_i^{(j)} \sim \mathcal{B}(a_i;\ \alpha_a^{(i)}, \beta_a^{(i)}),\ j = 1,\ldots,N \tag{6.26}$$

The above weights represent the approximate distribution of a_i and can be used as discussed in section (2.4.1) to estimate the moments of the distribution.

6.3.4 Inference of b

Similar to a_i, b is also constrained to be between 0 and 1. Hence a beta distribution $\mathcal{B}(b;\ \alpha_b, \beta_b)$ can be used as a prior for b as well. The Cauchy distribution for $p(c_t|c_{t-1}, b, \sigma_e)$ is expressed as follows.

$$p(c_t|c_{t-1}, b, \sigma_e) = \frac{\sigma_e}{\pi(\sigma_e^2 + (c_t - c_{t-1})^2)} \tag{6.27}$$

Substituting the above distribution and the prior for b into (6.14), the following expression is obtained for the posterior.

$$q(b) \propto \exp\left(-\sum_{t=2}^{T}\langle \ln(\sigma_e^2 + (c_t - bc_{t-1})^2)\rangle + (\alpha_b - 1)\ln\, b + (\beta_b - 1)\ln\,(1-b)\right) \tag{6.28}$$

Given the form of the Cauchy pdf, the above expression has the logarithm term inside the expectation. This makes the inference complicated as evaluating the expectation of logarithms is difficult. If a Monte-Carlo approach is followed, a very high computational cost is incurred. The Monte-Carlo approach for evaluating $\langle \ln(\sigma_e^2+(c_t-bc_{t-1})^2)\rangle$ would result in the following expression.

$$\langle \ln(\sigma_e^2 + (c_t - bc_{t-1})^2)\rangle \approx \sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} \ln\,(\sigma_e^{(j)^2} + (c_t^{(k)} - bx_{t-1}^{(l)})) \tag{6.29}$$

which is expensive to evaluate. Hence, a simplification based on the Taylor series expansion of the function is used [155, 156].

Say, there is a nonlinear function $f(X)$ whose expectation is to be evaluated. Here, X is the random variable and the moments of X are known. One can do a Taylor series approximation of the function around $\langle X\rangle$ which results in the following equations.

$$\begin{aligned}\langle f(X)\rangle &\approx \left\langle f(\langle X\rangle) + f_X(\langle X\rangle)\,(X - \langle X\rangle) + \frac{f_{XX}(\langle X\rangle)}{2!}(X-\langle X\rangle)^2\right\rangle \\ &= f(\langle X\rangle) + \frac{f_{XX}(\langle X\rangle)}{2}\mathrm{Var}[X]\end{aligned} \tag{6.30}$$

Here, $f_X(\cdot)$ and $f_{XX}(\cdot)$ represent the first and second derivative of $f(X)$ w.r.t X respectively. The above expression simplifies $\langle f(X)\rangle$ in terms of functions of $\langle X\rangle$ and its moments which can be evaluated easily. For a multivariate function, a similar approach is followed to arrive at the following result.

$$\begin{aligned}\langle f(X,Y)\rangle \approx f(\langle X\rangle,\langle Y\rangle) &+ \frac{f_{XX}(\langle X\rangle,\langle Y\rangle)}{2}\mathrm{Var}[X] + \frac{f_{YY}(\langle X\rangle,\langle Y\rangle)}{2}\mathrm{Var}[Y] \\ &+ f_{XY}(\langle X\rangle,\langle Y\rangle))\mathrm{Cov}[X,Y]\end{aligned} \tag{6.31}$$

The above expression is now applied to simplify $\langle \ln(\sigma_e^2+(c_t-bc_{t-1})^2)\rangle$. The objective here is to infer b, meaning to obtain an expression as a function of b. Hence, for this

Taylor series approximation, b is treated as a constant and the series approximation is done around $\langle\sigma_e\rangle$, $\langle c_t\rangle$, and $\langle c_{t-1}\rangle$.

$$\begin{aligned}\langle \ln(\sigma_e^2 + (c_t - bc_{t-1})^2)\rangle = \\ &\ln(\langle\sigma_e\rangle^2 + (\langle c_t\rangle - b\langle c_{t-1}\rangle)^2) + \frac{f_{\sigma_e,\sigma_e}(\langle\sigma_e\rangle, \langle c_t\rangle, \langle c_{t-1}\rangle)}{2}\ \mathrm{Var}[\sigma_e] \\ &+ \frac{f_{c_t,c_t}(\langle\sigma_e\rangle, \langle c_t\rangle, \langle c_{t-1}\rangle)}{2}\ \mathrm{Var}[c_t] + \frac{f_{c_{t-1},c_{t-1}}(\langle\sigma_e\rangle, \langle c_t\rangle, \langle c_{t-1}\rangle)}{2}\ \mathrm{Var}[c_{t-1}] \\ &+ f_{\sigma_e,c_t}(\langle\sigma_e\rangle, \langle c_t\rangle, \langle c_{t-1}\rangle)\ \mathrm{Cov}[\sigma_e, c_t] + f_{c_t,c_{t-1}}(\langle\sigma_e\rangle, \langle c_t\rangle, \langle c_{t-1}\rangle)\ \mathrm{Cov}[c_t, c_{t-1}] \\ &+ f_{c_{t-1},\sigma_e}(\langle\sigma_e\rangle, \langle c_t\rangle, \langle c_{t-1}\rangle)\ \mathrm{Cov}[c_{t-1}, \sigma_e] \end{aligned} \tag{6.32}$$

In the above expression, $\mathrm{Cov}[c_{t-1}, \sigma_e] = 0$ and $\mathrm{Cov}[\sigma_e, c_t] = 0$ due to the mean field approximation. The remaining second derivatives can be expressed in a simplified manner according to the following equations.

$$\begin{aligned} F_b(b) &= \frac{\langle\sigma_e\rangle^2 - (\langle c_t\rangle - b\langle c_{t-1}\rangle)^2}{(\langle\sigma_e\rangle^2 + (\langle c_t\rangle - b\langle c_{t-1}\rangle)^2)^2} \\ \frac{1}{2}f_{\sigma_e,\sigma_e} = -F_b(b);\quad \frac{1}{2}f_{c_t,c_t} &= F_b(b);\quad \frac{1}{2}f_{c_{t-1},c_{t-1}} = b^2F_b(b);\quad f_{c_t,c_{t-1}} = -2bF_b(b) \end{aligned} \tag{6.33}$$

The second derivatives in the above equation can be substituted in (6.32), which in turn can be substituted in (6.28). Finally, the resulting approximated expression for $q(b)$ would be in terms of moments of σ_e, c_t, and c_{t-1}, rather than the more complicated form of the expectation of a logarithm function. The final expression for $q(b)$ does not have a closed-form expression and hence the importance sampling method is used. The final expression for the weights of the distribution representing $q(b)$ can be expressed as

$$\begin{aligned} w_b^{(j)} \propto \exp\Bigg(-\sum_{t=2}^{T}\big\{&\ln(\langle\sigma_e\rangle^2 + (\langle c_t\rangle - b^{(j)}\langle c_{t-1}\rangle)^2) \\ &+F_b(b^{(j)})(\text{-}\mathrm{Var}[\sigma_e] + \mathrm{Var}[c_t] + (b^{(j)})^2\mathrm{Var}[c_{t-1}] - 2b^{(j)}\mathrm{Cov}[c_t, c_{t-1}])\big\}\Bigg) \\ b^{(j)} \sim \mathcal{B}(b;\ \alpha_b, \beta_b),\ j = 1, \ldots, N \end{aligned} \tag{6.34}$$

6.3.5 Inference of σ_e

The inference procedure for σ_e runs into the similar issue seen in the inference of b due to the Cauchy distribution. First, since σ_e is constrained to be positive, the

gamma distribution $\Gamma(\sigma_e;\ \alpha_e, \beta_e)$ is used as the prior. Applying (6.14) for inferring $q(\sigma_e)$, the following result is obtained.

$$q(\sigma_e) \propto \exp\left((T-1)\ \ln\ \sigma_e - \sum_{t=2}^{T}\langle \ln(\sigma_e^2 + (c_t - bc_{t-1})^2)\rangle + (\alpha_e - 1)\ln\ \sigma_e - \beta_e\sigma_e\right) \tag{6.35}$$

Similar to the previous case, the above expression has a logarithm inside an expectation. The Taylor series approximation is applied to this case as well, the difference being that now σ_e is taken as a constant, and the expansion is done around b, c_t, and c_{t-1}.

$$\begin{aligned}
&\langle \ln(\sigma_e^2 + (c_t - bc_{t-1})^2)\rangle = \\
&\qquad \ln(\sigma_e^2 + (\langle c_t\rangle - \langle b\rangle\langle c_{t-1}\rangle)^2) + \frac{f_{b,b}(\langle b\rangle, \langle c_t\rangle, \langle c_{t-1}\rangle)}{2}\ \mathrm{Var}[b] \\
&\qquad + \frac{f_{c_t,c_t}(\langle b\rangle, \langle c_t\rangle, \langle c_{t-1}\rangle)}{2}\ \mathrm{Var}[c_t] + \frac{f_{c_{t-1},c_{t-1}}(\langle b\rangle, \langle c_t\rangle, \langle c_{t-1}\rangle)}{2}\ \mathrm{Var}[c_{t-1}] \\
&\qquad + f_{b,c_t}(\langle b\rangle, \langle c_t\rangle, \langle c_{t-1}\rangle)\ \mathrm{Cov}[b, c_t] + f_{c_t,c_{t-1}}(\langle b\rangle, \langle c_t\rangle, \langle c_{t-1}\rangle)\ \mathrm{Cov}[c_t, c_{t-1}] \\
&\qquad + f_{c_{t-1},b}(\langle b\rangle, \langle c_t\rangle, \langle c_{t-1}\rangle)\ \mathrm{Cov}[c_{t-1}, b]
\end{aligned} \tag{6.36}$$

Due to the mean field approximation, $\mathrm{Cov}[c_{t-1}, b] = 0$ and $\mathrm{Cov}[b, c_t] = 0$. The remaining second order derivatives can be expressed as

$$\begin{aligned}
F_{\sigma_e}(\sigma_e) &= \frac{\sigma_e^2 - (\langle c_t\rangle - \langle b\rangle\langle c_{t-1}\rangle)^2}{(\langle\sigma_e\rangle^2 + (\langle c_t\rangle - \langle b\rangle\langle c_{t-1}\rangle)^2)^2} \\
\frac{1}{2}f_{b,b} = \langle c_{t-1}\rangle^2 F_{\sigma_e}(\sigma_e);&\quad \frac{1}{2}f_{c_t,c_t} = F_{\sigma_e}(\sigma_e); \\
\frac{1}{2}f_{c_{t-1},c_{t-1}} = \langle b\rangle^2 F_{\sigma_e}(\sigma_e);&\quad f_{c_t,c_{t-1}} = -2\langle b\rangle F_{\sigma_e}(\sigma_e)
\end{aligned} \tag{6.37}$$

The partial derivatives in the above equation can be substituted into (6.36) which in turn can be substituted into (6.35). The final expression for $q(\sigma_e)$ does not have a closed-form expression and can be approximated using the following importance weights.

$$\begin{aligned}
w_{\sigma_e}^{(j)} \propto \exp\Bigg((T-1)\ \ln\ \sigma_e^{(j)} - \sum_{t=2}^{T}\Big\{&\ln\left((\sigma_e^{(j)})^2 + (\langle c_t\rangle - \langle b\rangle\langle c_{t-1}\rangle)^2\right) \\
&+F_{\sigma_e}({\sigma_e}^{(j)})(\langle c_{t-1}\rangle^2\mathrm{Var}[b] + \mathrm{Var}[c_t] + \langle b\rangle^2\mathrm{Var}[c_{t-1}] - 2\langle b\rangle\mathrm{Cov}[c_t, c_{t-1}])\Big\}\Bigg) \\
\sigma_e^{(j)} \sim \Gamma(\sigma_e;\ \alpha_e, \beta_e),&\ \ j = 1, \ldots, N
\end{aligned} \tag{6.38}$$

6.3.6 Inference of $\mathcal{S}_t$

Similar to the previous cases, the inference of the states $\mathcal{S}_{1:T}$ is done by writing (6.14) w.r.t $\mathcal{S}_{1:T}$. This results in the following expression.

$$\begin{aligned} q(S) \propto \exp\Bigg(& -\sum_{t=1}^{T} \frac{1}{2}\langle(\boldsymbol{y}_t - H\mathcal{S}_t)'\Sigma_{\boldsymbol{u}}^{-1}(\boldsymbol{y}_t - H\mathcal{S}_t)\rangle \\ & -\sum_{t=2}^{T} \frac{1}{2}\langle(s_t - As_{t-1})'(I - AA')^{-1}(s_t - As_{t-1})\rangle \\ & -\sum_{t=2}^{T}\langle\ln(\sigma_e^2 + (c_t - bc_{t-1})^2)\rangle \\ & -\frac{1}{2}(s_1 - m_{s_1})'(s_1 - m_{s_1}) - \ln(\sigma_{c_1}^2 + (c_t - \mu_{c_1})^2)\Bigg) \end{aligned} \tag{6.39}$$

If one observes the above equation without $\langle\cdot\rangle$, this is the same as the objective of the smoothing procedure of state estimation. Because of $\langle\cdot\rangle$, one cannot do the state estimation directly. Similar to the previous cases where $\langle f(X)\rangle$ was written in terms of $f(\langle X\rangle)$ and so on, the same procedure is applied here as well. This is known as the mean and fluctuation decomposition in literature [157].

The idea of mean and fluctuation decomposition is essentially similar to the one discussed previously in the Taylor series approximations, i.e., bridging the gap between $\langle f(X)\rangle$ and $f(\langle X\rangle)$. If $f(\cdot)$ is a quadratic function, this gap can be exactly filled. For other nonlinear functions, this can only be approximated as one usually restricts only up to the second-order term of the Taylor series. Note that the expectation in this step is w.r.t all the parameters and hence it would be required to express (6.39) in terms of $\langle H\rangle$, $\langle A\rangle$, $\langle\tau_{\boldsymbol{u}}\rangle$, $\langle b\rangle$, and $\langle\sigma_e\rangle$. In that case, the filtering and smoothing algorithms can be applied directly.

For the measurement model terms, the gap can be filled as follows.

$$\begin{aligned} \langle(\boldsymbol{y}_t - H\mathcal{S}_t)'\Sigma_{\boldsymbol{u}}^{-1}(\boldsymbol{y}_t - H\mathcal{S}_t)\rangle = & (\boldsymbol{y}_t - \langle H\rangle\mathcal{S}_t)'\langle\Sigma_{\boldsymbol{u}}^{-1}\rangle(\boldsymbol{y}_t - \langle H\rangle\mathcal{S}_t) \\ & + \underbrace{\mathcal{S}_t'\langle H'\Sigma_{\boldsymbol{u}}^{-1}H\rangle\mathcal{S}_t - \mathcal{S}_t'\langle H\rangle'\langle\Sigma_{\boldsymbol{u}}^{-1}\rangle\langle H\rangle\mathcal{S}_t}_{\mathcal{S}_t'F_B\mathcal{S}_t} \end{aligned} \tag{6.40}$$

Note that $\Sigma_{\boldsymbol{u}}^{-1}$ is nothing but the diagonal matrix whose diagonal element is $\tau_{\boldsymbol{u}}^{(i)}$. Hence, $\langle\Sigma_{\boldsymbol{u}}^{-1}\rangle$ is nothing but a diagonal matrix of $\langle\tau_{\boldsymbol{u}}^{(i)}\rangle$ elements. For the SFs, the

gap can be filled as follows.

$$\begin{aligned}\langle (s_t - As_{t-1})'(I-AA')^{-1}(s_t - As_{t-1})\rangle \\ = (s_t - \langle A\rangle s_{t-1})'\langle (I-AA')^{-1}\rangle (s_t - \langle A\rangle s_{t-1}) \\ + \underbrace{\mathcal{S}'_{t-1}M'\langle A'(I-AA')^{-1}A\rangle M\mathcal{S}_{t-1} - \mathcal{S}'_{t-1}M'\langle A\rangle'\langle (I-AA')^{-1}\rangle\langle A\rangle M\mathcal{S}_{t-1}}_{\mathcal{S}'_{t-1}F_A\mathcal{S}_{t-1}}\end{aligned} \tag{6.41}$$

where, $M = \begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}$ such that $M\mathcal{S}_t = s_t$. The two "gap filling" or fluctuation terms in (6.40) and (6.41) are expressed as

$$\begin{aligned}F_B &= \langle H'\Sigma_u^{-1}H\rangle - \langle H\rangle'\langle\Sigma_u^{-1}\rangle\langle H\rangle \\ F_A &= M'\langle A'(I-AA')^{-1}A\rangle M - M'\langle A\rangle'\langle (I-AA')^{-1}\rangle\langle A\rangle M\end{aligned} \tag{6.42}$$

These two fluctuation terms can be compensated together as follows.

$$\begin{aligned}(\boldsymbol{y}_t - \langle H\rangle\mathcal{S}_t)'\langle\Sigma_{\boldsymbol{u}}^{-1}\rangle(\boldsymbol{y}_t - \langle H\rangle\mathcal{S}_t) + \mathcal{S}'_tF_B\mathcal{S}_t + \mathcal{S}'_tF_A\mathcal{S}_t \\ = (\tilde{\boldsymbol{y}}_t - \langle\tilde{H}\rangle\mathcal{S}_t)'\langle\tilde{\Sigma}_{\boldsymbol{u}}^{-1}\rangle(\tilde{\boldsymbol{y}}_t - \langle\tilde{H}\rangle\mathcal{S}_t)\end{aligned} \tag{6.43}$$

where,

$$\tilde{\boldsymbol{y}}_t = \begin{bmatrix}\boldsymbol{y}_t \\ \mathbf{0} \\ \mathbf{0}\end{bmatrix}; \tilde{H} = \begin{bmatrix}\langle H\rangle \\ U_A \\ U_B\end{bmatrix}; \tilde{\Sigma}_{\boldsymbol{u}}^{-1}\begin{bmatrix}\langle\Sigma_{\boldsymbol{u}}^{-1}\rangle & \mathbf{0} & \mathbf{0} \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I}\end{bmatrix} \tag{6.44}$$

with $U'_AU_A = F_A$ and $U'_BU_B = F_B$.

Now the remaining term to be expressed in terms of mean and fluctuation decomposition is the Cauchy distribution term $\sum_{t=2}^{T}\langle\ln(\sigma_e^2 + (c_t - bc_{t-1})^2)\rangle$. The same issue of logarithm inside an expectation appears here too. The Taylor series approximation can again be used which results in the following equation.

$$\begin{aligned}\langle\ln(\sigma_e^2 + (c_t - bc_{t-1})^2)\rangle &= \ln(\langle\sigma_e\rangle^2 + (c_t - \langle b\rangle c_{t-1})^2) \\ &\quad + \frac{f_{\sigma_e,\sigma_e}(\langle b\rangle, \langle\sigma_e\rangle)}{2}\ \mathrm{Var}[\sigma_e] + \frac{f_{b,b}(\langle b\rangle, \langle\sigma_e\rangle)}{2}\ \mathrm{Var}[b] \\ &\quad + f_{\sigma_e,b}(\langle b\rangle, \langle\sigma_e\rangle)\ \mathrm{Cov}[\sigma_e, b] \\ &= F(c_t|c_{t-1})\end{aligned} \tag{6.45}$$

$\mathrm{Cov}[\sigma_e, b] = 0$ because of the mean field approximation. The second derivatives can be expressed as follows.

$$\frac{1}{2}f_{\sigma_e,\sigma_e} = \frac{(c_t - \langle b\rangle c_{t-1})^2 - \langle\sigma_e\rangle^2}{(\langle\sigma_e\rangle^2 + (c_t - \langle b\rangle c_{t-1})^2)^2};\quad \frac{1}{2}f_{b,b} = c_{t-1}^2\frac{\langle\sigma_e\rangle^2 - (c_t - \langle b\rangle c_{t-1})^2}{(\langle\sigma_e\rangle^2 + (c_t - \langle b\rangle c_{t-1})^2)^2} \tag{6.46}$$

The above equations can be substituted in (6.45) to get a function of c_t and c_{t-1} for $\langle \ln(\sigma_e^2 + (c_t - bc_{t-1})^2) \rangle$. Let this function be represented as $F(c_t|c_t - 1)$ as given in (6.45). Now a pseudo transition distribution can be defined for c_t (only for filtering purposes) that accounts for the fluctuation term. The new distributions is given as

$$\tilde{p}(c_t|c_{t-1}) = \exp(-F(c_t|c_{t-1})) \tag{6.47}$$

where $F(c_t|c_{t-1})$ is given by equations (6.45) and (6.46). It can be noted that if b and σ_e are taken as deterministic quantities, $tildep(c_t|c_{t-1})$ becomes the original Cauchy distribution.

Finally, (6.39) can be reinterpreted as a filtering and smoothing problem with the following pdfs.

$$p(\tilde{\boldsymbol{y}}_t|\mathcal{S}_t) = \mathcal{N}(\tilde{\boldsymbol{y}}_t;\ \tilde{H}\mathcal{S}_t, \tilde{\Sigma}_{\boldsymbol{u}}) \tag{6.48}$$

$$p(s_t|s_{t-1}) = \mathcal{N}(s_t;\ \langle A \rangle s_{t-1}, \langle I - AA' \rangle) \tag{6.49}$$

$$\tilde{p}(c_t|c_{t-1}) \propto \exp(-F(c_t|c_{t-1})) \tag{6.50}$$

These pseudo distributions bridge the gap between $\langle \ln(\cdot) \rangle$ and $\ln(\langle \cdot \rangle)$ for the estimation of the states $\mathcal{S}$.

Since $p(c_t|c_{t-1})$ is a complicated distribution, the analytical derivation of the filtering and smoothing steps is not possible and hence particle filtering and smoothing methods are used. The algorithms are discussed in detail in section 2.4.2. In this case, the marginal particle filtering and smoothing algorithms are used. These are depicted in Algorithm. 6.1. The sampling distribution for the marginal particle filtering algorithm is a mixture distribution. Each $q(\mathcal{S}_t|\bar{\mathcal{S}}_{t-1}^{(j)}, \tilde{\boldsymbol{y}}_t)$ is taken as a Gaussian distribution calculated according to the Kalman filter equations. Let

$$\tilde{A} = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & A \end{bmatrix};\ \ \tilde{\Sigma}_v = \begin{bmatrix} \sigma_e & \mathbf{0} \\ \mathbf{0} & I - AA' \end{bmatrix} \tag{6.51}$$

Now, for each $\bar{\mathcal{S}}_{t-1}^{(j)}$ Kalman filter update equations are used which assume that c_t evolves according to a Gaussian. Hence,

$$\begin{aligned} \mu_t^{(j)} &= \tilde{A}\bar{\mathcal{S}}_{t-1}^{(j)} + K_t^{(j)}(\tilde{\boldsymbol{y}}_t - H\tilde{A}\bar{\mathcal{S}}_{t-1}^{(j)}) \\ P_t^{(j)} &= (I - K_t^{(j)}H)(\tilde{A}\hat{\Sigma}_{t-1}\tilde{A}' + \tilde{\Sigma}_v) \end{aligned} \tag{6.52}$$

Algorithm 6.1 Marginal particle filtering and particle smoothing for the proposed approach

Filtering

At $t = 1$

1. Sample $\mathcal{S}_1^{(i)} \sim q(\mathcal{S}_1|\tilde{\boldsymbol{y}}_1)$
2. Compute the weights and normalize them.

$$w(\mathcal{S}_1^{(i)}) = \frac{p(\tilde{\boldsymbol{y}}_1|\mathcal{S}_1^{(i)})\ p(s_1)\ p(c_1)}{q(\mathcal{S}_1^{(i)}|\tilde{\boldsymbol{y}}_1)}; \quad \tilde{w}(\mathcal{S}_1^{(i)}) \propto w(\mathcal{S}_1^{(i)})$$

3. Resample (if required) $\{\tilde{w}(\mathcal{S}_1^{(i)}), \mathcal{S}_1^{(i)}\}$ to get $\{1/N, \bar{\mathcal{S}}_1^{(i)}\}$

For $t = 2$ to T

1. Sample $\mathcal{S}_t^{(i)} \sim \sum_{j=1}^{N} \tilde{w}_{t-1}^{(j)}\ q(\mathcal{S}_t|\bar{\mathcal{S}}_{t-1}^{(j)}, \tilde{\boldsymbol{y}}_t)$
2. Compute the weights and normalize them.

$$w(\mathcal{S}_t^{(i)}) = \frac{p(\tilde{\boldsymbol{y}}_t|\mathcal{S}_t^{(i)})\ \sum_{j=1}^{N} \tilde{w}_{t-1}^{(j)}\ p(s_t^{(i)}|\bar{s}_{t-1}^{(j)})\tilde{p}(c_t^{(i)}|\bar{c}_{t-1}^{(j)})}{\sum_{j=1}^{N} \tilde{w}_{t-1}^{(j)}\ q(\mathcal{S}_t^{(i)}|\bar{\mathcal{S}}_{t-1}^{(j)}, y_t)}; \quad \tilde{w}(\mathcal{S}_t^{(i)}) \propto w(\mathcal{S}_t^{(i)})$$

3. Resample (if required) $\{\tilde{w}(\mathcal{S}_t^{(i)}), \mathcal{S}_t^{(i)}\}$ to get $\{1/N, \bar{\mathcal{S}}_t^{(i)}\}$

Smoothing

For $t = T - 1$ to 1

$$w(\mathcal{S}_{t|T}^{(i)}) = w(\mathcal{S}_t^{(i)}) \left[\sum_{j=1}^{N} w(\mathcal{S}_{t+1|T}^{(j)}) \frac{p(s_{t+1}^{(j)}|s_t^{(i)})\tilde{p}(c_{t+1}^{(j)}|c_t^{(i)})}{\sum_{k=1}^{N} w(\mathcal{S}_t^{(k)})\ p(s_{t+1}^{(j)}|s_t^{(k)})\tilde{p}(c_{t+1}^{(j)}|c_t^{(k)})}\right]$$

Here, $K_t^{(j)} = P_t^{(j-)} H'(H P_t^{(j-)} H' + \Sigma_{\boldsymbol{u}})$, with $P_t^{(j-)} = (\tilde{A}\hat{\Sigma}_{t-1}\tilde{A}' + \tilde{\Sigma}_v)$. The quantity $\hat{\Sigma}_{t-1}$ is the covariance of all the particles in the previous step $\{\tilde{w}(\mathcal{S}_{t-1}^{(i)}), \mathcal{S}_{t-1}^{(i)}\}$. Hence, the sampling distribution is a mixture Gaussian distribution.

The filtering step is followed by the smoothing step which uses the weights and the particles of the filtering step to obtain the new weights that represent the smoothed distribution. The smoothing step for this system is also shown in Algorithm. 6.1.

6.3.7 The iterative procedure

The VB inference scheme is an iterative procedure that minimizes the evidence lower bound (ELBO) as discussed in 2.3.2. The ELBO is expressed as follows.

$$\mathcal{L}(q(S,\theta)) = \int q(S,\theta) \ln \big(p(Y,S,\theta|\theta_{pr})\big) dS \; d\theta - \int q(S,\theta) \ln \big(q(S,\theta)\big) dS \; d\theta \quad (6.53)$$

This iterative procedure for the system under consideration is summarized in the following points.

1. Set the hyper-parameters corresponding to the parameters of the model A, b, σ_e, H, $\Sigma_{\boldsymbol{u}}$. These are the quantities that appear in the squared brackets outside the dotted boundary in Fig. 6.3.

2. Estimate $q(\mathcal{S}_{1:T})$ as described in Algorithm. 6.1.

3. Estimate H according to (6.20).

4. Estimate $\Sigma_{\boldsymbol{u}}$ according to (6.22) and (6.24).

5. Estimate A according to (6.26).

6. Estimate b according to (6.34).

7. Estimate σ_e according to (6.38).

8. Iterate steps 2 to 7 till the convergence of ELBO given in (6.53).

The overall scheme can be computationally expensive as this requires the implementation of the particle smoothing algorithm in each iteration. But since this iterative procedure is performed offline, such computational burden may be tolerated.

6.3.8 Online implementation

During the online implementation of the model for the soft sensor application considered here, the parameters learned during the iterative procedure of the training phase are used and only the estimation of the states is conducted. thus, only the particle filtering algorithm will have to be performed with the original pdfs given below.

$$p(s_t|s_{t-1}, A) = \mathcal{N}(s_t;\ As_{t-1}, I - AA') \tag{6.54}$$

$$p(c_t|c_{t-1}, b, \sigma_e) = \mathcal{C}(c_t;\ bc_{t-1}, \sigma_e) \tag{6.55}$$

$$p(x_t|c_t, s_t, M_1.M_2, \Sigma_w) = \mathcal{N}(x_t;\ M_1c_t + M_2s_t, \Sigma_w) \tag{6.56}$$

This hence is not very computationally expensive for online implementation as only one step of particle filtering needs to be performed before the next x_t is available. It must be noted that y_t will not be available during the online implementation and only x_t is available. Hence, the estimation of a state is only based on x_t and the distribution learned is $p(\mathcal{S}_t|x_{:t})$ with $t > T$. With these estimated states, y_t can be predicted according to the following equation

$$\hat{y}_t = H_1\hat{c}_t + H_2\hat{s}_t \tag{6.57}$$

In the iterative procedure discussed in the previous section, it was mentioned that the convergence of the procedure can be decided based on the ELBO value. Alternatively, for the soft sensor applications case, the convergence can also be based on the prediction performance of the model on a validation dataset (different from the training dataset used for learning the states and the parameters). Since the objective is to predict y_t, this prediction error-based stopping criterion can be useful as a regularization for the model learning. Additionally, the performance on the validation dataset may also serve as an indicator in selecting the number of slow latent variables.

6.4 Results

Two case studies are presented in this work to illustrate the effectiveness of the proposed approach. The first is a numerical example where the data is generated by a linear model with the characteristics that are considered in this work. The second

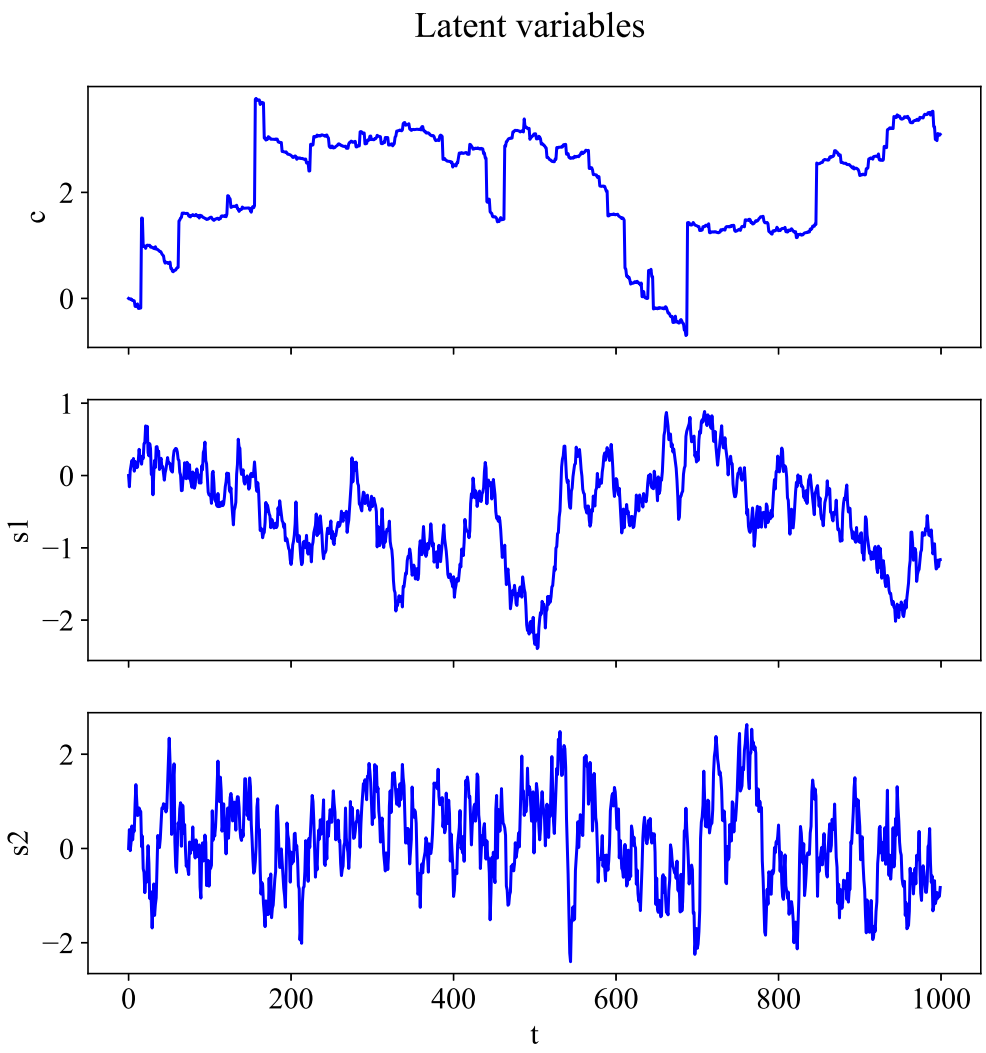


Figure 6.4: Generated latent variables c_t and s_t for the simulated case study

is an industrial case study obtained from a SAGD process. In both cases, the model developed through the proposed approach is used as a soft sensor.

6.4.1 Simulated case study

The objective of this work is to model the abrupt jumps and the other slow variations of the process. These two sources are latent in the observed data and hence are modeled by two dynamic models, characterized by a Cauchy distribution and Gaussian distribution respectively. Hence, the data corresponding to these two sources is

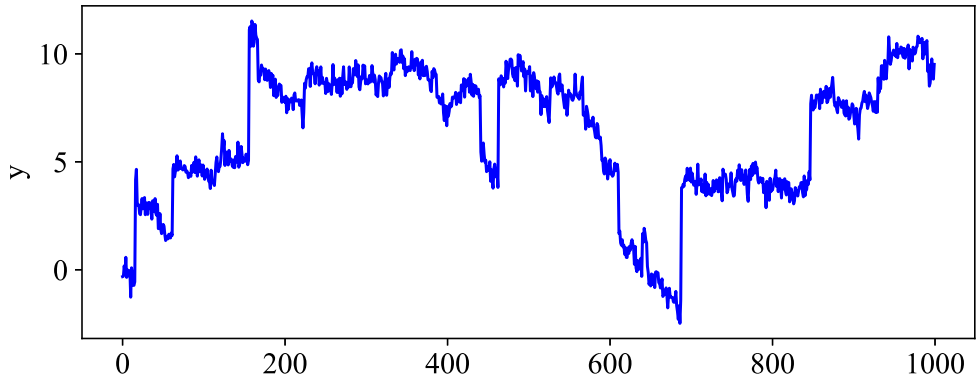


Figure 6.5: Generated output data y_t for the simulated case study

generated first. The following equations are used to generate the two LVs, c_t and s_t.

$$s_t = \begin{bmatrix} 0.99 & 0 \\ 0 & 0.9 \end{bmatrix} s_{t-1} + v_t, \quad v_t \sim \mathcal{N}\left(v_t;\ \mathbf{0}, \begin{bmatrix} 1-0.99^2 & 0 \\ 0 & 1-0.9^2 \end{bmatrix}\right) \tag{6.58}$$

$$c_t = c_{t-1} + e_t, \qquad e_t \sim \mathcal{C}(e_t;\ \mathbf{0}, 0.01) \tag{6.59}$$

A total of 1000 points are generated according to the above dynamic equations. The generated two slow features and one impulsive feature are used to produce the observed input and output data x_t and y_t respectively. Five input variables and one output variable are considered.

$$x_t = \begin{bmatrix} 1.25 \\ 0.5 \\ -0.5 \\ 0.25 \\ -0.5 \end{bmatrix} c_t + \begin{bmatrix} -1 & 2 \\ 0.5 & 1 \\ 2 & -0.5 \\ -1 & 1 \\ 0.5 & -1 \end{bmatrix} s_t + w_t, \quad w_t \sim \mathcal{N}(w_t;\ \mathbf{0}, \mathrm{diag}[0.2, 0.1, 0.05, 0.15, 0.25]) \tag{6.60}$$

$$y_t = 3\ c_t + \begin{bmatrix} 0.1 & -0.025 \end{bmatrix} s_t + u_t, \qquad u_t \sim \mathcal{N}(u_t;\ \mathbf{0}, 0.1) \tag{6.61}$$

It can be noted that in the output equation for y_t, more emphasis is given to c_t than s_t. This choice is to mimic the case where the impulsive behavior is more predominantly observed in the output and to a lesser extent in the inputs. Such a case is more difficult to handle by regular regression type of algorithms and is where the proposed approach is needed. If the abrupt jumps are predominant in x_t as well, regular linear methods may also capture such jumps in the predicted y_t. Thus the matrices H_1, H_2, M_1, and M_2 are selected such that the notions underlying the proposed approach

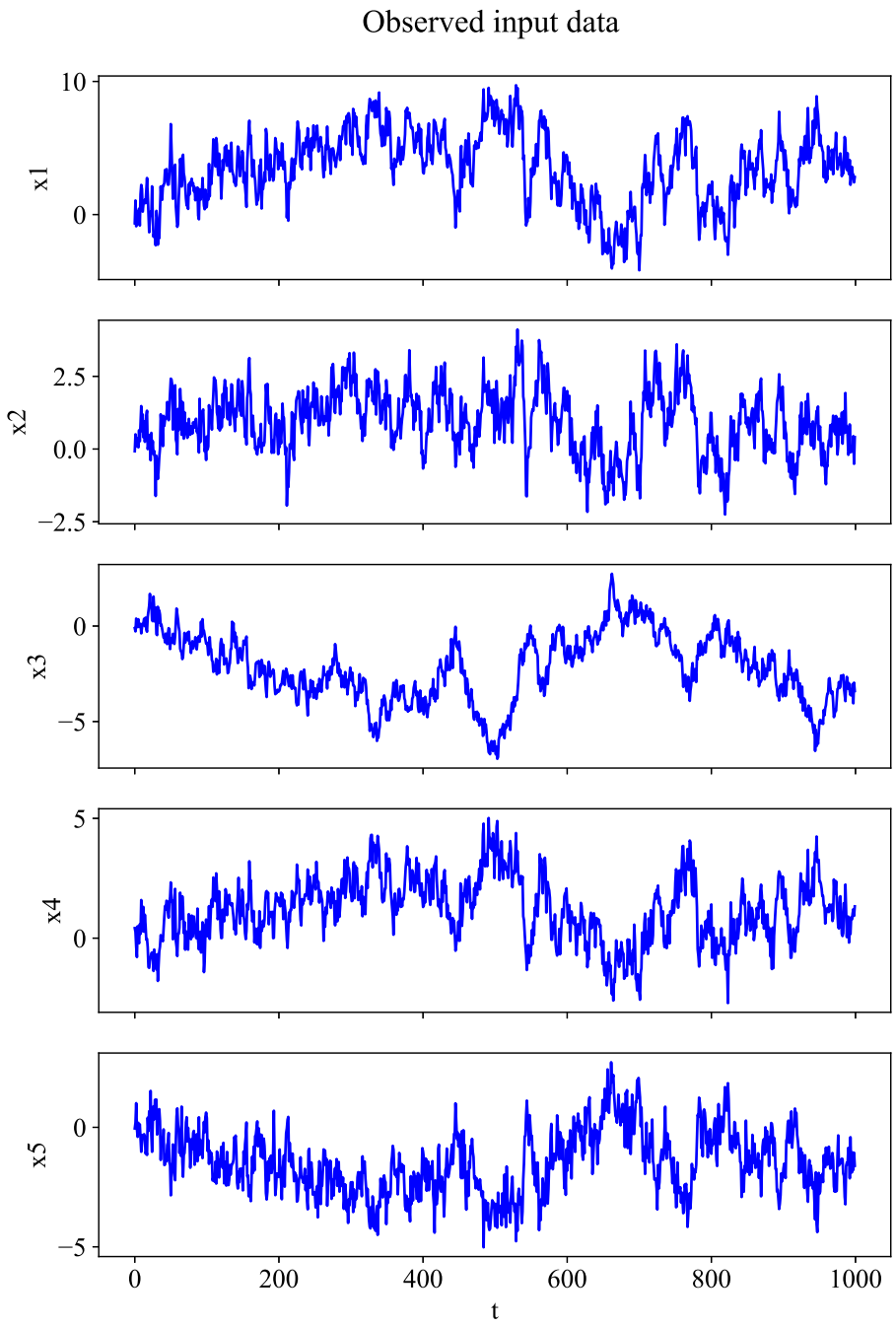


Figure 6.6: Generated input dataset x_t for the simulated case study. The abrupt jumps are not predominantly noticed in x_t.

are appropriately incorporated into the generated more challenging scenario which necessitates the proposed approach. The generated latent variables, output data and

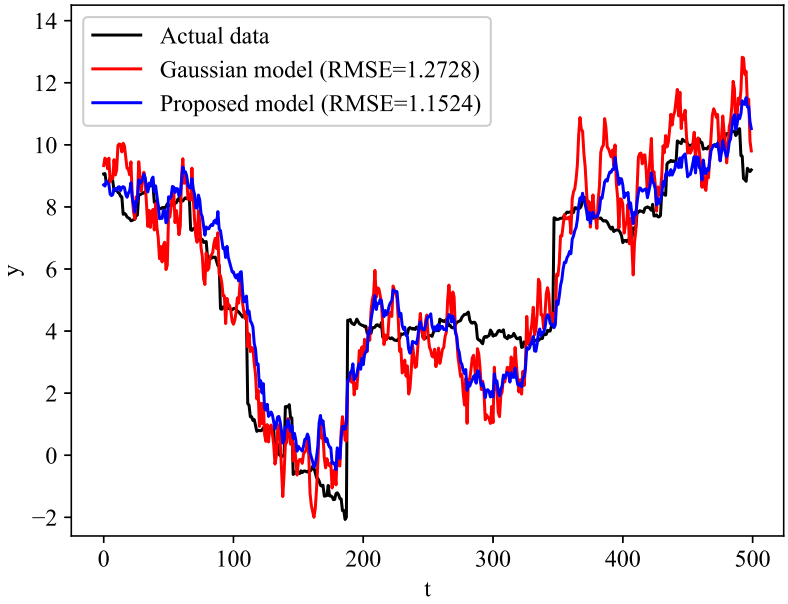
the input data are shown in Fig. 6.4, 6.5, and 6.6 respectively.

The generated dataset contains 1000 points of which the first 500 are used for training and the rest are used for testing. The proposed method is implemented on the generated dataset to learn the model parameters and is used to predict y_t of the test data. The obtained results from the proposed approach are compared with those obtained from PCR, PLS, SFR, and a probabilistic model exactly the same as that of the proposed model shown in (6.1) to (6.4), with the only difference that c_t in (6.2) is modeled as a Gaussian. This last model is similar to the approach used in Ma and Huang [65]. The obtained results are compared in terms of root mean squared error (RMSE) and R^2, and are shown in Table. 6.1. It can be observed that the deterministic approaches perform poorly in comparison with the probabilistic approaches. This is expected because the probabilistic approach handles the noise well. Among the proposed method and the method with a full Gaussian distribution including for c_t (named Gaussian model in Table. 6.1), the proposed method has an RMSE 9.46 % less than the Gaussian model.

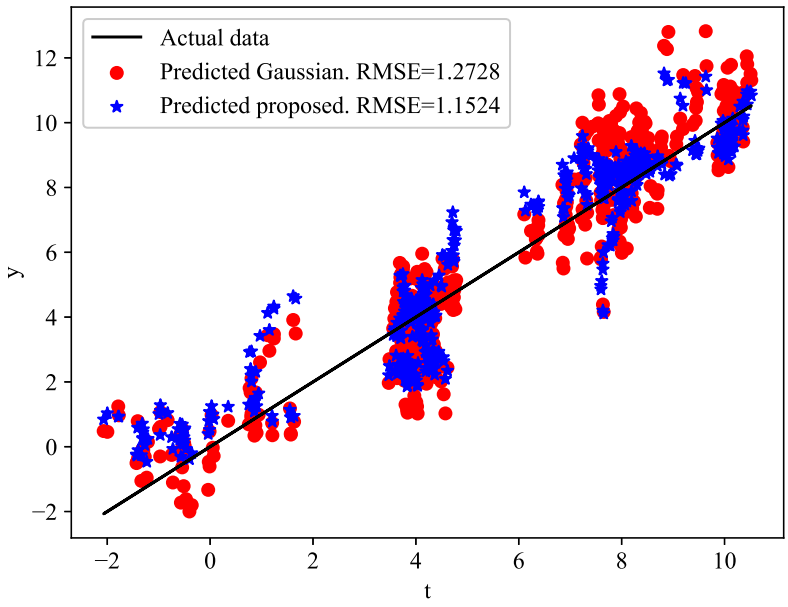
Table 6.1: Comparing the values of RMSE and R^2 obtained from the Gaussian model and the proposed model for the simulated case study

Method	RMSE	R^2
PCR	2.1686	0.5472
PLS	3.0832	0.0847
SFR	2.1650	0.5487
Gaussian model	1.2728	0.8440
Proposed model	1.1524	0.8721

Fig. 6.7 depicts the performance of the Gaussian model and the proposed model. Since the deterministic methods perform significantly poorly, the proposed method is only compared with the next best performer for a clutter-free representation. It can be observed from the signal plot in Fig. 6.7a, that the Gaussian model has more variations than the proposed model which degrades the performance of the Gaussian model. The reasoning for this behavior can be explained by the scaling parameter of the transition model for c_t, which is σ_e. In the training phase where all the parameters including σ_e are learned, both the models observe this abrupt change in y_t. Since the



(a) Comparison of the signals of predicted y_t and the actual one



(b) Scatter plot of the predicted y_t vs. actual y_t

Figure 6.7: Comparison of the performance of the Gaussian model and the proposed approach for the simulated case study. The Gaussian model has a high variance for c_t which degrades its performance.

Gaussian distribution has thinner tails than the Cauchy distribution, in order to obtain these jumps, it forces a higher variance for c_t. The proposed model does not face this issue because the Cauchy distribution is able to obtain such jumps which does not force a high σ_e. The obtained σ_e for the Gaussian model and the proposed Cauchy model are 0.1069 and 0.0107 respectively. Note that the σ_e value taken to generate the data is 0.01. It is this high value of σ_e which magnifies the smaller variations of c_t in the case of a Gaussian model that results in a poor performance in the regions where the process does not go through abrupt changes. To put it simply, in trying to model the abrupt jumps, the Gaussian model fails in regions where abrupt jumps are not seen. The Cauchy model has no such issue as it has heavy tails that allow it to accommodate both impulsive and stable behaviors.

As far as the overall performance of the Cauchy model is concerned, even when the does not have the kind of abrupt changes (black curve in Fig. 6.7a), it still performs better than the Gaussian model because of the correct estimation of σ_e. The reason for the absence of the sudden jumps for the Cauchy model is attributed to the fact that during the testing phase the states are estimated only based on x_t in which such abrupt jumps are not as apparently observed.

6.4.2 Industrial case study on a SAGD process

The second case study is that of a soft sensor development for predicting the emulsion flow in a SAGD process [158]. The SAGD process is a widely used process for recovering bitumen from oil sands. In this process, two horizontal wells are drilled into the oil reservoir, one above the other. Steam is injected into the upper well which heats the reservoir. This reduces the viscosity of the oil which flows into the bottom well due to gravity. The resulting oil and water emulsion is then pumped to the surface. The emulsion flow-rate obtained from a particular well is one of the key variables whose online measurements are required for control and optimization applications. This measurement will not be available in most cases because the emulsion flow from multiple adjacent wells is combined in a well pad to make a combined flow. Hence a soft sensor is needed to estimate the emulsion flow-rate from an individual well. For this process 16 key variables related to the pump and the reservoir conditions such as temperature, pressure, etc are available. They are used to develop a soft sensor

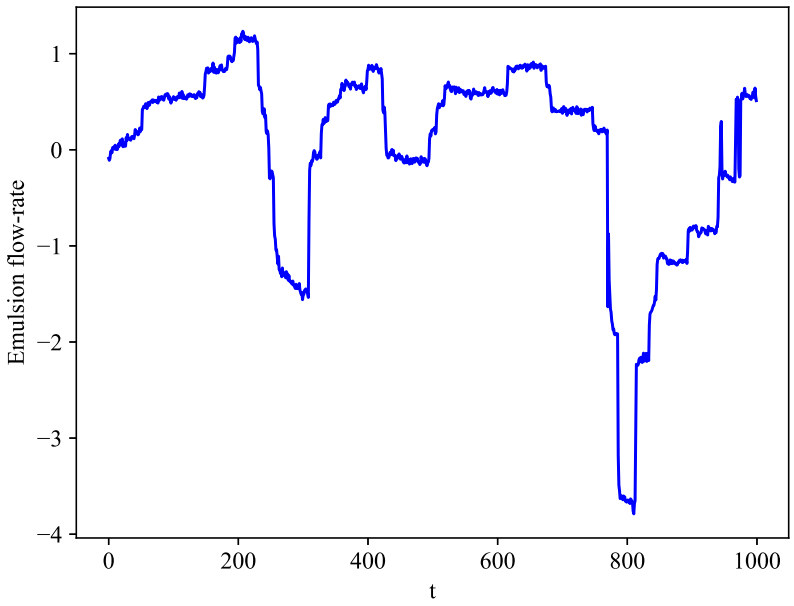


Figure 6.8: Emulsion flow-rate obtained from the SAGD process. The process is characterized by abrupt changes.

to estimate the real-time emulsion flow-rate. For soft sensor development, sparsely available measurements of the emulsion flow-rate are used which are obtained when the flow is directed to a test separator. The overall process goes through many abrupt changes as observed by the emulsion flow-rate measurement shown in Fig. 6.8.

The overall available data is appropriately down-sampled resulting in a set of 1000 data points. Similar to the previous case, half of this is used for training, and the other half for testing. To complicate the scenario of the abrupt jumps, additional synthetic Gaussian white noise is added to the data shown in Fig. 6.8. The proposed method is compared with PCR, PLS, SFR, and the Gaussian model, and the results are shown in Table. 6.2 and Fig. 6.9.

Overall, all the methods perform well on this dataset as observed by the R^2 values. The proposed method still has shown some improvement over the Gaussian model. But if one observes the performance section-wise, the advantages of the proposed approach become more apparent. If one draws attention to the first 270 points of the data in Table. 6.2, it can be observed that the proposed method performs better.

Table 6.2: Comparing the values of RMSE and R^2 obtained from the Gaussian model and the proposed model for the SAGD process dataset

Method	RMSE	R^2
PCR	0.2322	0.9623
PLS	0.2455	0.9578
SFR	0.2319	0.9624
Gaussian model	0.2230	0.9652
Proposed model	0.1916	0.9743

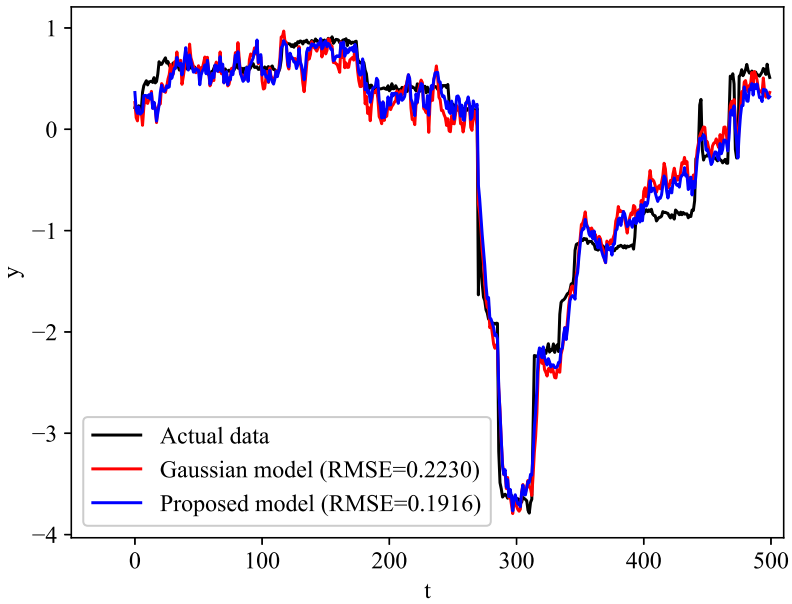
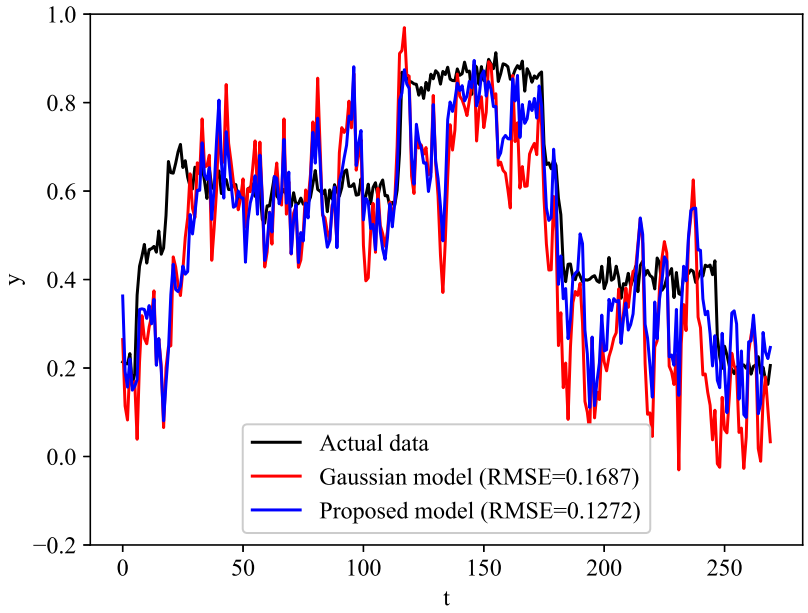


Figure 6.9: Comparing the predicted emulsion flow-rate obtained from the Gaussian and the proposed model

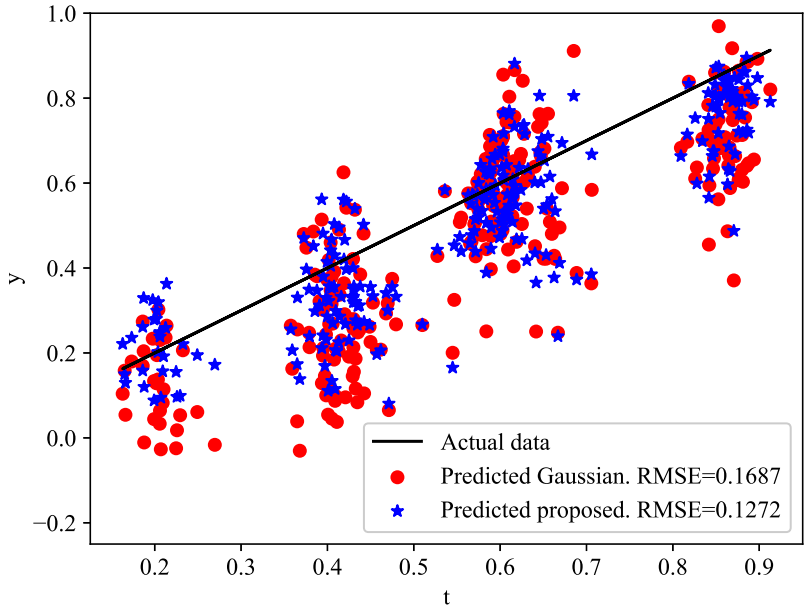
Table 6.3: Comparing the values of RMSE and R^2 obtained from the Gaussian model and the proposed model for the SAGD process dataset in the shorter time range

Method	RMSE	R^2
Gaussian model	0.1687	0.3091
Proposed model	0.1272	0.6072

This can be seen in Table. 6.3 Fig. 6.10. These are the results for the first 270 points where the Gaussian model sees more variation than the proposed Cauchy model.



(a) Comparison of the signals of predicted emulsion flow-rate and the actual one



(b) Scatter plot of the predicted emulsion flow-rate vs. actual y_t

Figure 6.10: Comparison of the performance of the Gaussian model and the proposed approach for the SAGD process dataset in the shorter time range

In this portion, the RMSE reduction is 24.6%, and the R^2 almost doubles. The R^2 of 0.3091 for the Gaussian model implies that the Gaussian model almost has no predictive ability. While R^2 of 0.6072 implies a significant predictive ability for the proposed Cauchy model. This is because of the high variation observed in the Gaussian model, the reasoning for which is similar to the previous case where it was explained that the Gaussian model imposes a higher σ_e on c_t than necessary. Hence, even though there is a high computational burden during the offline modeling step of the proposed approach, such complexity may be tolerated due to the enhanced accuracy, particularly as seen in the regions where the process is relatively stable.

6.5 Conclusion

This work presents a method of modeling and estimating systems whose signals exhibit abrupt changes. Modeling of such changes is done in the latent space of the data by considering a dynamic model whose evolution is defined by the Cauchy distribution. Meanwhile, the normal slower variations of the process are modeled according to the PSFA model. The model parameters are estimated in a Bayesian framework through the VB inference framework. The Cauchy distribution-based model, owing to its heavier tails of the process noise, is able to capture the jumps more accurately than its Gaussian counterpart. This results in a more accurate estimation of the process noise scale parameter through the Cauchy model. In the case of the Gaussian model, the thinner tails force the model to have a very high variance in order to capture the jumps. This degrades the performance of the model in the region where the process does not show abrupt changes. The two case studies presented here point to the efficacy of the Cauchy model where a performance improvement is observed particularly in the stable regions of the process due to the more accurate estimation of the variances.

Chapter 7

Concluding Remarks

This chapter outlines the conclusions drawn from the proposed methods and their implementations in various case studies. Additionally, potential future directions of research in this area are also outlined.

7.1 Conclusion

The overall underlining principle explored in this thesis is that of process-relevant latent variable modeling. Hence, all the proposed methods focus on certain properties of the LVs which are based on certain prior knowledge one may have regarding the process. The three properties explored here are slowness, monotonicity, and impulsivity. Each of these qualities is commonly observed in various industrial processes and are latent in the observed data, thus necessitating algorithms to extract them.

The first contribution presented in Chapter 3 focuses on the slowness aspect. The proposed approach aims at obtaining slowly varying features for supervised learning problems. In such cases, it is more appropriate to model the slowly varying LVs that are correlated with the outputs. To achieve this, the proposed approach combined SFA with PLS to impart temporal slowness and output correlation properties to the LVs. From the case studies, it is observed that the proposed method not only reduces the RMSE of prediction in comparison with methods such as PCR, PLS, etc, it also requires a fewer number of latent features to achieve it. This implies that the proposed approach is able to extract more meaningful LVs than other approaches. This is particularly observed in the second case study where the proposed algorithm needed only two LVs to describe the data while the conventional approaches needed

from 4 to as many as 12. Such an efficient representation of the latent space is possible due to concurrent considerations of slowness and output-relevance which is appropriate to processes driven by slowly varying LVs.

The second contribution builds on this idea of supervised learning of slow features by extending it to nonlinear systems. Complex industrial processes are usually nonlinear in nature and thus can be well modeled through nonlinear models. Deep learning is a powerful tool capable of obtaining rich representation which can be used to achieve the aforementioned objective. Slowness implies dealing with velocities which needs handling of two adjacent samples simultaneously. Hence, the Siamese neural networks are used. The temporal slowness and output-relevance aspects are imparted onto the LVs through modifications to the objective function of the Siamese networks. The proposed approach has been found to be more effective than feed-forward neural networks and other recent methods such as HELM and VW-SAE.

The third contribution presented in Chapter 5 focuses on the aspect of monotonicity which is an important character of degrading processes. Monotonicity is a special type of non-stationary character, and in degrading processes this aspect may not be apparent. The proposed method thus aims at modeling the latent sources of monotonic variation and other stationary variations of the observed dataset. A state space model is proposed to model such systems where the LMT evolves according to a CSN and the LSTs evolve according to a Gaussian distribution. The resulting estimation problem is solved through the EM algorithm which separates the state (E-step) and parameter (M-steps) estimation steps in an efficient manner. The E-step in particular is the challenging part of this approach as this requires a state estimation (filtering and smoothing) procedure for a system with CSN noises. A detailed derivation of these steps is provided in this work. Through the case studies, it is observed that the explicit modeling of the two latent sources of monotonicity and stationarity results in an efficient representation of the system. The consideration of the true nature of degradation results in accurate modeling of the stationary variations of the process. If the Gaussian distribution is used to model the non-stationarity, the lack of monotonicity constraints in such a model results in inaccurate modeling of the systems as it induces unnecessary variations in the extracted LST.

The fourth contribution presented in Chapter 6 considers the impulsivity aspect

of processes. This behavior is characterized by abrupt jumps in process variables observed during the operation. Such jumps which may be caused due to various reasons such as disturbances, capacity changes, etc need to be modeled explicitly for an accurate representation of the system. The proposed approach proceeds in a similar fashion as the third contribution where the two sources of variations of abrupt jumps and slower variations are modeled separately. These are modeled using Cauchy and Gaussian distribution respectively. The resulting model is estimated in a VB inference framework as this approach is more suitable when one has a certain modeling preference. The effectiveness of the proposed model is in terms of the accuracy of the estimated parameters, particularly the variance of the LV modeling the jumps. The proposed approach involves the Cauchy distribution which due to its heavy tails is able to generate the jumps. This results in more accurate learning of the variance of the process noise. On the other hand, if a Gaussian variable is to be used to model the jumps, the thinner tail forces it to have a higher variance so that the abrupt jumps may be modeled. This results in a degradation in the performance of the model in the regions where the abrupt jumps are absent. Hence, the proposed approach although computationally expensive due to the employment of the particle smoother algorithm (only offline though), is still desirable due to its enhanced accuracy.

In summary, each process has its own unique character and the LV modeling for such processes needs to consider its inherent character. Hence, instead of a pure data-based approach, infusion of the beliefs about the process into the data-based models through algorithmic modification such as the ones presented in this thesis is more advantageous.

7.2 Future scope

This section outlines the various possible avenues of research in terms of extensions and further improvements to the methods presented in thesis.

7.2.1 Slowness penalty for efficient feature representation

In Chapter 3, the proposed method resulted in an efficient feature extraction framework as the proposed approach needed fewer LVs than the conventional ones. This result is attributed to the right combination of the appropriate methods which in this case are PLS and SFA. Hence, the temporal penalty can be seen as an important aspect to consider while modeling slow processes. This notion was used in Chapter 3 for supervised learning application. But this notion can be extended to other applications of data-based modeling approaches involving slow processes. In particular, process monitoring applications can benefit from such approaches because a smaller number of representative features is more desirable for process monitoring. Hence, methods such as PCA, independent component analysis, etc can be designed in such a way that they have a temporal slowness penalty.

7.2.2 Extensions to the proposed monotonic feature extraction approach

The proposed approach in Chapter 4 results in a solution that can be computationally expensive because the dimensionality of the parameters of the skewness part of the CSN keeps on increasing. This makes the calculation of the moments difficult. It is a general problem encountered in CSN filtering problems. More research is thus needed in this regard. One approach could be to approximate the 2-dimensional CSN as a 1-dimensional one in every step of filtering. This would keep the dimensionality of the CSN constant thus controlling the dimensionality blow-up issue.

Another possible approach towards monotonic feature extraction would be the usage of the skew-t distribution to model the dynamics of the LMT. The skew-t distribution has some interesting properties and there are efficient filtering algorithms reported in the literature for systems with such a noise. The advantage of skew-t distribution comes from its heavier tails. In degrading processes, the degradation rate may see abrupt changes. The skew-t distribution can model such sudden jumps in degradation while the CSN cannot.

7.2.3 Distinguishing the outliers and abrupt process jumps

Often when there is an abrupt jump in the data (particularly of the type depicted in Fig. 6.2a), one is unsure as to whether the observed jump is caused by an actual change in the process condition or it is a measurement outlier. Thus it is important to make such distinctions particularly when the model is used under closed-loop conditions. The proposed method in Chapter 6 can be extended to such a case by incorporating the aspects of robust identification techniques. In this case, the measurement model can also be modeled using the Cauchy noise. The identified model in such a case will be able to predict if a deviation from normalcy is an outlier or an actual process change. In the case of an actual jump in the process, the latent feature c_t would also show a jump. If that is not the case, then the point must be deemed an outlier. This type of modeling not only enables such distinctions but also makes the obtained model robust to measurement outliers.

7.2.4 Process-relevant dynamic LV modeling through deep learning

Deep learning has been increasingly used in modeling industrial processes and its usage is only going to increase in the future. There are multiple architectures in deep learning that deal with the aspect of LVs. These include, but are not limited to autoencoders, variational autoencoders (VAE), recurrent neural networks and their variants such as long-short term memory (LSTM), convolutional neural networks, etc. In all these approaches, particularly for the applications in industrial processes, pure data-based training may not be effective. This might lead to model reliability issues. It would thus be important to perform process-relevant LV modeling through deep learning. In this regard, LSTM networks offer an attractive way of performing process-relevant LV modeling. LSTM (with a combination along with VAE) can be viewed as a nonlinear state space model. The PSFA model, and proposed approaches in chapters 5 and 6, all are state space models that incorporate beliefs about the process into them. Hence extending such LV models to nonlinear systems through LSTM networks is worth exploring.

References

[1] Z. Ge, Z. Song, S. X. Ding, and B. Huang, "Data Mining and Analytics in the Process Industry: The Role of Machine Learning," *IEEE Access*, vol. 5, pp. 20590–20616, 2017.

[2] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven Soft Sensors in the process industry," *Computers and Chemical Engineering*, vol. 33, no. 4, pp. 795–814, 2009.

[3] Z. Ge, Z. Song, and F. Gao, "Review of recent research on data-based process monitoring," *Industrial and Engineering Chemistry Research*, vol. 52, no. 10, pp. 3543–3562, 2013.

[4] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural computation*, vol. 14, no. 4, pp. 715–770, 2002.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[6] Q. Sun and Z. Ge, "A survey on deep learning for data-driven soft sensors," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 5853–5866, 2021.

[7] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature Verification using a "Siamese" Time Delay Neural Network," in *Advances in Neural Information Processing Systems 6* (J. D. Cowan, G. Tesauro, and J. Alspector, eds.), pp. 737–744, Morgan-Kaufmann, 1994.

[8] S. Khatibisepehr, B. Huang, and S. Khare, "Design of inferential sensors in the process industry: A review of bayesian methods," *Journal of Process Control*, vol. 23, no. 10, pp. 1575–1596, 2013.

[9] I. T. Jolliffe, *Principal Component Analysis*. Springer-Verlag New York, 2 ed., 2002.

[10] J. V. Kresta, J. F. Macgregor, and T. E. Marlin, "Multivariate statistical monitoring of process operating performance," *The Canadian Journal of Chemical Engineering*, vol. 69, no. 1, pp. 35–47, 1991.

[11] J. Yu, "Local and global principal component analysis for process monitoring," *Journal of Process Control*, vol. 22, no. 7, pp. 1358–1373, 2012.

[12] W. Ku, R. H. Storer, and C. Georgakis, "Disturbance detection and isolation by dynamic principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 30, no. 1, pp. 179–196, 1995.

[13] D. Kim and I. . Lee, "Process monitoring based on probabilistic pca," *Chemometrics and Intelligent Laboratory Systems*, vol. 67, no. 2, pp. 109–123, 2003.

[14] J. V. Kresta, T. E. Marlin, and J. F. MacGregor, "Development of inferential process models using pls," *Computers and Chemical Engineering*, vol. 18, no. 7, pp. 597–611, 1994.

[15] J. Liu, "Developing a soft sensor based on sparse partial least squares with variable selection," *Journal of Process Control*, vol. 24, no. 7, pp. 1046–1056, 2014.

[16] N. L. Ricker, "The use of biased least-squares estimators for parameters in discrete-time pulse-response models," *Industrial & Engineering Chemistry Research*, vol. 27, no. 2, pp. 343–350, 1988.

[17] S. Qin and T. McAvoy, "A data-based process modeling approach and its applications," *IFAC Proceedings Volumes*, vol. 25, no. 5, pp. 93 – 98, 1992. 3rd IFAC Symposium on Dynamics and Control of Chemical Reactors, Distillation Columns and Batch Processes (DYCORD+ '92), Maryland, USA, 26-29 April.

[18] M. H. Kaspar and W. H. Ray, "Dynamic pls modelling for process control," *Chemical Engineering Science*, vol. 48, no. 20, pp. 3447 – 3461, 1993.

[19] S. Lakshminarayanan, S. L. Shah, and K. Nandakumar, "Modeling and control of multivariable processes: Dynamic pls approach," *AIChE Journal*, vol. 43, no. 9, pp. 2307–2322, 1997.

[20] Y. Dong and S. J. Qin, "Regression on dynamic pls structures for supervised learning of dynamic data," *Journal of Process Control*, vol. 68, pp. 64–72, 2018.

[21] S. J. Qin, "Recursive pls algorithms for adaptive data modeling," *Computers and Chemical Engineering*, vol. 22, no. 4-5, pp. 503–514, 1998.

[22] W. Ni, S. D. Brown, and R. Man, "A localized adaptive soft sensor for dynamic system modeling," *Chemical Engineering Science*, vol. 111, pp. 350–363, 2014.

[23] H. Jin, X. Chen, L. Wang, K. Yang, and L. Wu, "Dual learning-based online ensemble regression approach for adaptive soft sensor modeling of nonlinear time-varying processes," *Chemometrics and Intelligent Laboratory Systems*, vol. 151, pp. 228–244, 2016.

[24] H. Kaneko and K. Funatsu, "Ensemble locally weighted partial least squares as a just-in-time modeling method," *AIChE Journal*, vol. 62, no. 3, pp. 717–725, 2016.

[25] M. Ma, S. Khatibisepehr, and B. Huang, "A bayesian framework for real-time identification of locally weighted partial least squares," *AIChE Journal*, vol. 61, no. 2, pp. 518–529, 2015.

[26] K. Hazama and M. Kano, "Covariance-based locally weighted partial least squares for high-performance adaptive modeling," *Chemometrics and Intelligent Laboratory Systems*, vol. 146, pp. 55–62, 2015.

[27] Z. Zhang and D. Tao, "Slow feature analysis for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 436–450, 2012.

[28] X. Hu, S. Hu, Y. Huang, H. Zhang, and H. Wu, "Video anomaly detection using deep incremental slow feature analysis network," *IET Computer Vision*, vol. 10, no. 4, pp. 258–265, 2016.

[29] K. Fan, P. Wang, and S. Zhuang, "Human fall detection using slow feature analysis," *Multimedia Tools and Applications*, vol. 78, no. 7, pp. 9101–9128, 2019.

[30] L. Sun, K. Jia, T. Chan, Y. Fang, G. Wang, and S. Yan, "Dl-sfa: Deeply-learned slow feature analysis for action recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2625–2632, 2014.

[31] C. Shang, B. Huang, F. Yang, and D. Huang, "Slow feature analysis for monitoring and diagnosis of control performance," *Journal of Process Control*, vol. 39, pp. 21–34, 2016.

[32] X. Gao, C. Shang, F. Yang, and D. Huang, "Detecting and isolating plant-wide oscillations via slow feature analysis," in *Proceedings of the American Control Conference*, vol. 2015-July, pp. 906–911, 2015.

[33] C. Shang, F. Yang, X. Gao, X. Huang, J. A. K. Suykens, and D. Huang, "Concurrent monitoring of operating condition deviations and process dynamics anomalies with slow feature analysis," *AIChE Journal*, vol. 61, no. 11, pp. 3666–3682, 2015.

[34] S. Zhang and C. Zhao, "Slow-feature-analysis-based batch process monitoring with comprehensive interpretation of operation condition deviation and dynamic anomaly," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 5, pp. 3773–3783, 2019.

[35] C. Shang, F. Yang, X. Gao, and D. Huang, "Extracting latent dynamics from process data for quality prediction and performance assessment via slow feature regression," in *Proceedings of the American Control Conference*, vol. 2015-July, pp. 912–917, 2015.

[36] X. Ma, Y. Si, Z. Yuan, Y. Qin, and Y. Wang, "Multistep dynamic slow feature analysis for industrial process monitoring," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 12, pp. 9535–9548, 2020.

[37] K. Zhong, D. Ma, and M. Han, "Distributed dynamic process monitoring based on dynamic slow feature analysis with minimal redundancy maximal relevance," *Control Engineering Practice*, vol. 104, 2020.

[38] C. Shang, F. Yang, B. Huang, and D. Huang, "Recursive slow feature analysis for adaptive monitoring of industrial processes," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 11, pp. 8895–8905, 2018.

[39] H. Hong, C. Jiang, X. Peng, and W. Zhong, "Concurrent monitoring strategy for static and dynamic deviations based on selective ensemble learning using slow feature analysis," *Industrial & Engineering Chemistry Research*, vol. 59, no. 10, pp. 4620–4635, 2020.

[40] X. Yuan, J. Zhou, and Y. Wang, "Locally weighted slow feature regression for nonlinear dynamic soft sensor modeling and its application to an industrial hydrocracking process," *Measurement Science and Technology*, vol. 31, no. 5, 2020.

[41] J. Wang and C. Zhao, "Variants of slow feature analysis framework for automatic detection and isolation of multiple oscillations in coupled control loops," *Computers and Chemical Engineering*, vol. 141, 2020.

[42] W. Yu and C. Zhao, "Recursive exponential slow feature analysis for fine-scale adaptive processes monitoring with comprehensive operation status identification," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3311–3323, 2019.

[43] Y. Song, S. Yang, C. Cheng, and P. Xie, "A novel fault detection method for running gear systems based on dynamic inner slow feature analysis," *IEEE Access*, vol. 8, pp. 211371–211379, 2020.

[44] Y. Xu, M. Jia, and Z. Mao, "A novel auto-regressive dynamic slow feature analysis method for dynamic chemical process monitoring," *Chemical Engineering Science*, vol. 248, 2022.

[45] C. Zhao and B. Huang, "A full-condition monitoring method for nonstationary dynamic chemical processes with cointegration and slow feature analysis," *AIChE Journal*, vol. 64, no. 5, pp. 1662–1681, 2018.

[46] C. Zhang, K. Peng, and J. Dong, "A nonlinear full condition process monitoring method for hot rolling process with dynamic characteristic," *ISA transactions*, vol. 112, pp. 363–372, 2021.

[47] C. Zhao, J. Chen, and H. Jing, "Condition-driven data analytics and monitoring for wide-range nonstationary and transient continuous processes," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 1563–1574, 2021.

[48] J. Zheng and C. Zhao, "Online monitoring of performance variations and process dynamic anomalies with performance-relevant full decomposition of slow feature analysis," *Journal of Process Control*, vol. 80, pp. 89–102, 2019.

[49] R. Raveendran, H. Kodamana, and B. Huang, "Process monitoring using a generalized probabilistic linear latent variable model," *Automatica*, vol. 96, pp. 73 – 83, 2018.

[50] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society.Series B: Statistical Methodology*, vol. 61, no. 3, pp. 611–622, 1999.

[51] J. Zheng, Z. Song, and Z. Ge, "Probabilistic learning of partial least squares regression model: Theory and industrial applications," *Chemometrics and Intelligent Laboratory Systems*, vol. 158, pp. 80–90, 2016.

[52] S. el Bouhaddani, H. . Uh, C. Hayward, G. Jongbloed, and J. Houwing-Duistermaat, "Probabilistic partial least squares model: Identifiability, estimation and application," *Journal of Multivariate Analysis*, vol. 167, pp. 331–346, 2018.

[53] R. Turner and M. Sahani, "A maximum-likelihood interpretation for slow feature analysis," *Neural computation*, vol. 19, no. 4, pp. 1022–1038, 2007.

[54] L. Zafeiriou, M. A. Nicolaou, S. Zafeiriou, S. Nikitidis, and M. Pantic, "Probabilistic slow features for behavior analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 5, pp. 1034–1048, 2016.

[55] C. Shang, B. Huang, F. Yang, and D. Huang, "Probabilistic slow feature analysis-based representation learning from massive process data for soft sensor modeling," *AIChE Journal*, vol. 61, no. 12, pp. 4126–4139, 2015.

[56] F. Guo, C. Shang, B. Huang, K. Wang, F. Yang, and D. Huang, "Monitoring of operating point and process dynamics via probabilistic slow feature analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 151, pp. 115–125, 2016.

[57] C. Shang, B. Huang, Y. Lu, F. Yang, and D. Huang, "Dynamic modeling of gross errors via probabilistic slow feature analysis applied to a mining slurry preparation process," in *IFAC-PapersOnLine*, vol. 49, pp. 25–30, 2016.

[58] L. Fan, H. Kodamana, and B. Huang, "Identification of robust probabilistic slow feature regression model for process data contaminated with outliers," *Chemometrics and Intelligent Laboratory Systems*, vol. 173, pp. 1–13, 2018.

[59] D. Scott, C. Shang, B. Huang, and D. Huang, "A holistic probabilistic framework for monitoring nonstationary dynamic industrial processes," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 5, pp. 2239–2246, 2021.

[60] V. K. Puli, R. Raveendran, and B. Huang, "Complex probabilistic slow feature extraction with applications in process data analytics," *Computers & Chemical Engineering*, vol. 154, p. 107456, 2021.

[61] L. Fan, H. Kodamana, and B. Huang, "Semi-supervised dynamic latent variable modeling: I/o probabilistic slow feature analysis approach," *AIChE Journal*, vol. 65, no. 3, pp. 964–979, 2019.

[62] C. Jiang, Y. Lu, W. Zhong, B. Huang, W. Song, D. Tan, and F. Qian, "Deep bayesian slow feature extraction with application to industrial inferential modeling," *IEEE Transactions on Industrial Informatics*, pp. 1–10, 2021.

[63] M. J. Beal, *Variational algorithms for approximate Bayesian inference*. University of London, University College London (United Kingdom), 2003.

[64] D. Ostwald, E. Kirilina, L. Starke, and F. Blankenburg, "A tutorial on variational bayes for latent linear stochastic time-series models," *Journal of mathematical psychology*, vol. 60, pp. 1–19, 2014.

[65] Y. Ma and B. Huang, "Bayesian learning for dynamic feature extraction with application in soft sensing," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 9, pp. 7171–7180, 2017.

[66] Y. Ma and B. Huang, "Extracting dynamic features with switching models for process data analytics and application in soft sensing," *AIChE Journal*, vol. 64, no. 6, pp. 2037–2051, 2018.

[67] J. Xie, B. Huang, and S. Dubljevic, "Transfer learning for dynamic feature extraction using variational bayesian inference," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021.

[68] R. E. Barlow and H. D. Brunk, "Isotonic regression problem and its dual," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 140–147, 1972.

[69] D. Gorinevsky, "Monotonic regression filters for trending deterioration faults," in *proceedings of the 2004 American Control Conference*, vol. 6, pp. 5394–5399, IEEE, 2004.

[70] S. Samar, D. Gorinevsky, and S. Boyd, "Moving horizon filter for monotonic trends," in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, vol. 3, pp. 3115–3120, IEEE, 2004.

[71] D. Gorinevsky, "Efficient filtering using monotonic walk model," in *2008 American Control Conference*, pp. 2816–2821, IEEE, 2008.

[72] J. M. van Noortwijk, "A survey of the application of gamma processes in maintenance," *Reliability Engineering & System Safety*, vol. 94, no. 1, pp. 2–21, 2009.

[73] A. Schirru, S. Pampuri, and G. De Nicolao, "Particle filtering of hidden gamma processes for robust predictive maintenance in semiconductor manufacturing," in *2010 IEEE International Conference on Automation Science and Engineering*, pp. 51–56, IEEE, 2010.

[74] G. A. Susto, A. Schirru, S. Pampuri, A. Beghi, and G. De Nicolao, "A hidden-gamma model-based filtering and prediction approach for monotonic health factors in manufacturing," *Control Engineering Practice*, vol. 74, pp. 84–94, 2018.

[75] M. Guida and G. Pulcini, "The inverse gamma process: A family of continuous stochastic models for describing state-dependent deterioration phenomena," *Reliability Engineering & System Safety*, vol. 120, pp. 72–79, 2013.

[76] X. Wang and D. Xu, "An inverse gaussian process model for degradation data," *Technometrics*, vol. 52, no. 2, pp. 188–197, 2010.

[77] A. Azzalini and A. Dalla Valle, "The multivariate skew-normal distribution," *Biometrika*, vol. 83, no. 4, pp. 715–726, 1996.

[78] J. B. Copas and H. G. Li, "Inference for non-random samples," *Journal of the Royal Statistical Society.Series B: Statistical Methodology*, vol. 59, no. 1, pp. 55–95, 1997.

[79] G. González-Farías, A. Domínguez-Molina, and A. K. Gupta, "Additive properties of skew normal random vectors," *Journal of Statistical Planning and Inference*, vol. 126, no. 2, pp. 521–534, 2004.

[80] P. Naveau, M. G. Genton, and X. Shen, "A skewed kalman filter," *Journal of multivariate Analysis*, vol. 94, no. 2, pp. 382–400, 2005.

[81] O. Karimi, H. Omre, and M. Mohammadzadeh, "Bayesian closed-skew gaussian inversion of seismic avo data for elastic material properties," *Geophysics*, vol. 75, no. 1, pp. R1–R11, 2010.

[82] J. Rezaie and J. Eidsvik, "Kalman filter variants in the closed skew normal setting," *Computational Statistics & Data Analysis*, vol. 75, pp. 1–14, 2014.

[83] J. Rezaie and J. Eidsvik, "A skewed unscented kalman filter," *International Journal of Control*, vol. 89, no. 12, pp. 2572–2583, 2016.

[84] R. B. Arellano-Valle, J. E. Contreras-Reyes, F. O. L. Quintero, and A. Valdebenito, "A skew-normal dynamic linear model and bayesian forecasting," *Computational Statistics*, vol. 34, no. 3, pp. 1055–1085, 2019.

[85] L. He, J. Chen, and Y. Qi, "Event-based state estimation: Optimal algorithm with generalized closed skew normal distribution," *IEEE Transactions on Automatic Control*, vol. 64, no. 1, pp. 321–328, 2019.

[86] H. Yu, J. Shang, and T. Chen, "On stochastic and deterministic event-based state estimation," *Automatica*, vol. 123, 2021.

[87] C. Peng and S. Tseng, "Statistical lifetime inference with skew-wiener linear degradation models," *IEEE Transactions on Reliability*, vol. 62, no. 2, pp. 338–350, 2013.

[88] Z. Huang, Z. Xu, X. Ke, W. Wang, and Y. Sun, "Remaining useful life prediction for an adaptive skew-wiener process model," *Mechanical Systems and Signal Processing*, vol. 87, pp. 294–306, 2017.

[89] M. A. Little and N. S. Jones, "Generalized methods and solvers for noise removal from piecewise constant signals. i. background theory," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 467, no. 2135, pp. 3088–3114, 2011.

[90] M. A. Little and N. S. Jones, "Generalized methods and solvers for noise removal from piecewise constant signals. ii. new methods," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 467, no. 2135, pp. 3115–3140, 2011.

[91] I. Selesnick, "Total variation denoising via the moreau envelope," *IEEE Signal Processing Letters*, vol. 24, no. 2, pp. 216–220, 2017.

[92] D. Lv, W. Cao, W. Hu, and M. Wu, *A New Total Variation Denoising Algorithm for Piecewise Constant Signals Based on Non-convex Penalty*, vol. 1449 of *Communications in Computer and Information Science*. 2021.

[93] M. J. Lombardi, "Bayesian inference for α-stable distributions: A random walk mcmc approach," *Computational Statistics and Data Analysis*, vol. 51, no. 5, pp. 2688–2700, 2007.

[94] M. Idan and J. L. Speyer, "Cauchy estimation for linear scalar systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 6, pp. 1329–1342, 2010.

[95] M. Idan and J. L. Speyer, "State estimation for linear scalar dynamic systems with additive cauchy noises: Characteristic function approach," *SIAM Journal on Control and Optimization*, vol. 50, no. 4, pp. 1971–1994, 2012.

[96] M. Idan and J. L. Speyer, "Multivariate cauchy estimator with scalar measurement and process noises," in *52nd IEEE Conference on Decision and Control*, pp. 5016–5023, 2013.

[97] M. Idan and J. L. Speyer, "An estimation approach for linear stochastic systems based on characteristic functions," *Automatica*, vol. 78, pp. 153–162, 2017.

[98] Y. Wang, Y. Qi, J. Zhu, J. Zhang, Y. Wang, G. Pan, X. Zheng, and Z. Wu, "A cauchy-based state-space model for seizure detection in eeg monitoring systems," *IEEE Intelligent Systems*, vol. 30, no. 1, pp. 6–12, 2014.

[99] J. H. Fernández, J. L. Speyer, and M. Idan, "Stochastic control for linear systems with additive cauchy noises," *IEEE Transactions on Automatic Control*, vol. 60, no. 12, pp. 3373–3378, 2015.

[100] R. Piché, S. Särkkä, and J. Hartikainen, "Recursive outlier-robust filtering and smoothing for nonlinear systems using the multivariate student-t distribution," in *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pp. 1–6, 2012.

[101] T. Liu and D. Tao, "On the robustness and generalization of cauchy regression," in *2014 4th IEEE International Conference on Information Science and Technology*, pp. 100–105, 2014.

[102] P. Geladi and B. R. Kowalski, "Partial least-squares regression: a tutorial," *Analytica Chimica Acta*, vol. 185, no. C, pp. 1–17, 1986.

[103] S. Wold, M. Sjöström, and L. Eriksson, "Pls-regression: A basic tool of chemometrics," *Chemometrics and Intelligent Laboratory Systems*, vol. 58, no. 2, pp. 109–130, 2001.

[104] S. de Jong, "Simpls: An alternative approach to partial least squares regression," *Chemometrics and Intelligent Laboratory Systems*, vol. 18, no. 3, pp. 251–263, 1993.

[105] A. Höskuldsson, "Pls regression methods," *Journal of Chemometrics*, vol. 2, no. 3, pp. 211–228, 1988.

[106] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[107] N. Sammaknejad, Y. Zhao, and B. Huang, "A review of the expectation maximization algorithm in data-driven process identification," *Journal of Process Control*, vol. 73, pp. 123–136, 2019.

[108] A. Tulsyan, R. Bhushan Gopaluni, and S. R. Khare, "Particle filtering without tears: A primer for beginners," *Computers and Chemical Engineering*, vol. 95, pp. 130–145, 2016.

[109] A. Doucet, A. M. Johansen, *et al.*, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of nonlinear filtering*, vol. 12, no. 656-704, p. 3, 2009.

[110] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[111] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[112] M. Klaas, N. De Freitas, and A. Doucet, "Toward practical n2 monte carlo: The marginal particle filter," in *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence, UAI 2005*, pp. 308–315, 2005.

[113] R. Chiplunkar and B. Huang, "Output relevant slow feature extraction using partial least squares," *Chemometrics and Intelligent Laboratory Systems*, vol. 191, pp. 148–157, 2019.

[114] C. Shang, X. Huang, J. A. K. Suykens, and D. Huang, "Enhancing dynamic soft sensors based on dpls: A temporal smoothness regularization approach," *Journal of Process Control*, vol. 28, pp. 17–26, 2015.

[115] L. Fortuna, S. Graziani, A. Rizzo, and M. G. Xibilia, *Soft Sensors for Monitoring and Control of Industrial Processes (Advances in Industrial Control)*. Springer-Verlag London, 1 ed., 2007.

[116] K. Fujiwara, M. Kano, S. Hasebe, and A. Takinami, "Soft-sensor development using correlation-based just-in-time modeling," *AIChE Journal*, vol. 55, no. 7, pp. 1754–1765, 2009.

[117] R. Chiplunkar and B. Huang, "Siamese neural network-based supervised slow feature extraction for soft sensor application," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 9, pp. 8953–8962, 2021.

[118] Y. Zhang, Y. Teng, and Y. Zhang, "Complex process quality prediction using modified kernel partial least squares," *Chemical Engineering Science*, vol. 65, no. 6, pp. 2153–2158, 2010.

[119] H. Kaneko and K. Funatsu, "Application of online support vector regression for soft sensors," *AIChE Journal*, vol. 60, no. 2, pp. 600–612, 2014.

[120] G. E. Hinton, S. Osindero, and Y. . Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[121] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems 19* (B. Schölkopf, J. C. Platt, and T. Hoffman, eds.), pp. 153–160, MIT Press, 2007.

[122] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[123] L. Fortuna, S. Graziani, and M. G. Xibilia, "Soft sensors for product quality monitoring in debutanizer distillation columns," *Control Engineering Practice*, vol. 13, no. 4, pp. 499–508, 2005.

[124] J. C. B. Gonzaga, L. A. C. Meleiro, C. Kiang, and R. Maciel Filho, "Ann-based soft-sensor for real-time process monitoring and control of an industrial polymerization process," *Computers and Chemical Engineering*, vol. 33, no. 1, pp. 43–49, 2009.

[125] S. A. Iliyas, M. Elshafei, M. A. Habib, and A. A. Adeniran, "Rbf neural network inferential sensor for process emission monitoring," *Control Engineering Practice*, vol. 21, no. 7, pp. 962–970, 2013.

[126] C. Shang, F. Yang, D. Huang, and W. Lyu, "Data-driven soft sensor development based on deep learning technique," *Journal of Process Control*, vol. 24, no. 3, pp. 223–233, 2014.

[127] Y. Liu, Y. Fan, and J. Chen, "Flame images for oxygen content prediction of combustion systems using dbn," *Energy and Fuels*, vol. 31, no. 8, pp. 8776–8783, 2017.

[128] L. Yao and Z. Ge, "Deep learning of semisupervised process data with hierarchical extreme learning machine and soft sensor application," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 2, pp. 1490–1498, 2017.

[129] W. Yan, D. Tang, and Y. Lin, "A data-driven soft sensor modeling method based on deep learning and its application," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 5, pp. 4237–4245, 2017.

[130] X. Yuan, B. Huang, Y. Wang, C. Yang, and W. Gui, "Deep learning-based feature representation and its application for soft sensor modeling with variable-wise weighted sae," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3235–3243, 2018.

[131] X. Yuan, C. Ou, and Y. Wang, "Development of nvw-saes with nonlinear correlation metrics for quality-relevant feature learning in process data modeling," *Measurement Science and Technology*, vol. 32, no. 1, 2021.

[132] X. Yuan, J. Zhou, B. Huang, Y. Wang, C. Yang, and W. Gui, "Hierarchical quality-relevant feature representation for soft sensor modeling: A novel deep learning strategy," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 3721–3730, 2020.

[133] R. Xie, N. M. Jan, K. Hao, L. Chen, and B. Huang, "Supervised variational autoencoders for soft sensor modeling with missing data," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2820–2828, 2020.

[134] B. Shen and Z. Ge, "Supervised nonlinear dynamic system for soft sensor application aided by variational auto-encoder," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 9, pp. 6132–6142, 2020.

[135] Q. Sun and Z. Ge, "Probabilistic sequential network for deep learning of complex process data and soft sensor application," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2700–2709, 2019.

[136] X. Yuan, L. Li, Y. Shardt, Y. Wang, and C. Yang, "Deep learning with spatiotemporal attention-based lstm for industrial soft sensor model development," *IEEE Transactions on Industrial Electronics*, pp. 1–1, 2020.

[137] X. Yuan, L. Li, and Y. Wang, "Nonlinear dynamic soft sensor modeling with supervised long short-term memory network," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3168–3176, 2020.

[138] Y. Huang, J. Zhao, Y. Liu, S. Luo, Q. Zou, and M. Tian, "Nonlinear dimensionality reduction using a temporal coherence principle," *Information Sciences*, vol. 181, no. 16, pp. 3284–3307, 2011.

[139] J. Weston, F. Ratle, and R. Collobert, "Deep learning via semi-supervised embedding," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 1168–1175, 2008.

[140] H. Mobahi, R. Collobert, and J. Weston, "Deep learning from temporal coherence in video," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, (New York, NY, USA), pp. 737–744, ACM, 2009.

[141] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. Lecun, "Unsupervised learning of spatiotemporally coherent metrics," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 International Conference on Computer Vision, ICCV 2015, pp. 4086–4093, 2015.

[142] D. Jayaraman and K. Grauman, "Slow and steady feature analysis: Higher order temporal coherence in video," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3852–3861, June 2016.

[143] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 1735–1742, 2006.

[144] R. Chiplunkar and B. Huang, "Filtering and smoothing of hidden monotonic trends and application to fouling detection," in *IFAC-PapersOnLine*, vol. 54, pp. 427–432, 2021.

[145] R. Chiplunkar and B. Huang, "Latent variable modeling and state estimation of non-stationary processes driven by monotonic trends," *Journal of Process Control*, vol. 108, pp. 40–54, 2021.

[146] S. Johansen and K. Juselius, "Maximum likelihood estimation and inference on cointegration — with applications to the demand for money," *Oxford Bulletin of Economics and Statistics*, vol. 52, no. 2, pp. 169–210, 1990.

[147] C. Zhao and B. Huang, "A full-condition monitoring method for nonstationary dynamic chemical processes with cointegration and slow feature analysis," *AIChE Journal*, vol. 64, no. 5, pp. 1662–1681, 2018.

[148] S. Kwak, Y. Ma, and B. Huang, "Extracting nonstationary features for process data analytics and application in fouling detection," *Computers and Chemical Engineering*, vol. 135, 2020.

[149] P. Von Bünau, F. C. Meinecke, F. C. Király, and K. . Müller, "Finding stationary subspaces in multivariate time series," *Physical Review Letters*, vol. 103, no. 21, 2009.

[150] S. Hara, Y. Kawahara, T. Washio, P. von Bünau, T. Tokunaga, and K. Yumoto, "Separation of stationary and non-stationary sources with a generalized eigenvalue problem," *Neural Networks*, vol. 33, pp. 7–20, 2012.

[151] J. Chen and C. Zhao, "Exponential stationary subspace analysis for stationary feature analytics and adaptive nonstationary process monitoring," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8345–8356, 2021.

[152] S. Särkkä, *Bayesian filtering and smoothing*, vol. 3. Cambridge University Press, 2013.

[153] D. H. Iversen, "Closed-skew distributions: Simulation, inversion and parameter estimation," Master's thesis, Institutt for matematiske fag, 2010.

[154] S. M. Alsadaie and I. M. Mujtaba, "Dynamic modelling of heat exchanger fouling in multistage flash (msf) desalination," *Desalination*, vol. 409, pp. 47–65, 2017.

[155] H. Benaroya, S. M. Han, and M. Nagurka, *Probability models in engineering and science*, vol. 192. CRC press, 2005.

[156] Y. W. Teh, D. Newman, and M. Welling, "A collapsed variational bayesian inference algorithm for latent dirichlet allocation," in *Advances in Neural Information Processing Systems*, pp. 1353–1360, 2007.

[157] D. B. S. Chiappa, "Unified inference for variational bayesian linear gaussian state-space models," in *Advances in neural information processing systems 19: Proceedings of the 2006 conference*, vol. 19, p. 81, MIT Press, 2007.

[158] Q. Jiang, B. Thornton, J. Russel-Houston, and S. Spence, "Review of thermal recovery technologies for the clearwater and lower grand rapids formations in cold lake, alberta," *Journal of Canadian Petroleum Technology*, vol. 49, no. 9, pp. 57–68, 2010.

[159] A. K. Gupta, G. González-Farías, and J. A. Domínguez-Molina, "A multivariate skew normal distribution," *Journal of Multivariate Analysis*, vol. 89, no. 1, pp. 181–190, 2004.

[160] W. Jeffrey Miller, "Lecture notes on advanced stochastic modeling," *Duke University, Durham, NC*, 2016.

Appendix A

Detailed derivation of the E-step of Chapter 5

A.1 Derivation of the prediction step

The derivation of the prediction step for our case is given in this section. Before proceeding, we state a few lemmas relevant to the derivation. The following lemma is from Gupta at al. [159].

Lemma 1 *If $V \sim \mathcal{N}_p(V;\ \mu, \Sigma)$, then for a given $q \times p$ matrix Γ, and $q \times 1$ vector a, the following result holds.*

$$\int \mathcal{N}_p(V;\ \mu, \Sigma)\Phi_q(a + \Gamma V;\ \nu, \Delta)\ dV = \Phi_q(a + \Gamma\mu;\ \nu, \Delta + \Gamma\Sigma\Gamma') \tag{A.1}$$

The following lemma gives expressions for the Bayesian rule for the case of Gaussian distributions [160].

Lemma 2 *Give a conditional distribution $p(y|z) \sim \mathcal{N}(y;\ Az+b, C)$ and the marginal $p(z) \sim \mathcal{N}(z;\ m, V)$, applying the Bayesian rule, the following equation holds.*

$$\begin{aligned}\mathcal{N}(y;\ Az + b, C) \times \mathcal{N}(z;\ m, V) =& \mathcal{N}(y;\ Am + b, AVA' + C) \\ & \times \mathcal{N}(z;\ m + K(y - (Am + b)), (I - KA)V) \\ =& \mathcal{N}\left(\begin{bmatrix} y \\ z \end{bmatrix}; \begin{bmatrix} Am + b \\ m \end{bmatrix}, \begin{bmatrix} AVA' + C & AV \\ VA' & V \end{bmatrix}\right)\end{aligned} \tag{A.2}$$

with $K = VA'(AVA' + C)^{-1}$

We can now proceed to derive the equations in the prediction step. Substituting (5.40) and (5.37) into (5.34) yields

$$\begin{aligned} p(m_t, s_t|Y_{t-1}) \propto \int & \mathcal{N}_{n_s+1}\left(\begin{bmatrix} m_t \\ s_t \end{bmatrix}; \begin{bmatrix} m_{t-1} \\ As_{t-1} \end{bmatrix}, \begin{bmatrix} \sigma_e^2 & \mathbf{0} \\ \mathbf{0} & \Sigma_v \end{bmatrix}\right) \Phi_1\left(\gamma \begin{bmatrix} m_t - m_{t-1} \\ s_t - As_{t-1} \end{bmatrix}; \, 0, \delta\right) \\ & \times \mathcal{N}_{n_s+1}\left(\begin{bmatrix} m_{t-1} \\ s_{t-1} \end{bmatrix}; \begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix}, \Sigma_{t-1|t-1}\right) \\ & \Phi_{t-1}\left(\Gamma_{t-1|t-1} \begin{bmatrix} m_{t-1} - m_{t-1|t-1} \\ s_{t-1} - s_{t-1|t-1} \end{bmatrix}; \nu_{t-1|t-1}, \Delta_{t-1|t-1}\right) dm_{t-1} \, ds_{t-1} \end{aligned} \tag{A.3}$$

A proportionality sign is used in the above equation as the normalizing constants are ignored in the pdfs. We now define the following two matrices.

$$\tilde{A} = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & A \end{bmatrix}; \quad \Sigma_S = \begin{bmatrix} \sigma_e^2 & \mathbf{0} \\ \mathbf{0} & \Sigma_v \end{bmatrix} \tag{A.4}$$

One can use Lemma 2 to convert the product of the two Gaussian distributions into the following product.

$$\begin{aligned} & \mathcal{N}_{n_s+1}\left(\begin{bmatrix} m_t \\ s_t \end{bmatrix}; \tilde{A}\begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix}, \tilde{A}\Sigma_{t-1|t-1}\tilde{A}' + \Sigma_S\right) \\ & \times \mathcal{N}_{n_s+1}\left(\begin{bmatrix} m_{t-1} \\ s_{t-1} \end{bmatrix}; \begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix} + J_t\left(\begin{bmatrix} m_t \\ s_t \end{bmatrix} - \tilde{A}\begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix}\right), (I - J_t\tilde{A})\Sigma_{t-1|t-1}\right) \end{aligned} \tag{A.5}$$

with $J_t = \Sigma_{t-1|t-1}\tilde{A}'(\tilde{A}\Sigma_{t-1|t-1}\tilde{A}' + \Sigma_S)^{-1}$. It can be observed that the first Gaussian term is independent of the integration variables which are the state variables at $t-1$. Hence it comes out of the integration. The two cdf terms in (A.3) can be combined as follows to get the following cdf.

$$\Phi_t\left(\begin{bmatrix} \Gamma_{t-1|t-1}\begin{bmatrix} m_{t-1} - m_{t-1|t-1} \\ s_{t-1} - s_{t-1|t-1} \end{bmatrix} \\ \gamma\begin{bmatrix} m_t - m_{t-1} \\ s_t - As_{t-1} \end{bmatrix} \end{bmatrix}; \begin{bmatrix} \nu_{t-1|t-1} \\ 0 \end{bmatrix}, \begin{bmatrix} \Delta_{t-1|t-1} & \mathbf{0} \\ \mathbf{0} & \delta \end{bmatrix}\right) \tag{A.6}$$

Substituting (A.6) and (A.5) into (A.3), one can arrive at a CSN whose Gaussian part is

$$\mathcal{N}_{n_s+1}\left(\begin{bmatrix} m_t \\ s_t \end{bmatrix}; \tilde{A}\begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix}, \tilde{A}\Sigma_{t-1|t-1}\tilde{A}' + \Sigma_S\right) \tag{A.7}$$

and the cdf part is the integral

$$\int \mathcal{N}_{n_s+1}\Bigg(\begin{bmatrix} m_{t-1} \\ s_{t-1} \end{bmatrix}; \begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix} + J_t \left(\begin{bmatrix} m_t \\ s_t \end{bmatrix} - \tilde{A} \begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix} \right), (I - J_t\tilde{A})\Sigma_{t-1|t-1} \Bigg) \times \Phi_t(\dots) dm_{t-1}\, ds_{t-1} \tag{A.8}$$

The above integral can be solved using Lemma 1 and with some algebraic simplifications, one can arrive at the CSN parameters given in (5.42).

A.2 Derivation of the update step

Given the prior

$$p(m_t, s_t|Y_{t-1}) = CSN_{n_s+1,t}\left(\begin{bmatrix} m_t \\ s_t \end{bmatrix};\ \begin{bmatrix} m_{t|t-1} \\ s_{t|t-1} \end{bmatrix}, \Sigma_{t|t-1}, \Gamma_{t|t-1}, \nu_{t|t-1}, \Delta_{t-1|t-1} \right) \tag{A.9}$$

and the likelihood

$$p(y_t|m_t, s_t) = \mathcal{N}_{n_y}(y_t;\ H_1 m_t + H_2 s_t, \Sigma_u), \tag{A.10}$$

the Bayesian rule given in (5.35) can be applied to get the posterior distribution.

$$p(m_t, s_t|Y_t) \propto \mathcal{N}_{n_y}(y_t;\ H_1 m_t + H_2 s_t, \Sigma_u) \times \mathcal{N}_{n_s+1}\left(\begin{bmatrix} m_t \\ s_t \end{bmatrix}; \begin{bmatrix} m_{t|t-1} \\ s_{t|t-1} \end{bmatrix}, \Sigma_{t|t-1} \right) \times \Phi_t\left(\Gamma_{t|t-1} \begin{bmatrix} m_t - m_{t|t-1} \\ s_t - s_{t|t-1} \end{bmatrix};\ \nu_{t|t-1}, \Delta_{t|t-1} \right) \tag{A.11}$$

Using Lemma 2, the product of the two Gaussian distributions in the above equation can be rewritten. One of the resulting pdfs will be independent of m_t and y_t and can be clubbed into the normalizing constant. This results in the following expression

$$p(m_t, s_t|Y_t) \propto \mathcal{N}_{n_s+1}\Bigg(\begin{bmatrix} m_t \\ s_t \end{bmatrix}; \begin{bmatrix} m_{t|t-1} \\ s_{t|t-1} \end{bmatrix} + K_t(y_t - H_1 m_{t|t-1} - H_2 s_{t|t-1}), (I - K_t H)\Sigma_{t|t-1} \Bigg) \times \Phi_t\left(\Gamma_{t|t-1} \begin{bmatrix} m_t - m_{t|t-1} \\ s_t - s_{t|t-1} \end{bmatrix};\ \nu_{t|t-1}, \Delta_{t|t-1} \right) \tag{A.12}$$

where $K_t = \Sigma_{t|t-1}H'(H\Sigma_{t|t-1}H' + \Sigma_u)^{-1}$. We can redefine

$$\begin{bmatrix} m_{t|t-1} \\ s_{t|t-1} \end{bmatrix} + K_t(y_t - H_1 m_{t|t-1} - H_2 s_{t|t-1}) = \begin{bmatrix} m_{t|t} \\ s_{t|t} \end{bmatrix}; (I - K_t H)\Sigma_{t|t-1} = \Sigma_{t|t} \tag{A.13}$$

To get a CSN form for (A.12), the cdf term needs to be readjusted such that the posterior μ parameter as defined in the above equation, is subtracted from the state variables. This leads to the updated parameters as defined in (5.45).

A.3 Derivation of the smoothing step

We derive the equations in the smoothing step by first determining the parameter updates from $t = T$ to $t = T-1$, and then from $t = T-1$ to $t = T-2$ which leads to the generalized recursive equations given in (5.47). Substituting the expressions of all the CSN distributions into (5.36) results in a cluttered and cumbersome representation of the equations. Hence, the derivation will be explained step by step showing the necessary expressions resulting from each step.

A.3.1 $t = T$ to $t = T - 1$

At the end of the forward pass we have

$$p(m_T, s_T|Y_T) = CSN_{n_s+1,T}\left(\begin{bmatrix} m_T \\ s_T \end{bmatrix};\ \begin{bmatrix} m_{T|T} \\ s_{T|T} \end{bmatrix}, \Sigma_{T|T}, \Gamma_{T|T}, \nu_{T|T}, \Delta_{T|T}\right) \tag{A.14}$$

The backward pass involves the calculations given in (5.36). It can be observed from this equation that there are two steps in this procedure. The first is a Bayesian inversion followed by a convolution of the obtained pdfs from Bayesian inversion and the smoothed pdf from the previous step.

$$\begin{aligned} p(m_{T-1}, s_{T-1}|Y_T) = \int & \frac{p(m_T, s_T|m_{T-1}, s_{T-1})\ p(m_{T-1}, s_{T-1}|Y_{T-1})}{\int p(m_T, s_T|m_{T-1}, s_{T-1})\ p(m_{T-1}, s_{T-1}|Y_{T-1})\ dm_{T-1}ds_{T-1}} \\ & \times p(m_T, s_T|Y_T)\ dm_T\ ds_T \end{aligned} \tag{A.15}$$

The numerator in the Bayesian inference part of the above equation is

$$\begin{aligned} & CSN_{n_s+1,1}\left(\begin{bmatrix} m_T \\ s_T \end{bmatrix};\ \begin{bmatrix} m_{T-1} \\ As_{T-1} \end{bmatrix}, \begin{bmatrix} \sigma_e^2 & \mathbf{0} \\ \mathbf{0} & \Sigma_v \end{bmatrix}, \gamma, 0, \delta\right) \\ & \times CSN_{n_s+1,T-1}\Bigg(\begin{bmatrix} m_{T-1} \\ s_{T-1} \end{bmatrix};\ \begin{bmatrix} m_{T-1|T-1} \\ s_{T-1|T-1} \end{bmatrix}, \Sigma_{T-1|T-1}, \\ & \qquad \Gamma_{T-1|T-1}, \nu_{T-1|T-1}, \Delta_{T-1|T-1}\Bigg) \end{aligned} \tag{A.16}$$

The denominator is nothing but the integration of the numerator w.r.t the state variables at $t = T - 1$. It can be observed that this product of two CSNs is nothing but the product of the CSNs obtained in the prediction step while moving from $t = T - 1$ to $t = T$. This simplifies a few things in the derivation. First, by applying Lemma 2 we get two Gaussian terms. One is the marginal of states at $t = T$ similar to the one in (A.7) and the second is a conditional of the form $T - 1|T$ similar to the

one in (A.8). The first one comes out of the integration canceling the same term in the denominator. The evaluation of the integral in the denominator now yields

$$\Phi_T\left(\Gamma_{T|T-1}\begin{bmatrix} m_T - m_{T|T-1} \\ s_T - s_{T|T-1}\end{bmatrix};\ \nu_{T|T-1}, \Delta_{T|T-1}\right) \tag{A.17}$$

From the update steps we know that the cdf term does not change in the Bayesian inference as we only merely readjust it to fit the definition of CSN. Hence this term cancels the cdf term of $p(m_T, s_T|Y_T)$. Equation (A.15) now becomes

$$\begin{aligned}&\int \mathcal{N}_{n_s+1}\left(\begin{bmatrix} m_{T-1} \\ s_{T-1}\end{bmatrix}; \begin{bmatrix} m_{T-1|T-1} \\ s_{T-1|T-1}\end{bmatrix} + J_T\begin{bmatrix} m_T - m_{T|T-1} \\ s_T - s_{T|T-1}\end{bmatrix}, (I - J_T\tilde{A})\Sigma_{T-1|T-1}\right)\\ &\times \Phi_1^p(\dots) \times \Phi_{T-1}(\dots) \times \mathcal{N}_{n_s+1}\left(\begin{bmatrix} m_T \\ s_T\end{bmatrix}; \begin{bmatrix} m_{T|T} \\ s_{T|T}\end{bmatrix}, \Sigma_{T|T}\right)\ dm_T\ ds_T\end{aligned} \tag{A.18}$$

Here, $\Phi_1^p(\dots)$ is the process noise CSN's cdf term and $\Phi_{T-1}(\dots)$ is the cdf term of the update step's CSN at $t = T-1$. Applying Lemma 2 to the two Gaussian terms, we get

$$\begin{aligned}&\mathcal{N}_{n_s+1}\left(\begin{bmatrix} m_{T-1} \\ s_{T-1}\end{bmatrix}; \begin{bmatrix} m_{T-1|T-1} \\ s_{T-1|T-1}\end{bmatrix} + J_T\begin{bmatrix} m_{T|T} - m_{T|T-1} \\ s_{T|T} - s_{T|T-1}\end{bmatrix}, (I - J_T\tilde{A})\Sigma_{T-1|T-1}\right) \times \Phi_{T-1}(\dots)\\ &\int \mathcal{N}_{n_s+1}\left(\begin{bmatrix} m_T \\ s_T\end{bmatrix}; \begin{bmatrix} m_{T|T} \\ s_{T|T}\end{bmatrix} + C_{T-1}\begin{bmatrix} m_{T-1} - m_{T-1|T} \\ s_{T-1} - s_{T-1|T}\end{bmatrix}, \Sigma^*_{T-1}\right) \times \Phi_1^p(\dots)\ dm_T\ ds_T\end{aligned} \tag{A.19}$$

Here,

$$\begin{aligned}C_{T-1} &= \Sigma_{T|T}J_T'(J_T\Sigma_{T|T}J_T' + (I - J_T\tilde{A})\Sigma_{T-1|T-1})^{-1}\\ \Sigma^*_{T-1} &= (I - C_{T-1}J_T)\Sigma_{T|T}\\ \begin{bmatrix} m_{T-1|T} \\ s_{T-1|T}\end{bmatrix} &= \begin{bmatrix} m_{T-1|T-1} \\ s_{T-1|T-1}\end{bmatrix} + J_T\begin{bmatrix} m_{T|T} - m_{T|T-1} \\ s_{T|T} - s_{T|T-1}\end{bmatrix}\\ \Sigma_{T-1|T} &= J_T\Sigma_{T|T}J_T' + (I - J_T\tilde{A})\Sigma_{T-1|T-1}\end{aligned} \tag{A.20}$$

The integral in the above equation can be solved using Lemma 1, and with some algebraic simplifications, we arrive at $\Phi_1^*(\cdots)$ characterized by

$$\begin{aligned}\Gamma^*_{T-1} &= \gamma(C_{T-1} - \tilde{A});\\ \nu^*_{T-1} &= -\gamma\begin{bmatrix} m_{T|T} - m_{T-1|T} \\ s_{T|T} - As_{T-1|T}\end{bmatrix};\\ \delta^*_{T-1} &= \delta + \gamma\Sigma^*_{T-1}\gamma'\end{aligned} \tag{A.21}$$

Finally, the $\Phi_{T-1}(\dots)$ term can be readjusted and combined with the Φ_1^* term to get the following parameters of the skewness part

$$\begin{aligned}
\Gamma_{T-1|T} &= \begin{bmatrix} \Gamma_{T-1|T-1} \\ \Gamma^*_{T-1} \end{bmatrix}; \\
\nu_{T-1|T} &= \begin{bmatrix} \nu_{T-1|T-1} - \Gamma_{T-1|T-1} \begin{bmatrix} m_{T-1|T} - m_{T-1|T-1} \\ s_{T-1|T} - s_{T-1|T-1} \end{bmatrix} \\ \nu^*_{T-1} \end{bmatrix}; \\
\Delta_{T-1|T} &= \begin{bmatrix} \Delta_{T-1|T-1} & \mathbf{0} \\ \mathbf{0} & \delta^*_{T-1} \end{bmatrix}
\end{aligned} \tag{A.22}$$

The CSN is defined as

$$p(m_{T-1}, s_{T-1}|Y_T) = CSN_{n_s+1,T}\left(\begin{bmatrix} m_{T-1} \\ s_{T-1} \end{bmatrix}; \begin{bmatrix} m_{T-1|T} \\ s_{T-1|T} \end{bmatrix}, \Sigma_{T-1|T}, \Gamma_{T-1|T}, \nu_{T-1|T}, \Delta_{T-1|T} \right) \tag{A.23}$$

It can be observed that the dimensionality of the skewness part of the CSN has remained constant in this step. It shall be seen that this dimension will remain constant at T throughout the backward pass.

A.3.2 $t = T-1$ to $t = T-2$

As in the previous step, the smoothed pdf at $t = T-2$ can be defined similar to (A.15). The important things to be noted in this step are listed in the following points:

1. Similar to the previous case, in the Bayesian inference, the denominator is left with $\Phi_{T-1}(\dots)$ term which is the same as the one on the forward pass.

2. The skewness term of $p(m_{T-1}, s_{T-1}|Y_T)$ can be written as $\Phi_{T-1}(\dots) \times \Phi_1^*(\dots)$. This results in the $\Phi_{T-1}(\dots)$ terms getting canceled.

3. The remaining Gaussian terms can be simplified using Lemma 2.

4. At the end, an equation similar to (A.19) is obtained but shifted by one time instant. The important difference is that, outside the integral, we have $T-2$ dimensional $\Phi_{T-2}(\dots)$ term and inside the integration, we have the product transition probability noise cdf term $\Phi_1^p(\dots)$ and $\Phi_1^*(\dots)$.

$$
\begin{aligned}
p(m_{T-2}, s_{T-2}|Y_T) \propto \mathcal{N}_{n_s+1}\Bigg(& \begin{bmatrix} m_{T-2} \\ s_{T-2} \end{bmatrix}; \begin{bmatrix} m_{T-2|T-2} \\ s_{T-2|T-2} \end{bmatrix} \\
& + J_{T-1} \begin{bmatrix} m_{T-1|T} - m_{T-1|T-2} \\ s_{T-1|T} - s_{T-1|T-2} \end{bmatrix}, (I - J_{T-1}\tilde{A})\Sigma_{T-2|T-2} \Bigg) \\
& \times \Phi_{T-2}(\dots) \\
& \int \mathcal{N}_{n_s+1}\Bigg(\begin{bmatrix} m_{T-1} \\ s_{T-1} \end{bmatrix}; \begin{bmatrix} m_{T-1|T} \\ s_{T-1|T} \end{bmatrix} \\
& + C_{T-2} \begin{bmatrix} m_{T-2} - m_{T-2|T} \\ s_{T-2} - s_{T-2|T} \end{bmatrix}, \Sigma^*_{T-2} \Bigg) \\
& \times \Phi_1^p(\dots) \times \Phi_1^* \dots \, dm_T \, ds_T
\end{aligned} \tag{A.24}
$$

5. Upon the application of Lemma 1 and some further algebraic simplifications, we arrive at a CSN with T dimensional skewness term.

The parameters of the CSN representing $p(m_{T-2}, s_{T-2}|Y_T)$ are given in the following equations.

$$
\begin{aligned}
C_{T-2} &= \Sigma_{T-1|T} J'_{T-1} \\
&\qquad \times (J_{T-1}\Sigma_{T-1|T}J'_{T-1} + (I - J_{T-1}\tilde{A})\Sigma_{T-2|T-2})^{-1} \\
\Sigma^*_{T-2} &= (I - C_{T-2}J_{T-1})\Sigma_{T-1|T} \\
\Gamma^*_{T-2} &= \begin{bmatrix} \gamma(C_{T-2} - \tilde{A}) \\ \Gamma^*_{T-1}C_{T-2} \end{bmatrix} \\
\nu^*_{T-2} &= \begin{bmatrix} -\gamma \begin{bmatrix} m_{T-1|T} - m_{T-2|T} \\ s_{T-1|T} - As_{T-2|T} \end{bmatrix} \\ \nu^*_{T-1} \end{bmatrix} \\
\delta^*_{T-2} &= \begin{bmatrix} \delta & \mathbf{0} \\ \mathbf{0} & \delta^*_{T-1} \end{bmatrix} + \begin{bmatrix} \gamma \\ \Gamma^*_{T-1} \end{bmatrix} \Sigma^*_{T-2} \begin{bmatrix} \gamma' & \Gamma^*_{T-1}{}' \end{bmatrix} \\
\begin{bmatrix} m_{T-2|T} \\ s_{T-2|T} \end{bmatrix} &= \begin{bmatrix} m_{T-2|T-2} \\ s_{T-2|T-2} \end{bmatrix} + J_{T-1} \begin{bmatrix} m_{T-1|T} - m_{T-1|T-2} \\ s_{T-1|T} - s_{T-1|T-2} \end{bmatrix} \\
\Sigma_{T-2|T} &= J_{T-1}\Sigma_{T-1|T}J'_{T-1} + (I - J_{T-1}\tilde{A})\Sigma_{T-2|T-2} \\
\Gamma_{T-2|T} &= \begin{bmatrix} \Gamma_{T-2|T-2} \\ \Gamma^*_{T-2} \end{bmatrix} \\
\nu_{T-2|T} &= \begin{bmatrix} \nu_{T-2|T-2} - \Gamma_{T-2|T-2} \begin{bmatrix} m_{T-2|T} - m_{T-2|T-2} \\ s_{T-2|T} - s_{T-2|T-2} \end{bmatrix} \\ \nu^*_{T-2} \end{bmatrix} \\
\Delta_{T-2|T} &= \begin{bmatrix} \Delta_{T-2|T-2} & \mathbf{0} \\ \mathbf{0} & \delta^*_{T-2} \end{bmatrix}
\end{aligned} \tag{A.25}
$$

The final distribution at $t = T - 2$ is

$$p(m_{T-2}, s_{T-2}|Y_T) = CSN_{n_s+1,T}\left(\begin{bmatrix} m_{T-2} \\ s_{T-2} \end{bmatrix}; \begin{bmatrix} m_{T-2|T} \\ s_{T-2|T} \end{bmatrix}, \Sigma_{T-2|T}, \Gamma_{T-2|T}, \nu_{T-2|T}, \Delta_{T-2|T} \right) \tag{A.26}$$

The parameters can then be updated recursively till $t = 1$ is reached.

A.4 Cross-time distribution

To estimate the parameters related to the transition probabilities, one needs to find the cross-time distributions that contain the cross-time covariances. Based on how the smoothing equation in (5.36) is arrived at, we have the following result:

$$\begin{aligned} p(m_{t-1}, s_{t-1}, m_t, s_t|Y_T) =& \frac{p(m_t, s_t|m_{t-1}, s_{t-1})p(m_{t-1}, s_{t-1}|Y_{t-1})}{\int p(m_t, s_t|m_{t-1}, s_{t-1})p(m_{t-1}, s_{t-1}|Y_{t-1})\ dm_{t-1}\ ds_{t-1}} \\ & \times p(m_t, s_t|Y_T) \end{aligned} \tag{A.27}$$

The smoothing equation is obtained by marginalizing the above equation. Hence by referring to (A.18) and (A.24), the following result is obtained.

$$\begin{aligned} p(m_{t-1}, s_{t-1}, m_t, s_t|Y_T) \propto \mathcal{N}_{n_s+1}\Bigg(& \begin{bmatrix} m_{t-1} \\ s_{t-1} \end{bmatrix}; \begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix} + J_t \begin{bmatrix} m_t - m_{t|t-1} \\ s_t - s_{t|t-1} \end{bmatrix}, \\ & (I - J_t\tilde{A})\Sigma_{t-1|t-1} \Bigg) \times \Phi_1^p(\dots) \times \Phi_{t-1}(\dots) \\ & \times \mathcal{N}_{n_s+1}\left(\begin{bmatrix} m_t \\ s_t \end{bmatrix}; \begin{bmatrix} m_{t|T} \\ s_{t|T} \end{bmatrix}, \Sigma_{t|T} \right) \times \Phi_{T-t}^*(\dots) \end{aligned} \tag{A.28}$$

The two Gaussian terms can be combined according to Lemma 2. Hence the Gaussian part of the CSN can be written as

$$\begin{aligned} \mathcal{N}_{2n_s+2}\Bigg(& \begin{bmatrix} m_{t-1} \\ s_{t-1} \\ m_t \\ s_t \end{bmatrix}; \begin{bmatrix} \begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1} \end{bmatrix} + J_t \begin{bmatrix} m_{t|T} - m_{t|t-1} \\ s_{t|T} - s_{t|t-1} \end{bmatrix} \\ \begin{bmatrix} mt|T \\ s_{t|T} \end{bmatrix} \end{bmatrix}, \\ & \begin{bmatrix} J_t\Sigma_{t|T}J_t' + (I - J_t\tilde{A})\Sigma_{t-1|t-1} & J_t\Sigma_{t|T} \\ \Sigma_{t|T}J_t' & \Sigma_{t|T} \end{bmatrix} \Bigg) \end{aligned} \tag{A.29}$$

The cdf terms can be combined as follows.

$$\Phi_T\left(\begin{bmatrix} \gamma\begin{bmatrix} m_t - m_{t-1} \\ s_t - As_{t-1}\end{bmatrix} \\ \Gamma_{t-1|t-1}\begin{bmatrix} m_{t-1} - m_{t-1|t-1} \\ s_{t-1} - s_{t-1|t-1}\end{bmatrix} \\ \Gamma_t^*\begin{bmatrix} m_t - m_{t|T} \\ s_t - s_{t|T}\end{bmatrix}\end{bmatrix}; \begin{bmatrix} 0 \\ \nu_{t-1|t-1} \\ \nu_t^*\end{bmatrix}, \begin{bmatrix} \delta & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Delta_{t-1|t-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Delta_t^*\end{bmatrix}\right) \tag{A.30}$$

It can be observed that the dimensionality of the combined $\Phi(\dots)$ is T. Considering the ν term, one can see that the dimensionality of the ν of the noise term, $\nu_{t-1|t-1}$, and ν_t^* are 1, $t-1$, and $T-t$ respectively. Now, the above $\Phi_T(\dots)$ term needs to be readjusted so that the overall distribution takes the form of a CSN. We define the following matrices to simplify the notations.

$$\begin{aligned} &I_1 = \begin{bmatrix} I & \mathbf{0}\end{bmatrix}; \quad I_2 = \begin{bmatrix} \mathbf{0} & I\end{bmatrix}; \\ &\mu_{t-1,t|T} = \begin{bmatrix} \begin{bmatrix} m_{t-1|t-1} \\ s_{t-1|t-1}\end{bmatrix} + J_t\begin{bmatrix} m_{t|T} - m_{t|t-1} \\ s_{t|T} - s_{t|t-1}\end{bmatrix} \\ \begin{bmatrix} m_{t|T} \\ s_{t|T}\end{bmatrix}\end{bmatrix} \end{aligned} \tag{A.31}$$

The dimensions of I_1 and I_2 are chosen such that

$$I_1\begin{bmatrix} m_{t-1} \\ s_{t-1} \\ m_t \\ s_t\end{bmatrix} = \begin{bmatrix} m_{t-1} \\ s_{t-1}\end{bmatrix}; \quad I_2\begin{bmatrix} m_{t-1} \\ s_{t-1} \\ m_t \\ s_t\end{bmatrix} = \begin{bmatrix} m_t \\ s_t\end{bmatrix} \tag{A.32}$$

With these definitions, we can rewrite the Gamma terms in the combined cdf term as follows.

$$\gamma\begin{bmatrix} m_t - m_{t-1} \\ s_t - As_{t-1}\end{bmatrix} = \gamma(I_2 - \tilde{A}I_1)\begin{bmatrix} m_{t-1} \\ s_{t-1} \\ m_t \\ s_t\end{bmatrix} \tag{A.33}$$

The above term and the corresponding ν term can be readjusted by subtracting $\mu_{t-1,t|T}$ from both to make it a CSN. The same procedure can be followed for all the Γ and ν terms of the cdf term in (A.30) to arrive at the CSN given in (5.50).

A.5 Calculating the moments of a CSN

We briefly discuss the evaluation of moments of a CSN with the dimension of the skewness term greater than 1. The approach is similar to the one given by He et

al. [85]. Consider $x \sim CSN_{n,q}(x;\ \mu, \Sigma, \Gamma, \nu, \Delta)$. To evaluate the mean one must evaluate the following integral.

$$\begin{aligned}\mathbb{E}[x] &= \int_{-\infty}^{\infty} x \frac{\mathcal{N}_n(x;\ \mu, \Sigma)\ \Phi_q(\Gamma(x-\mu);\ \nu, \Delta)}{\Phi_q(0;\ \nu, \Delta + \Gamma\Sigma\Gamma')} dx \\ &= \frac{1}{\Phi_q(0;\ \nu, \Delta + \Gamma\Sigma\Gamma')} \int_{-\infty}^{\infty} x\mathcal{N}_n(x;\ \mu, \Sigma) \int_{-\infty}^{0} \mathcal{N}_q(p;\ \nu - \Gamma(x-\mu), \Delta)\ dp\ dx\end{aligned} \tag{A.34}$$

The Gaussian term with x can be brought into the inner integration. Applying Lemma 2, the following result can be obtained.

$$\mathbb{E}[x] = \frac{1}{\Phi_q(0;\ \nu, \Delta + \Gamma\Sigma\Gamma')} \times \int_{-\infty}^{0} \mathcal{N}_q(p;\ \nu, \Delta + \Gamma\Sigma\Gamma') \int_{-\infty}^{\infty} x\mathcal{N}_n(x;\ \mu + K(p-\nu), (I + K\Gamma)\ dx\ dp \tag{A.35}$$

with $K = -\Sigma\Gamma'(\Delta + \Gamma\Sigma\Gamma')^{-1}$. The inner integral of the above equation is $\mu + K(p-\nu)$. It can be observed that the normalizing term in the denominator is $\int_{-\infty}^{0} \mathcal{N}_q(p;\ \nu, \Delta + \Gamma\Sigma\Gamma')\ dp$ which is a normalizing constant of a truncated Gaussian distribution with mean ν and variance $\Delta + \Gamma\Sigma\Gamma'$, truncated above zero in every dimension of p. Incorporating these, one can arrive at the following result.

$$\mathbb{E}[x] = \mu - K\nu + K\mathbb{E}[p];\ p \sim \frac{\mathcal{N}_q(p;\ \nu, \Delta + \Gamma\Sigma\Gamma')}{\Phi_q(0;\ \nu, \Delta + \Gamma\Sigma\Gamma')} \mathbf{1}(p \leq 0) \tag{A.36}$$

Since the mean of a truncated Gaussian cannot be calculated analytically for dimensions greater than 1, one can draw N samples of p from the truncated Gaussian distribution with the said parameters and calculate the sample mean and hence the expected value of x. Similarly, one can arrive at the following equation for the covariance matrix of a CSN.

$$\mathrm{Var}[x] = \Sigma + K\Gamma\Sigma + K(\mathrm{Var}[p])K' \tag{A.37}$$

The variance of p can be calculated from the samples generated from its corresponding truncated Gaussian distribution.