ADAPTIVE-MODEL BASED SELF-TUNING GENERALIZED PREDICTIVE CONTROL OF A BIODIESEL REACTOR

HO YONG KUEN

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2011

ADAPTIVE-MODEL BASED SELF-TUNING GENERALIZED PREDICTIVE CONTROL OF A BIODIESEL REACTOR

HO YONG KUEN

**DISSERTATION SUBMITTED IN FULFILMENT
OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF ENGINEERING SCIENCE**

**FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR**

2011

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: *Ho Yong Kuen*

Registration/ Matric No.: *KGA 080068*

Name of Degree: *Master of Engineering Science*

Title of Project/ Paper/ Research Report/ Dissertation/ Thesis ("this Work"): *Adaptive-Model Based Self-Tuning Generalized Predictive Control of a Biodiesel Reactor*

Field of Study: *Advanced Process Control*

I do solemnly and sincerely declare that:

(1) I am the sole author/ writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every tights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether I intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's signature Date:

Subscribed and solemnly declared before,

Witness's signature Date:

Name:

Designation:

ABSTRACT

Traditionally, the Recursive Least Squares (RLS) algorithm was used in the Generalized Predictive Control (GPC) framework solely for model adaptation purposes. In this work, the RLS algorithm was extended to also cater for self-tuning of the controller. Specifically, the analytical expressions proposed by Shridhar and Cooper (1997b) for offline tuning of the move suppression weight was deployed for online tuning. This new combination, denoted as the Adaptive-Model Based Self-Tuning Generalized Predictive Control (AS-GPC), contains both model adaptation and self-tuning capabilities within the same controller structure. Several RLS algorithms were screened and the Variable Forgetting Factor Recursive Least Squares (VFF-RLS) algorithm was selected to capture the dynamics of the process online for the purpose of model adaptation in the controller. Based on the evolution of the process dynamics given by the VFF-RLS algorithm in the form of First Order Plus Dead Time (FOPDT) model parameters, the move suppression weight for the AS-GPC was recalculated automatically at every time step based on the analytical tuning expressions. The proposed control scheme was tested and implemented on a validated mechanistic transesterification process, known for inherent nonlinearities. Closed loop simulation of the transesterification reactor revealed the superiority of the proposed control scheme in terms of servo and regulatory control as compared to other variants of advanced controllers and the conventional PID controller. Not only is the proposed control scheme adept in tackling issues of process nonlinearities, it also minimizes user involvement in the tuning of the controller and consequently reduces process interruptions.

ABSTRAK

Secara tradisional, algoritma Kuasa Dua Minima Rekursif (KDMR) digunakan dalam kerangka Kawalan Ramalan Am (KRA) semata-mata untuk tujuan adaptasi model. Dalam karya ini, algoritma KDMR dilanjutkan untuk melingkungi penalaan pengawal secara automatik. Khususnya, persamaan analitikal yang dicadangkan oleh Shridhar dan Cooper (1997a) untuk penalaan pemberat penekanan pergerakan secara luar talian digunakan untuk penalaan dalam talian. Gabungan baru ini, yang dinamakan sebagai Kawalan Ramalan Am berdasarkan Prinsip Adaptasi Model dan Penalaan Automatik (KRA-PAMPA), menpunyai keupayaan adaptasi model dan penalaan automatik dalam struktur pengawal yang sama. Beberapa algoritma KDMR telah ditapis dan algoritma Kuasa Dua Minima Rekursif dengan Faktor Perlupaan Berubah (KDMR-FPB) dipilih untuk menganggar dinamik proses secara dalam talian untuk tujuan adaptasi model dalam struktur pengawal. Berdasarkan evolusi dinamik proses yang dianggarkan oleh algoritma KDMR-FPB dalam bentuk parameter model Tertib Pertama dengan Masa Mati (TPMM), pemberat penekanan pergerakan untuk KRA-PAMPA dihitung semula secara automatik pada setiap sela waktu dengan menggunakan persamaan penalaan analitikal tersebut. Skema kawalan yang dicadangkan ini diuji dan dilaksanakan ke atas satu proses transesterifikasi mekanistik yang telah disahkan dan dikenali dengan dinamik sejati yang tidak lelurus. Simulasi gelung tertutup reaktor transesterifikasi memaparkan keunggulan skema kawalan yang dicadangkan, baik dalam kawalan servo mahupun dalam kawalan gangguan proses berbanding dengan pengawal termaju yang lain dan juga pengawal PID. Skema kawalan yang dicadangkan bukan sahaja mahir dalam menangani ketaklelurusan proses, tetapi juga dapat mengurangkan penglibatan pengguna dalam hal-ehwal penalaan pengawal dan seterusnya mengurangkan gangguan proses.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my God and Lord Jesus Christ, without whom I would not have the strength and patience to complete this thesis. To Him be the glory in everything that I do. Meanwhile, I would like to take this opportunity to thank:

i) My supervisors Assoc. Prof. Dr. Farouq S. Mjalli from the Petroleum & Chemical Engineering Department (Sultan Qaboos University) and Dr. Yeoh Hak Koon from the Department of Chemical Engineering (University of Malaya, Malaysia) for their invaluable support and guidance throughout this period. Without their guidance and supervision, this work would have been an impossible uphill task.

ii) Prof. Miroslav Fikar from the Department of Information Engineering and Process Control (Slovak University of Technology, Slovak Republic) for his willingness in always communicating and sharing his experience and knowledge in the areas of research concerned, from which I have greatly benefited.

iii) Prof. Karl Johan Åström from the Department of Automatic Control (Lund University, Sweden) and the Department of Mechanical Engineering (UC Santa Barbara, USA), from whom I was able to obtain clarification on specific technical details presented in his book (Åström & Wittenmark, 1994).

iv) Dr. Anthony Rossiter (Reader) from the Department of Automatic Control and Systems Engineering (University of Sheffield, United Kingdom) for his generosity in sharing, with illuminating clarity, the details presented in his book (Rossiter, 2003).

Given this opportunity, I would like to also express my sincere thanks to my family and friends, who have all along without fail granted me their love and support. Their persistence in showering me with their care and support enabled me to look beyond the obstacles and strive towards the accomplishment of this work. Last but not least, I would like to thank the University of Malaya for the financial support received through the PPP Grant (PS058/2009A) and the UMRG Grant (RG065/09SUS).

TABLE OF CONTENTS

LIST OF FIGURES

Page

Page

LIST OF TABLES

LIST OF ABBREVIATIONS

A-GPC	:	Adaptive - Model Based Generalized Predictive Control
ARX	:	Auto-Regressive eXogenous
AS-GPC	:	Adaptive - Model Based Self-Tuning Generalized Predictive Control
b.p.	:	boiling point
CARIMA	:	Controlled Auto-Regressive Integrated Moving Average
CLP	:	Closed Loop Poles
CV-101	:	control valve for the reactant flow rate
CV-102	:	control valve for the coolant flow rate
DG	:	diglycerides
DMC	:	Dynamic Matrix Control
EDF	:	Exponential and Directional Forgetting algorithm
ELS	:	Extended Least Squares
EWRLS	:	Exponential Weighting Recursive Least Squares
FAME	:	Fatty Acid Methyl Ester
FOPDT	:	First Order Plus Dead Time
G	:	glyceride
GPC	:	Generalized Predictive Control
IMC	:	Internal Model Control
KOH	:	potassium hydroxide catalyst
LTI	:	Linear Time Invariant
MG	:	monoglycerides
MIMO	:	Multi Inputs Multi Outputs

MMAC	:	multiple model adaptive control
MPC	:	Model Predictive Control
MRAS	:	Model Reference Adaptive Systems
PID	:	proportional integral derivative
PLC	:	Programmable Logic Controller
PSO	:	Particle Swarm Optimization
QP	:	Quadratic Programming
RGA	:	Relative Gain Array
RLS	:	Recursive Least Squares
MeOH	:	methanol
SISO	:	Single Input Single Output
TG	:	triglycerides
TITO	:	Two Inputs Two Outputs
VFF-RLS	:	Variable Forgetting Factor Recursive Least Squares
ZN	:	Ziegler-Nichols

LIST OF SYMBOLS

Alphabetical symbols:

$\boldsymbol{a}\left(z^{-1}\right)$	:	polynomial matrix associated with the outputs of the ARX/CARIMA model
$\boldsymbol{a}_i$	:	i-th coefficient matrix of $\boldsymbol{a}\left(z^{-1}\right)$
$\boldsymbol{b}\left(z^{-1}\right)$	:	polynomial matrix associated with the inputs of the ARX/CARIMA model
$\boldsymbol{b}_i$	:	i-th coefficient matrix of $\boldsymbol{b}\left(z^{-1}\right)$
C	:	design constant for the VFF-RLS algorithm
$\boldsymbol{C_A}$	:	matrix associated with $\underset{\rightarrow k}{\boldsymbol{y}}$ in the GPC prediction equation
C_{AO}	:	initial concentration of methanol
$\boldsymbol{C_b}$	:	matrix associated with $\Delta \underset{\rightarrow k-1}{\boldsymbol{u}}$ in the GPC prediction equation
$CEIL[\bullet]$	:	nearest next integer to $[\bullet]$
C_{ME}	:	concentration of FAME
C_{TGO}	:	concentration of triglycerides
D	:	discrete dead time
$\boldsymbol{D}$	:	diagonal matrix
d_1	:	disturbance 1
d_2	:	disturbance 2
$\boldsymbol{d}_k$	:	vector of bias parameters for the ARX model at the k-th instance
f	:	scaled move suppression coefficient

F_c	:	coolant flow rate
F_o	:	reactant flow rate
$G_p(z)$	:	hypothetical discrete time transfer function
$\boldsymbol{H_A}$	:	matrix associated with $\underset{\leftarrow k}{\boldsymbol{y}}$ in the GPC prediction equation
$\boldsymbol{H_b}$	:	matrix associated with $\Delta \underset{\leftarrow k-1}{\boldsymbol{u}}$ in the GPC prediction equation
$\boldsymbol{I}$	:	identity matrix
J	:	cost function associated with the GPC
k	:	sampling instance
K_c	:	proportional gain
K_p	:	process gain
L_2	:	Euclidean norm of the parameter error
m	:	number of inputs for the ARX/CARIMA model
M	:	control horizon
n	:	number of outputs for the ARX/CARIMA model
N	:	stirrer rotational speed
N_1	:	minimum prediction horizon
N_2	:	maximum prediction horizon
$NINT[\bullet]$	:	nearest integer to $[\bullet]$
$\boldsymbol{P}$	:	covariance matrix
$\boldsymbol{P}_0$	:	initial covariance matrix
P_r	:	reactor pressure
$\boldsymbol{r}$	:	vector of setpoints
$\boldsymbol{R}$	:	vector of move suppression weights
R^2	:	coefficient of determination

R_i	:	move suppression weight for the *i*-th input
$\boldsymbol{S}$	:	upper triangular matrix called the square root of $\boldsymbol{P}$
T	:	reactor temperature
$\boldsymbol{T}(z^{-1})$	:	design polynomial matrix for the CARIMA model
T_{CO}	:	coolant inlet temperature
T_O	:	feed temperature
t_s	:	sampling time
$t_{settling}$	:	settling time of a process
u	:	process input
$\boldsymbol{U}$	:	upper triangular matrix
$\boldsymbol{u}_{max}$	:	vector of upper limits for the inputs
$\boldsymbol{u}_{min}$	:	vector of lower limits for the inputs
u_c	:	command signal
$\boldsymbol{u}_k$	:	vector of inputs for the ARX/CARIMA model at the *k*-th instance
$\boldsymbol{v}_k$	:	vector of stochastic noise variable with normal distribution and zero mean for the ARX/CARIMA model at the *k*-th instance
V_k	:	cost function at the *k*-th instance associated with the RLS algorithm
$\boldsymbol{W}$	:	vector of weights for the output residuals
W_i	:	weight for a particular the *i*-th output residual
y	:	process output
$\boldsymbol{y}_k$	:	vector of outputs for the ARX/CARIMA model at the *k*-th instance
y_m	:	reference model output

y_{sp}	:	setpoint
ΔF_c	:	coolant slew rate
ΔF_o	:	reactant slew rate
$\hat{\boldsymbol{D}}_k$	:	matrix associated with $\Delta \underset{\leftarrow k-1}{\boldsymbol{u}}$ in the unconstrained GPC solution
$\hat{\boldsymbol{D}}_k\left(z^{-1}\right)$	:	polynomial matrix of $\hat{\boldsymbol{D}}_k$
$\hat{\boldsymbol{N}}_k$	:	matrix associated with $\underset{\leftarrow k}{\boldsymbol{y}}$ in the unconstrained GPC solution
$\hat{\boldsymbol{N}}_k\left(z^{-1}\right)$	:	polynomial matrix of $\hat{\boldsymbol{N}}_k$
$\hat{\boldsymbol{P}}_k$	:	matrix associated with $\underset{\rightarrow k}{\boldsymbol{r}}$ in the unconstrained GPC solution
$\hat{\boldsymbol{P}}_k\left(z\right)$	:	polynomial matrix of $\hat{\boldsymbol{P}}_k$
$\underset{\rightarrow k}{\boldsymbol{r}}$	:	vector of future values (not including the *k*-th instance) for the setpoints
$\overline{\boldsymbol{R}}$	:	positive definite diagonal weighting matrix for the slew rates in *J*
$\underset{\rightarrow k-1}{\boldsymbol{u}}$	:	vector of future values (not including the "$k-1$"-th instance) for the inputs
$\Delta\boldsymbol{u}$	:	vector of slew rates
$\Delta\boldsymbol{u}_{\max}$	:	vector of upper limits for the slew rates
$\Delta\boldsymbol{u}_{\min}$	:	vector of lower limits for the slew rates
$\Delta \underset{\rightarrow k-1}{\boldsymbol{u}}$	:	vector of future values (not including the "$k-1$"-th instance) for the slew rates

$\Delta \underset{\leftarrow k-1}{\boldsymbol{u}}$: vector of past values (including the "$k-1$"-th instance) for the slew rates

$\overline{\boldsymbol{W}}$: positive definite diagonal weighting matrix for the output residuals in J

$\underset{\rightarrow k}{\boldsymbol{y}}$: vector of future values (not including the k-th instance) for the outputs

$\underset{\leftarrow k}{\boldsymbol{y}}$: vector of past values (including the k-th instance) for the outputs

Greek symbols:

α : order of $\boldsymbol{a}\left(z^{-1}\right)$

β : order of $\boldsymbol{b}\left(z^{-1}\right)$

$\boldsymbol{\gamma}$: Kalman gain

δ : Bittanti factor in the EDF algorithm

$\boldsymbol{\varepsilon}_i$: vector of prediction error at the i-th instance for the RLS algorithm

$\boldsymbol{\theta}_c$: controller settings in self-tuning control strategy

θ_d : continuous dead time of the process

$\boldsymbol{\theta}_p$: estimated process model parameters in the form of a LTI model

$\boldsymbol{\theta}_0$: vector of true/known process model parameters

$\hat{\boldsymbol{\theta}}_0$: initial parameter estimates for the RLS algorithm

$\hat{\boldsymbol{\theta}}$: matrix of the estimated process model parameters

$\hat{\boldsymbol{\theta}}_c$: controller settings

κ : constant for a particular tuning expression for R_i

λ_{min} : minimum forgetting factor in the EWRLS algorithm

λ : forgetting factor

$\boldsymbol{\mu}$: matrix associated with the linear inequalities

π : constant for a particular tuning expression for R_i

ρ : design constant for the EWRLS algorithm

σ : design constant for the VFF-RLS algorithm

σ_w : variance of any process measurement noise

σ_w : variance of process output measurement noise

τ_c : IMC design parameter

τ_D : derivative time constant

τ_I : integral time constant

τ_p : process time constant

$\boldsymbol{\psi}$: regressor matrix

$\boldsymbol{\Omega}$: matrix associated with the linear inequalities

$1/\Delta$: integrator in the CARIMA model

CHAPTER 1

INTRODUCTION

1.1 Background of the Research

Although simple conventional controllers with fixed controller settings (*e.g.* the classical PID controllers) are still the most widely implemented automation strategy in the industry, there are cases where these controllers simply fail to deliver the expected control objectives. To deal with nonlinear processes where the process dynamics are poorly understood, process and control engineers often face difficulties in selecting the appropriate controller settings (*i.e.* controller tuning parameters) for the controller. The different response characteristics involved across the operational regions for a nonlinear process make it impossible to select a single set of controller settings which can give the controller equal performance across all operational regions in the process. Furthermore, in the event of unanticipated changes occurring in the process, *e.g.* stirrer failure, sticking valves *etc.*, a controller with fixed controller settings (*e.g.* tuned for normal operations) will not be able to perform accordingly in the interest of mitigating the losses due to the technical failures. In view of these uncertainties encountered in process control, it is undoubtedly needful for a more intelligent process control scheme to be implemented, where the time-varying dynamics of the process can be accounted for in the design of the controller output. In simple terms, the controller must be able to 'adapt' itself to the changing dynamics of the process; hence the phrase 'adaptive control', to the best of the author's knowledge, have been used at least from the beginning of the 1950's, *e.g.* an American patent was issued to Caldwell (1950) on the subject of adaptive regulator.

The aforementioned challenges and difficulties encountered by process and control engineers are more so true in the case of implementing Model Predictive Controllers (MPC). For one of the well-known MPC strategies in particular, *i.e.* the Generalized Predictive Controller (GPC) (Clarke, Mohtadi, & Tuffs, 1987a, 1987b), if such a controller is to be made adaptive, two key components within the structure of the controller can be made adaptive – the GPC internal model and the GPC tuning parameters. The need for the internal model and the tuning parameters of the GPC to be made adaptive is obvious: for a nonlinear and time-varying process, it is impossible to adopt a single Linear Time Invariant (LTI) model to represent the dynamics of the process across all operational regions and at all times, hence model adaptation and / or updating of the controller tuning parameters should be considered. To achieve the first objective, *viz*. to enable model adaptation in the GPC controller, the Recursive Least Squares (RLS) algorithm is normally used to capture the dynamics of the process in the form of process model parameters of a LTI system at every time step. As the process evolves in time, the model parameters change accordingly with the changing dynamics of the process, and consequently the internal model of the GPC controller is updated with time (Clarke, Mohtadi, and Tuffs, 1987a).

As pertaining to the second objective, several authors have proposed different methods to auto-tune the MPC controller online at every control interval. Majority of these self-tuning studies conducted were implemented in the framework of Dynamic Matrix Control (DMC) (Al-Ghazzawi, Ali, Nouh, & Zafiriou, 2001; Ali & Al-Ghazzawi, 2003; Han, Zhao, & Qian, 2006; Kawai *et al.*, 2007), and a thorough review revealed that only a handful were concerned with the self-tuning of the GPC (Liu & Wang, 2000; Valencia-Palomoa & Rossiter, 2010). However, despite the scarcity of literature in specific relation to the self-tuning of the GPC controller, the self-tuning strategies

developed for the DMC are in principle applicable for the GPC. In these methods, with exception to the work done by Valencia-Palomoa and Rossiter (2010), optimization routines were used to compute the optimal set of controller tuning parameters online. In short, optimization routines were used not only for the purpose of producing optimized control moves, but also for the purpose of computing the optimal controller tuning parameters. Although these methods do not require much knowledge about the process from the control engineer to initiate the tuning procedure --- thus alleviating the pains that control engineers faced in tuning the predictive controller --- these approaches which involved the implementation of additional optimization routines were computationally demanding and mathematically involved (Garriga & Soroush, 2010). Moreover, these studies utilized a static internal model, and no conscious attempts were made to account for the nonlinearities and time-varying dynamics of the process in the model itself.

This study aims to design a GPC controller with both model adaptation and self-tuning capabilities, but with greater implementation simplicity. Although in some cases a proper nonlinear control law is needed to obtain adequate results, these are beyond the scope of this thesis. The control algorithm developed in this work (which employs the general adaptive control framework) is best suited to controlling processes with slowly time-varying process model parameters. The overall simplified schematic diagram of the proposed strategy is illustrated in Figure 1.1. The main strategy here is to restrict the self-tuning implementation solely for the tuning of the move suppression weight, which is reported by various researchers to be an effective parameter in affecting the closed loop performance of the GPC (McIntosh, Shah, & Fisher, 1991; Shridhar & Cooper, 1997a, 1997b). Since the RLS identification scheme can be easily cast in the form of a First Order Plus Dead Time (FOPDT) parameter estimation problem, the output of the

RLS algorithm can be used both for model adaptation in the GPC controller and for online auto-tuning of the move suppression weight by utilizing the easy-to-use tuning correlations as proposed by Shridhar and Cooper (1997b). Furthermore, it will be shown through simulation results that the tuning correlations, although originally designed and intended for use with unconstrained predictive controllers, yielded good results even in the constrained case.

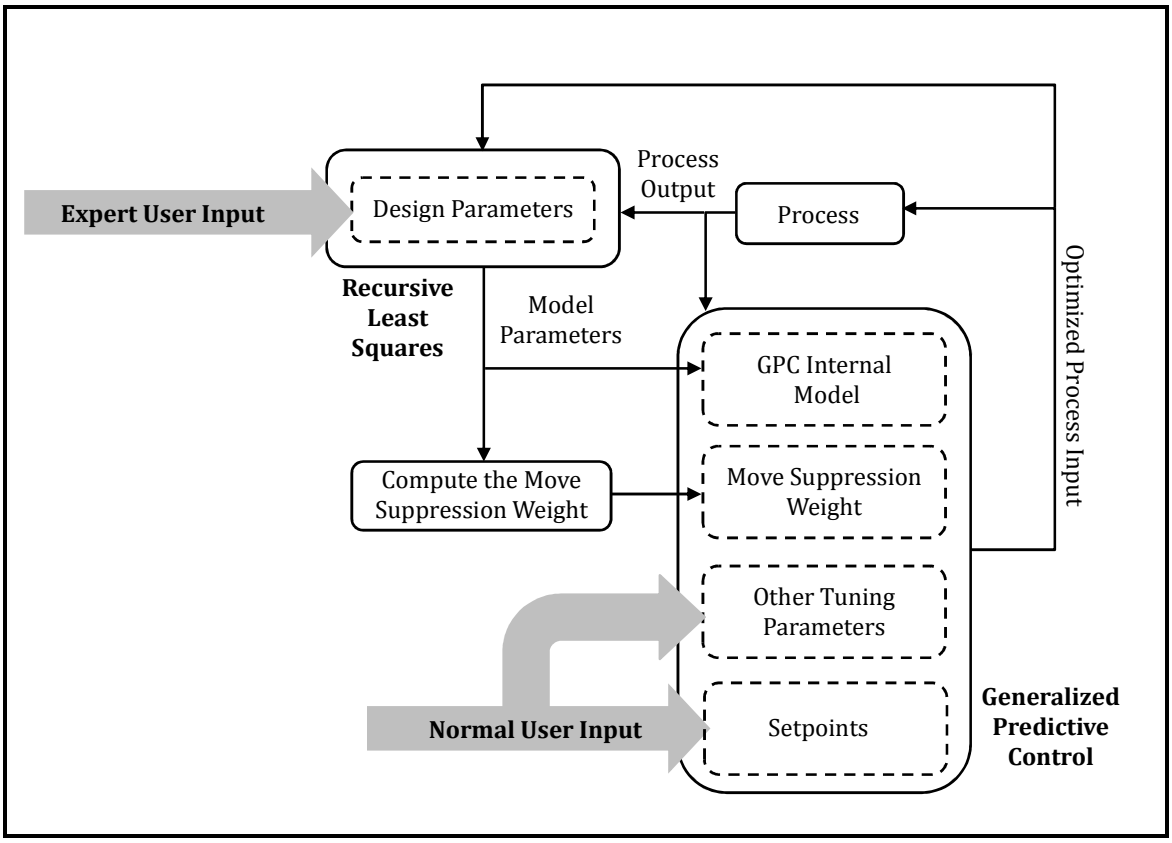

Figure 1.1: The simplified overall implementation schematic diagram of the newly proposed scheme.

As opposed to the conventional GPC strategy, where users are required to re-determine the model parameters and retune the controller manually when unsatisfactory controller performance arises, the newly proposed scheme (as illustrated in Figure 1.1) requires the users to only input a few components, *viz*. the RLS design parameters, a reduced amount of GPC tuning parameters (determined offline and held unchanged

throughout the entire course of implementation), and setpoints, while the re-modelling of the process and the retuning of the move suppression weight are taken care of by the controller itself. In practice, a normal user (*e.g*. operators and technicians) needs only be concerned about the values of the remaining tuning parameters and the setpoints, while the manipulation of the RLS design parameters (which normally are determined one-off) shall be reserved for expert users (*e.g*. engineers) only. In short, the proposed control algorithm relieves a normal user from the challenging efforts involved in retuning the GPC tuning parameters as well as re-modeling the process offline to combat poor controller performance arising from process nonlinearities. With these, the benefits of the predictive controller are enhanced via synergistic combination of model adaptation and self-tuning capabilities with little increase in computational cost. For ease of reference, the proposed control algorithm, which incorporates both the model adaptation and self-tuning strategies in a single controller, is referred to as the Adaptive-Model Based Self-Tuning Generalized Predictive Control (AS-GPC), while the GPC with model adaptation only (which was included for comparison purposes) is termed Adaptive-Model Based Generalized Predictive Control (A-GPC).

In this study, the AS-GPC was deployed on a validated mechanistic biodiesel (*i.e*. Fatty Acid Methyl Ester, FAME) transesterification reactor model developed by Mjalli, Lee, Kiew, and Hussain (2009). In addition, the performance of the AS-GPC scheme was benchmarked against that of the A-GPC and GPC schemes. Due to the complex set of chemical reactions as well as the complicated heat and mass transfer characteristics involved, the dynamics of the transesterification reactor is highly nonlinear. Figure 1.2 shows the simplified schematic diagram of the biodiesel production process. As in most chemical plants, the transesterification reactor is the most crucial unit operation to be controlled as it has primary effects on the quality of the

biodiesel. The overall reaction for the production of biodiesel in the transesterification reactor is shown here:

$$TG + 3MeOH \overset{KOH}{\rightleftharpoons} G + 3FAME \tag{1.1}$$

where TG = triglycerides, MeOH = methanol, G = glyceride, and KOH = potassium hydroxide catalyst. This reaction occurs as a sequence of three steps, where TG decomposes to diglycerides (DG) and monoglycerides (MG) with the production of glycerol (G) and FAME, as shown below:

$$\begin{aligned} TG + MeOH &\rightleftharpoons DG + FAME \\ DG + MeOH &\rightleftharpoons MG + FAME \\ MG + MeOH &\rightleftharpoons G + FAME \end{aligned} \tag{1.2}$$

As this work focuses on the development and deployment of the abovementioned advanced controllers on the transesterification reactor model, readers interested in the modeling of the transesterification reactor based on the reactions shown in Eqn. (1.2) are referred to the work of Mjalli, Lee, Kiew, and Hussain (2009).

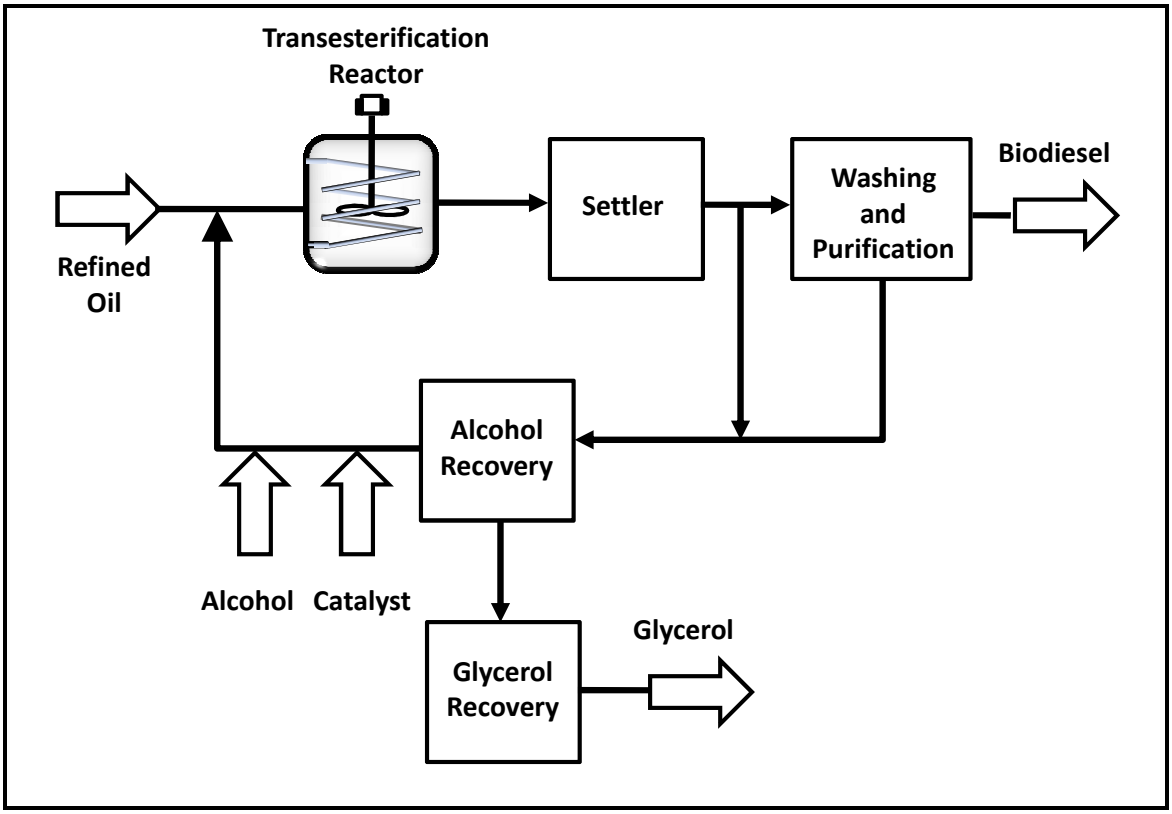

Figure 1.2: Simplified schematic diagram of a biodiesel production process.

Figure 1.3 shows the strategy by which the AS-GPC scheme was deployed on the biodiesel reactor. The A-GPC scheme used for comparison purpose in this work is a subset of the AS-GPC scheme, where $\hat{\theta}_c$ in the case of the A-GPC scheme excludes the online computed move suppression weight, indicating model adaptation only in the controller structure. The design of the reactor here is based on one of the available technologies as described in Tapasvi, Wiesenborn, and Gustafson (2004), where the biodiesel reactor is operated below the boiling point of methanol (b.p. = 64.7˚C). The pressure of the reactor is atmospheric and is not controlled. Instead, the temperature of the reactor is controlled to maximize the yield of biodiesel and to minimize the generation of unwanted by-products. Close control of the reactor temperature (*i.e.* within the range of 5 °C below the boiling point of methanol) is necessary as the rate of reaction increases with increasing reaction temperature (Leung, Wu, & Leung, 2010), but too high a temperature accelerates the saponification reaction of triglycerides (Eevera, Rajendran, & Saradha, 2009; Leung & Guo, 2006). In addition to controlling the reactor temperature, the concentration of the FAME is also controlled to ensure the stability, consistency and the quality of the biodiesel produced. This is important because the concentration of the biodiesel produced in the reactor must lie within the required specifications before proceeding to downstream processing.

To deal with this, the strategy here as illustrated in Figure 1.3 involves the deployment of two Single Input Single Output (SISO) AS-GPC control loops (*ie*. a decentralized AS-GPC strategy) to regulate the reactor temperature (T) and the FAME concentration (C_{ME}) by manipulating the reactant flow rate (F_o) and the coolant flow rate (F_c) respectively. In addition, four key variables were identified as major disturbances to the biodiesel reactor, *viz*. the feed temperature (T_O), initial concentration of triglycerides (C_{TGO}), coolant inlet temperature (T_{CO}), and stirrer rotational speed (N).

Although unable to fully account for the interactions between all variables as in the case of a centralized control structure, the decentralized control structure was chosen in this work due to its relative simplicity in design and implementation. Also, the decentralized design is used considering the nature of the tuning correlations employed in this work, which are only applicable for SISO loops.

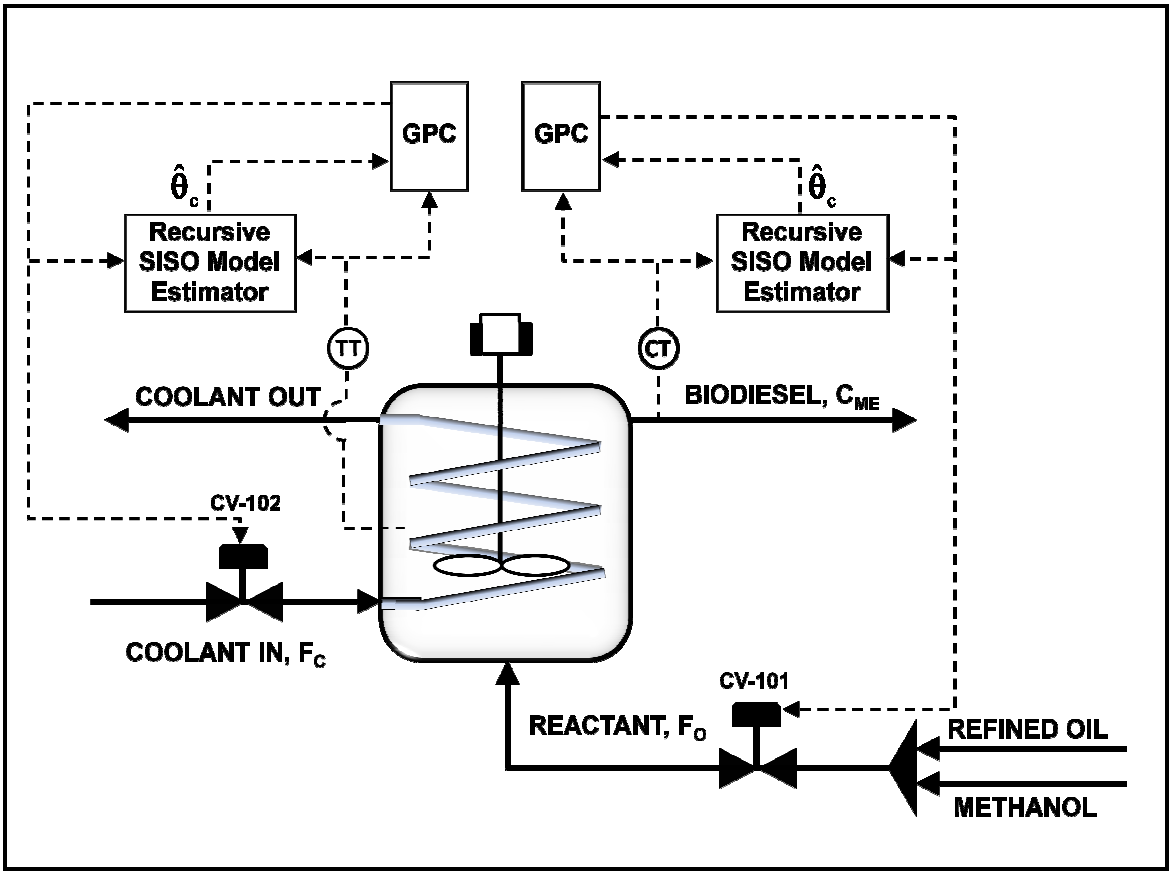

Figure 1.3: Simplified schematic of the decentralized AS-GPC scheme on the biodiesel reactor operating at atmospheric pressure. F_c is the coolant flow rate, F_o is the reactant flow rate, C_{ME} is the concentration of FAME, and $\hat{\theta}_c$ are the controller settings, *i.e.* the estimated process model parameters and the online computed move suppression weight.

1.2 Research Objectives

Following an overview of this research in the previous section, the objectives of this research are:

i) To screen diverse forms of the RLS algorithm for their strengths and weaknesses in tracking time-varying systems, and to select one for implementation.

ii) To perform an open loop dynamic system analysis on the transesterification process for the purpose of studying the nonlinearities and extent of loop interactions in the process.

iii) To perform offline system identification on the various operational regions of the transesterification process in the form of FOPDT model. The results were incorporated into the design of the AS-GPC as a backup and contingency measure.

iv) To design, code and develop the AS-GPC algorithm, where the output of the selected RLS algorithm is used not only for model adaptation in the GPC controller, but also for self-tuning of the move suppression weights by using the analytical tuning expressions proposed by Shridhar and Cooper (1997b).

v) To study the closed loop performance of the AS-GPC scheme in the constraint-free case for the transesterification process and to analyze the locations of the closed loop poles to ensure the soundness and stability of the basic unconstrained controller design.

vi) To deploy the newly proposed constrained AS-GPC algorithm on a nonlinear process, *i.e*. the transesterification process, and testing its performance in servo and regulatory control. Comparisons with the constrained A-GPC, GPC and conventional PID schemes were included where appropriate.

1.3 Structure of Thesis

The remaining six chapters are organized as follows:

a) Chapter 2 reviews the pertinent literature of this research, *i.e*. adaptive control, recursive system identification techniques, and the GPC strategy. Furthermore, recent developments in the offline and online tuning methods of the GPC were surveyed, which motivated the AS-GPC scheme. The necessary mathematical background involved in the development of the AS-GPC is also covered.

b) Chapter 3 discusses the methods used in meeting all the research objectives. In particular, the architecture of the AS-GPC scheme is elaborated.

c) Chapter 4 presents the results and discussions on the screening of the various RLS algorithms in fulfillment of objective (i), where one specific RLS algorithm was selected for implementation throughout this work.

d) Chapter 5 discusses the results of the open loop dynamic system analysis as well as the results of the offline system identification as stated by research objectives (ii) and (iii).

e) Chapter 6 is the core of this work, where objectives (iv) - (vi) are delivered. The closed loop performance of the AS-GPC was tested and simulated on a validated mechanistic transesterification reactor model. Necessary analysis and benchmarking with other control schemes were also included to demonstrate the superiority of the newly proposed scheme.

f) Chapter 7 concludes this research, and proposes future extensions.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction to Adaptive Control

The development of adaptive control was primarily motivated by the growth of the aerospace industry in the 1950's. However, these initial attempts were mostly unsuccessful (Åström, 1983). The rapid development of adaptive control strategy took place only in the 1970's, when the design of adaptive controllers was founded on more secure theoretical framework and modern control concepts. Since then, adaptive control has emerged as an active area of research. The rising need for the implementation of adaptive control in the chemical and process industry is in conjunction with the increasing complexity of modern day processes. The use of adaptive control systems was described by Seborg, Edgar, and Shah (1986) as having the capability to "offer significant potential benefits for difficult process control problems where the process is poorly understood and/ or changes in unpredictable ways". One example of the many complex processes where adaptive control has been widely applied is the polymerization reaction (Seborg, Edgar, & Shah, 1986) where many complex physicochemical phenomena are still poorly understood (Elicabe & Meira, 1988; Mendoza-Bustos, Penlidis, & Cluett, 1990). In addition to this, Seborg, Edgar, and Shah (1986) in their review on adaptive control also reported many other successful adaptive control experimental applications across a wide variety of processes (*i.e.* absorption/ desorption plants, chemical reactors, distillation columns *etc.*). Recent applications of adaptive control continue to be reported, *e.g*. Moon, Cole and Clark (2006), Khodabandeh and Bolandi (2007), Mjalli and Hussain (2009) *etc*.

The extent of coverage on the subject of adaptive control are rather widespread, with several excellent technical review papers (Åström, 1983; Åström, Borisson, Ljung, & Wittenmark, 1977; Seborg, Edgar, & Shah, 1986), books (Åström & Wittenmark, 1994; Clarke, 1981; Ioannou & Fidan, 2006), and tutorial (Isermann, 1982) available in the literature. In these resources, the various types of adaptive controllers available are documented in detail. In addition to these, Anderson and Dehghani (2008) gave a specific review on the different type of challenges present in adaptive control.

The orientation of subsequent sections in this chapter is as follows: First, the different forms of adaptive control strategies available will be addressed, with special attention being given to the self-tuning control strategy. Next, the various components of self-tuning control strategy will be elucidated in detail, *viz.* the necessary theoretical framework of the various RLS algorithms, the GPC algorithm and the various GPC tuning methods.

2.2 Different Forms of Adaptive Control

Different adaptive control strategies were developed to deal with nonlinearities in the process (Bequette, 1991; Di Marco, Semino, & Brambilla, 1997). Due to the innumerable amount of adaptive control strategies documented in the control literature, here a brief description of the more common adaptive control techniques will be given.

2.2.1 Gain Scheduling

Adaptive control in its simplest form can be implemented by having predetermined sets of controller settings for different operating points of a process

(Åström, 1983; Seborg, Edgar, & Shah, 1986). This form of adaptive control strategy, however, does not cater to processes which are time-varying with respect to a specific operating point. A simple implementation of such adaptive control strategy is the 'gain scheduling' method (Leith & Leithead, 2000; Rugh, 1991). In gain scheduling, if the process gain (K_p) changes in a predictable manner, the controller settings can be adjusted such that the product of K_p and K_c (the proportional gain) is a constant. In this case, the different values of K_p are predetermined for different operating points of the process, and interpolation is used to obtain the values of K_p in between operating points. Although the gain scheduling method is implementation-wise simple, Wong and Seborg (1986) showed that for a process with a large dead time, the standard gain scheduling based controllers exhibit poorer control of the process than conventional PID controllers. Further, it is not suitable for processes where the dynamics change with time and operational regions (*e.g.* the time constant and dead time of the process vary with time and operational regions).

2.2.2 Multiple Model Adaptive Control

The simple adaptation strategy as illustrated in Section 2.2.1 can also be extended to cater to model-based controllers. In this case, a number of local LTI models, each representing the dynamics of the process at a specific operating point are predetermined. This form of adaptive control technique is referred to as the Multiple Model Adaptive Control (MMAC). The reason for employing multiple models lies in the fact that a nonlinear process can be approximated by having multiple local LTI models (the more the better), each representing a particular operating region of the process (Banerjee, Arkun, Ogunnaike, & Pearson, 1997). However, due to the practical limits on the number of models feasible for implementation, the approximate dynamics

of the process in between the different operating points are usually obtained by some form of scheduling activity (*e.g.* interpolation). Chow, Kuznetsov, and Clarke (1998), for instance, showed that for predictive control, scheduling can be done by interpolating the poles and zeros of the corresponding local models. Gendron *et al.* (1993) in their design of a multiple model pole placement controller, weighted the process models based on the output variable. This weighted model was then used to design the pole placement controller. Townsend, Lightbody, Brown, and Irwin (1998) used the prediction error instead as a criterion for weighting the process models. All of these examples utilized the MMAC strategy to design a single controller (*i.e.* the models are interpolated and weighted and the results are implemented on a single controller). The MMAC strategy can also be implemented in another manner: the controller outputs can be interpolated and weighted instead of the process models (Dougherty & Cooper, 2003a, 2003b; Yu, Roy, Kaufman, & Bequette, 1992). In this case, multiple linear controllers are designed based on the various local LTI models which correspond to different operating points of the process. The control outputs from the various linear controllers are then weighted to produce a control output which is sent to the process. The MMAC approach, although a possible approximation of nonlinear processes, theoretically requires an infinite amount of models to accurately describe a highly nonlinear process. In reality, this is not achievable, giving process and control engineers a hard time in deciding the practical amount of models/controllers to be used. Furthermore, constructing multiple models by offline system identification is cumbersome and requires perturbation of the process from its nominal operating region, which may not be desirable during normal operations. Also, the use of the MMAC approach is restricted to systems which are nonlinear with respect to different operating regions only, and does not include systems which are non-stationary in time with respect to a certain operating region.

2.2.3 Model Reference Adaptive System

Another common adaptive control technique is the Model Reference Adaptive Systems (MRAS). This technique was first developed by Whitaker, Yamron and Kezer (1958) for control of aircrafts. Figure 2.1 shows a block diagram of a typical MRAS. In this technique, a reference model is used to specify the ideal response of the process output when subjected to a command signal (u_c). The error between the reference model output (y_m) and the real process output (y), which is an indicator of how the real process output deviates from the ideal response, is then used to drive the adjustment mechanism which determines how the controller parameters are changed. In the context of the MRAS, the adjustment mechanism refers to the MIT rule (Osbourne, Whitaker, & Kezer, 1961; Whitaker, 1959). The associated challenges in implementing this rule (*e.g.* the issue of instability) are given in the recent review of Anderson and Dehghani (2008).

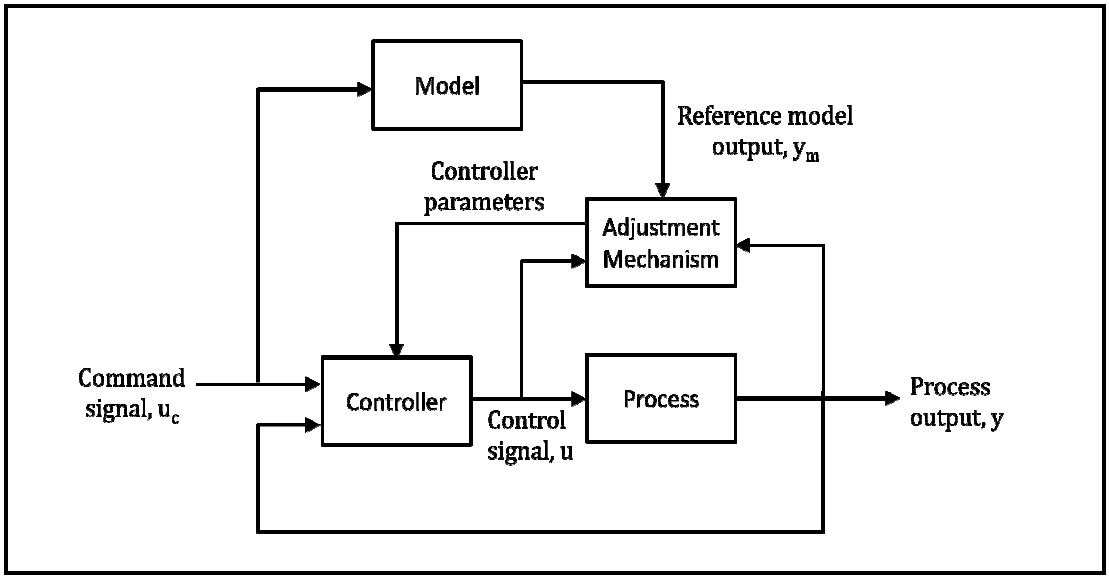

Figure 2.1: Block diagram of a typical Model Reference Adaptive Systems (MRAS) (adapted from Åström, 1983): u_c is the command signal, u is the process input, y is the process output, and y_m is the reference model output.

2.2.4 Self-tuning Control

Among the different forms of adaptive control strategies available, the self-tuning approach has received the most attention in the past decades (Åström, Borisson, Ljung, & Wittenmark, 1977; Seborg, Edgar, & Shah, 1986; Shah & Cluett, 1991). The basic idea behind the self-tuning approach is to produce a controller which is able to retune itself in real time in order to suit the changing dynamics of the process. Figure 2.2 shows the block diagram of the general self-tuning control framework, where u = process input, y = process output, d_1 and d_2 = disturbances, and y_{sp} = setpoint. From the figure, given the input and the output of the process, a recursive parameter estimator is used to capture the dynamics of the process online by means of estimating the process model parameters ($\boldsymbol{\theta}_p$) in the form of a LTI model recursively. Subsequently, based on the preferred choice of control law, $\boldsymbol{\theta}_p$ is then used to calculate the appropriate controller settings ($\boldsymbol{\theta}_c$) which caters to the most recent process dynamics. This sequence of estimating the process model parameters and retuning of the controller occurs in real time for as long as the process is in operation, which makes this control strategy particularly attractive for controlling difficult time-varying and nonlinear process. Hence, the success of self-tuning control in various processes is evident (Ahlberg & Cheyne, 1976; Bengtsson & Egardt, 1985; Buchholt & Kümmel, 1979; Clarke & Gawthrop, 1981; Corrêa, Corrêa, & Freire, 2002; Ertunc, Akay, Boyacioglu, & Hapoglu, 2009; Hallager, Goldschmidt, & Jorgensen, 1984; Harris, MacGregor, & Wright, 1978; Ho, Mjalli, & Yeoh, 2010a, 2010b; Hodgson & Clarke, 1982; Khodabandeh & Bolandi, 2007; Kwalik & Schork, 1985; McDermott, 1984; Moon, Clark, & Cole, 2005; Moon, Cole, and Clark, 2006; Tingdahl, 2007).

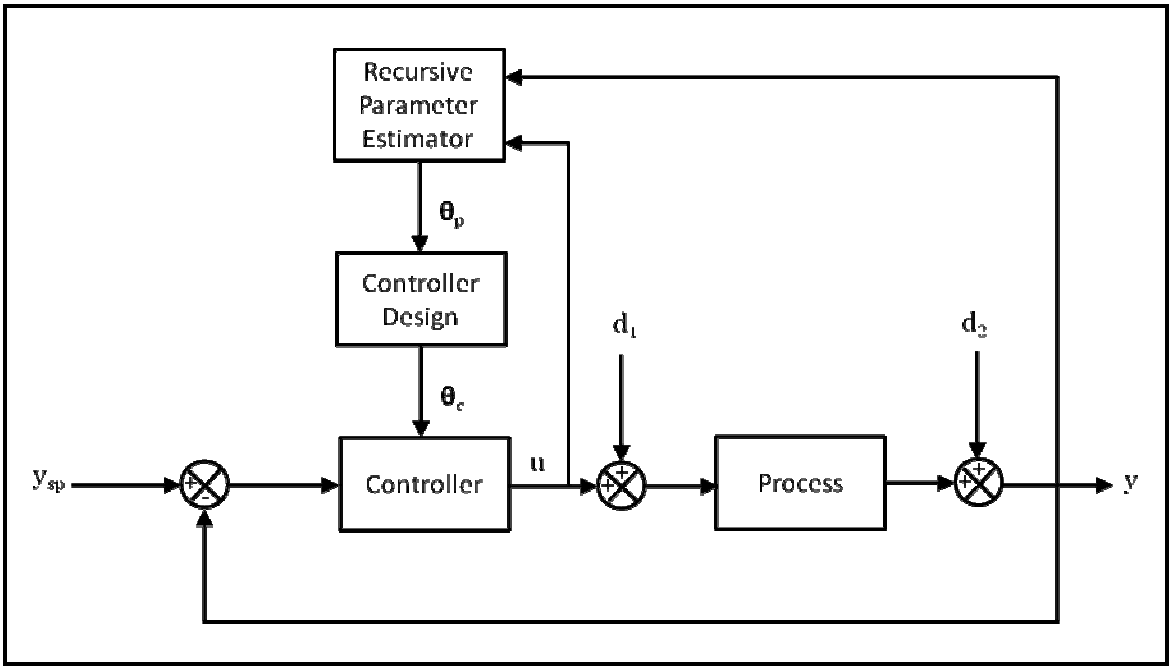

Figure 2.2: Block diagram of the general self-tuning control framework: u = process input, y = process output, d_1 and d_2 = disturbances, y_{sp} = setpoint, $\boldsymbol{\theta}_p$ = estimated process model parameters, and $\boldsymbol{\theta}_c$ = controller settings

In the self-tuning control framework, the recursive parameter estimator is a crucial component for capturing the dynamics of the process. Ljung and Söderstöm (1983) provided an in-depth theoretical treatment on the subject of recursive system identification, which is the basic building block for constructing the recursive parameter estimator. Among the many recursive system identification techniques available in the literature, the RLS algorithm (Ljung, 1987; Ljung & Gunnarsson, 1990; Ljung & Söderstöm, 1983) is the most popular parameter estimation technique used in self-tuning adaptive control due to its simplicity and rapid convergence when properly applied (Seborg, Edgar, & Shah, 1986; Shah & Cluett, 1991). Since this work focuses mainly on the self-tuning control strategy, unless mentioned otherwise, the terms 'adaptive control' and 'self-tuning control' will be used interchangeably hereafter.

Other forms of adaptive control strategies (*e.g.* gain scheduling, MRAS *etc.*) will be referred to their respective terms as described previously.

2.3 Recursive System Identification Techniques

As alluded to previously in Section 2.2.4, one of the key components in self-tuning control is the recursive parameter estimation, *viz.* the online process modeling. This technique seeks to overcome the shortcomings of the MMAC approach in approximating the true behavior of a process by estimating the process model parameters of a system recursively in real time. Such an approach is not only equivalent to having a huge models bank with infinite amount of local linear models (without the penalty of exorbitant efforts in constructing these models offline), but also is capable of dealing with time-varying processes.

Among the many recursive identification algorithms available in the literature (Ljung, 1987; Ljung & Söderstöm, 1983), Seborg, Edgar, and Shah (1986) in his comprehensive review on self-tuning control concluded that the RLS algorithm and the Extended Least Squares (ELS) algorithm are the two most frequently employed parameter estimation techniques in adaptive control. However, the RLS algorithm is more popular due to its simplicity and fast convergence when properly applied (Seborg, Edgar, & Shah, 1986). The following subsections are dedicated to address the various theoretical aspects of the RLS algorithms, including the efforts of various researchers in improving the properties of the RLS algorithm.

2.3.1 Model Structure

The local dynamics of a general multivariable process can be represented by a Multi Inputs Multi Outputs (MIMO) discrete time Auto-Regressive eXogenous (ARX) model. Consider a MIMO discrete time ARX model with m inputs ($\boldsymbol{u}_k$), n outputs ($\boldsymbol{y}_k$), a bias parameter ($\boldsymbol{d}_k$) and a stochastic noise variable with normal distribution and zero mean ($\boldsymbol{v}_k$):

$$\boldsymbol{a}\left(z^{-1}\right)\boldsymbol{y}_k = \boldsymbol{b}\left(z^{-1}\right)\boldsymbol{u}_{k-D} + \boldsymbol{d}_k + \boldsymbol{v}_k \tag{2.1}$$

where $\boldsymbol{a}$[n×n] and $\boldsymbol{b}$[n×m] are polynomial matrices in the z-domain given as:

$$\boldsymbol{a}\left(z^{-1}\right) = \boldsymbol{I} + \sum_{i=1}^{\alpha} \boldsymbol{a}_i z^{-i} \tag{2.2}$$

$$\boldsymbol{b}\left(z^{-1}\right) = \sum_{i=1}^{\beta} \boldsymbol{b}_i z^{-i} \tag{2.3}$$

In Eqn. (2.1), the subscript k is a nonnegative integer which denotes the sampling instance, (k = 0, 1, 2...), $D \geq 0$ is the known dead time of the process expressed as an integer multiple of the sampling time (t_s), whereas α and β in Eqns. (2.2) - (2.3) are known positive integers and $\boldsymbol{I}$ is the identity matrix. The form of the process model in Eqn. (2.1) can be easily simplified to cater to the SISO case by setting $n = m = 1$.

It is to be noted that the integers α and β represent the orders of the respective output and input associated polynomials, where having large values of α and β is equivalent to increasing the order of the model. In recursive parameter estimation, having a higher order process model implies an increase in the number of parameters to be estimated, which introduces additional computational burden. On the other hand, having a model order that is too low may not adequately describe the dynamics of the

process. Typical values of α and β recommended by Seborg, Edgar, and Shah (1986) are α = 2 or 3 and β = 2 or 3. Alternatively, choosing a valid *D* value and α = β = 1 would give a FOPDT ARX model, which in many case serves as a good approximation for the purpose of control system design (Seborg, Edgar, & Mellichamp, 2004).

As in all discrete time systems, there is a loss of dynamic information when a continuous process is subjected to sampling operation. Therefore, the sampling time of a discrete time system must be carefully selected to minimize the loss of dynamic information. While having a large sampling time causes slow controller action due to the inadequately sampled data, selecting a small value of sampling time may cause excessive control action and the possibility of the process model exhibiting non-minimum phase behavior (Åström, Hagander, & Sternby, 1984). One possible rule of thumb for selecting the value of sampling time was proposed by Åström and Wittenmark (1997), where the sampling time is selected in terms of the settling time (t_{settling}) of the process:

$$\frac{t_{\text{settling}}}{15} \le t_s \le \frac{t_{\text{settling}}}{6} \tag{2.4}$$

For FOPDT systems, a good rule of thumb for selecting the sampling time is to select it such that it is approximately one tenth of the process time constant, *i.e.* $t_s \approx 0.1\tau_p$ (Seborg, Edgar, & Shah, 1986). Other rule of thumbs are also available in the literature, *e.g.* Middleton (1991).

2.3.2 Recursive Least Squares Algorithm

To capture the dynamics of a slowly time-varying process, RLS algorithm is used for online estimation of the coefficient matrices in $\boldsymbol{a}(z^{-1})$ and $\boldsymbol{b}(z^{-1})$ in Eqn. (2.1).

With the data of the inputs and the outputs of the process constantly being fed to the RLS algorithm, the RLS seeks to minimize a weighted cost function, V of the form:

$$V_k = \sum_{i=1}^{k} \lambda^{k-i} \left\| \boldsymbol{\varepsilon}_i \right\|_2^2 \tag{2.5}$$

where $\lambda \in (0,1]$ is the forgetting factor, and $\boldsymbol{\varepsilon}_i \in \Re^n$ is the vector of prediction error at the i-th instance.

Many variants of the RLS algorithm are reported in the literature (Fortescue, Kershenbaum, & Ydstie, 1981; Kulhavy & Karny, 1985; Mikleš & Fikar, 2007; Park, Jun, & Kim, 1991; Rao Sripada & Fisher, 1987; Salgado, Goodwin, & Middleton, 1988; Shah & Cluett, 1991), with the aim of improving the tracking performance of the conventional RLS algorithm (Ljung, 1987; Ljung & Söderstöm, 1983). The conventional form of the RLS equations is shown here for reference (where $\lambda = 1$):

$$\boldsymbol{\varepsilon}_k = \boldsymbol{y}_k - \hat{\boldsymbol{\theta}}_{k-1}^T \boldsymbol{\psi}_k \tag{2.6}$$

$$\boldsymbol{\gamma}_k = \frac{\boldsymbol{P}_{k-1}\boldsymbol{\psi}_k}{\lambda + \boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1}\boldsymbol{\psi}_k} \tag{2.7}$$

$$\boldsymbol{P}_k = \frac{1}{\lambda}\left[\boldsymbol{P}_{k-1} - \frac{\boldsymbol{P}_{k-1}\boldsymbol{\psi}_k\boldsymbol{\psi}_k^T\boldsymbol{P}_{k-1}}{\lambda + \boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1}\boldsymbol{\psi}_k} \right] \tag{2.8}$$

$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \boldsymbol{\gamma}_k \boldsymbol{\varepsilon}_k \tag{2.9}$$

In these equations, $\boldsymbol{\gamma}$ is the Kalman gain and $\boldsymbol{P}$ is the covariance matrix of the prediction error. The regressor matrix, $\boldsymbol{\psi}$ and the matrix of the estimated process model parameters, $\hat{\boldsymbol{\theta}}$ (which in essence is the explicit form of $\boldsymbol{\theta}_p$ mentioned in Figure 2.2) are represented by:

$$\boldsymbol{\psi}^T = \left[-\boldsymbol{y}_{k-1}^T, ..., -\boldsymbol{y}_{k-\alpha}^T, \boldsymbol{u}_{k-D-1}^T, ..., \boldsymbol{u}_{k-D-\beta}^T, \boldsymbol{1} \right] \tag{2.10}$$

$$\hat{\boldsymbol{\theta}}^T = \left[\boldsymbol{a}_1, ..., \boldsymbol{a}_\alpha, \boldsymbol{b}_1, ..., \boldsymbol{b}_\beta, \boldsymbol{d} \right] \tag{2.11}$$

In the conventional RLS algorithm, the forgetting factor, λ is kept constant at unity. This form of the RLS algorithm is best suited for identifying the process model parameters of a LTI system only, and when implemented on time-varying systems, the algorithm loses its adaptivity in the long run. To overcome the shortcomings of the conventional RLS algorithm, a constant forgetting factor of $\lambda \in (0,1]$ is employed instead to weigh down older data as the most recent data are more important in a slowly time-varying environment (Ljung, 1987; Ljung & Söderstöm, 1983; Mikleš & Fikar, 2007; Shah & Cluett, 1991). However, it is difficult to select a proper value of the forgetting factor as too small a value would cause the parameter estimates to be very uncertain (*i.e.* $\boldsymbol{P}_k$ and $\boldsymbol{\gamma}_k$ becomes large), whereas a large forgetting factor would cause the algorithm to be insensitive to process parameter changes. With regards to this problem, Fortescue, Kershenbaum, and Ydstie (1981) proposed the use of a time-varying forgetting factor, where the forgetting factor is varied according to the changes in the prediction error. In addition to the time-varying forgetting factor scheme, Cordero and Mayne (1981) suggested an additional mechanism to ensure that the trace of the covariance matrix remains bounded even when there is no new information coming into the RLS algorithm. This algorithm is referred to as the Variable Forgetting Factor Recursive Least Squares (VFF-RLS) and is given here:

$$\boldsymbol{\varepsilon}_k = \boldsymbol{y}_k - \hat{\boldsymbol{\theta}}_{k-1}^T \boldsymbol{\psi}_k \tag{2.12}$$

$$\boldsymbol{\gamma}_k = \frac{\boldsymbol{P}_{k-1}\boldsymbol{\psi}_k}{1+\boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1}\boldsymbol{\psi}_k} \tag{2.13}$$

$$\lambda_k = 1 - \frac{\boldsymbol{\varepsilon}_k^T \boldsymbol{\varepsilon}_k}{\sigma\left[1+\boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1}\boldsymbol{\psi}_k\right]} \tag{2.14}$$

$$\boldsymbol{\omega}_k = \boldsymbol{P}_{k-1} - \boldsymbol{\gamma}_k \boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1} \tag{2.15}$$

$$\boldsymbol{P}_k = \begin{cases} \boldsymbol{\omega}_k / \lambda_k & \text{if trace of } \boldsymbol{\omega}_k / \lambda_k \le C \\ \boldsymbol{\omega}_k & \text{otherwise} \end{cases} \tag{2.16}$$

$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \boldsymbol{\gamma}_k \boldsymbol{\varepsilon}_k \tag{2.17}$$

where σ/σ_w is selected to be a large number ≈ 1000 (σ_w is the variance of any process output measurement noise), while C is a design constant. From Eqn. (2.14), when the prediction error is small, $\lambda_k \rightarrow 1$. On the other hand, when the prediction error is huge, λ_k is automatically adjusted to a smaller value ($\lambda_k < 1$). The VFF-RLS scheme has excellent performance in tracking time-varying process model parameters and was successfully implemented on a chip refiner and a paper box machine (Ydstie, Kershenbaum, & Sargent, 1985).

A similar approach was adopted to vary the value of the forgetting factor according to the prediction error of the RLS algorithm in the work of Park, Jun, and Kim (1991), where the equations of the forgetting factor are shown here:

$$\lambda_k = \lambda_{\min} + \left(1 - \lambda_{\min}\right) \cdot 2^{L_k} \tag{2.18}$$

$$L_k = -NINT\left[\rho \boldsymbol{\varepsilon}_k^T \boldsymbol{\varepsilon}_k\right] \tag{2.19}$$

In Eqns. (2.18) - (2.19), $\lambda_{\min}$ is the minimum forgetting factor, $NINT[\bullet]$ is defined as the nearest integer to $[\bullet]$, and ρ is a design parameter. Equations (2.6) - (2.9), with (2.18) and (2.19) are referred to as the Exponential Weighting Recursive Least Squares (EWRLS), following Park, Jun, and Kim (1991).

Another interesting approach in identifying time-varying process parameters is to forget only in the direction where new information is coming in (Ljung & Gunnarsson, 1990). This approach was demonstrated by Kulhavý and Kárný (1985) in the Exponential and Directional Forgetting (EDF) algorithm. To speed up convergence in the EDF algorithm, Bittanti, Bolzern, and Campi (1990) proposed an addition of a correction factor referred to as the Bittanti factor to the equation of covariance update. The modified EDF algorithm is shown here:

$$\boldsymbol{\varepsilon}_k = \boldsymbol{y}_k - \hat{\boldsymbol{\theta}}_{k-1}^T \boldsymbol{\psi}_k \tag{2.20}$$

$$r_k = \boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1} \boldsymbol{\psi}_k \tag{2.21}$$

$$\boldsymbol{\gamma}_k = \frac{\boldsymbol{P}_{k-1}\boldsymbol{\psi}_k}{1+r_k} \tag{2.22}$$

$$\beta_k = \begin{cases} \lambda - \dfrac{1-\lambda}{r_k} & \text{if } r_k > 0 \\ 1 & \text{if } r_k = 0 \end{cases} \tag{2.23}$$

$$\boldsymbol{P}_k = \boldsymbol{P}_{k-1} - \frac{\boldsymbol{P}_{k-1}\boldsymbol{\psi}_k \boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1}}{\beta_k^{-1} + r_k} + \delta \boldsymbol{I} \tag{2.24}$$

$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \boldsymbol{\gamma}_k \boldsymbol{\varepsilon}_k \tag{2.25}$$

where $\delta \in [0, 0.01]$ is the Bittanti factor. The value of the Bittanti factor used should be small to avoid over-sensitivity in the EDF algorithm, which could lead to erroneous parameter estimates.

2.3.3 Factorization of the Covariance Matrix

As the various equations of covariance update, *i.e*. Eqns. (2.8), (2.16) and (2.24), involve subtraction operations, round-up errors due to the limited accuracy of the computer, there is a possibility that the covariance matrix becomes non positive definite. The resulting instability caused by a non positive definite covariance matrix will affect the performance of the RLS algorithm significantly. Hence, to attain numerical stability during implementations of the RLS algorithms, the positive definite feature of the covariance matrix must be retained during recursions. One way to overcome this problem is to use the square root filtering technique (Potter & Stern, 1963), where the covariance matrix is factored as $\boldsymbol{P} = \boldsymbol{S}\boldsymbol{S}^{\mathrm{T}}$ (in which $\boldsymbol{S}$ is an upper triangular matrix called the square root of $\boldsymbol{P}$) and subsequent updates of $\boldsymbol{P}$ be accomplished through the factorized component $\boldsymbol{S}$. Alternatively, Bierman (1976) suggested to use the

factorization of $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$, where $\boldsymbol{U}$ is an upper triangular matrix and $\boldsymbol{D}$ is a diagonal matrix. As in the square root filtering technique, instead of updating the covariance matrix directly, the factorized components (*i.e.* $\boldsymbol{U}$ and $\boldsymbol{D}$) of the covariance matrix are updated instead. Both methods described above are useful for achieving numerical stability in RLS implementations (Ljung, 1987; Seborg, Edgar, & Shah, 1986), and the choice of factorization method is a matter of preference. In this work, Bierman's $\boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{\mathrm{T}}$ factorization method is used. Due to the amount of details involved (omitted in the original paper), the rigorous derivation of Bierman's method is shown in Appendix A.

2.4 Generalized Predictive Control Strategy

The development of modern control concepts in the past decades has led to the emergence of the MPC technology, where an explicit process model is employed within the control algorithm to predict the future behavior of the process response at every time step over a given prediction horizon. An overall concept of the MPC strategy is as illustrated in Figure 2.3. The predicted future output trajectories are then used to compute the optimal sequence of the future input trajectories over a specified control horizon by minimizing the error between the desired future setpoints and the predicted future outputs. From the optimal input sequence, only the first move is eventually implemented in the process. This sequence of calculation is repeated at every sampling time step.

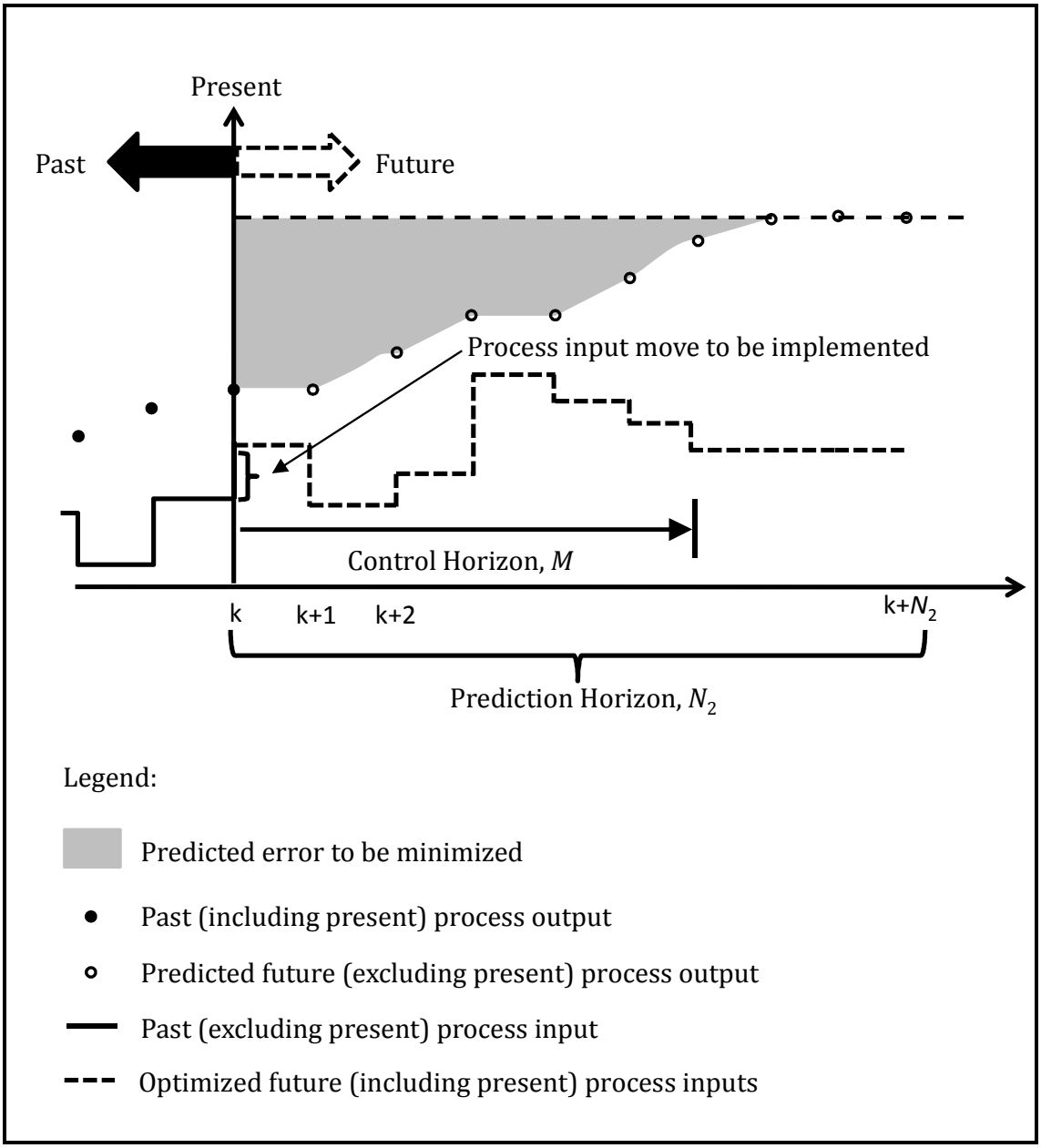

Figure 2.3: The basic concept of Model Predictive Control (MPC).

Excellent reviews and books on the development of various model predictive controllers are available in the literature (Camacho & Bordons, 1999; Garcia, Prett, & Morari, 1989; Qin & Badgwell, 2003; Rossiter, 2003). The different variants of the model predictive controllers in principle share the same basic concept as illustrated in Figure 2.3, with minor differences in the modeling/prediction assumptions involved. Among the many model predictive controllers available in the literature as well as in the industry (*e.g.* Dynamic Matrix Control, Quadratic Dynamic Matrix Control *etc.*), the

GPC devised by Clarke, Mohtadi, and Tuffs (1987a, 1987b) has attained the status of being one of the most popular MPC algorithms (Clarke, 1988; Garriga & Soroush, 2010). Originally developed for the purpose of adaptive control (in terms of model adaptation without real time adjustment of the tuning parameters), the GPC is known to be a "general purpose" algorithm capable of handling the following process control scenarios (Clarke, Mohtadi, & Tuffs, 1987a) all in one algorithm:

- A non-minimum phase plant
- An open loop unstable plant or plant with badly damped poles
- A plant with unknown order
- A plant with unknown or variable dead time

These properties of the GPC are of significant importance particularly in adaptive control applications when the plant order and dead time are usually unknown. To further illustrate the suitability of implementing the GPC algorithm for adaptive control purposes, Qin and Badgwell (2003) in their excellent review on the practical real world applications of the MPC controllers categorized the GPC under the category of "adaptive MPC algorithms". In addition, the GPC is proven to show good performance properties and a certain degree of robustness when implemented for the purpose of adaptive control (Clarke, Mohtadi, & Tuffs, 1987a). Figure 2.4 presents the simplified block diagram of the GPC strategy to aid understanding of the subsequent technical discussions. In GPC, the Controlled Auto-Regressive Integrated Moving Average (CARIMA) model is employed for prediction:

$$\boldsymbol{a}\left(z^{-1}\right)\boldsymbol{y}_k = \boldsymbol{b}\left(z^{-1}\right)\boldsymbol{u}_{k-D} + \frac{\boldsymbol{T}\left(z^{-1}\right)\boldsymbol{v}_k}{\Delta} \tag{2.26}$$

In Eqn. (2.26), $\boldsymbol{T}(z^{-1})$ is a design polynomial matrix (Clarke, Mohtadi, & Tuffs, 1987a, 1987b; Yoon & Clarke, 1995), and $\Delta = 1 - z^{-1}$. When $\boldsymbol{T}(z^{-1}) = \boldsymbol{I}$, the CARIMA model

takes the form of an ARX model with an integrated white noise term for modeling disturbances. Model adaptation in the GPC algorithm is done by having the coefficient matrices of $\boldsymbol{a}(z^{-1})$ and $\boldsymbol{b}(z^{-1})$ updated recursively by the RLS algorithm.

To facilitate explanation on the theoretical framework of the GPC algorithm, the symbols and variables used hereafter in this text will follow the conventions developed by Rositter (2003). First, the vector of future and past variables at sampling instance k are defined as:

$$\underset{\rightarrow k}{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{y}_{k+1} \\ \boldsymbol{y}_{k+2} \\ \vdots \\ \boldsymbol{y}_{k+N_2} \end{bmatrix}; \underset{\leftarrow k}{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{y}_{k} \\ \boldsymbol{y}_{k-1} \\ \vdots \\ \boldsymbol{y}_{k-\alpha} \end{bmatrix}; \underset{\rightarrow k-1}{\Delta \boldsymbol{u}} = \begin{bmatrix} \Delta\boldsymbol{u}_{k} \\ \Delta\boldsymbol{u}_{k+1} \\ \vdots \\ \Delta\boldsymbol{u}_{k+M-1} \end{bmatrix};$$

$$\underset{\leftarrow k-1}{\Delta \boldsymbol{u}} = \begin{bmatrix} \Delta\boldsymbol{u}_{k-1} \\ \Delta\boldsymbol{u}_{k-2} \\ \vdots \\ \Delta\boldsymbol{u}_{k-(D+\beta)+1} \end{bmatrix}; \underset{\rightarrow k-1}{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{u}_{k} \\ \boldsymbol{u}_{k+1} \\ \vdots \\ \boldsymbol{u}_{k+M-1} \end{bmatrix}; \underset{\rightarrow k}{\boldsymbol{r}} = \begin{bmatrix} \boldsymbol{r}_{k+1} \\ \boldsymbol{r}_{k+2} \\ \vdots \\ \boldsymbol{r}_{k+N_2} \end{bmatrix} \tag{2.27}$$

where the notation of arrows pointing right is used for strictly future (not including current value) vectors and the notation of arrows pointing left for past (including current value) vectors, $\Delta\boldsymbol{u}$ is the vector of change in the input variables (*i.e.* the slew rate), in which the elements are defined as $\Delta\boldsymbol{u}_i = \boldsymbol{u}_i - \boldsymbol{u}_{i\text{-}1}$, and $\boldsymbol{r}$ is the vector of set points. The positive integers N_2 and M are the maximum prediction horizon and control horizon respectively, which serve as tuning parameters for the GPC.

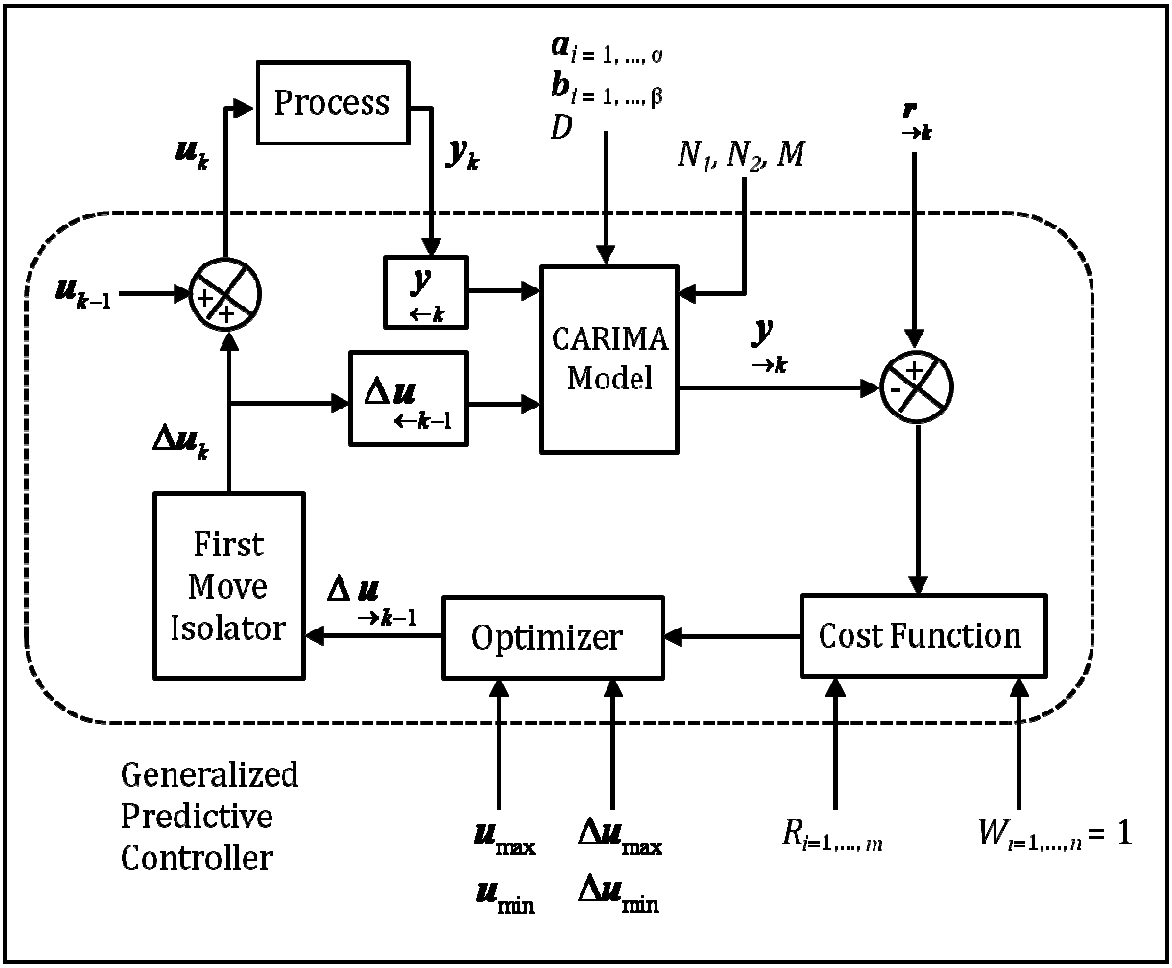

Figure 2.4: Simplified block diagram of the conventional GPC strategy. Meaning of symbols: $\boldsymbol{y}_k$ = process outputs, $\underset{\rightarrow k}{\boldsymbol{y}}$ = predicted future outputs, $\underset{\leftarrow k}{\boldsymbol{y}}$ = past outputs, $\boldsymbol{u}_k$ = process inputs, $\boldsymbol{u}_{k-1}$ = process inputs at previous instance, $\Delta\boldsymbol{u}_k$ = first slew rates from the optimized future slew rates, $\Delta\underset{\rightarrow k-1}{\boldsymbol{u}}$ = optimized future slew rates, $\Delta\underset{\leftarrow k-1}{\boldsymbol{u}}$ = past slew rates, $\underset{\rightarrow k}{\boldsymbol{r}}$ = future setpoints, $\boldsymbol{u}_{max}$ = upper limit of input constraints, $\boldsymbol{u}_{min}$ = lower limit of input constraints, $\Delta\boldsymbol{u}_{max}$ = upper limit of slew rate constraints, $\Delta\boldsymbol{u}_{min}$ = lower limit of slew rate constraints, N_1 = minimum prediction horizon, N_2 = maximum prediction horizon, M = control horizon, $R_{i=1,\ldots,m}$ = move suppression weights, $W_{i=1,\ldots,n}$ = weights on the output residuals, $\boldsymbol{a}_{i=1,\ldots,\alpha}$ and $\boldsymbol{b}_{i=1,\ldots,\beta}$ = coefficients of the polynomial matrix in the CARIMA model, D = discrete dead time.

To arrive at the form of model that is useful for predicting the future behavior of the process in the GPC algorithm, the CARIMA model shown in Eqn. (2.26) must first be written in the following form:

$$\boldsymbol{A}\left(z^{-1}\right)\left[\frac{\boldsymbol{y}_k}{\boldsymbol{T}\left(z^{-1}\right)}\right]=\boldsymbol{b}\left(z^{-1}\right)\left[\frac{\Delta\boldsymbol{u}_{k-D}}{\boldsymbol{T}\left(z^{-1}\right)}\right]+\boldsymbol{v}_k \tag{2.28}$$

where $\boldsymbol{A}\left(z^{-1}\right)=\boldsymbol{a}\left(z^{-1}\right)\Delta=\boldsymbol{I}+\boldsymbol{A}_1 z^{-1}+\ldots+\boldsymbol{A}_{\alpha+1}z^{-\alpha-1}$.

Since white noise $\boldsymbol{v}_k$ has zero mean and can be assumed zero in the future (Rossiter, 2003), with $\boldsymbol{T}(z^{-1}) = \boldsymbol{I}$, Eqn. (2.28) can be written for N_2 steps ahead into the future as shown:

$$\boldsymbol{C}_A \underset{\rightarrow k}{\boldsymbol{y}}+\boldsymbol{H}_A \underset{\leftarrow k}{\boldsymbol{y}}=\boldsymbol{C}_b\Delta\underset{\rightarrow k-1}{\boldsymbol{u}}+\boldsymbol{H}_b\Delta\underset{\leftarrow k-1}{\boldsymbol{u}} \tag{2.29}$$

The constructions of $\boldsymbol{C}_A$, $\boldsymbol{H}_A$, $\boldsymbol{C}_b$, and $\boldsymbol{H}_b$ matrices are given in Appendix B. Equation (2.29) can then be rearranged to arrive at the following formulation for prediction:

$$\underset{\rightarrow k}{\boldsymbol{y}}=\boldsymbol{H}\Delta\underset{\rightarrow k-1}{\boldsymbol{u}}+\boldsymbol{K}\Delta\underset{\leftarrow k-1}{\boldsymbol{u}}+\boldsymbol{Q}\underset{\leftarrow k}{\boldsymbol{y}} \tag{2.30}$$

where $\boldsymbol{H}=\boldsymbol{C}_A^{-1}\boldsymbol{C}_b$, $\boldsymbol{K}=\boldsymbol{C}_A^{-1}\boldsymbol{H}_b$, and $\boldsymbol{Q}=-\boldsymbol{C}_A^{-1}\boldsymbol{H}_A$. Suffice to note here that $\boldsymbol{H}\in\Re^{(N_2-N_1)\times M}$, $\boldsymbol{K}\in\Re^{(N_2-N_1)\times(D+\beta-1)}$, and $\boldsymbol{Q}\in\Re^{(N_2-N_1)\times(\alpha+1)}$. In this equation, the minimum prediction horizon N_1 is introduced, and consequently the value of the control horizon M should be lesser or equal to $(N_2 - N_1)$ in order for Eqn. (2.30) to be valid.

To compute the optimal sequence of future input trajectories, the GPC control law can be obtained by considering the following cost function:

$$J=\left(\underset{\rightarrow k}{\boldsymbol{r}}-\underset{\rightarrow k}{\boldsymbol{y}}\right)^T\overline{\boldsymbol{W}}\left(\underset{\rightarrow k}{\boldsymbol{r}}-\underset{\rightarrow k}{\boldsymbol{y}}\right)+\Delta\underset{\rightarrow k-1}{\boldsymbol{u}}^T\overline{\boldsymbol{R}}\Delta\underset{\rightarrow k-1}{\boldsymbol{u}} \tag{2.31}$$

In Eqn. (2.31), $\overline{\boldsymbol{W}}\in\Re^{[n\times(N_2-N_1)]\times[n\times(N_2-N_1)]}$ and $\overline{\boldsymbol{R}}\in\Re^{(m\times M)\times(m\times M)}$ are positive definite diagonal weighting matrices defined as:

$$\overline{\boldsymbol{W}} = \begin{bmatrix} \boldsymbol{W} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{W} \end{bmatrix}; \; \boldsymbol{W} = \begin{bmatrix} W_1 & 0 & \cdots & 0 \\ 0 & W_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & W_n \end{bmatrix}; \; \overline{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{R} \end{bmatrix};$$

$$\boldsymbol{R} = \begin{bmatrix} R_1 & 0 & \cdots & 0 \\ 0 & R_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & R_m \end{bmatrix}; \tag{2.32}$$

where the diagonal elements of the matrix $\boldsymbol{W}$ consists of the weights $W_{i=1,\ldots,n}$ for output residuals, whereas the diagonal elements of the matrix $\boldsymbol{R}$ consists of the move suppression weights $R_{i=1,\ldots,m}$ for changes in inputs. These weights are tunable parameters for obtaining good performance in the GPC controller. In the original GPC algorithm devised by Clarke, Mohtadi, and Tuffs (1987a), however, $W_i = 1$ (*i.e.* $\overline{\boldsymbol{W}} = \boldsymbol{I}$) was imposed on the GPC cost function and the same will be used in this work for simplicity and to reduce the amount of tuning parameters involved.

For the unconstrained GPC, the solution to $\dfrac{\partial J}{\partial \Delta \underset{\rightarrow k-1}{\boldsymbol{u}}} = 0$ in Eqn. (2.31) with $\overline{\boldsymbol{W}} = \boldsymbol{I}$ yields the following unconstrained GPC control law:

$$\Delta \boldsymbol{u}_k = \boldsymbol{e}_1^T \left(\boldsymbol{H}^T \boldsymbol{H} + \overline{\boldsymbol{R}} \right)^{-1} \boldsymbol{H}^T \left[\underset{\rightarrow k}{\boldsymbol{r}} - \boldsymbol{Q} \underset{\leftarrow k}{\boldsymbol{y}} - \boldsymbol{K} \Delta \underset{\leftarrow k-1}{\boldsymbol{u}} \right] \tag{2.33}$$

where $\boldsymbol{e}_1^T = \left[\boldsymbol{I}_{m \times m}, \boldsymbol{0}, \boldsymbol{0}, \ldots, \boldsymbol{0} \right]_{m \times mM}$.

In the unconstrained GPC, it is possible to derive the closed loop transfer function for the purpose of evaluating the closed loop poles. To do this, Eqn. (2.33) is first rewritten in the following form:

$$\Delta \boldsymbol{u}_k = \hat{\boldsymbol{P}}_k \underset{\rightarrow k}{\boldsymbol{r}} - \hat{\boldsymbol{N}}_k \underset{\leftarrow k}{\boldsymbol{y}} - \hat{\boldsymbol{D}}_k \Delta \underset{\leftarrow k-1}{\boldsymbol{u}} \tag{2.34}$$

where:

$$\hat{\boldsymbol{P}}_k = \boldsymbol{e}_1^T \left(\boldsymbol{H}^T \boldsymbol{H} + \overline{\boldsymbol{R}} \right)^{-1} \boldsymbol{H}^T$$

$$\hat{\boldsymbol{N}}_k = \boldsymbol{e}_1^T \left(\boldsymbol{H}^T \boldsymbol{H} + \overline{\boldsymbol{R}} \right)^{-1} \boldsymbol{H}^T \boldsymbol{Q} \tag{2.35}$$

$$\hat{\boldsymbol{D}}_k = \boldsymbol{e}_1^T \left(\boldsymbol{H}^T \boldsymbol{H} + \overline{\boldsymbol{R}} \right)^{-1} \boldsymbol{H}^T \boldsymbol{K}$$

Defining the components of Eqn. (2.35) as in Eqn. (2.36), Eqn. (2.34) can be rewritten as Eqn. (2.37):

$$\hat{\boldsymbol{P}}_k = \left[\hat{\boldsymbol{P}}_{k+N_1} \quad \hat{\boldsymbol{P}}_{k+N_1+1} \quad \hat{\boldsymbol{P}}_{k+N_1+2} \quad \cdots \quad \hat{\boldsymbol{P}}_{k+N_2} \right]$$

$$\hat{\boldsymbol{N}}_k = \left[\hat{\boldsymbol{N}}_k \quad \hat{\boldsymbol{N}}_{k-1} \quad \hat{\boldsymbol{N}}_{k-2} \quad \cdots \quad \hat{\boldsymbol{N}}_{k-\alpha} \right] \tag{2.36}$$

$$\hat{\boldsymbol{D}}_k = \left[\hat{\boldsymbol{D}}_{k-1} \quad \hat{\boldsymbol{D}}_{k-2} \quad \hat{\boldsymbol{D}}_{k-3} \quad \cdots \quad \hat{\boldsymbol{D}}_{k-D+(1-\beta)} \right]$$

$$\Delta \boldsymbol{u}_k = \hat{\boldsymbol{P}}_k \left(z \right) \boldsymbol{r}_k - \hat{\boldsymbol{N}}_k \left(z^{-1} \right) \boldsymbol{y}_k - \hat{\boldsymbol{D}}_k \left(z^{-1} \right) \Delta \boldsymbol{u}_k \tag{2.37}$$

where:

$$\hat{\boldsymbol{P}}_k \left(z \right) = \hat{\boldsymbol{P}}_{k+N_1} z^{N_1} + \hat{\boldsymbol{P}}_{k+N_1+1} z^{N_1+1} + \hat{\boldsymbol{P}}_{k+N_1+2} z^{N_1+2} + \ldots + \hat{\boldsymbol{P}}_{k+N_2} z^{N_2}$$

$$\hat{\boldsymbol{N}}_k \left(z^{-1} \right) = \hat{\boldsymbol{N}}_k + \hat{\boldsymbol{N}}_{k-1} z^{-1} + \hat{\boldsymbol{N}}_{k-2} z^{-2} + \ldots \hat{\boldsymbol{N}}_{k-\alpha} z^{-\alpha} \tag{2.38}$$

$$\hat{\boldsymbol{D}}_k \left(z^{-1} \right) = \hat{\boldsymbol{D}}_{k-1} z^{-1} + \hat{\boldsymbol{D}}_{k-2} z^{-2} + \hat{\boldsymbol{D}}_{k-3} z^{-3} + \ldots + \hat{\boldsymbol{D}}_{k-D+(1-\beta)} z^{-D+(1-\beta)}$$

Rearranging Eqn. (2.37) gives the following:

$$\left[\boldsymbol{I} + \hat{\boldsymbol{D}}_k \left(z^{-1} \right) \right] \Delta \boldsymbol{u}_k = \hat{\boldsymbol{P}}_k \left(z \right) \boldsymbol{r}_k - \hat{\boldsymbol{N}}_k \left(z^{-1} \right) \boldsymbol{y}_k \tag{2.39}$$

From Eqn. (2.1), ignoring the stochastic components (*i.e.* the bias parameter and the noise term), it can be shown that:

$$\boldsymbol{y}_k = \left[\boldsymbol{a} \left(z^{-1} \right) \right]^{-1} \boldsymbol{b} \left(z^{-1} \right) z^{-D} \boldsymbol{u}_k \tag{2.40}$$

Substituting Eqn. (2.40) in Eqn. (2.39) yields:

$$\boldsymbol{u}_k = \left[\boldsymbol{b}\left(z^{-1}\right)z^{-D}\right]^{-1}\left\{\left[\boldsymbol{I}+\hat{\boldsymbol{D}}_k\left(z^{-1}\right)\right]\Delta\left[\boldsymbol{b}\left(z^{-1}\right)z^{-D}\right]^{-1}+\hat{\boldsymbol{N}}_k\left(z^{-1}\right)\left[\boldsymbol{a}\left(z^{-1}\right)\right]^{-1}\right\}^{-1}\hat{\boldsymbol{P}}_k\left(z\right)\boldsymbol{r}_k \tag{2.41}$$

Hence, the closed loop transfer function is obtained in a straight forward manner by utilizing Eqn. (2.41) in Eqn. (2.40):

$$\boldsymbol{y}_k = \left[\boldsymbol{a}\left(z^{-1}\right)\right]^{-1}\left\{\left[\boldsymbol{I}+\hat{\boldsymbol{D}}_k\left(z^{-1}\right)\right]\Delta\left[\boldsymbol{b}\left(z^{-1}\right)z^{-D}\right]^{-1}+\hat{\boldsymbol{N}}_k\left(z^{-1}\right)\left[\boldsymbol{a}\left(z^{-1}\right)\right]^{-1}\right\}^{-1}\hat{\boldsymbol{P}}_k\left(z\right)\boldsymbol{r}_k \tag{2.42}$$

where the closed loop poles are given by the solution to the following characteristic equation:

$$\det\left(\left[\boldsymbol{I}+\hat{\boldsymbol{D}}_k\left(z^{-1}\right)\right]\Delta\left[\boldsymbol{b}\left(z^{-1}\right)z^{-D}\right]^{-1}+\hat{\boldsymbol{N}}_k\left(z^{-1}\right)\left[\boldsymbol{a}\left(z^{-1}\right)\right]^{-1}\right)=0 \tag{2.43}$$

The straightforward solution to the GPC cost function and the derivation of the GPC closed loop transfer function as described above are only possible with the unconstrained GPC. For the constrained GPC, the cost function J is minimized with respect to $\Delta\underset{\rightarrow k-1}{\boldsymbol{u}}$ while satisfying the following constraints:

$$\boldsymbol{u}_{\min} \le \underset{\rightarrow k-1}{\boldsymbol{u}} \le \boldsymbol{u}_{\max}$$

$$\Delta\boldsymbol{u}_{\min} \le \Delta\underset{\rightarrow k-1}{\boldsymbol{u}} \le \Delta\boldsymbol{u}_{\max} \tag{2.44}$$

The optimal sequence of the future input trajectories is then obtained by rearranging and solving Eqn. (2.31) in the form of a Quadratic Programming (QP) problem (given $\overline{\boldsymbol{W}} = \boldsymbol{I}$):

$$\min_{\Delta \boldsymbol{u}_{\rightarrow k-1}} \left(\frac{1}{2} \Delta \boldsymbol{u}^T_{\rightarrow k-1} \boldsymbol{B} \Delta \boldsymbol{u}_{\rightarrow k-1} + \boldsymbol{g}^T \Delta \boldsymbol{u}_{\rightarrow k-1} \right)$$
$$\text{Subject to } \boldsymbol{\Omega} \Delta \boldsymbol{u}_{\rightarrow k-1} - \boldsymbol{\mu} \leq \boldsymbol{0} \tag{2.45}$$

where $\boldsymbol{B}$ and $\boldsymbol{g}$ are defined as (Rossiter, 2003):

$$\boldsymbol{B} = \boldsymbol{H}^T \boldsymbol{H} + \overline{\boldsymbol{R}}$$

$$\boldsymbol{g} = \boldsymbol{H}^T \left[\boldsymbol{K} \Delta \boldsymbol{u}_{\leftarrow k-1} + \boldsymbol{Q} \boldsymbol{y}_{\leftarrow k} - \boldsymbol{r}_{\rightarrow k} \right] \tag{2.46}$$

The matrices $\boldsymbol{\Omega}$ and $\boldsymbol{\mu}$ in Eqn. (2.45) can be obtained by formulating Eqn. (2.44) in the form of linear inequalities, of which the details can be found in Camacho and Bordons (1999), Maciejowski (2002), and Rossiter (2003). The solution to the QP problem in Eqn. (2.45) yields the optimal future input trajectories, where only the first move is implemented in the process.

Regarding stability in the constrained case, although it can be guaranteed by parametric interpretations and the use of the well-known "dual-mode" algorithms with guaranteed convergence (Rossiter, 2003), in terms of evaluating stability by means of the closed loop poles, there has not been any possibility of doing so for the constrained GPC. Thus, this work considers only the stability in the unconstrained case through its closed loop transfer function. Despite the fact that stability in the unconstrained case does not guarantee stability in the constrained case, it is still worth doing a study in the stability of the former scenario (Rossiter, 2003) because if the control law in the unconstrained case does not give good performance, the corresponding constrained case cannot be expected to give good performance.

2.5 Generalized Predictive Control: Tuning Methods

As alluded to in the previous section, the GPC algorithm has several tuning parameters which can be adjusted to affect the performance of the controller. For ease of reference, the tuning parameters are recapitulated here:

a) Minimum prediction horizon (N_1)

b) Maximum prediction horizon (N_2)

c) Control horizon (M)

d) Move suppression weight (R_i)

Over the decades, many authors had proposed different formulations and strategies to tune the GPC controller. In a recent work, Garriga and Soroush (2010) presented a review on the various MPC (*i.e*. GPC and DMC) tuning methods available, with a specific section dedicated specifically to the subject of "auto-tuning" (synonymous to "self-tuning" in the context of this work). In the following subsections, a brief review on the various offline GPC tuning methods is given, followed by discussions on the developments of self-tuning methods.

2.5.1 Minimum and Maximum Prediction Horizons (N_1 and N_2)

In general, the value of N_1 in the GPC formulation is set to 1 if the dead time of the plant is not known *a priori* as selecting a value of N_1 much greater than the dead time of the plant will cause the controller to ignore the dynamics of the plant within the (N_1 - dead time) horizon. This may cause the controller performance to deteriorate since the ignored dynamics can be crucial in determining the appropriate control output. As such, the value of $N_1 = 1$ is a conservative choice if nothing is known about the dead

time of the process (Banerjee & Shah, 1992; Clarke, Mohtadi, & Tuffs, 1987b; McIntosh, Shah, & Fisher, 1989, 1991). Conversely, if the discrete dead time of the plant is known *a priori*, N_1 can be selected as equal to (discrete dead time + 1) to minimize computations (Camacho & Bordons, 1999; Clarke, Mohtadi, & Tuffs, 1987a). Other tuning guidelines for selecting the minimum prediction horizon based on the order of $\boldsymbol{a}(z^{-1})$ and $\boldsymbol{b}^*(z^{-1})$ [where $\boldsymbol{b}^*(z^{-1}) = z^{-D}\boldsymbol{b}\ (z^{-1})$] are also available in the literature (Clarke & Mohtadi, 1989; McIntosh, Shah, & Fisher, 1989, 1991).

As for N_2, in order to guarantee the stability of the closed loop system, its value should be set to infinite (Camacho & Bordons, 1999; Maciejowski, 2002; Rossiter, 2003), which in practical terms would mean setting N_2 to large values. If a finite value for N_2 is selected, its value should be carefully chosen based on tuning guidelines to ensure the closed loop stability of the system. Clarke, Mohtadi, and Tuffs (1987a) proposed setting the value of N_2 between the order of $\boldsymbol{b}^*(z^{-1})$ and the discrete rise time of the process. In another work, Clarke and Mohtadi (1989) proposed that the value of N_2 be chosen based on a rise time of 95 % of the process steady state. A more explicit tuning correlation was proposed by Shridhar and Cooper (1997b) for the selection of N_2 for SISO FOPDT systems:

$$N_2 = CEIL\left[5\left(\frac{\tau_p}{t_s}\right) + D\right] \tag{2.47}$$

where τ_p is the process time constant and $CEIL[\bullet]$ is defined as the nearest next integer to $[\bullet]$. Although Eqn. (2.47) was originally developed for the tuning of DMC, the authors claimed that the same equation can be also used for tuning the GPC. Many other tuning guides had been proposed for the selection of N_2 (Banerjee & Shah, 1992; McIntosh, Shah, & Fisher, 1989, 1990, 1991; Trierwieler & Farina, 2003; Yamuna & Unbehauen, 1997), however, the general agreement is that N_2 should be sufficiently

large to cover the important parts of the step response curve (Mikleš & Fikar, 2007) as well as to ensure the stability of the closed loop system.

2.5.2 Control Horizon (*M*)

In the GPC formulation, selecting a value of $M = 1$ produces a conservative controller, whereas values of $M > 1$ produces a more robust controller at the cost of increased computation load. Several authors have proposed to set the value of $M = 1$ (Banerjee & Shah, 1992; Clarke & Mohtadi, 1989; Clarke, Mohtadi, & Tuffs, 1987a, 1987b; McIntosh, Shah, & Fisher, 1989, 1991; Yamuna & Unbehauen, 1997). Other than that, Clarke and Mohtadi (1989) proposed that M be equated to the discrete dead time of the process, whereas McIntosh, Shah, and Fisher (1989, 1991) proposed that M be chosen based on the order of $\boldsymbol{a}(z^{-1})$ plus one. In his book on MPC, Rossiter (2003) stated that "a value of $M \geq 3$ often seems to give performance close to the global optimal". Other prediction horizon based and stability based tuning rules for selecting the value of M in the GPC framework are available in the work of Rawlings and Muske (1993) as well as in the work of Trierwieler and Farina (2003). In general, the guidelines given by these authors suggest that the value of M should not be overly large. Furthermore, the control horizon does not have significant effects on the closed loop performance in the presence of move suppression (Shridhar & Cooper, 1997b), which is the reason for the authors' proposal of the values of M between 1 to 6.

2.5.3 Move Suppression Weight (R_i)

The move suppression weight, also known as the move suppression coefficient, is used to penalize the slew rates such that a larger penalty produces a more robust but

sluggish controller. Conversely, selecting a small value of R_i or a complete absence of move suppression produces an aggressive but less robust controller. This section shall be restricted to the more common methods of tuning R_i (Garriga & Soroush, 2010) and shall not cover the concepts deployed in algorithms such as the Predictive Functional Control *etc* (Rossiter, 2003). Several authors have proposed to select the value of R_i based on actual control performance, *i.e*. trial and error (Karacan, Hapoglu, & Alpbaz, 2001; McIntosh, Shah, & Fisher, 1989, 1991). In these studies, the strategies used by the authors involved making R_i the single active tuning parameter, while all other tuning parameters were fixed. This strategy concurs with the perspective of optimal feedback control (*e.g*. the Linear Quadratic Regulator, LQR), where the weights of the objective function to be minimized are the sole tuning parameters of the controller (Mikleš & Fikar, 2007). Clarke and Mohtadi (1989) suggested setting the value of $R_i = 0$. However, if numerical stability is an issue of concern, the authors proposed to adopt values of $R_i \approx 0$. Banerjee and Shah (1992) suggested choosing values of R_i in the range between 1 - 2. Yamuna and Unbehauen (1997) proposed to tune the move suppression weight based on the following relationship:

$$R_i = \kappa N_2 + \pi \tag{2.48}$$

where κ and π are constants determined from past R_i and N_2 values. Using this relationship, R_i is recalculated each time N_2 is retuned based on the performance tuning procedure proposed by McIntosh, Shah, and Fisher (1990).

In another tuning formulation by McIntosh, Shah, and Fisher (1989), the authors proposed that an initial guess for the "relative control weighting (R_i^*)" be calculated based on trace($\boldsymbol{H}^T\boldsymbol{H}$) and subsequent values of R_i be calculated based on the relationship of $R_i = R_i^* \, \boldsymbol{b}^*(1)$. In addition, the value of R_i^* can be fine-tuned during operation to improve the overall performance. Most of the methods described to this juncture are not

suitable for RLS-based self-tuning purposes due to the lack of association between the tuning guides and the process model parameters. Shridhar and Cooper (1997b) in their attempt to tune R_i based on the process model parameters of an open loop stable SISO FOPDT model proposed the following analytical expressions to calculate its value:

$$R_i = f\,\mathrm{K}_\mathrm{p}^2 \tag{2.49}$$

where:

$$f = \begin{cases} 0 & M = 1 \\ \dfrac{M}{500}\left[\dfrac{3.5\tau_\mathrm{p}}{t_s} + 2 - \dfrac{(M-1)}{2}\right] & M > 1 \end{cases} \tag{2.50}$$

Equations (2.49) - (2.50), although developed originally for unconstrained DMC, can also be directly applied for tuning the R_i in the GPC (Mikleš & Fikar, 2007). The authors also proposed to select the usual values of M ranging from 1 to 6 when these expressions are applied. Since Eqns. (2.49) - (2.50) are developed based on a process model and are relatively easy to implement, the self-tuning component in the AS-GPC proposed in this work shall be designed based on these equations. As to the success of this tuning strategy, several researchers had obtained good results with it (Nithya, Gour, Sivakumaran, Radhakrishnan, & Anantharaman, 2007; Tahami & Ebad, 2009; Villanueva Perales, Ollero, Gutierrez Ortiz, & Gomez-Barea, 2009). However, in the review of Garriga and Soroush (2010), the authors concluded that this tuning strategy yielded faster response at the expense of more aggressive controller moves. To what extent this conclusion holds true when the tuning strategy is deployed in a recursive fashion, remains a question to be answered in Chapter 6 of this thesis.

2.5.4 Self-Tuning Methods

In these methods, tuning of the GPC is not done offline by control engineers, but rather, the controller automatically retunes itself in real time. As the effect of an

individual tuning parameter strongly depends on the settings of the other tuning parameters in the GPC controller (McIntosh, Shah, & Fisher, 1991), the task of controller tuning is not straightforward for an inexperienced control engineer. Hence, with the implementations of self-tuning strategies, minimal knowledge is required of the control engineer to initiate the tuning procedure.

In the literature, although the majority of the developments in self-tuning of the MPC were tailored specifically for the DMC, in general the same principle can be applied on the GPC. In the work of Al-Ghazzawi, Ali, Nouh, and Zafiriou (2001), the authors developed analytical sensitivity expressions relating the process outputs and the DMC tuning parameters (*i.e.* the weights on the output residuals and the move suppression coefficients) and utilized it for automatic online adjustments of the tuning parameters. The prediction and control horizons were set at predetermined values and only the weights on the output residuals and the move suppression weights were retuned in real time. Han, Zhao, and Qian (2006) proposed a self-tuning strategy based on Particle Swarm Optimization (PSO). A novel performance index was developed and PSO was used to search for the optimal tuning parameters while minimizing the performance index to cater to the worst operating conditions. Kawai *et al.* (2007) applied this method to tune the DMC. However, instead of computing the entire set of tuning parameters online, the optimization problem was restricted to computing the optimal values of the inputs and outputs weights only, hence reducing the computation load required. Ali and Al-Ghazzawi (2003) presented a self-tuning scheme based on fuzzy logics. In this scheme, the prediction horizon, the move suppression weights, and the weights of the output residuals were determined from predefined fuzzy rules which formulate the general tuning guides reported in the literature, while the control horizon was fixed at a constant value. As for the GPC, Liu and Wang (2000) proposed two

algorithms to carry out a multi-objective optimization for the purpose of obtaining the optimal set of tuning parameters as well as to optimize the control moves online. In a recent work for the GPC implemented through the Programmable Logic Controller (PLC), Valencia-Palomoa and Rossiter (2010) suggested to auto-tune the GPC based on estimates of model behavior as compared to standard second order characteristics (*i.e.* rise time, settling time, overshoot, process gain, dead time and sampling time of the process) of the process. In all these studies, a static internal model was employed for the predictive controller, and no model adaptation scheme was adopted explicitly. This work attempts to incorporate both the model adaptation and self-tuning strategies in the GPC controller by using the output of the RLS algorithm, of which the details are given in Section 3.6. To what extent the performance of the controller can be improved (if there is any at all), remains a question to be answered in this work.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 General Remarks

In this chapter, only the necessary procedures and methods used to meet the objectives of the research will be elucidated in detail, while the results of the respective sections are presented in the subsequent chapters. Unless mentioned otherwise, all simulations involved in this work were programmed using Simulink® version 7.1 of MATLAB® program version R2008a. Where the use of S-function in Simulink® is involved, the programs were coded as an M-file to be embedded within the S-function block. The use of MATLAB toolboxes where applicable will be highlighted in the sections concerned.

3.2 Coding and Screening of Various RLS algorithms

Selected RLS algorithms with Bierman's $\mathbf{UDU}^T$ factorization were coded as S-functions. An example code of the RLS algorithm, specifically the VFF-RLS algorithm, is given in Appendix C. To ensure proper and correct coding, the conventional RLS algorithm was first coded and implemented on a simple LTI model, with fixed process model parameters. The results obtained, *viz.* the estimated process model parameters from the simulation was validated with the results obtained from the sample program (which also utilizes the conventional RLS algorithm) given by Mikleš and Fikar (2007). The covariance matrix and the prediction error of the identification were also validated against those obtained from the sample program. Upon successful validation of the coded conventional RLS algorithm, other variants of the RLS algorithm were obtained

by slight modification of the code for the conventional RLS algorithm. The simulation results obtained by using the modified RLS algorithms should not differ much from those obtained from the conventional RLS algorithm, since the system involved here is linear time invariant, and these modifications of the RLS algorithm only show significant differences in performance when the system being identified is time-varying.

To test the performance of the various RLS algorithms in identifying time-varying systems, a hypothetical, stable, discrete time LTI transfer function in the z-domain was used as the process model:

$$G_p(z) = \frac{0.6z^{-1} + 0.7z^{-2} + 0.8z^{-3} + 0.9z^{-4} + 1.0z^{-5}}{1 + 0.1z^{-1} + 0.2z^{-2} + 0.3z^{-3} + 0.4z^{-4} + 0.5z^{-5}} \tag{3.1}$$

In Eqn. (3.1), there are altogether ten parameters to be identified. This number of parameters was chosen to make system identification more difficult in order to challenge each RLS algorithm. To simulate a process model where the process model parameters are time-varying, the process model parameters, which consist of random and arbitrary values but resemble a stable plant, were varied abruptly at the 70^{th} time step as follows:

$$G_p(z) = \frac{0.1z^{-1} + 0.9z^{-2} + 1.5z^{-3} - 0.2z^{-4} - 0.3z^{-5}}{1 + 1.1z^{-1} + 0.8z^{-2} + 0.6z^{-3} + 0.7z^{-4} + 0.3z^{-5}} \tag{3.2}$$

To ensure that the system is sufficiently excited, white Gaussian noise with zero mean and variance of 0.1 was used as an input to the system, while the same noise type with a variance of 0.01 was added to the output of the system to simulate a noisy environment. The Euclidean norm of the parameter error, $L_2 = \| \boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}} \|_2$ at every time step was calculated and plotted against time, where $\boldsymbol{\theta}_0$ is the vector of parameters which contains the true values of the process model parameters, as shown in Eqns. (3.1) and (3.2), and $\hat{\boldsymbol{\theta}}$ is the vector of estimated parameters. Based on the Euclidean norm of the

parameter error, the convergence and the steady state parameter mis-adjustment error of the various RLS algorithms were analyzed to explore the strengths and weaknesses of each algorithm. Based on the screening results, one of the RLS algorithms was selected for self-tuning control implementations in this work.

3.3 Coding of the Generalized Predictive Controller

The GPC algorithms (excluding the quadratic programming solver) for both the unconstrained and the constrained versions were coded as S-functions. For the constrained GPC, in order to solve the quadratic programming problem online at every time step, the subroutine 'quadprog' under the Optimization Toolbox™ was used with the default settings retained.

For validation purposes, assuming the knowledge of the internal GPC model used, *e.g.* a first order CARIMA model, it is possible to manually determine the numerical values of $\boldsymbol{C_A}$, $\boldsymbol{H_A}$, $\boldsymbol{C_b}$, and $\boldsymbol{H_b}$ matrices (as shown in Appendix B), which eventually can be used to determine the $\boldsymbol{H}$, $\boldsymbol{K}$, and $\boldsymbol{Q}$ matrices as shown in Eqns. (2.29) - (2.30). These manually determined results were used to validate the results obtained from the coded GPC algorithm. The S-functions for the unconstrained and constrained GPC algorithm and its variants developed here are shown in Appendix D and E respectively. Although decentralized controllers were designed in this work, the GPC algorithm coded here can handle multivariable control problems, as shown in one of the author's publications (Ho, Mjalli, & Yeoh, 2010a).

3.4 Open Loop Dynamic System Analysis of the Biodiesel Reactor

Open loop dynamic system analysis of the transesterification reactor model was done for the following reasons:

a) To show the nonlinearities exhibited by the transesterification process, hence justifying the need for an advanced control algorithm to be implemented.
b) To use the data of the open loop response for the transesterification process for offline system identification, where the parameters of a FOPDT model were estimated. The estimated model parameters were then used in the design of the control schemes presented in this work.

To study the open loop transients of the transesterification process, the manipulated variables (*i.e*. F_o and F_c) were increased in small steps across the respective operating ranges by manipulating the control valve openings. For ease of reference, the control valves for manipulating F_o and F_c were termed CV-101 and CV-102 respectively. In the work of Mjalli, Lee, Kiew, and Hussain (2009), loop pairings were done by using the Relative Gain Array (RGA) analysis, where F_o was paired against C_{ME}, while F_c was paired with T. With these pairing results retained, it is of interest in this study to show via the open loop transients the nonlinearities involved in the transesterification process as well as the extent of the process interactions observed. Series of step changes in the valve stem positions of CV-101 and CV-102 (which in turn manipulate F_o and F_c) were designed and introduced to the process in the manner as shown in Table 3.1. The values of the valve stem positions can be of any arbitrary numbers spanning from 0 % to 100 %, but since the nominal valve stem positions of CV-101 and CV-102 for the biodiesel reactor are 17 % and 26.8 % respectively, it is more convenient to choose combinations that step through these values.

Table 3.1: Trends of step changes in the valve stem positions of CV-101 and CV-102 at every time interval of 3000 s. The control valves fully shut and fully open at 0 % and 100 % respectively.

Simulation	Valve Stem Position for CV-101 (%)	Valve Stem Position for CV-102 (%)
1	7 : +10 : 97	26.8
2	97 : -10 : 7	26.8
3	17	6.8 : +10 : 96.8
4	17	96.8 : -10 : 6.8
5	7 : +10 : 97	6.8 : +10 : 96.8
6	7 : +10 : 97	96.8 : -10 : 6.8
7	97 : -10 : 7	6.8 : +10 : 96.8
8	97 : -10 : 7	96.8 : -10 : 6.8

Referring to simulation 1 from the table, the valve stem position for CV-101 was set at 7 % in the beginning of simulation (to be varied afterwards), while maintaining a constant valve stem position of 26.8 % for CV-102 throughout the entire simulation. Upon achieving steady state of the output variables, a positive step change of 10 % in the valve stem position for CV-101 was introduced at 3000 s. Subsequent introductions of positive step changes of the same magnitude were done at intervals of 3000 s until the valve approaches its high operating range at 97 %. The open loop transients of the input (*i.e.* F_o and F_c) and output (*i.e.* C_{ME} and T) variables as well as the corresponding valve stem positions were then recorded. It is important to note that the time duration of 3000 s selected was sufficiently long to capture the open loop transients of the process after every step change and the output variables were able to achieve steady states within the given time duration. In simulation 2, instead of having an ascending trend in

the valve stem position for CV-101, step changes of the same magnitude in the opposite direction (*i.e*. from 97 % to 7 %) were introduced.

In simulations 3 and 4, the same procedures as in simulations 1 and 2 were used to study the effect of the valve stem position for CV-102 on the output variables. The valve stem position for CV-101 was maintained at 17 %, while step changes of opposite directions were introduced on separate runs (*i.e*. the positive step changes for simulation 3 and the negative step changes for simulation 4) across the entire operating range of the valve (6.8 % - 96.8 %). In simulations 5 – 8, instead of only manipulating a single input as in simulations 1 – 4, a series of step changes was introduced to both inputs simultaneously in a fashion similar to the 2^2 factorial design, where all possible combinations of step changes in the ascending and descending trends were considered. The open loop transients in these experiments were important to show the nonlinearities and process interactions observed under all possible combinations of input perturbations.

3.5 Offline System Identification of the Biodiesel Reactor

As alluded to in Section 3.4, the data of the open loop transients are useful for offline system identification where simple LTI transfer function models can be identified and used in control design purposes. In this study, the FOPDT model was chosen as the model structure for system identification (the reasons for this choice will be elaborated in the following section). Although the biodiesel reactor is a multivariable process, for simplicity, system identification was performed in a decentralized fashion, where only $F_o - C_{ME}$ and $F_c - T$ relationships were modeled, thus omitting the possible process interactions involved. The iterative prediction-error minimization method (Ljung, 2008), denoted by the 'pem' subroutine under the System Identification

Toolbox^{TM}, was used to estimate the model parameters of a FOPDT model with a maximum allowable estimated dead time of 100 s. Since it is of interest in this study to identify individual SISO models, the data from simulations 1 and 3 in Table 3.1 were used for achieving this purpose. The estimated K_p, τ_p, and θ_d (*i.e.* dead time) were recorded and the R^2 (*i.e.* coefficient of determination) of the model fit was computed.

3.6 Advanced Control System Design for the Biodiesel Reactor

In this work, it is of interest to study the performance of the AS-GPC scheme as compared to the A-GPC and GPC schemes. The GPC scheme in this case is the conventional GPC algorithm without model adaptation or self-tuning. Before exploring the performances of these schemes under closed loop conditions on the transesterification reactor, the control architecture of the various schemes under consideration must be clearly defined. For all schemes, two controllers were designed, one for the F_o - C_{ME} loop and the other for the F_c - T loop. It should be noted that the procedures here are general and applicable to both the unconstrained and constrained versions of the particular control scheme under discussion.

The conventional GPC scheme differs from the AS-GPC and A-GPC schemes in that no RLS algorithm is involved. The model parameters used in the GPC remained unchanged throughout its entire implementation and were obtained by converting the continuous time model parameters (*i.e.* from the offline system identification outlined in Section 3.4) to the corresponding discrete version using Eqns. (3.3) - (3.5) (Seborg, Edgar, & Mellichamp, 2004):

$$a_1 = -\exp\left[-\frac{t_s}{\tau_p}\right] \tag{3.3}$$

$$b_1 = \mathrm{K_p}\left[1 - \exp\left(-\frac{t_s}{\tau_p}\right)\right] \tag{3.4}$$

$$D = CEIL\left[\frac{\theta_d}{t_s}\right] \tag{3.5}$$

The sampling time t_s selected for both C_{ME} and T loops were based on the values of τ_p identified around the nominal operating conditions as outlined in Section 3.4, where $t_s \approx 0.1\tau_p$ according to rule of thumb (Seborg, Edgar, & Shah, 1986). Values of θ_d were also obtained from the offline system identification around the nominal operating conditions. In the GPC scheme, the controller tuning parameters were calculated by using Eqns. (2.49) - (2.50) proposed by Shridhar and Cooper (1997b).

As opposed to the GPC scheme where no RLS algorithm is needed, for each separate control loop in the AS-GPC and A-GPC schemes, the screened and selected RLS algorithm by using the procedures outlined in Section 3.2 was used to identify the process model parameters of a SISO first order ARX model, which would then be incorporated in the CARIMA representation to be used for prediction in the controller. The reasons for identifying the process model parameters of a SISO first order ARX model, which can easily be recast in the form of a FOPDT model (*i.e.* ignoring the bias parameter and the noise term), are obvious:

i) The FOPDT model is a widely used model structure for control design purposes and often serves as a reasonable approximation for the dynamics of most processes (Seborg, Edgar, & Mellichamp, 2004).

ii) Tuning correlations based on the FOPDT model for predictive controllers are available (Shridhar & Cooper, 1997b), which can be conveniently adapted to design the AS-GPC controller.

Table 3.2 shows the necessary parameters to be defined in the SISO ARX model for each loop to cater for parameter estimation purposes. Among the parameters shown, the values of n, m, α and β were determined in a straightforward manner based on the structure of the model chosen (*i.e.* a SISO first order ARX model). As for the values of D and t_s, both were determined in the same manner as described for the GPC scheme.

Table 3.2: Parameters of the discrete time SISO ARX model which must be defined for parameter estimation purposes, both for the FAME concentration (C_{ME}) and the reactor temperature (T) loops.

Parameter	Symbol	Unit
Number of outputs	n	-
Number of inputs	m	-
Order of $a(z^{-1})$	α	-
Order of $b(z^{-1})$	β	-
Discrete dead time	D	-
Sampling time	t_s	s

Prior to coupling the screened and selected RLS algorithm (as outlined in Section 3.2) to the GPC algorithm for the design of the AS-GPC and A-GPC schemes, the parameters associated with the specific form of the RLS algorithm (*e.g.* σ and C for the VFF-RLS algorithm *etc*.) chosen for control system design must be determined. Readers are referred to Subsection 2.3.2 for more information on the parameters. Figures 3.1 and 3.2 show the adaptive strategies used in the AS-GPC and the A-GPC schemes respectively. In both the AS-GPC and the A-GPC schemes, the RLS algorithm estimates the process model parameters of a SISO first order ARX model, *viz*. a_1 and b_1, for both the C_{ME} and T loops, which will then be incorporated in the CARIMA representation as shown in Eqn. (2.26) for use in the controller. Ignoring the stochastic

components (*i.e.* the bias parameter and the noise term), the SISO first order ARX model can be recast in the following form, which in essence is a representation of a discrete time FOPDT model:

$$y_k = \left(\frac{b_1 z^{-1} z^{-D}}{1 + a_1 z^{-1}} \right) u_k \tag{3.6}$$

Generally, for any particular system to be stable, the poles must lie within the unit circle of the *z*-plane, *i.e.* $|z| < 1$ (Seborg, Edgar, & Mellichamp, 2004). Before the estimated SISO ARX model parameters can be passed to the GPC for model adaptation in the SISO CARIMA model or self-tuning purposes, the pole of the identified model (*i.e.* $z = -a_1$) was screened to ensure that the identified model had stable dynamics (since the transesterification reactor is an open loop stable process). In the case of a FOPDT model, the stability domain can be further simplified. To do this, consider the continuous time domain equivalent of Eqn. (3.6) given in Eqn. (3.7), which can only be stable, if $\tau_p > 0$:

$$y(s) = \left(\frac{K_p e^{-\theta_d s}}{\tau_p s + 1} \right) u(s) \tag{3.7}$$

Since the values of $\tau_p \in \Re$ and $t_s \in \Re$ must be positive for all real physical processes, imposing the conditions of $\tau_p > 0$ and $t_s > 0$, a_1 in Eqn. (3.3) can only take on negative real values. Knowing $z = -a_1$ for the particular system described by Eqn. (3.6), the criterion for the identified pole to imply stable dynamics is $z \in (0, 1)$ or $a_1 \in (-1, 0)$. Hence, the poles of the identified model for the AS-GPC and A-GPC schemes were screened based on this rule. In short, at any particular time step, if the value of the estimated a_1 falls between -1 and 0, the updated a_1 and b_1 are passed to the GPC; otherwise a backup model previously identified from Section 3.5 and converted to its discrete time equivalent using Eqns. (3.3) - (3.5) are used in the controller. This is

further illustrated in Figures 3.1 - 3.2, where for both the A-GPC and AS-GPC schemes, Route 1 is implemented on the basis of stable dynamics of the identified model, while Route 2 is implemented as a contingency option should the identified model be unstable.

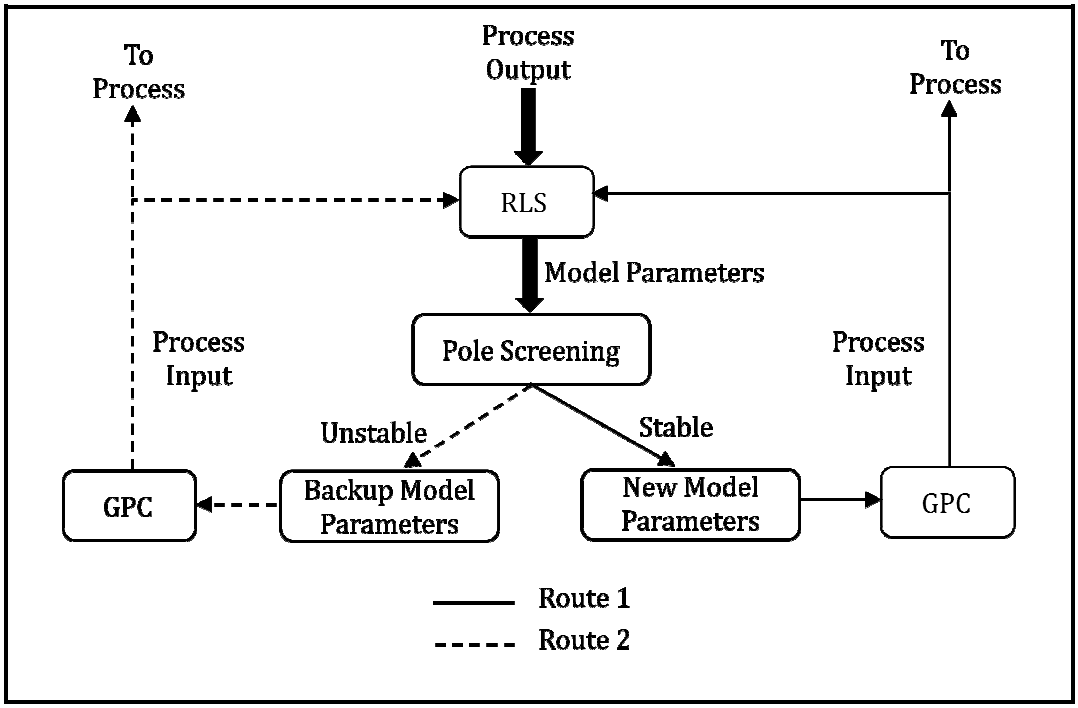

Figure 3.1: The adaptive strategy used in the A-GPC scheme. Pole screening dictates whether Route 1 or Route 2 is implemented at every time step.

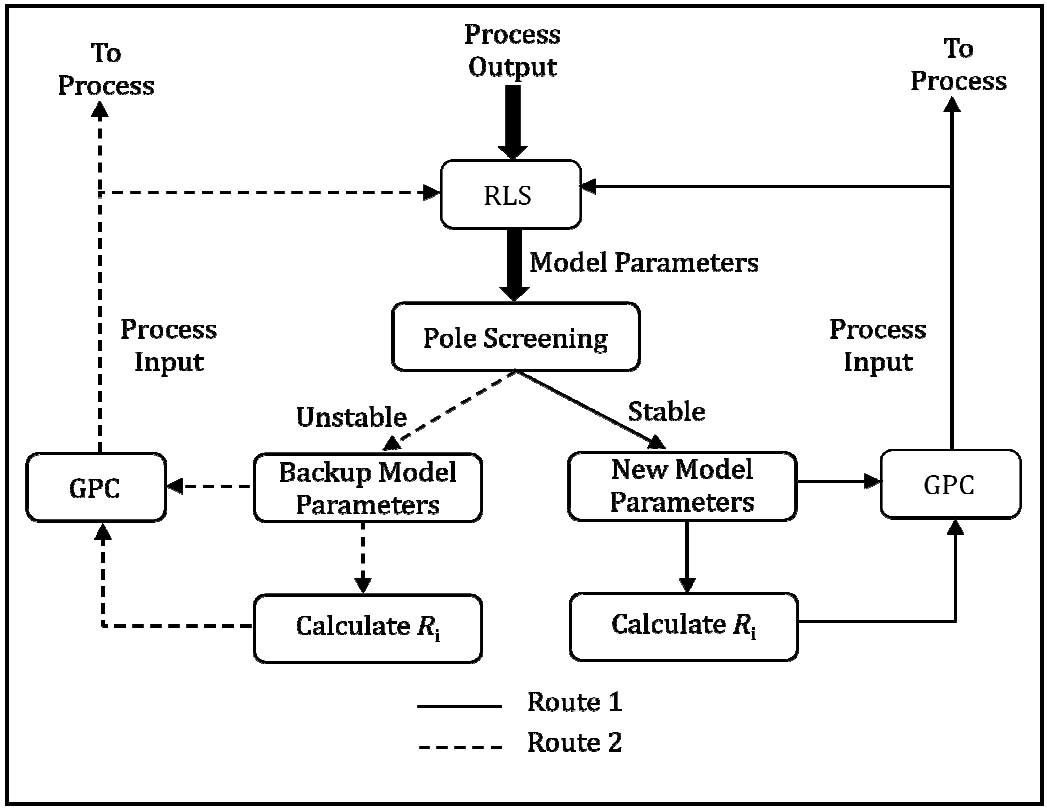

Figure 3.2: The adaptive strategy used in the AS-GPC scheme. Pole screening dictates whether Route 1 or Route 2 is implemented at every time step.

For the A-GPC scheme, the estimated process model parameters were used only for model adaptation in the GPC controller. In this case, the values of the tuning parameters were chosen based on the guideline summarized in Table 3.3 using the value of t_s and the nominal values of τ_p and D selected previously in this section. Most of the tuning guidelines presented in Table 3.3 were taken from the work of Shridhar and Cooper (1997b) except the one presented for N_1, which is a standard guideline throughout most MPC literature (Camacho & Bordons, 1999). In the A-GPC scheme, the tuning parameters were fixed throughout the entire closed loop implementation. As for the AS-GPC scheme, the values of N_1, N_2, M and R_i were initialized in the same manner as described for the A-GPC scheme. However, as soon as the RLS algorithm had stabilized, further processing of the model parameters were performed to compute the updated move suppression weight for each loop. In this strategy, the values of N_1, N_2

and M remained unchanged during implementation, leaving R_i as the single active tuning parameter to be manipulated automatically by the controller in real time. For this purpose, simple manipulation of Eqn. (3.3) is needed to solve for τ_p and K_p at every time step based on the updated value of a_1 from the RLS algorithm:

$$\tau_p = \frac{-t_s}{\ln(-a_1)} > 0 \text{ as } a_1 \in (-1, 0) \text{ and } K_p = \frac{b_1}{1+a_1} \tag{3.8}$$

Hence, with values of τ_p and K_p, R_i can be updated accordingly using the equations developed by Shridhar and Cooper (1997b), *i.e.* Eqns. (2.49) - (2.50). It is to be noted that Eqn. (3.8) is used only for computing R_i, while the values of N_1 and N_2 for a certain value of t_s were calculated based on the nominal values of τ_p and D obtained earlier through offline system identification.

Table 3.3: Tuning guideline used in this work for the A-GPC and AS-GPC schemes

Tuning Parameter	Symbol	Value	A-GPC	AS-GPC
Min. prediction horizon	N_1	$D + 1$	Fixed	Fixed
Max. prediction horizon	N_2	Eqn. (2.47)	Fixed	Fixed
Control horizon	M	1 - 6	Fixed	Fixed
Move suppression weight	R_i	Eqns. (2.49) - (2.50)	Fixed	Adaptive

3.7 Closed Loop Performance Validation and Analysis on the Biodiesel Reactor

Since the tuning correlation proposed by Shridhar and Cooper (1997b) were intended for use with unconstrained predictive controllers, the constraint-free version of the AS-GPC was implemented first on the biodiesel reactor and tested for the ability to track changes in setpoint. The nature of setpoints introduced to the C_{ME} and T loops was of magnitudes ±0.1 kmol/m^3 and ±2 K around the respective nominal operating conditions. The closed loop poles for the constraint-free AS-GPC were calculated based

on Eqn. (2.43) to study the stability of the unconstrained AS-GPC. Following this, the control strategy adopted in the AS-GPC was further pushed to its limits by incorporating constraints [Eqn. (2.44)] to the controller. The constrained AS-GPC scheme was tested for the ability of handling both the servo and the regulatory problems. Furthermore, the performance of the constrained AS-GPC scheme was benchmarked against that of the constrained A-GPC, constrained GPC and conventional PID schemes. The screenshot of the AS-GPC implementation in Simulink® version 7.1 is given in Figure 3.3.

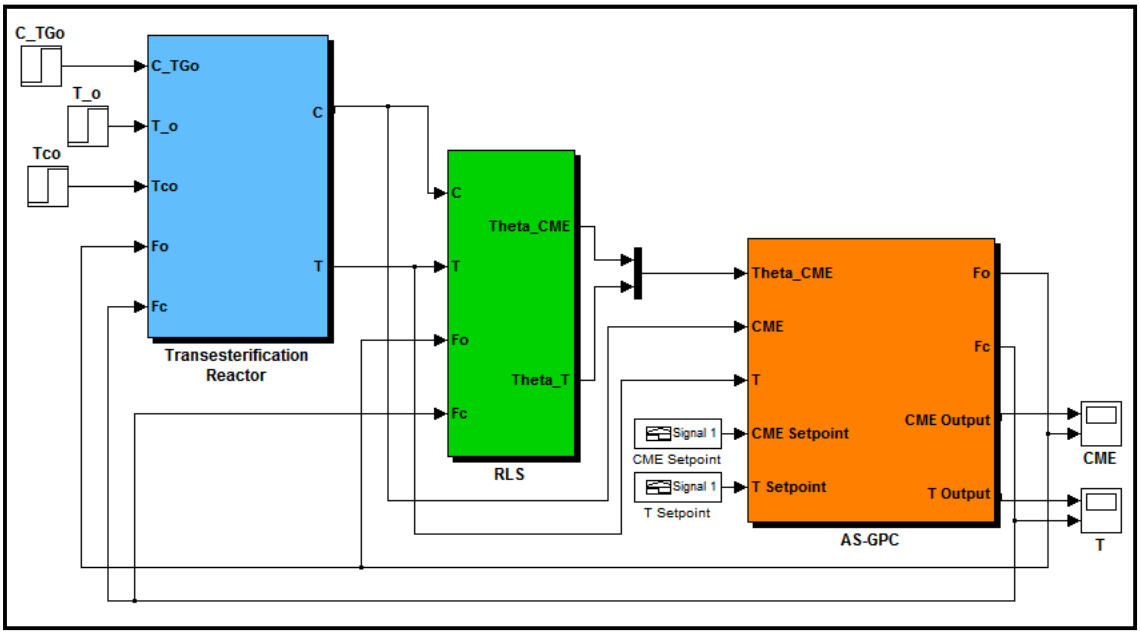

Figure 3.3: Screenshot of AS-GPC implementation in Simulink® version 7.1

The Internal Model Control (IMC) method (Chien & Fruehauf, 1990; Rivera, Morari, & Skogestad, 1986) and the Ziegler-Nichols (ZN) method (Ziegler & Nichols, 1942) as shown in Table 3.4 were used to tune the PID controllers. The selection of the IMC design parameter τ_c in this work was based on the rule of thumb proposed by Skogestad (2003, 2004), where $\tau_c = \theta_d$. The C_{ME} and T profiles under closed loop

conditions using the same setpoint changes as described above were plotted and the various performance indicators were assessed.

Table 3.4: Internal Model Control (IMC) and the open loop Ziegler-Nichols (ZN) based PI and PID controller settings, where K_c = proportional gain, τ_I = integral time constant, τ_D = derivative time constant, τ_c = IMC design parameter, K_p = process gain, τ_p = process time constant, and θ_d = process dead time.

Method	**Model**	$\mathbf{K_c}$	$\boldsymbol{\tau_I}$	$\boldsymbol{\tau_D}$
IMC PI	$\frac{K_p e^{-\theta_d s}}{\tau_p s+1}$	$\frac{\tau_p}{K_p(\tau_c+\theta_d)}$	τ_p	-
IMC PID	$\frac{K_p e^{-\theta_d s}}{\tau_p s+1}$	$\frac{\tau_p+\frac{\theta_d}{2}}{K_p\left(\tau_c+\frac{\theta_d}{2}\right)}$	$\tau_p+\frac{\theta_d}{2}$	$\frac{\tau_p\theta_d}{2\tau_p+\theta_d}$
ZN PI	$\frac{K_p e^{-\theta_d s}}{\tau_p s+1}$	$\frac{0.9\tau_p}{K_p\theta_d}$	$3.33\theta_d$	-
ZN PID	$\frac{K_p e^{-\theta_d s}}{\tau_p s+1}$	$\frac{1.2\tau_p}{K_p\theta_d}$	$2.0\theta_d$	$0.5\theta_d$

CHAPTER 4

SCREENING OF VARIOUS RLS ALGORITHMS

4.1 Chapter Overview

To screen the various RLS algorithm and to test their performance, the methods outlined in Section 3.2 were used. First, the effects of the various parameters of different RLS algorithms were explored to discover their effects on the sensitivity of different RLS algorithms. Following this, the performance of the various algorithms were compared and analyzed before a decision was made on the selection of the algorithm to be used in this work. To facilitate discussions, the essential differences (*i.e.* the equations involved for updating the covariance matrix) between the VFF-RLS, EWRLS, and EDF algorithms are given in Table 4.1. For a complete presentation of the equations, readers are referred to Section 2.3.2.

Table 4.1: Summary of essential differences between the VFF-RLS, EWRLS and EDF algorithms.

Algorithm	How the Covariance Matrix is Updated	Reference of Equation
VFF-RLS	1) Select a value for σ, then calculate the forgetting factor, λ_k:	
	$\lambda_k = 1 - \frac{\boldsymbol{\varepsilon}_k^T \boldsymbol{\varepsilon}_k}{\sigma\left[1 + \boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1} \boldsymbol{\psi}_k\right]}$	Eqn. (2.14)
	2) Use λ_k to update the covariance matrix $\boldsymbol{P}$:	
	$\boldsymbol{\omega}_k = \boldsymbol{P}_{k-1} - \boldsymbol{\gamma}_k \boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1}$	Eqn. (2.15)
	$\boldsymbol{P}_k = \begin{cases} \boldsymbol{\omega}_k / \lambda_k & \text{if trace of } \boldsymbol{\omega}_k / \lambda_k \le C \\ \boldsymbol{\omega}_k & \text{otherwise} \end{cases}$	Eqn. (2.16)
EWRLS	1) Select a value for ρ, then calculate the forgetting factor, λ_k:	
	$L_k = -NINT\left[\rho \boldsymbol{\varepsilon}_k^T \boldsymbol{\varepsilon}_k\right]$	Eqn. (2.18)
	$\lambda_k = \lambda_{\min} + \left(1 - \lambda_{\min}\right) \cdot 2^{L_k}$	Eqn. (2.19)
	2) Use λ_k to update the covariance matrix $\boldsymbol{P}$:	
	$\boldsymbol{P}_k = \frac{1}{\lambda}\left[\boldsymbol{P}_{k-1} - \frac{\boldsymbol{P}_{k-1} \boldsymbol{\psi}_k \boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1}}{\lambda + \boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1} \boldsymbol{\psi}_k}\right]$	Eqn. (2.8)
EDF	1) Select values for λ and δ, then calculate the value of β_k:	
	$r_k = \boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1} \boldsymbol{\psi}_k$	Eqn. (2.21)
	$\beta_k = \begin{cases} \lambda - \frac{1-\lambda}{r_k} & \text{if } r_k > 0 \\ 1 & \text{if } r_k = 0 \end{cases}$	Eqn. (2.23)
	2) Use β_k to update the covariance matrix $\boldsymbol{P}$:	
	$\boldsymbol{P}_k = \boldsymbol{P}_{k-1} - \frac{\boldsymbol{P}_{k-1} \boldsymbol{\psi}_k \boldsymbol{\psi}_k^T \boldsymbol{P}_{k-1}}{\beta_k^{-1} + r_k} + \delta \boldsymbol{I}$	Eqn. (2.24)

4.2 Effect of σ on the Performance of the VFF-RLS Algorithm

In the VFF-RLS algorithm, the design parameter σ can be adjusted to control the sensitivity of the forgetting factor (λ_k) towards the changes in the prediction error at every time step. The effects of the parameter σ on the convergence and the steady state mis-adjustment error of the VFF-RLS algorithm are shown in Figure 4.1 through the Euclidean norm of the parameter estimation error. At low values of σ, from Eqn. (2.14), the corresponding values of the forgetting factor are generally lower ($\lambda_k << 1$). This would mean that in general, great amount of old data are forgotten at every time step. At a small value of σ = 0.1, the Euclidean norm of the parameter estimation error fluctuates, which suggests that the algorithm forgets a great amount of past data, even the important ones. The remaining data are not sufficient for the algorithm to capture the dynamics of the system; hence the Euclidean norm of the parameter estimation error fluctuates, as though the process model parameters are always changing. At high values of $\sigma \geq 10$, huge values of the forgetting factor ($\lambda_k \approx 1$) are obtained at every time step, causing the VFF-RLS algorithm to remember most data so that the effects of small changes in the most recent prediction error are not reflected significantly in the adaptation of the parameter estimates. Thus the algorithm is insensitive to changes in the true system parameters and the Euclidean norm of the parameter estimation error remains at relatively high values. The guiding principle is that moderate values of $\sigma \sim 1$ are good values to be selected for implementation.

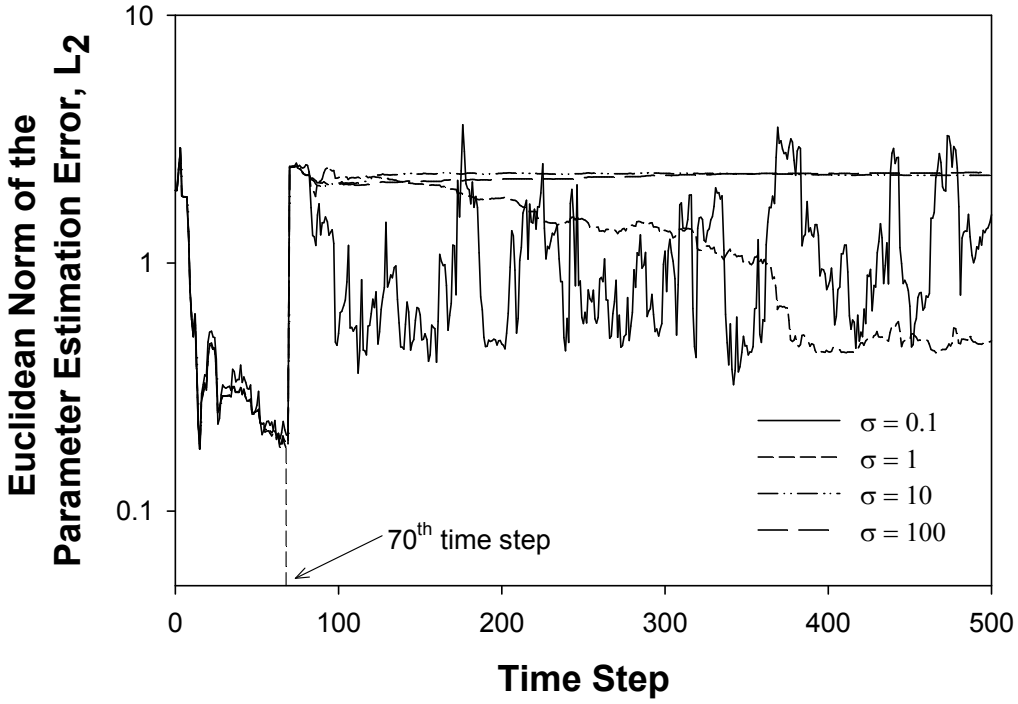

Figure 4.1: Effects of the design parameter σ on the performance of the VFF-RLS (Variable Forgetting Factor Recursive Least Squares) algorithm in tracking changes in parameters at the 70th time step for the hypothetical system in Section 3.2.

4.3 Effect of ρ on the Performance of the EWRLS Algorithm

In the EWRLS algorithm, the design parameter ρ can be adjusted to control the sensitivity of the forgetting factor towards changes in the prediction error at every time step. The Euclidean norm of the parameter estimation error for the EWRLS algorithm is shown in Figure 4.2. The same reasoning used to explain the effects of the parameter σ on the performance of the VFF-RLS algorithm can be used here to explain the effects of the parameter ρ on the performance of the EWRLS algorithm, since both algorithms used the same approach in handling time-varying system parameters. From the figure, at high values of ρ = 100, based on Eqns. (2.18) and (2.19), low values of forgetting factor ($\lambda_k << 1$) are obtained, which would affect the performance of the algorithm in capturing the dynamics of the system. On the other hand, at low values of ρ ≤ 1, the

resulting value of the forgetting factor obtained at every time step is closer to unity; hence the algorithm is insensitive to changes in the true system parameters. From the results, values of $\rho \sim 10$ are suitable to be selected for implementation.

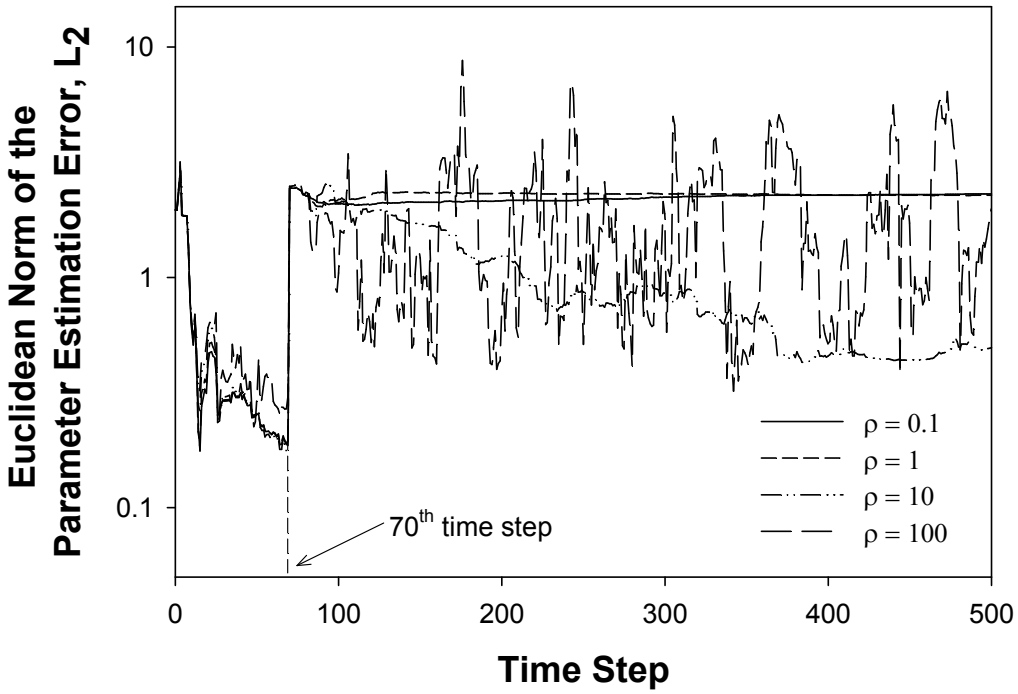

Figure 4.2: Effects of the design parameter ρ on the performance of the EWRLS (Exponential Weighting Recursive Least Squares) algorithm in tracking changes in parameters at the 70th time step for the hypothetical system in Section 3.2.

4.4 Effects of λ and δ on the Performance of the EDF Algorithm

Two parameters are important in the EDF algorithm in controlling the sensitivity of the algorithm towards parameter changes, *viz.* the constant forgetting factor λ and the Bittanti factor δ. The Euclidean norm of the parameter estimation error is shown in Figure 4.3. A 2^2 factorial design was used to pair different values of $\lambda \in (0,1]$ and $\delta \in [0, 0.01]$ introduced to the algorithm. From the figure, it can be inferred that the EDF algorithm tracks changes in process model parameters sufficiently well

under relatively high values of the forgetting factor λ and the Bittanti factor δ. At low values of the forgetting factor λ and the Bittanti factor δ, the Euclidean norm of the parameter estimation error remains at relatively large values, hence it can be concluded that these low values of the forgetting factor λ and the Bittanti factor δ causes the algorithm to be insensitive to changes in true system parameters. From the results, the combination of λ ~ 0.985 and δ ~ 0.01 is suitable to be selected for implementation.

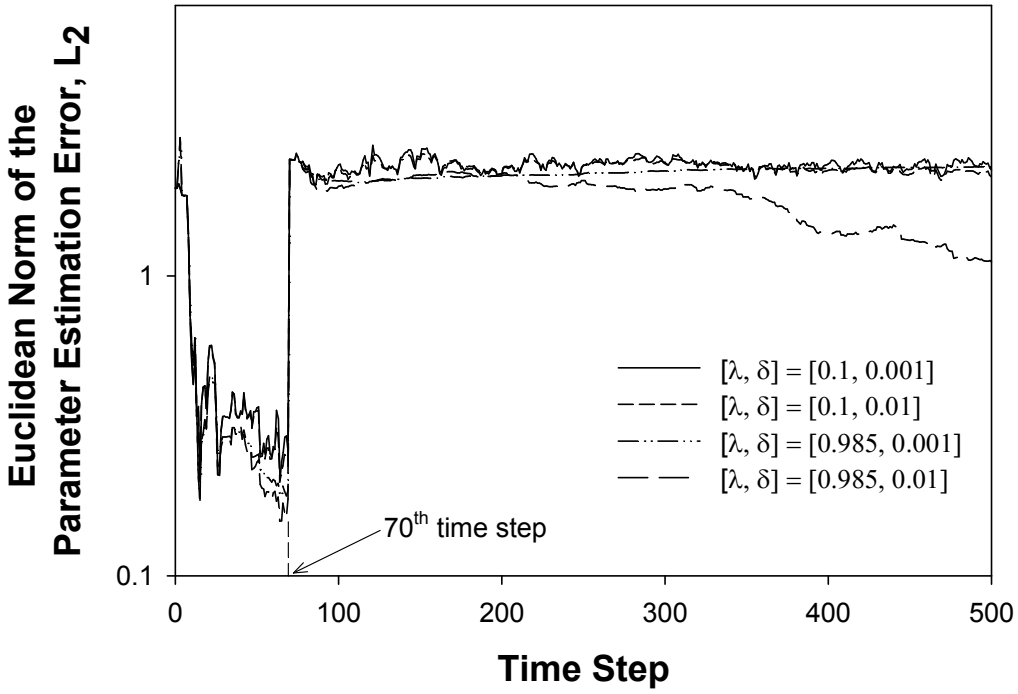

Figure 4.3: Effects of the forgetting factor λ and the Bittanti factor δ on the performance of the EDF (Exponential and Directional Forgetting) algorithm in tracking changes in parameters at the 70th time step for the hypothetical system in Section 3.2.

4.5 Performance Comparison between the VFF-RLS, EWRLS, and EDF Algorithms

Before comparing the performance of the various RLS algorithms, it is useful to summarize the recommended values of the design parameters for each RLS algorithm observed from the results above, as shown in Table 4.2. Based on these values, in each

of these algorithms (*i.e.* the VFF-RLS algorithm, the EWRLS algorithm, and the EDF algorithm), the respective design parameters that affect the sensitivity of the algorithm towards changes in true system parameters, as described by Eqns. (3.1) - (3.2) on p. 44, were tuned by trial and error such that the performance is the best in each variant.

Table 4.2: Summary of recommended values of the design parameters for all RLS algorithms

Algorithm	Design Parameter	Recommended Values
VFF-RLS	σ	~ 1
EWRLS	ρ	~ 10
EDF	$[\lambda, \delta]$	[~0.985, ~0.01]

Figure 4.4 shows the performance of different RLS algorithms in tracking parameter changes. The conventional RLS algorithm was also included as a reference. From the figure, the convergence of the VFF-RLS algorithm and the EWRLS algorithm are the fastest among all the algorithms. However, the steady state mis-adjustment error of the VFF-RLS algorithm is slightly lower than the EWRLS algorithm. The EDF algorithm is slow in tracking parameter changes as the Euclidean norm of the parameter estimation error decreases slowly with time. As observed in Figure 4.4 and in agreement with most literature (Mikleš & Fikar, 2007; Shah & Cluett, 1991), the performance of the conventional RLS algorithm is poor in tracking time-varying system parameters. From Figures 4.1 – 4.4, the VFF-RLS algorithm and the EWRLS algorithm are the preferable algorithms to be implemented in the subsequent studies on the biodiesel reactor, because these algorithms have easily adjustable parameters to control the sensitivity of the forgetting factor towards the changes in the prediction error. These

design parameters can have substantial influence on the performance of the algorithms in identifying time-varying systems. The EDF algorithm, as illustrated in Figure 4.3, has very narrow ranges of adjustable design parameter windows $\left(\lambda \in (0,1] \text{ and } \delta \in [0, 0.01]\right)$ and the various combinations of the two design parameters do not have significant effects on the performance of the EDF algorithm relative to the VFF-RLS algorithm and the EWRLS algorithm.

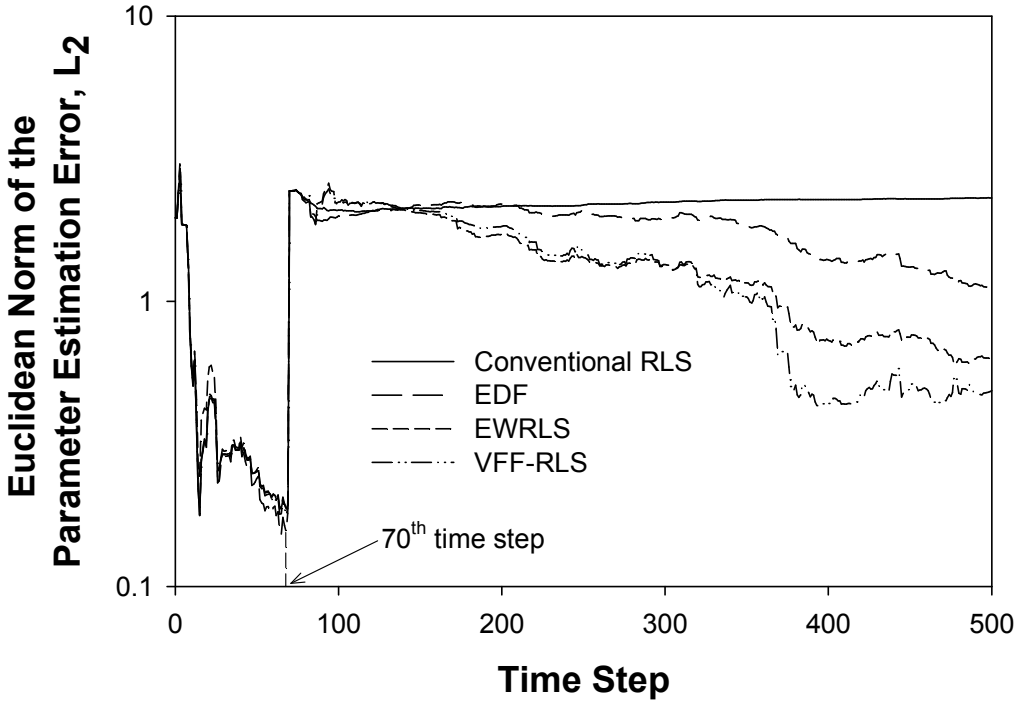

Figure 4.4: The relative performance of three variants of the Recursive Least Squares (RLS) algorithms against the conventional RLS in tracking changes in parameters at the 70th time step for the hypothetical system in Section 3.2: VFF-RLS is the Variable Forgetting Factor RLS algorithm (σ = 1), EWRLS is the Exponential Weighting RLS algorithm (ρ = 8), EDF is the Exponential and Directional Forgetting algorithm ([λ, δ] = [0.985, 0.01]).

4.6 Choice of RLS

In the previous section, the results showed that the VFF-RLS algorithm and the EWRLS algorithm are the more preferable recursive parameter estimation algorithms to be implemented in this work. Since the reliability of the VFF-RLS algorithm has been tested and validated by various researchers in adaptive control applications (Corrêa, Corrêa, & Freire, 2002; Ho, Mjalli, & Yeoh, 2010a; Ydstie, Kershenbaum, & Sargent, 1985), it shall be used throughout this work for parameter estimation purposes. However, the selection of the VFF-RLS algorithm in this work is by no means implying the inferiority of the EWRLS algorithm, as proven by the successful implementation in the author's recent work (Ho, Mjalli, & Yeoh, 2010b).

CHAPTER 5

OPEN LOOP DYNAMIC SYSTEM ANALYSIS AND OFFLINE SYSTEM IDENTIFICATION OF THE BIODIESEL REACTOR

5.1 Chapter Overview

Due to the greater challenges in designing advanced controllers as compared to designing the conventional PID controllers (*e.g.* requiring a deeper understanding of controller design knowledge, increased computational complexity *etc*.), the use of any advanced control algorithm on a particular process must be well justified. In this chapter, the open loop dynamics of the transesterification reactor were scrutinized to show the inherent nonlinearities and process interactions involved, thus justifying the use of the AS-GPC as advocated in this work. The procedures for obtaining the open loop transients are as stated in Section 3.4.

5.2 Dynamic System Analysis on the Biodiesel Reactor

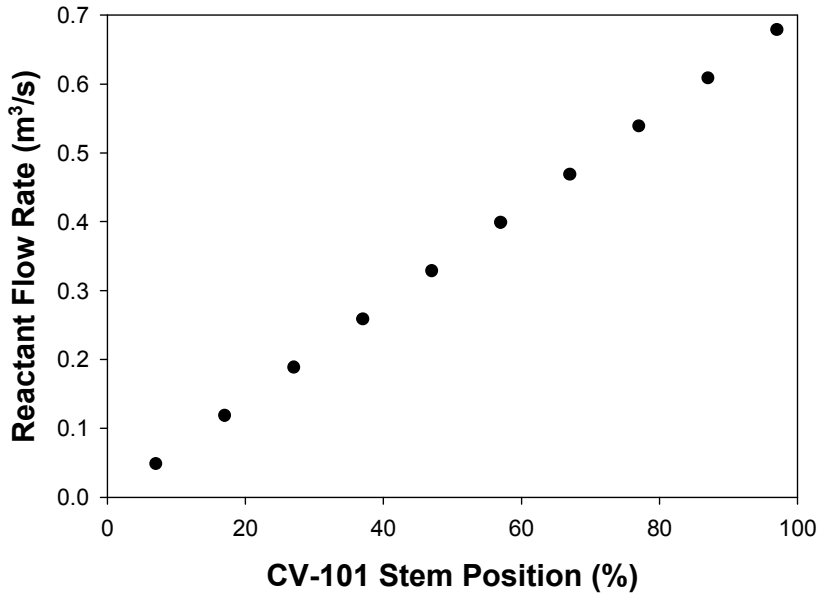

Figure 5.1: Flow characteristics of the control valve CV-101 for manipulating the reactant flow rate (F_o).

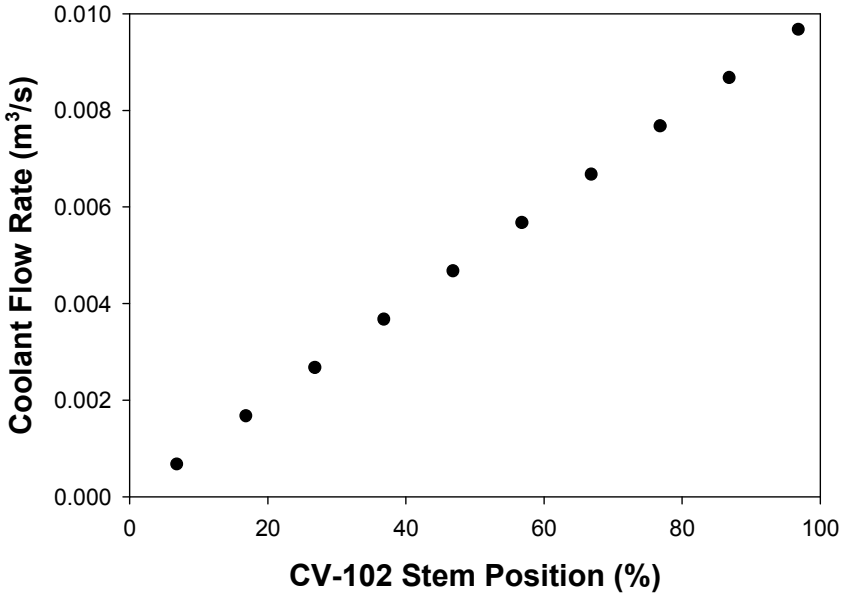

Figure 5.2: Flow characteristics of the control valve CV-102 for manipulating the coolant flow rate (F_c).

In this work, all control valves (*i.e*. CV-101 and CV-102 as shown in Figure 1.3, p. 8) were assumed linear in the simulations (as illustrated in Figures 5.1 – 5.2). Figures 5.3 – 5.4 show the open loop transients of C_{ME} and T as the valve stem position for CV-101 (and consequently F_o in a proportional manner) was increased/decreased at intervals of 3000 s across the entire input region with each step size of 10 % in magnitude. The valve stem position for CV-102 was held constant at 26.8 %, thus maintaining a constant F_c of 0.00268 m^3/s. From the figures, although the increments/decrements in the valve stem position of CV-101 were kept constant at every step change, the open loop transients for both C_{ME} and T exhibited increments/decrements of varying sizes. While changes in the valve stem position of CV-101 at low input regions produced huge changes in both C_{ME} and T, the same changes applied at higher input regions produced only marginal changes in the output variables. As such, the dynamics of the process are operating point dependent, and such process nonlinearities are known to be challenging issues in process control. Furthermore, since the loop pairings of F_o – C_{ME} and F_c – T as proposed by Mjalli, Lee, Kiew, and Hussain (2009) were retained in this work, the drifts in the T profile (while F_c was held constant by a constant valve stem position for CV-102) in these figures suggest the presence of significant process interactions which further complicates the process control scenario. Since the set of ordinary differential equations (*cf*. Mjalli, Lee, Kiew, & Hussain, 2009) describing the biodiesel reactor was nonlinear and coupled, the above observations were expected.

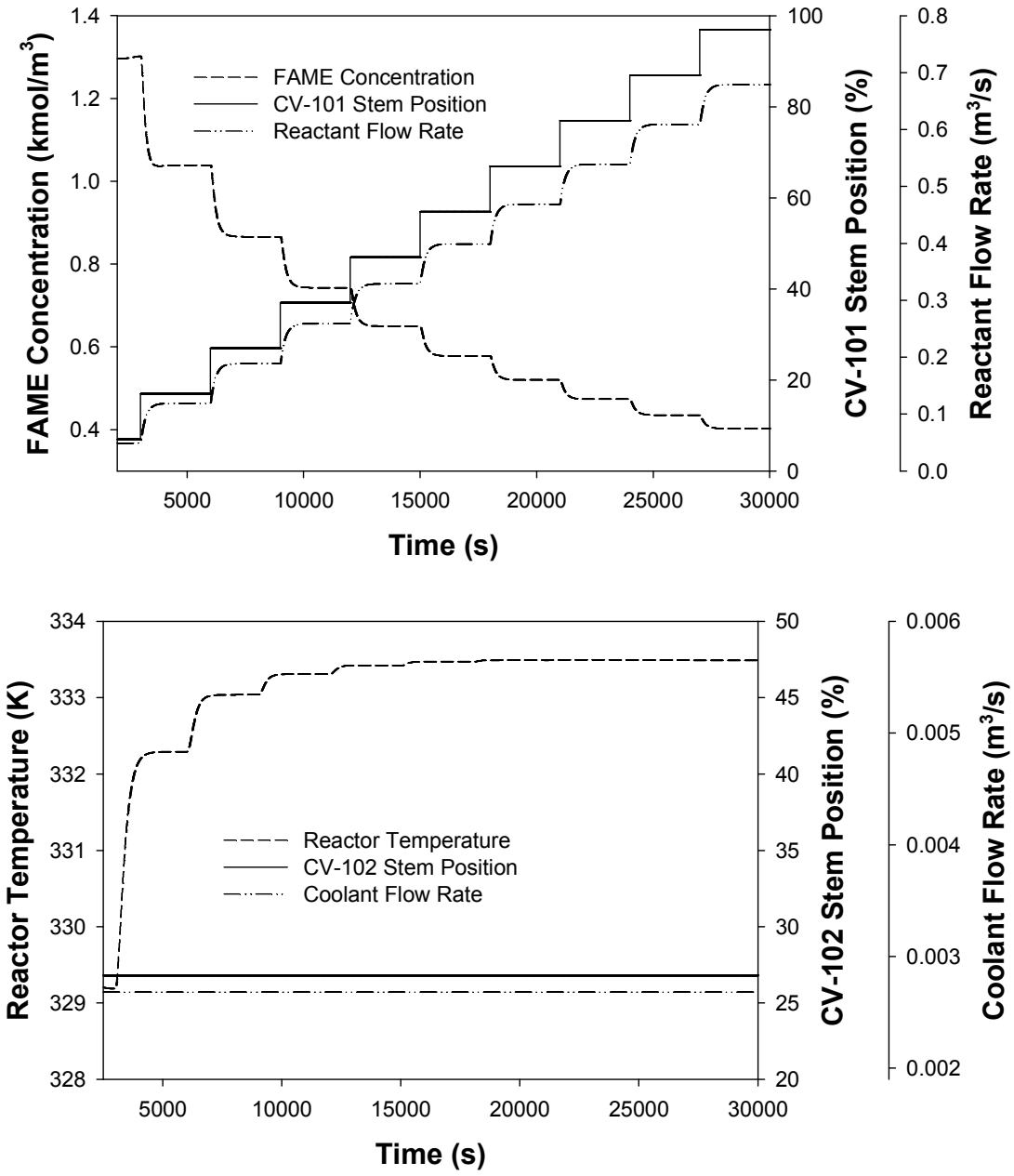

Figure 5.3: Open loop transients of the FAME concentration (C_{ME}) and the reactor temperature (T) as the stem position for CV-101, which was used to manipulate the reactant flow rate (F_o), was increased at intervals of 3000 s across the entire input region with step increments of 10% each. The coolant flow rate (F_c) was held constant by maintaining the stem position for CV-102 at 26.8 %.

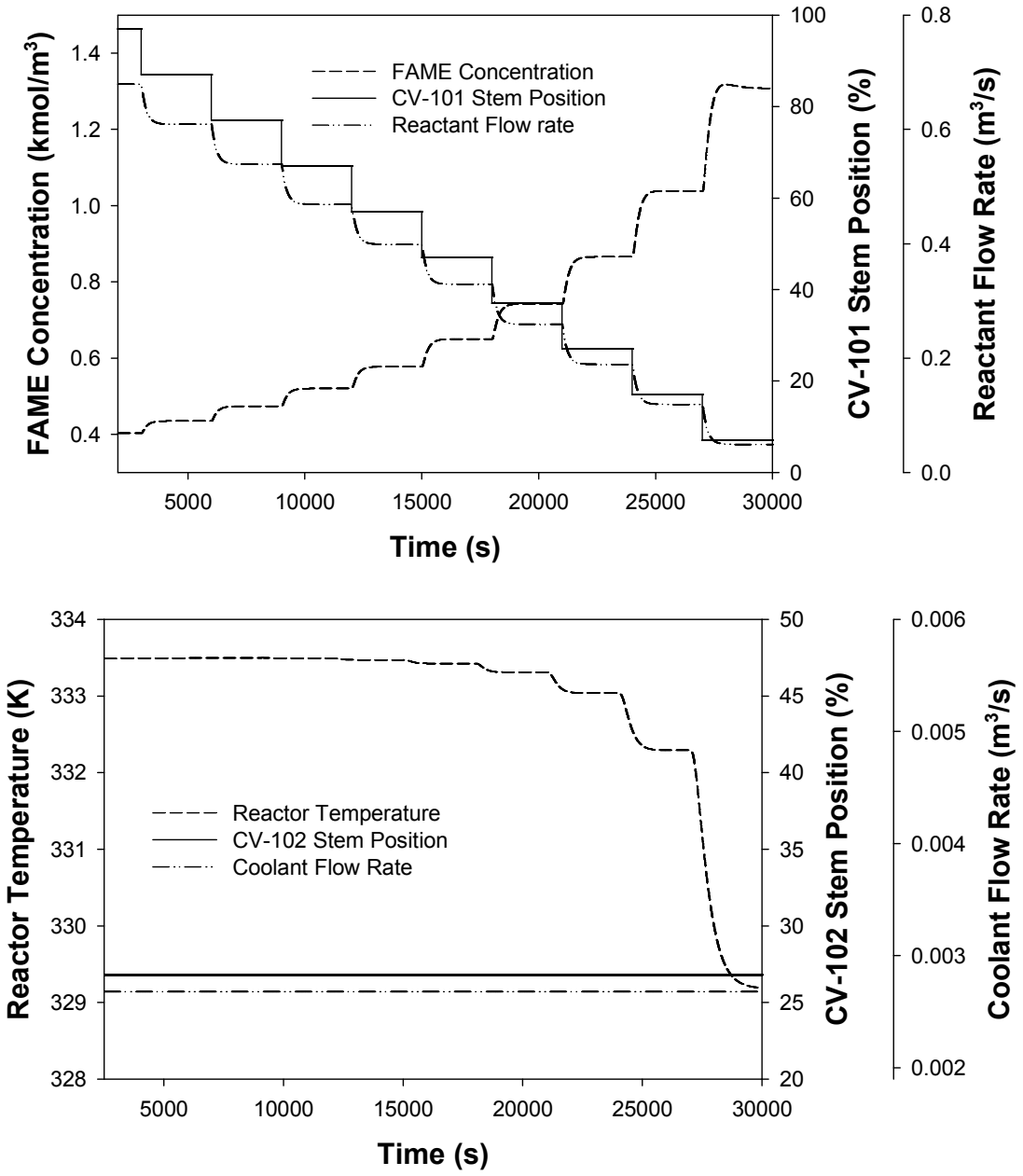

Figure 5.4: Open loop transients of the FAME concentration (C_{ME}) and the reactor temperature (T) as the stem position for CV-101, which was used to manipulate the reactant flow rate (F_o), was decreased at intervals of 3000 s across the entire input region with decrement step sizes of 10 % each. The coolant flow rate (F_c) was held constant by maintaining the stem position for CV-102 at 26.8 %.

As opposed to Figures 5.3 – 5.4 (where the stem position of CV-101 was varied while the stem position of CV-102 was held constant), Figures 5.5 – 5.6 show the open loop transients of C_{ME} and T as the stem position for CV-102 was increased/decreased at intervals of 3000 s across the entire input region with each increment/decrement in step size of 10 %. The stem position for CV-101 was held constant at 17 %. From the figures, process nonlinearity as pertaining to operating range dependent dynamics was observed, albeit to a lesser extent as compared to those observed in Figures 5.3 – 5.4. The open loop transients for both C_{ME} and T exhibited increments/decrements of slightly varying sizes under the input perturbations as depicted in the figures. Furthermore, the changes in the C_{ME} profiles as observed in these figures suggest that for a 2 × 2 transesterification process, the design of the decentralized controller for the F_o – C_{ME} loop (which is responsible for rejecting any drift in C_{ME} arising from changes in F_c) cannot ignore the effects of process interaction.

To further explore the dynamics of the transesterification process, instead of having a single input perturbation as in previous figures, the valve stem positions for both CV-101 and CV-102 were manipulated concurrently in manners as shown in Figures 5.7 – 5.10. In general, the nonlinearity patterns exhibited by previous figures were also present. The transient behaviors of the C_{ME} and T profiles within specified operating ranges were seen to be strictly local, where no two same process responses were observed. Furthermore, due to process interactions, “overshoots” were observed in Figures 5.7 and 5.10 for the open loop T profiles. As such, the biodiesel reactor is a nonlinear process due to both the operating point dependent dynamics and the process interactions observed, hence justifying the need for the implementation of an advanced control scheme.

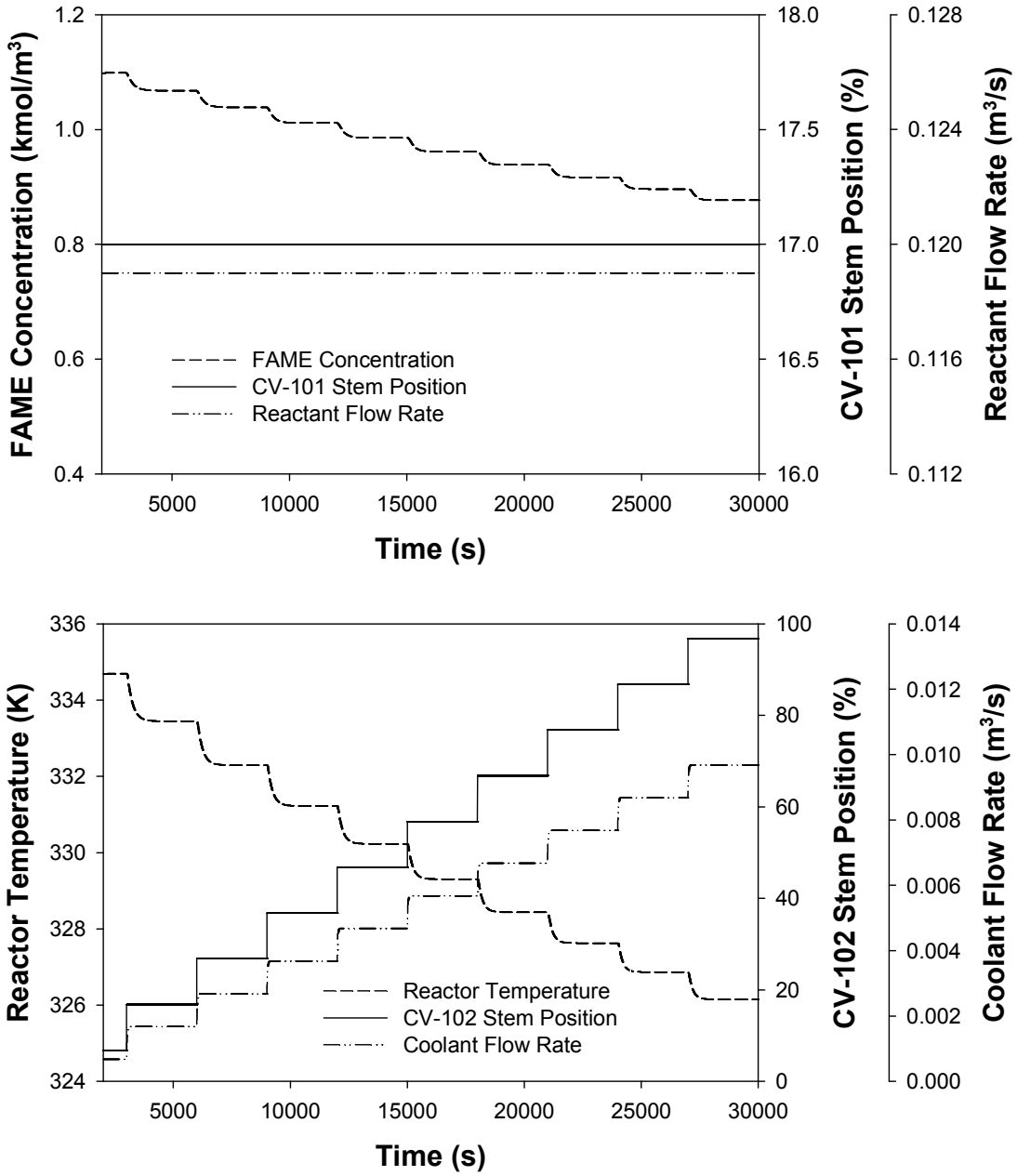

Figure 5.5: Open loop transients of the FAME concentration (C_{ME}) and the reactor temperature (T) as the stem position for CV-102, which was used to manipulate the coolant flow rate (F_c), was increased at intervals of 3000 s across the entire input region with each individual increment in step size of magnitude 10 %. The reactant flow rate (F_o) was held constant by maintaining the stem position for CV-101 at 17 %.

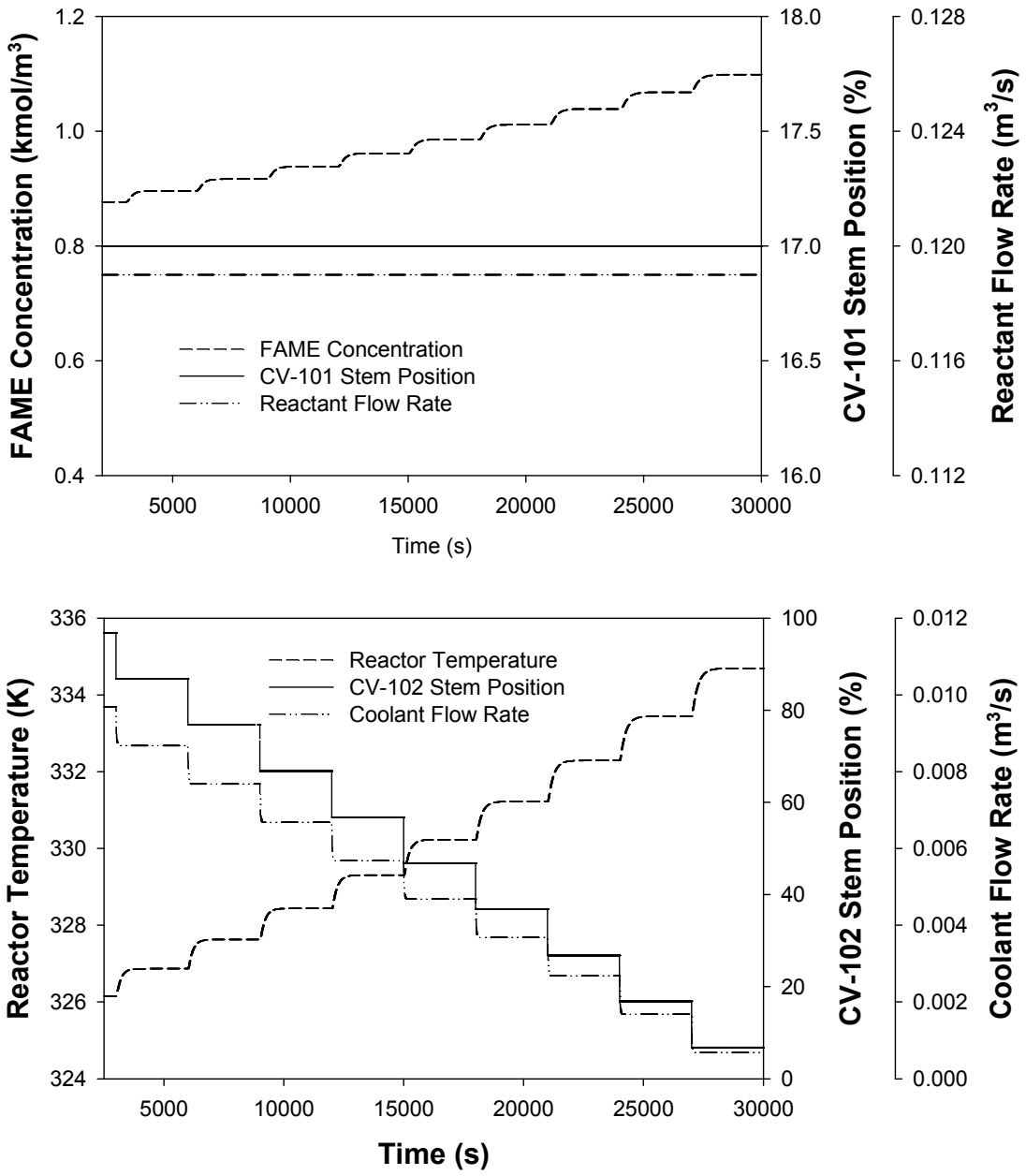

Figure 5.6: Open loop transients of the FAME concentration (C_{ME}) and the reactor temperature (T) as the stem position for CV-102, which was used to manipulate the coolant flow rate (F_c), was decreased at intervals of 3000 s across the entire input region with each individual decrement in step size of magnitude 10 %. The reactant flow rate (F_o) was held constant by maintaining the stem position for CV-101 at 17 %.

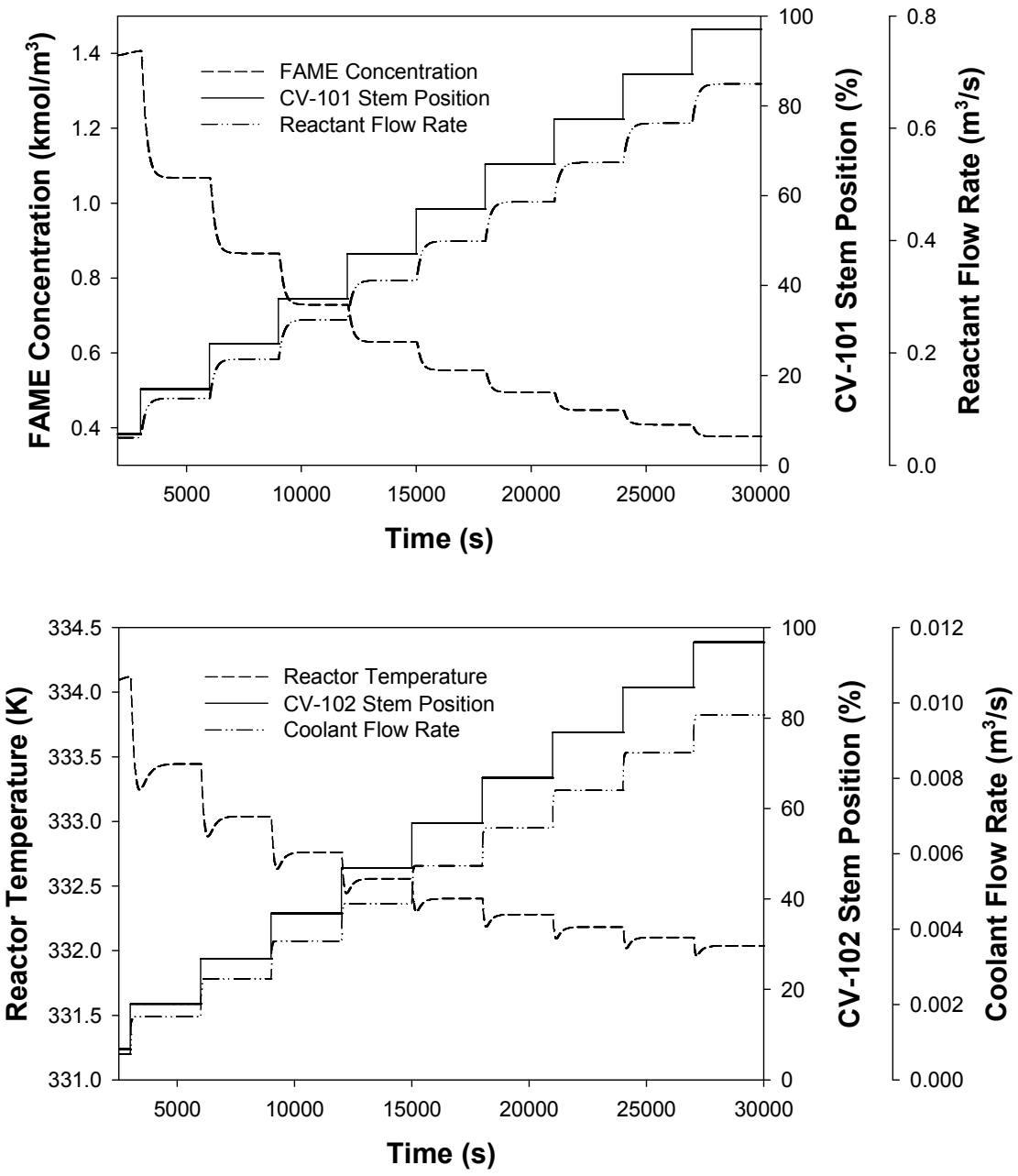

Figure 5.7: Open loop transients of the FAME concentration (C_{ME}) and the reactor temperature (T) as the stem positions for CV-101 and CV-102, and consequently the reactant flow rate (F_o) and coolant flow rate (F_c), were increased at intervals of 3000 s across the entire input region with each individual increment in step size of magnitude 10 %.

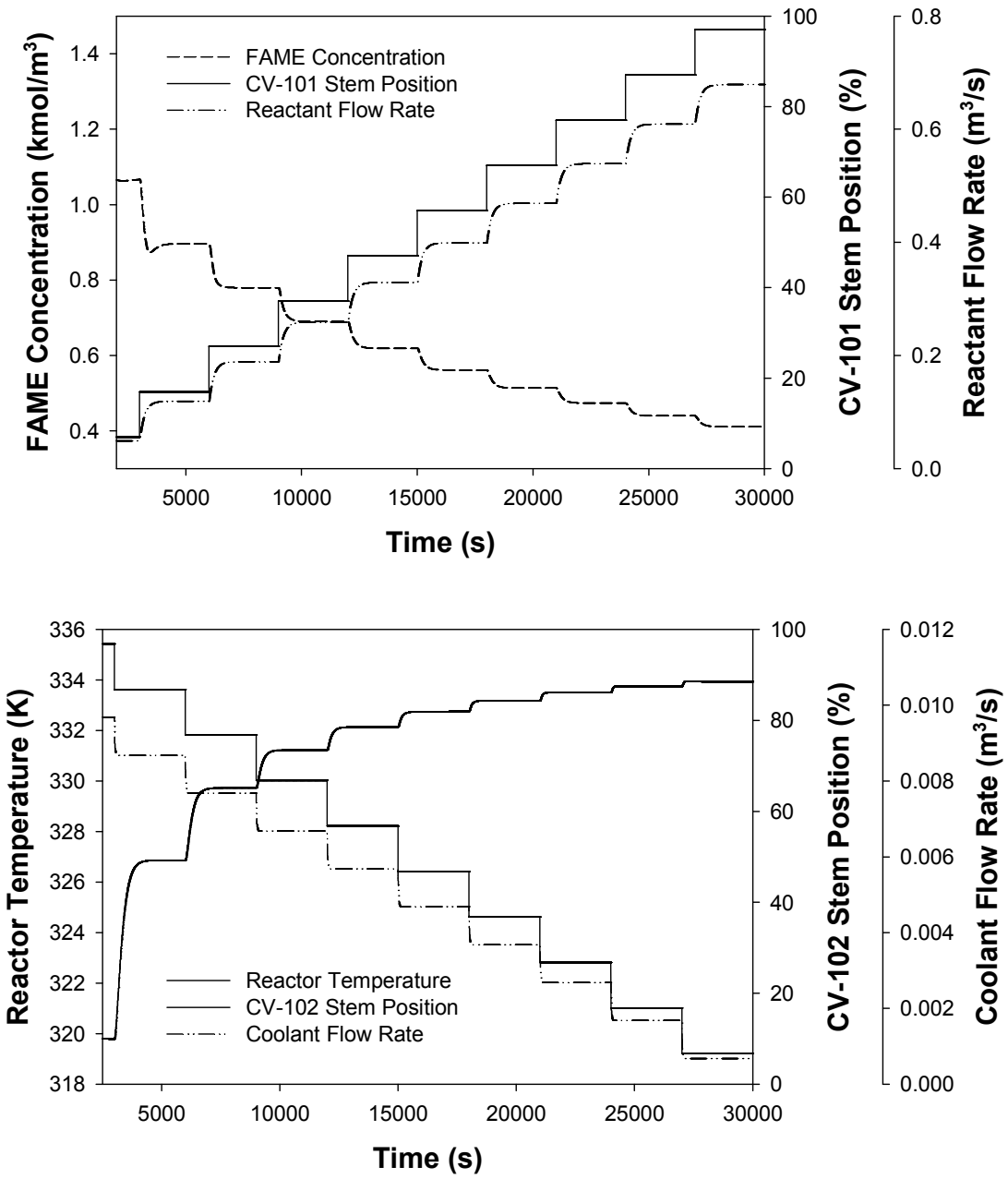

Figure 5.8: Open loop transients of the FAME concentration (C_{ME}) and the reactor temperature (T) as the stem positions for CV-101 and CV-102, and consequently the reactant flow rate (F_o) and coolant flow rate (F_c), were varied in opposite directions (*i.e.* the former had an ascending trend, while the latter had a descending trend) at intervals of 3000 s across the entire input regions with each individual change in step size of magnitude 10 %.

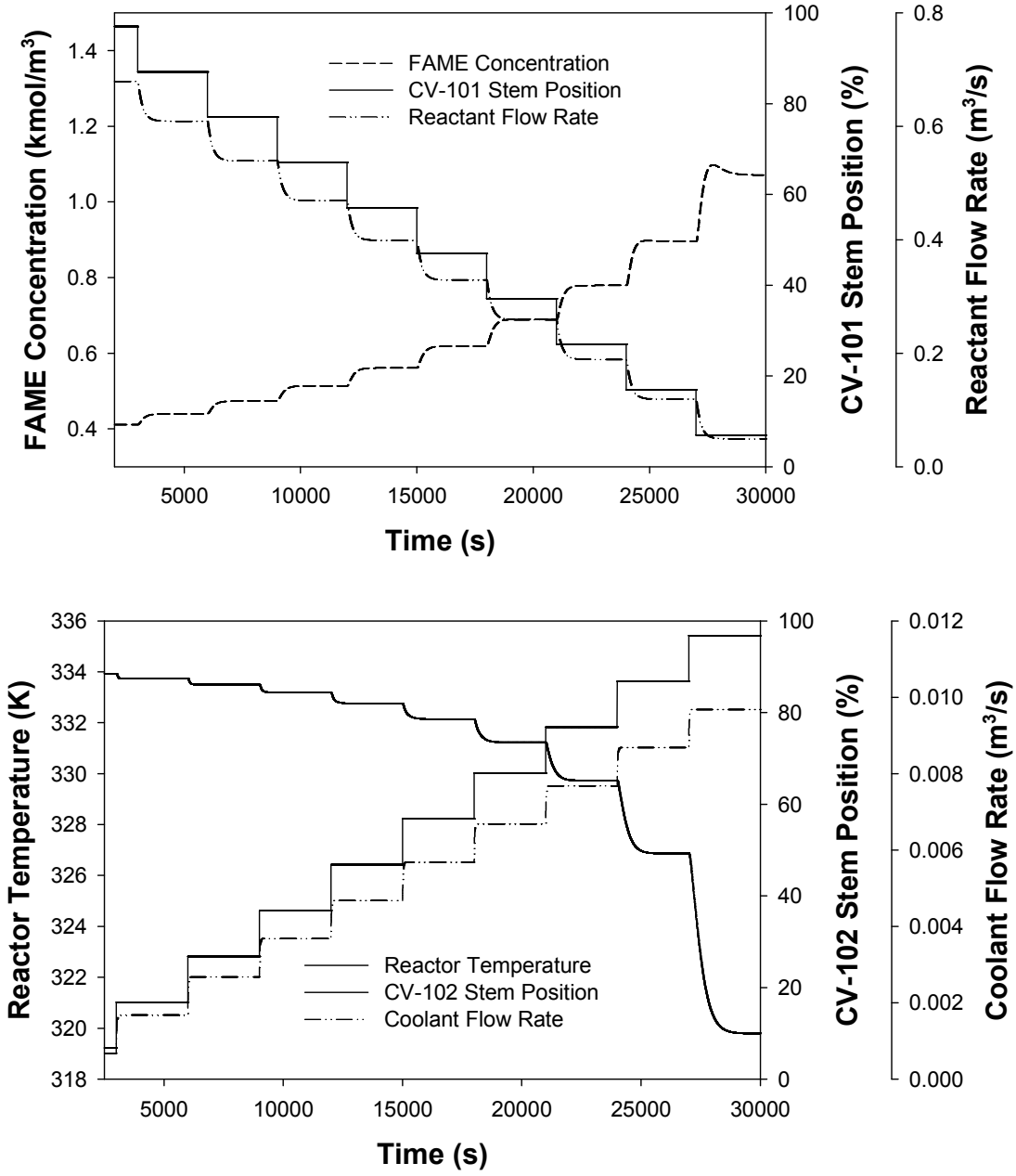

Figure 5.9: Open loop transients of the FAME concentration (C_{ME}) and the reactor temperature (T) as the stem positions for CV-101 and CV-102, and consequently the reactant flow rate (F_o) and coolant flow rate (F_c), were varied in opposite directions (*i.e.* the former had a descending trend, while the latter had an ascending trend) at intervals of 3000 s across the entire input regions with each individual change in step size of magnitude 10 %.

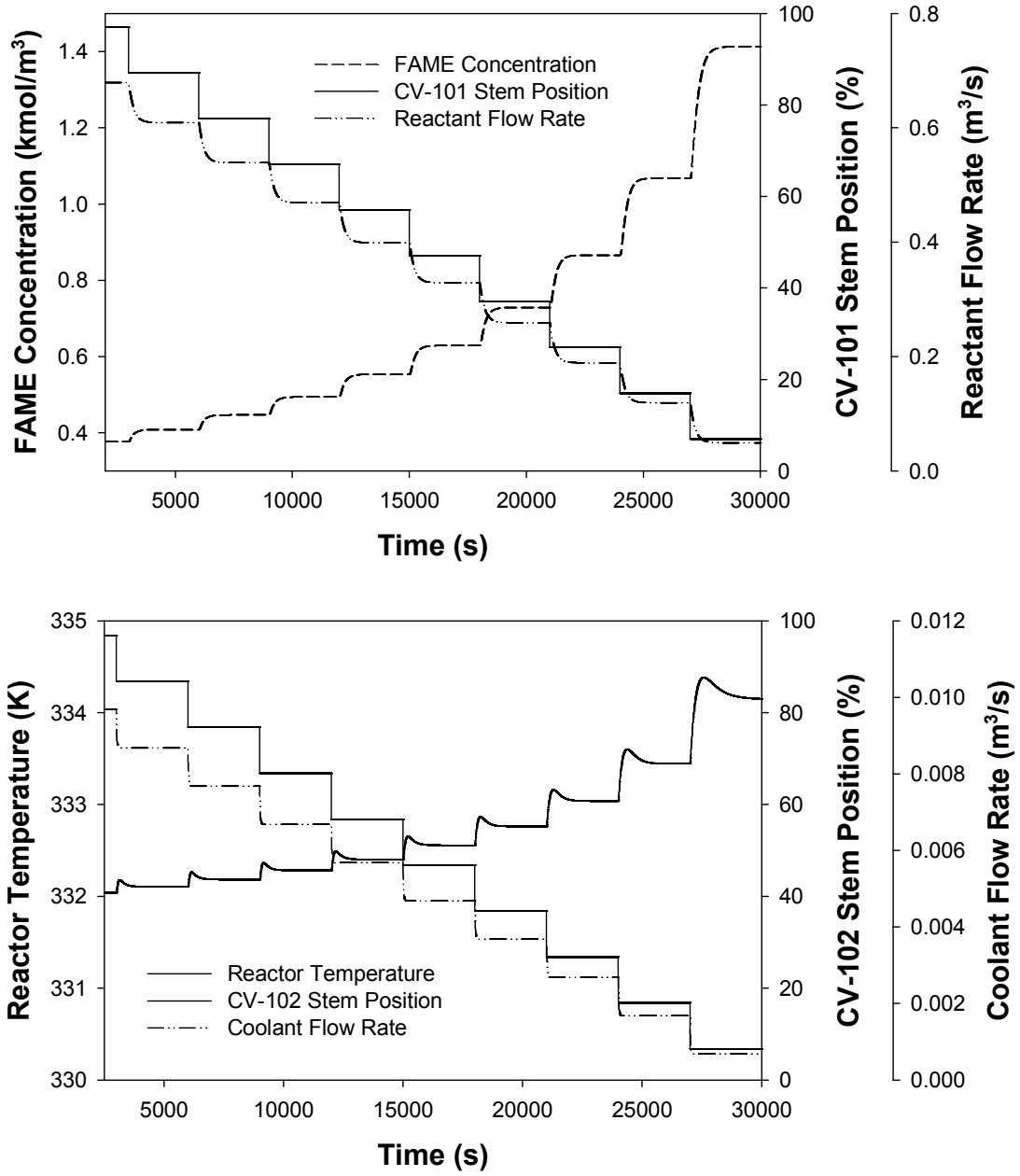

Figure 5.10: Open loop transients of the FAME concentration (C_{ME}) and the reactor temperature (T) as the stem positions for CV-101 and CV-102, and consequently the reactant flow rate (F_o) and coolant flow rate (F_c), were decreased at intervals of 3000 s across the entire input region with each individual decrement in step size of magnitude 10 %.

5.3 Offline First Order Plus Dead Time (FOPDT) System Identification

As alluded to previously in Section 3.5, the data of the open loop transients for the biodiesel reactor is useful for the purpose of estimating the model parameters of a FOPDT model. These estimated model parameters are of particular importance in the design of a non-adaptive, classical GPC (which is needed for comparison of performance to the AS-GPC). Other than that, in the design of the AS-GPC and A-GPC schemes, the estimated model parameters serve as a backup model in the case of failure in the online modeling mechanism, *viz*. the RLS algorithm. Since this study focused on the design of decentralized controllers for the transesterification process, only SISO models were identified here, *viz.* the $F_o - C_{ME}$ and $F_c - T$ relationships. As such, the open loop transients as shown in Figure 5.3 and 5.5 (*i.e.* for $F_o - C_{ME}$ and $F_c - T$ relationships respectively) were chosen for the purpose of offline system identification. Tables 5.1 – 5.2 show the FOPDT model parameters of the $F_o - C_{ME}$ and $F_c - T$ loops respectively. The results also showed that the estimated FOPDT models attained a high degree of fitting capabilities, as depicted by the coefficients of determination (R^2).

Table 5.1: First Order Plus Dead Time (FOPDT) model parameters of the reactant flow rate (F_o) – FAME concentration (C_{ME}) relationship. The offline system identification was performed on the various open loop transients of the C_{ME} profile (as shown in Figure 5.3) across its entire operating range.

Model	Changes in CV-101 Stem Position [%]	$K_p \times 10^3$ [(kmol/m^3)/%]	τ_p [s]	θ_d [s]	R^2
1	7 - 17	-26.5	141.0	60.9	0.9971
2	17 - 27	-17.3	185.1	40.6	0.9995
3	27 - 37	-12.3	193.9	34.5	0.9997
4	37 - 47	-9.3	196.3	30.8	0.9998
5	47 - 57	-7.2	196.8	28.3	0.9998
6	57 - 67	-5.7	197.0	25.8	0.9999
7	67 - 77	-4.7	197.0	23.8	0.9999
8	77 - 87	-3.9	197.0	21.9	0.9999
9	87 - 97	-3.2	196.8	20.5	0.9999

Table 5.2: First Order Plus Dead Time (FOPDT) model parameters of the coolant flow rate (F_c) – reactor temperature (T) relationship. The offline system identification was performed on the various open loop transients of the T profile (as shown in Figure 5.5) across its entire operating range.

Model	Changes in CV-102 Stem Position [%]	$K_p \times 10^2$ [(K)/%]	τ_p [s]	θ_d [s]	R^2
1	6.8 - 16.8	-12.5	224.4	19.8	0.9999
2	16.8 - 26.8	-11.5	215.8	19.8	0.9999
3	26.8 - 36.8	-10.7	207.8	19.7	0.9999
4	36.8 - 46.8	-9.9	200.4	19.6	0.9999
5	46.8 - 56.8	-9.3	193.6	19.6	0.9999
6	56.8 - 66.8	-8.7	187.1	19.5	0.9999
7	66.8 - 76.8	-8.1	181.2	19.4	0.9999
8	76.8 - 86.8	-7.6	175.5	19.4	0.9999
9	86.8 - 96.8	-7.1	170.3	19.3	0.9999

The model parameters shown in the tables above are in the continuous time domain. Suffice to note here that for the purpose of controller design (which will be discussed in the following chapter), the model parameters shown in the tables have to be converted to its discrete time equivalent (*i.e.* a_1, b_1, D, t_s), as described in Section 3.6.

5.4 Concluding Remarks

In this chapter, it was shown through open loop tests that the transesterification reactor is a nonlinear multivariable process. This is further corroborated by the results of the offline system identification shown in Tables 5.1 - 5.2, which show a wide range

of model parameters across the operating ranges of the biodiesel reactor. As such, the control scheme implemented on the reactor must be able to handle these challenges. In the next chapter, the closed loop performance of the AS-GPC scheme on the biodiesel reactor will be analyzed and discussed in detail. Comparisons with the A-GPC, GPC and conventional PID schemes will also be studied.

CHAPTER 6

DESIGN AND IMPLEMENTATION OF THE AS-GPC ON THE BIODIESEL REACTOR

6.1 Chapter Overview

In this chapter, the closed loop performance of the unconstrained AS-GPC is discussed in terms of the setpoint tracking ability, followed by an analysis of the closed loop poles at every sampling time to show that the proposed control scheme in the constraint-free setup is stable for the particular case study and time duration considered. The study on the closed loop poles in the unconstrained case was done with the sole purpose of ensuring the soundness in the controller design for this basic setup. Although the notion of closed loop stability in the unconstrained case carries no meaning when constraints are imposed on the controller, a poorly performing unconstrained AS-GPC cannot be expected to give good performance when the constraints are active. Thus, it is worthwhile to study the closed loop stability of the unconstrained AS-GPC.

To explore the full strength of the proposed control scheme, the constrained AS-GPC was implemented on the biodiesel reactor and compared with the constrained A-GPC and GPC schemes. Unless mentioned otherwise, in general the term “constrained” shall be omitted when referring to the constrained controllers throughout this chapter, be it the AS-GPC, A-GPC or GPC scheme. The performances of these schemes were then explored in terms of the ability in handling servo problem. A comparison with the conventional PID controller, tuned according to best practices (IMC and Ziegler-Nichols), was also included. Towards the end of this chapter, having revealed the

superiority of the AS-GPC scheme, the control scheme was tested for the ability in rejecting load disturbances in the process.

6.2 Control System Design

Before any discussions are presented on the closed loop performance of the various control schemes, it is necessary to first define several important parameters for the various decentralized control schemes as outlined in Section 3.6. The nominal operating conditions of the biodiesel reactor are shown in Table 6.1. Depending on the quality of the feed, the setpoint might have to be adjusted which overrides the nominal operating conditions. For the purpose of model selection for the GPC scheme and backup model selection for the AS-GPC and A-GPC schemes (as well as the calculation of the tuning parameters where applicable), the results of the offline system identification obtained around the nominal operating conditions of the reactor and converted to its discrete time equivalent were used (*i.e*. Model 2 of Table 5.1 and Model 3 of Table 5.2 for the C_{ME} and T loops respectively). This strategy is intuitive as the reactor is expected to operate at the nominal operating conditions should there be no other unexpected scenarios causing a change in the operating conditions. Hence, it is hereafter understood that where fixed model parameters (*viz*. a_1, b_1, D and t_s) or fixed tuning parameters (*viz*. N_1, N_2, and R_i) are involved, they shall be calculated based on these nominal settings. Table 6.2 shows the parameter values of the discrete time first order SISO ARX models (*cf*. Section 2.3.1) used in this work. The values shown in Table 6.2 were adopted for all VFF-RLS based control schemes presented in this study, whether unconstrained or constrained.

Table 6.1: Nominal operating conditions of the biodiesel reactor (Mjalli, Lee, Kiew, & Hussain, 2009)

Parameter	Symbol	Value	Unit
Reactor temperature	T	60	˚C
Reactor pressure	P_r	1	atm
Reactant flow rate	F_o	0.119	m^3/s
Coolant flow rate	F_c	0.00268	m^3/s
Mixer rotational speed	N	6	rps
Initial concentration of triglycerides	C_{TGO}	1.11	$kmol/m^3$
Initial concentration of methanol	C_{AO}	21.499	$kmol/m^3$

Table 6.2: Parameter values of discrete time SISO ARX models for both the FAME concentration (C_{ME}) and the reactor temperature (T) loops.

Parameter	Symbol	C_{ME} Loop	T Loop	Unit
Number of outputs	n	1	1	-
Number of inputs	m	1	1	-
Order of $a(z^{-1})$	α	1	1	-
Order of $b(z^{-1})$	β	1	1	-
Discrete dead time	D	3	1	-
Sampling time	t_s	20	20	s

From the screening results of the various RLS algorithms in Chapter 4, the VFF-RLS algorithm shall be employed. In this case, two VFF-RLS algorithms were designed; one each for the C_{ME} and T loops respectively. Table 6.3 shows the parameter values of the VFF-RLS algorithms for both loops. It is worth mentioning again that these settings were used for all VFF-RLS based control schemes presented in this study. In Table 6.3, the values of σ were chosen based on the rule of $\sigma/\sigma_w \approx 1000$ (Shah & Cluett, 1991), where σ_w is the variance of any process output measurement noise. Although the

transesterification reactor model was deterministic in nature (*i.e.* no noise addition to the process outputs), $\sigma_w = 0$ cannot be selected to avoid division by zero in Eqn. (2.14). Hence, σ_w was assumed to be 0.001 (*i.e.* a very small amount of noise) for both loops. As for the values of C, they were chosen based on process experience. The initial covariance matrix $\boldsymbol{P}_0$ was chosen as a matrix with large numbers in the diagonal for both loops to imply a huge uncertainty in the initial parameter estimates $\hat{\boldsymbol{\theta}}_0^T$, which is a standard setting given in the RLS literature (Mikleš & Fikar, 2007).

Table 6.3: Parameter values of the VFF-RLS algorithms for both the FAME concentration (C_{ME}) and reactor temperature (T) loops.

Parameter	**C_{ME} Loop**	**T Loop**
σ	1	1
C	100	100
$\boldsymbol{P}_0$	$1\times10^6\,\boldsymbol{I}_{3\times3}$	$1\times10^6\,\boldsymbol{I}_{3\times3}$
$\hat{\boldsymbol{\theta}}_0^T$	$\boldsymbol{0}_{1\times3}$	$\boldsymbol{0}_{1\times3}$

One key important feature in implementing the VFF-RLS algorithm is that the algorithm should only be deployed when all process inputs and outputs are at steady states. Furthermore, another critical condition is that there must be sufficient excitation in the system in order for the VFF-RLS to perform correctly, *i.e.* the condition of persistence of excitation (Ljung & Söderstöm, 1983). It has been reported that during closed loop conditions, the addition of perturbation signals as well as perturbation in setpoints can be used to excite the VFF-RLS algorithm to ensure proper convergence of the parameter estimates (Hägglund & Åström, 2000; Mikleš & Fikar, 2007; Seborg, Edgar, & Shah, 1986; Vogel, 1988). Hence, in this study, upon deployment of the VFF-RLS algorithm at time = 500 s (as long as the process is at steady state, the VFF-RLS

algorithm can be deployed at any time and not necessarily at time = 500 s as mentioned), the adaptive mechanisms in the different control schemes (whether the AS-GPC scheme and its unconstrained counterpart or the A-GPC scheme) were only activated 1000 s after the first change in setpoint for each respective loop. The time of adaptation can be chosen arbitrarily on the condition that it must be initiated after the process model parameters converged, which can be observed from the real-time transients of the model parameters in practice. In this case, the first setpoint change in the C_{ME} loop occurred at time = 500 s, whereas for the T loop, the first setpoint change occurred at time = 4000 s. The changes in setpoint for the two loops were introduced at different instances to capture the effect of loop interactions. As an additional measure to safeguard the convergence of the parameter estimator, small amount of white noises with zero mean and variances of 4×10^{-5} $(\text{kmol/m}^3)^2$ and 4×10^{-5} K^2 respectively were added to the data of the C_{ME} and T entering the VFF-RLS algorithm. The values of variance chosen were arbitrary as long as they were sufficiently small.

Tables 6.4 - 6.5 show the values of SISO CARIMA model (*cf.* Section 2.4) parameters and tuning parameters adopted in the AS-GPC, A-GPC and GPC schemes for both the C_{ME} and T loops. It should be noted again that the parameters a_1 and b_1 in the SISO CARIMA representation were estimated by the VFF-RLS algorithm and that $T(z^{-1}) = 1$ was chosen for simplicity. From the tables, the GPC scheme in this case served as a base case controller, where no adaptation mechanism was implemented. As mentioned above, the adaptive mechanisms in the AS-GPC and A-GPC schemes were only activated after the commencement of the first change in setpoint for both loops. The GPC internal model and the backup models for the AS-GPC and A-GPC schemes were calculated from the nominal continuous time equivalent. As for the values of the tuning parameters, they were calculated using the guideline presented in Table 3.3. As

the value of M ranges from 1 - 6 according to the guideline, it was decided that $M = 6$ be selected in this study to allow some robustness in the computation of control moves. In general, the tuning parameters were all fixed for all schemes with the exception that the value of R_i be made adaptive in the AS-GPC scheme.

Table 6.4: Values of SISO CARIMA model parameters [with $T(z^{-1}) = 1$] and tuning parameters of the AS-GPC, A-GPC and GPC schemes for the FAME concentration (C_{ME}) loop. For this loop, the adaptive mechanisms were activated at time = 1500 s (*i.e.* 1000 s after the first change in setpoint at time = 500 s) for the AS-GPC and A-GPC schemes.

Parameter	Symbol	Time	AS-GPC*	A-GPC	GPC
Model parameters	a_1, b_1, D	< 1500 s	a_1 = -0.89758 b_1 = -0.00177 D = 3	a_1 = -0.89758 b_1 = -0.00177 D = 3	a_1 = -0.89758 b_1 = -0.00177 D = 3
		≥ 1500 s	Adaptive	Adaptive	a_1 = -0.89758 b_1 = -0.00177 D = 3
Minimum prediction horizon	N_1	< 1500 s	4	4	4
		≥ 1500 s	4	4	4
Maximum prediction horizon	N_2	< 1500 s	50	50	50
		≥ 1500 s	50	50	50
Control horizon	M	< 1500 s	6	6	6
		≥ 1500 s	6	6	6
Move suppression weight	R_i	< 1500 s	0.000114	0.000114	0.000114
		≥ 1500 s	Adaptive	0.000114	0.000114

* For both constrained and unconstrained cases

Table 6.5: Values of SISO CARIMA model parameters [with $T(z^{-1}) = 1$] and tuning parameters of the AS-GPC, A-GPC and GPC schemes for the reactor temperature (T) loop. For this loop, the adaptive mechanisms were activated at time = 5000 s (*i.e.* 1000 s after the first change in setpoint at time = 4000 s) for the AS-GPC and A-GPC schemes.

Parameter	Symbol	Time	AS-GPC*	A-GPC	GPC
Model parameters	a_1, b_1, D	< 5000 s	a_1 = -0.90825 b_1 = -0.00981 D = 1	a_1 = -0.90825 b_1 = -0.00981 D = 1	a_1 = -0.90825 b_1 = -0.00981 D = 1
		≥ 5000 s	Adaptive	Adaptive	a_1 = -0.90825 b_1 = -0.00981 D = 1
Min. prediction horizon	N_1	< 5000 s	2	2	2
		≥ 5000 s	2	2	2
Max. prediction horizon	N_2	< 5000 s	53	53	53
		≥ 5000 s	53	53	53
Control horizon	M	< 5000 s	6	6	6
		≥ 5000 s	6	6	6
Move suppression weight	R_i	< 5000 s	0.004924	0.004924	0.004924
		≥ 5000 s	Adaptive	0.004924	0.004924

* For both constrained and unconstrained cases

6.3 Unconstrained AS-GPC Implementation and Analysis

Having defined the necessary parameters associated with the AS-GPC scheme as shown above, the proposed scheme without constraints on the controller was first tested on the biodiesel reactor for the performance in servo control. In this case, a series of setpoint changes in opposite directions from the nominal operating conditions of the C_{ME} and T loops was introduced to the process. Figures 6.1 - 6.2 show the C_{ME} and T profiles and the corresponding controller moves under successive, random setpoint changes. In general, the unconstrained AS-GPC performed well in terms of the ability in tracking setpoint changes for both loops. As mentioned above, the adaptive mechanisms in the unconstrained AS-GPC (*i.e*. model adaptation and self-tuning) were only activated at time = 1500 s. Hence, prior to that, chattering in the controller moves for the C_{ME} loop was observed, which caused an overshoot in the C_{ME} profile. The chattering phenomenon was caused by the nonlinearity across the operating region, of which a GPC based on a localized model and fixed tuning parameters was unable to cope. The chattering phenomenon disappeared upon deployment of the adaptive mechanisms in the unconstrained AS-GPC. This is an indication that for a nonlinear process, the adaptive mechanisms in the unconstrained AS-GPC were capable of regulating the controller moves even when the setpoint was changed to another operating region (*i.e*. moved away from the nominal operating condition). The effect of loop interactions was also observed for the C_{ME} loop, albeit in small magnitudes. As for the T loop, the unconstrained AS-GPC demonstrated good performance in terms of the ability to attain minimal overshoot and negligible effect of loop interactions. However, the controller moves observed were slightly aggressive, where slew rates of approximately 20 % were observed for a single actuator movement. Furthermore, actuator saturation was also observed at time = 18000 s (otherwise not observed

throughout the entire simulation), which contributed to a slight overshoot in the T profile. Although these observations were still practically realizable, clearly the situation could be improved if constraints were imposed on the controller. Despite all the minor shortcomings of the constraint-free AS-GPC, suffice to note here that the unconstrained AS-GPC generally performed well, with maximum rise times of 600 s and 500 s for the C_{ME} and T loops respectively. In addition, the maximum settling times for both loops were also observed to be identical to the maximum rise times.

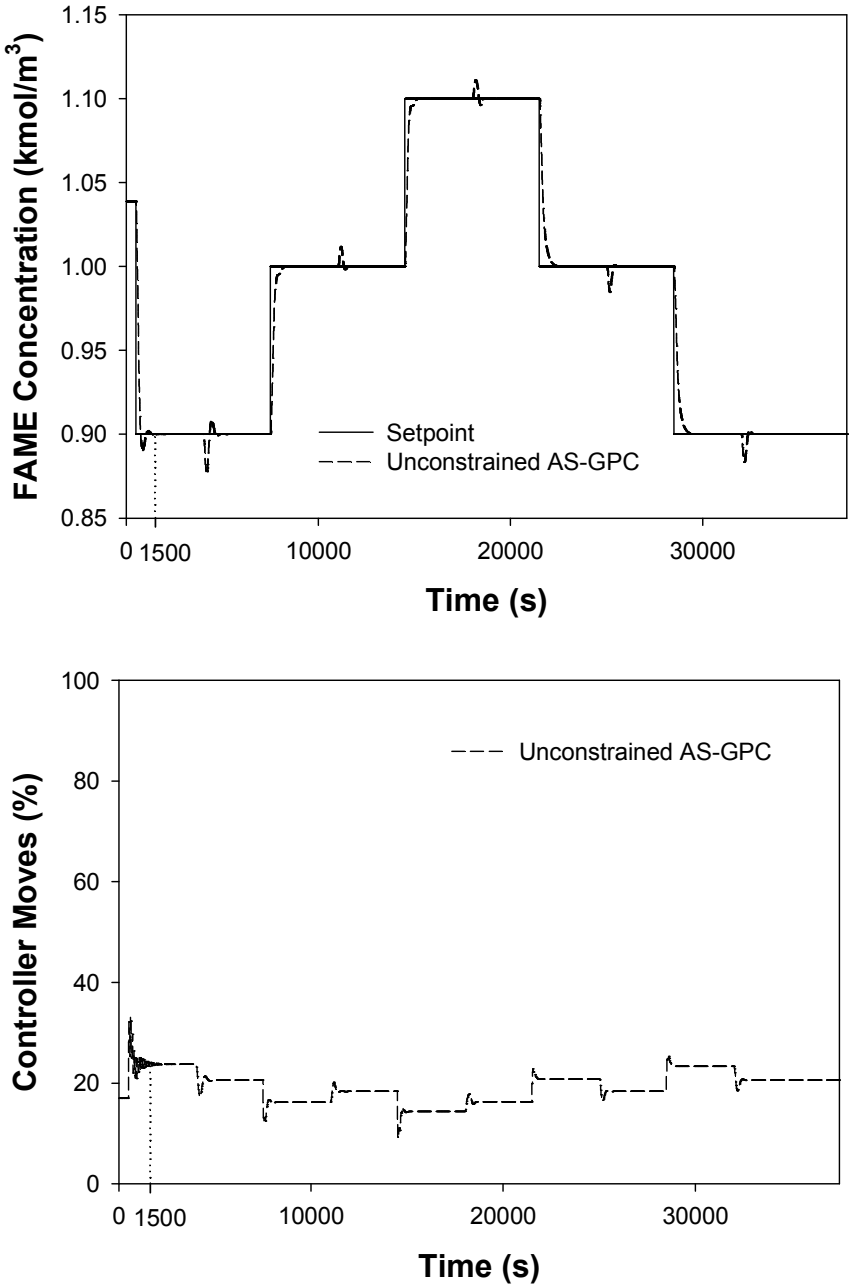

Figure 6.1: Performance of the unconstrained AS-GPC in tracking changes in setpoint for the FAME concentration (C_{ME}) loop and its corresponding controller moves. Model adaptation and self-tuning of the controller for this loop was activated at time = 1500 s.

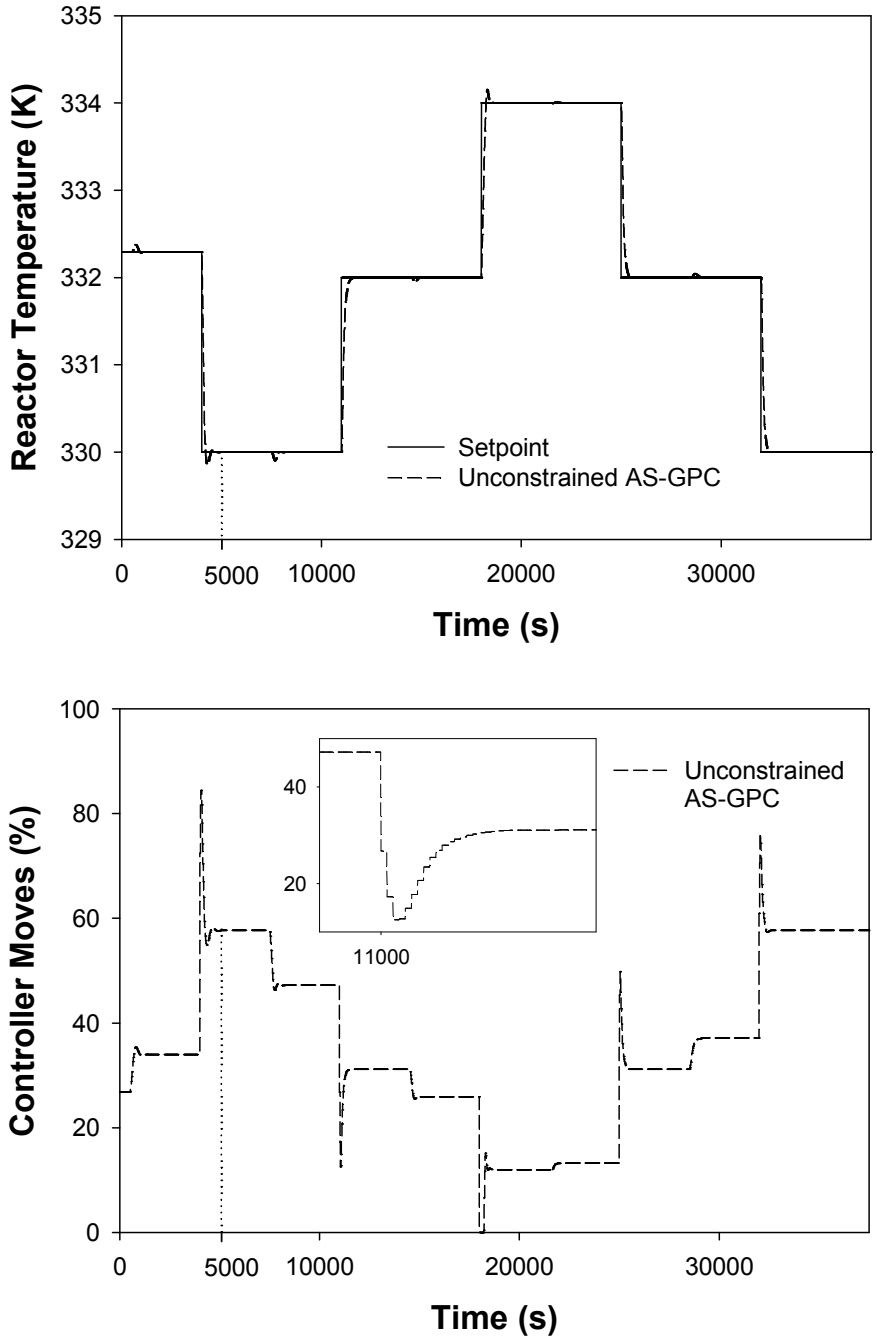

Figure 6.2: Performance of the unconstrained AS-GPC in tracking changes in setpoint for the reactor temperature (T) loop and its corresponding controller moves. Model adaptation and self-tuning of the controller for this loop was activated at time = 5000 s.

Figures 6.3 - 6.4 show the superpositions of the locations of closed loop poles (found with equation 2.43) in the *z*-domain evaluated at every sampling time for the unconstrained AS-GPC for both the C_{ME} and the T loops. In general, 5 closed loop poles were obtained for the C_{ME} loop, whereas 3 closed loop poles were obtained for the

T loop. The greater number of poles calculated for the C_{ME} loop as compared to the T loop was due to the larger dead time of the model imposed, resulting in higher order dynamics of the input for the C_{ME} loop. In general, all the closed loop poles for both loops were located within the unit circle, implying stable closed loop dynamics during the simulation interval. The evaluation of the closed loop poles in this case serves as an *a posteriori* check on the stability of the closed loop at a particular time instance. The general idea is that for as long as the evaluated closed loop poles are stable during implementation, the system is stable but the stability at the current instance is by no means an indication of future stability. For both the C_{ME} and T loops, only a pair of conjugate poles was observed to be distant from the origin, leaving the rest very near to zero. Thus, it can be inferred that while the poles near the origin were prone to producing near deadbeat behavior in the controller performance at the expense of aggressive controller moves, the single pair of conjugate poles some distance away from the origin tend to dampen the controller moves in such a way that a good compromise was achieved between aggressive controller moves and fast closed loop response. It should also be noted in the unconstrained AS-GPC framework, the model parameters adopted by the controller as well as the move suppression weights were changing continuously to adapt to the changing dynamics of the process. As such, the closed loop poles (which were dependent on the model parameters and the tuning parameters in the form of the closed loop characteristic equation) were not static in the locations. Instead, as observed in the figures, the location of each pole spread within certain vicinity in the *z*-domain. As aforementioned, although the stability results obtained from the analysis of the locations of unconstrained closed loop poles here cannot be used to assess the closed loop stability of the constrained AS-GPC, nonetheless the analysis here verified the soundness of the basic unconstrained

controller setup, without which it is impossible to further extend the current scheme to also include active constraints.

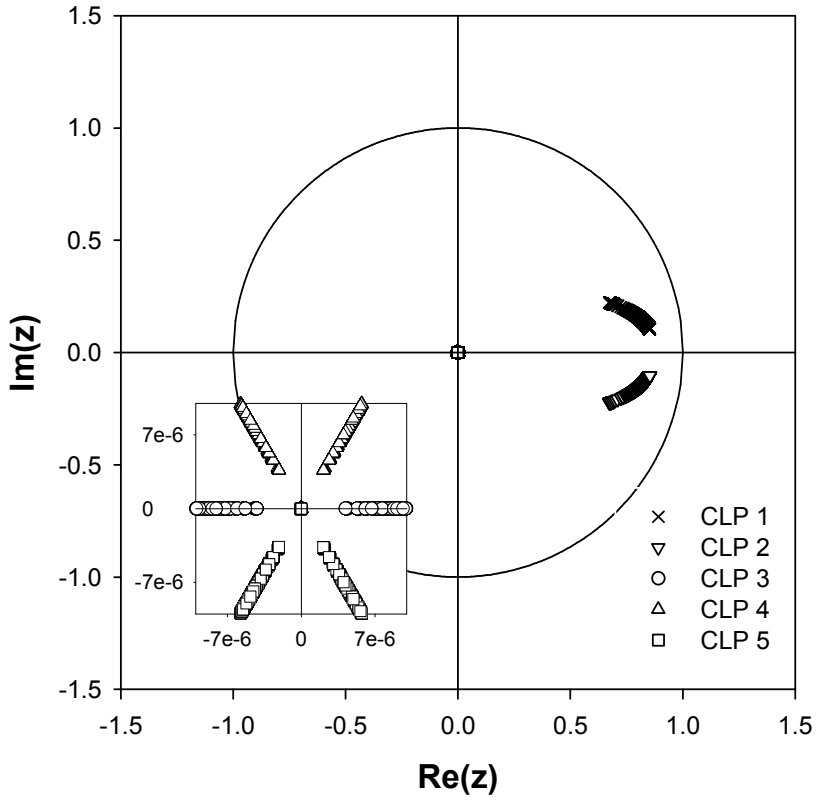

Figure 6.3: Locations of Closed Loop Poles (CLP) for the unconstrained AS-GPC in the *z*-domain for the FAME concentration (C_{ME}) loop. Five CLPs were obtained at every control interval, and shown here are superpositions of the CLPs obtained throughout the entire simulation. The unit circle was plotted to indicate the boundary of stability, where CLPs located within the unit circle stabilize the process.

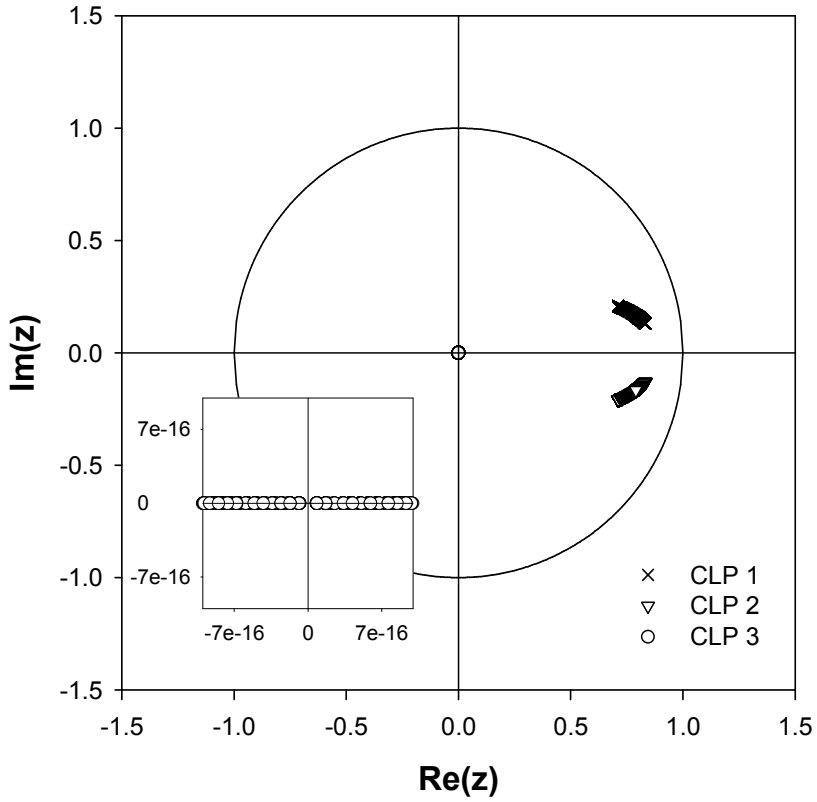

Figure 6.4: Locations of Closed Loop Poles (CLP) for the unconstrained AS-GPC in the *z*-domain for the reactor temperature (T) loop. Three CLPs were obtained at every control interval, and shown here are superpositions of the CLPs obtained throughout the entire simulation. The unit circle was plotted to indicate the boundary of stability, where CLPs located within the unit circle stabilize the process.

6.4 Relative Performance of the Constrained AS-GPC Scheme

Having validated the functionality of the unconstrained AS-GPC, the constrained AS-GPC scheme was deployed to explore the full strength of the controller. Comparisons with the constrained A-GPC and GPC schemes were also included to demonstrate the improved efficacy of the newly proposed AS-GPC scheme. Table 6.6 shows the constraints imposed on all three control schemes under comparison.

Table 6.6: Constraints imposed on the AS-GPC, A-GPC and GPC schemes expressed primarily in flow rates (m^3/s). The corresponding valve stem positions (%) are given in parentheses.

Parameters	Symbols	Lower Limit	Upper Limit
Reactant flow rate	F_o	2.10×10^{-2} m^3/s (3 %)	6.79×10^{-1} m^3/s (97 %)
Coolant flow rate	F_c	3.00×10^{-6} m^3/s (3 %)	9.70×10^{-5} m^3/s (97 %)
Reactant slew rate	ΔF_o	-3.50×10^{-2} m^3/s (-5 %)	3.50×10^{-2} m^3/s (5 %)
Coolant slew rate	ΔF_c	-5.00×10^{-6} m^3/s (-5 %)	5.00×10^{-6} m^3/s (5 %)

The setpoint changes as used above to test the performance of the unconstrained AS-GPC were retained. The results of the simulations were shown in Figures 6.5 - 6.8. From the figures, a general observation is that the GPC scheme showed fast closed loop responses with slight overshoots for both loops. Moreover, the effect of loop interactions was significant. Chattering behavior in the controller moves were also observed for the C_{ME} loop throughout the entire simulation. On the other hand, as expected, the shortcomings exhibited by the GPC scheme were mitigated to a certain extent by having model adaptation in the controller as in the A-GPC scheme. In this case, the A-GPC scheme successfully tackled the issue of overshoots, while maintaining similar speed of closed loop transients as in the GPC scheme. The extent of loop interactions observed was also less significant as compared to those observed for the GPC scheme. Further, the controller moves for the T loop produced by the A-GPC scheme showed a narrower range of operation for the same magnitude of change in setpoint as compared to those exhibited by the GPC scheme. Although the C_{ME} and T profiles as well as the controller moves for the T loop exhibited by the A-GPC scheme

showed improved performance, the controller moves exhibited by the A-GPC scheme were still not satisfactory for the C_{ME} loop, where chatterings were still observed.

The discussions thus far showed that model adaptation alone was not sufficient to ensure good performance in all performance indicators assessed, particularly when dealing with a nonlinear process such as the transesterification reactor. However, with the deployment of the AS-GPC (as illustrated in the same figures), the downsides of the A-GPC and GPC schemes previously mentioned were not noticeable. Not only did the AS-GPC scheme retained similar fast closed loop responses and the good loop interactions handling capability as demonstrated by the A-GPC scheme for both loops, the proposed scheme produced far better and smoother controller moves for the C_{ME} loop (as shown in Figure 6.6). Although the controller moves for the T loop did not show significant improvements as compared to those observed for the A-GPC scheme, nonetheless the controller moves were sufficiently smooth for practical implementation. In general, it can be concluded that the performance of the AS-GPC scheme was superior to that of the A-GPC and GPC schemes. Moreover, contrasting the performance of the unconstrained AS-GPC as shown in Figures 6.1 - 6.2, constrained AS-GPC had no issues with actuator saturation or excessive slew rates. In addition, although the biodiesel reactor process seemed to be largely FOPDT, simulation results showed that without a proper model and appropriate tuning parameters, the predictive controller was not able to perform as expected even for a seemingly FOPDT process. Hence, the use of the biodiesel reactor model was adequate to demonstrate the relevance of the AS-GPC.

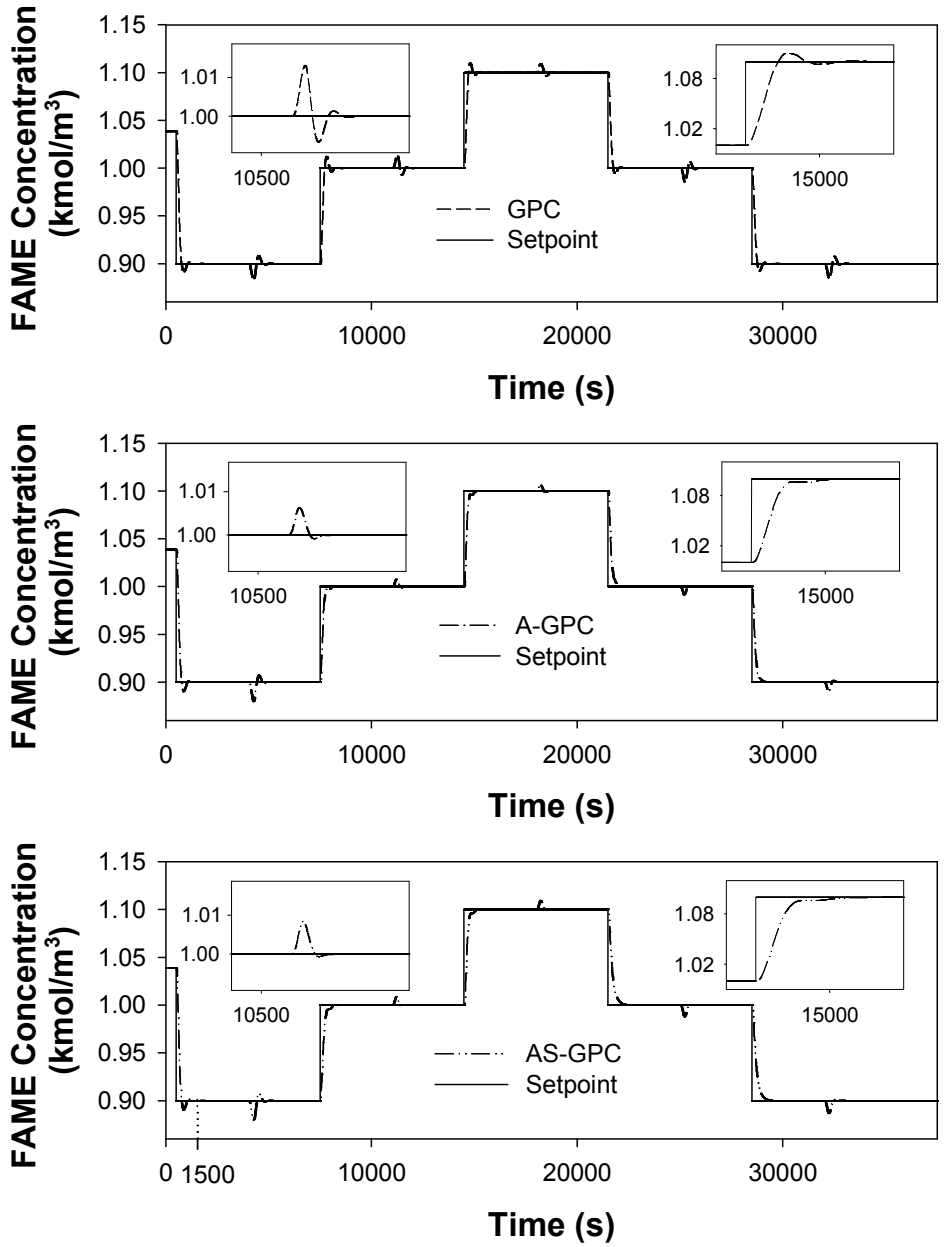

Figure 6.5: Comparison of performance between the GPC, A-GPC and AS-GPC schemes in tracking a series of changes in setpoint for the FAME concentration (C_{ME}) loop. Model adaptation and self-tuning of the AS-GPC for this loop was activated at time = 1500 s.

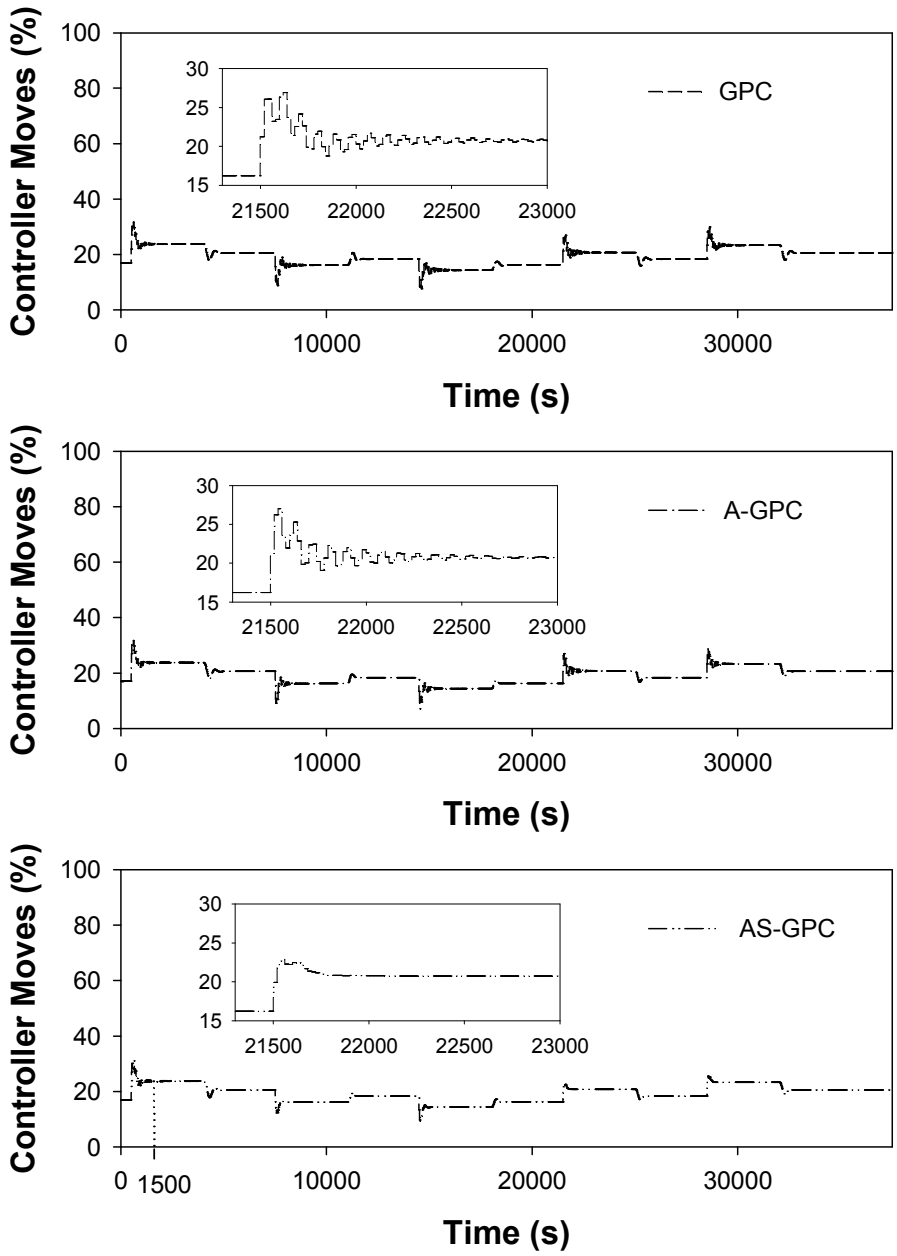

Figure 6.6: Controller moves produced by the GPC, A-GPC and AS-GPC schemes in tracking a series of changes in setpoint for the FAME concentration (C_{ME}) loop.

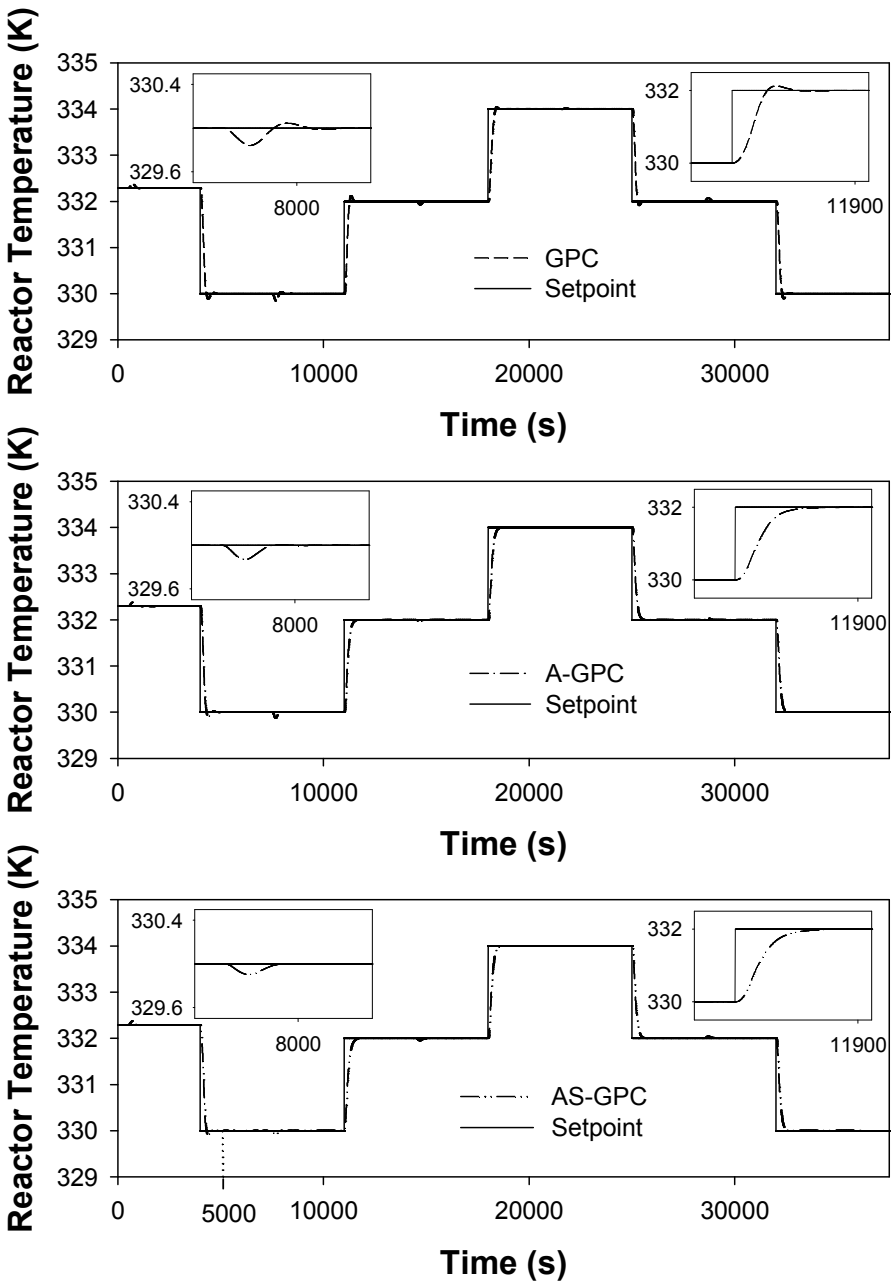

Figure 6.7: Comparison of performance between the GPC, A-GPC and AS-GPC schemes in tracking a series of changes in setpoint for the reactor temperature (T) loop. Model adaptation and self-tuning of the controller for this loop was activated at time = 5000 s.

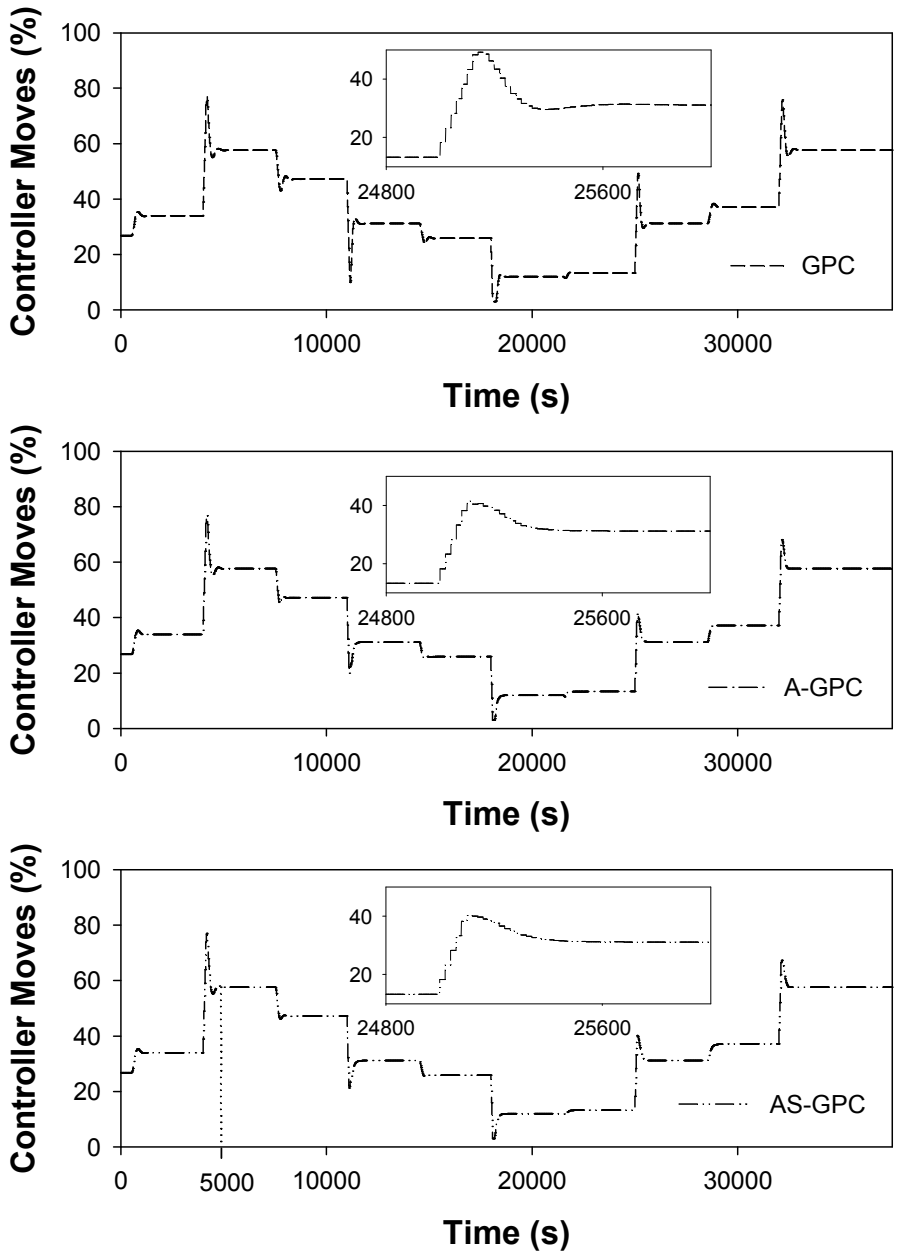

Figure 6.8: Controller moves produced by the GPC, A-GPC and AS-GPC schemes in tracking a series of changes in setpoint for the reactor temperature (T) loop. Model adaptation and self-tuning of the controller for this loop was activated at time = 5000 s.

Figures 6.9 - 6.10 show the respective transient of the process model parameters identified by the VFF-RLS algorithm as well as those eventually adopted as the internal model of the AS-GPC scheme for the C_{ME} and T loop. To ease referencing in discussions below, the parameters estimated by the VFF-RLS algorithm shall be represented by a_1 and b_1, whereas those eventually adopted in the controller shall be

represented by a_1' and b_1'. At the onset of simulation, these figures demonstrate that while the VFF-RLS algorithm had yet to stabilize, the values of a_1 and b_1 estimated by the VFF-RLS algorithm were not employed as the internal model of the AS-GPC scheme. Instead, a pre-calculated set of model parameters were used. The figures also show that the estimated process model parameters converged approximately 500 s after the first change in setpoint for both loops (the first change in setpoint occurred at 500 s and 4000 s for the C_{ME} and T loops respectively). The parameters did not converge at first due to insufficient excitation at the onset of simulation where the process was at the steady state. As soon as a setpoint change was introduced, the controller began to respond, which provided necessary excitation to the VFF-RLS algorithm for successful system identification. To ensure that adaptation in the controller was based on converged parameters, as indicated previously, the adaptive mechanisms in the AS-GPC were only activated 1000 s after the first change in setpoint for both loops, *i.e.* the controller eventually adopts the estimated parameters produced by the VFF-RLS algorithm for control calculations.

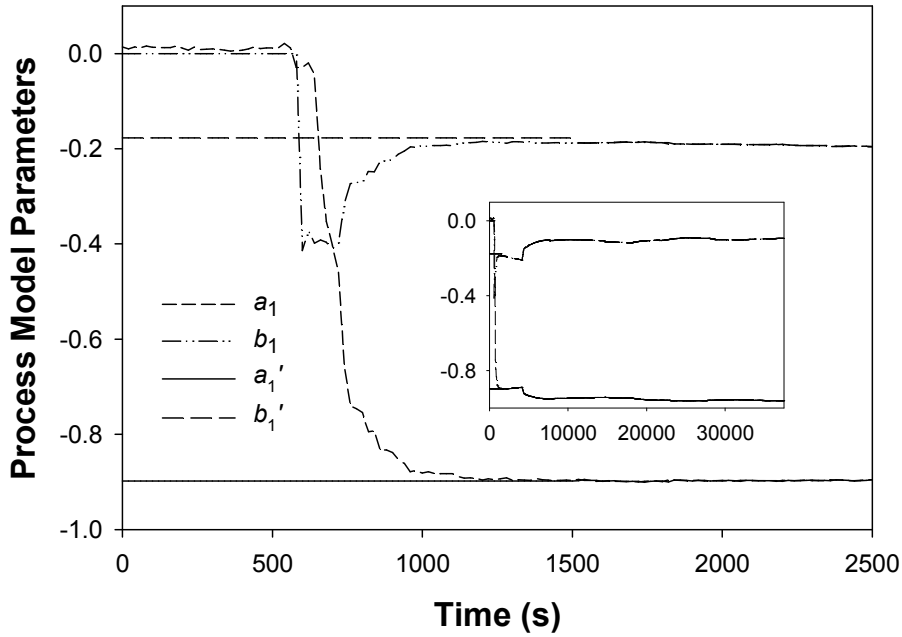

Figure 6.9: Transients of the process model parameters identified by the VFF-RLS algorithm (*i.e.* a_1 and b_1) as well as those eventually adopted as the internal model of the AS-GPC scheme (*i.e.* a_1*'* and b_1*'*) for the FAME concentration (C_{ME}) loop. The first set point change occurred at time = 500 s, while AS-GPC was activated at time = 1500 s.

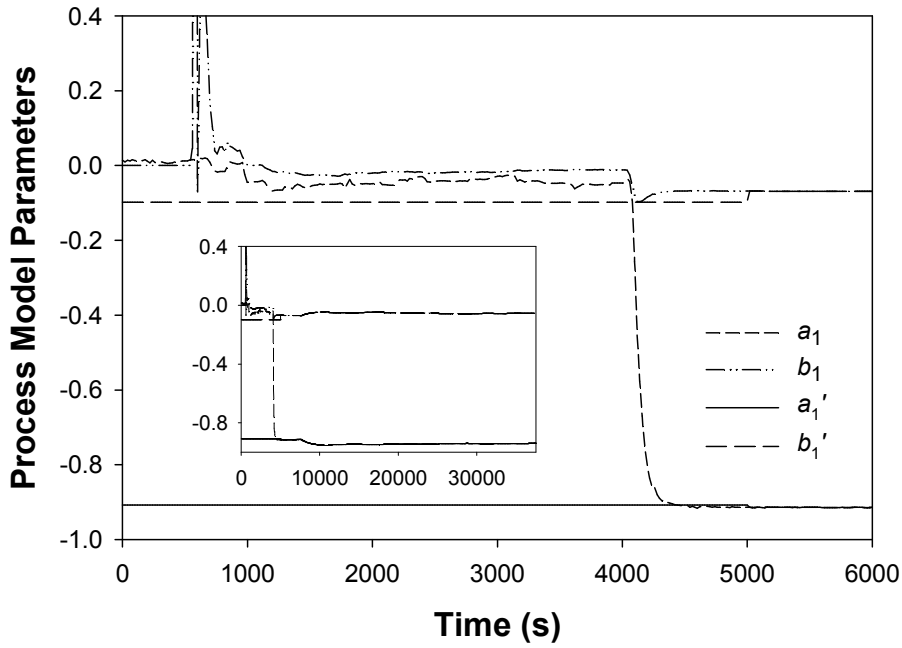

Figure 6.10: Transients of the process model parameters identified by the VFF-RLS algorithm (*i.e.* a_1 and b_1) as well as those eventually adopted as the internal model of the AS-GPC scheme (*i.e.* a_1*'* and b_1*'*) for the reactor temperature (T) loop. The first set point change occurred at time = 4000 s, while AS-GPC was activated at time = 5000 s.

Figure 6.11 shows the prediction error profiles for the VFF-RLS algorithms for both loops followed by the corresponding forgetting factor profiles in Figure 6.12. In general, due to the effect of loop interactions, changes in the C_{ME} loop affected the parameter estimation in the T loop, *vice versa*. Hence, the prediction error profiles for both loops fluctuate around zero, denoting the efforts of the VFF-RLS algorithms in tracking the possible changes in the process model parameters and constantly minimizing the prediction errors. In response to the fluctuating prediction errors, the forgetting factors for both loops changed accordingly as illustrated in Figure 6.12. When the prediction errors were large, possible reasons could be that the dynamics of the process had changed, and that past data were no longer relevant for the purpose of identifying the changed dynamics. Hence, the values of the forgetting factor would decrease in order to discard the unnecessary past data. The forgetting factors observed

for both loops were generally close to unity because for a slowly time-varying system, the prediction errors were generally within a small range of magnitudes (which meant that there was no sudden and abrupt change in the model parameters such that a huge amount of past data had to be forgotten). In addition to the above observations, the robustness of the AS-GPC was also demonstrated by the fact that the controller performance was adamant to slight plant model mis-match as inferred by the constant fluctuations in the prediction errors. It can be concluded thus far that parameter estimation by the VFF-RLS algorithms for both loops was successful.

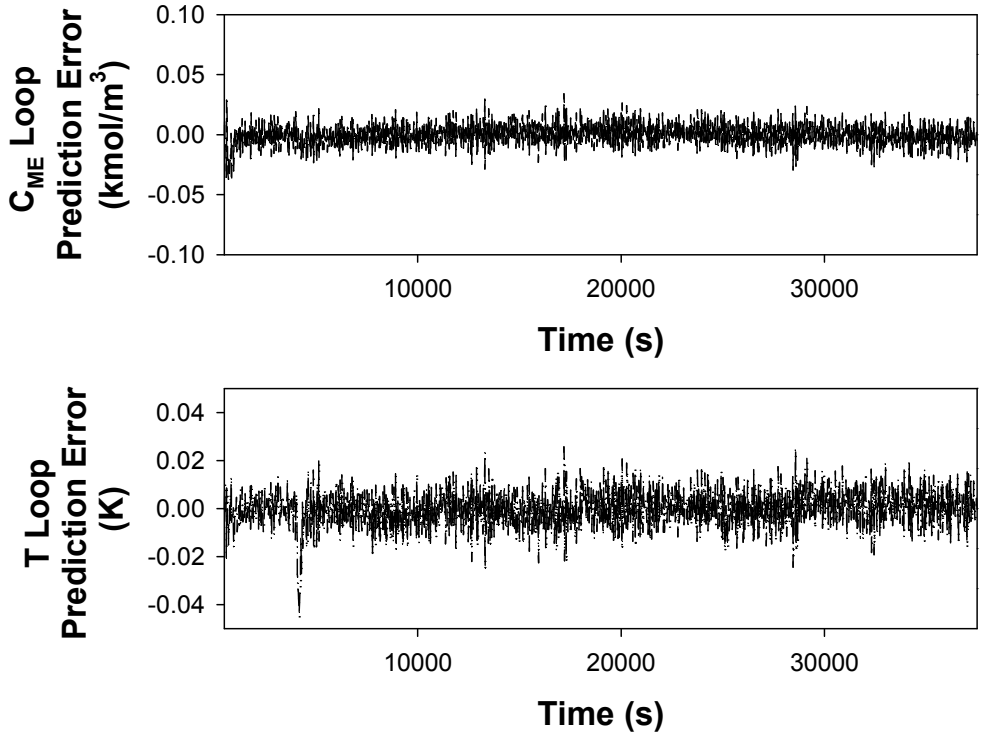

Figure 6.11: Prediction error profiles of the VFF-RLS algorithms for both the FAME concentration (C_{ME}) and reactor temperature (T) loops in the AS-GPC scheme.

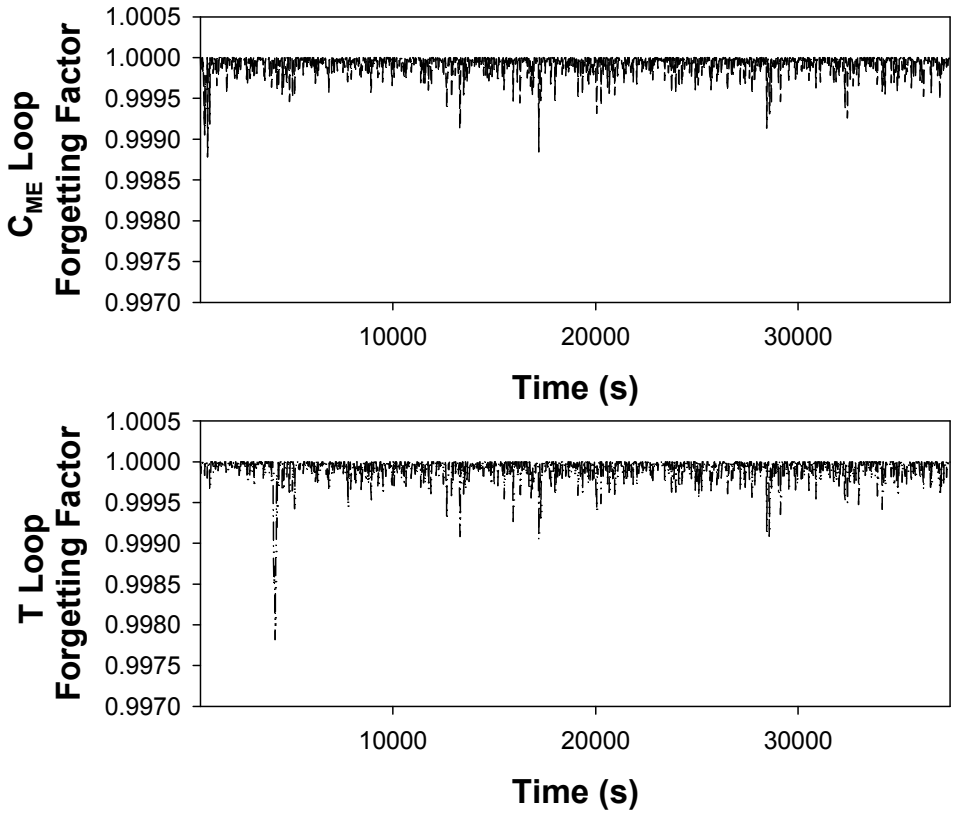

Figure 6.12: Forgetting factor profiles of the VFF-RLS algorithms for both the FAME concentration (C_{ME}) and reactor temperature (T) loops in the AS-GPC scheme.

Based on the reliable model estimates, the move suppression weights for both loops were calculated during the AS-GPC implementation and the temporal evolutions were shown in Figure 6.13. The fact that the move suppression weights were retuned in real time contributed to the improved performance of the AS-GPC over the other schemes, particularly for the C_{ME} loop in terms of the quality of the controller moves produced. In addition, although the tuning expressions for the move suppression weight as shown in Eqns. (2.49) - (2.50) were developed for unconstrained implementations, good results were obtained even when the expressions were used for constrained implementations.

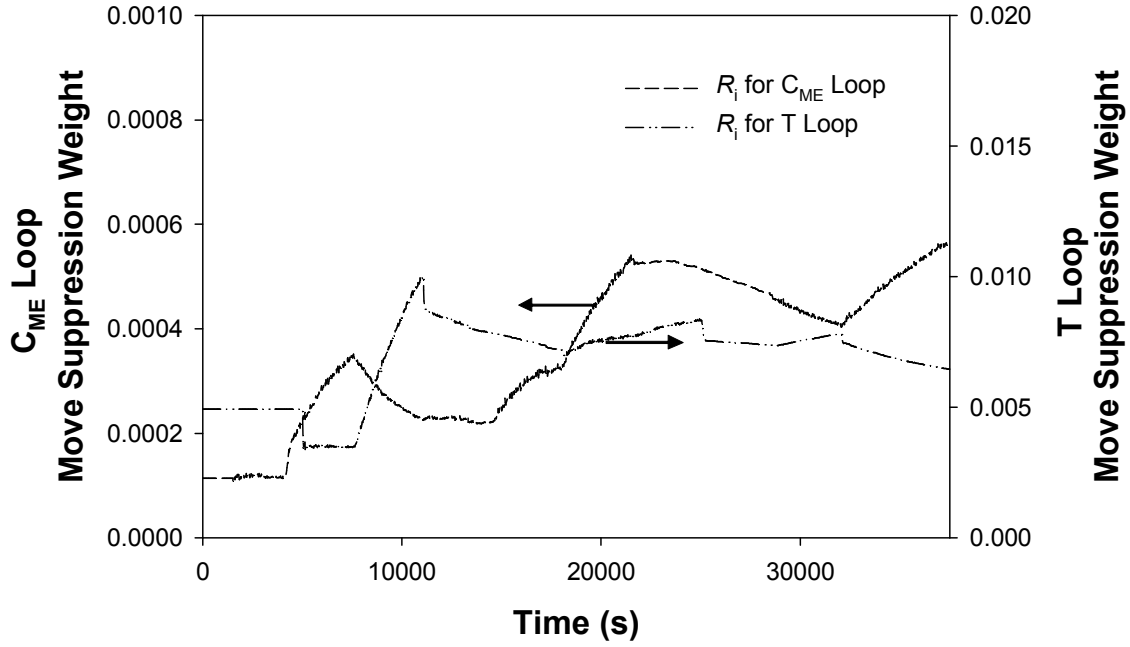

Figure 6.13: Temporal evolution of the move suppression weights (R_i) for both FAME concentration (C_{ME}) and reactor temperature (T) loops in the AS-GPC scheme.

The above results in general required some pre-calculations of the move suppression weights so that a reasonably tuned controller can be implemented right from the onset of simulation. Figures 6.14 - 6.15 demonstrate the efficacy of the AS-GPC scheme in salvaging situations where the move suppression weights are poorly tuned. In this case, assuming the absence of any *a priori* knowledge on the suitable values of the move suppression weights, the move suppression weights for the C_{ME} and T loops were deliberately chosen to consist of arbitrary positive integers (*i.e.* R_i = 10 and 45 for the C_{ME} and T loops respectively) which will impair the performance of the controller. In the figures, the AS-GPC scheme was activated at time = 17500 s. Prior to that, with the abovementioned fixed values of R_i for both loops, the control scheme is essentially the A-GPC scheme. From the figures, it is clear that a poor choice of the move suppression weight had devastating effect on the response of the controller. At time < 17500 s, the effect of improper tuning of the move suppression weight was clearer for the case of the C_{ME} loop where virtually no tracking of the setpoint was

observed. As for the T loop, the T profile observed was sluggish. Upon activation of the AS-GPC scheme at time = 17500 s, the controller performance (*i.e*. the response and the controller moves) for both the C_{ME} and T loops recovered speedily within 600 s. The AS-GPC scheme was able to automatically determine the suitable values of the move suppression weights for both loops, which contributed to the improved performance. The illustrations used here by no means implied that the engineer would arrive at such a bad initial estimate of the move suppression weights; but rather to emphasize that if it were to happen, the AS-GPC is still able to salvage the situation. The implications from these results are twofold: 1) model adaptation alone without a properly tuned move suppression weight is not sufficient to ensure good performance of the controller, 2) the AS-GPC scheme is well able to auto-tune the move suppression weight without *a priori* knowledge on its value.

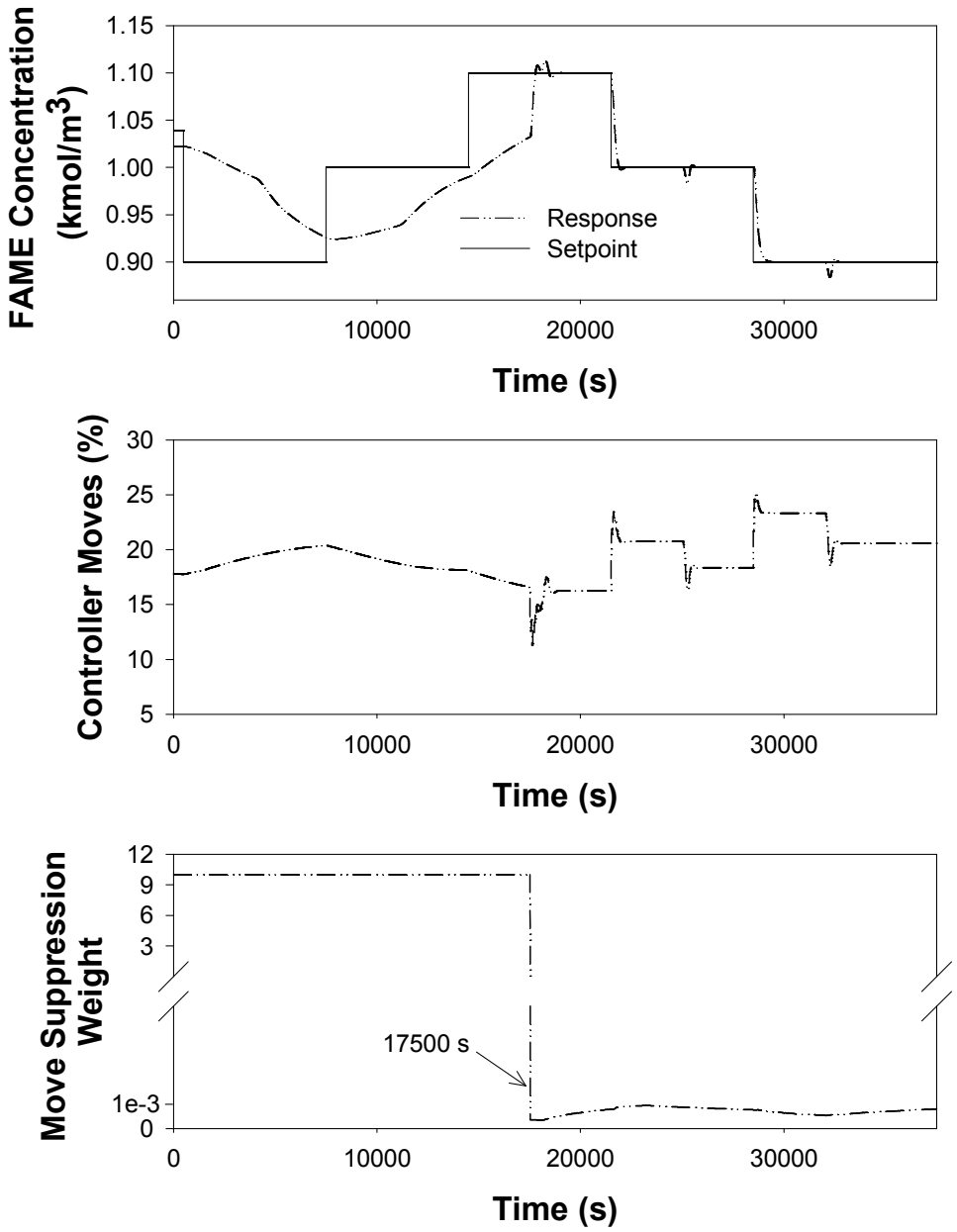

Figure 6.14: The corrective action produced by the AS-GPC scheme in salvaging the poor controller response caused by improper tuning of the move suppression coefficient for the FAME concentration (C_{ME}) loop. The self-tuning mechanism was activated at time = 17500 s. The move suppression weight at time < 17500 s is 10, upon tuning its magnitude was around 1×10^{-3}.

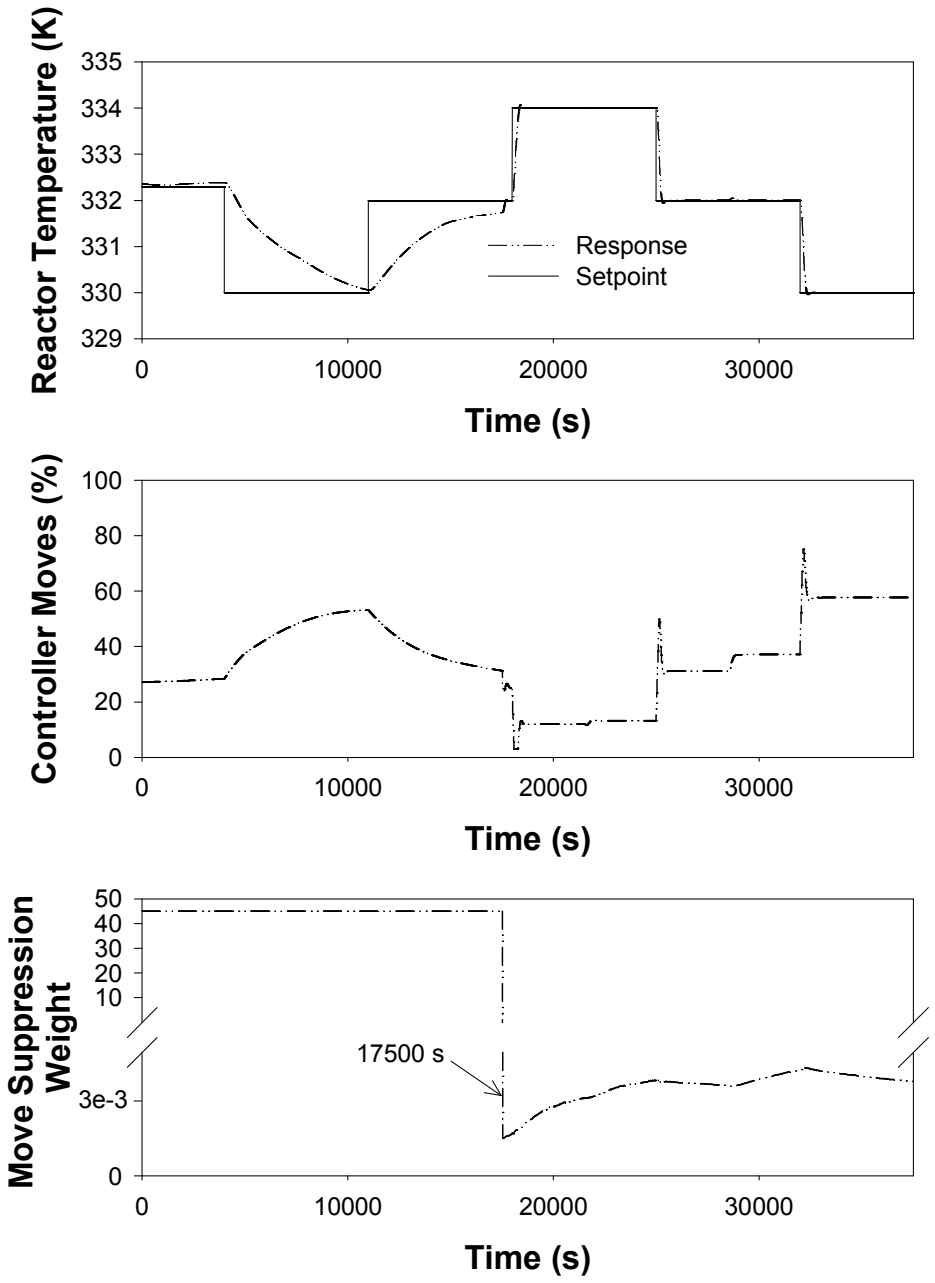

Figure 6.15: The corrective action produced by the AS-GPC scheme in salvaging the poor controller response caused by improper tuning of the move suppression coefficient for the reactor temperature (T) loop. The self-tuning mechanism was activated at time = 17500 s. The move suppression weight at time < 17500 s is 45, upon tuning its magnitude was around 3×10^{-3}.

6.5 Benchmarking with the Performance of Conventional PID Controllers

To reveal the superiority of the AS-GPC scheme against conventional PID schemes, the same setpoint tests as mentioned above were conducted on the biodiesel reactor model using conventional PID controllers. As mentioned in Section 3.7, the PID controllers were tuned using the IMC approach as it is one of the good PID tuning methods available (Chien & Fruehauf, 1990; Garcia & Morari, 1982; Rivera, Morari, & Skogestad, 1986). Besides that, the ZN tuning method was also used to tune the PID controllers as it is a widely used method in the industry. Calculations of the PID tuning parameters were done based on the nominal continuous time model parameters for each loop (*i.e*. Model 2 of Table 5.1 for the C_{ME} loop and Model 3 of Table 5.2 for the T loop) by utilizing the equations presented in Table 3.4. As the C_{ME} loop usually involved noisy process data in reality, the derivative action for the C_{ME} loop was turned off throughout this work. This decision is also justified by the poor performance obtained from the deployment of a PID controller tuned using the various methods described above on the C_{ME} loop during preliminary simulations. For the T loop, the opposite is true, where preliminary simulations showed that the PID controller performed better than the PI controller. Figures 6.16 - 6.19 show the results of the closed loop simulations for both the IMC-based and ZN-based PID controllers (or PI controllers for the C_{ME} loop) on the biodiesel reactor model, while Table 6.7 show the values of the tuning parameters employed for each scheme. To ease reading below, instead of writing "PI/PID controllers" to cater for the slight difference in both controllers, the general term "PID controllers" will be used.

Table 6.7: Values of Internal Model Control (IMC) and Ziegler-Nichols (ZN) based PID tuning parameters (K_c = proportional gain, τ_I = integral time constant, τ_D = derivative time constant, τ_c = IMC design parameter) for the FAME concentration (C_{ME}) and reactor temperature (T) loops.

Tuning method	Loop	K_c	τ_I	τ_D	τ_c
IMC	C_{ME}	-132.02	185.09	0	40.556
	T	-36.35	217.68	9.41	19.703
ZN	C_{ME}	-237.64	135.05	0	-
	T	-1183.41	39.41	9.85	-

Of the two methods used to tune the PID controllers, generally the IMC-based PID controllers exhibited better performance than the ZN-based PID controllers. The ZN-based PID controllers yielded high overshoots in the C_{ME} and T profiles. In addition, the controller moves produced were aggressive and chaotic especially for the T loop. Such controller moves were generally not realizable during practical implementations. The performance of the IMC-based PID controllers were better as compared to the ZN-based PID controllers in that the overshoots observed were lower and the controller moves produced were of fairly good quality for the C_{ME} loop and less chaotic for the T loop (although still not realizable for practical implementations). Despite the improved performance of the IMC-based PID controllers as compared to the ZN-based PID controllers, the performance of the IMC-based PID controllers (as revealed in Figures 6.16 - 6.19) paled in comparison to the performance of the AS-GPC (as demonstrated in Figures 6.5 - 6.8) in the ability to attain minimal overshoot and good controller moves. Table 6.8 shows the comparison of the maximum and minimum overshoots exhibited by the AS-GPC and the IMC-based PID controllers for both loops, which clearly revealed the superiority of the AS-GPC in achieving virtually no overshoots (as

illustrated by Figures 6.5 and 6.7). A common observation for the IMC-based PID controller is that a tradeoff exists between good closed loop response and realizable controller moves. Although the responses attained by the IMC-based PID controllers were fast, these were achieved at the expense of aggressive controller moves. Furthermore, these controllers with fixed controller settings lacked the ability to handle nonlinearities arising from changes in the operating range of the reactor. The AS-GPC scheme, however, was able to tackle all these issues and produced good setpoint tracking together with superior quality of the controller moves.

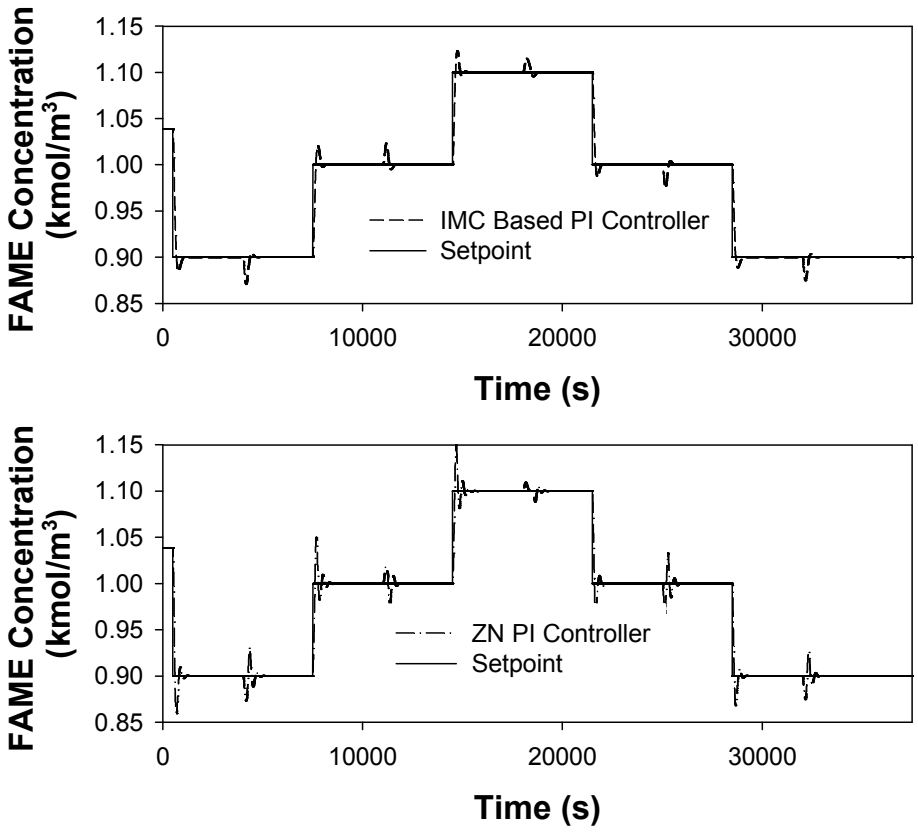

Figure 6.16: Performance of the Internal Model Control (IMC) PI controller and the Ziegler-Nichols (ZN) PI controller in tracking a series of changes in setpoint for the FAME concentration (C_{ME}) loop.

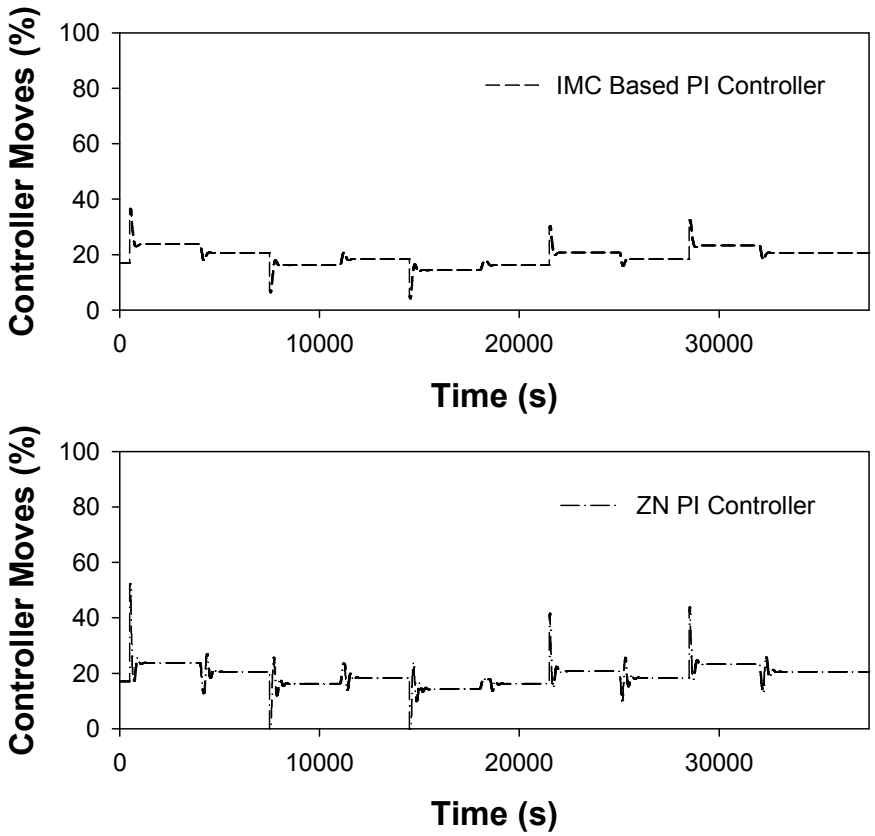

Figure 6.17: The corresponding controller moves produced by the Internal Model Control (IMC) PI controller and the Ziegler-Nichols (ZN) PI controller for tracking a series of changes in setpoint for the FAME concentration (C_{ME}) loop.

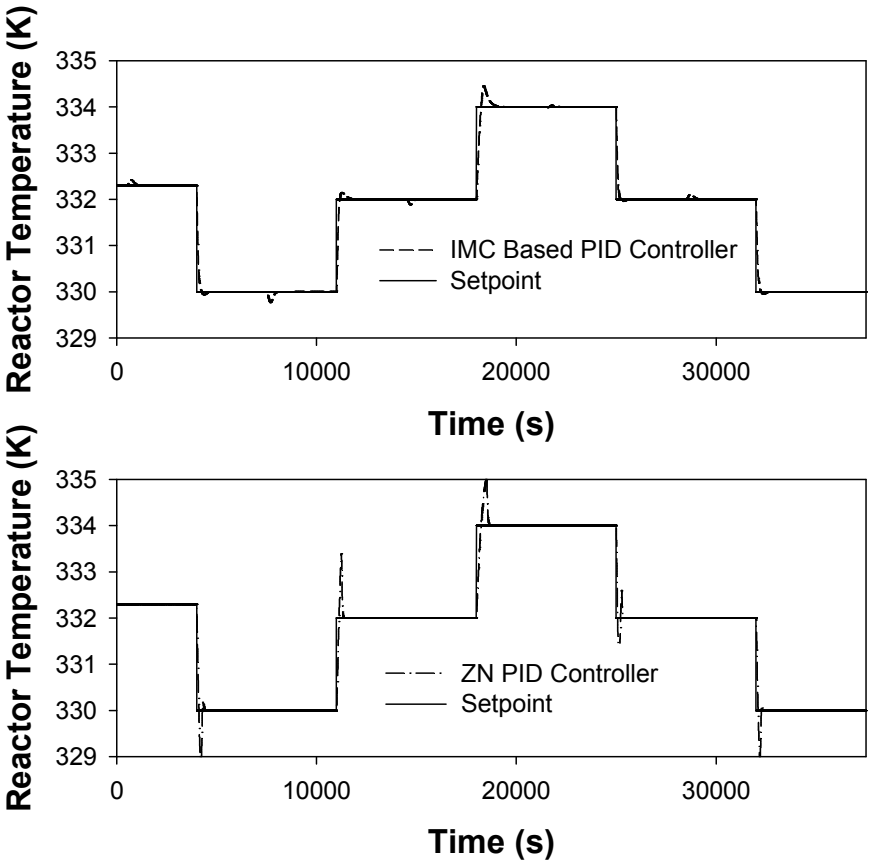

Figure 6.18: Performance of the Internal Model Control (IMC) PID controller and the Ziegler-Nichols (ZN) PID controller in tracking a series of changes in setpoint for the reactor temperature (T) loop.

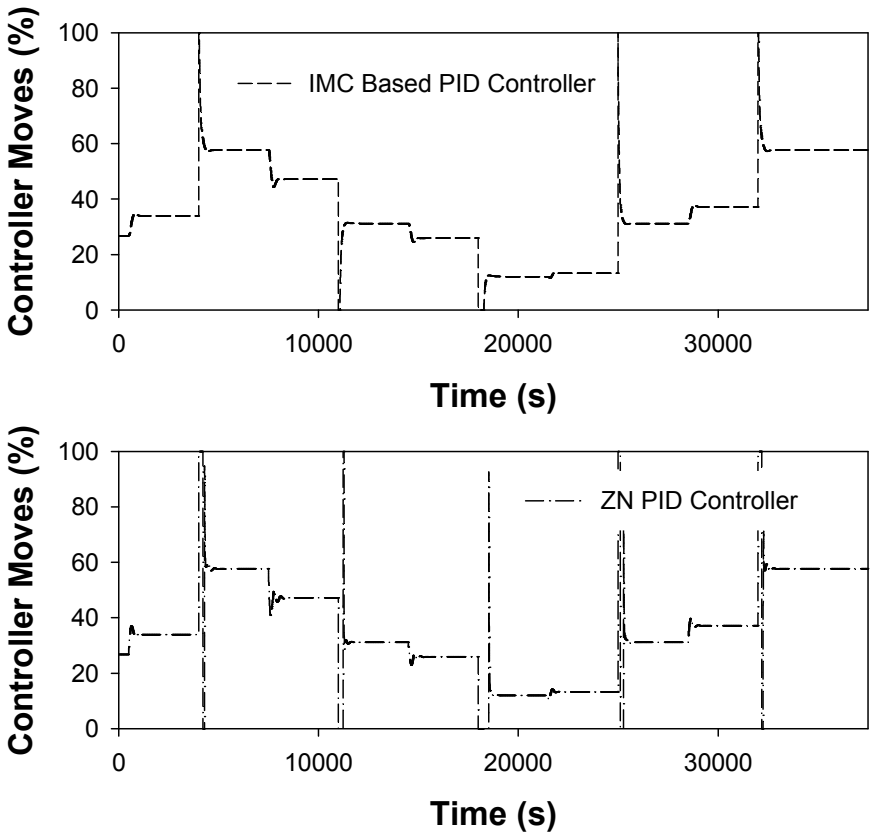

Figure 6.19: The corresponding controller moves produced by the Internal Model Control (IMC) PID controller and the Ziegler-Nichols (ZN) PID controller for tracking a series of changes in setpoint for the reactor temperature (T) loop.

Table 6.8: Comparison of the maximum and minimum overshoots exhibited by the AS-GPC and the IMC-based PID controllers for the FAME concentration (C_{ME}) and reactor temperature (T) loops. The sizes of the overshoots are expressed in terms of percentage of the corresponding setpoint change.

Loop	Parameter	AS-GPC	IMC-based PID
C_{ME}	Maximum overshoot (%)	Not observed	23.5
	Minimum overshoot (%)	Not observed	11.4
T	Maximum overshoot (%)	Not observed	22.9
	Minimum overshoot (%)	Not observed	2.25

6.6 Regulatory Performance of the Constrained AS-GPC

For a particular control scheme to be considered efficient in handling process control problems, the controller not only has to perform well in tracking setpoint changes, but it should also be able to reject disturbances. Hence, the AS-GPC scheme was further scrutinized for the efficacy in regulatory control. Four variables (*i.e.* T_O, C_{TGO}, T_{CO} and N) were identified as possible disturbance variables in the mechanistic transesterification model (Mjalli, Lee, Kiew, & Hussain, 2009) as alluded to in Section 1.1. A 5 % step increment in the nominal value of a quantity chosen from C_{TGO}, T_O, T_{CO}, and N was introduced at time = 37500 s, following the setpoint tests. Figures 6.20 - 6.21 show the performance of the AS-GPC scheme in rejecting the various load disturbances. The AS-GPC scheme successfully rejected all the disturbances within reasonable time, bringing both C_{ME} and T back to their set points within 1000 seconds at the most. Changes in C_{TGO} had the largest effect on the C_{ME} profile whereas disturbances in T_{CO} affected the pace of recovery of T most significantly. Further, it was observed that N had negligible effects on the profiles of both C_{ME} and T. It should be noted that a 5 % step increment in the disturbance variables did not require much control efforts to tame the process for the C_{ME} loop as revealed by the narrow range of controller movements. This, however, was not the case for the T loop, as a 5 % step increment in T_O required substantial effort in the controller movement (≈ + 40 %) to recover from the process interruption. Despite that, generally the controller moves for both loops were observed to be non-aggressive and actuator saturation did not occur.

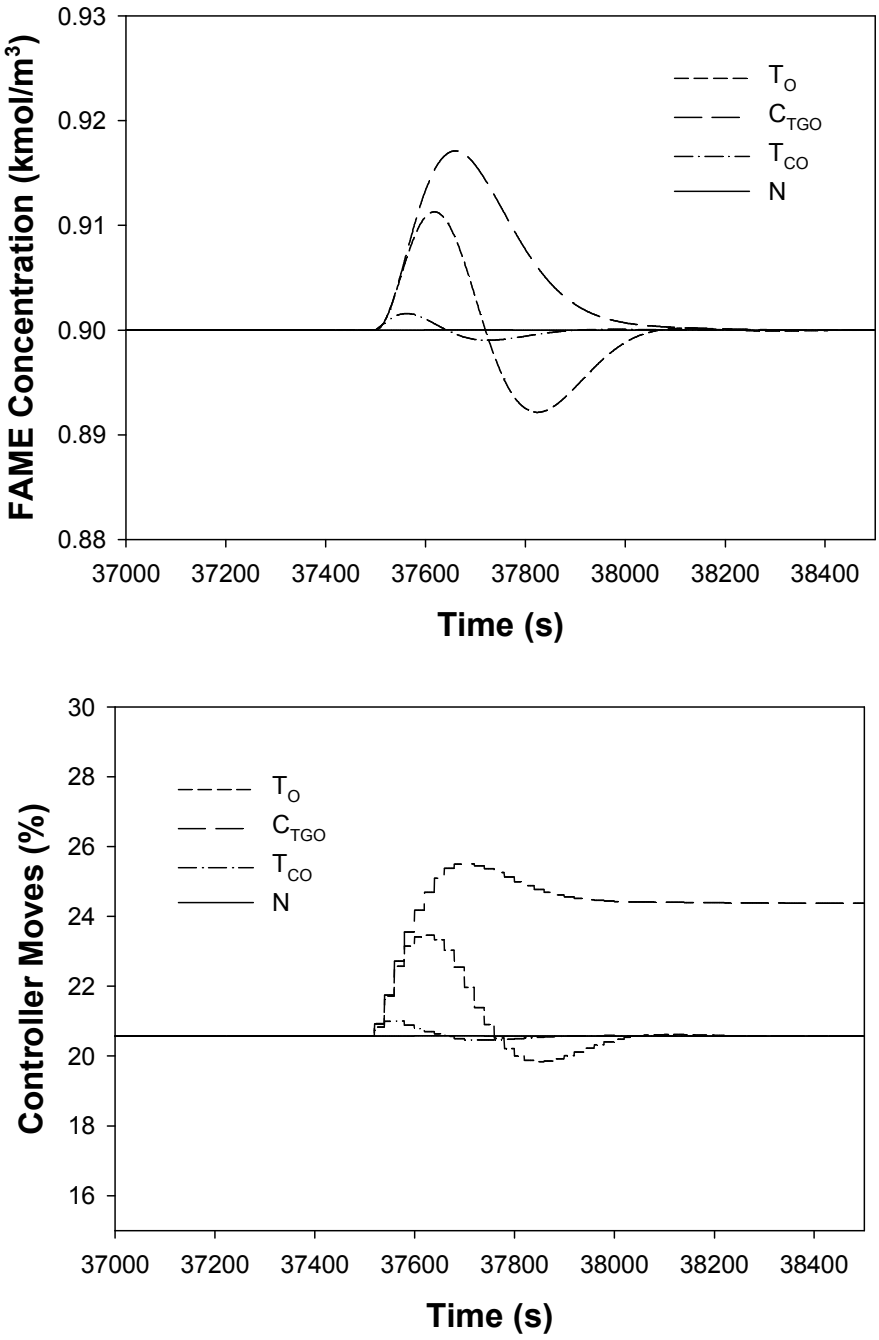

Figure 6.20: Effects of various individual disturbance variables, *viz.* the feed temperature (T_O), concentration of triglycerides (C_{TGO}), coolant inlet temperature (T_{CO}), and stirrer rotational speed (N), on the performance of the AS-GPC control scheme in controlling the FAME concentration (C_{ME}) and the corresponding controller moves for the reactant flow rate (F_o). Five percent step increment in the nominal values of T_O, C_{TGO}, T_{CO}, and N were introduced at time = 37500 s. These disturbances were introduced one at a time, hence shown here are superpositions of four separate runs.

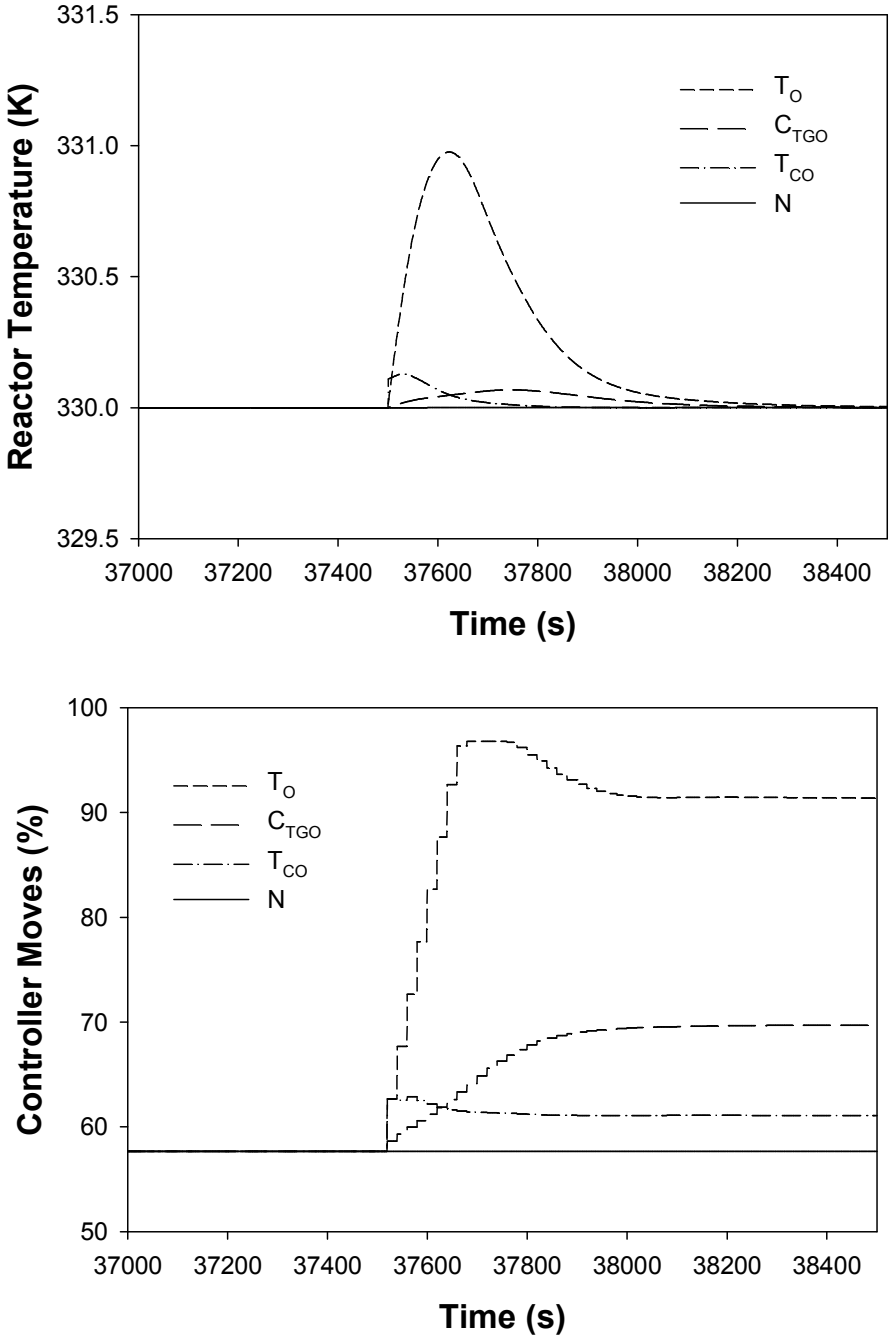

Figure 6.21: Effects of various individual disturbance variables, *viz.* the feed temperature (T_O), concentration of triglycerides (C_{TGO}), coolant inlet temperature (T_{CO}), and stirrer rotational speed (N), on the performance of the AS-GPC control scheme in controlling the reactor temperature (T) and the corresponding controller moves for the coolant flow rate (F_c). Five percent step increment in the nominal values of T_O, C_{TGO}, T_{CO}, and N were introduced at time = 37500 s. These disturbances were introduced one at a time, hence shown here are superpositions of four separate runs.

6.7 Concluding Remarks

In this chapter, it was shown that by incorporating the model adaptation and self-tuning mechanisms simultaneously in the AS-GPC, the controller moves produced were improved while achieving good closed loop responses as compared to A-GPC and GPC schemes. Moreover, for a nonlinear process such as the transesterification process, even the high performance IMC-based PID scheme, could not rival the performance of the AS-GPC. In this study, the AS-GPC scheme proved to be efficient in handling both servo and regulatory control problems. With the implementation of this scheme, the modeling and tuning issues were dealt with automatically by the controller, thus reducing human intervention and effort in troubleshooting control issues caused by incorrect modeling and tuning of the controller. Consequently, this resulted in improved control performance when dealing with process nonlinearities and uncertainties.

CHAPTER 7

CONCLUSIONS, THESIS AND RECOMMENDATIONS

7.1 Conclusions and Thesis

The main aim of this work is to design and develop a GPC controller with both model adaptation and self-tuning capabilities. To accomplish this, the output of the RLS algorithm is used not only for model adaptation in the GPC controller, but also for self-tuning through the use of the FOPDT-based explicit analytical expressions proposed by Shridhar and Cooper (1997b). This novel combination (*i.e.* the AS-GPC scheme) was not reported elsewhere in the academic literature. To systematically achieve this ultimate aim, several research objectives were formed and met, as summarized below:

i) Three variants of the RLS algorithms were screened, *i.e.* the EWRLS algorithm, VFF-RLS algorithm, and the EDF algorithm. The EWRLS and VFF-RLS algorithms had more flexibility in dealing with variability in the process dynamics. For this work, the VFF-RLS algorithm was chosen due to its successful implementation track records.

ii) Open loop dynamic system analysis was performed on the validated mechanistic biodiesel reactor model developed by Mjalli, Lee, Kiew, and Hussain (2009). The results showed that the dynamics of the process were operating point dependent as well as displaying loop interactions, hence justifying the need for an advanced control scheme.

iii) Offline FOPDT system identification was successfully performed on the biodiesel reactor at different operating regions. These results showed that the process model parameters identified at different operating regions were different, thus confirming the existence of nonlinearities in the process dynamics. The results obtained were also used to design a backup GPC controller for the AS-GPC scheme to prevent severe process upsets prior to the activation or during the failure of adaptive mechanisms.

iv) The AS-GPC scheme was successfully designed and developed. Closed loop implementation of the unconstrained AS-GPC on the biodiesel reactor showed good performance, albeit having rather large input slew rates due to the absence of active constraints on controller moves. The closed loop poles exhibited by the unconstrained AS-GPC implied stable dynamics.

v) The shortcoming of the unconstrained AS-GPC was rectified by imposing constraints on the controller. Better controller moves were achieved compared to those of the unconstrained cases. Simulation results also showed that while the A-GPC scheme with model adaptation alone performed better than the conventional GPC scheme, the AS-GPC scheme with both model adaptation and self-tuning of the move suppression weight outperformed the corresponding A-GPC scheme. The AS-GPC scheme also showed better performance than the conventional PID scheme tuned using among the best PID tuning methods, *i.e*. the IMC method. In addition, good regulatory control was also achieved by the AS-GPC scheme.

The evidence throughout this work supports the thesis that better controller performance can be obtained by having both model adaptation and self-tuning in the

GPC controller structure. One possibility of achieving this is to use the RLS algorithm both to model the process continuously in the form of a FOPDT model and to retune the controller using the tuning correlations proposed by Shridhar and Cooper (1997b).

7.2 Recommendations for Future Work

This work had shown that the traditional use of the RLS algorithm for model adaptation in the GPC controller can be extended further to cater also for the controller self-tuning. On the basis of this idea, several future extensions of the work are possible:

i) Instead of only tuning a single active parameter as in this work, self-tuning of all the GPC tuning parameters based on the output of the RLS algorithm can be made possible based on proposed tuning correlations available in the literature. This would further relieve control engineers of the efforts involved in tuning the controller. Furthermore, contrary to existing self-tuning methods, no real time optimization is needed to compute the optimal tuning parameters.

ii) A MIMO AS-GPC can be designed and developed based on the analytical FOPDT based tuning expressions proposed by Shridhar and Cooper (1997a) for MIMO open loop stable systems. It is expected that a MIMO AS-GPC scheme will demonstrate improved capability in handling loop interactions.

iii) New model-based analytical tuning expressions for the GPC can be developed based on second order process models to cater to processes with higher order dynamics which the FOPDT approximation fails to capture.

The same approach as adopted in this work can be used, where the RLS algorithm estimates the model parameters of the second order model, and the tuning parameters are calculated based on the estimated parameters.

iv) To more rigorously validate its performance, the AS-GPC scheme can be deployed on real world nonlinear processes (*e.g*. a lab scale biodiesel reactor), where noises, measurement errors plus hysteresis and nonlinearities of control valves *etc*. are prevalent. Comparison with auto-tuned PID controllers can also be done.

v) A proper stability analysis of the proposed constrained control scheme can be done. While the observations are that the closed loop implementation of the AS-GPC was stable on the biodiesel reactor model, in general this is not easy to prove; hence the issue of closed loop stability deserves deeper examination.

vi) The convergence rates for the RLS algorithm, the speed of change of the model parameters and the interaction between these and the control law updates (which are important in affecting the performance of the controller) can be studied at a deeper level.

LIST OF CONFERENCES ATTENDED AND PUBLICATIONS

Conferences Attended:

a) Presented a paper titled "Adaptive Control of a Transesterification Reactor" in the *International Conference of Recent and Emerging Advanced Technologies in Engineering 2009* (iCREATE 2009) at KLIA Pan Pacific Hotel on the 23th – 24th November 2009.

b) Presented a paper titled "Multivariable Adaptive Predictive Model Based Control of a Biodiesel Transesterification Reactor" in the *2010 International Conference on Process Engineering and Advanced Material* (ICPEAM 2010) at KLCC Convention Centre on the 15th – 17th June 2010.

Publications:

a) Ho, Y. K., Mjalli, F. S., & Yeoh, H. K. (2010a). Multivariable adaptive predictive model based control of a biodiesel transesterification reactor. *Journal of Applied Sciences, 10*(12), 1019-1027.

b) Ho, Y. K., Mjalli, F. S., & Yeoh, H. K. (2010b). Recursive least squares-based adaptive control of a biodiesel transesterification reactor. *Industrial & Engineering Chemistry Research, 49*(22), 11434-11442.

REFERENCES

Agee, W. S., & Turner, R. H. (1972). Triangular decomposition of a positive definite matrix plus a symmetric dyad with applications to Kalman Filtering. *Technical Report 38*

Ahlberg, D. T., & Cheyne, I. (1976). *Adaptive control of a polymerization reactor.* Paper presented at the AIChE Symposium Series.

Al-Ghazzawi, A., Ali, E., Nouh, A., & Zafiriou, E. (2001). Online tuning strategy for model predictive controllers. *Journal of Process Control, 11*, 265 - 284.

Ali, E., & Al-Ghazzawi, A. (2003). Online tuning of model predictive controllers using fuzzy logic. *Canadian Journal of Chemical Engineering, 81*, 1041 - 1054.

Anderson, B. D. O., & Dehghani, A. (2008). Challenges of adaptive control-past, permanent and future. *Annual Reviews in Control, 32*(2), 123-135.

Åström, K. J. (1983). Theory and applications of adaptive control - A survey. *Automatica, 19*(5), 471-486.

Åström, K. J., Borisson, U., Ljung, L., & Wittenmark, B. (1977). Theory and applications of self-tuning regulators. *Automatica, 13*(5), 457-476.

Åström, K. J., Hagander, P., & Sternby, J. (1984). Zeros of sampled systems. *Automatica, 20*(1), 31 - 38.

Åström, K. J., & Wittenmark, B. (1994). *Adaptive control* (2nd ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Åström, K. J., & Wittenmark, B. (1997). *Computer controlled systems: Theory and design*. New Jersey: Prentice Hall.

Banerjee, A., Arkun, Y., Ogunnaike, B., & Pearson, R. (1997). Estimation of nonlinear systems using linear multiple models. *AIChE Journal, 43*(5), 1204-1226.

Banerjee, P., & Shah, S. L. (1992, December 16 - 18). *Tuning guidelines for robust generalized predictive control.* Paper presented at the Proceedings of the 31st IEEE Conference on Decision and Control, Tucson, AZ.

Bengtsson, G., & Egardt, B. (1985). *Experiences with self-tuning control in the process industry.* Paper presented at the IFAC Proceedings Series.

Bequette, B. W. (1991). Nonlinear control of chemical processes: a review. *Industrial & Engineering Chemistry Research, 30*(7), 1391-1413. doi: 10.1021/ie00055a001

Bierman, G. J. (1976). Measurement updating using the U-D factorization. *Automatica, 12*(4), 375-382.

Buchholt, F., & Kümmel, M. (1979). Self-tuning control of a pH-neutralization process. *Automatica, 15*(6), 665-671.

Caldwell, W. I. (1950). United States of America Patent No.: A. Patent.

Camacho, E. F., & Bordons, C. (1999). *Model Predictive Control*. Berlin: Springer-Verlag.

Chien, I. L., & Fruehauf, P. S. (1990). Consider IMC tuning to improve controller performance. *Chemical Engineering Progress, 86*(10), 33-41.

Chow, C.-M., Kuznetsov, A. G., & Clarke, D. W. (1998). Successive one-step-ahead predictions in multiple model predictive control. *International Journal of Systems Science, 29*(9), 971 - 979.

Clarke, D. W. (1981). *Self-tuning and Adaptive Control: Theory and Applications.* London: Peregrinus.

Clarke, D. W. (1988). Application of generalized predictive control to industrial processes. *IEEE Control Systems Magazine, 8*(2), 49-55.

Clarke, D. W., & Gawthrop, P. J. (1981). Implementation and application of microprocessor-based self-tuners. *Automatica, 17*(1), 233-244.

Clarke, D. W., & Mohtadi, C. (1989). Properties of generalized predictive control. *Automatica, 25*(6), 859-875.

Clarke, D. W., Mohtadi, C., & Tuffs, P. S. (1987a). Generalized predictive control - Part I. The basic algorithm. *Automatica, 23*(2), 137-148.

Clarke, D. W., Mohtadi, C., & Tuffs, P. S. (1987b). Generalized predictive control - Part II Extensions and interpretations. *Automatica, 23*(2), 149-160.

Cordero, A. O., & Mayne, D. Q. (1981). Deterministic convergence of a self-tuning regulator with variable forgetting factor. *IEEE Proceedings D: Control Theory and Applications, 128*(1), 19-23.

Corrêa, N. A., Corrêa, R. G., & Freire, J. T. (2002). Self-tuning control of egg drying in spouted bed using the GPC algorithm. *Drying Technology: An International Journal, 20*(4), 813 - 828.

Di Marco, R., Semino, D., & Brambilla, A. (1997). From linear to nonlinear model predictive control: Comparison of different algorithms. *Industrial & Engineering Chemistry Research, 36*(5), 1708-1716.

Dougherty, D., & Cooper, D. (2003a). A practical multiple model adaptive strategy for multivariable model predictive control. *Control Engineering Practice, 11*(6), 649-664.

Dougherty, D., & Cooper, D. (2003b). A practical multiple model adaptive strategy for single-loop MPC. *Control Engineering Practice, 11*(2), 141-159.

Eevera, T., Rajendran, K., & Saradha, S. (2009). Biodiesel production process optimization and characterization to assess the suitability of the product for varied environmental conditions. *Renewable Energy, 34*(3), 762-765.

Elicabe, G. G., & Meira, G. R. (1988). Estimation and control in polymerization reactors. A review. *Polymer Engineering and Science, 28*(3), 121-135.

Ertunc, S., Akay, B., Boyacioglu, H., & Hapoglu, H. (2009). Self-tuning control of dissolved oxygen concentration in a batch bioreactor. *Food and Bioproducts Processing, 87*(1), 46-55.

Fortescue, T. R., Kershenbaum, L. S., & Ydstie, B. E. (1981). Implementation of self-tuning regulators with variable forgetting factors. *Automatica, 17*(6), 831-835.

Garcia, C. E., & Morari, M. (1982). Internal model control. 1. A unifying review and some new results. *Industrial and Engineering Chemistry Process Design and Development, 21*(2), 308-323.

Garcia, C. E., Prett, D. M., & Morari, M. (1989). Model predictive control: Theory and practice-A survey. *Automatica, 25*(3), 335-348.

Garriga, J. L., & Soroush, M. (2010). Model predictive control tuning methods: A review. *Industrial & Engineering Chemistry Research, 49*(8), 3505-3515. doi: 10.1021/ie900323c

Gendron, S., Perrier, M., Barretter, J., Amjad, M., Holko, A., & Legault, N. (1993). Deterministic adaptive control of SISO processes using model weighting adaptation. *International Journal of Control, 58*(5), 1105 - 1123.

Hägglund, T., & Åström, K. J. (2000). Supervision of adaptive control algorithms. *Automatica, 36*(8), 1171-1180.

Hallager, L., Goldschmidt, L., & Jorgensen, S. B. (1984). Multivariable adaptive identification and control of a distributed chemical reactor. *Large Scale Systems, 6*(3), 323-336.

Han, K., Zhao, J., & Qian, J. (2006, June 21 - 23). *A novel robust tuning strategy for model predictive control.* Paper presented at the 6th World Congress on Intelligent Control and Automation, Dalian, China.

Harris, T. J., MacGregor, J. F., & Wright, J. D. (1978). *An application of self-tuning regulators to catalytic reactor control.* Paper presented at the Proceedings JACC, Philadelphia.

Ho, Y. K., Mjalli, F. S., & Yeoh, H. K. (2010a). Multivariable adaptive predictive model based control of a biodiesel transesterification reactor. *Journal of Applied Sciences, 10*(12), 1019-1027.

Ho, Y. K., Mjalli, F. S., & Yeoh, H. K. (2010b). Recursive least squares-based adaptive control of a biodiesel transesterification reactor. *Industrial & Engineering Chemistry Research, 49*(22), 11434-11442.

Hodgson, A. J. F., & Clarke, D. W. (1982). *Self-tuning control applied to batch reactors.* Paper presented at the IEEE Conference on Application of Adaptive and Multivariable Control, Hull, England.

Ioannou, P. A., & Fidan, B. (2006). *Adaptive control tutorial*. Philadelphia: Society for Industrial and Applied Mathematics.

Isermann, R. (1982). Parameter adaptive control algorithms--A tutorial. *Automatica, 18*(5), 513-528.

Karacan, S., Hapoglu, H., & Alpbaz, M. (2001). Application of optimal adaptive generalized predictive control to a packed distillation column. *Chemical Engineering Journal, 84*, 389 - 396.

Kawai, F., Ito, H., Nakazawa, C., Matsui, T., Fukuyama, Y., Suzuki, R. (2007, October 1 - 3). *Automatic tuning for model predictive control: Can particle swarm optimization find a better parameter?* Paper presented at the 22nd IEEE International Symposium on Intelligent Control, Singapore.

Khodabandeh, M., & Bolandi, H. (2007). *Model predictive control with state estimation and adaptation mechanism for a continuous stirred tank reactor*.

Paper presented at the International Conference on Control, Automation and Systems 2007, COEX, Seoul, Korea.

Kulhavy, R., & Karny, M. (1985). *Tracking of slowly varying parameters by directional forgetting.* Paper presented at the IFAC Proceedings Series, Budapest.

Kwalik, K. M., & Schork, F. J. (1985). *Adaptive control of a continuous polymerization reactor.* Paper presented at the Proceedings of the American Control Conference, Boston.

Leith, D. J., & Leithead, W. E. (2000). Survey of gain-scheduling analysis and design. *International Journal of Control, 73*(11), 1001 - 1025.

Leung, D. Y. C., & Guo, Y. (2006). Transesterification of neat and used frying oil: Optimization for biodiesel production. *Fuel Processing Technology, 87*(10), 883-890.

Leung, D. Y. C., Wu, X., & Leung, M. K. H. (2010). A review on biodiesel production using catalyzed transesterification. *Applied Energy, 87*(4), 1083-1095.

Liu, W., & Wang, G. (2000, October 8 - 11). *Auto-tuning procedure for model predictive controller.* Paper presented at the Proceedings of the 2000 IEEE International Conference on Systems, Man, and Cybernetics, Nashville, TN.

Ljung, L. (1987). *System Identification: Theory for the User*. New Jersey: Prentice Hall.

Ljung, L. (2008). *System identification toolbox*: The Mathworks, Inc.

Ljung, L., & Gunnarsson, S. (1990). Adaptation and tracking in system identification - A survey. *Automatica, 26*(1), 7-21.

Ljung, L., & Söderstöm, T. (1983). *Theory and Practice of Recursive Identification*. Massachusettes: MIT Press.

Maciejowski, J. M. (2002). *Predictive Control with Constraints*. Englewood Cliffs, New Jersey: Prentice Hall.

McDermott, P. E. (1984). *Self-tuning control of a fixed bed autothermal reactor.* Ph.D, University of California, Santa Barbara.

McIntosh, A. R., Shah, S. L., & Fisher, D. G. (1989, June 21 - 23). *Selection of tuning parameters for adaptive generalized predictive control.* Paper presented at the Proceedings of the 1989 American Control Conference, Pittsburgh, PA.

McIntosh, A. R., Shah, S. L., & Fisher, D. G. (1990, August 13 - 17). *Performance tuning of adaptive generalized predictive control.* Paper presented at the Proceedings of the 11th IFAC Congress, Tallinn, USSR.

McIntosh, A. R., Shah, S. L., & Fisher, D. G. (1991). Analysis and tuning of adaptive generalized predictive control. *Canadian Journal of Chemical Engineering, 69*, 97-110.

Mendoza-Bustos, S. A., Penlidis, A., & Cluett, W. R. (1990). Adaptive control of conversion in a simulated solution polymerization continuous stirred tank reactor. *Industrial & Engineering Chemistry Research, 29*(1), 82-89.

Middleton, R. H. (1991). Tradeoffs in linear control system design. *Automatica, 27*(2), 281 - 292.

Mikleš, J., & Fikar, M. (2007). *Process Modelling, Identification and Control*. Berlin: Springer-verlag.

Mjalli, F. S., & Hussain, M. A. (2009). Approximate Predictive versus Self-Tuning Adaptive Control Strategies of Biodiesel Reactors. *Industrial & Engineering Chemistry Research, 48*(24), 11034-11047.

Mjalli, F. S., Lee, K. S., Kiew, C. Y., & Hussain, M. A. (2009). Dynamics and control of a biodiesel transesterification reactor. *Chemical Engineering & Technology, 32*(1), 13-26.

Moon, S. M., Clark, R. L., & Cole, D. G. (2005). The recursive generalized predictive feedback control: theory and experiments. *Journal of Sound and Vibration, 279*(1-2), 171-199.

Moon, S. M., Cole, D. G., & Clark, R. L. (2006). Real-time implementation of adaptive feedback and feedforward generalized predictive control algorithm. *Journal of Sound and Vibration, 294*(1-2), 82-96.

Nithya, S., Gour, A. S., Sivakumaran, N., Radhakrishnan, T., K., & Anantharaman, N. (2007). *Predictive controller design for a shell and tube heat exchanger.* Paper presented at the International Conference on Intelligent and Advanced Systems 2007, Kuala Lumpur, Malaysia.

Osbourne, P. V., Whitaker, H. P., & Kezer, A. (1961). New designs in the development of model reference adaptive control systems (pp. Paper 61-39): Institute of aeronautical services.

Park, D. J., Jun, B. E., & Kim, J. H. (1991). Fast tracking RLS algorithm using novel variable forgetting factor with unity zone. *Electronics Letters, 27*(23), 2150-2151.

Potter, J. E., & Stern, R. G. (1963). *Statistical filtering of space navigation measurements.* Paper presented at the AIAA Guidance and Control Conference.

Qin, S. J., & Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice, 11*(7), 733-764.

Rao Sripada, N., & Fisher, D. G. (1987). Improved least squares identification. *International Journal of Control, 46*(6), 1889-1913.

Rawlings, J. B., & Muske, K. R. (1993). The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control, 38*(10), 1512 - 1516.

Rivera, D. E., Morari, M., & Skogestad, S. (1986). Internal model control. 4. PID controller design. *Industrial & Engineering Chemistry Process Design and Development, 25*(1), 252-265.

Rossiter, J. A. (2003). *Model-Based Predictive Control*. Boca Raton, Florida: CRC Press.

Rugh, W. J. (1991). Analytical framework for gain scheduling. *IEEE Control Systems Magazine, 11*(1), 79-84.

Salgado, M. E., Goodwin, G. C., & Middleton, R. H. (1988). Modified least squares algorithm incorporating exponential resetting and forgetting. *International Journal of Control, 47*(2), 477-491.

Seborg, D. E., Edgar, T. F., & Mellichamp, D. A. (2004). *Process Dynamics and Control* (Second ed.). New Jersey: John Wiley & Sons, Inc.

Seborg, D. E., Edgar, T. F., & Shah, S. L. (1986). Adaptive control strategies for process control: A survey. *AIChE Journal, 32*(6), 881-913.

Shah, S. L., & Cluett, W. R. (1991). Recursive least squares based estimation schemes for self-tuning control. *Canadian Journal of Chemical Engineering, 69*(1), 89-96.

Shridhar, R., & Cooper, D. J. (1997a). A novel tuning strategy for multivariable model predictive control. *ISA Transactions, 36*(4), 273-280.

Shridhar, R., & Cooper, D. J. (1997b). A Tuning Strategy for Unconstrained SISO Model Predictive Control. *Industrial and Engineering Chemistry Research, 36*(3), 729-746.

Skogestad, S. (2003). Simple analytic rules for model reduction and PID controller tuning. *Journal of Process Control, 13*(4), 291-309.

Skogestad, S. (2004). Corrections and comments from the author on the paper “Simple analytic rules for model reduction and PID controller tuning”, Journal of Process Control 13 (2003) 291–309 *Journal of Process Control, 14*(4), 465.

Tahami, F., & Ebad, M. (2009). On model predictive control of quasi-resonant converters. *Journal of Circuits, Systems and Computers, 18*(7), 1167 - 1183.

Tapasvi, D., Wiesenborn, D., & Gustafson, C. (2004). *Process modeling approach for evaluating the economic feasibility of biodiesel production*. Paper presented at the North Central ASAE/CSAE Conference.

Tingdahl, D. A. (2007). *Implementation of an adaptive internal model controller for systems with large time delay.* Ph.D, Luleå University of Technology, Kiruna.

Townsend, S., Lightbody, G., Brown, M. D., & Irwin, G. W. (1998). Nonlinear dynamic matrix control using local models. *Transactions of the Institute of Measurement and Control, 20*(1), 47-56.

Trierwieler, J. O., & Farina, L. A. (2003). RPM tuning strategy for model predictive control. *Journal of Process Control, 13*, 591 - 598.

Valencia-Palomoa, G., & Rossiter, J. A. (2010). Programmable logic controller implementation of an auto-tuned predictive control based on minimal plant information. *ISA Transactions, 50*, 92 - 100.

Villanueva Perales, A. L., Ollero, P., Gutierrez Ortiz, F. J., & Gomez-Barea, A. (2009). Model predictive control of a wet limestone flue gas desulfurization pilot plant. *Industrial & Engineering Chemistry Research, 48*(11), 5399-5405.

Vogel, E. F. (1988). An adaptive pole placement controller for chemical processes with variable dead time. *Computers and Chemical Engineering, 12*(1), 15 - 26.

Whitaker, H. P. (1959). An adaptive control system for control of the dynamic performances of aircraft and spacecraft: Institute of aeronautical services.

Whitaker, H. P., Yamron, J., & Kezer, A. (1958). Design of model-reference adaptive control systems for aircrafts: Instrumentation Laboratory, MIT, Cambridge.

Wong, S. K. P., & Seborg, D. E. (1986). A theoretical analysis of Smith and analytical predictors. *AIChE Journal, 32*(10), 1597-1605.

Yamuna, R. K., & Unbehauen, H. (1997). Study of predictive controller tuning methods. *Automatica, 33*(12), 2243 - 2248.

Ydstie, B. E., Kershenbaum, L. S., & Sargent, R. W. H. (1985). Theory and application of an extended horizon self-tuning controller. *AIChE Journal, 31*(11), 1771-1780.

Yoon, T.-W., & Clarke, D. W. (1995). Observer design in receding-horizon predictive control. *International Journal of Control, 61*(1), 171 - 191.

Yu, C., Roy, R. J., Kaufman, H., & Bequette, B. W. (1992). Multiple-model adaptive predictive control of mean arterial pressure and cardiac output. *Biomedical Engineering, IEEE Transactions on, 39*(8), 765-778.

Ziegler, J. G., & Nichols, N. B. (1942). Optimum settings for automatic controllers. *Transactions of the ASME, 64*, 759-768.

APPENDIX A: DERIVATION OF BIERMAN'S UDU^T FACTORIZATION

To fill in the missing details in the original paper (Bierman, 1976), this appendix derives Bierman's $\boldsymbol{UDU}^T$ factorization method of the covariance matrix used in all RLS algorithm implementations throughout this study. Here, the general results of the $\boldsymbol{UDU}^T$ factorization are shown before further modifications are introduced to make the factorization method applicable to specific forms of RLS algorithms.

A.I Overview of the Method

Consider the equation of covariance update shown here:

$$\hat{\boldsymbol{P}} = \boldsymbol{P} - \frac{\boldsymbol{P}\boldsymbol{\psi}\boldsymbol{\psi}^T\boldsymbol{P}}{\varphi} \tag{A.1}$$

where $\varphi = r + \boldsymbol{\psi}^T\boldsymbol{P}\boldsymbol{\psi}$. For simplicity, variables which are not known *a priori* are written with the caret symbol, whereas the omission of the caret symbol denotes variables which are yet unknown.

Direct updating as prescribed in Eqn. (A.1) could lead to instability, hence an indirect method was proposed by Bierman. Supposing that the covariance matrix $\boldsymbol{P}$ at every instance can be factored in the form of Eqn. (A.2):

$$\boldsymbol{P} = \boldsymbol{UDU}^T \tag{A.2}$$

where $\boldsymbol{D} \in \Re^{n\times n}$ is a diagonal matrix, $\boldsymbol{U} \in \Re^{n\times n}$ is an upper triangular matrix with ones in the diagonal, and n is the number of rows (or number of columns) of $\boldsymbol{P}$. Bierman's

method strives to find the updated components of the factored form, *i.e.* the $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{D}}$ components, in terms of $\boldsymbol{U}$ and $\boldsymbol{D}$.

To do this, Eqn. (A.1) can be rewritten in the factored form:

$$\hat{\boldsymbol{U}}\hat{\boldsymbol{D}}\hat{\boldsymbol{U}}^T = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T - \frac{\boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T\boldsymbol{\psi}\boldsymbol{\psi}^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T}{\varphi} \tag{A.3}$$

Following this, Eqn. (A.3) can be rearranged to arrive at Eqn. (A.4):

$$\hat{\boldsymbol{U}}\hat{\boldsymbol{D}}\hat{\boldsymbol{U}}^T = \boldsymbol{U}\left[\boldsymbol{D} - \frac{\boldsymbol{D}\left(\boldsymbol{U}^T\boldsymbol{\psi}\right)\left(\boldsymbol{U}^T\boldsymbol{\psi}\right)^T\boldsymbol{D}}{\varphi}\right]\boldsymbol{U}^T \tag{A.4}$$

Defining $\boldsymbol{h} = \boldsymbol{U}^T\boldsymbol{\psi}$ and $\boldsymbol{x} = \boldsymbol{D}\boldsymbol{h}$, Eqn. (A.4) follows the following transformation:

$$\hat{\boldsymbol{U}}\hat{\boldsymbol{D}}\hat{\boldsymbol{U}}^T = \boldsymbol{U}\left[\boldsymbol{D} - \frac{\boldsymbol{D}\boldsymbol{h}\boldsymbol{h}^T\boldsymbol{D}}{\varphi}\right]\boldsymbol{U}^T \tag{A.5}$$

$$\hat{\boldsymbol{U}}\hat{\boldsymbol{D}}\hat{\boldsymbol{U}}^T = \boldsymbol{U}\left[\boldsymbol{D} - \frac{\boldsymbol{x}\boldsymbol{x}^T}{\varphi}\right]\boldsymbol{U}^T \tag{A.6}$$

since $\boldsymbol{D}^T = \boldsymbol{D}$.

Bierman suggested the following to complete the factorization of the right hand side of Eqn. (A.6): following the rationale of Agee and Turner (1972), the terms in the square bracket are further factored in the form of $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{D}}$:

$$\bar{\boldsymbol{U}}\bar{\boldsymbol{D}}\bar{\boldsymbol{U}}^T = \boldsymbol{D} - \frac{\boldsymbol{x}\boldsymbol{x}^T}{\varphi} \tag{A.7}$$

Upon obtaining $\bar{\boldsymbol{U}}\bar{\boldsymbol{D}}\bar{\boldsymbol{U}}^T$, substitution into A.6 and rearranging slightly, the new factors of the covariance matrix can be found:

$$\hat{\boldsymbol{U}}\hat{\boldsymbol{D}}\hat{\boldsymbol{U}}^T = \left(\boldsymbol{U}\bar{\boldsymbol{U}}\right)\bar{\boldsymbol{D}}\left(\boldsymbol{U}\bar{\boldsymbol{U}}\right)^T \tag{A.8}$$

where $\hat{\boldsymbol{U}} = \boldsymbol{U}\bar{\boldsymbol{U}}$ and $\hat{\boldsymbol{D}} = \bar{\boldsymbol{D}}$.

To conclude, the whole idea in Bierman's method is to update $\hat{\boldsymbol{P}}$ indirectly through its updated factorized components $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{D}}$ at every time instance. The challenge in obtaining $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{D}}$, however, lies in the factorization as shown in Eqn. (A.7), which the rigorous derivation will be shown in the next section.

A.II Rigorous Derivation of Bierman's UDU^T Factorization Method

To begin with the derivation of Bierman's method, consider the following:

$$\begin{aligned}\left[UDU^T\right]_{ik} &= \sum_{j}\left[UD\right]_{ij}U^T_{jk} \\ &= \sum_{j}\sum_{p}U_{ip}D_{pj}U^T_{jk}\end{aligned} \tag{A.9}$$

where the use of subscripts indicate the position of a particular numerical element in the corresponding matrix/vector array. For instance, U_{ip} represents the numerical element in the i-th row and p-th column of the matrix $\boldsymbol{U}$. The subscripts used here are unique within the scope of this appendix, and should not be confused with identical subscripts used in other chapters of this thesis. Eqn. (A.9) can be written as:

$$\begin{aligned}\left[UDU^T\right]_{ik} &= \sum_{j=i}^{n}U_{ij}D_{jj}U^T_{jk} \\ &= \sum_{j=i}^{n}U_{ij}U_{kj}D_{j}\end{aligned} \tag{A.10}$$

To enable the factorization as shown in Eqn. (A.7), it is necessary to revisit the work of Agee and Turner (1972), where the factorization of a positive definite matrix plus a symmetric dyad was first discussed in detail. Although Agee and Turner (1972) performed the factorization in the form of $\boldsymbol{LDL}^T$ (where $\boldsymbol{L}$ is the lower triangular matrix), similar strategy can be used to re-derive the method for the case of $\boldsymbol{UDU}^T$ factorization (which will be shown in this section).

Without going into the mathematical details, Figure A.1 presents the pictorial roadmap of the stages involved in deriving the factorization. To begin deriving the equations, consider the following factorization, where $\boldsymbol{U'}$ is an upper triangular matrix with ones in the diagonal, $\boldsymbol{D'}$ is a diagonal matrix, $\boldsymbol{A}$ is a positive definite matrix, c is a constant, and $\boldsymbol{w}$ is a column vector:

$$\boldsymbol{U}^{'}\boldsymbol{D}^{'}\boldsymbol{U}^{'T} = \boldsymbol{A} + c\boldsymbol{w}\boldsymbol{w}^{T} \tag{A.11}$$

The right hand side of Eqn. (A.11) resembles that of a positive definite matrix plus a symmetric dyad, which according to Agee and Turner (1972) can be factorized. The objective here is to deduce a set of general relationships which can be used to obtain $\boldsymbol{U}^{'}$ and $\boldsymbol{D}^{'}$.

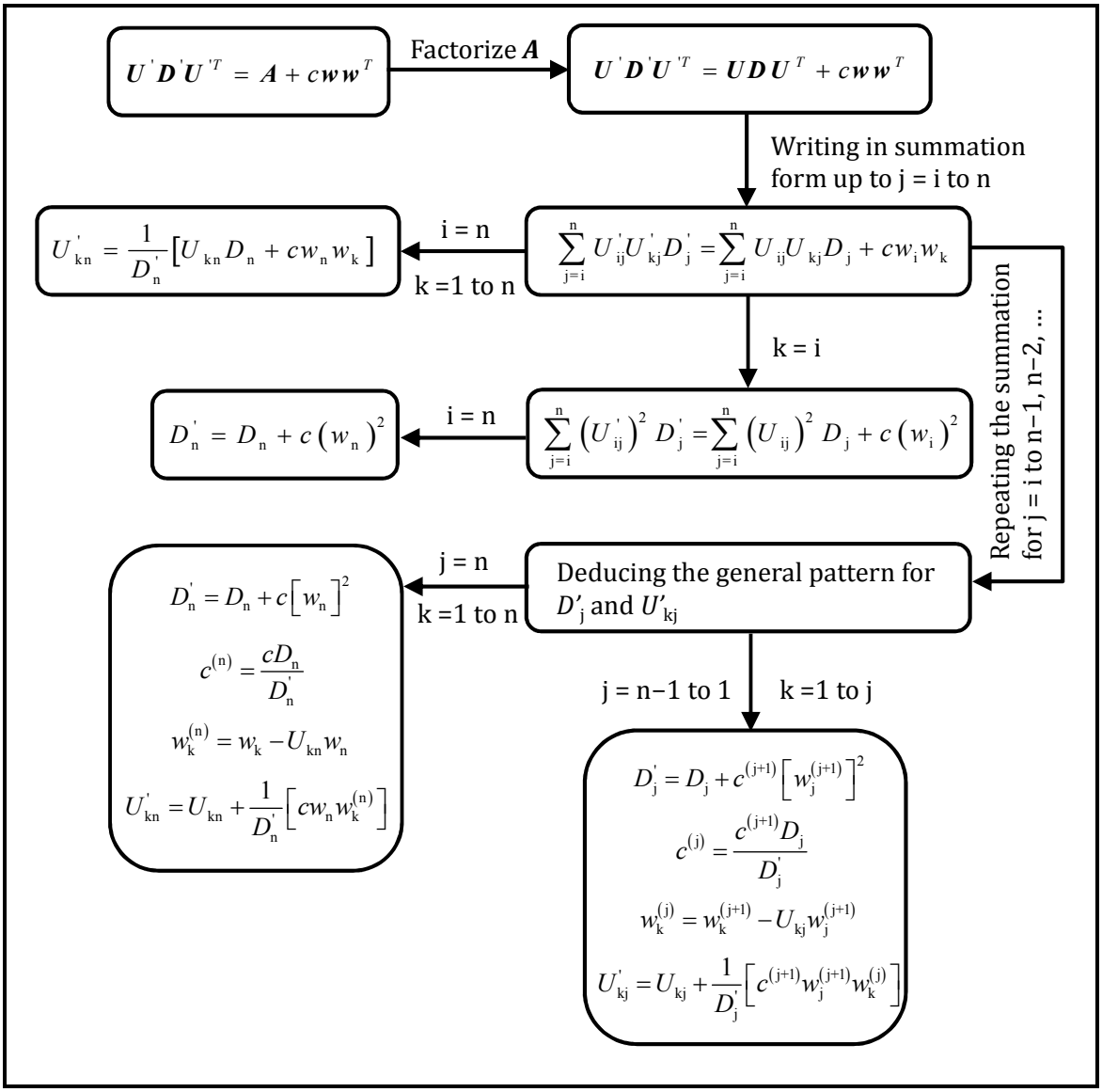

Figure A.1: Pictorial roadmap showing the stages involved in deriving useful equations for the factorization of a positive definite matrix plus a symmetric dyad.

From Eqn. (A.11), since $\boldsymbol{A}$ is a positive definite matrix, it can also be factored in the form of $\boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{T}$, resulting in:

$$\boldsymbol{U}'\boldsymbol{D}'\boldsymbol{U}'^{T} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{T} + c\boldsymbol{w}\boldsymbol{w}^{T} \tag{A.12}$$

Using Eqn. (A.10), Eqn. (A.12) can be written in the following form for every numerical element in its left hand side matrix (*i.e.* the $\boldsymbol{U}'\boldsymbol{D}'\boldsymbol{U}'^{T}$ matrix):

$$\sum_{j=i}^{n} U'_{ij}U'_{kj}D'_{j} = \sum_{j=i}^{n} U_{ij}U_{kj}D_{j} + cw_{i}w_{k} \tag{A.13}$$

When k = i:

$$\sum_{j=i}^{n}\left(U_{ij}^{'}\right)^{2} D_{j}^{'}=\sum_{j=i}^{n}\left(U_{ij}\right)^{2} D_{j}+c\left(w_{i}\right)^{2} \tag{A.14}$$

From Eqns. (A.13) and (A.14), when i = n, the summations are the simplest as they have one term each, the resulting expressions are as shown in Eqns. (A.15) and (A.16) respectively:

$$U_{nn}^{'}U_{kn}^{'}D_{n}^{'}=U_{nn}U_{kn}D_{n}+cw_{n}w_{k} \tag{A.15}$$

$$\left(U_{nn}^{'}\right)^{2} D_{n}^{'}=\left(U_{nn}\right)^{2} D_{n}+c\left(w_{n}\right)^{2} \tag{A.16}$$

Since $U_{nn}^{'}=U_{nn}=1$, Eqns. (A.15) and (A.16) becomes:

$$U_{kn}^{'}=\frac{1}{D_{n}^{'}}\left[U_{kn}D_{n}+cw_{n}w_{k}\right] \tag{A.17}$$

$$D_{n}^{'}=D_{n}+c\left(w_{n}\right)^{2} \tag{A.18}$$

The right hand sides of Eqns. (A.17) and (A.18) are known *a priori*, hence $D_{n}^{'}$ and $U_{kn}^{'}$ can be easily found, for $k=1,2,\ldots,n$.

Up to this point, it was shown how $D_{n}^{'}$ and $U_{kn}^{'}$ can be obtained. Ultimately, the goal of this section is to show how $D_{n}^{'},D_{n-1}^{'},\ldots,D_{1}^{'}$ and $U_{kn}^{'},U_{k,n-1}^{'},\ldots,U_{k1}^{'}$ can be found through a set of general relationships. For clarity, a comma is used to separate the row and column indexes for $U_{k,n-1}^{'}$ and this convention will be adopted hereafter for similar representations throughout this appendix. This is unrelated to the shorthand convention where a comma implies a derivative.

To deduce the general relationships, the derivations to obtain D'_{n-1}, D'_{n-2} and $U'_{k,n-1}, U'_{k,n-2}$ will be shown in the following text. To proceed, consider rewriting Eqn. (A.13) in the following form:

$$\sum_{j=i}^{n-1} U'_{ij} U'_{kj} D'_j = \sum_{j=i}^{n-1} U_{ij} U_{kj} D_j + E \tag{A.19}$$

where $E = U_{in} U_{kn} D_n - U'_{in} U'_{kn} D'_n + c w_i w_k$.

To simplify the expression *E* in Eqn. (A.19), consider Eqn. (A.17) with k = i:

$$U'_{in} D'_n = U_{in} D_n + c w_n w_i \tag{A.20}$$

Multiplying Eqns. (A.17) and (A.20) produces the following expression:

$$U'_{in} U'_{kn} D'_n = \frac{1}{D'_n}\left[U_{in} U_{kn} \left(D_n\right)^2 + \left(c w_n\right)^2 w_i w_k + c D_n w_n \left(U_{kn} w_i + U_{in} w_k\right) \right] \tag{A.21}$$

Substituting Eqn. (A.21) into the expression *E* and rearranging gives:

$$\begin{aligned} E &= \frac{c D_n}{D'_n}\left[U_{in} U_{kn} \left(w_n\right)^2 - U_{kn} w_i w_n - U_{in} w_k w_n + w_i w_k \right] \\ &= \frac{c D_n}{D'_n}\left(w_i - U_{in} w_n\right)\left(w_k - U_{kn} w_n\right) \\ &= c^{(n)} w_i^{(n)} w_k^{(n)} \end{aligned} \tag{A.22}$$

where:

$$c^{(n)} = \frac{c D_n}{D'_n}$$

$$w_i^{(n)} = w_i - U_{in} w_n \tag{A.23}$$

$$w_k^{(n)} = w_k - U_{kn} w_n$$

Rewriting Eqn. (A.19) with the simplified expression of *E* leads to Eqn. (A.24):

$$\sum_{j=i}^{n-1} U'_{ij} U'_{kj} D'_j = \sum_{j=i}^{n-1} U_{ij} U_{kj} D_j + c^{(n)} w_i^{(n)} w_k^{(n)} \tag{A.24}$$

Note that at this point, the form of Eqn. (A.24) is similar to that of Eqn. (A.13). To find D'_{n-1} and $U'_{k,n-1}$, similar procedures for obtaining Eqns. (A.14) – (A.18) as shown previously are repeated here. Thus, when k = i, Eqn. (A.24) becomes:

$$\sum_{j=i}^{n-1}\left(U'_{ij}\right)^2 D'_j = \sum_{j=i}^{n-1}\left(U_{ij}\right)^2 D_j + c^{(n)}\left[w_i^{(n)}\right]^2 \tag{A.25}$$

Again, starting from the smallest summation terms, from Eqns. (A.24) and (A.25), when i = n - 1, the resulting expressions are as shown in Eqns. (A.26) and (A.27) respectively:

$$U'_{n-1,n-1}U'_{k,n-1}D'_{n-1} = U_{n-1,n-1}U_{k,n-1}D_{n-1} + c^{(n)}w_{n-1}^{(n)}w_k^{(n)} \tag{A.26}$$

$$\left(U'_{n-1,n-1}\right)^2 D'_{n-1} = \left(U_{n-1,n-1}\right)^2 D_{n-1} + c^{(n)}\left[w_{n-1}^{(n)}\right]^2 \tag{A.27}$$

Substituting for $U'_{n-1,n-1} = U_{n-1,n-1} = 1$, Eqns. (A.26) and (A.27) become:

$$U'_{k,n-1} = \frac{1}{D'_{n-1}}\left[U_{k,n-1}D_{n-1} + c^{(n)}w_{n-1}^{(n)}w_k^{(n)}\right] \tag{A.28}$$

$$D'_{n-1} = D_{n-1} + c^{(n)}\left[w_{n-1}^{(n)}\right]^2 \tag{A.29}$$

Hence, D'_{n-1} and $U'_{k,n-1}$ are found from Eqns. (A.28) and (A.29), for $k = 1, 2, \ldots, n-1$.

Subsequently, similar procedures as shown above are repeated to obtain D'_{n-2} and $U'_{k,n-2}$. Equation (A.24) is first recast in the following form:

$$\sum_{j=i}^{n-2}U'_{ij}U'_{kj}D'_j = \sum_{j=i}^{n-2}U_{ij}U_{kj}D_j + F \tag{A.30}$$

where $F = U_{i,n-1}U_{k,n-1}D_{n-1} - U'_{i,n-1}U'_{k,n-1}D'_{n-1} + c^{(n)}w_i^{(n)}w_k^{(n)}$.

When k = i, Eqn. (A.28) becomes:

$$U'_{i,n-1}D'_{n-1} = U_{i,n-1}D_{n-1} + c^{(n)}w^{(n)}_{n-1}w^{(n)}_{i} \tag{A.31}$$

Similarly, multiplying Eqns. (A.28) and (A.31) produces the following expression:

$$\begin{aligned} &U'_{i,n-1}U'_{k,n-1}D'_{n-1} \\ &= \frac{1}{D'_{n-1}}\left\{U_{i,n-1}U_{k,n-1}\left(D_{n-1}\right)^2 + \left[c^{(n)}w^{(n)}_{n-1}\right]^2 w^{(n)}_{i}w^{(n)}_{k} + c^{(n)}D_{n-1}w^{(n)}_{n-1}\left[U_{k,n-1}w^{(n)}_{i} + U_{i,n-1}w^{(n)}_{k}\right]\right\} \end{aligned} \tag{A.32}$$

Substituting Eqn. (A.32) into the expression ***F*** and rearranging gives:

$$\begin{aligned} F &= \frac{c^{(n)}D_{n-1}}{D'_{n-1}}\left\{U_{i,n-1}U_{k,n-1}\left[w^{(n)}_{n-1}\right]^2 - U_{k,n-1}w^{(n)}_{i}w^{(n)}_{n-1} - U_{i,n-1}w^{(n)}_{k}w^{(n)}_{n-1} + w^{(n)}_{i}w^{(n)}_{k}\right\} \\ &= \frac{c^{(n)}D_{n-1}}{D'_{n-1}}\left[w^{(n)}_{i} - U_{i,n-1}w^{(n)}_{n-1}\right]\left[w^{(n)}_{k} - U_{k,n-1}w^{(n)}_{n-1}\right] \\ &= c^{(n-1)}w^{(n-1)}_{i}w^{(n-1)}_{k} \end{aligned} \tag{A.33}$$

where:

$$c^{(n-1)} = \frac{c^{(n)}D_{n-1}}{D'_{n-1}}$$

$$w^{(n-1)}_{i} = w^{(n)}_{i} - U_{i,n-1}w^{(n)}_{n-1} \tag{A.34}$$

$$w^{(n-1)}_{k} = w^{(n)}_{k} - U_{k,n-1}w^{(n)}_{n-1}$$

Equation (A.30) with the simplified expression for *F* becomes:

$$\sum_{j=i}^{n-2} U'_{ij}U'_{kj}D'_{j} = \sum_{j=i}^{n-2} U_{ij}U_{kj}D_{j} + c^{(n-1)}w^{(n-1)}_{i}w^{(n-1)}_{k} \tag{A.35}$$

When k = i, Eqn. (A.35) takes the form of:

$$\sum_{j=i}^{n-2}\left(U'_{ij}\right)^2 D'_{j} = \sum_{j=i}^{n-2}\left(U_{ij}\right)^2 D_{j} + c^{(n-1)}\left[w^{(n-1)}_{i}\right]^2 \tag{A.36}$$

When i = n − 2, Eqns. (A.35) and (A.36) are simplified to the following expressions:

$$U'_{n-2,n-2}U'_{k,n-2}D'_{n-2} = U_{n-2,n-2}U_{k,n-2}D_{n-2} + c^{(n-1)}w^{(n-1)}_{n-2}w^{(n-1)}_{k} \tag{A.37}$$

$$\left(U'_{n-2,n-2}\right)^2 D'_{n-2} = \left(U_{n-2,n-2}\right)^2 D_{n-2} + c^{(n-1)}\left[w^{(n-1)}_{n-2}\right]^2 \tag{A.38}$$

Given $U'_{n-2,n-2} = U_{n-2,n-2} = 1$, Eqns. (A.37) and (A.38) become:

$$U'_{k,n-2} = \frac{1}{D'_{n-2}}\left[U_{k,n-2}D_{n-2} + c^{(n-1)}w^{(n-1)}_{n-2}w^{(n-1)}_{k}\right] \tag{A.39}$$

$$D'_{n-2} = D_{n-2} + c^{(n-1)}\left[w^{(n-1)}_{n-2}\right]^2 \tag{A.40}$$

Hence, D'_{n-2} and $U'_{k,n-2}$ are obtained from Eqns. (A.39) and (A.40), for $k = 1, 2, \ldots, n-2$.

Similar procedures as shown above can be used to obtain the expressions for the remaining elements, *viz*. $D'_{n-3}, \ldots, D'_{1}$ and $U'_{k,n-3}, \ldots, U'_{k1}$. Derivations above suggest a set of general relationships which is more practical for computer implementations. They are:

$$\begin{aligned}
c^{(n)} &= \frac{cD_n}{D'_n} \\
c^{(n-1)} &= \frac{c^{(n)}D_{n-1}}{D'_{n-1}} \\
c^{(n-2)} &= \frac{c^{(n-1)}D_{n-2}}{D'_{n-2}} \\
&\vdots \\
c^{(j)} &= \frac{c^{(j+1)}D_j}{D'_j}
\end{aligned} \tag{A.41}$$

$$\begin{aligned}
w^{(n)}_{k} &= w_k - U_{kn}w_n \\
w^{(n-1)}_{k} &= w^{(n)}_{k} - U_{k,n-1}w^{(n)}_{n-1} \\
w^{(n-2)}_{k} &= w^{(n-1)}_{k} - U_{k,n-2}w^{(n-1)}_{n-2} \\
&\vdots \\
w^{(j)}_{k} &= w^{(j+1)}_{k} - U_{kj}w^{(j+1)}_{j}
\end{aligned} \tag{A.42}$$

$$
\begin{aligned}
D_{\mathrm{n}}^{'} &= D_{\mathrm{n}} + c\left(w_{\mathrm{n}}\right)^2 \\
D_{\mathrm{n-1}}^{'} &= D_{\mathrm{n-1}} + c^{(\mathrm{n})}\left[w_{\mathrm{n-1}}^{(\mathrm{n})}\right]^2 \\
D_{\mathrm{n-2}}^{'} &= D_{\mathrm{n-2}} + c^{(\mathrm{n-1})}\left[w_{\mathrm{n-2}}^{(\mathrm{n-1})}\right]^2 \\
&\vdots \\
D_{\mathrm{j}}^{'} &= D_{\mathrm{j}} + c^{(\mathrm{j+1})}\left[w_{\mathrm{j}}^{(\mathrm{j+1})}\right]^2
\end{aligned}
\tag{A.43}
$$

$$
\begin{aligned}
U_{\mathrm{kn}}^{'} &= \frac{1}{D_{\mathrm{n}}^{'}}\left[U_{\mathrm{kn}} D_{\mathrm{n}} + c w_{\mathrm{n}} w_{\mathrm{k}}\right] \\
U_{\mathrm{k,n-1}}^{'} &= \frac{1}{D_{\mathrm{n-1}}^{'}}\left[U_{\mathrm{k,n-1}} D_{\mathrm{n-1}} + c^{(\mathrm{n})} w_{\mathrm{n-1}}^{(\mathrm{n})} w_{\mathrm{k}}^{(\mathrm{n})}\right] \\
U_{\mathrm{k,n-2}}^{'} &= \frac{1}{D_{\mathrm{n-2}}^{'}}\left[U_{\mathrm{k,n-2}} D_{\mathrm{n-2}} + c^{(\mathrm{n-1})} w_{\mathrm{n-2}}^{(\mathrm{n-1})} w_{\mathrm{k}}^{(\mathrm{n-1})}\right] \\
&\vdots \\
U_{\mathrm{kj}}^{'} &= \frac{1}{D_{\mathrm{j}}^{'}}\left[U_{\mathrm{kj}} D_{\mathrm{j}} + c^{(\mathrm{j+1})} w_{\mathrm{j}}^{(\mathrm{j+1})} w_{\mathrm{k}}^{(\mathrm{j+1})}\right]
\end{aligned}
\tag{A.44}
$$

where $\mathrm{j} = 1, 2, \ldots, \mathrm{n}$ (note that $c^{(\mathrm{n+1})} = c$, $w_{\mathrm{k}}^{(\mathrm{n+1})} = w_{\mathrm{k}}$, and $w_{\mathrm{j}}^{(\mathrm{n+1})} = w_{\mathrm{j}}$) and $\mathrm{k} = 1, 2, \ldots, \mathrm{j}$. Using Eqns. (A.42) and (A.43), an alternative general form of Eqn. (A.44) can be derived:

$$
U_{\mathrm{kj}}^{'} = U_{\mathrm{kj}} + \frac{1}{D_{\mathrm{j}}^{'}}\left[c^{(\mathrm{j+1})} w_{\mathrm{j}}^{(\mathrm{j+1})} w_{\mathrm{k}}^{(\mathrm{j})}\right] \tag{A.45}
$$

To summarize, given the *a priori* knowledge of c, $\boldsymbol{U}$, $\boldsymbol{D}$, and $\boldsymbol{w}$, the sequence of calculations for obtaining $\boldsymbol{U}^{'}$ and $\boldsymbol{D}^{'}$ matrices are shown here:

Starting from $\mathrm{j} = \mathrm{n}$, calculate:

$$
D_{\mathrm{n}}^{'} = D_{\mathrm{n}} + c\left(w_{\mathrm{n}}\right)^2 \tag{A.46}
$$

$$
c^{(\mathrm{n})} = \frac{c D_{\mathrm{n}}}{D_{\mathrm{n}}^{'}} \tag{A.47}
$$

$$
\left.
\begin{aligned}
w_{\mathrm{k}}^{(\mathrm{n})} &= w_{\mathrm{k}} - U_{\mathrm{kn}} w_{\mathrm{n}} \\
U_{\mathrm{kn}}^{'} &= U_{\mathrm{kn}} + \frac{1}{D_{\mathrm{n}}^{'}}\left[c w_{\mathrm{n}} w_{\mathrm{k}}^{(\mathrm{n})}\right]
\end{aligned}
\right\} \mathrm{k} = 1, 2, \ldots, \mathrm{n} \tag{A.48}
$$

Subsequently, for j = n – 1 to 1, compute the following:

$$D_{\mathrm{j}}^{'} = D_{\mathrm{j}} + c^{(\mathrm{j}+1)}\left[w_{\mathrm{j}}^{(\mathrm{j}+1)}\right]^2 \tag{A.49}$$

$$c^{(\mathrm{j})} = \frac{c^{(\mathrm{j}+1)} D_{\mathrm{j}}}{D_{\mathrm{j}}^{'}} \tag{A.50}$$

$$\left.\begin{aligned} w_{\mathrm{k}}^{(\mathrm{j})} &= w_{\mathrm{k}}^{(\mathrm{j}+1)} - U_{\mathrm{kj}} w_{\mathrm{j}}^{(\mathrm{j}+1)} \\ U_{\mathrm{kj}}^{'} &= U_{\mathrm{kj}} + \frac{1}{D_{\mathrm{j}}^{'}}\left[c^{(\mathrm{j}+1)} w_{\mathrm{j}}^{(\mathrm{j}+1)} w_{\mathrm{k}}^{(\mathrm{j})}\right] \end{aligned}\right\} \mathrm{k} = 1, 2, \ldots, \mathrm{j} \tag{A.51}$$

With Eqns. (A.46) - (A.51), the factorization as shown in Eqn. (A.7) can now be done. For this specific factorization, consider the following relationships between Eqns. (A.7) and (A.11):

$$\boldsymbol{A} \equiv \boldsymbol{D}$$

$$c \equiv -\frac{1}{\varphi}$$

$$\boldsymbol{w} \equiv \boldsymbol{x}$$

$$\boldsymbol{U}^{'} \equiv \bar{\boldsymbol{U}}$$

$$\boldsymbol{D}^{'} \equiv \bar{\boldsymbol{D}} \tag{A.52}$$

In order for the first relationship in Eqn. (A.52) to hold true, clearly:

$$\begin{aligned} &\boldsymbol{U} = \boldsymbol{U}^T = \boldsymbol{I} \\ &\therefore \boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T = \boldsymbol{I}\boldsymbol{D}\boldsymbol{I}^T = \boldsymbol{D} \end{aligned} \tag{A.53}$$

Using Eqns. (A.52), Eqns. (A.46) - (A.51) after some rearrangements become:

Starting from j = n :

$$\bar{D}_{\mathrm{n}} = D_{\mathrm{n}} - \frac{1}{\varphi}\left(x_{\mathrm{n}}\right)^2 \tag{A.54}$$

$$\varphi^{(\mathrm{n})} = \frac{\varphi \bar{D}_{\mathrm{n}}}{D_{\mathrm{n}}} \tag{A.55}$$

$$x_k^{(n)} = \begin{cases} 0 & \text{if } k = n \\ x_k & \text{otherwise} \end{cases}$$

$$\bar{U}_{kn} = \begin{cases} 1 & \text{if } k = n \\ -\dfrac{1}{\varphi \bar{D}_n} x_n x_k & \text{otherwise} \end{cases} \Bigg\} k = 1, 2, \ldots, j \tag{A.56}$$

Subsequently, for j = n – 1 to 1:

$$\bar{D}_j = D_j - \frac{1}{\varphi^{(j+1)}} \left[x_j^{(j+1)} \right]^2 \tag{A.57}$$

$$\varphi^{(j)} = \frac{\varphi^{(j+1)} \bar{D}_j}{D_j} \tag{A.58}$$

$$x_k^{(j)} = \begin{cases} 0 & \text{if } k = j \\ x_k^{(j+1)} & \text{otherwise} \end{cases}$$

$$\bar{U}_{kj} = \begin{cases} 1 & \text{if } k = j \\ -\dfrac{1}{\varphi^{(j+1)} \bar{D}_j} x_j^{(j+1)} x_k^{(j)} & \text{otherwise} \end{cases} \Bigg\} k = 1, 2, \ldots, j \tag{A.59}$$

Up to this stage, the problem is in principle solved, *i.e.* $\bar{\boldsymbol{D}}$ and $\bar{\boldsymbol{U}}$ had been found in terms of known quantities. However, the equations are inconvenient as they involve φ; further they are not yet in the final form reported by Bierman. To ease understanding of the following derivations, Figure A.2 shows the pictorial roadmap of the steps ahead in obtaining the final form of equations.

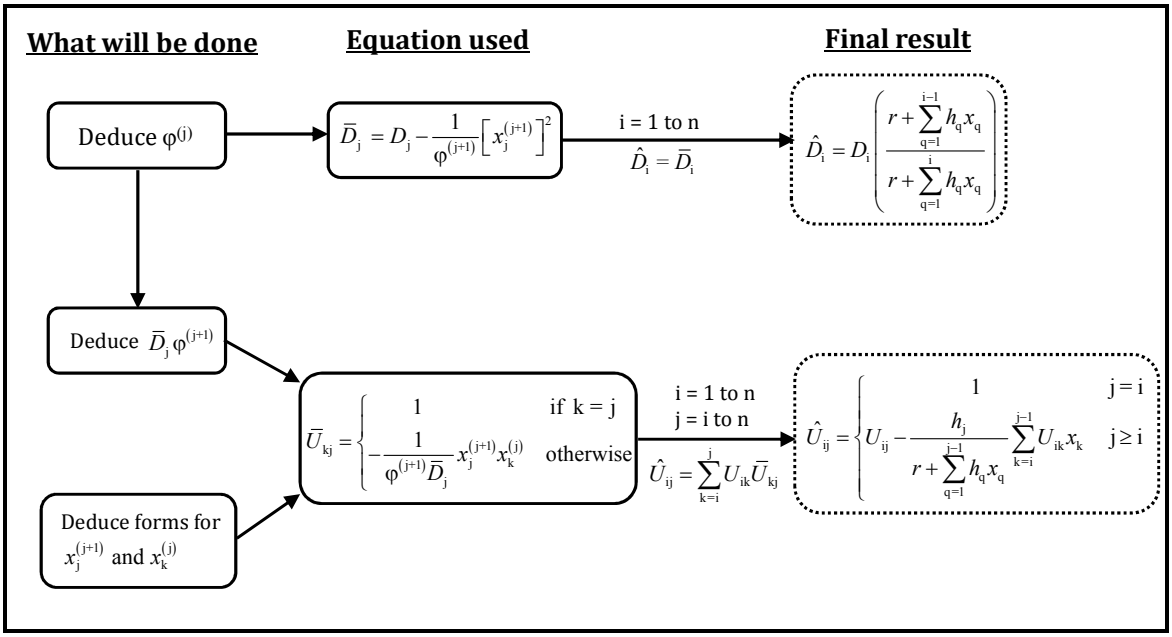

Figure A.2: Pictorial roadmap showing the steps involved to obtain the final version of Bierman's factorization algorithm. The blocks with dotted borders represent the final forms useful for implementation.

Taking into consideration the expression $\varphi = r + \boldsymbol{\psi}^T \boldsymbol{P}\boldsymbol{\psi}$, Eqns. (A.54) - (A.59) can be further simplified. To proceed, consider the following:

$$\begin{aligned}\varphi &= r + \boldsymbol{\psi}^T \boldsymbol{P}\boldsymbol{\psi} \\ &= r + \left(\boldsymbol{\psi}^T \boldsymbol{U}\right)\left(\boldsymbol{D}\boldsymbol{U}^T\boldsymbol{\psi}\right) \\ &= r + \boldsymbol{h}^T \boldsymbol{x} \\ &= r + \sum_{q=1}^{n} h_q x_q\end{aligned} \tag{A.60}$$

Using Eqn. (A.60) in Eqn. (A.54) gives:

$$\begin{aligned}\bar{D}_{\mathrm{n}} &= D_{\mathrm{n}} - \frac{1}{\varphi}\left(x_{\mathrm{n}}\right)^2 \\ &= \frac{\varphi D_{\mathrm{n}} - \left(x_{\mathrm{n}}\right)^2}{\varphi} \\ &= \frac{rD_{\mathrm{n}} + \left(\sum_{\mathrm{q}=1}^{\mathrm{n}} h_{\mathrm{q}} x_{\mathrm{q}}\right) D_{\mathrm{n}} - \left(x_{\mathrm{n}}\right)^2}{r + \sum_{\mathrm{q}=1}^{\mathrm{n}} h_{\mathrm{q}} x_{\mathrm{q}}} \\ &= \frac{rD_{\mathrm{n}} + \left(\sum_{\mathrm{q}=1}^{\mathrm{n}-1} h_{\mathrm{q}} x_{\mathrm{q}}\right) D_{\mathrm{n}} + h_{\mathrm{n}}\left(D_{\mathrm{n}} h_{\mathrm{n}}\right) D_{\mathrm{n}} - \left(D_{\mathrm{n}} h_{\mathrm{n}}\right)^2}{r + \sum_{\mathrm{q}=1}^{\mathrm{n}} h_{\mathrm{q}} x_{\mathrm{q}}} \\ &= D_{\mathrm{n}} \left(\frac{r + \sum_{\mathrm{q}=1}^{\mathrm{n}-1} h_{\mathrm{q}} x_{\mathrm{q}}}{r + \sum_{\mathrm{q}=1}^{\mathrm{n}} h_{\mathrm{q}} x_{\mathrm{q}}} \right)\end{aligned} \tag{A.61}$$

To find $\bar{D}_{\mathrm{n}-1}$, Eqns. (A.55) and (A.56) can first be rewritten in the following forms:

$$\begin{aligned}\varphi^{(\mathrm{n})} &= \frac{\varphi \bar{D}_{\mathrm{n}}}{D_{\mathrm{n}}} \\ &= \left(r + \sum_{\mathrm{q}=1}^{\mathrm{n}} h_{\mathrm{q}} x_{\mathrm{q}} \right) \left(\frac{r + \sum_{\mathrm{q}=1}^{\mathrm{n}-1} h_{\mathrm{q}} x_{\mathrm{q}}}{r + \sum_{\mathrm{q}=1}^{\mathrm{n}} h_{\mathrm{q}} x_{\mathrm{q}}} \right) \\ &= r + \sum_{\mathrm{q}=1}^{\mathrm{n}-1} h_{\mathrm{q}} x_{\mathrm{q}}\end{aligned} \tag{A.62}$$

$$x_{\mathrm{n}-1}^{(\mathrm{n})} = x_{\mathrm{n}-1} \tag{A.63}$$

Substituting Eqns. (A.62) − (A.63) in Eqns. (A.57) for j = n − 1 yields:

$$\begin{aligned}\bar{D}_{n-1} &= D_{n-1} - \frac{1}{r + \sum_{q=1}^{n-1} h_q x_q} \left(x_{n-1}\right)^2 \\ &= \frac{r D_{n-1} + \left(\sum_{q=1}^{n-2} h_q x_q\right) D_{n-1} + h_{n-1}\left(D_{n-1} h_{n-1}\right) D_{n-1} - \left(D_{n-1} h_{n-1}\right)^2}{r + \sum_{q=1}^{n-1} h_q x_q} \\ &= D_{n-1}\left(\frac{r + \sum_{q=1}^{n-2} h_q x_q}{r + \sum_{q=1}^{n-1} h_q x_q}\right)\end{aligned} \tag{A.64}$$

From Eqns. (A.61) and (A.64), a general relationship is observed (for j = n to 1):

$$\bar{D}_j = D_j \left(\frac{r + \sum_{q=1}^{j-1} h_q x_q}{r + \sum_{q=1}^{j} h_q x_q}\right) \tag{A.65}$$

where the solution to $\hat{\boldsymbol{D}}$ in Eqn. (A.8) is found through $\hat{D}_j = \bar{D}_j$.

Next, from Eqns. (A.56) and (A.59), the following can be written:

$$\begin{aligned}
\left[\boldsymbol{x}^{(\mathrm{n})}\right]^T &= \left[x_1^{(\mathrm{n})} \quad x_2^{(\mathrm{n})} \quad \cdots \quad x_{\mathrm{n}-2}^{(\mathrm{n})} \quad x_{\mathrm{n}-1}^{(\mathrm{n})} \quad x_{\mathrm{n}}^{(\mathrm{n})}\right] \\
&= \left[x_1 \quad x_2 \quad \cdots \quad x_{\mathrm{n}-2} \quad x_{\mathrm{n}-1} \quad 0\right] \\
\left[\boldsymbol{x}^{(\mathrm{n}-1)}\right]^T &= \left[x_1^{(\mathrm{n}-1)} \quad x_2^{(\mathrm{n}-1)} \quad \cdots \quad x_{\mathrm{n}-2}^{(\mathrm{n}-1)} \quad 0 \quad x_{\mathrm{n}}^{(\mathrm{n}-1)}\right] \\
&= \left[x_1^{(\mathrm{n})} \quad x_2^{(\mathrm{n})} \quad \cdots \quad x_{\mathrm{n}-2}^{(\mathrm{n})} \quad 0 \quad 0\right] \\
&= \left[x_1 \quad x_2 \quad \cdots \quad x_{n-2} \quad 0 \quad 0\right] \\
\left[\boldsymbol{x}^{(\mathrm{n}-2)}\right]^T &= \left[x_1^{(\mathrm{n}-2)} \quad x_2^{(\mathrm{n}-2)} \quad \cdots \quad 0 \quad x_{\mathrm{n}-1}^{(\mathrm{n}-2)} \quad x_{\mathrm{n}}^{(\mathrm{n}-2)}\right] \\
&= \left[x_1^{(\mathrm{n}-1)} \quad x_2^{(\mathrm{n}-1)} \quad \cdots \quad 0 \quad 0 \quad x_{\mathrm{n}}^{(\mathrm{n}-1)}\right] \\
&= \left[x_1^{(\mathrm{n})} \quad x_2^{(\mathrm{n})} \quad \cdots \quad 0 \quad 0 \quad 0\right] \\
&= \left[x_1 \quad x_2 \quad \cdots \quad 0 \quad 0 \quad 0\right] \\
&\vdots \\
\left[\boldsymbol{x}^{(2)}\right]^T &= \left[x_1 \quad 0 \quad \cdots \quad 0\right] \\
\left[\boldsymbol{x}^{(1)}\right]^T &= \left[0\right]
\end{aligned} \tag{A.66}$$

Hence, Eqn. (A.66) can be summarized as:

$$x_{\mathrm{k}}^{(\mathrm{j})} = \begin{cases} 0 & \text{if } \mathrm{k} \geq \mathrm{j} \\ x_{\mathrm{k}} & \text{otherwise} \end{cases} \tag{A.67}$$

$$x_{\mathrm{j}}^{(\mathrm{j}+1)} = x_{\mathrm{j}} \tag{A.68}$$

To deduce the general relationships for $\varphi^{(\mathrm{j})}$, recall from Eqn. (A.62) that $\varphi^{(\mathrm{n})} = r + \sum_{\mathrm{q}=1}^{\mathrm{n}-1} h_{\mathrm{q}} x_{\mathrm{q}}$. Following this, using Eqns. (A.58) and (A.65), the following expressions are deduced:

$$
\begin{aligned}
\varphi^{(n-1)} &= \frac{\varphi^{(n)}\bar{D}_{n-1}}{D_{n-1}} \\
&= \left(r + \sum_{q=1}^{n-1} h_q x_q \right) \left(\frac{r + \sum_{q=1}^{n-2} h_q x_q}{r + \sum_{q=1}^{n-1} h_q x_q} \right) \\
&= r + \sum_{q=1}^{n-2} h_q x_q \\
\varphi^{(n-2)} &= \frac{\varphi^{(n-1)}\bar{D}_{n-2}}{D_{n-2}} \\
&= \left(r + \sum_{q=1}^{n-2} h_q x_q \right) \left(\frac{r + \sum_{q=1}^{n-3} h_q x_q}{r + \sum_{q=1}^{n-3} h_q x_q} \right) \\
&= r + \sum_{q=1}^{n-3} h_q x_q \\
&\vdots \\
\varphi^{(j)} &= r + \sum_{q=1}^{j-1} h_q x_q
\end{aligned}
\tag{A.69}
$$

Using results from Eqn. (A.69), Eqn. (A.65) can be recast in the following form:

$$
\begin{aligned}
\bar{D}_j \varphi^{(j+1)} &= \varphi^{(j)} D_j \\
&= D_j \left(r + \sum_{q=1}^{j-1} h_q x_q \right)
\end{aligned}
\tag{A.70}
$$

Hence, using Eqns. (A.67) – (A.68), for k < j, Eqn. (A. 59) becomes:

$$
\begin{aligned}
\bar{U}_{kj} &= -\frac{x_j x_k}{D_j \left(r + \sum_{q=1}^{j-1} h_q x_q \right)} \\
&= -\frac{\left(D_j h_j \right) x_k}{D_j \left(r + \sum_{q=1}^{j-1} h_q x_q \right)} \\
&= -\frac{h_j x_k}{r + \sum_{q=1}^{j-1} h_q x_q}
\end{aligned}
\tag{A.71}
$$

From Eqn. (A.8), $\hat{\boldsymbol{U}} = \boldsymbol{U}\bar{\boldsymbol{U}}$. With this, using Eqn. (A.71), it can be shown that for $U_{ik} \neq 0$ (k ≥ i) and $\bar{U}_{kj} \neq 0$ (j ≥ k), the entries $\hat{U}_{ij} \neq 0$ (j ≥ i) become:

$$\begin{aligned}
\hat{U}_{ij} &= \sum_{k=i}^{j} U_{ik}\bar{U}_{kj} \\
&= U_{ij}\bar{U}_{jj} + \sum_{k=i}^{j-1} U_{ik}\bar{U}_{kj} \\
&= U_{ij} - \sum_{k=i}^{j-1} \frac{U_{ik}h_j x_k}{r + \sum_{q=1}^{j-1} h_q x_q} \quad \because \bar{U}_{jj} = 1 \\
&= \begin{cases} 1 & j = i \\ U_{ij} - \dfrac{h_j}{r + \sum_{q=1}^{j-1} h_q x_q} \sum_{k=i}^{j-1} U_{ik}x_k & j \geq i \end{cases}
\end{aligned} \tag{A.72}$$

To summarize the factorization method presented in this section, the final version of Bierman's algorithm tailored for Eqn. (A.1) is given here in sequence:

a) Compute the $\boldsymbol{U}$ and $\boldsymbol{D}$ matrices by the factorization of $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$.

b) Compute $\boldsymbol{h} = \boldsymbol{U}^T\boldsymbol{\psi}$.

c) Compute $\boldsymbol{x} = \boldsymbol{D}\boldsymbol{h}$.

d) Compute the following for i = 1 to n and j = i to n:

$$\hat{D}_i = D_i \left(\frac{r + \sum_{q=1}^{i-1} h_q x_q}{r + \sum_{q=1}^{i} h_q x_q} \right) \tag{A.73}$$

$$\hat{U}_{ij} = \begin{cases} 1 & j = i \\ U_{ij} - \dfrac{h_j}{r + \sum_{q=1}^{j-1} h_q x_q} \sum_{k=i}^{j-1} U_{ik}x_k & j \geq i \end{cases} \tag{A.74}$$

e) Steps (b) – (d) are used in a recursive fashion to update the covariance matrix $\boldsymbol{P}$ during RLS implementations.

A.III Application of Bierman's Factorization Method on Different RLS Algorithms

For the various RLS algorithms, modifications are made to improve the tracking performance of the various algorithms, which in turn resulted in slight differences in the equation of covariance update than that shown in Eqn. (A.1). In this section, the results presented in the previous section will be modified accordingly to cater to the various versions of RLS algorithms used in this work. The modifications for the VFF-RLS algorithm will first be discussed, followed by discussions on the EWRLS and the EDF algorithms.

For the VFF-RLS algorithm, the equation of covariance update takes on the following form, following the conventions as defined in Eqn. (A.1):

$$\hat{\boldsymbol{P}} = \begin{cases} \frac{1}{\lambda}\left(\boldsymbol{P} - \frac{\boldsymbol{P}\boldsymbol{\psi}\boldsymbol{\psi}^T\boldsymbol{P}}{1+\boldsymbol{\psi}^T\boldsymbol{P}\boldsymbol{\psi}} \right) & \text{if trace of } \frac{1}{\lambda}\left(\boldsymbol{P} - \frac{\boldsymbol{P}\boldsymbol{\psi}\boldsymbol{\psi}^T\boldsymbol{P}}{1+\boldsymbol{\psi}^T\boldsymbol{P}\boldsymbol{\psi}} \right) \le C \\ \boldsymbol{P} - \frac{\boldsymbol{P}\boldsymbol{\psi}\boldsymbol{\psi}^T\boldsymbol{P}}{1+\boldsymbol{\psi}^T\boldsymbol{P}\boldsymbol{\psi}} & \text{otherwise} \end{cases} \tag{A.75}$$

Using the same procedure as shown in Eqns. (A.1) – (A.6), Eqn. (A.75) becomes:

$$\hat{\boldsymbol{U}}\hat{\boldsymbol{D}}\hat{\boldsymbol{U}}^T = \begin{cases} \frac{1}{\lambda}\left[\boldsymbol{U}\left(\boldsymbol{D} - \frac{\boldsymbol{x}\boldsymbol{x}^T}{\varphi} \right)\boldsymbol{U}^T \right] & \text{if trace of } \frac{1}{\lambda}\left[\boldsymbol{U}\left(\boldsymbol{D} - \frac{\boldsymbol{x}\boldsymbol{x}^T}{\varphi} \right)\boldsymbol{U}^T \right] \le C \\ \boldsymbol{U}\left(\boldsymbol{D} - \frac{\boldsymbol{x}\boldsymbol{x}^T}{\varphi} \right)\boldsymbol{U}^T & \text{otherwise} \end{cases} \tag{A.76}$$

where $r = 1$ in $\varphi = r + \boldsymbol{\psi}^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T\boldsymbol{\psi}$. Following this, it is obvious that Eqn. (A.76) can be written as:

$$\hat{\boldsymbol{U}}\hat{\boldsymbol{D}}\hat{\boldsymbol{U}}^T = \begin{cases} \left(\boldsymbol{U}\bar{\boldsymbol{U}}\right)\frac{\bar{\boldsymbol{D}}}{\lambda}\left(\boldsymbol{U}\bar{\boldsymbol{U}}\right)^T & \text{if trace of } \left(\boldsymbol{U}\bar{\boldsymbol{U}}\right)\frac{\bar{\boldsymbol{D}}}{\lambda}\left(\boldsymbol{U}\bar{\boldsymbol{U}}\right)^T \le C \\ \left(\boldsymbol{U}\bar{\boldsymbol{U}}\right)\bar{\boldsymbol{D}}\left(\boldsymbol{U}\bar{\boldsymbol{U}}\right)^T & \text{otherwise} \end{cases} \tag{A.77}$$

With these, the modified Eqns. (A.73) and (A.74) for the VFF-RLS algorithm are:

For i = 1 to n and j = i to n:

$$\hat{D}_{\mathrm{i}} = \begin{cases} \dfrac{D_{\mathrm{i}}}{\lambda}\left(\dfrac{1+\sum_{\mathrm{q}=1}^{\mathrm{i}-1} h_{\mathrm{q}} x_{\mathrm{q}}}{1+\sum_{\mathrm{q}=1}^{\mathrm{i}} h_{\mathrm{q}} x_{\mathrm{q}}}\right) & \text{if trace of } \left(\boldsymbol{U}\bar{\boldsymbol{U}}\right)\dfrac{\bar{\boldsymbol{D}}}{\lambda}\left(\boldsymbol{U}\bar{\boldsymbol{U}}\right)^{T} \le C \\ D_{\mathrm{i}}\left(\dfrac{1+\sum_{\mathrm{q}=1}^{\mathrm{i}-1} h_{\mathrm{q}} x_{\mathrm{q}}}{1+\sum_{\mathrm{q}=1}^{\mathrm{i}} h_{\mathrm{q}} x_{\mathrm{q}}}\right) & \text{otherwise} \end{cases} \tag{A.78}$$

$$\hat{U}_{\mathrm{ij}} = \begin{cases} 1 & \mathrm{j} = \mathrm{i} \\ U_{\mathrm{ij}} - \dfrac{h_{\mathrm{j}}}{1+\sum_{\mathrm{q}=1}^{\mathrm{j}-1} h_{\mathrm{q}} x_{\mathrm{q}}} \sum_{\mathrm{k}=\mathrm{i}}^{\mathrm{j}-1} U_{\mathrm{ik}} x_{\mathrm{k}} & \mathrm{j} \ge \mathrm{i} \end{cases}$$

(A.79)

As for the EWRLS algorithm, it can be shown in the same manner as above that the corresponding Bierman's equations are:

For i = 1 to n and j = i to n:

$$\hat{D}_{\mathrm{i}} = \frac{D_{\mathrm{i}}}{\lambda}\left(\frac{\lambda+\sum_{\mathrm{q}=1}^{\mathrm{i}-1} h_{\mathrm{q}} x_{\mathrm{q}}}{\lambda+\sum_{\mathrm{q}=1}^{\mathrm{i}} h_{\mathrm{q}} x_{\mathrm{q}}}\right) \tag{A.80}$$

$$\hat{U}_{\mathrm{ij}} = \begin{cases} 1 & \mathrm{j} = \mathrm{i} \\ U_{\mathrm{ij}} - \dfrac{h_{\mathrm{j}}}{\lambda+\sum_{\mathrm{q}=1}^{\mathrm{j}-1} h_{\mathrm{q}} x_{\mathrm{q}}} \sum_{\mathrm{k}=\mathrm{i}}^{\mathrm{j}-1} U_{\mathrm{ik}} x_{\mathrm{k}} & \mathrm{j} \ge \mathrm{i} \end{cases} \tag{A.81}$$

where the original equation of covariance update is:

$$\hat{\boldsymbol{P}} = \frac{1}{\lambda}\left(\boldsymbol{P} - \frac{\boldsymbol{P}\boldsymbol{\psi}\boldsymbol{\psi}^{T}\boldsymbol{P}}{1+\boldsymbol{\psi}^{T}\boldsymbol{P}\boldsymbol{\psi}}\right) \tag{A.82}$$

Similarly, for the EDF algorithm, the following results are applicable:

For i = 1 to n and j = i to n:

$$\tilde{D}_{\mathrm{i}} = D_{\mathrm{i}} \left(\frac{\beta^{-1} + \sum_{\mathrm{q}=1}^{\mathrm{i}-1} h_{\mathrm{q}} x_{\mathrm{q}}}{\beta^{-1} + \sum_{\mathrm{q}=1}^{\mathrm{i}} h_{\mathrm{q}} x_{\mathrm{q}}} \right) \tag{A.83}$$

$$\tilde{U}_{\mathrm{ij}} = \begin{cases} 1 & \mathrm{j} = \mathrm{i} \\ U_{\mathrm{ij}} - \dfrac{h_{\mathrm{j}}}{\beta^{-1} + \sum_{\mathrm{q}=1}^{\mathrm{j}-1} h_{\mathrm{q}} x_{\mathrm{q}}} \sum_{\mathrm{k}=\mathrm{i}}^{\mathrm{j}-1} U_{\mathrm{ik}} x_{\mathrm{k}} & \mathrm{j} \geq \mathrm{i} \end{cases} \tag{A.84}$$

where:

$$\hat{\boldsymbol{U}}\hat{\boldsymbol{D}}\hat{\boldsymbol{U}}^{T} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{D}}\tilde{\boldsymbol{U}}^{T} + \delta \boldsymbol{I} \tag{A.85}$$

and the original equation of covariance update is:

$$\hat{\boldsymbol{P}} = \boldsymbol{P} - \frac{\boldsymbol{P}\boldsymbol{\psi}\boldsymbol{\psi}^{T}\boldsymbol{P}}{\beta^{-1} + r} + \delta \boldsymbol{I} \tag{A.86}$$

To wind up the discussion as pertaining to Bierman's $\boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{T}$ factorization method, suffice to note here that Eqns. (A.78) – (A.89), (A.80) – (A.81), and (A.83) – (A.85) are the final versions of equations used in this work for different RLS algorithms. The coding of the equations is dependent on the preference of the programmers and is omitted here.

APPENDIX B: AN EXAMPLE ON THE CONSTRUCTION OF C_A, H_A, C_b, and H_b MATRICES IN THE GPC PREDICTION EQUATION

For a system with α = β = 2, discrete dead time D = 4, and $\boldsymbol{T}(z^{-1}) = \boldsymbol{I}$, the Two Inputs Two Outputs (TITO) CARIMA model is expressed as:

$$\left[\boldsymbol{I} + \boldsymbol{a}_1 z^{-1} + \boldsymbol{a}_2 z^{-2}\right]\boldsymbol{y}_k = \left[\boldsymbol{b}_1 z^{-1} + \boldsymbol{b}_2 z^{-2}\right]\boldsymbol{u}_{k-4} + \frac{\boldsymbol{I}}{\Delta}\boldsymbol{v}_k \tag{B.1}$$

where the process model parameters $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, $\boldsymbol{b}_1$, and $\boldsymbol{b}_2$ are defined as:

$$\boldsymbol{a}_1 = \begin{bmatrix} a_{111} & a_{112} \\ a_{121} & a_{122} \end{bmatrix};\ \boldsymbol{a}_2 = \begin{bmatrix} a_{211} & a_{212} \\ a_{221} & a_{222} \end{bmatrix};\ \boldsymbol{b}_1 = \begin{bmatrix} b_{111} & b_{112} \\ b_{121} & b_{122} \end{bmatrix};\boldsymbol{b}_2 = \begin{bmatrix} b_{211} & b_{212} \\ b_{221} & b_{222} \end{bmatrix} \tag{B.2}$$

Rearranging Eqn. (B.1) in the form of Eqn. (27) gives the following:

$$\left[\boldsymbol{I} + \left(\boldsymbol{a}_1 - I\right)z^{-1} + (\boldsymbol{a}_2 - \boldsymbol{a}_1)z^{-2} + \left(-\boldsymbol{a}_3 z^{-3}\right)\right]\boldsymbol{y}_k = \left[\boldsymbol{b}_1 z^{-1} + \boldsymbol{b}_2 z^{-2}\right]z^{-4}\Delta\boldsymbol{u}_k + \boldsymbol{v}_k \tag{B.3}$$

For the sake of the following discussion, Eqn. (B.3) is rewritten as:

$$\boldsymbol{y}_k + \boldsymbol{A}_1\boldsymbol{y}_{k-1} + \boldsymbol{A}_2\boldsymbol{y}_{k-2} + \boldsymbol{A}_3\boldsymbol{y}_{k-3} = \boldsymbol{b}_1\Delta\boldsymbol{u}_{k-5} + \boldsymbol{b}_2\Delta\boldsymbol{u}_{k-6} + \boldsymbol{v}_k \tag{B.4}$$

where:

$$\boldsymbol{A}_1 = \boldsymbol{a}_1 - \boldsymbol{I} = \begin{bmatrix} A_{111} & A_{112} \\ A_{121} & A_{122} \end{bmatrix};\ \boldsymbol{A}_2 = \boldsymbol{a}_2 - \boldsymbol{a}_1 = \begin{bmatrix} A_{211} & A_{212} \\ A_{221} & A_{222} \end{bmatrix};\ \boldsymbol{A}_3 = -\boldsymbol{a}_3 = \begin{bmatrix} A_{311} & A_{312} \\ A_{321} & A_{322} \end{bmatrix} \tag{B.5}$$

Equation (B.4) in its explicit TITO form is given as:

$$\begin{aligned} &\begin{bmatrix} y_{1,k} \\ y_{2,k} \end{bmatrix} + \begin{bmatrix} A_{111} & A_{112} \\ A_{121} & A_{122} \end{bmatrix}\begin{bmatrix} y_{1,k-1} \\ y_{2,k-1} \end{bmatrix} + \begin{bmatrix} A_{211} & A_{212} \\ A_{221} & A_{222} \end{bmatrix}\begin{bmatrix} y_{1,k-2} \\ y_{2,k-2} \end{bmatrix} + \begin{bmatrix} A_{311} & A_{312} \\ A_{321} & A_{322} \end{bmatrix}\begin{bmatrix} y_{1,k-3} \\ y_{2,k-3} \end{bmatrix} \\ &= \begin{bmatrix} b_{111} & b_{112} \\ b_{121} & b_{122} \end{bmatrix}\begin{bmatrix} \Delta u_{1,k-5} \\ \Delta u_{2,k-5} \end{bmatrix} + \begin{bmatrix} b_{211} & b_{212} \\ b_{221} & b_{222} \end{bmatrix}\begin{bmatrix} \Delta u_{1,k-6} \\ \Delta u_{2,k-6} \end{bmatrix} + \begin{bmatrix} v_{1,k} \\ v_{2,k} \end{bmatrix} \end{aligned} \tag{B.6}$$

Since white noise $\boldsymbol{v}(k) = 0$ for future values, and choosing the value of maximum prediction horizon $N_2 = 7$, Eqn. (B.6) can be written for multiple instances and assembled in the following form:

$$\boldsymbol{C_A}\begin{bmatrix} y_{1,k+1} \\ y_{2,k+1} \\ y_{1,k+2} \\ y_{2,k+2} \\ y_{1,k+3} \\ y_{2,k+3} \\ y_{1,k+4} \\ y_{2,k+4} \\ y_{1,k+5} \\ y_{2,k+5} \\ y_{1,k+6} \\ y_{2,k+6} \\ y_{1,k+7} \\ y_{2,k+7} \end{bmatrix} + \boldsymbol{H_A}\begin{bmatrix} y_{1,k} \\ y_{2,k} \\ y_{1,k-1} \\ y_{2,k-1} \\ y_{1,k-2} \\ y_{2,k-2} \end{bmatrix} = \boldsymbol{C_b}\begin{bmatrix} \Delta u_{1,k} \\ \Delta u_{2,k} \\ \Delta u_{1,k+1} \\ \Delta u_{2,k+1} \\ \Delta u_{1,k+2} \\ \Delta u_{2,k+2} \end{bmatrix} + \boldsymbol{H_b}\begin{bmatrix} \Delta u_{1,k-1} \\ \Delta u_{2,k-1} \\ \Delta u_{1,k-2} \\ \Delta u_{2,k-2} \\ \Delta u_{1,k-3} \\ \Delta u_{2,k-3} \\ \Delta u_{1,k-4} \\ \Delta u_{2,k-4} \\ \Delta u_{1,k-5} \\ \Delta u_{2,k-5} \end{bmatrix} \tag{B.7}$$

where the $\boldsymbol{C_A}$, $\boldsymbol{H_A}$, $\boldsymbol{C_b}$, and $\boldsymbol{H_b}$ matrices are:

$$\boldsymbol{C_A} = \begin{bmatrix} 1 & 0 & & & & & & & & & & & \underline{\underline{\mathbf{0}}} \\ 0 & 1 & & & & & & & & & & & \\ A_{111} & A_{112} & 1 & 0 & & & & & & & & & \\ A_{121} & A_{122} & 0 & 1 & & & & & & & & & \\ A_{211} & A_{212} & A_{111} & A_{112} & 1 & 0 & & & & & & & \\ A_{221} & A_{222} & A_{121} & A_{122} & 0 & 1 & & & & & & & \\ A_{311} & A_{312} & A_{211} & A_{212} & A_{111} & A_{112} & 1 & 0 & & & & & \\ A_{321} & A_{322} & A_{221} & A_{222} & A_{121} & A_{122} & 0 & 1 & & & & & \\ & & A_{311} & A_{312} & A_{211} & A_{212} & A_{111} & A_{112} & 1 & 0 & & & \\ & & A_{321} & A_{322} & A_{221} & A_{222} & A_{121} & A_{122} & 0 & 1 & & & \\ & & & & A_{311} & A_{312} & A_{211} & A_{212} & A_{111} & A_{112} & 1 & 0 & \\ & & & & A_{321} & A_{322} & A_{221} & A_{222} & A_{121} & A_{122} & 0 & 1 & \\ \underline{\underline{\mathbf{0}}} & & & & & & A_{311} & A_{312} & A_{211} & A_{212} & A_{111} & A_{112} & 1 & 0 \\ & & & & & & A_{321} & A_{322} & A_{221} & A_{222} & A_{121} & A_{122} & 0 & 1 \end{bmatrix};$$

$$\boldsymbol{H}_{\boldsymbol{A}} = \begin{bmatrix} A_{111} & A_{112} & A_{211} & A_{212} & A_{311} & A_{312} \\ A_{121} & A_{122} & A_{221} & A_{222} & A_{321} & A_{322} \\ A_{211} & A_{212} & A_{311} & A_{312} & & \\ A_{221} & A_{222} & A_{321} & A_{322} & & \\ A_{311} & A_{312} & & & & \\ A_{321} & A_{322} & & & & \\ 0 & & & & & \\ 0 & & & & & \\ 0 & & & & & \\ 0 & & & \underline{\underline{\mathbf{0}}} & & \\ 0 & & & & & \\ 0 & & & & & \\ 0 & & & & & \\ 0 & & & & & \end{bmatrix}; \quad \boldsymbol{C}_{\boldsymbol{b}} = \begin{bmatrix} & & & & & 0 \\ & & & & & 0 \\ & & & & & 0 \\ & & & \underline{\underline{\mathbf{0}}} & & 0 \\ & & & & & 0 \\ & & & & & 0 \\ & & & & & 0 \\ & & & & & 0 \\ b_{111} & b_{112} & & & & 0 \\ b_{121} & b_{122} & & & & 0 \\ b_{211} & b_{212} & b_{111} & b_{112} & & 0 \\ b_{221} & b_{222} & b_{121} & b_{122} & & 0 \\ & & b_{211} & b_{212} & b_{111} & b_{112} \\ \underline{\underline{\mathbf{0}}} & & b_{221} & b_{222} & b_{121} & b_{122} \end{bmatrix};$$

$$\boldsymbol{H}_{\boldsymbol{b}} = \begin{bmatrix} \underline{\underline{\mathbf{0}}} & & & & & & b_{111} & b_{112} & b_{211} & b_{212} \\ & & & & & & b_{121} & b_{122} & b_{221} & b_{222} \\ & & & & b_{111} & b_{112} & b_{211} & b_{212} & & 0 \\ & & & & b_{121} & b_{122} & b_{221} & b_{222} & & 0 \\ & & b_{111} & b_{112} & b_{211} & b_{212} & & & & 0 \\ & & b_{121} & b_{122} & b_{221} & b_{222} & & & & 0 \\ b_{111} & b_{112} & b_{211} & b_{212} & & & & & & 0 \\ b_{121} & b_{122} & b_{221} & b_{222} & & & & & & 0 \\ b_{211} & b_{212} & & & & & & & & 0 \\ b_{221} & b_{222} & & & & & & & & 0 \\ & & & & & \underline{\underline{\mathbf{0}}} & & & & 0 \\ & & & & & & & & & 0 \\ & & & & & & & & & 0 \\ & & & & & & & & & 0 \end{bmatrix} \tag{B.8}$$

Hence, for any system, the $\boldsymbol{C}_{\boldsymbol{A}}$, $\boldsymbol{H}_{\boldsymbol{A}}$, $\boldsymbol{C}_{\boldsymbol{b}}$, and $\boldsymbol{H}_{\boldsymbol{b}}$ matrices can be obtained in the same manner as shown above. Note that at this juncture, for a given model order, dead time and the size of the process, N_2 is the only effective controller tuning parameter (*i.e.* among N_1, M, *etc.*) which determines the size of the $\boldsymbol{C}_{\boldsymbol{A}}$, $\boldsymbol{H}_{\boldsymbol{A}}$, $\boldsymbol{C}_{\boldsymbol{b}}$, and $\boldsymbol{H}_{\boldsymbol{b}}$ matrices. The roles of the remaining tuning parameters are only seen in the formulation procedures of the GPC cost function, *i.e.* Eqns. (2.30) – (2.31).

APPENDIX C: MATLAB S-FUNCTION FOR THE VFF-RLS SCHEME WITH BIERMAN'S FACTORIZATION

```
function [sys,x0,str,ts]=fortesque_udu(t,x,u,flag,tsamp,m,n,poc,kov,var,..
C,lmin)

%Main Symbols Used
%========================================================================
%tsamp - sampling time
%m - number of inputs
%n - number of outputs
%theta - estimated model parameters
%p - covariance matrix
%y - outputs
%z - regressor vector
%e - prediction error
%K - kalman gain
%l - forgetting factor
%poc - initial guess for theta
%kov - initial values for diagonal of covariance matrix
%var - design parameter
%C - design parameter
%lmin - minimum forgetting factor
%========================================================================

switch flag

    case 0

        sizes=simsizes;

        sizes.NumContStates = 0;
        sizes.NumDiscStates = m*n+n*n+m+n+n+1;
        sizes.NumOutputs = m*n+n*n+m+n+n+1;
        sizes.NumInputs = n+m;
        sizes.DirFeedthrough = 0;
        sizes.NumSampleTimes = 1;

        sys=simsizes(sizes);

        %Making sure the parameter inputs are correct
        if length(poc) ~= n*m
            disp('*VFF-RLS warning: Incorrect number of parameters !!!');
            disp('Changed to:');
            poc = 0.1 : 0.1 : n*m*0.1;
        end

        %Defining the initial covariance matrix
        if length(kov) == 1
            p = eye(n,n)*kov;
        elseif length(kov) == n*n
            p = zeros(n,n);
            p(:) = kov;
        else
            disp('*VFF-RLS warning: Incorrect covariance matrix !!!');
            disp('Changed to:');
            p = eye(n)*10e6;
        end
```

```
    str = [];
    ts = [tsamp 0];

    %Initial values of the state variables
    x0  = [poc,p(:)',zeros(1,m),zeros(1,n),zeros(1,n),zeros(1,1)]';

case 2

    %Zeroing the covariance matrix and theta
    p = zeros(n,n);
    theta = zeros(n,m);

    %Defining theta, y, p and z
    theta(:) = x(1:m*n);
    p(:) = x(m*n+1:m*n+n*n);

    y = u(1:m);
    z = u(m+1:m+n);

    %Calculation of e, K and l
    e = y-theta'*z;
    K = p*z/(1+z'*p*z);
    error = e'*e;
    l = 1 - ((error)/(var*(1 + z'*p*z)));

    %Ensure a lower bound on the forgetting factor
    if l < lmin
        l = lmin;
    end

    %UDU' factorization
    p1 = p;
    for j = n : -1 : 2
        D(j,j) = p1(j,j);
        alpha1 = 1/D(j,j);
        for k = 1 : 1 : j - 1
            beta = p1(k,j);
            U(k,j) = alpha1*beta;
            for i = 1 : 1 : k
                p1(i,k) = p1(i,k) - beta*U(i,j);
            end
        end
    end
    D(1,1) = p1(1,1);
    for i = 1 : 1 : n
        U(i,i) = 1;
    end
    u = U;
    d = D;

    %Bierman's UDU' updates
    f = u'*z;
    v = d*f;
    r = 1;
```

```
        alpha = 0;
        for i = 1:n
            if i == 1
                dummy = r;
            else
                dummy = alpha(i-1);
            end
            alpha(i) = dummy + v(i)*f(i);
        end

        for i = 1 : n
            if i == 1
                dummy1 = r;
            else
                dummy1 = alpha(i-1);
            end
            M(i,i) = (d(i,i)*(dummy1/alpha(i)));
            k = i;
            sum = 0;
            for j = i + 1 : n
                sum = sum + u(i,k)*v(k);
                M(i,j) = u(i,j) - f(j)*sum/(alpha(j-1));
                k = k + 1;
            end
        end

        U1 = triu(M,1)+eye(length(M));
        D1 = diag(diag(M));

        %Ensure upper bound of trace of P
        w = U1*D1*U1';

        if trace(w/l) <= C
            p = w/l;
        elseif trace(w/l) > C
            p = w;
        end

        %Updating theta
        theta = theta + K*e';

        sys = [theta(:);p(:);e;trace(p);l;error;];

    case 3
        sys=x;
    case {1 4 9}
        sys = [];
    otherwise
        error(['Unhandled flag = ',num2str(flag)]);
end
```

APPENDIX D: MATLAB S-FUNCTION FOR THE UNCONSTRAINED AS-GPC SCHEME

```
function [sys,x0,str,ts]=gpc_sfcn(t,x,u,flag,...
tsamp,den_order,num_order,dead_time,tstart,n,n_inputs,P,N,M,L,W,y_min,...
y_max,slew_min,slew_max,u_min,u_max,tune_start_c)

%Main Symbols Used
%=========================================================================
%tsamp - sampling time
%den_order - model denominator order
%num_order - model numerator order
%dead_time - dead time of model
%tstart - time at which model adaptation is turned on
%n - number of outputs (y)
%n_inputs - number of inputs (u)
%P - maximum prediction horizon
%N - minimum prediction horizon
%M - control horizon
%L - move suppression weight
%W - output weight
%y_min - minimum bound for y
%y_max - maximum bound for y
%slew_min - minimum bound for slew rate delta_u
%slew_max - maximum bound for slew rate delta_u
%u_min - minimum bound for u
%u_max - maximum bound for u
%tune_start_c - time at which self-tuning is turned on
%theta - model parameters received from the RLS
%tau - FOPDT process time constant
%K - process gain
%ympast - vector of past values of y
%umpast - vector of past values of delta_u
%uppast1 - vector of past values of u
%um_cpast - past value of u
%uu - first control move from the optimal solution sequence
%=========================================================================

switch flag

    case 0

        sizes=simsizes;

        sizes.NumContStates = 0;

        sizes.NumDiscStates = (den_order + 1)*n+n_inputs+n_inputs+...
                              (num_order + dead_time - 1)*n_inputs+...
                               n_inputs+n_inputs+1+n*n*den_order+...
                               n*n_inputs*num_order+n*n+6;

        sizes.NumOutputs = (den_order + 1)*n+n_inputs+n_inputs+...
                           (num_order + dead_time - 1)*n_inputs+...
                            n_inputs+n_inputs+1+n*n*den_order+...
                            n*n_inputs*num_order+n*n+6;

        sizes.NumInputs = n+n+n*n*den_order+n*n_inputs*num_order+n*n;
```

```
    sizes.DirFeedthrough = 1;

    sizes.NumSampleTimes = 1;

    sys = simsizes(sizes);

    str = [];
    ts = [tsamp 0];

    %Initial valus of the state vector
    x0 = [zeros((den_order + 1)*n,1);zeros(n_inputs,1);...
          zeros(n_inputs,1);...
          zeros((num_order + dead_time - 1)*n_inputs,1);...
          zeros(n_inputs,1);zeros(1,1);zeros(n_inputs,1);...
          zeros(n*n*den_order+n*n_inputs*num_order+n*n,1);zeros(6,1)];

case 2

        %Zeroing the variables
        ympast = zeros((den_order + 1)*n,1);
        uppast1 = zeros(n_inputs,1);
        um_cpast = zeros(n_inputs,1);
        umpast = zeros((num_order + dead_time - 1)*n_inputs,1);
        uu = zeros(n_inputs,1);

        %Extracting the previously saved state variables
        ympast(:)= x(1:(den_order + 1)*n);
        uppast1(:) = x((den_order + 1)*n+1:(den_order + 1)*n+n_inputs);
        um_cpast(:) = x((den_order + 1)*n+...
                        n_inputs+1:(den_order + 1)*n+...
                        n_inputs+n_inputs);
        umpast(:) = x((den_order + 1)*n+n_inputs+...
                       n_inputs+1:(den_order + 1)*n+...
                       n_inputs+n_inputs+...
                       (num_order + dead_time - 1)*n_inputs);
        uu(:) = x((den_order + 1 )*n+n_inputs+n_inputs+...
                  (num_order + dead_time - 1)*n_inputs+1:...
                  (den_order + 1)*n+n_inputs+n_inputs+...
                  (num_order + dead_time - 1)*n_inputs+n_inputs);

        %Updating ympast
        y = u(1:n);
        ympast = [y; ympast(1:length(ympast)-length(y))];

        %Switching between adaptive and backup controller
        if t > tstart
            theta = u(n+n+1:n+n+n*n*den_order+...
                      n*n_inputs*num_order+n*n);
            if theta(1) > 0 || theta(1) < -1
                theta = [-0.89758 -0.17703 0]';
            end
            %Calculation of move suppression weight
            L = L*((100-0)/(1.5-0.5))^2;
            if t > tune_start_c
                if theta(1) < 0 && theta(1) > -1
```

```
            tau = 1/(log(-theta(1))/-tsamp);
            if isreal(tau) == 1 && tau > 0
                f = (M/500)*((3.5*tau/tsamp)+2-((M-1)/2));
                K = theta(2)/(1+theta(1));
                L = f*K^2;
            end
        end
    end
else
    theta = [-0.89758 -0.17703 0]';
    L = L*((100-0)/(1.5-0.5))^2;
end

%Model Parameters
a111 = theta(1);
a112 = 0;
a121 = 0;
a122 = 0;

a211 = 0;
a212 = 0;
a221 = 0;
a222 = 0;

b111 = theta(2);
b112 = 0;
b113 = 0;
b121 = 0;
b122 = 0;
b123 = 0;

b211 = 0;
b212 = 0;
b213 = 0;
b221 = 0;
b222 = 0;
b223 = 0;

%Adding an integrator to the model
A0 = [1 0;0 1;];
A1 = [a111 - 1 a112;a121 a122 - 1;];
A2 = [a211 - a111 a212 - a112;a221 - a121 a222 - a122;];
A3 = [-a211 -a212;-a221 -a222;];
B1 = [b111 b112 b113;b121 b122 b123;];
B2 = [b211 b212 b213;b221 b222 b223;];

%Output Model Parameters (Matrix A)
A(1,1,1) = A0(1,1);
A(1,1,2) = A0(1,2);
A(1,2,1) = A0(2,1);
A(1,2,2) = A0(2,2);

A(2,1,1) = A1(1,1);
A(2,1,2) = A1(1,2);
A(2,2,1) = A1(2,1);
```

```
A(2,2,2) = A1(2,2);

A(3,1,1) = A2(1,1);
A(3,1,2) = A2(1,2);
A(3,2,1) = A2(2,1);
A(3,2,2) = A2(2,2);

A(4,1,1) = A3(1,1);
A(4,1,2) = A3(1,2);
A(4,2,1) = A3(2,1);
A(4,2,2) = A3(2,2);

%Input Model Parameters (Matrix B)
B(1,1,1) = B1(1,1);
B(1,1,2) = B1(1,2);
B(1,1,3) = B1(1,3);
B(1,2,1) = B1(2,1);
B(1,2,2) = B1(2,2);
B(1,2,3) = B1(2,3);

B(2,1,1) = B2(1,1);
B(2,1,2) = B2(1,2);
B(2,1,3) = B2(1,3);
B(2,2,1) = B2(2,1);
B(2,2,2) = B2(2,2);
B(2,2,3) = B2(2,3);

%Intermediate Information
dummy_nA = den_order + 1;

%CA and Czb Calculations
for i = 1 : P
    count = 0;
    for j =i : min(i + den_order + 1,P)
        count = count + 1;
        for col = n*i - n + 1
            for row =n*j - n + 1
                for ii = 1 : n
                    for jj = 1 : n
                        CA(row + ii - 1,col + jj - 1) = ...
                                                A(count,ii,jj);
                    end
                end
            end
        end
    end
    count = 0;
    for j= i : min(i + num_order - 1,P)
        count = count + 1;
        for col = n_inputs*i - n_inputs + 1
            for row = n*j - n + 1
                for ii = 1 : n
                    for jj = 1 : n_inputs
                        Czb(row + ii - 1,col + jj - 1) = ...
                                                B(count,ii,jj);
```

```
                    end
                end
            end
        end
    end
end

Czb = Czb(:,dead_time*n_inputs+1:end);

%HA Calculations
HA = zeros(n*P,n*dummy_nA);
for i = 1 : P
    count = i + 1;
    for j= 1 : dummy_nA
        for col = n*j - n + 1
            for row = n*i - n + 1
                for ii = 1 : n
                    for jj = 1 : n
                        HA(row + ii - 1,col + jj - 1) = ...
                                                A(count,ii,jj);
                    end
                end
            end
        end
        count = count + 1;
    end
    dummy_nA = dummy_nA - 1;
end

%Hzb Calculations
Hzb = zeros(n*P,n_inputs*(num_order - 1));
nn = num_order;
for i = 1 : nn - 1
    count = i + 1;
    for j = 1 : P
        if count <= nn
            for col = n_inputs*i - n_inputs + 1
                for row = n*j - n + 1
                    for ii = 1 : n
                        for jj = 1 : n_inputs
                            Hzb(row + ii - 1,col + jj - 1) =...
                                                B(count,ii,jj);
                        end
                    end
                end
            end
        end
        count = count + 1;
    end
end

%Hzb_plus Calculations
Hzb_plus = zeros(n*P,-n_inputs + n_inputs*dead_time +...
              n_inputs*num_order - ((num_order - 1)*n_inputs));
for bb = 1 : num_order
```

```
    for k = 1 : dead_time
        col = 1 + n_inputs*(dead_time - k);
        row = (1 - 2*n) + n*(bb + k);
        for ii = 1 : n
            for jj = 1 : n_inputs
                Hzb_plus(row + ii - 1,col + jj - 1) =...
                                                    B(bb,ii,jj);
            end
        end
    end
end
Hzb = [Hzb_plus Hzb];

%Computing the matrices for prediction equation
AA = CA\Czb;
for col = 1 : n_inputs*M
    for row = 1 : n*P
        H(row,col) = AA(row,col);
    end
end

H = H(n*N - n + 1:end,:);
PP = CA\Hzb;
PP = PP(n*N - n + 1:end,:);
Q = -CA\HA;
Q = Q(n*N - n + 1:end,:);

%Calculating unit vector for separating the first input
e = zeros(n_inputs,n_inputs*M);
for i = 1 : n_inputs
    for j = i
        e(i,j) = 1;
    end
end

%Define the reference signal
rr = [u(2)]';

for j = 1 : n : n*(P - N) + 1
    count = 1;
    for ii = 1 : n
        r(j + ii - 1,1) = rr(ii);
        count = count + 1;
    end
end

%Output weights for different output
for i = 1 : (P - N + 1)
    count = 1;
    for row = n*i - n + 1
        for col = n*i - n + 1
            for j = 1 : n
                Beta(row + j - 1,col + j - 1) = W(count);
                count = count + 1;
            end
```

```
        end
    end
end

%Move suppression weight for different inputs
for i = 1 : M
    count = 1;
    for row = n_inputs*i - n_inputs + 1
        for col = n_inputs*i - n_inputs + 1
            for j = 1 : n_inputs
                Lamda(row + j - 1,col + j - 1) = L(count);
                count = count + 1;
            end
        end
    end
end

%Hessian matrix HH and f
HH = round(1e14*(H'*Beta*H + Lamda))/1e14;
f = H'*Beta*(PP*umpast + Q*ympast - r);

%Calculations for the unconstrained case
PR = e*inv(HH)*H';
NK = e*inv(HH)*H'*Q;
DK = e*inv(HH)*H'*PP;

%Forming the closed loop characteristic equation to be used for
%calculation of poles later
DDK = [1 DK];
A_hat = [1 theta(1)];
B_hat = [zeros(1,dead_time+1) theta(2)];
Integrator = [1 -1];
multiply = polynomial_multiply(DDK,Integrator);
multiply1 = polynomial_multiply(multiply,A_hat);
multiply2 = polynomial_multiply(NK,B_hat);
total_multiply = multiply1 + multiply2;

%Turn off the constraints after 2500 s
if t <= 2500
    %Producing Toeplitz Matrix Identity for use in Linear
    %Inequalities
    D = eye(n_inputs,n_inputs);
    for i = 1 : M
        for j =i : M
            for col = n_inputs*i - n_inputs + 1
                for row =n_inputs*j - n_inputs + 1
                    for ii = 1 : n_inputs
                        for jj = 1 : n_inputs
                            C_delta(row + ii - 1,col + ...
                                                jj - 1) = D(ii,jj);
                        end
                    end
                end
            end
        end
```

```
end

%Producing LL[I]
DD = eye(n_inputs,n_inputs);
for i = 1
    for j = i : M;
        for col = n_inputs*i - n_inputs + 1
            for row =n_inputs*j - n_inputs + 1
                for ii = 1 : n_inputs
                    for jj = 1 : n_inputs
                            LL(row + ii - 1,col + ...
                                      jj - 1) = DD(ii,jj);
                    end
                end
            end
        end
    end
end

%Define the slew rate and input constraints over control
%horizon
for j = 1 : n_inputs : n_inputs*M - n_inputs + 1
    count = 1;
    for ii = 1 : n_inputs
        delta_u_min(j + ii - 1,1) = slew_min(ii);
        delta_u_max(j + ii - 1,1) = slew_max(ii);
        umin(j + ii - 1,1) = u_min(ii);
        umax(j + ii - 1,1) = u_max(ii);
        count = count + 1;
    end
end

%Define the output constraints over the prediction horizon
for j = 1 : n : n*(P - N) + 1
    count = 1;
    for ii = 1 : n
        ymin(j + ii - 1,1) = y_min(ii);
        ymax(j + ii - 1,1) = y_max(ii);
        count = count + 1;
    end
end

%Solving the QP problem by calling 'quadprog'
C = [eye(n_inputs*M,n_inputs*M);...
    -eye(n_inputs*M,n_inputs*M);H;-H;C_delta;-C_delta;];
b = [delta_u_max;-delta_u_min;...
    ymax - PP*umpast - Q*ympast;...
    -ymin + PP*umpast + Q*ympast;umax - LL*um_cpast;...
    -umin + LL*um_cpast;];
Options = optimset('LargeScale','off');
[delta_u,fval,exitflag,output] = ...
quadprog(HH,f,C,b,[],[],[],[],[],Options);

%Isolating the first delta_u
delta_u = e*delta_u;
```

```
        else
            delta_u = PR*r-NK*ympast-DK*umpast;
        end

        %Computing the control move u
        uu = delta_u + uppast1;

        %Updating the vectors of past moves
        um_cpast = uu;
        uppast1 = [uu; uppast1(1:length(uppast1)- length(uu))];
        umpast = [delta_u; umpast(1:length(umpast)- length(delta_u))];

        if num_order == 1 && dead_time == 0
            umpast = zeros((num_order + dead_time - 1)*n_inputs,1);
        end

        %Saving the state variables
        sys=[ympast(:);uppast1(:);um_cpast(:);umpast(:);uu(:);...
             P;L;theta(:);total_multiply(:);];

case 3

        y = u(1:n);

        %Zeroing the variables
        ympast = zeros((den_order + 1)*n,1);
        uppast1 = zeros(n_inputs,1);
        um_cpast = zeros(n_inputs,1);
        umpast = zeros((num_order + dead_time - 1)*n_inputs,1);
        uu = zeros(n_inputs,1);

        %Extracting the previously saved state variables
        ympast(:)= x(1:(den_order + 1)*n);
        uppast1(:) = x((den_order + 1)*n+1:(den_order + 1)*n+n_inputs);
        um_cpast(:) = x((den_order + 1)*n+...
                        n_inputs+1:(den_order + 1)*n+...
                        n_inputs+n_inputs);
        umpast(:) = x((den_order + 1)*n+n_inputs+...
                       n_inputs+1:(den_order + 1)*n+...
                       n_inputs+n_inputs+...
                       (num_order + dead_time - 1)*n_inputs);
        uu(:) = x((den_order + 1 )*n+n_inputs+n_inputs+...
                  (num_order + dead_time - 1)*n_inputs+1:...
                  (den_order + 1)*n+n_inputs+n_inputs+...
                  (num_order + dead_time - 1)*n_inputs+n_inputs);

        %Updating ympast
        ympast = [y; ympast(1:length(ympast)-length(y))];

        %Switching between adaptive and backup controller
        if t > tstart
            theta = u(n+n+1:n+n+n*n*den_order+...
                      n*n_inputs*num_order+n*n);
            if theta(1) > 0 || theta(1) < -1
                theta = [-0.89758 -0.17703 0]';
```

```
    end
    %Calculation of move suppression weight
    L = L*((100-0)/(1.5-0.5))^2;
    if t > tune_start_c
        if theta(1) < 0 && theta(1) > -1
            tau = 1/(log(-theta(1))/-tsamp);
            if isreal(tau) == 1 && tau > 0
                f = (M/500)*((3.5*tau/tsamp)+2-((M-1)/2));
                K = theta(2)/(1+theta(1));
                L = f*K^2;
            end
        end
    end
else
    theta = [-0.89758 -0.17703 0]';
    L = L*((100-0)/(1.5-0.5))^2;
end

%Model Parameters
a111 = theta(1);
a112 = 0;
a121 = 0;
a122 = 0;

a211 = 0;
a212 = 0;
a221 = 0;
a222 = 0;

b111 = theta(2);
b112 = 0;
b113 = 0;
b121 = 0;
b122 = 0;
b123 = 0;

b211 = 0;
b212 = 0;
b213 = 0;
b221 = 0;
b222 = 0;
b223 = 0;

%Adding an integrator to the model
A0 = [1 0;0 1;];
A1 = [a111 - 1 a112;a121 a122 - 1;];
A2 = [a211 - a111 a212 - a112;a221 - a121 a222 - a122;];
A3 = [-a211 -a212;-a221 -a222;];
B1 = [b111 b112 b113;b121 b122 b123;];
B2 = [b211 b212 b213;b221 b222 b223;];

%Output Model Parameters (Matrix A)
A(1,1,1) = A0(1,1);
A(1,1,2) = A0(1,2);
A(1,2,1) = A0(2,1);
```

```
A(1,2,2) = A0(2,2);

A(2,1,1) = A1(1,1);
A(2,1,2) = A1(1,2);
A(2,2,1) = A1(2,1);
A(2,2,2) = A1(2,2);

A(3,1,1) = A2(1,1);
A(3,1,2) = A2(1,2);
A(3,2,1) = A2(2,1);
A(3,2,2) = A2(2,2);

A(4,1,1) = A3(1,1);
A(4,1,2) = A3(1,2);
A(4,2,1) = A3(2,1);
A(4,2,2) = A3(2,2);

%Input Model Parameters (Matrix B)
B(1,1,1) = B1(1,1);
B(1,1,2) = B1(1,2);
B(1,1,3) = B1(1,3);
B(1,2,1) = B1(2,1);
B(1,2,2) = B1(2,2);
B(1,2,3) = B1(2,3);

B(2,1,1) = B2(1,1);
B(2,1,2) = B2(1,2);
B(2,1,3) = B2(1,3);
B(2,2,1) = B2(2,1);
B(2,2,2) = B2(2,2);
B(2,2,3) = B2(2,3);

%Intermediate Information
dummy_nA = den_order + 1;

%CA and Czb Calculations
for i = 1 : P
    count = 0;
    for j =i : min(i + den_order + 1,P)
        count = count + 1;
        for col = n*i - n + 1
            for row =n*j - n + 1
                for ii = 1 : n
                    for jj = 1 : n
                        CA(row + ii - 1,col + jj - 1) = ...
                                                A(count,ii,jj);
                    end
                end
            end
        end
    end
    count = 0;
    for j= i : min(i + num_order - 1,P)
        count = count + 1;
        for col = n_inputs*i - n_inputs + 1
```

```
            for row = n*j - n + 1
                for ii = 1 : n
                    for jj = 1 : n_inputs
                        Czb(row + ii - 1,col + jj - 1) = ...
                                                B(count,ii,jj);
                    end
                end
            end
        end
    end
end

Czb = Czb(:,dead_time*n_inputs+1:end);

%HA Calculations
HA = zeros(n*P,n*dummy_nA);
for i = 1 : P
    count = i + 1;
    for j= 1 : dummy_nA
        for col = n*j - n + 1
            for row = n*i - n + 1
                for ii = 1 : n
                    for jj = 1 : n
                        HA(row + ii - 1,col + jj - 1) = ...
                                                A(count,ii,jj);
                    end
                end
            end
        end
        count = count + 1;
    end
    dummy_nA = dummy_nA - 1;
end

%Hzb Calculations
Hzb = zeros(n*P,n_inputs*(num_order - 1));
nn = num_order;
for i = 1 : nn - 1
    count = i + 1;
    for j = 1 : P
        if count <= nn
            for col = n_inputs*i - n_inputs + 1
                for row = n*j - n + 1
                    for ii = 1 : n
                        for jj = 1 : n_inputs
                            Hzb(row + ii - 1,col + jj - 1) =...
                                                B(count,ii,jj);
                        end
                    end
                end
            end
        end
        count = count + 1;
    end
end
```

```
%Hzb_plus Calculations
Hzb_plus = zeros(n*P,-n_inputs + n_inputs*dead_time +...
              n_inputs*num_order - ((num_order - 1)*n_inputs));
for bb = 1 : num_order
    for k = 1 : dead_time
        col = 1 + n_inputs*(dead_time - k);
        row = (1 - 2*n) + n*(bb + k);
        for ii = 1 : n
            for jj = 1 : n_inputs
                Hzb_plus(row + ii - 1,col + jj - 1) =...
                                                    B(bb,ii,jj);
            end
        end
    end
end
Hzb = [Hzb_plus Hzb];

%Computing the matrices for prediction equation
AA = CA\Czb;
for col = 1 : n_inputs*M
    for row = 1 : n*P
        H(row,col) = AA(row,col);
    end
end

H = H(n*N - n + 1:end,:);
PP = CA\Hzb;
PP = PP(n*N - n + 1:end,:);
Q = -CA\HA;
Q = Q(n*N - n + 1:end,:);

%Calculating unit vector for separating the first input
e = zeros(n_inputs,n_inputs*M);
for i = 1 : n_inputs
    for j = i
        e(i,j) = 1;
    end
end

%Define the reference signal
rr = [u(2)]';

for j = 1 : n : n*(P - N) + 1
    count = 1;
    for ii = 1 : n
        r(j + ii - 1,1) = rr(ii);
        count = count + 1;
    end
end

%Output weights for different output
for i = 1 : (P - N + 1)
    count = 1;
    for row = n*i - n + 1
```

```
        for col = n*i - n + 1
            for j = 1 : n
                Beta(row + j - 1,col + j - 1) = W(count);
                count = count + 1;
            end
        end
    end
end

%Move suppression weight for different inputs
for i = 1 : M
    count = 1;
    for row = n_inputs*i - n_inputs + 1
        for col = n_inputs*i - n_inputs + 1
            for j = 1 : n_inputs
                Lamda(row + j - 1,col + j - 1) = L(count);
                count = count + 1;
            end
        end
    end
end

%Hessian matrix HH and f
HH = round(1e14*(H'*Beta*H + Lamda))/1e14;
f = H'*Beta*(PP*umpast + Q*ympast - r);

%Calculations for the unconstrained case
PR = e*inv(HH)*H';
NK = e*inv(HH)*H'*Q;
DK = e*inv(HH)*H'*PP;

%Turn off the constraints after 2500 s
if t <= 2500
    %Producing Toeplitz Matrix Identity for use in Linear
    %Inequalities
    D = eye(n_inputs,n_inputs);
    for i = 1 : M
        for j =i : M
            for col = n_inputs*i - n_inputs + 1
                for row =n_inputs*j - n_inputs + 1
                    for ii = 1 : n_inputs
                        for jj = 1 : n_inputs
                            C_delta(row + ii - 1,col + ...
                                                jj - 1) = D(ii,jj);
                        end
                    end
                end
            end
        end
    end

    %Producing LL[I]
    DD = eye(n_inputs,n_inputs);
    for i = 1
        for j = i : M;
```

```
                for col = n_inputs*i - n_inputs + 1
                    for row =n_inputs*j - n_inputs + 1
                        for ii = 1 : n_inputs
                            for jj = 1 : n_inputs
                                    LL(row + ii - 1,col + ...
                                                jj - 1) = DD(ii,jj);
                            end
                        end
                    end
                end
            end
        end

        %Define the slew rate and input constraints over control
        %horizon
        for j = 1 : n_inputs : n_inputs*M - n_inputs + 1
            count = 1;
            for ii = 1 : n_inputs
                delta_u_min(j + ii - 1,1) = slew_min(ii);
                delta_u_max(j + ii - 1,1) = slew_max(ii);
                umin(j + ii - 1,1) = u_min(ii);
                umax(j + ii - 1,1) = u_max(ii);
                count = count + 1;
            end
        end

        %Define the output constraints over the prediction horizon
        for j = 1 : n : n*(P - N) + 1
            count = 1;
            for ii = 1 : n
                ymin(j + ii - 1,1) = y_min(ii);
                ymax(j + ii - 1,1) = y_max(ii);
                count = count + 1;
            end
        end

        %Solving the QP problem by calling 'quadprog'
        C = [eye(n_inputs*M,n_inputs*M);...
            -eye(n_inputs*M,n_inputs*M);H;-H;C_delta;-C_delta;];
        b = [delta_u_max;-delta_u_min;...
            ymax - PP*umpast - Q*ympast;...
            -ymin + PP*umpast + Q*ympast;umax - LL*um_cpast;...
            -umin + LL*um_cpast;];
        Options = optimset('LargeScale','off');
        [delta_u,fval,exitflag,output] = ...
        quadprog(HH,f,C,b,[],[],[],[],[],Options);

        %Isolating the first delta_u
        delta_u = e*delta_u;
    else
        delta_u = PR*r-NK*ympast-DK*umpast;
    end

    %Computing the control move u
    uu = delta_u + uppast1;
```

```
            %Updating the vectors of past moves
            um_cpast = uu;
            uppast1 = [uu; uppast1(1:length(uppast1)- length(uu))];
            umpast = [delta_u; umpast(1:length(umpast)- length(delta_u))];

            if num_order == 1 && dead_time == 0
                umpast = zeros((num_order + dead_time - 1)*n_inputs,1);
            end

            %Sending the value of u for implementation to the process
            x((den_order + 1)*n+n_inputs+n_inputs+...
              (num_order + dead_time - 1)*n_inputs+...
               1:(den_order + 1)*n+n_inputs+n_inputs+...
              (num_order + dead_time - 1)*n_inputs+n_inputs) = uu(:);

        sys=x;

    case {1 4 9}

        sys=[];

    otherwise

        error(['Unhandled flag = ',num2str(flag)]);

end
```

APPENDIX E: MATLAB S-FUNCTION FOR THE CONSTRAINED AS-GPC SCHEME

```
function [sys,x0,str,ts]=gpc_sfcn(t,x,u,flag,...
tsamp,den_order,num_order,dead_time,tstart,n,n_inputs,P,N,M,L,W,y_min,...
y_max,slew_min,slew_max,u_min,u_max,tune_start_c)

%Main Symbols Used
%=========================================================================
%tsamp - sampling time
%den_order - model denominator order
%num_order - model numerator order
%dead_time - dead time of model
%tstart - time at which model adaptation is turned on
%n - number of outputs (y)
%n_inputs - number of inputs (u)
%P - maximum prediction horizon
%N - minimum prediction horizon
%M - control horizon
%L - move suppression weight
%W - output weight
%y_min - minimum bound for y
%y_max - maximum bound for y
%slew_min - minimum bound for slew rate delta_u
%slew_max - maximum bound for slew rate delta_u
%u_min - minimum bound for u
%u_max - maximum bound for u
%tune_start_c - time at which self-tuning is turned on
%theta - model parameters received from the RLS
%tau - FOPDT process time constant
%K - process gain
%ympast - vector of past values of y
%umpast - vector of past values of delta_u
%uppast1 - vector of past values of u
%um_cpast - past value of u
%uu - first control move from the optimal solution sequence
%=========================================================================

switch flag

    case 0

        sizes=simsizes;

        sizes.NumContStates = 0;

        sizes.NumDiscStates =(den_order + 1)*n+n_inputs+n_inputs+...
                             (num_order + dead_time - 1)*n_inputs+...
                              n_inputs+n_inputs+1+n*n*den_order+...
                              n*n_inputs*num_order+n*n;

        sizes.NumOutputs = (den_order + 1)*n+n_inputs+n_inputs+...
                           (num_order + dead_time - 1)*n_inputs+...
                            n_inputs+n_inputs+1+n*n*den_order+...
                            n*n_inputs*num_order+n*n;

        sizes.NumInputs = n+n+n*n*den_order+n*n_inputs*num_order+n*n;
```

```
    sizes.DirFeedthrough = 1;

    sizes.NumSampleTimes = 1;

    sys=simsizes(sizes);

    str=[];
    ts=[tsamp 0];

    %Initial valus of the state vector
    x0=[zeros((den_order + 1)*n,1);zeros(n_inputs,1);...
        zeros(n_inputs,1);...
        zeros((num_order + dead_time - 1)*n_inputs,1);...
        zeros(n_inputs,1);zeros(1,1);zeros(n_inputs,1);...
        zeros(n*n*den_order+n*n_inputs*num_order+n*n,1);];

case 2

        %Zeroing the variables
        ympast = zeros((den_order + 1)*n,1);
        uppast1 = zeros(n_inputs,1);
        um_cpast = zeros(n_inputs,1);
        umpast = zeros((num_order + dead_time - 1)*n_inputs,1);
        uu = zeros(n_inputs,1);

        %Extracting the previously saved state variables
        ympast(:)= x(1:(den_order + 1)*n);
        uppast1(:) = x((den_order + 1)*n+1:(den_order + 1)*n+n_inputs);
        um_cpast(:) = x((den_order + 1)*n+...
                        n_inputs+1:(den_order + 1)*n+...
                        n_inputs+n_inputs);
        umpast(:) = x((den_order + 1)*n+n_inputs+...
                       n_inputs+1:(den_order + 1)*n+...
                       n_inputs+n_inputs+...
                       (num_order + dead_time - 1)*n_inputs);
        uu(:) = x((den_order + 1 )*n+n_inputs+n_inputs+...
                  (num_order + dead_time - 1)*n_inputs+1:...
                  (den_order + 1)*n+n_inputs+n_inputs+...
                  (num_order + dead_time - 1)*n_inputs+n_inputs);

        %Updating ympast
        y = u(1:n);
        ympast = [y; ympast(1:length(ympast)-length(y))];

        %Switching between adaptive and backup controller
        if t > tstart
            theta = u(n+n+1:n+n+n*n*den_order+...
                      n*n_inputs*num_order+n*n);
            if theta(1) > 0 || theta(1) < -1
                theta = [-0.89758 -0.17703 0]';
            end
            %Calculation of move suppression weight
            L = L*((100-0)/(1.5-0.5))^2;
            if t > tune_start_c
                if theta(1) < 0 && theta(1) > -1
```

```
            tau = 1/(log(-theta(1))/-tsamp);
            if isreal(tau) == 1 && tau > 0
                f = (M/500)*((3.5*tau/tsamp)+2-((M-1)/2));
                K = theta(2)/(1+theta(1));
                L = f*K^2;
            end
        end
    end
else
    theta = [-0.89758 -0.17703 0]';
    L = L*((100-0)/(1.5-0.5))^2;
end

%Model Parameters
a111 = theta(1);
a112 = 0;
a121 = 0;
a122 = 0;

a211 = 0;
a212 = 0;
a221 = 0;
a222 = 0;

b111 = theta(2);
b112 = 0;
b113 = 0;
b121 = 0;
b122 = 0;
b123 = 0;

b211 = 0;
b212 = 0;
b213 = 0;
b221 = 0;
b222 = 0;
b223 = 0;

%Adding an integrator to the model
A0 = [1 0;0 1;];
A1 = [a111 - 1 a112;a121 a122 - 1;];
A2 = [a211 - a111 a212 - a112;a221 - a121 a222 - a122;];
A3 = [-a211 -a212;-a221 -a222;];
B1 = [b111 b112 b113;b121 b122 b123;];
B2 = [b211 b212 b213;b221 b222 b223;];

%Output Model Parameters (Matrix A)
A(1,1,1) = A0(1,1);
A(1,1,2) = A0(1,2);
A(1,2,1) = A0(2,1);
A(1,2,2) = A0(2,2);

A(2,1,1) = A1(1,1);
A(2,1,2) = A1(1,2);
A(2,2,1) = A1(2,1);
```

```
A(2,2,2) = A1(2,2);

A(3,1,1) = A2(1,1);
A(3,1,2) = A2(1,2);
A(3,2,1) = A2(2,1);
A(3,2,2) = A2(2,2);

A(4,1,1) = A3(1,1);
A(4,1,2) = A3(1,2);
A(4,2,1) = A3(2,1);
A(4,2,2) = A3(2,2);

%Input Model Parameters (Matrix B)
B(1,1,1) = B1(1,1);
B(1,1,2) = B1(1,2);
B(1,1,3) = B1(1,3);
B(1,2,1) = B1(2,1);
B(1,2,2) = B1(2,2);
B(1,2,3) = B1(2,3);

B(2,1,1) = B2(1,1);
B(2,1,2) = B2(1,2);
B(2,1,3) = B2(1,3);
B(2,2,1) = B2(2,1);
B(2,2,2) = B2(2,2);
B(2,2,3) = B2(2,3);

%Intermediate Information
dummy_nA = den_order + 1;

%CA and Czb Calculations
for i = 1 : P
    count = 0;
    for j =i : min(i + den_order + 1,P)
        count = count + 1;
        for col = n*i - n + 1
            for row =n*j - n + 1
                for ii = 1 : n
                    for jj = 1 : n
                        CA(row + ii - 1,col + jj - 1) = ...
                                                A(count,ii,jj);
                    end
                end
            end
        end
    end
    count = 0;
    for j= i : min(i + num_order - 1,P)
        count = count + 1;
        for col = n_inputs*i - n_inputs + 1
            for row = n*j - n + 1
                for ii = 1 : n
                    for jj = 1 : n_inputs
                        Czb(row + ii - 1,col + jj - 1) = ...
                                                B(count,ii,jj);
```

```
                    end
                end
            end
        end
    end
end

Czb = Czb(:,dead_time*n_inputs+1:end);

%HA Calculations
HA = zeros(n*P,n*dummy_nA);
for i = 1 : P
    count = i + 1;
    for j= 1 : dummy_nA
        for col = n*j - n + 1
            for row = n*i - n + 1
                for ii = 1 : n
                    for jj = 1 : n
                        HA(row + ii - 1,col + jj - 1) = ...
                                                A(count,ii,jj);
                    end
                end
            end
        end
        count = count + 1;
    end
    dummy_nA = dummy_nA - 1;
end

%Hzb Calculations
Hzb = zeros(n*P,n_inputs*(num_order - 1));
nn = num_order;
for i = 1 : nn - 1
    count = i + 1;
    for j = 1 : P
        if count <= nn
            for col = n_inputs*i - n_inputs + 1
                for row = n*j - n + 1
                    for ii = 1 : n
                        for jj = 1 : n_inputs
                            Hzb(row + ii - 1,col + jj - 1) =...
                                                B(count,ii,jj);
                        end
                    end
                end
            end
        end
        count = count + 1;
    end
end

%Hzb_plus Calculations
Hzb_plus = zeros(n*P,-n_inputs + n_inputs*dead_time +...
              n_inputs*num_order - ((num_order - 1)*n_inputs));
for bb = 1 : num_order
```

```
    for k = 1 : dead_time
        col = 1 + n_inputs*(dead_time - k);
        row = (1 - 2*n) + n*(bb + k);
        for ii = 1 : n
            for jj = 1 : n_inputs
                Hzb_plus(row + ii - 1,col + jj - 1) =...
                                                    B(bb,ii,jj);
            end
        end
    end
end
Hzb = [Hzb_plus Hzb];

%Computing the matrices for prediction equation
AA = CA\Czb;
for col = 1 : n_inputs*M
    for row = 1 : n*P
        H(row,col) = AA(row,col);
    end
end

H = H(n*N - n + 1:end,:);
PP = CA\Hzb;
PP = PP(n*N - n + 1:end,:);
Q = -CA\HA;
Q = Q(n*N - n + 1:end,:);

%Calculating unit vector for separating the first input
e = zeros(n_inputs,n_inputs*M);
for i = 1 : n_inputs
    for j = i
        e(i,j) = 1;
    end
end

%Define the reference signal
rr = [u(2)]';

for j = 1 : n : n*(P - N) + 1
    count = 1;
    for ii = 1 : n
        r(j + ii - 1,1) = rr(ii);
        count = count + 1;
    end
end

%Output weights for different output
for i = 1 : (P - N + 1)
    count = 1;
    for row = n*i - n + 1
        for col = n*i - n + 1
            for j = 1 : n
                Beta(row + j - 1,col + j - 1) = W(count);
                count = count + 1;
            end
```

```
        end
    end
end

%Move suppression weight for different inputs
for i = 1 : M
    count = 1;
    for row = n_inputs*i - n_inputs + 1
        for col = n_inputs*i - n_inputs + 1
            for j = 1 : n_inputs
                Lamda(row + j - 1,col + j - 1) = L(count);
                count = count + 1;
            end
        end
    end
end

%Hessian matrix HH and f
HH = round(1e14*(H'*Beta*H + Lamda))/1e14;
f = H'*Beta*(PP*umpast + Q*ympast - r);

%Producing Toeplitz Matrix Identity for use in Linear Inequalit
D = eye(n_inputs,n_inputs);
for i = 1 : M
    for j =i : M
        for col = n_inputs*i - n_inputs + 1
            for row =n_inputs*j - n_inputs + 1
                for ii = 1 : n_inputs
                    for jj = 1 : n_inputs
                        C_delta(row + ii - 1,col + jj - 1) =...
                                                    D(ii,jj);
                    end
                end
            end
        end
    end
end

%Producing LL[I]
DD = eye(n_inputs,n_inputs);
for i = 1
    for j = i : M;
        for col = n_inputs*i - n_inputs + 1
            for row =n_inputs*j - n_inputs + 1
                for ii = 1 : n_inputs
                    for jj = 1 : n_inputs
                            LL(row + ii - 1,col + jj - 1) =...
                                                    DD(ii,jj);
                    end
                end
            end
        end
    end
end
```

```
        %Define the slew rate and input constraints over control
        %horizon
        for j = 1 : n_inputs : n_inputs*M - n_inputs + 1
            count = 1;
            for ii = 1 : n_inputs
                delta_u_min(j + ii - 1,1) = slew_min(ii);
                delta_u_max(j + ii - 1,1) = slew_max(ii);
                umin(j + ii - 1,1) = u_min(ii);
                umax(j + ii - 1,1) = u_max(ii);
                count = count + 1;
            end
        end

        %Define the output constraints over the prediction horizon
        for j = 1 : n : n*(P - N) + 1
            count = 1;
            for ii = 1 : n
                ymin(j + ii - 1,1) = y_min(ii);
                ymax(j + ii - 1,1) = y_max(ii);
                count = count + 1;
            end
        end

        %Solving the QP problem by calling 'quadprog'
        C = [eye(n_inputs*M,n_inputs*M);-eye(n_inputs*M,n_inputs*M);...
            H;-H;C_delta;-C_delta;];
        b = [delta_u_max;-delta_u_min;ymax - PP*umpast - Q*ympast;...
            -ymin + PP*umpast + Q*ympast;umax - LL*um_cpast;...
            -umin + LL*um_cpast;];
        Options = optimset('LargeScale','off');
        [delta_u,fval,exitflag,output] = ...
        quadprog(HH,f,C,b,[],[],[],[],[],Options);

        %Isolating the first delta_u
        delta_u = e*delta_u;

        %Computing the control move u
        uu = delta_u + uppast1;

        %Updating the vectors of past moves
        um_cpast = uu;
        uppast1 = [uu; uppast1(1:length(uppast1)- length(uu))];
        umpast = [delta_u; umpast(1:length(umpast)- length(delta_u))];

        if num_order == 1 && dead_time == 0
            umpast = zeros((num_order + dead_time - 1)*n_inputs,1);
        end

        %Saving the state variables
        sys=[ympast(:);uppast1(:);um_cpast(:);umpast(:);uu(:);...
             P;L;theta(:);];

case 3

        y = u(1:n);
```

```
%Zeroing the variables
ympast = zeros((den_order + 1)*n,1);
uppast1 = zeros(n_inputs,1);
um_cpast = zeros(n_inputs,1);
umpast = zeros((num_order + dead_time - 1)*n_inputs,1);
uu = zeros(n_inputs,1);

%Extracting the previously saved state variables
ympast(:)= x(1:(den_order + 1)*n);
uppast1(:) = x((den_order + 1)*n+1:(den_order + 1)*n+n_inputs);
um_cpast(:) = x((den_order + 1)*n+...
                n_inputs+1:(den_order + 1)*n+...
                n_inputs+n_inputs);
umpast(:) = x((den_order + 1)*n+n_inputs+...
              n_inputs+1:(den_order + 1)*n+...
              n_inputs+n_inputs+...
              (num_order + dead_time - 1)*n_inputs);
uu(:) = x((den_order + 1 )*n+n_inputs+n_inputs+...
          (num_order + dead_time - 1)*n_inputs+1:...
          (den_order + 1)*n+n_inputs+n_inputs+...
          (num_order + dead_time - 1)*n_inputs+n_inputs);

%Updating ympast
ympast = [y; ympast(1:length(ympast)-length(y))];

%Switching between adaptive and backup controller
if t > tstart
    theta = u(n+n+1:n+n+n*n*den_order+...
              n*n_inputs*num_order+n*n);
    if theta(1) > 0 || theta(1) < -1
        theta = [-0.89758 -0.17703 0]';
    end
    %Calculation of move suppression weight
    L = L*((100-0)/(1.5-0.5))^2;
    if t > tune_start_c
        if theta(1) < 0 && theta(1) > -1
            tau = 1/(log(-theta(1))/-tsamp);
            if isreal(tau) == 1 && tau > 0
                f = (M/500)*((3.5*tau/tsamp)+2-((M-1)/2));
                K = theta(2)/(1+theta(1));
                L = f*K^2;
            end
        end
    end
else
    theta = [-0.89758 -0.17703 0]';
    L = L*((100-0)/(1.5-0.5))^2;
end

%Model Parameters
a111 = theta(1);
a112 = 0;
a121 = 0;
a122 = 0;
```

```
a211 = 0;
a212 = 0;
a221 = 0;
a222 = 0;

b111 = theta(2);
b112 = 0;
b113 = 0;
b121 = 0;
b122 = 0;
b123 = 0;

b211 = 0;
b212 = 0;
b213 = 0;
b221 = 0;
b222 = 0;
b223 = 0;

%Adding an integrator to the model
A0 = [1 0;0 1;];
A1 = [a111 - 1 a112;a121 a122 - 1;];
A2 = [a211 - a111 a212 - a112;a221 - a121 a222 - a122;];
A3 = [-a211 -a212;-a221 -a222;];
B1 = [b111 b112 b113;b121 b122 b123;];
B2 = [b211 b212 b213;b221 b222 b223;];

%Output Model Parameters (Matrix A)
A(1,1,1) = A0(1,1);
A(1,1,2) = A0(1,2);
A(1,2,1) = A0(2,1);
A(1,2,2) = A0(2,2);

A(2,1,1) = A1(1,1);
A(2,1,2) = A1(1,2);
A(2,2,1) = A1(2,1);
A(2,2,2) = A1(2,2);

A(3,1,1) = A2(1,1);
A(3,1,2) = A2(1,2);
A(3,2,1) = A2(2,1);
A(3,2,2) = A2(2,2);

A(4,1,1) = A3(1,1);
A(4,1,2) = A3(1,2);
A(4,2,1) = A3(2,1);
A(4,2,2) = A3(2,2);

%Input Model Parameters (Matrix B)
B(1,1,1) = B1(1,1);
B(1,1,2) = B1(1,2);
B(1,1,3) = B1(1,3);
B(1,2,1) = B1(2,1);
B(1,2,2) = B1(2,2);
```

```
B(1,2,3) = B1(2,3);

B(2,1,1) = B2(1,1);
B(2,1,2) = B2(1,2);
B(2,1,3) = B2(1,3);
B(2,2,1) = B2(2,1);
B(2,2,2) = B2(2,2);
B(2,2,3) = B2(2,3);

%Intermediate Information
dummy_nA = den_order + 1;

%CA and Czb Calculations
for i = 1 : P
    count = 0;
    for j =i : min(i + den_order + 1,P)
        count = count + 1;
        for col = n*i - n + 1
            for row =n*j - n + 1
                for ii = 1 : n
                    for jj = 1 : n
                        CA(row + ii - 1,col + jj - 1) = ...
                                                A(count,ii,jj);
                    end
                end
            end
        end
    end
    count = 0;
    for j= i : min(i + num_order - 1,P)
        count = count + 1;
        for col = n_inputs*i - n_inputs + 1
            for row = n*j - n + 1
                for ii = 1 : n
                    for jj = 1 : n_inputs
                        Czb(row + ii - 1,col + jj - 1) = ...
                                                B(count,ii,jj);
                    end
                end
            end
        end
    end
end

Czb = Czb(:,dead_time*n_inputs+1:end);

%HA Calculations
HA = zeros(n*P,n*dummy_nA);
for i = 1 : P
    count = i + 1;
    for j= 1 : dummy_nA
        for col = n*j - n + 1
            for row = n*i - n + 1
                for ii = 1 : n
                    for jj = 1 : n
```

```
                        HA(row + ii - 1,col + jj - 1) = ...
                                                A(count,ii,jj);
                    end
                end
            end
        end
        count = count + 1;
    end
    dummy_nA = dummy_nA - 1;
end

%Hzb Calculations
Hzb = zeros(n*P,n_inputs*(num_order - 1));
nn = num_order;
for i = 1 : nn - 1
    count = i + 1;
    for j = 1 : P
        if count <= nn
            for col = n_inputs*i - n_inputs + 1
                for row = n*j - n + 1
                    for ii = 1 : n
                        for jj = 1 : n_inputs
                            Hzb(row + ii - 1,col + jj - 1) =...
                                                B(count,ii,jj);
                        end
                    end
                end
            end
        end
        count = count + 1;
    end
end

%Hzb_plus Calculations
Hzb_plus = zeros(n*P,-n_inputs + n_inputs*dead_time +...
              n_inputs*num_order - ((num_order - 1)*n_inputs));
for bb = 1 : num_order
    for k = 1 : dead_time
        col = 1 + n_inputs*(dead_time - k);
        row = (1 - 2*n) + n*(bb + k);
        for ii = 1 : n
            for jj = 1 : n_inputs
                Hzb_plus(row + ii - 1,col + jj - 1) =...
                                                    B(bb,ii,jj);
            end
        end
    end
end
Hzb = [Hzb_plus Hzb];

%Computing the matrices for prediction equation
AA = CA\Czb;
for col = 1 : n_inputs*M
    for row = 1 : n*P
        H(row,col) = AA(row,col);
```

```
    end
end

H = H(n*N - n + 1:end,:);
PP = CA\Hzb;
PP = PP(n*N - n + 1:end,:);
Q = -CA\HA;
Q = Q(n*N - n + 1:end,:);

%Calculating unit vector for separating the first input
e = zeros(n_inputs,n_inputs*M);
for i = 1 : n_inputs
    for j = i
        e(i,j) = 1;
    end
end

%Define the reference signal
rr = [u(2)]';

for j = 1 : n : n*(P - N) + 1
    count = 1;
    for ii = 1 : n
        r(j + ii - 1,1) = rr(ii);
        count = count + 1;
    end
end

%Output weights for different output
for i = 1 : (P - N + 1)
    count = 1;
    for row = n*i - n + 1
        for col = n*i - n + 1
            for j = 1 : n
                Beta(row + j - 1,col + j - 1) = W(count);
                count = count + 1;
            end
        end
    end
end

%Move suppression weight for different inputs
for i = 1 : M
    count = 1;
    for row = n_inputs*i - n_inputs + 1
        for col = n_inputs*i - n_inputs + 1
            for j = 1 : n_inputs
                Lamda(row + j - 1,col + j - 1) = L(count);
                count = count + 1;
            end
        end
    end
end

%Hessian matrix HH and f
```

```
HH = round(1e14*(H'*Beta*H + Lamda))/1e14;
f = H'*Beta*(PP*umpast + Q*ympast - r);

%Producing Toeplitz Matrix Identity for use in Linear Inequalit
D = eye(n_inputs,n_inputs);
for i = 1 : M
    for j =i : M
        for col = n_inputs*i - n_inputs + 1
            for row =n_inputs*j - n_inputs + 1
                for ii = 1 : n_inputs
                    for jj = 1 : n_inputs
                        C_delta(row + ii - 1,col + jj - 1) =...
                                                    D(ii,jj);
                    end
                end
            end
        end
    end
end

%Producing LL[I]
DD = eye(n_inputs,n_inputs);
for i = 1
    for j = i : M;
        for col = n_inputs*i - n_inputs + 1
            for row =n_inputs*j - n_inputs + 1
                for ii = 1 : n_inputs
                    for jj = 1 : n_inputs
                            LL(row + ii - 1,col + jj - 1) =...
                                                    DD(ii,jj);
                    end
                end
            end
        end
    end
end

%Define the slew rate and input constraints over control
%horizon
for j = 1 : n_inputs : n_inputs*M - n_inputs + 1
    count = 1;
    for ii = 1 : n_inputs
        delta_u_min(j + ii - 1,1) = slew_min(ii);
        delta_u_max(j + ii - 1,1) = slew_max(ii);
        umin(j + ii - 1,1) = u_min(ii);
        umax(j + ii - 1,1) = u_max(ii);
        count = count + 1;
    end
end

%Define the output constraints over the prediction horizon
for j = 1 : n : n*(P - N) + 1
    count = 1;
    for ii = 1 : n
        ymin(j + ii - 1,1) = y_min(ii);
```

```
                    ymax(j + ii - 1,1) = y_max(ii);
                    count = count + 1;
                end
            end

            %Solving the QP problem by calling 'quadprog'
            C = [eye(n_inputs*M,n_inputs*M);-eye(n_inputs*M,n_inputs*M);...
                H;-H;C_delta;-C_delta;];
            b = [delta_u_max;-delta_u_min;ymax - PP*umpast - Q*ympast;...
                -ymin + PP*umpast + Q*ympast;umax - LL*um_cpast;...
                -umin + LL*um_cpast;];
            Options = optimset('LargeScale','off');
            [delta_u,fval,exitflag,output] = ...
            quadprog(HH,f,C,b,[],[],[],[],[],Options);

            %Isolating the first delta_u
            delta_u = e*delta_u;

            %Computing the control move u
            uu = delta_u + uppast1;

            %Updating the vectors of past moves
            um_cpast = uu;
            uppast1 = [uu; uppast1(1:length(uppast1)- length(uu))];
            umpast = [delta_u; umpast(1:length(umpast)- length(delta_u))];

            if num_order == 1 && dead_time == 0
                umpast = zeros((num_order + dead_time - 1)*n_inputs,1);
            end

            %Sending the value of u for implementation to the process
            x((den_order + 1)*n+n_inputs+n_inputs+...
              (num_order + dead_time - 1)*n_inputs+...
               1:(den_order + 1)*n+n_inputs+n_inputs+...
              (num_order + dead_time - 1)*n_inputs+n_inputs) = uu(:);

        sys=x;

    case {1 4 9}

        sys=[];

    otherwise

        error(['Unhandled flag = ',num2str(flag)]);

end
```