

**SUPPORT VECTOR REGRESSION BASED CONTROLLER DESIGN
METHODS FOR NONLINEAR SYSTEMS**



Ph.D. THESIS

Kemal UÇAK

Department of Control and Automation Engineering

Control and Automation Engineering Programme

JULY 2016

**SUPPORT VECTOR REGRESSION BASED CONTROLLER DESIGN
METHODS FOR NONLINEAR SYSTEMS**



Ph.D. THESIS

**Kemal UÇAK
(504112109)**

Department of Control and Automation Engineering

Control and Automation Engineering Programme

Thesis Advisor: Assist.Prof. Dr. Gülay ÖKE GÜNEL

JULY 2016

**LİNEER OLMAYAN SİSTEMLER İÇİN DESTEK VEKTÖR REGRESYON
TABANLI KONTROLÖR TASARIM METODLARI**

DOKTORA TEZİ

**Kemal UÇAK
(504112109)**

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Yrd. Doç. Dr. Gülay ÖKE GÜNEL

TEMMUZ 2016

Kemal UÇAK, a Ph.D. student of ITU Graduate School of Science Engineering and Technology student ID 504112109 , successfully defended the thesis entitled “SUPPORT VECTOR REGRESSION BASED CONTROLLER DESIGN METHODS FOR NONLINEAR SYSTEMS”, which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Assist.Prof. Dr. Gülay ÖKE GÜNEL**
Istanbul Technical University

Jury Members : **Prof. Dr. İbrahim EKSİN**
Istanbul Technical University

Prof. Dr. Müjde GÜZELKAYA
Istanbul Technical University

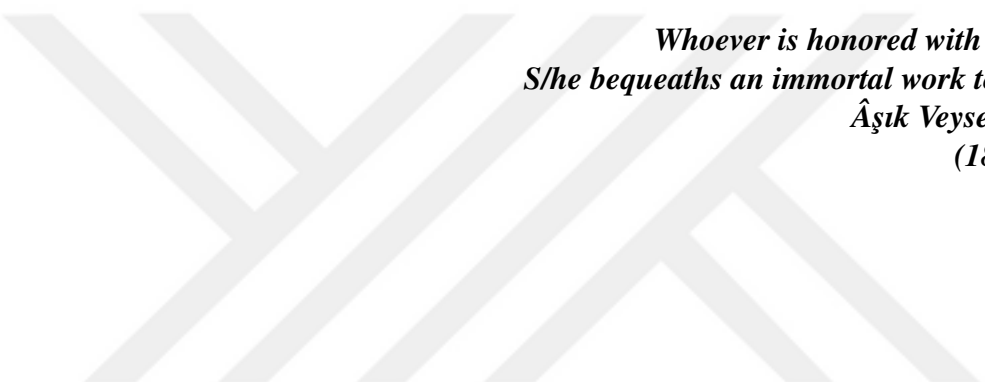
Assist. Prof. Dr. İlker ÜSTOĞLU
Yıldız Technical University

Assist. Prof. Dr. Dilek BİLGİN TÜKEL
Doğuş University

Date of Submission : **27 May 2016**

Date of Defense : **13 July 2016**





*Whoever is honored with this secret,
S/he bequeaths an immortal work to the world*
Âşık Veysel Şatıroğlu
(1894 - 1973)



FOREWORD

I would like to thank the following people without their help and support this Ph.D. thesis would not have been possible. Firstly, I would like to show my gratitude to my supervisor Asst.Prof. Dr. Gülay Öke Günel for her suggestions, encouragements and guidance in approaching the different challenges during the thesis. I would also like to thank my thesis progress jury members Prof. Dr. İbrahim Eksin and Asst. Prof. Dr. İlker Üstoğlu for their support, vision, and advisements which have helped to improve the quality of this thesis.

I am grateful to all the people from Department of Control and Automation Engineering, Istanbul Technical University, for being a family during the many years I spent in Istanbul.

Finally, I would like to thank my parents and my friends for their constant support during the time I studied.

July 2016

Kemal UÇAK
(Research Assistant)

TABLE OF CONTENTS

	<u>Page</u>
FOREWORD.....	ix
TABLE OF CONTENTS.....	xi
ABBREVIATIONS	xv
LIST OF TABLES	xvii
LIST OF FIGURES	xix
SUMMARY	xxiii
ÖZET	xxv
1. INTRODUCTION	1
2. A NOVEL ADAPTIVE NARMA-L2 CONTROLLER BASED ON ON-LINE SUPPORT VECTOR REGRESSION FOR NONLINEAR SYSTEMS .	7
2.1 Introduction	7
2.2 NARMA-L2 Model and Controller.....	13
2.3 Online ε - Support Vector Regression	15
2.4 NARMA Controller based on Online SVR	18
2.4.1 Identification of the NARMA model by SVR.....	18
2.4.2 Controller design	22
2.4.3 Adaptive predictive SVR _{NARMA-L2} controller.....	22
2.4.4 Adaptive predictive SVR _{NARMA-L2} controller with adaptive filter.....	25
2.4.5 Control procedure.....	28
2.5 Simulation Results.....	29
2.5.1 The bioreactor system.....	29
2.5.2 SVR design parameters	30
2.5.3 Simulation results	32
2.5.4 Comparison of the results with SVM-based PID controller.....	32
2.6 Conclusion.....	35
3. AN ADAPTIVE SUPPORT VECTOR REGRESSOR CONTROLLER FOR NONLINEAR SYSTEMS	41
3.1 Introduction	41
3.2 Online Support Vector Regression	46
3.2.1 An overview of support vector regression.....	46
3.2.2 Basic principles of online support vector regression.....	48
3.2.3 Derivation of update rules for Lagrange multipliers	49
3.2.4 Calculation of $\Delta\lambda_c$	52
3.3 Adaptive Online SVR Controller based on Model Estimated by an Online SVR	54
3.3.1 An overview of the proposed control architecture	54
3.3.2 Generalized closed-loop system margin.....	57

3.3.3 Online support vector regression for controller design	62
3.3.4 Stability analysis of the closed-loop system.....	65
3.4 Simulation Results.....	67
3.4.1 Simulation results for continuously stirred tank reactor system	69
3.4.1.1 Noiseless condition.....	70
3.4.1.2 Measurement noise	72
3.4.1.3 Uncertainty in system parameters.....	72
3.4.1.4 Closed-loop Lyapunov stability analysis	74
3.4.2 Simulation results for bioreactor system	74
3.4.2.1 Noiseless condition.....	75
3.4.2.2 Measurement noise	75
3.4.2.3 Uncertainty in system parameters.....	76
3.4.2.4 Closed-loop Lyapunov stability analysis	77
3.4.3 Comparison of the results with adaptive PID based on SVR	78
3.5 Conclusion.....	81
4. GENERALIZED SELF-TUNING REGULATOR BASED ON ONLINE SUPPORT VECTOR REGRESSION.....	85
4.1 Introduction	85
4.2 Online support vector regression.....	90
4.2.1 An overview of support vector regression.....	90
4.2.2 Online ε -support vector regression	92
4.3 Contruction of Optimization Problem for Self-Tuning Regulator	96
4.3.1 An overview of self-tuning regulators.....	96
4.3.2 Generalized STR structure based on SVR.....	96
4.3.3 PID type STR based on SVR.....	99
4.3.4 Fuzzy PID type STR based on SVR.....	100
4.3.5 Adaptive control algorithm for the generalized STR based on SVR	103
4.3.6 Generalized closed-loop system margin.....	105
4.3.7 Online support vector regression for parameter estimator	108
4.3.8 Stability analysis of the closed-loop system.....	110
4.3.8.1 Stability analysis for the generalized STR based on SVR.....	110
4.3.8.2 Derivation of the sensitivty functions for PID type STR based on SVR.....	113
4.3.8.3 Derivation of the sensitivty functions for Fuzzy PID type STR based on SVR	114
4.4 Simulation Results.....	116
4.4.1 Bioreactor system	116
4.4.2 Nominal case with no noise and parametric uncertainty	117
4.4.3 Meaurement noise	119
4.4.4 Uncertainty in system parameters	119
4.4.5 Closed-loop Lyapunov stability analysis.....	120
4.4.6 Comparison of the results with SVM-based PID controller.....	121
4.5 Conclusion.....	124
Conflict of Interest.....	125
5. CONCLUSIONS AND RECOMMENDATIONS.....	127

REFERENCES.....	129
CURRICULUM VITAE	139





ABBREVIATIONS

ABS	: Anti-lock Braking System
AIMC	: Adaptive Internal Model Controller
ANFIS	: Adaptive Neuro-Fuzzy Inference System
ANN	: Artificial Neural Network
CPU	: Central Processing Unit
CSTR	: Continuously Stirred Tank Reactor
DC	: Direct Current
DOF	: Degree Of Freedom
EMRAN	: Extended Minimal Resource Allocation Algorithm
EKF	: Extended Kalman Filter
FIS	: Fuzzy Inference System
FLC	: Fuzzy Logic Control
FPGA	: Field Programmable Gate Array
GA	: Genetic Algorithm
IMC	: Internal Model Controller
KKT	: Karush Kuhn Tucker
LSSVR	: Least Square Support Vector Regression
MAGLEV	: Magnetic Levitation System
MB	: Model Based Control
MLP	: Multi Layer Perceptron
MPC	: Model Predictive Controller
MRAC	: Model Reference Adaptive Control
NARMA	: Nonlinear Autoregressive Moving Average
NARX	: Nonlinear Autoregressive with Exogenous Inputs
NIMC	: Nonlinear Internal Model Control
OLSSVR	: Online Least Square Support Vector Regression
PID	: Proportional Integral Derivative
QP	: Quadratic Programming
RBFNN	: Radial Basis Function Neural Network
RNN	: Recurrent Neural Network
SISO	: Single Input Single Output
SMO	: Sequential Minimal Optimization
STR	: Self Tuning Regulator
SV	: Support Vector
SVC	: Support Vector Classifier
SVM	: Support Vector Machines
SVR	: Support Vector Regression
SVR_{controller}	: Support Vector Regressor Controller
SVR_{estimator}	: Support Vector Regressor Estimator
SVR_{NARMA-L2}	: NARMA-L2 Support Vector Regression Model
SVR_{NARX}	: NARX Support Vector Regression Model
SVR_{model}	: Support Vector Regressor Model
TS	: Takagi Sugeno



LIST OF TABLES

	<u>Page</u>
Table 2.1 : Computation times(in ms).....	35
Table 3.1 : Computation times(in ms) for SVR _{controller}	80
Table 4.1 : Maximum values of steady state errors ($e_{ss}(\%)$), settling times ($t_s(2\%)$ (sec)) and overshoots ($P.O.(\%)$).	124
Table 4.2 : Average values of steady state errors ($e_{ss}(\%)$), settling times ($t_s(2\%)$ (sec)) and overshoots ($P.O.(\%)$).	124



LIST OF FIGURES

	<u>Page</u>
Figure 1.1 : Block diagram of model based adaptive control systems.....	2
Figure 1.2 : Flow chart of the thesis.....	5
Figure 2.1 : NARMA model.	13
Figure 2.2 : NARMA controller.	14
Figure 2.3 : E , S and R subsets before (a) and after (b) training	18
Figure 2.4 : Decomposition of SVR _{NARX} model (a) to SVR _{NARMA-L2} model (b).	19
Figure 2.5 : SVR _{NARMA-L2} controller based on online SVR.	23
Figure 2.6 : Adaptive SVR _{NARMA-L2} controller based on online SVR.	24
Figure 2.7 : Adaptive SVR _{NARMA-L2} controller based on online SVR with adaptive filter.....	26
Figure 2.8 : Tracking performance surface with respect to penalty term (λ) and prediction horizon (K) for bioreactor.	31
Figure 2.9 : System output (a), control signal (b), and number of support vectors (c) with adaptive filter for the case with no noise or parametric uncertainty.....	33
Figure 2.10 : Adaptive controller (a,b) and filter parameters (c) for the case with no noise or parametric uncertainty.	34
Figure 2.11 : System output (a), control signal (b), and number of support vectors (c) with adaptive filter for the case with measurement noise.	35
Figure 2.12 : Adaptive controller (a,b) and filter parameters (c) for the case with measurement noise.	36
Figure 2.13 : System output (a), control signal (b), uncertain system parameter (c), and number of support vectors (d) with adaptive filter for the case with parametric uncertainty.....	37
Figure 2.14 : Adaptive controller (a,b) and filter parameters (c) for the case with parametric uncertainty.....	38
Figure 2.15 : Tracking performance of SVR _{NARMA-L2} controller (a) and SVM-based PID controller (b) for the case with no noise or parametric uncertainty.....	38
Figure 2.16 : Tracking performance of SVR _{NARMA-L2} controller (a) and SVM-based PID controller (b) for the case with measurement noise.	39
Figure 2.17 : Tracking performance of SVR _{NARMA-L2} controller (a) and SVM-based PID controller (b) for the case with parametric uncertainty (c).	39
Figure 2.18 : Tracking performance comparison of the controller with respect to the defined performance index (2.51).	40
Figure 3.1 : E , S and R subsets before (a) and after (b) training.	49
Figure 3.2 : The block diagram of the proposed control architecture.	54

Figure 3.3 : The adaptation mechanisms of SVR _{controller} and SVR _{model} .	55
Figure 3.4 : Margins of SVR _{controller} (a) and SVR _{model} (b).	58
Figure 3.5 : Closed-loop system margin in three dimensions.	59
Figure 3.6 : Projected closed loop margin before (a) and after (b) training.	59
Figure 3.7 : Projected closed loop margin before (a) and after (b) training.	61
Figure 3.8 : Closed-loop system margin in three dimensions.	61
Figure 3.9 : Combined controller and system model margins in three dimensions.	61
Figure 3.10 : System output (a), control signal (b) for variable step input.	70
Figure 3.11 : Adaptation of SVR _{controller} parameters (left), SVR _{model} parameters (right).	71
Figure 3.12 : System output (a), control signal (b) for sinusoidal input.	71
Figure 3.13 : Adaptation of SVR _{controller} parameters (left), SVR _{model} parameters (right).	71
Figure 3.14 : System output (a), control signal (b) for variable step input.	72
Figure 3.15 : Adaptation of SVR _{controller} parameters (left), SVR _{model} parameters (right).	73
Figure 3.16 : System output (a), control signal (b), time varying system parameter (c).	73
Figure 3.17 : $V(t)$ (a) and $\frac{\partial V(t)}{\partial t}$ (b) for noiseless (left), noisy (right) and with parametric uncertainty cases (middle).	74
Figure 3.18 : System output (a), control signal (b) for variable step input.	76
Figure 3.19 : Adaptation of SVR _{controller} parameters (left), SVR _{model} parameters (right).	76
Figure 3.20 : System output (a), control signal (b) for variable step input.	77
Figure 3.21 : Adaptation of SVR _{controller} parameters (left), SVR _{model} parameters (right).	77
Figure 3.22 : System output (a), control signal (b), time varying parameter (c).	78
Figure 3.23 : Adaptation of SVR _{controller} parameters (left), SVR _{model} parameters (right).	79
Figure 3.24 : $V(t)$ (a) and $\frac{\partial V(t)}{\partial t}$ (b) for noiseless (left), noisy (right) and with parametric uncertainty cases (middle).	80
Figure 3.25 : Tracking performance of SVR _{controller} (a) and SVM-based PID controller (b) with no noise.	81
Figure 3.26 : Tracking performance of SVR _{controller} (a) and SVM-based PID controller (b) with measurement noise.	82
Figure 3.27 : Tracking performance of SVR _{controller} (a) and SVM-based PID controller (b) with parametric uncertainty (c).	82
Figure 3.28 : Tracking performance of SVR _{controller} (a) and SVM-based PID controller (b) with no noise.	83
Figure 3.29 : Tracking performance of SVR _{controller} (a) and SVM-based PID controller (b) with measurement noise.	83
Figure 3.30 : Tracking performance of SVR _{controller} (a) and SVM-based PID controller (b) with parametric uncertainty (c).	84
Figure 3.31 : Tracking performance comparison of the controller with respect to the defined performance index (3.67).	84

Figure 4.1 : E , S and R subsets before (a) and after (b) training [1–4].....	92
Figure 4.2 : Self-tuning regulator.....	97
Figure 4.3 : Generalized self-tuning regulator based on online SVR.	98
Figure 4.4 : Fuzzy PID controller.....	100
Figure 4.5 : The membership functions for inputs and rule base.....	101
Figure 4.6 : Fuzzy control surface.....	101
Figure 4.7 : Margins of SVR _{estimator} (a), adaptive controller (b) and SVR _{model} (c).	106
Figure 4.8 : Closed-loop system margin in three dimensions.....	107
Figure 4.9 : Closed loop margin before (a) and after (b) training.....	108
Figure 4.10 : Closed loop margin before (a) and after (b) training.....	108
Figure 4.11 : Closed-loop system margin in three dimensions.....	108
Figure 4.12 : System output (a), control signal (b) for the case with no noise and parametric uncertainty.....	117
Figure 4.13 : PID type STR parameters.....	118
Figure 4.14 : Fuzzy PID type STR parameters.....	118
Figure 4.15 : Number of the support vectors for controllers (a) and system models (b).	118
Figure 4.16 : System output (a), control signal (b) for sinusoidal input.....	119
Figure 4.17 : System output (a), control signal (b) for the case with measurement noise (30 dB).	119
Figure 4.18 : System output (a), control signal (b) for the case with parametric uncertainty (c).	120
Figure 4.19 : $V(t)$ (a) and $\frac{\partial V(t)}{\partial t}$ (b) for the case with no noise and parametric uncertainty.....	120
Figure 4.20 : $V(t)$ (a) and $\frac{\partial V(t)}{\partial t}$ (b) for the case with measurement noise.....	121
Figure 4.21 : $V(t)$ (a) and $\frac{\partial V(t)}{\partial t}$ (b) for the case with parametric uncertainty.....	121
Figure 4.22 : Comparison of controllers for the case with no noise and parametric uncertainty (left) and for the case with measurement noise (right).....	122
Figure 4.23 : Comparison of controllers for the case with parametric uncertainty.....	123
Figure 4.24 : Performance comparison of controllers with respect to the defined performance index (4.68).....	123



SUPPORT VECTOR REGRESSION BASED CONTROLLER DESIGN METHODS FOR NONLINEAR SYSTEMS

SUMMARY

Conventional nonadaptive controllers may be convenient enough for most industrial processes in the case that the system does not include highly nonlinear and time varying dynamics. Depending on the developments in technology and increasing requirements of the people, systems are transforming into more and more complex structures. Nonadaptive controllers cannot counteract the tracking error in case the behaviour of the system changes perceptibly owing to internal or external factors during the course of operation and the control performance degrades. For this purpose, different controllers for different scenario conditions and operating points may be designed to resume successful control in industry. Controller design depending on different performance criteria is a momentous task since effective design of controllers can significantly reduce the costs in industry. However, that chore can also be automatically executed by an adaptive controller. Therefore, controllers that are able to adapt themselves according to varying system dynamics must be developed using optimization theory based solutions to enhance the success of the controllers in industry.

Adaptive controllers can generally be classified as model based adaptive controllers and model free adaptive controllers. Model based adaptive controllers utilize the system model to approximate the behaviour of the system dynamics whereas in model free adaptive controllers the adjustment mechanism does not need the model of the system. The performances of model based adaptive controllers are directly influenced by the accuracy of the system model.

Machine learning algorithms have frequently been utilized to identify the dynamical behaviour of the system precisely in order to derive effective adjustment rules for the parameters of adaptive controllers. Support Vector Regression (SVR), proposed by Vladimir Vapnik et al., is one of the most favorable nonlinear system identification methods since it guarantees the global extremum of the optimization problem.

Various adaptive controllers based on SVR have been proposed for the control of nonlinear systems in technical literature. The common property of these studies is that SVR is typically deployed to approximate the system Jacobian which is required to tune parameters of adaptive controllers through gradient based optimization algorithms.

In this thesis, three novel adaptive controller architectures based on SVR have been proposed. The basic novelty in these architectures is that SVR has been implemented directly as a controller and a parameter estimator for a generalized controller structure. The performance evaluations of the proposed controllers have been examined on various nonlinear benchmark systems by simulations. In addition, stability analysis

of the systems have been performed. The obtained results prove that the proposed adjustment mechanisms are effective in controlling processes with nonlinear dynamics, noise and uncertainties.



LİNEER OLMAYAN SİSTEMLER İÇİN DESTEK VEKTÖR REGRESYON TABANLI KONTROLÖR TASARIM METODLARI

ÖZET

Geleneksel kontrolörler, sistemin lineer olmayan ve zamanla değişen dinamik içermemesi durumunda, endüstriyel süreçlerin çoğunluğunda yeterince etkili ve verimli olabilirler. Teknolojideki yeniliklere ve insanların artan gereksinimlerine bağlı olarak, sistemler çok daha karmaşık yapılara dönüşmüştür. Eğer kontrol işlemi sırasında sistemin davranışı, etkiyen iç ve dış dinamiklerden dolayı, ciddi derecede değişirse, geleneksel kontrolör izleme hatasının etkisini yok edememekte ve kontrol performansı kötüleşmektedir. Bu nedenle, endüstride kontrol sürecinin devamlılığını sağlamak için, farklı çalışma noktaları ve çalışma senaryolarının her biri için farklı kontrolörler tasarlanabilir. Endüstrideki çoğu kontrol süreci için, farklı performans ölçütlerine bağlı olarak kontrolör tasarımı önemli işlemsel yük içermektedir. Bu nedenle kontrolörlerin efektif bir biçimde tasarımı, endüstride maliyeti önemli ölçüde düşürmektedir.

Lineer veya uyarlamalı olmayan, sabit katsayılı kontrolörler bir kez tasarlandığında, sadece başlangıçta tasarlandığı sistemi kontrol edebilmekte ve genellikle lineer olmayan sistemler için kabul edilebilir sistem davranışı sağlayamamaktadırlar. Çevre sürekli değişen dinamikler içerdiği için, bu değişimlere ayak uydurabilecek, esnek, uyarlamalı kontrolör yapılarına ihtiyaç bulunmaktadır. Sistemlerin lineer olmayan, zamanla değişen ve/veya zaman gecikmesi içeren dinamiklerinin üstesinden gelmek için, geleneksel kontrolör topolojilerine adaptasyon yeteneği katılarak esneklik kazandırılabilir. Bu bağlamda, iç/dış faktörlere bağlı olarak yapısını veya davranışını modifiye edebilen kontrolör, uyarlamalı kontrolör olarak adlandırılır.

Uyarlamalı kontrolörler, model bağımsız ve model tabanlı uyarlamalı kontrolörler olmak üzere iki ana başlık altında sınıflandırılabilirler. Model bağımsız uyarlamalı kontrolde herhangi bir sistem modeli ve sistem tanılama evresi gerekmemektedir.

Model tabanlı uyarlamalı kontrolde ise, sistemin gelecekteki durumu öngörülerek sistem modelinin tanılanması gerekmektedir. Amaç, sistem dinamiklerinin gelecekteki davranışını dikkate alan bir uyarlama kuralına kullanarak kontrolör parametrelerini uyarlayarak, sistemin çıkış işaretini referans işaretini takip etmeye zorlamaktır. Dolayısıyla, model tabanlı uyarlamalı kontrolde, sistem tanılama ve kontrolör tasarımı bir arada yapılmalıdır. Seçilen kontrolör yapısına, sistem modeline ve uyarlama kuralına bağlı olarak, çeşitli model tabanlı uyarlamalı kontrolör yapıları önerilebilir. Sistem dinamiğinin hassas bir şekilde kestirilmesi ve kontrolör parametrelerinin uyarlanması için modelin doğruluğu ve hassasiyeti büyük önem arz etmektedir. Model tabanlı kontrolörler genellikle sistem dinamiklerinin iyi bir şekilde temsiline dayanmaktadır. Bu amaçla, güçlü modelleme kapasitelerinden dolayı, yapay sinir ağları (YSA), uyarlamalı sinirsel bulanık çıkarım sistemi (UBSÇS) ve destek

vektör regresyon (DVR) gibi akıllı modelleme metodları, lineer olmayan sistemlerin dinamiklerini tanılamak için sıklıkla tercih edilmektedirler.

İlk olarak Vapnik vd. tarafından önerilen DVR yöntemi, konveks olmayan birincil formdaki optimizasyon problemini, konveks ikincil bir forma dönüştürerek global ekstremumu garanti ettiği için, makine öğrenmesi alanında en etkili regresyon tekniklerinden biridir. YSA ve UBSÇS, konveks olmayan amaç fonksiyonlarını optimize ettiklerinden lokal minimuma takılma riskine sahiptirler, dolayısıyla DVR tabanlı kontrolör yapıları son yıllarda sıklıkla YSA ve UBSÇS yerine kullanılmaktadır.

Teknik literatürde, çeşitli DVR tabanlı uyarlamalı kontrolör yapıları önerilmiştir. Bu yapılarının ortak özelliği, DVR'nin sistem dinamiğini modellemek veya türev tabanlı optimizasyon metodları ile geleneksel kontrolörlerin parametrelerinin uyarlanması için gereken sistem Jacobian bilgisini kestirmek için kullanılmasıdır.

Bu doktora çalışmasında, temel olarak DVR'nin doğrudan doğruya kontrolör veya parametre kestirici olarak kullanıldığı DVR tabanlı yeni uyarlama mekanizmalarının tasarımı amaçlanmıştır. Böylece, lineer olmayan sistemlerin kontrolü için, DVR tabanlı üç tane yeni uyarlamalı kontrol mekanizması önerilmiştir. Önerilen yapılar aşağıdaki gibi adlandırılmaktadır:

- DVR tabanlı uyarlamalı NARMA-L2 kontrolör
- Uyarlamalı DVR kontrolör
- DVR tabanlı genelleştirilmiş öz-uyarlamalı regülatör

İlk önerilen kontrolör yapısında, DVR'nin güçlü modelleme yeteneği ve NARMA-L2 kontrolörün fonksiyonelliğini birleştirilerek, lineer olmayan sistemler için DVR tabanlı NARMA-L2 kontrolör yapısı önerilmiştir. Bu yapıda, sistemin önceden elde edilmiş NARX modelinden yararlanarak NARMA-L2 alt modellerin parametrelerinin elde edilmesi ve bu alt modellerden NARMA-L2 kontrolörün tasarımı amaçlanmıştır. Sistemin NARX modelinden NARMA modeline geçiş için dönüşüm parametreleri kullanılmıştır. Dönüşümü sağlayan parametrelerin optimizasyonu için Levenberg-Marquardt optimizasyon algoritmasından yararlanılmıştır. Sistemin K-adım sonraki davranışı öngörülerek dönüşüm parametreleri optimize edilmiştir. Kontrol sisteminin başarımı, lineer olmayan bioreaktör sistemi üzerinde değerlendirilmiştir. Uyarlama mekanizmasının dayanıklılığı, sistemin çıkışına ölçme gürültüsü eklenmesi ve sisteme parametrik belirsizlik katılması durumlarında test edilmiştir. Önerilen kontrolörün performansı, [5]'de önerilen DVR tabanlı PID kontrolör yapısının performansı ile karşılaştırılmıştır. Sonuçlar, önerilen kontrol mekanizmasının, lineer olmayan dinamik sistemleri kontrol etmek için başarılı bir şekilde uygulanabileceğini göstermektedir.

İkinci önerilen kontrol yapısında, DVR, ilk defa doğrudan doğruya kontrolör olarak kullanılmıştır. Kontrolör çıkışı hakkında herhangi bir ön bilgi gerekmesizin kontrolör parametrelerinin uyarlanmasını sağlamak için, referans giriş işaretinin ve sistem çıkışının fonksiyonu olarak tanımlanan "kapalı-çevrim marjini" kavramı önerilmiştir. Böylece, kontrolör bloğundaki DVR'nin parametreleri, kapalı çevrim izleme hatasıyla optimize edilebilmektedir. Kontrol parametrelerinin uyarlanmasının kontrol edilecek sistem üzerindeki etkilerinin gözlemleyebilmek için, ikinci bir

DVR yapısı kullanılmıştır. Uyarlama mekanizmasının kararlılığı ayrıntılı bir şekilde incelenmiştir. DVR kontrolörün başarımı, nominal, ölçme gürültüsü ve parametrik belirsizlik durumları için lineer olmayan sürekli karıştırılan reaktör (CSTR) ve lineer olmayan bioreaktör sistemleri üzerinde, simülasyonlarla incelenmiştir. Buna ek olarak, DVR kontrolörün ve [5]'de önerilen DVR tabanlı PID kontrolörün performans karşılaştırmaları gerçekleştirilmiştir. Sonuçlar, DVR kontrolörün, düşük izleme hatası elde etmede, ölçme gürültüsü ve bozucuları bastırmada başarılı kontrol performansına sahip olduğunu göstermektedir.

Üçüncü olarak önerilen yapıda, lineer olmayan sistemler için, DVR tabanlı, uyarlanabilecek parametreye sahip herhangi bir kontrolöre uygulanabilecek, genelleştirilmiş öz-uyarlamalı regülatör yapısı tasarlanmıştır. Önerilen mekanizmanın yeniliği, kontrolör parametrelerini kestirebilmek için, DVR'nin ilk defa doğrudan doğruya parametre kestirici olarak kullanılmasıdır. Bu amaçla, ikinci önerilen mekanizmadaki "kapalı çevrim marjini" kavramı, öz-uyarlamalı regülatörler için tekrar düzenlenmiştir. Uyarlama mekanizması, sistemin davranışının değişimini kestirmek için kullanılan çevrimiçi DVR sistem modeli, uyarlanabilir parametreler içeren bir uyarlamalı kontrolör ve ayrı bir çevrim içi DVR ile gerçekleştirilen parametre kestiriciden oluşmaktadır. Genelleştirilmiş uyarlamalı kontrol yapısının ve parametre kestiricinin performans başarımı, kontrolör bloğunda ayrı ayrı PID ve bulanık PID kontrolörler kullanılarak, lineer olmayan bioreaktör sistemi üzerinde gerçekleştirilmiştir. Kontrolörün kararlılık analizi yapılmış ve kontrolörün başarımı [5]'de önerilen DVR tabanlı PID kontrolör ile karşılaştırılmıştır. Kontrolörlerin dayanıklılığı, sistemin nominal koşullarda çalışması durumunda, sistemin çıkışına ölçme gürültüsü eklenmesi durumunda ve parametrik belirsizlik durumlarında incelenmiştir. Simülasyon sonuçları, önerilen yapının, ölçme gürültüsü ve parametrik belirsizlik bastırmadaki etkinliğini göstermektedir.



1. INTRODUCTION

Adaptation is a substantial competency of living organisms which increases the resistance of species to adverse environmental conditions and provides them to transfer their genetic information to future generations [6]. The species which cannot accomplish adaptation are faced with danger of extinction. Adaptation exists almost in every area of life from the evolution of the species to human relations. Whereas people who are capable of adaptation to different environments are quickly accepted by the society and are socially successful, the bacteria adapting to the antibiotics acquire resistance against them. In many engineering fields, a variety of solutions which converge to an optimal value iteratively can be computed to solve problems by imitating the adaptation features of living organisms. In control engineering, robust controller structures which provide persistence of control processes in spite of changing conditions and disturbances have been proposed inspired by the adaptation features of biological systems. Once a nonadaptive controller is designed to control a specific system, it will provide acceptable performance only for that system for a particular operating condition, but can not provide acceptable system behaviour in all situations especially for nonlinear systems since the controller has rigid structure due to its fixed controller parameters [3, 7]. Therefore, the necessity for adaptive flexible controller structures emerges as the environment generally involves continuously changing dynamics. In order to overcome strong nonlinearities, time-delays and time varying dynamics of systems, flexibility can be acquired to conventional controller topologies by introducing adaptation. In this sequel, a controller which can modify its structure or behaviour depending on internal/external factors is called as an adaptive controller.

Adaptive controllers can be roughly classified under two main headings: model-free adaptive controllers and model based adaptive controller. In model-free adaptive (MFA) control, it is explicit that the adaptive control methodology does not require any system model and system identification phase [8].

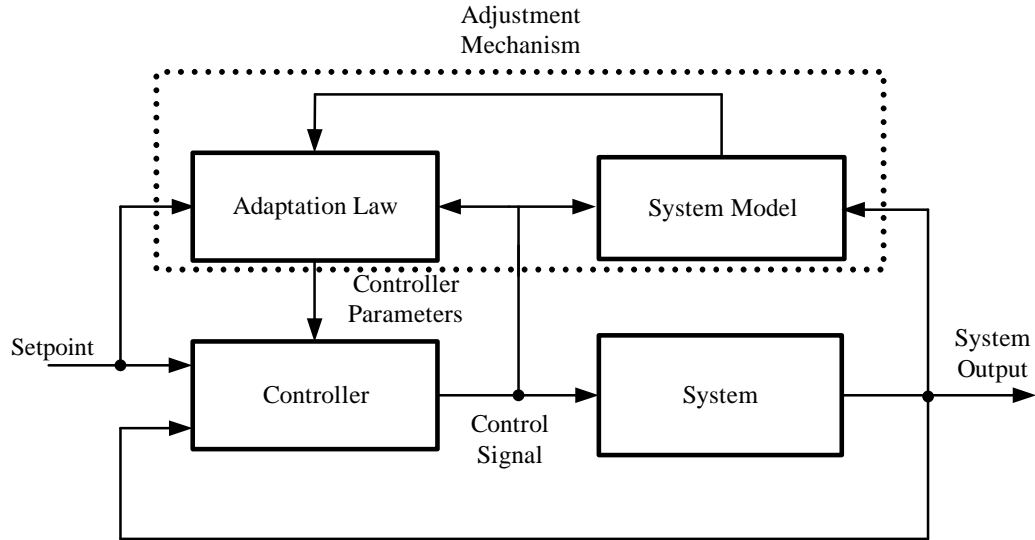


Figure 1.1 : Block diagram of model based adaptive control systems.

In model based adaptive control, identification of system model is required in order to interfuse adaptation ability to controller by approximating the future behaviour of the system . Hence, the system identification and controller design are aggregated [9]. A block diagram of a model based adaptive control system is illustrated in Figure 1.1. As depicted in Figure 1.1, in model based adaptive control the aim is to force system output to track reference signal by updating the controller parameters according to an adaptation law taking into account the future response of the system dynamics. Various model based adaptive controllers can be proposed depending on the chosen controller, system model and adaptation law. The accuracy of the model is crucial to accurately estimate the system behaviour and obtain proper direction vector to adjust controller parameters. The model based adaptive controllers are generally based on good representation of the systems dynamics. For this purpose, owing to their powerful modeling capacity, intelligent methods such as artificial neural networks (ANN), adaptive neuro-fuzzy inference systems (ANFIS) and support vector regression (SVR) have frequently been preferred to identify the dynamics of nonlinear systems.

Support Vector Regression (SVR) , first asserted by Vapnik et al. [10–12], is one of the most effective regression techniques. The main strength of SVR is that the non-convex primal form of the optimization problem can be converted to a new dual convex form in which global extremum is ensured. SVR based controller structures have been preferred to ANN and ANFIS in recent years since the ANN and ANFIS system models are only valid locally due to the their non-convex objective functions.

Various control architectures based on SVR have been proposed in technical literature. The common feature of these adaptive structures is that SVR is utilized to model system dynamics or approximate system Jacobian in order to tune the parameters of the adaptive controllers via derivative based optimization algorithms.

In this PhD thesis, it has constitutively been aimed to design an SVR based novel adaptation mechanism where SVR is directly utilized as controller or parameter estimator. Thus, three novel adaptive control mechanisms based on SVR have been proposed for nonlinear systems. These are:

- Adaptive NARMA-L2 Controller based on Support Vector Regression
- Adaptive Support Vector Regressor Controller
- Generalized Self-Tuning Regulator based on Support Vector Regression

In the first control architecture [13], the strong modelling capability of SVR and the functionality of the NARMA-L2 controller structure are merged and a novel online SVR based NARMA-L2 controller for nonlinear dynamical systems has been proposed. It has been aimed to acquire the parameters of NARMA-L2 submodels via the previously obtained NARX model of the system, and then NARMA-L2 controller is designed via the obtained NARMA-L2 submodels. In order to accomplish conversion from NARX model to NARMA model, conversion parameters are utilized. The conversion parameters are optimized via Levenberg-Marquard optimization algorithm by taking into account K- step ahead future behaviour of the system. The performance evaluation of the control system has been performed on a nonlinear bioreactor system. The robustness of the mechanism has been tested for measurement noise and parametric uncertainty cases. The performance of the proposed controller is compared with an SVM-based PID controller proposed in [5]. The results indicate that the proposed mechanism can be successfully employed to control nonlinear dynamical systems.

In the second control architecture [3], SVR is directly utilized as a controller for the first time in technical literature. In order to provide adaptation of the controller parameters without necessarily any prior information on controller output, "closed-loop margin" notion which is defined as the margin between reference input

and system output has been proposed. Thus, the SVR parameters in controller block are optimized via closed-loop tracking error. A second online SVR is employed in order to estimate the system dynamics and observe the impacts resulting from adaptation of the controller parameters on system behaviour to be controlled. The closed-loop system stability analysis of the adaptation mechanism has thoroughly been conducted. The performance evaluation of the SVR controller has been examined by simulations performed on continuously stirred tank reactor (CSTR) and bioreactor benchmark problems for nominal case and when measurement noise and parametric uncertainty are added. In addition to this, the performance comparison of the the controller has been executed with an SVM-based PID controller proposed in [5]. The results indicate that the online SVR controller has quite succesful control performance in attaining low tracking error, suppressing measurement noise and parametric uncertainties.

In the third control architecture [6], a novel generalized self tuning regulator based on online SVR which can be implemented for any controller with adjustable parameters has been introduced for nonlinear dynamical systems. The main novelty of the proposed mechanism is the direct utilization of the SVR as a parameter estimator for the first time in order to approximate controller parameters. For this purpose, the closed-loop margin notion proposed for the second control architecture has been reconfigured for STR's. The adaptation mechanism is composed of a forward SVR model of the system which is employed to estimate future dynamical change in system behaviour, an adaptive controller involving tunable parameters and a parameters estimator block realized by separate online SVR's to estimate each tunable controller parameter. The performance evaluation of the proposed generalized adaptive control architecture and parameter estimator is assessed on a bioreactor benchmark system by employing two different controller topologies: Adaptive PID and adaptive fuzzy PID controllers. The stability analysis of the generalized structure has been conducted and the tracking performance of both controllers are compared with SVM-based PID controller proposed in [5]. The robustnesses of the controllers have been examined for the noiseless case and when measurement noise and parametric uncertainty are added. The simulation results show the effectiveness of the proposed adjustment mechanism on measurement noise and parametric uncertainty rejection competencies.

Three journal articles reporting the accomplishments of this thesis have been published in journals indexed by SCI-Expanded. In chapter 2, 3 and 4, these articles are given. Hence, the organization of the thesis is as in Figure 1.2.

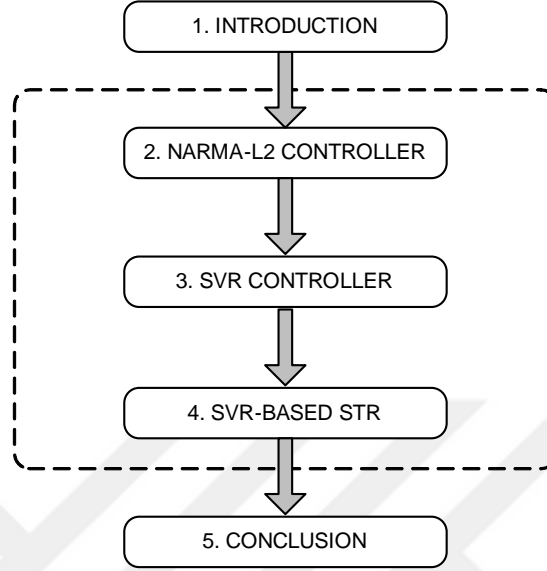


Figure 1.2 : Flow chart of the thesis.

In chapter 2, the paper titled “*A Novel Adaptive NARMA-L2 Controller based on Online Support Vector Regression for Nonlinear Systems*” (explaining in detail the first control architecture given above) which has been published online in *Neural Processing Letters* is available.

In chapter 3, the paper titled “*An Adaptive Support Vector Regressor Controller for Nonlinear Systems*” (explaining in detail the second control architecture summarized above) which has been published in *Soft Computing* is given.

In chapter 4, the paper titled “*Generalized Self-Tuning Regulator Based on Online Support Vector Regression*” (explaining in detail the third control architecture given above) which has been published online in *Neural Computing and Applications* is given.

The controller proposed in chapter 2 is structurally different from the controllers given in chapter 3 and 4. Therefore, chapter 2 can be studied independently from chapter 3 and 4. Since the controllers in chapter 3 and 4 are based on "closed-loop margin", it is preferred to view chapter 3 and 4 consecutively for better understanding. The thesis ends with a brief conclusion part in chapter 5.



2. A NOVEL ADAPTIVE NARMA-L2 CONTROLLER BASED ON ONLINE SUPPORT VECTOR REGRESSION FOR NONLINEAR SYSTEMS ¹

2.1 Introduction

The way to survival for living organisms is adaptation to the physical world. Similarly, continuation of the process in control systems depends on the adaptation skills of the controllers to internal and external factors affecting the system. In nonlinear systems, determination of the control input to cope with the nonlinear dynamics of the system is a challenging task. Therefore, representation of complex systems via simple models is crucial to design effective controllers. An ingenious model can be the key to obtain a high performance control method. Nonlinear autoregressive moving average (NARMA)-L2 controller, introduced by Narendra and Mukhopadhyay [14], is one of the most effective artificial neural network (ANN) controller architectures for nonlinear systems. The central idea of this type of controller is to extricate the control signal from nonlinear dynamics of the system model via Taylor expansion [15–17]. Thus, a nonlinear model is achieved with separate control signal term which is eluded from nonlinear inner dynamics of the system model. The controller is simply a rearrangement of neural network system model, which is trained offline in batch form [17, 18]. An important advantage of NARMA-L2 controller is that it does not require the design of extra network topology or controller training as in model reference adaptive control (MRAC). NARMA-L2 controller has been successfully utilized to control various nonlinear systems in technical literature. Majstorovic et al. [15] designed a NARMA-L2 controller for a two tank system which is derived via offline NARMA model of the system. Pedro et al. showed a realistic implementation of NARMA-L2 slip controller for an anti-lock braking system (ABS) [16]. Hagan et al. compared the performance of various controller structures based on ANNs on a continuously stirred tank reactor (CSTR), a single link robot arm and a magnetic

¹This chapter is based on the paper "Uçak K. and Günel G.Ö., (2016), A novel adaptive NARMA-L2 controller based on online support vector regression for nonlinear systems, Neural Processing Letters, doi: 10.1007/s11063-016-9500-7"

levitation system [17, 19]. They have controlled a catalytic CSTR via NARMA-L2 controller. Since NARMA-L2 controller produces an oscillatory control signal, Pukrittayakamee et al. [18] proposed to use a term which consists of scaled reference and system output in order to alleviate oscillation and chattering in control signal. Wahyudi et al. [20] smoothed NARMA-L2 controller to control a single link robot arm since nonlinearities and parametric uncertainties are exceptionally apparent in a single link manipulator. As underactuated systems are difficult to control and can be asymptotically stabilized via continuous static feedback law, Akbarimajd and Kia [21] utilized NARMA-L2 controller for a two-degree of freedom (DOF) underactuated planar manipulator. Vesselenyi et al. [22] have achieved to control the position of a pneumatic actuator via NARMA-L2 controller. The common feature of the works in literature is utilization of offline training phase to obtain NARMA-L2 model of the system [14–22].

The controller performance in model based adaptive methods is directly influenced by modeling inaccuracies. Nonlinear systems can be successfully modeled via various intelligent modeling techniques such as ANN [23, 24], adaptive neuro fuzzy inference system (ANFIS) [23–25] and support vector regression (SVR) [26–28]. Since the network topologies which are trained with backpropagation algorithm (ANN and ANFIS) may get stuck at local minima resulting from their non-convex objective function, it is likely that they obtain system model only locally. In order to reduce modelling inaccuracies and improve controller performance, SVR based identification and control methods have been popularly applied in recent years due to their non-linear prediction and generalization competency. Since the objective function in SVR is convex, the gradient effects which occur in ANN and ANFIS vanish in SVR and global extremum is ensured. Therefore, SVR-based controller structures [5, 29] have recently been used as leading model based adaptive controllers instead of ANN based approach, since they yield a unique solution and possess powerful generalization ability [29, 30].

Various controller structures based on SVR have been proposed to control nonlinear systems in literature. These structures can be grouped under four main headings: adaptive PID controllers, inverse controllers, internal model controllers (IMCs) and model predictive controllers (MPCs).

Shang et al. [31] utilized online least square SVR based on sliding window in order to model Jacobian of the system which is required to obtain update rules for PID parameters via gradient descent. Since the model based on sliding window forgets the transient dynamics of the system in steady state, Zhao et al. [32] proposed to alter the size of the sliding window to improve training performance of the model and also the controller performance which depends on its Jacobian approximation competency. The kernel functions used in SVR to provide nonlinearity have design parameters which have direct efficacy on regression performance. Ucak and Oke [29] considered to tune bandwidth parameter of the kernel function simultaneously with PID controller parameters via gradient descent method. Yuan et al. [33] proposed a composite controller structure which consists of feedforward and feedback parts. The feedforward part is derived using system Jacobian which is estimated by SVR system model. Conventional PID controller is employed as feedback controller in order to interfuse robustness to the controller structure against disturbances and estimated errors, and robustness of the control structure has also been examined. Iplikci [5] offered a support vector machines (SVM)-based PID controller in which the controller parameters are adjusted via Levenberg-Marquardt algorithm by approximating K-step ahead system Jacobian. The controller is formed of five components: classical incremental PID controller, ε -SVR nonlinear autoregressive with exogenous inputs (NARX) model of the system, line search block, control signal correction block and controller parameter tuner. The Jacobian matrix utilized in parameter tuner block is coalescence of correction block which includes K-step ahead system Jacobian and a gradient vector which involves first order derivative of control signal with respect to controller parameters. Control signal correction block produces a term which is added to the computed control signal since the adjusted controller parameters may not be adequate to compel the system output to track desired reference. The control signal correction term is derived via Taylor approximation of the control signal which includes correction term. Golden section method has been employed in line search block to determine optimum learning rate for the correction term. The main characteristics of these works is that SVR is utilized to model both the system and the Jacobian. Unlike previously mentioned studies, Takao et al. [34] used a support vector classifier (SVC) as a swithing structure in a decision tree framework to assign suitable controller parameters according to the operating point of the system. The

whole operating region is split into small uncertainty ranges via a priori information. Robust PID controllers for each small range are constructed. Then, a single SVC is designed for each small uncertainty range to determine whether the current operating conditions of the system belong to that range or not. By combining all SVCs, a main decision tree which covers the whole operating range of the system is constituted. Based on the final decision, the appropriate controller for the current state of the system is deployed.

The main aim in control theory is to design a controller which identifies the inverse dynamics of the system to be controlled as closely as possible. For this purpose, inverse controllers which mimic inverse dynamics of the system can be designed using data sampled approaches. The main stalemate of inverse controllers is that there may be no unique inverse controller, moreover, there is no guarantee for the existence of the system inverse. Liu et al. [35] employed an SVR to identify inverse dynamics of a nonlinear system and designed an inverse controller based on this model. The control signal error which is also the approximation error of SVR is compensated via a PID controller. Wang et al. [36] used an online SVR to design an inverse controller for a nonlinear system to be controlled. Yuan et al. [37] utilized two SVR structures as the inverse controller and forward model of the system. The parameters of controller and system model are optimized online via backpropagation algorithm. The stability and convergence analysis of the controller and model are given in order to guarantee convergence and fast learning.

Another model based control technique for nonlinear systems is nonlinear internal model control (NIMC). NIMC is frequently utilized as a nonlinear control method owing to its disturbance rejection capabilities [38] and robustness properties [39]. The drawbacks of NIMC are that its implementation is convenient, it is restricted only for open-loop stable systems [40] and it is based on the assumption that the system is reversible. Since the determination of system model is the most significant stage of the design [38], NIMC performance depends on accurate modelling. For this purpose, powerful generalization and modeling capacities of SVR have been combined with IMC in [38, 39, 41, 42]. Zhao et al. [38] employed LS-SVR to avoid complex inverse controller. Sun and Song [41] suggested an adaptive IMC for nonlinear systems by combining LS-SVR with sequential minimal optimization (SMO) based pruning

algorithm. Wang and Yuan [42] derived an approximate inverse control law via Taylor expansion of NARMA model where SVR is used to approximate system Jacobian and they also analyzed the stability of the proposed control law.

Model predictive control (MPC) can successfully be used to handle systems with large time delay, non-minimum phase properties or unstable dynamics. MPC provides very robust schemes [43, 44] to control highly nonlinear dynamical systems compared to PID controllers and can cope with structural changes [44–48]. All kinds of MPC are based on the same tactics: using the future predictions of the system model, a set of future control signals is obtained by solving, at each sampling instant, a finite-horizon open-loop optimal control problem and then the first element of the control sequence is applied to the system [43, 44]. In MPC, it is aimed to update the set of control signals so that the system will follow the reference signal closely by minimizing an objective function which includes future approximated tracking errors and estimated control signals. For this purpose, a direction vector should be determined to update the control signal vector. This direction vector is commonly composed of first and second order derivatives of system output with respect to control inputs which are estimated via intelligent model of the system. Iplikci [43, 44], Du and Wang [49] and Shin et al. [50] deployed SVR based system models in MPC in order to overcome modelling inaccuracies of highly non-linear systems. The performance of the MPC may devolve if the model of the controlled system cannot be accurately computed. Iplikci [44] and Du and Wang [49] proposed to utilize online SVR system model in MPC framework in order to cope with modelling inaccuracies resulting from disturbances or varying dynamics. Shin et al. [50] proposed to update the parameters of the previously offline trained SVR model via gradient descent algorithm to prevent deterioration in SVR system model.

In this paper, the strong modelling capability of SVR is combined with the functionality of the NARMA-L2 controller structure to propose a novel online SVR based NARMA-L2 controller for nonlinear single input single output (SISO) dynamical systems. The NARMA controller in general consists of two nonlinear submodels which are independent from current control signal applied to the system. If ANN is used to estimate these submodels, a subnetwork should be designed for each of them. The parameters of these subnetworks in NARMA based on ANN can

be easily adjusted without knowing the exact outputs of the submodels since update rules for each parameter of the subnetworks can be derived by using backpropagation algorithm and chain rule. However, estimation techniques based on ANN have a major drawback, they are likely to get stuck at local minima and the models attained by ANN will be valid only locally. This is the main rationale behind our choosing SVR to estimate NARMA-L2 submodels. The primary advantage of SVR compared to backpropagation based identification methods is that global extremum is assured, hence the system model prevalent in all regions is obtained precisely. Nevertheless, if NARMA submodels are estimated by SVR instead of ANNs, there is a major difficulty. To obtain the optimum mapping function in SVR the input-output training pairs must be available, however outputs of NARMA submodels are not known a priori, so estimating NARMA model with SVR is a hard task. In this paper, we overcome this problem by deriving the NARX model of the system first, and then obtaining NARMA submodels via NARX model of the system. Thus, it is achieved to design NARMA-L2 controller using NARMA-L2 model of the system.

The main contribution of the paper is to convert an online SVR model of the system to an online NARMA-L2 controller, without necessarily any prior information on submodel outputs. The parameters which are utilized in conversion are optimized via Levenberg-Marquardt algorithm by considering K-step ahead future behaviour of the system. The performance of the proposed SVR controller has been evaluated by simulations carried out on a bioreactor benchmark system. Robustness of the proposed controller has been examined by adding measurement noise and parametric uncertainty to the system. The performance of the controller has been compared with an SVM-based PID controller proposed by Iplikci [5]. The results indicate that the online SVR NARMA-L2 controller together with online SVR model attain good modelling and control performances.

The paper is organized as follows: Section 2.2 describes the basic principles of NARMA-L2 model and controller. Online ε -SVR is summarized in section 2.3. In section 2.4, the proposed control architecture is explained and the optimization problem to derive SVR submodels is given. In section 2.5, simulation results and performance analysis of the controller are given together with an assessment of real

time applicability. Also, the proposed method is compared with an SVM-based PID controller. The paper ends with a brief conclusion in Section 2.6.

2.2 NARMA-L2 Model and Controller

NARMA-L2 model represents the dynamics of a nonlinear system exactly in a neighborhood of the equilibrium state [14]. The output of the NARMA-L2 model is as follows:

$$\hat{y}_{\text{NARMA } n+d} = \hat{f}_n + \hat{g}_n u_n \quad (2.1)$$

where $\hat{f}_n = F_n(\mathbf{x}_n)$ and $\hat{g}_n = G_n(\mathbf{x}_n)$ are two nonlinear functions to be approximated and $\mathbf{x}_n = [u_{n-1} \cdots u_{n-n_u}, y_n \cdots y_{n-n_y+1}]^T$ stands for the current input feature vector of the model where n_u and n_y emblematis the number of the past control inputs and system outputs included in the feature vector, d indicates the relative degree. F_n and G_n denote the submodels computed by some numerical or intelligent estimator.

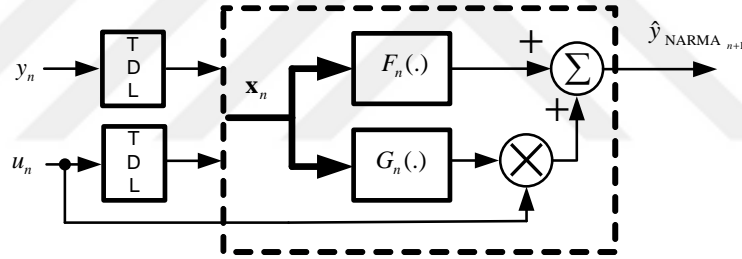


Figure 2.1 : NARMA model.

The control signal u_n is successfully extricated from the nonlinearities in the model [15, 19, 51]. The main superiority of this form is that it allows us to solve for the required control input which forces the system output to track the desired reference signal [15, 51]. As can be seen from (2.1), the next control signal u_n is separated from nonlinear dynamics. Thus, the control signal forcing the system to follow the desired behaviour can be obtained via \hat{f}_n and \hat{g}_n sub-models. In ANN based NARMA controllers, design of controller consists of two steps. The first step is to identify the dynamics of the system to be controlled by training $F_n(\cdot)$ and $G_n(\cdot)$ submodels. Second step is to design NARMA controller and approximate u_n using $F_n(\cdot)$ and $G_n(\cdot)$ trained at the previous step. The structure of the NARMA-L2 model for $d = 1$ is illustrated in Figure 2.1. The parameters of submodels $F_n(\cdot)$ and $G_n(\cdot)$ are optimized to minimize

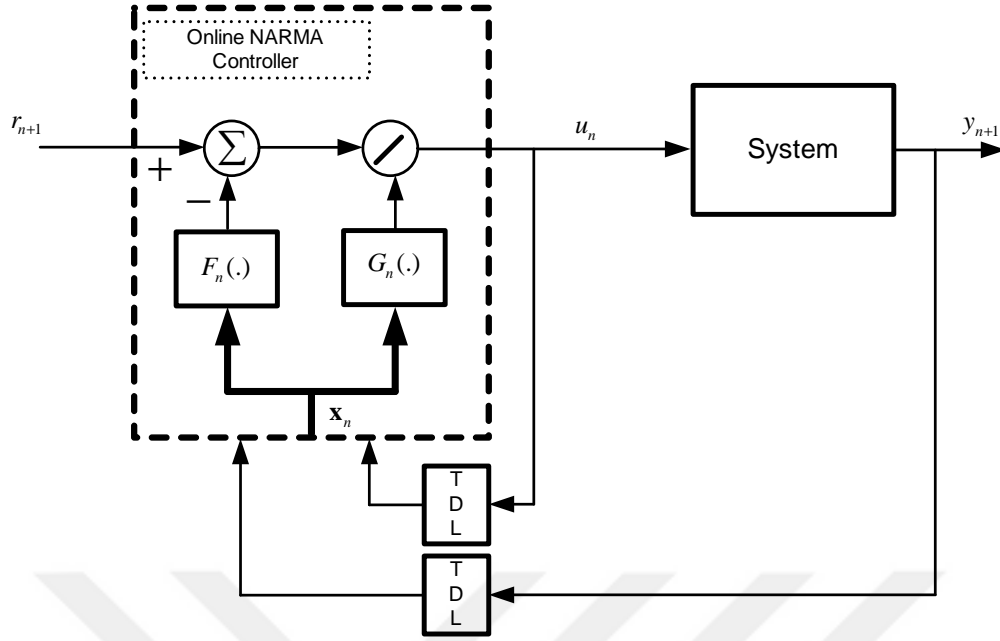


Figure 2.2 : NARMA controller.

the error between system and model outputs. After identifying the dynamics of the system as in model given in Figure 2.1, NARMA-L2 controller can be designed by substituting reference signal (r_{n+1}) in place of system output (y_{n+1}) since the system output is compelled to follow the reference signal. The control signal produced by the NARMA-L2 controller is obtained as follows:

$$u_n \cong \frac{r_{n+1} - \hat{f}_n}{\hat{g}_n} \quad (2.2)$$

where $\hat{f}_n = F_n(\mathbf{x}_n)$ and $\hat{g}_n = G_n(\mathbf{x}_n)$ are estimated outputs of submodels [19, 52, 53]. The block diagram of NARMA-L2 controller is illustrated in Figure 2.2. The advantage of NARMA-L2 controller is that it is simple to implement and it requires only two submodels of the system to be controlled without a separate controller [14, 21, 52]. However, the control input computed by NARMA-L2 controllers is typically oscillatory and this is a major drawback of this method [19]. Another weakness is the relatively complicated structure of NARMA-L2 compared to NARX model since two separate subblocks are used [52]. Also, if the inverse model is locally unstable or near stability margin the design of NARMA controller is not possible [52].

2.3 Online ε - Support Vector Regression

In regression, the aim is to approximate a function from a set of sample data. SVR, first asserted by Vapnik et al. [10–12], is one of the most effective regression techniques amongst data sampled modeling methods since it ensures global extremum. Let us consider a training data set:

$$\mathbf{T} = \{\mathbf{x}_i, y_i\}_{i=1}^N \quad \mathbf{x}_i \in \mathbf{X} \subseteq R^n, y_i \in R \quad (2.3)$$

where \mathbf{x}_i is the i th input data and y_i is the corresponding output value, N is the size of the training data and n is the dimension of the input space. The regression function (\hat{y}_i) which represents the relationship among training data set in (2.3) can be characterized via SVR model in (2.4):

$$\hat{y}_i = \mathbf{w}\Phi(\mathbf{x}_i) + b, \quad i = 1, 2, \dots, N \quad (2.4)$$

where \mathbf{w} is the weight vector of the regressor in feature space \mathbf{F} ; $\Phi(\mathbf{x}_i)$ is the image of input data in \mathbf{F} and b is the bias term [43].

In ε -SVR, an optimal regression surface which represents the given training data as accurately as possible is obtained. The optimal regressor is searched within a predefined ε -tube characterized with Vapnik's ε -insensitive loss function. The primal form of the optimization problem is described as follows [1, 43]:

$$\min_{\mathbf{w}, b, \xi, \xi^*} P = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) \quad (2.5)$$

with the following constraints constituted via Vapnik's ε -insensitive loss function

$$\begin{aligned} y_i - \mathbf{w}\Phi(\mathbf{x}_i) - b &\leq \varepsilon + \xi_i \\ \mathbf{w}\Phi(\mathbf{x}_i) + b - y_i &\leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0, \quad i = 1, 2, \dots, N \end{aligned} \quad (2.6)$$

where P indicates primal objective function, ε is the upper value of tolerable error, ξ 's and ξ^* 's are the slack variables representing the deviation from ε tube [1, 43]. Since the objective function in (2.5) is non-convex with respect to primal variables $(\mathbf{w}, b, \xi, \xi^*)$ and the training algorithm may get stuck at local minima, it is difficult to determine the optimal regressor parameters. Therefore, a Lagrangian function which

enables to convert the optimization problem to a new form where global minima is guaranteed can be derived via Lagrangian multiplier method. The Lagrangian, which is the combined version of primal objective function and its constraints, can be comprised as follows [1,43]:

$$L_P = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) + \sum_{i=1}^N \beta_i (y_i - \mathbf{w}\Phi(\mathbf{x}_i) - b - \varepsilon - \xi_i) + \sum_{i=1}^N \beta_i^* (\mathbf{w}\Phi(\mathbf{x}_i) + b - y_i - \varepsilon - \xi_i^*) + \sum_{i=1}^N (-\eta_i \xi_i - \eta_i^* \xi_i^*) \quad (2.7)$$

by inserting non-negative Lagrange multipliers β , β^* , η and η^* . First order optimality conditions for Lagrangian (L_P) are given as

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \longrightarrow \mathbf{w} - \sum_{i=1}^N \beta_i \Phi(\mathbf{x}_i) = 0 \quad (2.8)$$

$$\frac{\partial L_P}{\partial b} = 0 \longrightarrow \sum_{i=1}^N (\beta_i - \beta_i^*) = 0 \quad (2.9)$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 \longrightarrow C - \beta_i - \eta_i = 0, \quad i = 1, 2, \dots, N \quad (2.10)$$

$$\frac{\partial L_P}{\partial \xi_i^*} = 0 \longrightarrow C - \beta_i^* - \eta_i^* = 0, \quad i = 1, 2, \dots, N \quad (2.11)$$

Lagrangian (L_P) function has to satisfy the above optimality conditions at the solution. If the optimality conditions in (2.8-2.11) are substituted in (2.7), dual form of the ε -SVR optimization problem which is a quadratic problem is attained as follows:

$$D = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\beta_i - \beta_i^*)(\beta_j - \beta_j^*) K_{ij} + \varepsilon \sum_{i=1}^N (\beta_i + \beta_i^*) - y_i \sum_{i=1}^N (\beta_i - \beta_i^*) \quad (2.12)$$

subject to

$$\begin{aligned} 0 &\leq \beta_i \leq C, \\ 0 &\leq \beta_i^* \leq C, \\ \sum_{i=1}^N (\beta_i - \beta_i^*) &= 0, \quad i = 1, 2, \dots, N \end{aligned} \quad (2.13)$$

where $K_{ij} = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ is the kernel value which represents the similarity among the \mathbf{x}_i and \mathbf{x}_j training samples. The optimal values of the dual variables β_i , β_i^* in the constructed dual optimization problem formulated in (2.12-2.13) are obtained via quadratic programming (QP) solvers. The support vectors are the training data with Lagrange multipliers that have $\lambda_i = \beta_i - \beta_i^* \neq 0$ [5,43,54]. Thus, regression function

in (2.4) can be rewritten with respect to the support vectors and the corresponding Lagrange multipliers as in (2.14):

$$\hat{y}(\mathbf{x}) = \sum_{i \in SV} \lambda_i K(\mathbf{x}_i, \mathbf{x}) + b, \quad \lambda_i = \beta_i - \beta_i^* \quad (2.14)$$

The optimization problem in (2.12-2.13) is suitable for offline batch learning algorithms. In online learning, since every new added data changes the structure of the regression problem consistently, it is required to derive incremental tuning rules for model parameters by ensuring the Karush-Kuhn-Tucker (KKT) conditions. For this purpose, a new Lagrangian function which is derived from dual optimization problem (2.12-2.13) can be formulated as

$$\begin{aligned} L_D = & \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\beta_i - \beta_i^*)(\beta_j - \beta_j^*) K_{ij} + \varepsilon \sum_{i=1}^N (\beta_i + \beta_i^*) - \sum_{i=1}^N y_i (\beta_i - \beta_i^*) \\ & - \sum_{i=1}^N (\delta_i \beta_i + \delta_i^* \beta_i^*) - \sum_{i=1}^N u_i (C - \beta_i) + u_i^* (C - \beta_i^*) - \sum_{i=1}^N z (\beta_i + \beta_i^*) \end{aligned} \quad (2.15)$$

Thus, KKT optimality conditions for L_D are obtained in (2.16) via first order necessary optimality conditions [4, 55–58]:

$$\begin{aligned} \frac{\partial L_D}{\partial \beta_i} &= \sum_{j=1}^N (\beta_j - \beta_j^*) K_{ij} + \varepsilon - y_i - \sum_{i=1}^N \delta_i + \sum_{i=1}^N u_i + z = 0 \\ \frac{\partial L_D}{\partial \beta_i^*} &= - \sum_{j=1}^N (\beta_j - \beta_j^*) K_{ij} + \varepsilon + y_i - \sum_{i=1}^N \delta_i^* + \sum_{i=1}^N u_i^* - z = 0 \\ \delta_i^{(*)} &\geq 0, u_i^{(*)} \geq 0, \delta_i^{(*)} \beta_i^{(*)} = 0, u_i^{(*)} (C - \beta_i^{(*)}) = 0 \end{aligned} \quad (2.16)$$

According to KKT conditions, at most one of the β_i and β_i^* should be nonzero and both are nonnegative [4]. KKT conditions allow the reformulation of SVM for regression by dividing the whole training data set \mathbf{T} into the three main subsets: Error support vectors (\mathbf{E}), support vectors (\mathbf{S}) and remaining samples (\mathbf{R}) which are classified depending on their Lagrange and margin values in (2.17) [55].

$$\begin{aligned} \mathbf{E} &= \{i \mid |\lambda_i| = C, |h(\mathbf{x}_i)| \geq \varepsilon\} \\ \mathbf{S} &= \{i \mid 0 < |\lambda_i| < C, |h(\mathbf{x}_i)| = \varepsilon\} \\ \mathbf{R} &= \{i \mid |\lambda_i| = 0, |h(\mathbf{x}_i)| \leq \varepsilon\} \end{aligned} \quad (2.17)$$

Using the function to be approximated and the SVR, error margin can be defined as follows:

$$h(\mathbf{x}_i) = f(\mathbf{x}_i) - y_i = \sum_{j=1}^N \lambda_j K_{ij} + b - y_i \quad (2.18)$$

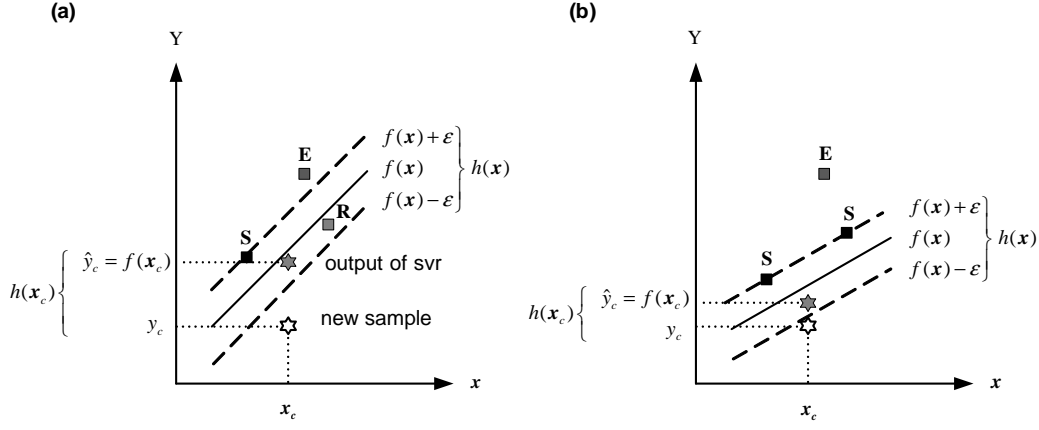


Figure 2.3 : E,S and R subsets before (a) and after (b) training

Convergence conditions for online SVR algorithm are given in (2.19) depending on KKT conditions (2.16) and error margin function (2.18)

$$\begin{aligned}
 h(\mathbf{x}_i) &\geq \varepsilon, \lambda_i = -C \\
 h(\mathbf{x}_i) &= \varepsilon, -C < \lambda_i < 0 \\
 -\varepsilon &\leq h(\mathbf{x}_i) \leq \varepsilon, \lambda_i = 0 \\
 h(\mathbf{x}_i) &= -\varepsilon, 0 < \lambda_i < C \\
 h(\mathbf{x}_i) &\leq -\varepsilon, \lambda_i = C
 \end{aligned} \tag{2.19}$$

The geometrical interpretation of incremental learning and these three subsets are visualized in Figure 2.3. In the incremental online SVR algorithm, when a new training data \mathbf{x}_c is received, its corresponding λ_c value is initially set to zero, which is later updated to $\Delta\lambda_c$ [43]. Then, the largest possible change $\Delta\lambda_c$ is calculated while at the same time keeping the system at the equilibrium state with respect to the new KKT conditions [43]. Depending on $\Delta\lambda_c$, the bias and the Lagrange multipliers of the samples in support set (S) are adjusted. Detailed information related to recursive algorithm can be achieved via [4, 55, 59].

2.4 NARMA Controller based on Online SVR

2.4.1 Identification of the NARMA model by SVR

The first step in the proposed control method is to obtain a NARMA model of the system. Initially, from the available input-output data, a NARX model is easily computed. In order to design a NARMA-L2 controller, the dynamics of the NARX

model should be decomposed into a NARMA model. Therefore two separate SVRs are designed as depicted in Figure 2.4. SVR_{NARX} calculates the NARX model of the system from the input-output dataset of the system, then $SVR_{NARMA-L2}$ decomposes this model to a NARMA model in order to design a convenient $SVR_{NARMA-L2}$ controller.

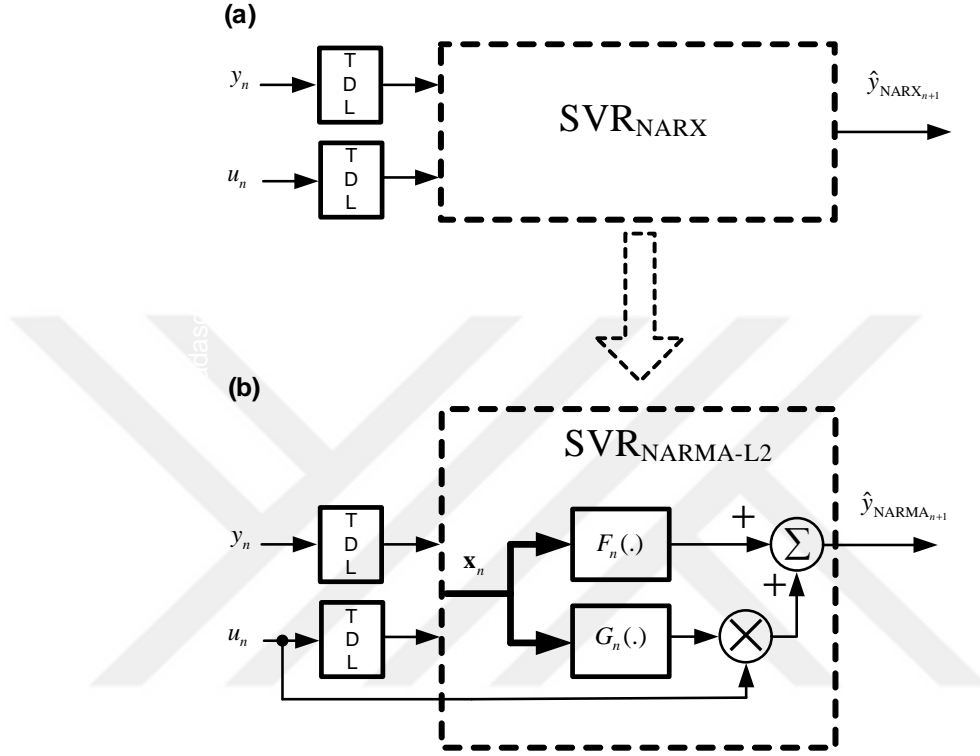


Figure 2.4 : Decomposition of SVR_{NARX} model (a) to $SVR_{NARMA-L2}$ model (b).

We have used online SVR for modelling so that the model and the controller can be adapted in accordance with the changing dynamics of the system. Since the parameters of the model are updated in online manner, the regression functions of submodels $F(\cdot)$ and $G(\cdot)$ may alter as the number of discrete time steps n proceeds, therefore $F(\cdot)$ and $G(\cdot)$ are symbolized as $F_n(\cdot)$ and $G_n(\cdot)$ as given in Figure 2.4. The greatest difficulty of SVR based modelling is the necessity of input-output data pairs. In ANN based modeling, the weights related to $F_n(\cdot)$ and $G_n(\cdot)$ subnetworks can be tuned simply using backpropagation algorithm and chain rule, even if the output of $F_n(\cdot)$ and $G_n(\cdot)$ are not known exactly. In SVR, input-output data pairs must be available to find the optimal regression function. Since the outputs of $F_n(\cdot)$ and $G_n(\cdot)$ are not known exactly, SVR submodels for $F_n(\cdot)$ and $G_n(\cdot)$ cannot be designed separately. Therefore, the focus in this study is to drive model parameters of $SVR_{NARMA-L2}$ from

the previously obtained SVR_{NARX} model of a system. To achieve this, firstly a main SVR_{NARX} model is designed as in Figure 2.4 (a), we call this as the "main model", then submodels given in Figure 2.4 (b) are decomposed depending on the main model. In $\text{SVR}_{\text{NARMA-L2}}$ controller, the aim is to find u_n . However as can be seen in (2.2), at time step n the model parameters of $F_n(\cdot)$ and $G_n(\cdot)$ are not known. Therefore, the outputs of these submodels can be approximated depending on the model parameters obtained at the previous step ($n - 1$). Hence, the output of the submodels at prediction phase can be expressed as follows:

$$\begin{aligned}\hat{f}_n^- &\cong F_{n-1}(\mathbf{x}_n) \\ \hat{g}_n^- &\cong G_{n-1}(\mathbf{x}_n)\end{aligned}\tag{2.20}$$

where the subscript stands for the time index of the state vector and the superscript is used to indicate whether the system model derived in time step n or $n - 1$ is used. More specifically, "-" in (2.20) means that the models computed in the previous time step $n - 1$, are evaluated with the current state vector \mathbf{x}_n to obtain \hat{f}_n^- and \hat{g}_n^- . Similarly, "+" in \hat{f}_n^+ and \hat{g}_n^+ indicates that the model obtained in current step (n) has been utilized together with the current state vector \mathbf{x}_n as follows:

$$\begin{aligned}\hat{f}_n^+ &\cong F_n(\mathbf{x}_n) \\ \hat{g}_n^+ &\cong G_n(\mathbf{x}_n)\end{aligned}\tag{2.21}$$

In order to compute the control signal (u_n), the parameters of $F_n(\cdot)$ and $G_n(\cdot)$ submodels must be obtained. Therefore, the following decomposition step is crucial to derive these parameters.

Modeling and decomposition step at time step n-1 : Consider the SVR_{NARX} model of the system depicted in Figure 2.4 (a). Regression function for this model is given by:

$$\hat{y}_{\text{NARX}_n} = \sum_{i=1}^N \beta_i K(\mathbf{x}_i, \mathbf{x}_c) + b_\beta\tag{2.22}$$

Since this model is not directly usable in feedback linerization, the main model (SVR_{NARX}) should be divided to submodels as in Figure 2.4 (b). Let us assume that the the corresponding regression function for $F_{n-1}(\mathbf{x}_{n-1})$ and $G_{n-1}(\mathbf{x}_{n-1})$ are as follows:

$$\begin{aligned}\hat{f}_{n-1}^+ &\cong F_{n-1}(\mathbf{x}_{n-1}) = \sum_{i=1}^N \alpha_i K(\mathbf{x}_i, \mathbf{x}_{n-1}) + b_f \\ \hat{g}_{n-1}^+ &\cong G_{n-1}(\mathbf{x}_{n-1}) = \sum_{i=1}^N \theta_i K(\mathbf{x}_i, \mathbf{x}_{n-1}) + b_g\end{aligned}\tag{2.23}$$

Thus, output of the SVR_{NARMA-L2} model is obtained as:

$$\begin{aligned}
\hat{y}_{\text{NARMA}_n} &= F_{n-1}(\mathbf{x}_{n-1}) + G_{n-1}(\mathbf{x}_{n-1})u_{n-1} \\
&= \sum_{i=1}^N \alpha_i K(\mathbf{x}_i, \mathbf{x}_{n-1}) + b_f + \left[\sum_{i=1}^N \theta_i K(\mathbf{x}_i, \mathbf{x}_{n-1}) + b_g \right] u_{n-1} \\
&= \sum_{i=1}^N \left[\alpha_i + \theta_i u_{n-1} \right] K(\mathbf{x}_i, \mathbf{x}_{n-1}) + b_f + b_g u_{n-1}
\end{aligned} \tag{2.24}$$

If SVR_{NARMA-L2} and SVR_{NARX} models are matched in (2.25), a relation between submodels of SVR_{NARMA-L2} and SVR_{NARX} can be attained as in (2.26).

$$\begin{aligned}
\hat{y}_{\text{NARX}_n} &\cong \hat{y}_{\text{NARMA}_n} \\
\sum_{i=1}^N \beta_i K(\mathbf{x}_i, \mathbf{x}_{n-1}) + b_\beta &= \sum_{i=1}^N \left[\alpha_i + \theta_i u_{n-1} \right] K(\mathbf{x}_i, \mathbf{x}_{n-1}) + b_f + b_g u_{n-1}
\end{aligned} \tag{2.25}$$

$$\begin{aligned}
\beta_i &= \alpha_i + \theta_i u_{n-1} \\
b_\beta &= b_f + b_g u_{n-1}
\end{aligned} \tag{2.26}$$

The following assumption has been proposed to utilize this relation effectively and to approximate the parameters of the submodels by this approach.

Assumption: Let us assume that the following relations exist between submodels.

$$\begin{aligned}
\alpha_i &= \mu_1(\cdot) \theta_i \\
b_f &= \mu_2(\cdot) b_g
\end{aligned} \tag{2.27}$$

Thus, the following equality is obtained:

$$F_{n-1}(\mathbf{x}_{n-1}) = \mu_1(\cdot) G_{n-1}(\mathbf{x}_{n-1}) + [\mu_2(\cdot) - \mu_1(\cdot)] b_g \tag{2.28}$$

As can be seen from (2.28), the derived SVR_{NARMA-L2} model depends only on $G_{n-1}(\mathbf{x}_{n-1})$ for the given values of $\mu_1(\cdot)$ and $\mu_2(\cdot)$. Using (2.26) and (2.27), the bias and the Lagrange multipliers of the SVR_{NARMA-L2} submodels $F_{n-1}(\mathbf{x}_{n-1})$ and $G_{n-1}(\mathbf{x}_{n-1})$ are obtained with respect to parameters of SVR_{NARX} model as follows:

$$\begin{aligned}
\theta_i &= \frac{\beta_i}{\mu_1(\cdot) + u_{n-1}}, \quad \alpha_i = \mu_1(\cdot) \theta_i \\
b_g &= \frac{b_\beta}{\mu_2(\cdot) + u_{n-1}}, \quad b_f = \mu_2(\cdot) b_g
\end{aligned} \tag{2.29}$$

Thus, outputs of the internal dynamics $(F_{n-1}(\mathbf{x}_{n-1}), G_{n-1}(\mathbf{x}_{n-1}))$ required for controller design can be calculated using main SVR_{NARX} model. The output of the derived SVR_{NARMA-L2} model is:

$$\begin{aligned}
\hat{y}_{\text{NARMA}_n} &= F_{n-1}(\mathbf{x}_{n-1}) + G_{n-1}(\mathbf{x}_{n-1})u_{n-1} \\
&= \hat{f}_{n-1}^+ + \hat{g}_{n-1}^+ u_{n-1} \\
\mathbf{x}_{n-1} &= [u_{n-2} \cdots u_{n-n_u-1}, y_{n-1} \cdots y_{n-n_y}]^T
\end{aligned} \tag{2.30}$$

This assumption is crucial in building the relationship between NARX and NARMA models and hence making it possible to obtain $\text{SVR}_{\text{NARMA-L2}}$ model from the main SVR_{NARX} model.

2.4.2 Controller design

Prediction step, Calculation of u_n at time step n : In this section, the control signal applied to the system is calculated via submodels $F_{n-1}(\cdot)$ and $G_{n-1}(\cdot)$ which are approximated using the trained SVR_{NARX} model at previous step $(n-1)$. The corresponding output of the submodels against current state vector (\mathbf{x}_n) are represented as \hat{f}_n^- and \hat{g}_n^- as given in (2.31).

$$\begin{aligned}\mathbf{x}_n &= [u_{n-1} \cdots u_{n-n_u}, y_n \cdots y_{n-n_y+1}]^T \\ \hat{f}_n^- &\cong F_{n-1}(\mathbf{x}_n) \\ \hat{g}_n^- &\cong G_{n-1}(\mathbf{x}_n)\end{aligned}\tag{2.31}$$

Thus, the control signal (u_n) produced by $\text{SVR}_{\text{NARMA-L2}}$ controller can be calculated as:

$$u_n \cong \frac{r_{n+1} - \hat{f}_n^-}{\hat{g}_n^-}\tag{2.32}$$

by means of the submodels obtained at previous step (F_{n-1}, G_{n-1}) . Then, u_n is applied to the system and y_{n+1} is computed. So, input-output data pair can be arranged for next training phase of main SVR_{NARX} model. Accordingly, SVR_{NARX} model of the system can be utilized to derive a $\text{SVR}_{\text{NARMA-L2}}$ controller law given in (2.32) without explicitly knowing the outputs of the submodels $F(\cdot)$ and $G(\cdot)$ separately. The proposed controller is illustrated in Figure 2.5 where y_{n+1} is the output of the system, r_{n+1} denotes reference signal and u_n is the control signal produced by the controller. It is anticipated that the tracking performance of the control system against varying dynamics is going to improve by using the online SVR models. The adaptive structure of the submodels has an impact on the controller performance.

2.4.3 Adaptive predictive $\text{SVR}_{\text{NARMA-L2}}$ controller

The tracking performance of the proposed controller depends on the parameters $(\mu_1(\cdot), \mu_2(\cdot))$ utilized to compute the outputs of submodels. We have employed a predictive structure which takes into account the K -step ahead future behaviour of the

system to adjust $\mu_1(\cdot)$ and $\mu_2(\cdot)$ parameters effectively. The optimization technique proposed by Iplikci for SVM-based PID controller [5] has been utilized to optimize $\mu_1(\cdot)$ and $\mu_2(\cdot)$ parameters. The objective function given in (2.33) is optimized via Levenberg-Marquardt algorithm to tune $\mu_1(\cdot)$ and $\mu_2(\cdot)$.

$$E(\boldsymbol{\mu}) = \frac{1}{2} \sum_{k=1}^K \hat{e}_{n+k}^2 + \frac{1}{2} \lambda [u_n - u_{n-1}]^2 \quad (2.33)$$

where $\hat{e}_{n+k} = r_{n+k} - \hat{y}_{n+k}$. The parameters can be adjusted using Levenberg-Marquardt algorithm as follows:

$$\begin{bmatrix} \mu_{1_{new}} \\ \mu_{2_{new}} \end{bmatrix} = \begin{bmatrix} \mu_{1_{old}} \\ \mu_{2_{old}} \end{bmatrix} + [\mathbf{J}^T \mathbf{J} + \eta \mathbf{I}]^{-1} \mathbf{J}^T \mathbf{e} \quad (2.34)$$

where

$$\begin{aligned} \mathbf{J} &= \begin{bmatrix} \frac{\partial e_{n+1}}{\partial \hat{y}_{n+1}} \frac{\partial \hat{y}_{n+1}}{\partial \mu_1} & \frac{\partial e_{n+1}}{\partial \hat{y}_{n+1}} \frac{\partial \hat{y}_{n+1}}{\partial \mu_2} \\ \vdots & \vdots \\ \frac{\partial e_{n+K}}{\partial \hat{y}_{n+K}} \frac{\partial \hat{y}_{n+K}}{\partial \mu_1} & \frac{\partial e_{n+K}}{\partial \hat{y}_{n+K}} \frac{\partial \hat{y}_{n+K}}{\partial \mu_2} \\ \frac{\partial \sqrt{\lambda} \Delta u_n}{\partial \mu_1} & \frac{\partial \sqrt{\lambda} \Delta u_n}{\partial \mu_2} \end{bmatrix} = \begin{bmatrix} -\frac{\partial \hat{y}_{n+1}}{\partial u_n} \frac{\partial u_n}{\partial \mu_1} & -\frac{\partial \hat{y}_{n+1}}{\partial u_n} \frac{\partial u_n}{\partial \mu_2} \\ \vdots & \vdots \\ -\frac{\partial \hat{y}_{n+K}}{\partial u_n} \frac{\partial u_n}{\partial \mu_1} & -\frac{\partial \hat{y}_{n+K}}{\partial u_n} \frac{\partial u_n}{\partial \mu_2} \\ \sqrt{\lambda} \frac{\partial u_n}{\partial \mu_1} & \sqrt{\lambda} \frac{\partial u_n}{\partial \mu_2} \end{bmatrix} \\ &= \begin{bmatrix} -\frac{\partial \hat{y}_{n+1}}{\partial u_n} \\ \vdots \\ -\frac{\partial \hat{y}_{n+K}}{\partial u_n} \\ \sqrt{\lambda} \end{bmatrix} \begin{bmatrix} \frac{\partial u_n}{\partial \mu_1} & \frac{\partial u_n}{\partial \mu_2} \end{bmatrix} = \begin{bmatrix} -\frac{\partial \hat{y}_{n+1}}{\partial u_n} \\ \vdots \\ -\frac{\partial \hat{y}_{n+K}}{\partial u_n} \\ \sqrt{\lambda} \end{bmatrix} \begin{bmatrix} \frac{b_g - g_n}{g_n} & \frac{-b_g}{g_n} \end{bmatrix} = \mathbf{J}_m \mathbf{J}_c \end{aligned} \quad (2.35)$$

and $\mathbf{e} = [\hat{e}_{n+1} \cdots \hat{e}_{n+K} \sqrt{\lambda} \Delta u_n]^T$. The adaptation mechanism for the controller is illustrated in Figure 2.6.

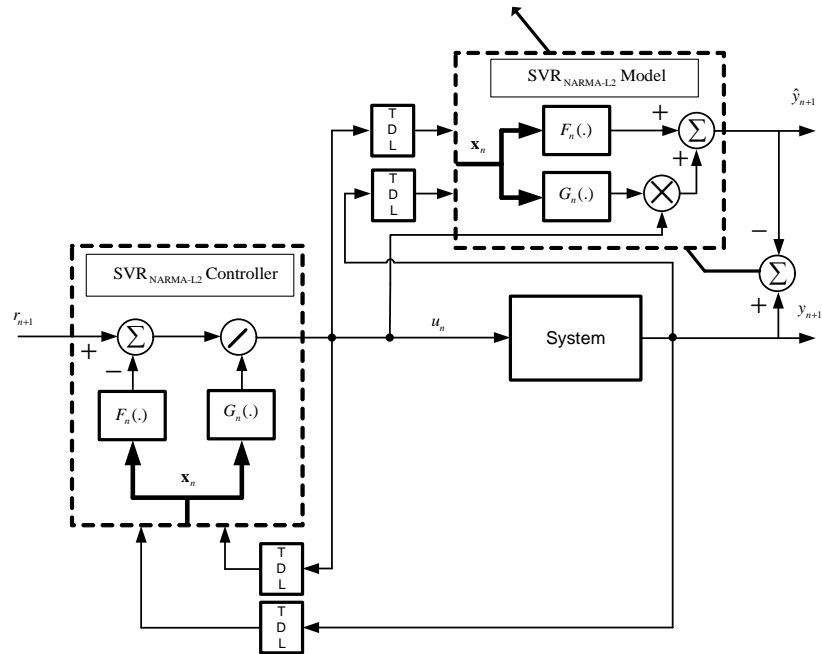


Figure 2.5 : SVR_{NARMA-L2} controller based on online SVR.

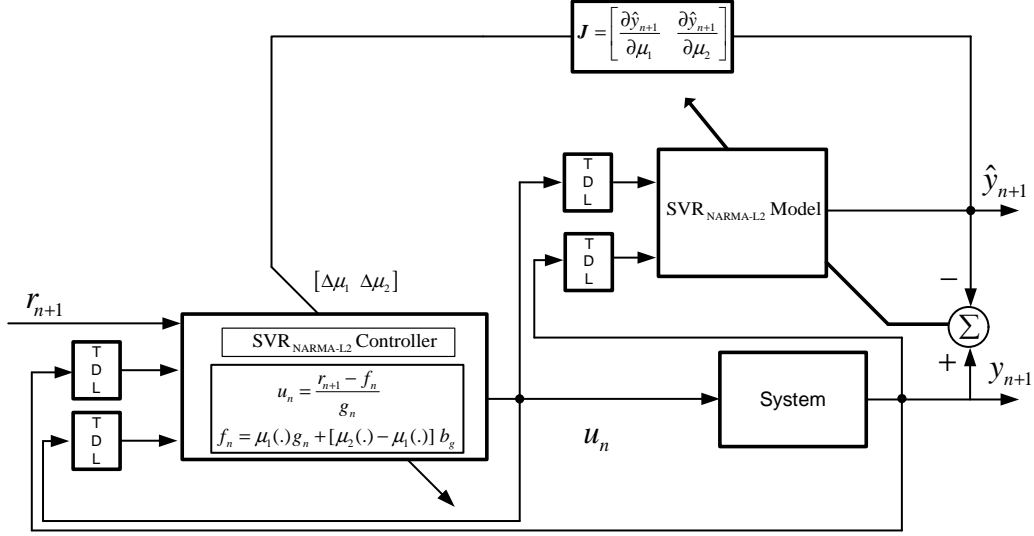


Figure 2.6 : Adaptive $\text{SVR}_{\text{NARMA-L2}}$ controller based on online SVR.

If the current state vector and j th support vector are $\mathbf{c}_n = [u_{n-1} \cdots u_{n-n_u}, y_n \cdots y_{n-n_y+1}]^T$ and $\mathbf{x}_j = [x_{j1} \cdots x_{jn_u}, x_{jn_u+1} \cdots x_{jn_u+n_y}]^T$ respectively, K -step ahead future behaviour of the system can be approximated as

$$\begin{aligned} \hat{y}_{n+k} &= f_n(\hat{\mathbf{c}}_{n+k}) + g_n(\hat{\mathbf{c}}_{n+k})u_n \\ \hat{\mathbf{c}}_{n+k} &= \underbrace{[u_n \cdots u_n]_k}_k \underbrace{[u_{n-1} \cdots u_{n+k-n_u}]_{n_u-k}}_{n_u-k} \underbrace{[\hat{y}_{n+k} \cdots \hat{y}_{n+1}]_{k-1}}_{k-1} \underbrace{[y_n \cdots y_{n+k-n_y}]_{n_y+1-k}}_{n_y+1-k} \end{aligned} \quad (2.36)$$

where

$$\begin{aligned} f_n(\hat{\mathbf{c}}_{n+k}) &= \sum_{j \in \text{SV}} \alpha_j K(d_{n+k,j}) + b_f = \sum_{j \in \text{SV}} \alpha_j \exp\left(\frac{-d_{n+k,j}}{2\sigma^2}\right) + b_f \\ g_n(\hat{\mathbf{c}}_{n+k}) &= \sum_{j \in \text{SV}} \theta_j K(d_{n+k,j}) + b_g = \sum_{j \in \text{SV}} \theta_j \exp\left(\frac{-d_{n+k,j}}{2\sigma^2}\right) + b_g \end{aligned} \quad (2.37)$$

$d_{n+k,j}$ is defined as the Euclidean distance between the state vector at time step $(n+k)$ th and j th support vector \mathbf{x}_j as:

$$d_{n+k,j} = \sqrt{(\mathbf{c}_{n+k} - \mathbf{x}_j)^T (\mathbf{c}_{n+k} - \mathbf{x}_j)} = \sqrt{D_{U_{n+k}} + D_{Y_{n+k}}} = \sqrt{A_{n+k}} \quad (2.38)$$

where

$$\begin{aligned} D_{U_{n+k}} &= \sum_{i=1}^{\min(k, n_u)} (u_n - x_{j,i})^2 \\ &+ \sum_{i=\min(k, n_u)+1}^{n_u} (u_{n+\min(k, n_u)-i} - x_{j,i})^2 \delta(n_u - k) \end{aligned} \quad (2.39)$$

$$D_{Y_{n+k}} = \sum_{i=1}^{\min(k-1, n_y)} (\hat{y}_{n+k-i} - x_{j, n_u+i})^2 + \sum_{i=\min(k-1, n_y)+1}^{n_y} (y_{n+\min(k-1, n_y)+1-i} - x_{j, n_u+i})^2 \delta(n_y + 1 - k)$$

and $\delta(\cdot)$ is unit step function. The system Jacobian is given as

$$\frac{\partial \hat{y}_{n+k}}{\partial u_n} = \frac{\partial f_n(\mathbf{c}_{n+k})}{\partial d_{n+k,j}} \frac{\partial d_{n+k,j}}{\partial u_n} + \frac{\partial g_n(\mathbf{c}_{n+k})}{\partial d_{n+k,j}} \frac{\partial d_{n+k,j}}{\partial u_n} u_n + g_n(\mathbf{c}_{n+k}) \quad (2.40)$$

where

$$\begin{aligned} \frac{\partial f_n(\mathbf{c}_{n+k})}{\partial d_{n+k,j}} &= \frac{-1}{2\sigma^2} \sum_{j \in \text{SV}} \alpha_j \exp\left(\frac{-d_{n+k,j}}{2\sigma^2}\right) \\ \frac{\partial g_n(\mathbf{c}_{n+k})}{\partial d_{n+k,j}} &= \frac{-1}{2\sigma^2} \sum_{j \in \text{SV}} \theta_j \exp\left(\frac{-d_{n+k,j}}{2\sigma^2}\right) \end{aligned} \quad (2.41)$$

and

$$\begin{aligned} \frac{\partial d_{n+k,j}}{\partial u_n} &= \frac{\partial d_{n+k,j}}{\partial A_{n+k}} \left[\frac{\partial A_{n+k}}{\partial D_{U_{n+k}}} \frac{\partial D_{U_{n+k}}}{\partial u_n} + \frac{\partial A_{n+k}}{\partial D_{Y_{n+k}}} \frac{\partial D_{Y_{n+k}}}{\partial u_n} \right] \\ \frac{\partial d_{n+k,j}}{\partial A_{n+k}} &= \frac{1}{2\sqrt{A_{n+k}}} = \frac{1}{2d_{n+k,j}} \\ \frac{\partial A_{n+k}}{\partial D_{U_{n+k}}} &= \frac{\partial A_{n+k}}{\partial D_{Y_{n+k}}} = 1 \\ \frac{\partial D_{U_{n+k}}}{\partial u_n} &= \sum_{i=1}^{\min(k, n_u)} 2(u_n - x_{j,i}) \\ \frac{\partial D_{Y_{n+k}}}{\partial u_n} &= \sum_{i=1}^{\min(k-1, n_y)} 2(\hat{y}_{n+k-i} - x_{j, n_u+i}) \frac{\partial \hat{y}_{n+k-i}}{\partial u_n} \delta(k-i) \end{aligned} \quad (2.42)$$

2.4.4 Adaptive predictive SVR_{NARMA-L2} controller with adaptive filter

As mentioned in section 2.2, NARMA-L2 controller generally produces a control signal with more oscillation and chattering compared to other controllers such as MRAC, MPC etc. [19]. Since the oscillatory control input contains high frequency components, the unmodeled high frequency dynamics of the system can be excited. In order to reduce chattering and oscillation, the control signal can be filtered [19]. The general form of the filter is given as:

$$H(z) = \frac{u_{f_n}}{u_{c_n}} = \frac{\sum_{m=0}^{d_{nu}} q_m z^{-m}}{1 + \sum_{k=1}^{d_{de}} v_k z^{-k}} \quad (2.43)$$

where v_k , $k \in \{1, \dots, d_{de}\}$ and q_m , $m \in \{0, 1, \dots, d_{nu}\}$ are "feed-backward" and "feed-forward" coefficients of the filter, and d_{de} and d_{nu} indicate the degree of the

"feed-backward" (denominator) and "feed-forward" (numerator) parts of the filter respectively. The adaptation mechanism for the controller is illustrated in Figure 2.7.

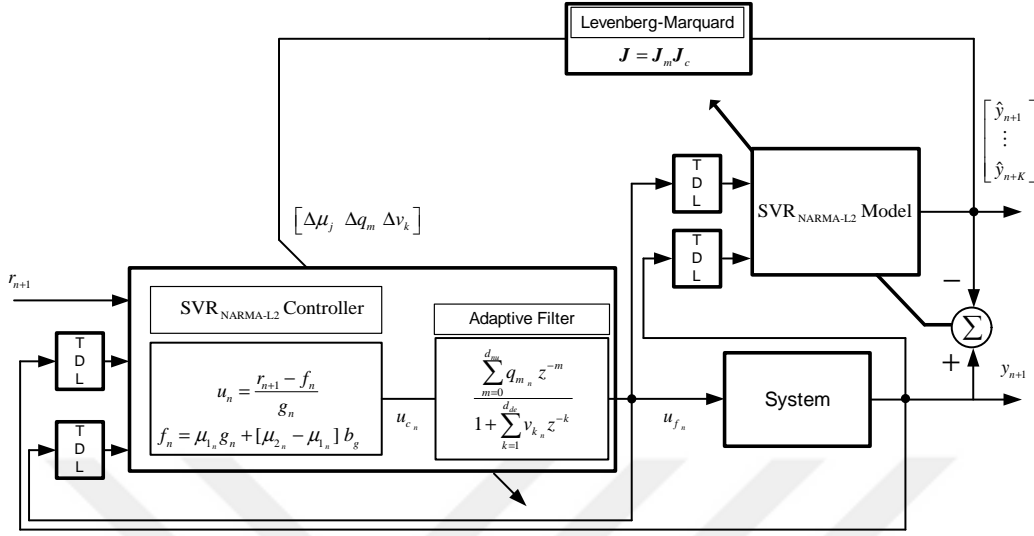


Figure 2.7 : Adaptive $SVR_{NARMA-L2}$ controller based on online SVR with adaptive filter.

The filtered control signal is computed as

$$u_{f_n} = - \sum_{k=1}^{d_{de}} v_{k_n} u_{f_{n-k}} + \sum_{m=0}^{d_{nu}} q_{m_n} u_{c_{n-m}} \quad (2.44)$$

The tracking performance of the proposed controller depends on the parameters $(\mu_j(\cdot), j \in \{1, 2\})$ used to compute output of submodels as well as the filter parameters (q_m, v_k) . A predictive structure which takes into account the K-step ahead future behaviour of system has been proposed to adjust both $\mu_j(\cdot), j \in \{1, 2\}$, $v_k, k \in \{1, \dots, d_{de}\}$ and $q_m, m \in \{0, 1, \dots, d_{nu}\}$ parameters effectively. The objective function given in (2.33) is optimized via Levenberg-Marquardt algorithm to tune parameters. The adjustable parameters can be adapted using Levenberg-Marquardt algorithm [56] as follows:

$$\begin{bmatrix} \mu_{1_{new}} \\ \mu_{2_{new}} \\ q_{0_{new}} \\ \vdots \\ q_{d_{nu_{new}}} \\ v_{1_{new}} \\ \vdots \\ v_{d_{de_{new}}} \end{bmatrix} = \begin{bmatrix} \mu_{1_{old}} \\ \mu_{2_{old}} \\ q_{0_{old}} \\ \vdots \\ q_{d_{nu_{old}}} \\ v_{1_{old}} \\ \vdots \\ v_{d_{de_{old}}} \end{bmatrix} + [J^T J + \eta I]^{-1} J^T \mathbf{e} \quad (2.45)$$

where

$$\begin{aligned}
\mathbf{J} &= \begin{bmatrix} \frac{\partial e_{n+1}}{\partial \mu_1} & \frac{\partial e_{n+1}}{\partial \mu_2} & \frac{\partial e_{n+1}}{\partial q_0} & \dots & \frac{\partial e_{n+1}}{\partial q_{dnu}} & \frac{\partial e_{n+1}}{\partial v_1} & \dots & \frac{\partial e_{n+1}}{\partial v_{dde}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{n+K}}{\partial \mu_1} & \frac{\partial e_{n+K}}{\partial \mu_2} & \frac{\partial e_{n+K}}{\partial q_0} & \dots & \frac{\partial e_{n+K}}{\partial q_{dnu}} & \frac{\partial e_{n+K}}{\partial v_1} & \dots & \frac{\partial e_{n+K}}{\partial v_{dde}} \\ \frac{\partial \sqrt{\lambda} \Delta u_{fn}}{\partial \mu_1} & \frac{\partial \sqrt{\lambda} \Delta u_{fn}}{\partial \mu_2} & \frac{\partial \sqrt{\lambda} \Delta u_{fn}}{\partial q_0} & \dots & \frac{\partial \sqrt{\lambda} \Delta u_{fn}}{\partial q_{dnu}} & \frac{\partial \sqrt{\lambda} \Delta u_{fn}}{\partial v_1} & \dots & \frac{\partial \sqrt{\lambda} \Delta u_{fn}}{\partial v_{dde}} \end{bmatrix} \\
&= \begin{bmatrix} -\frac{\partial \hat{y}_{n+1}}{\partial \mu_1} & -\frac{\partial \hat{y}_{n+1}}{\partial \mu_2} & \frac{\partial \hat{y}_{n+1}}{\partial q_0} & \dots & \frac{\partial \hat{y}_{n+1}}{\partial q_{dnu}} & -\frac{\partial \hat{y}_{n+1}}{\partial v_1} & \dots & -\frac{\partial \hat{y}_{n+1}}{\partial v_{dde}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -\frac{\partial \hat{y}_{n+K}}{\partial \mu_1} & -\frac{\partial \hat{y}_{n+K}}{\partial \mu_2} & \frac{\partial \hat{y}_{n+K}}{\partial q_0} & \dots & \frac{\partial \hat{y}_{n+K}}{\partial q_{dnu}} & -\frac{\partial \hat{y}_{n+K}}{\partial v_1} & \dots & -\frac{\partial \hat{y}_{n+K}}{\partial v_{dde}} \\ \sqrt{\lambda} \frac{\partial \Delta u_{fn}}{\partial \mu_1} & \sqrt{\lambda} \frac{\partial \Delta u_{fn}}{\partial \mu_2} & \sqrt{\lambda} u_{nn} & \dots & \sqrt{\lambda} u_{n-dnu} & \sqrt{\lambda} u_{fn-1} & \dots & \sqrt{\lambda} u_{fn-dde} \end{bmatrix} \quad (2.46)
\end{aligned}$$

$$\begin{aligned}
&= \begin{bmatrix} -\frac{\partial \hat{y}_{n+1}}{\partial u_{fn}} \\ \vdots \\ -\frac{\partial \hat{y}_{n+K}}{\partial u_{fn}} \\ \sqrt{\lambda} \end{bmatrix} \begin{bmatrix} \frac{\partial u_{fn}}{\partial u_{nn}} \frac{\partial u_{nn}}{\partial \mu_1} & \frac{\partial u_{fn}}{\partial u_{nn}} \frac{\partial u_{nn}}{\partial \mu_2} & u_{nn} & \dots & u_{n-dnu} & -u_{fn-1} & \dots & -u_{fn-dde} \end{bmatrix} \\
&= \begin{bmatrix} -\frac{\partial \hat{y}_{n+1}}{\partial u_{fn}} \\ \vdots \\ -\frac{\partial \hat{y}_{n+K}}{\partial u_{fn}} \\ \sqrt{\lambda} \end{bmatrix} \begin{bmatrix} q_{0n} \frac{b_g - g_n}{g_n} & q_{0n} \frac{-b_g}{g_n} & u_{nn} & \dots & u_{n-dnu} & -u_{fn-1} & \dots & -u_{fn-dde} \end{bmatrix} = \mathbf{J}_m \mathbf{J}_c
\end{aligned}$$

and $\mathbf{e} = [\hat{e}_{n+1} \dots \hat{e}_{n+K} \sqrt{\lambda} \Delta u_{fn}]^T$. The Jacobian matrix \mathbf{J}_m is derived via SVR_{NARMA-L2} model of the system as given in (2.40-2.42). An adaptive first order filter is adequate to successfully filter high frequency components and expeditiously force the system to track the reference signal for the proposed controller mechanism. Therefore, a first order adaptive low pass filter with an adjustable parameter is deployed to filter high frequency components of the control input as given by:

$$H(z) = \frac{u_{fn}}{u_{cn}} = \frac{q_{0n}}{1 + (1 - q_{0n})z^{-1}} \quad (2.47)$$

where $1 - q_0$, q_0 are "feed-backward" and "feed-forward" coefficients of the filter respectively. The adjustable parameters can be adapted using Levenberg-Marquardt algorithm [56] as follows:

$$\begin{bmatrix} \mu_{1new} \\ \mu_{2new} \\ q_{0new} \end{bmatrix} = \begin{bmatrix} \mu_{1old} \\ \mu_{2old} \\ q_{0old} \end{bmatrix} + [\mathbf{J}^T \mathbf{J} + \eta \mathbf{I}]^{-1} \mathbf{J}^T \mathbf{e} \quad (2.48)$$

where

$$\begin{aligned}
\mathbf{J} &= \begin{bmatrix} \frac{\partial e_{n+1}}{\partial \hat{y}_{n+1}} \frac{\partial \hat{y}_{n+1}}{\mu_1} & \frac{\partial e_{n+1}}{\partial \hat{y}_{n+1}} \frac{\partial \hat{y}_{n+1}}{\mu_2} & \frac{\partial e_{n+1}}{\partial \hat{y}_{n+1}} \frac{\partial \hat{y}_{n+1}}{\partial u_{f_n}} \frac{\partial u_{f_n}}{\partial q_0} \\ \vdots & \vdots & \vdots \\ \frac{\partial e_{n+K}}{\partial \hat{y}_{n+K}} \frac{\partial \hat{y}_{n+K}}{\mu_1} & \frac{\partial e_{n+K}}{\partial \hat{y}_{n+K}} \frac{\partial \hat{y}_{n+K}}{\mu_2} & \frac{\partial e_{n+K}}{\partial \hat{y}_{n+K}} \frac{\partial \hat{y}_{n+K}}{\partial u_{f_n}} \frac{\partial u_{f_n}}{\partial q_0} \\ \frac{\partial \sqrt{\lambda} \Delta u_{f_n}}{\partial \mu_1} & \frac{\partial \sqrt{\lambda} \Delta u_{f_n}}{\partial \mu_2} & \frac{\partial \sqrt{\lambda} \Delta u_{f_n}}{\partial q_0} \end{bmatrix} \\
&= \begin{bmatrix} -\frac{\partial \hat{y}_{n+1}}{\partial u_{f_n}} \frac{\partial u_{f_n}}{\partial u_{n_n}} \frac{\partial u_{n_n}}{\mu_1} & -\frac{\partial \hat{y}_{n+1}}{\partial u_{f_n}} \frac{\partial u_{f_n}}{\partial u_{n_n}} \frac{\partial u_{n_n}}{\mu_2} & -\frac{\partial \hat{y}_{n+1}}{\partial u_{f_n}} (u_{n_n} - u_{f_{n-1}}) \\ \vdots & \vdots & \vdots \\ -\frac{\partial \hat{y}_{n+K}}{\partial u_{f_n}} \frac{\partial u_{f_n}}{\partial u_{n_n}} \frac{\partial u_{n_n}}{\mu_1} & -\frac{\partial \hat{y}_{n+K}}{\partial u_{f_n}} \frac{\partial u_{f_n}}{\partial u_{n_n}} \frac{\partial u_{n_n}}{\mu_2} & -\frac{\partial \hat{y}_{n+K}}{\partial u_{f_n}} (u_{n_n} - u_{f_{n-1}}) \\ \sqrt{\lambda} \frac{\partial u_{f_n}}{\partial u_{n_n}} \frac{\partial u_{n_n}}{\mu_1} & \sqrt{\lambda} \frac{\partial u_{f_n}}{\partial u_{n_n}} \frac{\partial u_{n_n}}{\mu_2} & \sqrt{\lambda} (u_{n_n} - u_{f_{n-1}}) \end{bmatrix} \\
&= \begin{bmatrix} -\frac{\partial \hat{y}_{n+1}}{\partial u_{f_n}} \\ \vdots \\ -\frac{\partial \hat{y}_{n+K}}{\partial u_{f_n}} \\ \sqrt{\lambda} \end{bmatrix} \begin{bmatrix} \frac{\partial u_{f_n}}{\partial u_{n_n}} \frac{\partial u_{n_n}}{\partial \mu_1} & \frac{\partial u_{f_n}}{\partial u_{n_n}} \frac{\partial u_{n_n}}{\partial \mu_2} & (u_{n_n} - u_{f_{n-1}}) \end{bmatrix} \\
&= \begin{bmatrix} -\frac{\partial \hat{y}_{n+1}}{\partial u_{f_n}} \\ \vdots \\ -\frac{\partial \hat{y}_{n+K}}{\partial u_{f_n}} \\ \sqrt{\lambda} \end{bmatrix} \begin{bmatrix} q_{0_n} \frac{b_g - g_n}{g_n} & q_{0_n} \frac{-b_g}{g_n} & (u_{n_n} - u_{f_{n-1}}) \end{bmatrix} = \mathbf{J}_m \mathbf{J}_c
\end{aligned} \tag{2.49}$$

2.4.5 Control procedure

The control procedure for proposed SVR_{NARMA-L2} controller is as follows:

Step 1: Initialize controller, filter and model parameters.

-Controller parameters: $\mu_j, j \in \{1, 2\}$

-Filter parameters: $q_m, m \in \{0, 1 \dots, d_{nu}\}, v_k, k \in \{1 \dots, d_{de}\}$

-SVR_{NARX} model parameters: $\beta = b_\beta = 0, C = 1000, \varepsilon = 10^{-3}$

Step 2: Train SVR_{NARX} model using $(u_{f_{n-1}}, y_n)$ data pair

-Set index to n.

-Constitute $\mathbf{x}_{n-1} = [u_{n-2} \dots u_{n-n_u-1} y_{n-1} \dots y_{n-n_y}]$

-Predict \hat{y}_n via SVR_{NARX} model

-Calculate $e_{m_n} = y_n - \hat{y}_n$

If $|e_{m_n}| > \varepsilon$

Train system model where $e_{m_n} = y_n - \hat{y}_n$

else

Continue with system model parameters obtained at previous step

Step 3: Convert SVR_{NARX} model to SVR_{NARMA-L2} model

-Compute parameters of $F_{n-1}(\alpha, b_f)$ and $G_{n-1}(\theta, b_g)$ via parameters of $SVR_{NARX}(\beta, b_\beta)$ as in (2.29)

-Constitute $\mathbf{x}_n = [u_{n-1} \cdots u_{n-n_u} y_n \cdots y_{n-n_y+1}]$

-Compute \hat{f}_n^- and \hat{g}_n^- via $F_{n-1}(\cdot)$ and $G_{n-1}(\cdot)$

Step 4: Calculate the control input produced by $SVR_{NARMA-L2}$ controller (u_{c_n})

-Calculate the control signal (u_{c_n}) using the predictions of the submodels and control law in (2.32).

Step 5: Filter the high frequency components of control signal (u_{f_n})

- Compute the filtered control signal via (2.44)

Step 6: Apply the filtered control signal u_{f_n} to the system to compute y_{n+1}

- Training data pair of SVR_{NARX} model for the next step is obtained (u_{f_n}, y_{n+1})

Step 7: Calculate the Jacobian

-Apply u_{f_n} K-times to SVR_{NARX} model in order to constitute Jacobian matrix in (2.46 or 2.49)

Step 8: Learning step for controller and filter

-Update Controller parameters ($\mu_j, j \in \{1, 2\}$) and filter parameters ($q_m, m \in \{0, 1, \dots, d_{nu}\}, v_k, k \in \{1, \dots, d_{de}\}$) using (2.45 or 2.48)

Step 9: Increment $n = n + 1$ and go to step 2.

2.5 Simulation Results

2.5.1 The bioreactor system

The performance of the proposed $SVR_{NARMA-L2}$ controller is examined on a bioreactor system. Bioreactor is a tank containing water and cells (e.g., yeast or bacteria) which consume nutrients (substrate) and produce product (both desired and undesired) and more cells [60]. Since the uncontrolled equations of this system are highly nonlinear and exhibit limit cycles [60], it is a very difficult system to control and it has frequently been employed as a nonlinear benchmark problem to examine the performances of proposed adaptive controllers [5, 43, 61, 62]. The differential equations describing the cell concentration $[c_1(t)]$ and amount of nutrients $[c_2(t)]$ are given as follows:

$$\begin{aligned} \dot{c}_1(t) &= -c_1(t)u(t) + c_1(t)(1 - c_2(t))e^{\frac{c_2(t)}{\gamma(t)}} \\ \dot{c}_2(t) &= -c_2(t)u(t) + c_1(t)(1 - c_2(t))e^{\frac{c_2(t)}{\gamma(t)}} \frac{1 + \beta(t)}{1 + \beta(t) - c_2(t)} \end{aligned} \quad (2.50)$$

where $c_1(t)$ is the controlled output of the system ($y(t) = c_1(t)$), $u(t)$ is the flow rate as the control signal, $\gamma(t)$ is nutrient inhibition parameter, $\beta(t)$ is grow rate parameter [5, 43, 60, 62]. In the simulations, the max-min limitations for the magnitude of the control signal are given as $u_{min} = 0$ and $u_{max} = 2$; and duration of control signal is chosen as $\tau_{min} = \tau_{max} = 0.5s$. Since we have assumed that the dynamics of the system is not known, online SVR_{NARX} has been utilized to identify the unknown dynamics using the input-output data pairs. Then an equivalent SVR_{NARMA-L2} model is obtained via proposed method given in (2.25-2.29,2.34). Thus, SVR_{NARMA-L2} submodels can be computed to design the control law given in (2.32). The input feature vectors for SVR_{NARX} and SVR_{NARMA-L2} models are chosen as $\mathbf{M}_c = [u_{n-1} \cdots u_{n-n_u} y_n \cdots y_{n-n_y+1}]^T$ where $n_u = 3$ and $n_y = 3$ indicate the number of the past inputs and outputs, respectively.

2.5.2 SVR design parameters

Identification methods based on ε -SVR possess several design parameters which have direct influence on modeling performance. ε which is the most significant parameter in identification is the maximum permissible modeling error in training phase of ε -SVR. ε -SVR cannot accurately capture the dynamics of the system when ε is chosen too large and the controller may induce unacceptable control performance. If it is chosen too small, identification can result in too many support vectors than required to model successfully. Therefore ε , the maximum tolerable error for system model, must be chosen so that an acceptable control quality [5] is obtained while the number of support vector for SVR_{NARX} model is kept at a reasonable level. Accordingly, maximum tolerable error is assigned as $\varepsilon = 10^{-3}$. Also, the maximum number of the support vector is limited to a predefined value to decrease computational load in our simulations. As can be seen in (2.13), C is the boundary of the parameter search box in which Lagrange multipliers are searched. Since the control algorithm gets slower as the number of support vectors increases, C is fixed at a large value (1000) in order to reduce the number of the support vectors as low as possible. The mapping of the input data to feature space and the nonlinearity in SVR are provided via kernel functions, hence selection of kernel parameters is also crucial in regression performance. In our simulations, we utilized an exponential kernel function with $K(\mathbf{x}, \mathbf{y}) = e^{\frac{-\|\mathbf{x}-\mathbf{y}\|}{2\sigma^2}}$. The

value of σ effects the separability of the data in feature space. Since the kernel matrix stores similarities of the input data, very small σ maps similar data in input space to distant locations in feature space, the number of the support vectors for system model increases to capture the system dynamics. On the other hand, large σ values maps different data in input space to close locations in feature space and results in loss of the nonlinearity of the kernel that leads to inaccurate identification of system dynamics. Therefore, the similarities of the data in input space must be reflected in the feature space as correctly as possible. We have also observed that very small σ values cause fluctuations in control signal and system output since the number of the support vectors increases to capture the system dynamics. On the other hand, when σ is chosen too large, the nonlinearity of the kernel is withered and the control signal is too slow, so the system cannot be forced to track reference signal accurately. In simulations for bioreactor system SVR_{NARX} and SVR_{NARMA-L2} have been designed with $\sigma = 0.75$. The closed-loop performance of the system is observed for different values of prediction horizon (K) and penalty parameter (λ) using the performance criteria in (2.51) and the optimal values are determined with the grid search algorithm.

$$J_{\text{comp}} = \frac{1}{2} \sum_{n=1}^{\infty} [r_{n+1} - y_{n+1}]^2 + \lambda [u_n - u_{n-1}]^2 \quad (2.51)$$

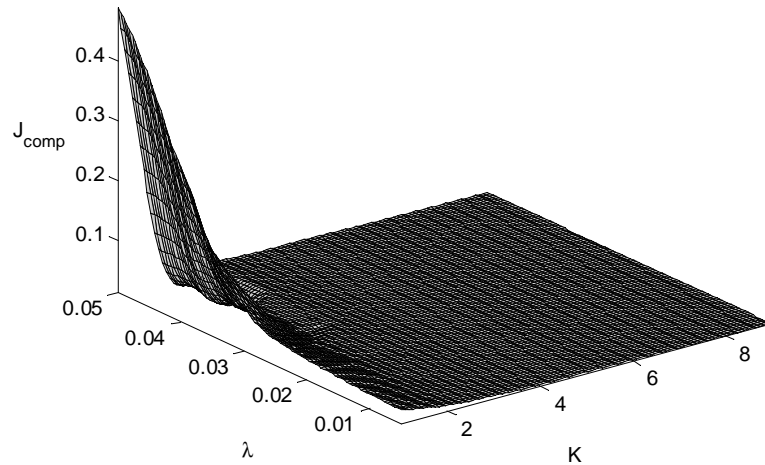


Figure 2.8 : Tracking performance surface with respect to penalty term (λ) and prediction horizon (K) for bioreactor.

As can be seen from Figure 2.8, as λ increases, the fluctuation of control signal is restricted, therefore, the tracking performance deteriorates. The performance is positively affected as K increases. Thus, the values of λ and K for bioreactor are

determined as $K = 9$, $\lambda = 0.005$, $J_{\text{comp}_{\min}} = 0.013235$ via the minimum of (2.51) given in Figure 2.8.

2.5.3 Simulation results

The tracking performance of controller for the case when no noise is applied to the system and all parameters are fully known is depicted in Figure 2.9. The controller and filter parameters are illustrated in Figure 2.10.

In order to assess the performance of the controller with respect to measurement noise, 30 dB measurement noise is added to the output of the system. The tracking performance is illustrated in Figure 2.11. Adaptation of the controller and filter parameters are given in Figure 2.12.

The robustness of the controller is inspected in terms of parametric uncertainty, $\gamma(t)$ is considered as the time-varying system parameter, where its nominal value is $\gamma_{\text{nom}}(t) = 0.48$. The value of $\gamma(t)$ is allowed to slowly alter in neighborhood of its nominal value as $\gamma(t) = 0.48 + 0.06 \sin(0.012\pi t)$. It can be interpreted from Figure 2.13 that the controller can successfully manage to reject the uncertainty in system parameters. Controller and filter parameters are illustrated in Figure 2.14. The maximum number of the support vectors are limited to 10, 25 and 35 for noiseless, noisy, and parametric uncertainty cases, respectively.

2.5.4 Comparison of the results with SVM-based PID controller

A performance comparison of $\text{SVR}_{\text{NARMA-L2}}$ controller proposed in this paper and the SVM-based PID controller implemented by Iplikci [5] has been carried out for cases with no noise, with measurement noise added to the system and with parametric uncertainty. The tuning mechanism described in [5] adjusts the parameters of a PID controller via Levenberg-Marquardt algorithm by using the approximated K-step ahead future system behaviour. The controller consists of five components: classical incremental PID controller, SVR_{NARX} model of the system, line search block, control signal correction block and controller parameter tuner. The Jacobian matrix utilized in parameter tuner block is coalescence of correction block which includes K-step ahead system Jacobian and a gradient vector which involves first order derivative of control signal with respect to controller parameters. K-step ahead future Jacobian of the

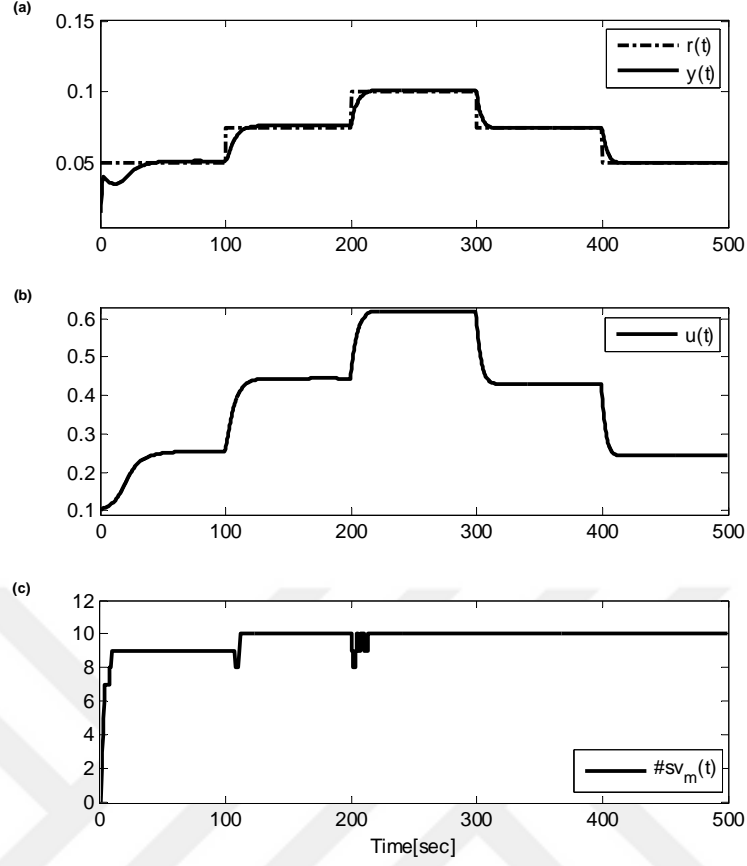


Figure 2.9 : System output (a), control signal (b), and number of support vectors (c) with adaptive filter for the case with no noise or parametric uncertainty.

system is approximated via SVR_{NARX} model of the system. Control signal correction block produces a correction term for the control signal since the adjusted controller parameters may not be adequate to force the system output to track the desired reference. The control signal correction term is derived via Taylor approximation of the control signal term which includes correction term. Golden section method has been employed in line search block to determine optimum learning rate for correction term. The simulation results for the $SVR_{NARMA-L2}$ controller proposed in this paper are compared with results obtained for the structure given in [5]. Same prediction horizon (K) and penalty term (λ) have been employed in both $SVR_{NARMA-L2}$ controller and SVM-based PID controller for comparison. The tracking performances of the controllers for cases with no noise, with measurement noise and with parametric uncertainty are illustrated in Figures 2.15-2.17 for bioreactor system, respectively. Comparisons of the performance index given by (2.51) for tracking performance of both controllers are illustrated in Figure 2.18 which shows that $SVR_{NARMA-L2}$ controller has better tracking performance than SVM-based PID controller for all

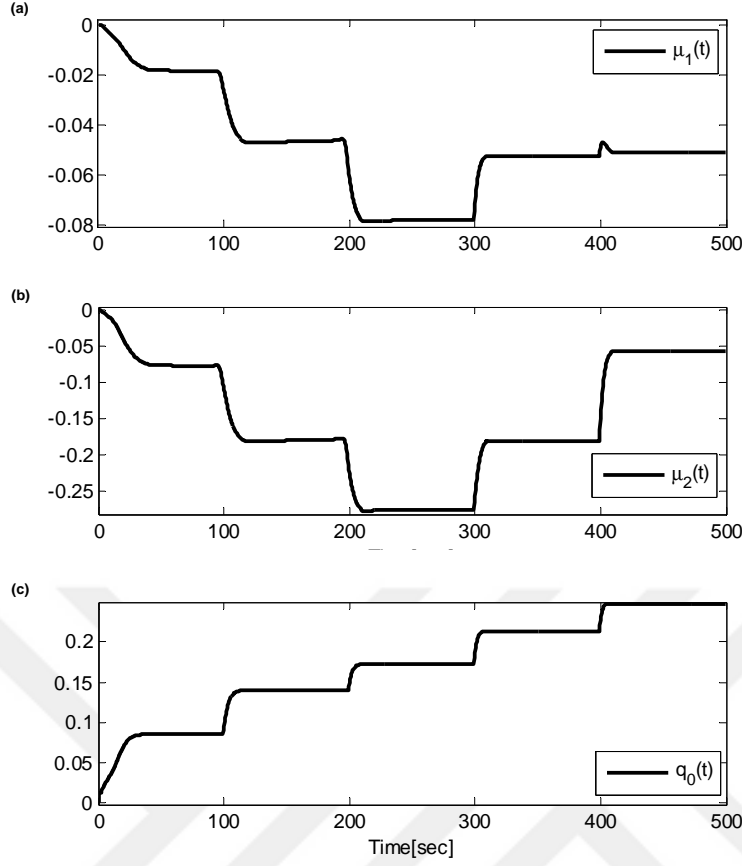


Figure 2.10 : Adaptive controller (a,b) and filter parameters (c) for the case with no noise or parametric uncertainty.

cases except the case with parametric uncertainty. The applicability of the proposed controller in real time is as significant as its tracking performance. For this purpose, computation times of each operation in the control algorithm have been registered for each case during every sampling period and the average response times of the operations have been listed in Table 2.1 for all conditions. In our simulations, a PC with 2.2 GHz core i7 CPU and 8 GB RAM has been operated for the implementation of the control algorithm without optimizing codes. As can be seen from the Table 2.1, the average response times of the controller accrete for measurement noise and parametric uncertainty cases since the controller struggles to tolerate the uncertainty in the system. In simulations, the sampling time is chosen as 100 ms and the total average response times of the proposed controller are less than 50 ms as given in Table 2.1. Consequently, it can be expressed that SVR_{NARMA-L2} controller can be deployed in real-time applications conveniently. Moreover, total response times can be minimised and the controller can be easily employed by implementing the algorithm on

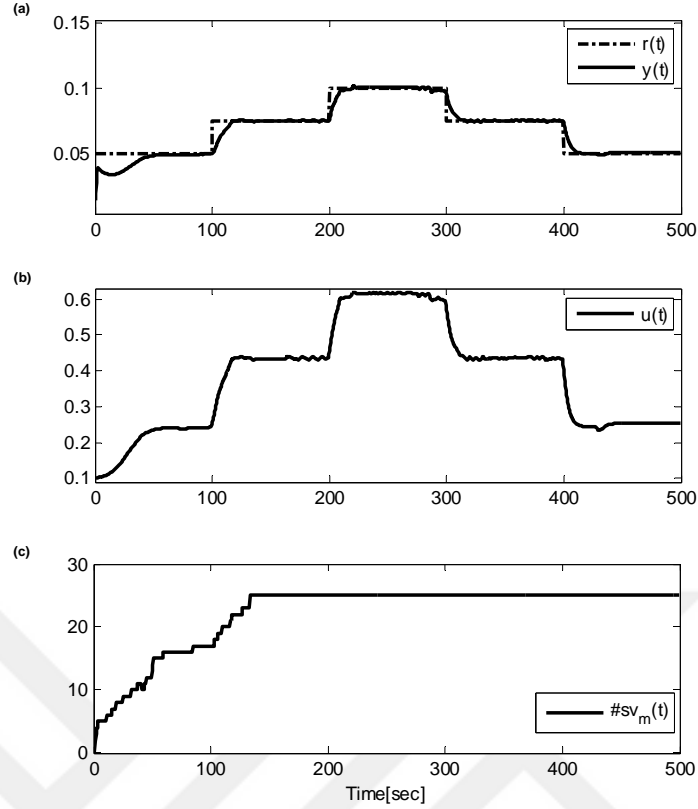


Figure 2.11 : System output (a), control signal (b), and number of support vectors (c) with adaptive filter for the case with measurement noise.

a lower level programming environment such as C/C++ utilizing effective hardwares (eg. FPGA) in real time application.

Table 2.1 : Computation times(in ms).

Operations	Bioreactor		
	Noiseless	Noisy	Uncertainty
SVR _{NARX} Model Training	6.8487	24.42	18.8102
Conversion to SVR _{NARMA-L2} Model	0.93768	0.95917	0.95705
K-step Jacobian	6.9377	8.8229	16.0968
Parameter Tuning (LM)	0.029813	0.031787	0.037375
Miscellaneous Tasks	2.9456	2.9348	3.0298
Total Time	17.6995	37.1686	38.9312

2.6 Conclusion

In this paper, a novel SVR_{NARMA-L2} controller is proposed where online SVR is used to estimate SVR_{NARMA-L2} submodels. Firstly, a SVR_{NARX} model is constructed to approximate the system, then the parameters of the SVR_{NARMA-L2} model are derived via SVR_{NARX} model of the system. Consequently, SVR_{NARMA-L2} controller

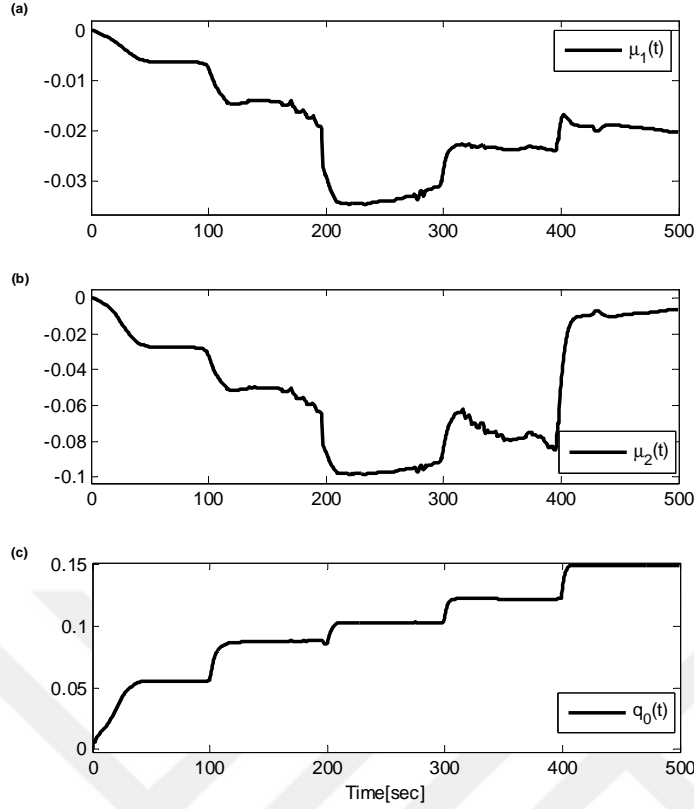


Figure 2.12 : Adaptive controller (a,b) and filter parameters (c) for the case with measurement noise.

is designed using the $SVR_{NARMA-L2}$ submodels. The main contribution of the paper is that it justifies the use of an online SVR_{NARX} model of the system directly to derive online $SVR_{NARMA-L2}$ controller as opposed to existing works in technical literature where SVR_{NARX} models are generally utilized for adjusting the parameters of the traditional controllers by approximating system Jacobians. Another novelty of this work is to use SVR to estimate system model in NARMA-L2 controller design instead of ANN. In technical literature, NARMA-L2 controllers have previously been designed using ANNs to approximate system model. However, ANNs are trained with backpropagation algorithm and have the drawback that they may get stuck at local minima which will lead to models valid only locally. The main strength of SVR in modelling is that it minimizes a convex goal function and achieves global minimum. The performance of the proposed controller is evaluated by simulations done on a bioreactor benchmark system. Additionally, the success of the controller is compared with an SVM-based PID controller. The robustness of the controller against system parameter uncertainty and measurement noise have also been examined. The results indicate that the proposed controller is quite succesful in attaining low tracking error,

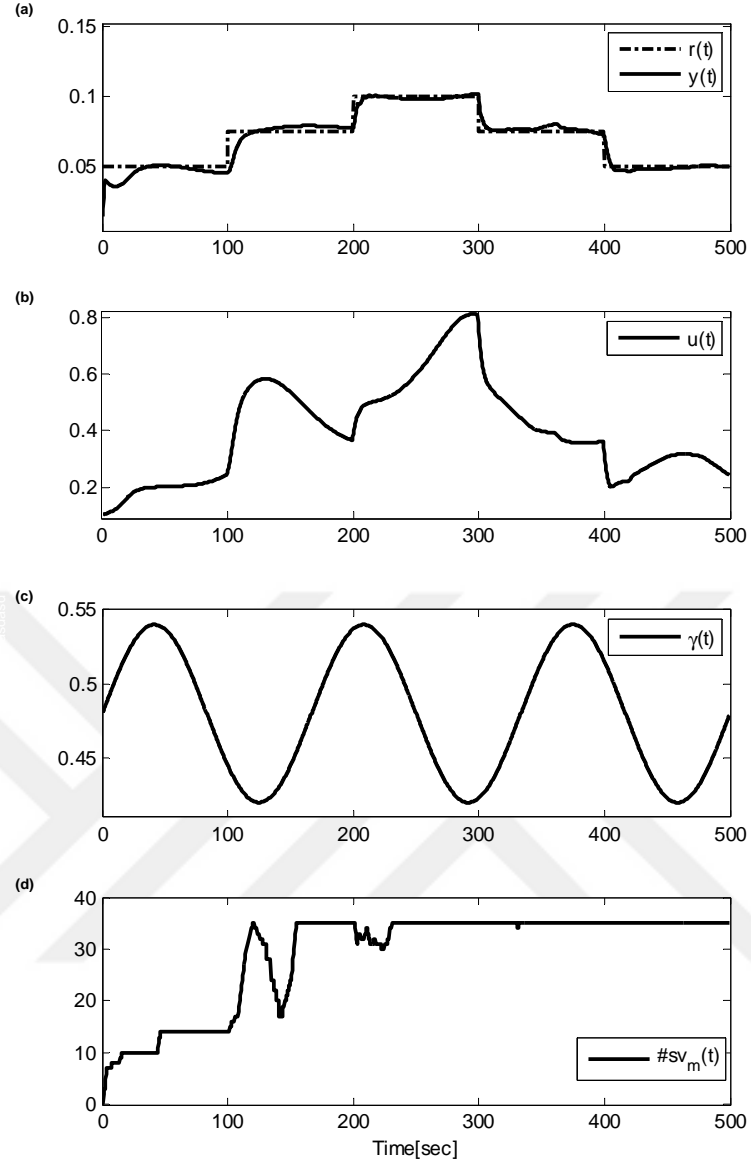


Figure 2.13 : System output (a), control signal (b), uncertain system parameter (c), and number of support vectors (d) with adaptive filter for the case with parametric uncertainty.

suppressing measurement noise and parametric uncertainties. In future works, it is planned to extend the derivation of $SVR_{NARMA-L2}$ model via SVR_{NARX} to develop new SVR type adaptive controller design methods.

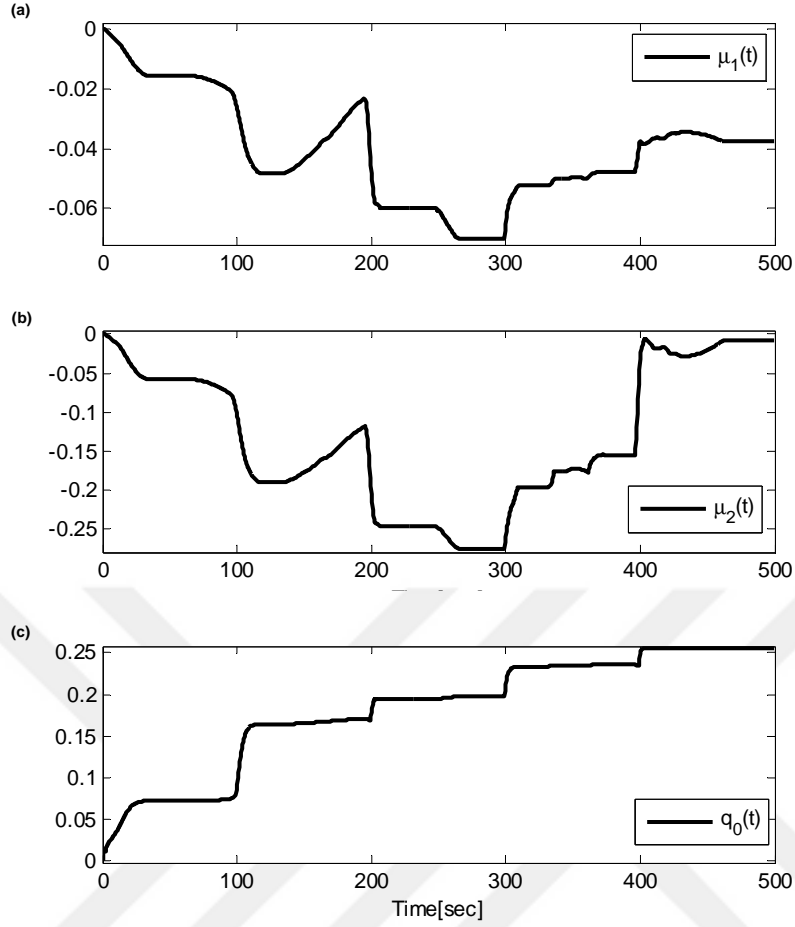


Figure 2.14 : Adaptive controller (a,b) and filter parameters (c) for the case with parametric uncertainty.

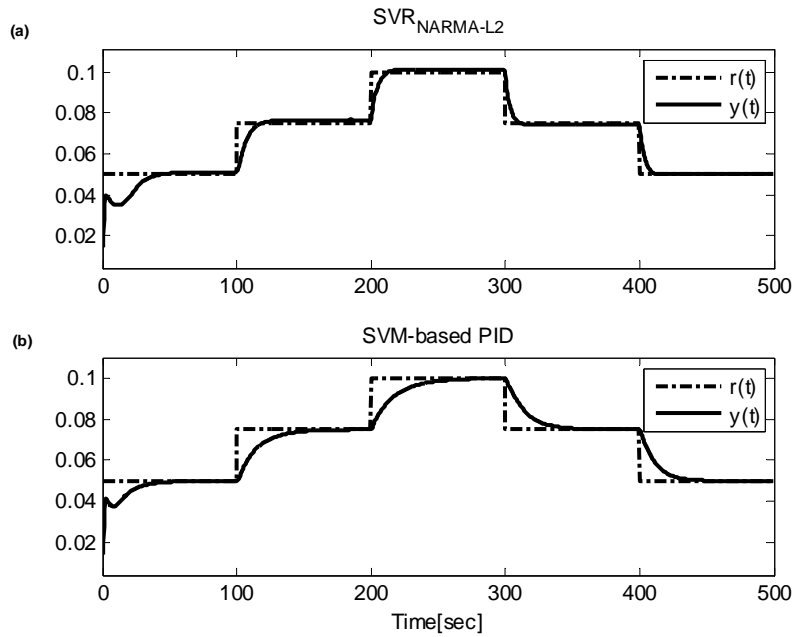


Figure 2.15 : Tracking performance of SVR_{NARMA-L2} controller (a) and SVM-based PID controller (b) for the case with no noise or parametric uncertainty.

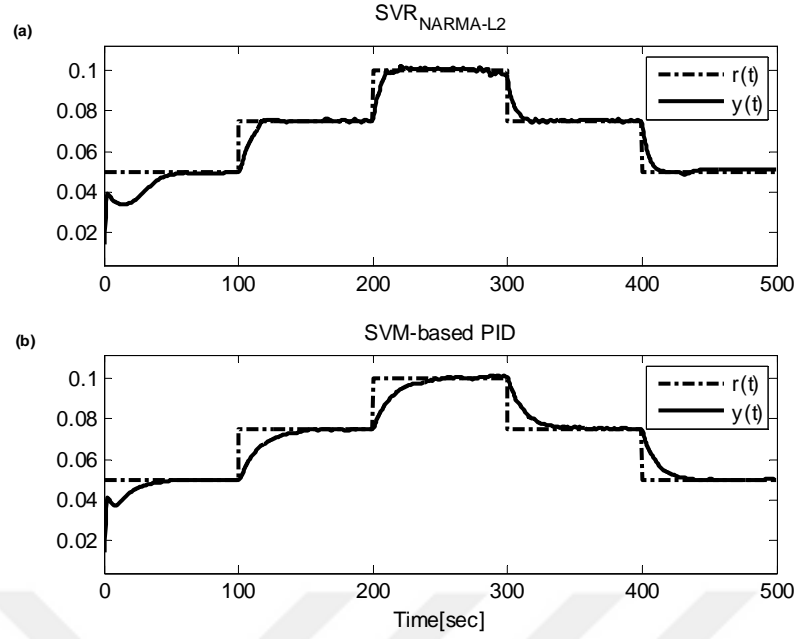


Figure 2.16 : Tracking performance of $SVR_{NARMA-L2}$ controller (a) and SVM-based PID controller (b) for the case with measurement noise.

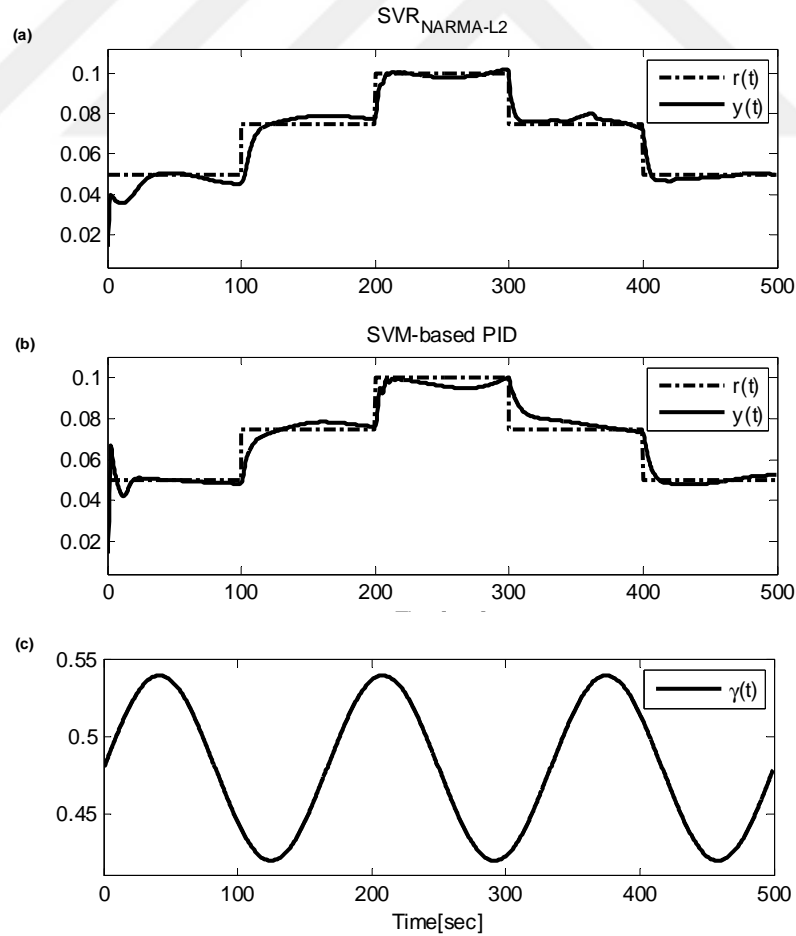


Figure 2.17 : Tracking performance of $SVR_{NARMA-L2}$ controller (a) and SVM-based PID controller (b) for the case with parametric uncertainty (c).

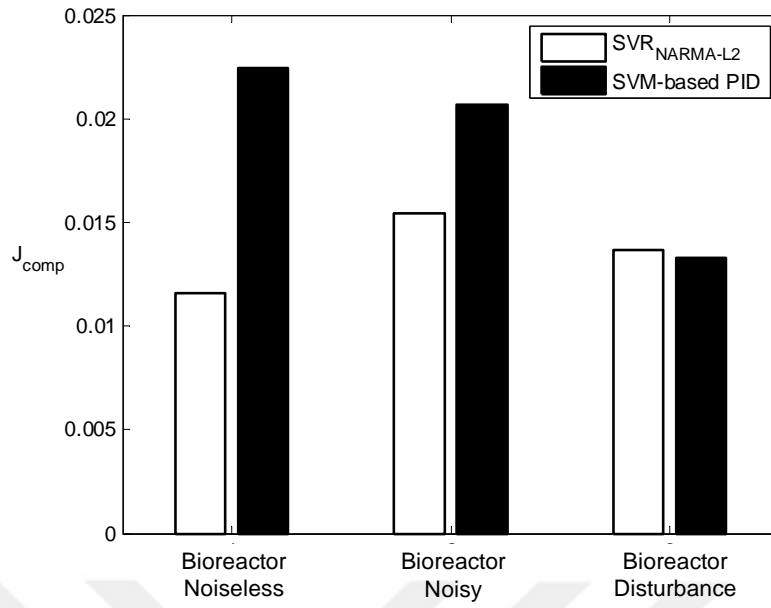


Figure 2.18 : Tracking performance comparison of the controller with respect to the defined performance index (2.51).

3. AN ADAPTIVE SUPPORT VECTOR REGRESSOR CONTROLLER FOR NONLINEAR SYSTEMS ²

3.1 Introduction

Altering a behaviour to comply with new circumstances is called "adaptation" in everyday language [63]. Adaptive controller is a type of controller which can modify its behaviour depending on system characteristics and external factors such as disturbance and noise [63]. It is obvious that controllers with fixed parameters cannot provide acceptable system behaviour in all situations [7]. Therefore, acquiring some knowledge about how the system will respond when it is manipulated in various ways is a vital step in controller design [52]. Since coping with strong nonlinearities and time varying dynamics is a hard task in nonlinear systems, numerous control techniques have been proposed to solve these problems. In order to assure good control performance, adaptation of the controller parameters to deal with the varying and nonlinear dynamics of the system can be necessary. To be able to adapt the controller to the changing dynamics of the system, firstly the system dynamics must be identified. When the dynamics of the system is known, its future behaviour can also be predicted and the parameters of the controller can be adjusted accordingly. Intelligent methods such as artificial neural networks (ANN), adaptive neuro-fuzzy inference systems (ANFIS) and support vector regressors (SVR) can be used to identify the dynamics and to obtain the future behaviour of a system. Intelligent methods can be employed not only to model the system dynamics, but also to design a controller in closed loop systems.

Although complicated nonlinear systems can be controlled successfully using intelligent methods, the performance of the controller is directly related to the modelling performance. Since ANN and ANFIS have non-convex objective functions, when they are used in modelling the solution may get stuck at local minima and

²This chapter is based on the paper "Uçak K. and Günel G.Ö., (2016), An adaptive support vector regressor controller for nonlinear systems, Soft Computing, Vol.: 20, Issue: 7, Page: 2531-2556"

the obtained model of the system is valid only locally. The controller performance in model based methods is highly influenced by modelling inaccuracies. In order to reduce modelling inaccuracies and improve controller performance, SVR based identification methods have popularly been applied in recent years due to their non-linear prediction ability and generalization performance. The main strength of SVR is that it uses a convex objective function and thus ensures the global solution. Models based on SVR have commonly been utilized in model based controller parameter tuning methods [5, 29], instead of ANN approach, since it yields a unique solution [29, 30] and offers powerful generalization ability with very few training data. In technical literature, there exist various controller structures based on SVR modelling. These structures can be examined under four main headings: PID control, inverse control, internal model control (IMC) and model predictive control (MPC).

Shang et al. [31] proposed to adjust the parameters of a PID controller via gradient descent by approximating system Jacobian using online least square support vector regression (OLSSVR) based on sliding window. The control performance depends on the modelling performance which is directly affected by the size of the sliding window. The main disadvantage of sliding window-based online learning is that the intelligent model forgets the transient dynamics of the system in steady state. Zhao et al. [32] proposed to increase or decrease the size of the sliding window to prevent overfitting or underfitting. In the case where the modelling performance is worse than the given performance interval, the kernel machine is not adequate to learn the data, so the size of the sliding window should be increased. If the size of the sliding window is larger than the optimum value the obtained model will be too complex than required and there may be overfitting. The regression performance of an SVR depends on the chosen kernel function which is generally parametric and the numerical values of these parameters are important in determining the location of the features mapped onto the feature space, however there is no theoretical method to determine them numerically. Because of this, the selection of the kernel function and the numerical values of its parameters are very crucial in terms of modelling and control performance. For this purpose, Ucak and Oke [29] proposed to tune kernel parameters simultaneously with PID controller parameters via gradient descent method to improve modelling and control performance of OLSSVR. Yuan et al.

[33] introduced a composite feedforward-feedback controller where the feedforward controller is derived based on approximation model method using SVR. NARMA model of the system has been employed to design feedforward controller to make use of advantages of approximate model method and conventional PID has been utilized as feedback controller in order to assure robust tracking properties. Robustness of the controller has also been investigated [33]. Iplikci [5] proposed an adaptive PID controller based on ε -SVR where controller parameters are updated depending on the approximated K-step ahead future system behaviour. The controller consists of five components, these are: classical PID controller, NARX model of the system, line search block, control signal correction block and controller parameter tuner. ε -SVR model has been used to approximate the system Jacobian. PID parameters have been tuned using Levenberg-Marquard algorithm. Jacobian matrix is separated into two blocks using chain rule, one block is for control signal correction and the other is for parameter tuning. The updated controller parameters may not be good enough to force the system output towards the desired trajectory, therefore, control signal correction block has been employed via Taylor approximation. Optimum step size has been obtained via golden section method. The common feature of these works is that SVR is utilized to identify both the dynamics of the system and the system Jacobian. Takao et al. [34] employed SVM as a classifier in a decision tree structure to select eligible controller parameters according to the operating point of the system. The wide range of uncertainty is firstly divided into small ranges using a priori information, then robust PID controllers are designed for each small range. The switching structure including the decision information for each small uncertainty range and corresponding robust PID controller parameters is constructed via an SVM classifier. Depending on the previously determined scenario and system behaviour, appropriate controller parameters for the specific operating range are loaded to the controller.

Since the aim in controller design is to realize almost the inverse dynamics of the controlled system, inverse model of the system can be used to manipulate the system behaviour. The main drawback using an inverse controller is that there is no guarantee for the existence of the system inverse or alternatively multiple inverse models of the controlled system can be obtained. Liu et al. [35] utilized the inverse model of the system as a controller and the control signal error has been compensated using a PID

controller. Wang et al. [36] deployed SVR to identify inverse dynamics of the system as an adaptive inverse controller via online learning algorithm. The parameters of the inverse controller and forward model of the system are adjusted via an adaptive law based on backpropagation algorithm by Yuan et al. [37]. To guarantee convergence and fast learning, adaptive learning rate and convergence theorems are developed [37]. Owing to its disturbance rejection capabilities [38] and robustness properties [39], nonlinear internal model control (NIMC) has a significant role among nonlinear control methods. In the implementation of nonlinear internal model control (NIMC), the determination of a system model constitutes an important stage of the design [38]. NIMC based on ANNs and fuzzy inference systems (FIS) were proposed in literature. Generally, these methods have some drawbacks in modelling: training speed is slow, generalization is not good, prior knowledge is needed to some degree [39]. The core idea of ANN IMC is that ANN model and ANN inverse model are utilized as the internal model and the IMC controller, respectively [38]. Since ANNs utilize gradient-based training algorithms such as backpropagation and the solution of algorithm may get stuck at local minima because of non-convex objective function, obtaining an inverse model of the nonlinear system via ANNs is notably a difficult task. Sun and Song [41] proposed an adaptive internal model controller (AIMC) to control nonlinear systems by combining LSSVR with sequential minimal optimization (SMO)-based pruning algorithm.

Model predictive control (MPC) techniques provide very robust schemes [43] to control highly nonlinear dynamic systems with large time delays and high-order dynamics compared to PID controllers [44–48]. All MPC techniques rely on the same strategy: based on the future predictions of a model of the system, a set of future control signals is obtained by solving, at each sampling instant, a finite-horizon open-loop optimal control problem and then the first element of the control sequence is applied to the system [43]. In order to optimize the set of the control signals depending on the future behavior of the system, K-step ahead future system Jacobian information is approximated via intelligent models. To reduce modelling inaccuracies of highly non-linear systems, Iplikci [43, 44], Du and Wang [49] and Shin et al. [50] employed SVR based system models in MPC. Iplikci [44] utilized offline SVR to model the dynamics of system. Shin et al. [50] proposed to tune the parameters of the previously

offline trained SVR model via gradient descent algorithm to prevent the deterioration in SVR system model. In order to overcome modelling inaccuracies resulting from disturbances or varying dynamics, Iplikci [43] and Du and Wang [49] deployed online SVR system model in MPC framework. In MPC, the aim is to adjust the set of control signals so that the system will follow the reference signal closely. For this purpose, an update direction for control vector should be determined. In order to obtain appropriate direction for control vector, first and second order derivatives of system output with respect to control inputs are estimated via SVR model depending on the optimization algorithm.

The common feature of the works in technical literature is that SVR has mainly been employed to approximate the system output or system Jacobian to optimize the parameters of different controllers using gradient descent or Levenberg-Marquard type optimization algorithms. SVMs have successfully been used to solve classification and regression problems owing to their good generalization property and ability to find the global minimum. However, to the best of our knowledge, they have not been used as controllers. The main reason for this is the unavailability of the required training data, namely the control input to be applied to the system, before the course of control.

In this paper, a novel online SVR controller is proposed to control a nonlinear dynamical system. The main contribution of the paper is that SVR is directly utilized as a controller for the first time, without necessarily any prior information on controller output. SVR parameters are tuned by optimizing the margin between reference input and system output. A second online SVR is used to estimate the model of the system to be controlled, the estimated system output is used in tuning the controller parameters. The performance of the proposed SVR controller has been evaluated by simulations carried out on continuously stirred tank reactor (CSTR) and bioreactor benchmark problems. Robustness of the proposed controller has been examined by adding measurement noise and parametric uncertainty to the system. Stability of the closed-loop system has also been analyzed. Additionally, the performance of the controller has been compared with an SVM-based PID controller proposed by Iplikci [5]. The results indicate that the online SVR controller together with online SVR model attain good modelling and control performances.

The paper is organized as follows: Section 3.2 describes the basic principles of online SVR. In section 3.3, the proposed control architecture is explained, the optimization problem to utilize SVR directly as a controller is constructed, main features of the problem are given via explicit figures. Training of online SVR parameters for controller design is summarized and parameter update rules are formulated. Also, the stability analysis of the closed loop system is presented. In section 3.4, simulation results and performance analysis of the controller are given together with an assessment of real time applicability. Also, the proposed method is compared with an SVM based PID controller. The paper ends with a brief conclusion in section 3.5.

3.2 Online Support Vector Regression

This section briefly reviews online support vector regression. The basic principles of support vector regression and online learning method are presented in section 3.2.1 to section 3.2.4.

3.2.1 An overview of support vector regression

Given a training data set

$$\mathbf{T} = \{\mathbf{x}_i, y_i\}_{i=1}^N, \quad \mathbf{x}_i \in \mathbf{X} \subseteq R^n, y_i \in R \quad (3.1)$$

where N is the size of the training data and n is the dimension of the input, the data can be represented using the regression model in (3.2).

$$y_i = \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b, \quad i = 1, 2, \dots, N \quad (3.2)$$

where $\Phi(\mathbf{x}_i)$ is the image of input data in feature space and b is the bias of the regressor. The optimization problem for regression seeks the optimal regressor within ε tube by optimizing the geometric margin and minimizing the training error. The primal form of the optimization problem is defined as follows

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) \quad (3.3)$$

subject to

$$\begin{aligned} y_i - \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle - b &\leq \varepsilon + \xi_i \\ \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b - y_i &\leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0, i = 1, 2, \dots, N \end{aligned} \quad (3.4)$$

where ε is the upper value of tolerable error, ξ 's and ξ^* 's are the slack variables representing the deviation from ε tube [1, 43]. The constraints of the problem are derived using ε -insensitive loss function [1, 54, 64]. The key idea in SVMs is to formulate a Lagrange function from the primal objective function and the corresponding constraints, by introducing a dual set of variables [1]. By combining the primal objective function and its constraints by introducing non-negative Lagrange multipliers β, β^*, η and η^* , Lagrangian can be derived as follows [1, 43]:

$$L = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) - \sum_{i=1}^N \beta_i (\varepsilon + \xi_i - y_i + \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b) - \sum_{i=1}^N \beta_i^* (\varepsilon + \xi_i^* + y_i - \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle - b) - \sum_{i=1}^N (\eta_i \xi_i + \eta_i^* \xi_i^*) \quad (3.5)$$

Due to the fact that the Lagrangian has a saddle point with respect to the primal and dual variables at the solution, the partial derivatives of L with respect to primal variables have to vanish for optimality [1]:

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \longrightarrow \mathbf{w} - \sum_{i=1}^N \beta_i \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle = 0 \quad (3.6)$$

$$\frac{\partial L}{\partial b} = 0 \longrightarrow \sum_{i=1}^N (\beta_i - \beta_i^*) = 0 \quad (3.7)$$

$$\frac{\partial L}{\partial \xi_i} = 0 \longrightarrow C - \beta_i - \eta_i = 0, i = 1, 2, \dots, N \quad (3.8)$$

$$\frac{\partial L}{\partial \xi_i^*} = 0 \longrightarrow C - \beta_i^* - \eta_i^* = 0, i = 1, 2, \dots, N \quad (3.9)$$

Utilizing optimality conditions (3.6-3.9) in (3.5), dual form of the optimization problem is formulated in (3.10)-(3.11):

$$\min_{\beta, \beta^*} D = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\beta_i - \beta_i^*)(\beta_j - \beta_j^*) K_{ij} + \varepsilon \sum_{i=1}^N (\beta_i + \beta_i^*) - y_i \sum_{i=1}^N (\beta_i - \beta_i^*) \quad (3.10)$$

subject to

$$0 \leq \beta_i \leq C, \quad 0 \leq \beta_i^* \leq C$$

$$\sum_{i=1}^N (\beta_i - \beta_i^*) = 0, \quad i = 1, 2, \dots, N \quad (3.11)$$

where $K_{ij} = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$. As can be seen in (3.10)-(3.11), dual form of the problem has a convex objective function with corresponding linear constraints, so a global solution is ensured. The optimal solution to the dual problem can be obtained by finding Lagrange multipliers utilizing a quadratic programming solver.

The support vectors are the training data related to nonzero Lagrange multipliers [5, 43, 54]. The solution of the regression problem in (3.2) can be approximated by the support vectors and the corresponding Lagrange multipliers as in (3.12):

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^N \lambda_i K(\mathbf{x}_i, \mathbf{x}) + b, \lambda_i = \beta_i - \beta_i^* \quad (3.12)$$

Regression and classification problems are generally linearly inseparable in input space. By using a kernel function, linearly inseparable problems can be mapped to a high dimensional feature space where nonlinearly distributed data are sparse as well as possibly more separable [65] and linear regression can be successfully carried out. The regression performance of an SVR depends on the chosen kernel function which is generally parametric and the numerical values of these parameters are important for distribution of the data in the feature space [65, 66]. For instance, if the bandwidth of a Gaussian kernel is chosen to be very small, similar data in the input space are mapped to distant locations in feature space and some data significant for the model may be discarded. This may yield to an SVR with perfect approximation which fully accords with the given output for each training instance, but with low generalization capability [67]. If the bandwidth is chosen to be very large, nonlinearity of the kernel is lost, dissimilar data in the input space can be mapped very close to each other in the feature space, this leads to a smooth SVR with strong generalization but with low approximation capability. Therefore, kernel parameters must be selected so that an optimal compromise between approximation and generalization capabilities is reached [67].

3.2.2 Basic principles of online support vector regression

In order to realize the main idea behind the online SVR, the definition of "margin" must be grasped. For this purpose, let us define a margin function $h(\mathbf{x}_i)$ for the i th sample \mathbf{x}_i as:

$$h(\mathbf{x}_i) = f(\mathbf{x}_i) - y_i = \sum_{j=1}^N \lambda_j K_{ij} + b - y_i \quad (3.13)$$

According to Lagrange multipliers and KKT conditions the training samples can be separated into three sets [2, 4] called as:

Set **E**:

Error Support Vectors $\mathbf{E} = \{i \mid |\lambda_i| = C, |h(\mathbf{x}_i)| \geq \varepsilon\}$

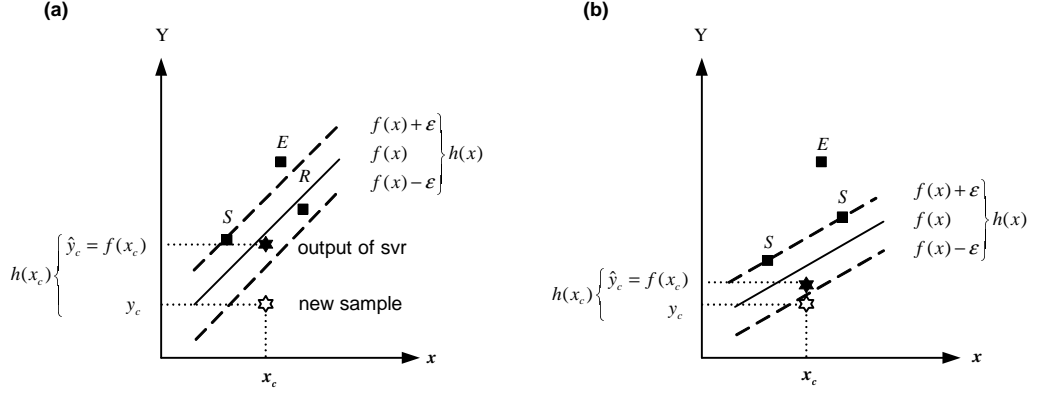


Figure 3.1 : E, S and R subsets before (a) and after (b) training.

Set **S**:

Margin Support Vectors $\mathbf{S} = \{i | 0 < |\lambda_i| < C, |h(\mathbf{x}_i)| = 0\}$

Set **R**:

Remaining Samples $\mathbf{R} = \{i | |\lambda_i| = 0, |h(\mathbf{x}_i)| \leq \varepsilon\}$

When a new sample \mathbf{x}_c is added, the goal is to let \mathbf{x}_c enter one of the three sets, while KKT conditions can still be satisfied automatically [2]. Let the Lagrange multiplier of the new added data be $\lambda = 0$, from (3.13), the margin for new added data is obtained as:

$$h(\mathbf{x}_c) = f(\mathbf{x}_c) - y_c = \sum_{i=1}^N \lambda_i K(\mathbf{x}_i, \mathbf{x}_c) + b - y_c \quad (3.14)$$

Then the value of λ_c , is gradually updated to provide all other samples satisfy KKT conditions. Depending on the adjustment of Lagrange parameter of the new added data, the values of the Lagrange multipliers (λ_i) and margin values of the previously learned samples ($h(\mathbf{x}_i)$) may change. Owing to this, the sets **E**, **S** and **R** of the previously learned samples may also change. Figure 3.1 illustrates how the sample in **R** immigrates to the **S** class and new learned sample enters into **R** class.

3.2.3 Derivation of update rules for Lagrange multipliers

In online learning, the new added data changes the structure of the regression problem and makes it necessary to update the parameters of the previous model by ensuring the KKT conditions. In order to derive online learning rules, the Lagrange function of the dual form is required. The Lagrange function for dual formulation can be written as

follows via (3.10,3.11).

$$L_D = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\beta_i - \beta_i^*)(\beta_j - \beta_j^*)K_{ij} + \varepsilon \sum_{i=1}^N (\beta_i + \beta_i^*) - \sum_{i=1}^N y_i(\beta_i - \beta_i^*) - \sum_{i=1}^N (\delta_i \beta_i + \delta_i^* \beta_i^*) - \sum_{i=1}^N u_i(C - \beta_i) + u_i^*(C - \beta_i^*) - \sum_{i=1}^N z(\beta_i + \beta_i^*) \quad (3.15)$$

KKT optimality conditions are obtained in (3.16) since the first order derivatives of Lagrange function with respect to dual variables equal to zero:

$$\begin{aligned} \frac{\partial L_D}{\partial \beta_i} &= \sum_{j=1}^N (\beta_j - \beta_j^*)K_{ij} + \varepsilon - y_i - \sum_{i=1}^N \delta_i + \sum_{i=1}^N u_i + z = 0 \\ \frac{\partial L_D}{\partial \beta_i^*} &= - \sum_{j=1}^N (\beta_j - \beta_j^*)K_{ij} + \varepsilon + y_i - \sum_{i=1}^N \delta_i^* + \sum_{i=1}^N u_i^* - z = 0 \\ \delta_i^{(*)} &\geq 0, u_i^{(*)} \geq 0, \delta_i^{(*)} \beta_i^{(*)} = 0, u_i^{(*)} (C - \beta_i^{(*)}) = 0 \end{aligned} \quad (3.16)$$

KKT condition indicates that at most one of the β_i and β_i^* should be nonzero and both are nonnegative [4]. Using the actual output function and SVR output, the following margin for the i th sample \mathbf{x}_i can be defined:

$$h(\mathbf{x}_i) = f(\mathbf{x}_i) - y_i = \sum_{j=1}^N \lambda_j K_{ij} + b - y_i \quad (3.17)$$

Thus, the following situations constructing the basics of the convergence and migration of the data are obtained:

$$\begin{aligned} h(\mathbf{x}_i) &\geq \varepsilon, \lambda_i = -C \\ h(\mathbf{x}_i) &= \varepsilon, -C < \lambda_i < 0 \\ -\varepsilon &\leq h(\mathbf{x}_i) \leq \varepsilon, \lambda_i = 0 \\ h(\mathbf{x}_i) &= -\varepsilon, 0 < \lambda_i < C \\ h(\mathbf{x}_i) &\leq -\varepsilon, \lambda_i = C \end{aligned} \quad (3.18)$$

If a new vector with influence λ_c is added without migration of vectors between sets \mathbf{S} , \mathbf{E} and \mathbf{R} , the variation in $\Delta h(\mathbf{x}_i)$ and λ_c are derived as in (3.19) via (3.13-3.18) [4, 55].

$$\Delta h(\mathbf{x}_i) = K_{ic} \Delta \lambda_c + \sum_{j=1}^N K_{ij} \Delta \lambda_j + \Delta b \quad (3.19)$$

From the dual constraints in (3.11)

$$\lambda_c + \sum_{j=1}^N \lambda_j = 0 \quad (3.20)$$

is obtained. If any vector related to previous or new data is an element of the subset \mathbf{E} or \mathbf{R} , the corresponding value of the Lagrange multiplier (λ_c) equals to "0" or "C". In particular, if support vectors must remain in \mathbf{S} , then $\Delta h(\mathbf{x}_i) = 0, i \in \mathbf{S}$ [55]. Thus, if the term $\Delta \lambda_c$ in equation (3.19) and (3.20) is isolated [55], the variations of Lagrange multipliers for the data in the support vector set can be easily computed for the obtained $\Delta \lambda_c$ as follows:

$$\sum_{j=1}^N K_{ij} \Delta \lambda_j + \Delta b = -K_{ic} \Delta \lambda_c, \quad \sum_{j \in SV} \Delta \lambda_j = -\Delta \lambda_c \quad (3.21)$$

If the indices of the samples in support vector set are defined as in

$$S = \{s_1, s_2, s_3, \dots, s_k\} \quad (3.22)$$

(3.21) can be represented in matrix form as

$$\begin{bmatrix} 0 & 1 & \dots & 1 \\ 1 & K_{s_1 s_1} & \dots & K_{s_1 s_k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & K_{s_k s_1} & \dots & K_{s_k s_k} \end{bmatrix} \begin{bmatrix} \Delta b \\ \Delta \lambda_{s_1} \\ \vdots \\ \Delta \lambda_{s_k} \end{bmatrix} = - \begin{bmatrix} 1 \\ K_{s_1 c} \\ \vdots \\ K_{s_k c} \end{bmatrix} \Delta \lambda_c \quad (3.23)$$

that is,

$$\Delta \boldsymbol{\lambda} = \begin{bmatrix} \Delta b \\ \Delta \lambda_{s_1} \\ \vdots \\ \Delta \lambda_{s_k} \end{bmatrix} = \boldsymbol{\beta} \Delta \lambda_c \quad (3.24)$$

where

$$\boldsymbol{\beta} = \begin{bmatrix} \beta \\ \beta_{s_1} \\ \vdots \\ \beta_{s_k} \end{bmatrix} = -\boldsymbol{\Theta} \begin{bmatrix} 1 \\ K_{s_1 c} \\ \vdots \\ K_{s_k c} \end{bmatrix}, \quad \boldsymbol{\Theta} = \begin{bmatrix} 0 & 1 & \dots & 1 \\ 1 & K_{s_1 s_1} & \dots & K_{s_1 s_k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & K_{s_k s_1} & \dots & K_{s_k s_k} \end{bmatrix}^{-1} \quad (3.25)$$

as given in [4]. Thus, for a given $\Delta \lambda_c$, the bias and the Lagrange multipliers of the samples in the support set can be updated using (3.23-3.25). Employing (3.18-3.19, 3.24), the alternation in margin for non-support samples can be updated as follows:

$$\begin{bmatrix} \Delta h(\mathbf{x}_{z_1}) \\ \Delta h(\mathbf{x}_{z_2}) \\ \vdots \\ \Delta h(\mathbf{x}_{z_m}) \end{bmatrix} = \boldsymbol{\gamma} \Delta \lambda_c, \quad \boldsymbol{\gamma} = \begin{bmatrix} K_{z_1 c} \\ K_{z_2 c} \\ \vdots \\ K_{z_m c} \end{bmatrix} + \begin{bmatrix} 1 & K_{z_1 s_1} & \dots & K_{z_1 s_l} \\ 1 & K_{z_2 s_1} & \dots & K_{z_2 s_l} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & K_{z_m s_1} & \dots & K_{z_m s_l} \end{bmatrix} \boldsymbol{\beta} \quad (3.26)$$

where z_1, z_2, \dots, z_m are the indices of non-support samples, $\boldsymbol{\gamma}$ are margin sensitivities and $\boldsymbol{\gamma} = 0$ for samples in \mathbf{S} . The alternation of the matrix $\boldsymbol{\Theta}$ for learning and forgetting

stages and detailed information related to the recursive algorithm can be attained via [2, 4, 55]. The update rules given for support set samples and non-support samples enable us to adjust all λ_i and $h(\mathbf{x}_i)$ via given $\Delta\lambda_c$ [4]. The method for finding an appropriate $\Delta\lambda_c$ is presented in the next subsection.

3.2.4 Calculation of $\Delta\lambda_c$

In the incremental online SVR algorithm, when a new training data \mathbf{x}_c is received, its corresponding λ_c value is initially set to zero, which is later updated to $\Delta\lambda_c$ [43]. Then, the largest possible change $\Delta\lambda_c$ is calculated while at the same time keeping the system at the equilibrium state with respect to the new KKT conditions [43]. $\Delta\lambda_c$ is calculated in two steps. The first step is to determine whether the change $\Delta\lambda_c$ should be positive or negative as follows [4]:

$$q = \text{sign}(\Delta\lambda_c) = \text{sign}(y_c - f(\mathbf{x}_c)) = -\text{sign}(h(\mathbf{x}_c)) \quad (3.27)$$

After the sign of $\Delta\lambda_c$ is specified, in the second step, a bound on $\Delta\lambda_c$ imposed by each sample in the training set is computed [4]. In order to calculate the largest possible $\Delta\lambda_c$, the following bookkeeping procedure which includes five possible cases is performed. To simplify exposition, only $\Delta\lambda_c > 0$ is regarded since the case $\Delta\lambda_c < 0$ is similar [4].

For the new sample \mathbf{x}_c , there are two possible cases:

Case 1: The new sample \mathbf{x}_c immigrates to set **S** from set **E**, so $h(\mathbf{x}_c)$ changes from $h(\mathbf{x}_c) < -\varepsilon$ to $h(\mathbf{x}_c) = -\varepsilon$ and the algorithm terminates. Check the variation value (L_{c_1}) of new sample \mathbf{x}_c

$$L_{c_1} = \frac{-h(\mathbf{x}_c) - q\varepsilon}{\gamma_c} \quad (3.28)$$

Case 2: \mathbf{x}_c immigrates to set **E** from set **S** if λ_c increases from $\lambda_c < C$ to $\lambda_c = C$, and the algorithm terminates. Check the variation value (L_{c_2}) of the new sample \mathbf{x}_c from set **E**:

$$L_{c_2} = qC - \lambda_c \quad (3.29)$$

For each previously learned sample \mathbf{x}_i already in set **S**:

Case 3: \mathbf{x}_i immigrates to set **E** from set **S** if λ_i changes from $0 < |\lambda_i| < C$ to $|\lambda_i| = C$.

If λ_i changes from $0 < |\lambda_i| < C$ to $|\lambda_i| = 0$, \mathbf{x}_i moves to set **R** from set **S**. Check the variation value (L_i^S) of each sample \mathbf{x}_i in set **S** to set **E** or set **R**:

$$\begin{aligned}
&\text{-If } q\beta_i > 0 \text{ and } 0 < \lambda_i < C, L_i^S = \frac{C - \lambda_i}{\beta_i} \\
&\text{-If } q\beta_i > 0 \text{ and } -C < \lambda_i < 0, L_i^S = \frac{-\lambda_i}{\beta_i} \\
&\text{-If } q\beta_i < 0 \text{ and } 0 < \lambda_i < C, L_i^S = \frac{-\lambda_i}{\beta_i} \\
&\text{-If } q\beta_i < 0 \text{ and } -C < \lambda_i < 0, L_i^S = \frac{-C - \lambda_i}{\beta_i}
\end{aligned} \tag{3.30}$$

For each previously learned sample \mathbf{x}_i already in set **E**:

Case 4: \mathbf{x}_i is moved from set **E** to set **S** when $h(\mathbf{x}_i)$ changes from $|h(\mathbf{x}_i)| > \varepsilon$ to $|h(\mathbf{x}_i)| = \varepsilon$. Check the variation value (L_i^E) of each sample \mathbf{x}_i in set **E** to set **S**:

$$L_i^E = \frac{-h(\mathbf{x}_i) - \text{sign}(q\beta_i)\varepsilon}{\gamma_i} \tag{3.31}$$

For each previously learned sample \mathbf{x}_i already in set **R**:

Case 5: \mathbf{x}_i is moved from set **R** to set **S** when $h(\mathbf{x}_i)$ changes from $|h(\mathbf{x}_i)| < \varepsilon$ to $|h(\mathbf{x}_i)| = \varepsilon$. Check the variation value (L_i^R) of each sample \mathbf{x}_i in set **R** to set **S**:

$$L_i^R = \frac{-h(\mathbf{x}_i) - \text{sign}(q\beta_i)\varepsilon}{\gamma_i} \tag{3.32}$$

The bookkeeping procedure is utilized to trace each sample in the training set against these five cases and determine the allowed $\Delta\lambda_c$ for each sample according to equation (3.24) or (3.26) [4]. $\Delta\lambda_c$ is calculated as the minimum absolute value among all possible $\Delta\lambda_c$. Thus increment of the current data is

$$\Delta\lambda_c = q \min(|L_{c1}|, |L_{c2}|, |\mathbf{L}^S|, |\mathbf{L}^E|, |\mathbf{L}^R|) \tag{3.33}$$

where $q = \text{sign}(-h(\mathbf{x}_c))$ and L_{c1}, L_{c2} are variations of the current sample and $\mathbf{L}^S = [L_i^S, i \in \mathbf{S}]$, $\mathbf{L}^E = [L_i^E, i \in \mathbf{E}]$, $\mathbf{L}^R = [L_i^R, i \in \mathbf{R}]$ are the variations of the \mathbf{x}_i data in sets **S**, **E**, **R** respectively [2].

3.3 Adaptive Online SVR Controller based on Model Estimated by an Online SVR

3.3.1 An overview of the proposed control architecture

The aim in adaptive control is to design a flexible controller that changes its parameters depending on alternation of system behaviour. The proposed adaptive SVR controller structure is depicted in Figure 3.2.

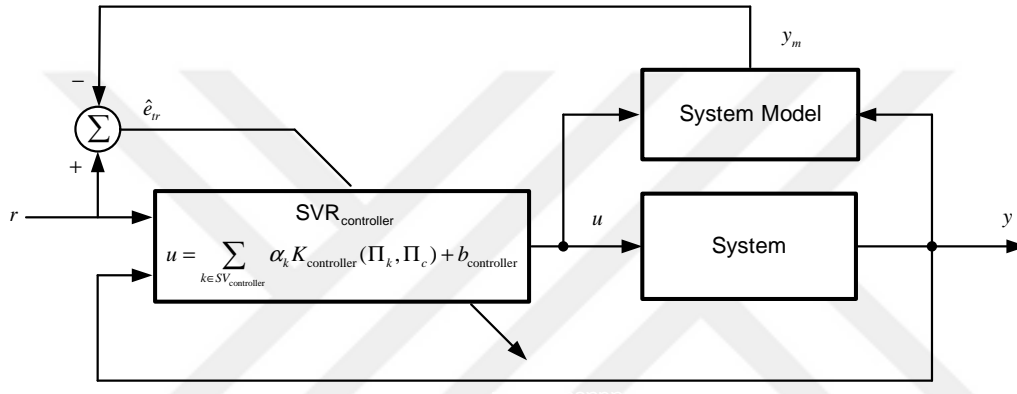


Figure 3.2 : The block diagram of the proposed control architecture.

The control methodology is based on estimating the model of the system to be controlled, predicting the system output with the obtained model, approximating the tracking error and designing the controller using the tracking error. Since the parameters of the controller are adjusted via approximated tracking error, system model is utilized as part of the adjustment mechanism for SVR controller. The tuning method of adaptive SVR controller based on estimated system model is illustrated in Figure 3.3. There are two SVR structures in the proposed architecture: $SVR_{\text{controller}}$ computes the control input to be applied to the system and SVR_{model} obtains an estimate of the system model. The online $SVR_{\text{controller}}$ computes a control signal as:

$$u_n = \sum_{k \in SV_{\text{controller}}} \alpha_k K_{\text{controller}}(\mathbf{\Pi}_c, \mathbf{\Pi}_k) + b_{\text{controller}} \quad (3.34)$$

where $\mathbf{\Pi}_c$ is input vector, $K_{\text{controller}}(.,.)$ is the kernel, α_k , $\mathbf{\Pi}_k$ and $b_{\text{controller}}$ are the parameters of the controller to be tuned at time index n . SVR_{model} is employed to

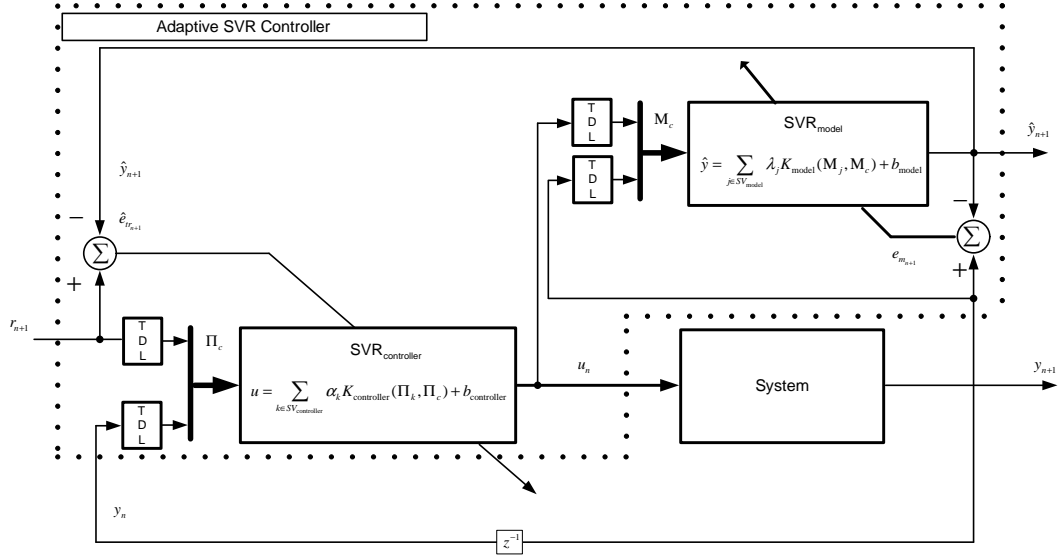


Figure 3.3 : The adaptation mechanisms of $SVR_{\text{controller}}$ and SVR_{model} .

forecast system behaviour and model output is calculated as

$$\hat{y}_{n+1} = \sum_{j \in SV_{\text{model}}} \lambda_j K_{\text{model}}(\mathbf{M}_c, \mathbf{M}_j) + b_{\text{model}} \quad (3.35)$$

where K_{model} is the kernel matrix of the system model, \mathbf{M}_c is current input, and λ_j, \mathbf{M}_j and b_{model} are the parameters of the system model to be adjusted. As shown in Figure 3.3, while SVR_{model} parameters are adjusted via modelling error $e_{m,n+1}$, $SVR_{\text{controller}}$ is optimized via approximated tracking error $\hat{e}_{tr,n+1}$. $SVR_{\text{controller}}$ and SVR_{model} are both used online to perform learning, prediction and control consecutively. When the parameters of $SVR_{\text{controller}}$ are optimized, in order to calculate and observe the impact of the computed control signal (u_n) on system behaviour and train $SVR_{\text{controller}}$ precisely, the computed control signal is applied to SVR_{model} at every step of training phase of the controller to predict behaviour of the system (y_{n+1}). Ideally, during the course of online working, it is expected that \hat{y}_{n+1} will eventually converge to y_{n+1} . After the training phase for $SVR_{\text{controller}}$ is concluded, the control signal is applied to the system. Thus, the input of system model \mathbf{M}_c and output y_{n+1} can be computed for training phase of system model.

The proposed control algorithm can be summarized as follows (in the algorithm given below u_n^- indicates the control signal predicted via controller parameters obtained at the previous step and u_n^+ indicates the control signal estimated via trained controller parameters at the current step):

Step 1: Initialize $SVR_{\text{controller}}$ and SVR_{model} parameters.

-SVR_{controller}(controller) parameters :

$$\alpha_k = b_{\text{controller}} = 0$$

-SVR_{model} (system model) parameters :

$$\lambda_j = b_{\text{model}} = 0$$

Step 2: Prediction step for controller (u_n^-)

-Set index to n.

-Constitute feature vector for controller ($\mathbf{\Pi}_c$). Some possible choices for controller feature vector are given as follows :

$$\mathbf{\Pi}_c = [r_n \cdots r_{n-n_r}, y_n \cdots y_{n-n_y}]^T$$

$$\mathbf{\Pi}_c = [P_n, I_n, D_n]^T$$

where $P_n = e_n - e_{n-1}$, $I_n = e_n$, $D_n = e_n - 2e_{n-1} + e_{n-2}$ and $e_n = r_n - y_n$. A combination of the reference signal, system output and controller output can also be used as a feature vector:

$$\mathbf{\Pi}_c = [P_n, I_n, D_n, r_n \cdots r_{n-n_r}, y_n \cdots y_{n-n_y}, u_{n-1} \cdots u_{n-n_u}]^T$$

where n_r , n_y and n_u indicate the number of the past behavior of the features.

-Calculate the control signal(u_n^-) produced by the

SVR_{controller} via controller parameters trained at the previous step (n-1) via (3.34).

Step 3: Prediction step for system model

-Constitute feature vector for system model (\mathbf{M}_c).

$$\mathbf{M}_c = [u_n^- \cdots u_{n-n_u}, y_n \cdots y_{n-n_y}]^T$$

-Apply u_n^- to SVR_{model} and calculate \hat{y}_{n+1} via (3.35)

Step 4: Learning step for controller

-Calculate $\hat{e}_{tr_{n+1}} = r_{n+1} - \hat{y}_{n+1}$

If $|\hat{e}_{tr_{n+1}}| > \epsilon_{\text{closed-loop}}$

-Train controller parameters via $\hat{e}_{tr_{n+1}} = r_{n+1} - \hat{y}_{n+1}$

else

-Continue with controller parameters obtained at previous step

Note: In step 4, the closed loop margin is tried to optimized via system model. Thus, on one hand the margin is optimized, so the tracking error is minimized, on the other hand the optimal parameters for controller are estimated indirectly.

Step 5: Calculation of control input by trained controller (u_n^+)

-Calculate the control signal (u_n^+) produced by the trained SVR_{controller} via (3.34).

Step 6: Apply u_n^+ to system to calculate y_{n+1} .

Step 7: Prediction step for system model

-Apply u_n^+ to model and calculate \hat{y}_{n+1} via (3.35).

Step 8: Learning Step for System Model

-Calculate $e_{m_{n+1}} = y_{n+1} - \hat{y}_{n+1}$

If $|e_{m_{n+1}}| > \epsilon_{model}$

-Train system model where $e_{m_{n+1}} = y_{n+1} - \hat{y}_{n+1}$

else

-Continue with system model parameters obtained at previous step

Step 9: Increment $n = n + 1$ and go to step 2.

3.3.2 Generalized closed-loop system margin

As mentioned in section 3.2, in order to obtain the optimal parameters which minimize the estimation error for SVR_{model} , the margin function in (3.13) should be minimized, given training data pairs (\mathbf{M}_c, y_{n+1}) as depicted in Figure 3.3. The major problem arises when we want to use SVR as a controller since although the inputs to the $SVR_{controller}$ are available $(\mathbf{\Pi}_c)$, the designer does not know the output of $SVR_{controller}$, namely the control input to be applied to the system in advance. This is the main problem that must be solved in order to use SVR as a controller. In this section, in order to describe training $SVR_{controller}$ with unknown outputs, we give two main definitions: open loop and closed-loop margins which are both crucial to understand how to train $SVR_{controller}$.

Definition 1 (Open-loop Margin): In this paper, the term "open-loop margin" is used to denote the regression margin related to SVR_{model} . In modelling, the aim is to minimize the error $e_{m_{n+1}} = y_{n+1} - \hat{y}_{n+1}$ given in Figure 3.3, for the feedforward system model. This is the error between the actual system output and the output of the "learned model". This model is required to attain a prior knowledge about how the system will respond to the adjusted parameters of the controller.

Definition 2 (Closed-loop Margin): The controller aims to force the controlled system to follow the reference signal closely, so it is designed to minimize the tracking error, which implies that the margin function of the controller depends on tracking error. The parameters of the controller are adjusted via tracking error. Online SVR_{model} is utilized to approximate system dynamics and compute the tracking error in the next step which

in turn is used to tune the parameters of the $\text{SVR}_{\text{controller}}$, $\text{SVR}_{\text{model}}$ also helps to assess whether the computed control signal can successfully attain reference tracking or not in the training phase. Hence both $\text{SVR}_{\text{model}}$ and $\text{SVR}_{\text{controller}}$ are effective in determining the controller margin. The closed-loop performance of the system is affected by both the modelling error and the tracking error, hence we define "closed-loop margin", which is a function of the controller and system model margins. The designer does not have direct control on the margin of the controller which emerges from the combined effects of the margin of the closed-loop system and margin of the system model. Considering $\text{SVR}_{\text{controller}}$ and $\text{SVR}_{\text{model}}$ independently, the margin of each subsystem is illustrated in Figure 3.4. Controller margin is depicted in Figure 3.4, part (a), and system model margin is shown in Figure 3.4, part (b) where $f_{\text{controller}}$ and f_{model} denote the regression functions of controller and system model, respectively. As can be seen from Figure 3.4, the input of system model estimator ($\text{SVR}_{\text{model}}$) is \mathbf{M}_c and its output is y_{n+1} while the input to the controller ($\text{SVR}_{\text{controller}}$) is $\mathbf{\Pi}_c$ and the output is u_n . In Figure 3.4, the axes for the input and output of $\text{SVR}_{\text{model}}$ are denominated as \mathbf{M} and Y_{sys} while the axes for $\text{SVR}_{\text{controller}}$ are named as $\mathbf{\Pi}$ and U . In Figure 3.5, subgraphs related to $\text{SVR}_{\text{model}}$ and $\text{SVR}_{\text{controller}}$ depicted in Figure 3.4 are combined to yield a three dimensional graph.

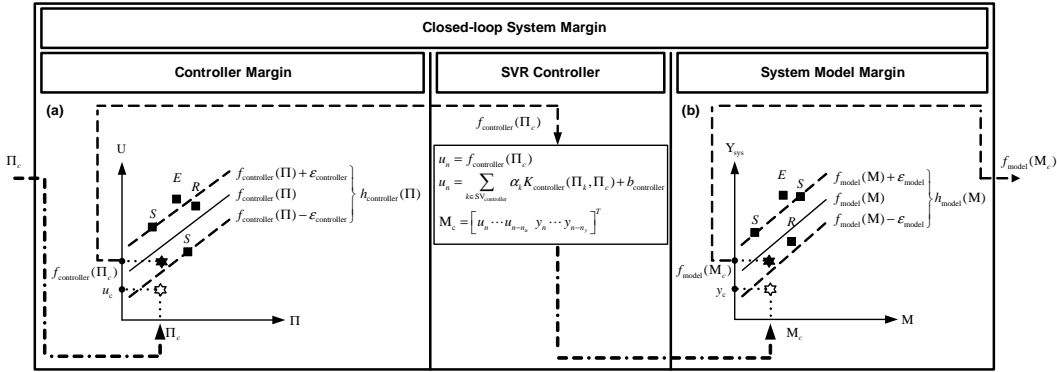


Figure 3.4 : Margins of $\text{SVR}_{\text{controller}}$ (a) and $\text{SVR}_{\text{model}}$ (b).

The horizontal axes, $\mathbf{\Pi}$ and \mathbf{M} stand for the input and output of $\text{SVR}_{\text{controller}}$ while \mathbf{M} and vertical axis Y_{sys} represent the input and output of $\text{SVR}_{\text{model}}$. Since vector \mathbf{M} includes u_n , the horizontal regression surface representing $\text{SVR}_{\text{controller}}$ can be depicted with axes $\mathbf{\Pi}$ and \mathbf{M} instead of $\mathbf{\Pi}$ and U . It must be noted that in the figures, for the sake of visualisation, all inputs and outputs are assumed to be one dimensional vectors. In real applications, the vectors will generally be multi-dimensional, resulting

in hypersurfaces. The closed loop margin which is the combination of the controller and system model margins is illustrated in Figure 3.5, in Π , M and Y_{sys} axes.

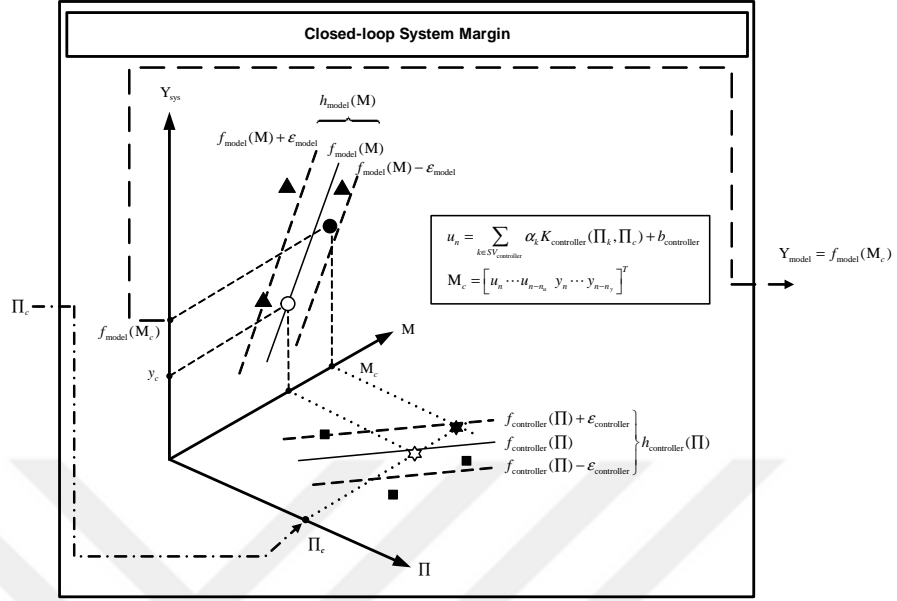


Figure 3.5 : Closed-loop system margin in three dimensions.

When the whole control structure is working online, the margins of the controller and model are fused, and closed loop margin between closed loop input and output is intuitively thought as a single margin, as depicted in Figure 3.6. In other words, the margin function of the system model and controller are embedded in the margin of the closed-loop system. In Figure 3.6, controller and system model margins are combined to yield the "closed-loop margin" and this is projected onto Y_{sys} - Π axes. The figure illustrates this projection before and after online training. Closed-loop margin is represented as $h_{closed-loop}(\Pi)$ which is a function of system model margin ($h_{model}(M)$) and controller margin ($h_{controller}(\Pi)$).

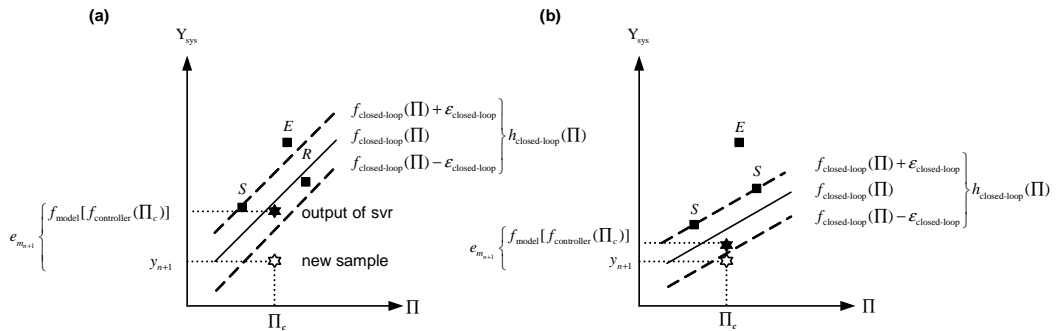


Figure 3.6 : Projected closed loop margin before (a) and after (b) training.

As mentioned previously, the aim in closed-loop control is to track the reference signal by minimizing the tracking error $e_{tr_{n+1}} = r_{n+1} - y_{n+1}$ via control and optimization techniques. To make the system output follow the reference signal with minimum error, closed-loop margin must be optimized. Also, the margin of $\text{SVR}_{\text{model}}$ is optimized independently from the closed loop system margin, since ϵ_{model} and $\epsilon_{\text{closed-loop}}$, the upper value of tolerable error for $\text{SVR}_{\text{model}}$ and the overall closed loop system ($\text{SVR}_{\text{model}}$ and $\text{SVR}_{\text{controller}}$ combined), respectively are set independently by the designer. However, the designer cannot set $\epsilon_{\text{controller}}$, the upper value of tolerable error for $\text{SVR}_{\text{controller}}$, and therefore does not have a direct influence on the margin of the controller, but controller margin can only be affected indirectly through the combined actions of $\text{SVR}_{\text{model}}$ and the controlled closed loop structure. Since the main goal in our system is to effectively minimize the tracking error, we aim to optimize primarily the closed loop margin. Of equivalent importance is the optimization of the $\text{SVR}_{\text{model}}$ margin, since for good closed-loop control performance we need a system model with minimum modelling error. We can deduce that optimization of closed-loop and $\text{SVR}_{\text{model}}$ margins will spontaneously lead to the optimization of $\text{SVR}_{\text{controller}}$ margin so the parameters of $\text{SVR}_{\text{controller}}$ can be obtained indirectly while we try to minimize tracking error (or equivalently we optimize closed-loop margin). In training phase of $\text{SVR}_{\text{model}}$, since $e_{m_{n+1}} = y_{n+1} - \hat{y}_{n+1}$ and system model is to forced to track y_{n+1} , input-output data pair is $(\mathbf{M}_{\mathbf{c}}, y_{n+1})$ as shown in Figure 3.3. Therefore, the input and output axes for regression surface are named as \mathbf{M} and Y_{sys} . On the other hand, the closed-loop system is forced to track the reference input, r_{n+1} , in training phase, so closed-loop training data pair is $(\mathbf{\Pi}_{\mathbf{c}}, r_{n+1})$. Based on this, the input and output axes for closed-loop system regression surface are termed as $\mathbf{\Pi}$ and R in Figure 3.7 and axis R is utilized in place of Y_{sys} for closed-loop system in Figures 3.7-3.9. If the system margin on $R - \mathbf{M}$ plane is dislocated throughout $\mathbf{\Pi} - \mathbf{M}$ plane, the closed-loop margin will have a prismatical shape where the optimal controller parameters are searched as in Figure 3.9.

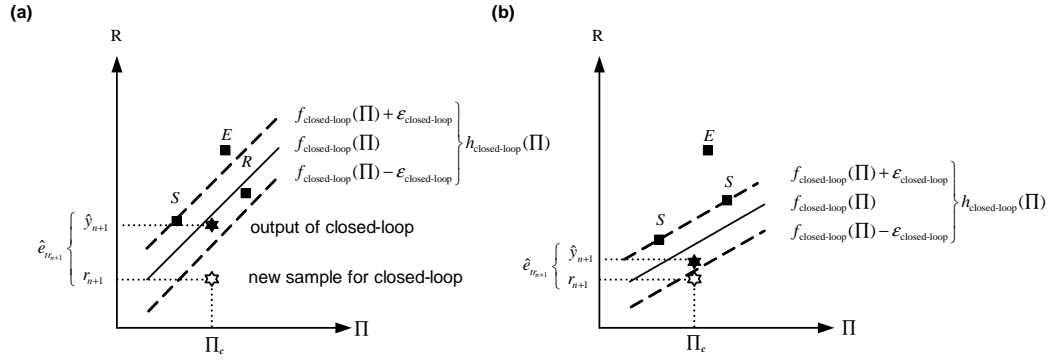


Figure 3.7 : Projected closed loop margin before (a) and after (b) training.

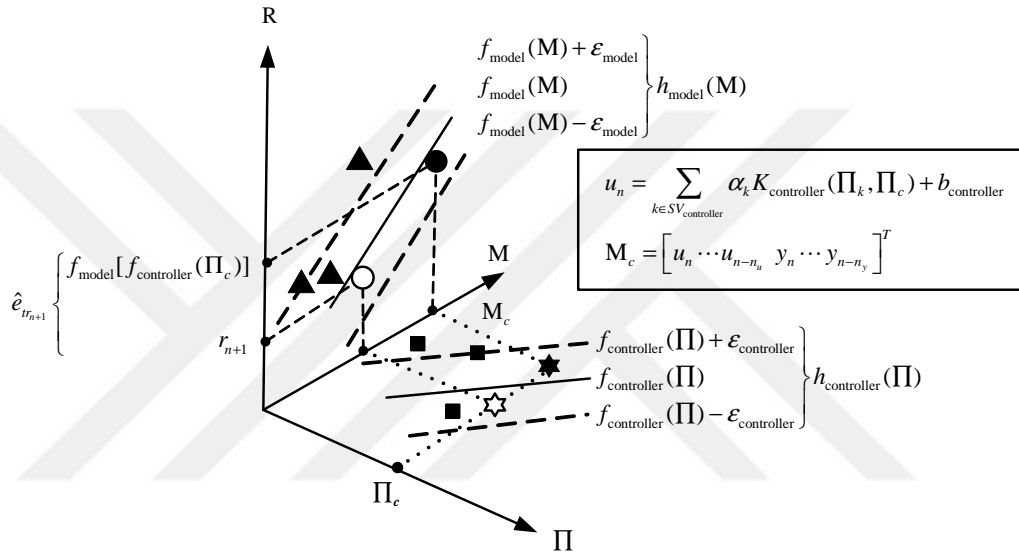


Figure 3.8 : Closed-loop system margin in three dimensions.

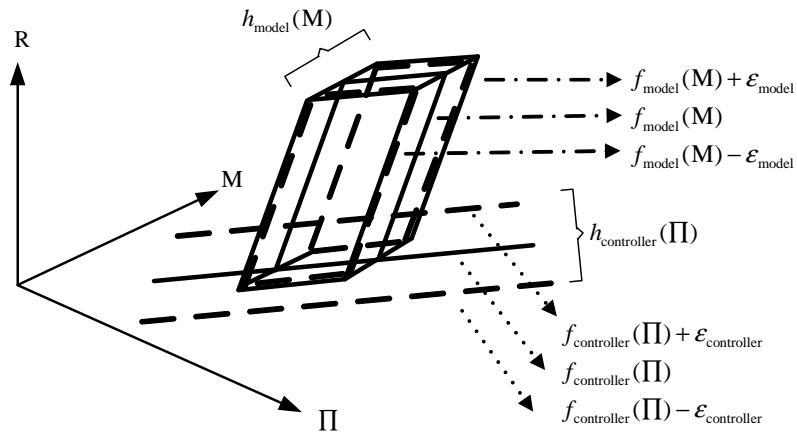


Figure 3.9 : Combined controller and system model margins in three dimensions.

3.3.3 Online support vector regression for controller design

Consider a training data set for closed-loop system as:

$$\mathbf{T} = \{\mathbf{\Pi}_i, r_{i+1}\}_{i=1}^N \quad \mathbf{\Pi}_i \in \mathbf{\Pi} \subseteq R^n, r_{i+1} \in R \quad (3.36)$$

where N is the size of the training data, n is the dimension of the input, $\mathbf{\Pi}_i$ is input feature vector of controller and r_{i+1} is the reference signal that system is forced to track, the closed-loop margin function of the system for the i th sample $\mathbf{\Pi}_i$ can be defined as:

$$h_{\text{closed-loop}}(\mathbf{\Pi}_i) = \hat{y}_{i+1} - r_{i+1} = f_{\text{model}}(\mathbf{M}_i) - r_{i+1} \quad (3.37)$$

where

$$\begin{aligned} \hat{y}_{i+1} &= f_{\text{model}}(\mathbf{M}_i) \\ &= \sum_{j \in SV_{\text{model}}} \lambda_j K_{\text{model}}(\mathbf{M}_j, \mathbf{M}_i) + b_{\text{model}} \\ \mathbf{M}_i &= [u_i \cdots u_{i-n_u}, y_i \cdots y_{i-n_y}] \\ u_i &= f_{\text{controller}}(\mathbf{\Pi}_i) = \sum_{k \in SV_{\text{controller}}} \alpha_k K_{\text{controller}}(\mathbf{\Pi}_k, \mathbf{\Pi}_i) + b_{\text{controller}} \\ \mathbf{\Pi}_i &= [r_i \cdots r_{i-n_r}, y_i \cdots y_{i-n_y}, u_{i-1} \cdots u_{i-n_u}] \end{aligned}$$

In the learning stage of the controller, the system model is fixed and system model parameters are known, so the closed loop margin can be rewritten as

$$h_{\text{closed-loop}}(\mathbf{\Pi}_i) = \hat{y}_{i+1} - r_{i+1} = f_{\text{closed-loop}}(\mathbf{\Pi}_i) - r_{i+1} = -\hat{e}_{tr_{i+1}} \quad (3.38)$$

with respect to an input-output data pair of closed-loop system $(\mathbf{\Pi}_i, r_{i+1})$ where $f_{\text{closed-loop}}$ is the approximated output of the closed-loop system. Thus, using $(\mathbf{\Pi}_i, r_{i+1})$ data pair and closed-loop margin in (3.38), online learning formulations for controller can be derived. The basic idea is to change the coefficient α_c corresponding to the new sample $\mathbf{\Pi}_c$ in a finite number of discrete steps until it meets the KKT conditions while ensuring that the existing samples in \mathbf{T} continue to satisfy the KKT conditions at each step [4]. The following situations construct the basics of convergence and migration of

the closed-loop data.

$$\begin{aligned}
h_{\text{closed-loop}}(\mathbf{\Pi}_i) &\geq \varepsilon_{\text{closed-loop}}, \alpha_i = -C_{\text{closed-loop}} \\
h_{\text{closed-loop}}(\mathbf{\Pi}_i) &= \varepsilon_{\text{closed-loop}}, -C_{\text{closed-loop}} < \alpha_i < 0 \\
-\varepsilon_{\text{closed-loop}} &\leq h_{\text{closed-loop}}(\mathbf{\Pi}_i) \leq \varepsilon_{\text{closed-loop}}, \alpha_i = 0 \\
h_{\text{closed-loop}}(\mathbf{\Pi}_i) &= -\varepsilon_{\text{closed-loop}}, 0 < \alpha_i < C_{\text{closed-loop}} \\
h_{\text{closed-loop}}(\mathbf{\Pi}_i) &\leq -\varepsilon_{\text{closed-loop}}, \alpha_i = C_{\text{closed-loop}}
\end{aligned} \tag{3.39}$$

The optimal controller parameters, α_k , $b_{\text{controller}}$ are sought within $\varepsilon_{\text{closed-loop}}$ tube by minimizing the tracking error using the online learning algorithm given in section 3.2.3. Equations (3.19-3.26) can be modified for training the closed-loop system using α , $b_{\text{controller}}$, $h_{\text{closed-loop}}$, $\varepsilon_{\text{closed-loop}}$, $\mathbf{\Pi}_i$ and $K_{\text{controller}}$ in place of λ , b , h , ε , \mathbf{x}_i and K . For this purpose, the training data set can be dissociated into \mathbf{E} , \mathbf{S} and \mathbf{R} subsets using (3.39) as follows:

$$\begin{aligned}
\mathbf{E} &= \{i \mid |\alpha_i| = C_{\text{closed-loop}}\} \\
\mathbf{S} &= \{i \mid 0 < |\alpha_i| < C_{\text{closed-loop}}\} \\
\mathbf{R} &= \{i \mid |\alpha_i| = 0\}
\end{aligned} \tag{3.40}$$

When a new input vector ($\mathbf{\Pi}_c$) with Lagrange multiplier α_c is trained without migration of vectors between sets \mathbf{S} , \mathbf{E} and \mathbf{R} , the relation among $(\Delta\alpha_c)$ for current data, the variation in $\Delta h_{\text{closed-loop}}(\mathbf{\Pi}_i)$ and α_i are derived as in (3.41) via (3.13-3.16, 3.37- 3.39) [4, 55].

$$\Delta h_{\text{closed-loop}}(\mathbf{\Pi}_i) = K_{\text{controller}_{ic}} \Delta\alpha_c + \sum_{j=1}^N K_{\text{controller}_{ij}} \Delta\alpha_j + \Delta b_{\text{controller}} \tag{3.41}$$

α_c must satisfy the the dual constraints in (3.11) as follows:

$$\alpha_c + \sum_{j=1}^N \alpha_j = 0 \tag{3.42}$$

When we isolate the term $\Delta\alpha_c$ in equations (3.41) and (3.42) [55], the variations of Lagrange multipliers for the data in support vector set \mathbf{S} can be calculated for given $\Delta\alpha_c$ as:

$$\begin{aligned}
\sum_{j=1}^N K_{\text{controller}_{ij}} \Delta\alpha_j + \Delta b_{\text{controller}} &= -K_{\text{controller}_{ic}} \Delta\alpha_c \\
\sum_{j \in SV_{\text{controller}}} \Delta\alpha_j &= -\Delta\alpha_c
\end{aligned} \tag{3.43}$$

Defining the samples in support vector set \mathbf{S} as in

$$\mathbf{S} = \{s_1, s_2, s_3, \dots, s_k\} \quad (3.44)$$

(3.43) can be rearranged in matrix form as

$$\begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & K_{\text{controller}_{s_1 s_1}} & \cdots & K_{\text{controller}_{s_1 s_k}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & K_{\text{controller}_{s_k s_1}} & \cdots & K_{\text{controller}_{s_k s_k}} \end{bmatrix} \begin{bmatrix} \Delta b_{\text{controller}} \\ \Delta \alpha_{s_1} \\ \vdots \\ \Delta \alpha_{s_k} \end{bmatrix} = - \begin{bmatrix} 1 \\ K_{\text{controller}_{s_1 c}} \\ \vdots \\ K_{\text{controller}_{s_k c}} \end{bmatrix} \Delta \alpha_c \quad (3.45)$$

Thus, $\Delta \alpha$ is adjusted for computed $\Delta \alpha_c$ as

$$\Delta \alpha = \begin{bmatrix} \Delta b_{\text{controller}} \\ \Delta \alpha_{s_1} \\ \vdots \\ \Delta \alpha_{s_k} \end{bmatrix} = \beta \Delta \alpha_c \quad (3.46)$$

where

$$\beta = \begin{bmatrix} \beta \\ \beta_{s_1} \\ \vdots \\ \beta_{s_k} \end{bmatrix} = -\Theta \begin{bmatrix} 1 \\ K_{\text{controller}_{s_1 c}} \\ \vdots \\ K_{\text{controller}_{s_k c}} \end{bmatrix}, \quad \Theta = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & K_{\text{controller}_{s_1 s_1}} & \cdots & K_{\text{controller}_{s_1 s_k}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & K_{\text{controller}_{s_k s_1}} & \cdots & K_{\text{controller}_{s_k s_k}} \end{bmatrix}^{-1} \quad (3.47)$$

Depending on $\Delta \alpha_c$, only the margin values of the non-support samples transume and the alternation in margin for non-support samples can be updated via (3.48)

$$\begin{bmatrix} \Delta h_{\text{closed-loop}}(\mathbf{\Pi}_{n_1}) \\ \Delta h_{\text{closed-loop}}(\mathbf{\Pi}_{n_2}) \\ \vdots \\ \Delta h_{\text{closed-loop}}(\mathbf{\Pi}_{n_z}) \end{bmatrix} = \gamma \Delta \lambda_c \quad (3.48)$$

$$\gamma = \begin{bmatrix} K_{\text{controller}_{n_1 c}} \\ K_{\text{controller}_{n_2 c}} \\ \vdots \\ K_{\text{controller}_{n_z c}} \end{bmatrix} + \begin{bmatrix} 1 & K_{\text{controller}_{n_1 s_1}} & \cdots & K_{\text{controller}_{n_1 s_l}} \\ 1 & K_{\text{controller}_{n_2 s_1}} & \cdots & K_{\text{controller}_{n_2 s_l}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & K_{\text{controller}_{n_z s_1}} & \cdots & K_{\text{controller}_{n_z s_l}} \end{bmatrix} \beta$$

where n_1, n_2, \dots, n_z are the indices of non-support samples, γ are margin sensitivities and $\gamma = 0$ for samples in \mathbf{S} . The alternation of the matrix Θ for learning and forgetting stages and detailed information related to the recursive algorithm can be accessed via [2,4,55]. Up to this point, it is assumed that $\Delta \alpha_c$ is known, and update rules for the parameters of all data except for current data are derived. As given in section 3.2.4, the increment for current data ($\Delta \alpha_c$) is defined as the one with minimum absolute

value among all possible $\Delta\alpha_c$ as follows [4]:

$$\Delta\alpha_c = q \min(|L_{c_1}|, |L_{c_2}|, |\mathbf{L}^S|, |\mathbf{L}^E|, |\mathbf{L}^R|) \quad (3.49)$$

where $q = \text{sign}(-h_{\text{closed-loop}}(\mathbf{\Pi}_c)) = \text{sign}(e_{tr_{n+1}})$ and L_{c_1} , L_{c_2} are variations of the current sample and $\mathbf{L}^S = [L_i^S, i \in S]$, $\mathbf{L}^E = [L_i^E, i \in E]$, $\mathbf{L}^R = [L_i^R, i \in R]$ are the variations of the $\mathbf{\Pi}_i$ data in sets \mathbf{S} , \mathbf{E} , \mathbf{R} , respectively [2].

3.3.4 Stability analysis of the closed-loop system

In order to analyze the stability of the closed-loop system, the regression functions of the controller and the system model are given in matrix form. In this sequel, the control signal produced by controller is obtained as

$$u_n = f_{\text{controller}}(\boldsymbol{\alpha}, \mathbf{\Pi}_c) = \begin{bmatrix} b_{\text{controller}} \\ \alpha_1 \\ \vdots \\ \alpha_k \end{bmatrix}^T \begin{bmatrix} 1 \\ K_{1\Pi_c} \\ \vdots \\ K_{k\Pi_c} \end{bmatrix} = \boldsymbol{\alpha}^T K_{\text{controller}}(\mathbf{\Pi}_c) \quad (3.50)$$

The system model which is utilized to approximate system Jacobian is given as

$$\hat{y}_{n+1} = f_{\text{model}}(\boldsymbol{\lambda}, \mathbf{M}_c) = \begin{bmatrix} b_{\text{model}} \\ \lambda_1 \\ \vdots \\ \lambda_k \end{bmatrix}^T \begin{bmatrix} 1 \\ K_{1M_c} \\ \vdots \\ K_{kM_c} \end{bmatrix} = \boldsymbol{\lambda}^T K_{\text{model}}(\mathbf{M}_c) \quad (3.51)$$

For stability analysis the following Lyapunov function is employed

$$V(e_{tr_{n+1}}) = \frac{e_{tr_{n+1}}^T \mathbf{P} e_{tr_{n+1}}}{2} \quad (3.52)$$

where $\mathbf{P} = \mathbf{I}$ (identity matrix). Both the stability of the system and the convergence of the controller are guaranteed when $\frac{\partial V}{\partial t} \leq 0$ [68]. Thus,

$$\frac{\partial V(e_{tr_{n+1}})}{\partial t} = e_{tr_{n+1}}^T \mathbf{P} \dot{e}_{tr_{n+1}} \quad (3.53)$$

where $\dot{e}_{tr_{n+1}} = \frac{\partial e_{tr_{n+1}}}{\partial t} = \frac{\partial e_{tr_{n+1}}}{\partial u_n} \frac{\partial u_n}{\partial t}$. Let us consider a small deviation from the equilibrium point, which corresponds to local stability analysis using equation (3.54).

The incremental change in the control signal is computed as

$$\Delta u_n = \left[\frac{\partial f_{\text{controller}}(\boldsymbol{\alpha}, \mathbf{\Pi}_c)}{\partial \boldsymbol{\alpha}} \right]^T \Delta \boldsymbol{\alpha} + \left[\frac{\partial f_{\text{controller}}(\boldsymbol{\alpha}, \mathbf{\Pi}_c)}{\partial \mathbf{\Pi}_c} \right]^T \Delta \mathbf{\Pi}_c \quad (3.54)$$

where $\mathbf{\Pi}_c$ and $f_{\text{controller}}$ are the input and the output of the controller respectively. If (3.54) is substituted in (3.53), the equation (3.53) can be rewritten as

$$\begin{aligned} \frac{\partial V(e_{tr_{n+1}})}{\partial t} &= e_{tr_{n+1}}^T \mathbf{P} \frac{\partial e_{tr_{n+1}}}{\partial u_n} \left[\frac{\partial f_{\text{controller}}(\boldsymbol{\alpha}, \mathbf{\Pi}_c)}{\partial \boldsymbol{\alpha}} \right]^T \Delta \boldsymbol{\alpha} \\ &+ e_{tr_{n+1}}^T \mathbf{P} \frac{\partial e_{tr_{n+1}}}{\partial u_n} \left[\frac{\partial f_{\text{controller}}(\boldsymbol{\alpha}, \mathbf{\Pi}_c)}{\partial \mathbf{\Pi}_c} \right]^T \Delta \mathbf{\Pi}_c \end{aligned} \quad (3.55)$$

with $\Delta \mathbf{\Pi}_c \cong e_{tr_{n+1}}$. As can be seen in (3.55), stability depends on the increments in Lagrange multipliers of the controller ($\Delta \boldsymbol{\alpha}$) given in (3.46). The increment for current data is

$$\Delta \alpha_c = q \min(|L_{c1}|, |L_{c2}|, |\mathbf{L}^S|, |\mathbf{L}^E|, |\mathbf{L}^R|) \quad (3.56)$$

where $q = \text{sign}(-h_{\text{closed-loop}}(\mathbf{\Pi}_c)) = \text{sign}(e_{tr_{n+1}})$ and L_{c1} , L_{c2} are variations of the current sample and $\mathbf{L}^S = [L_i^S, i \in \mathbf{S}]$, $\mathbf{L}^E = [L_i^E, i \in \mathbf{E}]$, $\mathbf{L}^R = [L_i^R, i \in \mathbf{R}]$ are the variations of the $\mathbf{\Pi}_i$ data in sets \mathbf{S} , \mathbf{E} , \mathbf{R} , respectively [2]. As explained in section 3.2.4, $\min(|L_{c1}|, |L_{c2}|, |\mathbf{L}^S|, |\mathbf{L}^E|, |\mathbf{L}^R|)$ term is a positive function of $e_{tr_{n+1}}$, $\boldsymbol{\alpha}$ and C . Thus, $\Delta \alpha_c$ can be written as:

$$\begin{aligned} \Delta \alpha_c &= q \min(|L_{c1}|, |L_{c2}|, |\mathbf{L}^S|, |\mathbf{L}^E|, |\mathbf{L}^R|) = \text{sign}(e_{tr_{n+1}}) \Psi(e_{tr_{n+1}}, \alpha_i, C) \\ &= \frac{e_{tr_{n+1}}}{|e_{tr_{n+1}}|} \Psi(e_{tr_{n+1}}, \alpha_i, C) = e_{tr_{n+1}} \frac{\Psi(e_{tr_{n+1}}, \alpha_i, C)}{|e_{tr_{n+1}}|} \\ &= \mu(e_{tr_{n+1}}, \alpha_i, C) e_{tr_{n+1}} \end{aligned} \quad (3.57)$$

where $\mu(e_{tr_{n+1}}, \alpha_i, C) \geq 0$, $\Psi(e_{tr_{n+1}}, \alpha_i, C) \geq 0$. The update rule for all Lagrange multipliers in set \mathbf{S} is

$$\Delta \boldsymbol{\alpha} = \boldsymbol{\beta} \Delta \alpha_c = \boldsymbol{\beta} \mu(e_{tr_{n+1}}, \alpha_i, C) e_{tr_{n+1}} \quad (3.58)$$

If equation (3.58) is utilized in (3.55)

$$\frac{\partial V(e_{tr_{n+1}})}{\partial t} = e_{tr_{n+1}}^T (\mathbf{Q} + \mathbf{W}) e_{tr_{n+1}} \quad (3.59)$$

where

$$\begin{aligned} \mathbf{Q} &= \mathbf{P} \frac{\partial e_{tr_{n+1}}}{\partial u_n} \left[\frac{\partial f_{\text{controller}}(\boldsymbol{\alpha}, \mathbf{\Pi}_c)}{\partial \boldsymbol{\alpha}} \right]^T \boldsymbol{\beta} \mu(e_{tr_{n+1}}, \alpha_i, C) \\ \mathbf{W} &= \mathbf{P} \frac{\partial e_{tr_{n+1}}}{\partial u_n} \left[\frac{\partial f_{\text{controller}}(\boldsymbol{\alpha}, \mathbf{\Pi}_c)}{\partial \mathbf{\Pi}_c} \right]^T \end{aligned} \quad (3.60)$$

The term $\frac{\partial e_{tr_{n+1}}}{\partial u_n}$ can be expanded as $\frac{\partial e_{tr_{n+1}}}{\partial y_{n+1}} \frac{\partial y_{n+1}}{\partial u_n} = -\frac{\partial y_{n+1}}{\partial u_n}$. Thus,

$$\begin{aligned}
\mathbf{Q} &= \mathbf{P} \frac{\partial e_{tr_{n+1}}}{\partial u_n} \left[\frac{\partial f_{\text{controller}}(\boldsymbol{\alpha}, \mathbf{\Pi}_c)}{\partial \boldsymbol{\alpha}} \right]^T \boldsymbol{\beta} \mu(e_{tr_{n+1}}, \alpha_i, C) \\
&= -\mathbf{P} \frac{\partial y_{n+1}}{\partial u_n} [K_{\text{controller}}(\mathbf{\Pi}_c)]^T \boldsymbol{\beta} \mu(e_{tr_{n+1}}, \alpha_i, C) \\
&= -\mathbf{G} \\
\mathbf{W} &= \mathbf{P} \frac{\partial e_{tr_{n+1}}}{\partial u_n} \left[\frac{\partial f_{\text{controller}}(\boldsymbol{\alpha}, \mathbf{\Pi}_c)}{\partial \mathbf{\Pi}_c} \right]^T \\
&= -\mathbf{P} \frac{\partial y_{n+1}}{\partial u_n} \left[\frac{\partial f_{\text{controller}}(\boldsymbol{\alpha}, \mathbf{\Pi}_c)}{\partial \mathbf{\Pi}_c} \right]^T \\
&= -\mathbf{Z}
\end{aligned} \tag{3.61}$$

where $\frac{\partial y_{n+1}}{\partial u_n}$ is approximated via system model (f_{model}). As a result, the following equation must be satisfied for stability

$$\begin{aligned}
\frac{\partial V(e_{tr_{n+1}})}{\partial t} &= e_{tr_{n+1}}^T (\mathbf{Q} + \mathbf{W}) e_{tr_{n+1}} \leq 0 \\
&= -e_{tr_{n+1}}^T (\mathbf{G} + \mathbf{Z}) e_{tr_{n+1}} \leq 0
\end{aligned} \tag{3.62}$$

Thus, the stability conditions for closed-loop system can be summarized as follows :

- **Condition 1:** If $\mathbf{G} \geq 0$ and $\mathbf{Z} \geq 0$, the closed-loop system is stable
- **Condition 2:** If $\mathbf{G} \geq 0$ and $\mathbf{Z} \leq 0$ and $\|\mathbf{G}\| \geq \|\mathbf{Z}\|$, the closed-loop system is stable
- **Condition 3:** If $\mathbf{G} \leq 0$ and $\mathbf{Z} \geq 0$ and $\|\mathbf{Z}\| \geq \|\mathbf{G}\|$, the closed-loop system is stable

3.4 Simulation Results

The performance of the online SVR_{controller} based on system model estimated by SVR_{model} is assessed on third order continuously stirred tank reactor (CSTR) and bioreactor benchmark problems. In order to establish the feature vectors as inputs to SVR_{controller} and SVR_{model}, yielding the best performance, different features have been tested. Reference signal, system output and control signal are main features of a closed-loop system. New feature vectors to use as inputs for SVR_{controller} and SVR_{model} can also be generated as functions of these basic features such as tracking error, derivative of tracking error, etc. For instance, $\mathbf{\Pi}_c = [r_n \cdots r_{n-n_r}, y_n \cdots y_{n-n_y}, u_{n-1} \cdots u_{n-n_u}]^T$ can be chosen as a feature vector for controller where n_r , n_u and n_y indicate the number of the past features included in the vector.

While the closed-loop system performance may not be good enough with basic features (reference, system output, control input), utilizing tracking error or integral of tracking error as a feature may be imperative for diminutive steady state error. That is, the chosen states of closed-loop system as features directly affect closed-loop and controller performances. In our work, NARX model is employed to identify the dynamics of the systems and $\mathbf{M}_c = [u_n \cdots u_{n-n_u}, y_n \cdots y_{n-n_y}]^T$ is chosen as the input feature vector for $\text{SVR}_{\text{model}}$. The number of the past inputs (n_u) and past outputs (n_y), which are the order of NARX model of system, have been selected as 2 for both systems. Several different input feature vectors have been employed for $\text{SVR}_{\text{controller}}$ depending on the particular conditions of the closed loop system as explained in detail in the following subsections.

$\text{SVR}_{\text{controller}}$ and $\text{SVR}_{\text{model}}$ contain several parameters which have direct influence on controller performance. Tolerance parameters $\epsilon_{\text{closed-loop}}$ and ϵ_{model} are the most significant parameters in our control architecture based on ϵ -SVR. If they are chosen too large, $\text{SVR}_{\text{model}}$ cannot learn the dynamics of the system accurately and $\text{SVR}_{\text{controller}}$ may induce unacceptable control performance. If they are chosen too small, the result will be an increase in the number of support vectors. Therefore, the maximum tolerable errors for closed-loop system response and system model, $\epsilon_{\text{closed-loop}}$ and ϵ_{model} must be chosen so that an acceptable control quality [5] is obtained while the number of support vectors for both $\text{SVR}_{\text{controller}}$ and $\text{SVR}_{\text{model}}$ are kept at a reasonable level. Accordingly, we have chosen $\epsilon_{\text{closed-loop}} = \epsilon_{\text{model}} = 10^{-3}$ for both systems. As the number of support vectors increases, the control algorithm gets slower. In order to reduce the number of the support vectors as low as possible, C is fixed at a large value (1000). Selection of kernel parameters is also crucial in having good performance. In our simulations, we employed an exponential kernel function with $K(x, y) = e^{\frac{-\|x-y\|}{2\sigma^2}}$. We have observed that very small σ values cause fluctuations in control signal and system output since the number of the support vectors for $\text{SVR}_{\text{controller}}$ and $\text{SVR}_{\text{model}}$ increase to capture the system dynamics. On the other hand, large σ values give rise to loss of the nonlinearity of the kernel that leads to inaccurate identification of system dynamics. In this case, the control signal is too slow and the system cannot be forced to track reference signal accurately. In simulations for continuously stirred tank reactor system (CSTR), $\sigma = 1$ is used in both $\text{SVR}_{\text{controller}}$

and SVR_{model} . For bioreactor system $SVR_{\text{controller}}$ and SVR_{model} have been designed with $\sigma = 0.75$.

3.4.1 Simulation results for continuously stirred tank reactor system

Continuously stirred tank reactor system (CSTR) is a plant widely used in industry to produce polymers, pharmaceuticals, and other various chemical products. It is a type of chemical reactor in which isothermal, liquid-phase, successive multicomponent chemical reactions can be carried out [69, 70] and the contents are well stirred and uniform throughout [71]. Alternatively, it is also referred as a vat or backmix reactor [71, 72]. Consider a chemical reaction system given as



where the inlet reactants (A,B) are mixed in a vessel with constant volume via an agitator and return the product C. Two reaction sites occur in the chemical reaction given in (3.63). First one is among A-B, and second is between B-C. The aim in this chemical reaction system is to control the concentration of product C by altering the molar feed rate of reactant B. The dynamics of the third order, highly non-linear, time varying CSTR system are defined with the following set of differential equations by Kravaris and Palanki [69] as:

$$\begin{aligned} \dot{x}_1(t) &= 1 - x_1(t) - Da_1x_1(t) + Da_2x_2^2(t) \\ \dot{x}_2(t) &= -x_2(t) + Da_1x_1(t) - Da_2x_2^2(t) - Da_3d_2x_2^2(t) + u(t) \\ \dot{x}_3(t) &= -x_3(t) + Da_3d_2(t)x_2^2(t) \end{aligned} \quad (3.64)$$

where $x_1(t)$, $x_2(t)$ and $x_3(t)$ are states obtained from the concentrations of reactant A, middle reactant B and product C, respectively, $Da_1 = 3$, $Da_2 = 0.5$, $Da_3 = 1$, $u(t)$ is the control signal, $x_3(t)$ is the output of the system, $d_2(t)$ is the time-varying parameter of the system which represents the activity of the reaction, the nominal value of which is $d_{2\text{nominal}}(t) = 1$ [5, 43, 60, 70]. In the simulations, magnitude of the control signal is allowed to vary between $u_{\min} = 0$ and $u_{\max} = 1$; and its duration is kept constant at $\tau_{\min} = \tau_{\max} = 1$ s. The sampling period is chosen as 0.1 s. Fourth order Runge Kutta method has been used in the simulations. Since the control performance depends on the chosen features, various input feature vectors for controller can be employed according

to the particulars of the system, whether there is noise, disturbance, parametric uncertainty or not, etc.

3.4.1.1 Noiseless condition

The proposed control architecture is used to control the CSTR system briefly described above, and the tracking performance of the controller for a variable step signal, control signal produced by controller and alternation of the controller parameters are given in Figures 3.10-3.11. The figures depict that the system tracks the reference signals accurately. The input feature vectors for $SVR_{\text{controller}}$ for noiseless condition is chosen as $\Pi_c = [r_n, y_n]^T$.

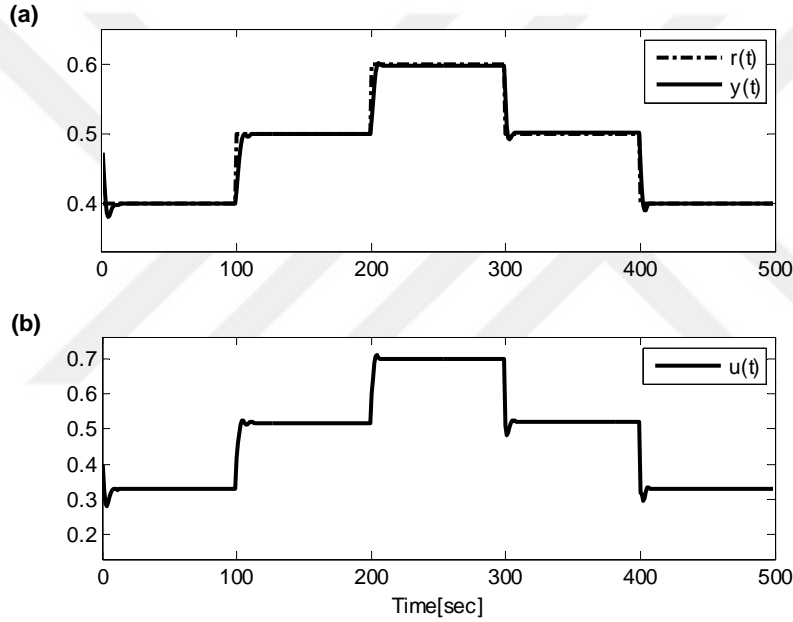


Figure 3.10 : System output (a), control signal (b) for variable step input.

In Figure 3.11, the first Lagrange multiplier and bias of $SVR_{\text{controller}}$ and SVR_{model} are illustrated to exemplify the inner learning mechanism of the SVR. As can be seen from Figure 3.11, the controller parameters and model can adapt themselves and learn new dynamics when reference signal changes. Number of the support vectors are also depicted in Figure 3.11 and it is observed that they do not change in certain times when learning is not required. The tracking performance of the controller for sinusoidal reference signal, control signal produced by controller and alternation of controller parameters are given in Figures 3.12-3.13 using the same input features for the controller and the system model.

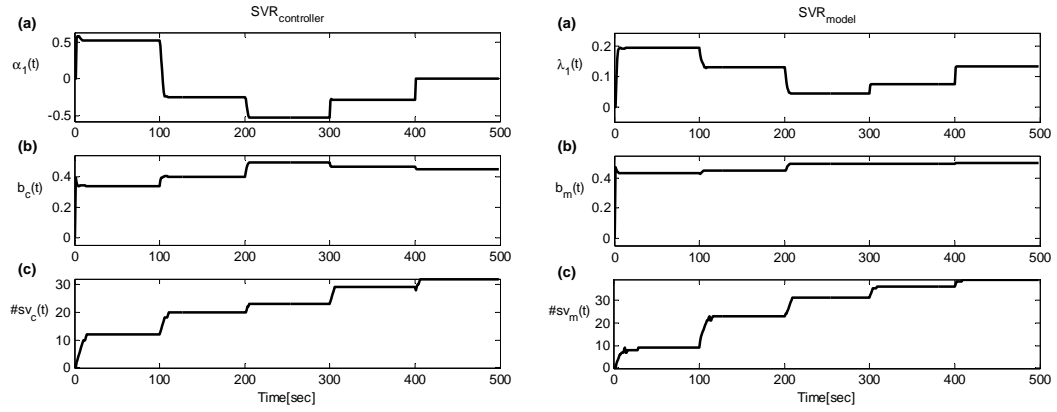


Figure 3.11 : Adaptation of SVR_{controller} parameters (left), SVR_{model} parameters (right).

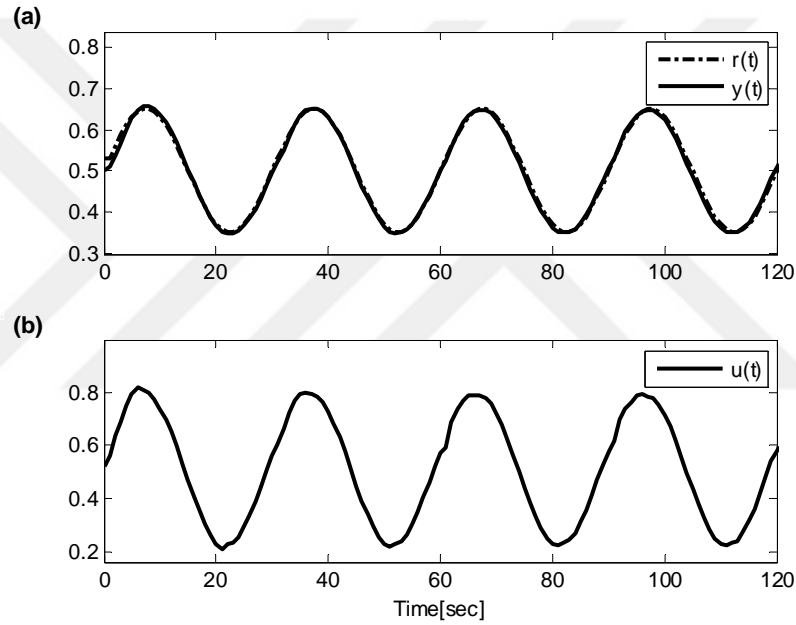


Figure 3.12 : System output (a), control signal (b) for sinusoidal input.

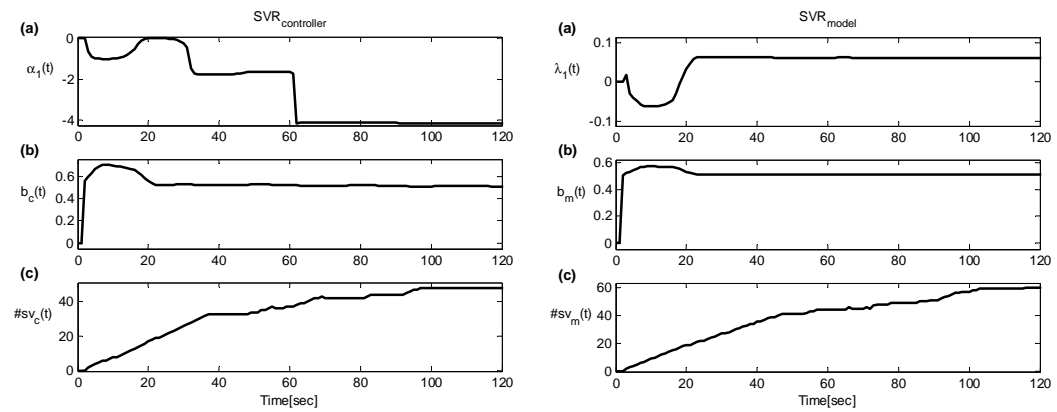


Figure 3.13 : Adaptation of SVR_{controller} parameters (left), SVR_{model} parameters (right).

3.4.1.2 Measurement noise

In order to evaluate and compare the robustness of $SVR_{\text{controller}}$ with respect to measurement noise, the measured output of system is corrupted by an additive zero mean Gaussian noise with a signal-to-noise ratio (SNR) of 30 dB [5]. SNR is given by

$$SNR = 10 \log_{10} \left(\frac{\sigma_y^2}{\sigma_u^2} \right) dB \quad (3.65)$$

where σ_y^2 and σ_u^2 are the variances of the measured output of the underlying system and the additive noise, respectively [5]. The input feature vectors for $SVR_{\text{controller}}$ is employed as $\mathbf{\Pi}_c = [r_n, y_n]^T$.

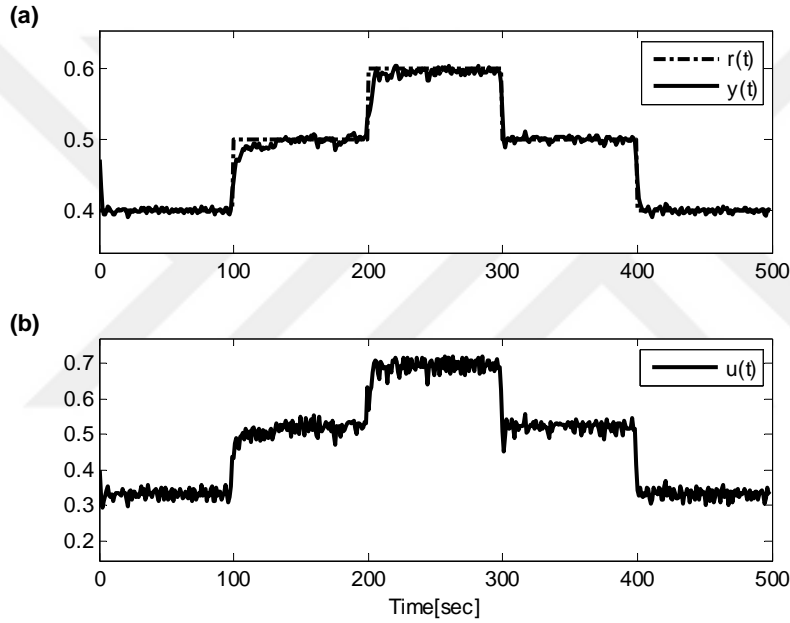


Figure 3.14 : System output (a), control signal (b) for variable step input.

Figure 3.14 shows the tracking performance and control input for $SVR_{\text{controller}}$ with Gaussian measurement noise added to the system. The tracking performance of the closed loop system and alternation of the controller parameters in noisy case are as in Figures 3.14-3.15.

3.4.1.3 Uncertainty in system parameters

In order to examine the robustness of the controller in terms of parameter uncertainty, the performance of the controller is examined under a time-varying system parameter.

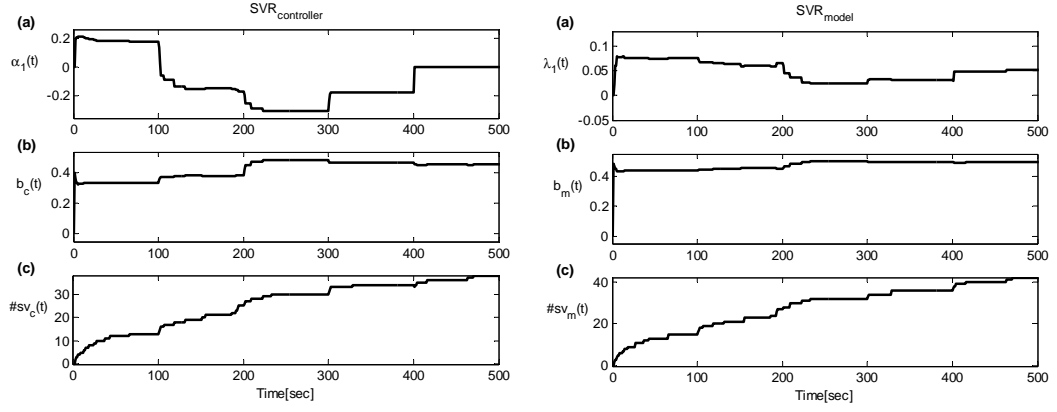


Figure 3.15 : Adaptation of SVR_{controller} parameters (left), SVR_{model} parameters (right).

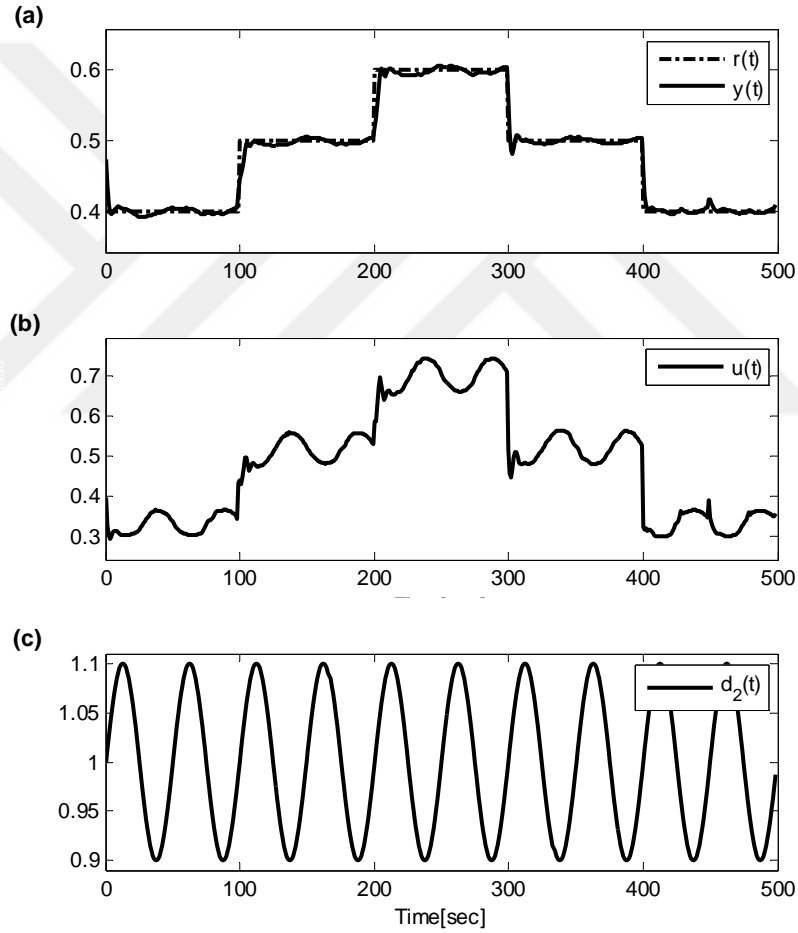


Figure 3.16 : System output (a), control signal (b), time varying system parameter (c).

In our system, $d_2(t)$ is selected as the time varying parameter and is allowed to vary slowly around its nominal value as $d_2(t) = 1 + 0.1 \sin(0.04\pi t)$. In simulations, it has been observed that the controller is not adequate to successfully manage to reject the disturbance using $\mathbf{\Pi}_c = [r_n, y_n]^T$, for this purpose, new features are considered and

input feature vector is specified as: $\Pi_c = [P_n \cdots P_{n-n_p}, I_n \cdots I_{n-n_i}, D_n \cdots D_{n-n_d}, y_n \cdots y_{n-n_y}, u_{n-1} \cdots u_{n-n_u}]^T$ where $P_n = e_n - e_{n-1}$, $I_n = e_n$, $D_n = e_n - 2e_{n-1} + e_{n-2}$ and $e_n = r_n - y_n$, and $n_p = 1, n_i = 2, n_d = 1, n_y = 0, n_u = 1$. When the control signal in Figure 3.10 is compared with Figure 3.16, it is observed that the controller successfully rejects the uncertainty in system parameter.

3.4.1.4 Closed-loop Lyapunov stability analysis

The Lyapunov stability analysis given in section 3.3.4 is performed for the proposed control methodology, and the numerical justification is illustrated in Figure 3.17 where $\frac{\partial V(t)}{\partial t}$ and $V(t)$ are depicted for the continuously stirred tank reactor system. Since it is observed that $V(t) \geq 0$ and $\frac{\partial V(t)}{\partial t} \leq 0$ during the course of control, we can conclude that the closed-loop system is stable for all cases.

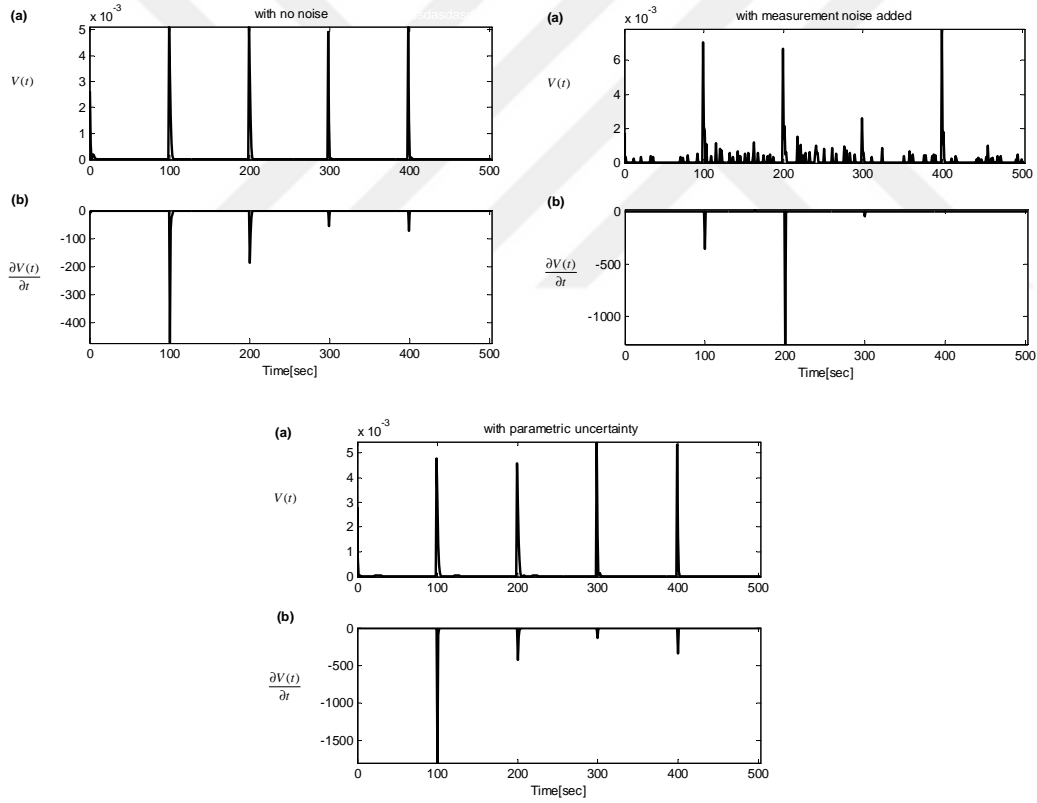


Figure 3.17 : $V(t)$ (a) and $\frac{\partial V(t)}{\partial t}$ (b) for noiseless (left), noisy (right) and with parametric uncertainty cases (middle).

3.4.2 Simulation results for bioreactor system

The bioreactor system has frequently been used as a benchmark system in nonlinear control theory to test the effectiveness of developed control methodologies [5, 43, 60,

62]. A bioreactor is a tank containing water and cells (e.g., yeast or bacteria) which consume nutrients (substrate) and produce product (both desired and undesired) and more cells [60]. This system is difficult to control since the system dynamics are highly nonlinear and exhibit limit cycles [60]. The differential equations describing the system are as follows [5, 43, 60, 62]:

$$\begin{aligned}\dot{c}_1(t) &= -c_1(t)u(t) + c_1(t)(1 - c_2(t))e^{\frac{c_2(t)}{\gamma(t)}} \\ \dot{c}_2(t) &= -c_2(t)u(t) + c_1(t)(1 - c_2(t))e^{\frac{c_2(t)}{\gamma(t)}} \frac{1 + \beta(t)}{1 + \beta(t) - c_2(t)}\end{aligned}\quad (3.66)$$

where $\gamma(t)$ is nutrient inhibition parameter and $\beta(t)$ is grow rate parameter. In the simulations, magnitude of the control signal is allowed to vary between $u_{min} = 0$ and $u_{max} = 2$; and its duration is kept constant at $\tau_{min} = \tau_{max} = 0.5s$. In this study, the proposed control architecture is tested by assuming that the dynamics are not known. Online SVR_{model} has been utilized to identify the unknown dynamics using the input-output data pairs. Since the control performance depends on the chosen features, various input feature vectors for controller can be employed according to the particulars of the system, whether there is noise, disturbance, parameter uncertainty or not, etc.

3.4.2.1 Noiseless condition

The proposed control architecture is used to control the bioreactor briefly described above, the tracking performance of the controller for noiseless condition is given in Figure 3.18. The input feature vector for controller is designated as: $\mathbf{\Pi}_c = [P_n \cdots P_{n-n_p}, I_n \cdots I_{n-n_i}, D_n \cdots D_{n-n_d}, y_n \cdots y_{n-n_y}, u_{n-1} \cdots u_{n-n_u}]^T$ where $P_n = e_n - e_{n-1}$, $I_n = e_n$, $D_n = e_n - 2e_{n-1} + e_{n-2}$ and $e_n = r_n - y_n$, and $n_p = 0, n_i = 2, n_d = 0, n_y = 0, n_u = 1$. The system tracks the reference signal accurately. The controller and system model parameters are illustrated in Figure 3.19.

3.4.2.2 Measurement noise

In order to evaluate and compare the robustness of the controller with respect to measurement noise, 30 dB measurement noise is added to the system output. Figure 3.20 shows the tracking performance and control input computed by the controller when Gaussian measurement noise is added to the system. The input feature vector for SVR_{controller} is chosen as: $\mathbf{\Pi}_c = [P_n \cdots P_{n-n_p}, I_n \cdots I_{n-n_i}, D_n \cdots D_{n-n_d}, y_n \cdots$

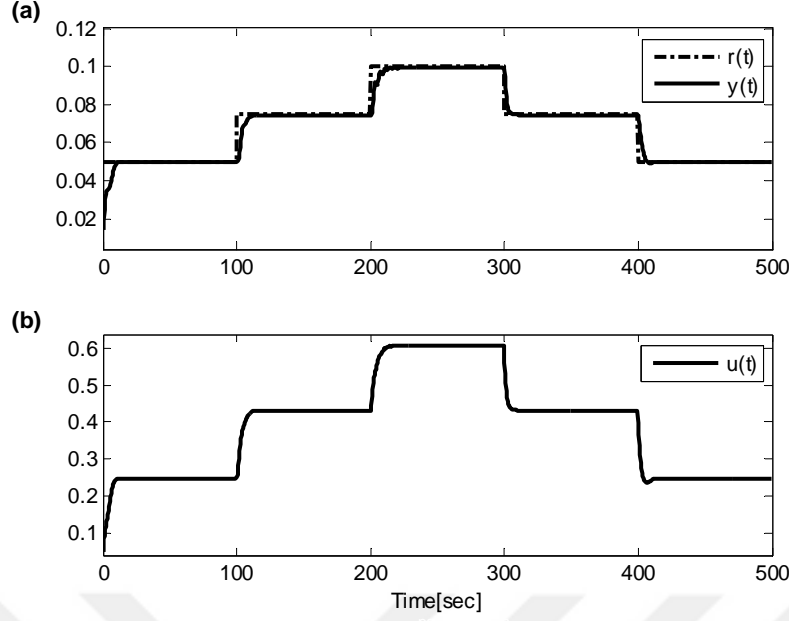


Figure 3.18 : System output (a), control signal (b) for variable step input.

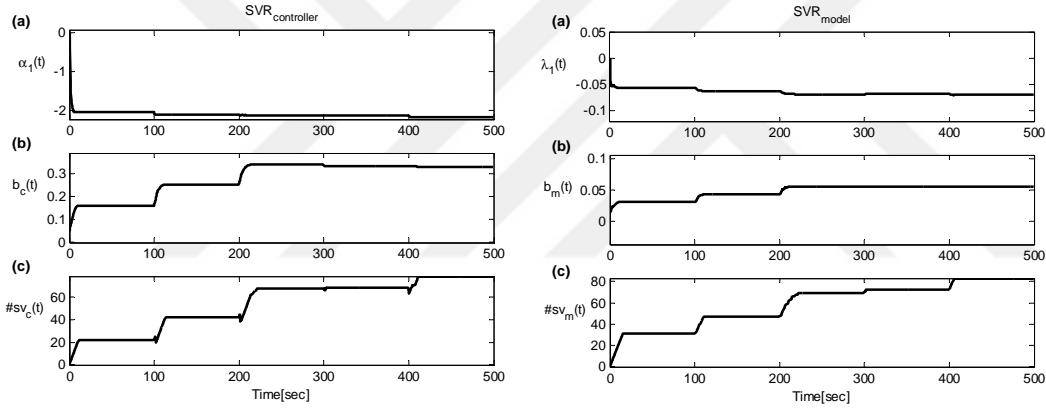


Figure 3.19 : Adaptation of $SVR_{\text{controller}}$ parameters (left), SVR_{model} parameters (right).

$y_{n-n_y}, u_{n-1} \cdots u_{n-n_u}]^T$ where $P_n = e_n - e_{n-1}$, $I_n = e_n$, $D_n = e_n - 2e_{n-1} + e_{n-2}$ and $e_n = r_n - y_n$, and $n_p = n_i = n_d = n_y = n_u = 1$. $SVR_{\text{controller}}$ and SVR_{model} parameters are depicted in Figure 3.21.

3.4.2.3 Uncertainty in system parameters

In order to examine the robustness of the proposed method with respect to parameter uncertainty, $\gamma(t)$ is considered as the time-varying parameter of the system, where its nominal value is $\gamma_{\text{nom}}(t) = 0.48$ and it is allowed to vary slowly in the purlieu of its nominal value as $\gamma(t) = 0.48 + 0.06\sin(0.008\pi t)$. The input feature vector for $SVR_{\text{controller}}$ is settled as: $\Pi_c = [r_n \cdots r_{n-n_r}, P_n \cdots P_{n-n_p}, I_n \cdots I_{n-n_i}, y_n \cdots y_{n-n_y}, u_{n-1} \cdots u_{n-n_u}]^T$ where $P_n = e_n - e_{n-1}$,

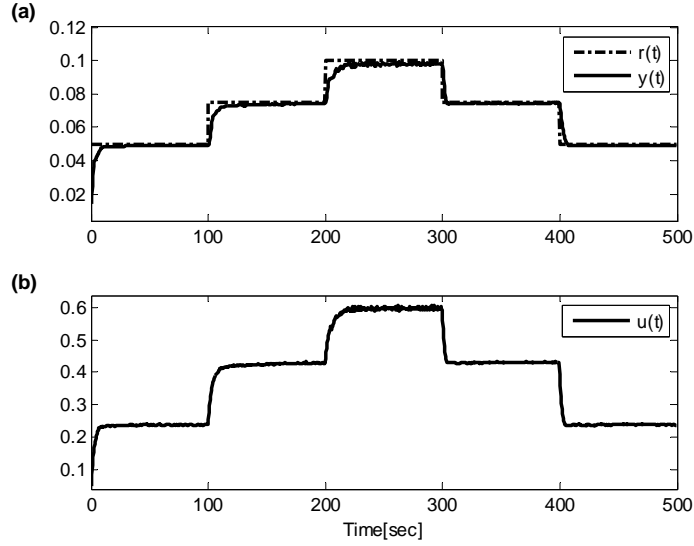


Figure 3.20 : System output (a), control signal (b) for variable step input.

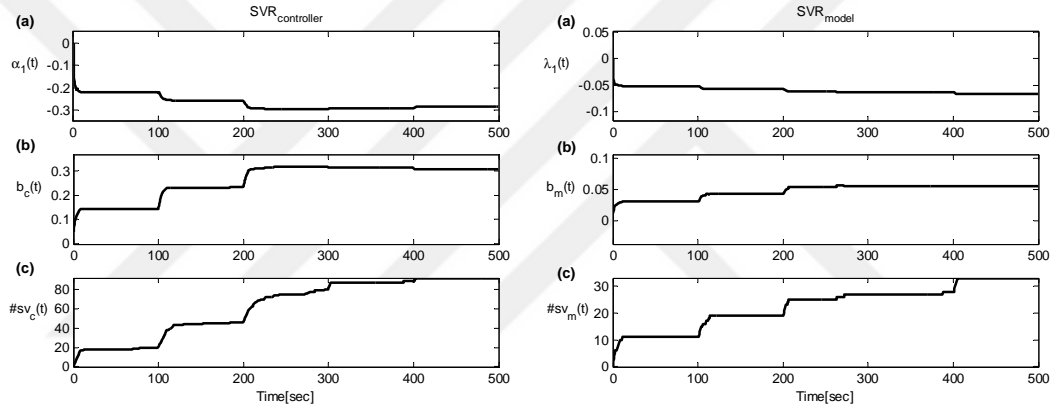


Figure 3.21 : Adaptation of $SVR_{\text{controller}}$ parameters (left), SVR_{model} parameters (right).

$I_n = e_n$, $D_n = e_n - 2e_{n-1} + e_{n-2}$ and $e_n = r_n - y_n$, and $n_r = 0, n_p = n_i = n_y = 1, n_u = 2$. Figure 3.22 illustrates the tracking performance of $SVR_{\text{controller}}$ and control signal applied to the system. When the control signal produced for nominal system parameters in Figure 3.18 and for the time varying parameter situation in Figure 3.22 are compared, it can be observed how the control signal in Figure 3.22 tries to tolerate the uncertainty of the time varying system parameter. Parameters of $SVR_{\text{controller}}$ and SVR_{model} are depicted in Figure 3.23.

3.4.2.4 Closed-loop Lyapunov stability analysis

The Lyapunov stability analysis given in section 3.3.4 is performed for the proposed control methodology, and the numerical justification is illustrated in Figure 3.24 where $\frac{\partial V(t)}{\partial t}$ and $V(t)$ are depicted for the bioreactor system. Since it is observed that $V(t) \geq 0$

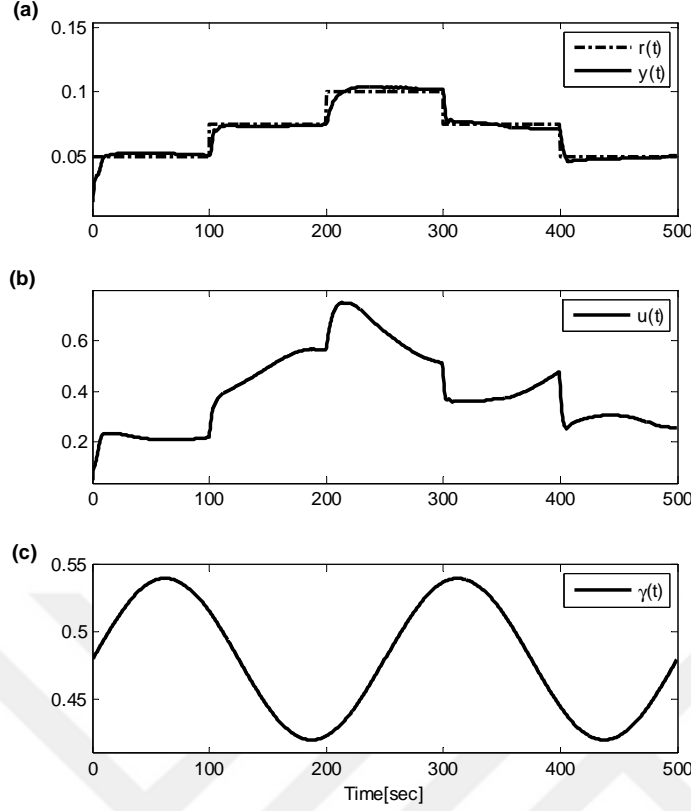


Figure 3.22 : System output (a), control signal (b), time varying parameter (c).

and $\frac{\partial V(t)}{\partial t} \leq 0$ during the course of control, we can conclude that the closed-loop system is stable for all cases.

3.4.3 Comparison of the results with adaptive PID based on SVR

The performance of the proposed controller employing $SVR_{\text{controller}}$ for tracking control and SVR_{model} for modeling is compared with the SVM-based PID controller implemented by Iplikci [5] for cases with no noise, with measurement noise added to the system and with parametric uncertainty. The method described in [5] adjusts adaptive PID controller parameters depending on the approximated K-step ahead future system behaviour. The tuning mechanism of the controller has five components: classical PID controller, NARX model of the system, line search block, control signal correction block and controller parameter tuner. SVR model of the system is employed to approximate the K-step ahead future Jacobian of the system. The parameters of the PID are tuned via Levenberg-Marquard algorithm. Jacobian matrix is dissociated into two blocks using chain rule. One block is utilized for control signal correction and the other is for parameter tuning. Control signal correction block which is based on Taylor

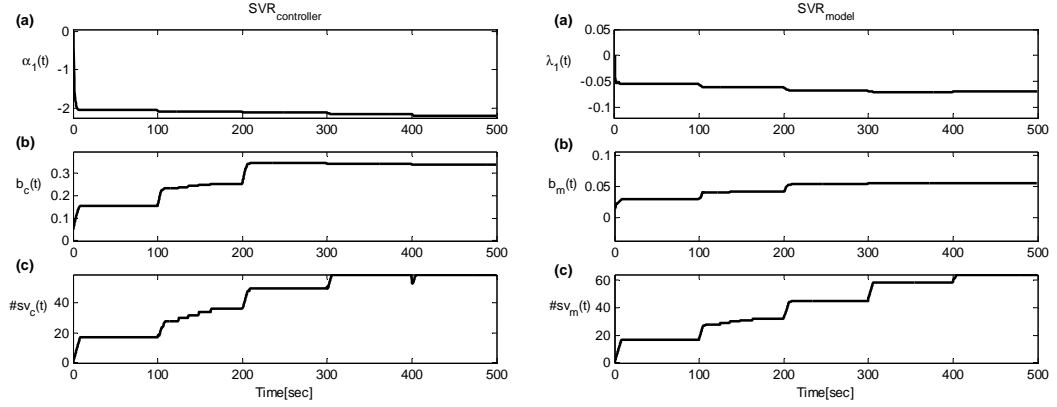


Figure 3.23 : Adaptation of $SVR_{\text{controller}}$ parameters (left), SVR_{model} parameters (right).

expansion of the control signal is employed in case the updated controller parameters are not good enough to force the system output to follow the desired trajectory. Optimal learning rate for control signal correction block is obtained in line search block via golden section method.

The simulation results obtained by our control architecture proposed in this paper are compared with the results attained by the SVM-based PID controller given in [5]. The tracking performances of the controllers for cases with no noise, with measurement noise added and with parametric uncertainty, respectively, are depicted in Figures 3.25-3.27 for CSTR and in Figures 3.28-3.30 for bioreactor system.

It can be deduced from the figures that the proposed $SVR_{\text{controller}}$ has better tracking performance than SVM-based PID controller. As the prediction horizon (K) of the PID controller is incremented, the performance as well as disturbance rejection properties improve, the best results are obtained when $K=5$. It is observed that $SVR_{\text{controller}}$ reaches the same level of performance with only $K=1$. The main reason for this is that $SVR_{\text{controller}}$ ensures global minima in all steps while SVM-based PID controller converges to local minima gradually. Comparisons of tracking performance of both controllers are illustrated in Figure 3.31 where the following performance index is utilized:

$$J_{\text{comp}} = \sum_{n=1}^{\infty} [r_{n+1} - y_{n+1}]^2 \quad (3.67)$$

Figure 3.31 illustrates that $SVR_{\text{controller}}$ has better tracking performance than SVM-based PID controller for all cases. Since the applicability of the proposed mechanism in real time is significant, computation times of each operation in the

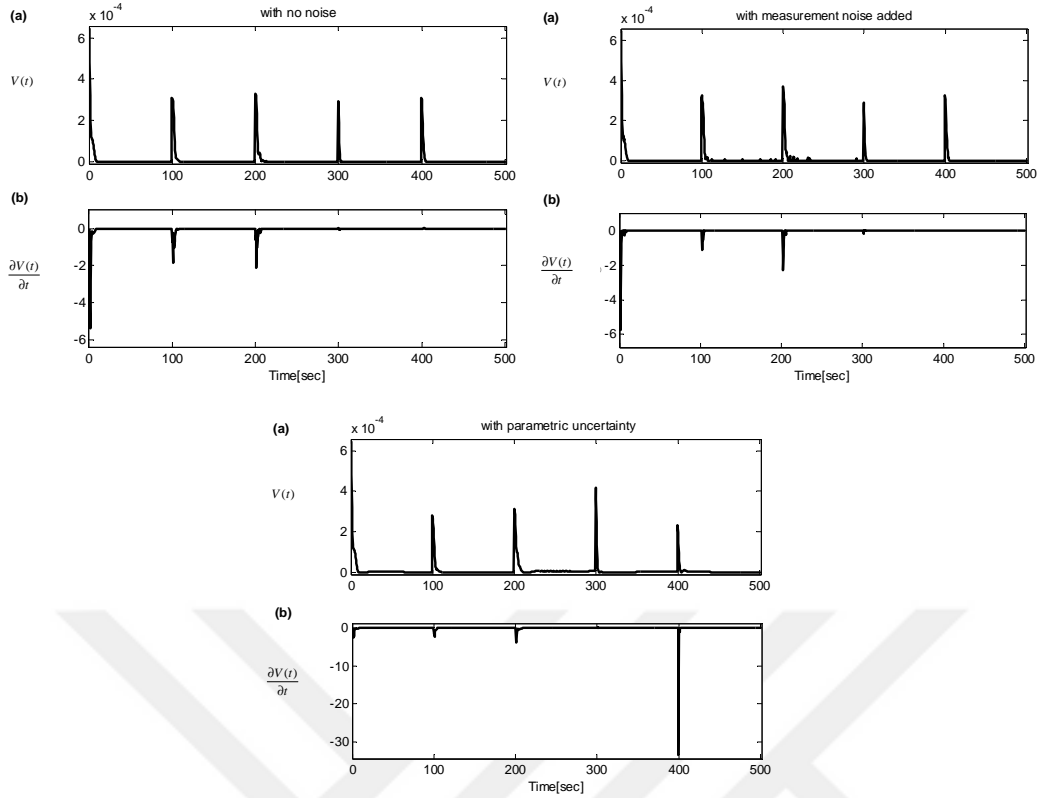


Figure 3.24 : $V(t)$ (a) and $\frac{\partial V(t)}{\partial t}$ (b) for noiseless (left), noisy (right) and with parametric uncertainty cases (middle).

control algorithm have been recorded for each case during every sampling period and the average response times of the operations have been listed in Table 3.1 for all conditions. As can be seen from the table, the average total response times of the proposed controller are less than 50 ms. Since sampling time is chosen as 100 ms in our simulations, it can be conceived that $SVR_{\text{controller}}$ can be used in real-time applications. Moreover, total response times can be minimised by utilizing effective hardwares such as FPGA or by optimizing codes in real time application. In our simulations, a PC with 2.2 GHz core i7 CPU and 8 GB RAM has been employed to implement the control algorithm and codes are not optimized.

Table 3.1 : Computation times(in ms) for $SVR_{\text{controller}}$.

Operations	CSTR			Bioreactor		
	Noiseless	Noisy	Disturbance	Noiseless	Noisy	Disturbance
$SVR_{\text{controller}}$ Training	6.65	14.819	16.728	13.731	37.341	25.471
SVR_{model} Training	16.979	9.8048	12.265	15.033	3.3563	15.369
Other	3.187	3.0002	2.966	2.881	3.0127	3.312
Total Time	26.816	27.624	31.959	31.674	43.71	44.152

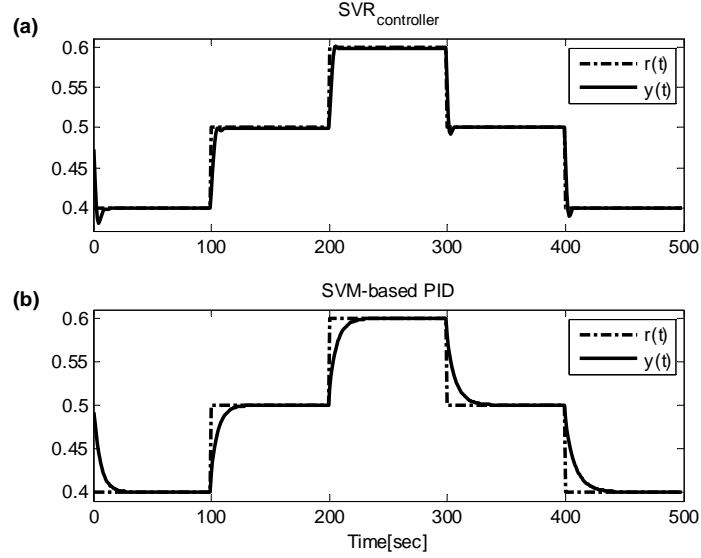


Figure 3.25 : Tracking performance of $SVR_{\text{controller}}$ (a) and SVM-based PID controller (b) with no noise.

3.5 Conclusion

In this paper, a novel control architecture is proposed where two online support vector regressors are used concurrently to minimize tracking error of a system. SVR_{model} is employed to approximate the system model and predict the output, while $SVR_{\text{controller}}$ computes the control input based on the tracking error of the closed loop system. $SVR_{\text{controller}}$ parameters are tuned without an explicit knowledge of the control signal applied to the system, hence in order to clarify the learning mechanism of the controller, the notions of "open loop" and "closed-loop" margins are introduced and explained in detail through numerous figures. The main contribution of the paper is that it justifies the use of an online SVR directly as a controller as opposed to existing works in technical literature where SVRs are generally utilized for modelling to approximate system Jacobians. The performance of the proposed controller is evaluated on CSTR and bioreactor benchmark systems. A thorough stability analysis of the closed-loop system is also presented. Additionally, the performance of the controller is compared with an SVM-based PID controller. The robustness of the controller against system parameter uncertainty and measurement noise have been examined. The results indicate that the proposed controller is quite succesful in attaining low tracking error, suppressing measurement noise and parametric uncertainties. In future works, it is planned to extend the closed-loop margin notion to develop new SVR type adaptive controller design methods.

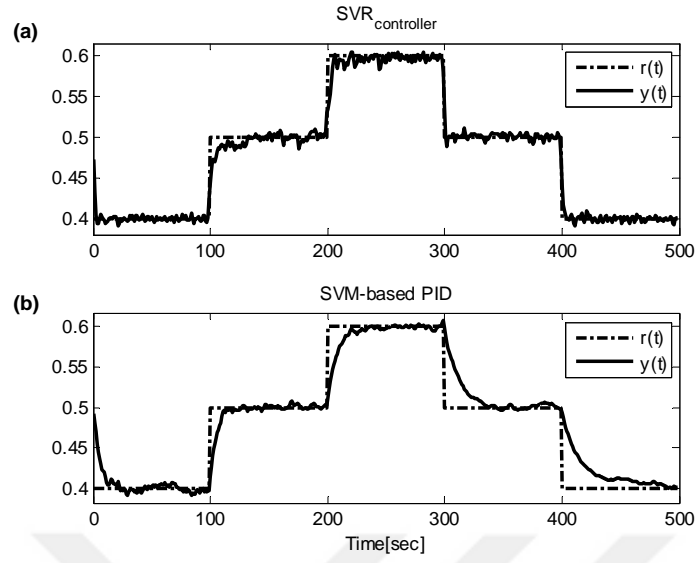


Figure 3.26 : Tracking performance of SVR_{controller} (a) and SVM-based PID controller (b) with measurement noise.

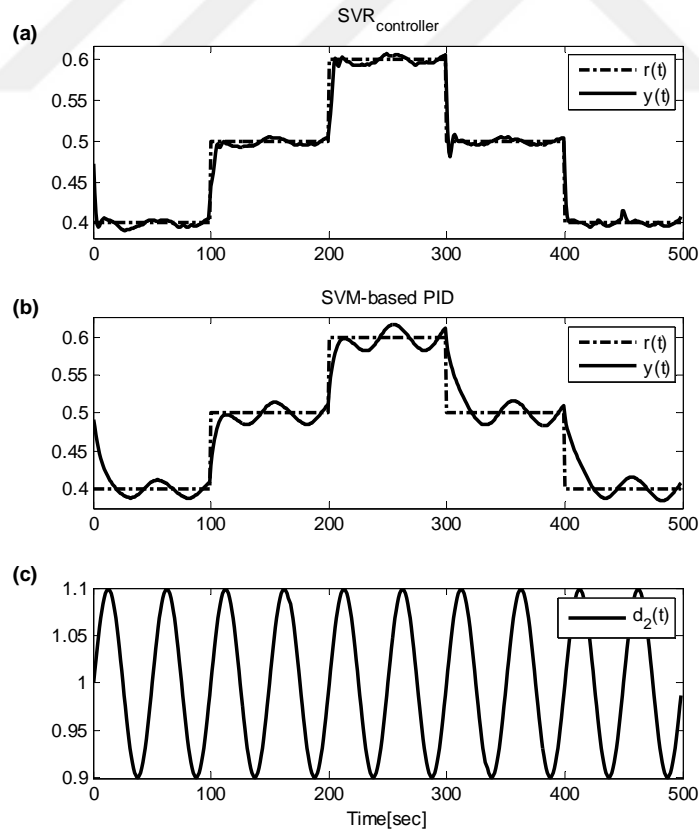


Figure 3.27 : Tracking performance of SVR_{controller} (a) and SVM-based PID controller (b) with parametric uncertainty (c).

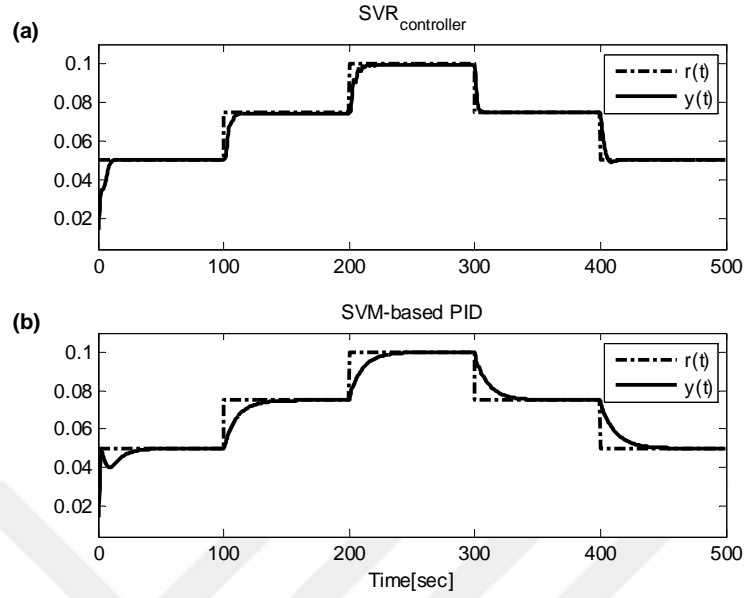


Figure 3.28 : Tracking performance of SVR_{controller} (a) and SVM-based PID controller (b) with no noise.

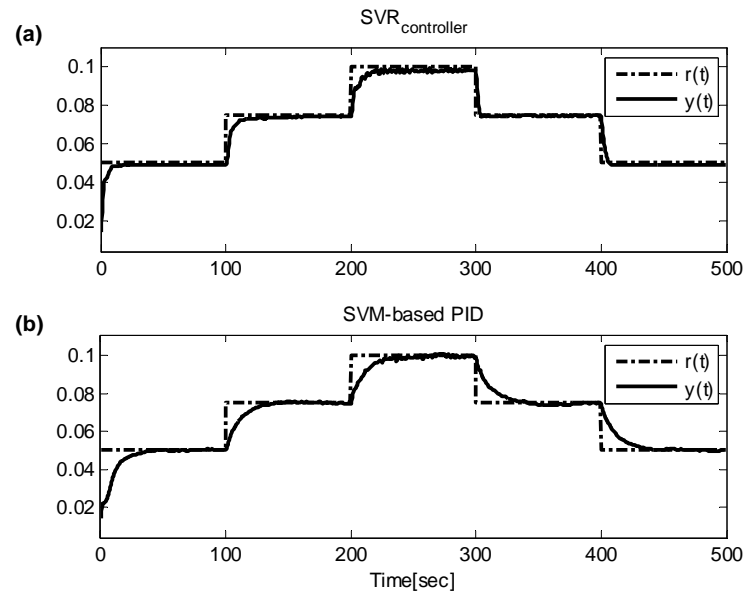


Figure 3.29 : Tracking performance of SVR_{controller} (a) and SVM-based PID controller (b) with measurement noise.

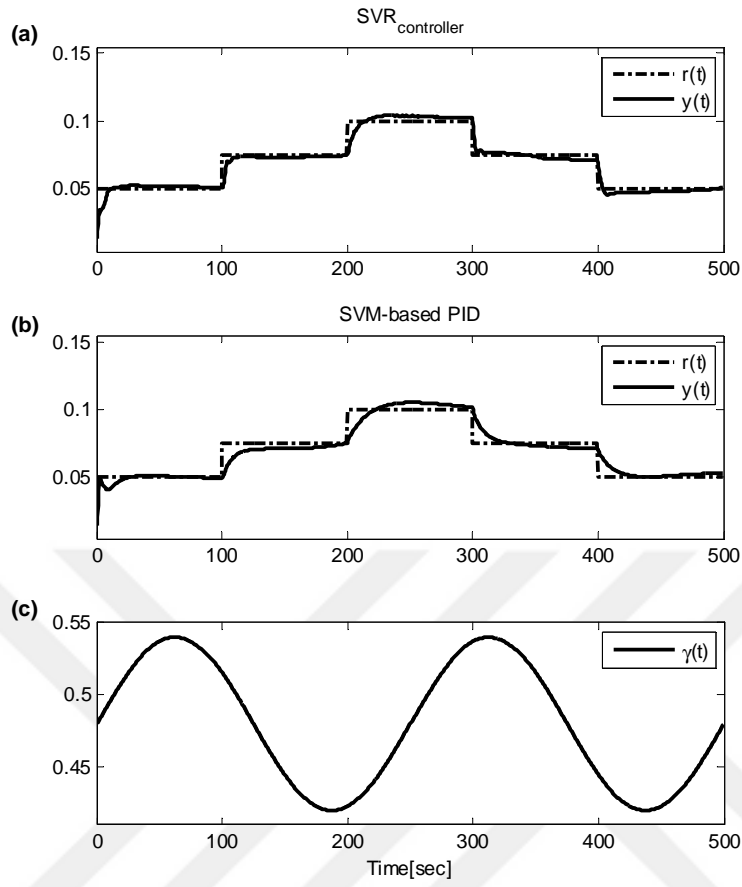


Figure 3.30 : Tracking performance of SVR_{controller} (a) and SVM-based PID controller (b) with parametric uncertainty (c).

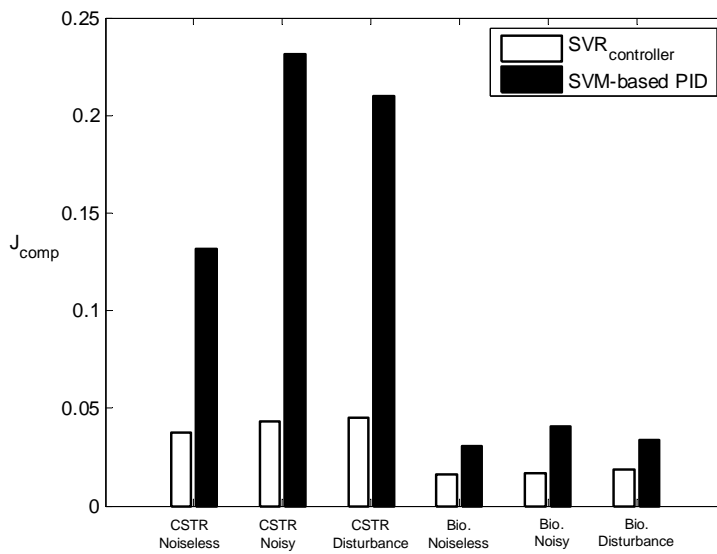


Figure 3.31 : Tracking performance comparison of the controller with respect to the defined performance index (3.67).

4. GENERALIZED SELF-TUNING REGULATOR BASED ON ONLINE SUPPORT VECTOR REGRESSION³

4.1 Introduction

Adaptation is a vital characteristic of living organisms that helps to increase their resistance to adverse environmental conditions. Every adaptation leads to some loss, in the form of material, energy or information for the organism [73]. The organism can be strengthened against adverse conditions as a result of repeated adaptation which is an accumulation of experiences that it can evaluate to minimize the losses involved in adaptation [73]. Thus, the organism can manage to learn how to alter its own characteristics against contingency.

By imitating the adaptation features of biological systems, a variety of solutions can be developed for problems in many engineering fields. Adaptation, when interpreted in terms of control theory, helps to analyse and control systems with changing parameters, model uncertainties or varying operating conditions. An adaptive control system includes a feedback structure to measure the quantity of adaptation and a mechanism, preferably incorporating some sort of intelligence, to use this quantitative information to design a controller. By introducing adaptation to a conventional controller, it is possible to use it to cope with strong nonlinearities, time delays and time-varying dynamics of systems. The necessity for adaptive controllers arises since the environment is continuously changing for many real-world systems [63].

The parameter adaptive control can be examined under three main headings in a common framework according to Astrom [74]: gain scheduling, model reference control, and self tuning regulators.

Gain scheduling can be reckoned as a mapping from previously defined scenarios between system parameters or system operating conditions to controller parameters

³This chapter is based on the paper "Uçak K. and Günel G.Ö., (2016), Generalized self-tuning regulator based on online support vector regression, Neural Computing and Applications, doi: 10.1007/s00521-016-2387-4"

[63]. In gain scheduling, the wide range of operating conditions of the system is firstly divided into small ranges via a priori information, and robust/optimal controllers are designed for each small range. Decision trees or lookup tables are employed so as to model the relationship between system conditions and controller parameters. The proper controller parameters most convenient to the situation of the system are selected.

Model reference adaptive control (MRAC) is applied to compel the closed-loop system to exhibit the same behaviour as a reference model. The reference model assigns the transient and steady-state specifications of the closed loop system. The goal of MRAC is to ensure convergence of the static and dynamic characteristics of the adjustable system, to the characteristics of the reference model [73]. The desired closed-loop behavior of the controlled system is specified by a model, error is computed as the difference between the model and closed-loop outputs, and controller parameters are obtained to minimize this error [63].

Self-tuning regulators (STR) are among the most convenient adaptive control methods for nonlinear systems. They are able to adapt controller parameters automatically [75]. This class of regulators are generally comprised of three parts: a model estimator, a controller and a block in which the controller parameters are determined from the estimated model parameters [76]. The regulator has two loops called as inner and outer loops. The inner loop is composed of the system to be controlled and an ordinary feedback controller [74]. The outer loop comprises a recursive model estimator and a design calculation to compute controller parameters [74]. For STR, controller design alternatives are very rich since it is possible to utilize various types of controllers and parameter estimators, by combining the powerful features of these components, more flexible and robust adaptive controllers can be successfully achieved. For example, by combining the nonlinear function approximation ability of artificial neural networks (ANN) and robustness of PID controllers, PID type ANN controllers can be designed to effectively control nonlinear systems [77, 78].

Model structure selection and its parametrization are significant issues for closed-loop control performance in self-tuning regulators [73]. In model based control (MBC) methodologies, controller performance is influenced by modeling inaccuracies. Although ANNs and adaptive neuro-fuzzy inference systems (ANFIS) have been

successfully employed in the identification and control of numerous nonlinear systems [19, 23, 25, 79, 80], their functionalities are impressed by local minimas resulting from non-convex objective functions. Since support vector machines (SVM) possess convex objective functions, they ensure global minimum and have better generalization capability with very few training data [1, 54, 81] compared to ANN and ANFIS. Hence SVM-based identification and control techniques have recently been utilized in adaptive control techniques instead of ANN and ANFIS [2, 5, 28, 31].

In technical literature, there are various controller structures related to STR design based on soft computing methods. Akhyar and Omatu [77] derived a self-tuning PID controller using an ANN parameter estimator to control linear and nonlinear systems. Wang et al. proposed an ANN parameter tuner to approximate the parameters of a conventional PI controller depending on various operating conditions since unmodeled system dynamics and disturbances hamper to determine suitable scheduling points in gain scheduling [78]. Ponce et al. [82] designed a self-tuning control system based on an ANN controller trained by tracking error instead of net output error to control nonlinear systems. In this approach, it is only required to know the sign of the system Jacobian to ensure the convergence of the weighting coefficients and the estimation of the sign of system Jacobian is uncomplicated compared to estimating system Jacobian [82]. Flynn et al. [83] proposed to use radial basis function neural network (RBFNN) to approximate system Jacobian for a turbogenerator system since the convergence for RBFNN is faster than multilayered feedforward neural networks (MLP), and RBF networks can be trained much more rapidly and conveniently than MLPs. Abdullah et al. [84] utilized self-tuning pole-zero placement controller for nonlinear unstable systems based on RBFNN. The nonlinear dynamics of the system is represented with a model including a simple linear time-varying sub-model derived via recursive least squares algorithm and a nonlinear sub-model identified using RBFNN. Wahyudi et al. [85] deployed an RBFNN parameter estimator trained with extended minimal resource allocation algorithm (EMRAN) which is a sequential learning technique and extended Kalman filter (EKF) to directly estimate the parameters of a PID controller. Firstly, nominal values of PID parameters are obtained with a standart controller design method, and then these parameters are tuned via RBF parameter estimator that is sequentially trained to compensate for system parameter variations. Guo and Yang [86]

adapted genetic algorithm (GA) to optimize the initial weights of an ANN parameter estimator to forecast the parameters of a PID controller for a hydro-turbine governor system. System Jacobian which is required to adjust the parameters of the ANN parameter estimator is approximated via a second ANN. Kang et al. [87] employed an ANN controller for speed control of a servo motor without using system model. In order to tune controller parameters, a linear combination of the tracking error and its derivative is deployed in place of system Jacobian. Pham and Karaboga [88] utilized a recurrent neural network (RNN) system model trained with GA for fuzzy STR to control linear and nonlinear systems. Initially, a RNN model of the system is determined by offline training with a set of input–output data pairs collected from the system. Then, the RNN model of the system is gradually improved during online control [88].

Fuzzy systems have frequently been employed to constitute adaptive mechanisms for controllers. Fuzzy logic based control methods can be categorized mainly into two classes, methods in the first category employ fuzzy estimators to tune the parameters of conventional controllers as in [89–91]. Methods in the second category use fuzzy logic controllers with tunable parameters which are updated via self tuning algorithms as in [92, 93]. He et al. [89] reduced the three parameters of the PID controller to a single unknown variable inspired by Ziegler-Nichols formula and considered a fuzzy adaptation mechanism to estimate this new single parameter. Gautam and Ha [90] proposed a fuzzy self-tuning estimator for PID parameters to control a quadrotor. They used EKF algorithm to update the parameters of the fuzzy estimator. Ahn et al. [91] approximated the parameters of a PID controller via three separate fuzzy estimators. The tunable parameters of the fuzzy estimator are adjusted by backpropagation algorithm. Evolutionary algorithms have also been used for finding the optimal parameters of fuzzy inference mechanisms [94, 95]. Bandyopadhyay et al. [94] deployed a fuzzy-genetic approach to tune the parameters of a self-tuning PID controller. The adjustable parameters of the PID controller are reduced to a single parameter using dead-beat control. The controller parameter is predicted by a fuzzy inference mechanism and the rule base of fuzzy model is optimized via genetic algorithm. Sharkawy [95] applied three independent fuzzy parameter estimator mechanisms to tune the parameters of an incremental PID controller.

Each PID parameter is tuned with a first order Takagi-Sugeno (TS) fuzzy inference system, whose parameters are optimally determined offline using a modified genetic algorithm (GA) [95]. Qiao and Mizumato [92] proposed a peak observer based tuning mechanism to adjust the scaling coefficients of a fuzzy PID controller. The mechanism updates the scaling coefficients when the system output has a peak point. Since the coefficients are not adjusted up to another peak, the proposed tuning mechanism has a limited field of use and is practical only for step type reference signals. In order to overcome drawbacks of the peak observer based tuning mechanism, Woo et al. [93] proposed to tune scaling coefficients of a fuzzy controller using a function of tracking error during the course of control. The controller parameters are successfully adapted even the system has no overshoot. Bouallègue et al. [96] utilized particle swarm optimization to adjust the parameters of a fuzzy PID controller to control an electrical DC drive. In fuzzy logic based control methodology, fuzzy rule base generally depends on the system to be controlled and the type of the controller to be implemented, so fuzzy rule base is established by intuition or practical experience [91]. In order to obtain a fuzzy system with suboptimal/optimal parameters, ANFIS based STRs combining the learning ability of ANNs with reasoning feature of fuzzy systems have been designed as in [97–99]. When backpropagation algorithm is used to train ANFIS directly as a parameter estimator, the model of the controlled system is needed. Li and Priemer [97] employed a modified random optimization method to train a neural network based fuzzy logic parameter estimator without requiring model of the system being controlled. Bishr et al. [98] developed an online training algorithm for ANFIS to estimate the parameters of a self-tuning PID controller. Lu et al. [99] used a wavelet type-2 fuzzy neural network system model for self-tuning predictive PID controllers to control liquid-level and heating processes. The parameters of the PIDs are updated using gradient descent method and the system Jacobians are approximated via wavelet type-2 fuzzy neural network system model.

In this paper, a generalized self-tuning regulator based on SVR methodology is proposed to control nonlinear dynamic systems. The main contribution of the paper is utilizing an online SVR to approximate the optimal parameter values of a self-tuning regulator. For this purpose, the "closed-loop margin" notion proposed in [3] has been expanded to design STRs and online SVR update equations are derived. The proposed

mechanism is used to optimize the parameters of two different type of controllers, namely PID and fuzzy PID controllers. Another contribution of the paper, unlike the existing research in literature, is using online learning method for estimating the system model. Stability of the closed-loop system has also been analyzed. The performance of the proposed generalized STRs has been examined on a nonlinear bioreactor system, and the performance of the generalized STRs has been compared with SVM-based PID controller proposed by Iplikci in [5]. The results show that the proposed generalized STR structure and online SVR model attain good modeling and control performances.

The organization of the paper is presented as follows: Section 4.2 describes the basic principles of online SVR. Construction of optimization problem so as to utilize SVR directly as an adaptive parameter estimator and the proposed STRs are explained in detail in section 4.3. Additionally, the stability analysis of the closed-loop system is performed. In section 4.4, the performance of the proposed mechanism is simulated and also, the performance of the proposed method is compared with an SVM-based PID controller. The study is briefly concluded in section 4.5.

4.2 Online support vector regression

This section briefly reviews online support vector regression. The basic principles of support vector regression and online learning method are presented in sections 4.2.1 and 4.2.2, respectively.

4.2.1 An overview of support vector regression

Consider the input-output training instances

$$\mathbf{T} = \{\mathbf{x}_i, y_i\}_{i=1}^N \quad \mathbf{x}_i \in \mathbf{X} \subseteq R^n, y_i \in R \quad (4.1)$$

where N and n indicate the number of the training samples and the dimension of the input samples, respectively. SVR model (4.2) can be deployed in order to capture the connection among input-output instances in (4.1).

$$\hat{y}_i = \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b, \quad i = 1, 2, \dots, N \quad (4.2)$$

where " \mathbf{w} " represents the weights of the SVR network in feature space (\mathbf{F}), " $\Phi(\mathbf{x}_i)$ " is the projection of the input samples to feature space, " b " typifies the bias of regressor

and $\langle \cdot, \cdot \rangle$ is inner product in \mathbf{F} [43]. The essence of the optimization problem for support vector machines (SVM) is based on finding the optimal separator or regressor. In classification, the separator that maximizes the margin between two different classes is searched. Similarly, in regression, the aim is to find the optimal regressor within a predefined margin via ε -insensitive loss function. The primal form for optimization problem is formulated using ε -insensitive loss function as:

$$\min_{\mathbf{w}, b, \xi, \xi^*} J_{Pr} = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) \quad (4.3)$$

with the following constraints

$$\begin{aligned} y_i - \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle - b &\leq \varepsilon + \xi_i \\ \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b - y_i &\leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0, \quad i = 1, 2, \dots, N \end{aligned} \quad (4.4)$$

where ε is the upper value of tolerable error, ξ 's and ξ^* 's denote the deviation from ε tube and called as slack variables [1, 43]. The primal form has non-convex objective function and the solution may get stuck at local minima. The dual form for the optimization problem can be obtained using Lagrangian multiplier method. Thus, the problem in (4.3-4.4) can be rendered to convex problem. For this purpose, a Lagrange function is derived by introducing non-negative Lagrange multipliers β, β^*, η and η^* as dual variables to compound objective function in (4.3) and constraints in (4.4) as follows :

$$\begin{aligned} L_{Pr} = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) - \sum_{i=1}^N \beta_i (\varepsilon + \xi_i - y_i + \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b) \\ & - \sum_{i=1}^N \beta_i^* (\varepsilon + \xi_i^* + y_i - \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle - b) - \sum_{i=1}^N (\eta_i \xi_i + \eta_i^* \xi_i^*) \end{aligned} \quad (4.5)$$

The optimality conditions for Lagrangian in (4.5) are acquired as:

$$\frac{\partial L_{Pr}}{\partial \mathbf{w}} = 0 \longrightarrow \mathbf{w} - \sum_{i=1}^N \beta_i \Phi(\mathbf{x}_i) = 0 \quad (4.6)$$

$$\frac{\partial L_{Pr}}{\partial b} = 0 \longrightarrow \sum_{i=1}^N (\beta_i - \beta_i^*) = 0 \quad (4.7)$$

$$\frac{\partial L_{Pr}}{\partial \xi_i} = 0 \longrightarrow C - \beta_i - \eta_i = 0, \quad i = 1, 2, \dots, N \quad (4.8)$$

$$\frac{\partial L_{Pr}}{\partial \xi_i^*} = 0 \longrightarrow C - \beta_i^* - \eta_i^* = 0, \quad i = 1, 2, \dots, N \quad (4.9)$$

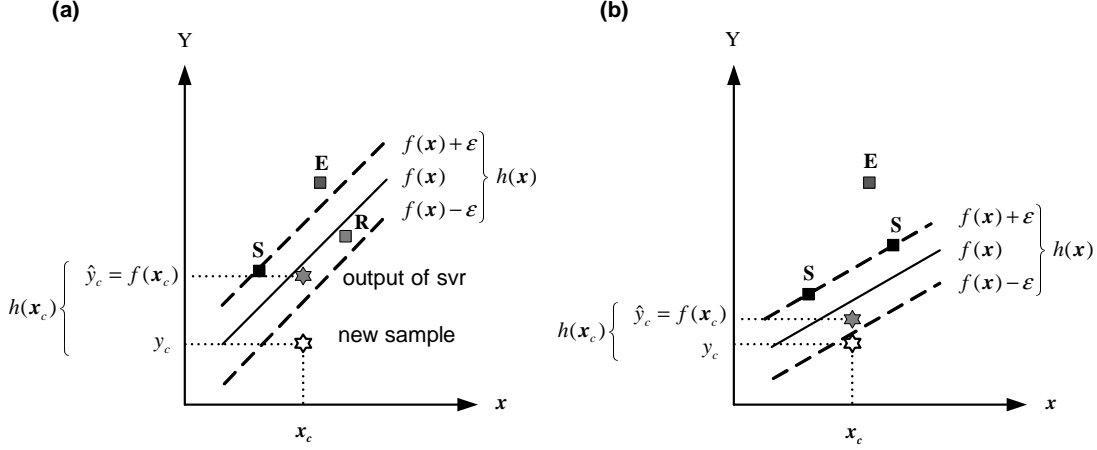


Figure 4.1 : E, S and R subsets before (a) and after (b) training [1–4].

Thus, substituting (4.6-4.9) in (4.5), dual form of the optimization problem is acquired in (4.10)-(4.11) as a quadratic programming (QP) problem:

$$J_D = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\beta_i - \beta_i^*)(\beta_j - \beta_j^*) K_{ij} + \varepsilon \sum_{i=1}^N (\beta_i + \beta_i^*) - y_i \sum_{i=1}^N (\beta_i - \beta_i^*) \quad (4.10)$$

with the following constraints

$$\begin{aligned} 0 \leq \beta_i \leq C, \quad 0 \leq \beta_i^* \leq C \\ \sum_{i=1}^N (\beta_i - \beta_i^*) = 0, \quad i = 1, 2, \dots, N \end{aligned} \quad (4.11)$$

where $K_{ij} = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$. Thus, the optimization problem is degraded to a problem with single type unknown parameter (β). Various QP algorithms can be implemented to obtain Lagrange multipliers (β). The solution of the regression problem in (4.2) can be approximated as in (4.12):

$$\hat{y}(\mathbf{x}) = \sum_{i \in SV} \lambda_i K(\mathbf{x}, \mathbf{x}_i) + b, \quad \lambda_i = \beta_i - \beta_i^* \quad (4.12)$$

where "SV" emblemizes support vectors. This method computes the solution offline, however online learning algorithms for support vector regression can also be derived via the objective function and the constraints given in (4.10,4.11).

4.2.2 Online ε -support vector regression

Notion of "margin" is the key to fully comprehend the fundamental idea of online SVR.

Let us predefine an error margin function as:

$$h(\mathbf{x}_i) = f(\mathbf{x}_i) - y_i = \sum_{j=1}^N \lambda_j K_{ij} + b - y_i \quad (4.13)$$

where $h(\mathbf{x}_i)$ is the margin value, $f(\mathbf{x}_i)$ is the output of regressor and y_i is the actual output corresponding to input \mathbf{x}_i from the data set. The training samples are separated into error support (**E**), margin support (**S**) and remaining support (**R**) subsets depending on their locations with respect to the margin function and Lagrange multipliers as follows:

$$\begin{aligned}\mathbf{E} &= \{i \mid |\lambda_i| = C, |h(\mathbf{x}_i)| \geq \varepsilon\} \\ \mathbf{S} &= \{i \mid 0 < |\lambda_i| < C, |h(\mathbf{x}_i)| = \varepsilon\} \\ \mathbf{R} &= \{i \mid |\lambda_i| = 0, |h(\mathbf{x}_i)| \leq \varepsilon\}\end{aligned}\tag{4.14}$$

When a new sample \mathbf{x}_c is imbibed by the regressor, it is required to include the new sample \mathbf{x}_c into one of these subsets (**E**,**S**,**R**) depending on the margin and Lagrange multiplier value of the new sample. During this learning process, KKT conditions must be satisfied automatically for all training instances [2]. Assuming that the Lagrange multiplier of the new added data is $\lambda_c = 0$, from (4.13), its margin value is acquired as:

$$h(\mathbf{x}_c) = f(\mathbf{x}_c) - y_c = \sum_{j=1}^N \lambda_j K(\mathbf{x}_j, \mathbf{x}_c) + b - y_c\tag{4.15}$$

The Lagrange value of current data (λ_c) and Lagrange values of previously added samples are gradually updated to provide all samples satisfy KKT conditions. As a result of the adjustment process, current data moves into one of the three sets (**E**,**S**,**R**) and the sets of the previously learned samples may change since Lagrange multiplier (λ_i) and margin values of the previously learned samples ($h(\mathbf{x}_i)$) may alter because of admittance of the new data into the regression problem. Thus, the new data is successfully ingested by the regressor. This situation is illustrated in Figure 4.1. Figure 4.1 (a), (b) depicts the margin before and after training, respectively. As seen in Figure 4.1 (a), the regressor cannot predict correctly for current data since it actually belongs to set **E** but the regressor result gives it as **R** for initial value of λ_c . For this reason, the Lagrange multiplier of the current data has to be adjusted. The incorporation of the newly added data changes the structure of the regression problem. All Lagrange multipliers are adjusted to yield low prediction error and correct classification of data. Note that Figure 4.1 shows how the sample in **R** immigrates to class **S** and newly learned sample enters into class **E**. In online learning, the optimal regression surface transumes when a new data is transcluded to training samples or a formerly trained instance is forgotten. In order to provide optimal representation of all

existing instances by the regressor, it is required to adjust the parameters of the previous model by ensuring the KKT conditions for training or forgetting phases. Derivation of online learning rules necessitates a Lagrange function which is a combination of a dual objective function and corresponding constraints. The Lagrange function for dual formulation can be expressed as follows via (4.10,4.11).

$$L_D = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\beta_i - \beta_i^*)(\beta_j - \beta_j^*)K_{ij} + \varepsilon \sum_{i=1}^N (\beta_i + \beta_i^*) - \sum_{i=1}^N y_i(\beta_i - \beta_i^*) - \sum_{i=1}^N (\delta_i \beta_i + \delta_i^* \beta_i^*) - \sum_{i=1}^N u_i(C - \beta_i) + u_i^*(C - \beta_i^*) + z \sum_{i=1}^N (\beta_i - \beta_i^*) \quad (4.16)$$

KKT optimality conditions for dual Lagrangian function in (4.16) are derived in (4.17) via dual variables:

$$\begin{aligned} \frac{\partial L_D}{\partial \beta_i} &= \sum_{j=1}^N (\beta_j - \beta_j^*)K_{ij} + \varepsilon - y_i - \delta_i + u_i + z = 0 \\ \frac{\partial L_D}{\partial \beta_i^*} &= - \sum_{j=1}^N (\beta_j - \beta_j^*)K_{ij} + \varepsilon + y_i - \delta_i^* + u_i^* - z = 0 \\ \delta_i^{(*)} &\geq 0, u_i^{(*)} \geq 0, \delta_i^{(*)} \beta_i^{(*)} = 0, u_i^{(*)} (C - \beta_i^{(*)}) = 0 \end{aligned} \quad (4.17)$$

where superscript $\delta_i^{(*)}$ represents both δ_i and δ_i^* . KKT condition indicates that at most one of β_i or β_i^* should be nonzero and both are nonnegative [4]. The margin for the i th sample \mathbf{x}_i can be designated with (4.18):

$$h(\mathbf{x}_i) = f(\mathbf{x}_i) - y_i = \sum_{j=1}^N \lambda_j K_{ij} + b - y_i \quad (4.18)$$

The convergence and migration of data in learning or forgetting phases occur according to the following conditions:

$$\begin{aligned} h(\mathbf{x}_i) &\geq \varepsilon, \lambda_i = -C \\ h(\mathbf{x}_i) &= \varepsilon, -C < \lambda_i < 0 \\ -\varepsilon &\leq h(\mathbf{x}_i) \leq \varepsilon, \lambda_i = 0 \\ h(\mathbf{x}_i) &= -\varepsilon, 0 < \lambda_i < C \\ h(\mathbf{x}_i) &\leq -\varepsilon, \lambda_i = C \end{aligned} \quad (4.19)$$

The variation on margin function values of previously learned samples ($\Delta h(\mathbf{x}_i)$) are derived as in (4.20) via (4.13-4.19) depending on Lagrange multiplier of the current sample ($\Delta \lambda_c$) and Δb [4, 55].

$$\Delta h(\mathbf{x}_i) = K_{ic} \Delta \lambda_c + \sum_{j=1}^N K_{ij} \Delta \lambda_j + \Delta b \quad (4.20)$$

The new added data has to satisfy the dual constraints in (4.11) at every parameter update step, so

$$\lambda_c + \sum_{j=1}^N \lambda_j = 0 \quad (4.21)$$

is extracted. As given in (4.14), Lagrange multiplier values of the vectors belonging to subsets **E** or **R** are equal to "0" or "C". The migration between subsets especially affects the Lagrange values of vectors in **S**. If a sample which belongs to set **S** remains in set **S** again, there is no change on margin values of mentioned sample, that is $\Delta h(\mathbf{x}_i) = 0, i \in \mathbf{S}$ [55]. Thus, relation between increments of current data ($\Delta\lambda_c$) and parameters of the previously obtained model can be formulated:

$$\sum_{j=1}^N K_{ij} \Delta\lambda_j + \Delta b = -K_{ic} \Delta\lambda_c, \quad \sum_{j \in SV} \Delta\lambda_j = -\Delta\lambda_c \quad (4.22)$$

and in matrix form is given as

$$\begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & K_{s_1 s_1} & \cdots & K_{s_1 s_k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & K_{s_k s_1} & \cdots & K_{s_k s_k} \end{bmatrix} \begin{bmatrix} \Delta b \\ \Delta\lambda_{s_1} \\ \vdots \\ \Delta\lambda_{s_k} \end{bmatrix} = - \begin{bmatrix} 1 \\ K_{s_1 c} \\ \vdots \\ K_{s_k c} \end{bmatrix} \Delta\lambda_c \quad (4.23)$$

where the indices of the samples in support vector set are defined as $\mathbf{S} = \{s_1, s_2, s_3, \dots, s_k\}$. As a consequence, $\Delta\boldsymbol{\lambda}$ is attained as:

$$\Delta\boldsymbol{\lambda} = \begin{bmatrix} \Delta b \\ \Delta\lambda_{s_1} \\ \vdots \\ \Delta\lambda_{s_k} \end{bmatrix} = \boldsymbol{\beta} \Delta\lambda_c \quad (4.24)$$

where

$$\boldsymbol{\beta} = \begin{bmatrix} \beta \\ \beta_{s_1} \\ \vdots \\ \beta_{s_k} \end{bmatrix} = -\boldsymbol{\Theta} \begin{bmatrix} 1 \\ K_{s_1 c} \\ \vdots \\ K_{s_k c} \end{bmatrix}, \quad \boldsymbol{\Theta} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & K_{s_1 s_1} & \cdots & K_{s_1 s_k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & K_{s_k s_1} & \cdots & K_{s_k s_k} \end{bmatrix}^{-1} \quad (4.25)$$

as given in [4]. Thus, the feasible increment directions for the bias and the Lagrange multipliers of the samples in **S** can be obtained for a given $\Delta\lambda_c$ using (4.23-4.25). The derivation and calculation of the Lagrange multiplier of current sample ($\Delta\lambda_c$) is detailed in [3]. The variation in margin values as a result of increment $\Delta\lambda_c$ for non-support samples can be calculated as follows using (4.18,4.20,4.24):

$$\begin{bmatrix} \Delta h(\mathbf{x}_{z_1}) \\ \Delta h(\mathbf{x}_{z_2}) \\ \vdots \\ \Delta h(\mathbf{x}_{z_m}) \end{bmatrix} = \boldsymbol{\gamma} \Delta\lambda_c, \quad \boldsymbol{\gamma} = \begin{bmatrix} K_{z_1 c} \\ K_{z_2 c} \\ \vdots \\ K_{z_m c} \end{bmatrix} + \begin{bmatrix} 1 & K_{z_1 s_1} & \cdots & K_{z_1 s_l} \\ 1 & K_{z_2 s_1} & \cdots & K_{z_2 s_l} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & K_{z_m s_1} & \cdots & K_{z_m s_l} \end{bmatrix} \boldsymbol{\beta} \quad (4.26)$$

where z_1, z_2, \dots, z_m are the indices of non-support samples, γ are margin sensitivities and $\gamma = 0$ for samples in \mathbf{S} . The alternation of the matrix Θ for learning and forgetting stages and detailed information related to recursive algorithm can be attained via [2–4, 55].

4.3 Construction of Optimization Problem for Self-Tuning Regulator

4.3.1 An overview of self-tuning regulators

The main components of a self-tuning regulator (STR) are system model, parameter estimator and controller blocks as given in Figure 4.2 where θ and \mathbf{X}_c indicate the controller parameters and input vector of the controller, respectively. In order to minimize tracking error and estimate feasible controller parameters, the future behaviour of the system is required, so system model block is essential to approximate the dynamics of the system. Parameter estimator block computes new controller parameters by regarding the future behaviour of the plant via the obtained system model, and then adjusted controller parameters are implemented in the controller block to make system track reference signal accurately. Any controller with adjustable parameters can be utilized in the generalized controller block given in Figure 4.2. In this work, the proposed STR structure is implemented with PID and fuzzy PID controllers as explained in detail below. Depending on the controlled systems and design techniques, numerous self-tuning architectures are possible in the parameter estimation block [76]. As for the system model part, various intelligent modeling techniques such as ANN [19,23,24], ANFIS [25,100] etc have been applied to identify the dynamics of system. In our controller structure, SVR is employed to model the dynamics of the controlled system since it has high generalization capacity and ensures global minimum in training. Subsequently, another SVR is used as parameter estimator to approximate controller parameters.

4.3.2 Generalized STR structure based on SVR

The tuning mechanism of the proposed STR architecture based on online SVR is depicted in Figure 4.3. There are two separate SVR structures in the proposed mechanism: $\text{SVR}_{\text{estimator}}$ to calculate the controller parameters and $\text{SVR}_{\text{model}}$

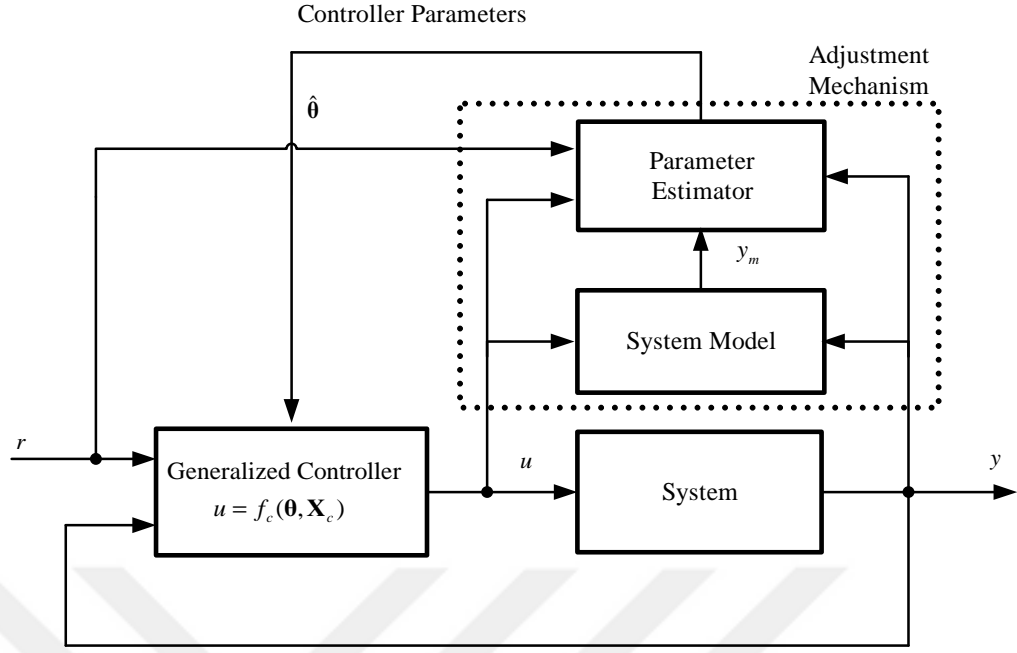


Figure 4.2 : Self-tuning regulator.

which predicts the future behaviour of the controlled system. Since SVR has multi-input-single-output (MISO) structure, a separate $SVR_{estimator}$ structure is deployed for each approximated parameter. Therefore, the number of the $SVR_{estimator}$ structures to be used in parameter estimator block depends on the number of the adjustable parameters in the controller. For instance, three $SVR_{estimator}$ structures are employed for PID type STR to forecast K_p , K_i and K_d parameters. The controller parameters are estimated via $SVR_{estimator}$ as:

$$\theta_m = f_{estimator_m}(\mathbf{\Pi}_{mc}) = \sum_{k \in SV_{estimator_m}} \alpha_{mk} K_{estimator_m}(\mathbf{\Pi}_{mc}, \mathbf{\Pi}_{mk}) + b_{estimator_m} \quad (4.27)$$

$$m \in \{1, 2, \dots, p\}$$

where $\mathbf{\Pi}_{mc}$ indicates the current input of m th estimator, $K_{estimator_m}(\cdot, \cdot)$ is the kernel matrix, α_{mk} , $\mathbf{\Pi}_{mk}$ and $b_{estimator_m}$ are the parameters of the m th parameter estimator, $f_{estimator_m}(\cdot, \cdot)$ is the regression function to be optimized in training. The controller computes a control signal as :

$$u_n = g_{controller}([\theta_1(\mathbf{\Pi}_{1c}), \dots, \theta_m(\mathbf{\Pi}_{mc})], \mathbf{X}_c) \quad (4.28)$$

where $g_{controller}$ indicates the control law computed as the output of the controller, θ_m is the m th parameter of controller and \mathbf{X}_c is the current input vector of the controller. SVR_{model} is employed to forecast system behaviour and it calculates the model output

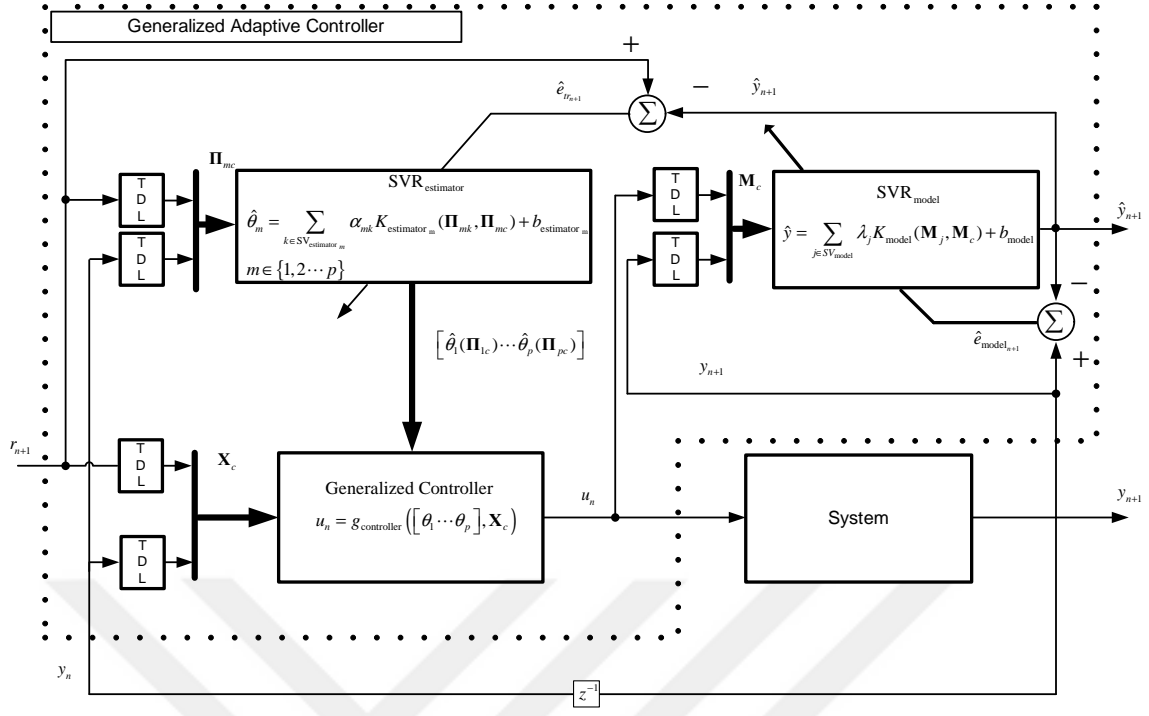


Figure 4.3 : Generalized self-tuning regulator based on online SVR.

as

$$\hat{y}_{n+1} = f_{model}(\mathbf{M}_c) = \sum_{j \in SV_{model}} \lambda_j K_{model}(\mathbf{M}_c, \mathbf{M}_j) + b_{model} \quad (4.29)$$

where f_{model} and K_{model} are the regression function and the kernel matrix of the system model respectively, \mathbf{M}_j 's are support vectors, \mathbf{M}_c is current input, and λ_j and b_{model} are the parameters of the system model to be adjusted. $SVR_{estimator}$ and SVR_{model} are both used online to perform learning, prediction and control consecutively. Ideally, during the course of online working, it is expected that \hat{y}_{n+1} will eventually converge to y_{n+1} . Therefore, when the parameters of $SVR_{estimator}$ are optimized, in order to calculate and observe the impact of the computed control signal (u_n) on system behaviour and train $SVR_{estimator}$ precisely, u_n is applied to SVR_{model} at every step of training phase of parameter estimator to predict system behaviour (y_{n+1}). The control signal applied to the real system is obtained via the trained parameter estimator and the actual output y_{n+1} is determined after applying the calculated control signal to the system. Thus, the current input of system model \mathbf{M}_c and output y_{n+1} can be computed for training phase of SVR_{model} . The detailed algorithm for the proposed adaptive control architecture is given in section 4.3.5.

4.3.3 PID type STR based on SVR

PID controller still predominates in the process industries due to its robustness, effectiveness for wide range of operating conditions and its functional simplicity [77]. It has been widely used in industry due to its simplicity, good control performance and excellent robustness to uncertainties [101]. The classical incremental PID controller produces a control signal as follows [5, 31, 73, 101–103]:

$$\begin{aligned} u_n &= u_{n-1} + \Delta u_n \\ \Delta u_n &= K_{p_n}[e_n - e_{n-1}] + K_{i_n}[e_n] + K_{d_n}[e_n - 2e_{n-1} + e_{n-2}] \end{aligned} \quad (4.30)$$

where K_{p_n} , K_{i_n} and K_{d_n} respectively indicate the parameters of proportional, integral and derivative parts of the controller to be tuned. For the proposed mechanism given in section 4.3.2, the incremental PID control law can be extracted as:

$$\begin{aligned} u_n &= g_{controller}(\boldsymbol{\theta}, \mathbf{X}_c) = u_{n-1} + \boldsymbol{\theta}^T \mathbf{X}_c \\ &= u_{n-1} + [K_{p_n} \ K_{i_n} \ K_{d_n}] \begin{bmatrix} e_n - e_{n-1} \\ e_n \\ e_n - 2e_{n-1} + e_{n-2} \end{bmatrix} \end{aligned} \quad (4.31)$$

In an adaptive control scheme, the initially assigned values of the controller parameters will generally not be optimal [5], hence, it is required to adjust the parameters via optimization methods [5, 56]. The controller parameters K_p , K_i and K_d are calculated via online SVR parameter estimator ($\text{SVR}_{\text{estimator}}$). For this purpose, an online SVR has been utilized for each controller parameter since SVR has multi-input-single-output structure, so parameter estimator is composed of three separate SVR identifiers. The PID controller parameters are estimated via $\text{SVR}_{\text{estimator}}$ as:

$$\begin{aligned} \boldsymbol{\theta} &= \begin{bmatrix} \hat{K}_{p_n} \\ \hat{K}_{i_n} \\ \hat{K}_{d_n} \end{bmatrix} = \begin{bmatrix} f_{\text{estimator}_p}(\boldsymbol{\Pi}_{pc}) \\ f_{\text{estimator}_i}(\boldsymbol{\Pi}_{ic}) \\ f_{\text{estimator}_d}(\boldsymbol{\Pi}_{dc}) \end{bmatrix} \\ &= \begin{bmatrix} \sum_{k \in \text{SV}_{\text{estimator}_p}} \alpha_{pk} K_{\text{estimator}_p}(\boldsymbol{\Pi}_{pc}, \boldsymbol{\Pi}_{pk}) + b_{\text{estimator}_p} \\ \sum_{k \in \text{SV}_{\text{estimator}_i}} \alpha_{ik} K_{\text{estimator}_i}(\boldsymbol{\Pi}_{ic}, \boldsymbol{\Pi}_{ik}) + b_{\text{estimator}_i} \\ \sum_{k \in \text{SV}_{\text{estimator}_d}} \alpha_{dk} K_{\text{estimator}_d}(\boldsymbol{\Pi}_{dc}, \boldsymbol{\Pi}_{dk}) + b_{\text{estimator}_d} \end{bmatrix} \end{aligned} \quad (4.32)$$

Since the parameters of the controller are estimated via SVR, it is named as "PID type STR based on SVR". This structure inherits both the robustness of PID controllers and the nonlinear generalization performance of SVR method.

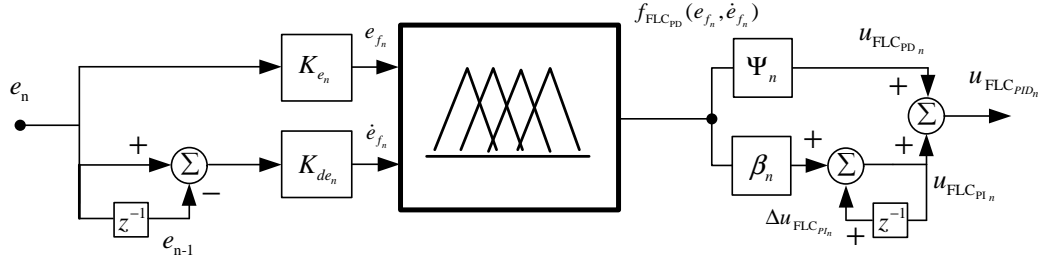


Figure 4.4 : Fuzzy PID controller.

4.3.4 Fuzzy PID type STR based on SVR

In Figure 4.4, the structure of an incremental fuzzy PID controller [96] is depicted, where e_n denotes the difference between reference signal and system output and $u_{FLC_{PIDn}}$ is the output of fuzzy PID controller at time index n . A fuzzy PID controller is also implemented as the generalized controller in Figure 4.3 and its parameters are tuned using an online $SVR_{\text{estimator}}$. Inputs of the fuzzy controller are:

$$\begin{aligned} e_{f_n} &= K_{e_n} e_n \\ \dot{e}_{f_n} &= K_{de_n} [e_n - e_{n-1}] \end{aligned} \quad (4.33)$$

The output of the controller in Figure 4.4 is computed as [92, 93]:

$$\begin{aligned} u_{FLC_{PDn}} &= \Psi_n f_{FLC_{PD}}(e_{f_n}, \dot{e}_{f_n}) \\ \Delta u_{FLC_{PI_n}} &= \beta_n f_{FLC_{PD}}(e_{f_n}, \dot{e}_{f_n}) \\ u_{FLC_{PI_n}} &= u_{FLC_{PI_{n-1}}} + \Delta u_{FLC_{PI_n}} \\ u_{FLC_{PIDn}} &= f_{FLC_{PID}}(e_{f_n}, \dot{e}_{f_n}, \Psi_n, \beta_n) = u_{FLC_{PI_n}} + u_{FLC_{PDn}} \end{aligned} \quad (4.34)$$

where the scaling factors K_{e_n} , K_{de_n} , Ψ_n and β_n are the controller parameters to be optimized. Ψ_n and β_n are the parameters for the PD and PI parts of the fuzzy PID controller, $f_{FLC_{PD}}$ is the fuzzy controller, e_{f_n} and \dot{e}_{f_n} are scaled tracking error and derivative of tracking error, respectively. The derivative and integral parts of the fuzzy controller can be associated depending on the requirements of the controlled system via input-output scaling coefficients. In our simulations, triangular membership functions with cores $\{-1, -0.4, 0, 0.4, 1\}$ as in [92] are chosen for both e_{f_n} and \dot{e}_{f_n} as shown in Figure 4.5 where Γ_{z_n} denotes the z th fired fuzzy rule. The formulation of fired fuzzy rule is given as:

$$\Gamma_{z_n} = f_{rules}(e_{f_n}, \dot{e}_{f_n}, k_{z1}, k_{z2}) = S_z + P_z - 1, \quad z \in \{1, 2, 3, 4\} \quad (4.35)$$

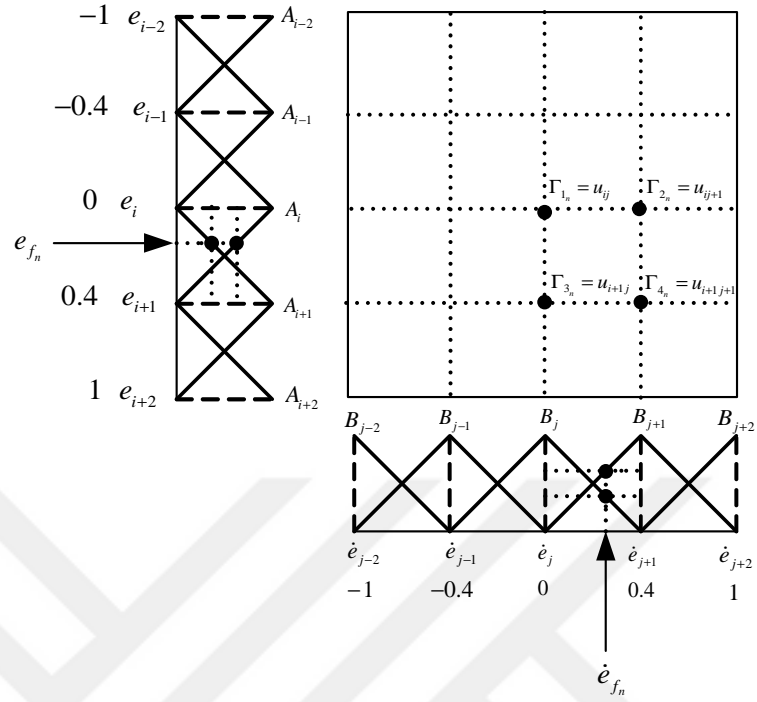


Figure 4.5 : The membership functions for inputs and rule base

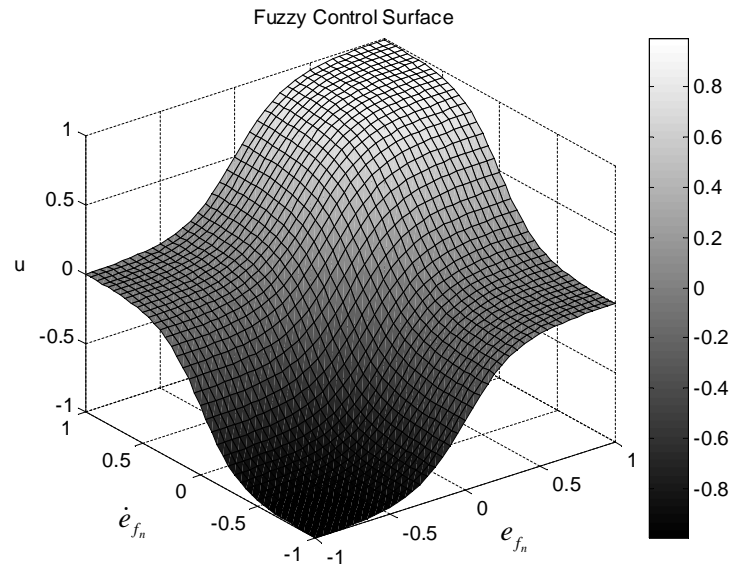


Figure 4.6 : Fuzzy control surface.

where

$$S_z = \frac{1}{1 + e^{-k_{z1}e_{fn}}} , P_z = \frac{1}{1 + e^{-k_{z2}\dot{e}_{fn}}}$$

and $k_{11} = k_{12} = 4, k_{21} = k_{22} = 5, k_{31} = k_{32} = 6, k_{41} = k_{42} = 7$. The fuzzy rule surface depicted in figure 4.6 is composed of two sigmoid functions as in (4.35). Center of gravity method has been utilized for defuzzification. Thus, the output of the fuzzy controller can be computed as [92]

$$f_{FLCPD}(e_{fn}, \dot{e}_{fn}) = \frac{\sum_{z=1}^4 w_{zn} \Gamma_{zn}}{\sum_{z=1}^4 w_{zn}} = \sum_{z=1}^4 w_{zn} \Gamma_{zn} , \sum_{z=1}^4 w_{zn} = 1 \quad (4.36)$$

where

$$\begin{aligned} w_{1n} &= A_i(e_{fn})B_j(\dot{e}_{fn}) \\ w_{2n} &= A_{i+1}(e_{fn})B_j(\dot{e}_{fn}) \\ w_{3n} &= A_i(e_{fn})B_{j+1}(\dot{e}_{fn}) \\ w_{4n} &= A_{i+1}(e_{fn})B_{j+1}(\dot{e}_{fn}) \end{aligned}$$

and

$$\begin{aligned} A_i(e_{fn}) &= \frac{e_{i+1} - e_{fn}}{e_{i+1} - e_i} , \quad A_{i+1}(e_{fn}) = \frac{e_{fn} - e_i}{e_{i+1} - e_i} \\ B_j(\dot{e}_{fn}) &= \frac{\dot{e}_{j+1} - \dot{e}_{fn}}{\dot{e}_{j+1} - \dot{e}_j} , \quad B_{j+1}(\dot{e}_{fn}) = \frac{\dot{e}_{fn} - \dot{e}_j}{\dot{e}_{j+1} - \dot{e}_j} \end{aligned}$$

are membership functions values. $\mathbf{\Gamma} = [\Gamma_{1n} \ \Gamma_{2n} \ \Gamma_{3n} \ \Gamma_{4n}] = [u_{ij} \ u_{i+1j} \ u_{ij+1} \ u_{i+1j+1}]$ indicate the fired rules in rule surface since four fuzzy rule are fired at a time depending on the defined input membership functions. The parameter estimator block tunes the controller vector $\boldsymbol{\theta}$ that consists of the K_{en}, K_{den}, Ψ_n and β_n . The controller parameters are estimated as:

$$\begin{aligned} \boldsymbol{\theta} &= \begin{bmatrix} \hat{K}_{en} \\ \hat{K}_{den} \\ \hat{\Psi}_n \\ \hat{\beta}_n \end{bmatrix} = \begin{bmatrix} f_{estimator_e}(\mathbf{\Pi}_{ec}) \\ f_{estimator_{de}}(\mathbf{\Pi}_{dec}) \\ f_{estimator_{\Psi}}(\mathbf{\Pi}_{\Psi_c}) \\ f_{estimator_{\beta}}(\mathbf{\Pi}_{\beta_c}) \end{bmatrix} \\ &= \begin{bmatrix} \sum_{k \in SV_{estimator_e}} \alpha_{ek} K_{estimator_e}(\mathbf{\Pi}_{ec}, \mathbf{\Pi}_{ek}) + b_{estimator_e} \\ \sum_{k \in SV_{estimator_{de}}} \alpha_{dek} K_{estimator_{de}}(\mathbf{\Pi}_{dec}, \mathbf{\Pi}_{dek}) + b_{estimator_{de}} \\ \sum_{k \in SV_{estimator_{\Psi}}} \alpha_{\Psi k} K_{estimator_{\Psi}}(\mathbf{\Pi}_{\Psi_c}, \mathbf{\Pi}_{\Psi k}) + b_{estimator_{\Psi}} \\ \sum_{k \in SV_{estimator_{\beta}}} \alpha_{\beta k} K_{estimator_{\beta}}(\mathbf{\Pi}_{\beta_c}, \mathbf{\Pi}_{\beta k}) + b_{estimator_{\beta}} \end{bmatrix} \end{aligned} \quad (4.37)$$

Thus, the control signal in (4.33,4.34) is obtained as:

$$\begin{aligned}
 u_{FLCPID_n} &= g_{controller}(\boldsymbol{\theta}, \mathbf{X}_c) = f_{FLCPID}(\boldsymbol{\theta}, \mathbf{X}_c) \\
 &= u_{FLCPID_{n-1}} + (\Psi_n + \beta_n) f_{FLCPD}(K_{e_n} x_1, K_{de_n} x_2) \\
 \mathbf{X}_c &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} e_n \\ e_n - e_{n-1} \end{bmatrix}
 \end{aligned} \tag{4.38}$$

where $\boldsymbol{\theta}$ indicates the controller parameters and \mathbf{X}_c is the input of the controller. By tuning fuzzy PID parameters using online SVR_{estimator}, the strong characteristics of fuzzy control technique and SVR methodology are merged to build a powerful controller for nonlinear systems.

4.3.5 Adaptive control algorithm for the generalized STR based on SVR

Input feature vector of parameter estimator($\boldsymbol{\Pi}$) should contain convenient feature variables that can well represent the closed-loop system's operating conditions. In the proposed STR, mainly reference signal and system output are utilized as input features. However, in order to enhance STR performance, the variables that are functions of reference and system output such as tracking error, integral of tracking error, derivative of tracking error and control signal etc. have also been employed as described in section 4.4. The control procedure for STR with "p" adjustable controller parameters can be summarized as follows (in the algorithm given below u_n^- indicates the control signal predicted with controller parameters obtained at the previous step and u_n^+ stands for the control signal estimated with trained controller parameters at the current step):

Step 1: Initialization of SVR_{estimator} and SVR_{model} parameters.

-SVR_{estimator}(estimator) parameters : $\alpha_{mk} = b_{estimator_m} = 0 \quad m \in \{1, 2, \dots, p\}$

-SVR_{model} (system model) parameters : $\lambda_j = b_{model} = 0$

Step 2: Prediction step for parameter estimator (θ_m^-)

-Set time step n .

-Constitute feature vector for parameter estimator ($\boldsymbol{\Pi}_{mc}$).

Some examples for parameter estimator feature vector are given as follows:

$$\boldsymbol{\Pi}_c = [r_n \dots r_{n-n_r}, y_n \dots y_{n-n_y}]$$

$$\boldsymbol{\Pi}_c = [P_n, I_n, D_n] \text{ where } P_n = e_n - e_{n-1}, I_n = e_n, D_n = e_n - 2e_{n-1} + e_{n-2} \text{ and } e_n = r_n - y_n.$$

Combination of the reference signal, system output and controller output can also be utilized in the feature vector. $\boldsymbol{\Pi}_c = [P_n, I_n, D_n, r_n \dots r_{n-n_r}, y_n \dots y_{n-n_y}, u_{n-1} \dots u_{n-n_u}]$ where n_r , n_y and n_u represent the number of the past instances of features included in the

feature vector.

-Calculate the approximated controller parameters θ_m^- by $\text{SVR}_{\text{estimator}}$ trained at previous step $(n - 1)$ via (4.27).

Step 3: Computation of control signal (u_n^-) and prediction step for system model(\hat{y}_{n+1}^-)

-Calculate the control signal u_n^- via (4.27-4.28).

-Constitute feature vector for system model (\mathbf{M}_c).

$$\mathbf{M}_c = [u_n^- \dots u_{n-n_u}, y_n \dots y_{n-n_y}]$$

-Apply u_n^- to $\text{SVR}_{\text{model}}$ and calculate \hat{y}_{n+1}^- by (4.29).

Step 4: Training step for parameter estimator

-Calculate $\hat{e}_{tr_{n+1}} = r_{n+1} - \hat{y}_{n+1}^-$

If $|\hat{e}_{tr_{n+1}}| > \epsilon_{\text{closed-loop}}$

Train parameter estimator via $\hat{e}_{tr_{n+1}} = r_{n+1} - \hat{y}_{n+1}^-$

else

Continue with parameter estimator obtained at previous step

end

Step 5: Prediction step for trained parameter estimator (θ_m^+) and computation of control input by trained estimator (u_n^+)

-Calculate the controller parameters by trained $\text{SVR}_{\text{estimator}}$ via (4.27).

-Calculate the control signal u_n^+ produced by the controller using the parameters obtained by trained $\text{SVR}_{\text{estimator}}$ via (4.27-4.28).

Step 6: Application of the control signal produced by adapted controller

-Apply u_n^+ to system to calculate y_{n+1} .

Step 7: Prediction and training step for system model(\hat{y}_{n+1}^+)

-Apply u_n^+ to $\text{SVR}_{\text{model}}$ and calculate \hat{y}_{n+1}^+ via (4.29).

-Calculate $e_{model_{n+1}} = y_{n+1} - \hat{y}_{n+1}^+$

If $|e_{model_{n+1}}| > \epsilon_{\text{model}}$

Train system model where $e_{model_{n+1}} = y_{n+1} - \hat{y}_{n+1}^+$

else

Continue with system model parameters obtained at previous step

end

Step 8: Incrementation of time step

-Increment $n = n + 1$ and back to step 2.

4.3.6 Generalized closed-loop system margin

In the proposed generalized adaptive STR architecture, two blocks contain separate online SVRs: In the system model block $\text{SVR}_{\text{model}}$ computes an estimate of the system model and in the parameter estimator block each tunable parameter of the controller is estimated by a different $\text{SVR}_{\text{estimator}}$. The training dataset for $\text{SVR}_{\text{model}}$ consists of pairs (\mathbf{M}_c, y_{n+1}) which are available during online operation, therefore the training process is straightforward as explained in section 4.2.2. However, the training of $\text{SVR}_{\text{estimator}}$ is not clear, since the input data $(\mathbf{\Pi}_{mc})$ is known, but the desired output of the estimator, namely the controller parameters $(\boldsymbol{\theta})$ to be implemented to produce control signal are not available to the designer in advance. This situation causes a significant dilemma to train SVR structures without the explicit information of desired output data. However, a similar problem which hampers to utilize SVR directly as a controller in control block was managed to overcome in [3]. For this purpose, Uçak and Öke Günel proposed "closed-loop system margin" notion to solve this situation. In this section, since training $\text{SVR}_{\text{estimator}}$ without a desired output dataset can be formulated in a similar way to the problem solved in [3], we will configure the "closed-loop margin" notion so as to train $\text{SVR}_{\text{estimator}}$ without the need to know the approximated controller parameters. Let us start by giving two main definitions.

As defined in [3], the regression margin related to $\text{SVR}_{\text{model}}$ is named as "open-loop margin". This margin is optimized by minimizing the feedforward modeling error $e_{\text{model}_{n+1}} = y_{n+1} - \hat{y}_{n+1}$, the error between the actual system output and the output of the "learned model". High modeling precision is needed to successfully identify the system dynamics and compute the tracking error in the next step. This information is used to tune the parameters of $\text{SVR}_{\text{estimator}}$ s which consequently compute the controller parameters. Note that, we employ a separate $\text{SVR}_{\text{estimator}}$ for each tunable controller parameter. Hence, the performances of $\text{SVR}_{\text{model}}$ and $\text{SVR}_{\text{estimator}}$ s are effective in the closed-loop success. The closed-loop performance of the overall system is affected by both the modeling error and the tracking error, so we define "closed-loop margin", which is a function of the system model margin and parameter estimator margins,

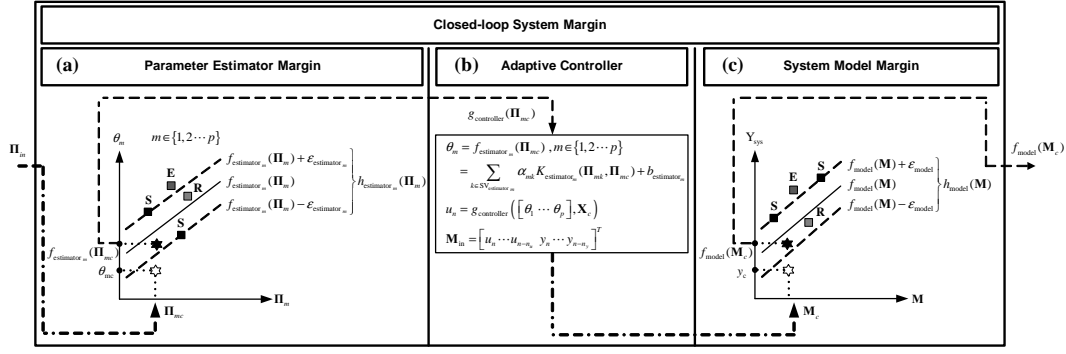


Figure 4.7 : Margins of $\text{SVR}_{estimator}$ (a), adaptive controller (b) and SVR_{model} (c).

whereas it is defined as a function of controller and system model margins in [3]. In tracking control, it is aimed to pursue reference signal as close as possible by minimizing the error $e_{tr_{n+1}} = r_{n+1} - y_{n+1}$, between reference input and closed-loop output, so the closed-loop margin should be minimized. The optimization of the SVR_{model} margin is also important, since for good closed-loop control performance we need a system model with minimum error. \mathcal{E}_{model} and $\mathcal{E}_{closed-loop}$, the upper values of tolerable error for SVR_{model} and the overall closed loop system (SVR_{model} and $\text{SVR}_{estimator}$ s combined), respectively are set independently by the designer, so the margin of SVR_{model} is optimized independently from the closed loop system margin. However, the designer cannot set $\mathcal{E}_{estimator}$, the upper value of tolerable error for $\text{SVR}_{estimator}$, and therefore does not have a direct influence on the margins of the parameter estimators, but parameter estimator margins can only be affected indirectly through the combined actions of SVR_{model} and the controlled closed loop structure. We can infer that optimization of closed-loop and SVR_{model} margins will spontaneously lead to the optimization of $\text{SVR}_{estimator}$ s margins so $\text{SVR}_{estimator}$ s parameters can be obtained indirectly while we try to minimize tracking error (or equivalently we optimize closed-loop margin). The margins of $\text{SVR}_{estimator}$ and SVR_{model} are illustrated in Figure 4.7. Parameter estimator $\text{SVR}_{estimator}$ margin is depicted in Figure 4.7 (a), adaptive controller is given in Figure 4.7 (b) and system model SVR_{model} margin is shown in Figure 4.7 (c) where $f_{estimator_m}$ and f_{model} denote the regression functions of m th parameter estimator and system model, and $g_{controller}$ indicates the control law. The input of SVR_{model} is \mathbf{M}_c and its output is y_{n+1} while the input to $\text{SVR}_{estimator}$ is Π_{mc} and the output is θ_m , so in Figure 4.7, the input-output axes of SVR_{model} are denominated as \mathbf{M} and Y_{sys} while the axes for $\text{SVR}_{estimator}$ are named

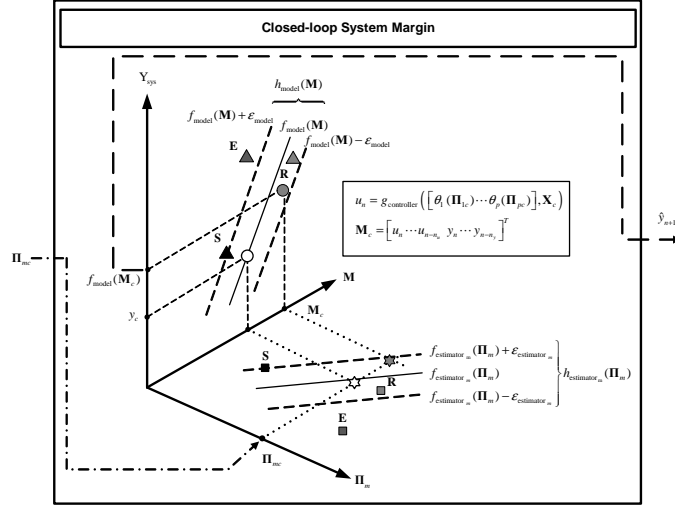


Figure 4.8 : Closed-loop system margin in three dimensions.

as Π_m and θ_m . Considering the margins of SVR_{model} and the separate $SVR_{estimator}$ s for each tunable controller parameter, we can combine the independent subgraphs to yield a multidimensional graph which depicts the closed-loop system margin. In applications, there will generally be several tunable controller parameters and input vectors will commonly be multi-dimensional, resulting in hypersurfaces when margins are fused. As a representative graph in Figure 4.8, we illustrate the case where all inputs and outputs are assumed to be one dimensional vectors and there is only a single tunable controller parameter, resulting in a hypercube. In Figure 4.8, $SVR_{estimator}$ margin is drawn on the horizontal plane with axes Π and M while SVR_{model} margin is given on the vertical plane, where axes M and Y_{sys} represent its input and output. Since vector M includes u_n and θ_m , the horizontal regression surface representing $SVR_{estimator}$ can be depicted with axes Π and M instead of Π and θ_m . During online operation of the whole control architecture, the margins of the controller parameter estimators and model are fused, and closed loop margin between closed loop input and output is intuitively thought as a single margin, as depicted in Figure 4.9. Here, controller parameter estimator and system model margins are combined resulting in the "closed-loop margin" and this is projected onto $\Pi - Y_{sys}$ axes. Figure 4.9 illustrates this projection for a single controller parameter estimator, before and after online training. Closed-loop margin is represented with, $h_{closed-loop}(\Pi)$, and it is a function of model margin ($h_{model}(M)$) and parameter estimator margin ($h_{estimator}(\Pi)$). When training the closed-loop system, we require that the closed-loop output tracks the reference input, r_{n+1} , and the error between closed-loop output and reference input $e_{tr_{n+1}} = r_{n+1} - y_{n+1}$

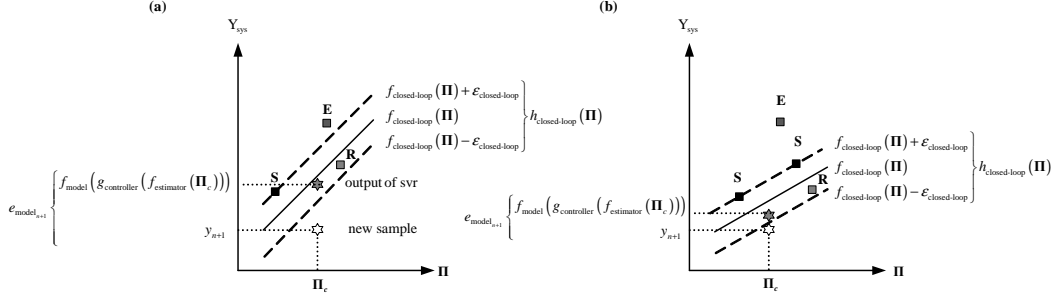


Figure 4.9 : Closed loop margin before (a) and after (b) training.

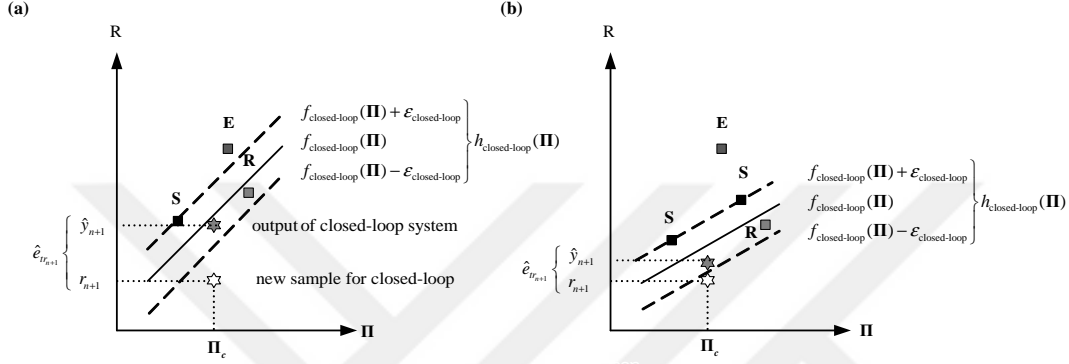


Figure 4.10 : Closed loop margin before (a) and after (b) training.

is minimized. Based on this, we have used data pairs (Π_c, r_{n+1}) in training. Hence, the input and output axes for closed-loop system regression surface are termed as Π and R in Figure 4.10 and axis R is utilized in place of Y_{sys} for closed-loop system in Figure 4.10-4.11.

4.3.7 Online support vector regression for parameter estimator

Let the training data set used for the closed-loop system be:

$$\mathbf{T} = \{\Pi_{mi}, r_{i+1}\}_{i=1}^N \quad \Pi_{mi} \in \Pi \subseteq R^n, r_{i+1} \in R \quad (4.39)$$

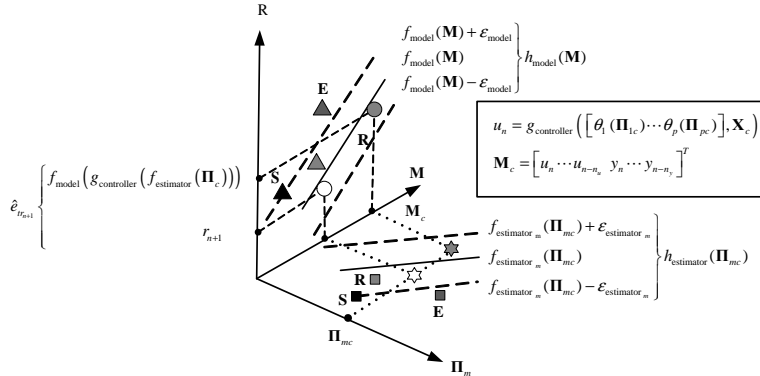


Figure 4.11 : Closed-loop system margin in three dimensions.

where N and n respectively indicate the number of the training samples and the dimension of the input samples, $\mathbf{\Pi}_{mi}$ is input feature vector of m th parameter estimator and r_{i+1} is the reference signal that system is required to track, the closed-loop margin function of the system for the i th sample $\mathbf{\Pi}_{mi}$ can be defined as

$$h_{closed-loop}([\mathbf{\Pi}_{1i} \cdots \mathbf{\Pi}_{mi}]) = \hat{y}_{i+1} - r_{i+1} = f_{model}(\mathbf{M}_i) - r_{i+1} \quad (4.40)$$

where

$$\begin{aligned} \hat{y}_{i+1} &= f_{model}(\mathbf{M}_i) = \sum_{j \in SV_{model}} \lambda_j K_{model}(\mathbf{M}_j, \mathbf{M}_i) + b_{model} \\ \lambda_j &= \beta_j - \beta_j^* \\ \mathbf{M}_i &= [u_i \cdots u_{i-n_u}, y_i \cdots y_{i-n_y}] \\ u_i &= g_{controller}([\theta_1(\mathbf{\Pi}_{1i}) \cdots \theta_m(\mathbf{\Pi}_{mi})], \mathbf{X}_c) \\ \theta_m(\mathbf{\Pi}_{mi}) &= f_{estimator_m}(\mathbf{\Pi}_{mi}) = \sum_{k \in SV_{estimator_m}} \alpha_{mk} K_{estimator_m}(\mathbf{\Pi}_{mk}, \mathbf{\Pi}_{mi}) + b_{estimator_m} \\ \alpha_{mk} &= \eta_{mk} - \eta_{mk}^*, m \in \{1, 2, \dots, p\} \\ \mathbf{\Pi}_{mi} &= [r_i \cdots r_{i-n_r}, y_i \cdots y_{i-n_y}, u_{i-1} \cdots u_{i-n_u}] \end{aligned}$$

Since parameters of SVR_{model} are fixed and known in training phase of $SVR_{estimator}$, and the sole unknown variables are the parameters of the $SVR_{estimator}$, the closed loop margin can be rewritten as

$$h_{closed-loop}([\mathbf{\Pi}_{1i} \cdots \mathbf{\Pi}_{mi}]) = \hat{y}_{i+1} - r_{i+1} = f_{closed-loop}([\mathbf{\Pi}_{1i} \cdots \mathbf{\Pi}_{mi}]) - r_{i+1} \quad (4.41)$$

with respect to an input-output data pair of closed-loop system $(\mathbf{\Pi}_{mi}, r_{i+1})$. Thus, using $(\mathbf{\Pi}_{mi}, r_{i+1})$ data pair and closed-loop margin in (4.40, 4.41), the incremental learning algorithm for $SVR_{estimator}$ can be obtained. When new sample $\mathbf{\Pi}_{mc}$ is introduced, the coefficient α_{mc} corresponding this the new sample should be changed in a finite number of discrete steps until it meets the KKT conditions while ensuring the existing samples in \mathbf{T} continue to satisfy the KKT conditions at each step [4]. The KKT conditions [3, 4, 43] that are fundamental in convergence and migration of the closed-loop data are given as :

$$\begin{aligned} h_{closed-loop}([\mathbf{\Pi}_{1i} \cdots \mathbf{\Pi}_{mi}]) &\geq \varepsilon_{closed-loop}, \alpha_i = -C_{closed-loop} \\ h_{closed-loop}([\mathbf{\Pi}_{1i} \cdots \mathbf{\Pi}_{mi}]) &= \varepsilon_{closed-loop}, -C_{closed-loop} < \alpha_i < 0 \\ -\varepsilon_{closed-loop} &\leq h_{closed-loop}([\mathbf{\Pi}_{1i} \cdots \mathbf{\Pi}_{mi}]) \leq \varepsilon_{closed-loop}, \alpha_i = 0 \\ h_{closed-loop}([\mathbf{\Pi}_{1i} \cdots \mathbf{\Pi}_{mi}]) &= -\varepsilon_{closed-loop}, 0 < \alpha_i < C_{closed-loop} \\ h_{closed-loop}([\mathbf{\Pi}_{1i} \cdots \mathbf{\Pi}_{mi}]) &\leq -\varepsilon_{closed-loop}, \alpha_i = C_{closed-loop} \end{aligned} \quad (4.42)$$

The incremental algorithm for $\text{SVR}_{\text{estimator}}$ can be derived by recasting equations (4.18-4.26) using α_m , $b_{\text{estimator}_m}$, $h_{\text{closed-loop}}$, $\varepsilon_{\text{closed-loop}}$, Π_{mi} and $K_{\text{estimator}_m}$ in place of λ , b , h , ε , \mathbf{x}_i and K . Thus, optimal parameters of $\text{SVR}_{\text{estimator}}$, α_{mk} , $b_{\text{estimator}_m}$ are sought within $\varepsilon_{\text{closed-loop}}$ tube by minimizing the tracking error via online learning algorithm given in section 4.2.2. Thus, the update direction vector for Lagrange multipliers of support set samples in m th parameter estimator $\Delta\alpha_m$ is attained as:

$$\Delta\alpha_m = \begin{bmatrix} \Delta b_{\text{estimator}_m} \\ \Delta\alpha_{s_1} \\ \vdots \\ \Delta\alpha_{s_k} \end{bmatrix} = \beta_m \Delta\alpha_{mc} \quad (4.43)$$

where

$$\beta_m = \begin{bmatrix} \beta \\ \beta_{s_1} \\ \vdots \\ \beta_{s_k} \end{bmatrix} = -\Theta_m \begin{bmatrix} 1 \\ K_{\text{estimator}_{ms_1c}} \\ \vdots \\ K_{\text{estimator}_{ms_kc}} \end{bmatrix}, \quad \Theta_m = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & K_{\text{estimator}_{ms_1s_1}} & \cdots & K_{\text{estimator}_{ms_1s_k}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & K_{\text{estimator}_{ms_ks_1}} & \cdots & K_{\text{estimator}_{ms_ks_k}} \end{bmatrix}^{-1} \quad (4.44)$$

The margin values of the non-support samples for $\Delta\alpha_{mc}$ can be calculated as follows :

$$\begin{aligned} & \begin{bmatrix} \Delta h_{\text{closed-loop}}(\begin{bmatrix} \Pi_{1z_1} & \cdots & \Pi_{mz_1} \end{bmatrix}) \\ \Delta h_{\text{closed-loop}}(\begin{bmatrix} \Pi_{1z_2} & \cdots & \Pi_{mz_2} \end{bmatrix}) \\ \vdots \\ \Delta h_{\text{closed-loop}}(\begin{bmatrix} \Pi_{1z_r} & \cdots & \Pi_{mz_r} \end{bmatrix}) \end{bmatrix} = \gamma_m \Delta\lambda_c \\ \gamma_m &= \begin{bmatrix} K_{\text{estimator}_{mz_1c}} \\ K_{\text{estimator}_{mz_2c}} \\ \vdots \\ K_{\text{estimator}_{mz_rc}} \end{bmatrix} + \begin{bmatrix} 1 & K_{\text{estimator}_{mz_1s_1}} & \cdots & K_{\text{estimator}_{mz_1s_l}} \\ 1 & K_{\text{estimator}_{mz_2s_1}} & \cdots & K_{\text{estimator}_{mz_2s_l}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & K_{\text{estimator}_{mz_rs_1}} & \cdots & K_{\text{estimator}_{mz_rs_l}} \end{bmatrix} \beta_m \end{aligned} \quad (4.45)$$

where z_1, z_2, \dots, z_r are the indices of non-support samples, γ_m are margin sensitivities.

4.3.8 Stability analysis of the closed-loop system

4.3.8.1 Stability analysis for the generalized STR based on SVR

In this subsection, the Lyapunov stability analysis of the generalized STR based on SVR has been conducted. In order to clearly explain the stability analysis of the closed-loop system, firstly, the regression functions of estimator and the system model are expressed in matrix or vector form. Thus, the regression function of estimator can

be expressed as

$$\theta_m(\mathbf{\Pi}_{mc}) = f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc}) = \begin{bmatrix} b_{estimator_m} \\ \alpha_{m1} \\ \vdots \\ \alpha_{mk} \end{bmatrix}^T \begin{bmatrix} 1 \\ K_{m1}\mathbf{\Pi}_{mc} \\ \vdots \\ K_{mk}\mathbf{\Pi}_{mc} \end{bmatrix} = \mathbf{\Omega}_m^T \mathbf{K}_{estimator_m}(\mathbf{\Pi}_{mc})$$

$$m \in \{1, 2 \dots p\}$$
(4.46)

Using (4.28) and (4.46), the output of the controller is defined as a function of $\mathbf{\Omega}$, $\mathbf{\Pi}$ and \mathbf{X} as follows:

$$u_n = g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})$$
(4.47)

The regression function of the system model (4.29) which is required to approximate system Jacobian in stability analysis is obtained as

$$\hat{y}_{n+1} = f_{model}(\boldsymbol{\lambda}, \mathbf{M}_c) = \begin{bmatrix} b_{model} \\ \lambda_1 \\ \vdots \\ \lambda_k \end{bmatrix}^T \begin{bmatrix} 1 \\ K_{1\mathbf{M}_c} \\ \vdots \\ K_{k\mathbf{M}_c} \end{bmatrix} = \boldsymbol{\lambda}^T \mathbf{K}_{model}(\mathbf{M}_c)$$
(4.48)

In order to derive stability conditions, the following Lyapunov function is deployed

$$V(e_{tr_{n+1}}) = \frac{e_{tr_{n+1}}^T \mathbf{P} e_{tr_{n+1}}}{2}$$
(4.49)

where $\mathbf{P} = \mathbf{I}$ (identity matrix). Both the stability of the system and the convergence of the controller are guaranteed when $\frac{\partial V}{\partial t} \leq 0$ [68]. Thus, the derivative of Lyapunov function ($\frac{\partial V}{\partial t}$) is acquired as

$$\frac{\partial V(e_{tr_{n+1}})}{\partial t} = e_{tr_{n+1}}^T \mathbf{P} \dot{e}_{tr_{n+1}}$$
(4.50)

where $\dot{e}_{tr_{n+1}} = \frac{\partial e_{tr_{n+1}}}{\partial t} = \frac{\partial e_{tr_{n+1}}}{\partial u_n} \frac{\partial u_n}{\partial t}$. Considering a small deviation from the equilibrium point, which corresponds to local stability analysis using equation (4.51) [3, 68], the incremental change in the control signal (4.47) is obtained as

$$\Delta u_n = \sum_{m=1}^p \left[\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m} \Delta \mathbf{\Omega}_m + \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m} \Delta \mathbf{\Pi}_m \right] + \sum_{k=1}^i \left[\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k} \Delta x_k \right]$$
(4.51)

where "p" indicates the number of the controller parameters, "i" denotes the number of the inputs for the controller and x_k is the k th input of the controller. Substituting (4.51) in (4.50), the equation (4.50) can be rearranged as

$$\frac{\partial V(e_{tr_{n+1}})}{\partial t} = e_{tr_{n+1}}^T \mathbf{P} \frac{\partial e_{tr_{n+1}}}{\partial u_n} \left[\sum_{m=1}^p \left[\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m} \Delta \mathbf{\Omega}_m + \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m} \Delta \mathbf{\Pi}_m \right] + \sum_{k=1}^i \left[\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k} \Delta x_k \right] \right]$$
(4.52)

with the assumption $\Delta \mathbf{\Pi}_m \cong e_{tr_{n+1}}$ and $\Delta x_k \cong e_{tr_{n+1}}$. Thus, as can be seen in (4.51-4.52), stability depends on the increments in Lagrange multipliers of the estimator ($\Delta \mathbf{\Omega}_m$). The adjustment rule for $\Delta \mathbf{\Omega}_m$ is derived in (4.43-4.44). The Lagrange value for current data of the m th estimator is computed as

$$\Delta \alpha_{mc} = q_m \min(|L_{c_1}^m|, |L_{c_2}^m|, |\mathbf{L}^{\mathbf{S}_m}|, |\mathbf{L}^{\mathbf{E}_m}|, |\mathbf{L}^{\mathbf{R}_m}|) \quad (4.53)$$

where $q_m = q = \text{sign}(-h_{\text{closed-loop}}([\mathbf{\Pi}_{1i} \cdots \mathbf{\Pi}_{mi}]) = \text{sign}(e_{tr_{n+1}})$ and $L_{c_1}^m, L_{c_2}^m$ are variations of the current sample and $\mathbf{L}^{\mathbf{S}_m} = [L_i^{\mathbf{S}_m}, i \in \mathbf{S}_m]$, $\mathbf{L}^{\mathbf{E}_m} = [L_i^{\mathbf{E}_m}, i \in \mathbf{E}_m]$, $\mathbf{L}^{\mathbf{R}_m} = [L_i^{\mathbf{R}_m}, i \in \mathbf{R}_m]$ are the variations of the $\mathbf{\Pi}_{mi}$ data in sets $\mathbf{S}_m, \mathbf{E}_m, \mathbf{R}_m$ respectively. The term $\min(|L_{c_1}^m|, |L_{c_2}^m|, |\mathbf{L}^{\mathbf{S}_m}|, |\mathbf{L}^{\mathbf{E}_m}|, |\mathbf{L}^{\mathbf{R}_m}|)$ in (4.53) is a positive function of $e_{tr_{n+1}}, \alpha_{mi}$ and C . Therefore, $\Delta \alpha_{mc}$ can be expressed as:

$$\begin{aligned} \Delta \alpha_{mc} &= q \min(|L_{c_1}^m|, |L_{c_2}^m|, |\mathbf{L}^{\mathbf{S}_m}|, |\mathbf{L}^{\mathbf{E}_m}|, |\mathbf{L}^{\mathbf{R}_m}|) = \text{sign}(e_{tr_{n+1}}) \Psi_m(e_{tr_{n+1}}, \alpha_{mi}, C) \\ &= \frac{e_{tr_{n+1}}}{|e_{tr_{n+1}}|} \Psi_m(e_{tr_{n+1}}, \alpha_{mi}, C) = e_{tr_{n+1}} \frac{\Psi_m(e_{tr_{n+1}}, \alpha_{mi}, C)}{|e_{tr_{n+1}}|} \\ &= \mu_m(e_{tr_{n+1}}, \alpha_{mi}, C) e_{tr_{n+1}} \end{aligned} \quad (4.54)$$

where $\mu_m(e_{tr_{n+1}}, \alpha_{mi}, C) \geq 0, \Psi_m(e_{tr_{n+1}}, \alpha_{mi}, C) \geq 0$. Thus, using (4.43) and (4.54) the adjustment rules for all Lagrange multipliers in set \mathbf{S}_m can be given as

$$\Delta \mathbf{\Omega}_m = \beta_m \Delta \alpha_{mc} = \beta_m \mu_m(e_{tr_{n+1}}, \alpha_{mi}, C) e_{tr_{n+1}} \quad (4.55)$$

By substituting (4.55) in (4.52) and expanding $\frac{\partial e_{tr_{n+1}}}{\partial u_n}$ as $\frac{\partial e_{tr_{n+1}}}{\partial y_{n+1}} \frac{\partial y_{n+1}}{\partial u_n} = -\frac{\partial y_{n+1}}{\partial u_n}$, equation (4.52) can be finalized as follows:

$$\begin{aligned} \frac{\partial V(e_{tr_{n+1}})}{\partial t} &= -e_{tr_{n+1}}^T \mathbf{P} \frac{\partial e_{tr_{n+1}}}{\partial u_n} \left[\sum_{m=1}^p \left[\frac{\partial g_{\text{controller}}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m} \beta_m \mu_m(e_{tr_{n+1}}, \alpha_{mi}, C) + \frac{\partial g_{\text{controller}}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m} \right] \right. \\ &\quad \left. + \sum_{k=1}^i \left[\frac{\partial g_{\text{controller}}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k} \right] \right] e_{tr_{n+1}} \\ \frac{\partial V(e_{tr_{n+1}})}{\partial t} &= -e_{tr_{n+1}}^T (\mathbf{Q} + \mathbf{W} + \mathbf{Z}) e_{tr_{n+1}} \end{aligned} \quad (4.56)$$

where $\frac{\partial y_{n+1}}{\partial u_n}$ is the system Jacobian approximated via system model (f_{model}) and

$$\begin{aligned} \mathbf{Q} &= \mathbf{P} \frac{\partial y_{n+1}}{\partial u_n} \sum_{m=1}^p \left[\frac{\partial g_{\text{controller}}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m} \beta_m \mu_m(e_{tr_{n+1}}, \alpha_{mi}, C) \right] \\ \mathbf{W} &= \mathbf{P} \frac{\partial y_{n+1}}{\partial u_n} \sum_{m=1}^p \left[\frac{\partial g_{\text{controller}}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m} \right] \\ \mathbf{Z} &= \mathbf{P} \frac{\partial y_{n+1}}{\partial u_n} \sum_{k=1}^i \left[\frac{\partial g_{\text{controller}}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_m} \right] \end{aligned} \quad (4.57)$$

In a nutshell, the stability conditions for closed-loop system in the sense of Lyapunov can be attained as follows:

- **Condition 1:** If $V(t) \geq 0$ and $\frac{\partial V(t)}{\partial t} \geq 0$, the stability of the closed-loop system is ensured
- **Condition 2:** If $\mathbf{Q} \geq 0$, $\mathbf{W} \geq 0$ and $\mathbf{Z} \geq 0$, the stability of the closed-loop system is ensured
- **Condition 3:** If $\mathbf{Q} \geq 0$, $\mathbf{W} \geq 0$ and $\mathbf{Z} \leq 0$ and $\|\mathbf{Q} + \mathbf{W}\| \geq \|\mathbf{Z}\|$, the stability of the closed-loop system is ensured
- **Condition 4:** If $\mathbf{Q} \geq 0$, $\mathbf{W} \leq 0$ and $\mathbf{Z} \geq 0$ and $\|\mathbf{Q} + \mathbf{Z}\| \geq \|\mathbf{W}\|$, the stability of the closed-loop system is ensured
- **Condition 5:** If $\mathbf{Q} \geq 0$, $\mathbf{W} \leq 0$ and $\mathbf{Z} \leq 0$ and $\|\mathbf{Q}\| \geq \|\mathbf{W} + \mathbf{Z}\|$, the stability of the closed-loop system is ensured
- **Condition 6:** If $\mathbf{Q} \leq 0$, $\mathbf{W} \geq 0$ and $\mathbf{Z} \geq 0$ and $\|\mathbf{W} + \mathbf{Z}\| \geq \|\mathbf{Q}\|$, the stability of the closed-loop system is ensured
- **Condition 7:** If $\mathbf{Q} \leq 0$, $\mathbf{W} \geq 0$ and $\mathbf{Z} \leq 0$ and $\|\mathbf{W}\| \geq \|\mathbf{Q} + \mathbf{Z}\|$, the stability of the closed-loop system is ensured
- **Condition 8:** If $\mathbf{Q} \leq 0$, $\mathbf{W} \leq 0$ and $\mathbf{Z} \geq 0$ and $\|\mathbf{Z}\| \geq \|\mathbf{Q} + \mathbf{W}\|$, the stability of the closed-loop system is ensured

In order to determine, \mathbf{Q} , \mathbf{Z} and \mathbf{W} , it is required to compute $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m}$, $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m}$ and $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k}$. The derivations of $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m}$, $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m}$ and $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k}$ change depending on the controller to be utilized. The derivations are given in sec 4.3.8.2 and sec 4.3.8.3 for PID type STR and Fuzzy PID type STR, respectively.

4.3.8.2 Derivation of the sensitivity functions for PID type STR based on SVR

In this subsection, the sensitivity of the control signal with respect to Lagrange multipliers of the parameters estimator ($\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m}$), the sensitivity of the control signal with respect to inputs of the parameters estimator ($\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m}$) and the sensitivity of the control signal with respect to inputs of the controller ($\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k}$) have been derived for PID type STR based on SVR. The computation

of $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m}$ is as in (4.58).

$$\begin{aligned}\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m} &= \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \boldsymbol{\theta}_m} \frac{\partial \boldsymbol{\theta}_m}{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})} \frac{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})}{\partial \mathbf{\Omega}_m} \\ \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m} &= x_m \left[\frac{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})}{\partial b_{estimator_m}} \frac{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})}{\partial \alpha_{m1}} \dots \frac{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})}{\partial \alpha_{mk}} \right] \\ \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m} &= x_m \mathbf{K}_{estimator_m}(\mathbf{\Pi}_{mc})\end{aligned}\quad (4.58)$$

The sensitivity $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m}$ is expressed as follows:

$$\begin{aligned}\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m} &= \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \boldsymbol{\theta}_m} \frac{\partial \boldsymbol{\theta}_m}{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})} \frac{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})}{\partial \mathbf{K}_{estimator_m}(\mathbf{\Pi}_{mc})} \frac{\partial \mathbf{K}_{estimator_m}(\mathbf{\Pi}_{mc})}{\partial \mathbf{\Pi}_m} \\ \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m} &= x_m \mathbf{\Omega}_m \frac{\partial \mathbf{K}_{estimator_m}(\mathbf{\Pi}_{mc})}{\partial \mathbf{\Pi}_m}\end{aligned}\quad (4.59)$$

The sensitivity $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k}$ is given as follows:

$$\begin{aligned}\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k} &= \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_1} \frac{\partial x_1}{\partial x_k} + \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_2} \frac{\partial x_2}{\partial x_k} \\ &\quad + \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_3} \frac{\partial x_3}{\partial x_k}\end{aligned}\quad (4.60)$$

Thus, the sensitivities for inputs can be obtained as

$$x_1 = e_n - e_{n-1} = x_2 - e_{n-1}$$

$$x_2 = e_n = x_2$$

$$x_3 = e_n - 2e_{n-1} + e_{n-2} = x_1 - 2e_{n-1} + e_{n-2} = x_2 - e_{n-1} + e_{n-2}$$

$$\begin{aligned}\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_1} &= \theta_1 + \theta_3 = K_{p_n} + K_{d_n} \\ \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_2} &= \theta_1 + \theta_2 + \theta_3 = K_{p_n} + K_{i_n} + K_{d_n} \\ \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_3} &= \theta_3 = K_{d_n}\end{aligned}\quad (4.61)$$

4.3.8.3 Derivation of the sensitivity functions for Fuzzy PID type STR based on SVR

The sensitivity of the control signal with respect to Lagrange multipliers of the parameters estimator ($\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m}$), the sensitivity of the control signal with respect to inputs of the parameters estimator ($\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m}$) and the sensitivity of the control signal with respect to inputs of the controller ($\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k}$) are derived for fuzzy PID type STR based on SVR as follows. The computation of $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m}$

is as follows:

$$\begin{aligned}
\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m} &= \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\theta}_m} \frac{\partial \mathbf{\theta}_m}{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})} \frac{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})}{\partial \mathbf{\Omega}_m} \\
\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m} &= \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\theta}_m} \left[\frac{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})}{\partial b_{estimator_m}} \frac{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})}{\partial \alpha_{m1}} \dots \frac{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})}{\partial \alpha_{mk}} \right] \\
\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Omega}_m} &= \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\theta}_m} \mathbf{K}_{estimator_m}(\mathbf{\Pi}_{mc})
\end{aligned} \tag{4.62}$$

The computation of $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\theta}_m} = \frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\theta}_m}$ is given as

$$\begin{aligned}
\frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial K_{e_n}} &= \frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})} \frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial e_{f_n}} \frac{\partial e_{f_n}}{\partial K_{e_n}} = (\beta_n + \Psi_n) \frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial e_{f_n}} x_1 \\
\frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial K_{de_n}} &= \frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})} \frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial \dot{e}_{f_n}} \frac{\partial \dot{e}_{f_n}}{\partial K_{de_n}} = (\beta_n + \Psi_n) \frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial \dot{e}_{f_n}} x_2 \\
\frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \beta_n} &= \frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \Psi_n} = f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})
\end{aligned} \tag{4.63}$$

The sensitivity $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m}$ is expressed as follows:

$$\begin{aligned}
\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m} &= \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\theta}_m} \frac{\partial \mathbf{\theta}_m}{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})} \frac{\partial f_{estimator_m}(\mathbf{\Omega}_m, \mathbf{\Pi}_{mc})}{\partial \mathbf{K}_{estimator_m}(\mathbf{\Pi}_{mc})} \frac{\partial \mathbf{K}_{estimator_m}(\mathbf{\Pi}_{mc})}{\partial \mathbf{\Pi}_m} \\
\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\Pi}_m} &= \frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial \mathbf{\theta}_m} \mathbf{\Omega}_m \frac{\partial \mathbf{K}_{estimator_m}(\mathbf{\Pi}_{mc})}{\partial \mathbf{\Pi}_m}
\end{aligned} \tag{4.64}$$

The sensitivity $\frac{\partial g_{controller}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k} = \frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k}$ is given as follows:

$$\begin{aligned}
\frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k} &= \frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})} \left(\frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial e_{f_n}} \frac{\partial e_{f_n}}{\partial x_k} + \frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial \dot{e}_{f_n}} \frac{\partial \dot{e}_{f_n}}{\partial x_k} \right) \\
\frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_k} &= (\beta_n + \Psi_n) \left(\frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial e_{f_n}} \frac{\partial e_{f_n}}{\partial x_k} + \frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial \dot{e}_{f_n}} \frac{\partial \dot{e}_{f_n}}{\partial x_k} \right)
\end{aligned} \tag{4.65}$$

$$\begin{aligned}
\frac{e_{f_n}}{\partial x_1} &= K_{e_n}, \quad \frac{\dot{e}_{f_n}}{\partial x_1} = K_{de_n}, \quad \frac{e_{f_n}}{\partial x_2} = 0, \quad \frac{\dot{e}_{f_n}}{\partial x_2} = K_{de_n} \\
\frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_1} &= (\beta_n + \Psi_n) \left(\frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial e_{f_n}} K_{e_n} + \frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial \dot{e}_{f_n}} K_{de_n} \right) \\
\frac{\partial f_{FLCPID}(\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{X})}{\partial x_2} &= (\beta_n + \Psi_n) \left(\frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial \dot{e}_{f_n}} K_{de_n} \right)
\end{aligned}$$

The term $\frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial e_{f_n}}$ and $\frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial \dot{e}_{f_n}}$ is computed as:

$$\begin{aligned}
\frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial e_{f_n}} &= \frac{(B_j(\dot{e}_{f_n})\Gamma_{2n} + B_{j+1}(\dot{e}_{f_n})\Gamma_{4n})^2 - (B_j(\dot{e}_{f_n})\Gamma_{1n} + B_{j+1}(\dot{e}_{f_n})\Gamma_{3n})^2}{e_{i+1} - e_i} \\
&\quad + (A_i(e_{f_n})B_j(\dot{e}_{f_n}))^2 \frac{k_{11}e^{-k_{11}e_{f_n}}}{(1 + e^{-k_{11}e_{f_n}})^2} + (A_{i+1}(e_{f_n})B_j(\dot{e}_{f_n}))^2 \frac{k_{21}e^{-k_{21}e_{f_n}}}{(1 + e^{-k_{21}e_{f_n}})^2} \\
&\quad + (A_i(e_{f_n})B_{j+1}(\dot{e}_{f_n}))^2 \frac{k_{31}e^{-k_{31}e_{f_n}}}{(1 + e^{-k_{31}e_{f_n}})^2} + (A_{i+1}(e_{f_n})B_{j+1}(\dot{e}_{f_n}))^2 \frac{k_{41}e^{-k_{41}e_{f_n}}}{(1 + e^{-k_{41}e_{f_n}})^2}
\end{aligned} \tag{4.66}$$

$$\begin{aligned} \frac{\partial f_{FLCPD}(e_{f_n}, \dot{e}_{f_n})}{\partial \dot{e}_{f_n}} = & \frac{(A_i(e_{f_n})\Gamma_{3_n} + A_{i+1}(e_{f_n})\Gamma_{4_n})^2 - (A_i(e_{f_n})\Gamma_{1_n} + A_{i+1}(e_{f_n})\Gamma_{2_n})^2}{\dot{e}_{j+1} - \dot{e}_j} \\ & + (A_i(e_{f_n})B_j(\dot{e}_{f_n}))^2 \frac{k_{12}e^{-k_{12}\dot{e}_{f_n}}}{(1 + e^{-k_{12}\dot{e}_{f_n}})^2} + (A_{i+1}(e_{f_n})B_j(\dot{e}_{f_n}))^2 \frac{k_{22}e^{-k_{22}\dot{e}_{f_n}}}{(1 + e^{-k_{22}\dot{e}_{f_n}})^2} \\ & + (A_i(e_{f_n})B_{j+1}(\dot{e}_{f_n}))^2 \frac{k_{32}e^{-k_{32}\dot{e}_{f_n}}}{(1 + e^{-k_{32}\dot{e}_{f_n}})^2} + (A_{i+1}(e_{f_n})B_{j+1}(\dot{e}_{f_n}))^2 \frac{k_{42}e^{-k_{42}\dot{e}_{f_n}}}{(1 + e^{-k_{42}\dot{e}_{f_n}})^2} \end{aligned}$$

4.4 Simulation Results

The performance of the generalized STR with adaptive PID and adaptive fuzzy PID controllers are evaluated on the bioreactor benchmark problem. Nevertheless, the SVR based self-tuning regulator scheme proposed in this paper can be implemented to control a diverse range of systems and to successfully solve fundamental control problems that frequently appear in practice such as nonlinearity, instability.

4.4.1 Bioreactor system

The bioreactor system is frequently used in technical literature as a benchmark nonlinear system so as to appraise and compare the performances of proposed control methodologies [5, 43, 60, 62]. A bioreactor is a vessel in which water, cells (e.g., yeast or bacteria) and nutrients (substrate) to be consumed by cells are mixed. As a result of this consuming, product (both desired and undesired) and more cells emerge [60]. This system is difficult to control since it has highly nonlinear dynamics and exhibits limit cycles [60]. The system dynamics can be represented via the following differential equations:

$$\begin{aligned} \dot{x}_1(t) &= -x_1(t)u(t) + x_1(t)(1 - x_2(t))e^{\frac{x_2(t)}{\gamma(t)}} \\ \dot{x}_2(t) &= -x_2(t)u(t) + x_1(t)(1 - x_2(t))e^{\frac{x_2(t)}{\gamma(t)}} \frac{1 + \beta(t)}{1 + \beta(t) - x_2(t)} \end{aligned} \quad (4.67)$$

where $x_1(t)$ symbolizes the cell concentration, $x_2(t)$ indicates the amount of nutrients in tank, $u(t)$ is the flow rate by which the bioreactor is controlled, $\gamma(t)$ is nutrient inhibition parameter, $\beta(t)$ is grow rate parameter [5, 43, 60–62]. In the closed-loop system, the aim is to control the cell concentration ($y(t) = x_1(t)$) by manipulating the flow rate ($u(t)$). The limitations for the magnitude of the control signal are $u_{min} = 0$ and $u_{max} = 2$. The continuation period of control signal is kept constant at $\tau_{min} = \tau_{max} = 0.5s$. The bioreactor system has been simulated using the proposed STR architecture with both PID and fuzzy PID controllers in the generalized controller block. Since

we have assumed that the dynamics of the system is not known, online SVR has been utilized to identify the unknown dynamics using the input-output data pairs. The input feature vector for $\text{SVR}_{\text{model}}$ is obtained as $\mathbf{M}_c = [u_n \cdots u_{n-n_u}, y_n \cdots y_{n-n_y}]^T$ where $n_u = n_y = 2$. Simulations have been performed for three separate cases. 1) Nominal case with no measurement noise and parametric uncertainty 2) Measurement noise is added to the controlled output of the system 3) Parametric uncertainty is introduced to the system. For nominal and measurement noise cases the input feature vectors are selected as $\mathbf{\Pi}_p = [r_n \ P_n \ u_{n-1}]^T$, $\mathbf{\Pi}_i = [r_n \ I_n \ I_{n-1} \ I_{n-2} \ y_n \ u_{n-1}]^T$ and $\mathbf{\Pi}_d = [r_n \ D_n \ u_{n-1}]^T$ for $\text{SVR}_{\text{estimator}}$ of PID type STR, and $\mathbf{\Pi}_{ke} = [r_n \ P_n \ u_{n-1}]^T$, $\mathbf{\Pi}_{kde} = [r_n \ I_n \ y_n]^T$, $\mathbf{\Pi}_\Psi = [r_n \ D_n]^T$, $\mathbf{\Pi}_\beta = [r_n \ I_n \ y_n \ u_{n-1}]^T$ are employed as input feature vector for $\text{SVR}_{\text{estimator}}$ of fuzzy PID STR where $P_n = e_n - e_{n-1}$, $I_n = e_n$, $D_n = e_n - 2e_{n-1} + e_{n-2}$ and $e_n = r_n - y_n$. For the case with parametric uncertainty, the input feature vectors for $\text{SVR}_{\text{estimator}}$ of PID type STR are utilized as $\mathbf{\Pi}_p = [r_n \ P_n \ y_n \ u_{n-1}]^T$, $\mathbf{\Pi}_i = [r_n \ I_n \ I_{n-1} \ I_{n-2} \ y_n \ u_{n-1}]^T$ and $\mathbf{\Pi}_d = [r_n \ D_n \ y_n \ u_{n-1}]^T$, and $\mathbf{\Pi}_{ke} = [r_n \ I_n \ I_{n-1} \ D_n \ y_n \ u_{n-1}]^T$, $\mathbf{\Pi}_{kde} = \mathbf{\Pi}_\Psi = \mathbf{\Pi}_\beta = \mathbf{\Pi}_{ke}$ are deployed as input feature vector for $\text{SVR}_{\text{estimator}}$ of fuzzy PID STR.

4.4.2 Nominal case with no noise and parametric uncertainty

The tracking performance of both controllers for the case when no noise and parametric uncertainty is applied to the system and all parameters are fully known is given in Figure 4.12. The system accurately tracks the reference input. The controller parameters are depicted in Figure 4.13-4.14. Number of the support vector for both controller and system models are illustrated in Figure 4.15. As can be seen from Figure 4.16, the controllers track the sinusoidal reference input accurately.

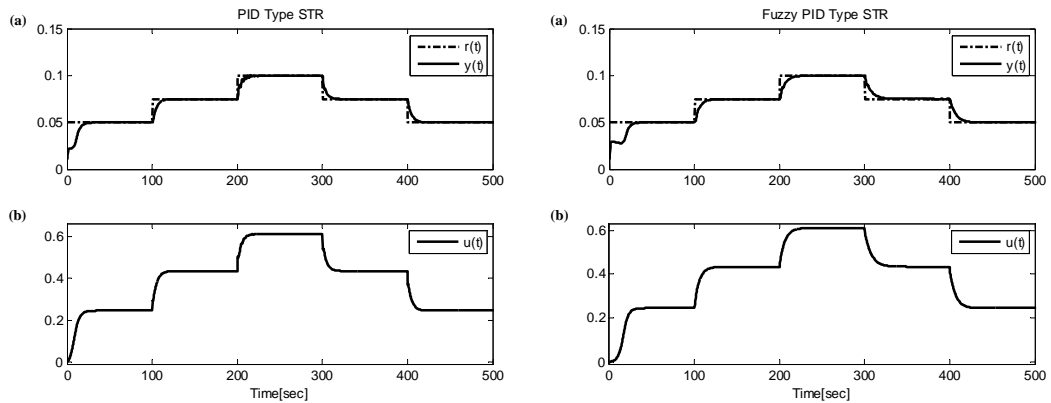


Figure 4.12 : System output (a), control signal (b) for the case with no noise and parametric uncertainty.

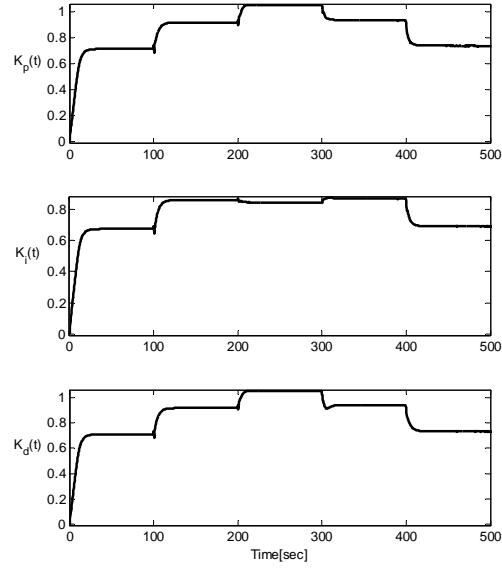


Figure 4.13 : PID type STR parameters.

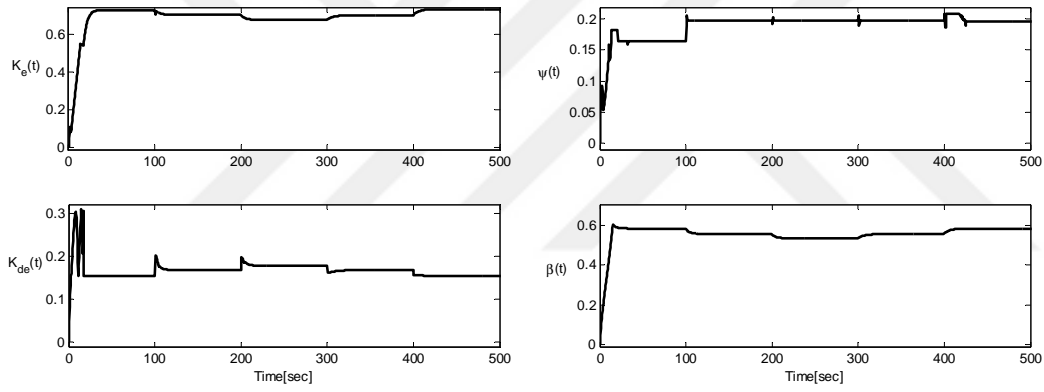


Figure 4.14 : Fuzzy PID type STR parameters.

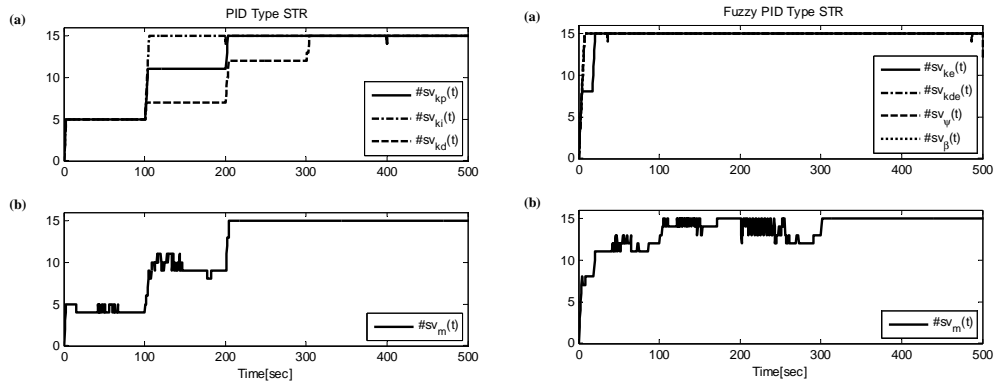


Figure 4.15 : Number of the support vectors for controllers (a) and system models (b).

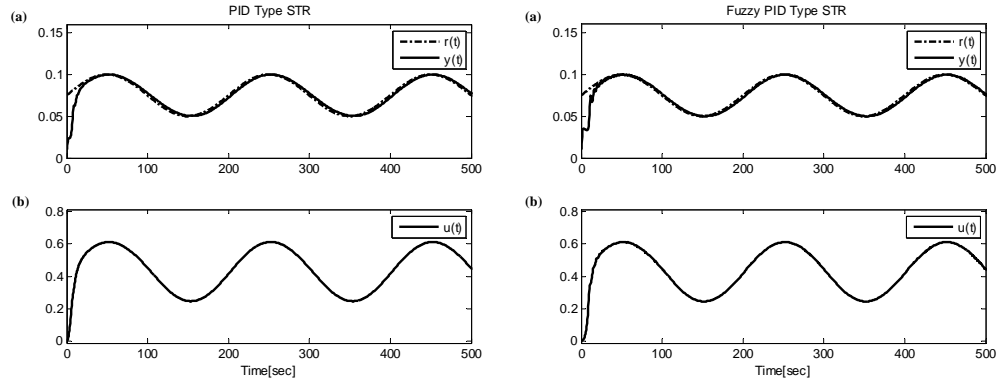


Figure 4.16 : System output (a), control signal (b) for sinusoidal input.

4.4.3 Measurement noise

The performance of the controllers under the influence of measurement noise is evaluated by adding a 30 dB measurement noise to the system output. The tracking performance and control input for controller are demonstrated in Figure 4.17.

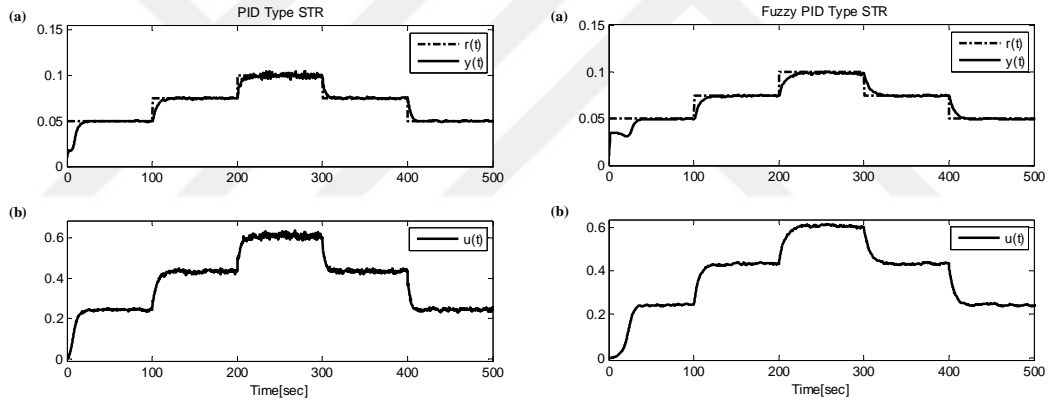


Figure 4.17 : System output (a), control signal (b) for the case with measurement noise (30 dB).

4.4.4 Uncertainty in system parameters

The robustness of the controllers are examined with respect to parametric uncertainty, $\gamma(t)$ is presumed as the uncertain system parameter, where its nominal value is $\gamma_{nom}(t) = 0.48$ and alters around its nominal value as $\gamma(t) = 0.48 + 0.06 \sin(0.016\pi t)$. Figure 4.18 illustrates the tracking performance of the controllers and control signal applied to the system in this case. By comparing Figure 4.18 with the nominal case given in Figure 4.12, it can be perceived that uncertainty caused by the time varying parameter is tolerated.

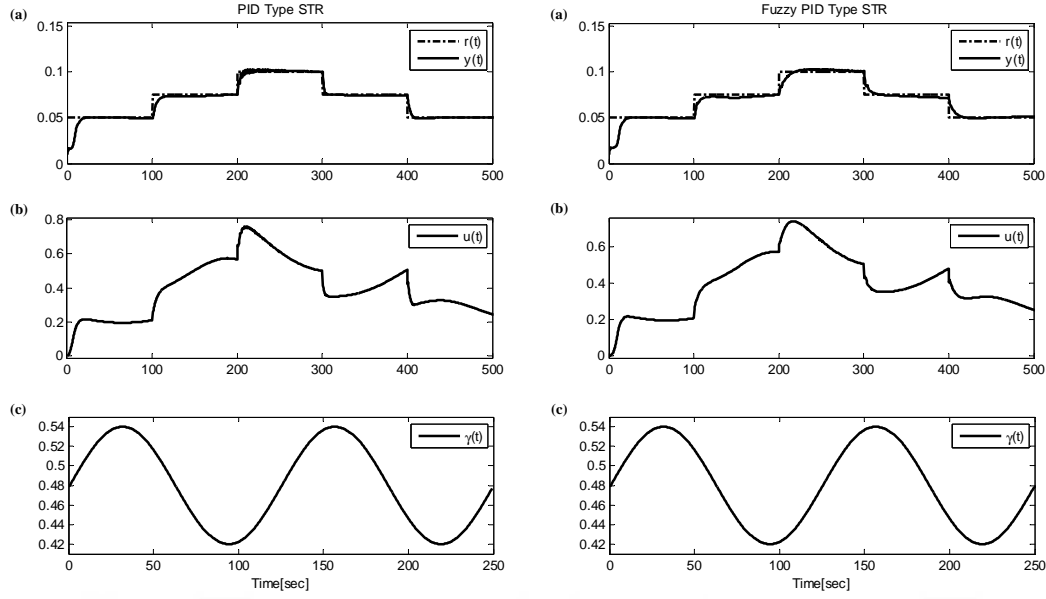


Figure 4.18 : System output (a), control signal (b) for the case with parametric uncertainty (c).

4.4.5 Closed-loop Lyapunov stability analysis

The stability analysis of the proposed control methodology derived in section 4.3.8 is carried out, and the numerical verifications for both PID type STR and Fuzzy PID type STR are depicted in Figures 4.19-4.21 for noiseless case, and when measurement noise and parametric uncertainty are added respectively. For stability in the sense of Lyapunov, both $V(e_{tr_{n+1}}) \geq 0$ and $\frac{\partial V(e_{tr_{n+1}})}{\partial t} \leq 0$ must be ensured simultaneously. As illustrated in Figures 4.19-4.21, it has been observed that $V(e_{tr_{n+1}}) \geq 0$ and $\frac{\partial V(e_{tr_{n+1}})}{\partial t} \leq 0$ for both controller during the course of control. In a nutshell, it can be inferred that the closed-loop systems are stable for all cases.

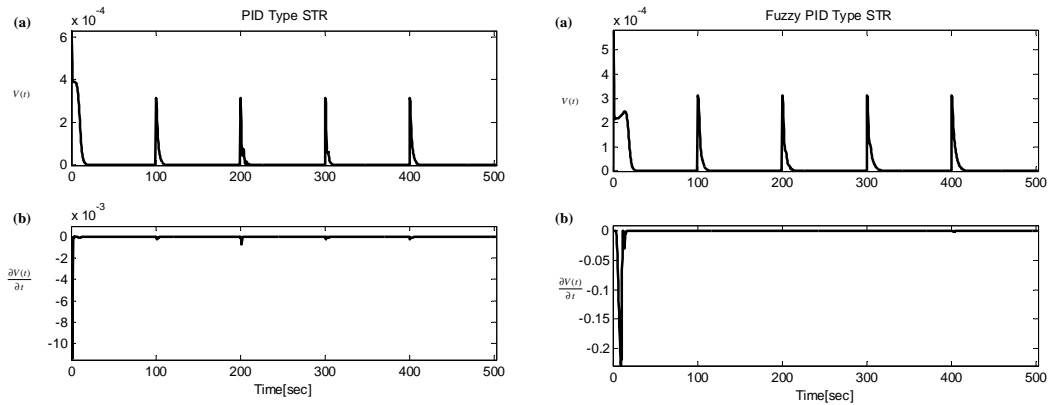


Figure 4.19 : $V(t)$ (a) and $\frac{\partial V(t)}{\partial t}$ (b) for the case with no noise and parametric uncertainty.

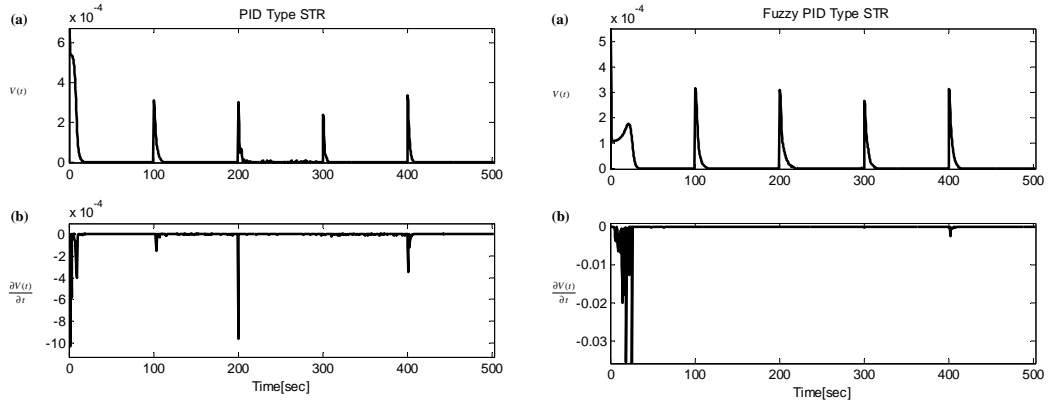


Figure 4.20 : $V(t)$ (a) and $\frac{\partial V(t)}{\partial t}$ (b) for the case with measurement noise.

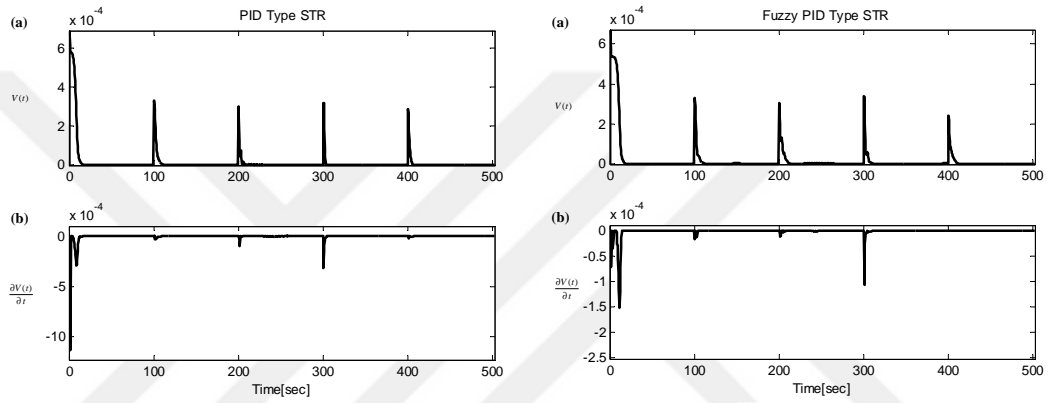


Figure 4.21 : $V(t)$ (a) and $\frac{\partial V(t)}{\partial t}$ (b) for the case with parametric uncertainty.

4.4.6 Comparison of the results with SVM-based PID controller

In order to evaluate the performance comparisons of the controllers, SVM-based PID controller proposed by Iplikci [5] is deployed to control bioreactor system for cases with no noise and parametric uncertainty, with measurement noise added to the system and with parametric uncertainty. In a nutshell, the adjustment mechanism proposed for SVM-based PID controller optimizes the parameters of a PID controller via Levenberg-Marquardt algorithm by utilizing K-step ahead future Jacobian of the system behaviour. The proposed mechanism includes five components : classical PID controller, controller parameter tuner, SVR NARX model of the system, control signal correction block and line search block. The PID controller has three tunable parameters (K_p , K_i , K_d) adjusted via Levenberg-Marquardt algorithm in controller parameter tuner block. SVR NARX model approximates K-step ahead Jacobians of the system in order to constitute Jacobian matrix which is required in Levenberg-Marquardt optimization algorithm. Since the updated controller parameters may not good enough to force the

system output to follow the desired trajectory in transient-state because of modeling inaccuracies and external disturbances, a control signal correction term which is derived via Taylor expansion of control signal is utilized in control signal correction block [3,5]. Line search block calculates the the optimal learning rate for control signal correction term via golden section method. The closed-loop tracking performances of the controllers proposed in this paper are compared with the closed-loop tracking competency of SVM-based PID controller given in [5]. When the prediction horizon of the SVM-based PID controller is increased, it is possible to obtain better or similar results than PID type STR and Fuzzy PID type STR. In order to compare the controller under the same conditions and obtain meaningful results, the prediction horizon (K) of SVM-based PID controller is set as " $K = 1$ ". The closed-loop performances of the controllers for the case with no noise and parametric uncertainty and the case with measurement noise are illustrated in Figure 4.22 (left and right) respectively. The robustnesses of the controllers with respect to parametric uncertainty are evaluated in Figure 4.23. In order to compare controller performances numerically, the following performance index function is employed

$$J_{\text{comp}} = \sum_{n=0}^{\infty} [r_{n+1} - y_{n+1}]^2 \quad (4.68)$$

and the performance comparisons are depicted in Figure 4.24.

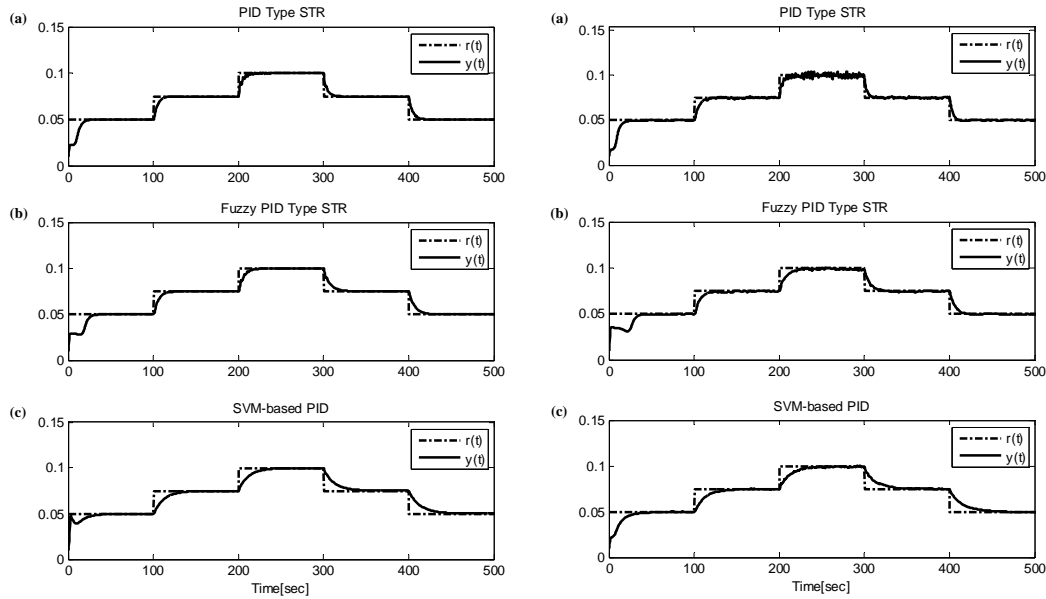


Figure 4.22 : Comparison of controllers for the case with no noise and parametric uncertainty (left) and for the case with measurement noise (right).

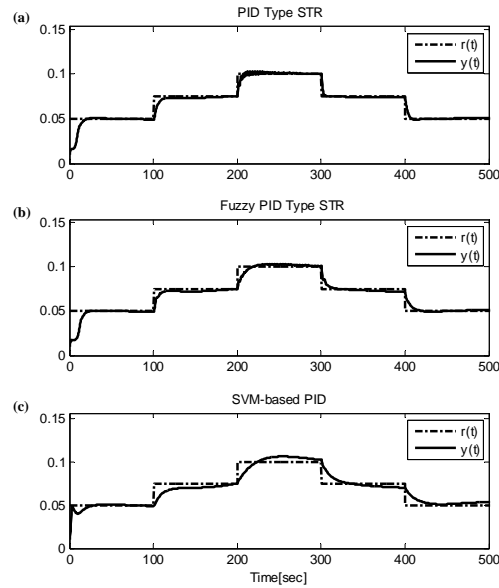


Figure 4.23 : Comparison of controllers for the case with parametric uncertainty.

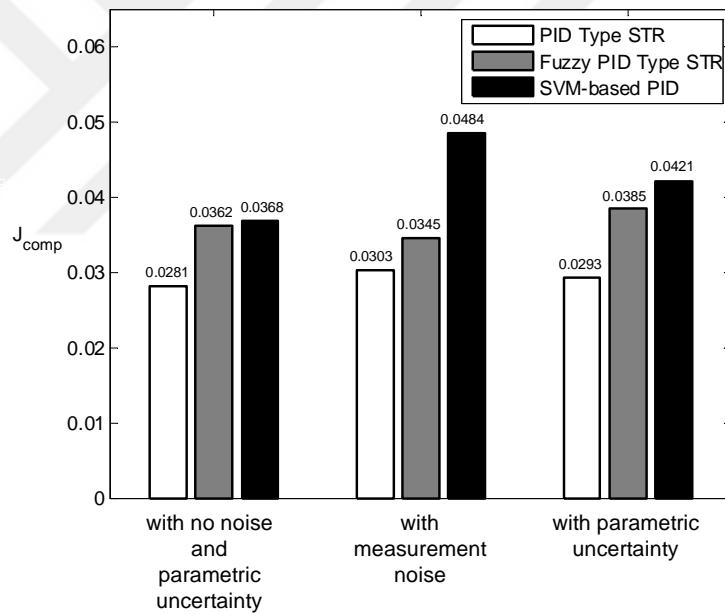


Figure 4.24 : Performance comparison of controllers with respect to the defined performance index (4.68).

As can be seen from Figure 4.24, PID type STR has the best performance for all cases. The fuzz PID type STR has better performance than SVM-based PID controller for all cases. Comparison of the PID type STR and SVM-based PID performances is more meaningful since the adaptation performance of the STR and Levenberg Marquardt can be comparable. Therefore, SVR based PID type STR has better tracking performances than SVM-based PID controller for all cases. In order to exhaustively compare the

transient state and steady state behaviour of the controllers, the accuracies of the controllers are compared in terms of maximum and average values of steady state errors, settling times (according to 2% error criterion) and overshoots separately for cases with no noise, with noise and with parametric uncertainty. The results are tabulated in Table 4.1 and 4.2 respectively. As can be seen from Figure 4.24, and

Table 4.1 : Maximum values of steady state errors ($e_{ss}(\%)$), settling times ($t_s(2\%)$ (sec)) and overshoots ($P.O.(\%)$).

Controllers	$e_{ss}(\%)$			$t_s(2\%)$			$P.O.(\%)$		
	Noiseless	Noisy	Uncert.	Noiseless	Noisy	Uncert.	Noiseless	Noisy	Uncert.
PID Type STR	0.002	2	2	19.5	27.5	17	1.2605	18.0158	12.9661
Fuzzy PID Type STR	0.0019	2	2	27	34	19.5	0.0016	5.1059	14.5094
SVM-based PID	0.0559	1.185	2	31	36	72.5	0	3.5891	24.5359

Table 4.2 : Average values of steady state errors ($e_{ss}(\%)$), settling times ($t_s(2\%)$ (sec)) and overshoots ($P.O.(\%)$).

Controllers	$e_{ss}(\%)$			$t_s(2\%)$			$P.O.(\%)$		
	Noiseless	Noisy	Uncert.	Noiseless	Noisy	Uncert.	Noiseless	Noisy	Uncert.
PID Type STR	8.4611×10^{-4}	1.4166	1.6691	10.6	14.5	7.9	0.2530	6.7918	4.5779
Fuzzy PID Type STR	8.0872×10^{-4}	0.9027	1.7872	15.6	17.2	12.7	5.3×10^{-4}	2.9685	6.1315
SVM-based PID	0.0363	0.5211	1.7716	25.3	27.2	27.8	0	1.9421	9.0258

Table 4.1 and 4.2, the controller with the best transient and steady state behaviour is PID type STR. The results given in Table 4.1 and 4.2 verify the results given in Figure 4.24.

4.5 Conclusion

Support vector machines have successfully been employed to cope with various classification and regression problems for nearly two decades. Their performance is justified to be superior than gradient based intelligent systems like ANNs, ANFISs due to their convex objective function and better generalization property. However, they have not been used as controllers since information about control input to be applied to the system, which is required for SVR training is not available beforehand. In this paper, a novel architecture where an online SVR is used to tune the parameters of a generalized STR which optimizes the margin between reference input and system output has been proposed. There are two online SVR structures employed in the control system, SVR_{model} calculates the model of the controlled system and predicts its future behaviour and $SVR_{\text{estimator}}$ estimates the controller parameters of the STR.

Two different controllers have been used in the controller block: adaptive PID and adaptive fuzzy PID. A separate $SVR_{\text{estimator}}$ is designed for each tunable parameter in the controllers.

The performance of the proposed generalized adaptive control architecture and parameter estimator is evaluated for both PID and fuzzy PID controllers on a bioreactor benchmark system. A comprehensive stability analysis of the generalized STR is performed. Furthermore, the closed-loop tracking performances of the STRs are compared with SVM-based PID controller proposed by Iplikci. The robustnesses of the controllers have also been assessed for the noiseless case and when measurement noise and parametric uncertainty are added. Simulation results indicate that the proposed adaptation mechanism for generalized STR accomplishes successful tracking performance as well as good noise rejection and high toleration to parametric uncertainties.

In future works, new SVR type adjustment mechanisms can be developed by employing closed-loop margin notion.

Conflict of Interest

The authors declare that there is no conflict of interests regarding the publication of this paper.



5. CONCLUSIONS AND RECOMMENDATIONS

In this PhD. thesis, three novel SVR based adaptive controller structures are introduced for nonlinear systems. In order to deploy SVR as a controller or parameter estimator, "closed-loop margin" notion is proposed to adjust the parameters of SVR based structures via tracking error without requiring the controller input as training data. Thus, closed-loop margin notion paves the way for direct utilization of SVR as a controller and parameter estimator.

In the first proposed mechanism, $SVR_{NARMA-L2}$ controller law is designed via SVR_{NARX} model of the system by achieving the $SVR_{NARMA-L2}$ submodels from previously obtained SVR_{NARX} model. For this purpose, conversion parameters which provide correlation among SVR_{NARX} system model and $SVR_{NARMA-L2}$ system model have been utilized. Since the conversion parameters have crucial impact on control performance, Levenberg-Marquardt algorithm has been employed to optimize conversion parameters by considering K-step ahead future system behaviour. Thus, strong identification competency of SVR and the simplicity of the NARMA model are successfully utilized in derivation of the control law.

In the second adaptive mechanism, SVR has been directly deployed as a controller. In the overall architecture, two separate SVR's are used: $SVR_{controller}$ is used as the controller and SVR_{model} is utilized to observe the dynamic alterations on system behaviour resulting from the parameter adjustment of $SVR_{controller}$. Closed-loop margin notion has been introduced to derive adaptation rules for $SVR_{controller}$.

Finally, a generalized self-tuning regulator mechanism which can be realized for any controller with adjustable parameters is introduced for nonlinear dynamical systems by reconfiguring the closed-loop margin notion for STR's. As in the second control architecture, SVR_{model} is used to approximate the dynamics of the system while a separate $SVR_{estimator}$ is designed for each tunable parameter of the controller.

The performance evaluations of the controllers have been performed on nonlinear continuously stirred tank reactor (CSTR) system or/and bioreactor benchmark system by simulation studies. The robustnesses of the controller against system parameter uncertainty and measurement noise have been evaluated. Also, the success of the control architectures are compared with the performance of a SVM-based PID controller proposed in [5]. Additionally, the stability analysis of closed-loop system for $SVR_{\text{controller}}$ and STR based on $SVR_{\text{estimator}}$ are conducted in detail.

It is significant to note that the implementation of intelligent methods to fast systems such as robot arm, servo systems etc is a hard task since the adjustment mechanisms have excessive computational load. In SVR based algorithms, most of this computational load is due to quadratic programming (QP) solver. In this thesis, it is justified that the proposed adjustment rules can be realized in real time control of nonlinear systems with large time constants. Yet, it is believed that the computational loads of SVR are substantially extenuated by courtesy of mathematical developments conducted to enhance QP solvers and developments in CPU technology. Depending on these developments, the proposed control mechanisms can be deployed to control nonlinear systems with small time constants also, in near future.

As future work, the closed-loop notion proposed in this thesis can be combined with other control methodologies to lead up to the design of various other SVR-based adaptive controllers. It is also possible to expand the proposed structures to control MIMO (multi input-multi output) systems. Additionally, predictive adjustment mechanisms can be interfused with $SVR_{\text{controller}}$ and STR based on $SVR_{\text{estimator}}$ architectures to enhance control performance.

REFERENCES

- [1] **Smola, A. and Scholkopf, B.** (2004). A tutorial on support vector regression, *Statistics and Computing*, 14(3), 199–222.
- [2] **Wang, X., Du, Z., Chen, J. and Pan, F.** (2009). Dynamic Modeling of Biotechnical Process Based on Online Support Vector Machine, *Journal of Computers*, 4(3), 251–258.
- [3] **Uçak, K. and Günel Öke, G.** (2016). An adaptive support vector regressor controller for nonlinear systems, *Soft Computing*, 20(7), 2531–2556.
- [4] **Ma, J., Theiler, J. and Perkins, S.** (2003). Accurate on-line support vector regression, *Neural Computation*, 15(11), 2683–2703.
- [5] **İplikci, S.** (2010). A comparative study on a novel model-based PID tuning and control mechanism for nonlinear systems, *International Journal of Robust and Nonlinear Control*, 20(13), 1483–1501.
- [6] **Uçak, K. and Günel Öke, G.** (2016). Generalized Self-Tuning Regulator Based on Online Support Vector Regression, *Neural Computing and Applications*.
- [7] **Butler, H.** (1992). *Model reference adaptive control: from theory to practice*, Prentice Hall College Div.
- [8] **Cheng, G.,** (2006). *Process Control and Optimization: Model-Free Adaptive(MFA) Control*, CRC Press, New York, 4. edition.
- [9] **Sastry, S. and Bodson, M.** (1989). *Adaptive Control: Stability, Convergence and Robustness*, Prentice Hall.
- [10] **Cortes, C. and Vapnik, V.** (1995). Support-Vector Networks, *Machine Learning*, 20(3), 273–297.
- [11] **Drucker, H., Burges, C.J., Kaufman, L., Smola, A. and Vapnik, V.** (1996). Support vector regression machines, *Conference on Neural Information Processing Systems (NIPS)*, DENVER, USA.
- [12] **Vapnik, V., Golowich, S.E. and Smola, A.** (1996). Support vector method for function approximation, regression estimation, and signal processing, *Conference on Neural Information Processing Systems (NIPS)*, DENVER, USA.
- [13] **Uçak, K. and Günel Öke, G.** (2016). A Novel Adaptive NARMA-L2 Controller based on Online Support Vector Regression for Nonlinear Systems, *Neural Processing Letters*.

- [14] **Narendra, K.S. and Mukhopadhyay, S.** (1997). Adaptive control using neural networks and approximate models, *IEEE Transactions on Neural Networks*, 8(3), 475–485.
- [15] **Majstorovic, M., Nikolic, I., Radovic, J. and Kvascey, G.** (2008). Neural network control approach for a two-tank system, *Symposium on Neural Network Applications in Electrical Engineering*, Belgrade, Serbia.
- [16] **Pedro, J.O., Nyandoro, O.T.C. and John, S.** (2009). Neural Network Based Feedback Linearisation Slip Control of an Anti-Lock Braking System, *Asian Control Conference*, Hong Kong, China.
- [17] **De Jesus, O., Pukrittayakamee, A. and Hagan, M.** (2001). A comparison of neural network control algorithms, *IEEE International Joint Conference on Neural Networks (IJCNN)*, Washington, D.C.
- [18] **Pukrittayakamee, A., De Jesus, O. and Hagan, M.T.** (2002). Smoothing the control action for NARMA-L2 controllers, *Midwest Symposium on Circuits and Systems*, Tulsa, OK.
- [19] **Hagan, M.T., Demuth, H.B. and De Jesus, O.** (2002). An introduction to the use of neural networks in control systems, *International Journal of Robust and Nonlinear Control*, 12(11), 959–985.
- [20] **Wahyudi, M., Mokri, S.S. and Shafie, A.A.** (2008). Real time implementation of NARMA L2 feedback linearization and Smoothed NARMA L2 controls of a single link manipulator, *International Conference on Computer and Communication Engineering*, Kuala Lumpur, Malaysia.
- [21] **Akbarimajd, A. and Kia, S.** (2010). NARMA-L2 Controller for 2-DoF Underactuated Planar Manipulator, *International Conference on Control, Automation, Robotics and Vision*, Singapore.
- [22] **Vesselenyi, T., Dzitac, S., Dzitac, I. and Manolescu, M.J.** (2007). Fuzzy and neural controllers for a pneumatic actuator, *International Journal of Computers, Communications and Control*, 2(4), 375–387.
- [23] **Efe, M.O. and Kaynak, O.** (2000). A comparative study of soft-computing methodologies in identification of robotic manipulators, *Robotics and Autonomous Systems*, 30(3), 221–230.
- [24] **Efe, M.O. and Kaynak, O.** (1999). A comparative study of neural network structures in identification of nonlinear systems, *Mechatronics*, 9(3), 287–300.
- [25] **Denai, M.A., Palis, F. and Zeghib, A.** (2004). ANFIS Based Modelling and Control of Non-linear Systems: A tutorial, *International Conference on Systems, Man and Cybernetics*, Netherlands.
- [26] **Gretton, A., Doucet, A., Herbrich, R., Rayner, P.J.W. and Scholkopf, B.** (2001). Support vector regression for black-box system identification, *IEEE Workshop on Statistical Signal Processing*, Singapore.

- [27] **Rong, H., Zhang, G. and Zhang, C.** (2005). Application of support vector machines to nonlinear system identification, *International Symposium on Autonomous Decentralized Systems*, Chengdu, China.
- [28] **Suykens, J.** (2001). Nonlinear modelling and support vector machines, *IEEE Instrumentation and Measurement Technology Conference*, Budapest, Hungary.
- [29] **Ucak, K. and Gunel, O.** (2011). Adaptive PID controller based on online LSSVR with kernel tuning, *International Symposium on Innovations in Intelligent Systems and Applications*, Istanbul, Turkey.
- [30] **Vapnik, V.N.** (1999). An overview of statistical learning theory, *IEEE Transactions on Neural Networks*, 10(5), 988–999.
- [31] **Shang, W., Zhao, S. and Shen, Y.** (2008). Adaptive PID Controller Based on Online LSSVM Identification, *IEEE ASME International Conference on Advanced Intelligent Mechatronics*, Xian, China.
- [32] **Zhao, J., Li, P. and Wang, X.s.** (2009). Intelligent PID Controller Design with Adaptive Criterion Adjustment via Least Squares Support Vector Machine, *Chinese Control and Decision Conference*, Guilin, China.
- [33] **Yuan, X., Wang, Y. and Wu, L.** (2008). Composite feedforward-feedback controller for generator excitation system, *Nonlinear Dynamics*, 54(4), 355–364.
- [34] **Takao, K., Yamamoto, T. and Hinamoto, T.** (2006). A design of PID controllers with a switching structure by a support vector machine, *IEEE International Joint Conference on Neural Networks*, Vancouver, Canada.
- [35] **Liu, X., Yi, J. and Zhao, D.** (2005). Adaptive inverse control system based on least squares support vector machines, *International Symposium on Neural Networks*, Chongqing, China.
- [36] **Wang, H., Pi, D. and Sun, Y.** (2007). Online SVM regression algorithm-based adaptive inverse control, *Neurocomputing*, 70(4-6), 952–959.
- [37] **Yuan, X.F., Wang, Y.N. and Wu, L.H.** (2008). Adaptive inverse control of excitation system with actuator uncertainty, *Neural Processing Letters*, 27(2), 125–136.
- [38] **Zhao, Z., Liu, Z., Xia, Z. and Zhang, J.** (2012). Internal Model Control Based on LS-SVM for a Class of Nonlinear Process, *International Conference on Solid State Devices and Materials Science*, Macao, China.
- [39] **Zhong, W., Pi, D., Sun, Y., Xu, C. and Chu, S.** (2006). SVM based internal model control for nonlinear systems, *International Symposium on Neural Networks*, Chengdu, China.
- [40] **Datta, A. and Ochoa, J.** (1996). Adaptive inverse control of excitation system with actuator uncertainty, *Automatica*, 32(2), 261–266.

- [41] **Sun, C. and Song, J.** (2007). An adaptive internal model control based on LS-SVM, *International Symposium on Neural Networks*, Nanjing, China.
- [42] **Wang, Y.N. and Yuan, X.F.** (2008). SVM Approximate-based Internal Model Control Strategy, *Acta Automatica Sinica*, 34(2), 172–179.
- [43] **Iplikci, S.** (2006). Online trained support vector machines-based generalized predictive control of non-linear systems, *International Journal of Adaptive Control and Signal Processing*, 20(10), 599–621.
- [44] **Iplikci, S.** (2006). Support vector machines-based generalized predictive control, *International Journal of Robust and Nonlinear Control*, 16(17), 843–862.
- [45] **Camacho, E.** (1993). Constrained Generalized Predictive Control, *IEEE Transactions on Automatic Control*, 38(2), 327–332.
- [46] **Clarke, D. and Mohtadi, C.** (1989). Properties of generalized predictive control, *Automatica*, 25(6), 859–875.
- [47] **Camacho, E.F. and Bordons, C.** (1999). *Model Predictive Control*, Springer-Verlag.
- [48] **Clarke, D., Mohtadi, C. and Tuffs, P.** (1987). Generalized Predictive Control - Part I, *Automatica*, 23(2), 137–148.
- [49] **Du, Z. and Wang, X.** (2008). Nonlinear Generalized Predictive Control Based on Online SVR, *International Symposium on Intelligent Information Technology Application*, Shanghai.
- [50] **Shin, J., Kim, H.J., Park, S. and Kim, Y.** (2010). Model predictive flight control using adaptive support vector regression, *Neurocomputing*, 73(4-6), 1031–1037.
- [51] **Abu Rub, H. and Awwad, A.** (2009). Artificial Neural Networks and Fuzzy Logic Based Control of AC Motors, *IEEE International Electric Machines and Drives Conference*, Miami, FL.
- [52] **Norgaard, M., Ravn, O., Poulsen, N. and Hansen, L.** (2000). *Neural Networks for Modelling and Control of Dynamic Systems*, Springer-Verlag.
- [53] **Ng, G.** (1997). *Application of Neural Networks to Adaptive Control of Nonlinear Systems*, Research Studies Press.
- [54] **Cristianini, N. and Shawe-Taylor, J.** (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press.
- [55] **Martin, M.** (2002). On-line support vector machine regression, *European Conference on Machine Learning (ECML)*, Helsinki, Finland.
- [56] **Luenberger, D.G. and Ye, Y.** (2008). *Linear and Nonlinear Programming*, Springer.

- [57] **Griva, E., Nash, S.G. and Sofer, A.** (2009). *Linear and Nonlinear Optimization*, SIAM.
- [58] **Nocedal, J. and Wright, S.J.** (1999). *Numerical Optimization*, Springer.
- [59] **Cauwenberghs, G. and Poggio, T.** (2000). Incremental and Decremental Support Vector Machine Learning, *Annual Neural Information Processing Systems Conference*, Denver, USA.
- [60] **Ungar, L.H.** (1996). *Neural Networks for Control: A Bioreactor Benchmark for Adaptive Network based Process Control*, MIT Press.
- [61] **Efe, M., Abadoglu, E. and Kaynak, O.** (1999). A novel analysis and design of a neural network assisted nonlinear controller for a bioreactor, *International Journal of Robust and Nonlinear Control*, 9(11), 799–815.
- [62] **Efe, M.O.** (2007). Discrete Time Fuzzy Sliding Mode Control of a Biochemical Process, *International Conference on Automatic Control, Modeling and Simulation*, Istanbul, Turkey.
- [63] **Astrom, K. and B., W.** (2008). *Adaptive Control*, Dover Publications.
- [64] **Haykin, S.** (1999). *Neural Networks: A Comprehensive Foundation*, Prentice Hall.
- [65] **Wu, K. and Wang, S.** (2009). Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space, *Pattern Recognition*, 42(5), 710–717.
- [66] **Han, F., Wang, Z., Lei, M. and Zhou, Z.** (2008). An Iterative Modified Kernel for Support Vector Regression, *IEEE International Conference on Cybernetic Intelligent Systems*, Chengdu, China.
- [67] **Xu, Z., Dai, M. and Meng, D.** (2009). Fast and Efficient Strategies for Model Selection of Gaussian Support Vector Machine, *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, 39(5), 1292–1307.
- [68] **Saadia, N., Amirat, Y., Pontnau, J. and MSirdi, N.** (2001). Neural hybrid control of manipulators, stability analysis, *Robotica*, 19(1), 41–51.
- [69] **Kravaris, C. and Palanki, S.** (1988). Robust Nonlinear State Feedback Under Structured Uncertainty, *AIChE Journal*, 34(7), 1119–1127.
- [70] **Wu, W. and Chou, Y.** (1999). Adaptive feedforward and feedback control of non-linear time-varying uncertain systems, *International Journal of Control*, 72(12), 1127–1138.
- [71] **Levenspiel, O.** (1999). *Chemical Reaction Engineering*, John Wiley and Sons.
- [72] **Fogler, H.S.** (2006). *Elements of Chemical Reaction Engineering*, Prentice Hall Professional Technical Reference.
- [73] **Bobal, V., Bohm, J., Fessl, J. and Machacek, J.** (2005). *Digital Self-tuning Controller*, Springer.

- [74] **Astrom, K.J.** (1983). Theory and Applications of Adaptive Control-A Survey, *Automatica*, 19(5), 471–486.
- [75] **Wellstead, P.E., Liptak, B.G. and Renganathan, S.** (2006). *Process Control and Optimization: Self-tuning Controllers*, CRC Press.
- [76] **Aström, K., U., B., L., L. and Wittenmark, B.** (1977). Theory and Applications of Self-Tuning Regulators, *Automatica*, 13(5), 457–476.
- [77] **Akhyar, S. and Omatu, S.** (1993). Self-Tuning PID Control by Neural Networks, *International Joint Conference on Neural Network*, Nagoya, Japan.
- [78] **Wang, G.J., Fong, C.T. and Chang, K.J.** (2001). Neural-network-based self-tuning PI controller for precise motion control of PMAC motors, *IEEE Transactions on Industrial Electronics*, 48(2), 408–415.
- [79] **Shu, H.L. and Pi, Y.G.** (2000). PID neural networks for time-delay systems, *Computers and Chemical Engineering*, 24(2-7), 859–862.
- [80] **Spooner, J.T. and Passino, K.** (1996). Stable Adaptive Control Using Fuzzy Systems and Neural Networks, *IEEE Transactions on Fuzzy Systems*, 4(3), 339–359.
- [81] **Scholkopf, B. and Smola, A.J.** (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, The MIT Press.
- [82] **Ponce, A.N., Behar, A.A., Hernandez, A.O. and Sitar, V.R.** (2004). Neural Networks for Self-tuning Control Systems, *Acta Polytechnica*, 44(1), 49–52.
- [83] **Flynn, D., McLoone, S., Irwin, G.W., Brown, M.D., Swidenbank, E. and Hogg, B.W.** (1997). Neural Control of Turbogenerator Systems, *Automatica*, 33(11), 1961–1973.
- [84] **Abdullah, R., Hussain, A.T. and Zayed, A.** (2005). A New RBF Neural Network Based Non-linear Self-tuning Pole-Zero Placement Controller, *International Conference on Artificial Neural Networks*, Warsaw, Poland.
- [85] **Wahyudi, S., Ahmad, W. and Htut, M.M.** (2009). Neural-tuned PID controller for Point-to-point (PTP) positioning system: Model reference approach, *International Colloquium on Signal Processing and Its Applications*, Kuala Lumpur, Malaysia.
- [86] **Guo, A. and Yang, J.** (2007). Self-tuning PID Control of Hydro-turbine Governor based on Genetic Neural Networks, *International Symposium on Intelligence Computation and Applications*, Wuhan, China.
- [87] **Kang, Y., Chu, M.H., Chang, C.W. and Chen, Y.W.** (2007). The Self-Tuning Neural Speed Regulator Applied to DC Servo Motor, *International Conference on Natural Computation*, Haikou, China.
- [88] **Pham, D.T. and Karaboğa, D.** (1999). Self-tuning fuzzy controller design using genetic optimisation and neural network modelling, *Artificial Intelligence in Engineering*, 13(2), 119–130.

- [89] **He, S.Z., Tan, S.H., Xu, F.L. and Wang, P.Z.** (1993). PID Self-tuning Control using a Fuzzy Adaptive Mechanism, *International Conference on Fuzzy Systems*, San Francisco, CA.
- [90] **Gautam, D. and Ha, C.** (2013). Control of a Quadrotor Using a Smart Self-Tuning Fuzzy PID Controller, *International Journal of Advanced Robotic Systems*, 10, 1–9.
- [91] **Ahn, K.K., Truong, D.Q., Thanh, T.Q. and Lee, B.R.** (2008). Online self-tuning fuzzy proportional–integral–derivative control for hydraulic load simulator, *Journal of Systems and Control Engineering*, 222(2), 81–95.
- [92] **Qiao, W.Z. and Mizumoto, M.** (1996). PID type fuzzy controller and parameters adaptive method, *Fuzzy Sets Systems*, 78(1), 23–35.
- [93] **Woo, Z.W., Chung, H.Y. and Lin, J.J.** (2000). A PID type fuzzy controller with self-tuning scaling factors, *Fuzzy Sets Systems*, 115(2), 321–326.
- [94] **Bandyopadhyay, R., Chakraborty, U.K. and Patranabis, D.** (2001). Autotuning a PID controller: A fuzzy-genetic approach, *Journal of Systems Architecture*, 47(7), 663–673.
- [95] **Sharkawy, A.B.** (2010). Genetic fuzzy self-tuning PID controllers for antilock braking systems, *Engineering Application of Artificial Intelligence*, 23(7), 1041–1052.
- [96] **Bouallegue, S., Haggege, J., Ayadi, M. and Benrejeb, M.** (2012). PID-type fuzzy logic controller tuning based on particle swarm optimization, *Engineering Application of Artificial Intelligence*, 25(3), 484–493.
- [97] **Li, C.S. and Priemer, R.** (1997). Self-learning General Purpose PID Controller, *Journal of the Franklin Institute*, 334(2), 167–189.
- [98] **Bishr, M., Yang, Y.G. and Lee, G.** (2000). Self-Tuning Pid Control Using an Adaptive Network–Based Fuzzy Inference System, *Intelligent Automation and Soft Computing*, 6(4), 271–280.
- [99] **Lu, C.H., Cheng, C.C., Liu, C.M. and Guo, J.Y.** (2012). Self-Tuning Predictive PID Controller Using Wavelet Type-2 Fuzzy Neural Networks, *International Conference on Fuzzy Theory and Its Applications*, Taichung, Taiwan.
- [100] **Jang, J.S.R.** (1993). ANFIS: Adaptive-Network-Based Fuzzy Inference System, *IEEE Transactions on Systems Man and Cybernetics*, 23(3), 665–685.
- [101] **Sung, S.W., Lee, J. and Beum, L.I.** (2009). *Process Identification and PID Control*, IEEE Press, John Wiley and Sons.
- [102] **Aström, K.J. and Hagglund, T.** (1995). *PID Controllers: Theory, Design and Tuning*, Instrument Society of America.
- [103] **Visioli, A.** (2006). *Practical PID Control*, Springer.



CURRICULUM VITAE



Name Surname: Kemal UÇAK

Place and Date of Birth: Ağlasun/ Burdur , 26.10.1983

Address: ITU, Ayazağa Campus, Faculty of Electrical-Electronics Engineering, Department of Control and Automation Engineering , Maslak/Istanbul

E-Mail: kemal.ucak@itu.edu.tr

B.Sc.: Pamukkale University, Department of Electrical-Electronics Engineering

M.Sc.: Istanbul Technical University, Department of Control and Automation Engineering

Professional Experience and Rewards:

- Research and Teaching Assistant, Muğla Sıtkı Koçman University, Department of Electrical-Electronics Engineering (2009-2009).
- Research and Teaching Assistant, Istanbul Technical University, Department of Control and Automation Engineering (2009-....).

List of Publications and Patents:

A) International Journal Papers

- **Uçak K.**, Günel Öke G., 2016: An adaptive support vector regressor controller for nonlinear systems, *Soft Computing*, 20(7), 2531-2556
- **Uçak K.**, Günel Öke G., 2016: A Novel Adaptive NARMA-L2 Controller based on Online Support Vector Regression for Nonlinear Systems, *Neural Processing Letters* (Accepted)
- **Uçak K.**, Günel Öke G., 2016: Generalized Self-Tuning Regulator Based on Online Support Vector Regression, *Neural Computing and Applications* (Accepted)

B) International Conference Papers

- **Uçak K.**, Günel Öke G., 2016: NARMA-L2 Controller Based on Online Support Vector Regression, *International Conference on Advanced Technology and Sciences (ICAT 2016)*, 1-3 September 2016, Konya, Turkey (Accepted).
- **Uçak K.**, Günel Öke G., 2015: Design of an Adaptive Support Vector Regressor Controller for a Spherical Tank System, *International Conference on Neural Information Processing (ICONIP 2015)*, 9-12 November 2015, Istanbul, Turkey.

- Karaman K., Bekaroğlu T. Y., Söylemez T. M., **Uçak K.**, Günel Öke G., 2015: Controlling 3- DOF Helicopter via Fuzzy PID Controller, *International Conference on Electrical and Electronics Engineering (ELECO 2015)*, 26-28 November 2015, Bursa, Turkey.
- **Uçak K.**, Caliskan, F., Oke G., 2013: Fault Diagnosis in a Nonlinear Three-Tank System via ANFIS, *International Conference on Electrical and Electronics Engineering (ELECO 2013)*, 28-30 November, 2013, Bursa, Turkey.
- Durmus S. M., **Uçak K.**, Soylemez T. M., Oke G., 2013: Train Speed Control in Moving-Block Railway Systems: An Online Adaptive PD Controller Desing, *IFAC Workshop on Advances in Control and Automation Theory for Transportation Applications (ACATTA 2013)*, 16-17 September, 2013, Istanbul, Turkey.
- **Uçak K.**, Oke G., 2013: RBF Neural Network Controller based on OLSSVR, *Asian Control Conference (ASCC 2013)*, 23-26 June, 2013, Istanbul, Turkey.
- **Uçak K.**, Oke G., 2013: RBF Neural Network Controller based on OLSSVR, *Asian Control Conference (ASCC 2013)*, 23-26 June, 2013, Istanbul, Turkey.
- Sariyildiz E., **Uçak K.**, Ohnishi K., Oke G., Temeltas H., 2013: Intelligent Systems Based Solutions for the Kinematics Problem of the Industrial Robot Arms, *Asian Control Conference (ASCC 2013)*, 23-26 June, 2013, Istanbul, Turkey.
- **Uçak K.**, Oke G., 2012: Adaptive Fuzzy PID Controller Based on Online LSSVR, *International Symposium on Innovations in Intelligent Systems and Applications (INISTA 2012)*, 2-4 July, 2012, Trabzon, Turkey.
- Sariyildiz E., **Uçak K.**, Oke G., Temeltas H., Ohnishi K., 2012: Support Vector Regression Based Inverse Kinematic Modeling for a 7-DOF Redundant Robot Arm, *International Symposium on Innovations in Intelligent Systems and Applications (INISTA 2012)*, 2-4 July, 2012, Trabzon, Turkey.
- **Uçak K.**, Oke G., 2011: An Improved Adaptive PID Controller Based on Online LSSVR with Multi RBF Kernel Tuning, *International Conference on Adaptive and Intelligent Systems (ICAIS 2011)*, 6-8 September, 2011, Klagenfurt, Austria.
- Sariyildiz E., **Uçak K.**, Oke G., Temeltas H., 2011: A Trajectory Tracking Application of Planar Robot Arm via Support Vector Machines, *International Conference on Adaptive and Intelligent Systems (ICAIS 2011)*, 6-8 September, 2011, Klagenfurt, Austria.
- **Uçak K.**, Oke G., 2011: Adaptive PID Controller Based on Online LSSVR with Kernel Tuning, *International Symposium on Innovations in Intelligent Systems and Applications (INISTA 2011)*, 15-18 June, 2011, Istanbul, Turkey.
- **Uçak K.**, Oke G., 2011: Modeling of Quadruple Tank System Using Support Vector Regression, *International Symposium on Innovations in Intelligent Systems and Applications (INISTA 2011)*, 15-18 June, 2011, Istanbul, Turkey.

C) National Conference Papers (in Turkish)

- **Uçak K.**, Günel Öke G., 2015: Uyarlamalı Destek Vektör Regressör Kontrolör, *Otomatik Kontrol Türk Milli Komitesi Ulusal Toplantısı (TOK 2015)*, 10-12 Eylül 2015, Denizli, Türkiye.

- Karaman K., Bekaroğlu T. Y., Söylemez T. M., **Uçak K.**, Günel Öke G., 2015: 3-eksenli Helikopter Kontrolü için Bulanık PID Kontrolör Tasarımı, *Otomatik Kontrol Türk Milli Komitesi Ulusal Toplantısı (TOK 2015)*, 10-12 Eylül 2015, Denizli, Türkiye.
- Sarıyıldız E., **Uçak K.**, Öke G., Temeltaş H., 2013: Endüstriyel Robot Kolu Dinamik Modelinin Destek Vektör Makinesi Kullanılarak Elde Edilmesi, *Otomatik Kontrol Türk Milli Komitesi Ulusal Toplantısı (TOK 2013)*, 26-28 Eylül 2013, Malatya, Türkiye.
- **Uçak K.**, Çalışkan F., Öke G., 2013: Bir Tank Sisteminde Uyarlamalı Sinirsel-Bulanık Çıkarım Sistemi Yardımıyla Arıza Teşhisi, *Otomatik Kontrol Türk Milli Komitesi Ulusal Toplantısı (TOK 2013)*, 26-28 Eylül 2013, Malatya, Türkiye.
- Sarıyıldız E., **Uçak K.**, Yalçın K. M., Öke G., Temeltaş H., 2012: 7-Serbestlik Dereceli PA-10 Robotunun Ters Kinematik Probleminin Destek Vektör Makinesi Kullanılarak Çözülmesi, *Otomatik Kontrol Türk Milli Komitesi Ulusal Toplantısı (TOK 2012)*, 11-13 Ekim 2012, Niğde, Türkiye.



