İSTANBUL TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

SERVO DENETİM SİSTEMLERİNİN İNCELENMESİ

YÜKSEK LİSANS TEZİ

Müh. Gökay Kadir HURMALI

Tezin Enstitüye Verildiği Tarih : 27 Ağustos 1992

Tezin Savunulduğu Tarih : 4 Eylül 1992

Tez Danışmanı : Y. Doç. Dr. Turgut Berat KARYOT

Diğer Jüri Üyeleri : Prof. Dr. Umur DAYBELGE

Doç. Dr. Fuat GÜRLEYEN

Eylül 1992

ÖNSÖZ

Bu tezin hazırlanmasında kıymetli fikirleri ve yapıcı uyarıları ile bana destek olan değerli hocam sayın Turgut Berat KARYOT'a ve her zaman desteklerini yanımda hissettiğim İ.T.Ü. Uçak ve Uzay Bilimleri Fakültesindeki değerli çalışma arkadaşlarıma en içten saygı ve teşekkürlerimi belirtmek isterim.

Gökay Kadir HURMALI
Eylül 1992, İstanbul

İÇİNDEKİLER

ÖZET

Zaman içinde değişen sistem parametreleri ve bozucu etkilere rağmen, verilen bir referans işaretini mümkün olan en az hata ile takip edecek denetim algoritmalarının tasarımı, uzun süredir araştırma konusu olmaya devam etmektedir. Bu çalışmada, bu algoritmalardan bazıları incelenmiş ve simülasyonlar yapılarak, performansları karşılaştırılmıştır.

Servo denetim yapısı olarak, ileri beslemeli iki serbestlik dereceli yapı alınmıştır. Bu yapı içinde geri besleme işaretini üretmek için orantı türev integral (PID), doğrusal ikincil Gauss dağılımlı (LQG) ve kendini ayarlayan (STR) denetim yasaları kullanılarak gerekli simülasyonlar yapılmış ve sonuçlar karşılaştırmalı olarak verilmiştir.

SUMMARY

ANALYSIS OF SERVO CONTROLLERS BY SIMULATION

Control algorithms capable of following a given reference signal, inspite of disturbances and system parameter variations are long beeing investigated. In this thesis, some of these algorithms are discussed and their performances are illustrated by the help of computer simulations.

Regulator type control systems try to keep the output and the states of the system fixed at a given set point. In the servo control problem, the objective is to make the states and outputs of the system to follow a desired trajectory. Practical systems often have specifications that involve both servo and regulation properties. This is traditionally solved by using a two degrees of freedom structure. Block diagram of two degrees of freedom structure is shown in Figure 1.

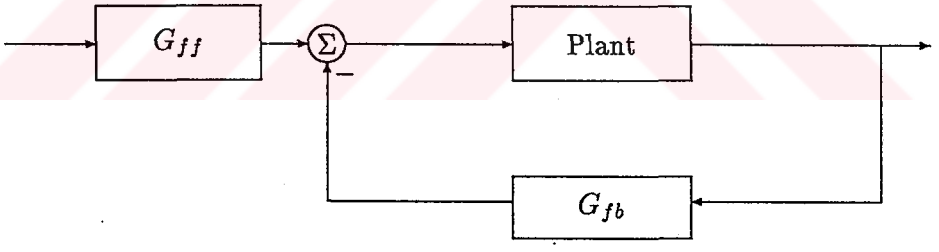


Figure 1 Block diagram of two degrees of freedom controller.

The feedback controller G_{fb} is first designed to obtain a closed loop system that is insensitive to load disturbances and to plant uncertainty. The feedforward compensator is then designed to obtain desired servo properties.

To calculate the feedforward signal, the one step ahead value of the reference is used. If the reference signal is not available, it have to be predicted using an appropriate model and a predictor. Let the closed loop systems transfer function, H(k), be given in discrete time as

$$\frac{y(k)}{u(k)} = H(k) = \frac{a_1 q + a_2}{q^2 + b_1 q + b_2} \tag{1}$$

Where, $y(k)$ is the measured system output, $u(k)$ is the control signal, and a_1, a_2, b_1 and b_2 are system parameters in discrete time. q is the forward shift operator in

discrete time with the following properties:

$$\begin{aligned} q^n f(kh) &= f(kh+nh) \\ q^n f(k) &= f(k+1) \end{aligned}$$

Then, we can rearrange (1) to give

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

In the servo problem, the objective is to make

$$y(k+1) = r(k+1) \tag{3}$$

where $r(k+1)$ is the reference signal at time $k+1$. If we substitute (3) in (1), for $u_m(k)$ we obtain

$$u_m(k) = \frac{1}{a_1}\left[r(k+1) + b_1 y(k) + b_2 y(k-1) - a_2 u(k-1)\right] \tag{4}$$

Here u_m is the feedforward compensator, and a_1, a_2, b_1 and b_2 are parameters of the closed loop system.

In this study we used 3 different types of regulators to obtain a robust and stable closed loop.

First of them is the PID regulator, described as

$$u_{fb}(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de}{dt}\right] \tag{5}$$

Here e is the difference between the reference signal and measured system output, $u_{fb}(t)$ is the feedback control signal, T_i is the integration time constant, T_d is the differentiation time constant and K is the gain of the controller.

Secondly, we used LQG control to obtain feedback signal. In LQG control, the system is assumed to be linear but it may be time varying. The objective is to choose the control signal in such a way that, a given quadratic loss function of control signal and system outputs is minimum. The obtained control rule is optimum if disturbance distributions are Gaussian random processes. The solution is given as follows:
Let the discrete time system be described in state space by

$$\begin{aligned} x(kh+h) &= \Phi x(kh) + \Gamma u(kh) + v(kh) \\ y(kh) &= Cx(kh) + e(kh) \end{aligned} \tag{6}$$

Here v and u are random variables with zero mean value and Gaussian distribution. Let's define

$$\begin{aligned} R_1 &= Variance\,(v(kh)v^T(kh)) \\ R_{12} &= Variance\,(v(kh)e(kh)) \\ R_2 &= Variance\,(e^2(kh)) \end{aligned}$$

Let's use Kalman filter, in (6), to estimate the one step ahead value of the system state. Kalman filter is then given by

$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + K(k)[y(k) - C\hat{x}(k)] \quad (7)$$

$$K(k) = \Phi P(k)C^T(R_2 + CP(k)C^T)^{-1} \quad (8)$$

$$P(k+1) = \Phi P(k)\Phi^T + R_1 - \Phi P(k)C^T(R_2 + CP(k)C^T)^{-1}CP(k)\Phi^T \quad (9)$$

Then there is a unique control strategy that minimizes the loss function given by (5.12). This control strategy is given by :

$$u(k) = -L(k)\hat{x}(k) \quad (10)$$

Here $\hat{x}$ is the estimated state, and L is the time varying gain matrix, given by (5.18).

The last regulator used to obtain feedback signal is the Self Tuning Regulator. STR is an adaptive control in the sense that it tunes its own parameters. A block diagram of STR is shown in Figure 2. For an STR to be realized, first, plant parameters have to be estimated using an appropriate estimation algorithm, like Least Squares Estimator or Kalman Filter. Secondly, a controller design procedure have to be given.

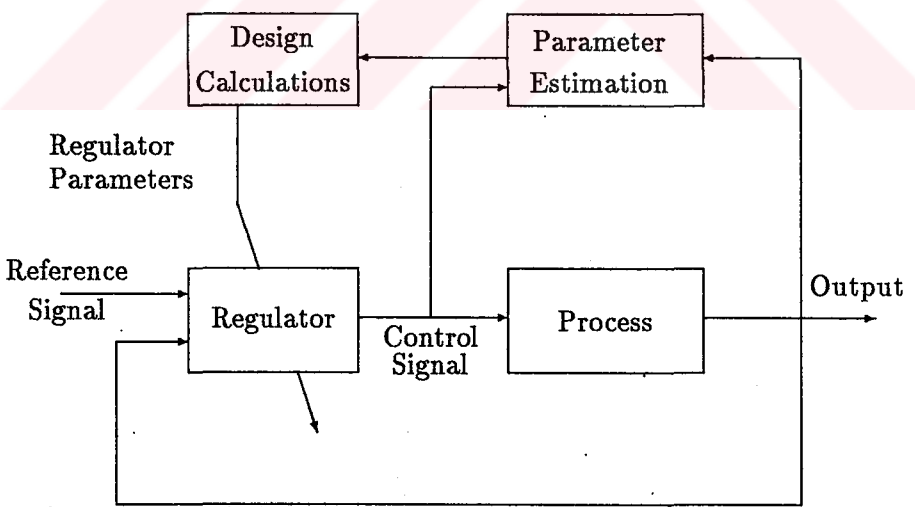


Figure 2 Block diagram of Self Tuning Regulator.

The organization of this thesis is as follows.

In Chapter 2, system and noise models are given. These models are used in the simulations. The equations of the harmonic oscillator and the double integrator are given as linear system models. Harmonic oscillator model will be used in the self tuning regulator problem. The given double integrator model, with time varying moment of inertia, will be the controlled system model in the simulations. The measurement noise is modelled as discretization noise.

Least squares parameter estimation method is given in Chapter 3. Here, the parameter estimation problem is formulated as an optimization problem, where the best model is the one that best fits the data according to a given criterion. For the least squares parameter estimation problem, the criterion is to minimize the discrete time loss function of :

$$J(\theta) = \sum_{k=1}^{N} g(\epsilon(k))$$

where, ϵ is the difference between the measured output and the computed output of the identification model. Solution of identification problem is given in Chapter 3.3. Then, this solution is rearranged in such a way that the results obtained for N observations could be used in order to get the estimates for $N+1$ observations. This procedure is referred as recursive least squares identifications. To illustrate the parameter estimation performance of the recursive least squares algorithm, two simulations are made. In the first simulation, parameters of a linear plant have been estimated and shown. In the second simulation, plant parameters are made to change with time. The simulation results of this time varying parameter estimation problem are given at the end of Chapter 3.

In Chapter 4, PID regulator equations are derived. Properties of discrete time PID regulator are discussed to some degree. Simulations are made with constant and time varying process models.

Linear quadratic optimal control problem is formulated in Chapter 5. In LQ problem, the process to be controlled is assumed to be linear, and a quadratic loss function is minimized by choosing appropriate control signals. The solution of the problem is obtained as a state feedback controller with time varying gain. The case with noise in the system is considered next. For the Gaussian random distributed noise, the problem is named LQG regulator problem, and the optimal solution that minimizes the quadratic loss function is given without proof. Simulation results indicate that LQ controller gives good results for deterministic cases, but its performance decreases for the noisy processes. LQG controller gave good performance for both deterministic case and noisy process.

In Chapter 6, self tuning regulator is discussed. A self tuner algorithm, with

pole placement design is given. System parameters are estimated by recursive least squares method, given in Chapter 3.

In Chapter 7, PID, LQG and STR regulators are used to generate feedback signals in a two degrees of freedom servo controller structure.

Simulation results of the servo controller with STR regulator have shown superior model following properties when compared to the servo controllers with LQG and PID feedback, but the control signal is remarked to be oscillating. Even though the produced control signals are different, using LQG and PID feedback in two degrees of freedom structure give similar reference following errors.

BÖLÜM 1

GİRİŞ

Sistem çıkışını belli bir değerde sabit tutmaya çalışan denetim sistemleri Regülatör olarak adlandırılırlar. Sistem çıkışını zaman içinde istenildiği gibi değiştiren denetleyiciler ise, Servo denetim sistemleri olarak isimlendirilirler.

Robot kolları gibi çok serbestlikli yapıların denetiminde, iki temel yaklaşım vardır. Birincisi, yapıya ait dinamik denklemlerin çözülerek denetim büyüklüklerinin üretilmesi, ikincisi ise, her serbestliğin, bağımsız bir servo denetim sistemi ile denetlenmesidir. Birinci yöntemin dezavantajları, gerçek zamandaki uygulamalar için hesap yükünün çok olması ve sistemin tam bir dinamik modeline ihtiyaç duymasıdır [1]. Bu tez çalışmasında, ikinci yöntem ayrıntılı olarak incelenecektir.

Servo denetim sistemlerinde, sistem parametrelerinin değişimine karşın performansın düşmemesi ve bozucu etkilerden bağımsız kalabilme istenen tasarım özellikleridir. Bu tip servo özelliklerin sağlanması için, genellikle, yüksek kazançlı denetleyiciler kullanılmıştır. Ancak, robot kolları gibi elastik mekanik sistemlerin denetiminde, mekanik salınıma sebep olmamak için, böylesi yüksek kazançlı denetleyiciler kullanılamaz. Optimal denetleyici, adaptif denetleyici veya optimal adaptif denetleyiciler tercih edilerek ve giriş işareti sınırlamalarına uyularak, istenilen performansa yaklaşmak bir çok durumda mümkündür.

Bu tez çalışmasında bazı servo sistemlerinin formülasyonu verilmiş ve performansları karşılaştırılmıştır. Değişik servo denetleyicilerin performanslarının karşılaştırılması için, bir sistem ve gürültü modeli oluşturulmuş ve konum hatasının ikincil (karesel) bir fonksiyonunun en küçük kılınması performans kriteri olarak alınmıştır. Değişik sistem ve gürültü modelleri kullanıldığında, çok farklı sonuçlar elde edilebilinir. Burada kullandığımız sistem modeli ile bir robot kolunun tek serbestliği simüle edilmiştir. Bu simülasyonu yaparken, robotun dinamik denklemlerini çözmek yerine, basitleştirilmiş, zamana göre değişen doğrusal bir sistem modeli kullanılmıştır. Gerçekte de, robotun cevabı bu modelden çok farklı olmayacaktır.

Bölüm 2 de denetlenecek sistemin özelliklerinin fiziksel tartışması yapılmış ve matematiksel modeli ortaya konmuştur. Simülasyonlarda kullanılacak olan referans girişleri, sistem girişindeki gürültü modeli, ölçüm hatasının modeli ve denetim işareti üzerindeki kısıtlamalar da bu bölümde verilmiştir.

3. bölümde, kendini ayarlayan denetim algoritmasında kullanılacak olan kestirici (estimator) en küçük kareler yöntemi ile formüle edilmiştir. Bu yöntem ile kestirilen parametreler, gerçek parametreler ile karşılaştırılmış ve sonuçlar tartışılmıştır.

Tezin 4. bölümünde orantı integral türev, PID (Proportional Integral Derivative), denetleyici denklemleri türetilmiştir.

5. bölümde ise, bir optimal denetim yöntemi olan doğrusal ikincil, LQ (Linear Quadratic), denetim ile ilgili tanımlar verilerek, LQ denetim problemi çözülmüştür.

6. bölümde kendini ayarlayan, STR (Self Tuning Regulator), denetim algoritması verilmiştir.

Bölüm 7'de iki serbestlik dereceli, TDOF (Two Degrees Of Freedom), ileri beslemeli yapı verilerek, STR, LQ ve PID denetim yöntemlerinin bu yapı içinde kullanılması tartışılmıştır.

Bölüm 3'te elde edilen kestiricinin ve Bölüm 4, 5, 6 ve 7'de elde edilen denetleyicilerin simülasyonları, her bölümün sonunda gösterilmiştir. Bu simülasyonlar Bölüm 2 de verilen sistem denklemi, gürültü modelleri ve denetim büyüklüğü üzerindeki sınırlamalar kullanılarak yapılmıştır.

Bölüm 8'de ise, verilen tüm denetim algoritmaları, simülasyon sonuçlarından da yararlanılarak karşılaştırılmış, zayıf ve kuvvetli yönleri ile tartışılmıştır.

BÖLÜM 2

SİSTEM DENKLEMİ, REFERANS İŞARETİ VE GÜRÜLTÜ MODELLERİ

2.1 Giriş

Bu bölümde, bilgisayar simülasyonlarında kullanılacak olan sistem modelleri verilmiştir.

Bölüm 2.2.1 de, denetlenecek tek girişli tek çıkışlı sistem denklemi ve bu denklemin ayrık zamandaki çözümü verilmiştir. Bölüm 2.2.2'de ikinci dereceden doğrusal sisteme ait ayrık zamandaki model sunulmuştur. Bu çalışmada kullanılacak olan optimal denetim algoritmaları ve parametre kestirici algoritmalar, doğrusal sistemler için geçerli olduklarından, bu algoritmaların içinde Bölüm 2.2.2'de verilen model kullanılmıştır. Ancak denetlenen sistem, Eşitlik 2.1'de verilen parametreleri zamana göre değişen sistem olarak belirlenmiştir.

Bölüm 2.3.1'de denetim işareti üzerindeki gürültünün fiziksel yapısı ve matematik modeli verilmiştir. Bölüm 2.3.2'de sayısal ölçüm sistemlerindeki gürültü tanımı ve modeli verilmiştir.

Bölüm 2.4'te, servo denetleyicilerin en az hata ile izlemesi istenen referans işareti verilmiştir.

2.2 Sistem Modeli

2.2.1 Sistem Modeli

Servo denetleyici algoritmalarının simülasyonlarında, tek girişli tek çıkışlı tork motoruna ait sistem denklemi kullanılmıştır.

$$J(t)\frac{d^2y}{dt^2} = u_t(t) + M_d(y) \tag{2.1}$$

Burada :

y : motorun çıkış açısı (rad),
J : motor şaftının ve motor şaftına bağlı yüklerin toplam eylemsizliği (kgm^2/rad),
u_t : motora uygulanan tork (Nm), ve
M_d : dışarıdan uygulanan bozucu torktur (Nm).

Simülasyonlarda $M_d = 0$ alınmıştır. $M_d \neq 0$ olduğu durumlarda bu torkun sistemin dinamik modelinden elde edilmesi veya Bölüm 3'te verilen en küçük kareler kestirim yöntemi veya Bölüm 5'te verilen Kalman süzgeci ve uygun bir model kullanılarak kestirilmesi gerekir. Böylece doğrusal denetim teorisi ve doğrusal sistem modeli kullanılarak elde edilen denetim işaretinden, kestirilen M_d değeri çıkarılarak, sistem sürülebilir.

Zamanla değişen doğrusal sistem denklemi

$$J(t)\frac{d^2y}{dt^2} = u_t(t) \tag{2.2}$$

şeklinde yazılabilir.

Çok serbestlikli yapılarda, $J(t)$ eylemsizliği, diğer serbestliklerin durumlarına bağlı bir fonksiyon olacaktır. Bu etkileri yansıtmak üzere $J(t)$ şu şekilde değiştirebilinir:

$$J(t) = \begin{cases} 0.500 + 0.133t & 0 \leq t < 3 \\ 1.000 & 3 \leq t < 7 \\ 2.167 - 0.133t & 7 \leq t \leq 10 \end{cases} \tag{2.3}$$

Şekil 2.1'de $J(t)$'nin değişimi gösterilmiştir.

u_t, denetim torkunun üst sınırını belirlemek için, $\frac{d^2y}{dt^2}$ ivmesinin, istenebilecek en büyük değerinin verilmesi gerekir.

$$\left|\frac{d^2y}{dt^2}\right| \leq 10rad/sn^2 \tag{2.4}$$

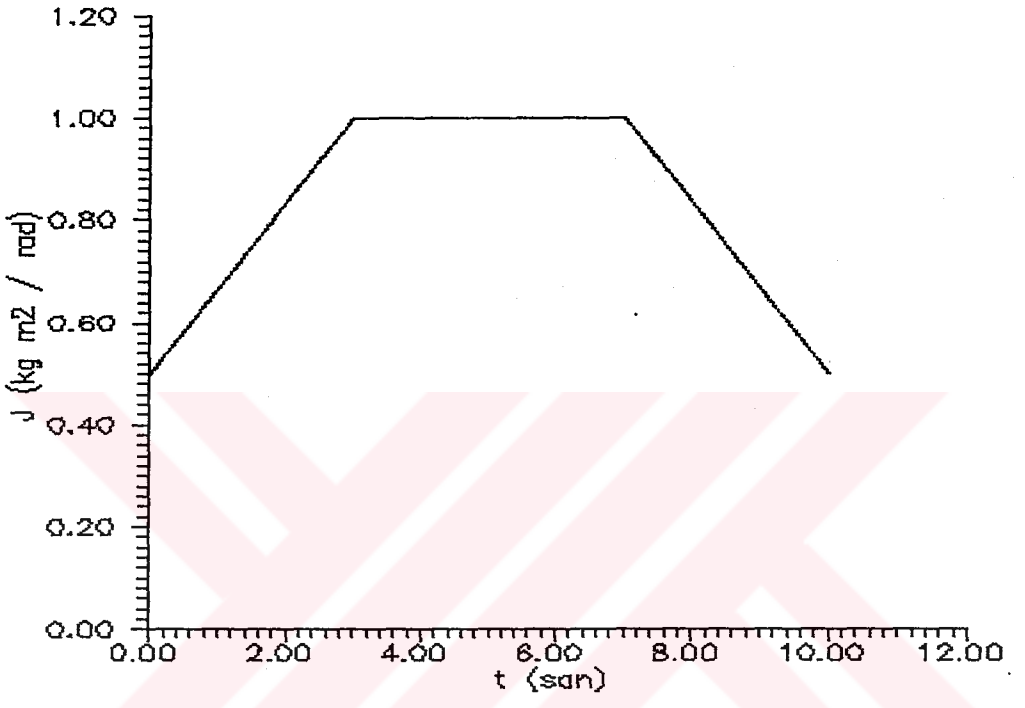
seçilirse, u_t

$$-10Nm \leq u_t \leq 10Nm$$

aralığında olduğunda, $J(t)$ eylemsizliğinin tüm değerleri için istenen ivme sağlanmış olur. Simülasyonlarda, denetim işareti

$$-20Nm \leq u_t \leq 20Nm \tag{2.5}$$

şeklinde sınırlandırılmıştır.



Şekil 2.1 $J(t)$'nin değişimi.

Burada dikkat edilmesi gereken nokta, denetim işareti bu şekilde sınırlandırıldıktan sonra, referans işaretinin ikinci türevinin $20 rad/san^2$'den küçük olması gerektiğidir. Referans işareti ise Eşitsizlik 2.4'deki şartı sağlayacak şekilde seçilmiştir.

$J = sabit$ için bu sistem çift entegratöre dönüşür. Çift entegratör denklemleri ve çözümleri Bölüm 2.2.2'de verilmiştir. Simülasyonlarda, örnekleme aralıklarında J eylemsizliğinin değişmediğini kabul edilmiştir. Böylece, Eşitlik 2.2 ile verilen, parametreleri zamanla değişen doğrusal sistemin çözümü için, $J(t)$ yerine $J(kh) = sabit$ konulabilir.

2.2.2 İkinci Derece Doğrusal Sistem Modelleri

Sönümlü Titreşim

Bölüm 3'te verilen en küçük kareler parametre kestirim algoritmasında, doğrusal sistem modeli olarak, sönümlü ikinci derece osilatör sistemi kullanılmıştır. Bu sistemin diferansiyel denklemi

$$\frac{d^2y}{dt^2} + 2\xi w_0 \frac{dy}{dt} + w_0^2 y = w_0^2 u \tag{2.6}$$

şeklindedir [2]. Burada w_0 doğal frekans ve ξ sönüm katsayısıdır.

Durum vektörünü

$$x(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix} \tag{2.7}$$

olarak tanımlarsak, durum uzayında sönümlü osilatör denklemleri,

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) = \begin{bmatrix} 0 & 1 \\ -w_0^2 & -2\xi w_0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ w_0^2 \end{bmatrix} u(t) \\ y(t) &= [1 \quad 0]\, x(t) \end{aligned} \tag{2.8}$$

olarak elde edilir. Bu sistemin ayrık durum uzayı denklemlerini hesaplamak için

$$\Psi = Ih + \frac{Ah^2}{2!} + \frac{A^2h^3}{3!} + \ldots + \frac{A^ih^{i+1}}{(i+1)!} + \ldots \tag{2.9}$$

tanımını yaparsak, Φ ve Γ

$$\begin{aligned} \Phi &= I + A\Psi \\ \Gamma &= \Psi B \end{aligned} \tag{2.10}$$

ile verilir. Ayrık durum denklemi

$$x(kh + h) = \Phi x(kh) + \Gamma u(kh) \tag{2.11}$$

şeklindedir.

Eşitlik 2.6'ya sıfır başlangıç koşullarıyla Laplace dönüşümü uygulandığı takdirde,

$$\frac{y(s)}{u(s)} = \frac{w_0^2}{s^2 + 2\xi w_0 s + w_0^2}$$

bulunur. Bu sistem ayrık zamanda

$$\frac{y(k)}{u(k)} = \frac{b_1 q + b_2}{q^2 + a_1 q + a_2} \tag{2.12}$$

$$
\begin{aligned}
b_1 &= 1-\alpha\left(\beta+\frac{\xi w_0}{w}\gamma\right)\\
b_2 &= \alpha^2+\alpha\left(\frac{\xi w_0}{w}\gamma-\beta\right)\\
a_1 &= -2\alpha\beta\\
a_2 &= \alpha^2\\
w &= w_0\sqrt{1-\xi^2}\\
\alpha &= e^{-\xi w_0 h}\\
\beta &= \cos(wh)\\
\gamma &= \sin(wh)
\end{aligned}
$$

şeklinde yazılır. Burada q ayrık zamanda ileri kaydırma işlemcisidir ve

$$
\begin{aligned}
q^n f(kh) &= f(kh+nh)\\
q^n f(k) &= f(k+1)
\end{aligned}
$$

şeklinde tanımlanır.

Çift Entegratör

Bilgisayar simülasyonlarında, ikinci derece doğrusal sistem olarak çift entegratör kullanılmıştır. Çift entegratör

$$J_e\frac{d^2y}{dt^2}=u(t) \tag{2.13}$$

diferansiyel denklemi ile tanımlanır. Laplace uzayında çift entegratör denklemi

$$\frac{Y(s)}{U(s)}=\frac{1}{s^2J_e} \tag{2.14}$$

dir. Eşitlik 2.7 ile verilen durum vektörü için durum uzayında çift entegratör denklemleri,

$$
\begin{aligned}
\dot{x}(t) &= \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}x(t)+\begin{bmatrix}0\\ 1/J_e\end{bmatrix}u(t)\\
y(t) &= [1 \quad 0]\,x(t)
\end{aligned} \tag{2.15}
$$

şeklinde yazılır. Eşitlik 2.13'deki sistem h örnekleme aralığı ile örneklenirse, ayrık zamanda durum denklemi

$$
\begin{aligned}
x(kh+h) &= \begin{bmatrix}1 & h\\ 0 & 1\end{bmatrix}x(kh)+\begin{bmatrix}h^2/(2J_e)\\ h/J_e\end{bmatrix}u(kh)\\
y(kh) &= [1 \quad 0]\,x(kh)
\end{aligned}
$$

olarak verilir. Ayrık zamandaki bu sistemin transfer denklemi

$$\frac{y(kh)}{u(kh)} = \frac{1}{J_e}\frac{h^2(q+1)}{2(q-1)^2} \tag{2.17}$$

dir.

2.3 Gürültü Modelleri

2.3.1 Denetim İşareti Üzerindeki Gürültü

Doğru akım motorlarının ürettiği tork, akım ile orantılıdır. Doğru akım motorlarında tork denetimi için orantı integral (PI) geri beslemesi uygulanır. Burada integral teriminden dolayı bir gecikme olmaktadır. Bu gecikmeden doğan hata, denetim işareti üzerinde bir gürültü olarak modellenebilir. Tipik bir doğru akım motorunda, birim basamak denetim işaretine tork cevabı, Şekil 2.2'te gösterilmiştir.

Deney sonuçlarına dayanarak, $h > 0.005$ san için, denetim işareti üzerindeki hata şu şekilde modellenebilir :

$$u_t(k) = u(k) - \frac{u(k) - u_t(k-1)}{h} 0.001g \tag{2.18}$$

Burada :

$u_t(k)$: k ile $k+1$ arasında motora etkiyen torkun ortalaması,

$u(k)$: k ile $k+1$ arasındaki denetim işareti,

h : örnekleme periyodu,

g : varyansı 0.3, ortalaması 0.5 olan beyaz gürültüdür.

2.3.2 Sayısal Ölçüm Sistemlerinde Gürültü Modelleri

Doğrusal veya açısal konum ölçen sayısal ölçüm aletlerindeki hata, ölçüm aletinin hassasiyetinin $\pm$ 0.5 biti ile sınırlıdır. Bir turda 50000 sayma yapabilen arttırmalı sayaç (incremental encoder) için, ölçülen açısal çıkış :

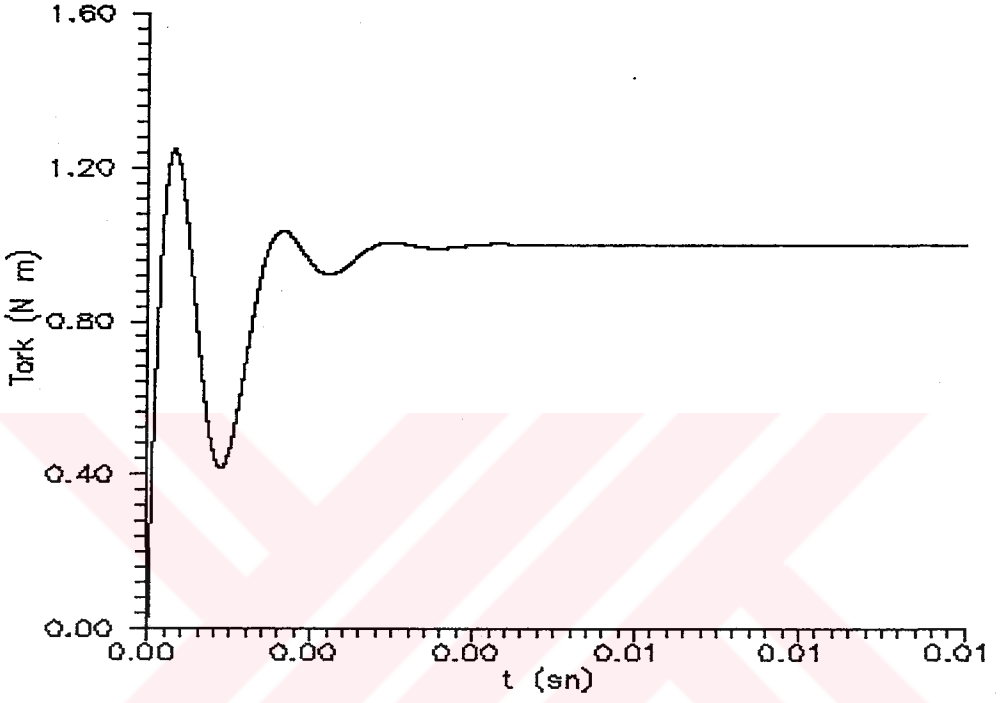
$$y_o = 2\pi \frac{tam(\frac{y}{2\pi}50000)}{50000}, \quad -\pi \leq y \leq \pi \tag{2.19}$$

olur. Burada :

y : radyan cinsinden açısal konum,

y_o : ölçülen açısal konum, ve

tam: argümanının sadece tam kısmını bildiren fonksiyondur.

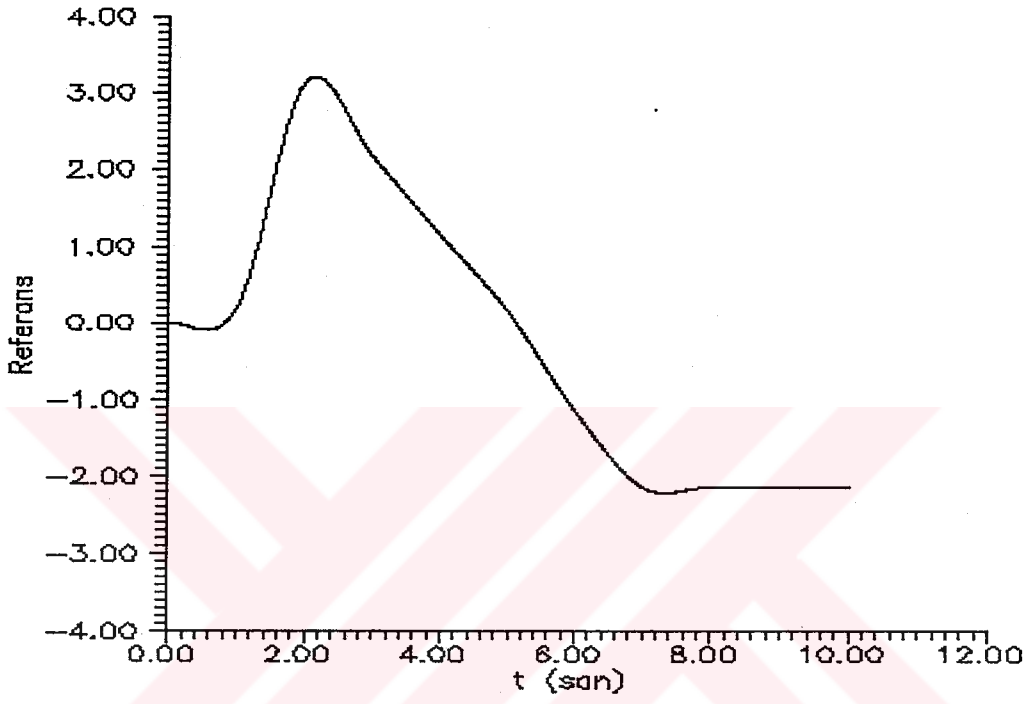


Şekil 2.2 PI geri beslemeli doğru akım motorunda birim basamak cevabı.

2.4 Referans İşareti

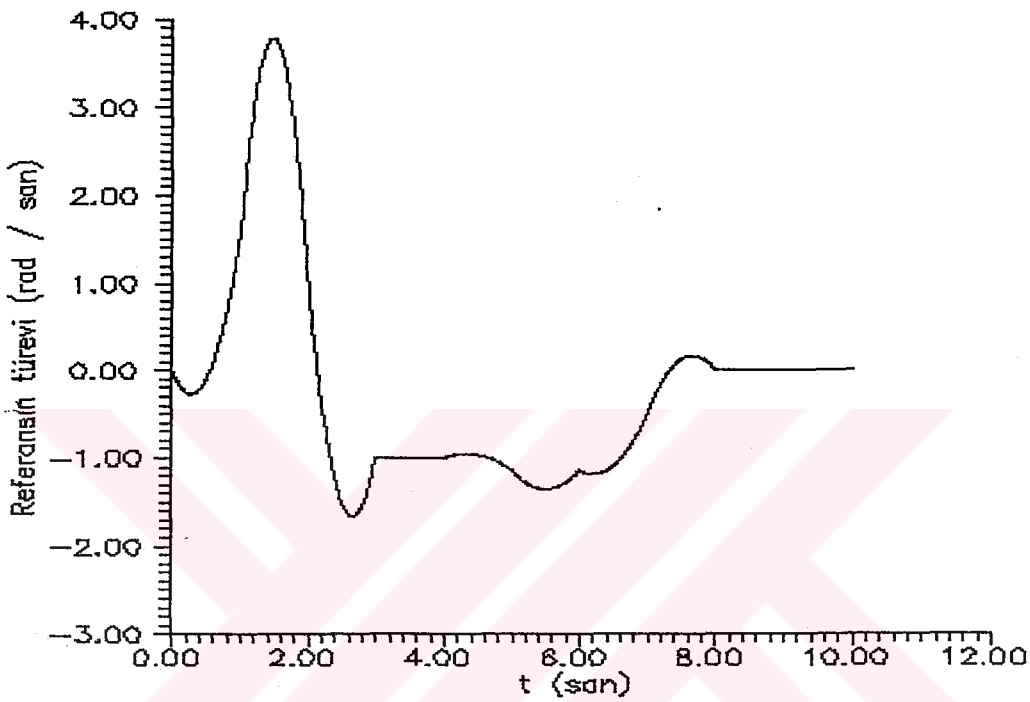
Servo denetim sistemlerinin mümkün olduğunca yakından takip etmeye çalışacakları referans işareti şu şekilde verilmiştir :

$$r(t) = \begin{cases} +1.17\,t^3 - 0.97\,t^2 + 0.00\,t + 0.00 & 0 \le t < 1 \\ -3.34\,t^3 + 14.73\,t^2 - 17.87\,t + 6.68 & 1 \le t < 2 \\ 1.97\,t^3 - 15.76\,t^2 + 40.37\,t - 30.32 & 2 \le t < 3 \\ -0.00\,t^3 + 0.00\,t^2 - 1.00\,t + 5.14 & 3 \le t < 4 \\ -0.14\,t^3 + 1.82\,t^2 - 8.84\,t + 16.34 & 4 \le t < 5 \\ 0.28\,t^3 - 4.62\,t^2 + 24.06\,t - 39.66 & 5 \le t < 6 \\ 0.36\,t^3 - 6.70\,t^2 + 40.38\,t - 79.98 & 6 \le t < 7 \\ -0.50\,t^3 + 11.50\,t^2 - 88.01\,t + 221.87 & 7 \le t < 8 \\ -0.00\,t^3 + 0.00\,t^2 - 0.00\,t - 2.14 & 8 \le t < 9 \\ 0.00\,t^3 + 0.00\,t^2 + 0.00\,t - 2.14 & 9 \le t \le 10 \end{cases} \tag{2.20}$$



Şekil 2.3 Referans işareti.

Verilen referans işaretinin kendisi ve birinci türevi, her t için süreklidir. Referans işaretinin ikinci türevi, $t = 1san$ anında $9.22rad/sn^2$ olarak en büyüktür ve Bölüm 2.2.1 de verilen sistem modelinde, izin verilen en büyük ivme olan $10rad/san^2$'den daha küçüktür. Eşitlik 2.20'de verilen referans işareti Şekil 2.3'de, türevi ise Şekil 2.4'de gösterilmiştir.



Şekil 2.4 Referans işaretinin türevi.

2.5 Sonuç

Bu bölümde bilgisayar simülasyonlarında kullanılan modeller verilmiştir.

Eşitlik 2.2 de verilen, parametreleri zamana göre değişen, doğrusal sistem modeli, simülasyonlarda denetlenecek sisteme ait olan modeldir. Bu modeli kurarken amaç, bir robot kolunun yer çekiminden ve diğer eksen hareketlerinden etkilenen bir serbestliğinin modellenmesidir. Yer çekimi nedeniyle meydana gelen bozucu tork hesaplarda gözönüne alınmamıştır. Bu torkun, kestirim algoritmaları veya sisteme ait dinamik denklemler kullanılarak hesaplanması ve etkisinin denetim işaretinden çıkarılması gerekir.

Denetim torku üzerindeki gürültü modeli kurulurken, pratik bilgilerden yararlanılmıştır. Günümüzde kullanılan geri beslemeli motor sürücülerinin hatası, gürültü

olarak modellenmiştir.

Bu bölümde verdiğimiz referans işareti, yüksek ivme, sabit hız ve sabit konum kısımlarından oluşmaktadır. Böylece denetleyicilerin bu üç durumdaki farkları izlenebilmiştir. Servo denetleyicilerin takip etmeye çalışacakları bu referans işaretinin kendisi ve birinci türevi süreklidir.

BÖLÜM 3

SİSTEM PARAMETRELERİNİN KESTİRİLMESİ

3.1 Giriş

Bilim ve mühendislikte matematik model gösterimi çok önemlidir. Matematiksel modeller, bir süreç hakkındaki bilgileri belirginleştirmek için çok kullanışlı bir araçtır. Süreç modelleri fiziğin temel ilkelerinden elde edilebilir. Ancak bu durumda, sistem davranışlarının, yüklerin ve bozucu etkilerin tam olarak bilinmesi gerekir. Doğrusal olmayan, çok girişli, çok çıkışlı bir sistemin modelinin tam olarak kurulması oldukça vakit alıcı bir işlemdir. Bu modelin gerçek zamanda çözülmesi ise, önemli bir hesap yüküdür [3].

Bu bölümde, Bölüm 3.2'de sistem parametrelerinin kestirilmesi ile ilgili temel kavramlar gözden geçirilecektir. Bölüm 3.3'te, en küçük kareler yöntemi açıklanacak ve en küçük kareler yönteminin sistem parametrelerinin kestirilmesinde nasıl kullanıldığı anlatılacaktır. Bölüm 3.4'te ise, rikörsif en küçük kareler yöntemi ile yapılan simülasyon sonuçları verilerek, en küçük kareler yönteminin sistem parametrelerinin değişimine cevabı ve gürültü hassasiyeti tartışılacaktır.

3.2 Tanımlar

Pratikte sistem parametrelerinin kestirilmesi iteratiftir.

Sistemin parametrelerinin kestirilebilmesi için giriş sinyalinin sistemin tüm kiplerini harekete geçirmesi gerekir. Geri beslemeli bir sistemde, giriş sinyali geri besleme yasalarından dolayı yeteri kadar zengin olmayabilir. Geri beslemeyi, doğrusal olmayan ve zamana göre değişir yapmak, veya giriş sinyaline beyaz gürültü eklemek, parametrelerin kestirilebilir olmasını sağlar.

Parametreleri belirlenecek olan ayrık zamandaki sistem denklemi

$$A(q)y(q) = B(q)u(k) + C(q)e(k) \tag{3.1}$$

olarak alınmıştır. Burada $u(k)$ giriş, $y(k)$ çıkış ve $e(k)$ ölçülemeyen beyaz gürültüdür.

Kriter

Bir belirleme problemi oluşturulurken, modelin deneysel verilerle ne kadar uyuştuğunun ölçüsü olan bir kriterin de ortaya konması gerekir. Ayrık zamandaki sistemler için kriter genellikle :

$$J(\theta) = \sum_{k=1}^{N} g(\epsilon(k))$$

olarak ifade edilir. Burada, ϵ giriş hatası, çıkış hatası ya da genelleştirilmiş hatadır. Kestirim hatası, genelleştirilmiş hatanın tipik bir örneğidir. g fonksiyonu genellikle ikincil olarak seçilir, ancak farklı biçimde olması da mümkündür.

Bir parametre bulma probleminin ilk formülasyonu, çözümü ve uygulanması, Gauss tarafından Ceres astroidinin yörüngesinin belirlenmesinde verilmiştir. Gauss, parametre bulma problemini bir en iyileme problemi olarak formüle etmiş ve en küçük kareler yöntemini (hatanın karelerinin toplamının minimizasyonuna dayanan bir yöntem) ortaya koymuştu. O zamandan beri, en küçük kareler yöntemi yaygın olarak kullanılmaktadır.

En küçük kareler yöntemi çok kolay ve anlaşılırdır. Bazı durumlarda yanlış ortalama değerli kestirimler üretir. Ancak bu, çeşitli eklemeler ile giderilebilir. En küçük kareler yöntemi bilinmeyenleri cinsinden doğrusal olan sistemlerde kullanılabilir.

Parametre kestirim probleminin çözümü için şunlar gereklidir:

- Sürecin giriş ve çıkış verileri,
- Bir model,
- Bir kriter.

Böylece parametre kestirimi, bir en iyileme problemi olarak formüle edilebilir. Burada, en iyi model, verilen bir kritere göre, verilere en iyi uyan modeldir.

3.3 En Küçük Kareler Yöntemi

Gauss'a göre modelin bilinmeyen parametreleri, gerçekte kestirilen ve hesaplanan çıkış değerleri arasındaki farkların karelerinin, ölçümün hassaslık derecesini ölçen katsayılarla çarpımlarının toplamını, en küçük yapacak şekilde seçilmelidir.

Genel Problem

Genel en küçük kareler yöntemi ile parametre belirleme probleminde, hesaplanan değer $\hat{y}$'nin

$$\hat{y} = \theta_1\varphi_1(x) + \theta_2\varphi_2(x) + \ldots + \theta_n\varphi_n(x) \tag{3.2}$$

modeli ile verildiği varsayılmıştır. Burada $\varphi_1, \varphi_2, \ldots, \varphi_n$ bilinen fonksiyonlar ve $\theta_1, \theta_2, \ldots, \theta_n$ bilinmeyen parametrelerdir. Bir deneyden $\{(x_i, y_i), i = 1, 2, ...N\}$ çiftleri elde edilir. Problem, Eşitlik 3.2'de verilen modelden hesaplanan $\hat{y}$ değişkenleri ve x_i deneysel değerlerinin, ölçülen y_i değişkenine mümkün olduğunca yakın olacak şekilde parametrelerin belirlenmesidir. Tüm ölçümlerin aynı hassasiyete sahip olduğu varsayılarak, en küçük kareler yöntemi, parametrelerin, kayıp fonksiyonu

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{N}\epsilon_i^2$$

$J(\theta)$'yı en küçük yapacak şekilde seçilmesi gerektiğini söyler. Burada,

$$\epsilon_i = y_i - \hat{y}_i = y_i - \theta_1\varphi_1(x_i) - \ldots - \theta_n\varphi_n(x_i), \quad i = 1, 2, \ldots, N$$

dir. Hesaplamaları basitleştirmek için aşağıdaki vektör gösterimi kullanılmıştır.

$$\varphi = [\varphi_1 \;\; \varphi_2 \cdots \varphi_n]^T$$

$$\theta = [\theta_1 \;\; \theta_2 \cdots \theta_n]^T$$

$$y = [y_1 \;\; y_2 \cdots y_N]^T$$

$$\epsilon = [\epsilon_1 \;\; \epsilon_2 \cdots \epsilon_n]^T$$

$$\Phi = \begin{bmatrix} \varphi^T(x_1) \\ \vdots \\ \varphi^T(x_N) \end{bmatrix}$$

Böylece, kayıp fonksiyonu $J(\theta)$

$$J(\theta) = \frac{1}{2}\epsilon^T\epsilon = \frac{1}{2}||\epsilon||^2 \tag{3.3}$$

şeklinde yazılabilir. Burada

$$\epsilon = y - \hat{y} \tag{3.4}$$

ve

$$\hat{y} = \Phi\theta \quad .$$

dır.

Problem, θ parametresinin, $||\epsilon||^2$'yi en küçük yapacak şekilde belirlenmesidir. En küçük kareler yönteminin çözümü şu şekildedir.

Eşitlik 3.3 ile verilen $J(\theta)$ fonksiyonu, θ parametreleri için

$$\Phi^T\Phi\hat{\theta} = \Phi^T y \tag{3.5}$$

şeklinde en küçüktür..

Eğer $\Phi^T\Phi$ tekil değil ise, en küçük tektir ve

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T y = \Phi^* y \tag{3.6}$$

ile verilir. Burada Φ^*, Φ'nin genelleştirilmiş tersi olarak adlandırılır.

Yukarıda verilen en küçük kareler probleminin çözümü şu şekilde ispatlanabilir:

Eşitlik 3.3'teki kayıp fonksiyonu

$$\begin{aligned} 2J(\theta) &= \epsilon^T\epsilon = [y - \Phi\theta]^T[y - \Phi\theta] \\ &= y^T y - y^T\Phi\theta - \theta^T\Phi^T y + \theta^T\Phi^T\Phi\theta \end{aligned} \tag{3.7}$$

şeklinde yazılabilir.

$\Phi^T\Phi$ matrisi her zaman sıfırdan farklı tanımlı olduğundan, $J(\theta)$ fonksiyonunun bir en küçüğü vardır.

Eşitlik 3.7'yi tekrar düzenlersek

$$\begin{aligned} 2J(\theta) &= y^T y - y^T\Phi\theta - \theta^T\Phi^T y + \theta^T\Phi^T\Phi\theta \\ &\qquad + y^T\Phi(\Phi^T\Phi)^{-1}\Phi^T y - y^T\Phi(\Phi^T\Phi)^{-1}\Phi^T y \\ &= (\theta - (\Phi^T\Phi)^{-1}\Phi^T y)(\Phi^T\Phi)(\theta - (\Phi^T\Phi)^{-1}\Phi^T y) \\ &\qquad - y^T\Phi(\Phi^T\Phi)^{-1}\Phi^T y + y^T y \end{aligned}$$

olur. Burada, en küçüğü bulunacak parametre vektörü

$$(\theta - (\Phi^T\Phi)^{-1}\Phi^T y)(\Phi^T\Phi)(\theta - (\Phi^T\Phi)^{-1}\Phi^T y) \tag{3.8}$$

teriminin içindedir.

$\Phi^T\Phi$ tekil değildir ve pozitiftir. Böylece 3.8 ile verilen terim her zaman sıfırdan büyüktür. Bu terim için en küçük sıfırdır. En küçük

$$\theta = \hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T y$$

için elde edilir. Böylece yukarıda verilen en küçük kareler probleminin çözümünün Eşitlik 3.6 ile belirlendiği gösterilmiş olur.

Sistem Parametrelerinin Kestirilmesi

En küçük kareler yöntemi, dinamik sistemlerin parametrelerinin belirlenmesinde kullanılabilir. Eşitlik 3.1'de tanımlanan sistemi göz önüne alalım. A polinomu n'inci, B polinomu ise $n-1$'inci dereceden olsun.

$$y(k) = \frac{b_1 q^{n-1} + b_2 q^{n-2} + \ldots + b_n}{q^n + a_1 q^{n-1} + a_2 q^{n-2} + \ldots + a_n} u(k) + C(q)e(k)$$

Sisteme $\{u(1), u(2), ..., u(N)\}$ girişlerinin uygulandığını ve bunlara karşılık gelen $\{y(1), y(2), ..., y(N)\}$ çıkışlarının ölçüldüğünü varsayalım. Böylece bilinmeyen parametreler

$$\theta = [a_1 \ldots a_n \; b_1 \ldots b_n] \tag{3.9}$$

olacaktır.

$$\varphi^T(k+1) = [-y(k) \ldots - y(k-n+1) \;\; | \;\; u(k) \ldots u(k-n+1)] \tag{3.10}$$

$$\Phi = \begin{bmatrix} \varphi^T(n+1) \\ \vdots \\ \varphi^T(N) \end{bmatrix}$$

tanımını yapalım.

En küçük kareler kestirim yöntemi, eğer $\Phi^T\Phi$ tekil değil ise, Eşitlik 3.6 ile verilmiştir. $\Phi^T\Phi$'ın tekil olmaması için ise girişin yeteri kadar zengin olması gerekmektedir, yani giriş işareti sistemin tüm kiplerini harekete geçirmelidir.

Eğer, ölçülemeyen gürültü $C(q)e(k)$'nin ortalaması sıfır değil ise, kestirilen parametreler, gerçek parametrelerden farklı olacaktır. Bu fark $C(q)e(k)$ 'nin ortalaması ile orantılıdır.

Rikörsif Hesaplamalar

Eğer N adet ölçüm için, en küçük kareler yöntemi ile problem çözüldü ise, yeni bir ölçüm yapıldığında, tüm problemi baştan çözmek bilgisayar zamanının israfı olarak görünmektedir. Bu bölümde amacımız, N ölçüm için elde edilen sonuçları, $N+1$ ölçüm ile yapılacak kestirimlerin elde edilmesinde kullanılabilecek şekilde düzenlemektir. Eşitlik 3.6 da verilen en küçük kareler yöntemi, rikörsif denklemler verecek şekilde tekrar düzenlenebilir. $\hat{\theta}$, N ölçümden elde edilen en küçük kareler kestirimini göstersin,

$$\Phi(N) = \begin{bmatrix} \varphi^T(x_1) \\ \vdots \\ \varphi^T(x_N) \end{bmatrix}, \quad y(N) = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

olduğuna göre, her N için, $\Phi^T\Phi$ matrisinin tekil olmadığı varsayılır ise, $\hat{\theta}$, Eşitlik 3.6 ile

$$\hat{\theta} = [\Phi^T(N)\Phi(N)]^{-1}\Phi^T(N)y(N)$$

şeklinde verilir. Yeni bir ölçüm yapıldığında Φ matrisine yeni bir sütun ve y vektörüne yeni bir eleman eklenir. Böylece

$$\Phi(N+1) = \begin{bmatrix} \Phi(N) \\ \varphi^T(N+1) \end{bmatrix}, \quad y(N+1) = \begin{bmatrix} y(N) \\ y_{N+1} \end{bmatrix} \tag{3.11}$$

şeklinde gelişir. $\hat{\theta}(N+1)$ ise :

$$\begin{aligned} \hat{\theta}(N+1) &= [\Phi^T(N+1)\Phi(N+1)]^{-1}\Phi(N+1)y(N+1) \\ &= [\Phi^T(N)\Phi(N) + \varphi(N+1)\varphi^T(N+1)]^{-1} \\ &\qquad x\ [\Phi^T(N)y(N) + \varphi(N+1)y_{N+1}] \end{aligned} \tag{3.12}$$

şeklinde yazılır. $\Phi^T\Phi$ matrisinin sıfırdan büyük tanımlı olduğu kabul edilirse, en küçük kareler kestirimi aşağıdaki rikörsif denklemi sağlar:

$$\hat{\theta}(N+1) = \hat{\theta}(N) + K(N)[y_{N+1} - \varphi^T(N+1)\hat{\theta}(N)] \tag{3.13}$$

Burada $K(N)$ şu şekilde tanımlanmıştır:

$$\begin{aligned} K(N) &= P(N+1)\varphi(N+1) \\ &= P(N)\varphi(N+1)[1 + \varphi^T(N+1)P(N)\varphi(N+1)]^{-1} \end{aligned} \tag{3.14}$$

$P(N+1)$ ise

$$P(N+1) = [I - K(N)\varphi^T(N+1)]P(N) \tag{3.15}$$

dır.

Bu çözümün ispatı için, ilk önce matris tersi Lemma'sını verelim:

A, C ve $C^{-1} + DA^{-1}B$ tekil olmayan matrisler olsunlar.

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B[C^{-1} + DA^{-1}B]^{-1}DA^{-1}$$

dir.

Matris tersi lemması $[A + BCD]$'yi tersi ile çarparak ispatlanabilinir.

$$\begin{aligned}
& [A + BCD]\{A^{-1} - A^{-1}B[C^{-1} + DA^{-1}B]^{-1}DA^{-1}\} \\
= \; & I + BCDA^{-1} - B[C^{-1} + DA^{-1}B]^{-1}DA^{-1} \\
& \qquad - BCDA^{-1}B[C^{-1} + DA^{-1}B]^{-1}DA^{-1} \\
= \; & I + BCDA^{-1} - BC[C^{-1} - DA^{-1}B][C^{-1} + DA^{-1}B]^{-1}DA^{-1} \\
= \; & I + BCDA^{-1} - BCDA^{-1} \\
= \; & I
\end{aligned}$$

Rikörsif en küçük kareler yöntemi ile parametre kestirme probleminin çözümünün ispatında, yazımı basitleştirmek için $\Phi(N)$ yerine Φ, $y(N)$ yerine y ve $\varphi^T(N+1)$ yerine φ^T yazalım. Bu durumda Eşitlik 3.12 şu şekilde yazılabilir.

$$\begin{aligned}
\hat{\theta}(N+1) &= [\Phi^T\Phi + \varphi\varphi^T]^{-1}[\Phi^T y + \varphi y_{N+1}] \\
&= (\Phi^T\Phi)^{-1}\Phi^T y + [(\Phi^T\Phi + \varphi\varphi^T)^{-1} - (\Phi^T\Phi)^{-1}]\Phi^T y \\
&\qquad + (\Phi^T\Phi + \varphi\varphi^T)^{-1}\varphi y_{N+1}
\end{aligned} \tag{3.16}$$

Burada

$$\hat{\theta}(N) = (\Phi^T\Phi)^{-1}\Phi^T y$$

ve

$$\begin{aligned}
[(\Phi^T\Phi + \varphi\varphi^T)^{-1} - \; & (\Phi^T\Phi)^{-1}]\Phi^T y \\
&= (\Phi^T\Phi + \varphi\varphi^T)^{-1}(\Phi^T\Phi - \Phi^T\Phi - \varphi\varphi^T)(\Phi^T\Phi)^{-1}\Phi^T y \\
&= -(\Phi^T\Phi + \varphi\varphi^T)^{-1}\varphi\varphi^T(\Phi^T\Phi)^{-1}\Phi^T y \\
&= -(\Phi^T\Phi + \varphi\varphi^T)^{-1}\varphi\varphi^T\theta
\end{aligned}$$

olduğu göz önüne alınarak, Eşitlik 3.16

$$\hat{\theta}(N+1) = \hat{\theta}(N) + K(N)\,[y_{N+1} - \varphi^T(N+1)]^{-1}\varphi(N+1)$$

şeklinde yazılır. Burada:

$$\begin{aligned} K(N) &= [\Phi^T(N)\Phi(N) + \varphi(N+1)\varphi^T(N+1)]^{-1}\varphi(N+1) \\ &= [\Phi^T(N+1)\Phi(N+1)]^{-1}\varphi(N+1) \end{aligned} \tag{3.17}$$

dir.

Ağırlık çarpanı $K(N)$ için rikörsif bir bağıntı elde etmek üzere, P büyüklüğünü

$$P(N) = [\Phi^T(N)\Phi(N)]^{-1} \tag{3.18}$$

şeklinde tanımlayalım. P kestirimlerin varyansı ile orantılı olacaktır. Matris tersi Lemma'sını $P(N+1)$'e uygularsak:

$$\begin{aligned} P(N+1) &= [\Phi^T(N+1)\Phi(N+1)]^{-1} = [\Phi^T\Phi + \varphi\varphi^T]^{-1} \\ &= (\Phi^T\Phi)^{-1} - (\Phi^T\Phi)^{-1}\varphi[I + \varphi^T(\Phi^T\Phi)^{-1}\varphi]^{-1}\varphi^T(\Phi^T\Phi)^{-1} \end{aligned}$$

olur. Böylece:

$$\begin{aligned} P(N+1) &= P(N) - P(N)\varphi(N+1) \\ &\quad *[I + \varphi^T(N+1)P(N)\varphi(N+1)]^{-1}\varphi^T(N+1)P(N) \end{aligned} \tag{3.19}$$

şeklinde elde edilir. Eşitlik 3.17 ve Eşitlik 3.18'i birleştirirsek

$$K(N) = P(N+1)\varphi(N+1)$$

elde ederiz. Burada $P(N+1)$ yerine Eşitlik 3.19'dan eşdeğerini koyarsak, basit hesaplamalar sonucu

$$K(N) = P(N)\varphi(N+1)[I + \varphi^T(N+1)P(N)\varphi(N+1)]^{-1}$$

elde edilir.

Burada P'yi hesaplamak için matris tersi almak gereklidir. Ancak, tersi alınacak matris ölçümlerle aynı boyutludur; yani, tek girişli, tek çıkışlı bir sistem için skalardır.

Eşitlik 3.13 başlangıç şartlarından etkilenir. $\hat{\theta}(N+1)$ kestirimi ise, bir önceki $\hat{\theta}(N)$ kestirimine bir düzeltme eklenerek bulunur. Düzeltme $y_{N+1} - \varphi^T(N+1)\hat{\theta}(N)$

ile orantılıdır. Burada, $\varphi^T(N+1)\hat{\theta}(N)$ terimine, Eşitlik 3.2'de verilen model ile kestirilen $N+1$ zamanındaki y değeri olarak bakılabilir. Böylece düzeltme terimi, daha önceki ölçümler kullanılarak kestirilen ve ölçülen y_{N+1} arasındaki fark ile orantılıdır. $K(N)$ vektörünün bileşenleri, düzeltmenin ve daha önceki kestirimin nasıl birleştirileceğini söyleyen ağırlık çarpanlarıdırlar. $K_i(N)$ bileşeni, $\varphi_i^T(N+1)$ ile orantılıdır.

$P(N)$ matrisi, sadece $\Phi^T\Phi$ tekil olmadığı zaman tanımlıdır.

$$\Phi^T(N)\Phi(N) = \sum_{k=1}^{N} \varphi(k)\varphi^T(k)$$

olduğundan, eğer N yeteri kadar küçük ise $\Phi^T\Phi$'nin her zaman tekil olacağı anlaşılır. P için bir başlangıç şartı elde etmek için, $\Phi^T\Phi$'nin tekil olmayacağı bir $N = N_0$ seçerek

$$\begin{aligned} P(N_0) &= [\Phi^T(N_0)\Phi(N_0)]^{-1} \\ \hat{\theta}(N_0) &= P(N_0)\Phi^T(N_0)y(N_0) \end{aligned}$$

hesabının yapılması gerekir. Böylece rikörsif denklemler $N > N_0$ için kullanılabilir. Ancak, rikörsif denklemlerin her adımda kullanılması istenir. Eğer rikörsif denklemler

$$P(0) = P_0$$

başlangıç şartı ile başlarsa

$$P(N) = [P_0^{-1} + \Phi^T(N)\Phi(N)]^{-1}$$

olur. Son yazılan eşitlikte P_0 yeteri kadar büyük seçilerek $P(N)$, $[\Phi^T(N)\Phi(N)]^{-1}$'e istenildiği kadar yakın yapılabilir.

Zamanla Değişen Sistemler

Eşitlik 3.3'teki kayıp fonksiyonu kullanıldığı zaman, tüm verilere eşit ağırlık verilmiş olur. Eğer parametreler zamanla değişiyorsa, eski verilerin etkilerinin azaltılması gerekir. Bu, üstel ağırlıklı bir kayıp fonksiyonu kullanılarak sağlanabilir.

$$J(\theta) = \sum_{k=1}^{N} \lambda^{N-k}[y(k) - \varphi^T(k)\theta]^2, \quad 0 < \lambda < 1 \tag{3.20}$$

Unutma çarpanı λ, birden küçüktür ve eski verilerin ne kadar çabuk unutulduğunun bir ölçüsüdür.

Eşitlik 3.20 deki kayıp fonksiyonu kullanılarak elde edilen en küçük kareler kestirimi

$$\begin{aligned} \hat{\theta}(k+1) &= \hat{\theta}(k) + K(k)[y_{k+1} - \varphi^T(k+1)\hat{\theta}(k)] \\ K(k) &= P(k)\varphi(k+1)[\lambda + \varphi^T(k+1)P(k)\varphi(k+1)]^{-1} \\ P(k+1) &= [I - K(k)\varphi^T(k+1)]P(k) \;/\; \lambda \end{aligned} \tag{3.21}$$

şeklindedir.

3.4 Simülasyonlar

Simülasyon 3.1

Eşitlik 2.6 ile verilen ikinci derece doğrusal sistemde

$$w_0 = 5rad/san., \quad \xi = 0.7 \;\; ve \;\; h = 0.01$$

için, ayrık zaman parametreleri Eşitlik 2.7'den.

$$a_1 = -1.986, \quad a_2 = 0.933, \quad b_1 = 0.00122, \quad b_2 = 0.00119$$

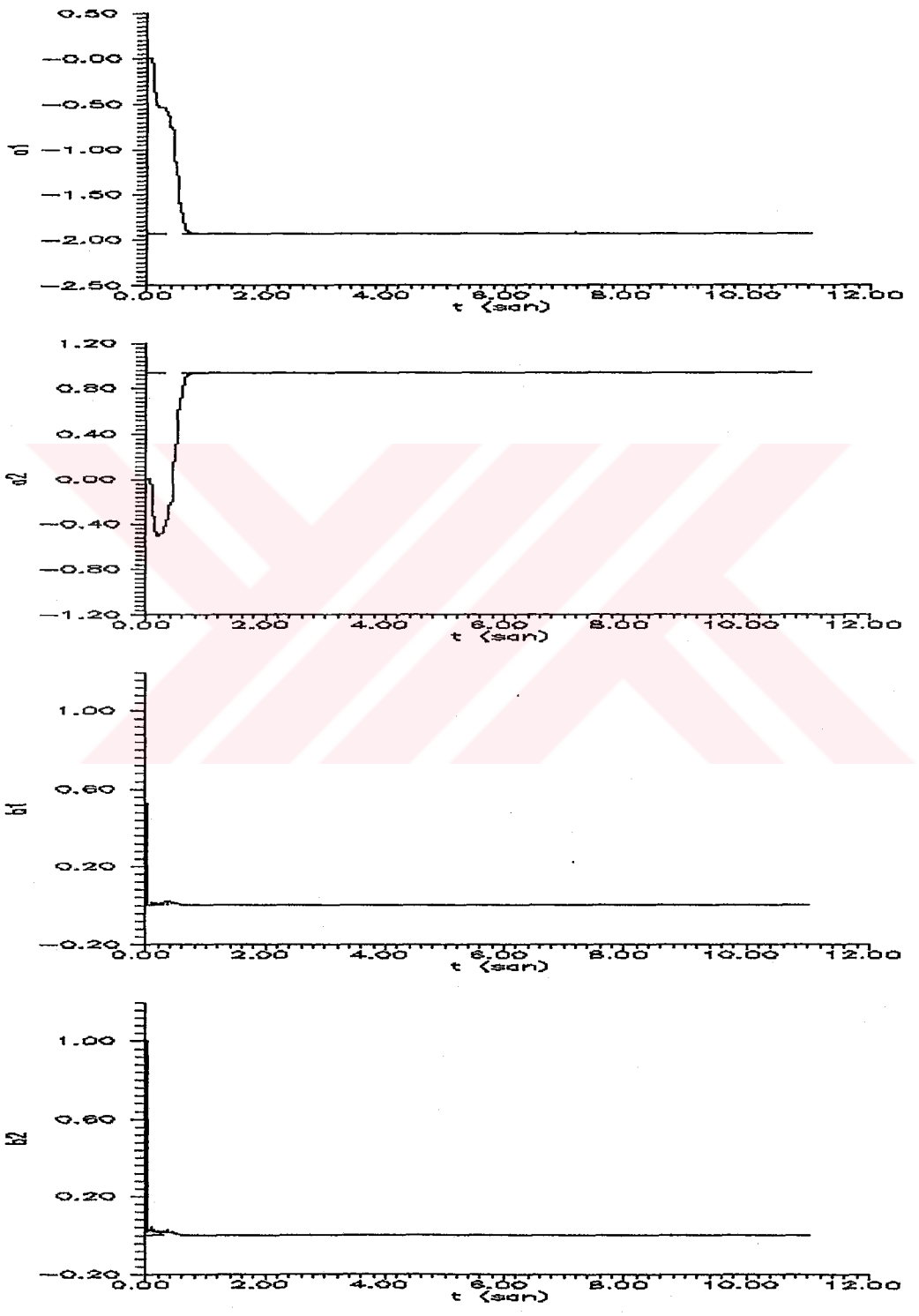
şeklinde bulunur.

Sisteme, sıfır başlangıç durumunda birim basamak + %15 gürültü uygulanmıştır. Ayrık zamandaki a_1, a_2, b_1 ve b_2 parametrelerinin kestirimleri Şekil 3.1'de gösterilmiştir. Kesikli çizgiler gerçek parametreleri, sürekli çizgiler ise Eşitlik 3.21 ile verilen rikörsif en küçük kareler yöntemi ile bulunan parametre kestirimlerini göstermektedir. Eşitlik 3.21'de

$$P(0) = 10I, \quad \lambda = 0.9, \quad \hat{\theta}(0) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

alınmıştır.

Bu durumda, kestirimlerin 50 zaman adımı süren bir geçiş döneminden sonra gerçek değerlere yeterli derecede yaklaştığı görülmektedir.



Şekil 3.1 En küçük kareler yöntemi ile parametre kestirimi, w_0=sabit.

Simülasyon 3.2

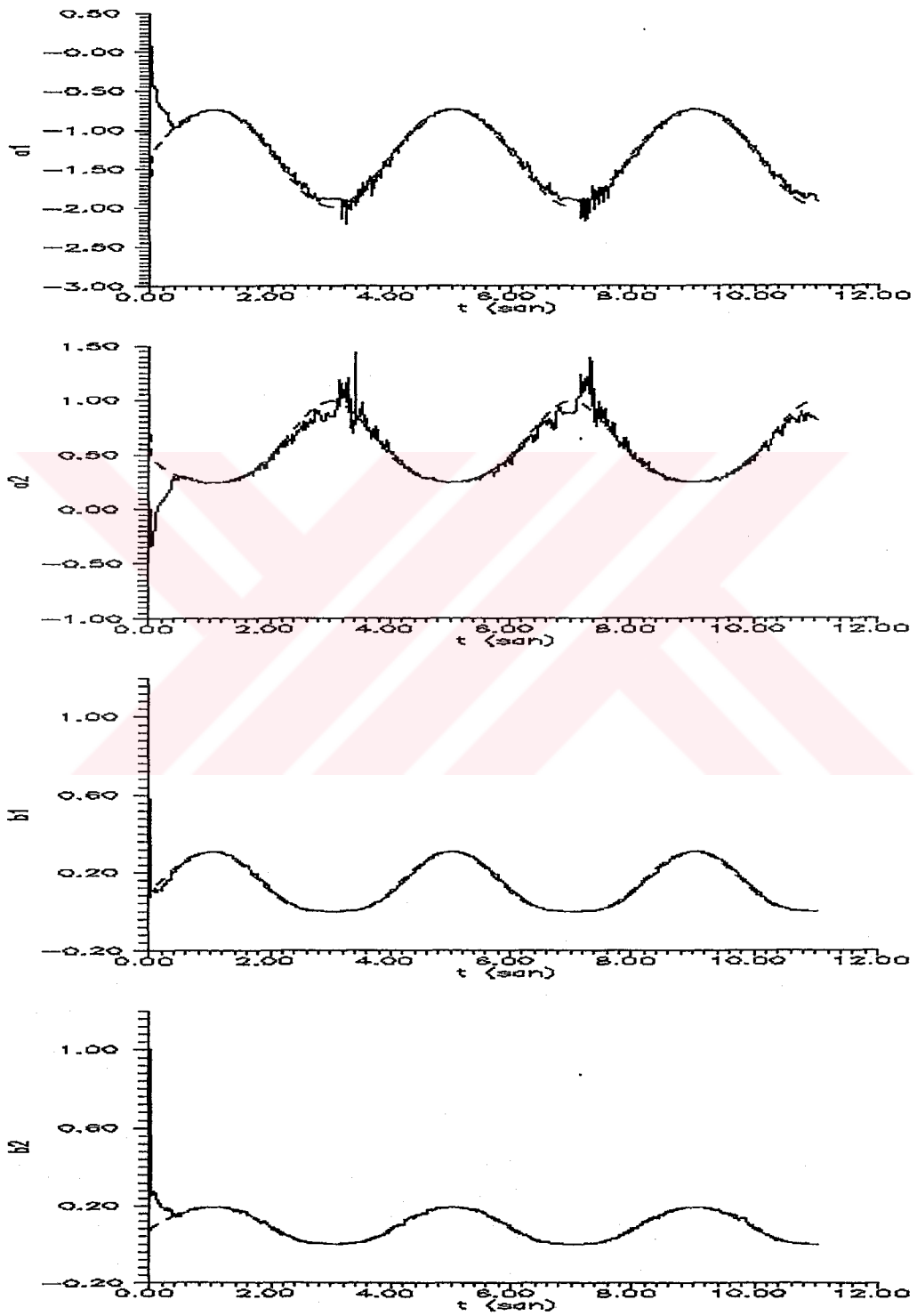
Simülasyon 1 şartları, doğal frekans, zamana göre

$$w_0 = 51 + 50\sin(0.5\pi t)$$

şeklinde değiştirilerek tekrarlanmıştır. Ayrık zamandaki a_1, a_2, b_1 ve b_2 parametrelerinin kestirimleri, Şekil 3.2'de gösterilmiştir. Kesikli çizgiler gerçek parametreleri, sürekli çizgiler ise Eşitlik 3.21 ile verilen rikörsif en küçük kareler elde edilerek, bulunan parametre kestirimlerini göstermektedir.

Bu durumda, sistemin doğal frekansı 2 saniye içinde $1rad/san$'den $101rad/san$'ye çıkmaktadır.

Simülasyon sonuçlarında, $t = 3san$ ve $t = 7san$ civarında kestirim hatasının arttığı görülmektedir. Bunun nedeni, simülasyonu yapılan sistemin doğal frekansı w_0'ın bu zamanlarda çok fazla büyümesidir ($w_0 > 90rad/san$). Büyük doğal frekanslı sistem, ağır olduğu ve zor hareket ettiği için, %15'lik gürültü sistemin tüm kiplerini harekete geçirememektedir. Bu sebeple de, en küçük kareler kestiriminin yaptığı hata artmaktadır.



Şekil 3.2 En küçük kareler yöntemi ile değişen parametre kestirimi, $w_0 = f(t)$.

3.5 Sonuç

Eşitlik 3.21'de en küçük kareler yöntemi ile parametre kestirim probleminin çözümü verilmiştir. Burada $P(0), \theta(0)$ başlangıç koşulları ve λ unutma faktörünün, belirlenmiş olması gerekmektedir.

λ'nın küçük seçilmesi, eski verilerin etkisinin az olması anlamına gelmektedir. Bu, parametreleri değişen bir sistem için istenen bir durumdur. Ancak, λ küçük olduğunda, parametre kestirimi gürültülere hassas olmaktadır. λ'nın değeri belirlenirken, sistemdeki gürültü genliği ve sistem parametrelerinin beklenen değişme hızları dikkate alınarak karar verilmelidir.

Yapılan tüm simülasyonlarda, parametre kestirimleri bir geçiş zamanından sonra gerçek parametrelere yakınsamıştır. Bölüm 5'de verilen LQ ve STR denetim algoritmaları, sistem parametrelerini kullandığından, geçiş zamanı süresince yanlış denetim sinyali üretilmektedir. Bu problemi çözmek üzere, N_0 adım için sistem parametrelerini kullanmayan denetim algoritmaları (örneğin: PID denetim) kullanılarak, veriler toplandıktan sonra, (19.5) ve (19.6)'dan $P(N_0)$ ve $\theta(N_0)$ başlangıç koşulları elde edilerek, rikörsif parametre kestirimi ve optimal denetim algoritmaları başlatılabilir.

BÖLÜM 4

ORANTI İNTEGRAL TÜREV DENETİMİ

4.1 Giriş

Bir çok denetim problemi Orantı İntegral Türev (PID) denetleyiciler kullanılarak çözülebilir. PID denetleyicinin yapısı son derece basit ve anlaşılırdır. PID denetleyicisinin ilk uygulamaları analog olarak gerçekleştirilmiştir. Sayısal PID denetleyicisinin donanımı daha ucuz ve daha güvenilir olduğu için, günümüzde, hemen tüm PID denetleyicileri sayısal olarak yapılmaktadır. En basit denetleyici olmasına karşın, PID denetleyicilerinin, ayrıklaştırılırken dikkat edilmesi gereken bazı özellikleri vardır.

Bu bölümde ayrık zamanda PID denetleyicinin denklemleri verilmiş ve bilgisayar simülasyonları yapılmıştır.

4.2 PID Denetleyici Denklemleri

En basit halde PID denetleyicisi şu şekilde tanımlanır :

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de}{dt}\right] \tag{4.1}$$

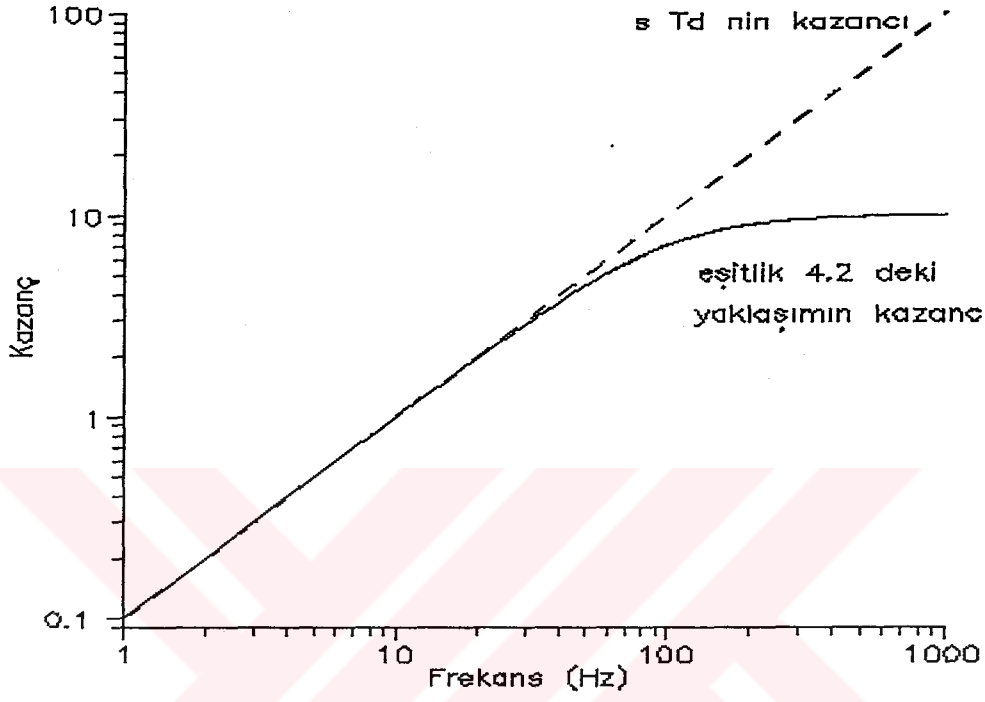
Burada $e(t)$ hata değeri, $e(t) = r(t) - y(t)$ şeklinde tanımlanmıştır ve referans işareti $r(t)$ ile süreç çıkışı $y(t)$ arasındaki farktır [4]. K denetleyici kazancı, T_i integral zaman sabiti ve T_d ise türev zaman sabitidir.

Verilen denetleyicide, türev kısmı aynen kullanılırsa ölçüm gürültüsü çok fazla yükselir. Bu yüzden, türev teriminin kazancı yüksek frekanslarda sınırlandırılmalıdır. Burada sT_d transfer fonksiyonunu

$$sT_d \cong \frac{sT_d}{1 + sT_d/K_h} \tag{4.2}$$

şeklinde değiştirerek, kazanç yüksek frekanslarda sınırlanabilir. Sağ taraftaki transfer fonksiyonu, düşük frekanslarda, türeve yakın sonuçlar verdiği halde, yüksek frekanslarda kazancı K_h ile sınırlıdır. $K_h = 10$ ve $T_d = 0.1$ için sT_d ve $\frac{sT_d}{1+sT_d/K_h}$

türevlerinin kazançları Şekil 4.1'de gösterilmiştir.



Şekil 4.1 Türev ve sınırlandırılmış türev terimlerinin frekans cevapları.

PID algoritması ile yapılan çalışmalar sırasında, referans işaretinde sıçramalar olduğu durumlarda türev teriminde, $e(t) = r(t) - y(t)$ hatası yerine sadece $-y(t)$ terimini kullanmanın daha iyi sonuçlar verdiği gözlemlenmiştir. Ancak, uygulamalarımızda kullanacağımız referans işareti sürekli olacağı için, hatanın türevi kullanılabilir.

Böylece PID denetim algoritması s uzayında

$$U(s) = K\left[R(s) - Y(s) + \frac{1}{sT_i}(R(s) - Y(s)) + \frac{sT_d}{1 + sT_d/K_h}(R(s) - Y(s))\right] \quad 4.3$$

şeklinde yazılabilir. Burada, $U(s), R(s)$ ve $Y(s)$, $u(t), r(t)$ ve $y(t)$'nin Laplace dönüşümünü göstermektedir.

Eşitlik 4.3 de verilen PID regülatörü ayrık zamana şu şekilde çevirilebilir.

Orantı terimi :

$$P(t) = K\left[r(t) - y(t)\right]$$

türev veya integral içermediği için bu terimi herhangi bir değişikliğe uğratmaya gerek

yoktur. İntegral terimi

$$I(t) = \frac{K}{T_i}\int_0^t e(\tau)d\tau$$

dikdörtgen yaklaşımı ile

$$I(kh+h) = I(kh) + \frac{Kh}{T_i}e(kh)$$

şeklinde elde edilir.

$$\frac{T_d}{K_h}\frac{dD}{dt} + D = -KT_d\frac{dy}{dt}$$

ile verilen türev kısmını, geri farklar ile yaklaşık olarak hesaplarsak

$$D(kh) = \frac{T_d}{T_d + K_h h}D(kh-h) - \frac{KT_dK_h}{T_d + K_h h}(y(kh) - y(kh-h))$$

elde edilir. Türev terimini bu şekilde ayrıklaştırmanın avantajı, sistemin her zaman kararlı olmasıdır.

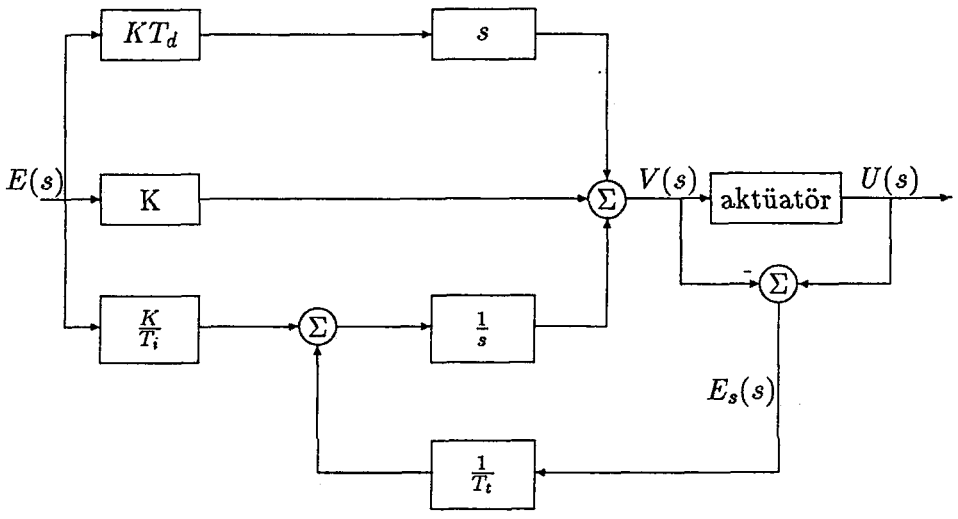
Böylece, denetim işareti

$$u(kh) = P(kh) + I(kh) + D(kh) \tag{4.4}$$

şeklinde verilir.

Entegratör Salınımı

PID denetleyicisinin ürettiği denetim işareti, hata çok büyük olduğunda, aktüatörün üretebileceği en büyük çıkıştan daha fazla olabilir. Böyle durumlarda aktüatör doyar. Böylece entegratör çok büyük değerlere entegre edebilir. Sonunda, hata azaldığında entegratörün içeriği o kadar büyük olabilir ki, tekrar normal bir değere ulaşması önemli bir vakit alabilir. Bu etki, entegratör salınımı olarak adlandırılır. Entegratör salınımını engellemek için kullanılan bir metod, şekil 4.2'de gösterilmiştir. Verilen salınımı önlenmiş PID denetleyicide, sisteme uygulanan denetim büyüklüğü olan aktüatör çıkışı $U(s)$ ölçülmekte ve denetleyici çıkışı $V(s)$ ile arasındaki fark alınarak, $E_s(s)$ hata işareti oluşturulmaktadır. Bu hata işareti, entegratöre $1/T_t$ kazancı ile geri beslenmektedir. Aktüatör doymadığı zamanlarda, hata işareti $E_s(s)$ sıfırdır. Aktüatör doyduğu zaman, eklenen geri besleme yolu $E_s(s)$'yi sıfır yapmaya çalışır. Böylece, denetleyici çıkışı doyma sınırında olacak şekilde, entegratör sıfırlanır ve entegratör uygun değere T_t zaman sabiti ile ulaşır. Bu algoritmanın kullanılabilmesi için, aktüatör çıkışının ölçülebilmesi veya modellenebilmesi gerekir.



Şekil 4.2 Entegratör salınımı önlenmiş PID denetleyici.

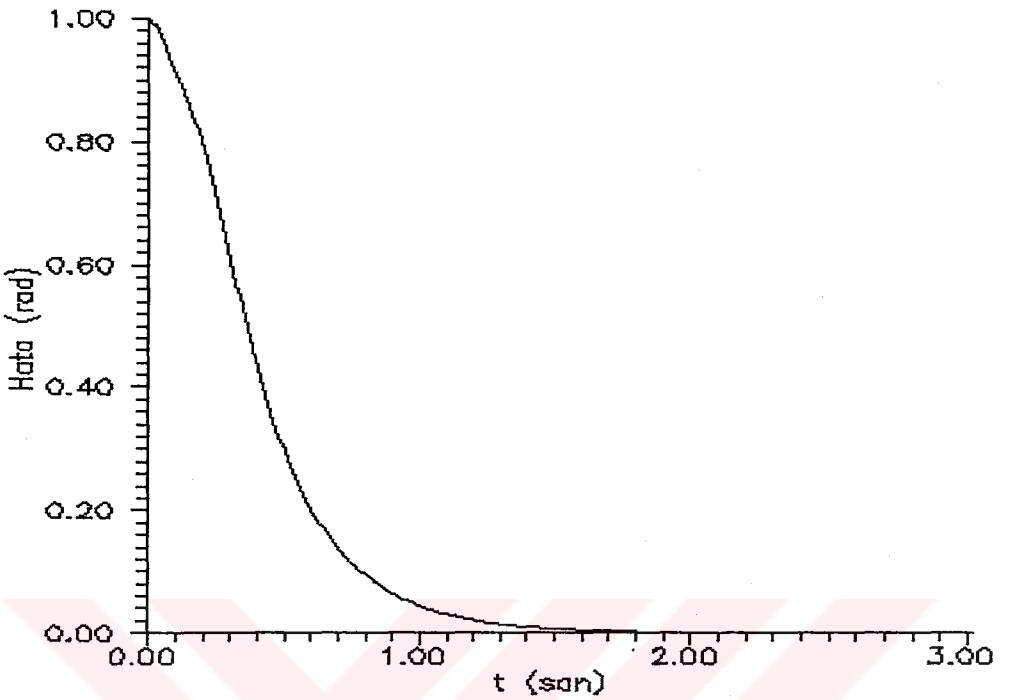
4.3 Simülasyon

Bölüm 2'de verilen, parametreleri zamanla değişen doğrusal olmayan sistem modeli, gürültü modelleri ve denetim işareti üzerindeki sınırlamalar kullanarak Eşitlik 4.4 ile verilen PID denetleyicinin simülasyonunu yapılmıştır. Referans işareti olarak birim basamak girişi alınmıştır. Simülasyonda kullanılan denetleyicide, Şekil 4.2'de gösterilen e_s geri beslemesi kullanılmıştır. Aktüatörün Bölüm 2 de verilen denetim işaretinin üst sınırlarında doyduğu kabul edilmiştir. PID parametreleri olarak

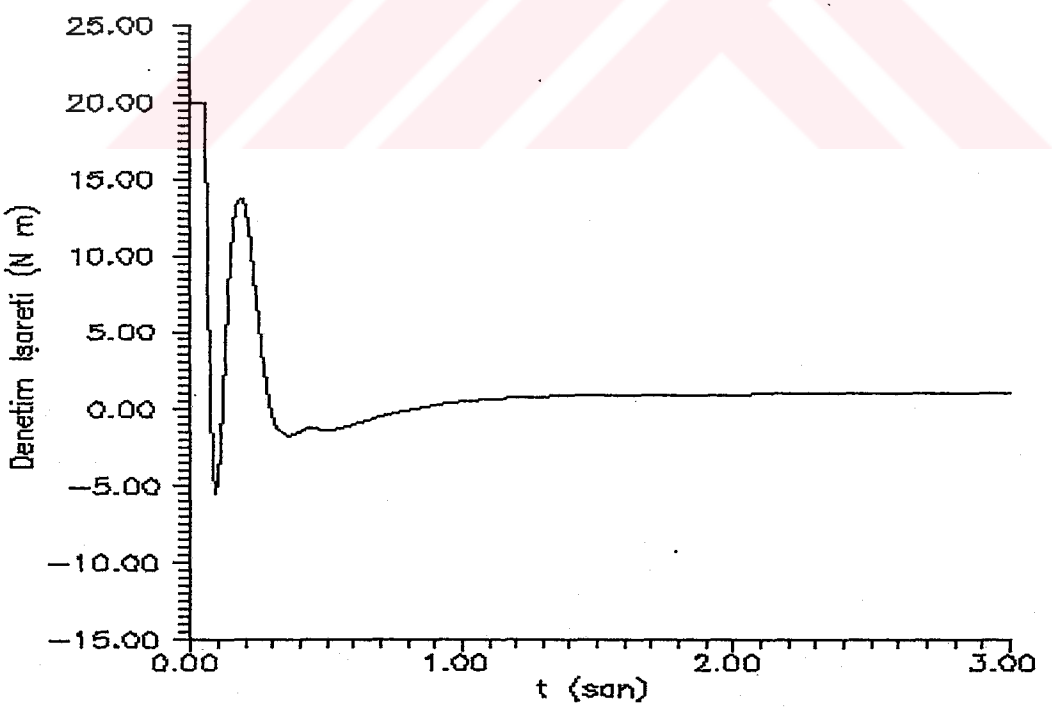
$$K = 200, \quad T_i = 0.4san, \quad T_d = 0.1san, \quad T_t = 0.1san, \quad K_h = 10$$

alınmıştır.

Simülasyon sonucu elde edilen, referans ile çıkış arasındaki fark olan $e(t)$ hatası Şekil 4.3'de, denetim işareti u ise Şekil 4.4'de gösterilmiştir.



Şekil 4.3 Hata, $e(t) = r(t) - y(t)$.



Şekil 4.4 Denetim işareti, $u(t)$.

4.3 Sonuç

Bu bölümde, PID denetleyicinin ayrık zamanda denklemleri çıkarılarak simülasyonu yapılmıştır. PID denetleyici bir regülatör olduğu için, referans girişi olarak birim basamak giriş kullanılmıştır.

PID denetleyicinin avantajı, oluşan kapalı çevrimin kararlılığının garanti edilmiş olmasıdır. Ayrıca, PID denetleyicinin sayısal denetim sistemlerinden talep edeceği hesap yükü, optimal veya adaptif denetleyiciler ile karşılaştırıldığında, son derece azdır. PID denetimi ile, sistemde izin verilen hatanın altında kalınabiliniyorsa, kararlılık kesin olduğu ve ucuz denetleyicilerle gerçeklenebileceği için, PID denetimi kullanmak, genelde iyi bir seçimdir.

PID denetleyicinin simülasyonu sonucu elde edilen Şekil 4.3'deki hata Şekil 2.5 ile karşılaştırılırsa, hatanın yüksek hızlarda arttığı gözlenir. PID denetleyicisi bir regülatör olduğu için bu sonuç beklendiği gibidir.

Bölüm 7'de PID denetleyici, ileri beslemeli servo yapısı içinde kullanılarak performansı diğer servo denetim yöntemleri ile karşılaştırılacaktır.

BÖLÜM 5

OPTİMAL DENETİM

5.1 Giriş

Optimal denetimin amacı, verilen bir kayıp fonksiyonunu en küçük yapacak şekilde, denetim işaretini üretmektir. Bu bölümde, Doğrusal İkincil (LQ) optimal denetim yöntemi, tek giriş ve tek çıkışlı sistemler göz önüne alınarak, incelenmiştir. Bölüm 5.2'de LQ denetim probleminin formülasyonu verilmiştir. Bölüm 5.3'de ise LQ problemi çözülmüştür.

Yapılan hesaplarda, örnekleme aralığı ile karşılaştırıldığında, hesaplama zamanının kısa olduğu dikkate alınarak, hesaplama zamanı ihmal edilmiştir.

5.2 LQ Problem Formülasyonu

Sistem modeli

Denetlenecek sürecin, tek girişli $u(t)$, tek çıkışlı $y(t)$ ve doğrusal olduğu kabul edilmiştir. Denetlenecek sürecin, sürekli zaman modeli

$$dx = Axdt + Budt \tag{5.1}$$

ile verilmiş olsun. Burada

$x(t)$: n boyutlu durum vektörü,
$u(t)$: denetim işareti,
A : $n \times n$ tekil olmayan sistem matrisi,
B : n boyutlu denetim matrisi'dir.

Verilen modelde A ve B matrisleri zamanla değişiyor olabilir. $u(t)$ girişi, örnekleme aralığı süresince sabittir. Eşitlik 5.1'in çözümü

$$x(t) = \Phi(t, kh)x(kh) + \Gamma(t, kh)u(kh) \tag{5.2}$$

ile verilir. Burada $\Phi(t, kh)$, Eşitlik 5.1'in temel matrisidir ve

$$\frac{d}{dt}\Phi(t, kh) = A(t)\Phi(t, kh); \qquad \Phi(kh, kh) = I \tag{5.3}$$

şartlarını sağlar, ve

$$\Gamma(t, kh) = \int_{kh}^{t} \Phi(t, s)B(s)ds \tag{5.4}$$

şeklinde tanımlıdır. Verilen model örneklenmiş halde

$$\begin{aligned} x(kh+h) &= \Phi(kh+h, kh)x(kh) + \Gamma(kh+h, kh)u(kh) \\ y(kh) &= Cx(kh) \end{aligned} \tag{5.5}$$

şeklindedir.

Kriter

LQ denetimin amacı

$$J = \int_0^{Nh} \left[x^T(t)Q_{1c}x(t) + 2x^T(t)Q_{12c}u(t) + u^T(t)Q_{2c}u(t)\right] dt + x^T(Nh)Q_{0c}x(Nh) \tag{5.6}$$

ile verilen kayıp fonksiyonunu en küçük yapmaktır. Burada Q_{0c}, Q_{1c} ve Q_{2c} simetrik ve sıfırdan büyük tanımlı matrisler, N en son ölçümün yapıldığı zaman adımıdır. Kayıp fonksiyonunun içindeki matrisler zamana bağlı olabilirler.

Eşitlik 5.6'da verilen kayıp fonksiyonu sürekli zamanda ifade edilmiştir. Eşitlik 5.6'nın h aralıkları boyunca integralini alarak

$$J = \sum_{k=0}^{N-1} J(k) + x^T(Nh)Q_{0c}x(Nh) \tag{5.7}$$

elde edilir. Burada

$$J(k) = \int_{kh}^{kh+h} \left[x^T(t)Q_{1c}x(t) + 2x^T(t)Q_{12c}u(t) + u^T(t)Q_{2c}u(t)\right] dt \tag{5.8}$$

dır. Eşitlik 5.8'de Eşitlik 5.2'yi ve $u(t)$'nin örnekleme aralığı süresince sabit olduğunu kullanarak

$$J(k) = x^T(kh)Q_1x(kh) + 2x^T(kh)Q_{12}u(kh) + u^T(kh)Q_2u(kh)$$

bulunur. Burada

$$Q_1 = \int_{kh}^{kh+h} \Phi^T(s, kh)Q_{1c}\Phi(s, kh)ds \tag{5.9}$$

$$Q_{12} = \int_{kh}^{kh+h} \Phi^T(s,kh)\,[Q_{1c}\Gamma(s,kh) + Q_{12c}]\,ds \tag{5.10}$$

$$Q_2 = \int_{kh}^{kh+h} \left[\Gamma^T(s,kh)Q_{1c}\Gamma(s,kh) + 2\Gamma^T(s,kh)Q_{12c} + Q_{2c}\right] ds \tag{5.11}$$

şeklindedir.

Örnekleme aralığı süresince $u(t)$ sabit olduğunda, Eşitlik 5.6 ile verilen sürekli zamandaki kayıp fonksiyonunu en küçük yapmak, ayrık zamandaki kayıp fonksiyonu

$$J = \sum_{k=0}^{N-1} \left[x^T(kh)Q_1x(kh) + 2x^T(kh)Q_{12}u(kh) + u^T(kh)Q_2u(kh)\right] + x^T(Nh)Q_0x(Nh) \tag{5.12}$$

ifadesini en küçük yapmakla aynı şeydir. Burada Q_1, Q_{12} ve Q_2 Eşitlik 5.9, 5.10 ve 5.11 ile verilmiştir ve $Q_0 = Q_{0c}$'dir. Q_1'in sıfırdan büyük yarı tanımlı ve Q_2'nin sıfırdan büyük tanımlı olduğu varsayılacaktır. Eşitlik 5.12 ile verilen örneklenmiş kayıp fonksiyonunda, $Q_{12c} = 0$ olsa bile, Q_{12} terimi sıfırdan farklı olabilir.

Böylece optimal denetim problemi, Eşitlik 5.12 ile verilen kayıp fonksiyonunun ayrık zamanda en küçük yapılması problemine dönüşmektedir. Denklemlerin çıkarılması sırasında, yazım kolaylığı açısından, $h = 1$ kabul edilmiştir.

$$\Phi(k+1,k) \equiv \Phi; \qquad \Gamma(k+1,k) \equiv \Gamma$$

şeklinde gösterilsin.

Durum ve denetim işareti arasındaki çarpımı yok etmek için bir dönüşüm yapalım. Yeni bir denetim işareti

$$\begin{aligned} \tilde{u} &= u + M^T x \\ M &= Q_{12}Q_2^{-1} \end{aligned}$$

şeklinde tanımlandığı takdirde, Eşitlik (5.5) ile verilen sistem

$$\begin{aligned} x(k+1) &= \tilde{\Phi}x(k) + \Gamma\tilde{u}(k) \\ y(k) &= Cx(k) \end{aligned}$$

şekline dönüşecektir. Burada

$$\tilde{\Phi} = \Phi - \Gamma M^T$$

dır.

Eşitlik 5.12'deki kayıp fonksiyonu da

$$J = \sum_{k=0}^{N-1} \left[x^T(k)\tilde{Q}_1 x(k) + \tilde{u}^T(k) Q_2 \tilde{u}(k) \right] + x^T(N) Q_0 x(N) \tag{5.13}$$

şekline dönüşecektir. Burada

$$\tilde{Q}_1 = Q_1 - Q_{12} Q_2^{-1} Q_{12}^T$$

dır.

Bu dönüşümlerle elde edilen sistemde, durum vektörü ile denetim işareti arasındaki çarpım terimi yok edilmiştir. Böylece, $Q_{12} = 0$ olduğu durum incelenebilir.

Kareleri tamamlama

İkincil fonksiyonların en küçüğünü bulma yöntemlerinden biri olan kareleri tamamlama yöntemi aşağıdaki gibidir.

$$F(u) = u^T S u + r^T u + u^T r$$

fonksiyonunu gözönüne alalım. Burada, S simetrik, sıfırdan büyük tanımlı, $n \times n$ boyutlu matris ve u ve r ise n boyutlu vektörlerdir. $F(u)$'nun en küçük değeri, $F(u)$ fonksiyonunu :

$$\begin{aligned} F(u) &= u^T S u + r^T u + u^T r \\ &= u^T S u + r^T u + u^T r + r^T S^{-1} r - r^T S^{-1} r \\ &= \left(u + S^{-1} r\right) S \left(u + S^{-1} r\right) - r^T S^{-1} r \end{aligned}$$

şeklinde tekrar yazarak elde edilebilir. İlk terim her zaman sıfırdan küçüktür. Böylece en küçük

$$u = -S^{-1} r \tag{5.14}$$

için elde edilir, ve en küçük değer

$$F_{min} = -r^T S^{-1} r \tag{5.15}$$

dir. Bu sonuç Bölüm 3.3'te Eşitlik 3.8 ile elde edilen sonuca benzemektedir.

5.3 LQ Denetim

Eşitlik 5.5 ile verilen

$$x(k+1) = \Phi x(k) + \Gamma u(k) \tag{5.16}$$

sisteminde, $x(0)$ başlangıç şartı ile LQ denetim probleminin çözümü, Eşitlik 5.12 ile verilen kayıp fonksiyonu en küçük olacak şekilde $u(0), u(1), \ldots, u(N-1)$ denetim işaretlerinin belirlenmesidir. Çözüm Teorem 5.1 ile verilmiştir.

Teorem 5.1: Eşitlik 5.16 ile verilen sistemi gözönüne alalım. Denetim işareti, $u(k)$, $x(k), x(k-1), \ldots$'ın fonksiyonu olsun.

$$\begin{aligned} S(k) &= \Phi^T S(k+1)\Phi + Q_1 - L^T(k)\left(Q_2 + \Gamma^T S(k+1)\Gamma\right) L(k) \\ &= \left[\Phi - \Gamma L\right]^T S(k+1)\Phi + Q_1 \\ &= \left[\Phi - \Gamma L\right]^T S(k+1)\left[\Phi - \Gamma L\right] + Q_1 + L^T Q_2 L \end{aligned} \tag{5.17}$$

tanımını yapalım. Burada L matrisi

$$L(k) = \left(Q_2 + \Gamma^T S(k+1)\Gamma\right)^{-1} \Gamma^T S(k+1)\Phi \tag{5.18}$$

şeklinde tanımlıdır. S için

$$S(N) = Q_0$$

şeklinde sınır şartı verilmiştir. $S(k)$'nin sıfırdan büyük yarı tanımlı bir çözümü olduğunu ve $Q_2 = \Gamma^T S(k)\Gamma$'nın sıfırdan büyük tanımlı olduğunu varsayalım. Bu durumda, $Q_{12} = 0$ olduğunda Eşitlik 5.12 ile verilen kaybı en küçük yapan tek bir denetim stratejisi vardır ve

$$u(k) = -L(k)x(k) \tag{5.19}$$

şeklinde verilir. Kaybın en küçük değeri,

$$minJ = V_0 = x^T(0)S(0)x(0) \tag{5.20}$$

dır.

İspat : Teorem 5.1'i ispatlamak için dinamik programlama kullanılacaktır.

$$V_k = \min_{u(k)\ldots u(N-1)} \left\{ \sum_{i=k}^{N-1} \left[x^T(i)Q_1 x(i) + u^T(i)Q_2 u(i)\right] + x^T(N)Q_0 x(N) \right\}$$

tanımını yapalım.

$k = N$ için

$$V_N = x^T(N)S(N)x(N)$$

dir. Burada

$$S(N) = Q_0$$

şeklinde daha önce sınır şartı olarak verilmişti.

$k = N - 1$ için

$$V_{N-1} = \min_{u(N-1)} \left\{ x^T(N-1)S(N)x(N-1) + u^T(N-1)Q_2u(N-1) + V_N \right\} \quad (5.21)$$

Eşitlik 5.5'i kullanarak

$$\begin{aligned} V_{N-1} &= \min_{u(N-1)} \{ x^T(N-1)Q_1x(N-1) + u^T(N-1)Q_2u(N-1) \\ &\quad + [\Phi x(N-1) + \Gamma u(N-1)]^T S(N) [\Phi x(N-1) + \Gamma u(N-1)] \} \\ &= \min_{u(N-1)} \{ x^T(N-1) \left[Q_1 + \Phi^T S(N)\Phi \right] x(N-1) \\ &\quad + x^T(N-1)\Phi^T S(N)\Gamma u(N-1) + u^T(N-1)\Gamma^T S(N)\Phi x(N-1) \\ &\quad + u^T(N-1) \left[\Gamma^T S(N)\Gamma + Q_2 \right] u(N-1) \} \end{aligned}$$

elde edilir. Eşitlik 5.14 ve 5.15 kullanılarak,

$$u(N-1) = -L(N-1)x(N-1)$$

şeklinde elde edilen denetim yasası,

$$V_{N-1} = x^T(N-1)S(N-1)x(N-1)$$

en küçük kaybını verir. Burada

$$S(N-1) = \Phi^T S(N)\Phi + Q_1 - L^T(N-1) \left(Q_2 + \Gamma^T S(N)\Gamma \right) L(N-1)$$

ve

$$L(N-1) = \left(Q_2 + \Gamma^T S(N)\Gamma \right)^{-1} \Gamma^T S(N)\Phi$$

dır. Aynı şekilde devam ederek

$$V_{N-2} = \min_{u(N-2)} \left\{ x^T(N-2)Q_1x(N-2) + u^T(N-2)Q_2u(N-2) + V_{N-1} \right\}$$

elde edilir. Bu Eşitlik 5.21 ile aynıdır, ancak zaman argümanları bir zaman adımı kaymıştır. Bu işlemi zaman içinde geri adımlarla devam ettirerek, J'nin en küçüğü olan V_0 elde edilir.

Eşitlik 5.19 ile verilen optimal denetim yasası, durum vektörünün zamanla değişen $L(k)$ kazancı ile geri beslenmesini gerektirir. Verilen LQ denetim problemi çözümünün kullanılabilmesi için, Eşitlik 5.17'den $S(k)$ matrisinin çözülerek Eşitlik 5.18'den $L(k)$ kazançlarının hesaplanması gerekir. Eşitlik 5.17 ayrık zamanda Riccati denklemi olarak adlandırılır. Riccati denkleminin $S(N) = Q_0$ sınır şartı

ve ağırlık matrisleri Q_1 ve Q_2 verildiğine göre, denetim işlemine başlamadan önce Eşitlik 5.17'den $S(k)$, $k = N-1, N-2, \ldots, 1, 0$ ve Eşitlik 5.18'den $L(k)$, $k = N-1, N-2, \ldots, 1, 0$ hesaplanarak, gerçek zamanda denetim sırasında kullanılmak üzere saklanabilir. Her adımda optimal denetim vektörü, durum vektörü $x(k)$'yi $-L(k)$ ile çarparak elde edilir.

Verilen LQ optimal denetleyici probleminin çözümünde sistemin tüm durumları kullanılmıştır. Eğer sistemin tüm durumları ölçülemiyorsa, uygun bir gözlemleyici kullanılarak ölçülen durumlardan ölçülemiyen durumlar gözlenmelidir.

Eşitlik 5.19 ile verilen doğrusal durum geri beslemeli denetleyicinin, n adet parametresi vardır. Ağırlık matrislerindeki elemanların birbirlerine göre büyüklüklerini değiştirerek, sistem çıkışının istenen referans değerine ulaşma hızı ile denetim işaretinin büyüklüğü arasında, denge kurmak mümkündür.

5.4 LQG Denetim

Bu bölümde, Doğrusal İkincil Gauss (Lineer Quadratic Gaussian, LQG) denetim probleminin çözümü ispatlanmadan verilmiştir. Bu problemde, denetlenen sistem doğrusal ve sistem üzerindeki gürültülerin Gauss dağılımına uygun olduğu kabul edilmiştir. İkincil bir performans kriterini en küçük yapacak denetim yasası aranmaktadır. Burada sistemin durumları Kalman süzgeci kullanılarak kestirilecektir.

Örneklenmiş sistem modeli

$$\begin{aligned} x(kh+h) &= \Phi x(kh) + \Gamma u(kh) + v(kh) \\ y(kh) &= Cx(kh) + e(kh) \end{aligned} \tag{5.22}$$

olarak yazılmış olsun. Burada, $v(k)$ ve $e(k)$'yi sıfır ortalamalı, Gauss olasılık dağılımına sahip rastgele değişkenler olarak tanımlanmıştır.

$$\begin{aligned} R_1 &= Varyans\,(v(kh)v^T(kh)) \\ R_{12} &= Varyans\,(v(kh)e(kh)) \\ R_2 &= Varyans\,(e^2(kh)) \end{aligned} \tag{5.23}$$

olarak verilmiştir.

Eşitlik 5.22 ile verilen süreçte bir adım sonrasının kestirimi için Kalman süzgeçi kullanılmaktadır. Bu durumda Kalman süzgeci, $h = 1$ için aşağıdaki şekilde verilir:

$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + K(k)\left[y(k) - C\hat{x}(k)\right] \tag{5.24}$$

$$K(k) \quad = \Phi P(k)C^T(R_2 + CP(k)C^T)^{-1} \tag{5.25}$$

$$P(k+1) = \Phi P(k)\Phi^T + R_1 - \Phi P(k)C^T(R_2 + CP(k)C^T)^{-1}CP(k)\Phi^T \tag{5.26}$$

Eşitlik 5.23 ile verilen sistemde, kabul edilen denetim yasaları $u(k)$'nin $y(k)$'nin fonksiyonu olacağı şekilde olsun. Eşitlik 5.17'nin, $S(N) = Q_0$ sınır şartı ile $S(k)$ şeklinde sıfırdan farklı tanımlı olsun. Böylece, Eşitlik 5.17'yi en küçük yapan kabul edilir tek bir denetim yasası vardır ve

$$u(k) = -L(k)\hat{x}(k) \tag{5.27}$$

ile verilir. Bu denetim yasası, gürültü gerçekten Gauss dağılımına uyuyorsa optimaldir.

5.5 Simülasyonlar

Simülasyon 5.1

Bölüm 2.2.2'de verilen sönümlü ikinci derece osilatörün simülasyonu yapılmıştır.

$$w_0 = 1, \qquad \xi = 0.7$$

için ayrık durum denklemi, Eşitlik 2.9 ve 2.10'dan

$$x(kh+h) = \begin{bmatrix} 1 & 0.0099 \\ -0.0099 & 0.9860 \end{bmatrix} x(kh) + \begin{bmatrix} 0.00005 \\ 0.00993 \end{bmatrix}$$

olarak elde edilir.

Eşitlik 5.17

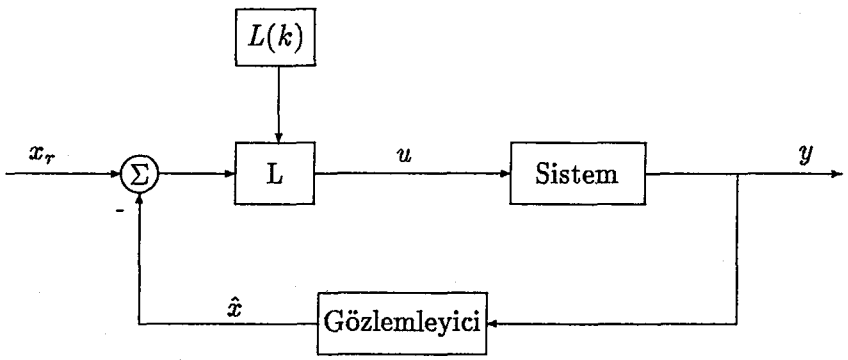
$$S(N) = Q_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

sınır şartı kullanılarak, geri adımlarla hesaplanmıştır. Eşitlik 5.19'dan geri besleme kazanç matrisi $L(k)$ hesaplanır. Optimal denetleyicinin ağırlık matrisleri

$$Q + 1 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \quad Q_2 = 0.5 \quad Q_{12} = 0$$

alınmıştır. Ağırlık matrislerini bu şekilde seçmekle, en büyük önem konum hatasına verilmiştir.

Elde edilen, LQ probleminin çözümü, Şekil 5.1 'de gösterilen durum geri beslemeli denetleyici haline dönüşmüştür. Burada her adımda L geri besleme kazancı önceden hasaplanan kazanç ile güncelleştirilmektedir.



Şekil 5.1 Durum geri beslemeli denetleyici.

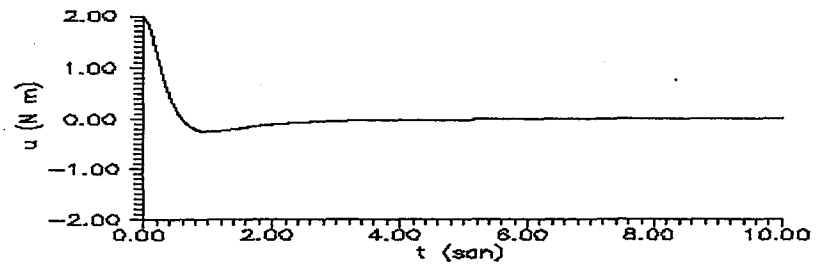
Simülasyonda sistemin, sadece $y(k)$ çıkışının ölçüldüğü kabul edildiği için durum vektörünü oluşturmak üzere, dy/dt Eşitlik 4.2 ile verilen gözlemleyici kullanılarak hesaplanmıştır. Burada $K_h = 10$, $T_d = 1$ alınmıştır.

Sistem

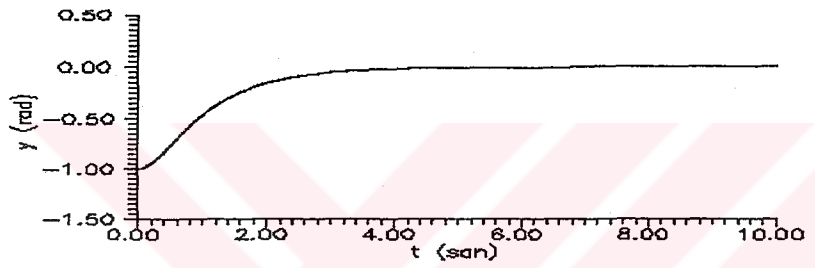
$$x(0) = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

başlangıç durumundan başlatılmış, ve sıfır referansına götürülmüştür. Şekil 5.1 de gösterilen k anındaki referans durum vektörü $x_r(k)$ sıfır alınmıştır. Böylelikle sistemin birim basamak cevabı elde edilmiştir. Eşitlik 5.22' ile verilen sistem gürültüleri sıfır alınmıştır.

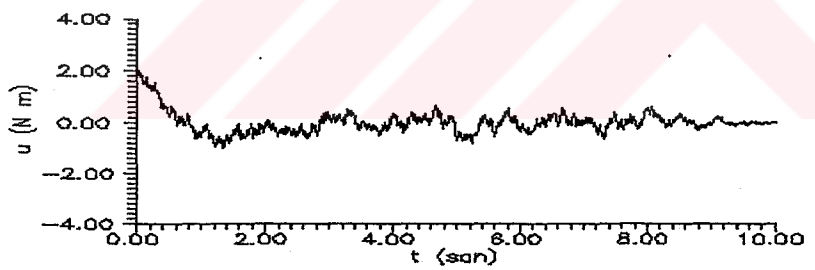
Şekil 5.2'de denetim işareti ve Şekil 5.3'te sistemin cevabı verilmiştir. Sisteme, ölçüm gürültüsü e için ortalaması 0, varyansı 0.01 olan beyaz gürültü ilave edilerek elde edilen denetim işareti ve sistem cevabı Şekil 5.4 ve 5.5'de gösterilmiştir. Burada sistem gürültüsüne rağmen LQ denetleyici kullanılmıştır. Şekil 5.6 ve 5.7 de L kazancının birinci ve ikinci bileşenleri zamana bağlı olarak gösterilmiştir. Ağırlık matrisleri ve sistem parametreleri zamanla değişmediği için Riccati denkleminden elde edilen S matrisi belli bir değere yakınsamaktadır. Dolayısıyla, $L(k)$ kazançları da yakınsar. Burada $L(k)$ hesabının $x(0)$ başlangıç durumundan etkilenmediğine dikkat etmek gerekir.



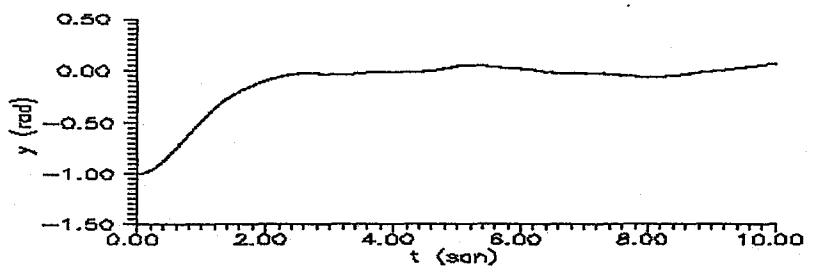
Şekil 5.2 Gürültüsüz durumda denetim işareti.



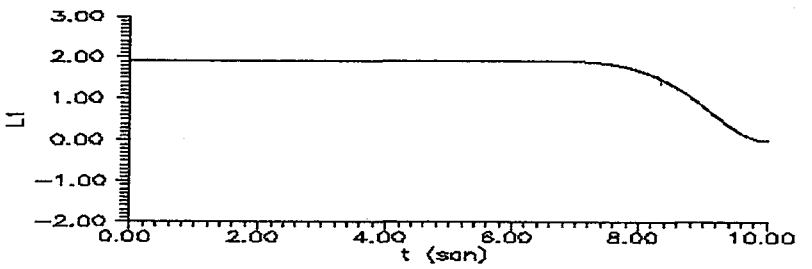
Şekil 5.3 Gürültüsüz durumda sistem çıkışı.



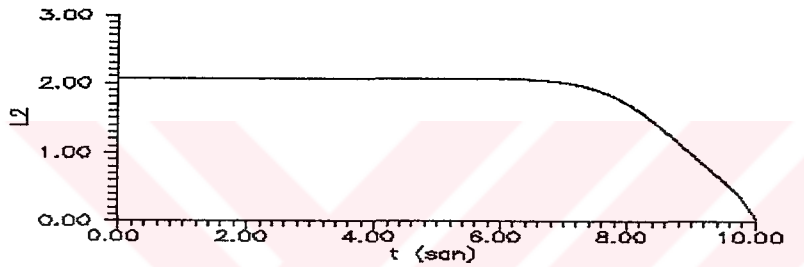
Şekil 5.4 Gürültülü durumda denetim işareti.



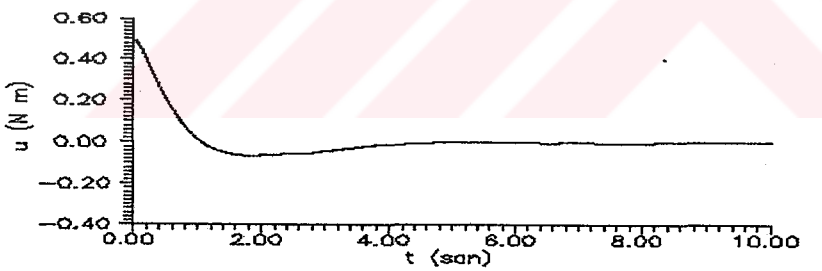
Şekil 5.5 Gürültülü durumda sistem çıkışı.



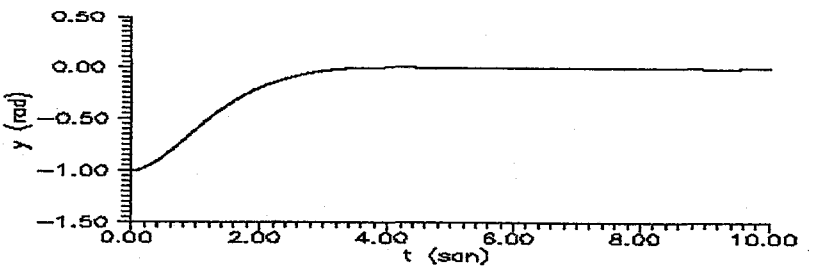
Şekil 5.6 L_1



Şekil 5.7 L_2



Şekil 5.8 $q_2 = 2$ için denetim işareti.

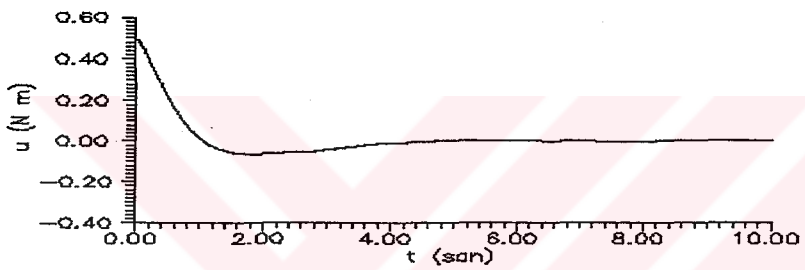


Şekil 5.9 $q_2 = 2$ için sistem çıkışı.

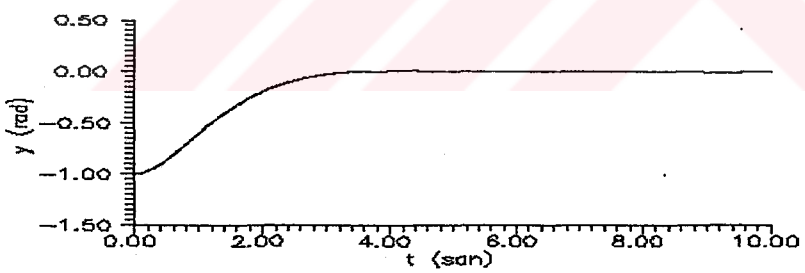
Ağırlık matrislerinin etkisini göstermek üzere,$q_2 = 2$ için aynı sistem ile yapılan simülasyon sonucu elde edilen denetim işareti ve sistem cevabı Şekil 5.8 ve 5.9 ile gösterilmiştir. Şekil 5.3 ile Şekil 5.10 karşılaştırılırsa denetim işaretinin genliğinin azaldığı görülür.

Simülasyon 5.2

Simülasyon 5.1'in aynısı, gürültü ilavesi ie yapılmıştır. Ancak denetim işareti, LQG denetim yasası kullanılarak elde edilmiştir. Denetim işareti $u(t)$ ve sistem çıkışı $y(t)$ Şekil 5.10 ve Şekil 5.11'de gösterilmiştir.



Şekil 5.10 LQG denetim için denetim işareti.



Şekil 5.11 LQG denetim için sistem cevabı.

5.6 Sonuç

Bu bölümde bir optimal denetim yöntemi olan LQ ve LQG denetim yasalarının türetilişi gösterilmiştir. Simülasyon 5.1'in sonuçlarından, sistemde gürültü olmadığı durumlarda LQ denetim yasasının yeterli olduğu görülmektedir. Ancak, Şekil 5.4 ve 5.5'deki sonuçlardan sisteme %1 ölçüm gürültüsüne karşılık gelen gürültü eklendiğinde LQ denetimin yetersiz kaldığı anlaşılmaktadır. Aynı gürültü ve aynı sistem için LQG denetim yasası kullanarak yapılan simülasyon sonuçlarından, LQG denetimin bu gürültüye rağmen doğru denetim işaretlerini üretebildiği görülmektedir.

BÖLÜM 6

KENDİNİ UYARLAYAN DENETİM

6.1 Giriş

Bölüm 4'de verilen PID denetleyici ve Bölüm 5.2'de verilen LQ denetleyiciyi pratikteki problemlere uygulamak, önemli bir gayret gerektirir. Sistemin modellenmesi, sistem parametrelerinin kestirilmesi, denetleyicinin tasarımı ve duyarlılık analizlerinin yapılması gerekir [5]. Tatmin edici bir sonuca ulaşılana kadar bu adımların birkaç kez tekrar edilmesi de gerekebilir [6]. Bu işlemler, daha karmaşık denetleyiciler kullanılarak basitleştirilebilinir. Bir seçenek, tüm işlemlerin otomatikleştirilmesidir. Bu mekanizma, denetleyiciye, parametre kestirim ve denetleyici tasarımı algoritmaları eklenerek sağlanabilir. Böyle bir yaklaşım, Kendini Ayarlayan Denetim, (Self Tuning Regulator, STR), olarak adlandırılır. Kendini ayarlayan denetim ile elde edilen kapalı çevrimler doğrusal değildirler ve zamanla değişirler.

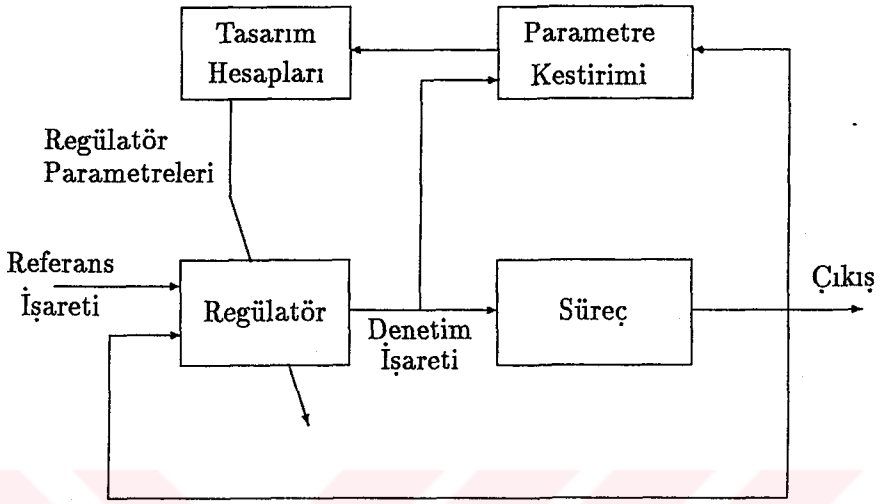
6.2 Kendini Ayarlayan Denetim

Kendini ayarlayan denetim için farklı bir çok yöntem kullanılabilinir. Bir yöntem aşağıdaki gibi olabilir:

1. Uygun bir model yapısı belirlenmesi,
2. Modelin parametrelerinin Bölüm 3'te verilen en küçük kareler yöntemi ile kestirilmesi,
3. Uygun bir denetleyici tasarım yöntemi kullanılarak, denetim işaretinin hesaplanması.

Elde edilen regülatör, kendi parametrelerini ayarlayabildiği için, kendini ayarlayan regülatör olarak adlandırılır. Regülatör, iki döngüden oluşmuş olarak düşünülebilinir. İç döngü, denetlenen sistemi ve doğrusal geri beslemeli bir regülatörü içerir. Regülatörün parametreleri, bir rikörsif parametre kestirici ve bir denetleyici tasarım

algoritmasından oluşan, dış döngü tarafından ayarlanır. Şekil 6.1 kendini ayarlayan bir denetleyiciyi göstermektedir.



Şekil 6.1 Kendini ayarlayan denetleyici.

Denetleyici tasarımı

Denetleyici tasarım yöntemi olarak, kutup yerleştirme problemini gözönüne alalım. Süreç

$$A(q)y(k) = B(q)u(k) \tag{6.1}$$

modeli ile tanımlanmış olsun. Burada $u(k)$ denetim işareti ve $y(k)$ sistemin ölçülen çıkış işaretidir. Referans girişinden, çıkış işaretine transfer fonksiyonu

$$H_m = \frac{y(k)}{r(k)} = \frac{B_m}{A_m} \tag{6.2}$$

şeklinde olan bir regülatör tasarlamak istensin. Burada $r(k)$ sistemin referans girişidir. Genel halde H_m transfer fonksiyonunu vermek yeterli değildir. Sistem çıkışından geri besleme yapıldığında, denetim işaretinin harekete geçiremiyeceği başka modlar olacaktır. Bu yüzden A_0 gözlemleyici polinomunun da verilmesi gerekir.

Elde edilecek regülatörün bir çıkışı $u(k)$ ve iki girişi, $r(k)$ ve $y(k)$, olacaktır. Genel halde bu regülatör için doğrusal bir yapı :

$$u(k) = \frac{T_1(q)}{R_1(q)}r(k) - \frac{S_1(q)}{R_2(q)}y(k) \tag{6.3}$$

şeklinde yazılabilir. Burada R_1, R_2, T_1 ve T_2, ileri kaydırma işlemcisi q cinsinden polinomlardır. Eşitlik 6.3 ile verilen denetim yasası :

$$R(q)u(k) = T(q)r(k) - S(q)y(k) \tag{6.4}$$

şeklinde yazılabilir. $R = R_1R_2$, $T = T_1R_2$ ve $S = S_1R_2$ dir. Burada R'nin en büyük kuvvetinin katsayısının bir birim olduğu kabul edilmiştir. Eşitlik 6.4 ile verilen denetim yasası, referans işareti $r(k)$'den transfer fonksiyonu :

$$H_{ff}(q) = \frac{T(q)}{R(q)} \tag{6.5}$$

olan bir ileri besleme ve
$y(k)$ çıkışından transfer fonksiyonu :

$$H_{fb}(q) = \frac{S(q)}{R(q)} \tag{6.6}$$

olan bir geri beslemeden oluşur. İleri ve geri besleme sinyallerinin nedensel olması için

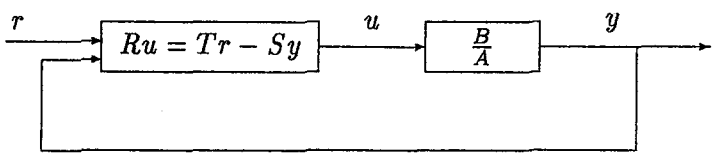
$$derece(R) \geq derece(T) \tag{6.7}$$

$$derece(R) \geq derece(S) \tag{6.8}$$

şartlarının sağlanması gerekir. Bilgisayarda denetim işaretini hesaplamak için gereken zamanın, örnekleme periyodunun küçük bir kesri olduğu kabul edilirse

$$derece(R) = derece(T) = derece(S) \tag{6.9}$$

olur. Elde edilen denetim sistemi, Şekil 6.2'de gösterilmiştir.



Şekil 6.2 Kutup yerleştirme probleminin gösterimi.

Eşitlik 6.1 ile 6.4 arasında $u(k)$ yok edilirse

$$y(k) = \frac{BT}{AR + BS} r(k) \tag{6.9}$$

elde edilir. Bu giriş çıkış ilişkisini Eşitlik 6.2'ye eşit kılarak

$$\frac{BT}{AR + BS} = \frac{B_m}{A_m} \tag{6.10}$$

bulunur. Böylece tasarım problemi, Eşitlik 6.10'u sağlayan R, S ve T polinomlarının bulunmasına dönüşür.

Eşitlik 6.10'un, birden çok çözümü vardır. Örneğin

$$R = BA_m, \qquad S = 0, \qquad T = AB_m$$

bir çözümdür ve sadece ileri beslemeli bir denetleyiciyi oluşturur. S ve T polinomları eşit seçilirse, denetim yasası :

$$Ru(k) = S\left[r(k) - y(k)\right] = Se(k)$$

şeklindedir ve sadece hatadan geri beslemeli bir denetleyicidir.

Şimdi, R, T ve S polinomlarının, kendini uyarlayan denetim içinde kullanılabilir bir denetim yasası oluşturacak biçimde belirlenmesi için bir yöntem verilecektir.

Eşitlik 6.10'dan, kapalı çevrim kutuplarının

$$AR + BS = 0 \tag{6.11}$$

karakteristik denkleminin çözümleri olduğu görülür. Kapalı çevrimli sistemin sıfırları, B ve T polinomlarının sıfırlarıdır. Genellikle kapalı çevrimli sistemin derecesi, verilen referans modelden daha yüksektir. Bu yüzden Eşitlik 6.10'un sağlanabilmesi için, kutuplarla sıfırlar arasında birbirini yok etme olmalıdır.

İlk önce açık çevrim sıfırlarını, yani B polinomunun sıfırlarını, gözönüne alalım. Eğer B'nin bir çarpanı B_m'nin bir çarpanı değilse, $AR + BS$'in bir çarpanı olmalıdır ve bir kapalı çevrim kutbu tarafından yok edilmelidir. Kapalı çevrimli sistemin kararlı olması gerektiğinden, sadece kararlı sıfırlar yok edilebilinir. B'yi

$$B = B^+ B^- \tag{6.12}$$

şeklinde çarpanlarına ayıralım. B^-'nin tüm çarpanları birim dairenin dışında, B^+'nın tüm çarpanları ise birim dairenin içinde olsun. B^+'nın en büyük kuvvetinin katsayısını 1 alalım.

B^- $AR + BS$'nin bir çarpanı olamayacağına göre, B_m'yi bölmelidir,

$$B_m = B^- B'_m \tag{6.13}$$

Bu, kararsız süreç sıfırlarının değiştirilemeyeceğini ve B_m'ye dahil edilmeleri gerektiğini belirtir. Böylece B^+, $AR + BS$'nin bir çarpanı olacaktır, buradan R'nin de bir çarpanı olduğu anlaşılır.

$$R = B^+ R' \tag{6.14}$$

Eşitlik 6.10,

$$\frac{B^+B^-T}{B^+(AR' + B^-S)} = \frac{B^-B'_m}{A_m}$$

şeklinde yazılabilir. Ortak çarpanlar yok edilirse,

$$\frac{T}{(AR' + B^-S)} = \frac{B'_m}{A_m}$$

bulunur. Buradan A_m'nin $AR' + B^-S$'nin bir çarpanı olduğu görülür. Referans işaretinden çıkışa olan transfer fonksiyonunda, gözlemleyici polinomunun yok edilmesi gerekir. Denetim işaretinin gözlemleyici hatalarını etkilememesi için, gözlemleyici kutuplarının transfer fonksiyonu kutuplarını içermemesi gerekir. Yukarıda yazılan eşitlik bu şart kullanılarak elde edilmiştir. Bu sebepten dolayı, A_0'ın $AR + BS$'nin bir çarpanı olması gerekir. Sonuçta şu şartlar elde edilir:

$$AR' + B^-S = A_0A_m \tag{6.15}$$

$$T = B'_mA_0 \tag{6.16}$$

Böylece kapalı çevrim karakteristik denklemi :

$$AR + BS = B^+A_0A_m \tag{6.17}$$

şeklinde elde edilir. Bu şekilde, kapalı çevrim kutupları ile kararlı süreç sıfırları B^+, model kutupları A_m ve gözlemleyici kutupları A_0 yok edilmiş olur.

Özetle, kutup yerleştirme probleminde, regülatör :

$$Ru(k) = Tr(k) - Sy(k) \tag{6.18}$$

ile verilir. R, S ve T polinomları

$$AR_1 + B^-S = A_mA_0 \tag{6.19}$$

Diophantine denklemi R_1 ve S için çözülerek bulunur. Burada

$$R = R_1B^+ \qquad T = A_0B'_m \tag{6.20}$$

ve

$$B_m = B^-B'_m \qquad B = B^-B^+$$

dır.

Kendini Ayarlayan Denetim Algoritması

Bir çok kendini ayarlayan denetim algoritması vardır. Bu algoritmalardan biri aşağıda verilmiştir :

1. Eşitlik 6.1'de verilen A ve B polinomlarının katsayılarını rikörsif en küçük kareler yöntemi ile kestir,
2. A ve B yerine Adım 1'de kestirilen değerlerini koy ve Eşitlik 6.19'dan R_1 ve S'yi çöz. Eşitlik 6.20'den R ve T'yi hesapla,
3. Eşitlik 6.18'den denetim işaretini hesapla.

Bu adımlar her örnekleme aralığında tekrar edilir.

6.3 Simülasyonlar

Simülasyon 6.1

Parametreleri zamanla değişmeyen çift entegratörün simülasyonu yapılmıştır. Çift entegratörün parametresi $J_e = 1$ alınmıştır. Referans modeli olarak

$$w_m = 1, \qquad \xi_m = 0.7$$

olan salınım devresi alınmıştır. Eşitlik 2.12'den

$$a_{m1} = -1.9860 \quad a_{m2} = 0.9860 \quad b_{m1} = 0.00005 \quad b_{m2} = 0.00005$$

elde edilir.

R, S, T ve A_0 polinomları birinci derece olarak kabul edilmiştir. Bu durumda, Bölüm 6.2'de tartışılan kutup yerleştirme probleminin çözümü şu şekilde verilir:

$$\begin{aligned}
r_2 &= ((-\hat{b}_2/\hat{b}_1 + a0) \times (\hat{b}_2^2/\hat{b}_1^2 - \hat{b}_2/\hat{b}_1 a_{m1} + a_{m2})/ \\
&\qquad (\hat{b}_2^2/\hat{b}_1^2 - \hat{a}_1\,\hat{b}_2/\hat{b}_1 + \hat{a}_2)) + \hat{b}_2/\hat{b}_1 \\
t_1 &= (1 + a_{m1} + a_{m2})/(\hat{b}_1 + \hat{b}_2) \\
t_2 &= t_1 * a0 \\
s_1 &= (a0 + a_{m1} - \hat{a}_1 - r_2)/\hat{b}_1 \\
s_2 &= (a_{m2} * a0 - r_2 * \hat{a}_2)/\hat{b}_2 \\
u(k) &= -r_2 * u(k-1) + t_1 * r(k) + t_2 * r(k-1) - s_1 * y(k) - s_2 * y(k-1)
\end{aligned}$$

Burada

$$\begin{aligned} R &= q + r_2 \\ S &= s_1 q + s_2 \\ A_0 &= q + a_0 \end{aligned}$$

olarak tanımlıdır. Bu çözümü elde ederken :

$$B = B^- = B_m, \qquad B^+ = 1, \qquad B_m^{'} = 1$$

olarak alınmıştır.

Gözlemleyici polinomunun kutbunu sıfıra getirmek için ise $a_0 = 0$ yapılmıştır.

Rikörsif en küçük kareler kestiriminde :

$$P(0) = 10I \qquad \lambda = 0.9$$

kullanılmıştır. Ancak, simülasyona başlamadan önce 100 adım sistem beyaz gürültü ile tahrik edilerek, rikörsif en küçük kareler yöntemi çalıştırılmıştır. Böylece, denetim işlemi başladığında sistem parametreleri doğruya yakın olarak kestirilebilir. Eğer, denetime $P(0)$ dağılımı ile başlanırsa, sistem parametreleri doğru kestirilmeye başlanana kadar yanlış denetim işareti üretilmektedir. Pratikte, kendini uyarlayan denetime başlamadan önce, parametrelerin kestirimi için sistemi beyaz gürültü ile tahrik etmek uygun değilse, kendini uyarlamayan bir denetim yasası, sistem parametreleri doğruya yaklaşana kadar kullanılmalıdır.

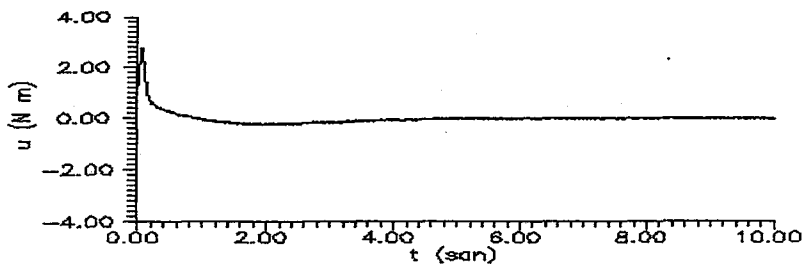
Simülasyonda sistem gözlenebilirliğini arttırmak için, denetim işaretinin üzerine varyansı 0.1, ortalaması 0 olan beyaz gürültü eklenmiştir.

Şekil 6.3'te, denetim işareti ve Şekil 6.4'te denetlenen sistemin birim basamak cevabı görülmektedir. Şekil 6.5'te, sistem cevabı ile verilen referans model cevabı arasındaki fark verilmiştir.

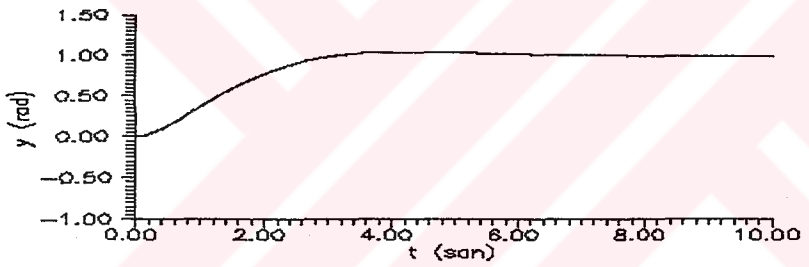
Simülasyon 6.2

Simülasyon 6.1 Eşitlik 2.20 ile verilen referans işareti için yapılmıştır. Simülasyon 6.1'de verilen denetim parametrelerinden sadece $w_m = 250$ olarak değiştirilmiştir. Sistem denkleminde ataletin Eşitlik 2.3 ile verildiği gibi değiştiği kabul edilmiştir. Denetim işlemi başlamadan önce sistem parametrelerinin hesaplanması sırasında,

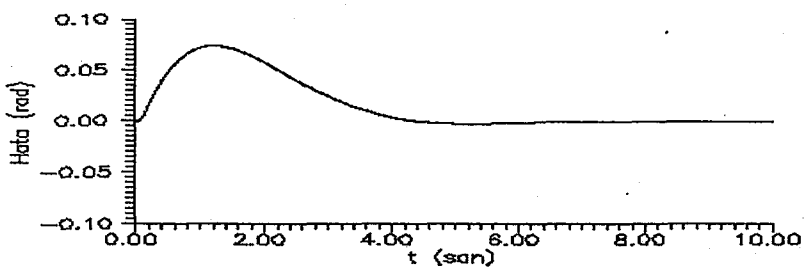
Eşitlik 2.3 ile verilen J ataleti 0.5 olarak sabit tutulmuştur. Şekil 6.6'da denetim işareti , Şekil 6.7'de sistemin cevabı ve Şekil 6.8'de sistem cevabı ile referans işareti arasındaki fark gösterilmiştir. Sistem cevabının, verilen referansı daha yakından takip etmesini sağlamak için, referans modelin doğal frekansının arttırılması gerekir. Ancak bu eylem, denetim işaretinin genliğinin artmasına sebep olur.



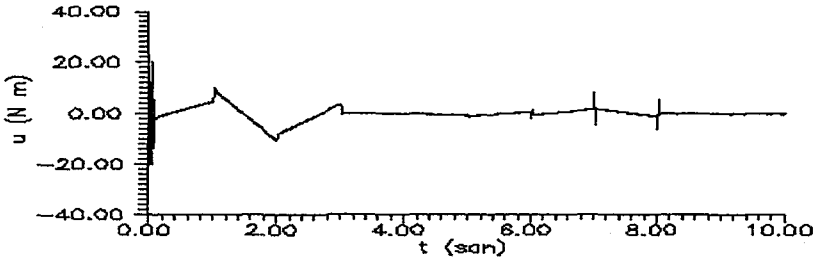
Şekil 6.3 Simülasyon 6.1 için denetim işareti.



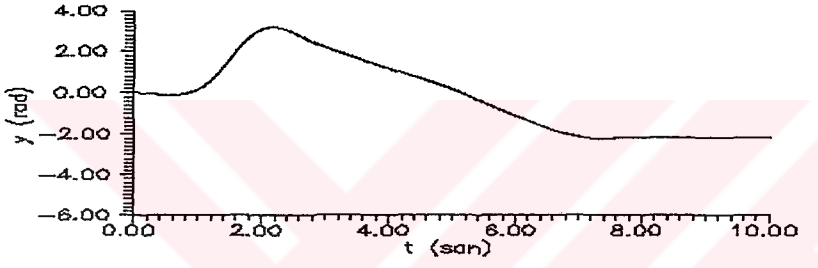
Şekil 6.4 Simülasyon 6.1 için sistem çıkışı.



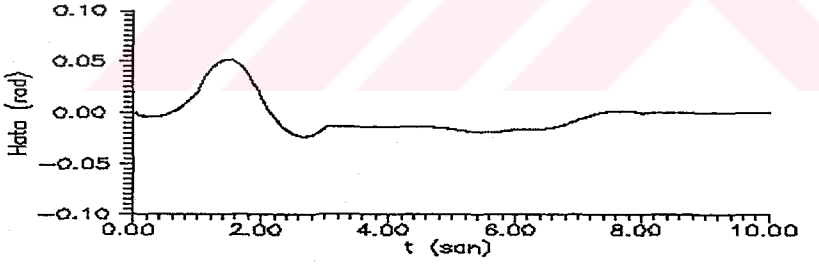
Şekil 6.5 Simülasyon 6.1'de referans model ile sistem çıkışının farkı.



Şekil 6.6 Simülasyon 6.2 için denetim işareti.



Şekil 6.7 Simülasyon 6.2 için sistem çıkışı.



Şekil 6.8 Simülasyon 6.2'de referans girişi ile sistem çıkışının farkı.

6.4 Sonuç

Kendini uyarlayan (adaptive) denetim yöntemlerinden biri olan kendini ayarlayan regülatör (STR) incelenmiştir. Denetim yasası, kutup yerleştirme problemi çözülerek elde edilmiştir. Sistemin parametreleri, en küçük kareler yöntemi kullanılarak sağlanmıştır.

Simülasyon 6.1'de parametreleri değişmeyen sistem için regülatörün birim basamak cevabı incelenmiştir. Sistem parametrelerinin kestirilebilirliğini sağlamak için ise, denetim işaretinin üzerine beyaz gürültü eklenmiştir.

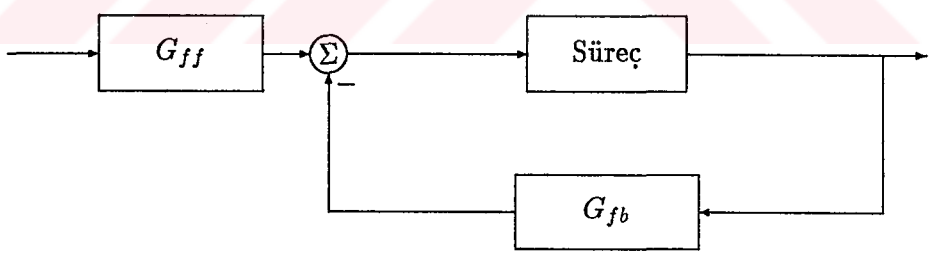
Simülasyon 6.2'de sistem denklemi olarak Bölüm 2'de verilen, parametreleri zamana göre değişen sistem simüle edilmiştir. Burada kendini ayarlayan regülatörü, servo denetleyici olarak kullanabilmek için, referans modelin doğal frekansı çok yüksek seçilmiştir. Bu, ancak referans işareti sürekli olduğu zaman yapılabilir. Referans işareti süreksiz olduğunda, referans modelin doğal frekansı yüksek ise, denetim işareti çok büyüyecektir ve sistem kararsız olacaktır.

BÖLÜM 7

İLERİ BESLEMELİ SERVO DENETLEYİCİ YAPISI

7.1 Giriş

Şimdiye kadar tartışılan tüm denetleyiciler regülatör tipinde idi. Regülatörlerde istenen, bozucu etkilerin yok edilerek sistem çıkışını istenen sabit bir değerde tutmaktır. Servo problemde ise istenen, sistem çıkışlarının ve durumlarının verilen yörüngeleri izlemesini sağlamaktır. Pratikte, denetim sistemlerinden hem servo hem de regülatör özelliklerini göstermeleri istenir. Bu istek, Şekil 7.1'de gösterildiği gibi, iki serbestlikli bir yapı kullanılarak sağlanabilir. Bu yapının özelliği, servo ve regülatör özelliklerinin ayrılmış olmasıdır. G_{fb} ile gösterilen geri beslemeli denetleyici, sistemin yükünde meydana gelen değişimlere ve süreç belirsizliklerine duyarsız bir kapalı çevrim elde etmek üzere tasarlanır [7]. Daha sonra istenen servo özelliklerini sağlamak üzere ileri besleme tasarlanır.



Şekil 7.1 İki serbestlikli servo denetleyicisinin gösterimi.

Regülatör özellikleri, önceki bölümlerde açıklandığı için bu bölüme servo özellikleri incelenecektir.

7.2 Servo Özellikleri

$$u(k) = -L\hat{x}(k) \tag{7.1}$$

ile verilen durum geri beslemeli denetimi gözönüne alalım. Burada $\hat{x}$ kestirilen du-

rumdur. Durum sıfıra götürülmek istenmiyorsa, denetim

$$u(k) = L\left[x_m(k) - \hat{x}(k)\right] + u_m(k) \tag{7.2}$$

şeklinde değiştirilebilinir. Burada, $x_m(k)$, k anında bulunulması istenen durum, ve $u_m(k)$, k anında denetim işaretinin modelden hesaplanan değeridir. $L\left[x_m(k) - \hat{x}(k)\right]$ terimi geri beslemeyi, $u_m(k)$ terimi ise ileri beslemeyi göstermektedir. Eğer kestirilen $\hat{x}(k)$ durumu, istenilen durum $x_m(k)$ ile aynı ise, geri besleme işareti $L\left[x_m(k) - \hat{x}(k)\right]$ sıfır olacaktır. Eğer x_m ve $\hat{x}$ arasında bir fark varsa, geri besleme düzeltme işareti üretecektir.

Bu çalışmada, Eşitlik 2.20 ile verilen $r(t)$ referans işaretinin izlenmesi amaçlanmaktadır. Bu durumda, t anında bulunulması istenen durum

$$x_m(t) = \begin{bmatrix} r(t) \\ \dot{r}(t) \end{bmatrix} \tag{7.3}$$

şeklinde yazılabilir. Bunun için $\dot{r}(t)$'nin sürekli olması gerekir. Referans işareti sürekli değilse, Bölüm 2.2.2'de verilen doğrusal sistem denklemlerinden biri kullanılarak, durum vektörü x_m elde edilmeli ve servo denetim probleminde kullanılmalıdır.

İleri besleme işareti $u_m(k)$'yi üretmek için, Eşitlik 2.13 ile verilen çift entegratör denklemi kullanılmıştır. Burada $J_e = 1$ olarak alınmıştır. Eşitlik 2.13, $J_e = 1$ alınarak tekrar yazılırsa:

$$\frac{d^2y}{dt^2} = u(t)$$

Bu ifade ayrık zamanda :

$$y(k+1) - 2y(k) + y(k-1) = \frac{h^2}{2}(u(k) + u(k-1))$$

olacaktır.
$(kh+h)$ anında $y(kh+h) = r(kh+h)$ çıkışını elde etmek için

$$u_m(kh) = \frac{2}{h^2}\left[r(kh+h) - 2y(kh) + y(kh)\right] - u_m(kh-h) \tag{7.4}$$

şeklinde alınmalıdır.

Geri besleme işaretini elde etmek için daha önce verilen PID, LQ ve STR regülatörleri burada karşılaştırılabilinir.

7.3 PID Geri Beslemesi

Bölüm 4'de verilen PID regülatör, iki serbestlikli servo denetleyicisi içinde kullanılabilinir. Böylece denetim işareti :

$$u(kh) = P + I + D + u_m(kh) \tag{7.5}$$

olarak verilir. Bu eşitlikteki P, I ve D Bölüm 4.2'de, $u_m(kh)$ ileri beslemesi ise Eşitlik 7.4 ile verilmiştir.

Simülasyon 7.1

Bölüm 2.2.1'de verilen doğrusal olmayan sistemin bilgisayar simülasyonu, PID regülatör parametreleri :

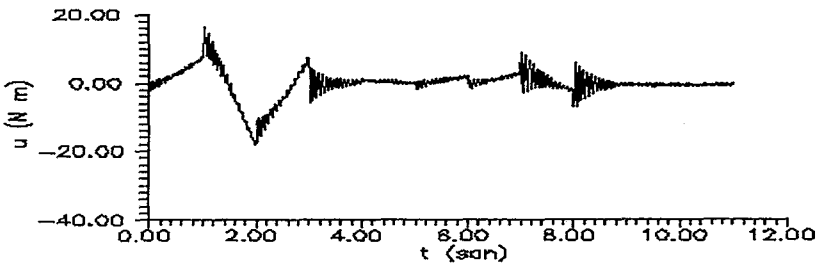
$$K = 200, \quad T_i = 0.4, \quad T_d = 0.1, \quad T_t = 0.1, \quad K_h = 10$$

alınarak, $h = 0.01san$ ile yapılmıştır. Referans girişi olarak Eşitlik 2.20 ile verilen $r(t)$ kullanılmıştır. Denetim işareti üzerinde Eşitlik 2.18 ile verilen gürültü eklenmiştir. Simülasyon sonucu elde edilen denetim işareti Şekil 7.2'de, referans ile sistem çıkışı arasındaki fark, $e(t)$, ise Şekil 7.3'de gösterilmiştir. Şekil 7.4'de hataların karelerinin toplamı gösterilmiştir.

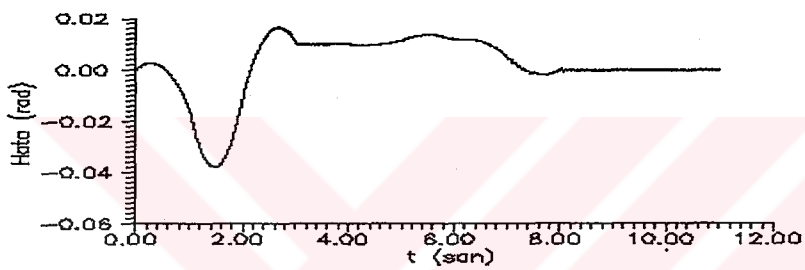
Eşitlik 7.5'den ileri besleme kısmı kaldırılırsa

$$u(kh) = P + I + D$$

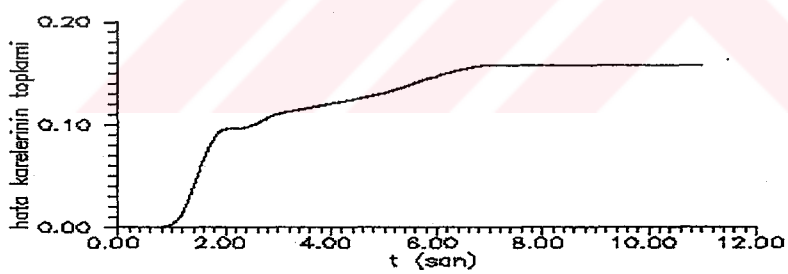
elde edilir. Bu regülatörün, yukarıda verilen parametrelerle simülasyonu yapılmış, ve sonuçlar Şekil 7.5, 7.6 ve 7.7'de gösterilmiştir.



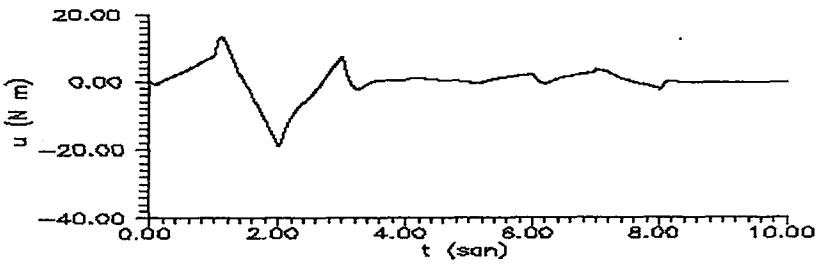
Şekil 7.2 Eşitlik 7.5 ile verilen servo sistemde denetim işareti.



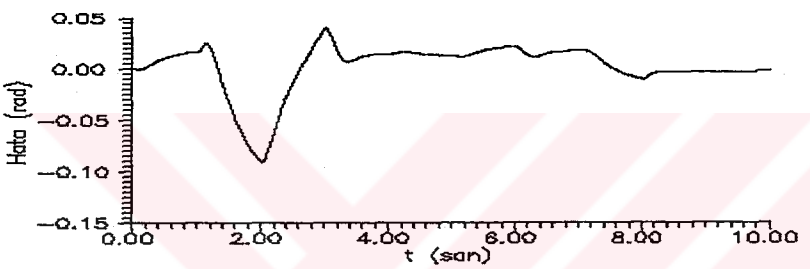
Şekil 7.3 Eşitlik 7.5 ile verilen servo sistemin hatası.



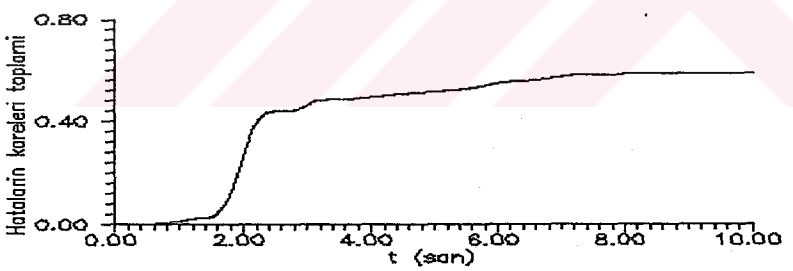
Şekil 7.4 Eşitlik 7.5 ile verilen servo sistemde hataların karelerinin toplamı.



Şekil 7.5 Referans izleyen PID regülatörde denetim işareti.



Şekil 7.6 Referans izleyen PID regülatörün hatası.



Şekil 7.7 Referans izleyen PID regülatör için hataların karelerinin toplamı.

7.4 LQG Geri Besleme

İleri beslemeli servo denetleyicisi için, geri besleme yasası olarak Bölüm 5'de verilen LQG denetim yasası kullanılmıştır.

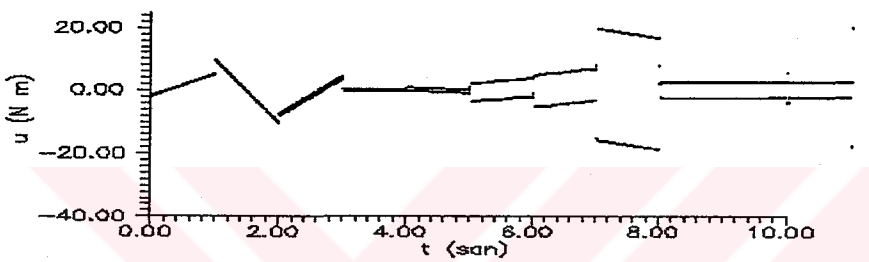
İleri besleme işareti, parametreleri zamanla değişmeyen çift entegratör kullanılarak elde edilmiştir. Bu ileri besleme yasası Eşitlik 7.4 ile verilmiştir.

Geri besleme işaretini bulmakta kullanılacak LQG regülatörün çözümü Eşitlik 5.24, 5.25, 5.26 ve 5.27 ile verilmiştir.

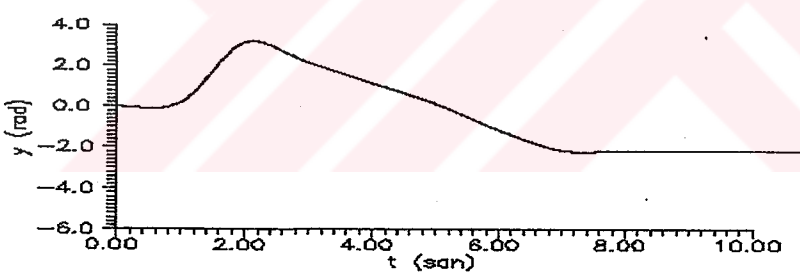
Simülasyon 7.2

Bölüm 2.2'de verilen parametreleri zamanla değişen sistemin simülasyonu yapılmıştır. Geri besleme işareti Simülasyon 5.2'de kullanılan LQG regülatörü ile elde edilmiştir. İleri besleme işareti, Eşitlik 7.4'den bulunmuştur.

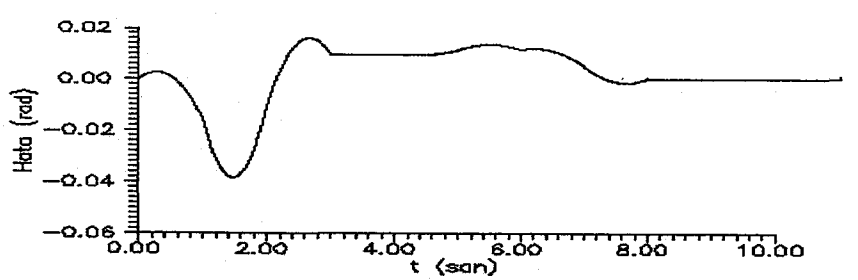
Simülasyon sonucu elde edilen denetim işareti Şekil 7.8'de, sistem çıkışı Şekil 7.9'da ve referans işareti ile sistem çıkışı arasındaki fark Şekil 7.10'da gösterilmiştir.



Şekil 7.8 Referans izleyen LQG regülatörde denetim işareti.



Şekil 7.9 Referans izleyen LQG regülatör çıkışı.



Şekil 7.10 Referans izleyen LQG regülatörün hatası.

7.5 Kendini Ayarlayan Geri Besleme

Şekil 7.1'de gösterilen iki serbestlikli yapı içinde, geri besleme, Bölüm 6'da verilen kendini ayarlayan denetleyici kullanılarak gerçekleştirilmiştir.

Geri beslemeli sistemin transfer fonksiyonu referans modelin transfer fonksiyonuna eşit alınabilir. Eşitlik 2.12, referans model için

$$y(k+2) + a_{m1}y(k+1) + a_{m2}y(k) = b_{m1}u(k+1) + b_{m2}u(k) \tag{7.6}$$

şeklinde yazılabilir. $k+2$ anında çıkışın $r(k+2)$ olması istendiği takdirde, Eşitlik 7.6, $y(k+2)$ yerine $r(k+2)$ yazılarak tekrar düzenlenirse

$$u_m(k) = \frac{1}{b_{m1}}[r(k+1) + a_{m1}y(k) + a_{m2}y(k-1) - b_{m2}u_m(k-1)] \tag{7.7}$$

elde edilir. Burada, $y(k)$ ölçülen sistem çıkışıdır. Eşitlik 7.7'nin nedensel olmadığına dikkat etmek gerekir. k anında $u_m(k)$ hesabının yapılması için bir sonraki örnekleme periyodundaki referans girişine ihtiyaç duyulmaktadır. Birçok durumda, referans işareti önceden bilindiği için bu bir sorun yaratmaz. Ancak, referans işareti önceden bilinmiyorsa, Kalman süzgeci gibi kestirim mekanizmaları kullanılarak ileriki referans işareti kestirilmelidir.

Eşitlik 7.7'da kullanılan b_{m1} ve b_{m2}'nin zamanla değiştiğine dikkat edilmelidir. Sistem sıfırları, referans model sıfırları ile yok edildiği için, ileri besleme işaretinin üretilmesi sırasında b_{m1} ve b_{m2} yerine en küçük kareler yöntemi ile yapılan kestirimlerden elde edilen $\hat{b}_1$ ve $\hat{b}_2$ konulmalıdır. Böylece, ileri besleme yasası :

$$u_m(k) = \frac{1}{\hat{b}_1}\left[r(k+1) + a_{m1}y(k) + a_{m2}y(k-1) - \hat{b}_2 u_m(k-1)\right] \tag{7.8}$$

olarak elde edilir. Burada, $u_m(k)$ ileri besleme işareti, $r(k)$ izlenmesi istenen referans, a_{m1} ve a_{m2} referans modelin parametreleri, $\hat{b}_1$ ve $\hat{b}_2$ en küçük kareler yöntemi ile (veya başka bir kestirim yöntemi ile) kestirilen gerçek sistemin parametreleri ve $y(k)$ ölçülen sistem çıkışıdır.

Simülasyon 7.3

Eşitlik 2.2 ile verilen parametreleri zamanla değişen sistemin bilgisayar simülasyonu yapılmıştır. İzlenmeye çalışılacak referans işareti Eşitlik 2.20 ile verilmiştir.

Denetleyici parametreleri Simülasyon 6.1 dekine benzer şekilde :

$$a_0 = 0; \quad w_{m0} = 1; \quad \xi_m = 0.7$$

olarak seçilmiştir[8]. Referans modelin ayrık zaman parametreleri, Simülasyon 6.1'de verilmiştir. Simülasyon 6.1'de elde edilen, kendini uyarlayan denetim yasası kullanılarak geri besleme işareti oluşturulmuştur. İleri besleme işareti Eşitlik 7.8 ile elde edilmiştir.

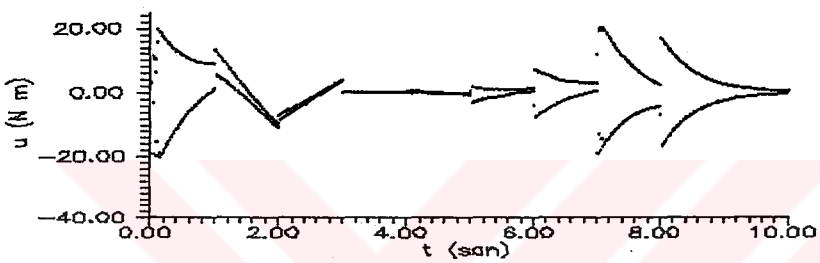
Denetim işlemi başlamadan önce, 100 adım boyunca sistem 0 ortalamalı 0.01 varyanslı beyaz gürültü ile tahrik edilerek en küçük kareler yöntemi ile parametre kestiriminde kullanılan P ağırlık matrisinin ilk değerinin, başlangıç anında gerçek parametrelere yakın değerler vermesi sağlanmıştır. Bu sırada, sistemin ataleti $J_e = 1$ olarak sabit tutulmuştur. Denetim işareti başladığı anda sistem ataleti $J_e = 0.5$ olduğundan $P(0)$ matrisi bu anda hatalıdır. Şekil 7.12'de, $t = 0$ civarında bir kaç noktada hatanın göreceli olarak yüksek olması buna bağlanabilir.

Şekil 7.11'de simülasyon sonucu elde edilen denetim işareti ve Şekil 7.12'de sistem çıkışı verilmiştir. Sistem çıkışının, izlenmesi istenen referans işaretinden farkı Şekil 7.13'de gösterilmiştir.

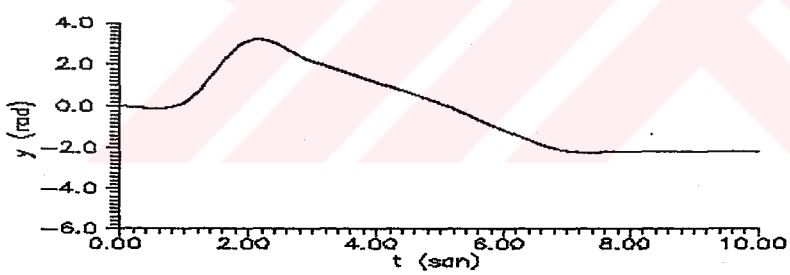
Şekil 7.11'de denetim işaretinin salınımının çok fazla olduğu görülmektedir. Bunun nedeni, kapalı çevrim içinde sistem kutuplarının yok edilmiş olmasıdır. Bölüm 6.2'de verilen kutup yerleştirme problemi, sistem kutupları yok edilmeden çözülürse, bu salınım olmaz. Şekil 7.12'de hatanın çok az olduğu görülmektedir. Ancak bu denetleyicinin, pratikte uygulanabilmesi için sistemin çok salınan denetim işaretinden etkilenmeyecek dinamik özelliklere sahip olması gerekmektedir.

7.6 Sonuç

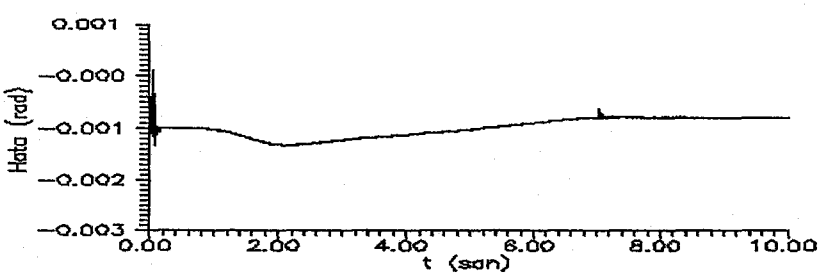
Bu bölümde servo denetim probleminin çözüm yöntemlerinden biri olan ileri beslemeli iki serbestlikli yapı verilmiştir. Bu yapı içinde, önceki bölümlerde elde edilen regülatör tipindeki denetleyiciler, kapalı çevrimi oluşturmak üzere kullanılmışlardır.



Şekil 7.11 Referans izleyen kendini ayarlayan regülatörde denetim işareti.



Şekil 7.12 Referans izleyen kendini ayarlayan regülatör çıkışı.



Şekil 7.13 Referans izleyen kendini ayarlayan regülatörün hatası.

BÖLÜM 8

SONUÇLAR

Bu çalışmada servo denetim sistemlerinden bazılarının performanslarının karşılaştırılması için, servo denetim yasaları verilerek simülasyonları yapılmıştır.

Servo denetim sistemi olarak iki serbestlikli yapı kullanılmıştır. Bu yapının avantajı, servo ve regülatör kısımlarının birbirinden ayrılmış olmasıdır. Bu sebeple, iki serbestlikli servo denetleyicinin servo ve regülatör özellikleri ayrı ayrı belirtilebilinir.

Yapılan bilgisayar simülasyonları sonucunda, iki serbestlikli yapı içinde kullanılan kendini ayarlayan denetleyicinin verilen referans işaretini çok az hata ile takip ettiği görülmüştür. Bu servo denetleyicinin simülasyonlar sonucu elde edilen hatası, pratikte istenen hata sınırının çok altında kalmıştır. Ancak, bu yöntemle elde edilen denetim işareti çok fazla salınmaktadır. Bu denetim işareti, mekanik sistemlerde rezonansa ve yorulmaya sebep olacaktır. Bu sebeple, elde edilen kendini uyarlayan servo denetleyicisi bu haliyle kullanılamaz. Geri besleme yasası üretilirken sistem sıfırlarını yok etmeyecek yöntemler kullanılırsa denetim işaretinin salınması azalacaktır, ancak sistemin hatasının da artacağı anlaşılmıştır.

İki serbestlikli servo denetim yapısı içinde, LQG denetim yasası kullanılarak elde edilen sonuçlar, PID denetim yasası ile elde edilenlere benzemektedir. Bölüm 7'de verilen simülasyonlarda PID servo denetleyici ile LQ servo denetleyicilerin hataları aynı olmasına rağmen, ürettikleri denetim işaretleri oldukça farklı oluşmaktadır.

KAYNAKLAR

[1] FU, L., Robust Adaptive Decentralized Control of Robot Manipulators, IEEE Transactions on Automatic Control, Vol. 37, No. 2, February 1992.

[2] OGATA, K., Modern Control Engineering,1990 Prentice Hall, Inc.

[3] SAGE, A.S., MELSA, J.L., System Identification, 1971 Academic Press, Inc.

[4] OGATA, K., Discrete Time Control Systems, 1987 Prentice Hall, Inc.

[5] NAIK, S.M., Robust Continuous Time Adaptive Control By Parameter Projection,IEEE Transactions on Automatic Control, Vol. 37, No. 2, February 1992.

[6] CHALAM, V.V., Adaptive Control Systems, 1987 Marcel Dekker, Inc.

[7] ASTROM, K.J., Computer Controlled Systems,1990 Prentice Hall, Inc.

[8] NARENDA, S.K., A Combined Direct, Indirect and Variable Structure Method For Robust Adaptive Control, IEEE Transactions on Automatic Control, Vol. 37, No. 2, February 1992.

EK

BİLGİSAYAR PROGRAMLARI

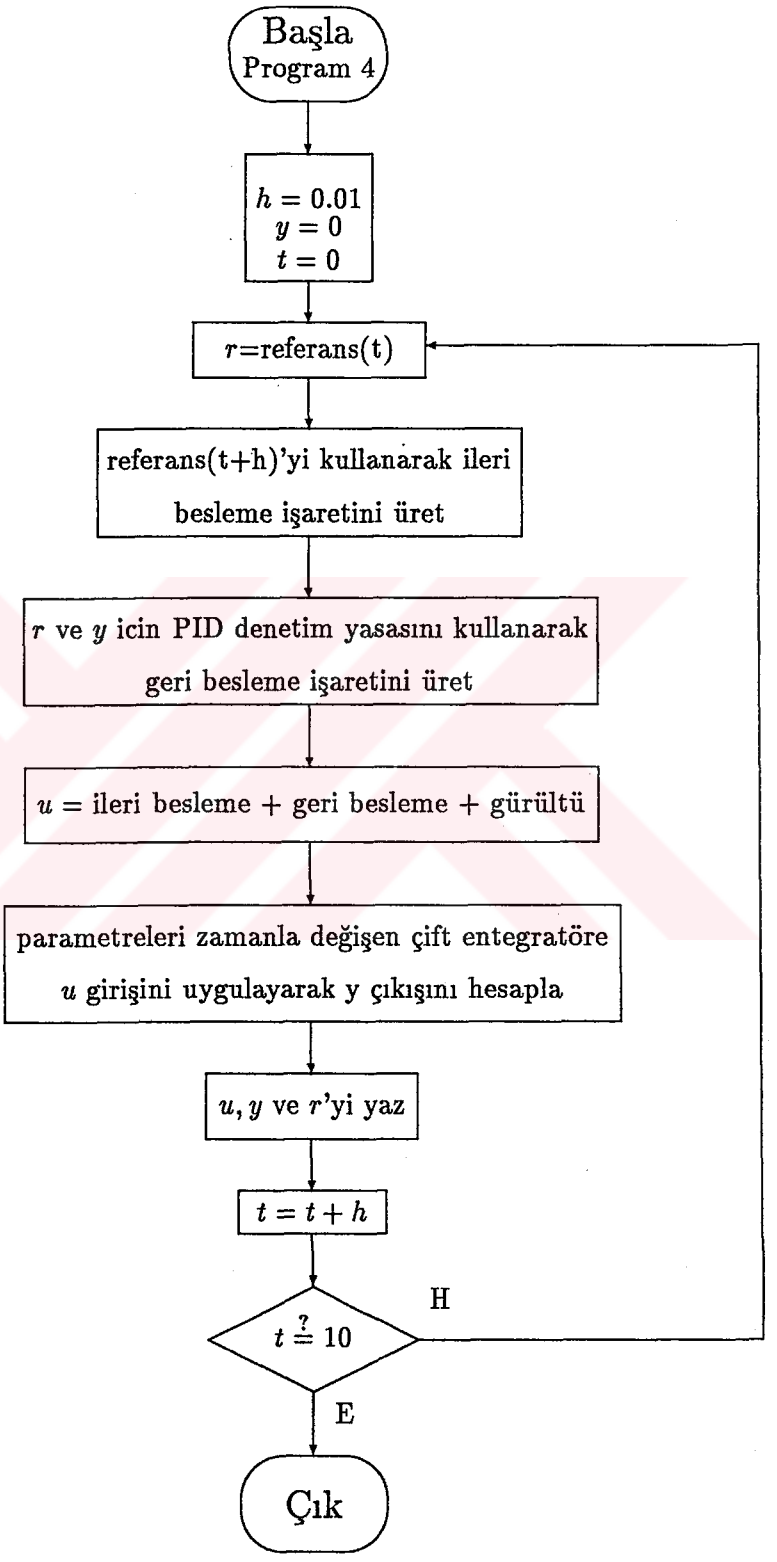
Programların Açıklamaları

Program 1, Bölüm 2'de verilen çift entegratör, katsayıları zamana göre değişen çift entegratör ve ikinci derece sönümlü salınım sistemlerinin cevaplarını hesaplamaktadır. Ayrıca referans işareti, referans işaretinin türevi ve beyaz gürültü büyüklükleri de bu bilgisayar programının içinde hesaplanmıştır. Bu bilgisayar programının içinde tanımlanan fonksiyonlar, tüm simülasyonlarda ortak olarak kullanılmıştır.

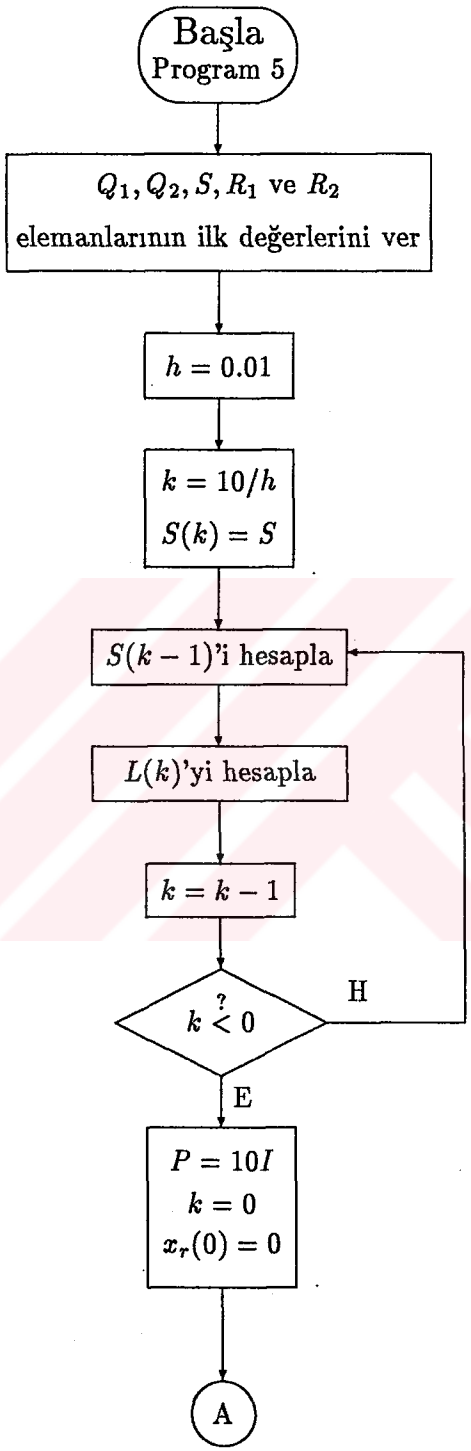
Program 2, 2 boyutlu noktalardan, kendileri ve birinci türevleri sürekli olan 3. derece polinomlar geçirmektedir. Bu bilgisayar programında verilen fonksiyonlar, Program 1 tarafından referans işaretini hesaplamak üzere çağrılmaktadır.

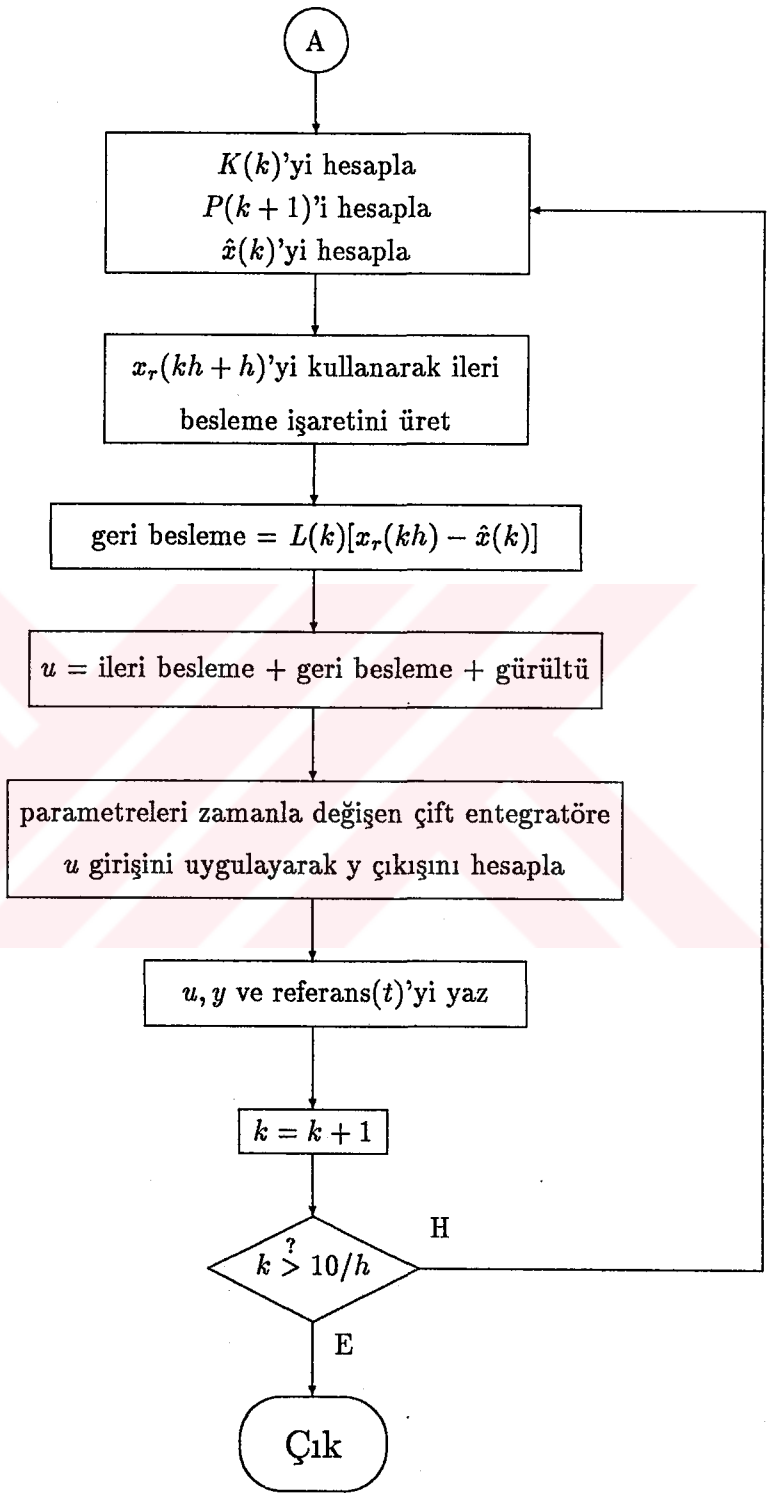
Program 3 ise, Bölüm 4'de verilen salınımı önlenmiş PID denetleyici için denetim işaretini hesaplamaktadır.

Simülasyon 7.1, 7.2 ve 7.3, sırasıyla, Program 4, 5 ve 6 ile gerçeklenmiştir. Şekil 1, 2 ve 3 ile, 4, 5 ve 6 numaralı bilgisayar programlarının akış diyagramları verilmiştir.

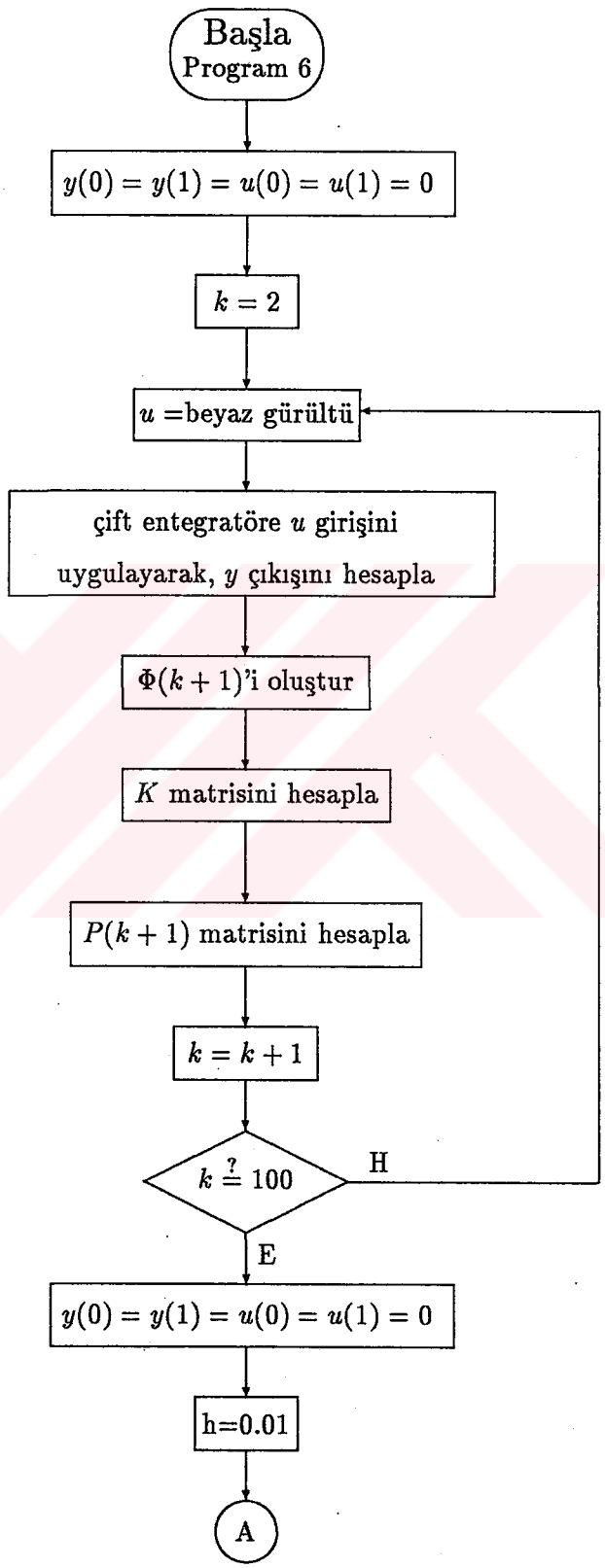


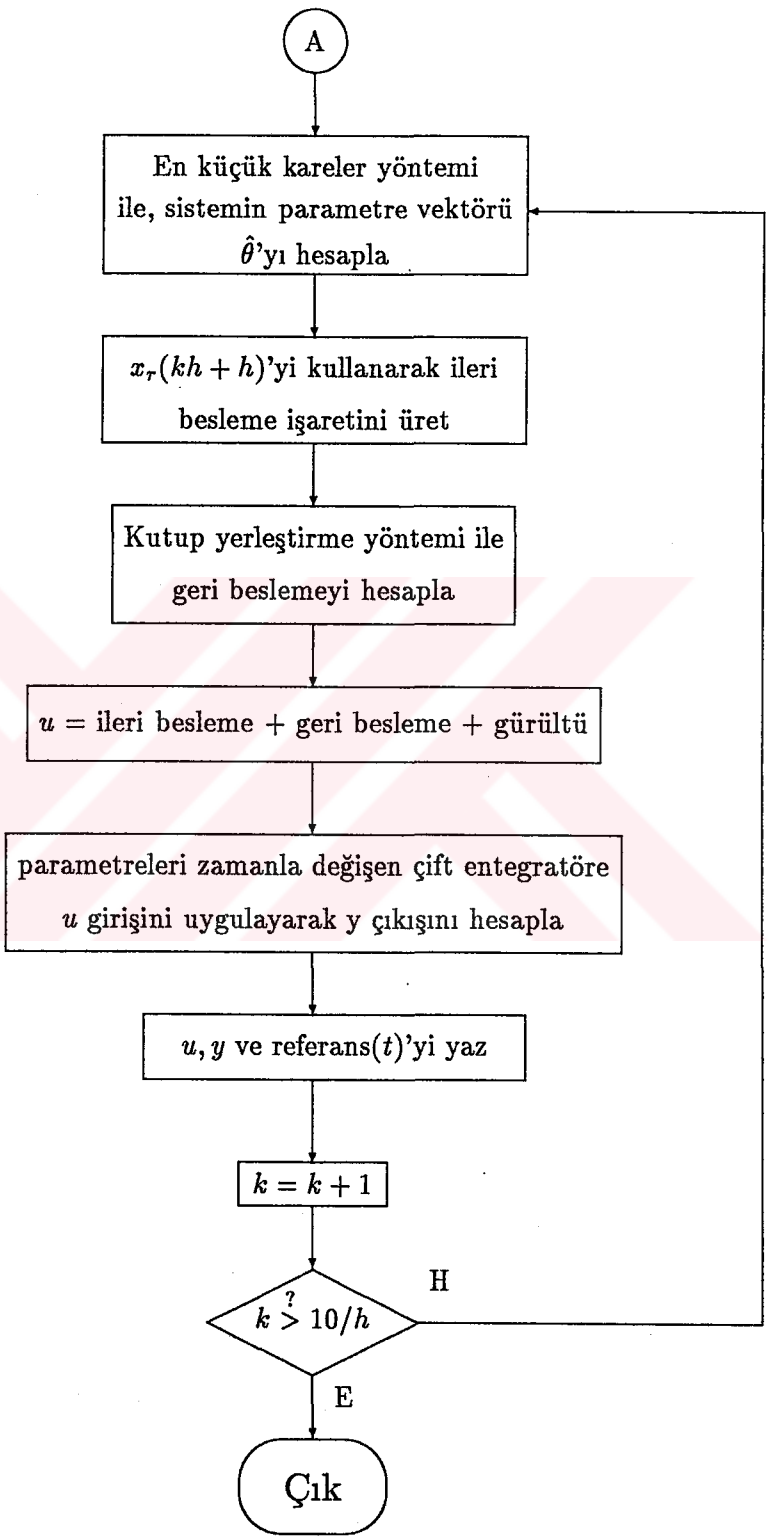
Şekil 1 PID geri beslemesi kullanılmış, iki serbestlikli denetleyicinin akış diyagramı.


Başla
Program 5
Q_1, Q_2, S, R_1 ve R_2
elemanlarının ilk değerlerini ver
$h = 0.01$
$k = 10/h$
$S(k) = S$
$S(k-1)$'i hesapla
$L(k)$'yi hesapla
$k = k - 1$
$k \overset{?}{<} 0$
H
E
$P = 10I$
$k = 0$
$x_r(0) = 0$
A



Şekil 2 LQG geri beslemesi kullanılmış, iki serbestlikli denetleyicinin akış diyagramı.


Başla
Program 6
$y(0) = y(1) = u(0) = u(1) = 0$
$k = 2$
u =beyaz gürültü
çift entegratöre u girişini
uygulayarak, y çıkışını hesapla
$\Phi(k+1)$'i oluştur
K matrisini hesapla
$P(k+1)$ matrisini hesapla
$k = k + 1$
$k \stackrel{?}{=} 100$
H
E
$y(0) = y(1) = u(0) = u(1) = 0$
h=0.01
A



Şekil 3 STR geri beslemesi kullanılmış, iki serbestlikli denetleyicinin akış diyagramı.

Bilgisayar Programları

```
// Program 1
/* Sistem cevaplarinin hesaplanmasi
 */
#ifdef _MY_SYS_H_
#else
#define _MY_SYS_H_
#include <stdio.h>
#include <dos.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include "my_sys.h"
#include "ls.h"
#include "sayfa20.h"

#define t_max 11   // saniye cinsinden en son zaman adimi
#define h 0.01     // saniye cinsinden ornekleme periyodu
#define u_high 20
#define u_low (-20)
int ref_0=1;

struct Kubik_Teta r_t[11];

/*
   Surekli zaman modeli G(s)=  w0^2 / (s^2 + 2 ksi w0 s + w0^2) olan
   sistemin ayrik zaman moelindeki a1,a2,b1,b2 katsayilari hesaplaniyor.
*/
void ABp(double w0,double ksi,double hh,double *a1,double *a2,double *b1,double *b2)
{
double alfa,beta,gama,w;
  if (ksi<=1)
  {
    w = w0*sqrt(1-ksi*ksi);
    beta = cos(w*hh);  gama = sin(w*hh); alfa = exp(-ksi*w0*hh);

    *b1=1-alfa*(beta+ksi*w0/w*gama);
    *b2=alfa*alfa+alfa*(ksi*w0/w*gama-beta);
    *a1=-2*alfa*beta;
    *a2=alfa*alfa;
  }
  else *a2=*a1=*b1=*b2=0;
}

/*  2. derece sistem cevabini simule ediyor.
 *  Sistemin baslangic kosullarini sifir kabul ediyor.
 */
double sys3_y_1=0,sys3_dydt=0; // y[k-1], y[k-2]
double sys3_y(double u)
{
```

```
  sys3_y_1= sys3_y_1 + h* sys3_dydt + h*h/2*u;
  sys3_dydt = sys3_dydt + h*u;
  return sys3_y_1;
}

/*  Double Integrator cevabini simule ediyor.
 *  Sistemin baslangic kosullarini sifir kabul ediyor.
 */
double sys2_w=1, sys2_ksi=0.7;
double sys2_a1,sys2_a2,sys2_b1,sys2_b2;
double sys2_y_1=0,sys2_y_2=0; // y[k-1], y[k-2]
double sys2_u_1=0, sys2_u_2=0;
int sys2_0=1;
double sys2_y(double u)
{
double y;
  if (sys2_0)
  {
    ABp(sys2_w,sys2_ksi,h, &sys2_a1, &sys2_a2, &sys2_b1, &sys2_b2);
    sys2_0=0;
  }
  y= -sys2_a1*sys2_y_1 -sys2_a2*sys2_y_2
     +sys2_b1*sys2_u_1 +sys2_b2*sys2_u_2;
  sys2_y_2=sys2_y_1;  sys2_y_1=y;
  sys2_u_2=sys2_u_1;  sys2_u_1=u;
  return y;
}

double system_J(double t)
{
double j;
  if ( (t>t_max+h) || (t<0) ) system_hata(1);
  if (t<3) j=0.5+t*0.5/3.0;  else
  if (t<7) j=1.0;            else
   j=1.0-(t-7)*0.5/3.0;
  if (j<0.49) j=0.5;
  return 1;
}

/*  Sistem cevabini hesapliyor
*/
#define y_quan 50000     /* -3.14<y<3.14 araliginda y 500*100 farkli deger */
 /*  alabiecek */
double t_eski=18920.1231;
double system_y_1=0,system_y_2=0;
double system_dydt=0;
double system_y(double t, double u)
{
double y;
   if ( (t_eski-0.00001<t) && (t<t_eski+0.00001) )
   {
     y=system_y_1;
```

```
    }
    else
    {
      t_eski=t;
      y= u/system_J(t) *h*h/2.0 + system_dydt * h + system_y_1;
      system_y_1=y;
      system_dydt=system_dydt+u/system_J(t)*h;
    }
    y= (double)( (long)(y*y_quan)/(double)y_quan );
    return y;
}


/* Referans isareti hesaplaniyor.
 */
struct nokta REF_X[11];
void ref_kat()
{
int i,i_max;
    setcolor(11);
    i_max=10;
    i=0;
    REF_X[i].t=i; REF_X[i].teta=0;
    i=1;
    REF_X[i].t=i ; REF_X[i].teta=0.2;
    i=2;
    REF_X[i].t=i ; REF_X[i].teta=3.14;
    i=3;
    REF_X[i].t=i ; REF_X[i].teta=2.14;
    i=4;
    REF_X[i].t=i ; REF_X[i].teta=1.14;
    i=5;
    REF_X[i].t=i ; REF_X[i].teta=0.14;
    i=6;
    REF_X[i].t=i ; REF_X[i].teta=-1.14;
    i=7;
    REF_X[i].t=i ; REF_X[i].teta=-2.14;
    i=8;
    REF_X[i].t=i ; REF_X[i].teta=-2.14;
    i=9;
    REF_X[i].t=i ; REF_X[i].teta=-2.14;
    Katsayilari_Hesapla(i_max,REF_X,r_t);
}

double reference(double t)
{
  if (ref_0)   // eger ilk defa calisiyorsa katsayilari hesaplasin
  {
    ref_kat();
    ref_0=0;
  }
  return Tetayi_Hesapla(t,r_t);
```

```
}

double ref_turev(double t)
{
  return Teta_Hizini_Hesapla(t,r_t);
}

struct f_b { double f11,f12,f13,f14,f15,
    f21,f22,f23,f24,f25,
    f31,f32,f33,f34,f35;
     double a11,a12,b11,b12,
    a21,a22,b21,b22,
    a31,a32,b31,b32;
   };

double beyaz_0=1;
double beyaz(double r)
{
  if(beyaz_0) { randomize(); beyaz_0=0; }
  return 2*r*(0.5-random(1000)/1000.0);
}

#endif

// Program 2
/* Verilen noktalardan kendisi ve turevi surekli olan 3.
 * derece egriler geciriliyor
 */
#if defined _SAYFA20_H
#else
#define _SAYFA20_H

#include "ters.h";


/*  t_i,Q_i noktalari verildiginde, bu noktalardan gecen
 * kendisi ve turevi surekli kubik fonksiyonlari hesaplayan rutinler
 */

struct nokta { float t,teta; };

struct Kubik_Teta {
  float t_baslangic; // Kubik fonksiyonun baslama zamani
  float t_son      ; // Kubik fonksiyonun bitis   zamani
  float a[4]     ; // teta = a[3]*t^3+a[2]*t^2+a[1]*t+a[0]
};

float Tetayi_Hesapla(float t, struct Kubik_Teta *r_t)
{ int i;
  i=0;
  do i++; while ( (r_t[i].t_baslangic<t) && (i<1000) );
  i--;
```

```
  if ( t>r_t[i].t_son  ) t=r_t[i].t_son;
    return (r_t[i].a[3]*t*t*t+r_t[i].a[2]*t*t+r_t[i].a[1]*t+r_t[i].a[0]);
}

float Teta_Hizini_Hesapla(float t, struct Kubik_Teta *r_t)
{ int i;
  i=0;
  do i++; while ( (r_t[i].t_baslangic<t) && (i<1000) );
  i--;
  if (t>r_t[i].t_son )  return 0;
  else
  return (3*r_t[i].a[3]*t*t+2*r_t[i].a[2]*t+r_t[i].a[1]);
}

void Katsayilari_Hesapla( int i_max, struct nokta *p,
  struct Kubik_Teta *r_t)
/*  i_max tane (p[i].t,p[i].teta) noktasindan 3. derece egriler gecirip
 * bu egrilerin katsayilarini, baslagic ve bitis zamanlarini r_t'ye
 * yazacak olan alt program.
 */
{
float V[20]; // i'inci noktadaki hiz.
float ma[1+4][1+4];
int i;
float a,b;

  // Hizlarin hesabi. Bakiniz Tez notlari Sayfa 16;
  V[0]=V[i_max-1]=0;
  if (i_max>2)
  for (i=1;i<(i_max-1);i++)
  {
    ma[1+0][1+0] = p[i-1].t*p[i-1].t; ma[1+0][1+1]=p[i-1].t; ma[1+0][1+2]=1;
    ma[1+1][1+0] = p[i  ].t*p[i  ].t; ma[1+1][1+1]=p[i  ].t; ma[1+1][1+2]=1;
    ma[1+2][1+0] = p[i+1].t*p[i+1].t; ma[1+2][1+1]=p[i+1].t; ma[1+2][1+2]=1;
    TersAl(*ma,(int)3,(int)4);
    a=p[i-1].teta*ma[1+0][1+0]+p[i].teta*ma[1+0][1+1]+p[i+1].teta*ma[1+0][1+2];
    b=p[i-1].teta*ma[1+1][1+0]+p[i].teta*ma[1+1][1+1]+p[i+1].teta*ma[1+1][1+2];
    V[i] = 2*a*p[i].t+b;
  }
  for (i=0;i<i_max;i++)
  {
    r_t[i].t_baslangic = p[i].t;
    r_t[i].t_son       = p[i+1].t;
  }
  i=i_max-1;
  r_t[i+1].t_baslangic=1e200; // Son kubik oldugunu belirtmek icin sifir yazdik.
  r_t[i].a[3]=r_t[i].a[2]=r_t[i].a[1]=0; r_t[i].a[0]=p[i].teta;//y=teta line.

  for (i=0;i<i_max-1;i++)
  {
    ma[1+0][1+0]=p[i].t*p[i].t*p[i].t; ma[1+0][1+1]=p[i].t*p[i].t;
      ma[1+0][1+2]=p[i].t; ma[1+0][1+3]=1;
```

```
    ma[1+1][1+0]=p[i+1].t*p[i+1].t*p[i+1].t; ma[1+1][1+1]=p[i+1].t*p[i+1].t;
      ma[1+1][1+2]=p[i+1].t; ma[1+1][1+3]=1;
    ma[1+2][1+0]=3*p[i].t*p[i].t; ma[1+2][1+1]=2*p[i].t;
      ma[1+2][1+2]=1; ma[1+2][1+3]=0;
    ma[1+3][1+0]=3*p[i+1].t*p[i+1].t; ma[1+3][1+1]=2*p[i+1].t;
      ma[1+3][1+2]=1; ma[1+3][1+3]=0;
    TersAl(*ma,(int)4,(int)4);
    r_t[i].a[3]=p[i].teta*ma[1+0][1+0]+p[i+1].teta*ma[1+0][1+1]+
V[i]*ma[1+0][1+2]+V[i+1]*ma[1+0][1+3];
    r_t[i].a[2]=p[i].teta*ma[1+1][1+0]+p[i+1].teta*ma[1+1][1+1]+
                V[i]*ma[1+1][1+2]+V[i+1]*ma[1+1][1+3];
    r_t[i].a[1]=p[i].teta*ma[1+2][1+0]+p[i+1].teta*ma[1+2][1+1]+
                V[i]*ma[1+2][1+2]+V[i+1]*ma[1+2][1+3];
    r_t[i].a[0]=p[i].teta*ma[1+3][1+0]+p[i+1].teta*ma[1+3][1+1]+
V[i]*ma[1+3][1+2]+V[i+1]*ma[1+3][1+3];
  }
}

#endif

// Program 3
/* PID denetleyici
 */
#ifdef _PID_H_
#else
#define _PID_H_

#define pid_K 200   /* regulator gain */
#define pid_Ti 4   /* integral time */
#define pid_Td 0.01  /* derivative time */
#define pid_Tt 0.1  /* reset time */
#define pid_N 10  /* maximum derivative gain */
#define pid_b 1   /* fraction of set point in derivative term */
#define pid_bi (pid_K*h/pid_Ti)
#define pid_ar (h/pid_Tt)
#define pid_bd (pid_K*pid_N*pid_Td/(pid_Td+pid_N*h))
#define pid_ad (pid_Td/(pid_Td+pid_N*h))

double pid_P, pid_I=0, pid_D=0, pid_y_1=0, pid_r_1=0;

double PID(double t, double y, double r)
{
double u,v;
  t=t;                // derleyicinin uyari vermemesi icin
  pid_P=pid_K*(pid_b*r-y);
  pid_D=pid_ad*pid_D+pid_bd*( (r-y) - (pid_r_1-pid_y_1) );
  pid_y_1=y;
  pid_r_1=r;
  v=pid_P+pid_I+pid_D;
  u=v;
  if (u<u_low)  u=u_low;
  if (u>u_high) u=u_high;
```

```
  pid_I=pid_I+pid_bi*(r-y)+pid_ar*(u-v);
  return u;
}
#endif

// Program 4

/* TDOF icinde PID geri besleme kullanilarak yapilan simulasyon
 */
#include "my_sys.h"
#include "pid.h"

double e=0,e2=0;
double t,u,um,ume1=0,r,y,ye1=0;

void main()
{
FILE *stream;

  stream=fopen("s7_1.dat","w");

  t=0; y=0;
  do{
     r=reference(t);
     um=2/h/h*( reference(t+h) - y - ref_turev(t)*h);
    ye1=y;
    ume1=um;
     u=PID(t,y,r) +um +beyaz(0.15);
     y=system_y(t,u+beyaz(0.15));
     e=r-y;
     e2+=e*e;
    fprintf(stream,"%7.4lf %7.4lf %7.4lf  %7.4lf %7.4lf %7.4lf\n",
     t,     e,     y,      u,     r,  e2  );
     t+=h;
   } while( (t<t_max) && !kbhit() );
   fclose(stream);

   getch();
}

// Program 5

/* TDOF icinde LQG kullanilarak yapilan simulasyon
 */
#include "my_sys.h"

double L[2][1100];
void main()
{
double phi[2][2],gama[2],a[2][2],b[2],psi[2][2],aa[2][2],dum[2][2],q1[2][2];
double q2,s[2][2];
double dum0,dum1[2],dum2[2][2],dum3[2][2];
```

```
double je=1;
int i,j,m,ii;
double nf,hh;
int k;
int kk=1000;
double ye1=0,dydt,Td=1,Kh=10;
double r1[2][2],r2,K[2],p[2][2];
double xs[2];
double c[2];
double ref,reft;
double um,ume1=0;
FILE *stream;
double t=0,y=0,r,u,e2,e;

   a[0][0] = 0;        a[0][1] = 1;
   a[1][0] = -1;        a[1][1] = -1.4;
   q1[0][0] = 2;       q1[0][1] = 0;
   q1[1][0] = 0;       q1[1][1] = 1;
   q2=0.5;
   b[0] = 0;           b[1] = 1/je;
   psi[0][0] = h;        psi[0][1] = 0;
   psi[1][0] = 0;        psi[1][1] = h;
   s[0][0]=1;            s[0][1]=0;
   s[1][0]=0;            s[1][1]=1;
   r1[0][0]=0.1;        r1[0][1]=0.01;
   r1[1][0]=0.01;       r1[1][1]=0.1;
   r2=0.1;

   for (i=0;i<2;i++) for (j=0;j<2;j++) aa[i][j]=a[i][j];
   nf=1; // n faktorial
   hh=h;
   for (k=1;k<10;k++)
   {
       nf=nf*(k+1);
       hh=hh*h;
       for (i=0;i<2;i++) for (j=0;j<2;j++) psi[i][j]+=hh/nf*aa[i][j];
       for (i=0;i<2;i++)    for (j=0;j<2;j++)
       {
 dum[i][j]=0;
 for(m=0;m<2;m++) dum[i][j]+=aa[i][m]*a[m][j];
}
       for (i=0;i<2;i++) for (j=0;j<2;j++)
aa[i][j]=dum[i][j];

   }
      for (i=0;i<2;i++)    for (j=0;j<2;j++)
       {
 if (i==j) phi[i][j]=1; else phi[i][j]=0;
 for(m=0;m<2;m++) phi[i][j]+=a[i][m]*psi[m][j];
}
gama[0]=psi[0][0]*b[0]+psi[0][1]*b[1];
gama[1]=psi[1][0]*b[0]+psi[1][1]*b[1];
```

```
     for (k=0;k<=1000;k++) L[0][k]=L[1][k]=1;
  // s+1 hesabi
      for (k=1;k<=kk;k++)
      {
// dum0=1/(q2 + gama transpoz s gama)
dum1[0]=s[0][0]*gama[0]+s[0][1]*gama[1];
dum1[1]=s[1][0]*gama[0]+s[1][1]*gama[1];
dum0=q2+gama[0]*dum1[0]+gama[1]*dum1[1];
dum0=1.0/dum0;

// dum1= gama transpoz s phi
for (i=0;i<2;i++) for (j=0;j<2;j++)
{
  dum[i][j]=0;
  for (m=0;m<2;m++) dum[i][j]+=s[i][m]*phi[m][j];
}
dum1[0]=gama[0]*dum[0][0]+gama[1]*dum[1][0];
dum1[1]=gama[0]*dum[0][1]+gama[1]*dum[1][1];

//dum = dum0 * gama * dum1 transpose
for (i=0;i<2;i++) for (j=0;j<2;j++)
  dum[i][j]=dum0*gama[i]*gama[j];

  //dum=(phi-dum)
for (i=0;i<2;i++) for (j=0;j<2;j++)
  dum[i][j]=phi[i][j]-dum[i][j];

// dum = dum transpoz
dum0=dum[0][1]; dum[0][1]=dum[1][0]; dum[1][0]=dum0;

//dum2 = s phi
for (i=0;i<2;i++) for (j=0;j<2;j++)
{
  dum2[i][j]=0;
  for (m=0;m<2;m++) dum2[i][j]+=s[i][m]*phi[m][j];
}

// dum3 = dum * dum2
for (i=0;i<2;i++) for (j=0;j<2;j++)
{
  dum3[i][j]=0;
  for (m=0;m<2;m++) dum3[i][j]+=dum[i][m]*dum2[m][j];
}

//s+1 = dum3+q1
for (i=0;i<2;i++) for (j=0;j<2;j++)
  s[i][j]=dum3[i][j]+q1[i][j];

// dum0=1/(q2 + gama transpoz s gama)
dum1[0]=s[0][0]*gama[0]+s[0][1]*gama[1];
dum1[1]=s[1][0]*gama[0]+s[1][1]*gama[1];
```

```
dum0=q2+gama[0]*dum1[0]+gama[1]*dum1[1];
dum0=1.0/dum0;

// dum1= gama transpoz s phi
for (i=0;i<2;i++) for (j=0;j<2;j++)
{
  dum[i][j]=0;
  for (m=0;m<2;m++) dum[i][j]+=s[i][m]*phi[m][j];
}
dum1[0]=gama[0]*dum[0][0]+gama[1]*dum[1][0];
dum1[1]=gama[0]*dum[0][1]+gama[1]*dum[1][1];

// l=dum0 dum1
L[0][kk-k]=dum0*dum1[0];
L[1][kk-k]=dum0*dum1[1];
      }

 stream=fopen("s7_11.dat","w");

 P[0][0]=10;   P[0][1]=0;
 P[1][0]=0; P[1][1]=10;
 c[0]=1;   c[1]=0;
 xs[0]=0; xs[1]=0;

 t=0;
 system_y_1=system_y_2=0;
 sys2_y_1=sys2_y_2=0;
 u=0;

 do
 {
   ref=reference(k*h);
   reft=ref_turev(k*h);
   //K(k) nin hesabi
   dum1[0]=P[0][0]*c[0]+P[0][1]*c[1];
   dum1[1]=P[1][0]*c[0]+P[1][1]*c[1];
   dum0=dum1[0]*c[0]+dum1[1]*c[1];
   dum0+=r2;
   dum0=1/dum0;
   K[0]=dum0*(phi[0][0]*dum1[0]+phi[0][1]*dum1[1]);
   K[1]=dum0*(phi[1][0]*dum1[0]+phi[1][1]*dum1[1]);

   //P(k+1) hesabi

   //dum= p phi^t
   for (i=0;i<2;i++) for(j=0;j<2;j++)
   {
     dum[i][j]=0;
     for(ii=0;ii<2;ii++) dum[i][j]+=p[i][ii]*phi[j][ii];
   }
   //dum2 = phi dum
   for (i=0;i<2;i++) for(j=0;j<2;j++)
```

```
    {
      dum2[i][j]=0;
      for(ii=0;ii<2;ii++) dum2[i][j]+=phi[i][ii]*dum[ii][j];
    }
    //dum1 = c * dum
    dum1[0]=c[0]*dum[0][0]+c[1]*dum[1][0];
    dum1[1]=c[0]*dum[0][1]+c[1]*dum[1][1];
    // dum= K dum1
    for (i=0;i<2;i++) for(j=0;j<2;j++) dum[i][j]=K[i]*dum1[j];
    // p(k+1) = dum2 + r1 - dum
    for (i=0;i<2;i++) for(j=0;j<2;j++)
       p[i][j]=dum2[i][j]+r1[i][j]-dum[i][j];

    // xs(k+1) hesabi
    dum0 =phi[0][0]*xs[0]+phi[0][1]*xs[1]+gama[0]*u+K[0]*(y-xs[0]);
    xs[1]=phi[1][0]*xs[0]+phi[1][1]*xs[1]+gama[1]*u+K[1]*(y-xs[0]);
    xs[0]=dum0;
    // ileri besleme
    um=2/h/h * ( reference(k*h+h) -2* y + ye1 ) - ume1;
    ume1=um;

    u=L[0][k]*(ref-xs[0])+L[1][k]*(reft-xs[1]);        //LQG

    u=u+um;
//    u=-L[0][k]*y-L[1][k]*dydt;                  //LQ
    if (abs(u)>20) if(u>0) u=20; else u=-20;
    ye1=y;
    y=system_y(t,u+beyaz(0.15));
    dydt=dydt*Td/(Td+Kh*h) + Td*Kh/(Td+Kh*h) * (y-ye1);
    fprintf(stream,"%lf %lf %lf %lf\n",t,y,u,ref-y);

    t+=h;
    k++;
  }while ( (t<t_max+h) && (!kbhit()) );
  fclose(stream);
  getch();

}

// Program 6

/* TDOF icinde STR kullanilarak yapilan simulasyon
 */
#include "my_sys.h"

#define ug 0.0  /* u sinyalinin uzerindeki olculemeyen gurultu yuzdesi */
#define rg 0.05  /* referans sinyalinin olculen gurultulu degisimi */
#define kf 2.6   /* 6. derece bessel filtresinin kesim frekansi */


#define k_max 1000
double landa=0.9;
```

```
double y_offs = 0;

double um=0,ume1=0;  // ileri besleme isareti
double a0 = 0;  // Observer polinomial : q + a0
double r2,t1,t2;

double a1,a2,b1,b2; /* sistemin gozlemlenen paremetreleri */
double am1,am2,bm1,bm2,wm=1,ksim=0.7; /* model sistemin parametreleri */
double s1,s2;  /* kontrol hesabinda kullanilacak parametreler */

double teta[5],K[5];
double y22;
int i_max,i,k,j,l;
double y[k_max+1],u[k_max+1];
double Fi[5];
char ch;
double top;
double P[5][5],DUM[5][5],DUM1[5][5];
double uu=0;
double uu1=0;

double x1,x2,alfa,beta,gama,w,w0,ksi,bb1,bb2;

double ref(int k)
{
  return y_offs+reference(k*h);
}

void main()
{
FILE *stream;

  wm=1;ksim=0.7;
  ABp(wm,ksim,h,&am1,&am2,&bm1,&bm2);

  randomize();

  for (i=1;i<5;i++)  for (j=1;j<5;j++)
  if (i==j) P[i][j]=10; else P[i][j]=0;
  teta[1]= -2; teta[2]=1;  teta[1]= teta[1]= 0.001;

  k=0;
  u[k]=0;y[k]=y_offs; k++;  u[k]=0;y[k]=y_offs;  k++;  u[k]=0;y[k]=y_offs;

  for (k=2;k<100;k++)
  {
      //Fi(k+1) matrisinin olusturulmasi
Fi[1]=-y[k-1];  Fi[2]=-y[k-2];
Fi[3]= u[k-1];  Fi[4]= u[k-2];

      //K matrisinin olusturulmasi:
```

```
// K = P x Fi
   for (i=1;i<5;i++)
   {
     K[i]=0;
     for (l=1;l<5;l++) K[i] += P[i][l]*Fi[l];
   }
// top= Fi transpose* K
   top=0;
   for (i=1;i<5;i++) top += Fi[i]*K[i];
 top+=landa;
 //K=K/top
 for (i=1;i<5;i++) K[i] = K[i] / top;
      // teta nin hesabi
 top=0;
 for (i=1;i<5;i++) top += Fi[i]*teta[i];
 top=y[k]-top;
 for (i=1;i<5;i++) teta[i] = teta[i]+K[i]*top;
      // P(k+1)'nin hesabi
 for (i=1;i<5;i++) for(j=1;j<5;j++)
 {
   if (i==j)  DUM[i][j] = (1 - K[i]*Fi[j])/landa;
   else       DUM[i][j] = (0 - K[i]*Fi[j])/landa;
 }
 for (i=1;i<5;i++) for(j=1;j<5;j++)
 {
   DUM1[i][j] = 0;
   for(l=1;l<5;l++) DUM1[i][j] += DUM[i][l]*P[l][j];
 }
 for (i=1;i<5;i++) for(j=1;j<5;j++) P[i][j]=DUM1[i][j];
      u[k]=beyaz(0.2);
      y[k+1]=sys2_y(u[k]);
  }

  stream=fopen("s7_8.dat","w");
  k=0;
  u[k]=0;y[k]=y_offs; k++;  u[k]=0;y[k]=y_offs;  k++;  u[k]=0;y[k]=y_offs;
  do {

    {

      //Fi(k+1) matrisinin olusturulmasi
Fi[1]=-y[k-1];  Fi[2]=-y[k-2];
Fi[3]= u[k-1];  Fi[4]= u[k-2];

      //K matrisinin olusturulmasi:
// K = P x Fi
   for (i=1;i<5;i++)
   {
     K[i]=0;
     for (l=1;l<5;l++) K[i] += P[i][l]*Fi[l];
   }
// top= Fi transpose* K
```

```
  top=0;
  for (i=1;i<5;i++) top += Fi[i]*K[i];
top+=landa;
//K=K/top
for (i=1;i<5;i++) K[i] = K[i] / top;
     // teta nin hesabi
top=0;
for (i=1;i<5;i++) top += Fi[i]*teta[i];
top=y[k]-top;
for (i=1;i<5;i++) teta[i] = teta[i]+K[i]*top;
     // P(k+1)'nin hesabi
for (i=1;i<5;i++) for(j=1;j<5;j++)
{
  if (i==j)  DUM[i][j] = (1 - K[i]*Fi[j])/landa;
  else       DUM[i][j] = (0 - K[i]*Fi[j])/landa;
}
for (i=1;i<5;i++) for(j=1;j<5;j++)
{
  DUM1[i][j] = 0;
  for(l=1;l<5;l++) DUM1[i][j] += DUM[i][l]*P[l][j];
}
for (i=1;i<5;i++) for(j=1;j<5;j++) P[i][j]=DUM1[i][j];
   }

     for (i=1;i<5;i++)
if (abs(teta[i])>3) if(teta[i]>0) teta[i]=23; else teta[i]=-23;


     a1=teta[1]; a2=teta[2]; b1=teta[3]; b2=teta[4];


   // u[k] nin hesabi
   if( ((b1!=0) && (b2!=0)))
   {
     r2= ( (-b2/b1+a0)*(b2*b2/b1/b1-b2/b1*am1+am2)
/ (b2*b2/b1/b1-a1*b2/b1+a2) ) + b2/b1;
     t1=(1+am1+am2)/(b1+b2); t2=t1*a0;
     s1=(a0+am1-a1-r2)/b1; s2=(am2*a0-r2*a2)/b2;
     u[k]= -r2*u[k-1] +t1*ref(k) +t2*ref(k-1) -s1*y[k] -s2*y[k-1]
   +(0.5-random(1000)/1000.0)*rg;
   }
   else u[k]=beyaz(rg);
   if (abs(u[k])>u_high) if (u[k]>0) u[k]=u_high; else u[k]=-u_high;

   // um ileri besleme isaretinin olusturulmasi
    if (b1!=0)
    {
      um= 1/bm1 * ( ref(k+1) + am1 * y[k] + am2 * y[k-1] - bm2 * ume1 );
      ume1=um;
    }
    u[k]=u[k]+um;
    uu1=uu; // u[k-1]
```

```
      uu=u[k]*(1+(-0.5+random(100)/100.0)*ug); // u[k]+white noise
     if (abs(u[k])>u_high) if (u[k]>0) u[k]=u_high; else u[k]=-u_high;
      y[k+1]=system_y(k*h,u[k]+beyaz(0.15));
     fprintf(stream,"%lf %lf %lf %lf\n",k*h,y[k]-y_offs,u[k],
       reference(k*h)-(y[k]-y_offs));

      k++;

   } while ((k<k_max) && !kbhit());

   fclose (stream);

}
```

ÖZGEÇMİŞ

Gökay Kadir HURMALI, 1968 yılında İzmir-Bergama'da doğmuştur. İlkokulu Kocaeli'nde Seka İlkokulunda, Ortaokul ve Liseyi Kadıköy Anadolu Lisesinde tamamlamıştır. 1990 yılında İstanbul Teknik Üniversitesi Uçak ve Uzay Bilimleri Fakültesi, Uzay Bilimleri ve Teknolojisi Bölümünden mezun olmuştur. Aynı yıl, İ.T.Ü. Fen Bilimleri Enstitüsü, Uzay Bilimleri ve Teknolojisi Programında Yüksek Lisans eğitimine başlamıştır. Aralık 1990 tarihinden itibaren İ.T.Ü. Uçak ve Uzay Bilimleri Fakültesinde Araştırma Görevlisi olarak çalışmaktadır.