AĞ KONTROL SİSTEMLERİNİN PERFORMANS ANALİZİ

GÜLSÜM YILDIRIZ

YÜKSEK LİSANS TEZİ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI
2012

CUMHURİYET ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

AĞ KONTROL SİSTEMLERİNİN PERFORMANS ANALİZİ

GÜLSÜM YILDIRIZ

YÜKSEK LİSANS TEZİ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

TEZ DANIŞMANI
DOÇ. DR. MANAFEDDİN NAMAZOV

SİVAS
2012

Bu çalışma Cumhuriyet Üniversitesi Fen/Sağlık Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlanmış ve jürimiz tarafından Maden Mühendisliği Anabilim Dalı'nda yüksek lisans tezi olarak kabul edilmiştir.

Başkan _____
Yrd. Doç. Dr. Gürkan YÜKSEK

Üye _____
Yrd. Doç. Dr. Yunis TORUN

Üye (Danışman) _____
Doç. Dr. Manafeddin NAMAZOV

ONAY

Bu tez çalışması, 06/02/2004 tarihinde Enstitü Yönetim Kurulu tarafından belirlenen ve yukarıda imzaları bulunan jüri üyeleri tarafından kabul edilmiştir.

_____
Prof. Dr. Mustafa DEĞİRMENCİ
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

Bu tez Cumhuriyet Üniversitesi Senatosu'nun 24.09.2008 tarihli ve 7 sayılı toplantısında kabul edilen Fen Bilimleri Enstitüsü Lisansüstü Tez Yazım Klavuzu adlı yönergeye göre hazırlanmıştır.

ÖZET

AĞ KONTROL SİSTEMLERİNİN PERFORMANS ANALİZİ

Gülsüm YILDIRIZ

Yüksek Lisans Tezi, Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışmanı: Doc. Dr. Manafeddin NAMAZOV

2012,109 sayfa

Gelişen teknolojiyle birlikte, kontrol sistemleri endüstriyel uygulamalarda vazgeçilmez hale gelmiştir. Kontrol sistemlerinin endüstriyel tesisler ile haberleşmeleri ancak belirli bir ağ üzerinden gerçekleştirilebilmektedir. Yakın zamana kadar, ağ üzerine kontrol sistemleri analiz ve tasarım araçları yalnız tek yönlü analizlere olanak sağlamakta, bu nedenle ağ üzerine kontrol sistemi analizlerinde ağ etkisi ihmal edilip, yalnız kontrol sistemi etkisi gözetilerek simülasyon yapılmaktaydı. Gelişen analiz araçlarıyla birlikte, ağ üzerine kontrol sistemleri geliştirme çalışmaları, bilgisayar ağları ve geleneksel kontrol sistemleri bilgi alanlarını birleştirir hale gelmiştir. Bu ancak eş-simülasyonla mümkündür.

Eş-simülasyon, kontrol sisteminin sürekli ve ayrık dinamiklerini, bilgisayar ağındaki ayrık olaylarla birlikte aynı anda simüle etme eylemidir. Eş-simülasyon sonuçları sistem tasarımları yapılırken, tasarım sürecinin gelişimi için faydalı bir kılavuz olacağı düşünülmektedir. Sistem tasarımcılarına, kendi AKS'ni anlamalarına ve geliştirmelerine yardımcı olmak için, eş-simülasyon için yeni araçlar keşfetme çalışmaları yapılmaktadır. Bu araçlardan biri de TrueTime Toolbox'dur.

TrueTime Toolbox, denetleyici görevler içeren çoklu-görevli ve gerçek-zamanlı kernellerin zamansal davranışının simülasyonunu yapmak için yazılmış bir Matlab/Simulink-tabanlı pakettir. Bu tezde TrueTime Toolbox kullanılarak ağ üzerine kontrol sistemlerinde ağ etkisi araştırılmış ve CSMA/CD (Ethernet), CSMA/AMP (CAN), Round Robin, FDMA, TDMA, Switched Ethernet sanayi ağlarının ve ağ etkisinin ihmal edildiği (ağın ideal kabul edildiği) durumun birbirlerine göre performans analizleri yapılmıştır.

Anahtar kelimeler: TrueTime, Sanayi Ağları, Ters Sarkaç

ABSTRACT

PERFORMANCE ANALYSIS OF NETWORK CONTROL SYSTEMS

Gülsüm YILDIRIZ

Master of Science Thesis, Department of Elekctrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Manafeddin NAMAZOV

2012, 109 pages

With the improving technology, control systems have become indispensable in industrial applications. Control systems to communicate with industrial facilities, only over a particular network can be realized. Until recently, network control systems analysis and design tools can provide only one-way analysis. Network effect on the analysis of network control system, therefore be neglected, the simulation was done considering only the effect of the control system. Together with the developing tools of analysis, control systems development studies on the network, fields of computer networks and traditional control systems knowledge have become combined. It is possible with only the co-simulation.

Co-simulation is to simulate the action of the dynamics of continuous and discrete control system with discrete events in computer network at the same time. While designing the system co-simulation results, for the development of the design process is thought to be a useful guide. To help the system designers to understand and develop their own network control system, studies are conducted to explore new tools for co-simulation. One of these tools is TrueTime Toolbox.

TrueTime is a Matlab/Simulink-based package which is designed to simulate the temporal behavior of multi-tasking real-time kernels containing controller tasks. This thesis studied the effect of network control systems on the network using TrueTime Toolbox and performance analysis of CSMA / CD (Ethernet), CSMA / AMP (CAN), Round Robin, FDMA, TDMA, Switched Ethernet industrial networks and the situation which network effect is neglected (the network is considered ideal) were performed according to each other.

Keywords: TrueTime, Industrial Networks, Inverted Pendulum.

TEŞEKKÜR

Yapılan bu araştırmada birçok kişinin desteği alınmıştır. Yüksek Lisans Tezi olarak seçtiğim bu konunun belirlenmesi, geliştirilmesi, planlaması ve yürütülmesi konusunda yardımlarını esirgemeyen Sayın Doç. Dr. Manafeddin NAMAZOV, bilgilerini benimle paylaşan Arş. Gör. Burak TEKGÜN ve Arş. Gör. İbrahim Emre ÇELİKKALE'ye;

Ayrıca kendilerinden çaldığım vakit ile bu çalışmayı sürdürürken sonsuz bir sabır ile destek olan sevgili eşim Emin YILDIRIZ ve doğacak bebeğimiz Kerem Batu YILDIRIZ'a;

Son olarak hiçbir zaman desteklerini esigemeyen ailem ile arkadaşım Bahar KORKMAZ'a ve bu çalışmaya katkı sağlayan herkese teşekkürü bir borç bilirim.

Gülsüm YILDIRIZ

Sivas - 2012

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER DİZİNİ

h_{max}	Maksimum iletim zamanı
h_{nece}	İletim zamanının gerekli olduğu durum
h_{true}	İletim zamanı sınırı(gerek ve yeter koşul)
h_{schur}	Sabit iletim periyodu kullanılırken ki iletim sınırı
τ_k	Kontrol gecikmesi
τ_k^{sd}	Sensör ve denetleyici arasında iletişim gecikmeleri
τ_k^{d}	Denetleyici içindeki sayısal gecikmeler
$\tau_k^{dç}$	Denetleyici ve çalıştırıcı arasında iletişim gecikmeleri
h	Örnekleme periyodu
ω	Kapalı döngü sisteminin doğal frekansı
φ_{lc}	Faz gecikmesi
N	toplam tesis sayısı
P	Periyodik AKS paketin boyutu
η	Ağ yolunun bant genişliği
θ	Sarkaç açısı
M	Aracın kütlesi
m	Sarkacın kütlesi
b	Aracın sürtünmesi
l	Ağırlık Merkezinin sarkaç uzunluğu()
I	Eylemsizlik momenti(Sarkaç)
r	Araç kasnağı yarıçapı
τ_M	Motor zaman sabiti
K_M	Motor kazancı
K_F	Geri beslemenin kazancı
F	Araca eklenen kuvvet
x	Araç Konum Koordinadı
T_s	Yerleşme zamanı
y	Süreç çıkışı
y_{max}	Süreç çıkışının maksimum tepe değeri
y_{kd}	Süreç çıkışının kararlılık durumu değeri
K	arka arkaya çarpışma
f	Frekans
$T_ö$	Örnekleme zamanı
K_p	Orantı kazancı
K_i	İntegral kazancı
K_d	Türev kazancı
T_p	Orantı denetim zamanı
T_i	İntegral denetim zamanı
T_d	Türev denetim zamanı
y_{sp}	Ayar noktası
K_u	Son kazanç
T_u	Son periyot
u	Kontrol sinyali
n	Mesaj içinde bitlerin sayısı

p	Gerçek bir bitin hatalı olma olasılığı
j	Hata kodlama eşiği
ω_c	Bant genişliği
ζ	Sönüm oranı

KISALTMALAR DİZİNİ

AKS	Ağ üzerine kontrol sistemleri
LTI	Lineer zamanla değişmeyen
MATI	İzin verilen maksimum aktarım aralığını
LME	Lineer matris eşitsizlik
BME	Bilineer matris eşitsizlik
A/D	Analog-Dijital
D/A	Dijital-Analog
as	Ağ simülatörü
FTP	Dosya aktarım protokolü
ODE	Adi türevsel denklemler
IP	İnternet protokolü
UDP	Kullanıcı veri bloğu iletişim protokolü
TS	Ters sarkaç
MSS	Merkezi sinir sistemi
TDTS	Taşıyıcı dengeli ters sarkaç
DC	Doğru akım
P	Orantı denetim etkisi
I	İntegral denetim etkisi
D	Türev denetim etkisi
PI	Orantı-İntegral denetimi etkisi
PD	Orantı-Türev denetim etkisi
PID	Orantı-İntegral-Türev denetim etkisi
FIP	Fabrika araçları protokolü
Profibus	Süreç alanı veri you
CAN	Denetleyici alan ağı
IEEE	Elektrik Elektronik Mühendisleri Enstitüsü
ATM	Asenkron aktarım modu
IEC	Uluslar arası elektroteknik komisyonu
TTP	Zaman tetikleme protokolü
LAS	Link aktif zamanlayıcı
CSMA/AMP	Mesaj önceliğinde keyfiyetli taşıyıcı algılı çoklu erişim
ISO	Uluslar arası standardizasyon örgütü
CSMA/CD	Çarpışma algılamalı taşıyıcı algılayıcı çoklu erişim
LAN	Yerel alan ağı
ID	Kimlik
FDMA	Frekans bölüşümlü çoklu erişim
TDMA	Zaman bölüşümlü çoklu erişim
FIFO	İlk giren ilk çıkar yöntemi
RT	Gerçek zamanlı
IRT	Eşzamanlı gerçek zamanlı
NRT	Gerçek olmayan zamanlı
Wlan	Kablosuz yerel alan ağı
SNR	Sinyal gürültü oranı
BLER	Blok hata oranını
Mac	Medya erişim kontrolü

NB	Geri çekilme sayısı
BE	Geri çekilme üssü
CRC	Döngüsel artıklık denetimi
BPSK	İkili faz kaydırmalıanahtarlama

1.GİRİŞ

1.1 Probleme Genel Bakış ve Motivasyon

Ağ üzerine kontrol sistemleri çalışması, bilgisayar ağları ve geleneksel kontrol sistemleri bilgi alanlarını birleştirir. Tarihsel olarak, ağ üzerine kontrol sistemleri için analiz ve tasarım araçları, bir diğerini ihmal ederek, etki alanlarından yalnız biri üzerine odaklanmıştı. Ortak tasarım için eş-simülasyonun temel ilkesi, kontrol sistemi tasarımı ve ağ tasarımının ayrılmaz olması ve dolayısıyla birlikte analiz (ve simüle) edilmesi gerektiğidir.

Eş-simülasyon, kontrol sisteminin sürekli ve ayrık dinamiklerini, bilgisayar ağındaki ayrık olaylarla birlikte aynı anda simüle etme eylemidir. Eş-simülasyon sonuçlarının, kontrol sistemleri ve bilgisayar ağlarının her ikisi için, sistem tasarımı karar verme sürecinin gelişimi ve eğitimi için kullanabilir olacağı düşünülmektedir. Sistem tasarımcılarına, kendi AKS(Ağ üzerine Kontrol Sistemleri)'ni anlamalarına ve geliştirmelerine yardımcı olmak için, yeni araçlar keşfetmek niyetiyle eş-simülasyon çalışmalarına yaklaşılmaktadır [2].

1.2 Önceki Çalışmalar

Ağ üzerine kontrol sistemleri araştırma alanı hala oldukça yenidir. Yine de, bu alanda ufuklar açan bir dizi çalışma mevcuttur.

Nilsson çalışmasında, AKS'nin birkaç önemli yönünü analiz eder. Öncelikle sabit gecikme, daha sonra bağımsız olarak rasgele (random) ve son olarak Markov süreci gibi AKS içindeki gecikmeler için modelleri tanıtmaktadır. Ayrıca, bağımsız olarak rasgele(independently random) ve Markovian gecikme modellerine dayanan AKS'ler için optimal stokastik kontrol teoremlerini tanıtmaktadır [3].

Nilsson, tez boyunca çoğu kullanılacak olan, AKS'ler için pek çok kritik varsayım yapar. Nilsson'ın varsayımları şunlardır:

- Tüm sensör düğümleri zaman odaklıdır. Bir sürecin çıkışı, herhangi bir zamanlama bozukluğu olmaksızın belirli aralıklarla örneklenir.
- Tüm denetleyici düğümler olay-güdümlüdür. Sensör bilgisi denetleyici düğümüne ulaşır ulaşmaz kontrol sinyali hesaplanır. Eğer denetleyici düğümüne hiç bilgi ulaşmazsa (paket kaybı nedeniyle), denetleyici yeni kontrol sinyalini hesaplamaz veya iletmez.

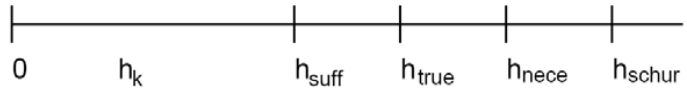
- Tüm çalıştırıcı düğümleri olay-güdümlüdür. Veri çalıştırıcı düğüme ulaşır ulaşmaz kontrol sinyali sürece eklenir. Eğer çalıştırıcı düğüme yeni bir kontrol sinyali gelmezse (paket kaybı nedeniyle), çalıştırıcı en son geçerli kontrol sinyalini eklemeye devam edecektir.

Otanez çalışmasında, ağ kullanımını azaltmak ve bant genişliği kullanımını artırmak için, bir AKS düğümünün kendi bilgisini gönderme yeteneğinde bir kısıtlama olan iletişim ölübandı tezini kullanır. Ölübantlı kapalı-döngü sistemlerinin kararlılığını analiz eder, ve sistem için optimal ölübandı keşfetmek için simülasyon yaklaşımını ve analitik yaklaşımı önerir [17].

Walsh ve ark. çalışmalarında, bir sürekli-zaman LTI(Lineer zamanla değişmeyen) sistemde sensör bilgisinin iletimi için statik ve dinamik zamanlama ilkelerini tanıtmaktadırlar. Bir sensörün, sensör verisini iletmesi için gereken en uzun zaman olan, izin verilen maksimum aktarım aralığını (MATI) tanıtırlar. Daha sonra AKS sonuçlarının stabil olması gibi, MATI üzerine sınırları türetirler. MATI dikkate alınarak görülür ki, incelenmekte olan sistemin Lyapunov fonksiyonu her zaman azalmaktadır [20].

Zhang, Walsh ve ark. çalışmalarında, iki farklı sınır kullanarak, her örnekleme anında bir ayrık-zaman LTI sistemi için bir Lyapunov fonksiyonunun düşüşünü sağlayan bir teorem geliştirdi. Çıkan sonuçlar Walsh ve ark.'a göre daha az korunumludur, çünkü Onlar'ın, sistemin Lyapunov fonksiyonunun sürekli azalmasına ihtiyaçları yoktur [23].

Zhang bir AKS için, bir h_{max} maksimum iletim zamanı olduğunu varsayar. Bu şu anlama gelir ki, eğer denetleyici h_{max} zamanı içinde bilgi almazsa, AKS, tesis durumunu iletime zorlar. Ek olarak, sistemin örnekleme periyodunun h_{suff}'a doğru sürekli yukarı değişimi ve sistemin kararlı hale gelmesi gibi bazı iletim periyodu $h_{suff}<h_{max}$ vardır. Şekil 1.4 iletim periyodlarının sınırlarını göstermektedir. h_{nece} gerekli durumu ifade eder, h_{true} sınırı(gerek ve yeter koşul) ifade eder ve h_{schur} sabit iletim periyodu kullanılırken ki sınırı ifade eder [23].



Şekil 1.1 İletim periyodunun sınırları [23]

Zhang, h_{suff} üzerindeki sınırları türetmek için bir Hurwitz matrisinin üstel iki farklı sınırını kullanır. Ayrıca, ağ-kaynaklı gecikme için bir kararlılık bölgesi fikrini tanıtır. Zhang'ın h_{suff} şöyle tanımlanabilir; bir sistem h_{suff} altındaki herhangi değerler arasında kendi örnekleme aralığı arasında keyfi geçiş yapabilir ve sistem katlanarak kararlı hale gelecektir. Bu sonuç ilginç olmakla birlikte, her örnekleme anında bir Lyapunov fonksiyonun düşüşünün garanti olması gibi, gerçek sistemler için ekseriyetle korunumludur. Bir sefer, bir Lyapunov fonksiyonuna zamanın bir bölümü için artışına izin verilebilir, örneğin, uzun bir süre boyunca garanti ederken, Lyapunov düşer. Hassibi ve arkadaşları çalışmalarında, harici, asenkron olaylar tarafından mod anahtarları başlatılan bir AKSye bir asenkron dinamik sistem gibi davranarak benzer bir yaklaşımı ele almışlardır. Yazarlar, farklı sistem modlarının oranları üzerinde sınırlar türetir. Bu gibi araçlar, bazı ayrık dizi içinde kendi örnekleme periyodunu anahtarlayan bir AKS'nin analizine uygundur [11].

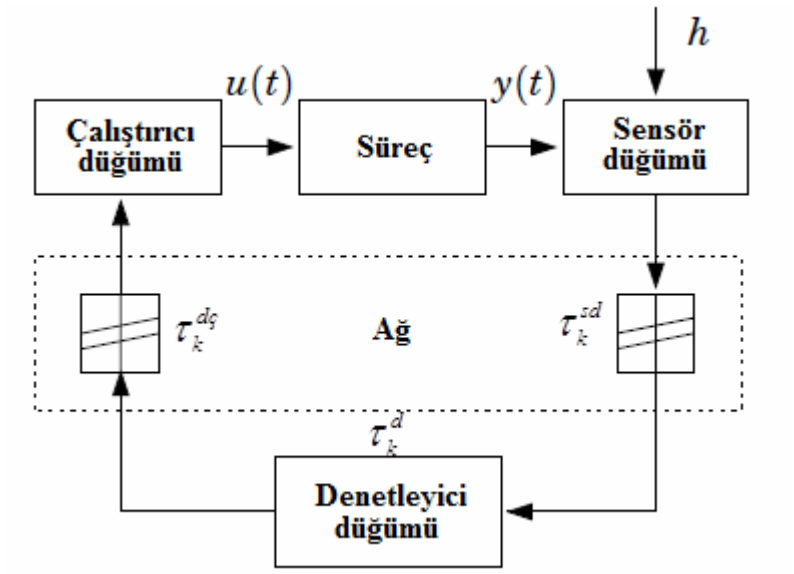
Branicky ve ark. çalışmalarında eş-tasarım için AKS eş-simülasyonu fikrini tanıtırlar. Bu yaklaşımda, as-2 ağ simülasyon paketi, bir Ethernet ağında etkileşerek AKS düğümlerinin(sensörler, çalıştırıcılar, denetleyiciler) simülasyonunu yapmak için genişler. AKS'lerin simülasyonunun yapılabildiği teknikleri ve ağ ve kontrol sistemlerinin her ikisinin de tasarımını geliştirmek için kullanılan sonuçları önerirler ve simülasyonlar için as-2'nin Ajan/Tesis uzantısı temeline dayandırırlar [8,13].

Czornik ve ark., Tzes ve ark., Xiao ve ark. ve Yu ve ark. çalışmalarında optimal anahtarlanmış AKS'lerin analizi ve tasarımı için farklı lineer matris eşitsizlik (LME) ve bilineer matris eşitsizlik (BME) araçlarını bir dizisini tanıtırlar [9, 19, 21, 22]. Yu ve ark. paket gecikmesinin biri gibi paket kaybı problemini yeniden formüle ederek paket ayrılmasını dahil eden modele uzanırken, Xiao ve ark. ağ kaynaklı gecikmenin bir Markov modeline dayanan anahtarlama sistemlerini sağlar [21,22]. Tzes ve ark., bu araçları kullanarak bir örnek AKS yapısının bir tasarımını sunar [2,19].

1.3 Ağ Üzerine Kontrol Sistemi Temelleri

Şekil 1.2'de kapalı döngü sistemi gösterilmiştir. Çalıştırıcılar ve sensörler bir iletişim ağına bağlanmıştır. Kontrol bilgilerini, sensör merkezi denetleyiciye, denetleyici de çalıştırıcıya gönderir. Merkezi denetleyici ağa bağlanmıştır, ve ağ üzerinde mesajları göndererek sensörler ve çalıştırıcılarla iletişimi sağlar. Ağ üzerinde mesaj göndermek genellikle biraz zaman alır. Sistem içindeki ağ ve zamanlama ilkelerine dayanarak, bu aktarım süresi farklı karakteristiklere sahip olabilir. Aktarım zamanı bazı ayarlamalarda

nerdeyse sabit olabilir, ama birçok durumda rasgele (random) bir şekilde değişmektedir. Örneğin aktarım gecikme süresi, ağ yüklemesine, diğer devam eden iletişim önceliklerine ve elektriksel bozukluklara bağlı olabilir [24]. Sensör, çalıştırıcı ve denetleyici düğümlerin senkronize edilmesine bağlı olarak çeşitli ayarlar bulunabilir. Önceki çalışmalarda, biraz farklı zamanlama ayarlı kontrol şemaları önerilmişti. Farklı ayarlar, düğümün olay-güdümlü mü yoksa zaman-güdümlü mü olduğundan gelmektedir. Olay-güdümlüden kasıt, düğüm, bir olay olduğunda, örneğin, bilgi ağı üzerinde farklı bir düğümden bilgi aldığında, aktivitesine başlar. Zaman-güdümlü ise şu anlama gelir; düğüm, aktivitesini önceden belirlenmiş zamanda başlatır, örneğin, düğüm periyodik olarak çalışabilir. Temelde, sistem içinde bilgisayar gecikmesinin üç çeşidi vardır, bkz. Şekil 1.2:



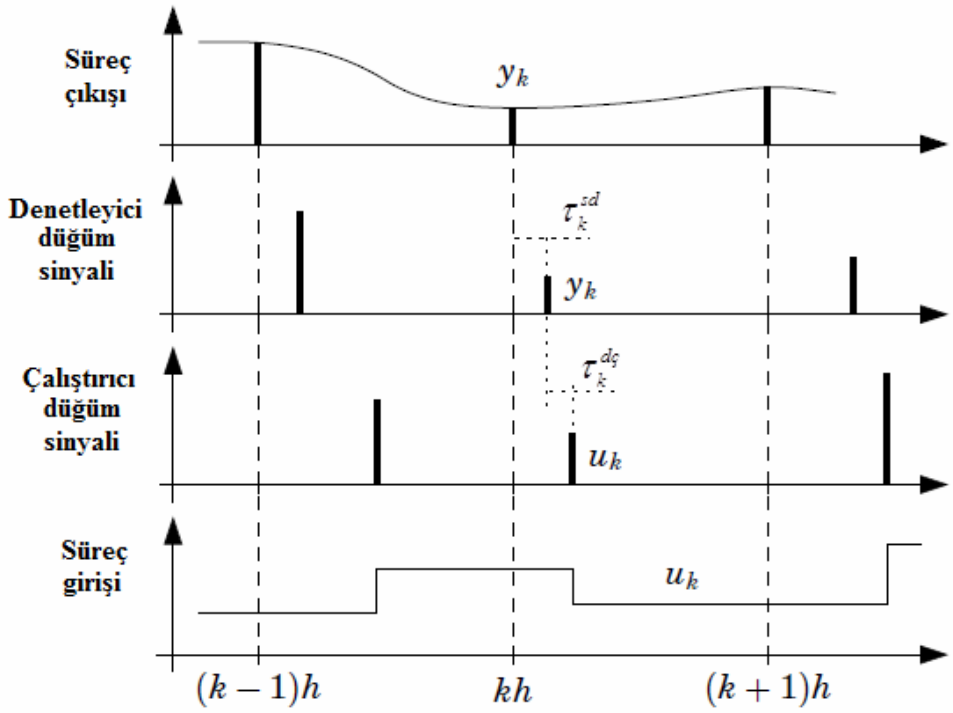
Şekil 1.2 τ_k^{sd} ve $\tau_k^{dç}$ gecikmeleri kaynaklı dağıtılmış kontrol sistemi. Denetleyici düğümündeki sayısal gecikme, τ_k^d, de belirtilir [3]

- Sensör ve denetleyici arasında iletişim gecikmesi τ_k^{sd}.
- Denetleyici içindeki sayısal gecikmeler τ_k^d.
- Denetleyici ve çalıştırıcı arasında iletişim gecikmeleri $\tau_k^{dç}$.

k indisi, gecikmelerin olası bir zamana bağımlılığını belirtmek için kullanılır.

Kontrol sistemi için kontrol gecikmesi τ_k, çalıştırıcı içinde kullanıldığında ölçülen bir sinyal örneklenirkenki zaman, bu gecikmelerin toplamıdır, yani,

$\tau_k = \tau_k^{sd} + \tau_k^{d} + \tau_k^{dç}$. Bu kontrol sistem ayarındaki önemli problemlerden biri, rasgele bir şekilde değişen gecikmelerdir. Bu, sistemi, zamanla-değişmeyen sistemlerin doğrudan kullanılamaması durumu için analiz ve tasarım sonuçların zamanla-değişen ve teorik sonuçları haline getirir. Zaman farklılıklarından kurtulmanın tek yolu denetleyici düğümü ve çalıştırıcı düğümündeki giriş üzerine, zamanlanmış tamponlar tanıtmaktır. Eğer tamponlar yeterince büyük yani en kötü gecikmeden büyük seçilirse, iki düğüm arasındaki bir aktarım için gecikme belirleyicidir. Bahsi geçen düzen, [25] örneğinde önerilmiştir. Döngü içindeki tamponların girişi şu anlama gelir, bazen gerektiğinde eski bilgilerimizi kullanırız. Bu, bir olay-güdümlü ayarla kıyaslandığında performansın düşmesine yol açan bir durumdur.



Şekil 1.3 Kontrol sisteminde sinyallerin zamanlaması. Şekil 1.1'le kıyaslanarak, Birinci şema süreç çıkışını ve örnekleme anlarını gösterir, ikinci şema denetleyici düğümündeki sinyali gösterir, üçüncü şema çalıştırıcı düğümdeki sinyali gösterir, ve dördüncü şema süreç girişini gösterir[3]

Bir örneklenmiş-veri kontrol perspektifinden bakıldığında, örnek periyod *h*'la eşit mesafede süreç çıkışın örneklenmesi doğaldır. Ayrıca, kontrol gecikmesinin mümkün olduğunca kısa tutulması da doğaldır. Nedeni şudur, zaman gecikmeleri, genellikle sistem kararlılığının ve performansının dejenere olmasına neden olan faz gecikmesine neden olmaktadır. Bu motivasyon olay-güdümlü denetleyici düğümlü ve olay-güdümlü

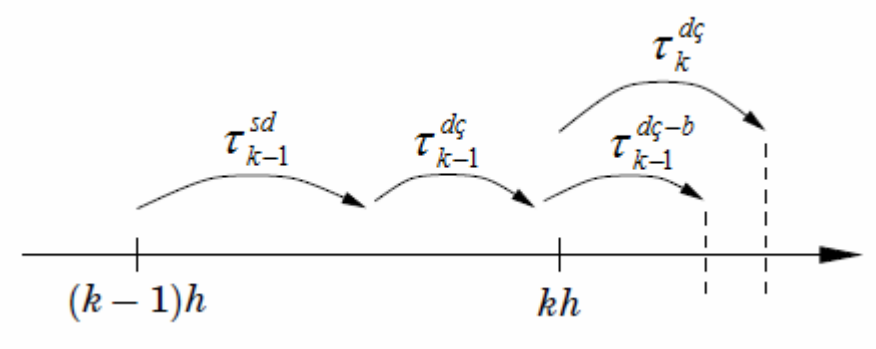
çalıştırıcı düğümlü bir sistem ayarı önerir. Bu da şu anlama gelir, yeni kontrol sinyalinin hesaplanması, sırasıyla yeni kontrol sinyalinin D/A-dönüşümü akabinde sensör düğümü ve denetleyici düğümünden alınır alınmaz gerçekleşir. Benzer bir sistemde zamanlama Şekil 1.3'de gösterilmiştir. Bu ayarın bir sakıncası, sistemin zaman-değişimli hale gelmesidir. Yine Şekil 1.3'den görülebileceği üzere, süreç girişi düzensiz zamanlarda değişir.

Kontrol gecikmesinin örnekleme periyodundan daha az olduğunu varsayarsak, $\tau_k \leq h$. Bu çeşitli yollarla yapılabilir. Görünürdeki bir kontrol noktasından, normal bir tasarım $0.2 \leq \omega h \leq 0.6$'dır, burada h örnekleme periyodu, ω kapalı döngü sisteminin doğal frekansıdır [26]. Örneğe eşit bir gecikmeyle, tasarım, denetleyici tarafından uyarılmış bir faz gecikmesine sahiptir, φ_{lc}, $11° \leq \varphi_{lc} \leq 34°$. Daha büyük bir faz gecikmesi, kontrole pek çok süreç zararı verebilir. Eğer örnekleme periyodundan, h, daha büyük bir kontrol gecikmemiz varsa, örnekler, çalıştırıcı-düğümüne kronolojik olmayan bir sırayla gelebilir. Bu, algoritmaların ve sistem analizinin her ikisinin uygulamasını çok daha zor hale getirir. Kontrol gecikmesinin h'den küçük olması durumu, kontrol gecikmesinin h'dan daha fazla değiştirilemeyeceği varsayımı ile değiştirilebilir, böylece örneklerin kronolojik sırayla ulaşmasını garanti edebilir. Eğer zaman aşımı gerçekleşirse, kontrol sinyali hesabı tahmine dayalı yapılır. Eğer gecikmenin olasılık fonksiyonu uzun kuyruklara sahipse, kontrol şeması bu özelliğiyle gerçekleştirilir. Örnekleri kaybettiğimizde, denetleyici aynı zamanda algılama ve kontrol için kullanılabilir. Bu olaylara boş örnekleme denir.

Aşağıda τ_k^{sd} ve $\tau_k^{dç}$ etkilerine bakılacaktır. τ_k^{d}'nın etkileri $\tau_k^{dç}$'nın içinde olabilir. Ayrıca, döngüdeki ilk aktarım gecikmesinin ne kadar büyük olduğu bilgisine sahip olunduğu varsayılır. Devamında bu özelliği uygulama yolları ele alınmaktadır [3].

1.3.1 Eski Zaman Gecikmeleri Bilgisi - Zaman Etiketleri

Düğümlerdeki senkronize saatleri kullanarak, gecikme bilgisi, her mesaja nesil zamanını (zaman etiketini) ekleyerek elde edilebilir. Çoğu ağlarda, zaman etiketi tarafından getirilen ek ağ yükü, mesaj ve ağ yükü ile kıyaslandığında ihmal edilebilir. Gecikme bilgisi denetleyici düğümünde kullanılabilir. Zaman etiketi, denetleyiciye, alınan ölçümün ne kadar eski olduğunu söyler. Bu bilgi daha sonra kontrol sinyali hesaplamasında denetleyici tarafından kullanılabilir.



Şekil 1.4 Bir zamanlama döngüsü esnasında zamanlama çizimi gösterilmektedir [3]

Denetleyici düğüm, ölçülen zaman etiketi ile denetleyici düğümün iç saati kıyaslayarak τ_k^{sd}'yı kolayca hesaplayabilir. Denetleyici $\tau_{k-1}^{dç}$ hakkında bilgi elde edebilir. Çoğu ağ uygulamalarında, ağ üzerinde son kontrol sinyali gönderildiğinde, ağ ara yüzünden bilgi almak mümkündür. Eğer bu bilgi uygunsa, u_k hesaplandığında, aktarım gecikmesi $\tau_{k-1}^{dç}$ bilinir. $\tau_{k-1}^{dç}$'nin denetleyiciye yayılmasını sağlamak için bir başka yol, hemen çalıştırıcı düğümden denetleyiciye, aktarım süresini $\tau_{k-1}^{dç}$ içeren bir mesaj göndermektir. Bununla birlikte, sıradaki kontrol sinyali hesaplandığında, denetleyicinin, bu mesajı alacağı belli değildir. İki kontrol sinyali arasında kontrol sistemi için zamanlama Şekil 1.4'te gösterilmektedir. $\tau_k^{dç}$'nin uzunluğu hakkında bilgiyi denetleyiciye geri gönderme aktarım-zamanı için aktarım gecikmesi $\tau_k^{dç-b}$ tanıtılmaktadır. Genel durumda, kontrol sinyali hesapladığında $\tau_{k-1}^{dç}$ durumu bilinebilir ve şöyle yazılabilir.

$$\tau_{k-1}^{sd} + \tau_{k-1}^{dç} + \tau_{k-1}^{dç-b} < \tau_{\mathrm{k}}^{\mathrm{sd}} + h \qquad [1.1]$$

Bir kontrol döngüsünde zaman gecikmelerinin toplamı, kontrol gecikmesinin örnekleme aralığından, *h,* az olduğunu varsayılır. Kontrol gecikmesi τ_k^{sd} ve $\tau_k^{dç}$'nın toplamı olduğundan, a<0,5 için, gecikmelerin her birinin [0,a*h*] aralığında dağıtıldığını varsaymak mantıklıdır. Eski zaman gecikmeleri bilgisinin garantilenmesi sorunu, iletişim ağ davranışının bazı özel durumları için analiz edilecektir.

Toplam rasgelelik Eğer iletişim ağları rasgele ve bağımsız zaman gecikmelerini verirse, eşitlik [1.1]'den de görüleceği üzere, zaman gecikmelerinin dağılım durumu

$$a < \frac{1}{3} \quad . \qquad [1.2]$$

Toplam sıra Eğer ağ mesajlarının sırayla iletilme şartı eklenirse, iletilir. Eğer

$$a < \frac{1}{2} \qquad [1.3]$$

ise $\tau_k^{dç}$'nın uzunluğunu içeren mesaj, bir sonraki mesajdan her zaman daha önce iletilecektir. Böyle bir sistemin bir uygulaması, ağda gönderilen mesajlar için genel bir sıra gibi bir durum gerektirecektir.

Öncelik Bazı iletişim ağları, mesajların önceliklerini mümkün kılar ve bekleyen mesajların öncelik sırasına göre gönderileceğini garanti eder. Böyle bir sistemde, yeni ölçüm y_k içeren mesajdan daha yüksek öncelikli $\tau_{k-1}^{dç}$ içeren mesajı verebiliriz. Eğer

$$a < \frac{1}{2} \qquad [1.4]$$

ise, bu $\tau_{k-1}^{dç}$'nin bilgisini garantiler [3].

1.3.2 Varsayımların Toplamı

Nilsson çalışmasında kontrol sistemi hakkında aşağıdaki varsayımlar yapmaktadır [3]:

- Sensör düğümü zaman-güdümlüdür. Süreç çıkışı, herhangi bir zamanlama bozukluğu olmadan periyodik olarak örneklenir. Örnekleme periyodu *h*'dır.
- Denetleyici düğümü olay-güdümlüdür. Sensör bilgisi denetleyici düğümüne ulaşır ulaşmaz, kontrol sinyali hesaplanır.
- Çalıştırıcı düğümü olay güdümlüdür. Bilgi çalıştırıcı düğüme ulaşır ulaşmaz kontrol sinyali sürece eklenir.
- İletişim gecikmeleri τ_k^{sd} ve $\tau_k^{dç}$, bilinen stokastik özelliklerle rasgele değişir. Toplam zaman gecikmesinin değişimi, $\tau_k^{sd} + \tau_k^{dç}$, bir örnekleme aralığından küçüktür.

- Geçmiş zaman gecikmesi uzunluğu denetleyicide bilinmektedir. Denetleyici, yeni ölçümler için zaman aşımına dayanmaktadır. Bu durumda denetleyici düğümü bazen olay-güdümlü ve zaman-güdümlü olabilir [3].

2. AĞ ÜZERİNE KONTROL SİSTEMİ ÖRNEĞİ

Bu bölümde, tez boyunca örnek olarak kullanılacak Ağ üzerine kontrol sistemleri tanıtılacaktır. AKS'nin iletişim kuracağı ağın mimarisi ve parametreleri anlatılacaktır. Bir denetleyici paylaşılarak tesislerin örnekleme sürelerini ayarlama politikalarından bazıları açıklanacaktır. Ek olarak, örnek bir kontrol sistemi açıklanacaktır.

2.1 Eş-Simülasyon

Ağ üzerine kontrol sistemi simülasyonu yapmak için, tek bir as-2(ağ simülatörü) simülasyon komut dosyası içinde bir ağ topolojisinin simülasyonuyla birlikte bir kontrol sisteminin durum-uzay eşitliğinin dinamiklerinin simülasyonunun birleştirilmesi gerekmektedir. Komut dosyası, FTP akışları gibi her ağ çapraz-trafiği kadar iletim, kuyruk, yönlendirme paketlerinin alınması gibi, ağ dinamiklerinin simülasyonlarını yapmak için as-2'nin ağ bileşenlerini kullanır. Komut dosyası, kontrol sistem bileşenlerinin örnekleme, kontrol ve tahrikin simülasyonunu yapmak için, Liberatore [8, 13] tarafından oluşturulan Ajan/Tesis nesnesini kullanır. Her tesisin dinamikleri ya birinci dereceden Euler yaklaşımları ile, ya da dış ODE çözen bir çağrı yoluyla satır içi simülasyonu yapılabilir

2.2 Ağ Topolojisi

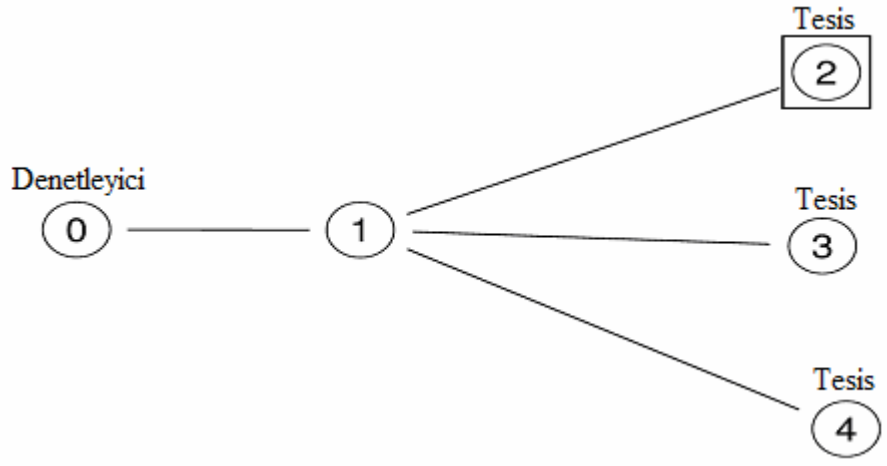
Bir ağ üzerine kontrol sisteminin denetleyicisi, sensörleri ve çalıştırıcısı bir bilgisayar ağının türünde düğümlerdir. Ağ kontrolünün yapılabildiği birçok farklı ağ tipleri ve protokolleri bulunmasına rağmen, burada IP üzerinden basit bir UDP-tabanlı protokol kullanarak heterojen ağlar üzerine odaklanma tercih edilmiştir.

Bu simülasyonlar için, denetleyici, 1.5 Mbs bant genişliğine ve 1ms bir sabit bağlantı gecikmesine sahip bir T1 hattı aracılığıyla bir yönlendiriciye doğrudan bağlıdır. Ağ üzerinde diğer tüm düğümler, hem tam durum geribesleme için sensörler hem de kontrol sinyali için çalıştırıcı içeren bir tesistir. Tüm tesis düğümleri, 0.1 ms sabit bağlantılı gecikmeler ile 10 Mbps bağlantılar aracılığıyla yönlendiriciye bağlanır. Simülasyon boyunca, ağ sisteminde birçok özelliği değiştirebiliriz, ek olarak:

- Ağ üzerindeki tesis sayısı,
- T1 bağlantısında yönlendiricinin arabelleğinin boyutu,
- Yönlendirici üzerinden çapraz trafik miktarı (FTP akışları gibi),

- T1 bağlantısın sabit gecikmesi, gibi.

Şekil 2.1 bizim üç tesis ve bir denetleyicili örnek ağımızın topolojisini göstermektedir. Denetleyici düğüm 0'da, yönlendirici düğüm 1'de, ve tesisler düğüm 2, 3 ve 4'tedir. Birinci tesis (düğüm 2) kutulanmıştır çünkü yalnız durum ve çıkışı izlenebilir. Diğer tüm tesisler için, yalnız ağ olaylarının simülasyonu yapılır; simülasyon esnasında işlemci zamanını muhafaza etmek için dinamiklerin simülasyonu yapılmaz. Tez boyunca, bir kontrol sinyalini hesaplamak için denetleyici için zamanın gerekli olduğu ve ağ üzerindeki iletimi göz ardı edecek kadar küçük başlatılabildiği varsayılacaktır. Gerçekte, her ne kadar bir miktar zaman gerekliyse de; denetleyici tarafından kontrol edilebilen tesislerin sayısı gibi, ağ parametrelerini seçerken, bu dikkate alınmalıdır. Yukarıda listelenen değişen ağ bağlantılarını da ekleyerek, ağ üzerinde tesisin örneklerinin zamanlamasının rasgele veya sabit seçildiği umulabilir. Sistem örnekleme zamanı rasgele seçilmediğinde, her tesis tasarlanan aynı örnek periyodu kullanır ve tesislerin örnekleri, örnek süre içinde birbirlerinden zaman içinde eşit uzaklıkta olması planlanır. Daha doğrusu, her tesisin olabileceği örnekleme periyodu ve örnekleme zamanı seçilir:



Şekil 2.1 Üç tesisli ağ topolojisi[2]

$$h_i = H, \quad i \in \{1,...,N\}$$
$$t_i[k] = \left(k + \frac{i-1}{N}\right)h_i, \quad i \in \{1,...,N\}, \quad k \in \{1,...\}, \qquad [2.1]$$

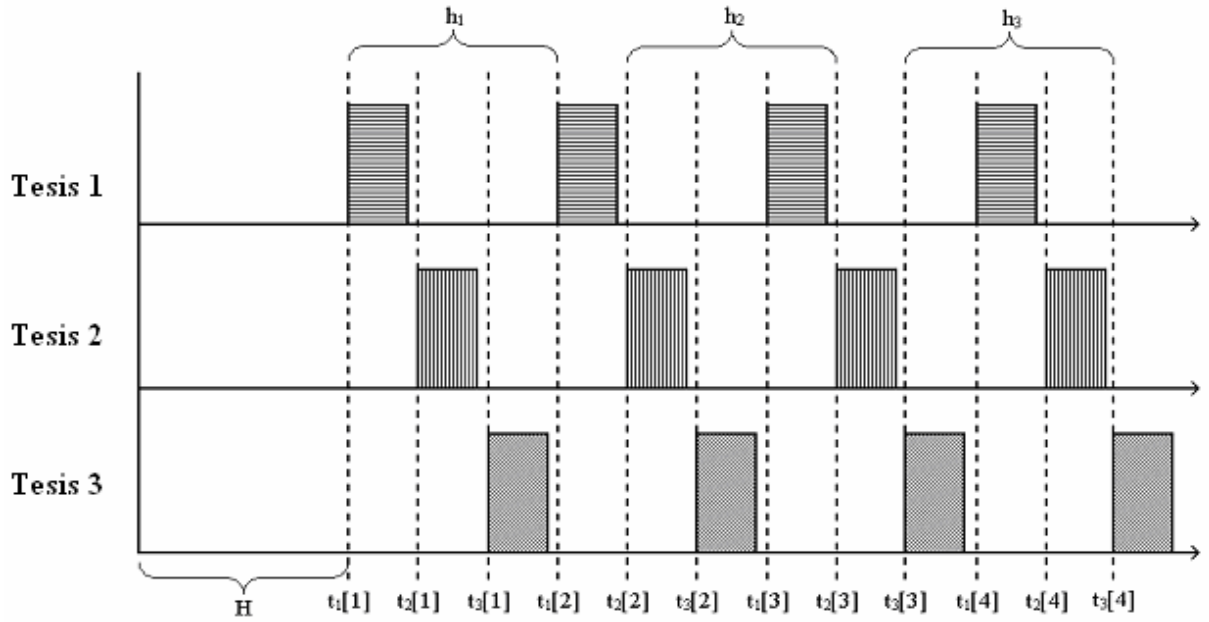
Burada h_i, i^{th} tesisinin örnekleme periyodudur, t_i [k] i^{th} tesisinin k^{th} örneğinin mutlak zamanıdır, H tasarlanan sistemin örnekleme periyodudur ve N toplam tesis sayısıdır.

Şekil 2.2 rasgele olmayan simülasyonlarda tesisin örnek zamanlama ayarını gösterir. Şekil 2.2, her paketin iletim süresinin, her paket için ayrılan bant genişliğinden daha az olduğunu varsayar. Daha doğrusu,

$$\frac{P}{\eta} < \frac{H}{N} \qquad [2.2]$$

Burada P, periyodik AKS paketin boyutu, η, ağ yolunun bant genişliği, ve H ve N daha öncekiler gibidir. Eğer bu varsayımlar doğru değilse, sonuçlanan ağa bir aşırı-görevlendirilen ağ diyoruz ki bu; ağ üzerinde, ağın sağlayabileceğinden daha fazla bant genişliği gerektiren, düğümlerin görevlendirilmesidir. Böyle bir durum, veri kaybıyla sonuçlanacaktır.

Tesisin örneklemesi Eşitlik [2.1]'e göre sabitlendiğinde, tüm tesislerin örnekleri arasında bir periyodiklik oluşur. Bu aynı zamanda ağ üzerinde her tesise maksimum bant genişliği miktarını sağlar.



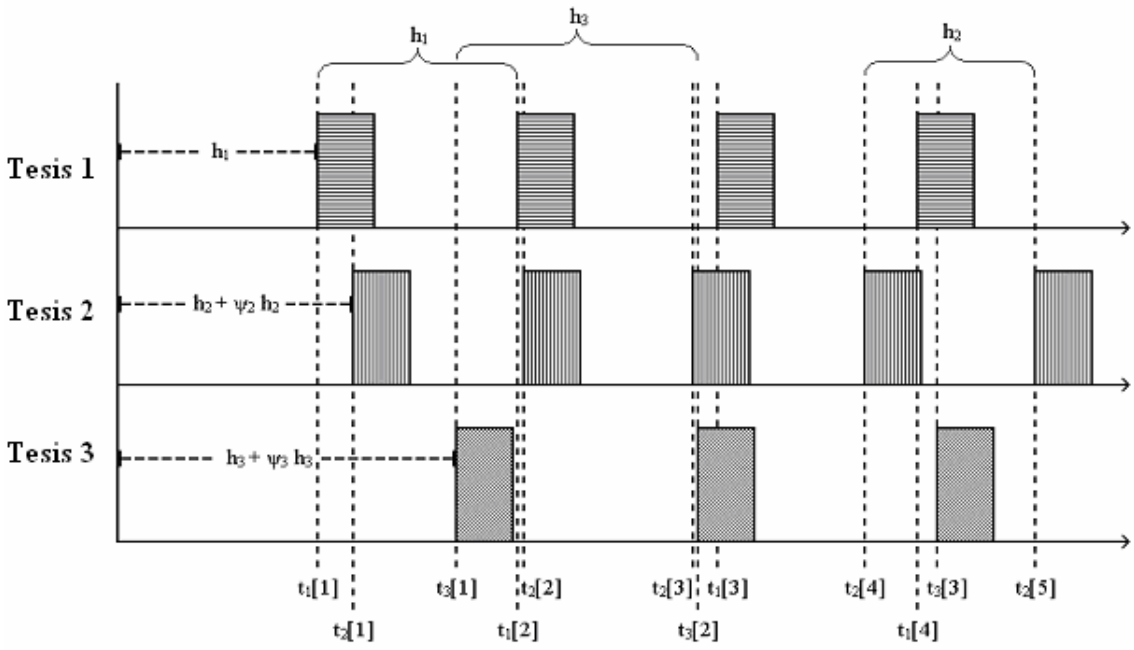
Şekil 2.2 Rasgele olmayan örnek zamanlama ayarının şeması[2]

Örnekleme zamanları rasgele olduğunda, her tesis (gözlemlenen ilk tesis hariç) %20'den yukarı yada tasarlanan örnekleme periyodundan aşağı bir örnekleme periyodu seçer. Ek olarak, tesislerin zamanlaması, örnekleme periyodu boyunca rasgeledir. Tesislerin zamanlamasında hiç kararsızlık olmadığını varsayarak, ilk tesisin örnekleme periyodunu ve örnekleme zamanını rasgele olmayan durumda olduğu gibi seçilir:

$$\begin{aligned} h_1 &= H, \\ t_1[k] &= kh_1 \end{aligned} \qquad [2.3]$$

Daha sonra her i tesisi için, kendi aralığında bir düzgün dağılımdan örnekleme periyodunun eğimini ve başlangıç örnek zamanın eğimini temsil eden iki rasgele değişken seçilir, $\rho_i \in [0.8,1.2]$ ve $\psi_i \in [0, 1)$. Daha sonra aşağıda verildiği gibi, her tesisin örnekleme periyodunu ve örnekleme zamanı sabitlenir.

$$\begin{aligned} &h_i = \rho_i H, \quad i \in \{2,...,N\}, \quad \rho_i \in [0.8, 1.2], \\ &t_i = (k + \psi_i)h_i, \quad i \in \{2,...,N\}, \quad k \in \{1,...\}, \quad \psi_i \in [0,1) \end{aligned} \qquad [2.4]$$



Şekil 2.3 Rasgele örnek zamanlamasını ayarlama şeması[2]

Şekil 2.3, rasgeleleştirilmiş simülasyonlarda tesislerin örnek zamanlama ayarını göstermektedir.

Tesislerin örneklemesi rasgeleleştirildiğinde, Eşitlik [2.1]'in rasgeleleştirilmemiş örnek zamanlamasında bulunan periyodikliği ortadan kaldırır; bu, saat asenkronluluğun ve eğiminin aperiyodik trafik sekanslarına neden olduğu gerçek dünya durumunda daha iyi yansıtılır. Ek olarak, ağ üzerindeki tek tek tesisler, rasgeleleştirilmemiş durumdan daha fazla veya az bant genişliği etkin olarak tahsis edilebilirler ve etkileri zamanla değişebilir [2].

2.3 Kontrol Sistemlerine Bir Örnek, Ters Sarkaç

Bu kısımda, örneklerde kullanılacak olan bir ters sarkaç sistemi tanıtılıp açıklanacaktır. Her ne kadar sistem doğrusalsızlık içermekte ise de, biz analizin kolaylığı için sistemi doğrusallaştıracağız. Ancak, simülasyon sonuçlarında doğrusallaştırılmamışın sonucunu gözlemlemek yerine, ters sarkaç sistemini hem doğrusal hem de doğrusal olmayan sistem eşitlikleri ile simüle edeceğiz.

Ters sarkaç uzun yıllar kontrol sistemleri gerçekleştirilen eğitimlerde örnek bir sistem olarak kullanılmıştır. Klasik bir örnek, ya ideal bir çubuğun sonunda bir nokta kütle ya da bir düzlemde 360° döner bir ortak noktaya tutturulmuş tek tip çubuktan oluşur. Bu, bazen sarkacın dönen noktasının hareket düzleminde bir parça üzerinde yatay hareket eden bir arabaya tutturulur. Arabanın ivmesi, sarkacın döneri içinde bir tork indüklemektedir; Sarkaç, arabanın ivmesini düzgün kontrol ederek, yayın üstünü dengelemek için imal edilebilir. Ters sarkaçla ilgili Matlab kodları içeren daha fazla bilgi için, bkz [10, 15, 18] [2].

Ters sarkacın(TS) bazı önemli uygulamaları arasında şunlar da vardır;

2.3.1 Bir Robotik Kolun Dinamiklerinin Simülasyonu

Ters sarkaç problemi, robotik kollarda mevcut olan kontrol sistemlerine benzer. Kol için basınç merkezinin ağırlık merkezinin altında yer aldığı yani böylece sistemin aynı zamanda kararsız olduğu durumda, ters sarkaç dinamikleri robotik kolun dinamiklerini simüle eder. Bu durumda robotik kol Ters Sarkaca çok benzer davranışlar sergiler [7].

2.3.2 Ayakta Bir İnsan Modeli

Ayakta düz vaziyetteyken kararlılığını koruma yeteneği insanların günlük aktiviteleri için büyük önem taşımaktadır. Merkezi sinir sistemi (MSS) vücudun pozunu kaydeder ve insan vücudunun pozu içinde değiştirir ve dengesini korumak için kasları harekete geçirir.

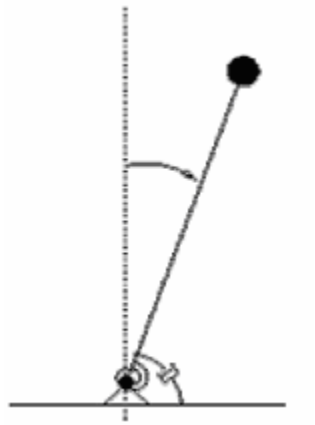
Ters sarkaç, bir insanın duruşuna (sessiz duran) dair yeterli bir model olarak kabul edilir.

Bir ters sarkaç (ekli yay olmadığını varsayarak) kararsızdır ve bu nedenle sarkacı dengelemek için sarkacın durumunun geri beslemesinin gerekliliği açıktır.

MSS geri besleme kontrolü için iki model genel olarak kabul edilir:

- Zamanla değişmeyen, doğrusal geri beslemeli kontrol;
- Bir eşik dışında doğrusal geri besleme. Eşik dahilinde hissedilen geribildirim yoktur.

Bir yay ve amortisör olarak modellenmiş kas ve destekleyici dokudaki sertlik gibi bazı pasif mekanizmalar vardır.



Şekil 2.4 Bir yay ve amortisör gibi modellenmiş pasif mekanizmalı ters sarkaç

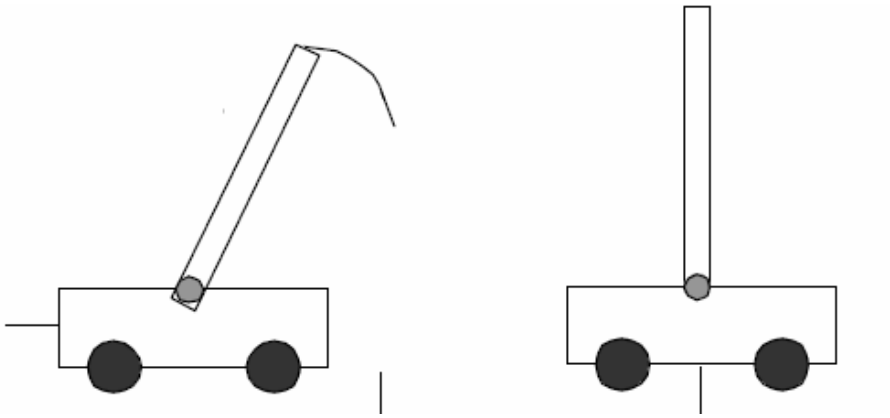
Eğer yay yeterince sert ise, yay ve amortisör negatif bir geri besleme döngüsüne yol açar ki bu, sarkacı dengelemek için yeterli olabilir [7].

2.3.3 Problem Tanımı

Sisteme bazı dış kuvvetler uygulanmadan ters pozisyondaki sarkacı dengelemek neredeyse imkansızdır. Aşağıda gösterilen Taşıyıcı Dengeli Ters Sarkaç (TDTS) sistemi, sarkaç taşıyıcısına bu kontrol gücünü uygulamaya olanak sağlar. TDTS, bir kayışlı tahrik sistemi aracılığıyla bir DC-servo moturun sayesinde taşıma için kontrol gücü sağlar. TDTS teçhizatının çıkışları taşıma konumu, taşıma hızı, sarkaç açısı ve sarkaç açısal hızı (bizim durumumuzda yalnız sarkaç açısı) olabilir. Sarkaç açısı, tutarlı ve sürekli çekiş gücü sağlayan servo motoru kontrol eden bir analog denetleyiciyle geri beslenir. Çalışmanın amacı, yolda taşıyıcının konumunun hızlı ve doğru kontrol

edilmesi ve sarkacın her zaman böyle hareketler sırasında ters pozisyonunu sıkıca muhafaza etmesi durumları gibi sarkacı dengede tutmaktır.

Problem ileri ve geri yönde hareket edebilen bir araç, aşağıda gösterildiği gibi sarkacın araçla aynı yönde hareket edebilmesini sağlayan uzunluğunun altında araca menteşeli bir sarkaç içerir. Yani araca monte edilmiş sarkaç, aracın hareket ekseni boyunca düşmekte serbesttir. Sistem, kontrol edilir böylece sarkaç dengede ve yukarı doğru dik kalır, ve basamak bozulmasına karşı dirençlidir.



Şekil 2.5 Ters sarkacın araca bağlantısı

Problem basit bir bağlaşık sistem içerir. Eğer sarkaç merkez dışına kayarsa, düşmeye başlayacaktır. Sarkaç araca birleştirilir ve araç ters yönde harekete başlar, böylece aracın hareketi sarkacın merkez dışına kayar hale gelmesine neden olur. Sistemin bir parçasının değiştirilmesi diğer parçalarda da değişime neden olur, bu ilk bakışta göründüğünden daha karmaşık bir kontrol sistemidir. Bu nedenle, problem bulanık kontrolün bir tanıtımı olarak kullanılmıştır.

Ters sarkaç aracı yol boyunca çalışır ve bir elektrik motoruna bağlı bir kayış tarafından çekilir. Bir potansiyometre, dönme hızından aracın pozisyonunu ölçer ve diğer bir potansiyometre sarkacın açısını ölçer.

Eğer çıkış, dikey ekseniyle(yukarı dik pozisyonda) ilgili olarak sarkacın açısı ise, sistemin kararsız olduğunu anlarız, eğer bunu küçük bir açı olarak bırakırsak sarkaç aşağı düşecektir. Sistemi dengeli hale getirmek için, yani, sarkacı yukarı doğru pozisyonda tutabilmek için, bir geri besleme kontrol sistemi kullanılmalıdır.

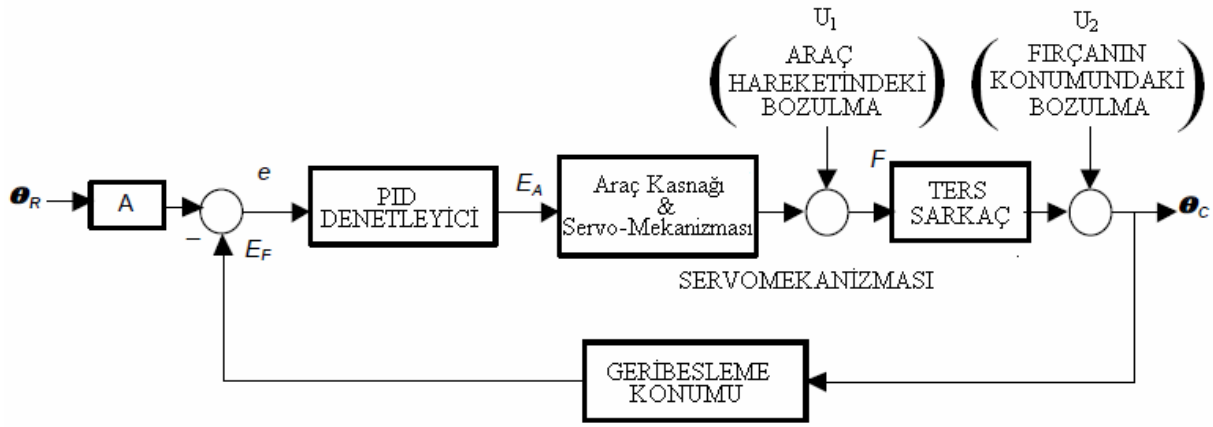
Ters sarkaç, doğrusal kontrol teorisi için mükemmel bir test yatağıdır. Bu klasik ters sarkaç kontrol deneyinde, sarkaç kolunu kararsız ters konumda dengede tutmak

için, bir geri besleme kontrol kuralı bulmaya çalışıyoruz. İlk bağlantı noktası doğrusal olarak taşınır, bizim ilk bağlantı noktamız döner.

Bu yüzden, Ters Sarkaç sistemi bir araç ve bir sarkaçtan oluşur. Denetleyicinin amacı, sarkacın devrilmesine sebep olmadan aracı kendi komutasındaki konuma taşımaktır. Açık döngü sisteminde, sistem kararsızdır.

Atanan görev, verilen ters sarkaç için bir kontrol döngüsü analizi, tasarımı ve geliştirilmesidir.

Geri beslemeli kontrol sistemi için tüm blok-şeması aşağıda verilmiştir.



Şekil 2.6 Ters Sarkaç Sistemi kontrolünün blok şeması [7]

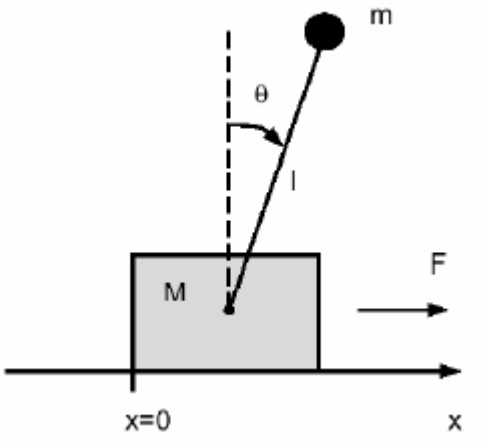
Uygulamamız yalnız sarkaç açısından geri besleme içerir (yani, dört durum arasından yalnız biri geri besleme için kullanır, diğer durumlar taşıma pozisyonu, taşıma hızı ve sarkaç açısal hızıdır). Uygulama, araç-konum kontrol döngüsü birleştirilerek artırılabilir.

Problemde, sarkaç başlangıçta manuel olarak yukarı doğru konumdadır, bu, kararsız bir denge konumudur, yada birkaç başlangıç yer değiştirmesi (konum) verilir. Denetleyici daha sonra sarkacı dengelemek ve bozukluğun varlığında bu dengeyi korumak için çalıştırılır. Basit bir bozulma dengelenmiş sarkaç üzerinde hafif bir vuruş olabilir. Karmaşık bir bozulma, örneğin bir rüzgar fırtınası olabilir.

Bu ayar açık döngü kararsız sistemin kontrolünü çalışılırken kullanılabilir. Bu, geri besleme kontrolün dengeleyici faydalarının bir göstergesidir. Basit faz ileri kompansatörden sinirsel ağ denetleyicilerine kadar, kontrol tekniklerinin bir dizisi uygulanabilir [7].

2.3.4 Matematiksel Analizler

Bir ters sarkaç klasik bir kontrol problemidir. Süreç, bir giriş sinyali ve çeşitli çıkış sinyalleriyle doğrusal olmayan ve kararsızdır. Amaç, bir motor tahrikli araç üzerine dikey bir sarkacı dengelemektir. Aşağıdaki şekil, bir ters sarkacı gösterir. Amaç, sarkaç düşmeden aracı x yönü boyunca istenilen noktaya hareket ettirmektir. Araç, bir denetleyici (bu uygulamada analog) tarafından kontrol edilen bir DC motorla tahrik edilmektedir. Aracın c konumu(bu uygulamada değil) ve sarkaç açısı θ ölçülür ve kontrol sistemine sunulur. Bir bozulma(gürültü) kuvveti, FBOZULMA, sarkacın tepesine uygulanabilir.



Şekil 2.7 Ters Sarkaç hareket düzlemi [7]

Tabana uygulanan bir kuvvet sonucunda sarkacın açısı verilerek, sistemin matematiksel bir modeli geliştirilmiştir [7].

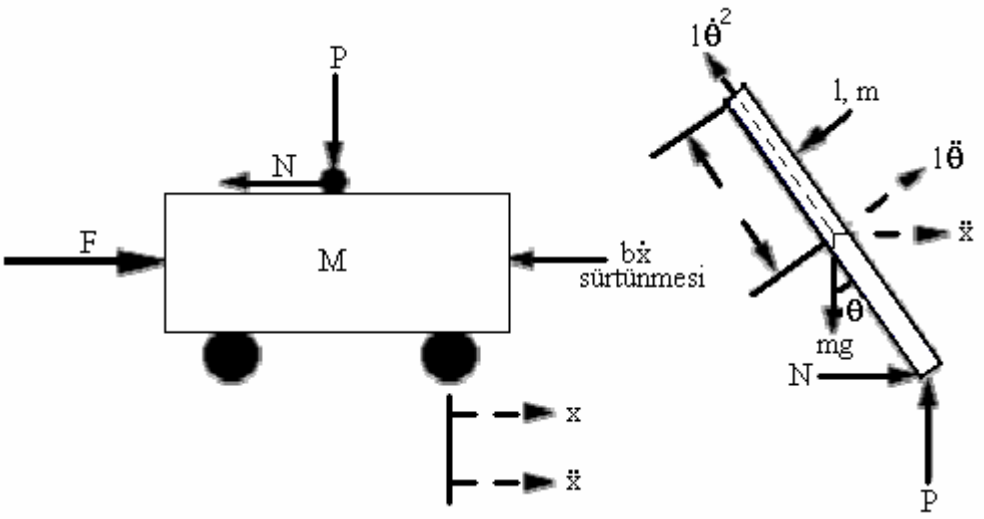
2.3.5 Kurulum Açıklaması

Ters sarkaç hareketli bir araca monte edilmiştir. Bir servomotor, bir kayış/kasnak mekanizması aracılığıyla aracın çeviri hareketini kontrol eder. Yani, araç, kasnak ve kayış mekanizması aracılığıyla bir servo dc-motorla birleştirilmiştir. Motor, ayrıca denetleyici-devreler de içeren servo elektronikleri tarafından elde edilmiştir. Servo elektroniğin tahrik-sinyali üretmesi amacıyla, sarkacın açısal hareketini geri beslemek için bir döner-potansiyometre kullanılır.

Denetleyici devreleri, servomotor ve sürülen kasnak/kayış mekanizması aracılığıyla aracı süren hata sinyalini işler. Araca ileri ve/veya geri hareket ters sarkaç üzerine momentler ekler ve böylece sarkacı dik tutar [7].

2.3.6 Ters Sarkaç Sistem Denklemleri

Sistemin Serbest Vücut Şeması, hareket denklemlerini elde etmek için kullanılır. Sistemin iki Serbest Vücut Şeması aşağıda verilmiştir.



Şekil 2.8 Ters Sarkaç üzerine etki eden kuvvetler [7]

Yatay yönde, aracın Serbest Vücut Şeması içinde güçleri toplayarak, aşağıdaki hareket denklemi elde edilir:

$$M\ddot{x} + b\dot{x} + N = F \qquad [2.5]$$

Ayrıca dikey yöndeki kuvvetler toplanabilir, ancak bu herhangi bir yararlı bilgi getirmez. Dikey yöndeki kuvvetlerin toplamı önemli değildir çünkü bu yönde bir hareket yoktur ve yerin reaksiyon kuvvetinin tüm düşey kuvveleri dengeleyeceği düşünülür.

Sarkaç üzerindeki moment için yatay yönde uygulanan kuvvet aşağıdaki gibi belirlenir:

$$\begin{aligned} \tau &= r \times F = I\ddot{\theta} \\ F &= \frac{l\ddot{\theta}}{r} \\ &= \frac{ml^2\ddot{\theta}}{l} \\ &= ml\ddot{\theta} \end{aligned} \qquad [2.6]$$

N yönündeki bu kuvvetin bileşeni $ml\ddot{\theta}\cos\theta$ 'dir.

Yatay eksen boyunca hareket eden merkezkaç kuvvetinin bileşeni aşağıdaki gibidir:

$$\begin{aligned} F &= \frac{l\dot{\theta}^2}{r} \\ &= \frac{ml^2\dot{\theta}^2}{l} \\ &= ml\dot{\theta}^2 \end{aligned} \qquad [2.7]$$

N yönündeki bu kuvvetin bileşeni $ml\dot{\theta}^2\sin\theta$ 'dir.

Yatay yöndeki sarkacın Serbest Vücut Şemasındaki kuvvetleri toplayarak, N için bir eşitlik elde edilebilir:

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \qquad [2.8]$$

Eğer birinci eşitliğin [2.5] yerine ikinci eşitlik [2.8] konursa, bu sistem için hareketin ilk denklemi elde edilmiş olur:

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \qquad [2.9]$$

Hareketin ikinci denklemini elde etmek için, sarkacın dik kuvvetleri toplanır. Bu eksen matematiksel karmaşıklığı basitleştirmek için seçilmiştir. Bu eksen boyunca sistem çözümü cebirsel tasarruf sağlar. Önceki denklemin elde edişi gibi, bu kuvvetlerin dikey bileşenleri aşağıdaki eşitliği sağlamak için şöyle kabul edilir:

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta \qquad [2.10]$$

Yukarıdaki denklemde P ve N koşullarından kurtularak, aşağıdaki denklemi elde etmek için sarkacın ağırlık merkezi etrafındaki momentler toplanır:

$$-Pl\sin\theta - Nl\cos\theta = l\ddot{\theta} \qquad [2.11]$$

Son iki eşitliği birleştirerek, ikinci dinamik denklem elde edilir:

$$(l + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \qquad [2.12]$$

Ters sarkacın dinamiklerini tamamen tanımlayan denklemler dizisi şunlardır:

$$(M + m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \qquad [2.9]$$

$$(l + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \qquad [2.12]$$

Bu iki eşitlik doğrusal değildir ve çalışma aralığı için doğrusallaştırılması gerekir. Sarkaç, kararlı denge pozisyonundaki 'Pi' radyan olan kararsız denge konumunda istikrar sağlarken denklemlerin bu dizisi teta=Pi ile ilgili doğrusallaştırılmalıdır. Teta=Pi+α kabul edilir, (burada α, yukarı dikey doğrultudaki küçük bir açıyı göstermektedir.)

Bu nedenle, $\cos(theta) = -1, \quad \sin(theta) = -\alpha \quad ve \quad \left(\frac{d(theta)}{dt}\right)^2 = 0$.

Hareketin iki denklemi doğrusallaştırıldıktan sonra şu hale gelir (burada u, giriştir)

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\varphi} = u \qquad [2.13]$$

$$(l + ml^2)\ddot{\varphi} - mgl\varphi = ml\ddot{x} \qquad [2.14]$$

Analitik olarak doğrusallaştırılmış sistem denklemlerinin transfer fonksiyonunu belirlemek için, öncelikle sistem denklemlerine Laplace dönüşümleri uygulanmalıdır. Laplace dönüşümleri

$$(M + m)X(s)s^2 + bX(s)s - ml\varphi(s)s^2 = U(s) \qquad [2.15]$$

$$(l + ml^2)\varphi(s)s^2 - mgl\varphi(s) = mlX(s)s^2 \qquad [2.16]$$

Transfer fonksiyonunu bulurken, başlangıç koşulları sıfır kabul edilir. Transfer fonksiyonu istenilen konumdan [Çıkış], araç üzerindeki kuvvete [Giriş] kadar olan değişimlerle ilgilidir.

Çıkışı ilgilendiren Phi açısına bakarak, X(s) için ilk denklemin çözümü,

$$X(s) = \left[\frac{(l - ml^2)}{ml} - \frac{g}{s^2} \right] \varphi(s) \qquad [2.17]$$

Daha sonra, ikinci eşitlikte yerine konursa:

$$(M+m)\left[\frac{(l + ml^2)}{ml} + \frac{g}{s} \right] \varphi(s)s^2 + b\left[\frac{(l + ml^2)}{ml} + \frac{g}{s} \right] \varphi(s)s - ml\varphi(s)s^2 = U(s) \qquad [2.18]$$

Yeniden düzenleyerek, transfer fonksiyonu:

$$\frac{\varphi(s)}{U(s)} = \frac{\frac{ml}{q} s^2}{s^4 + \frac{b(l + ml^2)}{q} s^3 - \frac{mgl(M+m)}{q} s^2 - \frac{bmgl}{q} s} \qquad [2.19]$$

burada,

$$q = (M+m)(l + ml^2) - (ml)^2 \qquad [2.20]$$

Yukarıdaki transfer fonksiyonundan görüleceği gibi, başlangıç noktasında hem bir kutup hem de bir sıfır mevcuttur. Bu iptal edilebilir ve transfer fonksiyonu şu hale gelir:

$$\frac{\varphi(s)}{U(s)} = \frac{\frac{ml}{q} s}{s^3 + \frac{b(l + ml^2)}{q} s^2 - \frac{mgl(M+m)}{q} s - \frac{bmgl}{q}} \qquad [2.21]$$

Transfer fonksiyonu daha da basitleştirilebilir:

$$\frac{\varphi(s)}{U(s)} = \frac{mls}{qs^3 + b(l+ml^2)s^2 - mgl(M+m)s - bmgl} \qquad [2.22]$$

burada $q = (M+m)(l+ml^2) - (ml)^2$.

Eğer sistemdeki sürtünme ihmal edilirse, sürtünme katsayısı b=0 alınır, böylece

$$\frac{\varphi(s)}{U(s)} = \frac{K_{IP}}{\frac{s^2}{A_{IP}^{\ 2}} - 1} \qquad [2.23]$$

burada $K_{IP} = \frac{1}{(M+m)g}$ ve $A_{IP} = \pm\sqrt{\frac{(M+m)mgl}{(M+m)(l+ml^2) - (ml)^2}}$

Böylece, IP için doğrusallaştırılmış yaklaşım transfer fonksiyonu elde edilmiş olur. Zaman etki alanında, transfer fonksiyonu şu şekilde ifade edilir [7]:

$$\frac{\varphi(t)}{u(t)} = \frac{K_{IP}}{\frac{D^2}{A_{IP}^{\ 2}} - 1} \qquad [2.24]$$

2.3.7 Çalıştırma Mekanizması

Kumanda mekanizması, bir kasnak ve kayış vasıtasıyla bir DC motor tarafından tahrik edilen hareketli bir araçtan (ray üzerinde) oluşur. Bu yüzden, çalıştırma mekanizmasının genel transfer fonksiyonu, "DC Motor" ve "Kasnak, Kayış&Araç" transfer fonksiyonuna bağlıdır [7].

2.3.8 Kasnak, Kayış ve Araç

Motorun Yük-Ataleti, makara (yarıçapı r) ve araç ve sarkacın kütlelerinden oluşur. Motor tarafından sağlanan yük-torku aşağıdaki gibi verilir [7]:

$$T_L = (M+m)r^2 D\omega \qquad [2.25]$$

burada: $T_L \propto r^2$ ve $F \propto r$.

2.3.9 Motor

Motorun dinamikleri, aynı zamanda çalıştırma mekanizmasının transfer fonksiyonunu etkileyecektir. Endüvi-kontrollü servo dc-motor için deneysel transfer fonksiyonu aşağıda verilmiştir:

$$\omega = K_m \frac{E}{\tau D + 1} \qquad [2.26]$$

Burada τ zaman sabitidir, ve yük sürücüsüne bağlıdır. Bundan, daha hafif yükün daha yüksek değeri τ olacaktır. K_M(raydan/saniye/volt), kararlı durum kazancıdır.

Bu nedenle, çalıştırıcı mekanizmanın tüm transfer fonksiyonu [7]:

$$\frac{U(s)}{E(s)} = K_m \frac{(M+m)rs}{(\tau_m s + 1)} \qquad [2.27]$$

2.3.10 Tüm Sistemin Transfer Fonksiyonu

Tüm (kontrolsüz) sistem için açık döngü ve doğrusallaştırılmış transfer fonksiyon aşağıdaki gibi verilebilir:

$$\frac{\varphi(s)}{E(s)} = K \frac{s}{(\tau_m s + 1)(\frac{s^2}{A_P{}^2} - 1)} \qquad [2.28]$$

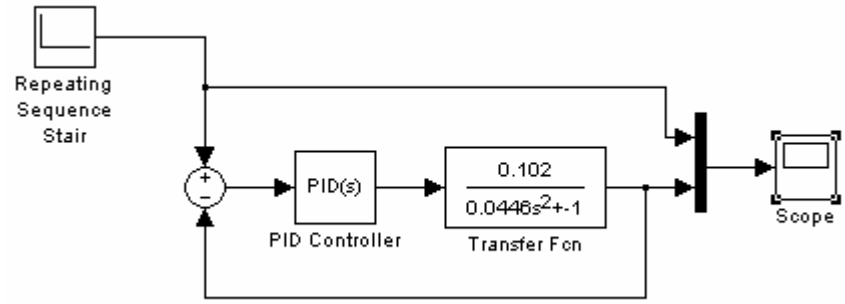
burada $K = K_F K_P K_M r(M+m)$, E(s)=Hata Voltajı ve φ(s)=Sarkacın açısal konumudur [7, 30, 31].

3. KONTROL SİSTEMİ SİMÜLASYONU

Bir ağ üzerine kontrol sisteminin dinamiklerinin simülasyonunu yapmak için farklı yollar vardır. Aynı şekilde, çoğu paketler, bir ağ sisteminin ayrık olaylarının simülasyonunu yapmak için vardır. Amaç, sistem dinamiklerinin ve ağ olaylarının eş zamanlı simülasyonunu yapmaktır (eş simülasyon). Sistem dinamiklerinin simülasyonunu yapma ve doğrulama metotları kadar 6. Bölümde eş-simülasyon için yapılar da değerlendirilecektir [2]. Eş simülasyon yönteminin, geleneksel ağın ideal kabul edildiği simülasyon yöntemiyle kıyaslanması için öncelikle aynı örnek, geleneksel bir yöntem kullanılarak simüle edilmektedir.

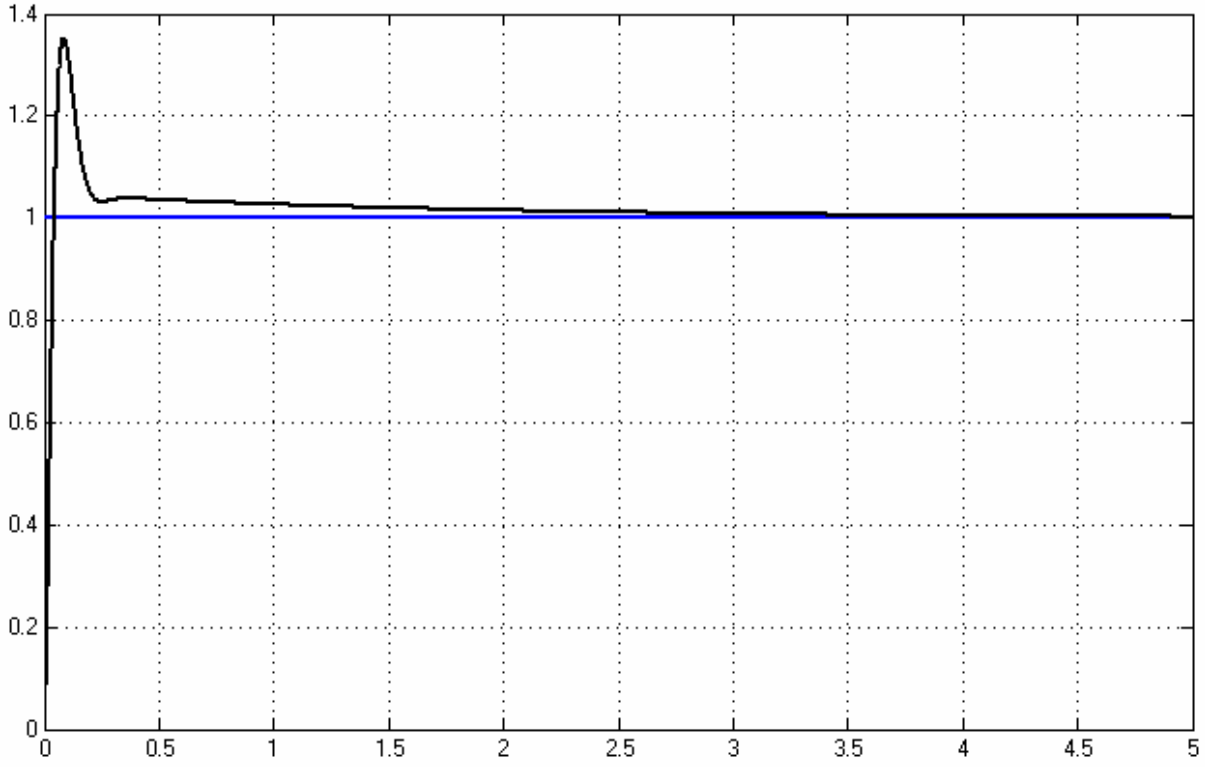
3.1 Ters Sarkacın MATLAB Ortamında Simülasyonu

Bu bölümde, Bölüm 2.3.1'de anlatılan Ters Sarkaç sisteminin bir örneğinin MATLAB üzerinde simülasyonunu yaparak, ağ etkisi ihmal edildiği durumda (ağın ideal kabul edildiği durum) sistemin verdiği tepki gözlemlenecektir. Bölüm 6.10'da yapacağımız simülasyonlarda daha çok, kullanılan ağ üzerine odaklanılacağı için, modellenmiş Ters Sarkaç sisteminde araç, kasnak, kayış ve motor modellemelerine gidilmemiş, daha basit bir düzenek olan [2.23]'de verilen eşitlikten yola çıkılarak Şekil 3.1'de MATLAB Simulink üzerinde modellenmesi gösterilen $\frac{0.102}{0.0446s^2 - 1}$ transfer fonksiyonuna [7] sahip bir Ters Sarkaç sisteminin simülasyonu yapılmıştır. Kullanılan PID parametreleri K_p=226.7593, K_i=122.5547 ve K_d=12.4501 olarak tasarlanmıştır. Kullanmış olduğumuz PID parametrelerin tespiti için kullanılan yöntemler Bölüm 5.2.3'de açıklanmıştır.

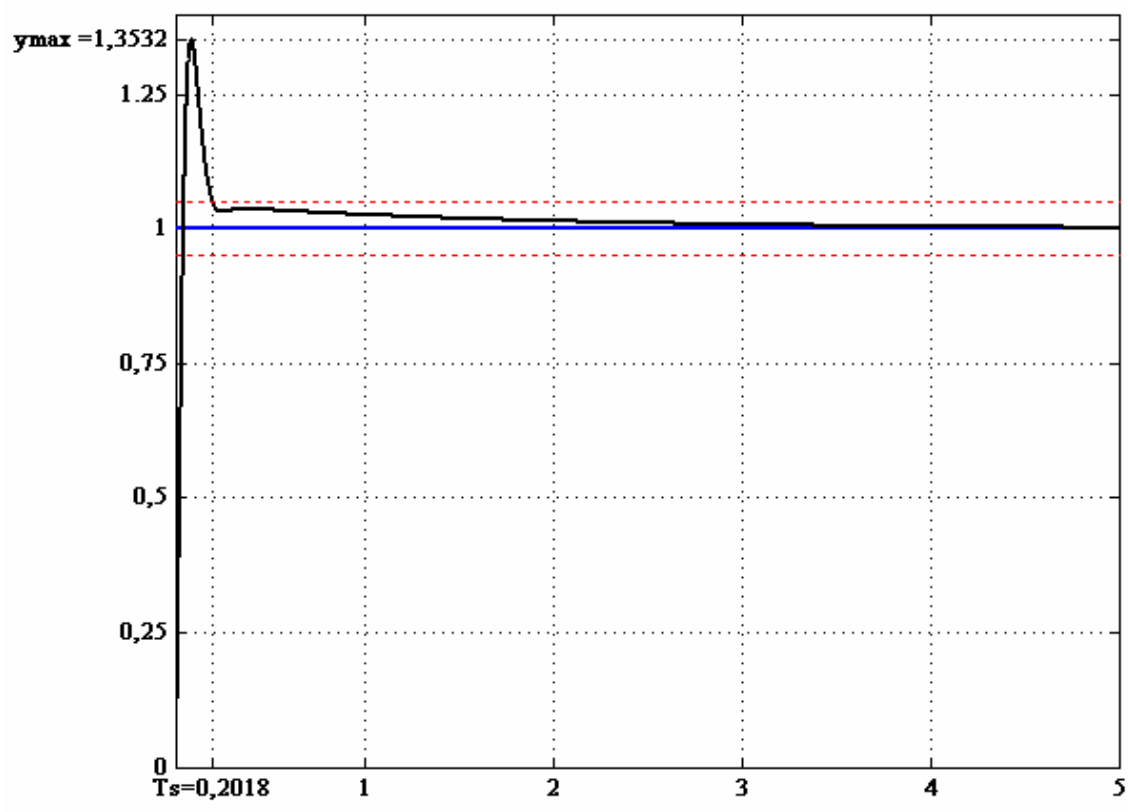


Şekil 3.1 Ters Sarkacın PID Kontrolü Matlab Modeli

Sistem modeli çıkışı Şekil 3.2'de verilmiştir. Burada sistemin haberleştiği ağ ideal olarak kabul edilmektedir. Bu sebeple ağın, sisteme etkisini gözlemlemek mümkün değildir. Burada gözlemlenen sonuç sistemin en ideal koşullarda çalıştığı durumlar için geçerlidir.



Şekil 3.2 Ters Sarkacın PID Kontrolü Matlab Simülasyonu Çıkışı

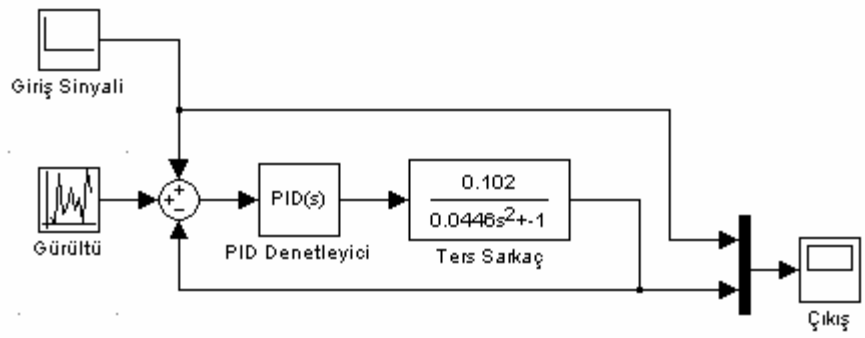


Şekil 3.3 Ters Sarkacın PID Kontrolü y_{max} ve T_s değerleri

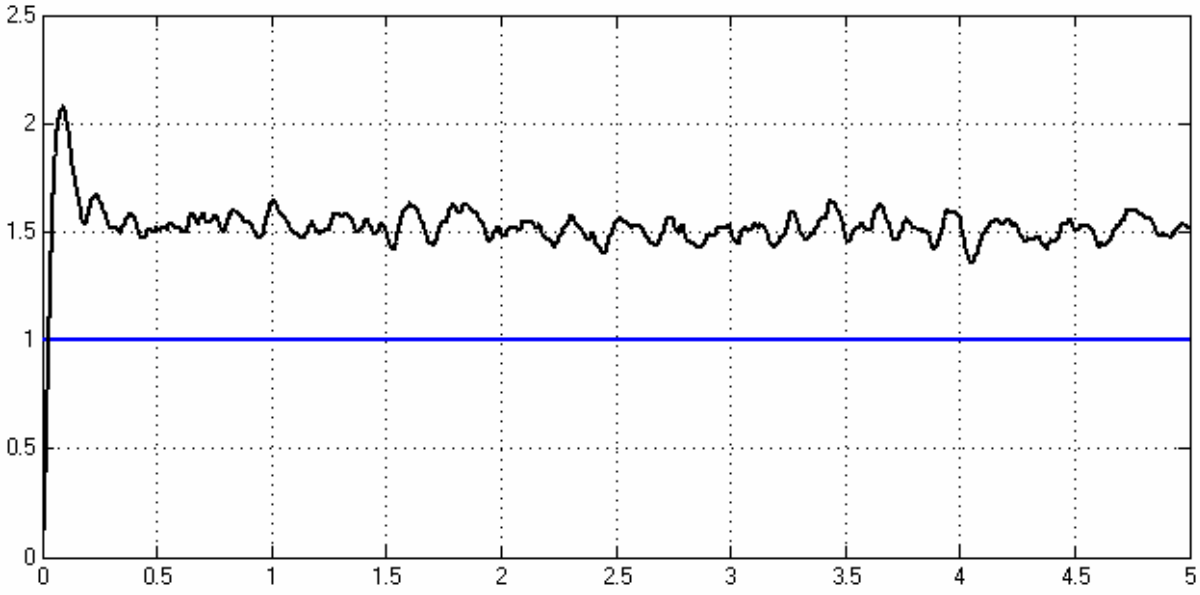
Sistemin yerleşme zamanı (settling time)T_s ve en büyük aşımını (overshoot) hesaplamak için gerekli y'nin maksimum değeri y_{max} ve kararlılık durumu y_{kd} değerleri Şekil 3.3'de gösterilmiştir.

Şekil 3.3'ten görüleceği gibi T_s=0.2018, y_{max}=1.3532 ve y_{kd}=1'dir. Buradan en büyük aşım değerinin yüzdesi $\frac{y_{\max} - y_{kd}}{y_{kd}} * 100$, formülünden hareketle, 35.32 bulunur.

Ancak gerçekte sistemlerin koşulları bu kadar ideal değildir. Sisteme etkiyen dış etkileri temsilen, sisteme bir gürültü(bozulma) etkisi eklemek, sistemin simülasyonunun gerçeğe daha yakın hale getirmektedir. Sistemin gürültü eklenmiş hali Şekil 3.4'de gösterilmektedir.



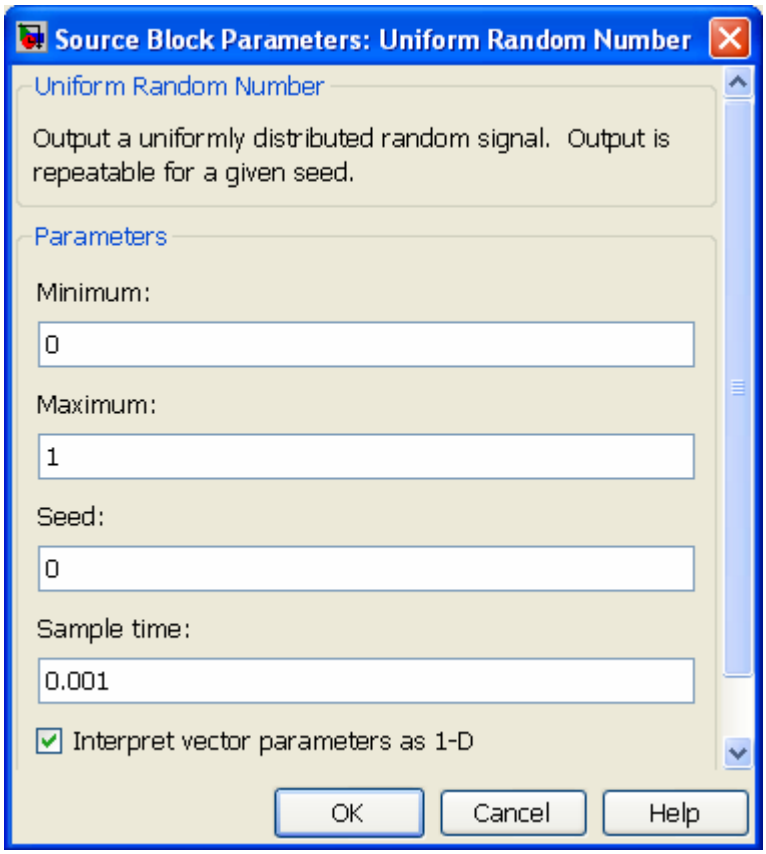
Şekil 3.4 Sisteme gürültü eklenmiş bir Ters Sarkacın PID Kontrolü Matlab Modeli



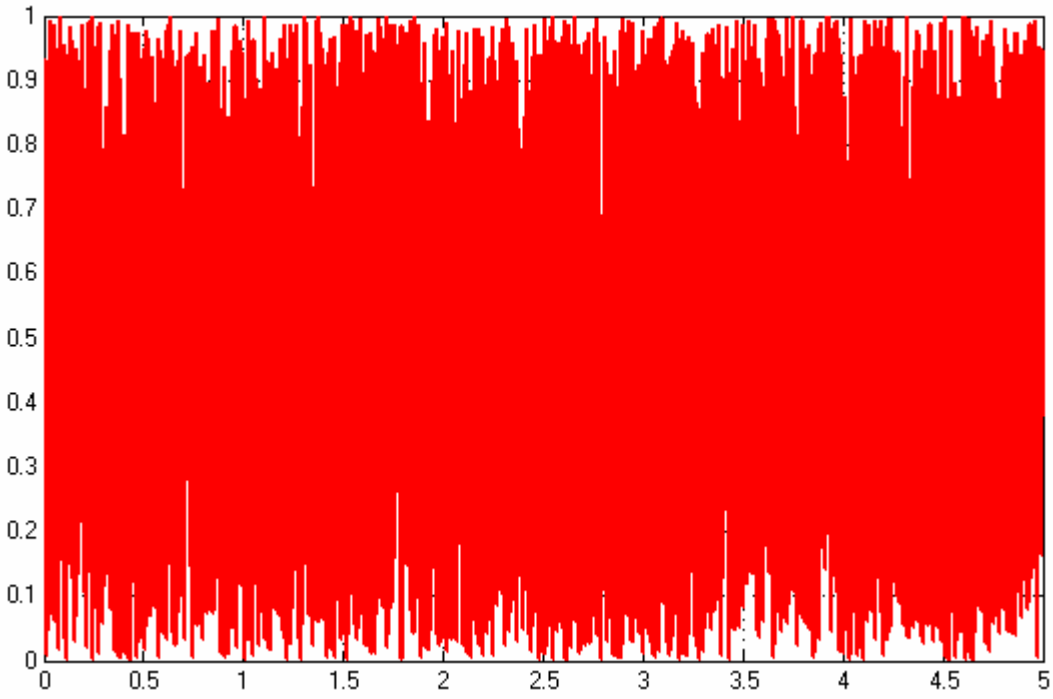
Şekil 3.5 Ters Sarkacın PID kontrol modeli Matlab çıkışı

Sistem modeli çıkışı Şekil 3.5'te verilmiştir. Burada görüleceği gibi, gürültü etkisi eklendikten sonra sistemin kararlılık noktası değişmekte ve her ne kadar sistem belli bir kararlılığa ulaşsa da sinyalde bozulma devam etmektedir.

Burada sisteme eklenen gürültünün parametreleri ve gürültü sinyali Şekil 3.6 ve Şekil 3.7'de verilmiştir.



Şekil 3.6 Sisteme eklenen gürültünün blok parametreleri



Şekil 3.7 Sisteme eklenen gürültü sinyali

4. KONTROL SİSTEMLERİNDE KULLANILAN AĞLAR

4.1 Kontrol Sistemlerinde Kullanılan Ağların Türleri

Haberleşme ağları 1970'lerde sayısal kontrol sistemleri içinde tanıtıldı. O zaman itici güç otomobil sektörüydü. İletişim ağı kurma nedenleri, kablolama masrafını, sistemlerin modüler kullanımını, ve sistem kurulumunun esnekliğini azaltmaktı. Ondan sonra, iletişim ağlarının çeşitli tipleri geliştirildi. İletişim protokolleri, endüstriyel haberleşmeler (örn. FIP ve PROFIBUS) , otomotiv haberleşmeler (örn. CAN), "diğer"makine haberleşmeleri (örn.1553B ve IEC tren haberleşme ağı), genel amaçlı ağlar (IEEE LANve ATM-LAN) ve bir dizi araştırma protokolleri (örn. TTP) şeklinde gruplandırılabilir, [27] Endüstriyel haberleşme gerçek-zaman kontrol uygulamaları içindir, ama bazı uygulamalarda diğer ağların kontrol için kullanılması gerekebilir. Örneğin, eğer diğer ağ halihazırda başka fonksiyonlar için kullanılıyorsa, kontrol için de bu ağı kullanmak uygun maliyetli olabilir. Endüstriyel haberleşme yalnızca düşük-seviyeli cihazların iletişimi için yapılmaktadır. Eğer yüksek-seviye fonksiyonu, örneğin bir iş istasyonu, bağlanırsa, diğer ağlar daha uygun hale gelebilir. Çok sayıda haberleşme protokolü ve endüstriyel haberleşme vardır. Burada bazılarının kısa bir özeti verilen en sık kullanılan endüstriyel haberleşmeleri için, bkz. [3, 28, 29].

4.1.1 FieldBus Protokolü

FieldBus, çeşitli büyük uluslararası otomasyon şirketleri içeren 100'den fazla üye firmayla birlikte kar amacı gütmeyen bir organizasyon olan Endüstriyel Haberleşme organizasyonu tarafından geliştirildi. Fieldbus, 31.25 kbit/s ve 1 Mbit/s olmak üzere, iki hız değeri için serbest bırakılır. 2.5 Mbit/s'lik veri yolu hızıyla daha hızlı bir veri yolu duyurulmaktadır. 31.25 kbit/s düşük hızlı veri yolu, bağlantı değiştirilmeden geleneksel 4-20 mA analog sinyallerin değiştirilmesi için tasarlanmıştır. Her veri yolunun 32 cihazı olabilir. Hiyerarşik ağ yapısı kullanılarak daha fazla cihaz bağlanabilir. Veri yoluna erişim, Link Aktif Zamanlayıcı, LAS, olarak adlandırılan bir merkezi veri yolu zamanlayıcı tarafından kontrol edilir. Fieldbus'ın yapılandırılması sırasında, veri yolu üzerindeki tüm cihazlar, ihtiyaç duyduğu bilgi ve bilgiye ihtiyaç duyulan zaman olan LAS'ı bildirir. Çalışma sırasında LAS, veri yayınlamak için, bir program kullanarak veri yoluna cihazları söyleyecektir. Tüm aboneler bu veriyi eş zamanlı alacaktır. Programda, programsız mesajlar için boş zaman ayrılmıştır. Bir sistem küresel saati,

fieldbus üzerine de dağıtılır. Dağıtılmış saat, 1ms içinde zamanı bilmek için aygıtların bağlanmasına müsaade eder [3].

4.1.2 FIP (Fabrika Araçları Protokolü)

FIP, Fransız, Alman ve İtalyan şirketler grubu tarafından geliştirildi. FIP bir çift bükülü iletken kullanır ve iletim hızları veri yolunun mekansal boyutuna bağlı olarak 31.25k/bit/s'den 2.5 Mbit/s'e kadar olabilir. 1 Mbit/s'lik bir iletim hızı için veri yolunun maksimum uzunluğu 500m'dir. Bir FIP ağı içinde düğümlerin maksimum sayısı 256'dır.

Bir FIP-ağında, düğümlerden biri veri yolu hakemi gibi davranır. Veri yolu hakemi ağ üzerinde kendi bilgisini yayınlamak için dönüşümlü olarak ağ üzerindeki tüm düğümleri yoklar. Düğümleri ilgilendiren bilgi gönderildiğinde etkin olmayan düğümler iletişimi dinler ve tanır. Veri tabanının periyodik olarak güncellendiği yerde FIP-ağı dağıtılmış bir veri tabanı olarak görülebilir[3].

4.1.3 PROFIBUS (Süreç Endüstriyel Haberleşmesi)

PROFIBUS bir grup Alman şirketi tarafından geliştirildi ve şuanda bir Alman standardıdır. Bir ekranlı çift bükümlü, iletken olarak kullanılır. İletim hızı 9.6 kbit/s'den 500 kbit/s'ye kadar olabilir. Veri yolunun maksimum uzunluğu 1200m'dir. Ağa en fazla 127 istasyon bağlanabilir. PROFIBUS mesajları 256 byte'a kadar çıkabilir. PROFIBUS bir belirteç-geçirme ağıdır. Düğümler aktif ve pasif düğümler olarak ikiye ayrılır. Belirteci tutan düğümün ağ üzerine veri göndermek için izni vardır. Belirteç, ağ üzerinde aktif düğümler arasından geçirilir. Aktif düğümler, belirteci tutarlarken iletilirler. Pasif düğümlerin, ağ üzerine veri göndermesine izin verilmesi için bir aktif düğüm tarafından adreslenmeye ihtiyacı vardır [3].

4.1.4 CSMA/AMP (CAN), (Denetleyici Alan Ağı)

CSMA/AMP, Mesaj Önceliğinde Keyfiyetli Taşıyıcı Algılı Çoklu Erişim'in kısaltmasıdır. CAN Alman şirketi Bosch tarafından araçlar için üretildi ve daha sonra otomasyon endüstrisinde kullanıldı. CAN ilk endüstriyel haberleşmelerden biriydi ve halen çoğu üretici tarafından arabalarda kullanılmaktadır. CAN, ISO 11898 ve 11519-1 standartlarıyla tanımlanır. Veri yolundaki iletim hızı programlanabilir. Eğer veri yolu 50m'den uzun değilse iletim hızı 1 Mbit/s'dir. Veri yolu 50m'den uzunsa iletim hızı 500 kbit/s'dir. Eğer kablo kalitesi düşükse, toplu üretilmiş araçlarda görüleceği gibi,

maksimum iletim hızı düşebilir. Düğümlerin sayısında herhangi bir sınır yoktur. Eğer veri yolu müsaitse bir düğüm istediği zaman iletimi başlatabilir. Eğer birkaç düğüm aynı anda iletimi denerse hakemlik devreye girer. Yüksek öncelikli mesaj göndermeyi deneyen düğüm veri yolunu kullanma sırasını elde eder. Mesajlar için 2^{29} farklı öncelik seviyesi vardır. CAN-denetleyicileri genellikle, bir mesaj gönderildiğinde bir kesintiye neden olmak için programlanabilir. Bu özellik, uygulamak için basitçe denetleyiciden çalıştırıcıya olan gecikmenin, $\tau_k^{dç}$, boyutunun geri yayılımını yapar [3].

CAN'da haberleşme esnasında eğer ağ meşgulse, gönderici, ağ müsait oluncaya kadar bekler. Eğer bir çarpışma meydana gelirse (yine, iki iletim 1 mikro saniye içinde başlarsa), en yüksek öncelikli mesaj (en düşük öncelikli sayıya tekabül eder) iletilmeye devam edecektir. Aynı önceliğe sahip iki mesajı aynı anda iletecek olursanız, hangisinin öncelikli iletileceğine dair keyfi bir seçim yapılır (Gerçek CAN uygulamalarında, bütün gönderilen düğümlerin, mesaj önceliği görevi olan eşsiz bir kimliği vardır) [1].

4.1.5 CSMA/CD (Ethernet) Ağı

Ethernet en çok kullanılan lokal alan ağı teknolojilerinden (LAN) biridir. Veriyi 10Mbit/s yada 100Mbit/s hızla iletir. Ethernet gerçek-zaman iletişimleri için uygun değildir. Bununla birlikte kurulu ethernetin yaygın oluşu, gerçek-zaman kontrol sistemlerinde kullanılmak için Ethernet'i cazip kılar. Hiçbir merkez veri yolu denetleyicisi yoktur, onun yerine Ethernet CSMA/CD (Çarpışma Algılamalı Taşıyıcı Algılayıcı Çoklu Erişim), adlandırılan bir veri yolu erişim metodunu kullanır. Ağa göndermeden önce istasyon, kanalı dinler, ve kanal boşta göründüğünde iletim başlar. Yani eğer ağ şebeke meşgul ise, gönderici şebeke müsait oluncaya kadar bekler. Bir mesaj 1 mikrosaniye içinde diğerine iletilirse bir çarpışma meydana gelir (Bu 200 metre kabloda yayılım gecikmesine karşılık gelir; kablo için iki yada daha fazla düğüm işlevsizken, çarpışmalarını sadece oluşmasının mümkün olmasından dolayı gerçek rakam o kadar önemli değildir) Örneğin eğer birkaç istasyon veri yoluna gönderimi başlatırsa, çarpılma tespit edilir. Bu durumda çarpışan istasyonlar geri çekilir, ve rasgele bir beklemeden sonra yeniden iletimi dener. Bir çarpışma meydana geldiğinde gönderici,

$$t_{geridönüş} = \frac{en\ küçük\ çerçeve\ boyutu}{veri\ hızı} * R \qquad [4.1]$$

vasıtasıyla bir zaman için geri dönüş yapacaktır, burada,

$$R = rand(0.2^K - 1) \qquad [4.2]$$

(ayrık tek düze dağılım) ve K arka arkaya çarpışma sayısıdır.(ama en fazla 10-yeniden iletim sayısında bir üst sınır yoktur.). CSMA/CD için bilinmelidir ki, minimum yapı boyutu 0 olamaz.

Bekledikten sonra, düğüm tekrar tekrar dener. Bir örnekte iki düğümün kendi iletimin bitirmek için üçüncü bir düğümü beklediği yerde, öncelikle olasılık 1 ile, daha sonra olasılık $\frac{1}{2}$ (K=1) ile, daha sonra $\frac{1}{4}$ (K=2) vb. ile çarpışacaktır.

Sınırsız sayıda istasyon Ethernet'e bağlanabilir. İstasyonların sayısı altı byte adresle sınırlandırılmıştır. İlk üç byte satıcı ID olarak kullanılır, ve son üç byte satıcı tarafından tanımlanır, böylece tüm Ethernet ara yüzü benzersiz bir adrese sahip olur. Bir Ethernet çerçevesi, yada paketi 64'le yaklaşık 1500 byte arasında uzunluktadır [1, 3].

4.1.6 Round Robin (Token Bus) Ağı

Ağdaki düğüm, her bir yapıyı iletmek için dönüşür(en düşükten en yükseğe düğüm sayısı). Dönüşümden sonra, ağ bir süre işlevsizdir

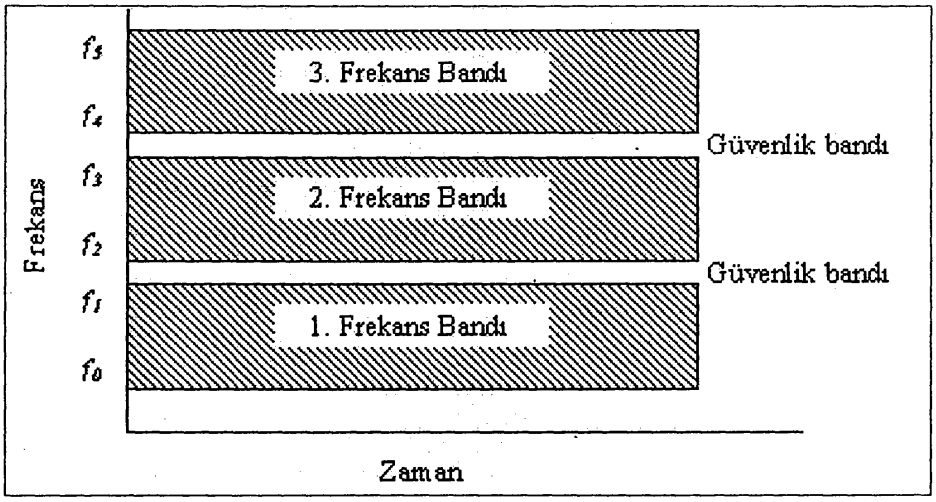
$$t_{mesgul} = \frac{en\ küçük\ çerçeve\ boyutu}{veri\ hızı} \qquad [4.3]$$

sıradaki düğüm bir belirteç geçmeyi temsil eder [1].

4.1.7 FDMA Ağı

FDMA, Frekans Bölüşümlü Çoklu Erişim'in kısaltmasıdır. FDMA, frekans bandı paylaşımı ile haberleşme kaynaklarının ortak kullanılmasını öngörmektedir. FDMA yöntemiyle, her işaretin iletimi için farklı bir frekans bandı kullanılarak birden fazla işaretin birbirine karışmadan aynı haberleşme ortamından iletilmesi mümkün olmaktadır. Frekans bandının bölüşümlü olarak kullanılması Şekil 4.1'de bir haberleşme kaynağının frekans-zaman uzayı paylaşımı biçiminde gösterilmektedir. Frekans spektrumu, belirli sayıdaki frekans bandına bölünmekte ve kullanıcıların iletim ihtiyacına göre kullanabilecekleri frekans bantları (haberleşme kanalları) atanmaktadır.

Örneğin, Şekil 4.1'de, frekans bantları arasında işaretlerin birbirine karışmadan geri ayrıştırılabilmesine olanak sağlayan güvenlik bantları bırakılmakta ve f_0 ve f_1 frekansları arası birinci bant, f_2 ve f_3 frekansları arası ikinci bant ve f_4 ve f_5 frekansları arası üçüncü bant olarak belirlenerek aynı anda üç farklı işaretin iletimi sağlanabilmektedir. FDMA sisteminde, frekans bantları genelde kullanıcı ihtiyacına bağlı olarak uzun süreli veya kalıcı olarak ayrılmaktadır.



Şekil 4.1 FDMA sisteminde haberleşme kaynaklarının ortak kullanımı[6]

Karışma olmaması için FDMA ile çoğullanacak işaretlerin bant genişliği mutlaka sınırlı olmalıdır ve bu nedenle FDMA ancak sürekli-zaman işaretlerinin çoğullanmasında kullanılabilmektedir. Dolayısıyla FDMA tipik olarak analog işaretlerin çoğullanması için kullanılmaktadır. Sayısal işaretler ise ancak modülasyon ile sürekli bir dalga biçimine dönüştürüldükten sonra (örneğin taşıyıcı dalgası kullanarak) FDMA ile çoğullanabilmektedir.

FDMA'da her alıcının, kendisine gönderilen işaretin hangi frekans bandından geleceğini bilmesi gerekmektedir. Alıcıda tipik olarak bir bant-geçiren süzgeç kullanılarak alınmak istenen işaret haberleşme ortamındaki diğer işaretlerden ayrıştırılmaktadır. Bu nedenle eğer frekans bantları geçici bir süreyle tahsis ediliyorsa ağ bağlantısının kurulum aşamasında verici ve alıcıya iletim için tahsis edilen frekans bandı bildirilmektedir.

Frekans bölüşümlü çoklu erişimin üstünlükleri

- Sistemin çok basit ve kolay tasarlanabilir olması,

- Zaman paylaşımı olmayıp tüm süre tek bir işaretin iletimine ayrıldığından sayısal modülasyonlarda sembol zaman diliminin daha fazla olması ve bunun sonucunda simgeler arası karışmanın daha az etki yapması,
- İletim sürekli olduğundan eşzamanlamanın kolay olup, çerçeve yapısı ve eşzamanlama için gönderilmesi gereken kontrol verisinin oldukça az olması,

olarak sayılabilmektedir.

Frekans bölüşümlü çoklu erişimin sakıncaları

- Frekans spektrumunda bırakılması gereken güvenlik bantlarının ağ kaynaklarının israfına neden olması,
- Ağ kaynaklarının paylaşım biçimi kolay bir şekilde değiştirilemediğinden yöntemin esnek olmaması,
- Frekans bandının kullanıcılara atanması sonucunda bir kullanıcı, iletim yapmasa dahi, ilgili kanalın daha yüksek bant genişliğine ihtiyaç duyan başka kullanıcılar tarafından kullanılmasının mümkün olmaması,
- Güvenlik bantlarının dar tutulabilmesi için yüksek maliyetli dar bantlı süzgeçlere ihtiyaç duyulması,

olarak sayılabilmektedir [6].

FDMA'da farklı düğümlerin iletimi, tamamen bağımsızdır ve hiçbir çarpışma meydana gelmez. Bu modda, ekstra bir özellik vardır.

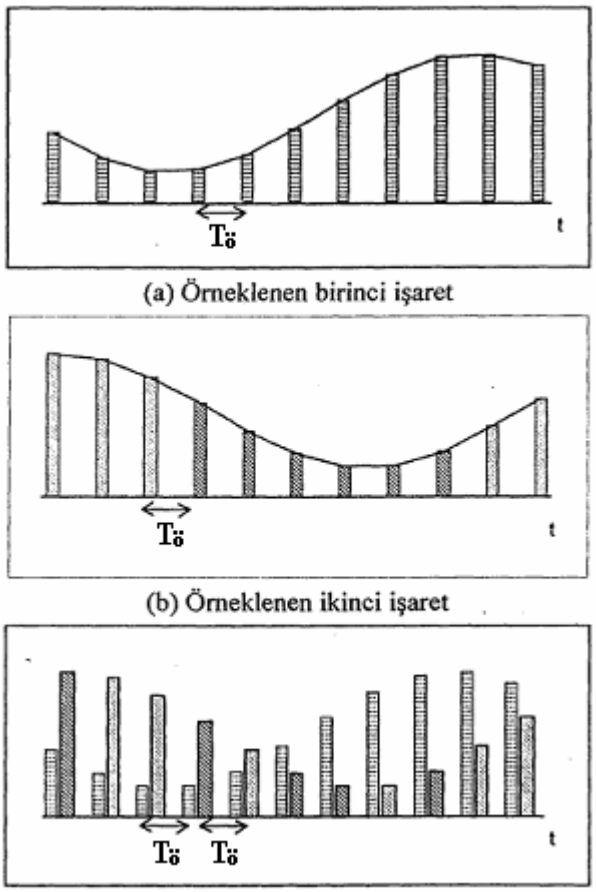
Bant genişliği Tahsisi En az bir olmak üzere gereken toplam gönderici düğümleri için paylaşımın bir vektörüdür. Burada,

Göndericinin gerçek bit hızı= Ayrılan bant genişliği * Veri hızı [4.4]

olarak hesaplanmıştır[1].

4.1.8 TDMA (TTP) Ağı

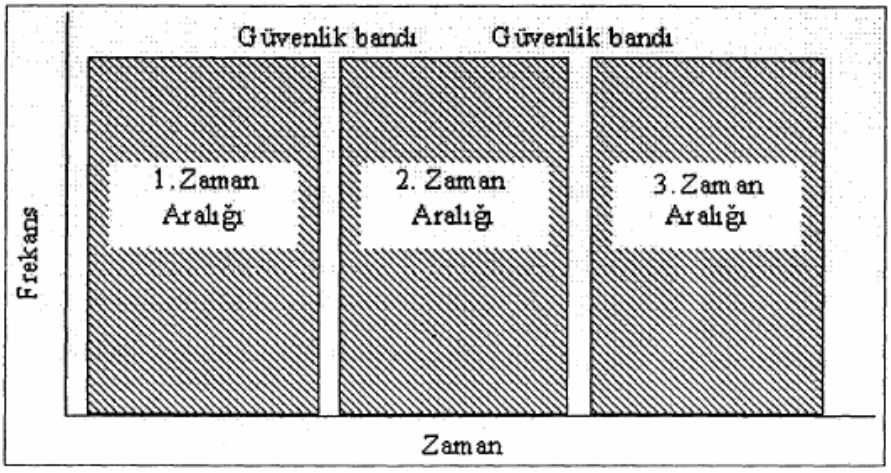
Zaman bölüşümlü çoklu erişim (TDMA) ağında, sayısal işaretlerin örnekleri arasındaki zaman aralığının diğer işaretlerin örneklerinin iletimi için değerlendirilmesi sayesinde zaman temelli paylaşım sağlanmakta ve bunun sonucunda haberleşme kaynaklarının ortak kullanılmasına olanak tanınmaktadır. Şekil 4.2'de gösterilen TDMA yönteminin mantığı, işaret örneklerinim zaman uzayında çoğullanarak sırayla iletilmesine dayanmaktadır.



Şekil 4.2 İki sayısal işaretin TDMA ile çoğullanmasına örnek[6]

TDMA sisteminde Şekil 4.3'de görüldüğü gibi zaman uzayı, zaman dilimlerine bölünmekte ve her bir sayısal işaret için farklı bir zaman dilimin kullanılması sonucu haberleşme kaynağının ortak kullanımı sağlanırken aynı zamanda işaretlerin birbirlerine karışması önlenmektedir. Bu sayede birden fazla sayısal işaretin aynı frekanstan farklı zaman dilimleri kullanarak iletim yapmasına olanak tanınmaktadır. Olası küçük senkronizasyon hatalarına karşı, şart olmamakla beraber, zaman dilimleri arasında güvenlik bantları bırakılabilmektedir.

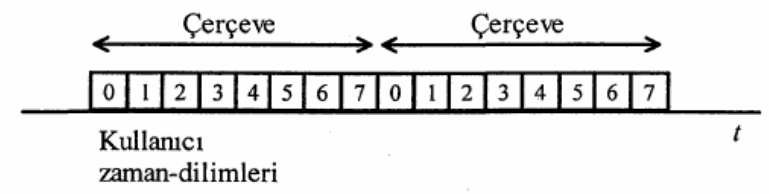
Avrupa'da kullanılan TDMA ağı 64 kbit/s bilgi transfer edecek şekilde oluşturulmaktadır. Bu nedenle en alt seviyedeki çoğullama seviyesinde her biri 8 bit'lik 32 adet zaman dilimi kullanılmaktadır. Amerika'da ise en alt seviyede 24 kanalın TDMA ile çoğullanması sonucu 1.544 Mbit/s oranında bir sayısal işaret elde edilmektedir. Çoğullanmış olan işaretlerin üst seviyelerde tekrar çoğullanması sayesinden çok seviyeli çoğullama gerçekleştirilebilmektedir.



Şekil 4.3 TDMA sisteminde haberleşme kaynaklarının ortak kullanımı[6]

Önemli bir nokta, TDMA yaklaşımlarının temelde işaret örnekleri arasında kalan zaman boşluklarının diğer işaret örneklerinin iletimi için kullanılması sonucu gerçekleştirilmesi nedeniyle sadece örneklenmiş, yani sayısal işaretler için kullanılabilmesidir. Bu nedenle, sürekli zaman analog bilgi işaretleri için TDMA kullanılması mümkün olmamaktadır.

TDMA'da her kullanıcı, kendisine ayrılan zaman-diliminde iletim yapabilmektedir. Kullanıcıya ayrılan zaman diliminde ağ kaynakları o kullanıcıya ayrılmış durumdadır. Zaman dilimlerinin tahsis biçimi çoğunlukla önceden belirlenmiş olup periyodik biçimde tekrar eden bir yapı şeklindedir ve bu yapının her periyodu bir çerçeve olarak adlandırılmaktadır. Şekil 4.4'de bu yapı gösterilmektedir. En temel TDMA sisteminde her kullanıcıya bir adet zaman-dilimi tahsis edilmektedir. Kullanıcıların iletim ihtiyaçlarına göre zaman-dilimlerinin dinamik olarak tahsis edilmesine olanak sağlayan genelleştirilmiş TDMA sisteminde ise ihtiyaç durumuna göre boş zaman dilimleri mevcut kullanıcılara tahsis edilerek kullanıcıların birden fazla zaman dilim kullanmasına izin verebilmektedir. Bu yaklaşım, özellikle değişken hızlarda haberleşme gerektirebilen veri iletimi uygulamalarında sistem kaynaklarının daha verimli kullanılmasına olanak tanımaktadır. TDMA sisteminin problemsiz çalışması için tüm kullanıcıların eşzamanlama içerisinde olması gerekmektedir ve bu nedenle her kullanıcı ne zaman ve ne kadar bir süre için iletim yapacağını bilmektedir.



Şekil 4.4 TDMA sisteminde zaman dilimi ve çerçeve yapısı [6]

Zaman bölüşümlü çoklu erişimin üstünlükleri

- Esnek iletim oranlarının destekleniyor olması (örneğin 32 kbps zaman dilimleri bulunduğunda 64 kbps hızında bir kullanıcıya iki adet zaman dilimi tahsis edilebilmektedir),
- Değişken bit hızı iletimlerine uygun olması (her kullanıcıya tahsis edilen zaman dilimi sayısı, çerçeveden çerçeveye değiştirilebilmektedir),
- Süzgeç kullanımının daha esnek olması,

olarak sayılabilmektedir.

Zaman bölüşümlü çoklu erişimin sakıncaları

- Kesin bir eşzamanlamaya gereksinim duyulması,
- Zamanlama hataları veya gecikme değişimlerine karşı bilgi iletiminin yapılmadığı güvenlik sürelerine (veya güvenlik bitlerine) gereksinim olması,
- İletim hızı arttığından uyarlanır denkleştirmeye ihtiyaç duyulması

olarak sayılabilmektedir[6].

TDMA ağında her düğümün yalnız planlanan yuva içindeki bant genişliğinin %100'e sahip olmasının dışında çalışması FDMA'ya benzer. Eğer bir yuvaya tam bir yapı iletilemeyecekse, iletim herhangi bir ekstra ceza olmadan sıradaki zamanlanmış yuvaya doğru devam eder. Diğer protokollerde olduğu gibi ek yükün her yapıya ekleneceğini unutulmamalıdır. İlave nitelikler;

Yuva boyutu (bits) Gönderilen yuvanın boyutudur. Bundan dolayı yuva zamanı aşağıdaki gibi verilir;

$$t_{yuva} = \frac{yuva\ boyutu}{veri\ hızı} \qquad [4.5]$$

Zamanlama Döngüsel bir gönderme programı belirtilen gönderici düğüm ID'nin(1…düğümlerin sayısı) vektörüdür. Sıfır, aynı zamanda izin verilmiş bir düğüm

ID'dir, bu şu anlama gelir; bu zaman diliminde hiçbir şeyin iletilmesine izin verilmez[1].

4.1.9 Switched Ethernet Ağı

Switched Ethernet'te, ağdaki her düğüm merkezi bir geçiş için kendi tam çift yönlü bağlantısına sahiptir. Sıradan bir ethernetle karşılaştırıldığında, bir Switched Ethernet'te ağ segmentinde hiçbir zaman çarpışma olmaz. Anahtar, bir tampon içinde gönderilmiş mesajları depolar ve daha sonra doğru hedef düğümlerine iletir. Bu ortak program, *saklama ve iletme* olarak bilinir.

Anahtar içindeki çoğu mesaj aynı düğüm için gidecekse, FIFO sırasına göre iletilirler. Ya anahtar içinde tutulan tüm mesajlar bir kuyruk olur ya da tüm çıkış segmentleri bir kuyruk olur. Yoğun trafik ve uzun mesajlar durumunda, anahtar bellek dışı çalışabilir. Aşağıdaki seçenekler Switched Ethernet ile ilişkilidir[1]:

4.1.9.1 Toplam Anahtarlama Belleği (bit)

Bu, anahtarda saklanan mesajlar için uygun bellek miktarıdır. Anahtardaki mesaj tamamen alındığı zaman, mesaj uzunluğuna eşit miktarda bir bellek tahsis edilir. Tam mesaj, nihai hedef düğümüne ulaştığı zaman, aynı hafızanın tahsisi kalkar [1].

4.1.9.2 Anahtarlama Tampon Türü

Bu ayar, belleğin anahtar içinde nasıl tahsis edildiğini açıklar. Ortak tampon şu anlama gelir ki; tüm mesajlar, tek bir FIFO kuyruğunda saklanır ve aynı bellek alanında paylaşılır. Simetrik çıkış tamponları şu anlama gelir, bellek, anahtara bağlı her çıkış segmenti için bir adet olan n adet eşit parçaya bölünmüştür. Bir çıkış sırası beleği dolduğunda, belirlenmiş kuyrukta daha fazla mesaj depolanamaz [1].

4.1.9.3 Anahtarın Taşma Davranışı

Bu seçenek, anahtar bellek dışı çalıştığında ne olacağını açıklar. Mesajın tamamı anahtar içine alındığında, bu silinir. Yeniden iletim, şu anlama gelir; anahtar, mesajı yeniden göndermeyi denemesi gereken gönderme düğümünü daha sonra bildirir. Düşme, şu anlama gelir; hiçbir bildirim yapılmaz-mesaj sadece silinir [1].

4.1.10 FlexRay Ağı

FlexRay'de iletim döngüsü üç art arda segment içerir. Birincisi, statik, segment TDMA protokolü gibi çalışır. Düğümler, bir programa göre veri iletimine dönüşür. Bir düğüm yalnız bunu yapmak için programlandığında iletilebilir. Eğer düğümün iletecek bir şeyi yoksa, ağ, düğüm programına göre değiştirilinceye değin işlevsizdir.

İkincisi, dinamik segment uygun zamanlamayla bir dizi küçük mini-yuva'ya bölünür. Her mini-yuva içinde planlanan düğüm iletimi başlatabilir ama sadece iletim, dinamik segmentin sonundan önce bitirilir. Statik segmentin aksine, iletim mini-yuva üzerinde olduğu zaman iletmeye devam edebilir. Planlanan düğüm, iletim için önemsiz olsaydı, ağ, bir mini-yuvanın boyutu olan çerçevenin iletimi için geçen sürede işlevsiz olacaktı. Unutulmamalıdır ki, bir mesaj iletilebilecek kadar büyük olabilir ve böylece, dinamik segmentte gönderim mesajları için gönderilen düğümü bloklayabilir.

Üçüncüsü, işlevsiz(*meşgul)*, segment ağın hiçbir şey iletemediği yerdir. Bu segment hem pencere sembolünü hem de FlexRay içinde ağın işlevsiz zamanını simüle etmekte kullanılır. FlexRay protokolün ek parametreleri aşağıda verilmiştir.

Slotsize [bits] Statik segment için yuvanın boyutu (bit)

Static schedule Statik segment için iletim planı

Dynamic schedule Dinamik segment için iletim planı

Mini slot size [bits] Dinamik segment için mini-yuva'nın boyutu (bit)

Network idle time [bits] İşlevsiz segment boyutu (bit)

Statik segment, uzunluk(Statik zamanlama)*Yuva boyutu saniye uzunluğundadır. Dinamik segment, benzer olarak, uzunluk(Dinamik plan)*Mini yuva boyutu saniye uzunluğundadır. Ağ daha sonra, $\frac{ag\ mesgul}{veri\ hızı}$ saniye için işlevsizdir. Buna ek olarak, tüm göndericiler, mesajın statik yada dinamik segmentte iletilmesinin gerekip gerekmediğinin belirtildiği yerde ekstra alana sahiptir [1].

4.1.11 PROFINET IO Ağı

PROFINET IO, 3 aralıklı, eşzamanlı RT Sınıf 3 ve RT Sınıf 1/NRT içeren döngüler gönderir. RT Sınıf 2 bu uygulamada desteklenmemektedir [1].

4.1.11.1 Eş Zamanlama

Eş zamanlı aralıklar esnasında, hiçbir düğüm mesaj gönderemez yada alamaz. Eş zamanlı aralıkların zamanı $\frac{Es\ zamanlama\ uzunlugu}{veri\ hızı}$ saniyedir[1].

4.1.11.2 RT Sınıf 3/IRT

Eş zamanlamanın ardından aralık RT Sınıf 3/IRT (Eş ve Gerçek Zamanlı) aralıktır. İletim sırası **IRT planı** tarafından belirlenir. Plan, 5 sütunlu ve bir IRT aralığı esnasında mesaj gönderecek kadar satırlı matris olarak gösterilir. İlk sütun iletilmesi gereken mesajın kimliğini (ID) belirler. Mesaj kimliği (ID) 0'dan büyük bir tam sayı olmalıdır. İkinci ve üçüncü sütun, mesajın hangi düğümden ve hangi sırayla gönderilmesi gerektiğini belirler. Dördüncü sütun IRT aralığının başlangıcına göre mesajın saniye bazında ne zaman gönderileceğini belirler. Beşinci sütun mesajın göndericiden alıcıya yayılımının ne kadar zaman sürdüğünü belirler. Her bir düğümdeki son gelen mesaj kaydedilir ve her bir IRT aralığında yeniden yayılır. Yayın IRT aralığında desteklenmez[1].

4.1.11.3 RT Sınıf 1/NRT

Üçüncü ve son aralık RT Sınıf 1/NRT (Gerçek Olmayan Zaman) aralığıdır. Aralığın uzunluğu, NRT uzunluk [bit] parametreleri ayarlanarak belirlenir. Düğüm bağlantı grafiğinde etiketlenmiş alanda, kullanıcı farklı düğümlerin nasıl kullanılacağını belirlemek zorundadır. Düğüm bağlantı grafiği, ağda düğümlerin var olduğu eşit satır ve sütun sayılı bir kare matriste gösterilmelidir. Matristeki her element (x,y) hangi düğümün bir mesajı, henüz x düğümündeyken, y düğümüne giden yola nasıl iletildiğini gösterir. Bu şekilde, ağ üzerinde gönderici düğümden alıcı düğüme birden fazla olası yol varsa bile tüm mesajlar deterministik bir şekilde iletilecektir.

Her bir düğüm 4 port olarak kabul edilir ve böylece 4 veya daha az düğüme bağlanabilir. Gelen bir mesaj, eğer ID'si 0'a eşit ise NRT mesaj olarak kabul edilir.

Eğer bir mesaj boş bir anahtar kuyruğuna sahip bir düğüme iletiliyorsa, mesaj bu düğümde kesilecektir. Bu şu anlama gelir ki, yalnız adres bitleri, boş anahtarın mesajı iletimini başlatması ile düğümden önce okunur. Adresin ilk 114 bitten oluştuğu kabul edilir.

PROFINET IO, FIFO yerine önceliği sayesinde kuyrukları sıralamak için imkan sunar. Protokol, Anahtarlamalı Ethernetli 2 parametre alanı sunar; Toplam anahtar belleği ve Anahtar taşma davranışı [1].

4.1.12 802.11b/g (WLAN) Protokolü

IEEE 802.11b/g günümüzde pek çok diz üstü ve mobil cihazlarda kullanılmaktadır. Protokol, bazı değişiklikler ile CSMA/CA'ya dayanmaktadır. Simülasyonda, bir paket iletim şöyle modellenmiştir: bir paketi iletmek isteyen düğüm, ortamın müsait olup olmadığını kontrol eder. Eğer ortam müsaitse ve 50 μs için bu halini korursa iletim işleme konur. Diğer yönden, eğer ortam müsait değilse rasgele bir geri-zaman seçilir ve aynı çarpışan da olduğu gibi azaltılır. Bir düğüm iletimi başlattığında, bu, aynı ağ üzerinde hesaplanan tüm diğer düğümlerle ve $\frac{1}{d^a}$ yol kaybı formülü ile hesaplanan tüm bu düğümlerin sinyal seviyesi ile ilişkilidir.

Eğer alıcı düğümdeki sinyal seviyesi alıcı sinyal eşiğinden büyükse, sinyali tespit etmenin mümkün olduğu varsayılır. Bu durumda, daha sonra sinyal-gürültü oranı (SNR) hesaplanır ve blok hata oranını(BLER) bulmak için kullanılır. SNR'ı hesaplarken tüm diğer iletimleri arka plan gürültüsüne eklemeyi unutmamalıdır. BLER (mesajın uzunluğu ile beraber), mesaj içindeki bit hatalarını hesaplar ve eğer bit hatalarının yüzdesi kodlama eşiğinden küçükse, kanal kodlama şemasının, mesajı tamamen yeniden yapabildiği varsayılır. Eğer halihazırda diğer düğümlerden alıcı düğüme devam eden yayın varsa ve kendi SNR'ı yenisinden daha düşük ise tüm bu mesajlar çarpışma olarak adlandırılır. Ayrıca, diğer gönderim halinde olan ve gönderme düğümü halen kendi iletimiyle gideceği yere varan yayınlar var ise bu mesajlar da çarpışma olarak adlandırılabilir. Unutulmamalıdır ki, bir gönderme düğümü, mesajın çarpışma olup olmadığını bilemez, bu nedenle, tanınan mesajlar MAC protokol katmanı üzerinden gönderilir. Gönderme düğümü açısından bakıldığında, kayıp mesajlar ve mesaj çarpışmaları aynıdır, yani hiçbir tanımlama alınmaz. Eğer tanımlı zaman aşımı süresi içinde hiç tanımlama alınmaz ise, mesaj, bir çekişme penceresi içinde rasgele geri çekilme zaman bekledikten sonra yeniden gönderilir. Çekişme pencere boyutu, belli bir mesajı her yeniden iletim için iki katına çıkarılır. Eğer ortam meşgulse veya en az 50μs için müsait değilse geri çekilme zamanlayıcı durdurulur. Gönderici mesajdan vazgeçinceye kadar, iletimin yalnız "Deneme" sınır sayısı kadar şansı vardır ve daha fazla yeniden iletilmeye çalışmaz [1].

4.1.13 802.15.4 (ZigBee) Protokolü

ZigBee sensörlerle ve akılda kolay kontrol ağları ile tasarlanmış bir protokoldür. Oldukça düşük bir bant genişliğine ve ayrıca oldukça düşük bir güç tüketimine sahiptir. 802.11b/g gibi CSMA/CA'a dayanmasına rağmen, çok daha basittir ve protokoller aynı değildir.

ZigBEE içinde paket iletim modeli WLAN'la benzerdir, ama MAC prosüdürü farklıdır ve aşağıdaki gibi modellenmiştir:

1. Başlatılamadı:
   NB=0
   BE=macMinBE
2. Tamsayılar[0,2BE-1] içinde geri çekilme periyotlarının rasgele bir sayısı için geciktir.
3. Ortam müsait mi?
   Evetse: gönder
   Diğer durumda: 4'e git
4. Geri çekilme sayaçları güncellemesi:
   NB=NB+1
   BE=min(BE+1,aMaxBE)
5. NB>macMaxCSMABackoff?
   Evetse: paketi aç
   Diğer durumda: 2'ye git

Değişken isimleri daha kolay karşılaştırma yapmak için standart olarak alınmıştır. İsimlere dair kısa birer açıklama aşağıda verilmiştir.

NB Geri çekilme sayısı

BE Geri çekilme üssü

macMinBE CSMA/CA algoritmasında geri çekilme üssünün minimum değeri. Öntanımlı değer 3'tür.

aMaxBE CSMA/CA algoritmasında geri çekilme üssünün maksimum değeri. Öntanımlı değeri 5'tir.

macMaxCSMABackoffs CSMA/CA algoritmasının bir kanal erişim hatası vermeden önce yapılacak teşebbüsleri geri çekilmelerinin maksimum sayısıdır. Ön tanımlı değeri 4'tür [1].

5. AĞ ÜZERİNE KONTROL YASALARI

5.1 Otomatik Kontrol Sistemleri

Kapalı döngü geri beslemeli kontrol sistemleri, denetlenen sistem ve denetim organları olarak ikiye ayrılmaktadır. Denetlenen süreç hazır bir sistemdir ve buna uygun bir denetim organı seçmek sistemin kararlılığı açısından önemlidir. Denetim organları, kendi içinde karşılaştırıcı veya hata seçici, denetim organı, motor eleman ve algılayıcı birimlerden meydana gelir. Karşılaştırıcı, istenen giriş değeri ile çıkış değerini karşılaştırarak aradaki farkı hata sinyali olarak belirler. Denetim organı bu hatayı giriş olarak alır ve hatanın şekline ve kendi denetim etkisine bağlı olarak motor elemanına kumanda sinyali gönderir. Motor elemanı kendisine gelen kumanda sinyali ile harekete geçerek hatayı azaltacak veya sıfırlayacak yönde bir değişim meydana getirir. Karşılaştırıcıda giriş büyüklüğü ile sistemin denetlenen çıkış büyüklüğünün aynı birim cinsinden olması gerekmektedir.

Uygulamada denetim organlarının fiziksel yapısı başlangıçtan bugüne kadar mekanikten, sayısal elektroniğe doğru bir evrim geçirmiştir. Her ne kadar denetim organlarının ilk örnekleri mekanik elemanlarından oluşmaktaydıysa da çeşitli nedenlerden dolayı (sürtünme, boşluk vb.) mekanik sistemlerin hassas bir denetim sağlanması mümkün değildir. Endüstriyel alandaki gelişmelere paralel olarak zaman içinde elektronömatik denetim elemanları kullanılmış ancak maliyet ve denetim elemanlarının yapısal karmaşıklığı nedeniyle zamanla terk edilerek, yeni gelişmeler neticesinde elektronik denetim elemanları yani denetleyiciler yaygın olarak kullanılmaya başlanmıştır.

Denetleyicilerin ilk uygulamaları benzeşik tümleşik elektronik devrelerden meydana gelmiştir. Ancak programlanabilme özelliğinin kısıtlı olmasından dolayı yerini zamanla sayısal elektronik denetleyiciler yani mikrodenetleyicilere bırakmıştır. Mikrodenetleyicilerin en önemli avantajı programlanabilir ve esnek olmalarıdır. Bu nedenle endüstriyel uygulamalarda mikrodenetleyiciler sıklıkla tercihe edilmektedir.

Basit, tek döngü geribeslemeli denetim sistemi denetleyicilerinde kullanılan denetim yasası veya denetim algoritması çeşitli yönlerden sınıflandırılabilir. Denetim etkisine göre denetim algoritması:

- Kesikli veya iki konumlu denetime etkisi
- Sürekli denetim etkisi

Olarak sınıflandırılabilir. Diğer yandan sürekli denetim etkisi de kendi içinde:

- Orantı denetim etkisi (P etki)
- İntegral denetim etkisi (I etki)
- Türev denetim etkisi (D etki)

PID denetimi, orantı, integral ve türev (P,I ve D) temel denetim etkilerini birleştiren sürekli denetim yasası ve algoritmasıdır. Yani bu denetimde hata mevcut olduğu sürece denetim komutu da sürekli mevcuttur. PID denetimi genelde çok basit yapıdadır ve endüstriyel uygulama alanında yeterli ve uygun bir denetim sağlar. PID denetimi daha çok doğrusal ve basit yapıda tek döngülü sistemlerde kolaylıkla uygulanabilmektedir.

Denetlenecek sistemin dinamik yapısına bağlı olarak PID denetiminde yer alan üç temel denetim etkisinin mümkün olan en basit bileşimleri kullanılır. Bunlar P, PI, PD ve PID denetimi biçiminde olabilir [5].

5.1.1 P Denetimi (Orantı Denetimi)

$T_i \to \infty$ ve $T_d \to 0$ halinde PID denetleyicisi yalnızca orantı etki ile çalışır. Orantı kazancı K_p'nin ayarı ile denetim organının denetim duyarlılığı artırılabilir. Orantı denetim, hatanın ansal değişimine orantılı denetim etkisi sağlar. Bu ifadenin matematiksel bağıntısı:

$$m(t) = K_p e(t) \qquad [5.1]$$

şeklindedir. Burada K_p orantı kazancı veya orantı etki parametresi olarak adlandırılır. Orantı denetim fonksiyonu:

$$\frac{M(s)}{E(s)} = K_p \qquad [5.2]$$

elde edilir. Orantı denetiminde, sabit bir K_p orantı kazancında, m(t) denetim etkisinin şiddeti hatanın şiddetine bağlıdır ve herhangi bir anda hatanın şiddeti ne kadar büyük olursa, orantı denetiminin hatayı düzeltme etkisi o kadar olur. Buna karşılık, hata küçüldükçe denetim etkisinin şiddeti küçülür ve hatanın belli bir en küçük değerinden sonra denetim etkisinin şiddeti hatayı tamamen düzeltmeye yetmez. Bu durumda geri beslemeli sistemde belli bir kalıcı durum hatası oluşabilir. Orantı kazancının büyük

tutulması ile, meydana gelen bu kalıcı durum hatası küçük tutulabilir. Ancak bu durumda da sistem aşırı salınımlı veya kararsız hale gelebilir. Bu nedenle orantı denetimi en iyi sonucu kararlı sistemlerin denetiminde verir.Burada kararlı sistemden kastımız, edinilen kazanç artımı karşısında kararsız duruma geçmeyen sistemdir [5].

5.1.2 PI Denetimi (Orantı ve İntegral Denetimi)

Orantı denetimine, integral denetimi ilavesi yapılarak veya PID denetiminde D türev denetiminin devre dışı bırakılması ile elde edilen PI tipi denetim organı yapısı nispeten basit olup endüstriyel uygulamaların büyük bir çoğunluğunda PI denetimi kullanılmaktadır. İntegral denetim biriktirilmiş hataya orantılı bir denetim çıkışı oluşturarak, hatanın zaman içinde sıfırlanmasını sağlar. Bu integral denetimin olumlu tarafıdır. Öte yandan, integral denetim, hatayı biriktirerek düzelttiği için, denetim etkisi ve hatayı düzeltmesi zaman alır. Bu nedenle PI denetim, yavaş çalışan bir denetim sistemidir. Ayrıca integral etkiden dolayı, paydaya katılan "s" çarpanı integral etkili denetime 90°'lik bir faz gecikmesi verir. Bu da geri beslemeli sistemin kararlılığını kötüleştirir.

Standart PI denetiminin matematiksel ifadesi:

$$m(t) = K_p\left(e(t) + \frac{1}{T_i}\int e(t)dt\right) \qquad [5.3]$$

buradan hareketle transfer fonksiyonu:

$$\frac{M(s)}{E(s)} = K_p\left(1 + \frac{1}{T_i s}\right) \qquad [5.4]$$

şeklindedir. Burada T_i integral zamanı küçüldükçe (diğer yandan $K_i=K_p/T_i$ integral kazancı büyür) hatanın giderilme hızı artmakta, ancak sistem aşırı salınımlı hale gelerek kararlılığı bozulmaktadır. Bu nedenle, uygun bir T_i integral zamanı seçerek, sistemin talebine uygun bir denetim sağlamak önemlidir [5].

5.1.3 PD Denetimi (Orantı ve Türev Denetimi)

Orantı denetimine, türev denetimi ilavesi yapılarak veya PID denetiminde I integral denetiminin devre dışı bırakılması ile elde edilmektedir. PD denetiminin matematiksel ifadesi:

$$m(t) = K_p\left(e(t) + T_d \frac{de}{dt} \right) \qquad [5.5]$$

buradan hareketle transfer fonksiyonu:

$$\frac{M(s)}{E(s)} = K_p\left(1 + T_d s\right) \qquad [5.6]$$

şeklinde elde edilir. Burada T_d türev denetimin zamanıdır. Türev denetimi aynı zamanda değişim oranı etkisi olarak da bilinir.

Türev denetimi, hatanın kendisi yerine hatanın değişimi üzerinde etkili olduğundan, hatanın değişimini önceden kestirerek, hatanın büyümesine meydan vermeden hatayı çok hızlı bir şekilde düzeltme özelliğine sahiptir. Türev denetiminin daha hata değişime başlar başlamaz harekete geçmesi, öngörü etkisi olarak da bilinir. Türev denetimi aynı zamanda geri beslemeli sisteme sönüm katar. Bu da sistemin cevap hızını değiştirmeden kararlılığını artırır. Diğer bir deyişle, türev denetimi açık döngü transfer fonksiyonu payına "s" çarpanı getirerek, sistem kararlılığı üzerinde iyileştirici bir etki yapar ve sisteme 90°'lik biz faz öndeliği getirir.

Bir sabitin türevi sıfır olduğundan, türev denetiminin zamanla değişmeyen, sabit kalan hata üzerinde bir etkisi yoktur. e(t)=sbt olması halinde $\frac{de}{dt} = 0$ ve dolayısıyla m(t)=0 olacağından, sabit kalan hatalar üzerinde türev denetimin etkisi olmaz. Bu nedenle türev etki yalnızca hatanın zamana göre değişimi karşısında etkili olduğundan, tek başına kullanımı tercih edilmez. Türev denetiminde T_d türev denetim zamanı büyüdükçe (aynı zamanda $K_d=K_pT_d$ türev kazancı büyür) sistemin cevap hızı artar ancak diğer yandan sistemin salınımı artarak kararlılığı bozulmaktadır. Türev etkinin diğer bir olumsuz yönü ise, ani hata değişimlerinde türev denetimin sağladığı denetim etkisi çok ani olmaktadır. Bu da sistemi, bilhassa yüksek frekans bölgesinde aşırı salınımlı hale getirmektedir [5].

5.1.4 PID Denetimi (Orantı, İntegral ve Türev Denetimi)

Uzun ölü zaman gecikmelerinin ortaya çıktığı süreç denetim sistemlerinde, PI denetimde integral denetiminin tamamlayıcısı olarak türev denetimi kullanılmaktadır. Düşük şiddetli bozucu girişlere maruz bir sistemde PI denetimi tek başına hatada meydana gelen değişimleri izlemeye ve düzeltmeye yetişemez. Bu durumda, türev denetim ilavesi, orantı kazancı ayarının daha yüksek tutulmasına olanak sağlayarak, denetim organı tepki süresini hızlandıracaktır. Aşırı salınımlı, sönümü düşük servomekanizmalarda PI denetim tek başına yeterli olamamaktadır. Bu durumda da türev denetim ilavesi, sistemde fazla bir kararsızlık problemi yaratmadan K_p orantı kazancının yüksek tutulmasını sağlayarak, sistemin kararsızlığa yatkınlığı önlenmiş olur. Böylece PID denetimi ile bir taraftan kalıcı-durum hatası sıfırlanırken diğer taraftan da sistemin geçici durum davranışı iyileştirilmiş olur.

Standart PID denetim yapısının blok şeması Şekil 5.1'de verildiği gibidir. Standart PID denetim yasasına bağlı m(t) denetim sinyali çıkışı:

$$m(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de}{dt} \qquad [5.7]$$

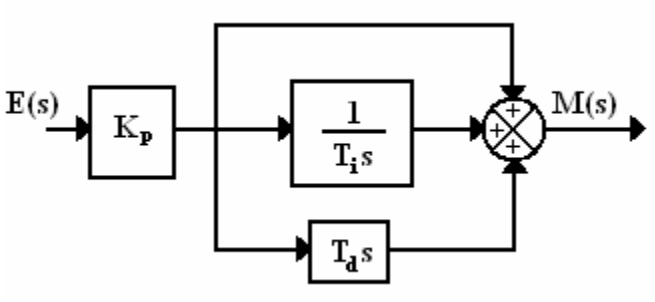
burada K_p parantezine alındığında:

$$m(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{de}{dt} \right) \qquad [5.8]$$

üç temel denetim etkisinin toplamı şeklinde ifade edilir ve buradan da transfer fonksiyonu:

$$\frac{M(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i s} + T_d s \right) \qquad [5.9]$$

olarak elde edilir.



Şekil 5.1 Standart (PID) denetimi

PID denetim, üç denetim etkisinin üstünlüklerini tek bir birim içinde birleştiren bir denetim etkisidir. İntegral etki, sistemde ortaya çıkabilecek kalıcı-durum hatasını sıfırlarken türev etki de, yalnızca PI denetim etkisi kullanılması haline kıyasla, sistemin aynı bağıl kararlılığı için cevap hızını artırır. Yani, PID denetim organı, sistemde sıfır kalıcı-durum hatası ile hızlı bir cevap sağlar [5].

5.2 Algoritmaların Ayarlama Metotları

PID denetleyicilerin tasarımı genel tasarım sorunundan farklıdır, çünkü denetleyicinin karmaşıklığı sınırlıdır. Genel tasarım yöntemleri denetleyiciye, süreç modeliyle eşleşen bir karmaşıklık sağlar. Kısıtlı karmaşıklığıyla bir denetleyiciyi elde etmek için süreç modellerini basitleştirebiliriz, böylece tasarım, PID denetleyiciyi verir, yada kompleks bir model için denetleyici tasarlayabilir ve PID denetleyiciye yaklaştırabiliriz. PID denetleyiciler için özel tasarım yöntemleri ortaya koymanın başka bir nedeni, basit tasarım yöntemine sahip olma arzusudur ki kontrol konusunda çok az bilgisi olan biri bile kullanabilsin. Durum, süreç bilgisinin gelişmesi ve daha kapsamlı hesaplamalar kullanılmasına izin vermesini mümkün kılan ayarlama araçları ve otomatik ayarların ortaya çıkmasıyla büyük ölçüde değişmiştir. Bu, PID denetleyicilerini, kontrol sistemleri tasarımı ana akımına daha yakın hale getirdi.

Bu bölümde, eski yöntemler ve yeni güçlü yöntemler arasında bir denge kurmak için çalışılmıştır. Bölüm 5.2.1'de, iyi ayarlamaya neden olmasa da PID kontrol uygulamasında büyük etkinliği olan Ziegler ve Nichols tarafından geliştirilen yöntemler anlatılmaktadır.

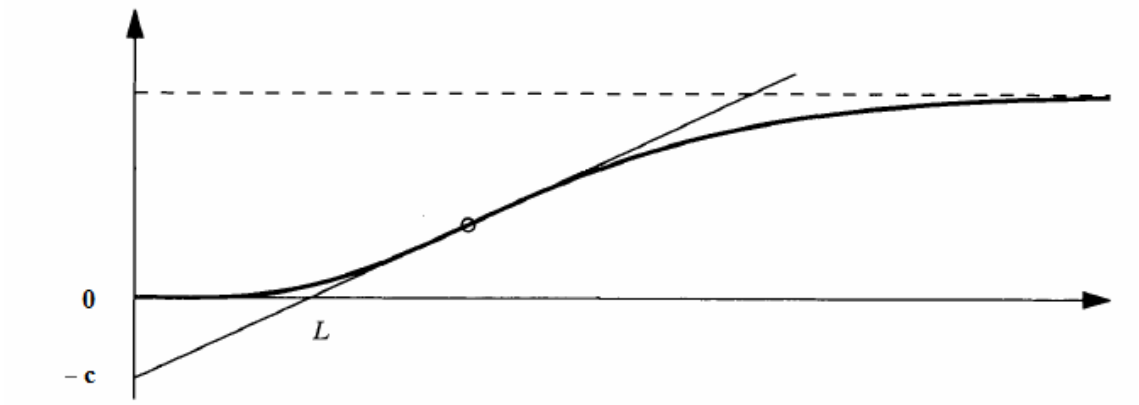
Genellikle, kapalı-döngü dinamiklerinin istenilen hedeflerini elde etmek için manuel ince ayarlı tasarım yöntemlerini tamamlamak gereklidir. Bu analitik ayarlama kuralları Bölüm 5.2.2'de tartışılmıştır. Bölüm 5.2.3, kontrol sistem tasarımında işimizi hayli kolaylaştıran MATLAB ile çözüm yöntemlerini sunmaktadır [4].

5.2.1 Ziegler-Nichols ve İlgili Yöntemler

1942'de Ziegler ve Nichols tarafından PID denetleyicileri hesaplamak için 2 klasik yöntem sunuldu. Bu yöntemler, ilk halleriyle yada bazı değişikliklerle halen yaygın olarak kullanılmaya devam etmektedir. Genellikle, denetleyici üreticileri ve süreç endüstrisi tarafından ayarlama işlemleri için temel oluştururlar. Yöntemler, süreç dinamiklerinin bazı özelliklerinin belirlenmesine dayanmaktadır. Basit formüller tarafından, denetleyici parametreleri özellikler cinsinden ifade edilir. Yöntemlere yaygın şekilde atıfta bulunulması şaşırtıcıdır çünkü yalnızca sınırlı durumlarda orta derecede iyi ayar sağlamaktadırlar. Tabi yöntemlerin basitliği ve temel kontrol derslerinde basit öğrenci alıştırmalarında kullanılabilmesi, tercih edilmesinin bir sebebi olabilir [4].

5.2.1.1 Basamak Tepkisi Yöntemi

Ziegler ve Nichols tarafından sunulan ilk tasarım yöntemi, açık döngü basamak tepkisi şeklinde süreç bilgilerine dayanmaktadır. Bu yöntem, basit bir süreç modelinin kullanıldığı yerde modelleme ve kontrole dayanan geleneksel bir yöntem olarak görülebilir. Basamak tepkisi, Şekil 6.1'de görüleceği gibi, c ve L parametreleriyle karakterize edilir.



Şekil 5.2 Ziegler-Nichols basamak yöntemindeki bir basamak tepkisinin karakteristiği[4]

Basamak tepkisinin eğiminin maksimum olduğu nokta ilk olarak tespit edilir ve bu noktadaki teğet çizilir. Teğetle koordinat ekseni arasındaki kavşaklar c ve L parametrelerini verir. Bölüm 2'de, kontrol edilen sürecin bir modeli, bu parametrelerden elde edilmişti. Bu, bir tamamlayıcı ve bir zaman gecikmesi vasıtasıyla bir süreci modellemeye tekabül eder. Ziegler ve Nichols, c ve L'nin fonksiyonları gibi doğrudan

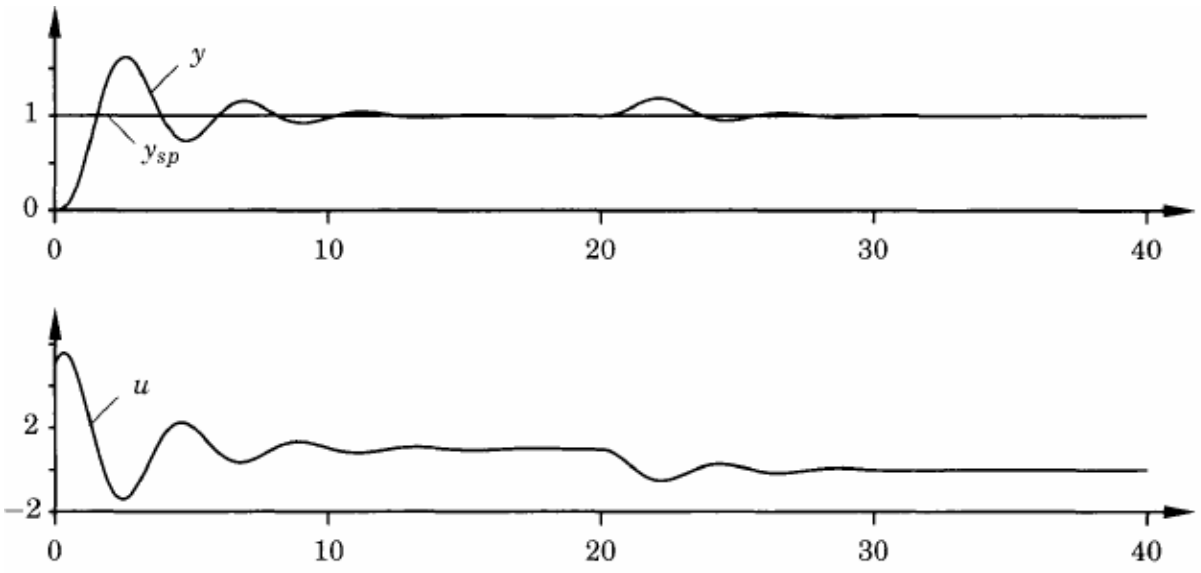
PID parametrelerini verir. Bunlar Çizelge 5.1'da verilmiştir. Kapalı-döngü sisteminin T periyodunun bir tahmini de çizelgede verilmiştir [4].

Çizelge 5.1 Ziegler-Nichols basamak tepki yöntemi için denetleyici parametreleri

Denetleyici	***cK***	***T_i/L***	***T_d/L***	***T_p/L***
P	1			4
PI	0.9	3		5.7
PID	1.2	2	L'2	3.4

Örnek 5.1 Ziegler-Nichols Basamak Tepki Yöntemi[4]

Ziegler-Nichols'ün yöntemi, $P(s)=\frac{1}{(s+1)^3}$ transfer fonksiyonlu bir sürece eklenir. Basamak tepkisi üzerine ölçümler c=0.218ve L=0.806 parametrelerini verir. Şimdi denetleyici parametreleri, Çizelge 5.1'den hesaplanabilir. Bir PID denetleyicinin parametreleri, K=5.50, T_i=1.61, ve T_d=0.403'dür. Şekil 5.3'de, yükteki bir basamak değişimini takiben ayar noktasındaki bir basamak değişimi için kapalı-döngü sistemlerin tepkisi gösterilmektedir. Denetleyicinin davranışı beklendiği gibidir. Basamak tepkisi için bozulma oranı dörtte birine yakındır. Yük bozulması için daha küçüktür. Ayar-noktası tepkisindeki aşma çok büyüktür. Bu, b ağırlığındaki ayar noktası vasıtasıyla artırılabilir.



Şekil 5.3 Ziegler-Nichols birim basamak yöntemiyle ayarlanmış bir PID denetleyici vasıtasıyla kontrol edilen, transfer fonksiyonu $\frac{1}{(s+1)^3}$ olan bir sürecin ayar-noktası ve yük bozulma tepkisi. Şemalar, ayar noktası y_{sp}, süreç çıkışı y, ve kontrol u'yu göstermektedir [4]

5.2.1.2 Frekans Tepkisi Yöntemi

Bu yöntem aynı zamanda süreç dinamiklerinin basitçe karakterize edilişine dayanmaktadır. Tasarım, Nyquist eğrisinin negatif gerçek eksenin kestiği yerde süreç transfer fonksiyonunun Nyqusit eğrisi üzerindeki noktasının bilgisine dayanmaktadır. Bu noktalar, K_{180} ve ω_{180} değerleriyle tanımlanır. Eskiden beri böyle kullanılmasından dolayı nokta, son nokta olarak anılmaktadır ve son kazanç ve son periyod olarak adlandırılan $K_u = \frac{1}{K_{180}}$ ve $T_u = \frac{2\pi}{\omega_{180}}$ ile tanımlanır. Bu parametreler, aşağıdaki şekilde tespit edilebilir. Bir denetleyiciyi sürece bağlayın, ve kontrol eylemi yalnız orantı etkili olacak şekilde parametreleri ayarlayın, yani $T_i = \infty$ ve $T_d = 0$. Süreç, salınımı başlatıncaya kadar, kazancı yavaşça artırın. Burada kazanç $K_u = \frac{1}{K_{180}}$ ve salınım periyodu $T_u = \frac{2\pi}{\omega_u}$'dur. Ziegler-Nichols, Çizelge 5.2'de gösterilen son kazanç ve son periyot açısından denetleyicinin parametreleri için basit formüller vermişlerdir. Kapalı döngü sistemin baskın dinamiklerinin T_0 periyodunun bir tahmini de çizelgede verilmiştir. Frekans tepkisi yöntemleri, denetleyici parametrelerinin bazı basit kurallarla birlikte süreç üzerine doğrudan deneylerle elde edildiği yerde, deneysel bir ayarlama işlemi olarak düşünülebilir. Orantılı bir denetleyici için kural, süreçte salınma başlayıncaya kadar kazancı basitçe artırmak ve daha sonra kazancı %50 oranında azaltmaktır [4].

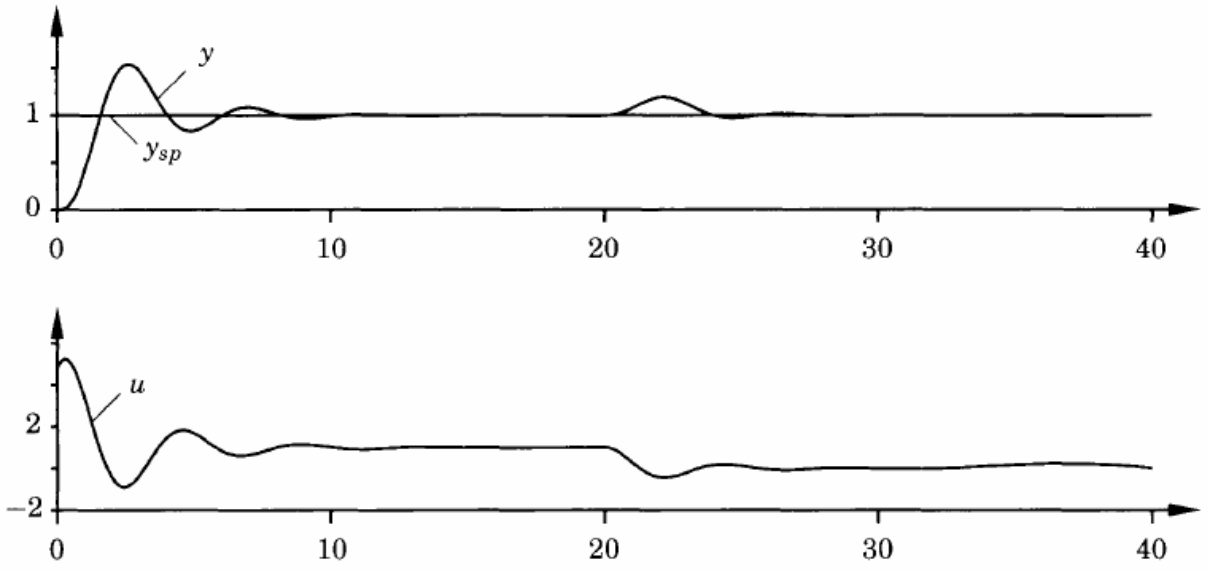
Çizelge 5.2 Ziegler-Nichols frekans tepkisi yöntemi için denetleyici parametreleri

Denetleyici	K/K_u	T_i/T_u	T_d/T_u	T_p/T_u
P	0.5			1.0
PI	0.4	0.8		1.4
PID	0.6	0.5	0.125	0.85

Örnek 5.2 Zeigler-Nichols Frekans Tepki Yöntemi [4]

Burada da Örnek 5.1'de sunulan aynı süreç üzerinde çalışılacaktır. Sürecin son kazancı K_u=8 ve son periyot $T_u = \frac{2\pi}{\sqrt{3}} = 3.63$ 'tür. Bir PID denetleyici için Çizelge 5.2'de verilen parametreler K = 4.8, T_i = 1.81 ve T_d = 0.44'dür. 5.1 tarafından verilen denetleyicinin sürece eklendiği zamanki yük bozulmaları ve kapalı-döngü ayar-noktası Şekil 5.4'te

gösterilmiştir. Frekans tepki yöntemiyle elde edilen denetleyicilerin parametreleri ve performansı, basamak tepki yöntemi vasıtasıyla elde edilene yakındır. Tepkilerin sönümlenmesi biraz daha iyidir.



Şekil 5.4 Ziegler-Nichols frekans tepki yöntemiyle ayarlanmış bir PID denetleyici vasıtasıyla kontrol edilen, $\frac{1}{(s+1)^3}$ transfer fonksiyonlu bir sürecin ayar noktası ve yük bozulma tepkisi. Şemalar ayar noktası y_{sp}, ve süreç çıkışı y, ve kontrol sinyali u'yu göstermektedir[4]

Ziegler-Nichols ayarlama kuralları aslında, yük bozulmaları için iyi tepkili sistemler vermesi için tasarlanmıştır. Kurallar, sonuçların manuel değerlendirmesiyle bir çok farklı sistemin kapsamlı simülasyonları vasıtasıyla elde edilmiştir. Tasarım kriteri, örneklerde görüldüğü gibi genelde çok büyük olan çeyrek genlik bozunum oranıdır. Birincil tasarım amacı yük bozulmalarını azaltmak olduğundan, tatmin edici bir ayar noktası tepkisi elde etmek için genellikle ayar-noktası ağırlığını dikkatli seçmek gereklidir.[4]

5.2.2 Analitik Ayar Yöntemleri

Denetlenen sistem (süreç) ve ölçme elamanı dinamik davranışlarının bilinmesi halinde, denetleyici türüne göre PID denetiminin K_p, T_i ve Td parametrelerinin en uygun değerini hesaplamak mümkündür. Bu hesaplamalarda bir takım optimizasyon ölçütleri kullanılır. Hesaplamaların teknik açıdan mümkün olmasına rağmen işlemler oldukça karışık ve zordur. Bununla beraber günümüzde hazır bilgisayar hesaplama ortamlarının

(MATLAB gibi) yaygınlaşmasıyla bu hesaplamalar kolay hale gelmiştir. Analitik veya teorik ayarda temel zorluk, sistem modelinin ve parametrelerinin tam olarak tespit edilememesinden kaynaklanmaktadır.

PID denetleyicisi parametrelerinin (K_p, T_i, T_d) en uygun değerleri bir optimizasyon ölçütü seçilerek ve bu ölçütü minimum yapan denetim parametre değerlerinin analitik yoldan hesaplanması ile belirlenebilir. Genelde performans göstergesi olarak bilinen optimizasyon ölçütü daha çok kapalı-döngü e(t) hata fonksiyonuna göre seçilir. Literatürde çok çeşitli performans göstergeleri tanımlanmış olup, bunların belli başlıları aşağıda verilmiştir.

$$I = \int_0^\infty e^2(t)dt, \quad I = \int_0^\infty te^2(t)dt, \quad I = \int_0^\infty |e(t)|dt, \quad I = \int_0^\infty t|e(t)|dt. \qquad [5.10]$$

En uygun performans göstergesi seçildikten sonra bu ifadeden istenen ayar parametrelerine göre kısmi türevler yoluyla

$$\frac{\partial I}{\partial K_p} = 0, \quad \frac{\partial I}{\partial T_i} = 0, \quad \frac{\partial I}{\partial T_d} = 0 \qquad [5.11]$$

K_p, T_i ve T_d için çözülür.

Sistemin kapalı-döngü çalışmasında arzu edilen değerlerini belirlemek ve bu değerleri sağlayan denetleyici parametrelerini tespit etmek yada deneysel ayar ile analitik ayar yönteminin birlikte kullanılması PID ayarında kullanılan analitik yöntemlerdendir.

Analitik yöntemlerde en kestirme yol ise Routh-Hurwitz kararlılık yöntemi yolu ile sistemin maksimum kazancını bulmak ve bu kazanca karşılık gelen sürekli salınım periyodunu belirlemektir. Elde edilen sonuçlara göre Zeigler-Nichols yöntemi ile denetim organı parametrelerinin kaba değerleri tespit edilir. Daha sonra denetim sisteminin kapalı-döngü çalışmasına ait çeşitli spesifikasyonlar ve simülasyon programları kullanarak denetim organının parametrelerinin ince ayarı yapılabilir. Simülasyon konusunda MATLAB gibi paket programlardan faydalanılabilir [5].

5.2.3 Matlab ile Çözüm

PID denetleyici tasarımında en uygun PID parametrelerini bulmak için Matlab üzerinde çeşitli yöntemler ve komutlar mevcuttur. Bu komutlardan biri "pidtune" komutudur.

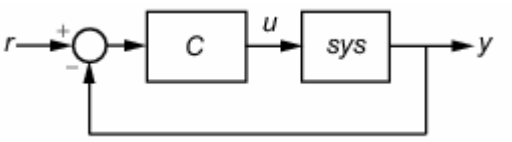
`C = pidtune(sys,type)`, *sys* tesisi için *type* türdeki bir paralel form PID denetleyici tasarlar.

`C = pidtune(sys,pidsys)`, *pidsys* denetleyici ile aynı tür ve forma sahip bir denetleyici tasarlar. Eğer *sys* ve *pidsys* ayrık-zaman modeliyse, `C` *pidsys* ile aynı ayrık integratör formülüne sahiptir.

`C = pidtune(sys,type,opts)` ve `C = pidtune(sys,pidsys,opts)`, *opts* ayarı seçeneği kullanan bir denetleyici tasarlar. *opts*'yi tanımlamak için `pidtuneOptions` komutu kullanılır.

`[C,info] = pidtune(...)` kapalı-döngü kararlılığı, açık-döngü kazanç geçiş frekansı ve faz payı ile ilgili ek bilgi içeren veri yapısı hakkında bilgi verir.

pidtune birim geri beslemeli tek bir kontrol döngüsünün ileri besleme yolunda bir denetleyici tasarlar:



Şekil 5.5 Temsili bir kontrol sistemi

Varsayılan ayarlarda, *type* girişi ile *pidtune* paralel formda bir PID denetleyiciye dönüşür. Standart formda bir denetleyici tasarlamak için, *pidsys* giriş değişkeni gibi bir *pidstd* kullanılır. *pidtune* otomatik olarak, performansı (tepki süresi) ve dayanıklılığı (kararlılık payı) dengelemek için PID kazancını ayarlar. Geçiş frekans sonucu veya faz payını elde etmek için *info* çıkış değişkeni kullanılır. Kendi geçiş frekansı ve faz payı hedeflerini belirtmek için *pidtuneOptions* kullanılır. Geçiş frekansı tepki süresi ile ters orantılıdır. Unutulmamalıdır ki, geçiş frekansının artırmak, tipik olarak sağlamlığı azaltır ve tam tersi durum da geçerlidir [32].

5.2.3.1 Giriş Değişkenleri

sys Denetleyici tasarımı için tesis modelinde ***sys***;

- Herhangi bir SISO LTI sistem (*ss*, *tf*, *zpk* yada *frd* gibi)

- Sistem Tanımlama Toolbox yazılımı kullanarak elde edilmiş herhangi bir SISO doğrusal modeli (*idarx*, *idfrd*, *idgrey*, *idpoly*, *idproc* yada *idss*)
- Bir sürekli yada ayrık zaman modeli
- Kararlı, kararsız yada bütünleşik. Kararsız kutuplu bir tesis, ancak, PID kontrolü altında kararlı olmayabilir.
- Zaman gecikmesinin herhangi bir türünü içeren bir model. Uzun zaman gecikmeli bir tesis, ancak, PID kontrolü altında yeterli performans elde edilmez.
- Tesis modellerinin bir dizisi. *sys* bir dizi ise, *pidtune* dizideki her tesis için ayrı bir denetleyici tasarlar.

Eğer tesisin kararsız kutupları varsa, ve *sys* aşağıdakilerden biriyse:

- Bir *frd* modeli
- I/O gecikmelerine dönüştürülemeyen bir iç zaman gecikmeli ss modeli

Daha sonra tesisteki kararsız kutupların sayısını belirlemek için *pidtuneOptions* komutu kullanılır.

Çizelge 5.3 Kontrol parametreleri formülleri [32]

Dizi	**Tür**	**Sürekli-Zaman Denetleyici Formülü**	**Ayrık-zaman Denetleyici Formülü (paralel form, ForwardEuler integrasyon yöntemi)**
'p'	Yalnızca orantı	K_p	K_p
'i'	Yalnızca integral	$\frac{K_i}{s}$	$K_i \frac{T_s}{z-1}$
'pi'	Orantı ve integral	$K_p + \frac{K_i}{s}$	$K_p + K_i \frac{T_s}{z-1}$
'pd'	Orantı ve türev	$K_p + K_d s$	$K_p + K_d \frac{z-1}{T_s}$
'pdf'	Türev döneminde birinci dereceden filtreli, orantı ve türev	$K_p + \frac{K_d s}{T_f s + 1}$	$K_p + K_d \frac{1}{T_f + \frac{T_s}{z-1}}$
'pid'	Orantı, integral ve türev	$K_p + \frac{K_i}{s} + K_d s$	$K_p + K_i \frac{T_s}{z-1} + K_d \frac{z-1}{T_s}$
'pidf'	Türev döneminde birinci dereceden filtreli, orantı, integral ve türev	$K_p + \frac{K_i}{s} + \frac{K_d s}{T_f s + 1}$	$K_p + K_i \frac{T_s}{z-1} + K_d \frac{1}{T_f + \frac{T_s}{z-1}}$

type Tasarım için denetleyicinin denetleyici türü Çizelge 5.3'teki denetleyici türlerinden biriyle belirtilebilir.

type girişi kullanıldığında, *pidtune* paralel (*pid*) formda bir denetleyici tasarlar. Eğer standart (*pidstd*) formda bir denetleyici tasarlanmak istenirse, *type* yerine *pidsys* girişi kulanılır.

Eğer *sys*, T_s örnekleme zamanlı bir ayrık-zaman modeliyse, *pidtune* aynı T_s örnekleme zamanlı bir ayrık zamanlı denetleyici tasarlar. Denetleyici, integral ve türev etkilerin her ikisi için de ForwardEuler ayrık integartör formüle sahiptir. Eğer farklı bir ayrık integratör formüle sahip bir denetleyici tasarlanmak isteniyorsa, *type* yerine *pidsys* girişi kullanılır.

pidsys *pid* yada *pidstd* denetleyicileri tasarlanmış denetleyicilerin özelliklerini belirtir. Eğer ***pidsys*** sağlanırsa, *pidtune*:

- *pidsys*'nin temsil ettiği türdeki bir denetleyici tasarımında.
- Eğer *pidsys* bir *pid* denetleyicisiyse, bir *pid* denetleyicisine dönüşür
- Eğer *pidsys* bir *pidstsd* denetleyicisiyse, bir *pidstd* denetleyicisine dönüşür.
- Eğer *sys* ayrık-zaman sistemi ise, *pidsys*'deki gibi aynı *Iformula* ve *Dformula* değerli bir denetleyiciye dönüşür.

opts *Pidtune* tasarım algoritması tarafından kullanılan ayarlama kriterleri belirten seçenek dizisidir. *opts* oluşturmak için *pidtuneOptions* kullanılmaktadır. Ayarlama kriteri aşağıdakileri içerir:

- *CrossoverFrequency*- açık döngü C**sys*'nin birinci 0dB geçişi için hedeflenen frekans ayarlanır.
- *PhaseMargin-pidtune* işlevi, bu değerden daha büyük yada bu değere eşit bir faz payı elde etmek için çalışır. Seçilen geçiş frekansı kullanılabilir faz payını kısıtlayabilir [32].

5.2.3.2 Çıkış Değişkenleri

C Denetleyici *sys* için tasarlanmaktadır. Eğer *sys* doğrusal modelin bir dizisi ise, pidtune, her bir doğrusal model için bir denetleyici tasarlar ve PID denetleyicilerin bir dizisine dönüştürür.

Denetleyici formu:

- Eğer *pidtune*'un ikinci değişkeni *type* ise, C bir *pid* denetleyicidir.
- Eğer *pidtune*'un ikinci değişkeni *pidsys* ise:
    - Eğer *pidsys* bir *pid* nesnesi ise, C bir *pid* nesnesidir.
    - Eğer *pidsys* bir *pidstd* nesnesi ise, C bir *pidstd* nesnesidir

Denetleyici türü: *type* değişkeni kullanıldığında, C genellikle *type*'la belirlenmiş türe sahiptir. *pidsys* değişkeni kullanıldığında, C genellikle *pidsys* ile aynı türe sahiptir. Bununla birlikte, her iki durumda da, algoritmanın *type* yada *pidsys* ile birlikte belirtilen daha düşük dereceden bir denetleyici kullanarak yeterli performans ve sağlamlık elde ettiği yerde, *pidtune*, belirtilenden daha az eyleme sahip bir C'ye döndürür. Örneğin, *type*, *pidf* olsa da C bir PI denetleyici olabilir.

Zaman etki alanı:

- C *sys* ile aynı zaman etki alanına sahiptir.
- Eğer *sys* ayrık-zaman modeline sahipse, C, *sys* ile aynı örnekleme zamanına sahiptir.
- Eğer pidsys belirtilirse, C, *pidsys* ile aynı *Iformula* ve *Dformula* sahiptir. Eğer hiç *pidsys* belirtilmezse, *Iformula* ve *Dformula*'nın her ikisi de ForwardEuler'dir.

Eğer *pidsys* belirtilirse, C ayrıca *pidsys*'den *Inputname* ve *Outputname* gibi LTI model özellikleri elde edilir.

info ayarlanan PID döngüsünün performansı ve sağlamlığı hakkında bilgi içeren veri yapısı. *info* alanı:

- *Stable* – Kapalı döngü kararlılığını gösteren Boolean değeri. Eğer kapalı döngü kararlı ise *stable* 1, aksi halde 0'dır.
- *CrossoverFrequency* – Açık döngü sistemi *C*sys*'nin birinci 0dB geçiş frekansı, raydan/saniye cinsinden.
- *PhaseMargin* – Ayarlanan PID döngüsünün faz payı, derece cinsinden.

Eğer *sys* tesis modelinin bir dizisi ise, *info*, her bir ayarlanan PID döngüsü hakkında bilgi içeren veri yapısının bir dizisidir [32].

Örnek 5.3 pidtune komutuyla PID tasarımı [32]

$G_c(s) = \dfrac{1}{(s+1)^3}$ sistemi için bir denetleyici tasarlayın.

İlk geçiş için basit bir PI denetleyci tasarlarsak:

Gc=zpk([],[-1 -1 -1],1); % tesisi tanımlar

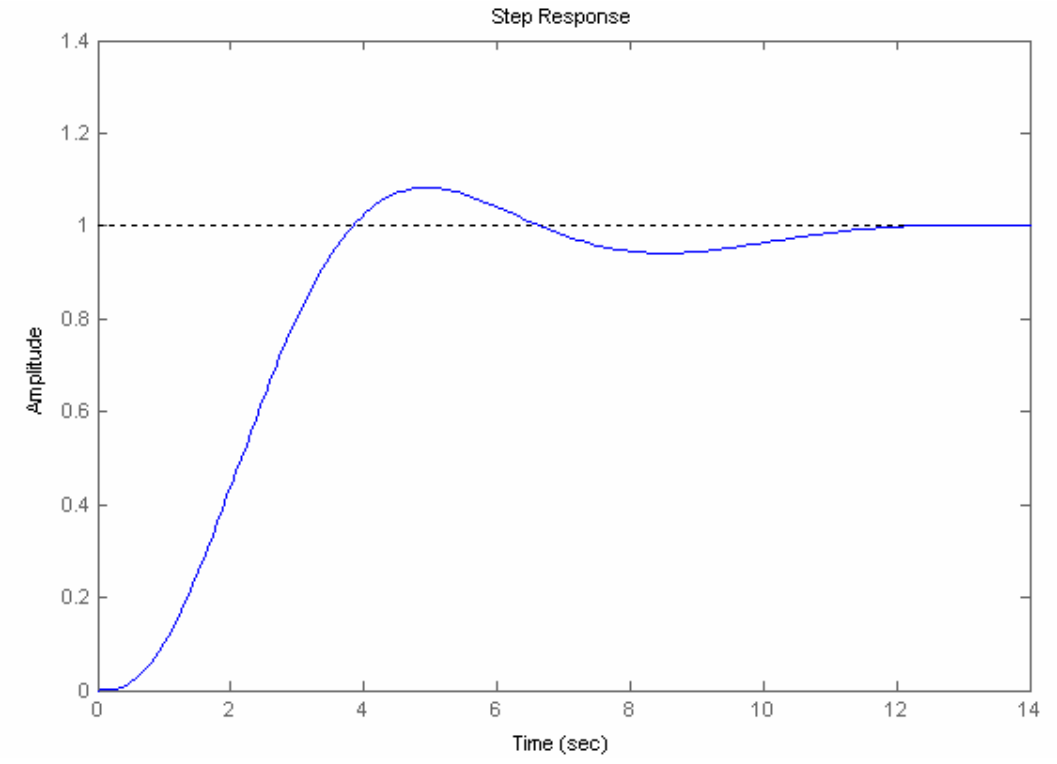
[C_pi,info]=pidtune(G_c,'pi')

Bu komutlar, bir PI denetleyiciyi temsil eden bir PID denetleyici içinde sonuçlanır. Ekranda şu çıkışı görürüz:

```
Continuous-time PI controller in parallel form:
           1
Kp + Ki * ---
           s
with Kp = 1.1367, Ki = 0.45408
info =
                Stable: 1
    CrossoverFrequency: 0.5205
           PhaseMargin: 60.0000
```

Ayarlama algoritması 0.52 rad/s'lik bir açık-döngü geçiş frekansı seçer. Denetlenen sistemin kapalı-döngü basamak tepkisini (referans izleme) incelersek:

```
Tpi = feedback(C_pi*Gc, 1);
step(Tpi)
```



Şekil 5.6 $\frac{1}{(s+1)^3}$ transfer fonksiyonuna sahip bir tesisin *pidtool* komutu ile PID tasarımı çıkışı

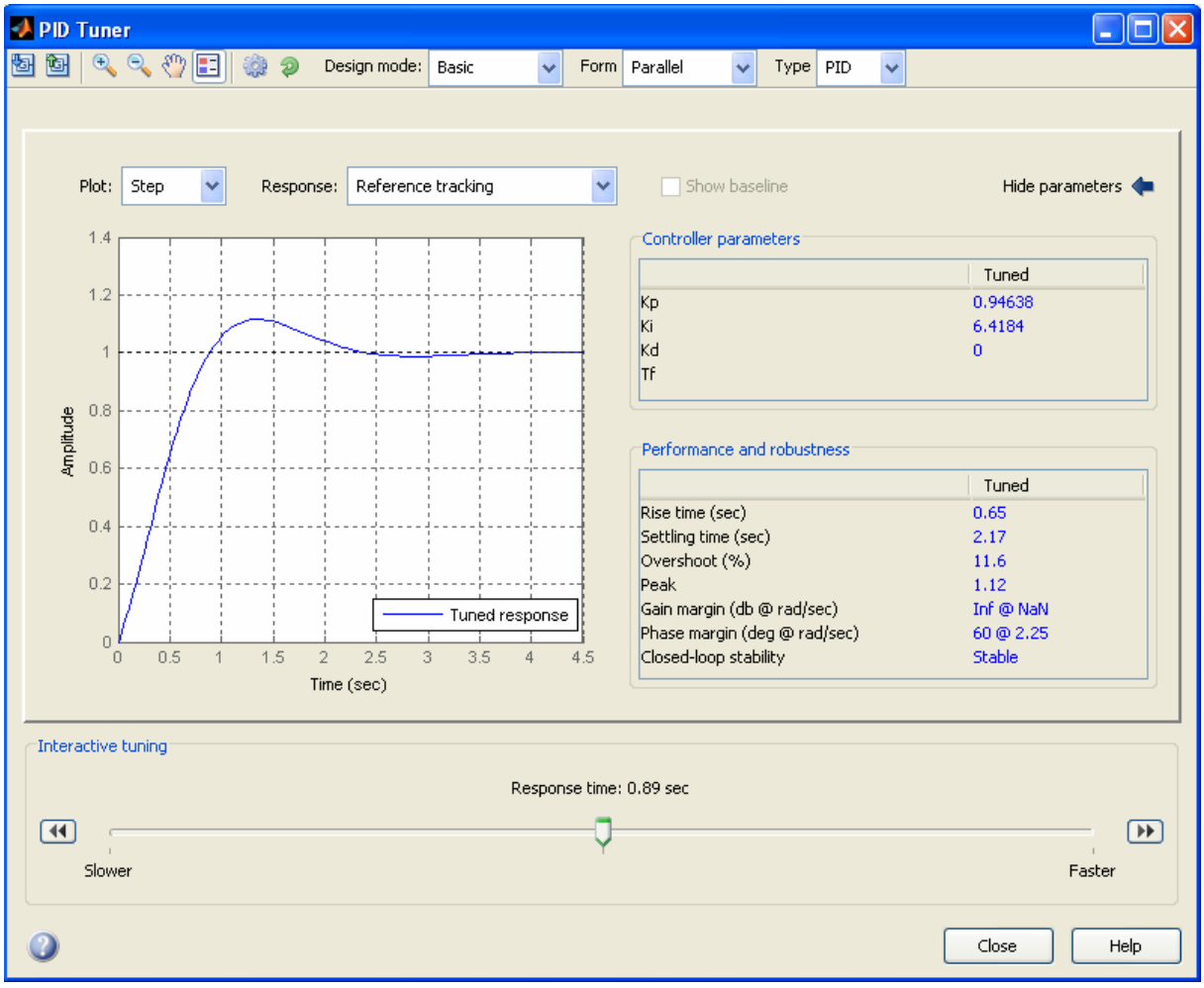
Benzer şekilde *pidtool* komutu kullanarak da PID tasarımı yapmak mümkündür.

Örnek 5.4

$\frac{1}{s+2}$ transfer fonksiyonu ile ifade edilen sistem için sürekli zamanlı bir PID denetleyici tasarlayın.

```
Gd=tf([1],[1 2]);
pidtool(Gd,'pid')
```

komutlarını yazdığımızda alacağımız çıkış ekranı Şekil 5.6 gibidir.



Şekil 5.7 $\frac{1}{s+2}$ transfer fonksiyonuna sahip bir tesisin *pidtune* komutu ile PID tasarımı çıkışı

6. KONTROL ALGORİTMALARININ TRUETIME TOOLBOX'DA GERÇEKLEŞTİRİLMELERİ

6.1 TrueTime Nedir?

Denetleyici görevler içeren çoklu-görevli gerçek-zamanlı kernellerin zamansal davranışının simülasyonunu yapmak için tasarlanan TrueTime, Henriksson ve arkadaşları tarafından yazılan bir Matlab/Simulink-tabanlı pakettir [12]. TrueTime simülasyon ortamı, iki simülasyon bloğu önerir; bir bilgisayar bloğu ve ağ bloğu. Her ikisi de olay-güdümlüdür. Bilgisayar bloğu, bir kontrol bilgisayarının aktivitesinin simülasyonunu yapar yani, kullanıcı tanımlı davranışlar ve kesme işleyiciler için görev yürütme ve anahtarlama. Ağ bloğu, mesaj yapısı gibi kullanıcı kaynaklı ve trafiği öncelikli kılmak için kullanılan bir öncelik fonksiyonuna dayalı bir bilgisayar ağının dinamiklerinin simülasyonunu yapar. Ağ bloğu, CAN tabanlı ve Ethernet tabanlı ağları içeren pek çok farklı tipte ağların simülasyonunu yapmak için yeterince esnektir.

Henriksson ve ark. bir Ethernet ağı üzerinde bir denetleyicili üç ters sarkaç iletiminin TrueTime simülasyonunu açıklarlar. Kendi simülasyonları, Ethernet ağının dinamiklerini içerirken, araştırma ve analizlerinin odağı optimal kontrol için görev ve alt görev zamanlaması üzerinedir.

TrueTime, AKS simülasyonunun bir metodunu sunarken, genel olarak ağ bloğu daha kompleks Ethernet ağlarının simülasyonu için yeterli değildir. Bu nedenle, keyfi karmaşık Ethernet ve IP tabanlı ağların simülasyonu için araçlar sağlayan diğer yapılar incelenecektir [2].

6.2 Neden TrueTime?

Tipik bir kontrol sistemi;

- bir endüstriyel denetleyici,
- kontrol edilen bir süreç
- bazı giriş/çıkış kanalları
- genel bir haberleşme ağı

içerir. Genellikle sık kullanılan simülatörlerde herhangi bir ağın modeli bulunmaz. Model basitleştirilmiştir ve yeterince güçlü ve hızlı bir iletişim ağına sahip olduğumuz varsayılır. Bu durum, kontrol sistemlerindeki iletişim ağından kaynaklanabilecek olumlu ve/veya olumsuz etkileri göz ardı etmemize neden olmaktadır.

Gerçek zaman sisteminde bir ağ, veri alışverişinde kullanılıyorsa, bu sistemin simülasyonundan doğru sonuçlar elde edebilmek için simülasyon şemasına, kullanılan ağ modelini de eklemek gereklidir. Kullanılan ağ tipi, özel iletişim kurallarını tanımlar ve süreç kontrolünde çıkış sonuçlarını değiştirebilecek nitelikte özgün davranışlar gösterir. Gecikmeler, veri kesilmeleri veya ağ yoğunluğu, gerçek zamanlı denetim döngüsünde, transfer edilen veri kalitesini düşürebilir. Bu nedenle simülasyon sonucunda gerçeğe daha yakın değerler elde edebilmek için, karmaşık veya gerçek zamanlı sistemlerin simülasyon modelleri, ağ modeli içermek zorundadır.

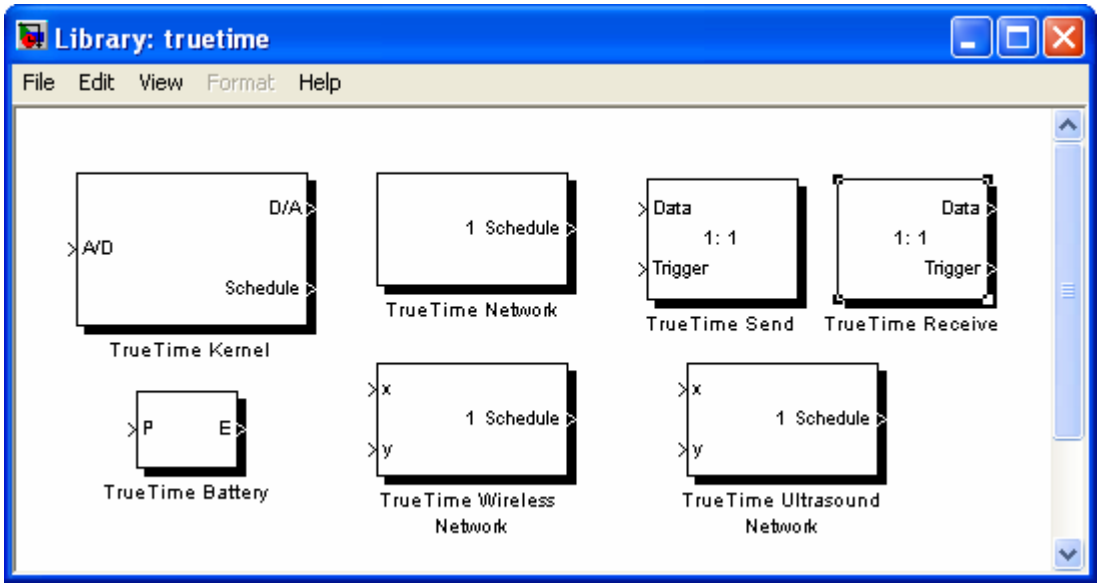
TrueTime, ağ tiplerinin sisteme etkisini anlamak için basit ve kolay bir yoldur.

6.3 TrueTime Toolbox'ına Bakış

TrueTime, gerçek-zaman kontrol sistemleri için Matlab/Simulink tabanlı bir Toolbox'tır Bu Toolbox, gerçek-zaman kernellerinde denetleyicilerin görevlerini yerine getirmesinin simülasyonunu kolaylaştırır. Görevler, M-dosyaları veya C++ fonksiyonları olarak yazılabilir veya kod fonksiyonları içeren blok şemaları çağrılabilir.

Her bir TrueTime Toolbox simülasyon şeması 3 önemli içerir:

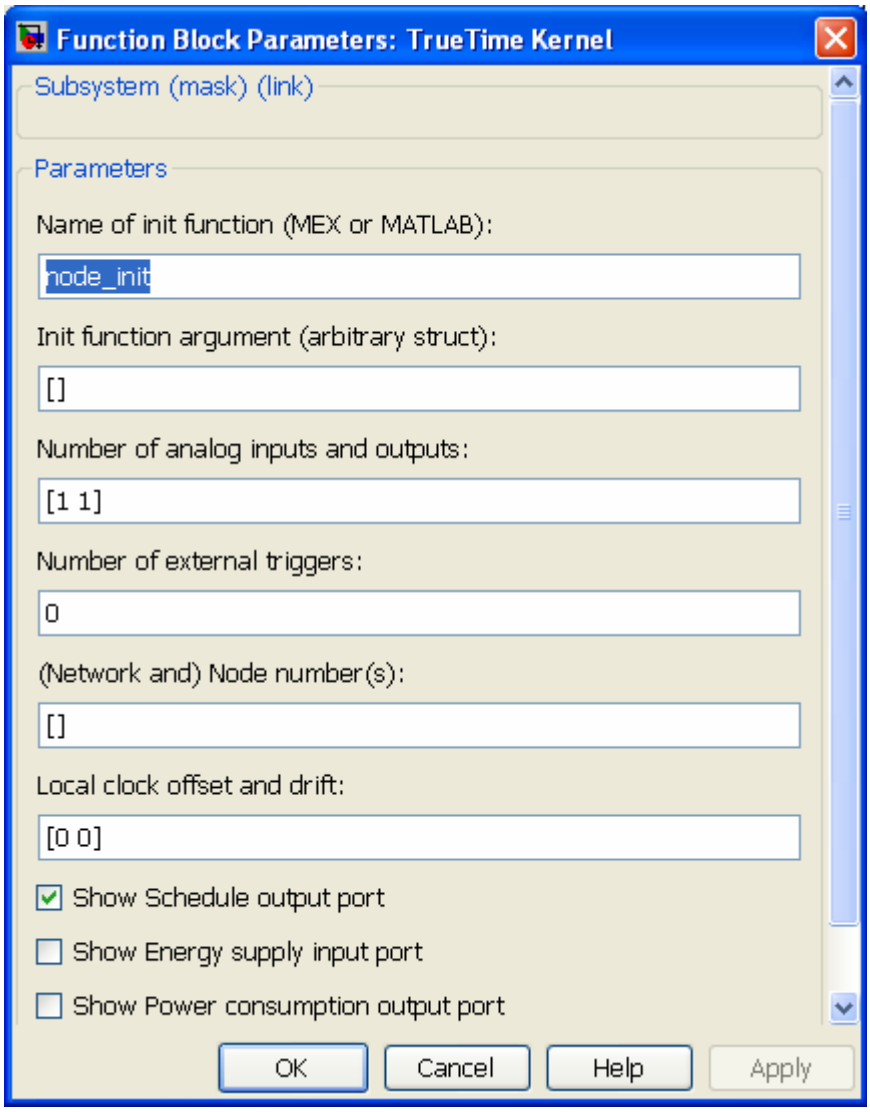
- TrueTime Kernel (Bilgisayar giriş/çıkış aygıtı veya bazı gömülü sistemler)
- TrueTime Ağı (ağ modeli)
- Denetimli bir süreç



Şekil 6.1 TrueTime Blok Şeması

6.4 TrueTime Kernel

TrueTime Kernel, giriş/çıkış, ağ veri toplama, veri işleme ve/veya hesaplamalardan sorumludur. TrueTime Kernel, sisteme bir algoritma veya lojik denetimi uygular ve tüm cihazların beyni görevi görür. Tüm bunları yapmak için, ihtiyacımıza göre modifiye ettiğimiz basit M-dosyalarını kullanır. Kernel'de aynı amaç üzerine birlikte çalışabilen, birbirinden bağımsız görevler gerçekleştirilebilir.



Şekil 6.2 TrueTime Kernel Blok parametreleri

6.4.1 Kernel Blok Parametreleri

Name of init function Başlatma komutunun adı girilir.

Init function argument Başlatma komut dosyası için opsiyonel bir değişkendir. Bu herhangi bir Matlab yapısı olabilir.

Number of analog inputs and outputs Analog giriş ve çıkışların sayısı belirlenir.

Number of externel triggers Harici tetikleyicilerin sayısı belirlenir.

(Network and) Node number(s) Düğümün numarası belirlenir.

Local clock offset and drift Clock offset nominal zamandan sabit bir zaman girişidir. Clock drift ise, eğer yerel saat, nominal saatten (gerçek simülasyon saatinden) %1 daha hızlı olması gerekirse, zaman sürüklenmesi, 0,01 olmalıdır.

6.5 TrueTime Bataryası

TrueTime batarya bloğu ekleyerek, seçilen TrueTime Kerneller için güç kaynağı ayarlanabilir. Batarya bloğunun bir parametresi vardır, o da konfigürasyon maskesi kullanarak ayarlanabilen başlangıç gücüdür. Batarya kullanmak için, kernel yapılandırma maskesi içindeki onay kutusu etkinleştirilir ve bataryanın çıkışı kernel bloğun E girişine bağlanır. Kernel bloğun, sıradan Simulink modellerin, ve kablosuz ağ bloklarının P çıkışları gibi tüm güç drain'leri, bataryanın P girişine bağlanır. Batarya basit bir integral model kullanır, böylece hem şarj hem de deşarj edilebilir.

Unutmamalıdır ki; eğer bu, kernel bloğu 0 yapmak için bataryaları ve P girişini kullanmak üzere yapılandırılmış ise, kernel hiçbir kodu çalıştıramayacaktır [1].

6.6 TrueTime Ağı

TRUETIME ağ bloğu, bir yerel alan ağ içinde orta erişim ve paket iletimini simüle eder. Ne zaman bir düğüm bir mesaj iletmeye çalışsa(ilkel ttSendMsg kullanarak), tetikleyici bir sinyal, ilgili giriş kanalı üzerinden ağ bloğuna gönderilir. Mesajın simüle edilmiş iletimi bittiğinde, ağ bloğu, çıkış kanalı ile ilgili alıcı düğümleri üzerine bir tetikleme sinyali gönderir. İletilen mesaj, alıcı bilgisayar düğümündeki bir tampona konulur. Bir mesaj, gönderici ve alıcı bilgisayar düğümü, keyfi kullanıcı verileri, (tipik olarak ölçüm sinyalleri yada kontrol sinyalleri), mesajın uzunluğu, ve bir öncelik veya bir süre gibi bağlayan opsiyonel gerçek zamana dair bilgileri içerir.

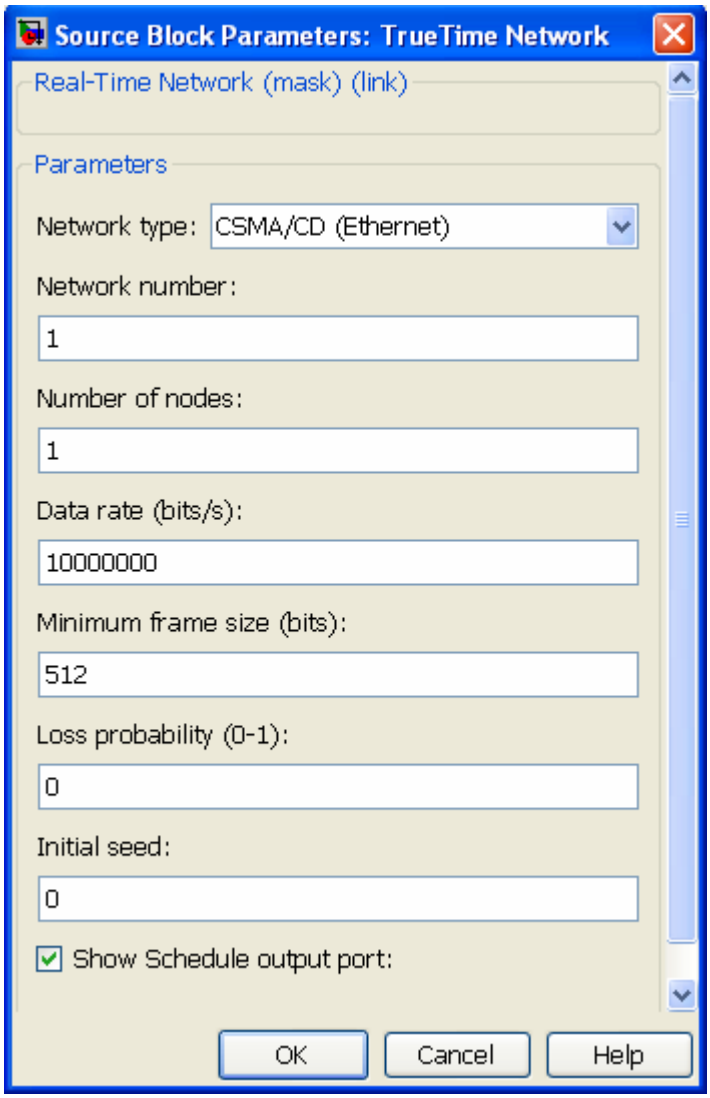
Ağın altı basit modeli desteklenir: CSMA/CD (örn.. Ethernet), CSMA/ AMP (örn.. CAN), Round Robin (örn.. Token Bus), FDMA, TDMA (örn.. TTP), ve Switched Ethernet. Yayılım gecikmesi, bir yerel alan ağında genellikle çok küçük olduğundan önemsenmez. Yalnız paket düzeyinde simülasyon desteklenir- kernel düğümlerindeki daha yüksek protokol seviyesinin, paketler vb. içinde uzun mesajlara bölündüğü varsayılır [1].

Ağ blok, blok maske diyalogu yoluyla yapılandırılmıştır, bkz. Şekil 6.3. ttSetNetworkParameter komutunu kullanarak, düğüm bazında bazı parametreleri değiştirmek mümkündür. Aşağıdaki ağ parametreleri tüm modelleri için ortaktır:

Network Number Ağ blok sayısı. Ağlar, 1 ve daha yukarı numaralandırılmalıdır. Kablolu ve kablosuz ağlar aynı numarayı kullanmaya izin vermez.

Number of nodes Ağa bağlı düğümlerin sayısı. Bu sayı, bloğun Snd, Rcv ve Schedule giriş 16 ve çıkışlarının ölçüsünü hesaplar.

Data rate (bit/s) Ağın hızı.



Şekil 6.3 TrueTimeAğı Blok Parametreleri

Minimum frame size (bits) Mesaj veya bundan daha kısa yapısı minimum boyutta vermek için tampon olur. Protokol tarafından eklenen her havai dahil minimum

yapı boyutunu gösterir. Örneğin, bir 14-byte başlık ve bir 4-byte CRC içeren minimum Ethernet yapı boyutu 512 bittir.

Loss probability (0–1) İletim esnasında bir ağ mesajının kaybolma ihtimali. Kayıp mesajlar, ağ bant genişliğini tüketir ama asla hedefe varamaz.

İnitial speed: Başlangıç hızıdır [1].

TrueTime Ağ bloğunda kullanılan ağ çeşitleri Bölüm 4.3'te işlenmiştir.

6.7 TrueTime Kablosuz Ağı

Kablosuz ağ bloğunun kullanımı ve çalışması kablolu ile aynıdır. Hesap içindeki radyo sinyallerinin yol kaybı da dikkate alınarak, düğümlerin gerçek konumlarını belirleyen x ve y girişleri vardır. İki ağ protokolü şu anda desteklenmektedir: IEEE 802.11b/g (WLAN) ve IEEE 802.15.4 (ZigBee). Kablosuz ağ hakkında daha fazla detay için [Ohlin,2006]'ya bakınız. Kullanılan radyo modeli aşağıdakiler için destek içerir:

- Amaca özel kablosuz ağları
- İzotropik anten
- Aynı zamanda mesaj gönderme ve alma yetersizliği
- $1/d^a$ olarak modellenen radyo sinyalleri yol kayıpları; d: metre cinsinden uzaklık ve a: modellemek için seçilmiş çevre parametresi.
- Diğer terminallerden parazitler

Kablosuz ağ bloğu, blok maske diyalogu yoluyla yapılandırılır, bkz. Şekil 6.4. Bazı parametreler, ttSetNetworkParameter komutuyla birlikte her düğüm baz alınarak da ayarlanabilir. Aşağıdaki parametreler tüm modeller için ortaktır:

Network type Kullanılan MAC protokolünü hesaplar. 802.11b/g (WLAN) yada 802.15.4 (ZigBee) olabilir.

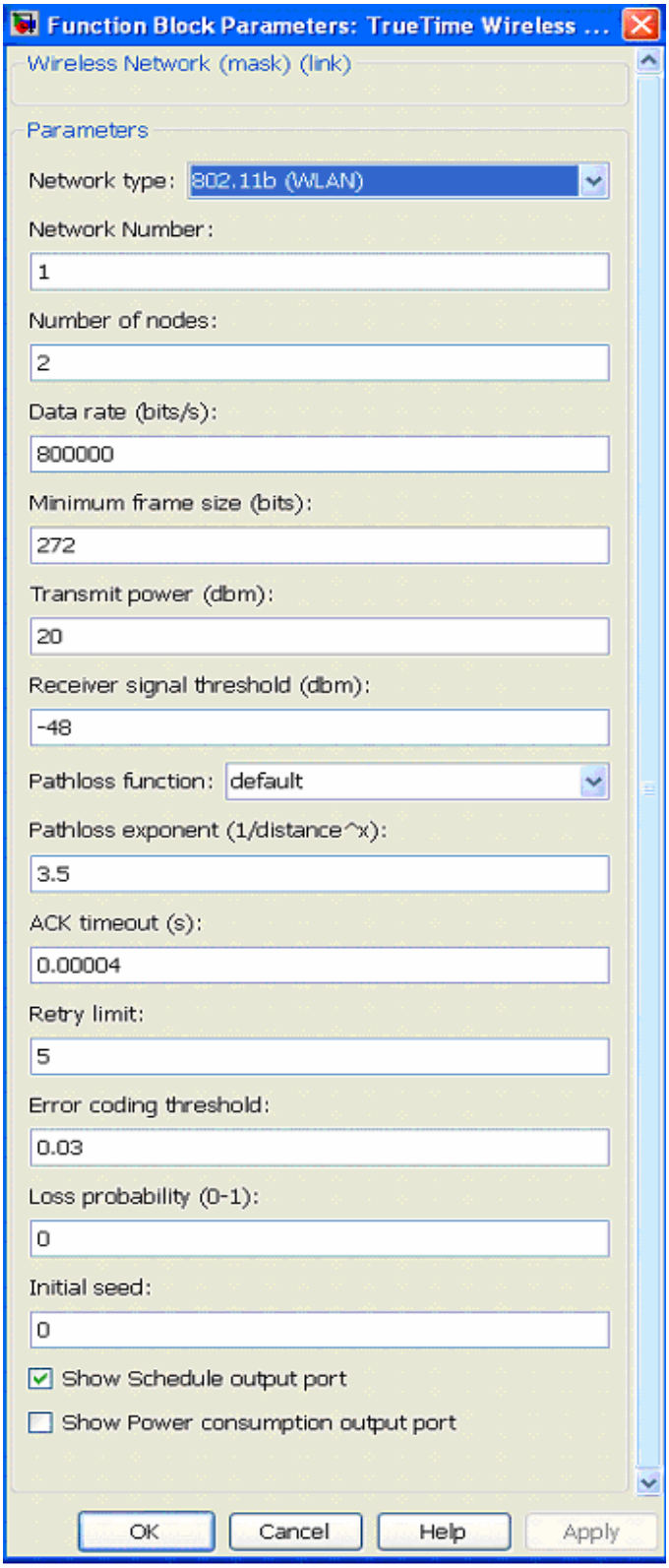
Network number Ağ bloklarının sayısıdır. Ağlar 1'den yukarı doğru numaralandırılmalıdır. Kablolu yada kablosuz ağlar aynı numarayı alamazlar.

Number of nodes Ağları birbirine bağlayan düğümlerin sayısıdır. Bu, bloğun Snd, Rcv ve Schedule giriş ve çıkışlarının boyutlarını hesaplar.

Data rate (bits/s) Ağın hızıdır.

Minimum frame size (bits) Bir mesaj veya daha kısa çerçeve, minimum uzunluğu vermek için dolgulu olacaktır. Protokol tarafından eklenen yük dahil, minimum çerçeve boyutunu gösterir. Örneğin, çoğu ağ protokolü, baştaki bitten

sondaki(kuyruk) bite kadar aynı numarayı alır, bu nedenle çerçeve en azından "baştakinin boyutu + sondakinin(kuyruk) boyutu" uzunluğunda olmalıdır.



Şekil 6.4 TrueTime Kablosuz Ağı Blok Parametreleri

Transmit power Radyo sinyalinin ne kadar güçlü olacağını ve dolayısıyla erişebileceği uzaklığı belirler.

Receiver signal threshold Alınan enerji bu eşiğin üzerinde ise, o zaman ortam meşgul olarak hesaplanır.

Pathloss function: Yol kaybı fonksiyonu belirlenmektedir.

Path-loss exponent (1/distance^x): Radyo sinyalinin yol kaybı $1/d^a$ olarak modellenir. Burada d metre cinsinden uzaklığı, a ise ortam modeline uygun şekilde seçilmiş bir parametreyi gösterir. Genellikle aralık 2-4 arası seçilmiştir.

ACK timeout Gönderilen bir düğüm, mesaj kaybolup yeniden gönderilmesi sonuçlanmadan önce ACK mesaj için beklerken olan zamandır.

Retry limit Bir düğümün, vazgeçmeden önce mesajı yeniden göndermek için geçireceği zamanın alabileceği maksimum değerdir.

Error coding threshold İşleyebilir kodlamada mesaj içindeki blok hatalarının yüzdesinin tanımlandığı, [0,1] aralığında ifade edilen bir sayıdır. Örneğin, blok hatalarının %3'ten az ise belirli kodlama şemaları tam bir mesajı yeniden oluşturabilir. Örneğin, %3'den az blok hatası olması durumunda, belirli kodlama şemaları mesajı tam olarak yeniden inşa eder. Blok hatalarının sayısı, gürültünün diğer tüm gürültülerle aynı yayın üzerinden gittiği yerde, sinyal-gürültü oranı kullanılarak hesaplanır.

Loss probability (0-1): Ağdaki paket kaybı olasılığıdır.

İnitial speed: Başlangıç hızıdır [1].

TrueTime Kablosuz Ağ bloğunda kullanılan ağ çeşitleri Bölüm 4.3'te işlenmiştir.

6.7.1 Hata Olasılıkları Hesabı

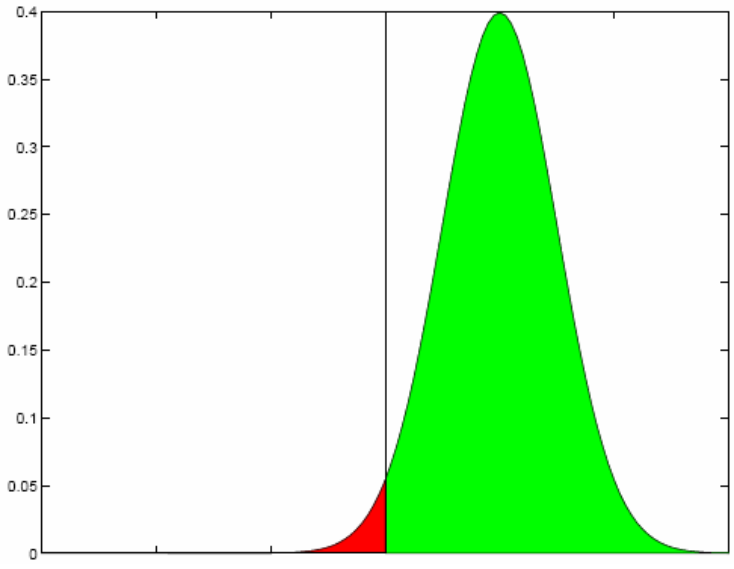
Hata olasılıkları hesaplama esnasında, BPSK1'in sürekli iletim için kullanıldığını varsaymak kolaylık içindir. Bu tabiki bir yaklaşımdır, ancak gerçeğe de yakındır.

Bir sembolün gönderildiğini varsayalım, bizim durumumuzda bu bir bit, yani 0 yada 1. Beyaz Gauss gürültüsünü eklemek, alıcı sembolü için olasılık yoğunluk fonksiyonunu verir, ki bazı sinyal-gürültü oranı Şekil 6.5'deki gibidir. Eğer alınan sembol 0 yada 1 ise, eşiğin kullanılmasına karar verilir. Karar verilen eşik, şeklin ortasında bir çizgide işaretlenir. Eşiğin solundaki daha karanlık alan sembol hatasının olasılığını verir. Gürültü oranının daha yüksek bir sinyali, hata olasılığını küçülterek eğriyi sağa iletir.

Yukarıdaki standart prosedür, mesaj içindeki her bit için ideal olarak yapılmalıdır. Hesaplanan bit hatalarının toplam sayısı, hata kodlama eşiği ile kaşlılaştırılmalıdır. Ancak bu, yapılmamaktadır, çünkü oldukça pahalı bir iştir. Bunun yerine, iletim esnasındaki maksimum gürültü seviyesi kaydedilir, ve en kötü SNR durumu hesaplanır. Bir mesaj içindeki bit hatalarının ilintisiz olduğu varsayılarak, mesaj içinde bitlerin sayısının n, gerçek bir bitin hatalı olma olasılığının p olduğu yerde, X B Bin(n,p) binom dağılımına bağlı bit hatalarının sayısı, X, çıkarılır. Eğer n değeri büyükse, merkezi limit teoremini kullanarak binom dağılımı, normal bir dağılıma yaklaşır. Bu, q=1-p olduğu yerde X B N(np,~npq) verir. Bizi esas ilgilendiren olasılıktır ki, j'nin hata kodlama eşiği olduğu yerde, jn, mesaj içinde bit hatalarının toplam sayısından büyüktür. Olasılık aşağıdaki formül kullanılarak hesaplanır:

$$P(X \le jn) = \begin{cases} \Phi\left(\dfrac{jn-np}{\sqrt{npq}}\right) & eger \quad jn-np > 0 \\ 1-\Phi\left(\dfrac{|jn-np|}{\sqrt{npq}}\right) & eger \quad jn-np \le 0 \end{cases} \qquad [6.1]$$

burada Φ standart normal kümülatif dağılım fonksiyonudur [1].



Şekil 6.5 Binari faz kaydırmalı anahtarlama ve beyaz Gauss gürültü kullanırken, bir alıcı sembol için olasılık yoğunluk fonksiyonu. Orta hat karar eşiktir. Eşiğin solundaki alan hatalı karar olasılığını verir. Sağdaki alan doğru bir karar olasılığını verir[1]

Örnek 6.1 Bir mesajın 100 bitten oluştuğunu varsayalım, yani n=100. Yukarıdaki metot kullanılarak esas bitin hatalı olma olasılığı 0,1 olarak hesaplanmıştır, yani, p=0,1 ve q=1-p=0,9. Hata kodlama eşiği %5 olarak ayarlanmıştır, yani j=0,05. O halde mesajın tamamını çözme olasılığı [1];

$$P(X \le jn) = 1 - \Phi\left(\frac{|jn - np|}{\sqrt{npq}}\right) = 1 - \Phi\left(\frac{5}{\sqrt{9}}\right) \approx 0.0478 \qquad [6.2]$$

6.7.2 Kullanıcı Tanımlı Yol-Kayıp Fonksiyonu

TrueTime kablosuz simülasyon içinde kullanılan öntanımlı yol kayıp fonksiyonu;

$$P_{alici} = \frac{1}{d^a} P_{verici} \qquad [6.3]$$

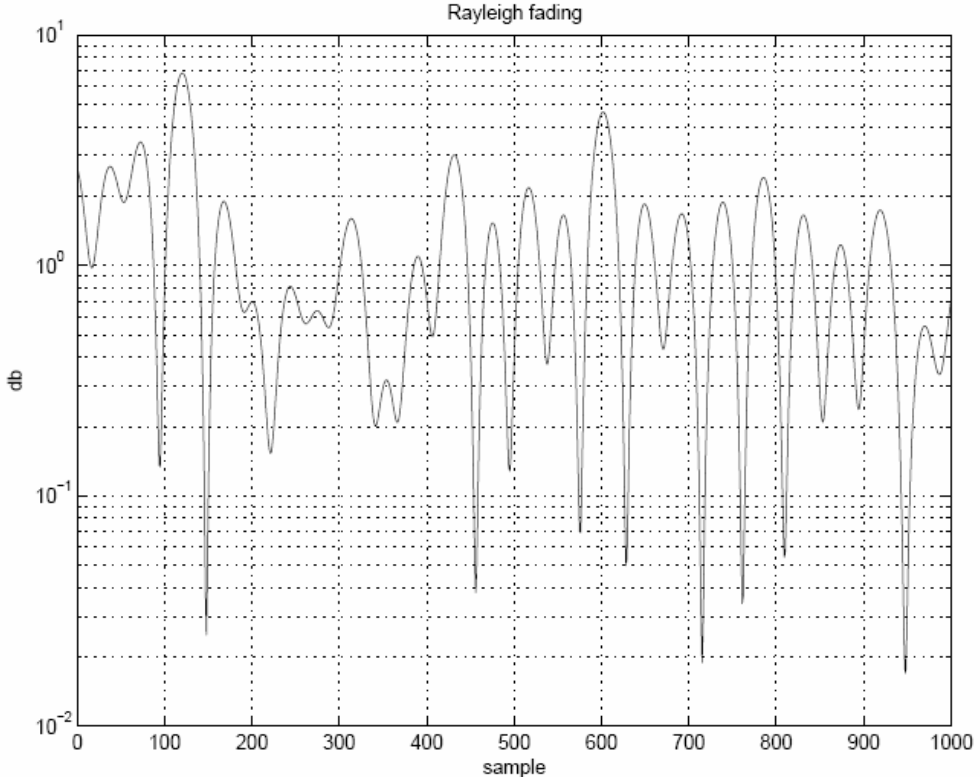
burada, P güç, d metre cinsinde mesafe ve a farklı ortamları modellemek için seçilebilir parametredir. Model genellikle simülasyonda kullanılır, ancak bazı durumlarda diğer modellerde kullanmak da avantajlı olabilir. TrueTime kullanıcı tanımlı bir yol-kayıp fonksiyonun kaydedilmesine imkan sağlar. Fonksiyon, bir Matlab fonksiyonu(M-dosyası yada MEX-fonksiyonu) gibi yazılır ve bu nedenle Matlab içinde yararlanabilir tüm tüm yerleşik fonksiyonlar avantajlarını alabilir. Özellikle, bu sürekli değişken kullanma imkanı içerir, yani, fonksiyonun çağrılar arasındaki hafızada korunan değişkenlerdir. Örneğin, fonksiyon bir Rayleigh2 solma modeli için yada belirli ortamda radyo sinyallerini engellemek için kullanılabilir. Şu anda, TrueTime çerçevesindeki düğümler yalnız x ve y koordinatlarına sahiptir ama, eğer bir yön tanıtılması söz konusu olursa, bu fonksiyon, anten davranışında model direktif etkisi için kullanılabilir.

Matlab fonksiyonu aşağıdaki argümanları alır

- İletim gücü
- Gönderici düğümünün ismi
- Gönderici düğümün x ve y koordinatları
- Alıcı düğümün ismi
- Alıcı düğümün x ve y koordinatları
- Gerçek simülasyon zamanı

Bir Rayleigh sönümlemesinin nasıl uygulandığının yapısını gösteren küçük bir örnek Liste 6.1'da görülmektedir.

Simülasyon hızını artırmak için, bir C MEX-fonksiyonu olarak Matlab fonksiyonunu uygulamak tavsiye edilir[1].



Şekil 6.6 Bir 2,4 MHz radyo frekansı kullanan bir Rayleigh sönümleme örneği. 6 km/s göreceli hızlı, 1 saniye örnekleme aralıklı ve 10 rasgele fazlı düğüm hareketleri ve alıcı düğümdeki sinyal gücünün dönüşü [1]

Rayleigh sönümleme modellemesinin yol-kayıp fonksiyonu örneği [1]

```
function power = rayleigh(transmitPower, node1, x1, y1, node2, x2, y2, time)
% Calculate the exponential pathloss
distance = sqrt((x1 - x2)^2 + (y1 - y2)^2);
power = transmitPower/(distance+1)^3.5;
% Kalman filter to get the relative velocity of the two nodes
velocity = kalman_velocity(node1, x1, y1, node2, x2, y2, time);
% Calculate the rayleigh fading
factor = calculate_rayleigh(node1, node2, velocity, time);
% Add the rayleigh fading to the exponential path loss
power = power * factor
```

6.8 TrueTime Bağımsız Ağ Blokları

Figür 1'de de gösterildiği gibi, TrueTime Gönder ve TrueTime Al diye adlandırılan bağımsız ağ blokları, kernel blokları olmadan ağ bloklarını kullanarak mesajları göndermek için kullanılabilir. Bu, kernellerin oluşturulmasını başlatmak ve kesme işleyicileri vb. yüklemek, zorunda olmadan TrueTime ağ simülasyonunu oluşturmayı mümkün kılar. Başka bir deyişle, hiçbir M-dosyaları olmadan Simulik'te bütün bir ağ simülasyonu oluşturmak mümkündür. Ayrıca bu blokları, kernel bloklarıyla karıştırmak da mümkündür, böylece diğerleri, bir kernel blok içinde kod fonksiyonu yürütmesindeki standart ttSend ve ttGetMsg temellerini kullanırken bazı istasyonlar bağımsız ağ bloklarını kullanırlar.

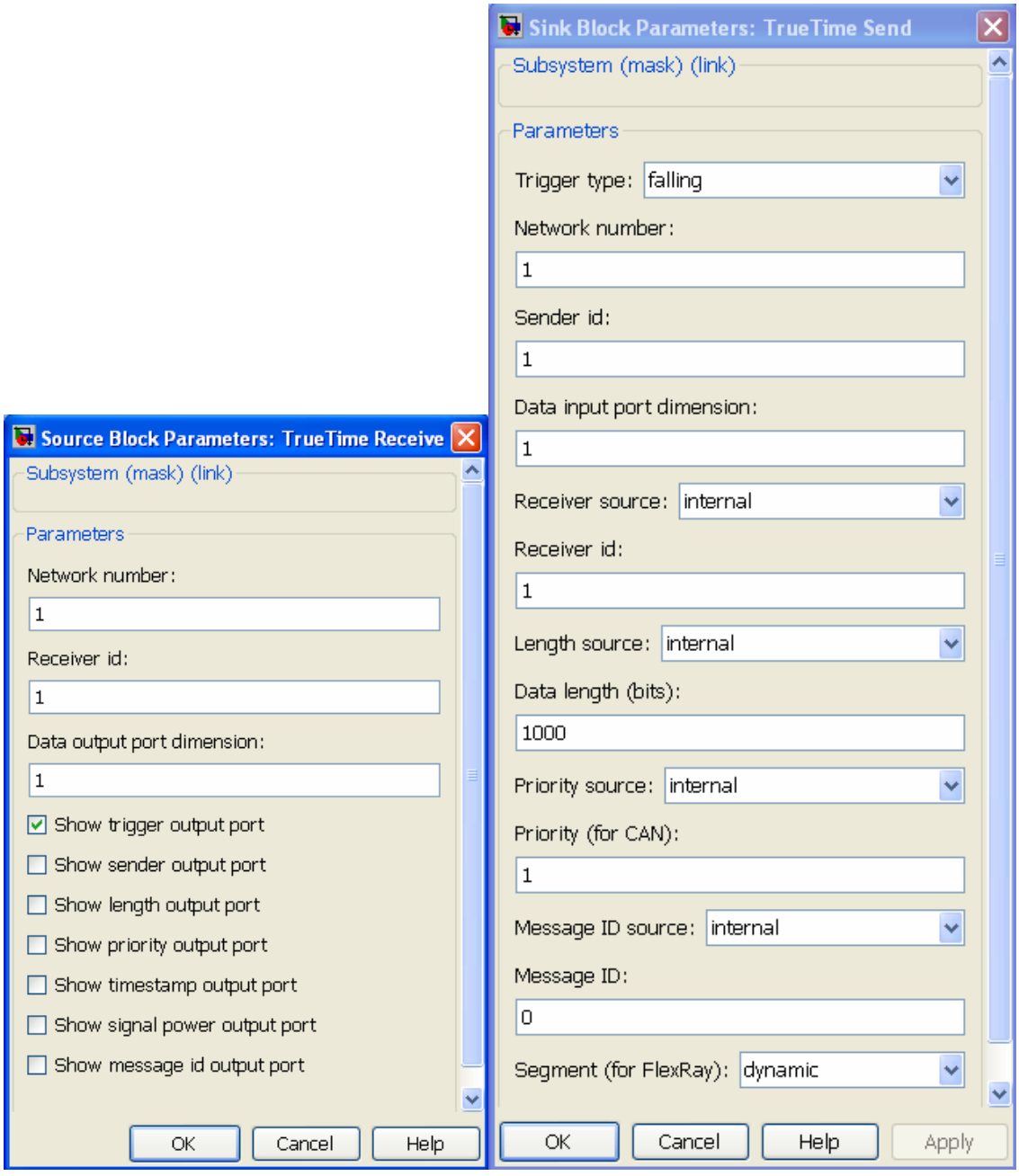
Şekil 6.7'de görüldüğü gibi, bağımsız ağ blokları blok maske diyalogları yoluyla yapılandırılır. Parametreler, temel "True-Time Gönder" ve "True-Time Al"'ın kullanımındakiler gibidir. TrueTime Gönder blok, yükselterek, düşürerek yada herhangi yakın değerlerde tetiklenerek yapılandırılabilen Simulink tetikleyici giriş portuna sahiptir. ttGetMsg bloğu opsiyonel tetikleme portuna sahiptir. Alınan mesajlar gibi, tetikleyici çıkışın değeri 0 ila 1 arası anahtarlanır. Bu port, yeni bir mesaj alındıktan sonra yürütülmesi gereken bir şeyin tetiklemesinde kullanılabilir [1].

Örnekler

$DIR/examples dizini on örnek içerir:

simple	Basit bir periyodik kontrol uygulaması için birkaç farklı metot gösterilmektedir. Hem M-file hem de C++ uygulamaları destekler.
threeservos	Aynı CPU üzerinde 3 periyodik kontrol görevinin yürütüldüğü yerde görev planlamasını ve kontrolünü gösterir. Hem M-file hem de C++ uygulamaları destekler.
networked	Dört bilgisayar düğümler ve kablolu ağ blok içeren ağ kontrolünü gösterir. Tek başına ağ arabirim bloklarını kullanan bir uygulama gibi hem M-file hem de C++ uygulamaları destekler.
wireless	Enerji tüketimini en aza indirmek için iletim gücü kontrolü ile kablosuz ağ kontrolünü gösterir. Yalnız M-file destekler.
AODV	AODV yönlendirme gibi yüksek-katman iletişim protokollerinin, TRUETIME'da nasıl uygulandığını gösterir. Yalnız M-file destekler.
soccer	10 mobil robotun futbol oynamasının simülasyon ve animasyonunu göstermede büyük bir örnektir. Yalnız M-file destekler.

RUNES_demo Ultrason trilaterasyon kullanarak bir sensör ağ üzerinden mobil robot navigasyonu ile ilgili büyük bir örnektir.



Şekil 6.7 Solda TrueTime gönderme bloğunun iletişim kutusu, ve sağda TrueTime alma bloğunun iletişim kutusu

İlk üç örnek hem Matlab hem de C++ sürümlerinde yer almaktadır. Ancak aşağıdaki açıklamalarda sadece Matlab durumu ele alınacaktır. Örneklerin nasıl derleneceği hususunda ayrıntılı talimatlar için, ilgili örnek dizinlerde README-dosyalarına bakılabilir.

TrueTime'ın kurulumu ve çalıştırılması EK-1'de verilmiştir.

TrueTime komutları EK-2'de verilmiştir.

6.9 Bir DC-servo'nun PID Kontrolü

6.9.1 Giriş

İlk örnek DC-servo prosesinin basit PID kontrolünü içerir, ve TRUTIME simülasyon ortamına dair temel bir giriş verir. Proses, bir TRUETIME kernel blok içinde uygulanan bir denetleyici görevi tarafından kontrol edilir. Denetleyici görevinin dört farklı uygulaması, uygulama periyodik aktivitesi için farklı yollar göstermek için verilmiştir. Dosyalar $DIR/examples/simple/matlab dizininde bulunur[1].

6.9.2 Süreç ve Denetleyici

DC-servo aşağıdaki sürekli-zaman transfer fonksiyonuyla açıklanır [1]:

$$G(s) = \frac{1000}{s(s+1)} \qquad [6.4]$$

PID-denetleyici aşağıdaki eşitliklere göre uygulanır

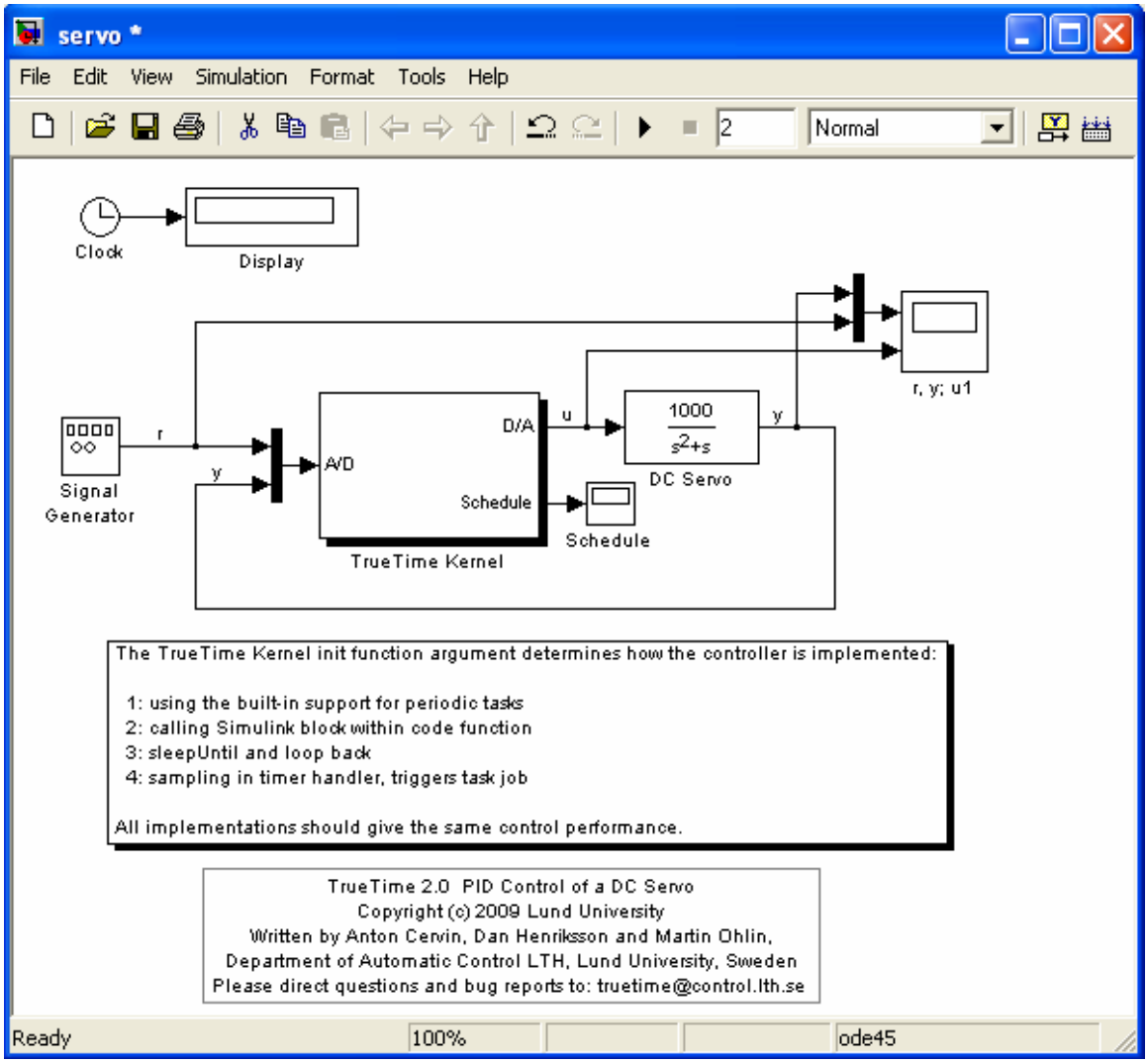
$$\begin{aligned} P(k) &= K.(\beta r(k) - y(k)) \\ I(k+1) &= I(k) + \frac{Kh}{T_i}(r(k) - y(k)) \\ D(k) &= a_d D(k-1) + b_d (y(k-1) - y(k)) \\ u(k) &= P(k) + I(k) + D(k) \end{aligned} \qquad [6.5]$$

Burada $a_d = \frac{T_d}{Nh + T_d}$ ve $b_d = \frac{NKT_d}{Nh + T_d}$, [16]. Denetleyici parametreleri, kapalı-devre sistemi verebilmek için bant genişliği ω_c = 20 rad/s ve göreli sönümleme ζ = 0.7 olarak verilmiştir[1].

6.9.3 Simülasyon Dosyaları

Liste 6.2 içinde başlatma komut dosyasının (servo_init.m) kısaltılmış bir versiyonu verilmiştir. Başlatma komut dosyasın içinde görüleceği üzere, periyodik kontrol

görevinin dört farklı uygulaması arasında bir seçim yapmak mümkündür. Bunlar, kernel blok iletişimi içinde başlatma fonksiyon parametresi tarafından belirlenir.

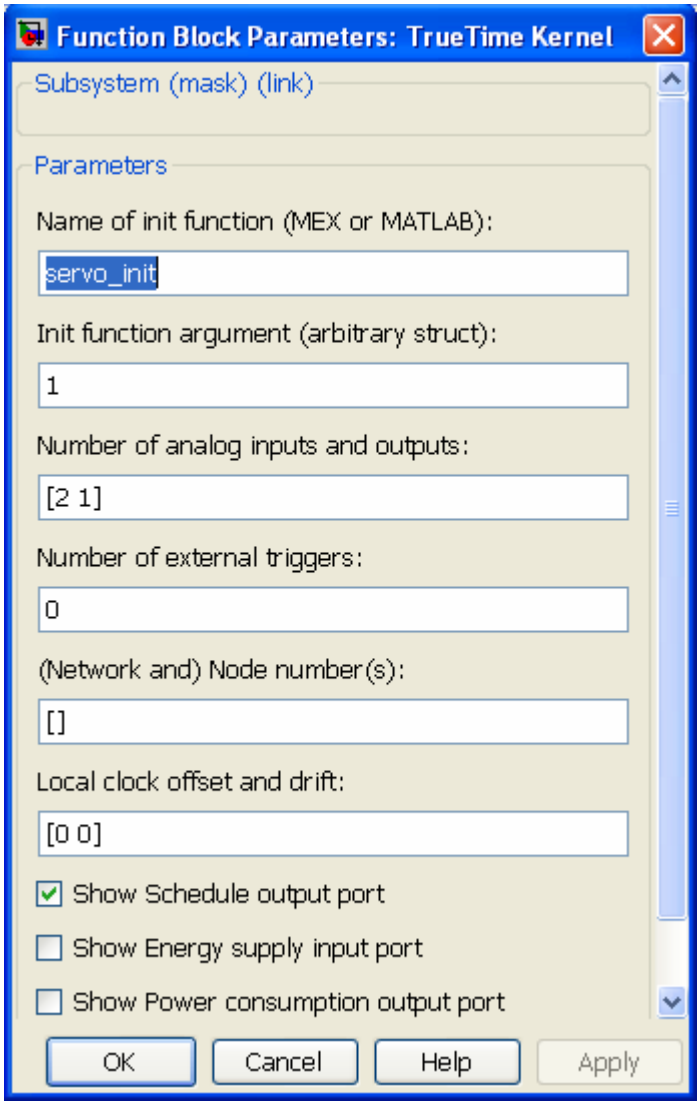


Şekil 6.8 DC Servo ağ modeli

- *Uygulama1:* Periyodik görevler için TRUETIME yerleşik destek kullanır, ve kod fonksiyonu pidcode1.m dosyasında verilmiştir.
- *Uygulama 2:* Periyodik görevler için TRUETIME yerleşik destek de kullanır, ancak her örnekte kontrol sinyalinin hesaplanması bir Simulink blok şeması çağrılarak yapılır. Kod fonksiyonu pidcode2.m dosyasında verilmiştir. Tüm denetleyici parametreleri ve durumları Simulink blok içerdiğinden görev verisi (veri2) yalnızca u kontrol sinyalini içerir.
- *Uygulama 3:* TRUETIME ilkel ttSleepUntil kullanarak periyodik görevi uygular. Kod fonksiyonu pidcode3.m dosyasında verilmiştir.

- *Uygulama 4:* Periyodik zamanlayıcı kullanarak, periyodik görevi uygular. İlişkili kesme işleyicisi, prosesi örnekler ve görev işlerini tetikler. İşleyici ve denetleyici görevi posta kutusu kullanarak iletişim kurar. İşleyici ve denetleyici için kod fonksiyonları, sırasıyla code.m ve pidcode4.m dosyalarında verilmiştir[1].

PID-kontrol örneği için başlatma komut dosyası [1] EK-3'de verilmiştir

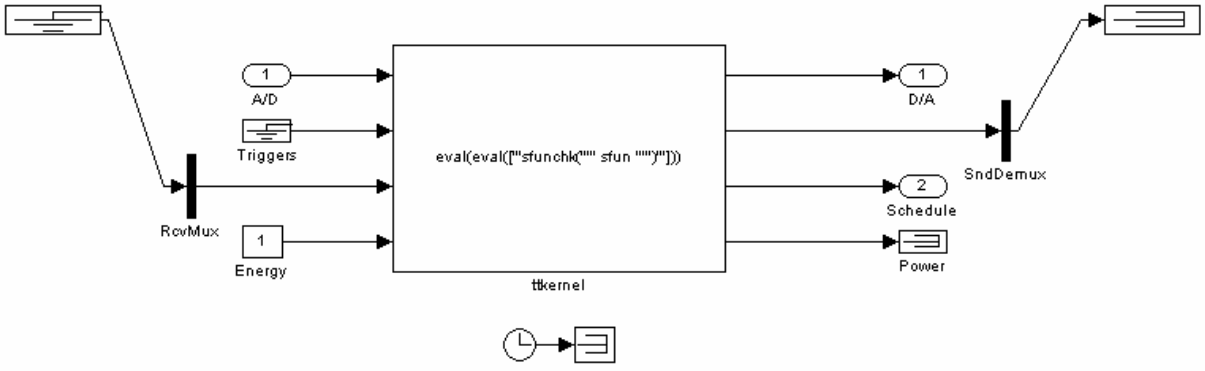


Şekil 6.9 DC Servo'nun PID kontrolü TrueTime Kernel'e birim fonksiyon yükleme

6.9.4 Çözümler

Simulink model *servo.mdl* olarak anılır ve Şekil 6.8'de verilmiştir. Simulink modeli açılır ve aşağıdakiler sırayla denenir.

- Bir simülasyon çalıştırılır ve denetleyicinin beklendiği gibi davrandığı doğrulanır. Kontrol sinyalinde 2 ms'lik hesaplama gecikmesine dikkat edilir. Pidcode1.m kod fonksiyonuyla karşılaştırılır. Program çizimi çalıştırılır (high=çalışıyor, medium=hazır, low=meşgul).
- Farklı giriş-çıkış gecikmelerinin etkilerini simüle etmek için, kod fonksiyonunun birinci segmentinin yürütme zamanını değiştirme denenir.
- Örnekleme periyodunu değiştirilir ve sonuçlanan kontrol performansı çalışılır.
- *controller.mdl* Simulink bloğunda bir PID-denetleyici uygulanmaktadır. Kernel blokun başlatma fonksiyon parametresi 1'den 2'ye değiştirilir, böylece 1 yerine uygulama 2 kullanılacaktır. İlgili *pidcode2.m* kod fonksiyonu çalışılır. Tüm örneklerde, bu kod fonksiyonu, kontrol sinyalini hesaplamak için Simulink blok kullanılır.
- Uygulama 3 değiştirilir ve simülasyon çalıştırılır. Pidcode3.m kod fonksiyonu çalışılır.
- Uygulama 4 değiştirilir ve bir simülasyon çalıştırılır. samplercode.m ve pidcode.m kod fonksiyonları çalışılır. Program çizimi içindeki işleyici kapsamı dahil edilir [1].



Şekil 6.10 DC Servo PID kontrolü TrueTime Kernel ağ bloğu alt sistemi

6.10 Ters Sarkaçın TrueTime'da PID Kontrolü

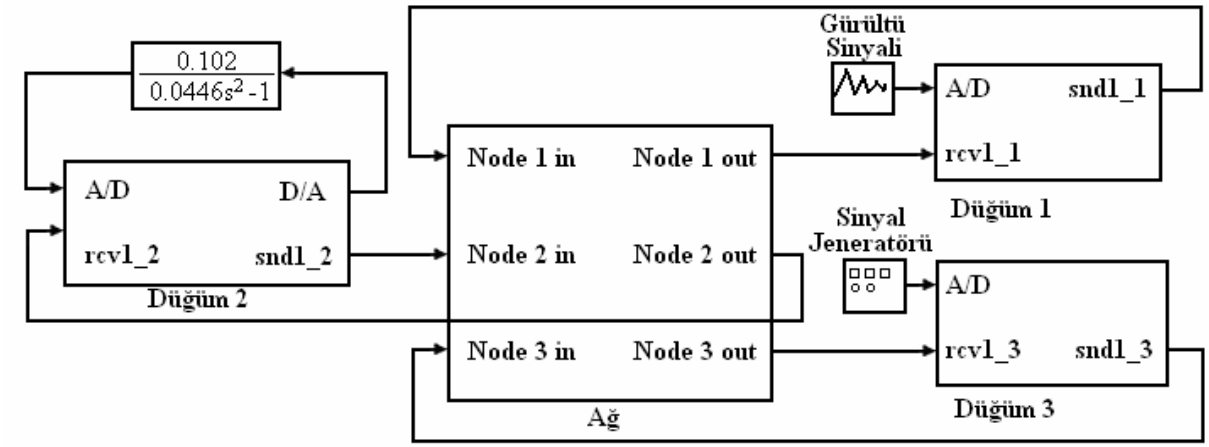
Tezin bu kısmında Bölüm 3.2'de verilmiş olan örneğin TrueTime Toolbox'ı üzerinde simülasyonu yapılarak, farklı ağların tesis üzerindeki etkileri tartışılacaktır.

Örnek 6.1

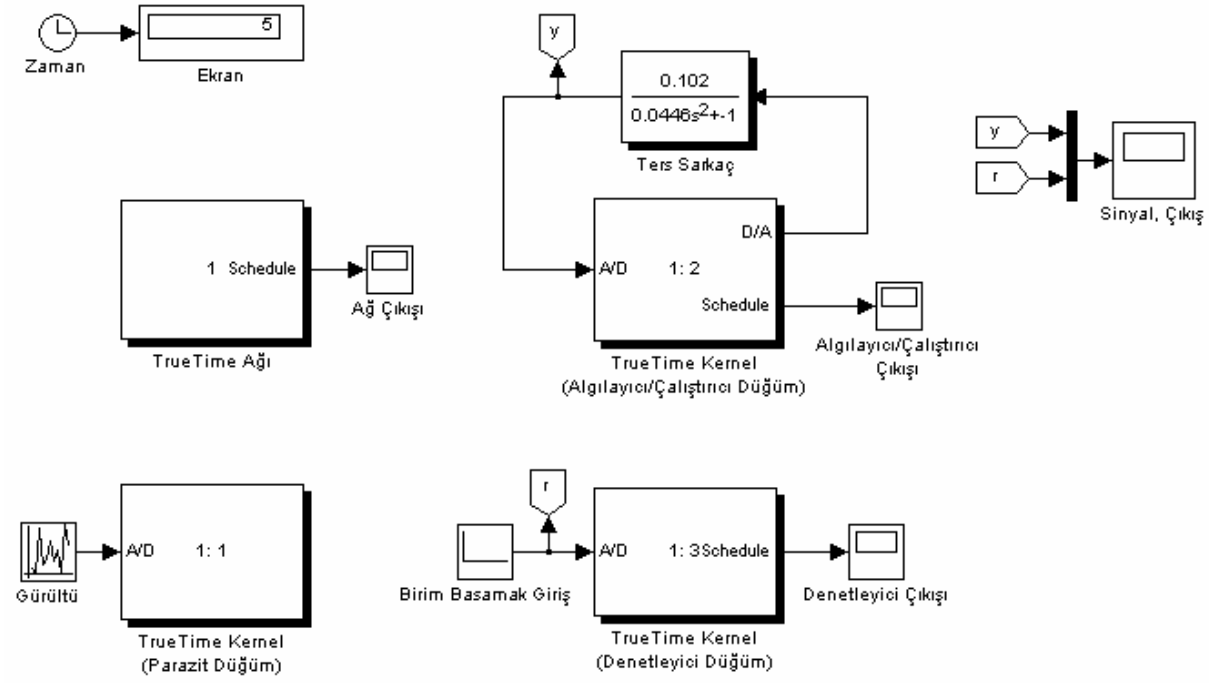
Bu örnekteki simülasyonda aynı ağ üzerinde çalışan 3 cihazın bağlanabildiği bir ağ kontrol sistemi tasarlanmıştır. Bunun için TrueTime ağ parametreleri girişinde düğüm

sayısı (Number of nodes) 3 olarak girilir. (Düğüm sayısı artırılarak daha fazla cihaz bağlamaya olanak sağlanabilir.) Örneğimizde Düğüm1, Düğüm2 ve Düğüm3 kernelleri kullanılmıştır. Burada;

- Düğüm1 Parazit Düğüm
- Düğüm2 Sensör(Algılayıcı)_Çalıştırıcı Düğüm
- Düğüm3 Denetleyici Düğüm



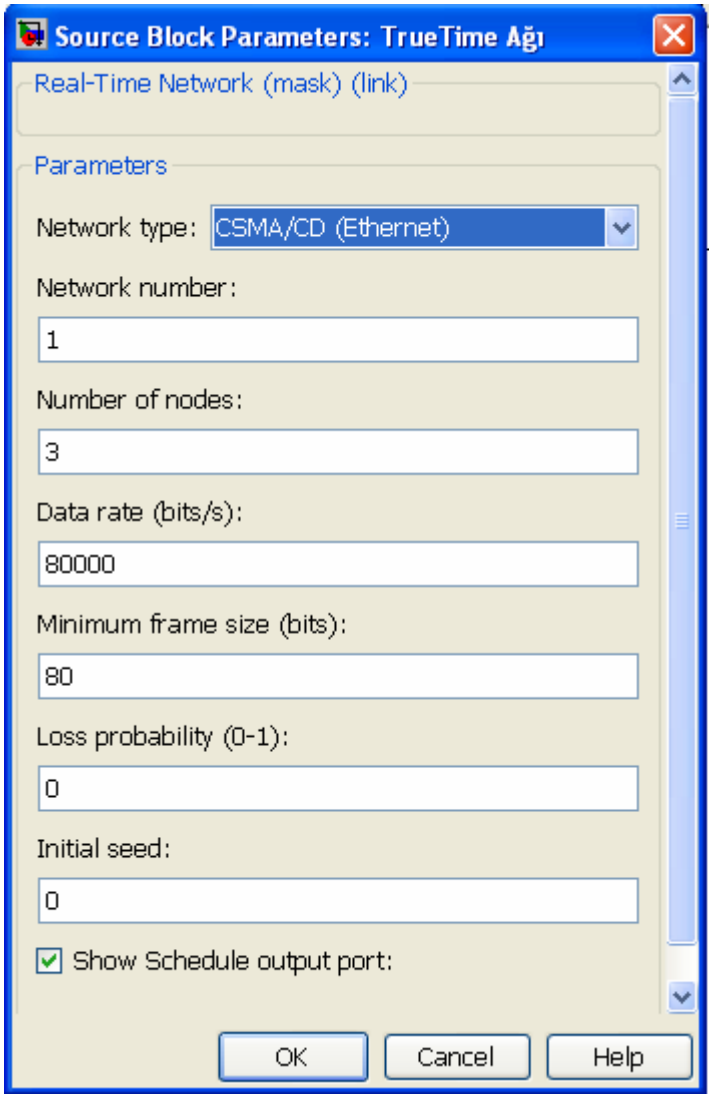
Şekil 6.11 Ters Sarkaç Kontrol Sisteminin TrueTime Toolbox'da ağ üzerinden simülasyonunun yapılmasına dair bir örnek



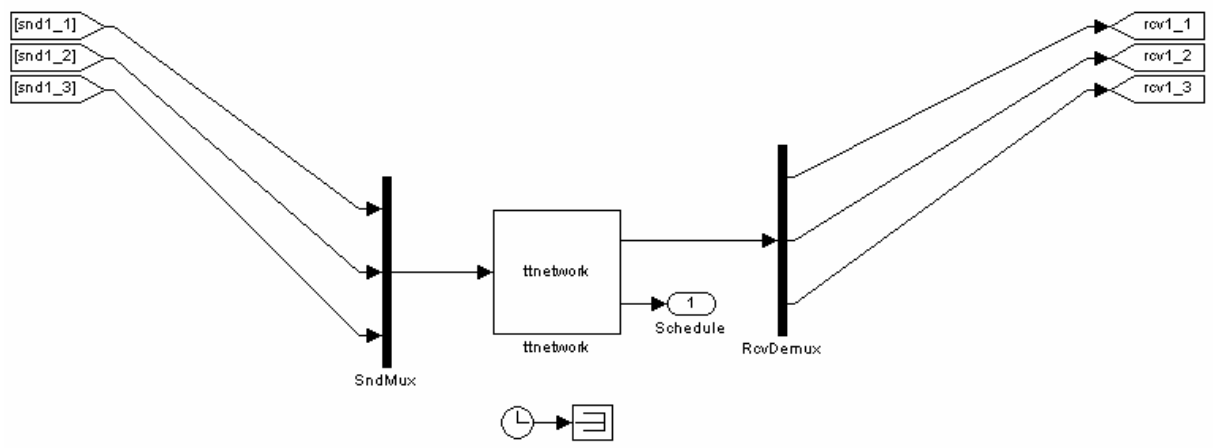
Şekil 6.12 Örneğin Matlab TrueTime üzerinde modellenmesi

6.10.1 Ağ

Şekil 6.13'de ağ blok parametreleri kutusundan görüldüğü gibi ağ, 3 düğüm bağlanacak şekilde tasarlanmıştır. Bağlanacak düğüm sayısı artırılmak istendiğinde, "Number of nodes" sayısı artırılır. Kullanmak istediğimiz ağ türü "Network Type" opsiyonundan seçilebilir. Yine, veri hızı blok parametrelerindeki data rate(bit/s) sayısı değiştirilerek, ağ hızının değişmesi sağlanabilir. Şekil 6.14'de ağ bloğunun alt sistem detayı verilmiştir. Burada yine 3 adet düğümün bağlantı şeması görülmektedir.

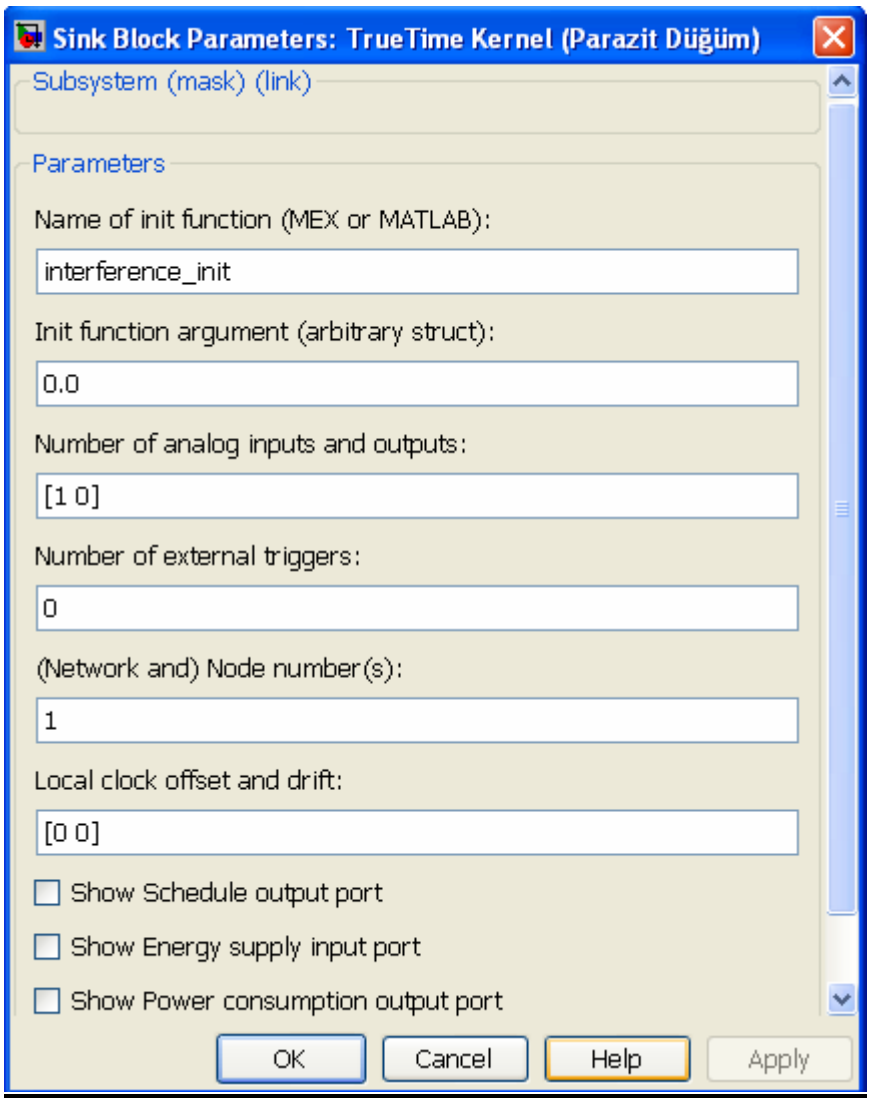


Şekil 6.13 Ağ blok parametreleri



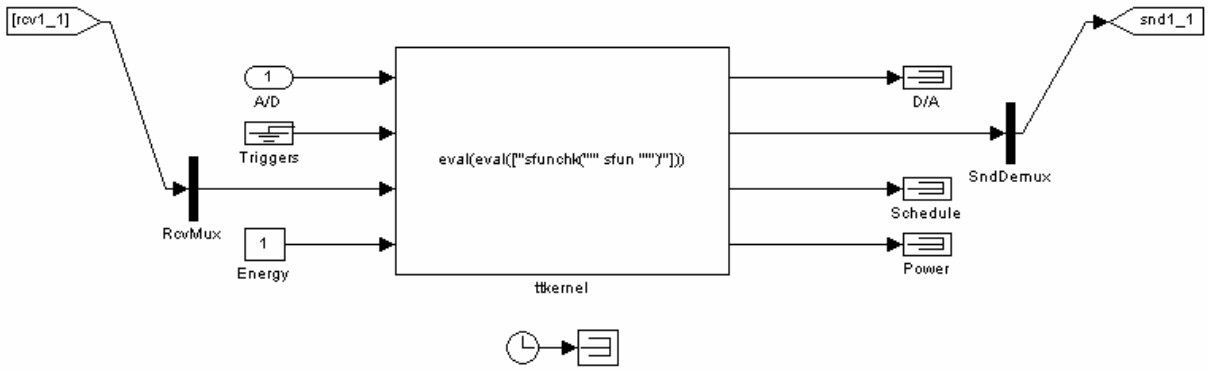
Şekil 6.14 TrueTime ağ bloğu alt sistemin detayı

6.10.2 Düğüm 1

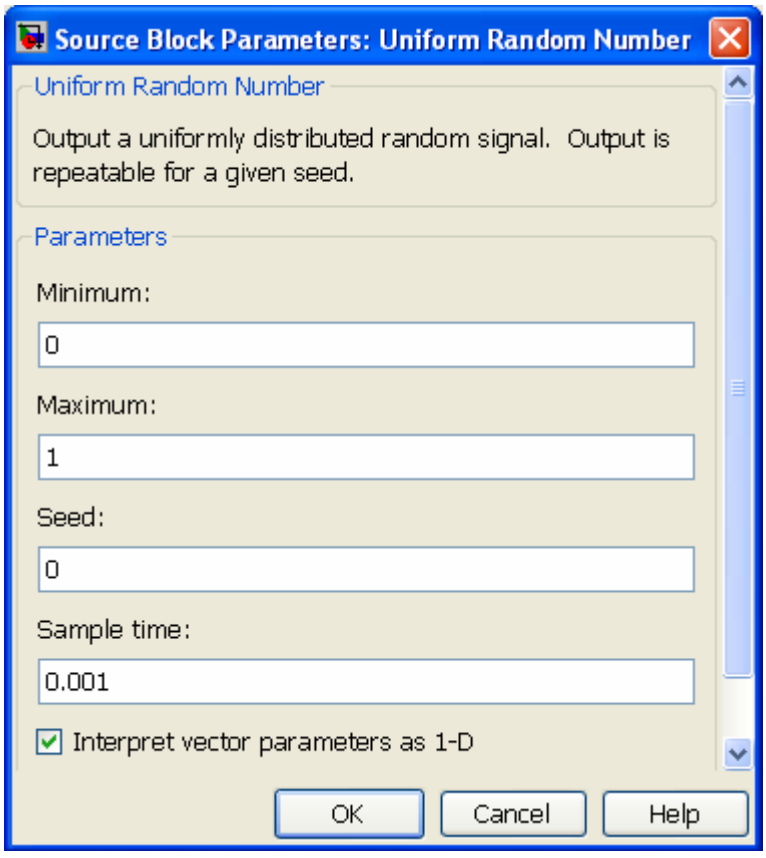


Şekil 6.15 Kernel'e birim fonksiyon yükleyerek Düğüm 1 haline getirme

Şekil 6.15'da Kernel blok parametrelerinin ayarlanarak Kernel'in Parazit düğüm haline getirilmesi verilmiştir. "Name of init function" kısmına parazit düğümün yazılımın bulunduğu interference_init.m dosyasının ismi yazılarak Kernel'e gerekli görev atanmıştır. Kernel'e Şekil 6.17'de blok parametreleri verilmiş olan gürültü sinyalini bağlayabilmek için, "Number of analog input" 1 seçilerek Kernel'de analog bir giriş oluşturulmuştur. Bu girişte A/D dönüştürücü sayesinde analog alınan sinyal ağa dijital olarak gönderilmektedir. Kernel'in alt sistemi Şekil 6.16'da görülmektedir.



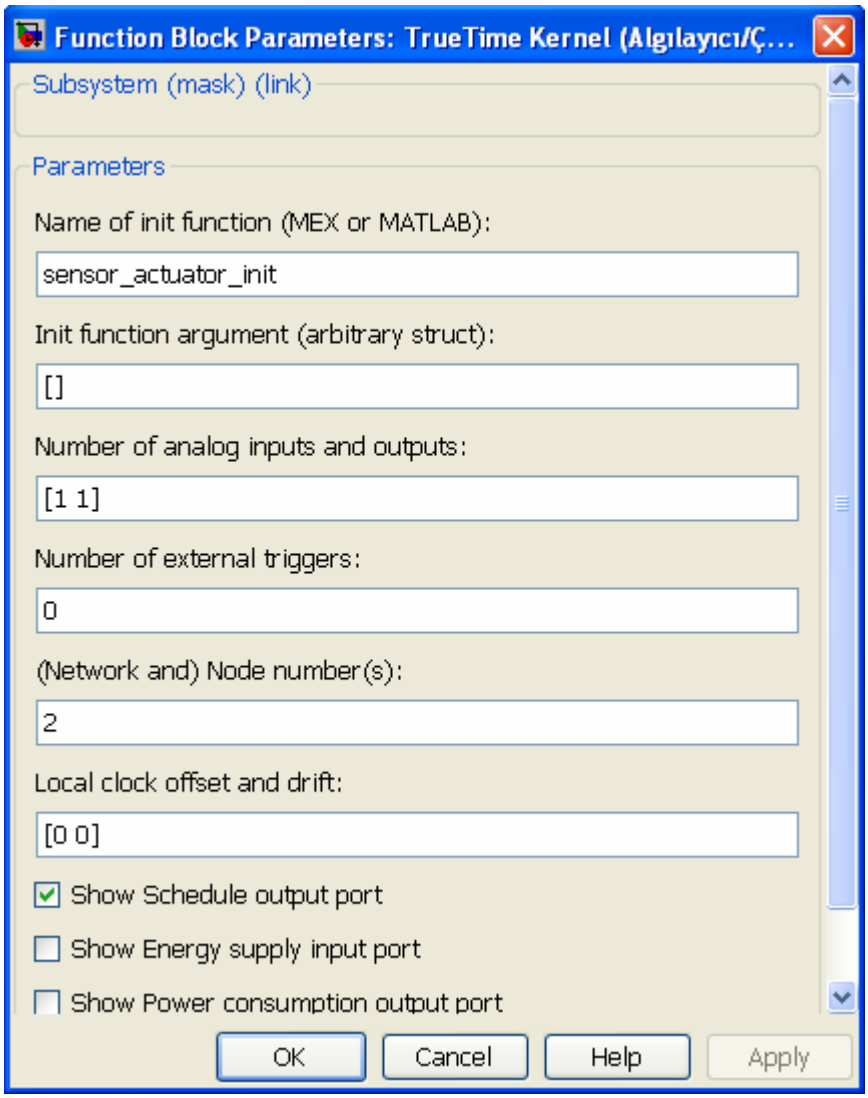
Şekil 6.16 Düğüm 1 alt sistemi



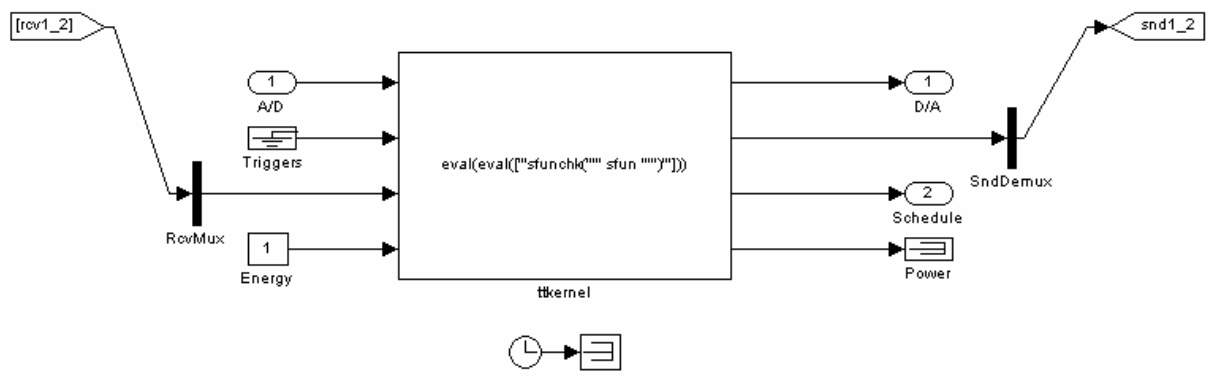
Şekil 6.17 Sisteme eklenen gürültünün blok parametreleri

6.10.3 Düğüm 2

Şekil 6.18'da Kernel blok parametrelerinin ayarlanarak Kernel'in Sensör(Algılayıcı) Çalıştırıcı düğüm haline getirilmesi verilmiştir. "Name of init function" kısmına sensör çalıştırıcı düğümün yazılımın bulunduğu sensor_actuator_init dosyasının ismi yazılarak Kernel'e gerekli görev atanmıştır. Kernel'e, ters sarkaç sistemi bağlayabilmek için "Number of analog inputs and outputs" [1 1] seçilerek Kernel'de analog bir giriş ve çıkış oluşturulmuştur. Burada dikkat edilmesi gerek konu Kernel içinde aynı zamanda A/D, D/A dönüşümlerinin yapılıyor olmasıdır. Ağdan dijital olarak alınan sinyal, analog sinyale dönüştürülerek Ters Sarkaç Sistemine iletilir. Ters Sarkaç Sisteminden alınan analog sinyal yeniden dijital sinyale dönüştürülerek ağa iletilmiş olur. Kernel'in alt sistemi Şekil 6.19'de görülmektedir.

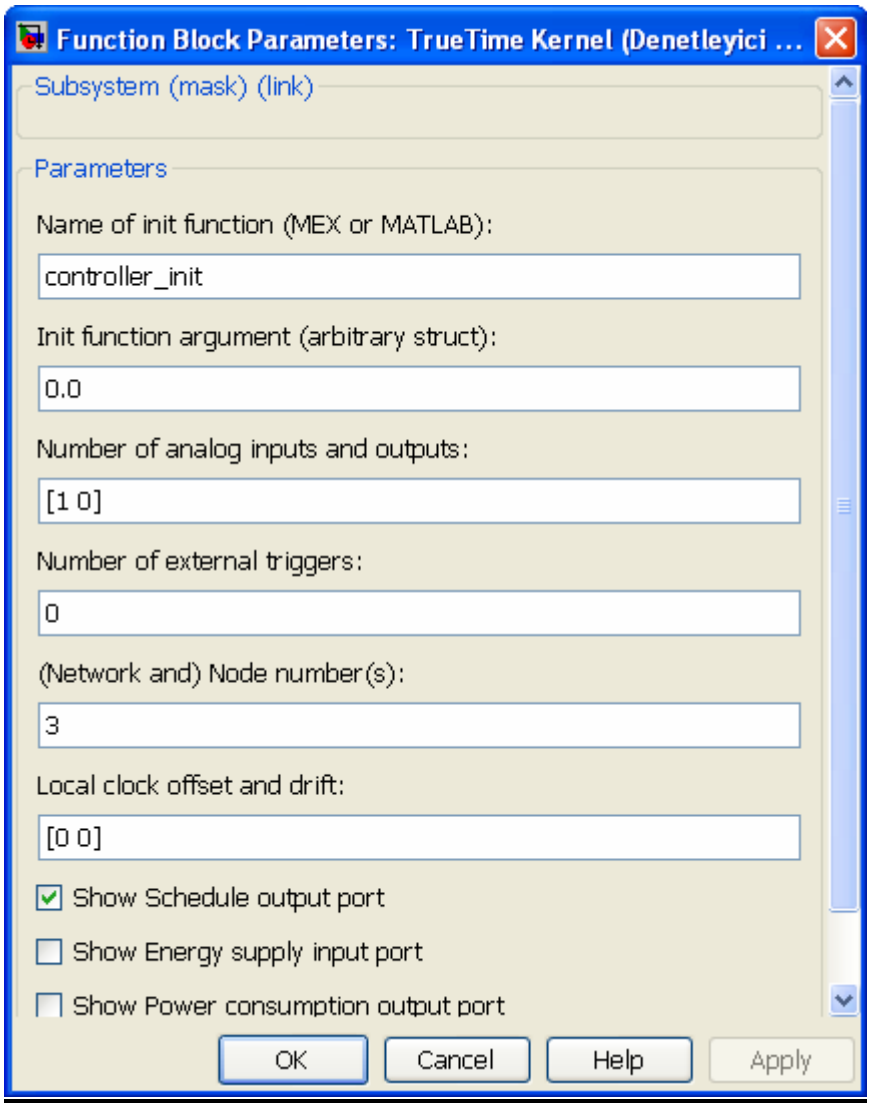


Şekil 6.18 Kernel'e birim fonksiyon yükleyerek Düğüm 2 haline getirme



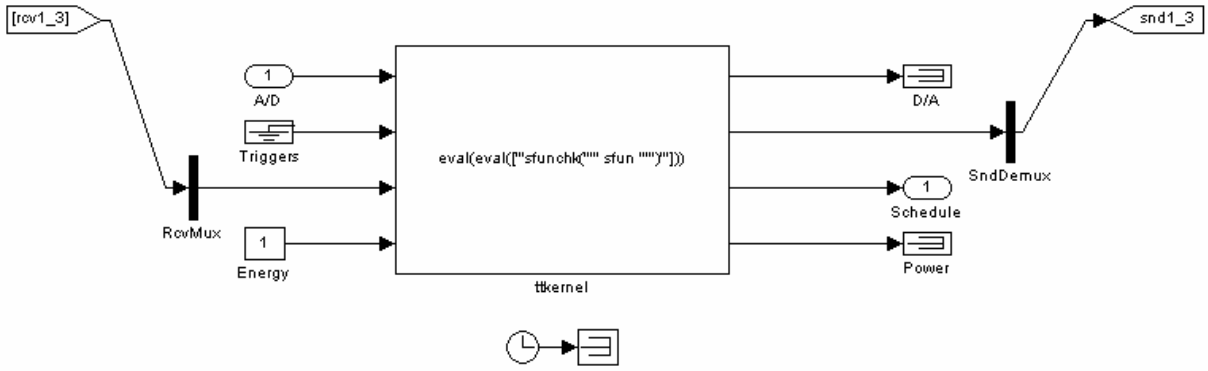
Şekil 6.19 Düğüm2 alt sistemi

6.10.4 Düğüm 3



Şekil 6.20 Kernel'e birim fonksiyon yükleyerek Düğüm 3 haline getirme

Şekil 6.20'te Kernel blok parametrelerinin ayarlanarak Kernel'in Denetleyici düğüm haline getirilmesi verilmiştir. "Name of init function" kısmına denetleyici düğümün yazılımın bulunduğu controller_init dosyasının ismi yazılarak Kernel'e gerekli görev atanmıştır. Kernel'e birim basamak sinyalini bağlayabilmek için, "Number of analog input" 1 seçilerek Kernel'de analog bir giriş oluşturulmuştur. Bu girişte A/D dönüştürücü sayesinde analog alınan sinyal ağa dijital olarak gönderilmektedir. Kernel'in alt sistemi Şekil 6.21'de görülmektedir.



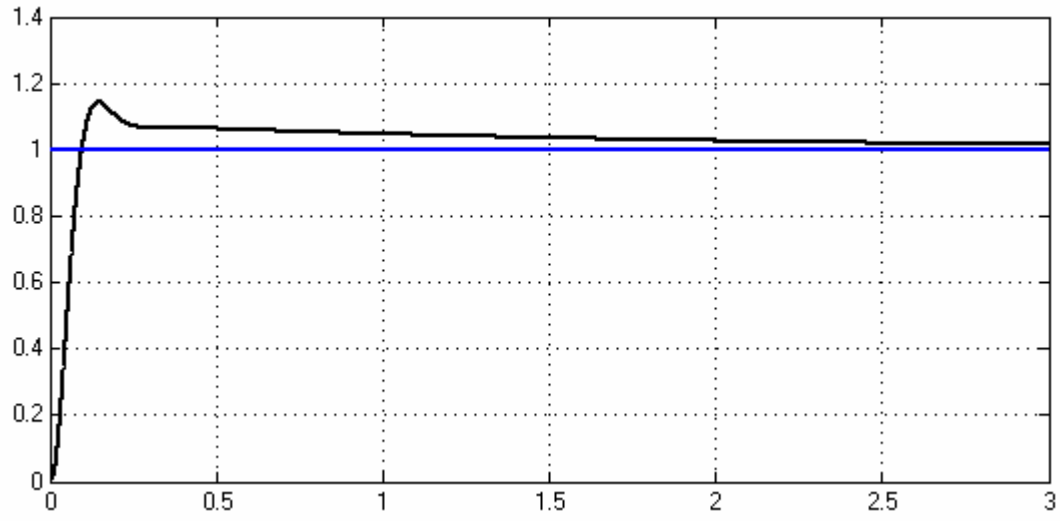
Şekil 6.21 Düğüm3 alt sistemi

Ters Sarkaç TrueTime Toolbox PID Kontrolü Komut Dosyaları EK-4'te verilmiştir.

6.10.5 ÇIKIŞ

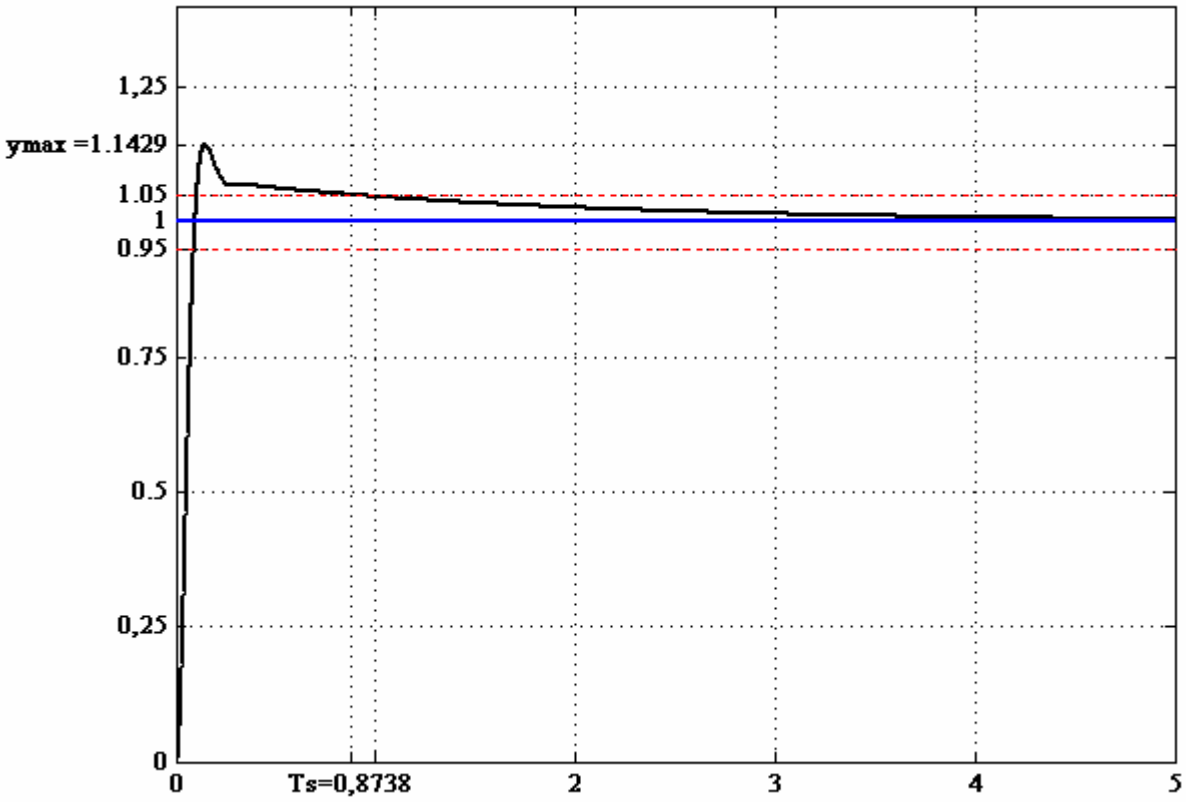
Bu bölümde, farklı ağlarda yapılan simülasyon sonuçları sunulmuştur.

Şekil 6.22'de Ters Sarkacın PID kontrol modelinin CSMA/CD(Ethernet) ağı için Matlab TrueTime çıkışı verilmiştir.



Şekil 6.22 Ters Sarkacın PID kontrol modelinin CSMA/CD(Ethernet) ağı için Matlab TrueTime çıkışı

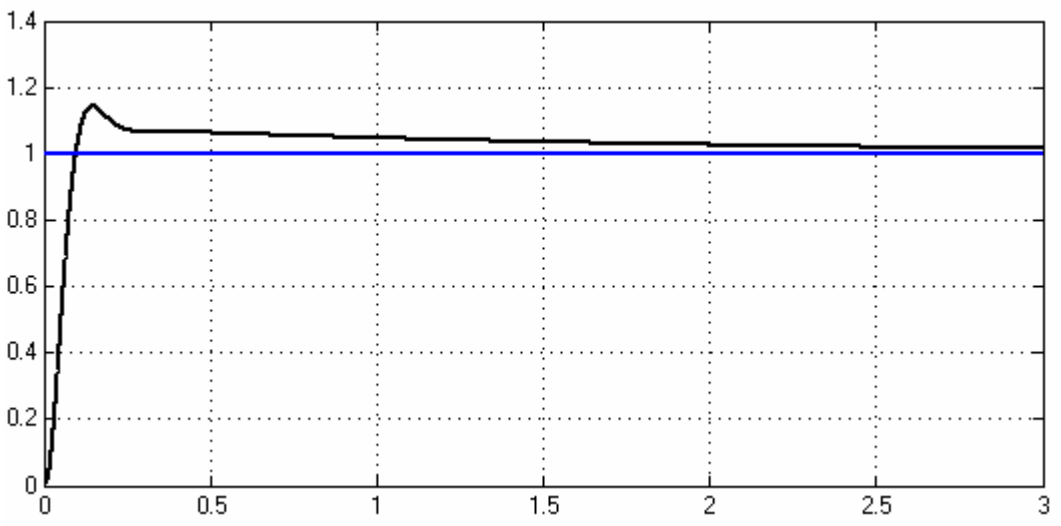
TrueTime'da kullanılan CSMA/CD(Ethernet) ağı çıkışının geçit sürecinin kalite parametreleri Şekil 6.23'ten görülmektedir. Şekilden görüldüğü üzere, T_s=0.8738, y_{max}=1.1429 ve y_{kd}=1'dir. Buradan en büyük aşım değerinin yüzdesi, %14.29 olarak bulunur.



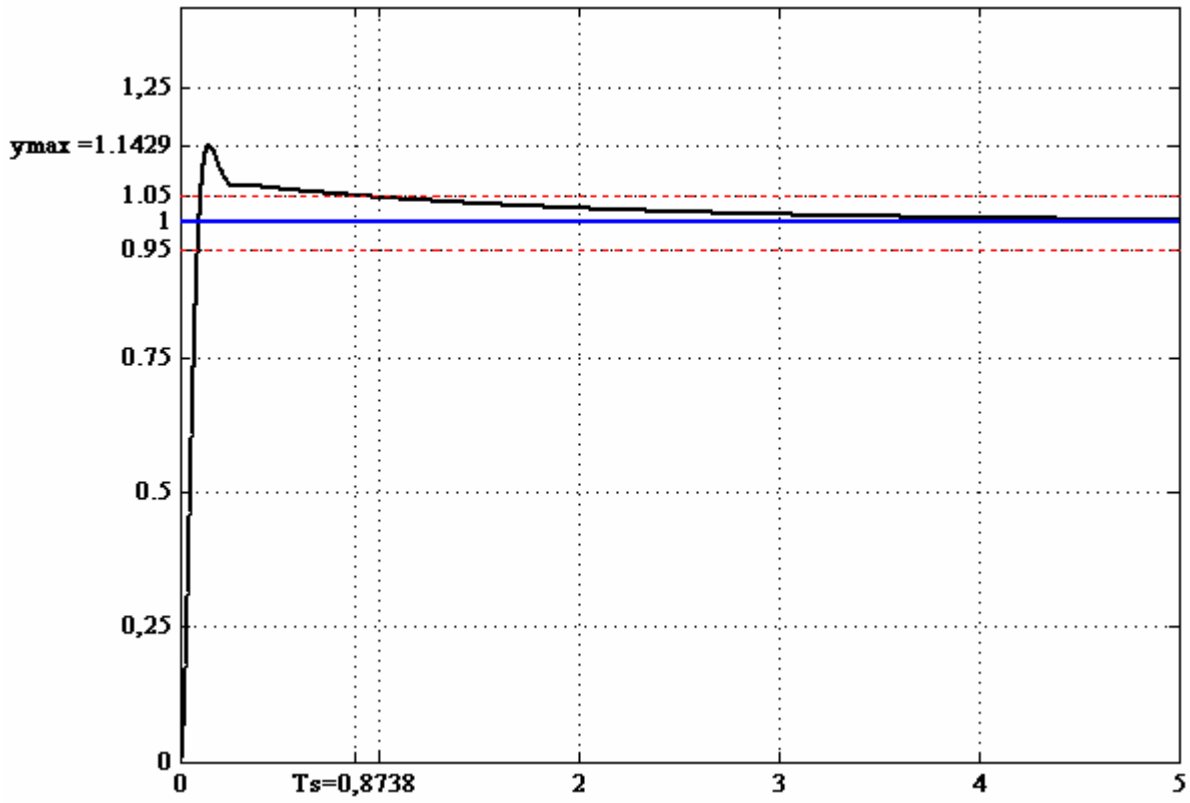
Şekil 6.23 Ters Sarkacın PID kontrol modelinin CSMA/CD(Ethernet) ağı için TrueTime çıkışında, sürecin en büyük değeri (y_{max}) ve yerleşme zamanı (T_s)

Şekil 6.24'de Ters Sarkacın PID kontrol modelinin CSMA/AMP(CAN) ağı için Matlab TrueTime çıkışı verilmiştir.

TrueTime'da kullanılan CSMA/AMP(CAN) ağında geçit sürecinin kalite parametreleri Şekil 6.25'den görülmektedir. Şekilden görüldüğü üzere, T_s=0.8738, y_{max}=1.1429 ve y_{kd}=1'dir. Buradan en büyük aşım değerinin yüzdesi, %14.29 bulunur.

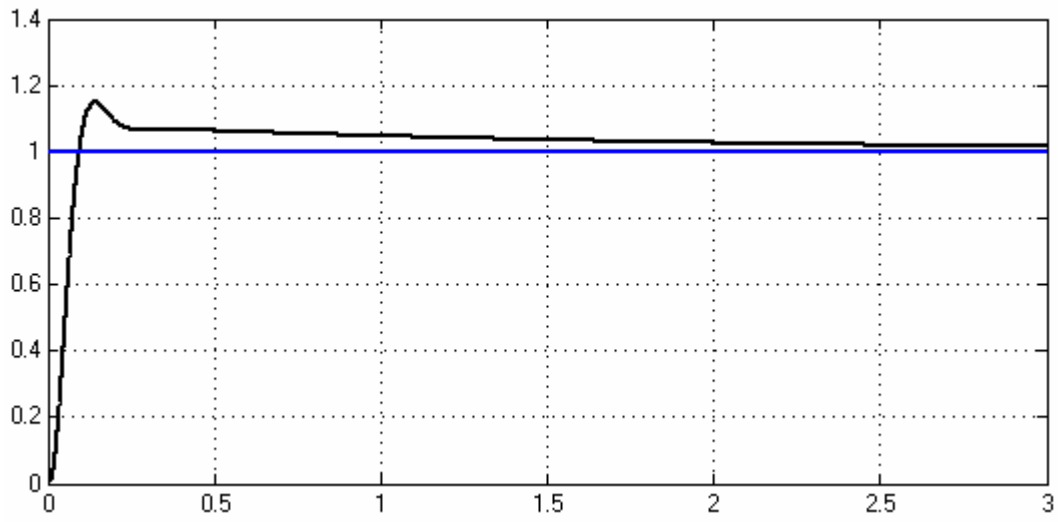


Şekil 6.24 Ters Sarkacın PID kontrol modelinin CSMA/AMP(CAN) ağı için Matlab TrueTime çıkışı

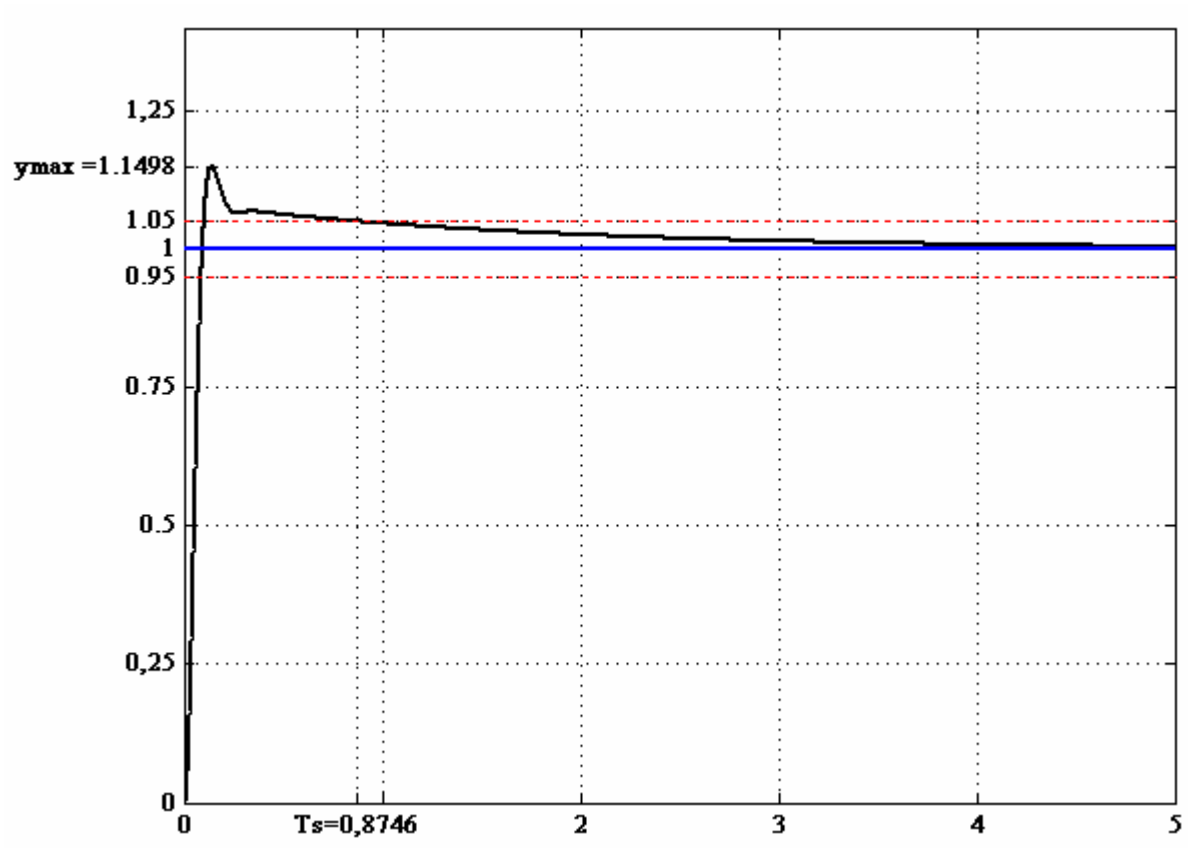


Şekil 6.25 Ters Sarkacın PID kontrol modelinin CSMA/AMP(CAN) ağı için TrueTime çıkışında, sürecin en büyük değeri (y_{max}) ve yerleşme zamanı (T_s)

Şekil 6.26'da Ters Sarkacın PID kontrol modelinin Round Robin ağı için Matlab TrueTime çıkışı verilmiştir.



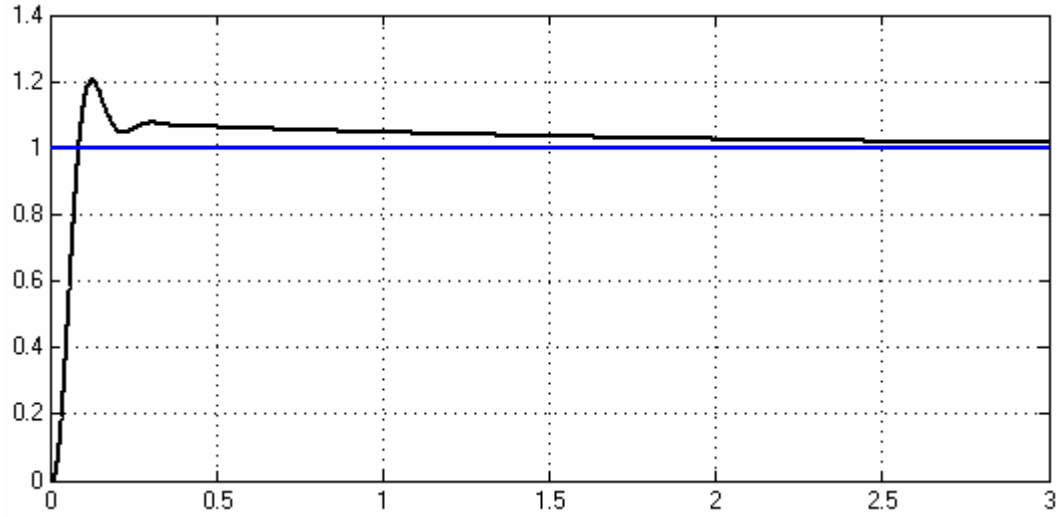
Şekil 6.26 Ters Sarkacın PID kontrol modelinin Round Robin ağı için Matlab TrueTime çıkışı



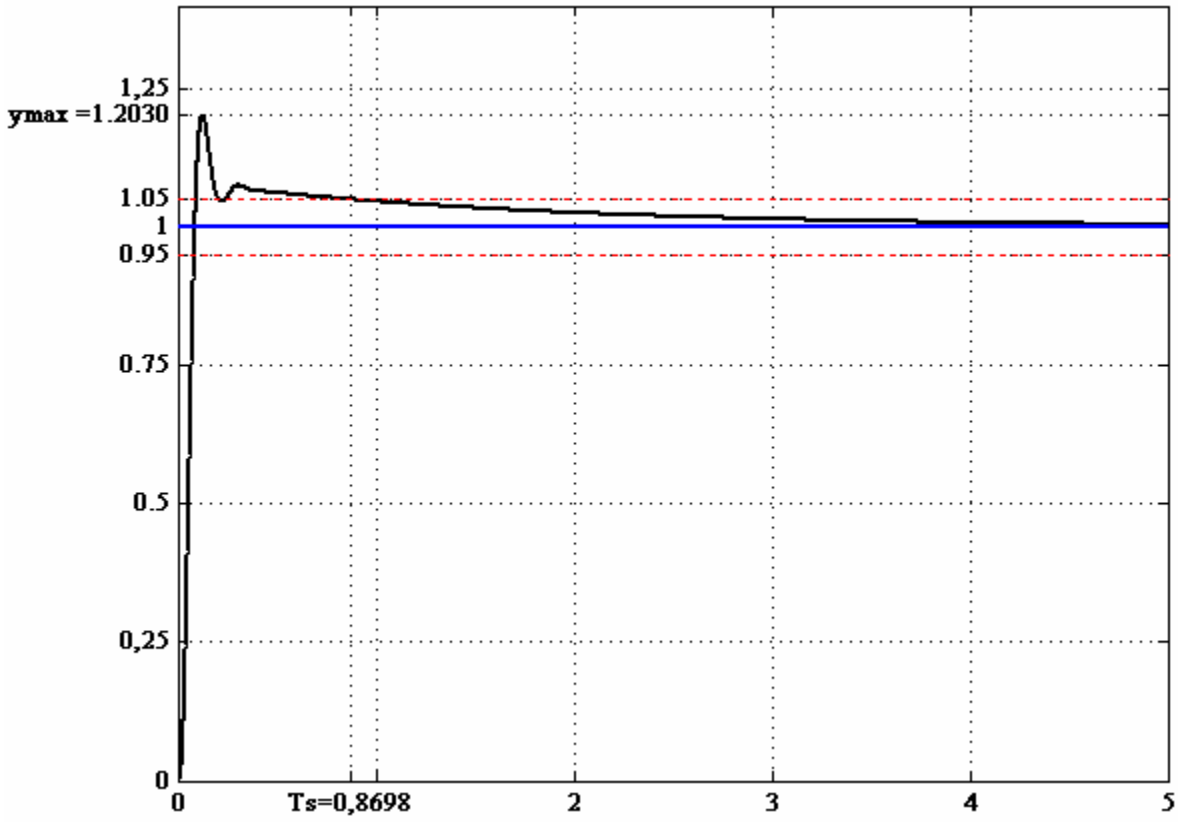
Şekil 6.27 Ters Sarkacın PID kontrol modelinin Round Robin ağı için TrueTime çıkışında, sürecin en büyük değeri (y_{max}) ve yerleşme zamanı (T_s)

TrueTime'da kullanılan Round Robin ağında geçit sürecinin kalite parametreleri Şekil 6.27'den görülmektedir. Şekilden görüldüğü üzere, T_s=0.8746, y_{max}=1.1498 ve y_{kd}=1'dir. Buradan en büyük aşım değerinin yüzdesi, 14.98 bulunur.

Şekil 6.28'de Ters Sarkacın PID kontrol modelinin FDMA ağı için Matlab TrueTime çıkışı verilmiştir.



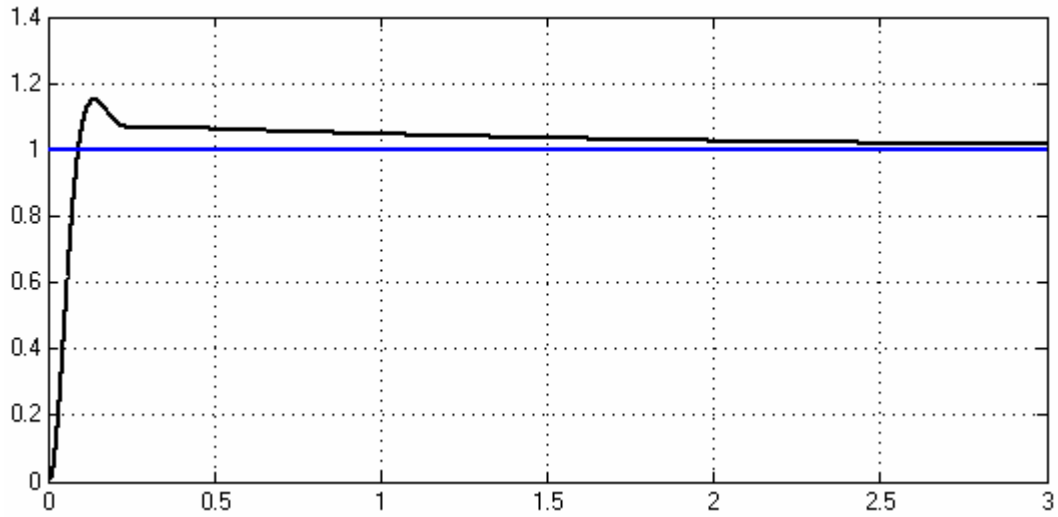
Şekil 6.28 Ters Sarkacın PID kontrol modelinin FDMA ağı için Matlab TrueTime çıkışı



Şekil 6.29 Ters Sarkacın PID kontrol modelinin FDMA ağı için TrueTime çıkışında, sürecin en büyük değeri (y_{max}) ve yerleşme zamanı (T_s)

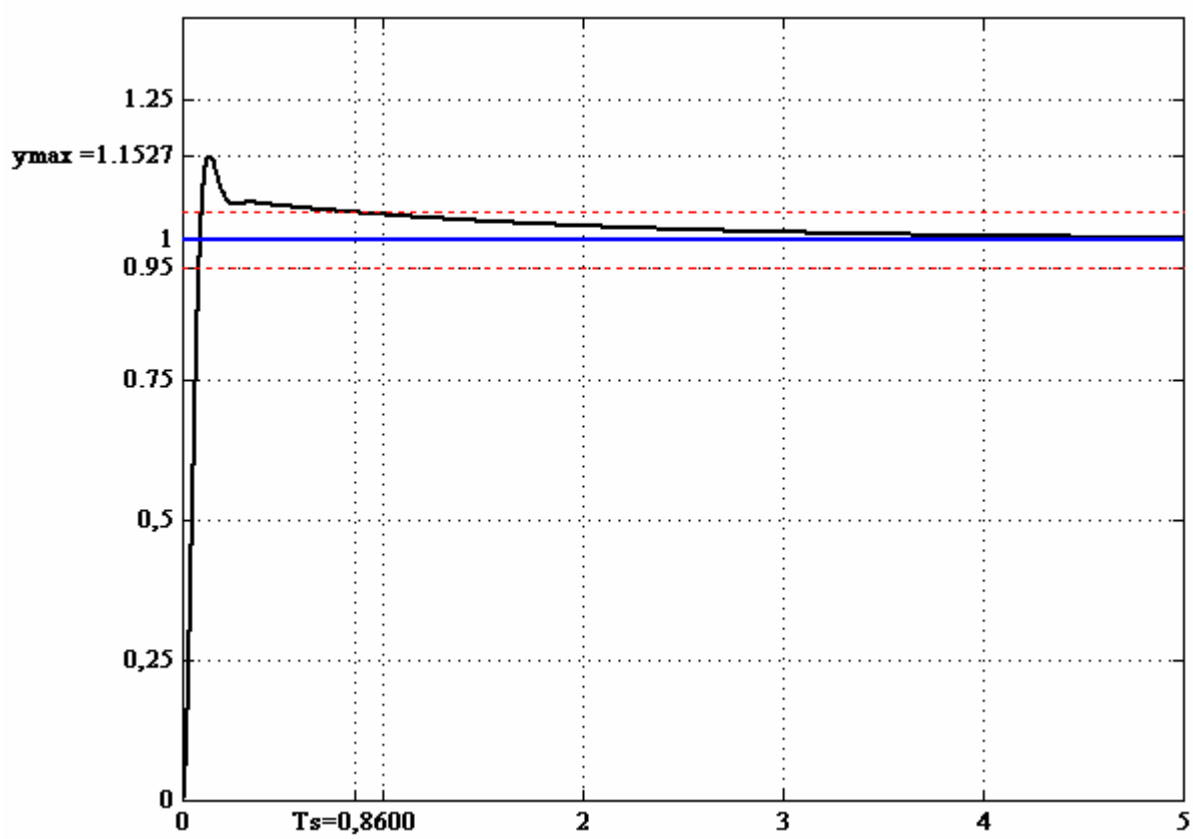
TrueTime'da kullanılan FDMA ağında geçit sürecinin kalite parametreleri Şekil 6.29'dan görülmektedir. Şekilden görüldüğü üzere, T_s=0.8698, y_{max}=1.2030 ve y_{kd}=1'dir. Buradan en büyük aşım değerinin yüzdesi, %20.30 bulunur.

Şekil 6.30'da Ters Sarkacın PID kontrol modelinin TDMA ağı için Matlab TrueTime çıkışı verilmiştir.



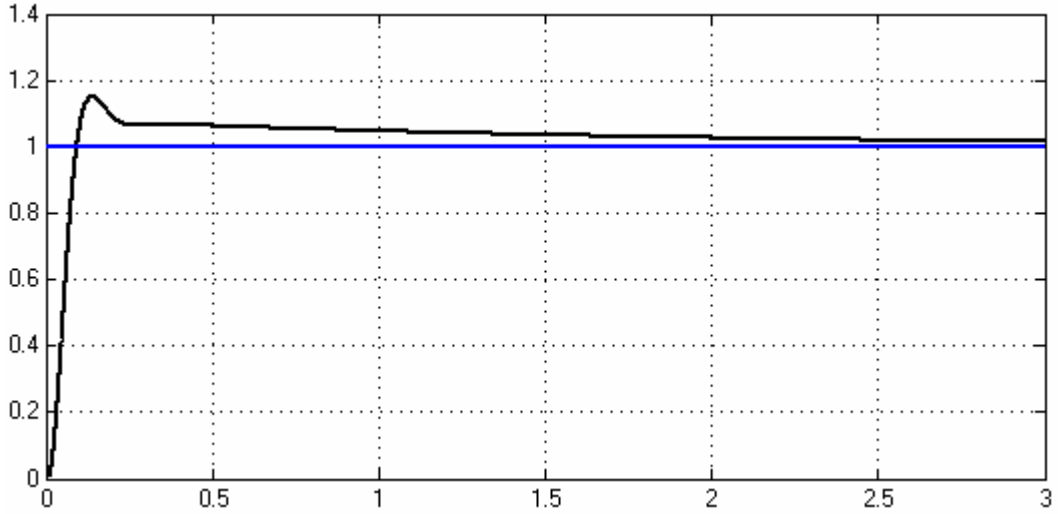
Şekil 6.30 Ters Sarkacın PID kontrol modelinin TDMA ağı için Matlab TrueTime çıkışı

TrueTime'da kullanılan TDMA ağında geçit sürecinin kalite parametreleri Şekil 6.31'den görülmektedir. Şekilden görüldüğü üzere, T_s=0.8600, y_{max}=1.1527 ve y_{kd}=1'dir. Buradan en büyük aşım değerinin yüzdesi, %15.27 bulunur.



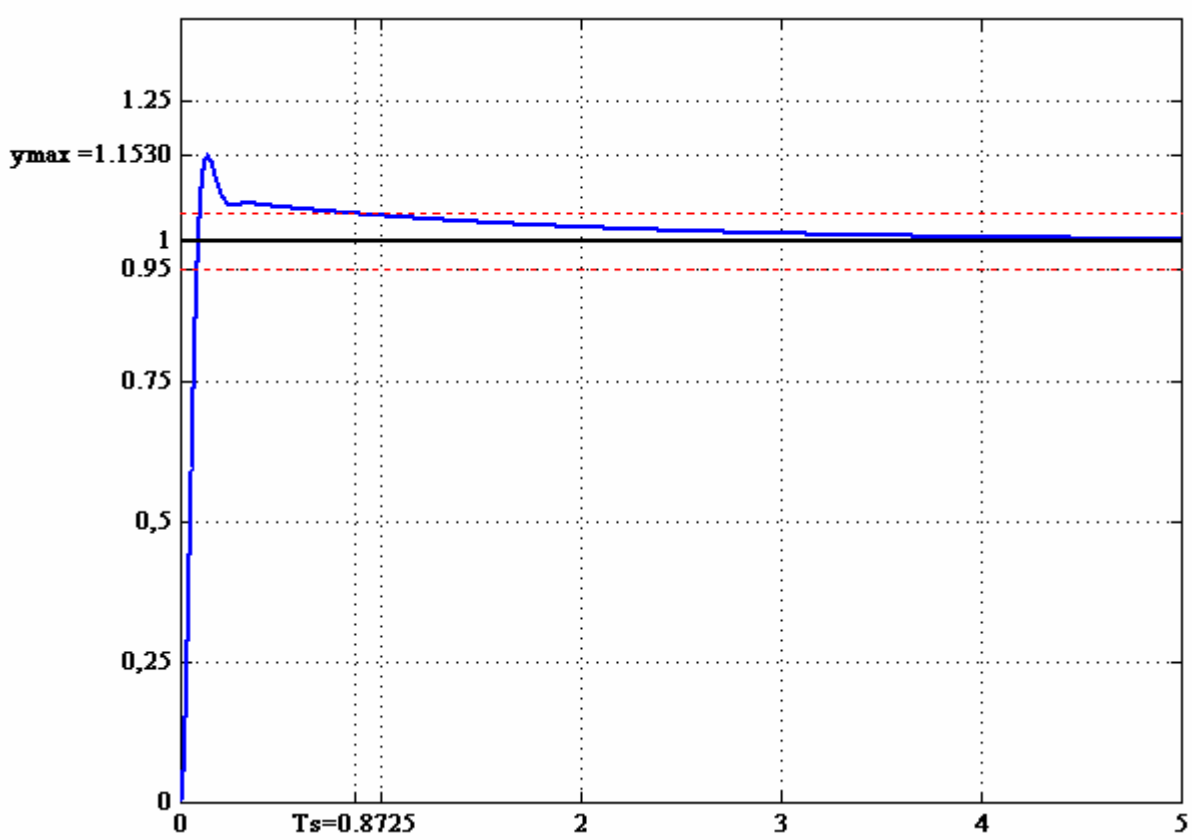
Şekil 6.31 Ters Sarkacın PID kontrol modelinin TDMA ağı için TrueTime çıkışında, sürecin en büyük değeri (y_{max}) ve yerleşme zamanı (T_s)

Şekil 6.32'de Ters Sarkacın PID kontrol modelinin Switched Ethernet ağı için Matlab TrueTime çıkışı verilmiştir.



Şekil 6.32 Ters Sarkacın PID kontrol modelinin Switched Ethernet ağı için Matlab TrueTime çıkışı

TrueTime'da kullanılan Switched Ethernet ağında geçit sürecinin kalite parametreleri Şekil 6.33'den görülmektedir. Şekilden görüldüğü üzere, T_s=0.8725, y_{max}=1.1530 ve y_{kd}=1'dir. Buradan en büyük aşım değerinin yüzdesi, %15.30 bulunur.



Şekil 6.33 Ters Sarkacın PID kontrol modelinin Switched Ethernet ağı için TrueTime çıkışında, sürecin en büyük değeri (y_{max}) ve yerleşme zamanı (T_s)

7. SONUÇ VE DEĞERLENDİRME

Yapılan araştırmalar sonucu aşağıdakiler ele alınmıştır.

1. Ağ üzerine kontrol sistemlerini simüle etmek için TrueTime Toolbox'ında Ters Sarkaç modeli tasarlanmış ve simüle edilmiştir. Denetleyici olarak PID kullanılmıştır.
2. Ağ üzerine kontrol kontrol sistemlerinde oluşan gecikmeler, kontrol sisteminin kalite parametrelerini etkilemektedir. Farklı ağlar için Ters Sarkaç Sisteminin PID kontrolü süreç çıkışları Bölüm 6.10.5'de bulunmuştur.
   - **2.1.** CSMA/CD(Ethernet) ağının yerleşme zamanı T_s=0.8738 ve en büyük aşım değerinin yüzdesi, %14.29'dur
   - **2.2.** CSMA/AMP(CAN) ağının yerleşme zamanı T_s=0.8738 ve en büyük aşım değerinin yüzdesi %14.29'dur.
   - **2.3.** Round Robin ağının yerleşme zamanı T_s=0.8746 ve en büyük aşım değerinin yüzdesi %14.98'dir.
   - **2.4.** FDMA ağının yerleşme zamanı T_s=0.8698 ve en büyük aşım değerinin yüzdesi %20.30'dur.
   - **2.5.** TDMA ağının yerleşme zamanı T_s=0.8600 ve en büyük aşım değerinin yüzdesi %15.27'dir.
   - **2.6.** Switched Ethernet ağının yerleşme zamanı T_s=0.8725 ve en büyük aşım değerinin yüzdesi %15.30'dur.
3. Sonuçlar karşılaştırıldığında karşımıza aşağıdaki çizelge çıkar.

Çizelge 7.1 Kullanılan ağlara göre sistemin en büyük aşım yüzdeleri ve yerleşme zamanları

Kullanılan Ağlar	**En Büyük Aşım (%)**	**Yerleşme Zamanı**
İdeal Ağ Sistemi(Bölüm 3.1)	35.32	0.2018
CSMA/CD(Ethernet)	14.29	0.8738
CSMA/AMP(CAN)	14.29	0.8738
Round Robin	14.98	0.8746
FDMA	20.30	0.8698
TDMA	15.27	0.8600
Switched Ethernet	15.30	0.8725

Çizelge 7.1'deki kalite parametrelerine göre ağlar kıyaslandığında;

- En büyük aşım % değeri en düşük olan sistemler CSMA/CD(Ethernet) ve CSMA/AMP(CAN) ağları olarak görülmektedir. Bu ağları takiben sırasıyla, Round Robin, TDMA, Switched Ethernet gelmektedir. Ağlar arasında en yüksek en büyük aşım değerine sahip ağ FDMA olarak görülmektedir. Ağın ideal olarak kabul edildiği örnekte ise en büyük aşım değeri tüm ağlara göre daha yüksektir. Halbuki ideal kabul edilen ağla verilmiş örnekte, ağ üzerinden kontrol edilen sistemdeki gürültü etkisi de bulunmamaktadır. Zira yine Bölüm 3.1'de verildiği gibi sisteme gürültü etkisi eklendiğinde sistem çıkışında bozulmalar devam etmekte ve süreç istenen kararlılığa ulaşamamaktadır. Burada ağların, salınım değerleri üzerinde ideal kabul edilmiş ağa nazaran olumlu etkisi olduğu gözlemlenmektedir. Bu olumlu etkinin nedenlerinden biri, ağ üzerine kontrol sistemi örneğimizde, sisteme giren sinyallerin A/D ve D/A dönüştürücülerden ve ağlardan geçerek işlenmesidir. Burada dönüştürücüler ve kullanılan ağlar bir nevi filtre görevi görmektedir.
- Yerleşim zamanı en düşük olan sistem, ideal kabul edilmiş ağ olarak görülmektedir. Daha sonra sırasıyla TDMA, FDMA, Switched Ethernet, CSMA/CD(Ethernet) ve CSMA/AMP(CAN) ağları görülmektedir. Yerleşim zamanı en yüksek olan ağ Round Robin olarak görülmektedir. Yerleşim zamanına göre kıyaslandığında ağlar arasında çok yüksek farklılıklar oluşmadığı görülmüştür. Ancak simüle edilen ağlarla, ideal kabul edilmiş ağ sistemi arasında kararlılığa ulaşma süresi bakımından oldukça büyük bir fark olduğu görülmektedir. Bu ağların, kararlılığa ulaşma süresi üzerinde ideal kabul edilmiş ağa nazaran olumsuz etkisi olduğu gözlemlenmektedir. Bu olumsuz etkinin nedeni ağlar üzerindeki gecikmeler olabilir.
- Genel olarak ağ etkisinin kontrol sistemleri simülasyonuna katılmasının, süreç çıkışında belirli farklılıklara neden olduğu gözlemlenmiştir. Buradan, ağ etkisi göz ardı edilerek yapılacak simülasyonların sonuçlarının, gerçek değerlere yeterince yakın olmadığı tespiti yapılabilir. Bu da ağ üzerine kontrol sistemlerinin önemini vurgulamaktadır.

KAYNAKLAR

[1] Cervin, A., Henriksson, D. ve Ohlin, M. (2010). TrueTime 2.0 beta-Reference Manual, Lund University.

[2] Hartman, J.R. (2004). Networked Control System Co-simulation for Co-design: Theory and Experiments, Case Western University.

[3] Nilsson, J. (1998). Real-Time Control System with Delays, Lunds Ofset AB Lund.

[4] Åström, K.J. ve Hägglund, T. (2006). Advenced PID Control, Lund University.

[5] Yüksel, İ. (2006). Otomatik Kontrol Sistem Dinamiği ve Denetim Sistemleri, Nobel Yayın Dağıtım.

[6] Ertürk, S. (2005). Sayısal Haberleşme, Birsen Yayınevi.

[7] Sultan, K. (2003). Inverted Pendulum Analysis, Design and Implementation, Sheffield Hallam University.

[8] Branicky, M. S. Liberatore, V. ve Phillips, S. M. (2003). Networked Control System Co-Simulation for Co-Design, in Proc. American Control Conference, Denver, USA, 4, 3341-3346.

[9] Czornik, A. ve Swierniak, A. (2003). On Direct Controllability of Discrete Time Jump Linear System, The Ohio State University.

[10] Deley, D. W. (2004). Controlling an Inverted Pendulum: An Example of a Digital Feedback Control System, http://members.cox.net/srice1/pendulum/synopsis.htm

[11] Hassibi, A., Boyd, S. P. ve How, J. P. (1999). Control of Asynchronous Dynamical Systems with Rate Constraints on Events, in Proc. IEEE Conference on Decision and Control, Phoenix, Arizona, USA, 2, 1345–1351.

[12] Henriksson, D., Cervin, A. ve Arzen K.E. (2002). TrueTime: Simulation of Control Loops Under Shared Computer Resources, in Proc. of the 15th IFAC World Congress on Automatic Control, Barcelona, Spain.

[13] Liberatore, V. (2002). Network Control Systems, http://vorlon.case.edu/~vxl11/NetBots/ncs.pdf

[14] Liberatore, V., Branicky, M. S., Phillips, S. M. ve Arora, P. (2004). Networked Control Systems Repository, http://home.cwru.edu/ncs/

[15] Messner, B. ve Tilbury, D. (1997). Control Tutorials for Matlab-Example: Modeling an Inverted Pendulum, http://www.engin.umich.edu/group/ctm/examples/pend/invpen.html

[16] Åström, K. J. ve Hägglund T. (1995). PID Controllers: Theory, Design, and Tuning, Instrument Society of America, Research Triangle Park, North Carolina.

[17] Otanez, P. (2002). Performance Optimization of Networked Control Systems, University of Michigan.

[18] Tammeraid, I., Majak, J., Pohjolainen, S. ve Luodeslampi, T. (1999). Applications of Linear Algebra with MATLAB-Example: A Cart and an Inverted Pendulum. http://www.cs.ut.ee/~toomas l/linalg/matlabap/control/control12.html

[19] Tzes, A., Nikolakopoulos, G. ve Koutroulis, I. (2003). Development and Experimental Verification of a Mobile Client-Centric Networked Controlled System, in Proc. European Control Conference, Cambridge, UK.
[20] Walsh, G. C., Ye, H. ve Bushnell, L. (1999). Stability Analysis of Networked Control Systems, in Proc. American Control Conference, San Diego, USA, 2876–2880.
[21] Xiao, L., Hassibi, A. ve How, J. P. (2000). Control with Random Communication Delays via a Discrete-Time Jump System Approach, in Proc. American Control Conference, Chicago, USA, 3, 2199–2204.
[22] Yu, M., Wang, L., Chu, T. ve Hao, F. (2003). An LMI Approach to Networked Control Systems with Data Packet Dropout and Transmission Delays, International Journal of Hybrid Systems, 3, 2-3.
[23] Zhang, W. (2001). Stability Analysis of Networked Control Systems, Case Western Reserve University.
[24] Ray, A. (1987). Performance evaluation of medium access control protocols for distributed digital avionics, Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control, 109, 370-377.
[25] Luck, R. Ve Ray, A. (1990). An observer-based compensator for distributed delays, Automatica, 26:5, 903-908.
[26] Åström, K. J. ve Wittenmark B. (1997). Computer-Controlled Systems, Prentice Hall.
[27] Törngren, M. (1995) Modelling and design of distributed real-time control applications, Royal Institute of Technology, KTH, Sweden.
[28] Olsson, G. ve Piani G. (1992) Computer Systems for Automation and Control, Prentice-Hall.
[29] Tindell, K. ve Hansson H. (1995). Real time systems and fixed priority scheduling, Uppsala University.
[30] Mırza, A. ve Hussain, S. (2000). Robust Controller for Nonlinear & Unstable System: Inverted Pendulum, AMSE Journal of Control & Design Simulation,55, 49-60. [www.amse-modeling.org]
[31] Mırza, A., Mahboob, I. ve Hussain S. (2001) Flexible Brom Balancing, AMSE Journal of C &D Simulation, 56, 1-2.
[32] MATLAB Help

EK-1

TRUETIME KURULUM VE ÇALIŞTIRMA KLAVUZU

Sistem Yazılım Gereksinimleri:

- TrueTime, MATLAB versiyonu için 6.1 (R12.1) veya daha üst sürümünü gerektirmektedir.
- Truetime, C++ versiyonu için de bir C++ Derleyici gerekmektedir.
  - Windows için Visual Studio C++
  - Linux için gcc, g++

Kurulum:

- Başlangıçta; TrueTime internet sitesinden «truetime-2.0» dosyasını indirip C:\Program Files\MATLAB\R2010b\toolbox içerisine kopyalanmalıdır.
- Truetime'ı çalıştırmak için TTKERNEL çevresel değişkeni tanımlanmalıdır. Bu, MATLAB'ın yeni versiyonlarında,

  ```
  setenv('TTKERNEL','C:\Program Files\MATLAB\R2010b\toolbox\
  truetime-2.0\kernel')
  ```

  komutu ile yapılabilir.
- Daha sonra MATLAB her açılışında gerekli Kernel yollarını bulabilmesi için çalışma(work) klasörü içerisine bir «startup.m» dosyası oluşturarak içerisine

  ```
  addpath([getenv('TTKERNEL')])
  addpath([getenv('TTKERNEL') '/matlab/help'])
  addpath([getenv('TTKERNEL') '/matlab'])
  ```

  yazarak, çalıştırılmalıdır.

TRUETIME arşivi önceden derlenmiş dosyaları içerir. Böylece derlemeler, API-M dosyalarına ihtiyaç duymadan TrueTime'da çalışabilir. Ancak, TRUETIME, derlenmiş olması şartıyla C++ kodunda yazılmış simülasyonları da destekler. Bu durumda, ilk olarak Matlab'da C++ derleyicisini yapılandırmak gereklidir. Bu, aşağıdaki komutlar kullanarak yapılabilir;

- "mex –setup" komutu girilip "Would you like mex to locate installed combiler [y]/n?" sorusu "y" olarak yanıtlanır.
- "Select a Compiler:" önerisinde "Visual C++" seçip, "make_truetime" komutuyla derleme tamamlanarak "TrueTime compiled succesfully!" yazısı görülür.
- Artık "truetime" komutu çalıştırılarak TrueTime başlatılabilir.
- Bundan sonra "Current Folder" kısmında içerisinde çalışılacak dosya açılır.
- Çalışılacak dosya içinde yeni bir model oluşturulup, TrueTime Library'den açılan model içine, kullanılacak model elemanları aktarılabilir.
- Oluşturmak istenilen model elemanının yapması istenilen işin kodlarından oluşan bir ".m" dosyası oluşturarak, ".m" dosyasının adı kernelin içerisine yazıldığında, kernel istenilen model elemanına dönüşmüş olacaktır.

EK-2
TRUETIME KOMUTLARI

Komut	**Açıklama**
ttInitKernel	Zamanlama politikası belirterek, kerneli başlat.
ttGetInitArg (yalnız C++)	Blok diyolog penceresinden giriş değişkenini al.
ttCreateTask	Periyodik olmayan bir görev oluştur.
ttCreatePeriodicTask	Periyodik bir görev başlat.
ttCreateLog	Bir günlük yapısı oluştur ve günlük verilerini belirle.
ttCreateHandler	Bir kesici işleyici oluştur.
ttCreateMonitor	Paylaşılan verileri koruması için bir izleme tertibatı (mutex) oluştur.
ttCreateEvent	Bir etkinlik (koşul değişkeni) oluştur.
ttCreateMailbox	Veriler arası iletişim için bir posta kutusu oluştur.
ttCreateSempahore	Bir semafor sayacı oluştur.
ttCreateCBS	Bir yumuşak(soft) veya sert(hard) bir bant genişliği sunucusu oluştur.
ttNoSchedule	Belirli görev yada kesici işleyici için zamanlama çizimini kapat.
ttNonPreemptible	Etkisizleştirilemez bir görev yap.
ttAttachTriggerHandler	Harici bir tetikleme için bir kesme işleyicisi ekle.
ttAttachNetworkHandler	Ağ arayüzü için bir kesici işleyici ekle.
ttAttachDLHandler	Göreve, zaman aşımı işleyicisi ekle.
ttAttachWCETHandler	Göreve, en kötü durum yürütme süresi aşımı kesicisi işleyicisi ekle.
ttAttachHook (yalnız C++)	Göreve, bir kernel zamanlama kancası ekle.
ttAttachCBS	Sabit bir bant genişliği sunucusuna bir görev ekle.
ttAbortSimulation	Simülasyonu durdur.
ttSetPeriod	Periyodik bir görevin periyodunu ayarla.
ttSetDeadline	Görevin ilgili son zamanını ayarla.
ttSetPriority	Görevin önceliğini ayarla.
ttSetWCET	Görevin en kötü durum yürütme (maksimum bütçe) zamanını ayarla.
ttSetData	Görevin yerel bellek veri yapısını güncelle.
ttSetAbsDeadline	Geçerli işin mutlak son zamanını ayarla.
ttSetBudget	Geçerli işin yürütme zaman bütçesini ayarla.
ttSetUserData	Keyfi kernel kullanıcı verisini ayarla (yalnız C++)
ttGetPeriod	Periyodik bir görevin periyodunu al.
ttGetDeadline	Bir görevin ilgili son zamanını al.
ttGetPriority	Görevin önceliğini al.
ttGetWCET	Görevin en kötü durum yürütme zamanını al.
ttGetData	Görevin yerel bellek veri yapısını geri al.
ttGetRelease	Geçerli işin bırakma süresini al.
ttGetAbsDeadline	Geçerli işin mutlak son zamanını al.
ttGetBudget	Geçerli işin yürütme zamanı bütçesini al.
ttSetUserData	Kernel kullanıcı verisini ayarla (yalnız C++).
ttGetUserData	Kernel kullanıcı verisini al. (yalnız C++).
ttSetCBSParameters	Sabit bant genişliği sunucusunun parametrelerini ayarla.
ttCreateJob	Bir görevin bir işini oluştur.
ttKillJob	Varsa, bir görevin çalışan(halihazırda) işini kes.
ttEnterMonitor	Bir izleme tertibatı (engellenmiş) gir.

ttExitMonitor	İzleme tertibatından çık.
ttWait	Bir olay(engellenmiş) için bekle.
ttNotify	Olay için bekleyen en yüksek öncelikli görevi bildir.
ttNotifyAll	Olay için bekleyen tüm görevleri bildir.
ttLogStart	Günlük için bir zamanlama ölçümü oluştur.
ttLogStop	Zamanlama ölçümünü durdur ve günlük içine kaydet.
ttLogNow	Geçerli saati tut.
ttLogValue	Skalar bir değeri tut.
ttTryPost	Mesajı posta kutusuna(engellenmemiş) gönder.
ttTryFetch	Posta kutusundan(engellenmemiş) mesajı çek.
ttPost	Mesajı posta kutusuna (engellenmiş) gönder.
ttFetch	Mesajı posta kutusuna (engellenmiş) gönder.
ttRetrieve	Posta kutusundan çekilen gerçek mesajı oku.
ttTake	Semaforu al.
ttGive	Semaforu ver.
ttCreateTimer	Bir tek-seferlik zamanlayıcı oluştur ve kesici işleyiciyi zamanlayıcıyla ilişkilendir.
TtCreatePeriodicTimer	Bir periyodik zamanlayıcı oluştur ve kesici işleyiciyle zamanlayıcıyı ilişkilendir.
ttRemoveTimer	Belirli bir zamanlayıcıyı kaldır.
ttCurrentTime	Her düğüm bazında simülasyonda geçerli saati al ve/veya ayarla.
ttSleepUntil	Zaman içinde belirli bir noktaya kadar uyu.
ttSleep	Zamanın belirli bir müddeti boyunca uyu.
ttAnalogIn	Analog giriş kanalındaki değeri oku.
ttAnalogOut	Analog çıkış kanalına değeri yaz.
ttSetNextSegment	Kod fonksiyonunda yürütülecek sıradaki segmenti ayarla (döngüler ve dallar uygulamak için).
ttGetInvoker	Harici tetiğin, ağ arayüzünün yada çalışan işleyiciyle tetiklenen aşma zamanlayıcısının adlarını al.
ttCallBlockSystem	Kod fonksiyonu içinden bir Simulink blok şeması çağır.
ttSendMsg	TRUETIME ağına (engellenmemiş) bir mesaj gönder.
ttGetMsg	TRUETIME ağına (engellenmemiş) ulaştığında bir mesaj al.
ttDiscardUnsentMessages	Gönderilmemiş tüm mesajları sil.
ttSetNetworkParameter	Her bir düğüm bazında belirli ağ parametrelerini ayarla.
ttSetKernelParameter	Her bir düğüm bazında belirli kernel parametrelerini ayarla.

EK-3

DC-SERVO TRUETIME TOOLBOX PID KONTROL ÖRNEĞİ

KOMUT DOSYALARI

PID-kontrol örneği için başlatma komut dosyası[1]

```
function servo_init(mode)
ttInitKernel(2, 1, 'prioFP'); % nbrOfInputs, nbrOfOutputs, fixed priority
period = 0.006;
deadline = period;
offset = 0.0;
prio = 1;
data.K = 0.96;
switch mode,
case 1,
% UYGULAMA 1: periyodik görevler için yerleşik desteği kullanarak
ttCreatePeriodicTask('pid_task',offset,period,prio,'pidcode1',data);
case 2,
% UYGULAMA 2: kod fonksiyonuyla birlikte Simulink bloğu çağırma
data2.u = 0;
ttCreatePeriodicTask('pid_task',offset,period,prio,'pidcode2',data2);
case 3,
% UYGULAMA 3: sleepUntil ve loop back
data.t = 0;
ttCreateTask('pid_task',deadline,prio,'pidcode3',data);
ttCreateJob('pid_task');
case 4,
% UYGULAMA 4: zaman işleyici, tetikleme görev işi içinde örnekleme
hdl_data.yChan = 2;
ttCreateInterruptHandler('timer_handler',prio,'samplercode',hdl_data);
ttCreatePeriodicTimer('timer',offset,period,'timer_handler');
ttCreateMailbox('Samples',10);
ttCreateTask('pid_task',deadline,prio,'pidcode4',data);
end
```

EK-4

TERS SARKAÇ TRUETIME TOOLBOX PID KONTROLÜ

KOMUT DOSYALARI

Düğüm 1'nin birim fonksiyonları

interference_init.m:

```
function interference_init(arg)
    % Dağıtılmış kontrol sistemi: parazit düğümü
    % Bant genişliği parametresiyle (blok maske içinde verilen) rahatsız edici bir
    % ağ trafiği oluştur

    % TrueTime kerneli başlat
ttInitKernel('prioFP');  % sabit öncelikli zamanlama

    % Parazit görevi
period = 0.001;
offset = 0.0005;
data = arg;
ttCreatePeriodicTask('interference_task', offset, period,
'interference_code', data);
```

interference_code.m:

```
function [exectime, data] = interference_code(seg, data)
BWshare = data;
ran = ttAnalogIn(1);
if ran < BWshare
  ttSendMsg(1, 1, 80);    % kendi içine 80 bit gönder
end
while ~isempty(ttGetMsg) % (eğer varsa) eski alınan mesajları oku
end
exectime = -1;
```

Düğüm 2'in birim fonksiyonları

sensor_actuator_init.m:

```
function sensor_actuator_init
    % Dağıtılmış sensör düğümü: sensör düğümü
```

```
    % Tesisi örnekle ve örnekleri denetleyici düğüme gönder.
    % Denetleyiciden gönderilen kontrolleri çalıştır.

    %TrueTime Kernel'i başlat.
    ttInitKernel('prioDM');    % son-monoton zamanlama

    % Periyodik sensör görevi
starttime = 0.0;
period = 0.010;
ttCreatePeriodicTask('sensor_task',  starttime,  period,
'sensor_code');

    % Arasıra çalıştırıcı görev
deadline = 10.0;
ttCreateTask('actuator_task', deadline, 'actuator_code');

    % Ağ işleyicisi
prio = 1.0;
data = 'actuator_task';
ttCreateHandler('network_handler',                  prio,
'nwhandler_code', data);
ttAttachNetworkHandler('network_handler')
```

sensor_code.m:

```
function [exectime, data] = actuator_code(seg, data)
persistent u
switch seg
 case 1
  u = ttGetMsg;
  exectime = 0.0005;
 otherwise
  if ~isempty(u)
    ttAnalogOut(1, u)
```

```
  else
    disp('Error: actuator received empty message!')
  end
  exectime = -1; % bitiş
end
```

actuator_code.m:

```
function [exectime, data] = actuator_code(seg, data)
persistent u
switch seg
 case 1
  u = ttGetMsg;
  exectime = 0.0005;
 otherwise
  if ~isempty(u)
    ttAnalogOut(1, u)
  else
    disp('Error: actuator received empty message!')
  end
  exectime = -1; % bitiş
end
```

Düğüm 3'ün birim fonksiyonu:

controller_init.m:

```
function ccontroller_init(arg)
      % Dağıtılmış kontrol sistemi: denetleyici düğümü.
      % Sensör düğümden mesajları al, kontrol sinyalini hesapla ve çalıştırıcı
      % düğüme gönder. Ayrıca yüksek öncelikli rahatsız edici görev içer.

      % TrueTime kerneli başlat
ttInitKernel('prioDM')     % son-monoton zamanlama
```

```
% Görev verisini oluştur (yerel bellek)
data.h = 0.01;
data.K = 226.7593;
data.Td = 0.0549;
data.Ti = 1.85;
data.N = 100000.0;
data.ad = data.Td/(data.N*data.h+data.Td);
data.bd = data.N*data.K*data.ad;
data.yold = 0;
data.Dold = 0;
data.Iold = 0;
data.u = 0;

% Arasıra denetleyici görevi
deadline = data.h;
ttCreateTask('controller_task',deadline,'controller_code'
,data);

% Yüksek öncelikli periyodik sahte görev
starttime = 0.0;
period = 0.007;
data = period*arg;
ttCreatePeriodicTask('dummy_task',   starttime,   period,
'dummy_code', data);
% Ağ kesme işleyicisini oluştur ve ekle
prio = 1.0;
data = 'controller_task';
ttCreateHandler('network_handler',1,'nwhandler_code',
data)
ttAttachNetworkHandler('network_handler')
ttNoSchedule('network_handler');
```

controller_code:

```
function [exectime, data] = controller_code(seg, data)
switch seg
 case 1
  y = ttGetMsg;      % Sensör değerini belirle
  if isempty(y)
    disp('Error in controller: no message received!');
    y = 0.0;
  end
   r = ttAnalogIn(1);      % Referans değerini belirle
   P = data.K*(r-y);
   I = data.Iold;
   D = data.Td/(data.N*data.h+data.Td)*data.Dold+data.N*
   data.K*data.Td/(data.N*data.h+data.Td)*(data.yold-y);
   data.u = P + I + D;
   data.Iold = data.Iold + data.K*data.h/data.Ti*(r-y);
   data.Dold = D;
   data.yold = y;
   exectime = 0.0005;
case 2
  ttSendMsg(2, data.u, 80);    % Düğüm2'ye 80 bit gönder (çalıştırıcı)
  exectime = -1;     % bitiş
end
```

dummy_code.m:

```
function [exectime, data] = dummy_code(seg, data)
switch seg
 case 1
  exectime = data;
 otherwise
  exectime = -1;
end
```

nwhandler_code.m:

```
function [exectime, data] = nwhandler_code(seg, data)

ttCreateJob(data)
exectime = -1;
```

ÖZGEÇMİŞ

Kişisel bilgiler	
Adı Soyadı	GülsümYILDIRIZ
Doğum Yeri ve Tarihi	Isparta, 22/04/1983
Medeni Hali	Evli
Yabancı Dil	İngilizce
İletişim Adresi	Sivas Numune Hastanesi Rahmi Günay cd., 58040-Sivas
E-posta Adresi	gulsum_yilmazer@yahoo.com

Eğitim ve Akademik Durumu	
Lise	Kütahya Fen Lisesi, 2001
Lisans	Gazi Üniversitesi, 2007

İş Tecrübesi	
Türk Telekom	Santral Mühendisi, 2008-2010
Sivas Numune Hastanesi	Elektrik Elektronik Mühendisi, 2010-

Ödüller, Teşvikler ve Üyelikler	
TMMOB/EMO	Asil Üye, 2007-