İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

UZUN KISA SÜRELİ BELLEK TABANLI SİSTEM TANIMA VE UYARLAMALI KONTROL

YÜKSEK LİSANS TEZİ

Çağatay SANATEL

Mekatronik Mühendisliği Ana Bilim Dalı

Mekatronik Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Gülay ÖKE GÜNEL

TEMMUZ 2020

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

UZUN KISA SÜRELİ BELLEK TABANLI SİSTEM TANIMA VE UYARLAMALI KONTROL

YÜKSEK LİSANS TEZİ

Çağatay SANATEL
(518161033)

Mekatronik Mühendisliği Ana Bilim Dalı

Mekatronik Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Gülay ÖKE GÜNEL

TEMMUZ 2020

İTÜ, Fen Bilimleri Enstitüsü'nün 518161033 numaralı Yüksek Lisans Öğrencisi olan Çağatay SANATEL, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "UZUN KISA SÜRELİ BELLEK TABANLI SİSTEM TANIMA VE UYARLAMALI KONTROL" başlıklı tezini aşağıdaki imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Doç. Dr. Gülay ÖKE GÜNEL** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. İbrahim Eksin** İstanbul Teknik Üniversitesi	.....
	Dr. Öğr. Üyesi Figen Özen Haliç Üniversitesi	.....

Teslim Tarihi : **8 Ağustos 2020**
Savunma Tarihi : **9 Temmuz 2020**

Aileme ve dostlarıma,

ÖNSÖZ

Tez konusunu seçerken ve yapım aşamasında bana büyük yardımları dokunan tez danışmanın sayın Doç.Dr.Gülay ÖKE GÜNEL'e teşekkürlerimi sunar, tez yazma sürecinde beni motive eden, tüm eğitim hayatım boyunca benden maddi ve manevi desteklerini esirgemeyen sevgili aileme ve bu süreçte hep yanımda olan dostlarıma teşekkürlerimi bir borç bilirim.

TEMMUZ 2020 Çağatay SANATEL

İÇİNDEKİLER

Sayfa

KISALTMALAR

UKSB : Uzun Kısa Süreli Bellek (Long Short Term Memory)
RNN : Tekrarlı Yapay Sinir Ağları (Recurrent Neural Network)
MRAC : Model Referans Adaptif Kontrol(Model Referance Adaptive Control)
SRC : Kendinden Ayarlamalı Kontrol (Self Regulator Control)
SID : Sistem Tanımlama (System Identification)
PID : Oransal Integral Türevsel Kontrolcü (Proportional Integral Derivative)
PRBS : Sözde Rassal Sayı Üreteci (Pseudo Random Binary Sequence)
ARMA : Otoregresif Entegre Hareketli Ortalama (Autoregressive–Moving-Average)
NARX : Doğrusal Olmayan Otoregresif Eksojen (Nonlineer Autoregressive Exogenous)
AR : Otoregresif Entegre (Autoregressive)
MA : Hareketli Ortalama (Moving Avarage)
GRU : Geçitli Tekrarlama Ünitesi (Gated Recurrent Unit)
CSTR : Sürekli Karıştırmalı Tank Reaktörü (Continuous Stirred-Tank Reactor)
CNN : Evrişimsel Sinir Ağları (Convolutional Neural Network)

SEMBOLLER

t	**:** Zaman
u(t)	**:** Sistem Giriş İşareti
u(t+k)	**:** k Adım Sonraki Giriş İşareti
y(t)	**:** Sistem Çıkışı
y(t+k)	**:** k Adım Sonraki Sistem Çıkışı
y_p	**:** Tahmin Edilen Sistem Çıkışı
v(t)	**:** Gürültü İşareti
q^{-t}	**:** t Adım Geriye Kaydırma Operatörü
ω	**:** Açısal Frekans
ϕ	**:** Faz
W	**:** Yapay Sinir Ağı Ağırlık Değeri
b	**:** Yapay Sinir Ağı Sapma Değeri
Z^{-t}	**:** t Adım Gecikme Elemanı
Out_t	**:** t Anındaki UKSB Çıkışı
X_t	**:** t Anındaki UKSB Girişi
C_t	**:** t Anındaki UKSB Bağlam Çıkışı
W_F	**:** Unutma Kapısı Ağırlık 1
U_F	**:** Unutma Kapısı Ağırlık 2
b_F	**:** Unutma Kapısı Sapma Değeri
W_A	**:** Aktivasyon Kapısı Ağırlık 1
U_A	**:** Aktivasyon Kapısı Ağırlık 2
b_A	**:** Aktivasyon Kapısı Sapma Değeri
W_O	**:** Çıkış Kapısı Ağırlık 1
U_O	**:** Çıkış Kapısı Ağırlık 2
b_O	**:** Çıkış Kapısı Sapma Değeri
W_I	**:** Giriş Kapısı Ağırlık 1
U_I	**:** Giriş Kapısı Ağırlık 2
b_I	**:** Giriş Kapısı Sapma Değeri
tanh	**:** Tanjant Hiperbolik Operatörü
σ	**:** Sigmoid Operatörü
K_p	**:** Oransal Katsayı
K_i	**:** İntegratör Katsayı
K_d	**:** Türevsel Katsayı

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

UZUN KISA SÜRELİ BELLEK TABANLI SİSTEM TANIMA VE UYARLAMALI KONTROL

ÖZET

Son yıllarda derin öğrenme algoritmalarının popülerliği hızla artmıştır. Derin öğrenme sayesinde farklı alanlardaki pek çok karmaşık problem çözülebilmiş bu sayede de derin öğrenme, mühendislik ve tıp gibi çok önemli alanlarda kendisine yer bulabilmiştir. Ayrıca işlemci teknolojilerinin gelişmesi ile işlem süreleri azalmaya başlamış ve derin öğrenme algoritmalarının gerçek sistemlerde uygulanabilirlikleri artmıştır. Derin öğrenme algoritmalarının bu denli hızlı gelişmesi sayesinde kontrol alanında da daha güçlü, çevresel faktörlere daha duyarlı ve hızlı cevap veren kontrolörler geliştirilebilmektedir. Geliştirilen kontrolörlerin bir sistemi kontrol edebilmesi için sistemi tanımlayan bir matematiksel model gerekmektedir. Bu matematiksel modeli kullanarak sistemler için uygun kontrolörler tasarlanabilir. Bir sistemin matematiksel modelini elde etmek her zaman kolay bir iş değildir. Bazı durumlarda matematiksel model sistemi tanımlamakta yetersiz kalabilir. Bu tip durumlarda sistemin giriş-çıkış verilerini kullanılarak sistem tanımlama gerçekleştirilir ve sistem için uygun bir matematiksel model, gerçek sistemden elde edilen giriş-çıkış verileri kullanılarak oluşturulur. Çalışmanın ilk bölümlerinde UKSB yani Uzun Kısa Süreli Bellek mekanizması ve sistem verileri kullanılarak bir sistem tanımlama gerçekleştirilmiştir. Bu aşamada sistem tanımlama için iki adet örnek sistem belirlenmiş ve testler bu sistemler üzerinde gerçekleştirilmiştir. Sistem tanımlamanın hemen ardından çalışmanın ikinci kısmında, yine UKSB tabanlı bir kendinden parametre ayarlı kontolör kullanılarak referans bir işaretin takibi sağlanmıştır. UKSB tabanlı kendinden parametre ayarlı kontrolör için UKSB ileri ve geri fazı, matematiksel olarak yeniden düzenlenmiştir. İkinci kısımda tasarlanan UKSB tabanlı kontrolör, örnek sistemlerin hem matematiksel modelleri hem de UKSB tabanlı sistem tanımlama modelleri üzerinde denenmiştir. Çalışmada sistem tanımlama ve kontrolör uygulamaları için Uzun Kısa Süreli Bellek (UKSB) kullanılmasının en büyük nedenlerinden biri geçmişe dönük sistem verilerini kullanarak sistemi diğer makine öğrenmesi algoritmalarına göre daha iyi modelleyebilmesidir. Bu sayede kontrol için kullanılacak uyarlamalı modelin gerçek modele daha benzer davranışlar sergilemesi ve sistem çıktılarının referans işareti daha yakından ve daha hızlı yerleşme zamanlarında takip edebilmesi sağlanır. Genel olarak çalışmanın ilk kısımlarında UKSB tabanlı sistem tanımlama ile örnek sistemlerin verilerinden matematiksel model elde edilmiş ardından da UKSB tabanlı bir uyarlamalı kontrol algoritması için derin öğrenme içerikli yeni bir yöntem sunulmuştur. Geliştirilen bu yöntem örnek sistemler üzerinde denenmiştir. Çalışma kapsamında sistem tanımlama, Uzun Kısa Süreli Bellek ve uyarlamalı kontrolör konuları hakkında da genel bilgiler verilmiştir. Çalışmanın sonuç kısmında ise elde edilen çıktılar klasik yapay sinir hücreleri ile oluşturulmuş bir uyarlamalı kontrolör çıktıları ile karşılaştırılmış ve birbirlerine göre üstünlükleri ve eksiklikleri tartışılmıştır.

LONG SHORT TERM MEMORY BASED SYSTEM IDENTIFICATION AND ADAPTIVE CONTROL

SUMMARY

In recent years, the popularity of deep learning has increased rapidly. Thanks to deep learning, many problems can be solved in diverse areas like engineering and medicine. Also with the rapid improvement of processor technology, computation times began to decrease and in this way, deep learning algorithm becomes feasible. With deep learning technology, it has become possible to implement control applications with better performance and with more sensitivity to changes.

In order to develop a controller, the mathematical model of the system to be controlled is needed. Finding a mathematical model of a system is not always an easy task. In some cases, defining the mathematical model system may be inadequate. In such cases, system identification is performed using the input-output data of the system and a mathematical model for the system could be created by using input-output data that is obtained from the real system.

In the second chapter of the study, system identification is mentioned. There are 3 types of system identification. These types are *white box system identification*, *grey box system identification* and *black box system identification*. Black box system identification is used throughout this study. In the black box system identification, we assume that we do not have the mathematical equation of the model, and the system model is derived only from system input-output data. Since machine learning is used in system identification, NARX model was preferred throughout the study. In NARX model, the model relates the current value of a time series to both "past values" and "current and past values of the exogenous series". In order to extract the data to be used in system identification, the input signal to be applied to the system must be determined. Most preferred input signals for this type of applications are *step function*, *pseudo random binary sequence*, *auto-regressive integrated moving average* and *sum of sinusoids*.In the study, pseudo-random binary sequence (PRBS) was used because it gives ideal results for system identification.

In the third chapter of the study, information about machine learning methods and the Long Short Term Memory (LSTM) architecture are given. There are three types of machine learning, these are supervised learning, unsupervised learning, and reinforcement learning. Supervised learning was used in both system identification and self-tuning regulator design. The study also includes general information about Long Short Term Memory and self-tuning regulator topics. The most important reason for using Long Short-Term Memory in the system identification and self-tuning regulator is that the Long Short-Term Memory can use historical data so in this way the system can be successfully modeled. This section also includes mathematical expressions of Long Short Term Memory forward and backward phase. Backpropagation algorithms were performed one by one to update the weights. In addition, information about different Long Short Term Memory cell architectures is given in this section.

In the fourth chapter of the study, Long Short Term Memory based system identification tested. A system identification application was carried out on two sample systems. The first system is a non-linear test system and the second system is a non-linear Continuous Stirred Tank Reactor (CSTR) system. Since real systems are not available, input-output data was generated using the system's mathematical models. For non-linear Continuous Stirred Tank Reactor, Kravaris and Palanki equations are used. In this section, Long Short Term Memory has been successful in identification of the system, and results have been plotted for training and testing results. Also in this section, *mean square error*, *mean absolute error*, *median absolute error*, *explained variance score* and *R2 score* are given for both train result and test result. For Long Short Term Memory based system identification, Python KERAS is used. The KERAS library is a fast and reliable way for machine learning algorithms.

In the fifth chapter of the study, after the system identification phase, a reference signal tracking is provided by using a Long Short Term Memory based self-tuning regulator controller. A new method with deep learning content is presented for a Long Short Term Memory based adaptive control algorithm. This new control method has been tested on test systems In this part, discrete time PID controller was used for the self-tuning regulator.

Also, Long Short Term Memory forward and backward phase formulas were rearranged and mathematically derived for the self-tuning regulator controller. The self-tuning regulator produces the input signal to be given to the real system so the system can follow the reference signal. new control method results have been tested on both the test system's mathematical models and system identification models. With the input signal generated with Long Short Term Memory based self-tuning regulator, the reference signal could be tracked very closely. In addition, the generated method produced low settling time results. Also in this chapter, system response with a feedback signal to containing measurement noise was investigated for robust control application. 60 dB and 40 dB white noise were applied to the test systems.

In the conclusion chapter of the study, the obtained outputs were compared with an adaptive controller output created by classical artificial neural cells, and pros and cons are discussed. A classical artificial neural network contains 2 input cells, 3 hidden cells, and 1 output cell. This artificial neural network generates an input signal that can directly follow the reference signal but Long Short Term Memory based self-tuning regulator generates appropriate K_P, K_i, and K_D signals and then the PID controller unit uses these values to generate the right input for tracking the reference signal. Long Short Term Memory based self-tuning regulator gave better settling time than the classical neural network controller on the other hand Long Short Term Memory based self-tuning regulator works slower. Within the scope of the study, Long Short Term Memory based self-tuning regulator's running time for the test system is 0.23867 and the classical neural network controller's running time is 0.20125. The difference between the two running times is not too great. In this way, a new control method has been developed and a deep learning architecture can be used in the adaptive control area.

The study aims to find solutions to these questions, "Can a mathematical model be predicted from system data for the system without a mathematical model?" and "Can a new adaptive control method involving deep learning algorithms be developed?". In addition, in the study, it was tried to create a new method for adaptive control and

system identification applications with deep learning. Nowadays, when deep learning is widespread, this method is aimed to implement control applications with deep learning area. A minimize the error signal based adaptive control has been developed for systems that do not have mathematical models or systems that are difficult to extract mathematical models. In future work, this new method can lead to new control applications in future studies and new deep learning algorithms can be applied to the adaptive control field. Different results can be examined by changing the Long Short Term Memory cell type. By testing in different control systems,new solutions can be produced for different problems.

1. GİRİŞ

Adaptasyonun çıkış noktası incelendiğinde temelinin canlı organizmalara dayandığı görülür. Canlı organizmalar, nesillerini devam ettirebilmek adına bulundukları ortama adapte olmaya çalışırlar. Bu adaptasyon sürecinde, çevresel olarak kendilerine dezavantaj oluşturacak olumsuz özelliklerinden kurtularak çevresel bakımdan kendisine avantaj sağlayacak olumlu özellikleri eklemeye çalışırlar. [15] Süregelen bu adaptasyonlar sonucunda organizmalar bulundukları çevrede kendileri için en avantajlı özelliklerin ne olduğunu öğrenmeye başlar. Bu süreç sonrasında ise çevresel şartlara ve değişimlere daha dayanıklı hale gelirler.

İnsanoğlu doğadaki pek çok yapıyı mühendislik alanlarına uyarladığı gibi canlılara ait adaptasyon özelliğini de mühendislik alanına uyarlamış ve bu uyarlama sayesinde geliştirdiği sistemler uyarlamalı hale gelerek değişimlere karşı dayanıklı olmaya başlamıştır. Bu mühendislik alanlarının en yaygın olanlarından biri de kontrol alanıdır. Uyarlamalı kontrol, özellikle değişken koşullar içeren kontrol problemlerinde ve klasik yöntemlerin zorlandığı çözümlerde yardımımıza koşar. Uyarlamalı kontrol sistematik bir yaklaşım sağlayan bir dizi tekniği kapsayan bir yapıdır. Uyarlamalı kontrol içeren sistemler istenilen seviyelere gerçek zamanlı yani dinamik olarak ayarlanır. Uyarlamalı kontrol tekniklerinde kapalı döngüde otomatik olarak ayarlar yapılır. Genel olarak bir sistemin modeli bilinmiyor ya da zamana göre değişiklik gösteriyorsa uyarlamalı kontrol tercih edilmektedir. [16] Sıklıkla kullanılan uyarlamalı kontrol tipleri; kazanç planlamalı kontrol, model referans kontrol ve kendinden ayarlamalı regülatörlerdir. [17]

Kazanç planlamalı kontrolörler, sistem için önceden tanımlanmış senaryolara uygun olarak, sistemin çalışma aralıkları belirlenerek ve bu çalışma aralıklarında uygun kontrolör parametreleri hesaplanarak oluşturulurlar. Tam anlamıyla bir adaptasyon söz konusu değildir ancak çalışma aralığı değişimlerinde kazanç değerlerinde bir değişim söz konusu olduğundan uyarlamalı yapılar olarak isimlendirilebilirler. Bu yapılarda her çalışma aralığı bir karar ağacına benzetilebilir ve koşullara uygun kontrolör

parametreleri bu karar ağaçlarına göre üretilmektedir. Karar ağaçları sistem için o andaki en uygun parametreyi seçmektedir. [16]

Model referans kontrolörler yada kısaca *MRAC*, bir referans model oluşturarak, kapalı kontrol sistemini bu referans model ile ortak davranışa zorlayan sistemlerdir. MRAC'nin amacı kapalı sistem ile referans sistem arasında bir yakınsama sağlamaktır. Referans model ve kapalı sistem model çıkışları karşılaştırılarak fark en aza indirilmeye çalışılır ve bu bağlamda parametreler hesaplanır. [16]

Kendinden ayarlamalı regülatörler yada kısaca *STR*, sistem karakteristiklerinin ve dış etkilerin tekrarlı olarak tahmin edilmesine, izlenmesine ve bu bilgiler kullanılarak optimal kontrolörü tasarlamak ilkesine dayanmaktadır. Bu yapılar sürekli olarak güncellenerek oluşturulduğundan sistem içerisinde gözlemlenemeyen ifadeleri de içerebilmektedir. [15]

Yukarıda belirtilen tüm bu yöntemler temelinde kontrol edilecek sistemin bir matematiksel modeline ihtiyaç duyarlar. Matematiksel modeller genel olarak sistemlerin girişleri ve gürültü sinyalleri üzerinden sistem çıkışları üretmemizi sağlayan yapılardır. Bir sistem için üzerinde yorum yapabileceğimiz ve sistemi gerçeklemesek bile durumlarını inceleyebileceğimiz sistemi tanımlayan denklemler kümesidir. [18] Ancak bazı durumlarda sistemi tanımlayan matematiksel model elimizde bulunmayabilir veya sistemi tam olarak ifade edemeyebilir. Bunun dışında sistemi doğrusal olarak değil de doğrusal olmayan bir denklem olarak ifade etmek istediğimiz durumlarla da karşılaşabiliriz. İşte bu gibi durumlar ile başa çıkabilmek için gerçek sistem veya simulasyon üzerinden elde edilen veriler kullanarak sistemin matematiksel modelini üretmeye ya da başka bir deyişle ***sistemi tanımlamaya (System Identification)*** yönelinir.

Sistem tanımlama, sistemin dinamik modellerini, sistem verilerden elde etmeye çalışmaktır. Kontrol teorisi, istatistik ve sinyal işleme alanlarında sıklıkla kullanılır. [5] Sistem hakkında elimizde bir denklem olmadığı veya kısmi bir denklem olduğu düşünülür. Tam bir denklem yerine elimizde sistemden elde edilen giriş ve çıkış değerleri vardır. Bu giriş-çıkış verileri kullanılarak sistem için matematiksel model yerine geçebilecek bir yapı oluşturulmaya çalışılır. Sistem tanımlama için günümüze kadar pek çok yöntem geliştirilmiştir ancak günümüzde yapay öğrenme algoritmaları

sıklıkla kullanılmaktadır. Yapay öğrenme algoritmaları çok çeşitlilik göstermesine karşın değişen problemlere karşı bazı yapay öğrenme algoritmaları daha iyi sonuç verirken bazıları kötü sonuçlar verebilmektedir. Çalışma kapsamında sistemlerin matematiksel modellerinin bilinmediği düşünülerek, kontrolör tasarlamadan önce ilk adım olarak sistemlerin verilerinden bir matematiksel model elde edilecek yani bir sistem tanımlama uygulaması geliştirilecektir. Burada geliştirilen matematiksel model aslında bir yapay öğrenme ağı olacak ve kendisine verilen girişlere göre sistemin gelecek değerlerini üretecektir. Bu işlem için tercih edilen yapay öğrenme ağı ise ***Uzun Kısa Süreli Bellek (Long Short-Term Memory - LSTM)*** ağıdır.

UKSB ağları günümüzde, özellikle geriye dönük verileri kullanması bakımından sıklıkla tercih edilmektedir. UKSB kullanılarak yazı tanımlama [19] [20], Çince gibi karmaşık dillerin karakterlerinin tanımlanması [21], sensör anomali tespiti [22], biomedikal ve volumetrik görüntü işleme [23] ve robotikte dinamik yol planlama [24] gibi pek çok uygulama geliştirilmiştir. Bunların yanı sıra UKSB geçmiş verileri kullanması bakımından dinamik sistem tanımlama [25] ve kontrol [26] alanlarında da sıklıkla tercih edilir. Kontrol ve sistem tanımlama alanlarında özellikle diferansiyel denklemlerle ifade edilen dinamik sistemlerde oldukça başarılı sonuçlar üretebilmektedir. [27] UKSB ağlarının uygulamalarda bu kadar sık tercih edilmesinin nedenleri klasik ağlardan farklı olarak bellek yapısı barındırması, gürültüler ile başa çıkabilmesi ve girişler arasındaki sıra bilgisini eğitiminin bir parçası olarak görebilmesidir. [28] Bu sayede bir sistem içerisinden gelen veriler arasında bağlantı kurarken sekans bilgisini de kullanmış olur ve gerçek sisteme daha yakın bir sistem modeli elde edebilir. Gerçek bir sistem için eğitilmiş bir ağ, sistemi iyi modeller ise bu durumda eğitilmiş bu ağa vereceğimiz girişler sonucunda üretilecek gelecek veriler gerçek sistem çıkışları ile yüksek yakınsaklıkta olacaktır. Sistem tanımlama aşamasında iyi tanımlanan sistemler için ikinci aşama olan kontrol kısmının da performansı artacaktır.

Sistem tanımlama adımının ardından kontrol kısmı gelir. Sistem tanımlama sonucunda elde edilen gelecek veriler kullanılarak UKSB tabanlı ***Kendinden Ayarlamalı Regülatör (STR)*** tasarlanmıştır. Tasarlanan bu kontrolör, örnek sistemlerin bir referans işareti takip etmesini sağlar. Kullanılan kontrol metodu olarak ***Oransal Integral Türevsel*** yani ***PID*** kullanılmıştır. Doğrusal olmayan şekilde bir sistemin yapay

öğrenme ile modellenmesi PID kontrolörünün dayanıklılığını arttırır ve kontrolörü güçlü kılar. [17] Bu kısımda sistem verilerine göre sistemin modelinin bir yapay öğrenme yöntemi ile güncellenebilmesi ve değişen şartlara uyum sağlayabilmesi bakımından bir adaptasyon söz konusudur. Kontrol edilecek sistem değişse bile kontrolör parametrelerini bu sisteme göre uyarlayacaktır.

Çalışma kapsamında ilk bölümde sistem tanımlama hakkında genel bilgilere yer verilmiştir. Sistem tanımlamanın ne olduğu, sistem tanımlama tipleri ve sistem tanımlamada kullanılan giriş işaretlerinden bahsedilmiştir. Ayrıca yine bu kısımda NARX sistem modeli hakkında da bilgiler mevcuttur. Çalışmada makine öğrenmesi tabanlı bir sistem tanımlama yöntemi kullanıldığından ikinci bölümünde yapay öğrenme kavramları ve yapay öğrenme tiplerinden bahsedilmiştir.

Uygulama kapsamında kullanılan yapay öğrenme algoritması Uzun Kısa Süreli Bellek olduğu için çalışmanın üçüncü bölümünde Uzun Kısa Süreli Bellek hücreleri ve ağları hakkında bilgiler mevcuttur. Özellikle bu bölümde ileri ve geri fazlar matematiksel çıkarımlar üzerinden açıklanmaya çalışılmış bölümün sonlarında ise farklı tipteki Uzun Kısa Süreli Ağ hücreleri hakkında bilgiler verilmiştir. Bu kısım çalışma genelinde kullanılan Uzun Kısa Süreli Ağ yapısının önsel hazırlık kısmıdır.

Dördüncü bölümde Uzun Kısa Süreli Bellek tabanlı sistem tanımlama adımları ve örnek sistemler üzerindeki sonuçları incelenmiştir. Ayrıca dördüncü bölümdeki örnek sistemler için eğitilen Uzun Kısa Süreli Bellek ağı beşinci bölümdeki Uzun Kısa Süreli Bellek Tabanlı Uyarlamalı Kontrolör içerisinde kullanılmış ve örnek referans sinyalleri takip etmesi sağlanmıştır. Bu kısımda ağ yapısının zincir kuralına göre çıkarımları yapılarak uyarlamalı ve Uzun Kısa Süreli Bellek tabanlı bir PID kontrolör tasarlanmış ve örnek sistemler üzerinde test edilmiştir. Çalışmanın son kısmında ise bir değerlendirme yapılarak "*elimizde herhangi bir matematiksel model olmadan sistem verilerinden derin öğrenme algoritmaları yardımıyla bir model oluşturulabilir mi?*" ve "*derin öğrenme algoritmaları ile yeni bir kontrol yöntemi geliştirilebilir mi?*" soruları cevaplandırılmaya çalışılmıştır.

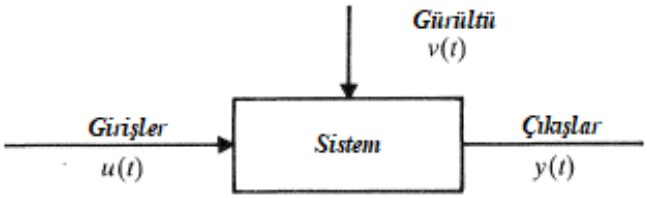
Çalışmanın genelinde amaçlanan asıl hedef "elimizde matematiksel modeli bulunmayan bir sistemin referans sinyal takibini hata değeri minimizasyonu ile kontrol edebilmek"tir. Ayrıca çalışmada uyarlamalı kontrol ve sistem tanımlama

uygulamalarına derin öğrenme ile yeni bir bakış açısı getirilmeye çalışılmıştır. Derin öğrenmenin yaygınlaştığı günümüzde, kontrol uygulamalarının da derin öğrenme alana taşınması hedeflenmiştir. Dökümanın ilerleyen kısımlarında sistem tanımlama, UKSB ve uyarlamalı kontrol konuları da genel olarak ele alınacaktır.

2. SİSTEM TANIMLAMA

2.1 Sistem Tanımlama

Sistem tanımlama, matematiksel modelini tam olarak çıkaramadığımız sistemlerde deneysel verileri kullanarak sistemi modelleme işlemidir. Bir sistem, giriş ve gürültülere maruz kalarak bunların sonucunda bir çıkış üretmektedir. Sistemler üzerindeki girişler kontrol edilebilir ancak gürültüler kontrol edilemez. [5] Burada sistem üzerinde kontrol hesaplamaları gerçekleştirebilmemiz için sistemin bir modeli olması gerekmektedir. Yani içerisini doldurmaya çalıştığımız alan Şekil 2.1'de gösterilen sistem kısmıdır. Bu kısmın içerisine yazacağımız ve sistemi ifade eden matematiksel ifade gerçek sistemi temsil etmektedir.



Şekil 2.1 : Temel sistem diyagramı.

Sistem kısmı içerisindeki model gerçek sisteme ne kadar benzer tasarlanırsa ardından oluşturulacak kontrol algoritması da gerçek sisteme uygulandığında o kadar iyi sonuçlar verir. Matematiksel olarak bir sistem üç şekilde ifade edilebilmektedir [5]:

1. *Mental, sezgisel yada sözsel:* Genel olarak araç kullanırken kullandığımız model tipidir. Sözsel ifadeler ile kontrol oluşturulur. Örneğin "lastiklerin dönmesi aracın dönmesine neden olur" gibi sözsel ifadeler kullanarak kontrol işlemi gerçekleştirilir.

2. *Grafikler veya tablolar:* Servo sistemlerin bode diyagramları bu tip sistemlere en iyi örnektir. Sistemin girişlerine karşılık gelen çıkış tablosu üzerinden sistem modellemesi sağlanmaktadır. Ancak tabloda olmayan ara değerleri tam olarak desteklemediğinden yakınsama içeren bir modelleme tekniğidir.

3. *Matematiksel modeller:* En sağlıklı modelleme tekniğidir. Bir sistemin girişleri ve çıkışları arasındaki bağlantılar matematiksel olarak gösterilmektedir. Dinamik sistemlerin matematiksel modelleri diferansiyel denklemler ile ifade edilirken statik sistemlerin modelleri türev ve integral içermez yani oransal veya sabittir. [29] Model, matematiksel ifadeler sayesinde analiz edilebilir, filtreler tasarlanabilir ve kontrol edilebilir. Ancak sistemler karmaşıklaştıkça matematiksel modellerini çıkarmak da aynı oranda zorlaşacaktır.

2.1.1 Matematiksel model ve sistem tanımlama

Temel olarak matematiksel bir modelin oluşturulması ve bu modelin kullanılmasının sağlıklı bir sonuç olacağı açıktır. Ancak her durumda matematiksel model oluşturamayacağımız ya da tam yakınsaklık sağlayamayacağımız unutulmamalıdır. Bir sistemde matematiksel model iki şekilde oluşturulabilmektedir [5]:

1. *Matematiksel Model:* Analitik bir yaklaşımdır. Doğa yasaları kullanılarak dinamik bir sistemin davranışları açıklanmaya çalışılır. Ancak pek çok doğa yasası yakınsama prensibi ile kullanılabileceğinden sistemde yer alan bazı özellikleri modele dahil edemeyebileceğimiz unutulmamalıdır.

2. *Sistem Tanımlama:* Deneysel bir metoddur. Fiziksel sistemden deneysel olarak giriş ve bu girişe karşılık gelen çıkış değerleri toplanır. Bir değerler bir veri paketi haline getirilerek belirli yöntemler ile sistem modeli çıkartılır. Son zamanlarda bu metod, makine öğrenmesi ile iç içe kullanılmaktadır.

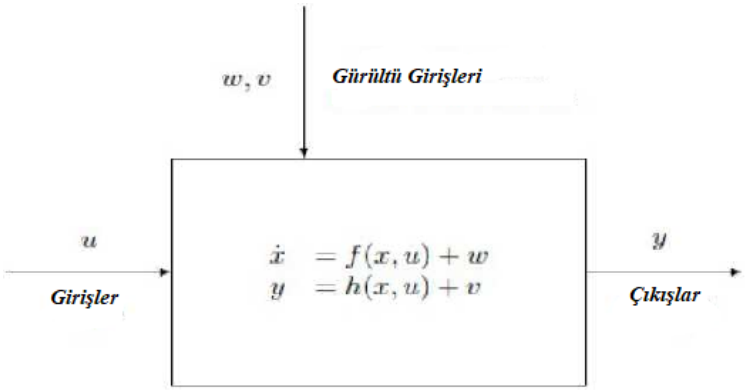
2.2 Sistem Tanımlama Tipleri

Üç farklı sistem tanımlama tipinden bahsedilebilir. Bunlar *beyaz kutu sistem tanımlama*, *gri kutu sistem tanımlama* ve *siyah kutu sistem tanımlama* tipleridir. [30]

2.2.1 Beyaz kutu sistem tanımlama

Beyaz kutu sistem tanımlama, sistemin tüm matematiksel ifadelerini bilmeye dayanan bir sistem tanımlamadır. Aslında direkt olarak matematiksel model çıkarımı olarak düşünülebilir. Bir model için Newton kanunları, elektrik yasaları gibi kurallar

kullanılarak matematiksel model çıkarılır ve davranışları incelenir. Şekil 2.2'de beyaz kutu sistem tanımlama yapısı gösterilmiştir.



Şekil 2.2 : Beyaz kutu sistem tanımlama. [1]

2.2.2 Gri kutu sistem tanımlama

Gri kutu sistemlerde, sistem içerisinde olup bitenler tam olarak bilinmemektedir. Sistem için bazı matematiksel modeller çıkarılmış ancak çıkarılamayan kısımlar ise veriler kullanılarak tanımlanmıştır. Yani bu tip sistem tanımlama uygulamalarında sistem için kısmi bir sistem tanımlama uygulaması kullanılmaktadır. [31]

2.2.3 Siyah kutu sistem tanımlama

Siyah kutu sistem tanımlamada, sistemin önceden tanımlanmış bir matematiksel modeli yoktur ve model denklemler kullanılarak değil de sistem için tanımlanmış giriş ve çıkış verileri kullanılarak elde edilir. Şekil 2.3'de siyah kutu sistem tanımlama ifade edilmiştir.

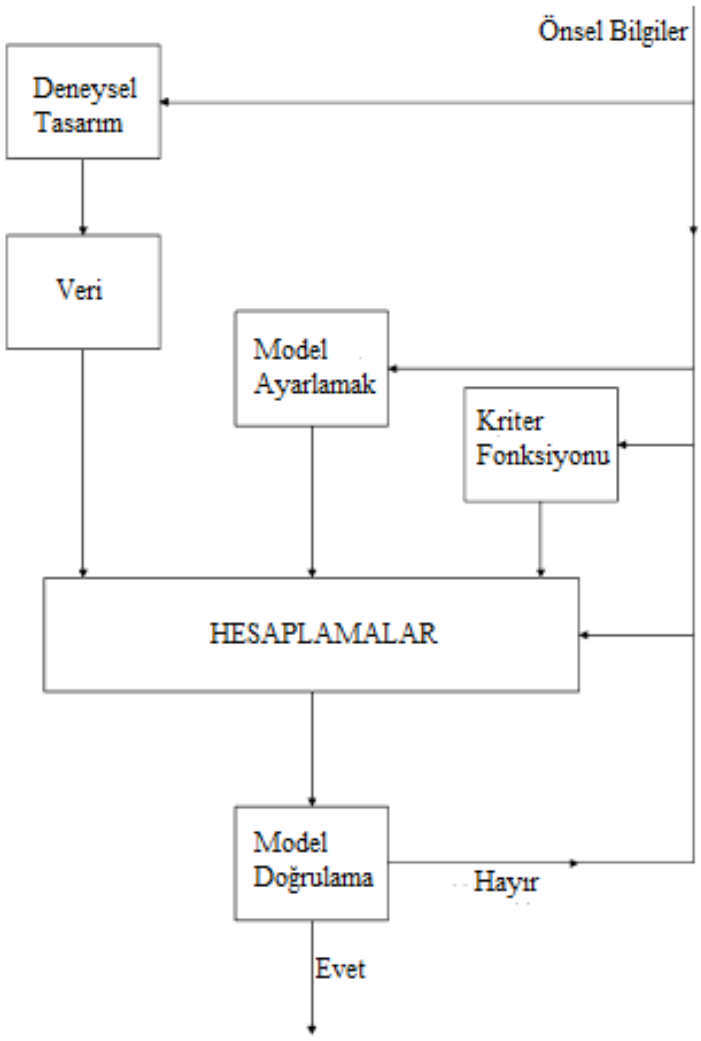
Şekil 2.3 : Siyah kutu sistem tanımlama. [2]

Çalışma içerisinde tercih edilen yöntem *siyah kutu sistem tanımlama* yöntemidir. Bu yöntem geleneksel yöntemlerin dışında, derin öğrenme algoritmaları kullanılarak gerçekleştirilmiştir. Bir sisteme ait giriş ve çıkış verileri derin öğrenme algoritmasına verilerek sistem için bir model oluşturulacaktır. Bu bağlamda UKSB algoritmasına başvurulmuştur.

2.3 Sistem Tanımlama Genel Akışı

Matematiksel modelin, gerçek sistemi tanımlayan yaklaşıksal bir ifade olduğundan bahsedilmişti. Pratikte bir sistemin karmaşıklığı, sistemi tanımamızda ve önsel bilgi edinmemizde bir sınırdır. Ancak sistem ve yeterli miktarda veri mevcutsa bu durumda tam anlamıyla kesin bir ifade bilinmesi gerekli değildir çünkü bu durumda model, kullanılmak için çok karmaşık bir yapıda olacaktır. [3]

Şekil 2.4'de sistem tanımlama genel akışı hakkında bilgi verilmiştir. Sistem tanımlamada, sistem hakkında önsel bilgiler ve veriler önemli rol oynar. [3] Bu iki değer birbirlerinden bağımsız değildir. Pek çok uygulamada veriler bu önsel bilgileri kullanarak toplanır. Önsel bilgiler kimi zaman veri setine ek olarak uygulanırken kimi zaman veri seti içerisinde yer alabilmektedir.

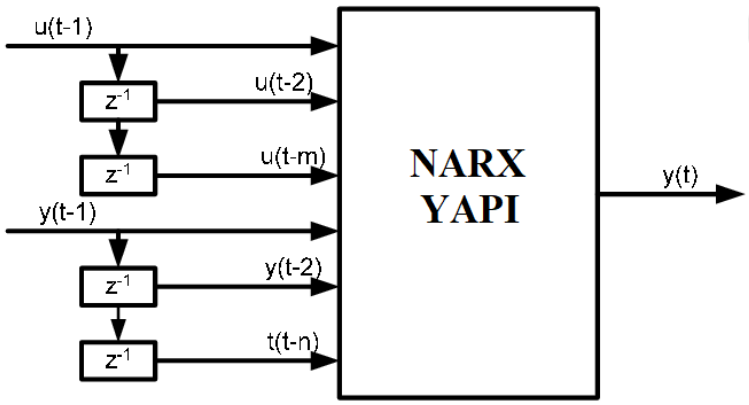


Şekil 2.4 : Sistem tanımlama genel akışı. [3]

Burada model çıktısı ve gözlemlenen veriler arasındaki fark, başarı kriteri olarak sayılır ve modelleme bu kriter üzerinden gerçekleştirilir. Sistemin gerçek sisteme ne kadar benzer sonuçlar ürettiği ise *"Model Doğrulama"* alanında kontrol edilir ve gerçek sistem ile model arasındeki fark belirlediğimiz bir sınırın altında ise sistem başarılı modellenmiş sayılır. Sınırın üstünde ise bu durumda modelleme tekrar yapılmalıdır. Modellemenin başarısız olmasının en büyük nedeni sistemi tanımlayacak uygun verilerin elde edilememiş olması veya eksik veri içermesinden kaynaklanmaktadır. [32]

2.4 Sistem Tanımlamada NARX Modeli

Bir sistem NARX model varsayımı ile tanımlanırken sistemin gelecek çıkışları, sistemin geçmiş çıkışları ve şimdiki girişleri ile geçmişteki girişlerinin bir fonksiyonu olarak yazılır. [33] Şekil 2.5'te NARX sistem için bir üretici örneği verilmiştir. NARX sistemlerin popülerleşmesinin en büyük nedenlerinden biri makine öğrenmesi yapılarına çok uygun olması ve veri yapısı ile direkt olarak uyum sağlamasıdır.



Şekil 2.5 : NARX işaret akış diyagramı. [4]

NARX modellerde bir sistemin t+1 anındaki çıkışı aşağıdaki şekilde modellenebilir. Bu model içerisindeki geçmiş girişler ve çıkışlar bir makine öğrenmesi algoritmasının girişleri olabilir. Çalışmanın ilerleyen kısımlarında bu yapı kullanılmıştır. Örnek bir NARX modelinin matematiksel ifadesi aşağıda verilmiştir:

$$y(t+1) = F(u(t), u(t-1), u(t-2), ..., y(t), y(t-1), y(t-2), ...) \qquad (2.1)$$

NARX modeller kullanılarak sistemin geçmiş değerlerinden gelecek değerleri elde edilebilmektedir. Bu yapısından ötürü NARX modeller genellikle makine öğrenmesi algoritmaları ile gerçekleştirilmeye oldukça uygundur.

2.5 Sistem Tanımlamada Giriş İşaretleri

Sistem tanımlamada giriş sinyali seçimi, sinyal tanımlama sonucunu direkt olarak değiştirebilen önemli bir etkiye sahiptir. Sisteme verilen giriş sinyalleri sonucu toplanan çıkışlar veri setimizi oluşturacaktır ve bu veri setinin sistemi en iyi şekilde temsil etmesi gerekmektedir. Sonuçta oluşacak veriler sistemin yüksek ve düşük frekanslı karakterlerini ortaya çıkarmalı ve sistemi tanımlayabilmelidir. [5] Aksi halde sistem tam anlamıyla tanımlanamaz ve oluşacak model kullanılamaz.

Genellikle sistemleri örneklenmiş veriler üzerinden tanımlarız. Bu durumda giriş ve çıkış verileri ayrık zamanda tanımlanır. Pek çok senaryoda da sistemi ayrık zamanda tanımlarız. Günümüzde kontrol sistemlerinin dijitalleşmesi de bu ayrık yapı için ön ayak olmuştur. Bir sisteme tanımlamada pek çok giriş sinyali kullanılabilir ancak aşağıdaki giriş sinyali tipleri sıklıkla kullanılan sinyal türleridir: [5]

- Basamak fonksiyonu
- Sözde rassal sayı üreteci(Pseudo Random Binary Sequence)
- Otoregresif entegre hareketli ortalama süreci(Autoregressive Integrated Moving Average Process)
- Sinüslerin toplamı

Yukarıdaki bu giriş işaretlerinden çok farklı işaret tipleri de vardır. Ancak günümüzde kullanılan ve değişik problemlerde iyi sonuçlar ürettiklerinden dolayı genellikle bu tipler tercih edilmektedir.

2.5.1 Basamak fonksiyonu

Basamak fonksiyonu sıklıkla kullanılan bir giriş işaretidir. İşareti içerisinde genlik değeri u_0 ayarlanarak istenilen işaret üretilebilir. Özellikle yüksek sinyal-gürültü oranı içeren sistemlerde sistem dinamiği hakkında değerli bilgiler üretir. Yükselme zamanı,

aşım ve statik kazanç basamak yanıtı ile direkt olarak etkilidir. [5] Aşağıdaki formül ile temel bir basamak fonksiyonu üretilebilir:

$$u(t) = \begin{cases} 0 & \text{eğer } t < 0 \\ u_0 & \text{eğer } t \geq 1 \end{cases} \tag{2.2}$$

2.5.2 Sözde rassal sayı üreteci(PRBS)

Sözde rassal sayı üreteci, iki seviye arasında bir sinyali kaydırarak oluşturulan giriş sinyalleridir. Sıklıkla tercih edilen bir giriş sinyalidir ve kayan yazmaçlarla sonlu durum makinesi oluşturarak gerçeklenebilir. Genellikle periyod deneysel verilerin sayısına eşit veya daha fazla seçilir. [5] Oluşturmak için bir saat sinyali kullanılır. Sinyal belirli bir değere geldiğinde bir seviyeden diğerine geçiş sağlanır ve sürekli olarak iki sinyal arasında bir geçiş vardır. Bu sinyal geçişleri için bir polinom fonksiyonu kullanılır ve bu fonksiyonun aşağıdaki gibi çeşitleri mevcuttur: [34]

- *PRBS7:* $x^7 + x^6 + 1$
- *PRBS9:* $x^9 + x^5 + 1$
- *PRBS11:* $x^{11} + x^9 + 1$
- *PRBS15:* $x^{15} + x^{14} + 1$
- *PRBS20:* $x^{20} + x^3 + 1$
- *PRBS23:* $x^{23} + x^{18} + 1$
- *PRBS31:* $x^{31} + x^{28} + 1$

Şekil 2.6'da MATLAB'da PRBS7 ile üretilmiş bir sinyal verilmiştir. Bu sinyal sistem girişine uygulanarak sistem içerisinde bilgi çıkarımı gerçekleştirilebilir. Ancak sözde rassal sayı üretecinde, üretecin çıkışı ancak iki seviye arasında değer alabilir. Bu durum sistem içerisindeki bazı özellikleri tanımamıza engel olabilir. Bu tip bir problemle başa çıkmak için PRBS sinyaline eklentiler yapılarak farklı seviyeler arasında da değişebilmesi sağlanmıştır. Bu özellik sayesinde farklı seviyeler sistem için farklı özellikleri açığa çıkarabilir.

İki farklı seviye arasında değişen sinyal birden fazla seviye arasında değişen hale getirilebilir. Çalışmada da bu tip bir giriş sinyali tercih edilmiştir. Şekil 2.7'de

Şekil 2.6 : PRBS7 ile üretilmiş sinyal.

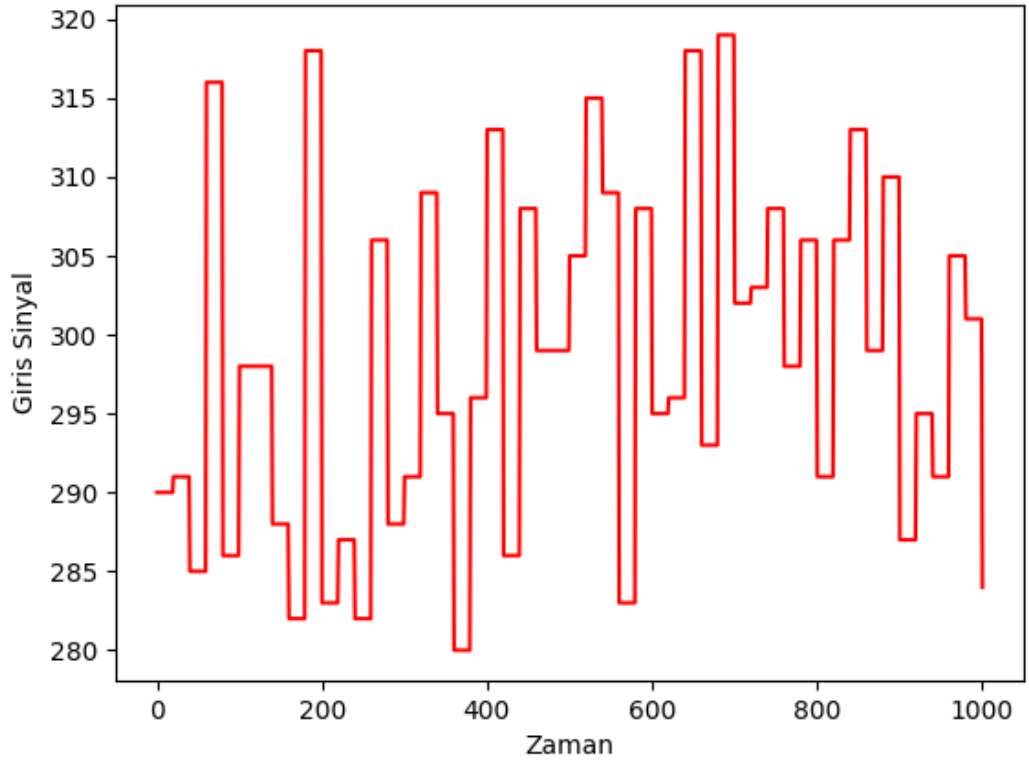
farklı seviyeler arasında değişen PRBS giriş sinyali oluşturmamızı sağlayacak sözde kodlar verilmiştir. Pek çok yazılım dilinin rastgele sayı oluşturma desteği ve random kütüphaneleri bu tip işaret üretiminde oldukça yardımcı olacaktır.

```
1  > BAŞLA
2  > değişim_zamanı belirlenir
3  > seviye değeri belirle
4  > döngü (t<simülasyon_zamanı)
5  >        giriş_işareti[t] =  seviye
6  >        eğer(degisim_zamanı) (Değişim zamanı geldiğinde)
7  >                rastgele_sayı = -n ve +n arasında rastgele sayı belirle
8  >                seviye = seviye + rastgele_sayı
9  > BİTİR
```

Şekil 2.7 : Sözde rassal işaret üreteci sözde kodları.

Bu tip bir yapının Python programlama dili kullanılarak gerçekleştirilmesi sonucunda Şekil 2.8'deki işarete benzer bir işaret meydana gelir. İşaretin PRBS sistemlerine benzediği ancak birden fazla seviye içerdiği gözlemlenebilir. Bu yapı farklı seviyeler içerdiğinden sistemin farklı davranışlarını da modelleyebilmekte ve sistem hakkında daha sağlıklı bilgiler üretebilmektedir.

Bu tip giriş sinyali üretiminde dikkat edilmesi gerek önemli bir konu seviye sayısı seçimidir. Az seviye seçmek sistemi tanımamızı zorlaştıracağı gibi fazla seviye seçmek de karmaşık bir yapı oluşmasına neden olacaktır. Bu yüzden seviye sayısı ve değerleri dikkat edilmesi gereken özelliklerdir. Çalışma kapsamında bu değerler deneme yanılma yöntemleri ile seçilmiştir. Kullanılan test sistemlerinin çalışma aralıkları göz önüne alınarak çalışma aralıklarını aşmayacak şekilde ve çok sık değişim gerçekleşmeyecek şekilde değerler seçilmiştir.



Şekil 2.8 : Çoklu seviye sözde rassal işaret üreteci işareti

2.5.3 Otoregresif entegre hareketli ortalama süreci

Sözde rassal sayı üretmenin bir yolu da otoregresif entegre hareketli ortalama kullanmaktadır. Aşağıdaki formülde *e(t)* beyaz gürültü benzeri bir sözde rassal sayı üreteci olsun; [5]

$$u(t) = \frac{1}{N}\sum_{t=1}^{N} e(t)e(t+\tau) \to 0 \quad ve \quad N \to \infty \quad (\tau \neq 0) \tag{2.3}$$

Yukarıdaki bu formül üzerinden *e(t)*, *u(t)* cinsinden aşağıdaki gibi bir eşitlik ile ifade edilebilir; [5]

$$u(t) + c_1 u(t-1) + \ldots + c_m u(t-m) = e(t) + d_1 e(t-1) + \ldots + d_m e(t-m) \tag{2.4}$$

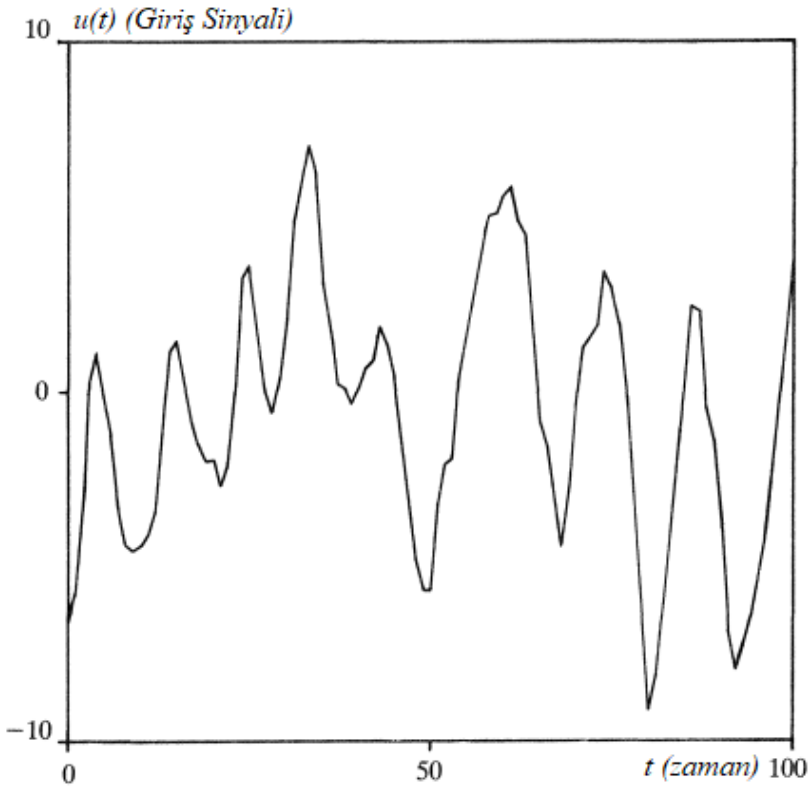
İşte yukarıdaki *u(t)*'yi veren bu ifade ARMA süreci olarak adlandırılmaktadır. Sistem tanımlamada veri çıkartmak için sisteme bu sinyal uygulanabilir. İfade içerisinde $c_i = 0$ olarak alınırsa süreç ***hareketli ortalama*** **(MA)** olarak anılır. Eğer ifade içerisinde $d_i = 0$ olarak alınırsa süreç ***otoregresif entegre*** **(AR)** olarak adlandırılacaktır. ARMA süreci genel bir ifade cinsinden aşağıdaki şekilde yazılabilir. [5]

$$u(t) = \frac{D(q^{-1})}{C(q^{-1})} e(t) \tag{2.5}$$

İfade içerisindeki q^{-1} geriye doğru zamanda kaydırma operatörüdür. Yani $q^{-1}e(t) = e(t-1)$ olarak düşünülmelidir. Bu kısımda $D(q^{-1})$ ve $C(q^{-1})$ ifadeleri aşağıdaki biçimde ifadelerdir:

$$C(q^{-1}) = 1 + c_1 q^{-1} + ... + c_m q^{-m} D(q^{-1}) = 1 + d_1 q^{-1} + ... + d_m q^{-m} \qquad (2.6)$$

Sistem tanımlamada *u(t)* işareti sisteme uygulanır ancak *u(t)* işaretini oluşturan c_i ve d_i paremetreleri doğru ayarlanmalıdır. *C(z)* ve *D(z)*'nin tüm sıfırları birim çemberin dışında seçilmelidir ve *C(z)*, *u(t)*'nin stabil olduğunu garanti etmelidir. Bu parametreler seçilirken de sistemin yüksek ve düşük frekans aralıklarını ifade eden verileri çıkarabilecek şekilde seçilmesine özen göstermek gerekmektedir. Aksi halde sistemi gerçekten tanımlayan veriler çıkarılamaz. Şekil 2.9'da $C(q^{-1}) = 1 - 1.5q^{-1} + 0.7q^{-2}$ ve $D(q^{-1}) = 1$ alınarak bir giriş işareti oluşturulmuştur. Bu işaret sistem girişine uygulanarak sistem giriş çıkış verileri toplanabilir.



Şekil 2.9 : $C(q^{-1}) = 1 - 1.5q^{-1} + 0.7q^{-2}$ ve $D(q^{-1}) = 1$ ARMA işaret. [5]

2.5.4 Sinüslerin toplamı

Sinüslerin toplamı giriş işareti mantığında, işaret oluşturulurken a_j genlikli m adet sinüsün işaretinin toplamı kullanılmaktadır. Aşağıdaki formül kullanılarak temel bir

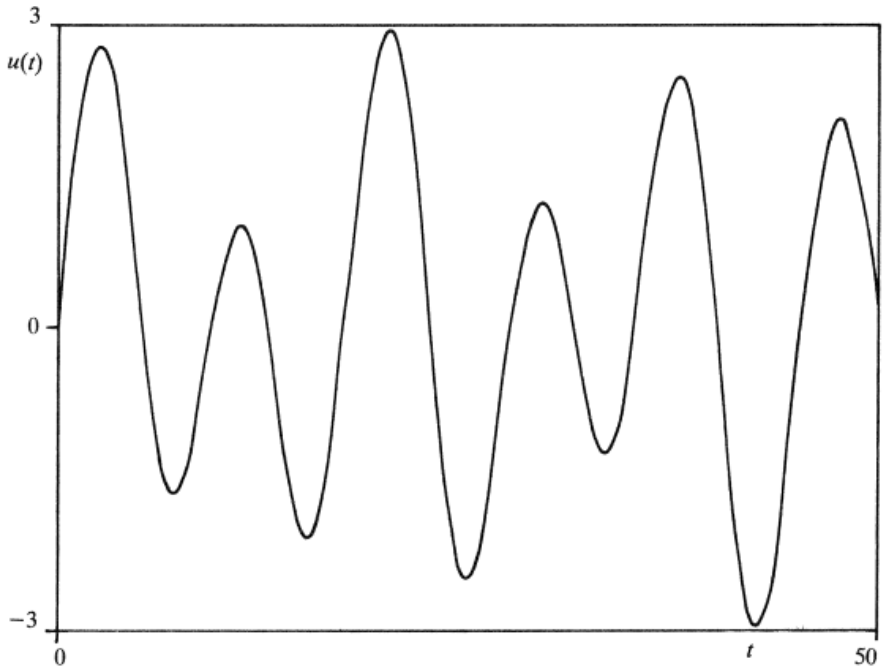
sinüslerin toplamı giriş işareti üretilebilir [5]:

$$u(t) = \sum_{j=0}^{m} a_j sin(\omega_j t + \phi_j) \tag{2.7}$$

Üretilen bu işaret için açısal frekanslar seçilirken

$$0 \leq \omega_1 < \omega_2 < ... < \omega_m \leq \pi \tag{2.8}$$

olarak seçilmesi gerekmektedir. Burada a_j sinüs işareti genliği, ω_i frekans ve ϕ_i faz olarak verilmiştir. Şekil 2.10'da, sin(0.4t) + 2.sin(0.7t) işareti için sinüslerin toplamı girişine bir örnek verilmiştir. Sinüslerin frekans ve genliklerinin seçimi bu kısımda önemli bir yer tutar. Seçilen işaretlerin frekans ve genlikleri, sistemin yüksek frekans ve alçak frekans özelliklerini modelleyebilecek şekilde seçilirse daha verimli bir model çıkarımı sağlanabilir. Dijital sistemlerde ise bu sinüs işaretinin sisteme uygun bir örnekleme frekansında örneklenerek verilmesi gerektiği unutulmamalıdır.



Şekil 2.10 : sin(0.4t) + 2.sin(0.7t) işaretlerinin toplamı. [5]

Çalışmanın sistem tanımlama kapsamında kullanılacak giriş işareti farklı seviyeleri de içeren PRBS sinyalidir. Özellikle üretimi kolay ve sonucunda oluşan veri seti daha verimli olduğundan bu sinyalin kullanılmasına karar verilmiştir. Ayrıca benzer çalışmalar incelendiğinde bu tip bir işaretin sıklıkla kullanıldığı gözlemlenmiştir. Benzer çalışmalarda giriş işaretleri seçilirken deneysel yöntemlerle tercih edildiğinden uygulama kapsamında da bu tip bir işaret deneysel yöntemlerle tercih edilmiştir.

2.6 Makine Öğrenmesi İle Sistem Tanımlama

Sistem tanımlama alanında günümüze kadar pek çok yöntem geliştirilmiş ve kullanılmıştır. Ancak makine öğrenmesi tekniklerinin son dönemlerde gelişmesi sayesinde sistem tanımlama makine öğrenmesi ile de gerçeklenebilir hale gelmiştir. Makine öğrenmesi literatüründe bu işlem ***Zaman Serisi Tahmini (Time Series Prediction)*** olarak isimlendirilir. Bu tip yapılarda amaç sistemin içinde bulunduğu anından sonraki değerlerini tahmin edebilmektir yani sistemin gelecek davranışlarını çıkarabilmektir. Makine öğrenmesi kullanılması geniş kütüphane destekleri ile rahat ve sistemi iyi modellemek için yeterli gibi görünse de aşağıdaki iki konuya dikkat etmek gerekmektedir. [25]

1. Pek çok doğrusal olmayan sistem tanımlama uygulaması yapay ağlar kullanır ancak bu ağlar çok fazla karmaşık seçilirse eğitim süresi aşırı uzayacaktır. Çok fazla sinir ağını eğitmek gereğinden fazla süre kaybına neden olur ve bu tip yapılar gerçek dünya uygulamalarında gerçekleştirilemez hale dönüşür.

2. Çok kompleks yapılar kullanıldığında, kullandığımız yapı sistemi öğrenmekten çok ezberlemeye yönelir. Bu durumda eğitim verisi içerisindeki gürültüler de ezberlenir ve sistem doğru modellenemez.

Buradan çıkarılacak ders sistemi modelleyebilecek en optimum ağ yapısını oluşturmak gerektiğidir. Bu yapıyı oluşturmak için kesin kurallar yoktur. Genellikle deneme yanılma türü yöntemler kullanılmaktadır. Uygulama kapsamında en optimum yapı deneme yanılma yöntemi ile seçilmiştir. Bu seçim yapılırken kullanılan test sistemleri de göz önünde bulundurmuştur.

3. UZUN KISA SÜRELİ BELLEK - UKSB

3.1 Yapay Öğrenme

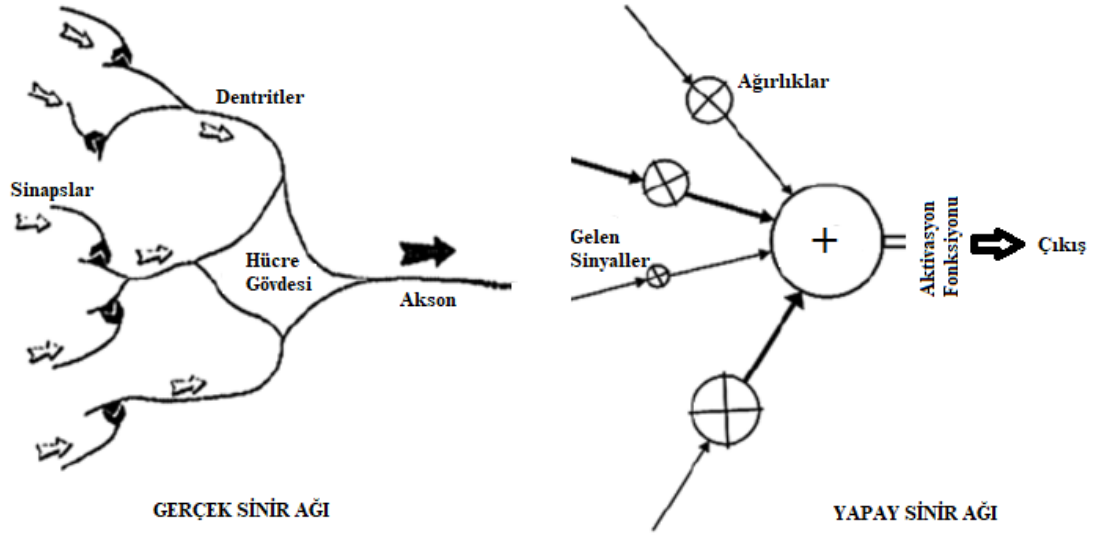
Bilgisayarlar üzerinde bir problemi çözüme kavuşturabilmek için bu probleme uygun bir algoritma yani çözüm yöntemi oluşturmamız gerekir. Bu sayede bilgisayar ortamında gerçek bir problemi çözebiliriz. Algoritma, kendisine verilen girişlere karşı çıkış üreten bir yapıdır. Ancak her zaman bir problem için algoritma üretemeyiz. Bu tip durumlarda elimizde yeterli miktarda giriş-çıkış verisi varsa bu veriyi kullanarak bilgisayarın algoritma üretmesi yani öğrenmesini bekleriz. Bilgisayar, yeterli miktardaki verileri saklar ve işler, ardından problem ile ilgili bir algoritma üretir. İşte bu yapı ***yapay öğrenme*** olarak adlandırılır. [35]

Makine öğrenmesi yada yapay öğrenme temelinde istatistik ve olasılık konularını kullanır ve bu alanları algoritma becerisi ile birleştirir. [35] Yeterli miktarda veriyi kullanarak bir sonuç çıkarmayı amaçlar. Burada veri sayısı az ise çıkarım eksik olacağı gibi aşırı miktarlardaki veriler ile de yanlı çıkarımlar gerçekleştirilir. En yaygın kullanılan yapay öğrenme metodu yapay sinir ağlarıdır.

3.1.1 Yapay sinir ağları

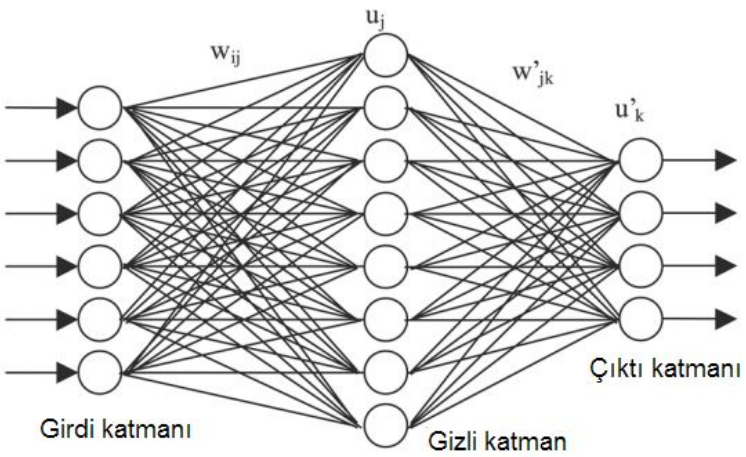
Yapay sinir ağları günümüzde pek çok alanda kullanılan ve organik sinir hücrelerinden yola çıkarak oluşturulmuş yapılardır. Şekil 3.1'de yapay sinir hücreleri ve gerçek sinir hücrelerinin gösterimleri verilmiştir. Yapay sinir hücreleri, içerislerindeki ağırlık değerlerini güncelleyerek girişler ve çıkışlar arasında ilişki kurmaya yarar.

İnsan beyninin çalışma mantığını anlayabilmek için çalışmalar yapılmıştır ve bilgisayar ortamında modellenmeye çalışılmıştır. 1943 yılında Warren McCulloch ve Walter Pitts tarafından ilk yapay sinir ağı modeli ortaya konmuştur. [36] Araştırmacılar elektrik devreleri ile insan beynindeki ağlara benzer yapılar ortaya çıkarmış ve yapay öğrenmenin temelini atmışlardır. Bu yapı bir sinir hücresinin öğrenme sırasındaki adımlarını modelleyebilmektedir. Oluşturulan sinir modeli ile ilk başta kolay problemler bile çözülememiş ve bu olay yapay öğrenme alanının yeterli ilgiyi



Şekil 3.1 : Gerçek ve yapay sinir hücresi. [6]

görememesine neden olmuştur. Ancak ilerleyen dönemlerde yapay sinir ağları bir araya getirilerek çok karmaşık problemler bile çözülebilmiş ve pek çok problemin çözümünde yapay sinir ağlarına başvurulmuştur. 1957 yılında Frank Rosentblatt, ***perseptron*** adı verilen ve günümüzdeki yapay sinir ağına oldukça benzer yapıyı ortaya atarak gelişimine büyük katkılar sağlamıştır. Şekil 3.2'de günümüzde de en sık kullanılan yapay sinir hücrelerinin birleşmesi ile oluşan bir yapay sinir ağı gösterilmiştir. Bu tip ağlarda gizli katman sayılarının arttırılabileceği unutulmamalıdır. Ancak gizli katman sayılarının arttırılması işlem yükünü de beraberinde getirir.



Şekil 3.2 : Yapay sinir ağı. [7]

3.1.2 Yapay öğrenme tipleri

Yapay öğrenmenin tipleri farklı şekilde sınıflandırılsa da temelinde yapay öğrenme üçe ayrılmaktadır. Bunlar:

- Öğreticili Öğrenme
- Öğreticisiz Öğrenme
- Pekiştirmeli Öğrenme

3.1.2.1 Öğreticili öğrenme

Girdi ve çıktılar arasındaki bağlantıyı öğrenebilmesi için sisteme hem girdi hem de çıktıların verildiği öğrenme tipidir. [37] Öğreticili öğrenmede sistem girişleri ve çıkışları bilir ve arasında bağlantı kurar. Öğreticili öğrenme iki tipte olabilir;

1. *Sınıflandırma:* Çıktıların kategoriler halinde olduğu eğitim tipidir. Sonuçlar kadın-erkek, hasta-sağlıklı, kedi-köpek-kuş gibi sınıflardan birinin olduğu öğrenme tipidir.
2. *Regresyon:* Çıktıların sayısal değerler olduğu eğitim tipidir. Genel olarak bu tip eğitimler eğri uydurma örnekleri gibi düşünülebilir.

3.1.2.2 Öğreticisiz öğrenme

Öğreticisiz öğrenme uygulamalarında sisteme sadece girdi verilir ve sistemin çıktıları bulması beklenir. Burada bir kümeleme söz konusudur. [37] Öğreticisiz öğrenme uygulamaları iki tipte olabilir;

1. *Kümeleme:* Farklı tipteki örneklerin ayrılarak aynı tipteki örneklerin aynı kümede toplanıldığı uygulamalardır. En sık kullanılan öğreticisiz öğrenme tipidir.
2. *Boyut Azaltma:* Veri kümesinin boyutu öğrenme performansı ve zamanını etkileyen bir kavramdır. Veri kümesi boyutu çok büyürse öğrenme zamanı artar. Bu durumu azaltmak için veri kümesi boyutu düşürülmelidir. Ancak burada önemli özellikler tutulurken önemi daha az özellikler atılmalıdır. Bu tip uygulamalar boyut azaltma uygulamaları olarak geçer.

3.1.2.3 Pekiştirmeli öğrenme

Pekiştirmeli öğrenme, en temel öğrenme yöntemlerinden ödül ve ceza sistemini kullanmaktadır. Sistem ödül sayısını maksimuma çıkartmaya çalışırken ceza sayısını minimuma indirmeye çalışır. Sistemler genellikle deneme yanılma yöntemi ile

öğrenmeye çalışırlar. Pekiştirmeli öğrenme, oyunlarda ve robotik alanında çok sık tercih edilen bir yöntemdir.

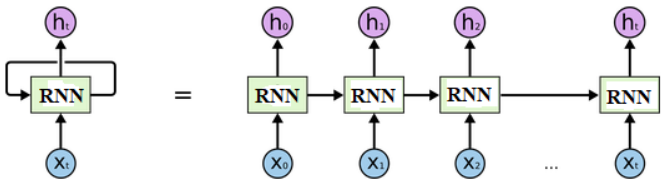
Çalışmada kullandığımız UKSB yapısı öğreticili öğrenme tipinde bir algoritmadır. Sistem tanımlaması yaparken UKSB yapısının içerisine hem girişler hem de çıkışlar verilir ve UKSB mimarisinin girişler ve çıkışlar arasında bir bağlantı kurmasını beklenir.

3.2 Uzun Kısa Süreli Bellek

Uzun kısa süreli belleklerin temelini ***Tekrarlı Yapay Sinir Ağları (RNN)*** yapısı oluşturur. Bu yapıların ileri beslemeli yapay sinir ağlarından farkları kendisine verilen veriler arasında sıra bilgisini önemli görmesi ve geçmiş verileri de saklayarak eğitim sırasında kullanmasıdır. Bu özelliklerinden ötürü özellikle sistem tanımlama ve zaman serisi tahmini uygulamalarında daha başarılı olurlar. Çalışmanın bu bölümünde UKSB yapısından önce RNN yapısının anlatılması daha yararlı olacaktır.

3.2.1 Tekrarlı yapay sinir ağları - RNN

İnsanların düşünce sisteminde öğrenilmiş bilgiler bir durum hakkında çıkarım yapılırken atılmaz, yani anlık bilgileri kullanmak yerine daha önceki bilgileri de anlık bilgilere ekleyerek çıkarımlar yapılır. İşte bu özellik RNN mimarisinin temelini oluşturur. [8] Klasik ileri beslemeli yapay sinir ağları mimarilerinde eski bilgiler kullanılmaz ve sadece o anda kullanılan bilgiler ile çıkarımlar yapılır. Bu durumda ağ, geçmiş veriler ile ilişki kuramaz. Geçmiş verileri kullanabilmek adına klasik nöron yapısına geri besleme mantığı getirilmiştir. Bu kısımda geri besleme yapısı kulağa korkutucu gelse de yapı Şekil 3.3'de açıldığında daha mantıklı bir hal alır. Burada bir zaman sinyali varmış gibi düşünülebilir ve sistem t anında iken sisteme t-1 anındaki verilerde verilir. Bu döngü sürekli olarak devam eder ve geçmiş veriler de sisteme dahil edilmiş olur.



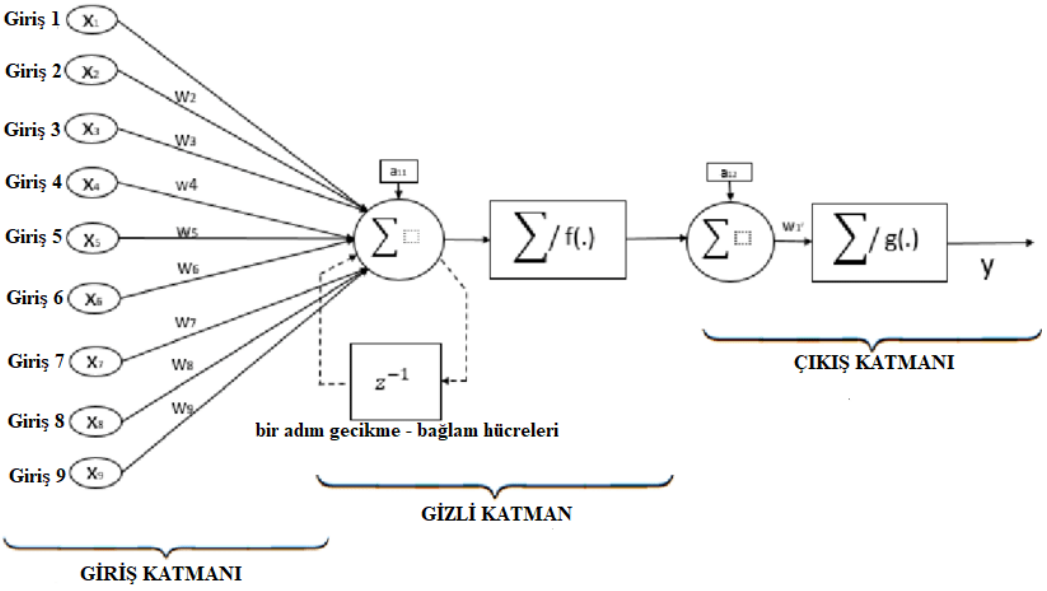
Şekil 3.3 : RNN geri besleme yapısı. [8]

RNN uygulamaları içerisinde genellikle üç tip ağ yapısı önerilmektedir. Bunlar:

- Elman Ağı
- Jordan Ağı
- Tam Bağlantılı Ağ

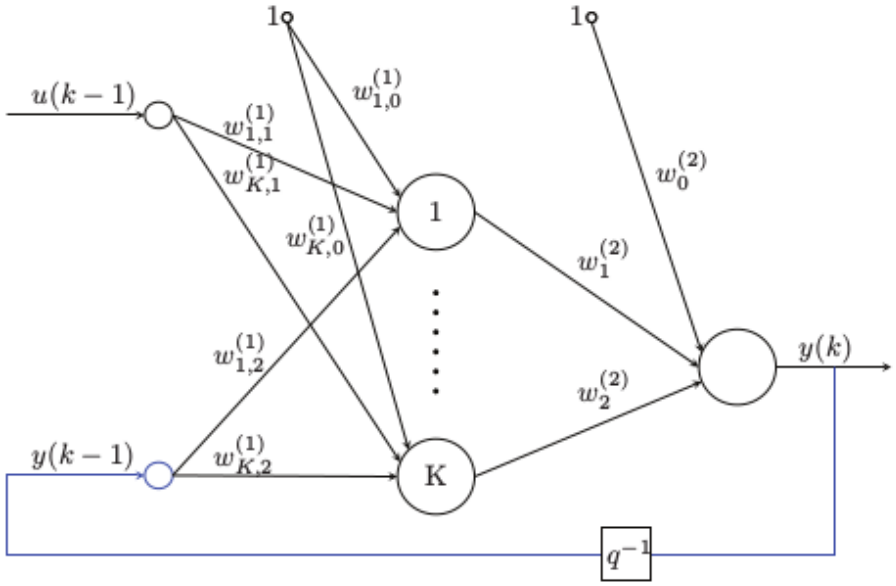
Bu yapılar en sık kullanılan RNN tipleridir. Birbirlerine göre avantajları ve dezavantajları mevcuttur. Bazılarının performansları yüksek ancak eğitim süreleri ve uygulanabilirlikleri zorken, bazılarının uygulanabilirlikleri kolay ancak performansları diğerlerine göre daha düşüktür.

Şekil 3.4'de ***Elman Ağı*** üç adet katman içermekte ve katmanlar arasında geriye dönük veri transferleri görülebilmektedir. En son katmanı ***bağlam hücreleri (context cell)*** olarak adlandırılır. Bu ağın özelliği standart geri yayılım algoritması ile eğitilebilmesidir. [38]



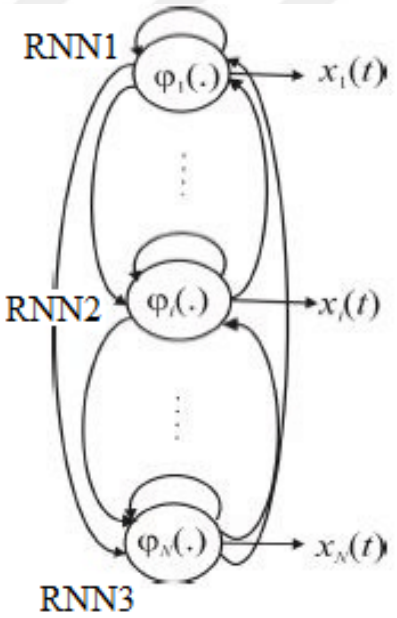
Şekil 3.4 : Elman ağı ve bağlam hücreleri. [9]

RNN uygulamalarında kullanılan bir diğer ağ ise ***Jordan Ağı***dır ve şekil 3.5'de gösterilmiştir. Jordan ağı, Elman ağına oldukça benzer bir yapıdadır ancak bağlam hücreleri gizli katman yerine çıkış katmanından beslenmektedir. [10] Jordan ağının bağlam hücreleri bazı kaynaklarda durum katmanı olarak da geçmektedir. Ayrıca başka düğüm olmadan kendilerini tekrarlayan yapılar da kullanılabilir. [39]



Şekil 3.5 : Jordan ağı ve bağlam hücreleri. [10]

Bunların dışında RNN uygulamalarında ***Tam Bağlantılı Ağ*** yapısı da kullanılabilir. Bu yapı standart geri yayılım algoritması ile eğitilemez ve Elman Ağına göre daha karmaşıktır. Şekil 3.6'de Tam Bağlantılı Ağ yapısı gösterilmiştir. [38]



Şekil 3.6 : Tam bağlantılı ağ yapısı örneği. [11]

3.2.2 Tekrarlı yapay sinir ağlarının zorlukları

Tekrarlı yapay sinir ağlarına güzel bir örnek olarak video işleme örneği verilebilir. Video verisi işlenirken bir önceki görüntü bilgisini de kullanmanın daha sağlıklı olacağı aşikardır. Makine önceki görüntüler yardımıyla şimdiki görüntüyü daha

sağlıklı tahmin eder. Klasik makine öğrenmesi algoritmalarının yapamayacağı ve tekrarlı ağlar için en sık verilen örneğe bir göz atalım. İnsan beyni *"Türkiyede yaşıyorum ve ...'yı çok iyi konuşurum."* cümlesindeki ... kısmını Türkçe olarak doldurabilir. Bunu yaparken cümle içerisindeki Türkiye verisinden yani geçmiş bir veriden yararlanır. Aynı mantıkla tekrarlı sinir ağları da bu veriyi kullanır ve klasik ağların dışında bu tip bir problem için çözüm üretebilir. Ancak tekrarlı sinir ağlarının önemli bir problemi vardır. Çıkarım yapmak istenilen problem zorlaştıkça daha fazla RNN hücresine ihtiyaç duyulur, bu durumda da ***Gradyan Kaybolma ve Patlama*** problemi ortaya çıkar ve çok yüksek ya da çok düşük güncelleme değerleri meydana gelmeye başlar. Çok yüksek güncelleme değerlerinde minimum noktayı ıskalama problemi meydana gelirken çok düşük güncelleme değerlerinde ise minimum noktaya varamama problemleri ile karşılaşırız. [38] Bunun dışında normal ağlara göre RNN eğitimleri daha zahmetlidir ve çok uzun sekanslarda tanh ve relu gibi aktivasyon fonksiyonları kullanılamaz. Peki bu durumları gidermek adına ne yapılabilir? İşte burada karşımıza UKSB mimarisi çıkar ve RNN ağları için tanımlanan problemler giderilebilir.

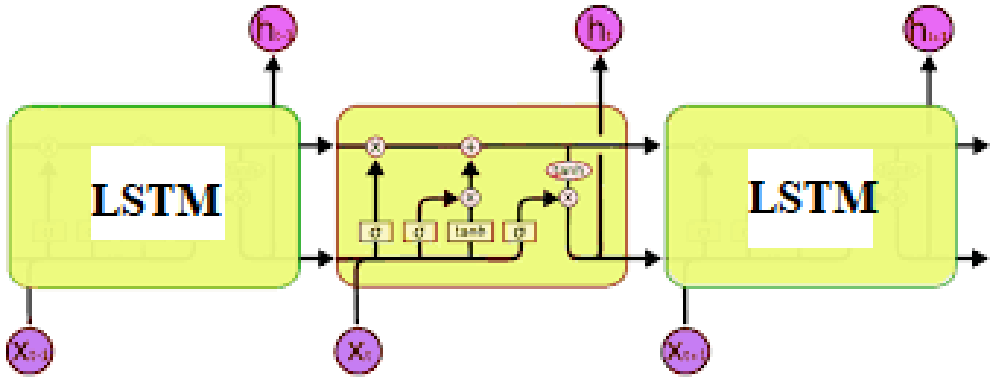
3.2.3 Gradyan kaybolma ve patlama problemi

Karmaşık bir problemi az sayıda RNN hücresini bir araya getirerek çözemeyiz. Bu durumda problemin kompleksliğine göre birden fazla hücreyi bir araya getirmek gerekir. Ancak RNN hücreleri bir araya getirilmeye başlandığında geri besleme sayıları artmaya başlar ve güncellemelerde sürekli olarak çok küçük sayılar veya büyük sayılar devreye girmeye başlar. Bu durumda güncelleme değerleri ya çok küçük olur ya da bir yerden sonra aşırı büyüyerek programlama dillerinin sınırlarını aşmaya başlar. Bu durumlarda ise ağırlıkları güncelleyemeyiz ve öğrenme durur.

Bu problem ile baş etmenin bir yolu ***kırpma*** mantığı kullanmaktır. Kırpma mantığında güncelleme değerleri bir eşiğin çok altında ya da çok üzerinde kalırsa güncelleme değeri olarak sabit bir değer belirlenip atanabilir. Ancak fark edileceği üzere bu yöntem çok sağlıklı bir yöntem olmayacaktır ve sürekli sabit değerler ile eğitim amaçlanmıştır. [40] Bu tip bir yöntemi denemek yerine problemden kökten kurtulmak adına Uzun Kısa Süreli Bellek yani UKSB ağları tercih edilebilir. UKSB mimarisi RNN ağlarının bu problemlerini ortadan kaldırmak için geliştirilmiş yapılardır.

3.2.4 Uzun kısa süreli bellek - UKSB

Uzun Kısa Süreli Bellek hücreleri ya da kısaca UKSB, RNN problemlerini gidermek ve daha verimli bir yapı kurmak için oluşturulmuş yapay sinir hücreleridir. Hochreiter ve Schmidhuber tarafından 1997 yılında ortaya atılmıştır. [41] Bir UKSB hücresi içerisinde birden fazla işlem gerçekleştirilir. UKSB hücreleri de RNN hücreleri gibi birbirlerine bağlanabilir. Bu sayede karmaşık problemlere çözüm bulunabilir. Şekil 3.7'da birbiri ardına bağlanmış UKSB hücreleri gösterilmiştir.



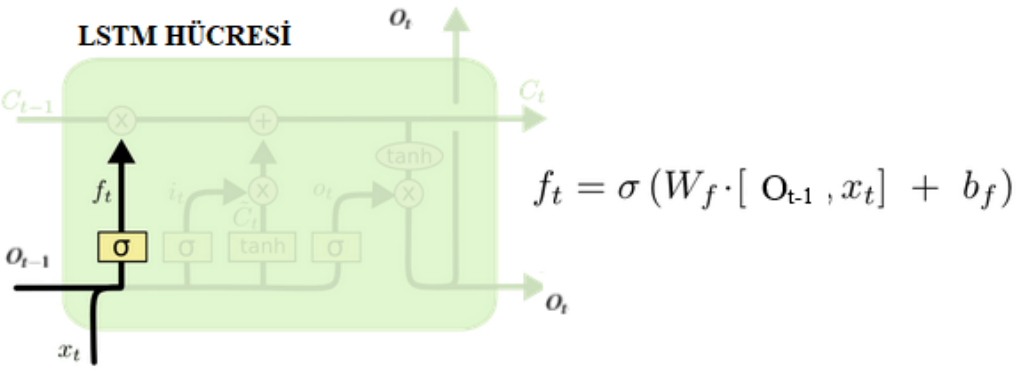
Şekil 3.7 : Birbirine bağlı UKSB hücreleri. [12]

Diğer sinir ağlarında olduğu gibi UKSB hücresinin de iki faz vardır. Bu fazlardan ilki ***İleri Faz*** olarak adlandırılırken ikincisi ise ağırlık değerlerini güncellemek için gerekli olan ***Geri Faz***dır. İleri faz aşamalarında UKSB için çıkışlar hesaplanır. Ardından ise ileri faz çıkışları kullanılarak hatalar hesaplanır ve bu hatalar geri faz kısmında ağırlık değerlerini güncellemek için kullanılır. Çalışmanın ilk adımında ileri faz incelenmiştir. Ardından geri faz ile ağırlıkların güncelleme değerlerinin nasıl hesaplanması gerektiği anlatılmıştır.

3.2.4.1 UKSB ileri fazı

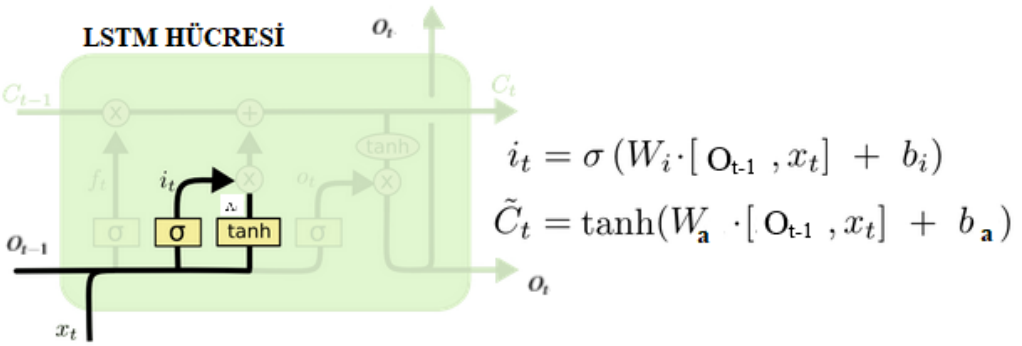
UKSB ileri fazı, giriş ve önsel verileri UKSB hücresi içerisinde işlemlere tabi tutarak çıkış üretmeye dayanmaktadır. Bu kısımda öncelikle kullanılması gereken girişler incelenmiştir. Giriş verisi olarak UKSB girişine gelen veri yani x_t ve bir önceki çıkışı ifade eden O_{t-1} birleştirilir. Bu birleşmeden aslında bir önceki verilerin de kullanılacağı anlaşılabilir. Yani veri sıralaması önem kazanır. Birleşmenin ardından oluşan veri ***Unutma Kapısı (Forget Gate)*** içerisindeki ***Unutma Kapısı Ağırlıkları*** ***(W_f)*** ile çarpılır ve sonucun üzerine bu kısma özgü ***Unutma Kapısı Sapma Değerleri***

(b_f) eklenir. En son adımda ise sigmoid fonksiyonundan geçirilerek unutma kapısı çıkışı elde edilmiş olur. Karmaşayı gidermek için Şekil 3.8'de bu hücre içerisindeki adımlar gösterilmiştir. İşlemlerin tamamlanmasının ardından tek bir UKSB hücresi için Unutma Kapısı alanı çıkışı elde edilmiş olur. [8] Bu çıkış ilerleyen adımlardaki çıkışlar ile birleştirilerek adım adım sonuca gidilecektir. [38]



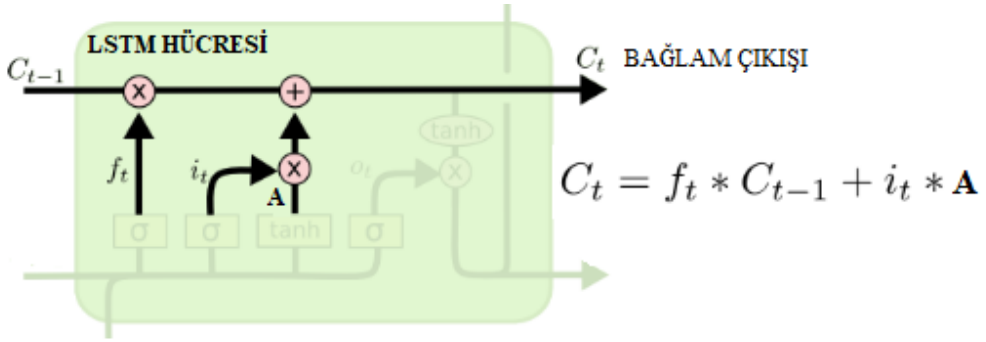
Şekil 3.8 : UKSB Unutma Kapısı hesaplaması. [8]

Unutma Kapısı çıkışı bulunduktan sonra ***Giriş Kapısı (Input Gate)*** ve ***Aktivasyon Kapısı (Activation Gate)*** çıkışları hesaplanır. Bu çıkış hesaplanırken Unutma Kapısı alanındaki gibi x_t ve O_{t-1} birleştirilir. Ardından Giriş Kapısı hesaplamaları için birleştirilen girişler, ***Giriş Kapısı Ağırlıkları*** *(W_i)* ile çarpılır ve bu değere ***Giriş Kapısı Sapma Değerleri*** *(b_i)* eklenir. Bu kısım sonucunda oluşan değer sigmoid fonksiyonuna tabi tutularak Giriş Kapısı çıkışları oluşur. Kısmın ikinci adımında ise birleştirilen x_t ve O_{t-1} değerleri önce ***Aktivasyon Kapısı Ağırlıkları*** *(W_a)* ile çarpılarak sonucun üzerine ***Aktivasyon Kapısı Sapma*** *(b_a)* ***Değerleri*** eklenir. Oluşan çıktı tanh fonksiyonuna tabi tutularak Aktivasyon Kapısı çıkışları oluşur. Bu bölümün son adımında ise Giriş Kapısı çıkışları ve Aktivasyon Kapısı çıkışları çarpılarak bu bölümün asıl çıktısı oluşturulur. Şekil 3.9'de yapılan işlemler görsel olarak açıklanmıştır. [8] [38]



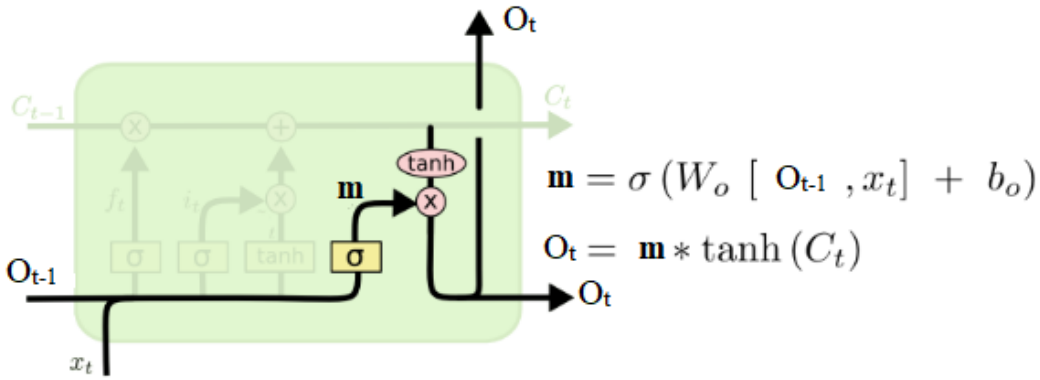
Şekil 3.9 : UKSB Giriş Kapısı ve Aktivasyon Kapısı hesaplaması. [8]

Bu kısma kadar sonuç olarak elimizde Giriş Kapısı, Aktivasyon Kapısı ve Unutma Kapısı çıkışları mevcuttur. Sıradaki adımda bu çıkışlar kullanılarak ***Bağlam Çıkışları (Context Output)*** değerleri elde edilecektir. Bu değerler aslında bir sonraki UKSB hücresi için kullanılacak değerlerdir ve amacı adından da anlaşılacağı gibi veri sırası arasında bir bağlantı kurmak yani sıralamayı gözetmektir. Şekil 3.10 incelenirse Unutma Kapısı çıkışı ile bir önceki bağlam çıkışının çarpılarak yeni bir çıkışın oluşturulduğu, ardından da Giriş Kapısı ve Aktivasyon Kapısı çıkışlarının çarpılmasıyla oluşan yeni bir çıkış ile çarpılarak bağlam çıkışlarının (context output) elde edildiği görülebilir. Oluşan bağlam çıkışları bir sonraki adımda kullanılacaktır. [8]



Şekil 3.10 : UKSB bağlam çıkışlarının hesapmalası. [8]

Son adımda UKSB çıkışları hesaplanmalıdır. Çıkışları hesaplamak için bir önceki çıkışlar ile giriş verisi birleştirilerek bir sigmoid fonksiyonundan geçirilir ve bu değer bir önceki adımda elde edilen bağlam çıkışlarını tanh fonksiyonundan geçirerek çarpılır. Elde edilen bu değer UKSB hücresinin çıkışıdır. Şekil 3.11'da anlatılanlar resmedilmiştir. [8]



Şekil 3.11 : UKSB çıkışlarının hesapmalası. [8]

Birden fazla hücre olması durumunda bağlam çıkışları ve hücre çıkışları şekil 3.7'deki gibi bağlanarak bir UKSB ağ yapısı oluşturulmaktadır. İleri fazlarda tüm

hücreler yukarıda anlatılanlar gibi hesaplanacaktır. Tüm UKSB hücreleri için çıkışlar hesaplandıktan sonra güncelleme değerleri için geri faz başlar. İleri fazda hesaplamalar en soldaki UKSB hücresinden başlar ve en sağdaki hücrenin çıkışı elde edilene kadar devam eder. Ardından zor olan kısım geri faz başlar. Geri faz ise ters yönde ilerletilerek en sağdan en sola doğru yani çıkıştan girişe doğru devam edecektir.

3.2.4.2 UKSB geri fazı

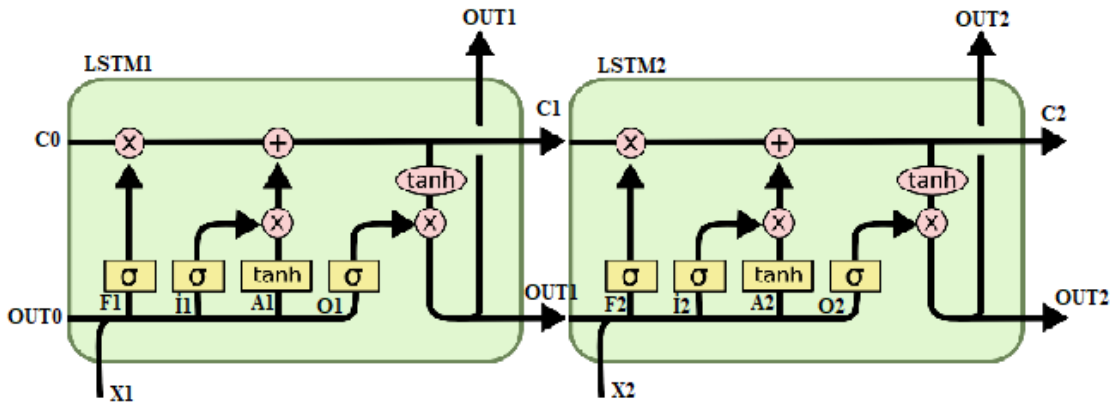
UKSB geri faz hesaplaması, ileri faz hesaplamasına göre biraz daha karmaşık bir yapıdır. Geriye doğru hatanın yayılması için çok sayıda türev ifadesi içerir. Bunun için ***Zincir Kuralı*** kullanılmaktadır. Zincir kuralı bir türev ifadesini içerikleri cinsinden açarak ayrı ayrı hesaplamaktır. [42] Zincir kuralı ile ilgili küçük bir örnek aşağıda gösterilmiştir. Örneğin;

$$y = u^{\frac{5}{3}} \quad ve \quad u = 7x^8 + 3x^3$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

$$\frac{5}{3}u^{2/3}.(56x^7 + 9x^2) \tag{3.1}$$

Geri faz kapsamında sıklıkla zincir kuralından yararlanılacaktır. Bu kısımda işlemlerin net algılanabilmesi ve temel bir formül çıkarabilmek için Şekil 3.12'de gösterilen örnek ağ yapısı kullanılacaktır. Hesaplamalar bu örnek ağ yapısı üzerinden gerçekleştirilecektir. [43]



Şekil 3.12 : Hesaplamalar için oluşturulmuş UKSB ağı. [8]

Öncelikle Şekil 3.12'de gösterilen ağın ileri fazı için formüllerini oluşturalım. Bu formüller üzerinden güncelleme değerleri formülleri oluşturulacaktır. [43]

UKSB 1:

$$F_1 = \sigma(W_{F1}X_1 + U_{F1}Out_0 + b_{F1}) \tag{3.2}$$

$$I_1 = \sigma(W_{I1}X_1 + U_{I1}Out_0 + b_{I1}) \tag{3.3}$$

$$A_1 = tanh(W_{A1}X_1 + U_{A1}Out_0 + b_{A1}) \tag{3.4}$$

$$O_1 = \sigma(W_{O1}X_1 + U_{O1}Out_0 + b_{O1}) \tag{3.5}$$

$$C_1 = C_0F_1 + I_1A_1 \tag{3.6}$$

$$Out_1 = O_1tanh(C_1) \tag{3.7}$$

$$HATA : E_1 = \frac{1}{2}(Out_1 - y_1)^2 \tag{3.8}$$

UKSB 2:

$$F_2 = \sigma(W_{F2}X_2 + U_{F2}Out_1 + b_{F2}) \tag{3.9}$$

$$I_2 = \sigma(W_{I2}X_2 + U_{I2}Out_1 + b_{I2}) \tag{3.10}$$

$$A_2 = tanh(W_{A2}X_2 + U_{A2}Out_1 + b_{A2}) \tag{3.11}$$

$$O_2 = \sigma(W_{O2}X_2 + U_{O2}Out_1 + b_{O2}) \tag{3.12}$$

$$C_2 = C_1F_2 + I_2A_2 \tag{3.13}$$

$$Out_2 = O_2tanh(C_2) \tag{3.14}$$

$$HATA : E_2 = \frac{1}{2}(Out_2 - y_2)^2 \tag{3.15}$$

İleri fazı hesaplarken en soldaki UKSB hücresinden en sağdakine doğru gidilir. Geri fazda ise en sağdaki yani bu ağda UKSB 2 hücresinden başlanır. Ardından UKSB 1

hücresine geçilir. En sondaki UKSB hücresinin çıkarımını yapmak en baştaki UKSB hücresine göre daha kolaydır. Ayrıca bağımlılık gereği en sondaki UKSB hücresinden başlayarak hata yayılmalıdır.

Öncelikle UKSB 2 hücresi içerisindeki ağırlıkları (W ve U değerleri) güncelleyerek başlanabilir. Güncelleme için zincir kuralı kullanılacaktır.

UKSB 2 Çıkış Kapısı Ağırlık 1 (W_{O2}):

$$\frac{\partial E_2}{\partial W_{O2}} = \frac{\partial E_2}{\partial Out_2}\frac{\partial Out_2}{\partial O_2(out)}\frac{\partial O_2(out)}{\partial O_2(in)}\frac{\partial O_2(in)}{\partial W_{O2}}$$

$$= (Out_2 - y_2)tanh(C_2)(\sigma(W_{O2}X_2 + U_{O2}Out_1 + b_{O2})$$

$$(1 - \sigma(W_{O2}X_2 + U_{O2}Out_1 + b_{O2})))X_2 \tag{3.16}$$

UKSB 2 Çıkış Kapısı Ağırlık 2 (U_{O2}):

$$\frac{\partial E_2}{\partial U_{O2}} = \frac{\partial E_2}{\partial Out_2}\frac{\partial Out_2}{\partial O_2(out)}\frac{\partial O_2(out)}{\partial O_2(in)}\frac{\partial O_2(in)}{\partial U_{O2}}$$

$$= (Out_2 - y_2)tanh(C_2)(\sigma(W_{O2}X_2 + U_{O2}Out_1 + b_{O2})$$

$$(1 - \sigma(W_{O2}X_2 + U_{O2}Out_1 + b_{O2})))Out_1 \tag{3.17}$$

UKSB 2 Aktivasyon Kapısı Ağırlık 1 (W_{A2}):

$$\frac{\partial E_2}{\partial W_{A2}} = \frac{\partial E_2}{\partial Out_2}\frac{\partial Out_2}{\partial C_2(out)}\frac{\partial C_2(out)}{\partial C_2(in)}\frac{\partial C_2(in)}{\partial A_2(out)}\frac{\partial A_2(out)}{\partial A_2(in)}\frac{\partial A_2(in)}{\partial W_{A2}}$$

$$= (Out_2 - y_2)(O_2)(1 - tanh(C_2)^2)(I_2)$$

$$(1 - tanh(W_{A2}X_2 + U_{A2}Out_1 + b_{A2})^2)X_2 \tag{3.18}$$

UKSB 2 Aktivasyon Kapısı Ağırlık 2 (U_{A2}):

$$\frac{\partial E_2}{\partial U_{A2}} = \frac{\partial E_2}{\partial Out_2}\frac{\partial Out_2}{\partial C_2(out)}\frac{\partial C_2(out)}{\partial C_2(in)}\frac{\partial C_2(in)}{\partial A_2(out)}\frac{\partial A_2(out)}{\partial A_2(in)}\frac{\partial A_2(in)}{\partial U_{A2}}$$

$$= (Out_2 - y_2)(O_2)(1 - tanh(C_2)^2)(I_2)$$

$$(1 - tanh(W_{A2}X_2 + U_{A2}Out_1 + b_{A2})^2)Out_1 \tag{3.19}$$

UKSB 2 Giriş Kapısı Ağırlık 1 (W_{I2}):

$$\frac{\partial E_2}{\partial W_{I2}} = \frac{\partial E_2}{\partial Out_2}\frac{\partial Out_2}{\partial C_2(out)}\frac{\partial C_2(out)}{\partial C_2(in)}\frac{\partial C_2(in)}{\partial I_2(out)}\frac{\partial I_2(out)}{\partial I_2(in)}\frac{\partial I_2(in)}{\partial W_{I2}}$$

$$= (Out_2 - y_2)(O_2)(1 - tanh(C_2)^2)(A_2)(\sigma(W_{I2}X_2 + U_{I2}Out_1 + b_{I2})$$

$$(1 - \sigma(W_{I2}X_2 + U_{I2}Out_1 + b_{I2})))X_2 \tag{3.20}$$

UKSB 2 Giriş Kapısı Ağırlık 2 (U_{I2}):

$$\frac{\partial E_2}{\partial U_{I2}} = \frac{\partial E_2}{\partial Out_2}\frac{\partial Out_2}{\partial C_2(out)}\frac{\partial C_2(out)}{\partial C_2(in)}\frac{\partial C_2(in)}{\partial I_2(out)}\frac{\partial I_2(out)}{\partial I_2(in)}\frac{\partial I_2(in)}{\partial U_{I2}}$$

$$= (Out_2 - y_2)(O_2)(1 - tanh(C_2)^2)(A_2)(\sigma(W_{I2}X_2 + U_{I2}Out_1 + b_{I2})$$

$$(1 - \sigma(W_{I2}X_2 + U_{I2}Out_1 + b_{I2})))Out_1 \tag{3.21}$$

UKSB 2 Unutma Kapısı Ağırlık 1 (W_{F2}):

$$\frac{\partial E_2}{\partial W_{F2}} = \frac{\partial E_2}{\partial Out_2}\frac{\partial Out_2}{\partial C_2(out)}\frac{\partial C_2(out)}{\partial C_2(in)}\frac{\partial C_2(in)}{\partial F_2(out)}\frac{\partial F_2(out)}{\partial F_2(in)}\frac{\partial F_2(in)}{\partial W_{F2}}$$

$$= (Out_2 - y_2)(O_2)(1 - tanh(C_2)^2)(S_1)(\sigma(W_{F2}X_2 + U_{F2}Out_1 + b_{F2})$$

$$(1 - \sigma(W_{F2}X_2 + U_{F2}Out_1 + b_{F2})))X_2 \tag{3.22}$$

UKSB 2 Unutma Kapısı Ağırlık 2 (U_{F2}):

$$\frac{\partial E_2}{\partial U_{F2}} = \frac{\partial E_2}{\partial Out_2}\frac{\partial Out_2}{\partial C_2(out)}\frac{\partial C_2(out)}{\partial C_2(in)}\frac{\partial C_2(in)}{\partial F_2(out)}\frac{\partial F_2(out)}{\partial F_2(in)}\frac{\partial F_2(in)}{\partial U_{F2}}$$

$$= (Out_2 - y_2)(O_2)(1 - tanh(C_2)^2)(S_1)(\sigma(W_{F2}X_2 + U_{F2}Out_1 + b_{F2})$$

$$(1-\sigma(W_{F2}X_2+U_{F2}Out_1+b_{F2})))Out_1 \tag{3.23}$$

UKSB 2 hücresi için ağırlıkların güncellenmesi, UKSB 1 hücresine göre biraz daha kolaydır. UKSB 1 hücresi içerisindeki ağırlıklar güncellenirken bağımlılıklar ve tüm durumlar dikkate alınmalıdır. Tüm durumların dikkate alınması ek çıkarımlar getirecek ve ifade kalabalıklaşacaktır. Bu kısımda UKSB 1 hücresi için ağırlıkları hesaplarken sadece W_{O1} ağırlığı için hesaplama yapılacaktır. Diğer hesaplamalar bu hesaplama baz alınarak benzer şekilde çıkartılabilir. [43] Burada önemli olan tüm durumların göz önünde bulundurulmasıdır.

UKSB 1 Çıkış Kapısı Ağırlık 1 (W_{O1}):

$$\frac{\partial E_{toplam}}{\partial W_{O1}} = \frac{\partial E_1}{\partial W_{O1}} + \frac{\partial E_2}{\partial W_{O1}} \tag{3.24}$$

Zincir kuralı uygulanması durumunda;

$$\frac{\partial E_{toplam}}{\partial W_{O1}} = \frac{\partial E_1}{\partial Out_1}\frac{\partial Out_1}{\partial O_1(out)}\frac{\partial O_1(out)}{\partial O_1(in)}\frac{\partial O_1(in)}{\partial W_{O1}} + \frac{\partial E_2}{\partial W_{O1}}$$

$$\frac{\partial E_{toplam}}{\partial W_{O1}} = (Out_1 - y_1)(tanh(C_1))$$

$$(\sigma(W_{O1}X_1+U_{O1}Out_0+b_{O1})(1-\sigma(W_{O1}X_1+U_{O1}Out_0+b_{O1})))X_1 + \frac{\partial E_2}{\partial W_{O1}} \tag{3.25}$$

İfade içerisindeki $\frac{\partial E_2}{\partial W_{O1}}$ kısmının açılması durumunda; (Her durumun incelendiğine dikkat edilmelidir.)

$$\frac{\partial E_2}{\partial W_{O1}} = \frac{\partial E_2}{\partial Out_2}\frac{\partial Out_2}{\partial O_2(out)}\frac{\partial O_2(out)}{\partial O_2(in)}\frac{\partial O_2(in)}{\partial Out_1}\frac{\partial Out_1}{\partial O_1(out)}\frac{\partial O_1(out)}{\partial O_1(in)}\frac{\partial O_1(in)}{\partial W_{O1}} +$$

$$\frac{\partial E_2}{\partial Out_2}\frac{\partial Out_2}{\partial C_2(out)}\frac{\partial C_2(out)}{\partial C_2(in)}\frac{\partial C_2(in)}{\partial A_2(out)}\frac{\partial A_2(out)}{\partial A_2(in)}\frac{\partial A_2(in)}{\partial Out_1}\frac{\partial Out_1}{\partial O_1(out)}\frac{\partial O_1(out)}{\partial O_1(in)}\frac{\partial O_1(in)}{\partial W_{O1}} +$$

$$\frac{\partial E_2}{\partial Out_2}\frac{\partial Out_2}{\partial C_2(out)}\frac{\partial C_2(out)}{\partial C_2(in)}\frac{\partial C_2(in)}{\partial I_2(out)}\frac{\partial I_2(out)}{\partial I_2(in)}\frac{\partial I_2(in)}{\partial Out_1}\frac{\partial Out_1}{\partial O_1(out)}\frac{\partial O_1(out)}{\partial O_1(in)}\frac{\partial O_1(in)}{\partial W_{O1}} +$$

$$\frac{\partial E_2}{\partial Out_2}\frac{\partial Out_2}{\partial C_2(out)}\frac{\partial C_2(out)}{\partial C_2(in)}\frac{\partial C_2(in)}{\partial F_2(out)}\frac{\partial F_2(out)}{\partial F_2(in)}\frac{\partial F_2(in)}{\partial Out_1}\frac{\partial Out_1}{\partial O_1(out)}\frac{\partial O_1(out)}{\partial O_1(in)}\frac{\partial O_1(in)}{\partial W_{O1}} \tag{3.26}$$

olarak hesaplanır. Zaten UKSB ağlarının hesaplanmasındaki zorluk, her durumu dahil etmekten gelmektedir. Sonuç olarak;

$$\frac{\partial E_{toplam}}{\partial W_{O1}} = \frac{\partial E_1}{\partial W_{O1}} + \frac{\partial E_2}{\partial W_{O1}} = (Out_1 - y_1)(tanh(C_1))(O_1)(1-O_1)))X_1 +$$

$$(Out_2 - y_2)(tanh(C_2))(O_2(1-O_2)(U_{O2})(tanh(C_1))(O_1(1-O_1)(X_1) +$$

$$(Out_2 - y_2)(O_2)(1-tanh(C_2)^2)(I_2)(1-A^2)(U_{A2})(tanh(C_1))(O_1(1-O_1))(X_1) +$$

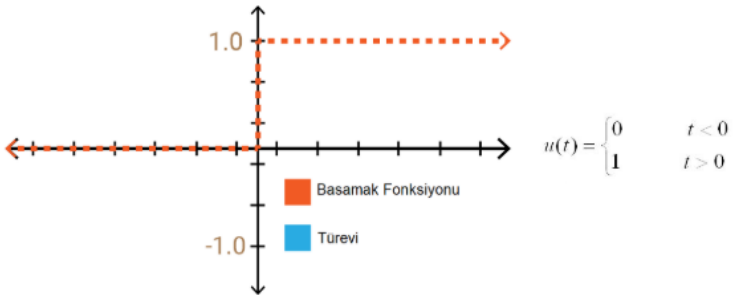
$$(Out_2 - y_2)(O_2)(1-tanh(C_2)^2)(A_2)(I_2(1-I_2))(U_{I2})(tanh(C_1))(O_1(1-O_1))(X_1) +$$

$$(Out_2 - y_2)(O_2)(1-tanh(C_2)^2)(C_1)(F_2(1-F_2))(U_{F2})(tanh(C_1))(O_1(1-O_1))(X_1) \quad (3.27)$$

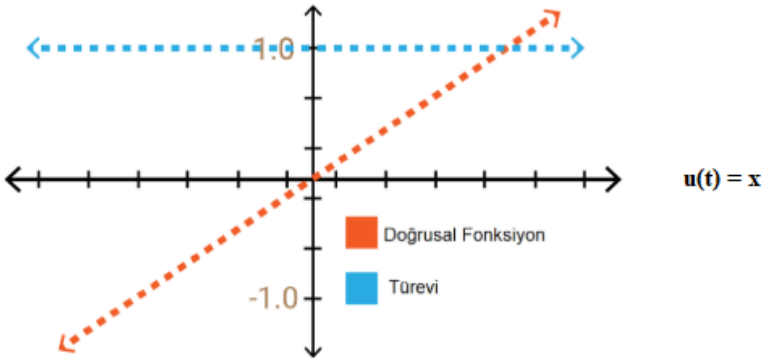
3.2.5 Yoğun katman içeren UKSB mimarisi

Yoğun katmanlar standart yapay sinir hücrelerinden farklı olmayan yapılardır. Kendisine gelen değerleri üzerindeki ağırlık değeri ile çarptıktan ve bu sonuca bir sapma ekledikten sonra elde ettiği sonucu aktivasyon fonksiyonundan geçiren yapılardır. [44] "$çıkış = aktivasyonfonksiyonu(W * X + b)$" formülü ile yoğun katmanın veri üzerindeki işlemi anlaşılabilir. Aktivasyon fonksiyonu kısmı çeşitlendirilebilir ve farklı tipte aktivasyon fonksiyonları tercih edilebilir. Aşağıda en sık kullanılan aktivasyon fonksiyonlarından bahsedilmiştir. [35]

- *Birim Basamak Fonksiyonu*: Sıfırdan büyük ve eşit olan her durumda 1 değeri alan basamak fonksiyonudur. W*x + b sonucu sıfırdan büyük ise çıkışı direkt olarak 1 yapar. Şekil 3.13'de bu tip bir fonksiyon verilmiştir.



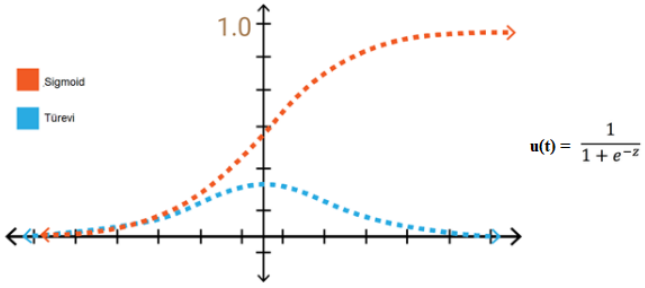
Şekil 3.13 : Birim basamak fonksiyonu [13]



Şekil 3.14 : Doğrusal fonksiyon [13]

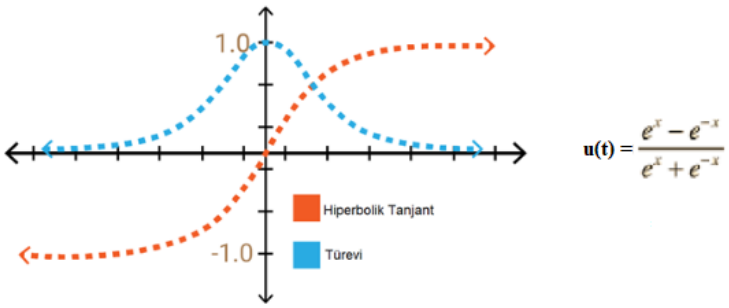
- *Doğrusal Fonksiyon*: Girişine verilen değerleri çıkışa direkt olarak aktaran fonksiyondur. Çalışma kapsamındaki kontrolör tasarımında bu tip bir aktivasyon fonksiyonu tercih edilmiştir. Şekil 3.14'de bu tip bir fonksiyon verilmiştir.

- *Sigmoid Fonksiyonu*: 0 ve 1 aralığında doğrusal olmayan değerler üreten bir fonksiyondur. Pek çok makine öğrenmesi mimarisinde tercih edilir. Şekil 3.15'de bu tip bir fonksiyon verilmiştir.



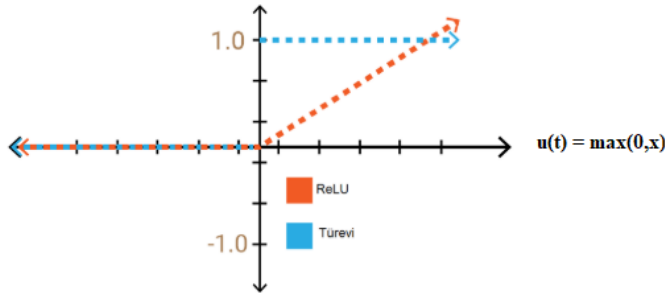
Şekil 3.15 : Sigmoid fonksiyonu [13]

- *Hiperbolik Tanjant Fonksiyonu*: Sigmoid fonksiyondan farklı olarak -1 ve 1 aralığında doğrusal olmayan değerler üretmektedir. Şekil 3.16'de bu tip bir fonksiyon verilmiştir.



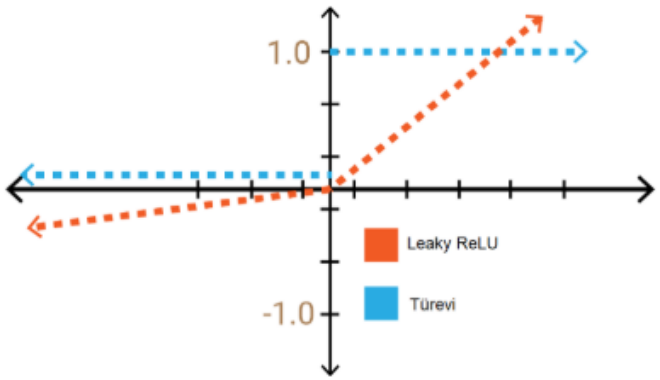
Şekil 3.16 : Hiperbolik Tanjant fonksiyonu [13]

- *ReLU Fonksiyonu*: Sıfırdan büyük değerlerde doğrusal fonksiyon gibi davranan ancak sıfırdan küçük değerlerde çıkışı sıfır olarak üreten bir fonksiyondur. Şekil 3.17'de bu tip bir fonksiyon verilmiştir.



Şekil 3.17 : ReLU fonksiyonu [13]

- *Sızıntı ReLU Fonksiyonu*: ReLU işaretinden farklı olarak sıfırdan küçük değerlerde direkt 0 çıkışı vermek yerine küçük bir sızıntı değeri üretmektedir. Şekil 3.18'de bu tip bir fonksiyon verilmiştir.



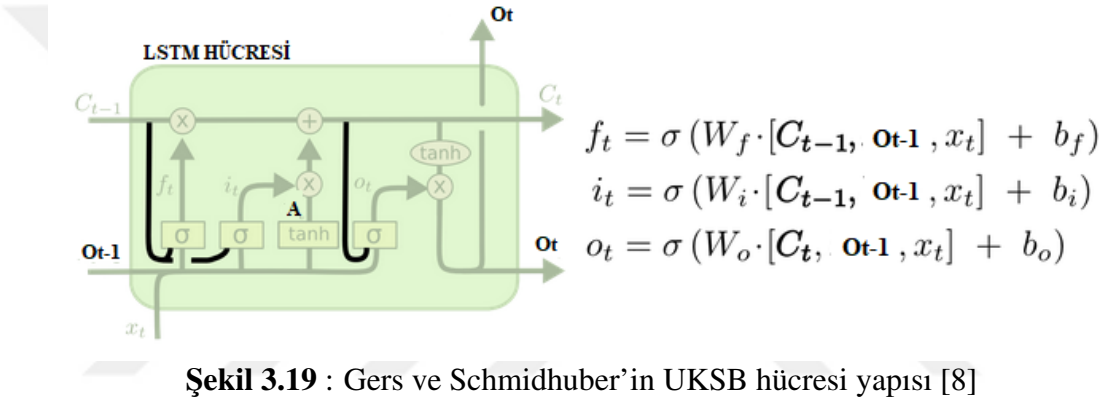
Şekil 3.18 : Sızıntı ReLU fonksiyonu [13]

Yoğun katmanların asıl amacı doğrusallığı kırarak doğrusal olmayan çıkışları da modelleyebilmeyi sağlamaktır. [45] Yoğun katman sayesinde ağlar birleştirilerek regresyon problemleri için sayısal çıkışların üretilebilmesi sağlanır. Karşımızda bir regresyon problemi varsa çıkışında sayısal bir veri elde etmek için yoğun katman kullanılmalıdır. Çalışma kapsamında sistem tanımlamada üretilecek çıkışlar ve kontrol için kullanılan UKSB mimarilerinin sonlarına yoğun katman eklenmiştir. Bu eklentiler sayesinde sistem çıkışları için gerekli sayısal veriler üretilebilmiştir. Bir ağa yoğun katman eklenmesi, geri faz ve ileri faz adımlarında ek hesaplamalar getirir. Çalışmanın ilerleyen bölümlerinde özellikle kontrol alanında yoğun katman içeren bir yapı için nasıl derivasyon yapılacağı açıklanmıştır.

3.2.6 Farklı UKSB mimarileri

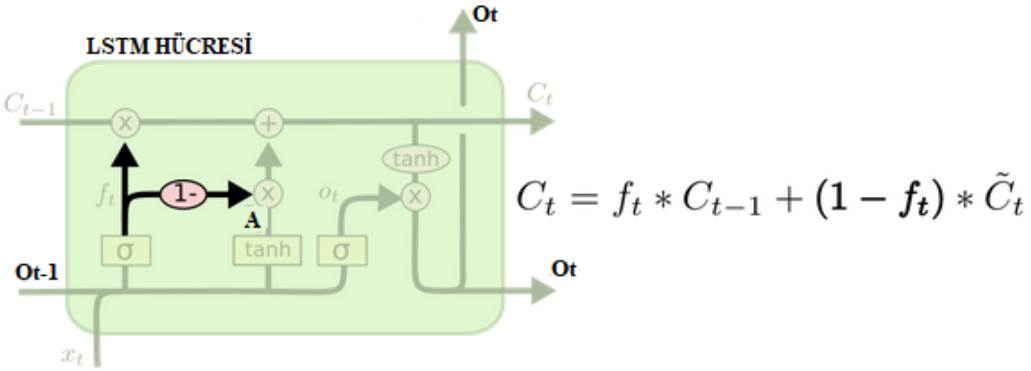
Yukarıdaki bölümlerde bahsedilen UKSB hücresinin her zaman aynı şekilde kullanılmasına gerek yoktur. Değişik uygulamalarda farklı UKSB hücreleri karşımıza çıkabilir. [8] Uygulamada kullanılan UKSB yapısı en sık kullanılan UKSB yapısı olmakla birlikte bu bölümde değişik UKSB hücrelerine de yer verilmiştir. [38]

Şekil 3.19'de Gers ve Schmidhuber'ın geliştirdiği UKSB yapısı gösterilmiştir. Çalışmada kullanılan UKSB yapısına ek olarak bağlantılar geliştirilmiştir. Ancak bu tip yapılarda bağlantıların sayısı arttığından işlem yükünün de artacağı unutulmamalıdır. [8]



Şekil 3.19 : Gers ve Schmidhuber'in UKSB hücresi yapısı [8]

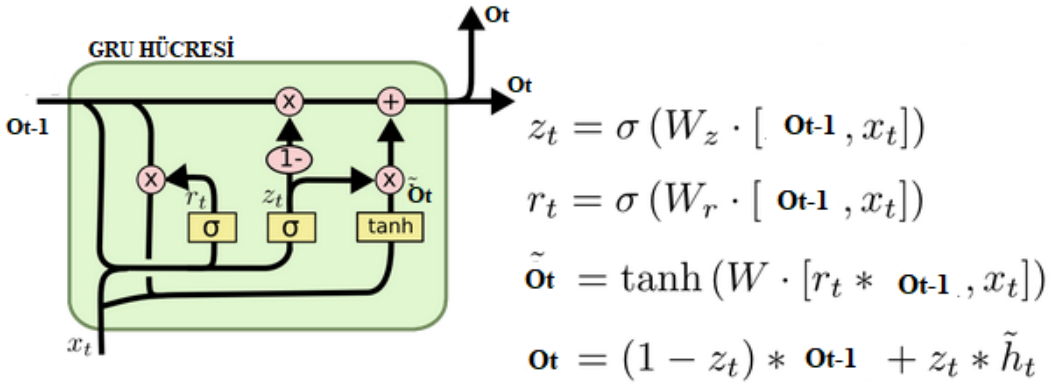
Şekil 3.20'deki UKSB hücresinde ise Giriş Kapısı ve Unutma Kapısı arasına bağlantı kurularak farklı bir yapı oluşturulmuştur.



Şekil 3.20 : Farklı bir UKSB hücresi yapısı [8]

Şekil 3.21'de ise en az UKSB mimarisi kadar ünlü ve bazı uygulamalarda daha başarılı olan ***Geçitli Tekrarlama Ünitesi (GRU)*** gösterilmiştir. Yeni bir teknoloji olan GRU ağlarının eğitimleri UKSB mimarilerine göre biraz daha karmaşıktır. İlerleyen

uygulamalarda bu tip UKSB hücreleri de tercih edilebilir. Ancak çalışma içerisinde 'ın 1997'de ortaya attığı klasik UKSB hücresi kullanılmıştır.



Şekil 3.21 : GRU hücresi yapısı [8]

3.2.7 KERAS kütüphanesi

Makine öğrenmesi algoritmalarını uygulamak için en kolay dil Python dilidir. Python kullanımı oldukça kolay bir dil olmasının yanı sıra geniş bir kütüphane desteğini de içerisinde barındırır. Bu kütüphanelerden KERAS, içerisinde pek çok makine öğrenmesi algoritması içeren geniş bir kütüphanedir. KERAS'ın önemli bir özelliği de, işlemleri olabildiğince optimum sürede gerçekleştirmesi ve paralel işlemler kullanmasıdır. KERAS, içerisinde UKSB kütüphanesi de bulundurur ve kullanımı oldukça kolay fonksiyonlar içerir. KERAS kütüphanesi içerisindeki UKSB yapısı, Hochreiter 1997'da geliştirilen ve Şekil 3.7'de gösterilen UKSB yapısıdır. [46] Bu hücre yapısı, çalışma kapsamında kullanılan UKSB ağına oldukça uygundur. KERAS kütüphanesi ve Python dilinin önemli bir özelliği de MATLAB benzeri bir yapıda olması ve kontrol işlemlerinde de bire bir dönüşüm sağlanabilmesidir. KERAS kütüphanesinde UKSB aşağıdaki fonksiyon şeklinde üretilebilir. Görüldüğü gibi temel parametreler ile kolay bir şekilde ağ inşa etmek mümkündür.

LSTM*(Hücre Sayısı,activation='tanh')*

Yukarıdaki yapı çalışmanın bir sonraki bölümünde daha detaylı açıklanacak ve kullanılan ağ hakkında bilgiler verilecektir. Uygulama genelinde KERAS ve Python sıklıkla tercih edilmiştir ve çıktılar üretilmiştir.

4. UKSB TABANLI SİSTEM TANIMLAMA

4.1 Giriş

Çalışmanın bu kısmında sistem tanımlama uygulamasının, derin öğrenme yöntemlerinden UKSB ile nasıl gerçekleştirildiği anlatılacaktır. Ardından UKSB ile sistem tanımlama, iki örnek sisteme uygulanarak sonuçlar incelenecektir.

Sistem tanımlama için UKSB kullanılmasının en büyük nedeni, UKSB mimarisinin geçmişe bağlı verileri kullanabilmesi ve bu veriler arasındaki sıralamayı önemsemesi olduğundan bir önceki bölümde bahsedilmişti. UKSB'nin sistemin şimdiki ve geçmiş verilerini kullanarak çıkışında sistemin N adet gelecek değeri tahmin etmeye çalışması beklenir. UKSB yapısı içerisine verilen geçmiş değer sayısı arttırıldığında sistem daha iyi sonuç vermeye başlar ancak çok fazla geçmiş değerin veri içerisine dahil edilmesi işlem süresini arttırır ve sistemi ezbere götürebilir. Bundan dolayı geçmiş veri sayısı optimum bir değerde seçilmelidir. [25]

4.2 UKSB Tabanlı Sistem Tanımlama Adımları

Sistem tanımlama yapısının ilk adımı sistem üzerinden veri elde etmektir. Bunun için de gerçek sisteme uygun giriş sinyali verilmelidir. Veriler toplandıktan sonra bu veriler UKSB yapısı için uygun hale getirilerek UKSB'e verilir, bu adıma verilerin önsel işlenmesi denir. Bu işlem ardından eğitim başlar. UKSB, sistem yapısını öğrendikten sonra artık sistem yerine UKSB ağı kullanılabilir.

4.2.1 Kullanılan giriş işareti

Önceki bölümlerde pek çok giriş sinyalinden bahsedilmişti. Bir sistemi tanımlamak için ikinci bölümde söz edilen giriş sinyallerden herhangi birisi kullanılabilir ancak bu sinyallerin etkileri birbirinden farklı olacaktır. Çalışma kapsamında iyi sonuç vermesi ve bu tip uygulamalarda çok sık kullanılmasından dolayı ***Sözde Rassal Sayı Üreteci(PRBS)***'nden faydalanılmıştır. Bu yapı direkt olarak kullanılmamış bunun yerine geliştirilerek farklı sinyal seviyeleri içeren bir yapı haline dönüştürülmüştür. Bu

sayede sistemlerin farklı davranışlarının da veri seti içerisinde yer alabilmesi ve sistem karakteristiğinin doğru çıkarılabilmesi amaçlanmıştır. Sisteme uygulayacağımız PRBS işareti rassal bir işaret olduğundan her seferinde farklı çıktılar üretecektir. Bunun sonucunda sistemden veri çıkarabileceğimiz bir giriş sinyali üretilebilir. Üretilecek sistem tanımlama giriş sinyali için algoritma adımları aşağıdaki gibi olacaktır:

1. Sistem için bir ilk değer seçilir. (Bu ilk değer çevresinde işaret üretileceği için sistemin giriş seviyesi sınırının orta noktası seçilmesi mantıklı olacaktır.)

2. Sistem girişi için bir aralık seçilir ve bu aralığın - ve + değerleri aralığında rastgele sayı üretilir.

3. İşaretin değişkenliğini değiştirmek amacıyla bir değişim zaman değeri seçilir. Buradaki amaç sistemin davranışlarını çözümlemek için sık sık işaret seviyesini değiştirmek yerine belirli zamanlarda değiştirmektir. Çok küçük değişim zamanları seçilirse UKSB, sistemi tanımlayamayacaktır. Çok büyük değişim zamanları seçilirse de bu kez UKSB, sistemi ezberlemeye gidecek ve test sonuçları oldukça kötü çıkacaktır. Bundan dolayı uygun bir değişim zamanı seçilmelidir.

4. Simulasyon zamanı başlatılır ve ilk değer olarak 1.adımda seçilen ilk sinyal seviyesi kullanılır. Değişim zamanı geldiğinde 2.adımdaki rastgele sayı üretilir ve üretilen sayı ilk değere eklenerek bir sonraki değişim zamanına kadar giriş işareti olarak kullanılır.

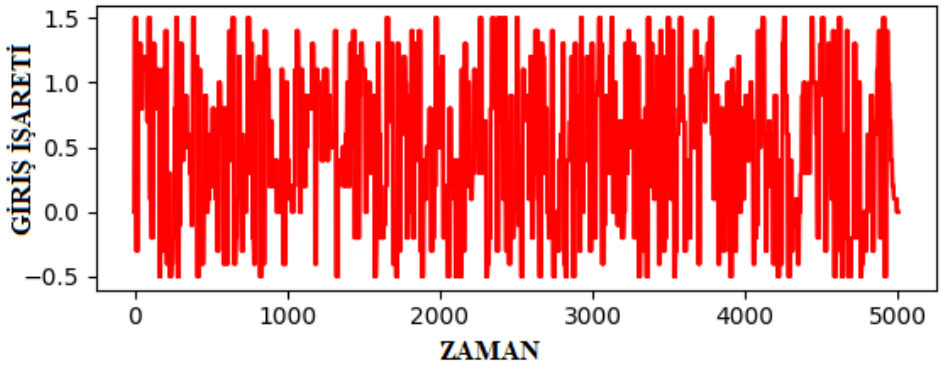
5. Oluşturulan işaret sisteme uygulanır ve sistem çıkışları toplanır. Girişler ve çıkışlardan oluşan bu veri seti ilerleyen aşamalarda UKSB'e verilerek, böylece UKSB sistemi öğrenir ve modeller.

6. Sistem için eğitilmiş UKSB, gerçek sisteme oldukça yakın değerler üretebiliyorsa bu durumda gerçek sistem yerine UKSB sistemi kullanılabilir.

Bu kısımda UKSB tabanlı sistem tanımlamayı uygulayabilmek için örnek olarak iki farklı sistemden bahsedilmiştir. Giriş sinyallerini daha iyi anlayabilmek için ilk sistem doğrusal olmayan ve zamanla değişen bir sistemdir. Sistemin simulasyonunu yaparak veri çıkarmak için aşağıdaki matematiksel ifade kullanılmıştır.

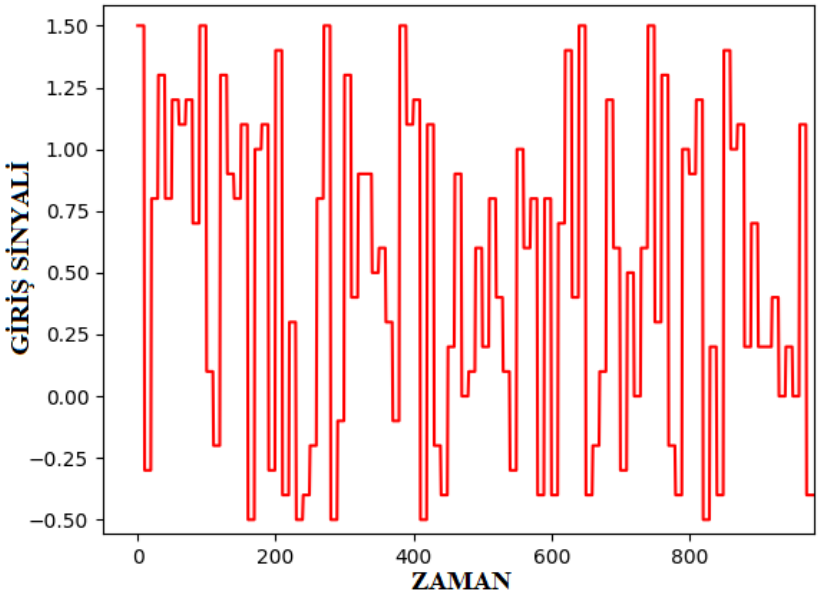
$$y(k+1) = \frac{1.2(1-0.8e^{-0.1k})y(k)}{1+y^2(k)} + u(k) \tag{4.1}$$

Denklemin girişi olan *u(k)*'ya PRBS sinyalini uygulayarak *y(k+1)* değerleri toplanacak ve UKSB için giriş verileri üretilecektir. Bu sisteme rassal işaret üretme algoritması uygulanmıştır. Rassal işaret algoritması için ilk değer olarak 0.5 değeri ve değişim aralığı olarak -1 ve 1 arası değerler seçilmiştir. Bu değerler ile sistem tanımlama için sistem girişine uygulanacak işaret şekil 4.1'deki gibi oluşturulacaktır.



Şekil 4.1 : Test sistemi için PRBS ile üretilen giriş işareti

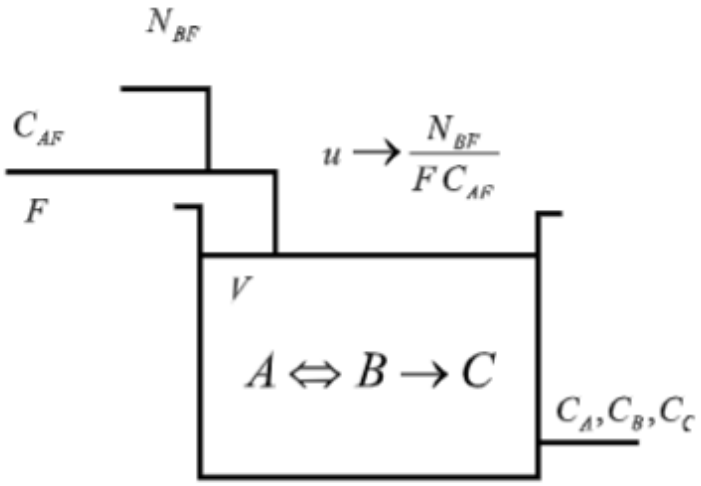
İşareti daha iyi anlayabilmek ve seviye değişimlerini daha anlaşılabilir bir şekilde görebilmek için Şekil 4.1'deki işaret için 0 - 1000 aralığına yakınlaştırma işlemi gerçekleştirilerek Şekil 4.2 deki görsel elde edilmiştir. Bu şekilde sistem girişine uygulanacak işaretin ne olduğu daha net gözlemlenebilir hale getirilmiştir.



Şekil 4.2 : Test sistem için PRBS ile üretilen giriş işaretinin yakınlaştırılmış hali

UKSB tabanlı sistem tanımlamayı test etmek için kullanılacak ikinci sistem ise çok sık kullanılan ve yine doğrusal olmayan bir sistem olan *CSTR* (continuously stirred tank

reactor) yani *sürekli karıştırmalı tank reaktör*üdür. Bu reaktör kimya endüstrisinde polimer, ilaç gibi kimyasalları üretmek için kullanılmaktadır.



Şekil 4.3 : Sürekli karıştırmalı tank reaktörü (CSTR) [14]

Şekil 4.3'de görüldüğü gibi bu problemde A ve B maddeleri sabit hacimli bir kap içerisinde sürekli olarak karıştırılarak yeni bir C maddesi oluşturulmaktadır. Kap içerisindeki reaksiyon iki aşamalıdır. İlk aşamada A ve B maddeleri arasında bir reaksiyon oluşur ardından gelen ikinci aşamada ise oluşan C maddesi ile B arasında yeni bir reaksiyon oluşmaktadır. Bu sistemde amaç, kabın girişi olan ve kabın sıcaklığına bağlı bir ifadeyi ayarlayarak C maddesinin *molar doyma hızı (molar feed rate)* değerini kontrol etmektir. Sistemin dinamik davranışları Kravaris ve Palanki tarafından bulunan dinamik denklemler ile aşağıdaki şekilde ifade edebilebilir: [14] Bu denklem kullanılarak gerçek sistemden veri çıkarma işlemi gerçeklenecektir.

$$\dot{X}_1(t) = 1 - X_1(t) - Da_1 X_1(t) + Da_2 X_2^2(t) \tag{4.2}$$

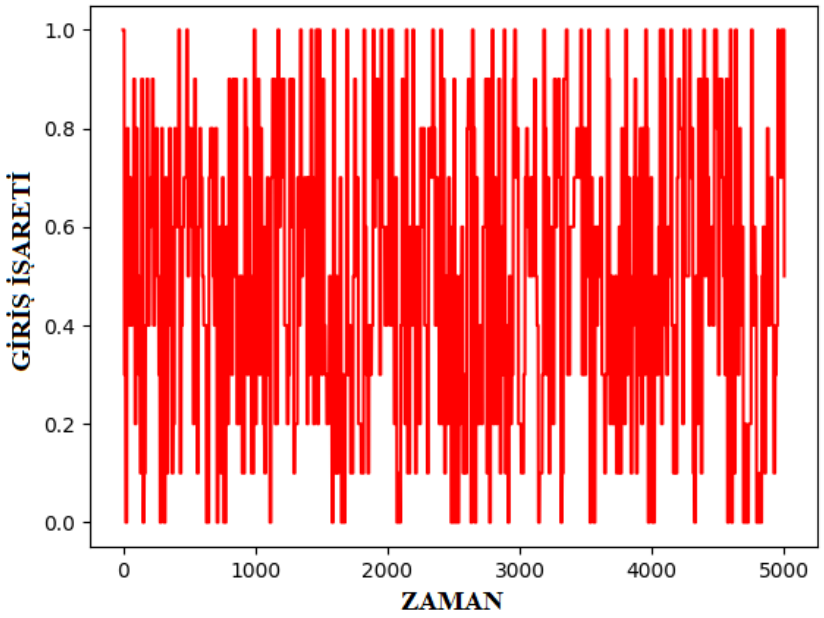
$$\dot{X}_2(t) = -X_2(t) + Da_1 X_1(t) - Da_2 X_2^2(t) - Da_3 d_2(t) X_2^2(t) + u(t) \tag{4.3}$$

$$\dot{X}_3(t) = -X_3(t) + Da_3 d_2(t) X_2^2(t) \tag{4.4}$$

$$u_{min} \leq u \leq u_{max} \tag{4.5}$$

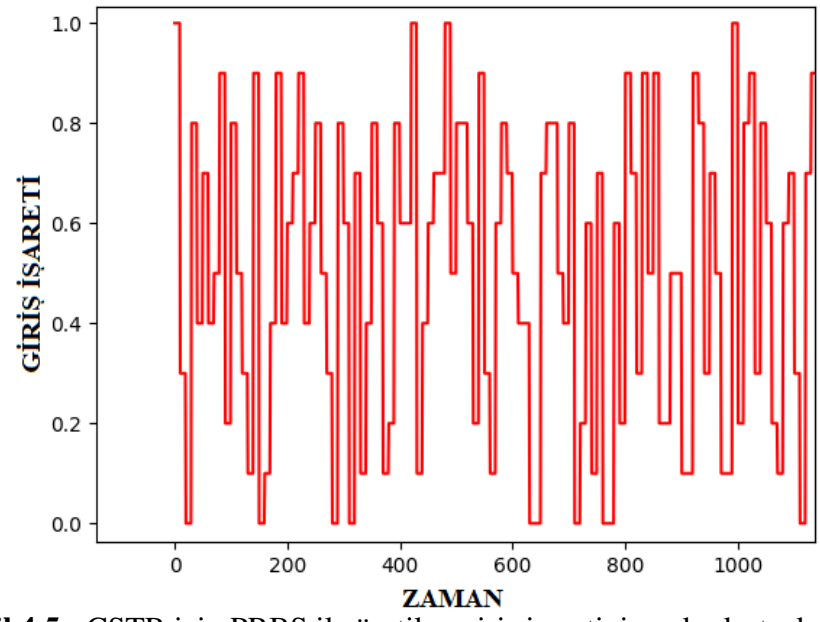
İfadede $X_1(t)$, $X_2(t)$ ve $X_3(t)$ sistem durumlarıdır. Çalışma genelinde $Da_1 = 3$, $Da_2 = 0.5$ ve $Da_3 = 1$ olarak alınmıştır. Uygulama içerisinde $u(t)$ giriş işaretidir ve

$u_{min} = 0$ ve $u_{max} = 1$ olarak seçilmiştir. Denklemler içerisindeki $d_2(t)$ zamana bağlı bir ifadedir ve reaksiyon aktivitesini ifade eder. Çalışmada nominal değer olarak $d_2(t) = 1$ alınabilir. [14] Seçilen ifadelere uygun olarak CSTR için, ilk değeri 0.5 ve değişim aralığı -1 ve 1 olarak seçilen bir rassal işaret üretilmiştir. Bu durumda CSTR girişine uygulanacak işaret Şekil 4.4'deki gibi üretilmiştir.



Şekil 4.4 : CSTR sistemi için PRBS ile üretilen giriş işareti

İşaretin daha iyi gözlemlenebilmesi için 0 - 1000 aralığına yakınlaştırma yapılabilir. Bu durumda karşımıza Şekil 4.5'deki görsel çıkacaktır. Bu görsel incelenirse girişe uygulanacak işaret daha net anlaşılabilmektedir.



Şekil 4.5 : CSTR için PRBS ile üretilen giriş işaretinin yakınlaştırılmış hali

Üretilen bu işaretler ve sonuçlarında oluşan çıkışlar UKSB girişlerine verilerek UKSB'in sistem davranışlarını öğrenmesi, hemen ardından da sistemin yerine eğitilmiş UKSB ağı yapısının geçmesi beklenir. Bu sayede matematiksel modeli bilinmeyen ancak elimizde verileri olan sistemler kontrol uygulamaları için kullanılabilir duruma gelecektir. Sisteme verilen rassal girişler ile çıkışların biraraya getirilmesiyle NARX modele uygun bir veri seti oluşturularak UKSB'e verileceği unutulmamalıdır.

4.2.2 UKSB mimarisi tasarımı

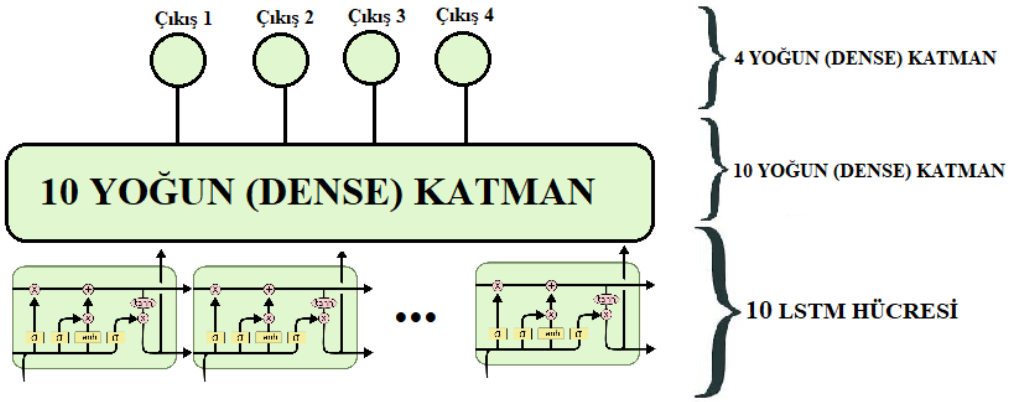
Giriş işareti oluşturulduktan sonra bu işaret ve sistem çıkışları bir araya getirilerek bir veri seti oluşturulmaktadır. Veri setinin bir kısmı oluşturulan UKSB mimarisine verilerek eğitiminde kullanılır, kalan kısmı ise eğitilmiş mimariyi test amacı ile kullanılmaktadır. Sisteme uygulanan giriş işareti ve sistem çıkışlarını kullanarak bir veri seti oluşturulur. İşaret içerisinde kullanılan geriye bağlı ifade sayısı 4 olarak belirlenmiştir. Bu durumda sistem tanımlamada UKSB girişine uygulanacak veri formatı aşağıdaki şekildedir:

$$[u_{t-3}, u_{t-2}, u_{t-1}, u_t, y_{t-3}, y_{t-2}, y_{t-1}, y_t] \tag{4.6}$$

UKSB ağı bu girişleri aldıktan sonra N adet sonraki sistem çıkışlarını tahmin etmeye çalışacaktır. Çalışma kapsamında ***öngörü ufku (prediction horizon)*** yani N = 4 olarak belirlenmiştir. Bu durumda UKSB, $y_{t+1}, y_{t+2}, y_{t+3}$ ve y_{t+4} değerlerini tahmin edecektir. Giriş formatı oluşturulduktan sonra UKSB ağ yapısı oluşturulmalıdır. Çalışma kapsamında, sistem tanımlama için aşağıdaki parametreler içeren bir UKSB ağı, KERAS kütüphanesi kullanılarak gerçekleştirilmiştir.

- Öngörü Ufku (Prediction Horizon) = 4
- UKSB Hücre Sayısı = 10
- UKSB Yoğunluk Katman (Dense Layer) Sayısı = [10,4]
- Dönem (Epoch) Sayısı = 50
- Paket (Batch) Sayısı = 1

Oluşturulan ağ yapısı çıkışında kullanılan *yoğunluk katmanının (dense layer)* amacı UKSB hücre çıkışlarının ardından gelerek değerleri sayısal değerlere dönüştürmektir. Şekil 4.6'da çalışma kapsamında kullanılan ağ yapısının görsel hali verilmiştir. Burada çıkış sayısı kaç adet geriye dönük veri tahmin etmek istediğimize göre arttırılabilir. Ancak bu durumda ek katmanlara gerek duyulacağından işlem yükü artar. Bundan dolayı bu kısımda bir optimizasyon yapılarak iki katmanlı bir yoğun katman modeli kullanılmış ve ilk yoğun katman için 10, hemen ardındaki yoğun katman için ise 4 nöron kullanılmıştır. Bu tip bir kullanımda ağ çıkışı 4 adım kadar ileriyi tahmin edebilir.

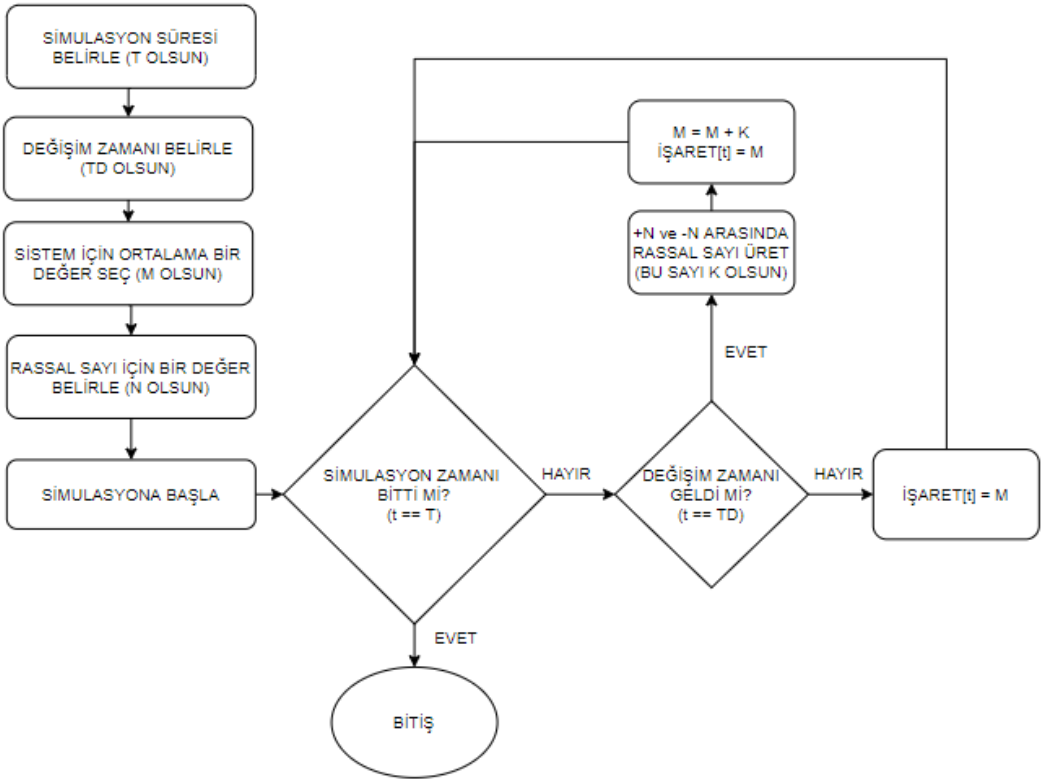


Şekil 4.6 : Sistem tanımlama için kullanılan ağ yapısı

Çalışmada, kullanılan dönem (epoch) sayısı seçimi de kritik bir rol oynar. Fazla seçilen dönem sayısı sistemi ezbere götürür ve UKSB, sistem içerisindeki gürültüleri de öğrenebilir. Az seçildiği durumlarda ise UKSB kararsızlaşır ve sistem tam olarak modellenemez. Dönem (Epoch) sayısının 50 seçilmesinin nedeni daha yüksek seçilen dönem sayılarında sistemin yitim (Loss) değerlerinin çok değişmemesidir. Daha fazla dönem değerlerinde de yitim değeri küçük değişimler halinde azalır ve gereksiz eğitim süresine yol açar. 50 dönem değeri, farklı denemeler sonucunda bulunmuş minimum dönem değeridir ve bu değerden daha büyük dönem değerleri başarıya büyük ölçüde etki etmemekle birlikte zaman kaybına yol açar. Çalışma kapsamındaki paket sayısı değeri ise verilerin eğitim sırasında UKSB'e kaçarlı paketler halinde aktarılacağı anlamına gelmektedir. Bu tip uygulamalarda UKSB'e verilerin teker teker aktarılması uygundur. Bu durumdan dolayı paket sayısı 1 olarak belirlenmiştir.

4.2.3 Algoritmik akışlar

UKSB tabanlı sistem tanımlama işlemi için iki adet akış şeması gösterilecektir. Ardından UKSB tabanlı sistem tanımlama blok diyagramı incelenecektir. İlk akış şeması daha önceden bahsettiğimiz ve sistem üzerinden veri çıkarmak için kullanılan sözde rassal sayı üreteci algoritmasının akış şemasıdır. Bu şema Şekil 4.7'de gösterilmiştir. Şemada görüldüğü gibi öncelikle sistem için ortalama bir değer belirlenir, bu değer üzerine değişim zamanı her geldiğinde + ve - aralıkta üretilen bir rassal sayı eklenerek sinyal oluşturulur.



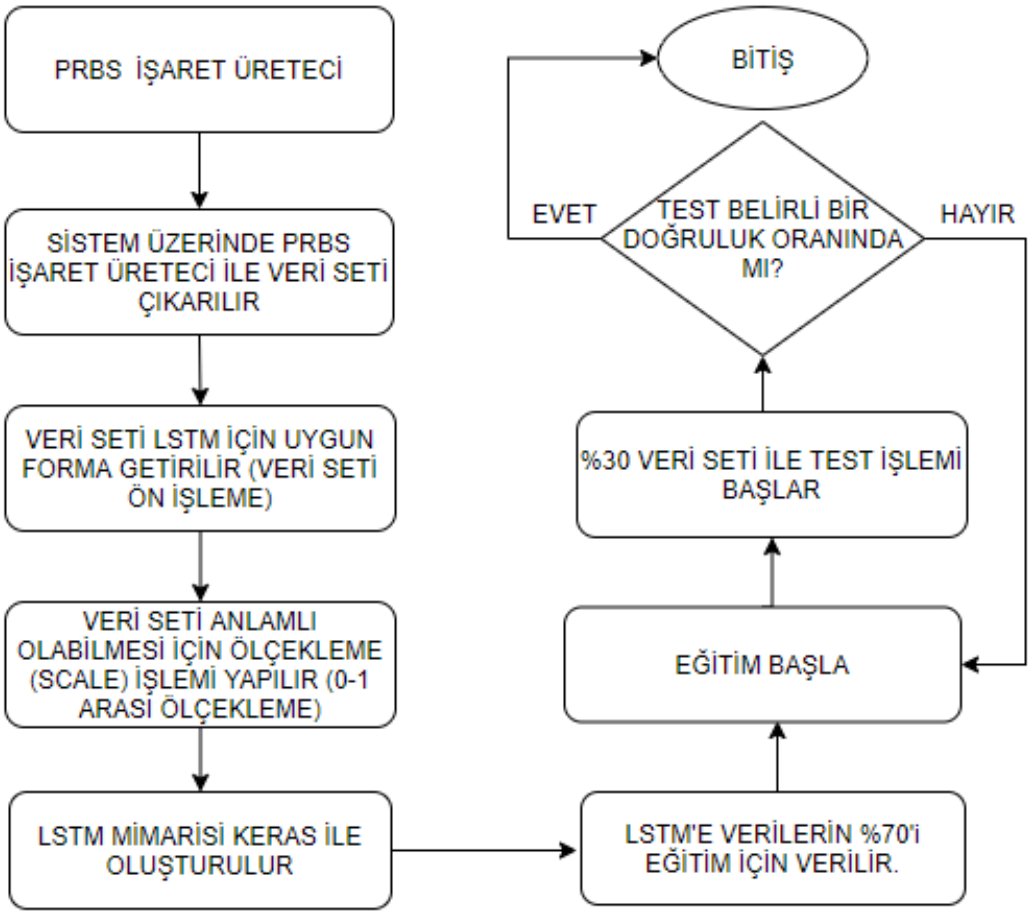
Şekil 4.7 : PRBS işaret üreteci algoritma akış diyagramı

İkinci akış diyagramında ise UKSB ile sistem tanımlama algoritması gösterilmiştir. Akış diyagramı içerisindeki alt akış diyagramı **PRBS İşaret Üreteci** kısmında açıklanan ve Şekil 4.7'de verilen akış diyagramıdır. Rassal işaret üretildikten sonra, üretilen işaret ile oluşturulan verilerin %70'i (3500 veri) ile eğitim, kalan %30'u (1500 veri) ile test işlemi uygulanmıştır. Test yapmak bu tip sistemlerde son derece önemlidir. Eğitim sonuçları iyi ancak test sonuçları kötü olan sonuçlardan sistemin öğrenilmediği, sadece eğitim setinin ezberlendiği anlaşılır.

Eğitim işlemi **çevrimdışı (offline)** gerçekleştirilmiş, yani önce veriler ile eğitim gerçekleştirilip eğitim tamamlandıktan sonra öğrenmiş yapının sistemin yerini alması

sağlanmıştır. Akış diyagramında da görüleceği gibi eğer test sonuçları belirli bir eşiğin altında kalırsa eğitim tekrarlanacaktır. Eşik değeri yazılımcı tarafından programa gömülebileceği gibi görsel olarak da belirlenebilir.

Şekil 4.8'de UKSB ile sistem tanımlama akış diyagramı gösterilmiştir. Bu akış diyagramlarına uyarak ilerleyen bölümlerde örnek sistemler üzerinde sistem tanımlama gerçekleştirilmiştir.

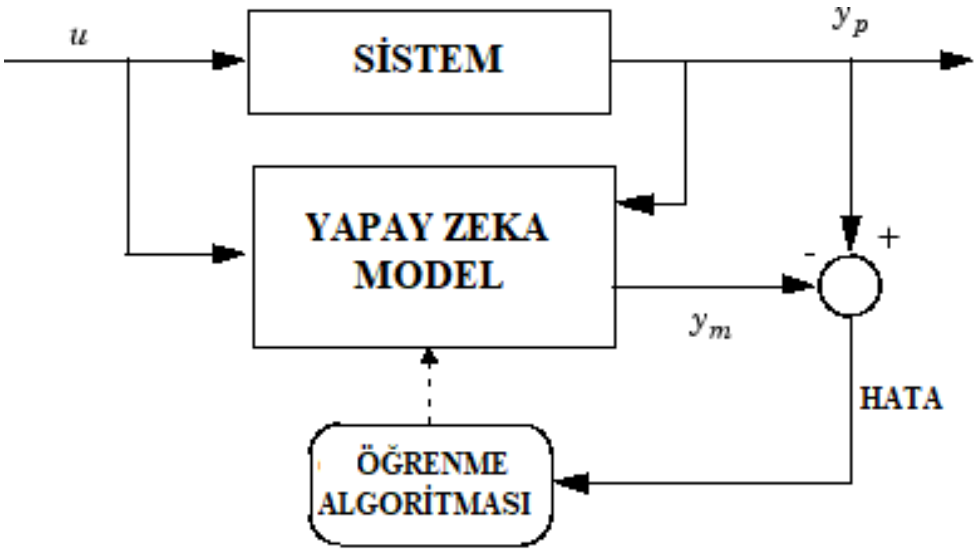


Şekil 4.8 : UKSB sistem tanımlama akış diyagramı

4.3 Örnek Sistemler Üzerinde Sistem Tanımlama Çıkışları

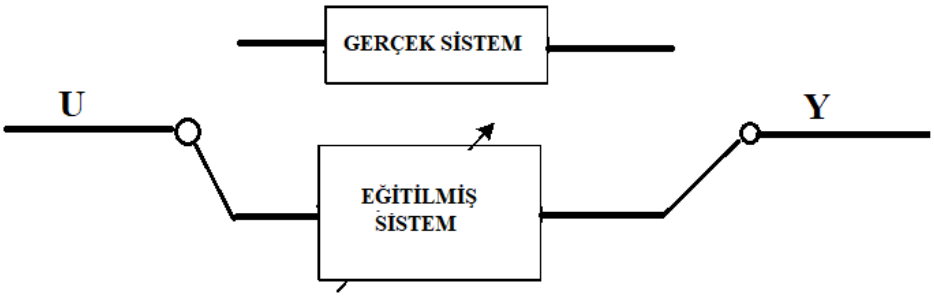
Sistem tanımlama kavramını blok diyagram şeklinde de gösterebiliriz. Şekil 4.9'da sistem tanımlama için genel bir blok diyagram gösterilmiştir. Blok diyagramdan anlaşılacağı gibi sistemden elde edilen veriler ile yapay zeka model eğitilmektedir. Öncelikle sistem verileri ve yapay zeka model çıkışları karşılaştırılarak bir hata değeri üretilir. Bu hata değeri öğrenme algoritmasına verilerek algoritmanın içeriğine uygun güncelleme değerleri üretilir. Hata oranı belirli bir değerin altına düştüğünde

güncelleme değerleri de azalmaya başlar ve algoritma durdurulur. Öğrenme işlemi ve test işlemleri tamamlandıktan sonra yapay zeka model, gerçek sistem yerine kullanılabilir. Çalışma içeriğine uygun bir şekilde yapay zeka model öğrenme algoritması olarak UKSB kullanılmıştır.



Şekil 4.9 : Yapay zeka tabanlı sistem tanımlama blok diyagram

Sistem tanımlamadaki asıl amaç, Şekil 4.10'da verilmiştir. Sistem tanımlama sonrasında Şekil 4.10'daki anahtar açılarak gerçek sistem yerini önceden eğitilmiş yapay zeka sisteme bırakır. Eğitilmiş yapay zeka sistem, gerçek sisteme yakın sonuçlar üretebilmektedir. Bu sayede sistemin giriş-çıkış verileri kullanılarak sistemin doğrusal olmayan bir modeli elde edilebilir ve gerçek sistem yerine kullanılabilir.



Şekil 4.10 : Anahtarlama prensibi

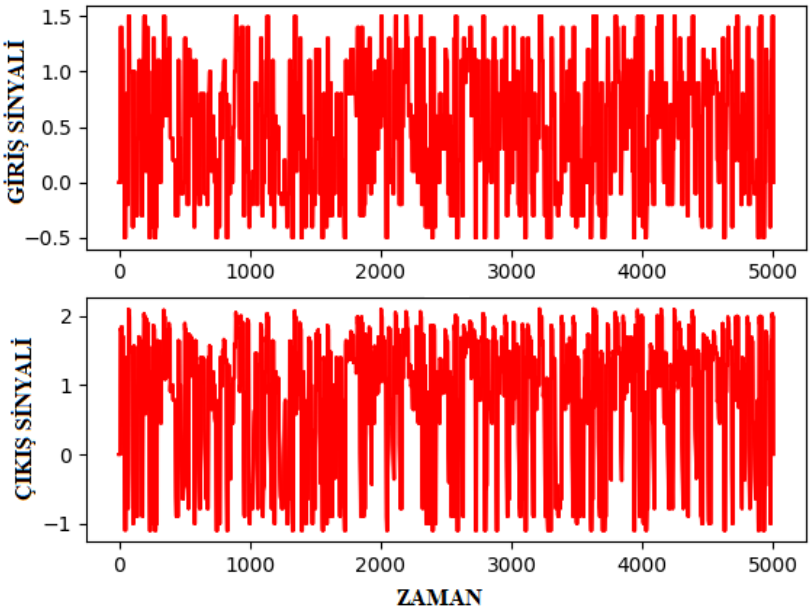
Bu kısımda iki adet örnek sistem için sistem tanımlama sonuçları incelenecektir. Her iki sistem için de eğitim ve test sonuçları verilmiştir. Ayrıca eğitim ve test için kabul gören başarı ve hata değerleri de hesaplanarak çizelgelerde gösterilmiştir. Bu değerlere bakarak eğitimin ve tahmin değerlerinin başarısı hakkında yorumlar yapılabilir.

4.3.1 Doğrusal olmayan test sistemi sonuçları

Örnek olarak kullanacağımız ilk sistemimiz doğrusal olmayan dinamiğe sahip bir sistemdir. Sistemin matematiksel ifadesi, dinamiği hatırlatmak amacıyla aşağıda tekrar verilmiştir:

$$y(k+1) = \frac{1.2(1-0.8e^{-0.1k})y(k)}{1+y^2(k)} + u(k) \tag{4.7}$$

Örnek sistemin matematiksel denklemine Şekil 4.11'deki giriş sinyali uygulandığında yine Şekil 4.11'deki çıkış sinyalleri elde edilir. Giriş işareti u(k)'ya uygulanır ve önceki y(k) çıkışları ile işleme dahil edilerek y(k+1) işareti elde edilir. y(k+1) işareti bir sonraki adımda y(k) yerine kullanılmıştır. İlk durum için y(k) = 0 alınmıştır.

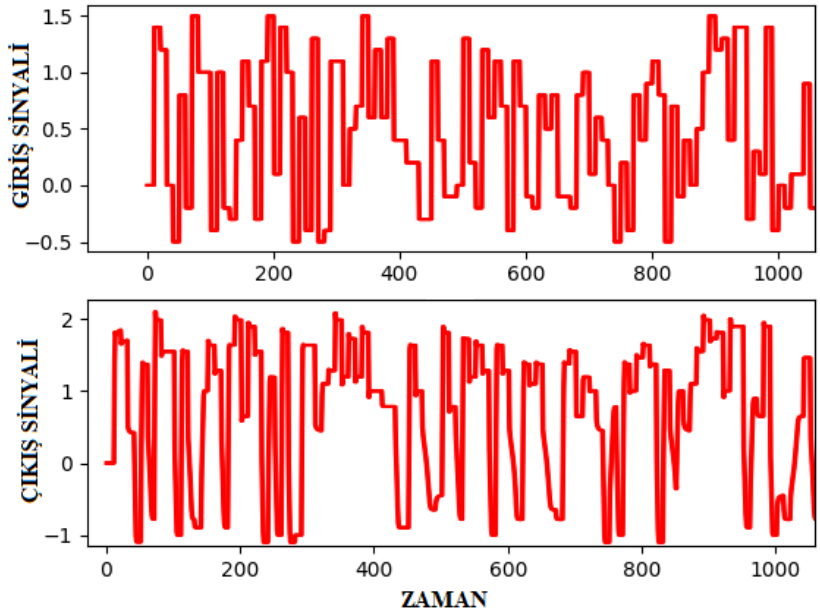


Şekil 4.11 : Sistem tanımlama girişleri ve çıkışları

Sisteme uygulanacak giriş-çıkış değerlerini daha yakından görmek işareti anlamak açısından yararlı olacaktır. Şekil 4.12'de giriş ve çıkış sinyallerinin 0 ile 1000 değerleri arasında yakınlaştırılmış halleri gösterilmiştir.

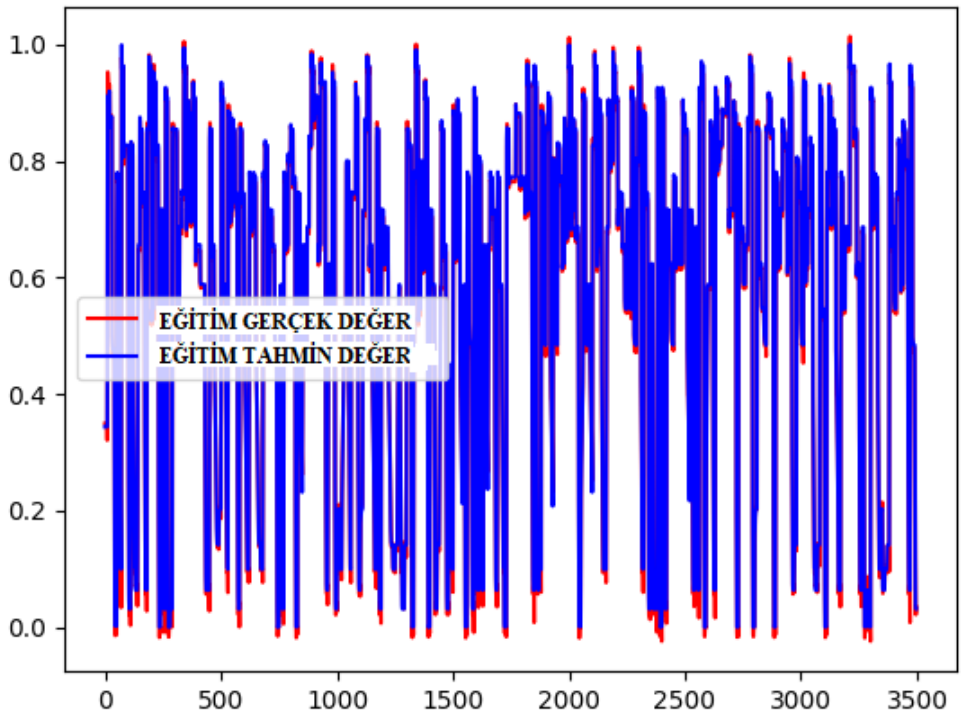
EĞİTİM SONUÇLARI:

UKSB tabanlı sistem tanımlamanın ilk adımında yukarıda elde edilen giriş ve çıkış sinyalleri, UKSB mimarisinin anlayabileceği veri formatına dönüştürülür. Bu işleme makine öğrenmesi literatüründe ***verinin ön işlenmesi*** adı verilir. Şekil 4.11'de toplanan giriş ve çıkış verilerinin %70'lik kısmı (3500 veri) eğitim amacıyla kullanılacak veriler olarak, kalan %30'luk kısmı (1500 veri) ise test amacıyla kullanılacak



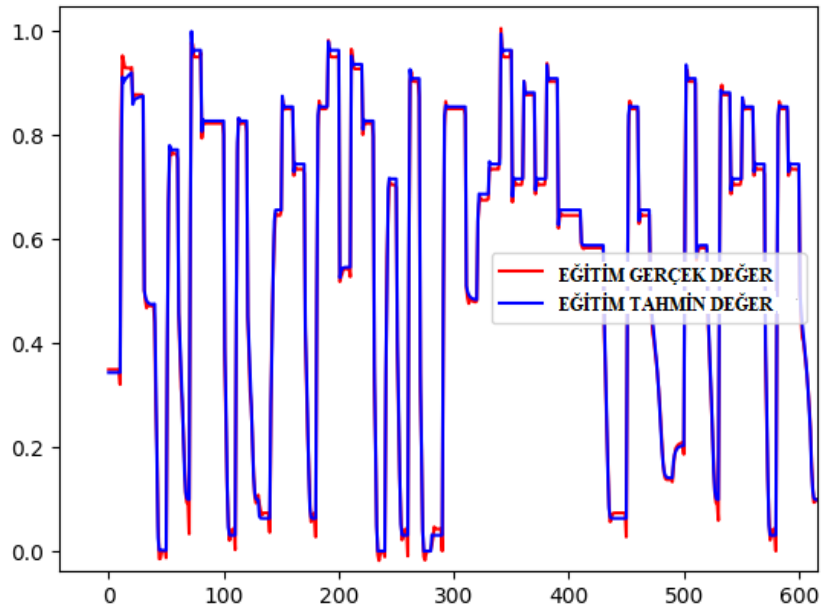
Şekil 4.12 : Sistem tanımlama girişleri ve çıkışlarının yakınlaştırılmış hali

veriler olarak ayrılmıştır. Ayrıca tüm veriler UKSB için uygun hale getirilmesi amacıyla 0 ile 1 arasında ölçeklenmiştir. Bu işlemi yapmanın amacı veri içerisindeki değerlerin birbirinden ayrık olmasının önüne geçerek algoritmanın veriler arasında ağırlık bağıntısı kurmamasını sağlamaktır. [35] Ölçekleme algoritması olarak ***min-max ölçekleme*** kullanılmıştır. Şekil 4.13'de eğitim sonrasında oluşan gerçek ve tahmin edilen değerlerin çıktıları gösterilmiştir.



Şekil 4.13 : Sistem tanımlama eğitim çıkışları

Sonuçları daha iyi görebilmek için Şekil 4.14'de sistem tanımlama çıkışları belirli bir aralık için yakınlaştırılmıştır. Eğitim sonuçları bu şekil üzerinden dikkatli incelenirse gerçek ve tahmin edilen değerler arasındaki yakınlık net olarak görülebilir. Bu çıkış iyiye işarettir ancak herşeyin mükemmel olduğu anlamına gelmez. UKSB sistemi ezberlemiş olabilir ve sadece %70'lik eğitim verisi için doğru sonuçlar üretiyor olabilir. Bu yanılgıdan kurtulmak için test işlemi gerçekleştirilmelidir.



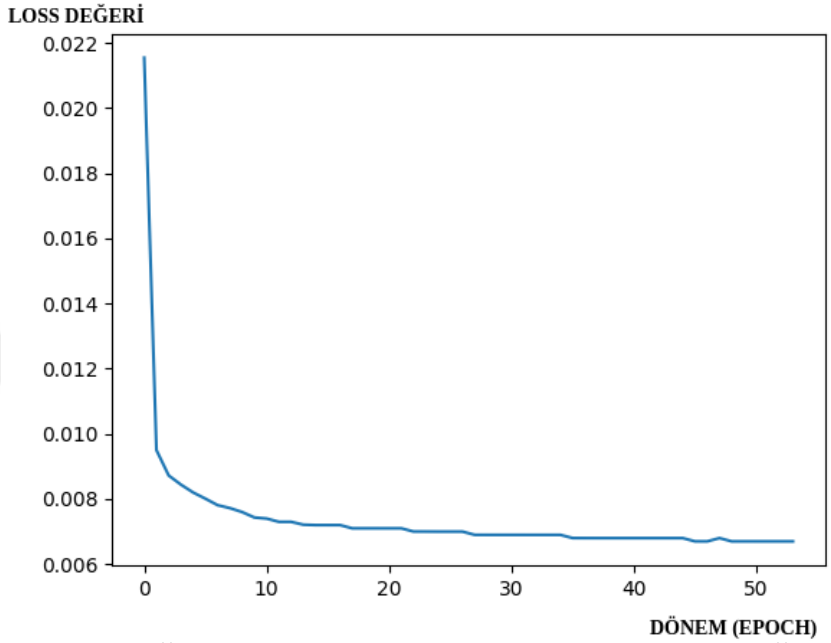
Şekil 4.14 : Sistem tanımlama eğitim çıkışlarının yakınlaştırılmış hali

Eğitim sonuçlarını daha matematiksel bir yöntem ile incelemek için belirli dönem (epoch) aşamalarındaki yitim değerleri Çizelge 4.1'de gösterilmiştir.

Çizelge 4.1 : Belirli dönemlerdeki yitim değerleri

Dönem (Epoch) Numarası	**yitim Değeri**
1	0.0215432
2	0.0095011
3	0x008722
10	0.007433
11	0.0074001
12	0.0073001
25	0.0070
26	0.0070
27	0.007001
28	0.0069006
49	0.0067
50	0.0067

Şekil 4.15 incelenirse örnek sistem için yitim fonksiyonu değişimlerininde bu şekilde olduğu gözlemlenebilir. Başarılı eğitimlerde yitim fonksiyonunun eğitim başlarında yüksek olması, hemen ardından ani bir düşüş göstermesi ve devamında da sabit gitmesi beklenmektedir. [35] Eğitim sonuçlarının başarısını matematiksel



Şekil 4.15 : Eğitim sırasında dönemlere (epoch) göre yitim değişimi

olarak anlamlandırabilmek için Çizelge 4.2'de *ortalama kesin hata (mean absolute error)*, *ortalama karesel hata (mean square error)*, *medyan kesin hata (median absolute error)*, *açıklanan varyans değer (explained variance score)* ve *R2 değer* sonuçlarıda aşağıdaki çizelgede verilmiştir. Hata değerlerinin düşük değerler olması ve buna karşılık başarı değerlerinin 1'e oldukça yakın olması eğitimin başarısını göstermektedir.

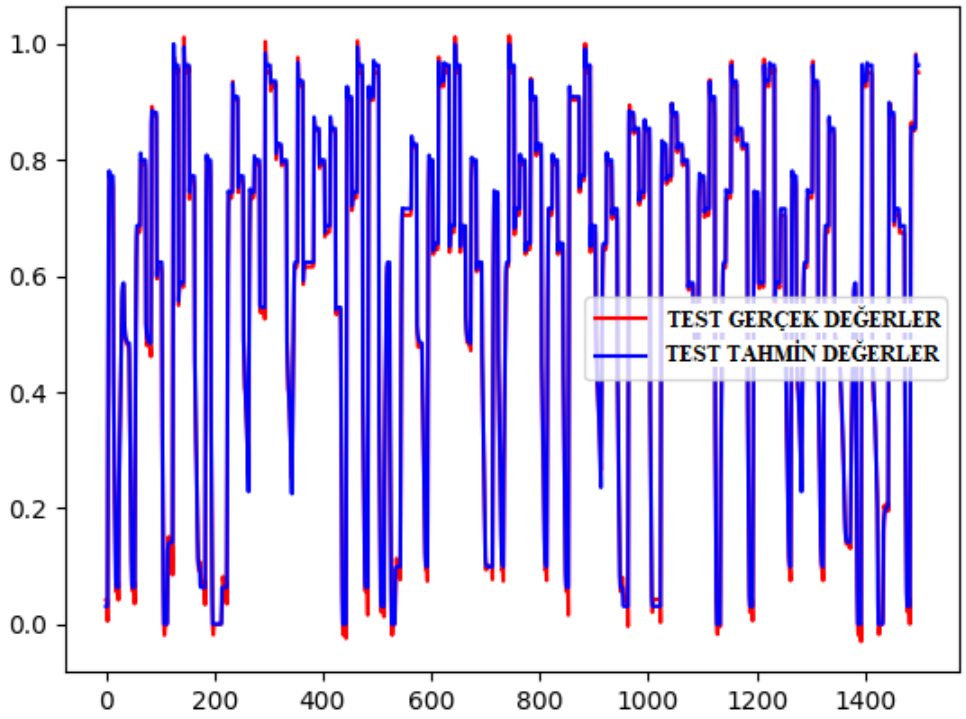
Çizelge 4.2 : Eğitim için belirli başarı ölçüt değerleri

Başarı Kriteri	**Değer**
Ortalama Kesin Hata	0.005867
Ortalama Karesel Hata	5.592064
Medyan Kesin Hata	0.0054014
Açıklanan Varyans Değer	0.99929
R2 Değer	0.999435

Test sistemini öğrenmek için gerekli toplam süre yaklaşık olarak 406.45 saniye civarındadır. Bu süre yaklaşık 6-7 dakika arasında bir süredir. Eğitim süresi işlemci kabiliyetine göre değişim gösterebileceğinden farklı mimarilerde farklı değerler elde edilebilmektedir. Sonuçlarda hatalar oldukça küçük başarı değerleri ise oldukça yüksek değerlerdedir. Gerçek anlamda başarı kriterini anlayabilmek için test sonuçlarınında başarılı olması beklenir. Bunun nedeni test için oluşturulmuş veriyi sistemin daha önceden görmemesidir. Daha önceden görmediği veriler üzerinde yapacağı tahminler sistem başarısı için daha doğru bir ölçümdür.

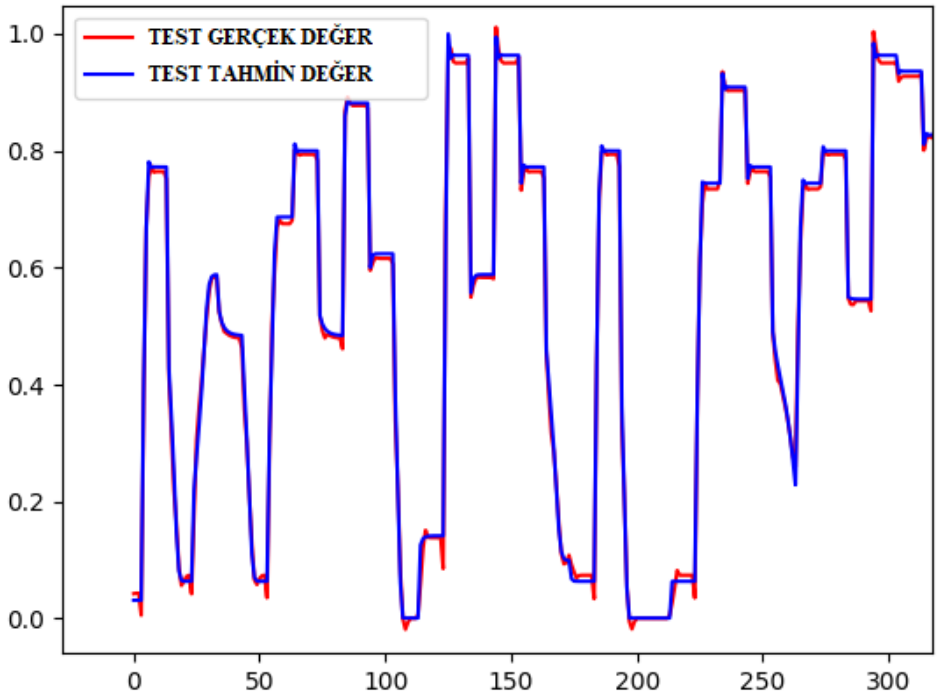
TEST SONUÇLARI:

Eğitimin başarısını tam anlamıyla kanıtlamak için test sonuçlarına da göz atmak gerekir. Genel olarak eğitimin başarılı olması sistemin tam anlamıyla öğrenildiği anlamına gelmez. Sistemin öğrenilme başarısını test sonuçları gösterecektir. Şekil 4.16'de örnek sistemin test sonuçları gösterilmiştir.



Şekil 4.16 : Sistem tanımlama test çıkışları

Eğitim sonuçlarının başarısını daha yakından gözlemek için Şekil 4.17'de belirli bir aralık için yakınlaştırma uygulanmıştır. Bu şekil üzerinden tahmin sonuçlarının gerçek sonuçları oldukça yakın takip ettiği gözlemlenebilir.



Şekil 4.17 : Sistem tanımlama test çıkışlarının yakınlaştırılmış hali

Test sonuçlarının başarısını matematiksel olarak anlamlandırabilmek için Çizelge 4.3'de *ortalama kesin hata*, *ortalama karesel hata*, *medyan kesin hata*, *açıklanan varyans değer* ve *R2 değer* sonuçları da aşağıdaki çizelgede verilmiştir. Hata değerlerinin düşük olması ve buna karşılık başarı değerlerinin 1'e oldukça yakın olması eğitimin başarısını göstermektedir.

Test sonuçları ve eğitim sonuçları hata ve değerlerinin birbirlerine yakın olması eğitimin başarılı olduğunun bir göstergesidir. Buradan anlaşılacağı üzere test sistemi için oluşturulan sistem tanımlama modeli gerçek sistem yerine kullanılabilir durumdadır.

Çizelge 4.3 : Test için belirli başarı ölçüt değerleri

Başarı Kriteri	**Değer**
Ortalama Kesin Hata	0.0058556
Ortalama Karesel Hata	5.730182
Medyan Kesin Hata	0.0053893
Açıklanan Varyans Değer	0.999345
R2 Değer	0.9993332

4.3.2 Doğrusal olmayan CSTR sistemi sonuçları

Örnek olarak kullanacağımız ikinci sistem yine doğrusal olmayan bir dinamiğe sahip olan CSTR sistemdir. Sistemin matematiksel ifadesi, dinamiği hatırlatmak amacıyla aşağıda tekrar verilmiştir. Çalışma kapsamında $Da_1 = 3$, $Da_2 = 0.5$, $Da_3 = 1$, $u_{min} = 0$ ve $u_{max} = 1$ olarak seçilmiştir. [14]

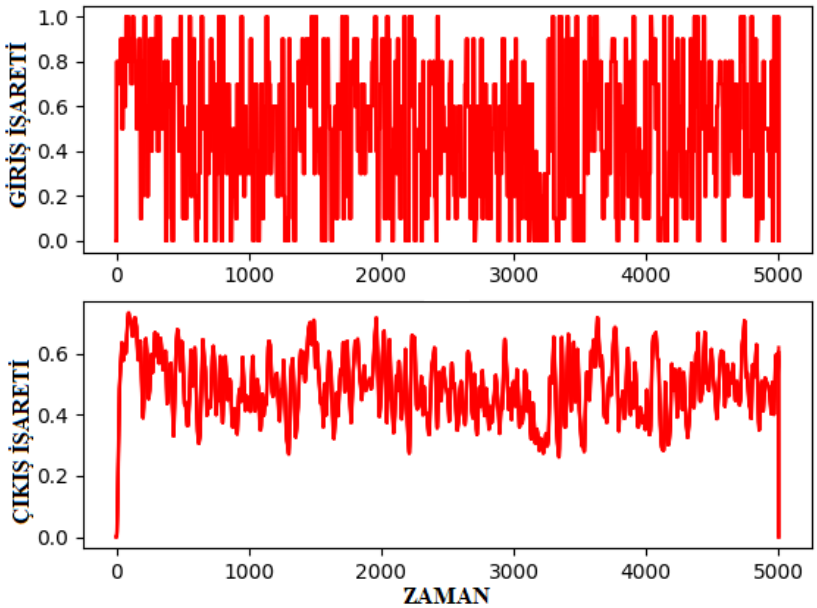
$$\dot{X}_1(t) = 1 - X_1(t) - Da_1 X_1(t) + Da_2 X_2^2(t) \tag{4.8}$$

$$\dot{X}_2(t) = -X_2(t) + Da_1 X_1(t) - Da_2 X_2^2(t) - Da_3 d_2(t) X_2^2(t) + u(t) \tag{4.9}$$

$$\dot{X}_3(t) = -X_3(t) + Da_3 d_2(t) X_2^2(t) \tag{4.10}$$

$$u_{min} \leq u \leq u_{max} \tag{4.11}$$

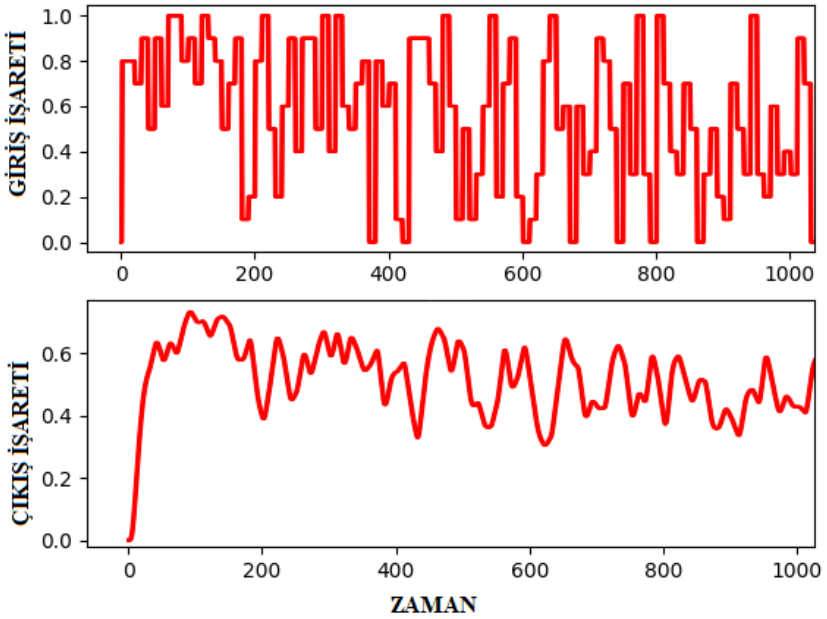
CSTR sistemin matematiksel denklemine Şekil 4.18'deki giriş sinyali uygulandığında yine Şekil 4.18'deki çıkış sinyalleri elde edilir. Giriş işareti u(k)'ya uygulanır ve önceki y(k) çıkışları ile işleme dahil edilerek y(k+1) işareti elde edilir. y(k+1) işareti bir sonraki adımda y(k) yerine kullanılmıştır. İlk durum için y(k) = 0 alınmıştır.



Şekil 4.18 : Sistem tanımlama girişleri ve çıkışları

Sisteme uygulanacak giriş-çıkış değerlerini daha yakından görmek girişleri ve çıkışları anlamak açısından yararlı olacaktır. Şekil 4.19'da giriş ve çıkış sinyallerinin 0 ile 1000 değerleri arasında yakınlaştırılmış halleri gösterilmiştir. Bu kısımdan

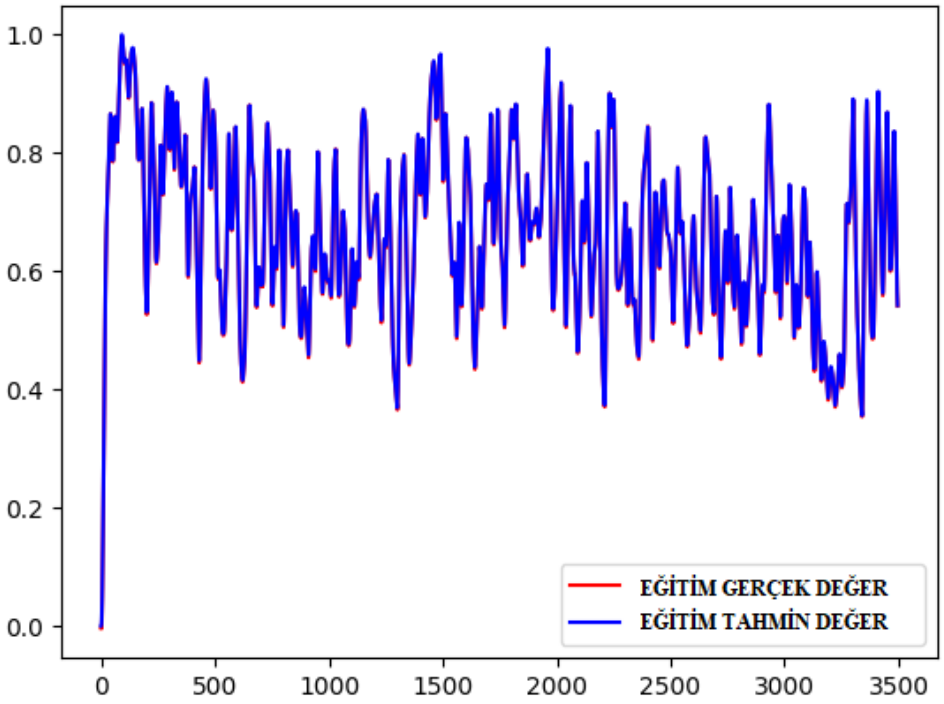
oluşturulan veriler UKSB için uygun formata getirilerek 0 ve 1 arasında ölçekleme (scaling) işlemine tabi tutulmalıdır. Bu işlemin önemi UKSB sisteminin tüm verileri belirli bir skalada görmesi ve bir veriye diğerinden daha fazla ağırlık vermemesinin sağlanmasıdır. Oluşturulan veri seti UKSB içerikli yapay zeka ağı içerisine verilir ve eğitim işlemi başlatılır.



Şekil 4.19 : Sistem tanımlama girişleri ve çıkışlarının yakınlaştırılmış hali

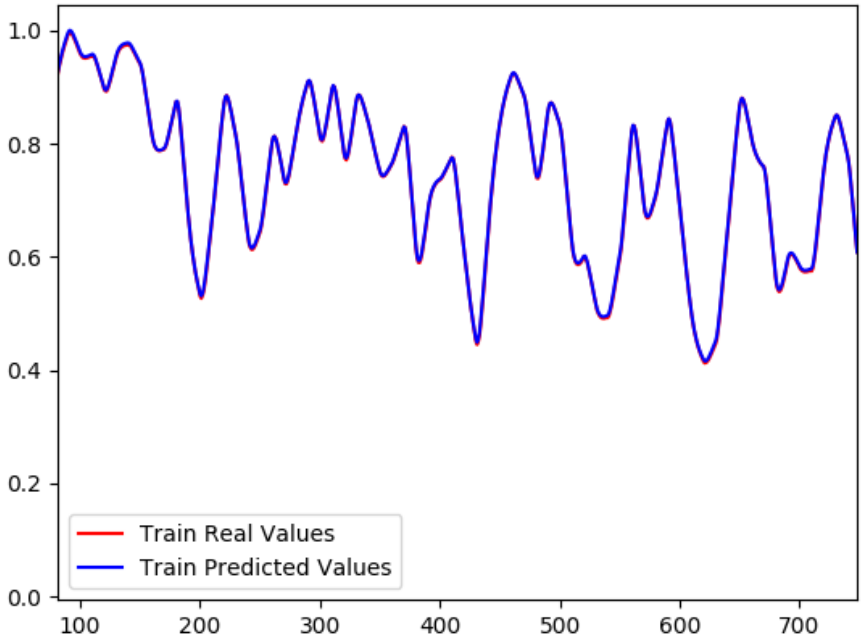
CSTR sisteminin giriş ve çıkışları oluşturularak bu giriş çıkışlar veri formatına dönüştürülmüştür. Şekil 4.18'de toplanan giriş ve çıkış verilerinin %70'lik kısmı (3500 veri) eğitim amacıyla kullanılacak veriler olarak, kalan %30'luk kısmı (1500 veri) ise test amacıyla kullanılacak veriler olarak ayrılmıştır. Ayrıca tüm veriler UKSB için uygun hale getirilmesi amacıyla 0 ile 1 arasında ölçeklenmiştir. Bu işlemi yapmanın amacı veri içerisindeki değerlerin birbirinden ayrık olmasının önüne geçerek algoritmanın veriler arasında ağırlık bağıntısı kurmamasını sağlamaktır. [35] Ölçekleme algoritması olarak ***min-max ölçekleme*** kullanılmıştır. Şekil 4.20'de eğitim sonrasında oluşan gerçek ve tahmin edilen değerlerin çıktıları gösterilmiştir. Eğitim başarısını değerlendirebilmek için eğitim sonuçları tam anlamıyla yeterli değildir, test sonuçları da incelenmelidir. Ancak eğitim sonuçları incelenirse tahmin değerlerinin gerçek değerlere yakın olduğu görülebilir. Bu durum modellemenin ilk adımının başarılı olduğuna bir işarettir.

Sonuçları daha iyi görebilmek için Şekil 4.21'de sistem tanımlama çıkışları belirli bir aralık için yakınlaştırılmıştır. Eğitim sonuçları bu şekil üzerinden dikkatli incelenirse



Şekil 4.20 : Sistem tanımlama eğitim çıkışları

gerçek ve tahmin edilen değerler arasındaki yakınlık açık bir şekilde görülebilir. Bu çıkış iyiye işarettir ancak herşeyin mükemmel olduğu anlamına gelmez. UKSB sistemi ezberlemiş olabilir ve sadece %70'lik eğitim verisi için doğru sonuçlar üretiyor olabilir. Bu yanılgıdan kurtulmak için test işlemi gerçekleştirilmelidir.



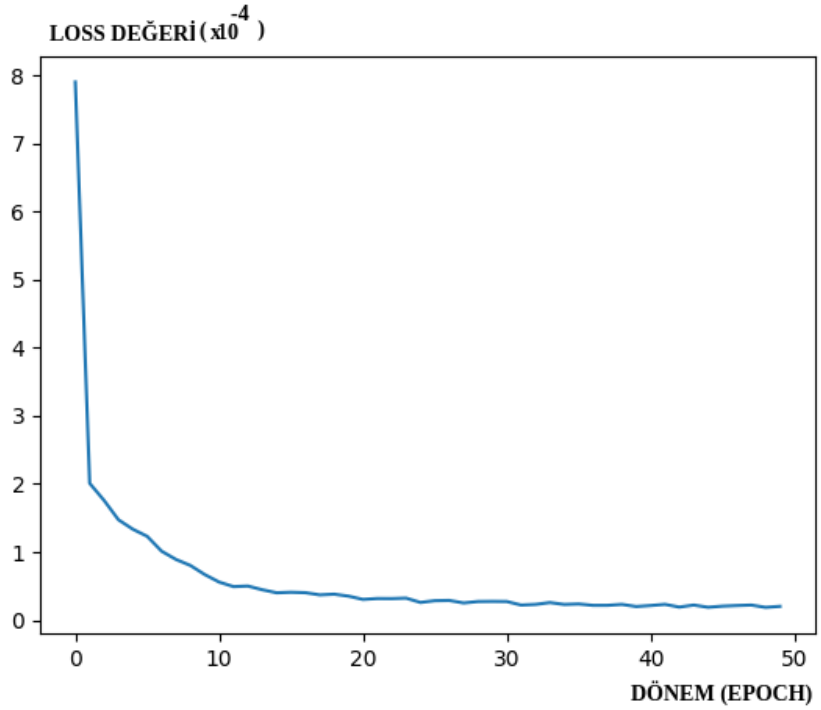
Şekil 4.21 : Sistem tanımlama eğitim çıkışlarının yakınlaştırılmış hali

Eğitim sonuçlarını daha matematiksel bir yöntem ile incelemek için belirli dönem (epoch) aşamalarındaki yitim değerleri Çizelge 4.4'de gösterilmiştir. Başarılı eğitimlerde yitim fonksiyonunun eğitim başlarında yüksek olması, hemen ardından ani bir düşüş göstermesi ve devamında da sabit gitmesi beklenmektedir. [35]

Çizelge 4.4 : Belirli dönemlerdeki yitim değerleri

Dönem (Epoch) Numarası	**Yitim Değeri** (10^{-4})
1	79.04
2	2.009
3	1.7582
10	0.6688
11	0.558541
12	0.49187
25	0.25905
26	0.285
27	0.2886
28	0.25179
49	0.18704
50	0.20179

Şekil 4.22 incelenirse CSTR sistemi için yitim fonksiyonu değişimlerinin de bu şekilde olduğu gözlemlenebilir. Yitim fonksiyonunun değişimi olması gerektiği gibidir.



Şekil 4.22 : Eğitim sırasında dönemlere (epoch) göre yitim değişimi

Eğitim sonuçlarının başarısını matematiksel olarak anlamlandırabilmek için Çizelge 4.5'da *ortalama kesin hata*, *ortalama karesel hata*, *medyan kesin hata*, *açıklanan varyans değer* ve *R2 değer* sonuçları da aşağıdaki çizelgede verilmiştir. Hata değerlerinin düşük olması ve buna karşılık başarı değerlerinin 100'e oldukça yakın olması eğitimin başarısını göstermektedir.

CSTR sistemini öğrenmek için gerekli toplam süre yaklaşık olarak 404.672 saniye civarındadır. Bu süre yaklaşık 6-7 dakika arasında bir süredir. Eğitim süresi işlemci kabiliyetine göre değişim gösterebileceğinden farklı mimarilerde farklı değerler elde edilebilir. Sonuçlarda hatalar oldukça küçük başarı değerleri ise oldukça yüksek değerlerdedir. Gerçek anlamda başarı kriterini anlayabilmek için test sonuçlarının da başarılı olması beklenir. Daha önceden görmediği veriler üzerinde yapacağı tahminler sistem başarısı için daha doğru bir ölçümdür.

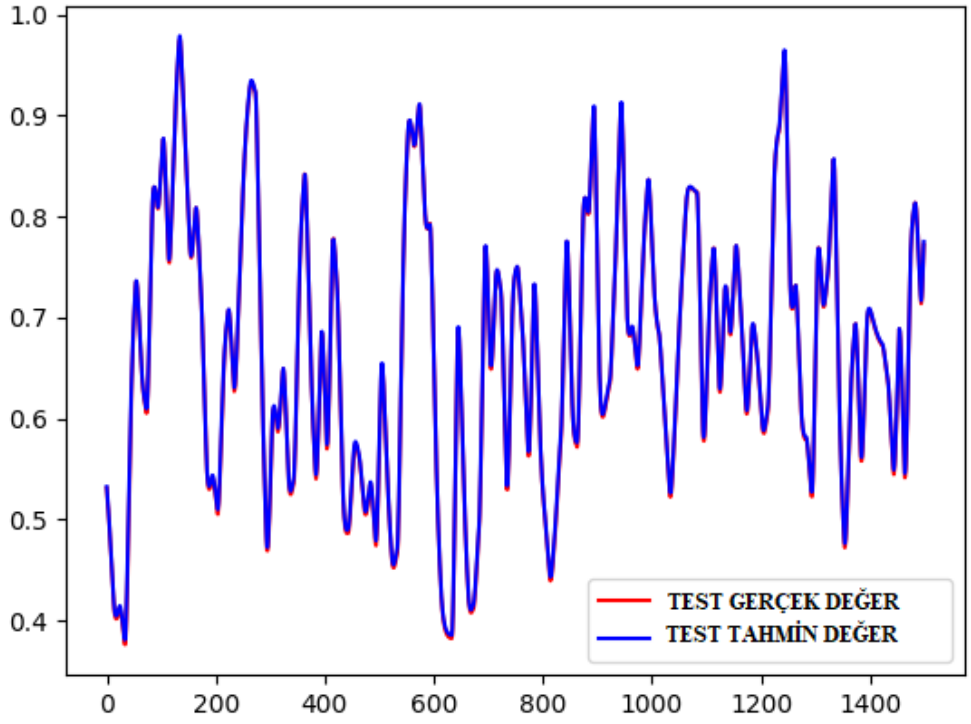
Çizelge 4.5 : Eğitim için belirli başarı ölçüt değerleri

Başarı Kriteri	**Değer**
Ortalama Kesin Hata	0.002369
Ortalama Karesel Hata	6.6870382
Medyan Kesin Hata	0.002246417
Açıklanan Varyans Değer	0.999960
R2 Değer	0.999981

TEST SONUÇLARI:

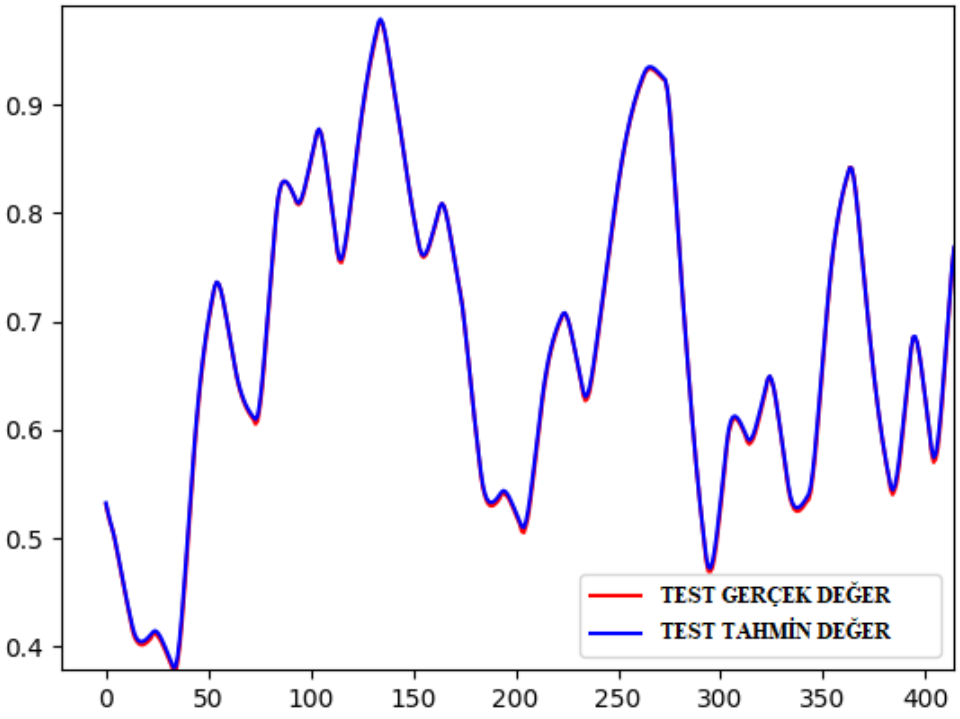
Eğitimin başarısını tam anlamıyla kanıtlamak için test sonuçlarına da göz atmak gerekir. Genel olarak eğitimin başarılı olması sistemin tam anlamıyla öğrenildiği anlamına gelmez. Sistemin öğrenilme başarısını test sonuçları gösterecektir. Şekil 4.23'de CSTR sisteminin test sonuçları gösterilmiştir.

Eğitim sonuçlarının başarısını daha yakından gözlemek için Şekil 4.24'de belirli bir aralık için yakınlaştırma uygulanmıştır. Bu şekil üzerinden tahmin sonuçlarının gerçek sonuçları oldukça yakın takip ettiği gözlemlenebilir. Bu durumda UKSB tabanlı sistem tanımlamanın başarılı sonuçlar ürettiği ortaya çıkmıştır. Her iki örnek sistem üzerinde de eğitim ve test sonuçları gerçeğe oldukça yakın değerler üretmektedir. Bu kısımdan sonraki aşamada örnek sistemler yerine UKSB tabanlı eğitilmiş sistem kullanılabilir.



Şekil 4.23 : Sistem tanımlama test çıkışları

Çalışma kapsamında sıradaki bölümde UKSB tabanlı eğitilmiş sistem yine UKSB tabanlı bir kontrolör ile kontrol edilerek referans bir işaretin takibi gerçekleştirilecektir.



Şekil 4.24 : Sistem tanımlama test çıkışlarının yakınlaştırılmış hali

Test sonuçlarının başarısını matematiksel olarak anlamlandırabilmek için Çizelge 4.6'da *ortalama kesin hata*, *ortalama karesel hata*, *medyan kesin hata*, *açıklanan*

varyans değer ve *R2 değer* sonuçları da aşağıdaki çizelgede verilmiştir. Hata değerlerinin düşük olması ve buna karşılık başarı değerlerinin 1'e oldukça yakın olması eğitimin başarısını göstermektedir.

Test sonuçları ve eğitim sonuçları hata ve değerlerinin birbirlerine yakın olması eğitimin başarılı olduğunun bir göstergesidir. Buradan anlaşılacağı üzere test sistemi için oluşturulan sistem tanımlama modeli gerçek sistem yerine kullanılabilir durumdadır.

Çizelge 4.6 : Test için belirli başarı ölçüt değerleri

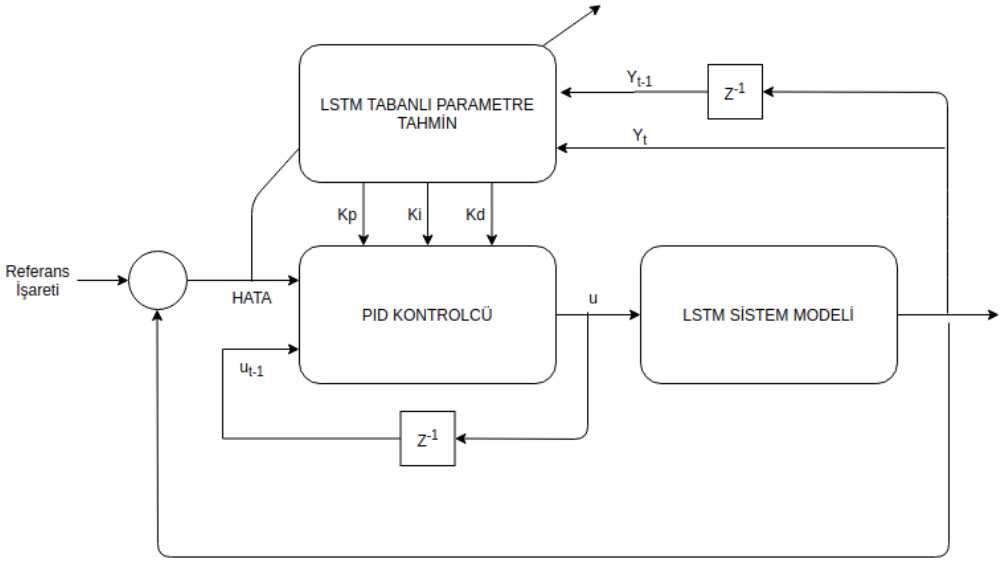
Başarı Kriteri	**Değer**
Ortalama Kesin Hata	0.0023907
Ortalama Karesel Hata	7.027309
Medyan Kesin Hata	0.00224684
Açıklanan Varyans Değer	0.999959
R2 Değer	0.999788

Her iki örnek sistem üzerinde de UKSB mekanizmasının başarısı sonuçlarla kanıtlanmıştır. Bu durumda sistemlerden çıkarılan uygun veriler ile UKSB kullanılarak sistemlerin matematiksel modelleri çıkarılabilir ve bu modeller gerçek modellere oldukça yakın sonuçlar üretebilmektedir.

5. UKSB TABANLI UYARLAMALI KONTROL

Bir önceki bölümde gerçek sistemden elde edilen veriler kullanarak UKSB tabanlı sistem tanımlama işlemi gerçekleştirilmişti. Sıradaki adımda ise oluşturulan UKSB tabanlı sistem modeli için yine UKSB tabanlı kendinden ayarlamalı kontrolör tasarlanmış ve uyarlamalı kontrol gerçekleştirilmiştir. Kendinden ayarlamalı kontrolör için özel bir UKSB mimarisi tasarlanmış ve bu mimari içinde kontrol mantığına uygun olarak ağırlıkların güncellenmesi sağlanmıştır. Bu bölümde amaç UKSB tabanlı sistem tanımlama ile modeli oluşturulan örnek sistemlerin, UKSB tabanlı bir kontrolör ile referans bir işaretin takibinin sağlanmasıdır.

Şekil 5.1'de uygulanacak mimarinin blok diyagramı gösterilmiştir. Mimaride kullanılan sistem modeli bir önceki bölümde çevrimdışı olarak elde ettiğimiz UKSB modelidir. Bu kısımda unutulmaması gereken nokta elimizde bir sistem matematiksel modeli olmadığı bunun yerine sistemden elde edilen verilerin olduğu mantığıdır.



Şekil 5.1 : UKSB tabanlı kontrol blok diyagramı

Blok diyagramda gösterildiği gibi ***UKSB Tabanlı Parametre Tahmin***'i, sistem için uyarlamalı olarak K_p, K_i ve K_d değerlerini hesaplar ve bu değerler ***kontrolör***

girişine gelerek sistemin referans sinyali takip edebilmesi için gerekli olan işareti üretir. Bölümde *UKSB Tabanlı Parametre Tahmin* ve *kontrolör* bloklarının içlerindeki fonksiyonlar anlatılmıştır. Ardından sistem tanımlama bölümünde kullanılan iki örnek sistem üzerinde test işlemleri gerçekleştirilmiştir. İlk olarak PID kontrolör kısmı açıklanacaktır.

5.1 PID kontrolör

PID kontrolör, UKSB Tabanlı Parametre Tahmin bölümüne göre daha kolaydır ve hesaplamaların içerisinde yer alır. Bundan dolayı ilk olarak bu kısım incelenmiştir. Pek çok kontrol uygulasında kolaylığı ve yüksek performansı sebebiyle PID kontrolör tercih edilir. Uygulamanın bu kısmında da kontrolör olarak PID tercih edilmiştir.

PID kontrolör bloğu genel olarak kendisine UKSB Tabanlı Parametre Tahmin bloğundan gelen K_p, K_i, K_d değerlerini ve bir önceki kontrol girdisi değerini kullanarak bir sonraki kontrol girdisi değerini üretir. Sayısal bir sistem üzerinden testler gerçekleştirildiği için ***ayrık zaman PID*** formülleri kullanılmıştır. Bu durumda oransal, türev ve integral işlemleri hatalar cinsinden oluşturulur. [47] Aşağıda P, I ve D için hata değerleri verilmiştir:

$$e_p(k) = e(k) - e(k-1) \tag{5.1}$$

$$e_i(k) = e(k) + e(k-1) \tag{5.2}$$

$$e_d(k) = e(k) - 2e(k-1) + e(k-2) \tag{5.3}$$

Bu durumda sistem için yeni kontrol girdisi aşağıdaki şekilde üretilmektedir. İlk değer olarak K_p, K_i, K_d değerleri 0 ve 1 arasında rassal değerler olarak atanır.

$$u(k) = u(k-1) + K_p e_p(t) + K_i e_i(t) + K_d e_d(t) \tag{5.4}$$

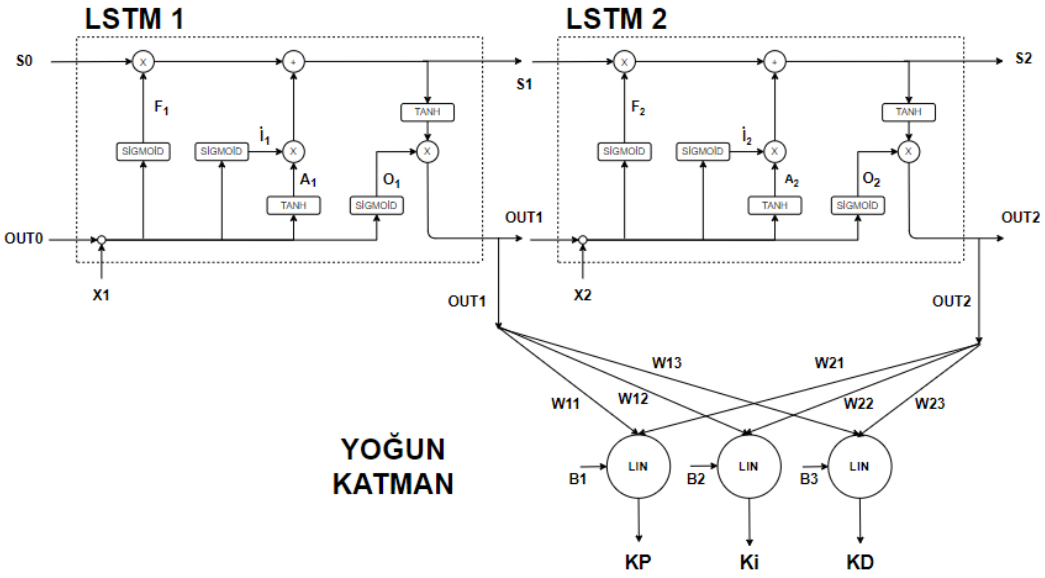
Bu bloğun yaptığı genel görev yukarıdaki formülü işleterek yeni kontrol girdilerini hesaplamaktır. Kendisine UKSB tarafından verilen K_p, K_i ve K_d değerilerinin yanı sıra bir önceki girdi değerini de kullanarak yeni kontrol girdisi değerini oluşturur. Çalışmada asıl görev bir önceki blok diyagram olan ***UKSB Tabanlı Parametre***

Tahmini bloğundadır. Bu blok içerisinde P, I ve D katsayıları uyarlamalı olarak üretilerek örnek sistemin bir referans işareti takip edebilmesi amaçlanır.

5.2 UKSB Tabanlı Parametre Tahmini

UKSB Tabanlı Parametre Tahmin bloğu, uyarlamalı olarak kendisine verilen girişleri kullanarak sistemin referans sinyali takip edebilmesi için K_p, K_i ve K_d değerlerini üretmektedir. Blok için UKSB hücreleri kullanılmış ve çıkarımlar sistem hatası baz alınarak yapılmıştır. Dolayısıyla blok eğitiminde kullanılan hata, referans işareti ve sistem çıkışı arasındaki farktır.

Öncelikle parametre tahmini için kuracağımız UKSB yapısını oluşturmamız gerekmektedir. Parametre tahmin çıkışlarında K_p, K_i ve K_d olmak üzere üç adet sayısal değer olacağından mimarinin sonunda 3 adet yoğun katman kullanılması gerekir. Yoğun katmanların önüne ise karmaşıklığın artmaması için 2 adet UKSB hücresi konulmuştur. Hücrelerden biri girişine sistem çıkışı y_t değeri, diğeri ise bir önceki sistem çıkışı y_{t-1} değerini almıştır. Şekil 5.2'de UKSB Tabanlı Parametre Tahmin bloğu için kullanılan makine öğrenmesi mimarisi gösterilmiştir.



Şekil 5.2 : UKSB tabanlı paremetre tahmini mimarisi

Makine öğrenmesi mimarisinin içerisindeki güncellemeler için formüllerin çıkarılması gerekmektedir. Ancak bu kısımda güncellemeler klasik yöntemden biraz daha farklı gerçekleştirilecektir. Bunun nedeni artık hata fonksiyonunun sistem çıkışı ve referans işaret arasındaki fark olarak tanımlanmasıdır. Pek çok uyarlamalı uygulamada amaç

sistem çıkışının referans sinyali olabildiğince yakın takip edebilmesidir. Dolayısıyla hata kavramı bu takibin ne kadar başarılı yapılabildiğidir.

Öncelikle ileri faz formülleri çıkarılmalıdır. Ardından ileri faz için bulunan hata geriye yayılarak ağırlık güncelleme değerleri bulunacaktır. Şekil 5.2'ye uygun olarak ileri faz aşağıdaki gibidir.

UKSB 1:

$$F_1 = \sigma(W_{F1}X_1 + U_{F1}Out_0) \tag{5.5}$$

$$I_1 = \sigma(W_{I1}X_1 + U_{I1}Out_0) \tag{5.6}$$

$$A_1 = tanh(W_{A1}X_1 + U_{A1}Out_0) \tag{5.7}$$

$$O_1 = \sigma(W_{O1}X_1 + U_{O1}Out_0) \tag{5.8}$$

$$S_1 = S_0F_1 + I_1A_1 \tag{5.9}$$

$$Out_1 = O_1tanh(S_1) \tag{5.10}$$

UKSB 2:

$$F_2 = \sigma(W_{F2}X_2 + U_{F2}Out_1) \tag{5.11}$$

$$I_2 = \sigma(W_{I2}X_2 + U_{I2}Out_1) \tag{5.12}$$

$$A_2 = tanh(W_{A2}X_2 + U_{A2}Out_1) \tag{5.13}$$

$$O_2 = \sigma(W_{O2}X_2 + U_{O2}Out_1) \tag{5.14}$$

$$S_2 = S_1F_2 + I_2A_2 \tag{5.15}$$

$$Out_2 = O_2tanh(S_2) \tag{5.16}$$

Yoğun Katman:

$$K_P = (Out_1 W_{11} + Out_2 W_{21} + B_1) \tag{5.17}$$

$$K_i = (Out_1 W_{12} + Out_2 W_{22} + B_2) \tag{5.18}$$

$$K_D = (Out_1 W_{13} + Out_2 W_{23} + B_3) \tag{5.19}$$

İleri faz yukarıdaki formüller ile hesaplandıktan sonra geri faz aşaması için hata fonksiyonunun belirlenmesi gerekir. Kontrol yapısında hata fonksiyonu sistem çıkışının referans sinyali ne kadar başarılı takip ettiği olduğundan aşağıdaki gibi bir hata fonksiyonu oluşturulabilir.

$$HATA = E = \frac{1}{2}(y_{ref} - y_s)^2 \tag{5.20}$$

Amaç bu hatayı geriye doğru yayarak güncelleme değerlerini hesaplamaktır. Bu durumda çıkışa en yakındaki ağırlıklardan başlayarak güncelleme değerleri hesaplanmıştır.

W_{11} için güncelleme:

$$\frac{\partial E}{\partial W_{11}} = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_p}\frac{\partial K_p}{\partial W_{11}} = (y_{ref} - y_s).\frac{\partial y_s}{\partial u}.e_p.Out_1 \tag{5.21}$$

Burada $\frac{\partial y_s}{\partial u}$ ifadesi ***Jacobi Ifadesi*** olarak geçer. Bu kısımda elimizde bir sistem denklemi olmadığından nümerik olarak hesaplama yoluna gidilebilir. Bir önceki bölümde oluşturduğumuz UKSB tabanlı sistem tanımlama bizim için sistemin bir sonraki değerini üretebildiğinden Jacobi ifadesi, aşağıdaki şekilde yazılabilmektedir. Burada y_{k+1} UKSB tabanlı sistem modelinden elde edilmiştir.

$$Jacobii fadesi = J = \frac{\partial y_s}{\partial u} = \frac{(y_{k+1} - y_k)}{(u_{k+1} - u_k)} \tag{5.22}$$

W_{21} için güncelleme:

$$\frac{\partial E}{\partial W_{21}} = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_p}\frac{\partial K_p}{\partial W_{21}} = (y_{ref} - y_s).J.e_p.Out_2 \tag{5.23}$$

W_{12} için güncelleme:

$$\frac{\partial E}{\partial W_{12}} = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i}\frac{\partial K_i}{\partial W_{12}} = (y_{ref} - y_s).J.e_i.Out_1 \tag{5.24}$$

W_{22} için güncelleme:

$$\frac{\partial E}{\partial W_{22}} = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i}\frac{\partial K_i}{\partial W_{22}} = (y_{ref} - y_s).J.e_i.Out_2 \tag{5.25}$$

W_{13} için güncelleme:

$$\frac{\partial E}{\partial W_{13}} = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d}\frac{\partial K_d}{\partial W_{13}} = (y_{ref} - y_s).J.e_d.Out_1 \tag{5.26}$$

W_{23} için güncelleme:

$$\frac{\partial E}{\partial W_{23}} = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d}\frac{\partial K_d}{\partial W_{23}} = (y_{ref} - y_s).J.e_d.Out_2 \tag{5.27}$$

Yoğun katman için ağırlıklar hesaplandıktan sonra sapma değerleri de benzer şekilde hesaplanmıştır. Sapma değerleri incelenirse ağırlıklara oldukça benzer ifadeler içerdiği görülebilir.

B_1 için güncelleme:

$$\frac{\partial E}{\partial B_1} = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_p}\frac{\partial K_p}{\partial B_1} = (y_{ref} - y_s).J.e_p \tag{5.28}$$

B_2 için güncelleme:

$$\frac{\partial E}{\partial B_2} = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i}\frac{\partial K_i}{\partial B_2} = (y_{ref} - y_s).J.e_i \tag{5.29}$$

B_3 için güncelleme:

$$\frac{\partial E}{\partial B_3} = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d}\frac{\partial K_d}{\partial B_3} = (y_{ref} - y_s).J.e_d \tag{5.30}$$

Yoğun katman güncelleme değerleri bulunduktan sonra Şekil 5.2'deki UKSB 2 ağırlıkları için güncelleme değerleri hesaplanabilir. Bunun için çıkışa en yakın ağırlıktan başlanarak en uzak ağırlığa doğru hesaplama yapılacak ve ağırlığa gidişteki her durumu göz önüne alınacaktır. Bundan dolayı güncelleme ifadeleri uzun çıkarımlar

içermektedir. Bu durumu gidermek için bazı tekrar eden ifadelere semboller atayarak daha kolay ifadeler elde edilmiştir. Bu sembolleri aşağıdaki gibidir:

$$T_P = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_p} = (y_{ref} - y_s).J.e_p \tag{5.31}$$

$$T_i = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i} = (y_{ref} - y_s).J.e_i \tag{5.32}$$

$$T_D = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d} = (y_{ref} - y_s).J.e_d \tag{5.33}$$

W_{O2} için güncelleme:

$$\Delta O_2 = \frac{\partial Out_2}{\partial O_2(out)}\frac{\partial O_2(out)}{\partial O_2(in)}\frac{\partial O_2(in)}{\partial W_{O2}}$$

$$= tanh(S_2)(\sigma(W_{O2}X_2 + U_{O2}Out_1)(1 - \sigma(W_{O2}X_2 + U_{O2}Out_1)))X_2 \tag{5.34}$$

Bu durumda güncelleme değeri aşağıdaki gibi hesaplanabilir.

$$\frac{\partial E}{\partial W_{O2}} = T_P\frac{\partial K_p}{\partial Out_2}\Delta O_2 + T_i\frac{\partial K_i}{\partial Out_2}\Delta O_2 + T_D\frac{\partial K_d}{\partial Out_2}\Delta O_2 \tag{5.35}$$

$$\frac{\partial E}{\partial W_{O2}} = (T_P W_{21} + T_i W_{22} + T_D W_{23})\Delta O_2 \tag{5.36}$$

U_{O2} için güncelleme:

$$\Delta U_{O2} = \frac{\partial Out_2}{\partial O_2(out)}\frac{\partial O_2(out)}{\partial O_2(in)}\frac{\partial O_2(in)}{\partial U_{O2}}$$

$$= tanh(S_2)(\sigma(W_{O2}X_2 + U_{O2}Out_1)(1 - \sigma(W_{O2}X_2 + U_{O2}Out_1)))Out_1 \tag{5.37}$$

Bu durumda güncelleme değeri aşağıdaki gibi hesaplanabilir.

$$\frac{\partial E}{\partial U_{O2}} = T_P\frac{\partial K_p}{\partial Out_2}\Delta U_{O2} + T_i\frac{\partial K_i}{\partial Out_2}\Delta U_{O2} + T_D\frac{\partial K_d}{\partial Out_2}\Delta U_{O2} \tag{5.38}$$

$$\frac{\partial E}{\partial U_{O2}} = (T_P W_{21} + T_i W_{22} + T_D W_{23})\Delta U_{O2} \tag{5.39}$$

Benzer şekilde W_{A2} ve U_{A2} değerleri de güncellenmiştir.

W_{A2} için güncelleme:

$$\Delta A_2 = \frac{\partial Out_2}{\partial S_2(out)} \frac{\partial S_2(out)}{\partial S_2(in)} \frac{\partial S_2(in)}{\partial A_2(out)} \frac{\partial A_2(out)}{\partial A_2(in)} \frac{\partial A_2(in)}{\partial W_{A2}}$$

$$= O_2(1 - tanh(S_2)^2)(i_2)(1 - A_2^2)X_2 \tag{5.40}$$

Bu durumda güncelleme değeri aşağıdaki gibi hesaplanabilir.

$$\frac{\partial E}{\partial W_{A2}} = T_P \frac{\partial K_p}{\partial Out_2} \Delta A_2 + T_i \frac{\partial K_i}{\partial Out_2} \Delta A_2 + T_D \frac{\partial K_d}{\partial Out_2} \Delta A_2 \tag{5.41}$$

$$\frac{\partial E}{\partial W_{A2}} = (T_P W_{21} + T_i W_{22} + T_D W_{23}) \Delta A_2 \tag{5.42}$$

U_{A2} için güncelleme:

$$\Delta U_{A2} = \frac{\partial Out_2}{\partial S_2(out)} \frac{\partial S_2(out)}{\partial S_2(in)} \frac{\partial S_2(in)}{\partial A_2(out)} \frac{\partial A_2(out)}{\partial A_2(in)} \frac{\partial A_2(in)}{\partial U_{A2}}$$

$$= O_2(1 - tanh(S_2)^2)(i_2)(1 - A_2^2)Out_1 \tag{5.43}$$

Bu durumda güncelleme değeri aşağıdaki gibi hesaplanır.

$$\frac{\partial E}{\partial U_{A2}} = T_P \frac{\partial K_p}{\partial Out_2} \Delta U_{A2} + T_i \frac{\partial K_i}{\partial Out_2} \Delta U_{A2} + T_D \frac{\partial K_d}{\partial Out_2} \Delta U_{A2} \tag{5.44}$$

$$\frac{\partial E}{\partial U_{A2}} = (T_P W_{21} + T_i W_{22} + T_D W_{23}) \Delta U_{A2} \tag{5.45}$$

UKSB 2 hücresi için sıradaki hesaplanacak ağırlık değeri W_{i2} ve U_{i2} ağırlıklarıdır. Bu ağırlıklar da benzer şekilde hesaplanmaya çalışılırsa aşağıdaki sonuçlar elde edilir.

W_{i2} için güncelleme:

$$\Delta i_2 = \frac{\partial Out_2}{\partial S_2(out)} \frac{\partial S_2(out)}{\partial S_2(in)} \frac{\partial S_2(in)}{\partial i_2(out)} \frac{\partial i_2(out)}{\partial i_2(in)} \frac{\partial i_2(in)}{\partial W_{i2}}$$

$$= O_2(1 - tanh(S_2)^2)(A_2)(i_2(1 - i_2))X_2 \tag{5.46}$$

Bu durumda güncelleme değeri aşağıdaki gibi hesaplanır.

$$\frac{\partial E}{\partial W_{i2}} = T_P \frac{\partial K_p}{\partial Out_2} \Delta i_2 + T_i \frac{\partial K_i}{\partial Out_2} \Delta i_2 + T_D \frac{\partial K_d}{\partial Out_2} \Delta i_2 \tag{5.47}$$

$$\frac{\partial E}{\partial W_{i2}} = (T_P W_{21} + T_i W_{22} + T_D W_{23}) \Delta i_2 \tag{5.48}$$

U_{i2} için güncelleme:

$$\Delta U_{i2} = \frac{\partial Out_2}{\partial S_2(out)} \frac{\partial S_2(out)}{\partial S_2(in)} \frac{\partial S_2(in)}{\partial i_2(out)} \frac{\partial i_2(out)}{\partial i_2(in)} \frac{\partial i_2(in)}{\partial U_{i2}}$$

$$= O_2(1 - tanh(S_2)^2)(A_2)(i_2(1 - i_2))Out_1 \tag{5.49}$$

Bu durumda güncelleme değeri aşağıdaki gibi hesaplanır.

$$\frac{\partial E}{\partial U_{i2}} = T_P \frac{\partial K_p}{\partial Out_2} \Delta U_{i2} + T_i \frac{\partial K_i}{\partial Out_2} \Delta U_{i2} + T_D \frac{\partial K_d}{\partial Out_2} \Delta U_{i2} \tag{5.50}$$

$$\frac{\partial E}{\partial U_{i2}} = (T_P W_{21} + T_i W_{22} + T_D W_{23}) \Delta U_{i2} \tag{5.51}$$

UKSB 2'nin son ağırlığı olan W_{F2} ve U_{F2} için güncelleme değerleri hesaplanmıştır.

W_{F2} için güncelleme:

$$\Delta F_2 = \frac{\partial Out_2}{\partial S_2(out)} \frac{\partial S_2(out)}{\partial S_2(in)} \frac{\partial S_2(in)}{\partial F_2(out)} \frac{\partial F_2(out)}{\partial F_2(in)} \frac{\partial F_2(in)}{\partial W_{F2}}$$

$$= O_2(1 - tanh(S_2)^2)(S_1)(F_2(1 - F_2))X_2 \tag{5.52}$$

Bu durumda güncelleme değeri aşağıdaki gibi hesaplanır.

$$\frac{\partial E}{\partial W_{F2}} = T_P \frac{\partial K_p}{\partial Out_2} \Delta F_2 + T_i \frac{\partial K_i}{\partial Out_2} \Delta F_2 + T_D \frac{\partial K_d}{\partial Out_2} \Delta F_2 \tag{5.53}$$

$$\frac{\partial E}{\partial W_{F2}} = (T_P W_{21} + T_i W_{22} + T_D W_{23}) \Delta F_2 \tag{5.54}$$

U_{F2} için güncelleme:

$$\Delta U_{F2} = \frac{\partial Out_2}{\partial S_2(out)} \frac{\partial S_2(out)}{\partial S_2(in)} \frac{\partial S_2(in)}{\partial F_2(out)} \frac{\partial F_2(out)}{\partial F_2(in)} \frac{\partial F_2(in)}{\partial U_{F2}}$$

$$= O_2(1 - tanh(S_2)^2)(S_1)(F_2(1 - F_2))Out_1 \tag{5.55}$$

Bu durumda güncelleme değeri aşağıdaki gibi hesaplanır.

$$\frac{\partial E}{\partial U_{F2}} = T_P \frac{\partial K_p}{\partial Out_2} \Delta U_{F2} + T_i \frac{\partial K_i}{\partial Out_2} \Delta U_{F2} + T_D \frac{\partial K_d}{\partial Out_2} \Delta U_{F2} \tag{5.56}$$

$$\frac{\partial E}{\partial U_{F2}} = (T_P W_{21} + T_i W_{22} + T_D W_{23}) \Delta U_{F2} \tag{5.57}$$

UKSB 2 için tüm ağırlıklar güncellendikten sonra UKSB 1 için ağırlıkların güncellemesi işlemine geçilir. UKSB 1 için dikkat edilmesi gereken husus UKSB 1 üzerine hem yoğun katmandan hem de UKSB 2 hücresi üzerinden geçiş olmasıdır. Bu durumda her durum toplam olarak ifadelere eklenmeli ve her durum göz önünde bulundurulmalıdır.

W_{O1} için güncelleme:

Her yol dikkate alınarak K_p, K_i ve K_d çıkışlarının herbiri için teker teker geriye dönük hata yayılımları hesaplanmalıdır. Öncelikle ΔO_1 değeri bulunarak başlanmıştır. Bu ifade çok sık kullanılacağından dolayı sembolik olarak formüllerin içine yazılması hedeflenmiştir.

$$\Delta O_1 = \frac{\partial Out_1}{\partial O_1(out)} \frac{\partial O_1(out)}{\partial O_1(in)} \frac{\partial O_1(in)}{\partial W_{O1}} = tanh(S_1)(O_1(1-O_1)X_1 \tag{5.58}$$

Güncelleme için matematiksel çıkarım aşağıdaki gibidir.

Yoğun Katman Üzerinden Direkt UKSB 1 Güncellemeleri:

$$YK = \frac{\partial E}{\partial y_s} \frac{\partial y_s}{\partial u} \frac{\partial u}{\partial K_p} \frac{\partial K_p}{\partial Out_1} \Delta O_1 + \frac{\partial E}{\partial y_s} \frac{\partial y_s}{\partial u} \frac{\partial u}{\partial K_i} \frac{\partial K_i}{\partial Out_1} \Delta O_1 + \frac{\partial E}{\partial y_s} \frac{\partial y_s}{\partial u} \frac{\partial u}{\partial K_d} \frac{\partial K_d}{\partial Out_1} \Delta O_1 \tag{5.59}$$

Yoğun Katmanın Ardından UKSB 2 Üzerinden UKSB 1 Güncellemeleri:

$$LS = \frac{\partial E}{\partial y_s} \frac{\partial y_s}{\partial u} \frac{\partial u}{\partial K_p} \frac{\partial K_p}{\partial Out_2} \frac{\partial Out_2}{\partial O_2(out)} \frac{\partial O_2(out)}{\partial O_2(in)} \frac{\partial O_2(in)}{\partial Out_1} DeltaO_1 +$$

$$\frac{\partial E}{\partial y_s} \frac{\partial y_s}{\partial u} \frac{\partial u}{\partial K_p} \frac{\partial K_p}{\partial Out_2} \frac{\partial Out_2}{\partial S_2(out)} \frac{\partial S_2(out)}{\partial S_2(in)} \frac{\partial S_2(in)}{\partial A_2(out)} \frac{\partial A_2(out)}{\partial A_2(in)} \frac{\partial A_2(in)}{\partial Out_1} DeltaO_1 +$$

$$\frac{\partial E}{\partial y_s} \frac{\partial y_s}{\partial u} \frac{\partial u}{\partial K_p} \frac{\partial K_p}{\partial Out_2} \frac{\partial Out_2}{\partial S_2(out)} \frac{\partial S_2(out)}{\partial S_2(in)} \frac{\partial S_2(in)}{\partial i_2(out)} \frac{\partial i_2(out)}{\partial i_2(in)} \frac{\partial i_2(in)}{\partial Out_1} DeltaO_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_p}\frac{\partial K_p}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial F_2(out)}\frac{\partial F_2(out)}{\partial F_2(in)}\frac{\partial F_2(in)}{\partial Out_1}DeltaO_1+$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i}\frac{\partial K_i}{\partial Out_2}\frac{\partial Out_2}{\partial O_2(out)}\frac{\partial O_2(out)}{\partial O_2(in)}\frac{\partial O_2(in)}{\partial Out_1}DeltaO_1+$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i}\frac{\partial K_i}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial A_2(out)}\frac{\partial A_2(out)}{\partial A_2(in)}\frac{\partial A_2(in)}{\partial Out_1}DeltaO_1+$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i}\frac{\partial K_i}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial i_2(out)}\frac{\partial i_2(out)}{\partial i_2(in)}\frac{\partial i_2(in)}{\partial Out_1}DeltaO_1+$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i}\frac{\partial K_i}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial F_2(out)}\frac{\partial F_2(out)}{\partial F_2(in)}\frac{\partial F_2(in)}{\partial Out_1}DeltaO_1+$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d}\frac{\partial K_d}{\partial Out_2}\frac{\partial Out_2}{\partial O_2(out)}\frac{\partial O_2(out)}{\partial O_2(in)}\frac{\partial O_2(in)}{\partial Out_1}DeltaO_1+$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d}\frac{\partial K_d}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial A_2(out)}\frac{\partial A_2(out)}{\partial A_2(in)}\frac{\partial A_2(in)}{\partial Out_1}DeltaO_1+$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d}\frac{\partial K_d}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial i_2(out)}\frac{\partial i_2(out)}{\partial i_2(in)}\frac{\partial i_2(in)}{\partial Out_1}DeltaO_1+$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d}\frac{\partial K_d}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial F_2(out)}\frac{\partial F_2(out)}{\partial F_2(in)}\frac{\partial F_2(in)}{\partial Out_1}DeltaO_1+ \quad (5.60)$$

Sonuç güncelleme ifadesi her iki güncelleme değerinin toplamı ile aşağıdaki şekilde hesaplanır.

$$\frac{\partial E}{\partial W_{O1}} = YK + LS \quad (5.61)$$

Değerleri belirli değişken dönüşümleri ile yerlerine yazarak sonuçta elde edilecek güncelleme ifadesi aşağıdaki gibi çıkarılabilir.

$$M = (T_P W_{21} + T_i W_{22} + T_D W_{23}) \quad (5.62)$$

$$O_P = tanh(S_2)(O_2(1 - O_2) \quad (5.63)$$

$$A_P = O_2(1-tanh(S_2)^2)(i_2)(1-A_2^2) \tag{5.64}$$

$$i_P = O_2(1-tanh(S_2)^2)(A_2)(i_2(1-i_2) \tag{5.65}$$

$$F_P = O_2(1-tanh(S_2)^2)(S_1)(F_2(1-F_2) \tag{5.66}$$

$$\frac{\partial E}{\partial W_{O1}} = (T_P W_{11} + T_i W_{12} + T_D W_{13})\Delta O_1 +$$

$$(MO_P W_{O2} + MA_P W_{A2} + Mi_P W_{i2} + MF_P W_{F2})\Delta O_1 \tag{5.67}$$

U_{O1} için güncelleme:

W_{O1}'e oldukça benzer yapıdadır. Sadece güncelleme yapılması gereken kısım ağırlığa özel kısımdır.

$$\Delta U_{O1} = tanh(S_1)(O_1(1-O_1))Out_0 \tag{5.68}$$

$$\frac{\partial E}{\partial W_{O1}} = (T_P W_{11} + T_i W_{12} + T_D W_{13})\Delta U_{O1} +$$

$$(MO_P W_{O2} + MA_P W_{A2} + Mi_P W_{i2} + MF_P W_{F2})\Delta U_{O1} \tag{5.69}$$

W_{A1} için güncelleme:

Küçük farklılıklar olacak şekilde W_{O1}'e benzer olarak güncelleme değerleri bulunacaktır. Öncelikle ΔA_1 değeri bulunmalıdır. (Bu ifade çok sık kullanılmıştır.)

$$\Delta A_1 = \frac{\partial Out_1}{\partial S_1(out)} \frac{\partial S_1(out)}{\partial S_1(in)} \frac{\partial S_1(in)}{\partial A_1(out)} \frac{\partial A_1(out)}{\partial A_1(in)} \frac{\partial A_1(in)}{\partial W_{A1}} \tag{5.70}$$

$$\Delta A_1 = (O_1)(1-tanh(S_1)^2)(i_1)(1-A_1^2)X_1 \tag{5.71}$$

Güncelleme değerlerini hesaplamak için iki aşamalı çıkarım gerçekleştirilecektir.

Yoğun Katman Üzerinden Direkt UKSB 1 Güncellemeleri:

$$YK = \frac{\partial E}{\partial y_s} \frac{\partial y_s}{\partial u} \frac{\partial u}{\partial K_p} \frac{\partial K_p}{\partial Out_1} \Delta A_1 + \frac{\partial E}{\partial y_s} \frac{\partial y_s}{\partial u} \frac{\partial u}{\partial K_i} \frac{\partial K_i}{\partial Out_1} \Delta A_1 + \frac{\partial E}{\partial y_s} \frac{\partial y_s}{\partial u} \frac{\partial u}{\partial K_d} \frac{\partial K_d}{\partial Out_1} \Delta A_1 \tag{5.72}$$

Yoğun Katmanın Ardından UKSB 2 Üzerinden UKSB 1 Güncellemeleri:

$$LS = \frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_p}\frac{\partial K_p}{\partial Out_2}\frac{\partial Out_2}{\partial O_2(out)}\frac{\partial O_2(out)}{\partial O_2(in)}\frac{\partial O_2(in)}{\partial Out_1}\Delta A_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_p}\frac{\partial K_p}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial A_2(out)}\frac{\partial A_2(out)}{\partial A_2(in)}\frac{\partial A_2(in)}{\partial Out_1}\Delta A_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_p}\frac{\partial K_p}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial i_2(out)}\frac{\partial i_2(out)}{\partial i_2(in)}\frac{\partial i_2(in)}{\partial Out_1}\Delta A_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_p}\frac{\partial K_p}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial F_2(out)}\frac{\partial F_2(out)}{\partial F_2(in)}\frac{\partial F_2(in)}{\partial Out_1}\Delta A_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_p}\frac{\partial K_p}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial S_1(out)}\frac{\partial S_1(out)}{\partial S_1(in)}\frac{\partial S_1(in)}{\partial A_1(out)}\frac{\partial A_1(out)}{\partial A_1(in)}\frac{\partial A_1(in)}{\partial W_{A1}} +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i}\frac{\partial K_i}{\partial Out_2}\frac{\partial Out_2}{\partial O_2(out)}\frac{\partial O_2(out)}{\partial O_2(in)}\frac{\partial O_2(in)}{\partial Out_1}\Delta A_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i}\frac{\partial K_i}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial A_2(out)}\frac{\partial A_2(out)}{\partial A_2(in)}\frac{\partial A_2(in)}{\partial Out_1}\Delta A_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i}\frac{\partial K_i}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial i_2(out)}\frac{\partial i_2(out)}{\partial i_2(in)}\frac{\partial i_2(in)}{\partial Out_1}\Delta A_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i}\frac{\partial K_i}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial F_2(out)}\frac{\partial F_2(out)}{\partial F_2(in)}\frac{\partial F_2(in)}{\partial Out_1}\Delta A_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_i}\frac{\partial K_i}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial S_1(out)}\frac{\partial S_1(out)}{\partial S_1(in)}\frac{\partial S_1(in)}{\partial A_1(out)}\frac{\partial A_1(out)}{\partial A_1(in)}\frac{\partial A_1(in)}{\partial W_{A1}} +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d}\frac{\partial K_d}{\partial Out_2}\frac{\partial Out_2}{\partial O_2(out)}\frac{\partial O_2(out)}{\partial O_2(in)}\frac{\partial O_2(in)}{\partial Out_1}\Delta A_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d}\frac{\partial K_d}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial A_2(out)}\frac{\partial A_2(out)}{\partial A_2(in)}\frac{\partial A_2(in)}{\partial Out_1}\Delta A_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d}\frac{\partial K_d}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial i_2(out)}\frac{\partial i_2(out)}{\partial i_2(in)}\frac{\partial i_2(in)}{\partial Out_1}\Delta A_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d}\frac{\partial K_d}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial F_2(out)}\frac{\partial F_2(out)}{\partial F_2(in)}\frac{\partial F_2(in)}{\partial Out_1}\Delta A_1 +$$

$$\frac{\partial E}{\partial y_s}\frac{\partial y_s}{\partial u}\frac{\partial u}{\partial K_d}\frac{\partial K_d}{\partial Out_2}\frac{\partial Out_2}{\partial S_2(out)}\frac{\partial S_2(out)}{\partial S_2(in)}\frac{\partial S_2(in)}{\partial S_1(out)}\frac{\partial S_1(out)}{\partial S_1(in)}\frac{\partial S_1(in)}{\partial A_1(out)}\frac{\partial A_1(out)}{\partial A_1(in)}\frac{\partial A_1(in)}{\partial W_{A1}} \quad (5.73)$$

Sonuç güncelleme ifadesi her iki güncelleme değerinin toplamı ile aşağıdaki şekilde hesaplanır.

$$\frac{\partial E}{\partial W_{A1}} = YK + LS \tag{5.74}$$

Değerleri belirli değişken dönüşümleri ile yerlerine yazarak sonuçta elde edilecek güncelleme ifadesi aşağıdaki gibi çıkarılabilir.

$$M = (T_P W_{21} + T_i W_{22} + T_D W_{23}) \tag{5.75}$$

$$O_P = tanh(S_2)(O_2(1 - O_2) \tag{5.76}$$

$$A_P = O_2(1 - tanh(S_2)^2)(i_2)(1 - A_2^2) \tag{5.77}$$

$$i_P = O_2(1 - tanh(S_2)^2)(A_2)(i_2(1 - i_2) \tag{5.78}$$

$$F_P = O_2(1 - tanh(S_2)^2)(S_1)(F_2(1 - F_2) \tag{5.79}$$

$$S_{derivasyon} = (M)((O_2)(1 - tanh(S_2)^2))((F_2)(1 - tanh(S_1))^2))((i_1)(1 - A_1^2)X_1) \tag{5.80}$$

$$\frac{\partial E}{\partial W_{A1}} = (T_P W_{11} + T_i W_{12} + T_D W_{13})\Delta A_1 +$$

$$(MO_P W_{O2} + MA_P W_{A2} + Mi_P W_{i2} + MF_P W_{F2})\Delta A_1 + S_{derivasyon} \tag{5.81}$$

U_{A1} için güncelleme:

W_{A1}'e oldukça benzer yapıdadır. Sadece güncelleme yapılması gereken kısım ağırlığa özel kısımdır.

$$\Delta U_{A1} = (O_1)(1 - tanh(S_1)^2)(i_1)(1 - A_1^2)Out_0 \tag{5.82}$$

$$S_{derivasyon} = (M)((O_2)(1 - tanh(S_2)^2))((F_2)(1 - tanh(S_1))^2))((i_1)(1 - A_1^2)Out_0) \tag{5.83}$$

$$\frac{\partial E}{\partial W_{A1}} = (T_P W_{11} + T_i W_{12} + T_D W_{13})\Delta A_1 +$$

$$(MO_P W_{O2} + MA_P W_{A2} + Mi_P W_{i2} + MF_P W_{F2})\Delta A_1 + S_{derivasyon} \tag{5.84}$$

W_{i1} için güncelleme:

W_{A1} ile çok benzerlik gösterdiği için sadece değişen ifadeler güncellenerek güncelleme değeri yazılmıştır.

$$\Delta i_1 = (O_1)(1 - tanh(S_1)^2)(A_1)(i_1(1 - i_1))X_1 \tag{5.85}$$

$$S_{derivasyon} = (M)((O_2)(1 - tanh(S_2)^2))((F_2)(1 - tanh(S_1))^2))((A_1)(i_1(1 - i_1))X_1) \tag{5.86}$$

$$\frac{\partial E}{\partial W_{i1}} = (T_P W_{11} + T_i W_{12} + T_D W_{13})\Delta i_1 +$$

$$(MO_P W_{O2} + MA_P W_{A2} + Mi_P W_{i2} + MF_P W_{F2})\Delta i_1 + S_{derivasyon} \tag{5.87}$$

U_{i1} için güncelleme:

$$\Delta U_{i1} = (O_1)(1 - tanh(S_1)^2)(A_1)(i_1(1 - i_1))Out_0 \tag{5.88}$$

$$S_{derivasyon} = (M)((O_2)(1 - tanh(S_2)^2))((F_2)(1 - tanh(S_1))^2))((A_1)(i_1(1 - i_1))Out_0) \tag{5.89}$$

$$\frac{\partial E}{\partial U_{i1}} = (T_P W_{11} + T_i W_{12} + T_D W_{13})\Delta U_{i1} +$$

$$(MO_P W_{O2} + MA_P W_{A2} + Mi_P W_{i2} + MF_P W_{F2})\Delta U_{i1} + S_{derivasyon} \tag{5.90}$$

W_{F1} için güncelleme:

W_{A1} ile çok benzerlik göstermektedir. Benzer biçimde çıkarımlar yapılmıştır. UKSB 1 hücresi için güncellenecek son ağırlık değeridir.

$$\Delta F_1 = (O_1)(1 - tanh(S_1)^2)(S_0)(F_1(1 - F_1))X_1 \tag{5.91}$$

$$S_{derivasyon} = (M)((O_2)(1 - tanh(S_2)^2))((F_2)(1 - tanh(S_1))^2))((S_0)(F_1(1 - F_1))X_1) \tag{5.92}$$

$$\frac{\partial E}{\partial W_{F1}} = (T_P W_{11} + T_i W_{12} + T_D W_{13})\Delta F_1 +$$

$$(MO_P W_{O2} + MA_P W_{A2} + Mi_P W_{i2} + MF_P W_{F2})\Delta F_1 + S_{derivasyon} \tag{5.93}$$

U_{F1} için güncelleme:

$$\Delta U_{F1} = (O_1)(1 - tanh(S_1)^2)(S_0)(F_1(1 - F_1))Out_0 \tag{5.94}$$

$$S_{derivasyon} = (M)((O_2)(1 - tanh(S_2)^2))((F_2)(1 - tanh(S_1))^2))((S_0)(F_1(1 - F_1))Out_0) \tag{5.95}$$

$$\frac{\partial E}{\partial U_{F1}} = (T_P W_{11} + T_i W_{12} + T_D W_{13})\Delta U_{F1} +$$

$$(MO_P W_{O2} + MA_P W_{A2} + Mi_P W_{i2} + MF_P W_{F2})\Delta U_{F1} + S_{derivasyon} \tag{5.96}$$

Güncelleme değerleri hesaplandıktan sonra her ağırlık için güncellemeler aşağıdaki şekilde yapılarak ağırlıkların bir sonraki adımdaki değerleri bulunabilir. Güncelleme denklemindeki k öğrenme faktörüdür ve deneysel olarak tercih edilebilir. Öğrenme faktörü uygulama bazında deneysel olarak $k = 0.01$ değeri seçilmiştir.

$$W(k+1) = W(k) + k\frac{\Delta E}{\Delta W} \tag{5.97}$$

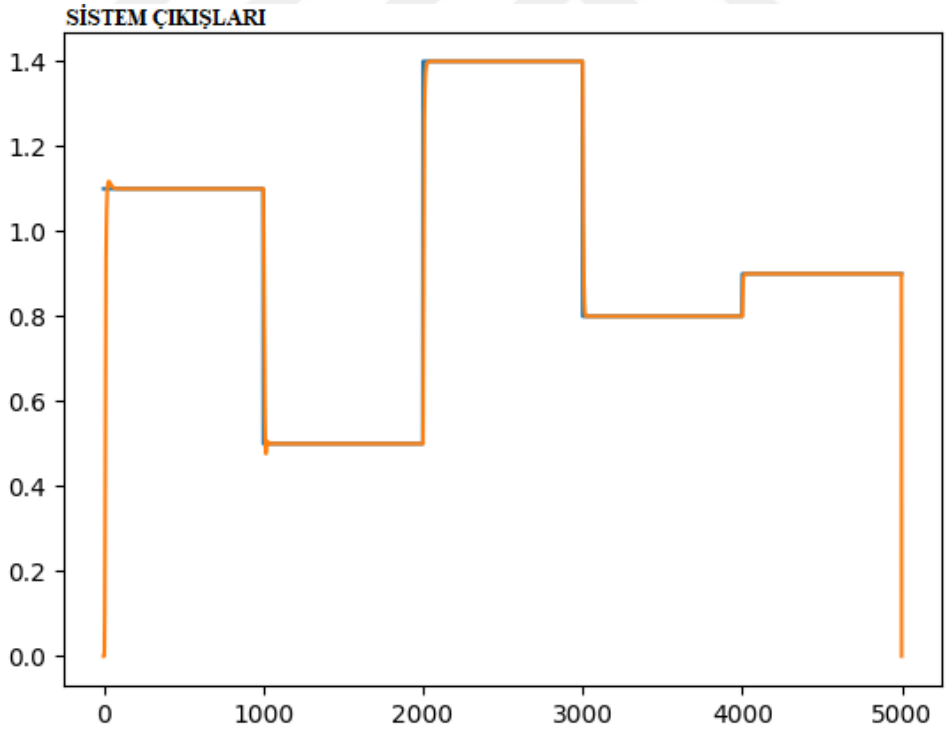
5.3 Örnek Sistemler Üzerinde Sonuçlar

UKSB tabanlı uyarlamalı kontrolör örnek sistemler üzerinde test edilmiştir. Bu kısımda kullanılan ilk sistem UKSB tabanlı sistem tanımlamada kullanılan doğrusal olmayan test sistemi, ikinci sistem ise doğrusal olmayan CSTR sistemidir. Sistem tanımlama bölümünde oluşturulan UKSB modeller yine bu bölümde kullanılmıştır. Kontrolör testleri sırasında hem matematiksel modeller üzerinde hemde sistem tanımlama ile oluşturulan modeller üzerinde testler gerçekleştirilmiştir. Referans sinyal örnek sistemlerin çalışma aralıklarına uygun olarak seçilmiştir.

Bu bölümde ayrıca ölçüm aletlerinin gürültülerini de katmak amacıyla sonuçların son kısımlarında geri besleme işaretlerine gürültü sinyalleri eklenmiştir. Burada amaç kontrolörün gürültülere karşı tepkisini ölçmektir. Bu testler daha güçlü kontrolörler geliştirebilmek adına önemlidir ancak sistemler her gürültü değerine karşı dayanıklı değildir. Her sistem için belirli bir eşik değeri vardır. Bu bölümde de belirli bir dB seviyesi seçilerek testler gerçekleştirilmiştir. Gürültü değeri aşırı arttırılırsa bu durumda kontrolör performansı doğal olarak düşecektir.

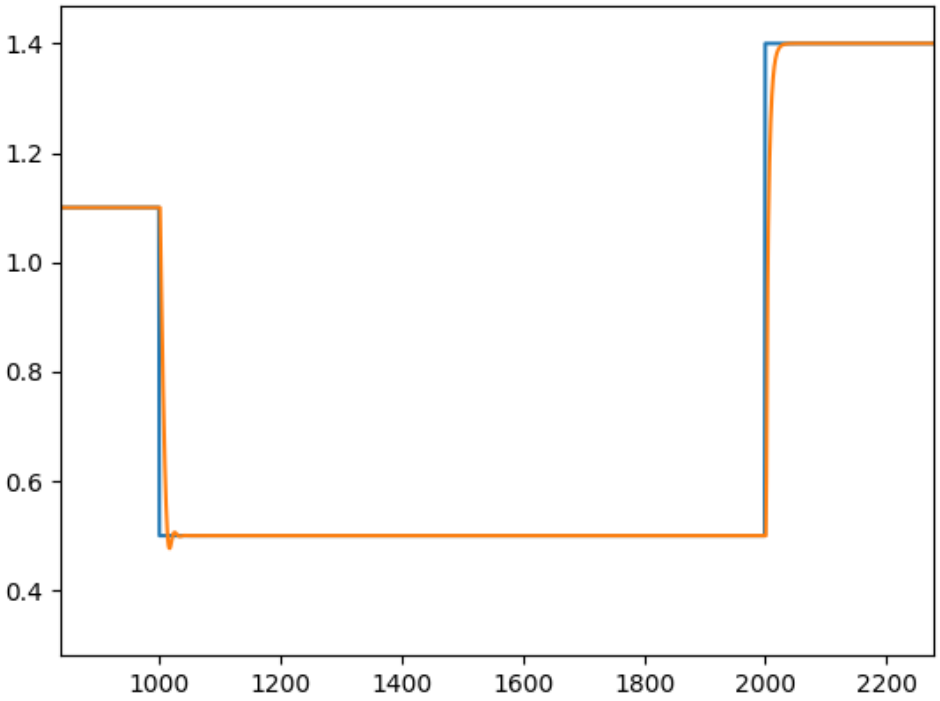
5.3.1 Doğrusal olmayan test sistemi sonuçları

UKSB tabanlı kontrolör ilk olarak doğrusal olmayan sistem üzerinde test edilmiştir. Testin ilk aşamasında sistem tanımlama sonuçlarını da değerlendirebilmek adına UKSB tabanlı kontrolör, sistem matematiksel denklemi üzerine uygulanmıştır. Şekil 5.3'de UKSB tabanlı kontrolör, doğrusal olmayan sistem denklemine uygulandığında oluşan çıktılar gösterilmiştir. Sistem çıktısının referans işareti ne kadar yakından takip ettiği incelenebilir ve bu sonuç UKSB tabanlı kontrolörnün başarısını göstermektedir. Direkt olarak matematiksel ifadesi elimizde olan sistemler için de UKSB tabanlı kontrolör tercih edilebilir. Bunun yanı sıra mimari değiştirilerek daha farklı sonuçlar elde etmek de mümkündür. Uygulama kapsamında kullanılan mimari deneysel sonuçlar ile tercih edilmiş ve fazla karmaşık olmayan bir mimaridir. Daha kompleks sistemlerde daha karmaşık mimariler tercih edilebilir ancak karmaşıklık arttıkça öğrenmenin her zaman doğru orantılı artmayacağı unutulmaması gereken bir husustur. Mimari seçimlerinde ne fazla basit ne de fazla karmaşık yapılar tercih edilmelidir.



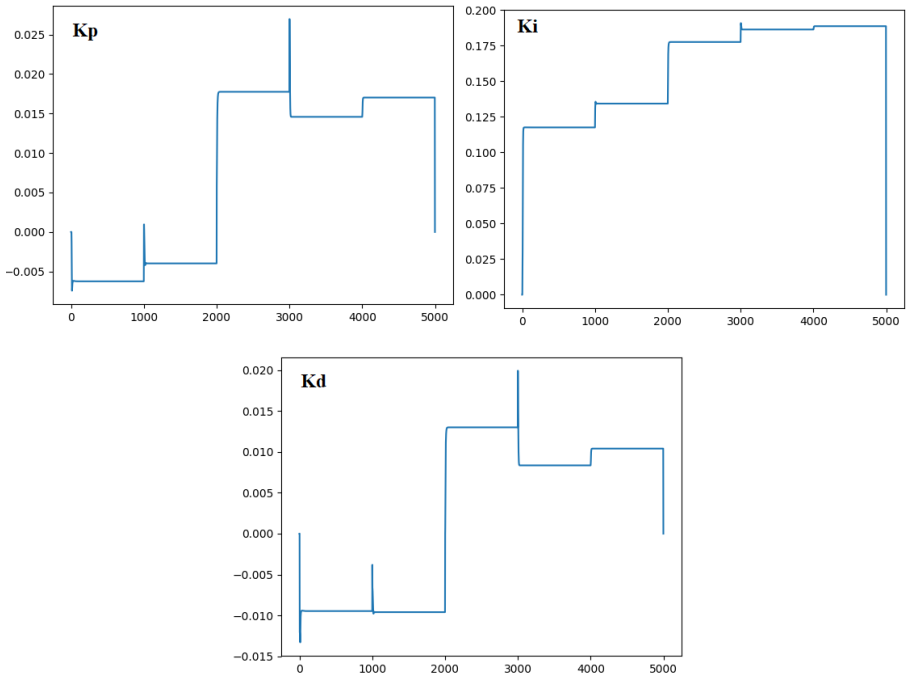
Şekil 5.3 : UKSB Tabanlı Uyarlamalı kontrolör ile kontrol edilen sistem çıkışları (Matematiksel model kullanılmıştır)

Şekil 5.4'de ise aşım ve yerleşme zamanını daha iyi görebilmek için yakınlaştırılmış hali gösterilmiştir. Bu şekile bakılarak yerleşme zamanlarının oldukça küçük değerlerde olduğu gözlemlenebilir.



Şekil 5.4 : UKSB Tabanlı Uyarlamalı kontrolör ile kontrol edilen sistem çıkışları yakınlaştırılmış hali (Matematiksel model kullanılmıştır)

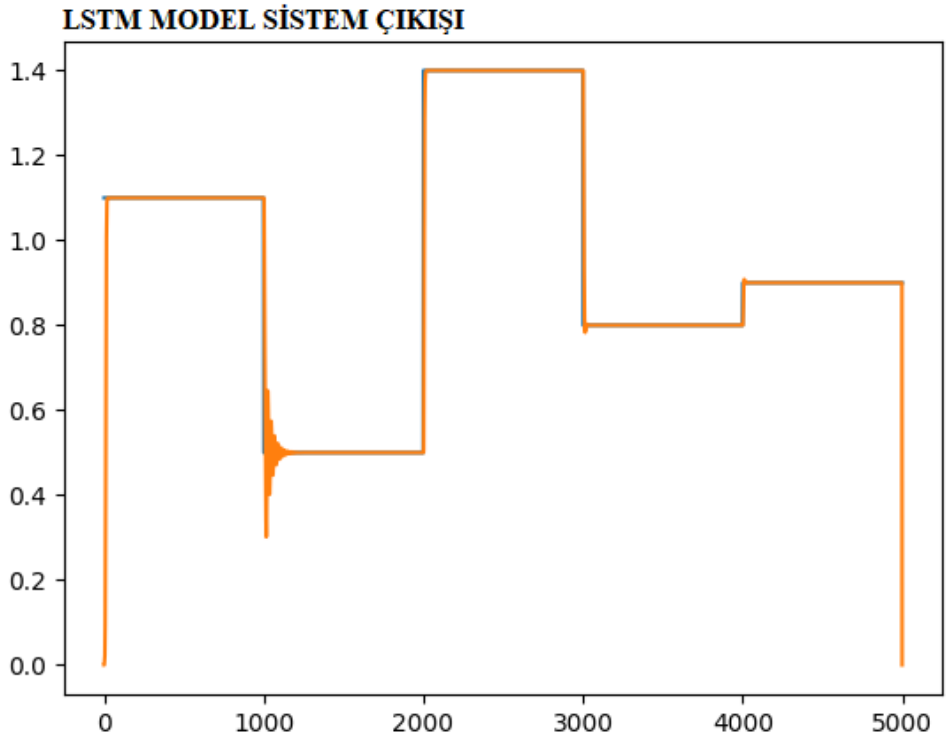
UKSB tabanlı uyarlamalı kontrolörün ürettiği K_p, K_i ve K_d değerlerinin değişimleri Şekil 5.5 'de gösterilmiştir. Dikkat edilirse sistem sürekli olarak kontrolör değerlerini değiştirmekte ve uyarlamalı olarak referans sinyali takip edecek şekilde ayarlamaktadır.



Şekil 5.5 : UKSB Tabanlı Uyarlamalı Kontrolör K_p, K_i ve K_d çıkışları

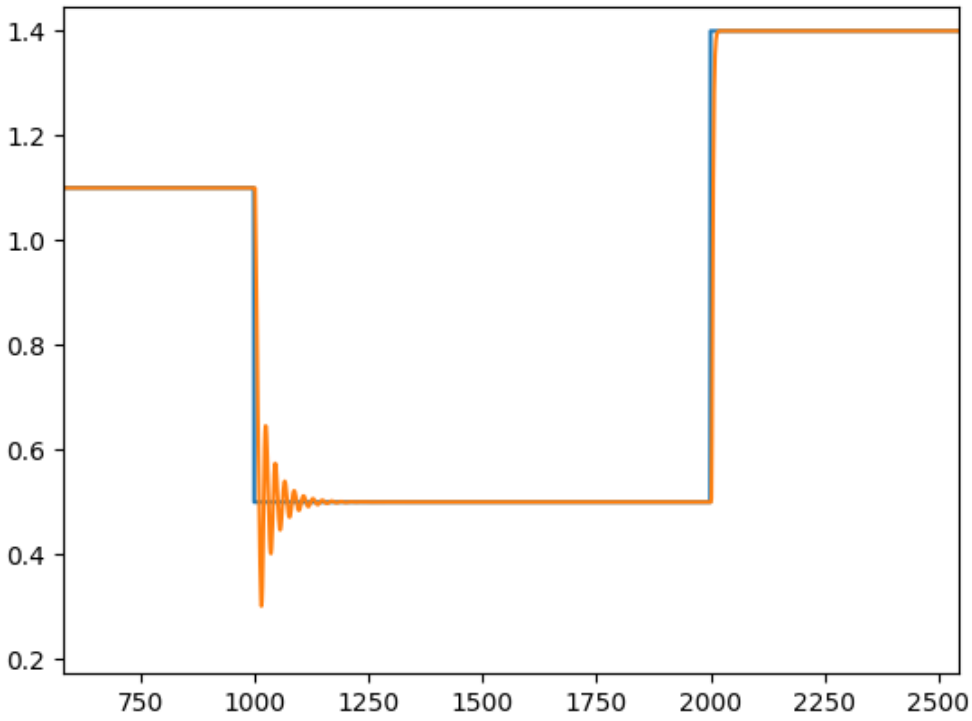
Sıradaki adımda ise UKSB tabanlı uyarlamalı kontrolör, daha önceki bölümlerde modellenen ve UKSB tabanlı sistem tanımlama ile elde edilen modele uygulanmıştır. Performansta düşüşler görülebilir. Bunun nedeni hiçbir modellemenin sistemi tam olarak yansıtamamasından kaynaklanmaktadır. Ancak her ne kadar ilk sisteme göre başarı düşük olsa da kontrol açısından başarının yüksek olduğu görülebilir. Şekil 5.6'da UKSB tabanlı uyarlamalı kontrolörün UKSB tabanlı sistem tanımlama modeline uygulanması sonucunda oluşan çıkışlar gösterilmiştir.

Burada daha önceden eğitilen UKSB tabanlı model sürekli olarak girişleri ve sistemin eski değerlerini alarak bir sonraki çıkışı üretir ve sistem gibi davranır. Temel prensibi sistem verilerini kullanarak sistem matematiksel modeli gibi davranmaktır. Ancak eğitim sırasında kayıplar olacağı aşikar olduğundan hiçbir zaman tam olarak sistem gibi davranamaz ancak yakınsaklık sağlar. Kontrolör performansı da bu kısımda modelin ne kadar yakınsaklık sağlayabildiğine bağlı olarak değişir.



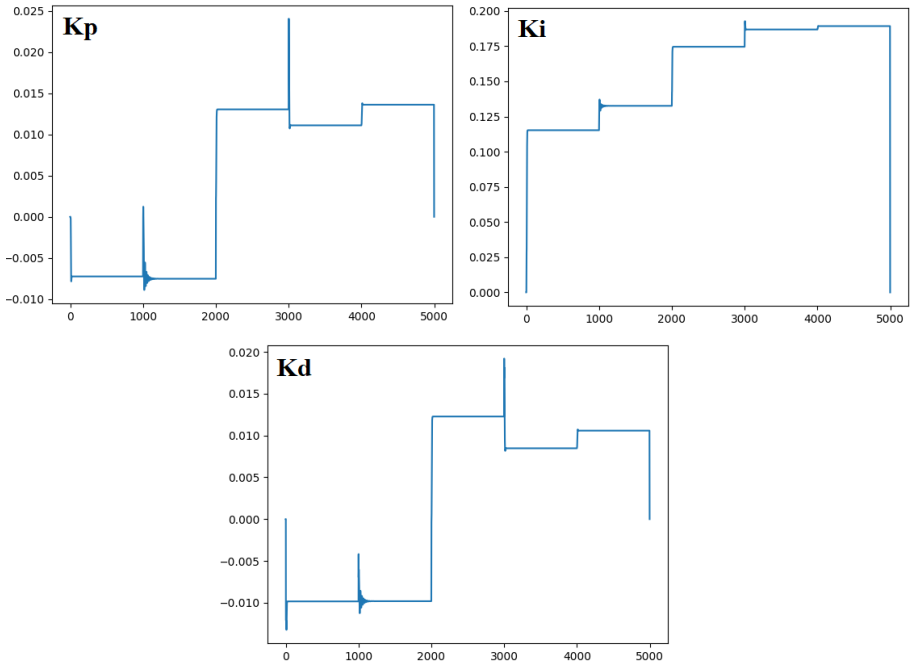
Şekil 5.6 : UKSB Tabanlı Uyarlamalı kontrolör ile kontrol edilen sistem çıkışları(Model UKSB Tabanlı Sistem Tanımlamadan elde edilmiştir.)

Şekil 5.7'de ise aşım ve yerleşme zamanını daha iyi görebilmek adına yakınlaştırılmış hali gösterilmiştir. Bu kısımda matematiksel model kullanılmadığından dolayı işaretin bazı bölümlerinde kontrolör gürültüsünün arttığı görülmektedir. Bazı referans değerlerine yerleşirken kontrolör salınımı artabilir.



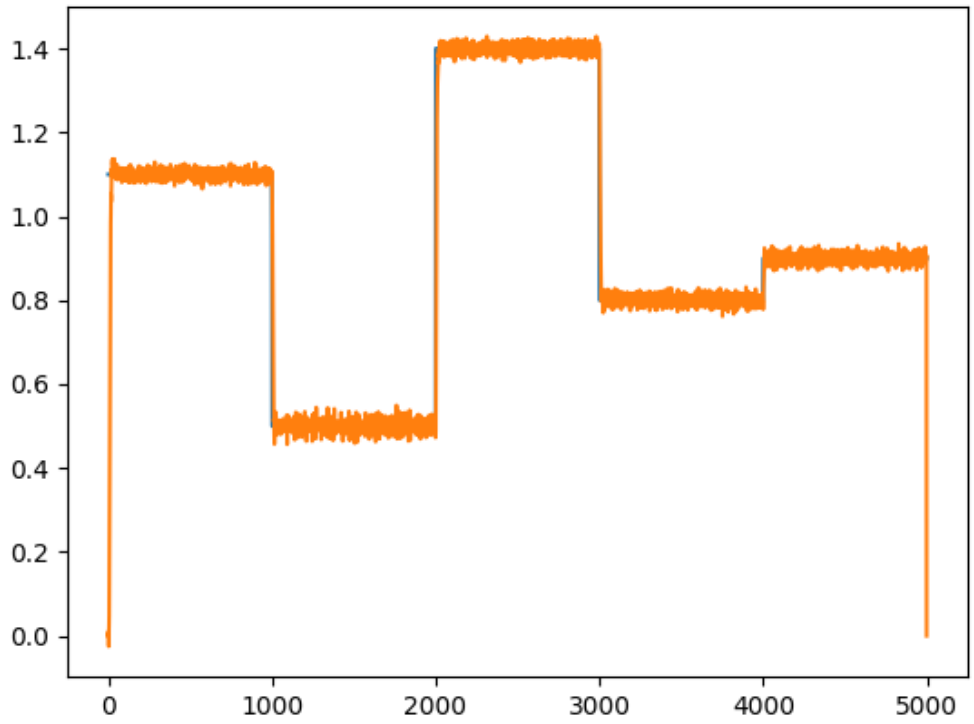
Şekil 5.7 : UKSB Tabanlı Uyarlamalı kontrolör ile kontrol edilen sistem çıkışları yakınlaştırılmış hali

UKSB tabanlı uyarlamalı kontrolörün K_p, K_i ve K_d değerlerinin değişimleri Şekil 5.8'de gösterilmiştir. Dikkat edilirse değişim noktalarının referans sinyalin değişim noktalarına denk geldiği gözlemlenebilir.



Şekil 5.8 : UKSB Tabanlı Uyarlamalı kontrolör K_p, K_i ve K_d çıkışları

Gerçek hayat uygulamalarında çıkışların geri beslemesi için kullanılan sensörler içerisinde bir gürültü bulunmaktadır. Bu gürültüden dolayı geri besleme tam anlamıyla doğru değerler içermeyebilir. UKSB tabanlı kontrolörün daha güçlü bir kontrolör olarak sınıflandırılabilmesi için bu sensör gürültüsünün de hesaba katılması gerekmektedir. Test sistemi için 40 dB'lik bir beyaz gürültü çıkışa eklenerek geri besleme bu gürültü ile gerçekleştirilmiştir. Bu aşamada sistem çıktısının referans sinyali bu gürültü altında da takip edebilmesi beklenmektedir. Şekil 5.9'de 40 dB beyaz gürültü içeren bir çıkış ölçümü altında kontrolör davranışı gösterilmiştir. Dikkat edilirse referans sinyal yine başarı ile takip edilebilmektedir. Gerçek hayattaki uygulamalarda sensörler gürültü içerdiği için çıkışlar yaklaşık olarak bu çıkışlara benzemektedir. Gürültü değerinin çok arttığı durumlarda kontrolörün başarımını giderek kaybettiği görülecektir. Kontrol uygulamalarında sensör kalitesinin önemi de bu kısımda anlaşılmaktadır.



Şekil 5.9 : UKSB Tabanlı Uyarlamalı kontrolör ile kontrol edilen sistem çıkışları (40 dB ölçüm gürültüsü içermektedir)

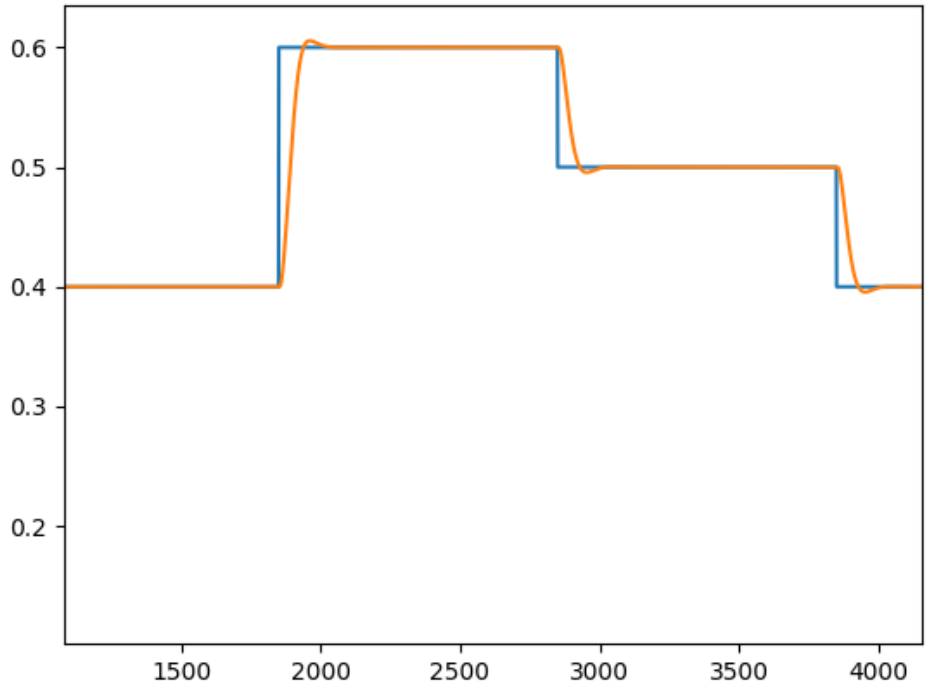
5.3.2 Doğrusal olmayan CSTR sistemi sonuçları

UKSB tabanlı kontrolör ikinci olarak doğrusal olmayan CSTR sistemi üzerinde test edilmiştir. Bu sistem için direkt olarak bir önceki bölümde elde edilen UKSB tabanlı

sistem tanımlama modeli kullanılmıştır. Şekil 5.10'da UKSB tabanlı kontrolör, UKSB tabanlı sistem tanımlama modeline uygulandığında oluşan çıktılar gösterilmiştir. Sistem çıktısının referans işareti ne kadar yakından takip ettiği incelenebilir ve bu sonuç UKSB tabanlı kontrolörün başarısını göstermektedir. Doğrudan matematiksel ifadesi elimizde olan sistemler için de UKSB tabanlı kontrolör tercih edilebilir. Bunun yanı sıra mimari değiştirilerek daha farklı sonuçlar elde etmek de mümkündür. Uygulama kapsamında kullanılan mimari deneysel sonuçlar ile tercih edilmiş ve fazla karmaşık olmayan bir mimaridir. Daha kompleks sistemlerde daha karmaşık mimariler tercih edilebilir ancak karmaşıklık arttıkça öğrenme her zaman doğru orantılı artmayacağı unutulmaması gereken bir husustur. Mimari seçimlerinde ne fazla basit ne de fazla karmaşık yapılar tercih edilmelidir.

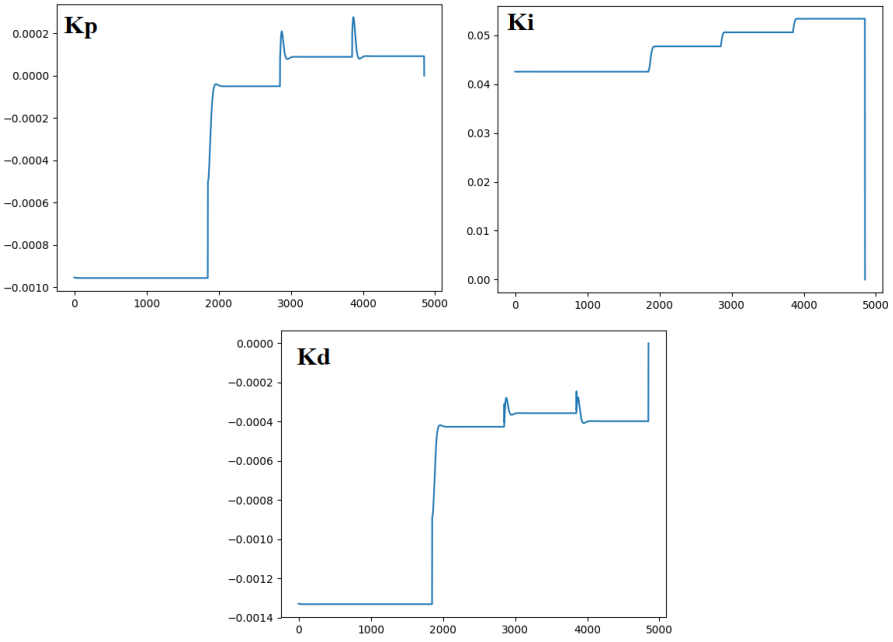
Şekil 5.10 : UKSB Tabanlı Uyarlamalı kontrolör ile kontrol edilen sistem çıkışları(Model UKSB Tabanlı Sistem Tanımlamadan elde edilmiştir.)

Şekil 5.11'de ise aşım ve yerleşme zamanını daha iyi görebilmek için yakınlaştırılmış hali gösterilmiştir. Bu kısımda matematiksel model kullanmak yerine direkt olarak bir önceki bölümde eğitilen CSTR sistem tanımlama modelinin kullanıldığına dikkat edilmesi gerekir. Referans sinyalin başarılı bir şekilde takip edilmesinin nedenlerinden biride sistem tanımlama başarısıdır.



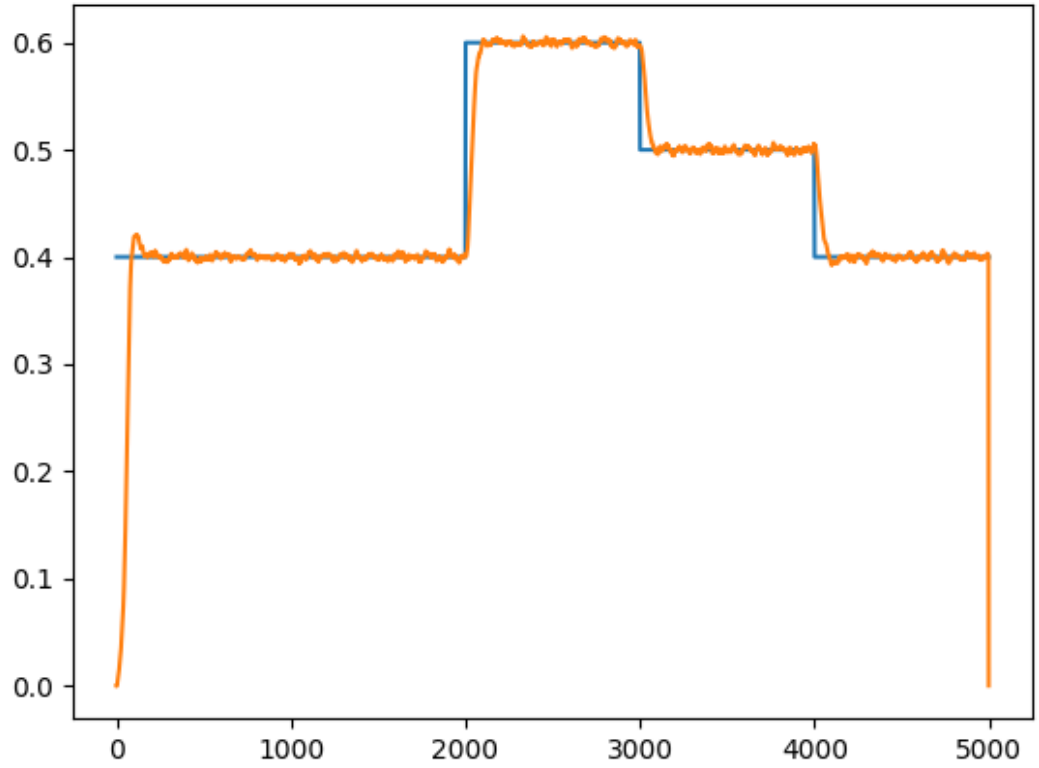
Şekil 5.11 : UKSB Tabanlı Uyarlamalı kontrolör ile kontrol edilen sistem çıkışları yakınlaştırılmış hali

UKSB tabanlı uyarlamalı kontrolörün K_p, K_i ve K_d değerlerinin değişimleri Şekil 5.12'de gösterilmiştir. Dikkat incelenirse değişim noktalarının referans sinyalin değişim noktalarına denk geldiği gözlemlenebilir. Bu noktalarda hızlı cevap alabilmek için uyarlamalı kontrolör agresif davranır ve çıkış değerlerini günceller.



Şekil 5.12 : UKSB Tabanlı Uyarlamalı kontrolör K_p, K_i ve K_d çıkışları

Gerçek hayat uygulamalarında çıkışların geri beslemesi için kullanılan sensörler içerisinde bir gürültü bulunmaktadır. Bu gürültüden dolayı geri besleme tam anlamıyla doğru değerler içermeyebilir. UKSB tabanlı kontrolörün daha güçlü bir kontrolör olarak sınıflandırılabilmesi için bu sensör gürültüsünün de hesaba katılması gerekmektedir. Test sistemi için 60 dB'lik bir beyaz gürültü çıkışa eklenerek geri besleme bu gürültü ile gerçekleştirilmiştir. Bu aşamada sistem çıktısının referans sinyali bu gürültü altında da takip edebilmesi beklenmektedir. Şekil 5.13'de 60 dB beyaz gürültü içeren bir çıkış ölçümü altında kontrolör davranışları gösterilmiştir. Dikkatli incelenirse referans sinyalin yine başarı ile takip edilebildiği gözlemlenebilir. Gerçek hayattaki uygulamalarda sensörler gürültü içerdiği için çıkışlar yaklaşık olarak bu çıkışlara benzemektedir. Gürültü değerinin çok arttığı durumlarda kontrolörün başarımını giderek kaybettiği görülecektir ve referans sinyali yeteri kadar iyi takip edilemeyecektir. Kontrol uygulamalarında sensör kalitesinin son derece önemi olduğu da bu kısımda anlaşılmaktadır.



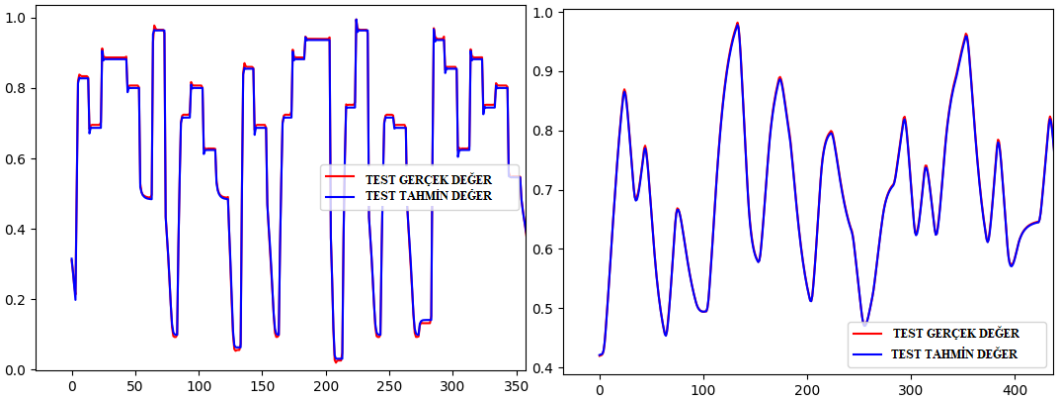
Şekil 5.13 : UKSB Tabanlı Uyarlamalı kontrolör ile kontrol edilen sistem çıkışları (60 dB ölçüm gürültüsü içermektedir)

6. SONUÇ VE ÖNERİLER

6.1 Sonuçlar

Çalışma iki aşamalı bir çalışma olduğu için çıktıları da iki aşamalı olarak değerlendirmek gerekir. Çalışmanın ilk kısmında elimizde giriş ve çıkış verileri olan bir sistemin, makine öğrenmesi algoritmalarından olan UKSB ile tanımlanması ve matematiksel model yerine eğitilmiş UKSB kullanılması amaçlanmıştır. İkinci kısmında ise sistem yerine kullanılan UKSB yapısının yine UKSB tabanlı bir kendinden ayarlamalı PID kontrolör ile kontrolü gerçekleştirilmiştir. Çalışmanın sonuçlarını bu iki perspektif üzerinden inceleyerek başarı oranlarını gözlemleyebiliriz, ardından çalışmadaki asıl amacımız olan "*elimizde herhangi bir matematiksel model olmadan sistem verilerinden derin öğrenme yardımıyla modelinin oluşturulması*" ve "*derin öğrenme algoritmaları ile yeni bir kontrol yöntemi geliştirilmesi*" kavramlarını ne kadar gerçekleştirebildiğimizi incelenebiliriz.

İlk kısımda gerçekleştirdiğimizi UKSB tabanlı sistem tanımlamanın Şekil 6.1'de solda doğrusal olmayan test sistemi ve sağda doğrusal olmayan CSTR sistemi test sonuçlarının yakınlaştırılmış halleri incelenirse gerçek sonuçlar ile neredeyse birebir sonuçlar ürettiği gözlemlenebilir.



Şekil 6.1 : Doğrusal olmayan test sistemi ve CSTR sistem tanımlama sonuçları yakınlaştırılmış halleri

Şekil 6.1'de test sonuçlarının verilmesinin nedeni ise eğitim sonuçlarının aldatıcı olabilmesi ve test verilerini, UKSB tabanlı eğitilmiş sistemin hiç görmemesidir.

Buradan anlaşılacağı üzere UKSB tabanlı eğitilmiş sistem hiç görmediği veriler üzerinde bile sistem gibi davranabilmektedir ve tıpkı matematiksel model gibi sistem için çıkışlar üretebilmektedir.

Eğitilmiş UKSB modelinin gerçek sisteme ne kadar yakın sonuçlar ürettiğini doğrusal olmayan test sistemi ve CSTR sistemi için sayısal olarak da inceleyebiliriz. Çizelge 6.1'da doğrusal olmayan test sistemi ve doğrusal olmayan CSTR sistemi için eğitim *ortalama kesin hata*, *ortalama karesel hata*, *medyan kesin hata*, *açıklanan varyans değer* ve *R2 değer* sonuçları verilmiştir.

Çizelge 6.1 : Örnek sistemler üzerinde eğitim başarı ölçütleri

Başarı Ölçütü	**Örnek Sistem**	**CSTR Sistem**
Ortalama Kesin Hata	0.005867	0.002369
Ortalama Karesel Hata	5.592064	6.6870382
Medyan Kesin Hata	0.0054014	0.002246417
Açıklanan Varyans Değer	0.99929	0.999960
R2 Değer	0.999435	0.999981

Benzer şekilde Çizelge 6.2'de doğrusal olmayan test sistemi ve doğrusal olmayan CSTR sistemi için test*ortalama kesin hata*, *ortalama karesel hata*, *medyan kesin hata*, *açıklanan varyans değer* ve *R2 değer* sonuçları verilmiştir.

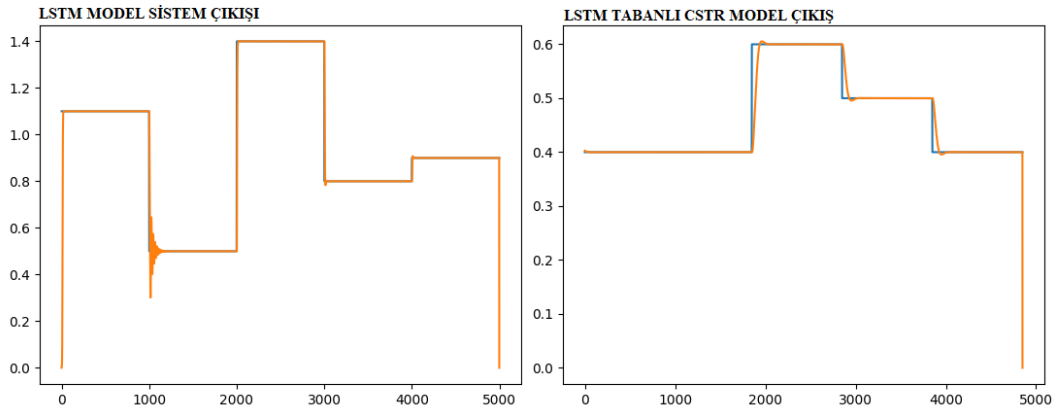
Çizelge 6.2 : Örnek sistemler üzerinde test başarı ölçütleri

Başarı Ölçütü	**Örnek Sistem**	**CSTR Sistem**
Ortalama Kesin Hata	0.0058556	0.0023907
Ortalama Karesel Hata	5.730182	7.027309
Medyan Kesin Hata	0.0053893	0.00224684
Açıklanan Varyans Değer	0.999345	0.999959
R2 Değer	0.9993332	0.999788

Yukarıdaki değerlerden anlaşılacağı gibi hata ölçütleri hem test hem de eğitim için oldukça küçük ve başarı değerleri 1'e yakındır. Bu durumda UKSB tabanlı

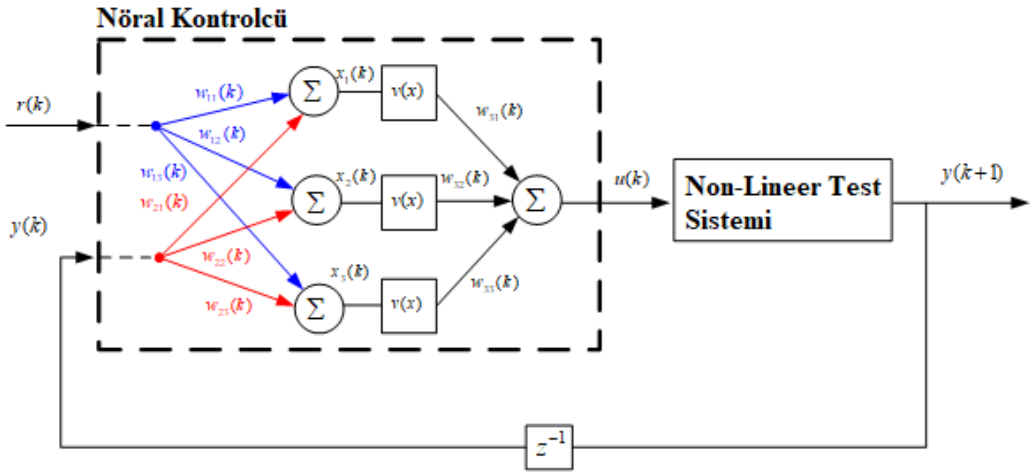
sistem tanımlama başarıya ulaşmıştır ve gerçek sistem modeli yerine UKSB modeli kullanılabilir.

İkinci kısımda ise eğitilen UKSB modeli için yine UKSB tabanlı kendi kendini ayarlayan bir PID kontrolör geliştirilmiş ve sistemin çıktısının referans sinyali olabildiğince yakın izlemesi amaçlanmıştır. Geliştirilen UKSB tabanlı PID kontrolör, ilk adımda oluşturulan önceden eğitilmiş UKSB tabanlı sistem modeline uygulanmıştır. Şekil 6.2'de solda doğrusal olmayan test sistemi ve sağda doğrusal olmayan CSTR sistemi için sonuçlar verilmiştir.



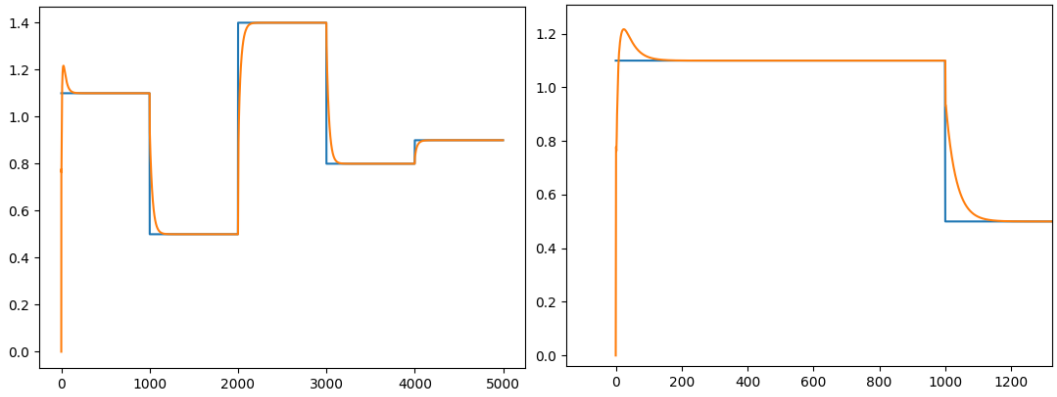
Şekil 6.2 : Doğrusal olmayan test sistemi ve CSTR UKSB tabanlı kontrolör sonuçları

Sonuçlar dikkatli incelenirse UKSB tabanlı uyarlamalı kontrolör sayesinde sistemin çıkışlarının referans sinyali oldukça yakın takip edebildiğini ve kontrolörün amacına uygun sonuçlar ürettiği gözlemlenebilir. Daha iyi bir sonuç çıkarabilmek için doğrusal olmayan test sistemi Şekil 6.3'deki gibi daha temel ve az sayıda nöron içeren temel bir makine öğrenmesi tabanlı uyarlamalı kontrolör ile karşılaştırılabilir.



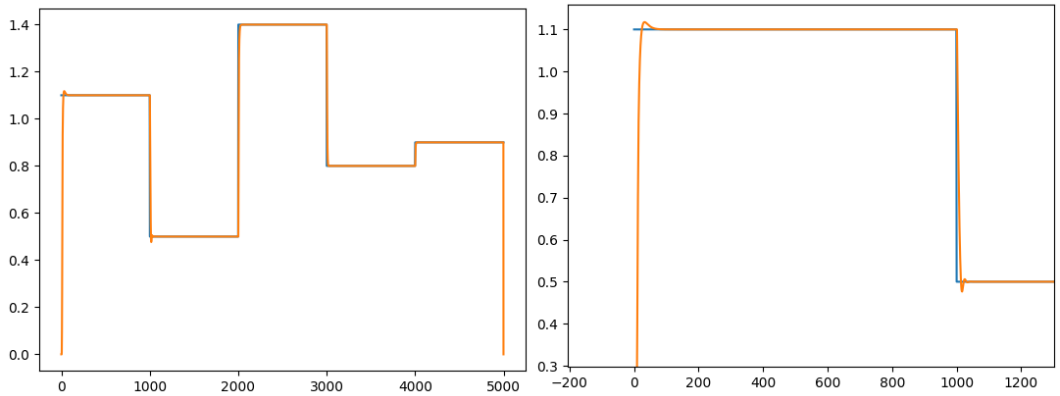
Şekil 6.3 : Temel yapay sinir ağı yapısı ile kontrol blok diyagramı

Temel makine ağ yapısı ile test sistemimizin referans bir işareti takip etmesi incelediğinde karşımıza Şekil 6.4'deki sonuçlar çıkacaktır. Bu yapıda test sistemi için matematiksel model kullanılmıştır. Temel makine ağ yapısı incelenirse tıpkı UKSB mimarisi gibi geçmiş çıkış değerlerini alarak referans sinyal ile hata ürettiği ve bu hata fonksiyonu ile ağırlıklarını güncellediği görülebilir. Bu yapı direkt olarak K_p, K_i ve K_d üretmez. Bunun yerine sistem için uygun giriş sinyalini üretmektedir. Bu giriş işareti örnek sistemin matematiksel modeline verilerek bir sonraki çıkış değeri hesaplanır.



Şekil 6.4 : Temel yapay sinir ağı yapısı ile kontrolör çıkışları ve yakınlaştırılmış hali

Aynı sistem çalışmada gerçekleştirdiğimiz UKSB tabanlı kontrolör ile kontrol edildiğinde Şekil 6.5'deki sonuçlar elde edilmektedir. Bu işlem esnasında yine sistemin matematiksel modeli kullanılmıştır. UKSB tabanlı sistem K_p, K_i ve K_d değerlerini üreterek bu değerler sonucunda sistem için uygun giriş işareti üretilmektedir. Bu kısımda 5.2'deki UKSB ağ yapısı kullanılmıştır.



Şekil 6.5 : UKSB yapısı ile kontrolör çıkışları ve yakınlaştırılmış hali

Burada her iki sonuç karşılaştırıldığında UKSB tabanlı kontrolörün yerleşme zamanının daha iyi olduğu ve referans sinyali büyük bir başarı ile takip edebildiği görülebilmektedir. Bu durumda ikinci amacımız olan derin öğrenme tabanlı uyarlamalı

kontrolörler için yeni bir yöntem amacı da UKSB ile gerçekleştirilmiş olur. Çizelge 6.3'de her iki yöntemin performansları süre olarak verilmiştir. UKSB tabanlı kontrolör daha uzun sürede yanıt üretebilmektedir ancak yerleşme zamanı yapay sinir ağı tabanlı kontrolöre göre daha iyidir. Süreler karşılaştırıldığında yöntemler arasında çok büyük farklar olmadığı gözlemlenebilir. Sonuç olarak çalışma, ilk başlarda verilen her iki amacı da sağlamaktadır.

Çizelge 6.3 : Yapay sinir ağı tabanlı kontrolör ve UKSB kontrolör zaman karşılaştırmaları

Metod	**Çalışma Zamanı (sn)**
Yapay sinir ağı tabanlı Kontrolör	0.20125
UKSB Tabanlı Kontrolör	0.23867

6.2 Öneriler

Çalışma kapsamında kullanılan UKSB mimarisinin geliştirilmesi mümkündür. Daha karmaşık problemlerde daha fazla katmanlı bir yapı kullanılarak daha iyi sonuçlar elde edilebilir ancak bu durum işlem yükünü arttıracaktır. İlerleyen çalışmalar kapsamında UKSB mimarisi değiştirilerek GRU UKSB mimarisi getirilebilir ve örnek sistemler üzerinde sonuçları incelenebilir. Yeni bir mimari geliştirilmesinin beraberinde hata fonksiyonuna bağlı yeni çıkarımlar gerektireceği unutulmamalıdır.

Başka bir yöntemde ise yine derin öğrenme algoritmalarından CNN (Convolutional Neural Network) yani Evrişimsel Sinir Ağları kullanılabilir. Bu yapılar genellikle matris tipi yapılarda (özellikle görüntü verilerinde) başarılı sonuçlar üretmektedir. Ancak veriler tek boyutlu matrislere dönüştürülerek örnek sistemler üzerinde kontrol uygulamaları denenebilir.

Makine öğrenmesi uygulamalarında en karmaşık sistem her zaman en iyi sonucu üretmez. Ockham'ın usturasının da ortaya koyduğu gibi zorunluluk olmadıkça en kolay yöntemi seçmek gerekmektedir. [48] Tercih edilecek ve seçilecek yöntem amaca uygun ve en basit yöntem olmalıdır. Makine öğrenmesi uygulamalarında karmaşık sistemlerin öğrenmekten çok ezberlemeye yönelimli olduğu unutulmamalıdır. Ayrıca karmaşık sistemler, işlem yüklerinden dolayı geç yanıt verdiklerinden kontrol

sistemlerinin hızına yetişemeyebilir. Bu gibi kavramların yöntem tasarlanırken göz önünde bulundurulması gerekmektedir.

KAYNAKLAR

[1] **Url-4**, Geometric methods for nonlinear state-space system identification, https://www.imperial.ac.uk, https://www.imperial.ac.uk/electrical-engineering/research/control-and-power/research/geometric-methods-for-nonlinear-state-space-system-identification.

[2] **Url-5**, Black box, https://www.wikiwand.com/en/Black_box.

[3] **Karel J. Kessman**, (2011). System Identification.An Introduction, Springer.

[4] **Fazlina Ahmat Ruslan**, **Abd. Manan Samad ve Ramli Adnan** (2013). Flood prediction using NARX neural network and EKF prediction technique: A comparative study.

[5] **Petre Stoica ve Torsten Söderström**, (1989). System Identification.Prentice Hall International Series in Systems and Control Engineering, Springer, London.

[6] **Debadatta Swain**, **MM Ali ve Robert Weller** (2006). Estimation of mixed-layer depth from surface parameters.

[7] **Url-7**, (2015), Dünyayı Değiştirmekte Olan Yapay Sinir Ağları Nedir?, https://bilimfili.com/dunyayi-degistirmekte-olan-yapay-sinir-aglari-nedir/.

[8] **Url-9**, Understanding LSTM Networks, https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[9] **Larisa Besic**, **Imer Muhovic**, **Adna Asic**, **Aida Catic**, **Lejla Gurbeta ve Almir Badnjevic** (2017). Application of neural networks to the prediction of a phenotypic trait of pacific lampreys based on single nucleotide polymorphism (SNP) genetic markers.

[10] **Antoni Wysocki ve Maciej Lawrynczuk** (2015). Jordan Neural Network for Modelling and Predictive Control of Dynamic Systems.

[11] **Aparajita Dutta** (2014). Hybridization of Recurrent Neural Network and Bacterial Foraging Optimization for Inferring Gene Regulatory Network.

[12] **Pranjal Srivastava** (2017). Essentials of Deep Learning : Introduction to Long Short Term Memory.

[13] **Url-16**, Derin Öğrenme İçin Aktivasyon Fonksiyonlarının Karşılaştırılması, https://medium.com/@ayyucekizrak/derin-%C3%B6%

C4%9Frenme-i%C3%A7in-aktivasyon-fonksiyonlar%C4%B1n%C4%B1n-kar%C5%9F%C4%B1la%C5%9Ft%C4%B1r%C4%B1lmas%C4%B1-cee17fd1d9cd.

[14] **Kemal Uçak ve Gülay Öke Günel** (Dec 2017). Fuzzy PID type STR based on SVR for nonlinear systems, 5.

[15] **Bobal, V.**, **Bohm, J.**, **Fessl, J. ve Machacek, J.**, (2005). Digital self-tunning controllers.Advanced textbooks in control and signal processing., Springer, London, s.5–9.

[16] **Ioan Dore, L.**, **Rogelio, L.**, **Mohammed, M. ve Alireza, K.**, (2011). Adaptive Control.Algorithm, Analysis and Applications, Springer, London, s. 1.

[17] **Kemal Uçak ve Gülay Öke Günel** (August 2015). Generalized self-tuning regulator based on online support vector regression, 1.

[18] **Tamar Flash ve Neville Hogan** (July 1985). The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model, 2.

[19] **Chunting Zhou**, **Chonglin Sun**, **Zhiyuan Liu ve Francis C.M. Lau** (Nov 2015). A C-LSTM Neural Network for Text Classification, 1.

[20] **Gang Liu ve Jiabao Guo** (April 2019). Bidirectional LSTM with attention mechanism and convolutional layer for text classification, 1.

[21] **Chuanhai Dong**, **Jiajun Zhang**, **Chengqing Zong**, **Masanori Hattori ve Hui Di** (Dec 2016). Character-Based LSTM-CRF with Radical-Level Features for Chinese Named Entity Recognition.

[22] **Pankaj Malhotra**, **Anusha Ramakrishnan**, **Gaurangi Anand**, **Lovekesh Vig**, **Puneet Agarwal ve Gautam Shroff** (Jul 2016). LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection.

[23] **Marijn F. Stollenga**, **Wonmin Byeon**, **Marcus Liwicki ve Jürgen Schmidhuber** (2015). Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation.

[24] **Fiorato Nicola**, **Yasutaka Fujimoto ve Roberto Oboe** (2018). A LSTM Neural Network applied to Mobile Robots Path Planning.

[25] **Yu Wang** (May 2017). A new concept using LSTM Neural Networks for dynamic system identification.

[26] **Samuel Park**, **Eric Patterson ve Carl Baum** (2019). Long Short-Term Memory and Convolutional Neural Networks for Active Noise Control.

[27] **Jesús Gonzalez ve Wen Yu** (June 2018). Non-linear system modeling using LSTM neural networks.

[28] **Url-1**, (2017), Time series prediction using ARIMA vs LSTM, https://datascience.stackexchange.com/questions/12721/time-series-prediction-using-arima-vs-lstm.

[29] **Url-2**, (2008), Mathematical model, https://en.wikipedia.org/wiki/Mathematical_model.

[30] **Url-3**, System Identification, https://en.wikipedia.org/wiki/System_identification.

[31] **Nikolaos Dervilis**, (2017). Special Topics in Structural Dynamics, Volume 6, Springer.

[32] **Paolo Arena**, **Luigi Fortuna**, **Giovanni Muscato ve Maria Gabriella Xibilia**, (1998). Neural Networks in Multidimensional Domains, Springer.

[33] **H.S.Niranjana Murthy** (2019). A NARX Model to Predict Myocardial Ischemic Beats from ECG Using Features Extracted by ICA and WPD.

[34] **Url-6**, (2018), Use Readily Available Components to Generate Pseudo-Random Binary Sequences and White Noise, https://www.digikey.nl/nl/articles/techzone/2018/mar/use-readily-available-components-generate-binary-sequences-white-noise.

[35] **Ethem Alpaydın**, (2012). Yapay Öğrenme, Boğaziçi Üniversitesi Yayınevi.

[36] **Mustafa Furkan Keskenler ve Eyüp Fahri Keskenler** (Aralık 2017). Geçmişten Günümüze Yapay Sinir Ağları ve Tarihçesi.

[37] **Url-8**, Makine Öğrenmesi, https://medium.com/.

[38] **Ralf C. Staudemeyer ve Eric Rothstein Morris** (Sep 2019). Understanding LSTM a tutorial into Long Short-Term MemoryRecurrent Neural Networks.

[39] **Url-10**, (1995), Recurrent Neural Networks, https://www.sciencedirect.com/topics/chemical-engineering/recurrent-neural-networks.

[40] **Url-11**, The curious case of the vanishing and exploding gradient, https://medium.com/learn-love-ai/the-curious-case-of-the-vanishing-exploding-gradient-bf58ec6822eb/.

[41] **Url-17**, LSTM, https://en.wikipedia.org/wiki/Long_short-term_memory#:~:text=1997%3A%20LSTM%20was%20proposed%20by%20Sepp%20Hochreiter%20and%20J%C3%BCrgen%20Schmidhuber.

[42] **Url-12**, [Back to Basics] Deriving Back Propagation on simple RNN/LSTM (feat. Aidan Gomez), https://towardsdatascience.com/back-to-basics-deriving-back-propagation-on-simple-rnn-lstm-feat-aidan-gomez-c7f286ba973d.

[43] **Url-13**, Only Numpy: Deriving Forward feed and Back Propagation in Long Short Term Memory (LSTM) part 1, https://towardsdatascience.com/only-numpy-deriving-forward-feed-and-back-

propagation-in-long-short-term-memory-lstm-part-1-4ee82c14a652.

[44] **Ye, A.**, A Guide to Neural Network Layers with Applications in Keras, https://towardsdatascience.com/a-guide-to-neural-network-layers-with-applications-in-keras-40ccb7ebb57a.

[45] **Url-14**, Dense layers explained in a simple way, https://medium.com/datathings/dense-layers-explained-in-a-simple-way-62fe1db0ed75.

[46] **Url-15**, Keras LSTM, https://keras.io/layers/recurrent/.

[47] **Ali Zribi**, **Mohamed Chtourou ve Mohamed Djemel** (2015). A New PID Neural Network Controller Design for Nonlinear Processes.

[48] **R.T.Carroll** (2001). Skeptic's Dictionary: Occam's razor, Available.

ÖZGEÇMİŞ

Ad Soyad: Çağatay Sanatel

Doğum Tarihi ve Yeri: 01.06.1993 İstanbul

E-Posta: cagataysanatel@gmail.com

ÖĞRENİM DURUMU:

- **Lisans:** 2016, Sakarya Üniversitesi, Mühendislik Fakültesi, Elektrik ve Elektronik Mühendisliği
- **Lisans:** 2016, Sakarya Üniversitesi, Bilgisayar ve Bilişim Fakültesi, Bilgisayar Mühendisliği
- **Y. Lisans:** 2020, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Mekatronik Mühendisliği.

MESLEKİ DENEYİMLER VE ÖDÜLLER:

- 2015-2016 yılları arasında Sakarya Üniversitesi Bilgisayar Laboratuvarları'nda gömülü sistemler üzerine çalıştı.
- 2016-2018 yılları arasında Verisun Teknoloji şirketinde donanım ve yazılım mühendisi olarak çalıştı.
- 2018 yılından itibaren TUSAŞ şirketinde yazılım mühendisi konumunda görev yapmaktadır.