T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DERİN PEKİŞTİRMELİ ÖĞRENME İLE ROBOT KOL TORK KONTROLÜ

YÜKSEK LİSANS TEZİ

MUHAMMED RAŞİT EVDÜZEN

DENİZLİ, TEMMUZ - 2021

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DERİN PEKİŞTİRMELİ ÖĞRENME İLE ROBOT KOL TORK KONTROLÜ

YÜKSEK LİSANS TEZİ

MUHAMMED RAŞİT EVDÜZEN

DENİZLİ, TEMMUZ - 2021

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

MUHAMMED RAŞİT EVDÜZEN

ÖZET

DERİN PEKİŞTİRMELİ ÖĞRENME İLE ROBOT KOL TORK KONTROLÜ

YÜKSEK LİSANS TEZİ

MUHAMMED RAŞİT EVDÜZEN

PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

(TEZ DANIŞMANI: PROF.DR.SERDAR İPLİKÇİ)

DENİZLİ, TEMMUZ - 2021

Robot teknolojisini geliştirmek için literatürde ve endüstriyel uygulamalarda model tabanlı yaklaşım kullanılmaktadır. Model tabanlı yaklaşıma alternatif olarak makine öğrenmesinin gelişmesi ve akıllı algoritmalar oluşturulmasıyla her alanda makine öğrenmesi kullanılmaya başlanmıştır. Robotik sistemlerin geliştirilmesi sürecinde sistemin matematiksel modelinin oluşturulması ve kontrol algoritmalarının geliştirilmesi gerekmektedir. Geliştirilen matematiksel modellerin fiziksel sistemleri ifade etmesi gerekmektedir fakat model üzerindeki belirsizlik, analitik çözümün olmaması gibi durumlarda model tabanlı kontrol algoritmaları beklenen performansı üretmemektedir. Pekiştirmeli öğrenme yaklaşımı kullanılarak robot kolun kontrol problemi çözülebilmektedir. Bu tez çalışmasında 3 eksenli seri manipulator (RRR) tipi bir robot kolun tork kontrol problemi incelenecektir. İncelenen robotun kontrol problemi klasik kontrol teorisi ve model tabanlı yaklaşım ile çözülüp, makine öğrenmesinin bir alt kolu olan pekiştirmeli öğrenme algoritmasıyla çözülecektir. Pekiştirmeli öğrenme algoritmalarından Q-öğrenme ve Sarsa öğrenme metodları incelenip ayrık durum ve aksiyon için kontrol problemleri çözülecektir. Robot kontrol problemi sürekli zamanlı olduğu için pekiştirmeli öğrenmenin geleneksel algoritmaları çözüm üretememektedir. Pekiştirmeli öğrenme algoritmalarının sürekli durum ve aksiyon için geliştirilmiş algoritmaları bu tez kapsamında incelenecektir. Derin Q-öğrenme algoritması ile sürekli durum için kontrol problemi çözülecektir. Sürekli durum ve aksiyon için geliştirilen derin deterministik politika gradyanı (DDPG) algoritması ile sürekli zamanlı robot tork kontrol problemi çözülecektir. Sonuç olarak MATLAB ve PYTHON ortamında geliştirilen algoritma ile robotun kontrol problemi çözülüp, robotik sistemlerde oluşan model tabanlı yaklaşımların problemleri pekiştirmeli öğrenme metodu kullanılarak çözülmektedir.

ANAHTAR KELİMELER: Pekiştirmeli Öğrenme, Markov Karar Süreci, Robot Kinematiği, Robot Dinamiği, Optimal Kontrol Teorisi, Derin Yapay Sinir Ağları,Derin Deterministik Politika Gradyanı

ABSTRACT

ROBOTIC ARM TORQUE CONTROL VIA DEEP REINFORCEMENT LEARNING
MSC THESIS
MUHAMMED RAŞİT EVDÜZEN
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
ELECTRICAL AND ELECTRONICS ENGİNEERING
(SUPERVISOR: PROF.DR.SERDAR İPLİKÇİ)

DENİZLİ, JULY 2021

The model based approach is used to develop robot technology in literature and industrial applications. Machine learning has started to be used in forming smart algorithms and the development of machine learning, as an alternative to the model based approach. During the development of robotic systems, the forming of mathematical models and the development of control algorithms are needed. Developed mathematical models need to reflect physical systems but uncertainty on the model, and situations where analytic situations are not found, causes unexpected performances in model based control algorithms. Problems with robotic arm control can be solved by using the reinforcement learning approach. In this thesis we'll be analyzing the problem of torque control, in an three axis serial manipulator (RRR) type robot arm. The control problem of the robot arm under analysis, is solved using classical control theory and the model based approach, thus this problem can be solved by using the reinforcement learning technique, which is based on machine learning. The problem of discrete state and action control is solved by using, Q learning and Sarsa learning methods, which are a part of reinforcement learning algorithms. As the robot control problem is continuous time the traditional algorithm of reinforcement learning is unable to form solutions. In this thesis we will be analysing reinforcement learning algorithms, which are developed for continuous time state and action. The control problem for, continuous time state, will be solved by using the deep Q learning algorithm. The continuous time robot torque control problem, will be solved, by using the deep deterministic policy gradient (DDPG) algorithm, which is developed for continuous state and action. As a result the robots' control problem can be solved by using algorithms which are developed in the MATLAB and PYTHON environment. Model based approach problems which are found in robotic systems can be solved by using the reinforcement learning method.

KEYWORDS: Reinforcement Learning, Markov Decision Process, Robot Kinematics, Robot Dynamics, Optimal Control Theory, Deep Neural Network, Deep Deterministic Policy Gradient

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

SEMBOL LİSTESİ

$\boldsymbol{\theta_i: i = 1, 2, 3, \ldots, n}$	:	Robot Kolun Eklem Açıları
${}^{i-1}_{i}\boldsymbol{T}:\ \boldsymbol{i = 1, 2, 3, \ldots, n}$	:	N Serbestlik Dereceli Sistem İçin Homojen 4*x*4 Transformasyon Matrisi
$\boldsymbol{{}^{0}P_{n}}$	:	N Serbestlik Dereceli Sistem İçin 3*x*1 Boyutunda Konum Vektörü
$\boldsymbol{{}^{0}R_{n}}$	:	N Serbestlik Dereceli Sistem İçin 3*x*3 Boyutunda Oryantasyon Matrisi
$\boldsymbol{O_c^0}$	:	Robot Kolun Merkezi ile Bileğini İfade Eden 3*x*1 Boyutunda Konum Vektörü
$\boldsymbol{J(\theta_1, \theta_2)}$	:	2 Eksenli Robot Kolun Jacobian Matrisi
$\|\|.\|\|$	:	Öklid Normu
$\|.\|$	:	Determinant
$\boldsymbol{t}$	:	Zaman Değişkeni
$\boldsymbol{V, \omega}$	:	Sırasıyla Çizgisel ve Açısal Hız
$\boldsymbol{\dot{x}}$	:	X'in Zamana Göre 1. Türevi
$\mathbb{R}^n, \mathfrak{R}^n$	:	N Boyutlu Reel Sayılar Uzayı
$\frac{d}{dx}$	:	İlgili Değişkenin X'e Göre 1. Türev Operatörü
$\frac{\partial}{\partial x}$	:	İlgili Değişkenin X'e Göre 1. Kısmi Türev Operatörü
$\boldsymbol{\tau_k}:\quad \boldsymbol{k = 1, 2, 3, \ldots, n}$	:	K Eksenli Dereceli Robot Kolun Eklemlerinde Oluşan Tork Değeri
$\boldsymbol{q_i}:\quad \boldsymbol{i = 1, 2, 3, \ldots, n}$	:	N Serbestlik Dereceli Robotun Genelleştirilmiş Koordinatlara Göre Her Bir Eklemin Açısal Konum Vektörü
$\boldsymbol{\dot{q}_i}:\quad \boldsymbol{i = 1, 2, 3, \ldots, n}$	:	N Serbestlik Dereceli Robotun Genelleştirilmiş Koordinatlara Göre Her Bir Eklemin Açısal Hız Vektörü
$\boldsymbol{J_m, J_l}$	:	Sırasıyla Motor ve Robot Kolun Atalet Momentleri
$\boldsymbol{B_m, B_l}$	:	Sırasıyla Motor ve Robot Kolun Viskoz Sürtünme Katsayısı
$\boldsymbol{T_s}$	:	Örnekleme Periyodu
$\boldsymbol{K_p, K_d, K_i}$	:	Robot Kol Kontrol Algoritması Parametreleri
$\boldsymbol{e = q_d - q}$ $\boldsymbol{\dot{e} = \dot{q}_d - \dot{q}}$	:	Robot Kolun Hedeflenen Konum ve Hız Değerleri ile Gerçekleşen Konum ve Hız Değerleri Arasındaki Hata Vektörü
$\boldsymbol{J}$	:	Optimal Kontrol Kuramı İçin Oluşturulan Maaliyet Fonksiyonu
$\mathbf{u}, \overset{*}{\boldsymbol{u}}$	:	Sırasıyla Kontrol ve Optimal Kontrol İşareti
$\boldsymbol{\nabla f(x), \nabla^2 f(x)}$	:	Sırasıyla $f(x)$ Fonksiyonunun Gradyan ve Hessian Matrisini Oluşturmaktadır

$\{\boldsymbol{X_i}, \boldsymbol{Y_i}\}$: $\boldsymbol{i1, 2, 3, \ldots, N}$	:	N Adet Giriş Çıkış Verisini İfade Etmektedir.
$\boldsymbol{H(x)}$	:	Yapay Sinir Ağı İçin Kullanılan Aktivasyon Fonksiyonu
R_t	:	Yapay Ajanın Çevreden Almış Olduğu Ödül Değeri
A_t	:	Yapay Ajanın Çevreye Uygulamış Olduğu Aksiyon Değeri
S_t	:	Yapay Ajanın Çevre İçinde Bulunduğu Durum
$P_{ss'} = P[S_{t+1} = s' \mid S_t = s]$	:	Markov Süreci Geçiş Olasılıkları Matrisi
γ	:	Azaltma Faktörü
$E[.]$	:	Beklenen Değer Operatörü
G_t	:	Gelecekteki Azaltılmış Toplam Ödül Değeri
$V(s)$	:	Durum Değer Fonksiyonu
π	:	Yapay Ajanın Uygulayacağı Politika
$Q(s, a)$	:	Aksiyon Değer Fonksiyonu
$V_*(s)$	:	Optimal Durum Değer Fonksiyonu
$Q_*(s, a)$	:	Optimal Aksiyon Değer Fonksiyonu
N	:	Sıfır Ortalamalı Birim Varyanslı Gauss Dağılımı

KISALTMALAR LİSTESİ

MPC	**:**	Model-Predictive Control
NMPC	**:**	Nonlinear Model-Predictive Control
PID	**:**	Proportional-Integral-Derivative
MP	**:**	Markov Processes
MRP	**:**	Markov Reward Processes
MDP	**:**	Markov Desicion Processes
LTI	**:**	Linear Time-Invariant
DNN	**:**	Deep Neural Network
RK	**:**	Runge Kutta
MIMO	**:**	Multiple Input Multiple Output
SISO	**:**	Single Input Single Output
DP	**:**	Dynamic Programming
RMSE	**:**	Root Mean Square Error
SSE	**:**	Sum Square Error
DDPG	**:**	Deep Deterministic Policy Gradient
D-H	**:**	Denavit-Hartenberg
SS	**:**	State Space
NSS	**:**	Nonlinear State Space
ADAM	**:**	Adaptive Moment Estimation
RL	**:**	Reinforcement Learning
DQN	**:**	Deep Q Network
LTV	**:**	Linear Time-Variant
LQR	**:**	Linear-Quadratic Regulator
CTC	**:**	Computed Torque Control
DOF	**:**	Degrees Of Freedom
HJB	**:**	Hamilton Jacobi Bellman
TD	**:**	Temporal Differencing
MC	**:**	Monte Carlo

IID	**:**	Independent and Identically Distributed Random Variables
PDF	**:**	Probability Density Dunction
CDF	**:**	Cumulative Distribution Function
PMF	**:**	Probability Mass Function
JPDF	**:**	Joint Probability Density Dunction
JPMF	**:**	Joint Probability Mass Function
POMDP	**:**	Partially Observable Markov Decision Process
YSA	**:**	Yapay Sinir Ağları
ANN	**:**	Artificial Neural Network
HOT	**:**	High Order Term

ÖNSÖZ

Lisans ve yüksek lisans eğitim hayatım boyunca bilgi ve tecrübeleri ile beni aydınlatan, yol gösteren, tez danışmanım çok değerli hocam sayın Prof.Dr.Serdar İPLİKÇİ'ye sonsuz teşekkürlerimi sunarım.

Eğitim hayatım boyunca desteklerini esirgemeyen çok kıymetli babaannem Zuhal BAKIRTAŞ'a ve annem Münire EVDÜZEN'e sonsuz teşekkürlerimi sunarım. Yüksek lisans eğitimim sırasında bilgileri ile beni aydınlatan kıymetli hocam Dr.Öğr.Üyesi.H.Hilal EZERCAN KAYIR'a teşekkürlerimi sunarım. Yüksek lisans tez yazım aşamasında manevi desteklerinden dolayı kıymetli nişanlım Huriye ŞENOL'a sonsuz teşekkürlerimi sunarım.

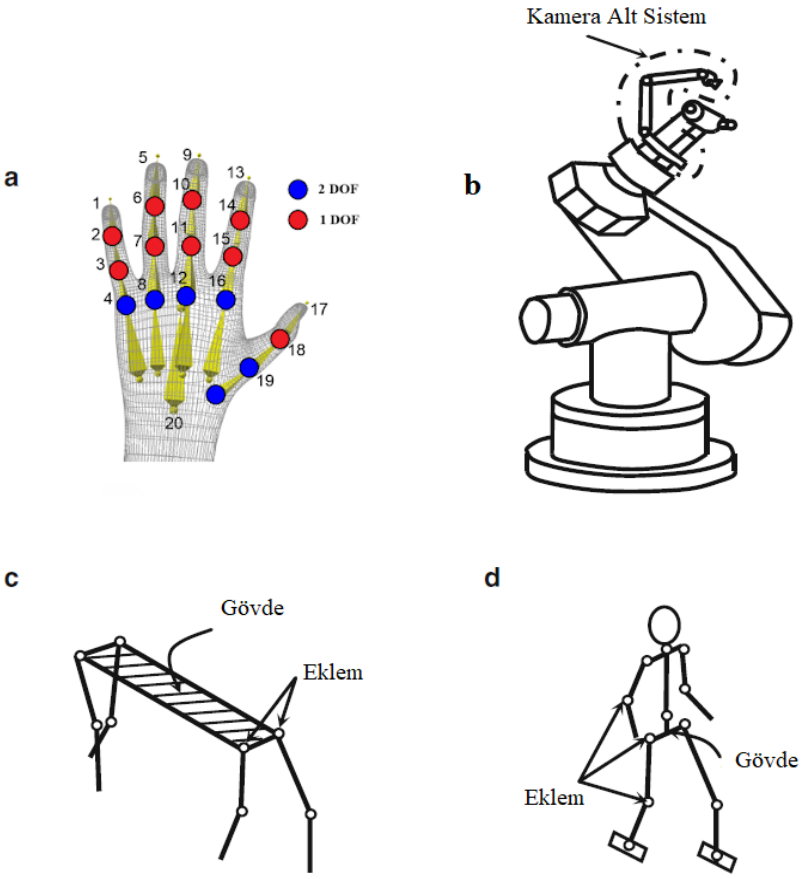
1. GİRİŞ

Makine öğrenmesinin hızla gelişmesi günümüzde birçok problemin çözümünü sağlamaktadır. Makine öğrenmesinin bir alt kolu olan pekiştirmeli öğrenme algoritmaları robotik, otonom araçlar, finans, biyoloji vb birçok alanda uygulanmaktadır. Pekiştirmeli öğrenme yaklaşımı son dönemde robot teknolojisiyle birleşerek, robotların karmaşık fiziksel yapılarına rağmen kontrol edilmesini ve gelişmesini sağlamaktadır. Bu tez çalışmasında pekiştirmeli öğrenme ve kontrol teorisi ile seri tip robot kolun tork kontrolü yapılmaktadır. Bölüm ikide robot kolun kontrol problemini çözmek için model tabanlı yaklaşım önerilmiştir. Bu başlık altında endüstriyel tip altı eksenli robot kol için ileri ve ters kinematik denklemlerin analitik ve sayısal çözümleri, robotun dinamik modelinin oluşturulması ve kontrol algoritmasıyla birleştirilmesi, son olarak optimal kontrol teorisi incelenmiştir. Üçüncü bölümde sayısal optimizasyon teorisi incelenip, makine öğrenmesi algoritması olan yapay sinir ağları ve derin yapay sinir ağlarının eğitim algoritması incelenmiştir. Oluşturulan derin yapay sinir ağları, derin pekiştirmeli öğrenme için bir alt yapı oluşturmaktadır. Üçüncü bölümde pekiştirmeli öğrenmenin temelleri incelenmiştir. Pekiştirmeli öğrenme algoritmalarıyla robot yörünge planlaması ve 3DOF robot kolun tork kontrol problemi çözülmüştür. Sonuç ve öneriler bölümünde pekiştirmeli öğrenmenin robotik ve otonom sistem teknolojisi için önemi vurgulanmıştır.

2. ROBOT KOL KONTROLÜ İÇİN MODEL TABANLI YAKLAŞIM

2.1 Endüstriyel Robot Teknolojisi

Robot teknolojisi otomobil, kimya, elektronik ve diğer birçok alanda kullanılmaktadır. Robot teknolojisinin gelişimi, açık kinematik zincir ile birbiri ardına seri bağlanan robotların geliştirilmesiyle sağlanmıştır. Seri tip manipülatörlerin gelişmesi robot teknolojisinde yeni tip robot mimarilerinin gelişmesine öncülük etmektedir. Yeni tip robot mimarileri çok parmaklı tutucu, insansı robot, ayaklı robot vb. şekil 2.1'de genel robot tipleri görülmektedir. Robot teknolojisi günümüzde birçok disiplini içerisinde barındırmaktadır. Bu disiplinlerden makine mühendisliği robotun kinematik ve dinamik modelinin oluşturulmasında, güç aktarma organı tasarımında, robotun gövdesinin üretilmesinde ve robotun işletme koşullarında, tasarımın uygunluğunun analizini oluşturmada önemli rol oynamaktadır. Elektrik elektronik mühendisliği robotun aktüatörlerinin tasarımı ve kontrolünde, robotun güç ünitesinde ve motor sürücülerinin tasarımında, kablolama sisteminin alt yapısının oluşturulmasında, robot üzerinde çalışacak gerekli yazılımların hazırlanmasında görev almaktadır. Bilgisayar mühendisliği robotun gerekli yazılımsal ve donanımsal alt yapılarının hazırlanmasında görev almaktadır. Matematik mühendisliği robotun karmaşık geometrisi ve buna bağlı olarak oluşturulacak algoritmaların matematiksel alt yapılarını sağlamak ve geliştirmek üzere görev almaktadır. Robot teknolojileri uygulama alanına göre diğer farklı mühendislik disiplinlerini içinde barındırarak günümüzde en popular araştırma alanlarından birisi haline gelmiştir. Sürekli gelişen bu teknoloji makine öğrenmesi ve yapay zeka çalışmalarıyla desteklenmektedir. Makine öğrenmesi algoritmalarının robotik sistemlere entegre edilmesiyle birlikte robotlar daha akıllı hale gelmiş ve kendi kendilerine çıkarım yaparak karar alıp, robotikte yürüme vb zor problemleri çözebilir hale gelmiştir. Makine öğrenmesinin gelişmesiyle birlikte günümüzde robot çalışmaları hızlanmakta ve her yeni gelişme ile birlikte yeni araştırma alanları oluşmaktadır (Saha ve diğ. 2013).

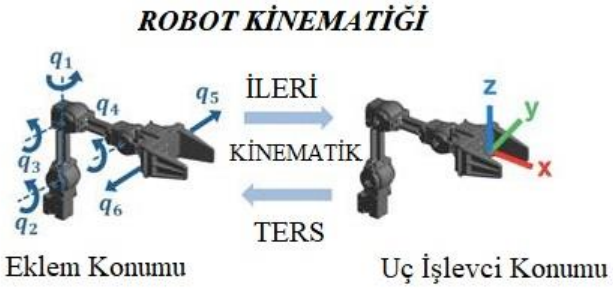


Şekil 2.1: Genel Robot Tipleri.

2.2 Robot Kinematiği

Kinematik bilimi fiziğin bir alt dalı olup birçok mühendislik problemini modellemek için kullanılmaktadır. Kinematik bilimi fiziksel olarak cisimlerin devinimleri ile ilgilenmektedir. Mühendislik problemlerini modellemek için genelde cisimlerin katı (rijit) yani şekil değiştirmediğini, sıvı yani parçacıklara uygulanan kayma gerilmeleri sonucunda cismin şekil değiştirdiği varsayımı kabul edilmektedir. Katı ve sıvı cisimlerin bağlı oldukları referans koordinat sistemine göre konum, hız, ivme ve jerk değerlini incelemek için kinematik bilimi kullanılmaktadır. Kinematik bilimi fiziksel cisimlerin bu özelliklerini incelerken harekete sebep olan iç veya dış kuvvetler ve momentlerle ilgilenmemektedir. Harekete sebep olan kuvvet ve momentleri inceleyen fiziğin alt dalı kinetik bilimi olarak karşımıza çıkmaktadır. Kinematik bilimi mühendislik problemlerini çözerken tasarlanan fiziksel sistemlerin analizinde kullanılmaktadır. Mühendisliğin bu tip problemler icin uygulandığı mekanizma tekniği, birbiri ile seri veya paralel bağlı katı mafsalların konum, hız, ivme ve jerk değerlerini analiz etmek icin kullanılmaktadır. Endüstriyel robot teknolojisinde

tasarlanan robotun tipine göre seri veya paralel manipulatör olarak karşımıza çıkmaktadır. Seri bir robot manipülatörü uç uca eklenen uzuvlardan oluştuğu ve her bir uzvun hareketinin bir diğerini etkilediği böylece robotun tabanından, ucuna kadar hareketin aktarıldığı robot konfigürasyonudur. Kinematik bilimi robotik sistemlerde ileri kinematik ve ters kinematik olarak iki konuda incelenmektedir. Bu tez kapsamında seri tip konfigürasyona sahip robotlar üzerinde çalışılmıştır. Bu kapsamda birbiri ardına eklenen uzuvların ileri kinematik analizi, robotun uzuvlarının bağlı olduğu eklem noktalarının ötelemeli veya dönmeli hareketine göre robotun uç noktasının, bulunduğu çalışma uzayı içerisindeki konum ve oryantasyonuyla ilgilenmektedir. Böylece robotun eklem değişkenlerinin değişimine bağlı olarak robotun uç kısmının uzayda konum ve oryantasyonu değişmektedir. Ters kinematik robotun bulunduğu çalışma uzayı içerisinde hedeflenen konum ve oryantasyonu gerçekleştirmesi için robotun eklem değişkenlerinin ne olması gerektiğini belirler. Bu durum şekil 2.2'de görülmektedir.

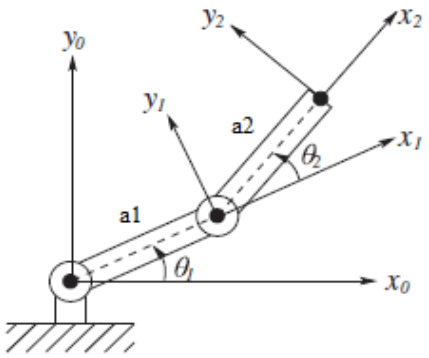


Şekil 2.2: Seri Robot Kolun İleri ve Ters Kinematik Diyagramı.

Robotların kinematik analizinin yapılması için literatürde farklı yöntemler bulunmaktadır. Takip eden alt bölümde robotun ileri kinematik analizi için Denavit – Hartenberg (DH) metodu kullanılacaktır. Bu metod ile robotun ileri kinematik problemi sistematik bir şekilde çözülmektedir. Robotların ileri kinematik çözümleri için geometrik yaklaşımlar da mevcuttur.

2.2.1 Robot İleri Kinematiği D-H Metodu

Bu başlıkta iki eksenli bir robot kolun geometrik ve altı eksenli endüstriyel tip seri robot manipülatörünün DH metodu ile ileri kinematik problemi çözülecektir. Çözümlenecek olan altı eksenli robot kol tez için önerilen modelden bağımsız metod olan pekiştirmeli öğrenme yaklaşımı için bir alt yapı sağlayacaktır. İki eksenli robot kolun ileri kinematik probleminin geometrik ifadesi şekil 2.3'de görülmektedir.



Şekil 2.3: 2 Eksenli Robot Kol Geometrik Şekli.

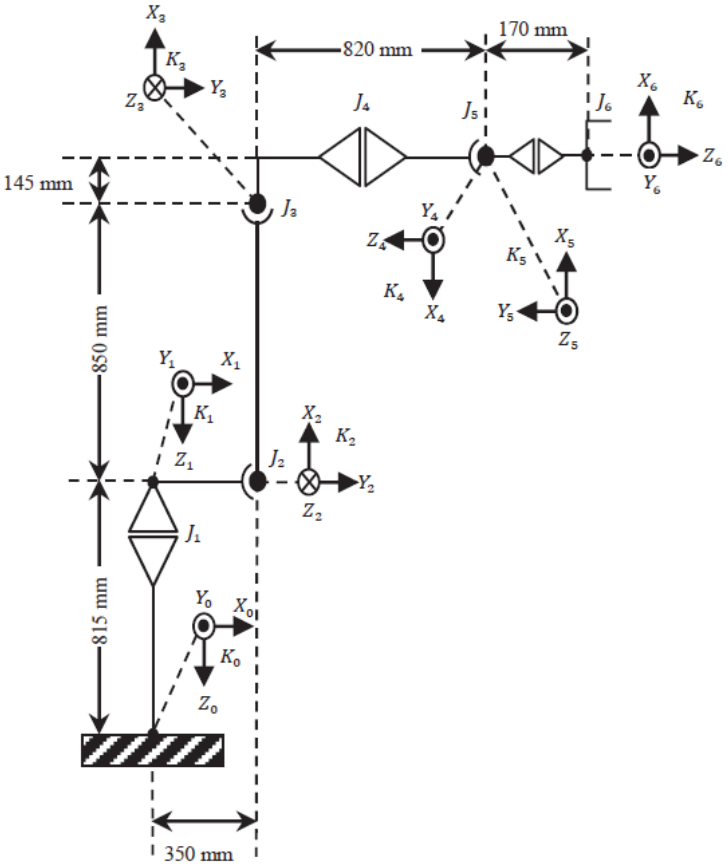
Robotun bağlı olduğu referans koordinat sistemi (x_0, y_0) ile ifade edilmiştir. Sırasıyla (x_1, y_1) robotun birinci uzvunun ucuna ve (x_2, y_2) robotun ikinci uzvunun ucuna eklenmiş koordinat sistemleridir. Amacımız eklem değişkenleri olan (θ_1, θ_2)'nin açısal değişimi sonucunda robotun uç işlevcisinin konumunu referans koordinat sistemine göre hesaplamaktır. Robotun ileri kinematik denklemlerinin geometrik olarak çözümü denklem (2.1) ile hesaplanmaktadır.

$$x_2 = a1\cos(\theta_1) + a2\cos(\theta_1 + \theta_2)$$
$$y_2 = a1\sin(\theta_1) + a2\sin(\theta_1 + \theta_2) \qquad (2.1)$$

Altı eksenli robot kolun ileri kinematik probleminin DH metodu ile çözümü denklem 2.3'deki gibidir. Robotların DH metodu ile ileri kinematik probleminin çözülmesi için robotun açık kinematik zincirine koordinat sistemleri sistematik bir şekilde yerleştirilmelidir. Yerleştirilen koordinat sistemlerine bağlı olarak DH parametreleri oluşturulmalı ve her bir link için DH parametresine bağlı olarak homojen 4x4 transformasyon matrisi yazılmalıdır. Yazılan transformasyon matrisleri birbiri ardına sağdan çarpım yapılarak robotun ileri kinematik problemi çözülmektedir.

DH metodu için kapsamlı çalışma Jun-Di Sun ve diğ. (2017). Bu tez kapsamında altı eksenli seri bir manipülatörün modellenmesi incelencektir. Şekil 2.4'de endüstriyel tip altı eksenli bir robot ve şekil 2.5'de robotun açık kinematik zinciri görülmektedir.

Şekil 2.4: Endüstriyel 6 Eksenli Seri Robot Manipülatörü.



Şekil 2.5: 6 Eksenli Seri Robot Manipülatörünün Açık Kinematik Zinciri.

Robotun açık kinematik zinciri üzerine yerleştirilen koordinat sistemleri ve robotun fiziksel parametreleri DH tablosuna eklenip sonrasında genel dönüşüm matrisi ile her bir uzuv icin 4x4 homojen transformasyon matrisleri oluşturulmalıdır. Aşağıdaki tablo 2.1'de robotun DH tablosu görülmektedir.

Tablo 2.1: 6 Eksenli Robot Kolun DH Parametreleri.

i	**a**	**α**	**d**	**θ**
0	350	0	-	-
1	0	$\frac{\pi}{2}$	-815	$0 + \theta_1$
2	850	0	0	$-\frac{\pi}{2} + \theta_2$
3	145	$\frac{\pi}{2}$	0	$0 + \theta_3$
4	0	$-\frac{\pi}{2}$	-820	$\pi + \theta_4$
5	0	$\frac{\pi}{2}$	0	$\pi + \theta_5$
6	-	-	170	$0 + \theta_6$

Belirlenen DH tablosundaki parametrelere göre robotun 4x4 transformasyon matrislerinin belirlenmesi gerekmektedir. Bu matris en genel halde denklem 2.2'deki gibidir.

$$\mathbf{{}_{i}^{i-1}T} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \cos\alpha_{i-1}\sin\theta_i & \cos\alpha_{i-1}\cos\theta_i & -\sin\alpha_{i-1} & -d_i\sin\alpha_{i-1} \\ \sin\alpha_{i-1}\sin\theta_i & \sin\alpha_{i-1}\cos\theta_i & \cos\alpha_{i-1} & d_i\cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.2)$$

4x4 homojen dönüşüm matrislerinin her bir uzuv için DH tablosundan değerlerinin matrise yerleştirilmesi ve ardından altı eksenli robotun ileri kinematik çözümü için bu matrisleri çarpılması gerekmektedir. Bu işlem denklem (2.3) ile hesaplanır.

$$\mathbf{{}_{6}^{0}T} = \mathbf{{}_{1}^{0}\,T_2^1T_3^2T_4^3T_5^4T_6^5T} \quad (2.3)$$

$$\mathbf{{}^{0}_{6}T} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{pmatrix} \quad (2.4)$$

Oluşturulan $\mathbf{{}^{0}_{6}T}$ matrisi robotun uç işlevcisinin, robotun tabanına yerleştirilen referans koordinat sistemine göre konum ve oryantasyonunu belirlemektedir. Dikkat edilmelidir ki bu bilgi robotun eklem değişkenleri olan θ_i (i = 1,2,3,4,5,6) değişmesi ile değişmektedir. $\mathbf{{}^{0}_{6}T}$ matrisinin $\mathbf{T_{ij}}$ (i=1,2,3: j=4) elemanı olan $\mathbf{{}^{0}p_{6}}$ vektörü robotun uç işlevcisinin referans koordinat sistemine göre konum vektörünü ifade etmektedir. $\mathbf{{}^{0}_{6}T}$ matrisinin $\mathbf{T_{ij}}$ (i=1,2,3: j=1,2,3) matrisi $\mathbf{{}^{0}R_{6}}$ ise robotun uç işlevcisinde bulunan koordinat sisteminin referans koordinat sistemine göre euler açılarını yani oryantasyon bilgisini vermektedir. (Craig, John J. 2005)

$$\mathbf{{}^{0}p_{6}} = \begin{bmatrix} r_{14} \\ r_{24} \\ r_{34} \end{bmatrix} \quad (2.5)$$

Konum Vektörü

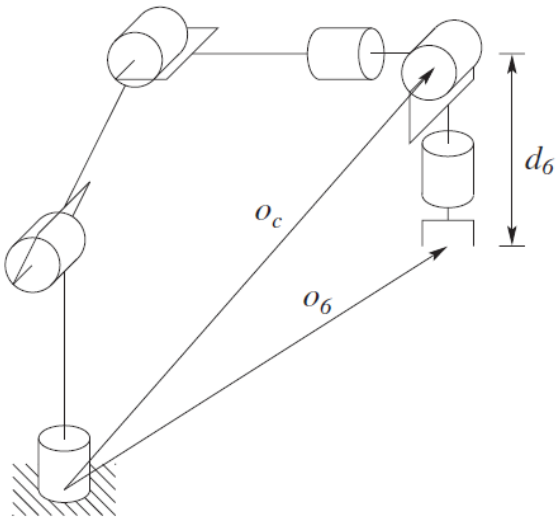
$$\mathbf{{}^{0}R_{6}} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad (2.6)$$

Oryantasyon Matrisi

2.2.2 Robot Ters Kinematiği Geometrik Metod

Robotların ters kinematik problemi, robotun uç işlevcisi için hedeflenen $\mathbf{{}^{0}_{6}T}$ θ_i (i = 1,2,3,4,5,6) gerçekleşmesi için robotun eklemlerindeki açılsal yer değiştirme ne olmalıdır sorusunun cevabını vermektedir. Literatürde robotların ters kinematik problemlerini çözmek icin birçok metod önerilmiştir. Bu tez kapsamında altı eksenli bir robot kolun ters kinematik problemi, kinematik ayrıştırma ile geometrik çözüm metodu ve iki eksenli bir robotun nümerik çözüm metodu incelenecektir. Robotların ters kinematik çözümlerinin kapalı formda analitik olarak çözülmesi her robot konfigürasyon için mümkün olmamakla birlikte bazı durumlarda sonsuz çözüm olması söz konusudur. Robotun kinematik denklemlerinin analitik olarak çözülebildiği ve endüstriye uygun olarak üretilebildiği robot konfigürasyonları için

kinematik ayrıştırma söz konusudur. Endüstriyel altı eksenli robot kol için kinematik ayrıştırma sonucunda elde edilen denklemler ters kinematik problemini, ters konum kinematiği ve ters oryantasyon kinematiği olarak iki ayrı problem olarak incelemektedir. Kinematik ayrıştırmanın mümkün olduğu robot tasarım konfigürasyonu günümüzde endüstriyel robotlarda sıkça kullanılmaktadır. Kinematik ayrıştırma sonucunda robotun uç işlevcisinin koordinat sisteminden robotun bilek koordinat sistemine geriye doğru bir geçiş sağlanır. Böylece robotun bilek koordinat sistemi referans alınarak robotun ilk üç eklemi çözümlür. Bu çözümlemenin ardından robotun son üç eklemi için çözümler yazılmaktadır. Dikkat edilmelidir ki bu yaklaşım küresel bilek tipinde seri robot manipülatörleri için uygulanmaktadır. Şekil 2.6'da görüldüğü gibi robotun bilek merkezi kinematik olarak ayrıştırılmaktadır.



Şekil 2.6: 6 Serbestlik Dereceli Robot Kolun Kinematik Ayrıştırması.

Burada $\mathbf{O_c}$ vektörü robotun uç işlevcisinin konum vektöründen ayrıştırılarak üretilmiştir. Bu ayrıştırma matematiksel olarak denklem (2.7) ve (2.8)'de görüldüğü gibidir.

$$\mathbf{O_c^0} = \mathbf{O}_6 - d_6\mathbf{R}\begin{bmatrix}0\\0\\1\end{bmatrix} \tag{2.7}$$

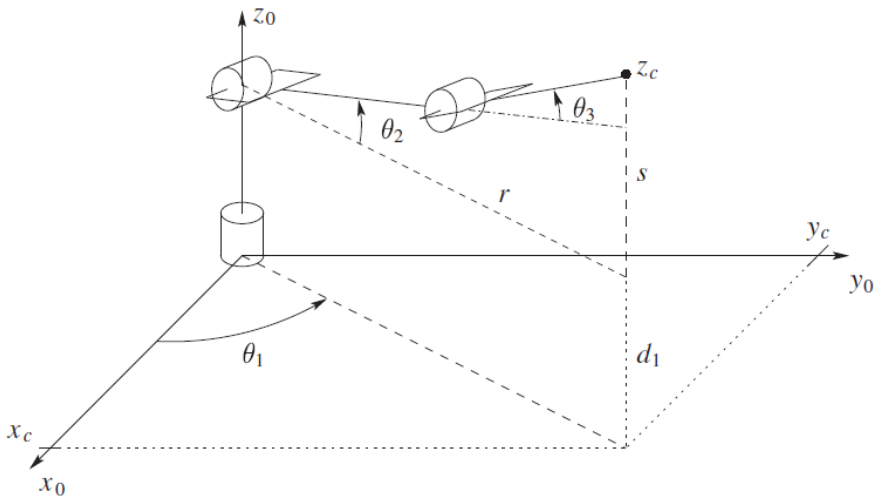
$$\begin{bmatrix}\boldsymbol{x_c}\\\boldsymbol{y_c}\\\boldsymbol{z_c}\end{bmatrix} = \begin{bmatrix}\mathbf{O_x} & - & d_6 r_{13}\\\mathbf{O_y} & - & d_6 r_{23}\\\mathbf{O_z} & - & d_6 r_{33}\end{bmatrix} \tag{2.8}$$

Robotun uç işlevcisinin oryantasyon matrisinin ayrıştırılması matematiksel olarak denklem (2.9) da görüldüğü gibidir.

$$\mathbf{R_6^0} = \mathbf{R_3^0 R_6^3} \tag{2.9}$$

$$\mathbf{R_6^3} = (\mathbf{R_3^0})^{-1}\mathbf{R_6^0} = (\mathbf{R_3^0})^{\mathbf{T}}\mathbf{R_6^0} \tag{2.10}$$

Altı eksenli seri robotun ayrıştırma sonucunda $\mathbf{R_6^3}$ matrisi yardımıyla euler açıları bulunmaktadır. (2.10) denklemine dikkat edilecek olursak kinematik konum ayrıştırması sonucunda $[\theta_1, \theta_2, \theta_3]$ eklem değişkenleri bulunabilir ve buna bağlı olarak $\mathbf{R_3^0}$ matrisi ve tersi hesaplanabilecektir. $\mathbf{R_3^0}$ matrisi ortonormal matris olduğu icin tersi transpozesine eşittir. $\mathbf{R_6^0}$ robot kolun hedeflenen nokta icin uç işlevcinin gerçekleştirmesi gereken oryantasyon bilgisidir ve ters kinematik probleminin çözülmesi icin $\mathbf{T_6^0}$ matrisinden elde edilmektedir. Kinematik ayrıştırma matematiksel olarak gerçekleştirildikten sonra $\mathbf{O_c^0}$ elde edilen matris üzerinden altı eksenli robot fiziksel olarak şekil 2.7'deki gibi olmaktadır.

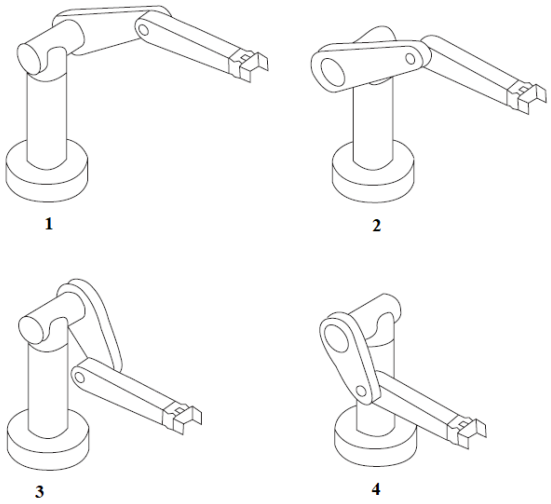


Şekil 2.7: 3DOF Kinematik Ayrıştırma Yapılmış Robot Manipülatör.

Kinematik ayrıştırma yapıldıktan sonra robotun hedeflenen $\mathbf{T_6^0}$ noktasına erişebilmesi için öncelikle $[\theta_1, \theta_2, \theta_3]$ açıları geometrik olarak çözümlenecektir. $[\theta_1, \theta_2, \theta_3]$ Açılarının matematiksel çözümü aşağıdaki gibidir.

$$\theta_1 = A\,tan\,2\,(x_c, y_c)$$
$$\theta_2 = A\,tan\,2\,(\sqrt{x_c^2 + y_c^2 - d^2}, z_c - d_1) - A\,tan\,2\,(a_2 + a_3c_3, a_3s_3)$$
$$\theta_3 = A\,tan\,2\,(D, \pm\sqrt{1 - D^2})$$
$$D = \frac{x_c^2 + y_c^2 - d^2 + (z_c - d_1)^2 - a_2^2 - a_3^2}{2a_2a_3} \quad (2.11)$$

Denklem 2.11'e dikkat edilirse robotun hedeflenen $\mathbf{O_c^0}$ konumuna gelebilmesi için birden fazla çözümün olduğu görülmektedir. Bu çözümler içerisinden seçim yapılması gerekmektedir bu durum şekil 2.8'de görülmektedir.



Şekil 2.8: 3 Eksenli Robot Kol Olası Çözümler.

[$\theta_1, \theta_2, \theta_3$] Açılarının analitik olarak çözümlenmesi ve uygun çözümlerin seçilmesinden sonra $\mathbf{R_3^0}$ matrisi hesaplanabilmektedir. $[\theta_4, \theta_5, \theta_6]$ çözümü için (2.12) denklemi kullanılmaktadır.

$$\mathbf{R_6^3} = (\mathbf{R_3^0})^{-1}\mathbf{R_6^0} = (\mathbf{R_3^0})^{T}\mathbf{R_6^0} \quad (2.12)$$

$$\mathbf{R_3^0} = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 \\ s_1c_{23} & -s_1s_{23} & -c_1 \\ s_{23} & c_{23} & 0 \end{bmatrix} \quad (2.13)$$

$$\mathbf{R_6^3} = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5c_6 + c_4c_6 & s_4s_5 \\ -s_5c_6 & s_5s_6 & c_5 \end{bmatrix} \quad (2.14)$$

Denklem (2.12) eşitliklerini tekrar yazıp taraf tarafa eşitlik yazarsak Denklem (2.15) elde edilir.

$$\mathbf{R_6^3} = (\mathbf{R_3^0})^{\mathbf{T}}\mathbf{R_6^0}$$

$$\begin{aligned} c_4s_5 &= c_1c_{23}r_{13} + s_1c_{23}r_{23} + s_{23}r_{33} \\ s_4s_5 &= -c_1s_{23}r_{13} - s_1s_{23}r_{23} + c_{23}r_{33} \\ c_5 &= s_1r_{13} - c_1r_{23} \end{aligned} \quad (2.15)$$

Denklem (2.15) çözümlenirse, $[\theta_4, \theta_5, \theta_6]$ analitik olarak (2.16) ile elde edilir.

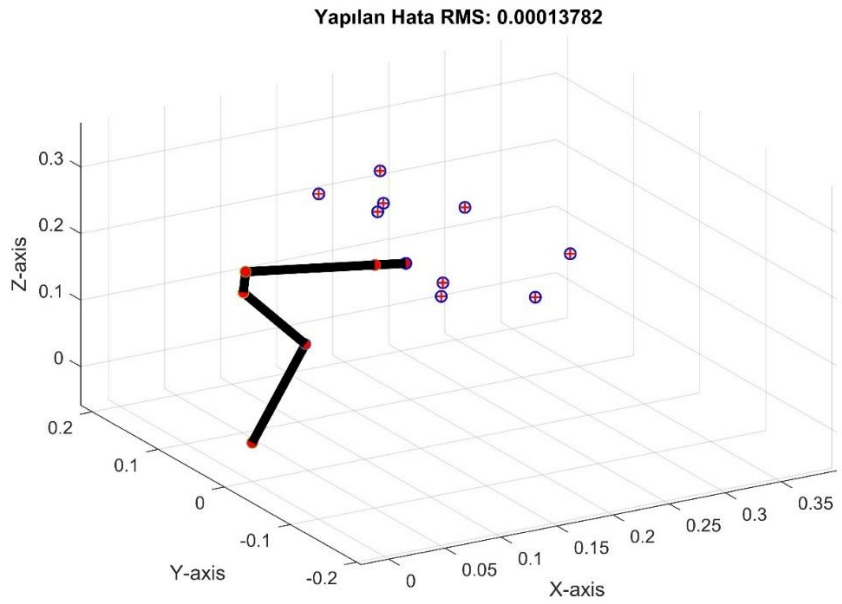
$$\begin{aligned} \theta_5 &= A\,tan\,2\,(s_1r_{13} - c_1r_{23}, \pm\sqrt{1 - (s_1r_{13} - c_1r_{23})^2}) \\ \theta_4 &= A\,tan\,2\,(c_1c_{23}r_{13} + s_1c_{23}r_{23} + s_{23}r_{33}, -c_1s_{23}r_{13} - s_1s_{23}r_{23} + c_{23}r_{33}) \\ \theta_6 &= A\,tan\,2\,(-s_1r_{11} + c_1r_{21}, s_1r_{12} - c_1r_{22}) \end{aligned} \quad (2.16)$$

Kinematik ayrıştıma yaparak altı eksenli seri bir manipülatörün analitik çözümleri elde edilmiş olur. Robot kolun uç işlevcisinin çalışma uzayında belirli bir konum ve oryantasyon bilgisini tutan $\mathbf{T_6^0}$ verildiği zaman analitik olarak $[\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$ hesaplanabilmektedir. Dikkat edilmelidir ki çözüm içerisinde bulunan denklemlerdeki işaretler birden fazla çözüm oluşturmaktadır. Altı serbestlik dereceli bir robot kol icin genel olarak 2^6çözüm oluşmaktadır. Fakat bu çözümlerin yarısı sanal yani fiziksel olarak gerçekleştirilmesi pek mümkün olmamaktadır. Kalan çözümler içerisinden robotun yapacağı işe göre çözümün seçilmesi gerekmektedir. Tablo 2.2'de altı eksenli robot kolun ters kinematik çözüm algoritması görülmektedir.

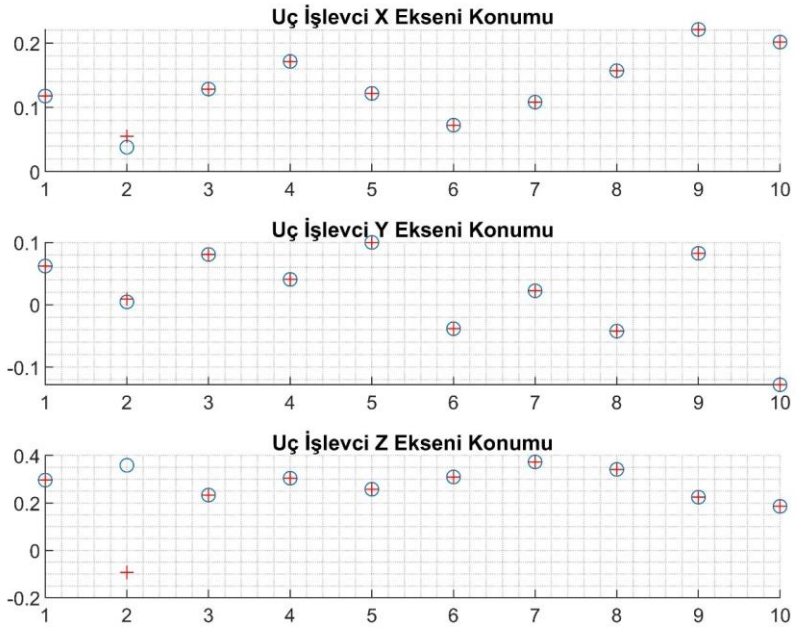
Tablo 2.2: 6 Eksenli Robot Kol Ters Kinematik Çözüm Algoritması.

ALGORİTMA : 6 Eksenli Robot Kol Ters Kinematik Çözümü		
1	**:**	$\theta_i(i = 1,2,3,4,5,6)$ *Her bir eksen için random açı değerleri üret (rad)*
2	**:**	*Robotun D-H parametrelerini gir*
3	**:**	*Her bir uzuv için* ${}_{i}^{i-1}\boldsymbol{T}$ 4x4 *homojen tansformasyon matrisi oluştur*
4	**:**	${}_{6}^{0}\boldsymbol{T} = {}_{1}^{0}\boldsymbol{T}\,{}_{2}^{1}\boldsymbol{T}\,{}_{3}^{2}\boldsymbol{T}\,{}_{4}^{3}\boldsymbol{T}\,{}_{5}^{4}\boldsymbol{T}\,{}_{6}^{5}\boldsymbol{T}$ *matrisini hesapla*
5	**:**	${}_{6}^{0}T$ *matrisini kullanarak tersine çözüm yaparak* $\theta_i(i = 1,2,3,4,5,6)$
6	**:**	$\theta_i(i = 1,2,3,4,5,6)$ *değerleri için çözüm seçimi yap*
7	**:**	*Hatayı hesapla*

Tablo 2.2'deki algoritmanın rastgele üretilen $\theta_i(i = 1,2,3,4,5,6)$ on açı değeri için matlab grafik sonuçları 2.9'da görüldüğü gibidir. Şekil 2.10'da robot kolun uç işlevcisinin çalışma uzayındaki X,Y,Z koordinatları ileri ve ters kinematik çözümleri görülmektedir. Rastgele üretilen noktaların sonucunda robotun ters kinematik çözümlerinin hatası olmadığı gözlemlenmiştir. Robotun ters kinematik denklemleri analitik olarak çözümlendiği için hata değeri yoktur. Bir sonraki bölümde robotun ters kinematik denklemleri sayısal olarak çözümlenecektir ve robotun ters kinematik çözümü yaklaşık olarak hesaplanacaktır. (Spong, Mark W. ve diğ. 2020)



Şekil 2.9: 6 Eksenli Robot Kol Matlab Simulasyonu.



Şekil 2.10: 6 Eksenli Robot Kolun Çalışma Uzayındaki X-Y-Z Konumları.

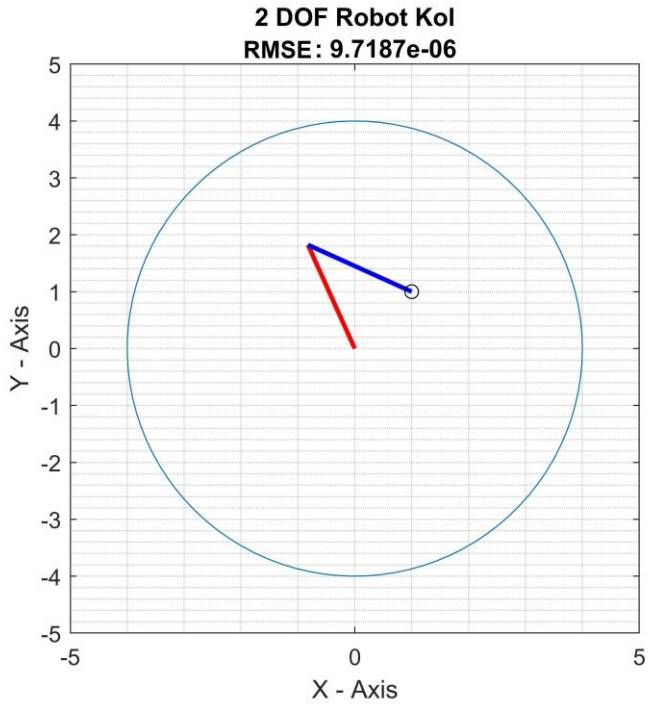
2.2.3 Robot Ters Kinematiği Nümerik Metod

Endüstriyel robotların kinematik problemlerinin çözümlenmesi için analitik metodlar bir önceki bölümde incelemiş ve problemlere kapalı formda analitik çözümler yazılmıştır. Nümerik çözümleme, kapalı formda analitik çözümlerden farklı olarak iteratif olarak sayısal çözümleme teknikleriyle robotların ters kinematik problemini çözmektedir. Bu bölümde iki eksenli bir robot kolun ters kinematik problemini bir optimizasyon problem olarak ele alacağız ve gradyan azalan algoritması ile iteratif olarak çözeceğiz. Yüksek serbestlik dereceli artık (redundant) robotların veya paralel robotların çoğunun kapalı formda analitik çözümlerini elde etmek imkansızdır. Sayısal optimizasyon metodları ile çözüme belirli bir hata toleransında yakınsamak mümkündür. Fakat sayısal optimizasyon metodları çözümü garanti etmemektedir. Kullanılan optimizasyon metodu başlangıç parametrelerine, adım aralığına ve robotun tekil noktalarına çok duyarlıdır. Tablo2.3'de gradyan azalan algoritması görülmektedir.

Tablo 2.3: 2DOF Robot Gradyan Azalan İle Ters Kinematik Çözümü.

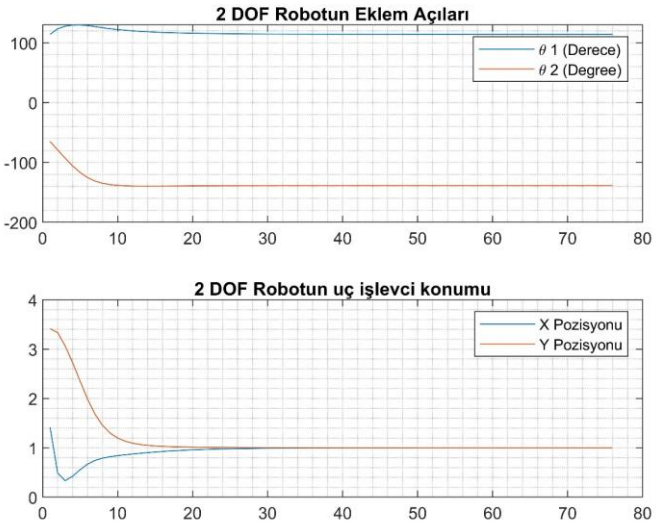
ALGORİTMA : 2 Eksenli Robot Kol Gradyan Azalan Algoritmasıyla Ters Kinematik			
1	**:**	$\theta_i(i = 1{,}2)$	*Başlangıç parametrelerini belirle*
2	**:**	$[x_d, y_d]$	*Robotun ulaşması gereken hedef noktayı belirle*
3	**:**	*Sonsuz Döngü*	
4	**:**	$[x, y] = f(\theta_1, \theta_2)$	*Robotun ileri kinematik çözümünü hesapla*
5	**:**	$[e_1, e_2] = [x_d, y_d] - [x, y]$	*Hata vektörünü hesapla*
6	**:**	$\boldsymbol{J(\theta_1, \theta_2)}$	*Sistemin Jacobian matrisini hesapla*
7	**:**	$\boldsymbol{p} = \sum_{i=1}^{2} \boldsymbol{J}$	*Aday adım aralığı*
8	**:**	s =AltınOran $(\boldsymbol{x}, \boldsymbol{p})$	*Altın oran ile adım aralığını hesapla*
9	**:**	$\boldsymbol{x} = \boldsymbol{x} - s * \boldsymbol{J} * \boldsymbol{e}$	*Parametreleri her bir iterasyonda güncelle*
10	**:**	$if \parallel \boldsymbol{e} \parallel \leq 1e-6$	*Algoritmanın durma koşulu*
11	**:**	*Algoritmayı Durdur*	

Sayısal optimizasyon ile ters kinematik denklemlerinin matlab sonuçları şekil 2.11'de görülmektedir. Sayısal optimizasyon ile çözülen ters kinematik problem başlangıç koşullarına, algoritmanın adım aralığına ve robotun tekil noktalarına bağlıdır. Robotun tekil noktaları $|\boldsymbol{J(\theta_1, \theta_2)}| = \boldsymbol{0}$ olduğu yerlerde görülmektedir. Bu determinantın sıfır olduğu yerlerde jakobiyen matrisi rank kaybetmektedir ve çözüm oluşmamaktadır. Şekil 2.11'de robotun başlangıç açıları $[\theta_1, \theta_2] = [90^0, -45^0]$ ve hedef nokta $[x_d, y_d] = [1{,}1]$ için algoritma 76 iterasyonda sonucu 9.7187e-06 hata ile hesaplamıştır.



Şekil 2.11: 2DOF Robot Gradyan Azalan İle Ters Kinematik Çözümü.

Şekil 2.12'de robotun eklem açılarının ve robotun uç işlevcisinin değişim grafiği görülmektedir. $[\theta_1, \theta_2] = [90^0, -45^0]$ başlangıç eklem açılarından, $[\theta_1, \theta_2] = [114.3^0, -138.6^0]$ eklem açı değerlerine çözümü iteratif olarak hesaplamaktadır.(Siciliano, Bruno. ve diğ. 2009)



Şekil 2.12: 2 DOF Robot Optimizasyonla Ters Kinematik Çözümü.

2.3 Robot Yörünge Planlaması

Robotik teknolojisinde yörünge planlaması genel olarak iki alt başlıkta incelenmektedir. Birinci yaklaşım robotun çalışma uzayı içerisinde yörünge planlamasıdır. Robot çalışmasını gerçekleştirirken dinamik veya statik bir planlama yapabilir. Dinamik planlamada, robot çalışmasını gerçekleştirirken bulunduğu çalışma uzayı içerisindeki engellere göre yeniden planlama yapabilir. Robotun çalışma uzayındaki engeller hareketli ve zamanla konumları değişiyorsa robot için dinamik bir yörünge planlaması yapması gerekmektedir. İkinci olarak robotların eklem uzayında yörünge planlaması yapılmaktadır. Eklem uzayında yörünge planlaması, robotun çalışma uzayında uç işlevcisinin bir noktadan başka bir noktaya hareketini gerçekleştirmesi için her bir eklemin izlemesi gereken konum, hız, ivme ve jerk değerlerini belirlenmesi gerekmektedir. Bu yörünge planlama yaklaşımı endüstriyel tip robotlarda çok kullanılmaktadır. Robotun uç işlevcisi noktadan noktaya (point to point) yaklaşımla bir yörünge planlaması yaparak robot, bu hareketi gerçekleştirirken her bir eklemin konum, hız, ivme ve jerk değerleri hesaplanmalıdır. Oluşturulan bu yörüngeler bir sonraki alt bölümde robot kontrol konusunda tekrar incelenecek ve robotlar icin tasarlanan kontrolcülere referans giriş işareti olarak kullanılacaktır. Noktadan noktaya yörünge planlamasının bir diğer avantajı işe robotun hareketinin başlangıcından bitişine kadar hız, ivme ve jerk bileşenlerinin kontrol altında olmasıdır.

Bu alt bölümde iki tip yörünge planlaması üzerinde durulacaktır. Birinci tip kübik polinomlarla yörünge planlaması, ikinci tip yamuk polinomlarla yörünge planlamasıdır. (Craig, John J. 2005)

2.3.1 Eklem Uzayında Kübik Polinom Metodu

Craig, John J. (2005) Kübik polinom yaklaşımıyla robotun eklem uzayında yörünge planlamasının yapılması için robotun her bir ekleminin takip etmesi gereken yörünge denklemi bulunmalıdır. Yörünge denklemi bazı kısıtları sağlamalıdır. Denklem (2.17)'de görüldüğü gibi yörünge profili üçüncü dereceden bir polinom olarak yazılmıştır.

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{2.17}$$

Denklem (2.17) robotun eklemlerinin takip etmesi gereken açısal profili ifade etmektedir ve denklem (2.18)'deki kısıtları sağlamalıdır.

$$\begin{aligned} \theta(0) &= \theta_0 \\ \theta(tf) &= \theta_{tf} \end{aligned} \tag{2.18}$$

Denklem (2.18)'e dikkat edilecek olursa (θ_0, θ_{tf}) robotun uç işlevcisinin noktadan noktaya hareketini gerçekleştirirken, eklem değişkenlerinin nasıl değişeceğini belirlemektedir ve bunlar sınır koşullarıdır. (θ_0, θ_{tf}) değerleri robotun ters kinematik denkleminin çözümünden elde edilmektedir. Denklem (2.17)'nin sağlaması gereken diğer kısıtlamalar robotun hız ve ivme profilleri için geçerlidir ve denklem (2.19)'daki eşitliklerle ifade edilmektedir.

$$\begin{aligned} \dot{\theta}(0) &= 0 \\ \dot{\theta}(t_f) &= 0 \end{aligned} \tag{2.19}$$

Denklem (2.19)'daki kısıt robotun her bir ekleminin başlangıç ve bitiş noktalarındaki hızların sıfır olması gerektiğini ifade etmektedir. Denklem (2.18) ve (2.19) kısıtlarına bağlı kalarak denklem (2.17)'deki polinomun katsayılarının bulunması

gerekmektedir. Denklem (2.17) tekrardan düzenlenerek denklem (2.20)'deki gibi yazılıp eşitlikler oluşturulur.

$$\begin{aligned}\theta(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ \dot{\theta}(t) &= a_1 + 2a_2 t + 3a_3 t^2 \\ \ddot{\theta}(t) &= 2a_2 + 6a_3 t\end{aligned} \tag{2.20}$$

Denklem (2.20) ve (2.18), (2.19) kısıtları beraber çözümlenerek denklem (2.17)'nin katsayıları bulunabilmektedir.

$$\begin{aligned}\theta_0 &= a_0 \\ \theta_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\ 0 &= a_1 \\ 0 &= a_1 + 2a_2 t_f + 3a_3 t_f^2\end{aligned} \tag{2.21}$$

Eşitlik (2.21)'den yararlanarak katsayıların çözümü (2.22) ile hesaplanır.

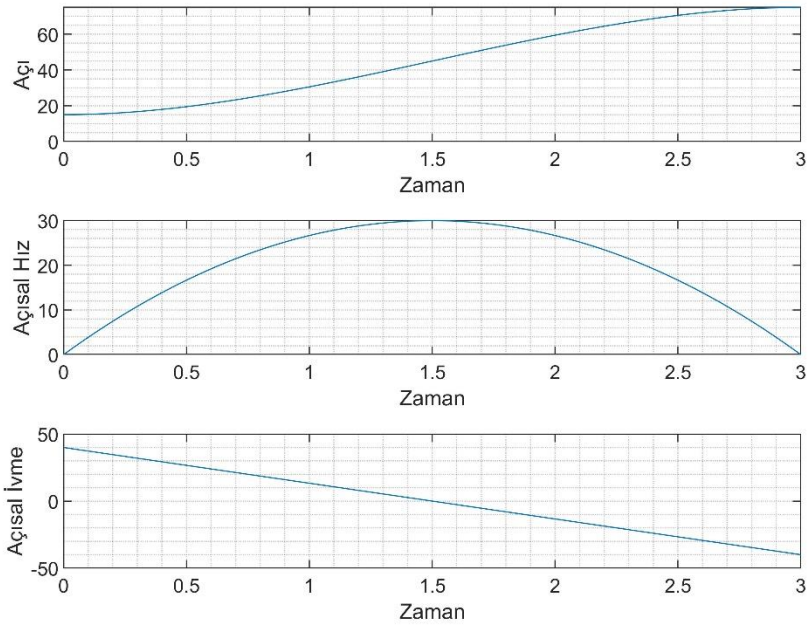
$$\begin{aligned}a_0 &= \theta_0 \\ a_1 &= 0 \\ a_2 &= \tfrac{3}{t_f^2}(\theta_f - \theta_0) \\ a_3 &= -\tfrac{2}{t_f^3}(\theta_f - \theta_0)\end{aligned} \tag{2.22}$$

Kübik polinom yörünge planlayıcısının tek eksenli bir robot kol için başlangıç konumu $\theta(0) = 15^o$ 'den $\theta(f) = 75^o$ 'ye üç saniye içerisinde ulaşması hedeflenmektedir. Robotun $\dot{\theta}(0) = \dot{\theta}(f) = 0$ olması için kübik polinom katsayıları (2.22) eşitliğinden hesaplanmaktadır. Eşitlik (2.23)'de katsayıların hesabı ve denklemler görülmektedir.

$$a_0 = 15.0$$
$$a_1 = 0.0$$
$$a_2 = 20.0$$
$$a_3 = -4.44$$

$$\theta(t) = 15 + 20t^2 - 4.44t^3$$
$$\dot{\theta}(t) = 40t - 13.33t^2$$
$$\ddot{\theta}(t) = 40 - 26.66t \quad (2.23)$$

Denklem (2.23)'de hesaplanan katsayıların matlab sonucları şekil 2.13'de görülmektedir.



Şekil 2.13: Eklem Uzayında Kübik Polinom Yörünge Planlayıcısı.

2.3.2 Eklem Uzayında Yamuk Polinom Metodu

Lewis, Frank L. ve diğ. (2006) Yamuk polinom yaklaşımıyla yörünge planlaması endüstride çok kullanılmaktadır. Bunun sebebi endüstriyel aktüatörlerin hareketini gerçekleştirirken hız ve ivme profillerini özel olarak ayarlamaktır. Aktüatör hareketin büyük bir kısmında sabit hız ile hareket etmektedir. Böylece sıfır ivme ile hareketi gerçekleştirmektedir bu durum fiziksel sistem üzerine etkiyecek sanal kuvvetleri ve titreşimi bir miktar yok etmektedir. Bu durum basitçe denklem (2.24)'de görülmektedir. Denklemin sağ tarafındaki açısal ivme ve çizgisel ivme bileşenlerinin sıfır olması sanal kuvvetleri yok etmektedir.

$$\sum F = m\vec{a}$$
$$\sum M = I\vec{\alpha} \tag{2.24}$$

Yamuk profili ile hareketin gerçekleşmesi için hareketi üç ana parçaya bölüyoruz. Birinci parça sistemin hızlandığı, ikinci parca sabit hız ile hareket ettiği ve son olarak üçüncü parça sistemin yavaşlamasını ifade etmektedir. Böylece yumuşak bir geçiş ile aktüatör hareket ettirilmiş olacaktır. Sistem ilk anda hızlanırken sabit bir hız profili ile hızlanmaktadır bu sayede sistemin kütlesi ve ataleti, ayrıca yükün kütlesi ve ataletinden bağımsız olarak kademeli bir şekilde hızlanacaktır. Bu davranış robotik sistemlerde salınım hareketini azaltmaktadır. Yörünge için öncelikle t_0, t_b zaman aralığında kuadratik bir konum profili oluşturur, böylece sabit hızlanma profili elde edilir. Sistem t_b zamanına geldiğinde konum profilinin değişmektedir böylece konum profili artık lineer olmakta ve hız profili sabit olmaktadır. Son olarak $t_f - t_b$ zamanında sistem azalan kuadratik bir konum profili ve azalan sabit hız profili izlemektedir. Yörüngenin öncelikle t_0, t_b zaman aralığı için denklemleri (2.25)'deki gibidir.

$$t_0 = 0$$
$$\dot{\theta}(t_0) = \dot{\theta}(t_f) = 0 \tag{2.25}$$

t_0, t_b zaman aralığı için konum ve hız denklemi (2.26)'daki gibidir.

$$\theta(t) = a_0 + a_1 t + a_2 t^2$$
$$\dot{\theta}(t) = a_1 + 2a_2 t \tag{2.26}$$

Denklem (2.25)'deki kısıtlar denklem (2.26)'da yerine yazılarak katsayılar denklem (2.27)'deki gibi belirlenir.

$$\begin{aligned} a_0 &= \theta(t_0) \\ a_1 &= 0 \end{aligned} \tag{2.27}$$

t_b zamanında hız profilimizin verilen bir sabit V değerine eşit olmasını isteriz böylece denklem (2.28) elde edilir.

$$\dot{\theta}(t_b) = 2a_2 t_b = V \tag{2.28}$$

Böylece denklem (2.28)'den $a_2$katsayısı denklem (2.29)'daki gibi hesaplanır.

$$a_2 = \frac{V}{2t_b} \tag{2.29}$$

Denklem (2.25) ve (2.29) t_0, t_b zaman aralığı için daha kompakt bir halde denklem (2.30)'da verildiği gibidir.

$$\begin{aligned} \theta(t) &= \theta(t_0) + \frac{V}{2t_b}t^2 = \theta(t_0) + \frac{\alpha}{2}t^2 \\ \dot{\theta}(t) &= \frac{V}{t_b}t = \alpha t \\ \ddot{\theta} &= \frac{V}{t_b} = \alpha \end{aligned} \tag{2.30}$$

Denklem (2.30)'da α ivme terimini ifade etmektedir. Yörüngemiz t_b ve $t_f - t_b$ zaman aralıklarında lineer bir V hızına sahiptir . Bu yörünge denklem (2.31)'de görüldüğü gibidir.

$$\theta(t) = \theta(t_b) + V(t - t_b) \tag{2.31}$$

Yörüngemiz $\frac{t_f}{2}$ zaman aralığında simetri olması gerektiği için (2.32) eşitliği yazılabilir.

$$\theta(\frac{t_f}{2}) = \frac{\theta(t_0)-\theta(t_f)}{2} \tag{2.32}$$

Eşitlik (2.31) ve (2.32) türetmeye devam edersek. Denklem (2.33) elde edilir.

$$\frac{\theta(t_0)-\theta(t_f)}{2} = \theta(t_b) + V(\frac{t_f}{2} - t_b)$$
$$\theta(t_b) = \frac{\theta(t_0)-\theta(t_f)}{2} - V(\frac{t_f}{2} - t_b) \tag{2.33}$$

Denklem (2.33) ve (2.30) t_b noktasında biribirini sağlaması gerektiği için denklem (2.34) elde edilir.

$$\theta(t_0) + \frac{V}{2}t_b = \frac{\theta(t_0)+\theta(t_f)-Vt_f}{2} + Vt_b$$
$$t_b = \frac{\theta(t_0)-\theta(t_f)+Vt_f}{V} \tag{2.34}$$

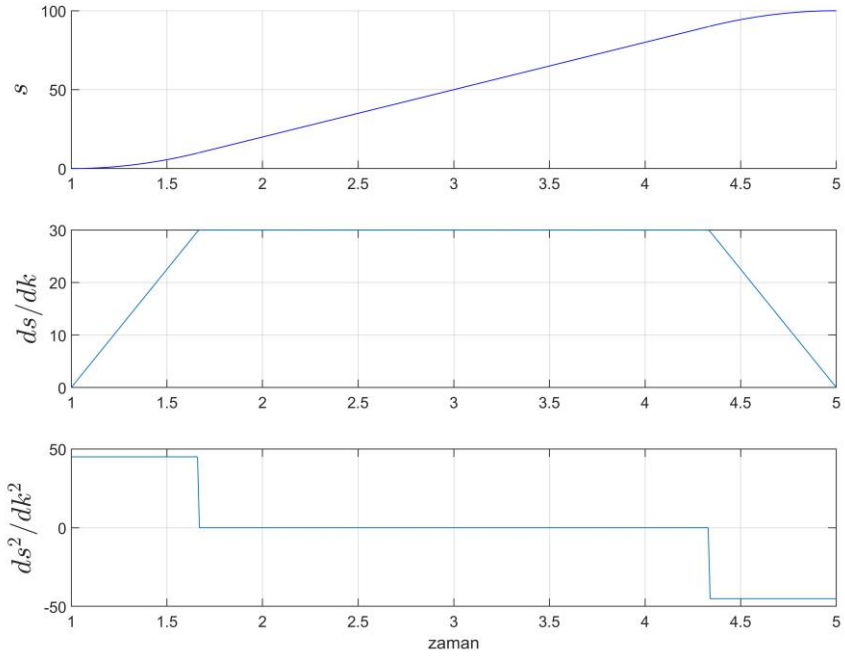
t_b zaman aralığının $0 < t_b \leq \frac{t_f}{2}$ aralığında olması gerektiği için (2.35)'deki eşitlik sağlanmalıdır.

$$\frac{\theta(t_f)-\theta(t_0)}{V} < t_f \leq \frac{2(\theta(t_f)-\theta(t_0))}{V}$$
$$\frac{\theta(t_f)-\theta(t_0)}{t_f} < V \leq \frac{2(\theta(t_f)-\theta(t_0))}{t_f} \tag{2.35}$$

Eşitlik (2.35) düşünüldüğünde V belirtilen hızın bu aralıkta olması gerekmektedir. Yörüngemizin üçüncü parçası olan $t_f - t_b$ ile t_f zaman aralığı simetri şartını sağlaması gerekmektedir Böylece $t_0 - t_f$ zaman aralığı için yörünge denklemimiz (2.36)'daki gibidir.

$$\theta(t)\begin{cases} \theta(t_0)+\frac{\alpha}{2}t^2 & 0<t\le t_b \\ \frac{\theta(t_f)-\theta(t_0)-Vt_f}{2}+Vt & t_b<t\le t_f-t_b \\ \theta(t_f)-\frac{\alpha t_f^2}{2}+\alpha t_f t-\frac{\alpha}{2}t^2 & t_f-t_b<t\le t_f \end{cases} \quad (2.36)$$

Denklem (2.36)'nın matlab simulasyonu şekil 2.14'de görüldüğü gibidir. Simulasyon parametreleri $\theta(0)=0^o,\ \theta(f)=100^o,\ t=5sn,\ V=30m/sn$ olarak ayarlanmıştır. (Siciliano, Bruno. ve diğ. 2009)



Şekil 2.14: Eklem Uzayında Yamuk Yörünge Planlayıcısı.

2.4 Robot Dinamiği

Robotların kinematiğini incelediğimiz bir önceki bölümde robotun hareketine sebep olan iç, dış kuvvet ve momentlerle ilgilenmemiştir. Robotların sadece hareketi sonucunda her bir eklemin ve robotun uç işlevcisinin konum, hız, ivme ve jerk değerlerini inceledik. İlgilendiğimiz bu değişkenler mekanik bilimi için uç değişkenler olarak ifade edilmektedir ve bir referansa göre ölçümlerinin yapılması gerekmektedir. Robot dinamiği bölümünde harekete sebep olan kuvvet ve momentlerle ilgileneceğiz. Bu bölümde ifade edilen kuvvet ve momentler iç değişkenlerdir.

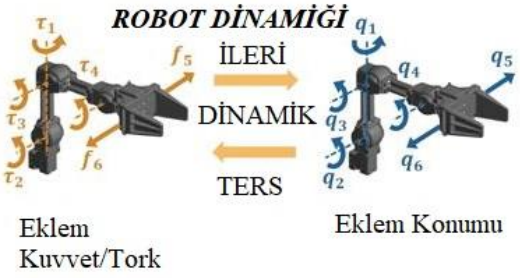
Dinamik bilimi mekaniğin önemli bir alanı olarak kendisine birçok uygulama alanı bulmaktadır. Robot teknolojisinin gelişmesi için kinematik ve dinamik alanının birbirine bağlanması gerekmektedir. Bu bağlantı robotik sistemin uç değişkenleri ile iç değişkenlerini birbirine bağlamaktadır. Bu bağlantı matematiksel olarak diferansiyel denklemlerle ifade edilmektedir. Robotik sistemlerde fiziksel sistemin diferansiyel denklemi genel olarak lineer olmayan bir formda ve durum uzayı şekilde ifade edilmektedir. Denklem (2.37)'de genel olarak lineer olmayan zamanla değişen diferansiyel denklem takımının matematiksel gösterimi verilmiştir. (Spong, Mark W. ve diğ. 2020)

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) \tag{2.37}$$

Denklem (2.37)'de $\boldsymbol{x} \in \mathbb{R}^n$ durum vektörü , $\boldsymbol{u} \in \mathbb{R}^m$ giriş vektörü ve $t \in \mathbb{R}$ zaman vektörünü ifade etmektedir. $\boldsymbol{f}(.)$ fonksiyonu t'ye açık bir şekilde bağımlı değilse sistem zamanla değişmeyen biçimdedir. Denklem (2.37)'nin açık yazımı denklem (2.38)'de görüldüğü gibidir.

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_m, t) \\ \dot{x}_2 &= f_2(x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_m, t) \\ &\vdots \\ \dot{x}_n &= f_n(x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_m, t) \end{aligned} \tag{2.38}$$

Fiziksel sistemlerin matematiksel modelini çıkarmak için literatürde önerilen birçok metod bulunmaktadır. Bu bölümde robotların dinamik modellerini oluşturmak için Euler-Lagrange metodu kullanılacaktır. Robot dinamiği robot kinematiğinde olduğu gibi ileri ve ters olarak iki alt başlıkta incelenmektedir. Şekil 2.15'de robotun ileri ve ters dinamik ifadesi görülmektedir.



Şekil 2.15: Robot Kolun İleri Ve Ters Dinamiği.

2.4.1 Euler Lagrange Metodu

Euler-Lagrange metodu ile robot kolun hareket denklemi denklem (2.38)'de önerildiği gibi ifade edilebilmektedir. Euler Lagrange metodunu uygulamanın iki yolu vardır. Sanal iş ve hamilton prensibiyle hareket denklemlerini türetmek mümkündür. Bu bölümde sanal iş prensibiyle robotun dinamik denklemleri türetilecektir. Robotun hareket denklemlerini türetmeden önce Euler Lagrange metodunu tek serbestlik dereceli bir fiziksel sisteme uygulamasını Newton'un yasalarından türetelim. Newton'un ikinci yasasına göre bir parçacığın hareket denklemi (2.39)'da görüldüğü gibidir.

$$m\ddot{y} = f - mg \tag{2.39}$$

Denklem (2.39)'un sol tarafının farklı bir yazım formu denklem (2.40) görülmektedir.

$$m\ddot{y} = \frac{d}{dt}(m\dot{y}) = \frac{d}{dt}\frac{\partial}{\partial \dot{y}}(\frac{1}{2}m\dot{y}^2) = \frac{d}{dt}\frac{\partial K}{\partial \dot{y}} \tag{2.40}$$

Denklem (2.40)'da $K = \frac{1}{2}m\dot{y}^2$ tek serbestlik dereceli sistemin kinetik enerjisini ifade etmektedir. Denklem (2.40)'da kısmi türev kullanmamızın sebebi kinetik enerji denklemi genişletildiğinde birden fazla değişkene bağlı olduğundan kullanılmaktadır. Denklem (2.39)' sağ tarafındaki yerçekimi ifadesinide denklem (2.41)'deki gibi genişletelim.

$$mg = \frac{\partial}{\partial y}(mgy) = \frac{\partial P}{\partial y} \tag{2.41}$$

Denklem (2.41)'de $P = mgy$ tek serbestlik dereceli sistemin potansiyel enerjisini ifade etmektedir. Denklem (2.41) ve (2.40) birleştirerek Lagrange denklemini (2.42)'deki gibi türetebiliriz.

$$L = K - P = \frac{1}{2}m\dot{y}^2 - mgy \tag{2.42}$$

Denklem (2.42)'nin Newton'un ikinci yasasına uygun olması için kısmi türevlerinin alınması ve düzenlenmesi gerekmektedir. Denklem (2.43)'de Newton denklemlerinin kinetik ve potansiyel enerji yaklaşımıyla modellenen lagrange denklemi ifadesi görülmektedir.

$$\frac{\partial L}{\partial \dot{y}} = \frac{\partial K}{\partial \dot{y}} \frac{\partial L}{\partial y} = -\frac{\partial P}{\partial y}$$

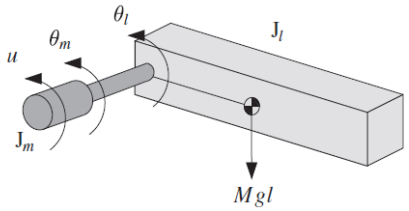
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{y}} - \frac{\partial L}{\partial y} = f \tag{2.43}$$

Denklem (2.43)'de L kinetik ve potansiyel enerji farklarından oluşan lagrange fonksiyonunu belirtmektedir. Denklem (2.43) lagrange ifadesi genel Euler-Lagrange denklemi olarak ifade edilmektedir. Euler-Lagrange metodu ile robotların dinamik denklemlerini oluştururken sistemin serbestlik derecesine bağlı olarak $(q_1, q_2, \ldots, q_n)$ n-DOF sistemin kinetik ve potansiyel enerji ifadeleri yazılmalıdır. Yazılan ifadeler Lagrange yaklaşımıyla gibi denklem (2.44) ile hesaplanır. (Siciliano, Bruno. ve diğ. 2009)

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = \tau_k k = 1,2,\ldots,n \tag{2.44}$$

2.4.1.1 Bir DOF Robot Kol Dinamik Modeli

Denklem (2.44)'de n-DOF bir sistem için dinamik denklemlerin nasıl oluşturulacağı ifade edilmişti. Bu alt bölümde bir serbestlik dereceli robot kol icin dinamik denklemlerini türeteceğiz. Şekil 2.16'da bir serbestlik dereceli robot kol görülmektedir.



Şekil 2.16: Bir Serbestlik Dereceli Robot Kol**.**

Şekil 2.16'da robot kol, dişli sistemiyle dc motora bağlanmıştır. θ_l, θ_m sırasıyla robot kolun açısal hareketi ve motorun açısal hareketini ifade etmektedir. θ_l, θ_m arasındaki ilişki denklem (2.45) görülmektedir.

$$\theta_m = r\theta_l \tag{2.45}$$

Robot kol ile motor arasındaki dişli oranını $r = 1$ ifade etmekteyiz. θ_l, θ_m arasındaki bağlantı sistemin bir serbestlik dereceli olduğunu belirlemektedir. Lagrange denklemini yazmadan önce $q = \theta_l$ genelleştirilmiş koordinat olarak seçelim. Böylece sistemin kinetik enerji denklemleri (2.46)'daki gibidir.

$$K = \frac{1}{2} J_m \dot{\theta}_m^2 + \frac{1}{2} J_l \dot{\theta}_l^2$$
$$K = \tfrac{1}{2}(r^2 J_m + J_l)\dot{q}^2 \tag{2.46}$$

Denklem (2.46)'da J_m, J_l sırasıyla motorun ve robot kolun atalet momentlerini ifade etmektedir. Denklem (2.47)'de sistemin potansiyel enerji denklemi görülmektedir.

$$P = Mgl(1 - cos(q)) \tag{2.47}$$

Denklem (2.47)'de M robot kolun kütlesini ifade etmektedir. l robot kolun kütle merkezinin koordinat sistemine olan uzaklığını ifade etmektedir. $I = r^2 J_m + J_l$ tanımlaması yaparak Lagrange denklemi (2.48)'deki gibi yazalım.

$$L = \tfrac{1}{2} I\dot{q}^2 - Mgl(1 - cos(q)) \tag{2.48}$$

Denklem (2.44)'de n=1 için, (2.48)'deki sisteminin Euler Lagrange denklemi (2.49)'daki gibidir.

$$I\ddot{q} + Mgl\, sin(q) = \tau_l \tag{2.49}$$

Denklem (2.49)'da τ_l robot kolun oluşturduğu tork ifadesini vermektedir. Robot kolun tork ifadesi ile motorun tork ifadesi arasındaki ilişkiyi türetelim. Öncelikle denklem (2.50)'de motorun ürettiği tork ifadesini tanımlayalım.

$$u = r\tau_m \tag{2.50}$$

Denklem (2.50)'deki motorun tork ifadesinin, eklemin tork ifadesine eşit olması gerekmektedir. Motorun tork ifadesi ile robot kol arasındaki tork ifadesi korunumsuz kuvvet olduğu için arasındaki ilişki denklem (2.51)'de görüldüğü gibidir.

$$\tau_l = u - B\dot{q} \tag{2.51}$$

Denklem (2.51)'deki $B = rB_m + B_l$ ifade etmektedir ve sırasıyla B_m, B_l motorun ve robot kolun viskoz sürtünme katsayılarını ifade etmektedir. Denklem (2.51) ve (2.49) birleştirerek bir serbestlik dereceli robot kolun Euler Lagrange denklemi (2.52)'deki gibidir.

$$I\ddot{q} + B\dot{q} + Mgl\,sin(q) = u \tag{2.52}$$

Denklem (2.52) dikkat edilirse, bir serbestlik dereceli robot kol için ikinci dereceden lineer olmayan diferansiyel denklem olduğu görülmektedir. Sistemin M, I kütle ve atalet terimleri zamanla değişmediği için (2.52) zamanla değişmeyen formdadır. Bu denklem bir serbestlik dereceli robot kolun uç değişkeni olan q açısal yerdeğiştirme ile iç değişkeni olan u tork ifadesi arasında bir ilişki kurmaktadır. Denklem (2.52) 'nin çözülmesiyle robotun ileri dinamik modeli çözümlenmiş olur. Denklem (2.52) ifadesi n-DOF sistem için genellenebilir. Dikkat edilirse bu diferansiyel denklem lineer olmayan bir formda olduğu için kapalı formda analitik çözüm bulunmamaktadır. Bir sonraki alt bölümlerde (2.52)'nin durum uzay formunda ifadesi oluşturulup sayısal olarak çözümleri incelecektir. Robotun ileri dinamik denklemi olan (2.52) ifadesi, robotun eklemlerine uygulanan tork ifadesi ile eklemlerin açısal konum ve hızlarının ne olacağını belirlemektedir. Robotun ters dinamik denklemi ise robotun eklem değişkenlerinin açısal konum ve hız olarak robota belirlenmesinden sonra robotun eklemlerinde oluşacak reaksiyon kuvvet ve momentlerinin belirlemesidir. (Spong, Mark W. ve diğ. 2020)

2.4.2 Dinamik Denklemlerin Durum Uzay Modeli

Bu bölümde 2-DOF robot kolun Euler-Lagrange metodu ile dinamik denklemlerinin durum uzay modeli oluşturulacaktır. Denklemler oluşturulmadan önce n-DOF sistem için Euler-Lagrange denkleminin genel tanımlaması yapılacaktır. Denklem (2.44)'den hatırlanacagı üzere fiziksel sistemin kinetik ve potansiyel enerji denklemleri yazılmaktadır. Kinetik enerji ifadesi n-DOF bir sistem için sistemin çizgisel ve açısal hızıyla iki parça olarak hesaplanmaktadır. Bu durum, denklem (2.53)'de görülmektedir.

$$K = \frac{1}{2} m v^T v + \frac{1}{2} \omega^T I \omega \tag{2.53}$$

Denklem (2.53)'de m sistemin kütle matrisini, I_{3x3} simetrik atalet tensörünü, sırasıyla v, ω çizgisel ve açısal hızlarını ifade etmektedir. Denklem (2.53)'ü n-DOF sisteme uygulamadan önce, n-DOF bir sistemin eklem değişkenlerinin açısal ve çizgisel hızlarıyla robotun uç işlevcisinin açısal ve çizgisel hızlarını birbirine bağlayan (2.54) denklemi oluşturulmalıdır.

$$v_i = J_{v_i}(q)\dot{q}\,\omega_i = J_{\omega_i}(q)\dot{q} \tag{2.54}$$

Denklem (2.54)'de J_{v_i},J_{ω_i} (3xn) n-DOF sistemin Jacobian matrislerini ifade etmektedir. Denklem (2.54)'ü denklem (2.53)'ile birleştirelim. Dikkat edilmelidir ki n-DOF sistemin m kütle matrisi homojen ve cismin ağırlık merkezini ortalamaktadır. I atalet tensörü cismin ağırlık merkezinde ve simetrik biçimde olmaktadır, böylece çarpım atalet terimleri $I\delta_{i,j}$ yok olmaktadır. Denklem (2.55)'de n-DOF sistem için kinetik enerji ifadesi görülmektedir.

$$K = \frac{1}{2}\dot{q}^T \left[\sum_{i=1}^{n} \{ m_i J_{v_i}(q)^T J_{v_i}(q) + J_{\omega_i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega_i}(q) \} \right] \dot{q}$$

$$K = \frac{1}{2}\dot{q}^T D(q)\dot{q}$$

$$D(q) = \left[\sum_{i=1}^{n} \{ m_i J_{v_i}(q)^T J_{v_i}(q) + J_{\omega_i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega_i}(q) \} \right] \tag{2.55}$$

Denklem (2.55)'de $D(q)$ ifadesi nxn eklem değişkenleri olan, q'lara bağlı atalet veya kütle matrisi olarak ifade edilmektedir. Bazı kaynaklarda $D(q) = M(q)$ olarak ifade edilmektedir. Bu tez çalışmasında $D(q)$ olarak ifade edilecektir. $D(q)$ atalet

matrisi pozitif tanımlı ve simetrik bir matristir. Matrisin pozitif tanımlı olması hiçbir zaman negatif olmayacağı anlamındadır. Eklem hareketleri sıfır olduğu zaman sistemin kinetik enerjisi de sıfır olmaktadır. Denklem (2.44)'ün potansiyel enerji terimini n-DOF bir sistem için, denklem (2.56) ve (2.57)'deki gibidir.

$$P_i = m_i g^T r_{ci} \tag{2.56}$$

Denklem (2.56)'da g terimi yerçekimi vektörünü, r_{ci} terimi i. uzvun ağırlık merkezine olan uzaklığı ifade etmektedir. n-DOF bir sistem için genel potansiyel enerji denklemi (2.57)'deki gibidir.

$$P = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} m_i g^T r_{ci} \tag{2.57}$$

Denklem (2.57) ve (2.55) ile n-DOF robot için Euler-Lagrange denklemini tanımlamak için kinetik ve potansiyel enerji denklemlerinini, eklem değişkenlerine göre kısmi türevlerinin alınması ve kinetik enerji teriminin tekrar zamana göre türevlenmesi gerekmektedir. Lagrange fonksiyonunun eklem değişkenlerine göre türevlenmiş ifadesi denklem (2.58)'de görüldüğü gibidir.

$$K = \frac{1}{2}\dot{q}^T D(q)\dot{q} = \frac{1}{2}\sum_{i,j} d_{i,j}(q)\dot{q}_i\dot{q}_j \tag{2.58}$$

Denklem (2.58) ile kinetik enerji terimi, kuadratik gösterimi olarak ifade edilmektedir. Potansiyel enerji terimi $P = P(q)$ eklem değişkeninin hızından $\dot{q}$ bağımsızdır. Euler-Lagrange denklemi (2.58) ile tekrar yazılırsa (2.59)'da görüldüğü gibidir.

$$L = K - P = \frac{1}{2}\sum_{i,j} d_{i,j}(q)\dot{q}_i\dot{q}_j - P(q) \tag{2.59}$$

Denklem (2.59)'daki Lagrange fonksiyonunun eklem değişkenlerinin hızlarına göre kısmi türevli ifadesi denklem (2.60)'daki gibidir.

$$\frac{\partial L}{\partial \dot{q}_k} = \sum_j d_{kj}\dot{q}_j \tag{2.60}$$

Denklem (2.60)'n zamana göre türevlenmesi denklem (2.61)'de görüldüğü gibidir.

$$\begin{aligned}\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} &= \sum_j d_{kj}\ddot{q}_j + \sum_j \frac{d}{dt} d_{kj}\dot{q}_j \\ &= \sum_j d_{kj}\ddot{q}_j + \sum_j \frac{\partial d_{kj}}{\partial q_i}\dot{q}_i\dot{q}_j \end{aligned} \tag{2.61}$$

Benzer şekilde denklem (2.59)'un eklem değişkenlerine göre kısmi türevi denklem (2.62)'deki gibidir.

$$\frac{\partial L}{\partial q_k} = \frac{1}{2}\sum_{i,j}\frac{\partial d_{ij}}{\partial q_k}\dot{q}_i\dot{q}_j - \frac{\partial P}{\partial q_k} \tag{2.62}$$

Böylece $k = 1, \ldots, n$ için Euler-Lagrange denklemi (2.63)'deki gibi oluşturulur.

$$\sum_j d_{kj}\ddot{q}_j + \sum_{i,j}\left\{\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i\dot{q}_j + \frac{\partial P}{\partial q_k} = \tau_k \tag{2.63}$$

Denklem (2.63)'de toplama sırasını değiştirerek ve simetri özelliğinden faydalanarak (2.63) daha kompakt bir hale getirilebilir.

$$\sum_{i,j}\left\{\frac{\partial d_{kj}}{\partial q_i}\right\}\dot{q}_i\dot{q}_j = \frac{1}{2}\sum_{i,j}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j}\right\}\dot{q}_i\dot{q}_j \tag{2.64}$$

Denklem (2.64) denklem (2.63)'de yerine koyabiliriz.

$$\begin{aligned}\sum_{i,j}\left\{\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i\dot{q}_j &= \sum_{i,j}\frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i\dot{q}_j \\ &= \sum_{i,j} c_{ijk}\,\dot{q}_i\dot{q}_j \end{aligned} \tag{2.65}$$

Denklem (2.65)'deki c_{ijk}terimi denklem (2.66)'da görüldüğü gibidir.

$$c_{ijk} = \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\} \tag{2.66}$$

Son olarak denklem (2.67) tanımlamasını yapalım.

$$g_k = \frac{\partial P}{\partial q_k} \tag{2.67}$$

Böylece (2.63)'deki Euler-Lagrange denklemi en sade biçimiyle denklem (2.68)'de verildiği gibidir.

$$\sum_j d_{kj}\ddot{q}_j + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ijk}(q)\, \dot{q}_i \dot{q}_j + g_k(q) = \tau_k \qquad k = 1, \ldots, n \tag{2.68}$$

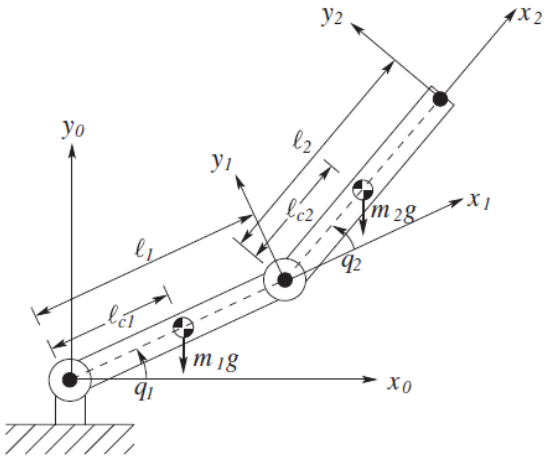
Denklem (2.68)'de üç tip terim bulunmaktadır. Birinci tip, eklem değişkenlerinin ikinci türevini içermektedir. İkinci tip terim eklem değişkenlerinini birinci türevini içermektedir. Burada $\dot{q}_i\dot{q}_j$, $i \neq j$ için $\dot{q}_i^2$ terimini oluşturmaktadır. Bu terim merkezkaç kuvvetini oluşturmaktadır. Denklemdeki $\dot{q}_i\dot{q}_j$ terim coriolis kuvvetini oluşturmaktadır. Denklemdeki üçüncü terim eklem değişkenlerine bağlıdır. Denklem (2.68)'in kompakt bir biçimde gösterimi denklem (2.69)'da gibi ifade edilir.

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \tag{2.69}$$

Denklem (2.69) n-DOF bir sistemin ileri dinamik denklemini oluşturmaktadır ve zamanla değişen lineer olmayan diferansiyel denklem olarak karşımıza çıkmatadır. Denklem (2.69)'un çoğu durumda kapalı formda analitik bir çözümü bulunmamaktadır. Dinamik denklemler çoğunlukla sayısal olarak çözülmektedir. (Craig, John J. 2005)

2.4.2.1 İki ve Üç DOF Robot Kol Matematiksel Modeli

Bu tez çalışmasında iki DOF robot kolun model tabanlı kontrolü ve üç DOF robot kolun pekiştirmeli öğrenme tabanlı tork kontrolü yapılmıştır. Geliştirilecek algoritmaların bilgisayar yazılımı yardımıyla simulasyonunun yapılması için 2 DOF ve üç DOF robot kolun dinamik ve kinematik denklemlerinin oluşturulup, sayısal olarak çözülmesi gerekmektedir. Bu bölümde iki DOF ve üç DOF robot kolun dinamik denklemleri çıkarılmıştır. Şekil 2.17'de iki DOF düzlemsel robot kolun modeli görülmektedir.



Şekil 2.17: 2 Dof Robot Kol Dinamik Modeli.

Şekil 2.17'de $i = 1,2$ için, q_i eklem açılarını, m_i uzuvların kütlesini, l_i uzuv uzunluklarını, l_{ci} ilgili uzvun kütle merkezini, I_i ilgili uzvun atalet tensörünü, g yerçekimi terimini ifade etmektedir. İki DOF robot kol dinamik denklemleri için öncelikle kinetik enerji terimini oluşturalım.

$$v_{c1} = J_{v_{c1}}\dot{q} \tag{2.70}$$

Denklem (2.70)'deki Jacobian matrisi denklem (2.71) ile ifade edilmektedir.

$$J_{v_{c1}} = \begin{bmatrix} -l_{c1}\, sin\, q_1 & 0 \\ l_{c1}\, cos\, q_1 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.71}$$

Benzer şekilde.

$$v_{c2} = J_{v_{c2}}\dot{q} \tag{2.72}$$

Denklem (2.72)'nin Jacobian matrisi denklem (2.73)'ile ifade edilmektedir.

$$J_{v_{c2}} = \begin{bmatrix} -l_1 \sin q_1 - l_{c2} \sin(q_1 + q_2) & -l_{c2} \sin(q_1 + q_2) \\ l_1 \cos q_1 + l_{c2} \cos(q_1 + q_2) & l_{c2} \cos(q_1 + q_2) \\ 0 & 0 \end{bmatrix} \tag{2.73}$$

Böylece kinetik enerjinin çizgisel hıza bağlı olan parçası hesaplanır. Denklem (2.74)'de Kinetik enerjinin çizgisel hız ifadesi görülmektedir.

$$\frac{1}{2}m_1 v_{c1}^T v_{c1} + \frac{1}{2}m_2 v_{c2}^T v_{c2} = \frac{1}{2}\dot{q}\{m_1 J_{v_{c1}}^T J_{v_{c1}} + m_2 J_{v_{c2}}^T J_{v_{c2}}\}\dot{q} \tag{2.74}$$

Kinetik enerji teriminin açısal hız ifadesini oluşturalım. Denklem (2.75) açısal hız terimini ifade etmektedir.

$$\omega_1 = \dot{q}_1 k \omega_2 = (\dot{q}_1 + \dot{q}_2)k \tag{2.75}$$

Açısal kinetik enerji terimi denklem (2.76)'da görüldüğü gibidir.

$$\frac{1}{2}I_i \omega_i^2 \tag{2.76}$$

Denklem (2.76)'da I_i ilgili uzvun atalet tensörünü oluşturmaktadır. Denklem (2.76) ve (2.75) birleştirerek, denklem (2.77) elde edilir.

$$\frac{1}{2}\dot{q}^T\left\{I_1\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}+I_2\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}\right\}\dot{q} \tag{2.77}$$

Denklem (2.77) ve (2.74) birleştirilerek, $D(q)$ atalet veya kütle matrisi, denklem (2.78)'deki gibi yazılır.

$$D(q)=m_1J_{v_{c1}}^TJ_{v_{c1}}+m_2J_{v_{c2}}^TJ_{v_{c2}}+\begin{bmatrix}I_1+I_2 & I_2\\ I_2 & I_2\end{bmatrix} \tag{2.78}$$

Denklem (2.78)'in elemanları denklem (2.79)'daki gibidir.

$$\begin{aligned}d_{11}&=m_1l_{c1}^2+m_2(l_1^2+l_{c2}^2+2l_1l_{c2}\,cos\,q\,2)+I_1+I_2\\ d_{12}&=d_{21}=m_2(l_{c2}^2+l_1l_{c2}\,cos\,q\,2)+I_2\\ d_{22}&=m_2l_{c2}^2+I_2\end{aligned} \tag{2.79}$$

2 DOF robot kolun Euler-Lagrange denklemini oluşturmaya devam edelim. Denklem (2.80)'de denklem (2.66) terimleri bulunmaktadır.

$$\begin{aligned}c_{111}&=\frac{1}{2}\frac{\partial d_{11}}{\partial q_1}=0\\ c_{121}&=c_{211}=\frac{1}{2}\frac{\partial d_{11}}{\partial q_2}=-m_2l_1l_{c2}\,sin\,q_2=h\\ c_{221}&=\frac{\partial d_{12}}{\partial q_2}-\frac{1}{2}\frac{\partial d_{22}}{\partial q_1}=h\\ c_{112}&=\frac{\partial d_{21}}{\partial q_1}-\frac{1}{2}\frac{\partial d_{11}}{\partial q_2}=-h\\ c_{122}&=c_{212}=\frac{1}{2}\frac{\partial d_{22}}{\partial q_1}=0\\ c_{222}&=\frac{1}{2}\frac{\partial d_{22}}{\partial q_2}=0\end{aligned} \tag{2.80}$$

2 DOF robot kol için potansiyel enerji terimi denklem (2.81)'deki gibidir.

$$P_1 = m_1 g l_{c1} \sin q_1$$
$$P_2 = m_2 g (l_1 \sin q_1 + l_{c2} \sin(q_1 + q_2)) \quad (2.81)$$

Denklem (2.81) ile toplam potansiyel enerji terimi denklem (2.82)'de verilmektedir.

$$P = P_1 + P_2 = (m_1 l_{c1} + m_2 l_1) g \sin q_1 + m_2 l_{c2} g \sin(q_1 + q_2) \quad (2.82)$$

Denklem (2.82)'den yararlanarak denklem (2.67)'nin oluşturulması, denklem (2.83)'deki gibidir.

$$g_1 = \frac{\partial P}{\partial q_1} = (m_1 l_{c1} + m_2 l_1) g \cos q_1 + m_2 l_{c2} g \cos(q_1 + q_2)$$
$$g_2 = \frac{\partial P}{\partial q_2} = m_2 l_{c2} g \cos(q_1 + q_2) \quad (2.83)$$

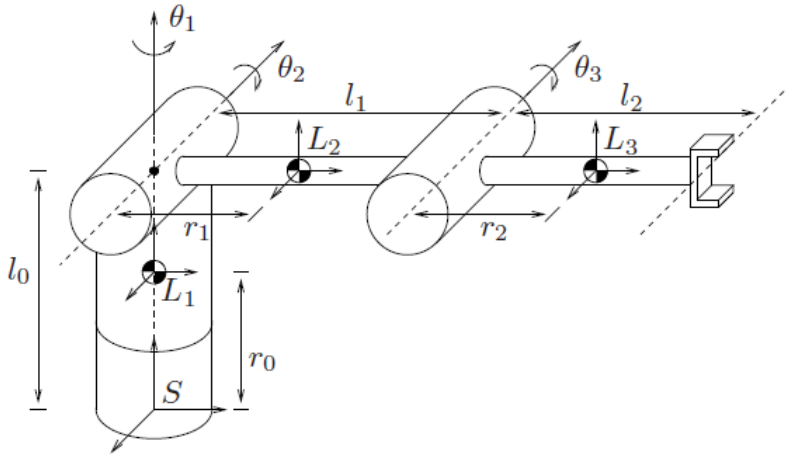
2 DOF robot kolun hareket denklemleri böylece kompakt bir ifadeyle denklem (2.84)'de verildiği gibidir.

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + c_{121}\dot{q}_1\dot{q}_2 + c_{211}\dot{q}_2\dot{q}_1 + c_{221}\dot{q}_2^2 + g_1 = \tau_1$$
$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + c_{112}\dot{q}_1^2 + g_2 = \tau_2 \quad (2.84)$$

Denklem (2.84) matris vektör gösterimi ile denklem (2.85)'deki gibi ifade edilir. Uygun durum değişkenlerinin seçilmesiyle (2.85) durum uzay denklemlerine dönüştürülebilir.

$$\begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} h\dot{q}_2 + & h\dot{q}_1 + h\dot{q}_2 \\ -h\dot{q}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \quad (2.85)$$

Şekil 2.18'de 3 DOF robot kolun açık kinematik zincir ifadesi bulunmaktadır. Bu tez çalışmasında pekiştirmeli öğrenme yaklaşımıyla 3 DOF robot kolun tork kontrolü yapılacaktır. Bu çalışmasında kullanılan simulasyon için, bu bölümde 3 DOF robot kolun dinamik denklemleri 2 DOF robota benzer şekilde oluşturulmuştur. (Spong, Mark W. ve diğ 2020)



Şekil 2.18: 3 DOF Robot Kol Dinamik Modeli.

Denklem (2.86)'da sistemin atalet veya kütle matrisi görülmektedir.

$$D(q) = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} = J_1^T D_1 J_1 + J_2^T D_2 J_2 + J_3^T D_3 J_3$$

$$\begin{aligned} d_{11} &= I_{y2}s_2^2 + I_{y3}s_{23}^2 + I_{z1} + I_{22}c_s^2 + I_{z3}c_{23}^2 \\ d_{12} &= d_{13} = d_{21} = 0 \\ d_{22} &= I_{x2} + I_{x3} + m_3 l_1^2 + m_2 r_1^2 + m_3 r_2^2 + 2m_3 l_1 r_2 c_3 \\ d_{23} &= I_{x3} + m_3 r_2^2 + m_3 l_1 r_2 c_3 \\ d_{31} &= 0 \\ d_{32} &= I_{x3} + m_3 r_2^2 + m_3 l_1 r_2 c_3 \\ d_{33} &= I_{x3} + m_3 r_2^2 \end{aligned} \qquad (2.86)$$

Denklem (2.86)'da $D_i \quad i = 1,2,3$ elemanları denklem (2.87)'deki gibidir.

$$D_i = \begin{bmatrix} m_i & 0 & 0 \\ 0 & m_i & 0\ 0 \\ 0 & 0 & m_i \\ I_{xi} & 0 & 0 \\ 0\ 0 & I_{yi} & 0 \\ 0 & 0 & I_{zi} \end{bmatrix} \tag{2.87}$$

Denklem (2.87)'nin m_i elemanı ilgili uzvun kütlesini, I_{xi}, I_{yi}, I_{zi} elemanları ilgili uzvun x, y, z eksenleri doğrultusundaki ataletlerini ifade etmektedir. Denklem (2.86)'daki sistemin Jacobian matrisleri denklem (2.88)'deki gibidir.

$$J_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad J_2 = \begin{bmatrix} -r_1 c_2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -r_1 & 0 \\ 0 & -1 & 0 \\ -s_2 & 0 & 0 \\ c_2 & 0 & 0 \end{bmatrix}$$

$$J_3 = \begin{bmatrix} -l_2 c_2 - r_2 c_{23} & 0 & 0 \\ 0 & l_1 s_3 & 0 \\ 0 & -r_2 - l_1 c_3 & -r_2 \\ 0 & -1 & -1 \\ -s_{23} & 0 & 0 \\ c_{23} & 0 & 0 \end{bmatrix} \tag{2.88}$$

Denklem (2.86) ile üç DOF robot kolun, denklem (2.69)'daki $C(q, \dot{q})$ coriolis ve merkez kaç kuvvetlerini denklem (2.89) ile oluşturabiliriz.

$$C_{ij}(q, \dot{q}) = \sum_{k=1}^{n} c_{ijk} \dot{q}_k = \frac{1}{2} \sum_{k=1}^{n} \left(\frac{\partial D_{ij}}{\partial q_k} + \frac{\partial D_{ik}}{\partial q_j} - \frac{\partial D_{kj}}{\partial q_i} \right) \dot{q}_k \tag{2.89}$$

Denklem (2.89)'un c_{ijk} terimleri denklem (2.90)'daki gibidir.

$$
\begin{aligned}
c_{112} &= (I_{y2} - I_{z2} - m_2 r_1^2) c_2 s_2 + (I_{y3} - I_{z3}) c_{23} s_{23} - m_3 (l_1 c_2 + r_2 c_{23})(l_1 s_2 \\
&\quad + r_2 s_{23}) \\
c_{113} &= (I_{y3} - I_{z3}) c_{23} s_{23} - m_3 r_2 s_{23} (l_1 c_2 + r_2 c_{23}) \\
c_{121} &= (I_{y2} - I_{z2} - m_2 r_1^2) c_2 s_2 + (I_{y3} - I_{z3}) c_{23} s_{23} - m_3 (l_1 c_2 + r_2 c_{23})(l_1 s_2 \\
&\quad + r_2 s_{23}) \\
c_{131} &= (I_{y3} - I_{z3}) c_{23} s_{23} - m_3 r_2 s_{23} (l_1 c_2 + r_2 c_{23}) \\
c_{211} &= (I_{z2} - I_{y2} - m_2 r_1^2) c_2 s_2 + (I_{z3} - I_{y3}) c_{23} s_{23} + m_3 (l_1 c_2 + r_2 c_{23})(l_1 s_2 \\
&\quad + r_2 s_{23}) \\
c_{223} &= -l_1 m_3 r_2 s_3 \\
c_{232} &= -l_1 m_3 r_2 s_3 \\
c_{233} &= -l_1 m_3 r_2 s_3 \\
c_{311} &= (I_{z3} - I_{y3}) c_{23} s_{23} + m_3 r_2 s_{23} (l_1 c_2 + r_2 c_{23}) \\
c_{322} &= l_1 m_3 r_2 s_3
\end{aligned} \tag{2.90}
$$

Son olarak denklem (2.68)'daki $g(q)$ yerçekimi terimi denklem (2.91)'deki gibi yazılır.

$$
g(q) = \frac{\partial P}{\partial q}
$$

$$
P(q) = m_1 g h_1(q) + m_2 g h_2(q) + m_3 g h_3(q) \tag{2.91}
$$

Denklem (2.91)'deki $h_i i = 1,2,3$ ilgili uzvun kütle merkezinin koordinatlarını ifade eder ve robotun ileri kinematik modelinden denklem (2.92)'deki gibi hesaplanır.

$$
\begin{aligned}
h_1(q) &= r_0 \\
h_2(q) &= l_0 - r_1 \sin q_2 \\
h_3(q) &= l_0 - l_1 \sin q_2 - r_2 \sin(q_1 + q_2)
\end{aligned} \tag{2.92}
$$

Denklem (2.92) ve (2.91) ifadeleri birleştirilerek, robotun $g(q)$ terimi denklem (2.93)'deki gibi yazılır.

$$
g(q) = \frac{\partial P}{\partial q} = \begin{bmatrix} 0 \\ -(m_2 g r_1 + m_3 g l_1) \cos q_2 - m_3 r_2 \cos(q_2 + q_3) \\ -m_3 g r_2 \cos(q_2 + q_3) \end{bmatrix} \tag{2.93}
$$

Denklem (2.87), (2.90) ve (2.93) terimleri birleştirilerek 3 DOF robot kolun ileri dinamik modeli oluşturulur. (Murray ve diğ. 1994)

2.5 Dinamik Denklemlerin Sayısal Çözümleri

Bilim insanları tasarladıkları sistemleri matematiksel olarak ifade ederler. Tasarlanan sistem elektromekanik, kimyasal, biyolojik vb farklı alanlardan oluşabilir. Tasarlanan sistemlerin matematiksel ifadelerini integral denklemler, diferansiyel denklemler, fark denklemleri veya integro-diferansiyel denklemlerle oluşturabiliriz. Oluşturulan matematiksel modeller, fiziksel sistemleri doğru bir biçimde ifade etmesi durumunda, bu denklemler üzerinden sistem analizi ve sentezi gerçekleştirilir. Bu bölümde, bölüm (2.4)'de oluşturulan robotun dinamik denklemlerinin çözümleri yapılacaktır. Robot sistemlerinin uç ve iç değişkenleri arasındaki ilişki lineer olmayan diferansiyel denklemlerle ifade edilir. Bu diferansiyel denklemlerin analitik olarak genel çözümlerinin bulunması, birçok denklem için mümkün olmamaktadır. Diferansiyel denklemlerin analitik çözümlerinin olmadığı durumlarda sayısal çözümleme yapılmaktadır. Sayısal çözümleme metodları iteratif olarak sistemin çözümünü elde etmektedir. Sayısal çözümleme sürekli zamanlı tanımlanmış diferansiyel denklemleri ayrık zamanlı formuna geçirerek iteratif olarak çözmektedir. Sayısal çözümleme metodları içerisinde geliştirilen birçok algoritma mevcuttur. Bu algoritmaların en bilineni euler ayrıklaştırması veya integrasyonu olarak bilinmektedir. Bu ifade türevin ayrık zamanlı tanımından türetilmektedir ve denklem (2.94)'deki gibi ifade edilmektedir.

$$\frac{dy}{dt} = \lim_{\triangle t \to 0} \frac{y(t+1)-y(t)}{\triangle t} \tag{2.94}$$

Denklem (2.94) ileri farklar metodu ile türev ifadesinin ayrık zamanlı hali olarak düşünülebilir. Sayısal türev metodları arasında, geri farklar ve merkezi farklar metodları bulunmaktadır. Denklem (2.94)'deki $\triangle t$ ifadesi zamansal farkı ifade etmektedir. Denklem (2.94) kullanarak birinci dereceden homojen olmayan bir diferansiyel denklem, denklem (2.95)'deki gibi fark denklemi haline getirilebilir.

$$\begin{aligned} \frac{dy(t)}{dt} &= 5u(t) \\ y(n+1) &= y(n) + 5u(n) \qquad (t = nTs), Ts = 1 \end{aligned} \tag{2.95}$$

Denklem (2.95)'in ayrık zamanlı fark denklemi haline gelmesiyle, bir başlangıç koşulu için denklem iteratif şekilde çözülür. Euler metodunun bir problemide sistemin çözümünü yaparken belirli bir hata ile hesaplamaktadır. Euler metodunun problemlerinin giderilmesi için Runge Kutta metodunu bu tez kapsamında kullanılacaktır.

Runge Kutta'nın diferansiyel denklem sistemleri veya integral hesaplaması için 4 adımda hesaplama sistematiğini kullanarak, bir önceki bölümde elde edilen diferansiyel denklem sistemlerinin sayısal olarak çözümlemesi yapılır. (İplikçi, Serdar. 2018)

2.5.1 Runge Kutta Metodu ile Dinamik Denklemlerin Çözümü

İplikçi, Serdar. (2018) Runge Kutta metodu ile bir önceki bölümde denklemleri veriler iki DOF ve üç DOF robotun sayısal çözümlemesi yapılacaktır. Böylece robot kol kontrolü için gerekli olan alt yapı ve bölüm 4'de önerilen pekiştirmeli öğrenme yaklaşımıyla robot kol kontrolünün simulasyon ortamında yapılması için bir alt yapı oluşturulacaktır. Fiziksel sistemlerin simulasyonlarını oluştururken sistemi ifade eden matematiksel modellerin yazılıp iteratif olarak çözümlenmesi gerekmektedir. Hatırlanacağı üzere en genel halde lineer olmayan zamanla değişen çok girişli çok çıkışlı (MIMO) diferansiyel denklem sistemi denklem (2.96)'daki gibidir.

$$\begin{bmatrix} \dot{x}_1(t) \\ \vdots \\ \dot{x}_N(t) \end{bmatrix} = \begin{bmatrix} f_1(x_1(t),\dots,x_N(t),u_1(t),\dots.u_M(t),t) \\ \vdots \\ f_N(x_1(t),\dots,x_N(t),u_1(t),\dots.u_M(t),t) \end{bmatrix}$$

$$y(t) = g(x_1(t),\dots,x_N(t),u_1(t),\dots.u_M(t),t) \qquad (2.96)$$

Denklem (2.96)'da t sürekli zaman değişkeni, $x_i(t) i = 1,2,\dots,N$'ler sistemin durum değişkenleri, $u_i(t) i = 1,2,\dots,M$ 'ler sistemin giriş işaretini, $y(t)$ sistemin çıkış denklemidir. f_i, g fonksiyonları robotik sistemin dinamik denklemlerini oluşturmaktadır ve t zamana bağlı olarak değişmektedir. Denklem (2.96)'da $t = nT_s$ olmak üzere sistemin giriş işareti ve durumlarının nT_s anlarındaki değerlerinin bilindiğini varsayalım. Burada T_s sistemin örnekleme periyodunu oluşturmaktadır. Sistemin nT_s anındaki çıkışı, denklem (2.97)'deki gibidir.

$$y(nT_s) = g(x_1(nT_s),\dots,x_N(nT_s),u_1(nT_s),\dots.u_M(nT_s),nT_s) \qquad (2.97)$$

Giriş işareti $u_i(t) i = 1,2,\dots,M$ 'ların iki örnekleme aralığında değişmediğini varsayarak, bir sonraki örnekleme anı olan $(n+1)T_s$sistemin durum değişkenlerini ve çıkışın alacağı değeri bulmaya çalışıyoruz.

Bunun için dördüncü dereceden Runge Kutta algoritması kullanılabilir. $(n+1)T_s$ anında sistemin durum değişkenlerinin alacağı değer denklem (2.98)'de görüldüğü gibidir.

$$\begin{aligned}\hat{x}_1(n+1) &= x_1(n) + \frac{K_{11}(n)}{6} + \frac{K_{12}(n)}{3} + \frac{K_{13}(n)}{3} + \frac{K_{14}(n)}{6} \\ &\vdots \\ \hat{x}_N(n+1) &= x_N(n) + \frac{K_{N1}(n)}{6} + \frac{K_{N2}(n)}{3} + \frac{K_{N3}(n)}{3} + \frac{K_{N4}(n)}{6}\end{aligned} \quad (2.98)$$

Denklem (2.98)'de $K_{ij} i = 1,2,\ldots, Nj = 1,2,3,4$ terimleri, denklem (2.99)'daki gibi hesaplanmaktadır.

$$\begin{aligned}K_{11}(n) &= T_s f_1(\hat{x}_1(n),\ldots,\hat{x}_N(n),u_1(n),\ldots,u_M(n),n) \\ &\vdots \\ K_{N1}(n) &= T_s f_N(\hat{x}_1(n),\ldots,\hat{x}_N(n),u_1(n),\ldots,u_M(n),n) \\ &- \\ K_{12}(n) &= T_s f_1(\hat{x}_1(n)+\frac{K_{11}(n)}{2},\ldots,\hat{x}_N(n)+\frac{K_{N1}(n)}{2},u_1(n),\ldots,u_M(n),n) \\ &\vdots \\ K_{N2}(n) &= T_s f_N(\hat{x}_1(n)+\frac{K_{11}(n)}{2},\ldots,\hat{x}_N(n)+\frac{K_{N1}(n)}{2},u_1(n),\ldots,u_M(n),n) \\ &- \\ K_{13}(n) &= T_s f_1(\hat{x}_1(n)+\frac{K_{12}(n)}{2},\ldots,\hat{x}_N(n)+\frac{K_{N2}(n)}{2},u_1(n),\ldots,u_M(n),n) \\ &\vdots \\ K_{N3}(n) &= T_s f_N(\hat{x}_1(n)+\frac{K_{12}(n)}{2},\ldots,\hat{x}_N(n)+\frac{K_{N2}(n)}{2},u_1(n),\ldots,u_M(n),n) \\ &- \\ K_{14}(n) &= T_s f_1(\hat{x}_1(n)+K_{13}(n),\ldots,\hat{x}_N(n)+K_{N3}(n),u_1(n),\ldots,u_M(n),n) \\ &\vdots \\ K_{N4}(n) &= T_s f_N(\hat{x}_1(n)+K_{13}(n),\ldots,\hat{x}_N(n)+K_{N3}(n),u_1(n),\ldots,u_M(n),n)\end{aligned} \quad (2.99)$$

Benzer şekilde $(n+1)T_s$ anında sistemin çıkışının alacağı değer denklem (2.100)'ile ifade edilmektedir.

$$y((n+1)T_s) = g(x_1((n+1)T_s),\ldots,x_N((n+1)T_s),u_1((n+1)T_s),\ldots u_M((n+1)T_s),nT_s) \quad (2.100)$$

Denklemler matris vektör notasyonuyla daha kompakt bir formda denklem (2.101)'deki gibidir.

$$\begin{aligned} \hat{\boldsymbol{x}}(n+1) &= \hat{\boldsymbol{f}}(\hat{\boldsymbol{x}}(n), \boldsymbol{u}(n), n) \\ \hat{y}(n+1) &= g(\hat{\boldsymbol{x}}(n), \boldsymbol{u}(n), n) \end{aligned} \tag{2.101}$$

Denklem (2.101) kullanılarak $t = nT_s$anında durum değişkenlerinin şu anki değeri olan $x_1(n), \ldots, x_N(n)$ ve giriş işaretinin şu anki değeri olan $u_1(n), \ldots, u_M(n)$ değerleri bilindiğinde, bir sonraki örnekleme anı için $(t + T_s = (n+1)T_s)$ anında sistemin durum değişkenlerinin ve çıkışın alacağı değer hesaplanır. Denklemlere dikkat edilirse lineer olmayan sürekli zamanlı durum uzay denklemlerini ayrıklaştırarak başlangıç değer problemini sayısal olarak çözmektedir. Bir sonraki bölümde robot kontrol konusu içerisinde bu denklemlerin matlab simulasyon sonucları verilecektir.

2.6 Robot Kol Kontrol Algoritmaları

Bu bölümde model tabanlı kontrol algoritmaları ele alınacaktır. Önceki bölümlerde robot kolun dinamik denklemleri durum uzay formunda oluşturuldu. Bu bölümde robotun dinamik denklemlerini kullanarak kontrolcü tasarımı gerçekleştireceğiz. Şekil 2.15 'de robot dinamiği ileri ve ters dinamik olarak ikiye ayrılmaktadır. Ters dinamik, robotun eklem hareketlerinin $q_i, \dot{q}_i, \ddot{q}_i i = 1,2,\ldots,n$ açısal konum, açısal hız ve açılsal ivme terimiyle, robotun hareketi sonucunda her bir eklemde gerçekleşen tork ifadelerini $\tau_i i = 1,2,\ldots,n$ bulmayı hedefler. İleri dinamik denklemler robotun her bir eklemine uygulanan tork ifadesi $\tau_i i = 1,2,\ldots,n$ ile eklem değişkenlerinin $q_i, \dot{q}_i, \ddot{q}_i i = 1,2,\ldots,n$ bulunmasını hedefler bu durum denklem (2.69)'da görüldüğü gibidir. Denklem (2.69) tekrardan düzenlenip denklem (2.102)'deki n DOF robot için ters dinamik denklem elde edilir.

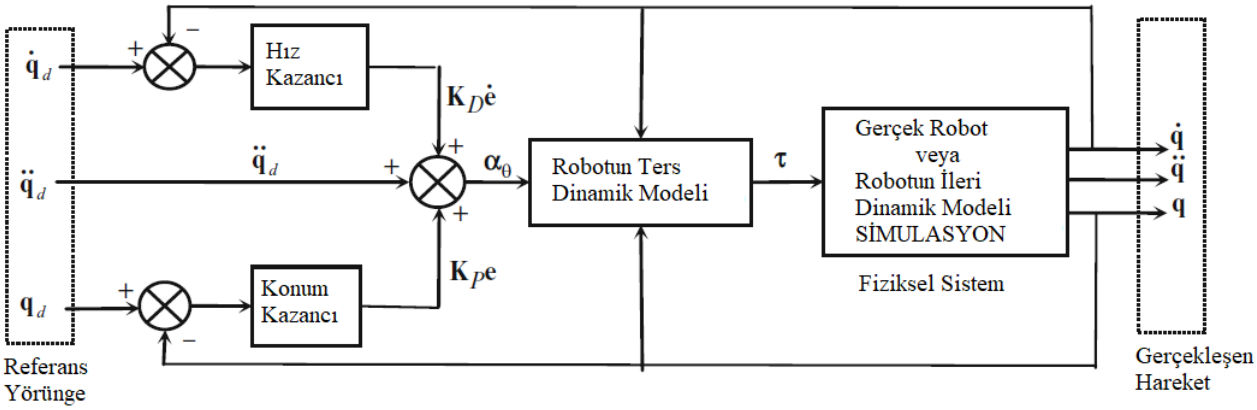
$$\begin{aligned} D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) &= \tau && \text{İleri dinamik denklemler} \\ \ddot{q} &= D(q)^{-1}\{\tau - C(q,\dot{q})\dot{q} - g(q)\} && \text{Ters dinamik denklemler} \end{aligned} \tag{2.102}$$

Denklem (2.102) dikkat edilirse, denklemin solundaki ifade çift katlı integratör sistemi oldugu görülmektedir. Hatırlanacağı üzere $D(q)$ matrisi pozitif tanımlı bir matris olduğu için herzaman $D(q)^{-1}$ hesaplanabilir. Robot kontrol algoritmalarının geliştirilmesi için denklem (2.102) kullanılmaktadır. Bu alt bölümde robotun dinamik

denklemlerini inceleyip iki DOF robot için model tabanlı kontrolcü tasarımı yapılacaktır. Robot için geliştirilen model tabanlı kontrol algoritmalarına örnek olarak hesaplanmış tork kontrol metodu ve ileri beslemeli kontrol yapısı örnek olarak gösterilebilir. Bu tez çalışmasında hesaplanmış tork kontrol metodu ile robot iki DOF robotun model tabanlı kontrolü oluşturulacaktır. (Lewis, Frank L. ve diğ 2006)

2.6.1 Robot Kol Hesaplanmış Tork Kontrolü

Bu alt bölümde denklem (2.85) ile ifade edilen 2 DOF robotun ileri dinamik modelinden, denklem (2.102) ile ters dinamik modeli oluşturulup hesaplanmış tork kontrol metodu uygulancaktır. 2 DOF robotun ters dinamik modeli denklem (2.102)'den görüleceği üzere lineer olmayan zamanla değişen çok girişli çok çıkışlı diferansiyel denklem formundadır. Bu denklemleri çözmek için bölüm 2.5.1'de önerilen Runge Kutta metodu kullanılacaktır. Yapılan simulasyon sonuçları matlab ortamında oluşturulmuştur. Şekil 2.19'da hesaplanmış tork kontrol PD metodu görülmektedir.



Şekil 2.19: Hesaplanmış Tork Kontrol PD Metodu Şeması.

Şekil 2.19'da hesaplanmış kontrol PD metodu görülmektedir. Şekil üzerinde $q_d, \dot{q}_d, \ddot{q}_d$ robotun yörünge planlayıcısından hesaplanmaktadır. Robotun çalışma uzayı içerisinde belirli bir noktaya erişmesi için uygun konum ve oryantasyon matrisi ${}^0_n\boldsymbol{T}$ verilmiş olsun. Bu matris üzerinden robotun ters kinematik denklemleri çözümü gerçekleştirilir, böylece $q_i \;\; i = 1,2,\ldots,n$ için gerekli eklem değişkenleri belirlenir. Robotun hedeflenen konum ve oryantasyon hareketini gerçekleştirmesi için $q_i \;\; i = 1,2,\ldots,n$ ve yörünge planlayıcısı algoritması kullanılarak robotun eklem uzayındaki referans yörüngeler olan $q_d, \dot{q}_d, \ddot{q}_d$ hesaplanır.

$q_d, \dot{q}_d, \ddot{q}_d$hesaplanması ile robotun hedeflenen bu yörüngeyi gerçekleştirmesi için gerekli olan tork kontrolü oluşturulur. Hesaplanmış tork kontrol metodu şekil 2.19'da görüldüğü gibi robotun referans işareti olan $q_d, \dot{q}_d, \ddot{q}_d$ ile, hız ve konum kazancıyla hata dinamiklerini $e, \dot{e}$ oluşturmaktadır. Hesaplanan hata dinamikleri robotun ters dinamik denklemlerine giriş işareti olarak uygulanır ve robotun ters dinamik modeli çözümlenerek tork değerleri elde edilir. Denklemlerin çözümlerinin sonucunda elde edilen tork ifadeleri τ fiziksel sisteme veya simulasyonda robotun ileri dinamik modeline giriş işareti olarak uygulanır. Bu bölümde simulasyon üzerinden çalışıldığı için robotun ileri dinamik denklemleri giriş işareti τ ifadeleriyle beraber Runge Kutta algoritması ile integrasyonu yapılmaktadır. İntegrasyon sonucunda robotun fiziksel olarak gerçekleştirdiği eklem parametreleri olan $q, \dot{q}, \ddot{q}$ hesaplanmaktadır. Hesaplanan bu parametreler üzerinden denklem (2.103)'deki hata dinamikleri oluşturulur.

$$e = q_d - q$$
$$\dot{e} = \dot{q}_d - \dot{q} \tag{2.103}$$

Hesaplanmış tork metodu hata dinamikleri olan $e, \dot{e}$ üzerinden bir kontrolü tasarımını oluşturarak denklem (2.104)'ü sağlamaya çalışır.

$$\lim_{t \to \infty} e, \dot{e} = 0 \tag{2.104}$$

Denklem (2.104) hatanın sıfıra yaklaşması robotun istenen referans yörüngeyi takip etmesi için gerekli olan tork τ ifadelerini oluşturması demektir. Denklem (2.105)'de hesaplanmış tork kontrol algoritmasının matematiksel ifadesi bulunmaktadır.

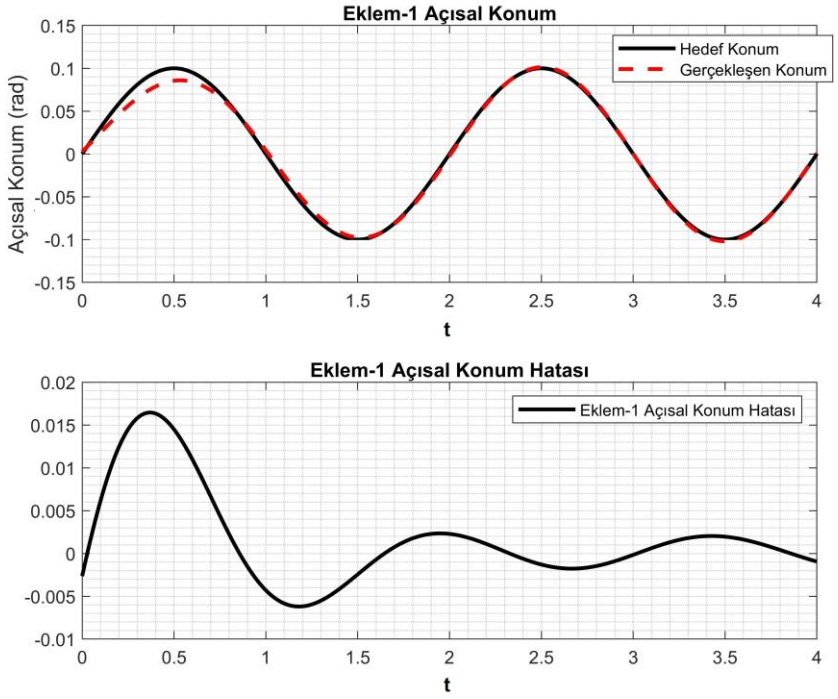
$$\tau = \boldsymbol{D}(\boldsymbol{q})\boldsymbol{q_d} + K_v\dot{\boldsymbol{e}} + K_p\boldsymbol{e} + K_i \int \boldsymbol{e} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{2.105}$$

Hesaplanmış tork kontrol metodunun bir avantajı 2 DOF sistemin lineer olmayan dinamikleri üzerinden kontrolcü tasarımı yerine, sistemin hata dinamikleri üzerinden lineer bir kontrolcü tasarlıyor olmasıdır. Tablo 2.4'de hesaplanmış tork kontrol metoduyla tasarlanmış PID kontrol algoritması görülmektedir. (Lewis, Frank L. ve diğ 2006)

Tablo 2.4: 2 DOF Robot Kol Hesaplanmış Tork Kontrol Metodu.

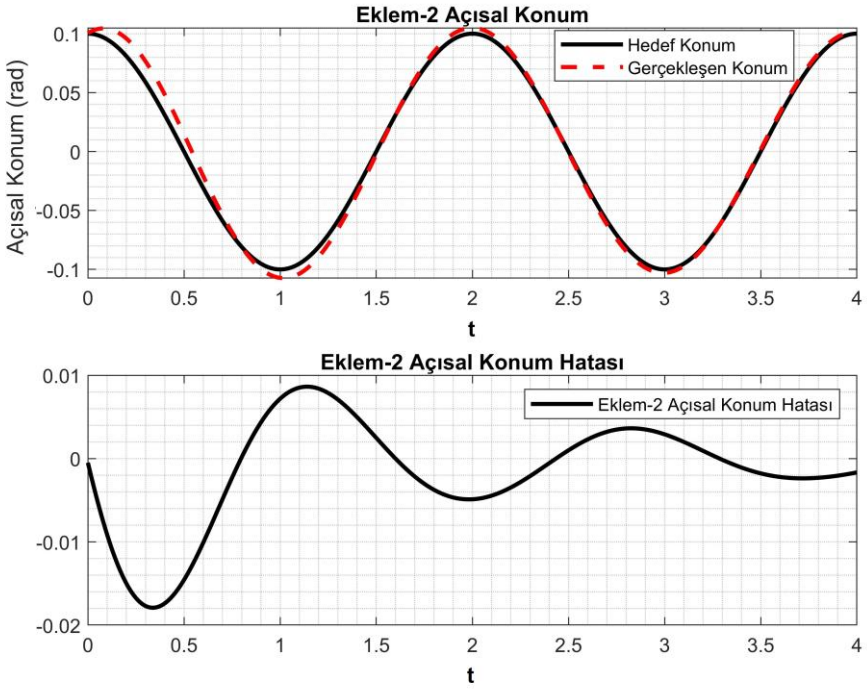
ALGORİTMA : 2 DOF Robot Kol Hesaplanmış Tork Kontrol Metodu		
1	**:**	tsimulasyon = 4 *Algoritmanın simülasyon zamanı*
2	**:**	Ts = 1e-2 *Örnekleme periyodu*
3	**:**	Tork =0, q_b=0 *Robotun tork ve açısal konum değerlerinin başlangıcı*
4	**:**	Ksi = 1 *Kontrolcünün sönüm oranı*
5	**:**	Wn = 1/Ksi *Kontrolcünün sönümsüz doğal frekansı*
6	**:**	Kp = 10, Kv = 2, Ki = 5 *PID kontrolcünün parametreleri*
7	**:**	L1 = 1, L2 = 1 cm *2 DOF robotun uzuv uzunlukları*
8	**:**	M1 = 2, M2 = 2 kg g=9.91 *2 DOF robotun uzuv kütleleri ve yerçekimi*
9	**:**	x_d, y_d *2 DOF robotun çalışma uzayında gitmesi gereken konum*
10	**:**	q_{1d}, q_{2d} = Ters Kinematik (x_d, y_d) *Robotun ters kinematik çözümü*
11	**:**	$q_{1d}, q_{2d}, \dot{q}_{1d}, \dot{q}_{2d}, \ddot{q}_{1d}, \ddot{q}_{2d}$ = *Yörünge Planlayıcı* (q_b,q_{1d}, q_{2d},Ts,tsimulasyon)
12	**:**	for i=1:Ts:tsimulasyon
13	**:**	[$\widehat{\boldsymbol{q}}$,D,C]=RungeKutta(M1,M2,L1,L2,g,$\boldsymbol{q}$,Tork,Ts) *Robotun integrasyonu*
14	**:**	$\boldsymbol{e} = \boldsymbol{q}_d - \boldsymbol{q}$ *Hata dinamiklerini hesapla*
15	**:**	$\boldsymbol{\tau} = \boldsymbol{D}(\boldsymbol{q})\boldsymbol{q_d} + K_v\dot{\boldsymbol{e}} + K_p\boldsymbol{e} + K_i \int \boldsymbol{e} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ *Hesaplanmış tork PID denklemi*
16	**:**	*Son*

Tablo 2.4'de ifade edilen algoritmanın matlab simulasyon sonuçları şekil 2.20'de robot kolun birinci uzvunun hedef ve gerçekleşen açısal konum değeri, oluşan açısal konum hata değeri görülmektedir.



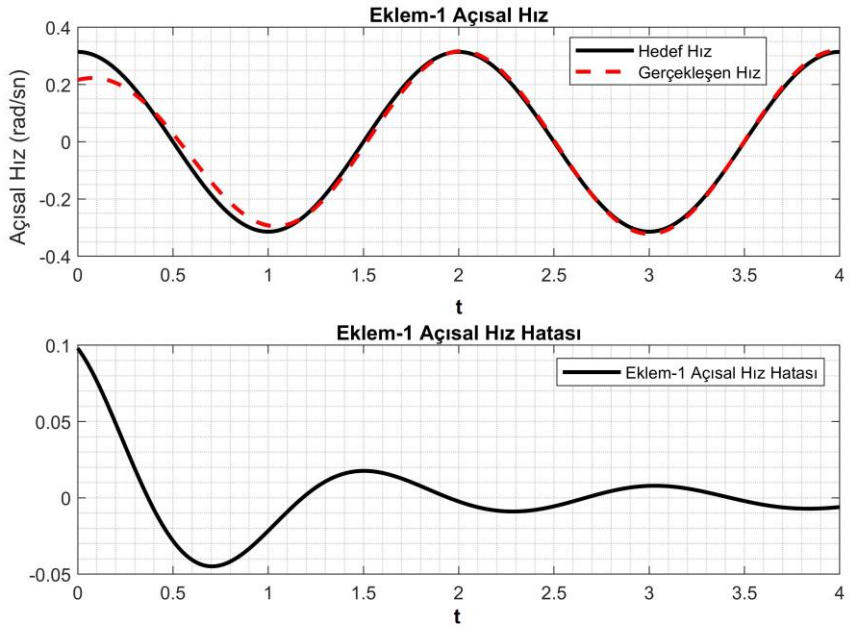
Şekil 2.20: 1. Ekleminin Yörüngesi, Gerçekleşen Açısal Konum ve Hata.

Şekil 2.21'de robot kolun ikinci uzvunun hedef ve gerçekleşen açısal konum değeri, oluşan açısal konum hata değeri görülmektedir.



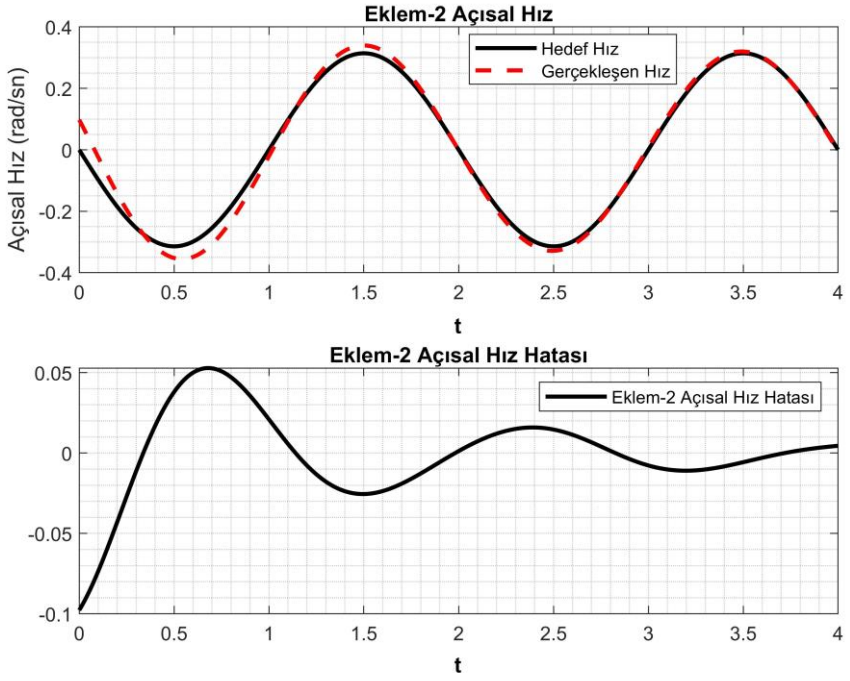
Şekil 2.21: 2. Ekleminin Yörüngesi, Gerçekleşen Açısal Konum ve Hata.

Şekil 2.22'de robot kolun birinci uzvunun hedef ve gerçekleşen açısal hız değeri, oluşan açısal hız hata değeri görülmektedir.



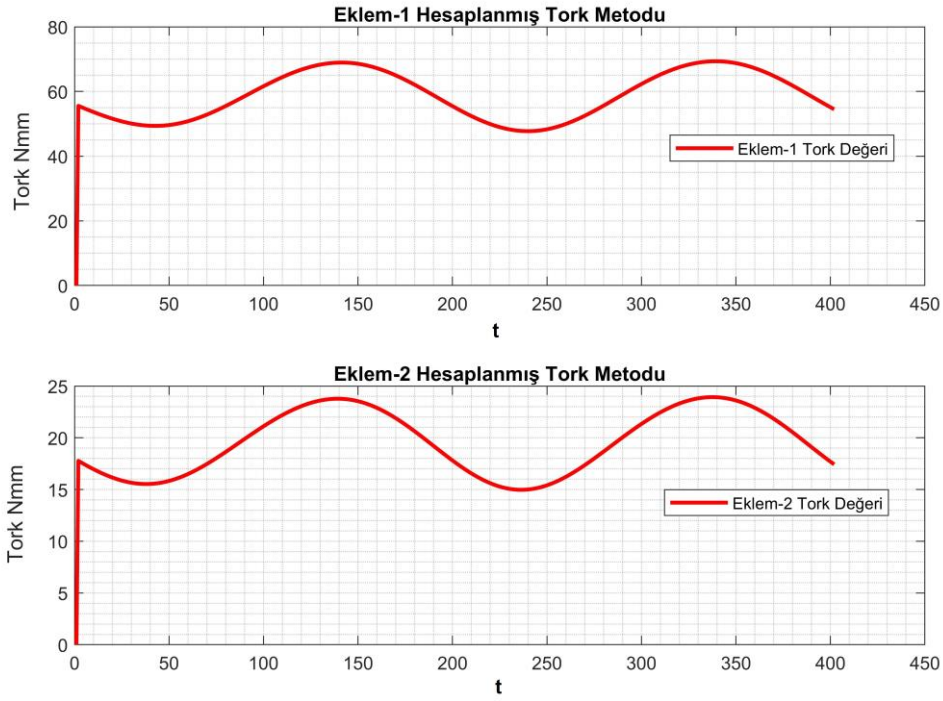
Şekil 2.22: 1. Ekleminin Yörüngesi, Gerçekleşen Açısal Hız ve Hata.

Şekil 2.23'de robot kolun ikinci uzvunun hedef ve gerçekleşen açısal hız değeri ve oluşan açısal hız hata değeri görülmektedir.



Şekil 2.23: 2. Ekleminin Yörüngesi, Gerçekleşen Açısal Hız ve Hata.

Şekil 2.24'de robot kolun hedeflenen yörüngeyi takip ederken birinci ve ikinci ekleme uygulamış olduğu tork değeri görülmektedir.



Şekil 2.24: 1. Ve 2. Ekleminin Tork Değeri.

2.7 Optimal Kontrol

Kirk, Donald E. (1970) Bu bölümde dinamik sistemlerin optimal kontrolü ele alınacaktır. Optimal kontrol, dinamik bir sistemi kontrol ederken bazı kısıtlamaları sağlamasını ve belirlenen bir amaç fonksiyonunu mininmum veya maksimum yapmasını hedeflemektedir. Fiziksel sistemlerin dinamik denklemleri en genel biçimde denklem (2.106)'daki gibidir.

$$\begin{bmatrix} \dot{x}_1(t) \\ \vdots \\ \dot{x}_N(t) \end{bmatrix} = \begin{bmatrix} f_1(x_1(t), \ldots, x_N(t), u_1(t), \ldots . u_M(t), t) \\ \vdots \\ f_N(x_1(t), \ldots, x_N(t), u_1(t), \ldots . u_M(t), t) \end{bmatrix}$$
$$y(t) = g(x_1(t), \ldots, x_N(t), u_1(t), \ldots . u_M(t), t) \quad (2.106)$$

Denklem (2.107)'de denklem (2.106)'nın matris vektör gösterimi ile daha kompakt bir biçimde görülmektedir.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$$
$$\boldsymbol{y}(t) = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, t) \quad (2.107)$$

Denklem (2.107)'de görüldüğü gibi fiziksel sistem lineer olmayan zamanla değişen çok girişli çok çıkışlı diferansiyel denklem sistemiyle ifade edilmektedir. $[t_0, t_f]$ zaman aralığında kontrol işareti $\boldsymbol{u}$, kontrol geçmişi olarak isimlendirilmektedir. $[t_0, t_f]$ zaman aralığında sistemin durum vektörü $\boldsymbol{x}$ 'ler sistemin yörüngesini oluşturmaktadır. Denklem (2.108)'de sistemin sağlaması gereken başlangıç değer ve sınır değer kısıtları görülmektedir.

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \boldsymbol{x}(t_f) = \boldsymbol{x}_f \quad (2.108)$$

Denklem (2.109)'da sistemin değişkenlerinin kısıtları görülmektedir.

$$\underline{X} \leq \boldsymbol{x}(t) \leq \bar{X} \quad (2.109)$$

Benzer şekilde kontrol işaretinin sağlaması gereken kısıtlar denklem (2.110)'da görülmektedir.

$$\underline{U} \leq \boldsymbol{u}(t) \leq \bar{U} \tag{2.110}$$

$[t_0, t_f]$ zaman aralığında denklem (2.110)'u sağlayan kontrol işaretine, uygulanabilir kontrol olarak ifade edilir. Benzer şekilde $[t_0, t_f]$ zaman aralığında denklem (2.108) ve (2.109) kısıtlamalarını sağlayan durum değişkenleri, uygulanabilir yörünge olarak ifade edilir. Denklem (2.111)'de maaliyet (cost) fonksiyonu görülmektedir.

$$J = h(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \tag{2.111}$$

Denklem (2.111)'de $h, g \in \mathbb{R}$ ayrıca denklemin sağ tarafındaki integralin üst sınırı t_f sabit veya serbest olabilmektedir. Optimal kontrol problemi, uygulanabilir kontrol işareti içerisinden uygun bir kontrol işareti $\boldsymbol{u}^*$oluşturmayı hedefler, oluşturululan bu kontrol işareti $\boldsymbol{u}^*$ denklem (2.112)'de görülen sistemi sağlamalı ayrıca denklem (2.113)'deki performans kriterini minimum yapmalıdır.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{2.112}$$

$$J = h(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \tag{2.113}$$

Denklem (2.113)'deki performans kriterini minimum yapan $\boldsymbol{u}^*$ kontrol işareti, uygulanabilir yörünge üzerinden denklem (2.113)'ü sağlayan optimum yörünge $\boldsymbol{x}^*$ olmaktadır. $(\boldsymbol{u}^*, \boldsymbol{x}^*)$ optimal yörünge ve kontrol çiftdir. Optimal kontrol teorisi denklem (2.112) ve (2.113)'ü sağlayan uygun $(\boldsymbol{u}^*, \boldsymbol{x}^*)$ bulmayı garanti etmemektedir. Optimal kontrol probleminde performans kriteri olan J fonksiyonunun global çözümünü aramaktadır. Amaç fonksiyonu olan J'in maksimizasyonu yapılacaksa fonksiyon tekrar düzenlenerek $\min_u -J$ oluşturulur. Optimal kontrol açık çevrim ve kapalı çevrim olarak iki kısımda incelenebilir. Denklem (2.114)'de kapalı çevrim optimal kontrol kuralı verilmektedir.

$$\boldsymbol{u}^* = \pi(\boldsymbol{x}(t), t) \tag{2.114}$$

Denklem (2.114)'de π optimal kontrol kanunu veya optimal politika (optimal policy) olarak isimlendirilmektedir. Denklem (2.114) optimal kontrol kanunu için kapalı çevrim bir yapı oluşturmaktadır. Denklem (2.114)'ü sağlayan bir optimal kontrol kanunu olarak denklem (2.115) önerilebilir.

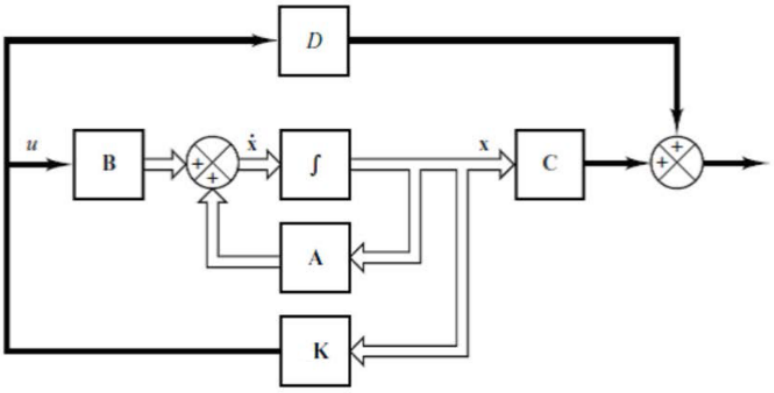
$$\pi(\boldsymbol{x}(t), t) = F\boldsymbol{x}(t) \tag{2.115}$$

Denklem (2.115)'in özel bir hali olarak modern kontrol teorisinde durum geri beslemeli kontrol önerilmiştir. Sistemin durumları K kazanç matrisi ile geri besleme yapılmaktadır. K kazanç matrisinin optimal olarak belirlenmesiyle birlikte uygulanan kontrol işareti optimal bir kontrol kanunu oluşturmaktadır. Denklem (2.116)'da durum geri beslemeli kontrol kuramı görülmektedir.

$$\boldsymbol{u} = K\boldsymbol{x} \tag{2.116}$$

Denklem (2.116)'da belirlenen K kazanç matrisi sistemin durumlarından, sistemin girişine doğru bir geri besleme oluşturmaktadır. Bu durum şekil 2.24'de görüldüğü gibidir. Denklem (2.116)'da K kazanç matrisi sistem için yazılmış olan (2.113)'deki performans kriterini sağlaması durumunda denklem (2.117)'deki gibi ifade edilmektedir.

$$\boldsymbol{u}^* = K\boldsymbol{x} \tag{2.117}$$



Şekil 2.25: Durum Uzayında Durum Geri Beslemeli Kontrol Şeması.

Optimal kontrol kanunun açık çevrim olması durumunda optimal kontrol kuralı denklem (2.118)'de görüldüğü gibidir.

$$\boldsymbol{u}^* = e(\boldsymbol{x}(t_0), t) \tag{2.118}$$

Denklem (2.118)'de optimal kontrol işareti $\boldsymbol{u}^*$ durumların sadece başlangıç değerleri için oluşturulmaktadır. (Pavone, Marco. 2018)

2.7.1 Optimal Kontrol Çözüm Metodları

Optimal kontrol teorisi denklem (2.112) ile ifade edilen sistemi, denklem (2.113) ile oluşturulan amaç fonksiyonunu, minimum veya maksimum yapmayı hedeflemektedir. Matematiksel olarak fiziksel sistemler sürekli zamanlı veya ayrık zamanlı olarak tanımlanmaktadır. Fiziksel sistem sürekli zamanlı olarak ifade ediliyorsa denklem (2.119) ile optimal kontrol problem ifade edilmektedir. (Hagan, Martin T. ve diğ. 2014)

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= f(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \\ J(\boldsymbol{x}(0)) &= h(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau) d\tau \end{aligned} \tag{2.119}$$

Denklem (2.119) J maliyet fonksiyonunu, $h(\boldsymbol{x}(t_f), t_f)$ terminal (sınır koşulu) maaliyetini ifade etmektedir. Denklem (2.119)'un ayrık zamanlı gösterimi matematiksel olarak denklem (2.120)'deki gibidir.

$$\begin{aligned} \boldsymbol{x}_{k+1} &= f(\boldsymbol{x}_k, \boldsymbol{u}_k, k) \\ J(\boldsymbol{x}_0) &= h_N(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} g(\boldsymbol{x}_k, \boldsymbol{u}_k, k) \end{aligned} \tag{2.120}$$

Denklem (2.120) ve denklem (2.119)'da ifade edilen optimal kontrol problemi integral ve toplam ifadelerinin üst sınırlarına bağlı olarak optimal kontrol problem, sonsuz ufuk veya sınırlı ufuk olarak incelenmektedir. Bu durum optimal kontrol probleminin çözümü için uygun stratejinin belirlenmesini gerektirmektedir. Optimal kontrol probleminde tanımlı olan J maliyet fonksiyonu tasarımcıya göre değişmektedir.

Genel olarak J maaliyet fonksiyonu minimum zaman, minimum enerji, minimum yakıt vb kısıtları oluşturacak şekilde matematiksel olarak formülasyonu yapılır. Denklem (2.120) ayrık zamanlı olarak tanımlandığı için bu problemin çözümünde matematiksel olarak dinamik programlama yaklaşımı kullanılmaktadır. Dinamik programlama pekiştirmeli öğrenme bölümünde incelenecektir. Denklem (2.119) HJB denklemi olarak ifade edilmektedir. Bu denklem lineer olmayan kısmi diferansiyel denklem olarak karşımıza çıkmaktadır. Bu denklemin çözümü için varyasyon hesabı veya Hamiltonian yaklaşımıyla iteratif çözüm yaklaşımı kullanılmaktadır. Optimal kontrol teorisinde, üzerinde çalıştığımız fiziksel sistemin matematiksel formulasyonu lineer veya lineer olmayan formda olabilir. Optimal kontrol kuralı için geliştirdiğimiz metod lineer, lineer olmayan veya kuadratik yapıda olabilir. Bu bölümde lineer kuadratik formda yazılan optimal kontrol kuralı incelenecektir. Lineer kuadratik düzeltici (LQR) olarak ifade edilen bu kontrol metodunun matematiksel ifadesi denklem (2.121)'de görüldüğü gibidir.

$$J(\boldsymbol{x}_0) = \frac{1}{2}\boldsymbol{x}(t_f)^T \boldsymbol{H}\boldsymbol{x}(t_f) + \frac{1}{2}\int_{t_0}^{t_f} [\boldsymbol{x}(t)^T \boldsymbol{Q}(t)\boldsymbol{x}(t) + \boldsymbol{u}(t)^T \boldsymbol{R}(t)\boldsymbol{u}(t)]\, dt \quad (2.121)$$

Uygun bir aday kontrol işareti $\boldsymbol{u}^*$ denklem (2.121)'i minimum yapmaktadır. Denklem (2.121) dikkat edilirse kuadratik formdadır. Denklem (2.121) enerji kriterini minimum yapmayı hedeflemektedir ve $\boldsymbol{H}, \boldsymbol{Q}, \boldsymbol{R}$ matrisleri ağırlık matrisleri olarak adlandırılmaktadır. Denklem (2.121) performans kriterini sağlaması gereken fiziksel sistem denklem (2.122)'de görüldüğü gibi lineer zamanla değişen bir sistem olduğu varsayılsın.

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= A(t)\boldsymbol{x}(t) + B(t)\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= C(t)\boldsymbol{x}(t) + D(t)\boldsymbol{u}(t) \end{aligned} \quad (2.122)$$

Denklem (2.121)'de fiziksel sistemin durumları ve kontrol işareti $\boldsymbol{x}(t), \boldsymbol{u}(t)$ herhangi bir kısıt oluşturmamaktadır. Denklem (2.121) Hamiltonian ifadesi ile genişletip çözüm adımlarını oluşturalım. Denklem (2.123)'de Hamiltonian fonksiyonu görülmektedir.

$$H = \frac{1}{2}\boldsymbol{x}(t)^T Q(t)\boldsymbol{x}(t) + \frac{1}{2}\boldsymbol{u}(t)^T R(t)\boldsymbol{u}(t) + J_x^*(\boldsymbol{x}(t), t)^T [A(t)\boldsymbol{x}(t) + B(t)\boldsymbol{u}(t)] \quad (2.123)$$

Denklem (2.123)'ü minimum yapmak için gerekli olan $\boldsymbol{u}(t), \frac{\partial \boldsymbol{H}}{\partial \boldsymbol{u}} = 0$ denklem (2.124)'de görüldüğü gibidir.

$$\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{u}} = R(t)\boldsymbol{u}(t) + B(t)^T J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t) = 0 \tag{2.124}$$

Denklem (2.124)'de $\frac{\partial^2 \boldsymbol{H}}{\partial \boldsymbol{u}^2} = R(t) > 0$ olduğu için $\boldsymbol{u}^*(t)$ optimal çözüm denklem (2.125)'deki gibi bulunur.

$$\boldsymbol{u}^*(t) = -R^{-1}(t)B(t)^T J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t) \tag{2.125}$$

Denklem (2.125), denklem (2.123)'deki Hamilton fonksiyonunda yerine yazılarak denklem (2.126) elde edilir.

$$\begin{aligned} H = {} & \frac{1}{2}\boldsymbol{x}(t)^T Q(t)\boldsymbol{x}(t) + \frac{1}{2}J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t)^T B(t)R(t)^{-1}B(t)^T J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t) \\ & + J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t)^T A(t)\boldsymbol{x}(t) - J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t)^T B(t)R(t)^{-1}B(t)^T J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t) \\ H = {} & \frac{1}{2}\boldsymbol{x}(t)^T Q(t)\boldsymbol{x}(t) - \frac{1}{2}J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t)^T B(t)R(t)^{-1}B(t)^T J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t) + J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t)^T A(t)\boldsymbol{x}(t) \end{aligned} \tag{2.126}$$

Denklem (2.126) denklem (2.124) ile tekrar yazılarak, HJB denklem (2.127)'deki gibi ifade edilmektedir.

$$\begin{aligned} 0 = {} & J^*(\boldsymbol{x}(t), t) + \tfrac{1}{2}\boldsymbol{x}(t)^T Q(t)\boldsymbol{x}(t) - \tfrac{1}{2}J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t)^T B(t)R(t)^{-1}B(t)^T J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t) + \\ & J_{\boldsymbol{x}}^*(\boldsymbol{x}(t), t)^T A(t)\boldsymbol{x}(t) \end{aligned} \tag{2.127}$$

Denklem (2.127) ilave olarak sınır koşulu denklem (2.128)'de verildiği gibidir.

$$J_t^*(\boldsymbol{x}(t_f), t_f) = \frac{1}{2}\boldsymbol{x}(t_f)^T H \boldsymbol{x}(t_f) \tag{2.128}$$

Denklem (2.127) $J^*(\boldsymbol{x}(t), t)$ terimi için bir sınır koşulu denklem (2.128)'deki tanımlamaya benzerşekilde denklem (2.129)'da tanımlanabilir.

$$J^*(\boldsymbol{x}(t), t) = \frac{1}{2}\boldsymbol{x}(t)^T K(t)\boldsymbol{x}(t) \forall t \in K(t) > 0 \tag{2.129}$$

Elde edilen ifadeleri düzenleyip, Hamilton fonksiyonunu genişletelim. Bu durum denklem (2.130)'da görüldüğü gibidir.

$$0 = \frac{1}{2}\boldsymbol{x}(t)^T \dot{K}(t)\boldsymbol{x}(t) + \frac{1}{2}\boldsymbol{x}(t)^T Q(t)\boldsymbol{x}(t) - \frac{1}{2}\boldsymbol{x}(t)^T K(t)B(t)R(t)^{-1}B(t)^T K(t)\boldsymbol{x}(t) + \boldsymbol{x}(t)^T K(t)A(t)\boldsymbol{x}(t) \tag{2.130}$$

Denklem (2.130)'da $J_{\boldsymbol{x}}^* = K(t)x(t)$ ve $J_t^* = \frac{1}{2}\boldsymbol{x}(t)^T \dot{K}(t)\boldsymbol{x}(t)$ olmaktadır. Denklem (2.131)'de ifadelerin kompakt yazımı görülmektedir.

$$\boldsymbol{x}(t)^T K(t)A(t)\boldsymbol{x}(t) = \frac{1}{2}\boldsymbol{x}(t)^T K(t)A(t)\boldsymbol{x}(t) + \frac{1}{2}\boldsymbol{x}(t)^T K(t)A(t)\boldsymbol{x}(t) \tag{2.131}$$

Böylece (2.130) ve (2.131) ifadelerinin (2.127)'de yerlerine yazılmasıyla denklem (2.132) elde edilir.

$$0 = \frac{1}{2}\boldsymbol{x}(t)^T [\dot{K}(t) + Q(t) - K(t)B(t)R(t)^{-1}B(t)^T K(t) + K(t)A(t) + A(t)^T K(t)]\boldsymbol{x}(t) \tag{2.132}$$

Bu denklemin tüm $\boldsymbol{x}(t)$ 'ler için sağlaması gerektiğinden denklem (2.133) eşitliği elde edilir.

$$-\dot{K}(t) = Q(t) - K(t)B(t)R(t)^{-1}B(t)^{T}K(t) + K(t)A(t) + A(t)^{T}K(t) \quad (2.133)$$

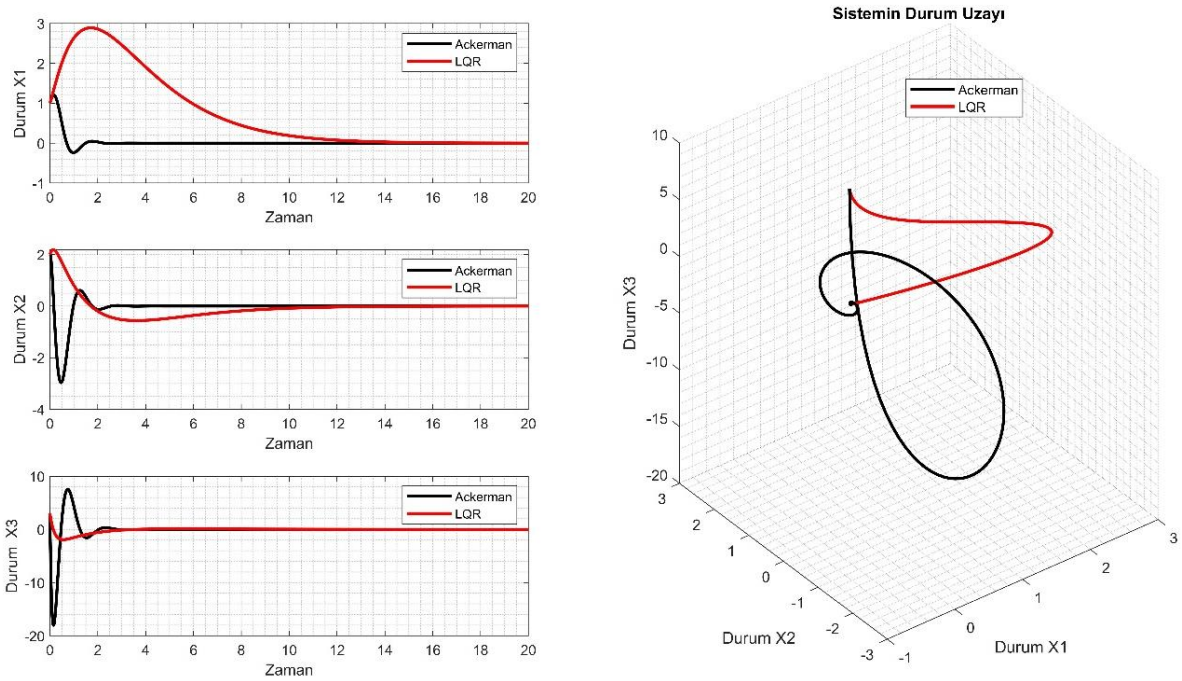
Sınır koşulu denklem (2.134)'de görülmektedir.

$$K(t_f) = H \quad (2.134)$$

Denklem (2.133) ve (2.134) ifadesi HJB denklemini Riccati diferansiyel denklemine indirgenmiş olur. Riccati diferansiyel denklemi geriye doğru integre edilerek çözümleme yapılır. Denklem (2.133) ile $K(t)$ hesaplanıp sistem için önerdiğimiz optimal kontrol kuralı $\boldsymbol{u}^{*}(t)$ denklem (2.135)'deki gibi hesaplanır.

$$\boldsymbol{u}^{*}(t) = -R(t)^{-1}B(t)^{T}K(t)\boldsymbol{x}(t) \quad (2.135)$$

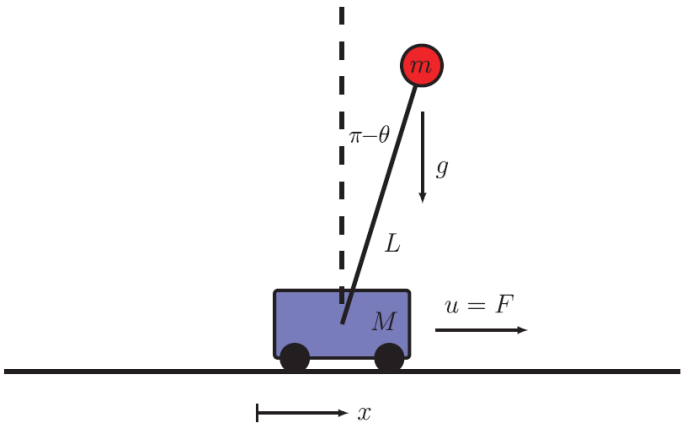
Şekil 2.25'de lineer zamanla değişmeyen sistemin $\boldsymbol{u}(t) = K\boldsymbol{x}(t)$ durum geri beslemeli kontrol kuramıyla simulasyonu görülmektedir. Sistemin zaman domenindeki karakteristikleri olan maksimum aşım, oturma zamanı, kalıcı durum hatası, yükselme zamanı vb parametrelere göre ackerman formülasyonuyla hesaplanan K kazanç matrisi ile sistemin kontrolü ve K matrisinin denklem (2.121) ve (2.122) optimal kontrol kuralına göre hesaplanan ifadesi görülmektedir. Dikkat edilecek olursa sistemin durum uzayında takip etmiş olduğu yörünge optimal kontrol kuralıyla minimum enerji kriterini sağlamaktadır. (Pavone, Marco. 2018)



Şekil 2.26: Durum Geri Beslemeli Kontrol ve Optimal Kontrol.

2.7.2 Ters Sarkacın LQR Metodu ile Optimal Kontrolü

Kontrol teorisinde kullanılan ters sarkaç sistemi lineer olmayan denklemlerle modellenmektedir. Ters sarkaç sistemi şekil…'da görüldüğü gibidir.



Şekil 2.27: Ters Sarkaç Sistemi.

Şekil 2.27' da M arabanın kütlesini, m sarkacın kütlesini, g yerçekimi ivmesini, L sarkacın uzunluğunu, $u = F$arabaya uygulanan eksenel kuvveti belirtmektedir. Ters sarkaç probleminde arabaya uygulanan giriş işareti ile sarkacı $\theta = \pi$ noktasında dengede tutmak hedeflenmektedir. Şekil 2.27'da sistemin iki tane denge noktası mevcuttur. $\theta = 0$ ve $\theta = \pi$ olmak üzere. Sistemin denge noktalarında $\omega = v = 0$ açısal ve çizgisel hızı sıfır olmata ve arabanın pozisyonu x serbest değişken olmaktadır. Ters sarkaç sisteminin Euler-Lagrange ifadesi ile durum uzay denklemleri türetilebilir. Sistemin durum uzay modeli denklem (2.136)'da görülmektedir.

$$\dot{x} = v$$
$$\dot{v} = \frac{-m^2L^2g\,cos(\theta)\,sin(\theta) + mL^2(mL\omega^2\,sin(\theta) - \delta v) + mL^2u}{mL^2(M + m(1 - cos(\theta)^2))}$$
$$\dot{\theta} = \omega$$
$$\dot{\omega} = \frac{(m + M)mgL\,sin(\theta) - mL\,cos(\theta)(mL\omega^2\,sin(\theta) - \delta v) + mL\,cos(\theta)u}{mL^2(M + m(1 - cos(\theta)^2))} \quad (2.136)$$

Denklem (2.136)'da bulunan terimler tablo 2.5'da görüldüğü gibidir.

Tablo 2.5: Ters Sarkaç Sisteminin Parametreleri.

M	Arabanın kütlesi
m	Sarkacın kütlesi
L	Sarkacın uzunluğu
δ	Sürtünme katsayısı
g	Yerçekimi ivmesi
x, v	Sırasıyla arabanın çizgisel konumu ve çizgisel hızı
θ, ω	Sırasıyla sarkacın açısal konumu ve açısal hızı
u	Kontrol işareti

Denklem (2.136) görülen sistem denklemleri lineer olmayan bir yapıdadır. Lineer olmayan sistemler lineer kontrolcülerle kontrol edilememektedir. Sistem için tasarlanacak olan optimal kontrol kuramı LQR, sistemin $\theta = \pi$ noktası civarındaki küçük açı değişimlerine göre lineerleştirilip tasarlanacaktır. Böylece küçük açı değişimleri için $sin(\theta) = 0$, $cos(\theta) = 1$ kabul edilerek denklem (2.136)'nın lineerleştirilmiş durum uzay formunda denklemi (2.137)'deki gibidir.

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{\delta}{M} & b\frac{mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -b\frac{\delta}{ML} & -b\frac{(m+M)g}{ML} & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{b}{ML} \end{bmatrix} u \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x \\ v \\ \theta \\ \omega \end{bmatrix} \tag{2.137}$$

Denklem (2.137)'de sistem matrisleri olan (A,B) ile oluşturulan kontrol edilebilirlik matrisi denklem (2.138)'de görüldüğü gibidir.

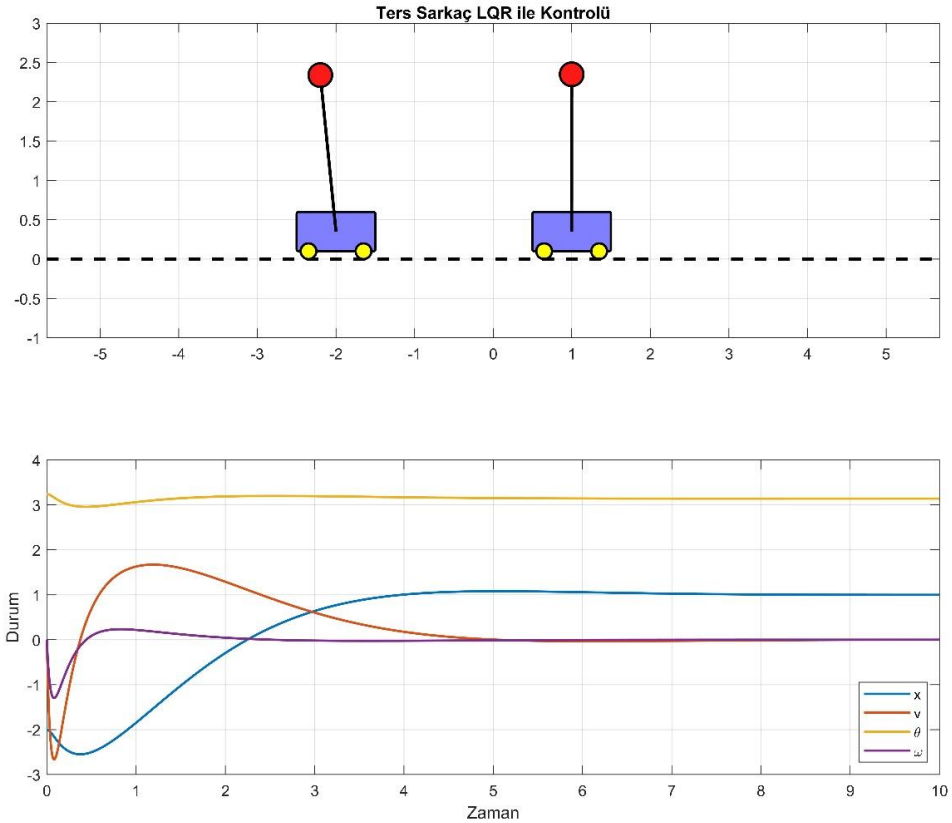
$$C = [A \quad AB \quad A^2B \quad A^3B] \tag{2.138}$$

Denklem (2.138) $rank(C) = 4$ olduğu için sistem tüm durum kontrol edilebilirdir. Tablo 2.6'da ters sarkaç sisteminin kontrol algoritması görülmektedir.

Tablo 2.6: Ters Sarkaç Sisteminin LQR Kontrol Algoritması.

ALGORİTMA : Ters Sarkaç Sistemi LQR Kontrolcü Tasarımı			
1	**:**	M=5, m=1, L=2, g=-9.81, d=1, b=1	*Sistem parametrelerini tanımla*
2	**:**	$\dot{\boldsymbol{x}} = A\boldsymbol{x} + Bu$	*Sistemin lineerleştirilmiş durum uzay modeli*
3	**:**	Q,R	*Matrislerini tanımla*
4	**:**	K = LQR(A,B,Q,R)	*Optimal kontrol kuralına göre K kazanç matrisini hesapla*
5	**:**	$\boldsymbol{x}(0) = [-2 \quad 0 \quad \pi + 0.1 \quad 0]^T$	*Sistemin başlangıç durumlarını belirle*
6	**:**	$r = [1 \quad 0 \quad \pi \quad 0]^T$	*Sistemin referans işaretini oluştur*
7	**:**	$u = -K(x - r)$	*Sistemin kontrol işaretini oluştur*
8	**:**	$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, u)$	*Sistemin lineer olmayan durum uzay denklemlerini Runge Kutta ile çözüm*

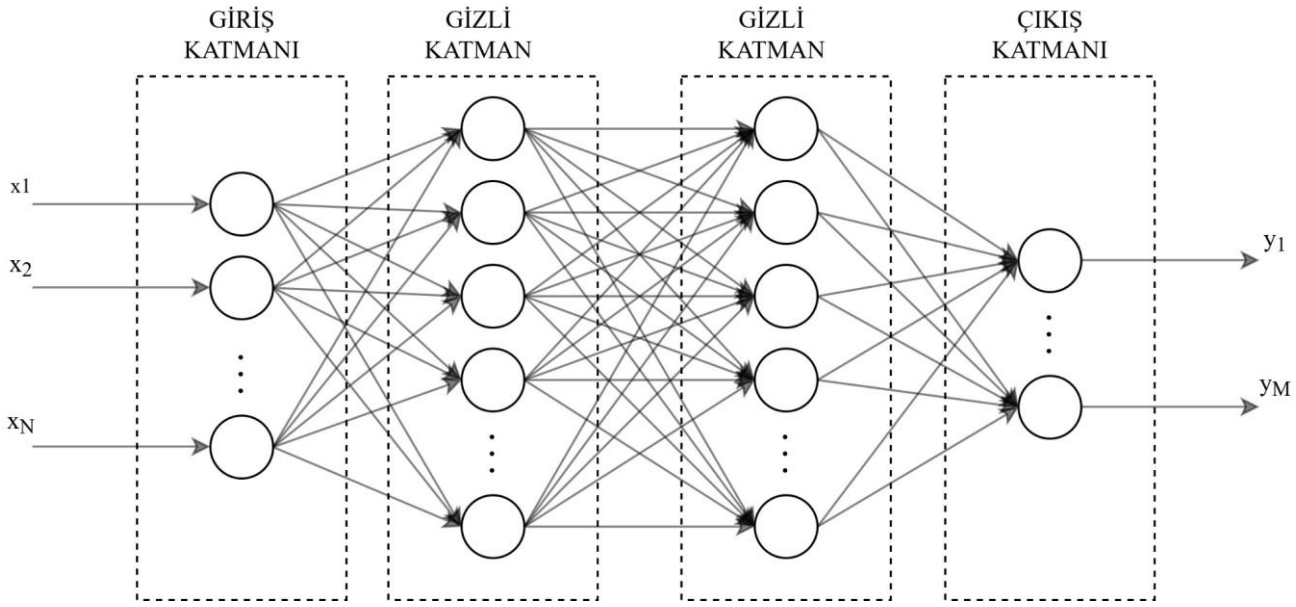
Tablo 2.6'da önerilen algoritmanın matlab sonuçları şekil 2.27'de görülmektedir. Sistemin başlangıç durumlarından $[x \quad v \quad \theta \quad \omega] = [-2 \quad 0 \quad \pi + 0.1 \quad 0]$, hedef durum $[x \quad v \quad \theta \quad \omega] = [1 \quad 0 \quad \pi \quad 0]$'a optimal kontrol işaretiyle varmaktadır. (Siradjuddin, Indrazno. ve diğ. 2017)



Şekil 2.28: Ters Sarkaç Sisteminin Optimal Kontrolü.

3. YAPAY SİNİR AĞLARI

Yapay zekanın alt kolu olan makine öğrenmesi günümüzde her alanda kullanılmaktadır. Makine öğrenmesinin gelişmesi, oluşturulan algoritmaların gelişmesiyle mümkündür. Makine öğrenmesi algoritması olan yapay sinir ağları insan zihnindeki nöronların basit matematiksel ifadesi olarak tasarlanmıştır. Yapay sinir ağları günümüzde birçok problem çözmek için kullanılmaktadır. Klasik yapay sinir ağının topolojisi şekil 3.1'de görüldüğü gibidir. (Hagan, Martin T. ve diğ. 2014)



Şekil 3.1: Klasik Yapay Sinir Ağı.

Yapay sinir ağlarında şekil 3.1'de görüldüğü gibi üç temel katman bulunmaktadır. Giriş katmanı, sinir ağına giren verileri ifade etmektedir. Gizli katman nöronların bulunduğu katmandır ve çıkış katmanı sinir ağının çıkışını ifade etmektedir. Yapay sinir ağlarında gizli katman sadece bir katmandan oluşuyorsa bu tip sinir ağı, sığ (shallow) sinir ağı olarak isimlendirilmektedir. Yapay sinir ağlarında gizli katmanda birden fazla katmanın bulunması durumunda derin (deep) yapay sinir ağı olarak isimlendirilmektedir. Bu bölümde tek katmanlı yapay sinir ağları ve çok katmanlı yapay sinir ağları incelenecektir. Çok katmanlı yapay sinir ağı bu tez çalışmasında uygulanan pekiştirmeli öğrenme algoritmalarında kullanılacaktır. Yapay sinir ağlarında modelin eğitilmesi için optimizasyon algoritması kullanılması gerekmektedir. Optimizasyon en iyilemenin bilimi olarak karşımıza çıkmaktadır. Yapay sinir ağlarının eğitimini oluşturmak için birçok optimizasyon algoritması kullanılmaktadır.

Sayısal optimizasyon teknikleri, sezgisel optimizasyon teknikleri, olasılıksal optimizasyon teknikleri yapay sinir ağlarının eğitilmesi için kullanılmaktadır. Yapay sinir ağlarını oluştururken bu tez kapsamında lineer olmayan kısıtsız sayısal optimizasyon incelenecektir.

3.1 Lineer Olmayan Kısıtsız Optimizasyon

Optimizasyon teknikleri günümüzde her alanda uygulanmaktadır. Optimizasyon en iyilemenin bilimidir. Optimizasyon teknikleri günümüzde matematiksel programlama, sezgisel optimizasyon, sayısal optimizasyon vb tekniklerle uygulama alanı bulmaktadır. Bu bölümde genel optimizasyon probleminin tanımı yapılacak sonrasında çok boyutlu kısıtsız optimizasyon problemleri için çözüm metodları incelenecektir. En genel halde bir optimizasyon problemi denklem (3.1)'de görüldüğü gibidir. (İplikçi, Serdar. 2017)

$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ &g_i(\boldsymbol{x}) = 0 i \in E \\ &g_i(\boldsymbol{x}) \leq 0 i \in J \end{aligned} \tag{3.1}$$

Denklem 3.1'de $f(\boldsymbol{x})$ optimize etmek istediğimiz amaç fonksiyonu / maaliyet fonksiyonu / kayıp fonksiyonu olarak isimlendirilmektedir. $\boldsymbol{x}$ tasarım değişkenlerimizi ifade etmektedir ve $\boldsymbol{x} \in \Re^n$ olmak üzere $f(\boldsymbol{x}): \Re^n \rightarrow \Re$ tanımlıdır. $g_i(\boldsymbol{x}) = 0$ tasarım değişkenlerinin eşitlik kısıtlarını ifade etmekte olup, E eşitlik kısıtları kümesini belirtmektedir. $g_i(\boldsymbol{x}) \leq 0$ tasarım değişkenlerinin eşitsizlik kısıtı olup, J eşitsizlik kısıtları kümesini belirtmektedir. Denklem (3.1) ile ifade edilen optimizasyon problemi, tasarım değişkenlerinde kısıtlama kümesi olduğundan ve amaç fonksiyonumuzun tasarım değişkenlerine göre lineer olmayan bir fonksiyon olmasından dolayı lineer olmayan kısıtlı optimizasyon problemi olarak isimlendirilmektedir. Bu bölümde denklem (3.2) ile ifade edilen kısıtsız lineer olmayan çok değişkenli optimizasyon problemleri için çözüm metodları incelenecektir.

$$\min_{\boldsymbol{x}} \ f(\boldsymbol{x}) \tag{3.2}$$

Denklem (3.2)'de $f(\boldsymbol{x}): \Re^n \rightarrow \Re$ olmakla birlikte sürekli ve türevlenebilir olmalıdır.

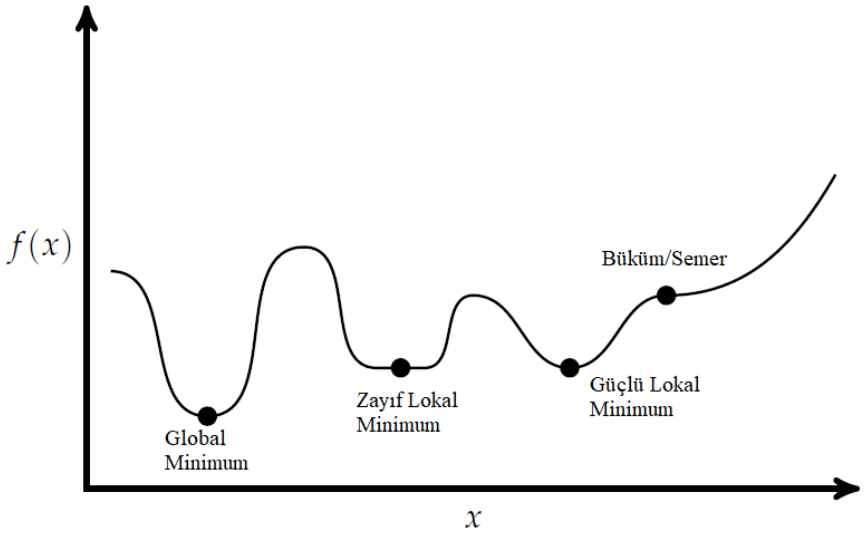
$f(\boldsymbol{x})$ fonksiyonu için optimum değeri ifade eden aday bir $\boldsymbol{x}^*$ değeri için $e > 0$ ifadesi için denklem (3.3) sağlanıyorsa bu durumda aday $\boldsymbol{x}^*$ noktası yerel minimum değerini ifade etmektedir.

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \parallel \boldsymbol{x} - \boldsymbol{x}^* \parallel < e \tag{3.3}$$

Bulunan $\boldsymbol{x}^*$ aday nokta için denklem (3.4) sağlanıyorsa global minimum olarak ifade edilmektedir.

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \in \Re^n \tag{3.4}$$

Denklem (3.3) ve (3.4) ifadelerindeki eşitsizlik değeri ($\leq \rightarrow <$) olması durumunda kesin yerel ve kesin global minimum olarak ifade edilir. $f(\boldsymbol{x})$ optimize etmek istediğimiz fonksiyon konveks olmayan biçimde olabilir ve fonksiyonda birden fazla yerel minimum buluna bilir. Bu durum şekil 3.2'de görüldüğü gibidir.



Şekil 3.2: Konveks olmayan amaç fonksiyonu ve kritik noktalar.

3.1.1 Optimallik İçin Gerekli Koşullar

Denklem (3.2)'nin optimum noktasını bulmak için $f(\boldsymbol{x})$ fonksiyonunun $\Delta\boldsymbol{x}$ değişimiyle azaldığını incelemek gerekir ve fonksiyonun bu noktada nasıl davrandığının analiz edilmesi gerekmektedir. Denklem (3.2)'de fonksiyon $\boldsymbol{x}^*$ noktasında $f(\boldsymbol{x}^*)$ değeri almaktayken $\boldsymbol{x}^* + \Delta\boldsymbol{x}^*$ noktasında hangi değeri aldığı $f(\boldsymbol{x}^* + \Delta\boldsymbol{x}^*)$ noktasında Taylor açılımıyla incelenmektedir. Denklem (3.2)'nin birinci dereceden taylor açılımı denklem (3.5)'deki gibidir. (İplikçi, Serdar. 2017)

$$f(\boldsymbol{x}^* + \Delta\boldsymbol{x}) = f(\boldsymbol{x}^*) + \nabla f(\boldsymbol{x}^*)'\Delta\boldsymbol{x} + HOT \tag{3.5}$$

Denklem (3.5)'de HOT Taylor açılımında yüksek dereceden terimleri ifade etmektedir. Denklem (3.5)'de $HOT = 0$ alınırsa denklem (3.6) elde edilir.

$$f(\boldsymbol{x}^* + \Delta\boldsymbol{x}) - f(\boldsymbol{x}^*) \approx \nabla f(\boldsymbol{x}^*)'\Delta\boldsymbol{x} \tag{3.6}$$

Denklem (3.6)'ya benzer şekilde Taylor açılımda ikinci dereceden terimleri açılıma dahil edilerek yaklaşık olarak denklem (3.7) elde edilir.

$$f(\boldsymbol{x}^* + \Delta\boldsymbol{x}) - f(\boldsymbol{x}^*) \approx \nabla f(\boldsymbol{x}^*)'\Delta\boldsymbol{x} + \frac{1}{2}\Delta\boldsymbol{x}'\nabla^2 f(\boldsymbol{x}^*)\Delta\boldsymbol{x} \tag{3.7}$$

Optimallik koşulları için $\boldsymbol{x}^*$noktası kısıtsız yerel minimum nokta olması durumunda birinci türev bilgisindeki $\Delta\boldsymbol{x}$değişiminin pozitif olması gerekmektedir. Bu durum denklem (3.8)'de görüldüğü gibidir.

$$\nabla f(\boldsymbol{x}^*)'\Delta\boldsymbol{x} = \sum_{i=1}^{n} \frac{\partial f(x^*)}{\partial x_i}\Delta x_i \geq 0 \tag{3.8}$$

Benzer şekilde denklem (3.8) optimallik için birinci gerek şart, denklem (3.9)'daki gibidir.

$$\nabla f(\boldsymbol{x}^*) = 0 \tag{3.9}$$

Denklem (3.9)'da bulunan $\boldsymbol{x}^*$ noktası durağan nokta veya kritik nokta olarak isimlendirilmektedir. Bu noktada fonksiyonun değişmediği varsayılmaktadır. Bu durum optimallik için birinci dereceden gerek koşulu ifade etmektedir. Benzer şekilde ikinci dereceden koşul icin denklem (3.7) incelenerek, denklem (3.10) elde edilir.

$$\nabla f(\boldsymbol{x}^*)'\Delta\boldsymbol{x} + \frac{1}{2}\Delta\boldsymbol{x}'\nabla^2 f(\boldsymbol{x}^*)\Delta\boldsymbol{x} \geq 0 \tag{3.7}$$

Denklem (3.9) ve (3.10) birleştirilerek denklem (3.8) elde edilir.

$$\Delta\boldsymbol{x}'\nabla^2 f(\boldsymbol{x}^*)\Delta\boldsymbol{x} \geq 0 \tag{3.10}$$

Denklem (3.10)'da $\forall\boldsymbol{x}$ değeri için eşitlik sağlanacağı için, eşitliğin oluşması için ikinci dereceden koşul denklem (3.11)'deki gibi elde edilir.

$$\nabla^2 f(\boldsymbol{x}^*) \geq 0 \tag{3.11}$$

Denklem (3.11)'de ∇^2 ifadesi amaç fonksiyonunun hessian matrisini oluşturmaktadır ve tanım gereği pozitif tanımlı olmak zorundadır. $f(.)\colon \Re^n \to \Re$ olmak üzere amaç fonksiyonunun hessian ve gradyan matrisi denklem (3.12)'deki gibi hesaplanmaktadır.

$$\nabla^2 f(\boldsymbol{x}) = H = \begin{bmatrix} \frac{\partial^2 f}{\partial {x_1}^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial {x_1}^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial {x_n}^2} \end{bmatrix}$$

$$\nabla f(\boldsymbol{x}) = G = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \tag{3.12}$$

Denklem (3.9)'da bulunan $\boldsymbol{x}^*$ kritik noktanın yorumu denklem (3.11) ile yapılmalıdır. Böylece uygun hessian matrisinin durumuna göre $\boldsymbol{x}^*$ noktasının durumu değişmektedir. Bu durum tablo 3.1'de görüldüğü gibidir.

Tablo 3.1: Hessian matrisi ve kritik nokta ilişkisi.

$\nabla^2 f(\boldsymbol{x}^*) \geq 0$	$\boldsymbol{x}^*$ noktası yerel minimum
$\nabla^2 f(\boldsymbol{x}^*) > 0$	$\boldsymbol{x}^*$ noktası kesin yerel minimum
$\nabla^2 f(\boldsymbol{x}^*) \leq 0$	$\boldsymbol{x}^*$ noktası yerel maksimum
$\nabla^2 f(\boldsymbol{x}^*) < 0$	$\boldsymbol{x}^*$noktası kesin yerel maksimum
$\nabla^2 f(\boldsymbol{x}^*) = 0$	$\boldsymbol{x}^*$semer noktası

3.1.2 Gradyan Yöntemler

Denklem (3.2) amaç fonksiyonunu minimum yapan $\boldsymbol{x}^*$ noktasını bulmak için $f(\boldsymbol{x})$ fonksiyonunun birinci türev olan $\nabla f(\boldsymbol{x})$ gradyan vektörü ve ikinci türev olan $\nabla^2 f(\boldsymbol{x})$ hessian matrisini kullanmaktadır. Belirli bir $\boldsymbol{x}_0$ başlangıç noktasından $\boldsymbol{x}^*$noktasının bulunması için iteratif olarak parametrelerin güncellenmesi gerekmektedir. Bu durum denklem (3.13)'de görüldüğü gibidir.

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + s_i \boldsymbol{p}_i \tag{3.13}$$

Denklem (3.13)'de $\boldsymbol{p}_i$iniş yönünü ifade etmekte olup birinci dereceden yöntemler için fonksiyonun gradyan vektörünün tersine eşittir bu durum denklem (3.14)'de görülmektedir.

$$\boldsymbol{p}_i = -\nabla f(\boldsymbol{x}_i) \tag{3.14}$$

Denklem (3.13)'de s_i adım aralığını ifade etmektedir. Böylece denklem (3.13) 'deki güncelleme kuralı ile denklem (3.2)'yi minimum yapan noktalar belirlenebilir. Denklem (3.13) ifadesindeki $\boldsymbol{p}_i$ amaç fonksiyonunun, birinci dereceden türev bilgisini veya ikinci dereceden türev bilgisini kullanmaktadır. (İplikçi, Serdar. 2017)

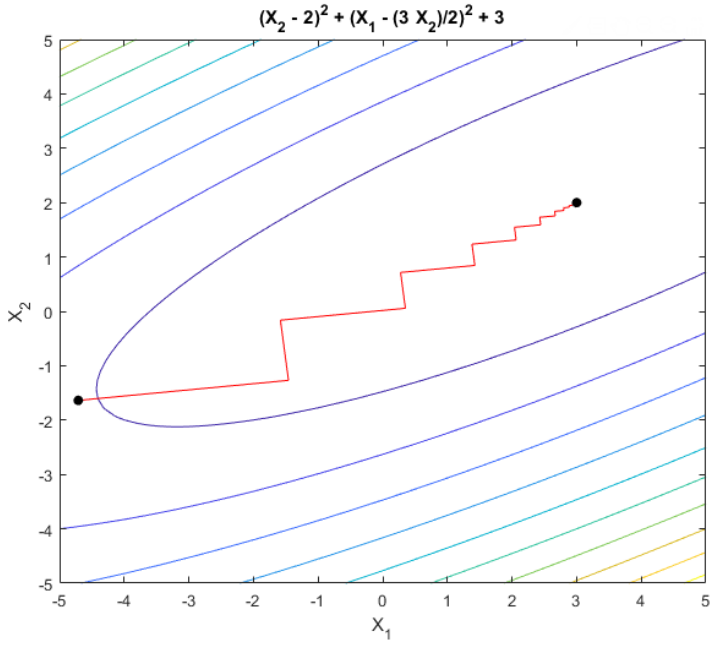
3.1.3 Gradyan Azalan Algoritması

Gradyan azalan algoritması tablo 3.2'de görüldüğü gibidir. Algoritma optimum parametre değerlerini $\boldsymbol{x}^*$ bulmak için amaç fonksiyonunun gradyan vektörünü kullanmaktadır. (Jorge, Nocedal. ve diğ 2006)

Tablo 3.2: Gradyan Azalan Algoritması.

ALGORİTMA : Gradyan Azalan Algoritması		
1	:	$\boldsymbol{x}_0, s$ *Parametrelerin başlangıç noktasını belirle*
2	:	$i \leftarrow 0$
3	:	$\boldsymbol{x}_i$ *noktasında* $\nabla f(\boldsymbol{x}_i)$ *hesapla*
4	:	$\boldsymbol{p}_i = -\nabla f(\boldsymbol{x}_i)$ *İlerleme yönü belirle*
5	:	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + s\boldsymbol{p}_i$ *Parametreleri güncelle*
6	:	$i \leftarrow i + 1$
7	:	*Eğer* $\| \nabla f(\boldsymbol{x}_i) \| < 1e-6$ *Algoritmayı sonlandır*
8	:	*Değilse 3. adıma geri dön*

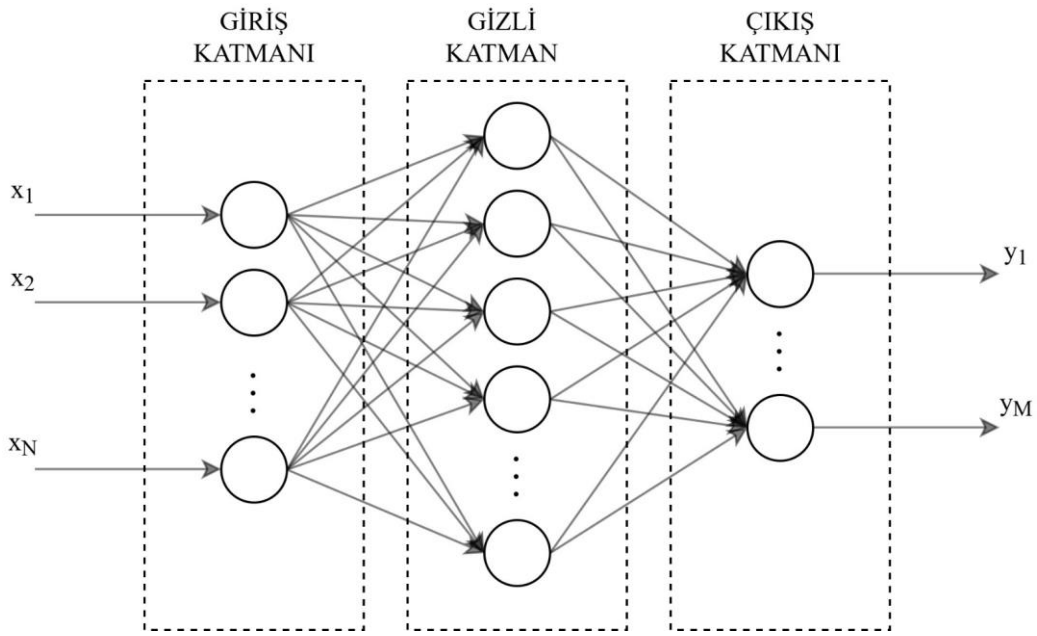
Gradyan azalan algoritmasının matlab uygulaması şekil 3.3'de görülmektedir.



Şekil 3.3: Gradyan Azalan Algoritmasıyla Optimizasyon.

3.2 Tek Katmanlı Yapay Sinir Ağları

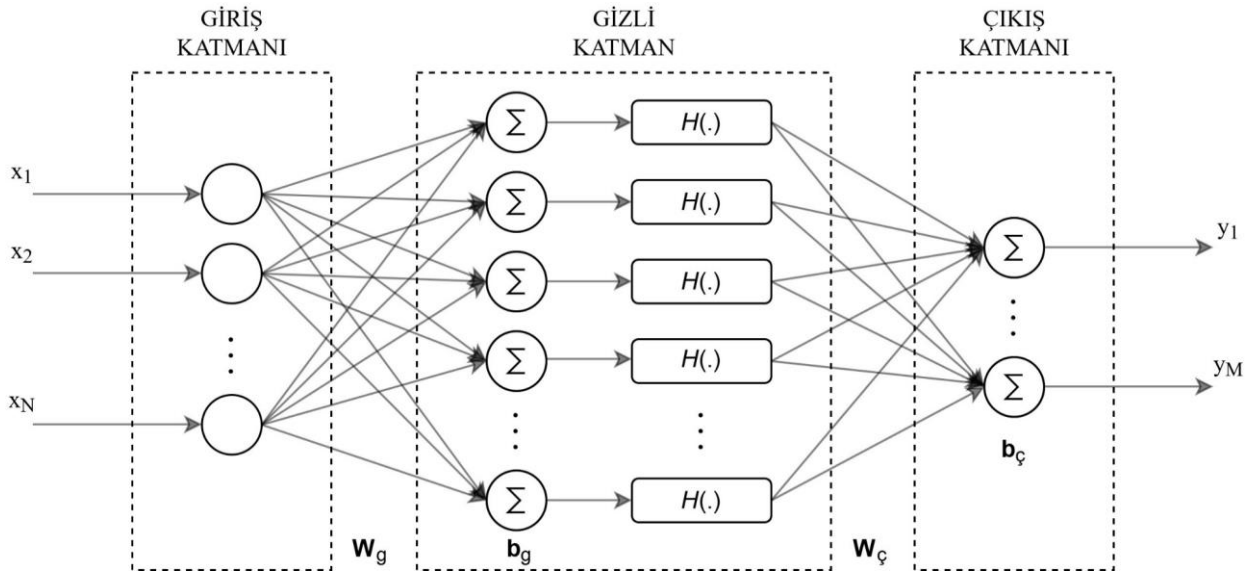
Çok girişli çok çıkışlı (MİMO) yapay sinir ağları makine öğrenmesi algoritması olup, danışmanlı öğrenme algoritmaları sınıfında yer almaktadır. Danışmanlı öğrenme yönteminde $\{X_i, Y_i\} i = 1,2,3,\ldots,N$ veri setimiz bulunmaktadır. Çok girişli yapay sinir ağında giriş uzayının boyutu giriş verisi olan $X \in \Re^N$ ile belirlenmektedir. Benzer şekilde çıkış uzayının boyutu $Y \in \Re^M$ olmaktadır. Yapay sinir ağı giriş verisiyle çıkış verisi arasındaki ilişkiyi parametrelerini güncelleyerek öğrenmeye çalışır. Çok girişli çok çıkışlı yapay sinir ağı bir giriş katmanı, bir çıkış katmanı birde gizli katman barındırmaktadır. Sinir ağının topolojisi şekil 3.4'de görülmektedir. (Goodfellow, Ian. ve diğ. 2017)



Şekil 3.4: Çok Girişli-Çıkışlı Tek Katmanlı Yapay Sinir Ağı.

3.2.1 Tek Katmanlı Yapay Sinir Ağı İleri Modeli

Tek katmanlı çok giriş çok çıkış yapay sinir ağının ileri modeli şekil 3.5'de görülmektedir. (Goodfellow, Ian. ve diğ. 2017)



Şekil 3.5: Tek Katmanlı Çok Giriş Çok Çıkış Yapay Sinir Ağı.

Tek katmanlı çok giriş çok çıkışlı yapay sinir ağının ileri modeli matematiksel olarak denklem (3.15)'de görülmektedir.

$$\hat{\boldsymbol{y}}_j = \boldsymbol{W}_{ç} H\left(\boldsymbol{W}_g \boldsymbol{x}_i + \boldsymbol{b}_g\right) + \boldsymbol{b}_{ç} \qquad i = 1,2,\ldots,Nj = 1,2,\ldots,M \tag{3.15}$$

Denklem (3.15) ile yapay sinir ağının parametrelerine bağlı olarak uygulanan giriş ile çıkış üretilmektedir. Denklem (3.15)'de $H(.)$ aktivasyon fonksiyonu olarak isimlendirilmektedir. Aktivasyon fonksiyonuna örnek olarak denklem (3.16) verilebilir. Literatürde yapay sinir ağları için önerilmiş birçok aktivasyon fonksiyonu bulunmaktadır. Bu tez kapsamında denklem (3.16)'daki aktivasyon fonksiyonu kullanılmıştır.

$$H(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.16}$$

3.2.2 Tek Katmanlı Yapay Sinir Ağı Eğitim

Denklem (3.15) ile önerilen yapay sinir ağının eğitm aşaması, ağın parametreleri olan $\boldsymbol{W}_g, \boldsymbol{b}_g, \boldsymbol{W}_ç, \boldsymbol{b}_ç$ bulunması problemidir. Yapay sinir ağlarında model parametreleri rastgele bir değerden başlayarak, iteratif olarak güncellenir. Bu güncelleme, yapay sinir ağları en iyi modeli elde edene kadar model parametreleri güncellenmeye devam etmektedir. Güncelleme işlemi tablo 3.2'de önerilen gradyan azalan algoritmasıyla yapılmaktadır. Elimizde var olan veri seti $T_i, Y_i i = 1,2,3,\ldots,N$ adet veri ve bunlara karşılık yapay sinir ağlarının çıkış değeri tablo 3.3'de verildiği gibi olsun.

Tablo 3.3: Danışmanlı Öğrenme Giriş Çıkış Veri Seti.

MIMO	**Giriş Verisi**				**Çıkış Verisi**				**Model Çıkışı**			
i	$T_i \in \Re^R$				$Y_i \in \Re^M$				$\hat{Y}_i \in \Re^M$			
1	X_{11}	X_{12}	...	X_{1R}	Y_{11}	Y_{12}	...	Y_{1M}	$\hat{Y}_{11}$	$\hat{Y}_{12}$	...	$\hat{Y}_{1M}$
2	X_{21}	X_{22}	...	X_{2R}	Y_{21}	Y_{22}	...	Y_{2M}	$\hat{Y}_{21}$	$\hat{Y}_{22}$	...	$\hat{Y}_{2M}$
3	X_{31}	X_{32}	...	X_{3R}	Y_{31}	Y_{32}	...	Y_{3M}	$\hat{Y}_{31}$	$\hat{Y}_{32}$	...	$\hat{Y}_{3M}$
⋮	⋮				⋮				⋮			
N	X_{N1}	X_{N2}	...	X_{NR}	Y_{N1}	Y_{N2}	...	Y_{NM}	$\hat{Y}_{N1}$	$\hat{Y}_{N2}$	...	$\hat{Y}_{NM}$

Elimizdeki veriler $T_i, Y_i i = 1,2,3,\ldots,N$ olsun. Çıkış verilerimiz $Y_i \in \{-1,1\}$ ise sınıflandırma problemi, çıkış verilerimiz $Y_i \in \Re^M$ olduğunda regresyon problemi olarak isimlendirilmektedir. Mevcut veri setimizi eğitim, test ve doğrulama olarak üç parçaya bölmemiz gerekmektedir. Örnek olarak %70 eğitim %20 test ve %10 doğrulama olmak üzere elimizdeki $T_i, Y_i i = 1,2,3,\ldots,N$ veri kümesinden uniform örnekleme yapmamız gerekmektedir. Eğitim veri setiyle yapay sinir ağlarını eğitmemiz gerekmektedir. Eğitim işlemi ağın optimum $\boldsymbol{W}_g, \boldsymbol{b}_g, \boldsymbol{W}_ç, \boldsymbol{b}_ç$ parametrelerini bulma problemidir. Tablo 3.2'de bulunan gradyan azalan algoritması ile, oluşturulan amaç fonksiyonunun parametrelerine göre minimum yapan değerini bulabiliriz. Veri setimizdeki eğitim verisiyle optimizasyon problemimizi çözüp, test verisiyle geliştirdiğimiz yapay modeli test edeceğiz, böylece en iyi modeli belirlemiş olacağız. Kalan veri setimiz olan doğrulama verileriyle, modelimizin doğrulamasını gerçekleştireceğiz. Yapay sinir ağları için denklem (3.17)'de maaliyet fonksiyonu görülmektedir.

$$\frac{1}{n}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \tag{3.17}$$

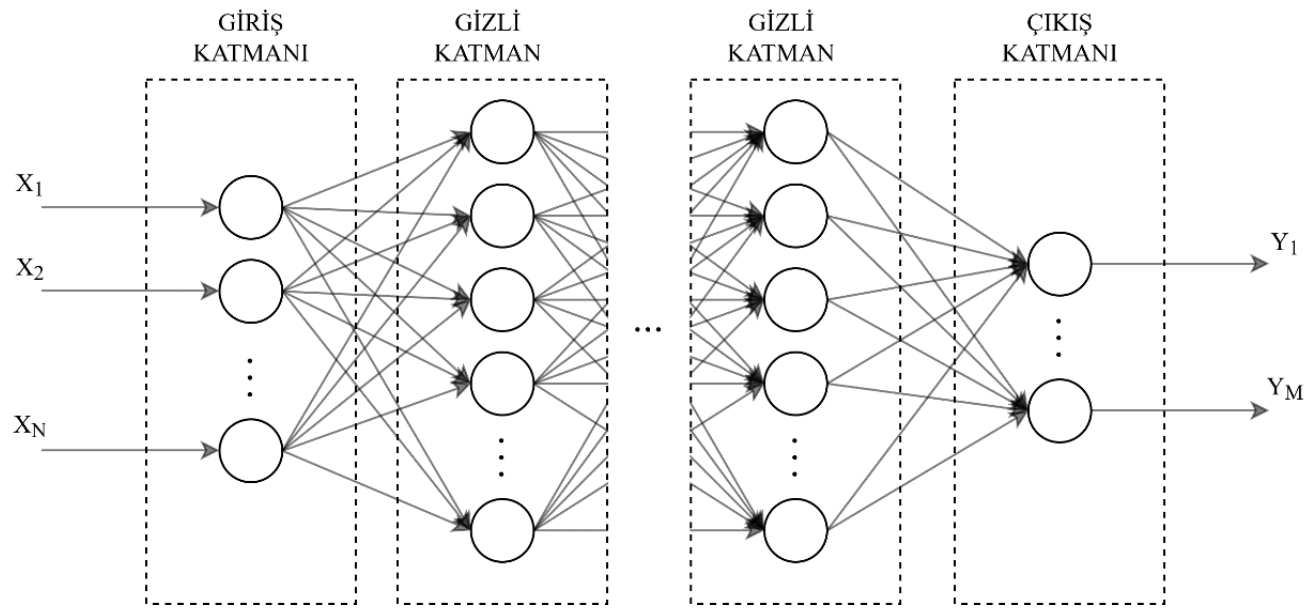
Denklem (3.17)'de önerilen maaliyet fonksiyonunun minimum değerinin bulunması için optimizasyon gerekmektedir. Denklem (3.15)'de yapay sinir ağının ileri modeli görülmektedir. Tablo 3.2'de önerilen gradyan azalan algoritmasıyla model parametreleri optimize edilir. Tablo 3.4'de yapay sinir ağlarının eğitimi için gradyan azalan algoritması görülmektedir. (Goodfellow, Ian. ve diğ. 2017)

Tablo 3.4: Yapay Sinir Ağlarının Gradyan Azalan Algoritmasıyla Eğitimi.

ALGORİTMA : Yapay Sinir Ağlarının Gradyan Azalan Algoritmasıyla Eğitimi		
1	**:**	$\boldsymbol{W}_g, \boldsymbol{b}_g, \boldsymbol{W}_ç, \boldsymbol{b}_ç, s$ *Parametrelerin başlangıç değerlerini belirle*
2	**:**	$T_i, Y_i i = 1,2,3,\ldots,N$ *Veri setini üç parçaya böl*
3	**:**	$i \leftarrow 0$
4	**:**	$\hat{Y}_j = \boldsymbol{W}_ç H(\boldsymbol{W}_g T_i + \boldsymbol{b}_g) + \boldsymbol{b}_ç$ *Yapay sinir ağının eğitim verisiyle çıkışını hesapla*
5	**:**	$\boldsymbol{e}_i = \frac{1}{n}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$ *Yapılan hatayı hesapla*
6	**:**	$\boldsymbol{p}_i = -\nabla\hat{Y}(\boldsymbol{W}_g, \boldsymbol{b}_g, \boldsymbol{W}_ç, \boldsymbol{b}_ç)$ *Parametrelere göre gradyan vektörünü hesapla*
7	**:**	$\boldsymbol{W}_g, \boldsymbol{b}_g, \boldsymbol{W}_ç, \boldsymbol{b}_ç(i+1) = \boldsymbol{W}_g, \boldsymbol{b}_g, \boldsymbol{W}_ç, \boldsymbol{b}_ç(i) + \boldsymbol{p}_i \boldsymbol{e}_i$ *Parametreleri güncelle*
8	**:**	*Eğer* $\parallel \boldsymbol{e}_i \parallel < 1e-6$ *Algoritmayı sonlandır*
9	**:**	*Değilse 3. adıma geri dön*

3.3 Çok Katmanlı Yapay Sinir Ağı

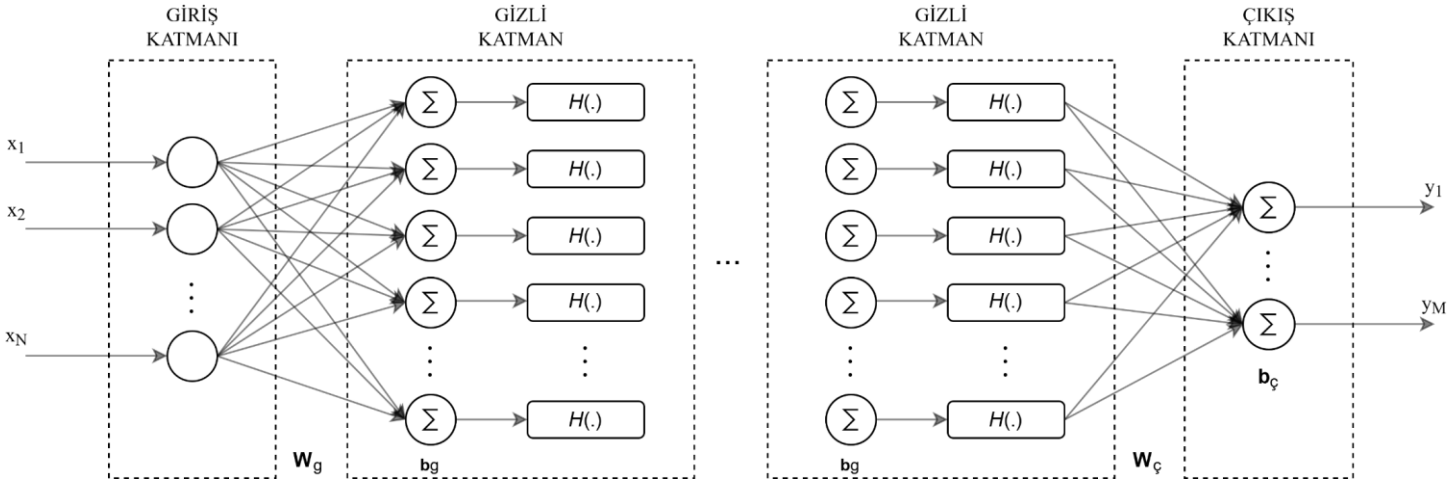
Çok katmanlı yapay sinir ağları, tek katmanlı yapay sinir ağları ile benzerlik göstermektedir. Tek katmanlı yapay sinir ağlarında gizli katman bir adet olmasına karşılık çok katmanlı yapay sinir ağlarında gizli katman uygulamaya ve problemin zorluğuna göre eklenmektedir. Yapay sinir ağları, literatürde evrensel fonksiyon yaklaşıklayıcı olarak ifade edilmektedir. Bu tez çalışmasında pekiştirmeli öğrenme algoritması, DDPG algoritması için çok katmanlı yapay sinir ağları kullanılmıştır. Çok katmanlı yapay sinir ağının topolojik yapısı şekil 3.6'da görülmektedir. Şekil 3.6'da görülen çok katmanlı yapay sinir ağında her bir katman için veri setinin zorluğuna göre nöron eklenmektedir.



Şekil 3.6: Çok Katmanlı Yapay Sinir Ağı.

3.3.1 Çok Katmanlı Yapay Sinir Ağının İleri Modeli

Çok katmanlı çok girişli çok çıkışlı yapay sinir ağının ileri modeli şekil 3.7'de görüldüğü gibidir. (Goodfellow, Ian. ve diğ. 2017)



Şekil 3.7: Çok Katmanlı Yapay Sinir Ağı İleri Modeli.

Şekil 3.7'de görülen çok katmanlı yapay sinir ağının ileri modelinin matematiksel ifadesi denklem (3.18)'de görüldüğü gibidir.

$$\widehat{\boldsymbol{Y}}_j = \boldsymbol{W}_{ç} H(\boldsymbol{W}_{gk} \ldots H(\boldsymbol{W}_{g2} H(\boldsymbol{W}_{g1} \boldsymbol{X}_i + \boldsymbol{b}_{g1}) + \boldsymbol{b}_{g2}) \ldots + \boldsymbol{b}_{gk}) + \boldsymbol{b}_{ç}$$
$$i = 1,2,\ldots,Nj = 1,2,\ldots,Mk = 1,2,\ldots,K \qquad (3.18)$$

Denklem (3.18)'de K adet gizli katman, $X \in \Re^N$ ve $Y \in \Re^M$ olmaktadır. Çok katmanlı yapay sinir ağlarının optimizasyon problemi tablo 3.3'de önerilen algoritmayla çözülmektedir. Tablo 3.3'de önerilen gradyan azalan algoritmasının geliştirilmiş hali olan ADAM algoritması, bu tez çalışmasında kullanılmaktadır.

3.3.2 Yapay Sinir Ağının ADAM Algoritması ile Eğitimi

Adam algoritması stokastik tanımlanan amaç fonksiyonu ve fonksiyonun gradyanı üzerinden geliştirilen bir algoritmadır. Adam algoritması adını adaptif moment tahmininden almaktadır. Algoritma çok katmanlı yapay sinir ağlarının eğitimi için önerilmiştir. Bu tez çalışmasında pekiştirmeli öğrenme algoritmaları için oluşturulan çok katmanlı yapay sinir ağlarının eğitimi için kullanılmıştır. Adam algoritması tablo 3.5'de görülmektedir. Bu algoritma tablo 3.2'de önerilen gradyan azalan algoritmasıyla beraber uygulanarak derin yapay sinir ağları eğitilmektedir. (Kingma, Diederik P. ve diğ. 2015)

Tablo 3.5: ADAM Algoritması.

ALGORİTMA : ADAM Algoritması			
1	**:**	α	*Adım aralığını belirle (0.001) önerilen*
2	**:**	$\beta_1, \beta_2 \in [0,1)$	*Moment değerinin üstel çürüme parametresi*
3	**:**	$f(\boldsymbol{\theta})$	*Stokastik amaç fonksiyonu*
4	**:**	$\boldsymbol{\theta}_0$	*Parametrelerin başlangıç değeri*
5	**:**	$\boldsymbol{m}_0$	*Birinci dereceden moment vektörü*
6	**:**	$\boldsymbol{v}_0$	*İkinci dereceden moment vektörü*
7	**:**	$t \leftarrow 0$	
8	**:**	$t \leftarrow t + 1$	
9	**:**	$\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} f_t(\boldsymbol{\theta}_{t-1})$	*Stokastik amaç fonksiyonunun gradyan vektörü*
10	**:**	$\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$	*Birinci moment tahminin güncellenmesi*
11	**:**	$\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2$	*İkinci moment tahmini güncellemesi*
12	**:**	$\widehat{\boldsymbol{m}}_t \leftarrow \frac{\boldsymbol{m}_t}{(1-\beta_1^t)}$	*Birinci momentin gerçek değeri*
13	**:**	$\widehat{\boldsymbol{v}}_t \leftarrow \frac{\boldsymbol{v}_t}{(1-\beta_2^t)}$	*İkinci momentin gerçek değeri*
14	**:**	$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \frac{\widehat{\boldsymbol{m}}_t}{\sqrt{\widehat{\boldsymbol{v}}_t + \varepsilon}}$	*Parametrelerin güncellenmesi* $\varepsilon = 1e-8$
15	**:**	Eğer $\| \boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1} \| < 1e-6$	
16	**:**	Algoritmayı sonlandır	

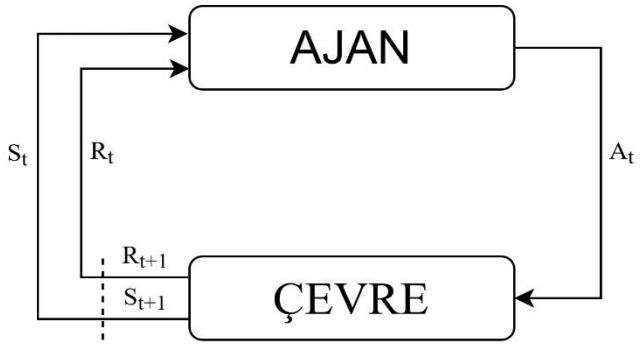
Denklem (3.18) ile önerilen çok katmanlı çok girişli çok çıkışlı yapay sinir ağı ADAM algoritmasıyla eğitilerek model parametrelerinin optimum değeri belirlenir.

4. ROBOT KOL KONTROLÜ İÇİN PEKİŞTİRMELİ ÖĞRENME YAKLAŞIMI

Günümüzde yapay zeka çalışmalarının büyük bir çoğunluğu makine öğrenmesi / yapay öğrenme konusunda yoğunlaşmaktadır. Makine öğrenmesinde problemlerin modellenme biçimi temel olarak danışmanlı öğrenme, danışmansız öğrenme ve pekiştirmeli öğrenme olarak üç sınıfta incelenmektedir. Makine öğrenmesi yöntemleri veri odaklı olup gerçek hayat problemlerine çözüm üretmektedirler. Makine öğrenmesi paradigması olan danışmanlı öğrenme konseptinde, makine öğrenmesi algoritması için giriş ve çıkış verisi bulunmaktadır. Çıkış verileri kategorik veya sembolik veriler olabilir. Danışmanlı öğrenmenin çözmeye çalıştığı problemler temelde, sınıflandırma ve regresyon problemine dönüştürülmektedir. Sınıflandırma probleminin çözümüyle örüntü tanıma, yüz tanıma, parmak izi tanıma, plaka tanıma, spam mail tanımlama, karar verme, tıbbi teşhis, hava durumu tahmin gibi birçok alanda uygulama yapılmaktadır. Regresyon probleminin çözümüyle süreç modelleme, zaman serisi tahmini, kontrol teorisi vb uygulama alanı bulunmaktadır. Diğer bir makine öğrenmesi paradigması danışmansız öğrenmedir. Bu problem tipi temelde kümeleme problemleriyle uğraşmaktadır. Danışmansız öğrenme modelinde gerçek hayattan veya bir süreçten toplanmış giriş verilerimiz bulunmaktadır. Giriş verilerimize karşılık düşen etiket verileri bulunmadan, geliştirilen makine öğrenmesi modelleri veri içerisindeki kümelenmeleri belirlemede, anomali belirleme, karar verme, örüntü çıkarma vb birçok problemi çözmektedir. Diğer bir paradigma olan pekiştirmeli öğrenmede, yapay modelimiz için giriş ve çıkış verileri bulunmamaktadır, yapay modelimizin bulunduğu ortam ile etkileşime girerek belirlenen hedefi yerine getirmesi için ödül toplaması ve toplanan bu ödüllerin maksimum veya minimum seviyede oluşturulmasını hedeflemektedir. Makine öğrenmesinin problemlerine çözüm olarak, makine öğrenmesi algoritmaları kullanılmaktadır. Bu algoritmalar sayesinde danısmanlı, danışmansız ve pekiştirmeli öğrenme ile problemler çözülmektedir. Bu tez çalışmasında bölüm ikide ifade edilen model tabanlı, robot kolun tork kontrolü problemine pekiştirmeli öğrenme modeli geliştirilerek çözüm oluşturulacaktır. Günümüzde pekiştirmeli öğrenme finans, robotik, havacılık vb birçok kritik uygulamada birçok başarı kaydetmiştir. Pekiştirmeli öğrenme optimal kontrol teorisinin çözemediği birçok problemi çözmektedir. (Sutton, Richard S. ve diğ. 2018)

4.1 Pekiştirmeli Öğrenme Temelleri

Pekiştirmeli (takviyeli) öğrenme, bilinmeyen bir dinamik ortamda oluşturulan yapay ajanın belirlenen görevi yerine getirmek için çevre ile etkileşime girmesi ve etkileşim sonucunda ödül değerlerini toplamasıdır. Pekiştirmeli öğrenme yaklaşımında yapay ajan çevre ile etkilişim sonucunda topladığı ödül değerini maksimum veya minimum seviyeye çekmeyi hedeflemektedir. Yapay ajan, çevreyle her etkileşim sonucunda çevre içerisinde bir durumda bulunmaktadır. Ajan bulunduğu durumda bir dizi aksiyonlar gerçekleştirip bir karar almaktadır. Ajan aldığı kararlar sonucunda çevre içerisinde yeni bir duruma geçmektedir ve bu durum sonucunda ajan çevreden pozitif veya negatif ödül toplamaktadır. Toplanan ödüllerin maksimum seviyeye geçmesi ve yapay ajanın vermiş olduğu sıralı kararlar sonucunda hedeflenen işi yapmayı öğrenmesi beklenmektedir. Bu durum döngüsel olarak şekil 4.1'de görülmektedir. (Sutton, Richard S. ve diğ. 2018)



Şekil 4.1: Pekiştirmeli Öğrenme Ajan ve Ortam İlişkisi.

Şekil 4.1'de R_t yapay ajanın çevreden almış olduğu ödülü ifade etmektedir. A_t yapay ajanın çevreye göndermiş olduğu aksiyonları ifade etmektedir. S_t yapay ajanın çevre içerisinde bulunduğu durumları ifade etmektedir. Yapay ajan herhangi bir t anında, S_t durumunda bulunurken A_t aksiyonunu gerçekleştirerek R_{t+1} ödülünü almakta ve buna bağlı olarak S_{t+1} durumuna geçmektedir.

4.1.1 Pekiştirmeli Öğrenmede Kullanılan Kavramlar

- **Ajan:** Çevreyi algılayan/keşfeden ve buna göre hareket edebilen yapı.
- **Çevre:** Ajanın içinde bulunduğu ve bağlı olduğu yer, çevre ajanın öğrenme gerçekleştireceği ve etkileşime girebileceği biçimdedir. Ajanın içinde bulunduğu çevre kısmen gözlemlenebilir veya tamamen gözlemlenebilirdir. Pekiştirmeli öğrenmede ajanın içinde bulunduğu ortam ve gözlemler rastsal olabilir.
- **Aksiyon/Eylem:** Aksiyonlar, ajan tarafından ortam içerisinde alınan kararlar olarak düşünülebilir ve $a \in A$ ile ifade edilir. Ajanın alacağı aksiyonlar sürekli değerli veya ayrık değerli olabilmektedir.
- **Durum:** Durum ajanın çevre içerisinde ne şekilde bulunduğu bilgisini tutar ve $s \in S$ olarak ifade edilir. Çevrenin içerisinde, ajan için gerekli minimum bilgiyi taşımakta olup, çevreye ve problemin tanımına göre sürekli zamanlı ve ayrık zamanlı olarak ifade edilmektedir.
- **Ödül:** Ajanın gerçekleştirdiği eylemler sonucunda çevreden almış olduğu geribildirimlerdir. Ödül pozitif veya negatif olabilir.
- **Politika:** Politika, t anında s_t durumunda bulunan ajanın aksiyon kümesi $A_t$içerisinden nasıl bir aksiyon seçeceğini belirlemektedir. Politika mevcut durum $\boldsymbol{s_t}$ile $a_t$aksiyonunu birbirine haritalamaktadır. Ajanın takip edeceği politika stratejisi, deterministik veya stokastik olarak modellenebilmektedir.
- **Yörünge:** Ajanın pekiştirmeli öğrenme algoritması sonucunda gerçekleştirdiği hedefe bağlı olarak, içinde bulunduğu durum ve bir sonraki durumlar arasındaki geçişlerin hepsini kapsamaktadır. Pekiştirmeli öğrenme ile ajanın belirlenen hedefi öğrenmesi ile, $\{s_1, s_2, \ldots, s_N\}$ durumlar kümesi yörüngemizi oluşturmaktadır.

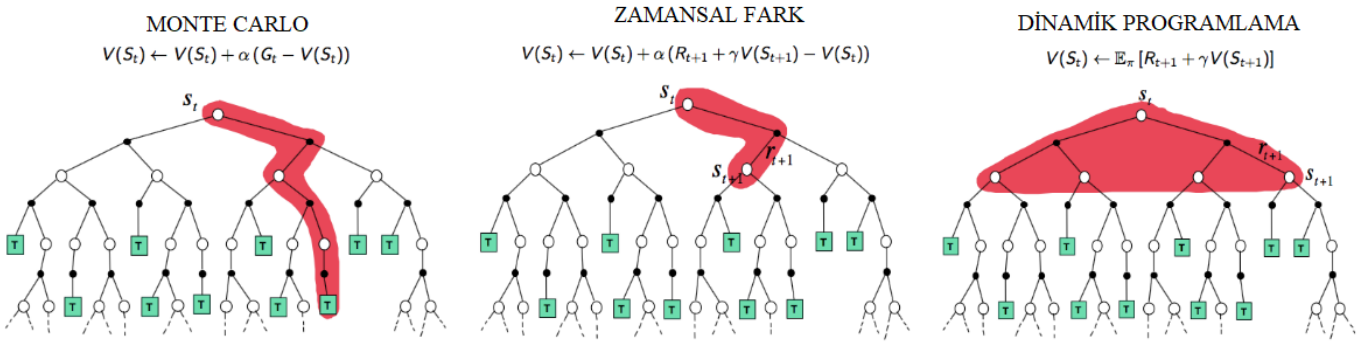
4.1.2 Pekiştirmeli Öğrenme Modelleri

Pekiştirmeli öğrenme algoritmaları çok çeşitli olarak sınıflandırılmaktadır. Pekiştirmeli öğrenmeyi model tabanlı ve modelden bağımsız olarak iki sınıfa ayırmak mümkündür. Model tabanlı algoritmalar çevre hakkında öncül bir bilgi veya çevreyi ifade eden matematiksel model üzerinde oluşturulmuştur. Model tabanlı pekiştirmeli öğrenme optimal kontrol metodu olan model öngörülü kontrole (MPC) benzerlik göstermektedir. Modelden bağımsız pekiştirmeli öğrenme algoritmaları, çevre hakkında hiçbir bilgiye sahip olmamaktadır. Çevre genellikle rastgele süreç olan Markov karar süreci ile modellenerek algoritmalar geliştirilmektedir. Yapay ajan çevreyi keşfederek çevre hakkında bilgi toplayarak optimal kararlar

vermektedir. Bu süreç pekiştirmeli öğrenme yaklaşımını deneme, yanılma ve ödül toplama olarak oluşturmaktadır. Pekiştirmeli öğrenme günümüzde birçok alanda uygulanmakta olup, başarı elde etmesinin en temel sebeplerinden biri çevre hakkında bir modele ihtiyaç duymamasıdır. Bu sayede birçok mühendislik problemini çevre içerisinde etkileşime girerek çözmektedir. Bu tez çalışmasında modelden bağımsız pekiştirmeli öğrenme algoritmaları üzerinde durulacaktır. Pekiştirmeli öğrenmenin bir diğer sınıflandırması sürekli veya ayrık durum/aksiyon çiftleri üzerindedir. Bu sınıflandırmada ajanın içerisinde bulunduğu çevrenin durumları sürekli zamanlı veya ayrık zamanlı olabilmektedir. Ajanın çevre içerisinde etkileşime girmek için uyguladığı aksiyonlar sürekli ve ayrık olabilmektedir. Robot ve kontrol problemleri için, durum ve aksiyon çiftinin sürekli olması gerekmektedir. Ajanın içinde bulunduğu ortam sonlu boyutlu ayrık aksiyon ve durum çiftinden oluşuyorsa pekiştirmeli öğrenme algoritmaları tablo yöntemlerini temel alan algoritmalarla çözüm üretmektedir. Ajanın etkileşimde bulunduğu çevre veya aksiyon çiftinin sürekli değerler alması durumunda tablo yönteminde önerilen algoritmalar etkili olmamaktadır. Bu tez çalışmasında sürekli ve ayrık aksiyon ve durum çiftleri ile oluşturulan algoritmalar üzerinde durulacaktır. Bölüm ikide ifade edilen robot kolun tork kontrol problemi sürekli askiyon ve durum çiftiyle modellendiği için derin deterministik politika gradyanı algoritmasıyla problem çözümlenecektir. Pekiştirmeli öğrenmenin diğer bir sınıflandırılması, değer tabanlı ve politika tabanlı algoritmalardır. Değer tabanlı pekiştirmeli öğrenme algoritmaları, yapay ajanın çevre içerisinde bulunduğu bir s_t durumunda belirli bir a_t eyleminde bulunmanın değerlerini öğrenmektedir. Değer tabanlı yaklaşımda algoritmaların uygulamış olduğu politika, ajanın çevreye uyguladığı eylemlerin getirisinin hangisi en yüksek ise onun seçmesi şeklindedir. Böyle bir yaklaşım aç gözlü ($greedy$) yaklaşım olarak isimlendirilmektedir. Ajan bulunduğu durumda uyguladığı eylemlerin hangisi en büyük getiriyi sağlıyorsa bu aksiyonu seçmektedir. Böylece ajanın politikası aç gözlü yaklaşım olarak isimlendirilmektedir. Politika tabanlı pekiştirmeli öğrenme algoritmalarında, ajanın içinde bulunduğu s_t durumlarının değerlerini öğrenmek yerine uygun politikayı keşfetmeyi hedeflemektedir. Bu yaklaşımda ajanın takip ettiği aç gözlü yaklaşımdan farklı olarak her bir durumda uygun kararlar vermesini sağlamak amaçlanmaktadır. Böylece ajan içinde bulunduğu ortam içerisinde politikayı öğrenerek bir davranış biçimi geliştirmektedir. Değer tabanlı yaklaşım ve politika tabanlı yaklaşım ile uygulanan probleme göre algoritmalar farklılık göstermektedir. Değer tabanlı ve politika tabanlı yaklaşımların beraber kullanıldığı pekiştirmeli öğrenme algoritmaları mevcuttur. Böyle bir durumda ajan s_t durumunda olmanın getirdiği değer ve uygun politikaları öğrenmektedir. Bu tez çalışmasında önerilen derin deterministik politika gradyanı algoritması hem değer hemde politika tabanlı algoritma ailesinden gelmektedir. Pekiştirmeli öğrenmede algoritmaların diğer bir sınıflandırma biçimi açık politika ve kapalı politikadır. Açık politikada ajan, eğitim esnasında her adımda uyguladığı politikayı değiştirmektedir böylece en güncel politika üzerinden kararlar almaktadır. Kapalı politikada ajan çevre içerisinde bilgi topladıktan sonra politikasını güncellemektedir. Kapalı politika yaklaşımı ajanın daha kararlı öğrenmesini sağlamaktadır. Ajanın her adımda güncellediği politika,

uygun politika olmayıp yanlış bir karar almasına sebep olabilir. Pekiştirmeli öğrenme yaklaşımlarından model tabanlı yaklaşımda ortamın tam bir matematiksel modeline sahip olduğumuz için kapsamlı bir arama algoritmasıyla, yani tüm olası sonuçları araştırarak sonuca varabiliriz. Arama işlemi işlemsel karmaşıklık sebebiyle basit uygulamalarda örneğin tic-tac-toe oyununda uygulanabilmektedir. Arama karmaşıklığı ortamın durumları ve aksiyonlarına bağlı olarak artmaktadır. $s = 100$ durumlu bir ortam hakkında olası durum uzayımızın boyutu 2^{100} ve her bir durum için olası aksiyonlar düşünüldüğünde çevre modelinin bildiği varsayımıyla arama yöntemiyle sonuç bulmak birçok problem için pratik değildir. Çevre hakkında matematiksel bir modele sahip olduğumuz durumda, çözüm için kullanılan diğer bir yaklaşım ise dinamik programlamadır. Dinamik programlamada olası durum ve aksiyonlara bağlı olarak kapsamlı arama yapmak yerine optimal politikayı iteratif olarak çözmemizi sağlar. Dinamik programlama optimal çözüm için bir metodoloji geliştirmemizi sağlar ve iteratif olarak çözümü hesaplar. Çevre hakkında kesin bir bilgiye sahipsek, dinamik programlama ve kapsamlı arama yaklaşımları optimal politikayı garanti etmektedir. Pekiştirmeli öğrenme yaklaşımlarının çoğu, çevre hakkında detaylı bir öncül bilgi olmadığını ve bazı algoritmalarda çevrenin tüm durumlarına erişilemediğini (POMDP) varsaymaktadır. Fiziksel dünyada kurulan sistemler düşünüldüğü zaman çoğunlukla zamanla değişen, belirlenemeyen dinamikler, gürültü vb problemler karşımıza çıkmaktadır. Bu sebeplerden birçok fiziksel sistem stokastik davranışlar sergilemektedir. Dinamik programlama yaklaşımı deterministik veya stokasik olmasına karşılık matematiksel modele ihtiyaç duymaktadır. Pekiştirmeli öğrenmenin modelden bağımsız yaklaşımı sayesinde dinamik programlamanın yaklaşık bir çözümü oluşturulmaktadır. Dinamik programlama bir önceki bölümde ifade edilen optimal kontrol problemi içinde uygulanmaktadır. Optimal politikayı yani kontrol işaretini belirli bir amaç fonksiyonunu minimize ederek oluşturmaktadır. Pekiştirmeli öğrenmenin modelden bağımsız yaklaşımında çevre ile etkileşime girerek oluşturulan ödül fonksiyonu sayesinde, yapay ajanın topladığı ödülü maksimize veya minimize etmeyi amaçlarız. Modelden bağımsız pekiştirmeli öğrenmede iki temel yaklaşım olan, monte carlo ve zamansal fark metodları kullanılmaktadır. Monte carlo yaklaşımında, bir bölüm sonunda veya ortam içerisinde son duruma ulaşıncaya kadar algoritma kendini güncellememektedir. Bölümün sonuna ulaşıldığında algoritmanın güncellemesi oluşturulacaktır. Zamansal fark metodunda belirli bir zaman aralığında algoritmanın güncellemesi gerçekleşmektedir. Zamansal fark algoritmaları genellikle TD ile ifade edilmektedir ve matematiksel olarak $TD(0), TD(1)$ şeklinde ifade edilir. İfade biçimine bağlı olarak en genel gösterim $TD(\lambda),\ \lambda \in [0,1]$ olarak gösterilmektedir. $TD(0)$ yaklaşımı zamansal fark olarak sadece bir adım ileriyi incelediğimizi ifade etmektedir. $TD(1)$ yaklaşımı monte carlo yaklaşımına eşdeğerdir yani bölüm sonunda güncelleme oluşturduğumuz anlamındadır. Böylece $TD(\lambda)$ ifadesinde λ terimi, algoritmanın gelecekte nasıl bir ufuk inceleyeceğini belirtmektedir. $TD(0)$ ve $TD(1) = MC$ metodu arasındaki farklar, MC metodu yüksek varyansa sahipken TD metodu düşük varyansa sahiptir. Bunun sebebi MC metodunun bir bölüm boyunca bütün aksiyon durum çifti üzerinden güncelleme yaparken, TD metodu tek bir

adımda güncelleme kuralına sahip olmasıdır. MC metodu parametreleri güncellemek için bir bölümü göz önüne aldığı için düşük yanlılık değerine sahipken, TD metodu anlık güncellemeler üzerinden ilerlediği için yüksek yanlılık değerine sahiptir. Bu tez çalışmasında incelenen algoritmalar TD metodunu kullanarak oluşturulmuştur. TD metodu temelde Sarsa ve Q öğrenme algoritmaları olarak iki temel algoritmaya sahiptir. Model tabanlı algoritmalar ve modelden bağımsız yaklaşım şekil 4.2'de görülmektedir. (Sutton, Richard S. ve diğ. 2018)



Şekil 4.2: Pekiştirmeli Öğrenme Metodları.

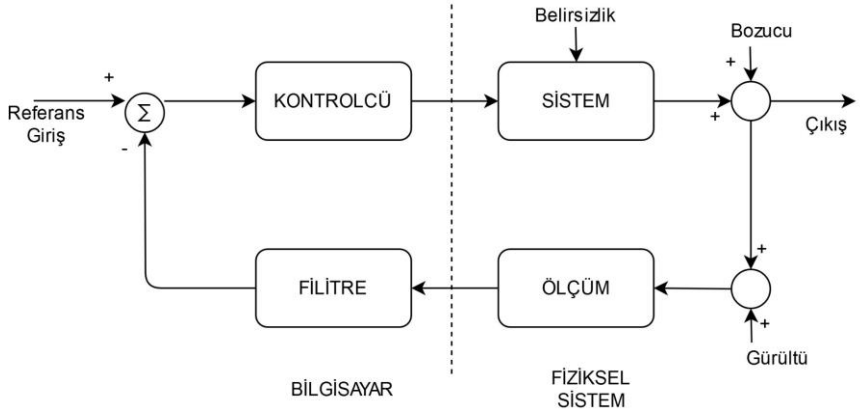
4.1.3 Pekiştirmeli Öğrenme Keşif ve Sömürü

Modelden bağımsız pekiştirmeli öğrenme yaklaşımında ajanın çevre içerisinde hiçbir bilgiye sahip olmadığı varsayılmaktadır. Ajanın ortamla etkileşime girip kararlar alması için ortamı keşfetmesi gerekmektedir. Ortamdaki durumlar ve durumlara göre alınan ödül değerleri üzerinden karar vermesi ve optimum politikayı oluşturması için ortamda ajanın gezmesi gerekmektedir. Her durum ve eylem çifti için sıfır değeri tahminiyle başladığımızı varsayalım. Pozitif ödül veren bir eylemde bulunduğumuz anda aç gözlü politikayı izleyen bir algoritma, bu eylemi hiç bozmadan uygulamaya devam edecektir. Uygulanan eylemin optimal olup olmadığı belirli olmadan pekiştirmeli öğrenme algoritmamız lokal mininmum noktaya takılıp kalacaktır. Bu durumda yapay ajanın karar vermeden keşif ve sömürü konseptine uyması gerekmektedir. Ajanın bulduğu optimal politikadan daha alt optimal politikalar bulunması olasıdır. Ajan ortam içerisinde başlangıçta rastsal değerle gezinerek arama işlemi yapmalı ve ortamı iyice keşfetmelidir. Ortam hakkındaki neredeyse bütün dinamiklere erişip, değerlere erişim sağladıktan ve optimal kararlar almaya

başladıktan sonra algoritmamızın keşif yaklaşımını bırakıp sömürü yaklaşımına yönelmesi, yani önceden edindiği bilgileri kullanması gerekmektedir. Hatırlanacağı üzere modelden bağımsız yaklaşımlar optimal politikayı model tabanlı algoritma olan DP gibi garanti etmemektedir. Modelden bağımsız yaklaşımda optimal politikayı oluşturmak için ajanın ortam ile etkileşimde bulunması ve en optimal kararları alması beklenmektedir. Algoritmanın keşif aşamasında rastgele davranışlar sergileyip ortam içerisinde stokastik bir biçimde gezmesi, kötü ve iyi olan tüm sonuçları toplaması gerekmektedir. Sömürü aşamasında algoritmamız en yüksek değerleri olan kararlar üzerinden harekete geçmelidir. Pekiştirmeli öğrenmede algoritmalarımızın keşif ve sömürü arasındaki dengeyi iyi kurması beklenmektedir. Statik bir ortam içerisinde algoritmamızın başlangıç anında maksimum keşif, öğrenme aşamasından sonra tecrübelerinden yola çıkarak uygun politikayı takip etmesi beklenmektedir. Keşif ve sömürü arasındaki dengeyi matematiksel olarak oluşturmak için $\varepsilon - greedy$ yaklaşımı kullanılmaktadır. Bu yaklaşımda ajan başlangıçta $\varepsilon \in [0,1]$ değeriyle olasılıksal bir aksiyon gerçekleştirerek maksimum düzeyde rastgele eylem uygulamakta böylece bulunduğu ortamı keşfetmektedir. Zaman ilerledikçe ε değeri üstel bir azalma gösterek, daha az keşif yöneliminde olup sömürü aşamasına geçmektedir. Böylece öğrenme gerçekleştikçe algoritmanın tecrübelerini kullanması sağlanmaktadır. (Kolter, J. Zico. 2016)

4.1.4 Pekiştirmeli Öğrenme ve Kontrol Teorisi

Kontrol teorisinin temel problemi, fiziksel bir sistem için uygun kontrol işareti üreterek hedeflenen referans işaretini takip etmektir. Klasik bir kapalı döngü kontrol sisteminin blok şeması şekil 4.3'de görülmektedir.



Şekil 4.3: Genel Kontrol Yapısı.

Şekil 4.3'de genel kontrol yapısıyla fiziksel sistemleri kontrol etmek mümkündür. Kontrol teorisinde fiziksel sistemin matematiksel bir modeli varsa model tabanlı kontrol sistemi oluşturmak mümkündür.

Fiziksel sistemin modelinin bilinmediği durumlarda, klasik kontrol teorisi yaklaşımında sistem tanılama teknikleri kullanılmaktadır. Fiziksel sistem üzerine etki eden bozucu girişler, gürültü ve modellenemeyen dinamikler oluşturulan kontrolcünün performansını etkilemektedir. Optimal kontrol teorisi bir önceki bölümde incelenerek belirli bir performans kriterini sağlayacak optimal kontrol olarak düşünülebilir. Optimal kontrol teorisinde fiziksel sistemin modelinin olması ve optimal kararlar verilmesi model tabanlı pekiştirmeli öğrenme konseptiyle benzerlik göstermektedir. Pekiştirmeli öğrenmenin optimal kontrol teorisine göre avantajı ödül fonksiyonunu özgürce yazabiliyor olmamızdır. Optimal kontrol teorisinde, amaç fonksiyonu belirli bazı fonksiyonlardan oluşmaktadır ve çözüm bu fonksiyonlara göre oluşturulmaktadır. Kontrol teorisinde oluşturulan kontrol işareti $u(t)$ pekiştirmeli öğrenmede aksiyonları $a(t)$ oluşturmaktadır. Kontrol teorisinde tasarlanan kontrolcü, pekiştirmeli öğrenmedeki politikaya karşılık düşmektedir. Kontrol teorisinde oluşturulan kontrolcüler, fiziksel sisteme ve kontrol performansına göre değişmektedir. Optimal bir kontrolcü oluşturmak için literatürde, model ön görülü kontrol MPC-NMPC, lineer kuadratik düzeltici LQR vb metodlar önerilmiştir. Kontrolcünün gürbüz veya adaptif olması olması gerektiği durumlarda Hinf, MRAC vb kontrol metodları önerilmiştir. Endüstriyel birçok sistemde PID kontrolcü kullanılmaktadır. PID kontrolcünün avantajı, ampirik bir şekilde parametrelerinin ayarlanıyor olması ve fiziksel sistemin matematiksel modelinden bağımsız olmasıdır. Pekiştirmeli öğrenmede oluşturulan ortam/çevre fiziksel sistemin dinamik denklemlerine karşı düşmektedir ve genellikle lineer olmayan zamanla değişen stokastik diferansiyel denklem formatında ifade edilmektedir. Fiziksel dünyada birçok sistem model tabanlı kontrol algoritmalarıyla otomatik bir şekilde kontrol edilmektedir. Kontrol teorisi günümüzde, otonom araçlar, roket sistemleri, uçaklar, robotlar vb birçok alanda endüstrinin ve sanayinin temelini oluşturmaktadır. Kontrol teorisi, matematiksel modeller belirli olduğunda analitik çözümler ürettiği için çok güçlüdür. Bu tez çalışmasında, robot kolun matematiksel modeli oluşturulmuş ve tork kontrolü yapılmıştır. Kontrol sistemlerinin kurulması için fiziksel sistemin matematiksel modellerinin çok iyi yazılması gerekmektedir. Oluşturulan kontrolcünün optimal veya adaptif kontrol performansı sergilemesi beklenmektedir. Kontrol teorisinde, fiziksel sistemlerin modellerinin kurulamaması veya sistemdeki belirsizlikler problem oluşturmaktadır. Robot teknolojisinde, paralel manipülatörler ve modelleri birçok problem içermektedir. Robotikte yürüme, insan hareketlerini taklit etme vb problemleri çözmek için, güçlü matematiksel modelleme ve kontrol bilgisine ihtiyaç duymaktadır. Oluşturulan algoritmalar her zaman en iyi performansı sunmamaktadır. Makine öğrenmesi ve pekiştirmeli öğrenmenin gelişmesiyle kontrol teorisinde karşılaşılan zorluklar ve problemler çözülmektedir. Bu tez çalışmasında, bölüm ikide oluşturulan robot kol tork kontrol problemi pekiştirmeli öğrenme algoritmasıyla çözülmüştür. Böylece fiziksel sistemden ve bozucu girişlerden bağımsız olarak optimal çözümler oluşturulmuştur. Pekiştirmeli öğrenme için esnek olarak yazılan ödül fonksiyonları sayesinde fiziksel sistemler için uygun çok farklı optimal kontrol oluşturulmaktadır. (Kolter, J. Zico. 2016)

4.2 Rastgele Süreçler ve Markov Süreci

Bu tez çalışmasında modelden bağımsız, ve zamansal fark metodunu kullanan pekiştirmeli öğrenme algoritmaları incelenecektir. Fiziksel sistemlerin matematiksel modelleri lineer olmayan diferansiyel denklem ile 4.1'deki gibi yazılmaktadır.

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= f(t, \boldsymbol{x}, \boldsymbol{u}, w) \\ \boldsymbol{y}(t) &= g(t, \boldsymbol{x}, \boldsymbol{u}, v) \end{aligned} \tag{4.1}$$

Denklem 4.1'de t zamanı, $\boldsymbol{x}$ durum vektörünü, $\boldsymbol{u}$ kontrol işaretini, w proses gürültüsünü, v ölçüm gürültüsünü $f(.)$ sistem modelini oluşturulan lineer olmayan diferansiyel denklemleri, $g(.)$ çıkış denklemini oluşturmaktadır. Fiziksel sistemin matematiksel modelinin çözümü biliniyorsa, durum geçiş matrisi üzerinden durumlar elde edilebilir. Fakat birçok fiziksel sistem belirsizlik, gürültü ve modellenemeyen dinamikler yüzünden çözülememektedir. Fiziksel sistemlerin, lineer olmayan modelleri için çoğunlukla analitik çözümü bulunmamaktadır. Rastgele süreçlerle sistemin matematiksel modelini oluşturmak için sistemin durumları olan $\boldsymbol{x}$'ler rastgele değişken olarak ifade edilir. Böylece durumlar arasındaki geçişler stokastik bir bicimde modellenir. Pekiştirmeli öğrenme yaklaşımıyla, fiziksel sistemler ve durumları arasındaki geçişleri rastgele süreç olan Markov karar süreci ile modellemektedir. Mühendislik ve bilimsel uygulamalarda rastgele olaylar bir dizi biçiminde karşımıza çıkmaktadır. Rastgele olaylar dizisi zamanın veya konumun bir veya çok boyutlu fonksiyonlarıyla ifade edilebilir. Böyle durumlarda rastgele süreç kavramıyla modelleme yapılır. Rastgele süreç matematiksel olarak $\boldsymbol{X}(t), \boldsymbol{X}(n)$ ile ifade edilmektedir. Olasılık uzayındaki her bir olaya belirli bir kurala göre bir fonksiyon atanmasıyla oluşturulan fonksiyon topluluğu rastgele süreç veya stokastik süreç olarak adlandırılmaktadır. Rastgele süreci ifade eden fonksiyon topluluğunda, basit (PDF) veya ortak olasılık yoğunluk fonksiyonları (JPDF), zamanla değişmiyorsa bu süreç durağan olarak isimlendirilmektedir. Durağan süreçlerde, sürecin tüm momentleri zamandan bağımsızdır. Bazı durumlarda rastgele süreci ifade eden fonksiyonların aynı istatistiksel özellikleri sergilemektedir. Bu durum rastgele sürecin ergodiklik özelliğini taşıması anlamındadır. Bu gibi özel durumlarda sürece ilişkin gerçeklemelerde rastgele süreci ifade eden tek bir örnek fonksiyonun bilinmesiyle tüm sürecin istatistiksel özellikleri modellenebilmektedir. Rastgele süreçler ayrık değerli ayrık zamanlı rastgele süreç, ayrık değerli sürekli zamanlı rastgele süreç, sürekli değerli ayrık zamanlı rastgele süreç ve son olarak sürekli değerli sürekli zamanlı süreç olarak dört sınıfta incelenmektedir. Bu tez kapsamında ayrık değerli ayrık zamanlı rastgele süreç olan Markov süreci ve Markov sürecinin genişletilmiş durumları incelenecektir. Denklem 4.2'yi sağlayan rastgele süreçler Markov süreci olarak adlandırılmaktadır.

$$P[X(t_{k+1}) = x_{k+1}|X(t_k) = x_k, \ldots, X(t_1) = x_1] = P[X(t_{k+1}) = x_{k+1}|X(t_k) = x_k] \quad (4.2)$$

Denklem 4.2'den görüleceği üzere $X(t)$ rastgele sürecinin, $X(t_{k+1})$ değeri bir geçmiş değer olan $X(t)$ 'ye bağlıdır. Bu özellik Markov özelliği olarak isimlendirilmektedir. Bu özelliği sağlayan rastgele süreçler Markov süreci olarak isimlendirilmektedir. Denklem 4.2'de $t_1 < t_2 < t_3 < \ldots < t_k < t_{k+1}$ zaman indisini ifade etmektedir. Rastgele sürece ilişkin ortak olasılık yoğunluk fonksiyonunun (JPDF) bilinmesi durumunda Markov süreci durum geçiş matrisiyle ifade edilmektedir. Bu durum denklem (4.3)'de görülmektedir.

$$P[X_{n+1} = j|X_n = i] = p_{ij} \quad (4.3)$$

Denklem 4.3'de X_n homojen geçiş olasılıkları olarak isimlendirilmektedir. $X_n, \ldots, X_0$ ortak olasılık yoğunluk fonksiyonun verilmesiyle denklem 4.3, denklem 4.4'deki gibi ifade edilebilir.

$$P[X_n = i_n, \ldots, X_0 = i_0] = p_{i_{n-1}, i_n \ldots} p_{i_0 i_1} p_{i_0}(0) \quad (4.4)$$

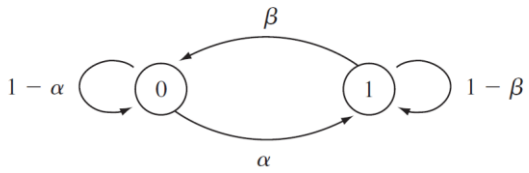
Böylece geçiş olasılıkları matrisi P, başlangıç değerleri olan $p_i(0)$ verilmesi durumunda denklem (4.5)'deki gibi elde edilir.

$$P = \begin{bmatrix} p_{00} & p_{01} & p_2 & \cdots \\ \vdots & \vdots & \cdots & \cdots \\ \vdots & \vdots & \cdots & \cdots \\ p_{i0} & p_{i1} & \cdots & \cdots \end{bmatrix} \quad (4.5)$$

Denklem 4.5'de ifade edilen durum geçiş matrisinin sütunları üzerinden toplamı bir yapmalıdır. Sütun toplamı ifadesi denklem 4.6'da görülmektedir.

$$\sum_j P[X_{n+1} = j|X_n = i] = \sum_j pij = 1 \quad (4.6)$$

Şekil 4.4'de iki durumlu Markov süreci için bir örnek görülmektedir.



Şekil 4.4: İki Durumlu Markov Zinciri Durum Geçiş Diyagramı.

Şekil 4.4'le ifade edilen iki durumlu Markov zincirinin durum geçiş matrisi denklem 4.7'ile ifade edilmektedir.

$$P = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix} \tag{4.7}$$

Şekil 4.4'de ifade edilen iki durumlu Markov zincirinin zamana bağlı olarak hesaplanması için denklem 4.7'nin kuvvetlerinin alınması gerekmektedir. Bu durum n adım hesaplaması için denklem 4.8'deki gibidir.

$$P^n = \underbrace{\begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}\begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}\cdots\begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}}_{n} \tag{4.8}$$

Denklem 4.8'ile n adımlı durum geçiş olasılıkları, P durum geçiş matrisinin bilinmesiyle hesaplanabilir. Bir X_k rastgele süreci bağımsız ve aynı olasılık dağılımıyla ifade edilmiş ise bu süreç IID olarak ifade edilmektedir ve denklem 4.9 ile matematiksel olarak ifade edilir.

$$\begin{aligned} F_{X_1,X_2,\ldots,X_k}(x_1,x_2,\ldots,x_k) &= P[X_1 \le x_1, X_2 \le x_2, \ldots, X_k \le x_k] \\ F_{X_1,X_2,\ldots,X_k}(x_1,x_2,\ldots,x_k) &= F_X(x_1)F_X(x_2)\ldots F_X(x_k) \end{aligned} \tag{4.9}$$

Denklem 4.9, X_k rastgele değişkenlerinin istatistiksel bağımsızlığını ifade etmektedir. Bu bölümde oluşturulan pekiştirmeli öğrenme algoritmaları için ajanın bulunduğu ortamdan elde ettiği veriler IID olmamalıdır. Bu şekilde elde edilen veriler ile oluşturulan algoritmaların lokal noktaya takılma problemi ortadan kalkmaktadır. (Garcia, Alberto Leon. 2005)

4.2.1 Markov Ödül Süreci

Pekiştirmeli öğrenme yaklaşımında, yapay ajanın içinde bulunduğu ortamın matematiksel modeli bilinmediği varsayılmaktadır. Yapay ajanın bulunduğu ortam ve durumlar arasındaki geçiş Markov sürecinin genişletilmiş hali olan Markov ödül süreçleriyle modellenmektedir. Bu tez kapsamında ortam durumlarına tam olarak erişilebildiği varsayılmaktadır. Ortam durumlarına tam olarak erişimin sağlanmadığı durumda Markov karar süreçleri, kısmi gözlemlenen Markov karar süreci olarak isimlendirilmektedir. Markov özelliği ve durum geçiş olasılığı matrisi denklem 4.10'daki gibidir.

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, S_2, \ldots, S_t]$$

$$P = \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \tag{4.10}$$

Denklem 4.10'dan Markov özelliği sağlayan rastgele sürecin belleksiz olduğu görülmektedir. Markov süreci veya Markov zinciri (S, P) ile tanımlanmaktadır. S sonlu durum uzayını ifade etmektedir. P durum geçiş olasılıkları matrisidir. $P_{ss'} = P[S_{t+1} = s'|S_t = s]$ ile matematiksel olarak ifade edilmektedir ve s' bir sonraki durumu ifade etmektedir. $P_{ss'}$, $s \rightarrow s'$ geçiş olasılığını ifade etmektedir. Markov ödül süreci, Markov sürecinin değer ifadesiyle genişletilmiş halidir. Markov ödül süreci (S, P, R, γ), S, P Markov sürecinde belirtildiği gibi, R ödül fonksiyonunu belirtmekte olup $R_s = E[R_{t+1}|S_t = s]$, γ azaltma faktörü olarak $\gamma \in [0,1]$ aralığında değer almaktadır. Markov ödül sürecinde tanımlanan G_t fonksiyonu zaman adımı üzerinden toplam azaltılmış ödülü ifade etmektedir ve matematiksel olarak denklem 4.11'deki gibi tanımlanmaktadır.

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{4.11}$$

Denklem 4.11'de ifade edilen γ parametresi, G_t ödül fonksiyonunun değerinin gelecekteki ödüllere ne kadar bağlı olduğunu ifade etmektedir. $\gamma = 0$ için algoritma bulunduğu t anında almış olduğu R_{t+1} ödülüne bağlıdır. $\gamma = 1$ için G_t ödül fonksiyonu gelecekteki bütün ödüllere bağlılığı ifade etmektedir. γ parametresi gelecek ve şimdi arasındaki dengeyi kurmaktadır. Optimal kontrol teorisinde kullanılan MPC algoritması, ufuk kontrolü olarak isimlendirilmektedir. Burada oluşturulan kontrol ufku ifadesi optimizasyon sürecinin gelecekteki kaç örneği kapsaması gerektiğini ifade etmektedir.

Pekiştirmeli öğrenmede kullanılan modelden bağımsız yaklaşım ile yapay ajan ortamı ve ortama bağlı olarak alacağı ödülü tam olarak bilmediği için gelecek ve şimdi arasında bir bağlantı kurması gerekmektedir. γ parametresi gelecek ve şimdiki ödüller arasındaki dengeyi sağlamaktadır. Markov ödül sürecinin çözümü için değer metodu kullanılmaktadır. Değer fonksiyonu yaklaşımı denklem 4.12'de görülmektedir.

$$V(s) = E[G_t|S_t = s] \tag{4.12}$$

Denklem 4.12'de $V(s)$ değer fonksiyonu denklem 4.11 ile ifade edilen G_t fonksiyonunun beklenen değerini oluşturmaktadır. Değer fonksiyonu 4.12'yi iki parçaya bölünmektedir. Bu durum denklem 4.13'de görülmektedir.

$$\begin{aligned} V(s) &= E[G_t|S_t = s] \\ &= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots |S_t = s] \\ &= E[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots)|S_t = s] \\ &= E[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= E[R_{t+1} + \gamma V(S_{t+1})|S_t = s] \end{aligned} \tag{4.13}$$

Denklem 4.13 denklem 4.12'de ifade edilen değer fonksiyonunun Bellman denklemi olarak ifade edilmiş biçimidir. 4.13 denklemi lineer denklem sistemi olarak ifade edilmiş hali denklem 4.14'de görülmektedir.

$$V(s) = R_t + \gamma \sum_{s' \in S} P_{ss'} V(s') \tag{4.14}$$

Denklem 4.14 ile ödül fonksiyonu R_t ve $P_{ss'}$ durum geçiş matrisinin bilinmesiyle Markov ödül süreci için, s durumunda bulunmanın değeri hesaplanabilir. 4.14 ile ifade edilen Bellman denklemi, matris vektör olarak denklem 4.15'de görüldüğü gibi yazılabilir.

$$\begin{bmatrix} V(1) \\ \vdots \\ V(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \begin{bmatrix} V(1) \\ \vdots \\ V(n) \end{bmatrix} \tag{4.15}$$

Denklem 4.15 kullanılarak Markov ödül sürecinin çözümü denklem 4.16'da görüldüğü gibidir.

$$\begin{aligned} V &= R + \gamma PV \\ (1 - \gamma P)V &= R \\ V &= (1 - \gamma P)^{-1}R \end{aligned} \tag{4.16}$$

Denklem 4.16 ile Vdeğer fonksiyonu hesaplanabilir. Böylece Markov ödül süreciyle modellenen ortamda, yapay ajanın her bir durum içinde bulunmasının ödülleri hesaplanmış olur. $V(s)$ değer fonksiyonu $s \in S$ durum kümesine bağlıdır. Markov ödül süreci ayrık durumlu ayrık değerli olduğu için sonlu sayıda, sayılabilir durum kümesine sahiptir. Denklem 4.16 sınırlı sayıda durum değeri için çözüm hesaplanabilir. Denklem 4.16'nın hesapsal karmaşıklık değeri n adet durum $O(n^3)$ olmaktadır. Çok büyük boyutlu MRP için, iteratif çözüm metodları, dinamik programlama, monte carlo metodu ve zamansal fark metodu çözüm için kullanılmaktadır. (Kolter, J. Zico. 2016)

4.2.2 Markov Karar Süreci

Markov karar süreci, Markov ödül sürecinin $a \in A$ aksiyon kümesiyle genişletişmiş halidir ve matematiksel olarak, S Markov özelliği sağlayan sonlu ve ayrık durum kümesi, A aksiyon kümesi, $P_{ss'}^{a} = P[S_{t+1} = s' | S_t = s, A_t = a]$ koşullu geçiş olasılığı, $R_s^a = E[R_{t+1} | S_t = s, A_t = a]$ ödül fonksiyonu ve $\gamma \in [0,1]$ azaltma faktörüyle tanımlanmaktadır. Markov karar sürecinde tanımlanan iki önemli yaklaşım, politika ve değer fonksiyonudur. Politika π ile gösterilmektedir. Politika matematiksel olarak denklem 4.17'de görülmektedir.

$$\pi(a|s) = P[A_t = a | S_t = s] \tag{4.17}$$

Denklem 4.17 ile ifade edilen politika ajanın s durumunda a aksiyonunu almasını, yani ajanın davranışını ifade etmektedir. MDP'de politika ajanın içinde bulunduğu $s_t$durumuna bağlıdır, gelecekteki veya geçmişteki durumlara bağlı değildir. Politika zamandan bağımsızdır. Bu durum denklem 4.18'de görülmektedir.

$$A_t \sim \pi(.|S_t) \forall t > 0 \tag{4.18}$$

Markov karar süreci $M = (S, A, P, R, \gamma)$ ve bir politika π verildiğinde, $S_1, S_2, \ldots$ durumları Markov özelliği (S, P^π) sağlamaktadır. Durum ve ödül değerleri olan $S_1, R_1, S_2, R_2, \ldots$ Markov karar sürecini $(S, P^\pi, R^\pi, \gamma)$ ifade etmektedir. Denklem 4.19'da P^π, R^π matematiksel olarak tanımlanmaktadır.

$$P_{ss'}^\pi = \sum_{a \in A} \pi(a|s) P_{ss'}^a$$

$$R_s^\pi = \sum_{a \in A} \pi(a|s)\, R_s^a \tag{4.19}$$

Markov karar sürecinde tanımlanan diğer bir ifade, değer fonksiyonudur. Durum değer fonksiyonu matematiksel olarak denklem 4.20'de ifade edilmektedir.

$$V_\pi(s) = E_\pi[G_t|S_t = s] \tag{4.20}$$

Denklem 4.20'de durum-değer fonksiyonu görülmektedir. Yapay ajan Markov karar süreci ile modellenen çevre içerisinde herhangi bir s durumundan başlayıp belirli bir politikayı π takip ederek alacağı ödülleri ifade etmektedir. Markov karar sürecinde diğer bir tanımlanan fonksiyon, aksiyon-değer fonksiyonudur ve matematiksel olarak denklem 4.21'deki gibi ifade edilmektedir.

$$q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a] \tag{4.21}$$

Denklem 4.21'de yapay ajan Markov karar süreci ile modellenen çevre içerisinde herhangi bir s durumundan başlayıp belirli bir politika π takip edip ve belirli bir aksiyon a uygulaması sonucunda almış olduğu ödül değerlerini ifade etmektedir. Markov karar sürecinde ajanın ortam içerisinde aksiyon ve ödüller toplaması sonucunda Markov karar sürecinin iki farklı çözüm yaklaşımı bulunmaktadır. Markov karar sürecinde oluşturulan, durum-değer ve aksiyon-değer fonksiyonlarının iteratif olarak Bellman denklemine dönüştürülmüş ifadesi matematiksel olarak sırasıyla denklem 4.22 ve 4.23'de ifade edilmiştir.

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s] \tag{4.22}$$

Denklem 4.22 durum-değer fonksiyonunun Bellman denklemiyle oluşturulmasını ifade etmektedir. Bu yaklaşımla durum-değer fonksiyonu belirli bir $S_t = s$durumunda belirli bir politikayı π takip ederek $V_\pi(s)$ değerleri hesaplanabilir.

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a] \tag{4.23}$$

Denklem 4.23 aksiyon-değer fonksiyonu ifade edilmektedir. Bu denklem sayesinde yapay ajan, Markov karar süreci ile modellenen çevre içerisinde belirli bir politika π takip ederek, $S_t = s \;\; A_t = a$ uygulaması sonucunda almış olduğu değerleri ifade etmektedir. Denklem 4.22'nin lineer denklem sistemi çözümü denklem 4.24'de ifade edildiği gibidir.

$$\begin{aligned} V_\pi &= R^\pi + \gamma P^\pi V_\pi \\ V_\pi &= (1 - \gamma P^\pi)^{-1} R^\pi \end{aligned} \tag{4.24}$$

Markov ödül sürecinde, denklem 4.16'da ifade edilen değer fonksiyonunun çözümünün oluşturulması gibi, Markov karar sürecinde de denklem 4.24 ile ifade edilen değer fonksiyonunun çözümü vardır. Denklem 4.24 çözümünün uygulanması sonlu sayıda S durumları için P^π geçiş olasılıkları matrisinin bilinmesiyle mümkündür. Geçiş olasılığı matrisinin bilinmemesi durumunda MDP'in yaklaşık çözümleri olan, yaklaşık dinamik programlama metodu, zamansal fark metodu, ve monte carlo metodu yaklaşık çözüm olarak kullanılmaktadır. Markov karar sürecinde ifade edilen durum değer fonksiyonunun 4.22 optimal çözümü denklem 4.25'de ifade edilmektedir.

$$V_*(s) = \max_\pi \; V_\pi(s) \tag{4.25}$$

Denklem 4.25 ile MDP'in optimal değer fonksiyonu $V_*(s)$, tüm politika değerlerinin maksimum olarak seçilmesiyle hesaplanmaktadır. Benzer şekilde optimal aksiyon-değer fonksiyonu denklem 4.26 ile hesaplanmaktadır.

$$q_*(s,a) = \max_\pi \; q_\pi(s,a) \tag{4.26}$$

Denklem 4.25 optimal durum-değer fonksiyonu MDP'deki en iyi sonucu ifade etmektedir. Optimal durum-değer fonksiyonu denklem 4.25 Markov karar sürecinin çözümünü oluşturmaktadır. Optimal durum-değer fonksiyonu ile MDP çözümüne benzer şekilde, denklem 4.26'ile verilen optimal aksiyon-değer fonksiyonunun belirlenmesiyle MDP çözülebilir. Böylece Markov karar süreçlerinin durum-değer ve aksiyon-değer olmak üzere iki farklı çözüm yaklaşımı mevcuttur. Optimal durum-değer fonksiyonu birbirini takip eden politikalar üzerinden denklem 4.27'deki gibi tanımlanmaktadır.

$$\pi \geq \pi' \quad V_{\pi}(s) \geq V_{\pi'}(s) \quad \forall s \tag{4.27}$$

Denklem 4.27'de bir politika değeri π, bir sonraki durumdaki politika değerinden π' büyük eşit ise bu politika değerlerini takip eden durum-değer fonksiyonlarıda tüm durumlar üzerinden büyük eşit olmaktadır. Markov karar sürecinde, optimal bir politika π_* vardır ve bu politika $\pi_* \geq \pi \forall \pi$ sağlamaktadır. Optimal politika değeri, optimal durum-değer fonksiyonunu sağlamaktadır. Bu durum matematiksel olarak denklem 4.28 ile ifade edilmektedir.

$$V_{\pi_*}(s) = V_*(s) \tag{4.28}$$

Denklem 4.28'e benzer şekilde tüm optimal politika değeri optimal aksiyon-değer fonksiyonunu sağlamaktadır. Bu durum denklem 4.29'da matematiksel olarak ifade edilmektedir.

$$q_{\pi_*}(s,a) = q_*(s,a) \tag{4.29}$$

Optimal politika değeri, denklem 4.30 ile $q_*(s,a)$ maksimum yapan değer olarak bulunabilmektedir.

$$\pi_*(a \mid s) = \begin{cases} 1 & a = \arg\max_{a \in A} q_*(s,a) \\ 0 & \text{diğer durumlar} \end{cases} \tag{4.30}$$

Hatırlanacağı üzere bir politika, yapay ajanın s durumunda $a \in A$ aksiyonunu nasıl seçeceğini belirlemektedir. Denklem 4.30 ile ifadesinde optimal politika seçimi belirlenmiştir. Politika değerine bağlı olarak denklem 4.28 optimal durum-değer fonksiyonu ve denklem 4.29 optimal aksiyon-değer fonksiyonları hesaplanabilmektedir. Markov karar süreci için optimal bir politika değeri belirlenebilir. Denklem 4.29 ve 4.28 arasında Bellman'ın özyineli olarak hesaplama bağıntısı denklem 4.31 ile ifade edilir.

$$V_*(s) = \max_a \; q_*(s,a) \tag{4.31}$$

Denklem 4.31, denklem 4.30'da ifade edilen optimal politikayı takip edilerek optimal durum-değer fonksiyonu ile optimal aksiyon-değer fonksiyonlarını birbirine eşitlemektedir. Optimal durum-değer fonksiyonu, Bellman denklemiyle 4.32'deki gibi ifade edilmektedir.

$$V_*(s) = \max_a \; R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_*(s') \tag{4.32}$$

Denklem 4.31 ve 4.32'ye benzer şekilde optimal aksiyon-değer fonksiyonu optimal durum-değer fonksiyonu cinsinden denklem 4.33'deki gibi ifade edilmektedir.

$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_*(s') \tag{4.33}$$

Denklem 4.33 ile denklem 4.31 birleştirerek optimal aksiyon-değer fonksiyonu denklem 4.34'de ifade edildiği gibi yazılmaktadır.

$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} \; q_*(s',a') \tag{4.34}$$

Denklem 4.32 ve 4.34 Markov karar sürecinin durum-değer ve aksiyon-değer fonksiyonlarının iteratif çözümleri için önerilmiştir. Bu denklemlerin çözümleriyle Markov karar sürecinin optimum değerleri hesaplanmaktadır. Bellman'ın optimal denkleminin çözümü lineer olmayan formda olup, kapalı formda analitik çözümleri bulunmamaktadır. Denklemde, yapay ajan $P_{ss'}^a$ bilmemektedir.

Bu durumda denklem 4.34 ve denklem 4.32 iteratif metodlarla çözümlenmektedir. Bir sonraki bölümde Markov karar süreçlerinin iteratif çözümleri olan değer tabanlı ve politika tabanlı yaklaşımlar incelenecektir. (Silver, David.2015)

4.3 Markov Karar Süreci Çözüm

Markov karar sürecini $M = (S, A, P, R, \gamma)$ parametreleriyle matematiksel olarak ifade edilebilir. Burada S durumları, A aksiyonları, P durum geçiş olasılığı matrisini, R ödül fonksiyonunu ve son olarak γ azaltma faktörünü belirtmektedir. Markov karar sürecinde P, R bilinmiyor olabilir veya durumlar sürekli zamanlı olup hesaplanamayabilir. MDP çözümü için denklem 4.34 ve 4.32 tanımlanmıştı, deterministik bir politika $\pi: S \rightarrow A$ durumları aksiyonlara haritalamaktadır. Bu durumda yapay ajanın bir durumda nasıl bir karar alacağı ve uygulayacağı matematiksel olarak tanımlıdır ve deterministiktir. Stokastik bir politika $\pi: S\boldsymbol{x}A \rightarrow [0,1]$ ile tanımlanmakta olup karar vericinin belirli bir durumda uyguladığı kararları belirli bir olasılık değerine haritalamaktadır. MDP'nin çözümü için yapay ajanın belirli bir politikayı izleyerek durum-değer yaklaşımının optimum değerini ve aksiyon-değer yaklaşımının optimum değerini hesaplaması gerekmektedir. Değer fonksiyonunun Bellman denklemi ile genişletilip, ajanın belirli bir durumda, bir politika π takip ederek oluşturacağı değer fonksiyonu denklem 4.35'de görülmektedir.

$$\hat{V}_{\pi}(s) = R(s) + \lambda \sum_{s \in S} P(s'|s, \pi(s))\, \hat{V}_{\pi}(s') \tag{4.35}$$

Optimal durum değer fonksiyonu, optimal politika π_* olarak belirlenir ve denklem 4.36'daki gibi hesaplanır.

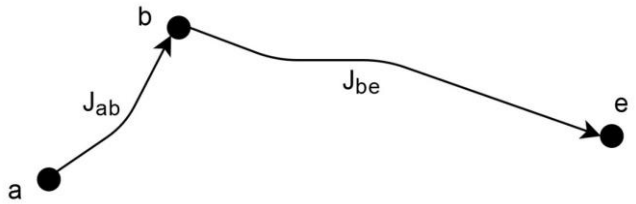
$$\hat{V}_{*}(s) = R(s) + \gamma \max_{a \in A} \sum_{s' \in S} P(s'|s, a) \hat{V}_{*}(s') \tag{4.36}$$

Markov karar sürecinin uygun politika ile değer tabanlı çözümü denklem 4.35 ile ifade edilmektedir. Markov karar sürecinin değer tabanlı çözümü denklem 4.36'da ifade edilmektedir. Denklem 4.36 Markov karar sürecindeki durumların değerleri üzerinden bir çözüm oluşturmaktadır. Dikkat edilecek olursa gelecekteki indirimli ödüllerin toplamını hesaplarken aksiyonların maksimum olduğu değeri işleme koymaktadır. Değer tabanlı yaklaşımda kullanılan politika aç gözlü yaklaşım ($\varepsilon - greedy$) olmaktadır. Denklem 4.35'de politika tabanlı yaklaşımda her bir durumda belirli bir politikayı takip ederek çözüm oluşturulur. Markov karar sürecinde R, P bilinmediği durumlarda yaklaşık çözümler oluşturulmaktadır.

Denklem 4.35 ve 4.36 Markov karar sürecinin, değer tabanlı çözümlerini oluşturmaktadır. Pekiştirmeli öğrenmenin model tabanlı yaklaşımında diğer bir çözüm metodu politika tabanlı çözümlerdir. Bu algoritmalar ajanın bulunduğu s durumlarının değerleriyle ilgilenmek yerine bir davranışı π bulmayı hedeflemektedir. 4.35 ve 4.36 yaklaşımları değer tabanlı yaklaşımlar olup model tabanlı pekiştirmeli öğrenme için önerilen Markov karar sürecinin çözümlerini oluşturmaktadır. Pekiştirmeli öğrenmede $P(s'|s_t, a_t) \forall s_t \in S$ için durum geçiş matrisi bilinmemektedir. Böylece Markov karar süreci değer tabanlı çözüm algoritmaları iteratif olarak çözülmektedir. Çözüm için zamansal fark ve monte carlo metodu kullanılmaktadır. Bu tez çalışmasında zamansal fark yaklaşımıyla durum değer tabanlı ve aksiyon değer tabanlı Markov karar süreci çözümleri yaklaşık olarak hesaplanacaktır. Bu yaklaşımlar değer tabanlı çözüm algoritması olan Q-öğrenme ve Sarsa algoritmalarının temelini oluşturmaktadır. (Silver, David.2015)

4.4 Dinamik Programlama

Sıralı karar verme problemleri, ve optimizasyon problemlerinin iteratif olarak alt parçalara bölünmesi durumunda dinamik programlama kullanılmaktadır. Büyük ve karmaşık problemleri daha küçük ve alt parçalara bölerek, alt parçaların çözümü oluşturulur ve bu çözümler bellekte depolanarak aynı çözüm tekrardan hesaplanmaz. Hesaplanan alt çözümler ile optimal çözüm oluşturulur. Dinamik programlama, diğer matematiksel programlama teknikleri gibi genel bir metoda sahip değildir, her problem için farklı bir uygulaması mevcuttur. Dinamik programlamanın altındaki temel kavram optimallik prensibidir. Dinamik programlama, sıralı karar problemleri ve optimizasyon problerini çözmektedir. İkinci bölümde anlatılan optimal kontrol problemi dinamik programlamayla iteratif olarak alt problemlere ayrıştırılarak çözülebilir. Şekil 4.5'de çoklu karar verme ve optimal yol çıkarma problemi görülmektedir.



Şekil 4.5: Optimal Yol Problemi.

Şekil 4.5'de $a \rightarrow b$ gitmenin maaliyeti J_{ab} olarak, $b \rightarrow e$ gitmenin maaliyeti J_{be} olarak ifade edilirse, optimal maliyet denklem 4.37'deki gibi olcaktır.

$$J_{ae}^{*} = J_{ab} + J_{be} \tag{4.37}$$

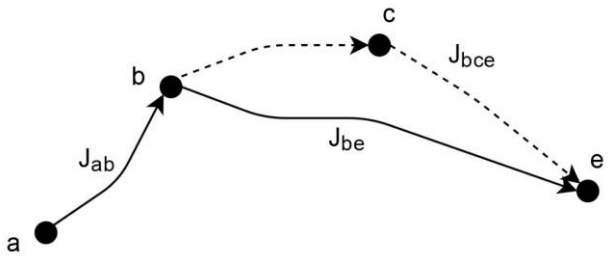
Eğer $a \rightarrow e$ optimal maliyetimiz J_{ae}^{*} ise $b \rightarrow e$ de optimal olmak zorundadır. Eğer $b \rightarrow c \rightarrow e$ yolu $b \rightarrow e$ yolundan daha optimal ise bir ikilem oluşmaktadır. Matematiksel olarak denklem 4.38 ile ifade edilir.

$$J_{bce} < J_{be} \tag{4.38}$$

Denklem 4.38 denklem 4.37 yerine yazılarak denklem 4.39 elde edilir.

$$J_{ab} + J_{bce} < J_{ab} + J_{be} = J_{ae}^{*} \tag{4.39}$$

Denklem 4.39'dan görüleceği üzere optimallik prensibine aykırı bir durum oluşturmaktadır. Eğer J_{ae}^{*}, $a \rightarrow e$ optimal bir yol ise daha optimal bir alt yol oluşturulamaz. Bu durum şekil 4.6'da görülmektedir.



Şekil 4.6: Optimal Yol Problemi Çelişkisi.

Optimallik prensibi ayrık zamanlı sistemler için, $\pi^{*} = \{\pi_0^{*}, \pi_1^{*}, \ldots, \pi_{N-1}^{*}\}$ optimal kararlar dizi olmalıdır. S_t durumları erişilebilir olduğu varsayımıyla yani matematiksel bir modelin oluşturulmasıyla, S_t durumundan t'ye bağlı olarak alt problemlerin geçiş maliyetleri minimum yapılmalıdır. Böylece optimallik prensibi alt optimal problemlere bölünmektedir. Dinamik programlama oluşturulan alt optimal problemlerin çözümü için ileri yönlü ve geriye doğru olarak iki farklı biçimde uygulanmaktadır.

Dinamik programlama durumlar arasındaki geçişler deterministik ise deterministik dinamik programlama, durumlar arası geçişler rastsal ise stokastik dinamik programlama olarak isimlendirilmektedir. Dinamik programlama sıralı karar verme problemlerinde kullanılmakta olup model tabanlı pekiştirmeli öğrenme problemlerinde kullanılmaktadır. Dinamik programlama ile Markov karar sürecinin çözülmesi için durum geçiş matrisinin bilinmesi gerekmektedir. (Sutton , Richard S. ve diğ 2018)

4.5 Bellman Denklemi

Modelden bağımsız pekiştirmeli öğrenmede ajanın içinde bulunduğu ortamı Markov karar süreciyle modellenir. Amacımız Markov karar sürecinin çözümü olan optimal politika veya optimal değer fonksiyonlarını hesaplamaktır. Optimal değer fonksiyonu $V_*(s)$ olarak denklem 4.25 ile ifade edilmektedir. Yapay ajanın optimal değer ifadesini hesaplaması için her bir durumun değerlerini oluşturması gerekmektedir. Bu durum denklem 4.40'daki gibidir.

$$V(s) = E[G_t|S_t = s] \tag{4.40}$$

Denklem 4.40'da $G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ ifade etmekte olup gelecekteki ödül değerlerini ifade etmektedir. Denklem 4.40'ın hesaplamasında iteratif bir yaklaşım oluşturan Bellman denklemi, denklem 4.41'deki gibidir.

$$\begin{aligned} V(s) &= E[G_t|S_t = s] \\ &= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots |S_t = s] \\ &= E[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots)|S_t = s] \\ &= E[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= E[R_{t+1} + \gamma V(S_{t+1})|S_t = s] \end{aligned} \tag{4.41}$$

Denklem 4.41 ile durum değer fonksiyonu ifade edilmektedir. Denklem 4.41'in beklenen değer ifadisi $E[]$bir sonraki durumlarla ifade edilmektedir. $E[R_{t+1}] \rightarrow R_{t+1}$ olmaktadır böylece beklenen değer sadece denklemin sağ tarafındaki $E[\gamma V(S_{t+1})]$ terim ile hesaplanmaktadır. Beklenen değer yani ortalama hesabının denklem 4.41'e eklenerek iteratif olarak denklem 4.42 elde edilmektedir. Bu denklem Bellman denklemi olarak ifade edilmektedir.

$$V(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} V(s') \quad (4.42)$$

Denklem 4.42 her bir durumun gelecekteki azaltılmış ödül değerleriyle birlikte durum-değer fonksiyonu yazılmaktadır. Bellman denklemi aksiyon değer fonksiyonuyla birlikte denklem 4.43'deki gibi ifade edilmektedir.

$$q(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a q(s', a') \quad (4.43)$$

Denklem 4.43 ve denklem 4.42 sayesinde Markov karar süreciyle modellenen ortam içerisinde değer tabanlı pekiştirmeli öğrenme yaklaşımı iteratif olarak hesaplanabilir. Modelden bağımsız pekiştirmeli öğrenme yaklaşımında $P_{ss'}, P_{ss'}^a$ durum geçiş olasılıkları matrisi bilinmemektedir. Yapay ajanın bu durumda Markov karar sürecinin çözümünü oluşturamaz. Zamansal fark veya monte karlo yaklaşımıyla denklem 4.43 ve denklem 4.42 yaklaşık olarak hesaplanabilir. Bu tez çalışmasında zamansal fark algoritmalarıyla denklem 4.42 ve denklem 4.43'ün iteratif çözümleri incelenecektir. (Silver, David.2015)

4.6 Pekiştirmeli Öğrenme Politika Tabanlı Çözüm

Pekiştirmeli öğrenmenin politika tabanlı çözümü, MDP'ye değer tabanlı yaklaşıma alternatif olarak durumların değerlerini oluşturmak yerine bir politika bulmayı hedeflemektedir. Bir önceki bölümde MDP'nin çözümü için durum-değer ve aksiyon-değer fonksiyonları incelenmişti. Bu yaklaşımlarda yapay ajanın ortam içerisinde almış olduğu ödüllerle $V_\pi(s)$, $q_\pi(s,a)$ fonksiyonları güncellendi. Güncelleme işlemi için ajanın oluşturacağı bir politika incelenmedi. Örneğin durum değerlerini maksimum yapan bir politika seçimi yapıldı. Deterministik bir politika için bu durum denklem 4.30'de ifade edilmektedir. Pekiştirmeli öğrenmenin çözümü için önerilen politika tabanlı yaklaşım, yapay ajanın ortam içerisinde bir davranış geliştirmesini sağlamaktadır. Ajanın davranış biçimi determinisitik veya stokastik olabilir. Deterministik bir politika denklem 4.44'de görülmektedir.

$$\pi_\theta(s) = \max_{a \in A} \theta(s,a) \quad (4.44)$$

Denklem 4.44'e benzer şekilde politikanın stokastik olması durumunda denklem 4.45 ile ifade edilen politika oluşturulabilir.

$$\pi_\theta(a|s) = \frac{exp\,\theta\,(s,a)}{\sum_{a'\in A} exp\,\theta\,(s,a')} \tag{4.45}$$

Denklem 4.44 ve 4.45 ile oluşturulan politika, Markov karar sürecinin çözümü için yazılan denklem 4.30 ile ifade edilen politikadan farklıdır. Dikkat edilmelidir ki θ burda keyfi bir parametredir. Bir politikanın değeri denklem 4.22 ile azaltılmış gelecek ödüllerin beklenen değerini ifade etmektedir. Bu durum denklem 4.46'da görülmektedir.

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s] \tag{4.46}$$

Pekiştirmeli öğrenmenin politika tabanlı çözümü için oluşturulan optimizasyon problemi denklem 4.46 genişletilerek denklem 4.47'deki gibi ifade edilmektedir.

$$V_\theta(s) = E[\sum_{t=1}^{\infty} \gamma^t r_t|s_{t+1}\sim P(s'|s_t,\pi_\theta(s_t)), s_1 = s] \tag{4.47}$$

Denklem 4.47'den ifade edildiği gibi politika uzun bir ufuk boyunca hesaplanabilmektedir. Denklem 4.47 ile ifade edilen politika değerinin analitik olarak gradyanı $\nabla_\theta V_\theta(s)$ bilinmemektedir. Optimizasyon probleminin çözülmesi için $\nabla_\theta V_\theta(s)$ ifadesinin yaklaşık çözümü gerekmektedir. Basit bir politika arama algoritması tablo 4.1'deki gibidir.

Tablo 4.1: Basit Politika Arama Algoritması.

ALGORİTMA : Basit Politika Arama Algoritması			
1	**:**	$\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(M)}$	*M deneme için parametrelere rastgele gürültü ekle ve ampirik olarak ödül toplamını* $V^{(i)} = \sum_{t=1}^{T} \gamma^t r_t$ *hesapla.*
2	**:**	$V^{(i)} \approx f(\theta^{(i)})$	*herhangi bir parametrik makine öğrenmesi modelini danışmanlı öğrenme modeline göre eğit.*
3	**:**	$\theta \leftarrow \theta + \alpha \nabla_\theta f(\theta)$	*parametreleri güncelle.*

Tablo 4.1 ile ifade edilen basit politika arama algoritmasının dışında diğer bir politika arama algoritması politika gradyanı ve takviye (reinforce) metodudur. Bu metod birçok politika tabanlı pekiştirmeli öğrenme algoritmasının temelini oluşturmaktadır.$\tau = (s_1, a_1, s_2, a_2, \ldots, s_T, a_T)$ durum aksiyon çifti olarak izlediğimiz yörüngeyi ifade etmek üzere politikanın değer fonksiyonu denklem 4.48'deki gibi ifade edilebilir.

$$V_\theta(s) = E[R(\tau); \theta] = \int p(\tau; \theta)\, R(\tau) d\tau \tag{4.48}$$

Takviye algoritmasının oluşturulması için denklem 4.48'in düzenlenmesi gerekmektedir. Bu düzenleme öncelikle denklem 4.48'in gradyan hesabı üzerinden denklem 4.49'daki gibi ifade edilmektedir.

$$\begin{aligned} \nabla_\theta V_\theta(s) &= \nabla_\theta \int p(\tau; \theta)\, R(\tau) d\tau \\ &= \int \nabla_\theta p(\tau; \theta)\, R(\tau) d\tau \\ &= \int \frac{p(\tau; \theta)}{p(\tau; \theta)} \nabla_\theta p(\tau; \theta)\, R(\tau) d\tau \\ &= \int p(\tau; \theta) \nabla_\theta \log p\,(\tau; \theta)\, R(\tau) d\tau \\ &= E[\nabla_\theta \log p\,(\tau; \theta) R(\tau)] \end{aligned} \tag{4.49}$$

Denklem 4.49 ile $V_\theta(s)$ fonksiyonunun gradyanı $\nabla_\theta V_\theta(s)$ hesaplanablir. Denklem 4.49'a dikkat edilmelidir ki bir beklenen değer hesabı yapılmakta böylece örneklem üzerinden bir tahmin oluşturulmaktadır. Politika arama için önerilen politika gradyanı ve takviye algoritmasının ikinci düzenlemesi denklem 4.49 üzerinden yapılmaktadır. Bu durum denklem 4.50'de ifade edilmektedir.

$$\nabla_\theta \log p\,(\tau;\theta) = \nabla_\theta \log\Big(\prod_{t=1}^{T} p(s_{t+1}|s_t,a_t)\pi_\theta(a_t|s_t)\Big)$$
$$= \nabla_\theta \sum_{t=1}^{T} (\log p\,(s_{t+1}|s_t,a_t)) + \nabla_\theta \log \pi_\theta\,(a_t|s_t)$$
$$= \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\,(a_t|s_t) \tag{4.50}$$

Denklem 4.50 kullanılarak politika optimizasyonu algoritması tablo 4.2'de görülmektedir. Denklem 4.50'de gradyan geçiş olasılıklarına bağlı olmadığı için, eşitliğin sağındaki ilk kısım sıfır olmaktadır.

Tablo 4.2: Politika Gradyanı ve Takviye Algoritması.

ALGORİTMA : Politika Gradyanı ve Takviye Algoritması		
1	**:**	*Stokastik bir politika* π_θ *için M adımda* τ *yörüngeyi oluşturun.*
2	**:**	*Gradyan değerini yaklaşık olarak hesaplayın.* $g_\theta \leftarrow \frac{1}{M}(\sum_{i=1}^{M}(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\,(a_t^{(i)}|s_t^{(i)}))\,R(\tau^{(i)}))$
3	**:**	*Parametreleri güncelle.* $\theta \leftarrow \theta + \alpha g_\theta$

Tablo 4.2'de önerilen algoritma, gradyan'ın örneklem üzerinden hesaplanması olup gerçek gradyan değerini hesaplamamaktadır. Dikkat edilmelidir ki ajanın ortam içindeki örnekleminin çok iyi olup politikayı en iyi şekilde belirlemesi gerekmektedir. Modelden bağımsız pekiştirmeli öğrenme yaklaşımında ortam hakkında bir bilgi oluşmadığı için ajanın ortamı keşfetmesi ve sonrasında keşiflerinden edindiği tecrübeleri kullanması gerekmektedir. Değer tabanlı veya politika tabanlı algoritmalar ortamı bir süre gezip bir bilgi toplaması gerekmektedir. Bir önceki bölümde ifade edilen $\varepsilon - greedy$ yaklaşımı denklem 4.51 ile pekiştirmeli öğrenme algoritmaları için kullanılmaktadır. (Kolter, J. Zico. 2016)

$$\pi(s) = \begin{cases} \max\limits_{a\in A} \hat{Q}(s,a) & 1-\varepsilon \\ \text{rastgele aksiyon} & \text{diğer durumlar} \end{cases} \tag{4.51}$$

4.7 Pekiştirmeli Öğrenme Değer Tabanlı Çözüm

Markov karar süreci, pekiştirmeli öğrenme için matematiksel bir altyapı sağlamaktadır. Markov karar süreci bir önceki bölümde detaylıca incelenmiştir. Markov karar sürecinin çözümü optimal durum-değer veya aksiyon-değer fonksiyonlarının çözülmesiyle elde edilmektedir. Bu durum denklem 4.52'de ifade edilmektedir.

$$V_*(s) = \max_a \; R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_*(s')$$
$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} \; q_*(s', a') \qquad (4.52)$$

Denklem 4.52'deki $V_*(s)$ durum-değer fonksiyonunun optimal çözümünü ifade etmekte olup MDP'nin çözümünü oluşturmaktadır. Benzer şekilde $q_*(s,a)$ aksiyon-değer fonksiyonunun optimal bir çözümü olup MDP'nin çözümüdür. Denklem 4.52 ile ifade edilen yaklaşımlarda yapay ajan MDP ile modellenen ortam içerisinde nasıl bir ödül değeri alacağını ve bu ödül değerlerine göre fonksiyonları nasıl güncelleyeceği ifade edilmektedir. Bu çözüm yaklaşımları MDP'nin değer tabanlı çözümlerini oluşturmaktadır. Denklem 4.52'de ifade edilen $P_{ss'}^a, R_s^a$bilinmemektedir. Durum geçiş matrisi ve ödül fonksiyonu bilinmediği için MDP'nin kapalı formda çözümü hesaplanamamaktadır. Yapay ajan bulunduğu ortam içerisinde gözlemlerde bulunarak s_{t+1} durumunu $P(s'|s_t, a_t)$ dağılım fonksiyonundan örneklemiş olmaktadır. Elimizde var olan bu gözlem değerleri ve r_t ortam içerisinde durumlar üzerinden yazılan ödül fonksiyonu ile yaklaşık hesaplama yapılmaktadır. Bu durum denklem 4.53'de ifade edilmektedir.

$$\hat{V}^\pi(s_t) = r_t + \gamma \hat{V}^\pi(s_{t+1}) \qquad (4.53)$$

Denklem 4.53 s_t durumunun değerini t anındaki ödül ve bir sonraki s_{t+1} durumu ve bir azaltma faktörüyle $r_t + \gamma \hat{V}^\pi(s_{t+1})$'deki gibi güncellemektedir. Bu durumdaki güncelleme bir sonraki durum değerinin pozitif veya negatif olmasına karşılık t anındaki durumu etkilemektedir. Denklem 4.53'e güncelleme yaparak daha etkili bir güncelleme kuralı olan denklem 4.54 elde edilmektedir.

$$\hat{V}^{\pi}(s_t) = (1-\alpha)\hat{V}^{\pi}(s_t) + \alpha(r_t + \gamma\hat{V}^{\pi}(s_{t+1}))$$
$$\hat{V}^{\pi}(s_t) = \hat{V}^{\pi}(s_t) + \alpha(r_t + \underbrace{\gamma\hat{V}^{\pi}(s_{t+1}) - \hat{V}^{\pi}(s_t)}_{Zamansal\ Fark})$$
$$\alpha < 1 \qquad (4.54)$$

Denklem 4.54'e dikkat edilecek olursa t anındaki $\hat{V}^{\pi}(s_t)$ değerlerinin güncellemesi gene $\hat{V}^{\pi}(s_t)$'ye ilave olarak zamansal fark kısmının eklenmesiyle oluşmaktadır. Zamansal fark yaklaşımında ajanın almış olacağı gelecekteki indirimli ödül değerine $\gamma\hat{V}^{\pi}(s_{t+1})$ bağlı olarak güncelleme gerçekleşecektir. $\gamma\hat{V}^{\pi}(s_{t+1}) - \hat{V}^{\pi}(s_t)$ ifadesi zamansal fark olarak isimlendirilmekte olup gelecekteki $\hat{V}^{\pi}(s_{t+1})$ değeriyle, şimdiki $\hat{V}^{\pi}(s_t)$ arasında bir denge kurmaktadır. Gelecekte tahmin edilen değer ile şimdiki değer arasındaki fark ne kadar iyi bir karar verdiğimizi ifade etmektedir. Böylece ajanın t anında elde etmiş olduğu ödül değerleri denklem 4.54 üzerinden denklem 4.53'e göre çok daha başarılı bir şekilde güncellenecektir. Denklem 4.53'deki güncelleme denklem 4.54'e göre çok daha başarısız olmasının sebebi içinde bulunduğumuz durum ödülü ve tahmin edilen gelecekteki ödül ile güncelleme yapıyor olmamızdır. Denklem 4.54'de ise içinde bulunduğumuz durum değeri güncellenirken, içinde bulunduğumuz durum değerine ilave olarak gelecekte ne kadar doğru kararlar verdiğimizi ifade eden zamansal fark ile güncellenmektedir. Denklem 4.54 modelden bağımsız pekiştirmeli öğrenme yaklaşımının zamansal fark yaklaşımıyla çözülmesini sağlamaktadır. Zamansal fark yaklaşımının aksiyon-değer ve durum-değer yaklaşımlarına uygulanmaktadır. Zamansal fark algoritması tablo 4.3'de ifade edilmektedir.

Tablo 4.3: Zamansal Fark Algoritması.

ALGORİTMA : Zamansal Fark Algoritması			
1	**:**	$\hat{V}^{\pi}(s) = 0 \forall s \in S$	*Algoritma başlangıç parametresi.*
2	**:**	$t \leftarrow 0$	
3	**:**	s_t, r_t	*Durumları ve ödül değerlerini gözlemle.*
4	**:**	$a = \pi(s)$	*Belirli bir politika ile bir aksiyon uygula.*
5	**:**	s_{t+1}	*Bir sonraki durumu gözlemle.*
6	**:**	$\hat{V}^{\pi}(s_t) = \hat{V}^{\pi}(s_t) + \alpha(r_t + \gamma\hat{V}^{\pi}(s_{t+1}) - \hat{V}^{\pi}(s_t))$	*Zamansal Fark ile güncelleme yap.*
7	**:**	$t \leftarrow t + 1$	

Zamansal fark algoritması MDP'yi oluşturmadan $\hat{V}^{\pi}(s_t) \approx V^{\pi}(s_t)$ yaklaşık olarak hesaplamamızı sağlar. (Kolter, J. Zico. 2016)

4.7.1 Q Öğrenme Algoritması

Q öğrenme algoritması zamansal fark algoritmasıyla beraber aksiyon-değer tabanlı bir algoritma olup kapalı politikadır. Hatırlanacağı üzere kapalı politika yaklaşımında ajanın takip ettiği politika değeri güncellenmemektedir. Q öğrenme algoritması değer tabanlı yaklaşıma benzerlik göstermektedir. Değer tabanlı yaklaşıma alternatif olarak Q öğrenme algoritmasında aksiyon ve durumlar üzerinden denklem 4.55'deki gibi ifade edilmektedir.

$$Q^{\pi}(s,a) = R(s) + \gamma \sum_{s' \in S} P(s'|s,a) Q^{\pi}(s', \pi(s')) \tag{4.55}$$

Değer tabanlı çözüm algoritması olan Q öğrenme algoritması denklem 4.55'de görüldüğü gibidir. Denklemde ajanın takip ettiği politika $\pi(s')$ ile ifade edilmektedir. Denklem 4.55'in optimal çözümü denklem 4.56'daki gibidir.

$$Q^{*}(s,a) = R(s) + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A} Q^{*}(s', a') \tag{4.56}$$

Denklem 4.56'da ajanın sahip olduğu politika $\pi(s')$ Q değerleri üzerinden maksimum değeri üreten aksiyon seçimi olarak oluşturulabilir. Denklem 4.56 Q öğrenme algoritmasının temelini oluşturmaktadır. Böylece ajan MDP içerisinde durum ve aksiyon değer fonksiyonuyla ifade edilmektedir. Denklem 4.56'da $P(s'|s,a), R(s)$ bilinmediği durumda, yani MDP tam olarak bilinmiyor sadece t anındaki ölçümler elimizde varsa, MDP'nin çözümü olan optimal durum aksiyon değer fonksiyonunun $Q^{*}(s,a)$ bulunması için denklem 4.56'nın zamansal fark yaklaşımıyla güncellenmesi gerekmektedir. Bu durum denklem 4.57'de ifade edilmektedir.

$$\begin{aligned} \hat{Q}^{*}(s,a) &= (1-\alpha)\hat{Q}^{*}(s,a) + \alpha(r + \gamma \max_{a' \in A} \hat{Q}^{*}(s', a')) \\ \hat{Q}^{*}(s,a) &= \hat{Q}^{*}(s,a) + \alpha(r + \underbrace{\gamma \max_{a' \in A} \hat{Q}^{*}(s', a') - \hat{Q}^{*}(s,a)}_{ZamansalFark}) \end{aligned} \tag{4.57}$$

Denklem 4.57 ile oluşturulan Q öğrenme algoritması, ajanın bulunduğu ortam içerisinde yeterince durum ve aksiyonları uygulamasıyla gerçek değerlerine $Q^{*}(s,a) \approx \hat{Q}^{*}(s,a)$ yakınsamaktadır. Q öğrenmenin bir avantajı MDP elimizde bulunmadan aksiyon seçimleri gerçekleştirerek optimal politika değerlerini öğrenebiliyor olmamızdır.

Bu durum denklem 4.58'de ifade edilmektedir.

$$\pi^*(s) = \arg\max_{a \in A} \hat{Q}^*(s,a) \tag{4.58}$$

Denklem 4.57 ile ifade edilen Q öğrenme algoritması tablo 4.4'de görülmektedir.

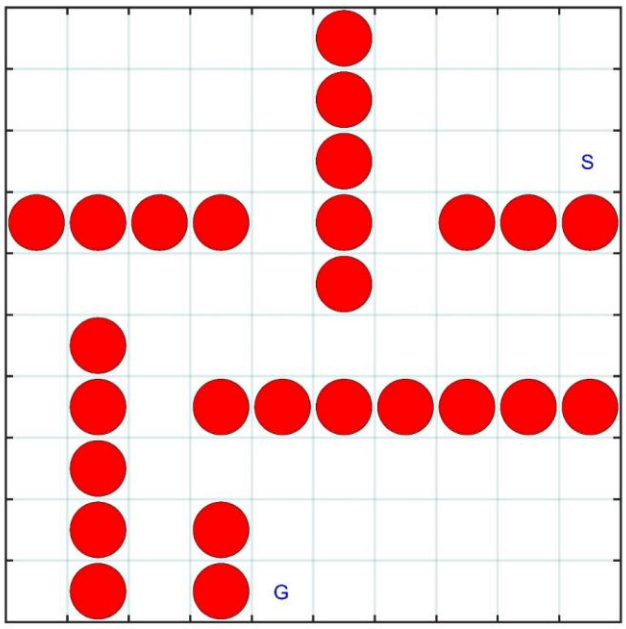
Tablo 4.4: Q Öğrenme Algoritması.

ALGORİTMA : Q Öğrenme Algoritması		
1	**:**	$\alpha \in (0,1], \gamma$ *Öğrenme oranı ve azaltma parametresini belirle.*
2	**:**	$\varepsilon > 0$ *Rastgele aksiyon seçimi için epsilon çok küçük bir sayı.*
3	**:**	$Q(s,a) = 0 s \in S, a \in A$ Q *değerinin başlangıcı aksiyon ve durumlar için sıfır.*
4	**:**	$Eps = 1000$ *Algoritmanın kaç bölüm çalışacağı.*
5	**:**	$T = 100$ *Algoritmanın her bölüm içerisinde kaç adım atacağı.*
6	**:**	$Eps \leftarrow 0$
7	**:**	s *Ajanın başlangıç durumu.*
8	**:**	$T \leftarrow 0$
9	**:**	ε *olasılıkla rastgele* a t
10	**:**	$1-\varepsilon$ *olasılıkla* $a = \max_{a \in A} Q(s,a)$ *aksiyon seç.*
11	**:**	a *aksiyon uygula,* s', r *bir sonraki durum ve ödülü kaydet.*
12	**:**	$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a' \in A} Q(s', a') - Q(s,a))$ Q *tablosunu güncelle*
13	**:**	$s \leftarrow s'$

Tablo 4.4'de ifade edilen Q öğrenme algoritması sonlu ayrık durum ve aksiyon çifti için uygulanabilir olmaktadır. Kontrol teorisinde durumlar ve aksiyonlar sürekli zamanlı olduğu için Q öğrenme algoritması sürekli durum ve aksiyon çifti için uygulanmamaktadır. (Kolter, J. Zico. 2016)

4.7.1.1 Q Öğrenme Algoritmasıyla Robot Yörünge Planlaması

Mobil bir robotun içinde bulunduğu çalışma uzayında yörünge planlaması problemi önemli bir konudur. Robot bulunduğu ortam içerisinde sabit veya hareketli engeller bulunmasına göre en kısa yol veya minimum enerji gibi kısıtları sağlamak üzere bir yol planlaması gerçekleştirmelidir. Bu bölümde mobil bir robotun içinde bulunduğu ortamda sabit engellerin bulunduğu ve robotun belirli bir başlangıç noktasından hedef rotaya ulaşması için en optimal yolu oluşturması hedeflenmektedir. Bu problem literatürde yörünge planlaması olarak geçmektedir ve bu problemin çözümü için birçok optimizasyon metodu önerilmektedir. Örneğin Astar, Potansiyel alan vb algoritmalar sayesinde robot en kısa yol problemini çözmektedir. Bu bölümde aynı problemi Q öğrenme algoritmasıyla robotun bulunduğu çalışma ortamını ayrık zamanlı durumlar olarak kabul ederek çözüm yapılacaktır. Şekil 4.7'de robotumuzun s başlangıç durumundan, g hedef durumuna en kısa yoldan gitmesi hedeflenmektedir.



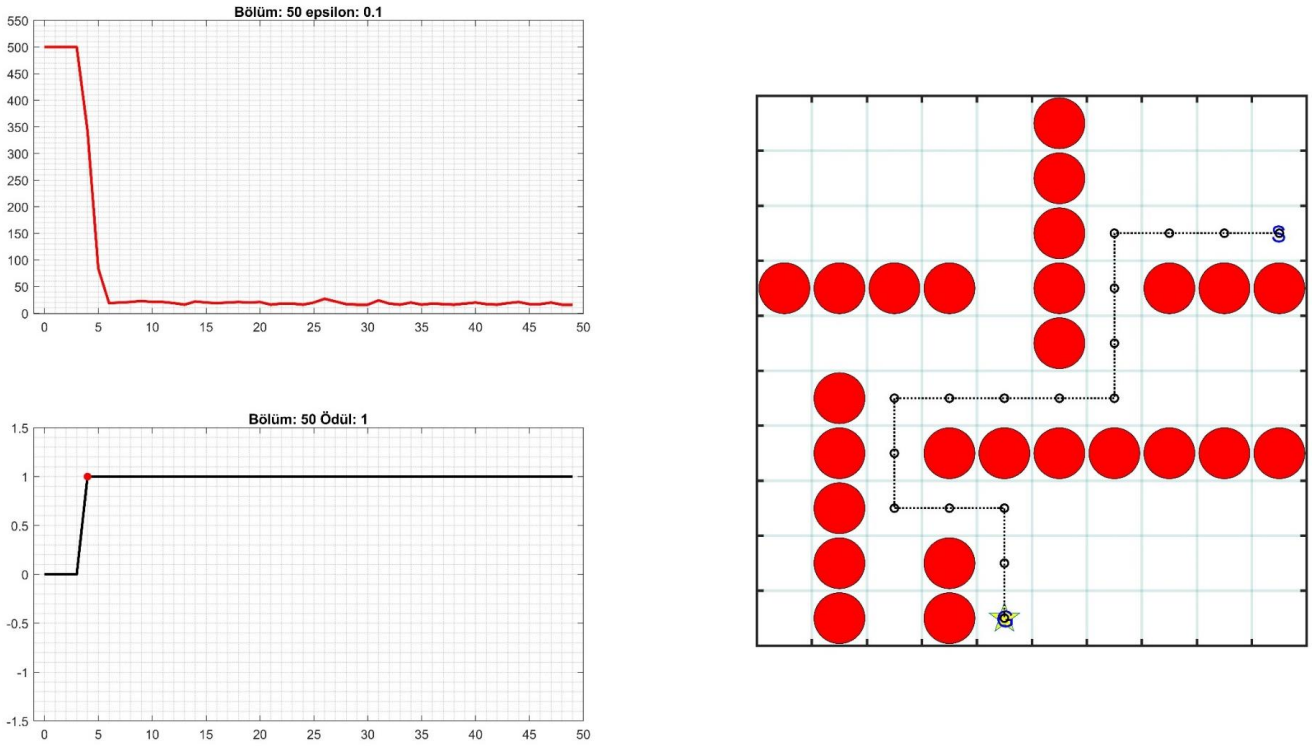
Şekil 4.7: Robot Yörünge Planlama Problemi.

Robotumuz ortam içersinde hiçbir bilgiye sahip olmadığı için Q öğrenme algoritmasıyla ortamı modelleyip çözüm oluşturabiliriz. Robotumuzun bulunduğu durum uzayını yüz durumla 10x10 matris olarak ifade etmekteyiz. Ajanın gerçekleştirebilceği dört aksiyon $a \in (\uparrow, \downarrow, \rightarrow, \leftarrow)$ bulunmaktadır.

Ajanın amacı g hedef noktasında oluşturulan ödülü elde etmek için en kısa yolu oluşturmasıdır. Ajanın ödül fonksiyonu denklem 4.58'de belirtildiği gibidir.

$$r_t = \begin{cases} 1 & s_t = hedef \\ 0 & diğer\ durumlar \end{cases} \tag{4.58}$$

Denklem 4.58'de verilen ödül için ajanın hedef duruma geldiği zaman 1 birim ödül kazanmaktadır. Q-Öğrenme algoritmamız 4 bölümde öğrenme işlemini tamamlayıp robot için en uygun yolu oluşturmuştur. Şekil 4.8'de robot için en uygun yörünge görülmektedir.



Şekil 4.8: Robot Yörünge Planlama Problemi Q Öğrenme İle Çözümü.

4.7.2 Sarsa Algoritması

Q öğrenme algoritması kapalı politika algoritmadır, politika değeri değişmemekte olup denklem 4.51'deki gibi sabit bir politikayı takipetmektedir. Q öğrenme algoritması sabit bir politika altında durum-aksiyon değerleri üzerinden Markov karar sürecini yaklaşık olarak çözmektedir. Sarsa algoritması Q öğrenme algoritmasına çok benzerlik göstermektedir. Matematiksel olarak Sarsa algoritması denklem 4.59'daki gibi ifade edilmektedir.

$$\hat{Q}^{\pi}(s,a) = (1-\alpha)\hat{Q}^{\pi}(s,a) + \alpha(r + \gamma\hat{Q}^{\pi}(s^{'},\pi(s^{'})))$$
$$\hat{Q}^{\pi}(s,a) = \hat{Q}^{\pi}(s,a) + \alpha(r + \underbrace{\gamma\hat{Q}^{\pi}(s^{'},\pi(s^{'})) - \hat{Q}^{\pi}(s,a)}_{ZamansalFark}) \qquad (4.59)$$

Denklem 4.59'a dikkat edilirse Q değerleri belirli bir politika $\pi(s)$ üzerinden güncellenmektedir. Q öğrenme algoritmasında sabit olan politika, Sarsa algoritmasında parametrik olmaktadır. Bu yaklaşım Sarsa algoritmasını açık politika bir algoritma olmasını sağlamaktadır. Q öğrenme algoritmasında bir önceki adımda takip edilen politika değerleri üzerinden güncelleme yapılırken, Sarsa öğrenme algoritmasında bir sonraki Q değerleri belirli bir politika üzerinden hesaplanıp güncelleme sağlanır. Q öğrenmede güncelleme için kullanılan yeni Q değerleri $\max_{a^{'}\in A} Q(s^{'},a^{'})$ ile aç gözlü bir yaklaşımla hesaplanmaktadır. Sarsa öğrenme algoritmasında bu durum yeni Q değerleri için $\hat{Q}^{\pi}(s^{'},\pi(s^{'}))$ bir politika takip edilerek hesaplanmaktadır. Bu yaklaşım Q öğrenme ve Sarsa öğrenme arasındaki temel farkı oluşturmaktadır. Sarsa algoritması tablo 4.5'de ifade edilmektedir.

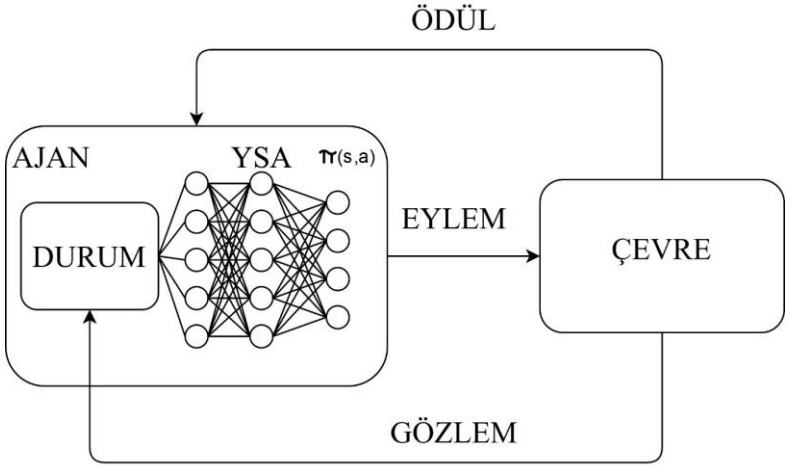
Tablo 4.5: Sarsa Algoritması.

ALGORİTMA : Sarsa Algoritması		
1	**:**	$\alpha \in (0,1], \gamma$ *Öğrenme oranı ve azaltma parametresini belirle.*
2	**:**	$\varepsilon > 0$ *Rastgele aksiyon seçimi için epsilon çok küçük bir sayı.*
3	**:**	$Q(s,a) = 0 s \in S, a \in A$ Q *değerinin başlangıcı aksiyon ve durumlar için sıfır.*
4	**:**	$Eps = 1000$ *Algoritmanın kaç bölüm çalışacağı.*
5	**:**	$T = 100$ *Algoritmanın her bölüm içerisinde kaç adım atacağı.*
6	**:**	$Eps \leftarrow 0$
7	**:**	s *Ajanın başlangıç durumu.*
8	**:**	a *Ajanın* s *durumu için alması gereken aksiyonu belirliyoruz*
9	**:**	$T \leftarrow 0$
10	**:**	a *Aksiyonunu uygula* ve $r, s^{'}$ *kaydet.*
11	**:**	$a^{'}$, *Ajanın* $s^{'}$ *durumu için alması gereken aksiyonu belirliyoruz*
12	**:**	$Q(s,a) = Q(s,a) + \alpha(r + \gamma Q(s^{'},a^{'}) - Q(s,a))$ Q *tablosunu güncelle*
13	**:**	$s \leftarrow s^{'}, \quad a \leftarrow a^{'}$

Sarsa ve Q öğrenme arasında temel fark, Q öğrenme optimal yolu oluşturmaya çalışırken, Sarsa öğrenme optimal yolu en güvenli şekilde oluşturmayı hedeflemektedir. Q öğrenme sadece maksimum değerler üzerinden hareket ederken Sarsa öğrenme algoritması maksimum değerleri gelecekteki durumlar ve aksiyonlarını güncelleyerek yapmaktadır. Böylece Sarsa öğrenme bir sonraki kararları güncel politika üzerinden hesaplamaktadır. Q öğrenmede bir sonraki kararlar sadece maksimum değerler üzerinden hesaplanmaktadır. (Sutton , Richard S. ve diğ 2018)

4.8 Derin Pekiştirmeli Q Öğrenme Algoritması

Bir önceki bölümde ifade edilen Q öğrenme ve Sarsa öğrenme algoritmaları literatürde tablo metodu olarak isimlendirilmektedir. Bu metodlar ayrık durum ve aksiyon uzayı için uygulanabilirdir. Gerçek hayatta kontrol probleminin, durum uzayı ve kontrol işareti süreli zamanlı olarak ifade edilmektedir. Robot kontrol probleminde, kontrol işaretimiz olan tork değerleri $\tau \in \Re^N$, N serbestlik dereceli sistem için reel sayılar uzayında tanımlı olup sonsuz değer almakta ve robotun eklem değişkenleri olan $q, \dot{q}, \ddot{q} \in \Re^N$, N serbestlik dereceli sistem için reel sayılar uzayından yani sonsuz değerler almaktadır. Pekiştirmeli öğrenmede pratik bir uygulama olarak sürekli durum ve aksiyon uzaylarına ayrıklaştırma yaparak Q öğrenme ve Sarsa öğrenme algoritmaları uygulanabilir, fakat pratikte ayrıklaştırmadan dolayı oluşacak hatalar kontrol uygulamalarında sorun teşkil etmektedir. Pekiştirmeli öğrenmenin tablo metodunun bir diğer problemi hesaplama maliyetini oluşturmaktır. Ayrık durum ve aksiyon uzayımızın boyutunun artması hesapsal karmaşıklığa ve hesapmala maliyetinin artmasına sebep olmaktadır. Makine öğrenmesinde kullanılan derin yapay sinir ağları evrensel fonksiyon yaklaşıklayıcıdır. Yapay sinir ağlarının fonksiyon yaklaşıklayıcı özelliği ile pekiştirmeli öğrenmenin Q öğrenme algoritması birleştirilerek derin pekiştirmeli öğrenme yöntemleri geliştirilmiştir. Derin pekiştirmeli öğrenme algoritması, derin yapay sinir ağları ve Q öğrenme algoritmasının birleştirilmesiyle elde edilmektedir. Derin pekiştirmeli öğrenme algoritmasıyla sürekli durumların bulunduğu kontrol problemleri çözülmektedir. Derin pekiştirmeli öğrenme algoritmasının dezavantajı kontrol probleminde kontrol işaretimiz $u(t)$, zamanın sürekli bir fonksiyonudur. Kontrol işaretimizin pekiştirmeli öğrenmedeki karşılığı olan aksiyonlarımız $a(t)$ zamanın birer sürekli fonksiyonu olarak kontrol problemlerimizde karşımıza çıkmaktadır. Aksiyonlarımızın ayrık değerler alması sebebiyle, pekiştirmeli öğrenme ile kontrol problemlerini çözmek için önerilen derin Q öğrenme algoritması yeterli sonuç üretememektedir. Derin Q öğrenme algoritmasının blok diyagramı şekil 4.9'da görülmektedir.



Şekil 4.9: Derin Q Öğrenme Algoritması.

Derin pekiştirmeli öğrenme algoritması 2015 yılında önerilmiştir. Algoritmanın matematiksel temellerine geçmeden önce zamansal fark ile oluşturulan Q öğrenme algoritmasını inceleyelim. Denklem 4.60'da Q öğrenme algoritması görülmektedir.

$$Q(s,a) = Q(s,a) + \alpha(r_s + \gamma \max_{a \in A} Q(s',a') - Q(s,a)) \tag{4.60}$$

Denklem 4.60 iteratif olarak güncellenmektedir. Sürekli durum değerleri için iteratif çözüm mümkün olmamaktadır. Derin Q öğrenme algoritmasının temelini fonksiyon yaklaşıklama oluşturmaktadır. Fonksiyon yaklaşıklama derin yapay sinir ağlarıyla oluşturulmaktadır. Bu durum denklem 4.61'de ifade edilmektedir.

$$Q(s,a:\theta) \approx Q^*(s,a) \tag{4.61}$$

Denklem 4.61'de θ parametresi derin yapay sinir ağlarının parametrelerini oluşturmaktadır. Denklem 4.60'ı denklem 4.62'deki gibi tekrar oluşturalım.

$$Q(s,a) = Q(s,a) + \alpha(\underbrace{\underbrace{r_s + \gamma \max_{a \in A} Q(s',a')}_{Hedef} - \underbrace{Q(s,a)}_{Tahmin}}_{Maliyet}) \tag{4.62}$$

Denklem 4.62'yi iki parçaya ayırmaktayız. Birinci parça hedef, diğer parça tahmin edileni oluşturmaktadır. Bu durum denklem 4.63'de ifade edilmektedir.

$$\tilde{y} = r_s + \gamma \max_{a \in A} Q(s', a' : \theta)$$
$$\hat{y} = Q(s, a : \theta) \tag{4.63}$$

Denklem 4.63'den yararlanarak maliyet fonksiyonu denklem 4.64'deki gibi ifade edilmektedir.

$$L_i(\theta_i) = E_{s,a,r}[(\tilde{y} - Q(s, a, \theta_i))^2] \tag{4.64}$$

Denklem 4.64'den yararlanarak gradyan azalan algoritmasıyla iteratif olarak L maliyet fonksiyonunu minimum yapan θ parametreleri bulunabilir. Denklem 4.65'de maliyet fonksiyonunun gradyanı görülmektedir.

$$\nabla_{\theta_i} L(\theta_i) = E_{s,a,r,s'}[(r + \gamma \max_{a \in A} Q(s', a' : \theta_{i-1}) - Q(s, a : \theta_i)) \nabla_{\theta_i} Q(s, a : \theta_i)] \tag{4.65}$$

Denklem 4.64 ve 4.65 kullanılarak derin Q öğrenme algoritması oluşturulabilir. Derin Q öğrenme algoritması modelden bağımsız bir yaklaşım olup, kapalı politika bir algoritmadır. Sabit bir politika olarak $\varepsilon - greedy$ algoritması kullanılmaktadır. Derin Q öğrenme algoritması tablo 4.6'da görülmektedir. (Kavukcuoglu, Koray. ve diğ. 2015)

Tablo 4.6: Derin Pekiştirmeli Öğrenme Algoritması.

ALGORİTMA : Derin Q Öğrenme Algoritması		
1	:	D, C *Hafıza alanı ve C güncelleme adım parametrelerini oluştur.*
2	:	$\tilde{Q}(s, a: \theta)$ *Tahmin değerini rastgele üretilen* θ *parametreleriyle hesapla.*
3	:	$\hat{Q}(s, a: \theta)$ *Hedef değerini rastgele üretilen* θ *parametreleriyle hesapla.*
4	:	$Eps \leftarrow 0$
5	:	s *Ajanın başlangıç durumu.*
6	:	$t \leftarrow 0$
7	:	ε *olasılıkla rastgele* a_t *aksiyon seç.*
8	:	$1-\varepsilon$ *olasılıkla* $a_t = \underset{a \in A}{argmax} Q(s_t, a_t: \theta)$ *aksiyon seç.*
9	:	a_t *aksiyonunu sisteme uygula ve* r_t, s_{t+1} *ödül ve bir sonraki durumu hesapla.*
10	:	$D = (s_t, a_t, r_t, s_{t+1})$ *Belleğe verileri kaydet.*
11	:	$D = (s_j, a_j, r_j, s_{j+1})$ *Bellekten rastgele veriler oluştur.*
12	:	$\hat{y} = \begin{cases} r_j & t = j+1 \\ r_j + \gamma \max_{a \in A} \hat{Q}(s_{j+1}, a, \theta_{t-1}) & dd \end{cases}$
13	:	$(\hat{y} - \tilde{Q}(s_j, a_j: \theta_t))^2$ *Maaliyet fonksiyonunu gradyan azalan ile optimum yap.*
14	:	*Her C adımda bir* $\hat{Q} = \tilde{Q}$

4.9 Derin Deterministik Politika Gradyanı Algoritması

Derin deterministik politika gradyanı algoritması, deterministik politika gradyanı algoritması ve derin Q öğrenme algoritmasının birleştirilmesiyle oluşturulmuştur. Derin Q öğrenme algoritması sürekli durum ve ayrık aksiyon kümesi üzerinde çalışmaktadır. Birçok kontrol problemi sürekli durum ve sürekli aksiyon uzayına sahiptir. Derin Q öğrenme algoritması kapalı politika bir algoritma olup $\varepsilon - greedy$ yaklaşımıyla aksiyonları seçmektedir. Sürekli aksiyon uzayında bu yaklaşım uygulanmamaktadır. Sürekli aksiyon uzayına derin Q öğrenme algoritmasını uygulamak için aksiyon uzayının ayrıklaştırılması gerekmektedir. Fakat aksiyon uzayını ayrıklaştırmak birçok problem oluşturur. Örneğin 7DOF bir robot kol için, her bir eklem uygun aksiyon değerleri $a_i \in \{-k, 0, k\}$ olsun böylece aksiyon uzayımızdaki eleman sayısı $3^7 = 2187$ olmaktadır. Aksiyon uzayındaki üç elemanın çözünürlüğünün yetmediği durumlarda daha fazla aksiyon eklemek gerekmektedir ve bu durumda 7DOF bir sistem için üstel olarak büyümektedir.

Aksiyon uzayının çok büyük olması durumunda ajanın yeterince aksiyon uygulayıp bulunduğu ortamı keşfetmesi imkansız olacaktır ve derin Q öğrenme algoritmasının eğitimi gerçekleşmeyecektir. Ayrıca aksiyon uzayının ayrıklaştırılması bilgi kaybına sebep olmaktadır. Derin deterministik politika gradyanı algoritması sürekli aksiyon uzayının bulunduğu ortamlarda politika değerini fonksiyon yaklaşıklayıcı yapılarla öğrenip, bu problemi çözmektedir. Deterministik politika gradyanı algoritması aktör-kritik algoritması olarak önerilmektedir. Fakat bu algoritmanın kararlılık problemi vardır. Aktör-kritik algoritması ile derin Q öğrenme algoritması birleştirilerek derin determinisitik politika gradyanı algoritması oluşmaktadır. Derin Q öğrenme algoritmasında kullanılan lineer olmayan fonksiyon yaklaşıklayıcının başarımı, algoritmaya yapılan iki yenilikle güncellenmiştir. Bellek alanına kaydedilen değerler arasında en az korelasyonun olduğu ve bu bellek alanından örnekleme yapılarak öğrenme sağlanan bir yapı sayesinde derin Q öğrenme algoritması daha başarılı bir şekilde yakınsama sağlamaktadır. İkinci bir yenilik ise $\hat{Q}(s, a: \theta)$ oluşturulup hedef model yapılmaktadır ve algoritma buna göre güncellenmektedir. Tablo 4.6'da derin Q öğrenme algoritmasında bu durum ifade edilmektedir. Derin öğrenme teknikleri ve pekiştirmeli öğrenme tekniklerinin birleşmesiyle robotlar veya otonom sistemler ham veriler üzerinden öğrenme gerçekleştirmektedir. Örneğin bir kamera görüntüsünü alıp işleyen ve elde edilen bilgiler üzerinden öğrenme gerçekleştiren derin pekiştirmeli öğrenme uygulamaları mevcuttur. Geliştirilen algoritmalar klasik model tabanlı kontrol yaklaşımından daha çok başarı sergilemektedir. Robotik uygulamalarında eklem bilgileri ve kartezyen uzaydaki değişken değerlerini algoritmaya giriş verisi olarak uygulayıp uygun politika değerlerini öğretmemiz mümkündür. Yapay ajanın ayrık zaman adımlarında bulunduğu ortam içerisine bir $a_t \in \Re^N$ eylemi uygulayıp, s_{t+1}, r_t kaydetmektedir. Yapay ajanın amacı $R_t = \sum_{t=1}^{T} \gamma^{t-1} r_t(s_t, a_t)$ gelecekte oluşturacağı tüm ödüllerin toplamını maksimum yapmaktır. π politikamıza bağlı olarak belirli bir durumda hangi aksiyonu uygulayacağımız ve buna bağlı olarak elde edeceğimiz ödül değerleri değişmektedir. Politika ifademiz deterministik veya stokastik olabilmektedir. Denklem 4.66'da belirli bir s_t durumunda bir politika π takip ederek uygulanan a_t eyleminin aksiyon değer fonksiyonu ifade edilmektedir. (Lillicrap, Timothy P. ve diğ. 2016)

$$Q^{\pi}(s_t, a_t) = E[R_t | s_t, a_t] \tag{4.66}$$

Birçok pekiştirmeli öğrenme yaklaşımı denklem 4.66'nın öz yinelemeli yaklaşım olarak Bellman denklemiyle genişletilmiş ifadesini kullanmaktadır. Böylece hesaplamalar öz yinelemeli olarak yapılmaktadır. Bu durum denklem 4.67'de ifade edilmektedir.

$$Q^{\pi}(s_t, a_t) = E[r(s_t, a_t) + \gamma Q^{\pi}(s_{t+1}, a_{t+1})] \tag{4.67}$$

Yapay ajan deterministik bir politika kullanıyorsa denklem 4.67 denklem 4.68'deki gibi ifade edilmektedir.

$$Q^{\pi}(s_t, a_t) = E[r(s_t, a_t) + \gamma Q^{\pi}(s_{t+1}, \pi(s_{t+1}))] \tag{4.68}$$

Q öğrenme algoritmasında $\pi(s_{t+1})$ politikası $\varepsilon - greedy$ yaklaşımıdır. Denklem 4.68 ifadesinin parametrik olarak yazımı denklem 4.69'daki gibidir.

$$\begin{aligned} L(\theta^Q) &= E[(Q(s_t, a_t : \theta^Q) - y_t)^2] \\ y_t &= r(s_t, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1}) : \theta^Q) \end{aligned} \tag{4.69}$$

Denklem 4.69 lineer olmayan cok katmanlı yapay sinir ağlarıyla yaklaşıklamak mümkündür. Q öğrenme algoritmasıyla yüksek boyutlu aksiyonu uzayında denklem 4.69 hesaplanamamaktadır. Bu problemi çözmek için aktör-kritik yaklaşımı kullanılmaktadır. Deterministik politika gradyanı, deterministik parametrik bir politika fonksiyonu $\pi(s|\theta^{\mu})$ kullanarak durumları eylemlerle eşleştirmektedir. Kritik $Q(s, a)$ Bellman denklemini kullanarak Q öğrenmedeki gibi hesaplanmaktadır. Aktör, denklem 4.70'i kullanarak gradyan tabanlı bir güncelleme kullanarak parametrelerini güncellemektedir.

$$\begin{aligned} \nabla_{\theta^{\pi}} J &= E[\nabla_{\theta^{\pi}} Q(s, a|\theta^Q)|s = s_t, a = \pi(s_t|\theta^{\pi})] \\ \nabla_{\theta^{\pi}} J &= E[\nabla_a Q(s, a|\theta^Q)|s = s_t, a = \pi(s_t) \nabla_{\theta_{\pi}} \pi(s_t|\theta^{\pi})|s = s_t] \end{aligned} \tag{4.70}$$

Denklem 4.70 politika gradyanı olarak isimlendirilmektedir. Deterministik politika gradyanı algoritması derin Q öğrenme algoritması yaklaşımıyla $\pi(s_t|\theta^{\pi})$ fonkisyonu oluşturulmaktadır. $\pi(s_t|\theta^{\pi})$ fonksiyonunu yaklaşıklamak için derin yapay sinir ağları kullanılmaktadır. Pekiştirmeli öğrenmede kullanılan derin yapay sinir ağları modellerinin öğrenmesini sağlamak için veriler arasındaki korelasyonu en aza indirmek gereklidir. Böylece algoritmanın performansı artmaktadır. Pekiştirmeli öğrenme içinde ajan çevre ile etkileşime girerek veri topladığı için veriler arasında bir korelasyon bulunmaktadır. Bu durumun ortadan kaldırılması için derin Q öğrenme algoritmasında oluşturulan bellek alanı, derin deterministik politika gradyanı algoritmasında da oluşturulup, bellekten alınan rastgele örnekler üzerinen eğitim gerçekleşmektedir.

Oluşturulan bellekte $D = (s_t, a_t, r_t, s_{t+1})$ değerleri kaydedilmektedir. s_t durumları bir robot için açısal konum, açısal hız vb değerlerdir. Verilerde bu farklı nicel özellikler farklı ölçeklendirmeye sebep olmaktadır. Bellekte oluşan verilerin normalizasyonu sağlanarak eğitilecek olan yapay sinir ağlarının parametrelerini optimize etmesi sağlanır. Sürekli aksiyon uzayındaki en büyük zorluk uygun aksiyonlar uygulayarak ajanın bulunduğu ortamı keşfetmesidir. Derin deterministik politika gradyanı algoritması kapalı politika olduğu için keşif sorununu algoritmadan bağımsız olarak oluşturabilmemizi sağlamaktadır. Böylece aktör'ün üretmiş olduğu politika değerine $N(0,1)$ sıfır ortalamalı birim varyanslı gauss gürültüsü ekleyerek keşif sağlanmaktadır. Bu durum denklem 4.71'de görülmektedir.

$$\pi'(s_t) = \pi(s_t|\theta_t^{\pi}) + N \tag{4.71}$$

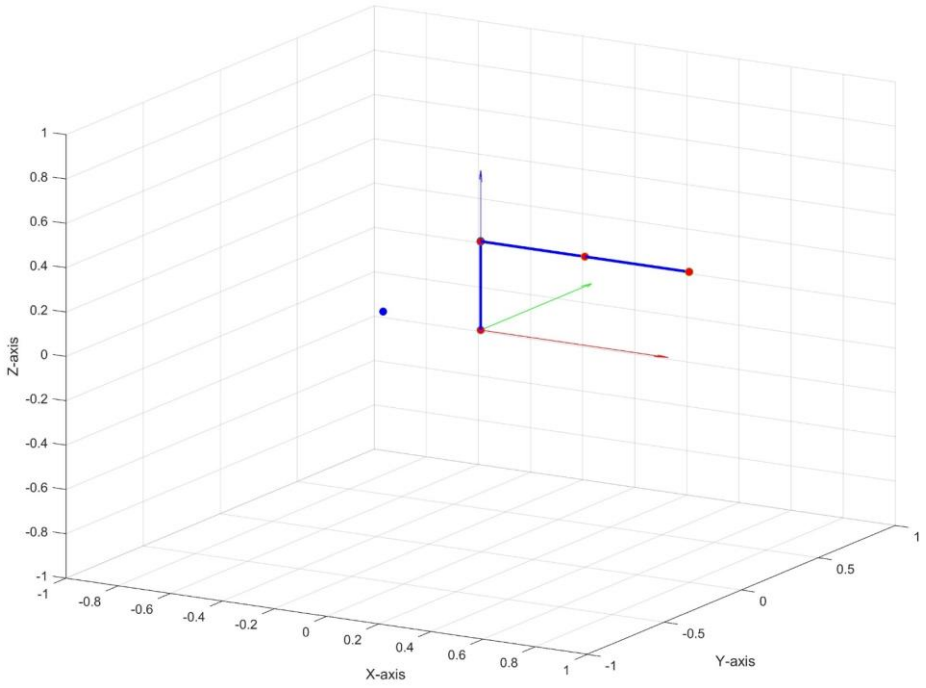
Derin deterministik politika gradyanı algoritması tablo 4.7'de ifade edilmektedir.

Tablo 4.7: Derin Deterministik Politika Gradyanı Algoritması.

ALGORİTMA : Derin Deterministik Politika Gradyanı Algoritması		
1	:	$Q(s,a\|\theta^Q), \pi(s\|\theta^{\pi})$ *Kritik ve aktör yapay sinir ağlarını rastgele parametreler* θ^Q, θ^{π} *ile başlangıcını oluştur.*
2	:	$\hat{Q}, \hat{\pi}$ *Hedef kritik ve aktör yapay sinir ağlarını* $\theta^{\hat{Q}} \leftarrow \theta^Q, \theta^{\hat{\pi}} \leftarrow \theta^{\pi}$ *parametreleriyle başlangıç değerlerini oluştur.*
3	:	D *Bellek alanını oluştur.*
4	:	M *Algoritmanın kaç bölüm çalışacağını oluştur.*
5	:	T *Algoritmanın her bir bölüm içerisinde kaç adım atacağını oluştur.*
6	:	$Bölüm = 1:M$
7	:	*Ajanın kefiş oluşturması için,* T *adet* N *rastgele değer üret.*
8	:	*Başlangıç durumunu* s_1 *oluştur.*
9	:	$t = 1:T$
10	:	*Aksiyon seçimi* $a_t = \pi(s_t\|\theta^{\pi}) + N_t$ *oluştur.*
11	:	*Aksiyonu* a_t *ortama uygula* ve r_t, s_{t+1} *değerlerini gözlemle.*
12	:	$D = (s_i, a_i, r_i, s_{i+1})$ *Değerlerini hafızaya kaydet.*
13	:	*Hafızadan P adet uniform örnek değerler oluştur.*
14	:	$y_i = r_i + \gamma \hat{Q}(s_{t+1}, \hat{\pi}(s_{t+1}\|\theta^{\hat{\pi}})\|\theta^{\hat{Q}})$ *Değerini oluştur*
15	:	$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i\|\theta^Q))^2$ *Kritik* yapay sinir ağını güncelle.
16	:	$\nabla_{\theta^{\pi}} J = \frac{1}{N}\sum_{i=1}^{N} \nabla_a Q(s,a\|\theta^Q)\,\|s = s_i, a = \pi(s_i)\nabla_{\theta^{\pi}}\pi(s\|\theta^{\pi})\|s_i$ *Aktör yapay sinir ağını politika gradyanı ile güncelle.*
17	:	$\theta^{\hat{Q}} \leftarrow \tau\theta^Q + (1-\tau)\theta^{\hat{Q}}$ $\theta^{\hat{\pi}} \leftarrow \tau\theta^{\pi} + (1-\tau)\theta^{\hat{\pi}}$ *Hedef yapay sinir ağlarının parametresini güncelle.*

4.10 DDPG İle 3DOF Robot Kol Tork Kontrolü

Bu tez çalışmasında derin deterministik politika gradyanı algoritmasıyla 3DOF robot kolun tork kontrolü yapılmıştır. Robot kolun tork kontrol problemi bölüm ikide detaylı olarak incelenmiştir. Bölüm ikide yapılan tork kontrol yöntemi model tabanlı olup robotun denklemlerinin ve kontrol algoritmasının oluşturulması gerekmektedir. 3DOF robot kolun DDPG algoritmasıyla çözülmesi yüksek serbestlik dereceli fiziksel sistemler için temel oluşturmaktadır. DDPG algoritması için geliştirilen Python ve Matlab yazılımı sayesinde, robot kol hedeflenen tork kontrolünü sağlayarak noktadan noktaya yönelim işlemini gerçekleştirmektedir. Şekil 4.10'da robotun içinde bulunduğu ortam ve başlangıç durumları görülmektedir.



Şekil 4.10: 3DOF Robot Kol Başlangıç Pozisyonu.

Robot kolun derin deterministik politika gradyanı algoritmasıyla öğrenme gerçekleştirebilmesi için simülasyonda iteratif olarak hesaplamalar gerçekleştirmesi gerekmektedir. Fiziksel bir robot üzerinden öğrenme hem çok maliyetli hemde çok zaman alacağı için robotun simülasyon ortamında fiziksel bir kopyasını oluşturmak gerekmektedir. Bu tez çalışmasında robotun kinematik ve dinamik denklemleri simülasyon ortamında oluşturulmuştur. Robotun dinamik denklemleri bölüm ikide ifade edilmektedir. Simulasyon içerisinde Runge Kutta metodu ile dinamik denklemlerin çözümü sağlanmıştır.

Fiziksel sistemimiz yüksek serbestlik dereceli ve modellenemeyecek kadar karmaşık bir yapıda ise, fiziksel sistemimizin tasarlanan 3D modelinin fizik motoru tabanlı simülasyon yazılımları ile sonlu elemanlar vb metodlarla sayısal simülasyon sonuçları yardımıyla dinamik analizi yapılmaktadır. Bu analiz sonucunda fiziksel sistemin uç ve iç değişkenlerinin değerleri sayısal olarak hesaplanmaktadır. Bu tez çalışmasında 3DOF robotun fiziksel sistemini ifade eden dinamik denklemleri çözümlenmektedir. Robot kontrol probleminde robotun ters kinematik denklemleri ve ters dinamik denklemleri sayısal olarak çözülmektedir. Derin deterministik politika gradyanı yaklaşımıyla robotun ters kinematik ve ters dinamik denklemleri çözülmemektedir. Robotun iç değişkenleri olan $\tau_i i = 1,2,3$ tork değerleriyle sistem simülasyonu sağlanmaktadır. Tork ifadelerimiz kontrol değişkenlerini oluşturmaktadır. Tork değerlerine rastgele gürültü ekleyerek robotun ileri dinamik modeli Runge Kutta algoritmasıyla iteratif olarak çözülmektedir. Bu sayede robotun eklem açıları bulunmaktadır. Bulunan eklem açılarıyla robotun ileri kinematik modelinden robotun uç işlevcisinin çalışma uzayı içerisinde konumu belirlenmektedir. Uç işlevcinin konumu ile robotun ulaşması gereken hedef konum arasında bir ödül fonksiyonu yazılmaktadır. Robot ödülü maksimum yapmak için hedef konum ile robotun uç işlevcisi konumu arasındaki uzaklığı azaltması gerekmektedir. Robotun bu durumu gerçekleştirmesi için uygun bir politika takip ederek tork değerlerini oluşturmalıdır. Oluşturulan tork değerleriyle robotun kontolü sağlanmaktadır. Derin deterministik politika gradyanı ile 3DOF robotun tork kontrol algoritması tablo 4.8'de görülmektedir.

Tablo 4.8: DDPG Algoritmasıyla 3DOF Robot Kol Tork Kontrolü.

ALGORİTMA : DDPG Algoritmasıyla 3DOF Robot Kol Tork Kontrolü

1 : *AktörYsa , HedefAktörYsa, KritikYsa, HedefKritikYsa Başlangıç parametrelerini rastgele oluştur.*

2 : *D Bellek alanını oluştur.*

3 : $\tau \leftarrow 0.01, \gamma = 0.99$ *HedefYsa parametre güncelleme çarpanı ve gamma parametresi*

4 : *Bölüm* = 20000 *Algoritmanın bölüm sayısı.*

5 : T = 1000 *Algoritmanın her bir bölüm için adım sayısı.*

6 : Ts = 0.01 *Runge Kutta ingtegrasyon adımı.*

7 : B=1:Bölüm

8 : *Robot için kartezyen uzayda uniform rastgele hedef noktaları belirle.*

9 : *Robotun başlangıç değeri.* $q_i, \dot{q}_i, \tau_i = 0 i = 1,2,3$

10 : *Robotun ileri kinematik çözümü.*

11 : $[N(0,5)]_{3xT}$ *Rastgele gürültü değeri üret.*

12 : t=1:T

13 : *Robotun durumlarını oluştur.*
$s(q_i, \dot{q}_i, \ddot{q}_i, Hedef(x,y,z), Robot(x,y,z))$

14 : $u \leftarrow s$ *AktörYsa'ya durumları gönder ve tork değerlerini üret.*

15 : $u \leftarrow u + [N(0,5)]_{3xt}$ *Tork değerlerine rastgele gürültü ekle ve normalize et.*

16 : $s' \leftarrow RungeKutta(u)$ *Sistem integrasyonu yaparak yeni durum değerlerini hesapla.*

17 : s' *yeni durum değerleri olan* $q_i i = 1,2,3$ *eklem açılarıyla robotun uç işlevci konumunu kullanarak* r_t *ödül değerini hesapla.*

18 : $D(s_t, a_t, r_t, s_{t+1})$ *değerlerini hafızaya kaydet.*

19 : *D hafıza boyutu* > 5000 *ise algoritmanın eğitimini başlat.*

20 : $(s_t, a_t, r_t, s_{t+1}) \leftarrow$ *D Hafızadan uniform örneklem oluştur.*

21 : $y = r + \gamma HedefKritikYsa(s_{t+1}, HedefAktörYsa(s_{t+1}))$

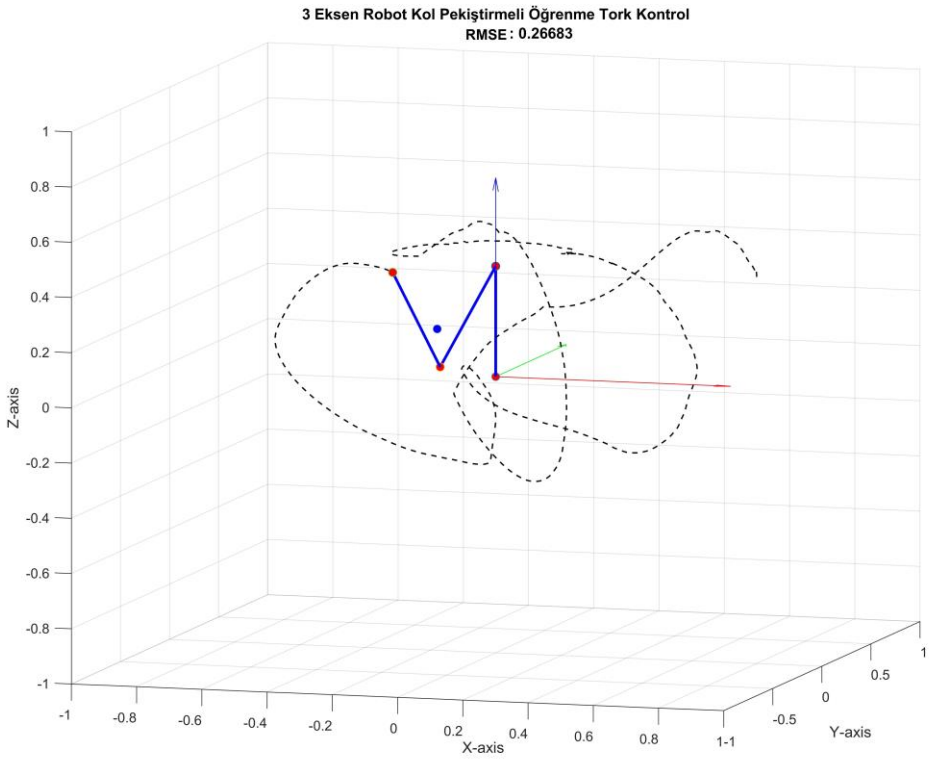
22 : $L = \frac{1}{N}\sum_{i=1}^{N}(y - KritikYsa(s,a))^2$ *KritikYsa güncelle.*

23 : $\nabla_{\theta^\pi} J = \frac{1}{N}\sum_{i=1}^{N} \nabla_a Q(s,a|\theta^Q)\,|s = s_i, a = \pi(s_i) \nabla_{\theta^\pi} \pi(s|\theta^\pi)|s_i$ *AktörYsa politika gradyanı algoritmasıyla güncelle.*

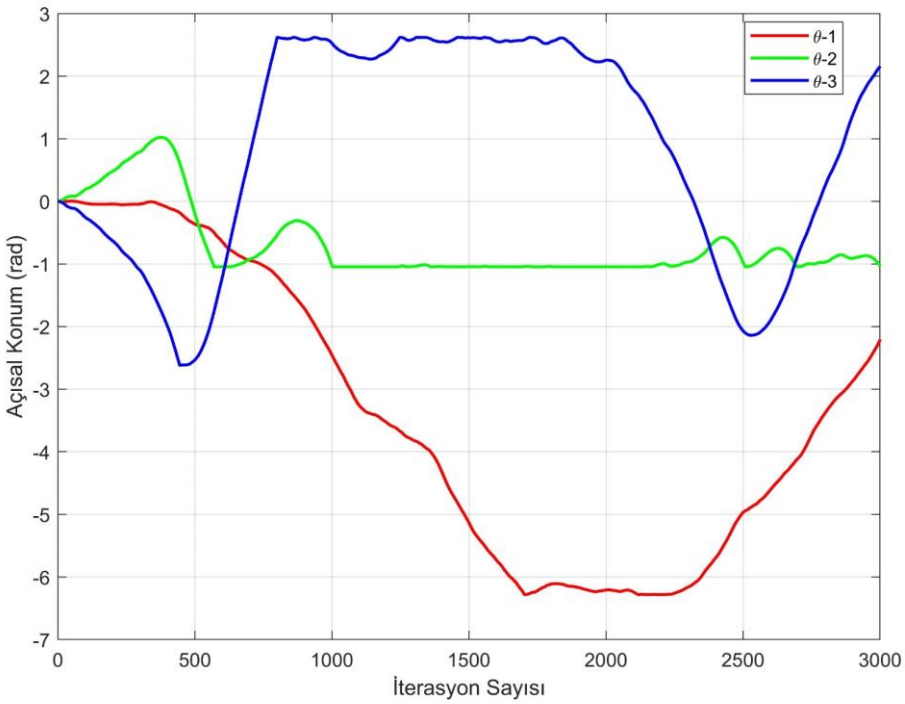
24 : $\theta^{\hat{Q}} \leftarrow \tau\theta^Q + (1-\tau)\theta^{\hat{Q}}$
$\theta^{\hat{\pi}} \leftarrow \tau\theta^\pi + (1-\tau)\theta^{\hat{\pi}}$ *Hedef Yapay Sinir Ağlarının Parametrelerini güncelle.*

Tablo 4.8'de önerilen algoritmada kullanılan yapay sinir ağları iki gizli katmanlı olup her bir katmanda 400 nöron bulunmaktadır ve aksivasyon fonksiyonu olarak tanjant hiperbolik fonksiyonu kullanılmaktadır.

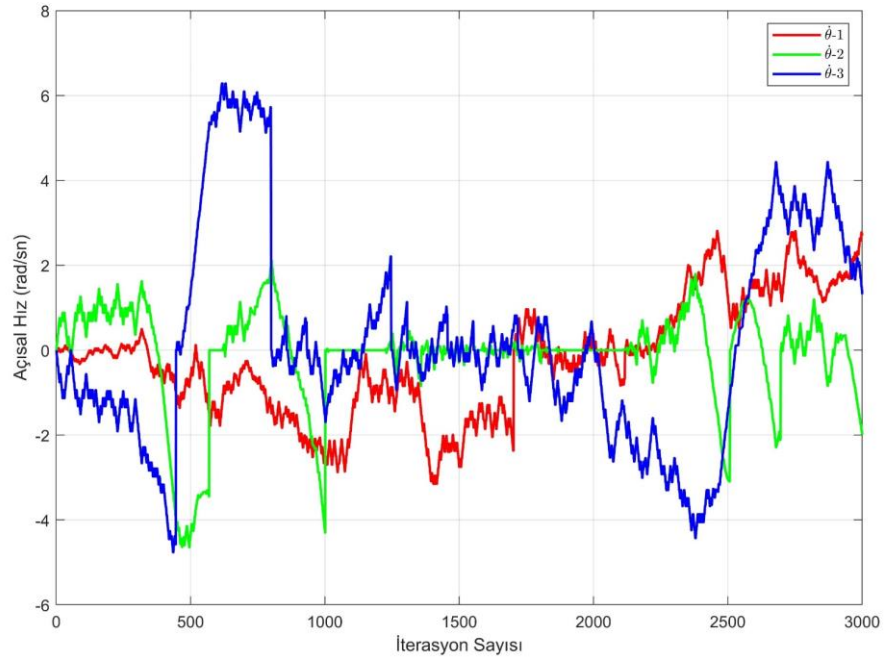
Algoritmada kullanılan yapay sinir ağları ve diğer değişkenler Nvdia CUDA alt yapısı kullanılarak oluşturulmuştur. CUDA kütüphanesi sayesinde bilgisayarın ekran kartı üzerinden hesaplamalar daha hızlı yapılmaktadır. Algoritmanın eğitimi için kullanılan bilgisayarın teknik özellikleri intel i7 9750H 4.5 Ghz işlemci, 32 GB ddr4 ram, Nvdia 1660 Ti ekran kartına sahiptir. Algoritmanın başarılı bir eğitim gerçekleştirmesi için en az 2000 bölüm çalışması gerekmektedir. Böylece hesaplama adımı olarak 2.000.000 adım sonunda robot kol çalışma uzayında hedeflenen bölge üzerinde iyi bir öğrenme gerçekleştirmektedir. Robot kol için oluşturulan ödül fonksiyonu $r = \sqrt{(\hat{x} - \tilde{x})^2 + (\hat{y} - \tilde{y})^2 + (\hat{z} - \tilde{z})^2}$'dir. Önerilen algoritmada robotun hedef noktası her bir bölümde rastgele üretildiği için robotun çalışma uzayını iyice genellemesi gerekmektedir. Böylece eğitilen model daha iyi genelleme yapabilmektedir. Algoritmada oluşturulan rastgele tork değerlerine karşılık robot kol çalışma uzayında rastgele değerler almaktadır. Bu durum şekil 4.11-22'de görülmektedir.



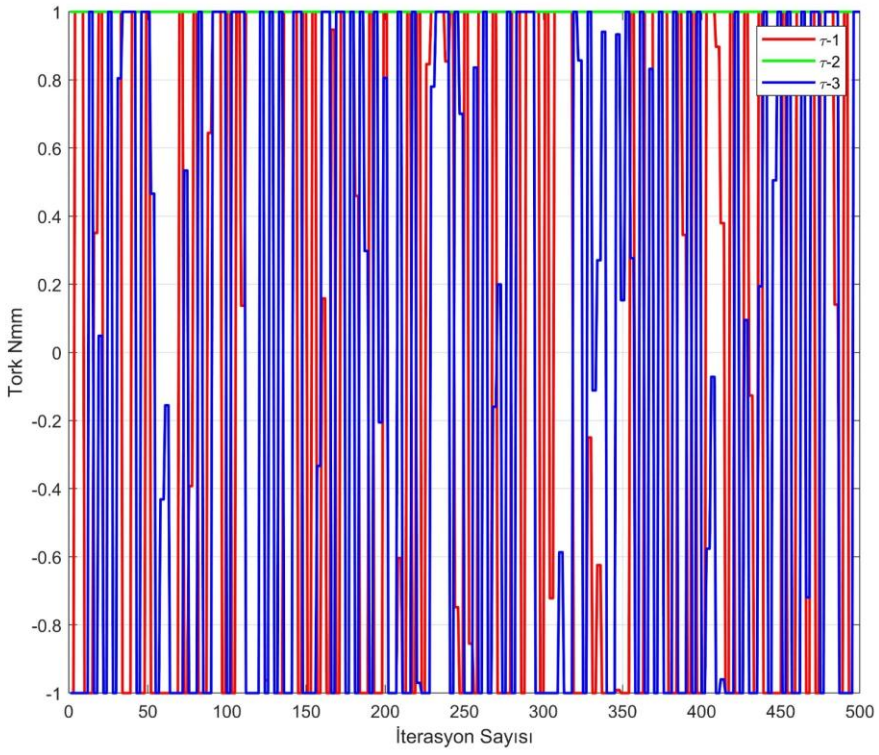
Şekil 4.11: Robotun Eğitim Aşaması – 1 Uç İşlevci.



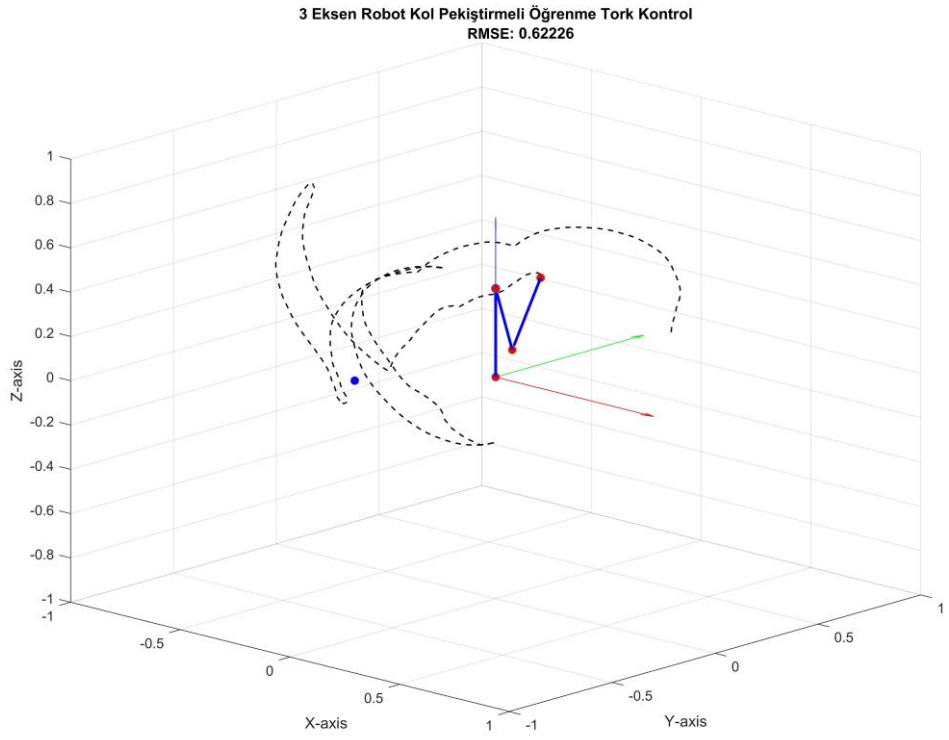
Şekil 4.12: Robotun Eğitim Aşaması – 1 Eklem Açı Değerleri.



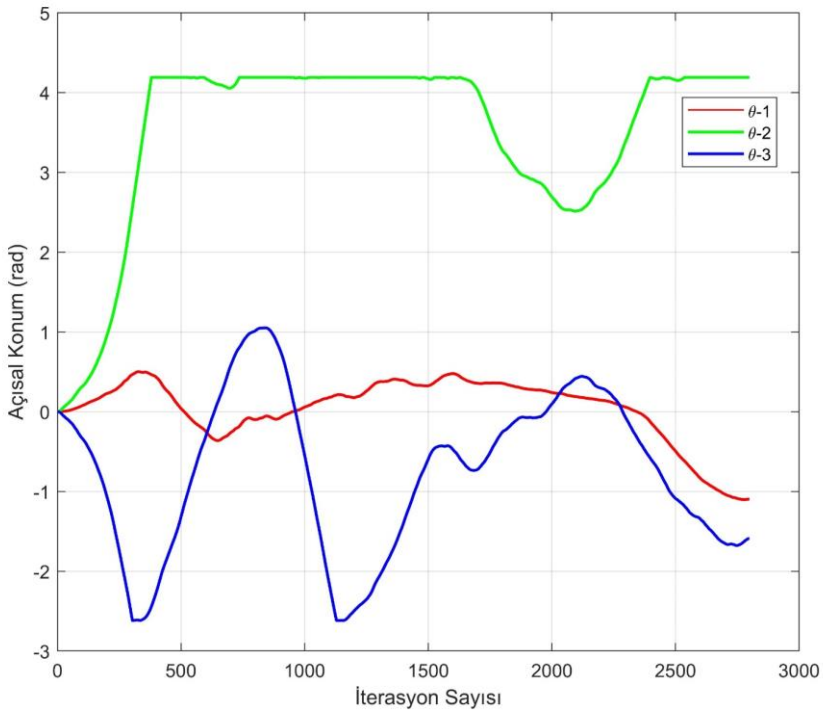
Şekil 4.13: Robotun Eğitim Aşaması – 1 Eklem Açısal Hız Değerleri.



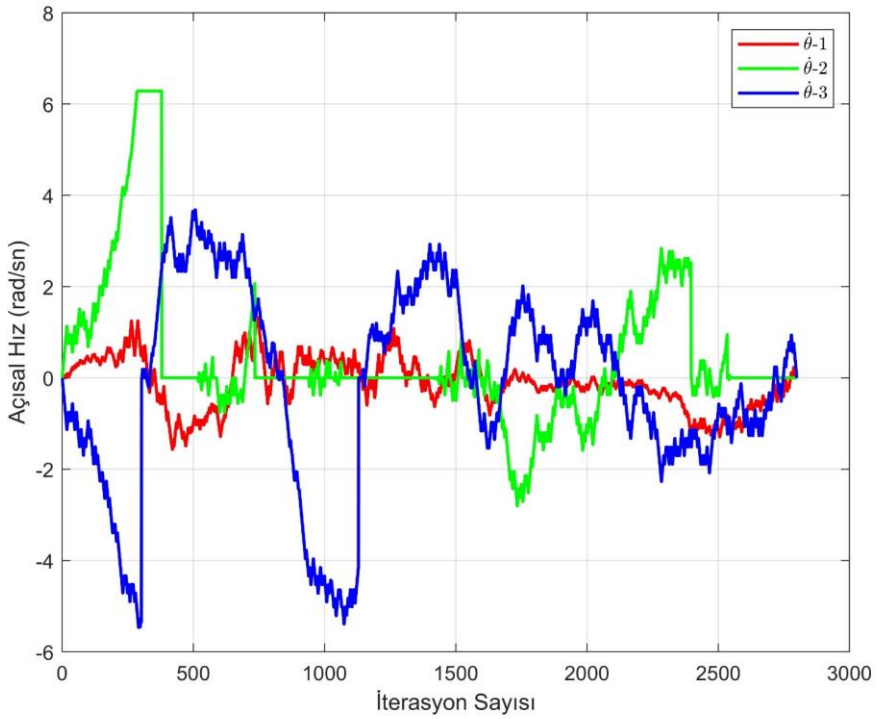
Şekil 4.14: Robotun Eğitim Aşaması – 1 Eklem Tork Değerleri.



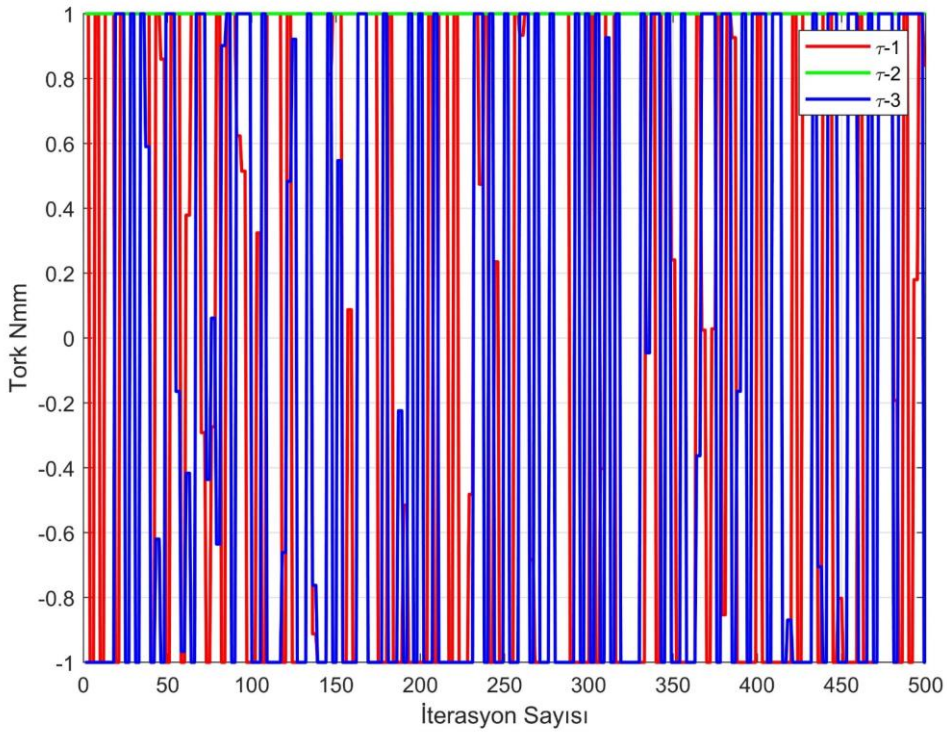
Şekil 4.15: Robotun Eğitim Aşaması – 2 Uç İşlevci.



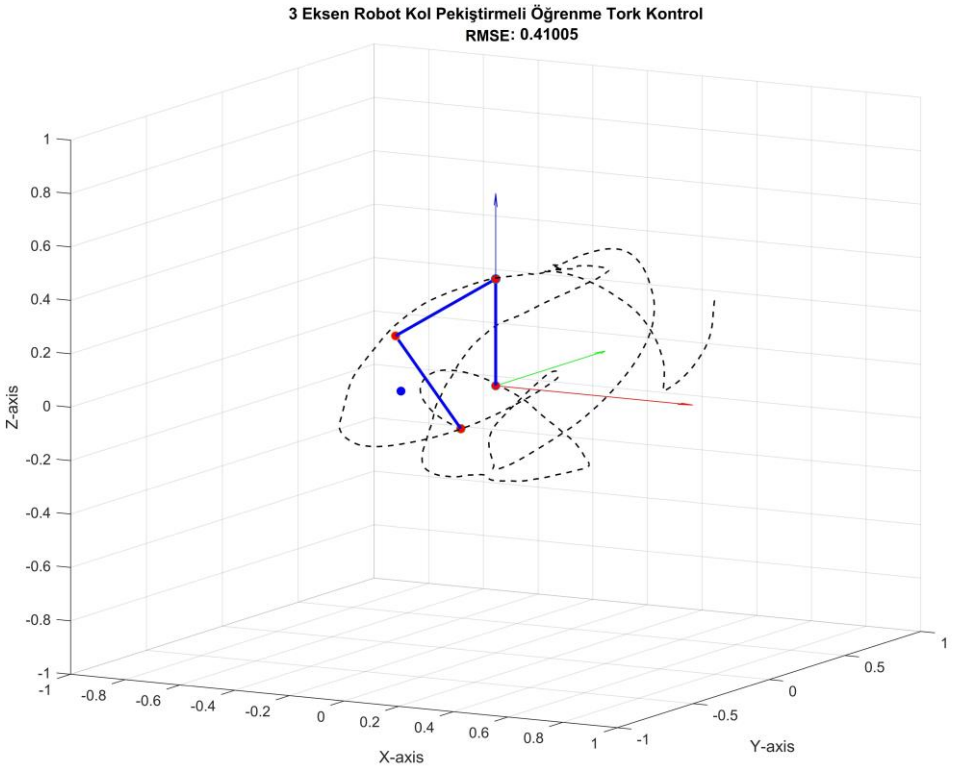
Şekil 4.16: Robotun Eğitim Aşaması – 2 Eklem Açı Değerleri.



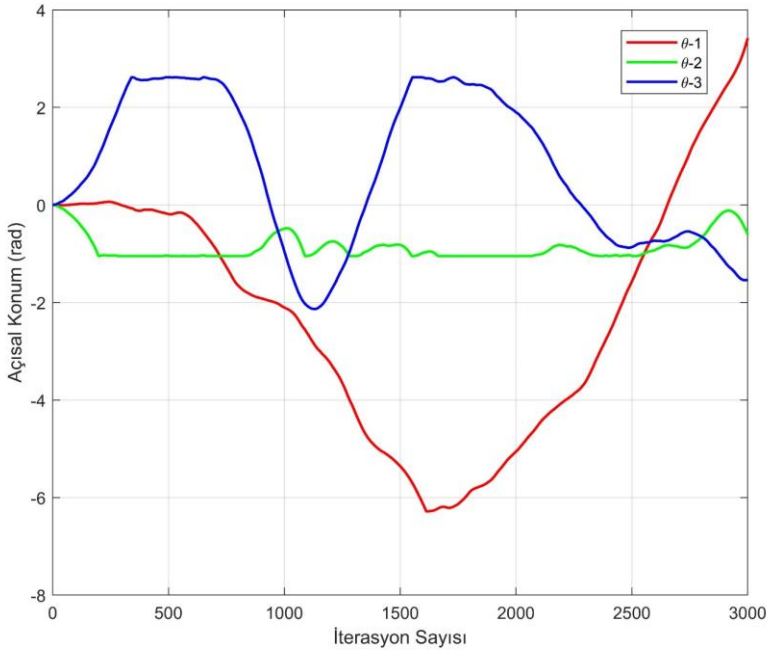
Şekil 4.17: Robotun Eğitim Aşaması – 2 Eklem Açısal Hız Değerleri.



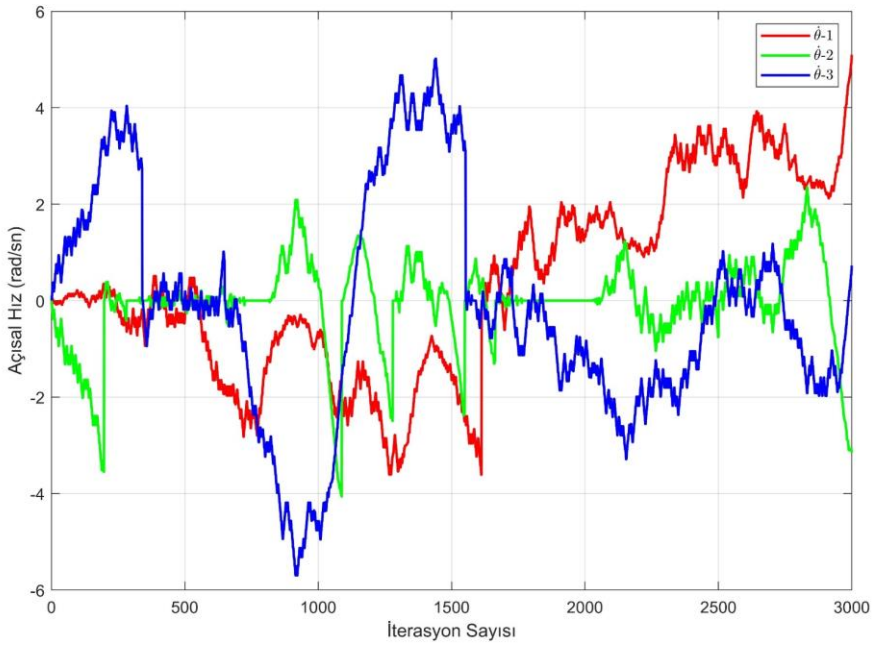
Şekil 4.18: Robotun Eğitim Aşaması – 2 Eklem Tork Değerleri.



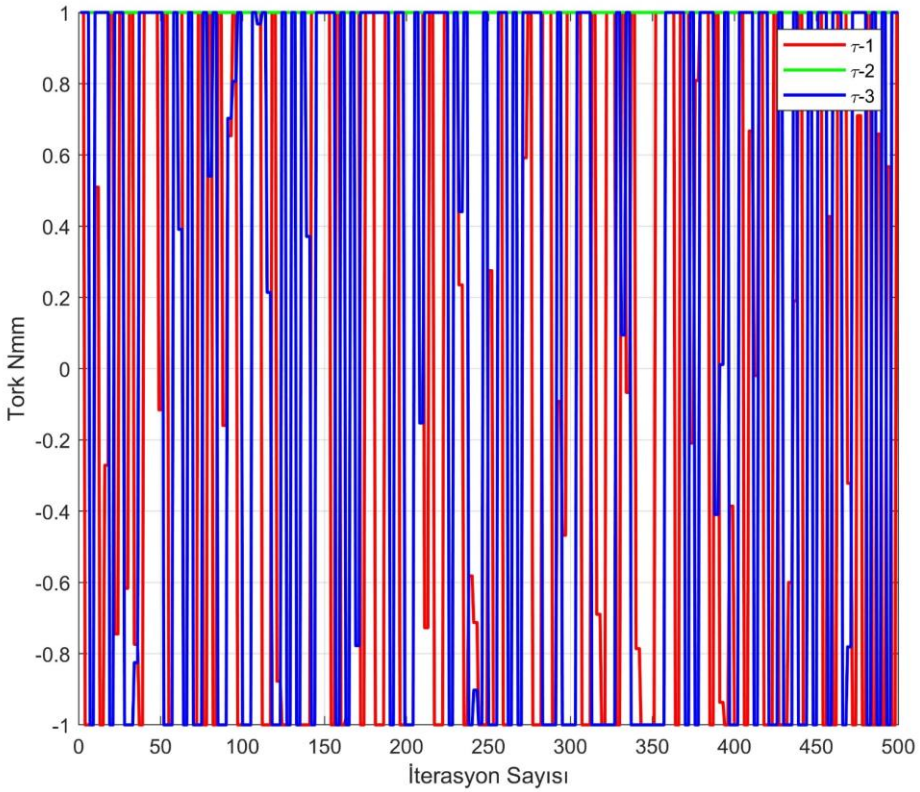
Şekil 4.19: Robotun Eğitim Aşaması – 3 Uç İşlevci.



Şekil 4.20: Robotun Eğitim Aşaması – 3 Eklem Açı Değerleri.

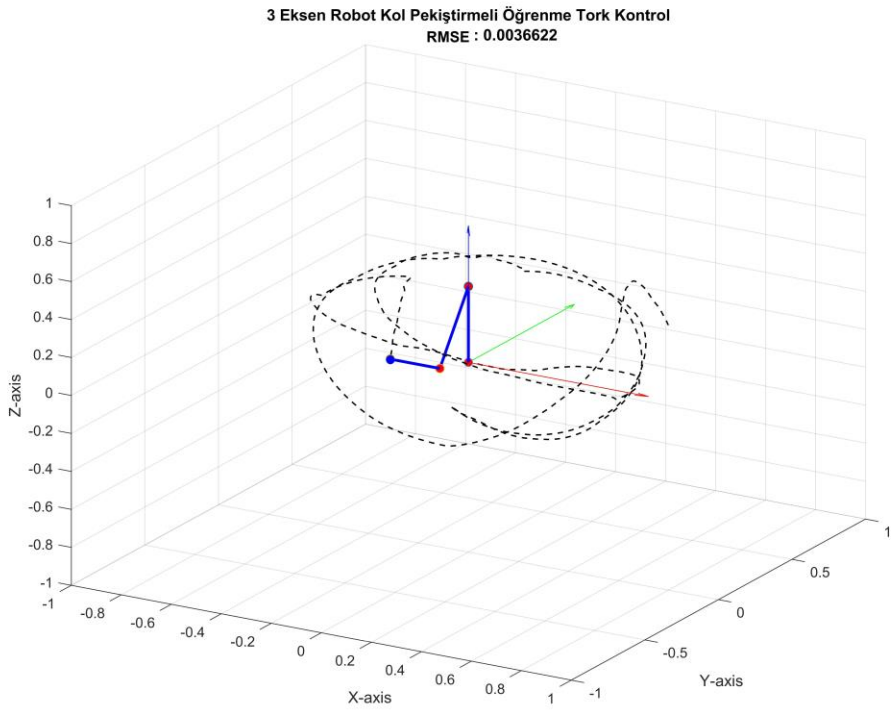


Şekil 4.21: Robotun Eğitim Aşaması – 3 Eklem Açısal Hız Değerleri.

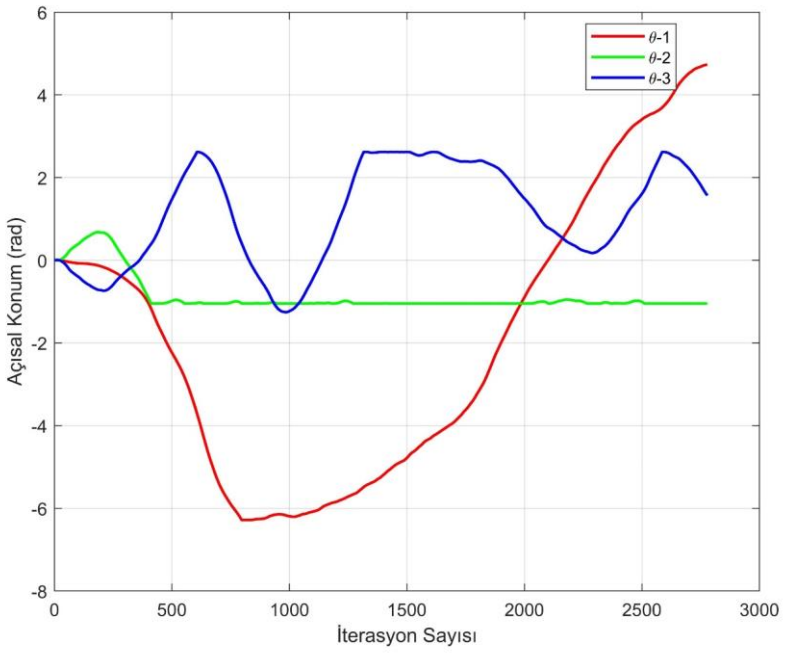


Şekil 4.22: Robotun Eğitim Aşaması – 3 Eklem Tork Değerleri.

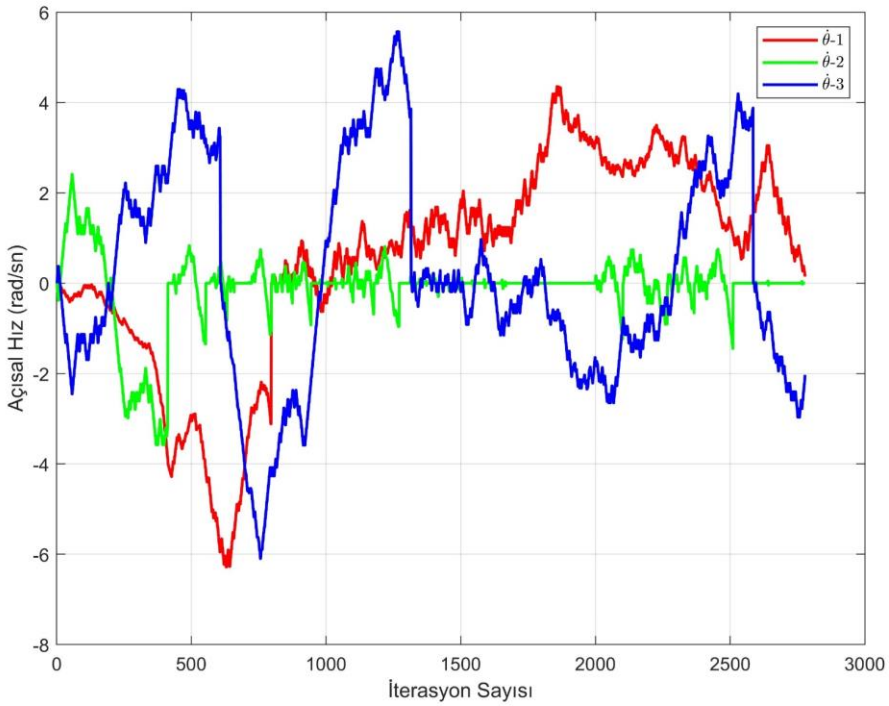
Robot herhangi bir adımda öğrenme gerçekleştirdikçe, tork değerlerine uygulanan gürültü işareti üstel olarak azalmakta böylece robot öğrendiği tecrübeler ile yeni hareketler uygulamaktadır. Bu durum robotun çalışma uzayı içerisinde takip ettiği yörüngeyi etkilemektedir. Öğrenme süreci adımları şekil 4.23-38 görülmektedir.



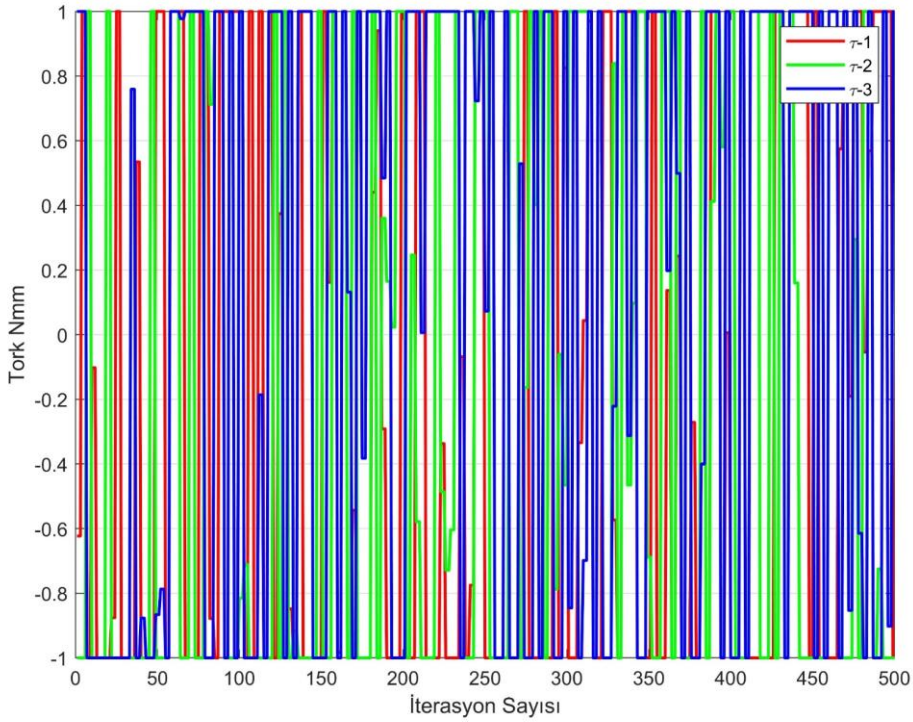
Şekil 4.23: 3DOF Robot Kontrol Çözüm - 1 Uç İşlevci.



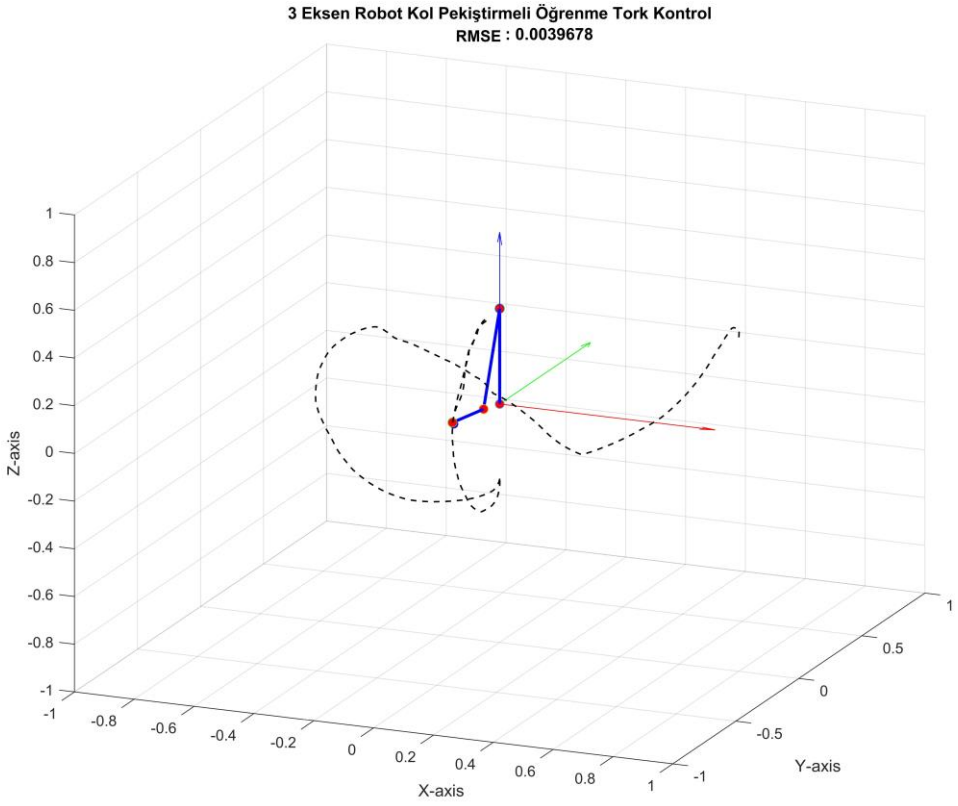
Şekil 4.24: 3DOF Robot Kontrol Çözüm - 1 Eklem Açı Değerleri.



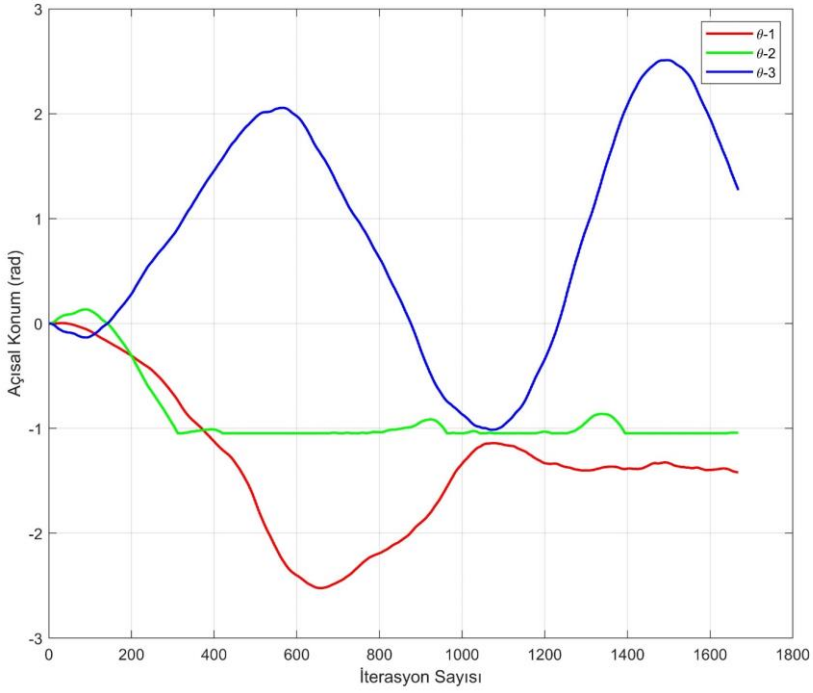
Şekil 4.25: 3DOF Robot Kontrol Çözüm - 1 Eklem Açısal Hız Değerleri.



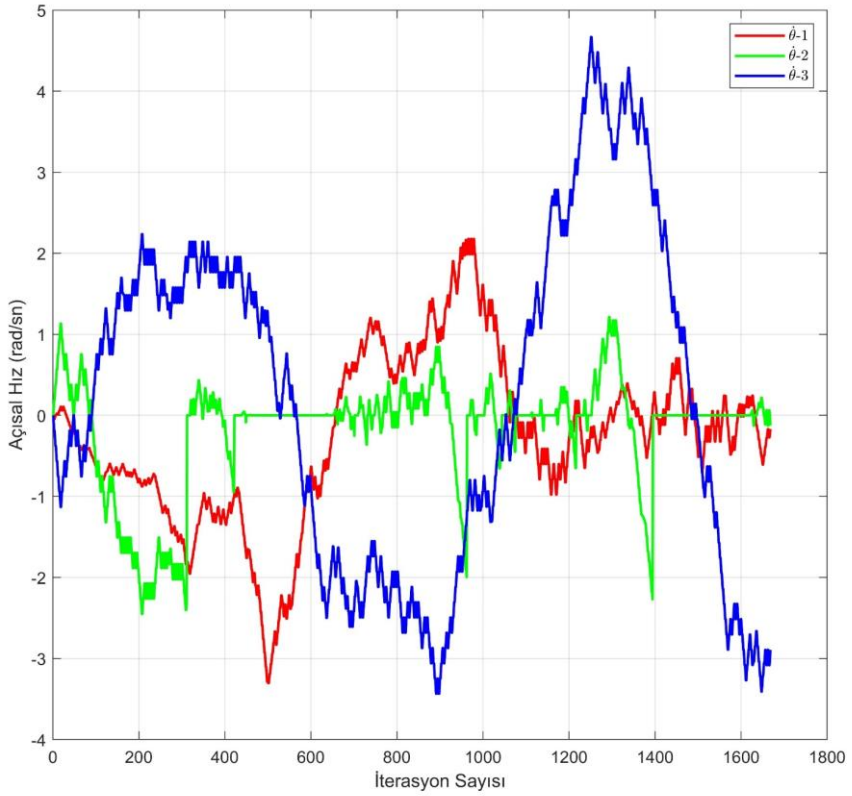
Şekil 4.26: 3DOF Robot Kontrol Çözüm - 1 Eklem Tork Değerleri.



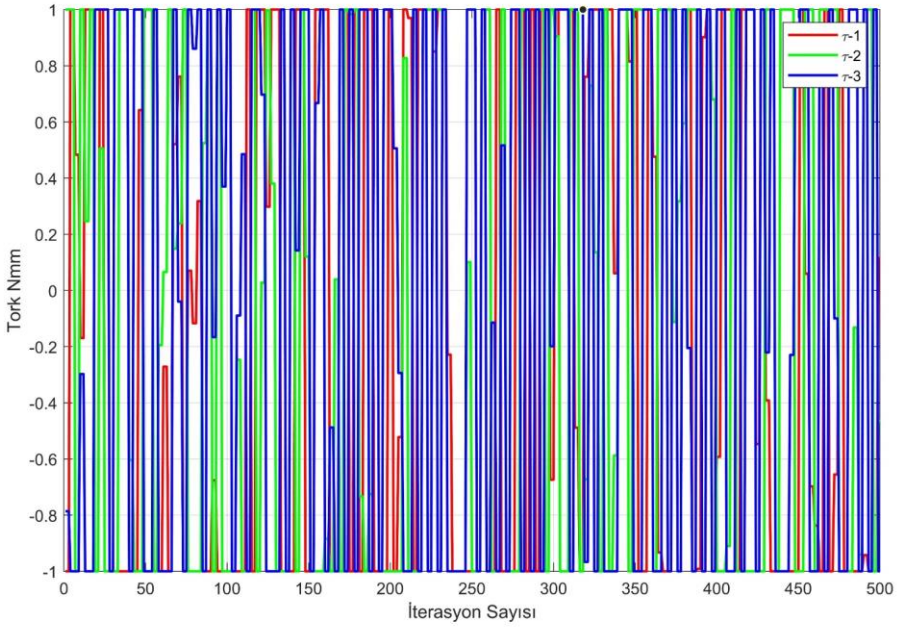
Şekil 4.27: 3DOF Robot Kontrol Çözüm - 2 Uç İşlevci.



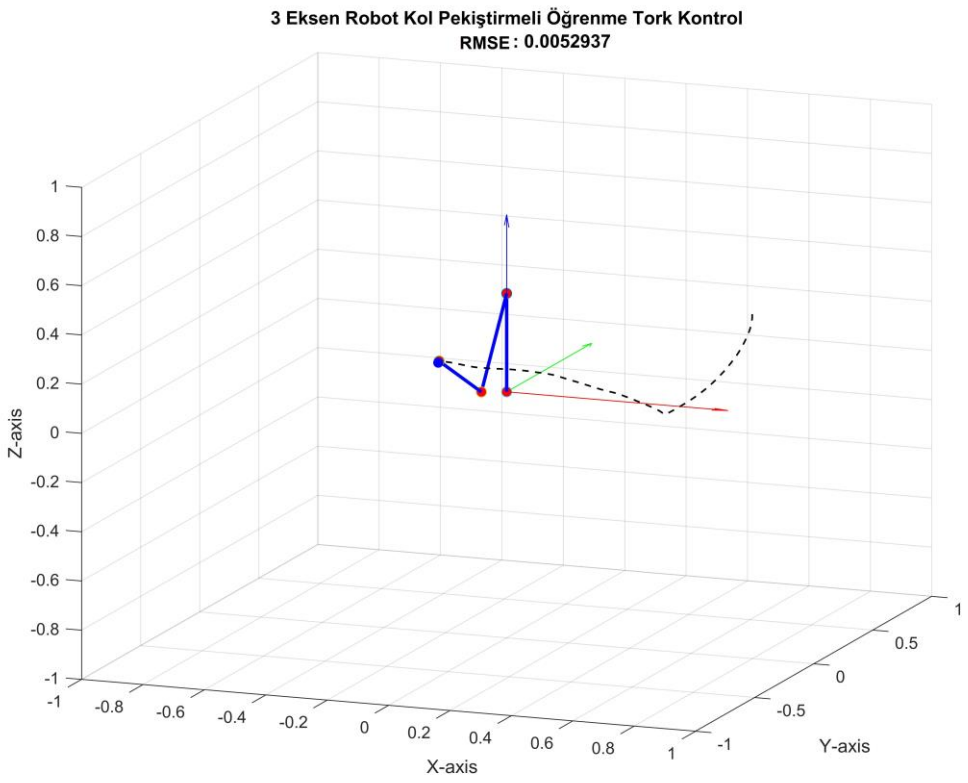
Şekil 4.28: 3DOF Robot Kontrol Çözüm - 2 Eklem Açı Değerleri.



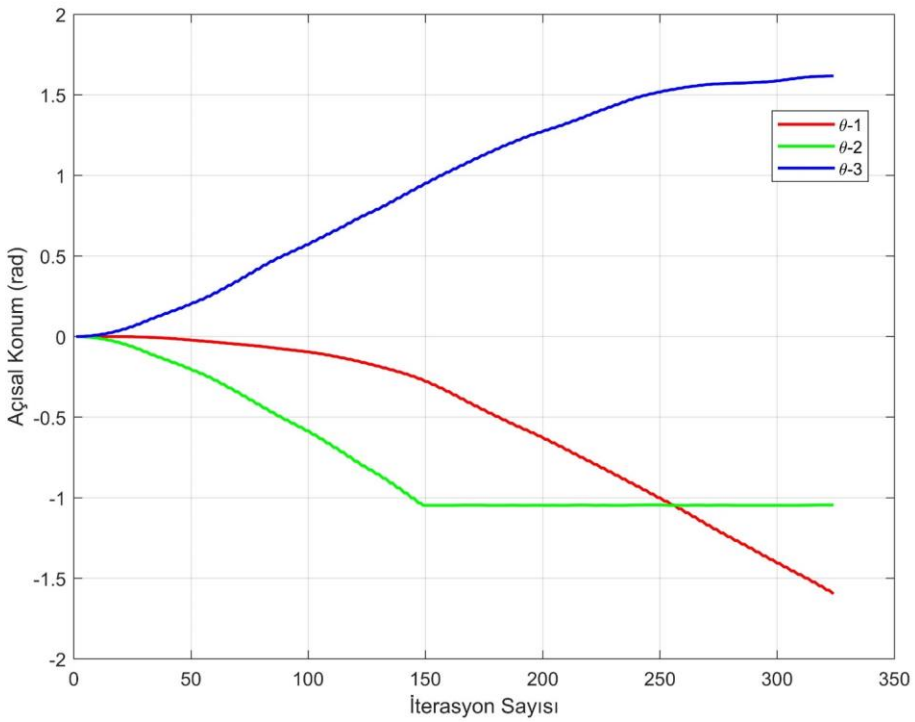
Şekil 4.29: 3DOF Robot Kontrol Çözüm - 2 Eklem Açısal Hız Değerleri.



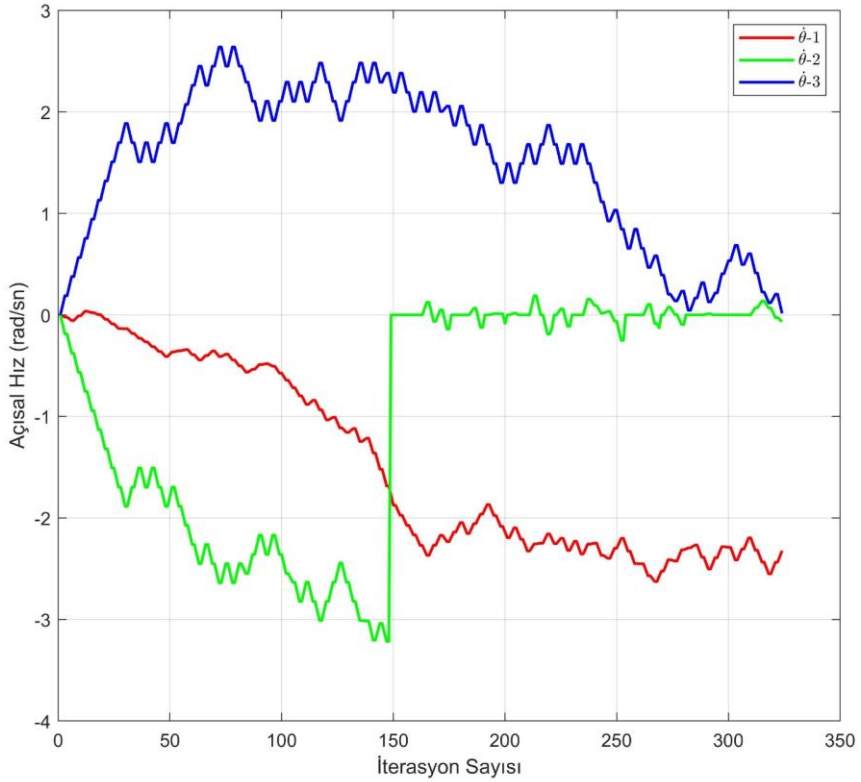
Şekil 4.30: 3DOF Robot Kontrol Çözüm - 2 Eklem Tork Değerleri.



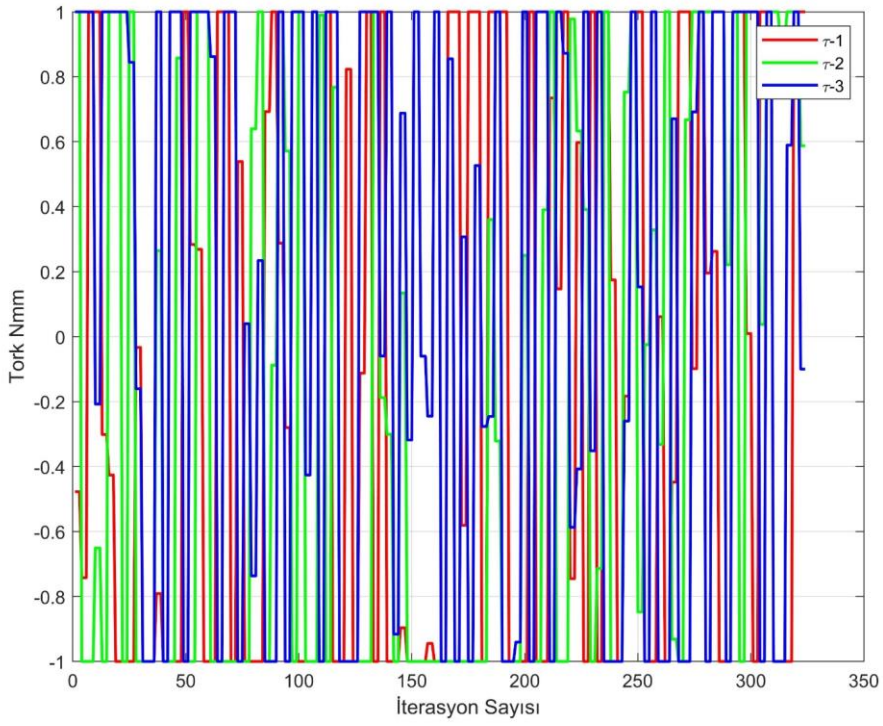
Şekil 4.31: 3DOF Robot Kontrol Çözüm - 3 Uç İşlevci.



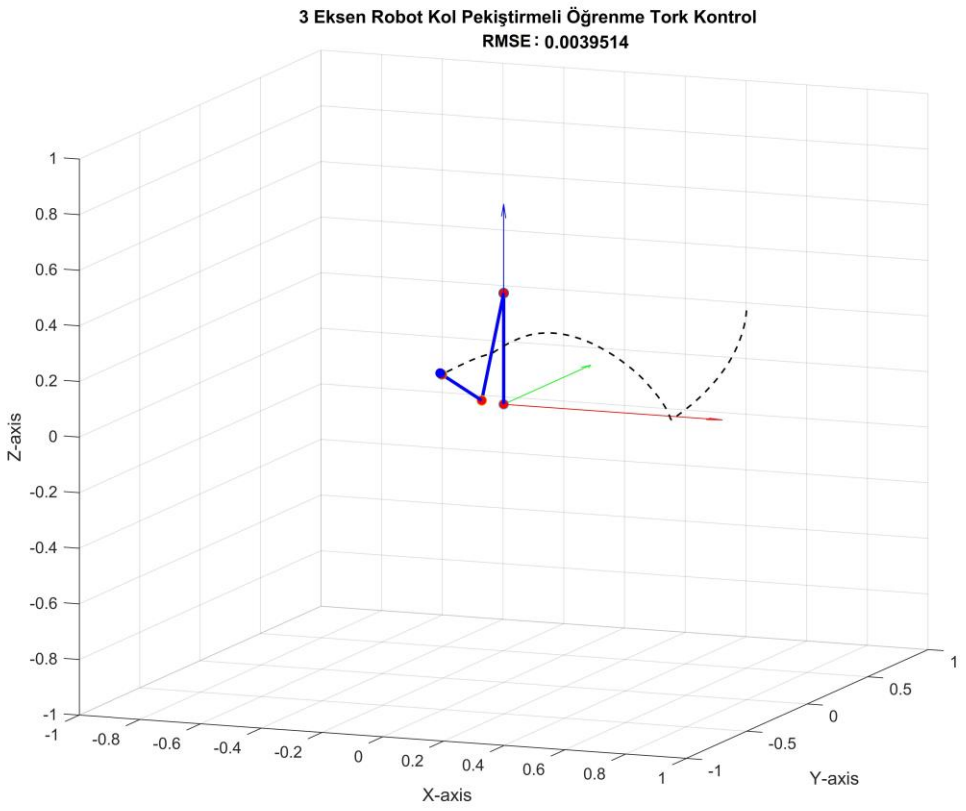
Şekil 4.32: 3DOF Robot Kontrol Çözüm - 3 Eklem Açı Değerleri.



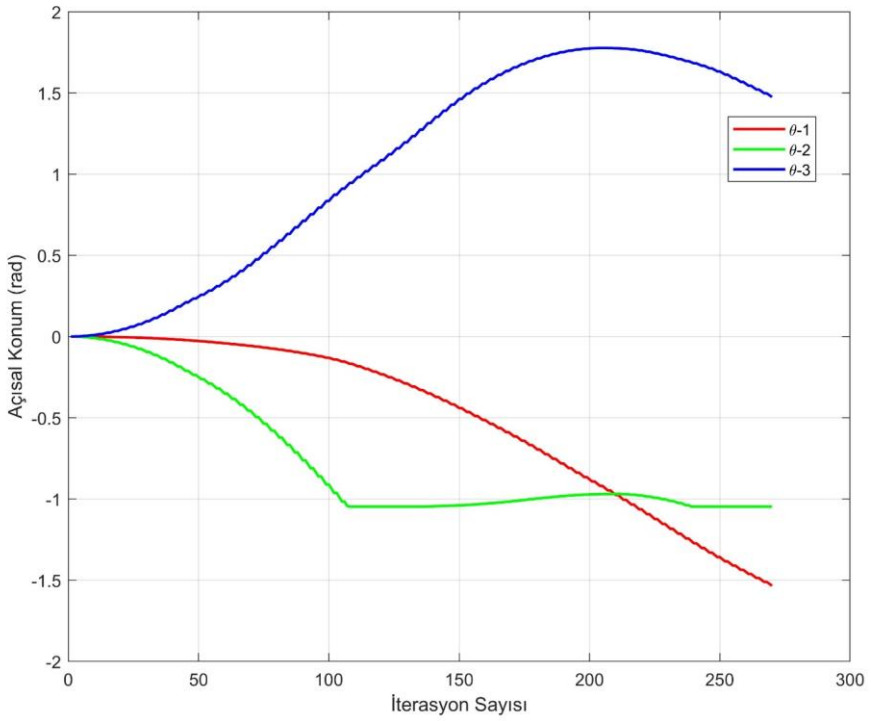
Şekil 4.33: 3DOF Robot Kontrol Çözüm - 3 Eklem Açısal Hız Değerleri.



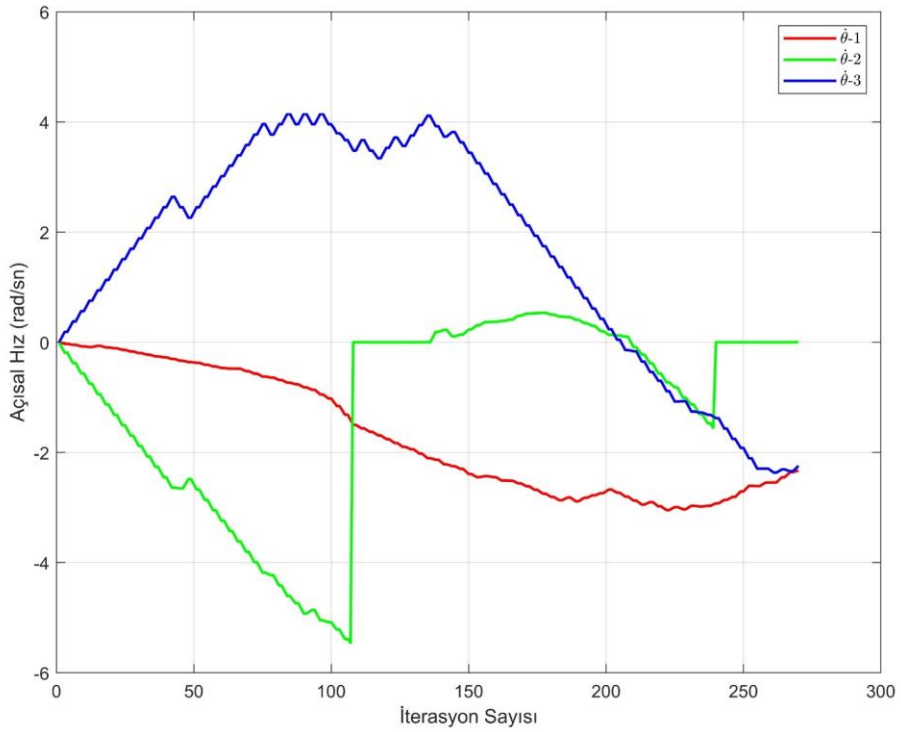
Şekil 4.34: 3DOF Robot Kontrol Çözüm - 3 Eklem Tork Değerleri.



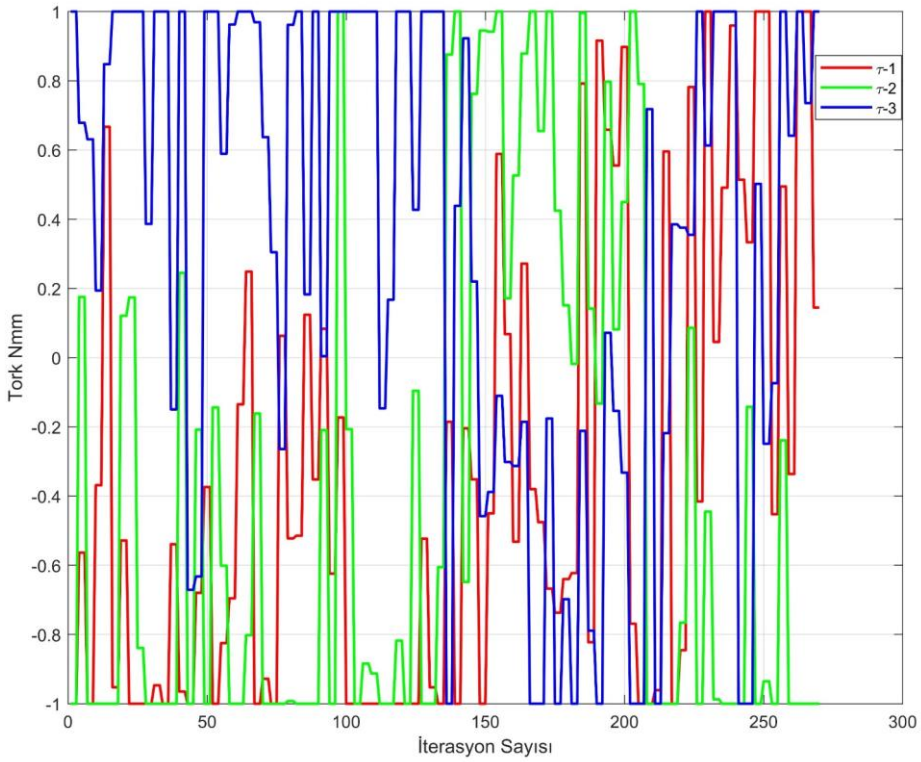
Şekil 4.35: 3DOF Robot Kontrol Çözüm - 4 Uç İşlevci.



Şekil 4.36: 3DOF Robot Kontrol Çözüm - 4 Eklem Açı Değerleri.



Şekil 4.37: 3DOF Robot Kontrol Çözüm - 4 Eklem Açısal Hız Değerleri.



Şekil 4.38: 3DOF Robot Kontrol Çözüm - 4 Eklem Tork Değerleri.

Şekil 4.23-38 ile DDPG algoritmasının robot kolun tork kontrolünde eğitim ilerledikçe daha başarılı sonuçlar ortaya koymaktadır. Robot kolun DDPG ile kontrolü için robota uygun ödül fonksiyonları yazılarak başarımın artması sağlanabilmektedir. Robot için yazılan ödül fonksiyonu robotun gerçekleştirmesi gereken eylemlere göre oluşturulmalıdır. Bu tez çalışmasında noktadan noktaya erişim sağlamak için uygun ödül fonksiyonu yazılmıştır.

5. SONUÇ VE ÖNERİLER

Bu tez çalışmasında, makine öğrenmesinin bir alt kolu olan pekiştirmeli öğrenme yaklaşımı kullanılarak 3DOF robot kolun tork kontrol problemi çözülmüştür. Dinamik sistemlerin modellenmesi ve kontrolü bölüm ikide model tabanlı olarak oluşturulmuştur. Robotik sistemlerde yüksek serbestlik derecesi veya modellenemeyen dinamiklerin olması, fiziksel sistemin kontrolünü zorlaştırmaktadır. Pekiştirmeli öğrenme yaklaşımı kullanılarak robotun matematiksel modelinden bağımsız olarak kontrol problemi çözülmektedir. Yüksek serbestlik dereceli veya paralel robotların modellenmesi aşamasında derin pekiştirmeli öğrenme algoritmaları çözüm oluşturmaktadır. Pekiştirmeli öğrenme yaklaşımının klasik model tabanlı yaklaşım ve optimal kontrol'e göre üstünlüğü, ödül fonksiyonunun probleme özel oluşturulmasıdır. Pekiştirmeli öğrenmede oluşturulan ödül ifadesi optimal kontrol teorisindeki amaç fonksiyonudur. Böylece pekiştirmeli öğrenmede durumlar ve durumlar üzerinden oluşturulan ödüller ile öğrenme süreci gerçekleşmektedir. Pekiştirmeli öğrenme yaklaşımıyla esnek ödül fonksiyonları yazarak robotun gerçekleştirmesi gereken görevi öğrenmesi sağlanmaktadır. Fiziksel birçok sistemin matematiksel modelini oluşturmak, model parametrelerini belirlemek ve kontrolcü tasarlamak, birçok lineer olmayan sistem için mümkün olmamaktadır. Bu duruma alternatif olarak tasarımı gerçekleştirilen fiziksel sistem, simulasyonda sonlu elemanlar, sonlu farklar vb metodlar ile sayısal çözümü gerçekleştirilip sistemin durumları gözlemlenebilmektedir. Fiziksel sistem üzerinden gözlemlenen durumlar sayesinde pekiştirmeli öğrenme algoritmaları öğrenme süreci gerçekleştirmektedir. Robotik sistemlerde, pekiştirmeli öğrenme düşük seviye kontrol ve yüksek seviye kontrol problemlerini çözmektedir. Bu tez çalışmasında robotik sistemlerde düşük seviye kontrol olarak robotun tork kontrol problemi model tabanlı ve modelden bağımsız olarak çözülmüştür. Dinamik sistemlerde kontrol problemini çözmek için fiziksel sistemin matematiksel modeli tam olarak oluşturulmalıdır. Sistemin modelinin tam olarak bilinmediği durumlarda, fiziksel sistem modeli sistem tanılama yaklaşımıyla modellenmektedir. Gerçek dünyada robotik sistemler ve sistem üzerine etki eden bozucu değişken, belirlenemeyen parametreler vardır ve fiziksel sistemin matematiksel modelinin genellikle analitik çözümü bulunmamaktadır. Pekiştirmeli öğrenme fiziksel sistemin modelinden bağımsız olarak öğrenme süreci gerçekleştirdiği için model tabanlı kontrole göre üstünlük sağlamaktadır. Robotik sistemlerde yüksek seviye kontrol için robotun, engellerin bulunduğu ortamda kendisine yol planlamasını oluşturması gerekmektedir. Dinamik engellerin bulunduğu ortam içinde robotun en kısa yol ile hedefe giden yörüngeyi oluşturması gerekmektedir. Bu problem pekiştirmeli öğrenme yaklaşımlarıyla çözülmektedir. Bu tez çalışmasında Q öğrenme algoritmasıyla mobil robotun çalışma uzayında sabit engeller arasında hedefine giden en kısa yolu bulması sağlanmıştır.

6. KAYNAKLAR

Craig, John J., *Introduction to Robotics Mechanics and Control*, Third Edition, United States of America: Pearson Prentice Hall, (2005).

Garcia, Alberto Leon., *Probability, Statistics, and Random Processes for Electrical Engineering*, Third Edition, United States of America: Pearson Prentice Hall, (2005).

Goodfellow, Ian. Bengio, Yoshua. Courville, Aaron., *Deep Learning*, United States of America: MIT Press, (2017).

Hagan, Martin T. Demuth, Howard B. Beale, Mark Hudson., *Neural Network Design Second Edition*, (2014).

Hokayem, Peter Al. Gallestey, Eduardo., *Lecture Notes on Nonlinear Systems and sControl*, ETH Zurich: Springer, (2018).

İplikçi, Serdar., *Optimizasyon Teknikleri*, Pamukkale Üniversitesi Elektrik Elektronik Mühendisliği, (2017).

İplikçi, Serdar., *Sürekli-Zamanlı Sistemlerin Sayısal Benzetimi*, Pamukkale Üniversitesi Elektrik Elektronik Mühendisliği, (2018).

Jun-Di, Sun., Guang-Zhong, Cao., Wen-Bo, Li., Yu-Xin, Liang., Su-Dan, Huang., "Analytical Inverse Kinematic Solution Using the D-H Method for a 6-DOF Robot ", *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, (2017).

Kavukcuoglu, Koray., Mnih, Volodymyr., Silver, David., "Human-level control through deep reinforcement learning", Nature, (2015).

Kingma, Diederik P. Lei Ba, Jimmy., "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION", *International Conference on Learning Representations (ICLR)*, (2015).

Kirk, Donald E., *Optimal Control Theory An Introduction*, United States of America: Prentice-Hall Inc, (1970).

Kochenderfer, Mykel J. Wheeler, Tim A., *Algorithms for Optimization* The MIT Press (2019).

Kolter, J. Zico., *Introduction to Reinforcement Learning*, Carnegie Mellon University, (2016).

Kolter, J. Zico., *Markov Decision Processes*, Carnegie Mellon University, (2016).

Kolter, J. Zico., *Reinforcement Learning*, Carnegie Mellon University, (2016).

Lewis, Frank L. Dawson, Darren M. Abdallah, Chaouki T., *Robot Manipulator Control Theory and Practice*, Second Edition, New York: Marcel Dekker, Transferred to Digital Printing, (2006).

Lillicrap, Timothy P., Hunt, Jonathan J., Silver, David., "Continuous Control With Deep Reinforcement Learning", *International Conference on Learning Representations (ICLR)*, (2016).

Murray , Richard M. Li, Zexiang. Sastry, S. Shankar ., *A Mathematical Introduction to Robotic Manipulation*, (1994).

Jorge, Nocedal. Stephen, Wright., *Numerical Optimization*, (2006).

Pavone, Marco., *Optimal and Learning-based Control - Actor-critic and advanced policy gradient*, Stanford University, (2018).

Pavone, Marco., *Optimal and Learning-based Control - Dynamic programming*, Stanford University, (2018).

Pavone, Marco., *Optimal and Learning-based Control - HJB Equation and Continuous LQR*, Stanford University, (2018).

Pavone, Marco., *Optimal and Learning-based Control - Intro to reinforcement learning*, Stanford University, (2018).

Saha, Subir Kumar. Shah, Suril Vijaykumar. Dutt, Jayanta Kumar ., *Dynamics of Tree-Type Robotic Systems*, New York London: Springer Dordrecht Heidelberg, (2013).

Siciliano, Bruno. Sciavicco, Lorenzo. Villani, Luigi. Oriolo, Giuseppe., *Robotics Modelling, Planning and Control*, London: Springer Verlag, (2009).

Silver, David., *Reinforcement Learning Markov Decision Processes*, Lecture 2, United Kingdom: University College London, (2015).

Siradjuddin, Indrazno., Setiawan, Budhy., Amalia, Zakiyah., Fahmi, Ahmad., "State space control using LQR method for a cart-inverted pendulum linearised model", *International Journal of Mechanical & Mechatronics Engineering (IJMME-IJENS)*, (2017).

Spong, Mark W. Hutchinson, Seth. Vidyasagar, M., *Robot Modeling and Control*, Second Edition, United Kingdom: CPI Antony Rowe, (2020).

Sutton , Richard S. Barto, Andrew G., *Reinforcement Learning An Introduction*, Second Edition, London England: The MIT Press, (2018).

Szepesvári, Csaba., *Algorithms for Reinforcement Learning*, Morgan & Claypool Publishers, (2010).

Zhang, Xiaoqin., Ma, Huimin., “Pretraining Deep Actor-Critic Reinforcement Learning Algorithms With Expert Demonstrations”, *Dept. of EE, Tsinghua University*, (2018).