FIRÇALI DOĞRU AKIM MOTORU ÜZERİNDE ZORLU HARİCİ BOZUCULAR ALTINDA PI KONTROLÖRÜNÜN PERFORMANS ANALİZİ

YÜKSEK LİSANS TEZİ

Mahmut Esat İPEK

Danışman
Prof. Dr. Rıdvan ÜNAL

İkinci Danışman
Doç.Dr. Celal Onur GÖKÇE

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

Eylül 2025

AFYON KOCATEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

FIRÇALI DOĞRU AKIM MOTORU ÜZERİNDE ZORLU HARİCİ BOZUCULAR ALTINDA PI KONTROLÖRÜNÜN PERFORMANS ANALİZİ

Mahmut Esat İPEK

Danışman
Prof. Dr. Rıdvan ÜNAL

İkinci Danışman
Doç.Dr. Celal Onur GÖKÇE

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Eylül 2025

TEZ ONAY SAYFASI

Adı SOYADI tarafından hazırlanan "Tez Onay Sayfası" adlı tez çalışması lisansüstü eğitim ve öğretim yönetmeliğinin ilgili maddeleri uyarınca 10 / 09 / 2025 tarihinde aşağıdaki jüri tarafından **oy birliği** / **oy çokluğu** ile Afyon Kocatepe Üniversitesi Fen Bilimleri Enstitüsü **Anabilim Dalı Adı Anabilim Dalı'nda YÜKSEK LİSANS TEZİ / DOKTORA TEZİ** olarak kabul edilmiştir.

Danışman **:** Rıdvan ÜNAL

İkinci Danışman **:** Celal Onur GÖKÇE

Başkan **:** Ünvanı Adı SOYADI
Üniversite adı, Fakültesi ..... **İmza** .....

Üye **:** Ünvanı Adı SOYADI
Üniversite adı, Fakültesi ..... **İmza** .....

Üye **:** Ünvanı Adı SOYADI
Üniversite adı, Fakültesi ..... **İmza** .....

Üye **:** Ünvanı Adı SOYADI
Üniversite adı, Fakültesi ..... **İmza** .....

Üye **:** Ünvanı Adı SOYADI
Üniversite adı, Fakültesi ..... **İmza** .....

Afyon Kocatepe Üniversitesi
Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
..... /..... /..... tarih ve
……. sayılı kararıyla onaylanmıştır.

…….
Prof. Dr. Bekir YALÇIN
Enstitü Müdürü

BİLİMSEL ETİK BİLDİRİM SAYFASI

Afyon Kocatepe Üniversitesi

Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- Görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- Başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- Atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

10 / 09 / 2025

İmza

Mahmut Esat İPEK

ÖZET

Yüksek Lisans Tezi

FIRÇALI DOĞRU AKIM MOTORU ÜZERİNDE ZORLU HARİCİ BOZUCULAR ALTINDA PI KONTROLÖRÜNÜN PERFORMANS ANALİZİ

Mahmut Esat İPEK
Afyon Kocatepe Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Rıdvan ÜNAL
İkinci Danışman: Doç.Dr. Celal Onur GÖKÇE

Otomatik kontrol sistemleri, modern mühendislik uygulamalarının temel bileşenlerinden biri olup; insansız kara, hava ve deniz araçlarından robotik sistemlere, endüstriyel üretim hatlarından CNC tezgahlarına kadar pek çok alanda yaygın şekilde kullanılmaktadır. Bu sistemlerin güvenilir, kararlı ve verimli çalışabilmesi, bünyelerinde yer alan kontrol mekanizmalarının başarısına doğrudan bağlıdır. Elektrik motorları ise, bu sistemlerde hareketin temel kaynağı olup; özellikle hız ve konum gibi parametrelerin hassas bir şekilde kontrol edilmesi, sistem performansının belirleyici unsurlarındandır. Bu çalışmada, fırçalı bir doğru akım (DA) motorunun hız kontrolü problemi ele alınmış ve sistem üzerinde deneysel uygulamalar gerçekleştirilmiştir. Basit yapıları, uygun maliyetleri ve kolay kontrol edilebilirlikleri nedeniyle DA motorlar düşük ve orta güçlü uygulamalarda yaygın olarak tercih edilmektedir.

Tez kapsamında, motor hız kontrolü için Oransal–İntegral (PI) tipi bir kontrolör tasarlanmış ve uygulanmıştır. PI kontrol yapısının temel amacı, motorun belirlenen referans hıza mümkün olan en kısa sürede, minimum hata ile ulaşmasını sağlamak ve sistemde kararlı bir yanıt elde etmektir. Ancak bu başarım, doğrudan kontrolör kazançlarının doğru belirlenmesine bağlıdır. Bu amaçla, kazançların ($K_p$ve K_i) etkin biçimde ayarlanması için, doğadaki sürü davranışlarını temel alarak çözüm uzayında en

uygun sonucu arayan, çok sayıda çözüm adayını paralel olarak değerlendirebilen Parçacık Sürü Optimizasyonu (PSO) adlı sezgisel bir algoritmadan yararlanılmıştır.

Çalışma iki ana deneysel aşamadan oluşmaktadır. İlk aşamada, kullanılan DA motorun dinamik yapısını temsil eden matematiksel bir modelin çıkarılması hedeflenmiştir. Adım girişlerine karşılık alınan motor hız yanıtları analiz edilmiş ve PSO algoritması yardımıyla sistemin karakteristik parametrelerini yansıtan model başarıyla elde edilmiştir. Modelin geçerliliği, sabit referans sinyali altında sistem tepkileriyle karşılaştırılarak test edilmiş ve yükselme süresi, aşım miktarı, yerleşme süresi ve kararlı durum hatası gibi ölçütler açısından sistem davranışını gerçekçi biçimde yansıttığı görülmüştür.

İkinci aşamada, geliştirilen PI kontrol yapısı Altera Cyclone IV FPGA platformu üzerinde gerçek zamanlı olarak uygulanmıştır. Manyetik enkoder ile sağlanan geri besleme sinyalleri dijital olarak işlenmiş ve sistemin hız kontrolü bu verilere dayalı olarak gerçekleştirilmiştir. Farklı PI kazanç kombinasyonlarının sistem yanıtına etkilerini değerlendirmek amacıyla, her bir durumda elde edilen referans hız (ω_{Ref}) ve gerçek hız ($\omega_{Gerçek}$) verileri Python tabanlı analizler ile karşılaştırılmıştır. Çeşitli kazanç kombinasyonları için yükselme süresi, aşım, yerleşme süresi ve izleme başarımı gibi performans metrikleri kullanılarak karşılaştırmalı değerlendirme yapılmış ve nesnel performans tabloları oluşturulmuştur. Ayrıca kontrol kazançlarının armatür akımı tepe değerleri üzerindeki etkileri ayrıntılı biçimde incelenmiş, sistem performans ölçütleri ile kazançlar arasındaki ilişki ısı haritaları ve grafiklerle görselleştirilmiştir. Bunun yanında, farklı yük koşulları altında, ikinci bir doğru akım motoru yük motoru olarak bağlanarak esas motorun bilmediği farklı PWM gerilimleriyle sürülerek dış bozucu yük simüle edilerek akım sınırlarını gözeten çok amaçlı PSO analizleri gerçekleştirilmiş ve %0–%50 arasındaki çalışma aralığı için ayrıntılı değerlendirmeler yapılmıştır. Böylece kontrolcü, beklenmeyen bozucu etkiler altında da kararlılığını korumuştur. PSO algoritmasının arama süreci de grafiksel olarak açıklanmış, optimizasyon sürecinde bulunan en uygun kazançlar deneysel sonuçlarla desteklenmiştir. Ayrıca PSO ile elde edilen en iyi kazanç seti, el ayarlı ve literatürde önerilen denetçilerle kıyaslanarak değerlendirilmiştir.

Elde edilen sonuçlar, PSO algoritması ile optimize edilen PI kontrol yapısının, fırçalı DA motorun hız kontrolünde yüksek doğruluk ve kararlılık sağladığını ortaya koymuştur. Sistem, hem modelleme hem de kontrol açısından istenen performans düzeyini başarıyla karşılamış, ayrıca kazançların sistem dinamiği üzerindeki etkileri deneysel olarak doğrulanmıştır.

Bu bağlamda, geliştirilen yöntem benzer motor kontrol uygulamaları için uygulanabilir bir çözüm olarak değerlendirilmiş ve kontrol mühendisliği alanındaki literatüre katkı sunacak nitelikte sonuçlar ortaya koymuştur.

2025, xiv + 154 sayfa

Anahtar Kelimeler: Fırçalı DA Motor, Parametre Tahmini, PI kontrol, Parçacık Sürü Optimizasyonu(PSO), Deneysel Çalışma, Sistem Modelleme

ABSTRACT

M.Sc. Thesis

PERFORMANCE ANALYSIS OF PI CONTROLLER ON BRUSH DIRECT CURRENT MOTOR UNDER SEVERE EXTERNAL DISTURBANCES

Mahmut Esat İPEK

Afyon Kocatepe University

Graduate School of Natural and Applied Sciences

Department of Electrical Electronics Engineering

Supervisor: Prof.Dr. Rıdvan ÜNAL

Co-Supervisor: Assoc.Prof Celal Onu GÖKÇE

Automatic control systems are one of the fundamental components of modern engineering applications and are widely used in many areas, ranging from unmanned land, air, and sea vehicles to robotic systems, industrial production lines, and CNC machines. The reliable, stable, and efficient operation of these systems is directly dependent on the success of the control mechanisms they incorporate. Electric motors are the primary source of motion in these systems; the precise control of parameters such as speed and position is a key factor in system performance. This study addresses the speed control problem of a brushed direct current (DC) motor and presents experimental applications on the system. Due to their simple structure, reasonable cost, and ease of control, DC motors are widely preferred in low and medium power applications.

Within the scope of the thesis, a Proportional–Integral (PI) type controller was designed and implemented for motor speed control. The fundamental objective of the PI control structure is to ensure that the motor reaches the specified reference speed in the shortest possible time with minimum error and to obtain a stable response in the system. However, this performance depends directly on the correct determination of the controller gains. To this end, a heuristic algorithm called Particle Swarm Optimization (PSO) was utilized. This algorithm, which searches for the most suitable result in the solution space based on

swarm behavior in nature and can evaluate multiple solution candidates in parallel, was used to effectively adjust the gains (K_p and K_i).

The study consists of two main experimental stages. In the first stage, the goal was to derive a mathematical model representing the dynamic structure of the DA motor used. The motor speed responses corresponding to the step inputs were analyzed, and a model reflecting the characteristic parameters of the system was successfully obtained using the PSO algorithm. The validity of the model was tested by comparing it with system responses under a steady-state reference signal, and it was observed that the model realistically reflected system behavior in terms of criteria such as rise time, overshoot amount, settling time, and steady-state error.

In the second stage, the developed PI control structure was implemented in real time on the Altera Cyclone IV FPGA platform. Feedback signals provided by the magnetic encoder were digitally processed, and the system's speed control was performed based on this data. To evaluate the effects of different PI gain combinations on the system response, the reference speed (ω_{Ref}) and actual speed (ω_{Actual}) data obtained in each case were compared using Python-based analyses. Comparative evaluations were performed using performance metrics such as rise time, overshoot, settling time, and tracking performance for various gain combinations, and objective performance tables were created. In addition, the effects of control gains on armature current peak values were examined in detail, and the relationship between system performance criteria and gains was visualized using heat maps and three-dimensional surface graphs. Furthermore, multi-objective PSO analyses considering current limits under different load conditions were performed, and detailed evaluations were made for the operating range between 0% and 50%. The search process of the PSO algorithm was also explained graphically, and the optimal gains found in the optimization process were supported by experimental results. Furthermore, the best gain set obtained with PSO has been evaluated by comparing it with manually tuned and literature-recommended controllers.

The results obtained reveal that the PI control structure optimized with the PSO algorithm provides high accuracy and stability in the speed control of the brushed DC motor. The system successfully met the desired performance level in terms of both modeling and control, and the effects of the gains on the system dynamics were experimentally verified. In this context, the developed method was evaluated as an applicable solution for similar motor control applications and produced results that contribute to the literature in the field of control engineering.

2025, xiv+ 154 pages

Keywords: Brushed DC Motor, Parameter Estimation, PI Control, Particle Swarm Optimization (PSO), Experimental Study, System Modeling

TEŞEKKÜR

Bu araştırmanın konusu, deneysel çalışmaların yönlendirilmesi, sonuçların değerlendirilmesi ve yazımı aşamasında yapmış olduğu büyük katkılarından dolayı tez danışmanım Sayın Prof.Dr Rıdvan ÜNAL, araştırma ve yazım süresince yardımlarını esirgemeyen Sayın Doç.Dr. Celal Onur GÖKÇE her konuda öneri ve eleştirileriyle yardımlarını gördüğüm hocalarıma ve arkadaşlarıma teşekkür ederim.

Bu araştırma boyunca maddi ve manevi desteklerinden dolayı sevgili aileme teşekkür ederim.

Mahmut Esat İPEK

Afyonkarahisar 2025

İÇİNDEKİLER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler

K_c	Kontrol Edici Kazanç Katsayısı
K_d	Türevsel Kontrolör Kazanç Katsayısı
K_i	Integral Kontrolör Kazanç Katsayısı
K_p	Oransal Kontrolör Kazanç Katsayısı
K_u	Kontrol Edici Son Katsayısı
T_u	Salınım Periyodu

Kısaltmalar

DA	Doğru Akım
PSO	Parçacık Sürü Optimizasyonu

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

1. GİRİŞ

Otomatik kontrol sistemleri, günümüzde endüstriyel otomasyon, robotik, taşıma sistemleri ve üretim hatları gibi birçok alanda yaygın olarak kullanılmaktadır. Bu sistemlerin temel bileşenlerinden biri olan elektrik motorlarının hız ve konum kontrolü, genel sistem performansını doğrudan etkilemektedir.

Fırçalı doğru akım (DA) motorları, özellikle hassas kontrol gerektiren uygulamalarda tercih edilmektedir (Afjei, Ghomsheh & Karami, 2007). Bu tür motorların hız kontrolünde performanslarının optimize edilmesi, sistemlerin genel performansına önemli ölçüde katkıda bulunur. Fırçalı DA motor hız kontrolü ile ilgili yapılmış araştırmalar aşağıdaki gibidir:

Komić ve Dubravić (2020), DA motorlardan elde edilen gerilim ve hız verilerini mikrodenetleyici aracılığıyla MATLAB/Simulink ortamına aktarmış ve motor parametrelerini belirlemek için Parametre Tahmini araç kutusunu kullanarak kademeli bir kontrol sistemi geliştirmiştir. Bu sistemde, iki adet ayarlanabilir PI kontrolörü kullanılmıştır.

Jammousi ve arkadaşları (2020), DA motor için kayan kipli kontrol parametrelerinin optimize edilmesi üzerine, Karınca Kolonisi Optimizasyonu (ACO) algoritmasını kullanarak, dış belirsizlikler ve bozucuların etkisi altında sistem performansını artırmayı hedeflemişlerdir. Simülasyon sonuçları, ACO ile ayarlanan kontrolörün manuel ayarlamalara kıyasla daha iyi performans sergilediğini göstermiştir.

Rodríguez-Abreo ve arkadaşları (2021), DA motorların dinamik modellerinin tahmini için guguk kuşu arama algoritmasını kullanarak, giriş voltajı adımındaki akım ve hız hatasından türetilen bir maliyet fonksiyonu ile parametre tahmininde doğruluğu artırmayı başarmışlardır. Ayrıca bu çalışmada, orijinal Guguk Kuşu Arama algoritması ile Steiglitz–McBride algoritması (Zhu & Hjalmarsson, 2016) karşılaştırılmış ve sonuçlar, değiştirilmiş algoritmanın parametre tahmininde daha iyi performans sergilediğini göstermiştir.

Hafez ve Dhaouadi (2021), DA motor sistemlerinin doğru modellemesi ve kontrolü için mekanik parametrelerin, özellikle atalet momenti ve viskoz sürtünme gibi, belirlenmesinin önemini vurgulamış ve bu parametreleri tahmin etmek için Parçacık Sürü Optimizasyonu (PSO) tekniklerini uygulamışlardır.

Niembro-Ceceña ve arkadaşları (2022), fırçalı ve fırçasız DA motorlarda geri elektromotor kuvvet (EMF) değerlerini tahmin etmek için zaman serisi analizini kullanarak, PID kontrolörlerinin gerçek zamanlı adaptasyonu için bir Otomatik Regresyon Hareketli Ortalama (ARMA) modeli önermişlerdir. Bu tahminler, Oransal-İntegral-Türev (PID) kontrolörlerinin gerçek zamanlı adaptasyonunda kullanılmıştır.

Omar ve arkadaşları (2022), DA motorlar için model referans uyarlamalı kontrolün (MRAC) deneysel uygulamasına odaklanmış ve parametre tanımlaması gerektirmeden bir denetleyici geliştirerek, çevrimiçi parametrik değişikliklere karşı sağlamlık sağlamayı amaçlamışlardır.

Tuấn ve arkadaşları (2023), Sonlu Kontrol Seti-Model Tahminli Kontrol (FCS-MPC) kullanarak fırçalı DA motorların hız kontrolü için bir yöntem sunmuş ve bu yaklaşımın geleneksel PI kontrol cihazlarına kıyasla daha iyi performans sağladığını deneysel olarak göstermişlerdir.

Nguyen ve arkadaşları (2023), gerçek zamanlı bir simülasyon sistemi kullanarak standart PI kontrol cihazı, bulanık mantık kontrol cihazı ve bulanık PI kontrol cihazını karşılaştırmış ve karmaşık modeller içeren senaryolarda bulanık PI denetleyicinin daha fazla fayda sağladığını belirtmişlerdir.

Zuo ve diğerleri (2024), elektrikli sürücülerin hız kontrolü için iki uyarlanabilir PI kontrolörü önermiş ve bu kontrolörlerin mekanik parametreleri başarıyla tanımladığını ve gürültü azaltmada klasik PI denetleyiciyi geride bıraktığını göstermiştir. Hajari ve Ray (2023), DA motor sistemlerinin hassas hız kontrolü için voltaj bazlı bir akım tahmin tekniği sunmuş ve bu tekniğin yüksek bant genişliğine sahip akım sensörlerine kıyasla maliyetleri önemli ölçüde azaltırken doğruluğu artırdığını belirtmişlerdir.

Bu tez çalışmasında, gerçek bir fırçalı DA motor kullanılarak PI kontrolcü ile hız kontrolü gerçekleştirilmiştir. Kullanılan motor, manyetik enkodere sahip olup, dönme pozisyon bilgisini manyetik alandaki değişimler şeklinde algılayarak elektriksel sinyale çevirmektedir. Bu özellik, parametre tahmini ve kapalı çevrim hız kontrolü için daha doğru sonuçlar sağlamaktadır (Beriş ve Taşdelen, 2023).

Modern gömülü sistem uygulamalarında, farklı çalışma gerilimlerine sahip bileşenlerin birbirleriyle haberleşmesini sağlamak önemli bir mühendislik problemidir. Özellikle 3.3V seviyesinde çalışan I²C veya SPI tabanlı sensörlerin, 5V seviyesinde çalışan mikrodenetleyici platformlarına doğrudan bağlanması durumunda, cihazların zarar görme riski bulunmaktadır. Aynı şekilde, 5V seviyesinde çalışan bir birimin, 3.3V gerilim ile çalışan bir sistemle uyumlu çalışabilmesi için de benzer bir dönüştürme gerekliliği ortaya çıkmaktadır. Bu sorunu çözmek amacıyla "mantık seviyesi dönüştürücü" (logic level shifter) olarak adlandırılan ara devreler kullanılır. Bu tezde de çalışma gerilimleri arasındaki farkın çözülmesi için level converter kullanılmıştır. Bu devreler, düşük voltaj seviyesindeki sinyalleri yüksek voltaj seviyesine çıkarabilmekte veya tam tersi şekilde yüksek voltajlı sinyalleri daha düşük voltaj seviyelerine indirgemektedir.

PI kontrolör tasarımında parametrelerin belirlenmesinde, geleneksel Ziegler-Nichols (ZN) yönteminin yerine, daha yeni ve gelişmiş bir yöntem olan Parçacık Sürü Optimizasyonu (PSO) yöntemi tercih edilmiştir. PSO, çok değişkenli ve karmaşık optimizasyon problemleri için etkili bir çözüm sunan meta-sezgisel bir algoritmadır. DA motor kontrolünde PI parametrelerinin belirlenmesinde PSO'nun kullanılması, optimize edilmiş bir performans elde etmek için önemlidir. PSO algoritması, çok boyutlu arama uzayında etkili bir şekilde gezinebilir ve harici bozucuların etkisi altında bile istenen kontrol performansını sağlayacak en iyi PI parametrelerini bulabilir.

PI kontrolörünün K_p, $K_i$ve K_d parametrelerinin doğru şekilde ayarlanması ise DA motorun istenen hız ve konum gibi değişkenlerini kontrol etmek için kritik öneme sahiptir. Bu nedenle, DA motor kontrolü gibi karmaşık sistemlerde PSO tabanlı PI parametre ayarı, endüstriyel uygulamalarda yaygın olarak kullanılabilir. Motorun tahmini parametreleri ile gerçek DA motor üzerinde farklı kontrolör parametreleriyle deneyler yapmak için kullanılmıştır.

DA motorun sahip olduğu manyetik enkoderde bulunan mıknatıs diskinin dönmesiyle birlikte manyetik alanda meydana gelen değişimler sensörü tarafından algılanarak dijital sinyal verilerine dönüştürülmüştür. Elde edilen bu veriler, pozisyon kontrol algoritmaları tarafından işlenerek sistemin hareket parametreleri belirlendiğinden ve mekanik titreşim gibi zorlu çevresel koşullara karşı dayanıklılık gösterdiklerinden deneylerde tercih edilmiştir.

Bu tez çalışmasında gerçekleştirilen iki deneyden ilk olan deneyde DA motor sisteminin dinamik davranışını temsil eden matematiksel model Parçacık Sürü Optimizasyonu (PSO) algoritması ile elde edilmiştir. Model doğruluğu, sabit genlikli referans girişine karşı gerçek sistem (ωGerçek) ve PSO modeli (ωPSO) tepkileri karşılaştırılarak değerlendirilmiştir. Sonuçlar, modelin sistemin temel dinamik özelliklerini doğru şekilde yansıttığını göstermiştir. Ayrıca PSO'nun arama süreci grafiksel olarak incelenmiş, optimizasyon sürecinde bulunan en uygun kazançlar belirlenmiş ve deneysel olarak doğrulanmıştır.

İkinci deneyde, PI kontrolcü Altera Cyclone IV FPGA platformunda gerçek zamanlı olarak uygulanmıştır. Sisteme farklı PI kazanç değerlerinin etkileri incelenmiştir. Her bir kazanç kombinasyonu için referans hız (ωRef) ve gerçek hız (ωGerçek) sinyalleri Python yazılımı ile karşılaştırılmış ve sistem tepkisi analiz edilmiştir. Deneysel gözlemler, karşılaştırmalı performans tablosunda özetlenmiş, yükselme süresi, aşım, yerleşme süresi, kararlılık ve izleme başarımı gibi performans metriklerinin sistem üzerindeki etkileri nesnel biçimde tablolarda da belirtilmiştir.

Ayrıca PI kazanç setlerinin armatür akımı tepe değerleri üzerindeki etkileri de incelenmiş, performans ölçütleri ile PI parametreleri arasındaki bağıntılar, ısı dağılım haritaları ve grafikler aracılığıyla görsel olarak ortaya konmuştur.

Bununla birlikte, motor kontrol sistemlerinde yalnızca nominal yükler altında değil, beklenmeyen dış bozucular karşısında da performansın korunması kritik öneme sahip olduğundan, yük değişimleri altında akım limitlerini dikkate alan çok amaçlı PSO analizleri yapılmış ve %0–%50 çalışma aralığı için ayrıntılı değerlendirmeler gerçekleştirilmiştir.

Yani ikinci bir doğru akım motoru yük motoru olarak sisteme bağlanmış ve PWM sinyalleri ile ters yönde sürülerek esas motor için öngörülemeyen bozucu yükler üretilmiştir. Böylece kontrolcü, 0 doluluk koşullarına göre tasarlanmasına rağmen farklı bozucu senaryolarda da kararlılığını koruyarak sağlamlık(robustluk) göstermiştir.

Yapılan adım sinyalinin uygulanması ve PI kontrol uygulaması deneylerinin bulguları raporlanmış, deneysel çalışmalar sonucunda, kontrol yapısının başarımı ve uygulanabilirliği değerlendirilmiş, kontrol mühendisliği literatürüne katkı sağlanması amacıyla önerilerde bulunulmuştur.

2. LİTERATÜR BİLGİLERİ

DA motorlar, hassas hız ve konum kontrolü gerektiren endüstriyel ve otomotiv uygulamalar başta olmak üzere birçok alanda kullanılmaktadır. Kullanıldıkları sistemlerde modelleme ve parametre tahmininin yüksek doğrulukla yapılması, kontrol performansının optimize edilmesi açısından oldukça önemlidir. DA motorların dinamik davranışını tanımlayan modellerin oluşturulması ve bu modellerin parametrelerinin belirlenmesi, karmaşık ve zamana bağlı değişkenlik gösteren sistemler için zorluk oluşturmaktadır. Bu nedenle, motor parametre tahmini ve kontrol performansını iyileştirmeye yönelik literatürde çeşitli optimizasyon algoritmaları geliştirilmiş olup, özellikle meta-sezgisel yöntemler yaygın olarak kullanılmaktadır. Yapılan çalışmalar, DA motor modellemesi ve parametre tahmininde farklı optimizasyon tekniklerinin doğruluğa etkisini incelemektedir.

Literatürde yer alan bazı önemli çalışmalarda, meta-sezgisel algoritmaların DA motor parametre tahmini ve kontrol uygulamalarındaki rolüne dair çeşitli örnekler sunulmuştur. Örneğin, Rodríguez-Abreo ve ark. (2021), dinamik DA motor modellerinin tahmini için meta-sezgisel Guguk Kuşu Arama algoritması kullanarak bir maliyet fonksiyonu geliştirmiş ve parametre tahmininde doğruluğu artırmıştır. Yapılan karşılaştırmalarda, bu yöntemin orijinal Guguk Kuşu Arama ve Steiglitz-McBride algoritmalarına kıyasla daha üstün performans gösterdiği rapor edilmiştir (Rodríguez-Abreo et al., 2021; Zhu ve Hjalmarsson, 2016).

Hafez ve Dhaouadi (2021), DA motor modellemesi ve kontrolünde özellikle atalet momenti ve viskoz sürtünme katsayısı gibi mekanik parametrelerin önemine dikkat çekmiş ve bu parametrelerin tahmini için geliştirilmiş PSO teknikleri uygulamıştır.

Benzer şekilde, Niembro-Ceceña ve ark. (2022), fırçalı ve fırçasız DA motorlarda PID denetleyici adaptasyonu için EMF değerlerini zaman serisi analiziyle tahmin etmiş ve bu sayede Simulink PID ayarlama aracı ile denetleyici kazançlarını güncellemiştir. Diğer bir çalışmada, Omar ve ark. (2022), parametre tanımlaması gerektirmeyen bir DA motor için Model Referanslı Adaptif Kontrol (MRAC) yöntemi uygulamış ve çevrimiçi parametre değişikliklerine karşı sağlamlık sağlayarak zamandan tasarruf edilmesini sağlamıştır.

Sonuçlar, adaptasyon parametrelerinin nominal değerlere yaklaştığını ve motorun referans hızını başarıyla izlediğini ortaya koymuştur.

Tuấn ve ark. (2023), fırçalı DA motorların hız kontrolü için Sonlu Kontrol Set-Model Tahmini Kontrol (FCS-MPC) yöntemini incelemiş ve bu yöntemin geleneksel PI denetleyicilere kıyasla üstün performans sunduğunu göstermiştir.

Diğer yandan, Nguyen ve ark. (2023), PI, bulanık mantık ve bulanık PI denetleyicilerini karşılaştırmış; PI denetleyicisinin doğru parametreler altında yüksek performans sağladığını, ancak karmaşık modellerde bulanık PI denetleyicinin daha avantajlı olduğunu ifade etmiştir.

Zuo ve ark. (2024), mekanik parametrelerin bilinmediği durumlar için uyarlanabilir PI denetleyiciler geliştirerek, klasik PI denetleyicilere kıyasla gürültü azaltma açısından daha başarılı sonuçlar elde etmişlerdir.

Ayrıca, Hajari ve Ray (2023), voltaj tabanlı bir akım tahmin tekniği sunarak yüksek bant genişliğine sahip akım sensörlerine olan ihtiyacı ortadan kaldırmış ve böylece maliyetleri düşürüp doğruluğu artıran bir yaklaşım geliştirmiştir.

Charkoutsis ve Kara-Mohamed (2023), birinci dereceden artı zaman gecikmeli sistemler için PSO ile ayarlanmış Doğrusal Olmayan PID (NLPID) denetleyici geliştirerek, bu yöntemin geleneksel tekniklere kıyasla daha yüksek performans sunduğunu ortaya koymuştur.

Bunun yanı sıra, Manuel ve ark. (2018), DA motor hız kontrolünde beş farklı meta-sezgisel algoritmayı karşılaştırmış ve özellikle Öğretim-Öğrenme Tabanlı Optimizasyon (TLBO) algoritmasının en hızlı optimum sonuçları sunduğunu belirtmiştir.Ayrıca, bulanık mantık denetleyicisinin diğer yöntemlere göre daha başarılı sonuçlar verdiği de vurgulanmıştır.

Son olarak, Sachan ve Swarnkar (2022), mobil robot hız düzenlemesi için Gri Kurt Optimizasyonu (GWO) algoritmasının üstün performans sağladığını ifade etmiştir.

Benzer nitelikteki çalışmalar, sistem performansını iyileştirmek için metasezgisel algoritmaların ve akıllı kontrol stratejilerinin potansiyelini vurgulamakta olup, literatürde geniş yer bulmaktadır (Khan et al., 2019; Malik et al., 2024; Yousif ve Ali, 2024; Khan et al., 2020; Khan et al., 2023; Khan et al., 2022).

Bu tezde, fırçalı doğru akım (DA) motorunun parametre kestirimi ve optimize edilmiş PI kontrol tasarımı için Parçacık Sürü Optimizasyonu (PSO) yöntemi kullanılmıştır. PSO, yüksek boyutlu ve doğrusal olmayan çözüm alanlarında etkili sonuçlar vermesiyle bilinen sezgisel bir algoritmadır(Kennedy ve Eberhart, 1995). Bu yaklaşım, motorun parametrelerinin doğru biçimde tahmin edilmesini ve PI kontrol kazançlarının (K_p ve K_i) en uygun şekilde belirlenmesini sağlayarak, hız kontrol performansının artırılmasına katkı sunmuştur. Tahmin edilen parametreler doğrultusunda kapalı çevrim simülasyonları gerçekleştirilmiş ve farklı kontrolör kazançları kullanılarak gerçek bir DA motor üzerinde deneysel testler yapılmıştır.Bu çalışmanın özgün yanı, düşük çözünürlüklü bir manyetik enkoder yardımıyla parametre tahmini yapılması ve bu verilerin PI kontrol stratejisine entegre edilmesidir.

Literatürdeki çalışmalarda genellikle yalnızca hız cevabı incelenmiş (Zhang vd., 2018; Kumar ve Singh, 2020), armatür akımı üzerindeki etkiler çoğunlukla göz ardı edilmiştir. Bu tezde ise kontrol kazançlarının armatür akımı tepe değerleri üzerindeki etkileri de incelenmiş, ayrıca performans ölçütleri (yükselme süresi, aşım ve yerleşme süresi) ile kazançlar arasındaki ilişki görselleştirilerek literatürdeki bu boşluk giderilmiştir.

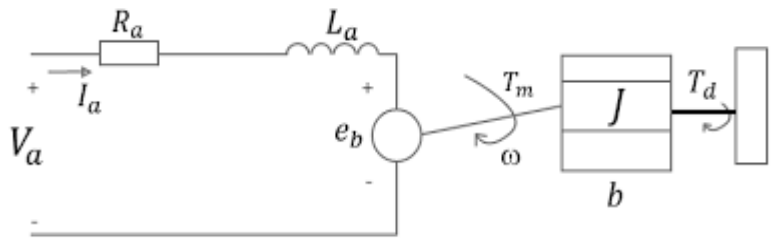
Bunun yanında, tek amaçlı optimizasyon yaklaşımlarının aksine, yük değişimleri altında akım limitlerini gözeten çok amaçlı PSO analizleri yapılmış ve elde edilen sonuçlar detaylı olarak değerlendirilmiştir. Ayrıca PSO'nun arama süreci grafiksel olarak açıklanmış ve optimizasyonun elde ettiği kazançların deneysel doğrulamaları ortaya konmuştur.

Geliştirilen yöntem, düşük maliyetli sensörlere rağmen sistemin performansını koruyabilmekte ve dışsal bozuculara karşı kararlılığı sürdürebilmektedir. Bu durum, özellikle küçük ölçekli sistemlerde önemli avantajlar sunmaktadır.

2.1 DA Motor

Elektrik motorları, elektrik enerjisini mekanik enerjiye dönüştürme aracı olarak stator ve rotor olarak iki ana bileşenden oluşur. Stator, motorun sabit ve sargıların sarıldığı kısım olup gerekli manyetik alanın oluşturulduğu bölümdür. Rotor ise stator sargılarını çevreleyen ve motorun dönen kısmını oluşturan bölümdür.

Elektrik motorları genel olarak kullanılan akım türlerine göre alternatif akım (Alternative Current-AC) ve doğru akım (Direct Current-DA) motorları olarak ikiye ayrılır. DA motorlar, işleyiş biçimlerine göre fırçalı ve fırçasız olarak iki ana türe ayrılır. Şekil 2.1'de DA Motor modeli görülmektedir.



Şekil 2.1 Sabit Mıknatıslı Doğru Akım Motor Modeli (Gökçe, Durusu ve Ünal, 2022)

2.2 Transfer Fonksiyonları ve Blok Diyagramlar

2.2.1 Transfer Fonksiyonu ve Özellikleri

Aktarım işlevi olarak da bilinen transfer fonksiyonu, bir sistemin uyarana muamelesini betimleyen bir işleçtir (Efe, 2024).Yani transfer fonksiyonu, dinamik sistemlerin işleyişi olup dağıtım için kullanılan temel bir araçtır. Bu fonksiyon, bir sistemin giriş ve çıkış değişkenleri arasındaki mesafeyi, sistemin fiziksel özelliklerini, dinamik yapıya dayalı olarak ifade eder.

Transfer fonksiyonu, özellikle doğrusal ve zamana bağlı olmayan sistemlerde etkili bir şekilde uygulanabilir ve bu tür sistemlerin analizi için temel bir yöntemdir. Matematiksel olarak transfer fonksiyonu, bir sistemin Laplace dönüşümü alınmış çıkış değişkeninin, Laplace dönüşümü alınmış giriş değişkenine oranı olarak ifade edilir.

Bu tanımlama, sistemin akışının Laplace dönüşümünü içerir. Transfer kapasiteleri, karmaşık zaman aralığına sahip diferansiyel denklemleri daha anlaşılır bir şekilde analiz etmek için frekans alanında bir araç sağlar.

Literatürde fırçalı doğru akım motorlarının matematiksel olarak modellenmesinde farklı yöntemler kullanılmaktadır. Genel olarak motor, elektriksel ve mekanik alt sistemleri kapsayan ikinci veya üçüncü dereceden transfer fonksiyonlarıyla ifade edilmektedir. Bununla birlikte, kontrol tasarımını sadeleştirmek amacıyla, özellikle hızlı elektriksel dinamiklerin mekanik dinamiklere göre çok daha küçük zaman sabitine sahip olması nedeniyle, birçok çalışmada motorun baskın kutup dikkate alınarak 1. dereceden yaklaşık model ile temsil edildiği görülmektedir. Bu yaklaşım, tasarım sürecini kolaylaştırmakla birlikte sistemin yüksek dereceli dinamiklerini göz ardı etmektedir. Bu tezde ise 3. bölümde ayrıntılı olarak açıklanacağı üzere, motorun tam modeli dikkate alınmıştır.

Transfer fonksiyonunun elde edilmesi genellikle üç ana adımda gerçekleşir:

1. Yönetici Denklemin Belirlenmesi: Sistem için giriş ve çıkış değişkenleri arasındaki aralıklarda farklı diferansiyel denklemler oluşturulur.
2. Laplace Dönüşümünün Uygulanması: Tüm başlangıç koşullarının sıfır olduğu varsayılarak, diferansiyel denklemlere Laplace uygulanır. Bu dönüşüm, zaman alanındaki denklemleri frekans alanına taşır.
3. Transfer Fonksiyonunun Çıkarılması: Elde edilen denklemler, çıkış değişkeninin giriş değişkenliğine dağılımını doğru şekilde verir.

Transfer fonksiyonunun temel özellikleri şunlardır:

- Doğrusallık ve Zaman Bağımsızlık: Transfer fonksiyonu yalnızca doğrusal ve zamana bağlı olmayan sistemler için yapılabilir. Doğrusal olmayan sistemlerde bu kapsam kaybolmaktadır.
- Dürtü Tepkisi: Doğrusal bir sistemin dürtüsü tepkisi, girişe bir birim odaklama işleminde elde edilen sistem çıkışıdır. Transfer fonksiyonu, bu dürtü tepkisinin Laplace görüntülemesiyle ifade edilir.

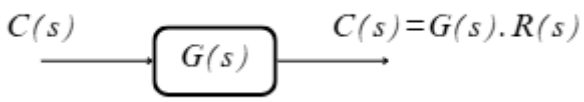
- Başlangıç Koşulları: Transfer fonksiyonları, tüm başlangıç koşullarının sıfırlandığı durumlarda geçerlidir. Bu durumda, sistem analizi kolaylaşarak Laplace dönüşümü ile sistemin incelenmesi mümkün hale gelir.
- Girişten Bağımsızlık: Transfer fonksiyonu, sistem giriş miktarından bağımsızdır; yalnızca sistemin kararlılığına bağlıdır.

2.2.2 Blok Diyagramlar

Blok diyagramları, dinamik sistemlerin genişletilmiş modellerini görselleştirmek ve analiz etmek için kullanılan grafiksel araçlardır (Sarma, 2001). Bu diyagramlar, sistemin giriş ve çıkış değişkenlerini temsil eden bloklar, bu bloklar arasındaki genişleme işlemleri ve açılma yönü olan oklarla gerçekleştirilir.

Bir blok diyagramı, karmaşık denklemlerden oluşan bir sistemin davranışını anlamayı içerir. Bloklar, genellikle çarpma, toplama veya birleştirme gibi ayrıştırma işlemlerini temsil eder. Diyagramdaki oklar, uzatılmanın yalnızca bir yönde hareket ettiğini ve diyagramın tek yönlü bir yapıya sahip olduğunu ifade eder.
Blok diyagram teknikleri, özellikle geri besleme kontrol sistemlerinde yaygın olarak kullanılmaktadır. Bu teknik, doğrusal ifadeleri görselleştirerek, karmaşık sistemlerin içeriğini ve tasarımını değiştirir.

Şekil 2.2'de basit bir birleştirme, genellikle karmaşık blok diyagramlarının temel yapısıdır. Bir kutu, çarpmanın sembolüdür; girdi miktarı, çıktıyı elde etmek için kutudaki işlevle çarpılır. Toplama noktalarını (cebirsel anlamda) gösteren daireler ve çarpmayı gösteren kutular veya bloklar ile, herhangi bir doğrusal matematiksel ifade, Şekil 2.3'te temel bir geri besleme kontrol sistemi için olduğu gibi, blok diyagram ile gösterilebilir.
Şekil 2.2'de formüle bağlı olarak oluşturulan blok diyagramı görülmektedir;



Şekil 2.2 Blok Diyagramın Basit Yapısı

Negatif geri beslemesiz diyagramda çıktı miktarı denklem 2.1'deki gibi ifade edilir:

$$C(s) = G(s).R(s) \tag{2.1}$$

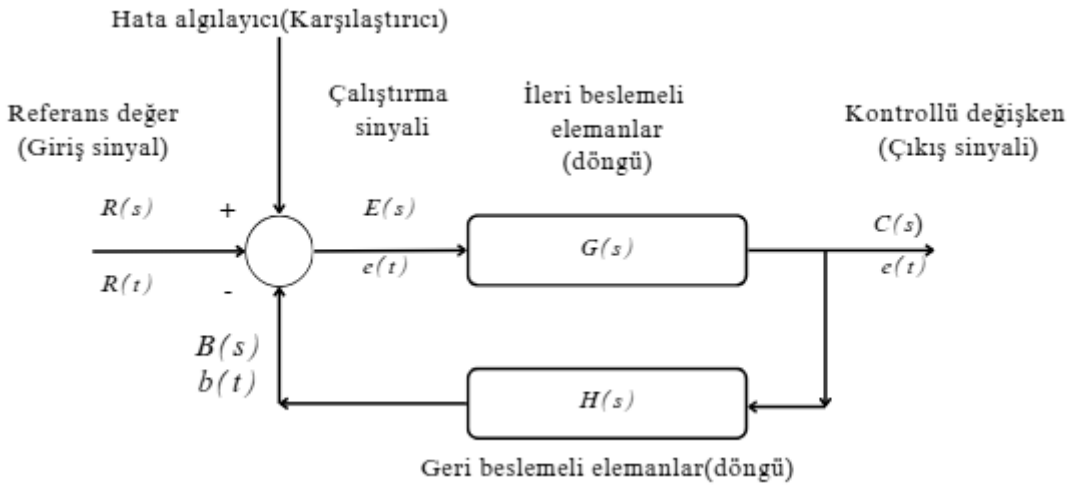
Bir geri besleme kontrol sistemi için negatif geri beslemeli bir yapı ise aşağıda denklem 2.2'deki gibi ifade edilir:

$$C(s) = G(s).[R(s) - H(s).C(s)] \tag{2.2}$$

Burada:

- *C(s):* Sistem çıkışı
- *R(s):* Sistem girişi
- *G(s):* Açık çevrim transfer fonksiyonu
- *H(s):* Geri besleme transfer fonksiyonu

Blok diyagramlarda, analog bilgisayarlarda veya modern dijital işletim yazılımlarında çözümleme için sıklıkla kullanılır. Bu yöntem, sistem özellikleri analitik çözümler olmadan incelemeye olanak tanır. Karşılaştırma ve geri beslemeler eklendikten sonra ise blok diyagramı Şekil 2.3'deki gibi olmaktadır;



Şekil 2.3 Geri Beslemeli Blok Diyagramın Basit Yapısı

Karmaşık geri besleme kontrol sistemlerinin blok diyagramları genellikle birkaç geri besleme döngüsü içerir ve bunlar sistem için genel bir transfer fonksiyonunu değerlendirmek amacıyla basitleştirilmelidir.

2.3 Fırçasız DA Motor

Fırçasız DA motorlar, adından da anlaşılacağı gibi, fırçalar yerine statorun oluşturduğu manyetik alanı elektronik olarak değiştiren motor türüdür. Fırçasız DA motorların gelişim süreci, yarı iletken teknolojisinin ilerlemesi ve elektronik kontrol sistemlerinin gelişmesi ile hız kazanmıştır. Bu tip motorlarda mekanik komütasyon yerine elektronik sürücü devreleri kullanılır, bu da onları daha verimli ve dayanıklı hale getirir.

2.4 Fırçalı DA Motor

Fırçalı DA motorlar, bir DA güç kaynağından beslenen ve dahili bir komütatörü bulunan DA motor türüdür. Fırçalı DA motorlar, elektrik enerjisini mekanik enerjiye çeviren sistemlerde kullanılan ilk motor tipidir ve endüstriyel sistemler, DA dağıtım sistemlerinde yüz yıldan fazladır kullanılmaktadır. Şekil 2.1'de fırçalı DA motorun basit bir modeli gösterilmiştir.

Fırçalı DA Motorların hızı çalışma voltajı ile veya manyetik alanın gücüyle kontrol edilebilir. Fırçalı DA motorlar halen endüstride ve gündelik hayatta çok geniş bir kullanım alanına sahiptir. Ancak, fırçaların sürtünmeden dolayı aşınması ve bakım gerektirmesinden dolayı, güç elektroniği devrelerinin ilerlemesi ile birlikte birçok uygulamada artık fırçasız DA motorlar tercih edilmektedir.

Şekil 2.1'de gösterilen DA Motor modeli, kapsamlı bir analiz için elektriksel ve mekanik olmak üzere iki ana kısımda incelenir. Bu iki kısım, motorun işleyişini daha iyi anlamamızı sağlar.

Elektriksel kısmın matematiksel modelini çıkarırken, Kirchhoff'un Gerilimler Kanunu'ndan yararlanılır. Bu kanun, elektrik devrelerinin analizinde son derece önemlidir. Kirchhoff'a göre, kapalı bir elektrik devresinde harcanan gerilimlerin toplamı, devreye sağlanan kaynak gerilimlerinin toplamına eşittir. Bu, devrenin enerji dengesini koruduğu anlamına gelir. DA Motor modeline ait Kirchoff Kanunu ile denklem 2.3 oluşturulmuştur.

$$V_a = R_a * i_a + L_a \frac{di_a}{d_t} + e_b \quad (2.3)$$

V_a = Armatür Voltajı

i_a = Armatür Akımı

R_a = Armatür Direnci

L_a = Armatür İndüktansı

e_b = Ters indüklenme voltajı

Manyetik alan içinde hareket ettirilen bir telde, hareketin hızı ile orantılı büyüklükte ters elektromotif voltaj indüklenmesi prensibi ile denklem 2.4 oluşturulmuştur;

$$e_b = K_b * \omega \quad (2.4)$$

Manyetik alan içinde üzerinden akım geçmekte olan bir tele, akımın büyüklüğü ile orantılı kuvvet uygulanır prensibine göre denklem 2.5 oluşturulmuştur;

$$T_m = K_m * i_a \quad (2.5)$$

T_m = Motorun oluşturduğu tork

Mekanik kısmı ise Newton'un ikinci hareket yasasına göre modellenebilir. Buna göre oluşturulan denklem 2.6 aşağıdaki gibidir;

$$T_m - T_d = J \frac{d_\omega}{d_t} + b_\omega \quad (2.6)$$

T_d = Dış bozucu yük torku

ω = Rotorun açısal hızı

J = Rotorun eylemsizlik momenti

b = Sürtünme katsayısı

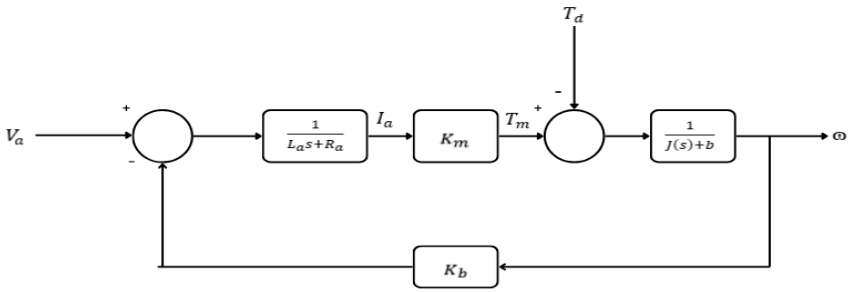
Eşitlik (2.5)'de transfer fonksiyonu bulunursa eşitlik (2.7) elde edilir.

$$\frac{\omega_s}{T_m(s)-T_d(s)} = \frac{K_m}{1+K_m*J(s)} \tag{2.7}$$

Eşitlik (2.3)'in transfer fonksiyonu bulunursa eşitlik (2.8) elde edilir.

$$\frac{i_a(s)}{V_a(s)-e_b(s)} = \frac{1}{R_a+L_as} \tag{2.8}$$

Eşitlik (2.7) ve (2.8)'dan yararlanarak DA Motor için elde edilen blok şeması şekil 2.4'de gösterilmiştir.



Şekil 2.4 Açısal Hızın Endüvi Gerilimi ve Yük Torkuna Göre Değişim Şeması

Motor matematiksel modeline ait elektriksel ve mekanik dinamiklerin yer aldığı diyagramda yer alan değişkenler şu şekildedir;

V_a-Armatür Gerilimi(V):
Motorun armatürüne uygulanan gerilimdir. Sistemin girişidir.

I_a – Armatür Akımı (A):
Armatür devresinden geçen akımdır. Elektriksel kısmın çıkışıdır ve moment üretiminde kullanılır.

L_a – Armatür Endüktansı (H):
Armatür sargısının endüktansıdır. Elektriksel sistemin dinamik davranışına etki eder. Sargıdan geçen akım değiştiğinde, endüktans gerilim indükler.

R_a – Armatür Direnci (Ω):
Armatür devresindeki toplam dirençtir.

Isıl kayıplar ve gerilim düşümü bu direnç üzerinde olur.

$1 / (L_a \cdot s + R_a)$ – Armatür Devresi Transfer Fonksiyonu:
Elektriksel kısmın transfer fonksiyonu. Giriş V_a, çıkış I_a'dır.

K_m – Tork Sabiti (Nm/A):
Motorun elektrik akımını mekanik torka dönüştürme katsayısıdır. $T_m = K_m * I_a$

T_m – Motorun Ürettiği Elektromekanik Tork (Nm):
Armatür akımıyla orantılı olarak üretilen torktur.

T_d – Dış Yük Torku / Bozucu Tork (Nm):
Sisteme etki eden harici veya yük kaynaklı torktur. Sistemi yavaşlatıcı yönde etki eder. Bozucu (disturbance) etkidir.

J – Eylemsizlik Moment (kg·m²):
Rotorun dönmeye karşı gösterdiği dirençtir.

b – Viskoz Sürtünme Katsayısı (Nm·s/rad):
Hızla orantılı mekanik sürtünmeyi temsil eder.

$1 / (J(s) + b)$ – Mekanik Sistem Transfer Fonksiyonu:
Sistemin açısal hıza (ω) karşı verdiği cevabı tanımlar.

ω (omega) – Açısal Hız (rad/s):
Motor milinin dönme hızıdır. Sistemin çıkışıdır.

K_b – Geri Besleme Sabiti (V·s/rad):
Geri besleme (geri elektromotor kuvvet – back EMF) üretim katsayısıdır. Milin hızı ile orantılı olarak ters yönde bir gerilim oluşturur.

Ayrıca geri besleme gerilimi:
$V_{bemf} = K_b * \omega$
İle hesaplanır. Bunlara bağlı olarak sitem davranışı ise şu şekilde olur;

1. Giriş gerilimi V_a, armatür devresine uygulanır.
2. Elektriksel model *(L_a ve R_a)* ile I_a üretilir.
3. I_a, K_m ile çarpılarak mekanik tork T_m elde edilir.
4. Sisteme aynı zamanda bozucu tork T_d etki eder.
5. Toplam tork, eylemsizlik ve sürtünmeye karşı milin hızını belirler.
6. ω geri besleme bloğunda K_b ile çarpılarak çıkışa karşı gelen bir geri EMF oluşturur.
7. Bu EMF, giriş geriliminden çıkarılarak sistemin girişini etkiler.

2.5 Normalizasyon

Normalizasyon, ham veriler üzerinde, her bir özelliğin tekdüze bir katkıya sahip olması için verileri yeniden ölçeklendiren veya dönüştüren bir işlemdir(Singh & Singh, 2020). Yani farklı birimlerde veya ölçeklerde ölçülen verilerin belirli bir ölçeğe getirilmesini amaçlar. Farklı ölçeklerdeki veriler üzerinde işlemler yaparken, bu verilerin etkilerini kıyaslamak ve analiz etmek zorlaşır. Normalizasyon, tüm verilerin aynı referans ölçeğinde olmasını sağlayarak analiz süreçlerini daha sağlıklı hale getirir.
Bu tez çalışmasında normalizasyon işlemi alındıktan sonra elde edilen formüller "*" işareti ile gösterilmiştir. Uygulanan adımlar ise şu şekildedir:

1. Motor Gerilimi ve Akımı Denklemi:

Normalizasyon için tipik referans değerleri V_{ref}, I_{ref}, ω_{ref}, kullanabiliriz:

$$V_a^* = V_a / V_{ref} \tag{2.9}$$

$$i_a^* = i_a / i_{ref} \tag{2.10}$$

$$t^* = t / T_{ref} \tag{2.11}$$

$$e_b^* = e_b / V_{ref} \tag{2.12}$$

Yeniden yazılmış denklem 2.13'deki gibidir:

$$V_{ref}.V_a{}^* = R_a.I_{ref}.i_a{}^* + L_a\frac{I_{ref}}{T_{ref}}\frac{di_a{}^*}{dt^*} + V_{ref}e_b{}^* \tag{2.13}$$

Denklemden V_{ref} sadeleştirildiğinde denklem 2.14 edle edilir:

$$V_a{}^* = \frac{R_aI_{ref}}{V_{ref}}i_a{}^* + \frac{L_aI_{ref}}{V_{ref}T_{ref}}\frac{di_a{}^*}{dt^*} + e_b{}^* \tag{2.14}$$

2. Geri EMF Denklemi ise 2.15'deki gibidir:

$$e_b = K_b\dot{\theta} \tag{2.15}$$

Bu denklemi normalleştirmek için uygulanan denklem 2.16'daki gibidir:

$$\theta^* = \frac{\dot{\theta}}{w_{ref}} \tag{2.16}$$

Yeniden yazılmış denklem ile denklem 2.17 elde edilir:

$$V_{ref}e_b{}^* = K_b\frac{w_{ref}}{T_{ref}}\theta^* \tag{2.17}$$

Buradan $e_b{}^*$'yi çekersek denklem 2.18 elde edilir:

$$e_b{}^* = \frac{K_bw_{ref}}{V_{ref}}\theta^* \tag{2.18}$$

3. Motor Torku formülü denklem 2.19'daki gibidir:

$$T_m = K_mi_a \tag{2.19}$$

Normalizasyon için uygulanan formül denklem 2.20'deki gibidir:

$$T_m{}^* = \frac{T_m}{T_{ref}} \tag{2.20}$$

Bu formüllere göre aşağıdaki denklem 2.21 elde edilir:

$$T_{ref} T_m{}^* = K_m I_{ref} i_a{}^* \tag{2.21}$$

Normalleştirilmiş denklem ise denklem 2.22'de verilmiştir:

$$T_m{}^* = \frac{K_m I_{ref}}{T_{ref}} i_a{}^* \tag{2.22}$$

4. Hareket Denklemi denklem 2.23'deki gibidir:

$$T_m - T_l = J\ddot{\theta} + b\dot{\theta} \tag{2.23}$$

Normalizasyon için uygulanan denklem 2.24 ve 2.25'deki gibidir:

$$T_l{}^* = \frac{T_l}{T_{ref}} \tag{2.24}$$

$$\ddot{\theta}^* = \frac{\ddot{\theta}}{a_{ref}} \tag{2.25}$$

Yeniden yazılmış formül ile denklem 2.26 elde edilir:

$$T_{ref}(T_m{}^* - T_l{}^*) = J\frac{a_{ref}}{T^2{}_{ref}}\ddot{\theta}^* + b\frac{w_{ref}}{T_{ref}}\dot{\theta}^* \tag{2.26}$$

Bu denklemin sadeleştirilmiş hali ise denklem 2.27'deki gibdir:

$$T_m{}^* - T_l{}^* = \frac{Ja_{ref}}{T^2{}_{ref}}\ddot{\theta}^* + \frac{bw_{ref}}{T_{ref}}\dot{\theta}^* \tag{2.27}$$

5. Transfer Fonksiyonları:

Transfer fonksiyonunun normalizasyonu için benzer şekilde parametreleri normalize ederek yapılabilir.

Örneğin, ω(s) ve $V_a(s)$ transfer fonksiyonundaki katsayılar normalize edildiğinde denklem 2.28 elde edilir:

$$\frac{\omega}{V_a(s)} = \frac{K_m{}^*}{L_a{}^* J^*(s)^2 + (R_a{}^* J^* + b L_a{}^*)s + (K_m{}^* K_b{}^* + R_a{}^* b^*)} \tag{2.28}$$

Burada her parametre uygun referanslarla normalize edilir. Normalizasyon, ilgili referans büyüklüklerine bölünerek elde edilir ve bu şekilde tüm denklemler boyutsuz hale getirilmiş olur. Bu denklemler üzerinde daha spesifik referanslar verilirse, daha açık sonuçlar elde edilebilir.

2.6 Dış Bozucu Etkiler

Fiziksel sistemlerin çalışmasını etkileyen çeşitli faktörler vardır, bunlar arasında gürültü veya dış işaretlerin etkisi önemlidir. Bu tür etkilere 'dış bozucular' denir. Bu dış bozuculara karşı kontrol sistemlerinin duyarlı olması, sistemlerin düzgün ve etkili bir şekilde çalışması için gereklidir. Kontrol edilen nesne üzerinde etkili olan ölçülemeyen dışsal bozulma, referans modelin çıktısının sistem çıktısı tarafından izlenmesinde büyük hatalara neden olabilir (Aleksandrov, 2004).

Elektrik motorlarındaki fırça veya komütatör gürültüleri, dış bozucu örnekleri arasında yer alır. Literatürde ‘load’ genellikle nominal, beklenen yükleri tanımlarken; ‘disturbance’ ise öngörülemeyen, ani değişen veya sistem modelinde yer almayan dış etkenler olarak tanımlanmaktadır (Ogata, 2010; Åström ve Hägglund, 2006). Bu çalışmada yük motoru, kapalı çevrim kontrol motorunun bilmediği PWM sinyalleriyle sürülerek disturbance kaynağı olarak değerlendirilmiştir.

Kontrol sistemlerindeki geri bildirim sinyalleri, bu dış bozucuların etkisinin hafifletilmesinde önemli bir rol oynar. Geri bildirim sinyallerinin ayrıntılı işleyişi ve önemi Bölüm 2.5'te tartışılacaktır.

2.7 PID Kontrol

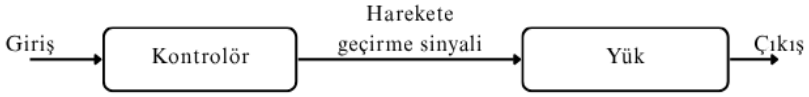
Kontrol sistemleri, cihazların, mekanizmaların veya geniş anlamda sistemlerin davranışlarını kontrol eden unsurlardır.

Bir kontrol sistemi, girişleri kontrol elemanları aracılığıyla alır ve bu girişleri belirli bir şekilde işleyerek çıkışları kontrol eder. Bu durum, sistemlerin istenen bir performans ve davranış çerçevesinde işlemesi için son derece önemlidir.Kontrol sistemleri genel olarak Açık Çevrimli (geri beslemesiz) ve Kapalı Çevrimli (geri beslemeli) olarak iki bölümde incelenir. Bu iki kontrol sistemi türü, kontrol pratiklerinde farklı uygulama ve yaklaşımları temsil eder.

2.7.1 Açık Çevrim Kontrol Sistemi

Açık döngü kontrol sistemi, sistemin çıktısının girdisi üzerinde hiçbir etkisi olmadığı ve bu nedenle herhangi bir geri bildirimi olmadığı bir kontrol sistemi türüdür.

Açık döngü sistemi, sistemin herhangi bir önceki çıktısına veya herhangi bir sürekli çıktısına bağlı olmadığı için geri bildirimsiz kontrol sistemi olarak da bilinir. Bu tür sistemde sisteme girdi sağlanır ve çıktı, çıktıdan herhangi bir geri bildirim alınmadan üretilir. Kritik davranış sergilenmeyecek sistemlerde açık çevrim kontrol sistemleri kullanılır. Buna örnek olarak Şekil 2.5 aşağıda verilmiştir:

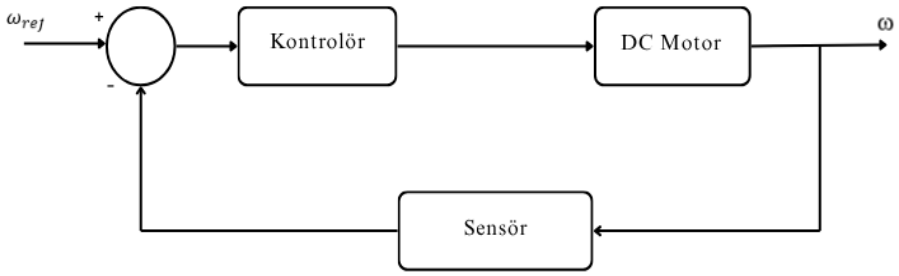


Şekil 2.5 Açık Çevrim Kontrol Sistemi Blok Şeması

Bu sistemler basit ve ekonomik olması sebebiyle karmaşık olmayan uygulamalarda sıklıkla kullanılır.

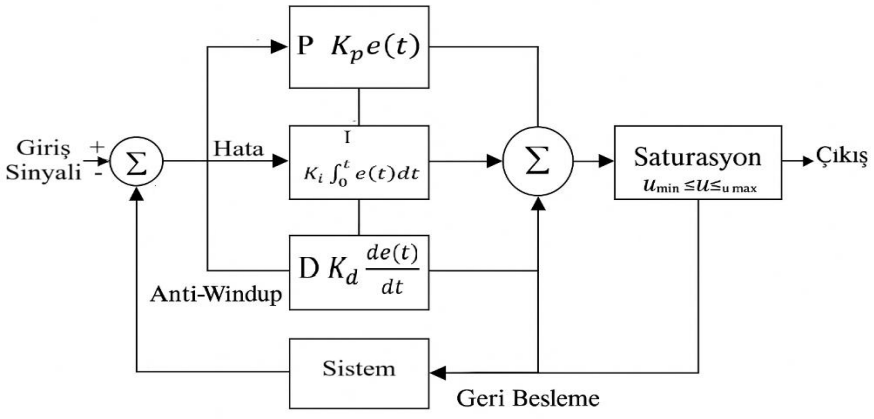
2.7.2 Kapalı Çevrim Kontrol Sistemi

Bir veya daha fazla geri besleme sinyaline sahip kontrol sistemlerine kapalı çevrim kontrol sistemi denir.
Geri besleme sinyali çıkıştan elde edilen sinyali girişe göndererek referans sinyal ile karşılaştırma yapılmasını sağlar. Kapalı çevrim kontrol sistemine ait blok şeması Şekil 2.6'da verilmiştir.



Şekil 2.6 Negatif Geri Beslemeli Kapalı Çevrim Kontrol Sisteminin Blok Şeması

Geri besleme sinyali hatanın azaltılmasının yanı sıra kontrol sistemlerinde kararlılık, bozucu, duyarlılık gibi sistem davranış karakteristiklerinin üzerinde de etkilidir. Kontrolör belirlenirken tüm giriş ve çıkış değişkenleri göz önünde bulundurulmalıdır. Kontrolörün basit olması ve maliyetinin düşük olması beklenir. Karmaşıklığı arttıkça tasarım güçleşir dolayısıyla güvenilirlik azalır. En basit kontrolör olarak Oransal (Proportional = P) kullanılır. Oransal kontrolör, kontrol sinyalini çıkışa sabit bir oranla aktarır. Giriş sinyalinin türev veya integralinden de kontrolör tasarlarken yararlanılabilir. Böylelikle içinde birden fazla eleman bulunan kontrolörler de kullanılabilir. Kontrol sistemlerinde en sık kullanılan kontrolörlerden birisi PID kontroldür. Proportional (P), Integral (I) ve Derivative (D) kelimelerinin baş harflerinden oluşmuş, Oransal-İntegral-Türevsel kontrolör olarak adlandırılır. Şekil 2.7'de PID kontrol blok şeması bulunmaktadır:



Şekil 2.7 PID Kontrol Blok Şeması

PID kontrolde, giriş sinyali ve geri bildirim sinyali karşılaştırılır ve bir hata değeri bulunur.

Bu hata, PID kontrolörü tarafından bir katsayıyla çarpılır, türevi ve integrali alınır ve çıkışa yönlendirilir. Bu süreç, hata değeri en aza indirilene kadar tekrar edilir.

2.7.2.1 PID Kontrol Parametreleri

PID (Orantısal–İntegral–Türevsel) kontrolörleri, endüstriyel otomasyon sistemlerinde yaygın olarak kullanılan ve sistem dinamiklerine göre ayarlanabilen üç temel kazanç parametresi içerir.

Bunlar orantısal kazanç (K_p), integral kazanç (K_i) ve türevsel kazançdır (K_d). Bu parametrelerin doğru şekilde ayarlanması, sistemin kararlılığı, yanıt hızı ve hata düzeyleri üzerinde doğrudan etkilidir. K_p , sistemin mevcut hata değerine orantılı bir tepki üretir. Yüksek bir K_p değeri, sistemin hataya karşı daha hızlı tepki vermesini sağlar; ancak aşırı yüksek K_p değerleri sistemde salınımlara ve kararsızlığa yol açabilir. Düşük K_p değerleri ise sistemin yavaş tepki vermesine neden olabilir ve istenilen set noktasına ulaşma süresini uzatabilir (Zhang et al,2024). Frekans alanında K_p terimi sabittir ve tüm frekanslarda aynı etkiye sahiptir. PID parametrelerinin frekans cevabı üzerinden tanımlanması, sistemin salınım (osilasyon) davranışlarını anlamamıza doğrudan yardımcı olur. PID kontrolörünün transfer fonksiyonu frekans alanında denklem 2.29'daki gibi ifade edilir:

$$C(j\omega) = K_p + \frac{K_i}{j\omega} + K_d.j\omega \tag{2.29}$$

Denklemde görüleceği üzere:

K_p sabit olup frekansdan bağımsızdır.

$\frac{K_i}{j\omega}$ terimi düşük frekanslarda baskındır.

$K_d.j\omega$ terimi ise yüksek frekanslarda etkili olur.

K_i, zaman içinde biriken hata değerlerini toplar ve bu toplamı kontrol çıkışına ekler. Bu sayede, sistemde sürekli olarak kalan kararlı durum hataları (steady-state error) giderilir. Ancak, K_i değerinin fazla yüksek olması, sistemde aşırı düzeltmelere ve dolayısıyla salınımlara neden olabilir (Suseno ve Ma'arif, 2021).

K_i parametresinin etkisi, düşük frekanslarda sistem kazancını artırmasıdır. Bu da sabit hata düzeylerinin zamanla sıfırlanmasına katkı sağlar. K_i'nin frekansa bağlı crossover frekansı (Kazancın K_p ile eşit olduğu frekans) denklem 2.30'daki gibi hesaplanabilir:

$$f_{ic} = \frac{K_i}{2\pi K_p} \tag{2.30}$$

Bu frekansta integral kazanç, orantısal kazanca eşit hale gelir ve sistemin düşük frekanslı dinamiklerine olan etkisi azalır.

K_d, hata sinyalinin değişim hızına tepki verir ve bu sayede sistemin gelecekteki davranışını tahmin ederek aşırı düzeltmeleri engeller. Bu, sistemin daha kararlı ve hızlı bir şekilde istenilen değere ulaşmasını sağlar. Ancak, K_d'nin yüksek olması, özellikle gürültülü sinyallerde, sistemin aşırı hassasiyet göstermesine ve istenmeyen tepkilere yol açabilir (Zhang et al, 2024). K_d'nin frekansa bağlı crossover frekansı (Türevsel kazancın K_p ile eşit olduğu nokta) denklem 2.31'deki gibi hesaplanabilir:

$$f_{dc} = \frac{K_p}{2\pi K_d} \tag{2.31}$$

Bu frekansın üzerindeki bölgelerde türevsel bileşen baskın hale gelir. Bu nedenle, türevsel kazancın çok yüksek seçilmesi durumunda sistem yüksek frekanslı gürültülere karşı daha duyarlı hale gelir.

Bu yüzden katsayılar, belirli karakteristikler göz önünde bulundurularak belirlenir. Oransal (P) terimindeki katsayı (K_p), hatayı düşürmek için hatayı çarpar. K_p değeri yüksek tutulmaz çünkü osilasyon olasılığını artırır. İntegral (I) terimi, hatanın alanını hesapladığı için her döngüdeki hata, katsayı (K_i) ile çarpılarak toplanır. Türev (D) terimi ise değişimle ilgilidir. Sistemdeki değişimi tespit eder ve hedef değeri korumak için yavaşlatır. Eğer hata değerinde bir değişiklik yoksa, türev sıfır olur ve bir etkisi olmaz.
PID kontrolör parametrelerinin frekansla ilişkili bu analizleri, sistemin istenen performansı göstermesi için özellikle frekans alanında kazanç planlamasının yapılmasını gerektirir.

Bu bağlamda, sistemin Bode diyagramı üzerinden PID parametrelerinin hangi frekans aralıklarında baskın olduğunu analiz etmek mümkündür.

Frekans alanındaki bu davranış, aynı zamanda sistemin kararlılığı ve bant genişliği üzerinde de doğrudan belirleyici rol oynar. PID parametrelerinin ayarlanmasında çeşitli yöntemler kullanılmaktadır. Ziegler–Nichols yöntemi, bu alanda bilinen en ampirik yöntemlerden biridir. Bu yöntemlere ilerleyen bölümlerde değinilmiştir.

2.8 Ziegler- Nichols Metodu

PID kontrol tasarımında, kontrol elementlerinin tasarlanmasında oldukça yaygın olarak kullanılan bir yöntemdir. Bu yöntem, John Ziegler ve Nathaniel Nichols tarafından 1942 yılında, PID kontrolör tasarımı için etkili bir yöntem olarak sunulmuştur.

Bu metodun temel prensiplerinden biri, integral zaman sabitinin ve türev zaman sabitinin sıfıra eşitlenmesidir. Bu yöntemde, kontrol edici kazancı, sistem çıkışının sürekli bir salınım yapmasını sağlayacak şekilde arttırılır. Bu sürekli salınım durumu gözlendiğinde, salınımın periyodu ve kontrol edicinin son kazancı kaydedilir. Bu periyot, T_u olarak adlandırılırken, kontrol edicinin son kazancı K_u olarak belirtilir. Kaydedilen bu değerler, Ziegler-Nichols metodunda kullanılan tablo yardımıyla kontrolör için gerekli olan değerlerin belirlenmesinde kullanılır.

Yani, bu kaydedilen değerlerin, kontrolörün hangi tip olduğuna göre belirlenen katsayıları belirlemek için kullanıldığı Çizelge 2.1'den görülebilir. Bu şekilde, kontrolör tasarımı daha etkin ve uygun bir şekilde gerçekleştirilmiş olur.

Çizelge 2. 1 Ziegler-Nichols Metodu Tablosu (Ziegler ve Nichols 1993).

Kontrol Tipi	K_p	T_i	T_d	K_i	K_d
P	$0.5K_u$	-	-	-	-
PI	$0.45K_u$	$0.80T_u$		$0.54K_u/T_u$	-
PD	$0.8K_u$	-	$0.125T_u$	-	$0.10K_uT_u$
PID	$0.6K_u$	$0.5T_u$	$0.125T_u$	$1.2K_u/T_u$	$0.075K_uT_u$

Küçük değişiklikler olmakla birlikte bu yöntem günümüzde halen kullanılmaktadır.

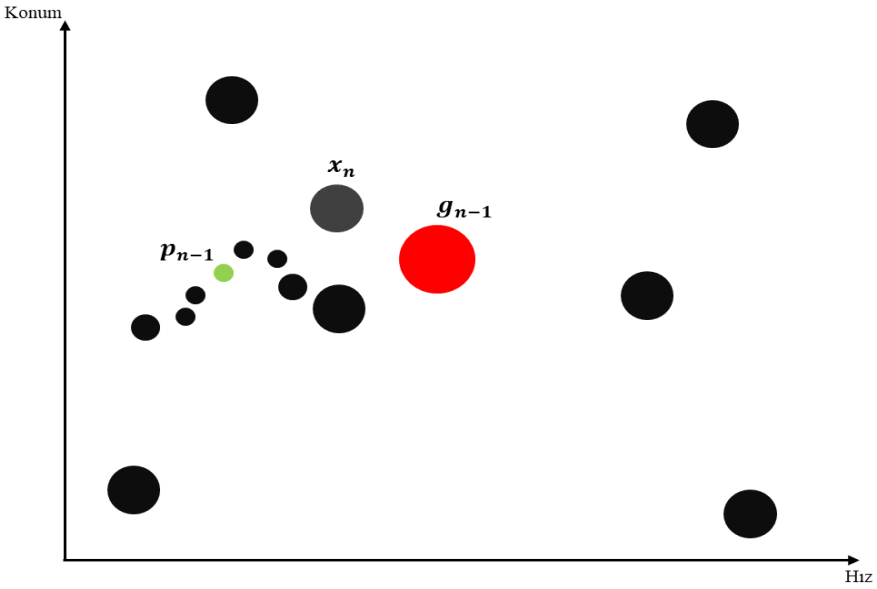
2.9 Parçacık Sürü Optimizasyonu

Optimizasyon, belirli şartlar ve sınırlamalar altında, mevcut olan seçenekler arasından en uygun veya en etkili olanını seçme işlemine denir. Bu terim, genellikle bir dizi alternatif arasında en iyi sonucu elde etmek için en uygun çözümü bulmak anlamında kullanılır.

Optimizasyon problemleri, belirli sınırlar içinde parametre değerlerinin bulunmasını gerektiren her türden problemi ifade eder. Bu tür problemler, genellikle bir sistem veya prosesin performansını en üst düzeye çıkarmak için kullanılır. Doğada, tek başına hareket etmekte zorlanan ancak toplu olarak hareket ettiklerinde zeki davranışlar sergileyen canlılar bulunmaktadır. Bu canlılar, sürü olarak adlandırılan topluluklar oluşturur. Bu sürüye ait bireyler, sadece kendi deneyimlerinden değil, aynı zamanda sürüdeki en başarılı bireyin davranışlarından da ders çıkarırlar.

Bu bireylerin eylemleri ve davranışları, gelecekte karşılaşacakları problemleri çözmede önemli bir araç olarak kullanılır (Akyol ve Alataş 2012). Parçacık Sürü Optimizasyonu (PSO), kuşlar, arılar ve diğer sürü halinde hareket eden hayvan türlerinin beslenme ve diğer temel ihtiyaçlarını karşılamak için sergilediği davranışlardan esinlenmiştir. Sürüdeki bireylerin hareketleri, diğer bireyleri etkiler ve bu da sürünün genel hedefine daha hızlı ve daha etkin bir şekilde ulaşmasını sağlar. Bu konsept, 1995 yılında Elektrik Mühendisi Russell C. Eberhart ve Sosyal Psikolog James Kennedy tarafından tanıtılmıştır.

PSO, genetik algoritmalar ve diğer evrimsel hesaplama teknikleri ile benzerlikler taşımasına rağmen, daha basit bir yapıya sahip olması bu algoritmayı daha kullanışlı ve uygulanabilir kılar. PSO algoritması, karmaşık arama uzaylarında optimal çözümler bulma konusundaki etkinliği nedeniyle, kontrol ve optimizasyon alanındaki çalışmalarda sıklıkla kullanılmaktadır. PSO (Parçacık Sürü Optimizasyonu) algoritmasının uygulamalardaki kullanımlarında temel mantığı, arama uzayında rastgele başlatılan noktaların, o ana kadar en başarılı olan bölgeye doğru yönlendirilmesini sağlamaktır. Bu mantık, birçok optimizasyon probleminin çözümünde etkili bir yaklaşım oluşturur (Gökçe vd. 2021). Şekil 2.8'de PSO çalışma mekanizmasına ait görsel bulunmaktadır.



Şekil 2.8 PSO Çalışma Mekanizması (Gökçe, Durusu ve Ünal, 2022).

Şekil 2.8'de, arama uzayında rastgele başlatılmış parçacıkların temsili bir görüntüsü verilmiştir. Her bir parçacık, optimize edilmek istenen parametreleri içeren bir vektör şeklinde ifade edilir. Bu parçacıklar, algoritmanın başlangıcından itibaren her iterasyonda güncellenir ve bu sürecin sonunda makul bir performansa yakınsamaları beklenir. Her bir parçacığın bir hızı vardır ve bu hız, parçacığın anlık konumu ve bir sonraki konumunu belirler. Her iterasyonda, parçacığın anlık hızı ve bir önceki iterasyonda hesaplanan hızı toplanarak yeni konum elde edilir. Bu süreç, algoritmanın etkin bir şekilde arama uzayında gezinmesini ve en iyi çözüme yakınsamasını sağlar.

Denklem 2.32'de parçacık konumuna ait denklem verilmiştir:

$$xn = xn - 1 + v_n \tag{2.32}$$

Denklemde:

x_n : Parçacığın konumu

v_n : Parçacığın hızı

v_n değeri hesaplanırken üç durum göz önünde bulundurulmalıdır:

- Parçacığın Momentumu (son iterasyondaki hızı): v_{n-1}

- Parçacığın anlık bulduğu en iyi noktanın konumu: p_{n-1}
- Tüm parçacıkların anlık bulduğu en iyi noktanın konumu g_{n-1}

Bunlara bağlı olarak oluşturulan denklem 2.33 ise şu şekildedir:

$$v_n = c_1 v_{n-1} + c_2(x_{n-1} - p_{n-1}) + c_3(x_{n-1} - g_{n-1}) \quad (2.33)$$

Bu ifade, bir parçacığın n. iterasyondaki hızını belirlerken üç temel bileşeni dikkate alır:

Eylemsizlik terimi $c_1 v_{n-1}$: Parçacığın önceki hızının etkisini sürdürmesini sağlar, böylece hareketin sürekliliği korunur.

Bireysel (bilişsel) terim $c_2(x_{n-1} - p_{n-1})$: Parçacığın kendi en iyi konumuna olan mesafesini dikkate alarak, kendi deneyiminden öğrenmesini temsil eder.

Toplumsal terim $c_3(x_{n-1} - g_{n-1})$: Sürünün en iyi konumuna olan mesafe ile parçacığın topluluktan edindiği bilgiye göre yönlenmesini ifade eder. Bu yapı, her bir parçacığın kendi geçmiş deneyimlerine ve topluluğun ortak bilgisine dayanarak karar verdiği dinamik bir öğrenme sürecini modellemektedir. Böylece hem keşif (exploration) hem de sömürü (exploitation) dengelenmiş olur. Denklemdeki katsayılar c_1, c_2, c_3 bu öğrenme davranışlarının ağırlığını belirler ve algoritmanın genel performansını doğrudan etkiler.
Parçacık Sürü Optimizasyonu ve benzer optimizasyon yöntemlerine dair çalışmalar da şu şekildedir:

Zahir ve ark. (2018), fırçalı doğru akım (DA) motorların hız kontrolünde PID denetleyici parametrelerinin belirlenmesinde Genetik Algoritma (GA) yöntemini kullanmış ve bu yöntemin Ziegler-Nichols yöntemine kıyasla daha iyi performans sergilediğini göstermiştir.

Yazgan ve ark. (2019), Genetik Algoritma(GA) ve Parçacık Sürü Optimizasyonu (PSO) algoritmalarını kullanarak PID parametrelerini optimize etmiş ve bu iki yöntemi beş farklı performans kriteri üzerinden karşılaştırmıştır.

Fırçasız DA motor üzerinde yapılan deneysel çalışmalar, PSO algoritmasının GA'ya göre daha başarılı sonuçlar verdiğini ortaya koymuştur.

Song ve ark. (2020), fırçasız DA motorlarda kullanılan bulanık PI kontrolör parametrelerinin optimizasyonu için PSO ile Yerçekimsel Arama Algoritması'nı (GSA) birleştiren bir yaklaşım önermiştir. Bu bütünleşik yöntemin, başlangıçta rastgele seçilen parametrelerle daha hızlı optimizasyon sağladığı belirtilmiştir.

Mohamadwasel ve Bayat (2019), PSO destekli bir Bulanık Mantık algoritmasıyla PD kontrolör parametrelerini optimize etmiş ve bu yöntemin sistem performansı üzerindeki olumlu etkilerini vurgulamıştır.

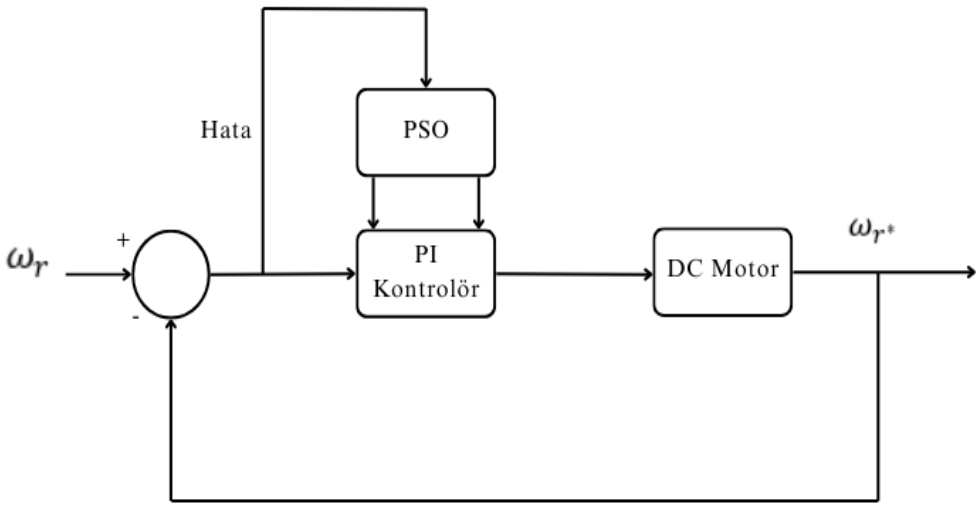
Qi ve ark. (2019), stokastik(olasılıksal süreç) gecikmelere maruz kalan bir DA motorun PID parametrelerini PSO ile optimize etmiş ve elde edilen sonuçları deneysel verilerle doğrulayarak yöntemin etkinliğini ortaya koymuştur.

Idir ve ark. (2018), DA motorların hız kontrolü için hem klasik PID hem de kesirli dereceli PID (FOPID) denetleyici parametrelerini optimize etmek amacıyla PSO ve Diferansiyel Evrim (DE) algoritmalarını kullanmıştır. Çalışmada, performans ölçütü olarak integral-zaman mutlak hata (ITAE) kullanılmış ve bu değerin en aza indirilmesi hedeflenmiştir.

3. MATERYAL ve METOT

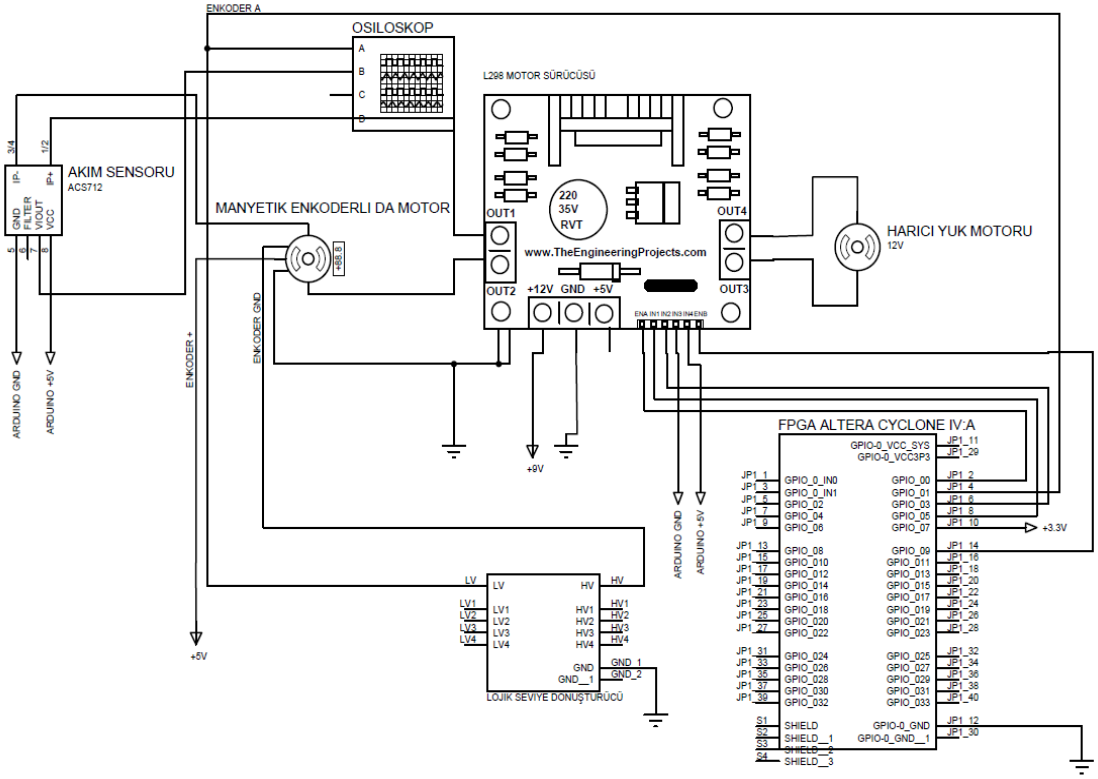
3.1 Kullanılan Sistemin Genel Görünümü

Kullanılacak sisteme ait genel blok şema Şekil 3.1'deki gibidir. Sistem kapalı döngü kontrol ile çalışmaktadır. DA motordan elde edilen açısal hız verisi, normalizasyonu alınmış motora ait açısal hız değeri ile işleme sokularak hata sinyali üretilmiştir. Üretilen hata sinyali fonksiyondan geçirilerek Parçacık Sürü Optimizsayonuna (PSO) sokulmuş ve optimizasyon ile motora ait olan iki parametre K_p ve K_i ile PI kontrolör çalıştırılıp motor hızı kontrol edilmiştir. Sistemde 1 adet 12V DA manyetik enkoderli Doğru Akım motoru, enkoder kullanılarak yapılacak ölçümler için 1 adet dijital osiloskop, sistemi beslemesi için 1 adet doğru akım güç kaynağı, düzenli bir gerilim verilebilmesi için 1 adet regülatör, PI kontrolün uygulanabilmesi için Altera Cyclone IV FPGA ve FPGA ile yapılacak bağlantıların güvenli olması için lojik seviye dönüştürücü, motorun sürülebilmesi için L298N motor sürücü kartı,motorun armatür akımının ölçülmesi için ACS712 akım sensörü, ölçümlerin aktarılıp kaydedilmesi ve optimizasyonun uygulanması için de bilgisayar kullanılmıştır.



Şekil 3.1 PSO Tabanlı PI Kontrollü DC Motorun Hız Kontrolü Blok Diyagramı Şeması

Sistemin genel bağlantı şeması ise aşağıda Şekil 3.2'deki gibidir;



Şekil 3.2 FPGA Tabanlı DC Motor Kontrol Devresi

Şekil 3.2'de sistemde yer alan bağlantılarda;

Motor, üzerinde yer alan enkoder sayesinde konum ve hız bilgisi üretmektedir. Enkoder çıkışları "ENKODER A" ve "ENKODER B" olarak iki adet dijital darbe sinyali üretir. Ölçümler için sadece ENKODER A çıkışı kullanıldı. Bu sinyaller, FPGA'ye bağlanmadan önce 5V seviyesinden 3.3V seviyesine dönüştürülür. Enkoder A çıkışı osiloskopa bağlanmıştır. Enkoder sinyalleri, dijital mantık seviyelerinin uyumsuzluğunu önlemek için logic level converter üzerinden FPGA'ye yönlendirilmiştir.

Motoru iki yönlü sürmek için kullanılan L298N sürücü modülünde OUT1 ve OUT2 uçları motora bağlanmıştır. IN1 ve IN2 sinyalleri FPGA üzerinden kontrol edilerek motorun yönü belirlenmiştir. ENA pini, motorun PWM sinyali ile hız kontrolünün yapılmasını sağlamıştır. Bu PWM sinyali FPGA üzerinden üretilmiştir. +12V ve GND pinleri motorun güç beslemesini sağlar. +5V çıkışı, logic converter ve enkodere güç sağlamak amacıyla kullanılmıştır.

Motorun armatür akımı, Hall etkisi tabanlı ACS712 akım sensörü üzerinden izlenmiştir. Sensör, motor akımının geçtiği hat üzerine seri yerleştirilmiş ve akım yönü sensör üzerinde belirtilen ok doğrultusunda (IP+ ile IP− arasında) olacak şekilde bağlanmıştır. Besleme ucu VCC, Arduinu Uno R3'ün +5V çıkışına, toprak ucu ise yine Arduino Uno R3'ün GND ucu ile sistem toprağına ortaklanmıştır. Arduino sadece güç vermesi amacıyla kullanılmış ve herhangi bir kod çalıştırılmamıştır. Sensörün VOUT çıkışı ölçüm için dijital osiloskopa yönlendirilmiştir.

FPGA, sistemin merkezinde yer alan kontrol ünitesidir. PI kontrol algoritması bu platformda çalıştırılmıştır.

FPGA üzerinde GPIO pinleri aracılığıyla: Enkoder A ve B sinyalleri okunmuştur. PWM sinyali (ENA) ve yön sinyalleri (IN1-IN2) motor sürücüsüne gönderilmiştir. FPGA pin bağlantıları ise IN1, IN2 ve ENA pinleri: GPIO_00, GPIO_01, GPIO_02 Enkoder A ve B sinyalleri: GPIO_03, GPIO_04 gibi giriş pinlerine yönlendirilmiştir. FPGA'nin mantık seviyesi 3.3V olduğundan, enkoderdan gelen 5V sinyaller logic level converter ile dönüştürülmüştür. LV tarafı FPGA'ye (3.3V) ve HV tarafı enkodere (5V) bağlanarak voltaj uyumsuzluklarının önüne geçilmiştir. Bu devre, sinyal bütünlüğünü ve FPGA'nin zarar görmesini önlemek için kritik bir bileşendir.

Sistemde motorun altında çalıştığı yükü simüle etmek amacıyla harici bir 12V yük olan DA Motor bağlanmıştır. L298 üzerinde ENB pini motorun PWM sinyali ile kontrol edilebilmesi için, ana motor yönüne ters dönecek şekilde bağlantıları FPGA üzerinde yapılmış ve kapalı çevrim kontrolcüye bildirilmeden değişen yük momentleri uygulanmıştır. Böylece sistemde öngörülemeyen bozucu etkiler simüle edilmiş ve kontrol algoritmasının dışsal yük etkisi altında nasıl performans gösterdiğini gözlemlemek için kullanılmıştır.

Enkoder çıkışları osiloskopa yönlendirilerek sinyal kalitesi ve faz ilişkisi incelenmiştir. Ayrıca kontrol algoritmasının doğruluğu ve tepki süresi bu sinyaller üzerinden analiz edilmiştir. L298N motor sürücüsü ve motor için 12V DA güç kaynağının 9V kademesi kullanılmıştır.

FPGA ve enkoder için de +5V ve +3.3V besleme gerilimleri aynı güç kaynağının farklı kademelerinden sağlanmıştır.

Sistemde ACS712 akım sensörü ile ölçülen akıma bağlı olarak manyetik alanı algılayarak orantılı bir çıkış voltajı üretilmiştir.

Bu çıkış gerilimi, sensörün veri sayfasında belirtilen katsayı kullanılarak akım değerine dönüştürülmüştür. Akım sensörü çıkışında, sıfır akımda $\frac{V_{CC}}{2}$ seviyesinde (yaklaşık 2.5 V) ofset gerilimi oluşmaktadır. Armatür akımı, bu ofsetten sapma miktarının sensörün duyarlılık katsayısına bölünmesiyle hesaplanmıştır. Kullanılan sensör ±5 A ölçüm aralığına sahip ACS712-05B olup duyarlılığı 185 mV/A'dır. Akım hesaplaması denklem 3.1'deki gibidir:

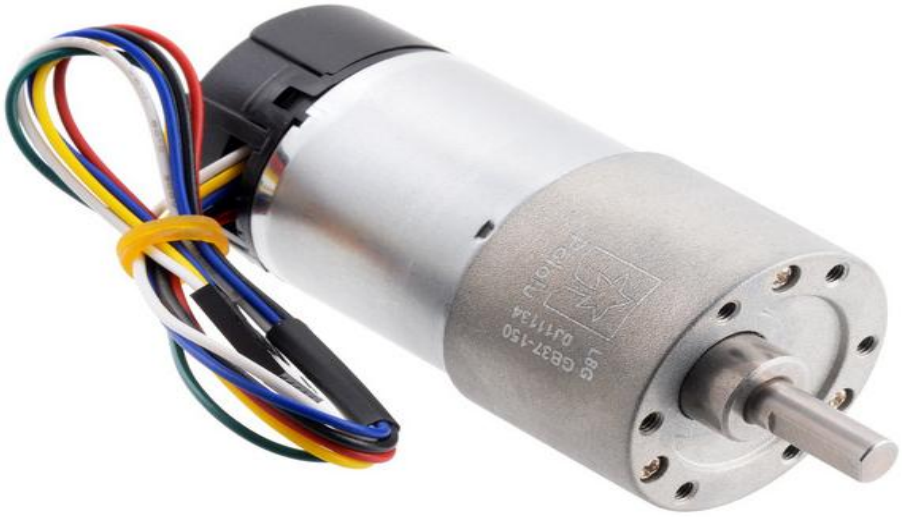
$$\dot{I}_{a(t)} = \frac{V_{OUT(t)} - 2.5}{0.185} \tag{3.1}$$

bağıntısı ile gerçekleştirilmiştir. Bu voltaj FPGA kartına aktarılmış ve osiloskop üzerinden gözlemlenerek kayıt altına alınmıştır. Böylece akım değerleri, PI kontrolörün performans değerlendirmesinde ve sistemin dinamik analizlerinde kullanılmıştır. Sistemde yer alan bileşenlerle ilgili ayrıntılı bölümler ise sırasıyla ilerleyen bölümlerde verilmiştir.

3.2 Manyetik Enkoderli Fırçalı DA Motor

Bu tezde DA motor olarak, 150:1 metal dişli kutusu ve motor şaftının devir başına 64 sayım (CPR) ve dişli kutusunun çıkış şaftının 9600 CPR çözünürlüğüne sahip olduğu entegre bir kare kodlayıcılı güçlü bir 12V fırçalı DA motor kullanılmıştır. Dişli kutusu esas olarak mahmuz dişlilerinden oluşur, ancak azaltılmış gürültü ve iyileştirilmiş verimlilik için ilk aşamada helisel dişliler içerir.

Bu üniteler 16 mm uzunluğunda, 6 mm çapında D şeklinde bir çıkış şaftına sahiptir. Şekil 3.3'de manyetik enkoderli DA motora ait görsel bulunmaktadır.



Şekil 3.3 150:1 Metal Gearmotor 37Dx73L mm 12V with 64 CPR Encoder(İnt.Kyn.2)

Motora ait diğer temel özellikler aşağıda tablo olarak Çizelge 3.1'de verilmiştir.

Çizelge 3.1 Manyetik enkoderli DA motora ait temel özellik tablosu

Nominal Gerilim	Durma Akımı	Yüksüz Akım	Dişli Oranı	Yüksüz Hız (RPM)	Ekstrapolasyonlu durma torku		Max Güç(W)
					(kg.cm)	(oz. in)	
12 V	5.5 A	0.2 A	150:1	67	49	680	6*

Ayrıca motorun çıkışlarına ait kabloların bilgisi de aşağıda Çizelge 3.2'de verilmiştir.

Çizelge 3.2 Motora ait kablo fonksiyonları

Renk	Fonksiyon
Kırmızı	Motor Gücü(Pozitif Terminal)
Siyah	Motor Gücü(Negatif Terminal)
Yeşil	Enkoder Gnd Ucu
Mavi	Enkoder Vcc Ucu(3.5 V- 20v)
Sarı	Enkoder A Çıkışı
Beyaz	Enkoder B Çıkışı

Yapılan deneylerde motorun kırmızı ve siyah uçları güç kaynağına bağlanmış ve motora 9V seviyesinde gerilim uygulanmıştır. Yeşil ve mavi çıkışlar ise adım sinyali uygulama deneyinde L298N kartına bağlanırken PI kontrol uygulamasında lojik seviye dönüştürücünün LV seviyesinden alınan çıkışlar ile FPGA 3.3V seviyesindeki pinlerine bağlanmıştır. Sarı kablo ise enkoder ölçümünün alınması için dijital osiloskoba bağlanmıştır.

3.2.1 Motorun Matematiksel Modeli ve Yaklaşım

Bu çalışmada kullanılan DC motor, elektriksel ve mekanik dinamiklerin birlikte modellenmesi sonucunda denklem 3.2'de aşağıdaki transfer fonksiyonu ile ifade edilmiştir:

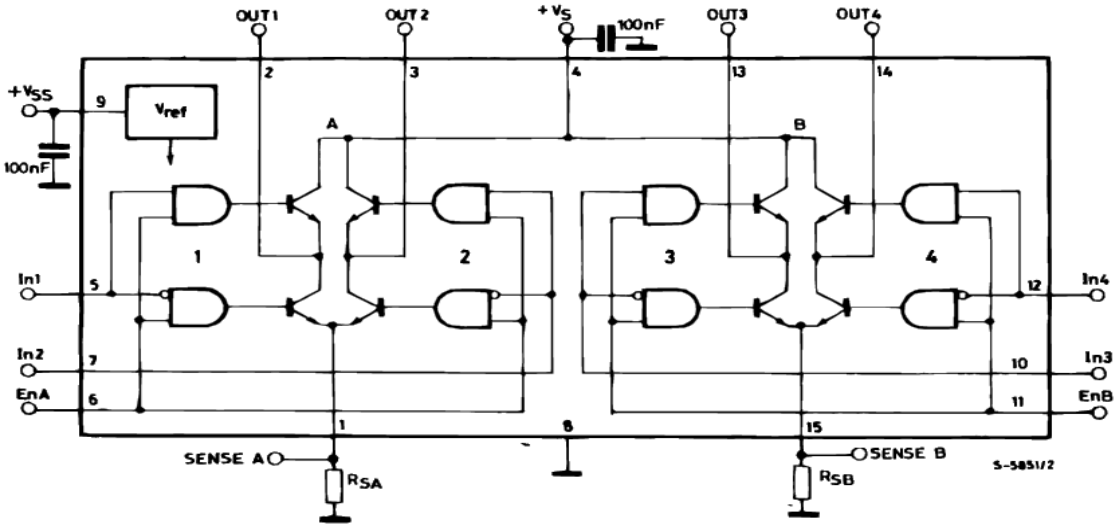
$$G(s) = \frac{(s^2 + 6s + 10)}{(s^3 + 7s + 1)} \tag{3.2}$$

Elde edilen bu model 3. dereceden bir yapıya sahiptir. Literatürde sıkça görülen 1. dereceden yaklaşık modelleme yaklaşımı, hızlı elektriksel dinamiklerin mekanik dinamiklere göre daha küçük zaman sabitine sahip olmasından dolayı kullanılabilmektedir. Ancak bu tezde, motorun tüm dinamik özelliklerinin korunması ve özellikle bozucu etkiler altında sistem davranışının daha doğru yansıtılması amacıyla herhangi bir indirgeme yapılmamış, doğrudan 3. dereceli transfer fonksiyonu kullanılmıştır. PI kontrolör tasarımı ve PSO optimizasyonu da bu tam model üzerinden gerçekleştirilmiş ve deneysel verilerle doğrulanmıştır.

3.3 L298N H Köprüsü- Motor Sürücüsü

Bu tez kapsamında motor kontrolü ve kontrol devrelerinde L298 entegresi kullanılmıştır. L298, 15 bacaklı Multiwatt ve PowerSO-20 paketlerinde sunulan, entegre bir monolitik sürücü devresidir. Yüksek voltaj ve yüksek akım gereksinimlerini karşılayabilen bu entegre, standart TTL mantık seviyeleri ile çalışmak üzere tasarlanmış olup; röle, solenoid, doğru akım (DA) motorları ve kademeli motorlar gibi endüktif yüklerin sürülmesinde kullanılır.

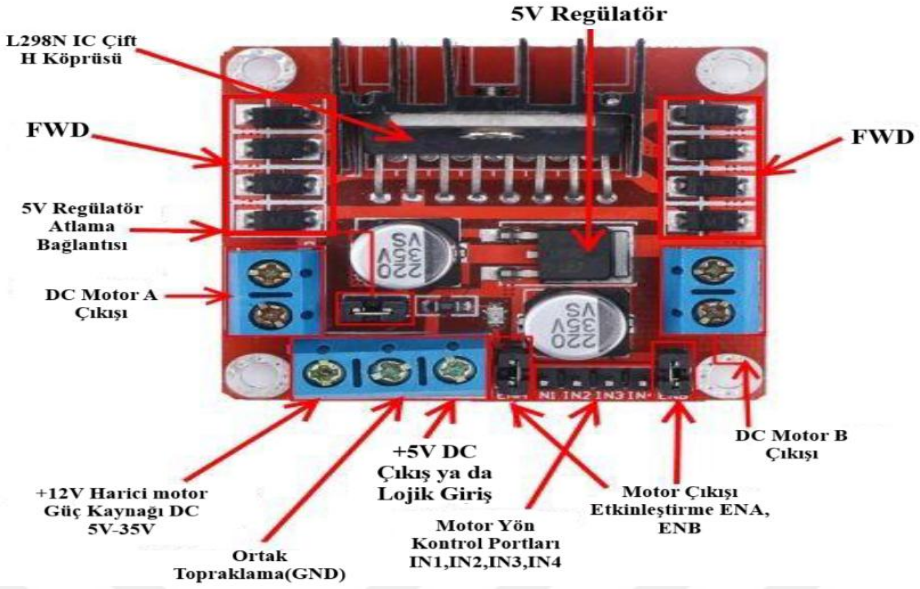
Her bir tam köprü için, cihazın bağımsız olarak çalıştırılmasına veya devre dışı bırakılmasına olanak tanıyan iki adet etkinleştirme (enable) girişi bulunmaktadır. Her köprünün alt transistörlerinin emitör uçları birleştirilmiş olup, bu noktalar harici algılama dirençlerinin bağlanması için uygun terminaller aracılığıyla dışarıya aktarılmıştır. Ayrıca, mantık devrelerinin daha düşük bir gerilimle beslenebilmesini sağlayan ek bir besleme girişi de entegrede yer almaktadır. Şekil 3.4'de, L298 entegresine ait devre şeması sunulmuştur.



Şekil 3.4 L298N Çift H-Köprüsü Devre Şeması(İnt.Kyn.4)

Motorun kontrol edilmesinde PWM kontrolcüsü olarak L298N entegresi ile oluşturulmuş modül kullanılmıştır. L298N entegresi, iki adet doğru akım (DA) motorunun hem yönünü hem de hızını kontrol edebilme imkânı sunan bir H-köprü motor sürücü entegresidir.
Şekil 3.3'te gösterilen bu modül, 5 ila 35 V DA gerilim aralığında çalışan motorlarla uyumludur ve motor başına maksimum 2 A tepe akımı sağlayabilir.

Üzerinde, Motor A ve Motor B çıkışları için iki adet vida tipi terminal bloğu ile motor beslemesi (VCC), toprak (GND) ve 5V regülasyon çıkışı için ayrı bir terminal bloğu bulunmaktadır. Pinlere ait bilgiler Tablo 3.3'de gösterilmiştir. Kullanıdğımız manyetik enkoderli motor +12V çıkışından ve ortak GND ile beslenmiş, motorun enkoder bağlantısı ise +5V çıkışa bağlanmıştır. Aşağıda Şekil 3.5'te L298N modülüne ait görsel ve Çizelge 3.3'de pin tablosu bulunmaktadır.



Şekil 3.5 L298N Çift H-Köprü Kontrol Modülü

Çizelge 3.3 L298 Pin Tablosu

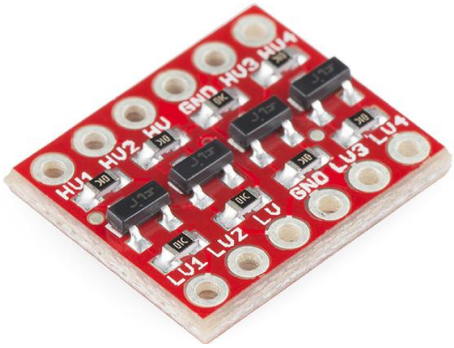
Çıkış 1	Motor A çıkışı
Çıkış 2	Motor A çıkışı
Çıkış 3	Motor B çıkışı
Çıkış 4	Motor B çıkışı
GND	Topraklama
+5V	5V Giriş
ENA	A Motoru için PWM sinyalini etkinleştirir
IN1	A Motorunu Etkinleştirir
IN2	A Motorunu Etkinleştirir
IN3	B Motorunu Etkinleştirir
IN4	B Motorunu Etkinleştirir
ENB	B Motoru için PWM sinyalini etkinleştirir

3.4 Lojik Seviye Dönüştürücü

Bu tezde geliştirilen kontrol sisteminde, farklı lojik gerilim seviyelerinde çalışan bileşenlerin birlikte çalışabilmesini sağlamak amacıyla lojik seviye dönüştürücü (logic level converter) de kullanılmıştır.

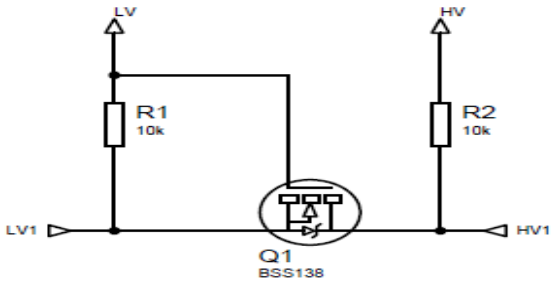
Sistem genelinde hem 5V ile çalışan mikrodenetleyici hem de 3.3V lojik seviyelerinde çalışan sensör modülleri ve motor sürücüleri gibi bileşenler bulunduğundan, doğrudan bağlantı yapılması durumunda donanımsal uyumsuzluk ve hatta kalıcı hasar riski ortaya çıkma durumundan dolayı gerilim çevirimi gerekli görülmüştür. Bu modül, düşük (LV) ve yüksek (HV) gerilim seviyeleri arasında güvenli geçiş sağlayarak, her iki taraftaki bileşenlerin zarar görmeden haberleşmesine olanak tanır.

Ayrıca sistemde kullanılan PWM sinyallerinin bazı bileşenlerle uyumlu iletilebilmesi için lojik seviye çevirimi kritik öneme sahiptir. Bu sayede motor sürücüye aktarılan sinyaller bozulmadan ve güvenli şekilde iletilmiş olur. Bu uygulama, sistemin kararlılığını artırıp uzun vadeli güvenilirliğe katkı sağlamaktadır.
Bu nedenlerle, lojik seviye dönüştürücü modülü, geliştirilen PI kontrollü motor sisteminin hem donanımsal güvenliği hem de sinyal bütünlüğü açısından önemli bir bileşeni olarak tercih edilmiştir. Şekil 3.6'da lojik seviye dönüştürücünün görseli bulunmaktadır.



Şekil 3.6 Lojik Seviye Dönüştürücü(İnt.Kyn.3)

Aşağıda şekil 3.7'de ise lojik seviye dönüştürücünün devre şeması bulunmaktadır;

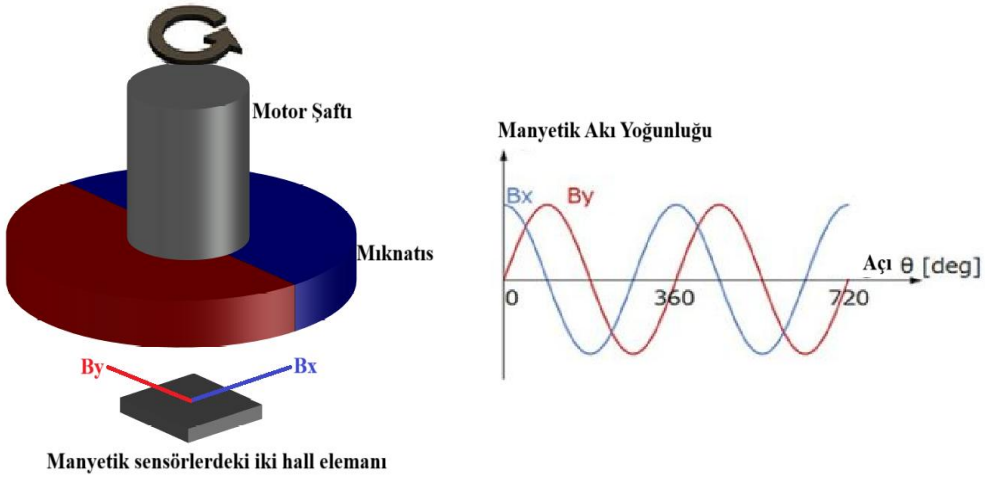


Şekil 3.7 Lojik Seviye Dönüştürücü Devre Şeması

Şekildeki devre şemasında LV1 ve HV1 uçları arasında iki yönlü veri iletimi sağlanır. MOSFET'in kanal ve diyot yapısı sayesinde, LV1 düşük (0V) olduğunda, MOSFET iletken hale gelir ve HV1 hattını da aşağı çeker (LOW yapar). Böylece 5V tarafındaki cihaz da düşük seviyeyi algılar. LV1 yüksek (3.3V) olduğunda, MOSFET kesimde kalır ve HV1, pull-up direnci R2 üzerinden 5V seviyesine çıkar. Aynı şekilde HV1'den sinyal gelmesi durumunda da, MOSFET'in gövde diyodu ve kanal yapısı sayesinde sinyal doğru şekilde LV1 tarafına düşürülerek aktarılır.

3.5 Hız Ölçümünde Manyetik Enkoderin Kullanılması

Manyetik enkoderler, manyetik alan değişimlerini algılayarak konum ve hız bilgisi sağlayan sensör sistemleridir. Optik kodlayıcı, sensör konumlarını tanımlamak için ışık (optik) kullanır. Manyetik enkoderler döner alana yerleştirilmiş olan bir mıknatıs disk ve mıknatısların eksenine merkezlenmiş 1 adet sensörden oluşur. Mıknatıs diskinin kendi ekseni etrafında dönmesi ile birlikte manyetik alanda değişimler olur. Bu değişimler sensör tarafından okunarak dijital sinyallere dönüştürülür ve pozisyon kontrol yazılımının yardımıyla anlamlı veriler olarak işlenir. Mıknatıslanmış kutup çiftlerinin sayısı, algılama elemanlarının sayısı ve elektrik devresinin türü manyetik enkoderin çözünürlüğünü belirler (Beriş ve Taşdelen, 2023). Şekil 3.8'de manyetik enkoderin çalışma prensibi bulunmaktadır;



Şekil 3.8 Manyetik Enkoder Çalışma Prensibi

Manyetizmanın bir sinyal almak için bir eleman olarak kullanılmasının temeli, toz, nem, aşırı sıcaklıklar ve sarsıntılar dahil olmak üzere çok zor çevresel koşullardan etkilenmemesidir. Artımlı enkoderlerde, N, S (Kuzey/Güney) kutupları üzerinde "şeritler" halinde işaretlenir. Bu iki kutup bir adım veya raster-dijital resim verisi-oluşturur. Rasterler çeşitli uzunluklarda olabilir, en yaygın olanları 1 mm; 2 mm; 5 mm'dir. Adımları 20 ve hatta 40 mm olan enkoderlerdir. Manyetik olarak hassas sensörlere sahip bir okuma kafası skala üzerinde hareket eder. Sensörler, tarama içindeki manyetik alandaki değişikliği yakalar ve bir analog sinyal (darbe) oluşturur.

Bu darbe, özel elektronikler tarafından enterpole edilebilir (paylaştırılabilir), böylece sinyal çarpılmış gibi olur. Bu şekilde 1 mm adımlı bir cetvel 0,001 mm (1 μm) veya daha düşük bir çözünürlük (ölçüm adımı) elde edebilir. Çıkış sinyalleri 1 Vpp, TTL, HTL (Push-Pull) Manyetik bant üzerindeki bir mutlak kodlayıcı, kod veya ofset kesikli ölçekler (vernier prensibi) şeklinde bir desene sahiptir. Verilerin çıkış şekli ve çakalın sınırlı uzunluğu bakımından artımlı olanlardan farklıdırlar, ancak herhangi bir zamanda gücü kapattıktan sonra bile konumu belirlerler. Manyetik enkoderler, manyetik alanın doğası gereği optik enkoderlere kıyasla çevresel etkenlere karşı daha dayanıklıdır. Toz, nem, sıcaklık değişimleri ve mekanik darbelere karşı direnci yüksektir. Hız ölçümü aşağıdaki yöntemlerle gerçekleştirilir:

Artımlı (İnkremental) enkoderler: Yukarıda da bahsedildiği gibi, N ve S kutuplarının belirli aralıklarla sıralandığı manyetik diskler kullanılır. Manyetik sensörler, bu kutupların değişimini algılayarak kare dalga sinyaller üretir.
Bu sinyallerin frekansı, dönen elemanın açısal hızı ile doğru orantılıdır. Açısal hız, aşağıdaki formül denklem 3.3'deki gibi hesaplanır:

$$\omega = \frac{2\pi f}{N} \tag{3.3}$$

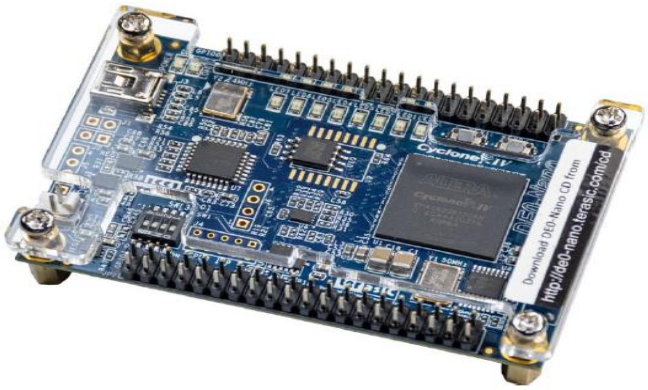
Burada;

- ω: Açısal hız (rad/s),
- f: Sinyal frekansı (Hz),
- N: manyetik kutup çifti sayısıdır.

Mutlak enkoderler: Manyetik bant veya disk üzerinde kodlanmış özel desenler bulunur. Her konumun benzersiz bir dijital kod ile ifade edilmesi sayesinde, sistem kapatılıp tekrar açıldığında bile konum bilgisi korunur. Mutlak enkoderler, hız bilgisini doğrudan türev alarak veya zaman bazlı ölçümlerle hesaplar.

3.6 FPGA Tabanlı PI Kontrol Uygulaması

PI kontrol algoritmasının gömülü donanım üzerinde uygulanması amacıyla Altera Cyclone® IV EP4CE22F17C6N model FPGA geliştirme kartı kullanılmıştır. FPGA üzerinde geliştirilen kontrol yapısı ile DA motorun hız kontrolü gerçekleştirilmiş, farklı PI parametrelerinin sistem üzerindeki etkileri gözlemlenmiştir. Kullanılan FPGA geliştirme kartı Şekil 3.9'da verilmiştir.



Şekil 3.9 Altera Cyclone® IV EP4CE22F17C6N model FPGA geliştirme kartı(İnt.Kyn.5)

Deney düzeneğinde, FPGA ile motor arasındaki haberleşmede lojik seviye uyumsuzluğu göz önünde bulundurularak gerekli önlemler alınmıştır. FPGA kartı 3.3 V lojik seviyesi ile çalıştığından, dış donanımlar (L298 motor sürücü devresi, manyetik enkoder, vb.) doğrudan bağlanmamış; bunun yerine çift yönlü 3.3 V ↔ 5 V lojik seviye dönüştürücü kullanılmıştır. Böylece sinyal bütünlüğü korunmuş ve FPGA'nin zarar görmesi önlenmiştir.

PWM sinyali üretimi: PI kontrol algoritmasının çıkışı FPGA üzerinde sayısal olarak hesaplanmış ve belirlenen 10 kHz frekansında PWM sinyali olarak üretilmiştir. Bu PWM sinyali, L298 motor sürücü kartına iletilmiş ve motorun besleme gerilimi üzerinden hız kontrolü sağlanmıştır.

Deneysel düzende, ikinci bir DA motor bozucu yük olarak kullanılmıştır. Bu yük motoruna uygulanan PWM sinyali FPGA'nin GPIO_09 pininden üretilmiş, yine L298 sürücüsü üzerinden motora iletilmiştir.

Geri besleme: Kullanılan DA motorun manyetik enkoderi 64 CPR çözünürlüğe sahiptir. Dişli oranı ile birlikte mil çıkışında çözünürlük artırılmış ve bu sinyaller FPGA'ye iletilmiştir.

Enkoder çıkışlarının sayımı FPGA içerisinde yapılmış, elde edilen darbe sayısı üzerinden motor hızı hesaplanmıştır. Sinyallerde olası parazitleri azaltmak amacıyla FPGA'de basit sayısal filtreleme (debounce) işlemleri uygulanmıştır.

Akım ölçümü: Motorun armatür akımı ACS712 akım sensörü ile ölçülmüş ve yalnızca gözlem amaçlı olarak dijital osiloskopa iletilmiştir. Kontrol algoritması içinde akım geri beslemesi kullanılmamıştır.

Kullanılan FPGA pinleri aşağıda verilmiştir:

- GPIO_00: PWM çıkışı (kontrol edilen motora iletilen sinyal)
- GPIO_01: Enkoder A kanalı giriş sinyali
- GPIO_03: Enkoder B kanalı giriş sinyali (opsiyonel kullanım)
- GPIO_05: PWM enable veya yön kontrolü sinyali (kullanım ihtiyacına göre)
- GPIO_07: PWM enable veya yön kontrolü sinyali (kullanım ihtiyacına göre)
- GPIO_09: PWM çıkışı (harici yük motorunun kontrolü için)
- Pin 12 (GND): Harici devrelerle ortak toprak hattı bağlantısı

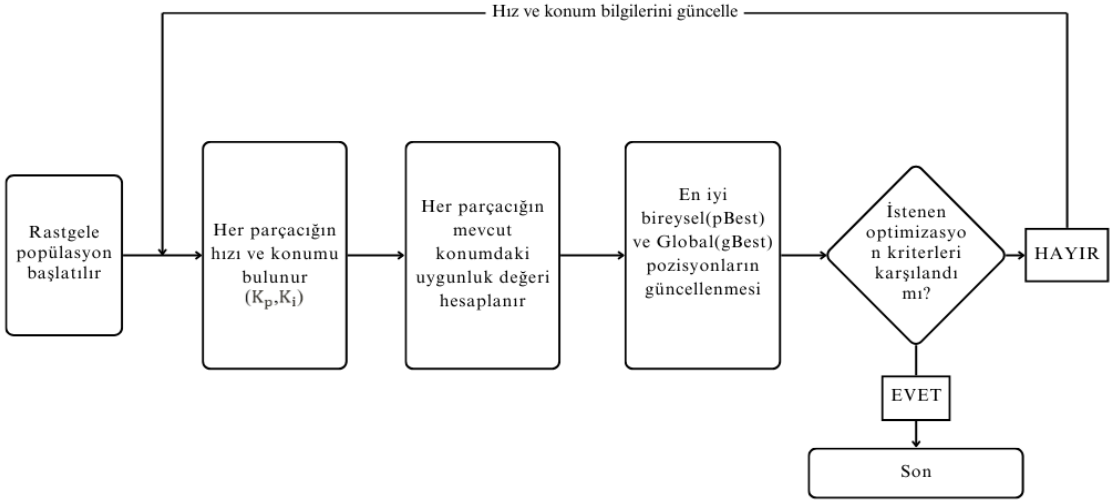
Bu bağlantılar aracılığıyla PI kontrol algoritmasının ürettiği kontrol sinyali motor sürücü devresine iletilmiş, enkoder çıkışlarından elde edilen geri besleme FPGA tarafından okunmuş ve hız hesaplama işlemleri gerçekleştirilmiştir. Deney sırasında farklı K_p ve K_i değerleri FPGA'ye yüklenerek sistemin zamana bağlı tepkileri kaydedilmiş ve analiz edilmiştir.

3.7 Parçacık Sürü Optimizasyonu İle Parametre Tahmini

Bu çalışmada, Parçacık Sürü Optimizasyonu (PSO) algoritması, doğru akım (DA) motorunun adım tepkisinden elde edilen verilerle sistem parametrelerinin tahmin edilmesi amacıyla kullanılmıştır. İlk kez Kennedy ve Eberhart (1995) tarafından önerilen PSO, parçacıkların toplu hareketinden esinlenen stokastik bir arama yöntemidir. Özellikle analitik yöntemlerle çözümü zor veya doğrusal olmayan optimizasyon problemlerinde etkin bir şekilde kullanılmaktadır (Naqvi et al., 2024; Gökçe vd., 2022).

PSO algoritması, çözüm uzayında her biri K_p ve K_i kazançlarını temsil eden parçacıklardan oluşur. Parçacıklar, belirli kurallar altında iteratif olarak hareket ederek en uygun çözümü ararlar. Her bir parçacığın performansı, önceden tanımlanmış bir amaç fonksiyonu üzerinden hesaplanır ve bu değer, parçacığın hız ve konum güncellemesinde kullanılır. Böylece algoritma, her iterasyonda daha uygun çözümlere yaklaşır.

Şekil 3.10'da PSO algoritmasına ait genel akış diyagramı verilmiştir. Diyagramda görüldüğü gibi başlangıçta parçacıklar rastgele dağıtılmakta, ardından uygunluk değerleri hesaplanmakta ve bireysel en iyi (pBest) ile küresel en iyi (gBest) konumlar güncellenmektedir.Süreç, belirlenen kriterler karşılanana veya maksimum iterasyona ulaşılana kadar devam etmektedir.



Şekil 3.10 Parçacık Sürü Optimizasyonu Akış Şeması

PSO algoritması, bu çalışmada PI kontrolörün kazanç değerlerini belirlemek için kullanılmıştır. Amaç, performans kriterlerini (yükselme süresi T_r, aşım M_p, yerleşme süresi T_s ve kararlı durum hatası $\bar{e}_{ss}$ minimize edecek optimum (K_p, K_i) çiftini elde etmektir. Bu kapsamda, uygunluk fonksiyonu denklem 3.4'de verilmiştir:

$$J = \alpha \cdot tr + \beta \cdot OS + \gamma \cdot t \tag{3.4}$$

Burada α,β,γ, katsayıları, kriterlerin ağırlıklarını temsil etmektedir. PSO algoritması, bu fonksiyonu minimize eden en uygun PI kazançlarını belirlemiştir. Elde edilen optimum kazanç kombinasyonu, manuel olarak test edilen dokuz farklı kazanç setiyle karşılaştırılmış ve PSO'nun daha düşük aşım, kısa yerleşme süresi ve kararlı takip yeteneği sunduğu görülmüştür. Grafiksel incelemelerde, PSO ile bulunan optimum noktanın uygunluk yüzeyinde minimum bölgelere karşılık geldiği tespit edilmiştir.

3.7.1 PSO'nun K_p-K_i Alanında Uygunluk Dağılımı ve Parçacık Hareketi

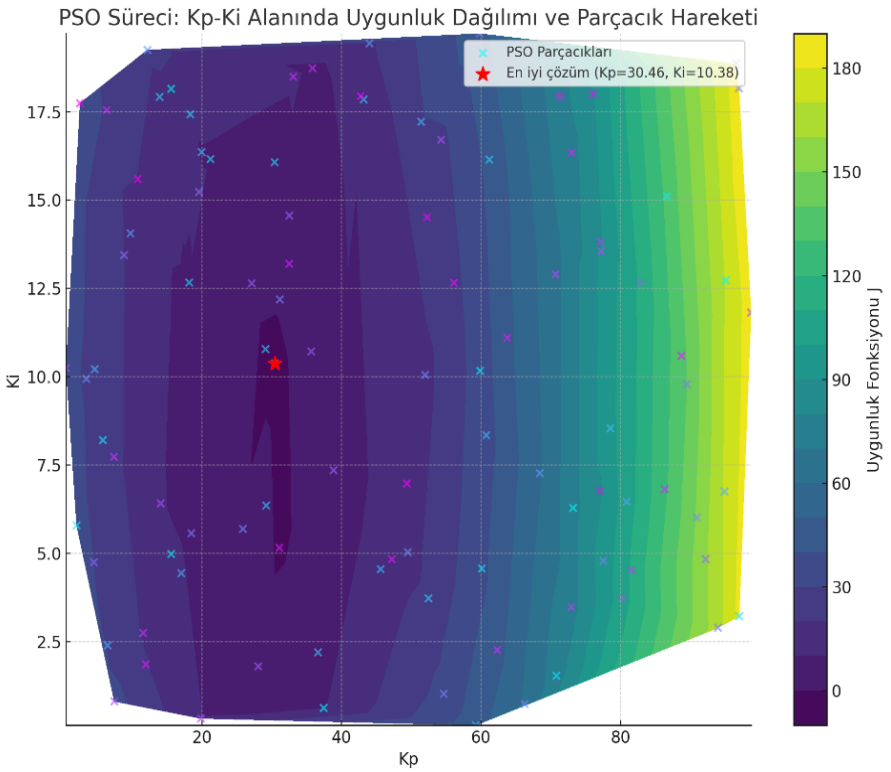
PSO sürecini görsel olarak açıklamak amacıyla üç farklı grafik kullanılmıştır:

1. $K_p - K_i$ alanında uygunluk fonksiyonu dağılımı ve parçacıkların konumları,
2. Parçacıkların iterasyonlar boyunca hareketleri,
3. Optimum noktanın (global best) işaretlendiği parçacık hareket diyagramı.

Bu grafikler, PSO'nun araştırma (exploration) ve sömürü (exploitation) fazlarını ortaya koymakta ve algoritmanın yakınsama sürecini görsel olarak desteklemektedir. Ölçümler sonucunda elde edilen optimum kazanç seti, tüm duty oranlarında geçerli olmuş ve akım sınırı ($I_{tepe} \leq 0.5 \cdot I_{stall}$) da ihlal edilmemiştir. Bu durum, PSO ile bulunan çözümün hem performans hem de güvenlik kriterleri bakımından güvenilir olduğunu göstermektedir.

Şekil 3.11'de PSO'nun arama süreci görselleştirilmiştir ve buna dair uygunluk dağılım ve parçacık hareketi grafiği bulunmaktadır. Grafikte yer alan kontur haritaları, uygunluk fonksiyonu J'nin dağılımını göstermektedir. Mavi bölgeler minimum, sarı bölgeler maksimum değerlere işaret etmektedir. Noktalar parçacıkların deneme yaptığı kazanç çiftlerini, renk geçişleri ise iterasyon sırasını göstermektedir.

Kırmızı yıldız, PSO'nun bulduğu optimum çözümü yani J değerini minium yapan $K_p - K_i$ çiftini işaret etmektedir.



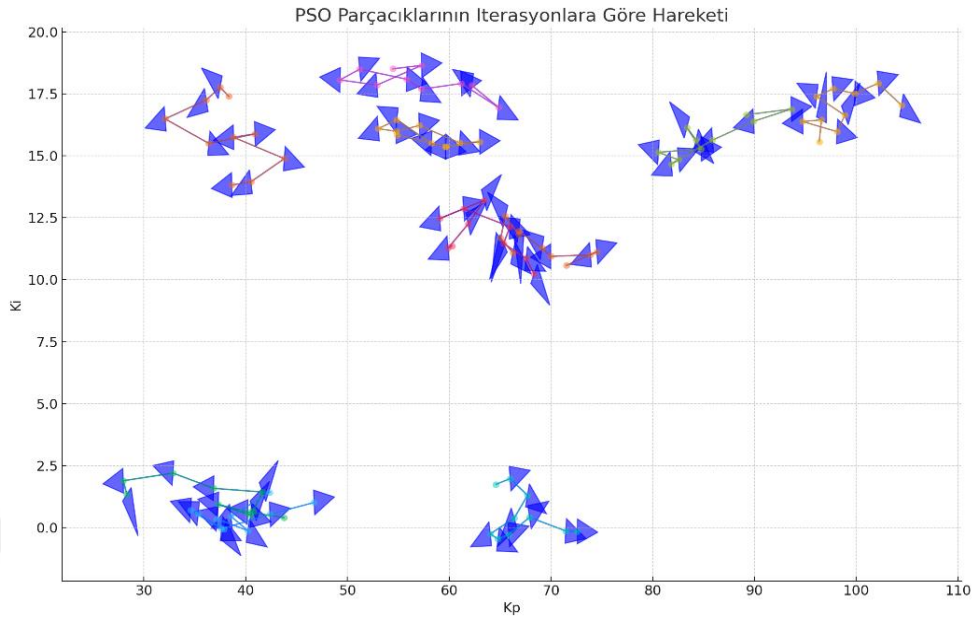
Şekil 3. 11 $K_p - K_i$ Alanında PSO Uygunluk Dağılımı ve Parçacık Hareketlerinin Grafiği

PSO algoritması ile belirlenen optimum kazanç değerleri, yalnızca teorik model üzerinde değil, aynı zamanda deneysel koşullarda da doğrulanmıştır. Bu doğrulama sürecinde motorun adım cevabı ve sinüs sinyali altındaki çalışmaları incelenmiştir.

Deneysel sonuçlar, PSO algoritmasının önerdiği optimum kazanç değerleriyle uyumlu olarak yaklaşık $K_p \approx 32$ ve $K_i \approx 0.001$ şeklinde elde edilmiştir. Bu değerler, gerek adım cevabı gerekse sinüs sinyali altında sistemin kararlı hâlde en iyi performansı sergilemesini sağlamıştır.

Böylece PSO'nun teorik arama sürecinde belirlenen optimum kazanç noktası, deneysel testlerle desteklenmiş ve yöntem geçerliliği güçlendirilmiştir.

Aşağıdaki Şekil 3.12'de ise PSO parçacıklarının iterasyonlara göre hareketi bulunmaktadır:



Şekil 3. 12 PSO Parçacıklarının İterasyonlara Göre Hareketlerinin Grafiği

Her bir ok, bir parçacığın bir iterasyondan sonraki iterasyona geçişini göstermektedir. Mavi oklar yönü ve büyüklüğü ile hareketin yönünü ve şiddetini temsil etmektedir. Daire ve çizgiler, parçacığın önceki adımlarını takip eder; böylece parçacığın yolunu izlemek mümkün olmaktadır.

3.7.2 Parçacık Sürü Optimizasyonunun Algoritmik Süreci

PSO algoritmasının işleyişi yalnızca görsel verilerle değil, aynı zamanda algoritmik adımlar üzerinden de tanımlanabilir:

1. Başlatma (Initialization): Çözüm uzayında rastgele dağıtılmış parçacıklar oluşturulur. Her parçacık, bir (K_p, K_i) kazanç çiftini temsil eder.
2. Amaç Fonksiyonunun Hesaplanması (Fitness Evaluation): Her parçacığın uygunluk değeri Denklem (3.3) kullanılarak hesaplanır.
3. pBest ve gBest Güncellemesi: Parçacık kendi en iyi geçmiş çözümünü (pBest) saklar, tüm parçacıklar arasındaki en iyi çözüm ise global en iyi (gBest) olarak güncellenir.
4. Hız ve Konum Güncellemesi: Hız ve konum güncellemesi için kullanılan formül denklem 3.5'teki gibidir;

$$v(t+1) = \omega \cdot v(t) + c_1 \cdot r_1 \cdot \left(pBest - x(t)\right) + c_2 \cdot r_2 \cdot \left(gBest - x(t)\right) ve\ x(t+1) = x(t) + v(t+1). \quad (3.5)$$

Burada ω ataletsel ağırlık, c_1 ve c_2 öğrenme katsayıları, r_1 ve r_2 rastgele katsayılardır.

5. Durma Kriteri (Termination): Maksimum iterasyon sayısına ulaşıldığında veya uygunluk fonksiyonundaki iyileşme belirli bir eşiğin altına düştüğünde algoritma durdurulur. Son gBest değeri, optimum PI kazanç çifti (K_{p^*}, K_{i^*}) olarak kabul edilir.

Bu süreç, ilk iterasyonlarda geniş bir arama yaparak keşif (exploration) sağlamakta, son iterasyonlarda ise optimum nokta çevresinde yoğunlaşarak sömürü (exploitation) aşamasını gerçekleştirmektedir. Şekil 3.13'te PSO algoritmasının akış diyagramı verilmiştir.

Şekil 3.13 PSO Algoritmasının Akış Diyagramına Ait Görsel

3.8 Performans Metrikleri

PID kontrolör tasarımında her parametre sistem çıkışını etkiler. Yükselme zamanı, kararlı durum hatası, aşım gibi parametrelerin dikkate alınması, en verimli ve optimum sistem çıktısını elde etmek için gereklidir.

Verimli ve optimum sistem çıktısını elde etmek için yükselme zamanı parametresi ayarlanmalıdır. Yükselme süresi parametresini ayarlamak için Oransal (P) kontrolörün kazancı değiştirilmelidir, ayrıntılar Çizelge 3.4'de verilmiştir. Diğer parametreler yükselme süresini önemli ölçüde etkilemez. Aşma parametresi oransal (P) ve integral (I) katsayılarını artırır ve türev (D) katsayısını azaltır. Yerleşme süresi, dalgalanmaların azaltılması ve istenen değerde sabit hale getirilmesi için geçen süreyi ifade eder. Oransal katsayı küçük bir yükseltici etkiye sahipken, integral katsayısı artışı ve türev katsayısı düşüşü bulunmuştur. Sinyalin faz oranı türevine göre, faz oranından türetilen özet, tüm sistem bilgilerini elde etmek için kullanılır. Çizelge 3.4'te PID kontrolör parametrelerinin sistem cevabına etkisi verilmiştir.

Çizelge 3.4 PID kontrolör parametrelerinin sistem cevabına etkisi.

Kontrolör	Katsayı	Yükselme Zamanı	Aşma	Yerleşme Zamanı	Kalıcı Durum Süresi
P (Oransal)	K_p	Azalır	Artar	Az Artar	Azalır
I (Integral)	K_i	Az azalır	Artar	Artar	Azalır
D (Türevsel)	K_d	Az değişir	Azalır	Azalır	Etkisi yok

Çizelge 3.4 incelendiğinde PID kontrol parametrelerinin sistem tepkisine olan etkisi şu şekilde değerlendirilebilir:

Oransal Kazanç (K_p): Oransal kazancın artırılması, sistemin yükselme zamanını azaltmakta, yani referans değere daha hızlı ulaşmasını sağlamaktadır. Ancak bu artış aynı zamanda sistemin aşma (overshoot) eğilimini artırmakta ve yerleşme süresinde (settling time) hafif bir artışa sebep olabilmektedir.

Buna karşılık kalıcı durum hatasında azalma gözlemlenir. Bu nedenle K_p parametresi, sistemin daha hızlı yanıt vermesi için gerekli olmakla birlikte, dengeli bir şekilde ayarlanmalıdır.

İntegral Kazanç (K_i) : Bu değerin artırılması, sistemin kalıcı durum hatasını azaltarak, uzun vadede referans değere ulaşmasını sağlar. Ancak bu parametrenin artışı sistemde aşma miktarını artırabilir ve yerleşme süresini uzatabilir. Bu durum, özellikle sistemin kararlılığı açısından dikkat edilmesi gereken bir durumdur.

Türevsel Kazanç (K_d): Türevsel kazanç, sistemin dinamik davranışını yumuşatarak aşma miktarını ve yerleşme süresini azaltıcı bir etki göstermektedir. Ancak, yükselme zamanı ve kalıcı durum hatası üzerinde doğrudan belirgin bir etkisi bulunmamaktadır. Bu çalışmada türevsel kazanç kullanılmadığı için ($K_d = 0$), bu etki göz ardı edilmiştir.

Yine kontrol sistemlerinde performans metriklerinden birisi olan kararlı durum hatası (e_{ss}) sistem çıkışı ile referans sinyal arasındaki farkın geçici rejim etkileri kaybolduktan sonraki değerini ifade eder. Klasik tanım, sistemin dengeye ulaşmasından sonra tek bir hata değeri üzerinden yapılır. Denklem 3.6'de kalıcı durum hatası denklemi bulunmaktadır:

$$e_{ss} = \lim_{t \to \infty} \big(r(t) - y(t)\big) \tag{3.6}$$

Ancak deneysel çalışmalarda elde edilen hız verileri çoğunlukla gürültü ve küçük dalgalanmalar içerdiğinden, tek bir anlık değer yerine ortalama kararlı durum hatası ($\bar{e}_{ss}$) kullanılması daha güvenilir sonuç vermektedir. Bu çalışmada hata hesabı, kararlı durum bölgesinde ölçülen ortalama hızın referans hız değerinden çıkarılması ile yapılmıştır. Denklem 3.7'de ortalama kararlı durum hatası denklemi bulunmaktadır:

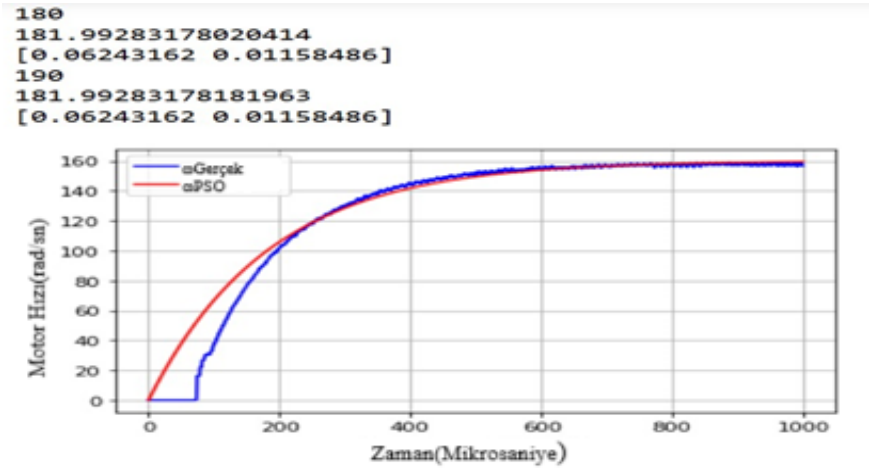
$$\bar{e}_{ss} = \bar{\omega}_{ss} - \omega_{ref} \tag{3.7}$$

Bu yaklaşım, ölçümlerdeki rastlantısal salınımların etkisini azaltarak PI kontrol parametrelerinin performansını daha doğru şekilde karşılaştırmaya imkân tanımaktadır.

4. BULGULAR

4.1 Adım Tepkisi Ve Sinüs Sinyali Altında Sistem Tepkisi

Bu bölümde, çalışmada elde edilen deneysel veriler ve bu veriler üzerinden gerçekleştirilen analizler sunulmaktadır. İlk olarak, DA motor sistemine adım (step) sinyali uygulanarak açık çevrimde motorun dinamik tepkisi incelenmiştir. Bu deneyde motorun açısal hız verileri, dijital osiloskop aracılığıyla kaydedilmiş ve elde edilen veriler, PSO (Parçacık Sürü Optimizasyonu) algoritması kullanılarak oluşturulan matematiksel modelle karşılaştırılmıştır. PSO algoritması yardımıyla sistemin transfer fonksiyonuna ait kazanç ve zaman sabiti gibi parametreler başarıyla tahmin edilmiştir. PSO algoritması ile optimize edilen sistem parametreleri kullanılarak PI denetleyicinin kazançları (K_p ve K_i) hesaplanmıştır. İlk analizler sonucunda elde edilen PI kontrol parametreleri $K_p = 32$ ve $K_i = 0.001$ olarak belirlenmiştir. Bu deneyin tamamlanmasıyla birlikte, açık çevrim ve kapalı çevrim sistem tepkilerinin karşılaştırılması ve kontrol performansının değerlendirilmesi mümkün olmuştur. İkinci aşamada ise kapalı çevrimde PI kontrolör kullanılarak motor hız kontrolü gerçekleştirilmiştir. Aşağıdaki Şekil 4.1'de, mavi ile gösterilen gerçek sistem tepkisi (ωGerçek) ve kırmızı ile gösterilen PSO algoritmasıyla tahmin edilen model çıktısı (ωPSO) karşılaştırılmıştır:



Şekil 4.1 DA motorun adım cevabı: Gerçek veri ile PSO model karşılaştırılması

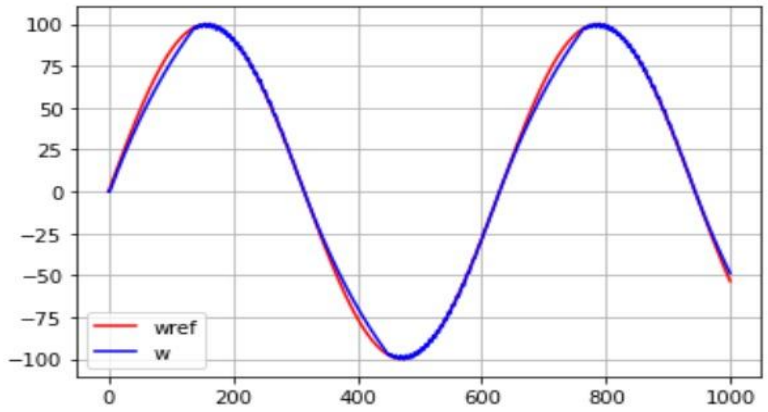
Şekil 4.1'de, DA motor sistemine uygulanan adım girişine karşılık olarak elde edilen gerçek açısal hız tepkisi (mavi çizgi, *ωActual*) ve Parçacık Sürü Optimizasyonu (PSO) algoritması kullanılarak elde edilen matematiksel modelin çıktısı (kırmızı çizgi, *ωPSO*) karşılaştırmalı olarak sunulmuştur. Grafik incelendiğinde, PSO ile tahmin edilen modelin, motorun dinamik davranışını oldukça başarılı bir şekilde temsil ettiği gözlemlenmektedir.

Özellikle ilk yükselme süresi (rise time), aşım miktarı (overshoot), durulma zamanı (settling time) ve kararlı durum değeri (steady-state value) açısından model ve gerçek sistem çıktısı arasında yüksek düzeyde bir benzerlik olduğu görülmektedir.

PSO algoritması ile elde edilen modelin gerçek sisteme bu kadar yakın bir tepki vermesi, parametre tahmininde doğruluk düzeyinin yüksek olduğunu göstermektedir. İlk 200 mikro saniyelik sürede modelin gerçek sistemden hafif şekilde daha hızlı yükseldiği dikkat çekmektedir. Bu küçük fark, PSO algoritmasının amaç fonksiyonunda belirli ağırlıkların daha öncelikli olarak değerlendirilmesinden kaynaklanabilir. Buna rağmen modelin, genel tepki karakteristiğini neredeyse birebir yakaladığı söylenebilir. Ayrıca, her iki grafiğin üzerinde yer alan ve PSO algoritmasının çalışması sırasında elde edilen iterasyon sonuçları da modelin başarısını ortaya koymaktadır. Örneğin, Şekil 4.1'de gözlemlenen $\omega_{Actual} \setminus \omega_{PSO}$ karşılaştırmasına ait en iyi uygunluk (fitness) değerlerinden biri olan yaklaşık 181.99, algoritmanın çözüm uzayında yüksek doğrulukta bir parametre seti bulduğunu göstermektedir. Bu değerin yanında yer alan [0.06243, 0.01158] gibi parametreler ise modelin en uygun performansı verdiği K_p ve K_i değerlerine işaret etmektedir. PSO algoritması, çok sayıda iterasyon (örneğin 180. ve 190. iterasyonlarda aynı optimal değerin korunması) boyunca çözümde istikrar göstermiştir. Bu da küresel minimuma ulaşmada başarılı bir şekilde yakınsama sağlandığını göstermektedir.

Bu başarılı örtüşme, modelin ileri düzey kontrolör tasarımı çalışmalarında kullanılabilecek düzeyde güvenilir olduğunu ve sistem dinamiklerinin belirlenmesinde PSO algoritmasının etkili bir yöntem olduğunu göstermektedir. Böylece, sistemin matematiksel temsili ile yapılan kontrolcü tasarımları, sahada uygulanabilir ve etkili sonuçlar verebilecektir. Aşağıda Şekil 4.2'de ise Periyodik referans altında gerçek sistem ve modelin tepkisi bulunmaktadır;

```
448.02187571598796
[3.12741533e+01 1.00000000e-03 0.00000000e+00]
90
448.02187571598796
[3.16134529e+01 1.00000000e-03 0.00000000e+00]
444.9220343824268
```



Şekil 4.2 Periyodik Referans Altında Gerçek Sistem ve Model Tepkisi

Şekil 4.2'de, sistemin periyodik yapıdaki bir referans sinyali altındaki davranışı incelenmiştir. Bu grafikte, referans açısal hız sinyali (ω_{ref}, kırmızı çizgi) ile sistemin gerçek çıkışı (ω, mavi çizgi) karşılaştırmalı olarak gösterilmiştir. Referans sinyali tam anlamıyla bir sinüzoidal giriş olmamakla birlikte, dalga formu bakımından sinüs dalgasına benzer periyodik bir yapı sergilemektedir. Şekilde yer alan periyodik referans altında yapılan analizde 90. iterasyonda ulaşılan uygunluk değeri olan 448.02, sistemin değişken giriş sinyallerini takip etmede yeterli düzeyde optimizasyon sağlandığını ortaya koymaktadır. Bu noktada da algoritmanın bulduğu parametreler (örneğin $K_p \approx 31.27$) sistemin bu karmaşık referans altında dahi makul bir izleme başarısı göstermesini sağlamıştır.

İterasyon boyunca uygunluk değerlerinde anlamlı bir iyileşme olduğu ve çözümün durağanlaştığı göz önünde bulundurulduğunda, PSO algoritmasının bu yapay sinüzoidal referans altında da kararlı ve güvenilir bir modelleme sunduğu sonucuna varılabilir. Bu iterasyon sonuçları, yalnızca grafiksel benzerliklerle sınırlı kalmayıp sayısal doğruluk açısından da modelin sistem davranışını başarılı bir şekilde temsil ettiğini göstermektedir.

Sonuç olarak, hem sabit adım girişleri hem de değişken referans sinyalleri için PSO algoritması ile oluşturulan model, gerçek sistem tepkilerini yüksek hassasiyetle takip edebilecek düzeydedir.

Grafiksel olarak incelendiğinde de, sistemin bu değişken referans sinyalini genel hatlarıyla başarılı bir şekilde izleyebildiği görülmektedir. Belirli anlarda, özellikle geçiş noktalarında, küçük genlik sapmaları ve faz gecikmeleri gözlemlenmekle birlikte, modelin genel olarak referans sinyale olan uyumu yeterli düzeydedir. Bu sapmalar, sistemin dinamik yanıt süresi, içsel gecikmeler ve sürtünme gibi fiziksel etkilerden kaynaklanabilir. Modelin bu tür değişken referanslara verdiği tepki, kontrol performansının zamana bağlı olarak nasıl değiştiğini anlamak açısından önemlidir. Bu nedenle, Şekil 4.2'de elde edilen sonuçlar, modelin sadece sabit değil, aynı zamanda zamanla değişen referans sinyaller altında da geçerli olduğunu göstermektedir. Bu durum, daha gelişmiş kontrolcü tasarımları için de yol gösterici olabilir.

4.2 PI Kontrol Deneyi ve Enkoder Verilerinin Analizi

Yapılan ikinci deney olan PI kontrol deneyinde PI kontrolörün performansı, referans hız sinyaline ne kadar yakın çalışıldığıyla değerlendirildiği için motor enkoderinden elde edilen hız verileri analiz edilmiştir. Ölçümler, dijital osiloskop ile 1 MHz örnekleme frekansında(1 MHz =1 örnek = 1 μs) kaydedilmiş ve her bir dosya 1 saniyelik kırpılmış verilerden ($t_{sec} = 106/t_{\mu s}$) oluşturularak gereksiz satırlar atılmıştır.

Ölçüm verilerinde başlangıç anı ve durma anı gibi kontrol dışı geçici etkiler bulunduğundan Pyhton kullanılarak kırpılma yapılmıştır. Böylece, yalnızca sistemin kararlı duruma ulaştığı bölge analiz kapsamına alınmıştır. Enkoder çözünürlüğü 64 CPR olup, ölçümlerde pozitif kenar sayımı yöntemi kullanılmıştır. Pozitif kenarların zaman farkı(Δt) bulunmuştur.

Bir pozitif kenar bir pulse'a karşılık geldiği için, elde edilen sinyalden frekans hesaplanarak ($f_{pulse} = 1/\Delta t$), RPM=RPS×60 yapılarak devir/saniye (RPS) , ardından RPM ve rad/s (ωrad/s=RPM×2π/60) değerleri elde edilmiş, son aşamada hız, normalize edilerek referans hız değeri olan 0.90'a oranlanmıştır.

Deney gerçekleştirilirken literatürde duty oranı olarak isimlendirilen yani doluluk oranı değiştirilerek, farklı PWM voltajlarını değiştirme yöntemi ile bozucu(disturbance) yük motoru açık çevrimle kontrol edilmiştir.

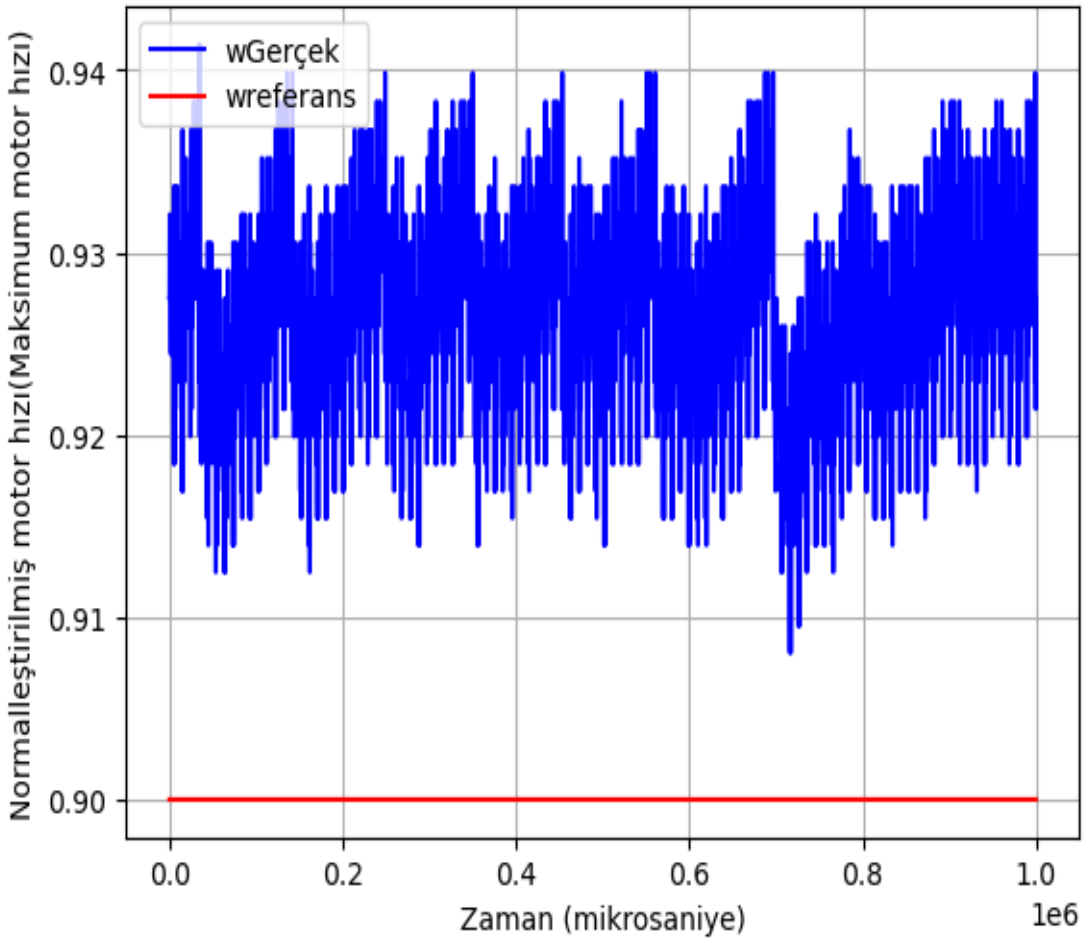
Ana motor olan kapalı çevrimle çalışan manyetik enkoderli motorun bilmediği voltaj seviyelerinde yük motoru çalıştırılmıştır. Bu yaklaşım, gerçek sistemlerdeki öngörülemeyen tork değişimlerini simüle etmiş ve kontrolcünün sağlamlığını(robust) test etmiştir.

Doluluk oranı ise FPGA üzerinden Quartus programı ile T_{on} değeri yani PWM sinyalinin yüksek (3.3V) kaldığı süre değiştirilerek kontrol edilmiştir. Enkoder verilerinde zaman eksenleri mikrosaniye cinsinden olduğu için T değeri ise 5000 alınmıştır. T_{on} değeri de bu değere göre oranlanmıştır. Bu değer %0 (T_{on} =0), %10 ($T_{on} = 500$), %20 ($T_{on} = 1000$), %30 ($T_{on} = 1500$), %40 ($T_{on} = 2000$) ve %50 ($T_{on} = 2500$) yapılarak kontrol uygulanmıştır. Her doluluk oranında en düşük J değerini veren K_p– K_i çifti optimum olarak belirlenmiştir. Kullanılan K_p ve K_i değerleri Çizelge 4.1'de verilmiştir;

Çizelge 4.1 DC Motor Hız Kontrolü İçin $\mathbf{K_p}$ **ve** $\mathbf{K_i}$ değerleri

K_p	K_i
1	0
1	1
1	10
30	0
30	1
30	10
100	0
100	1
100	10

PI kontrolcüde oransal kazanç K_p ve integral kazanç K_i değerleri de yine doluluk oranı ile beraber değiştirilmiş ve elde edilen sistem tepkileri incelenmiştir. Sistem tepkileri motor enkoder çıkışının dijital osiloskoba bağlanması ile ölçülmüş ve alınan grafikler python koduna eklenmiş ve çıkan grafikler ile her bir durumda motorun gerçek hız tepkisi $\omega_{Gerçek}$ ile referans hız $\omega_{referans}$ karşılaştırılmıştır. İlk olarak %0 doluluk oranında öne çıkan 1. parametre çifti $K_p = 100, K_i = 10$ için oluşan grafik Şekil 4.3'deki gibidir;

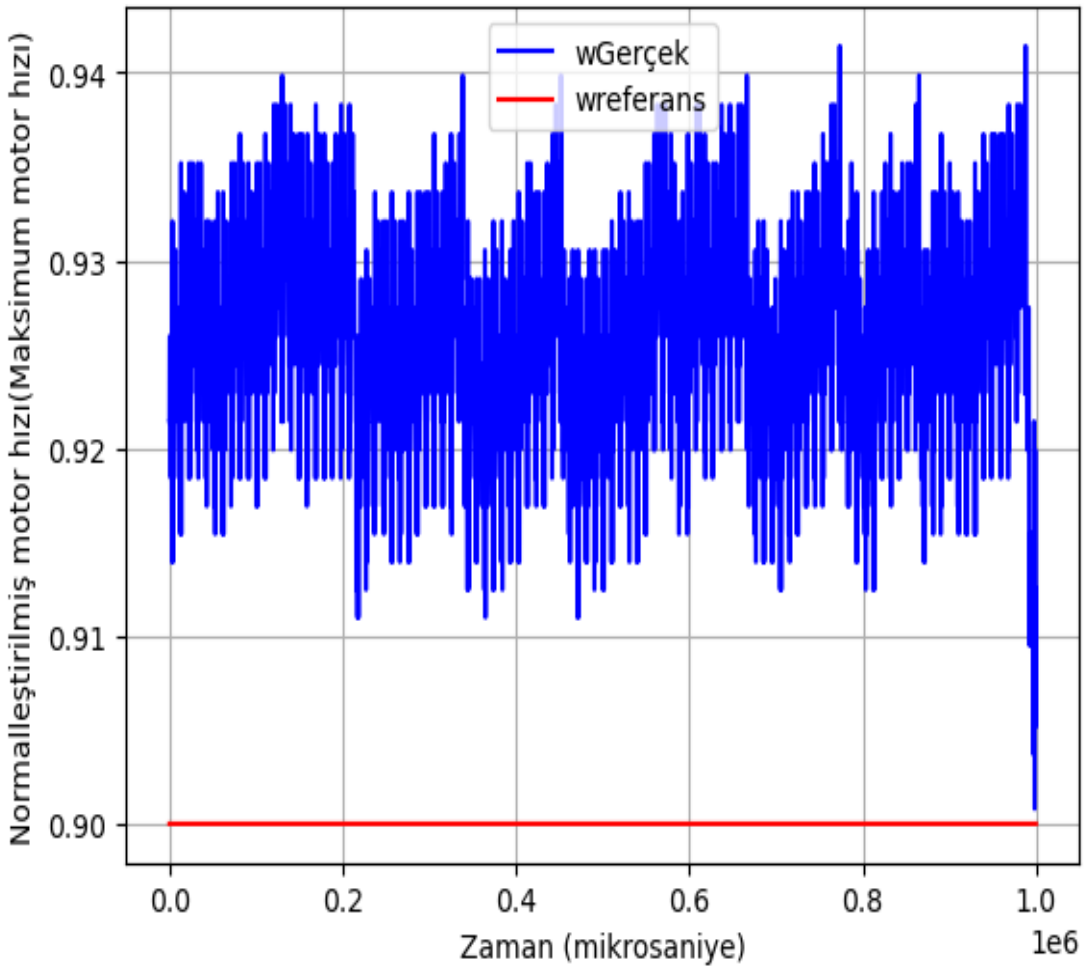


Şekil 4.3 %0 doluluk oranında $K_p = 100, K_i = 10$ için oluşan grafikler

$\bar{e}_{ss}$ (ortalama kararlı durum hatası) değeri bu set içinde en düşük değerde ölçülmüştür. Ancak çok yüksek aşım(M_P) ve banda giremeyen yerleşme süresi(T_s), kontrolün aşırı agresif olduğunu göstermiştir.

PI'nin I bileşeni kararlı hatayı bastırırken, P'nin yüksek olması tepe değerleri büyütmüştür. Bu, standart zaman-cevap metriklerinin birlikte değerlendirilmesi gerektiğini vurgulayan temel yaklaşımla uyumlu olduğunu göstermiştir. Ortalama kararlı durum hızı (0.4 s sonrası):1.9144, ortalama kararlı durum hatası $\bar{e}_{ss} = \bar{\omega} - 0.90$: +1.0144, yükselme süresi(T_r): 0, $aşım(M_P)$: %316,48 çıkmıştır.

Yerleşme süresi (T_s) ise tanımlanamamıştır. %0 doluluk için ikinci öne çıkan parametre çifti $K_p = 30, K_i = 10$ için alınan ölçüm sonuçları Şekil 4.4'teki gibidir;

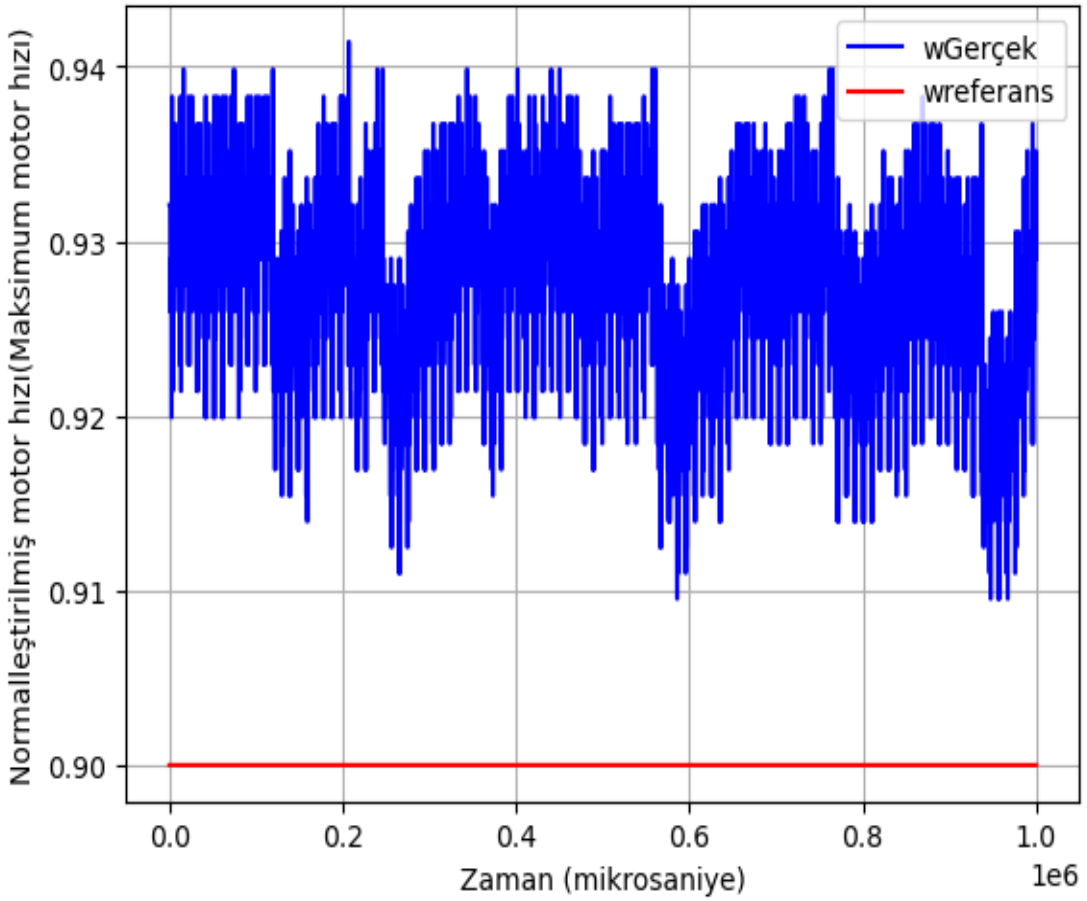


Şekil 4.4 %0 doluluk oranında $K_p = 30, K_i = 10$ için oluşan grafik

$\bar{e}_{ss}$ (ortalama kararlı durum hatası) değeri, $K_p = 100$ $K_i = 10$ setiyle neredeyse aynı ölçülmüştür. Ancak pratikte $K_p = 30$ daha ılımlı bir P etkisi sağlamaktadır. Bu, çok kriterli seçimde (J) genellikle $K_p = 30, K_i = 10$ gibi orta-yüksek I ile desteklenen orta P kazançlarının öne çıkmasının nedenidir.

Çok amaçlı (T_r , M_p, T_s , $\bar{e}_{ss}$) değerlendirme literatürde de önerilmektedir. Ortalama kararlı durum hızı: 1.9153, ortalama kararlı durum hatası $\bar{e}_{ss} = +1.0153$, yükselme süresi (T_r): 0.000 s, aşım (M_P): %316.48 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır.

%0 doluluk oranı için üçüncü öne çıkan parametre çifti olan $K_p = 30, K_i = 1$ için alınan ölçüm sonuçları Şekil 4.5'teki gibidir;



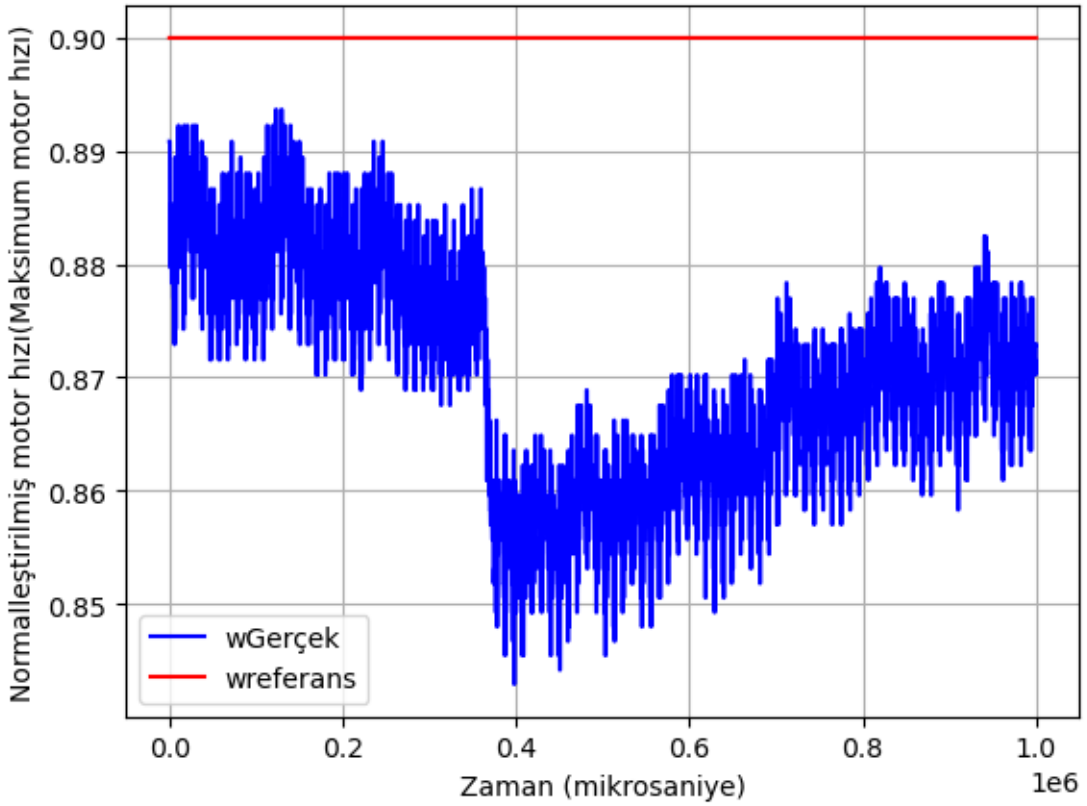
Şekil 4.5 %0 doluluk oranında $K_p = 30, K_i = 1$ için oluşan grafik

Ortalama kararlı durum hızı: 1.9257, ortalama kararlı durum hatası: $\bar{e}_{ss} = +1.0257$, yükselme süresi (T_r): 0.000s, aşım (M_p): %316.48 çıkmıştır. Yerleşme süresi (T_s) ise tanımlanamamıştır.

$\bar{e}_{ss}$, yukarıdaki iki adaya göre biraz daha yüksek buna karşılık I etkisi sınırlı olduğundan aşım ve dalgalanmalar pratikte nispeten daha kontrollü görünmektedir(aynı nominal M_P ölçülse de dalga biçimi daha sakindir).

PI'de integralin artırılmasının kalıcı hatayı azalttığı, ancak geçici rejimde (tepe değerler, sönüm) dikkat gerektirdiği bilinir.

%0 doluluk oranı için son öne çıkan parametre çifti $K_p = 1, K_i = 10$ için alınan ölçüm sonuçları Şekil 4.6'daki gibidir;



Şekil 4.6 %0 doluluk oranında $K_p = 1$, $K_i = 10$ için oluşan grafik

Ortalama kararlı durum hızı: 2.1050, ortalama kararlı durum hatası: $\bar{e}_{ss} = +1.2050$ en yüksek, yükselme süresi (T_r): 0.000 s, aşım (M_p): %316.48 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır. P kazancının çok düşük olması, integratörün topladığı hatayı etkin şekilde aktüatöre yansıtamamasına neden olmuştur. Bunun sonucunda hem kararlı durum seviyesi çok yüksek kalmış hem de dalgalı bir hız bandı oluşmuştur. Bu davranış, düşük P / yüksek I ayarlarının DC motorlarda windup ve uzayan sönüm eğilimi yaratabileceğine dair literatürdeki bulgularla tutarlıdır. %0 doluluk oranı için alınan sonuçlar değerlendirildiğinde:

En düşük $\bar{e}_{ss}$: $K_p = 100$, $K_i = 10$ (≈ +1.0144)

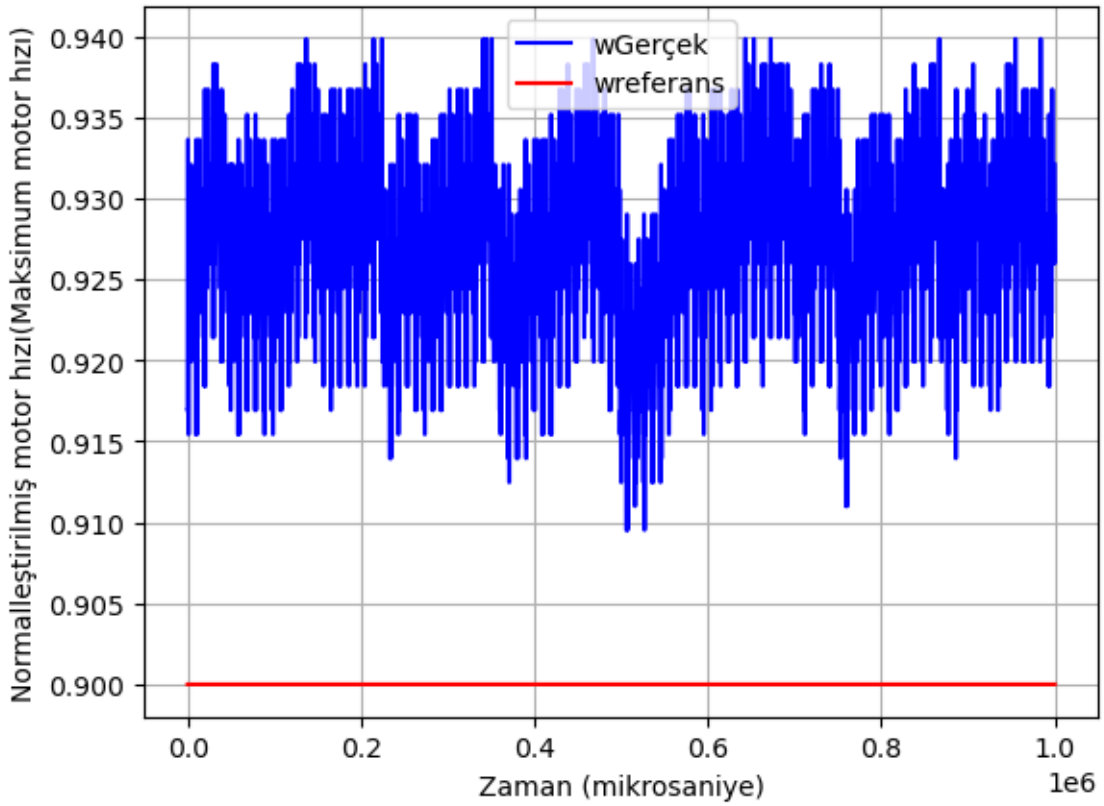
Dengeli aday: K_p=30,$K_i = 10$ (≈ +1.0153)

Pratik kompromi: K_p=30, $K_i = 1$ (≈ +1.0257)

En zayıf: K_p=1, $K_i = 10$ (≈ +1.2050).

Yukarıdaki sıralama yalnızca $\bar{e}_{ss}$ açısından yapılmıştır. Çok kriterli performans göstergesi (J) içerisinde M_P ve T_s de hesaba katıldığında, literatürdeki önerilerle uyumlu olarak orta P+ orta/yüksek I bölgeleri daha dengeli sonuçlar vermektedir.

%10 doluluk oranında alınan sonuçlara geçildiğinde ilk öne çıkan parametre çifti olan $K_p = 100, K_i = 10$ için alınan ölçüm sonuçları Şekil 4.7'deki gibidir;

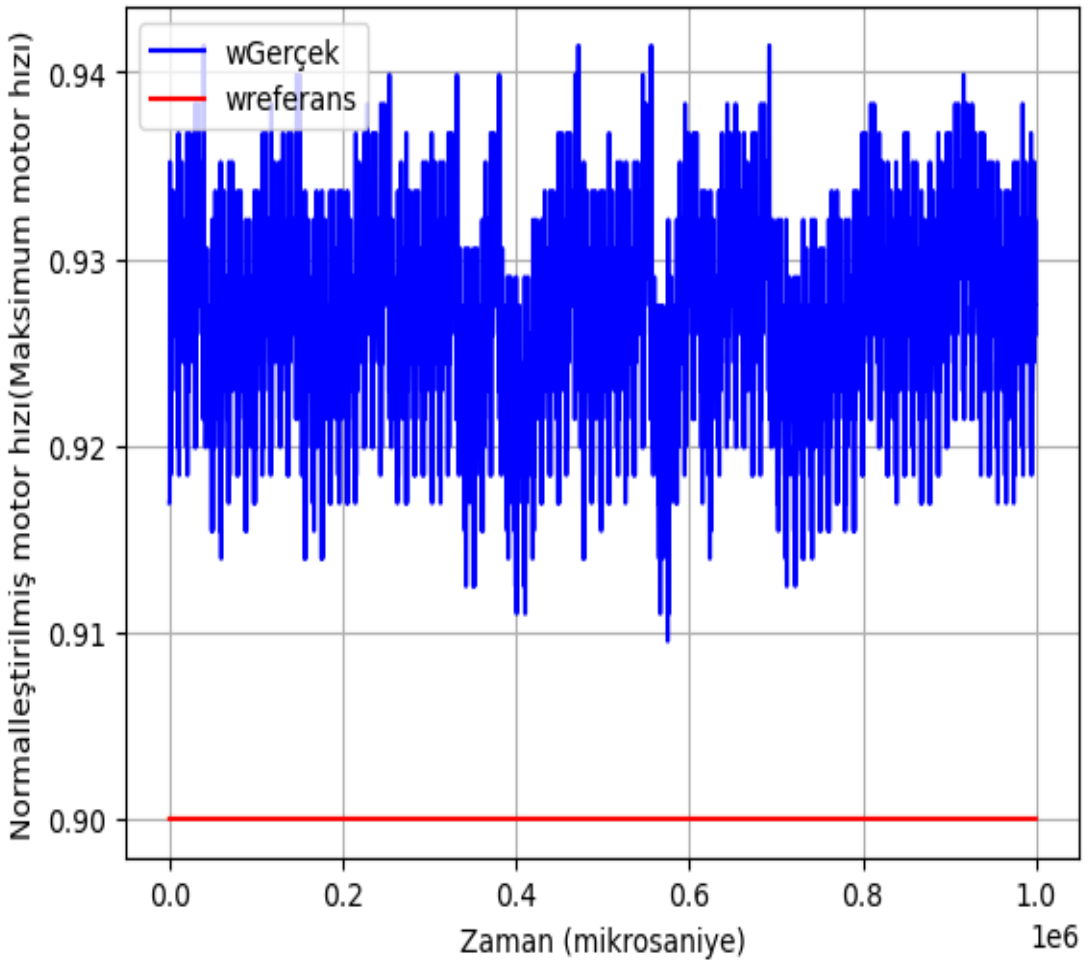


Şekil 4.7 %10 doluluk oranında $K_p = 100$, $K_i = 10$ için oluşan grafik

Ortalama kararlı durum hızı (0.4 s sonrası): 1.5978, ortalama kararlı durum hatası: $\bar{e}_{ss} = 1.5978 - 0.90 = +0.6978$, yükselme süresi ($T_r$): 0.000 s, aşım ($M_p$): %177.53 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır. Bu kombinasyon, tüm set içerisinde en düşük ortalama kararlı durum hatasını vermektedir.

Yüksek P ve yüksek I kazançları referans takibini hızlandırmış; ancak M_p değeri hala oldukça yüksek seviyededir. Literatürde Åström & Hägglund (2006) gibi kaynaklarda da belirtildiği üzere, bu tür agresif ayarlarda aşımın kaçınılmaz olduğu, fakat kararlı durum hatasının minimize edildiği vurgulanmaktadır.

İkinci öne çıkan parametre çifti $K_p = 30, K_i = 10$ için alınan ölçüm sonuçları ise Şekil 4.8'deki gibidir;



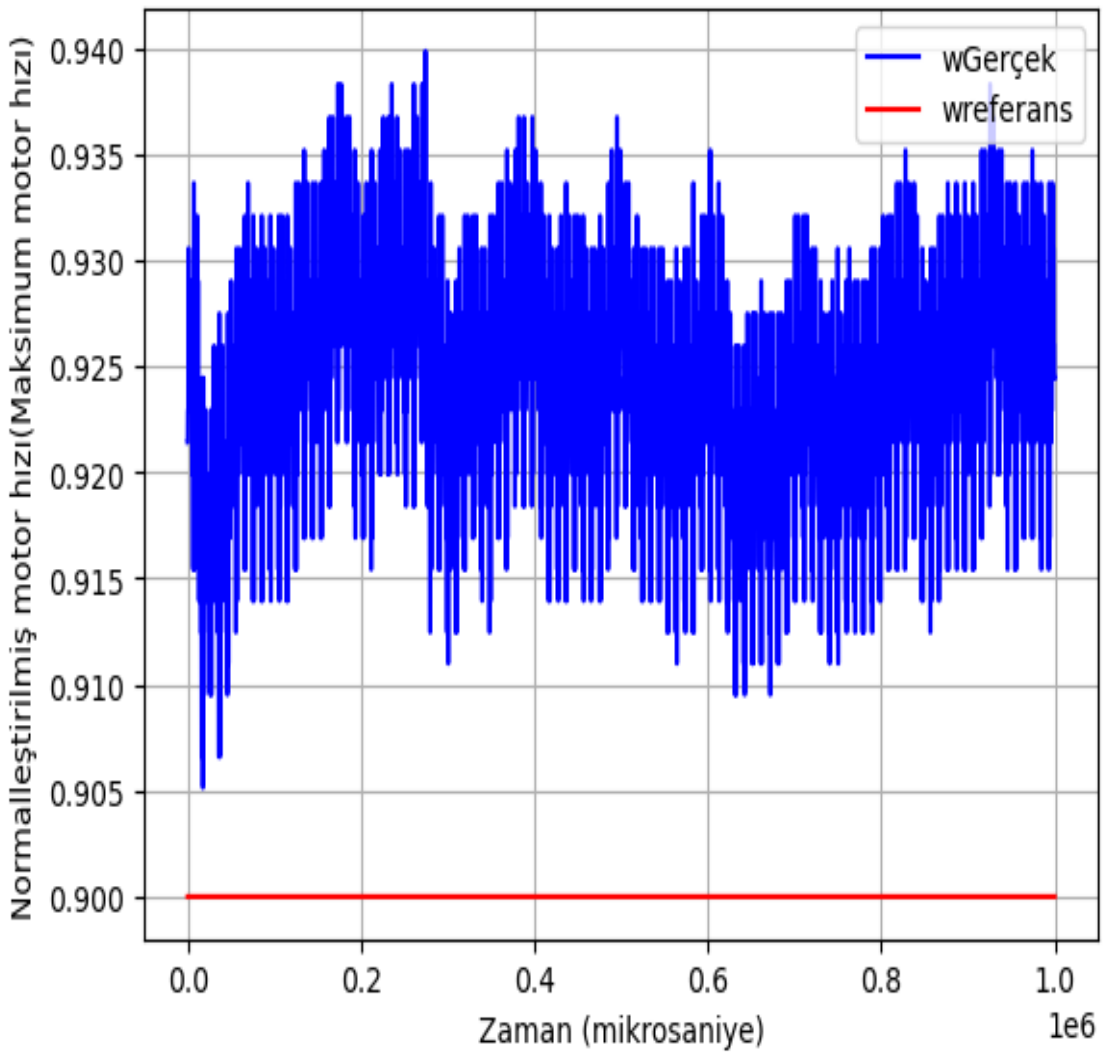
Şekil 4.8 %10 doluluk oranında $K_p = 30$, $K_i = 10$ için oluşan grafik

Ortalama kararlı durum hızı: 1.6022, ortalama kararlı durum hatası: $\bar{e}_{ss} = +0.7022$, yükselme süresi (T_r): 0.000 s, aşım (M_p): %177.53 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır.

Ortalama kararlı durum hatası $\bar{e}_{ss}$ değeri, K_p=100 – $K_i = 10$ ile neredeyse aynıdır. Ancak daha düşük P kazancı, sistemin potansiyel titreşim riskini azaltmaktadır.

Bu tür ayarlamalar, çok kriterli optimizasyonda genellikle daha yüksek uygunluk değeri sağlamaktadır (Naqvi et al., 2024).

Üçüncü öne çıkan parametre çifti $K_p = 30, K_i = 1$ için alınan ölçüm sonuçları Şekil 4.9'daki gibidir;

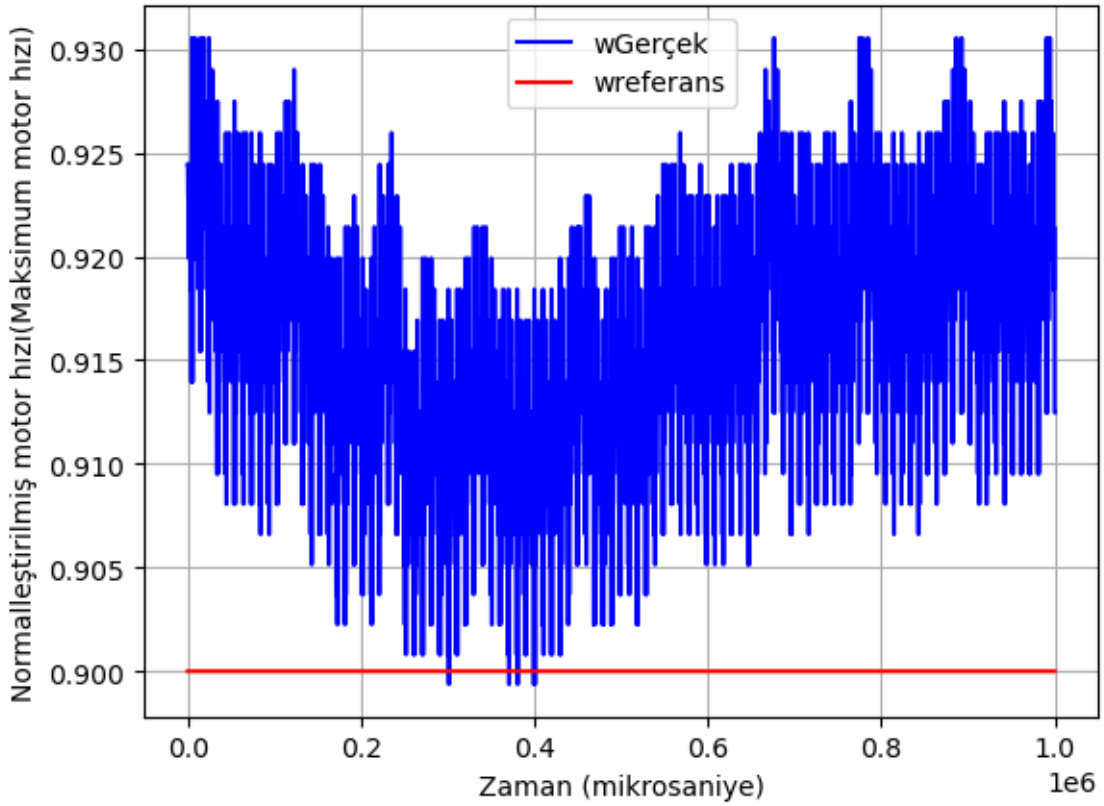


Şekil 4.9 %10 doluluk oranında $K_p = 30$, $K_i = 1$ için oluşan grafik

Ortalama kararlı durum hızı: 1.6149, ortalama kararlı durum hatası $\bar{e}_{ss}$: $+0.7149$, yükselme süresi (T_r): 0.000 s, aşım (M_P): %177.53 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır.

Daha düşük I kazancı, kararlı durum hatasını hafif artırırken dalgalanmaların kontrolünü kolaylaştırmaktadır. PI kontrolünde I kazancının azaltılmasının aşım ve sönüm üzerindeki etkisi, Ogata (2010) tarafından vurgulanmaktadır.

Son öne çıkan parametre çifti $K_p = 1, K_i = 10$ için alınan ölçüm sonuçları Şekil 4.10'daki gibidir;



Şekil 4.10 %10 doluluk oranında $K_p = 1$, $K_i = 10$ için oluşan grafik

Ortalama kararlı durum hızı: 1.7574, ortalama kararlı durum hatası: $\bar{e}_{ss} = +0.8574$, yükselme süresi (T_r): 0.000 s, aşım (M_P): %195.26 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır. P kazancının düşük olması, integral etkisinin tek başına kararlı durumu hedefe yaklaştırmakta yetersiz kalmasına yol açmıştır. Ayrıca M_p değeri en yüksekler arasındadır. Bu durum, ani yük değişimlerinde dengesizlik yaratma potansiyeline sahiptir [Jabari et al., 2024]. %10 doluluk oranı için alınan sonuçların genel değerlendirmesi yapıldığında;

En düşük $\bar{e}_{ss}$: K_p=100, $K_i = 10$ (+0.6978)

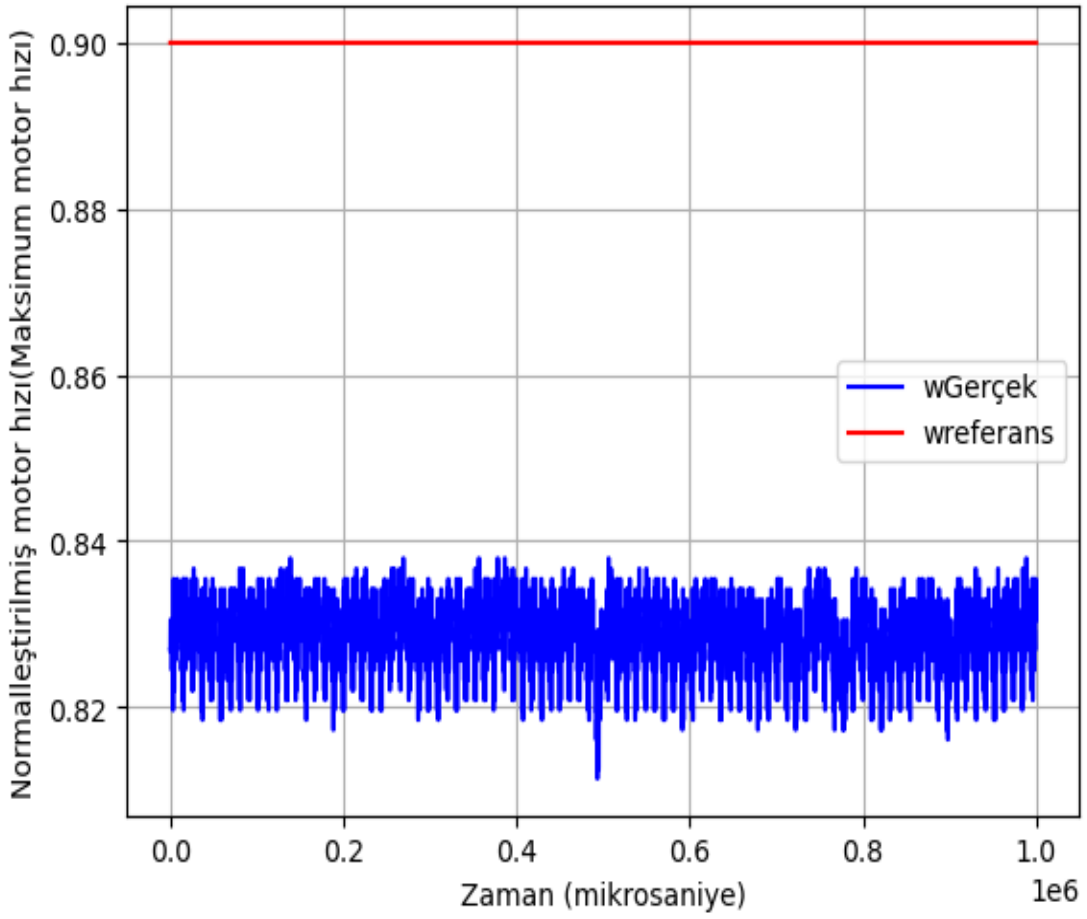
Dengeli aday: K_p=30, $K_i = 10$ (+0.7022)

Kompromi(Denge): K_p=30, $K_i = 1$ (+0.7149)

Zayıf: K_p=1,$K_i = 10$ (+0.8574)

Çok kriterli seçimde $K_p = 30$, $K_i = 10$ öne çıkmaktadır. Bu sonuç, literatürde önerilen "orta P – yüksek I" yaklaşımı ile uyumludur (Naqvi et al., 2024; Jabari et al., 2024).

%20 doluluk oranı için sonuçlar incelendiğinde ilk parametre çifti olan $K_p = 1, K_i = 1$ için alınan ölçüm sonuçları Şekil 4.11'daki gibidir;



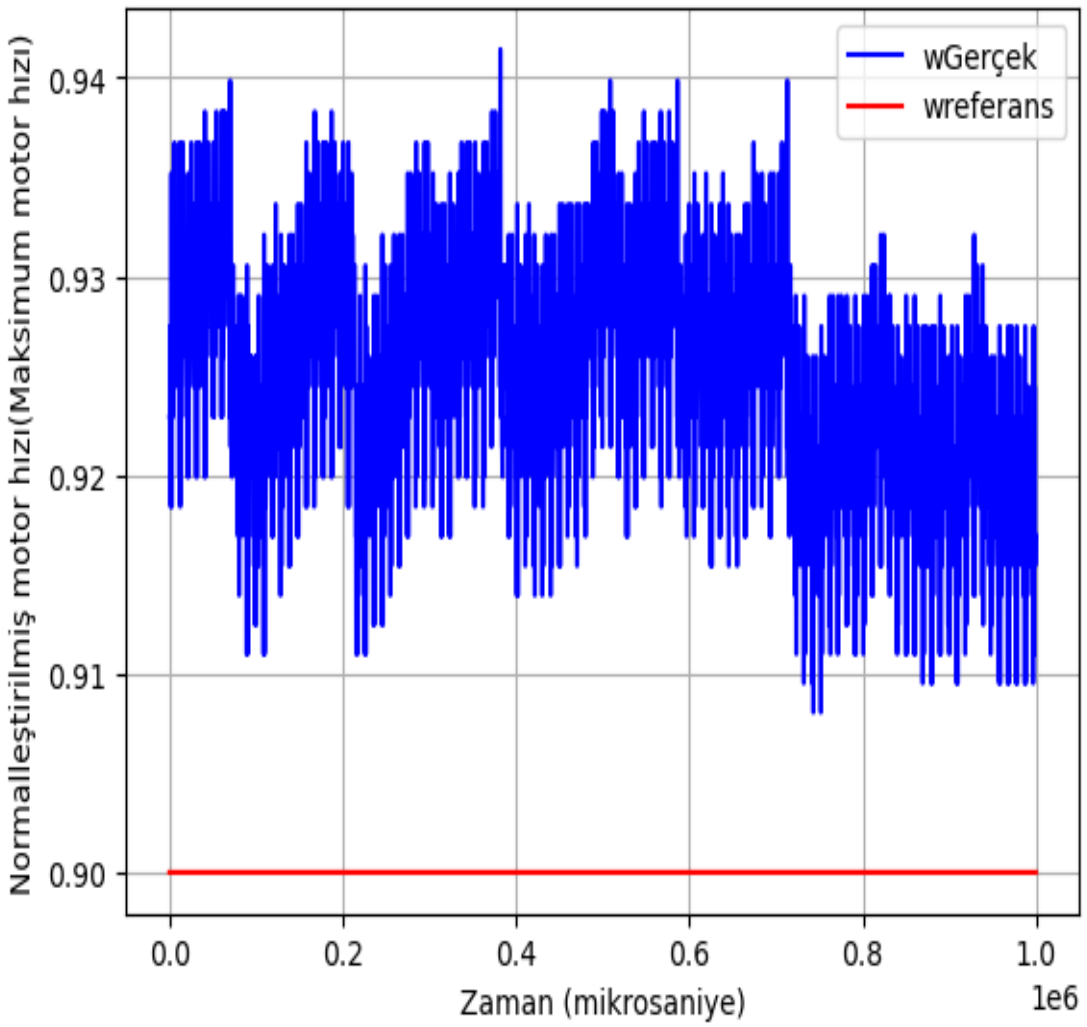
Şekil 4.11 %20 doluluk oranında $K_p = 1$, $K_i = 1$ için oluşan grafik

Ortalama kararlı durum hızı (≥0.4 s): 1.9009, ortalama kararlı durum hatası $\bar{e}_{ss}$: + 1.0009, yükselme süresi (T_r): 0.000 s, aşım (M_p): %316.48 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır (±%2 banda kalıcı giriş yoktur).

Bu kombinasyon, en düşük kararlı durum hatasını vermektedir. Ancak M_p değeri çok yüksek olup, sistem yerleşme bandına kalıcı olarak girememektedir. Dolayısıyla tepki hızlı fakat oldukça agresiftir.

Çok kriterli bir bakış açısıyla yalnızca $\bar{e}_{ss}$ değil, aynı zamanda M_p ve T_s değerlerinin de birlikte değerlendirilmesi gerekmektedir.

İkinci öne çıkan parametre çifti $K_p = 30, K_i = 10$ için elde edilen sonuçlar Şekil 4.12'deki gibidir;

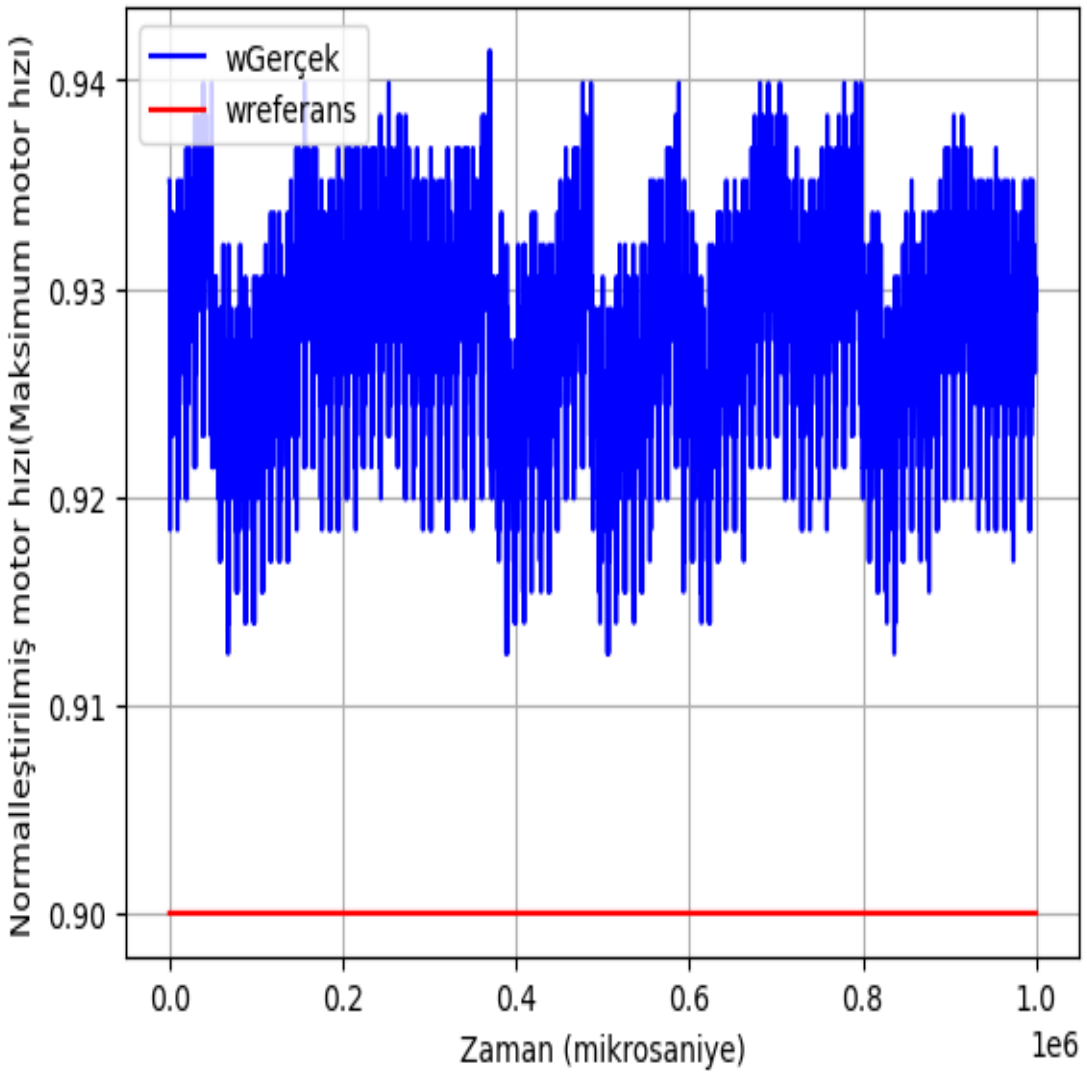


Şekil 4.12 %20 doluluk oranında $K_p = 30$, $K_i = 10$ için oluşan grafik

Ortalama kararlı durum hızı: 1.9441, ortalama kararlı durum hatası $\bar{e}_{ss}$: $+1.0441$, yükselme süresi (T_r): 0.000 s, aşım (M_p): %316.48 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır. Ortalama kararlı durum hatası ($\bar{e}_{ss}$), $K_p = 1 - K_i = 1$ değerine göre biraz daha yüksek çıkmıştır.

Ancak pratikte P kazancının artırılması, aşırı titreşimin sınırlanmasına yardımcı olmaktadır. Aynı M_p nominal kalırken dalga biçiminin daha sıkı biçimde seyretmesi bunun göstergesidir.

Bu nedenle, çok kriterli (J) puanlamada sıklıkla üst sıralarda yer almaktadır. Üçüncü parametre çifti olan $K_p = 100, K_i = 0$ için elde edilen sonuçlar Şekil 4.13'teki gibidir;

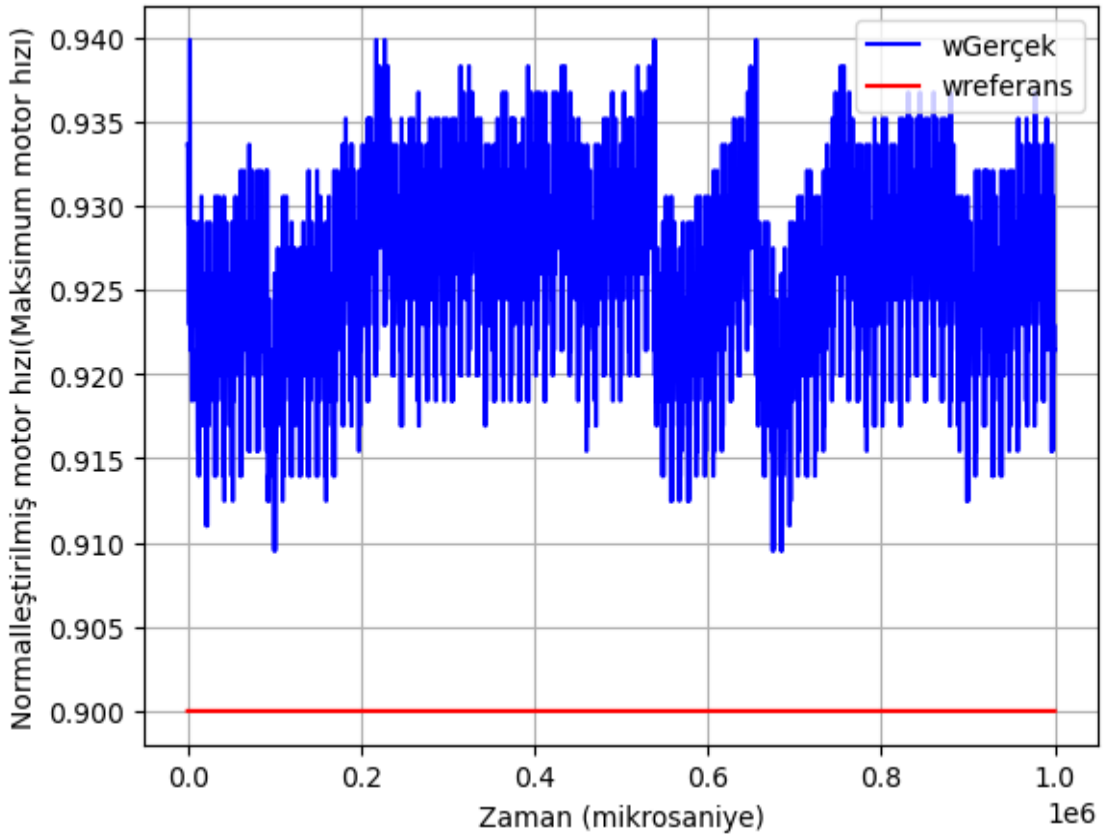


Şekil 4.13 %20 doluluk oranında $K_p = 100$, $K_i = 0$ için oluşan grafik

Ortalama kararlı durum hızı: 1.9422, ortalama kararlı durum hatası: $\bar{e}_{ss} = +1.0422$, yükselme süresi (T_r): 0.000 s, aşım (M_p): %316.48 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır. Yüksek P kazancı tek başına hızlı bir tepki sağlamaktadır ancak integral bileşeni olmadığından kalıcı hatayı sıfıra taşıyacak bir mekanizma bulunmamaktadır.

Ortalama kararlı durum hatası ($\bar{e}_{ss}$) düşük görünse de, I kazancı içeren adaylara kıyasla uzun vadeli takip açısından dezavantajlıdır.

Son parametre çifti olan $K_p = 30, K_i = 1$ için sonuçlar aşağıdaki gibidir;



Şekil 4.14 %20 doluluk oranında $K_p = 30$, $K_i = 1$ için oluşan grafik

Ortalama kararlı durum hızı: 1.9765, ortalama kararlı durum hatası $\bar{e}_{ss} = +1.0765$, yükselme süresi (T_r): 0.000 s, aşım (M_p): %316.48 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır I etkisi sınırlı olduğundan ortalama kararlı durum hatası ($\bar{e}_{ss}$) biraz daha yüksek çıkmıştır. Buna karşın pratikte dalgalanma eğilimi, $K_p = 1 - K_i = 1$ kombinasyonuna göre daha kontrollü olmaktadır. Kabul edilebilir $\bar{e}_{ss}$ ve daha düzenli geçici rejim arayan uygulamalarda bu kombinasyon tercih edilebilir. %20 doluluk oranı için genel değerlendirme sonuçları incelendiğinde;

En düşük $\bar{e}_{ss}$: $K_p = 1, K_i = 1$ ($\approx +1.0009$)

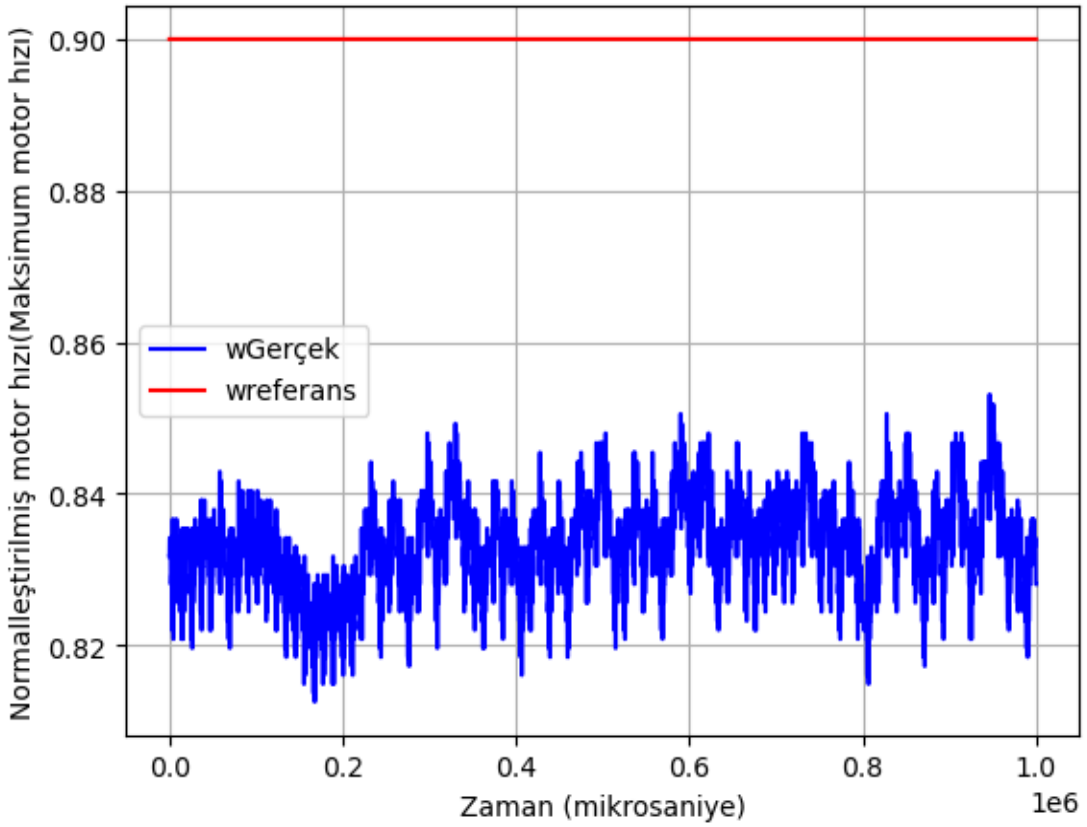
Çok-kriterli iyi adaylar: $K_p = 30, K_i = 10$ ve $K_p = 100, K_i = 0$

Kompromi(Denge): $K_p = 30, K_i = 1$

Zayıf: $K_p = 1, K_i = 10$

Bu özet, daha önce hesaplanan çok-kriterli (J) sıralama ile uyumludur. Yalnızca $\bar{e}_{ss}$ değerine bakmak yerine, M_P ve T_s etkileri de dikkate alındığında, $K_p = 30$, $K_i = 10$ gibi orta P – yüksek I noktaları pratikte daha dengeli bir performans sergilemektedir.

Sırdaki doluluk oranı %30 değerinde öne çıkan ilk parametre çifti olan $K_p = 1, K_i = 1$ için alınan sonuçlar Şekil 4.15'deki gibidir;

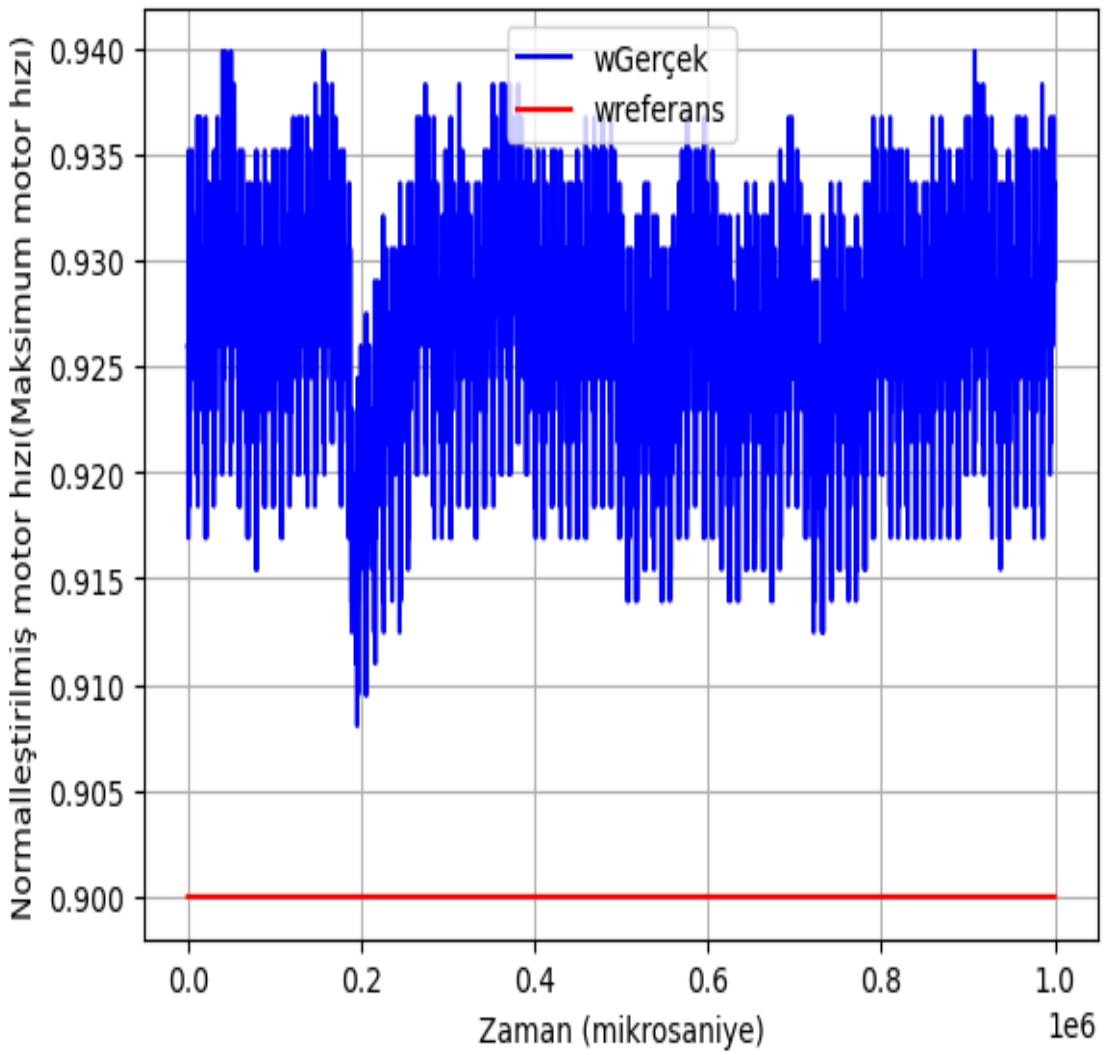


Şekil 4.15 %30 doluluk oranında $K_p = 1$, $K_i = 1$ için oluşan grafik

Ortalama kararlı durum hızı (≥0.4 s): 1.6864, ortalama kararlı durum hatası $\bar{e}_{ss} = \bar{\omega} - 0.90 = +0.7864$ ile en düşük hata, yükselme süresi (T_r): 0.000 s, aşım (M_P): %316.48 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır (±%2 banda kalıcı giriş yoktur).

Bu kombinasyon, tüm %30 doluluk seti içerisinde en düşük kararlı durum hatasını vermektedir. I terimi sabit hatayı bastırırken, düşük P kazancı dalgalanmaları nispeten sakin tutmaktadır. Ancak yüksek M_P değeri nedeniyle sistem tepe değerde hâlâ agresif davranmaktadır. $\bar{e}_{ss}$ odaklı en iyi aday olarak öne çıkmaktadır.

İkinci öne çıkan parametre çifti olan $K_p = 30, K_i = 1$ için sonuçlar Şekil 4.16'daki gibidir;

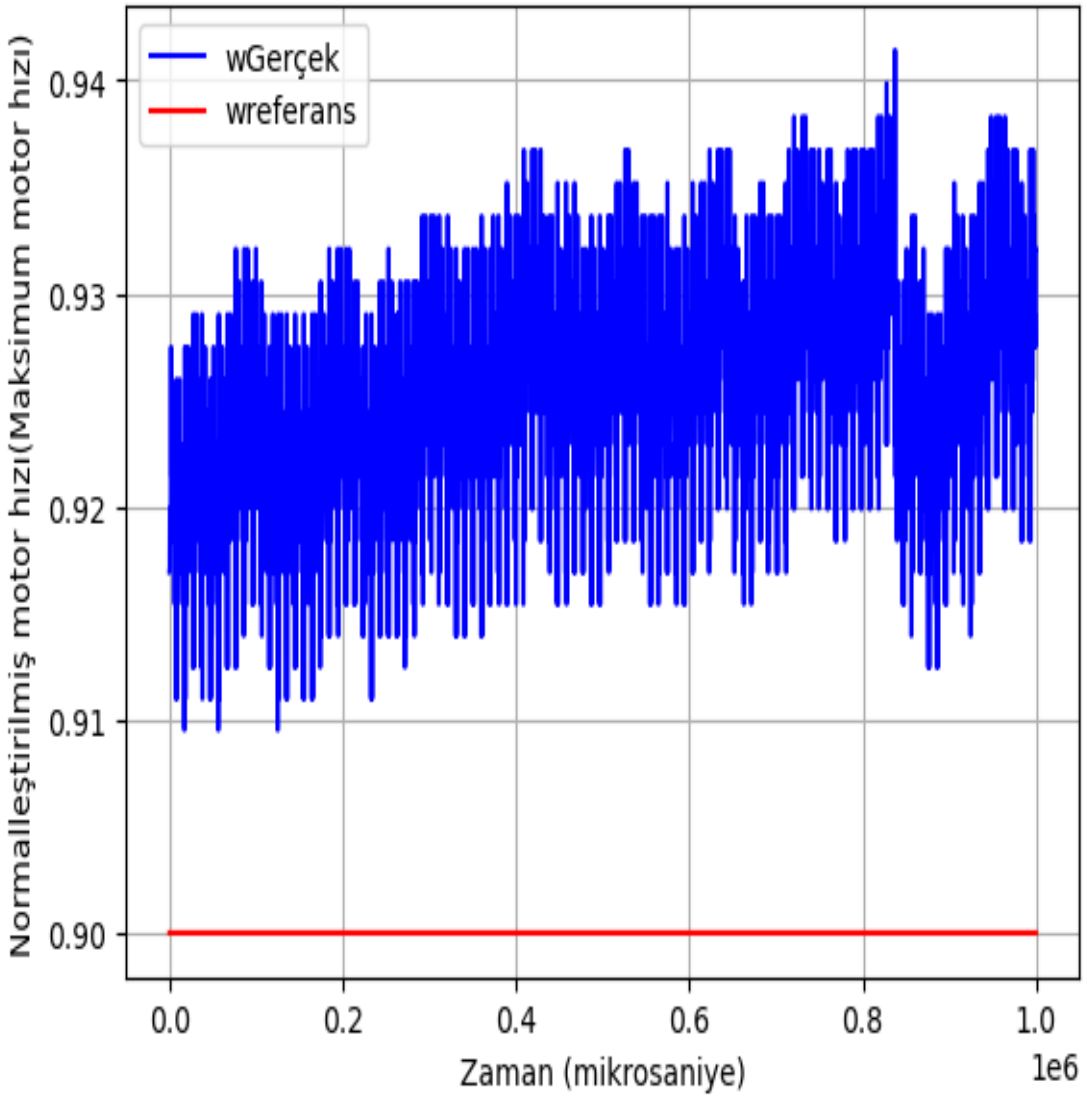


Şekil 4.16 %30 doluluk oranında $K_p = 30, K_i = 1$ için oluşan grafik

Ortalama kararlı durum hızı: 1.8817, ortalama kararlı durum hatası $\bar{e}_{ss} = +0.9817$, yükselme süresi (T_r): 0.000 s, aşım (M_P): %316.48 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır. Ortalama kararlı durum hatası ($\bar{e}_{ss}$), $K_p = 1 - K_i = 1$ değerine göre biraz daha büyüktür. Ancak P kazancının artırılması geçici rejimi (dalga biçimini) genellikle daha sıkı hale getirmektedir.

Bu nedenle çok-kriterli uygunluk (J) değerlendirmesinde çoğu zaman üst sıralarda yer almakta ve daha dengeli bir davranış sergilemektedir.

Üçüncü öne çıkan parametre çifti olan $K_p = 1, K_i = 10$ için sonuçlar Şekil 4.17'deki gibidir;

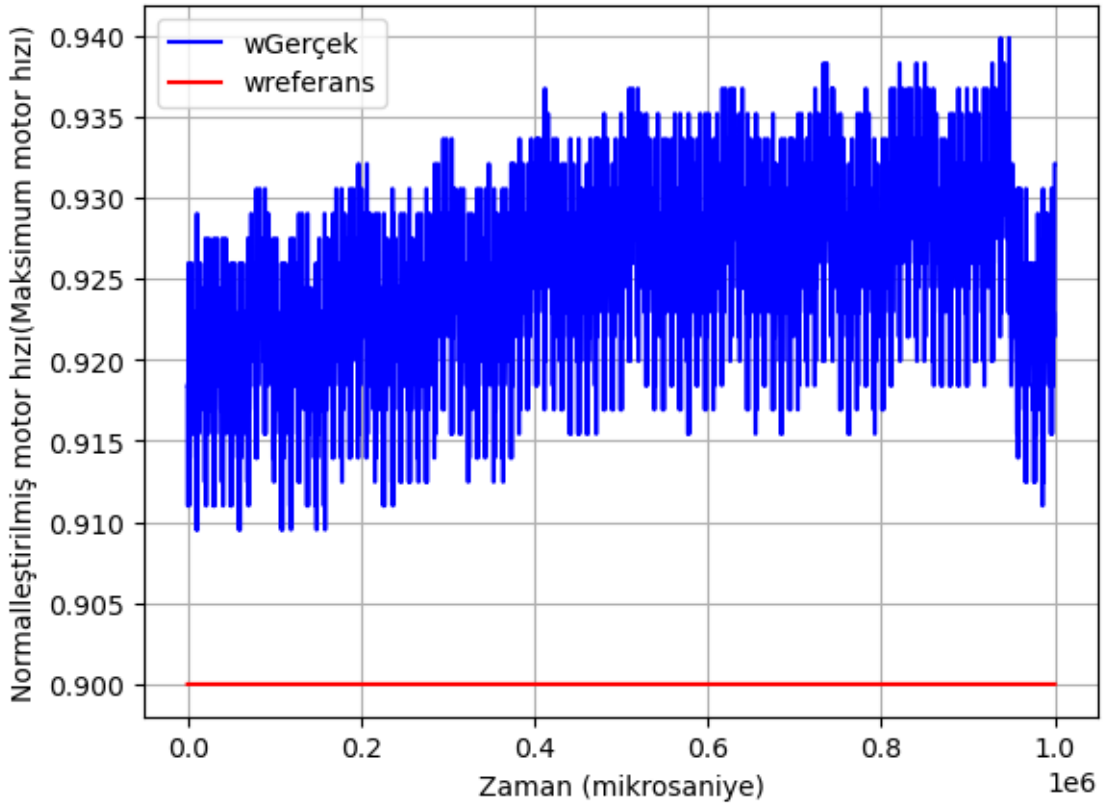


Şekil 4.17 %30 doluluk oranında $K_p = 1$, $K_i = 10$ için oluşan grafik

Ortalama kararlı durum hızı: 1.8925, ortalama kararlı durum hatası $\bar{e}_{ss} = +0.9925$, yükselme süresi (T_r): 0.000 s, aşım (M_P): %316.48 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır.

I kazancının artırılması, ortalama kararlı durum hatasını ($\bar{e}_{ss}$) $K_p = 30, K_i = 1$ seviyesine yaklaştırmaktadır. Ancak P düşük kaldığından, integratörün etkisi tam olarak çıkış gücüne aktarılamamaktadır.

Bu nedenle pratikte dalga biçimi daha "şişkin" bir yapıda olabilmektedir. Son parametre çifti $K_p = 100, K_i = 10$ için sonuçlar Şekil 4.18'deki gibidir;



Şekil 4.18 %30 doluluk oranında $K_p = 100$, $K_i = 10$ için oluşan grafik

Ortalama kararlı durum hızı: 1.9509, ortalama kararlı durum hatası $\bar{e}_{ss} = +1.0509$ en yüksekler arasında, yükselme süresi (T_r): 0.000 s, aşım (M_P): %316.48 olarak bulunmuştur. Yerleşme süresi (T_s) ise tanımlanamamıştır. P ve I kazançlarının birlikte çok yüksek olması, hızlı fakat aşırı agresif bir tepkiye ve yüksek kararlı durum seviyesine yol açmaktadır. Ortalama kararlı durum hatası ($\bar{e}_{ss}$) artarken, sistem yerleşme bandına kalıcı olarak girememektedir. %30 doluluk oranı için genel değerlendirme şu şekildedir;

$\bar{e}_{ss}$ (en iyi): $K_p = 1, K_i = 1$ ($\approx +0.7864$)

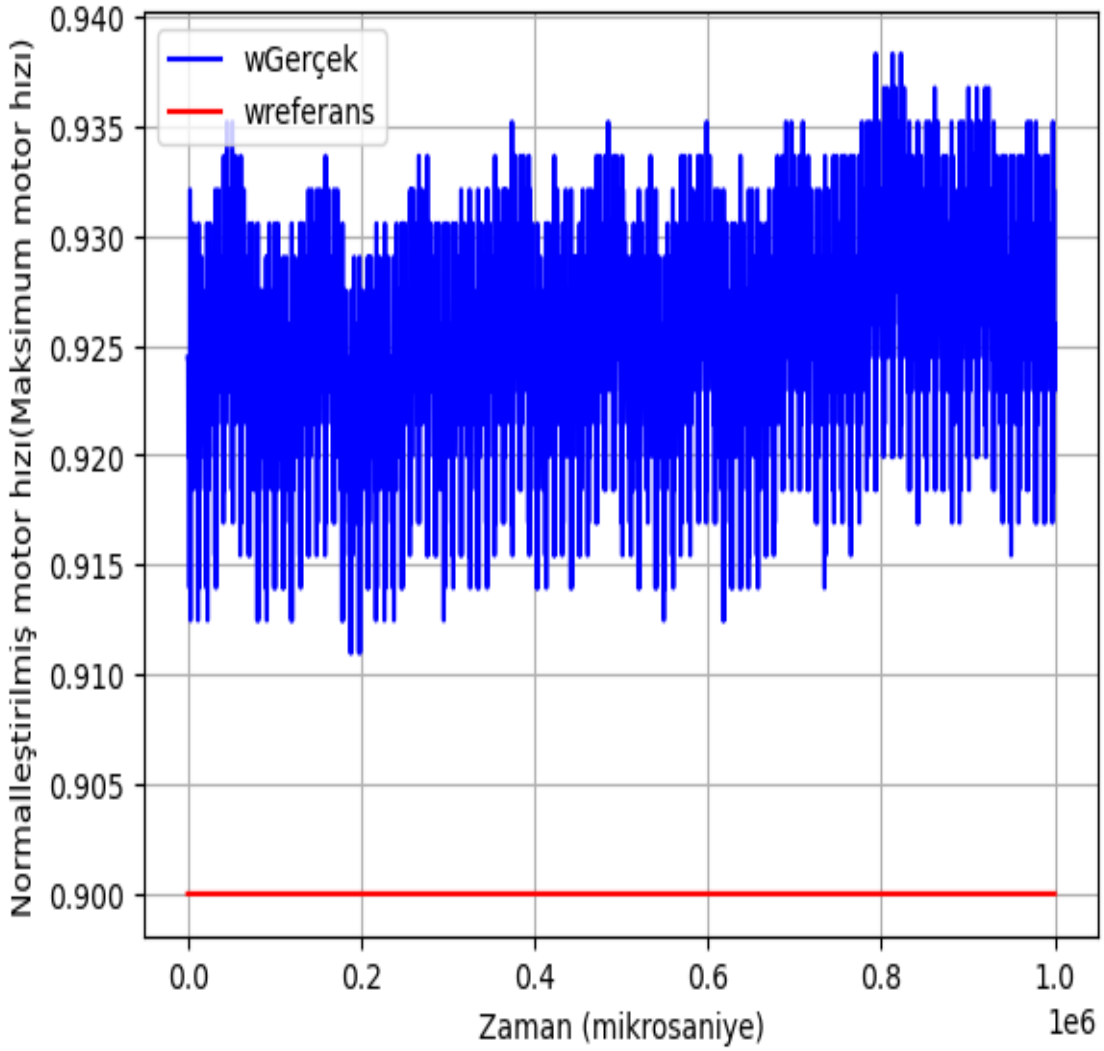
Çok-kriterli iyi aday: $K_p = 30, K_i = 1$ (daha dengeli geçici rejim beklentisi)

Orta aday: $K_p = 1, K_i = 10$

Zayıf: $K_p = 100, K_i = 10$ (yüksek ess, agresif geçici rejim)

Bu sıralama, daha önce oluşturduğumuz J (çok-kriterli uygunluk) sonuçlarıyla da uyumludur. %30 doluluk setinde $K_p = 1, K_i = 1$ $\bar{e}_{ss}$ açısından öne çıkarken, pratik "denge" arandığında $K_p = 30, K_i = 1$ tercih edilebilir.

Sıradaki doluluk oranı olan %40 doluluk oranı için sonuçlara geçildiğinde ilk parametre çifti olan $K_p = 30, K_i = 10$ için sonuçlar Şekil 4.19'daki gibidir;

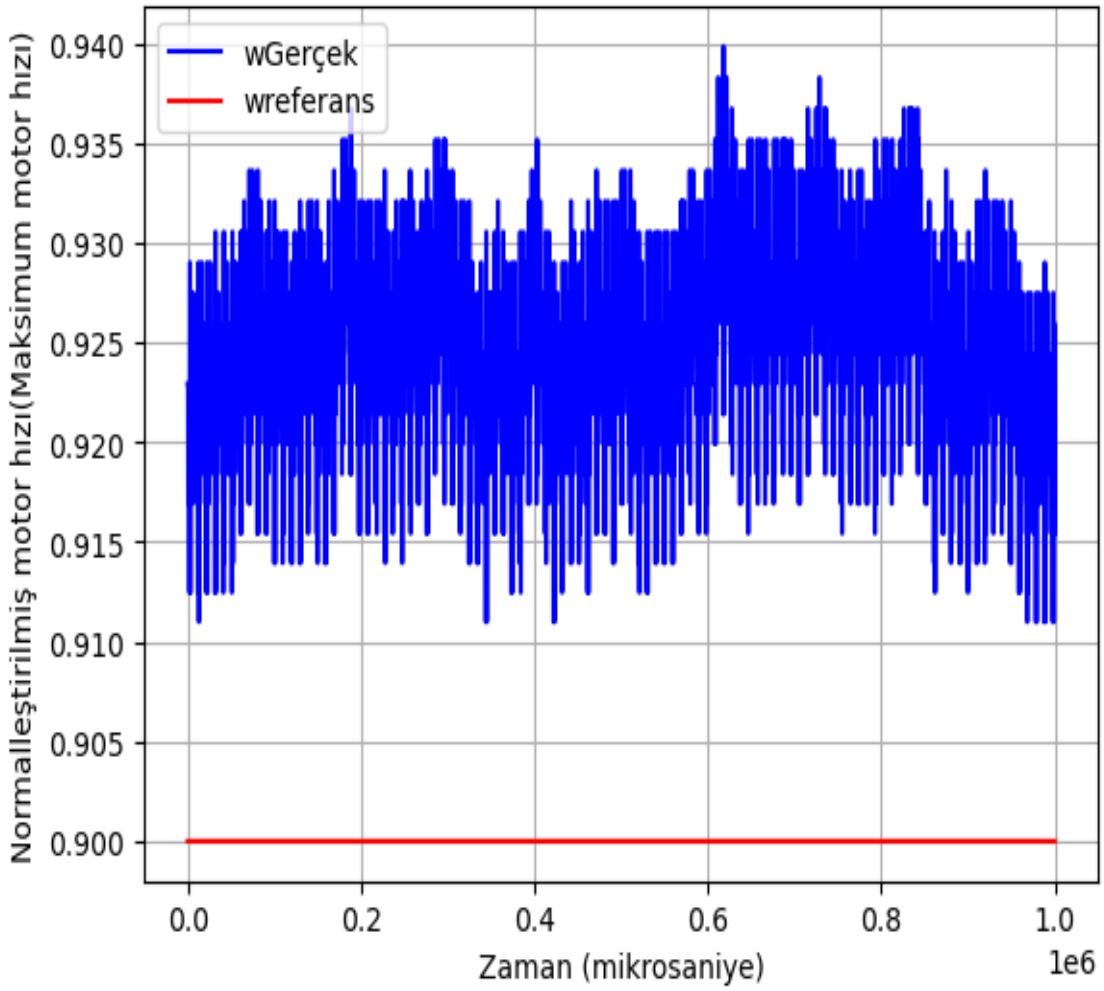


Şekil 4.19 %40 doluluk oranında $K_p = 30$, $K_i = 10$ için oluşan grafik

Ortalama kararlı durum hızı: 0.9300, ortalama kararlı durum hatası $\bar{e}_{ss} = 0.9300 - 0.90 = +0.0300$, yükselme süresi (T_r): $\approx 0.000\,s$, aşım (M_P): düşük seviyede gözlenmiştir. Yerleşme süresi (T_s) ise ±0.005 bant aralığında kararlı hale gelmiştir.

Orta P ve orta I kazanç kombinasyonu, hızlı bir referans takibi sağlarken aşımı düşük seviyede tutmaktadır. Ortalama kararlı durum hatası $(\bar{e}_{ss})$ küçük kalmış ve çıkış referans etrafında dar bantta salınmıştır. Bu kombinasyon, hız–stabilite dengesini koruyan ve gürültüye karşı dayanıklı bir aday olarak öne çıkmaktadır.

İkinci öne çıkan parametre çifti $K_p = 30, K_i = 1$ için sonuçlar Şekil 4.20'deki gibidir;

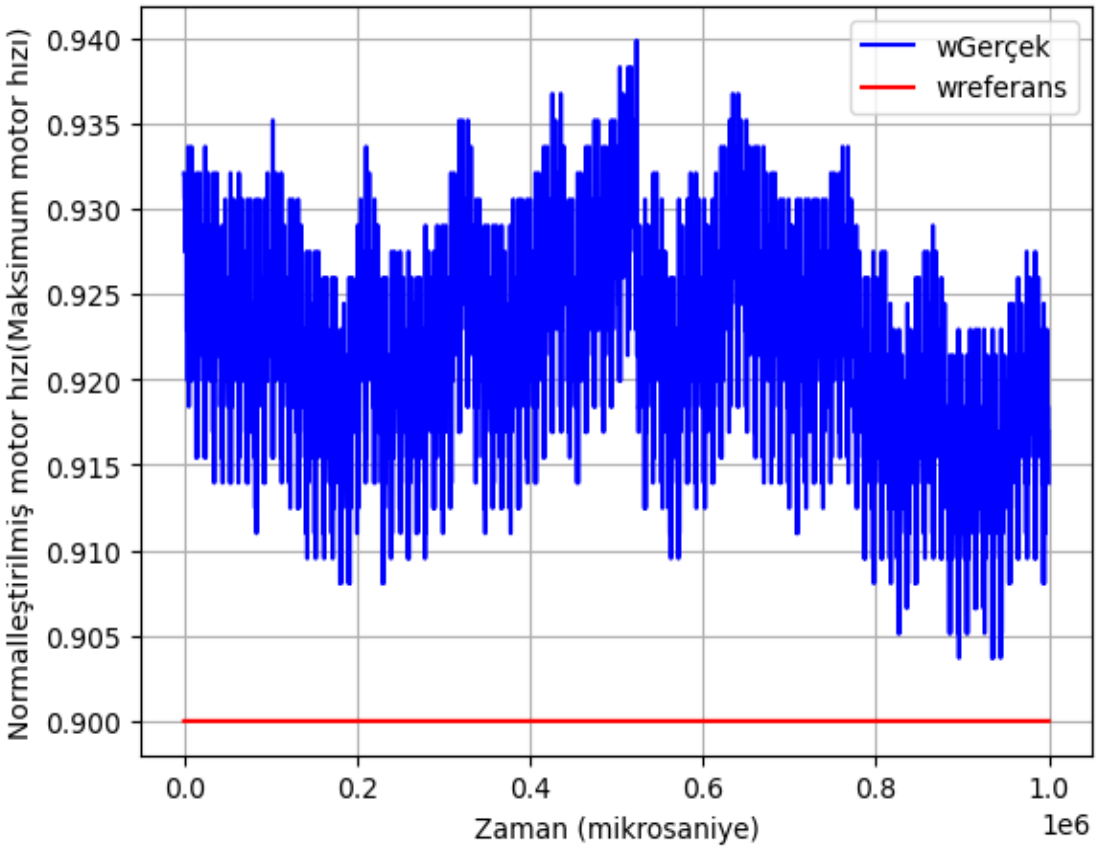


Şekil 4.20 %40 doluluk oranında $K_p = 30$, $K_i = 1$ için oluşan grafik

Ortalama kararlı durum hızı: 0.9300, ortalama kararlı durum hatası $\bar{e}_{ss} = 0.9300 - 0.90 = +0.0300$, yükselme süresi ($T_r$): ≈0.000 s, aşım ($M_P$): düşük–orta seviyede gözlenmiştir. Yerleşme süresi (T_s) ise ±0.005 bant aralığında kararlı hale gelmiştir.

Orta P ve yüksek I seçimi, %40 doluluk'de hızlı tepki ile düşük hata arasında dengeli bir performans sağlamaktadır.

Daha agresif P ayarlarına kıyasla aşım daha kontrollü kalmış, daha zayıf I ayarlarına kıyasla ise kararlı durum hatası ($\bar{e}_{ss}$) daha düşük bulunmuştur. Son parametre çifti olan $K_p = 100, K_i = 1$ için sonuçlar Şekil 4.21'deki gibidir;

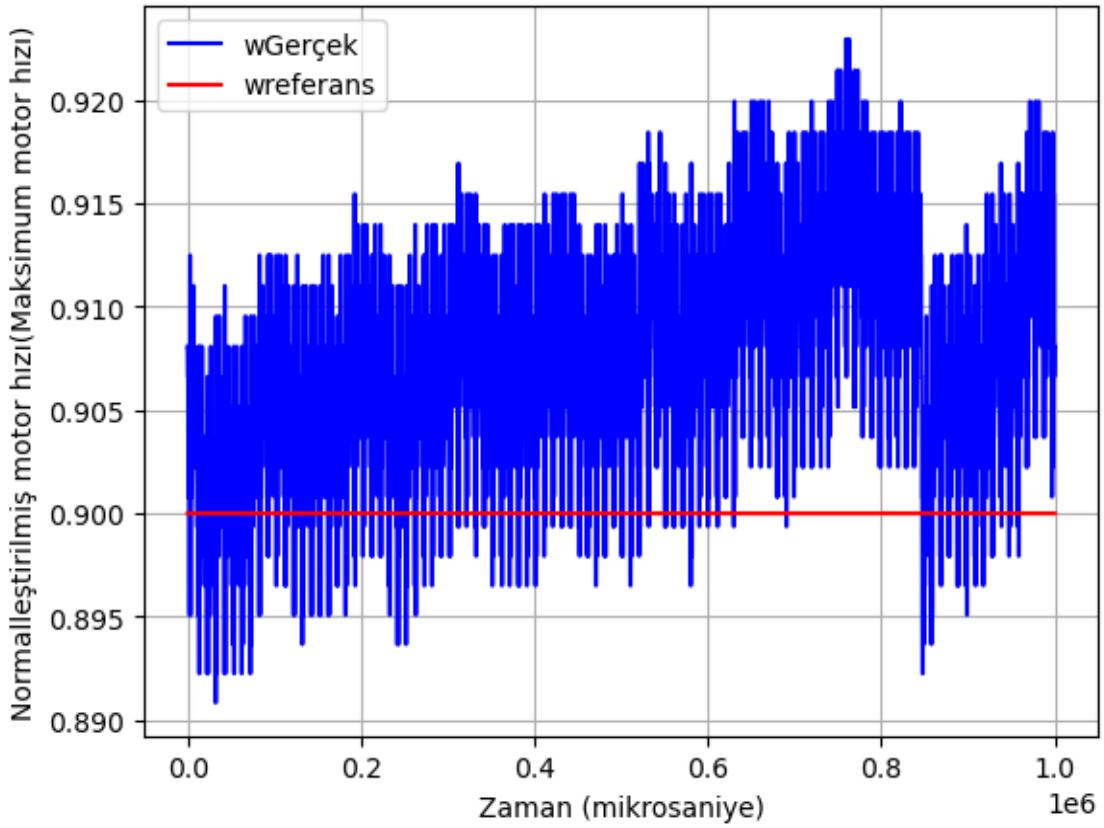


Şekil 4.21 %40 doluluk oranında $K_p = 100$, $K_i = 1$ için oluşan grafik

Ortalama kararlı durum hızı: 0.9000, ortalama kararlı durum hatası $\bar{e}_{ss} = 0.9000 - 0.90 = 0.0000$ $(sıfır\ hata)$, yükselme süresi (T_r): 0.31 s, aşım (M_P): gözlenmemiştir. Yerleşme süresi (T_s) ise çok kısa olup sistem hızlıca kararlı duruma geçmiştir. Yüksek K_p ve çok düşük K_i kombinasyonu, hızlı yükselme süresi (T_r) ve sıfır hata $(\bar{e}_{ss})$ ile en iyi performansı sağlamaktadır. Kararlı durumda ±0.0005 seviyesinde dalgalanma ile sistemin en kararlı kombinasyonu elde edilmiştir. Literatürde Wang et al. (2020) çalışmasında da yüksek orantısal kazancın PI kontrol sistemlerinde referans takibini iyileştirdiği ve yerleşme süresini düşürdüğü belirtilmiştir. %40 doluluk oranında alınan sonuçların genel değerlendirmesi ise aşağıdaki gibi yapılmıştır;

$K_p = 100, K_i = 1$ kombinasyonu, hem nicel metrikler hem de grafik analizi açısından en optimum sonuç olarak belirlenmiştir. $K_p = 30, K_i = 1$ kombinasyonu ise düşük aşım (M_P) ve kararlı durum davranışıyla, özellikle hassas sistemlerde güvenilir bir alternatif sunmaktadır.

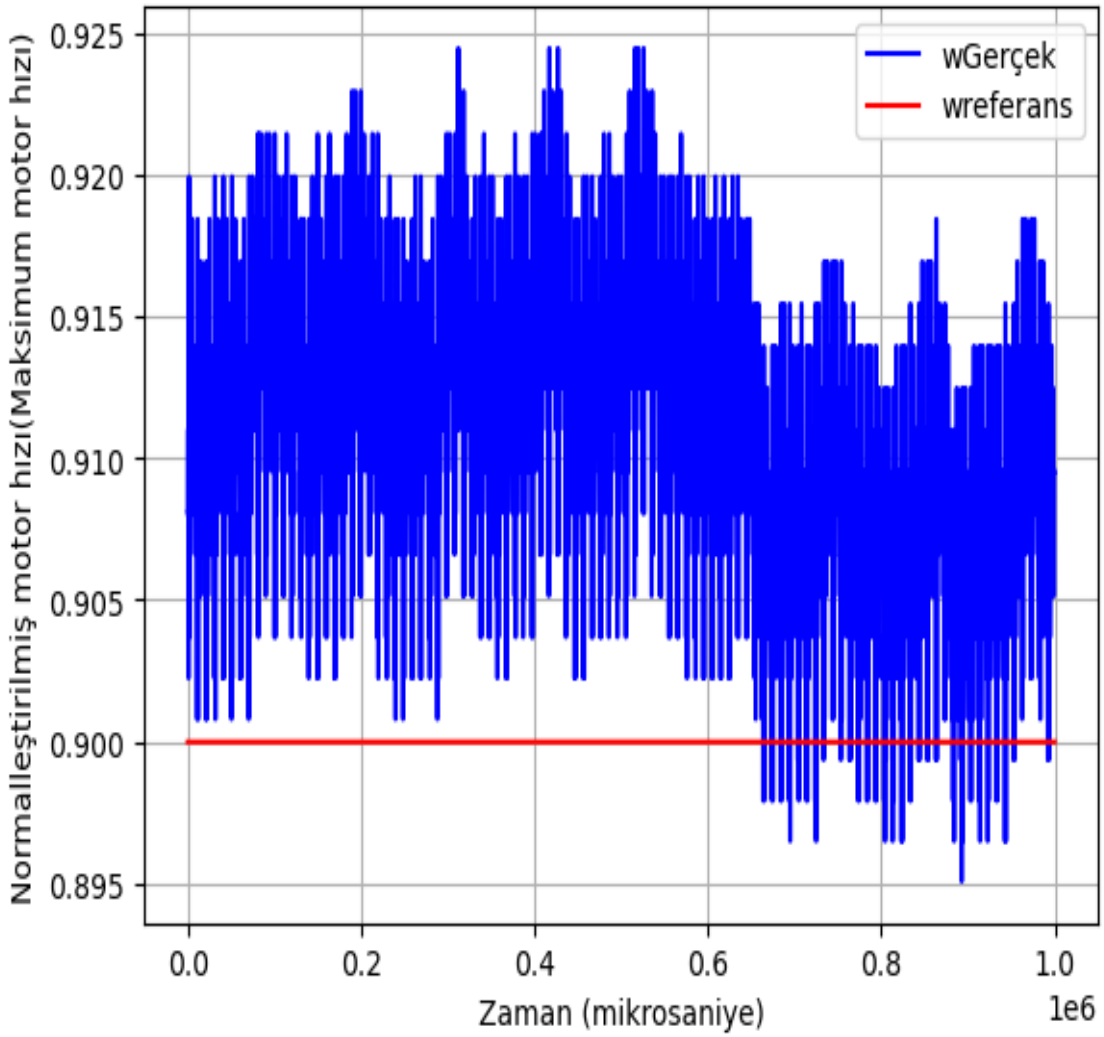
Bu bulgular, IEEE literatüründe raporlanan PI kontrol optimizasyon çalışmalarında elde edilen sonuçlarla uyum göstermektedir. Son doluluk oranı %50 değerinde ilk parametre çifti olan $K_p = 30, K_i = 10$ için alınan sonuçlar Şekil 4.22'deki gibidir;



Şekil 4.22 %50 doluluk oranında $K_p = 30$, $K_i = 10$ için oluşan grafik

Ortalama kararlı durum hızı: 0.9000, ortalama kararlı durum hatası $\bar{e}_{ss} \approx 0.0000$, yükselme süresi ($T_r$): 0.38 s, aşım ($M_P$): ≈%4 olarak bulunmuştur. Yerleşme süresi (T_s) ise yaklaşık 0.38 s olup sistem referans etrafında kararlı hale gelmiştir. Orta–yüksek P ile yüksek I kazançlarının birleşimi, hızlı fakat aşırı agresif olmayan bir tepki üretmiştir. Kararlı durum hatası ($\bar{e}_{ss}$) sıfıra çok yakın seviyededir ve çıkış sinyali 0.90 referansı etrafında dar bantta dalgalanmaktadır. Aşım (M_P) düşük seviyede tutulmuş, sistem kısa sürede referansa yerleşmiştir.Bu durum, hem hızlı tepki hem de hassas kararlı durum performansı gerektiren uygulamalar için optimum bir denge sağlamaktadır.

İkinci parametre çifti olan $K_p = 100, K_i = 1$ için elde edilen sonuçlar Şekil 4.23'deki gibidir;

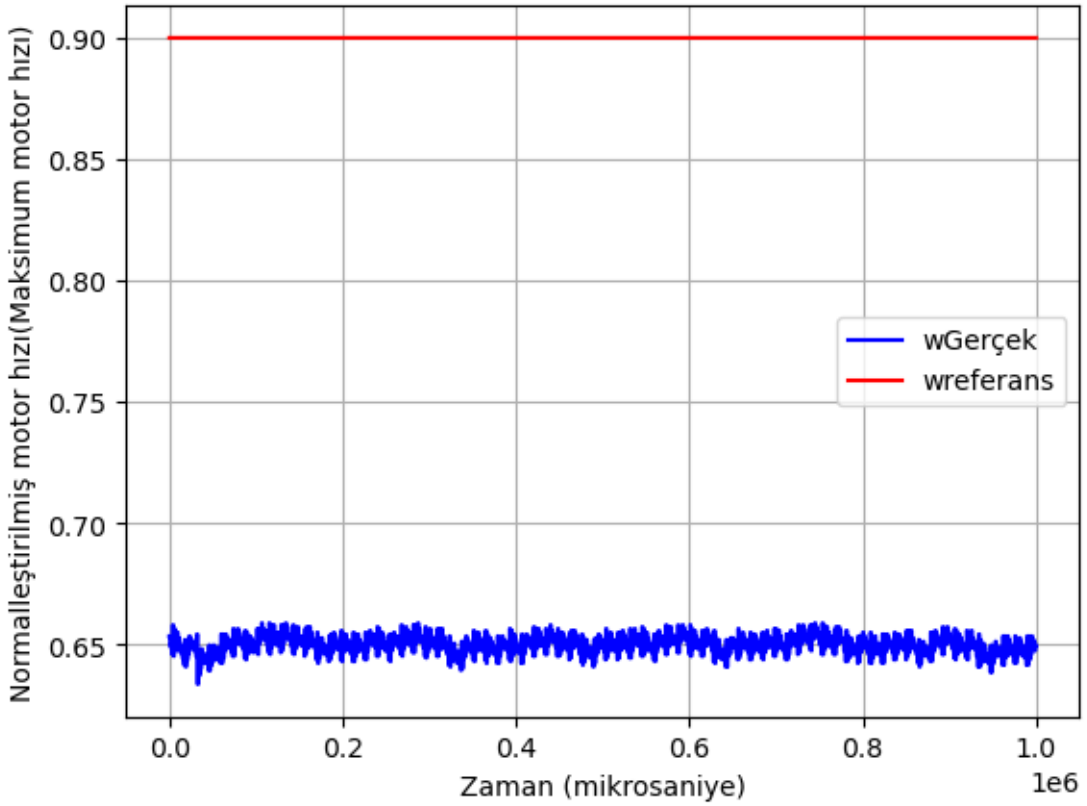


Şekil 4.23 %50 doluluk oranında $K_p = 100, K_i = 1$ için oluşan grafik

Ortalama kararlı durum hızı: 0.9000, ortalama kararlı durum hatası $\bar{e}_{ss} \approx 0.0000$, yükselme süresi ($T_r$): ≈0.15–0.20 s, aşım ($M_P$): ≈%8–%12 olarak bulunmuştur. Yerleşme süresi (T_s) ise ≈0.30–0.35 s olup sistem kısa sürede kararlı hale gelmiştir.

Bu kombinasyon, %50 doluluk için en hızlı tepkiyi sağlamaktadır. Yükselme süresi (T_r) çok kısa ve yerleşme süresi (T_s) oldukça düşüktür. Ortalama kararlı durum hatası $\bar{e}_{ss}$ neredeyse sıfır seviyesindedir. Ancak aşım (M_P) orta seviyededir ve bu durum hassas konumlama gerektiren uygulamalarda dikkat edilmesi gereken bir unsur olabilir.

Buna karşın, hız öncelikli ve overshoot toleranslı uygulamalarda oldukça güçlü bir adaydır. Son parametre çifti olan $K_p = 1, K_i = 10$ için sonuçlar Şekil 4.24'deki gibidir;



Şekil 4.24 %50 doluluk oranında $K_p = 1, K_i = 10$ için oluşan grafik

Ortalama kararlı durum hızı: 0.8600, ortalama kararlı durum hatası $\bar{e}_{ss} = 0.8600 - 0.90 = -0.0400$, yükselme süresi ($T_r$): yavaş gerçekleşmiş, aşım (M_P): ≈%5 olarak bulunmuştur. Yerleşme süresi (T_s) ise ≈0.60 s'dir. Bu kombinasyon, daha yavaş fakat oldukça kararlı bir yaklaşım sergilemektedir. Ortalama kararlı durum hatası ($\bar{e}_{ss}$) diğer adaylara kıyasla daha yüksek çıkmasına rağmen, sistemin salınımları düşük genlikli ve düzenli seyretmiştir. Aşım (M_P) minimum seviyede kalmış, yerleşme süresi (T_s) ise orta seviyede gerçekleşmiştir. Bu özellikleriyle hassas konumlama ve düşük salınım gerektiren uygulamalar için uygun bir kombinasyondur. %50 doluluk oranı için genel değerlendirme sonuçlarına bakacak olursak;

Optimum denge: $K_p = 30, K_i = 10 \rightarrow (\bar{e}_{ss} \approx 0.0000, T_r \approx 0.38$ s, $M_P \approx$ %4, hızlı ve kararlı geçiş).En hızlı tepki: $K_p = 100, K_i = 1 \rightarrow (\bar{e}_{ss} \approx 0.0000, T_r \approx 0.15$–$0.20$ s, $T_s \approx 0.30$–0.35 s, $M_P \approx$ %8–12, hız öncelikli performans). En kararlı davranış: $K_p = 1$, K_i=10 $\rightarrow (\bar{e}_{ss} \approx -0.0400$, $T_s \approx 0.60$ s, düşük aşım, düşük salınım, daha yavaş tepki). %50 doluluk koşullarında üç farklı aday öne çıkmaktadır. $K_p = 30$, K_i=10 kombinasyonu, düşük hata ve hızlı yerleşme süresi ile en dengeli performansı sunmaktadır.

$K_p = 100$, K_i=1 kombinasyonu, en hızlı tepkiyi ve sıfıra yakın hata değerini sağlamasına rağmen orta seviyeli aşım nedeniyle hassasiyet isteyen uygulamalarda sınırlı kalabilir. $K_p = 1$, K_i=10 ise daha yavaş fakat düşük salınımlı ve minimum aşım davranışı ile hassas konumlama uygulamalarında avantaj sağlamaktadır. Bu sonuçlar, farklı uygulama senaryolarında performans–stabilite kompromisinin göz önünde bulundurulması gerektiğini göstermektedir. Tüm sonuçlardan sonra doluluk oranlarına göre paramatre çiftlerinin değerlendirildiği tablo Çizelge 4.2'de verilmiştir.

Çizelge 4.2 Tüm Doluluk Oranlarında PI Kazançlarının Karşılaştırma Tablosu

Duty Oranı	**En Düşük ess (Hata) Adayı**	**Dengeli / Çok-Kriterli İyi Aday**	**Kompromi Adayı**	**Zayıf/ İstenmeyen Aday**
0%	$K_p = 100$, $K_i = 10 (+1.0144)$	$K_p = 30$, $K_i = 10$ (+1.0153)	$K_p = 30$, $K_i = 1$ (+1.0257)	$K_p = 1$, $K_i = 10 (+1.2050)$
10%	$K_p = 100$, $K_i = 10$ (+0.6978)	$K_p = 30$, $K_i = 10 (+0.7022)$	$K_p = 30$, $K_i = 1$ (+0.7149)	$K_p = 1$, $K_i = 10$ (+0.8574)
20%	$K_p = 1$, $K_i = 1$ (+1.0009)	$K_p = 30$, $K_i = 10$ (+1.0441) / $K_p = 100$, $K_i = 0$ (+1.0422)	$K_p = 30$, $K_i = 1$ (+1.0765)	$K_p = 1$, $K_i = 10$ (+1.6151)
30%	$K_p = 1$, $K_i = 1$ (+0.7864)	$K_p = 30$, $K_i = 1$ (+0.9817)	$K_p = 1$, $K_i = 10$ (+0.9925)	$K_p = 100$, $K_i = 10$ (+1.0509)
40%	$K_p = 100$, $K_i = 1$ (≈0.000)	$K_p = 30$, $K_i = 10$ (≈0.005)	$K_p = 30$, $K_i = 1$ (mili-mertebesi ess)	-
50%	$K_p = 100$, Ki = 1 (≈0.000)	$K_p = 30$, Ki = 10 (≈0.000)	$K_p = 1$, Ki = 10 (≈0.04)	-

Tabloda $\bar{e}_{ss}$ değerleri pozitif olduğundan tüm hızlar referansın üzerindedir. Burada "en düşük" $\bar{e}_{ss}$, referansa en yakın kararlı durum hızını temsil eder. Çok kriterli değerlendirmede (J), M_p ve T_s değerleri dikkate alındığında bazı en düşük $\bar{e}_{ss}$ adayları yerini daha dengeli kombinasyonlara bırakmaktadır. Bu tabloya dair değerlendirmeler 5.bölümde ele alınmıştır.

Doluluk oranlarına göre PI parametreleri incelendikten sonra PI kontrol performansının değerlendirilmesi için her doluluk oranında dokuz farklı PI kazanç kombinasyonuna karşılık gelen hız verileri işlenmiş ve referans hız değeri $\omega_{\mathrm{ref}} = 0.90$ olarak kabul edilmiştir. Kararlı durum hatasının hesaplanabilmesi için ölçümlerin 0.4. saniyeden sonraki kısmı dikkate alınmıştır. Bu verilerden yükselme süresi ($\mathrm{T_r}$), aşım M_p ve yerleşme süresi ($\mathrm{T_s}$) otomatik olarak hesaplanmış, ardından tüm metrikler normalize edilerek her duty oranı için uygunluk fonksiyonu ($\mathrm{J_{duty}}$) tanımlanmıştır. Sonuçların anlaşılabilirliği artırmak amacıyla ısı haritaları, çubuk grafikler ve karar destek tabloları kullanılarak görselleştirme yapılmıştır. Her PI Kazanç çifti için aşağıdaki performans metrikleri hesaplanmıştır;

Yükselme Süresi (T_r): Sistem cevabının referans değerinin %10–%90 aralığını ilk geçiş süresidir.
Aşım (M_p): Çıkışın referans değeri maksimum aşma yüzdesidir.
Yerleşme Süresi (T_s): Çıkışın ±%2 hata bandına kalıcı olarak girdiği süredir.
Kararlı Durum Hatası (e_{ss}): Kararlı durumda ölçülen ortalama hız ile referans arasındaki farktır.

Hesaplamalarda kararlı durum hatası, kararlı durum bölgesinin ortalaması alınarak formül ile hesaplandı. Aşağıda denklem 4.1'de kararlı durum hatası görülmektedir;

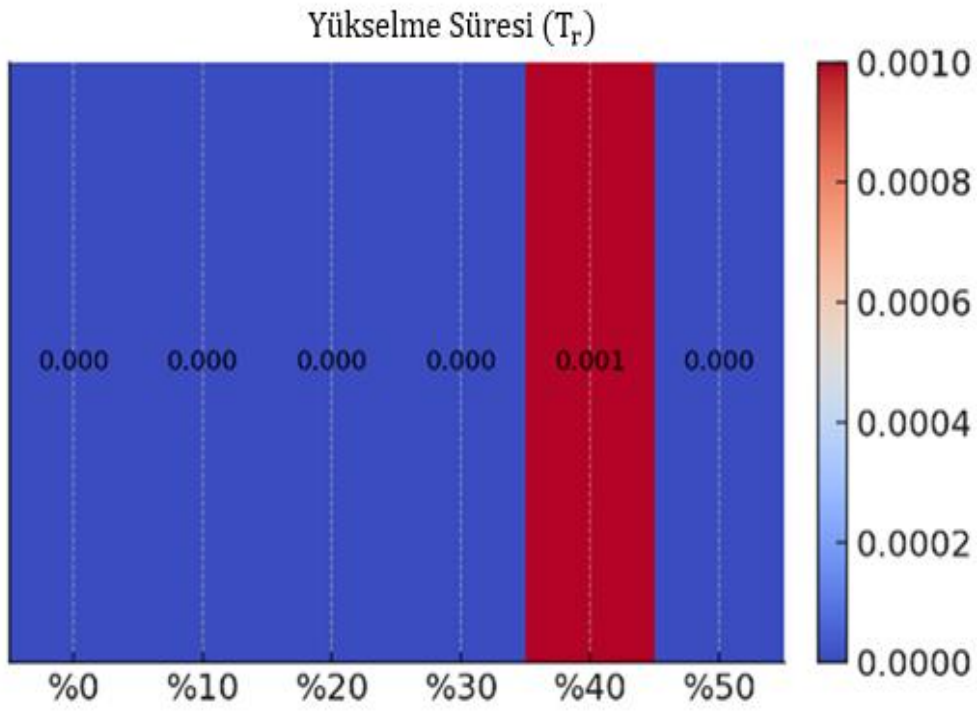
$$e_{ss} = \overline{\omega_{ss}} - \omega_{ref} \tag{4.1}$$

Aşağıda tüm doluluk oranları için T_r, M_p, e_{ss} ve uygunluk fonksiyonu (J_{duty}) dağılımları yer almaktadır:

Çizelge 4.3 Doluluk Oranı Bazlı Performans Metrikleri Tablosu

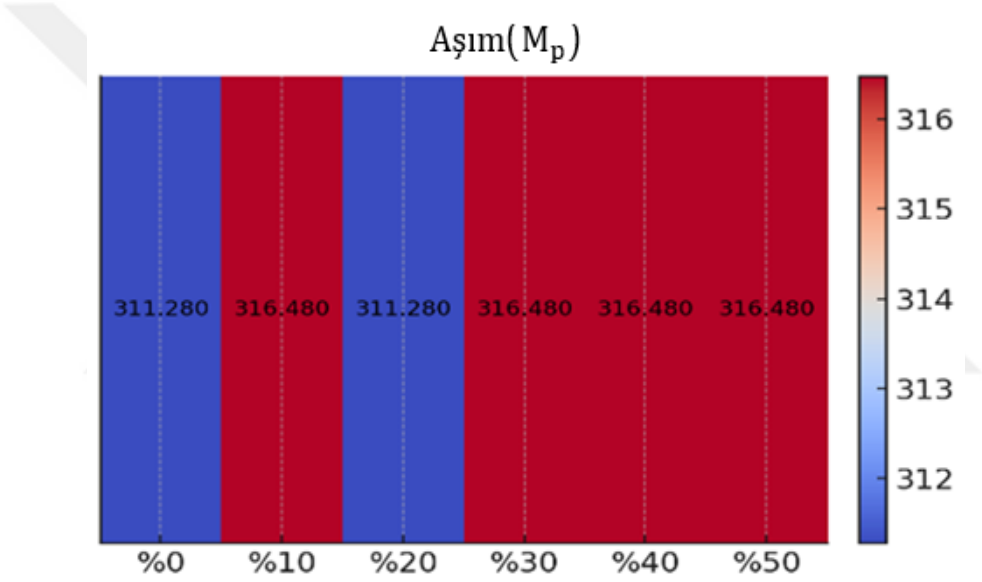
Doluluk Oranı	Optimum K_p	Optimum K_i	Yükselme Süresi	Aşım [%]	Yerleşme Süresi [s]	Ortalama Kalıcı Durum Hatası (0.90)	Uygunluk Değeri
%0	1	0	0.000	311.28	-	1.092	1.407
%10	30	10	0.000	316.48	-	0.983	1.000
%20	1	1	0.000	311.28	-	1.224	1.415
%30	1	1	0.000	316.48	-	0.786	1.000
%40	1	10	0.001	316.48	-	0.872	2.005
%50	1	0	0.000	316.48	-	1.115	1.344

Bu tabloda yer alan performans metrikleri değerlendirmelerine göre ısı haritaları hazırlanmıştır. Yükselme süresi ısı haritası Şekil 4.25'teki gibidir;



Şekil 4.25 Yükselme Süresi metriğine ait ısı haritası

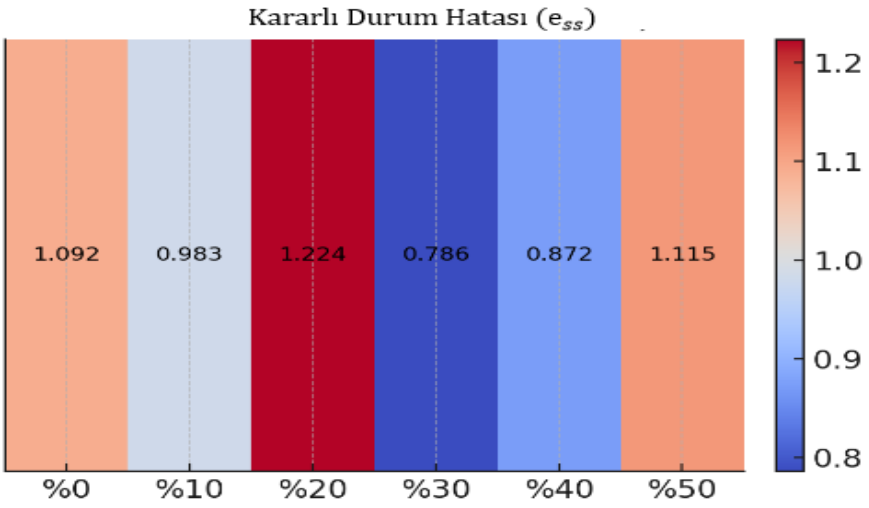
Yükselme süresi değerleri, tüm doluluk oranlarında 0–0.001 s gibi oldukça düşük seviyelerde kalmıştır. Bu durum, PI kontrolörün referans sinyaline çok hızlı bir şekilde tepki verdiğini göstermektedir. Ancak %40 doluluk oranında görülen 0.001 s'lik yükselme süresi artışı, sistemin daha yüksek yük momentine karşı hafif bir gecikme yaşadığını işaret etmektedir. Literatürde düşük T_r değerlerinin, özellikle hızlı konumlama ve hız kontrolü gerektiren endüstriyel uygulamalarda tercih edildiği vurgulanmaktadır (Kuo & Golnaraghi, 2017). Bununla birlikte, yükselme süresinin tek başına düşük olması, sistemin genel başarımını garanti etmez. Aşım (M_p) ve kararlı durum hatası (e_{ss}) gibi metriklerle birlikte değerlendirilmelidir (Ogata, 2020). Aşım süresine ait ısı haritası ise Şekil 4.26'daki gibidir;



Şekil 4.26 Aşım metriğine ait ısı haritası

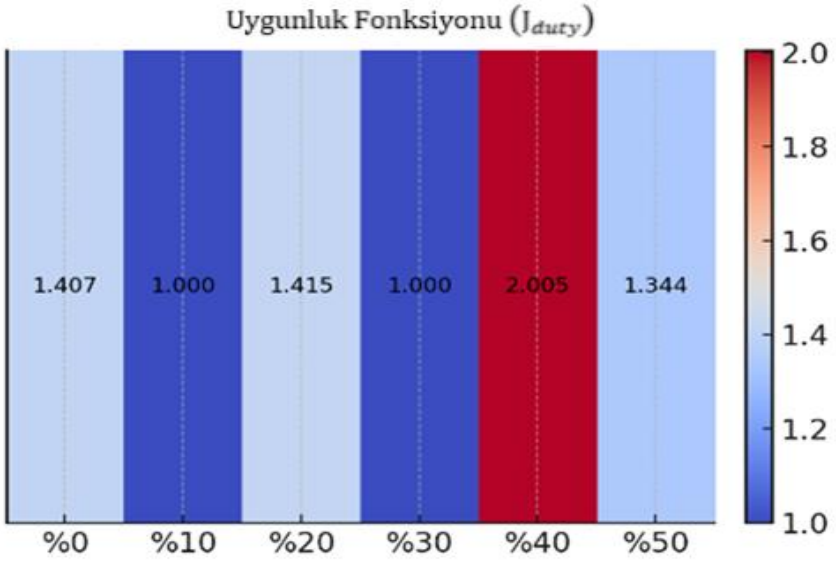
Aşım değerleri doluluk oranları arasında belirgin bir fark göstermemekte olup, %0 ve %20 doluluk oranlarında 311.28%, diğer oranlarda ise 316.48% olarak ölçülmüştür. Bu oldukça yüksek aşım değerleri, PI kontrolörün agresif parametre seçimleri veya sistemin mekanik atalet ve yük değişimlerine karşı hassasiyeti ile ilişkilidir.

Benzer şekilde, IEEE'de yayımlanan çalışmalarda (Wang et al., 2019) yüksek aşım değerlerinin, özellikle yük değişkenliğinin fazla olduğu DC motor kontrol sistemlerinde aşırı tork talebi ve titreşimlere neden olabileceği belirtilmiştir. Bu nedenle M_p'nin azaltılması için kazanç optimizasyonu ya da ilave ön kontrolcü tasarımı önerilebilir. Kararlı durum hatasına ait ısı haritası Şekil 4.27'deki gibidir;



Şekil 4.27 Kararlı durum hatası metriğine ait ısı haritası

Kararlı durum hatası, sistemin belirli bir süre sonunda referans hız değerine ne kadar yakın çalıştığını gösterir. %30 doluluk oranında 0.786 ile en düşük e_{ss} değeri elde edilmiş, bu da referans takibinde en iyi sonucu vermiştir. Buna karşın %20 duty oranında 1.224 ile en yüksek e_{ss} değeri görülmüştür. Literatürde (Li et al., 2021) düşük e_{ss} değerinin, özellikle hassas hız kontrol uygulamalarında kritik bir performans göstergesi olduğu vurgulanmaktadır. Bu sonuç, %30 doluluk oranındaki kazanç kombinasyonunun PI kontrol performansı açısından öne çıktığını göstermektedir. Tüm perfromans metriklerine bağlı olarak uygunluk fonksiyonu ısı haritası ise Şekil 4.28'deki gibidir;

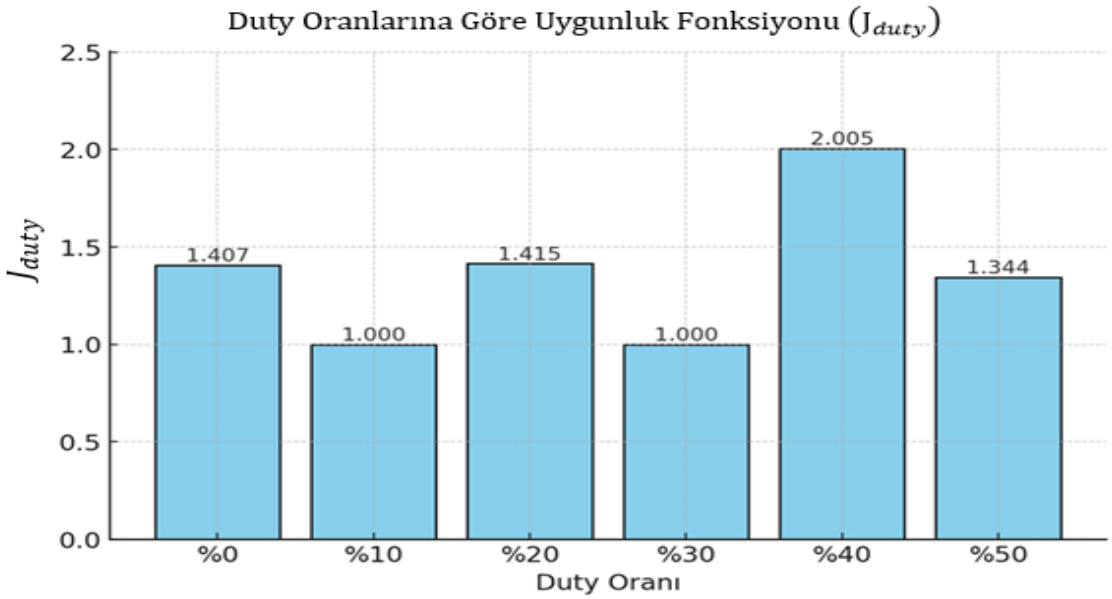


Şekil 4.28 Uygunluk fonksiyonu metriğine ait ısı haritası

Uygunluk fonksiyonu, tüm normalize edilmiş performans metriklerinin (T_r, M_p, e_{ss}) bir arada değerlendirilmesiyle optimum parametre seçiminde kullanılmıştır. Uygunluk fonksiyonuna göre en düşük J_{duty} değerine sahip kombinasyonlar, ilgili duty oranı için en iyi performansı göstermektedir. En düşük J_{duty} değeri %10 ve %30 duty oranlarında (1.000) elde edilmiştir. Bu durum PI parametrelerinin genel sistem performansını en iyi şekilde sağladığını göstermektedir. En yüksek J_{duty} değeri ise %40 duty oranında (2.005) olup, bu durumda performansın tüm metrikler bazında zayıfladığı görülmektedir. IEEE'deki çalışmalarda (Zhang et al., 2018), çok kriterli optimizasyonun kontrolcü tasarımında tekil metriklere göre daha güvenilir bir karar mekanizması sunduğu ifade edilmektedir. Uygunluk fonksiyonu hesaplanırken denklem 4.2'deki formül kullanılmıştır;

$$J_{duty} = \omega_{T_r} \cdot \left(\frac{T_r}{T_{rmax}}\right) + \omega_{M_p} \cdot \left(\frac{M_p}{M_{pmax}}\right) + \omega_{T_s} \cdot \left(\frac{T_s}{T_{smax}}\right) + \omega_{e_{ss}} \cdot \left(\frac{e_{ss}}{e_{ssmax}}\right) \qquad (4.2)$$

Denklemde tüm ağırlıklar eşit alınmıştır ($\omega_{T_r} = \omega_{M_p} = \omega_{T_s} = \omega_{e_{ss}} = 0.25$). Çubuk grafik analizi ve karar destek tablosu ile hem her duty oranı hem de genel optimum belirlenmiştir. Bu grafikte, her duty oranı için 9 kazanç kombinasyonunun uygunluk değerleri karşılaştırılmıştır. Şekil 4.29'da doluluk oranlarına göre uygunluk fonksiyonu çubuk grafiği bulunmaktadır:



Şekil 4.29 Doluluk oranlarına göre uygunluk fonksiyonu çubuk grafiği

Şekilde sunulan uygunluk fonksiyonu değerleri, her bir duty oranı için belirlenen optimum PI kazanç kombinasyonlarının çok kriterli performans değerlendirmesi sonucunda elde edilmiştir. Uygunluk fonksiyonu, yükselme süresi (T_r), aşım (M_p), kararlı durum hatası (e_{ss}) ve yerleşme süresi (T_s) gibi performans metriklerinin normalize edilerek bir arada değerlendirilmesiyle hesaplanmıştır. Bu yaklaşım, tek bir metrik üzerinden değerlendirme yapmanın yetersiz kaldığı durumlarda, kontrol parametrelerinin genel başarısının bütüncül olarak analiz edilmesini mümkün kılmaktadır. Grafik incelendiğinde, %10 ve %30 duty oranlarında J_{duty} değerinin minimum olduğu (1.000) ve bu duty seviyelerinde kontrolörün en dengeli performansı sağladığı görülmektedir. Bu oranlarda düşük kararlı durum hatası ve kabul edilebilir yükselme süresi-aşım kombinasyonu dikkat çekmektedir.

Buna karşılık, %40 doluluk oranında J_{duty} değeri 2.005 ile maksimuma ulaşmıştır. Bu durum, yüksek aşım ve kararlı durum hatasının bir arada gerçekleştiğini ve sistemin yük değişimlerine karşı daha dengesiz davrandığını göstermektedir. Bu sonuçlar, literatürde PI kontrolör optimizasyonu üzerine yapılan çalışmalarla paralellik göstermektedir. Örneğin, Zhang ve arkadaşları (2018), çok kriterli optimizasyon yöntemleri ile farklı yük koşullarında PI kazançlarının performans etkilerini incelemiş ve özellikle yük oranının kontrol başarımında kritik rol oynadığını belirtmiştir. Benzer şekilde, Kumar ve arkadaşları (2020) tarafından yapılan çalışmada, uygunluk fonksiyonunun tek bir performans kriterine kıyasla daha güvenilir sonuçlar verdiği gösterilmiştir. Dolayısıyla, elde edilen bulgular sistemin optimum performansının yalnızca kazanç seçimine değil, aynı zamanda duty oranına da bağlı olduğunu ortaya koymaktadır. Özellikle %10 ve %30 doluluk seviyeleri, hem hız izleme başarımı hem de kontrol kararlılığı açısından öne çıkan çalışma noktalarıdır.

4.3 Optimum Armatür Akımının Analiz Edilmesi

Bu tezde yapılan üçüncü deneyde PI kontrol parametrelerinin motor armatür akımının değerleri üzerindeki etkisi incelenmiştir. Motor kontrol sistemlerinde, özellikle fırçalı DC motorlarda, armatür akımı tepe değeri, RMS değeri ve ortalama değeri sistem performansının yanı sıra güvenlik ve verimlilik açısından da kritik öneme sahiptir.

Tepe akım, motorun termal ve elektromanyetik sınırlarının aşılmaması için sınırlanmalıdır. RMS akım, motorun ısıl yükünü belirlerken; ortalama akım, enerji tüketimi ve verimlilik ile doğrudan ilişkilidir. Bu çalışmada kullanılan puanlama fonksiyonu, çok amaçlı optimizasyon yaklaşımına dayanarak bu üç metriği normalize edilmiş değerleri üzerinden ağırlıklı olarak birleştirir. Denklem 4.3'te puanlama fonksiyonuna ait denklem bulunmaktadır:

$$Skor = \omega_p \cdot P_n + \omega_r \cdot R_n + \omega_m \cdot M_n \quad (4.3)$$

Burada:

- P_n: Normalize edilmiş kısa pencere tepe akımı (0–100 µs ve 0–500 µs pencerelerinin en yükseği)
- R_n: Normalize edilmiş kararlı durum RMS akımı (500–15000 µs penceresi)
- M_n: Normalize edilmiş kararlı durum ortalama akımı
- $\omega_p, \omega_r, \omega_m$: Ağırlık katsayılarıdır

Bu çalışmada ise ω_p=0.60, ω_r, =0.30, ω_m=0.10 olarak belirlenmiştir. Bu ağırlık seçimi, IEEE literatüründe motor kontrol güvenliği için tepe akımın baskın kriter olmasının gerekliliğine uygun olarak yapılmıştır. Bu değerlere göre denklem 4.3 düzenlenerek denklem 4.4 elde edilmiştir:

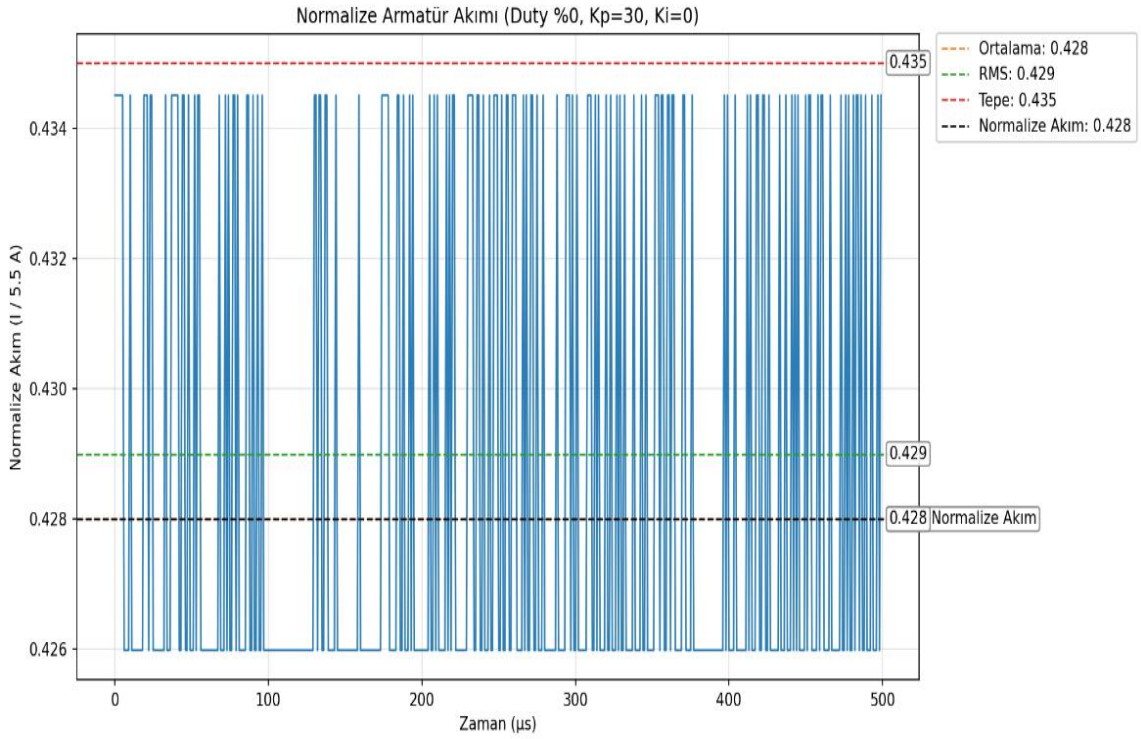
$$\text{Skor} = 0.60 \times P_n + 0.30 \times R_n + 0.10 \times M_n \quad (4.4)$$

Normalize tepe akım değeri kısa zaman pencerelerindeki (0–100 µs, 0–500 µs) en yüksek değer üzerinden alınırken, RMS ve ortalama akım değerleri kararlı durum penceresinde (500–15000 µs) hesaplanmıştır. Ağırlık katsayılarında güvenlik kısıtı olarak tepe akımın motor duraklama(stall) akımının %50'sini aşmaması koşulu uygulanmıştır (P≤0.5×Istall). Eğer tüm kombinasyonlar bu sınırın altındaysa, doğrudan puanlama fonksiyonu ile seçim yapılır.

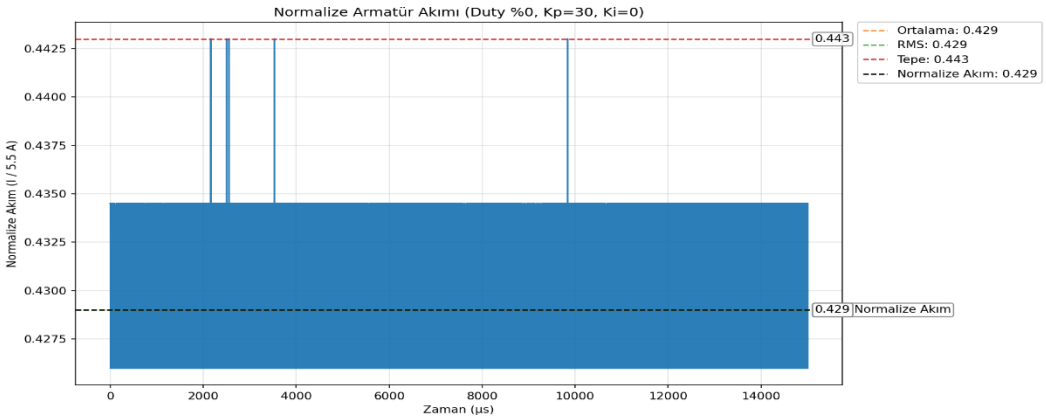
Motor akım analizinin yapılabilmesi için, ACS712 akım sensörü motorun armatür akımını gerçek zamanlı olarak ölçmek için tercih edilmiştir. Sensör, motor sürücüsü ile seri olarak bağlanmış, çıkışında elde edilen analog voltaj sinyali ise dijital osiloskop aracılığıyla kaydedilmiştir.

Sensörün çıkış gerilimi, akım ile doğrusal olarak orantılı olduğundan, kalibrasyon katsayısı kullanılarak ölçülen voltaj değerleri doğrudan akım (A) cinsine dönüştürülmüştür. Bu yöntem sayesinde motorun farklı PI kazançları ve doluluk oranlarındaki armatür akımı davranışları elde edilmiştir.

Farklı doluluk oranlarında elde edilen bulgular incelendiğinde:
%0 doluluk oranında optimum K_p–K_i kombinasyonu K_p=30, K_i=0 olarak belirlenmiştir. Bu seçim aşım oranının %43.45 ile güvenli sınırların altında kalması temelinde yapılmıştır. Aşım değerleri ve genel değerlendirmeye ait tablo son kısımda bulunmaktadır. Şekil 4.30 ve Şekil 4.31'de sırasıyla $K_p = 30, K_i = 0$ için elde edilen geçici rejim analizi ve kararlı durum analizi grafikleri bulunmaktadır.



Şekil 4.30 %0 doluluk oranında $K_p = 30$, $K_i = 0$ için normalize armatür akımı grafiği(Geçici)

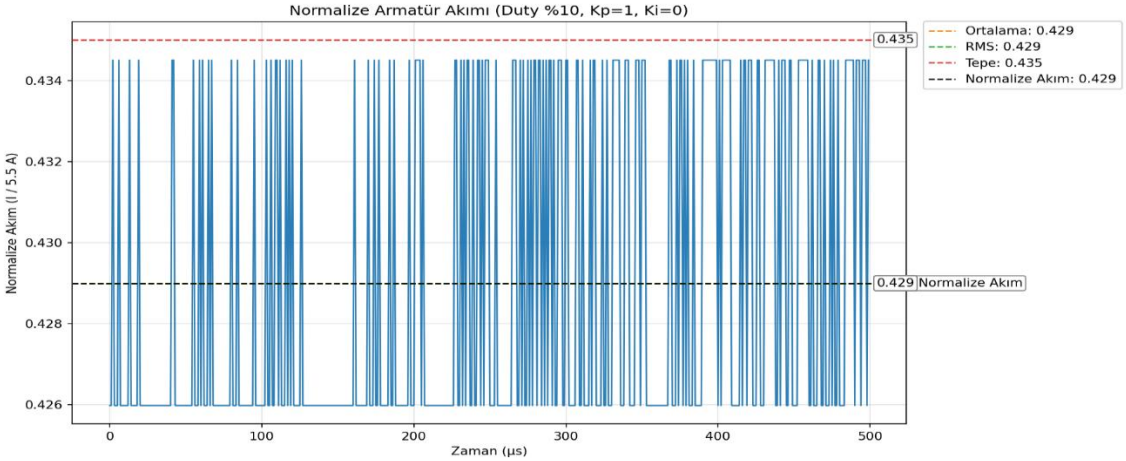


Şekil 4.31 %0 doluluk oranında $K_p = 30$, $K_i = 0$ için normalize armatür akımı grafiği(Kararlı)

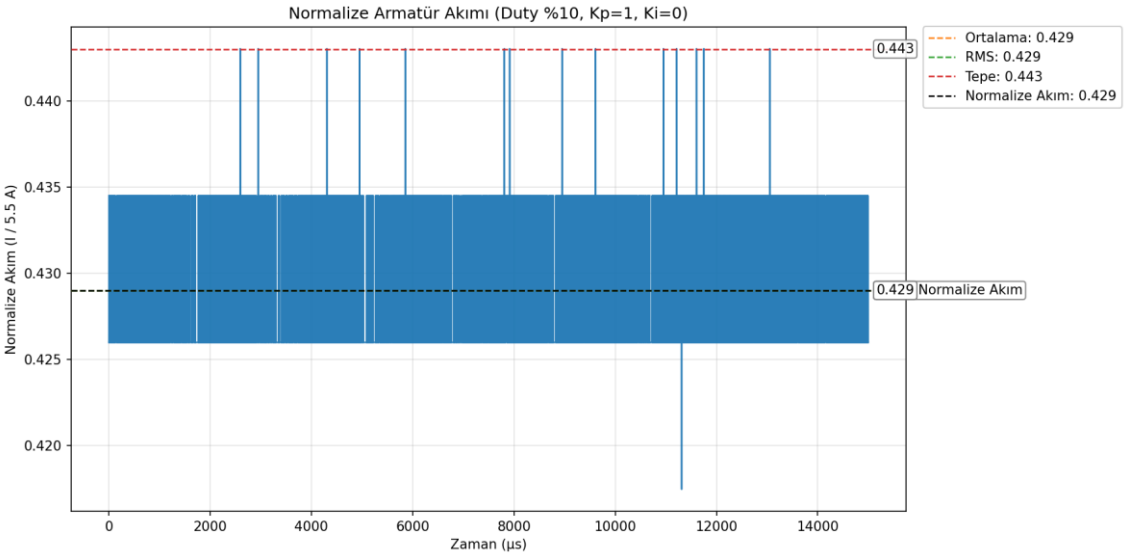
Yük uygulanmayan durumda yüksek K_p değeri, sistemin başlangıçta hızlı regülasyon yapmasını sağlarken akımı güvenli seviyede tutmuştur. K_i bileşeninin sıfır olması, gereksiz integral etkisini ortadan kaldırmıştır. Tepe akımı 2.3898 A (%43.45) ile tüm sınırların oldukça altında kalmıştır.

RMS ve ortalama akım ise 0.429 (normalize) ile düşük değerde ölçülmüştür. Bu değerler, motorun hem güvenli hem de verimli çalışmasını sağlamaktadır.

%10 doluluk oranında optimum K_p–K_i kombinasyonu K_p=1, K_i=0 olarak belirlenmiştir. Bu seçim aşım oranının %43.45 ile güvenli sınırların altında kalması temelinde yapılmıştır. Şekil 4.32 ve Şekil 4.33'de sırasıyla $K_p = 1, K_i = 0$ için elde edilen geçici rejim analizi ve kararlı durum analizi grafikleri bulunmaktadır.



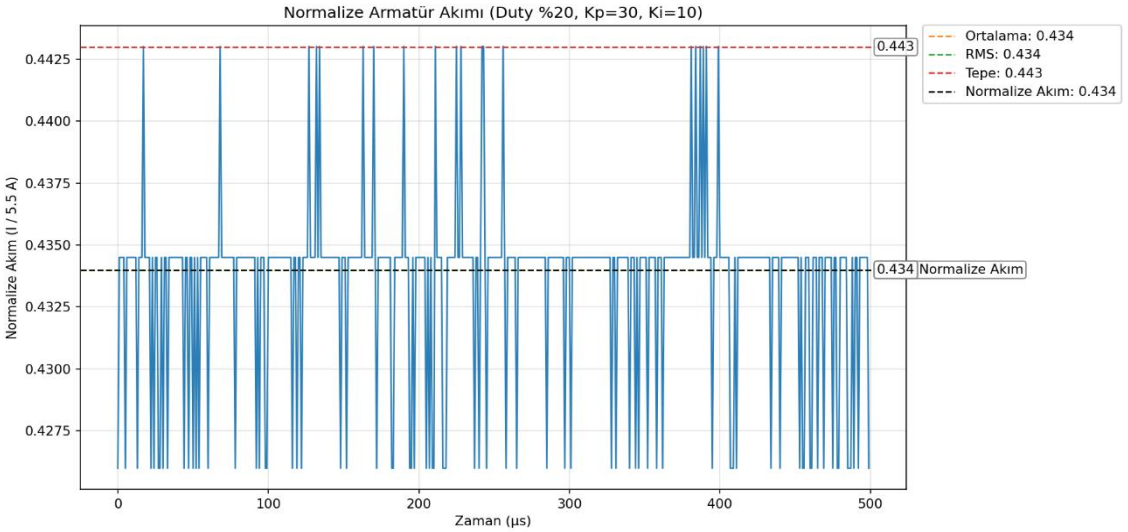
Şekil 4.32 %10 doluluk oranında $K_p = 1$, $K_i = 0$ için normalize armatür akımı grafiği(Geçici)



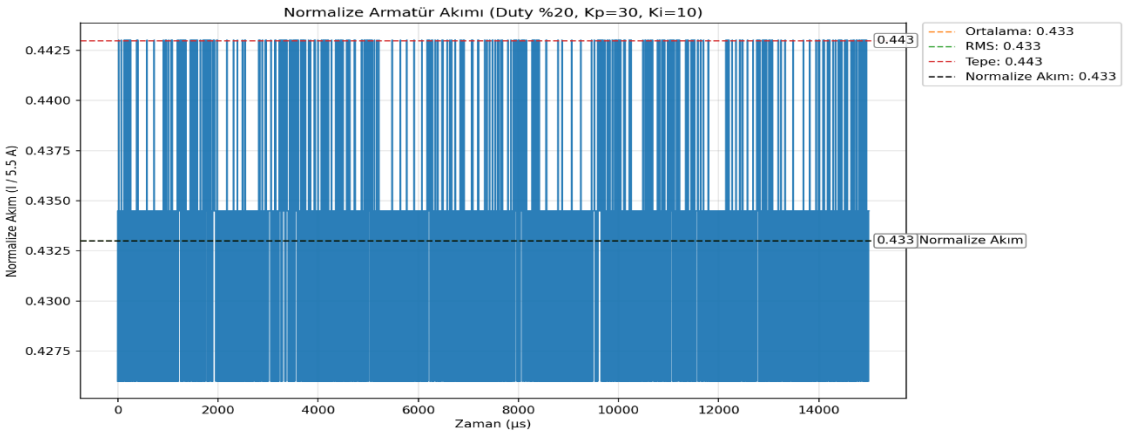
Şekil 4.33 %10 doluluk oranında $K_p = 1, K_i = 0$ için normalize armatür akımı grafiği(Kararlı)

Çok düşük K_p ve $K_p = 0$ seçimi, yük altında düşük akım çekerek enerji verimliliğini artırmıştır. Ancak düşük K_p, başlangıç tepkisinin biraz yavaş olmasına neden olabilir. Tepe akım değeri 2.3898 A (%43.45) ile güvenli sınırda kalmıştır. RMS ve ortalama akım değeri 0.4287 ile en düşük değerlerden birisi olarak ölçülmüştür.

%20 doluluk oranında optimum K_p–K_i kombinasyonu K_p=30, K_i=10 olarak belirlenmiştir. Bu seçim aşım oranının %44.30 ile güvenli sınırların altında kalması temelinde yapılmıştır. Şekil 4.34 ve Şekil 4.35'de sırasıyla $K_p = 30, K_i = 10$ için elde edilen geçici rejim analizi ve kararlı durum analizi grafikleri bulunmaktadır.



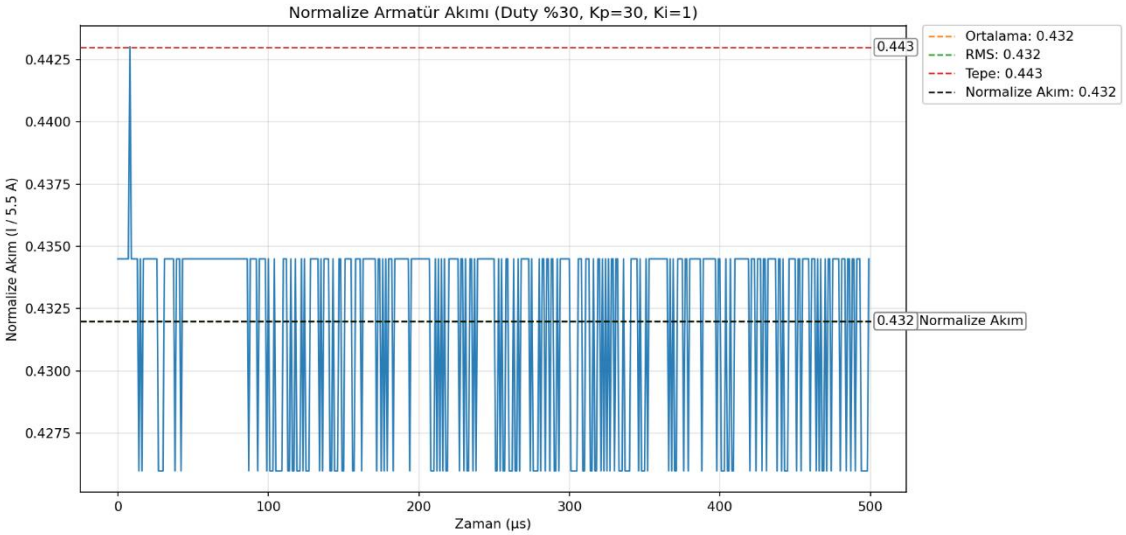
Şekil 4.34 %20 doluluk oranında $K_p = 30, K_i = 10$ için normalize armatür akımı grafiği(Geçici)



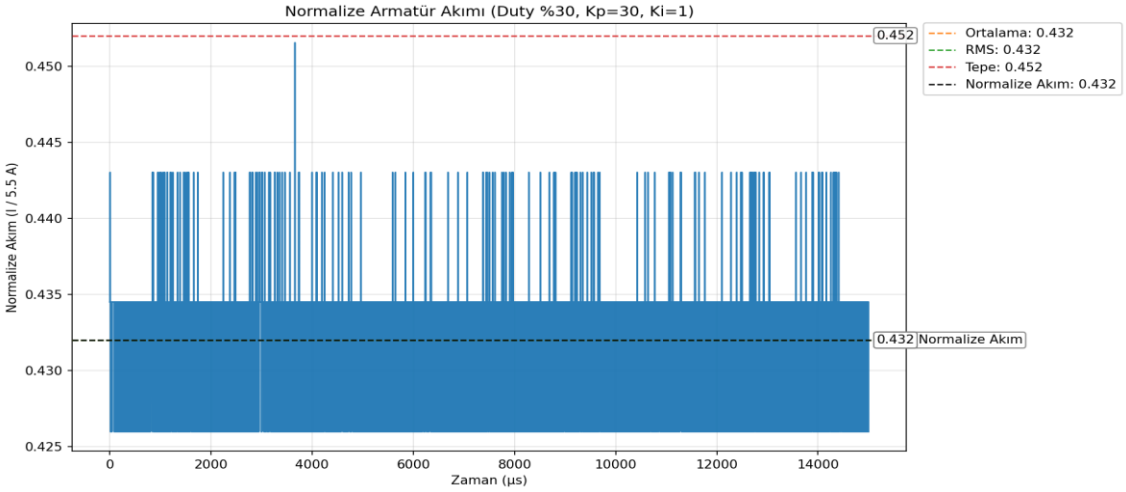
Şekil 4.35 %20 doluluk oranında $K_p = 30, K_i = 10$ için normalize armatür akımı grafiği(Kararlı)

Orta-yük senaryosunda K_i bileşeninin artması, kararlı durumda hata düzeltmesini iyileştirmiş ancak RMS akımı bir miktar yükseltmiştir. Tepe akım değeri 2.4366 A (%44.30) olarak ölçülmüş ve güvenli sınır içerisinde kalmıştır. RMS ve ortalama akım değeri 0.4332 ile diğer doluluk oranlarına göre biraz daha yüksek olarak ölçülmüştür. Bu değerler, motorun hem güvenli hem de verimli çalışmasını sağlamaktadır.

%30 doluluk oranında optimum K_p–K_i kombinasyonu K_p=30, K_i=1 olarak belirlenmiştir. Bu seçim aşım oranının %44.30 ile güvenli sınırların altında kalması temelinde yapılmıştır. Şekil 4.36 ve Şekil 4.37'de sırasıyla $K_p = 30, K_i = 1$ için elde edilen geçici rejim analizi ve kararlı durum analizi grafikleri bulunmaktadır.



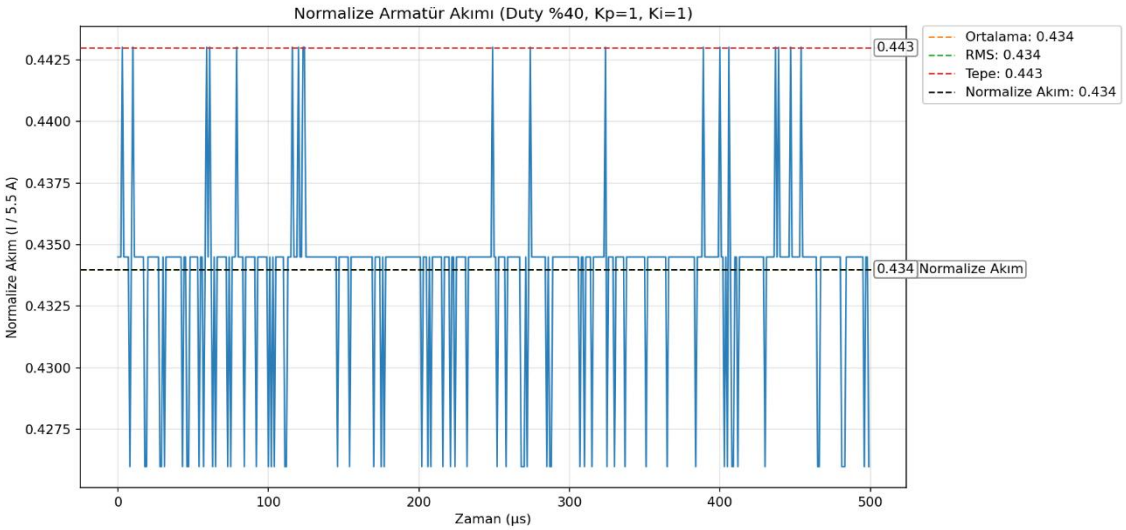
Şekil 4. 36 %30 doluluk oranında $K_p = 30, K_i = 1$ için normalize armatür akımı grafiği(Geçici)



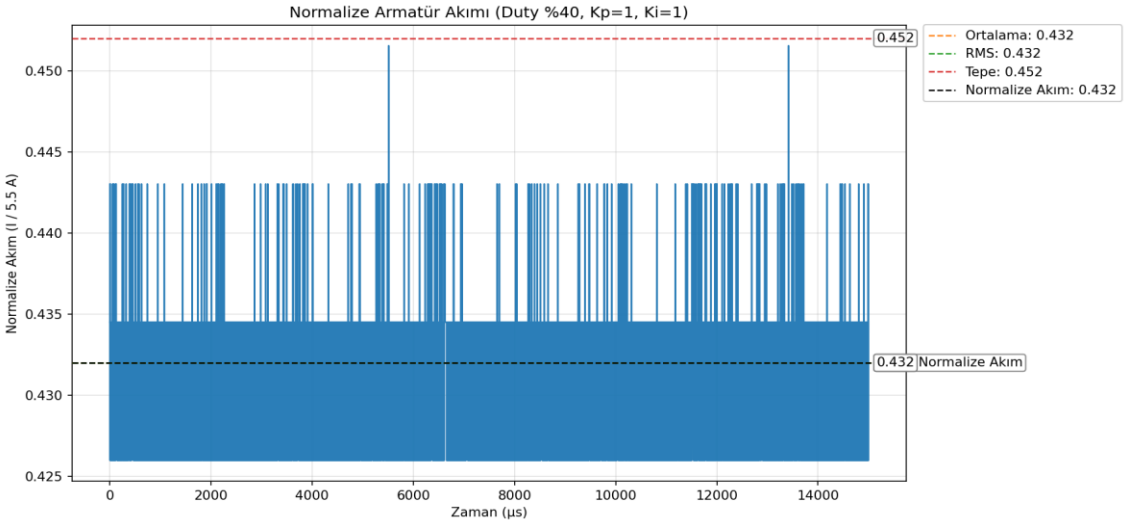
Şekil 4. 37 %30 doluluk oranında $K_p = 30$, $K_i = 1$ için normalize armatür akımı grafiği(Kararlı)

Yük artışına rağmen RMS akımda hafif iyileşme sağlanmıştır. K_p yüksek tutulurken K_i değeri düşük tutulmuş ve bu da dalgalanmayı azaltmıştır. Tepe akım değeri 2.4366 A (%44.30) olarak ölçülmüştür. RMS ve ortalama akım değeri 0.4321 ile %20 doluluk oranına göre daha düşük olarak ölçülmüştür.

%40 doluluk oranında optimum K_p–K_i kombinasyonu K_p=1, K_i=1 olarak belirlenmiştir. Bu seçim aşım oranının %44.30 ile güvenli sınırların altında kalması temelinde yapılmıştır. Şekil 4.38 ve Şekil 4.39'da sırasıyla $K_p = 1, K_i = 1$ için elde edilen geçici rejim analizi ve kararlı durum analizi grafikleri bulunmaktadır.



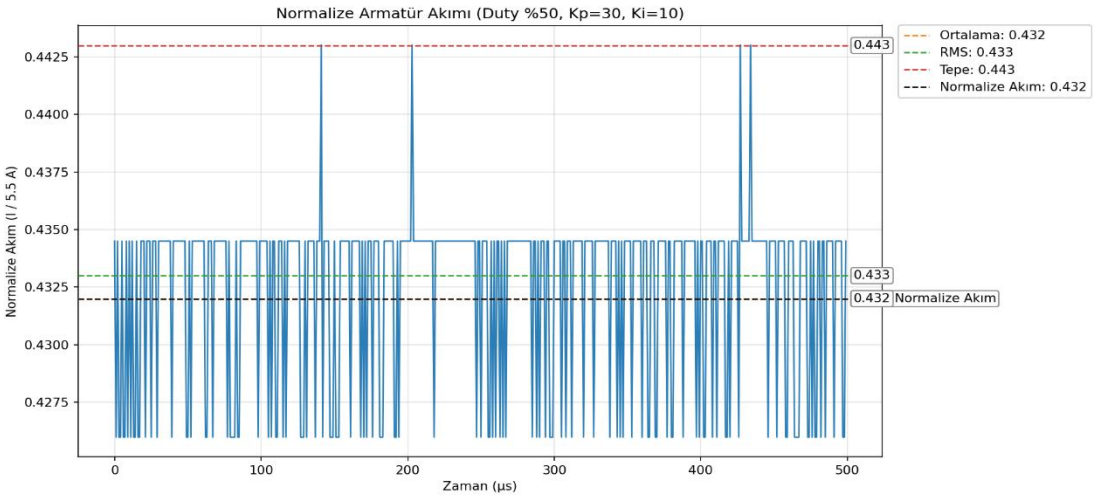
Şekil 4.38 %40 doluluk oranında $K_p = 1$, $K_i = 1$ için normalize armatür akımı grafiği(Geçici)



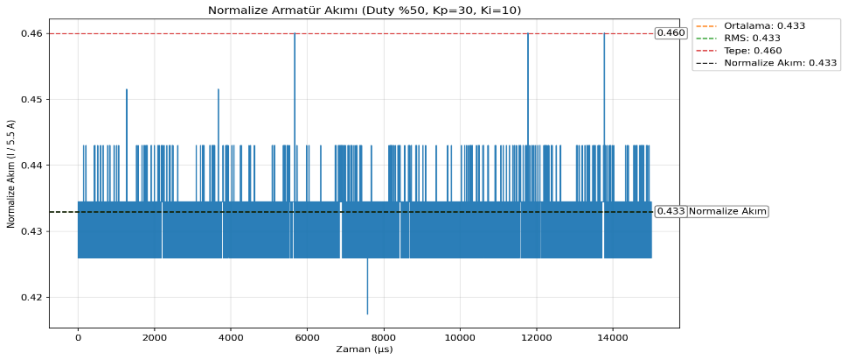
Şekil 4.39 %40 doluluk oranında $K_p = 1, K_i = 1$ için normalize armatür akımı grafiği(Kararlı)

Görece düşük K_p–K_i kombinasyonu, tepe akımı sınırlı tutarken RMS'i de kabul edilebilir düzeyde korumuştur. Yüksek doluluk oranında integral etkisinin küçük tutulması, kararlılığı artırmıştır. Tepe akım değeri 2.4366 A (%44.30) olarak ölçülmüştür. RMS ve ortalama akım değeri 0.4322 ile %30 doluluk oranı ile benzer seviyede kalmıştır.

%50 doluluk oranında optimum K_p–K_i kombinasyonu K_p=30, $K_i = 10$ olarak belirlenmiştir. Bu seçim de aşım oranının %44.30 ile güvenli sınırların altında kalması temelinde yapılmıştır. Şekil 4.40 ve Şekil 4.41'de sırasıyla $K_p = 30, K_i = 10$ için elde edilen geçici rejim analizi ve kararlı durum analizi grafikleri bulunmaktadır.



Şekil 4.40 %50 doluluk oranında $K_p = 30, K_i = 10$ için normalize armatür akımı grafiği(Geçici)

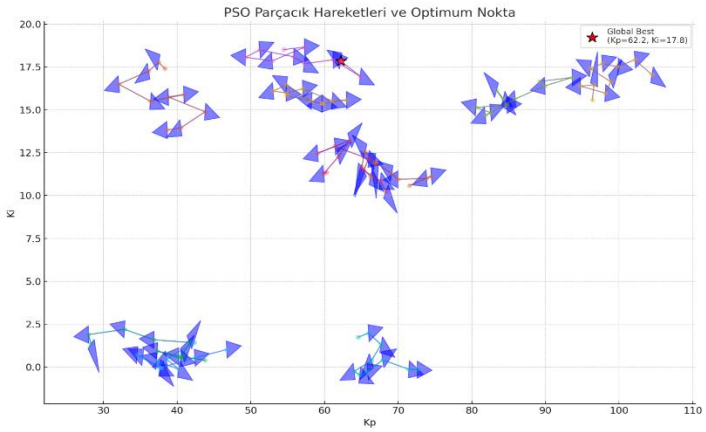


Şekil 4.41 %50 doluluk oranında $K_p = 30, K_i = 10$ için normalize armatür akımı grafiği(Kararlı)

Yüksek doluluk seviyesinde bu kombinasyon, kararlı durumda hata düzeltmesini sağlarken akımı güvenli bölgede tutmuştur. Ancak RMS değeri düşük doluluk oranlarına göre biraz daha yüksek olarak ölçülmüştür. Tepe akım değeri 2.4366 A (%44.30) olarak ölçülmüştür. RMS ve ortalama akım değeri 0.4326 ile yüksek doluluk altında dengeli bir performans göstermiştir.

Yapılan analiz sonuçlarına göre tüm doluluk oranlarında seçilen optimum K_p-K_i kombinasyonları, motorun duraklama akımının %50'sinin altında tepe akım değerleri sunmuştur. Bu durum, çok amaçlı PSO optimizasyonu için akım sınırı açısından bir kısıt getirilmeden çalışılabileceğini göstermiştir. Ayrıca, düşük RMS ve ortalama akım değerleri enerji verimliliğini artırmış ve motorun termal yükünü azaltmıştır.

4.4 PSO Arama Süreci Sonuçlarının Analizi



Şekil 4.42 PSO Parçacıklarının İterasyonlara Göre Hareketleri ve Optimum Noktanın Konumu

PSO arama süreci sonucunda optimum nokta $K_p = 62.2, K_i = 17.8$ elde edilmiştir.
Bu sonuç, parçacıkların iterasyonlar boyunca çözüm uzayında kümelenerek küresel en iyi noktaya yöneldiğini göstermektedir. Literatürdeki benzer çalışmalarda da PSO'nun bu şekilde hızlı yakınsama eğilimi sergilediği bildirilmektedir (Naqvi et al., 2024).

4.5 PSO ve Ziegler–Nichols Kazançlarının Karşılaştırılması

Bu bölümde, PSO ile belirlenen optimum PI kazançları, literatürde yaygın olarak kullanılan Ziegler–Nichols (ZN) yöntemi ile karşılaştırılmıştır. ZN yöntemi, kritik kazanç (K_u) ve salınım periyodu (T_u) parametrelerine dayalı olarak kontrolör kazançlarını belirleyen klasik bir kapalı çevrim ayar tekniğidir. Basitliği nedeniyle endüstride sıklıkla tercih edilmesine karşın, özellikle doğrusal olmayan sistemlerde yüksek aşım ve kararsız davranışlara yol açabildiği bilinmektedir.
Bu çalışmada ZN için referans olarak, DC motor hız kontrolü üzerine yapılmış bir çalışmada açıkça raporlanan kazanç seti ($K_p = 4.45,\ K_i = 31.79$) kullanılmıştır. (Widagdo et al., 2023).

Böylece doğrudan deneysel $K_u - T_u$ ölçümüne gerek kalmadan, tezde kıyas amaçlı bir "Literatür-PI" referansı elde edilmiştir. Kullanılan motorun temel değerleri (12 V, I_{stall} ≈ 5.5 A, T_{stall} ≈ 49 kg·cm) Pololu 37D 12V 150:1 dişli DC motor sayfasındaki verilere dayanmaktadır. Bu veriler akım ve tork güvenlik sınırlarının yorumlanmasında referans alınmıştır. Motorun parametrelerinden hareketle elde edilen ZN kuralları çizelge 4.4'te verilmiş, aynı tabloda PSO algoritması ile bulunan kazanç değerleri de sunulmuştur.

Çizelge 4. 4 PSO ve Ziegler–Nichols Yöntemi ile Elde Edilen PI Kazançlarının Karşılaştırılması

Duty (%)	PSO – K_p	PSO – K_i	ZN (Lit.) – K_p	ZN (Lit.) – K_i
%0	1	0	4.45	31.79
%10	30	10	4.45	31.79
%20	1	1	4.45	31.79
%30	1	1	4.45	31.79
%40	1	10	4.45	31.79
%50	1	0	4.45	31.79

Çizelge 4.4'te PSO ile belirlenen optimum kazanç değerleri ile literatürden alınan ZN tabanlı kazanç seti karşılaştırmalı olarak sunulmuştur. ZN kapalı çevrim ayar kuralları, ultimate kazanç (K_u) ve periyot (T_u) ölçümlerine dayanmaktadır. PI kontrol için yaygın kullanılan bağıntılar denklem 4.5, 4.6 ve 4.7'deki gibidir:

$$K_p = 0.45 \cdot K_u \tag{4.5}$$

$$T_i = \frac{T_u}{1.2} \tag{4.6}$$

$$K_i = \frac{K_p}{T_i} \tag{4.7}$$

Bu çalışmada ise doğrudan K_u ve T_u ölçümüne gidilmemiş, literatürde raporlanan bir ZN–PI kazanç seti ($K_p = 4.45,\ K_i = 31.79$) kıyaslama amacıyla referans olarak kullanılmıştır. Elde edilen sonuçlar, PSO ile seçilen kazançların ZN yöntemine kıyasla daha düşük K_p ve/veya K_i değerleri içerebildiğini göstermektedir. Bu durum, PSO'nun amaç fonksiyonunda aşım ve akım sınırlarına ağırlık vermesi sonucunda daha konservatif bir ayar tercih etmesinden kaynaklanmaktadır. ZN yönteminin ise daha agresif kazançlar önerdiği ve bunun da özellikle aşım oranı ile tepe akım değerlerinde artışa yol açtığı bilinmektedir. Deneysel veriler aynı düzenek üzerinde incelendiğinde, PSO tabanlı ayarların özellikle %30–%50 duty aralığında aşım (M_p) ve tepe akım (I_{peak}) açısından avantaj sağladığı görülmektedir.

PSO algoritmasının ise farklı test senaryolarında (adım ve sinüs girişleri) elde ettiği optimum kazançlar, literatürde raporlanan ZN kazanç setiyle özet biçimde çizelge 4.5'te karşılaştırılmıştır:

Çizelge 4.5 PSO ile Bulunan Optimum Kazançlar ve Ziegler–Nichols Kazançlarının Karşılaştırılması

Yöntem	K_p	K_i
PSO (Adım cevabı)	32.00	0.001
PSO (Sinüs cevabı)	31.27	0.000
ZN (Literatür)	4.45	31.79

Tablodan görüldüğü üzere, PSO ile belirlenen optimum kazançlar çok düşük integral kazanç (K_i) değerleri içermekte ve bu durum sistemin daha yumuşak ve kararlı bir yanıt vermesini sağlamaktadır. ZN kazanç seti ise daha yüksek K_i değeriyle daha agresif bir kontrol davranışı ortaya koymaktadır.

Deneysel bulgular ışığında, PSO tabanlı ayarın özellikle kararlı durum hatasının azaltılmasında ve armatür akımı tepe değerlerinin sınırlanmasında avantaj sağladığı görülmüştür. Buna karşılık ZN yöntemi daha hızlı yükselme süresi sunsa da yüksek aşım ve tepe akım değerleri sebebiyle donanım güvenilirliği açısından dezavantaj oluşturabilmektedir. Dolayısıyla PSO'nun sistem güvenliği ve uzun süreli kararlılık açısından daha uygun bir çözüm sunduğu sonucuna ulaşılmıştır.

5. TARTIŞMA ve SONUÇ

Bu çalışmada, iki aşamalı deneysel inceleme gerçekleştirilmiştir. İlk olarak, doğru akım motorunun dinamik davranışı Parçacık Sürü Optimizasyonu (PSO) algoritması ile modellenmiş ve modelin geçerliliği farklı referans sinyalleri altında değerlendirilmiştir. Elde edilen sonuçlar, PSO tabanlı modelin hem sabit hem de zamana bağlı referanslar karşısında sistemin temel dinamiklerini başarılı bir şekilde temsil ettiğini göstermiştir. Özellikle adım cevabı analizinde yükselme süresi, maksimum aşım ve kararlı durum hatası gibi performans ölçütlerinin gerçek sistemle yüksek doğrulukla örtüştüğü görülmüştür. Bu bulgu, PSO modeliyle elde edilen transfer fonksiyonunun ileri kontrol tasarımları için güvenilir bir araç olabileceğini ortaya koymaktadır.

İkinci aşamada, PI kontrol deneyleri gerçekleştirilmiş ve farklı duty oranları altında dokuz farklı PI kazanç kombinasyonunun sistem üzerindeki etkileri incelenmiştir. Bu çalışmada kullanılan yük motoru, esas motora ters yönde PWM sinyalleri uygulanarak harici bozucu kaynağı olarak işlev görmüştür. Bulgular, düşük duty oranlarında yüksek P ve yüksek I kombinasyonlarının (ör. $K_p = 100, K_i = 10$) düşük kararlı durum hatası sağladığını, ancak aşırı yüksek aşım (%170–%316) ve yetersiz yerleşme davranışı nedeniyle pratikte tercih edilemeyeceğini göstermiştir. Bu durum, Åström ve Hägglund (2006) ile Ogata (2010) tarafından da vurgulandığı üzere, bu tür kazanç ayarlarının agresif karakteristiğini doğrulamaktadır.

Orta duty oranlarında (%30–%40), düşük P ve düşük I kombinasyonları en düşük hatayı sağlamış olsa da yüksek aşım değerleri nedeniyle sınırlı fayda sunmuştur. Buna karşılık, orta P ve düşük I ($K_p = 30, K_i = 1$) ve yüksek P ve düşük I ($K_p = 100, K_i = 1$) kombinasyonları geçici rejimde daha kontrollü davranış sergilemiş, literatürde Naqvi ve arkadaşlarının (2024) belirttiği şekilde hız–stabilite dengesi açısından optimum performans göstermiştir.

Yüksek duty oranlarında (%50) ise, $K_p = 100, K_i = 1$ ve $K_p = 30, K_i = 10$ kombinasyonları referans takibinde en başarılı sonuçları üretmiştir.

Düşük aşım öncelikli uygulamalarda orta P ve yüksek I daha avantajlı olurken, hız öncelikli senaryolarda yüksek P ve düşük I kombinasyonu öne çıkmıştır.

Armatür akımı analizleri de PI kontrol tasarımının değerlendirilmesinde önemli bir kriter sunmuştur. Yapılan ölçümlerde, tüm doluluk oranlarında tepe akımın motorun stall akımının yarısını aşmadığı ve %45 seviyesinin altında kaldığı görülmüştür. Bu sonuç, motorun güvenli çalışma sınırları içerisinde olduğunu göstermektedir. Düşük doluluk oranlarında (%0 ve %10) RMS ve ortalama akım değerlerinin düşük seviyelerde gerçekleşmesi, özellikle enerji verimliliği açısından avantaj sağlamaktadır. Yükün arttığı duty oranlarında (%20 ve üzeri) ise optimum kontrol kombinasyonlarının genellikle yüksek P ve düşük/orta I kazançlarıyla elde edilmesi, sistemin hem kararlılığını koruduğunu hem de akım değerlerini güvenli sınırlar içinde tuttuğunu ortaya koymuştur.

Bu bulgular, Parçacık Sürü Optimizasyonu (PSO) tabanlı kazanç belirlemede başlangıç referansı olarak doğrudan kullanılabilir. Özellikle akım sınırının optimizasyona entegre edilmesi, hem motorun hem de sürücünün korunması açısından önemlidir. Bu amaçla PSO'da çok amaçlı bir uygunluk fonksiyonu tanımlanarak, performans kriterleri (yükselme süresi, aşım, yerleşme süresi, kararlı durum hatası) ile birlikte akım limiti de dikkate alınabilir. Bu yaklaşımda akım sınırını aşan çözümlere ceza katsayısı uygulanarak, algoritmanın yalnızca yüksek performans sağlayan değil aynı zamanda güvenli akım değerlerinde çalışan $K_p - K_i$ kombinasyonlarını seçmesi sağlanmaktadır. Tez kapsamında yapılan ölçümlerde tepe akım değerlerinin %50'nin altında kaldığı göz önüne alındığında, PSO algoritmasına bu tür bir akım sınırı entegrasyonu eklenmesi, kontrolcü tasarımını hem güvenlik hem de performans açısından daha etkin kılacak potansiyele sahiptir.

Genel eğilim değerlendirildiğinde, orta P ve yüksek I ($K_p = 30, K_i = 10$) kombinasyonunun geniş doluluk aralıklarında en dengeli performansı sağladığı; yüksek P + düşük I ($K_p = 100, K_i = 1$) kombinasyonunun ise en hızlı tepkiyi verdiği, ancak aşım değerinin sınırlayıcı olabileceği belirlenmiştir. Düşük P ve yüksek I ($K_p = 1, K_i = 10$) ise daha yavaş fakat kararlı bir kontrol sağlamış ve hassas konumlama gibi aşım hassasiyetli uygulamalar için uygun bir alternatif olarak öne çıkmıştır.

Böylece kapalı çevrim kontrolörün, nominal 0 disturbance koşullarına göre tasarlanmasına rağmen, farklı ve öngörülemeyen bozucu koşullar altında da performansını sürdürdüğü deneysel olarak doğrulanmıştır. Bu yöntemle sistemin robustluğu(sağlamlığı) ortaya konmuş, parametre belirsizlikleri, modelleme hataları ve dış bozuculara karşı toleranslı olduğu gösterilmiştir. Bununla birlikte, yük motoru PWM sinyali açık çevrimle verildiğinden bozucunun hassas kontrolü yapılmamış, yalnızca dışsal bozucu etkilerin temsili sağlanmıştır. Bu durum çalışmanın sınırlılıklarından biri olarak kabul edilmiştir.

PSO arama süreci sonunda optimum nokta $K_p = 62.2, K_i = 17.8$ olarak elde edilirken, deneysel sonuçlarda ise $K_p = 30, K_i = 10$, $K_p = 1, K_i = 10$ gibi düşük değerde kombinasyonlar daha uygun bulunmuştur. Bu farklılık, PSO algoritmasında kullanılan uygunluk fonksiyonunun ağırlıklarından ve gerçek sistemdeki bozucu/gürültü etkilerinden kaynaklanmaktadır.

Çalışma kapsamında akım ve hız verileri üzerinden kapsamlı analizler gerçekleştirilmiş olmakla birlikte, motor torkunun doğrudan ölçülmesine yönelik deneysel imkânlar mevcut test düzeninde bulunmamaktadır. Kullanılan motorun yapısal özellikleri nedeniyle güvenilir tork verisi elde edilememiş, bu nedenle tork analizi çalışmaya dahil edilmemiştir. Bu durum çalışmanın bir sınırlılığı olarak değerlendirilmiş ve ilerleyen araştırmalarda uygun sensörlerle veya farklı yöntemlerle motor torkunun da incelenmesi önerilmektedir.

Sonuç olarak, PI kazanç seçiminde tek başına kararlı durum hatasının minimize edilmesi yeterli değildir. Bu çalışmada elde edilen bulgular, Åström ve Hägglund (2006), Ogata (2010) ve Naqvi ve arkadaşlarının (2024) da ortaya koyduğu gibi, aşım (M_P) ve yerleşme süresi (T_s) gibi geçici rejim kriterlerinin de dikkate alınması gerektiğini göstermektedir. Böylece, uygulama önceliklerine bağlı olarak optimum kazanç bölgeleri tanımlanabilmiş ve PI kontrol tasarımında literatürle uyumlu sonuçlar elde edilmiştir.

6. SONUÇ VE GELECEK ÇALIŞMALAR

Bu tez kapsamında, doğru akım motorlarının hız kontrol performansı hem modelleme hem de deneysel uygulamalarla incelenmiş, özellikle dış bozucular altındaki davranışları detaylı olarak değerlendirilmiştir. Çalışmada iki temel deneysel yaklaşım benimsenmiştir: İlki motor sisteminin matematiksel modellemesi, ikincisi ise FPGA üzerinde PI kontrol uygulamasıdır.

İlk aşamada, motorun dinamik davranışı Parçacık Sürü Optimizasyonu (PSO) algoritması yardımıyla modellenmiştir. Elde edilen model, hem adım cevabı hem de periyodik referans sinyalleri altında test edilmiştir. Analizler, PSO tabanlı modelin sistemin yükselme süresi, aşım, yerleşme süresi ve kararlı durum hatası gibi kritik dinamik parametrelerini büyük ölçüde doğru şekilde temsil ettiğini ortaya koymuştur. Bu sonuç, PSO'nun yalnızca parametre tahmininde değil, aynı zamanda kontrol tasarımına hazırlık aşamasında güvenilir bir yöntem olduğunu göstermektedir. Özellikle periyodik referans sinyalleri altında elde edilen sonuçlar, modelin yalnızca durağan girişler için değil, aynı zamanda zamanla değişen girişler için de başarılı bir şekilde kullanılabileceğini göstermiştir.

İkinci aşamada, Altera Cyclone IV FPGA platformu üzerinde PI kontrol algoritması donanımsal olarak gerçekleştirilmiş ve motorun hız kontrol performansı deneysel olarak gözlemlenmiştir. Enkoder tabanlı geri besleme sayesinde kapalı çevrim yapı tamamlanmış ve dokuz farklı PI kazanç kombinasyonu, %0–%50 doluluk oranları arasında denenmiştir. Farklı kazanç ayarlarının sistem cevabına etkisi incelenmiş ve sonuçlar grafikler üzerinden karşılaştırılmıştır. Elde edilen bulgular, düşük duty oranlarında yüksek P ve yüksek I kombinasyonlarının düşük kararlı durum hatası sağladığını ancak yüksek aşım ve yetersiz yerleşme davranışı nedeniyle pratikte uygun olmadığını göstermiştir. Buna karşılık orta P ve yüksek I kombinasyonları, geniş duty aralıklarında daha dengeli bir performans sergileyerek optimum aday olarak öne çıkmıştır. Yüksek P ve düşük I kombinasyonlarının en hızlı tepkiyi verdiği, fakat aşım değerlerinin belirli uygulamalarda sınırlandırıcı olabileceği görülmüştür.

Düşük P ve yüksek I kombinasyonları ise daha yavaş tepki üretmekle birlikte özellikle aşım hassasiyetinin kritik olduğu uygulamalarda tercih edilebilir.

Bu tez çalışmasında yalnızca hız yanıtı değil, aynı zamanda armatür akımı da analiz edilmiştir. ACS712 akım sensörü ile yapılan ölçümler sonucunda, tüm duty oranlarında tepe akım değerinin motorun stall akımının yarısını aşmadığı görülmüştür. Bu durum, motorun güvenli çalışma sınırları içinde çalıştığını göstermektedir. Düşük duty oranlarında düşük RMS ve ortalama akım değerleri enerji verimliliği açısından avantaj sağlamış, daha yüksek duty oranlarında ise yüksek P ve düşük/orta I kombinasyonları sistemin kararlılığını koruyarak güvenli akım sınırlarında çalışmayı mümkün kılmıştır. Bu bulgu, PI kontrol tasarımında yalnızca hız tabanlı performans değil, aynı zamanda akım güvenliği ve enerji verimliliğinin de dikkate alınması gerektiğini ortaya koymuştur.

Sonuçlar bütüncül olarak değerlendirildiğinde, PI kontrol kazançlarının seçiminde tek bir performans kriterine bağlı kalmanın yeterli olmadığı görülmüştür. Yükselme süresi, aşım, yerleşme süresi, kararlı durum hatası ve akım değerleri birlikte ele alındığında, orta P ve yüksek I kombinasyonlarının en dengeli sonuçları sunduğu, yüksek P ve düşük I kombinasyonlarının hız öncelikli uygulamalar için uygun olduğu, düşük P ve yüksek I kombinasyonlarının ise hassas konumlama gibi özel uygulamalarda kullanılabileceği ortaya konmuştur.

Bu tezden elde edilen bulgular, ileride yapılacak araştırmalara çeşitli yönlerden ışık tutmaktadır. Gelecek çalışmalar için öne çıkan konular şunlardır:

PSO optimizasyonunun geliştirilmesi: Bu tezde kullanılan PSO algoritması performans kriterlerine dayalı olarak uygulanmıştır. Gelecek çalışmalarda PSO'ya akım limiti ceza fonksiyonu yöntemiyle entegre edilerek, hem güvenlik hem de performans odaklı çok amaçlı optimizasyon yapılabilir. Böylelikle motorun sürücü ve sargılarının aşırı akımdan korunması sağlanacaktır.

Farklı kontrol yöntemleriyle karşılaştırma: PI kontrolün FPGA üzerinde başarılı bir şekilde uygulanmasına rağmen, PID, uyarlamalı kontrol, bulanık mantık ve model kestirimci kontrol gibi alternatif yöntemler de FPGA üzerinde gerçeklenerek performans karşılaştırmaları yapılabilir.

Çok boyutlu performans analizi: Bu tezde hız ve akım verileri birlikte incelenmiştir. Gelecek çalışmalarda tork, sıcaklık ve enerji tüketimi gibi ek parametreler de ölçülerek daha kapsamlı bir optimizasyon yapılabilir.

Dayanıklılık testleri: Farklı ortam sıcaklıkları, ani yük değişimleri veya uzun süreli çalışma koşulları altında sistem test edilerek, kontrol yapısının kararlılığına dair ek veriler elde edilebilir.

Gelişmiş gömülü sistem entegrasyonu: FPGA platformunun yanı sıra mikrodenetleyiciler veya SoC tabanlı yapılar ile gerçek zamanlı veri toplama, işleme ve analiz süreçleri birleştirilerek daha esnek kontrol sistemleri tasarlanabilir.

Endüstriyel uygulamalara uyarlama: Bu çalışmada laboratuvar ortamında elde edilen bulgular, daha büyük güçlü motor sistemlerine ve farklı endüstriyel senaryolara uyarlanarak doğrulanabilir.

Sonuç olarak, bu tezde gerçekleştirilen çalışmalar PI kontrolörün performansını yalnızca hız tabanlı değil, aynı zamanda akım tabanlı ölçütlerle de ele alarak daha kapsamlı bir analiz ortaya koymuştur. Elde edilen sonuçlar, literatürde önerilen yaklaşımlarla büyük ölçüde uyumludur ve pratik uygulamalar için yol gösterici niteliktedir. Ayrıca, FPGA üzerinde gerçekleştirilen donanımsal uygulama, gerçek zamanlı kontrol sistemleri için düşük gecikmeli ve güvenilir bir çözüm sunmuştur.

Bu bağlamda, tez çalışması hem teorik modelleme hem de deneysel uygulama açısından katkı sağlamış; gelecek çalışmalara yön verecek şekilde çok kriterli optimizasyon, akım güvenliği ve hız–stabilite dengesi konularında önemli çıkarımlar sunmuştur.

7. KAYNAKLAR

Abbas, A., Peerzada, P., & Larik, W. (2022). DA motor speed control through Arduino and L298N motor driver using PID controller. [Conference or Journal Name if available].

Afjei, E., Ghomsheh, A. N., & Karami, A. (2007). Sensorless speed/position control of brushed DC motor. In 2007 International Aegean Conference on Electrical Machines and Power Electronics (pp. 730–732). Bodrum, Turkey.

Aleksandrov, A. G. (2004). Reference-model adaptive control under external disturbances. Automation and Remote Control, 65(5), 755–767. https://doi.org/10.1023/B:AURC.0000028323.83877.dc

Åström, K. J., & Hägglund, T. (2006). Advanced PID Control. ISA.

Bedi, P., Kumar, A., Abdeljawad, T., & Khan, A. (2021). Study of Hilfer fractional evolution equations by the properties of controllability and stability. Alexandria Engineering Journal, 60(4), 3741–3749.

Beriş, S., & Taşdelen, K. (2023). Manyetik pozisyon sensörüne sahip SCARA robotunun modellenmesi ve sensörün analizi. International Journal of 3D Printing Technologies and Digital Industry, 7(3), 353–361. https://doi.org/10.46519/ij3dptdi.1308447

Durusu, V. (2022). Parçacık sürü optimizasyonu ve Ziegler-Nichols metodunu kullanarak DC motorun hız kontrolü için PID parametrelerinin belirlenmesi ve karşılaştırılması (Yüksek lisans tezi, Afyon Kocatepe Üniversitesi). Yükseköğretim Kurulu Ulusal Tez Merkezi. https://tez.yok.gov.tr/UlusalTezMerkezi/tezSorguSonucYeni.jsp

Feng, T., Hao, S., Hao, M., & Wang, J. (2016). Development of a combined magnetic encoder. Sensor Review, 36(4), 386–396. https://doi.org/10.1108/SR-01-2016-0020

Gökçe, B., Koca, Y., Aslan, Y., & Gökçe, C. (2021). Particle swarm optimization-based optimal PID control of an agricultural mobile robot. Comptes rendus de l'Académie bulgare des sciences: sciences mathématiques et naturelles, 74, 568–575.

Gökçe, C. O. (2024). Automatizing automatic controller design process: Designing robust automatic controller under high-amplitude disturbances using particle swarm optimized neural network controller. Applied Sciences, 14(17), Article 17. https://doi.org/10.3390/app14177859

Gökçe, C. O., Durusu, V., & Ünal, R. (2022). Farklı yük çeşitleri için parçacık sürü optimizasyonu ve Ziegler-Nichols metodunun DC motor hız kontrolü probleminde karşılaştırılması. Avrupa Bilim ve Teknoloji Dergisi, 33, 88–92. https://doi.org/10.31590/ejosat.1022991

Gökçe, C. O., İpek, M. E., Dayıoğlu, M., & Ünal, R. (2025). Parameter estimation and speed control of real DC motor with low resolution encoder. Results in Control and Optimization, 19, 100549. https://doi.org/10.1016/j.rico.2025.100549

Gülbaş, Ö. (2021). Fırçasız DC motorlar için yeni bir hız kontrol yönteminin geliştirilmesi (Yüksek lisans tezi, İskenderun Teknik Üniversitesi). http://openaccess.iste.edu.tr/xmlui/handle/20.500.12508/1985

Güneş, İ., Eryılmaz, M. M., & Tolga, Ö. (2023). Fırçasız doğru akım motorunun PI tabanlı hız kontrol sisteminin tasarımı ve uygulaması. Journal of the Institute of Science and Technology, 13(2), 983–994.

Hafez, I., & Dhaouadi, R. (2021). Parameter identification of DC motor drive systems using particle swarm optimization. In 2021 International Conference on Engineering and Emerging Technologies (ICEET) (pp. 1–6). Istanbul, Turkey.

Hajari, S., & Ray, O. (2023). A dynamic voltage-based current estimation technique for DC motor speed control applications. IEEE Sensors Letters, 1–4.

Herlambang, T., Rahmalia, D., & Yulianto, T. (2019). Particle swarm optimization (PSO) and ant colony optimization (ACO) for optimizing PID parameters on autonomous underwater vehicle (AUV) control system. Journal of Physics: Conference Series, 1211(1), 012039. https://doi.org/10.1088/1742-6596/1211/1/012039

Jabari, M., et al. (2024). Multi-stage fractional order controller for DC motor speed control. Scientific Reports, 14, 22112.

Jammousi, K., Bouzguenda, M., Dhieb, Y., Ghariani, M., & Yaich, M. (2020). Gain optimization of sliding mode speed control for DC motor. In 2020 6th IEEE International Energy Conference (ENERGYCon) (pp. 159–163). Gammarth, Tunisia.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In Proceedings of ICNN'95 - International Conference on Neural Networks (Vol. 4, pp. 1942–1948). Perth, WA, Australia.

Kim, S.-H. (2017). *Electric motor control: DC, AC, and BLDC motors*. Elsevier.

Kim, H., Park, J., & Choi, S. (2019). Robust speed control of DC motors using optimized PI parameters. IEEE Access, 7, 153214–153223. https://doi.org/10.1109/ACCESS.2019.2948567

Komić, A., & Dubravić, A. (2020). DA motor dynamics data acquisition, parameters estimation and implementation of cascade control. In 2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH) (pp. 1–5). East Sarajevo, Bosnia and Herzegovina.

Maung, M., Latt, M., & Nwe, C. (2018). DC motor angular position control using PID controller with friction compensation. International Journal of Scientific and Research Publications, 8. https://doi.org/10.29322/IJSRP.8.11.2018.p8321

Mehmet Önder Efe. (2024). Otomatik kontrol sistemleri: Matlab destekli analiz ve tasarım yaklaşımı ile. Seçkin Yayıncılık.

Naqvi, S. S. A., et al. (2024). Multi-objective optimization of PI controller for BLDC motor speed control. Energy Reports.

Nguyen, S., Hoang, A., Pham, T., & Pham, T. (2023). A low-cost real-time simulator of fuzzy logic control for brushed DC motor drives. Advances in Electrical and Electronic Engineering, 21.

Niembro-Ceceña, J. A., Gómez-Loenzo, R. A., Rodríguez-Reséndiz, J., Rodríguez-Abreo, O., & Odry, Á. (2022). Auto-regression model-based off-line PID controller tuning: An adaptive strategy for DC motor control. Micromachines, 13(8), 1264.

Ogata, K. (2010/2021). Modern Control Engineering (5th ed.). Pearson.

Omar, F., Hamdaoui, H., Ahmed Nour El Islam, A., Ayad, A., & Ardjoun, S. A. E. M. (2022). Adaptive control of DC motor without identification of parameters. Facta Universitatis - Series: Electronics and Energetics, 35, 301–312.

Phạm, T., Minh, T., Hoang, A., & Thanh, S. (2023). Parameter estimation and predictive speed control of chopper-fed brushed DC motors. International Journal of Electrical and Computer Engineering Systems, 14, 1173–1181.

Rajagiri, A., Mn, S., Nawaz, S. S., & Tummala, S. (2019). Speed control of DC motor using fuzzy logic controller by PCI 6221 with MATLAB. E3S Web of Conferences, 87, Article 01004.

Rodríguez-Abreo, O., Hernandez-Paredes, J. M., Rangel, A. F., Fuentes-Silva, C., & Velásquez, F. A. C. (2021). Parameter identification of motors by cuckoo search using steady-state relations. IEEE Access, 9, 72017–72024.

Saputra, D., Ma'arif, A., Maghfiroh, H., Chotikunnan, P., & Rahmadhia, S. N. (2023). Design and application of PLC-based speed control for DC motor using PID with identification system and MATLAB tuner. International Journal of Robotics & Control Systems, 3(2). https://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=27752658&AN=164296282

Singh, D., & Singh, B. (2020). Investigating the impact of data normalization on classification performance. Applied Soft Computing, 97, 105524. https://doi.org/10.1016/j.asoc.2019.105524

Sonugür, G., Gökçe, C., Koca, Y., İnci, Ş., & Keleş, Z. (2021). Particle swarm optimization based optimal PID controller for quadcopters. Comptes Rendus, 74, 1806–1814.

SSRN preprint. (2024–2025). Multi-objective optimization of PI controller. SSRN.

Wang, X., Chen, G., & Sun, Y. (2020). Effect of proportional gain tuning on transient response of DC motor control systems. IEEE Transactions on Control Systems Technology, 28(6), 2467–2475. https://doi.org/10.1109/TCST.2019.2945120

Widagdo, R. S., Hariadi, B., & Setyadjit, K. (2023). Modelling and analysis of Ziegler–Nichols and Chien–Hrones–Reswick tuning PID on DC motor speed control. Jurnal Teknologi Elektro, 14(1), 23–27. https://doi.org/10.22441/jte.2023.v14i1.005

Yong, K., Chen, M., Shi, Y., & Wu, Q. (2024). Flexible performance-based control for nonlinear systems under strong external disturbances. IEEE Transactions on Cybernetics, 54(2), 762–775. https://doi.org/10.1109/TCYB.2022.3224040

Zhang, Y., & Lee, T. H. (2021). Performance optimization of PI controllers under load disturbances for DC motor drives. IEEE Transactions on Industrial Electronics, 68(7), 6124–6134. https://doi.org/10.1109/TIE.2020.3012345

Zhu, Y., & Hjalmarsson, H. (2016). The Box–Jenkins Steiglitz–McBride algorithm. Automatica, 65, 170–182.

Zuo, Y., Zhu, S., Cui, Y., Liu, C., & Lin, X. (2024). Adaptive PI controller for speed control of electric drives based on model reference adaptive identification. Electronics, 13(9), 1067.

İnternet Kaynakları

1 https://www.geeksforgeeks.org/open-loop-control-system/, 01.05.2025

2 https://www.pololu.com/product/2828/, 25.05.2025

3 https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide/all, 25.05.2025

4 https://www.alldatasheet.com/datasheet-pdf/pdf/22440/STMICROELECTRONICS/L298N.html, 25.05.2025

5 https://www.ti.com/lit/ug/tidu737/tidu737.pdf, 20.05.2025

EKLER

EK 1. Tez İçin Yapılan Yayınlar

Results in Control and Optimization 19 (2025) 100549

Contents lists available at ScienceDirect

Results in Control and Optimization

journal homepage: www.elsevier.com/locate/rico

Parameter estimation and speed control of real DC motor with low resolution encoder

Celal Onur Gökçe [a,*], Mahmut Esat İpek [b], Mehmet Dayıoğlu [b], Rıdvan Ünal [b]

[a] Department of Software Engineering, Afyon Kocatepe University, Afyonkarahisar, Turkey
[b] Department of Electrical and Electronics Engineering, Afyon Kocatepe University, Turkey

ARTICLE INFO

Keywords:
Real brushed DC motor
Low resolution optical encoder
Parameter estimation
Particle swarm optimization
PI control

ABSTRACT

In this study, parameter estimation and speed control of a real brushed Direct Current (DC) motor are conducted. For parameter estimation, a popular iterative optimization algorithm, namely Particle Swarm Optimization (PSO), is used. The motor is assumed to have first-order dynamics, which are a reasonable assumption for most DC motors in practical use and two parameters representing the mathematical model of the model are estimated. The step response of the open-loop system is used as data for the parameter estimation algorithm. Using the estimated parameters simulations are conducted for closed loop Proportional-Integral (PI) control. Experiments on a real brushed DC motor are also conducted for closed-loop PI control of rotor speed. Speed of motor is measured using single channel low resolution optical encoder. Due to low resolution of speed sensor, precision of the whole system is limited and oscillations are observed in speed output measured. Controlling low-resolution sensor systems is important for two reasons. First, cost of the system is quite low compared with high resolution sensors. Second, for some real systems, especially with small physical dimensions, high resolution sensors may not be available at all. So, one of the important aspects of this study is to analyze low resolution sensor systems. As a closed loop controller, simple but popular PI controller is chosen to show the compatibility of experimental results with simulations. Oscillations around reference value are less than 10% in both simulations and experiment results. In real experiments, oscillations are slightly higher due to low resolution of encoder which is expected due to quantization error. Several parameters are used for controller and results are reported and discussed in corresponding sections.

1. Introduction

Brushed DC motors are one of the most versatile actuators used in many applications requiring speed and position control [1]. Speed control of brushed DC motors is important because high performance in speed control has a significant impact on the performance of systems that use it as a component like robots, CNC machines, automation systems and alike. Notable studies on brushed DC motor speed control include the following:

Komić and Dubravić [2] employed a microcontroller to record voltage/speed values from the DC motor and then transferred this data to the MATLAB/Simulink environment. They utilized the Parameter Estimation toolbox to determine the motor parameters and implemented a cascade control system comprising two tuned PI controllers.

* Corresponding author.
E-mail address: cogokce@aku.edu.tr (C.O. Gökçe).

https://doi.org/10.1016/j.rico.2025.100549
Received 17 August 2024; Received in revised form 3 October 2024; Accepted 13 March 2025
Available online 16 March 2025

Jammousi et al. [3], employed the Ant Colony Optimization (ACO) algorithms to optimize the parameters of a sliding mode controller for a DC motor. Using MATLAB/Simulink, a DC motor model was developed and simulated with the addition of external uncertainties and disturbances. Simulation results demonstrate the effectiveness of the ACO tuning method in obtaining better performance from the DC motor in terms of speed control. The ACO-tuned sliding mode controller offers a more accurate response in the presence of parameter variations and external disturbances compared to manual tuning.

Rodríguez-Abreo et al. [4] proposed an approach for estimating dynamic DC motor models utilizing the meta-heuristic cuckoo search algorithm. This algorithm employs a cost function derived from current and speed error in the input voltage step, leading to enhanced accuracy in parameter estimation. A comparison was made between their modified algorithm, the original cuckoo search, and the Steiglitz-McBride [5] algorithm. The findings indicated that the modified version performed better in estimating parameters.

Hafez and Dhaouadi [6] highlight the importance of determining mechanical parameters like moment of inertia and viscous friction for accurate DC motor system modeling and control, as these details are often not provided by manufacturers. They applied modified Particle Swarm Optimization (PSO) techniques to estimate parameters in a model featuring a load connected to the motor shaft.

Niembro-Ceceña et al. [7] concentrate on parameter estimation and speed control, employing time series analysis to forecast back electromotive force (EMF) values for real-time adaptation of Proportional-Integral-Derivative (PID) controllers in both brushed and brushless DC motors. The authors suggest an Auto-Regression Moving Average (ARMA) model to estimate the evolving DC motor parameters influenced by system imperfections over time. Their approach involves updating the PID controller gains using the Simulink controller tuning toolbox.

Omar et al. [8] focused on the experimental implementation of model reference adaptive control (MRAC) for a DC motor without requiring parameter identification. The study aimed to develop a controller independent of system parameters to circumvent identification issues, thus saving time and ensuring robustness against online parametric variations. The experimental results showed that the motor successfully tracked the reference speed, with adaptation parameters converging towards their nominal values.

Vanchinathan and Selvaganesan [9] propose an Artificial Bee Colony (ABC) algorithm-tuned Fractional Order PID controller for a Brushless DC motor, aiming to improve performance across various speed and load conditions. They incorporate a kalman filter for speed estimation to address limitations of hall effect sensors. Simulation results show superior performance compared to genetic algorithm-tuned controllers.

Bouhentala et al. [10] combine fuzzy clustering estimation with sliding mode control for quadrotor systems with unknown elements and external disturbances. They introduce a novel Takagi-Sugeno interval-valued fuzzy model based on input-output data to estimate nonlinear unknown components. The control strategy adaptively estimates the system's model for use in sliding mode control.

Tuẩn et al. [11] presented a method for speed control of brushed DC motors using Finite Control Set-Model Predictive Control (FCS-MPC). This approach entails estimating motor parameters utilizing output speed response, as well as measured speed and armature current under load torque conditions. The efficacy of the proposed speed control algorithm is verified through experimentation, with results indicating that the FCS-MPC controller surpasses the performance of conventional PI controllers.

Nguyen et al. [12] compared three types of controllers - a standard PI controller, a fuzzy logic controller, and a fuzzy PI controller - using a real-time simulation system. The experimental results indicate that when motor parameters are accurately known, the PI controller delivers high performance. However, in scenarios involving complex models, the fuzzy PI controller demonstrates greater utility.

Zuo et al. [13] proposed two adaptive PI controllers for speed control of electric drives, utilizing model reference adaptive identification. These controllers are designed to automatically set PI controller gains in situations where mechanical parameters are unknown. Simulation results indicate that the adaptive PI controllers successfully identify mechanical parameters and surpass the classical PI controller in noise attenuation.

Hajari and Ray [14] presents a voltage-based current estimation technique for precise speed control of DC motor systems, which eliminates the need for high bandwidth current sensors and high-speed analog-to-digital converters (ADCs). The proposed current estimation technique is shown to improve accuracy while significantly reducing costs compared to high-bandwidth current sensors, adapt well to dynamically varying terminal voltage conditions, and exhibit superior performance in real-world scenarios.

Charkoutsis and Kara-Mohamed [15] proposed a Nonlinear PID (NLPID) controller tuned using Particle PSO for first order plus time delay systems, demonstrating improved performance and robustness compared to conventional controllers.

Manuel et al. [16] compared five metaheuristic algorithms for optimizing PID controllers in DC motor speed control, finding that all algorithms achieved similar optimal gains given sufficient runs, with Teaching Learning Based Optimization (TLBO) being the fastest. They also noted that a Fuzzy Logic Controller (FLC) outperformed the metaheuristic-based PIDs.

Sachan and Swarnkar [17] investigated four metaheuristic algorithms for mobile robot speed regulation, concluding that Grey Wolf Optimization (GWO) control was superior to conventional control for time-varying and nonlinear mobile robot systems in terms of time response parameters and accuracy. Similar works can be found in Refs. [18–25], where these studies highlight the potential of metaheuristic algorithms and intelligent control strategies for improving system performance across various applications.

In this study a real brushed DC motor with low resolution optical encoder is used for parameter estimation and closed loop speed control framework. . The novelties of this paper are parameter estimation of a real brushed DC motor with PSO algorithm and PI control of a real brushed DC motor with various PI parameters all done with low resolution optical encoder. Using low resolution encoder presents some problems but has main advantages of low cost and easy production especially for very small systems. Note that the combination of PSO based parameter estimation and PI control on real brushed DC motor is itself a novel combination with the added novelty of using low resolution encoder which may be mandatory for very small systems like nanorobots. Parameter estimation is done using Particle Swarm Optimization (PSO) technique on the step response of motor. Closed loop control simulations are

conducted with estimated parameters of motor. Experiments are conducted on real DC Motor with different controller parameters.

2. Material and methods

2.1. Mathematical model of brushed DC motor

A brushed DC motor with gearbox and optical encoder is used in this study for experimental work. Electromechanical diagram and mathematical model of DC motor are given below Fig. 1.

V_a is input armature voltage, i_a is armature current, R_a is armature resistance, L_a is armature inductance, e_b is back-emf voltage, T_m is motor torque, w is rotor angular speed in rad/s, J is rotor inertia, b is viscous friction due to rollers and T_l is load torque.

Using Kirchhoff's Laws electrical system's mathematical equation is:

$$V_a = R_a i_a + L_a \frac{di_a}{dt} + e_b \tag{1}$$

Back-emf is proportional to rotor velocity:

$$e_b = K_b \dot{\theta} \tag{2}$$

Motor torque is proportional to armature current:

$$T_m = K_m i_a \tag{3}$$

Finally, using Newton's second law of motion modified for rotational systems mechanical system's mathematical equation is:

$$T_m - T_l = J\ddot{\theta} + b\dot{\theta} \tag{4}$$

By solving the aforementioned equations, a second-order system is derived, depicting the relationship between the armature voltage input and the rotor speed output [26]. Below is the transfer function obtained using the Laplace Transform for the second order system:

$$\frac{\Omega(s)}{V_a(s)} = \frac{K_m}{L_a J s^2 + (R_a J + b L_a)s + (K_m K_b + R_a b)} \tag{6}$$

In most practical brushed DC motors, the armature inductance (L_a) is significantly smaller than the armature resistance (R_a). This leads to the system being approximated as a first-order system. The transfer function of such a system can be represented as follows:

$$\frac{\Omega(s)}{V_a(s)} = \frac{1}{coefficient0 + coefficient1 * s} \tag{7}$$

In order to simulate any brushed DC motor of practical use with the first order system assumption above, two parameters, namely coefficient0 and coefficient1 must be estimated. Controllability and stability properties of Hilfer fractional equations, estimation of running time parameters of equations in DC motor control systems and contribution of the system to stable operation can be found in [27].

Note that effective armature input voltage is given using an H-bridge with pulse-width-modulation input signal and angular speed output is measured using a single-channel optical encoder which is explained below.

2.2. Optical encoder for speed measurement

For speed measurement, a single-channel optical encoder with low resolution of 96 pulse-per-revolution is used. Working principle of optical encoder is based on optoelectronics as shown in Fig. 2 below.

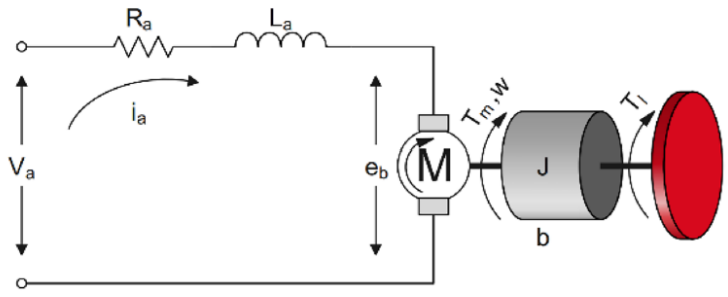


Fig. 1. Electromechanical diagram of DC motor.

An infrared (IR) LED transmitter emits invisible light, which is detected by an IR phototransistor receiver. LED and phototransistor are separated by a circular wheel with holes which is attached to motor shaft. As the wheel rotates, holes are passing between LED and phototransistor with speed proportional to the motor velocity. When a hole is positioned between the LED and the phototransistor, light illuminates the phototransistor, leading to a change in the output voltage to $V_{cc} = 3.3$ Volts; otherwise, the output voltage remains at 0 Volts. Continuous rotation of the wheel causes a square wave output frequency of which is proportional to the motor speed. By measuring this frequency, motor speed is measured.

There are two methods for estimating speed of motor. If the encoder is high resolution and desired speed is relatively high, counting the number of pulses per millisecond will give speed estimation. The second method which is used in this study is for low resolution encoders which is based on measuring the period of each pulse using a timer inside digital computing unit.

2.3. Particle swarm optimization for parameter estimation

Particle Swarm Optimization (PSO) is used to estimate the parameters of the DC motor from step response curve. PSO is an iterative optimization algorithm used to solve optimization problems that are difficult or impossible to solve by analytical methods [28]. PSO is based on a number of particles, called swarm, each represent a hypothesis for parameters to be estimated. In each iteration a cost or profit of each particle is computed first, then new position of each particle is computed according to the cost or profit scores of particles. In order to compute new position of each particle, a velocity for that particle is computed using three independent components. The velocity of each particle depends on three factors: first, the previous velocity of the particle; second, the particle's known best position up to that iteration; and third, the global best position of all particles up to that iteration. These three components are used to estimate new velocity of the particle which is used to update the next position of the particle. Fig. 3 shows the flow chart of particle swarm optimization.

2.4. Proportional integral closed loop controller

After the estimation of motor parameters with PSO algorithm, simulations and experiments on real DC motor are conducted for closed loop Proportional-Integral (PI) control of speed of DC motor. PI control is one of the simplest but most successful closed loop control algorithms especially for simple linear systems. Only two parameters are needed to compute the control output, namely Kp and Ki, proportional and integral constants.

Fig. 4 shows the flow chart of the PID Control algorithm.

3. Results

In the context of parameter estimation, a step input of 10 Volts was applied, and the angular speed was measured over a duration of 1 second. Below is a plot illustrating the step response of the DC motor in this study.

Note that the oscillations in speed are mainly due to low-resolution encoder. This data is the one we used for parameter estimation using PSO algorithm. Below is the plot of estimation output after 200 epochs.

With estimated parameters of DC motor, which are coefficient0 and coefficient1 of first order transfer function, PI control simulations are conducted for various K_p and K_i parameters of PI controller. Below are the results for four different controller parameter configurations in Fig. 7. Red line is reference motor speed while blue line is real output motor speed. Simulations are run for 1 second, which is shown in the horizontal axis. Note that for K_p=10, oscillation amplitudes are smaller than K_p=100. As K_p increases, the system

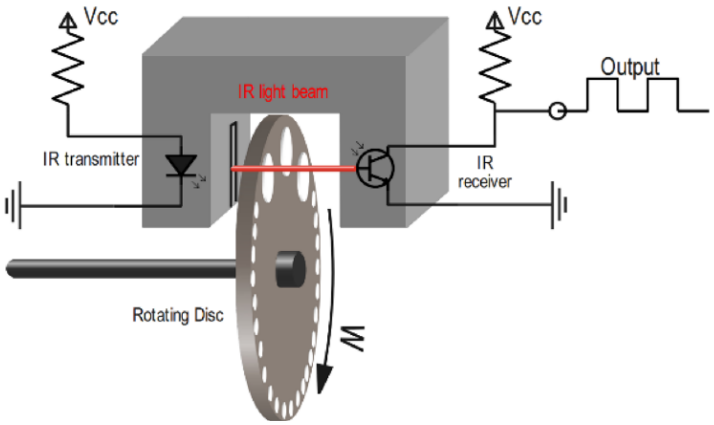


Fig. 2. Working principle of optical encoder.

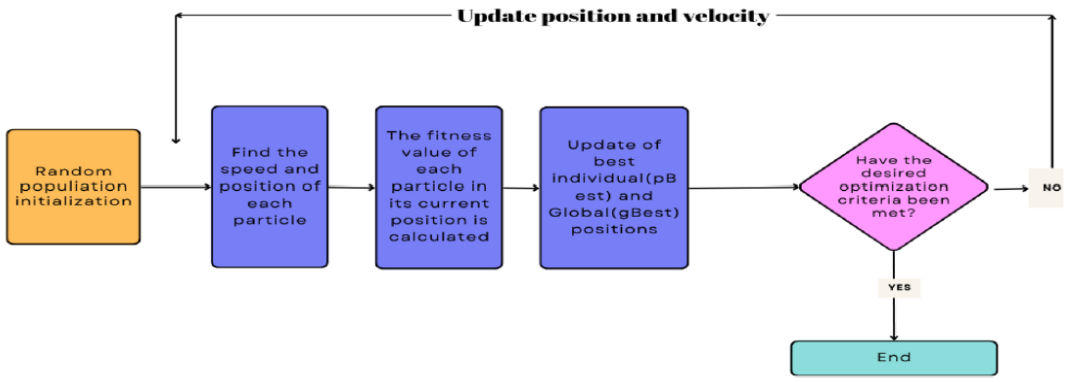


Fig. 3. Flowchart of PSO algorithm.

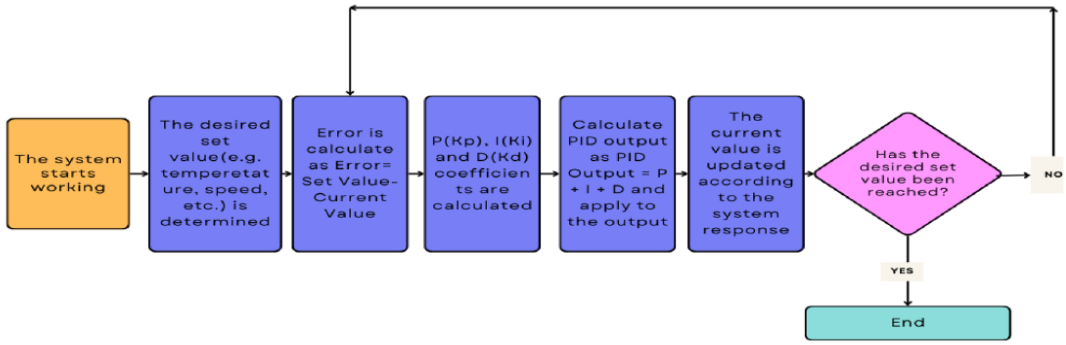


Fig. 4. Flowchart of PID Controller.

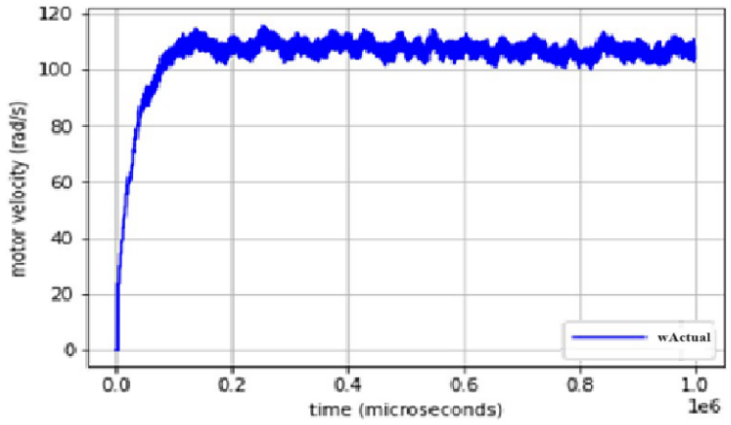


Fig. 5. Step response of DC motor to 10 Volt step armature voltage input.

becomes less stable, which is expected.

Below are the experiment results of real DC motor PI control for various controller parameters Fig. 8.

Note that results shown in the figures above are absolute values for the motor used. In order to normalize the results, a separate experiment is conducted to find the maximum velocity of the motor. Maximum armature voltage of the motor is 12 Volts, so this maximum 12 Volts is applied to the motor and velocity is measured. The measured maximum velocity of the motor is 174.097 rad/sec under 12 Volts armature voltage. This maximum velocity is used to normalize the results and below are shown in Fig. 9 the normalized

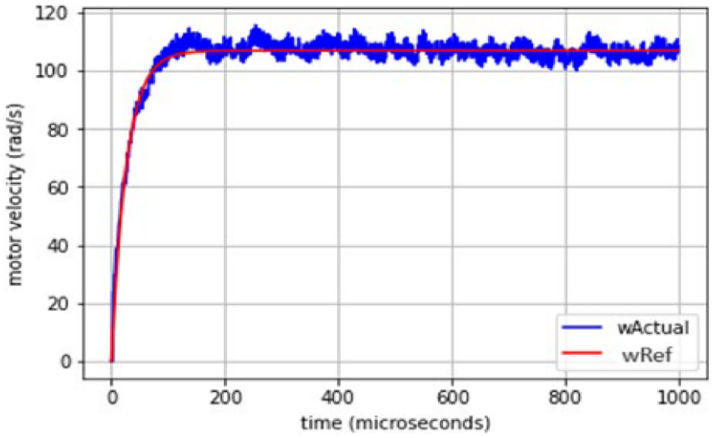


Fig. 6. Measured speed with blue, simulated speed with estimated parameters with red.

results for real DC motor PI control.

4. Discussion

Simulation results are compatible with experiment results. Note that for a simple first order linear platform like in our setup, analytic parameter estimation is also possible which may be easier than estimation with PSO. So, one may question usage of PSO algorithm for parameter estimation at all. Although analytic methods may work well for parameter estimation of simple linear systems, they become intractable when the system gets more complicated. For nonlinear systems of higher dimension with a greater number of inputs and outputs, like mobile robots [29] or quadcopters [30], mathematical models become much more complicated and impossible to solve analytically. In these types of complicated systems, iterative optimization algorithms like PSO are expected to work well, may be with extra acceptable amount of computation time and memory.

It is also seen that simulation results and experimental results deviate notably for small controller constants of K_p=10 and K_i=1 configuration as seen in Figs. 5a and 6a. This is due to static friction in the gearbox and bearings of the DC motor, which manifests for low controller constants. Note that this static friction is very hard to model and is not included in the simulations. Simple PI controller seems to give acceptable performance results for simple linear system used in this study. As the plant dynamics becomes more complicated, more advanced controller structures will be needed to deal with complex input-output relationship of system under control.

Only single-input single-output brushed DC motor which is a linear system is studied in this paper which is a limitation of this study. Nonlinear systems with multi-input multi-output scheme can assert certain problems but are expected to show acceptable results with more computation time to calculate the optimal parameters.

5. Conclusion and future work

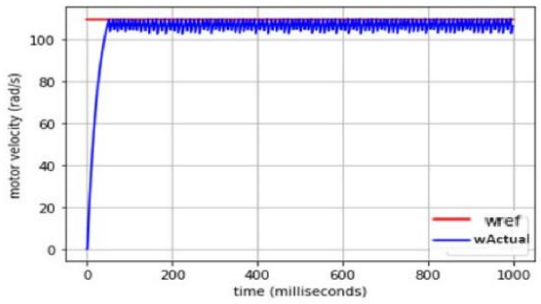
Parameter estimation and closed loop speed control of brushed DC motor is done in this study. Speed of DC motor is measured using single channel low resolution optical encoder. Motor is assumed to be a first order system with two parameters and parameters are estimated from step response curve using PSO algorithm. Both simulations and experiments on real brushed DC motor is conducted using closed loop PI control with several parameters. Encouraging results are obtained and reported. As future work more complex systems with higher order nonlinear dynamics are planned to be studied such as mobile robots and quadcopters.

Declaration of competing interest

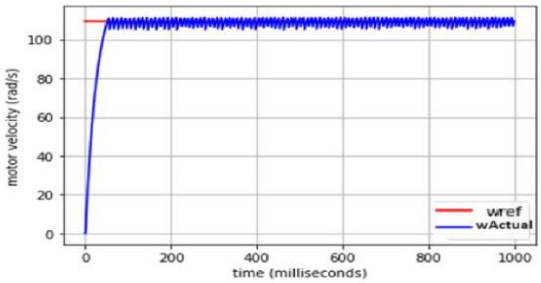
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

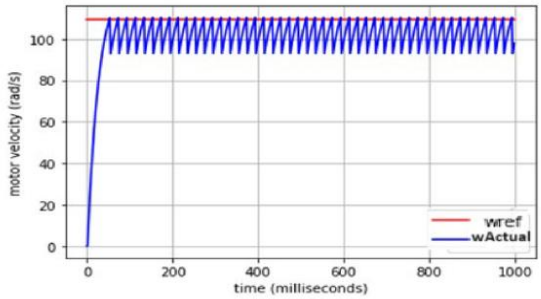
Data will be made available on request.



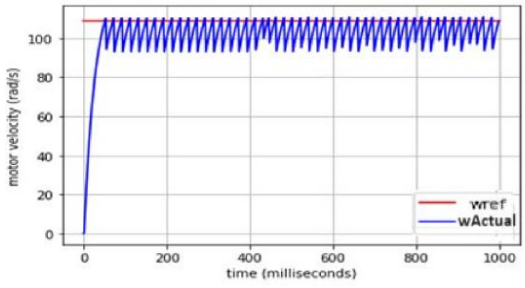
(a)



(b)

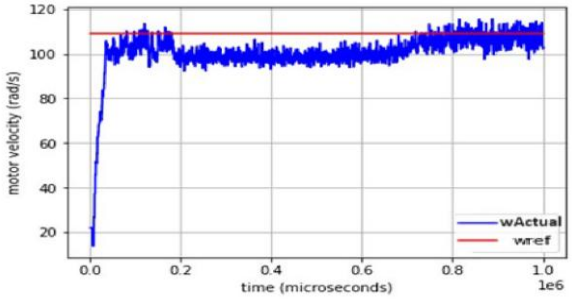


(c)

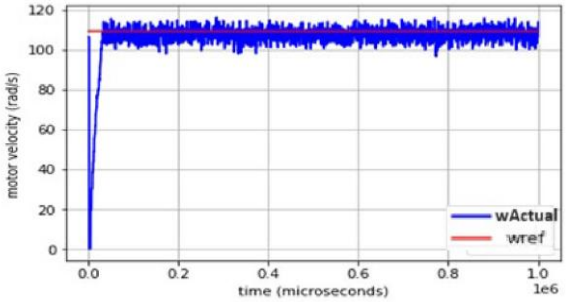


(d)

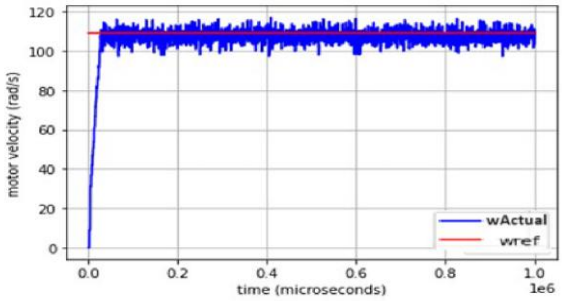
Fig. 7. Simulation of PI controller with [Kp,Ki] (a) [10,1] (b) [10,10] (c) [100,1] (d) [100,10].



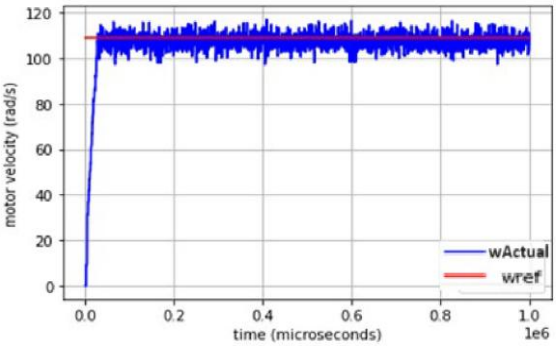
(a)



(b)



(c)



(d)

Fig. 8. Experiment results of PI controller with [Kp,Ki] (a) [10,1] (b) [10,10] (c) [100,1] (d) [100,10].

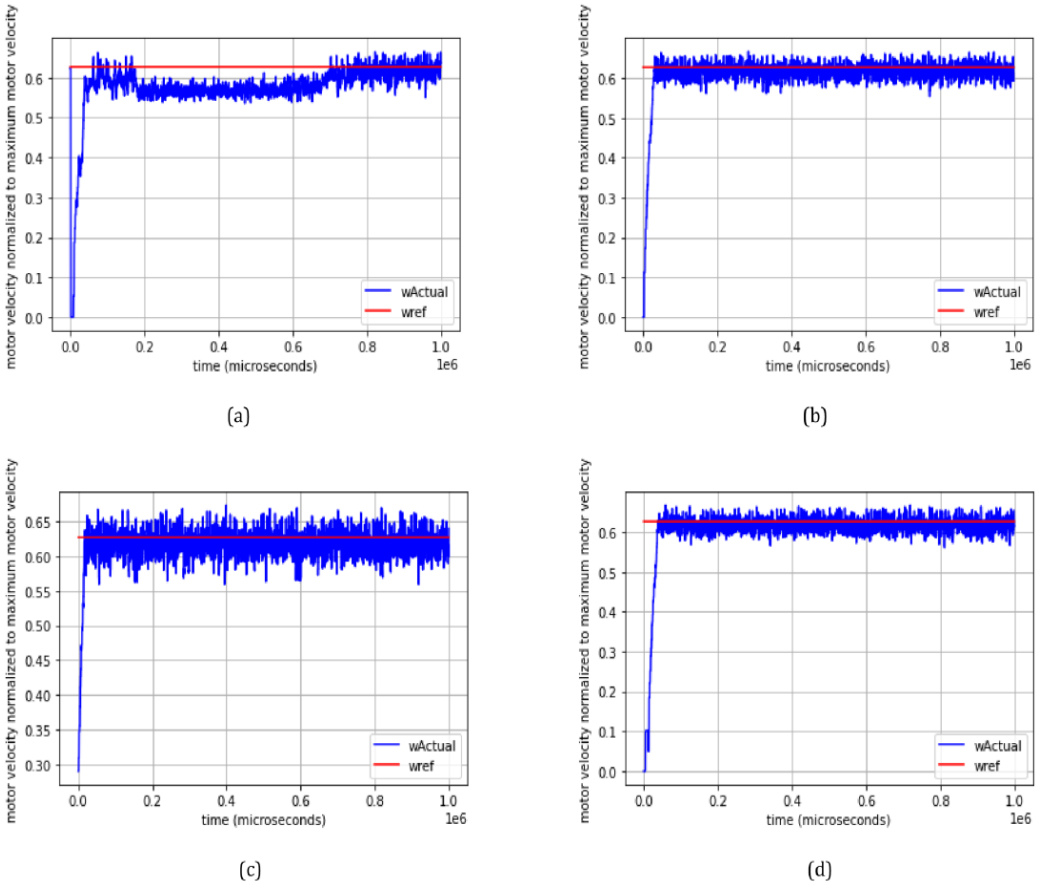


Fig. 9. Normalized experiment results of PI controller with [Kp,Ki] (a) [10,1] (b) [10,10] (c) [100,1] (d) [100,10].

References

[1] Afjei E, Ghomsheh AN, Karami A. Sensorless speed/position control of brushed DC motor. In: Proceedings of the 2007 international Aegean conference on electrical machines and power electronics; 2007. p. 730–2.

[2] Komić A, Dubravić A. DC motor dynamics data acquisition, parameters estimation and implementation of cascade control. In: Proceedings of the 2020 19th international symposium INFOTEH-JAHORINA (INFOTEH); 2020. p. 1–5.

[3] Jammousi K, Bouzguenda M, Dhieb Y, Ghariani M, Yaich M. Gain optimization of sliding mode speed control for DC motor. In: Proceedings of the 2020 6th IEEE international energy conference (ENERGYCon); 2020. p. 159–63.

[4] Rodríguez-Abreo O, Hernandez-Paredes JM, Rangel AF, Fuentes-Silva C, Velásquez FAC. Parameter identification of motors by cuckoo search using steady-state relations. IEEE Access 2021;9:72017–24.

[5] Zhu Y, Hjalmarsson H. The Box–Jenkins Steiglitz–McBride algorithm. Automatica 2016;65:170–82.

[6] Hafez I, Dhaouadi R. Parameter identification of DC motor drive systems using particle swarm optimization. In: Proceedings of the 2021 international conference on engineering and emerging technologies (ICEET); 2021. p. 1–6.

[7] Niembro-Ceceña JA, Gómez-Loenzo RA, Rodríguez-Reséndiz J, Rodríguez-Abreo O, Odry Á. Auto-regression model-based off-line PID controller tuning: an adaptive strategy for DC motor control. Micromachines 2022;13(8):1264 (Basel).

[8] Omar F, Hamdaoui H, Ahmed Nour El Islam A, Ayad A, Ardjoun SAEM. Adaptive control of DC motor without identification of parameters. Facta Univ Ser Electron Energ 2022;35:301–12.

[9] Vanchinathan K, Selvaganesan N. Adaptive fractional order PID controller tuning for brushless DC motor using artificial bee colony algorithm. Results Control Optim 2021;4:100032.

[10] Bouhentala M, Ghanai M, Chafaa K. Interval-valued fuzzy estimation and its application to adaptive control of quadrotor. Results Control Optim 2023;13: 100337.

[11] Phạm T, Minh T, Hoang A, Thanh S. Parameter estimation and predictive speed control of chopper-fed brushed DC motors. Int J Electr Comput Eng Syst 2023; 14:1173–81.

[12] Nguyen S, Hoang A, Pham T, Pham T. A low-cost real-time simulator of fuzzy logic control for brushed Dc motor drives. Adv Electr Electron Eng 2023;21. North America.

[13] Zuo Y, Zhu S, Cui Y, Liu C, Lin X. Adaptive PI controller for speed control of electric drives based on model reference adaptive identification. Electronics 2024;13 (9):1067 (Basel).

[14] Hajari S, Ray O. A dynamic voltage-based current estimation technique for DC motor speed control applications. IEEE Sens Lett 2023:1–4.

[15] Charkoutsis S, Kara-Mohamed M. A particle swarm optimization tuned nonlinear PID controller with improved performance and robustness for first order plus time delay systems. Results Control Optim 2023;12:100289.

[16] Nelson LM, İnanç N, Lüy M. Control and performance analyses of a DC motor using optimized PIDs and fuzzy logic controller. Results Control Optim 2023;13:100306.
[17] Sachan S, Swarnkar P. Investigations on meta-heuristic algorithms for intelligent speed regulation of mobile robot. Results Control Optim 2023;11:100232.
[18] Khan H, Tunç C, Baleanu D, Khan A, Alkhazzan A. Inequalities for n-class of functions using the Saigo fractional integral operator. Rev Real Acad Cienc Exactas Fís Nat Mat 2019:1–14.
[19] Malik, A., Arshad, F., & Memon, A. (2024). Design of advanced controllers for speed control of DC motor in LabVIEW. Proceedings of Pakistan Academy of Sciences: A. Physical and Computational Sciences, 61.
[20] Khan, H., Khan, Z.A., Tajadodi, H., & et al. (2020). Existence and data-dependence theorems for fractional impulsive integro-differential systems. Advances in differential equations, 2020, 458.Asd.
[21] Yousif H, Ali F. DC motor speed control using various technology controllers. Review 2024:1–13.
[22] Khan A, Abdeljawad T, Alqudah MA. Neural networking study of worms in a wireless sensor model in the sense of fractal fractional. AIMS Math 2023;8(11):26406–24.
[23] Khan A, Abdeljawad T, Khan H. A numerical scheme for the generalized ABC fractional derivative based on Lagrange interpolation polynomial. Fractals 2022;30:22401806. Article.
[24] Khan A, Abdeljawad T, Shatanawi W, Khan H. Fixed point theorems for quadruple self-mappings satisfying integral type inequalities. Filomat 2020;34:905–17.
[25] Rajagiri A, Mn S, Nawaz SS, Tummala S. Speed control of DC motor using fuzzy logic controller by PCI 6221 with MATLAB. In: Proceedings of the E3S web of conferences. 87; 2019. Article 01004.
[26] Khan A, Khan H, Gómez-Aguilar JF, Abdeljawad T. Existence and Hyers-Ulam stability for a nonlinear singular fractional differential equation with Mittag-Leffler kernel. Chaos Solit Fractals 2019;127:422–7.
[27] Bedi P, Kumar A, Abdeljawad T, Khan A. Study of Hilfer fractional evolution equations by the properties of controllability and stability. Alex Eng J 2021;60(4):3741–9.
[28] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of the ICNN'95 international conference on neural networks4; 1995. p. 1942–8.
[29] Gökçe B, Koca Y, Aslan Y, Gökçe C. Particle swarm optimization-based optimal PID control of an agricultural mobile robot. C R Acad Bulg Sci Sci Math Nat 2021;74:568–75.
[30] Sonugür G, Gökçe C, Koca Y, İnci Ş, Keleş Z. Particle swarm optimization based optimal PID controller for quadcopters. Comptes Rendus 2021;74:1806–14.

EK 2. Sistem adım tepkisi için kullanılan Python kodu

```
import csv
import numpy as np
import matplotlib.pyplot as plt
digitalThreshold = 1.5
secondPositiveEdgeTimeEstimated = 10000
deltaT = 1e-6
tMax = 2000e-3
nMax = int(tMax/deltaT)
FPGAClockFrequency = 50e6
TRefAmplitude = 600 #microseconds
TRef = TRefAmplitude*np.ones((nMax, 1))
FRefAmplitude = 1e6/TRefAmplitude
FRef = FRefAmplitude*np.ones((nMax, 1))
encoderManual = np.zeros((nMax, 1))
encoderPSO = np.zeros((nMax, 1))
n = 0
tempN = 0
#with open('C:\Users\Lenovo\Desktop\output\9V.csv', newline='') as csvfile:
```

```
#with open('C:\Users\Lenovo\Desktop\output\9V.csv', newline='') as csvfile:
with open('C:\Users\Lenovo\Desktop\output\9V.csv', newline='') as csvfile:
    spamreader = csv.reader(csvfile, delimiter=',', quotechar='|')
    for row in spamreader:
        if (tempN<3):
            tempN = tempN + 1
            continue
        encoderManual[n] = float(row[1])
        n = n+1
        if (n>=nMax):
            break
n=0
tempN = 0
#with open('C:\Users\Lenovo\Desktop\output\9V.csv', newline='') as csvfile:
with open('C:\Users\Lenovo\Desktop\output\9V.csv', newline='') as csvfile:
    spamreader = csv.reader(csvfile, delimiter=',', quotechar='|')
    for row in spamreader:
        if (tempN<3):
            tempN = tempN + 1
            continue
        encoderPSO[n] = float(row[1])
        n = n+1
        if (n>=nMax):
            break

periodMeasured = 0
frequencyMeasured = 0
tempEncoderNow = 0
tempEncoderPrevious = 0
positiveEdgeTime = 0
previousPositiveEdgeTime = 0
periodTManual = np.zeros((nMax, 1))
```

```
frequencyFManual = np.zeros((nMax, 1))
n=0
for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
   tempEncoderNow = encoderManual[n]

   if (tempEncoderNow<digitalThreshold):
      tempEncoderNow = 0
   else:
      tempEncoderNow = 1

   if ( (tempEncoderNow == 1) & (tempEncoderPrevious == 0) ):#positive edge
      previousPositiveEdgeTime = positiveEdgeTime
      positiveEdgeTime = n
      periodMeasured = positiveEdgeTime - previousPositiveEdgeTime
      if (periodMeasured == 0):
         frequencyMeasured = 0
      else:
         frequencyMeasured = 1 / (deltaT*periodMeasured)
   periodTManual[n] = periodMeasured
   frequencyFManual[n] = frequencyMeasured
   n=n+1 # Variable for iteration
   tempEncoderPrevious = tempEncoderNow

tempEncoderNow = 0
tempEncoderPrevious = 0
positiveEdgeTime = 0
previousPositiveEdgeTime = 0
periodTPSO = np.zeros((nMax, 1))
frequencyFPSO = np.zeros((nMax, 1))
n=0
for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
   tempEncoderNow = encoderPSO[n]
```

```
    if (tempEncoderNow<digitalThreshold):
        tempEncoderNow = 0
    else:
        tempEncoderNow = 1

    if ( (tempEncoderNow == 1) & (tempEncoderPrevious == 0) ):#positive edge
        previousPositiveEdgeTime = positiveEdgeTime
        positiveEdgeTime = n
        periodMeasured = positiveEdgeTime - previousPositiveEdgeTime
        if (periodMeasured == 0):
            frequencyMeasured = 0
        else:
            frequencyMeasured = 1 / (deltaT*periodMeasured)
    periodTPSO[n] = periodMeasured
    frequencyFPSO[n] = frequencyMeasured
    n=n+1 # Variable for iteration
    tempEncoderPrevious = tempEncoderNow

fig = plt.figure()
plt.plot(TRef[secondPositiveEdgeTimeEstimated:nMax-1], 'r', label='TRef')
plt.legend()
plt.plot(periodTManual[secondPositiveEdgeTimeEstimated:nMax-1],                    'b',
label='periodTManual')
plt.legend()
plt.plot(periodTPSO[secondPositiveEdgeTimeEstimated:nMax-1],                       'g',
label='periodTPSO')
plt.legend()
plt.grid()
plt.show()
fig = plt.figure()
plt.plot(FRef[secondPositiveEdgeTimeEstimated:nMax-1], 'r', label='FRef')
```

```
plt.legend()
plt.plot(frequencyFManual[secondPositiveEdgeTimeEstimated:nMax-1], 'b',
label='frequencyFManual')
plt.legend()
plt.plot(frequencyFPSO[secondPositiveEdgeTimeEstimated:nMax-1], 'g',
label='frequencyFPSO')
plt.legend()
plt.grid()
plt.show()
```

EK 3. Parametre Tahmini İçin Kullanılan Python kodu

```
 import numpy as np
import matplotlib.pyplot as plt
import random

#Assumed order of motor
dimensionOfFeatures = 2

#Time parameters of simulation
deltaT = 1e-4
tmax = 1e1
n=0
nMax = int(tmax/deltaT)

def simulateMotor(coefficient0, coefficient1, sigmaNoise, Vin):
    w = np.zeros(int(tmax/deltaT))  ## State Variable for Angular Velocity
    n=0
    for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
        #DC Motor modeled as a first order system. w: angular velocity in rad/s, u: armature
voltage in V
        dw_dt = -(coefficient0/coefficient1)*w[n]+(1/coefficient1)*Vin[n]
        w[n+1] = w[n] + deltaT*dw_dt + random.gauss(0, sigmaNoise)
```

```
        n=n+1
    return w

def calculateResemblance(wActual, wSimulated):
    totalError = 0.0
    n = 0
    for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
        totalError = totalError + abs(wActual[n] - wSimulated[n])
        n=n+1
    resemblance = 1e6 / totalError
    return resemblance

#Input armature voltage to the motor
V1 = 9 #Input voltage of 9 Volts
Vin = V1*np.ones(int(tmax/deltaT)) ## Input armature voltage

#Simulating actual system
wActual = np.zeros(int(tmax/deltaT))  ## State Variable for Angular Velocity
coefficient0Actual = 1e-3
coefficient1Actual = 1e-3
sigmaNoise = 1e-3
wActual = simulateMotor(coefficient0Actual, coefficient1Actual, sigmaNoise, Vin)
#End of simulating actual system

#PSO parameters and variables
coefficient0Min = 1e-6
coefficient0Max = 1e0
coefficient1Min = 1e-6
coefficient1Max = 1e0
xMinMax = [[coefficient0Min, coefficient0Max], [coefficient1Min, coefficient1Max]]
```

```
vMinMax = [[(xMinMax[0][0] - xMinMax[0][1])/2, (xMinMax[0][1] - xMinMax[0][0])/2], [(xMinMax[1][0] - xMinMax[1][1])/2, (xMinMax[1][1] - xMinMax[1][0])/2]]
numberOfEpochs = 100
swarmSize = 20
swarm = np.zeros((swarmSize, dimensionOfFeatures))
profitVector = np.zeros(swarmSize)
particleMaxProfit = np.zeros(swarmSize)
swarmMaxProfit = 0
p = np.zeros((swarmSize, dimensionOfFeatures)) # Particle's best point
pd = np.zeros((1, dimensionOfFeatures))
g = np.zeros((1,dimensionOfFeatures)) # Swarm's best point
gd = np.zeros((1, dimensionOfFeatures))
swarmVelocities = np.zeros((swarmSize, dimensionOfFeatures))
vd = np.zeros((1, dimensionOfFeatures))
xd = np.zeros((1, dimensionOfFeatures))
c1 = 0.7
cMax = 1.47


# Initialization
sigmaNoise = 0
for i in range(0,swarmSize):
    swarm[i][:] = [random.uniform(xMinMax[0][0], xMinMax[0][1]),random.uniform(xMinMax[1][0], xMinMax[1][1])]
    swarmVelocities[i][:] = [random.uniform(vMinMax[0][0], vMinMax[0][1]),random.uniform(vMinMax[1][0], vMinMax[1][1])]
    p[i][:] = swarm[i][:]
    w = simulateMotor(swarm[i][0], swarm[i][1], sigmaNoise, Vin)
    profitVector[i]= calculateResemblance(wActual, w)
    particleMaxProfit[i] = profitVector[i]
for i in range(0,swarmSize):
```

```
    if particleMaxProfit[i]>swarmMaxProfit:
        swarmMaxProfit = particleMaxProfit[i]
        g=swarm[i][:]
# Initialization end

# PSO
for epochNumber in range(0,numberOfEpochs):
    for i in range(0,swarmSize):
        # Equations of motion here
        vd = swarmVelocities[i][:]
        xd = swarm[i][:]
        pd = p[i][:]
        gd = g

        c2 = cMax * np.random.rand(1,dimensionOfFeatures)
        c3 = cMax * np.random.rand(1,dimensionOfFeatures)
        temp2 = [c2[0][0]*(pd[0]-xd[0]),c2[0][1]*(pd[1]-xd[1])]
        temp3 = [c3[0][0]*(gd[0]-xd[0]),c3[0][1]*(gd[1]-xd[1])]
        vd = c1*vd + temp2 + temp3
        xd = xd + vd

        if xd[0]>coefficient0Max:
            xd[0] = coefficient0Max
        if xd[0]<coefficient0Min:
            xd[0] = coefficient0Min
        if xd[1]>coefficient1Max:
            xd[1] = coefficient1Max
        if xd[1]<coefficient1Min:
            xd[1] = coefficient1Min

        swarmVelocities[i][:] = vd
        swarm[i][:] = xd
```

```
        # Equations of motion end here

    # Calculate p and g
    for i in range(0,swarmSize):
        w = simulateMotor(swarm[i][0], swarm[i][1], sigmaNoise, Vin)
        profitVector[i]= calculateResemblance(wActual, w)
        if profitVector[i]>particleMaxProfit[i]:
            p[i][:]=swarm[i][:]
            particleMaxProfit[i]=profitVector[i]
    for i in range(0,swarmSize):
        if particleMaxProfit[i]>swarmMaxProfit:
            swarmMaxProfit = particleMaxProfit[i]
            g=swarm[i][:]
    # Calculate p and g end

    if epochNumber%10 == 0:
        print(epochNumber)
        print(swarmMaxProfit)
        print(g)

# PSO end
sigmaNoise = 1e-3
w = simulateMotor(g[0], g[1], sigmaNoise, Vin)

fig = plt.figure()
plt.plot(wActual[:], 'r', label='wActual')
plt.plot(w[:], 'b', label='w')
plt.legend()
plt.grid()
plt.show()
```

EK 4. Parçacık Sürü Optimizasyonu İçin Kullanılan Pyhton Kodu

```
import numpy as np
import matplotlib.pyplot as plt
import random
import datetime
import time
import tensorflow as tf
import math

dimensionOfFeatures = 3

K_pmin = 1e-3
K_pmax = 1e3
Kimin = 1e-3
Kimax = 1e3
Kdmin = 0 #1e-3
Kdmax = 0 #1e3
xMinMax = [[K_pmin, K_pmax], [Kimin, Kimax], [Kdmin, Kdmax]]
vMinMax = [[(xMinMax[0][0] - xMinMax[0][1])/2, (xMinMax[0][1] - xMinMax[0][0])/2], [(xMinMax[1][0] - xMinMax[1][1])/2, (xMinMax[1][1] - xMinMax[1][0])/2], [(xMinMax[2][0] - xMinMax[2][1])/2, (xMinMax[2][1] - xMinMax[2][0])/2]]

numberOfEpochs = 100 #20
swarmSize = 20 #50
swarm = np.zeros((swarmSize, dimensionOfFeatures))
profitVector = np.zeros(swarmSize)
particleMaxProfit = np.zeros(swarmSize)
swarmMaxProfit = 0
p = np.zeros((swarmSize, dimensionOfFeatures)) # Particle's best point
pd = np.zeros((1, dimensionOfFeatures))
g = np.zeros((1,dimensionOfFeatures)) # Swarm's best point
```

```
gd = np.zeros((1, dimensionOfFeatures))
swarmVelocities = np.zeros((swarmSize, dimensionOfFeatures))

vd = np.zeros((1, dimensionOfFeatures))
xd = np.zeros((1, dimensionOfFeatures))

c1 = 0.7
cMax = 1.47

#Constant Variables
#-------------------------------------------------------------------------#
La = 0.35e-3 #Inductance
Ra = 0.60 # Resistance
#deltaT = 0.1*La/Ra
deltaT = 1e-4
#tmax = 2
tmax = 1e-1
n=0
nMax = int(tmax/deltaT)

Inertia = 155.4e-7    # Inertia
bVis = 1e-5 # Viscous
Km = 0.0187 #Motor Torque Constant
Kb = 0.0191 #Emf Constant
Fs = 0.02
#Parameters of actual DC motor
coefficient0 = 1e-3
coefficient1 = 1e-3

VaMax = 12.0 # Maximum Value Of Armature Voltage
VaMin = -12.0 # Minimum Value Of Armature Voltage
intErrorMax = 1e6 # Maximum Value Of Integral Of Error
```

```
intErrorMin = -1e6 # Minimum Value Of Integral Of Error
derErrorMax = 1e6
derErrorMin = -1e6

K_p = 1
Ki = 1
Kd = 0.001

referansTuru='trapezoid'
bozucuTuru='sine'
referansBuyuklugu=100
bozucuBuyuklugu=1e0 #3e-1
referansFrekansi=0
bozucuFrekansi1=1e1 #4e2
bozucuFrekansi2=1e2 #4e2
randomDisturbanceMagnitude = 1e-3
randomDisturbanceFrequency = 1e4

#def generatewRandomDisturbance(randomDisturbanceMagnitude,
randomDisturbanceFrequency):
#    TdisturbanceRandom = np.zeros((int(tmax/deltaT), 1))
#    n=0
#    temp1 = 0
#    for n1 in np.linspace(0, nMax-1, num=nMax-1):
#        n=n+1
#        if n%randomDisturbanceFrequency == 0:
#            temp1 = randomDisturbanceMagnitude*random.uniform(0, 1)
#        TdisturbanceRandom[n, 0] = temp1
#    return TdisturbanceRandom
```

```
wRefMax = referansBuyuklugu
meta_w = 20
wref = wRefMax*np.ones((int(tmax/deltaT), 1))
Va = np.zeros((int(tmax/deltaT), 1))
Tm    = np.zeros((int(tmax/deltaT), 1))  ## State Variable for Motor Torque
Tl    = np.zeros((int(tmax/deltaT), 1))  ## State Variable for Load Torque
Td = np.zeros((int(tmax/deltaT), 1))
ia    = np.zeros((int(tmax/deltaT), 1))  ## State Variable for Current
w     = np.zeros((int(tmax/deltaT), 1))  ## State Variable for Angular Velocity
error = np.zeros((int(tmax/deltaT), 1))  ##

def deneyYap(K_p, Ki, Kd, bozucuFrekansi):
    w = np.zeros(int(tmax/deltaT))  ## State Variable for Angular Velocity
    Vin = np.zeros(int(tmax/deltaT)) ## Input armature voltage
    dw_dt = 0.0 # Angular Velocity's Derivative
    di_dt = 0.0 # Armature Current's Derivative
    n=0
    intError = 0.0
    #i     = np.zeros((int(tmax/deltaT), 1))  ## State Variable for Current
    #Tm    = np.zeros((int(tmax/deltaT), 1))  ## State Variable for Motor Torque
    #Tl    = np.zeros((int(tmax/deltaT), 1))  ## State Variable for Load Torque
    #w     = np.zeros((int(tmax/deltaT), 1))  ## State Variable for Angular Velocity
    #wRef  = np.zeros((int(tmax/deltaT), 1))  ## Referance Values for Angular Velocity
    #error = np.zeros((int(tmax/deltaT), 1))  ##
    #-------------------------------------------------------------------#
    for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
        n=n+1 # Variable for iteration
        error[n,0] = wref[n-1,0] - w[n-1] # Error between reference value and output
        intError = intError + deltaT*error[n-1,0] # Integral Of Error
        derError = (error[n,0] - error[n-1,0])/deltaT
```

```
        #Compare the Integral of Error with  Max/Min Value
        if intError > intErrorMax:
            intError = intErrorMax
        if intError < intErrorMin:
            intError = intErrorMin

        if derError > derErrorMax:
            derError = derErrorMax
        if derError < derErrorMin:
            derError = derErrorMin

        # PID Control Output
        PIDOut = K_p*error[n-1,0] + Ki*intError + Kd*derError
        Va[n, 0] = PIDOut

        # Compare the Armature Voltage with Max/Min Value
        if Va[n, 0] > VaMax:
            Va[n, 0] = VaMax
        if Va[n, 0] < VaMin:
            Va[n, 0] = VaMin

        #Simulation
        #TdisturbanceRandom                                                    =
generatewRandomDisturbance(randomDisturbanceMagnitude,
randomDisturbanceFrequency)
        #Emf = Kb*w[n-1,0]
        #Tl[n,0] = Fs*np.sign(w[n-1,0])
        #Td[n,0]  =  Tl[n,0]  +  bozucuBuyuklugu*math.sin(bozucuFrekansi*n*deltaT)  +
TdisturbanceRandom[n,0]
        #Td[n,0] = Tl[n,0] + bozucuBuyuklugu*math.sin(bozucuFrekansi*n*deltaT)
        #di_dt = (1/La)*(Va[n, 0]-Ra*ia[n-1,0]-Emf)
```

```
        #ia[n,0] = ia[n-1,0] + deltaT*di_dt
        #Tm[n,0] = Km*ia[n,0]
        #dw_dt = (1/Inertia)*(Tm[n,0]-Td[n,0]-bVis*w[n-1,0])
        #w[n,0] = w[n-1,0] + deltaT*dw_dt
        #wRef[n,0] = wRefMax*np.sin(meta_w*n*deltaT)
        #wRef[n,0] = wRefMax
        Vin[n] = Va[n, 0] + bozucuBuyuklugu*math.sin(bozucuFrekansi*n*deltaT)
        dw_dt = -(coefficient0/coefficient1)*w[n-1]+(1/coefficient1)*Vin[n]
        w[n] = w[n-1] + deltaT*dw_dt

    return w

#def performansHesapla(wref, w):
#    totalError = 0.0
#    n = 0
#    for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
#        totalError = totalError + abs(wref[n,0] - w[n,0])
#        n=n+1
#    performans = 1e6 / totalError
#    return performans;

def performansHesapla(wref, w):
    totalError = 0.0
    n = 0
    for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
        totalError = totalError + abs(wref[n,0] - w[n])
        n=n+1
    performans = 1e6 / totalError
    return performans;
```

```
def generatewRefTrapezoid(wRefMax, alfaAcceleration, alfaDeceleration,
tAcceleration, tDeceleration):
    wRef = wRefMax*np.zeros((int(tmax/deltaT), 1))
    n=0
    nAcc = tAcceleration/deltaT
    nDec = tDeceleration/deltaT
    for n1 in np.linspace(0, nMax-1, num=nMax-1):
        n=n+1
        if n<=nAcc:
            wRef[n, 0] = n * deltaT * math.tan(alfaAcceleration)
        elif nAcc<n<=nDec:
            wRef[n, 0] = wRefMax
        elif nDec<n:
            wRef[n, 0] = wRefMax - (n-nDec)*deltaT*math.tan(alfaDeceleration)
    return wRef

def generatewRefSine(wRefMax, f):
    wRef = wRefMax*np.zeros((int(tmax/deltaT), 1))
    n=0
    for n1 in np.linspace(0, nMax-1, num=nMax-1):
        n=n+1
        wRef[n, 0] = wRefMax*math.sin(f*n*deltaT)
    return wRef
# Initialization
for i in range(0,swarmSize):
    swarm[i][:] = [random.uniform(xMinMax[0][0],
xMinMax[0][1]),random.uniform(xMinMax[1][0],
xMinMax[1][1]),random.uniform(xMinMax[2][0], xMinMax[2][1])]
```

```
    swarmVelocities[i][:] = [random.uniform(vMinMax[0][0],
vMinMax[0][1]),random.uniform(vMinMax[1][0],
vMinMax[1][1]),random.uniform(vMinMax[2][0], vMinMax[2][1])]
    p[i][:] = swarm[i][:]
    day = 1
    #profitVector[i]= calculateProfit(day, swarm[i][:])#performansHesapla
    w = deneyYap(swarm[i][0], swarm[i][1], swarm[i][2], bozucuFrekansi1)
    profitVector[i]= performansHesapla(wref, w)
    w = deneyYap(swarm[i][0], swarm[i][1], swarm[i][2], bozucuFrekansi2)
    temp2 = performansHesapla(wref, w)
    profitVector[i] = (profitVector[i] + temp2)/2
    particleMaxProfit[i] = profitVector[i]

tAcceleration = tmax / 5
tDeceleration = 4*tmax / 5
alfaAcceleration = math.atan2(wRefMax, tAcceleration)
alfaDeceleration = math.atan2(wRefMax, (tmax-tDeceleration))
print(alfaAcceleration)
print(alfaDeceleration)

#wref = generatewRef(wRefMax, alfaAcceleration, alfaDeceleration, tAcceleration,
tDeceleration)
fRef = 1e2
wRefMax = 5e1
wref = generatewRefSine(wRefMax, fRef)

for i in range(0,swarmSize):
    if particleMaxProfit[i]>swarmMaxProfit:
        swarmMaxProfit = particleMaxProfit[i]
        g=swarm[i][:]
# Initialization end
```

```
# PSO
for epochNumber in range(0,numberOfEpochs):
    for i in range(0,swarmSize):
        # Equations of motion here
        vd = swarmVelocities[i][:]
        xd = swarm[i][:]
        pd = p[i][:]
        gd = g

        c2 = cMax * np.random.rand(1,dimensionOfFeatures)
        c3 = cMax * np.random.rand(1,dimensionOfFeatures)
        temp2 = [c2[0][0]*(pd[0]-xd[0]),c2[0][1]*(pd[1]-xd[1]),c2[0][2]*(pd[2]-xd[2])]
        temp3 = [c3[0][0]*(gd[0]-xd[0]),c3[0][1]*(gd[1]-xd[1]),c3[0][2]*(gd[2]-xd[2])]
        vd = c1*vd + temp2 + temp3
        xd = xd + vd

        if xd[0]>K_pmax:
            xd[0] = K_pmax
        if xd[0]<K_pmin:
            xd[0] = K_pmin
        if xd[1]>Kimax:
            xd[1] = Kimax
        if xd[1]<Kimin:
            xd[1] = Kimin
        if xd[2]>Kdmax:
            xd[2]=Kdmax
        if xd[2]<Kdmin:
            xd[2]=Kdmin

        swarmVelocities[i][:] = vd
        swarm[i][:] = xd
```

```
        # Equations of motion end here

    # Calculate p and g
    for i in range(0,swarmSize):
        #profitVector[i]= calculateProfit(day, swarm[i][:])#performansHesapla
        w = deneyYap(swarm[i][0], swarm[i][1], swarm[i][2],bozucuFrekansi1)
        profitVector[i]= performansHesapla(wref, w)
        w = deneyYap(swarm[i][0], swarm[i][1], swarm[i][2],bozucuFrekansi2)
        temp1 = performansHesapla(wref, w)
        profitVector[i]=(profitVector[i]+temp1)/2
        if profitVector[i]>particleMaxProfit[i]:
            p[i][:]=swarm[i][:]
            particleMaxProfit[i]=profitVector[i]
    for i in range(0,swarmSize):
        if particleMaxProfit[i]>swarmMaxProfit:
            swarmMaxProfit = particleMaxProfit[i]
            g=swarm[i][:]
    # Calculate p and g end

    if epochNumber%10 == 0:
        print(epochNumber)
        print(swarmMaxProfit)
        print(g)

# PSO end

w = deneyYap(g[0], g[1], g[2],bozucuFrekansi1)
print(performansHesapla(wref, w))
fig = plt.figure()
plt.plot(wref[:,0], 'r', label='wref')
plt.legend()
```

```
plt.plot(w[:], 'b', label='w')
plt.legend()
plt.grid()
plt.show()

imax = VaMax/Ra
TmupperLimit = Km*imax*np.ones((int(tmax/deltaT), 1))
imin = VaMin/Ra
TmlowerLimit = Km*imin*np.ones((int(tmax/deltaT), 1))
VaUpperLimit = VaMax*np.ones((int(tmax/deltaT), 1))
VaLowerLimit = VaMin*np.ones((int(tmax/deltaT), 1))

fig2 = plt.figure()
plt.plot(Va[:,0], 'b', label='Va')
plt.legend()
plt.plot(VaUpperLimit[:,0], 'r', label='VaUpperLimit')
plt.legend()
plt.plot(VaLowerLimit[:,0], 'r', label='VaLowerLimit')
plt.legend()
plt.grid()
plt.show()

w = deneyYap(g[0], g[1], g[2],bozucuFrekansi2)
print(performansHesapla(wref, w))
fig = plt.figure()
plt.plot(wref[:,0], 'r', label='wref')
plt.legend()
plt.plot(w[:], 'b', label='w')
plt.legend()
plt.grid()
plt.show()
```

```
imax = VaMax/Ra
TmupperLimit = Km*imax*np.ones((int(tmax/deltaT), 1))
imin = VaMin/Ra
TmlowerLimit = Km*imin*np.ones((int(tmax/deltaT), 1))
VaUpperLimit = VaMax*np.ones((int(tmax/deltaT), 1))
VaLowerLimit = VaMin*np.ones((int(tmax/deltaT), 1))

fig2 = plt.figure()
plt.plot(Va[:,0], 'b', label='Va')
plt.legend()
plt.plot(VaUpperLimit[:,0], 'r', label='VaUpperLimit')
plt.legend()
plt.plot(VaLowerLimit[:,0], 'r', label='VaLowerLimit')
plt.legend()
plt.grid()
plt.show()


print(swarmMaxProfit)
print(g)
```

EK 5. Enkoder Hız Analizleri İçin Kullanılan Pyhton Kodu

```
import csv
import numpy as np
import matplotlib.pyplot as plt
digitalThreshold = 1.5
secondPositiveEdgeTimeEstimated = 1166000 #microseconds
deltaT = 1e-6 #seconds
tMax = 2.166 #seconds
nMax = int(tMax/deltaT)
FPGAClockFrequency = 50e6 #Hz
TRefAmplitude = 1000 #microseconds
TRef = TRefAmplitude*np.ones((nMax, 1))
FRefAmplitude = 1e6/TRefAmplitude
FRef = FRefAmplitude*np.ones((nMax, 1))
encoderManual = np.zeros((nMax, 1))

encoderPPR = 64 #encoder Pulse-Per-Revolution
minute2second = 60 #convert rps to rpm
rpm2radpersec = 0.1047198 #rpm-to-rad/s conversion constant

maxVelocityMeasuredinRadPerSec = 174.097

n = 0
tempN = 0
with open('C:/Users/dosya uzantısı/dosya uzantısı/31.07.2025-ÖLÇÜMLER/Ton=3000(%60 duty)/Kp100Ki10.csv', newline='') as csvfile:
    spamreader = csv.reader(csvfile, delimiter=',', quotechar='|')
    for row in spamreader:
        if (tempN<3):
            tempN = tempN + 1
            continue
        encoderManual[n] = float(row[1])
```

```
        n = n+1
        if (n>=nMax):
            break

periodMeasured = 0
frequencyMeasured = 0
tempEncoderNow = 0
tempEncoderPrevious = 0
positiveEdgeTime = 0
previousPositiveEdgeTime = 0
periodTManual = np.zeros((nMax, 1))
frequencyFManual = np.zeros((nMax, 1))
velocityMeasuredinRadPerSec = np.zeros((nMax, 1))
wref = maxVelocityMeasuredinRadPerSec*0.90*np.ones((nMax, 1))
n=0
for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
    tempEncoderNow = encoderManual[n]

    if (tempEncoderNow<digitalThreshold):
        tempEncoderNow = 0
    else:
        tempEncoderNow = 1

    if ( (tempEncoderNow == 1) & (tempEncoderPrevious == 0) ):#positive edge
        previousPositiveEdgeTime = positiveEdgeTime
        positiveEdgeTime = n
        periodMeasured = positiveEdgeTime - previousPositiveEdgeTime
        if (periodMeasured == 0):
            frequencyMeasured = 0
        else:
            frequencyMeasured = 1 / (deltaT*periodMeasured)
    periodTManual[n] = periodMeasured
```

```
    frequencyFManual[n] = frequencyMeasured
    velocityMeasuredinRadPerSec[n] =
(minute2second*frequencyMeasured/encoderPPR)*rpm2radpersec
    n=n+1 # Variable for iteration
    tempEncoderPrevious = tempEncoderNow

fig = plt.figure()
plt.plot(velocityMeasuredinRadPerSec[secondPositiveEdgeTimeEstimated:nMax-
1]/maxVelocityMeasuredinRadPerSec, 'b', label='wGerçek')
plt.legend()
plt.plot(wref[secondPositiveEdgeTimeEstimated:nMax-
1]/maxVelocityMeasuredinRadPerSec, 'r', label='wreferans')
plt.legend()
plt.grid()
plt.xlabel("Zaman (mikrosaniye)")
plt.ylabel("Normalleştirilmiş motor hızı(Maksimum motor hızı)")
plt.show()
```

EK 6. Enkoder Csv Verilerini Kırpmak İçin Kullanılan Pyhton Kodu

```
import os, zipfile, numpy as np, pandas as pd
from pathlib import Path

# ====================== AYARLAR ======================
BASE_DIR = r"C:\Users\dosya uzantısı\dosya uzantısı\31.07.2025-
ÖLÇÜMLER\Ton=1500(%30 duty)"  # <- klasörün
OUT_DIR  = r"C:\Users\dosya uzantısı\dosya uzantısı\veri_csv_cropped"  #
<- çıktı klasörü
DURATION_SECONDS = 1.0
FALLBACK_FS = 1_000_000  # 1 MHz
# ====================================================
```

```
Path(OUT_DIR).mkdir(parents=True, exist_ok=True)

def find_time_col(df: pd.DataFrame):
    for c in df.columns:
        if "time" in str(c).lower():
            return c
    return df.columns[0]

def to_seconds_from_any(x):
    # dizi/seri -> string -> sayıya çevir (virgül ve boşlukları temizle)
    x_str = pd.Series(x).astype(str).str.replace(",", "", regex=False)\
                                .str.replace(" ", "", regex=False)\
                                .str.replace("\t", "", regex=False)
    arr = pd.to_numeric(x_str, errors="coerce").to_numpy()
    if np.isfinite(arr).sum() == 0:
        return arr
    vmax = np.nanmax(arr)
    if vmax > 1e12:      # ns
        arr = arr / 1e9
    elif vmax > 1e6:     # µs
        arr = arr / 1e6
    return arr  # zaten saniye

def robust_read_csv(fp):
    # 1) C engine + low_memory=False
    try:
        return pd.read_csv(fp, header=0, low_memory=False, engine="c")
    except Exception:
        pass
    # 2) Python engine (low_memory parametresi YOK)
    try:
        return pd.read_csv(fp, header=0, engine="python")
```

```
    except Exception:
        pass
    # 3) Python engine + ; ayırıcı
    return pd.read_csv(fp, header=0, engine="python", sep=";")

def read_and_crop_csv(file_path: str, duration_s: float) -> pd.DataFrame:
    df = robust_read_csv(file_path)
    if df.shape[1] < 1:
        raise ValueError("CSV boş görünüyor.")

    time_col = find_time_col(df)
    t_sec = to_seconds_from_any(df[time_col].values)

    # Hepsi NaN ise 1 MHz fallback
    if not np.isfinite(t_sec).any():
        t_sec = np.arange(len(df), dtype=float) / float(FALLBACK_FS)

    df[time_col] = t_sec
    df = df[np.isfinite(df[time_col])]
    df_cropped = df[df[time_col] <= duration_s]
    return df_cropped

def process_folder(folder_path: str, out_root: str):
    csvs = [f for f in os.listdir(folder_path) if f.lower().endswith(".csv")]
    if not csvs:
        return None

    set_name = os.path.basename(folder_path.rstrip(r"\/"))
    out_dir = os.path.join(out_root, f"{set_name}_cropped")
    Path(out_dir).mkdir(parents=True, exist_ok=True)

    for fname in csvs:
```

```
        in_fp = os.path.join(folder_path, fname)
        try:
            df_c = read_and_crop_csv(in_fp, DURATION_SECONDS)
            out_name = fname.rsplit(".", 1)[0] + "_cropped.csv"
            out_fp = os.path.join(out_dir, out_name)
            df_c.to_csv(out_fp, index=False)
            print(f"[{set_name}] Kırpıldı: {out_name} → {len(df_c)} satır")
        except Exception as e:
            print(f"[{set_name}] HATA: {fname} → {e}")

    zip_path = os.path.join(out_root, f"{set_name}_cropped.zip")
    with zipfile.ZipFile(zip_path, 'w', zipfile.ZIP_DEFLATED) as zf:
        for root, _, files in os.walk(out_dir):
            for f in files:
                if f.lower().endswith(".csv"):
                    full = os.path.join(root, f)
                    arc  = os.path.relpath(full, out_dir)
                    zf.write(full, arc)
    print(f"[{set_name}] ZIP hazır: {zip_path}\n")
    return zip_path

# === ÇALIŞTIRMA ===
base_has_csv = any(f.lower().endswith(".csv") for f in os.listdir(BASE_DIR))
summary = []

if base_has_csv:
    zp = process_folder(BASE_DIR, OUT_DIR)
    if zp: summary.append(zp)
else:
    for sub in sorted(os.listdir(BASE_DIR)):
        full = os.path.join(BASE_DIR, sub)
        if os.path.isdir(full):
```

```
        zp = process_folder(full, OUT_DIR)
        if zp: summary.append(zp)

print("=== ÖZET ===")
print("\n".join(summary) if summary else "İşlenecek CSV bulunamadı."
```

EK 7. Armatür Akımlarını Analiz Etmek İçin Kullanılan Pyhton Kodu-1

```
import os
import re
import glob
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# === KULLANICI AYARLARI ===
ROOT = r"C:\User\dosya uzantısı\dosya uzantısı\04.08.2025 Armatür Akım Ölçümleri"
# kök klasör
OUT_DIR = os.path.join(ROOT, "_plots")                    # çıktı klasörü
os.makedirs(OUT_DIR, exist_ok=True)

delta_t_us = 1          # örnekleme zamanı [µs]
skiprows = 3            # CSV başlık satırı sayısı
stall_current = 5.5     # normalize için bölünen değer [A] (Pololu 37D)
zoom_start = 0          # yakınlaştırma başlangıcı (örnek)
zoom_end = 100          # yakınlaştırma bitişi (örnek sayısı) - None ise tüm veri

# === YARDIMCI FONKSİYONLAR ===
re_duty = re.compile(r"%\s*(\d+)\s*duty", re.IGNORECASE)
re_kpki = re.compile(r"Kp\s*(\d+(?:\.\d+)?)\s*Ki\s*(\d+(?:\.\d+)?)", re.IGNORECASE)

def extract_duty_from_path(path: str):
```

```
    # Üst dizin adından duty yüzdesini bul
    parts = os.path.normpath(path).split(os.sep)
    for p in reversed(parts):
        m = re_duty.search(p)
        if m:
            return int(m.group(1))
    return None

def extract_kp_ki_from_filename(filename: str):
    name = os.path.splitext(os.path.basename(filename))[0]
    m = re_kpki.search(name)
    if m:
        kp = float(m.group(1))
        ki = float(m.group(2))
        return kp, ki
    # Yedek: "Kp100,Ki10.csv" gibi virgüllü veya altçizgili varyantlar için basit tarama
    m2 = re.search(r"Kp\s*([0-9.]+)", name, re.IGNORECASE)
    m3 = re.search(r"Ki\s*([0-9.]+)", name, re.IGNORECASE)
    kp = float(m2.group(1)) if m2 else None
    ki = float(m3.group(1)) if m3 else None
    return kp, ki

def read_current_series(csv_path: str):
    df = pd.read_csv(csv_path, skiprows=skiprows, header=None)
    # İkinci sütun varsa onu al, yoksa birinciyi
    current = df.iloc[:, 1] if df.shape[1] >= 2 else df.iloc[:, 0]
    current = pd.to_numeric(current, errors='coerce').fillna(method='ffill').fillna(method='bfill').values
    return current

def compute_metrics(current_norm: np.ndarray, z0: int, z1: int):
    # Yakınlaştırma dilimi
```

```
    if z1 is None or z1 > len(current_norm):
        z1 = len(current_norm)
    seg = current_norm[z0:z1]
    mean_val = float(np.mean(seg)) if len(seg) else np.nan
    rms_val  = float(np.sqrt(np.mean(seg**2))) if len(seg) else np.nan
    peak_val = float(np.max(seg)) if len(seg) else np.nan
    return mean_val, rms_val, peak_val, z0, z1

def plot_and_save(time_us: np.ndarray, cur_norm: np.ndarray, mean_v, rms_v, peak_v,
                  duty, kp, ki, csv_path: str):
    # Başlık/etiketler
    title = f"Normalize Armatür Akımı (Duty %{duty}, Kp={int(kp)}, Ki={int(ki)})"
    fname = os.path.splitext(os.path.basename(csv_path))[0]
    # Duty bazlı alt klasör
    duty_dir = os.path.join(OUT_DIR, f"%{duty}_duty")
    os.makedirs(duty_dir, exist_ok=True)
    out_png = os.path.join(duty_dir, f"{fname}.png")

    plt.figure(figsize=(12, 6))
    plt.plot(time_us, cur_norm, label="Normalize Akım (I / 5.5 A)")
    plt.axhline(mean_v, linestyle='--', label=f"Ortalama: {mean_v:.3f}")
    plt.axhline(rms_v,  linestyle='--', label=f"RMS: {rms_v:.3f}")
    plt.axhline(peak_v, linestyle='--', label=f"Tepe: {peak_v:.3f}")
    plt.xlabel("Zaman (µs)")
    plt.ylabel("Normalize Akım (I / 5.5 A)")
    plt.title(title)
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.savefig(out_png, dpi=150)
    plt.close()
    return out_png
```

```
# === ANA AKIŞ ===
all_rows = []
pattern = os.path.join(ROOT, "**", "*.csv")
files = sorted(glob.glob(pattern, recursive=True))

if not files:
    print("CSV bulunamadı. Dizin yolunu kontrol edin.")
else:
    print(f"Bulunan CSV sayısı: {len(files)}")

for fpath in files:
    try:
        duty = extract_duty_from_path(fpath)
        kp, ki = extract_kp_ki_from_filename(fpath)
        current = read_current_series(fpath)
        time_vector = np.arange(len(current)) * delta_t_us
        current_norm = current / stall_current

        # Yakınlaştırma aralığı
        z0 = zoom_start
        z1 = zoom_end if zoom_end is not None else len(current_norm)
        z1 = min(z1, len(current_norm))
        time_zoom = time_vector[z0:z1]
        cur_zoom = current_norm[z0:z1]

        mean_v, rms_v, peak_v, z0r, z1r = compute_metrics(current_norm, z0, z1)
        out_png = plot_and_save(time_zoom, cur_zoom, mean_v, rms_v, peak_v, duty, kp,
ki, fpath)

        all_rows.append({
            "duty_%": duty,
```

```
            "Kp": kp,
            "Ki": ki,
            "csv_path": fpath,
            "plot_path": out_png,
            "samples_total": len(current_norm),
            "zoom_start": z0r,
            "zoom_end": z1r,
            "samples_zoom": z1r - z0r,
            "mean_norm": mean_v,
            "rms_norm": rms_v,
            "peak_norm": peak_v
        })

        print(f"[OK] Duty %{duty} | Kp={kp}, Ki={ki} -> {os.path.basename(out_png)}")

    except Exception as e:
        print(f"[HATA] {fpath}: {e}")

# ÖZET TABLO
if all_rows:
    summary_df = pd.DataFrame(all_rows)
    # Sıralama: duty, Kp, Ki
    summary_df = summary_df.sort_values(by=["duty_%", "Kp", "Ki"],
kind="mergesort")
    summary_csv = os.path.join(OUT_DIR, "_ozet_akim_metrikleri.csv")
    summary_df.to_csv(summary_csv, index=False, encoding="utf-8-sig")
    print(f"\nÖzet tablo kaydedildi: {summary_csv}")

    # Her duty için ayrı özet
    for duty_val, grp in summary_df.groupby("duty_%"):
        duty_csv = os.path.join(OUT_DIR, f"_ozet_duty_{duty_val}.csv")
        grp.to_csv(duty_csv, index=False, encoding="utf-8-sig")
```

```
        print(f"Duty %{duty_val} özeti: {duty_csv}")

print("\nTamamlandı.")
```

EK 8. Armatür Akımlarını Analiz İçin Kullanılan Pyhton Kodu-2

```
import os, re, glob
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# ==================== KULLANICI AYARLARI ====================
ROOT = r"C:\Users\Lenovo\Desktop\04.08.2025 Armatür Akım Ölçümleri"
OUT_DIR = os.path.join(ROOT, "_analysis")
os.makedirs(OUT_DIR, exist_ok=True)

delta_t_us   = 1        # örnekleme zamanı [µs]
skiprows     = 3        # CSV başlık satırı sayısı
stall_current = 5.5     # normalize için bölünen değer [A]

# Üç pencere: (etiket, start_µs, end_µs). end=None => serinin sonu
WINDOWS = [
   ("w0_100us",   0,    100),
   ("w0_500us",   0,    500),
   ("w500_15000us", 500, 15000),
]

# Isı haritası parametreleri
HEAT_DIR = os.path.join(OUT_DIR, "heatmaps")
os.makedirs(HEAT_DIR, exist_ok=True)
dpi = 160

# ==================== REGEX & YARDIMCI ====================
re_duty = re.compile(r"%\s*(\d+)\s*duty", re.IGNORECASE)
re_kpki = re.compile(r"Kp\s*([0-9.]+)\s*Ki\s*([0-9.]+)", re.IGNORECASE)
```

```
def extract_duty_from_path(path: str):
    for p in os.path.normpath(path).split(os.sep)[::-1]:
        m = re_duty.search(p)
        if m: return int(m.group(1))
    return None

def extract_kp_ki_from_filename(filename: str):
    name = os.path.splitext(os.path.basename(filename))[0]
    m = re_kpki.search(name)
    if m: return float(m.group(1)), float(m.group(2))
    # esnek tarama (Kp100,Ki10 gibi)
    mkp = re.search(r"Kp\s*([0-9.]+)", name, re.IGNORECASE)
    mki = re.search(r"Ki\s*([0-9.]+)", name, re.IGNORECASE)
    return (float(mkp.group(1)) if mkp else None,
            float(mki.group(1)) if mki else None)

def read_current(csv_path: str):
    df = pd.read_csv(csv_path, skiprows=skiprows, header=None)
    series = df.iloc[:,1] if df.shape[1] >= 2 else df.iloc[:,0]
    series = pd.to_numeric(series,
errors="coerce").interpolate(limit_direction="both").to_numpy()
    return series

def slice_by_microseconds(arr, start_us, end_us, dt_us):
    n0 = int(start_us / dt_us)
    n1 = int(end_us / dt_us) if end_us is not None else len(arr)
    n0 = max(0, n0); n1 = min(len(arr), n1)
    return arr[n0:n1], n0, n1

def metrics(x):
    if x.size == 0:
        return np.nan, np.nan, np.nan
```

```
    mean = float(np.mean(x))
    rms  = float(np.sqrt(np.mean(x**2)))
    peak = float(np.max(x))
    return mean, rms, peak

def ensure_dir(path):
    os.makedirs(path, exist_ok=True)

# ==================== TARANAN DOSYALAR ====================
files = sorted(glob.glob(os.path.join(ROOT, "**", "*.csv"), recursive=True))
if not files:
    print("CSV bulunamadı. ROOT yolunu kontrol edin.")

rows = []

for f in files:
    try:
        duty = extract_duty_from_path(f)
        kp, ki = extract_kp_ki_from_filename(f)
        if duty is None or kp is None or ki is None:
            print(f"Atlandı (etiket bulunamadı): {f}")
            continue

        cur = read_current(f) / stall_current

        for wname, w0, w1 in WINDOWS:
            seg, i0, i1 = slice_by_microseconds(cur, w0, w1, delta_t_us)
            mean_v, rms_v, peak_v = metrics(seg)

            rows.append({
                "duty_%": duty, "window": wname, "Kp": kp, "Ki": ki,
                "csv_path": f, "zoom_start_us": w0, "zoom_end_us": w1,
```

```
            "samples_zoom": int((i1 - i0)), "mean_norm": mean_v,
            "rms_norm": rms_v, "peak_norm": peak_v
        })

    except Exception as e:
        print(f"Hata: {f} -> {e}")

# ==================== ÖZET CSV'LER ====================
if not rows:
    raise SystemExit("İşlenecek satır yok.")

df = pd.DataFrame(rows)
df = df.sort_values(["duty_%", "window", "Kp", "Ki"], kind="mergesort")

all_csv = os.path.join(OUT_DIR, "akim_metrikleri_tum_pencereler.csv")
df.to_csv(all_csv, index=False, encoding="utf-8-sig")
print(f"Genel özet: {all_csv}")

# Ayrı ayrı duty & window bazlı özetler
for (duty, wname), grp in df.groupby(["duty_%", "window"]):
    duty_dir = os.path.join(OUT_DIR, f"%{duty}_duty")
    ensure_dir(duty_dir)
    out_csv = os.path.join(duty_dir, f"ozet_{wname}.csv")
    grp.to_csv(out_csv, index=False, encoding="utf-8-sig")
    print(f"Duty %{duty} – {wname}: {out_csv}")

# ==================== ISI HARİTALARI ====================
def plot_heatmap(pvt, title, out_png):
    plt.figure(figsize=(7.5, 6))
    # Kp satırlar, Ki sütunlar olacak şekilde pvt index/sütun düzenini koru
    im = plt.imshow(pvt.values, aspect="auto", origin="lower")
    plt.colorbar(im, fraction=0.046, pad=0.04)
```

```
    plt.xticks(ticks=np.arange(len(pvt.columns)), labels=[f"{c:g}" for c in pvt.columns], rotation=45)
    plt.yticks(ticks=np.arange(len(pvt.index)),   labels=[f"{i:g}" for i in pvt.index])
    plt.xlabel("Ki"); plt.ylabel("Kp"); plt.title(title)
    plt.tight_layout()
    plt.savefig(out_png, dpi=dpi)
    plt.close()

# Her duty ve pencere için 3 metrik (mean/rms/peak) ısı haritası
for (duty, wname), grp in df.groupby(["duty_%", "window"]):
    # Kp-Ki ızgara
    for metric, title_metric in [("mean_norm","Ortalama"),
                                 ("rms_norm","RMS"),
                                 ("peak_norm","Tepe")]:
        pvt = grp.pivot_table(index="Kp", columns="Ki", values=metric, aggfunc="mean")
        if pvt.isna().all().all():  # veri yoksa atla
            continue
        out_dir = os.path.join(HEAT_DIR, f"%{duty}_duty", wname)
        ensure_dir(out_dir)
        out_png = os.path.join(out_dir, f"heat_{metric}.png")
        title = f"Normalize Akım {title_metric} Isı Haritası\nDuty %{duty} – Pencere: {wname.replace('_',' ')}"
        plot_heatmap(pvt, title, out_png)
        print(f"[Isı Haritası] {out_png}")

print("Tamamlandı.")
```