PARÇACIK SÜRÜ OPTİMİZASYONU VE ZIEGLER-NICHOLS METODUNU KULLANARAK DC MOTORUN HIZ KONTROLÜ İÇİN PID PARAMETRELERİNİN BELİRLENMESİ VE KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Volkan DURUSU

Danışman
Prof. Dr. Rıdvan ÜNAL

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Temmuz 2022

AFYON KOCATEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

PARÇACIK SÜRÜ OPTİMİZASYONU VE ZIEGLER-NICHOLS METODUNU KULLANARAK DC MOTORUN HIZ KONTROLÜ İÇİN PID PARAMETRELERİNİN BELİRLENMESİ VE KARŞILAŞTIRILMASI

Volkan DURUSU

Danışman
Prof. Dr. Rıdvan ÜNAL

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Temmuz 2022

ÖZET

Yüksek Lisans Tezi

PARÇACIK SÜRÜ OPTİMİZASYONU VE ZIEGLER-NICHOLS METODUNU KULLANARAK DC MOTORUN HIZ KONTROLÜ İÇİN PID PARAMETRELERİNİN BELİRLENMESİ VE KARŞILAŞTIRILMASI

Volkan DURUSU
Afyon Kocatepe Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Rıdvan ÜNAL

Otonom sistemlerin en önemli bileşenlerinden birisi otomatik kontrol sistemleridir. Otomatik kontrol konusunda hem çok sayıda araştırma yapılmakta hem de çok sayıda proje geliştirilmektedir. DC motorların gerek otonom sistemlerde gerekse endüstride çok fazla kullanılmasından dolayı bu çalışmada DC motorlar seçilmiştir. DC motor hız kontrolü için PID kontrolör kullanılmış ve parametreleri iki farklı yaklaşımla ayarlanmıştır. Birinci yaklaşım Ziegler-Nichols metodu (ZN), ikinci yaklaşım ise nispeten daha yeni bir yöntem olan Parçacık Sürü Optimizasyonu (PSO) metodudur. Mutlak hataların toplamı türünden performans metriği ile yaklaşımların birbirine göre performansları hesaplanmıştır. Sistem farklı referans çeşitleri ve farklı yük çeşitleri altında test edilmiş ve sonuçlar raporlanmıştır. Uygulanan tüm referans ve yük çeşitlerinde PSO yaklaşımının ZN'ye göre performansının daha yüksek olduğu görülmüştür. Özellikle karmaşık referans ve yük çeşitlerinde performans farkının daha da arttığı gözlemlenmiş ve sonuçlar paylaşılmıştır. İlerideki çalışmalarda daha karmaşık referans ve yük çeşidi uygulayıp, sonuçların gerçek sistem sonuçları ile karşılaştırılması önerilmektedir.

2022, viii + 76 sayfa

Anahtar Kelimeler: DC Motor, PID Kontrol, Ziegler-Nichols, Parçacık Sürü Optimizasyonu

ABSTRACT

M.Sc Thesis

DETERMINATION AND COMPARISON OF PID PARAMETERS FOR SPEED CONTROL OF DC MOTOR USING PARTICLE SWARM OPTIMIZATION AND ZIEGLER-NICHOLS METHOD

Volkan DURUSU

Afyon Kocatepe University

Graduated School of Natural and Applied Sciences

Department of Electrical and Electronics Engineering

Supervisor: Prof. Rıdvan ÜNAL

One of the most important components of every autonomous system is the automatic control system. A lot of research and projects are done on automatic control applications. Since DC Motors are widely used in both autonomous systems and industry, they are chosen as the platform to be controlled in this study. PID controller has been used for DC Motor velocity control and its parameters are determined using two different types of techniques. The first techique is the one that is mentioned widely in automatic control books, namely Ziegler-Nichols (ZN) method, and the second technique is relatively new method of Particle Swarm Optimization (PSO). The performance of the approaches were measured using the sum of absolute errors performance metric. System is tested with various reference types under different types of load and the results are reported. It has been observed that the performance of the PSO approach is higher than the ZN approach in all reference and load types applied in the study. It has been observed that the performance difference has increased even more, especially in complex reference and complex load types, and the results have been reported. In future studies, it is suggested to apply more complex reference and load types and compare the results with the actual system results.

2022, viii + 76 pages

Keywords: DC Motor, PID control, Ziegler-Nichols, Particle Swarm Optimization

TEŞEKKÜR

Bu çalışmanın yayınlanması için bana hiçbir zaman desteğini esirgemeyen, lisans 1.sınıftan beri bir hocadan çok daha fazlası olan, söylediği her cümlenin altın değerinde olduğunu çok daha iyi anladığım, değerli ve saygıdeğer danışman Hocam Sayın Prof. Dr. Rıdvan ÜNAL'a yoluma ışık olduğu için en içten dileklerimle teşekkür ederim.

Tanıdığım en zeki insanlardan biri ve bir o kadar da alçak gönüllü olan, hiçbir zaman yardımını esirgemeyip her sorumu cevaplayan, çalışmanın buralarda gelmesinde en büyük etkilerden biri olan değerli ve saygıdeğer Hocam Sayın Dr. Celal Onur GÖKÇE'ye çok teşekkür ederim.

Bugüne kadar yaptığım her eylem, verdiğim her kararda arkamda olan sevgili aileme teşekkür ederim.

Çalışmamda yüreklendirici ve pes etmemem gerektiğini söyleyip yanımda duran, desteklerini hiç esirgemeyen değerli dostlarıma çok teşekkür ederim.

Lisans ve yüksek lisans eğitimim boyunca kendilerinden ders alma fırsatı bulduğum tüm hocalarıma çok teşekkür ederim.

Bize bırakılan manevi miras ilim ve aklı kullanarak ülkeme ve milletime destek olmama yardımı dokunmuş ve dokunacak herkese şimdiden teşekkür ederim.

Volkan DURUSU

Afyonkarahisar, 2022

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

DC	Direct Current
FOPID	Fractional Order Proportional-Integral-Derivative
GA	Genetik Algorithm
ITAE	Integral Time Absolute Error
LQR	Linear Quadratic Regulator
PID	Proportional-Integral-Derivative
PSO	Particle Swarm Optimization
ZN	Ziegler-Nichols

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

1. GİRİŞ

Alessandro Volta'nın 1800 yılında bataryayı keşfinden sonra elektrik biliminde yeni bir çağ başlamıştır. Bu keşiften tam yirmi yıl sonra, 1820 yılında H. C. Oersted tarafından elektrik akımını kullanarak manyetik alan elde edilebileceği bulunmuştur. Aynı yıl A. M. Ampére selenoidi icat etmiştir. 1821 yılına gelindiğinde ise M. Faraday, elektromanyetik dönüşü gösteren bir deney gerçekleştirmiştir.

Daha sonra 1825 yılında William Sturgeon tarafından elektromıknatıs keşfedilmiş ve elektrik motorlarının keşfinin önü açılmıştır. 1831 yılında yine Faraday tarafından elektromanyetik indüksiyon keşfedilmiştir. Oersted'in keşfinin tersi olarak ifade edebileceğimiz bu keşif, değişken bir manyetik alan içinden iletken bir tel geçirildiğinde bir gerilim oluşacağından bahsetmektedir.

Birçok motor üretme girişimlerinin ardından ilk dönen elektrik motorunu 1834 yılında M. Jacobi yapmıştır ve motor çıkışından kayda değer bir mekanik enerji elde etmeyi başarmıştır. Buradan hareketle elektrik motorunun tanımı da ortaya çıkmaktadır. Elektrik enerjisini mekanik enerjiye çeviren motorlara elektrik motoru denir (İnt. Kyn. 1).

Otomatik kontrol, modern hayatta pek çok farklı uygulaması olan ileri bir mühendislik konusudur. İnsanlı/insansız araçlar, endüstride kullanılan bilgisayarlı sayısal kontrol (computer numerical control-CNC) tezgahlar, robotlar ve diğer üretim sistemleri, evlerde kullanılan elektronik aletler başta olmak üzere pek çok karmaşık sistemin merkezinde otomatik kontrol vardır. Yine pek çok karmaşık sistemin önemli bir parçası elektrik motorlarıdır ve elektrik motorlarının özellikle hız ve konum kontrollerini hassas bir şekilde yapabilmek parçası oldukları sistemin nihai performansında kritik bir önem taşır.

Bu çalışmada Fırçalı Doğru Akım (Direct Current-DC) motorunun hız kontrolü üzerine odaklanılmıştır. Özellikle DC motor seçilmesinin iki ana nedeni vardır. Birincisi çok yaygın olarak kullanılması, ikincisi bir giriş bir çıkışlı doğrusal bir sistem olması nedeniyle analitik olarak analizinin nispeten kolay olması ile önerilen yeni bir otomatik kontrol algoritması için çok güzel bir test platformu oluşturmasıdır.

Gerçek hayatta kullanılan otomatik kontrol mekanizmalarının çok büyük bir kısmını Oransal-İntegral-Türevsel (Proportional-Integral-Derivative-PID) kontrolörler oluşturmaktadır. Bunun birkaç sebebi arasında yapısının basit olması, ayarlanması gereken değişken sayısının az olması ve fiziksel olarak gerçekleştirilmesinin kolay olması sayılabilir. PID kontrolör tasarımında yapılan işlem, her bir kontrol değişkeni için üç adet parametrenin seçilmesidir. Bu parametreler K_p, K_i ve K_d olarak adlandırılan ayrıntıları ilerleyen bölümlerde verilen parametrelerdir.

Otomatik kontrol derslerinde ve kitaplarında sıklıkla anlatılan, artık klasikleşmiş bir PID kontrolör parametre bulma yöntemi Ziegler-Nichols (ZN) adı verilen yöntemdir. Bu çalışmada, ZN yöntemi ile nispeten yenilerde önerilmiş olan Parçacık Sürü Optimizasyonu (PSO) yöntemi karşılaştırılmış ve bulgular raporlanarak sonuçlar hakkında yorumlar yapılmıştır.

2. LİTERATÜR TARAMASI

2.1 DC Motor

Elektrik motorlarının temelde stator ve rotor olmak üzere iki kısmı vardır. Stator sargıların sarıldığı kısım olup gerekli manyetik alanın sağlandığı kısımdır. Rotor ise stator sargılarının arasında bulunan, motorun dönen kısmıdır. Elektrik motorları akım türlerine göre alternatif akım (Alternative Current-AC) ve DC olarak ikiye ayrılır.

DC motorlar başlıca; fırçalı ve fırçasız olarak çeşitlendirilir.

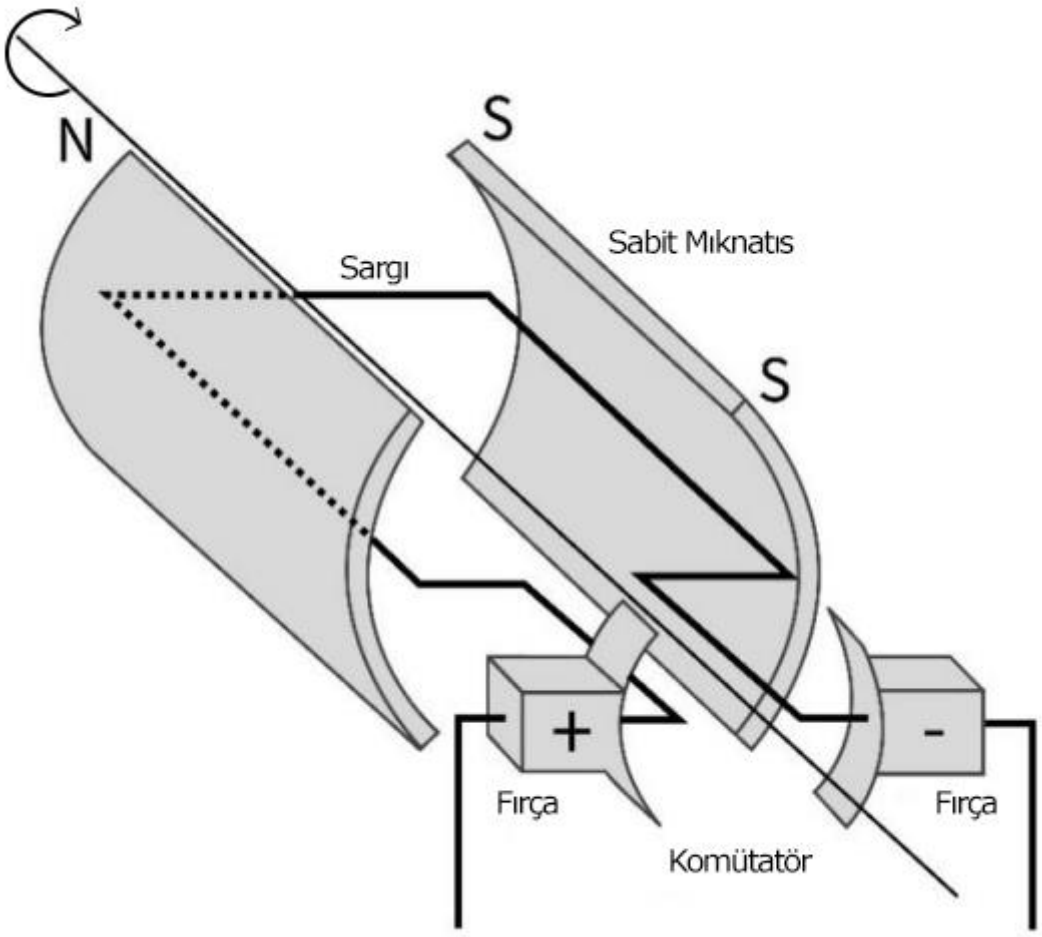
2.2 Fırçasız DC Motorlar

Fırçasız DC motorlar, Fırçalı DC motorların aksine fırçalar yerine statorun oluşturduğu alanı elektronik olarak değiştiren motorlardır. Fırçasız DC motorların gelişim süreci yarı iletken teknolojisinin ilerlemesi ile hız kazanmıştır. Bu tip motorlarda mekanik komütasyon yerine elektronik sürücü devreleri kullanılır.

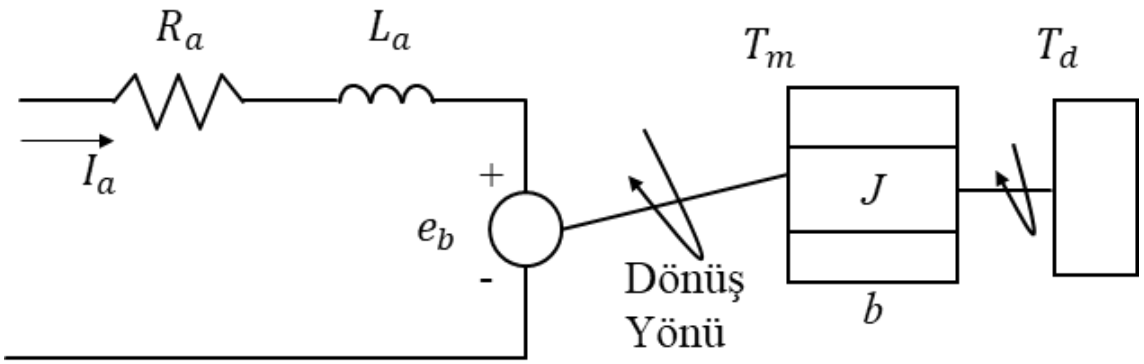
2.3 Fırçalı DC Motor

Bir DC güç kaynağından beslenen, dahili komütatörü bulunan DC motorlara fırçalı DC Motor denir. Fırçalı DC motorlar ticari olarak elektrik enerjisini mekanik enerjiyi çeviren sistemlerde kullanılan ilk motor tipidir. Endüstriyel sistemler, DC dağıtım sistemlerinde yüz yıldan fazladır kullanılmaktadır. Şekil 2.1'de fırçalı DC motorun modeli gösterilmiştir.

Fırçalı DC Motorların hızı çalışma voltajı veya manyetik alanın gücüyle kontrol edilebilir. Fırçalı DC 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 uygulama da artık fırçasız DC motorlar kullanılmaktadır.



Şekil 2.1 Fırçalı DC Motor Modeli (İnt. Kyn.2).



Şekil 2.2 DC Motor Modeli.

Şekil 2.2'de gösterilen DC Motor modeli elektriksel ve mekanik olarak iki kısımda incelenir.

Elektriksel kısmın matematiksel modeli çıkarılırken Kirchhoff'un Gerilimler Kanunu'ndan yararlanılır (İnt. Kyn. 3).

Kirchhoff'a göre kapalı bir elektrik devresinde harcanan gerilimlerin toplamı, kaynak gerilimlerinin toplamına eşittir.

$$V_a = R_a i_a + L_a \frac{di}{dt} + e_b \tag{2.1}$$

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 eden bir iletkende, hareket hızı ile ters orantılı büyüklükte ters indüklenme voltajı oluşur.

$$e_b = K_b w \tag{2.2}$$

Manyetik alan içinde iken üzerinden akım geçen bir iletkene, akımın büyüklüğü ile orantılı bir kuvvet uygulanır.

$$T_m = K_m i_a \tag{2.3}$$

T_m = Motorun oluşturduğu tork

Mekanik kısmın matematiksel modeli çıkarılırken Newton'un İkinci Hareket Yasası'ndan yararlanılır.

$$T_m - T_d = J \frac{dw}{dt} + bw \tag{2.4}$$

T_d = Dış bozucu yük torku

w = Rotorun açısal hızı

J = Rotorun eylemsizlik momenti

b = Sürtünme katsayısı

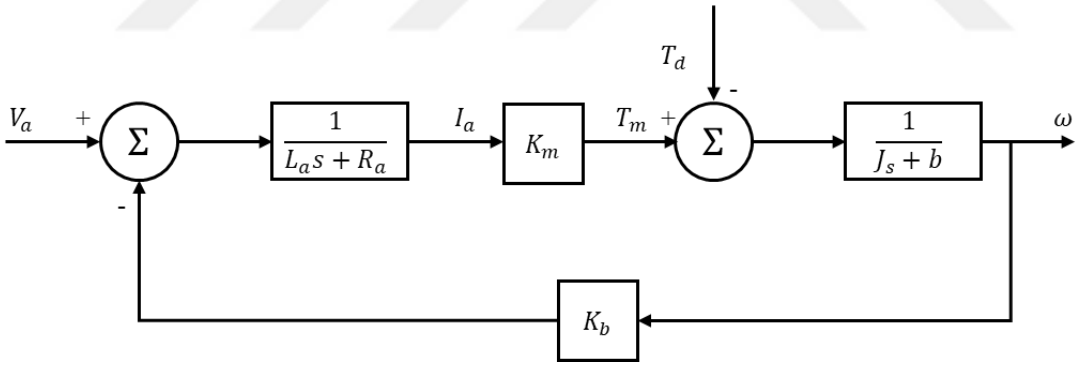
Eşitlik (2.4)'ün transfer fonksiyonu bulunduğunda eşitlik (2.5) elde edilir.

$$\frac{w(s)}{T_m(s) - T_d(s)} = \frac{K_m}{1 + K_m J s} \tag{2.5}$$

Eşitlik (2.1)'in transfer fonksiyonu bulunduğunda eşitlik (2.6) elde edilir.

$$\frac{i_a(s)}{V_a(s) - e_b(s)} = \frac{1}{R_a + L_a s} \tag{2.6}$$

Eşitlik (2.5) ve (2.6)'dan yararlanarak DC Motor için elde edilen blok şeması şekilde gösterilmiştir.



Şekil 2.3 DC Motor Blok Şeması.

Bölüm 4'te uygulanan kontrol uygulamaları Şekil 2.3 dikkate alınarak hazırlanmıştır.

2.4 Dış Bozucular

Fiziksel sistemler çalışırken gürültü veya dış işaretlerin etkisi altında kalabilir. Bu tip etkilere dış bozucular denir. Kontrol sistemlerinin dış bozuculara karşı duyarsız olması istenir. Örnek vermek gerekirse elektrik motorlarındaki fırça veya komütatör gürültüleri

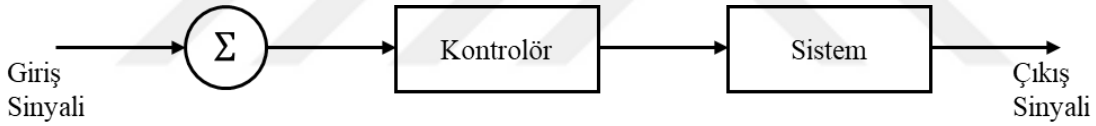
gösterilebilir. Kontrol sistemlerindeki geri besleme sinyalleri dış bozucuların etkisinin azaltılmasında önemli bir yer tutmaktadır. Geri besleme sinyalinden Bölüm 2.5'de bahsedilecektir.

2.5 PID Kontrol

Cihazların, mekanizmaların veya sistemlerin davranışlarını kontrol eden sistemlere kontrol sistemi denir. Bir kontrol sisteminin amacı, kontrol elemanları ile girişleri kullanarak çıkışları bir şekilde kontrol etmektir. Kontrol sistemleri, Açık Çevrimli (geri beslemesiz) ve Kapalı Çevrimli (geri beslemeli) olarak iki bölümde incelenir (İnt. Kyn. 4).

2.5.1 Açık Çevrim Kontrol Sistemi

Açık çevrim kontrol sistemleri kritik davranış sergilemeyecek sistemler için kullanılır.

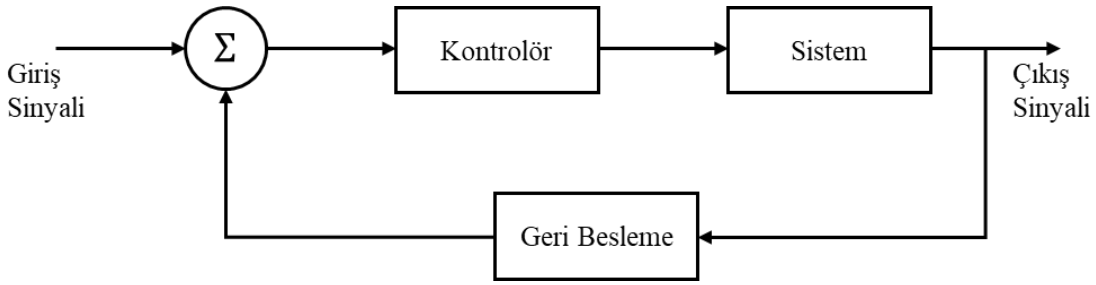


Şekil 2.4 Açık Çevrim Kontrol Sistemi Blok Şeması.

Açık çevrim kontrol sistemleri basit ve ekonomik olması nedeni ile karmaşık olmayan uygulamalarda sıklıkla kullanılır.

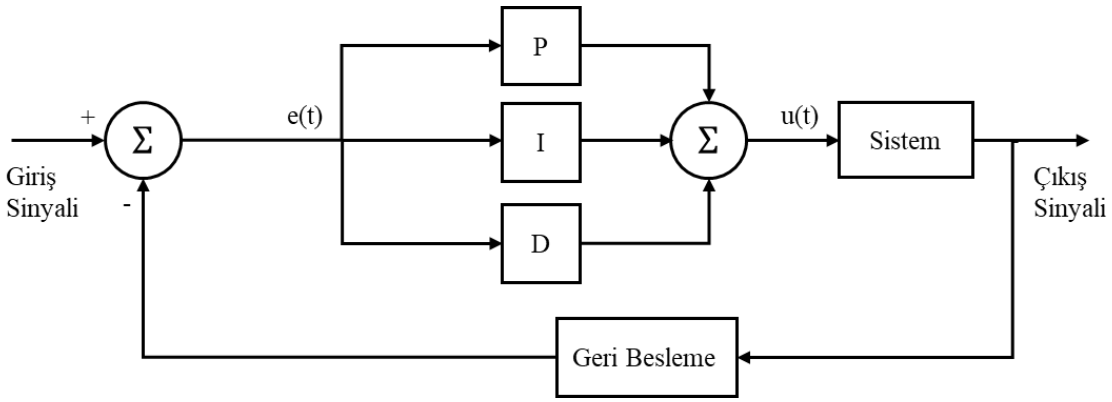
2.5.2 Kapalı Çevrim Kontrol Sistemi

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.



Şekil 2.5 Kapalı Çevrim Kontrol Sistemi 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.



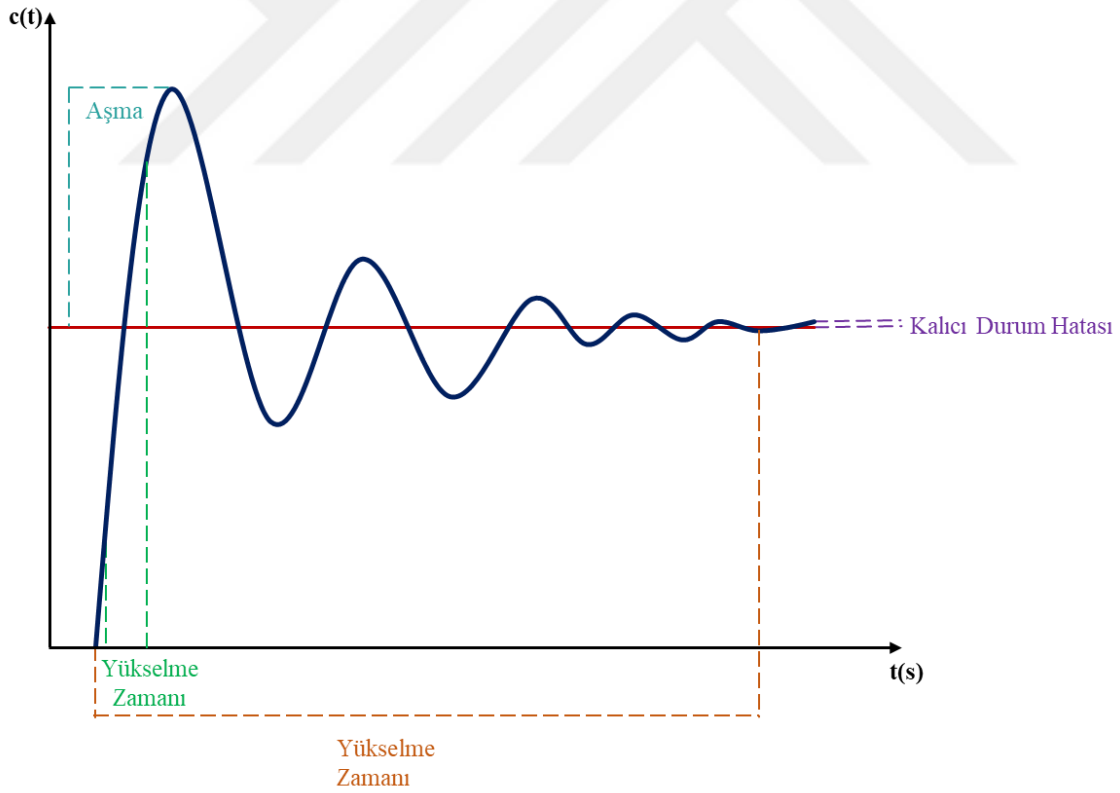
Şekil 2.6 PID Kontrol Blok Şeması.

PID kontrolde, geri besleme sinyali ile giriş sinyali karşılaştırılıp bir hata değeri hesaplanır. Bu hata PID kontrolörde bir katsayı ile çarpılıp, türev ve integrali alındıktan sonra çıkışa gönderilir. Hata değeri minimuma ulaşıncaya kadar süreç tekrarlanır.

Katsayılar, bazı karakteristik davranışlar göz önünde bulunarak belirlenir. Oransal (P) terimdeki katsayı (K_p), hatayı çarparak düşürmeyi hedefler. Osilasyon görülme ihtimalini arttırdığından salınımları önlemek için K_p değeri çok yüksek seçilmez. İntegral (I) terimde, hatanın alanı hesaplanır. Her periyottaki hata, katsayı (K_i) ile çarpılarak toplanır. Türev (D) terimi değişimle ilgilidir. Sistemdeki değişimi fark ederek, istenen değerde tutmak için yavaşlatır. Hata değerinde herhangi bir değişim söz konusu değil ise türev sıfır olacağından bir etkide bulunmaz (Golnaraghi ve Kuo 2017).

2.6 Performans Metrikleri

PID kontrolör tasarımında parametrelerin her biri sistem çıktısına etki etmektedir. Sistemin çıkışını en optimal ve verimli elde etmek için göz önünde bulundurulması gereken bazı metrikler bulunmaktadır.



Şekil 2.7 PID Kontrolör Performans Metrikleri.

Sistem başlatıldıktan sonra belirlenen referans değerin %85-90'ına ulaşma süresine yükselme süresi denir. Yükselme parametresini ayarlamak için Çizelge 2.1'de görüleceği üzere Oransal(P) kontrolörün katsayısını değiştirmek gerekir. Diğer parametrelerin yüksele süresi üzerinde gözle görülür bir etkisi yoktur. Aşma parametresini Oransal (P) ve Integral (I) katsayıları artırıcı yönde etki yaparken Türevsel (D) katsayı azaltıcı bir etki yapar. Yerleşme zamanı, salınımların azaltılarak, istenen değere kalıcı olarak getirilmesine kadar geçen süreye denir. Yerleşme zamanı üzerinde Oransal katsayı az etkili bir artış yaparken Integral katsayı artırıcı, Türevsel katsayı ise azaltıcı bir etki yapar. Kalıcı durum süresi üzerine ise Türevsel katsayının bir etkisi yok iken Oransal ve Integral katsayıları azaltıcı bir etki yapar.

Çizelge 2.1 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

2.7 Ziegler-Nichols Metodu

PID kontrol tasarımında kontrolörleri tasarlamada en çok kullanılan yöntemlerden birisi de Ziegler – Nichols metodudur. John Ziegler ve Nathaniel Nichols, 1942 yılında PID kontrolör tasarımı için bu yöntemi ortaya koydular. Yöntemde integral zaman sabiti ve türev zaman sabiti sıfırlanır. Kontrol edici kazancı (K_c) sistem çıkışı sürekli salınım yapana kadar arttırılır. Sürekli salınım gözlendiğinde salınım periyodu (T_u) ve kontrol edici son kazancı (K_u) kaydedilir. Kaydedilen bu değerlerden yararlanılarak, Ziegler-Nichols metodunda kullanılan tablodan kullanılacak kontrolör için değerler belirlenir. Kontrolör tipine göre katsayıları belirlemek için Çizelge 2.2'den yararlanılır.

Çizelge 2.2 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.

2.8 Parçacık Sürü Optimizasyonu

Belirli şartlar altında, imkân dahilindeki seçenekler arasından en iyisini seçme işine optimizasyon denir.

Belirli sınırlar içinde parametre değerlerinin bulunmasını içeren herhangi bir probleme ise optimizasyon problemi denir.

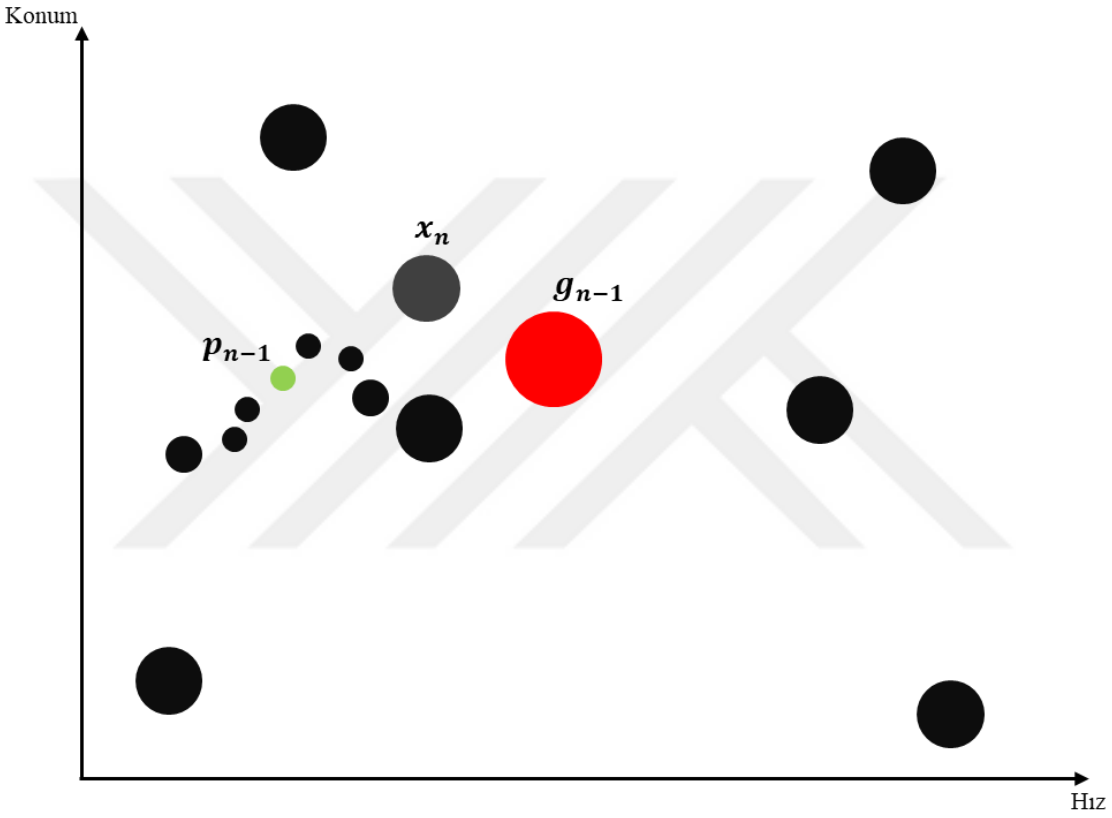
Doğada tek başına iş yapmakta zorlanan, toplu olarak hareket ettiklerinde zeki davranışlar sergileyen canlılar bulunur. Bu topluluğa veya diğer bir deyişle sürüye ait bireyler, kendi tecrübelerinden yararlanmakla kalmayıp sürüdeki en iyi bireyin davranışından da yorumlamalar yaparak ileride karşılaşacakları problemleri çözmede bir araç olarak kullanır. (Akyol ve Alataş 2012).

Parçacık Sürü Optimizasyonu (PSO), kuş, arı gibi sürü halinde hareket hayvan türlerinin beslenme gibi temel ihtiyaçlarını bulmaya giderken sergiledikleri hareketlerden esinlenilmiştir. Sürüdeki hareketli bireylerin, diğer bireyleri etkilediğinin ve sürünün amacına daha hızlı ve kolay ulaştığı gözlemlenmiştir. 1995 yılında Elektrik Mühendisi Russell C. Eberhart ve Sosyal Psikolog James Kennedy tarafından yayınlanmıştır. PSO, genetik algoritmalar ve diğer evrimsel hesaplama teknikleri ile benzerlik gösterse de görece daha basit olması, algoritmayı uygulaması daha kolay bir hale getirmektedir.

PSO algoritması karmaşık arama uzaylarında optimal çözümler bulma konusundaki

başarısından dolayı kontrol ve optimizasyon alanındaki çalışmalarda sıklıkla kullanılmaktadır.

PSO algoritmasının uygulamalardaki kullanımlarda temel mantığı, arama uzayında rastgele başlatılan noktaların, o ana kadar en başarılı olan bölgeye doğru yönlendirilmesini sağlamaktır (Gökçe vd. 2021).



Şekil 2.8 PSO Çalışma Mekanizması.

Şekil 2.8'de arama uzayında rastgele başlatılmış parçacıkların temsili görüntüsü verilmiştir. Her parçacığın konumu, optimize edilmek istenen parametreleri içeren bir vektördür. Başlangıçtan itibaren her iterasyonda güncellenip belirli bir süre sonunda makul bir performansa yakınsaması beklenir.

Her parçacığın bir hızı vardır. Bu hız anlık ve bir sonraki konumu belirler. Her iterasyonda parçacığın anlık hızı ve bir önceki iterasyonda hesaplanan hızı toplanarak yeni konum elde edilir.

$$x_n = x_{n-1} + v_n \tag{2.7}$$

x_n : Parçacığın konumu

v_n : Parçacığın hızı

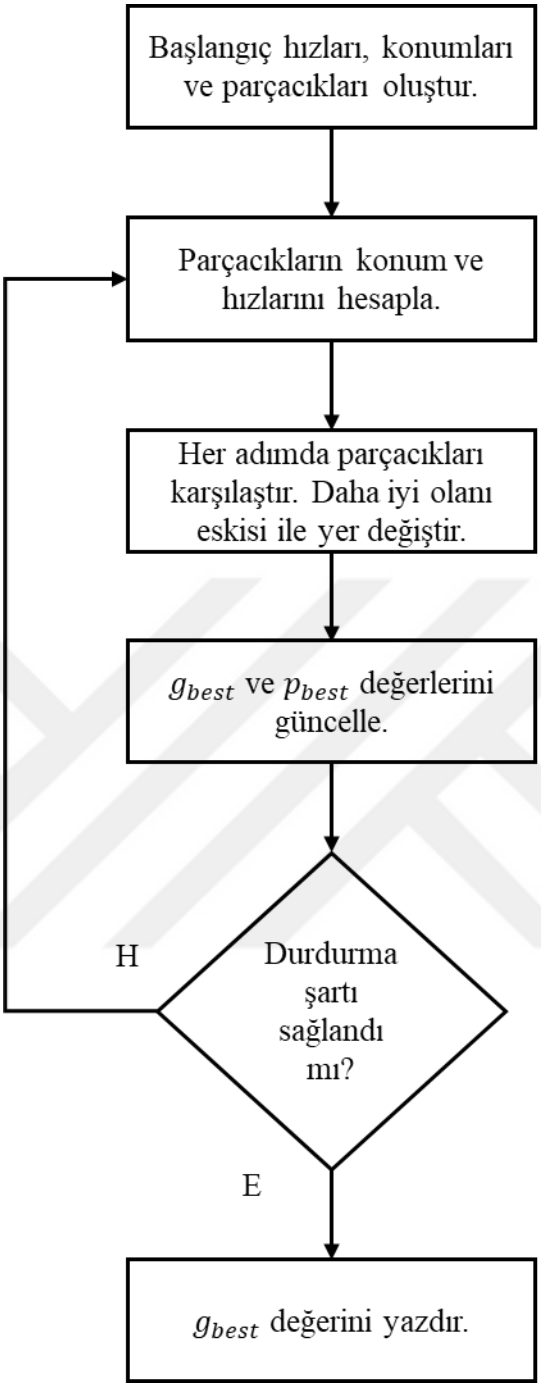
v_n Değeri hesaplanırken üç durum göz önünde bulundurulmalıdır:

- 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}

$$v_n = c_1 v_{n-1} + c_2\left(x_{n-1} - p_{(n-1)}\right) + c_3(x_{n-1} - g_{n-1}) \tag{2.8}$$

Parçacığın yeni hızı, c_1, c_2, c_3 belirli katsayılar olmak üzere eşitlik (2.8) ile hesaplanır.

Şekil 2.9'da PSO için temel bir akış şeması verilmiştir. Kodlama adımları bu akış şeması dikkate alınarak oluşturulmuştur.



Şekil 2.9 PSO Akış Şeması.

Zahir vd. (2018) çalışmalarında Fırçalı DC Motor hız kontrolünde PID parametrelerini belirlerken Genetik Algoritma (GA) kullanmış ve Ziegler-Nichols yönteminden daha iyi bir başarı gösterdiğini ortaya koymuştur.

Yazgan vd. (2019) çalışmalarında Genetik Algoritma (GA) ve PSO ile PID

parametrelerini optimize edip 5 kriterde karşılaştırmasını yapmıştır. Fırçasız DC Motor üzerinde gerçekleştirilen testlerde PSO'nun GA'dan daha iyi sonuç verdiğini tespit etmiştir.

Song vd. (2020) çalışmalarında Fırçasız DC Motor için Bulanık (Fuzzy) PI kontrolör parametrelerinin optimizasyonu için PSO ve Yerçekimsel Arama Algoritmasını birleştiren bir yaklaşım kullanmıştır. Bu iki yöntemi birleştirerek optimizasyon için başlangıçta kullanılan rastgele parametrelerin işlemlerinin daha hızlı sonuca ulaştığını gözlemlemiştir.

Ibrahim vd. (2014) çalışmalarında PID parametrelerinin belirlenmesinde PSO ve Bakteriyel Yiyecek Arama (Bacterial Foraging) yöntemlerini kullanıp, yöntemlerin karşılaştırmasını yapmıştır. Çalışma Matlab ortamında simülasyon olarak gerçekleştirilmiştir. Kriterler göz önünde bulundurulduğunda gözler görülür bir iyileştirme olduğunu fark etmiştir.

Mohamadwasel ve Bayat (2019) çalışmalarında PSO destekli Bulanık Mantık algoritması kullanarak PD kontrolör parametrelerini optimize etmiştir.

Qi vd. (2019) makalesinde stokastik gecikmelere maruz bırakılan CAN tabanlı bir dc motorun PID parametrelerini PSO kullanarak optimize etmek için önceki verilere dayanarak simülasyonunu gerçekleştirmiş ve deneysel sonuçlar ile algoritmanın sonuçlarını doğrulamıştır.

Weerasooriya ve El-Sharkawi (1991) geri yayılım tabanlı yapay sinir ağları kullanarak DC Motorun lineer olmayan dinamiğini kontrol etmeyi amaçlamıştır.

Idir vd. (2018) çalışmada DC Motorun hız kontrolünde PID ve Kesirli-düzen PID (fractional order PID-FOPID) kontrolör parametrelerini belirlemek için PSO ve Diferansiyel Evrim algoritmalarını kullanmıştır. Performans fonksiyonu olarak integral-zaman mutlak hata (integral time absolute error-ITAE) kullanılmış ve bu değerin minimize edilmesi amaçlanmıştır.

Çıkan sonuçlar Diferansiyel Evrim tabanlı FOPID tasarımı yaklaşımının referans takibi, hata miktarı ve gürültü azaltma metriklerinde daha iyi performans sergilediği gözlemlenmiştir.

Khalilpour vd. (2011) yılında yaptıkları çalışmada parametrelerin optimizasyonu konusunda İstilacı Ot Optimizasyonu (IOO) ile PSO yöntemlerinden hangisinin daha verimli olduğunu bulmak için karşılaştırma yapmıştır. Uygulama Matlab ortamında gerçekleştirilmiştir.

Nasri vd. (2007) yılında yaptıkları çalışmada lineer fırçasız DC motor hız kontrolünde PID kontrolör parametrelerini belirlemek için PSO algoritması kullanmıştır. DC Motor modellemesini Simulink üzerinde, PSO algoritmasını ise Matlab üzerinde gerçekleştirmiştir. Çıkan sonuçlar Genetik Algoritma (GA) ve Lineer İkinci Dereceden Düzenleyici yöntemleri ile karşılaştırılmıştır. PSO'nun yükselme süresi, aşma gibi sistem tepkilerini iyileştirmede daha etkili olduğunu ortaya koymuştur.

3. MATERYAL VE METOT

Bu çalışmada gerçek DC motor parametrelerinden yararlanıldığından Bölüm 1'de bahsedilmiştir. Simülasyonda DC motor parametreleri olarak Çizelge 3.1'de verilen, Moog firmasının C23-L33 serisi DC Motorlarının parametreleri kullanılmıştır.

Çizelge 3.1 DC Motor Parametreleri (İnt. Kyn. 5).

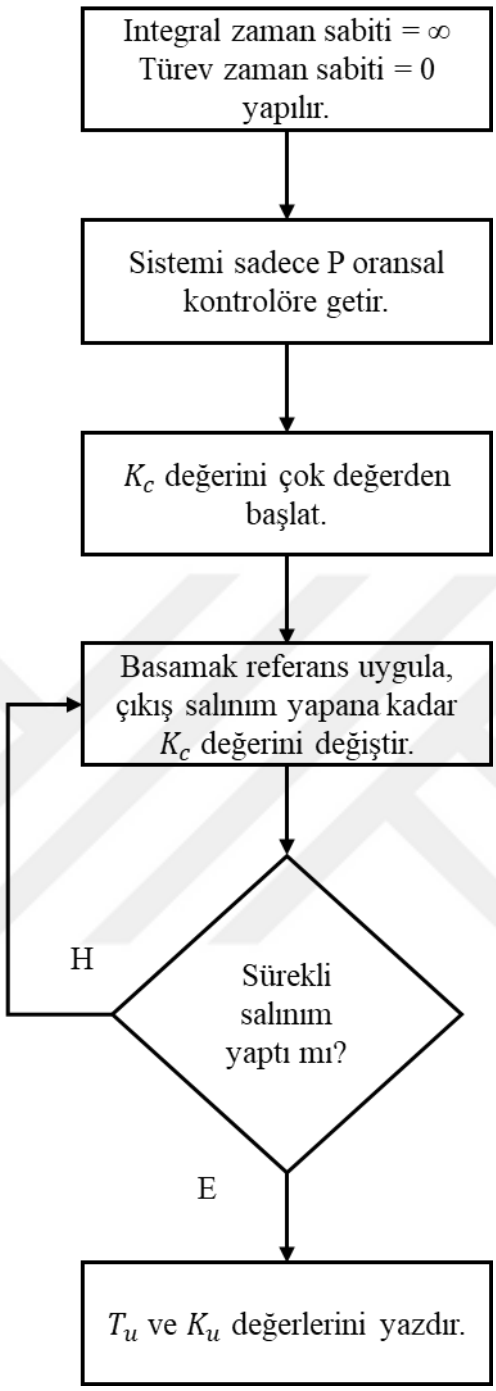
Rotor Eylemsizlik	Sürtünme Torku	Viskoz	Motor Tork Sabiti	EMK Sabiti
155,4e-7	0,2	1e-5	0,0187	0,0191

Çalışmada simülasyonlar için Anaconda-Spyder IDE'si ve Python programlama dili kullanılmıştır. Veri bilimi, optimizasyon, makine öğrenmesi gibi birçok alanda kullanılan Spyder, içinde barındırdığı araçlar ve açık kaynak olması sebebiyle çok tercih edilmektedir. Python ile veriler üzerinde çalışmak isteyen birçok bilim insanı Spyder kullanmaktadır. Günümüzde veri bilimi, optimizasyon, yapay zekâ gibi alanlarda çalışan bilim insanlarının büyük bir çoğunluğu Python kullanmaktadır.

Python programlama dili, nesne tabanlı, yorumlamalı ve etkileşimli yüksek seviyeli bir dildir. Açık kaynak bir dil olduğundan geliştirilmiş projelere ve kütüphanelere ulaşımı kolaydır. Gömülü sistemlerden internet tabanlı uygulamalara, mobil uygulamalardan masaüstü uygulamalarına kadar neredeyse her alanda Python ile proje geliştirilebilmektedir.

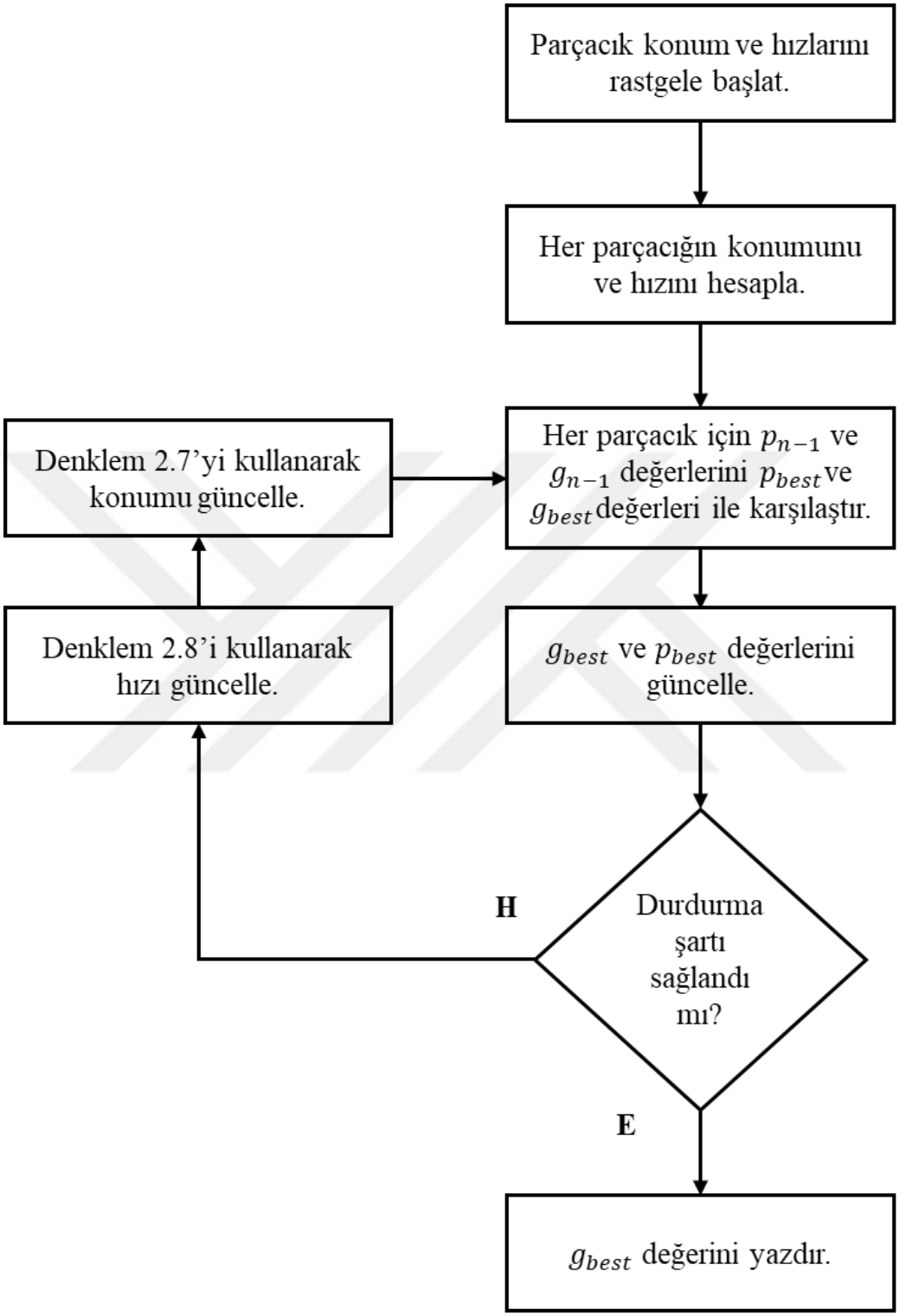
Çalışmada ZN ile PID parametreleri belirlenirken Şekil 3.1'de uygulama adımları verilen klasik ZN yönteminde kullanılan yöntem kullanılmıştır.

Başlangıçta I ve D kontrolörlerinin zaman sabitleri sıfırlanır ve sistem oransal kontrole alınır. P oransal kontrolörünün K_c kazanç katsayısı küçük bir değerden başlanarak sistem çıktısı salınım yapana kadar arttırılmaya devam edilir. Yöntem için kullanılan yol, şekildeki akış şemasında gösterilmiştir. Akış şemasında belirtilen değerlerin ve yöntemin açıklamaları Bölüm 2.7'de anlatılmıştır.



Şekil 3.1 Ziegler-Nichols Yöntemi için kullanılan akış şeması.

PSO için Bölüm 2.8'de tanımlanan eşitlikler kullanılarak hız ve konum güncellemeleri yapılmış ve uygunluk değerleri karşılaştırılmıştır. Çalışmada PSO hesaplamaları için kullanılan akış şeması Şekil 3.2'de gösterilmiştir.



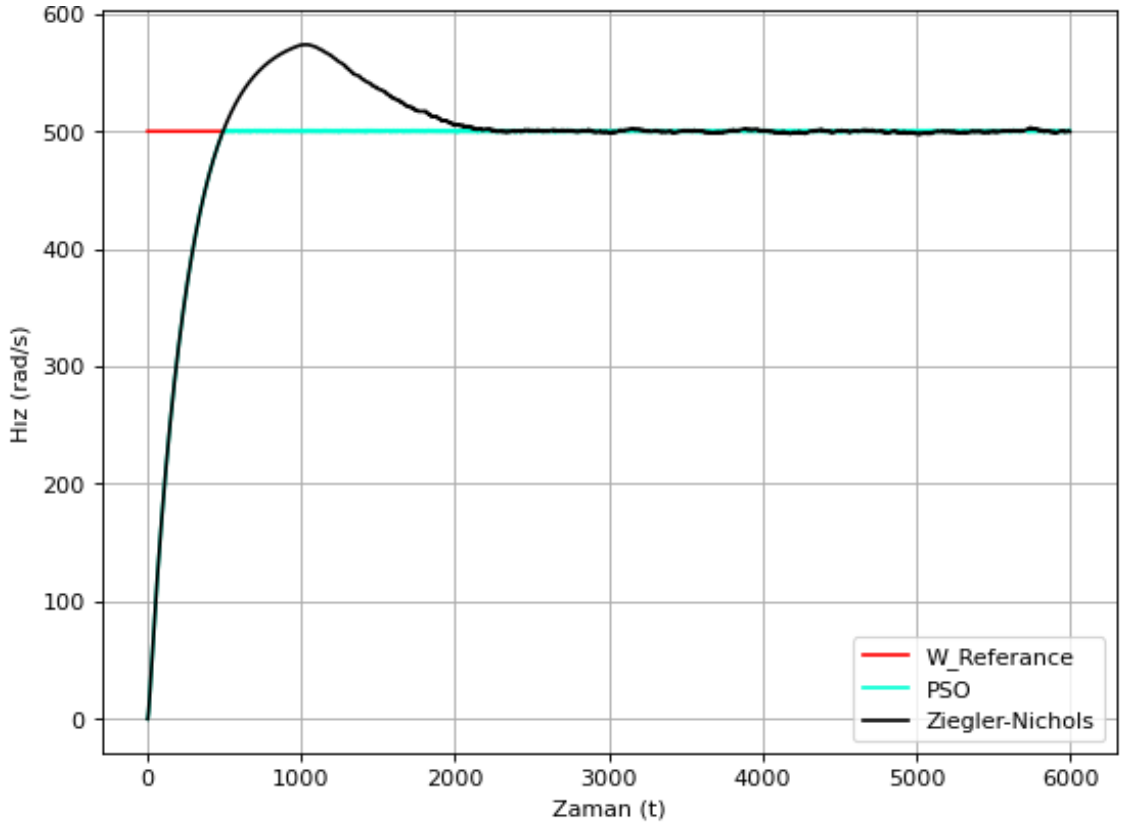
Şekil 3.2 PSO hesaplamaları için kullanılan akış şeması.

Çalışmada kesikli zaman türevleri yamuk kuralına göre hesaplanmıştır.

Performans ölçüm değeri olarak mutlak toplam hata fonksiyonu kullanılmıştır. Her adımda referans değer ile gerçek değer arasındaki farkın mutlak değerleri alınıp toplanmış ve 10^6 değerine bölünerek performans değeri hesaplanmıştır. 10^6 ifadesinin seçilmesinin nedeni maksimum hata değerinin bu değerden küçük çıkmasıdır. Dolayısıyla hata değeri hiçbir zaman sıfır veya negatif çıkmamaktadır. Performans değerleri çizelgeler halinde Bölüm 4'te sunulmuştur.

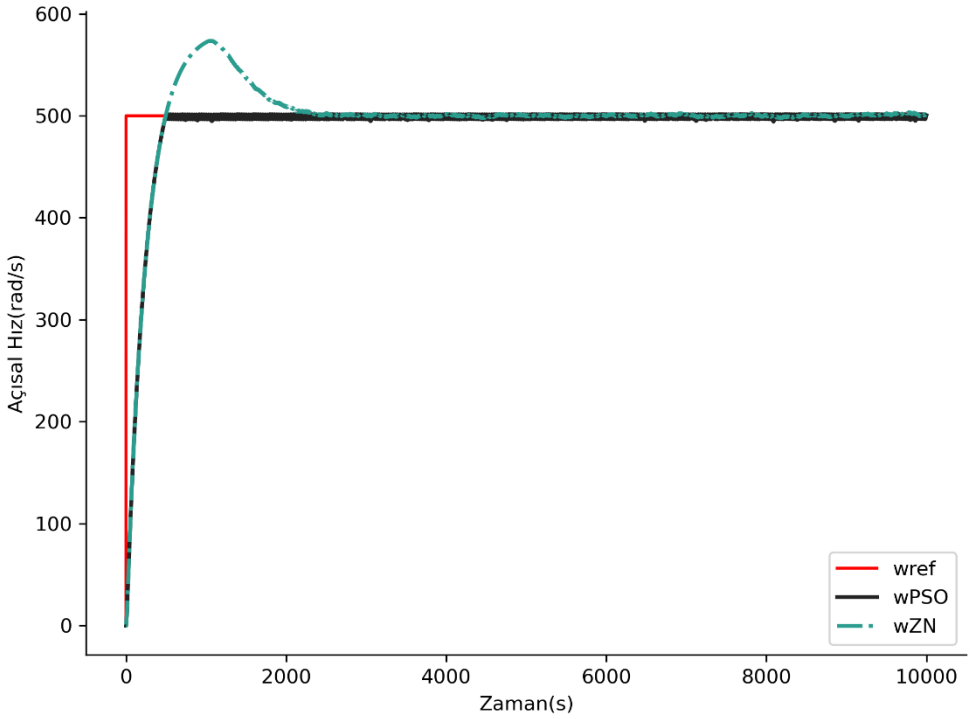
4. BULGULAR

Performans değerlendirmesi yapmak için motor farklı yükler altında test edilmiştir. Öncelikle boşta dönen, yüksüz motor için deneyler yapılmış ve daha sonra farklı yük çeşitleri altında, farklı genliklerde deneyler yapılmıştır.



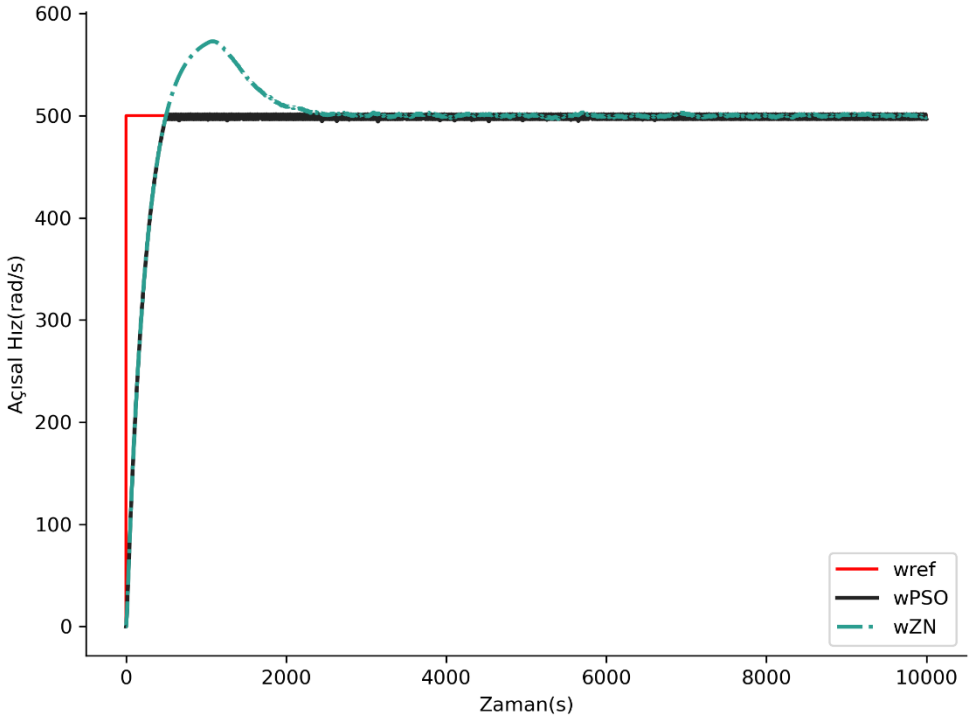
Şekil 4.1 Boşta dönen motorun basamak referans altında sistem tepkisi.

Motora öncelikle adım basamak referansı uygulanmış ve sistem tepkisi gözlemlenmiştir. Şekil 4.1'de PSO parametrelerinin referansı hızlı bir şekilde takip ettiği görülmektedir. Yükselme zamanı performansı açısından bakıldığında iki parametrede aynı sürede yükselmiş fakat ZN parametrelerinde aşma gözlemlenmiştir. Performans değerleri hesaplandığında PSO için 9.496 değeri, ZN için 6.316 değeri bulunmuştur. Performans değerlerine bakıldığında yaklaşık %33,5 fark bulunmaktadır.



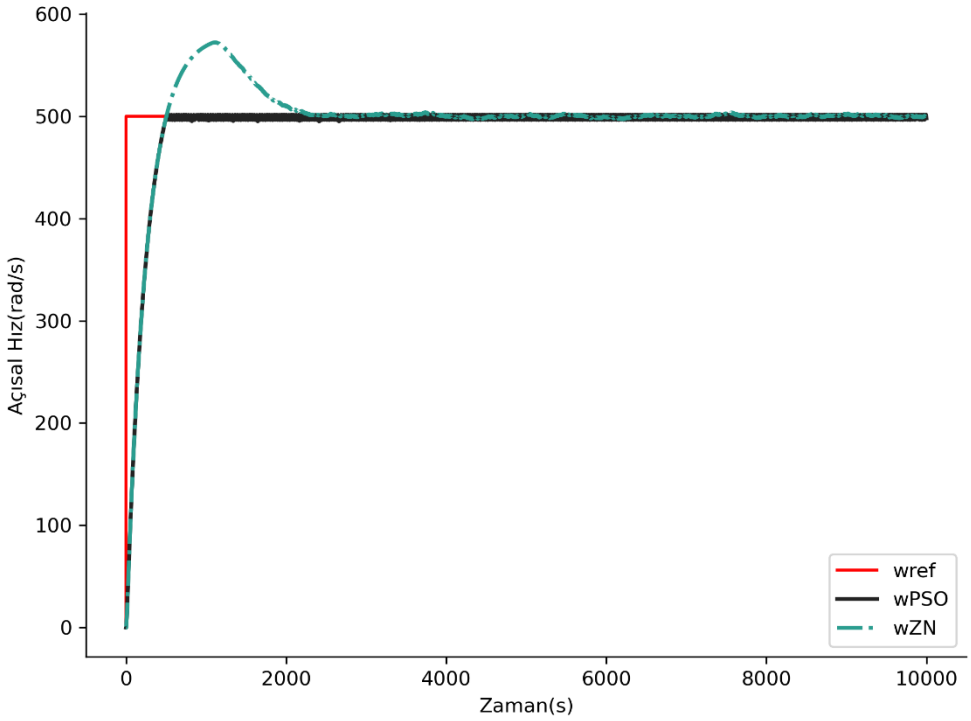
Şekil 4.2 Basamak referansta genliği 0,001 olan basamak yük uygulanması.

Sistem basamak referans altında basamak yük verilerek test edilmiş ve Şekil 4.2 – Şekil 4.15'te verilmiştir. Şekil 4.2'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,001 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



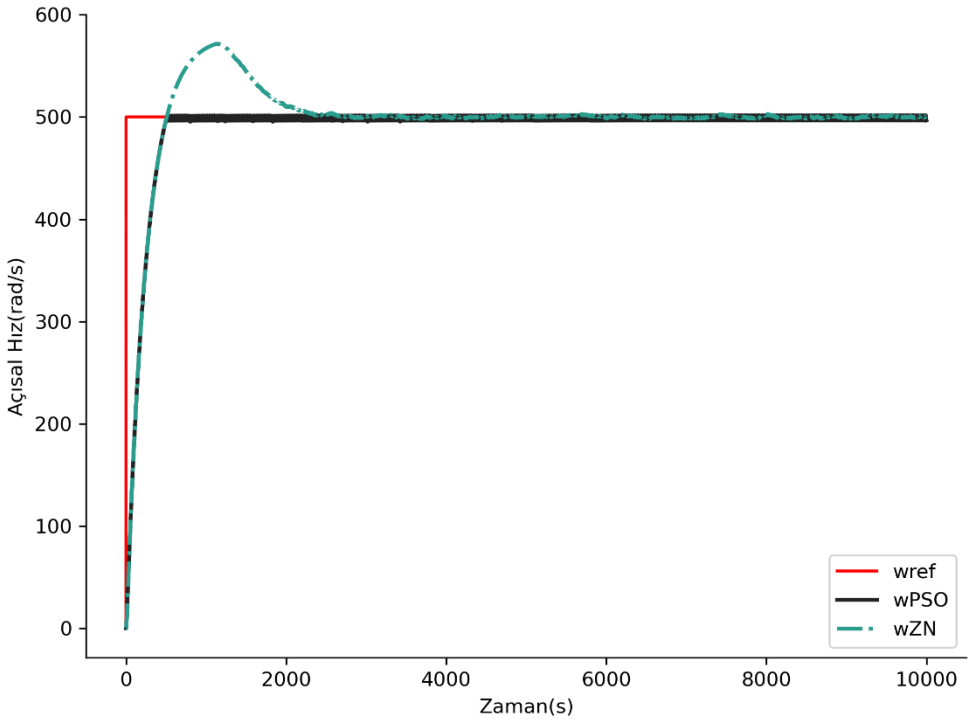
Şekil 4.3 Basamak referansta genliği 0,002 olan basamak yük uygulanması.

Şekil 4.3'te görüldüğü üzere kalıcı durum hatası, yük genliği 0,002 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



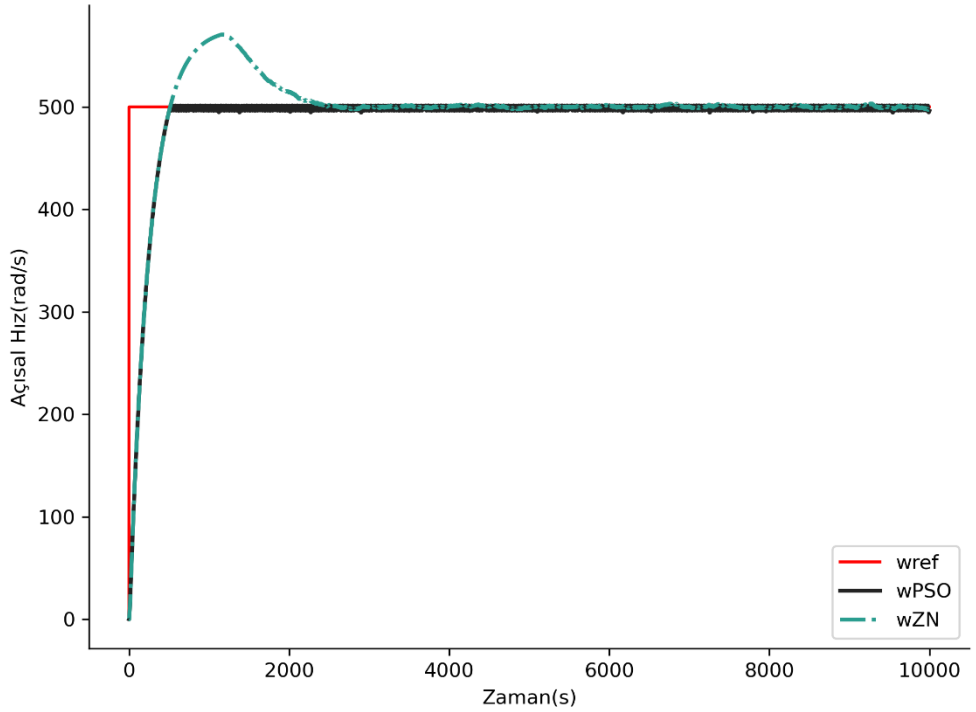
Şekil 4.4 Basamak referansta genliği 0,003 olan basamak yük uygulanması.

Şekil 4.4'te görüldüğü üzere kalıcı durum hatası, yük genliği 0,003 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



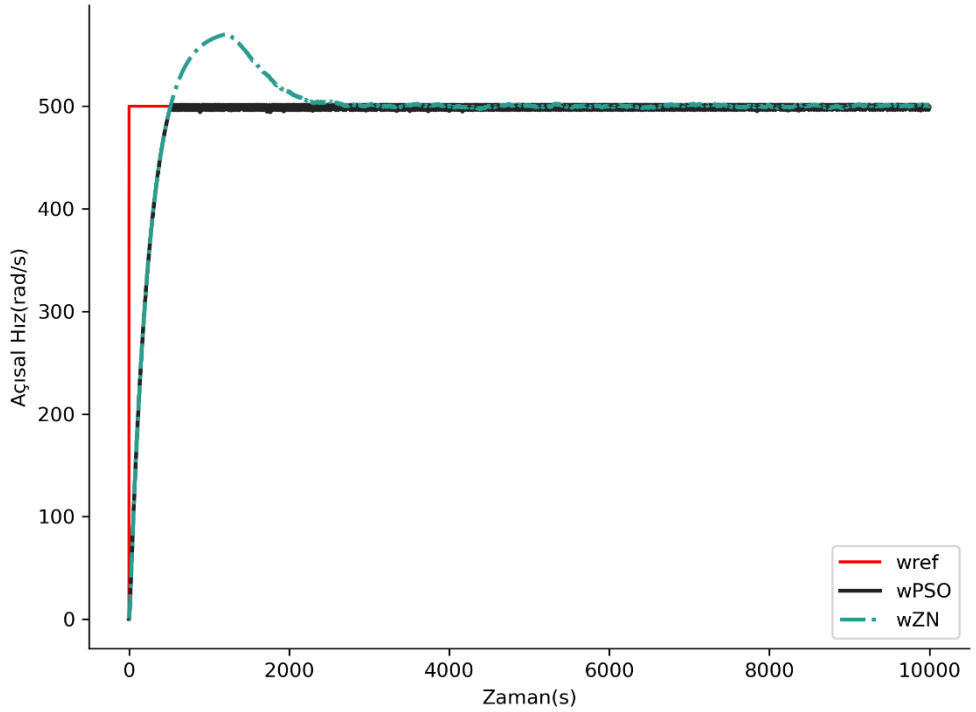
Şekil 4.5 Basamak referansta genliği 0,004 olan basamak yük uygulanması.

Şekil 4.5'te görüldüğü üzere kalıcı durum hatası, yük genliği 0,004 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



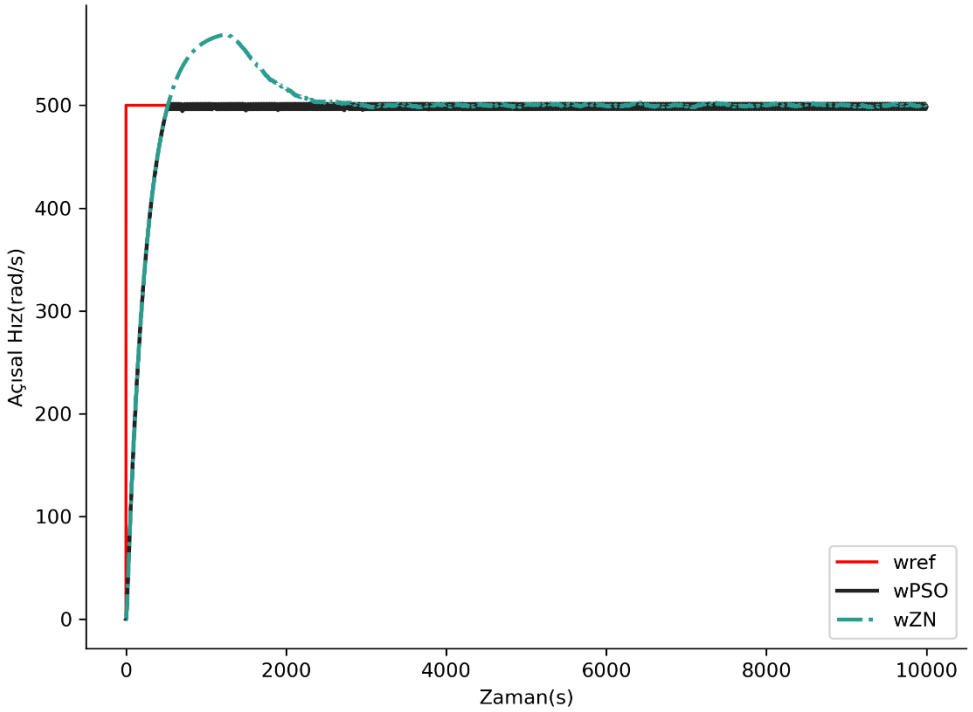
Şekil 4.6 Basamak referansta genliği 0,005 olan basamak yük uygulanması.

Şekil 4.6'da görüldüğü üzere kalıcı durum hatası, yük genliği 0,005 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



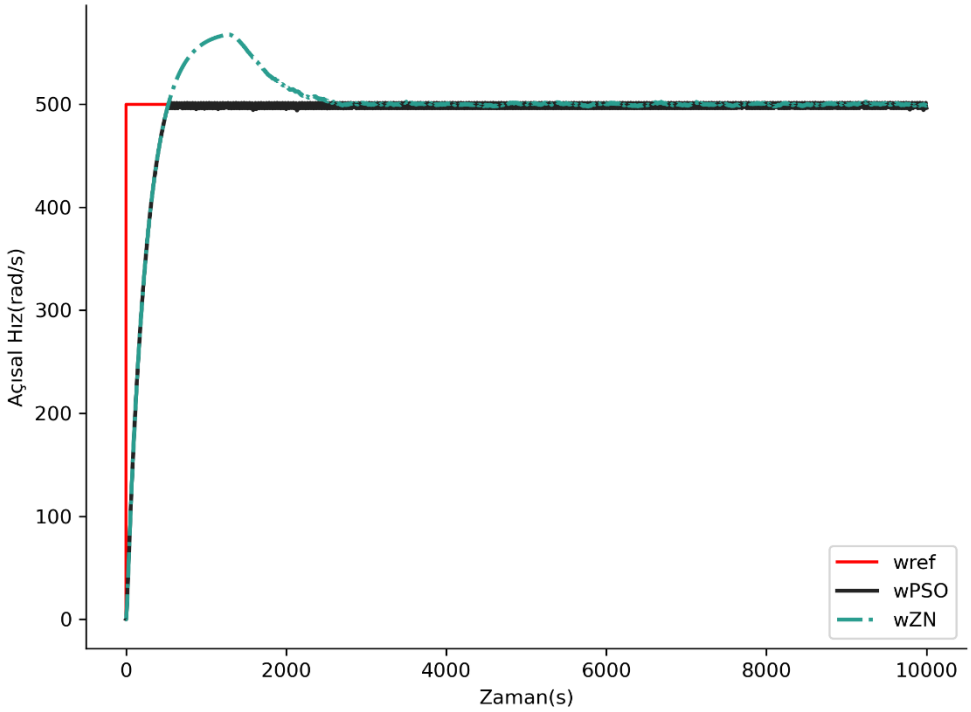
Şekil 4.7 Basamak referansta genliği 0,006 olan basamak yük uygulanması

Şekil 4.7'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,006 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



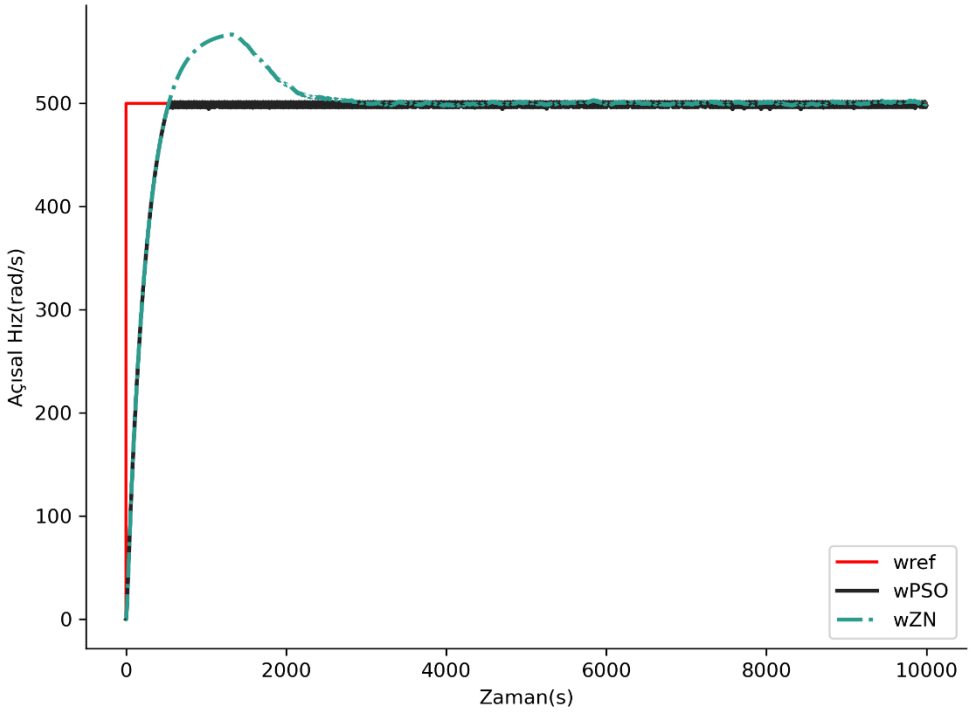
Şekil 4.8 Basamak referansta genliği 0,007 olan basamak yük uygulanması.

Şekil 4.8'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,007 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



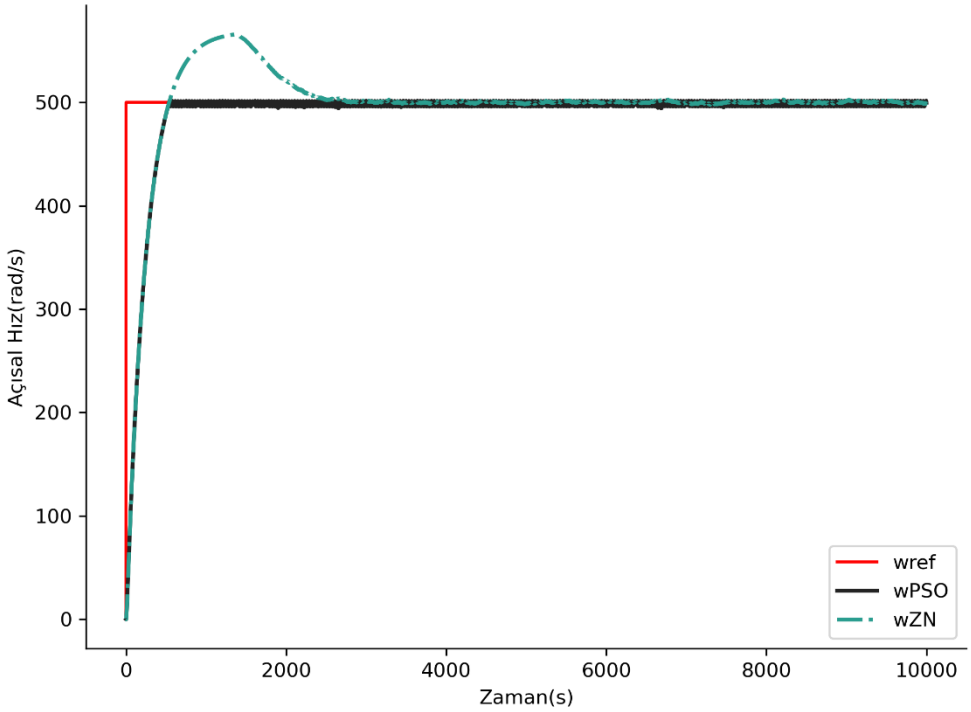
Şekil 4.9 Basamak referansta genliği 0,008 olan basamak yük uygulanması.

Şekil 4.9'da görüldüğü üzere kalıcı durum hatası, yük genliği 0,008 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



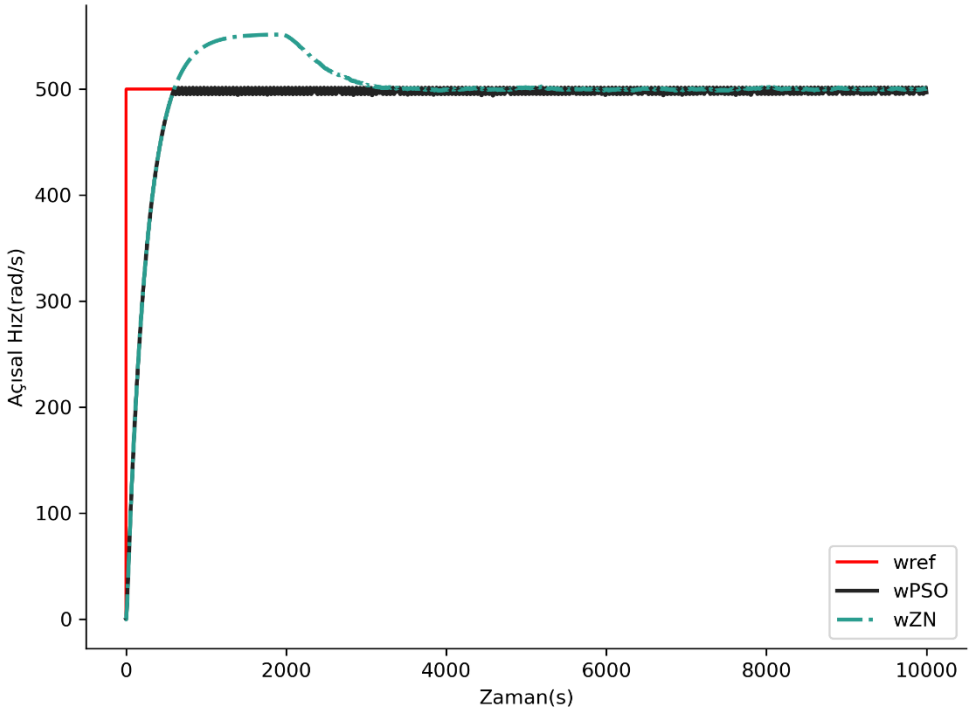
Şekil 4.10 Basamak referansta genliği 0,009 olan basamak yük uygulanması.

Şekil 4.10'da görüldüğü üzere kalıcı durum hatası, yük genliği 0,009 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



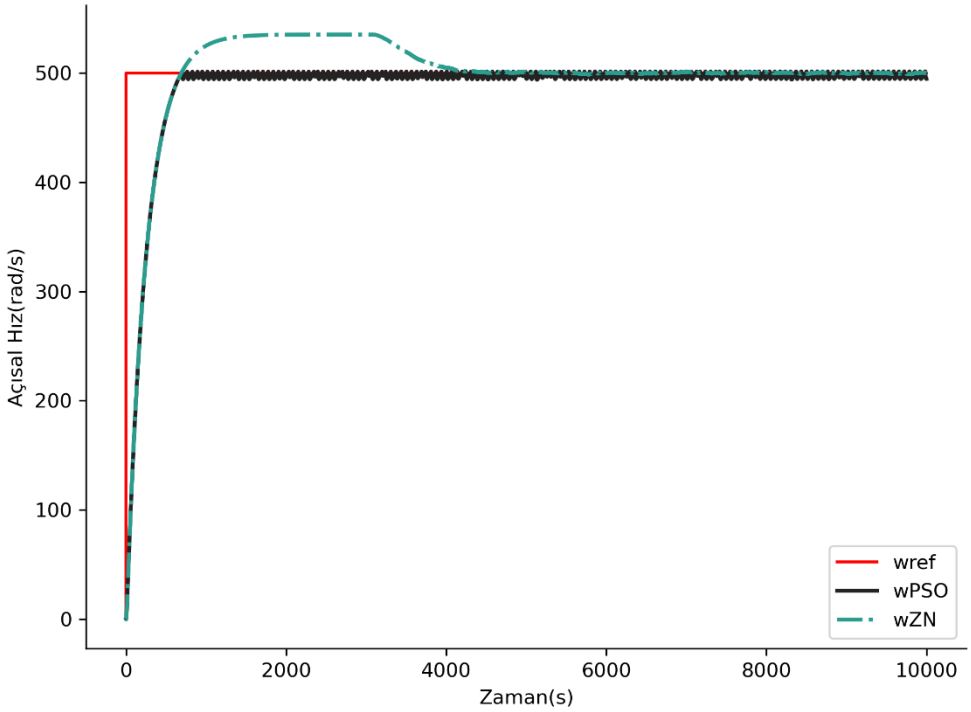
Şekil 4.11 Basamak referansta genliği 0,01 olan basamak yük uygulanması.

Şekil 4.11'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,01 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



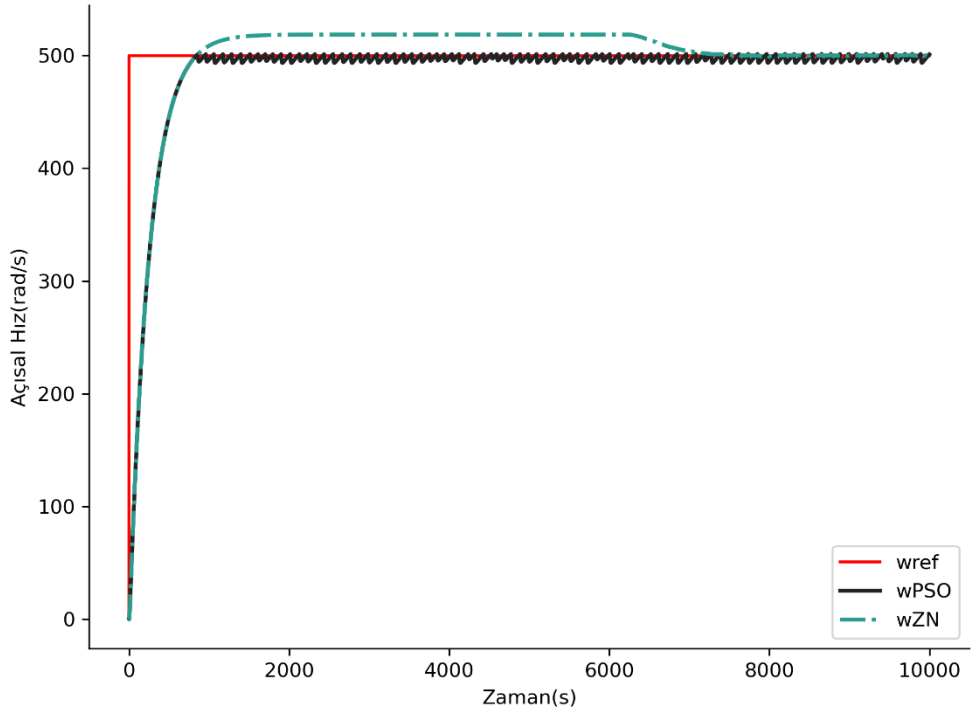
Şekil 4.12 Basamak referansta genliği 0,02 olan basamak yük uygulanması.

Şekil 4.12'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,02 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenmeye devam etmiş ancak aşma değerinde azalma olmuştur. Oturma süresi ZN'da artış göstermiştir. PSO yaklaşımında bu değerlerde gözle görülür bir değişiklik yine gözlemlenmemektedir. Referans değere PSO yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



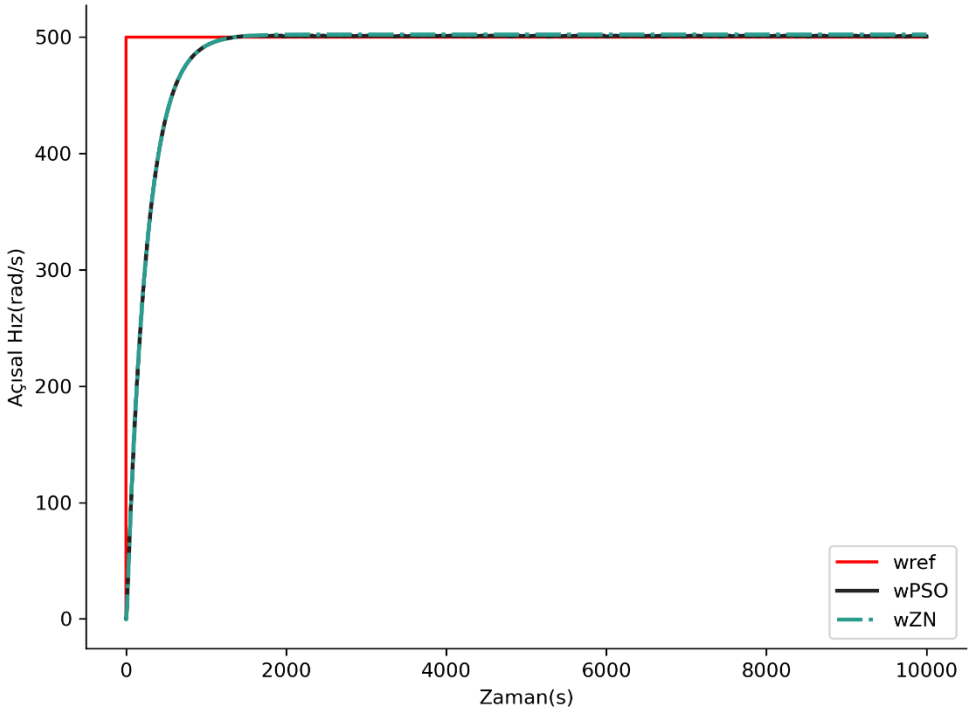
Şekil 4.13 Basamak referansta genliği 0,03 olan basamak yük uygulanması.

Şekil 4.13'te görüldüğü üzere kalıcı durum hatası, yük genliği 0,03 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenmeye devam etmiş ancak aşma değerinde azalma olmuştur. Oturma süresi ZN'da artış göstermiştir. PSO yaklaşımında bu değerlerde gözle görülür bir değişiklik yine gözlemlenmemektedir. Referans değere PSO yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



Şekil 4.14 Basamak referansta genliği 0,04 olan basamak yük uygulanması.

Şekil 4.14'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,04 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenmeye devam etmiş ancak aşma değerinde azalma olmuştur. Oturma süresi ZN'da artış göstermiştir. PSO yaklaşımında bu değerlerde gözle görülür bir değişiklik yine gözlemlenmemektedir. Referans değere PSO yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır.



Şekil 4.15 Basamak referansta genliği 0,05 olan basamak yük uygulanması

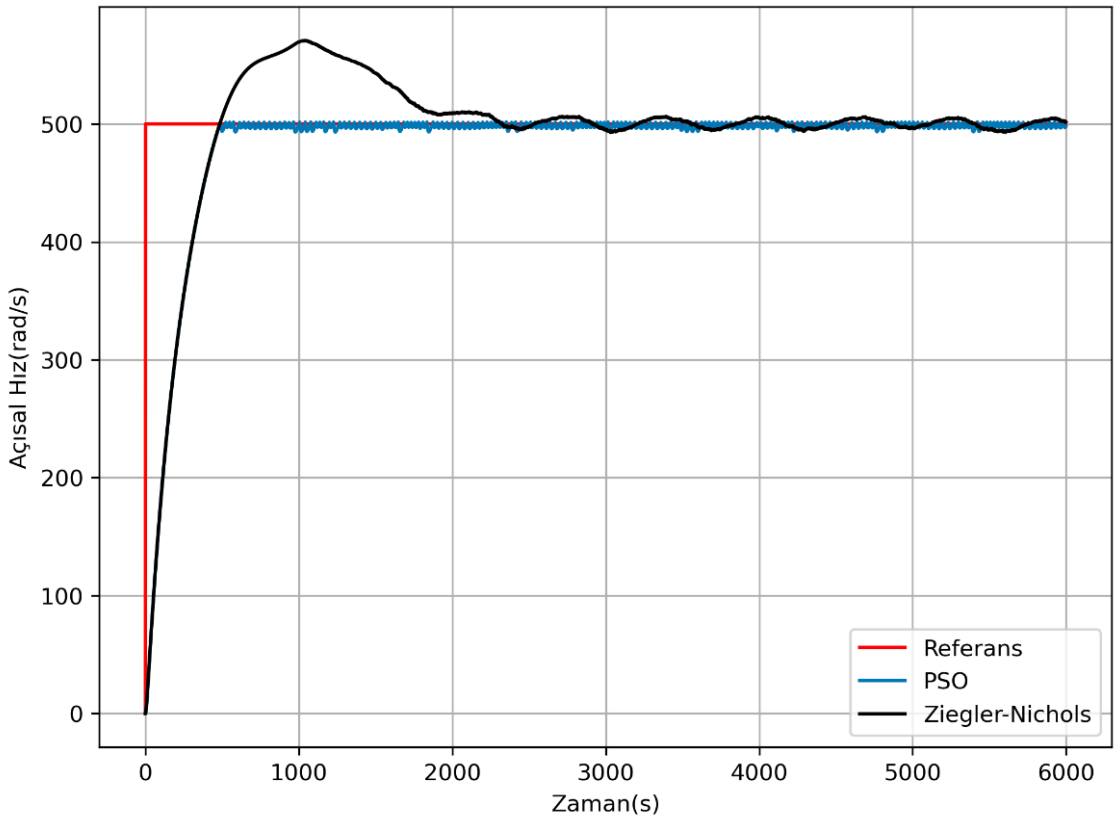
Şekil 4.15'te görüldüğü üzere kalıcı durum hatası, yük genliği 0,05 için neredeyse aynıdır. ZN yaklaşımında aşma bu değerde gözlemlenmemektedir. Oturma süresi bakımından iki yöntemde de yaklaşık aynı sonuçlar elde edilmiştir. Referans değere PSO yaklaşımının ve ZN yaklaşımının neredeyse aynı olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır. Bozucu reddi açısından PSO yaklaşımının ZN yaklaşımına göre çok daha iyi sonuç verdiği görülmektedir.

Basamak yük genliği 0,005 ile 0,05 arasında değiştirilerek uygulanmış ve Şekil 4.3 ile Şekil 4.15 arasındaki şekillerde görülen sonuçlar alınmıştır. Kalıcı durum hatası bütün yük genlikleri için PSO yaklaşımında bariz şekilde daha azdır. Ayrıca ZN yaklaşımında maksimum aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır. Bütün bunları göz önünde bulundurursak basamak referans ve basamak yükte PSO yaklaşımının ZN yaklaşımından daha iyi olduğu söylenebilir. Bozucu reddi söz konusu olduğunda PSO yaklaşımı ZN yaklaşımına göre çok daha iyi sonuçlar vermektedir.

Performans farkları Çizelge 4.1'de gösterilmiştir.

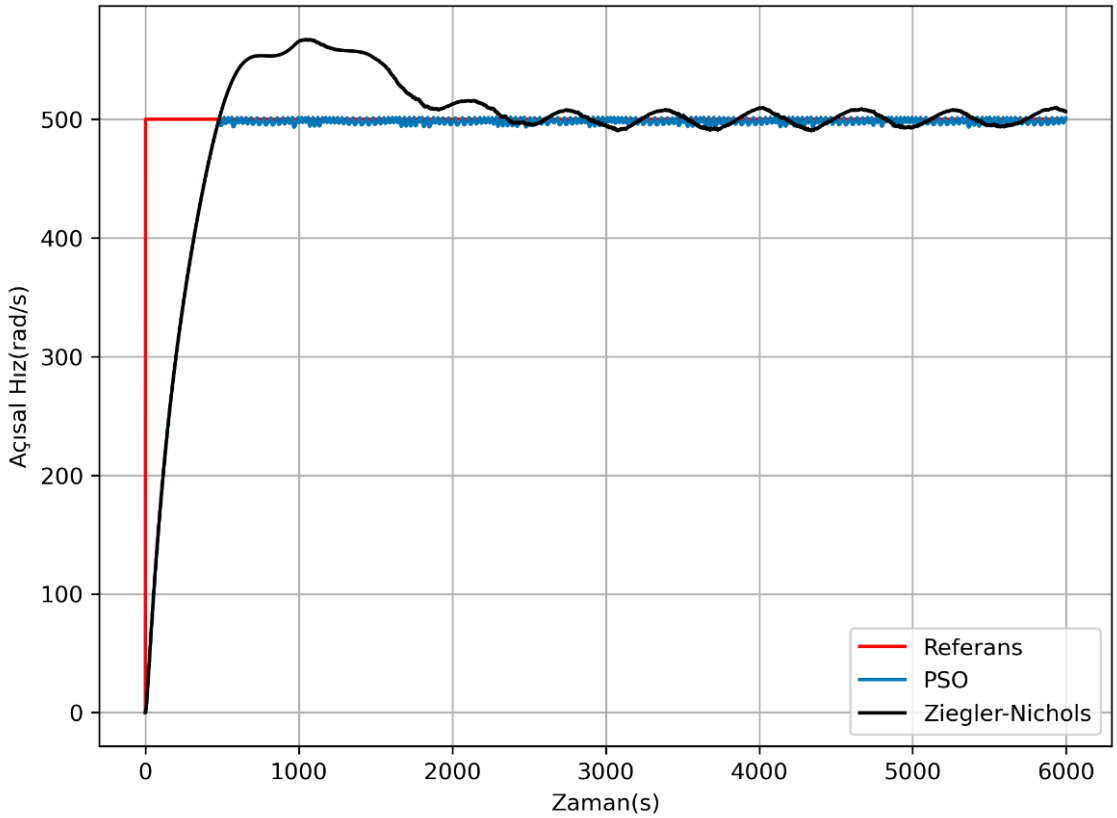
Çizelge 4.1 Basamak referansta genliği 0,001-0,05 aralığında basamak yük uygulanması performans farkları.

Referans	Yük Çeşidi	Büyüklük	PSO	ZN	Yüzdesel Fark
Basamak	Basamak	0,001	9,438	6,289	33,37
Basamak	Basamak	0,002	9,385	6,223	33,69
Basamak	Basamak	0,003	9,334	6,154	34,07
Basamak	Basamak	0,004	9,305	6,125	34,18
Basamak	Basamak	0,005	9,191	6,073	33,92
Basamak	Basamak	0,006	9,132	6,037	33,89
Basamak	Basamak	0,007	9,108	5,993	34,20
Basamak	Basamak	0,008	9,014	5,966	33,81
Basamak	Basamak	0,009	8,952	5,858	34,56
Basamak	Basamak	0,01	8,883	5,876	33,85
Basamak	Basamak	0,02	8,284	5,418	34,60
Basamak	Basamak	0,03	7,725	4,992	35,38
Basamak	Basamak	0,04	7,253	4,548	37,29
Basamak	Basamak	0,05	7,464	6,912	7,40



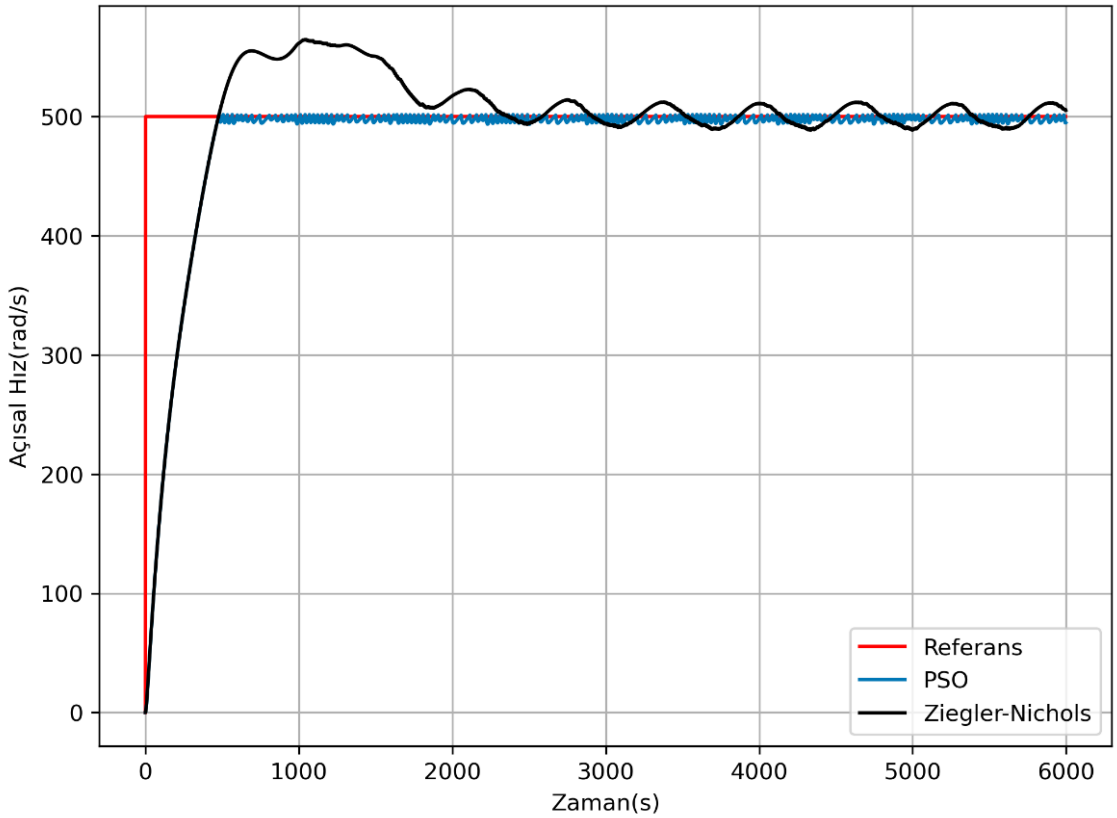
Şekil 4.16 Basamak referansta genliği 0,01 olan sinüsoidal yük uygulanması.

Şekil 4.16'da görüldüğü üzere kalıcı durum hatası, yük genliği 0,01 için PSO yaklaşımında ZN'a göre daha azdır. ZN yaklaşımında aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır. Bozucu reddi açısından PSO yaklaşımının ZN yaklaşımına göre çok daha iyi sonuç verdiği görülmektedir.



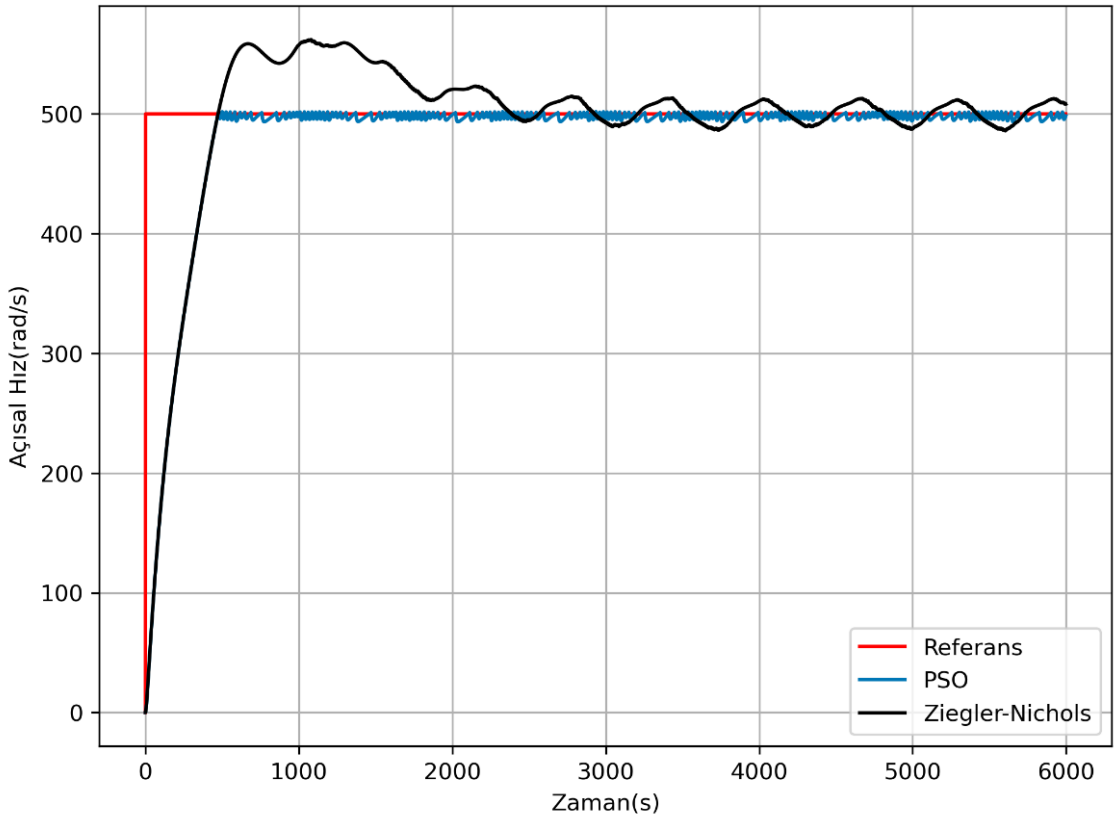
Şekil 4.17 Basamak referansta genliği 0,02 olan sinüsoidal yük uygulanması.

Şekil 4.17'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,02 için PSO yaklaşımında ZN'a göre yine daha azdır. ZN yaklaşımında aşma gözlemlenmekte ve artış göstermektedir. PSO yaklaşımında aşma gözlemlenmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları iki yaklaşımda da yine neredeyse aynıdır.



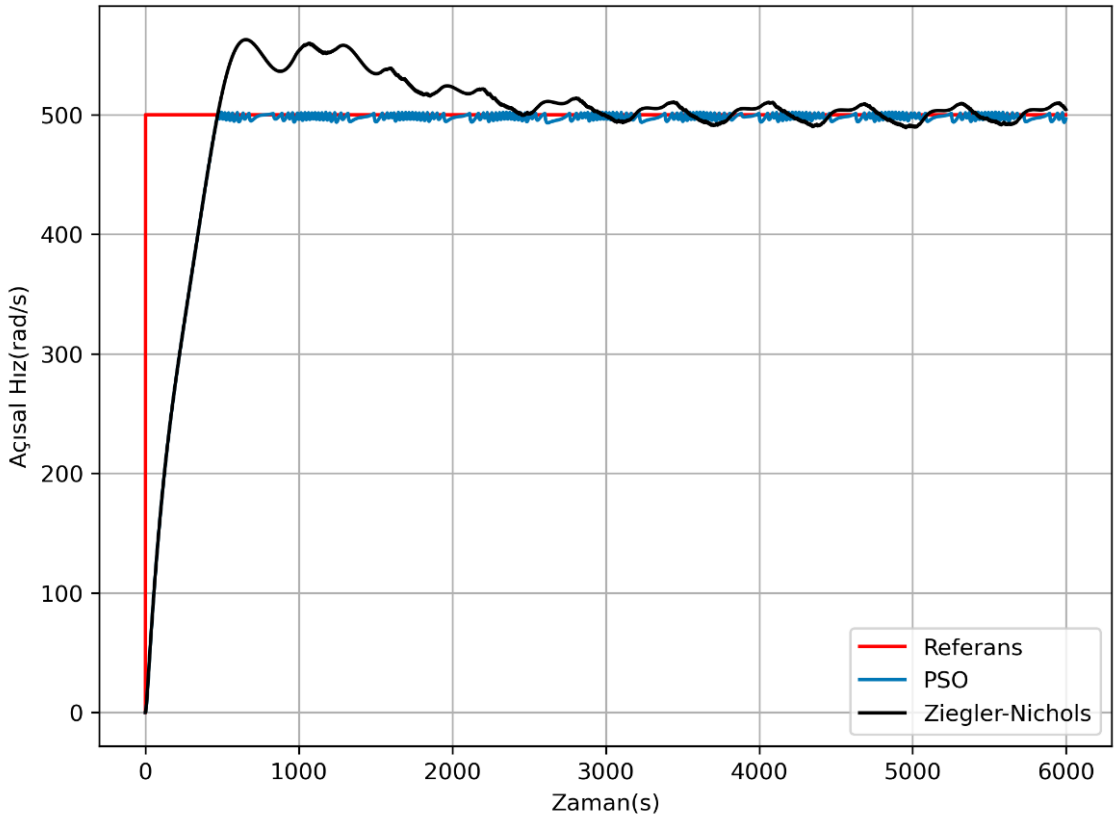
Şekil 4.18 Basamak referansta genliği 0,03 olan sinüsoidal yük uygulanması.

Şekil 4.18'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,03 için PSO yaklaşımında ZN'a göre yine daha azdır. ZN yaklaşımında aşma gözlemlenmekte ve artış göstermektedir. ZN yaklaşımı yük genliği arttıkça performans metriklerinin tamamında PSO yaklaşımına göre gerileme göstermektedir. Aşma miktarının artmasının yanı sıra referans takibi konusunda da zayıflamaya devam etmektedir. PSO yaklaşımında yine aşma görülmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları açısından yine fark görülmemektedir.



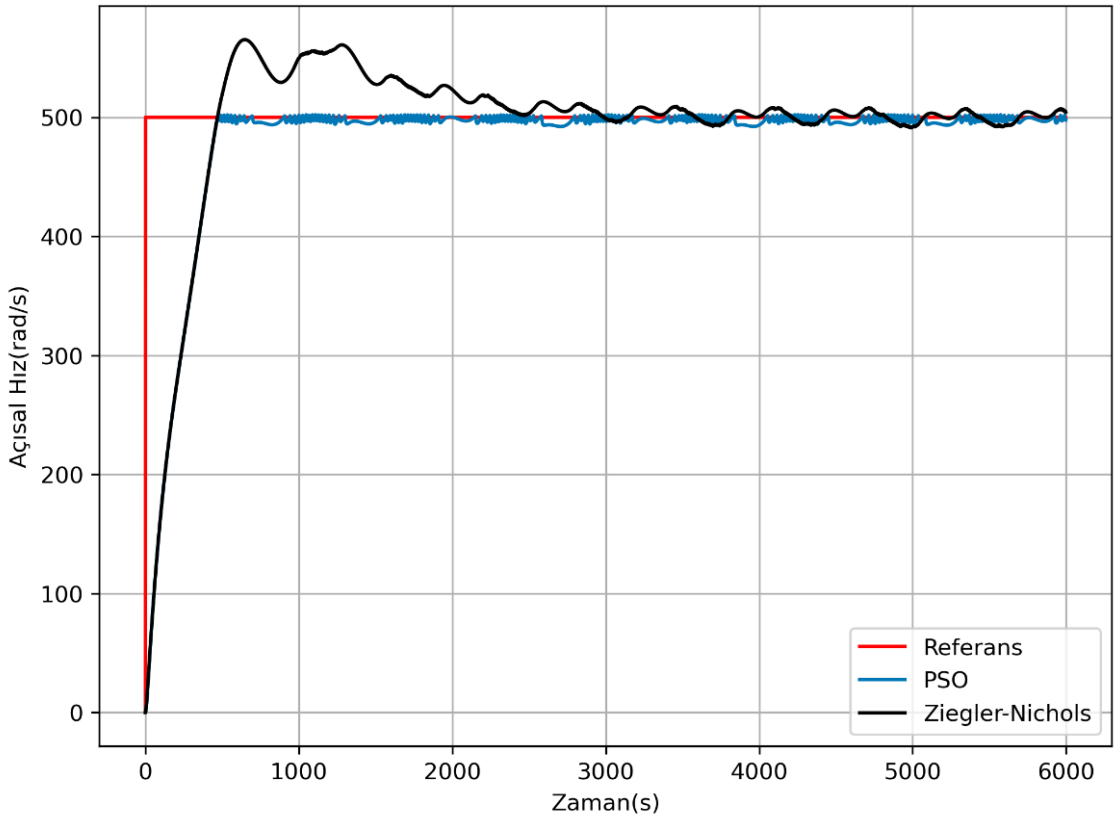
Şekil 4.19 Basamak referansta genliği 0,04 olan sinüsoidal yük uygulanması.

Şekil 4.19'da görüldüğü üzere kalıcı durum hatası, yük genliği 0,04 için PSO yaklaşımında ZN'a göre yine daha azdır. ZN yaklaşımında aşma gözlemlenmekte ve artış göstermektedir. ZN yaklaşımı yük genliği arttıkça performans metriklerinin tamamında PSO yaklaşımına göre gerileme göstermektedir. Aşma miktarının artmasının yanı sıra referans takibi konusunda da zayıflamaya devam etmektedir. PSO yaklaşımında yine aşma görülmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları açısından yine fark görülmemektedir.



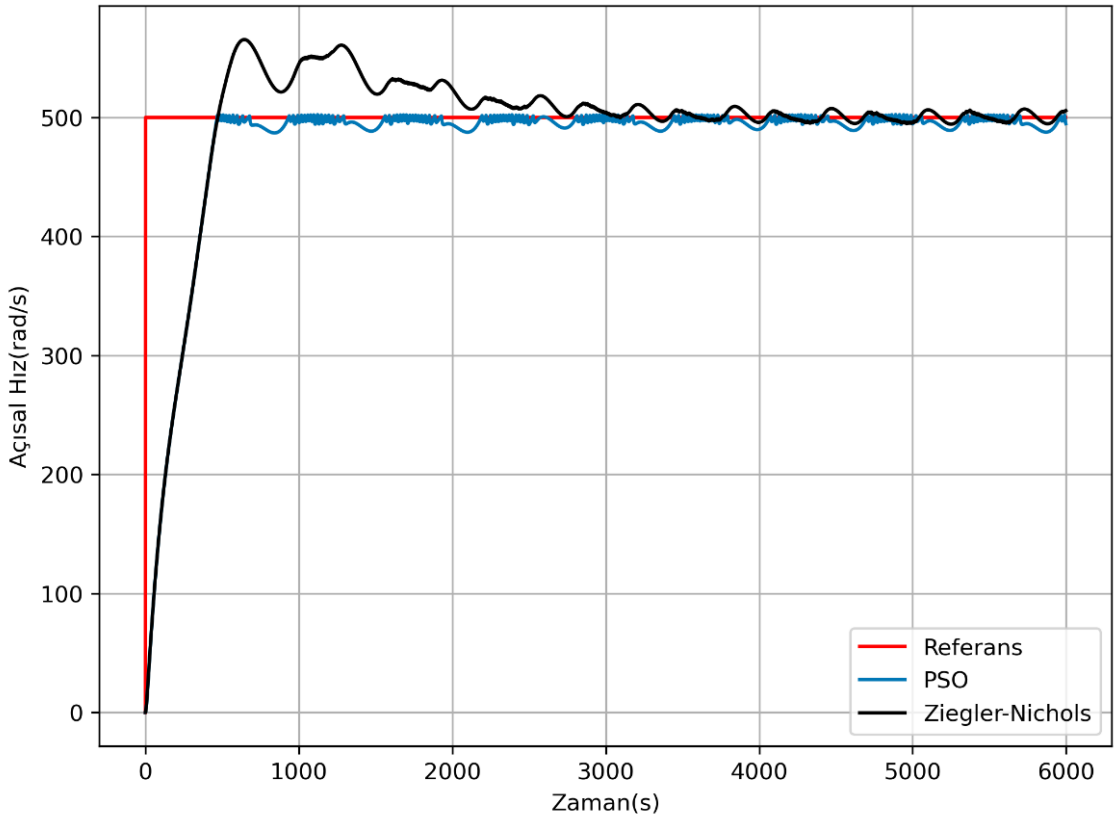
Şekil 4.20 Basamak referansta genliği 0,05 olan sinüsoidal yük uygulanması.

Şekil 4.20'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,05 için PSO yaklaşımında yine ZN'a göre daha azdır. Bu değerden sonra PSO yaklaşımında hafif dalgalanmalar görülmekte ancak yine ZN'a göre referans takibi konusunda çok daha iyi performans göstermektedir. ZN yaklaşımında aşma her genlik artırımında artış göstermeye devam etmiştir. PSO yaklaşımında yine aşma görülmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları açısından yine fark görülmemektedir.



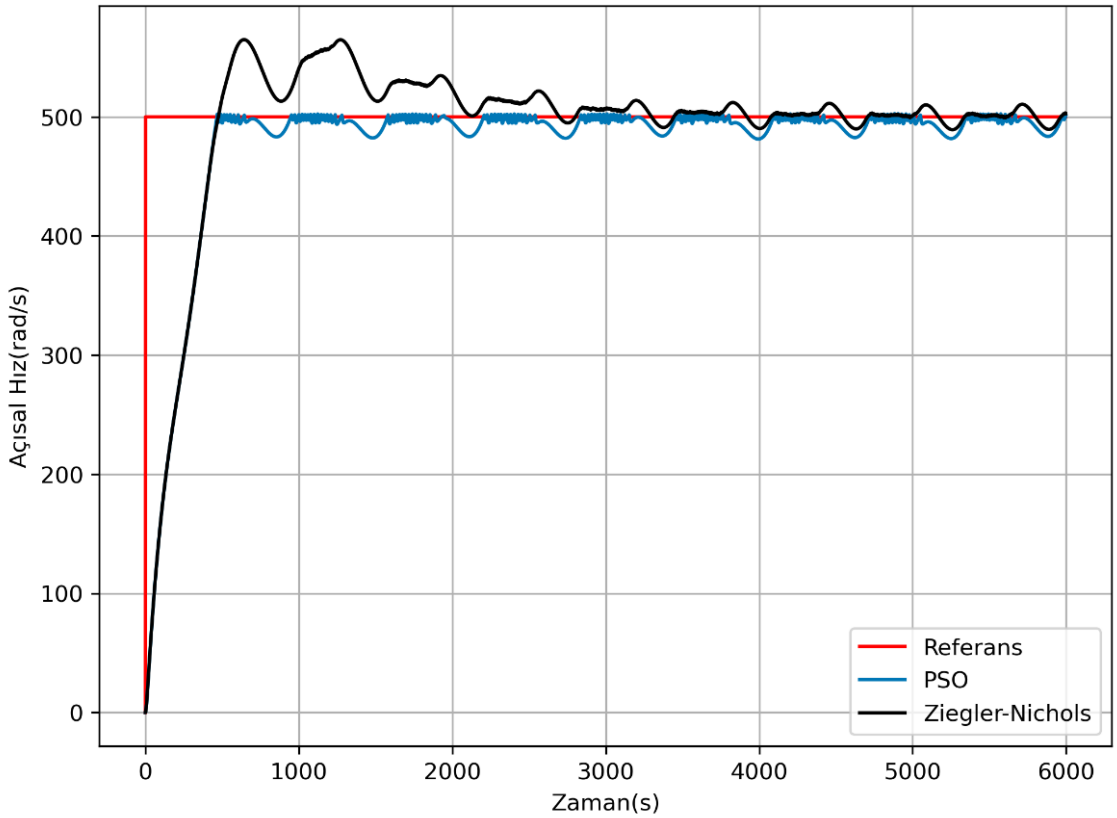
Şekil 4.21 Basamak referansta genliği 0,06 olan sinüsoidal yük uygulanması.

Şekil 4.21'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,06 için PSO yaklaşımında yine ZN'a göre daha azdır. Bu değerden sonra PSO yaklaşımında dalgalanmalar görülmeye devam etmektedir ancak yine ZN'a göre referans takibi konusunda çok daha iyi performans göstermektedir. ZN yaklaşımında aşma her genlik artırımında artış göstermeye devam etmiştir. PSO yaklaşımında yine aşma görülmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları açısından yine fark görülmemektedir.



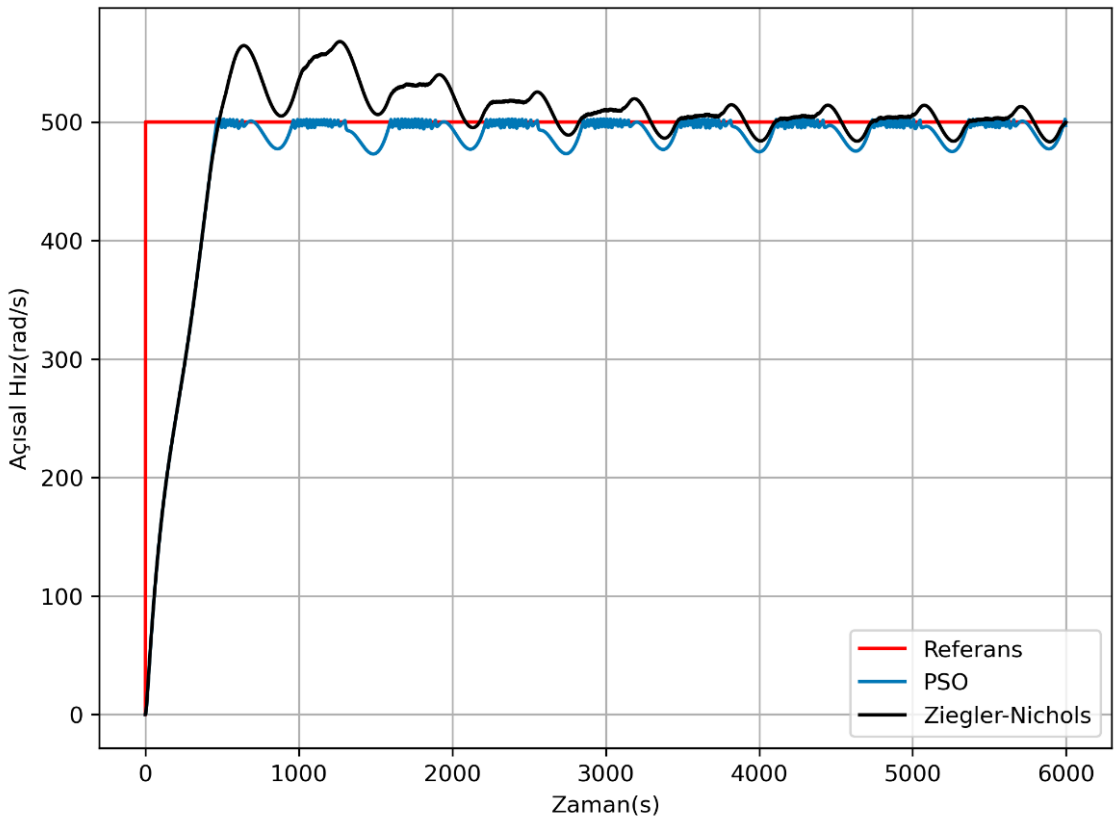
Şekil 4.22 Basamak referansta genliği 0,07 olan sinüsoidal yük uygulanması.

Şekil 4.22'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,07 için PSO yaklaşımında yine ZN'a göre daha azdır. Bu değerden sonra PSO yaklaşımında dalgalanmalar görülmeye devam etmektedir ancak yine ZN'a göre referans takibi konusunda çok daha iyi performans göstermektedir. ZN yaklaşımında aşma her genlik artırımında artış göstermeye devam etmiştir. PSO yaklaşımında yine aşma görülmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları açısından yine fark görülmemektedir.



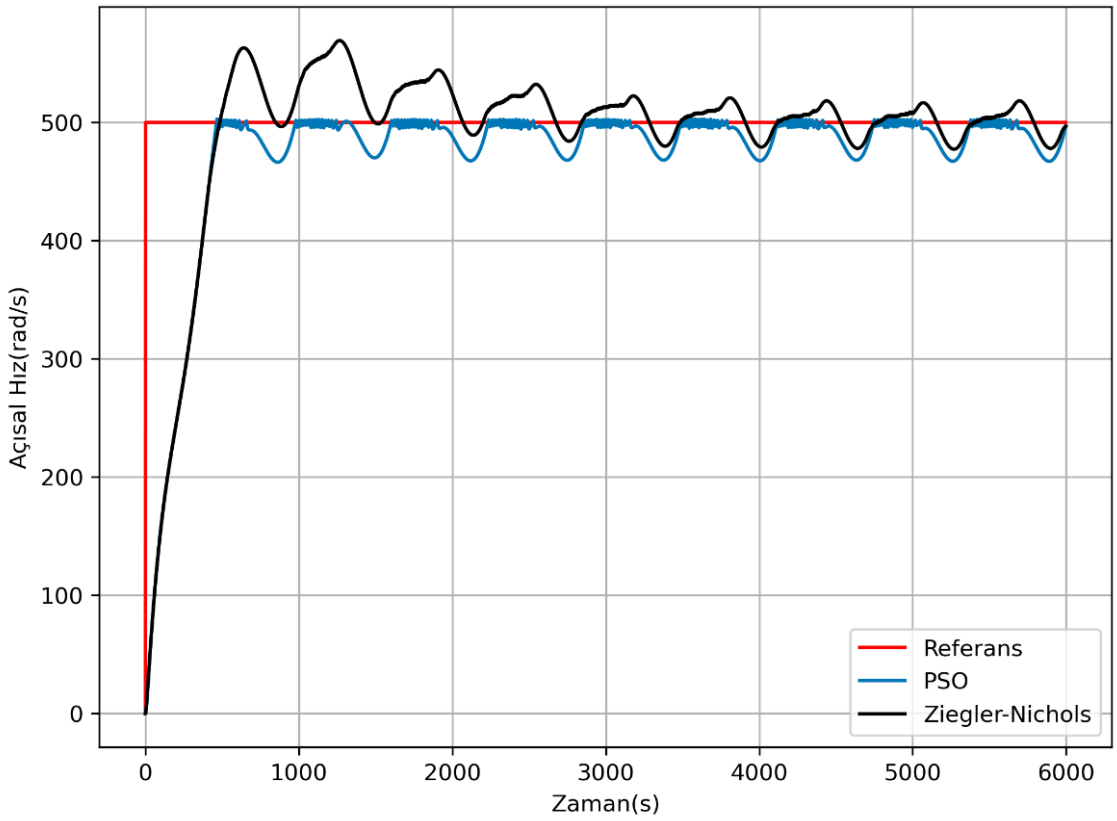
Şekil 4.23 Basamak referansta genliği 0,08 olan sinüsoidal yük uygulanması.

Şekil 4.23'te görüldüğü üzere kalıcı durum hatası, yük genliği 0,08 için PSO yaklaşımında yine ZN'a göre daha azdır. Bu değerden sonra PSO yaklaşımında dalgalanmalar görülmeye devam etmektedir ancak yine ZN'a göre referans takibi konusunda çok daha iyi performans göstermektedir. ZN yaklaşımında aşma her genlik artırımında artış göstermeye devam etmiştir. PSO yaklaşımında yine aşma görülmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları açısından yine fark görülmemektedir.



Şekil 4.24 Basamak referansta genliği 0,09 olan sinüsoidal yük uygulanması.

Şekil 4.24'te görüldüğü üzere kalıcı durum hatası, yük genliği 0,09 için PSO yaklaşımında artış göstermekte ancak ZN yaklaşımına göre yine daha azdır. Bu değerden sonra PSO yaklaşımında dalgalanmalar görülmeye devam etmektedir ancak yine ZN'a göre referans takibi konusunda çok daha iyi performans göstermektedir. ZN yaklaşımında aşma her genlik artırımında artış göstermeye devam etmiştir. PSO yaklaşımında yine aşma görülmemektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları açısından yine fark görülmemektedir.



Şekil 4.25 Basamak referansta genliği 0,1 olan sinüsoidal yük uygulanması.

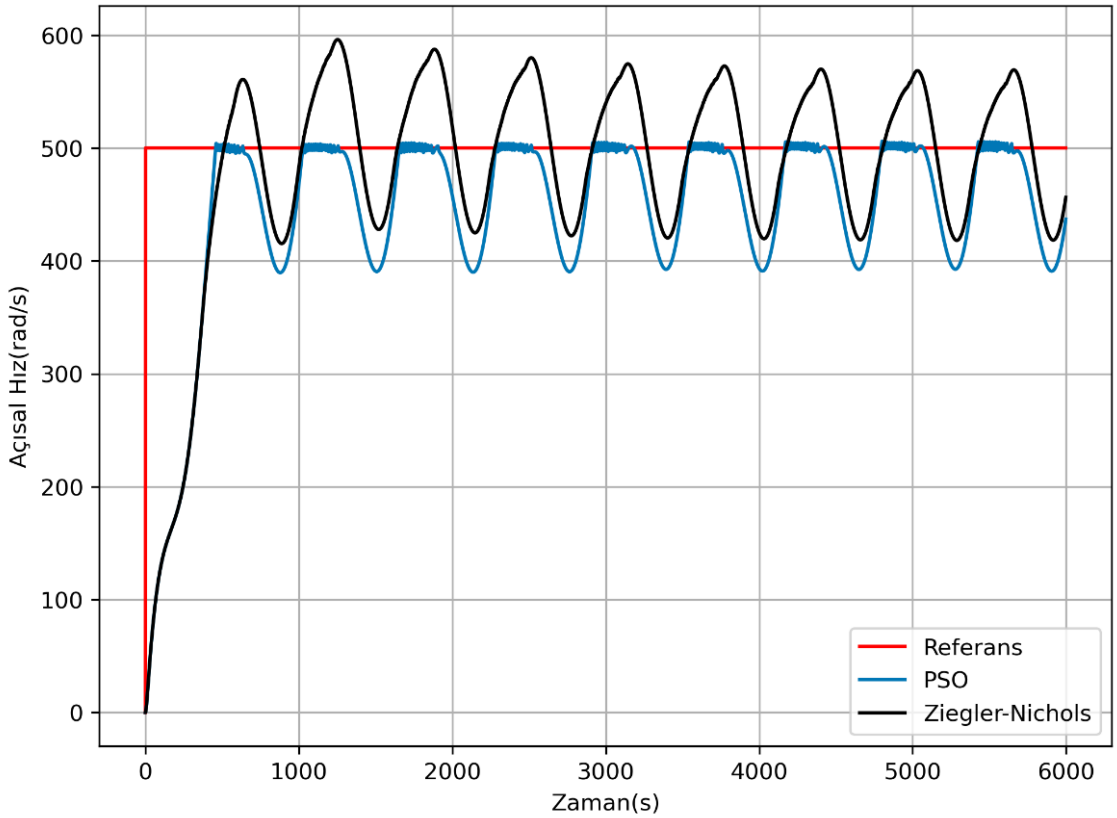
Sinüsoidal yük genliği 0,01 birim ile 0,1 birim arasında değiştirilerek verilmiş ve Şekil 4.16 ile Şekil 4.25 arasındaki şekillerde görülen sonuçlar alınmıştır. Kalıcı durum hatası bütün yük genlikleri için PSO yaklaşımında bariz şekilde daha azdır. Ayrıca ZN yaklaşımında maksimum aşma gözlemlenirken PSO yaklaşımında gözlemlenmemektedir. Yükselme zamanları iki yaklaşımda da hemen hemen aynıdır. Bütün bunları göz önünde bulundurursak basamak referans ve sinüsoidal yükte PSO yaklaşımının ZN yaklaşımından daha gürbüz olduğunu söyleyebiliriz. Bozucu reddi söz konusu olduğunda PSO yaklaşımı ZN yaklaşımına göre çok daha iyi sonuçlar vermektedir.

Performans farkları Çizelge 4.2'de gösterilmiştir.

Çizelge 4.2 Basamak yükte genliği 0,01-0,1 aralığında sinüsoidal yük uygulanması performans farkları.

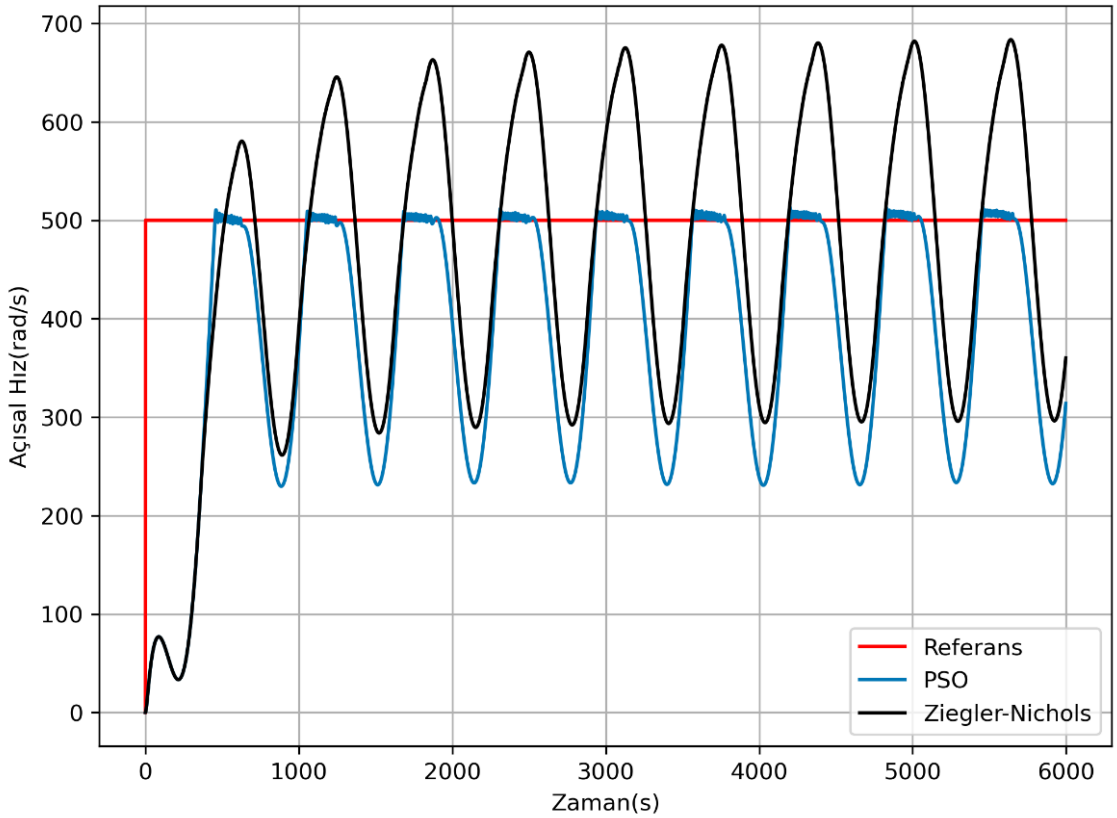
Referans	Yük Çeşidi	Büyüklük	PSO	ZN	Yüzdesel Fark
Basamak	Sinüsoidal	0,01	9,132	5,569	39,02
Basamak	Sinüsoidal	0,02	8,799	5,039	42,73
Basamak	Sinüsoidal	0,03	8,529	4,662	45,34
Basamak	Sinüsoidal	0,04	8,367	4,508	46,12
Basamak	Sinüsoidal	0,05	8,379	4,817	42,51
Basamak	Sinüsoidal	0,06	7,954	5,050	36,51
Basamak	Sinüsoidal	0,07	8,048	5,548	31,06
Basamak	Sinüsoidal	0,08	7,417	5,429	26,80
Basamak	Sinüsoidal	0,09	6,735	5,147	23,58
Basamak	Sinüsoidal	0,1	6,054	4,872	19,52

Çizelge 4.2'deki değerler incelendiğinde PSO yaklaşımının, ZN yaklaşımına göre sayısal olarak ne kadar iyi performans sergilediği görülmektedir. 0,01 – 0,05 genlik değeri aralığında PSO yaklaşık iki kat daha iyi performans sergilemiştir. 0,06 – 0,1 genlik değeri aralığında ise yine ZN yaklaşımına göre PSO yaklaşımı fark azalsa da daha iyi performans göstermiştir. Daha yüksek genlik değerleri için sistem test edilmeye devam etmiştir. Elde edilen bulgular paylaşılmıştır.



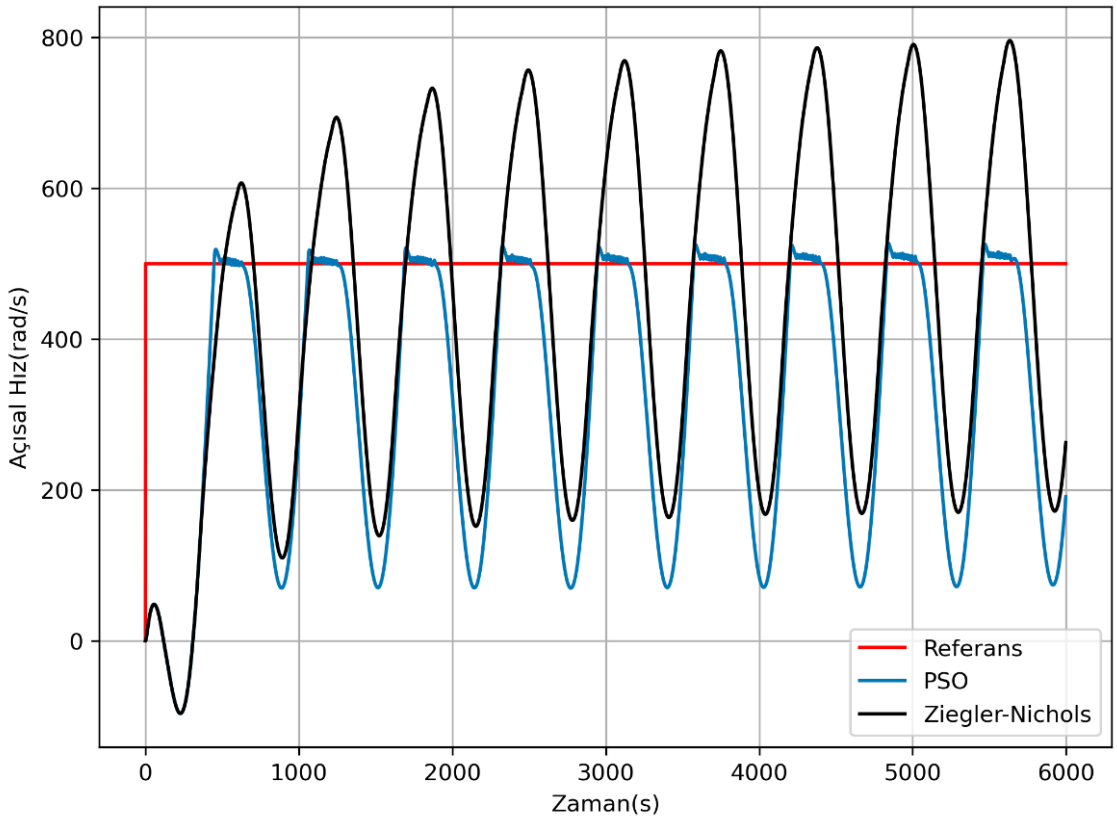
Şekil 4.26 basamak referansta genliği 0,2 olan sinüsoidal yük uygulanması.

Şekil 4.26'da görüldüğü üzere kalıcı durum hatası, yük genliği 0,2 için iki yaklaşımda da artış göstermektedir. Performans metrikleri açısından bakıldığında yine PSO yaklaşımının ZN yaklaşımından daha iyi olduğu görülmektedir. Aşma ZN yaklaşımında artış göstermeye devam etmiştir. PSO yaklaşımında çok küçük bir aşma görülmektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları açısından yine fark görülmemekte ancak yükselme zamanın biraz arttığı gözlemlenmektedir.



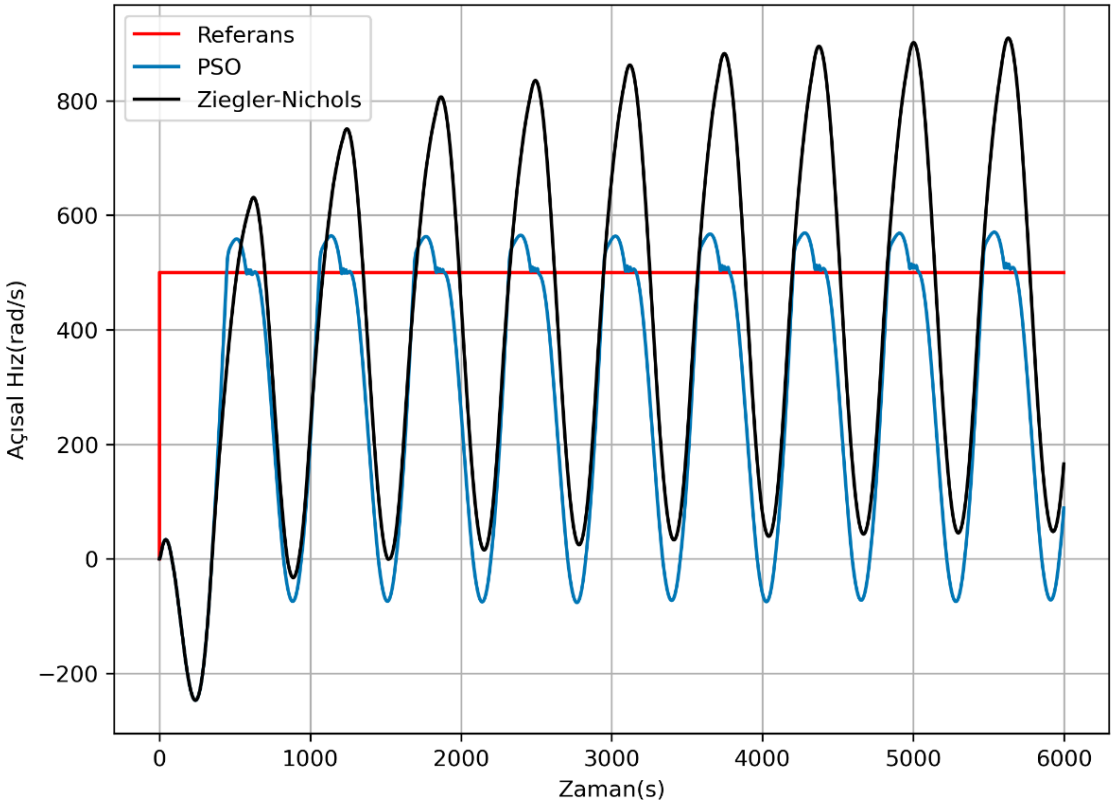
Şekil 4.27 Basamak referansta genliği 0,4 olan sinüsoidal yük uygulanması.

Şekil 4.27'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,4 için iki yaklaşımda da artış göstermektedir. Performans metrikleri açısından bakıldığında yine PSO yaklaşımının ZN yaklaşımından daha iyi olduğu görülmektedir. Aşma ZN yaklaşımında artış göstermeye devam etmiştir. PSO yaklaşımında aşma değerinin biraz daha arttığı görülmektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları açısından yine fark görülmemekte ancak yükselme zamanın biraz daha arttığı gözlemlenmektedir.



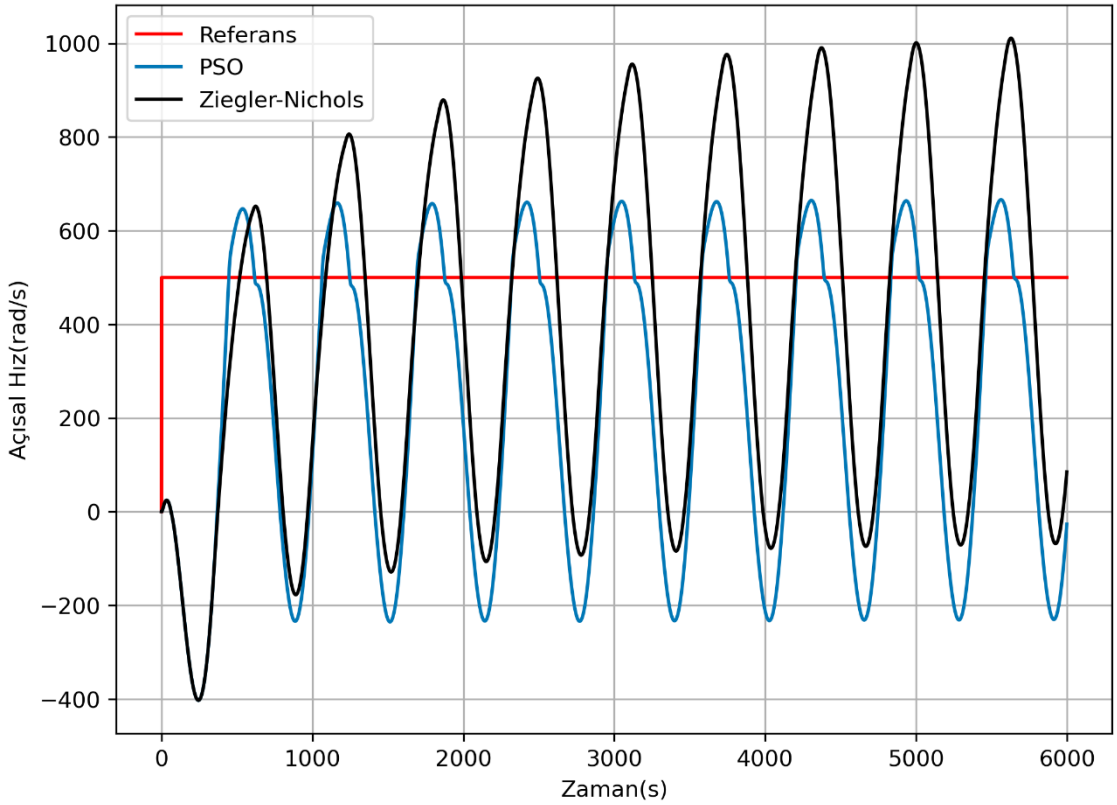
Şekil 4.28 Basamak referansta genliği 0,6 olan sinüsoidal yük uygulanması.

Şekil 4.28'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,6 için iki yaklaşımda da artış göstermektedir. Performans metrikleri açısından bakıldığında yine PSO yaklaşımının ZN yaklaşımından daha iyi olduğu görülmektedir. Aşma ZN yaklaşımında artış göstermeye devam etmiştir. PSO yaklaşımında aşma değerinin biraz daha arttığı görülmektedir. Referans değere yaklaşımının daha iyi olduğu görülmektedir. Yükselme zamanları açısından yine fark görülmemekte ancak yükselme zamanın biraz daha arttığı gözlemlenmektedir.



Şekil 4.29 Basamak referansta genliği 0,8 olan sinüsoidal yük uygulanması.

Şekil 4.29'da görüldüğü üzere kalıcı durum hatası, yük genliği 0,8 için iki yaklaşımda da artış göstermektedir. Performans metrikleri açısından bakıldığında yine PSO yaklaşımının ZN yaklaşımından daha iyi olduğu görülmektedir. Aşma ZN yaklaşımında artış göstermeye devam etmiştir. PSO yaklaşımında aşma değerinin arttığı görülmektedir. Yükselme zamanları açısından yine fark görülmemekte ancak yükselme zamanın giderek arttığı gözlemlenmektedir.



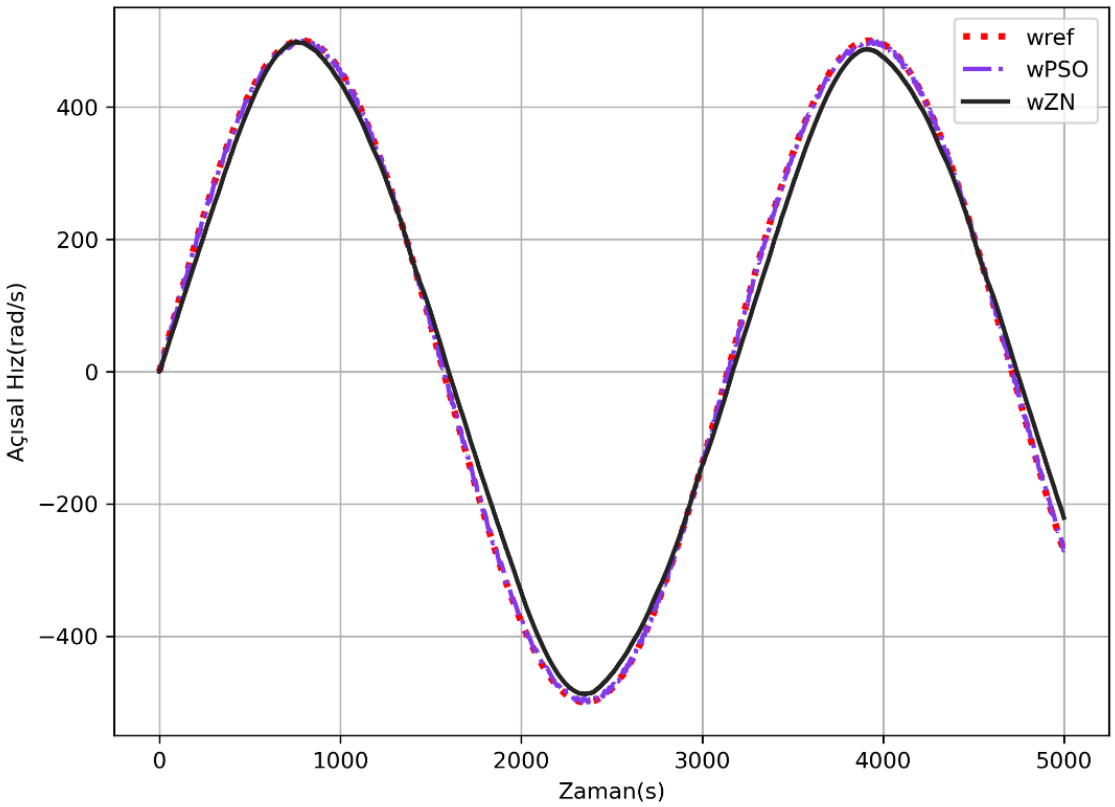
Şekil 4.30 Basamak referansta genliği 1 olan sinüsoidal yük uygulanması.

Şekil 4.17- Şekil 4.30'da görüldüğü üzere basamak referans sinüsoidal yükte PSO tüm performans metriklerinde ZN'den daha iyi sonuç vermektedir. Bu bağlamda değerlendirildiğinde bozucu reddi konusunda PSO yaklaşımının ZN yaklaşımından daha iyi olduğu söylenebilir. Performans değerleri Çizelge 4.3'te verilmiştir.

Çizelge 4.3 Basamak yükte genliği 0,2-1 aralığında sinüsoidal yük uygulanması performans farkları.

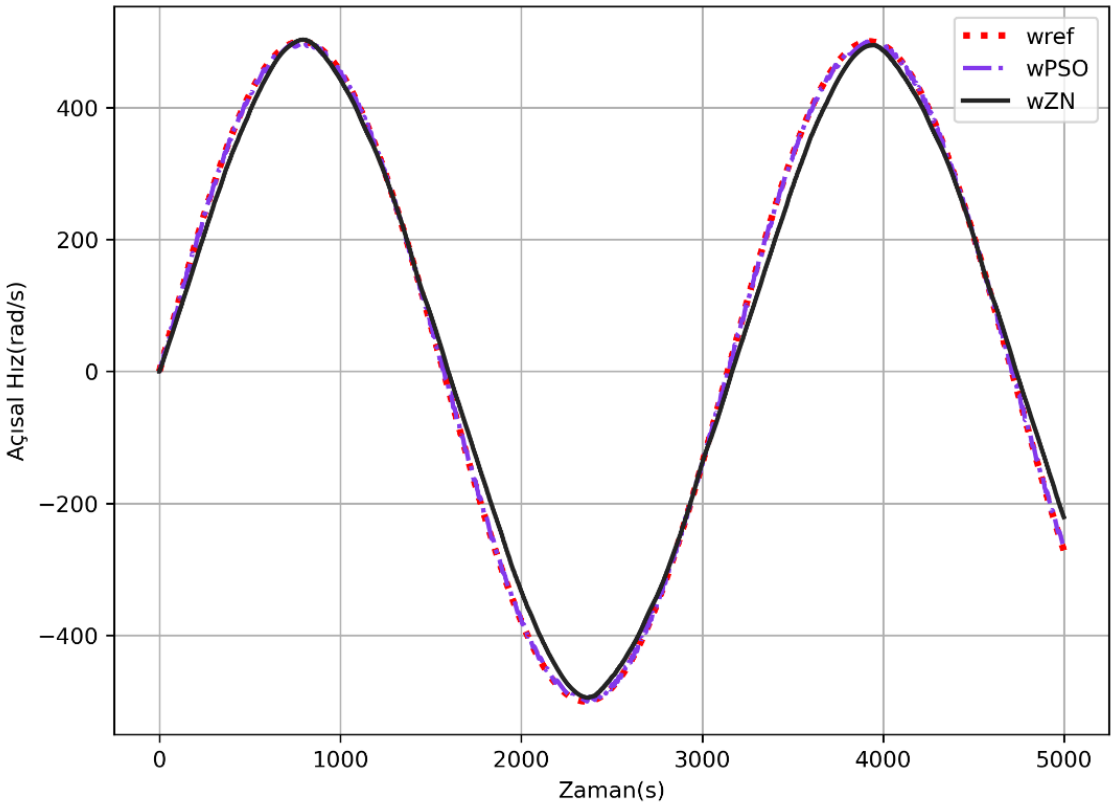
Referans	Yük Çeşidi	Büyüklük	PSO	ZN	Yüzdesel Fark
Basamak	Sinüsoidal	0,2	2.959	2.545	13,99
Basamak	Sinüsoidal	0,3	1,89	1,66	12,43
Basamak	Sinüsoidal	0,4	1,38	1,222	11,45
Basamak	Sinüsoidal	0,5	1,085	0,973	10,32
Basamak	Sinüsoidal	0,6	0,894	0,806	9,84
Basamak	Sinüsoidal	0,7	0,762	0,687	9,84
Basamak	Sinüsoidal	0,8	0,657	0,599	8,83
Basamak	Sinüsoidal	0,9	0,568	0,535	5,81
Basamak	Sinüsoidal	1	0,491	0,482	1,83

Sistem son olarak sinüsoidal referans sinüsoidal yükte test edilmiştir.



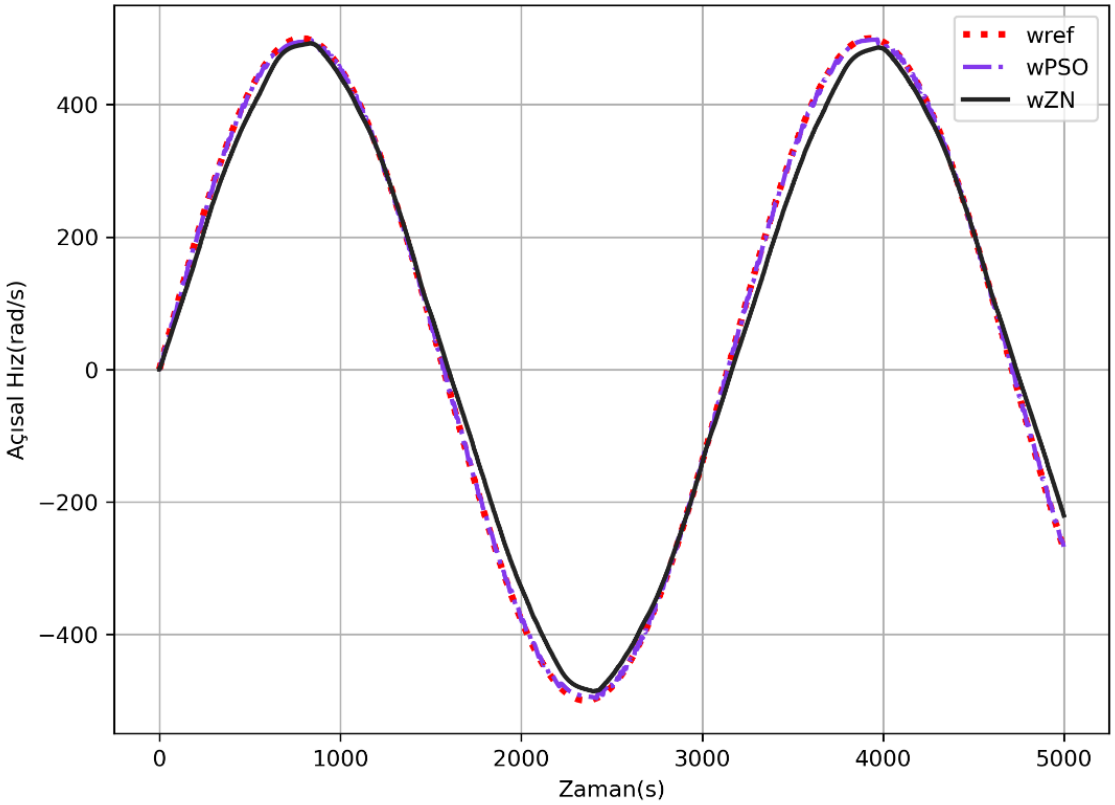
Şekil 4.31 Sinüsoidal referansta genliği 0,01 olan sinüsoidal yük uygulanması.

Şekil 4.31'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,01 için PSO yaklaşımında ZN'a göre daha azdır. Referans değere takibi bakımından PSO yaklaşımı çok iyi sonuç gösterirken ZN yaklaşımında hata olduğu görülmektedir. 0,01 genlik değerli yük altında bozucu reddi açısından PSO yaklaşımının ZN yaklaşımına göre çok daha iyi sonuç verdiği görülmektedir.



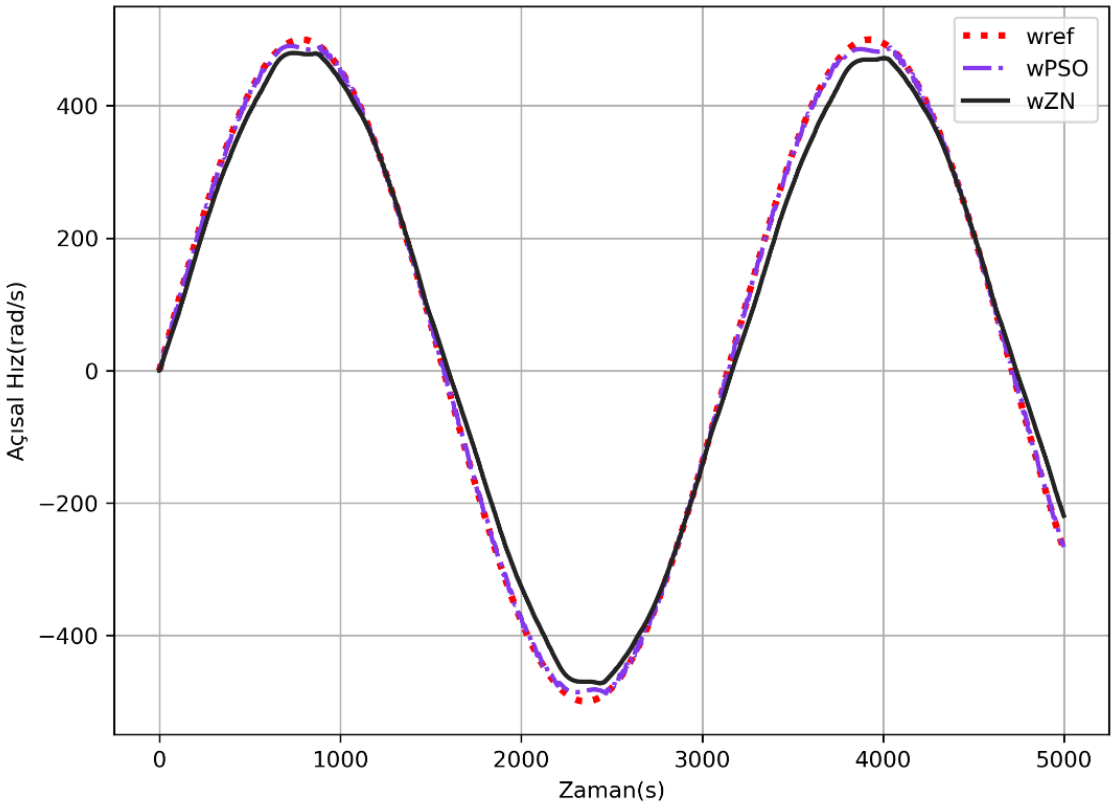
Şekil 4.32 Sinüsoidal referansta genliği 0,03 olan sinüsoidal yük uygulanması.

Şekil 4.32'de görüldüğü üzere kalıcı durum hatası, yük genliği 0,03 için PSO yaklaşımında ZN'a göre daha azdır. Referans değere takibi bakımından PSO yaklaşımı yine çok iyi sonuç gösterirken ZN yaklaşımında hata olduğu görülmektedir. 0,03 genlik değerli yük altında bozucu reddi açısından yine PSO yaklaşımının ZN yaklaşımına göre çok daha iyi sonuç verdiği görülmektedir.



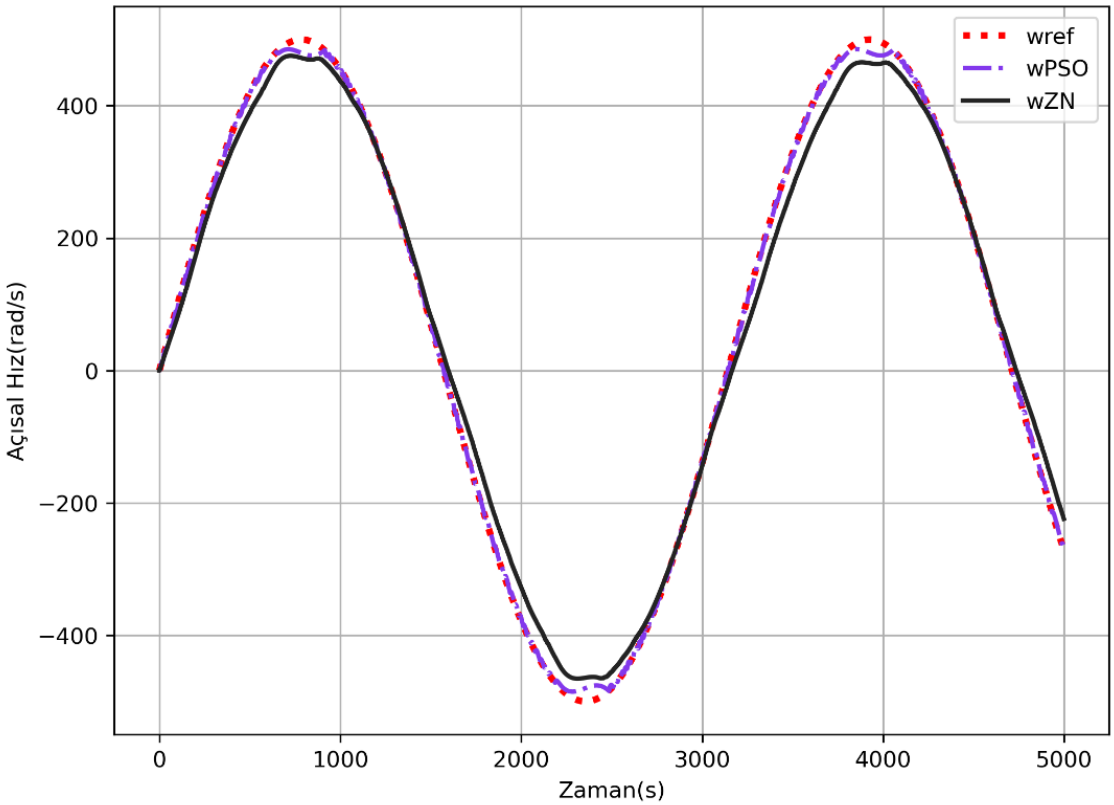
Şekil 4.33 Sinüsoidal referansta genliği 0,05 olan sinüsoidal yük uygulanması.

Şekil 4.33'te görüldüğü üzere kalıcı durum hatası, yük genliği 0,05 için PSO yaklaşımında ZN'a göre daha azdır. Referans değer takibi bakımından PSO yaklaşımı yine çok iyi sonuç gösterirken ZN yaklaşımında hata artışı olduğu görülmektedir. 0,05 genlik değerli yük altında bozucu reddi açısından yine PSO yaklaşımının ZN yaklaşımına göre çok daha iyi sonuç verdiği görülmektedir.



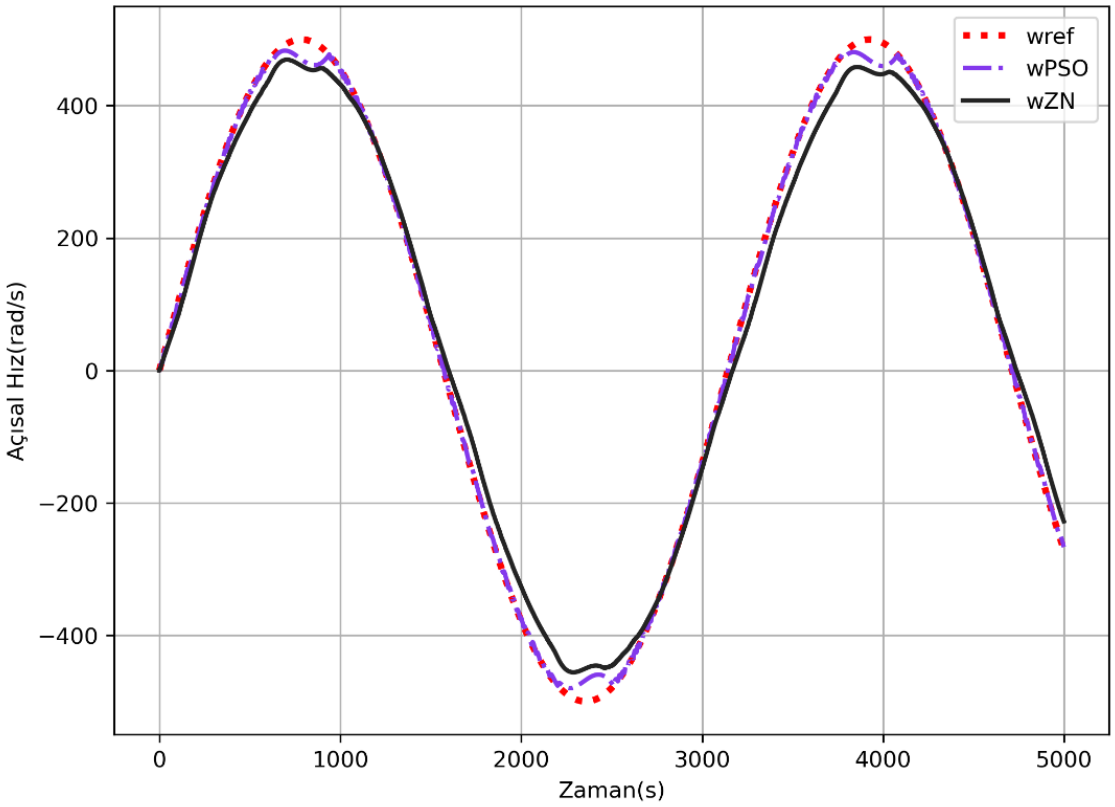
Şekil 4.34 Sinüsoidal referansta genliği 0,07 olan sinüsoidal yük uygulanması.

Şekil 4.34'te görüldüğü üzere kalıcı durum hatası, yük genliği 0,07 için PSO yaklaşımında ZN'a göre daha azdır. Referans değer takibi bakımından PSO yaklaşımı yine çok iyi sonuç gösterirken ZN yaklaşımında hata artış göstermiştir. 0,07 genlik değerli yük altında bozucu reddi açısından yine PSO yaklaşımının ZN yaklaşımına göre çok daha iyi sonuç verdiği görülmektedir.



Şekil 4.35 Sinüsoidal referansta genliği 0,08 olan sinüsoidal yük uygulanması.

Şekil 4.35'te görüldüğü üzere kalıcı durum hatası, yük genliği 0,08 için iki yaklaşımda da artmıştır. Referans değer takibi bakımından PSO yaklaşımı, ufakta olsa hataya sahip olmakla birlikte yine ZN yaklaşımına göre bariz daha iyi olduğu görülmektedir. 0,08 genlik değerli yük altında bozucu reddi açısından yine PSO yaklaşımının ZN yaklaşımına göre çok daha iyi sonuç verdiği görülmektedir.



Şekil 4.36 Sinüsoidal referansta genliği 0,1 olan sinüsoidal yük uygulanması.

Şekil 4.31 – Şekil 4.36 incelendiğinde PSO yaklaşımının sinüsoidal referansı çok daha az bir hatayla izlediği gözlenmektedir. Bozucu reddi söz konusu olduğunda ise yine PSO yaklaşımı ZN yaklaşımına göre çok daha iyi sonuçlar vermektedir. Performans değerleri Çizelge 4.4'te verilmiştir.

Çizelge 4.4 Sinüsoidal referans sinüsoidal yük uygulanması performans farkları.

Referans	Yük Çeşidi	Büyüklük	PSO	ZN	Yüzdesel Fark
Sinüsoidal	Sinüsoidal	0,01	72,92	8,88	87,82
Sinüsoidal	Sinüsoidal	0,02	71,54	9,20	87,14
Sinüsoidal	Sinüsoidal	0,03	72,48	9,47	86,93
Sinüsoidal	Sinüsoidal	0,04	74,12	9,25	87,53
Sinüsoidal	Sinüsoidal	0,05	70,83	8,87	87,47
Sinüsoidal	Sinüsoidal	0,06	66,03	8,68	86,86
Sinüsoidal	Sinüsoidal	0,07	52,50	8,21	84,36
Sinüsoidal	Sinüsoidal	0,08	45,41	7,88	82,65
Sinüsoidal	Sinüsoidal	0,09	38,16	7,58	80,12

5. TARTIŞMA VE SONUÇ

Günümüz teknolojileri baz alındığında, sistemlerin verimli çalışmasının en temel amaçlardan biri olduğu görülmektedir. Dolayısıyla sistem parametrelerinin uygunluğu son derece önem taşımaktadır. Parametreler belirlenirken en iyi algoritmanın seçilmesi, uygulanması ve test edilme aşamaları son derece önem arz etmektedir. Günümüzde halen güncelliğini koruyan Parçacık Sürü Optimizasyonu algoritması, bu algoritmalar arasında uygulaması diğerlerine göre nispeten kolay ve verimli sonuçlar elde edilen bir algoritmadır.

Tüm referans çeşitleri, yük çeşitleri ve değerleri için PSO algoritması ZN yöntemine göre çok daha üstün bir performans göstermektedir. Basamak Referans – Basamak Yük altında test edilen genlik değerlerinde Çizelge 4.1'de görüldüğü üzere PSO yaklaşımı ZN yaklaşımından %32,44 daha iyi performans göstermiştir. Basamak Referans – Sinüsoidal Yük altında test edilen genlik değerlerinde Çizelge 4.2 ve Çizelge 4.3'teki değerlere göre %23,03 daha iyi performans göstermiştir. Sinüsoidal Referans – Sinüsoidal Yük altında test edilen genlik değerlerinde Çizelge 4.4'te değerler baz alındığında %85,65 gibi çok yüksek bir performans farkı ortaya koymuştur.

Sinüsoidal Referans – Sinüsoidal Yük altında performans farkının çok çok daha iyi olduğu görülmektedir. Bu referans ve yük çeşidinde PSO algoritmasının üstün başarı göstermesinin sebeplerinde biri, referans değerin yavaş değişmesi ve motorun referansı takipteki kolaylığı olarak açıklanabilir. Yük çeşidi karmaşıklaştıkça yine PSO algoritmasının daha iyi bir bozucu reddetme performansı gösterdiği görülmektedir.

Sonraki çalışmalarda, ilk olarak çalışmayı gerçek bir sistem üzerinde test edip deneysel sonuçlar ile gerçek sonuçların karşılaştırılması amaçlanmaktadır. Böylelikle oluşturulan kontrol sisteminin fiziksel bir sistemde tepkisi ile karşılaştırma yapılıp algoritmanın iyileştirilmesi ve dolayısıyla parametrelerinde iyileştirilmesi düşünülmektedir. Daha farklı yük çeşitleri ve genlikleri altında sistemin nasıl bir tepki vereceğinin görülmesi ve performans metrikleri açısından karşılaştırılmasının yapılması amaçlanmaktadır.

Daha sonraki aşamalarda ise yapay sinir ağları, makine öğrenmesi algoritmaları kullanılarak PSO yaklaşımının bu yaklaşımlar ile farklarının ortaya konması düşünülmektedir. Günümüzde yaygınlaşmaya başlamış ve kendi kendine öğrenebilme özelliği sunan Pekiştirmeli Öğrenme (Reinforcement Learning) yöntemi kullanılarak performans karşılaştırması yapılması hedeflenmektedir. Bahsi geçen tüm yöntemler ile PID parametreleri, yük altında iken bulunabilir ve karşılaştırmaları yapılabilir. Kontrol sistemlerinin öneminin daha iyi anlaşıldığı mobil robot, biped robot, kuadkopter gibi karmaşık sistemler üzerinde bu yöntemlerin karşılaştırılarak literatüre katkıda bulunulması amaçlanmıştır.

6. KAYNAKLAR

Akyol S, Alataş B, 2012, Güncel Sürü Zekası Optamizasyon Algoritmaları. Nevşehir Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 1, 1, 36-50.

Al-Khayyat A T A, 2021, Robot Path Planning by Area Segmentation Using the Fuzzy C-Means Algorithm and Particle Swarm Optimization, Altınbaş Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 55, İstanbul.

Clerc M, 2006, Particle Swarm Optimization, ISTE Ltd, 244, London.

Gençkaya Ö, 2021, DC Motorun Sürü Algoritma Tabanlı PID Kontrolörle Performans Analizi, Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 65, Elazığ.

Golnaraghi F, Kuo B C, 2017, Automatic Control Systems, McGraw-Hill Education, 944, New Jersey.

Gökçe B, Koca Y B, Aslan Y, Gökçe C O, 2021, Particle Swarm Optimization-based Optimal PID Control of an Agricultural Mobile Robot, Comptes rendus de l'Académie bulgare des Sciences, 74, 4, 568-575.

Halder R K, 2021, Particle Swarm Optimization in Global Path Planning for Swarm of Robots. Applying Particle Swarm Optimization, International Series in Operations Research & Management Science, 306, 250-292.

Ibrahim H E A, Hassan F N, Shomer A O, 2014, Optimal PID Control of A Brushless DC Motor Using PSO and BF Techniques, Ain Shams Engineering Journal, 5, 2, 391-398.

Idir A, Kidouche M, Bensafia Y, Khettab K, Tadjer S A, 2018, Speed Control of DC Motor Using PID and FOPID Controllers Based on Differential Evolution and PSO. International Journal of Intelligent Engineering and Systems, 11, 4, 241-249.

Khalilpour M, Razmjooy N, Hosseini, H, Moallem P, 2011, Optimal Control of DC Motor Using Invasive Weed Optimization (IWO) Algorithm. Majlesi Conference on Electrical Engineering, August 2011, Isfahan, Iran.

Mohamadwasel N B, Bayat O, 2019, Improve DC Motor System Using Fuzzy Logic Control by Particle Swarm Optimization in Use Scale Factors, International Journal of Computer Science and Mobile Computing, 8, 3, 152-160.

Nasri M, Nezamabadi-Pour H, Maghfoori M, 2007, A PSO-Based Optimum Design of PID Controller for a Linear Brushless DC Motor, World Academy of Science, Engineering and Technology, 26, 40, 211-215.

Qi Z, Shi Q, Zhang H, 2019, Tuning of Digital PID Controllers Using Particle Swarm Optimization Algorithm for a CAN-Based DC Motor Subject to Stochastic Delays, IEEE Transactions on Industrial Electronics, 67, 7, 5637-5646.

Song B, Xiao Y, Xu L, 2020, Design of Fuzzy PI Controller for Brushless DC Motor Based on PSO–GSA Algorithm, Systems Science & Control Engineering, 8, 1, 67-77.

Sungthong A, Assawinchaichote W, 2016, Particle Swarm Optimization Based Optimal PID Parameters for Air Heater Temperature Control System, Procedia Computer Science, 86, 108-111.

Weerasooriya S, El-Sharkawi M A, 1991, Identification and Control of a DC Motor Using Back-Propagation Neural Networks, IEEE Transactions on Energy Conversion, 6, 4, 663-669.

Yazgan H, Yener F, Soysal S, Gür A, 2019, Comparison Performances of PSO and GA to Tuning PID Controller for the DC Motor, Sakarya University Journal of Science, 23, 2, 162-174.

Zahir A A M, Alhady S S N, Othman W A F W, Ahmad M F, 2018, Genetic Algorithm Optimization of PID Controller for Brushed DC Motor, Intelligent Manufacturing & Mechatronics: Proceedings of Symposium, 29 January 2018, Pahang, Malaysia.

Ziegler J G, Nichols N B, 1942, Optimum Settings for Automatic Controllers, Transactions of the American Society of Mechanical Engineers, 64, 11, 759-765.

İnternet Kaynakları

1- https://commons.princeton.edu/josephhenry/wp-content/uploads/sites/71/2019/08/electric-motor-history.pdf, 14.04.2022.
2- https://en.wikipedia.org/wiki/Brushed_DC_electric_motor, 15.04.2022.
3- https://tryengineering.org/wp-content/uploads/dcmotorsrevisedREALLYFINAL.pdf, 15.04.2022.
4- https://homepages.laas.fr/lzaccari/seminars/DCmotors.pdf, 12.05.2022.
5- https://www.moog.com/content/dam/moog/literature/MCG/moc23series.pdf, 01.06.2022.

EKLER

EK 1. Basamak Referans / Basamak Yük için Python kodları

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

KpPSO = 1e3
KiPSO = 1e2
KdPSO = 1e-2

KpZN = 3.3
KiZN = 44
KdZN = 0.06

#Constant Variables
#---------------------------------------------------------------------#
La = 0.35e-3 #Inductance
Ra = 0.60 # Resistance
#deltaT = 0.1*La/Ra
deltaT = 1e-4
#tmax = 2
tmax = 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

VaMax = 12.0 # Maximum Value Of Armature Voltage
VaMin = -12.0 # Minimum Value Of Armature Voltage
intErrorMax = 1e3 # Maximum Value Of Integral Of Error
intErrorMin = -1e3 # Minimum Value Of Integral Of Error

referansTuru='step'
```

EK 1. (devam) Basamak Referans / Basamak Yük için Python kodları

```
bozucuTuru='step'
referansBuyuklugu=500
bozucuBuyuklugu=1e-6
referansFrekansi=0
bozucuFrekansi=0

def wrefOlustur(referansTuru, referansBuyuklugu, referansFrekansi):
    n = 0
    wref  = np.zeros((int(tmax/deltaT), 1))  ## Referance Values for Angular Velocity
    for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
        n=n+1 # Variable for iteration
        if referansTuru == 'step':
            wref[n,0] = referansBuyuklugu
        elif referansTuru == 'sine':
            wref[n,0] = referansBuyuklugu*np.sin(referansFrekansi*n*deltaT)
    return wref

def disturbanceOlustur(bozucuTuru, bozucuBuyuklugu, bozucuFrekansi):
    n = 0
    Td = np.zeros((int(tmax/deltaT), 1))  ## State Variable for Disturbance Torque
    for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
        n=n+1 # Variable for iteration
        if bozucuTuru == 'step':
            Td[n,0] = bozucuBuyuklugu
        elif bozucuTuru == 'sine':
            Td[n,0] = bozucuBuyuklugu*np.sin(bozucuFrekansi*n*deltaT)
    return Td

def deneyYap(wref, Kp, Ki, Kd, Td):
    Va = 12.0 # 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
```

EK 1. (devam) Basamak Referans / Basamak Yük için Python kodları

```
w     = np.zeros((int(tmax/deltaT), 1))  ## State Variable 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-1,0] # 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

        # PI Control Output
        PIOut = Kp*error[n-1,0] + Ki*intError + Kd*derError
        Va = PIOut

        # Compare the Armature Voltage with Max/Min Value
        if Va>VaMax:
            Va = VaMax
        if Va < VaMin:
            Va= VaMin

        #Simulation
        Emf = Kb*w[n-1,0]
        Tl[n,0] = Fs*np.sign(w[n-1,0])
        Tl[n,0] = Tl[n,0] + Td[n, 0]
        di_dt = (1/La)*(Va-Ra*i[n-1,0]-Emf)
        i[n,0] = i[n-1,0] + deltaT*di_dt
        Tm[n,0] = Km*i[n,0]
        dw_dt = (1/Inertia)*(Tm[n,0]-Tl[n,0]-bVis*w[n-1,0])
        w[n,0] = w[n-1,0] + deltaT*dw_dt

    return w
```

EK 1. (devam) Basamak Referans / Basamak Yük için Python kodları

```
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;

wref = wrefOlustur(referansTuru, referansBuyuklugu, referansFrekansi)
Td = disturbanceOlustur(bozucuTuru, bozucuBuyuklugu, bozucuFrekansi)
wPSO = deneyYap(wref, KpPSO, KiPSO, KdPSO, Td)
performansPSO = performansHesapla(wref, wPSO)

wref = wrefOlustur(referansTuru, referansBuyuklugu,referansFrekansi)
Td = disturbanceOlustur(bozucuTuru, bozucuBuyuklugu, bozucuFrekansi)
wZN = deneyYap(wref, KpZN, KiZN, KdZN, Td)
performansZN = performansHesapla(wref, wZN)

fig = plt.figure(figsize=(8,6),dpi =300)
plt.plot(wref[:,0], 'r', label='wref')
plt.legend()
plt.plot(wPSO[:,0], 'b', label='wPSO')
plt.legend()
plt.plot(wZN[:,0], 'g', label='wZN')
plt.title(f"Basamak Referans / Basamak Yük / Genlik = {bozucuBuyuklugu} ")
plt.xlabel("Zaman(s)")
plt.ylabel("Açısal Hız(rad/s)")
plt.legend()
plt.grid()
plt.show()

print(performansPSO)
print(performansZN)
```

EK 2. Basamak Referans / Sinüsoidal Yük için Python Kodları

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

KpPSO = 1e3
KiPSO = 1e2
KdPSO = 1e-2

KpZN = 3.3
KiZN = 44
KdZN = 0.06

#Constant Variables
#-----------------------------------------------------------------------#
La = 0.35e-3 #Inductance
Ra = 0.60 # Resistance
#deltaT = 0.1*La/Ra
deltaT = 1e-4
#tmax = 2
tmax = 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

VaMax = 12.0 # Maximum Value Of Armature Voltage
VaMin = -12.0 # Minimum Value Of Armature Voltage
intErrorMax = 1e3 # Maximum Value Of Integral Of Error
intErrorMin = -1e3 # Minimum Value Of Integral Of Error

referansTuru='step'
bozucuTuru='sine'
referansBuyuklugu=500
```

EK 2. (devam) Basamak Referans / Sinüsoidal Yük için Python Kodları

```
bozucuBuyuklugu=80e-2
referansFrekansi=0
bozucuFrekansi=1e2

def wrefOlustur(referansTuru, referansBuyuklugu, referansFrekansi):
    n = 0
    wref  = np.zeros((int(tmax/deltaT), 1))  ## Referance Values for Angular Velocity
    for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
        n=n+1 # Variable for iteration
        if referansTuru == 'step':
            wref[n,0] = referansBuyuklugu
        elif referansTuru == 'sine':
            wref[n,0] = referansBuyuklugu*np.sin(referansFrekansi*n*deltaT)
    return wref

def disturbanceOlustur(bozucuTuru, bozucuBuyuklugu, bozucuFrekansi):
    n = 0
    Td = np.zeros((int(tmax/deltaT), 1))  ## State Variable for Disturbance Torque
    for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
        n=n+1 # Variable for iteration
        if bozucuTuru == 'step':
            Td[n,0] = bozucuBuyuklugu
        elif bozucuTuru == 'sine':
            Td[n,0] = bozucuBuyuklugu*np.sin(bozucuFrekansi*n*deltaT)
    return Td

def deneyYap(wref, Kp, Ki, Kd, Td):
    Va = 12.0 # 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
    error = np.zeros((int(tmax/deltaT), 1))  ##
```

EK 2. (devam) Basamak Referans / Sinüsoidal Yük için Python Kodları

```
#-------------------------------------------------------------------------#
    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-1,0] # 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

        # PI Control Output
        PIOut = Kp*error[n-1,0] + Ki*intError + Kd*derError
        Va = PIOut

        # Compare the Armature Voltage with Max/Min Value
        if Va>VaMax:
            Va = VaMax
        if Va < VaMin:
            Va= VaMin

        #Simulation
        Emf = Kb*w[n-1,0]
        Tl[n,0] = Fs*np.sign(w[n-1,0])
        Tl[n,0] = Tl[n,0] + Td[n, 0]
        di_dt = (1/La)*(Va-Ra*i[n-1,0]-Emf)
        i[n,0] = i[n-1,0] + deltaT*di_dt
        Tm[n,0] = Km*i[n,0]
        dw_dt = (1/Inertia)*(Tm[n,0]-Tl[n,0]-bVis*w[n-1,0])
        w[n,0] = w[n-1,0] + deltaT*dw_dt

    return w

def performansHesapla(wref, w):
```

EK 2. (devam) Basamak Referans / Sinüsoidal Yük için Python Kodları

```
 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;

wref = wrefOlustur(referansTuru, referansBuyuklugu, referansFrekansi)
Td = disturbanceOlustur(bozucuTuru, bozucuBuyuklugu, bozucuFrekansi)
wPSO = deneyYap(wref, KpPSO, KiPSO, KdPSO, Td)
performansPSO = performansHesapla(wref, wPSO)

wref = wrefOlustur(referansTuru, referansBuyuklugu,referansFrekansi)
Td = disturbanceOlustur(bozucuTuru, bozucuBuyuklugu, bozucuFrekansi)
wZN = deneyYap(wref, KpZN, KiZN, KdZN, Td)
performansZN = performansHesapla(wref, wZN)

fig = plt.figure(figsize=(8,6),dpi =300)
plt.plot(wref[:,0], 'r', label='wref')
plt.legend()
plt.plot(wPSO[:,0], '#8338ec',ls='-.', label='wPSO',linewidth = 2)
plt.legend()
plt.plot(wZN[:,0], '#232323', label='wZN', linewidth = 2)
plt.title(f"Sinüsoidal Yük / Genlik = {bozucuBuyuklugu} ")
plt.xlabel("Zaman(s)")
plt.ylabel("Açısal Hız(rad/s)")
plt.legend()
plt.grid()
plt.show()

print(f"PSO Performansı : {performansPSO}")
print(f"ZN  Performansı : {performansZN}")
```

EK 3. Sinüsoidal Referans / Sinüsoidal Yük için Python kodları

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

KpPSO = 1e3
KiPSO = 1e2
KdPSO = 1e-2

KpZN = 3.3
KiZN = 44
KdZN = 0.06

#Constant Variables
#----------------------------------------------------------------------#
La = 0.35e-3 #Inductance
Ra = 0.60 # Resistance
#deltaT = 0.1*La/Ra
deltaT = 1e-4
#tmax = 2
tmax = 5e-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

VaMax = 12.0 # Maximum Value Of Armature Voltage
VaMin = -12.0 # Minimum Value Of Armature Voltage
intErrorMax = 1e3 # Maximum Value Of Integral Of Error
intErrorMin = -1e3 # Minimum Value Of Integral Of Error

referansTuru='sine'
bozucuTuru='sine'
referansBuyuklugu=500
```

EK 3. (devam) Sinüsoidal Referans / Sinüsoidal Yük için Python kodları

```
bozucuBuyuklugu=10e-2
referansFrekansi=2e1
bozucuFrekansi=1e2

def wrefOlustur(referansTuru, referansBuyuklugu, referansFrekansi):
    n = 0
    wref  = np.zeros((int(tmax/deltaT), 1))  ## Referance Values for Angular Velocity
    for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
        n=n+1 # Variable for iteration
        if referansTuru == 'step':
            wref[n,0] = referansBuyuklugu
        elif referansTuru == 'sine':
            wref[n,0] = referansBuyuklugu*np.sin(referansFrekansi*n*deltaT)
    return wref

def disturbanceOlustur(bozucuTuru, bozucuBuyuklugu, bozucuFrekansi):
    n = 0
    Td = np.zeros((int(tmax/deltaT), 1))  ## State Variable for Disturbance Torque
    for n1 in np.linspace(0, nMax-1, num=nMax-1): # For Loop Starts Here
        n=n+1 # Variable for iteration
        if bozucuTuru == 'step':
            Td[n,0] = bozucuBuyuklugu
        elif bozucuTuru == 'sine':
            Td[n,0] = bozucuBuyuklugu*np.sin(bozucuFrekansi*n*deltaT)
    return Td

def deneyYap(wref, Kp, Ki, Kd, Td):
    Va = 12.0 # 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
    error = np.zeros((int(tmax/deltaT), 1))  ##
```

EK 3. (devam) Sinüsoidal Referans / Sinüsoidal Yük için Python kodları

```
#-------------------------------------------------------------------------#
    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-1,0] # 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

        # PI Control Output
        PIOut = Kp*error[n-1,0] + Ki*intError + Kd*derError
        Va = PIOut

        # Compare the Armature Voltage with Max/Min Value
        if Va>VaMax:
            Va = VaMax
        if Va < VaMin:
            Va= VaMin

        #Simulation
        Emf = Kb*w[n-1,0]
        Tl[n,0] = Fs*np.sign(w[n-1,0])
        Tl[n,0] = Tl[n,0] + Td[n, 0]
        di_dt = (1/La)*(Va-Ra*i[n-1,0]-Emf)
        i[n,0] = i[n-1,0] + deltaT*di_dt
        Tm[n,0] = Km*i[n,0]
        dw_dt = (1/Inertia)*(Tm[n,0]-Tl[n,0]-bVis*w[n-1,0])
        w[n,0] = w[n-1,0] + deltaT*dw_dt

    return w

def performansHesapla(wref, w):
```

EK 3. (devam) Sinüsoidal Referans / Sinüsoidal Yük için Python kodları

```
    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;

wref = wrefOlustur(referansTuru, referansBuyuklugu, referansFrekansi)
Td = disturbanceOlustur(bozucuTuru, bozucuBuyuklugu, bozucuFrekansi)
wPSO = deneyYap(wref, KpPSO, KiPSO, KdPSO, Td)
performansPSO = performansHesapla(wref, wPSO)

wref = wrefOlustur(referansTuru, referansBuyuklugu,referansFrekansi)
Td = disturbanceOlustur(bozucuTuru, bozucuBuyuklugu, bozucuFrekansi)
wZN = deneyYap(wref, KpZN, KiZN, KdZN, Td)
performansZN = performansHesapla(wref, wZN)

fig = plt.figure(figsize=(8,6),dpi =300)
plt.plot(wref[:,0], 'r:', label='wref',linewidth = 3)
plt.legend()
plt.plot(wPSO[:,0], '#8338ec',ls='-.', label='wPSO',linewidth = 2)
plt.legend()
plt.plot(wZN[:,0], '#232323', label='wZN', linewidth = 2)
plt.legend()
plt.title(f"Sinüsoidal Yük / Genlik = {bozucuBuyuklugu} ")
plt.xlabel("Zaman(s)")
plt.ylabel("Açısal Hız(rad/s)")
plt.grid()
plt.show()

print(f"PSO Performansı : {performansPSO}")
print(f"ZN  Performansı : {performansZN}")
```