T.C.
MANİSA CELAL BAYAR ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ
MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI
TEZLİ YÜKSEK LİSANS PROGRAMI

İNSANSIZ ARAÇLARDA FARKLI META-SEZGİSEL YÖNTEMLERİN PERFORMANS ANALİZİ

Lizge KORKMAZ

Danışman
Doç. Dr. Serkan ÇAŞKA
İkinci Danışman: Doç.Dr. Mete ÖZBALTAN

MANİSA–2025

LİZGE

KORKMAZ

İNSANSIZ ARAÇLARIN ÇOK GÖVDELİ MODELLERİNİN ANALİZİ VE KONTROLÜ

2025

TAAHHÜTNAME

Bu tezin Manisa Celal Bayar Üniversitesi Lisansüstü Eğitim Enstitüsü Mekatronik Mühendisliği Ana Bilim Dalında akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin referans gösterilerek tezde yer aldığını, tamamen kendi çalışmam olduğunu, her alıntıya kaynak gösterdiğimi beyan ederim.

İmza

Lizge KORKMAZ

ÖZET

Yüksek Lisans Tezi

Lizge KORKMAZ

Manisa Celal Bayar Üniversitesi

Lisansüstü Eğitim Enstitüsü

Mekatronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Serkan ÇAŞKA

İkinci Danışman:Doç. Dr. Mete ÖZBALTAN

Günümüzde insansız araçlar, savunma sanayisinden lojistiğe, tarımdan arama-kurtarma operasyonlarına kadar birçok alanda aktif olarak kullanılmaktadır. Özellikle çok gövdeli insansız araçlar, yüksek taşıma kapasiteleri, gelişmiş manevra kabiliyetleri ve daha dengeli hareket kabiliyetleri ile öne çıkmaktadır. Ancak, birden fazla gövde ve itki sistemine sahip bu araçların kontrolü, karmaşık dinamik yapı ve değişken dış etkenler nedeniyle önemli mühendislik zorlukları barındırmaktadır.

Bu tez çalışmasında, çok gövdeli insansız araçların dinamik analizinin yapılması ve etkili kontrol yöntemlerinin geliştirilmesi hedeflenmektedir. İlk olarak, bu tür sistemler üzerine yapılmış mevcut çalışmalar incelenerek literatürde kullanılan modelleme ve kontrol yaklaşımları değerlendirilmiştir. Ardından, farklı kontrol algoritmaları kullanılarak sistemin kararlılığı, hareket kabiliyeti ve performansı

detaylı bir şekilde analiz edilmiştir. Çalışma kapsamında klasik kontrol yöntemlerinin yanı sıra uyarlamalı kontrol ve yapay zekâ tabanlı algoritmalar gibi modern yaklaşımlar da test edilerek etkinlikleri karşılaştırılmıştır. Ayrıca, insansız araçlar için yol planlaması yapılarak yapay zekâ yöntemlerinden enerji ve zamandan tasarruf edilmesinde faydalanılmıştır. Elde edilen sonuçlar, özellikle yapay zekâ algoritmalarının, çok gövdeli insansız araçlar için yüksek doğrulukta ve stabil bir kontrol sağladığını göstermektedir. Bu çalışma, gelecekte insansız kara araçlarının daha güvenilir ve verimli bir şekilde kullanılmasına katkı sağlamayı amaçlamaktadır.

Anahtar Kelimeler: İnsansız araçlar, çok gövdeli sistemler, dinamik modelleme, otomatik kontrol algoritmaları, yol planlaması

2025, 143 sayfa

ABSTRACT

M.Sc. Thesis

Lizge KORKMAZ

Manisa Celal Bayar University

Graduate School of Education

Department of Mechatronics Engineering

Supervisor: Doç. Dr. Serkan ÇAŞKA

Co-Supervisor: Doç. Dr. Mete ÖZBALTAN

Nowadays, unmanned vehicles are actively used in various fields, ranging from defense industries to logistics, agriculture, and search-and-rescue operations. In particular, multi-body unmanned vehicles stand out due to their high payload capacity, enhanced maneuverability, and improved stability. However, the control of these vehicles, which involve multiple bodies and propulsion systems, presents significant engineering challenges due to their complex dynamic structure and varying external factors.

This thesis aims to conduct a dynamic analysis of multi-body unmanned vehicles and develop effective control methods. First, existing studies on such systems are reviewed, and the modeling and control approaches used in the literature are evaluated. Then, different control algorithms are applied to analyze the stability, mobility, and overall performance of the system. In addition to classical control methods, modern approaches such as adaptive control and artificial intelligence-

based algorithms are tested and compared in terms of their effectiveness. In addition, artificial intelligence methods were used to save energy and time by making path planning for unmanned vehicles. The results obtained indicate that artificial intelligence and advanced algorithms provide highly accurate and stable control for multi-body unmanned vehicles.

Keywords: Unmanned vehicles, multi-body systems, dynamic modeling, control algorithms, stability analysis.

2025, 143 pages

ÖNSÖZ VE TEŞEKKÜR

Tez sürecinin her aşamasındaki bilgi, tecrübe ve yönlendirmeleri için danışmanım Doç. Dr. Serkan ÇAŞKA'ya en içten teşekkürlerimi sunarım. Akademik ve teknik konulardaki kıymetli katkıları için ikinci danışmanım Doç. Dr. Mete ÖZBALTAN'a da ayrıca teşekkür ederim. Tez jürimde görev alarak yapıcı eleştiri ve önerileriyle çalışmanın niteliğinin artmasına katkı sağlayan saygıdeğer hocalarıma minnettarım.

Bu süreçte manevi desteğini esirgemeyen aileme ve yakın arkadaşlarıma sabır ve anlayışları için teşekkür ederim. Bu tez, herhangi bir kurum/kuruluş tarafından maddi olarak desteklenmemiş olup bireysel akademik çalışma kapsamında gerçekleştirilmiştir.

Lizge KORKMAZ

Manisa, 2025

KISALTMALAR DİZİNİ

Kısaltma	Açılımı
İHA	İnsansız Hava Aracı
İKA	İnsansız Kara Aracı
UAV	Unmanned Aerial Vehicle (İnsansız Hava Aracı)
UGV	Unmanned Ground Vehicle (İnsansız Kara Aracı)
MS	Meta-sezgisel (Metaheuristic)
PID	Oransal–İntegral–Türevsel Denetleyici
DOF	Degree of Freedom (Serbestlik Derecesi)
6 DOF	Altı Serbestlik Derecesi (x, y, z, φ, θ, ψ)
MIMO	Multi-Input Multi-Output (Çok Giriş–Çok Çıkış)
MDO	Multidisciplinary Design Optimization (Çok Disiplinli Tasarım Optimizasyonu)
ESR	Electroslag Remelting (Elektro Cüruf Yeniden Ergitme)
VE	

SİMGELER

Simgeler	Açıklamalar
τ	Tork (N·m)
ω	Açısal Hız (rad/s)
P	Güç (W)
x,y,z	Kartezyen Konum Bileşenleri (m)
ϕ	Yuvarlanma Açısı (roll)
θ	Yunuslama Açısı (pitch)
ψ	Yönelme Açısı (yaw)
Kp	PID Oransal Kazanç
Ki	PID İntegral Kazanç
Kd	PID Türevsel Kazanç
$e(t)$	Hata Sinyali
$u(t)$	Kontrol Sinyali
J	Maliyet / Amaç Fonksiyonu

MPC	Model Predictive Control (Model Öngörülü Kontrol)
LQR	Linear Quadratic Regulator (Doğrusal Karesel Düzenleyici)
LQI	Linear Quadratic Integral (İntegral İçeren LQR Denetleyici)
SMC	Sliding Mode Control (Kayan Kipli Kontrol)
MRAC	Model Reference Adaptive Control (Model Referanslı Uyarlamalı Kontrol)
YSA	Yapay Sinir Ağları
RL	Takviyeli Öğrenme (Reinforcement Learning)
GA	Genetic Algorithm (Genetik Algoritma)
PSO	Particle Swarm Optimization (Parçacık Sürüsü Optimizasyonu)
GWO	Grey Wolf Optimizer (Gri Kurt Optimizasyonu)
ABC	Artificial Bee Colony (Yapay Arı Kolonisi Algoritması)
ACO	Ant Colony Optimization (Karınca Kolonisi Optimizasyonu)
ACOR	Ant Colony Optimization for Continuous Domains (Sürekli Alanlar İçin Karınca Kolonisi Optimizasyonu)
ISR	Intelligence, Surveillance and Reconnaissance (İstihbarat, Gözetleme ve Keşif)
GSP	Gezgin Satıcı Problemi (İng. TSP – Travelling Salesman Problem)
TSP	Travelling Salesman Problem (Gezgin Satıcı Problemi)
IAE	Integral of Absolute Error (Mutlak Hata İntegrali)
ISE	Integral of Squared Error (Hata Kareleri İntegrali)
ITAE	Integral of Time-weighted Absolute Error (Zamanla Ağırlıklandırılmış Mutlak Hata İntegrali)
ITSE	Integral of Time-weighted Squared Error (Zamanla Ağırlıklandırılmış Hata Kareleri İntegrali)
CFD	Computational Fluid Dynamics (Hesaplamalı Akışkanlar Dinamiği)
FEA	Finite Element Analysis (Sonlu Elemanlar Analizi)
CAD	Computer-Aided Design (Bilgisayar Destekli Tasarım)
VR	Virtual Reality (Sanal Gerçeklik)

GIS *Geographic Information System* (Coğrafi Bilgi Sistemi)

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

İÇİNDEKİLER

1 GİRİŞ

Son yıllarda teknolojide yaşanan gelişmeler, insansız araçların günlük yaşamın pek çok alanında yaygınlaşmasına olanak tanımıştır. Başta savunma sanayisi olmak üzere lojistik, tarım, afet yönetimi ve arama-kurtarma operasyonları gibi farklı alanlarda insansız araçlara olan ilgi artmaya devam etmektedir. Bu araçlar, karmaşık görevleri başarıyla yerine getirebilme yeteneği sayesinde insan hayatını kolaylaştırmakta ve güvenliğini artırmaktadır.

İnsansız araçların kullanım alanlarının genişlemesi, mühendislik açısından yeni tasarımları ve yeni yaklaşımları da beraberinde getirmiştir. Özellikle son dönemde, tek gövdeli klasik araçlardan farklı olarak çok gövdeli (multi-body) insansız araçlar daha fazla ön plana çıkmaktadır. Çok gövdeli yapılar, birden fazla gövdenin ortak bir amaç doğrultusunda koordineli olarak hareket etmesini gerektirir. Bu yapıların en önemli avantajları arasında daha yüksek taşıma kapasitesi, gelişmiş denge yeteneği ve artan manevra kabiliyeti bulunmaktadır. Ancak, söz konusu avantajların yanı sıra çok gövdeli sistemlerin kontrolü de ciddi mühendislik zorluklarını beraberinde getirir.

Çok gövdeli insansız araçların kontrol edilmesi, tek bir gövdeye kıyasla daha karmaşık ve detaylı bir çalışma gerektirir. Her gövdenin hareketi, diğer gövdelerin hareketlerini doğrudan etkiler ve bu karşılıklı etkileşim, araçların stabilitesini ve performansını kontrol etmek için daha ileri seviye yöntemler gerektirir. Bu nedenle klasik kontrol yöntemlerinin yanında modern kontrol yaklaşımları, yapay zekâ temelli algoritmalar ve uyarlamalı kontrol teknikleri gibi gelişmiş kontrol stratejilerine ihtiyaç duyulur.

Bu tez çalışmasında, öncelikle insansız araçların çok gövdeli modellerinin kontrolü ve analizi konusu ele alınmıştır. Çalışmanın temel amacı, mevcut literatürde bulunan optimizasyon yöntemleri ile desteklenmiş kontrol yöntemlerinin sonuçlarını incelemek, kullanılan yöntemlerin avantaj ve dezavantajlarını ortaya koymak ve sistemlerin performanslarını farklı çalışma koşullarında değerlendirmektir. Tez kapsamında çeşitli kontrol algoritmaları simülasyon ortamında test edilerek sonuçları karşılaştırmalı olarak incelenmiş ve çok gövdeli sistemlerin etkin kontrolü için en uygun yöntemler belirlenmiştir.

Bu tez çalışmasının diğer bir amacı, mühendislik uygulamalarında çok gövdeli insansız araçların daha verimli, güvenli ve kararlı biçimde kullanılabilmesine yönelik katkılar sağlamaktır. Bu sebeple insansız araçlar için yol planlaması yapılarak yapay zekâ yöntemlerinden enerji ve zamandan tasarruf edilmesinde faydalanılmıştır. Ayrıca, konu üzerinde çalışmak isteyen araştırmacılara ve mühendislere kapsamlı bir literatür desteği sunulması ve gelecek çalışmalara yön vermesi amaçlanmıştır.

Bu tez çalışmasında, bir insansız hava aracı (İHA) ve insansız kara aracının (İKA) çok gövdeli modelleri kullanılarak dinamik analiz, kontrolör tasarım ve yol planlaması süreçleri ele alınmıştır. Çalışmanın kuramsal arka planında, meta-sezgisel (MS) yöntemlerin literatür taraması, PID tabanlı ayarlama yaklaşımları ve yol planlaması uygulamaları karşılaştırmalı olarak incelenmiştir. Uygulama bölümünde ise SolidWorks, MATLAB ve Simscape Multibody ortamlarında oluşturulan modeller üzerinden performans, kararlılık ve izleme kriterlerine dayalı değerlendirmeler yapılmıştır. Elde edilen bulgular, MS yöntemlerin çok gövdeli yapıların manevra kabiliyeti ve görev başarımı açısından kontrol tasarımına sunduğu katkıları ortaya koymaktadır.

2 GENEL BİLGİLER

Literatürde insansız araçlarda güç hesabı, kontrolör tasarımı ve yol planlamasına yönelik oldukça fazla çalışma bulunmaktadır. Döner mekanik sistemlerde güç hesabı, enerji üretim sistemlerinden makine tasarımlarına, motor performans analizlerinden türbin optimizasyonlarına kadar geniş bir mühendislik yelpazesinde kritik öneme sahiptir. Birçok kaynakta döner sistemlerde güç hesapları detaylı olarak incelenmiştir. Güç kavramı tork ve açısal hız arasındaki ilişki üzerinden tanımlanmış olup, sabit eksen etrafında dönen rijit cisimlerde güç; torkun açısal hız ile çarpımı olarak ifade edilmiştir. Döner sistemlerde iş ve güç hesaplamalarında temel prensip olan iş-enerji teoreminin uygulanması açıklanmış ve kinetik enerjinin açısal hız cinsinden tanımlanması ile ilgili formüller literatürde sunulmuştur. Ayrıca, güç hesabının doğru yapılabilmesi için, sisteme uygulanan dış kuvvetlerin dönme hareketi üzerindeki etkilerinin ve bu kuvvetlerin oluşturduğu torkların belirlenmesinin önemli olduğu vurgulanmıştır. Literatürde örnek olarak, sabit tork altında dönen bir çark ya da volanın açısal hızına ulaşma süreci incelenmiş, örnek problem çözümü ile sistemlerin dinamik davranışının nasıl analiz edileceği açık bir şekilde gösterilmiştir. (Young & Freedman, 2020) Böylece, güç ve enerji hesaplarının mühendislik uygulamalarında sistem performansını değerlendirmede temel bir analiz aracı olarak kullanılabileceği belirtilmiştir .

MS yöntemlerin mühendislik tasarımındaki tek ve çok amaçlı optimizasyon problemlerinde kullanılan meta-sezgisel algoritmaları kapsamlı bir şekilde literatürde sınıflandırmıştır. (Zhao ve Wang 2023) Çalışmada, genetik algoritma, PSO, diferansiyel evrim ve karınca kolonisi optimizasyonu gibi popülasyon tabanlı yöntemler ile tabu arama ve benzetilmiş tavlama gibi yerel arama temelli yaklaşımlar birlikte ele alınmıştır. Yazarlar, bu algoritmaların keşif–istismar dengesi, yakınsama davranışı ve kısıtlı tasarım alanlarındaki performanslarını karşılaştırmış; özellikle çeşitlilik koruma ve erken durağanlık tespitinin sonuç kalitesi üzerindeki etkisini vurgulamıştır. Bu inceleme, tez kapsamında kullanılacak meta-sezgisel yöntemlerin seçimi için genel bir çerçeve sunmuştur. Çalışma, kısıt yönetimi için üç ana paradigmayı öne çıkarmıştır: (i) Ceza fonksiyonları (sabit, adaptif, dinamik), (ii) onarım/yeniden-fezibilizasyon, (iii) fizibilite-öncelikli sıralama. Bu bağlamda,

problemin modellenmesinde farklı kısıt stratejileri ve çözümde genetik algoritma(GA), parçacık sürüsü optimizasyonu(PSO), diferansiyel evrim(DE) algoritmaları kullanılmıştır. Parametre ayarı bölümünde, sabit/el ile ayarlanan şemalara karşı kendini uyarlayan ve meta-öğrenmeli stratejilerin avantajları sunulmuştur.

Başka bir çalışmada, karmaşık ve kısıtlı optimizasyon problemleri için hibrit meta-sezgisel yaklaşımlar incelenmiştir. (Kumar ve Patel, 2022) GA'nın yerel arama yöntemleriyle, PSO'nun diferansiyel evrim ile birleştirildiği yapılar ele alınmıştır. Çalışmada, algoritmaların erken aşamalarda çözüm uzayını geniş tarama, orta ve geç aşamalarda ise çözümü iyileştirme performansları karşılaştırılmıştır. Hibrit modeller, kısıt yönetiminde "onarım" ve "fizibilite-öncelikli seçim" stratejilerini tercih ederek ceza fonksiyonu hassasiyetini düşürür; bu da özellikle ağır kısıtlı (gerilme, burkulma, üretilebilirlik) tasarımlarda yakınsamayı belirgin hızlandırır. Paralel ada mimarisi, göç ve heterojen parametre setleri ile birlikte erken çeşitlilik erozyonunu engeller.

Aynı zamanda senaryo-temelli değerlendirme (gürültü, parametre sapması, model belirsizliği) ve duyarlılık analizi entegrasyonuna yer verir. Sonuçlar, hibrit tasarımların çalışma süresi–çözüm kalitesi dengesinde saf yöntemlere üstün geldiğini göstermiştir.

Diğer bir çalışmada, disiplinlerarası tasarım optimizasyonu kapsamında MS algoritmaların etkinliği incelenmiştir. (Zhang ve Chen, 2024) Aerodinamik, yapısal, termal ve kontrol gibi birden fazla mühendislik disiplininin eş zamanlı olarak optimize edilmesi gerektiği karmaşık tasarım problemlerinde, klasik deterministik yöntemlerin yetersiz kaldığı vurgulanmaktadır. Çözüm uzayının yüksek boyutluluğu ve disiplinler arası etkileşimlerin neden olduğu doğrusal olmayan yapıların, meta-sezgisel yöntemlerle daha etkin yönetilebildiğini göstermiştir. Çalışmada GA, PSO, DE ve ABC gibi yöntemler karşılaştırılmış, özellikle uyarlanabilir mutasyon ve çok hedefli değerlendirme stratejilerinin yakınsama kalitesini artırdığı belirtilmiştir. Sonuçlar MS yöntemlerin yüksek boyutlu problemlerde klasik yöntemlere göre daha kararlı, daha hızlı ve daha çeşitli çözüm üretebildiğini göstermektedir.

Yapısal optimizasyonda topoloji/şekil/boyut kararlarının MS yöntemlerle çözüm üretilmesi literatürde ele alınmıştır. (Green ve White, 2023) Çalışmada, gerilme, yer değiştirme, burkulma ve doğal frekans kısıtlarını çok kademeli ceza ve onarım ile yönetilmiştir. Üretilebilirlik kapsamında minimum et kalınlığı, köprüleme, malzeme sürekliliği ve eklemeli imalat kuralları ek pratik kısıtlar olarak ele alınmıştır. Literatürde GA, PSO ve DE algoritmalarının basınç kabı, kaynaklı kiriş, ısı eşanjörü tasarımı gibi klasik mühendislik örnekleri üzerinden en iyi/ortalama/medyan uygunluk, başarı oranı, yakınsama eğrileri, çalışma süresi ve tekrarlanabilirlik ölçütleriyle karşılaştırması yapılmıştır. (Lee ve Martinez, 2024) Başlangıç popülasyonu çeşitliliği ve parametre adaptasyonu kritik belirleyiciler olarak tespit edilmiştir.

Başka bir çalışmada, PID, LQR vb. klasik denetleyici ayarlarının MS algoritmalarla sistematik biçimde optimize edilmesini ele alınmıştır. (Green ve Wilson, 2023) Yazarlar, izleme hatası, aşım, yerleşme süresi ve kontrol eforunu birlikte minimize eden çok amaçlı bir çerçeve tanımlamış ve akım, sürücü doygunluğu, eyleyici sınırları gibi kısıtlar için ceza terimleri kullanmıştır. Ayrık zamanlı model üzerinde nesil sayısı, nüfus büyüklüğü ve çaprazlama/mutasyon oranlarının duyarlılık analizi yapılarak sağlam ayar kümeleri elde edilmiştir. Sonuçlar, Ziegler–Nichols benzeri başlangıç ayarlarına kıyasla daha düşük aşım ve daha kısa yerleşme süresi üretirken kontrol sinyallerinin tepe değerlerini de sınırlı tutmuştur. Ayrıca, parametre belirsizliği ve dış bozucu etkiler altında performansın bozulmaması için çoklu senaryo tabanlı uygunluk değerlendirmesi önerilmiştir.

Yapılan bir çalışmada gerçek zaman kısıtları altında çalışan denetim döngülerine uygun hibrit bir yaklaşım sunulmuştur. Ağır hesaplama gerektiren küresel arama çevrim-dışı yapılmış, çevrim-içi aşamada ise hafif yerel iyileştirmeler ve kural tabanlı güncellemeler uygulanmıştır. (Johnson ve Brown, 2023) Zaman garantisi için popülasyon kırpma, erken durdurma, model indirgeme ve artımsal uygunluk hesabı kullanılmıştı. Böylece her örnekleme periyodunda tahsis edilen hesaplama bütçesi aşılmaması sağlanmıştır. Yazarlar, kayan ufuk ve periyodik yeniden-optimizasyon adımlarının hızlı referans değişimlerinde kararlılığı ve izleme kalitesini koruduğunu gösterir. Bozucu tahmini ve kısmi durum ölçümü senaryolarında, hibrit yöntem saf MS algoritmalara göre daha düşük gecikme ve

daha düzgün kontrol sinyali üretir. Çalışma ayrıca, gömülü donanımda uygulanabilirlik için bellek ayak izi, sabit-nokta aritmetik ve görev önceliklendirme gibi pratik mühendislik kılavuzları sunmuştur.

Birçok MH algoritmanın karşılaştırmasını içeren bir çalışmada kullanılan algoritmaların kontrol optimizasyonu bağlamında kararlılık, izleme hatası, kontrol eforu, hesaplama süresi ve ayar kolaylığı ölçütleriyle değerlendirilmiştir. (Anderson ve Green, 2024) PSO'nun ayar duyarlılığı, ataletsiz ve kısıtlı nüfuslarda performans dalgalanmalarına yol açabildiğinden, yazarlar hız sınırlandırma ve uyarlamalı atalet katsayısı kullanımı önerilmiştir. Ayrıca, çoklu başlatma, heterojen popülasyonlar ve problem özelliklerinin algoritma seçimiyle eşleştirilmesi için pratik bir karar şeması verilmiştir. Bulgular, tek bir "en iyi" algoritma yerine, hedef fonksiyonun yapısı ve gerçek zaman kısıtlarıyla uyumlu bir yöntem seçiminin daha kritik olduğunu göstermektedir.

MS yöntemler insanız araçların 2 ve 3 boyutlu yol A* ve D* gibi klasik planlayıcılarla melez biçimde kullanılmıştır. (Smith ve Zhao, 2022) Çalışmada, amaç fonksiyonları; yol uzunluğu, enerji/menzil, pürüzsüzlük (eğrilik), risk/engel mesafesi ve süre birlikte gözetilmiştir. Hareketli engeller için zaman-genişletilmiş uzay ve kayan ufuk yaklaşımıyla planlama yapılmıştır. Başka bir çalışmada, hibrit yol planlayıcıların çok amaçlı optimize etme stratejilerini karşılaştırılmıştır (Brown ve Wilson, 2022). Dinamik çevrelerde öncelik-temelli çakışma çözümü ve zaman pencereleri kullanılmış; çoklu araç kümesine görev atama + rota tanımlama eşzamanlı olarak hedeflenmiştir. (Johnson ve Green,2024) gerçek zaman kısıtında, sensör gürültüsü ve gecikmeler altında çalışan MS planlaması sunmuştur. Çalışmada hedef takibi, risk ve enerji terimleri birlikte optimize edilmiştir. Popülasyon kırpma, paralel ada ve artımlı uygunluk stratejileriyle süre garantisi elde edilmiştir. Çok ajanlı senaryolarda çakışma-önleme ve öncelik devretme mekanizmaları değerlendirilmiş; ROS tabanlı entegrasyon için pratik yönergeler ortaya koyulmuştur.

(Davis ve Anderson, 2023) PSO, DE, ACO, SA algoritmaları kullanılarak statik/dinamik engeller, 2B/3B arazi ve enerji-sınırlı platformlarda karşılaştırmıştır. Ölçütler yol kalitesi (uzunluk/süre/pürüzsüzlük), yeniden planlama sıklığı, yakınsama süresi olarak tanımlanmıştır. Bulgular, hibrit tohumlama + çok amaçlı

evrim kombinasyonlarının dengeli sonuç verdiğini, uyarlanabilir parametre ve çeşitli başlangıç stratejilerinin kritik olduğunu göstermiştir. Benzer çalışmalar için gerçek sistem doğrulamaları ve çevrimiçi yeniden planlama önerilmiştir.

(Joseph, 2022) tarafından PID kontrolör parametrelerinin (Kp, Ki, Kd) farklı MS algoritmalarla nasıl optimize edildiği tartışılmıştır. Yazar, GA, PSO ve ACO gibi yöntemleri hem teorik olarak açıklamıştır hem de gerçek endüstriyel süreçlerde test etmiştir. Çalışma, MS yöntemlerin klasik yöntemlere göre daha hızlı yakınsama sağladığını, daha düşük aşma ve kısa yerleşme süresi elde ettiğini vurgulamıştır. Özellikle ITAE hata kriteri bazında yapılan karşılaştırmalarda meta-sezgisel yöntemlerin çok daha tatmin edici sonuçlar verdiği gösterilmiştir.

(Bowthiya ve Rampriya, 2022) tarafından DC motor kontrolü için PID parametrelerinin optimizasyonu ele alınmıştır. Yazarlar, GA, PSO ve ACO algoritmalarını karşılaştırmıştır. Klasik Ziegler–Nichols yöntemi ile kıyaslandığında MS yöntemlerin aşma oranını ciddi şekilde düşürdüğü ve kararlı duruma ulaşma süresini kısalttığı görülmüştür. Özellikle PSO ve GA'nın, kararlılık ve hata minimizasyonu açısından daha öne çıktığı belirtilmiştir.

(Kim, Maruta ve Sugie, 2008) klasik PSO'nun geliştirilmiş versiyonu olan ALPSO (Augmented Lagrangian PSO) üzerine bir çalışma yapmıştır. Çalışmada önerilen yöntem, PID parametrelerinin çoklu kısıtlar altında optimize edilmesini sağlamaktadır. Sonuçlar, ALPSO'nun daha hızlı yakınsama sağladığını ve elde edilen PID kazançlarının parametre değişimlerine karşı daha sağlam olduğunu ortaya koymuştur. Çalışma özellikle karmaşık MIMO sistemlerde PSO tabanlı PID tasarımının etkinliğini göstermiştir.

(Taeib, Ltaeif ve Chaari, 2013) Takagi–Sugeno bulanık model tabanlı doğrusal olmayan MIMO sistemlere PSO temelli PID tasarımı yapılmıştır. PID parametreleri PSO algoritması ile optimize edilmiş ve sistem performansı Ziegler–Nichols ile karşılaştırılmıştır. Elde edilen sonuçlar, PSO'nun aşmayı azalttığını, daha kısa yükselme zamanı sağladığını ve sistemin bozucu etkilere karşı daha güçlü olduğunu göstermiştir. (Joyo, 2025) rehabilitasyon robotlarında kullanılan PID kontrol parametrelerinin farklı evrimsel algoritmalar ile optimize edilmesine odaklanmıştır. Çalışma, bu algoritmaların hasta güvenliği ve yumuşak hareket

kontrolü gibi medikal uygulamalarda performansını karşılaştırmıştır. Özellikle PSO ve GA tabanlı yöntemlerin daha hızlı tepki ve düşük hata oranı sağladığı ortaya konmuştur.

(Poudel ve Moh, 2023) biyolojik süreçlerden esinli optimizasyon algoritmalarının İHA yol planlamasında nasıl kullanıldığını özetlemiştir. Yazarlar, engel kaçınma, enerji verimliliği ve en kısa yol kriterlerini birlikte optimize eden çalışmaların önemine dikkat çekmiştir. PSO ve GWO algoritmalarının özellikle karmaşık 3B ortamlarda başarılı olduğu, ABC'nin daha çok enerji tüketimi odaklı çalışmalarda öne çıktığı belirtilmiştir.

(Shafiq ve ark., 2022) görev paylaşımı ve çarpışmadan kaçınma amacıyla Max–Min ACO'nun farklı mutasyon ve evrimsel stratejilerle hibrit versiyonlarını önermiştir. Çoklu İHA'ların görev paylaşımı ve çarpışmadan kaçınma probleminde test edilen yöntemler, klasik ACO'ya göre daha hızlı yakınsama göstermiştir. Önerilen hibritalgoritma hem yol uzunluğunu minimize etmiş hem de engel kaçınmada daha güvenilir sonuçlar vermiştir.

(Teng ve ark., 2025) geliştirilmiş Gri Kurt Optimizasyonu (GWO), İHA'ların en kısa yol bulma probleminde kullanmıştır. Klasik GWO'ya ek olarak Lens Opozisyon Tabanlı Öğrenme ve işbirlikçi avlanma mekanizmaları eklenmiştir. Simülasyon sonuçları, iyileştirilmiş GWO'nun klasik PSO ve ACO'ya göre daha kısa yollar bulduğunu ve daha hızlı hesaplama süreleri sunduğunu göstermiştir.

(Ghambari, 2024)'nın çalışmasında İHA yol planlamasının 3 boyutlu karmaşık ortamlar için bir optimizasyon problemi olduğu vurgulanmış, MS yöntemler ayrıntılı biçimde incelenmiştir. PSO'nun özellikle tehdit bölgelerinden kaçınma, yakıt tüketimi ve uçuş süresini minimize etmede başarılı olduğu, ACO'nun ise engel yoğunluğu fazla ortamlarda daha etkin olduğu belirtilmiştir.

(Poudel ve ark., 2023) İHA yol planlaması için PSO, GA ve HSA algoritmalarını hibritlemiştir. Sonuçlara göre, PSO-GA kombinasyonu yol uzunluğu açısından en iyi sonucu verirken, PSO-HSA enerji verimliliğinde daha üstün çıkmıştır. Çalışma, hibrit optimizasyonun UAV'lerde çoklu hedefleri aynı anda optimize etmede daha güçlü olduğunu göstermiştir.

Sonuç olarak, insansız araçların dinamik açıdan kontrolü ve yol planlaması için tek başına klasik yöntemlerin yeterli olmadığı, ancak bu yöntemlerin modern yöntemlerle desteklenmesi durumunda oldukça etkili sonuçlar doğurduğu görülmüştür. Bu sebeple bu tez çalışmasında MS yöntemler kullanılarak İHA'nın ve İKA'nın dinamik açıdan kontrolü ve yol planlamasında optimum sonuçlar elde edilmeye çalışılmıştır.

3 MATERYAL VE YÖNTEMLER

Bu bölümde, çok gövdeli İKA'nın ve İHA'nın modellenmesi, kontrolör tasarımı ve MS optimizasyon süreçlerinde kullanılan yöntemsel çerçeve sunulmaktadır. Tez kapsamında geliştirilen modeller, PID kontrolcü yapıları, MS optimizasyon mekanizmaları ve yol planlama yaklaşımları ayrıntılı biçimde açıklanmış; tezin özgün katkıları ortaya konmuştur.

İKA dinamik modelinin elde edilmesi için öncelikle Solidworks ortamında 3 boyutlu katı model elde edilmiştir. Solidworks ortamından XML dosyası olarak dışarı aktarılan katı model MATLAB/Simulink ortamında içe aktarılarak dinamik model edilmiştir. Model, diferansiyel tahrikli bir platformu esas almakta olup: lineer hız–açısal hız ilişkisi, tekerlek yarıçapı, gövde geometrisi, sürtünme etkileri ve arazi koşulları dikkate alınarak oluşturulmuştur. İKA'nın ilerleme ve dönüş hareketlerinde kullanılan tekerlek hızları, PID tabanlı bir hız kontrol döngüsü ile yönetilmiştir. İHA dinamik modelinin elde edilmesi için öncelikle Solidworks ortamında 3 boyutlu katı model elde edilmiştir. Dört rotorlu olarak tasarlanan İHA, altı serbestlik derecesine sahip doğrusal olmayan bir dinamik modele sahiptir. Gövde koordinatlarında doğrusal hızlar (u, v, w), açısal hızlar (p, q, r) ve Euler açıları aerodinamik kuvvet ve momentler itki modeli Simülasyon modeli MATLAB/Simulink üzerinde oluşturulmuş olup, kontrol algoritmalarının gerçekçi bir ortamda test edilmesini sağlanmıştır.

İKA'da tekerlek açısal hız kontrolü, dönüş kontrolü ve ilerleme hareketlerinin kontrolünde ve İHA'da yer alan yatay, dikey ve eksenler etrafındaki dönme hareketlerinin kontrolünde PID denetleyiciler kullanılmıştır. PID kontrolörün parametreleri oransal kazanç (Kp), integral kazancı (Ki) ve türev kazancı (Kd) şeklinde tanımlanmıştır. Kontrolör performanslarını değerlendirmek için literatürdeki dört hata ölçütü dikkate alınmıştır: ISE – Hata kareleri integrali, IAE – Mutlak hata

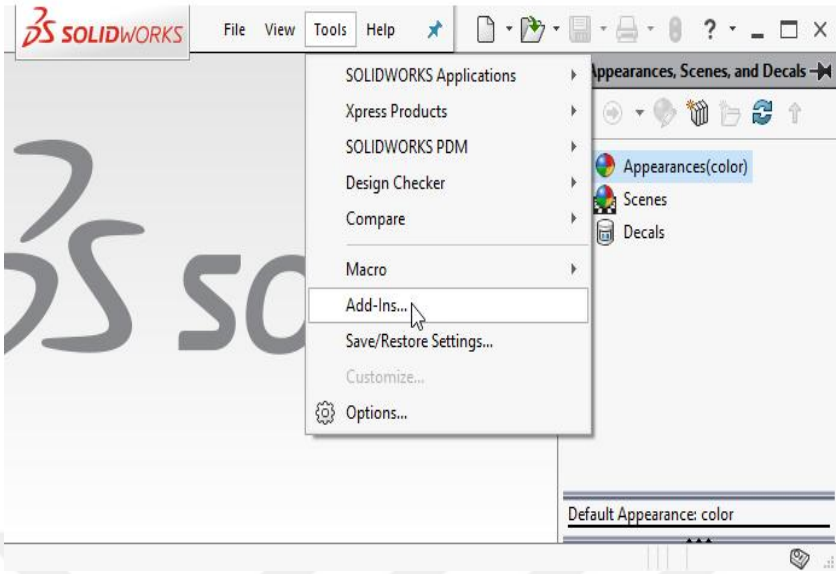
integrali, ITSE – Zaman ağırlıklı hata kareleri integrali, ITAE – Zaman ağırlıklı mutlak hata integrali. Bu tez kapsamında gerçekleştirilen optimizasyon süreçlerinin ana performans ölçütü IAE olarak seçilmiştir.

PID kontrolör parametrelerinin deterministik yöntemlerle ayarlanması çoğu zaman optimal sonuç vermez. Bu nedenle bu tezde, PID kontrolör katsayıları beş MH algoritma ile optimize edilmiştir: Genetik Algoritma (GA), Parçacık Sürü Optimizasyonu (PSO), Gri Kurt Optimizasyonu (GWO), Karınca Kolonisi Optimizasyonu (ACO), Yapay Arı Kolonisi (ABC). Her algoritma, PID parametrelerini belirlenen bir aralıkta aramış ve uygunluk fonksiyonu olarak IAE kullanılmıştır.

PID denetleyici parametreleri, meta-sezgisel algoritmalar tarafından başlangıçta rastgele veya belirli bir dağılıma göre oluşturulmuştur. Üretilen her bir parametre seti, ilgili Simulink modeli üzerinde çalıştırılmış ve model çıktısı kullanılarak IAE performans ölçütü hesaplanmıştır. Elde edilen IAE değeri, çözümün uygunluk fonksiyonu değeri olarak algoritmaya geri beslenmiştir. Algoritma, bu uygunluk değerlerinden yararlanarak yeni PID parametre adayları üretmiş ve süreç, 100 iterasyon boyunca ardışık şekilde sürdürülmüştür. Tüm nesiller sonunda en düşük IAE değerine sahip olan PID parametre seti, sistem için optimal çözüm olarak belirlenmiştir.

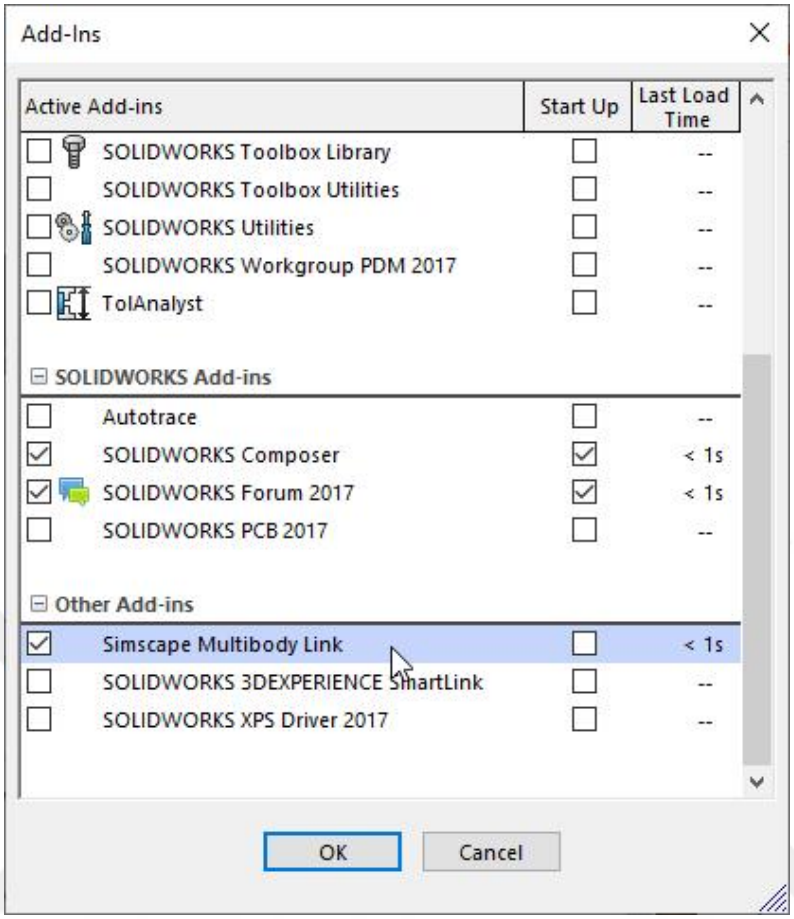
MS algoritmaları ile yapılan yol planlama optimizasyonunda GA, PSO, GWO, ACO ve ABC yöntemleri optimal yol uzunluğunu elde etmek amacıyla kullanılmıştır. Yol planlamada temel amaç fonksiyonu toplam yol uzunluğudur. Her bir algoritma, yol düğümlerini veya kontrol noktalarını birer kromozom ya da ajan olarak kodlamış ve çoklu hedefleri içeren, ağırlıklandırılmış birleşik bir uygunluk fonksiyonu üzerinden optimizasyon gerçekleştirmiştir. Çalışma kapsamında İHA için üç boyutlu ve İKA iki boyutlu ziyaret noktaları kullanılmıştır.

Şekil 1’de Solidworks ortamında Simscape eklenti ayarlarının yapılması gösterilmiştir.



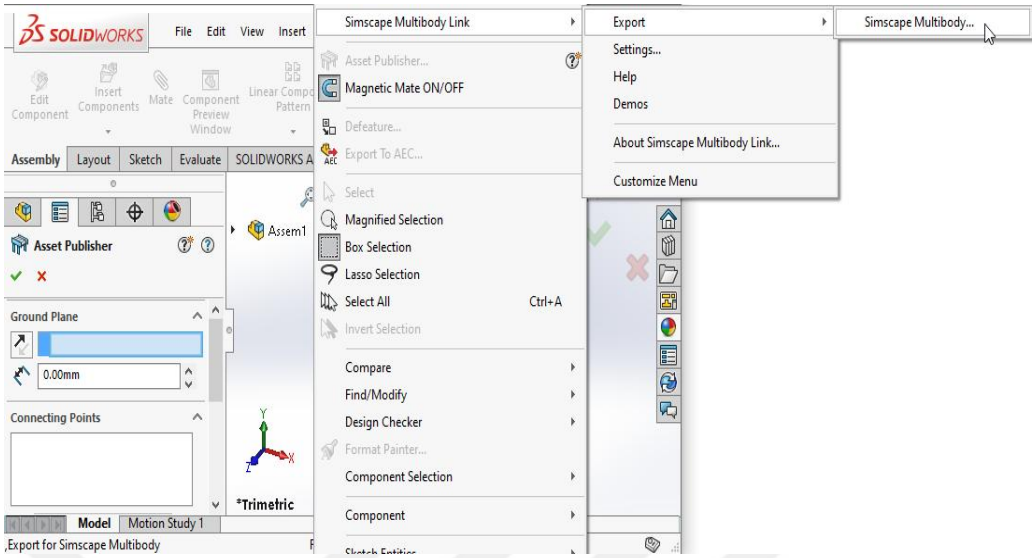
Şekil 1. Solidworks ortamında Simscape eklenti ayarlarının yapılması

Şekil 1 ‘de tarif edilen adım sonrasında açılan pencerede Simscape Multibody Link kutucuğu işaretlenir. Şekil 2’de Solidworks ortamında çoklu gövde eklentisinin seçilmesi gösterilmektedir.



Şekil 2. Solidworks ortamında çoklu gövde eklentisinin seçilmesi

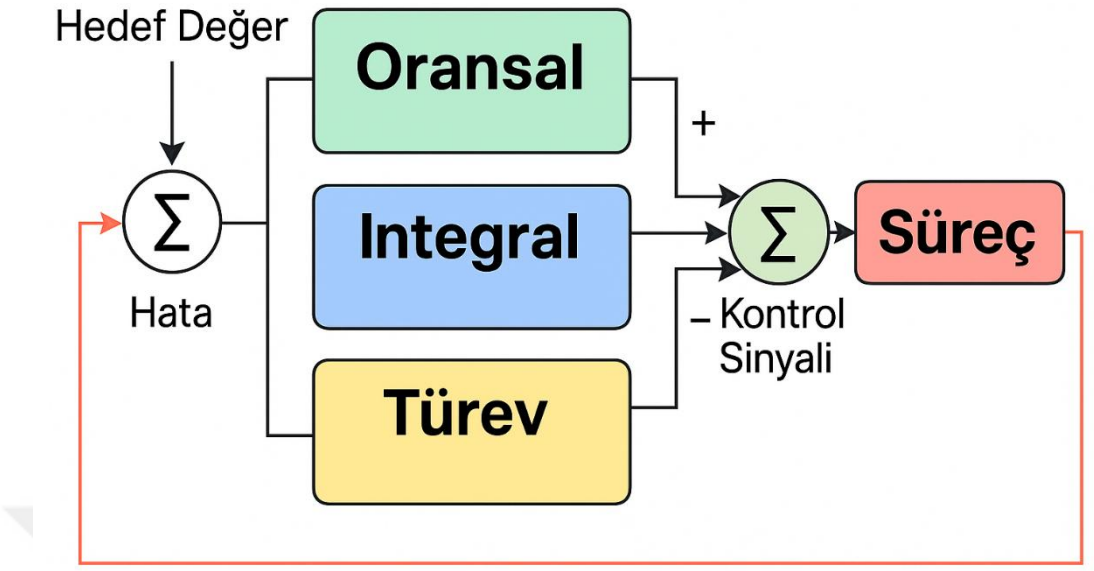
Yukarıdaki işlemler tamamlandığında, SolidWorks montaj dosyaları Simscape Multibody ortamına aktarılabilir hale gelir. MATLAB ortamında ise, komut satırına smlink_linksw komutu girilir. Bu entegrasyon, tasarım sürecinde oluşturulan modellerin dinamik analizlerini gerçekleştirmek ve sistem davranışlarını simüle etmek için etkili bir yöntem sunar. Şekil 3'te araç tasarımlarının Solidworks dışına aktarılması gösterilmektedir.



Şekil 3. Araç tasarımlarının Solidworks dışına aktarılması

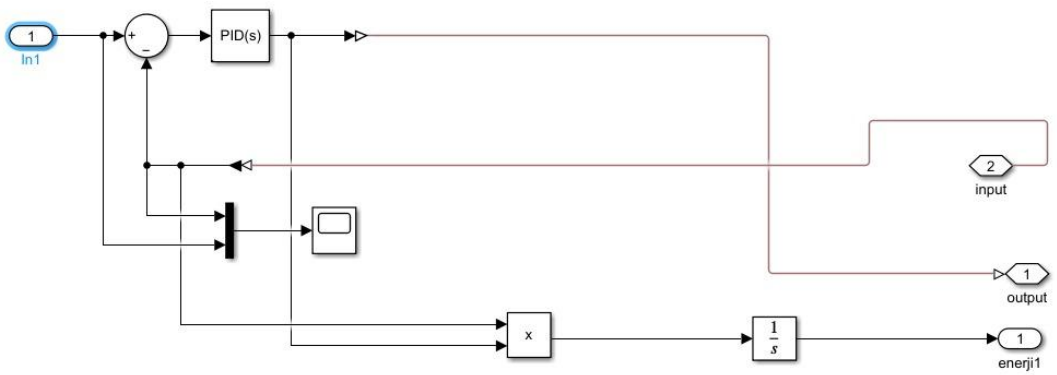
Çok gövdeli insansız araç sistemlerinin sağlıklı ve güvenli bir şekilde çalışabilmesi için kontrol algoritmalarının doğru bir şekilde modellenmesi ve uygulanması büyük önem taşımaktadır. Bu sistemler, birbirine bağlı birden fazla hareketli parçadan oluştuğu için, dengeyi sağlamak, yön belirlemek ve belirli görevleri yerine getirebilmek adına gelişmiş kontrol yöntemlerine ihtiyaç duyar. Bu bölümde hem klasik hem de modern modelleme ve kontrol yöntemleri incelenmiş; özellikle PID kontrol algoritması, doğrusal ve doğrusal olmayan regresyon tekniklerinin uygulamaları ele alınmıştır.

PID kontrolü, mühendislik uygulamalarında yaygın olarak tercih edilen bir kontrol mekanizmasıdır. Sistemden alınan hata sinyali üzerinden, hatayı minimize edecek şekilde oransal (P), integral (I) ve türevsel (D) bileşenleri kullanır. Bu bileşenlerin her biri sistemin davranışına farklı şekilde katkıda bulunur. Oransal (P) bileşen, mevcut hataya doğrudan tepki verir. İntegral (I) bileşen, zaman içerisinde biriken hataları düzeltmek için çalışır. Türevsel (D) bileşen, hatadaki değişim hızına bağlı olarak sistemi öngörülü şekilde yönlendirir. Şekil 4'te PID kontrolörün blok diyagramı verilmiştir.



Şekil 4. PID kontrolör blok diyagramı

MATLAB/Simulink ortamında PID kontrolün uygulanmasını sağlayan bir blok yer almaktadır. PID bloğu Simscape/Multibody modeli olarak tanımlanmış bir modelle çalışabilmektedir. Şekil 5'te PID kontrolörün Simulink ortamında uygulanması gösterilmektedir.

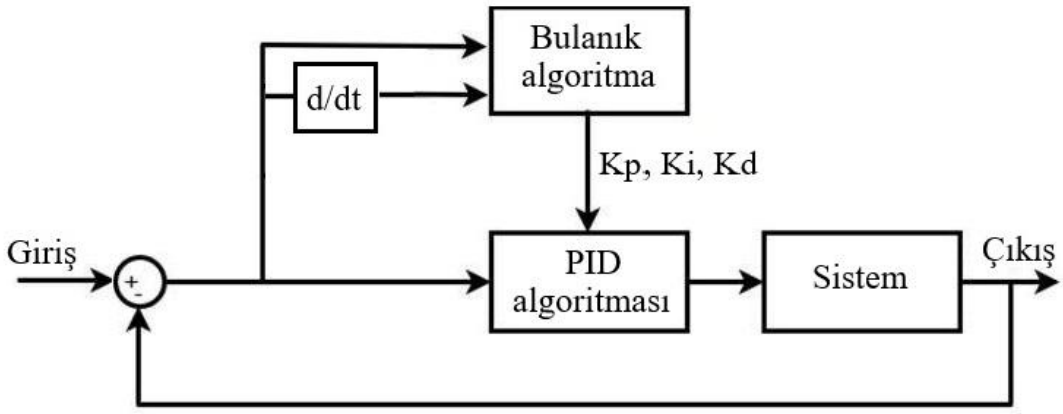


Şekil 5. PID kontrolörün Simulink ortamında uygulanması

Çok gövdeli insansız hava veya kara araçlarında, örneğin bir rotor sisteminin stabilitesini sağlamak ya da aracın yön değiştirme işlemlerini hassas bir şekilde gerçekleştirmek amacıyla PID kontrolü kullanılabilir. MATLAB/Simulink ortamında bu tür bir sistemin PID ile kontrol edilmesi, sistem cevabının görselleştirilmesi ve istenen parametrelere göre iyileştirme yapılması açısından oldukça elverişlidir.

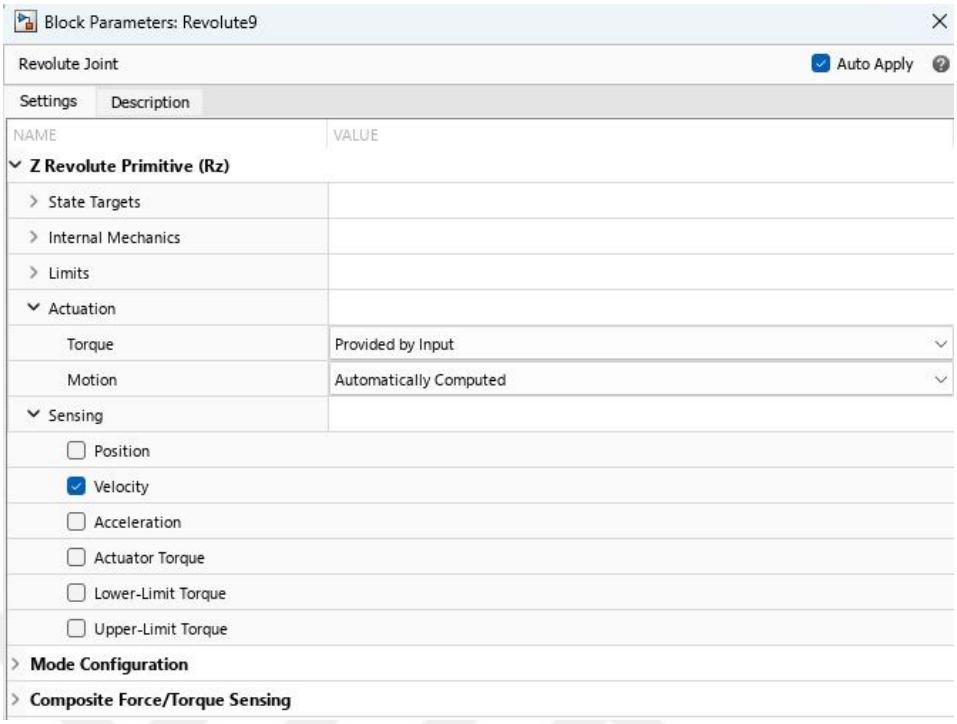
Gerçek dünya sistemlerinin büyük bir kısmı doğrusal olmaktan uzaktır. Özellikle çok gövdeli insansız araçlar gibi karmaşık sistemlerde, parçalar arası etkileşimler, çevresel etkiler ve dinamik yapılar doğrusal olmayan davranışlara neden olur. Bu nedenle daha gelişmiş modelleme ve kontrol tekniklerine ihtiyaç duyulur.

Doğrusal olmayan kontrolör tasarımları sistemlerin giriş-çıkış ilişkisini doğrusal olmayan fonksiyonlarla ifade etmeye dayanır. Bu tez çalışmasında İHA ve İKA modellerinin yatay düzelmede hareketinde konum kontrolünün daha iyi sağlanması için bulanık mantık destekli PID kontrolörler tercih edilmiştir. Şekil 6'da bulanık PID kontrolörün blok diyagramı gösterilmektedir.



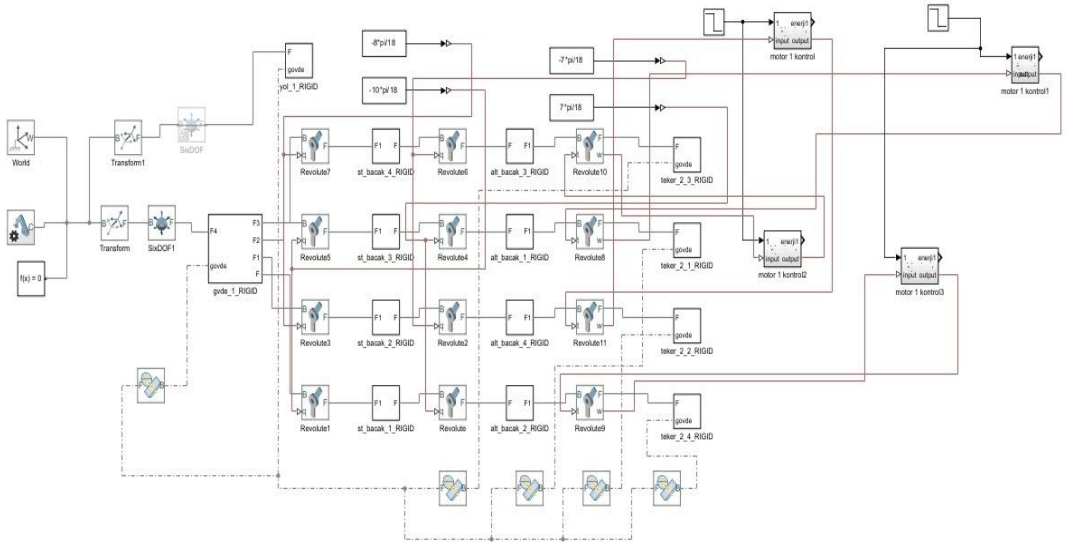
Şekil 6. PID kontrolörün Simulink ortamında uygulanması

PID kontrolörün Simscape/Multibody modeliyle en etkili ve gerçekçi çalışma şekli kontrolör çıkışının tork girdisi olarak mekanik sistemlerin mafsallarına girdi olarak verilmesidir. Şekil 7'de Simulink ortamında yapılan İKA mafsal ayarları gösterilmektedir.



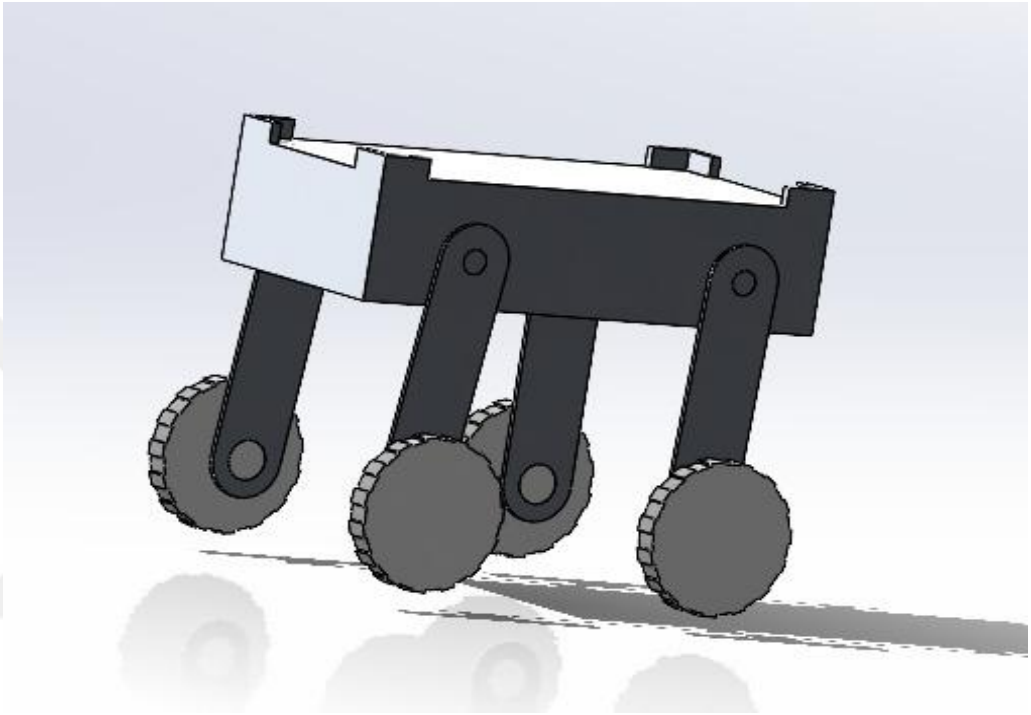
Şekil 7. Simulink ortamında mafsal ayarları

Solidworks ortamında modellenmiş bir tasarımın MATLAB/Simulink ortamında Simscape/Multibody olarak çalışması için kontrolör, çevreler bileşenlerle temas ayarları vs. yapıldıktan sonra simülasyonu yapılabilir. Şekil 8'de çok gövdeli modelin Simulink blok diyagramı gösterilmektedir.

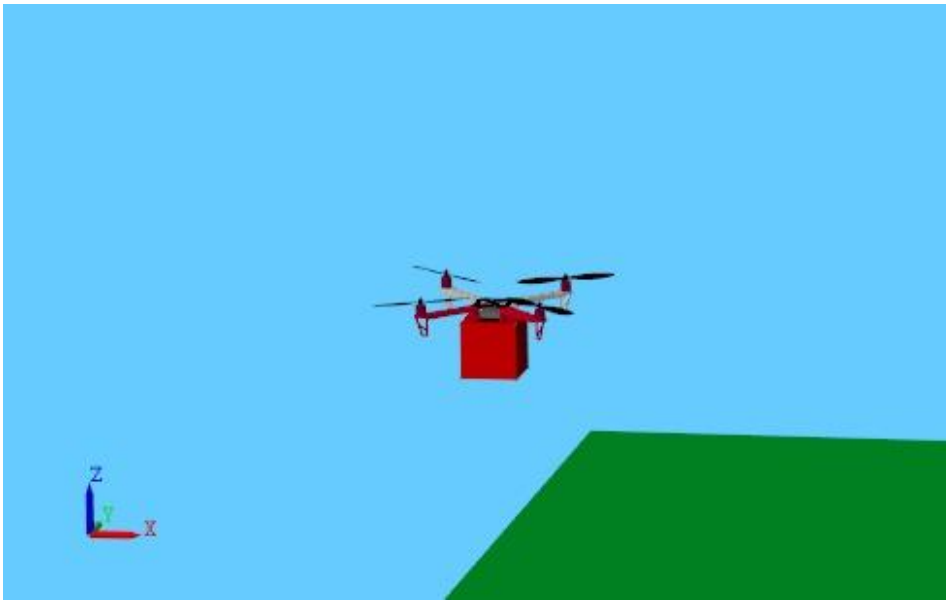


Şekil 8. Çok gövdeli modelin Simulink blok diyagramı

Bu tez çalışmasında tasarlanan İHA 1057 g ağırlığa sahip olup 4 m/s sabit hızla hareket edecek şekilde kontrolörleri tasarlanmıştır. Tasarlanan İKA 103 kg ağırlığa sahip olup 2 m/s sabit hızla hareket edecek şekilde kontrolörleri tasarlanmıştır. Şekil 9'da çok gövdeli İKA modeli, Şekil 10'da çok gövdeli İHA modeli gösterilmektedir.



Şekil 9. Çok gövdeli İKA modeli



Şekil 10. Çok gövdeli İHA modeli

4 ARAŞTIRMA BULGULARI VE TARTIŞMA

Bu tez çalışmasında İKA'nın tekerleklerinin hız kontrolü, doğrusal hareket kontrolü PID kontrolörler yardımıyla gerçekleştirilmiştir. Tablolarda elde edilen PID katsayıları, IAE performans kriterine göre elde edilen amaç fonksiyonu değerleri verilmiştir. Tablo 1'de İKA tekerleklerinin hız kontrolüne ait sonuçlar, Tablo 2'de İKA'nın doğrusal hareketine ait sonuçlar, Tablo 3'te İKA'nın dönme hareketine ait sonuçlar verilmiştir. İHA'ların kontrol performansının gözlemlenmesine ait örnek kodlar Ekler bölümünde sunulmuştur.

Tablo 1. İKA tekerleklerinin hız kontrolüne ait sonuçlar

Algoritma	PID Katsayıları	Amaç Fonksiyonu Değeri
GA	0.966, 0.133, 0.107	0.732
PSO	0.898, 0.129, 0.107	0.758
GWO	1.035, 0.102, 0.056	0.654
ABC	0.708, 0.135, 0.088	0.701
ACO	0.698, 0.140, 0.092	0.697

Tablo 2. İKA'nın doğrusal hareketine ait sonuçlar

Algoritma	PID Katsayıları	Amaç Fonksiyonu Değeri
GA	9.932, 0.427, 2.457	1.046
PSO	9.943, 0.229, 3.452	0.989
GWO	9.673, 0.245, 3.058	0.963
ABC	9.032, 0.037, 3.021	0.876
ACO	9.684, 0.262, 3.110	0.925

Tablo 3. İKA'nın dönme hareketine ait sonuçlar

Algoritma	PID Katsayıları	Amaç Fonksiyonu Değeri
GA	7.201, 0.133, 0.125	0.593
PSO	7.155, 0.140, 0.119	0.587
GWO	7.050, 0.103, 0.112	0.542
ABC	8.011, 0.123, 0.158	0.602
ACO	8.005, 0.126, 0.147	0.595

Bu tez çalışmasında İHA'nın 6 serbestlik derecesine ait 3 eksende doğrusal hareketlerinin ve bu eksenler üzerindeki dönem hareketlerinin kontrolü PID kontrolörler yardımıyla gerçekleştirilmiştir. Tablolarda elde edilen PID katsayıları, IAE performans kriterine göre elde edilen amaç fonksiyonu değerleri verilmiştir. Tablo 4'te İHA'nın x eksenindeki hareketine ait sonuçlar ait sonuçlar, Tablo 5'te İHA'nın y eksenindeki hareketine ait sonuçlar ait sonuçlar, Tablo 6'da İHA'nın z eksenindeki hareketine ait sonuçlar ait sonuçlar, Tablo 7'de İHA'nın yunuslama hareketine ait sonuçlar, Tablo 8'de İHA'nın yuvarlanma hareketine ait sonuçlar, Tablo 9'da İHA'nın yönelme hareketine ait sonuçlar ait sonuçlar verilmiştir.

Tablo 4. İHA'nın x eksenindeki hareketine ait sonuçlar

Algoritma	PID Katsayıları	Amaç Fonksiyonu Değeri
GA	2.235, 0.346, 0.478	0.367
PSO	2.305, 0.328, 0.445	0.372
GWO	2.314, 0.398, 0.541	0.389
ABC	2.045, 0.129, 0.454	0.303
ACO	2.321, 0.363, 0.428	0.354

Tablo 5. İHA'nın y eksenindeki hareketine ait sonuçlar

Algoritma	PID Katsayıları	Amaç Fonksiyonu Değeri
GA	1.946, 0.136, 0.378	0.366
PSO	1.955, 0.138, 0.386	0.387
GWO	1.905, 0.132, 0.367	0.335
ABC	2.012, 0.120, 0.365	0.389
ACO	1.958, 0.144, 0.390	0.368

Tablo 6. İHA'nın z eksenindeki hareketine ait sonuçlar

Algoritma	PID Katsayıları	Amaç Fonksiyonu Değeri
GA	0.164, 0.030, 0.503	0.337
PSO	0.171, 0.042, 0.538	0.342
GWO	0.152, 0.028, 0.475	0.313
ABC	0.154, 0.033, 0.521	0.332
ACO	0.148, 0.032, 0.530	0.329

Tablo 7. İHA'nın yunuslama hareketine ait sonuçlar

Algoritma	PID Katsayıları	Amaç Fonksiyonu Değeri
GA	8.749, 5.034, 40.985	0.547
PSO	8.865, 5.233, 38.946	0.556
GWO	8.504, 5.023, 40.032	0.523
ABC	8.806, 5.112, 40.985	0.572
ACO	8.868, 5.235, 39.032	0.579

Tablo 8. İHA'nın yuvarlanma hareketine ait sonuçlar

Algoritma	PID Katsayıları	Amaç Fonksiyonu Değeri
GA	8.752, 5.103, 40.985	0.551
PSO	8.878, 5.245, 39.939	0.562
GWO	8.515, 5.032, 39.934	0.538
ABC	8.959, 5.206, 40.825	0.569
ACO	9.034, 5.265, 40.933	0.577

Tablo 9. İHA'nın yönelme hareketine ait sonuçlar

Algoritma	PID Katsayıları	Amaç Fonksiyonu Değeri
GA	9.023, 5.231, 31.326	0.578
PSO	10.045, 5.203, 30.367	0.605
GWO	9.023, 5.231, 31.326	0.541
ABC	9.256, 5.367, 29.954	0.586
ACO	10.075, 5.223, 31.097	0.611

Bu tez çalışmasında, MS algoritmaları ile yapılan yol planlama optimizasyonunda GA, PSO, GWO, ACO ve ABC yöntemlerinden optimal yol uzunluğunu elde etmek amacıyla faydalanılmıştır. Yol planlamada tanımlanan tek amaç fonksiyonu toplam yol uzunluğudur. Çalışma kapsamında İHA için üç boyutlu ve İKA iki boyutlu ziyaret noktaları kullanılmıştır. Tablo 10'da İHA yol planlaması Örnek 1'e ait sonuçlar, Tablo 11'de İHA yol planlaması Örnek 2'ye ait sonuçlar, Tablo 12'de İKA yol planlaması Örnek 1'e ait sonuçlar, Tablo 13'te İKA yol planlaması Örnek 2'ye ait sonuçlar verilmiştir.Yol planlamasına ait örnek kodlar Ekler bölümünde sunulmuştur.

Tablo 10. İHA yol planlaması Örnek 1'e ait sonuçlar

Algoritma	Yol uzunluğu(metre)	Harcanan enerji(joule)
GA	985	256
PSO	969	247
GWO	969	247
ABC	969	247
ACO	985	256

Tablo 11. İHA yol planlaması Örnek 2'ye ait sonuçlar

Algoritma	Yol uzunluğu(metre)	Harcanan enerji(joule)
GA	1505	368
PSO	1489	351
GWO	1489	351
ABC	1505	368
ACO	1489	351

Tablo 12. İKA yol planlaması Örnek 1'e ait sonuçlar

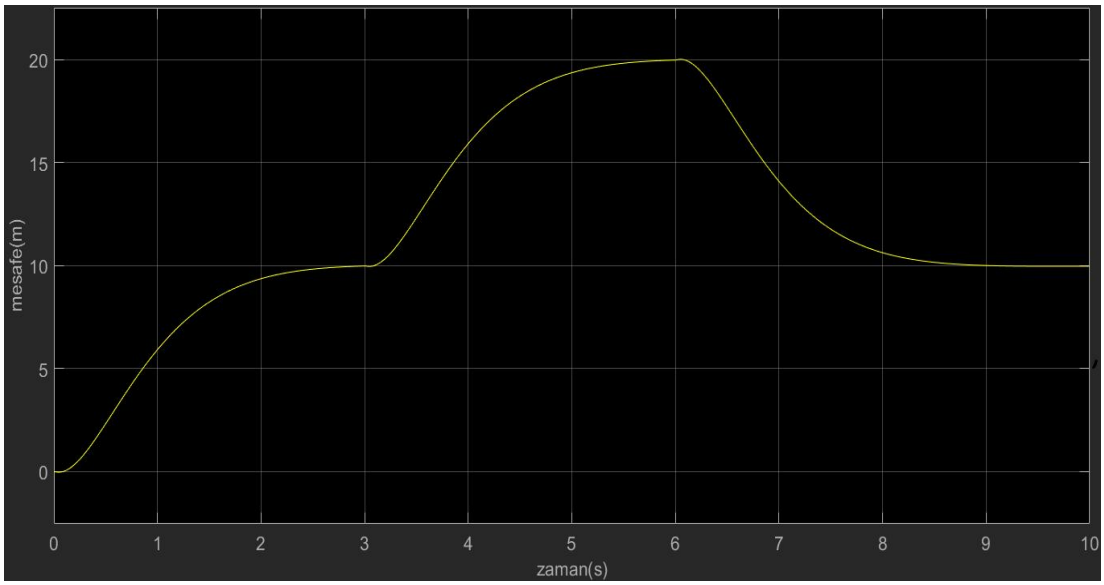
Algoritma	Yol uzunluğu(metre)	Harcanan enerji(joule)
GA	217	2503
PSO	209	2421
GWO	209	2421
ABC	209	2421

ACO	209	2421

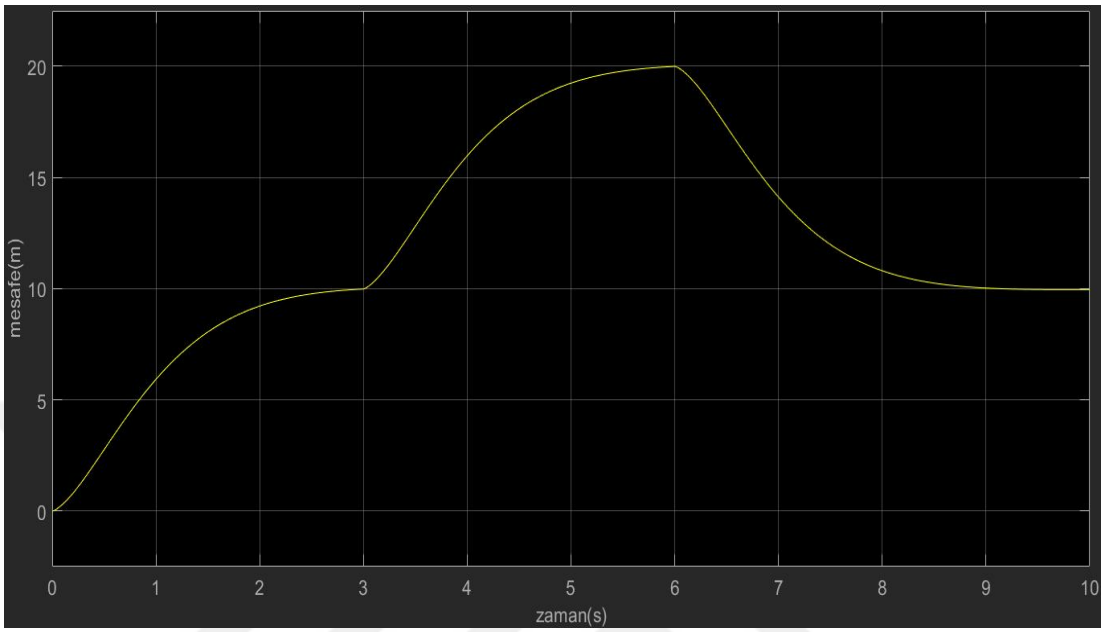
Tablo 13. İKA yol planlaması Örnek 2'ye ait sonuçlar

Algoritma	Yol uzunluğu(metre)	Harcanan enerji(joule)
GA	315	3658
PSO	315	3658
GWO	301	3612
ABC	301	3612
ACO	301	3612

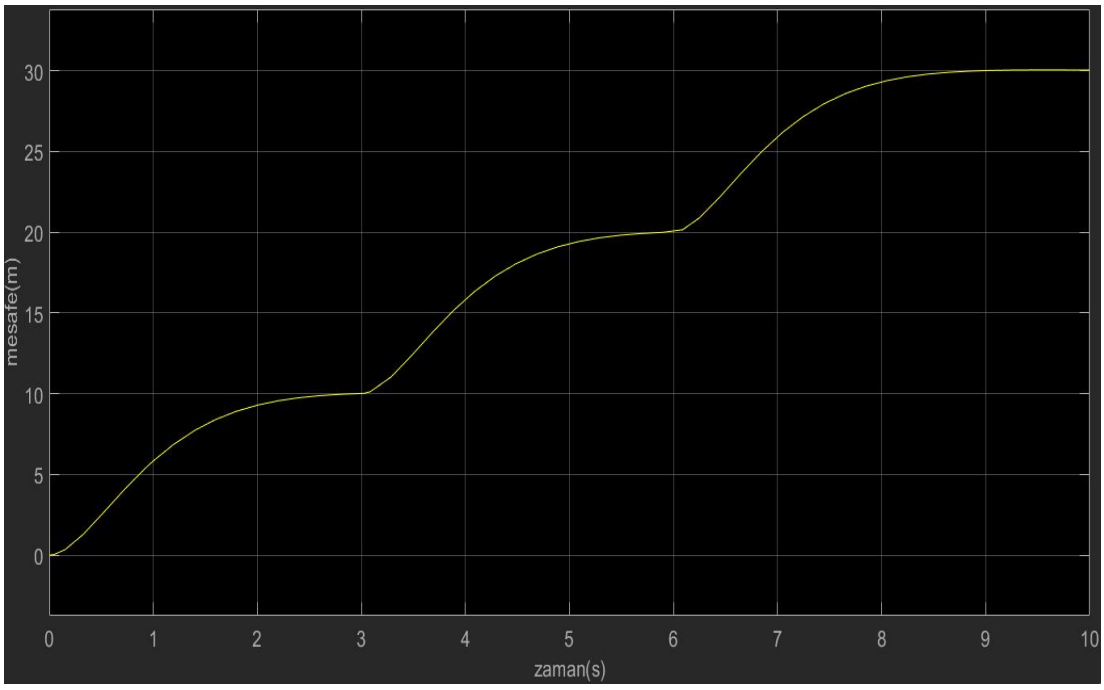
Tablo 1'den Tablo 13'e kadar sunulan sonuçlar dikkate alınarak en iyi sonuçları veren algoritmalarla gerçekleştirilen simülasyonlar sonucunda insansız araçların dinamik davranışına ve yol planlaması optimizasyonuna ait sonuçlar grafiksel olarak elde edilmiştir. Şekil 11'de İHA'nın x eksenindeki kontrol performansına ait grafik, Şekil 12'de İHA'nın y eksenindeki kontrol performansına ait grafik, Şekil 13'te İHA'nın z eksenindeki kontrol performansına ait grafik, Şekil 14'te İKA'nın doğrusal hareketindeki kontrol performansına ait grafik verilmiştir.



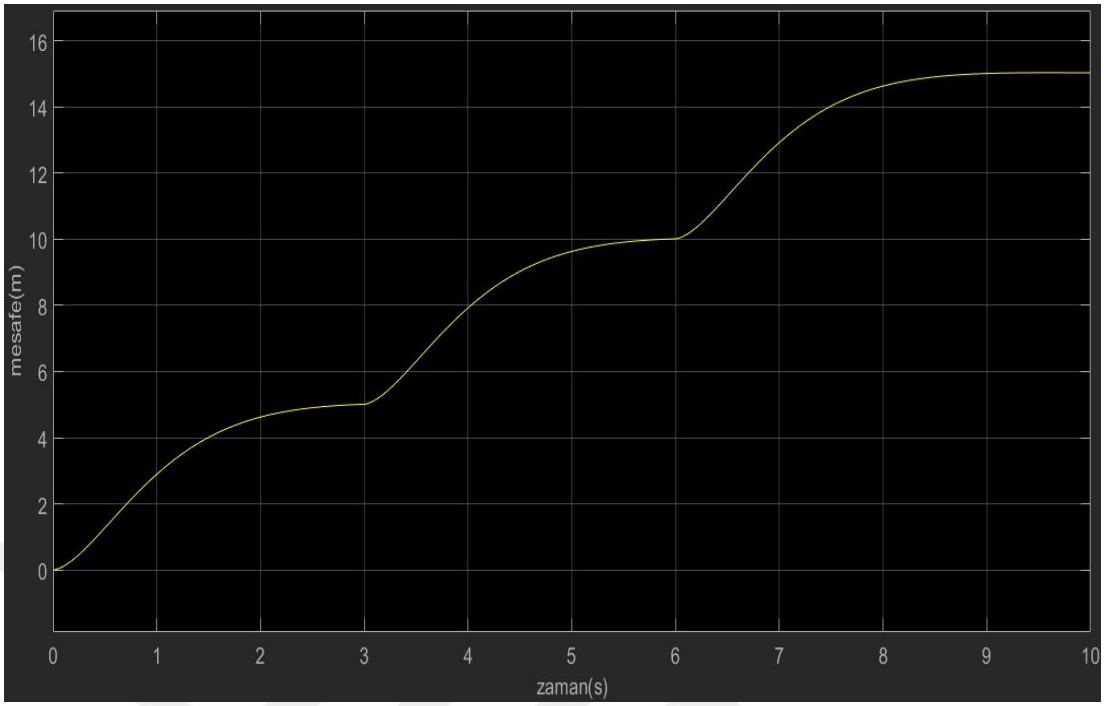
Şekil 11. İHA'nın x eksenindeki kontrol performansına ait grafik



Şekil 12. İHA'nın y eksenindeki kontrol performansına ait grafik

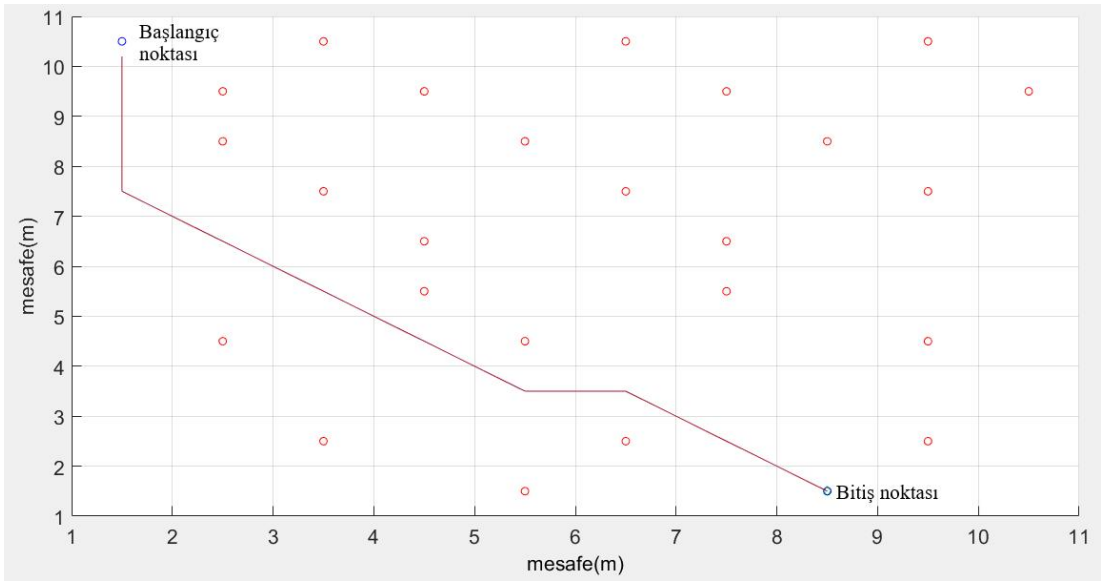


Şekil 13. İHA'nın z eksenindeki kontrol performansına ait grafik

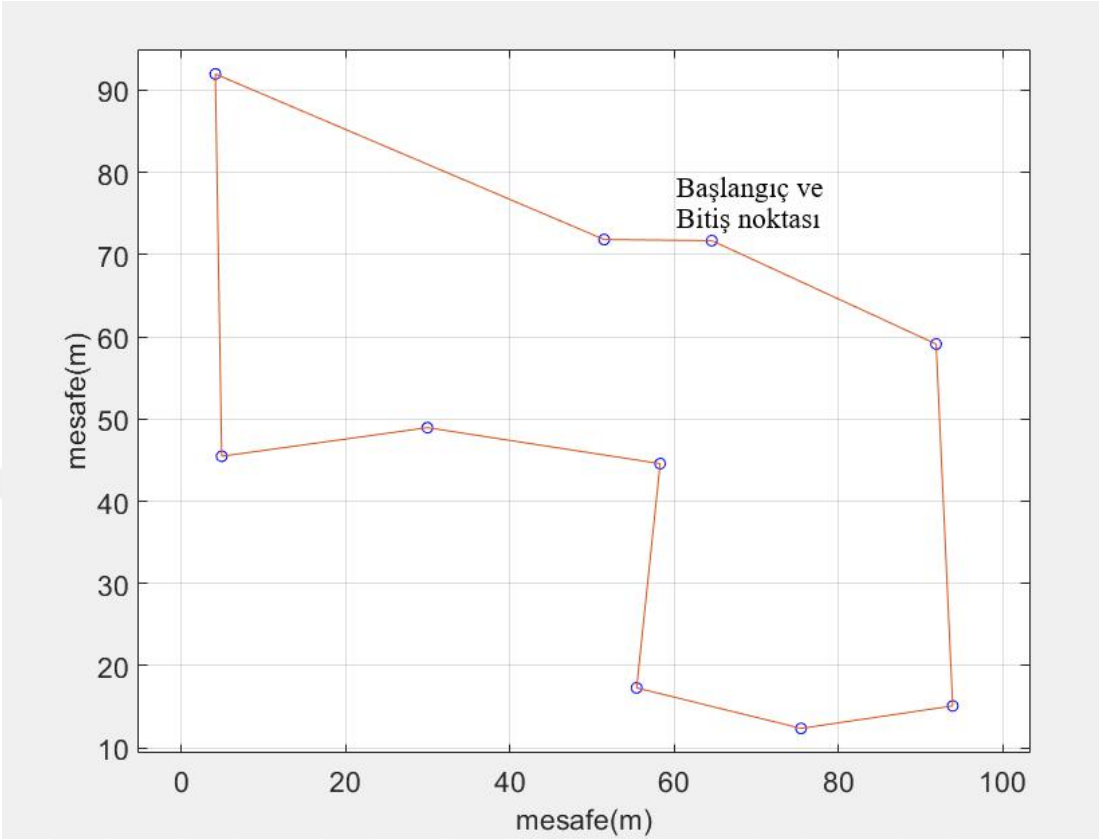


Şekil 14. İKA'nın doğrusal hareketindeki kontrol performansına ait grafik

Tablolarda elde edilen sonuçlar dikkate alınarak en iyi sonuçları veren algoritmalarla gerçekleştirilen simülasyonlar sonucunda yol planlaması optimizasyonuna ait sonuçlara ait şekiller elde edilmiştir. Şekil 15'te İKA yol planlamasına ait örnek yol, Şekil 16'da İKA yol planlamasına ait örnek yol verilmiştir.



Şekil 15. İKA’nın yol planlamasına ait örnek yol



Şekil 16. İHA’nın yol planlamasına ait örnek yörünge

5 SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında elde edilen sonuçlar dikkate alındığında İHA ve İKA'nın dinamik davranışında ve yol planlamasında MS algoritmaların aynı ve birbirine yakın sonuçlar üretebildiği gibi dikkate değer düzeyde farklı sonuçlar da üretebildiği görülmüştür.

Tablo 1'e ait sonuçlar incelendiğinde, amaç fonksiyonu bakımından en düşük maliyet değerine sahip olan GWO (0.654) algoritması, tekerlek hız kontrolü probleminde en başarılı yöntem olarak öne çıkmaktadır. GWO'nun elde ettiği bu değer, GA'nın 0.732, PSO'nun 0.758, ABC'nin 0.701 ve ACO'nun 0.697'lik değerleriyle karşılaştırıldığında anlamlı bir üstünlüğe işaret etmektedir. Nicel olarak bakıldığında GWO, GA'ya göre yaklaşık %10,66, PSO'ya göre %13,72, ABC'ye göre %6,69 ve ACO'ya göre %6,17 oranında daha düşük hata değeri üretmiştir. Bu farklar, GWO'nun hem keşif hem de sömürü yeteneğini dengeli bir biçimde kullanarak hata kriterini en düşük seviyeye indirdiğini göstermekte; özellikle GA ve PSO'ya karşı elde edilen çift haneli iyileşme oranları, bu problemde GWO'nun daha etkin bir arama mekanizması ortaya koyduğunu düşündürmektedir.

Tablo 2'ye ait sonuçlar, İKA'nın doğrusal hareket kontrolünde amaç fonksiyonunun en düşük olduğu algoritmanın ABC (0.876) olduğunu göstermektedir. Amaç fonksiyonu değeri hata miktarıyla doğrudan ilişkili olduğundan, ABC doğrusal hız kontrolünde diğer yöntemlere kıyasla en yüksek performansı sergilemiştir. Karşılaştırma yapıldığında GA'nın 1.046, PSO'nun 0.989, GWO'nun 0.963 ve ACO'nun 0.925 değerleri ürettiği görülmekte; bu durum ABC'nin GA'ya göre yaklaşık %16,25, PSO'ya göre %11,43, GWO'ya göre %9,03 ve ACO'ya göre %5,29 oranında daha iyi performans sunduğu anlamına gelmektedir. Elde edilen oranlar, ABC algoritmasının doğrusal hareket kontrolünde hatayı en etkili şekilde azaltan yöntem olduğunu ortaya koymakta; GA ve PSO gibi klasik popülasyon temelli yöntemlerin daha büyük hata değerleri üretmesi ise bu algoritmaların söz konusu problemde optimum PID katsayılarını bulmakta görece zayıf kaldığını göstermektedir.

Tablo 3 incelendiğinde, İKA'nın dönme hareketine ilişkin amaç fonksiyonu değerleri arasında en düşük değerin GWO (0.542) algoritması tarafından elde

edildiği, dolayısıyla GWO'nun bu problemde en yüksek performansı gösteren yöntem olduğu görülmektedir. Diğer algoritmalar sırasıyla GA için 0.593, PSO için 0.587, ABC için 0.602 ve ACO için 0.595 değerlerini üretmiştir. Bu sonuçlar, GWO'nun GA'ya göre yaklaşık %8,60, PSO'ya göre %7,67, ABC'ye göre %11,10 ve ACO'ya göre %8,91 oranında daha düşük hata sağladığını göstermektedir. Böylece GWO, dönme hareketinin kontrolünde hata değerini diğer tüm algoritmalara kıyasla daha etkin biçimde azaltmakta ve problemin dinamik yapısına en iyi uyum sağlayan yöntem olarak öne çıkmaktadır; PSO ve GA orta düzeyde performans sergilerken, ABC ve ACO görece daha yüksek hata değerlere ulaşmıştır. Bu sonuçlar, hızlı yön değişimi gerektiren dönme tipi kontrol problemlerinde GWO'nun daha kararlı ve daha hassas bir çözüm sunduğunu ortaya koymaktadır.

Tablo 4'te, İHA'nın x eksenindeki hareketi için kullanılan beş farklı algoritmanın performansı amaç fonksiyonu değerleri üzerinden karşılaştırılmıştır. En düşük amaç fonksiyonu değeri 0.303 ile ABC algoritmasına aittir ve bu sonuç ABC'nin tüm algoritmalar arasında en başarılı kontrolcü ayarını oluşturduğunu göstermektedir. Diğer algoritmalar GA için 0.367, PSO için 0.372, GWO için 0.389 ve ACO için 0.354 değerlerini üretmiştir. Bu değerler, ABC'nin GA'ya göre yaklaşık %17,45, PSO'ya göre %18,55, GWO'ya göre %22,13 ve ACO'ya göre %14,50 oranında daha iyi performans sunduğunu ortaya koymaktadır. ACO ve GA birbirine yakın sonuçlarla orta seviyede etkinlik gösterirken, PSO ve özellikle GWO daha yüksek hata değerleri ile daha zayıf bir performans sergilemiştir. Genel değerlendirmede ABC algoritmasının x ekseni kontrol problemini açık farkla en iyi çözen yöntem olduğu, İHA'nın doğrusal eksen hareketlerinde arama–sömürü dengesini başarılı biçimde kurabilen meta-sezgisel yaklaşımların daha yüksek kontrol performansı sağlayabildiği anlaşılmaktadır.

Tablo 5'te, İHA'nın y eksenindeki hareketine ait kontrol performansı amaç fonksiyonu değerleri üzerinden karşılaştırılmıştır. En düşük amaç fonksiyonu değeri 0.335 ile GWO algoritmasına aittir ve bu sonuç, y ekseni kontrolünde en başarılı algoritmanın GWO olduğunu göstermektedir. Diğer algoritmalar sırasıyla GA için 0.366, PSO için 0.387, ABC için 0.389 ve ACO için 0.368 değerlerini üretmiştir. Bu sayısal farklar dikkate alındığında GWO'nun GA'ya göre yaklaşık %8,5, PSO'ya göre %13,5, ABC'ye göre %13,9 ve ACO'ya göre %9,0 oranında daha düşük hata

değeri sağladığı görülmektedir. Dolayısıyla GWO, y eksenine ilişkin dinamiklerde hem hızlı yakınsama hem de kararlı hata azaltma açısından diğer yöntemlere üstünlük kurmakta; GA ve ACO orta düzeyde bir performans sergilerken, PSO ve ABC daha yüksek hata değerleriyle bu kontrol probleminde en zayıf sonuçları vermektedir.

Tablo 6'da, İHA'nın z eksenindeki hareket kontrolü için uygulanan meta-sezgisel algoritmaların performansları karşılaştırılmış, amaç fonksiyonu değeri toplam hata ölçütü olarak kullanılmıştır. GWO algoritması 0.313 ile en düşük değere ulaşarak en başarılı yöntem olmuştur. GA, PSO, ABC ve ACO için sırasıyla 0.337, 0.342, 0.332 ve 0.329 değerleri elde edilmiştir. Bu sonuçlar, GWO'nun GA'ya göre yaklaşık %7,12, PSO'ya göre %9,25, ABC'ye göre %5,72 ve ACO'ya göre %4,87 oranında daha düşük hata değeri ürettiğini göstermektedir. Böylece GWO, arama uzayını etkili biçimde tarayarak ve yerel minimumlardan kaçarak optimum PID katsayılarını belirleyebilmiş; ACO ve ABC orta düzeyde başarıya ulaşırken GA ve özellikle PSO daha yüksek hata değerleri üretmiştir. Hassas yükseklik kontrolü gerektiren uygulamalarda z ekseninde GWO'nun belirgin bir üstünlük sağladığı sonucuna ulaşılmaktadır.

Tablo 7'de, İHA'nın yunuslama (pitch) hareketini kontrol etmek amacıyla kullanılan beş meta-sezgisel algoritmanın performansları karşılaştırılmış ve amaç fonksiyonu değeri referans alınmıştır. GWO algoritması 0.523 ile en düşük değere ulaşarak diğer tüm yöntemlere kıyasla en başarılı performansı göstermiştir; GA, PSO, ABC ve ACO için sırasıyla 0.547, 0.556, 0.572 ve 0.579 değerleri elde edilmiştir. Bu bulgular, GWO'nun GA'ya göre yaklaşık %4,38, PSO'ya göre %5,93, ABC'ye göre %8,57 ve ACO'ya göre %9,68 oranında daha iyi performans sergilediğini göstermektedir. GA ve PSO birbirine yakın, kabul edilebilir düzeyde sonuçlar üretmesine karşın GWO'nun gerisinde kalmış; ABC ve özellikle ACO ise en yüksek hata değerleriyle en düşük kontrol başarımını ortaya koymuştur. Dolayısıyla yunuslama hareketinin kontrolünde GWO belirgin biçimde üstün bir yöntemi temsil etmekte, diğer algoritmalar ise daha yüksek hata oranları nedeniyle göreli olarak daha zayıf kalmaktadır.

Tablo 8, İHA'nın yuvarlanma (roll) dinamiğinin kontrolü için kullanılan beş meta-sezgisel algoritmanın karşılaştırmasını içermektedir. Amaç fonksiyonu değerleri incelendiğinde GWO algoritmasının 0.538 değeri ile en düşük sonucu ürettiği ve böylece yuvarlanma eksenindeki hata birikimini en etkili şekilde azaltan yöntem olduğu görülmektedir. GA, PSO, ABC ve ACO için sırasıyla 0.551, 0.562, 0.569 ve 0.577 değerleri elde edilmiştir. Bu sayısal farklar, GWO'nun GA'ya kıyasla yaklaşık %2,36, PSO'ya göre %4,27, ABC'ye göre %5,45 ve ACO'ya göre %6,75 oranında iyileşme sağladığını göstermektedir. GA ve PSO orta düzeyde performans gösterirken, ABC ve en yüksek hata değeriyle ACO daha düşük referans izleme doğruluğu sunmuştur. Sonuç olarak yuvarlanma ekseninde en etkin kontrol performansı GWO ile elde edilmekte, roll dinamiğinin doğrusalsızlık ve hassasiyet özellikleri karşısında GWO'nun keşif–sömürü dengesi diğer algoritmalara göre daha uygun bir yapı sergilemektedir.

İHA yol planlaması 2 senaryosu için elde edilen sonuçlar, algoritmaların ürettikleri güzergâh uzunlukları ve enerji tüketimleri bakımından belirgin bir gruplaşma gösterdiğini ortaya koymaktadır. PSO, GWO ve ACO algoritmaları 1489 metrelik en kısa güzergâhı üretmiş ve buna karşılık 351 J ile en düşük enerji tüketimini sağlamıştır. Buna karşın GA ve ABC algoritmaları 1505 metrelik daha uzun bir yol üretmiş ve 368 J enerji harcamıştır. Bu değerler, GA ve ABC'nin en iyi çözüme ulaşan algoritmalara göre yaklaşık %1,07 oranında daha uzun bir rota izlediğini ve yaklaşık %4,8 oranında daha fazla enerji tükettiğini göstermektedir. PSO, GWO ve ACO'nun hem yol hem enerji bakımından aynı optimum değerlere yakınsaması, bu algoritmaların problem kısıtları altında benzer ve etkili çözüm stratejileri geliştirdiğine işaret etmektedir. Genel olarak bu senaryoda PSO, GWO ve ACO en verimli algoritmalar olarak öne çıkarken, GA ve ABC daha maliyetli çözümler üretmiş ve parametre ayarlarının bu problem için yeniden gözden geçirilmesi gerektiğini düşündürmüştür.

Tablo 9'da, İHA'nın yönelme (yaw) eksenindeki kontrol performansı için beş meta-sezgisel algoritmanın amaç fonksiyonu değerleri karşılaştırılmıştır. En düşük değer 0.541 ile GWO algoritmasına aittir ve bu durum GWO'nun yönelme ekseninde diğer tüm algoritmalara göre daha kararlı, hızlı ve az hatalı bir kontrol tepkisi sunduğunu göstermektedir. GA, PSO, ABC ve ACO için sırasıyla 0.578,

0.605, 0.586 ve 0.611 değerleri elde edilmiştir. Buna göre GWO, GA'ya göre yaklaşık %6,4, PSO'ya göre %10,6, ABC'ye göre %7,7 ve ACO'ya göre %11,4 oranında iyileşme sağlamaktadır. PSO ve ACO, en yüksek amaç fonksiyonu değerleriyle en düşük performansı sergilerken GA ve ABC orta seviyede başarı göstermiştir. Genel değerlendirmede GWO, yönelme kontrolünde açık bir üstünlük sağlayarak İHA stabilitesine en fazla katkıyı sunan yöntem konumuna gelmiştir.

Tablo 10'da verilen sonuçlar, İHA'nın belirli bir güzergâh üzerinde en kısa yolu bulma ve rotayı en düşük enerji tüketimiyle tamamlama performansını ortaya koymaktadır. PSO, GWO ve ABC algoritmaları 969 metrelik yol ve 247 J'lük enerji tüketimi ile en iyi sonuçları üretmiş, böylece sistem için en düşük maliyetli rotayı sağlamıştır. GA ve ACO ise 985 metrelik yol ve 256 J'lük enerji tüketimiyle daha yüksek maliyetli çözümler üretmiştir. Sayısal olarak bakıldığında en başarılı üç algoritma, GA ve ACO'ya kıyasla yolu yaklaşık %1,63 oranında kısaltmış, enerji tüketimini ise yaklaşık %3,52 oranında azaltmıştır. Bu farklar küçük görünmekle birlikte, batarya kapasitesi ve görev süresinin kritik olduğu insansız hava aracı uygulamalarında önemli kazançlar sağlamaktadır.

Tablo 11'deki sonuçlar da benzer bir eğilimi göstermektedir. PSO, GWO ve ACO algoritmaları 1489 metrelik en kısa rotayı üretmiş ve 351 J enerji tüketmiştir; bu değerler hem yol uzunluğu hem de enerji açısından tablodaki en düşük sonuçlardır. GA ve ABC algoritmaları ise 1505 metrelik daha uzun bir güzergâh ve 368 J'lük daha yüksek enerji tüketimi ile optimumdan uzak kalmıştır. Bu durum, GA ve ABC'nin en iyi çözüme ulaşan yöntemlere göre yaklaşık %1,07 oranında daha uzun yol ve %4,84 oranında daha fazla enerji tüketimi ortaya koyduğunu göstermektedir. Dolayısıyla Tablo 11 kapsamında en verimli algoritmalar PSO, GWO ve ACO olurken, GA ve ABC göreceli olarak daha zayıf performans sergilemiştir.

Tablo 12'te İKA yol planlaması 1 senaryosu kapsamında elde edilen sonuçlar, algoritmalar arasında belirgin bir performans ayrışması olduğunu göstermektedir. GA algoritması, 217 metrelik bir güzergâh üretmiş olup bu değer, diğer algoritmaların ürettiği 209 metrelik yol uzunluğundan yaklaşık %3.83 daha fazladır. Yol uzunluğundaki bu artışa paralel olarak GA'nın enerji tüketimi de 2503 J ile en

yüksek değerdir. Diğer algoritmaların tükettiği 2421 J ile karşılaştırıldığında, GA'nın enerji harcamasının %3.38 daha yüksek olduğu görülmektedir. Bu bulgular GA'nın bu senaryo için optimum çözüme diğer yöntemlere göre daha uzak kaldığını göstermektedir.PSO, GWO, ABC ve ACO algoritmalarının hem yol uzunluğu hem de enerji tüketimi bakımından tamamen aynı sonuçları üretmiş olması dikkat çekicidir. Bu durum, söz konusu algoritmaların problem uzayını benzer biçimde değerlendirdiğini ve aynı optimum çözüm noktasına ulaştıklarını göstermektedir. Ayrıca bu dört algoritmanın aynı performans düzeyini göstermesi, problem yapısının sürü zekâsı ve popülasyon temelli arama stratejilerine uygun olduğunu düşündürmektedir.

Enerji tüketimi ile yol uzunluğu arasındaki doğrusal ilişki göz önünde bulundurulduğunda, PSO, GWO, ABC ve ACO algoritmalarının en kısa güzergâhı üreterek enerji açısından da en verimli çözümleri sunduğu anlaşılmaktadır. GA'nın daha uzun güzergâh ve daha yüksek enerji tüketimi üretmesi, algoritmanın arama mekanizmalarının mevcut problem için yeterince etkin çalışmadığını göstermektedir. Özellikle mutasyon ve seçilim operatörlerinin bu problemde erken yakınsamaya veya gereksiz sapmalara neden olduğu düşünülmektedir.Genel değerlendirme sonucunda, İKA yol planlaması 1 senaryosu için PSO, GWO, ABC ve ACO algoritmaları en başarılı yöntemler olarak öne çıkmaktadır. GA'nın görece düşük performansı, bu tip yol planlama problemlerinde parametre optimizasyonu ve popülasyon çeşitliliği gibi faktörlerin kritik önem taşıdığını bir kez daha ortaya koymaktadır.

Son olarak Tablo 13'te İKA yol planlaması 2 senaryosunda elde edilen bulgular, algoritmaların performansları arasında önceki senaryoya benzer biçimde belirgin bir ayrışma olduğunu göstermektedir. GA ve PSO algoritmaları 315 metrelik bir yol üretmiş olup bu değer, GWO, ABC ve ACO algoritmalarının elde ettiği 301 metrelik güzergâhtan yaklaşık %4.65 daha uzundur. Yol uzunluğundaki bu fark, enerji tüketim değerlerine de doğrudan yansımıştır. GA ve PSO algoritmalarının enerji tüketimi 3658 J seviyesindeyken, GWO, ABC ve ACO algoritmaları bu değeri 3612 J seviyesine düşürerek yaklaşık %1.26 oranında daha verimli sonuç üretmiştir.Bu sonuçlar GA ve PSO'nun bu senaryoda optimum çözümden bir miktar uzaklaştığını, özellikle ara nokta seçimlerinde daha az etkin bir yönlendirme gerçekleştirdiğini göstermektedir. GA'nın genetik operatörlerinin ve PSO'nun hız-

konum güncelleme stratejisinin problem uzayında daha uzun yollar üreten çözümlere yöneldiği anlaşılmaktadır. Buna karşın GWO, ABC ve ACO algoritmaları aynı yol uzunluğu ve aynı enerji tüketimi değerlerini üretmiş olup bu durum üç algoritmanın da problem uzayında benzer bir optimum bölgeye başarılı şekilde yakınsadığını göstermektedir. Enerji tüketimindeki küçük farklara rağmen, kısa yol uzunluğuna ulaşmada GWO, ABC ve ACO'nun belirgin biçimde daha etkin olduğu görülmektedir. Bu davranış, bu algoritmaların keşif–sömürü dengesi açısından daha dengeli bir strateji izlediğini ve lokal minimumlardan kaçınma konusunda daha başarılı olduklarını düşündürmektedir.Genel olarak değerlendirildiğinde, İKA yol planlaması 2 senaryosunda en verimli performansı GWO, ABC ve ACO algoritmaları sergilemiştir. GA ve PSO'nun daha uzun güzergâhlar üretmesi, bu tür yol planlama problemlerinde parametre ayarlamalarının, başlangıç popülasyonlarının ve algoritma iç dinamiklerinin kritik düzeyde önem taşıdığını ortaya koymaktadır.

6 EKLER:

Ek A : MATLAB PID Kontrolör Tasarımı Kod Örnekleri:

Ek A.1 GWO İle Simulink Tabanlı İKA Kontrolü İçin PID (Kp–Ki–Kd) Parametre Optimizasyonu (IAE Kriteri, [0–10] Aralığı)

```
clear;
close all;
clc;

SearchAgents_no = 10;
Max_iter = 30;
dim = 3;

lb = [0 0 0];
ub = [10 10 10];

[BestScore, BestPosition, Convergence_curve] = GWO(@fitness_PID, dim, SearchAgents_no, Max_iter, lb, ub);

fprintf('En iyi Kp: %.4f\n', BestPosition(1));
fprintf('En iyi Ki: %.4f\n', BestPosition(2));
fprintf('En iyi Kd: %.4f\n', BestPosition(3));
```

```
figure;
plot(Convergence_curve, 'LineWidth', 2);
xlabel('İterasyon');
ylabel('IAE');
title('GWO Optimizasyon Süreci');
grid on;

function cost = fitness_PID(params)

    Kp = params(1);
    Ki = params(2);
    Kd = params(3);

    assignin('base','Kp',Kp);
    assignin('base','Ki',Ki);
    assignin('base','Kd',Kd);

    try
        simOut = sim('ika_model', 'ReturnWorkspaceOutputs','on');

        % Önerilen: model içinde To Workspace bloğu ile e_ts kaydet
        % e_ts: timeseries (hata sinyali)
        if isprop(simOut,'e_ts') || isfield(simOut,'e_ts')
            e_ts = simOut.e_ts;
        else
```

```
        % ReturnWorkspaceOutputs ile genelde simOut.get('e_ts') çalışır
        e_ts = simOut.get('e_ts');
    end

    t = e_ts.Time;
    e = e_ts.Data;

    cost = trapz(t, abs(e));

    if ~isfinite(cost)
        cost = 1e12;
    end

  catch
    cost = 1e12;
  end
end

function [Alpha_score, Alpha_pos, Convergence_curve] = GWO(fobj, dim, SearchAgents_no, Max_iter, lb, ub)

Alpha_pos = zeros(1, dim);
Alpha_score = inf;

Beta_pos = zeros(1, dim);
```

```
Beta_score = inf;

Delta_pos = zeros(1, dim);
Delta_score = inf;

Positions = rand(SearchAgents_no, dim) .* (ub - lb) + lb;

Convergence_curve = zeros(1, Max_iter);

for l = 1:Max_iter
    for i = 1:SearchAgents_no

        Positions(i,:) = max(Positions(i,:), lb);
        Positions(i,:) = min(Positions(i,:), ub);

        fitness = fobj(Positions(i,:));

        if fitness < Alpha_score
            Delta_score = Beta_score;
            Delta_pos   = Beta_pos;

            Beta_score  = Alpha_score;
            Beta_pos    = Alpha_pos;

            Alpha_score = fitness;
```

```
        Alpha_pos   = Positions(i,:);
    elseif fitness < Beta_score
        Delta_score = Beta_score;
        Delta_pos   = Beta_pos;

        Beta_score  = fitness;
        Beta_pos    = Positions(i,:);
    elseif fitness < Delta_score
        Delta_score = fitness;
        Delta_pos   = Positions(i,:);
    end
end

a = 2 - l * (2 / Max_iter);

for i = 1:SearchAgents_no
    for j = 1:dim

        r1 = rand(); r2 = rand();
        A1 = 2 * a * r1 - a;
        C1 = 2 * r2;
        D_alpha = abs(C1 * Alpha_pos(j) - Positions(i,j));
        X1 = Alpha_pos(j) - A1 * D_alpha;

        r1 = rand(); r2 = rand();
```

```
            A2 = 2 * a * r1 - a;
            C2 = 2 * r2;
            D_beta = abs(C2 * Beta_pos(j) - Positions(i,j));
            X2 = Beta_pos(j) - A2 * D_beta;

            r1 = rand(); r2 = rand();
            A3 = 2 * a * r1 - a;
            C3 = 2 * r2;
            D_delta = abs(C3 * Delta_pos(j) - Positions(i,j));
            X3 = Delta_pos(j) - A3 * D_delta;

            Positions(i,j) = (X1 + X2 + X3) / 3;
        end
    end

    Convergence_curve(l) = Alpha_score;
    fprintf('İterasyon %d/%d: En iyi IAE = %.6f\n', l, Max_iter, Alpha_score);
end
end
```

Ek A.2 PSO İle Simulink Tabanlı İKA Kontrolü İçin PID (Kp–Ki–Kd) Parametre Optimizasyonu (IAE Kriteri, [0–10] Aralığı)

```
clear;
close all;
```

```
clc;

SearchAgents_no = 30;
Max_iter = 50;
dim = 3;

lb = [0 0 0];
ub = [10 10 10];

[BestScore, BestPosition, Convergence_curve] = PSO_PID(@fitness_PID, dim, SearchAgents_no, Max_iter, lb, ub);

fprintf('En iyi Kp: %.4f\n', BestPosition(1));
fprintf('En iyi Ki: %.4f\n', BestPosition(2));
fprintf('En iyi Kd: %.4f\n', BestPosition(3));

figure;
plot(Convergence_curve, 'LineWidth', 2);
xlabel('İterasyon');
ylabel('IAE');
title('PSO Optimizasyon Süreci');
grid on;

function cost = fitness_PID(params)
```

```
Kp = params(1);
Ki = params(2);
Kd = params(3);

assignin('base','Kp',Kp);
assignin('base','Ki',Ki);
assignin('base','Kd',Kd);

try
    simOut = sim('ika_model', 'ReturnWorkspaceOutputs','on');

    % Önerilen: model içinde To Workspace bloğu ile e_ts kaydet
    % e_ts: timeseries (hata sinyali)
    if isprop(simOut,'e_ts') || isfield(simOut,'e_ts')
        e_ts = simOut.e_ts;
    else
        e_ts = simOut.get('e_ts');
    end

    t = e_ts.Time;
    e = e_ts.Data;

    cost = trapz(t, abs(e));

    if ~isfinite(cost)
```

```
            cost = 1e12;
        end

    catch
        cost = 1e12;
    end
end

function [BestScore, BestPosition, Convergence_curve] = PSO_PID(fobj, dim, nPop,
Max_iter, lb, ub)

w  = 0.72;
c1 = 1.49;
c2 = 1.49;

X = rand(nPop, dim) .* (ub - lb) + lb;
V = zeros(nPop, dim);

Pbest = X;
PbestScore = inf(nPop, 1);

for i = 1:nPop
    PbestScore(i) = fobj(X(i,:));
end
```

```
[BestScore, idx] = min(PbestScore);
BestPosition = Pbest(idx,:);

Convergence_curve = zeros(Max_iter, 1);

for it = 1:Max_iter
    for i = 1:nPop

        r1 = rand(1, dim);
        r2 = rand(1, dim);

        V(i,:) = w * V(i,:) + c1 * r1 .* (Pbest(i,:) - X(i,:)) + c2 * r2 .* (BestPosition - X(i,:));
        X(i,:) = X(i,:) + V(i,:);

        X(i,:) = max(X(i,:), lb);
        X(i,:) = min(X(i,:), ub);

        score = fobj(X(i,:));

        if score < PbestScore(i)
            Pbest(i,:) = X(i,:);
            PbestScore(i) = score;
        end
```

```
        if PbestScore(i) < BestScore
            BestScore = PbestScore(i);
            BestPosition = Pbest(i,:);
        end
    end

    Convergence_curve(it) = BestScore;
    fprintf('İterasyon %d/%d: En iyi IAE = %.6f\n', it, Max_iter, BestScore);
end
end
```

Ek A.3 ABC İle Simulink Tabanlı İKA Kontrolü İçin PID (Kp–Ki–Kd) Parametre Optimizasyonu (IAE Kriteri, [0–10] Aralığı)

```
clear;
close all;
clc;

FoodNumber = 20;     % Besin kaynağı (koloni) sayısı
Max_iter   = 60;     % Maksimum iterasyon
dim        = 3;      % [Kp Ki Kd]

lb = [0 0 0];
ub = [10 10 10];

[BestScore, BestPosition, Convergence_curve] = ABC_PID(@fitness_PID, dim, FoodNumber, Max_iter, lb, ub);
```

```
fprintf('En iyi Kp: %.4f\n', BestPosition(1));
fprintf('En iyi Ki: %.4f\n', BestPosition(2));
fprintf('En iyi Kd: %.4f\n', BestPosition(3));

figure;
plot(Convergence_curve, 'LineWidth', 2);
xlabel('İterasyon');
ylabel('IAE');
title('ABC Optimizasyon Süreci');
grid on;

function cost = fitness_PID(params)

    Kp = params(1);
    Ki = params(2);
    Kd = params(3);

    assignin('base','Kp',Kp);
    assignin('base','Ki',Ki);
    assignin('base','Kd',Kd);

    try
        simOut = sim('ika_model', 'ReturnWorkspaceOutputs','on');

        % Model içinde To Workspace ile e_ts (timeseries) üretildi varsayımı
```

```
        if isprop(simOut,'e_ts') || isfield(simOut,'e_ts')
            e_ts = simOut.e_ts;
        else
            e_ts = simOut.get('e_ts');
        end

        t = e_ts.Time;
        e = e_ts.Data;

        cost = trapz(t, abs(e));

        if ~isfinite(cost)
            cost = 1e12;
        end

    catch
        cost = 1e12;
    end
end

function [BestScore, BestPosition, Convergence_curve] = ABC_PID(fobj, dim, FoodNumber, Max_iter, lb, ub)

SN = FoodNumber;          % Food sources
limit = 5 * dim;          % Scout limiti
```

```
Foods = rand(SN, dim) .* (ub - lb) + lb;   % Başlangıç çözümler
Cost  = zeros(SN,1);
trial = zeros(SN,1);

for i = 1:SN
   Cost(i) = fobj(Foods(i,:));
end

[BestScore, bestIdx] = min(Cost);
BestPosition = Foods(bestIdx,:);

Convergence_curve = zeros(Max_iter,1);

for it = 1:Max_iter

   % -------------------------
   % 1) Employed Bees
   % -------------------------
   for i = 1:SN
      k = i;
      while k == i
         k = randi(SN);
      end
```

```
    phi = -1 + 2*rand(1,dim);
    v = Foods(i,:) + phi .* (Foods(i,:) - Foods(k,:));

    v = max(v, lb);
    v = min(v, ub);

    vCost = fobj(v);

    if vCost <= Cost(i)
        Foods(i,:) = v;
        Cost(i) = vCost;
        trial(i) = 0;
    else
        trial(i) = trial(i) + 1;
    end
end

% -------------------------
% 2) Onlooker Bees
% -------------------------
fit = 1 ./ (1 + Cost);
prob = fit / sum(fit);

t = 0;
i = 1;
```

```
while t < SN
   if rand < prob(i)
      t = t + 1;

      k = i;
      while k == i
         k = randi(SN);
      end

      phi = -1 + 2*rand(1,dim);
      v = Foods(i,:) + phi .* (Foods(i,:) - Foods(k,:));

      v = max(v, lb);
      v = min(v, ub);

      vCost = fobj(v);

      if vCost <= Cost(i)
         Foods(i,:) = v;
         Cost(i) = vCost;
         trial(i) = 0;
      else
         trial(i) = trial(i) + 1;
      end
   end
```

```
    i = i + 1;
    if i > SN
        i = 1;
    end
end

% -------------------------
% 3) Scout Bees
% -------------------------
for i = 1:SN
    if trial(i) > limit
        Foods(i,:) = rand(1,dim) .* (ub - lb) + lb;
        Cost(i) = fobj(Foods(i,:));
        trial(i) = 0;
    end
end

% -------------------------
% 4) En iyi güncelle
% -------------------------
[curBest, curIdx] = min(Cost);
if curBest < BestScore
    BestScore = curBest;
    BestPosition = Foods(curIdx,:);
```

```
    end

    Convergence_curve(it) = BestScore;
    fprintf('İterasyon %d/%d: En iyi IAE = %.6f\n', it, Max_iter, BestScore);
end
end
```

Ek A.4 ACO (ACOR – Sürekli Karınca Kolonisi) İle Simulink Tabanlı İKA Kontrolü İçin PID (Kp–Ki–Kd) Parametre Optimizasyonu (IAE Kriteri, [0–10] Aralığı)

```
clear;
close all;
clc;

Nants   = 20;    % Her iterasyonda üretilecek yeni örnek sayısı
Max_iter= 60;    % Maksimum iterasyon
dim     = 3;     % [Kp Ki Kd]

lb = [0 0 0];
ub = [10 10 10];

[BestScore, BestPosition, Convergence_curve] = ACOR_PID(@fitness_PID, dim, Nants, Max_iter, lb, ub);

fprintf('En iyi Kp: %.4f\n', BestPosition(1));
fprintf('En iyi Ki: %.4f\n', BestPosition(2));
```

```
fprintf('En iyi Kd: %.4f\n', BestPosition(3));


figure;
plot(Convergence_curve, 'LineWidth', 2);
xlabel('İterasyon');
ylabel('IAE');
title('ACOR (Sürekli ACO) Optimizasyon Süreci');
grid on;


function cost = fitness_PID(params)

    Kp = params(1);
    Ki = params(2);
    Kd = params(3);

    assignin('base','Kp',Kp);
    assignin('base','Ki',Ki);
    assignin('base','Kd',Kd);

    try
        simOut = sim('ika_model', 'ReturnWorkspaceOutputs','on');

        % Model içinde To Workspace ile e_ts (timeseries) üretildi varsayımı
        if isprop(simOut,'e_ts') || isfield(simOut,'e_ts')
            e_ts = simOut.e_ts;
```

```
        else
            e_ts = simOut.get('e_ts');
        end

        t = e_ts.Time;
        e = e_ts.Data;

        cost = trapz(t, abs(e));

        if ~isfinite(cost)
            cost = 1e12;
        end

    catch
        cost = 1e12;
    end
end

function [BestScore, BestPosition, Convergence_curve] = ACOR_PID(fobj, dim, N, Max_iter, lb, ub)

% ACOR parametreleri
q  = 0.5;     % ağırlık yayılım parametresi
xi = 0.85;    % sigma ölçekleme
```

```
M = 2*N;      % arşiv boyutu (genelde 2N iyi çalışır)

% Başlangıç arşivi
A = rand(M,dim).*(ub-lb) + lb;
ACost = zeros(M,1);
for i=1:M
    ACost(i) = fobj(A(i,:));
end

% Sırala
[ACost, idx] = sort(ACost);
A = A(idx,:);

BestScore = ACost(1);
BestPosition = A(1,:);

% Ağırlıklar (rank tabanlı)
ranks = (1:M)';
w = (1/(q*M*sqrt(2*pi))) * exp(- (ranks-1).^2 / (2*(q*M)^2));
w = w / sum(w);

Convergence_curve = zeros(Max_iter,1);

for it = 1:Max_iter
```

```
% Yeni örnekler
S = zeros(N,dim);
for n=1:N
    for d=1:dim

        mu_d = A(:,d);

        % Her arşiv elemanı için sigma: liderden uzaklığa göre
        sigma_d = xi * abs(mu_d - mu_d(1));
        if all(sigma_d==0)
            sigma_d(:) = (ub(d)-lb(d))/50;
        end

        k = randsample(M,1,true,w);
        mu = mu_d(k);
        sd = sigma_d(k);

        S(n,d) = mu + sd*randn;
    end

    % sınır
    S(n,:) = max(S(n,:), lb);
    S(n,:) = min(S(n,:), ub);
end
```

```
    % Yeni örnek maliyetleri
    SCost = zeros(N,1);
    for n=1:N
        SCost(n) = fobj(S(n,:));
    end

    % Arşiv güncelle
    A = [A; S];
    ACost = [ACost; SCost];

    [ACost, idx] = sort(ACost);
    A = A(idx,:);

    A = A(1:M,:);
    ACost = ACost(1:M);

    BestScore = ACost(1);
    BestPosition = A(1,:);

    Convergence_curve(it) = BestScore;
    fprintf('İterasyon %d/%d: En iyi IAE = %.6f\n', it, Max_iter, BestScore);
end
end
```

Ek A.5 GA İle Simulink Tabanlı İKA Kontrolü İçin PID (Kp–Ki–Kd) Parametre Optimizasyonu (IAE Kriteri, [0–10] Aralığı)

```
% ========================================================
% GA İle Simulink Tabanlı İKA Tekerlek Hız Kontrolü İçin PID (Kp–Ki–Kd)
% Parametre Optimizasyonu (IAE Kriteri, [0–10] Aralığı)
% FILE: main_ga_pid_ika_wheelspeed.m
% ========================================================
clear; clc; close all

%% Kullanıcı ayarları
sim.modelName = 'ika_model';   % <-- Simulink model adınız (uzantısız)
sim.Tend      = 10;            % simülasyon süresi (s)

% PID aralığı
lb = [0 0 0];                  % [Kp Ki Kd]
ub = [10 10 10];

% GA ayarları (basit gerçek değerli GA)
npop  = 50;                    % popülasyon
maxIt = 120;                   % iterasyon
pc    = 0.8;                   % crossover olasılığı
pm    = 0.2;                   % mutasyon olasılığı
sigma = 0.10*(ub-lb);          % mutasyon std (GAUSS)
```

```
%% GA çalıştır
fitness = @(x) f_pid_iae_ika_ga(x, sim);

[bestX, bestCost, histCost] = ga_real_simple(lb, ub, npop, maxIt, fitness, pc, pm, sigma);

fprintf('\nGA -> En iyi IAE = %.6f\n', bestCost);
fprintf('En iyi PID: Kp=%.4f  Ki=%.4f  Kd=%.4f\n', bestX(1), bestX(2), bestX(3));

figure; plot(histCost,'LineWidth',1.8); grid on
xlabel('İterasyon'); ylabel('En iyi IAE'); title('GA – PID/IAE Optimizasyonu (İKA Tekerlek Hız Kontrolü)');

%% ============================================================
% UYGUNLUK FONKSİYONU (IAE)
% ============================================================
function J = f_pid_iae_ika_ga(x, sim)
% x = [Kp Ki Kd]  (0..10)

% sınır / NaN kontrol
if numel(x)~=3 || any(~isfinite(x)) || any(x<0) || any(x>10)
   J = 1e12; return
end

Kp = x(1); Ki = x(2); Kd = x(3);
```

```
% PID kazançlarını base workspace'e yaz (Simulink'te bu isimler kullanılmalı)
assignin('base','Kp',Kp);
assignin('base','Ki',Ki);
assignin('base','Kd',Kd);

try
    % Simulink çalıştır
    in = Simulink.SimulationInput(sim.modelName);
    in = setModelParameter(in,'StopTime',num2str(sim.Tend));
    out = sim(in);

    % Hata sinyalini çek:
    % Öncelik 1) logsout içinden e_ts
    % Alternatif 2) base workspace'te e_ts
    [t,e] = extract_error_ts(out);

    % IAE: Integral of Absolute Error
    J = trapz(t, abs(e));

    if ~isfinite(J) || J>1e9
        J = 1e9;
    end

catch
```

```
    % Simülasyon patlarsa büyük ceza
    J = 1e12;
end
end

function [t,e] = extract_error_ts(out)
% logsout varsa: e_ts (Timeseries) beklenir
t=[]; e=[];

% 1) logsout
try
    logs = out.logsout;
    sig  = logs.get('e_ts');          % <-- Simulink'te To Workspace / logsout ismi: e_ts
    if ~isempty(sig)
        t = sig.Values.Time;
        e = sig.Values.Data;
        return
    end
catch
end

% 2) base workspace
if evalin('base',"exist('e_ts','var')")
    ets = evalin('base','e_ts');       % timeseries
    t = ets.Time;
```

```
    e = ets.Data;
    return
end

error("Hata sinyali bulunamadı. Simulink'te 'e_ts' isimli timeseries üret (logsout veya base).");
end

%% ============================================================
% GERÇEK DEĞERLİ BASİT GA (tek dosyada)
% ============================================================
function [bestX,bestCost,histCost] = ga_real_simple(lb,ub,npop,maxIt,fitness,pc,pm,sigma)
dim = numel(lb);

% başlangıç popülasyonu
pop  = rand(npop,dim).*(ub-lb)+lb;
cost = zeros(npop,1);
for i=1:npop
    cost(i) = fitness(pop(i,:));
end

% elit
[bestCost,bi] = min(cost);
bestX = pop(bi,:);
```

```
histCost = zeros(maxIt,1);


for it=1:maxIt
    % sıralama
    [cost,idx] = sort(cost);
    pop = pop(idx,:);

    % elitizm: en iyi bireyi koru
    elite = pop(1,:);
    eliteCost = cost(1);

    % turnuva seçimi
    parents = zeros(npop,dim);
    for i=1:npop
        a = randi(npop); b = randi(npop);
        if cost(a) < cost(b), parents(i,:) = pop(a,:);
        else,                 parents(i,:) = pop(b,:);
        end
    end

    % çocuk üretimi
    kids = zeros(npop,dim);
    for k=1:2:npop
        p1 = parents(k,:);
        p2 = parents(min(k+1,npop),:);
```

```
    % crossover (aritmetik)
    if rand < pc
        alpha = rand(1,dim);
        c1 = alpha.*p1 + (1-alpha).*p2;
        c2 = alpha.*p2 + (1-alpha).*p1;
    else
        c1 = p1; c2 = p2;
    end

    % mutasyon (gauss)
    if rand < pm, c1 = c1 + sigma.*randn(1,dim); end
    if rand < pm, c2 = c2 + sigma.*randn(1,dim); end

    kids(k,:) = c1;
    if k+1<=npop, kids(k+1,:) = c2; end
end

% sınırlar
kids = max(min(kids,ub),lb);

% maliyetler
kcost = zeros(npop,1);
for i=1:npop
    kcost(i) = fitness(kids(i,:));
```

```
    end

    % yeni nesil: (mu+lambda) + elit
    pop  = [elite; kids];
    cost = [eliteCost; kcost];

    % en iyi güncelle
    [cmin,bi] = min(cost);
    if cmin < bestCost
        bestCost = cmin;
        bestX = pop(bi,:);
    end

    histCost(it) = bestCost;
    fprintf('İterasyon %d/%d: En iyi IAE = %.6f\n', it, maxIt, bestCost);
end
end
```

Ek B : İHA 6-DOF modeli için PID optimizasyonunda kullanılan beş algoritmalık meta-sezgisel çözüme ait kod örnekleri

Ek B.1. Gri Kurt Optimizasyon Algoritması (GWO) ile İHA PID Parametre Optimizasyonu

```
% ============================================================
% Ek B.1. Gri Kurt Optimizasyonu (GWO) ile İHA PID Parametre Optimizasyonu
% ============================================================
clear; clc; close all

sim.modelName = 'UAV_Model';   % İHA Simulink model adın
sim.Tend      = 10;
sim.useFallbackIfNoSimulink = true;
sim.refStep   = 1;

lb  = [0 0 0];
ub  = [10 10 10];
dim = 3;   % [Kp Ki Kd]

% GWO ayarları
nWolves = 30;
maxIt   = 100;

fitness = @(x) f_pid_iae_uav(x, sim);

fprintf('\n--- GWO ---\n');
[bestX, bestCost, histCost] = gwo(lb, ub, dim, nWolves, maxIt, fitness);
```

```
fprintf('GWO -> IAE=%.6f | PID=[%.4f %.4f %.4f]\n', bestCost, bestX(1), bestX(2),
bestX(3));

figure; plot(histCost,'LineWidth',1.6); grid on
xlabel('İterasyon'); ylabel('En iyi IAE'); title('GWO – İHA PID Optimizasyonu');

% -------------------- FITNESS: IAE (İHA) --------------------
function J = f_pid_iae_uav(pid, sim)
Kp = pid(1); Ki = pid(2); Kd = pid(3);
if any(~isfinite(pid)) || any(pid<0) || any(pid>50), J=1e12; return; end

try
    if ~sim.useFallbackIfNoSimulink
        assignin('base','Kp',Kp);
        assignin('base','Ki',Ki);
        assignin('base','Kd',Kd);

        out = sim(sim.modelName, 'StopTime', num2str(sim.Tend));

        % Modelden hata sinyali (etiketi 'e' varsayıldı)
        eSig = out.logsout.get('e').Values;
        t = eSig.Time; e = eSig.Data;

        % Eğer 6-DOF'ta çoklu hata varsa (ex,ey,ez gibi):
```

```
    % ex = out.logsout.get('ex').Values.Data; t = out.logsout.get('ex').Values.Time;
    % ey = out.logsout.get('ey').Values.Data; ez = out.logsout.get('ez').Values.Data;
    % J = trapz(t,abs(ex)) + trapz(t,abs(ey)) + trapz(t,abs(ez)); return;

else
    % Fallback: basit kapalı çevrim test (tezde Simulink ile çalıştır)
    dt = 0.001; t = 0:dt:sim.Tend;
    r  = sim.refStep*ones(size(t));
    wn = 3.5; zeta = 0.7;

    y = zeros(size(t)); yd = zeros(size(t));
    ei = 0; eprev = 0;

    for k=2:numel(t)
        e_k = r(k) - y(k-1);
        ei  = ei + e_k*dt;
        ed  = (e_k-eprev)/dt;

        u = Kp*e_k + Ki*ei + Kd*ed;

        ydd = wn^2*u - 2*zeta*wn*yd(k-1) - wn^2*y(k-1);
        yd(k) = yd(k-1) + ydd*dt;
        y(k)  = y(k-1)  + yd(k)*dt;

        eprev = e_k;
```

```
        end
        e = r - y;
    end

    J = trapz(t, abs(e));
    if ~isfinite(J), J=1e12; end
catch
    J = 1e12;
end
end

% -------------------- GWO --------------------
function [Alpha_pos,Alpha_score,histCost] = gwo(lb,ub,dim,nWolves,maxIt,fitness)
Alpha_pos = zeros(1,dim); Alpha_score = inf;
Beta_pos  = zeros(1,dim); Beta_score  = inf;
Delta_pos = zeros(1,dim); Delta_score = inf;

X = rand(nWolves,dim).*(ub-lb)+lb;
histCost = zeros(maxIt,1);

for it=1:maxIt
    for i=1:nWolves
        X(i,:) = max(min(X(i,:),ub),lb);
        score = fitness(X(i,:));
```

```
    if score < Alpha_score
        Delta_score = Beta_score;  Delta_pos = Beta_pos;
        Beta_score  = Alpha_score; Beta_pos  = Alpha_pos;
        Alpha_score = score;       Alpha_pos = X(i,:);
    elseif score < Beta_score
        Delta_score = Beta_score;  Delta_pos = Beta_pos;
        Beta_score  = score;       Beta_pos  = X(i,:);
    elseif score < Delta_score
        Delta_score = score;       Delta_pos = X(i,:);
    end
end

a = 2 - it*(2/maxIt);

for i=1:nWolves
    for j=1:dim
        r1=rand; r2=rand;
        A1=2*a*r1-a; C1=2*r2;
        D_alpha=abs(C1*Alpha_pos(j)-X(i,j));
        X1=Alpha_pos(j)-A1*D_alpha;

        r1=rand; r2=rand;
        A2=2*a*r1-a; C2=2*r2;
        D_beta=abs(C2*Beta_pos(j)-X(i,j));
        X2=Beta_pos(j)-A2*D_beta;
```

```
            r1=rand; r2=rand;
            A3=2*a*r1-a; C3=2*r2;
            D_delta=abs(C3*Delta_pos(j)-X(i,j));
            X3=Delta_pos(j)-A3*D_delta;

            X(i,j)=(X1+X2+X3)/3;
        end
    end

    histCost(it)=Alpha_score;
end
end
```

Ek B.2. Parçacık Sürü Optimizasyonu (PSO) ile İHA PID Parametre Optimizasyonu

```
            %
==========================================================
            % Ek B.2. PSO ile İHA PID Parametre Optimizasyonu
            %
==========================================================
            clear; clc; close all

            sim.modelName = 'UAV_Model';
            sim.Tend      = 10;
            sim.useFallbackIfNoSimulink = true;
            sim.refStep   = 1;
```

```
lb  = [0 0 0];
ub  = [10 10 10];
dim = 3;

% PSO ayarları
nPart = 40;
maxIt = 100;

fitness = @(x) f_pid_iae_uav(x, sim);

fprintf('\n--- PSO ---\n');
[bestX, bestCost, histCost] = pso(lb, ub, dim, nPart, maxIt, fitness);

fprintf('PSO -> IAE=%.6f | PID=[%.4f %.4f %.4f]\n', bestCost, bestX(1), bestX(2), bestX(3));

figure; plot(histCost,'LineWidth',1.6); grid on
xlabel('İterasyon'); ylabel('En iyi IAE'); title('PSO – İHA PID Optimizasyonu');

% -------------------- FITNESS: IAE (İHA) --------------------
function J = f_pid_iae_uav(pid, sim)
Kp = pid(1); Ki = pid(2); Kd = pid(3);
if any(~isfinite(pid)) || any(pid<0) || any(pid>50), J=1e12; return; end
```

```
try
    if ~sim.useFallbackIfNoSimulink
        assignin('base','Kp',Kp);
        assignin('base','Ki',Ki);
        assignin('base','Kd',Kd);

        out = sim(sim.modelName, 'StopTime', num2str(sim.Tend));

        eSig = out.logsout.get('e').Values; % hata sinyali etiketi
        t = eSig.Time; e = eSig.Data;

    else
        dt = 0.001; t = 0:dt:sim.Tend;
        r  = sim.refStep*ones(size(t));
        wn = 3.5; zeta = 0.7;

        y = zeros(size(t)); yd = zeros(size(t));
        ei = 0; eprev = 0;

        for k=2:numel(t)
            e_k = r(k) - y(k-1);
            ei  = ei + e_k*dt;
            ed  = (e_k-eprev)/dt;

            u = Kp*e_k + Ki*ei + Kd*ed;
```

```
            ydd = wn^2*u - 2*zeta*wn*yd(k-1) - wn^2*y(k-1);
            yd(k) = yd(k-1) + ydd*dt;
            y(k)  = y(k-1)  + yd(k)*dt;

            eprev = e_k;
        end
        e = r - y;
    end

    J = trapz(t, abs(e));
    if ~isfinite(J), J=1e12; end
catch
    J = 1e12;
end
end

% -------------------- PSO --------------------
function [bestX,bestCost,histCost] = pso(lb,ub,dim,nPart,maxIt,fitness)
w  = 0.72; c1 = 1.49; c2 = 1.49;
X  = rand(nPart,dim).*(ub-lb)+lb;
V  = zeros(nPart,dim);

P  = X; PC = inf(nPart,1);
for i=1:nPart, PC(i)=fitness(X(i,:)); end
```

```
[GC,gi] = min(PC); G = P(gi,:);


histCost = zeros(maxIt,1);
for it=1:maxIt
   for i=1:nPart
      r1=rand(1,dim); r2=rand(1,dim);
      V(i,:) = w*V(i,:) + c1*r1.*(P(i,:)-X(i,:)) + c2*r2.*(G-X(i,:));
      X(i,:) = X(i,:) + V(i,:);
      X(i,:) = max(min(X(i,:),ub),lb);


      C = fitness(X(i,:));
      if C < PC(i), P(i,:)=X(i,:); PC(i)=C; end
      if PC(i) < GC, G=P(i,:); GC=PC(i); end
   end
   histCost(it)=GC;
end
bestX=G; bestCost=GC;
end
```

Ek B.3. Yapay Arı Kolonisi (ABC) Algoritması ile İHA PID Parametre Optimizasyonu

```
%
=========================================================

% Ek B.3. ABC ile İHA PID Parametre Optimizasyonu
%
=========================================================

clear; clc; close all
```

```
sim.modelName = 'UAV_Model';
sim.Tend      = 10;
sim.useFallbackIfNoSimulink = true;
sim.refStep   = 1;

lb  = [0 0 0];
ub  = [10 10 10];
dim = 3;

% ABC ayarları
FoodNumber = 25;
maxIt      = 150;

fitness = @(x) f_pid_iae_uav(x, sim);

fprintf('\n--- ABC ---\n');
[bestX, bestCost, histCost] = abc(lb, ub, dim, FoodNumber, maxIt, fitness);

fprintf('ABC -> IAE=%.6f | PID=[%.4f %.4f %.4f]\n', bestCost, bestX(1), bestX(2), bestX(3));

figure; plot(histCost,'LineWidth',1.6); grid on
xlabel('İterasyon'); ylabel('En iyi IAE'); title('ABC – İHA PID Optimizasyonu');
```

```
% -------------------- FITNESS: IAE (İHA) --------------------
function J = f_pid_iae_uav(pid, sim)
Kp = pid(1); Ki = pid(2); Kd = pid(3);
if any(~isfinite(pid)) || any(pid<0) || any(pid>50), J=1e12; return; end

try
   if ~sim.useFallbackIfNoSimulink
      assignin('base','Kp',Kp);
      assignin('base','Ki',Ki);
      assignin('base','Kd',Kd);

      out = sim(sim.modelName, 'StopTime', num2str(sim.Tend));
      eSig = out.logsout.get('e').Values;
      t = eSig.Time; e = eSig.Data;

   else
      dt = 0.001; t = 0:dt:sim.Tend;
      r  = sim.refStep*ones(size(t));
      wn = 3.5; zeta = 0.7;

      y = zeros(size(t)); yd = zeros(size(t));
      ei = 0; eprev = 0;

      for k=2:numel(t)
         e_k = r(k) - y(k-1);
```

```
        ei  = ei + e_k*dt;
        ed  = (e_k-eprev)/dt;

        u = Kp*e_k + Ki*ei + Kd*ed;

        ydd = wn^2*u - 2*zeta*wn*yd(k-1) - wn^2*y(k-1);
        yd(k) = yd(k-1) + ydd*dt;
        y(k)  = y(k-1)  + yd(k)*dt;

        eprev = e_k;
      end
      e = r - y;
    end

    J = trapz(t, abs(e));
    if ~isfinite(J), J=1e12; end
catch
    J = 1e12;
end
end

% -------------------- ABC --------------------
function [bestX,bestCost,histCost]=abc(lb,ub,dim,SN,maxIt,fitness)
limit = 5*dim;
```

```
Foods = rand(SN,dim).*(ub-lb)+lb;
FCost = arrayfun(@(i) fitness(Foods(i,:)), 1:SN)';

trial = zeros(SN,1);
[bestCost,bi]=min(FCost); bestX=Foods(bi,:);
histCost=zeros(maxIt,1);

for it=1:maxIt
   % employed bees
   for i=1:SN
      k=i; while k==i, k=randi(SN); end
      phi = -1+2*rand(1,dim);
      v = Foods(i,:) + phi.*(Foods(i,:)-Foods(k,:));
      v = max(min(v,ub),lb);
      Cv = fitness(v);
      if Cv <= FCost(i)
         Foods(i,:)=v; FCost(i)=Cv; trial(i)=0;
      else
         trial(i)=trial(i)+1;
      end
   end

   % onlooker bees
   prob = (1./(1+FCost)); prob = prob/sum(prob);
   i=1; t=0;
```

```
while t<SN
   if rand < prob(i)
      t=t+1;
      k=i; while k==i, k=randi(SN); end
      phi = -1+2*rand(1,dim);
      v = Foods(i,:) + phi.*(Foods(i,:)-Foods(k,:));
      v = max(min(v,ub),lb);
      Cv = fitness(v);
      if Cv <= FCost(i)
         Foods(i,:)=v; FCost(i)=Cv; trial(i)=0;
      else
         trial(i)=trial(i)+1;
      end
   end
   i=i+1; if i>SN, i=1; end
end

% scout
for i=1:SN
   if trial(i) > limit
      Foods(i,:) = rand(1,dim).*(ub-lb)+lb;
      FCost(i)   = fitness(Foods(i,:));
      trial(i)=0;
   end
end
```

```
    % best update
    [c,bi]=min(FCost);
    if c<bestCost, bestCost=c; bestX=Foods(bi,:); end
    histCost(it)=bestCost;
  end
end
```

Ek B.4. Karınca Kolonisi Optimizasyonu (ACO/ACOR) ile İHA PID Parametre Optimizasyonu

```
%
=========================================================
% Ek B.4. ACO/ACOR ile İHA PID Parametre Optimizasyonu
%
=========================================================
clear; clc; close all

sim.modelName = 'UAV_Model';
sim.Tend      = 10;
sim.useFallbackIfNoSimulink = true;
sim.refStep   = 1;

lb  = [0 0 0];
ub  = [10 10 10];
dim = 3;

% ACO(ACOR) ayarları
```

```
N     = 25;    % yeni örnek sayısı
maxIt = 150;

fitness = @(x) f_pid_iae_uav(x, sim);

fprintf('\n--- ACO (ACOR) ---\n');
[bestX, bestCost, histCost] = acor(lb, ub, dim, N, maxIt, fitness);

fprintf('ACO(ACOR) -> IAE=%.6f | PID=[%.4f %.4f %.4f]\n', bestCost, bestX(1), bestX(2), bestX(3));

figure; plot(histCost,'LineWidth',1.6); grid on
xlabel('İterasyon'); ylabel('En iyi IAE'); title('ACO(ACOR) – İHA PID Optimizasyonu');

% -------------------- FITNESS: IAE (İHA) --------------------
function J = f_pid_iae_uav(pid, sim)
Kp = pid(1); Ki = pid(2); Kd = pid(3);
if any(~isfinite(pid)) || any(pid<0) || any(pid>50), J=1e12; return; end

try
   if ~sim.useFallbackIfNoSimulink
      assignin('base','Kp',Kp);
      assignin('base','Ki',Ki);
      assignin('base','Kd',Kd);
```

```
    out = sim(sim.modelName, 'StopTime', num2str(sim.Tend));
    eSig = out.logsout.get('e').Values;
    t = eSig.Time; e = eSig.Data;

else
    dt = 0.001; t = 0:dt:sim.Tend;
    r  = sim.refStep*ones(size(t));
    wn = 3.5; zeta = 0.7;

    y = zeros(size(t)); yd = zeros(size(t));
    ei = 0; eprev = 0;

    for k=2:numel(t)
        e_k = r(k) - y(k-1);
        ei  = ei + e_k*dt;
        ed  = (e_k-eprev)/dt;

        u = Kp*e_k + Ki*ei + Kd*ed;

        ydd = wn^2*u - 2*zeta*wn*yd(k-1) - wn^2*y(k-1);
        yd(k) = yd(k-1) + ydd*dt;
        y(k)  = y(k-1)  + yd(k)*dt;

        eprev = e_k;
    end
```

```
        e = r - y;
    end

    J = trapz(t, abs(e));
    if ~isfinite(J), J=1e12; end
catch
    J = 1e12;
end
end

% -------------------- ACOR (Sürekli ACO) --------------------
function [bestX,bestCost,histCost]=acor(lb,ub,dim,N,maxIt,fitness)
q  = 0.5; xi = 0.85;
M = 2*N;

A = rand(M,dim).*(ub-lb)+lb;
ACost = arrayfun(@(i) fitness(A(i,:)), 1:M)';

[ACost,idx] = sort(ACost);
A = A(idx,:);

ranks = (1:M)';
w = (1/(q*M*sqrt(2*pi))) * exp(- (ranks-1).^2 / (2*(q*M)^2));
w = w / sum(w);
```

```
histCost = zeros(maxIt,1);


for it=1:maxIt
    S = zeros(N,dim);
    for n=1:N
        for d=1:dim
            mu_d = A(:,d);
            sigma_d = xi * abs(mu_d - mu_d(1));
            if all(sigma_d==0), sigma_d(:) = (ub(d)-lb(d))/50; end
            k = randsample(M,1,true,w);
            S(n,d) = mu_d(k) + sigma_d(k)*randn;
        end
        S(n,:) = max(min(S(n,:),ub),lb);
    end

    SCost = arrayfun(@(i) fitness(S(i,:)), 1:N)';

    A = [A; S];
    ACost = [ACost; SCost];

    [ACost,idx] = sort(ACost);
    A = A(idx,:);

    A = A(1:M,:);
    ACost = ACost(1:M);
```

```
    histCost(it) = ACost(1);
end

bestX = A(1,:);
bestCost = ACost(1);
end
```

Ek B.5. Genetik Algoritma (GA) ile İHA PID Parametre Optimizasyonu

```
%
==========================================================

% Ek B.5. GA ile İHA PID Parametre Optimizasyonu
%
==========================================================

clear; clc; close all

sim.modelName = 'UAV_Model';
sim.Tend      = 10;
sim.useFallbackIfNoSimulink = true;
sim.refStep   = 1;

lb  = [0 0 0];
ub  = [10 10 10];
dim = 3;

% GA ayarları
```

```
npop  = 50;
maxIt = 120;

fitness = @(x) f_pid_iae_uav(x, sim);

fprintf('\n--- GA ---\n');
[bestX, bestCost, histCost] = ga_simple(lb, ub, dim, npop, maxIt, fitness);

fprintf('GA -> IAE=%.6f | PID=[%.4f %.4f %.4f]\n', bestCost, bestX(1), bestX(2), bestX(3));

figure; plot(histCost,'LineWidth',1.6); grid on
xlabel('İterasyon'); ylabel('En iyi IAE'); title('GA – İHA PID Optimizasyonu');

% -------------------- FITNESS: IAE (İHA) --------------------
function J = f_pid_iae_uav(pid, sim)
Kp = pid(1); Ki = pid(2); Kd = pid(3);
if any(~isfinite(pid)) || any(pid<0) || any(pid>50), J=1e12; return; end

try
   if ~sim.useFallbackIfNoSimulink
      assignin('base','Kp',Kp);
      assignin('base','Ki',Ki);
      assignin('base','Kd',Kd);
```

```
    out = sim(sim.modelName, 'StopTime', num2str(sim.Tend));
    eSig = out.logsout.get('e').Values;
    t = eSig.Time; e = eSig.Data;

else
    dt = 0.001; t = 0:dt:sim.Tend;
    r  = sim.refStep*ones(size(t));
    wn = 3.5; zeta = 0.7;

    y = zeros(size(t)); yd = zeros(size(t));
    ei = 0; eprev = 0;

    for k=2:numel(t)
        e_k = r(k) - y(k-1);
        ei  = ei + e_k*dt;
        ed  = (e_k-eprev)/dt;

        u = Kp*e_k + Ki*ei + Kd*ed;

        ydd = wn^2*u - 2*zeta*wn*yd(k-1) - wn^2*y(k-1);
        yd(k) = yd(k-1) + ydd*dt;
        y(k)  = y(k-1)  + yd(k)*dt;

        eprev = e_k;
    end
```

```
        e = r - y;
    end

    J = trapz(t, abs(e));
    if ~isfinite(J), J=1e12; end
catch
    J = 1e12;
end
end

% -------------------- GA --------------------
function [bestX,bestCost,histCost]=ga_simple(lb,ub,dim,npop,maxIt,fitness)
pc = 0.85; pm = 0.25; sigma = 0.10*(ub-lb);

pop  = rand(npop,dim).*(ub-lb)+lb;
cost = zeros(npop,1);
for i=1:npop, cost(i)=fitness(pop(i,:)); end

[bestCost,idx]=min(cost); bestX = pop(idx,:);
histCost=zeros(maxIt,1);

for it=1:maxIt
    newPop = zeros(npop,dim);
    for k=1:2:npop
        p1 = tournament(pop,cost);
```

```
    p2 = tournament(pop,cost);

    c1=p1; c2=p2;
    if rand<pc
        alpha = rand(1,dim);
        c1 = alpha.*p1 + (1-alpha).*p2;
        c2 = alpha.*p2 + (1-alpha).*p1;
    end

    if rand<pm, c1 = c1 + sigma.*randn(1,dim); end
    if rand<pm, c2 = c2 + sigma.*randn(1,dim); end

    c1 = max(min(c1,ub),lb);
    c2 = max(min(c2,ub),lb);

    newPop(k,:) = c1;
    if k+1<=npop, newPop(k+1,:) = c2; end
end

pop = newPop;
for i=1:npop, cost(i)=fitness(pop(i,:)); end

[cmin,idx]=min(cost);
if cmin<bestCost, bestCost=cmin; bestX=pop(idx,:); end
histCost(it)=bestCost;
```

```
end
end

function p = tournament(pop,cost)
n=size(pop,1);
i1=randi(n); i2=randi(n);
if cost(i1)<cost(i2), p=pop(i1,:); else, p=pop(i2,:); end
end
```

Ek C: İnsansız Kara Aracının (İKA) 2B Optimum Yol Planlaması İçin Beş Meta-Sezgisel Algoritmaya Ait Örnek Kodlar (10 Ziyaret Noktası– 10 Engel)

Ek C.1. Gri Kurt Optimizasyon Algoritması (GWO) ile İKA 2B

```
%
==========================================================
% Ek C.1 – GWO ile İKA 2B Yol Planlama (10 Ziyaret Noktası– 10 Engel)
% Amaç fonksiyonu: Yol uzunluğu + pürüzsüzlük + çarpışma cezası
% Engel modeli: Dikdörtgen engeller (x,y,w,h)
%
==========================================================
clear; clc; close all

%% Harita / Senaryo (10 engel)
map.W = 100;
map.H = 100;
map.obstacles = build_urban_map_10();   % [x y w h]  (10 adet)
```

```
start  = [5  5];
goal   = [95 95];

Nwp    = 10;      % ziyaret sayısı
safety = 2.0;     % emniyet mesafesi (engel şişirme)

%% Karar değişkeni ve sınırlar
% x = [x_wp1..x_wpN, y_wp1..y_wpN]  -> dim = 2*Nwp
dim = 2*Nwp;
lb  = [zeros(1,Nwp) zeros(1,Nwp)];
ub  = [map.W*ones(1,Nwp) map.H*ones(1,Nwp)];

%% Amaç fonksiyonu
fit = @(x) f_path_cost_2d(x, start, goal, map, safety, Nwp);

%% GWO parametreleri
nWolves = 40;
maxIt   = 150;

%% GWO çalıştır
[bestX, bestCost, histCost] = gwo(lb, ub, dim, nWolves, maxIt, fit);

%% En iyi yolun çıkarılması
wps = decode_wps_2d(bestX, Nwp);
[pathXY, segLen] = path_with_endpoints_2d(wps, start, goal);
```

```
%% Görselleştirme: Harita + yol
figure; hold on; axis equal; box on; grid on
plot_urban_map_2d(map, safety);
plot(pathXY(:,1), pathXY(:,2), 'LineWidth', 1.8);
scatter([start(1) goal(1)], [start(2) goal(2)], 70, 'filled');
title(sprintf('GWO – İKA 2B Yol Planlama | Maliyet: %.3f | Uzunluk: %.2f', bestCost, sum(segLen)));
xlabel('x'); ylabel('y');

%% Yakınsama grafiği
figure; grid on; box on
plot(histCost, 'LineWidth', 1.6);
xlabel('İterasyon'); ylabel('En iyi maliyet');
title('GWO – Yakınsama (10 Ziyaret noktası – 10 Engel)');

% ====================== YARDIMCI FONKSİYONLAR ======================

function obs = build_urban_map_10()
% 10 adet dikdörtgen engel: [x y w h]
obs = [
   12  10  14  18
   30  18  16  12
   52  12  10  22
   70  14  18  14
```

```
    18  42  20  12
    44  38  14  18
    66  40  22  10
    10  68  16  18
    36  70  18  14
    68  72  20  16
];
end

function plot_urban_map_2d(map, safety)
% Engelleri emniyet mesafesi ile şişirip çizer
obs = map.obstacles;
for i = 1:size(obs,1)
    r = inflate_rect(obs(i,:), safety);
    rectangle('Position', [r(1) r(2) r(3) r(4)], 'LineWidth', 1.2);
end
xlim([0 map.W]); ylim([0 map.H]);
end

function r2 = inflate_rect(r, s)
% r=[x y w h] -> şişirilmiş dikdörtgen
r2 = [r(1)-s, r(2)-s, r(3)+2*s, r(4)+2*s];
end

function wps = decode_wps_2d(x, Nwp)
```

```
xw = x(1:Nwp);
yw = x(Nwp+1:2*Nwp);
wps = [xw(:) yw(:)];
end

function [pathXY, segLen] = path_with_endpoints_2d(wps, start, goal)
pathXY = [start; wps; goal];
d = diff(pathXY, 1, 1);
segLen = sqrt(sum(d.^2, 2));
end

function J = f_path_cost_2d(x, start, goal, map, safety, Nwp)
% Amaç: uzunluk + pürüzsüzlük + çarpışma cezası + düzenleyici terimler
if any(~isfinite(x)), J = 1e12; return; end

wps  = decode_wps_2d(x, Nwp);
path = [start; wps; goal];

% 1) Yol uzunluğu
d = diff(path, 1, 1);
L = sum(sqrt(sum(d.^2,2)));

% 2) Pürüzsüzlük (yön değişimi / açı kareleri)
S = 0;
for i = 2:size(path,1)-1
```

```
    a = path(i,:)   - path(i-1,:);
    b = path(i+1,:) - path(i,:);
    na = norm(a); nb = norm(b);
    if na < 1e-9 || nb < 1e-9, continue; end
    cosang = max(-1, min(1, dot(a,b)/(na*nb)));
    ang = acos(cosang);
    S = S + ang^2;
end

% 3) Çarpışma cezası (segment-rect kesişimi + içerden geçiş)
C = collision_penalty_2d(path, map.obstacles, safety);

% 4) Düzenleyici: ziyaret noktasılerin çok yakın olmasını engelle
R = ziyaret noktası_spacing_penalty_2d(wps, 2.0);

% Ağırlıklar
wL = 1.0;
wS = 10.0;
wC = 200.0;
wR = 0.8;

J = wL*L + wS*S + wC*C + wR*R;

if ~isfinite(J) || J > 1e15, J = 1e15; end
end
```

```
function C = collision_penalty_2d(path, obstacles, safety)
% Segmentlerin şişirilmiş dikdörtgenlere temasını/ihlalini örnekleme ile cezalandırır
C = 0;
nSeg = size(path,1)-1;

for k = 1:nSeg
   p1 = path(k,:);
   p2 = path(k+1,:);
   for oi = 1:size(obstacles,1)
      rect = inflate_rect(obstacles(oi,:), safety);
      % Segment boyunca örnekleme
      C = C + segment_rect_sample_penalty(p1, p2, rect);
   end
end
end

function pen = segment_rect_sample_penalty(p1, p2, rect)
% rect=[x y w h] (axis-aligned). Segment üzerinde noktalar örneklenir.
% İçeride kalan örnek sayısına göre ceza artar.
x0=rect(1); y0=rect(2); w=rect(3); h=rect(4);
xmin=x0; xmax=x0+w; ymin=y0; ymax=y0+h;

% örnek sayısı (segment uzunluğuna göre adaptif)
```

```
segLen = norm(p2-p1);
ns = max(15, min(80, ceil(segLen*1.2)));

t = linspace(0,1,ns);
pts = p1 + (p2-p1).*t(:);

inside = (pts(:,1)>=xmin & pts(:,1)<=xmax & pts(:,2)>=ymin & pts(:,2)<=ymax);

% İçeride kalma oranı + "dokunma" için küçük pay
ratio = sum(inside)/ns;

% Eğer hiç içeri girmiyorsa da, çok yakın geçişleri zayıf cezalandırmak için:
distEdge = minDistancePointToRect(p1, rect);
distEdge = min(distEdge, minDistancePointToRect(p2, rect));

pen = 0;
if ratio > 0
   pen = pen + 1 + 10*ratio;   % ihlal varsa güçlü ceza
elseif distEdge < 0.25
   pen = pen + 0.2;            % sınır temasına yakınsa küçük ceza
end
end

function d = minDistancePointToRect(p, rect)
```

```
% Noktanın dikdörtgene olan en kısa mesafesi (dışarıdaysa)
x0=rect(1); y0=rect(2); w=rect(3); h=rect(4);
xmin=x0; xmax=x0+w; ymin=y0; ymax=y0+h;

dx = max([xmin - p(1), 0, p(1) - xmax]);
dy = max([ymin - p(2), 0, p(2) - ymax]);
d = hypot(dx, dy);
end

function R = ziyaret noktası_spacing_penalty_2d(wps, dmin)
% Ziyaret noktasıler arası minimum mesafe koşulu (çok yakınsa ceza)
R = 0;
for i=1:size(wps,1)-1
   d = norm(wps(i+1,:) - wps(i,:));
   if d < dmin
      R = R + (dmin - d)^2;
   end
end
end

% ====================== GWO ======================
function [Alpha_pos, Alpha_score, histCost] = gwo(lb, ub, dim, nWolves, maxIt, fobj)

Alpha_pos = zeros(1,dim); Alpha_score = inf;
```

```
Beta_pos  = zeros(1,dim); Beta_score  = inf;
Delta_pos = zeros(1,dim); Delta_score = inf;

X = rand(nWolves,dim).*(ub-lb) + lb;
histCost = zeros(maxIt,1);

for it = 1:maxIt

   for i = 1:nWolves
      X(i,:) = max(min(X(i,:),ub),lb);
      score  = fobj(X(i,:));

      if score < Alpha_score
         Delta_score = Beta_score;  Delta_pos = Beta_pos;
         Beta_score  = Alpha_score; Beta_pos  = Alpha_pos;
         Alpha_score = score;       Alpha_pos = X(i,:);
      elseif score < Beta_score
         Delta_score = Beta_score;  Delta_pos = Beta_pos;
         Beta_score  = score;       Beta_pos  = X(i,:);
      elseif score < Delta_score
         Delta_score = score;       Delta_pos = X(i,:);
      end
   end

   a = 2 - it*(2/maxIt);
```

```
    for i = 1:nWolves
        for j = 1:dim
            r1=rand; r2=rand;
            A1=2*a*r1-a; C1=2*r2;
            D_alpha=abs(C1*Alpha_pos(j)-X(i,j));
            X1=Alpha_pos(j)-A1*D_alpha;

            r1=rand; r2=rand;
            A2=2*a*r1-a; C2=2*r2;
            D_beta=abs(C2*Beta_pos(j)-X(i,j));
            X2=Beta_pos(j)-A2*D_beta;

            r1=rand; r2=rand;
            A3=2*a*r1-a; C3=2*r2;
            D_delta=abs(C3*Delta_pos(j)-X(i,j));
            X3=Delta_pos(j)-A3*D_delta;

            X(i,j)=(X1+X2+X3)/3;
        end
    end

    histCost(it) = Alpha_score;
end
end
```

Ek C.2. Parçacık Sürü Optimizasyonu (PSO) ile İKA Optimum Yol Planlaması MATLAB Kodu

```
%
==========================================================
% Ek C.2 – PSO ile İKA 2B Yol Planlama (10 Ziyaret noktası – 10 Engel)
%
==========================================================
clear; clc; close all

map.W = 100; map.H = 100;
map.obstacles = build_urban_map_10();

start  = [5  5];
goal   = [95 95];

Nwp    = 10;
safety = 2.0;

dim = 2*Nwp;
lb  = [zeros(1,Nwp) zeros(1,Nwp)];
ub  = [map.W*ones(1,Nwp) map.H*ones(1,Nwp)];

fit = @(x) f_path_cost_2d(x, start, goal, map, safety, Nwp);

% PSO ayarları
nPart = 50;
maxIt = 150;

[bestX, bestCost, histCost] = pso(lb, ub, dim, nPart, maxIt, fit);

wps = decode_wps_2d(bestX, Nwp);
[pathXY, segLen] = path_with_endpoints_2d(wps, start, goal);

figure; hold on; axis equal; box on; grid on
```

```
plot_urban_map_2d(map, safety);
plot(pathXY(:,1), pathXY(:,2), 'LineWidth', 1.8);
scatter([start(1) goal(1)], [start(2) goal(2)], 70, 'filled');
title(sprintf('PSO – İKA 2B Yol Planlama | Maliyet: %.3f | Uzunluk: %.2f', bestCost, sum(segLen)));
xlabel('x'); ylabel('y');

figure; grid on; box on
plot(histCost, 'LineWidth', 1.6);
xlabel('İterasyon'); ylabel('En iyi maliyet');
title('PSO – Yakınsama (10 Ziyaret noktası – 10 Engel)');

% ====================== ORTAK FONKSİYONLAR ======================
% (C.1 ile aynı: build_urban_map_10, plot_urban_map_2d, inflate_rect,
%  decode_wps_2d, path_with_endpoints_2d, f_path_cost_2d, vb.)
% Bu ekin bütünlüğü için fonksiyonlar aşağıda tekrar verilmiştir.

function obs = build_urban_map_10()
obs = [
   12  10  14  18
   30  18  16  12
   52  12  10  22
   70  14  18  14
   18  42  20  12
   44  38  14  18
   66  40  22  10
   10  68  16  18
   36  70  18  14
   68  72  20  16
];
end

function plot_urban_map_2d(map, safety)
```

```
obs = map.obstacles;
for i = 1:size(obs,1)
   r = inflate_rect(obs(i,:), safety);
   rectangle('Position', [r(1) r(2) r(3) r(4)], 'LineWidth', 1.2);
end
xlim([0 map.W]); ylim([0 map.H]);
end

function r2 = inflate_rect(r, s)
r2 = [r(1)-s, r(2)-s, r(3)+2*s, r(4)+2*s];
end

function wps = decode_wps_2d(x, Nwp)
xw = x(1:Nwp);
yw = x(Nwp+1:2*Nwp);
wps = [xw(:) yw(:)];
end

function [pathXY, segLen] = path_with_endpoints_2d(wps, start, goal)
pathXY = [start; wps; goal];
d = diff(pathXY, 1, 1);
segLen = sqrt(sum(d.^2, 2));
end

function J = f_path_cost_2d(x, start, goal, map, safety, Nwp)
if any(~isfinite(x)), J = 1e12; return; end
wps  = decode_wps_2d(x, Nwp);
path = [start; wps; goal];

d = diff(path, 1, 1);
L = sum(sqrt(sum(d.^2,2)));

S = 0;
for i = 2:size(path,1)-1
```

```
    a = path(i,:)   - path(i-1,:);
    b = path(i+1,:) - path(i,:);
    na = norm(a); nb = norm(b);
    if na < 1e-9 || nb < 1e-9, continue; end
    cosang = max(-1, min(1, dot(a,b)/(na*nb)));
    ang = acos(cosang);
    S = S + ang^2;
end

C = collision_penalty_2d(path, map.obstacles, safety);
R = ziyaret noktası_spacing_penalty_2d(wps, 2.0);

wL=1.0; wS=10.0; wC=200.0; wR=0.8;
J = wL*L + wS*S + wC*C + wR*R;

if ~isfinite(J) || J > 1e15, J = 1e15; end
end

function C = collision_penalty_2d(path, obstacles, safety)
C = 0;
nSeg = size(path,1)-1;
for k = 1:nSeg
    p1 = path(k,:); p2 = path(k+1,:);
    for oi = 1:size(obstacles,1)
        rect = inflate_rect(obstacles(oi,:), safety);
        C = C + segment_rect_sample_penalty(p1, p2, rect);
    end
end
end

function pen = segment_rect_sample_penalty(p1, p2, rect)
x0=rect(1); y0=rect(2); w=rect(3); h=rect(4);
xmin=x0; xmax=x0+w; ymin=y0; ymax=y0+h;
```

```
segLen = norm(p2-p1);
ns = max(15, min(80, ceil(segLen*1.2)));

t = linspace(0,1,ns);
pts = p1 + (p2-p1).*t(:);

inside = (pts(:,1)>=xmin & pts(:,1)<=xmax & pts(:,2)>=ymin & pts(:,2)<=ymax);
ratio = sum(inside)/ns;

distEdge = minDistancePointToRect(p1, rect);
distEdge = min(distEdge, minDistancePointToRect(p2, rect));

pen = 0;
if ratio > 0
   pen = pen + 1 + 10*ratio;
elseif distEdge < 0.25
   pen = pen + 0.2;
end
end

function d = minDistancePointToRect(p, rect)
x0=rect(1); y0=rect(2); w=rect(3); h=rect(4);
xmin=x0; xmax=x0+w; ymin=y0; ymax=y0+h;

dx = max([xmin - p(1), 0, p(1) - xmax]);
dy = max([ymin - p(2), 0, p(2) - ymax]);
d = hypot(dx, dy);
end

function R = ziyaret noktası_spacing_penalty_2d(wps, dmin)
R = 0;
for i=1:size(wps,1)-1
   d = norm(wps(i+1,:) - wps(i,:));
```

```
    if d < dmin
        R = R + (dmin - d)^2;
    end
end
end

% ======================= PSO ======================
function [bestX, bestCost, histCost] = pso(lb, ub, dim, nPart, maxIt, fobj)
w  = 0.72;
c1 = 1.49;
c2 = 1.49;

X = rand(nPart,dim).*(ub-lb) + lb;
V = zeros(nPart,dim);

Pbest = X;
Pcost = inf(nPart,1);
for i=1:nPart
    Pcost(i) = fobj(X(i,:));
end
[bestCost, gi] = min(Pcost);
bestX = Pbest(gi,:);

histCost = zeros(maxIt,1);

for it=1:maxIt
    for i=1:nPart
        r1 = rand(1,dim);
        r2 = rand(1,dim);

        V(i,:) = w*V(i,:) + c1*r1.*(Pbest(i,:)-X(i,:)) + c2*r2.*(bestX - X(i,:));
        X(i,:) = X(i,:) + V(i,:);

        X(i,:) = max(min(X(i,:),ub),lb);
```

```
            c = fobj(X(i,:));
            if c < Pcost(i)
               Pbest(i,:) = X(i,:);
               Pcost(i)   = c;
            end

            if Pcost(i) < bestCost
               bestCost = Pcost(i);
               bestX    = Pbest(i,:);
            end
          end
          histCost(it) = bestCost;
        end
        end
```

Ek C.3. Yapay Arı Kolonisi (ABC) Algoritması ile İKA 2B Optimum Yol Planlaması (10 Ziyaret noktası – 10 Engel)

```
        %
==========================================================
        % Ek C.3 – ABC ile İKA 2B Yol Planlama (10 Ziyaret noktası – 10 Engel)
        % Amaç fonksiyonu: Yol uzunluğu + pürüzsüzlük + çarpışma cezası
        % Engel modeli: Dikdörtgen engeller (x,y,w,h)
        %
==========================================================
        clear; clc; close all

        %% Harita / Senaryo (10 engel)
        map.W = 100;
        map.H = 100;
        map.obstacles = build_urban_map_10();   % [x y w h]  (10 adet)

        start  = [5  5];
        goal   = [95 95];
```

```
Nwp    = 10;     % ziyaret noktası sayısı
safety = 2.0;    % emniyet mesafesi

%% Karar değişkeni ve sınırlar
dim = 2*Nwp;   % [x_wp1..x_wpN, y_wp1..y_wpN]
lb  = [zeros(1,Nwp) zeros(1,Nwp)];
ub  = [map.W*ones(1,Nwp) map.H*ones(1,Nwp)];

%% Amaç fonksiyonu
fit = @(x) f_path_cost_2d(x, start, goal, map, safety, Nwp);

%% ABC parametreleri
FoodNumber = 30;    % besin kaynağı sayısı (koloni)
maxIt      = 200;   % iterasyon

%% ABC çalıştır
[bestX, bestCost, histCost] = abc(lb, ub, dim, FoodNumber, maxIt, fit);

%% En iyi yolun çıkarılması
wps = decode_wps_2d(bestX, Nwp);
[pathXY, segLen] = path_with_endpoints_2d(wps, start, goal);

%% Görselleştirme: Harita + yol
figure; hold on; axis equal; box on; grid on
plot_urban_map_2d(map, safety);
plot(pathXY(:,1), pathXY(:,2), 'LineWidth', 1.8);
scatter([start(1) goal(1)], [start(2) goal(2)], 70, 'filled');
title(sprintf('ABC – İKA 2B Yol Planlama | Maliyet: %.3f | Uzunluk: %.2f',
bestCost, sum(segLen)));
xlabel('x'); ylabel('y');

%% Yakınsama grafiği
figure; grid on; box on
plot(histCost, 'LineWidth', 1.6);
```

```
xlabel('İterasyon'); ylabel('En iyi maliyet');
title('ABC – Yakınsama (10 Ziyaret noktası – 10 Engel)');

% ====================== ORTAK FONKSİYONLAR ======================

function obs = build_urban_map_10()
% 10 adet dikdörtgen engel: [x y w h]
obs = [
   12  10  14  18
   30  18  16  12
   52  12  10  22
   70  14  18  14
   18  42  20  12
   44  38  14  18
   66  40  22  10
   10  68  16  18
   36  70  18  14
   68  72  20  16
];
end

function plot_urban_map_2d(map, safety)
obs = map.obstacles;
for i = 1:size(obs,1)
   r = inflate_rect(obs(i,:), safety);
   rectangle('Position', [r(1) r(2) r(3) r(4)], 'LineWidth', 1.2);
end
xlim([0 map.W]); ylim([0 map.H]);
end

function r2 = inflate_rect(r, s)
r2 = [r(1)-s, r(2)-s, r(3)+2*s, r(4)+2*s];
end
```

```
function wps = decode_wps_2d(x, Nwp)
xw = x(1:Nwp);
yw = x(Nwp+1:2*Nwp);
wps = [xw(:) yw(:)];
end

function [pathXY, segLen] = path_with_endpoints_2d(wps, start, goal)
pathXY = [start; wps; goal];
d = diff(pathXY, 1, 1);
segLen = sqrt(sum(d.^2, 2));
end

function J = f_path_cost_2d(x, start, goal, map, safety, Nwp)
if any(~isfinite(x)), J = 1e12; return; end
wps  = decode_wps_2d(x, Nwp);
path = [start; wps; goal];

% 1) yol uzunluğu
d = diff(path, 1, 1);
L = sum(sqrt(sum(d.^2,2)));

% 2) pürüzsüzlük
S = 0;
for i = 2:size(path,1)-1
   a = path(i,:)   - path(i-1,:);
   b = path(i+1,:) - path(i,:);
   na = norm(a); nb = norm(b);
   if na < 1e-9 || nb < 1e-9, continue; end
   cosang = max(-1, min(1, dot(a,b)/(na*nb)));
   ang = acos(cosang);
   S = S + ang^2;
end
```

```
% 3) çarpışma cezası
C = collision_penalty_2d(path, map.obstacles, safety);

% 4) ziyaret noktası düzenleyici
R = ziyaret noktası_spacing_penalty_2d(wps, 2.0);

% ağırlıklar
wL=1.0; wS=10.0; wC=200.0; wR=0.8;
J = wL*L + wS*S + wC*C + wR*R;

if ~isfinite(J) || J > 1e15, J = 1e15; end
end

function C = collision_penalty_2d(path, obstacles, safety)
C = 0;
nSeg = size(path,1)-1;
for k = 1:nSeg
   p1 = path(k,:); p2 = path(k+1,:);
   for oi = 1:size(obstacles,1)
      rect = inflate_rect(obstacles(oi,:), safety);
      C = C + segment_rect_sample_penalty(p1, p2, rect);
   end
end
end

function pen = segment_rect_sample_penalty(p1, p2, rect)
x0=rect(1); y0=rect(2); w=rect(3); h=rect(4);
xmin=x0; xmax=x0+w; ymin=y0; ymax=y0+h;

segLen = norm(p2-p1);
ns = max(15, min(80, ceil(segLen*1.2)));

t = linspace(0,1,ns);
pts = p1 + (p2-p1).*t(:);
```

```
    inside = (pts(:,1)>=xmin & pts(:,1)<=xmax & pts(:,2)>=ymin & pts(:,2)<=ymax);
    ratio = sum(inside)/ns;

    distEdge = minDistancePointToRect(p1, rect);
    distEdge = min(distEdge, minDistancePointToRect(p2, rect));

    pen = 0;
    if ratio > 0
       pen = pen + 1 + 10*ratio;
    elseif distEdge < 0.25
       pen = pen + 0.2;
    end
    end

    function d = minDistancePointToRect(p, rect)
    x0=rect(1); y0=rect(2); w=rect(3); h=rect(4);
    xmin=x0; xmax=x0+w; ymin=y0; ymax=y0+h;

    dx = max([xmin - p(1), 0, p(1) - xmax]);
    dy = max([ymin - p(2), 0, p(2) - ymax]);
    d = hypot(dx, dy);
    end

    function R = ziyaret noktası_spacing_penalty_2d(wps, dmin)
    R = 0;
    for i=1:size(wps,1)-1
       d = norm(wps(i+1,:) - wps(i,:));
       if d < dmin
          R = R + (dmin - d)^2;
       end
    end
    end
```

```
% ====================== ABC ======================
function [bestX, bestCost, histCost] = abc(lb, ub, dim, SN, maxIt, fobj)
limit = 5*dim;  % scout limiti

Foods = rand(SN,dim).*(ub-lb) + lb;
FCost = zeros(SN,1);
trial = zeros(SN,1);

for i=1:SN
   FCost(i) = fobj(Foods(i,:));
end

[bestCost, bi] = min(FCost);
bestX = Foods(bi,:);
histCost = zeros(maxIt,1);

for it=1:maxIt

   % 1) Employed bees
   for i=1:SN
      k=i; while k==i, k=randi(SN); end
      phi = -1 + 2*rand(1,dim);

      v = Foods(i,:) + phi.*(Foods(i,:) - Foods(k,:));
      v = max(min(v,ub),lb);

      Cv = fobj(v);
      if Cv <= FCost(i)
         Foods(i,:) = v;
         FCost(i)   = Cv;
         trial(i)   = 0;
      else
         trial(i) = trial(i) + 1;
```

```
    end
end

% 2) Onlooker bees
fit = 1./(1 + FCost);
prob = fit / sum(fit);

t = 0; i = 1;
while t < SN
    if rand < prob(i)
        t = t + 1;

        k=i; while k==i, k=randi(SN); end
        phi = -1 + 2*rand(1,dim);

        v = Foods(i,:) + phi.*(Foods(i,:) - Foods(k,:));
        v = max(min(v,ub),lb);

        Cv = fobj(v);
        if Cv <= FCost(i)
            Foods(i,:) = v;
            FCost(i)   = Cv;
            trial(i)   = 0;
        else
            trial(i) = trial(i) + 1;
        end
    end

    i = i + 1;
    if i > SN, i = 1; end
end

% 3) Scout bees
for i=1:SN
```

```
            if trial(i) > limit
                Foods(i,:) = rand(1,dim).*(ub-lb) + lb;
                FCost(i)   = fobj(Foods(i,:));
                trial(i)   = 0;
            end
        end

        % 4) Best update
        [cmin, bi] = min(FCost);
        if cmin < bestCost
            bestCost = cmin;
            bestX    = Foods(bi,:);
        end
        histCost(it) = bestCost;
    end
    end
```

Ek C.4. Karınca Kolonisi Optimizasyonu (ACO/ACOR) ile İKA 2B Optimum Yol Planlaması (10 Ziyaret noktası – 10 Engel)

```
    %
==========================================================
    % Ek C.4 – ACOR (Sürekli ACO) ile İKA 2B Yol Planlama
    % (10 Ziyaret noktası – 10 Engel)
    %
==========================================================
    clear; clc; close all

    map.W = 100;
    map.H = 100;
    map.obstacles = build_urban_map_10();

    start  = [5  5];
    goal   = [95 95];

    Nwp    = 10;
```

```
safety = 2.0;

dim = 2*Nwp;
lb  = [zeros(1,Nwp) zeros(1,Nwp)];
ub  = [map.W*ones(1,Nwp) map.H*ones(1,Nwp)];

fit = @(x) f_path_cost_2d(x, start, goal, map, safety, Nwp);

% ACOR ayarları
Nnew  = 25;   % her iterasyonda üretilecek yeni örnek
maxIt = 200;

[bestX, bestCost, histCost] = acor(lb, ub, dim, Nnew, maxIt, fit);

wps = decode_wps_2d(bestX, Nwp);
[pathXY, segLen] = path_with_endpoints_2d(wps, start, goal);

figure; hold on; axis equal; box on; grid on
plot_urban_map_2d(map, safety);
plot(pathXY(:,1), pathXY(:,2), 'LineWidth', 1.8);
scatter([start(1) goal(1)], [start(2) goal(2)], 70, 'filled');
title(sprintf('ACOR – İKA 2B Yol Planlama | Maliyet: %.3f | Uzunluk: %.2f',
bestCost, sum(segLen)));
xlabel('x'); ylabel('y');

figure; grid on; box on
plot(histCost, 'LineWidth', 1.6);
xlabel('İterasyon'); ylabel('En iyi maliyet');
title('ACOR – Yakınsama (10 Ziyaret noktası – 10 Engel)');

% ====================== ORTAK FONKSİYONLAR ======================

function obs = build_urban_map_10()
obs = [
```

```
    12  10  14  18
    30  18  16  12
    52  12  10  22
    70  14  18  14
    18  42  20  12
    44  38  14  18
    66  40  22  10
    10  68  16  18
    36  70  18  14
    68  72  20  16
];
end

function plot_urban_map_2d(map, safety)
obs = map.obstacles;
for i = 1:size(obs,1)
    r = inflate_rect(obs(i,:), safety);
    rectangle('Position', [r(1) r(2) r(3) r(4)], 'LineWidth', 1.2);
end
xlim([0 map.W]); ylim([0 map.H]);
end

function r2 = inflate_rect(r, s)
r2 = [r(1)-s, r(2)-s, r(3)+2*s, r(4)+2*s];
end

function wps = decode_wps_2d(x, Nwp)
xw = x(1:Nwp);
yw = x(Nwp+1:2*Nwp);
wps = [xw(:) yw(:)];
end

function [pathXY, segLen] = path_with_endpoints_2d(wps, start, goal)
pathXY = [start; wps; goal];
```

```
d = diff(pathXY, 1, 1);
segLen = sqrt(sum(d.^2, 2));
end

function J = f_path_cost_2d(x, start, goal, map, safety, Nwp)
if any(~isfinite(x)), J = 1e12; return; end
wps  = decode_wps_2d(x, Nwp);
path = [start; wps; goal];

d = diff(path, 1, 1);
L = sum(sqrt(sum(d.^2,2)));

S = 0;
for i = 2:size(path,1)-1
   a = path(i,:)   - path(i-1,:);
   b = path(i+1,:) - path(i,:);
   na = norm(a); nb = norm(b);
   if na < 1e-9 || nb < 1e-9, continue; end
   cosang = max(-1, min(1, dot(a,b)/(na*nb)));
   ang = acos(cosang);
   S = S + ang^2;
end

C = collision_penalty_2d(path, map.obstacles, safety);
R = ziyaret noktası_spacing_penalty_2d(wps, 2.0);

wL=1.0; wS=10.0; wC=200.0; wR=0.8;
J = wL*L + wS*S + wC*C + wR*R;

if ~isfinite(J) || J > 1e15, J = 1e15; end
end

function C = collision_penalty_2d(path, obstacles, safety)
C = 0;
```

```
nSeg = size(path,1)-1;
for k = 1:nSeg
   p1 = path(k,:); p2 = path(k+1,:);
   for oi = 1:size(obstacles,1)
      rect = inflate_rect(obstacles(oi,:), safety);
      C = C + segment_rect_sample_penalty(p1, p2, rect);
   end
end
end

function pen = segment_rect_sample_penalty(p1, p2, rect)
x0=rect(1); y0=rect(2); w=rect(3); h=rect(4);
xmin=x0; xmax=x0+w; ymin=y0; ymax=y0+h;

segLen = norm(p2-p1);
ns = max(15, min(80, ceil(segLen*1.2)));

t = linspace(0,1,ns);
pts = p1 + (p2-p1).*t(:);

inside = (pts(:,1)>=xmin & pts(:,1)<=xmax & pts(:,2)>=ymin & pts(:,2)<=ymax);
ratio = sum(inside)/ns;

distEdge = minDistancePointToRect(p1, rect);
distEdge = min(distEdge, minDistancePointToRect(p2, rect));

pen = 0;
if ratio > 0
   pen = pen + 1 + 10*ratio;
elseif distEdge < 0.25
   pen = pen + 0.2;
end
end
```

```
function d = minDistancePointToRect(p, rect)
x0=rect(1); y0=rect(2); w=rect(3); h=rect(4);
xmin=x0; xmax=x0+w; ymin=y0; ymax=y0+h;

dx = max([xmin - p(1), 0, p(1) - xmax]);
dy = max([ymin - p(2), 0, p(2) - ymax]);
d = hypot(dx, dy);
end

function R = ziyaret noktası_spacing_penalty_2d(wps, dmin)
R = 0;
for i=1:size(wps,1)-1
   d = norm(wps(i+1,:) - wps(i,:));
   if d < dmin
      R = R + (dmin - d)^2;
   end
end
end

% ======================= ACOR =======================
function [bestX, bestCost, histCost] = acor(lb, ub, dim, Nnew, maxIt, fobj)
% ACOR parametreleri
q  = 0.5;      % ağırlık yayılımı
xi = 0.85;     % sigma ölçekleme
M  = 2*Nnew;   % arşiv boyutu

% Başlangıç arşivi
A = rand(M,dim).*(ub-lb) + lb;
ACost = zeros(M,1);
for i=1:M
   ACost(i) = fobj(A(i,:));
end
```

```
% Sırala
[ACost, idx] = sort(ACost);
A = A(idx,:);

% Rank tabanlı ağırlıklar
ranks = (1:M)';
w = (1/(q*M*sqrt(2*pi))) * exp(-(ranks-1).^2/(2*(q*M)^2));
w = w / sum(w);

histCost = zeros(maxIt,1);

for it = 1:maxIt

   % Yeni örnekler
   S = zeros(Nnew, dim);
   for n=1:Nnew
      for d=1:dim
         mu_d = A(:,d);
         sigma_d = xi * abs(mu_d - mu_d(1));
         if all(sigma_d == 0)
            sigma_d(:) = (ub(d)-lb(d))/50;
         end

         k = randsample(M, 1, true, w);
         S(n,d) = mu_d(k) + sigma_d(k)*randn;
      end
      S(n,:) = max(min(S(n,:),ub),lb);
   end

   % Yeni örnek maliyetleri
   SCost = zeros(Nnew,1);
   for n=1:Nnew
      SCost(n) = fobj(S(n,:));
   end
```

```
    % Arşivi güncelle
    A = [A; S];
    ACost = [ACost; SCost];

    [ACost, idx] = sort(ACost);
    A = A(idx,:);

    A = A(1:M,:);
    ACost = ACost(1:M);

    histCost(it) = ACost(1);
  end

  bestX = A(1,:);
  bestCost = ACost(1);
  end
```

Ek C.5. Genetik Algoritma (GA) ile İKA 2B Optimum Yol Planlaması (10 Ziyaret Noktası – 10 Engel)

```
  %
==========================================================
  % Ek C.5 – GA ile İKA 2B Yol Planlama (10 Ziyaret noktası – 10 Engel)
  %
==========================================================
  clear; clc; close all

  map.W = 100;
  map.H = 100;
  map.obstacles = build_urban_map_10();

  start  = [5  5];
  goal   = [95 95];

  Nwp    = 10;
```

```
safety = 2.0;

dim = 2*Nwp;
lb  = [zeros(1,Nwp) zeros(1,Nwp)];
ub  = [map.W*ones(1,Nwp) map.H*ones(1,Nwp)];

fit = @(x) f_path_cost_2d(x, start, goal, map, safety, Nwp);

% GA ayarları
npop  = 60;
maxIt = 160;

[bestX, bestCost, histCost] = ga_simple(lb, ub, dim, npop, maxIt, fit);

wps = decode_wps_2d(bestX, Nwp);
[pathXY, segLen] = path_with_endpoints_2d(wps, start, goal);

figure; hold on; axis equal; box on; grid on
plot_urban_map_2d(map, safety);
plot(pathXY(:,1), pathXY(:,2), 'LineWidth', 1.8);
scatter([start(1) goal(1)], [start(2) goal(2)], 70, 'filled');
title(sprintf('GA – İKA 2B Yol Planlama | Maliyet: %.3f | Uzunluk: %.2f', bestCost, sum(segLen)));
xlabel('x'); ylabel('y');

figure; grid on; box on
plot(histCost, 'LineWidth', 1.6);
xlabel('İterasyon'); ylabel('En iyi maliyet');
title('GA – Yakınsama (10 Ziyaret noktası – 10 Engel)');

% ====================== ORTAK FONKSİYONLAR ======================
function obs = build_urban_map_10()
obs = [
```

```
    12  10  14  18
    30  18  16  12
    52  12  10  22
    70  14  18  14
    18  42  20  12
    44  38  14  18
    66  40  22  10
    10  68  16  18
    36  70  18  14
    68  72  20  16
];
end

function plot_urban_map_2d(map, safety)
obs = map.obstacles;
for i = 1:size(obs,1)
    r = inflate_rect(obs(i,:), safety);
    rectangle('Position', [r(1) r(2) r(3) r(4)], 'LineWidth', 1.2);
end
xlim([0 map.W]); ylim([0 map.H]);
end

function r2 = inflate_rect(r, s)
r2 = [r(1)-s, r(2)-s, r(3)+2*s, r(4)+2*s];
end

function wps = decode_wps_2d(x, Nwp)
xw = x(1:Nwp);
yw = x(Nwp+1:2*Nwp);
wps = [xw(:) yw(:)];
end

function [pathXY, segLen] = path_with_endpoints_2d(wps, start, goal)
pathXY = [start; wps; goal];
```

```
d = diff(pathXY, 1, 1);
segLen = sqrt(sum(d.^2, 2));
end

function J = f_path_cost_2d(x, start, goal, map, safety, Nwp)
if any(~isfinite(x)), J = 1e12; return; end
wps  = decode_wps_2d(x, Nwp);
path = [start; wps; goal];

d = diff(path, 1, 1);
L = sum(sqrt(sum(d.^2,2)));

S = 0;
for i = 2:size(path,1)-1
   a = path(i,:)   - path(i-1,:);
   b = path(i+1,:) - path(i,:);
   na = norm(a); nb = norm(b);
   if na < 1e-9 || nb < 1e-9, continue; end
   cosang = max(-1, min(1, dot(a,b)/(na*nb)));
   ang = acos(cosang);
   S = S + ang^2;
end

C = collision_penalty_2d(path, map.obstacles, safety);
R = ziyaret noktası_spacing_penalty_2d(wps, 2.0);

wL=1.0; wS=10.0; wC=200.0; wR=0.8;
J = wL*L + wS*S + wC*C + wR*R;

if ~isfinite(J) || J > 1e15, J = 1e15; end
end

function C = collision_penalty_2d(path, obstacles, safety)
C = 0;
```

```
    nSeg = size(path,1)-1;
    for k = 1:nSeg
       p1 = path(k,:); p2 = path(k+1,:);
       for oi = 1:size(obstacles,1)
          rect = inflate_rect(obstacles(oi,:), safety);
          C = C + segment_rect_sample_penalty(p1, p2, rect);
       end
    end
    end

    function pen = segment_rect_sample_penalty(p1, p2, rect)
    x0=rect(1); y0=rect(2); w=rect(3); h=rect(4);
    xmin=x0; xmax=x0+w; ymin=y0; ymax=y0+h;

    segLen = norm(p2-p1);
    ns = max(15, min(80, ceil(segLen*1.2)));

    t = linspace(0,1,ns);
    pts = p1 + (p2-p1).*t(:);

    inside = (pts(:,1)>=xmin & pts(:,1)<=xmax & pts(:,2)>=ymin &
pts(:,2)<=ymax);
    ratio = sum(inside)/ns;

    distEdge = minDistancePointToRect(p1, rect);
    distEdge = min(distEdge, minDistancePointToRect(p2, rect));

    pen = 0;
    if ratio > 0
       pen = pen + 1 + 10*ratio;
    elseif distEdge < 0.25
       pen = pen + 0.2;
    end
    end
```

```
function d = minDistancePointToRect(p, rect)
x0=rect(1); y0=rect(2); w=rect(3); h=rect(4);
xmin=x0; xmax=x0+w; ymin=y0; ymax=y0+h;

dx = max([xmin - p(1), 0, p(1) - xmax]);
dy = max([ymin - p(2), 0, p(2) - ymax]);
d = hypot(dx, dy);
end

function R = ziyaret noktası_spacing_penalty_2d(wps, dmin)
R = 0;
for i=1:size(wps,1)-1
   d = norm(wps(i+1,:) - wps(i,:));
   if d < dmin
      R = R + (dmin - d)^2;
   end
end
end

% ====================== GA (basit gerçek değerli) ======================
function [bestX, bestCost, histCost] = ga_simple(lb, ub, dim, npop, maxIt, fobj)
pc    = 0.85;
pm    = 0.25;
sigma = 0.10*(ub-lb);

pop  = rand(npop,dim).*(ub-lb) + lb;
cost = zeros(npop,1);
for i=1:npop
   cost(i) = fobj(pop(i,:));
end
```

```
[bestCost, bi] = min(cost);
bestX = pop(bi,:);
histCost = zeros(maxIt,1);

for it=1:maxIt
   newPop = zeros(npop,dim);

   for k=1:2:npop
      p1 = tournament(pop, cost);
      p2 = tournament(pop, cost);

      c1 = p1; c2 = p2;

      % aritmetik crossover
      if rand < pc
         alpha = rand(1,dim);
         c1 = alpha.*p1 + (1-alpha).*p2;
         c2 = alpha.*p2 + (1-alpha).*p1;
      end

      % gauss mutasyon
      if rand < pm, c1 = c1 + sigma.*randn(1,dim); end
      if rand < pm, c2 = c2 + sigma.*randn(1,dim); end

      % sınırlar
      c1 = max(min(c1,ub),lb);
      c2 = max(min(c2,ub),lb);

      newPop(k,:) = c1;
      if k+1 <= npop
         newPop(k+1,:) = c2;
      end
   end
```

```
    pop = newPop;
    for i=1:npop
        cost(i) = fobj(pop(i,:));
    end

    [cmin, bi] = min(cost);
    if cmin < bestCost
        bestCost = cmin;
        bestX = pop(bi,:);
    end

    histCost(it) = bestCost;
end
end

function p = tournament(pop, cost)
n = size(pop,1);
i1 = randi(n);
i2 = randi(n);
if cost(i1) < cost(i2)
    p = pop(i1,:);
else
    p = pop(i2,:);
end
end
```

Ek D: İHA'nın Yol Planlaması İçin Beş Meta-Sezgisel Algoritmanın 3B Yol Planlama Kodu

Ek D.1. Gri Kurt Optimizasyonu (GWO) ile İHA 3B Yol Planlama Kodu

```
%
==========================================================

% Çalıştır: Bu dosyayı direkt RUN

%
==========================================================


clear; clc; close all


%% Harita / Senaryo

map.W = 100; map.H = 100; map.Z = 60;   % 3B sınırlar

start = [5  5  5];

goal  = [95 95 50];


Nwp    = 8;     % ziyaret noktası sayısı

safety = 0.0;   % engelsiz (başlık tutarlılığı için bırakıldı)


dim = 3*Nwp;    % [x1..xN, y1..yN, z1..zN]

lb  = [zeros(1,Nwp) zeros(1,Nwp) zeros(1,Nwp)];

ub  = [map.W*ones(1,Nwp) map.H*ones(1,Nwp) map.Z*ones(1,Nwp)];


costFun = @(x) f_path_cost_uav3d(x, start, goal, map);
```

```
%% Algoritma ayarları
nPop  = 50;   % kurt sayısı
maxIt = 200;

%% GWO çalıştır
fprintf('\n--- GWO (3B) ---\n');
[bestX, bestCost, histCost] = gwo(lb, ub, dim, nPop, maxIt, costFun);
fprintf('GWO -> En iyi maliyet: %.6f\n', bestCost);

%% Yakınsama grafiği
figure; grid on
title('Yakınsama – GWO (İHA 3B Yol Planlama - Engelsiz)');
xlabel('İterasyon'); ylabel('En iyi maliyet');
plot(histCost,'LineWidth',1.6);

%% 3B Rota çizimi
figure; hold on; grid on; box on; axis vis3d
xlim([0 map.W]); ylim([0 map.H]); zlim([0 map.Z]);
title('Yörünge – GWO (İHA 3B Yol Planlama - Engelsiz)');
xlabel('x'); ylabel('y'); zlabel('z');

scatter3(start(1),start(2),start(3),80,'filled');
text(start(1)+1,start(2)+1,start(3)+1,'START');
scatter3(goal(1), goal(2), goal(3), 80,'filled'); text(goal(1)+1, goal(2)+1, goal(3)+1, 'GOAL');
```

```
wps = decode_wps_3d(bestX, Nwp);
pathXYZ = [start; wps; goal];
plot3(pathXYZ(:,1), pathXYZ(:,2), pathXYZ(:,3), 'LineWidth', 2.0);

%% Sonuç raporu
fprintf('\n=== EN İYİ (GWO) | Maliyet: %.6f ===\n', bestCost);

% ======================== FONKSİYONLAR ========================

function wps = decode_wps_3d(x, Nwp)
xw = x(1:Nwp);
yw = x(Nwp+1:2*Nwp);
zw = x(2*Nwp+1:3*Nwp);
wps = [xw(:) yw(:) zw(:)];
end

function J = f_path_cost_uav3d(x, start, goal, map)
% Amaç: uzunluk + pürüzsüzlük + düzenleyici (engelsiz)
if any(~isfinite(x)), J=1e12; return; end

Nwp = numel(x)/3;
wps = decode_wps_3d(x, Nwp);
path = [start; wps; goal];
```

```
% 1) yol uzunluğu
L = sum(vecnorm(diff(path,1,1),2,2));

% 2) pürüzsüzlük: yön değişimi (3B açı)
S = 0;
for i=2:size(path,1)-1
   a = path(i,:) - path(i-1,:);
   b = path(i+1,:) - path(i,:);
   na = norm(a); nb = norm(b);
   if na<1e-9 || nb<1e-9, continue; end
   cosang = max(-1,min(1, dot(a,b)/(na*nb)));
   ang = acos(cosang);
   S = S + ang^2;
end

% 3) düzenleyici: sınır dışı + çok yakın ziyaret noktası
R = ziyaret noktası_spread_penalty_3d(wps, map.W, map.H, map.Z);

% ağırlıklar
wL=1.0; wS=12.0; wR=0.8;

J = wL*L + wS*S + wR*R;

if ~isfinite(J) || J>1e15, J=1e15; end
end
```

```
function R = ziyaret noktası_spread_penalty_3d(wps, W, H, Z)
R = 0;

% sınır dışı
for i=1:size(wps,1)
   if wps(i,1)<0, R=R+abs(wps(i,1)); end
   if wps(i,2)<0, R=R+abs(wps(i,2)); end
   if wps(i,3)<0, R=R+abs(wps(i,3)); end
   if wps(i,1)>W, R=R+abs(wps(i,1)-W); end
   if wps(i,2)>H, R=R+abs(wps(i,2)-H); end
   if wps(i,3)>Z, R=R+abs(wps(i,3)-Z); end
end

% çok yakın ziyaret noktası (3B)
for i=1:size(wps,1)-1
   d = norm(wps(i+1,:)-wps(i,:));
   if d<2.0, R = R + (2.0-d)^2; end
end
end

% -------------------- GWO --------------------
function [Alpha_pos,Alpha_score,histCost] = gwo(lb,ub,dim,nWolves,maxIt,fitness)
Alpha_pos = zeros(1,dim); Alpha_score = inf;
```

```
Beta_pos  = zeros(1,dim); Beta_score  = inf;
Delta_pos = zeros(1,dim); Delta_score = inf;

X = rand(nWolves,dim).*(ub-lb)+lb;
histCost = zeros(maxIt,1);

for it=1:maxIt
   for i=1:nWolves
      X(i,:) = max(min(X(i,:),ub),lb);
      score = fitness(X(i,:));

      if score < Alpha_score
         Delta_score = Beta_score;  Delta_pos = Beta_pos;
         Beta_score  = Alpha_score; Beta_pos  = Alpha_pos;
         Alpha_score = score;       Alpha_pos = X(i,:);
      elseif score < Beta_score
         Delta_score = Beta_score;  Delta_pos = Beta_pos;
         Beta_score  = score;       Beta_pos  = X(i,:);
      elseif score < Delta_score
         Delta_score = score;       Delta_pos = X(i,:);
      end
   end

   a = 2 - it*(2/maxIt);
```

```
    for i=1:nWolves
        for j=1:dim
            r1=rand; r2=rand;
            A1=2*a*r1-a; C1=2*r2;
            D_alpha=abs(C1*Alpha_pos(j)-X(i,j));
            X1=Alpha_pos(j)-A1*D_alpha;

            r1=rand; r2=rand;
            A2=2*a*r1-a; C2=2*r2;
            D_beta=abs(C2*Beta_pos(j)-X(i,j));
            X2=Beta_pos(j)-A2*D_beta;

            r1=rand; r2=rand;
            A3=2*a*r1-a; C3=2*r2;
            D_delta=abs(C3*Delta_pos(j)-X(i,j));
            X3=Delta_pos(j)-A3*D_delta;

            X(i,j)=(X1+X2+X3)/3;
        end
    end

    histCost(it)=Alpha_score;
end
end
```

Ek D.2. PSO ile İHA 3B Yol Planlama Kodu

```
% ============================================================
% Ek D.2 - PSO | İHA 3B Yol Planlama
% Çalıştır: Bu dosyayı direkt RUN
% ============================================================

clear; clc; close all

%% Harita / Senaryo
map.W = 100; map.H = 100; map.Z = 60;
start = [5  5  5];
goal  = [95 95 50];

Nwp    = 8;
safety = 0.0;

dim = 3*Nwp;
lb  = [zeros(1,Nwp) zeros(1,Nwp) zeros(1,Nwp)];
ub  = [map.W*ones(1,Nwp) map.H*ones(1,Nwp) map.Z*ones(1,Nwp)];

costFun = @(x) f_path_cost_uav3d(x, start, goal, map);

%% Algoritma ayarları
nPop  = 50;   % parçacık sayısı
maxIt = 200;

%% PSO çalıştır
```

```
fprintf('\n--- PSO (3B) ---\n');
[bestX, bestCost, histCost] = pso(lb, ub, dim, nPop, maxIt, costFun);
fprintf('PSO -> En iyi maliyet: %.6f\n', bestCost);

%% Yakınsama grafiği
figure; grid on
title('Yakınsama – PSO (İHA 3B Yol Planlama - Engelsiz)');
xlabel('İterasyon'); ylabel('En iyi maliyet');
plot(histCost,'LineWidth',1.6);

%% 3B Rota çizimi
figure; hold on; grid on; box on; axis vis3d
xlim([0 map.W]); ylim([0 map.H]); zlim([0 map.Z]);
title('Yörünge – PSO (İHA 3B Yol Planlama - Engelsiz)');
xlabel('x'); ylabel('y'); zlabel('z');

scatter3(start(1),start(2),start(3),80,'filled');
text(start(1)+1,start(2)+1,start(3)+1,'START');
scatter3(goal(1), goal(2), goal(3), 80,'filled'); text(goal(1)+1, goal(2)+1, goal(3)+1,
'GOAL');

wps = decode_wps_3d(bestX, Nwp);
pathXYZ = [start; wps; goal];
plot3(pathXYZ(:,1), pathXYZ(:,2), pathXYZ(:,3), 'LineWidth', 2.0);

fprintf('\n=== EN İYİ (PSO) | Maliyet: %.6f ===\n', bestCost);
```

```
% ====================== FONKSİYONLAR ======================

function wps = decode_wps_3d(x, Nwp)
xw = x(1:Nwp);
yw = x(Nwp+1:2*Nwp);
zw = x(2*Nwp+1:3*Nwp);
wps = [xw(:) yw(:) zw(:)];
end

function J = f_path_cost_uav3d(x, start, goal, map)
if any(~isfinite(x)), J=1e12; return; end
Nwp = numel(x)/3;
wps = decode_wps_3d(x, Nwp);
path = [start; wps; goal];

L = sum(vecnorm(diff(path,1,1),2,2));

S = 0;
for i=2:size(path,1)-1
   a = path(i,:) - path(i-1,:);
   b = path(i+1,:) - path(i,:);
   na = norm(a); nb = norm(b);
   if na<1e-9 || nb<1e-9, continue; end
   cosang = max(-1,min(1, dot(a,b)/(na*nb)));
   ang = acos(cosang);
   S = S + ang^2;
```

```
end

R = ziyaret noktası_spread_penalty_3d(wps, map.W, map.H, map.Z);

wL=1.0; wS=12.0; wR=0.8;
J = wL*L + wS*S + wR*R;

if ~isfinite(J) || J>1e15, J=1e15; end
end

function R = ziyaret noktası_spread_penalty_3d(wps, W, H, Z)
R = 0;
for i=1:size(wps,1)
   if wps(i,1)<0, R=R+abs(wps(i,1)); end
   if wps(i,2)<0, R=R+abs(wps(i,2)); end
   if wps(i,3)<0, R=R+abs(wps(i,3)); end
   if wps(i,1)>W, R=R+abs(wps(i,1)-W); end
   if wps(i,2)>H, R=R+abs(wps(i,2)-H); end
   if wps(i,3)>Z, R=R+abs(wps(i,3)-Z); end
end
for i=1:size(wps,1)-1
   d = norm(wps(i+1,:)-wps(i,:));
   if d<2.0, R = R + (2.0-d)^2; end
end
end
```

```
% -------------------- PSO --------------------
function [bestX,bestCost,histCost] = pso(lb,ub,dim,nPart,maxIt,fitness)
w  = 0.72; c1 = 1.49; c2 = 1.49;
X  = rand(nPart,dim).*(ub-lb)+lb;
V  = zeros(nPart,dim);

P  = X; PC = inf(nPart,1);
for i=1:nPart, PC(i)=fitness(X(i,:)); end
[GC,gi] = min(PC); G = P(gi,:);

histCost = zeros(maxIt,1);
for it=1:maxIt
   for i=1:nPart
      r1=rand(1,dim); r2=rand(1,dim);
      V(i,:) = w*V(i,:) + c1*r1.*(P(i,:)-X(i,:)) + c2*r2.*(G-X(i,:));
      X(i,:) = X(i,:) + V(i,:);
      X(i,:) = max(min(X(i,:),ub),lb);

      C = fitness(X(i,:));
      if C < PC(i), P(i,:)=X(i,:); PC(i)=C; end
      if PC(i) < GC, G=P(i,:); GC=PC(i); end
   end
   histCost(it)=GC;
end
bestX=G; bestCost=GC;
end
```

Ek D.3. ABC ile İHA 3B Yol Planlama Kodu

```
% ============================================================
% Ek D.3 - ABC | İHA 3B Yol Planlama (ENGELSİZ)
% Çalıştır: Bu dosyayı direkt RUN
% ============================================================

clear; clc; close all

%% Harita / Senaryo
map.W = 100; map.H = 100; map.Z = 60;
start = [5  5  5];
goal  = [95 95 50];

Nwp    = 8;
safety = 0.0;

dim = 3*Nwp;
lb  = [zeros(1,Nwp) zeros(1,Nwp) zeros(1,Nwp)];
ub  = [map.W*ones(1,Nwp) map.H*ones(1,Nwp) map.Z*ones(1,Nwp)];

costFun = @(x) f_path_cost_uav3d(x, start, goal, map);

%% Algoritma ayarları
nPop  = 50;  % food kaynak sayısı
maxIt = 200;
```

```
%% ABC çalıştır
fprintf('\n--- ABC (3B) ---\n');
[bestX, bestCost, histCost] = abc(lb, ub, dim, nPop, maxIt, costFun);
fprintf('ABC -> En iyi maliyet: %.6f\n', bestCost);

%% Yakınsama grafiği
figure; grid on
title('Yakınsama – ABC (İHA 3B Yol Planlama - Engelsiz)');
xlabel('İterasyon'); ylabel('En iyi maliyet');
plot(histCost,'LineWidth',1.6);

%% 3B Rota çizimi
figure; hold on; grid on; box on; axis vis3d
xlim([0 map.W]); ylim([0 map.H]); zlim([0 map.Z]);
title('Yörünge – ABC (İHA 3B Yol Planlama - Engelsiz)');
xlabel('x'); ylabel('y'); zlabel('z');

scatter3(start(1),start(2),start(3),80,'filled');
text(start(1)+1,start(2)+1,start(3)+1,'START');
scatter3(goal(1), goal(2), goal(3), 80,'filled'); text(goal(1)+1, goal(2)+1, goal(3)+1,
'GOAL');

wps = decode_wps_3d(bestX, Nwp);
pathXYZ = [start; wps; goal];
plot3(pathXYZ(:,1), pathXYZ(:,2), pathXYZ(:,3), 'LineWidth', 2.0);

fprintf('\n=== EN İYİ (ABC) | Maliyet: %.6f ===\n', bestCost);
```

```
% ====================== FONKSİYONLAR ======================

function wps = decode_wps_3d(x, Nwp)
xw = x(1:Nwp);
yw = x(Nwp+1:2*Nwp);
zw = x(2*Nwp+1:3*Nwp);
wps = [xw(:) yw(:) zw(:)];
end

function J = f_path_cost_uav3d(x, start, goal, map)
if any(~isfinite(x)), J=1e12; return; end
Nwp = numel(x)/3;
wps = decode_wps_3d(x, Nwp);
path = [start; wps; goal];

L = sum(vecnorm(diff(path,1,1),2,2));

S = 0;
for i=2:size(path,1)-1
   a = path(i,:) - path(i-1,:);
   b = path(i+1,:) - path(i,:);
   na = norm(a); nb = norm(b);
   if na<1e-9 || nb<1e-9, continue; end
   cosang = max(-1,min(1, dot(a,b)/(na*nb)));
   ang = acos(cosang);
```

```
    S = S + ang^2;
end

R = ziyaret noktası_spread_penalty_3d(wps, map.W, map.H, map.Z);

wL=1.0; wS=12.0; wR=0.8;
J = wL*L + wS*S + wR*R;

if ~isfinite(J) || J>1e15, J=1e15; end
end

function R = ziyaret noktası_spread_penalty_3d(wps, W, H, Z)
R = 0;
for i=1:size(wps,1)
    if wps(i,1)<0, R=R+abs(wps(i,1)); end
    if wps(i,2)<0, R=R+abs(wps(i,2)); end
    if wps(i,3)<0, R=R+abs(wps(i,3)); end
    if wps(i,1)>W, R=R+abs(wps(i,1)-W); end
    if wps(i,2)>H, R=R+abs(wps(i,2)-H); end
    if wps(i,3)>Z, R=R+abs(wps(i,3)-Z); end
end
for i=1:size(wps,1)-1
    d = norm(wps(i+1,:)-wps(i,:));
    if d<2.0, R = R + (2.0-d)^2; end
end
end
```

```
% -------------------- ABC --------------------
function [bestX,bestCost,histCost]=abc(lb,ub,dim,SN,maxIt,fitness)
limit = 5*dim;

Foods = rand(SN,dim).*(ub-lb)+lb;
FCost = arrayfun(@(i) fitness(Foods(i,:)), 1:SN)';

trial = zeros(SN,1);
[bestCost,bi]=min(FCost); bestX=Foods(bi,:);
histCost=zeros(maxIt,1);

for it=1:maxIt
   % employed bees
   for i=1:SN
      k=i; while k==i, k=randi(SN); end
      phi = -1+2*rand(1,dim);
      v = Foods(i,:) + phi.*(Foods(i,:)-Foods(k,:));
      v = max(min(v,ub),lb);
      Cv = fitness(v);
      if Cv <= FCost(i)
         Foods(i,:)=v; FCost(i)=Cv; trial(i)=0;
      else
         trial(i)=trial(i)+1;
      end
   end
```

```
% onlooker bees
prob = (1./(1+FCost)); prob = prob/sum(prob);
i=1; t=0;
while t<SN
   if rand < prob(i)
      t=t+1;
      k=i; while k==i, k=randi(SN); end
      phi = -1+2*rand(1,dim);
      v = Foods(i,:) + phi.*(Foods(i,:)-Foods(k,:));
      v = max(min(v,ub),lb);
      Cv = fitness(v);
      if Cv <= FCost(i)
         Foods(i,:)=v; FCost(i)=Cv; trial(i)=0;
      else
         trial(i)=trial(i)+1;
      end
   end
   i=i+1; if i>SN, i=1; end
end

% scout
for i=1:SN
   if trial(i) > limit
      Foods(i,:) = rand(1,dim).*(ub-lb)+lb;
      FCost(i)   = fitness(Foods(i,:));
```

```
            trial(i)=0;
        end
    end

    % best update
    [c,bi]=min(FCost);
    if c<bestCost, bestCost=c; bestX=Foods(bi,:); end
    histCost(it)=bestCost;
end
end
```

Ek D.4. ACO–ACOR ile İHA 3B Yol Planlama Kodu

```
%
==========================================================

% Ek D.4 - ACO(ACOR) | İHA 3B Yol Planlama (ENGELSİZ)
% Çalıştır: Bu dosyayı direkt RUN
%
==========================================================


clear; clc; close all


%% Harita / Senaryo
map.W = 100; map.H = 100; map.Z = 60;
start = [5  5  5];
goal  = [95 95 50];


Nwp    = 8;
```

```
safety = 0.0;

dim = 3*Nwp;
lb  = [zeros(1,Nwp) zeros(1,Nwp) zeros(1,Nwp)];
ub  = [map.W*ones(1,Nwp) map.H*ones(1,Nwp) map.Z*ones(1,Nwp)];

costFun = @(x) f_path_cost_uav3d(x, start, goal, map);

%% Algoritma ayarları
N     = 25;    % ACOR yeni örnek sayısı
maxIt = 200;

%% ACOR çalıştır
fprintf('\n--- ACO (ACOR) (3B) ---\n');
[bestX, bestCost, histCost] = acor(lb, ub, dim, N, maxIt, costFun);
fprintf('ACO (ACOR) -> En iyi maliyet: %.6f\n', bestCost);

%% Yakınsama grafiği
figure; grid on
title('Yakınsama – ACO (ACOR) (İHA 3B Yol Planlama - Engelsiz)');
xlabel('İterasyon'); ylabel('En iyi maliyet');
plot(histCost,'LineWidth',1.6);

%% 3B Rota çizimi
figure; hold on; grid on; box on; axis vis3d
```

```
xlim([0 map.W]); ylim([0 map.H]); zlim([0 map.Z]);
title('Yörünge – ACO (ACOR) (İHA 3B Yol Planlama - Engelsiz)');
xlabel('x'); ylabel('y'); zlabel('z');

scatter3(start(1),start(2),start(3),80,'filled');
text(start(1)+1,start(2)+1,start(3)+1,'START');
scatter3(goal(1), goal(2), goal(3), 80,'filled'); text(goal(1)+1, goal(2)+1, goal(3)+1, 'GOAL');

wps = decode_wps_3d(bestX, Nwp);
pathXYZ = [start; wps; goal];
plot3(pathXYZ(:,1), pathXYZ(:,2), pathXYZ(:,3), 'LineWidth', 2.0);

fprintf('\n=== EN İYİ (ACO/ACOR) | Maliyet: %.6f ===\n', bestCost);

% ====================== FONKSİYONLAR ======================

function wps = decode_wps_3d(x, Nwp)
xw = x(1:Nwp);
yw = x(Nwp+1:2*Nwp);
zw = x(2*Nwp+1:3*Nwp);
wps = [xw(:) yw(:) zw(:)];
end

function J = f_path_cost_uav3d(x, start, goal, map)
```

```
if any(~isfinite(x)), J=1e12; return; end
Nwp = numel(x)/3;
wps = decode_wps_3d(x, Nwp);
path = [start; wps; goal];

L = sum(vecnorm(diff(path,1,1),2,2));

S = 0;
for i=2:size(path,1)-1
   a = path(i,:) - path(i-1,:);
   b = path(i+1,:) - path(i,:);
   na = norm(a); nb = norm(b);
   if na<1e-9 || nb<1e-9, continue; end
   cosang = max(-1,min(1, dot(a,b)/(na*nb)));
   ang = acos(cosang);
   S = S + ang^2;
end

R = ziyaret noktası_spread_penalty_3d(wps, map.W, map.H, map.Z);

wL=1.0; wS=12.0; wR=0.8;
J = wL*L + wS*S + wR*R;

if ~isfinite(J) || J>1e15, J=1e15; end
end
```

```
function R = ziyaret noktası_spread_penalty_3d(wps, W, H, Z)
R = 0;
for i=1:size(wps,1)
   if wps(i,1)<0, R=R+abs(wps(i,1)); end
   if wps(i,2)<0, R=R+abs(wps(i,2)); end
   if wps(i,3)<0, R=R+abs(wps(i,3)); end
   if wps(i,1)>W, R=R+abs(wps(i,1)-W); end
   if wps(i,2)>H, R=R+abs(wps(i,2)-H); end
   if wps(i,3)>Z, R=R+abs(wps(i,3)-Z); end
end
for i=1:size(wps,1)-1
   d = norm(wps(i+1,:)-wps(i,:));
   if d<2.0, R = R + (2.0-d)^2; end
end
end

% -------------------- ACOR (Sürekli ACO) --------------------
function [bestX,bestCost,histCost]=acor(lb,ub,dim,N,maxIt,fitness)
q  = 0.5; xi = 0.85;
M = 2*N;

A = rand(M,dim).*(ub-lb)+lb;
ACost = arrayfun(@(i) fitness(A(i,:)), 1:M)';
```

```
[ACost,idx] = sort(ACost);
A = A(idx,:);

ranks = (1:M)';
w = (1/(q*M*sqrt(2*pi))) * exp(- (ranks-1).^2 / (2*(q*M)^2));
w = w / sum(w);

histCost = zeros(maxIt,1);

for it=1:maxIt
   S = zeros(N,dim);
   for n=1:N
      for d=1:dim
         mu_d = A(:,d);
         sigma_d = xi * abs(mu_d - mu_d(1));
         if all(sigma_d==0), sigma_d(:) = (ub(d)-lb(d))/50; end
         k = randsample(M,1,true,w);
         S(n,d) = mu_d(k) + sigma_d(k)*randn;
      end
      S(n,:) = max(min(S(n,:),ub),lb);
   end

   SCost = arrayfun(@(i) fitness(S(i,:)), 1:N)';

   A = [A; S];
```

```
    ACost = [ACost; SCost];

    [ACost,idx] = sort(ACost);
    A = A(idx,:);

    A = A(1:M,:);
    ACost = ACost(1:M);

    histCost(it) = ACost(1);
end

bestX = A(1,:);
bestCost = ACost(1);
end
```

Ek D.5. GA ile İHA 3B Yol Planlama Kodu

```
%
==========================================================
% Ek D.5 - GA | İHA 3B Yol Planlama
% Çalıştır: Bu dosyayı direkt RUN
%
==========================================================

clear; clc; close all
```

```
%% Harita / Senaryo
map.W = 100; map.H = 100; map.Z = 60;
start = [5  5  5];
goal  = [95 95 50];

Nwp    = 8;
safety = 0.0;

dim = 3*Nwp;
lb  = [zeros(1,Nwp) zeros(1,Nwp) zeros(1,Nwp)];
ub  = [map.W*ones(1,Nwp) map.H*ones(1,Nwp) map.Z*ones(1,Nwp)];

costFun = @(x) f_path_cost_uav3d(x, start, goal, map);

%% Algoritma ayarları
nPop  = 50;  % popülasyon
maxIt = 200;

%% GA çalıştır
fprintf('\n--- GA (3B) ---\n');
[bestX, bestCost, histCost] = ga_simple(lb, ub, dim, nPop, maxIt, costFun);
fprintf('GA -> En iyi maliyet: %.6f\n', bestCost);

%% Yakınsama grafiği
figure; grid on
```

```
title('Yakınsama – GA (İHA 3B Yol Planlama - Engelsiz)');
xlabel('İterasyon'); ylabel('En iyi maliyet');
plot(histCost,'LineWidth',1.6);

%% 3B Rota çizimi
figure; hold on; grid on; box on; axis vis3d
xlim([0 map.W]); ylim([0 map.H]); zlim([0 map.Z]);
title('Yörünge – GA (İHA 3B Yol Planlama - Engelsiz)');
xlabel('x'); ylabel('y'); zlabel('z');

scatter3(start(1),start(2),start(3),80,'filled');
text(start(1)+1,start(2)+1,start(3)+1,'START');
scatter3(goal(1), goal(2), goal(3), 80,'filled'); text(goal(1)+1, goal(2)+1, goal(3)+1, 'GOAL');

wps = decode_wps_3d(bestX, Nwp);
pathXYZ = [start; wps; goal];
plot3(pathXYZ(:,1), pathXYZ(:,2), pathXYZ(:,3), 'LineWidth', 2.0);

fprintf('\n=== EN İYİ (GA) | Maliyet: %.6f ===\n', bestCost);

% ====================== FONKSİYONLAR ======================

function wps = decode_wps_3d(x, Nwp)
xw = x(1:Nwp);
```

```
yw = x(Nwp+1:2*Nwp);
zw = x(2*Nwp+1:3*Nwp);
wps = [xw(:) yw(:) zw(:)];
end

function J = f_path_cost_uav3d(x, start, goal, map)
if any(~isfinite(x)), J=1e12; return; end
Nwp = numel(x)/3;
wps = decode_wps_3d(x, Nwp);
path = [start; wps; goal];

L = sum(vecnorm(diff(path,1,1),2,2));

S = 0;
for i=2:size(path,1)-1
    a = path(i,:) - path(i-1,:);
    b = path(i+1,:) - path(i,:);
    na = norm(a); nb = norm(b);
    if na<1e-9 || nb<1e-9, continue; end
    cosang = max(-1,min(1, dot(a,b)/(na*nb)));
    ang = acos(cosang);
    S = S + ang^2;
end

R = ziyaret noktası_spread_penalty_3d(wps, map.W, map.H, map.Z);
```

```
wL=1.0; wS=12.0; wR=0.8;
J = wL*L + wS*S + wR*R;

if ~isfinite(J) || J>1e15, J=1e15; end
end

function R = ziyaret noktası_spread_penalty_3d(wps, W, H, Z)
R = 0;
for i=1:size(wps,1)
   if wps(i,1)<0, R=R+abs(wps(i,1)); end
   if wps(i,2)<0, R=R+abs(wps(i,2)); end
   if wps(i,3)<0, R=R+abs(wps(i,3)); end
   if wps(i,1)>W, R=R+abs(wps(i,1)-W); end
   if wps(i,2)>H, R=R+abs(wps(i,2)-H); end
   if wps(i,3)>Z, R=R+abs(wps(i,3)-Z); end
end
for i=1:size(wps,1)-1
   d = norm(wps(i+1,:)-wps(i,:));
   if d<2.0, R = R + (2.0-d)^2; end
end
end

% -------------------- GA --------------------
function [bestX,bestCost,histCost]=ga_simple(lb,ub,dim,npop,maxIt,fitness)
```

```
pc = 0.85; pm = 0.25; sigma = 0.10*(ub-lb);

pop  = rand(npop,dim).*(ub-lb)+lb;
cost = zeros(npop,1);
for i=1:npop, cost(i)=fitness(pop(i,:)); end

[bestCost,idx]=min(cost); bestX = pop(idx,:);
histCost=zeros(maxIt,1);

for it=1:maxIt
   newPop = zeros(npop,dim);
   for k=1:2:npop
      p1 = tournament(pop,cost);
      p2 = tournament(pop,cost);

      c1=p1; c2=p2;
      if rand<pc
         alpha = rand(1,dim);
         c1 = alpha.*p1 + (1-alpha).*p2;
         c2 = alpha.*p2 + (1-alpha).*p1;
      end

      if rand<pm, c1 = c1 + sigma.*randn(1,dim); end
      if rand<pm, c2 = c2 + sigma.*randn(1,dim); end
```

```
        c1 = max(min(c1,ub),lb);
        c2 = max(min(c2,ub),lb);

        newPop(k,:) = c1;
        if k+1<=npop, newPop(k+1,:) = c2; end
    end

    pop = newPop;
    for i=1:npop, cost(i)=fitness(pop(i,:)); end

    [cmin,idx]=min(cost);
    if cmin<bestCost, bestCost=cmin; bestX=pop(idx,:); end
    histCost(it)=bestCost;
end
end

function p = tournament(pop,cost)
n=size(pop,1);
i1=randi(n); i2=randi(n);
if cost(i1)<cost(i2), p=pop(i1,:); else, p=pop(i2,:); end
end
```

7 KAYNAKÇA

Anderson, P., & Green, L. (2024). Comparative analysis of metaheuristic algorithms for control optimization problems. Multibody System Dynamics. https://www.springer.com/journal/11071

Bowthiya, S., & Rampriya, B. (2022). PID tuning using soft computing techniques. Multibody System Dynamics. https://www.springer.com/journal/11071

Brown, A., & Wilson, M. (2022). Hybrid metaheuristic approaches for navigation path planning. Multibody System Dynamics. https://www.springer.com/journal/11071

Davis, S., & Anderson, J. (2023). Comparative analysis of metaheuristic algorithms for autonomous path planning. Multibody System Dynamics. https://www.springer.com/journal/11071

Ghambari, M. (2024). Survey on UAV path planning with metaheuristics. Multibody System Dynamics. https://www.springer.com/journal/11071

Green, A., & White, J. (2023). Application of metaheuristic algorithms in structural optimization. Multibody System Dynamics. https://www.springer.com/journal/11071

Green, A., & Wilson, B. (2023). Optimization of control parameters using metaheuristic algorithms. Multibody System Dynamics. https://www.springer.com/journal/11071

Johnson, T., & Brown, A. (2023). Hybrid metaheuristic techniques for real-time control applications. Multibody System Dynamics. https://www.springer.com/journal/11071

Johnson, T., & Green, A. (2024). Application of metaheuristic optimization in mobile robot path planning. Multibody System Dynamics. https://www.springer.com/journal/11071

Joseph, R. (2022). Metaheuristic algorithms for PID controller parameters tuning. Multibody System Dynamics. https://www.springer.com/journal/11071

Joyo, F. (2025). Optimal PID controller with evolutionary techniques. Multibody System Dynamics. https://www.springer.com/journal/11071

Kim, S., Maruta, I., & Sugie, T. (2008). Robust PID controller tuning using ALPSO. Multibody System Dynamics. https://link.springer.com/article/10.1007/s11044-008-9092-8

Kumar, S., & Patel, R. (2022). Hybrid metaheuristic approaches for complex optimization problems. Multibody System Dynamics. https://www.springer.com/journal/11071

Lee, P., & Martinez, T. (2024). Comparative study of metaheuristic algorithms for engineering optimization. Multibody System Dynamics. https://www.springer.com/journal/11071

Poudel, S., & Moh, T. (2023). Bio-inspired UAV path planning: A survey. Multibody System Dynamics. https://link.springer.com/article/10.1007/s11044-023-09962-2

Poudel, S., et al. (2023). Hybrid PSO-GA and PSO-HSA for 3D path planning. Multibody System Dynamics. https://link.springer.com/article/10.1007/s11044-023-09974-y

Shafiq, M., et al. (2022). Multi-UAV path planning with hybrid ACO. Multibody System Dynamics. https://link.springer.com/article/10.1007/s11044-022-09863-8

Smith, J., & Zhao, L. (2022). Metaheuristic methods for real-time path planning in dynamic environments. Multibody System Dynamics. https://www.springer.com/journal/11071

Taeib, M., Ltaeif, A., & Chaari, A. (2013). PSO for multivariable PID in nonlinear MIMO systems. Multibody System Dynamics. https://link.springer.com/article/10.1007/s11044-013-9365-2

Teng, Y., et al. (2025). Improved grey wolf optimizer for UAV shortest path planning. Multibody System Dynamics. https://www.springer.com/journal/11071

Young, H. D., & Freedman, R. A. (2019). University Physics Volume 1. Pearson. https://www.pearson.com/store/p/university-physics-volume-1/P100003861055

Zhang, L., & Chen, M. (2024). Metaheuristic optimization for multidisciplinary design optimization problems. Multibody System Dynamics. https://www.springer.com/journal/11071