GENETİK ALGORİTMA TABANLI BULANIK KONROLÜN UÇUŞ KONTROL SİSTEM TASARIMINA UYGULANMASI

Yasemin IŞIK
Doktora Tezi

Fen Bilimleri Enstitüsü
Sivil Havacılık Anabilim Dalı
Kasım – 2006

JÜRİ VE ENSTİTÜ ONAYI

Yasemin Işık'ın **"Genetik Algoritma Tabanlı Bulanık Kontrolün Uçuş Kontrol Sistem Tasarımına Uygulanması"** başlıklı **Sivil Havacılık** Anabilim Dalındaki, Doktora Tezi 24.11.2006 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı-Soyadı	**İmza**
Üye (Tez Danışmanı)	**: Yard. Doç. Dr. AYŞE KAHVECİOĞLU**	**.....**
Üye	**: Prof. Dr. ABDURRAHMAN KARAMANCIOĞLU**	**.....**
Üye	**: Doç. Dr. OSMAN PARLAKTUNA**	**.....**
Üye	**: Yard. Doç. Dr. HAKAN KORUL**	**.....**
Üye	**: Yard. Doç. Dr. MUZAFFER KAPANOĞLU**	**.....**

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …… tarih ve …… sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ÖZET

Doktora Tezi

GENETİK ALGORİTMA TABANLI BULANIK KONTROLÜN UÇUŞ KONTROL SİSTEM TASARIMINA UYGULANMASI

Yasemin IŞIK

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Sivil Havacılık Anabilim Dalı

Danışman: Yard. Doç. Dr. Ayşe KAHVECİOĞLU
2006, 189 sayfa

Uçağın uçuş zarfı içinde istenen bir uçuş durumunda uzun bir süre tutulması başlıca uçuş kontrol problemidir. İç ve dış ortam uyarılarından kaynaklanan bozucu etkilerden dolayı, uçağın dinamik durumu kısa bir süre içinde değişebilir. Dolayısıyla herhangi bir uçuş durumu için tasarlanmış bir kontrol sistemi (klasik kontrol sistemleri), bu denge durumundan sapma halinde istenen kararlılık ve performans özelliklerini vermeyebilir. Literatürde uçuş kontrol sistem tasarımında farklı kontrol uygulamalarının kullanıldığı görülmektedir. Bu kontrol uygulamalarından olan bulanık kontrol, sistem transfer fonksiyonunun tanımlanmadığı fakat uygulanacak denetimin sözel olarak ifade edildiği sistemlerde kullanılmaktadır. Ayrıca, transfer fonksiyonu bilinen sistemlerde klasik denetleyicilerin yerine sistem performansını iyileştirme amacıyla da uygulanabilir. Bulanık sistemlerin en iyilemesi için genetik algoritmaların kullanılması da oldukça yaygın bir yöntemdir. Bulanık uçuş kontrol sistem tasarımı konusunda yapılan çalışmaların büyük çoğunluğu ise benzetim seviyesindedir. Bu çalışmada uçuş kontrol sisteminde önemli bir yer tutan irtifa kontrolünün genetik bulanık PD denetleyici kullanılarak tasarımı incelenmiş ve geniş gövdeli, dört motorlu jet yolcu uçağına ait elde edilen uygulama sonuçları, klasik PD denetleyici kullanılarak elde edilen tepkilerle birlikte değerlendirilmiştir.

Anahtar Kelimeler: Bulanık Mantık, Genetik Algoritmalar, Bulanık Kontrol, Uçuş Kontrol Sistem Tasarımı, Genetik Bulanık Denetleyici

ABSTRACT

PhD Dissertation

FLIGHT CONTROL SYSTEM DESIGN USING GENETIC FUZZY CONTROL

Yasemin IŞIK

Anadolu University
Graduate School of Sciences
Civil Aviation Program

Supervisor: Asist. Prof. Dr. Ayşe KAHVECİOĞLU
2006, 189 pages

Maintaining desired filght conditions in flight envelope for a long period is the primary control problem in aircraft flight control. Due to internal and external disturbances, dynamic behaviour of aircraft may change in a short period of time. Thus a control system designed for a flight condition, may not provide the desired stability and performance characteristics in case of deviation from the equilibrium point. There are numerous studies regarding flight control in the literature. Fuzzy logic controllers (FLCs) from their inception have demonstrated a vast range of applicability to processes where the plant transfer function is not defined but the control action can be described in terms of linguistic variables. FLC's are also being used with improved performance instead of "classical" controllers where the plant transfer function is known. Genetic algorithms is a method used for optimization of fuzzy systems. Most of the applications about the design of fuzzy flight control are in simulation level. In this study, the design of genetic fuzzy PD controller for the altitude control system is analyzed and the results for a very large, four-engine passenger jet aircraft are evaluated and compared with the result of classsical PD controller.

Keywords: Fuzzy Logic, Genetic Algorithms, Fuzzy Control, Design of Flight Control System, Genetic Fuzzy Controller

TEŞEKKÜR

Çalışmam süresince her türlü yardımı gösteren danışman sayın hocam Yard. Doç. Dr. Ayşe KAHVECİOĞLU'na, tez izleme hocalarım sayın Doç. Dr. Osman PARLAKTUNA'ya, sayın Yard. Doç. Dr. Hakan KORUL'a, eleştiri ve destekleri için sayın Yard. Doç. Dr. Muzaffer KAPANOĞLU'na ve Sivil Havacılık Yüksekokulundan arkadaşlarım Emre KIYAK, Korkut KIZILDERE ve Sinem KAHVECİOĞLU'na teşekkür ederim.

Ayrıca tezin her aşamasında bana manevi desteğini esirgemeyen aileme ve arkadaşlarıma teşekkür ederim.

Yasemin IŞIK

Kasım, 2006

İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

e	: Hata
$\dot{e}$	: Hatanın değişimi
$\mathbf{x}$	: Durum vektörü
$\mathbf{u}$	: Kontrol girdi vektörü
ξ	: Bozucu etki
t	: Zaman (sn)
p_c	: Çaprazlama oranı
p_m	: Mutasyon oranı
H	: Hata
NCK	: Negatif çok küçük
NK	: Negatif küçük
NASA	: National Aeronautics Space Administration
Y	: Yok
PB	: Pozitif büyük
PCB	: Pozitif çok büyük
$\mathbf{A}$	: Durum katsayılar matrisi
$\mathbf{B}$	: Girdi katsayılar matrisi
δ_E	: İrtifa dümeni açısı (rad)
$X_{(.)}, Y_{(.)}, Z_{(.)}$	: Kararlılık türevleri
U	: İleri doğru hız (ms^{-1})
q	: Yunuslama açısal hızı (rad^{-1})
θ	: Yunuslama açısı (rad)
g	: Yerçekimi vektörü (ms^{-2})
α	: Hücum açısı (rad)
$(.)_0$	: Denge durumu
$\overline{q}$	: Dinamik basınç (Nm^{-2})
γ	: Yörünge açısı (rad)

1. GİRİŞ

Herhangi bir tip aracın hareketi ele alındığında, bu hareketi aracın hız vektörüyle tam olarak karakterize etmek mümkündür. Bu vektörün zamana göre integrali aracın uzay içindeki yoludur. $\dot{x}$ ile gösterilen hız vektörü aracın uzay içindeki pozisyonunun (x), kontrol girdisinin (u), herhangi bir bozucu etkinin (ξ) ve zamanın (t) fonksiyonu şeklinde gösterilebilir.

$$\dot{x} = f(x,u,\xi,t) \tag{1.1}$$

Burada f bir vektör fonksiyonudur.

Uçağın uçuş zarfı içinde istenen bir uçuş durumunda uzun bir süre tutulması başlıca kontrol problemidir. İç ve dış ortam uyarılarından kaynaklanan bozucu etkilerden dolayı, uçağın dinamik durumu kısa bir süre içinde değişebilir. Bu nedenle herhangi bir uçuş durumu için tasarlanmış bir kontrol sistemi (klasik kontrol sistemleri), bu denge durumundan sapma halinde istenen kararlılık ve performans özelliklerini vermeyebilir. Örneğin, beklenmedik durumları (değişen hava şartları, sistem arızaları vb.) modelleyerek uygun bir klasik kontrol sistem tasarımı oldukça güçtür [1,2].

Uçuş kontrol sistem tasarımında geliştirilen diğer yöntemler; uyarlamalı kontrol [3,4], μ sentez kontrol [5,6], H_∞ kontrol [7,8], kazanç ölçeklemeli kontrol [9,10], yapay sinir ağları kontrolü [11,12], katlı model yaklaşımlı kontrol [13,14] ve bulanık kontrol [15] şeklinde sıralanabilir.

Bu yöntemlerden bulanık kontrol, kontrol edilmesi gereken sürecin durum bilgisi ve kontrol işlemi arasındaki bulanık işlem algoritmasına dayanır. Bir sistem için matematiksel modelden ziyade, sistemin çalışması ile ilgili bilgilerin elde olduğu belirsizlik durumlarında, özellikle sistem transfer fonksiyonunun tanımlanmadığı fakat uygulanacak denetimin sözel olarak ifade edildiği sistemlerde bulanık kontrol kullanılmaktadır. Ayrıca transfer fonksiyonu bilinen sistemlerde klasik denetleyicilerin yerine sistem performansını iyileştirme amacıyla da uygulanabilir [16,17]. Bulanık sistemlerin optimizasyonu (uygun üyelik fonksiyonlarının ve kural ağırlık değerlerinin oluşturulması) için kullanılan yöntemlerden biri de genetik algoritmalardır.

Bu çalışmada uçuş kontrol sisteminde önemli bir yer tutan irtifa kontrolünün genetik bulanık PD denetleyici kullanılarak tasarımı incelenmiş ve geniş gövdeli, dört motorlu jet yolcu uçağına ait elde edilen uygulama sonuçları, klasik PD denetleyici kullanılarak elde edilen tepkilerle birlikte değerlendirilmiştir.

İkinci bölümde ilk olarak bulanık mantık, bulanık mantığın tarihçesi, bulanık sistemlerin gelişimi incelendikten sonra bulanık sistemlerin yapısı, bulanık kontrol ve bulanık PD kontrol ayrıntılı olarak anlatılmıştır.

Üçüncü bölümde anlatılan genetik algoritmalar (GA) ise, evrim teorisinden esinlenerek türetilen hesaplama modelleridir. Oldukça geniş uygulama alanına sahip olan genetik algoritmalar daha çok bir en iyileme yöntemi olarak görülür [1].

Dördüncü bölümde, genetik algoritma tabanlı bulanık PD denetleyici yapısı ayrıntılı olarak verildikten sonra irtifa kontrolünün klasik PD ve genetik algoritma tabanlı bulanık PD denetleyici kullanılarak tasarımı incelenmiş ve geniş gövdeli, dört motorlu jet yolcu uçağına ait elde edilen sonuçlar farklı uçuş durumları için karşılaştırılmıştır.

2. BULANIK MANTIK

2.1. Giriş

Her insan günlük hayatında kesin olarak bilinemeyen bazen de önceden kesinmiş gibi düşünülen, ama sonuçta kesinlik arz etmeyen durumlarla karşılaşır. Bu durumların örgün (sistematik) bir şekilde önceden planlanarak sayısal öngörülerinin yapılması ise, ancak bir takım kabul ve varsayımlardan sonra mümkün olabilmektedir.

Gerçek dünya karmaşıktır. Bu karmaşıklık genel olarak belirsizlik, kesin düşünceden yoksunluk ve karar verilemeyişten kaynaklanır. Birçok sosyal, iktisadi ve teknik konularda insan düşüncesinin tam anlamı ile olgunlaşmamış oluşundan dolayı belirsizlikler her zaman bulunur. İnsan tarafından geliştirilmiş olan bilgisayarlar, bu tür belirsizlikleri işleyemezler ve çalışmaları için sayısal bilgiler gereklidir. Gerçek bir olayın kavranılması insan bilgisinin yetersizliği sebebiyle tam anlamı ile mümkün olamadığından insan, düşünce sisteminde ve zihninde bu gibi olayları yaklaşık olarak canlandırarak yorumlarda bulunur. Bilgisayarlardan farklı olarak insanın, yaklaşık düşünme, oldukça yetersiz, eksik ve belirsizlik içeren veri ve bilgi ile işlem yapabilme yeteneği vardır. Genel olarak, değişik biçimlerde ortaya çıkan karmaşıklık ve belirsizlik gibi tam ve kesin olmayan bilgi kaynaklarına bulanık (fuzzy) kaynaklar adı verilir [18].

Bulanık mantık ise belirsizliklerin anlatımı ve belirsizliklerle çalışılabilmesi için kurulmuş bir matematik düzen olarak tanımlanabilir [19].

Bulanık mantığın temel amacı, insanların tam ve kesin olmayan bilgiler ışığında, tutarlı ve doğru kararlar vermelerini sağlayan, düşünme ve karar verme mekanizmalarının modellenmesidir. Temel olarak çok değerli mantık, olasılık, yapay zekâ, yapay sinir ağları alanları ile ilişkilidir [20].

Bulanık mantık yaklaşımı, makinelere insanların özel verilerini işleyebilme ve onların deneyimlerinden ve önsezilerinden yararlanarak çalışabilme yeteneği verir. Bu yeteneği kazandırırken sayısal ifadeler yerine sembolik ifadeler kullanır. İşte bu sembolik ifadelerin makinelere aktarılması matematiksel bir temele dayanır. Bu matematiksel temel, bulanık mantık kümeler kuramı ve buna dayanan bulanık mantıktır [21].

2.2. Bulanık Mantığın Tarihçesi

Matematiğin doğruluğundaki ve bütünlüğündeki başarısında Aristoteles'in ve onun izinden giden düşünürlerin büyük katkısı olmuştur. Onların mantık teorisini oluşturma çabaları ile matematik gelişmiş ve "Düşüncenin Yasaları" oluşturulmuştur. Bu yasalardan biri, her önermenin ya "Doğru" ya da "Yanlış" olması gerektiğini öngörmüştür. Bu anlayışa geleneksel anlayış ya da "Aristo Mantığı" adı verilir.

Bu yüzyılda matematik ve bilimde görülen çeşitli fikir değişiklikleri arasında belirsizlik kavramıyla ilgili olanı belki de en dikkat çekici olanıdır. Bilimde bu değişiklik, belirsizliği istenilmeyen bir durum olarak gören ve mümkün bütün durumlarda kaçınılması gerektiğinde ısrar eden geleneksel anlayıştan, belirsizliği tolere eden ve bilimde bundan kaçınılmasının mümkün olmadığını iddia eden alternatif bakış açısına doğru bir geçişle ortaya konulmuştur. Geleneksel yaklaşımdaki anlayış, bilimin ortaya koyduğu açıklamalarda bir kesinliğe ulaşılmasının gerekliliğidir. Bundan dolayı da belirsizlik bilimsel olmayan bir kavram olarak kabul görülmüştür. Alternatif bakış açısına göre ise belirsizlik, sadece kaçınılması mümkün olmayan bir durum değil, aynı zamanda büyük bir fayda sağlayan ve üzerinde çalışılması gereken önemli bir durumdur [16].

Heisenberg'in belirsizlik ilkesi 1920'ler ve 1930'larda çok değerli mantık sistemlerinin gelişmesine yol açmıştır. Kuantum araştırmacıları iki değerli mantık sistemlerinin 'doğru' ve 'yanlış'tan oluşan değer kümesine bir üçüncü değeri ekleyerek belirsizliğin ifade edilmesine imkân sağlamışlardır. Bundan sonraki aşamada 'doğru' ve 'yanlış', belirsizliğin sınır koşulları olarak görülüp belirsizlik derecelendirilmiştir. Az sayıda batılı filozof çok değerliliği benimsemesine rağmen Lukasiewicz, Godel ve Black bulanık mantık küme kavramlarını geliştirmişlerdir. 1930'ların başında Polonyalı mantıkçı Jan Lukasiewicz üç değerli mantık sistemini geliştirmiştir [22]. Kuantum filozofu Max Black 1937 yılında "Philosophy of Science" dergisinde "Vagueness: An Exercise in Logical Analysis" isimli makalesinde bulanık küme üyelik fonksiyonlarından bahseden ilk kişi olmuştur [22, 23].

Max Black tarafından belirsizliği açıklayıcı öncü kavramlar geliştirilmiş olsa da, 1965'de California Üniversitesinden Prof. Dr. Lotfi Zadeh tarafından "Information and Controller" dergisinde yayınlanan "Fuzzy Sets" isimli makale modern anlamda belirsizlik kavramının değerlendirilmesinde önemli bir nokta olarak kabul edilir. Zadeh bu makalede, kesin olmayan sınırlara sahip nesnelerin oluşturduğu bulanık küme teorisini ortaya koymuştur. Zadeh'in bu makalesinin önemi sadece 'İhtimaller Teorisi'ne karşı duruşu ile ilgili değil, ayrıca ihtimaller teorisinin temelini oluşturan Aristo Mantığı'na karşı da bir meydan okumadır [16,22].

Bulanık Mantık hakkında ilk bilgiler, Azerbaycan asıllı Zadeh tarafından literatüre maledilmesine karşılık bu fikirler Batı dünyasında şüpheyle karşılanmış ve oldukça yoğun tenkit almıştır. Ancak 1970 yıllarından sonra Doğu dünyasında, özellikle Japonya'da bulanık mantık ve sistem kavramlarına önem verilmiştir. Bunların teknolojik cihaz yapım ve işleyişinde kullanılması, günümüzde, bütün dünyada yaygın hale gelmiştir. Batıda gecikmenin ana sebebi batı kültürünün temelinde ikili mantık yani Aristo mantığının ve olaylara evet-hayır, beyaz-siyah, kurak-sulak, artı-eksi, 0–1 vb. gibi ikili esasta yaklaşılmasıdır. Bu iki değer arasında başka seçeneklere kesin değil düşüncesi ile yer verilmemiştir. Batıda bulanık kelimesi, güvensizliği ifade eder. Doğuda ise bu güvensizlikte bile güzelliklerin bulunabileceği düşüncesi vardır.

Zadeh tarafından ortaya atılan Bulanık Küme, Mantık ve Sistem Kavramları, bu araştırmacının uzun yıllar boyunca kontrol alanında çalışması, istediği kontrolü elde edebilmesi için fazlaca doğrusal olmayan denklemlerin işin içine girmesi, yöntemin karmaşıklaşması ve çözümün zorlaşması neticesinde ortaya çıkmıştır.

Bulanık kavramların ortaya atılmasıyla beraber özellikle ihtimaller teorisi ve istatistik gibi zaten belirsizlikleri konu edinen bilim dalları bulunduğundan bu konularda çalışan araştırmacılar, bulanık sistemlere açık bir şekilde karşı çıkmışlardır. Bulanık yöntemlerle yapılan her türlü hesaplamanın, ihtimal ve istatistik hesaplamalarla yapılabileceğini ileri sürmüşlerdir.

İlk ortaya atıldığı zamanlarda bulanık sistemlerin doğrudan uygulaması olmadığından yapılan tartışmalar daha ziyade felsefi seviyede kalmış ve bunun sonucunda daha kuvvetli felsefi ve teorik temelleri olan ihtimaller teorisi ve

istatistik yöntemler ağır basmıştır. Ancak burada gözden kaçan husus, sözel bilgilerin bulunması halinde istatistiğin fazlaca işe yaramamasıdır. Her ne kadar Bayesian teorisi gibi bir istatistiksel yöntem ile sözel bazı ifadelerin hesaplamalarda kullanılması mümkün olsa da, bu yöntemlerin işleyişinde normal dağılmış olmak, doğrusal olmak gibi bazı temel kabuller pratikte iyi sonuçlar vermemiştir [16].

2.3. Bulanık Sistemlerin Gelişimi

Bulanık kavram ve sistemlerin dünyanın değişik araştırma merkezlerinde dikkat kazanması, 1975 yılında Mamdani ve Asilyan tarafından yapılan gerçek bir kontrol uygulamasıyla olmuştur. Bu araştırmacılar, ilk defa bir buhar makinesi kontrolünün bulanık sistem ile modellenmesini başarmıştır [18]. Bu uygulama sonucunda, doğrusal olmayan kontrol problemleri için bulanık denetleyicinin klasik denetleyicilere göre çok daha kolay geliştirildiği ve oldukça iyi sonuçlar verdiği belirtilmiştir. 1978 yılında da ilk kez bir çimento fırınına tüm endüstriyel süreç için bulanık bir denetleyici geliştirilmiştir. 1980'de Sugeno, Japonların ilk bulanık mantık uygulaması olan Fuji elektrik su arıtma tesisinin kontrolünü gerçekleştirmiştir [20].

1987'de ikinci IFSA kongresinde ilk bulanık mantık denetleyicileri sergilenmiştir. Bu denetimler 1984 yılında araştırmalara başlayan Omron şirketinin yaptığı 700'den fazla uygulamayı içermektedir [21].

Kronolojik sıra içerisinde bundan sonraki en önemli aşama Japonya'da 1987 yılında görülmüştür. Hitachi firması, Sedai Metro sisteminde çalışan trenlerin otomatik olarak denetimi için bulanık mantık sistemini geliştirmiştir.

Bu faaliyetler Batıda çok yavaş olurken, doğuda özellikle Japonya, Singapur, Kore ve Malezya'da kendisini daha fazla göstermiştir. Teknolojiye duyarlı olan Japon mühendisler bulanık kontrol birimlerini kurmanın ne kadar kolay olduğunu görerek, bunları birçok cihazın yapımında kullanmaya başlamışlardır.

1980'den sonra bulanık sistemlerin; elektrikli süpürge, çamaşır makinesi, asansör, televizyon, müzik seti, metro ve şirket işletimi gibi konularda

kullanılmasında patlama olmuştur. Son yıllarda birçok mühendislik dallarında, veri tabanlarının sözelleştirilmesinde, tele-sekreterlerin cevaplamasında ve birçok konuda (arabalarda yakıt püskürtme ve ateşleme sistemlerinin denetimi, ısı, elektrik akımı, sıvı gaz akımı denetimi, kimyasal ve fiziksel süreç denetimleri) bulanık mantık bütün dünyada kullanılır hale gelmiştir [16]. Çizelge 2.1'de bulanık mantık yaklaşımının kullanıldığı birkaç örnek görülmektedir. [21].

Çizelge 2.1. Bulanık mantık denetimin endüstriyel uygulamaları [21]

ÜRÜN	**ŞİRKET**
Çamaşır makinesi	AEG, Sharp, Goldstar
Pirinç fırını	Goldstar
Fırın/Kızartıcı	Tefal
Mikrodalga fırın	Sharp
Elektrikli Traş Makinesi	Sharp
Buzdolabı	Whirlpool
Batarya şarj cihazı	Bosch
Elektrikli Süpürge	Philips, Siemens
Camcorder	Canon, Sanyo, JVC
Klima Denetimi	Ford
Isı Denetimi	NASA uzay mekiği
Kredi Kartı	GE Corporation

Bundan başka uzay araştırmaları ve havacılık endüstrisinde de bulanık mantık kullanılmaktadır. Bulanık mantık bir helikopter modelinin kontrolü; sözlü talimatla radyo kontrolü, yetersiz motor durumlarında otomatik rota girişi ve deniz kurtarmaları için insansız helikopterlerin kontrolünde de kullanılmaktadır. Helikopter uygulamasında bulanık mantık, pilotla, rüzgâr hızı ve yönünü

kapsayan uçuş durumlarıyla en iyi operasyon hareketlerini ayarlar. Yine FAA'nın yaptığı bir benzetim çalışmasında pilotlar, uçuş tecrübesi çok, az tecrübeli, öğrenci durumunda ve hiç tecrübesi olmayanlar olmak üzere dört grupta incelenmiştir. Her gruptaki 6 kişi üzerinde yapılan çalışmalarda sistemin çalışmasıyla ilgili kişiye minimum tanımlama yapılmış ve herhangi bir eğitim verilmemiştir. Sonuç olarak; bulanık mantık burada performans kriteri aşamasında kullanılmış, sistemin değişken hatalarını ve aşma (overshoot) miktarını düşürdüğü, öğrenme süresini azalttığı, kullanım için daha az çaba gerektirdiği ve tüm gruptaki elemanlar tarafından tercih edildiği görülmüştür. Bulanık mantık, ayrıca küçük bir ticari uçağın uçuş kontrol sisteminde de kullanılmıştır. Bulanık mantık denetleyicisinin kullanımıyla, uçakta kullanılan iki adet sensörün herhangi bir arıza durumunda, hangisinin arızalı olduğunu belirlemek mümkün olmuş ve daha az donanım kullanılması sağlanmıştır. Sistemin güvenliğinin arttırılmasının yanında, maliyet düşürülmüştür.

Uçaklarda güç sistemine bakım planlaması yapılması uygulamasında ise, bulanık mantık uçaktaki ekipmanların güvenlik ve güvenilirliğini arttırmak, arıza meydana gelmeden önce muhtemel arızanın belirlenmesini tesbit etmede kullanılmıştır. Havacılıkta, özellikle şirketlerin kar yapabileceği bakım faaliyetlerinde bu uygulama önemli yer tutmaktadır [16].

2.4. Bulanık Sistemler

Aristo mantığı ve bu mantık üzerine bina edilen klasik küme yaklaşımı dünyayı doğru ve yanlış, siyah ve beyaz olarak ikiye böler. Bunların dışındaki bütün seçenekler görmezlikten gelinir.

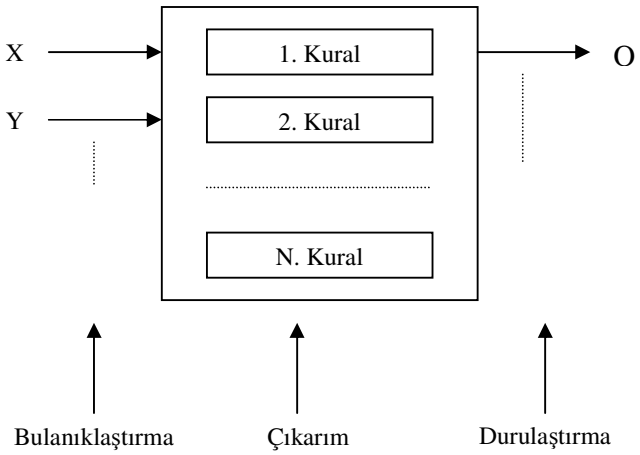
Bulanık mantık ve bulanık küme ise verdiğimiz bütün hükümleri bir derece problemi olarak görür. Bu açıdan bakıldığında bize siyahla beyaz arasında gri renklerin var olduğu gibi doğru ile yanlış arasında da başka seçeneklerin olduğunu gösterir.

Bulanık mantık kavramını basit bir şekilde anlamak için, 'biraz sıcak', 'hemen hemen doğru', 'çok hızlı' vs. cümlelerine bakılacak olursa, bu cümlelerin matematiksel açıdan bir durum ifade etmemelerine karşın, bir problemi çözme

açısından günlük hayatta kullanılan ve sıkça karşılaşılan örnekler olduğu görülür. Bulanık mantık bir insanın anlayabileceği ve çözüme ulaştırabileceği şekilde sistemlerin ya da cihazların çalışmasına izin verir. Kelime anlamı olarak, belirsiz bir durum içeriyor gibi gözükse de, matematiksel uygulamalarda oldukça kullanışlı olmaktadır.

Aristo Mantığı'nın en fazla kullanıldığı sahalar içinde mühendislik ve özellikle de doğal olayları incelemeyen mühendislik konuları gelir. Matematik, klasik fizik ve kimya ilkeleri de Aristo mantığına göre gelişmiştir. Ancak doğal olaylarla ilgilenen mühendisliklerde, durum tamamen bulanık dünya ve mantığa göre olması gerekirken, yapılan kabul, varsayım, idealleştirme, doğrusallaştırma gibi kabullerle insan aklının şartlı olarak ikili mantığa göre olay algılayabileceği seviyeye getirilir.

Bulanık bir süreç, genelde üç ayrı birimden oluşmaktadır. Bu birimler; sırası ile bulanıklaştırıcı birim, çıkarım birimi ve durulaştırıcı birimdir. Şekil 2.1'de genel bir bulanık sistem yapısı gösterilmektedir.



Şekil 2.1. Bulanık sistem yapısının genel gösterimi [16]

Bu akış düzeninde, bulanıklaştırıcı birim, bulanık işlem sisteminin ilk birimi olarak devreye girmektedir. Kesin veya geri besleme sonuçları biçiminde bu birime giren bilgiler, burada bir ölçek değişikliğine uğrayarak bulanıklaştırılmaktadır. Başka bir deyişle; bu bilgilerin her birine bir üyelik değeri atanıp, dilsel bir yapıya dönüştürülerek, buradan çıkarım birimine gönderilir.

Çıkarım birimine gelen bilgiler, depolanmış bir şekilde bulunan kural tabanına dayalı ‘eğer ... ise, ...olsun’ (if... and... then ... else) gibi bilgilerle birleştirilir.

Örneğin “1. giriş sıcak, 2. giriş normal ise, çıkış yüksektir.” gibi bir kural satırında görüldüğü gibi, kural tabanını oluşturan bilgiler, tamamen dilsel ifadelerdir. Burada sözü edilen mantıksal önermeler, problemin yapısına göre sayısal değerlerle de kurulabilmektedir. Bu durumda oluşturulacak kural satırları “1. giriş sıcak. 2. giriş normal ise, çıkış l.5’tir.” şeklinde bir kuralın benzeri olabilir.

Son adımda; problemin yapısına uygun mantıksal karar önermeleri kullanılarak elde edilen sonuçlar durulaştırıcı birime gönderilir. Durulaştırıcı birime gönderilen bulanık küme ilişkilerinde, bir ölçek değişikliği daha gerçekleştirilerek bulanık haldeki bilgilerin her biri gerçel sayılara dönüştürülür.

2.4.1. Bulanıklaştırma ve fonksiyonların oluşturulması

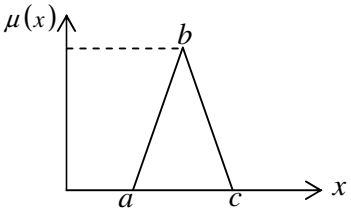
Bulanık küme kuramı, ‘belirsizlik’in bir tür biçimlenişi, formüllendirilmesidir. Bir çeşit çok değerli küme kuramıdır. Kümedeki her bir birey, iki değerli küme kuramlarında olduğu gibi ‘üye’ ya da ‘üye değil’ olarak değil, bir dereceye kadar üye olarak görülür. Bir aralıkta bulunabilecek öğelerin hepsinin, 1’e eşit üyelik derecesine sahip olacak yerde, 0 ile 1 arasında değişik değerlere sahip olması düşünülür ve $\mu_A(x)$, x elemanının ‘A’ kümesine ait olma derecesi (0 ila 1 arasında bir değer) şeklinde gösterilir. Böyle bir gösterimle birçok elemandan oluşmuş bir kümede, her eleman için o kümeye ait olma derecesi belirlenir.

Bulanık küme ve sistem işlemleri için klasik küme şeklinde belirtilen değişim aralıklarının bulanıklaştırılması gereklidir. Bulanıklaştırma sürecinde ele alınan üyelik fonksiyonları, problemin yapısına ve amacına uygun olmalıdır. Genel anlamda üyelik fonksiyonları sezgisel, matematik, geometrik ya da istatistiksel yaklaşımlara dayandırılabilmektedir.

Bulanık kümelerin gerek üyelik derecelerinin gerekse bunların tümünü temsil edebilecek üyelik fonksiyonlarının belirlenmesinde, ilk başlayanlar tarafından bile kişisel sezgi, mantık ve tecrübelerin kullanılmasına sıkça rastlanır.

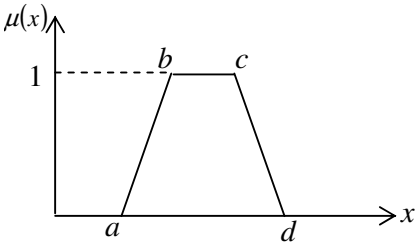
Zaten pratikte birçok sorunun üstesinden gelebilmek için bu yaklaşımlar çoğu zaman yeterlidir. Öyle olmasa bile, ilk yaklaşım olarak bu esaslara göre davranmaları faydalıdır. Üyelik fonksiyonlarının belirlenmesinde kullanılan başlıca yöntemler; a) Sezgi, b) Çıkarım c) Mertebeleme d) Açılı bulanık kümeler e) Yapay sinir ağları f) Genetik algoritmalar, g) Çıkarımcı muhakeme olarak sıralanabilir.

Bulanık üyelik fonksiyonları, olayların gerçek uzaylarını ya da dağılımlarını içerecek özellikleri sergilemelidir. Şekil 2.2'de çok sık kullanılan fonksiyonların adları, denklemleri ve grafikleri gösterilmiştir. Üyelik fonksiyonlarının; genellikle üçgen ya da yamuk şeklinde, ya da diğer uygun formlarda olmasının yanı sıra alt kümelerin de birbiri ile örtüşecek şekilde olması gerekmektedir. Örnek olması açısından sıcaklık ile ilgili durumu gösteren, üçgen üyelik fonksiyonlarındaki örtüşmeli geçişler Şekil 2.3'de gösterilmiştir.



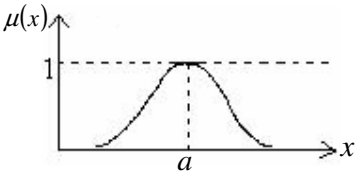
(a)

$$\mu(x) = 0, x \le a$$
$$\mu(x) = (x-a)/(b-a), a \le x \le b$$
$$\mu(x) = (c-x)/(c-b), b \le x \le c \quad (2.1)$$
$$\mu(x) = 0, c \le x$$



(b)

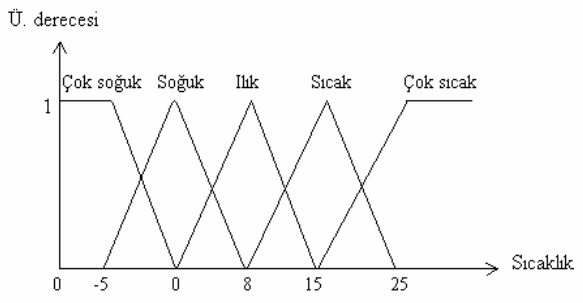
$$\mu(x) = 0, x \le a$$
$$\mu(x) = (x-a)/(b-a), a \le x \le b$$
$$\mu(x) = 1, b \le x \le c \quad (2.2)$$
$$\mu(x) = (d-x)/(d-c), c \le x \le d$$
$$\mu(x) = 0, d \le x$$



(c)

$$\mu(x) = e^{\frac{1}{2}\left(\frac{x-a}{\sigma}\right)^2} \quad (2.3)$$

Şekil 2.2. Önemli üyelik fonksiyonları (a) Üçgen üyelik fonksiyonu (b) Yamuk üyelik fonksiyonu (c) Gauss üyelik fonksiyonu



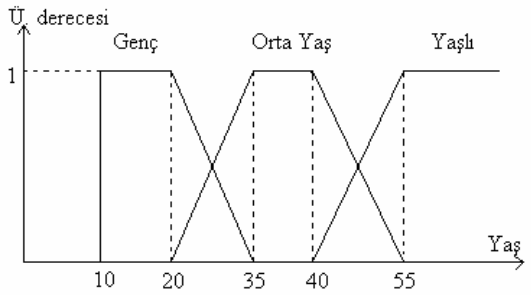
Şekil 2.3. Üyelik fonksiyonlarında örtüşmeli geçişe bir örnek

Fonksiyonları ve üyelik derecesini anlamak açısından; genç, orta yaşlı ve yaşlı insan kavramını temsil eden ve [10,90] aralığında tanımlı üç bulanık küme Şekil 2.4'de gösterilmektedir. Öncelikle bu yaş gruplarının, isteğe bağlı bir şekilde aralıklarının belirlenmesi gerekmektedir. Bu üç yaş grubunun şu şekilde fonksiyonlarını yazmak mümkündür:

$$\mu_{Genç}(x) = \begin{cases} 1 & x \le 20 \\ (30-x)/15 & 20 < x < 35 \\ 0 & x \ge 35 \end{cases}$$

$$\mu_{OrtaYaş}(x) = \begin{cases} 0 & x \le 20, x \ge 55 \\ (x-20)/15 & 20 < x < 35 \\ (55-x)/15 & 40 < x < 55 \\ 1 & 35 \le x \le 40 \end{cases} \qquad (2.4)$$

$$\mu_{Yaşlı}(x) = \begin{cases} 1 & x \ge 55 \\ (x-55)/15 & 40 < x < 55 \\ 0 & x \le 40 \end{cases}$$



Şekil 2.4. Genç, orta yaşlı ve yaşlı kavramlarını temsil eden üyelik fonksiyonları

Bu örnekteki yaşın alt kümeleri ile ilgili fonksiyonlardan, bir yaşın o kümeye ne kadar ait olduğu yani üyelik derecesi tespit edilir.

2.4.2. Çıkarım

Çıkarım birimi, bulanık sistemin çekirdek kısmıdır. Bu kısım insanın karar verme ve çıkarım yapma yeteneğinin benzeri bir yolla bulanık kavramları işler ve çıkarım yaparak gerekli denetimi belirler. Burada birçok bulanık gerçekleme yapılır. Yani insan beyninin bir benzetimi yapılmaya çalışılmaktadır.

Bulanık sistem içindeki bu benzetim bulanık kurallar yardımıyla gerçekleştirilmektedir. Genel olarak bir bulanık kural 'eğer ... ise, ... olsun' şeklinde (örneğin X değeri A ise, Y değeri B olsun) sözel girdi ve çıktı terimlerine sahip olmalıdır. 'eğer ...' bölümüne durum; '... olsun' bölümüne ise sonuç ya da karar kısmı adı verilir. 'X değeri A ise, Y değeri B olsun' örneğinde, A ve B sözel kelimelerdir. Bulanık kümelerde X ve Y değerlerinin, hangi duruma ait olduğunu gösterirler. Günlük hayatta kullanılan bazı bulanık ifadelere dayanan kurallar, örnek olması açısından aşağıda gösterilmiştir:

'Eğer basınç yüksekse, hacim küçük olsun.'

'Eğer bir domates kırmızı ise, o domates olgun bir domatestir.'

1973 yılında Zadeh, bulanık değişkenler ya da sözel ifadeler ile ilgili kavramlar ortaya koymuştur. Bu kavramlardan önemli olanlardan bir tanesi bulanık nesne kavramıdır. Nesne olarak tanımlanmış sensör girdilerine örnek olarak 'sıcaklık', 'yer değiştirme', 'hız', 'akış', 'basınç' vb. gösterilebilir. Fark değeri olan 'hata' sinyalini de aynı kategoriye sokmak mümkündür. Bulanık mantıkta ifadelerin sıfat olarak kullanımına örnek olarak ise, 'büyük pozitif' hata, 'küçük pozitif' hata, 'sıfır' hata vb. örnekler gösterilebilir. Her bir parametre için 'büyük', 'küçük' ve 'sıfır' gibi değişkenler ile ifadeler hakkında bilgi verilir. Bunlara ilaveten, 'çok büyük', 'çok küçük' vb. ifadelerle de çok doğrusal olmayan ya da istisna durumlar için kullanımda daha esneklik kazandırılabilir.

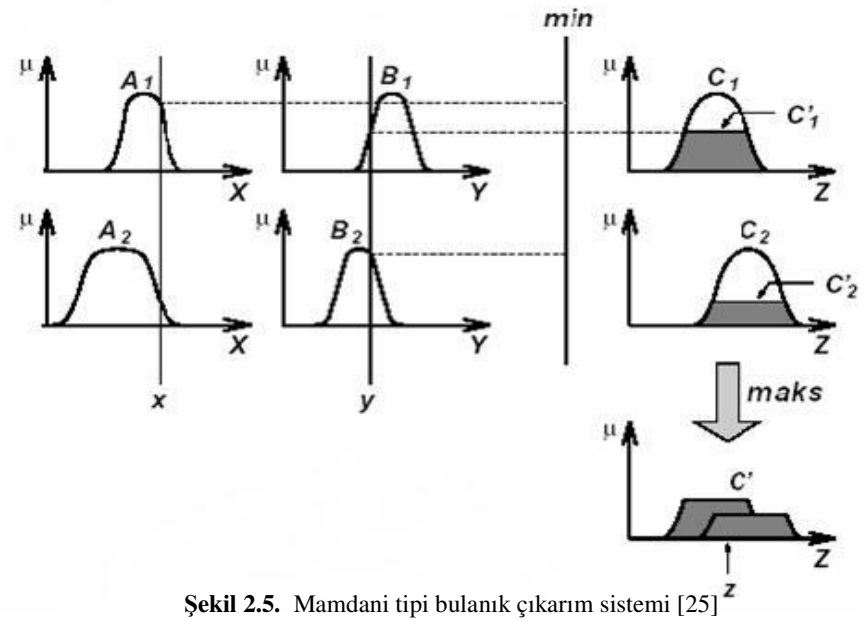
Çıkarım birimi karar verme işlemlerinde, bilgi tabanına (veri+kural tabanı) gidip, veri tabanından üyelik fonksiyonlarıyla ilgili bilgileri, kural tabanından ise

değişik giriş değerleri için tespit edilmiş olan kontrol çıkışları bilgisini alır. Bu bakımdan bilgi tabanı ve çıkarım ünitesi sürekli ilişki halindedir [16,18,24].

Bulanık çıkarım için birçok farklı yapı bulunmaktadır. Bu yapılar içinde en çok Mamdani ve Takagi – Sugeno tipi bulanık modellemeler kullanılmaktadır. Mamdani tipi bulanık model çok kolay oluşturulur ve insan davranışlarına çok uygundur. Bu nedenle çok yaygın bir kullanıma sahiptir ve diğer bulanık mantık modellerin temelini oluşturur. İlk defa bir buhar motorunun insan tecrübelerinden elde edilen sözel kontrol kuralları yardımıyla kontrolü amacıyla kullanılmıştır (Mamdani ve Assilian, 1975). Bu modelde Şekil 2.5'de görüldüğü gibi hem girdi değişkenleri hem de çıktı değişkeni kapalı formdaki üyelik fonksiyonları ile ifade edilir.

Kural 1: Eğer x = A1 VE y = B1 İse z = C1

Kural 2: Eğer x = A2 VE y = B2 İse z = C2



Şekil 2.5. Mamdani tipi bulanık çıkarım sistemi [25]

Takagi – Sugeno bulanık mantık ya da Sugeno bulanık mantık ise ilk kez 1985 yılında kullanılmaya başlanmıştır. Mamdani bulanık mantık yönteminin bir uyarlamasıdır. Girdi değişkenlerinin bulanıklaştırılması ve bulanık mantık işlemleri Mamdani bulanık modelleme ile tamamen aynıdır. İki yöntem arasındaki fark çıktı üyelik fonksiyonlarındadır. Sugeno tipi bulanık modellemede çıktı üyelik fonksiyonları sadece doğrusal ya da sabittir. Çıktı üyelik fonksiyonları

sabit olduğu zaman, sıfırıncı derece, 1. derece doğru denklemi şeklinde olduğu zaman ise birinci derece Sugeno bulanık model olarak adlandırılırlar. Bir birinci derece Sugeno bulanık model kural ifadesi aşağıdaki gibi tanımlanabilir.

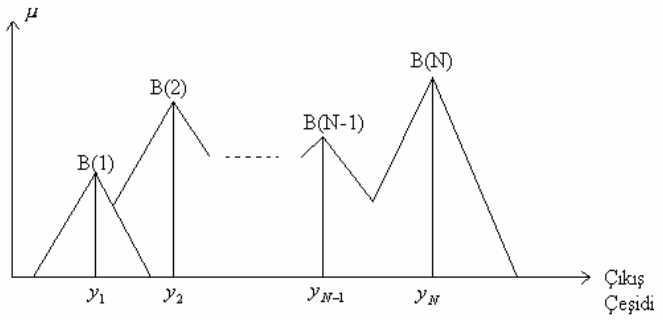
Eğer x = A ve y = B, *İse* z = f(x,y) = px+qy+r

Burada A ve B, x ve y üyelik fonksiyonları için tanımlanmış durum kısmındaki bulanık kümeler, p, q ve r ise karar parametreleridir. Böylece her bir kural için bir çıktı değeri elde edilir [25].

2.4.3. Durulaştırma

Çıkarım biriminde elde edilen bilgilerin pratik uygulamalarda, özellikle cihaz ve mühendislik plan, proje ve tasarımlarında boyutlandırmalar için kesin sayısal değerlere dönüştürülmesi gerekmektedir. Bu bilgilerin kesin sonuçlar haline dönüştürülmesi için yapılan işlemlerin tümüne birden durulaştırma işlemleri adı verilir. Durulaştırma işleminde farklı yöntemler kullanılmakta olup en sık uygulananlar ise Yükseklik Yöntemi ve Ağırlık Merkezi yöntemidir.

Yükseklik yönteminin kullanılabilmesi Şekil 2.6'da gösterilen biçimde tepeleri olan bulanık çıkarım kümelerine gerek vardır.



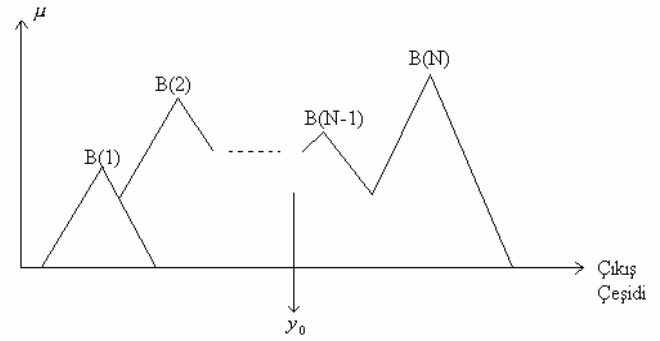
Şekil 2.6. Yükseklik yönteminin gösterimi [16]

B(1), B(2), ……B(N), 1'den N'e kadar olan kuralların çıkışlarını göstermektedir. Yükseklik yöntemine göre durulaştırma işlemi yapıldığında, y_0 çıkış değeri,

$$y_0 = \frac{\sum \mu(y_i) y_i}{\sum \mu(y_i)} \tag{2.5}$$

eşitliğinden faydalanılarak bulunur. Burada y_i, bulanıklaştırmada oluşmuş her bir fonksiyonun üyelik derecesi en büyük olan elemanlarıdır. $\mu(y_i)$ değerleri ise, bu elemanlara karşılık gelen üyelik derecelerini belirtir.

Durulaştırma işlemlerinde, yaygın olarak kullanılan işlemlerden diğeri de ağırlık merkezi yöntemidir. Şekil 2.7'de de görüldüğü gibi bu yöntemle, bulanık çıkış fonksiyonunun altında kalan bölgenin alanının ağırlık merkezi bulunur.



Şekil 2.7. Ağırlık merkezi yönteminin gösterimi [16]

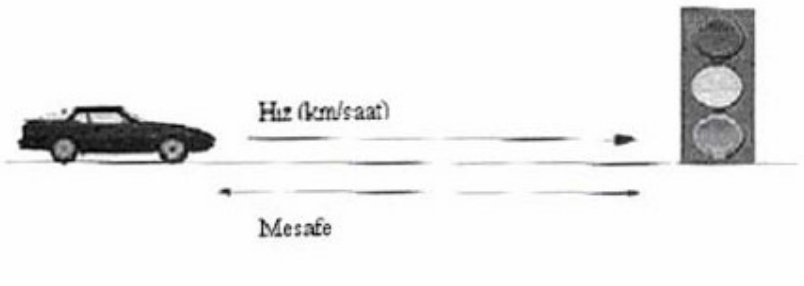
Bu yönteme göre y_0 çıkış değeri,

$$y_0 = \frac{\int \mu(y) y}{\int \mu(y)} \tag{2.6}$$

eşitliği kullanılarak hesaplanır. Bu iki yöntemin dışında ayrıca üyelik derecesi en büyük olan elemanların aritmetik ortalamasına dayanan, en büyüklerin ortası yöntemi ve ağırlıklı ortalama yöntemleri de mevcuttur.

Aşağıda bulanık sistemlerin işleyişine örnek olarak bir aracın, trafik ışığına göre hareketinin denetlenmesinde kullanılacak terimler açıklanmıştır. Şekil 2.8 incelendiğinde, bu durum için kullanılacak bilgiler, aşağıda gösterilmektedir.

- Trafik lambasının rengi
- Trafik lambası ile araç arasındaki mesafe
- Aracın hızı



Şekil 2.8. Araç ile trafik lambası arasındaki durumu gösteren örnek [16]

Bu bilgileri kullanarak, araç üzerinde yapılacak denetleme işlemleri ise şunlardır:

- Fren yapmak
- Hızı korumak
- Hızlanmak

Bu örnek için karşılaşılabilecek bazı durumları gösteren kurallar, aşağıda gösterilmektedir.

- Eğer trafik lambası kırmızı renkte ve aracın hızı yüksekse, frene basılmalıdır.
- Eğer trafik lambası kırmızı renkte, aracın hızı düşük ve mesafe uzaksa, hız korunmalıdır.
- Eğer trafik lambası sarı renkte, aracın hızı orta seviyede ve mesafe uzaksa, frene basılmalıdır.
- Eğer trafik lambası yeşil renkte, aracın hızı çok düşük seviyede ve mesafe çok yakınsa, araç hızlanmalıdır.

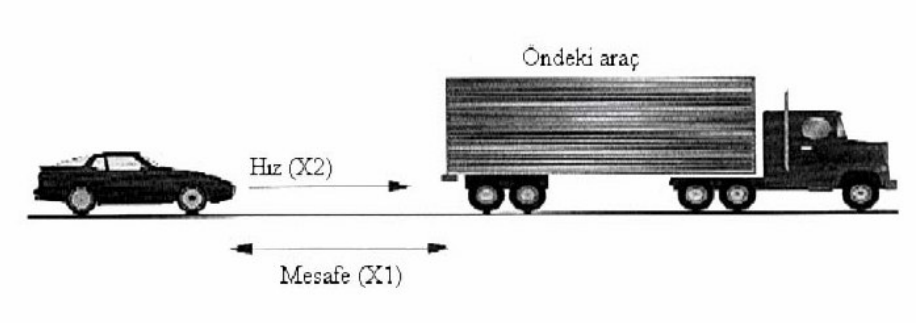
Bu örnekteki bazı ifadelerin nasıl bilgiler olduğu aşağıda gösterilmiştir:

Gözlemlenen veriler: Trafik lambasının yeşil olması; araç hızının orta seviyede olması; araç ile trafik lambası arasındaki mesafenin uzak olması

Referans alınacak terimler: Kırmızı, sarı, yeşil; çok düşük, düşük, orta, yüksek; çok yakın, yakın, uzak, çok uzak

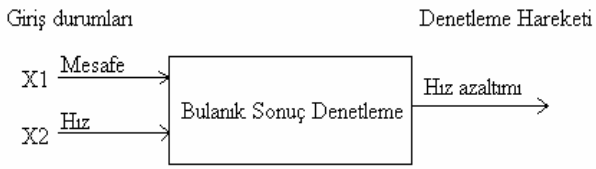
Sonuç: Hızın korunması

Şekil 2.9'daki 2. örnek ise, bulanık mantığın üç aşamasının yapıldığı (bulanıklaştırma-kurallar-durulaştırma) bir uygulamadır.



Şekil 2.9. İki araç arasındaki durumu gösteren örnek [16]

İki araç arasındaki durumu gösteren bu örnekten de anlaşılacağı gibi, burada amaç, arkadaki aracın hızını uygun şekilde denetlemektir. Girişteki durumu ve bu ifadelere karşılık gelen bir denetleme hareketini gösteren örnek blok diyagram Şekil 2.10'da gösterilmektedir.

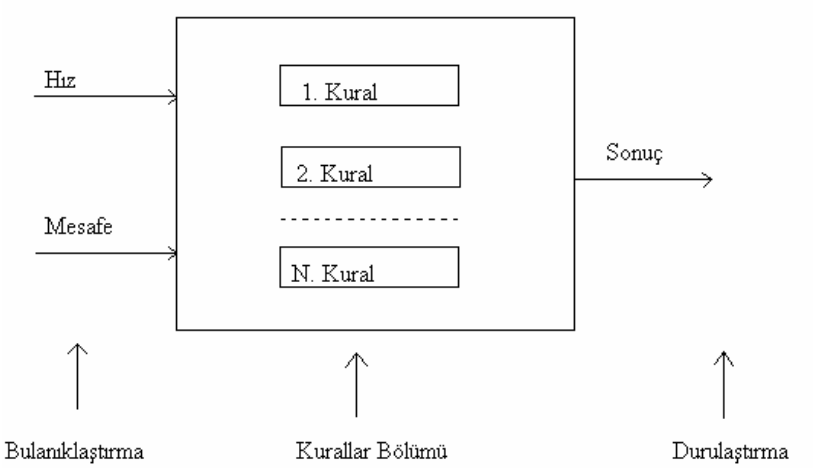


Şekil 2.10. Giriş ve denetleme hareketiyle ilgili örnek blok diyagram

Bu örnekte mesafe, hız ve hız azaltımı gibi ifadelere, sözel değişkenler adı verilir. Bu sözel değişkenlerin fiziksel limitleri, içinde bulunduğu şartlara bağlıdır (Ülkedeki hız limitleri, bölgenin coğrafik durumu, hava şartları vb.). 'Çok düşük', 'düşük', 'orta', 'yüksek', 'çok yüksek' vb. ifadeler ise bulanık kümelerdir.

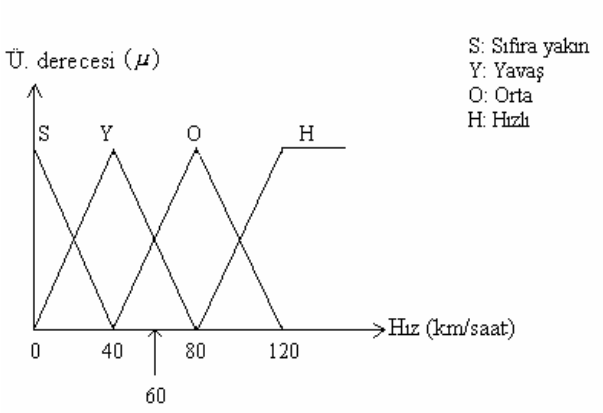
Burada, 'hızın çok yüksek olması' ifadesi bulanık bir ifadedir. 'Eğer hız yüksek ve mesafe azsa, frene basılmalıdır' ifadesi ise bir bulanık kuraldır. 'Eğer hız yüksek ve mesafe azsa' ifadesine durum, 'frene basılmalıdır' ifadesine ise karar ya da sonuç bölümü adı verilir.

Olası bütün durumları gösteren kurallardan, bulanık algoritmalar oluşturulur. Bulanık mantık sürecinin daha kolay anlaşılması amacıyla, duruma uygun bir blok diyagram Şekil 2.11'de gösterilmektedir.



Şekil 2.11. Araç örneği için bulanık mantığın blok diyagramla gösterimi [16]

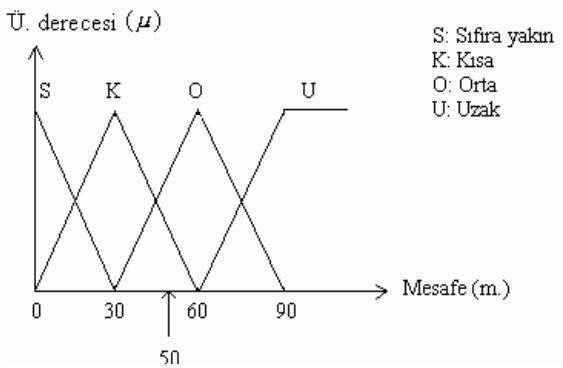
Bu örnek için ilk olarak yapılacak işlem bulanıklaştırma olacağından kolaylık sağlanması açısından üyelik fonksiyonları üçgen ve yamuk şeklinde seçilmiştir. Şekil 2.12'de hıza ait üyelik fonksiyonları, Şekil 2.13'de ise mesafeye ait üyelik fonksiyonları görülmektedir.



Şekil 2.12. Araç hızıyla ilgili hız durumlarının üçgen üyelik fonksiyonuyla gösterimi [16]

Şekil 2.12 incelendiğinde, 60 km/saat hızının üyelik derecelerini şu şekilde göstermek mümkündür:

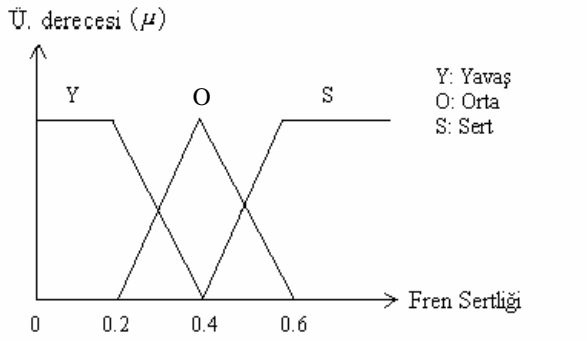
$$\begin{aligned} &\mu\,[sıfır\ \ yakın(60)] = 0 \\ &\mu\,[yavaş\,(60)] = 0.5 \\ &\mu\,[orta\,(60)] = 0.5 \\ &\mu\,[hızlı(60)] = 0 \end{aligned} \tag{2.7}$$



Şekil 2.13. Mesafe ile ilgili üyelik fonksiyonları [16]

$$\begin{aligned} &\mu\,[sıfıra\ \ yakın\,(50)] = 0 \\ &\mu\,[kısa(50)] = 0.3 \\ &\mu\,[orta\,(50)] = 0.66 \\ &\mu\,[uzak(50)] = 0 \end{aligned} \tag{2.8}$$

Şekil 2.14'de çıkışta kullanılacak olan fren sertliği ile ilgili fonksiyonlar gösterilmektedir.

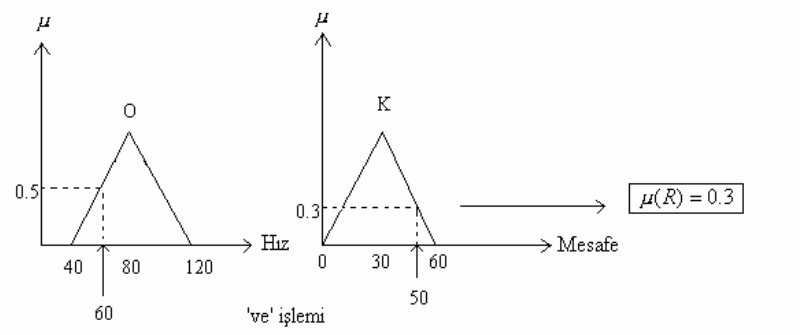


Şekil 2.14. Çıkış üyelik fonksiyonları [16]

$$\begin{aligned} &\mu\,[yavaş\,(0.4)] = 0 \\ &\mu\,[orta\,(0.4)\,] = 1 \\ &\mu\,[sert\,(0.4)\,] = 0 \end{aligned} \tag{2.9}$$

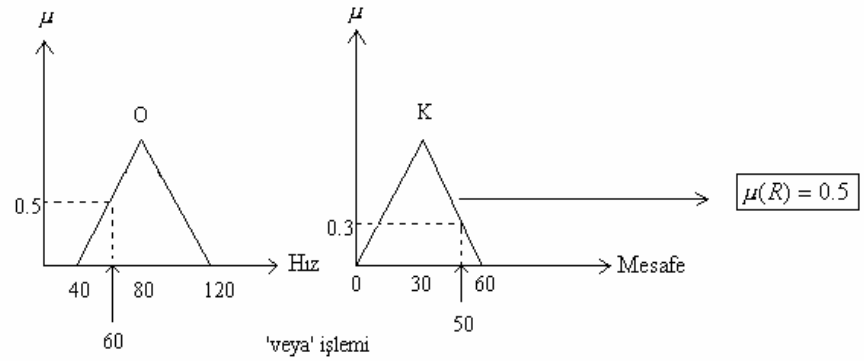
Bulanıklaştırma sürecinde en önemli işlem, uygun üyelik fonksiyonlarının seçilmesi işlemidir. Uygun üyelik fonksiyonlarının seçilmesinden sonra, ilgilenilen durumlarla ilgili işlemler, üyelik dereceleri ile yapılır.

Burada ilk kuralın ‘eğer hız orta ve mesafe kısa ise ... ’ ifadesi için işlemin nasıl yapılacağı incelenecektir. Bu durumda orta dereceli hız ve kısa mesafeyle ilgili üyelik fonksiyonlarından Şekil 2.15’te gösterilen uygun örnek değerler seçilip, bulanık mantıkta işlemler bölümünde geçen ‘ve’ işlemi uygulanacaktır. Bu durumda üyelik derecesi küçük olan hesaba katılır.



Şekil 2.15. ‘Orta hız’ ve ‘kısa mesafe’ ile ilgili örnek gösterim (‘ve’ işlemi) [16]

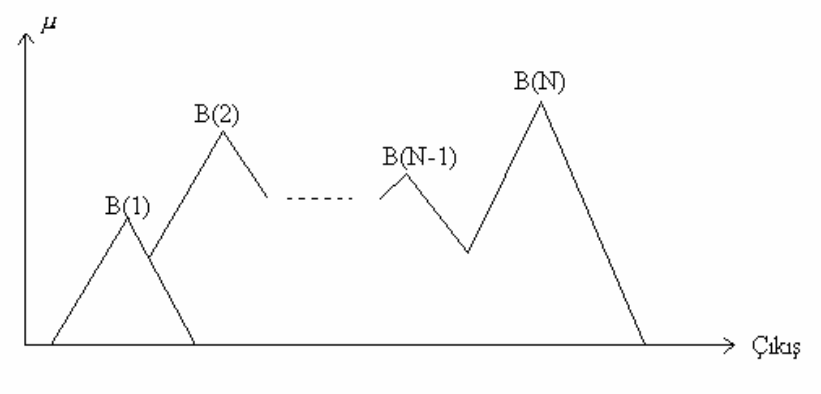
Bu örnekte kurala uygun şekilde ‘ve’ işlemi yapılmıştır. ‘Eğer hız orta veya mesafe kısa ise ... ’ ifadesi ilk kural olursa, bu durumda Şekil 2.16’daki durum gerçekleşir ve üyelik derecesinden büyük olan hesaba katılır.



Şekil 2.16. ‘Orta hız’ ve ‘kısa mesafe’ ile ilgili örnek gösterim (‘veya’ işlemi) [16]

Bütün kurallar, bu şekilde iki farklı yönteme dayandırılarak çıkış fonksiyonları elde edilir ve bu fonksiyonlar grafik üzerinde birleşim işlemine tabi

tutulur. Çıkış fonksiyonlarının birleşiminden oluşacak gösterim, örnek olması açısından Şekil 2.17'de verilmektedir.



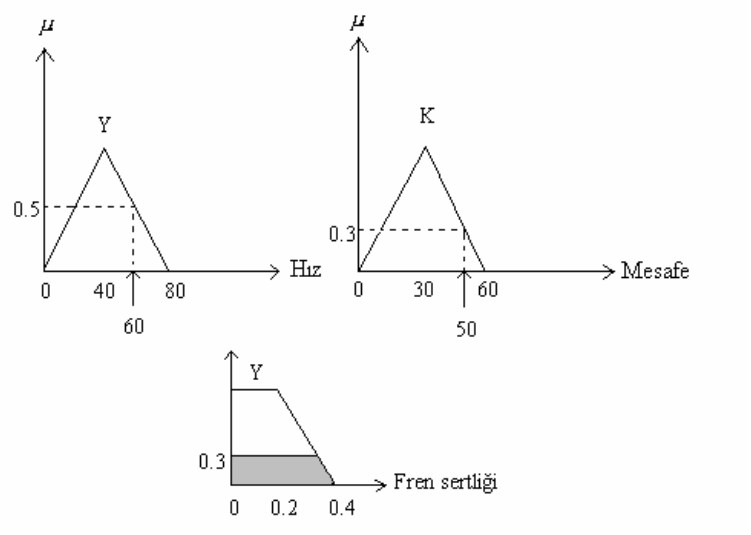
Şekil 2.17. Bulanıklaştırma işlemi sonucu ortaya çıkacak fonksiyonlar için örnek gösterim [16]

Sonuçta bu iki değişken göz önüne alınarak, kurallara uygun şekilde bulanıklaştırma işlemi yapılır. Örneğin ilk kuralın, 'eğer mesafe kısa ve hız yüksekse, frene sert bir şekilde basılmalıdır' ifadesi olduğu farz edilirse, buradaki ara işlem 've' işlemi olduğundan, buna dikkat edilerek, çıkış fonksiyonu olan fren sertliğinin derecelendirmesi yapılır. Bundan sonra gelecek kurallar içinde benzer işlemler gerçekleştirilip, her kurala uygun şekilde üyelik dereceleri elde edilir ve bunlar daha önce anlatıldığı şekilde birleştirilir.

Araç örneği ile ilgili üyelik fonksiyonları esas alınarak aşağıda bulanıklaştırma işlemleri gerçekleştirilecektir. Bu durum için, araçlar arasındaki mesafenin (X1) 50 m ve arkadaki araç hızının (X2) 60 km/saat olduğu farzedilecek, bulanıklaştırma işlemleri için ise tek kural devreye girecektir, bu kurala göre bulanıklaştırma işlemi yapılacaktır.

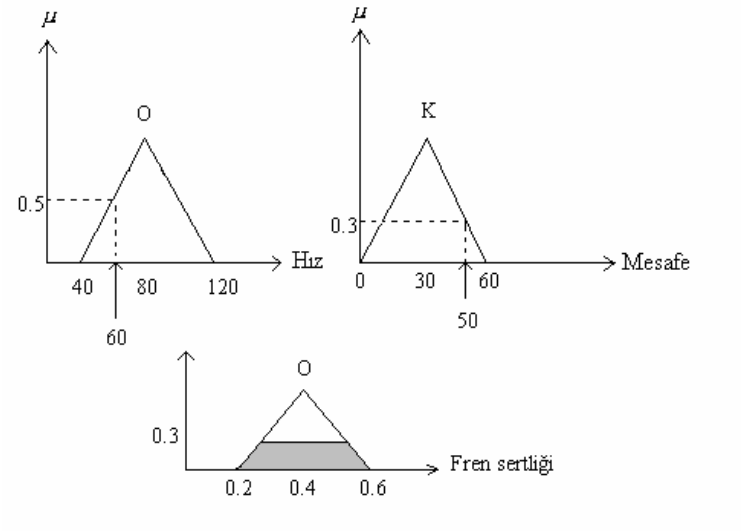
50 m mesafe 'kısa' ve 'orta' kümelerine belli üyelik dereceleriyle üyedir. Aynı şekilde 60 km/saat hız 'orta' ve 'yavaş' kümelerinin elemanıdır. Bu bilgilere dikkat edilirse, ara işlemde 've' işlemi olduğundan, çıkışta küçük olan üyelik derecesi hesaba katılarak, her bir kural çıktısı Şekil 2.18 ve Şekil 2.19'da olduğu gibi elde edilir. Bu şartlar için çıkışta kullanılacak fren sertliği kümeleri 'yavaş' ve 'orta' kümeleri ile ilgili çıkışlar olacaktır.

$$\begin{aligned} &\mu(mesafe = kısa) = 0.3 \\ &\mu(hız = yavaş) = 0.5 \\ &\mu(1.\ kural) = \wedge[\min(0.3, 0.5)] = 0.3 \end{aligned} \tag{2.10}$$



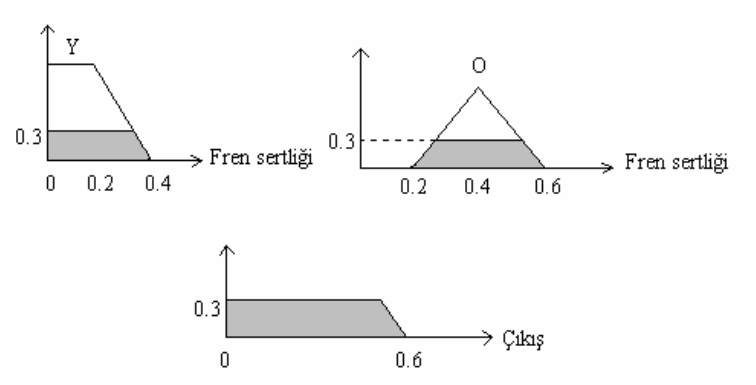
Şekil 2.18. Araç örneği için 1. kural çıktısı [16]

$$\mu(mesafe = kısa) = 0.3$$
$$\mu(hız = orta) = 0.5 \qquad (2.11)$$
$$\mu(2.\ kural) = \wedge[\min(0.3, 0.5)] = 0.3$$



Şekil 2.19. Araç örneği için 2. kural çıktısı [16]

Bulanıklaştırmadaki son işlem, bu iki kuralla ilgili fonksiyonların birleştirilmesi ve çıkış fonksiyonunun elde edilmesidir. Çıkış fonksiyonu Şekil 2.20'de gösterilmektedir.



Şekil 2.20. Çıkış fonksiyonunun gösterimi [16]

Kurallara dayalı bulanıklaştırma işleminden sonra durulaştırma işlemi tatbik edilir. Burada ağırlık merkezi yöntemine göre durulaştırma işlemi yapılırsa fren çıkış değeri; 0.281 olarak elde edilir [16].

2.5. Bulanık Kontrol

Endüstriyel bir süreç denetleyicisinin; sistemin güvenliği ve kararlılığını sağlaması, kolay, anlaşılır, tamir edilebilir ve değiştirilebilir olması, sistemin performansını istenen seviyeye çıkarması, yatırım ve işletme açısından ucuz olması istenir. Bu koşulların gerçekleştirilmesi için denetlenecek sistemin yapısının ve dinamik özelliklerinin çok iyi bilinip matematiksel modelin kurulması gerekir. Bazı sistemlerin matematiksel modellemesi mümkün olmayabilir. Sistemin değişkenleri matematiksel modelleme yapılabilecek kadar kesin olarak bilinmeyebilir veya bu değişkenler zaman içinde değişiklik gösterebilir.

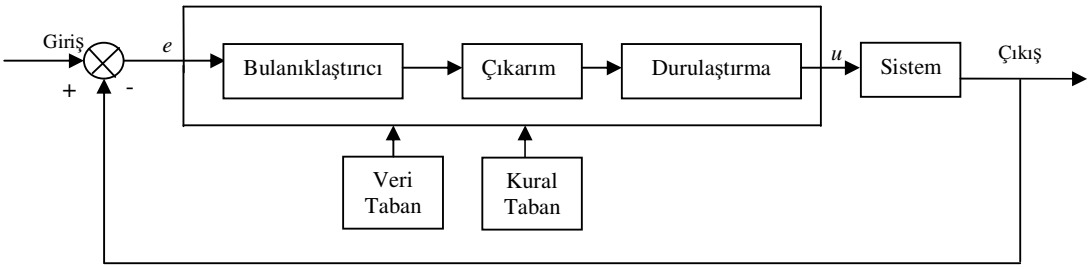
Bazı sistemlerde modelleme doğru şekilde yapılsa bile elde edilen modelin denetleyici tasarımında kullanımı karmaşık problemler ve oldukça yüksek maliyete neden olabilir. Bu nedenle, bazı denetim algoritmalarının belirsiz, doğru olmayan, iyi tanımlanmamış, zamanla değişen ve karmaşık sistemlere uygulanması mümkün olmayabilir. Bu durumda ya hiç çözüm üretilememekte ya da elde edilen denetleyicinin performansı yeterince iyi olmamaktadır.

Bu gibi durumlarda genellikle bir uzman kişinin bilgi ve deneyimlerinden yararlanılma yoluna gidilir. Uzman kişinin belirttiği az, çok, pek az, pek çok,

biraz az, biraz çok gibi günlük hayatta sıkça kullanılan dilsel niteleyiciler doğrultusunda bir denetim gerçekleştirir. Bu dilsel ifadeler doğru bir şekilde bilgisayara aktarılırsa hem uzman kişiye ihtiyaç kalmamakta hem de uzman kişiler arasındaki denetim farkı ortadan kalkmaktadır. Böylece denetim mekanizması esnek bir yapıya kavuşmaktadır. Bulanık mantığın temeli, insanın herhangi bir sistemi denetlemedeki düşünce ve sezgilerine bağlı davranışının, benzetimine dayanmaktadır. Dolayısıyla bir insan bir sistemi bulunduğu gerçek durumdan, istenilen duruma götürmek için sezgilerine ve deneyimlerine bağlı olarak bir denetim stratejisi uygulayarak amaca ulaşmaktadır. İşte bulanık denetim bu tür denetim ilişkileri üzerine kurulmuştur [21].

Bir sistem için matematiksel modelden ziyade, sistemin çalışması ile ilgili bilgilerin elde olduğu belirsizlik durumlarında, özellikle sistem transfer fonksiyonunun tanımlanmadığı fakat uygulanacak denetimin sözel olarak ifade edildiği sistemlerde bulanık kontrol kullanılmaktadır. Ayrıca transfer fonksiyonu bilinen klasik denetleyicilerin yerine sistem performansını iyileştirme amacıyla da uygulanabilir [17].

Bulanık kontrol, kontrol edilmesi gereken sürecin durum bilgisi ile kontrol işlemi arasındaki bulanık işlem algoritmasına dayanır. Şekil 2.21'de genel bir kapalı döngü bulanık kontrol sisteminin yapısı görülmektedir.



Şekil 2.21. Kapalı döngü bulanık denetleyici sistemi

Şekil 2.21'de görüldüğü gibi bulanık denetleyici bulanıklaştırıcı, çıkarım ve durulaştırma birimlerinden oluşmaktadır. Sistemde denetleyici girişindeki hata sinyali (e) bulanıklaştırma biriminde bir ölçek değişikliğine uğrayarak bulanıklaştırılmaktadır. Çıkarım birimine gelen bilgiler, bilgi tabanına (veri+kural

tabanı) dayalı, kural ağırlık tablosu şeklinde gösterilen 'eğer…ve….. ise, … olsun' (if…and…then…else) gibi kural işleme bilgileri ile birleştirilir.

Son adımda problemin yapısına uygun mantıksal karar önermeleri kullanılarak elde edilen sonuçlar durulaştırıcı birime gönderilir. Durulaştırma biriminde bilgilerin her biri gerçel sayılara dönüştürülür ve sisteme verilir. Durulaştırma işleminde yükseklik yöntemi, ağırlık merkezi yöntemi ve ağırlıklı ortalama yöntemi en çok kullanılan yöntemlerdir. [16,17].

Bulanık denetleyicilerde, klasik denetleyiciler de olduğu gibi bulanık P, bulanık PD, bulanık PI, bulanık karma (PI+PD, PD+I) denetleyiciler biçiminde farklı yapılarda tasarlanmaktadırlar. Bu denetleyicilerde kullanılan kural yapıları aşağıda görüldüğü gibi tanımlanmaktadır.

Bulanık P Denetleyici: Eğer $e = \text{A}$ ise $u = \text{C}$
Bulanık PD Denetleyici: Eğer $e = \text{A}$ ve $\dot{e} = \text{B}$ ise $u = \text{C}$
Bulanık PI Denetleyici: Eğer $e = \text{A}$ ve $\dot{e} = \text{B}$ ise $\dot{u} = \text{C}$

Bu ifadelerde A,B ve C sırası ile giriş ve çıkışa ait bulanık üyelik fonksiyonlarının değerleridir. Bölüm 2.5.1'de bu çalışmada kullanılan bulanık PD tip denetleyici ayrıntılı olarak verilmektedir.

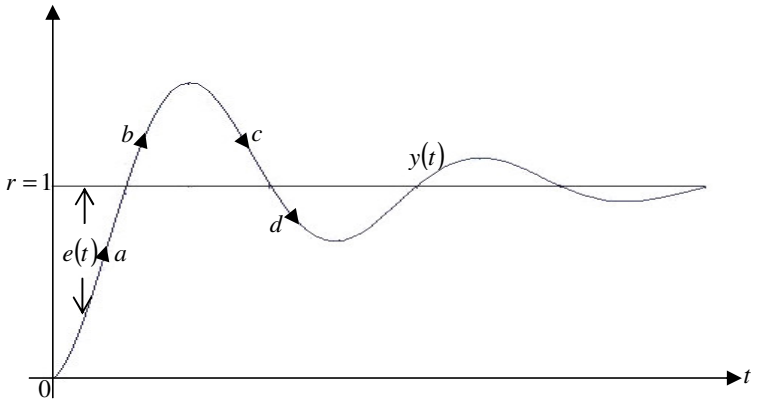
2.5.1. Bulanık PD Kontrol

İkinci mertebeden bir sistemin birim basamak referans girdiye tepkisi Şekil 2.22'de görüldüğü gibidir. Denetleyici girişindeki hata, referans girdi (r) ile çıkış ($y(t)$) arasındaki farka eşittir.

$$e(t) = r - y(t) \tag{2.12}$$

Böyle bir sistemde, bulanık denetleyici kural tabanı oluşturulurken, sadece hata sinyalinin kullanılması yeterli olmamaktadır. Örneğin $e\rangle 0$ durumu ele alınırsa, bu durumda

$$e = (r - y)\rangle 0 \quad \text{veya} \quad r\rangle y \tag{2.13}$$



Şekil 2.22. İkinci mertebeden bir sistemin birim basamak tepkisi [26]

dir ve sistem çıkışı y referans noktasının aşağısındadır. Bu durum çıkışın (y) Şekil 2.22'de görüleceği üzere a veya d noktasında (pozisyonunda) olduğunu gösterir. Bu bilgi, çıkışı (y) referans noktasına götürecek denetim bilgisini oluşturmak için yeterli değildir. Çünkü eğer çıkış (y), a noktasında ise denetleyicinin çıkışı arttıracak yönde bir kontrol oluşturması gerekmekte iken; eğer çıkış (y), d noktasında ise denetleyicinin çıkışı ters yöne çevirecek yönde bir kontrol oluşturması gerekmektedir. İlk durumda (a) denetleyicinin önceki durumunu devam ettirmesi gerekirken ikinci durumda (d) önceki durumunun tersi yönde bir denetim bilgisi oluşturması gerekmektedir. Dolayısıyla bu iki durumu birbirinden ayıracak farklı bir bilgiye ihtiyaç doğmaktadır.

Matematikten bilineceği üzere eğer bir eğri artan yönde ise türevi pozitif, azalan yönde ise türevi negatiftir. Buradan hata sinyalinin değişimi ($\dot{e}$), a ve d, benzer şekilde b ve c noktalarını birbirinden ayırt etmek için yeterli bilgiyi verecektir. Dolayısıyla uygun bulanıklaştırma ve kural işleme bilgilerini oluşturabilmek için denetleyici girişine hata ile birlikte hatanın değişimini uygulamak gerekecektir. Bu tip denetleyici bulanık PD denetleyici adını almaktadır.

Bulanık PD tip kontrol bulanık denetleyiciler içinde en fazla kullanılan denetleyici tipidir. Bu tip denetleyicide yukarıda da bahsedildiği üzere bulanıklaştırma ve kural işleme bilgileri oluşturulurken hata (e) ve hatanın değişimi ($\dot{e}$) bilgileri kullanılmaktadır. Bu şekilde Sugeno tipi modellemeye göre

oluşturulmuş ve çalışmanın uygulama bölümünde de kullanılan örnek bir kural ağırlık çizelgesi Çizelge 2.2'de verilmektedir.

Çizelge 2.2. Kural ağırlık değer çizelgesi

		$\dot{e}$				
e	u	HDNCK	HDNK	HDY	HDPB	HDPCB
	HNCK	A_1	A_2	A_3	A_4	A_5
	HNK	A_6	A_7	A_8	A_9	A_{10}
	HY	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}
	HPB	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}
	HPCB	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}

Burada; A_1, A_2……..A_n değerleri gerçel sayılardır ve kural ağırlık değerlerini göstermektedir. HNCK, HNK, HY, HPB, HPCB, HNCK, HNK, HY, HPB ve HPCB ise hata ve hatanın değişimine ait üyelik fonksiyonlarını göstermektedir (H:hata; D:değişim; NCK: negatif çok küçük; NK: negatif küçük, Y:yok; PB: pozitif büyük; PCB: pozitif çok büyük). Bu tabloya göre 25 adet kural oluşmaktadır. Bu kurallar;

Eğer e=HNCK ve $\dot{e}$=HDNCK ise u=A_1
Eğer e=HNCK ve $\dot{e}$=HDNK ise u=A_2

⋮

Eğer e=HPCB ve $\dot{e}$=HDPCB ise u=A_{25}

biçimindedir.

İlk olarak hata ve hatanın değişimine ait değerler bulanıklaştırılarak çıkarım biriminden geçirilir. Durulaştırma biriminde ağırlıklı ortalama yöntemine göre, yorumlanan her bir kurala ait en küçük üyelik derece değeri aksiyon ağırlık değeri ile çarpılır ve bulunan sonuçlar toplanır. Bu değer, kuralların toplam üyelik değerlerine bölünerek denetleyici çıkışı bulunur [16,17,24,26].

3. GENETİK ALGORİTMALAR

3.1. Giriş

Günlük olayların ve araştırmaların en önemli sorularından birisi yapılan çaba ve çalışmaların en iyileme (Eİ) ile sonuçlandırılabilir olup olmayacağıdır. Bunun nedeni elde bulunan imkânların (malzeme, zaman, mekân) kısıtlı olması dolayısı ile hizmetin Eİ biçiminde verilmesi ile önemli kazançların sağlanmasıdır. Bu, endüstride önemli olduğu kadar doğal kaynakların (su, enerji, gıda, vb.) en iyi üretimde kullanılması konularında da aynıdır [27].

Karmaşık bir sistemin en iyilemesinde karşılaşılan iki problem vardır. Birincisi, en iyileme algoritması sistem için uygun olmalıdır. İkincisi ise en iyileme algoritmasının çeşitli parametreleri verimliliği artırmak için ayarlanabilmelidir. Buna göre, karmaşık bir sistemin geniş bir aralıkta en iyilemesinin yapılabilmesi için kullanılan en uygun yöntemlerden birisi genetik algoritmadır [1].

Genetik algoritmalar (GA), evrim teorisinden esinlenerek türetilen hesaplama modelleridir. Makine öğrenmesi konusunda çalışan Holland (1975), canlılardaki genetik işlemleri sanal ortamda gerçekleştirmeye çalışarak bu işlemlerin etkinliğini açıklamıştır. Başlangıçta pratik bir yararı olmadığı düşünülen GA'lara olan ilgi, Holland'ın öğrencisi olan Goldberg'in yaptığı doktora teziyle National Science Foundation tarafından verilen Genç araştırmacı ödülünü alması ve dört yıl sonrada klasik eserini yayınlamasıyla çoğalmıştır. (Goldberg 1989).

Bu algoritmalarda, belirlenen bir problemin potansiyel çözümleri, ikili (binary) ya da ikili olmayan sistemlere dayalı veri yapısında basit diziler olarak şifrelenir ve kritik bilgileri saklamak için bu dizilere bir takım işlemler uygulanır. GA yöntemi, evrim teorisi esaslarına göre çalışarak verilen bir sorun için en iyi çözüm veya çözümleri arayarak bulmaya yarar. Bu esaslar ortama en fazla uyum sağlayan canlıların hayata devam etmesi ve uyum sağlayamayanların da elenmesi olarak algılanmalıdır. GA'lar bu iki kuralı bir arada kullanarak en iyiyi aramayı hedef edinen bir Eİ yöntemidir [27].

Oldukça geniş uygulama alanına sahip olan genetik algoritmalar daha çok bir en iyileme fonksiyonu olarak görülür. Genel olarak, en iyileme (en küçükleme-en büyükleme) sorunlarının çözümü için uygun olan GA kullanımı, diğer yöntemlerle kıyas edildiğinde çok daha iyi çözümlere en kısa zamanda sayısal örgünlük ve rasgele düzen içinde gidebilmektedir. Genetik algoritmalar aynı zamanda oldukça geniş bir araştırma uzayında, bölgesel araştırma tekniklerinin (gradiyent yöntemi vb.) uyguladığı birçok ihmali eleyen global en iyileme teknikleri olarak da görülebilir.

Genetik algoritma, mühendislik, bilim, ekonomi gibi çok değişik alanlardaki problemler için gürbüz bir en iyileme aracı olarak son yıllarda büyük bir önem kazanmıştır. Genetik algoritmanın uygulama alanlarından bazıları, haberleşme şebekeleri tasarımı, elektronik devre tasarımı, gaz boruları şebeke en iyilemesi, görüntü ve ses tanıma, veri tabanı sorgulama en iyilemesi, uçak tasarımı, fiziksel sistemlerin kontrolü, gezgin satıcı problemlerinin çözümü, ulaşım problemleri ve optimal kontrol problemleridir [1, 24].

3.2. Genetik Algoritmaların Diğer En İyileme Tekniklerine Göre Farklılıkları

Genetik algoritmalar gürbüzlük araştırmalarında diğer geleneksel en iyileme yöntemlerine göre daha üstündür. Genetik algoritmaların faklılığı dört ana başlık altında toplanabilir:

1) Genetik algoritmalar, parametrelerin kendisiyle değil parametre setini kodlayarak çalışır:

Genetik algoritmalar, sonlu bir alfabe üzerinden sonlu uzunlukta diziler kodlamak için, en iyileme probleminin doğal parametre setini kullanır. Örneğin, $f(x) = x^2$ fonksiyonunun [0,31] aralığında en büyüklemesi yapılmak istendiğinde, diğer en iyileme yöntemleri x parametresini küçük küçük değiştirerek amaç fonksiyon değerinin en yüksek olduğu yeri bulmaya çalışırlar. Genetik algoritmalarla en iyileme işleminin yürütülmesinde ise ilk adım, x parametresinin sonlu uzunlukta dizi olarak değişik şekillerde kodlanması ve kodlanan parametre ile çalışılarak en iyilemenin sağlanmasıdır. Karar değişkenlerinin karakterleri topluca karar uzayında bir noktayı temsil eder. Pek çok durumda ikili (binary) kodlama

kullanılır. Fakat genetik algoritma için bu bir zorunluluk değildir. Gerçel sayı kodlama, ağaç yapılı kodlama (tree coding) gibi farklı kodlama biçimleri de kullanılabilir. Parametrelerin kodlanması sonucunda, diğer yöntemleri sınırlayan birtakım özelliklerde de büyük ölçüde serbestlik sağlanmış olur (süreklilik, türevin varlığı, vb.)

2) Genetik algoritmalar tek bir noktada değil bir noktalar kümesinde en iyileme araştırması yapar:

Birçok en iyileme probleminde, tanım aralığı içindeki tek bir noktadan hareketle bazı geçiş kurallarına göre bir sonraki inceleme noktası bulunur. Bu, noktadan noktaya yönelme yöntemi, çok sayıda tepe noktası bulunan araştırma uzayı için risklidir. Çünkü bölgesel tepe değerine yaklaşma hatası oluşabilir. Genetik algoritmalarda, çok sayıda noktalardan oluşan bir veri tabanı ile (dizilerin nüfusu) çalışıldığından aynı anda pek çok tepe noktasına atlanabilir. Böylece noktadan noktaya geçme yöntemindeki yanlış tepe noktasının bulunma olasılığı azalır. Sonuç olarak, genetik algoritma dizilerden oluşan bir nüfus ile başlar ve bu dizilerden daha başarılı nüfuslar üretir. Bu nüfus, problemin bütün olası çözümlerini temsil eden uzayı oluşturur. Başlangıç nüfusu genellikle rasgele üretilen bireyleri içerir.

3) Genetik algoritmalar, türevler ya da diğer yardımcı bilgiler değil sadece amaç fonksiyon bilgisini kullanır:

Çoğu araştırma tekniklerinin, doğru bir şekilde çalışması için yardımcı bazı bilgilere ihtiyaçları vardır. Örneğin, gradient tekniklerinde, tepe değerine doğru yükselmenin olup olmadığını anlamak için, nümerik ya da analitik olarak hesaplanan türev bilgisi gereklidir. Ayrıca, farklı araştırma problemleri için de çok farklı yardımcı bilgilere ihtiyaç olabilir. Bunun aksine, genetik algoritmalarda hiçbir yardımcı bilgiye ihtiyaç yoktur. Verimli bir araştırma yapmak için gerekli olan, her bir dizinin değerlendirileceği bir amaç fonksiyonudur. En iyileme sırasında problemle ilgili özel birtakım bilgilerin kullanılmaması, genetik algoritmaların performansını yükseltmede son derece etkilidir.

4) Genetik algoritmalar kesin bilinen kuralları değil olasılığa dayalı (stokastik) kuralları kullanır:

Genetik algoritmalar araştırmaya yön vermek için, birçok en iyileme tekniğinin aksine, olasılığa dayalı geçiş kuralları kullanır. Bunun sonucunda araştırmanın, araştırma uzayının hangi bölgesine doğru yöneleceğine karar vermek için rasgele seçim tekniği kullanılır.

Bahsedilen bu dört farklılığın bir arada bulunması, genetik algoritmaların gürbüzlüğüne ve sonuca ulaşma üstünlüğüne olumlu yönde katkıda bulunur.

3.3. Genetik Algoritma Terminolojisi

Genetik algoritmalar, hem doğal genetikler hem de bilgisayar alanında kullanıldığı için genetik algoritma literatüründe kullanılan terminoloji, doğal ve yapay ortamdaki terimlerin karışımından oluşmaktadır. Bu nedenle, genetik algoritmalarla çalışan araştırmacıların kullanacağı temel bazı terimlere ihtiyaç vardır. Bilgisayar ortamında tanımlanan diziler, alfabeler, dizi pozisyonları ve buna benzer bazı terimlerin doğal genetiklerde kullanılan karşılıkları çizelge 3.1'de görülmektedir. Örneğin; yapay genetik sistemlerdeki diziler, biyolojik sistemlerdeki kromozomlara benzemektedir. Doğal sistemlerde, bir ya da daha fazla kromozom, bir organizmanın yapısı ve çalışması için toplam genetik bilgileri oluşturacak şekilde birleşir. Bu toplam genetik paket genotip olarak adlandırılır. Yapay genetik sistemlerde ise dizilerin toplamı genetik yapı paketi olarak adlandırılır. Doğal sistemlerde, toplam genetik paketin içinde bulunduğu şartlara göre birbirini etkilemesiyle şekillenen organizma fenotip (phenotip) olarak tanımlanır. Yapay genetik sistemlerde ise yapılar, belli bir parametre setini, olası bir çözümü ya da çözüm uzayında bir noktayı düzenlemek için şifreli durumu çözer. Bir yapay genetik sistem tasarımcısı hem nümerik hem de nümerik olmayan parametreleri kodlamak için çeşitli alternatiflere sahiptir.

Doğal terminolojide kromozomlar, belli sayıda değer alabilen genlerin birleşiminden oluşmaktadır. Bir gen'in pozisyonu onun gen olarak fonksiyonundan ayrı tanımlanır. Örneğin; bir hayvanın göz rengi geni ele alındığında bu genin pozisyonu ya da yeri 10, bu kodun değeri de mavi göze karşılık gelebilir. Yapay genetik araştırmalarda ise diziler, farklı değerler alan

özellikler ya da karakterlerden meydana gelmektedir. Özellikler, dizi üzerinde farklı pozisyonlara yerleştirilebilir.

Çizelge 3.1. Doğal ve GA terminoloji karşılaştırması [28]

Doğal terminoloji	Genetik algoritma
kromozom	dizi
gen	özellik, karakter veya dedektör
allele	özellik değeri
konum	dizi pozisyonu
genotip	yapı
fenotip	parametre seti, alternatif çözüm, kodlanmış yapı

Belli bir karakter ve onun pozisyonu arasındaki farkı ayırt etmek gerekirse; bir dizi içinde bir bitin pozisyonu, nüfusun her yerinde ve her zaman o bitin nasıl çözüldüğü ile bulunabilir. Örneğin; ikili sayı sistemine göre 10000 dizisi, işaretsiz tamsayı olarak onlu sayı sistemine göre 16 olarak çözülebilir. Burada 1, onlu sayı sisteminde 16'nın yerine geçmektedir [1,28,29].

Basit bir genetik algoritma (Simple Genetic Algorithm), problem en iyilemesi için son derece uygun ve güçlü bir yöntemdir. Literatürde genetik algoritma tekniği içerisinde, SGA (Simple Genetic Algorithm)'dan başka, Mikro Genetik Algoritma (μGA), Kararlı Durum Genetik Algoritma (KDGA), Hiyerarşik Genetik Algoritma (HGA) ve Dağınık Genetik Algoritma (mGA) gibi farklı modeller ve paralel çalışan genetik algoritma modelleri tanımlanmıştır.

Genetik algoritma, en uygun (optimal) çözümü bulmak için birden çok noktadan aramaya başladığı için bu arama noktalarının başlangıç değerlerini oluşturmak önemlidir. İlk popülasyon genellikle rastlantısal olarak oluşturulur. Fakat yapılan bazı çalışmalarda ilk popülasyonun daha önceden elde edilen bir bilgiye veya sezgiye (heuristic) bağlı olarak üretildiği görülmektedir [26]. Varılması istenilen Eİ noktasının yaklaşık konumu hakkında ipuçları bulunuyorsa dizi değerlerinin böyle bir çözüm noktası etrafında rasgele ama daha dar bir karar uzayından seçilmesi uygundur.

3.4. Basit Genetik Algoritma

Birçok problemin çözümünde iyi sonuçlar veren basit bir genetik algoritma (SGA); kopyalama (reproduction), çaprazlama (crossover) ve mutasyon (mutation) olmak üzere üç genetik işlemin birleşiminden oluşur. Bir jenerasyon boyunca yürütülen bu işlemler en büyükleme ya da en küçükleme probleminde jenerasyonlarda elde edilen optimal değerler arasındaki fark sıfırlandığında ya da belli bir değere yakınsadığında sona erdirilir. Ayrıca bir genetik algoritma, programın başında belirlenen bir jenerasyon sayısı kadar tekrarlanıp bitirilebilir. Bu sayının yeterince büyük olması sonuçta elde edilecek değerin, fonksiyonun optimal çözümü olma şansını artırır.

3.4.1. Kopyalama

Kopyalama, nüfus içindeki çözüm seçenekleri olan diziler arasında bir sonraki aşamada daha da iyiye gidebilecek olanların seçilmesi işlemidir. Kopyalama diziyle özdeş dizi üreten bir operatördür. Başka bir deyişle her bir dizinin amaç fonksiyon değerine (f) göre, gelecek jenerasyona kopyalanmasını sağlayan bir işlemdir. (f) fonksiyonu en büyükleme yapılmak istenen uygunluk, yararlılık ya da iyiliğin bir ölçüsü olarak düşünülebilir. Uygunluk değerlerine göre dizilerin kopyalanması, yüksek değere sahip olan dizilerin gelecek jenerasyona bir ya da daha fazla ürün olarak katkıda bulunmasıdır. Bu işlem doğal seçimin yapay bir versiyonudur. Darwin'e göre de, varlıklar arasında en iyilerin yaşama şansı her zaman daha yüksektir. Klasik yöntemlerin tümünde her an yegâne diyebileceğimiz bir tane en iyi çözüm olmasına karşılık, GA'larda çözüm olmaya aday bir nüfus vardır. Bunlar arasında sadece bir tanesi en iyidir. Diğerleri de bulundukları karar değişkeni bölgesinde en iyi olmaya adaydır. Yaygın olarak kullanılan kopyalama teknikleri, bütünüyle yer değiştirme, jenerasyon aralığı ve kararlı durum yöntemleridir.

Bütünüyle yer değiştirme yönteminde, eski jenerasyondaki bütün diziler yeni jenerasyondaki dizilerle yer değiştirir. Yeni dizi seçilince, genetik operatörler

diziyi şekillendirmeye başlar. İşlemler, dizi sayısı popülasyon büyüklüğüne eşit oluncaya kadar devam eder. Bütünüyle yer değiştirme en eski kopyalama tekniklerinden biridir ve genetik algoritma uygulamalarının çoğu bu tekniği kullanmaktadır.

Jenerasyon aralığı (λ) yöntemi, bütünüyle yer değiştirme yöntemi ile oldukça benzerdir. Aralarındaki fark, eski jenerasyondaki dizilerin belli bir yüzdesinin yeni jenerasyona taşınmasıdır. Örnek olarak λ=0.5'lik jenerasyon aralığı, eski jenerasyondaki bireylerin %50'sinin yeni jenerasyona taşındığı anlamına gelir. %100'lük bir jenerasyon aralığı, bütünüyle yer değiştirme yöntemini oluşturur. Yer değiştirilecek diziler rasgele seçilebilir veya bir tür elistik model kullanılabilir. Elistik model'de jenerasyondaki en yüksek uygunluğa sahip olan dizinin bir sonraki jenerasyonda devamı sağlanır. Fakat jenerasyon aralığı yöntemi, dizilerin bir önceki jenerasyondan daha yüksek uygunluk değerlerine sahip olmasını garanti etmez. Böylece özellikle küçük jenerasyon aralığı değerlerinde jenerasyon aralığı yöntemi, bütünüyle yer değiştirme yönteminden oldukça farklı sonuçlar verir.

Kararlı durum kopyalama yöntemi ise kararlı durum genetik algoritmada (KDGA) kullanılan kopyalama yöntemidir. Bu kopyalama yönteminde, her jenerasyondaki popülasyona ait dizilerden sadece 1 adedi silinir ve yeni 1 adet dizi üretilir. Böylece her jenerasyonda sadece 1 adet dizi üretilmiş olur.

Seçim yöntemleri ise genel olarak en iyi olan yaşar prensibine göre oluşturulur. Amaç, yeni jenerasyonda daha yüksek uygunluğa sahip bireylerin sayısını arttırmaktır. Genetik algoritmada kullanılan seçim yöntemlerini aşağıdaki gibi 3 ana grupta toplayabiliriz.

1) Uygunluk değeri orantılı seçim yöntemleri (Fitness proportionate selection)
2) Turnuva seçim yöntemi (Tournament selection)
3) Sıralı seçim yöntemi (Rank selection)

Uygunluk değeri orantılı seçim yöntemlerinin genel özelliği, nüfusu oluşturmak için kullanılacak dizileri seçmek için, dizilerin uygunluk değerlerine

bakılmasıdır. Bu yöntemlerden, rulet çarkı seçim yönteminde nüfus içindeki her dizi, rulet çarkı üzerinde uygunluk değeri ile orantılı olacak kadar yer kaplar. Örneğin, $f(x)=x^2$ fonksiyonunun [0,31] tam sayı aralığında en büyükleme probleminin genetik algoritma ile çözümünün elle simülasyonu yapılırsa algoritma, belirlenen aralıktan rasgele seçilen x değişken değerlerinden başlangıç nüfusunun oluşturulmasıyla başlar. Başlangıç nüfusu sayısı $n=4$ olarak verildiğinde, verilen aralıktan rasgele seçilen başlangıç nüfusu,

0 1 1 0 1
1 1 0 0 0
0 1 0 0 0
1 0 0 1 1

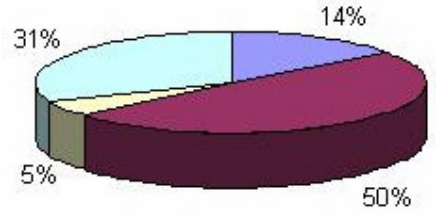
olabilir. Bu dört diziden oluşan örnek nüfusun amaç ya da uygunluk fonksiyonu, (f) , değerleri ve nüfusun toplam uygunluğuna göre her dizinin yüzdesi, Çizelge 3.2.'de gösterilmektedir.

Çizelge 3.2. Örnek problemin dizileri ve uygunluk değerleri [1]

No.	Diziler	Uygunluk	% Toplam
1	0 1 1 0 1	169	14
2	1 1 0 0 0	576	50
3	0 1 0 0 0	64	5
4	1 0 0 1 1	361	31
Toplam		1170	100.0

Örnek problemin nüfusunu oluşturan dört dizinin toplam uygunluk değeri 1170 olarak bulunmaktadır. Ağırlıklı rulet çarkına göre bu jenerasyonun kopyalanması Şekil 3.1'de gösterilmektedir. Dizilerin, gelecek jenerasyondaki kopyalanmasını bulmak için, ağırlıklı rulet çarkının dört kez döndürüldüğü varsayılır.

Örnek problem için, 1 numaralı dizinin uygunluk değeri 169'dur ve bu değerin toplam uygunluk değerine göre yüzdesi 14'dür. Sonuç olarak, 1 nolu dizi rulet çarkının yüzde 14'lük kısmını kaplar. Rulet çarkı bir defa döndürüldüğünde 1 nolu dizinin gelme olasılığı 0.14'dür. Uygunluk değeri yüksek olan diziler rulet çarkı üzerinde daha büyük alan oluşturacağından, rulet çarkının bir defa döndürülmesinde bu dizilerin gelme olasılığı daha yüksek dolayısıyla bir sonraki jenerasyonda görünmesi şansı da daha fazladır.



Şekil 3.1. Her bir dizinin uygunluk değerine göre rulet çarkında kapladığı alan

Turnuva seçim yönteminde, diziler rasgele olarak gruplanır ve gruptaki diziler aralarında seçim işlemi için rekabete sokulur. Grup içinde, en yüksek uygunluk değerine sahip olan dizi, bir sonraki jenerasyondaki nüfusu oluşturmak için dizi olarak seçilmiş olur. Bu işlem, toplam popülasyon sayısı oluşturuluncaya kadar devam eder. Bu seçim yönteminde grup büyüklüğü önemlidir ve seçim yönteminin performansını önemli ölçüde etkiler. Bazı uygulamalarda grup büyüklüğü 2 olarak seçilirken bazı uygulamalarda çok daha büyük gruplar oluşturulur. Turnuva seçim yöntemi küçük popülasyon kullanan uygulamalarda uygunluk değeri orantılı seçim yöntemlerinden daha iyi sonuç verir. Bu seçim yönteminde bireyin seçilme olasılığı eşitlik (3.1) ile verilir [24] .

$$p_i = \frac{1}{N^q}\left((N-i+1)^q - (N-i)^q\right) \tag{3.1}$$

Eşitlik (3.1)'de; N popülasyon büyüklüğü, p_i $i.$ bireyin seçilme olasılığını ve q turnuva büyüklüğünü temsil eder.

Sıralı seçim yöntemi ise Baker tarafından gerçekleştirilmiştir [24]. Bu yöntemde diziler uygunluk değerlerine göre büyükten küçüğe sıralanır. Dizinin seçim şansı uygunluk değerinden ziyade oluşturulan liste içindeki yerine bağlıdır.

Oluşturulan liste içindeki dizilere kopya sayısı atanır. Baker'in geliştirdiği Doğrusal Atama Fonksiyonunu kullanan yöntemin algoritması aşağıda verilmiştir.

1) Popülasyondaki her dizi uygunluk değerine göre küçükten büyüğe doğru sıralanır.

2) Kullanıcı son sıradaki bireyin beklenen seçim değerini hesaplar (Max).

3) t adımda, i inci bireyin beklenen seçim şansı Eşitlik (3.2)'den bulunur.

4) İşlem N popülasyon büyüklüğüne ulaşıncaya kadar devam eder.

$$Min + (Max - Min)((Sira_{(i,t)} - 1)N - 1) \tag{3.2}$$

Eşitlik (3.2)'de Min =1.bireyin seçilme şansıdır ve $(Min = 2 - Max)$ olarak seçilir. $Max = (1 \le Max \le 2)$ olarak seçilir. Baker bu değeri 1.1 olarak seçmiştir.

Bir başlangıç nüfusundan başarılı nüfuslar üreten basit genetik algoritmanın performans ve verimliliğini etkileyen en önemli faktörlerden birisi nüfusun boyutudur. Algoritmanın başında belirlenen nüfus sayısının az olması genetik algoritmaları genellikle başarısız yapar. Çünkü nüfus sayısının az olması problemin çözümü için araştırma uzayından verimsiz örnek sayısı sağlar. Nüfus sayısının çok olması araştırma uzayında daha çok ve etkili örnek seçilmesi açısından uygundur. Böylece genetik algoritmalar daha çok bilgi içeren araştırmalar yapabilir ve alt optimal çözümlere hatalı yaklaşımlar önlenir. Diğer taraftan, büyük nüfus sayılarında her jenerasyonda daha çok değerlendirme yapıldığından olası sonuca yaklaşım daha yavaş olmaktadır.

3.4.2. Çaprazlama

Kopyalama işleminden sonra her jenerasyonda, mevcut nüfus dışında, aynı uzay içinde farklı noktalara ulaşmak ya da araştırma uzayının diğer noktalarını da incelemek için yeniden düzenleyici genetik işlemler uygulanarak yeni nüfus içinde bazı değişimler ortaya çıkarılır. Çaprazlama, dizilerin karakterleri birbirleriyle değiştirilerek farklı dizilerin elde edilmesi işlemidir. Çaprazlama

işlemi de farklı yöntemlerle (tek kesimli, çift kesimli, çok kesimli, tekdüze, tersleme) gerçekleştirilebilmektedir.

Genetik algoritmada geleneksel olarak, çaprazlama nokta sayısı 1'dir. Algoritmanın başında belirlenen çaprazlama oranı (p_c), bir dizi için uygulanacak çaprazlama işleminin olasılığını gösterir. Bu oran tüm algoritma boyunca sabit olabileceği gibi, jenerasyonlara göre değişebilir. Bu oranın çok yüksek olması nüfus içinde yeni dizilerin daha hızlı oluşmasını sağlar. Dolayısıyla yüksek performanslı diziler de hızlı bir şekilde atılabilir. Diğer taraftan, çaprazlama oranı çok düşük olursa değişime uğrayacak dizi sayısı az olacağından araştırma durgunlaşıp yavaşlayabilir.

Basit bir çaprazlama işlemi iki adımda yürütülür. İlk olarak çaprazlama işlemine uğrayacak diziler rasgele belirlenir. İkinci adımda ise çaprazlama işlemini uygulamak için dizi üzerinde bir k tam sayı pozisyonu yine rasgele seçilir. Diziyi oluşturan karakterlerin sayısı ya da dizi uzunluğu l ise, k çaprazlama pozisyon değeri 1 ile l-1 arasında olmalıdır. Buna göre rasgele seçilen bir dizi çiftinde, k+1 ile l arasındaki bütün karakterler karşılıklı olarak yer değiştirerek iki yeni dizi elde edilir. Örnek problemin başlangıç nüfusundan A_1 ve A_2 dizileri seçilmiş olsun.

$$A_1 = 0 \quad 1 \quad 1 \quad 0 \quad | \quad 1$$
$$A_2 = 1 \quad 1 \quad 0 \quad 0 \quad | \quad 0$$

Çaprazlama işleminin uygulanacağı nokta k=4 olarak seçildiğinde, işlem sonucu elde edilen yeni diziler, (çaprazlama işleminin uygulanacağı nokta "|"sembolüyle gösterilmektedir.)

$$A_1' = 0 \quad 1 \quad 1 \quad 0 \quad | \quad 0$$
$$A_2' = 1 \quad 1 \quad 0 \quad 0 \quad | \quad 1$$

olur ve yeni jenerasyonda kopyalama işlemi için değerlendirilir.

Çift kesimli çaprazlama işleminde tek kesimli çaprazlama işlemine benzer işlem, kesimin dizi boyunca iki yerde yapılması ile aynen tekrarlanır.

$$A_1 = 1 \quad | \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{1} \quad \mathbf{0} \quad | \quad 1 \quad 1$$
$$A_2 = 1 \quad | \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{0} \quad \mathbf{1} \quad | \quad 0 \quad 0$$

Burada dizi çiftleri iki farklı yerden kesilerek çaprazlama uygulanır. Koyu renkli genler çaprazlanmıştır.

$A_1' = 1 \;\; | \; \mathbf{1} \;\; \mathbf{1} \;\; \mathbf{0} \;\; \mathbf{1} \; | \; 1 \;\; 1$

$A_2' = 1 \;\; | \; \mathbf{0} \;\; \mathbf{0} \;\; \mathbf{1} \;\; \mathbf{0} \; | \; 0 \;\; 0$

Görüleceği üzere bu kez de yeni şekil alan diziler sonucunda temsil ettikleri sayılar yine değişmiştir. İki noktalı çaprazlamada her dizi üç parçaya bölünür. Bu parçalardan karşılıklı her ikisinin çaprazlama yer değiştirmesi ile birbirinden farklı 6 tane yeni dizi elde edilir. Aşağıda iki noktalı çaprazlamaya maruz bırakılan dizilerde düz, italik ve koyu kısımlar olmak üzere üç parça gösterilmiştir.

000*10111***00110**

001*1111000***01001**

Çaprazlamanın sırası ile düz, italik ve koyu parçalar arasında yapılması ile birbirinden farklı kromozomlar düz kısımlar arasında,

001*10111***00110**

000*1111000***01001**

italik kısımlar arasında,

000*1111000***00110**

001*10111***01001**

ve son olarak da koyu kısımlar arasında,

000*10111***01001**

001*1111000***00110**

şeklinde olmak üzere 6 tanedir. Yeni dizilerin tümünün kullanılması gerekli değildir. Bu seçimde ya düz, ya italik veya koyu parçaların değiştirilmesinden elde edilen yeni dizilerden istenilen kadarı rasgele seçim ile nüfusa katılabilir.

Çok kesimli çaprazlama, dizilerin çapraz olarak ikiden daha fazla yerden rasgele kesilerek karakterlerin yer değiştirilmesi ile sağlanır.

$A_1 = 1 \;\; | \;\; \mathbf{0} \;\; \mathbf{0} \;\; | \;\; \mathbf{1} \;\; \mathbf{0} \;\; | \;\; 1 \;\; | \;\; 1$

$A_2 = 1 \;\; | \;\; \mathbf{1} \;\; \mathbf{1} \;\; | \;\; \mathbf{0} \;\; \mathbf{1} \;\; | \;\; 0 \;\; | \;\; 0$

Bu çaprazlama sonucunda oluşan diziler

$A_1' = 1 \;\; \mathbf{1} \;\; \mathbf{1} \;\; \mathbf{1} \;\; \mathbf{0} \;\; 0 \;\; 1$

$A_2' = 1 \;\; \mathbf{0} \;\; \mathbf{0} \;\; \mathbf{0} \;\; \mathbf{1} \;\; 1 \;\; 0$

gibidir. Burada da karşılıklı parçalar arasında çaprazlama ile çok sayıda yeni dizi elde edilebilir.

Tekdüze çaprazlama işleminin aslı rasgele hanelerin iki dizi arasında yer değiştirmesi ile olur.

1 **0** 0 **1** 0 1 **1**

↑ ↓ ↑ ↓ ↑ ↓

1 **1** 1 **0** 1 0 **0**

Burada her hane için yazı tura atılır. Örneğin yazı gelirse çaprazlama yapılsın, tura gelirse çaprazlama yapılmasın kuralına uyulur. Koyu renkli karakterler için yazı geldiğinden bu karakterler için çaprazlama uygulanarak karakterlerin yeni şekilleri

1 1 0 0 0 1 0

1 0 1 1 1 0 1

biçiminde oluşmuştur.

3.4.3. Mutasyon

Araştırma uzayının diğer noktalarını da incelemek için uygulanan genetik işlemlerden bir diğeri de mutasyondur. Algoritmanın başında belirlenen mutasyon oranı p_m, tüm nüfus içinde değişime uğrayacak karakter oranını belirler. Bu oran tüm algoritma boyunca sabit alınabileceği gibi jenerasyonlara göre farklı değerlerde de seçilebilir. Bu işleme göre, bir jenerasyonda yaklaşık olarak $p_m * n * l$ karakterde mutasyon meydana gelir. Burada n nüfus sayısı, l ise dizi uzunluğudur. Mutasyon tüm nüfus içinden rasgele seçilen bir karakterin değerinin değiştirilmesidir. Örneğin, binary olarak kodlanan bir problemde mutasyon işlemi, seçilen bir karakterin (bitin) değerini 1 ise 0, 0 ise 1 olarak değiştirmektir. Mutasyon oranının düşük olması, verilen herhangi bir karakter pozisyonunun tüm nüfus içinde daima tek bir değere yakınsayarak kalmasını önlemeye yardımcı olur. Yüksek oranda mutasyon ise tamamen rasgele bir araştırmayı baştan kabul etmek anlamına gelir.

Muhlenbein ve Backin'in yaptığı çalışmalar, mutasyonun olasılığının p_m, optimal oranının dizinin uzunluğu ve problemin çözüm uzayı ile orantılı olduğunu

ortaya koymuştur [24]. Çaprazlama olasılığı, genellikle 0.25 ila 1 arasında, mutasyon olasılığı 0.01 ila 0.001 arasında seçilir. Özellikle küçük nüfus sayılarında sistemin performansını çaprazlama olasılığından çok mutasyon olasılığı belirler.

Yapılan bazı çalışmalarda ise dinamik mutasyon ve çaprazlama olasılıkları kullanılmıştır. Bu çalışmalarda, jenerasyon sayısı arttıkça mutasyon olasılığı artmakta fakat çaprazlama olasılığı azalmaktadır. Dinamik çaprazlama ve mutasyon olasılıkları aşağıdaki gibi ifade edilebilir [24].

$$P_c = e^{(-T_G / M)}$$
$$P_m = e^{(0.005 * T_G / M)} \tag{3.3}$$

Eşitlik (3.3)'de T_G o andaki jenerasyon değerini, M maksimum jenerasyon sayısını göstermektedir.

Genetik algoritmanın yürütülmesi sırasında seçilen bu parametreler en iyileme probleminin performansını etkileyen en önemli faktörlerdir.

3.5. Genetik Algoritmanın Elle Yürütülmesi

Basit genetik algoritma işleyiş adımları aşağıdaki gibi sıralanabilir:

1) Başlangıç: Probleme aday çözümler için n tane l bit uzunluğunda dizilerden oluşmuş başlangıç nüfusunun rasgele oluşturulması,

2) Uygunluk: Nüfus içindeki her x dizisi için $f(x)$ uygunluk değerinin hesaplanması,

3) n tane diziye sahip yeni nüfus oluşuncaya kadar aşağıdaki adımların tekrar edilmesi,

4) Belirlenen seçim yöntemine göre başlangıç nüfusu içinden dizilerin seçilmesi. (Yüksek uygunluk değeri seçilme şansını artırır)

5) Yeni diziler oluşturmak için çaprazlama olasılığına göre, rasgele seçilen dizi çiftlerinin, rasgele belirlenen noktalarından çaprazlama işlemine tabi tutulması, (Eğer çaprazlama yapılmazsa yeni diziler eski dizilerin kopyası olacaktır)

6) Mutasyon olasılığına göre rasgele dizilerde mutasyon işleminin uygulanması,

7) Eldeki nüfusun yeni nüfusla yer değiştirilmesi,

8) Test: Bir jenerasyon boyunca yürütülen bu işlemlerin programın başında belirlenen bir jenerasyon sayısı kadar tekrarlanıp bitirilmesi veya en büyükleme ya da en küçükleme probleminde jenerasyonlarda elde edilen optimal değerler arasındaki fark sıfırlandığında ya da belli bir değere yakınsadığında sona erdirilmesi,

9) Döngü:2. adıma geri dönülmesi.

Basit bir genetik algoritmanın bir en iyileme problemine adım adım uygulanmasını görmek için bölüm 3.4.1'de verilen $f(x) = x^2$, $x \in$ [1,31], fonksiyonunun en büyüklemesi ele alınsın. Genetik algoritmanın uygulanmasında öncelikle, problemdeki değişkenleri sonlu uzunlukta dizilerle göstermek için kullanılacak koda karar vermek gerekir. Bu problem için, x değişkeni 5 bit uzunluğunda işaretsiz tam sayı olarak ikili sayı sisteminde kodlanabilir. Böylece x değişkeninin alt ve üst sınır değerleri 00000 (0) ve 11111 (31) olarak gösterilebilir. Başlangıç nüfusu, $n = 4$ için, bölüm 3.5.1'de verilen başlangıç nüfusu ile aynı seçilirse, algoritmanın bir sonraki safhası başlangıç nüfusundan yeni jenerasyonun seçilmesidir. Uygunluk fonksiyonuna göre aldıkları değere bağlı olarak, gelecek jenerasyona geçmesi beklenen dizileri belirlemek için ağırlıklı rulet çarkının dört kez çevrildiği kabul edilir. Çizelge 3.3'de gösterildiği gibi 1 ve 4 nolu diziler bir sonraki jenerasyonda bir kez görünmesine karşın 2 nolu dizi iki kez tekrarlanarak görülmektedir. 3 nolu kromozomun etkisi ise uygunluk değerinden dolayı sıfırdır.

Genetik algoritma içinde kopyalamadan sonra yürütülecek ikinci işlem çaprazlamadır. Çarprazlama oranı $p_c = 1.0$ seçildiğinde tüm diziler çaprazlama işlemine uğrar. Seçilen 1 ve 2 nolu diziler için çaprazlama yeri 4 seçildiğinde, 01101 ve 11000 dizilerinden farklı olarak 01100 ve 11001 dizileri elde edilir. Benzer şekilde, 3 ve 4 nolu diziler için çaprazlama yeri 2 olarak seçilirse, çaprazlama işleminden sonra iki farklı dizi elde edilmiş olur. Son işlem bit bit yürütülen mutasyon işlemidir. Örnek problem için mutasyon oranı 0.001 olarak

verildiğinde, tüm nüfus için toplam bit sayısı 20 olduğundan mutasyona uğrayacak bit sayısı 20*0.001'den 0.02 bit olarak bulunur. Sonuç olarak nüfus içindeki hiçbir bitin değeri 0'dan 1'e ya da 1'den 0'a değişmeden kalır.

Genetik algoritmanın tüm işlemlerinin ardından yeni bir jenerasyon nüfusunun dizileri belirlenmiş olur ve yeni jenerasyon test edilmeye hazırdır. Basit bir genetik algoritmayla yaratılan yeni diziler, onlu sayı sistemine göre x değerlerine dönüştürülerek uygunluk fonksiyonuyla değerlendirilir.

Çizelge 3.3. Genetik algoritmanın elle yürütülmesi [1]

Dizi	Başlangıç	x	$f(x)$	Seçilme		Beklenen
Gerçek-						
No.	Nüfusu	Değeri	x^2	Olasılığı	Sayı	Sayı
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Toplam			1170	1.00	4.00	4.0
Ortalama			293	0.25	1.00	1.0
Maksimum			576	0.49	1.97	2.0

Çaprazlanacak	Çaprazlanacak	Çaprazlama	Yeni	x	$f(x)$
Diziler	Dizi No.	Noktası	Nüfus	Değeri	x^2
0 1 1 0 I 1	2	4	0 1 1 0 0	12	144
1 1 0 0 I 0	1	4	1 1 0 0 1	25	625
1 1 I 0 0 0	4	2	1 1 0 0 1	27	729
1 0 I 0 1 1	3	2	1 1 0 0 1	16	256
Toplam					1754
Ortalama					439
Maksimum					729

Bir jenerasyon boyunca yürütülen tüm işlemlerin gösterildiği Çizelge 3.3'e bakıldığında dizilerin performansının bir jenerasyon sonra bile oldukça iyi derecede arttığı, uygunluk fonksiyonunun toplam, ortalama ve maksimum değerlerindeki artıştan gözlenebilmektedir. Bir jenerasyon boyunca yürütülen bu işlemler en büyükleme ya da en küçükleme probleminde jenerasyonlarda elde

edilen optimal değerler arasındaki fark sıfırlandığında ya da belli bir değere yakınsadığında sona erdirilir. Ayrıca bir genetik algoritma, programın başında belirlenen bir jenerasyon sayısı kadar tekrarlanıp bitirilebilir. Bu sayının yeterince büyük olması sonuçta elde edilecek değerin, fonksiyonun optimal çözümü olma şansını artırır [1, 24, 27].

3.6. Bulanık Mantık ve Genetik Bulanık Mantık Konusunda Yapılan Çalışmalar

Literatürde, bulanık mantık ve genetik bulanık mantık konusunda pek çok çalışma vardır. Genetik bulanık sistemler ile ilgili çalışmaların genel özeti Cordon ve ark. [30] çalışmalarında bulunmaktadır.

Arslan ve Kaya [31] çalışmalarında bulanık üyelik fonksiyonlarının belirlenmesinde genetik algoritmalardan yararlanmışlardır.

Gürocak [32] çalışmasında bulanık denetleyici kural tabanının oluşturulmasında genetik algoritmaya dayanan bir yöntem önermiştir.

Homafiar ve McCormick [33] ise genetik algoritma kullanarak, bulanık denetleyici üyelik fonksiyonlarının ve kural tabanının tasarlanması üzerinde durmuştur. Çalışmada, bulanık kurallar ve üyelik fonksiyonları tek bir dizi içerisinde kodlanmıştır.

Shi ve Eberhart [34] ise kullandığı genetik algoritma dizi yapısı içerisinde bulanık üyelik fonksiyonu ve kural tabanına ilave olarak üyelik fonksiyonlarının türlerini katmış ve farklı bir dizi yapısı oluşturmuştur.

Bu konuda yapılan diğer bazı çalışmalar ise Hu ve ark. [35] tarafından gerçekleştirilen genetik bulanık PID denetleyici, Zhou ve Lai [36] tarafından yapılan genetik algoritmalarla optimal bulanık denetleyici tasarımı şeklinde sıralanabilir.

Karr ve Gentry [37] çalışmalarında bir maden ayıklama ünitesinde, pH düzeyini ayarlayan genetik bulanık denetleyici kullanmıştır. Çalışmada basit genetik algoritma yapısındaki bir genetik algoritma ile ikili kodlama yöntemi kullanılarak yamuk şeklindeki bulanık üyelik fonksiyonları kodlanmış, seçim yöntemi olarak turnuva seçim yöntemi seçilmiştir. Elde edilen sonuçların oldukça başarılı olduğu görülmüştür.

Chin ve Qi [38] bulanık denetleyici performansının artırılması için genetik algoritmayı kullanmışlardır. Bu amaçla denetleyicide kullanılan kural sayısını azaltmak için verilen kurallardan gereksiz olanları çıkarmada genetik algoritmadan yararlanmışlardır. Azaltılan kuralların ve orijinal kuralların kullanıldığı sistemler (ters çevrilmiş sarkaç) arasında karşılaştırma yapmışlar ve kuralların azaltıldığı sistemlerin orijinal sistemden daha iyi bir performansa sahip olduğu sonucuna varmışlardır.

NASA, pilotların eğitiminde kullanılmak üzere Gulfstream II jet uçağının değiştirilmiş (modifiye edilmiş) hali olan Shuttle eğitim uçağı için bulanık kontrolün kullanıldığı bir simülatör geliştirmiştir [39].

Amerikan madencilik bürosu ve Alabama üniversitesindeki araştırmacılar A.B.D ordusu ile birlikte UH–1 helikopterinin kontrolü için genetik bulanık bir sistem geliştirmişlerdir. Genetik algoritmalar kural tabanının oluşturulmasında kullanılmıştır. Geliştirilen sistem hem benzetim hem de uçuş test aşamalarında başarılı sonuçlar vermiştir. Sistem, genetik algoritmalar yardımıyla uygun kuralların oluşturulması sonucunda farklı tip helikopterlerde de kullanılabilecektir [40].

1997 yılında Avrupada bulunan havacılık ve teknoloji grubu üyelerinden Schram ve Verbruggen [2] geniş gövdeli çift motorlu sivil yolcu uçağının iniş kontrol problemi için bulanık denetleyici tasarlamışlar ve başarılı benzetim sonuçları elde etmişlerdir.

Livchitz ve ark. [41] insansız küçük bir uçağın otomatik inişi için bulanık bir denetleyici geliştirmişlerdir. Geliştirilen sistem hem benzetimlerle hem de uçuşlarla test edilmiş ve oldukça başarılı sonuçlar elde edilmiştir.

FAA tarafından ise Piper Malibu tipi küçük bir uçağın bulanık kontrol sistemi için genel havacılık simülatöründe çalışmalar yapılmıştır. Bu çalışmalarda pilotlar, uçuş tecrübesi çok, az tecrübeli, öğrenci ve hiç tecrübesi olmayanlar olmak üzere dört grupta incelenmiştir. Her gruptaki 6 kişi üzerinde yapılan çalışmalarda sistemin çalışmasıyla ilgili kişiye minimum tanımlama yapılmış ve herhangi bir eğitim verilmemiştir. Sonuç olarak bulanık mantık burada performans kriteri aşamasında kullanılmış; sistemin değişken hatalarını ve aşma

miktarını düşürdüğü, öğrenme süresini azalttığı, kullanım için daha az çaba gerektirdiği ve tüm gruptaki elemanlar tarafından tercih edildiği görülmüştür [42].

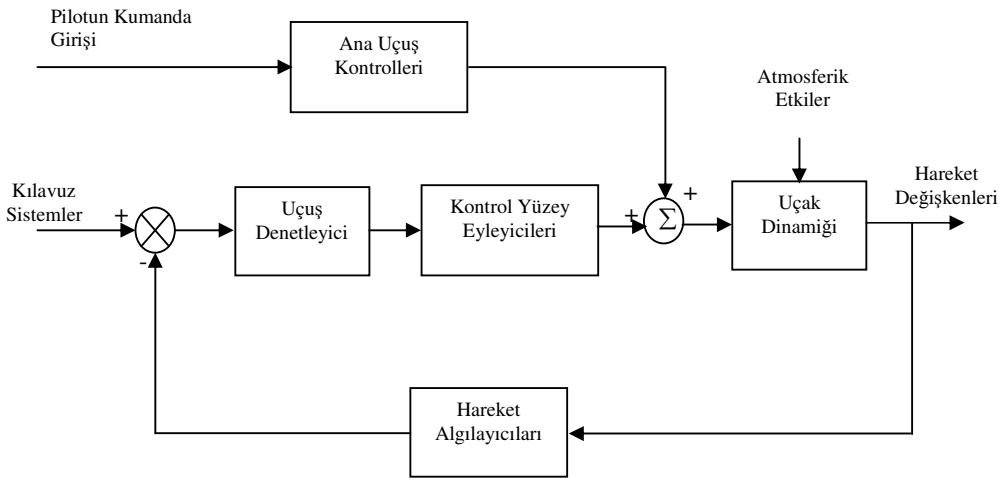
4. GENETİK ALGORİTMA TABANLI BULANIK PD DENETLEYİCİ TASARIMI VE UÇAK İRTİFA KONTROL SİSTEMİNE UYGULANMASI

Bu bölümde, ilk olarak uçuş kontrol sistemi ve uçak irtifa kontrol sistemi incelendikten sonra uçuş irtifa kontrol sistemi için klasik PD denetleyici tasarımı ve bulanık üyelik fonksiyonlarının sınır değerleri ve kural ağırlık değerleri genetik algoritma tarafından bulunan bulanık PD denetleyicinin tasarımı ele alınacaktır.

4.1. Uçuş Kontrol Sistemi

Bir uçağın hız vektörünü, yön, büyüklük ya da her ikisinde birden herhangi bir değişmeye karşı dirençli yapan özellikler, uçağın dinamik ve statik kararlılığıdır. Hız vektörünün kolaylıkla değiştirilebilir olması uçağın kontrolünün ne kadar iyi yapıldığı ile ilgilidir. Düzgün ve ivmesiz bir uçuş yörüngesinin muhafazası kararlılıkla mümkündür. Düzgün ve belli bir seviyede uçmak için sürekli kontrollü düzeltmeler ya pilot tarafından ya da otomatik uçuş kontrol sistemiyle yapılmalıdır.

Uçağı, üç boyutlu uçuş yörüngesinde belirlenen bir hedefe yöneltmek için gerekli ivmeleri üreten kuvvetler ve momentler, her uçakta bulunan kontrol yüzeyleriyle sağlanır. En temel kontrol yüzeyleri irtifa dümeni (elevator), kanatçıklar (ailerons) ve istikamet dümenidir (rudder). Bu kontrol yüzeylerine ilave olarak, uçağı yönlendirmek için her uçakta bulunan diğer elemanlar ise hareket algılayıcılarıdır. Bu algılayıcılar, uçak, atmosfer içinde herhangi bir bozucu etkiyle karşılaştığında ya da pilot tarafından uçağa bir kumanda verildiğinde hareket değişkenlerindeki değişimin ölçülmesini sağlar. Algılayıcılardan gelen bu sinyallerin bir göstergeyle pilot tarafından görülmesi sağlanabilir ya da bu sinyaller otomatik uçuş kontrol sistemi için geri besleme sinyalleri olarak kullanılabilir. Böylece bir otomatik uçuş kontrol sisteminin genel yapısı Şekil 4.1'deki blok şemayla gösterilebilir.



Şekil 4.1. Otomatik uçuş kontrol sisteminin genel yapısı [1]

Bu şema içinde denetleyicinin özelliği, kumanda edilen hareket ile ölçülen hareketi karşılaştırmak, eğer bir farklılık varsa uygulanacak kontrol kuvvet ve momentini meydana getiren kontrol yüzeylerinin hareketini sağlamak için eyleyicilere kumanda sinyalleri üretmektir. Bu işlem, kumanda edilen hareketle ölçülen hareket arasında belli bir uygunluk sağlanıncaya kadar devam eder [1].

Newton'un ikinci hareket yasasından elde edilen uçak hareket denklemleri incelendiğinde doğrusal olmadığı görülür. Çok basitleştirilmiş haller dışında bu eşitlik analitik olarak çözülememektedir. Bu nedenle, uçuş kontrol sistem tasarımında ilk olarak bu denklemlerin doğrusallaştırılması gerekir. Bu denklemler belli denge noktaları etrafında Taylor serisi ile doğrusallaştırılarak,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{4.1}$$

şeklinde durum uzayı formunda gösterilebilir. Bu ifadede; $\mathbf{x}$ durum değişkenleri vektörünü, $\mathbf{u}$ girdi vektörünü, $\mathbf{A}$ ve $\mathbf{B}$ ise katsayı matrislerini göstermektedir. Doğrusallaştırılan uçak hareket denklemlerinin durum uzayı yaklaşımı kullanarak gösterilmesi, girdi olarak seçilen kontrol yüzeyinin hareketinin her bir durum değişkenine etkisini gösteren transfer fonksiyonlarının oluşturulmasında büyük kolaylık sağlar.

En genel halde elde edilen bu denklemler, uzunlamasına ve yanlamasına hareket dinamiklerine ayrıldığında, uzunlamasına hareket durum değişkenleri ve kontrol girdisi;

$$\mathbf{x} = \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} \delta_E \end{bmatrix} \tag{4.2}$$

olarak elde edilmektedir. Kararlılık türevlerinden oluşan $\mathbf{A}$ ve $\mathbf{B}$ matrisleri ise;

$$\mathbf{A} = \begin{bmatrix} X_u & X_w & 0 & -g \\ Z_u & Z_w & U_0 & 0 \\ \tilde{M}_u & \tilde{M}_w & \tilde{M}_q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} X_{\delta_E} \\ Z_{\delta_E} \\ \tilde{M}_{\delta_E} \\ 0 \end{bmatrix} \tag{4.3}$$

biçiminde ifade edilmektedir. Bu ifadelerde u uçağın ileriye doğru hızı, w uçağın dönüş hızı, U_0 uçağın ileriye doğru denge hızı, q yunuslama açısal hızı, θ yunuslama açısı, δ_E irtifa dümeni açısı, g yerçekimi ivmesi, $X_u, X_w, Z_u,\ Z_w\ ,\ , X_{\delta_E}, Z_{\delta_E}\ M_u, M_q, M_{\dot{w}}, M_w$ ve M_{δ_E} ise ilgilenilen uçuş durumundaki kararlılık türevleridir. $\tilde{M}_u, \tilde{M}_w, \tilde{M}_q$ ve $\tilde{M}_{\delta_E}$ ise Eşitlik (4.4)'de görüldüğü gibi ifade edilmektedir.

$$\begin{aligned} \tilde{M}_u &= \left(M_u + M_{\dot{w}} Z_u\right) \\ \tilde{M}_w &= \left(M_w + M_{\dot{w}} Z_w\right) \\ \tilde{M}_q &= \left(M_q + U_0 M_{\dot{w}}\right) \\ \tilde{M}_{\delta_E} &= \left(M_{\delta_E} + M_{\dot{w}} Z_{\delta_E}\right) \end{aligned} \tag{4.4}$$

Bu çalışmada durum değişkeni olarak irtifa (h) kullanılacağı için durum değişkenleri arasına irtifayı dahil etmek üzere aşağıdaki dönüşümler yapıldığında küçük hücum açıları (α) için;

$$\alpha = \frac{w}{U_0} \tag{4.5}$$

$$\frac{\alpha(s)}{\delta_E(s)} = \frac{1}{U_0}\frac{w(s)}{\delta_E(s)} \tag{4.6}$$

$$a_{z_{cg}} = \dot{w} - U_0 q = -\ddot{h} \tag{4.7}$$

$$-\frac{s^2 h(s)}{\delta_E(s)} = \frac{sw(s)}{\delta_E(s)} - U_0\frac{q(s)}{\delta_E(s)} \tag{4.8}$$

eşitlikleri elde edilir. Eşitlik (5.8)'den kararlılık türevlerine bağlı olarak kısa periyod mod için $\frac{h(s)}{\delta_E(s)}$ uçak dinamiği;

$$\frac{h(s)}{\delta_E(s)} = \frac{1}{s^2}\left[\frac{-K_w s(1+sT_1) - U_0 K_q(1+sT_2)}{\Delta_{sp}(s)}\right] \tag{4.9}$$

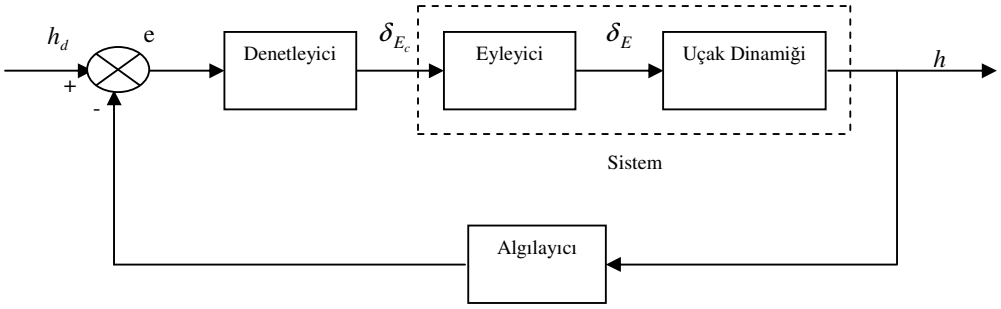
olarak elde edilir. Bu ifadede; $\Delta_{sp}(s)$, K_w, T_1, K_q ve T_2 aşağıdaki eşitliklerle verilmektedir.

$$\begin{aligned}
&\Delta_{sp}(s) = s^2 - \left[Z_w + M_q + M_{\dot{w}}U_0\right]s + \left[Z_w M_q - U_0 M_w\right] \\
&K_w = U_0 M_{\delta_E} - M_q Z_{\delta_E} \\
&T_1 = Z_{\delta_E} / K_w \\
&K_q = Z_{\delta_E} M_w - M_{\delta_E} Z_w \\
&T_2 = \left(M_{\delta_E} + Z_{\delta_E} M_{\dot{w}}\right)/ K_q
\end{aligned} \tag{4.10}$$

Eşitlik (4.5-4.10)'da; α hücum açısını, $a_{z_{cg}}$ uçağın ağırlık merkezinde ölçülen bozulmuş hareketin normal ivmesini, göstermektedir.

4.1.1. Uçak irtifa kontrol sistemi

Uçuş kontrol sisteminde önemli bir yer tutan uçak irtifa kontrol sistemine ait blok diyagram Şekil 4.2'de görülmektedir.



Şekil 4.2. Uçak irtifa kontrol sistemi

Sistem çıkışındaki irtifanın (h) girişteki referans irtifa değerini (h_d) izlemesi için eyleyici girişindeki irtifa dümeni açısı (δ_{E_c}) kontrol edilmektedir. Uçak dinamiği, belli bir uçuş durumu için eşitlik (4.9)'dan elde edilmektedir. Eyleyici ve algılayıcı dinamikleri ise $\frac{1}{1+0.1s}$ ifadesine eşittir.

Seçilen geniş gövdeli, dört motorlu jet yolcu uçağı için ilgilenilen uçuş durumlarındaki uçuş parametreleri Çizelge 4.1'de, bu uçuş durumları için elde edilen kararlılık türevleri ise Çizelge 4.2'de verilmektedir [45].

Çizelge 4.1. Uçuş durumu parametreleri [45]

Parametre	Uçuş Durumu 1	Uçuş Durumu 2	Uçuş Durumu 3
İrtifa (m)	6100	6100	12200
Mach no	0.5	0.8	0.8
$U_0\left(ms^{-1}\right)$	158	250	250
$\bar{q}\left(Nm^{-2}\right)$	8667	24420	9911
$\alpha_0(derece)$	6.8	0	4.6
$\gamma_0(derece)$	0	0	0

Çizelge 4.2. Kararlılık türevleri (uzunlamasına hareket) [45]

Kararlılık Türevi	Uçuş Durumu 1	Uçuş Durumu 2	Uçuş Durumu 3
X_u	0.003	-0.0002	0.0002
X_w	0.078	0.026	0.039
X_{δ_E}	0.616	0.0	0.44
$X_{\delta_{th}}$	$3.434\text{x}10^{-6}$	$3.434\text{x}10^{-6}$	$3.434\text{x}10^{-6}$
Z_u	-0.07	-0.09	-0.07
Z_w	-0.433	-0.624	-0.317
Z_q	-1.95	-3.04	-1.57
Z_{δ_E}	-5.15	-8.05	-5.46
$Z_{\delta_{th}}$	$-1.5\text{x}10^{-7}$	$-1.5\text{x}10^{-7}$	$-1.5\text{x}10^{-7}$
M_u	0.00008	-0.00007	0.00006
M_w	-0.006	-0.005	-0.003
$M_{\dot{w}}$	-0.0004	-0.0007	-0.0004
M_α	-0.948	-1.25	-0.75
$M_{\dot{\alpha}}$	-0.0632	-0.175	-0.1
Z_α	-68.414	-156	-79.25
M_q	-0.421	-0.668	-0.339
M_{δ_E}	-1.09	-2.08	-1.16
$M_{\delta_{th}}$	$0.67\text{x}10^{-7}$	$0.67\text{x}10^{-7}$	$0.67\text{x}10^{-7}$

Çizelge 4.1'de $\overline{q}$ dinamik basıncı, α_0 denge hücum açısını, γ_0 denge yörünge açısını göstermektedir.

Bu değerlere göre sırasıyla üç farklı uçuş durumu için irtifa ile irtifa dümeni arasındaki uçak dinamikleri;

(1.uçuş durumu)

$$\frac{h(s)}{\delta_E(s)} = \frac{-5.057s^2 - 346.213s - 69.678}{s^4 + 0.917s^3 + 1.13s^2} \tag{4.11}$$

(2.uçuş durumu)

$$\frac{h(s)}{\delta_E(s)} = \frac{-8.0382s^2 - 1043.8518s - 314.4175}{s^4 + 1.467s^3 + 1.666s^2} \tag{4.12}$$

(3.uçuş durumu)

$$\frac{h(s)}{\delta_E(s)} = \frac{-5.255s^2 - 582.127s - 87.75}{s^4 + 0.756s^3 + 0.857s^2} \tag{4.13}$$

olarak elde edilir.

Seçilen uçuş durumları için, klasik PD ve genetik bulanık PD denetleyici kullanılması durumlarında sisteme ait hata (*e*), irtifa dümeni açısı (δ_{E_c}), irtifa (*h*) ve irtifa değişiminin ($\dot{h}$) zamana (*t*) bağlı değişimleri, Matlab 6.5 kullanılarak kodlanmış program yardımıyla bulunmuştur.

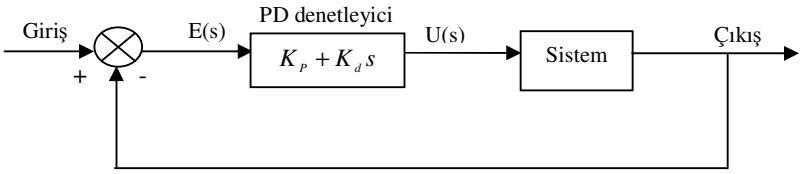
4.2. Klasik PD Denetleyici Uygulaması

Bir kapalı döngü denetim sistemi içinde denetim elemanının görevi ölçme elemanı üzerinden geri beslenen çıkış büyüklüğünü, giriş büyüklüğü ile karşılaştırmak ve karşılaştırmadan ortaya çıkabilecek hata değerinin yapısına ve kendi denetim etkisine bağlı olarak uygun bir kumanda veya denetim sinyali üretmektir. Denetim elemanlarında kullanılan belli başlı dört temel denetim etkisi vardır. Bunlar; ikili veya aç-kapa (on-off) etkisi, orantı (P- proportional) etkisi, integral (I-integral) etkisi ve türev (D- differential) denetim etkisidir. Bu temel denetim etkilerinin bir veya birkaçının bir arada uygun şekilde kullanılmasıyla değişik denetim etkilerinde çalışan denetim organları (P, PI, PD, PID) oluşturulur.

Şekil 4.3'te verilen çalışmamızda kullandığımız PD tip denetleyici D etkisinden dolayı hızlı bir çalışma sağlar, sönümü artırır ve maksimum aşma, yükselme ve yatışkın duruma ulaşma zamanlarını azaltır. Yatışkın durum hatası üzerinde pek etkili değildir. Az sönümlü ya da kararsız sistemlerde de etkili olmaz. Şekil 4.3'e göre denetleyici çıkış ifadesi

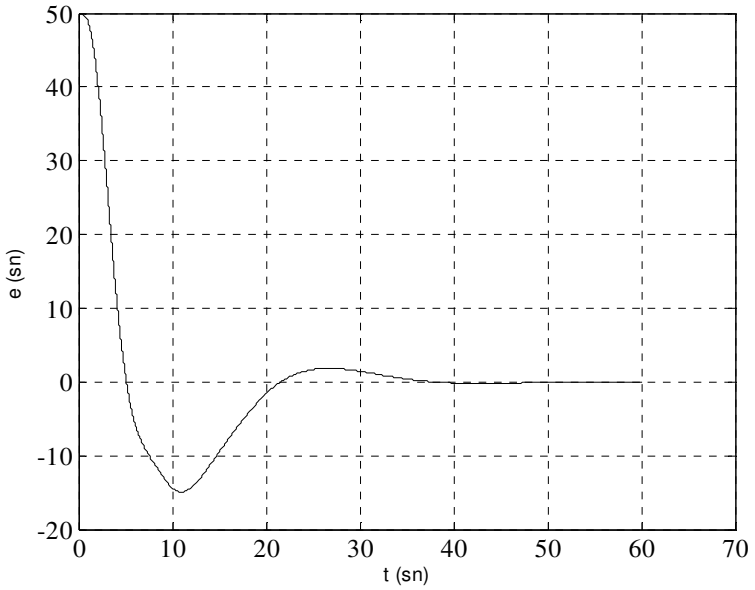
$$U(s) = (K_p + K_d s)E(s) \tag{4.14}$$

biçiminde ifade edilmektedir [43,44].

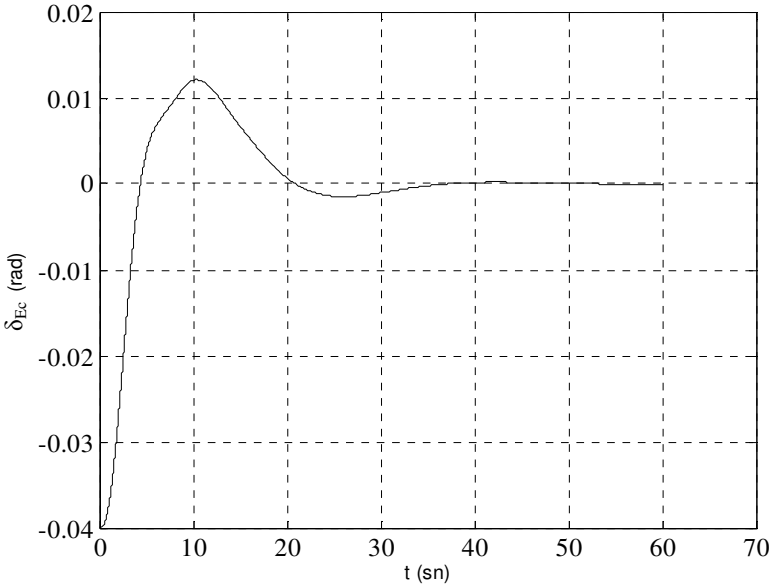


Şekil 4.3. Klasik PD kontrol sistemi

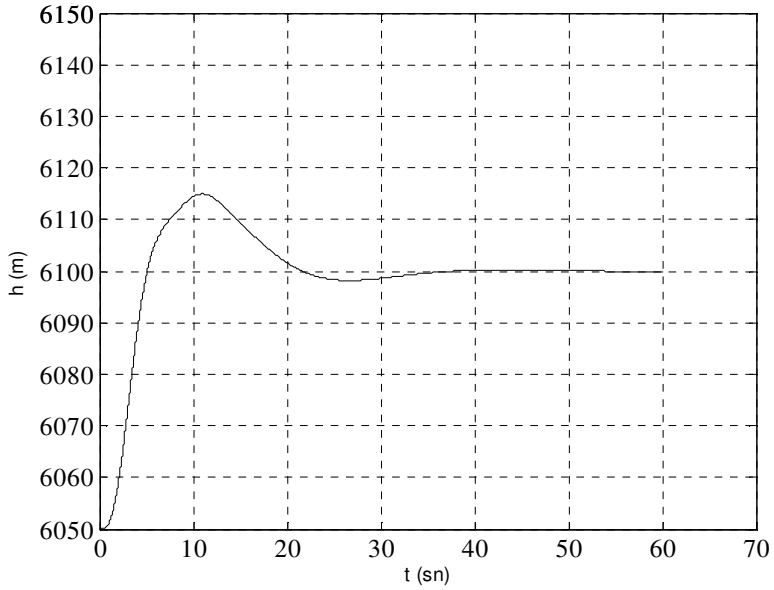
Denetleyici sistemde, K_p=−0.0008 ve K_d=−0.0005 olarak alındığında ve nominal uçuş durumlarından (1.uçuş durumu:6100 m, 2.uçuş durumu:6100 m, 3.uçuş durumu:12200 m) 50 m irtifa kaybı oluşması durumunda, üç uçuş durumu için sisteme ait $e(t), \delta_{E_c}(t), h(t), \dot{h}(t)$ değişimleri Şekil (4.4-4.15)'de sırasıyla görülmektedir.



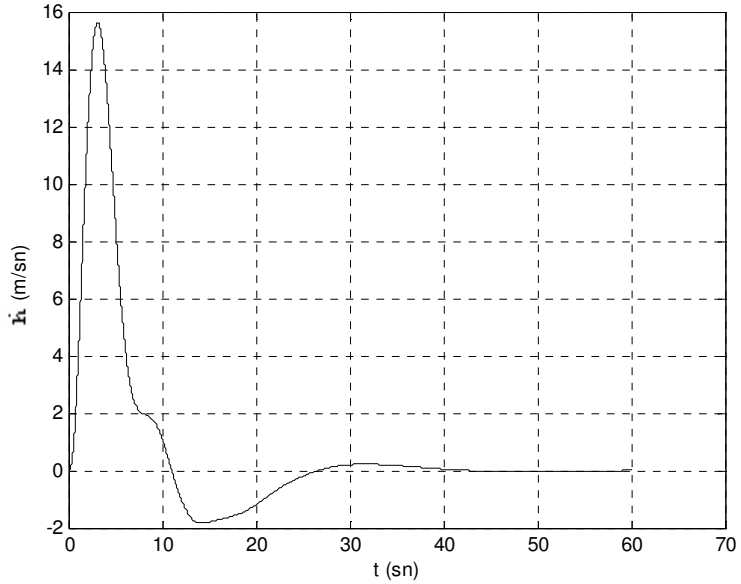
Şekil 4.4. Hatanın zamana göre değişimi
(1.uçuş durumu K_p=−0.0008 ve K_d=−0.0005)



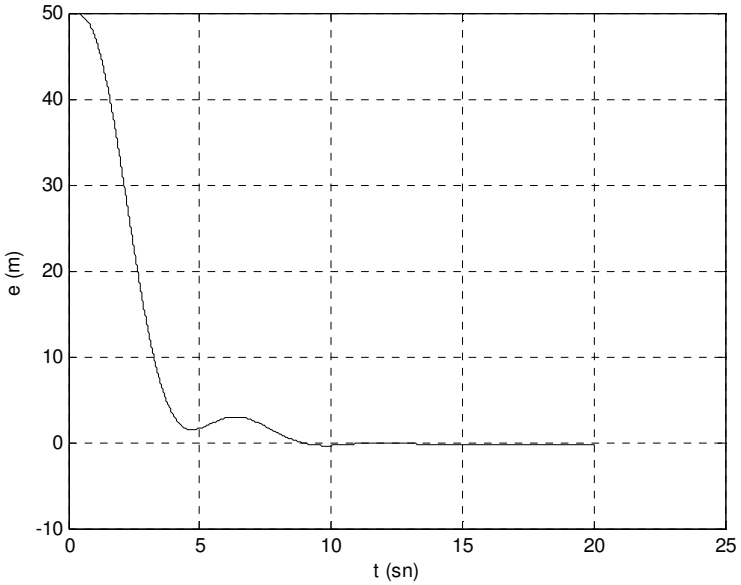
Şekil 4.5. İrtifa dümeni açısının zamana göre değişimi
(1.uçuş durumu K_p=−0.0008 ve K_d=−0.0005)



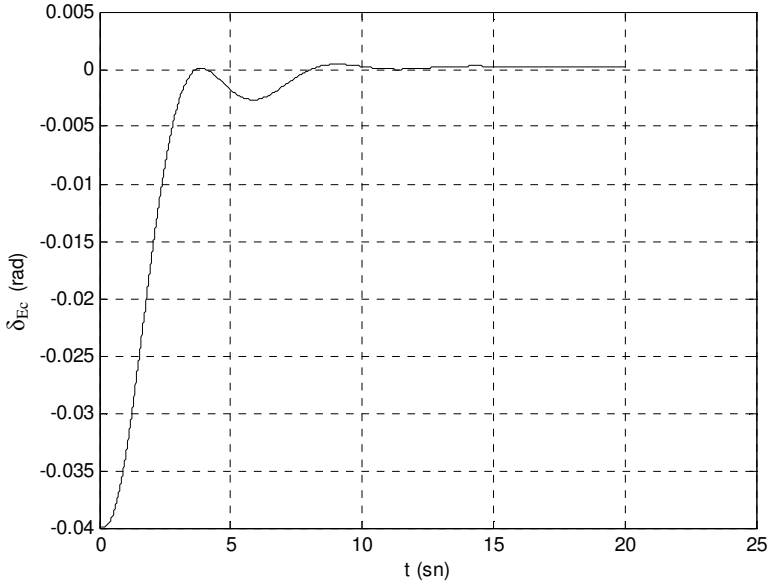
Şekil 4.6. İrtifanın zamana göre değişimi
(1.uçuş durumu K_p=−0.0008 ve K_d=−0.0005)



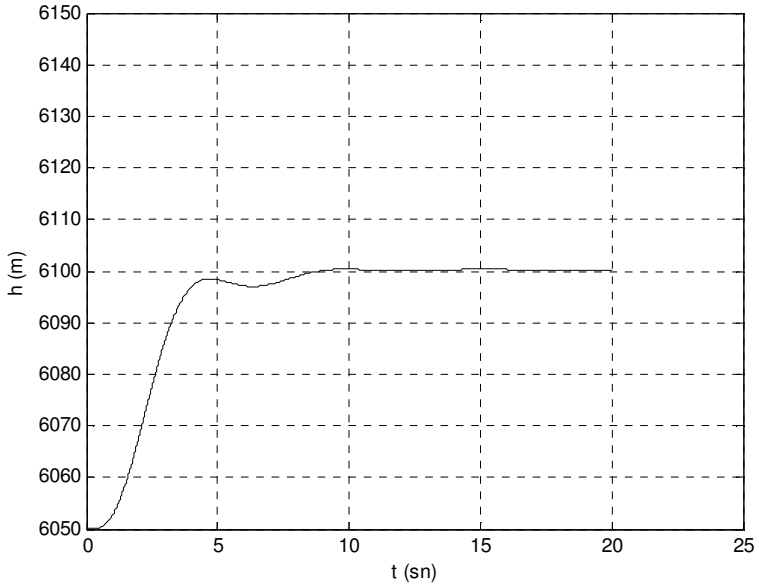
Şekil 4.7. İrtifa hızının zamana göre değişimi
(1.uçuş durumu K_p=−0.0008 ve K_d=−0.0005)



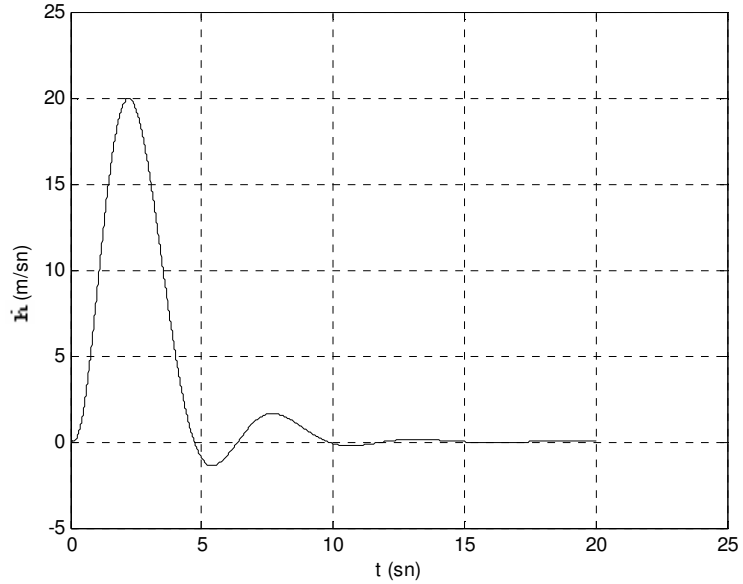
Şekil 4.8. Hatanın zamana göre değişimi
(2.uçuş durumu K_p=−0.0008 ve K_d=−0.0005)



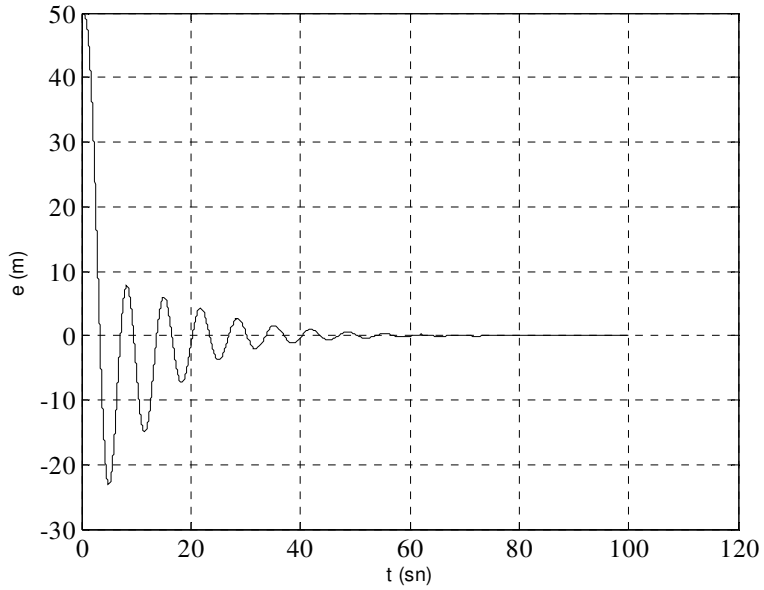
Şekil 4.9. İrtifa dümeni açısının zamana göre değişimi
(2.uçuş durumu K_p=−0.0008 ve K_d=−0.0005)



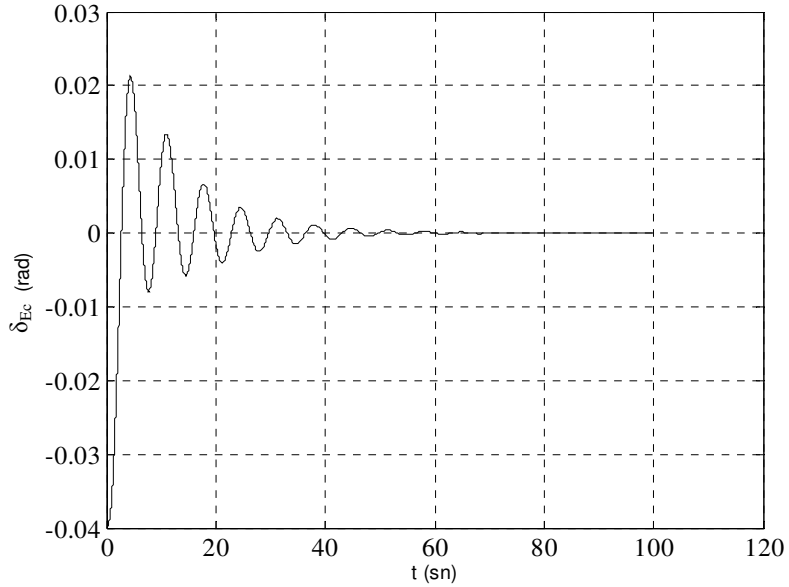
Şekil 4.10. İrtifanın zamana göre değişimi
(2.uçuş durumu K_p=−0.0008 ve K_d=−0.0005)



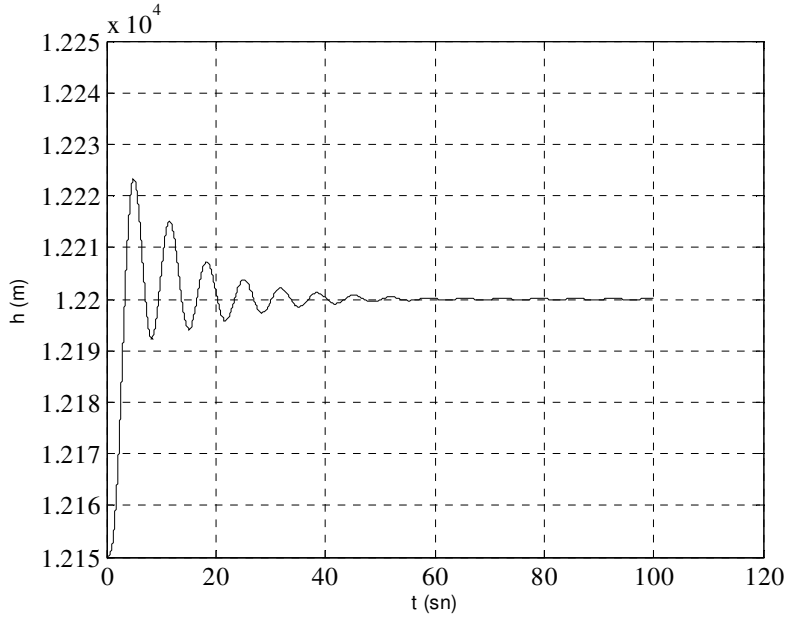
Şekil 4.11. İrtifa hızının zamana göre değişimi
(2.uçuş durumu K_p=−0.0008 ve K_d=−0.0005)



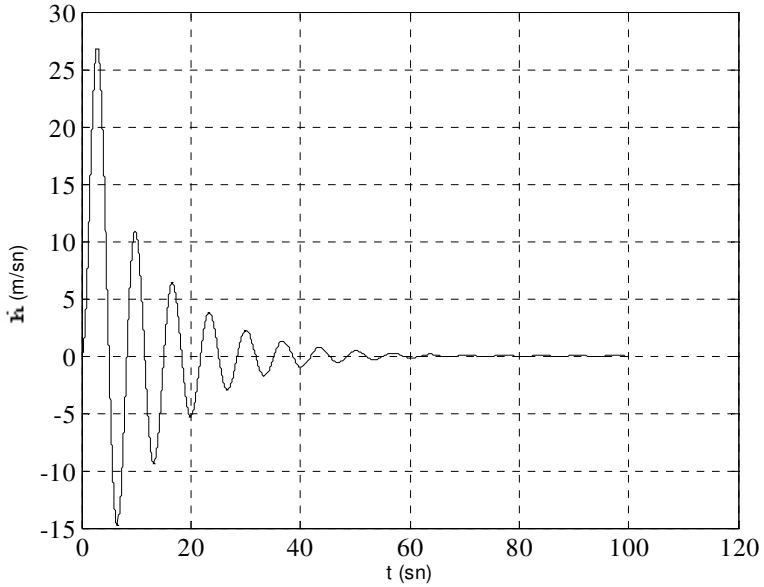
Şekil 4.12. Hatanın zamana göre değişimi
(3.uçuş durumu K_p=−0.0008 ve K_d=−0.0005)



Şekil 4.13. İrtifa dümeni açısının zamana göre değişimi
(3.uçuş durumu K_p=−0.0008 ve K_d=−0.0005)

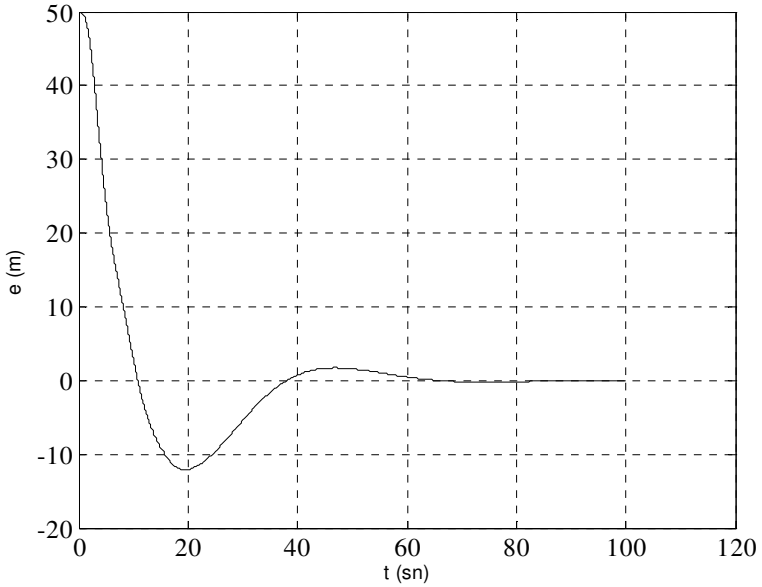


Şekil 4.14. İrtifanın zamana göre değişimi
(3.uçuş durumu K_p=−0.0008 ve K_d=−0.0005)

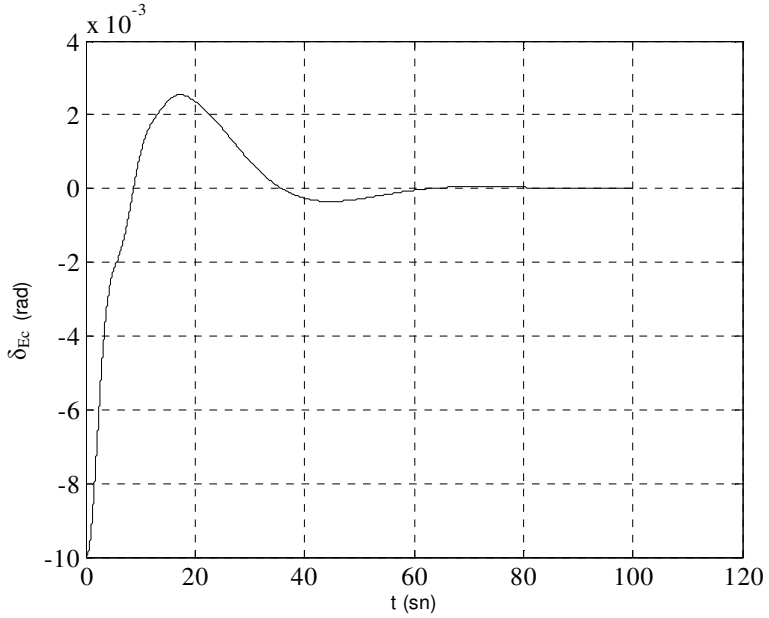


Şekil 4.15. İrtifa hızının zamana göre değişimi
(3.uçuş durumu K_p=−0.0008 ve K_d=−0.0005)

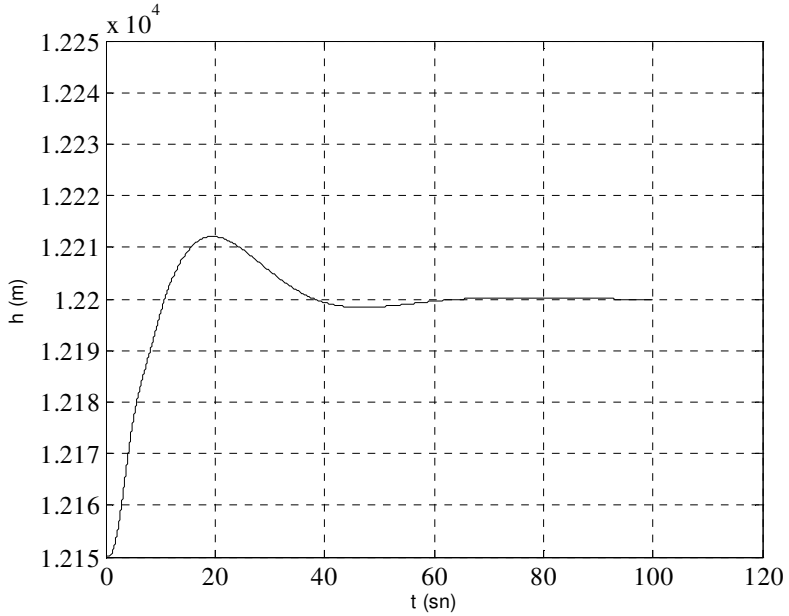
Şekil 4.14'de görüldüğü gibi 3. uçuş durumu için irtifa değişim periyodunun kısa, yatışkın duruma ulaşma zamanının uzun ve tepkinin düzgün olmadığı (osilasyon frekansı yüksek) görülmüştür. Dolayısıyla bir uçuş durumu için uygun olan PD denetleyicinin farklı bir uçuş durumunda aynı performansı sağlamadığı görülmüştür. 3. uçuş durumu için farklı K_p ve K_d değerleri denenmiş ve Şekil (4.16-4.19)'da görüldüğü gibi daha iyi çıkış tepkilerine K_p=−0.0002 ve K_d=−0.0004 seçildiğinde ulaşıldığı görülmüştür.



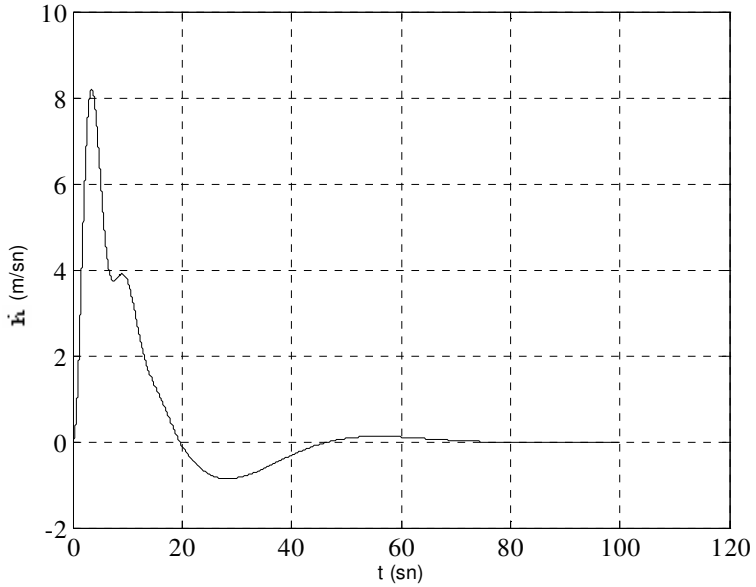
Şekil 4.16. Hatanın zamana göre değişimi
(3.uçuş durumu K_p=–0.0002 ve K_d=–0.0004)



Şekil 4.17. İrtifa dümeni açısının zamana göre değişimi
(3.uçuş durumu K_p=–0.0002 ve K_d=–0.0004)



Şekil 4.18. İrtifanın zamana göre değişimi
(3.uçuş durumu K_p=−0.0002 ve K_d=−0.0004)



Şekil 4.19. İrtifa hızının zamana göre değişimi
(3.uçuş durumu K_p=−0.0002 ve K_d=−0.0004)

4.3. Genetik Algoritma Tabanlı Bulanık Denetleyici Tasarımı

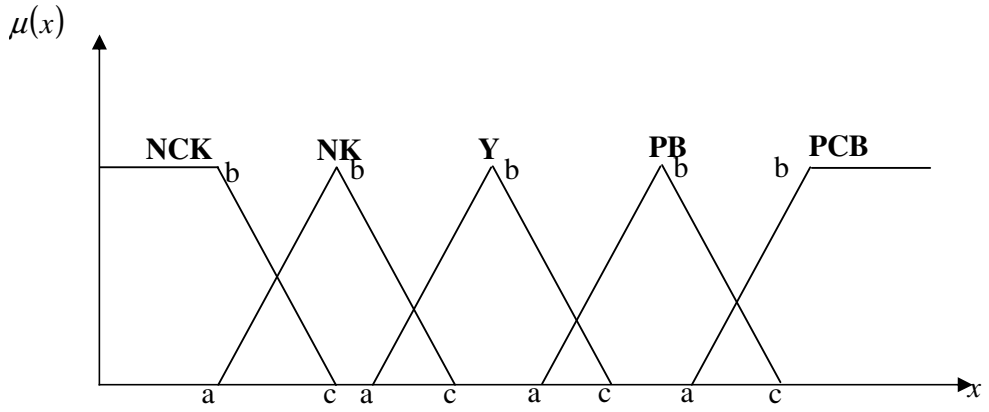
Bulanık kontrol sistem tasarımında en önemli süreç bilgi tabanının oluşturulması sürecidir. Önceki bölümlerde de belirtildiği gibi sistemin giriş ve çıkış değişkenleri, üyelik fonksiyon dağılımları (veri tabanı) şeklinde gösterilir. Bilgi tabanı veri tabanının yanında kural tabanını da içermektedir. Sisteme uygun bilgi tabanının oluşturulması oldukça zor bir süreçtir. Birçok uygulamada bilgi tabanının oluşturulmasında kişisel sezgi, mantık ve tecrübelerin kullanılmasına sıkça rastlanır. Bu durumda kontrol edilecek sistem hakkında yeterli ve doğru bilgilere sahip olunması gerekmektedir. Fakat bazı durumlarda yeterli derecede bilgi elde etmek mümkün olmamaktadır. Dolayısıyla optimal bir sistem elde etmek için kişisel sezgi, mantık ve tecrübeler yeterli olmamaktadır. Bu durumda bilgi tabanının oluşturulmasında kullanılan en etkili yöntemlerden birisi de genetik algoritmalardır.

Genetik bulanık ve bulanık sistemlerin birçok alanda (kontrol, imalat, ulaştırma, modelleme ve karar verme vb.) başarılı uygulamaları (yük trenlerinin hızlarının denetlenmesi, hızlı trenlerin enerji tüketiminin ve seyahat zamanının optimizasyonu, helikopter uçuş kontrol sistemi, havalandırma sistemlerinin kontrolü vb.) mevcuttur. Bu sistemler, maliyeti düşük, tasarım için fazla zaman gerektirmeyen performansı oldukça yüksek sistemlerdir [30,40,46,47,48].

4.3.1. Genetik Algoritma İle Bilgi Tabanının Kodlanması

Bilgi tabanı, veri tabanı ve kural tabanı birleşiminden oluşmaktadır. Sistemin giriş ve çıkış değişkenleri, üyelik fonksiyon dağılımları (veri tabanı) şeklinde gösterilir. Genetik algoritma kullanılarak bulanık denetleyici veri tabanı kodlanmasında pek çok yöntem geliştirilmiştir. Bu yöntemlerin hepsi daha önceden tanımlanmış bir kural tabanına ihtiyaç duyar ve kullanılan her bir dizi bütün üyelik fonksiyonlarının tiplerini ve sınırlarını içerir. Bu duruma göre oluşturulan dizi yapısı Şekil 4.20'de görülmektedir. Dizilerin kodlanmasında, pek çok durumda ikili kodlama kullanılır. Fakat genetik algoritma için bu bir zorunluluk değildir. Gerçel sayı kodlama, ağaç yapılı kodlama (tree coding) gibi

farklı kodlama biçimleri de kullanılabilir. Oluşturulan bir nüfusta, farklı veri tabanı bilgileri olacaktır.



(a)

NCK_b	NCK_c	NK_a	NK_b	NK_c	Y_a	Y_b	Y_c	PB_a	PB_b	PB_c	PCB_a	PCB_c

(b)

Şekil 4.20. Bulanık veri tabanının bir dizi içinde kodlanması (a) Bulanık üyelik fonksiyonu (b) Dizi yapısı

Burada NCK, NK, Y, PB ve PCB, *x* değişkenine ait üyelik fonksiyonlarını göstermektedir (NCK: negatif çok küçük; NK: negatif küçük, Y: yok; PB: pozitif büyük; PCB: pozitif çok büyük).

Genetik algoritma ile bulanık PD denetleyici kural tabanının kodlanmasında ise bulanık denetleyici kural tablosu bir dizi içerisinde kodlanmakta ve genetik algoritma kullanılarak öğrenme işlemi gerçekleştirilmektedir. Sugeno çıkarım yöntemindeki kural ağırlık değerlerinin kodlandığı örnek bir durum Şekil 4.21'de görülmektedir.

			e			
	u	HNCK	HNK	HY	HPB	HPCB
	HDNCK	A_1	A_2	A_3	A_4	A_5
	HDNK	A_6	A_7	A_8	A_9	A_{10}
$\dot{e}$	HDY	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}
	HDPB	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}
	HDPCB	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}

(a)

A_1	A_2	A_3	……	…..	…..	…..			A_{23}	A_{24}	A_{25}

(b)

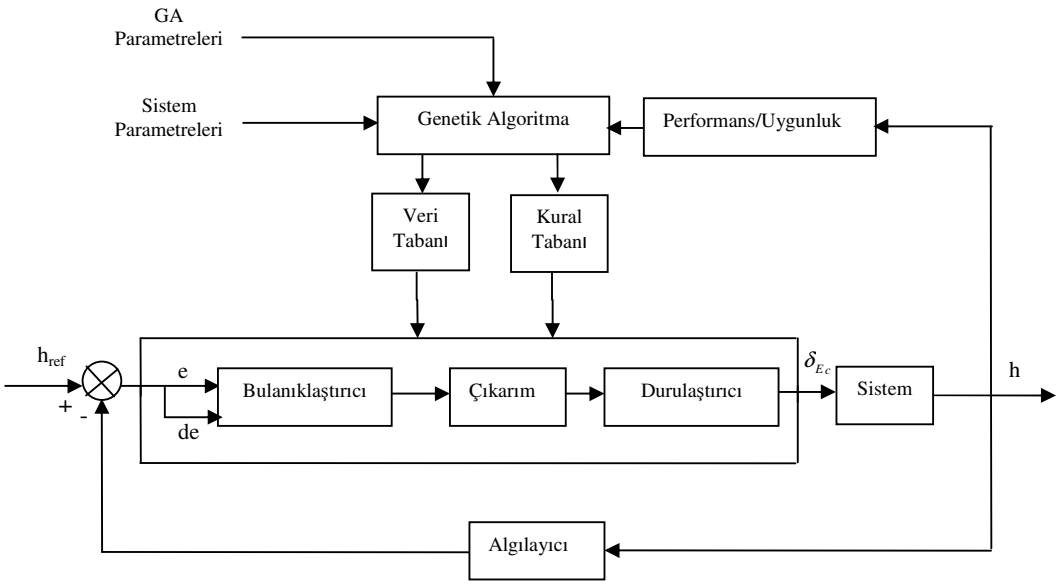
Şekil 4.21. Bulanık kural tabanının bir dizi içinde kodlanması (a) Kural ağırlık değer çizelgesi (b) Dizi yapısı

Burada; A_1, A_2……..A_n değerleri gerçel sayılardır ve kural ağırlık değerlerini göstermektedir. HNCK, HNK, HY, HPB, HPCB, HDNCK, HDNK, HDY, HDPB ve HDPCB ise hata ve hatanın değişimine ait üyelik fonksiyonlarını belirtmektedir. (H:hata; D:değişim; NCK: negatif çok küçük; NK: negatif küçük, Y:yok; PB: pozitif büyük; PCB: pozitif çok büyük)

4.4. Genetik Algoritma Tabanlı Bulanık PD Denetleyici Uygulaması

Bu bölümde, uçuş kontrol sisteminde önemli bir yer tutan uçak irtifa kontrol sistemi için bulanık üyelik fonksiyonlarının sınır değerleri ve kural ağırlık değerleri (kural tablosundaki gerçel sayı değerleri) genetik algoritma tarafından bulunan bir bulanık PD denetleyicinin tasarlanması incelenecektir. Böyle bir sistemin yapısı Şekil 4.22'de görülmektedir. Sistem çıkışındaki yüksekliğin girişteki referans yükseklik değerini izlemesi için irtifa dümeni açısı (δ_{Ec}) kontrol edilmektedir. Bulanık denetleyici; girişindeki hata ve hatanın değişimine göre uygun irtifa dümeni açı değerini hesaplamaktadır. Sistem çıkışındaki

yükseklik değeri ayrıca genetik algoritma uygunluk fonksiyonunun oluşturulmasında da kullanılmaktadır. Böyle bir sistemin geliştirilmesindeki en önemli süreç bulanık denetleyici bilgi tabanının genetik algoritma ile kodlanması sürecidir.



Şekil 4.22. Genetik algoritma ile bulanık kontrol parametrelerinin oluşturulması

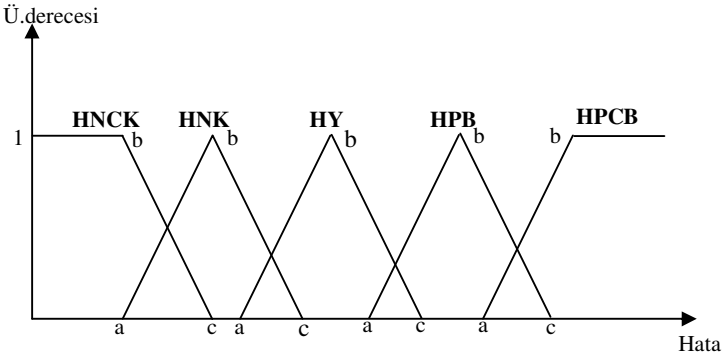
4.4.1. Bulanık denetleyici bilgi tabanının genetik algoritma ile kodlanması

Bölüm 3'de belirtildiği gibi genetik algoritma çözülecek problemin kendisi ile değil kodlanmış şekli üzerinden işlem yapar. Dolayısıyla çözülecek problemin kodlanma biçimi, genetik algoritmanın performansı üzerinde oldukça önemli bir etkiye sahiptir. Gerçekleştirdiğimiz çalışmada, literatürde en yaygın olarak kullanılan Ağırlıklı İkili Kodlama (Weighted Binary Code) biçimi kullanılmıştır.

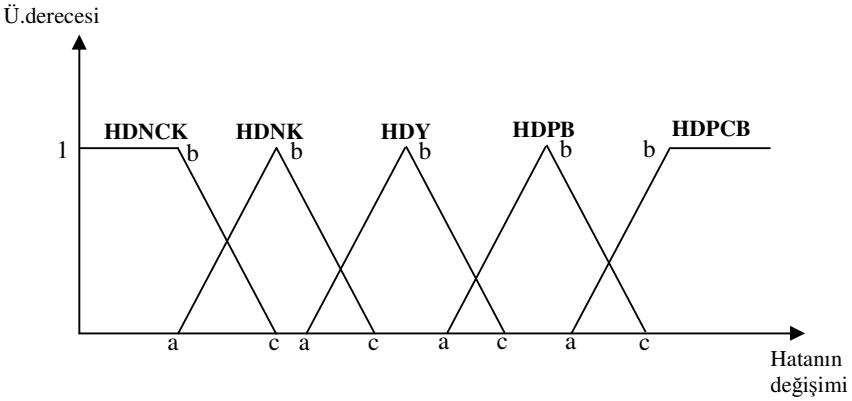
Şekil 4.23 ve Şekil 4.24'de sırası ile tasarlanan bulanık denetleyicinin hata ve hatanın değişimi girişlerine ait üyelik fonksiyonları görülmektedir. Fonksiyonların başlangıç ve bitiş değerleri kullanıcı tarafından program üzerinde istenen sınırlar arasında seçilmektedir.

Denetleyici sistemde, hata ve hatanın değişimine ait üyelik fonksiyonları Şekil 4.23 ve Şekil 4.24'de görüldüğü gibi 5'er adet, toplam 10 adet üyelik fonksiyonu ile tanımlanmıştır. Üçgen üyelik fonksiyonlarının seçilme nedeni, bu

üyelik fonksiyonlarının sadece üç parametre ile tanımlanabilmesidir. Bu parametreler, üçgenin taban sınır değerleri ve tepe noktasıdır. Üçgen üyelik fonksiyonlarına ait parametrelerin değerleri Şekil 2.2 ve Eşitlik (2.1)'de verilmiştir**.**



Şekil 4.23. Hata üyelik fonksiyonları



Şekil 4.24. Hatanın değişimi üyelik fonksiyonları

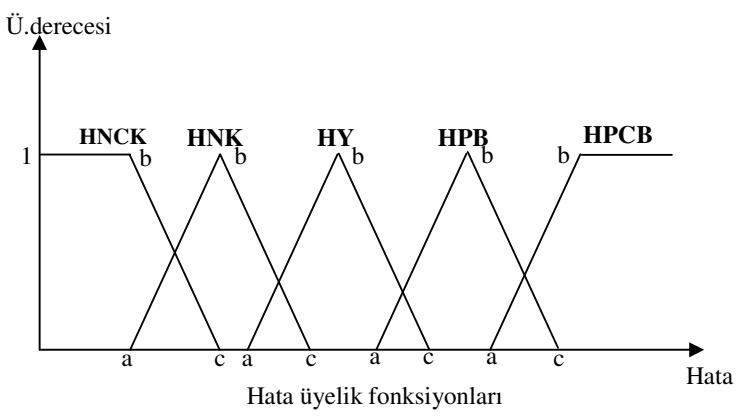
Bu şekilde, hata ve hatanın değişimi üyelik fonksiyonlarını tanımlamak için, toplam 26 adet parametrenin kodlanması gerekmektedir. Burada her bir parametre bir dizi olarak düşünülürse, verilen başlangıç değer aralıkları ve hassasiyet değerinin 0.1 seçilmesi durumunda, hata ve hatanın değişimine ait bütün üyelik fonksiyonlarını kodlamak için toplam 260 bitlik ikili bit dizisine ihtiyaç duyulur. Örnek olarak üyelik fonksiyonu sınır değerlerini kodlamak için gerekli bit sayısı aşağıda görüldüğü şekilde bulunmaktadır.

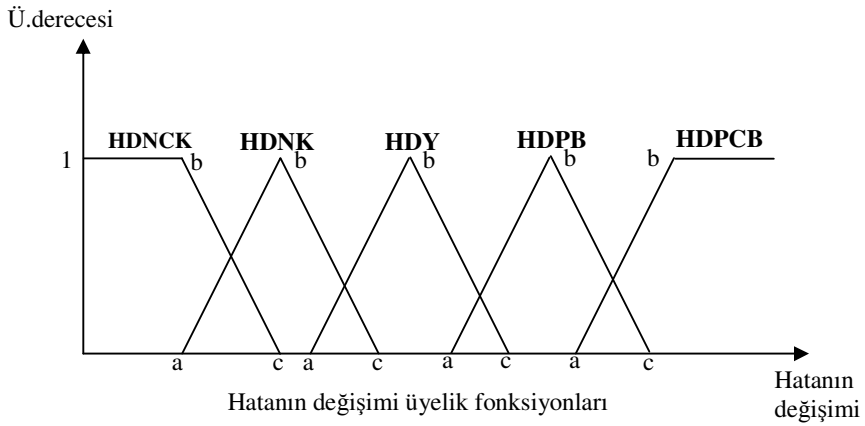
$$X_{\min} = -50$$
$$X_{\max} = 50$$
$$\text{Hassasiyet} = 0.1$$
$$\frac{X_{\max} - X_{\min}}{\text{Hassasiyet}} = \frac{50 - (-50)}{0.1} = 1000$$

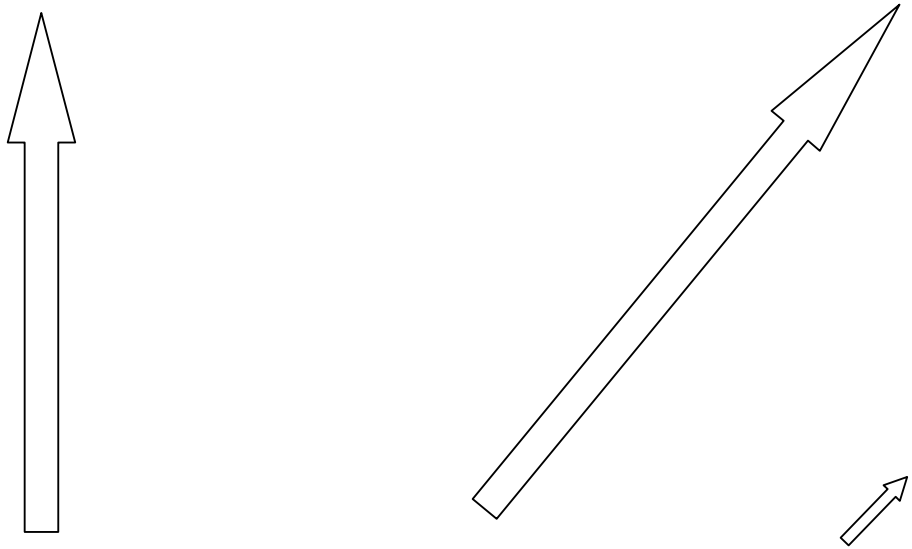
Burada $X_{\max}$ ve $X_{\min}$ üyelik fonksiyonu sınır değerlerini göstermektedir. 1000 sayısı ise 1111101000 şeklinde binary olarak 10 bit olarak kodlanmaktadır. Üyelik fonksiyonları sınır değerleri için toplam bit sayısı ise 26 adet parametre kodlayacağımız için 26*10=260 olacaktır.

Tasarladığımız bulanık denetleyici Sugeno tipinde bir denetleyicidir. Buna göre oluşturulan bulanık denetleyicinin kural ağırlık çizelgesi Şekil 4.25'de verilmiştir. Kural ağırlık çizelgesinde toplam 25 adet kural ağırlığı vardır. Kural ağırlık değer aralığının -2 ila 2 arasında ve hassasiyet değerinin 0.1 seçilmesi durumunda toplam 150 bitlik bir dizi oluşturulur. Bit sayısının bulunmasında üyelik fonksiyonları bit sayısının bulunmasında kullanılan yol izlenmektedir.

Bulanık denetleyicinin tasarlanmasında bütün üyelik fonksiyonları ve kural ağırlık değerleri tek bir dizi oluşturacak şekilde kodlama yapılmıştır. Bu şekilde, elde ettiğimiz dizide (26+25) olmak üzere toplam 51 adet parametre vardır. Bu dizinin toplam uzunluğu ise 410 ikili bit dizisi olur. Bu şekilde oluşturulmuş olan dizi yapısı Şekil 4.25'de gösterilmiştir. Başlangıçta verilen popülasyon sayısı kadar hata üyelik fonksiyonları, hatanın değişimi üyelik fonksiyonları ve kural ağırlık değerleri için kullanıcı tarafından verilen değer aralıklarında rasgele diziler oluşturulmaktadır.

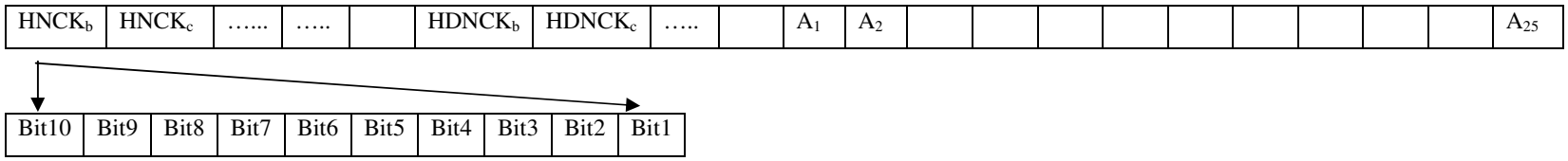






Kural ağırlık çizelgesi

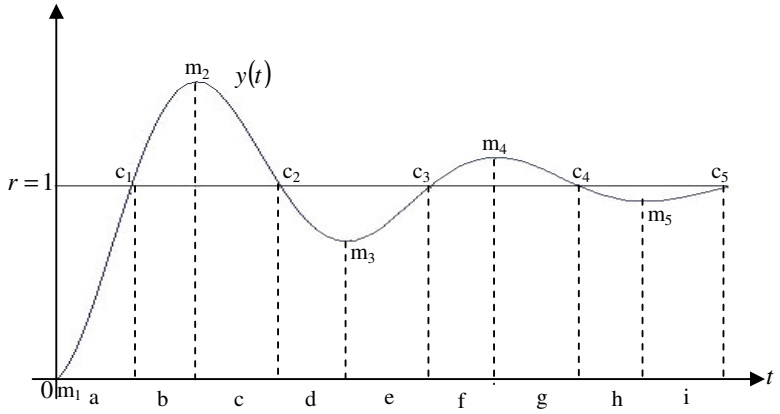
$\dot{e}$ \ e	HNCK	HNK	HY	HPB	HPCB
HDNCK	A_1	A_2	A_3	A_4	A_5
HDNK	A_6	A_7	A_8	A_9	A_{10}
HDY	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}
HDPB	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}
HDPCB	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}



Şekil 4.25. Bulanık PD denetleyici bilgi tabanının bir dizi içinde kodlanması

Gerçekleştirilen sistemde, bulanık denetleyici kural ağırlık değerlerinin oluşturulmasında genetik algoritmayla birlikte sezgisel yaklaşım kullanılmıştır. Literatürde, genetik bulanık sistemlerde, bulanık denetleyici kural ağırlık değerlerinin oluşturulması için yalnızca genetik yaklaşım kullanılmaktadır.

Kontrol teorisine göre ikinci mertebeden bir sistemin birim basamak referans girdiye karşılık çıkışındaki sinyalin zamanla değişimi Şekil 4.26'da görüldüğü gibidir. Bu grafiğe göre hata ve hatanın değişimi bilgilerinin bölgelere göre değişimi Çizelge 4.3'de verilmiştir. Böyle bir sistemde, bulanık denetleyici kural tabanı oluşturulurken, sistemin geçici rejim ve kalıcı rejimdeki kriterleri göz önüne alınır.



Şekil 4.26. İkinci mertebeden bir sistemin birim basamak tepkisi

Çizelge 4.3. Hata ve hatanın değişiminin bölgelere göre değişimi

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
e	+	-	-	+	+	-	-	+	+
$\dot{e}$	-	-	+	+	-	-	+	+	-

Burada c_1, c_2, c_3, c_4 referans geçiş noktalarını, m_1, m_2, m_3, m_4 referans uç noktalarını ve a, b, c, d, e, f, g, h, i referans aralıklarını ifade etmektedir.

Şekil 4.26'dan geçiş ve uç noktaları için aşağıdaki özellikleri çıkarmak mümkündür;

c_1: $(e > 0 \rightarrow e < 0)$ ve $\dot{e} \ll 0$,

c_2: $(e < 0 \rightarrow e > 0)$ ve $\dot{e} \gg 0$,

c_3: $(e > 0 \rightarrow e < 0)$ ve $\dot{e} < 0$,

c_4: $(e < 0 \rightarrow e > 0)$ ve $\dot{e} > 0$,

m_1: $\dot{e} \cong 0$ ve $e \gg 0$,

m_2: $\dot{e} \cong 0$ ve $e \ll 0$,

m_3: $\dot{e} \cong 0$ ve $e > 0$,

m_4: $\dot{e} \cong 0$ ve $e < 0$,

Tecrübelere ve dinamik işaret analizine göre bulanık denetim kuralları Çizelge 4.4' de listelenmiştir.

Çizelge 4.4. Geçiş ve uç noktalarındaki bulanık denetim kuralları

Kural no	e	$\dot{e}$	u	Referans Noktası
1	HY	HDNCK	NCK	c_1
2	HY	HDPCB	PCB	c_2
3	HY	HDNK	NK	c_3
4	HY	HDPB	PB	c_4
5	HPCB	HDY	PCB	m_1
6	HNCK	HDY	NCK	m_2
7	HPB	HDY	PB	m_3
8	HNK	HDY	NK	m_4

Denetim kurallarının belirlenmesinde kullanılan bazı yorumlar şu şekildedir:

1. d, h aralıklarında: e ="+" ve $\dot{e}$ ="+", hata pozitif ve artmaktadır. Dolayısıyla hatayı azaltmak için pozitif denetim girişi u verilmelidir.
2. e, i aralıklarında: e ="+" ve $\dot{e}$ ="-", hata pozitiftir ancak yavaş yavaş düşmektedir. Bu durumda denetim girişi u küçük olacak şekilde kurulmalıdır.
3. b, f aralıklarında: e ="-" ve $\dot{e}$ ="-", bu durum (1) deki durumun tam tersidir.
4. c,g aralıklarında e ="-" ve $\dot{e}$ ="+", bu durum ise (2) deki durumun tam tersidir.

Yukarıdaki çıkarımlara göre referans aralıklarında elde edilen sözel kuralların listesi Çizelge 4.5'de verilmektedir.

Çizelge 4.5. Bulanık denetim kural çizelgesi

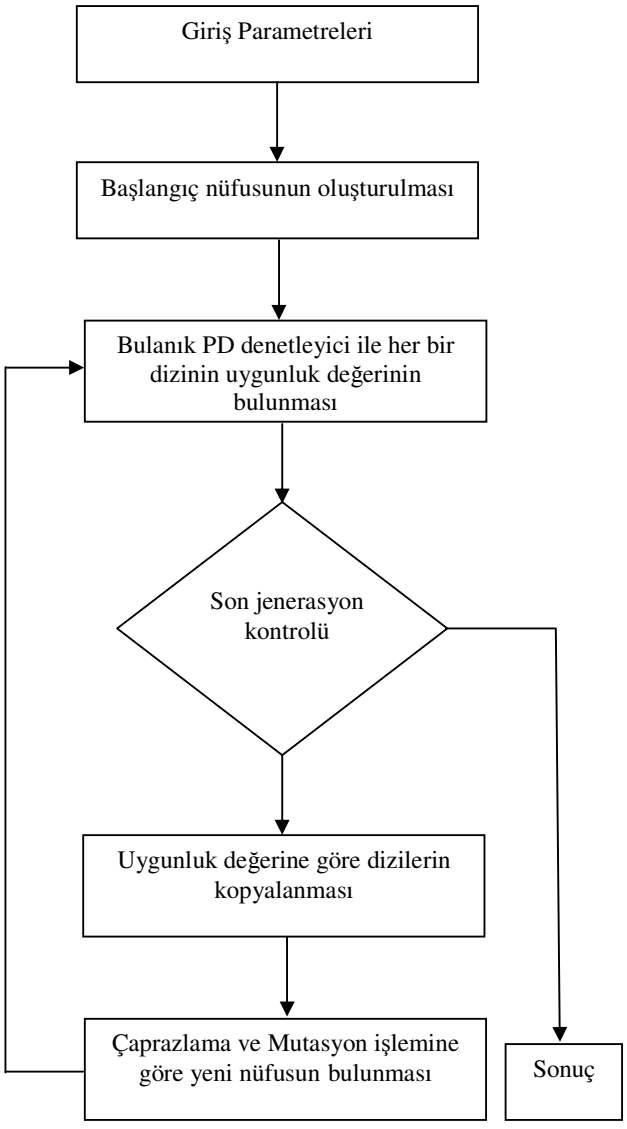
	e					
ė	*u*	HNCK	HNK	HY	HPB	HPCB
	HDNCK	NB	NB	NB	S	S
	HDNK	NB	NB	N	P	P
	HDY	NB	N	S	P	PB
	HDPB	NO	N	P	PB	PB
	HDPCB	NO	S	PB	PB	PB

Bu yaklaşım, Sugeno bulanık modelini kullanan bir genetik bulanık denetleyici için düzenlenirse, oluşturulacak dizi yapısı için aşağıdaki düzenlemeler yapılmalıdır:

- Kural tablosunda S (sıfır) ile gösterilen kural ağırlık değerlerine karşılık gelen dizi bit değerleri sıfır yapılır.
- Kural tablosunda P (pozitif) ve PB (pozitif büyük) ile gösterilen kural ağırlık değerlerine karşılık gelen dizi bit değerleri pozitif yapılır. (Kullandığımız uçuş irtifa kontrol sisteminde pozitif giriş negatif irtifa dümeni açısı ile sağlandığından programda P (pozitif) ve PB (pozitif büyük) ile gösterilen kural ağırlık değerlerine karşılık gelen dizi bit değerleri negatif yapılmıştır)
- Kural tablosunda NK (negatif küçük), NO (negatif orta) ve NCK (negatif çok küçük) ile gösterilen kural ağırlık değerlerine karşılık gelen dizi bit değerleri negatif yapılır. (Kullandığımız uçuş irtifa kontrol sisteminde negatif giriş pozitif irtifa dümeni açısı ile sağlandığından dolayı programda N (negatif) ve NB (negatif büyük) ile gösterilen kural ağırlık değerlerine karşılık gelen dizi bit değerleri pozitif yapılmıştır) [21,24].

4.4.2. Program akış diyagramı (Sistemin yazılım yapısı)

Bu çalışmada seçilen uçuş durumu (durumları) için sisteme ait hata (*e*), irtifa dümeni açısı (δ_{E_c}), irtifa (*h*) ve irtifa değişiminin ($\dot{h}$) zamana (*t*) bağlı değişimleri, genetik bulanık PD denetleyici kullanılması durumunda Matlab 6.5 kullanılarak kodlanmış program yardımıyla bulunmuştur. Programın akış diyagramı Şekil 4.27'de görülmektedir.



Şekil 4.27. Genetik bulanık denetleyici programının akış diyagramı

Giriş Parametreleri:

tr: Çalışma süresi

ts: Runge Kutta integrasyon aralığı

GA Parametreleri:

Xmin, Xmax: Üyelik fonksiyonları sınır değerleri (seçilen nominal uçuş durumundan olabilecek hata sınır değerleri)

ka_{min}, ka_{max}: Kural ağırlık sınır değerleri

Hassasiyet: Hassasiyet

Pn: Nüfus sayısı

Pc: Çaprazlama oranı

Pm: Mutasyon oranı

gn: Jenerasyon sayısı

Başlangıç nüfusunun oluşturulması: Basit genetik algoritma yapısına uygun olarak geliştirilen genetik1 alt programında hata ve hatanın değişimine ait üyelik fonksiyonu sınır değerleri ve kural ağırlık değerleri oluşturulmaktadır.

Her dizinin uygunluk değerinin bulunması: Popülasyon içindeki her bir dizi (üyelik fonksiyonu sınır değerleri ve kural ağırlık değerleri), onlu sayı sistemine göre x değerlerine dönüştürülerek, bulanık denetleyiciye verilir ve tam bir sistem tepkisi elde edilir.

Bulanık denetleyici sisteme verilecek δ_{E_c} değerini hesaplarken genetik algoritmanın bulduğu üyelik fonksiyonu sınır değerlerini, kural ağırlık değerleri ile hata ve hatanın değişimini kullanmaktadır. Öncelikle hata ve hatanın değişimine ait giriş değerleri, genetik algoritma tarafından değerleri bulunan üyelik fonksiyonlarıyla bulanıklaştırılır. Elde edilen sonuç, çıkarım ünitesinde önceden tanımlanan 25 adet kural ile yorumlanır. Durulaştırma biriminde ağırlıklı ortalama yöntemine göre, yorumlanan her bir kurala ait en küçük üyelik derece değeri kural ağırlık değeri ile çarpılır ve bulunan sonuçlar toplanır. Bu değer, kuralların toplam üyelik değerlerine bölünerek denetleyici çıkışı bulunur.

Sisteme ait irtifa dümeni açı değerlerini içeren kural ağırlık değer çizelgesinde irtifa dümeni açı değerleri derece cinsinden verilmektedir. Fakat transfer fonksiyonlarındaki irtifa dümeni değişimleri sistemi radyan cinsinden etkilediğinden, kodlanan programda denetleyici çıkışında elde edilen irtifa dümeni

açı değerleri radyana çevrilmektedir. Bu şekilde Sugeno tipinde bir bulanık denetleyici gerçekleştirilmiş olur. Elde edilen kontrol bilgisi uçak dinamiğine (durum uzayı formunda) uygulanır. Çıkış seviyesine ait hata ve hatanın değişimine ait değerler tekrar bulanık denetleyiciye uygulanarak bir döngü elde edilir. Bu şekilde elde edilen değerler ile, sistemin genetik bulanık denetleyici tarafından bulunan bulanık değerlere göre çalışması sağlanır. Kullanıcı tarafından verilen "tr" süresi boyunca her bir jenerasyona ait minimum hatayı veren dizi değerinden (üyelik fonksiyonları sınır değerleri ve kural ağırlık değerleri) elde edilen bulanık denetim bilgisi sisteme verilmektedir.

Her diziye ait uygunluk değeri ise $f_t(i) = \frac{1}{abs(e_t(i))}$ ifadesine eşittir. $e_t(i)$;i. diziye ait toplam (run time (tr) boyunca elde edilen) hatayı göstermektedir. Genetik algoritmada f_t'nin maksimizasyonu problemi ele alınmakta dolayısıyla hatanın minimizasyonu sağlanmaya çalışılmaktadır.

Son jenerasyon kontrolü: Genetik algoritma, programın başında belirlenen jenerasyon sayısı kadar tekrarlanıp bitirilir. Bu sayının yeterince büyük olması sonuçta elde edilecek değerin, fonksiyonun optimal çözümü olma şansını artırır.

Uygunluk değerine göre dizilerin kopyalanması: Dizilerin kopyalanmasında ağırlıklı rulet çarkı yöntemi kullanılmıştır. Bu yöntemde uygunluk değerine göre dizilerin kopyalanması, yüksek değere sahip olan dizilerin gelecek jenerasyona bir ya da daha fazla ürün olarak katkıda bulunma olasılığının yüksek olmasıdır. Dizilerin kopyalanmasında elistik modelden yararlanılmıştır. Kopyalama, çaprazlama ve mutasyon işlemleri sonrasında jenerasyondaki en yüksek uygunluğa sahip dizi sonraki jenerasyona aktarılmayabilir. Bunu önlemek için bu işlemlerden sonra oluşan yeni jenerasyona bir önceki jenerasyonun en iyi (elit) dizisi aktarılır.

Çaprazlama ve mutasyon işlemine göre yeni nüfusun bulunması: Kopyalama işleminden sonra her jenerasyonda, mevcut nüfus dışında, aynı uzay içinde farklı noktalara ulaşmak ya da araştırma uzayının diğer noktalarını da incelemek için yeniden düzenleyici genetik işlemler (çaprazlama (tek kesimli) ve mutasyon) uygulanarak yeni nüfus içinde bazı değişimler ortaya çıkarılır**.**

Sonuç: Program sonunda seçilen uçuş durumu için (jenerasyon sayısı*tr) süresi sonunda tr aralıklarla herbir jenerasyona ait minimum hatayı veren dizi değerinden (üyelik fonksiyonları sınır değerleri ve kural ağırlık değerleri) elde edilen $f(t,e), f(t,h), f(t,\delta_{E_c}), f(t,\dot{h})$ değişimleri çizdirilmektedir.

Ayrıca son jenerasyonda minimum hatayı veren diziye ait, üyelik fonksiyonları sınır değerleri, kural ağırlık değerleri ilgili dosyaya yazdırılmakta ve bu diziye ait seçilen uçuş durumu için elde edilen $f(t,e), f(t,h), f(t,\delta_{E_c}), f(t,\dot{h})$ değişimleri çizdirilmektedir.

4.4.3. Genetik bulanık PD denetleyicinin çeşitli uçuş durumlarına uygulanması

Aşağıdaki örneklerde farklı uçuş durumları için Ek-2'de verilen programda, genetik algoritma parametreleri, çalışma süresi ve runge-kutta integrasyon aralığı farklı değerlerde seçilerek, (jenerasyon sayısı*tr) süresi sonunda tr aralıklarla her bir jenerasyona ait minimum hatayı veren dizi değerinden (üyelik fonksiyonları sınır değerleri ve kural ağırlık değerleri) elde edilen $f(t,e), f(t,h), f(t,\delta_{E_c}), f(t,\dot{h})$ değişimleri ve ayrıca son jenerasyonda minimum hatayı veren diziye ait, üyelik fonksiyonları sınır değerleri, kural ağırlık değerleri ilgili dosyaya yazdırılmakta ve bu diziye ait seçilen uçuş durumu için elde edilen $f(t,e), f(t,h), f(t,\delta_{E_c}), f(t,\dot{h})$ değişimleri elde edilmiştir. Ayrıca farklı parametreler için elde edilen değişimler de Ek-1'de verilmiştir.

Örnek 1: 1.Uçuş durumu

Nüfus sayısı: 80

Jenerasyon sayısı: 30

Hassasiyet: 0.1

Çaprazlama oranı: 0.7

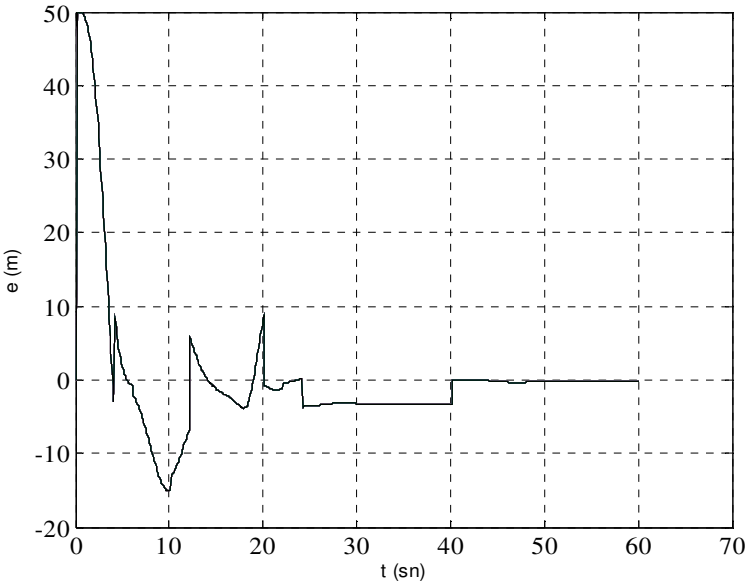
Mutasyon oranı: 0.002

Çalışma süresi: 2 sn

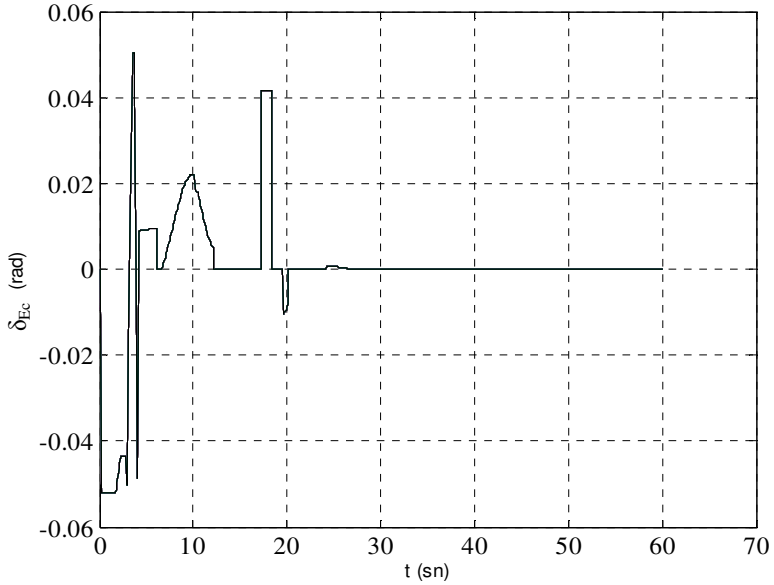
Runge-Kutta integrasyon aralığı: 0.1

Üyelik fonksiyonları sınır değerleri: ±50 m

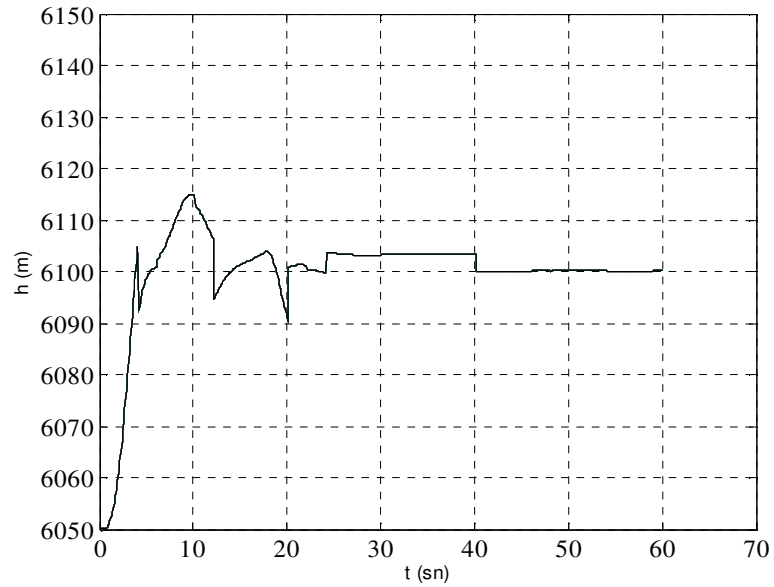
İrtifa dümeni açısı sınır değerleri: ±3°



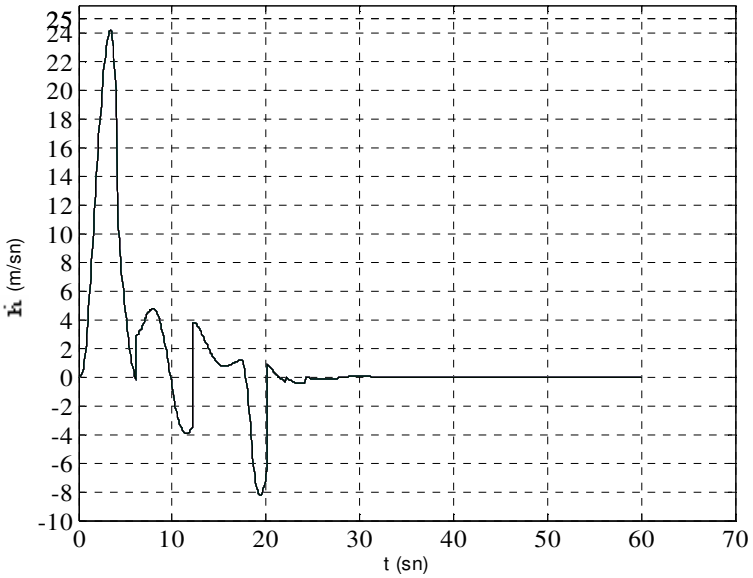
Şekil 4.28. Örnek 1 için hatanın zamana göre değişimi
(tüm jenerasyonlar)



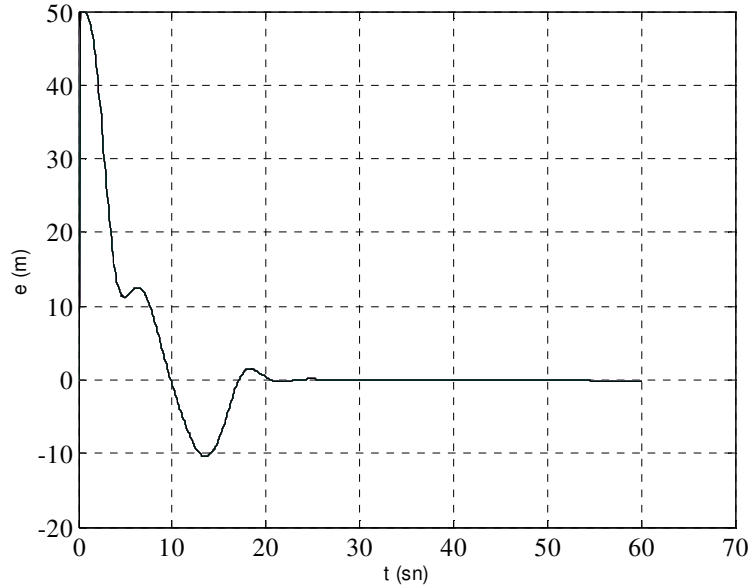
Şekil 4.29. Örnek 1 için irtifa dümeni açısının zamana göre değişimi (tüm jenerasyonlar)



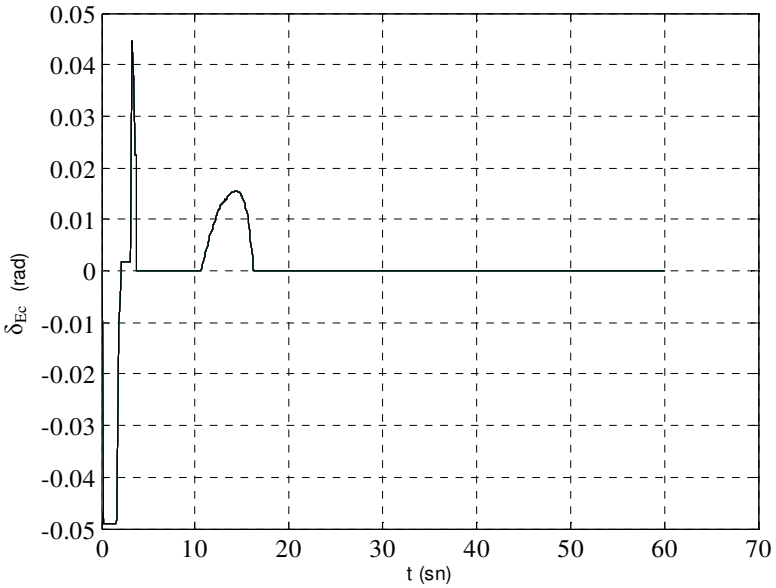
Şekil 4.30. Örnek 1 için irtifanın zamana göre değişimi (tüm jenerasyonlar)



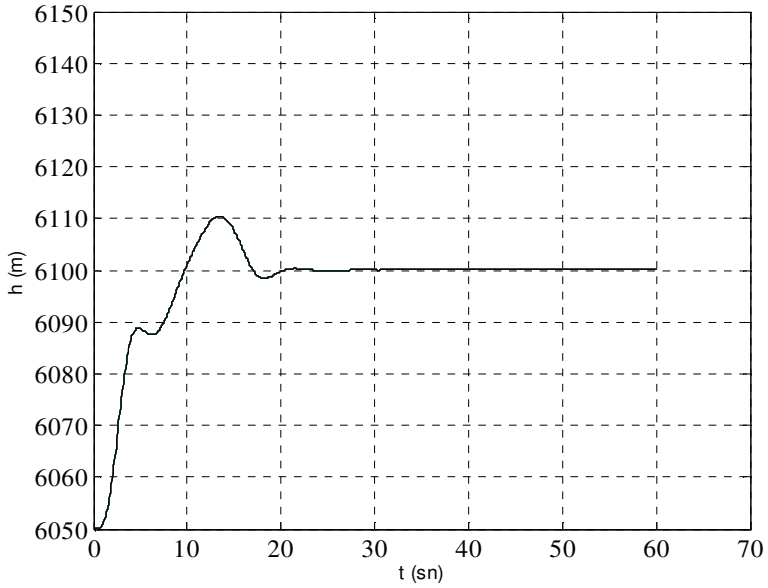
Şekil 4.31. Örnek 1 için irtifa hızının zamana göre değişimi
(tüm jenerasyonlar)



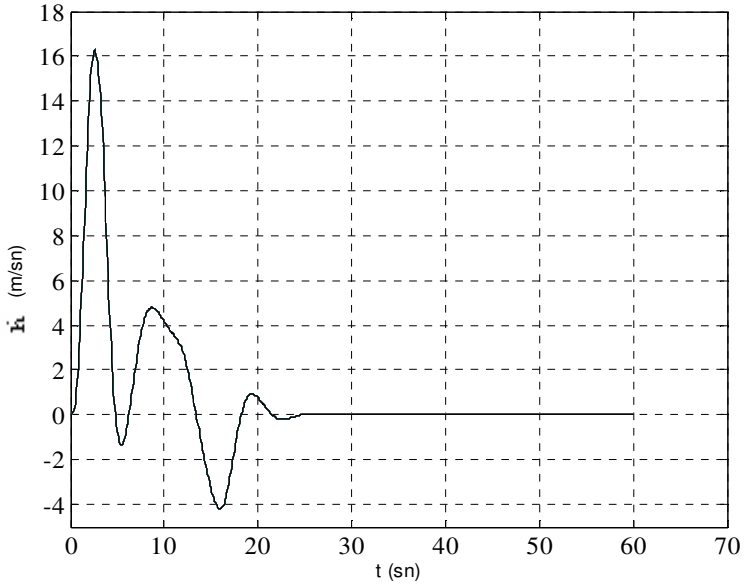
Şekil 4.32. Örnek 1 için hatanın zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 4.33. Örnek 1 için irtifa dümeni açısının zamana göre değişimi (son jenerasyondaki en iyi dizi)



Şekil 4.34. Örnek 1 için irtifanın zamana göre değişimi (son jenerasyondaki en iyi dizi)



Şekil 4.35. Örnek 1 için irtifa hızının zamana göre değişimi
(son jenerasyondaki en iyi dizi)

1. uçuş durumu için elde edilen sonuçlar incelendiğinde, özellikle son jenerasyondaki en iyi dizinin kullanılması durumunda elde edilen tepkilerin oldukça düzgün, aşma miktarlarının oldukça düşük yatışkın duruma ulaşma zamanlarının da oldukça kısa olduğu görülmektedir. Yine aynı şekilde tüm jenerasyonlardaki en iyi dizilere ait elde edilen tepkilerin de son jenerasyondaki en iyi dizi tepkisine yakın olduğu görülmüştür.

Örnek 2: 2.Uçuş durumu

Nüfus sayısı: 60

Jenerasyon sayısı: 10

Hassasiyet: 0.1

Çaprazlama oranı: 0.6

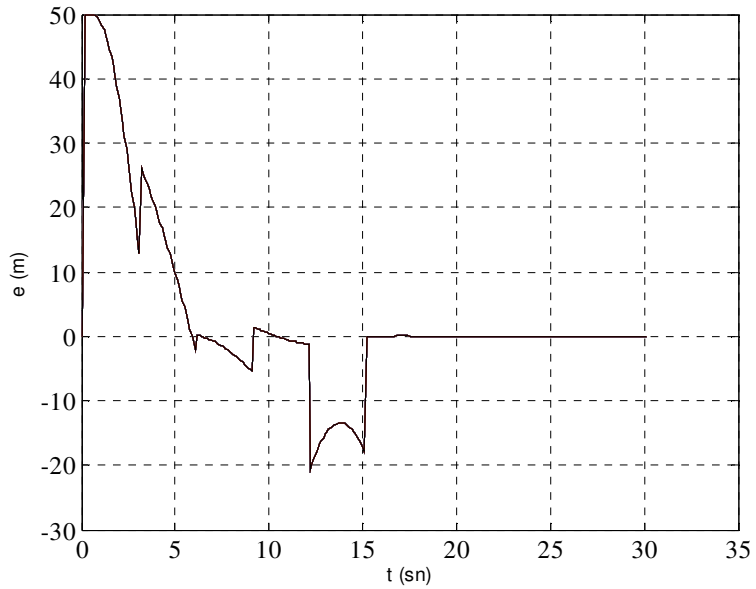
Mutasyon oranı: 0.002

Çalışma süresi: 3 sn

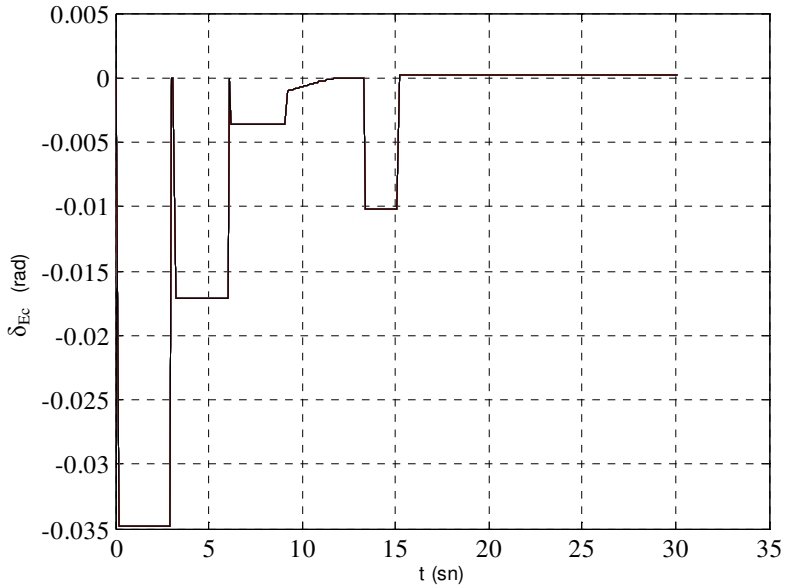
Runge-Kutta integrasyon aralığı: 0.1

Üyelik fonksiyonları sınır değerleri: ±50 m

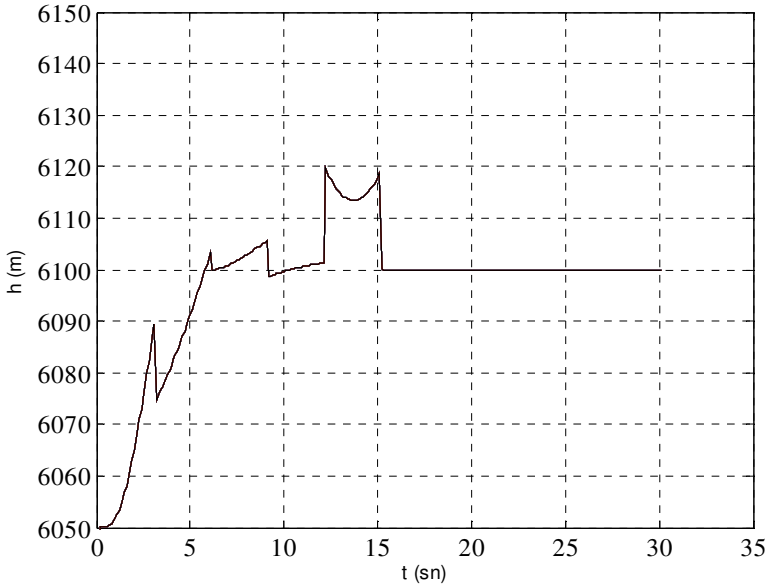
İrtifa dümeni açısı sınır değerleri: ±2°



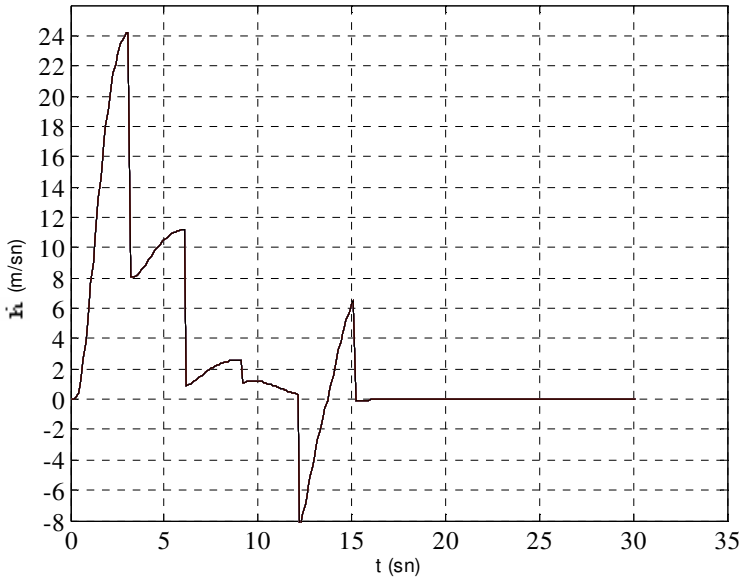
Şekil 4.36. Örnek 2 için hatanın zamana göre değişimi
(tüm jenerasyonlar)



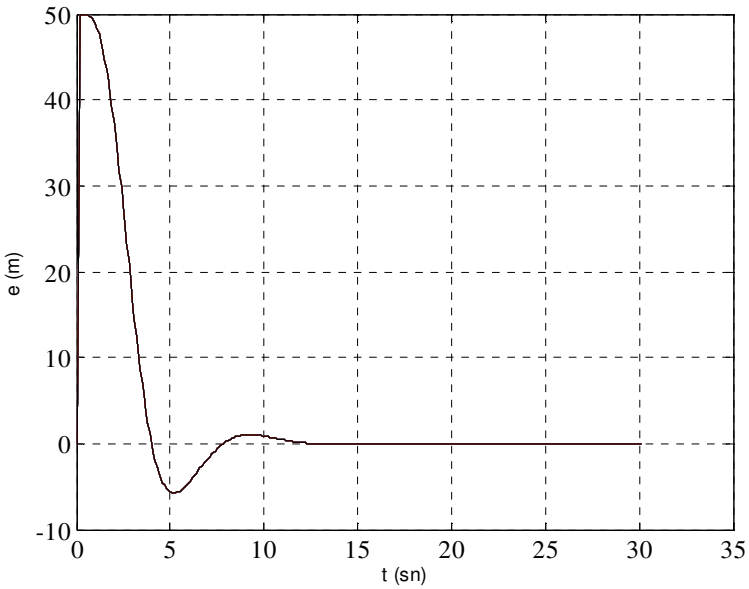
Şekil 4.37. Örnek 2 için **i**rtifa dümeni açısının zamana göre değişimi (tüm jenerasyonlar)



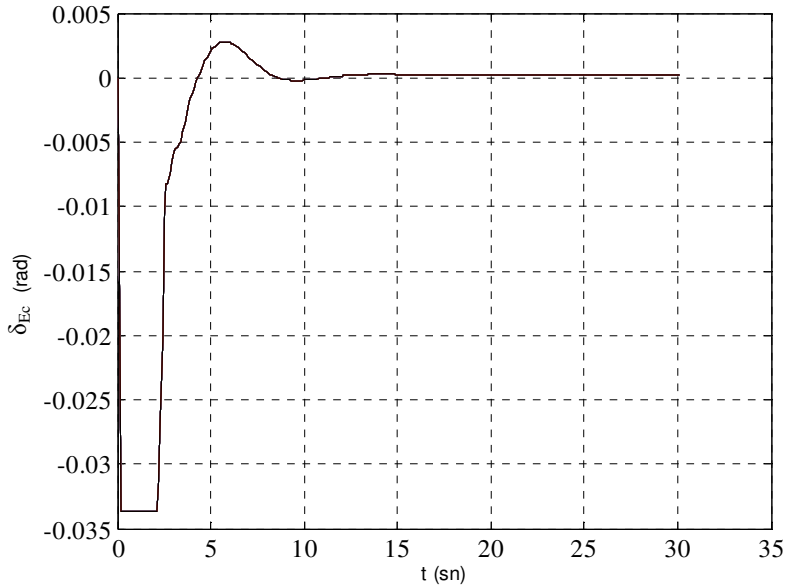
Şekil 4.38. Örnek 2 için irtifanın zamana göre değişimi (tüm jenerasyonlar)



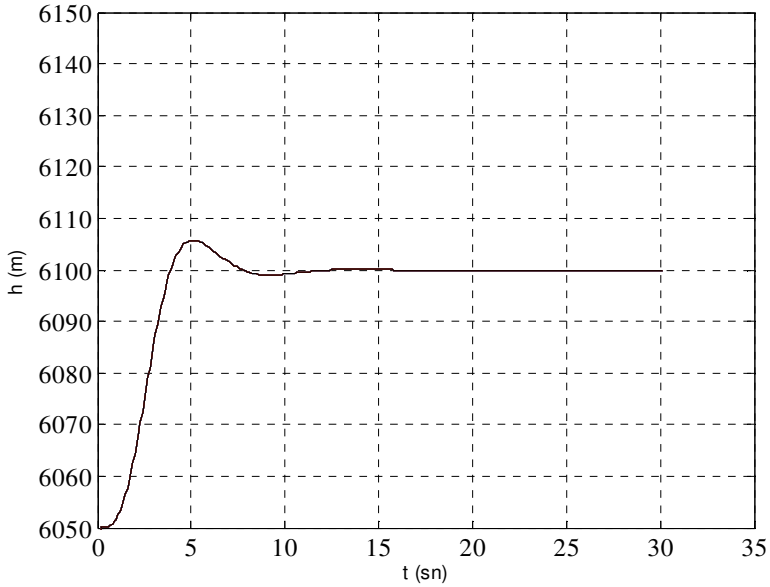
Şekil 4.39. Örnek 2 için irtifa hızının zamana göre değişimi
(tüm jenerasyonlar)



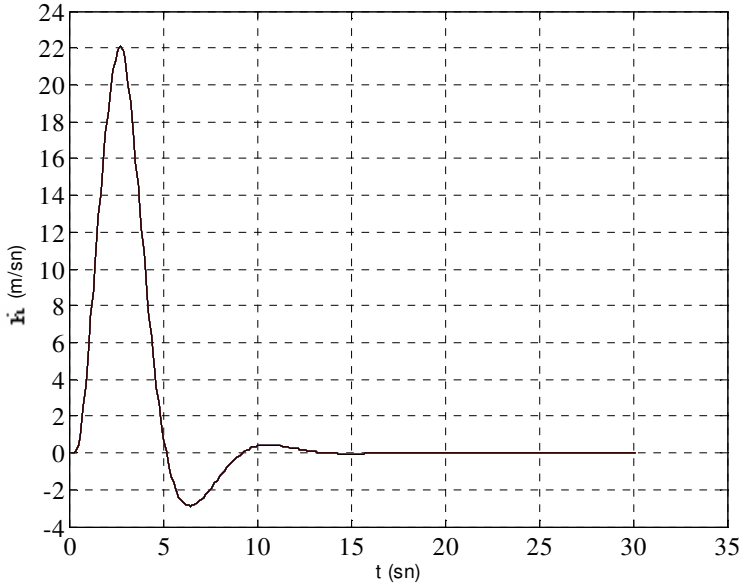
Şekil 4.40. Örnek 2 için hatanın zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 4.41. Örnek 2 için irtifa dümeni açısının zamana göre değişimi (son jenerasyondaki en iyi dizi)



Şekil 4.42. Örnek 2 için irtifanın zamana göre değişimi (son jenerasyondaki en iyi dizi)



Şekil 4.43. Örnek 2 için irtifa hızının zamana göre değişimi
(son jenerasyondaki en iyi dizi)

2. uçuş durumu için verilen parametreler ve elde edilen tepkiler incelendiğinde, 1. uçuş durumuna göre daha düşük popülasyon ve jenerasyon sayısı kullanılmasına rağmen oldukça iyi sonuçlar elde edildiği görülmektedir.

Örnek 3: 3.Uçuş durumu

Nüfus sayısı: 60

Jenerasyon sayısı: 30

Hassasiyet: 0.1

Çaprazlama oranı: 0.6

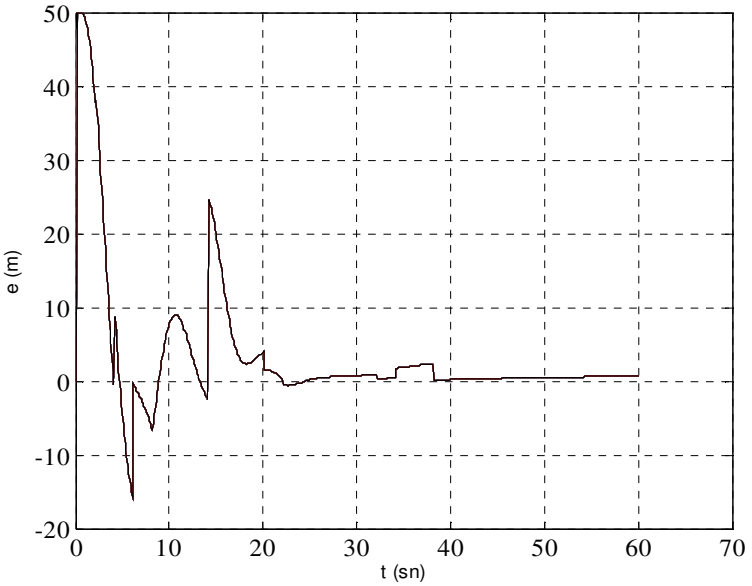
Mutasyon oranı: 0.002

Çalışma süresi: 2 sn

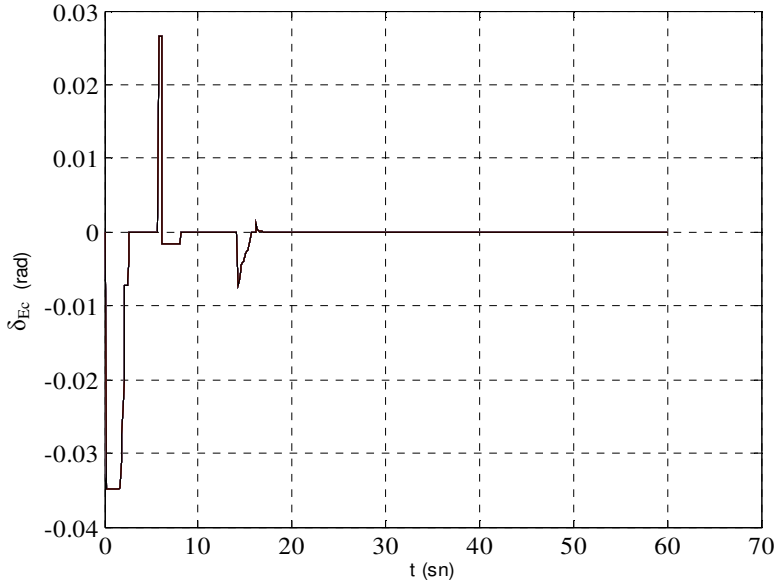
Runge-Kutta integrasyon aralığı: 0.1

Üyelik fonksiyonları sınır değerleri: ±50 m

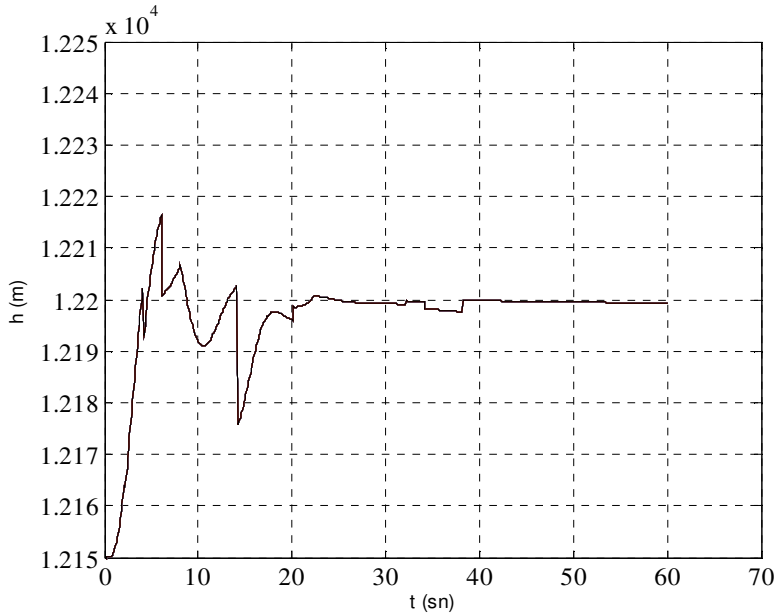
İrtifa dümeni açısı sınır değerleri: ±2°



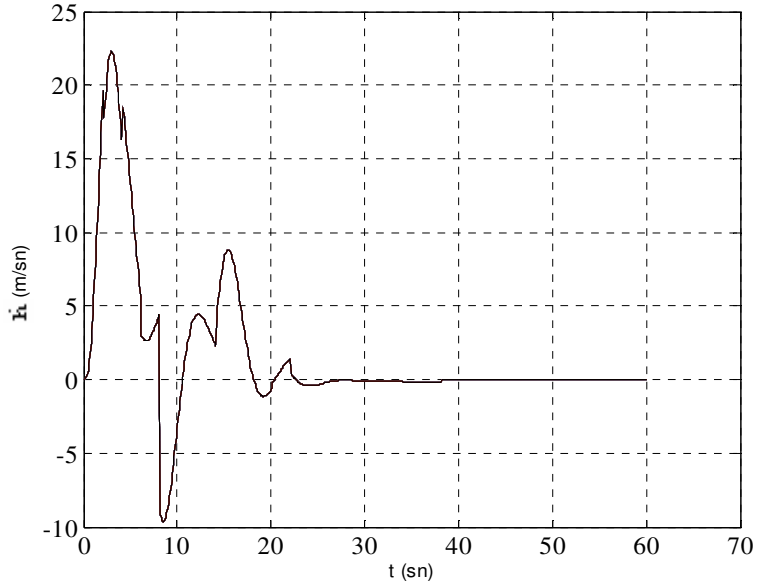
Şekil 4.44. Örnek 3 için hatanın zamana göre değişimi (tüm jenerasyonlar)



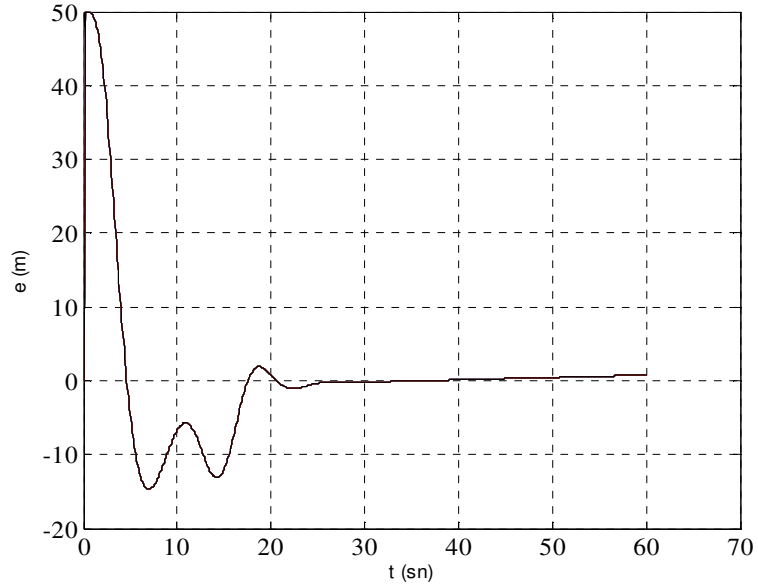
Şekil 4.45. Örnek 3 için irtifa dümeni açısının zamana göre değişimi (tüm jenerasyonlar)



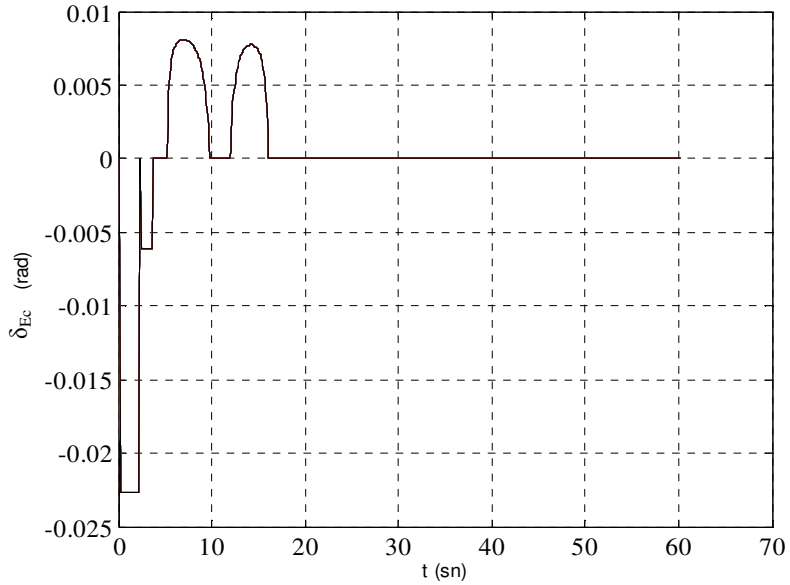
Şekil 4.46. Örnek 3 için irtifanın zamana göre değişimi (tüm jenerasyonlar)



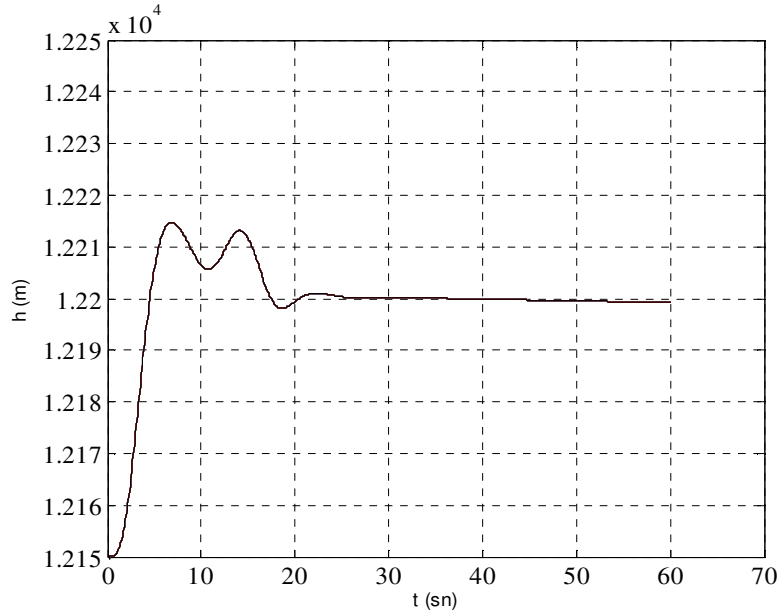
Şekil 4.47. Örnek 3 için irtifa hızının zamana göre değişimi
(tüm jenerasyonlar)



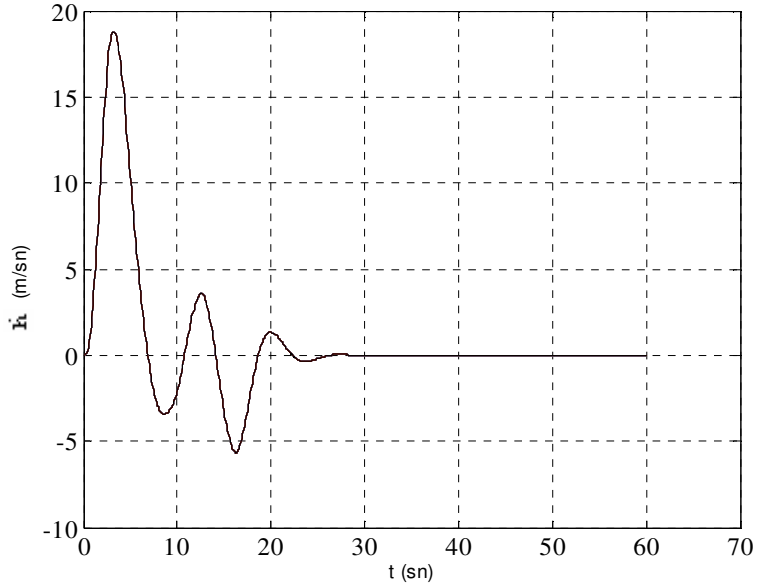
Şekil 4.48. Örnek 3 için hatanın zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 4.49. Örnek 3 için irtifa dümeni açısının zamana göre değişimi (son jenerasyondaki en iyi dizi)



Şekil 4.50. Örnek 3 için irtifanın zamana göre değişimi (son jenerasyondaki en iyi dizi)



Şekil 4.51. Örnek 3 için irtifa hızının zamana göre değişimi
(son jenerasyondaki en iyi dizi)

3. uçuş durumu için verilen örnekten de görüleceği üzere elde edilen tepkiler oldukça düzgün ve yatışkın duruma ulaşma zamanının oldukça kısa olduğu görülmektedir. Üç farklı uçuş durumu için verilen örnekler incelendiğinde popülasyon sayısının 60 ila 100, jenerasyon sayısının 10 ila 80, çaprazlama oranının 0.6 ve mutasyon oranının 0.002 seçilmesi durumunda oldukça iyi sonuçlar elde edildiği görülmektedir. Seçilen irtifa dümeni açıları ise üç uçuş durumu için farklı değerlerde (1 uçuş durumu için ±3°, 2.uçuş durumu için ±2°, 3. uçuş durumu için ±2°) seçilmiştir. İncelenen örnek uçağın irtifa dümeni açısı ±20° arasında değiştirilebilmektedir. İrtifa dümeni açısı uçağın ne kadar burun yukarı/burun aşağı hareket yapacağını değil ne hızla burun yukarı/burun aşağı yapacağını belirler ki buna da tırmanma (yunuslama) oranı adı verilir. Uçağın tırmanma açısını belirlemek ise irtifa dümenine verilen kumandanın ne kadar o vaziyette tutulduğu ile ilgilidir. Örneğin; pistte koşmakta olan bir uçak çoğunlukla 15 derece burun yukarı yaparak tırmanır. Ancak lövye (irtifa dümeni kumanda kolu) az geri çekilirse yani irtifa dümenine az açıda kumanda verilirse uçağın 15 derece tırmanma açısına ulaşması 15 saniye alıyorsa, lövyeye verilen yüksek

açıdaki bir kumandada bu hareket 2 saniyede gerçekleşir. Bu değer uçağın uçuş durumuna da bağlıdır. Dolayısıyla farklı uçuş durumları için istenen tepkiyi veren uygun irtifa dümeni açı değerleri de birbirinden farklı olmaktadır.

4.5. Klasik PD ve Genetik Algoritma Tabanlı Bulanık PD Denetleyici Uygulamalarının Karşılaştırması

Bölüm 4.4.3 ve Ek 1'de verilen örnekler incelendiğinde genetik bulanık PD tip denetleyici kullanılması durumunda elde edilen tepkilerin her üç uçuş durumu için de klasik PD denetleyici kullanıldığında elde edilen tepkilere oldukça yakın sonuçlar verdiği görülmektedir. Ayrıca genetik bulanık PD kontrolün uygulandığı tepkilerde (özellikle son jenerasyondaki en iyi dizi kullanılması durumunda) aşma miktarının klasik PD kontrol uygulamasına göre daha düşük olduğu ve tepkilerinde daha düzgün olduğu gözlenmektedir. Şekil 4.34'de 1.uçuş durumunda, irtifa genetik bulanık PD denetleyici durumunda (son jenerasyondaki en iyi dizi kullanılması durumu) maksimum 6110 m değerini alırken, klasik PD denetleyici durumunda Şekil 4.6'da görüleceği üzere 6115 m değerini almaktadır. Yine aynı durum için 1.uçuş durumunda yatışkın duruma ulaşma zamanı genetik bulanık PD denetleyici durumunda yaklaşık olarak 23 sn iken, klasik PD denetleyici durumunda yaklaşık 35 sn değerini almaktadır.

Klasik PD tip denetleyici kullanılması durumunda farklı K_p ve K_d değerleri denenmiş ve ilk iki uçuş durumu için en iyi çıkış tepkisine $K_p=-0{,}0008$ ve $K_d=-0{,}0005$ seçildiğinde ulaşıldığı görülmüştür. Üçüncü uçuş durumunda daha iyi çıkış tepkisi ise $K_p=-0{,}0002$ ve $K_d=-0{,}0004$ değerleri seçildiğinde elde edilmiştir. Bu durumda elde edilen çıkış tepkisinin frekansının düşük, yatışkın duruma ulaşma zamanının kısa ve daha düzgün olduğu görülmüştür.

Dolayısıyla klasik PD denetleyici kullanılması durumunda, herhangi bir uçuş durumu için tasarlanmış denetleyici sistemin farklı bir uçuş durumunda aynı performansı sağlamadığı sonucuna ulaşılmıştır.

3. uçuş durumunda, farklı bir klasik PD denetleyici ($K_p=-0.0002$ ve $K_d=-0.0004$) kullanılarak genetik bulanık denetleyici ile benzer sonuçlara ulaşılabilmiştir. Buna rağmen genetik bulanık PD denetleyici durumunda (son jenerasyondaki en iyi dizi kullanılması durumu) Şekil 4.50'de görüldüğü gibi

yatışkın duruma ulaşma zamanı 35 sn iken, klasik PD denetleyici durumunda Şekil 4.18'de görüldüğü gibi bu değerin 60 sn olduğu görülmüştür.

Sonuç olarak, bir uçuş durumu için tasarlanan klasik PD denetleyici, farklı uçuş durumları için aynı performans karakteristiklerini vermezken, genetik bulanık PD denetleyici tüm uçuş durumları için daha iyi çıkış karakteristikleri vermektedir.

5. SONUÇ VE ÖNERİLER

Bu çalışmada uçuş kontrol sisteminde önemli bir yer tutan irtifa kontrolünün genetik bulanık PD denetleyici kullanılarak tasarımı incelenmiş ve geniş gövdeli, dört motorlu jet yolcu uçağına ait elde edilen uygulama sonuçları, klasik PD denetleyici kullanılarak elde edilen tepkilerle birlikte değerlendirilmiştir.

Bulanık denetleyici sistem tasarımında en önemli süreç sisteme uygun bilgi tabanının oluşturulması sürecidir. Kontrol edilecek sistem hakkında yeterli ve doğru bilgilerin elde olduğu durumlarda bilgi tabanının oluşturulmasında kişisel sezgi, mantık ve tecrübelerin kullanılmasına sıkça rastlanır. Fakat bazı durumlarda yeterli derecede bilgi elde etmek mümkün olmamaktadır. Bu durumda sisteme uygun bilgi tabanının (uygun üyelik fonksiyonlarının ve kural ağırlık değerlerinin) oluşturulmasında kullanılan en etkili yöntemlerden birisi de genetik algoritmalardır. Bu çalışmada tasarlanan bulanık denetleyici sistemde üyelik fonksiyonları sınır değerleri ve kural ağırlık değerleri genetik algoritma yardımıyla elde edilmiştir.

Genetik bulanık denetleyici kullanılması durumunda, belli bir jenerasyon sonunda minimum hatayı veren dizi değerlerinden elde edilen üyelik fonksiyonları sınır değerleri ve kural ağırlık değerleri ile çalıştırılan bulanık denetleyici oldukça iyi çıkış tepkileri vermektedir. Ayrıca son jenerasyondaki en iyi dizi elde edilinceye kadar süre kaybı olmaksızın her bir jenerasyondaki minimum hatayı veren dizi değerlerinden elde edilen üyelik fonksiyonları sınır değerleri ve kural ağırlık değerleri de sisteme aynı anda (online olarak) verilerek oldukça iyi sonuçlar elde edilmiştir.

Sonuç olarak, bir uçuş durumu için tasarlanan klasik PD denetleyici, farklı uçuş durumları için aynı performans karakteristiklerini vermezken, genetik bulanık PD denetleyici tüm uçuş durumları için daha iyi çıkış karakteristikleri vermektedir.

Ayrıca, bu çalışma genetik algoritma içinde birden fazla performans kriterini aynı anda içeren uygunluk fonksiyonları seçilerek daha fazla parametrenin en iyilemesini gerçekleştiren denetleyici tasarımı için geliştirilebilir.

KAYNAKLAR

[1] Kahvecioğlu, A., *Uçuş kontrol sistem tasarımında katlı- model yaklaşımı ve genetik algoritma tekniğinin uygulanması*, Doktora Tezi, Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Eskişehir, 2000.

[2] Verbruggen, H.B., Zimmerman H.J., Babuska, R., *Fuzzy algorithms for control*, Kluwer Academic Publishers, Massachusetts, U.S.A.,1999.

[3] Aström, K. J., WITTENMARK, B., *Adaptive control*, Lund Institute of Technology, Addison Wesley Publishing Company, New York, U.S.A.,1989.

[4] Anonim, *Experience with the X–15 adaptive flight control system*, NASA Technical Note, Washington, 2006.
http://www.nasa.gov/centers/dryden/pdf/87785main_H-618.pdf

[5] Enns, D.F., Robustness of dynamic inversion vs. Mi synthesis: Lateral-directional flight control example*, Proceedings of the AIAA Conference on Guidance, Navigation, and Control*, Portland, Oregon, 1990.

[6] Hayes, J**.** R., Kureemu and Bates, D.G., "LFT-based uncertainty modelling and mu-analysis of the HIRM+RIDE flight control law", *Proc. of the IEEE International Symposium on Computer Aided Control Systems Design* (CACSD'02), Glasgow, 242–247, 2002.

[7] http://www.control.eng.cam.ac.uk/gp202/invsess/sess99.out.html(Ekim 2005)

[8] Nichols, R.A., Reichert, R.T., Rugh, W.J., Gain scheduling for H-infinity controllers: A flight control example, IEEE Transactions on Control Systems Technology, **1(2)**, 69-79, 1993.

[9] http://www.op.dlr.de/FF-DR-ER/research /flight /flight.html (Ekim 2005)

[10] http://www.dcsc.tudelft.nl/Research/Current/project_mo_rb.html (Ekim 2005)

[11] http://www.kbs.twi.tudelft.nl/Publications/Report/2004-Liang-DKS04-04.html (Ekim 2005)

[12] http://www.ndengineering.com/otherprojects.html (Kasım 2005)

[13] Magni, J. F., Multimodel eigenstructure assignment in flight control design, *Elseiver Science*, **3(3)**, April, 141-151, 1999.

[14] Ackermann, J., Multi-Model Approaches to Robust Control System Design, *Lecture Notes in Control and Information Science*, **(70)** Springer Verlag , Berlin,1985.

[15] http://www2.ing,unipi.it/~dia/research/flight. htm (Kasım 2005)

[16] Kıyak, E., *Bulanık mantık ve uçuş kontrol uygulamaları*, Yüksek Lisans Tezi, Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Eskişehir, 2003.

[17] Rotton, S. K., Brehm, T., Sandhu, G. S., *Analysis and Design of a Proportional Fuzzy Logic Controller*, Department of Electrical Engineering Wright State University, Dayton, Ohio, 2006. http://www.iugaza.edu/Staff/StaffWeb/Files/844/docs/papers/pflc _design.pdf (Ekim 2005)

[18] Şen, Z., *Bulanık (fuzzy) mantık ve modelleme ilkeleri*, Bilge Sanat Yapım Yayınları, İstanbul., 2001.

[19] Güney, Z., *Eğitim amaçlı bulanık mantık denetleyicisi tasarımı,* Yüksek Lisans Tezi, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 2002.

[20] Pehlivan, İ., *Bulanık mantık kontrolörler ile klasik PID kontrolörlerin karşılaştırılması ve bir bulanık mantık kontrolör tasarımı*, Yüksek Lisans Tezi, Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Sakarya, 2001.

[21] Elmas, Ç., *Bulanık mantık denetleyiciler (kuram, uygulama, sinirsel bulanık mantık)*, Seçkin yayınları, Ankara., 2003.

[22] Direkçi, S., *Bulanık mantık kontrol sistemleri*, Yüksek Lisans Tezi, Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 2002.

[23] http://www.ercb.com/ddj/1993/ddj.9306.html (Eylül 2006)

[24] Topuz, V., *Bulanık genetik proses kontrolü*, Doktora Tezi, Marmara Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2002.

[25] Yılmaz, M., Arslan E., Bulanık mantığın jeodezik problemlerin çözümünde kullanılması, *Harita ve Kadastro Mühendisleri Odası, Mühendislik Ölçmeleri STB Komisyonu 2. Mühendislik Ölçmeleri Sempozyumu*, İstanbul, 2005. www.hkmo.org.tr/resimler/ekler/a68c75c4a77c87f_ek.pdf

[26] Chen, G., Pham, T. T., *Introduction to Fuzzy Sets, Fuzzy Logic and Fuzzy control Systems*, CRC Pres LLC, Florida, U.S.A., 2001.

[27] ŞEN, Z., Genetik algoritmalar ve en iyileme yöntemleri, Su Vakfı Yayınları, İstanbul., 2004.

[28] Goldberg, D.E., *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley Publishing Company, Inc, New York, U.S.A., 1989.

[29] KURT, M., SEMETAY, C., *Genetik algoritma ve uygulama alanları*, Marmara Üniversitesi Teknik Eğitim Bölümü, İstanbul. http://www.mmo.org.tr/muhendismakina/arsiv/2001/ekim/Genetik_Algoritma.htm (Temmuz 2006)

[30] Cordon, O., Gamide, F., Herrera, F., Magdelen, H.L., Ten Years of Genetic Fuzzy Systems Current Framework and New trends, *Fuzzy Sets and Systems*, **141(1)**, January, 5-31, 2004.

[31] Arslan, A., Kaya, M., Determination of fuzzy logic membership functions using genetic algorithms, *Fuzzy Sets and Systems*, **118(2)**, March , 297-306, 2001.

[32] Gürocak, H., B., A genetic-algorithm-based method for tuning fuzzy logic controllers, *Fuzzy Sets and Systems*, **108(1)**, November, 39-47, 1999.

[33] Homaifar. A., McCormick, E., Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, *IEEE transactions on fuzzy sysytems*, **3(2)**, May, 1995.

[34] Shi, Y., Eberhart, R., Chen, Y., Implementation of evolutionary fuzzy systems, *IEEE transactions on fuzzy sysytems*, **7(2)**, April, 1999.

[35] Hu, B., Mann, G.K.I., Gosine, R.G., New methodology for analytical and optimal design of fuzzy PID Controllers, *IEEE transactions on fuzzy sysytems*, **7(5)**, October, 1999.

[36] Zhou, Y.S., Lai, L.Y., Optimal design for fuzzy controllers by genetic algorithms, *IEEE transactions on industry applications*, **36(1)**, January/February, 2000.

[37] Karr, C.L., Gentry, E.J., Fuzzy control of pH using genetic algorithms, *IEEE transactions on fuzzy systems*, **1(1)**, February 1993.

[38] Chin, T.C., Qi, X., Genetic algorithms for learning the rule base of fuzzy logic controller, *Fuzzy Sets and Systems*, **97(1)**, 1-7, July, 1998.

[39] Jamshidi, M., Titli, A., Zadeh, L., Boverie, S., Applications of fuzzy logic towards high machine intelligence quotient systems, Prentice Hall PTR, New Jersey, 1997.

[40] Sanchez, E., Shibata, T., Zadeh, L.A., *Genetic algorithms and fuzzy logic systems, soft computing perspectives*, World Scientific Publishing Co. Pte. Ltd., Singapore., 1997.

[41] Livchitz, M., Abershitz, A., Soudak, U., Development of an automated fuzzy-logic-based expert system for an unmanned landing, *Fuzzy Sets and Systems,* **93(2)**, 16, 145-159, 1998.

[42] http://www.hf.faa.gov/docs/508/docs/cami/0207.pdf (Eylül 2006).

[43] YÜKSEL, İ., *Otomatik kontrol*, Uludağ üniversitesi Güçlendirme Vakfı yayını, No 47, Bursa., 2001.

[44] BİR, A., *Otomatik Kontrol Sistemleri*, Literatür Yayıncılık, İstanbul., 1999.

[45] McLEAN, D., *Automatic Flight Control Systems*. Prentice Hall International (UK) Ltd, New York, U.S.A., 1990.

[46] Winter, G., Periaux, J., Galan, M., Cuesta, P., *Genetic algorithms in engineering and computer science*, John Wiley&Sons Ltd., England., 1995.

[47] Nandi, A. K., Pratihar, D. K., Design of genetic fuzzy system to predict surface fnish and power requirement in grinding, *Fuzzy Sets and Systems*, **148(3)**, 487-504 , 2004.

[48] Kumar, M., Garg, D. P., Intelligent learning of fuzzy logic controllers via neural Network and genetic algorithm, *Proceedings of 2004 JUSFA, USA Symposium on Flexible Automation Denver*, Coloroda, 2004.

Ek–1: Farklı genetik algoritma parametreleri, çalışma süresi ve Runge-Kutta integrasyon aralığı değerleri için farklı uçuş durumlarında elde edilen $f(t,e), f(t,h), f(t,\delta_{E_c}), f(t,\dot{h})$ değişimleri.

Örnek 1:1.Uçuş durumu

Nüfus sayısı: 90

Jenerasyon sayısı: 40

Hassasiyet: 0.1

Çaprazlama oranı: 0.7

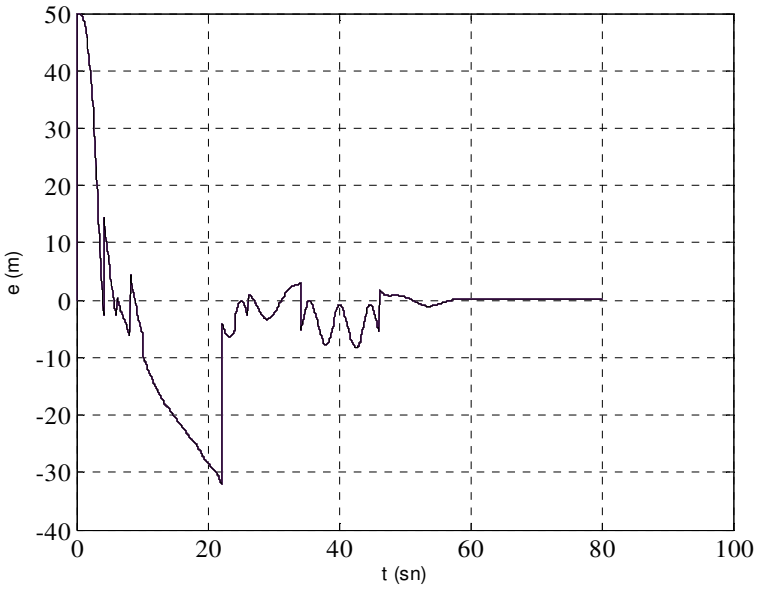
Mutasyon oranı: 0.002

Çalışma süresi: 2 sn

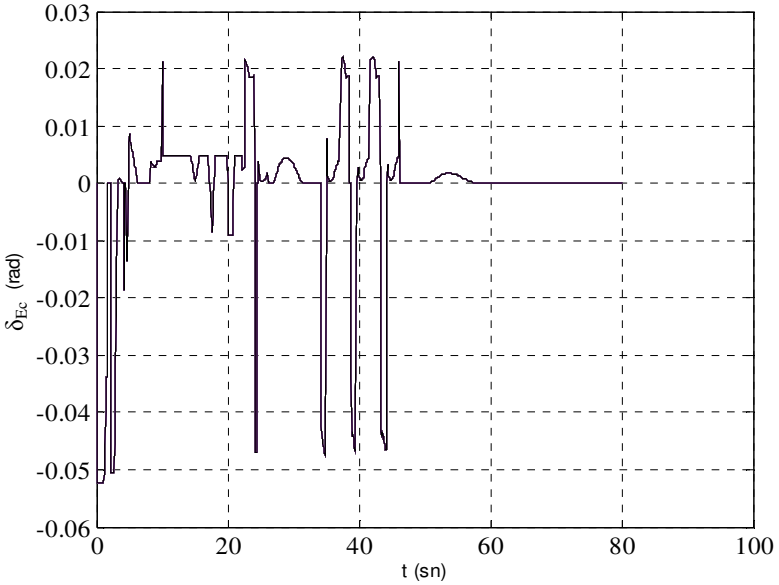
Runge-Kutta integrasyon aralığı: 0.1

Üyelik fonksiyonları sınır değerleri: ±50 m

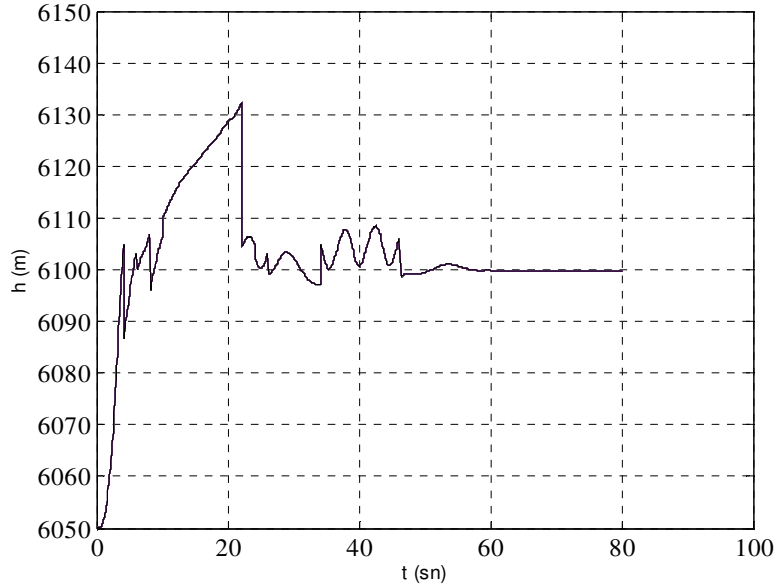
İrtifa dümeni açısı sınır değerleri: ±3°



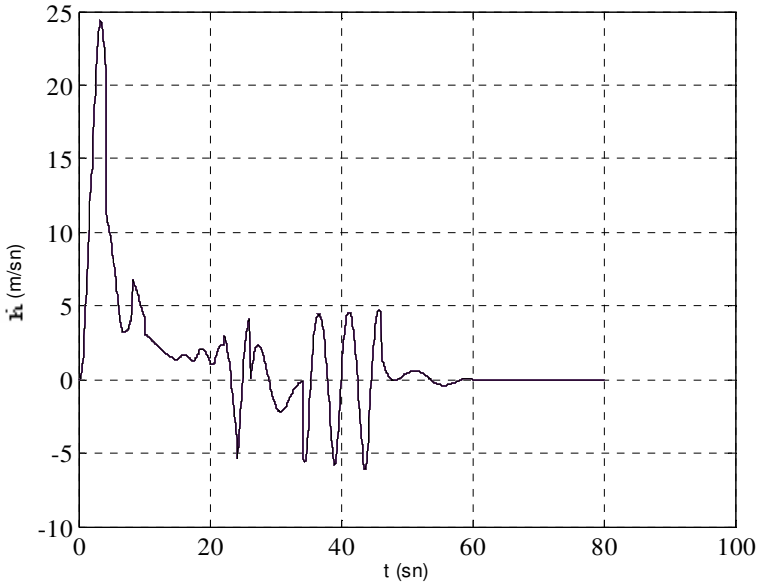
Şekil 1. Örnek1 için hatanın zamana göre değişimi
(tüm jenerasyonlar)



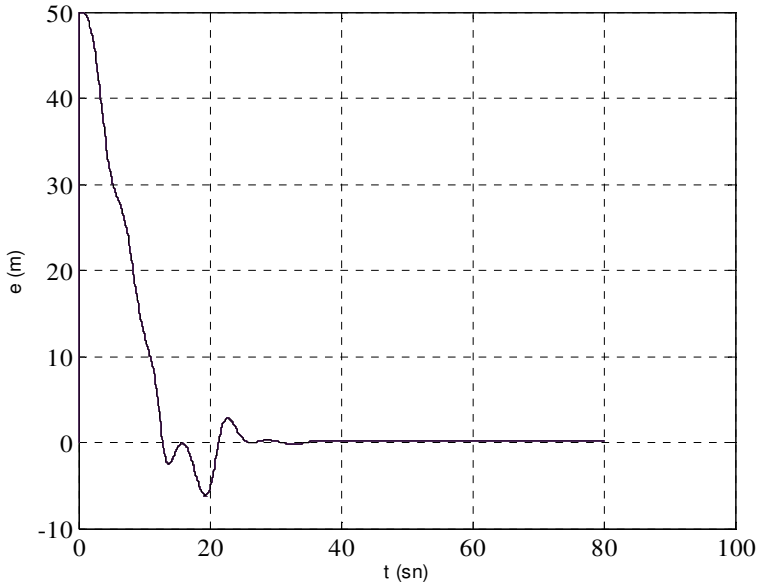
Şekil 2. Örnek1 için irtifa dümeni açısının zamana göre değişimi (tüm jenerasyonlar)



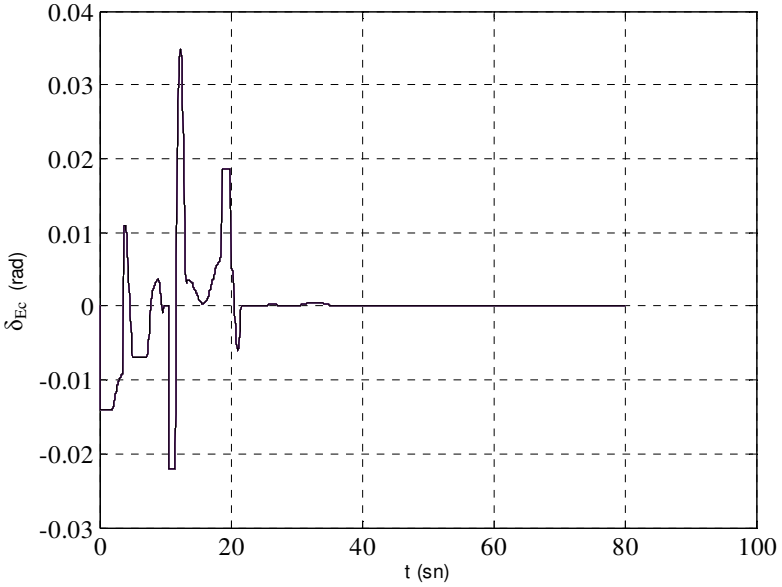
Şekil 3. Örnek1 için irtifanın zamana göre değişimi (tüm jenerasyonlar)



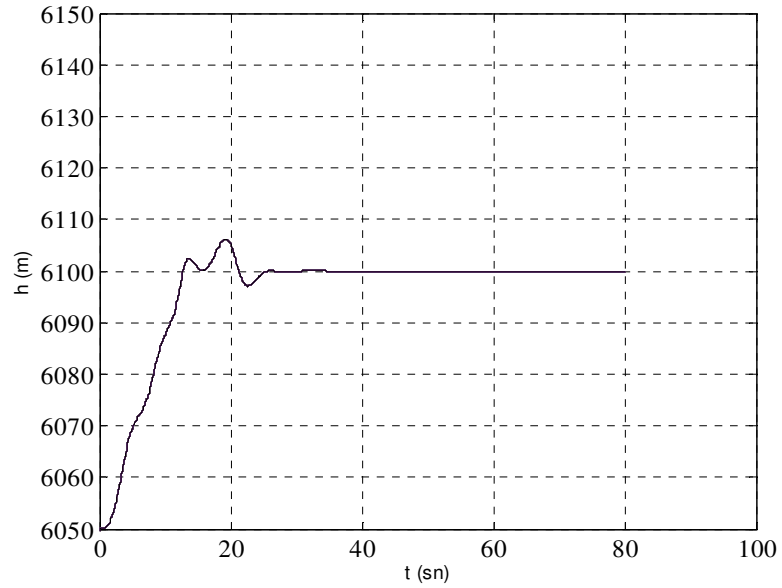
Şekil 4. Örnek1 için irtifa hızının zamana göre değişimi
(tüm jenerasyonlar)



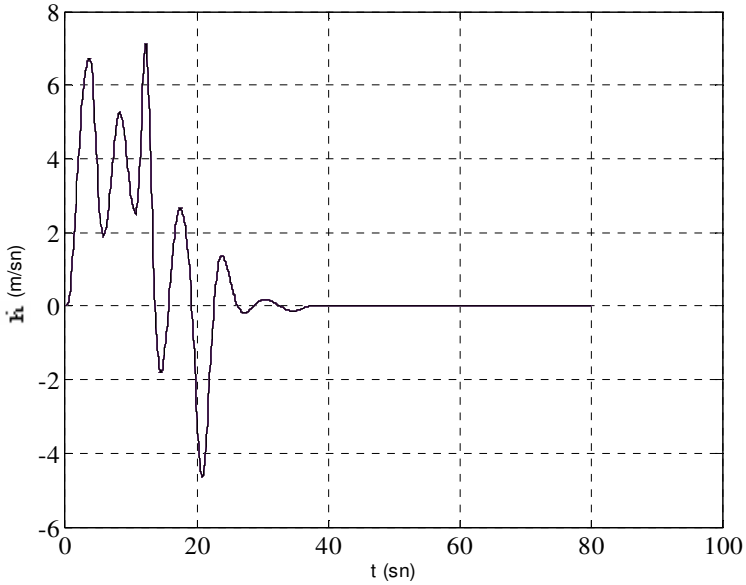
Şekil 5. Örnek1 için hatanın zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 6. Örnek1 için irtifa dümeni açısının zamana göre değişimi (son jenerasyondaki en iyi dizi)



Şekil 7. Örnek1 için irtifanın zamana göre değişimi (son jenerasyondaki en iyi dizi)



Şekil 8. Örnek1 için irtifa hızının zamana göre değişimi
(son jenerasyondaki en iyi dizi)

Örnek 2:1.Uçuş durumu

Nüfus sayısı: 50

Jenerasyon sayısı: 30

Hassasiyet: 0.1

Çaprazlama oranı: 0.6

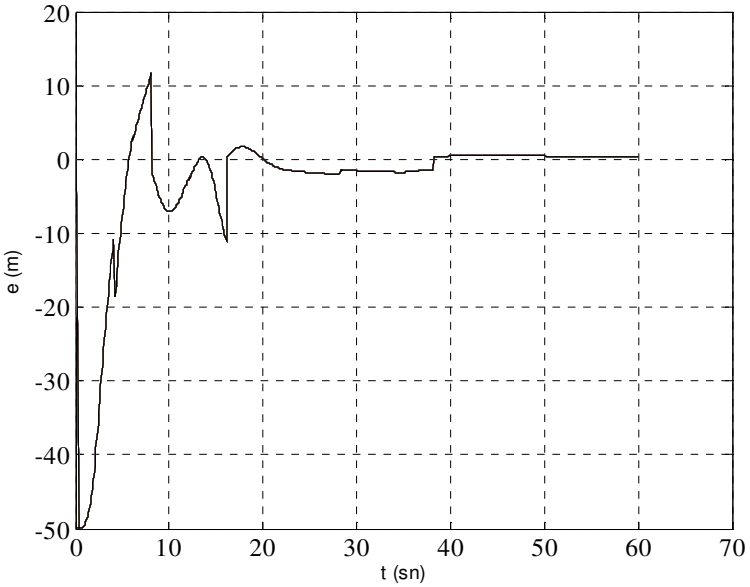
Mutasyon oranı: 0.002

Çalışma süresi: 2 sn

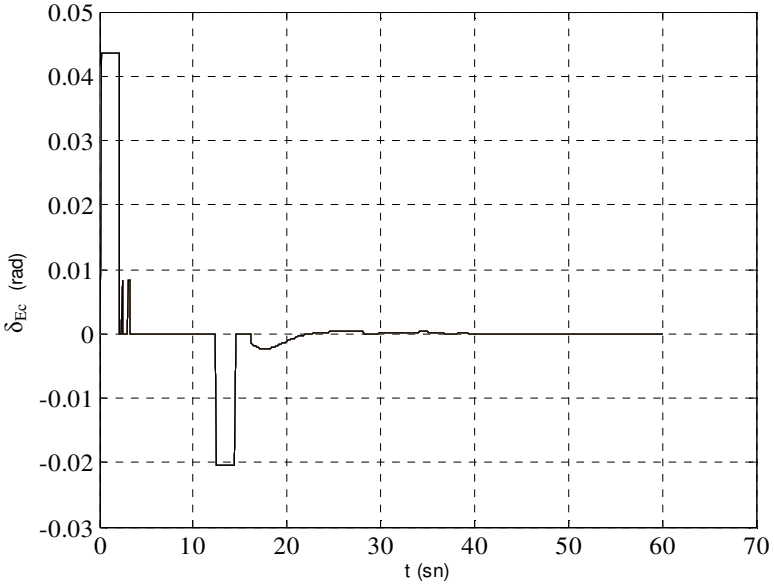
Runge-Kutta integrasyon aralığı: 0.1

Üyelik fonksiyonları sınır değerleri: ±50 m

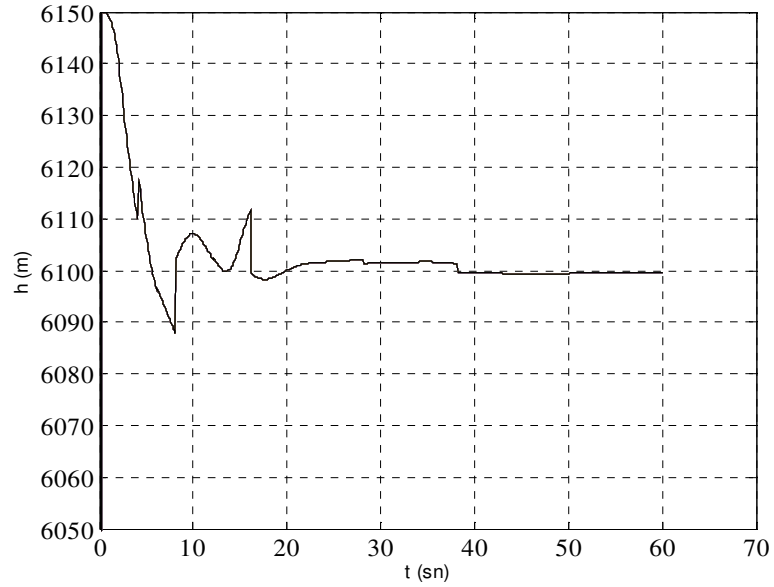
İrtifa dümeni açısı sınır değerleri: ±3°



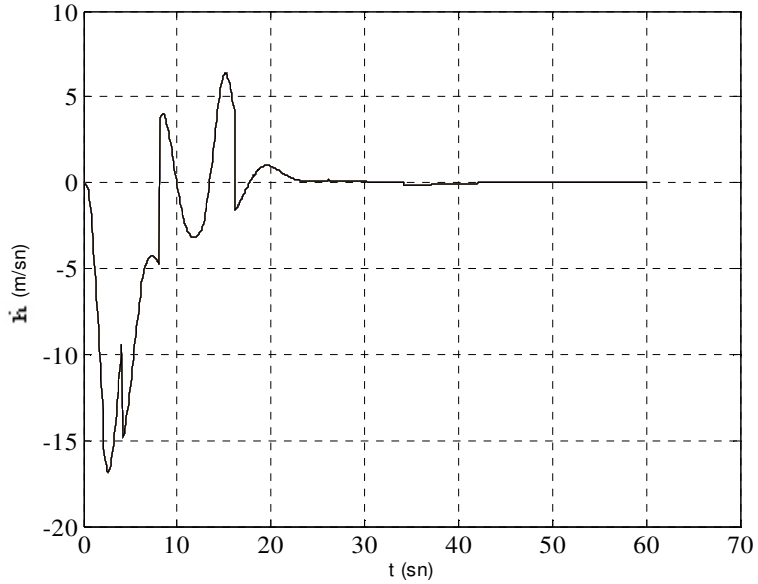
Şekil 9. Örnek 2 için hatanın zamana göre değişimi (tüm jenerasyonlar)



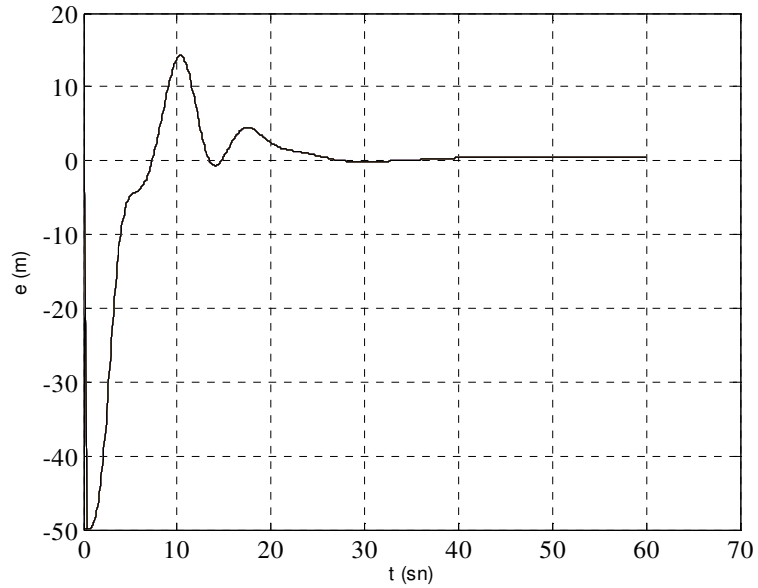
Şekil 10. Örnek 2 için irtifa dümeni açısının zamana göre değişimi (tüm jenerasyonlar)



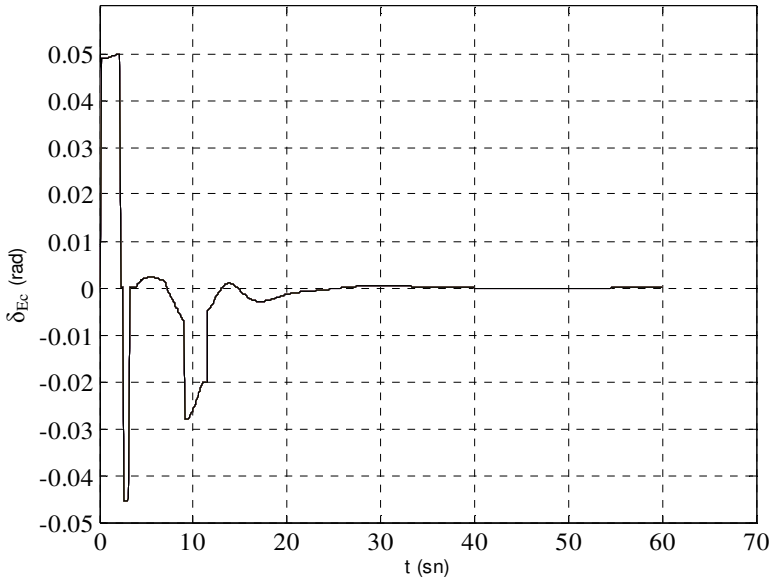
Şekil 11. Örnek 2 için irtifanın zamana göre değişimi (tüm jenerasyonlar)



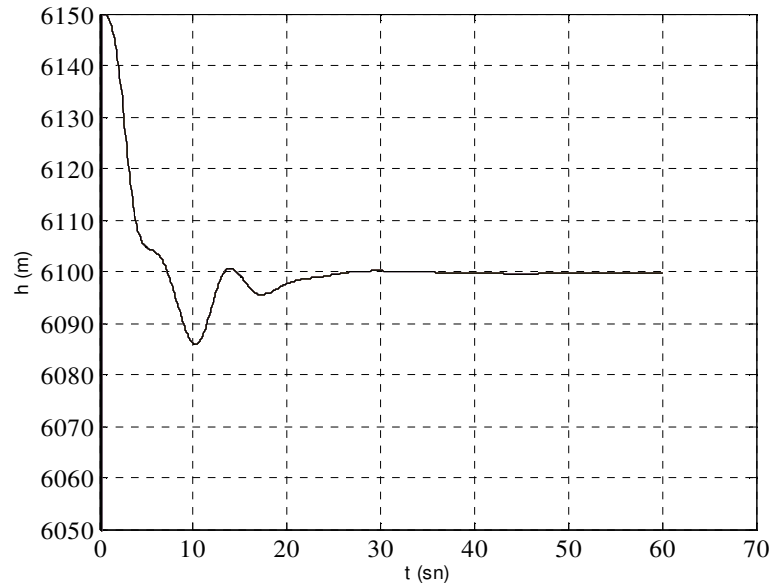
Şekil 12. Örnek 2 için irtifa hızının zamana göre değişimi
(tüm jenerasyonlar)



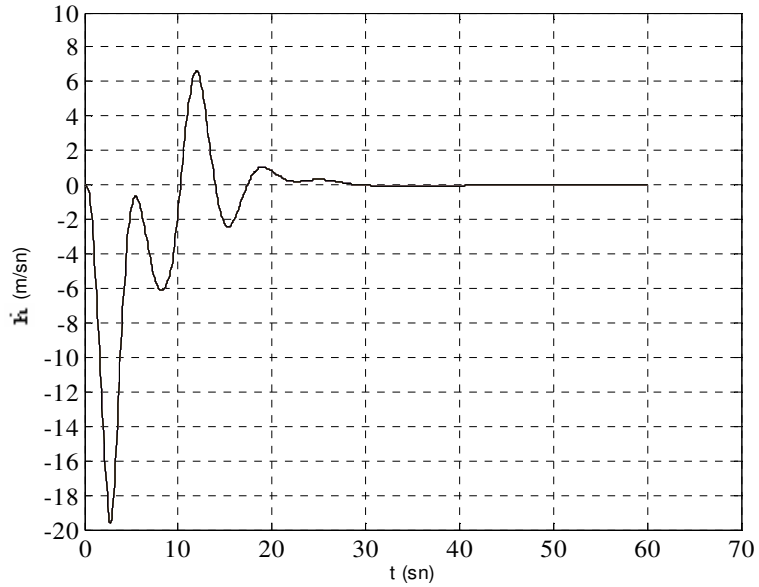
Şekil 13. Örnek 2 için hatanın zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 14. Örnek 2 için irtifa dümeni açısının zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 15. Örnek 2 için irtifanın zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 16. Örnek 2 için irtifa hızının zamana göre değişimi
(son jenerasyondaki en iyi dizi)

Örnek 3: 2.Uçuş durumu

Nüfus sayısı: 60

Jenerasyon sayısı: 40

Hassasiyet: 0.1

Çaprazlama oranı: 0.6

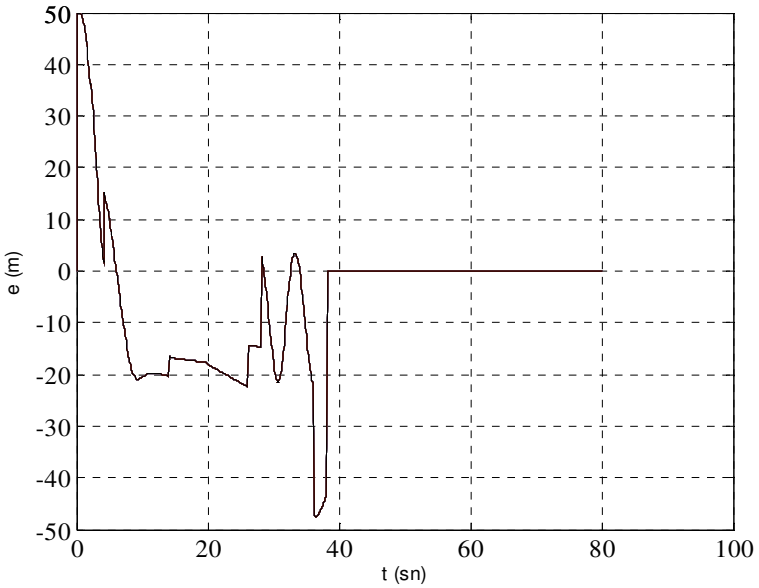
Mutasyon oranı: 0.002

Çalışma süresi: 2 sn

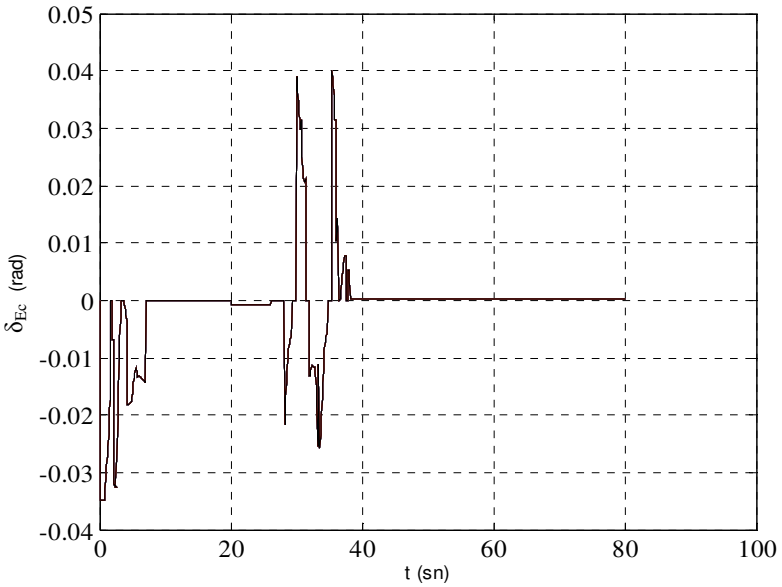
Runge-Kutta integrasyon aralığı: 0.1

Üyelik fonksiyonları sınır değerleri: ±50 m

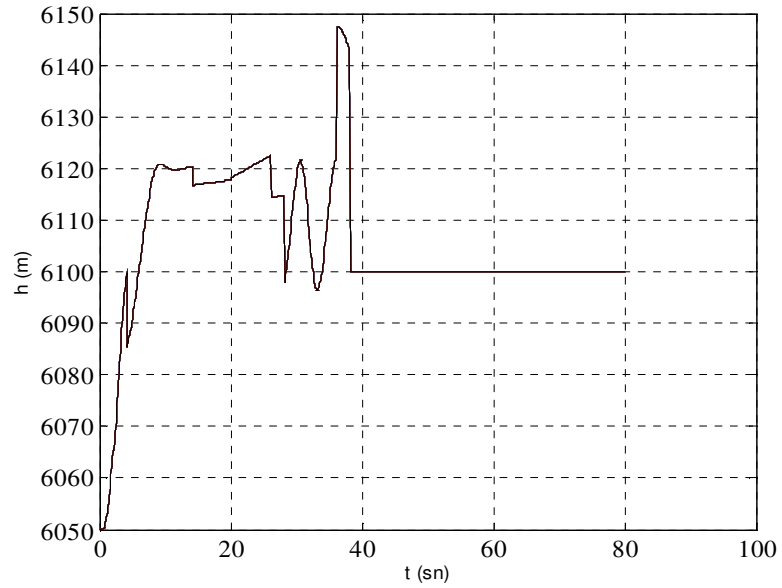
İrtifa dümeni açısı sınır değerleri: ±2°



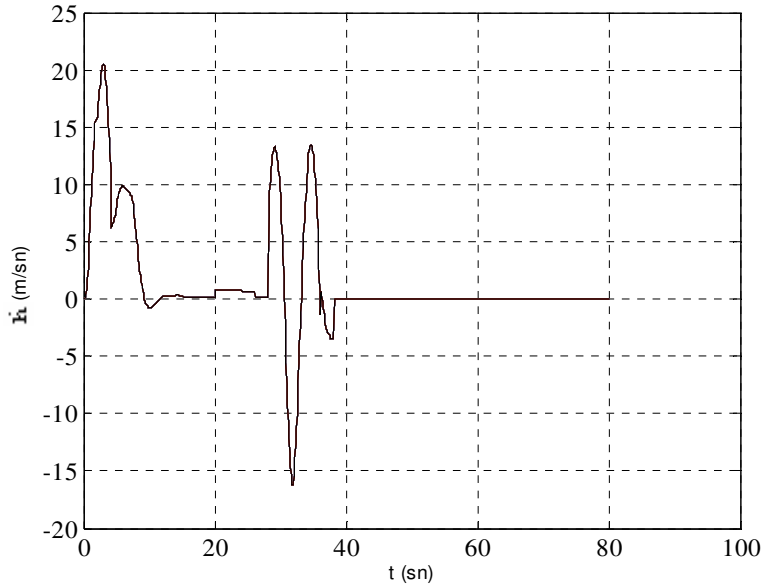
Şekil 17. Örnek 3 için hatanın zamana göre değişimi
(tüm jenerasyonlar)



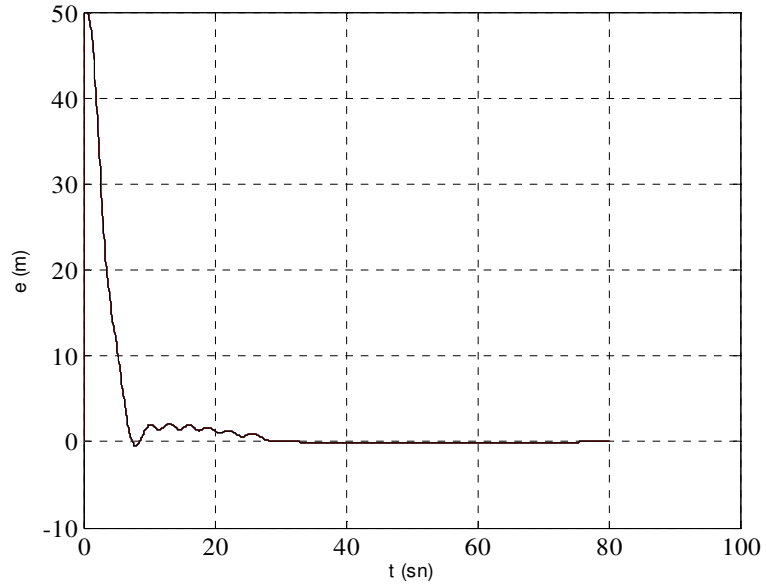
Şekil 18. Örnek 3 için irtifa dümeni açısının zamana göre değişimi (tüm jenerasyonlar)



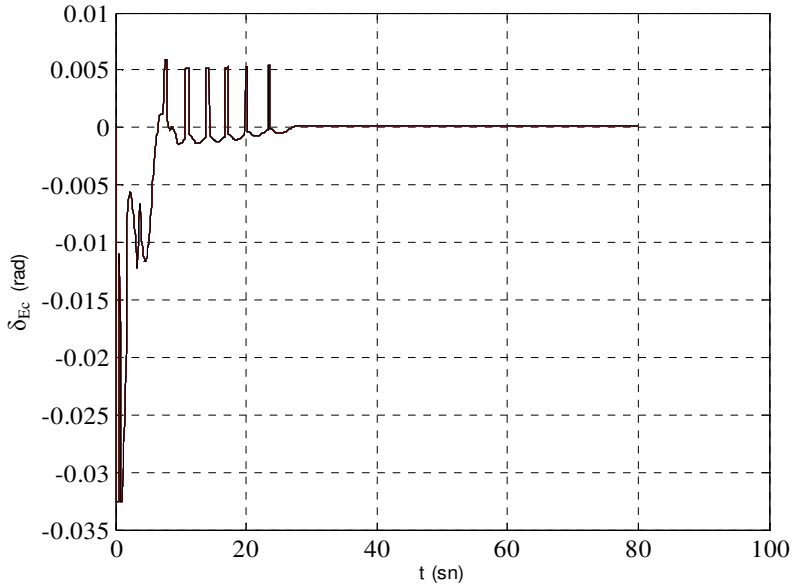
Şekil 19. Örnek 3 için irtifanın zamana göre değişimi (tüm jenerasyonlar)



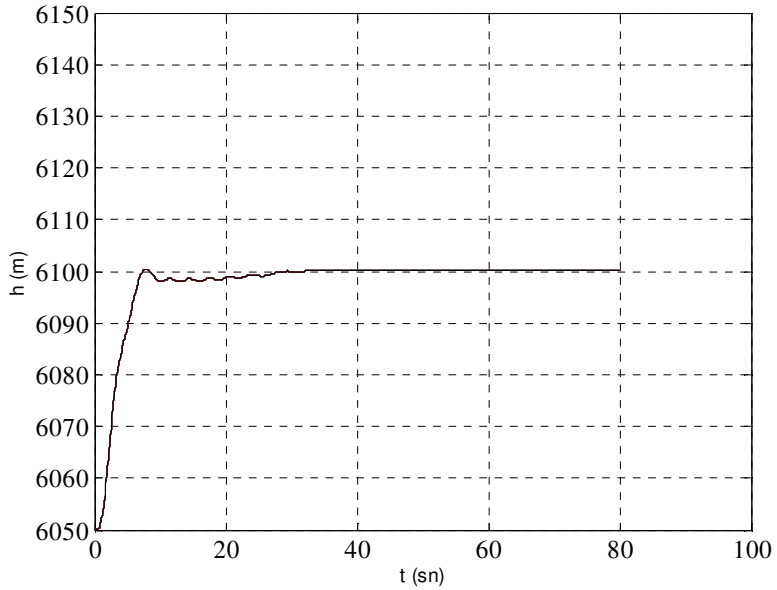
Şekil 20. Örnek 3 için irtifa hızının zamana göre değişimi
(tüm jenerasyonlar)



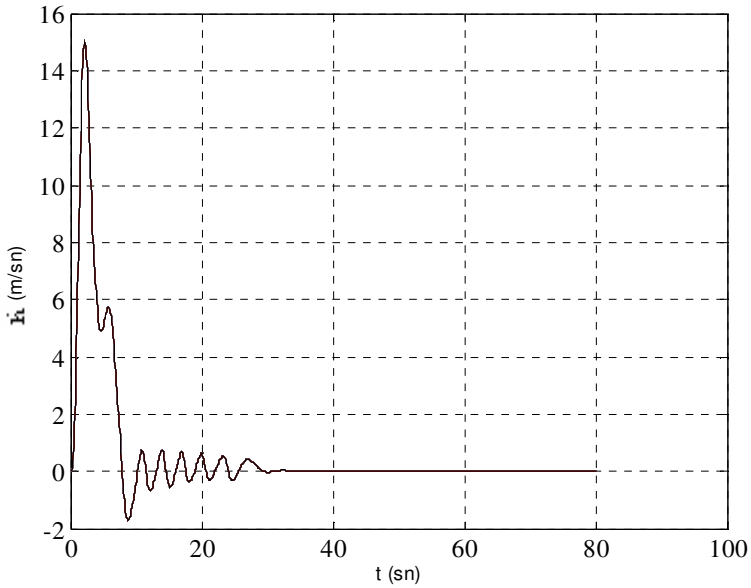
Şekil 21. Örnek 3 için hatanın zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 22. Örnek 3 için irtifa dümeni açısının zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 23. Örnek 3 için irtifanın zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 24. Örnek 3 için irtifa hızının zamana göre değişimi
(son jenerasyondaki en iyi dizi)

Örnek 4: 2.Uçuş durumu

Nüfus sayısı: 70

Jenerasyon sayısı: 30

Hassasiyet: 0.1

Çaprazlama oranı: 0.6

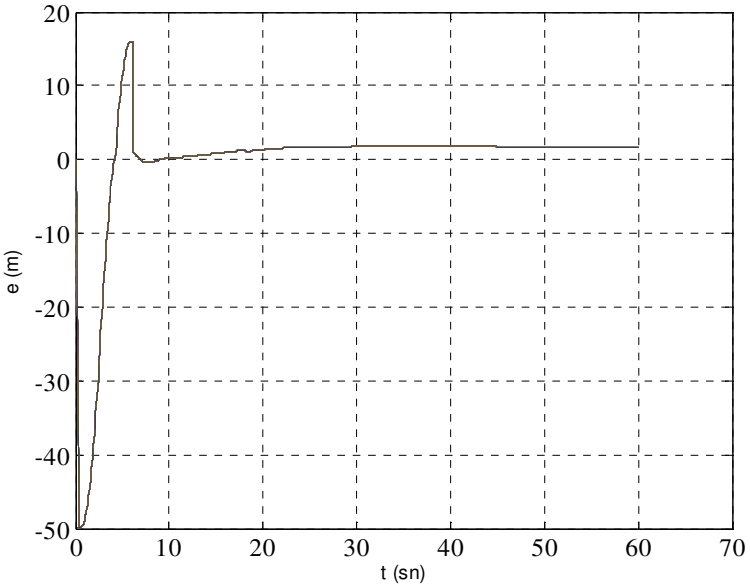
Mutasyon oranı: 0.002

Çalışma süresi: 2 sn

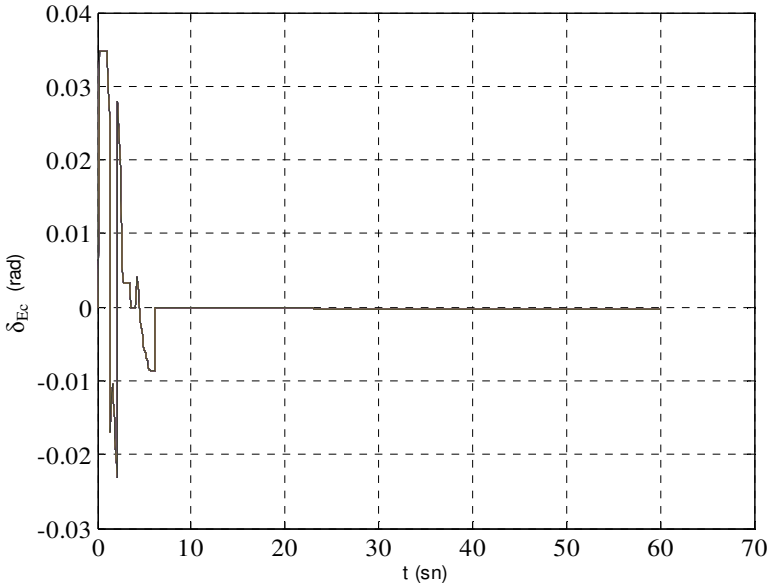
Runge-Kutta integrasyon aralığı: 0.1

Üyelik fonksiyonları sınır değerleri: ±50 m

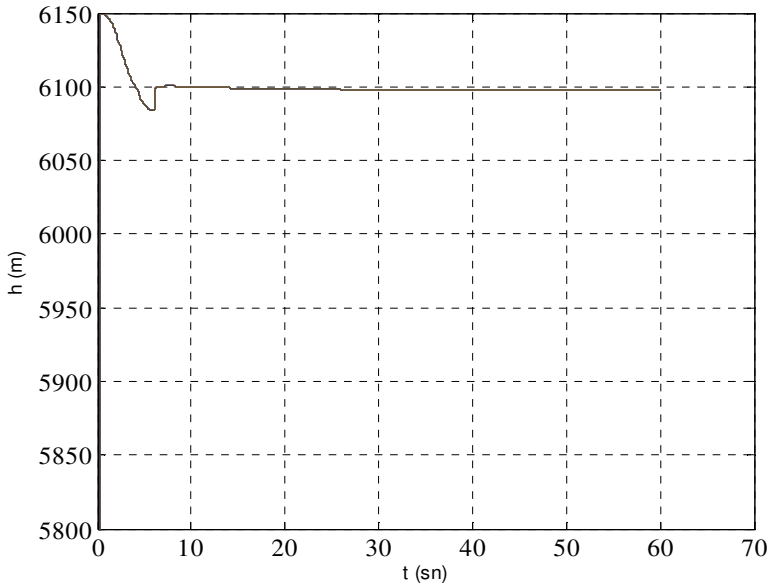
İrtifa dümeni açısı sınır değerleri: ±2°



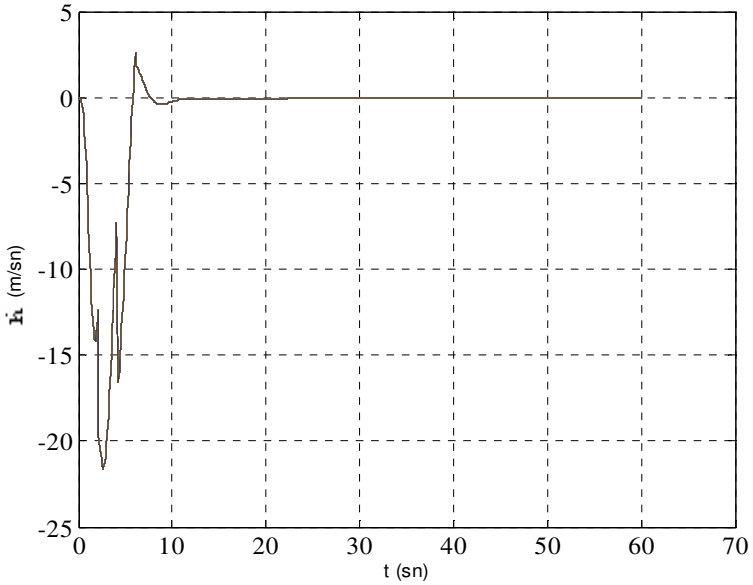
Şekil 25. Örnek 4 için hatanın zamana göre değişimi (tüm jenerasyonlar)



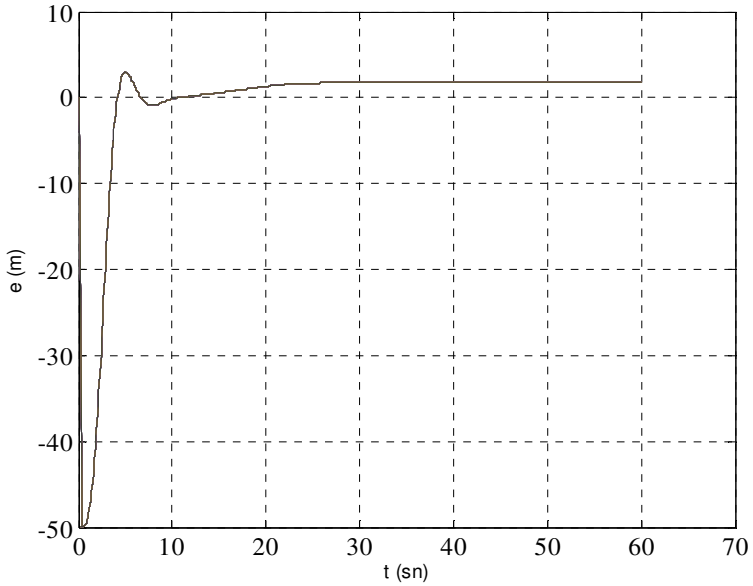
Şekil 26. Örnek 4 için irtifa dümeni açısının zamana göre değişimi (tüm jenerasyonlar)



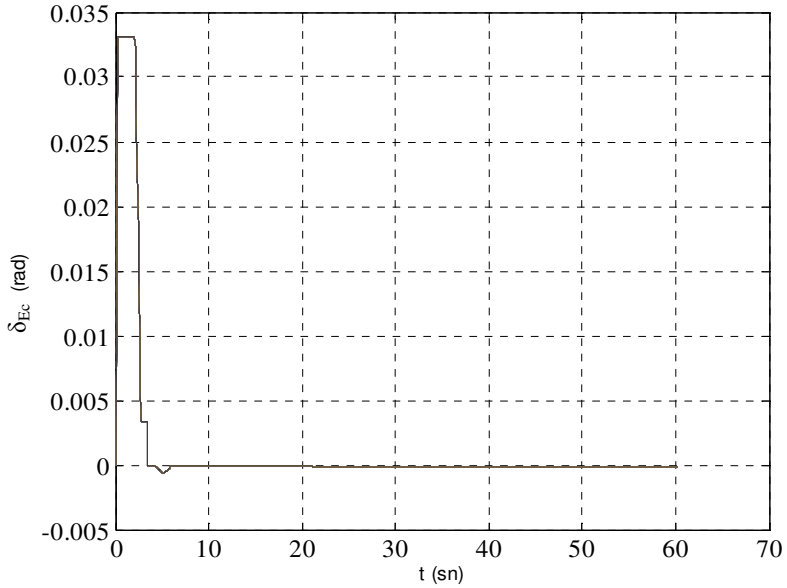
Şekil 27. Örnek 4 için irtifanın zamana göre değişimi (tüm jenerasyonlar)



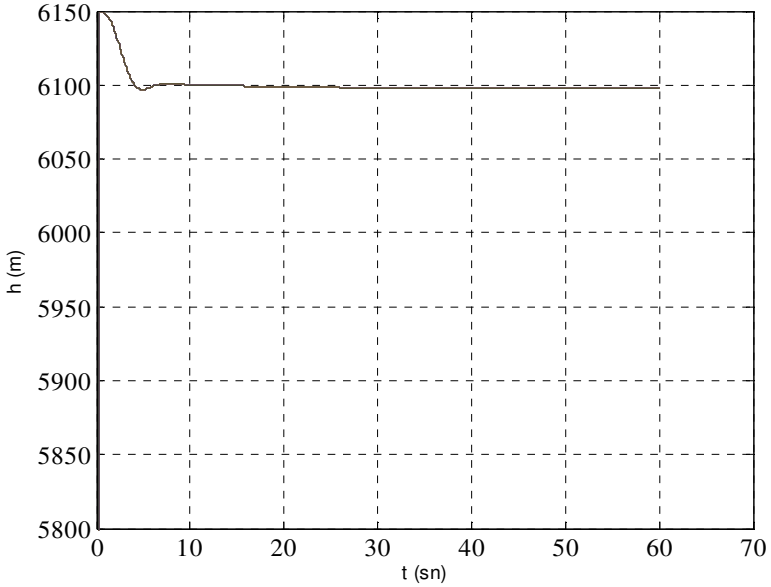
Şekil 28. Örnek 4 için irtifa hızının zamana göre değişimi
(tüm jenerasyonlar)



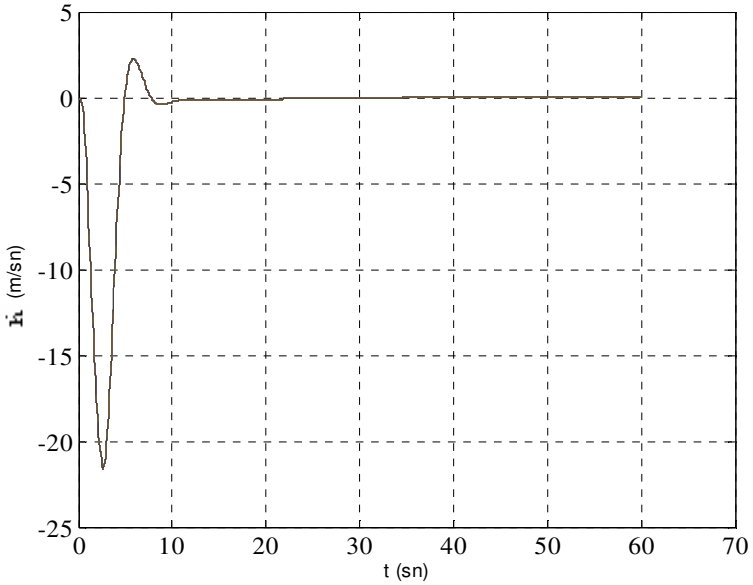
Şekil 29. Örnek 4 için hatanın zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 30. Örnek 4 için irtifa dümeni açısının zamana göre değişimi (son jenerasyondaki en iyi dizi)



Şekil 31. Örnek 4 için irtifanın zamana göre değişimi (son jenerasyondaki en iyi dizi)



Şekil 32. Örnek 4 için irtifa hızının zamana göre değişimi
(son jenerasyondaki en iyi dizi)

Örnek 5: 3.Uçuş durumu

Nüfus sayısı:100

Jenerasyon sayısı: 80

Hassasiyet: 0.1

Çaprazlama oranı: 0.6

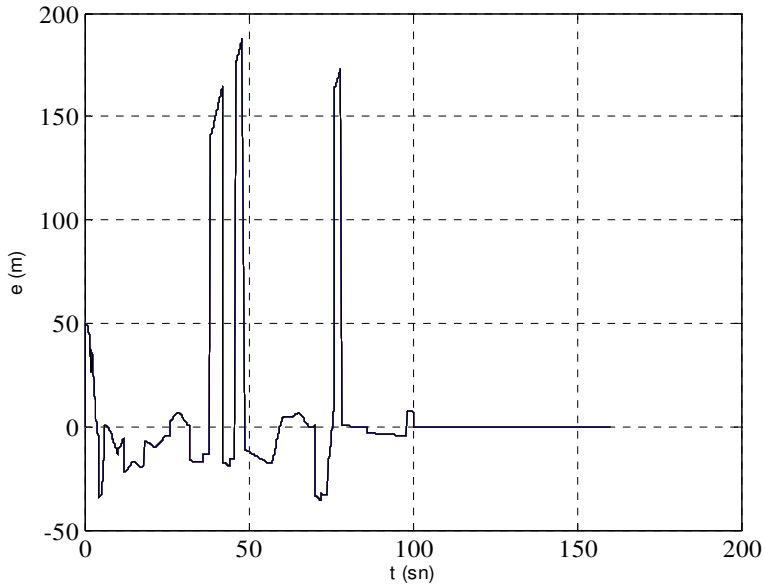
Mutasyon oranı: 0.002

Çalışma süresi: 2 sn

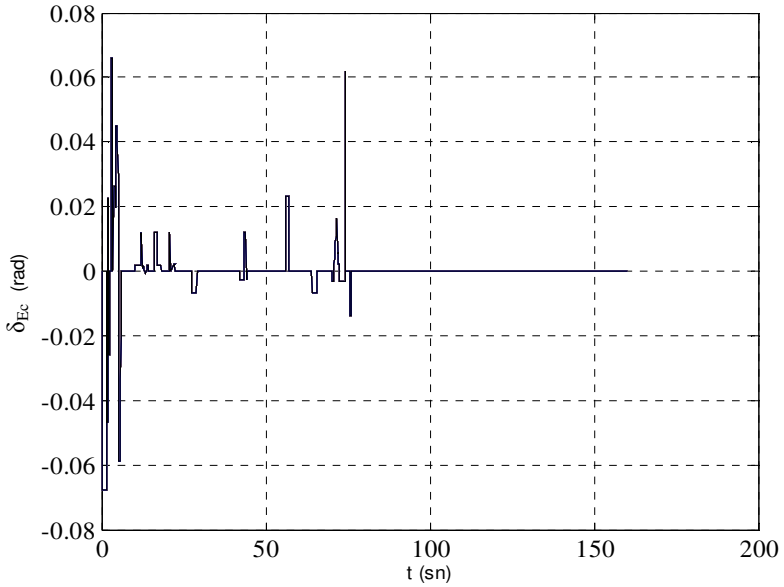
Runge-Kutta integrasyon aralığı: 0.1

Üyelik fonksiyonları sınır değerleri: ±50 m

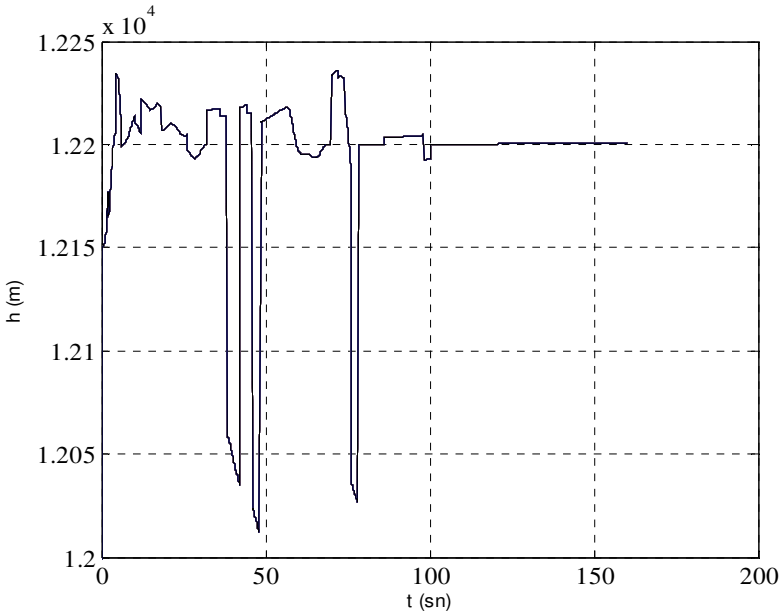
İrtifa dümeni açısı sınır değerleri: ±4°



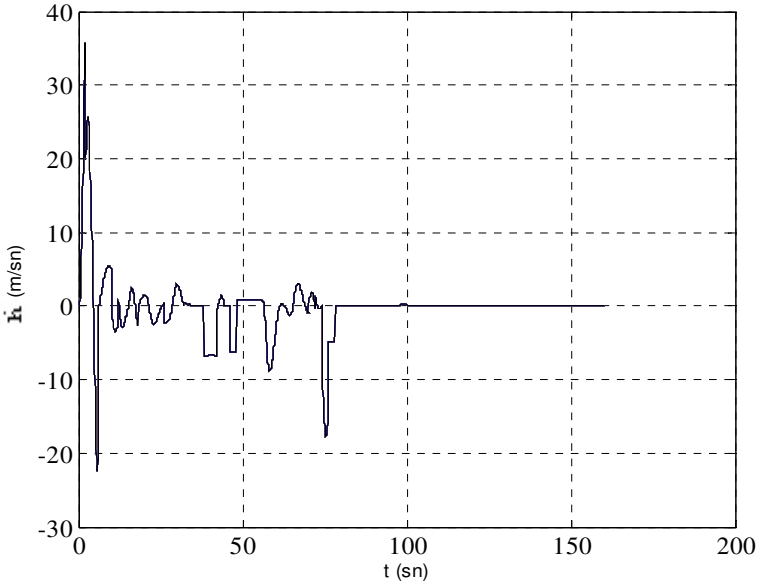
Şekil 33. Örnek 5 için hatanın zamana göre değişimi (tüm jenerasyonlar)



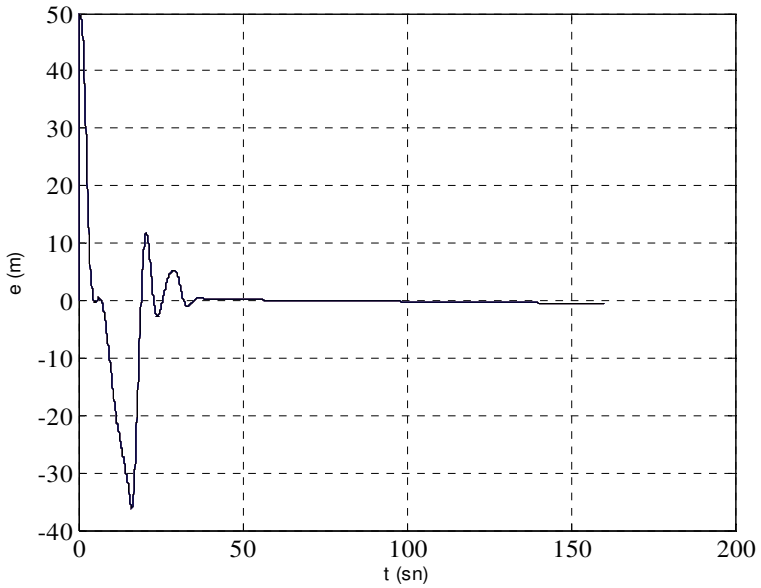
Şekil 34. Örnek 5 için irtifa dümeni açısının zamana göre değişimi (tüm jenerasyonlar)



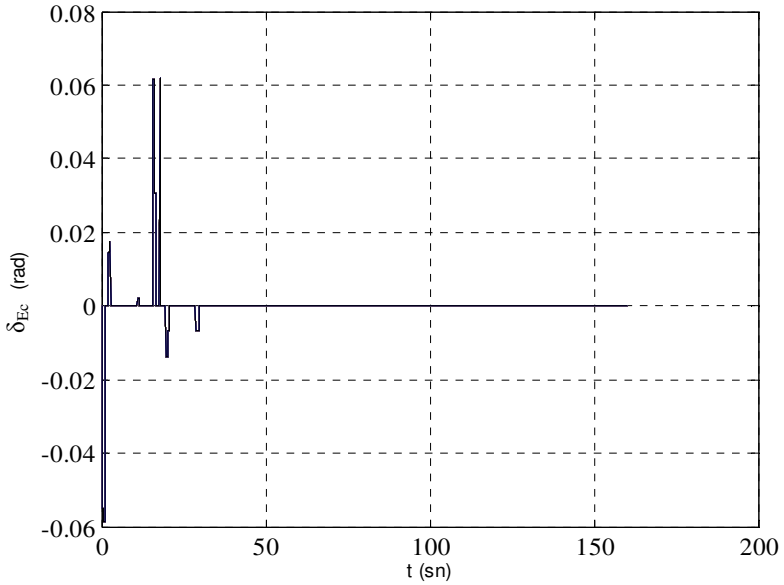
Şekil 35. Örnek 5 için irtifanın zamana göre değişimi (tüm jenerasyonlar)



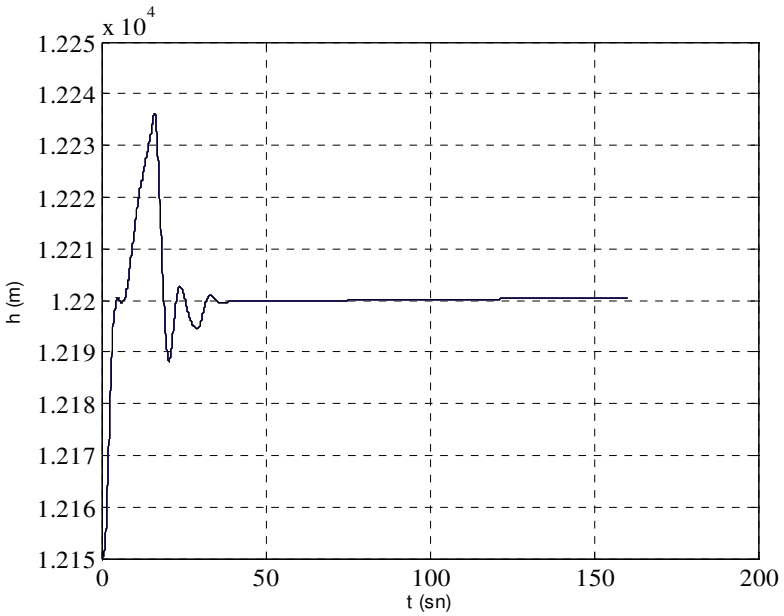
Şekil 36. Örnek 5 için irtifa hızının zamana göre değişimi
(tüm jenerasyonlar)



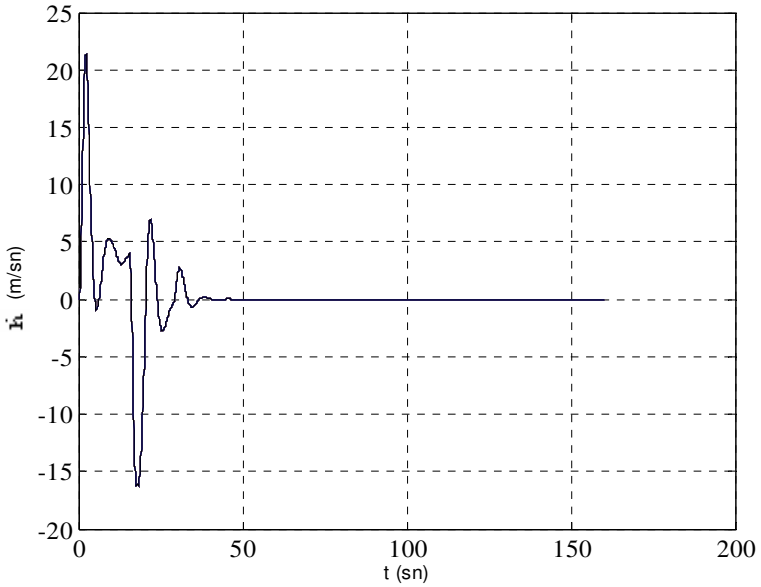
Şekil 37. Örnek 5 için hatanın zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 38. Örnek 5 için irtifa dümeni açısının zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 39. Örnek 5 için irtifanın zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 40. Örnek 5 için irtifa hızının zamana göre değişimi
(son jenerasyondaki en iyi dizi)

Örnek 6: 3.Uçuş durumu

Nüfus sayısı: 60

Jenerasyon sayısı: 30

Çaprazlama oranı: 0.6

Mutasyon oranı: 0.002

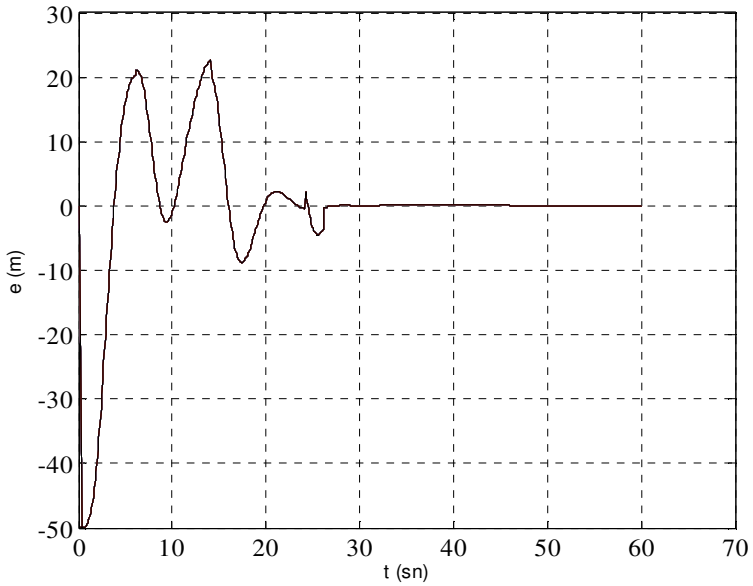
Hassasiyet: 0.1

Çalışma süresi: 2 sn

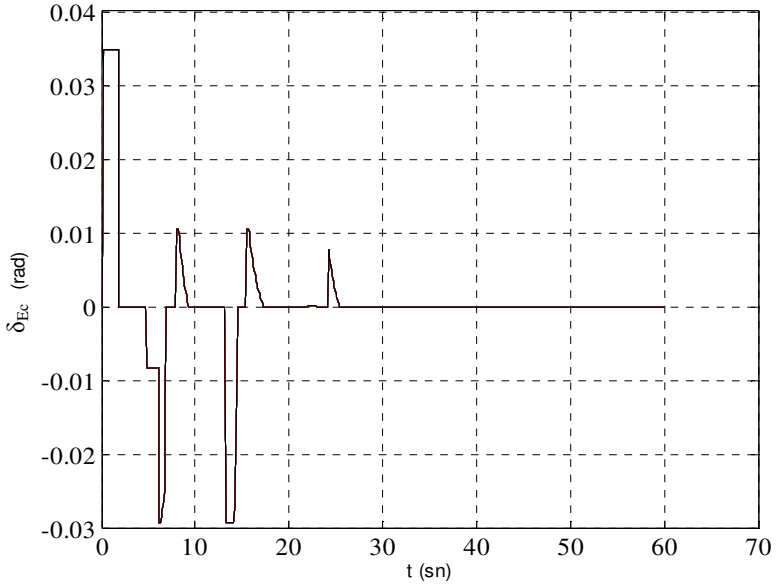
Runge-Kutta integrasyon aralığı: 0.1

Üyelik fonksiyonları sınır değerleri: ±50 m

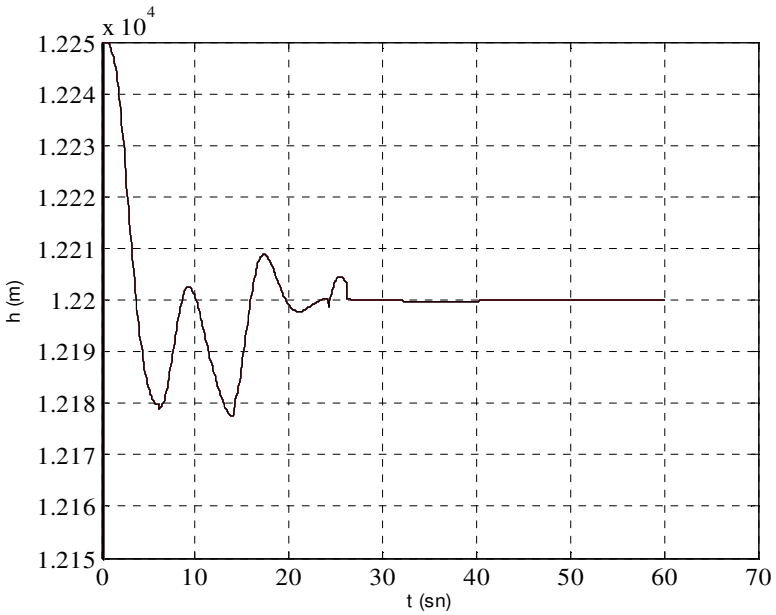
İrtifa dümeni açısı sınır değerleri: ±2°



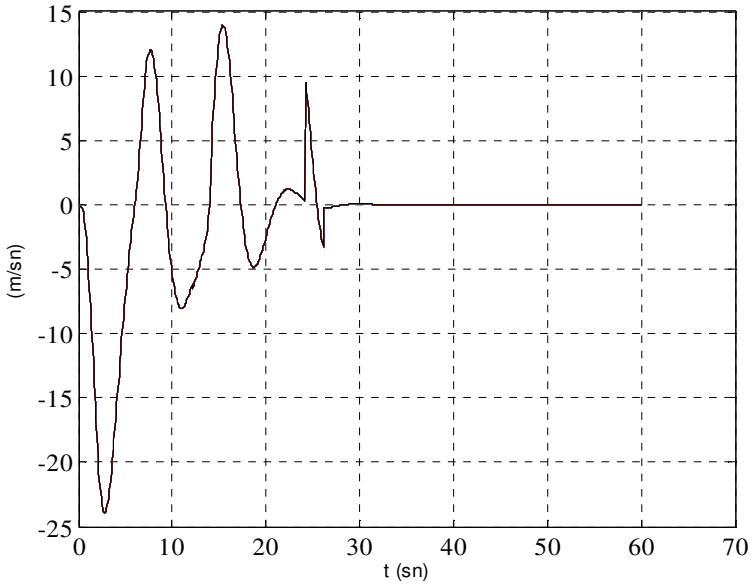
Şekil 41. Örnek 6 için hatanın zamana göre değişimi (tüm jenerasyonlar)



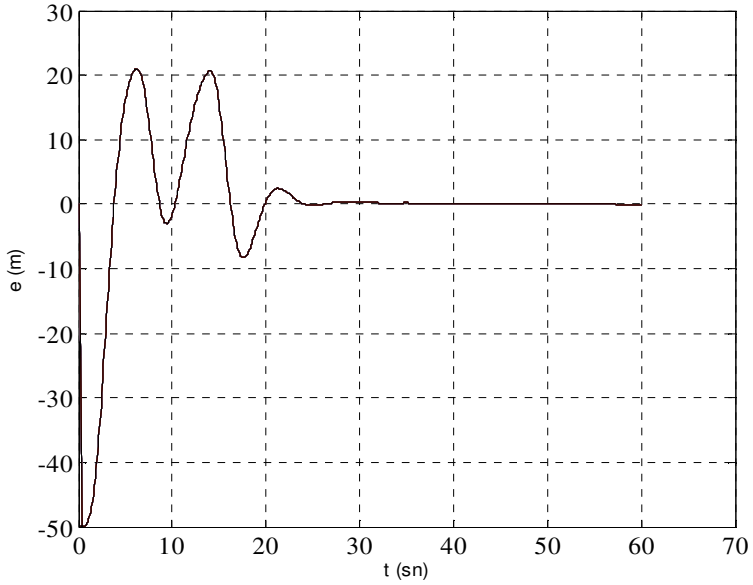
Şekil 42. Örnek 6 için irtifa dümeni açısının zamana göre değişimi (tüm jenerasyonlar)



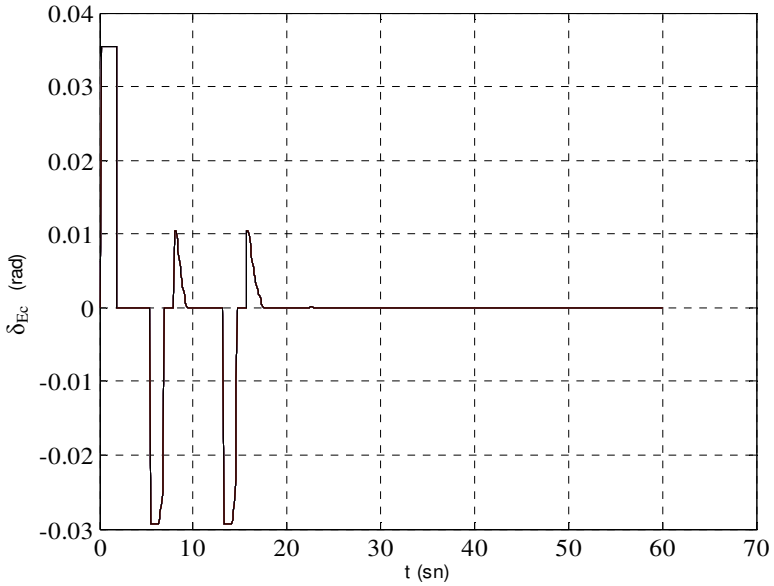
Şekil 43. Örnek 6 için irtifanın zamana göre değişimi (tüm jenerasyonlar)



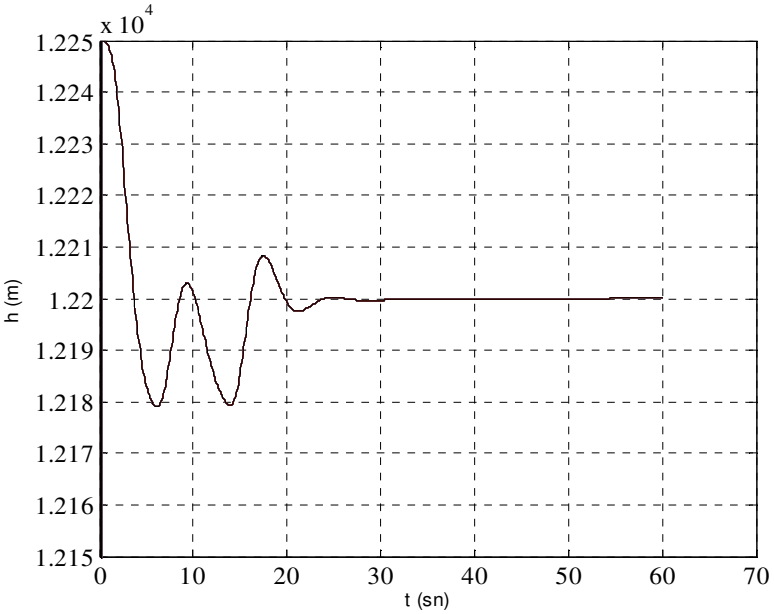
Şekil 44. Örnek 6 için irtifa hızının zamana göre değişimi
(tüm jenerasyonlar)



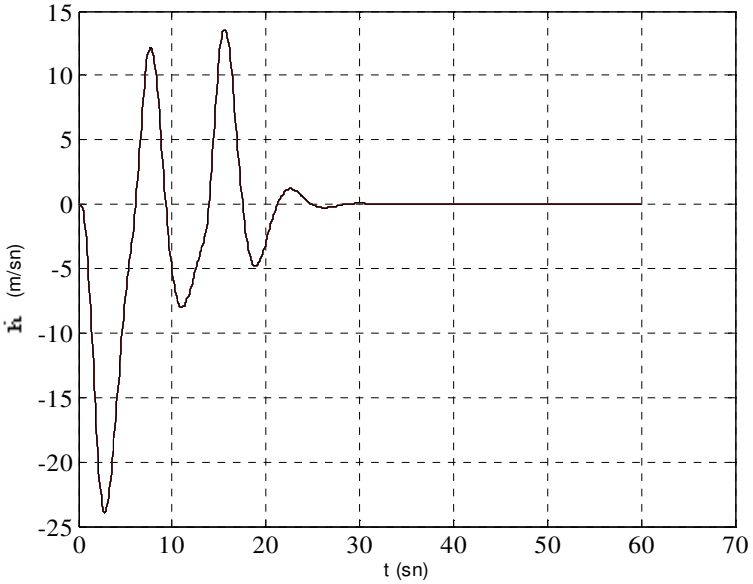
Şekil 45. Örnek 6 için hatanın zamana göre değişimi
(son jenerasyondaki en iyi dizi)



Şekil 46. Örnek 6 için irtifa dümeni açısının zamana göre değişimi (son jenerasyondaki en iyi dizi)



Şekil 47. Örnek 6 için irtifanın zamana göre değişimi (son jenerasyondaki en iyi dizi)



Şekil 48. Örnek 6 için irtifa hızının zamana göre değişimi
(son jenerasyondaki en iyi dizi)

Ek-2: Matlab 6.5 kullanılarak kodlanan genetik algoritma tabanlı bulanık denetleyici programı

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%Mainf3g programı%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%tr          :Çalışma süresi
%ts          :Runge Kutta integrasyon aralığı
%eo          :Hata dizisi
%eom         :Son jenerasyondaki minimum hata dizisi
%eomc        :Her jenerasyona ait minimum hata dizisi
%f           :1/eo dizisi
%ft          :ft Toplamı
%deltaeo     :İrtifa dümeni açısı dizisi
%deltaeom    :Son jenerasyonda minimum hatayı veren bireye ait
irtifa dümeni açısı dizisi
%deltaeomc   :Her jenerasyonda minimum hatayı veren bireye ait
İrtifa dümeni açısı dizisi
%x1o         :İrtifa dizisi
%x1om        :Son jenerasyonda minimum hatayı veren bireye ait
irtifa dizisi
%x1omc       :Her jenerasyonda minimum hatayı veren bireye ait
irtifa dizisi
%x2o         :İrtifa hızı dizisi
%x2om        :Son jenerasyonda minimum hatayı veren bireye ait
irtifa hızı dizisi
%x2omc       :Her jenerasyonda minimum hatayı veren bireye ait
irtifa hızı dizisi
%Sft         :ft değerlerinin toplamı
%Avft        :ft değerlerinin ortalaması
%Maxft       :ft değerlerinin maksimumu
%Pselect     :Probability select vektörü
%Excnt       :Expected count
%Accnt       :Actual count
%SPselect    :Probability select toplamı
%SExcnt      :Expected count toplamı
%SAccnt      :Actual count toplamı
%rac         :Actual count vektörünün satır sayısı
%colac       :Actual count vektörünün sütun sayısı
%reprodec    :Reproduction işleminde oluşan desimal sayıları tutan
vektör
%reprobin    :Reproduction işleminde oluşan desimal sayıların
binary
%değerlerini tutan matris
%n2a         :Actual count'da 2 ve daha üstü
%n1          :Actual count'da 1 olan değerlerin sayısı
%n0          :Actual count'da 0 olan değerlerin sayısı
%mate        :Crossover'a uğrayacak bireylerin numaralarını  tutan
matris
%crossite    :Random olarak belirlenen crossite noktalarını tutan
vektör
%mut         :Mutasyona uğrayacak bit numaralarını tutan vektör

clear
warning off
nx=6;
tr=input('Çalışma süresi: ');
disp(' ')
```

```
ts=input('RUNGE-KUTTA integrasyon aralığı: ');

%ntd=round(tr/ts);
   %x(1:nx)=zeros(1,nx);
 %xp(1:nx)=zeros(1,nx);

genetik1

%ifadea=fopen('verilera.dat','w+');
%ifadeb=fopen('verilerb.dat','w+');
%ifadec=fopen('verilerc.dat','w+');
%ifaded=fopen('verilerd.dat','w+');
ifade1=fopen('veriler1.dat','w+');
ifade2=fopen('veriler2.dat','w+');
%ifade3a=fopen('veriler3a.dat','w+');
%ifade3b=fopen('veriler3b.dat','w+');
%ifade3c=fopen('veriler3c.dat','w+');
%ifade3d=fopen('veriler3d.dat','w+');
%ifade4a=fopen('veriler4a.dat','w+');
%ifade4b=fopen('veriler4b.dat','w+');
%ifade4c=fopen('veriler4c.dat','w+');
%ifade4d=fopen('veriler4d.dat','w+');
ifade5=fopen('veriler5.dat','w+');

for jen=1:1:gn,

   % fprintf(ifadea,'%1.0f',jen);
   % fprintf(ifadeb,'%1.0f',jen);
   % fprintf(ifadec,'%1.0f',jen);
   % fprintf(ifaded,'%1.0f',jen);
    fprintf(ifade1,'%1.0f',jen);
    fprintf(ifade2,'%1.0f',jen);
    %fprintf(ifade3a,'%1.0f',jen);
    %fprintf(ifade3b,'%1.0f',jen);
    %fprintf(ifade3c,'%1.0f',jen);
    %fprintf(ifade3d,'%1.0f',jen);
    %fprintf(ifade4a,'%1.0f',jen);
    %fprintf(ifade4b,'%1.0f',jen);
    %fprintf(ifade4c,'%1.0f',jen);
    %fprintf(ifade4d,'%1.0f',jen);
%    fprintf(ifade5,'%1.0f',jen);
    %fprintf(ifadea,'.jenerasyon......');
    %fprintf(ifadeb,'.jenerasyon......');
    %fprintf(ifadec,'.jenerasyon......');
    %fprintf(ifaded,'.jenerasyon......');
    fprintf(ifade1,'.jenerasyon......');
    fprintf(ifade2,'.jenerasyon......');
    %fprintf(ifade3a,'.jenerasyon......');
    %fprintf(ifade3b,'.jenerasyon......');
    %fprintf(ifade3c,'.jenerasyon......');
    %fprintf(ifade3d,'.jenerasyon......');
    %fprintf(ifade4a,'.jenerasyon......');
    %fprintf(ifade4b,'.jenerasyon......');
    %fprintf(ifade4c,'.jenerasyon......');
    %fprintf(ifade4d,'.jenerasyon......');
%    fprintf(ifade5,'.jenerasyon......');
    %fprintf(ifadea,'\n');
    %fprintf(ifadeb,'\n');
```

```
    %fprintf(ifadec,'\n');
    %fprintf(ifaded,'\n');
    fprintf(ifade1,'\n');
    fprintf(ifade2,'\n');
    %fprintf(ifade3a,'\n');
    %fprintf(ifade3b,'\n');
    %fprintf(ifade3c,'\n');
    %fprintf(ifade3d,'\n');
    %fprintf(ifade4a,'\n');
    %fprintf(ifade4b,'\n');
    %fprintf(ifade4c,'\n');
    %fprintf(ifade4d,'\n');
%    fprintf(ifade5,'\n');

 % ft değerlerinin tutulduğu vektörlerin oluşturulması

 tr1=jen*tr;
         ntd=round(tr1/ts);
     for i=1:1:pn
         xx=[6070 0 0 0 0 0];
         xxp=[0 0 0 0 0 0];
         ft(i,1)=0;
         time=0.0;
         it=1;
         ip=1;
         t(ip,i)=time;

          for kid=1:ntd
            Ctrf3g
            Runkutf3g
            it=it+1;
            time=it*ts;
            ip=ip+1;
            t(ip,i)=time;
            if e(i,1)==0;
             eo(ip,i)=0.0000001;
            else
            eo(ip,i)=e(i,1);
            end
            f(ip,i)=1/abs(eo(ip,i));
            ft(i,1)=ft(i,1)+f(ip,i);
            deltaeo(ip,i)=deltae(i,1);
            x1o(ip,i)=xx(1);x2o(ip,i)=xx(2);
         end
     end

     Sft=sum(ft);
     Avft=Sft/pn;
     Maxft=max(ft);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
 %genetik2 subprogram----------------------------------------

 % Pselect, Expected count ve actual count değerlerinin
hesaplanması--------
```

```
for i=1:1:pn,
     Pselect(i,1)=ft(i,1)/Sft;
     Excnt(i,1)=ft(i,1)/Avft;
     Accnt(i,1)=round(Excnt(i,1));
 end

 SPselectx=sum(Pselect);
 SExcnt=sum(Excnt);
 SAcnt=sum(Accnt);
 [rac,colac]=size(Accnt);

 %-----------------------------------------------------------------

 %REPRODUCTION iŞLEMİ
 % actual count değerlerine göre ilgili kromozomların
tekrarlanması

    reprobin=[ ];
    for i=1:1:pn,
        m=Accnt(i,1);
        for j=1:1:m,
            v=binpop(i,:);
            reprobin=[reprobin;v];
        end
        clear v
    end
    [row,col]=size(reprobin);

 % popülasyondaki birey sayısından eksik olması durumu

     while row-pn<0,
           n0=0;
           n1=0;
           for i=1:rac,
               if isequal (0,Accnt(i,1))==1,
                  n0=n0+1;
               end
           end
           for i=1:rac,
                if isequal(1,Accnt(i,1))==1,
                   n1=n1+1;
                end
           end
               n2a=rac-(n0+n1);
               i=1;
                   while (row-pn<0)&(n2a~=0)
                         m=Accnt(i,1);
                         if m>=2,
                            reprobin=[reprobin;binpop(i,:)];
                            n2a=n2a-1;
                         end
                        [row,col]=size(reprobin);
                        i=i+1;
                    end
                    i=1;
                    if n2a==0,
                       while (row-pn<0)&(n1~=0)
                             m=Accnt(i,1);
```

```
                        if m==1,
                           reprobin=[reprobin;binpop(i,:)];
                           n1=n1-1;
                        end
                       [row,col]=size(reprobin);
                       i=i+1;
                    end
                end
            end

% popülasyondaki birey sayısından fazla olması durumu

        while row-pn>0
            n0=0;
            n1=0;
            for i=1:rac,
                if isequal(0,Accnt(i,1))==1,
                    n0=n0+1;
                end
            end
            for i=1:rac,
                if isequal(1,Accnt(i,1))==1,
                    n1=n1+1;
                end
            end
            n2a=rac-(n0+n1);
            i=1;
            while (row-pn>0)&(n1~=0)
                  m=Accnt(i,1);
                  if m==1,
                     Accnt(i,1)=Accnt(i,1)-1;
                     n1=n1-1;
                  end
                  reprobin=[ ];
                  for j=1:1:pn,
                      n=Accnt(j,1);
                      for y=1:1:n,
                          v=binpop(j,:);
                          reprobin=[reprobin;v];
                      end
                      clear v
                  end
                  [row,col]=size(reprobin);
                  i=i+1;
              end
              i=1;
              if n1==0,
                 while (row-pn>0)&(n2a~=0)
                        m=Accnt(i,1);
                        if m>=2,
                           Accnt(i,1)=Accnt(i,1)-1;
                           n2a=n2a-1;
                        end
                        reprobin=[ ];
                        for j=1:1:pn,
                            n=Accnt(j,1);
                            for y=1:1:n,
                                v=binpop(j,:);
```

```
                                    reprobin=[reprobin;v];
                                end
                                clear v
                            end
                            [row,col]=size(reprobin);
                            i=i+1;
                        end
                    end
                end

 %reprobin=de2bi(reprodec,bitsayt,'left-msb');

%-----------------------------------------------------


% CROSSOVER İŞLEMİ---------------------------------
% crosovera uğrayacak bireylerin random olarak belirlenmesi

                if crs~=0,
                   r=[1:pn];
                   m=randsrc(1,1,r);
                   for i=2:crs,
                       mm=randsrc(1,1,r);
                       while ismember(mm,m)==1,
                             mm=randsrc(1,1,r);
                       end
                       m=[m;mm];
                   end
                   for i=1:1:crs/2,
                       mate1(i,1)=m(i*2-1,1);
                       mate2(i,1)=m(i*2,1);
                   end
                   mate=[mate1 mate2];

% crossiteların random olarak belirlenmesi

                r=[2:bitsayt-1];
                for i=1:crs/2,
                    crosssite(i,1)=randsrc(1,1,r);
                end
              %  fprintf(ifade3a,'\n');
               % fprintf(ifade3a,'crossovera uğrayacak bireylerin
numaraları ve crosssite noktaları:');
               % fprintf(ifade3a,'\n');
                %for i=1:1:crs/2,
                 %   fprintf(ifade3a,'  kromozom ');
                  %  fprintf(ifade3a, '%1.0f',mate1(i,1));
                   % fprintf(ifade3a,'-------- kromozom');
                    %fprintf(ifade3a, '%1.0f',mate2(i,1));
                    %fprintf(ifade3a,'-------- crosssite');
                    %fprintf(ifade3a, '%1.0f',crosssite(i,1));
                    %fprintf(ifade3a,'\n');
                %end
                %fprintf(ifade3a,'\n');
```

```
                for i=1:1:pn,
                    No(i,1)=i;
                end

                %fprintf(ifade3a,'\n');
                %fprintf(ifade3a,'       No
reproduction(crossover mutasyon öncesi hata');
                %fprintf(ifade3a,'\n');
                %binpopv=reprobin;
                %bitsayh=bitsay1t/2;
                %for i=1:pn,
                 %   for j=1:bitsayh,
                  %      binpoph(i,j)=binpopv(i,j);
                   % end
                %end
                %for i=1:pn,
                 %   fprintf(ifade3a,'%6.0f',No(i,1));
                  %  fprintf(ifade3a,'%3.0f',binpoph(i,:));
                   % fprintf(ifade3a,'\n');
                %end
                %fprintf(ifade3a,'\n');

                %fprintf(ifade3b,'\n');
                %fprintf(ifade3b,'       No
reproduction(crossover mutasyon öncesi hatad');
                %fprintf(ifade3b,'\n');
                %for i=1:pn,
                 %   for j=1:bitsayh,
                  %      binpophd(i,j)=binpopv(i,bitsayh+j);
                   % end
                %end
               % for i=1:pn,
                %    fprintf(ifade3b,'%6.0f',No(i,1));
                 %   fprintf(ifade3b,'%3.0f',binpophd(i,:));
                  %  fprintf(ifade3b,'\n');
                %end
                %fprintf(ifade3b,'\n');

                %fprintf(ifade3c,'\n');
                %fprintf(ifade3c,'       No
reproduction(crossover mutasyon öncesi kurala');
                %fprintf(ifade3c,'\n');
                %bitsaykur=bitsay2t/2;
                %for i=1:pn,
                 %   for j=1:bitsaykur,
                  %      binpopkura(i,j)=binpopv(i,bitsay1t+j);
                   % end
                %end
                %for i=1:pn,
                 %   fprintf(ifade3c,'%6.0f',No(i,1));
                  %  fprintf(ifade3c,'%3.0f',binpopkura(i,:));
                   % fprintf(ifade3c,'\n');
                %end
                %fprintf(ifade3c,'\n');

                %fprintf(ifade3d,'\n');
                %fprintf(ifade3d,'       No
reproduction(crossover mutasyon öncesi kuralb');
```

```
                %fprintf(ifade3d,'\n');
                %bitsaykurc=bitsay1t+bitsaykur;
                %for i=1:pn,
                 %   for j=1:bitsaykur,
                  %      binpopkurb(i,j)=binpopv(i,bitsaykurc+j);
                   % end
                %end
                %for i=1:pn,
                 %   fprintf(ifade3d,'%6.0f',No(i,1));
                  %  fprintf(ifade3d,'%3.0f',binpopkurb(i,:));
                   % fprintf(ifade3d,'\n');
                %end
                %fprintf(ifade3d,'\n');



%crossover işleminin uygulanması

                for i=1:1:length(mate),
                    c=crosssite(i,1);
                    for k=c+1:1:bitsayt,
                        a=reprobin(mate(i,1),k);
                        b=reprobin(mate(i,2),k);
                        d=a;
                        a=b;
                        b=d;
                        reprobin(mate(i,1),k)=a;
                        reprobin(mate(i,2),k)=b;
                    end
                end
            end
                %-------------------------------------------------

            %MUTASYON İŞLEMİ-----------------------------------
               % mutasyona uğrayacak bitlerin random olarak
belirlenmesi
                    if ms~=0,
                       r=[1:pn];
                       c=[1:bitsayt];
                       row=randsrc(1,1,r);
                       col=randsrc(1,1,c);
                       for i=2:ms,
                           rr=randsrc(1,1,r);
                           cc=randsrc(1,1,c);
                            while ismember(rr,row)==1,
                             rr=randsrc(1,1,r);
                            end
                            %while ismember(cc,col)==1,
                             cc=randsrc(1,1,c);
                            %end
                       row=[row;rr];
                       col=[col;cc];
                       end
                       mut=[row col];
                 %      fprintf(ifade3a,'mutasyona uğrayacak
bitlerin numaraları  :  ');
                  %     fprintf(ifade3a,'\n');
                   %    for i=1:1:ms,
```

```
                    %   fprintf(ifade3a,'  kromozom ');
                     %  fprintf(ifade3a,'%1.0f',row(i));
                      % fprintf(ifade3a,'--------- bit  ');
                       %fprintf(ifade3a,'%1.0f',col(i));
                       %fprintf(ifade3a,'\n');
                      % end

                 % mutasyon işleminin uygulanması

                        for i=1:1:length(mut),
                            if reprobin(mut(i,1),mut(i,2))==1,
                               reprobin(mut(i,1),mut(i,2))=0;
                            else
                               reprobin(mut(i,1),mut(i,2))=1;
                            end
                        end
                 end

                 %-------------------------------------------------

                %% fprintf(ifadea,'\n');
                 %fprintf(ifadea,'        No         İlk populasyon
hata');
                 % fprintf(ifadea,'\n');

                %  bitsayh=bitsay1t/2;
                 % for i=1:pn,
                  %    for j=1:bitsayh,
                   %       binpoph(i,j)=binpop(i,j);
                    %  end
                 % end
                  %for i=1:pn,
                   %   fprintf(ifadea,'%6.0f',No(i,1));
                    %  fprintf(ifadea,'%3.0f',binpoph(i,:));
                     % fprintf(ifadea,'\n');
                 %end
                 %fprintf(ifadea,'\n');

                 %fprintf(ifadeb,'\n');
                 %fprintf(ifadeb,'        No       İlk   populasyon
hatad');
                 %fprintf(ifadeb,'\n');

                 %for i=1:pn,
                  %   for j=1:bitsayh,
                   %      binpophd(i,j)=binpop(i,bitsayh+j);
                    % end
                 %end
                 %for i=1:pn,
                  %   fprintf(ifadeb,'%6.0f',No(i,1));
                   %  fprintf(ifadeb,'%3.0f',binpophd(i,:));
                    % fprintf(ifadeb,'\n');
                 %end
                 %fprintf(ifadeb,'\n');

                 %fprintf(ifadec,'\n');
```

```
                %fprintf(ifadec,'        No      İlk   populasyon
kurala');
                %fprintf(ifadec,'\n');

                %bitsaykur=bitsay2t/2;
                %for i=1:pn,
                 %   for j=1:bitsaykur,
                  %      binpopkura(i,j)=binpop(i,bitsay1t+j);
                   % end
                 %end
                % for i=1:pn,
                 %    fprintf(ifadec,'%6.0f',No(i,1));
                  %   fprintf(ifadec,'%3.0f',binpopkura(i,:));
                   %  fprintf(ifadec,'\n');
                 %end
                % fprintf(ifadec,'\n');

                %fprintf(ifaded,'\n');
                %fprintf(ifaded,'        No   ilk populasyon
kuralb');
                %fprintf(ifaded,'\n');
                %bitsaykurc=bitsay1t+bitsaykur;
                %for i=1:pn,
                 %   for j=1:bitsaykur,
                  %      binpopkurb(i,j)=binpop(i,bitsaykurc+j);
                   % end
                % end
                 %for i=1:pn,
                  %   fprintf(ifaded,'%6.0f',No(i,1));
                   %  fprintf(ifaded,'%3.0f',binpopkurb(i,:));
                    % fprintf(ifaded,'\n');
                 %end
                 %fprintf(ifaded,'\n');


                 fprintf(ifade1,'\n');
                 fprintf(ifade1,'        No          x değerleri');
                 fprintf(ifade1,'\n');
                 fprintf(ifade2,'\n');
                 fprintf(ifade2,'        No          f(t)
Accnt');
                 fprintf(ifade2,'\n');

                 for i=1:pn,

                     fprintf(ifade1,'%6.0f',No(i,1));
                     fprintf(ifade1,'%9.2f',x1(i,1));
                     fprintf(ifade1,'%9.2f',x2(i,1));
                     fprintf(ifade1,'%9.2f',x3(i,1));
                     fprintf(ifade1,'%9.2f',x4(i,1));
                     fprintf(ifade1,'%9.2f',x5(i,1));
                     fprintf(ifade1,'%9.2f',x6(i,1));
                     fprintf(ifade1,'%9.2f',x7(i,1));
                     fprintf(ifade1,'%9.2f',x8(i,1));
                     fprintf(ifade1,'%9.2f',x9(i,1));
                     fprintf(ifade1,'%9.2f',x10(i,1));
                     fprintf(ifade1,'%9.2f',x11(i,1));
                     fprintf(ifade1,'%9.2f',x12(i,1));
```

```
                    fprintf(ifade1,'%9.2f',x13(i,1));
                    fprintf(ifade1,'%9.2f',x1d(i,1));
                    fprintf(ifade1,'%9.2f',x2d(i,1));
                    fprintf(ifade1,'%9.2f',x3d(i,1));
                    fprintf(ifade1,'%9.2f',x4d(i,1));
                    fprintf(ifade1,'%9.2f',x5d(i,1));
                    fprintf(ifade1,'%9.2f',x6d(i,1));
                    fprintf(ifade1,'%9.2f',x7d(i,1));
                    fprintf(ifade1,'%9.2f',x8d(i,1));
                    fprintf(ifade1,'%9.2f',x9d(i,1));
                    fprintf(ifade1,'%9.2f',x10d(i,1));
                    fprintf(ifade1,'%9.2f',x11d(i,1));
                    fprintf(ifade1,'%9.2f',x12d(i,1));
                    fprintf(ifade1,'%9.2f',x13d(i,1));
                    fprintf(ifade1,'%9.2f',x14(i,1));
                    fprintf(ifade1,'%9.2f',x15(i,1));
                    fprintf(ifade1,'%9.2f',x16(i,1));
                    fprintf(ifade1,'%9.2f',x17(i,1));
                    fprintf(ifade1,'%9.2f',x18(i,1));
                    fprintf(ifade1,'%9.2f',x19(i,1));
                    fprintf(ifade1,'%9.2f',x20(i,1));
                    fprintf(ifade1,'%9.2f',x21(i,1));
                    fprintf(ifade1,'%9.2f',x22(i,1));
                    fprintf(ifade1,'%9.2f',x23(i,1));
                    fprintf(ifade1,'%9.2f',x24(i,1));
                    fprintf(ifade1,'%9.2f',x25(i,1));
                    fprintf(ifade1,'%9.2f',x26(i,1));
                    fprintf(ifade1,'%9.2f',x27(i,1));
                    fprintf(ifade1,'%9.2f',x28(i,1));
                    fprintf(ifade1,'%9.2f',x29(i,1));
                    fprintf(ifade1,'%9.2f',x30(i,1));
                    fprintf(ifade1,'%9.2f',x31(i,1));
                    fprintf(ifade1,'%9.2f',x32(i,1));
                    fprintf(ifade1,'%9.2f',x33(i,1));
                    fprintf(ifade1,'%9.2f',x34(i,1));
                    fprintf(ifade1,'%9.2f',x35(i,1));
                    fprintf(ifade1,'%9.2f',x36(i,1));
                    fprintf(ifade1,'%9.2f',x37(i,1));
                    fprintf(ifade1,'%9.2f',x38(i,1));
                    fprintf(ifade1,'\n');
                    fprintf(ifade2,'%10.2f',ft(i,1));
                    fprintf(ifade2,'%7.0f',Accnt(i,1));
                    fprintf(ifade2,'\n');

                end

                fprintf(ifade1,'\n');
                fprintf(ifade2,'\n');

                %fprintf(ifade4a,'\n');
                %fprintf(ifade4a,'       No      reproduction
crossover mutasyon sonrası hata');
                %fprintf(ifade4a,'\n');
                %binpopv=reprobin;
                %bitsayh=bitsay1t/2;
                %for i=1:pn,
                 %   for j=1:bitsayh,
                  %      binpoph(i,j)=binpopv(i,j);
```

```
                   % end
                %end
                %for i=1:pn,
                 %   fprintf(ifade4a,'%6.0f',No(i,1));
                  %  fprintf(ifade4a,'%3.0f',binpoph(i,:));
                   % fprintf(ifade4a,'\n');
                %end
                %fprintf(ifade4a,'\n');

                %fprintf(ifade4b,'\n');
                %fprintf(ifade4b,'       No     reproduction
crossover mutasyon sonrası hatad');
                %fprintf(ifade4b,'\n');
                %for i=1:pn,
                %    for j=1:bitsayh,
                 %       binpophd(i,j)=binpopv(i,bitsayh+j);
                  %  end
                %end
                %for i=1:pn,
                 %   fprintf(ifade4b,'%6.0f',No(i,1));
                  %  fprintf(ifade4b,'%3.0f',binpophd(i,:));
                  %  fprintf(ifade4b,'\n');

                %end
                %fprintf(ifade4b,'\n');

                %fprintf(ifade4c,'\n');
                %fprintf(ifade4c,'       No      reproduction
crossover mutasyon sonrası kurala');
                %fprintf(ifade4c,'\n');
                %bitsaykur=bitsay2t/2;
                %for i=1:pn,
                 %   for j=1:bitsaykur,
                  %      binpopkura(i,j)=binpopv(i,bitsay1t+j);
                   % end
                %end
                %for i=1:pn,
                 %   fprintf(ifade4c,'%6.0f',No(i,1));
                  %  fprintf(ifade4c,'%3.0f',binpopkura(i,:));
                  %  fprintf(ifade4c,'\n');
                %end
                %fprintf(ifade4c,'\n');

               % fprintf(ifade4d,'\n');
                %fprintf(ifade4d,'       No
reproduction(crossover mutasyon sonrası kuralb');
                %fprintf(ifade4d,'\n');
                %bitsaykurc=bitsay1t+bitsaykur;
                %for i=1:pn,
                 %   for j=1:bitsaykur,
                  %      binpopkurb(i,j)=binpopv(i,bitsaykurc+j);
                   % end
                %end
                %for i=1:pn,
                    %fprintf(ifade4d,'%6.0f',No(i,1));
                    %fprintf(ifade4d,'%3.0f',binpopkurb(i,:));
                    %fprintf(ifade4d,'\n');
                %end
```

```
        %fprintf(ifade4d,'\n');


        %-------------------------------------------------
binpop=reprobin;
decpop=bi2de(binpop,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop1(i,j)=binpop(i,j);
    end
end
decpop1=bi2de(binpop1,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop2(i,j)=binpop(i,bitsay1+j);
    end
end
decpop2=bi2de(binpop2,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop3(i,j)=binpop(i,2*bitsay1+j);
    end
end
decpop3=bi2de(binpop3,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop4(i,j)=binpop(i,3*bitsay1+j);
    end
end
decpop4=bi2de(binpop4,'left-msb');

 for i=1:pn,
    for j=1:bitsay1,
        binpop5(i,j)=binpop(i,4*bitsay1+j);
    end
end
decpop5=bi2de(binpop5,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop6(i,j)=binpop(i,5*bitsay1+j);
    end
end
decpop6=bi2de(binpop6,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop7(i,j)=binpop(i,6*bitsay1+j);
    end
end
decpop7=bi2de(binpop7,'left-msb');

  for i=1:pn,
```

```
    for j=1:bitsay1,
        binpop8(i,j)=binpop(i,7*bitsay1+j);
    end
end
decpop8=bi2de(binpop8,'left-msb');

  for i=1:pn,
    for j=1:bitsay1,
        binpop9(i,j)=binpop(i,8*bitsay1+j);
    end
end
decpop9=bi2de(binpop9,'left-msb');

 for i=1:pn,
    for j=1:bitsay1,
        binpop10(i,j)=binpop(i,9*bitsay1+j);
    end
end
decpop10=bi2de(binpop10,'left-msb');

 for i=1:pn,
    for j=1:bitsay1,
        binpop11(i,j)=binpop(i,10*bitsay1+j);
    end
end
decpop11=bi2de(binpop11,'left-msb');

 for i=1:pn,
    for j=1:bitsay1,
        binpop12(i,j)=binpop(i,11*bitsay1+j);
    end
end
decpop12=bi2de(binpop12,'left-msb');

 for i=1:pn,
    for j=1:bitsay1,
        binpop13(i,j)=binpop(i,12*bitsay1+j);
    end
end
decpop13=bi2de(binpop13,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop1d(i,j)=binpop(i,13*bitsay1+j);
    end
end
decpop1d=bi2de(binpop1d,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop2d(i,j)=binpop(i,14*bitsay1+j);
    end
end
decpop2d=bi2de(binpop2d,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
```

```
        binpop3d(i,j)=binpop(i,15*bitsay1+j);
    end
end
decpop3d=bi2de(binpop3d,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop4d(i,j)=binpop(i,16*bitsay1+j);
    end
end
decpop4d=bi2de(binpop4d,'left-msb');

 for i=1:pn,
    for j=1:bitsay1,
        binpop5d(i,j)=binpop(i,17*bitsay1+j);
    end
end
decpop5d=bi2de(binpop5d,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop6d(i,j)=binpop(i,18*bitsay1+j);
    end
end
decpop6d=bi2de(binpop6d,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop7d(i,j)=binpop(i,19*bitsay1+j);
    end
end
decpop7d=bi2de(binpop7d,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop8d(i,j)=binpop(i,20*bitsay1+j);
    end
end
decpop8d=bi2de(binpop8d,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop9d(i,j)=binpop(i,21*bitsay1+j);
    end
end
decpop9d=bi2de(binpop9d,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop10d(i,j)=binpop(i,22*bitsay1+j);
    end
end
decpop10d=bi2de(binpop10d,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop11d(i,j)=binpop(i,23*bitsay1+j);
```

```
    end
end
decpop11d=bi2de(binpop11d,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop12d(i,j)=binpop(i,24*bitsay1+j);
    end
end
decpop12d=bi2de(binpop12d,'left-msb');

for i=1:pn,
    for j=1:bitsay1,
        binpop13d(i,j)=binpop(i,25*bitsay1+j);
    end
end
decpop13d=bi2de(binpop13d,'left-msb');

bitsayk=bitsay1t;
for i=1:pn,
    for j=1:bitsay2,
        binpop14(i,j)=binpop(i,bitsayk+j);
    end
end
decpop14=bi2de(binpop14,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop15(i,j)=binpop(i,bitsayk+j);
    end
end
decpop15=bi2de(binpop15,'left-msb');


 bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop16(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop16=bi2de(binpop16,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop17(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop17=bi2de(binpop17,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop18(i,j)=binpop(i,bitsayk+j);
    end
 end
```

```
decpop18=bi2de(binpop18,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop19(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop19=bi2de(binpop19,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop20(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop20=bi2de(binpop20,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop21(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop21=bi2de(binpop21,'left-msb');


bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop22(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop22=bi2de(binpop22,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop23(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop23=bi2de(binpop23,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop24(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop24=bi2de(binpop24,'left-msb');

 bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop25(i,j)=binpop(i,bitsayk+j);
    end
 end
```

```
decpop25=bi2de(binpop25,'left-msb');

 bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop26(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop26=bi2de(binpop26,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop27(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop27=bi2de(binpop27,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop28(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop28=bi2de(binpop28,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop29(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop29=bi2de(binpop29,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop30(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop30=bi2de(binpop30,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop31(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop31=bi2de(binpop31,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop32(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop32=bi2de(binpop32,'left-msb');
```

```
bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop33(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop33=bi2de(binpop33,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop34(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop34=bi2de(binpop34,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop35(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop35=bi2de(binpop35,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop36(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop36=bi2de(binpop36,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop37(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop37=bi2de(binpop37,'left-msb');

bitsayk=bitsayk+bitsay2;
 for i=1:pn,
    for j=1:bitsay2,
        binpop38(i,j)=binpop(i,bitsayk+j);
    end
 end
decpop38=bi2de(binpop38,'left-msb');

maximum=ft(1,1);
ii=1;
for i=2:pn,
    if ft(i,1)>=maximum,
        maximum=ft(i,1);
        ii=i;
    end
end
```

```
%       fprintf(ifade5,'maximum f(t)(minimum e değeri) değeri
:   ');
 %      fprintf(ifade5,'%1.5f',maximum);
 %      fprintf(ifade5,'\n');
  %     fprintf(ifade5,'maximum f(t) yi sağlayan x1 değeri   :
');
   %    fprintf(ifade5,'%1.2f',x1(ii,1));
   %    fprintf(ifade5,'\n');
   %    fprintf(ifade5,'maximum f(t) yi sağlayan x2değeri   :  ');
    %   fprintf(ifade5,'%1.2f',x2(ii,1));
     %  fprintf(ifade5,'\n');
      % fprintf(ifade5,'maximum f(t) yi sağlayan x3 değeri   :
');
       %fprintf(ifade5,'%1.2f',x3(ii,1));
      % fprintf(ifade5,'\n');
      % fprintf(ifade5,'maximum f(t) yi sağlayan x4 değeri   :
');

      %fprintf(ifade5,'%1.2f',x4(ii,1));
      % fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x5 değeri   :
');
       %fprintf(ifade5,'%1.2f',x5(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x6 değeri   :
');
       %fprintf(ifade5,'%1.2f',x6(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x7 değeri   :
');
       %fprintf(ifade5,'%1.2f',x7(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x8 değeri   :
');
       %fprintf(ifade5,'%1.2f',x8(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x9 değeri   :
');
       %fprintf(ifade5,'%1.2f',x9(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x10 değeri   :
');
       %fprintf(ifade5,'%1.2f',x10(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x11 değeri   :
');
       %fprintf(ifade5,'%1.2f',x11(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x12 değeri   :
');
       %fprintf(ifade5,'%1.2f',x12(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x13 değeri   :
');
       %fprintf(ifade5,'%1.2f',x13(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x14 değeri   :
');
       %fprintf(ifade5,'%1.2f',x14(ii,1));
```

```
        %fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x1d değeri   :
');
        %fprintf(ifade5,'%1.2f',x1d(ii,1));
        %fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x2d değeri   :
');
        %fprintf(ifade5,'%1.2f',x2d(ii,1));
        %fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x3d değeri   :
');
        %fprintf(ifade5,'%1.2f',x3d(ii,1));
        %fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x4d değeri   :
');
        %fprintf(ifade5,'%1.2f',x4d(ii,1));
        %fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x5d değeri   :
');
        %fprintf(ifade5,'%1.2f',x5d(ii,1));
        %fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x6d değeri   :
');
        %fprintf(ifade5,'%1.2f',x6d(ii,1));
        %fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x7d değeri   :
');
        %fprintf(ifade5,'%1.2f',x7d(ii,1));
        %fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x8d değeri   :
');
        %fprintf(ifade5,'%1.2f',x8d(ii,1));
        %fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x9d değeri   :
');
        %fprintf(ifade5,'%1.2f',x9d(ii,1));
       % fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x10d değeri   :
');
        %fprintf(ifade5,'%1.2f',x10d(ii,1));
        %fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x11d değeri   :
');
        %fprintf(ifade5,'%1.2f',x11d(ii,1));
        %fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x12d değeri   :
');
        %fprintf(ifade5,'%1.2f',x12d(ii,1));
        %fprintf(ifade5,'\n');
        %fprintf(ifade5,'maximum f(t) yi sağlayan x13d değeri   :
');
        %fprintf(ifade5,'%1.2f',x13d(ii,1));
        %fprintf(ifade5,'\n');

        %fprintf(ifade5,'maximum f(t) yi sağlayan x15 değeri   :
');
        %fprintf(ifade5,'%1.2f',x15(ii,1));
       % fprintf(ifade5,'\n');
```

```
       %fprintf(ifade5,'maximum f(t) yi sağlayan x16 değeri   :
');
      % fprintf(ifade5,'%1.2f',x16(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x17 değeri   :
');
       %fprintf(ifade5,'%1.2f',x17(ii,1));
      % fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x18 değeri   :
');
       %fprintf(ifade5,'%1.2f',x18(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x19 değeri   :
');
       %fprintf(ifade5,'%1.2f',x19(ii,1));
      % fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x20 değeri   :
');
       %fprintf(ifade5,'%1.2f',x20(ii,1));
       %fprintf(ifade5,'\n');
      % fprintf(ifade5,'maximum f(t) yi sağlayan x21 değeri   :
');
       %fprintf(ifade5,'%1.2f',x21(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x22 değeri   :
');
       %fprintf(ifade5,'%1.2f',x22(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x23 değeri   :
');
       %fprintf(ifade5,'%1.2f',x23(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x24 değeri   :
');
       %fprintf(ifade5,'%1.2f',x24(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x25 değeri   :
');
       %fprintf(ifade5,'%1.2f',x25(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x26 değeri   :
');
       %fprintf(ifade5,'%1.2f',x26(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x27 değeri   :
');
      % fprintf(ifade5,'%1.2f',x27(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x28 değeri   :
');
       %fprintf(ifade5,'%1.2f',x28(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x29 değeri   :
');
       %fprintf(ifade5,'%1.2f',x29(ii,1));
       %fprintf(ifade5,'\n');
      % fprintf(ifade5,'maximum f(t) yi sağlayan x30 değeri   :
');
       %fprintf(ifade5,'%1.2f',x30(ii,1));
```

```
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x31 değeri   :
');
      % fprintf(ifade5,'%1.2f',x31(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x32 değeri   :
');
       %fprintf(ifade5,'%1.2f',x32(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x33 değeri   :
');
       %fprintf(ifade5,'%1.2f',x33(ii,1));
      % fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x34 değeri   :
');
       %fprintf(ifade5,'%1.2f',x34(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x35 değeri   :
');
       %fprintf(ifade5,'%1.2f',x35(ii,1));
       %fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x36 değeri   :
');
       %fprintf(ifade5,'%1.2f',x36(ii,1));
       %fprintf(ifade5,'\n');
      % fprintf(ifade5,'maximum f(t) yi sağlayan x37 değeri   :
');
       %fprintf(ifade5,'%1.2f',x37(ii,1));
      % fprintf(ifade5,'\n');
       %fprintf(ifade5,'maximum f(t) yi sağlayan x38 değeri   :
');
       %fprintf(ifade5,'%1.2f',x38(ii,1));
       %fprintf(ifade5,'\n');

       % önceki jenerasyondaki en iyi dizi (minimum e'yi veren)
bir sonraki
       % jenerasyondaki popülasyonun ilk dizisi olarak garanti
altına
       % alınıyor.



x1(1,1)=x1(ii,1);x2(1,1)=x2(ii,1);x3(1,1)=x3(ii,1);x4(1,1)=x4(ii,1
);x5(1,1)=x5(ii,1);

x6(1,1)=x6(ii,1);x7(1,1)=x7(ii,1);x8(1,1)=x8(ii,1);x9(1,1)=x9(ii,1
);x10(1,1)=x10(ii,1);
        x11(1,1)=x11(ii,1); x12(1,1)=x12(ii,1);x13(1,1)=x13(ii,1);


x1d(1,1)=x1d(ii,1);x2d(1,1)=x2d(ii,1);x3d(1,1)=x3d(ii,1);x4d(1,1)=
x4d(ii,1);x5d(1,1)=x5d(ii,1);

x6d(1,1)=x6d(ii,1);x7d(1,1)=x7d(ii,1);x8d(1,1)=x8d(ii,1);x9d(1,1)=
x9d(ii,1);x10d(1,1)=x10d(ii,1);

x11d(1,1)=x11d(ii,1);x12d(1,1)=x12d(ii,1);x13d(1,1)=x13d(ii,1);
```

```
        x14(1,1)=x14(ii,1);x15(1,1)=x15(ii,1);
x16(1,1)=x16(ii,1);x17(1,1)=x17(ii,1);x18(1,1)=x18(ii,1);
        x19(1,1)=x19(ii,1);
x20(1,1)=x20(ii,1);x21(1,1)=x21(ii,1);x22(1,1)=x22(ii,1);x23(1,1)=
x23(ii,1);x24(1,1)=x24(ii,1);
        x25(1,1)=x25(ii,1);
x26(1,1)=x26(ii,1);x27(1,1)=x27(ii,1);x28(1,1)=x28(ii,1);
        x29(1,1)=x29(ii,1);
x30(1,1)=x30(ii,1);x31(1,1)=x31(ii,1);x32(1,1)=x32(ii,1);
        x33(1,1)=x33(ii,1);
x34(1,1)=x34(ii,1);x35(1,1)=x35(ii,1);x36(1,1)=x36(ii,1);
        x37(1,1)=x37(ii,1);x38(1,1)=x38(ii,1);


        for i=2:1:pn
            x1(i,1)=xmin+(yhas1*decpop1(i,1));
        end

        for i=2:1:pn
            x2(i,1)=xmin+(yhas1*decpop2(i,1));
        end

        for i=2:1:pn
            x3(i,1)=xmin+(yhas1*decpop3(i,1));
        end

        for i=2:1:pn
            x4(i,1)=xmin+(yhas1*decpop4(i,1));
        end

        for i=2:1:pn
            x5(i,1)=xmin+(yhas1*decpop5(i,1));
        end

        for i=2:1:pn
            x6(i,1)=xmin+(yhas1*decpop6(i,1));
        end

        for i=2:1:pn
            x7(i,1)=xmin+(yhas1*decpop7(i,1));
        end

        for i=2:1:pn
            x8(i,1)=xmin+(yhas1*decpop8(i,1));
        end

        for i=2:1:pn
            x9(i,1)=xmin+(yhas1*decpop9(i,1));
        end

        for i=2:1:pn
            x10(i,1)=xmin+(yhas1*decpop10(i,1));
        end

        for i=2:1:pn
            x11(i,1)=xmin+(yhas1*decpop11(i,1));
```

```
end

for i=2:1:pn
    x12(i,1)=xmin+(yhas1*decpop12(i,1));
end

for i=2:1:pn
    x13(i,1)=xmin+(yhas1*decpop13(i,1));
end

%//////


for i=2:1:pn
    x1d(i,1)=xmin+(yhas1*decpop1d(i,1));
end

for i=2:1:pn
    x2d(i,1)=xmin+(yhas1*decpop2d(i,1));
end

for i=2:1:pn
    x3d(i,1)=xmin+(yhas1*decpop3d(i,1));
end

for i=2:1:pn
    x4d(i,1)=xmin+(yhas1*decpop4d(i,1));
end

for i=2:1:pn
    x5d(i,1)=xmin+(yhas1*decpop5d(i,1));
end

for i=2:1:pn
    x6d(i,1)=xmin+(yhas1*decpop6d(i,1));
end

for i=2:1:pn
    x7d(i,1)=xmin+(yhas1*decpop7d(i,1));
end

for i=2:1:pn
    x8d(i,1)=xmin+(yhas1*decpop8d(i,1));
end

for i=2:1:pn
    x9d(i,1)=xmin+(yhas1*decpop9d(i,1));
end

for i=2:1:pn
    x10d(i,1)=xmin+(yhas1*decpop10d(i,1));
end

for i=2:1:pn
    x11d(i,1)=xmin+(yhas1*decpop11d(i,1));
end
```

```
for i=2:1:pn
    x12d(i,1)=xmin+(yhas1*decpop12d(i,1));
end

for i=2:1:pn
    x13d(i,1)=xmin+(yhas1*decpop13d(i,1));
end
%/////

for i=2:1:pn
    x14(i,1)=0+(yhas2*decpop14(i,1));
end

for i=2:1:pn
    x15(i,1)=0+(yhas2*decpop15(i,1));
end

for i=2:1:pn
    x16(i,1)=0+(yhas2*decpop16(i,1));
end

for i=2:1:pn
    x17(i,1)=0;
end

for i=2:1:pn
    x18(i,1)=0;
end

for i=2:1:pn
    x19(i,1)=0+(yhas2*decpop19(i,1));
end

for i=2:1:pn
    x20(i,1)=0+(yhas2*decpop20(i,1));
end

for i=2:1:pn
    x21(i,1)=0+(yhas2*decpop21(i,1));
end

for i=2:1:pn
    x22(i,1)=kamin+(yhas2*decpop22(i,1));
end

for i=2:1:pn
    x23(i,1)=kamin+(yhas2*decpop23(i,1));
end

for i=2:1:pn
    x24(i,1)=0+(yhas2*decpop24(i,1));
end

for i=2:1:pn
    x25(i,1)=0+(yhas2*decpop25(i,1));
end
```

```
for i=2:1:pn
    x26(i,1)=0;
end

for i=2:1:pn
    x27(i,1)=kamin+(yhas2*decpop27(i,1));
end

for i=2:1:pn
    x28(i,1)=kamin+(yhas2*decpop28(i,1));
end

for i=2:1:pn
    x29(i,1)=0+(yhas2*decpop29(i,1));
end

for i=2:1:pn
    x30(i,1)=0+(yhas2*decpop30(i,1));
end

for i=2:1:pn
    x31(i,1)=kamin+(yhas2*decpop31(i,1));
end

for i=2:1:pn
    x32(i,1)=kamin+(yhas2*decpop32(i,1));
end

for i=2:1:pn
    x33(i,1)=kamin+(yhas2*decpop33(i,1));
end

for i=2:1:pn
    x34(i,1)=0;
end

for i=2:1:pn
    x35(i,1)=0;
end

for i=2:1:pn
    x36(i,1)=kamin+(yhas2*decpop36(i,1));
end

for i=2:1:pn
    x37(i,1)=kamin+(yhas2*decpop37(i,1));
end

for i=2:1:pn
    x38(i,1)=kamin+(yhas2*decpop38(i,1));
end

clear reprobin
```

```
%fclose(ifadea);
%fclose(ifadeb);
%fclose(ifadec);
%fclose(ifaded);
%fclose(ifade1);
%fclose(ifade2);
%fclose(ifade3a);
%fclose(ifade3b);
%fclose(ifade3c);
%fclose(ifade3d);
%fclose(ifade4a);
%fclose(ifade4b);
%fclose(ifade4c);
%fclose(ifade4d);
%fclose(ifade5);

    ip=1;
  for kid=1:ntd,
     ip=ip+1;
     deltaeom(ip,1)=deltaeo(ip,ii);
     eom(ip,1)=eo(ip,ii);
     x1om(ip,1)=x1o(ip,ii);
     x2om(ip,1)=x2o(ip,ii);
  end

  if jen==1;
      iip=1;ip=1;
   for kid=1:ntd,
       iip=iip+1;ip=ip+1;
       deltaeomc(iip,1)=deltaeom(ip,1);
       eomc(iip,1)=eom(ip,1);
       x1omc(iip,1)=x1om(ip,1);
       x2omc(iip,1)=x2om(ip,1);
   end
else
    for kid=iip:ntd,
       iip=iip+1;
       deltaeomc(iip,1)=deltaeom(iip,1);
       eomc(iip,1)=eom(iip,1);
       x1omc(iip,1)=x1om(iip,1);
       x2omc(iip,1)=x2om(iip,1);
   end
end

end

%cizim

     ip=1;
  for kid=1:ntd,
     ip=ip+1;
     deltaeom(ip,1)=deltaeo(ip,ii);
     eom(ip,1)=eo(ip,ii);
     x1om(ip,1)=x1o(ip,ii);
     x2om(ip,1)=x2o(ip,ii);
  end
```

```
% her jenerasyondaki en iyi dizi için grafikler

plot(t,eomc);
hold on
 figure(2)
plot(t,deltaeomc);
hold on
 figure(3)
plot(t,x1omc);
hold on
 figure(4)
plot(t,x2omc);

% son jenerasyondaki en iyi dizi için grafikler

 figure(5)
plot(t,eom);
 hold on
  figure(6)
 plot(t,deltaeom);
 hold on
  figure(7)
 plot(t,x1om);
 hold on
  figure(8)
 plot(t,x2om);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%Genetik1 programı%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%xmin      :Üyelik fonksiyonu minimum değeri
%xmax      :Üyelik fonksiyonu maksimum değeri
%has       :Hassasiyet
%pn        :Nüfus sayısı (popülasyondaki birey sayısı)
%kamin     :İrtifa dümeni açısı minimum değeri
%kamax     :İrtifa dümeni açısı maksimum değeri
%Pc        :Çaprazlama oranı
%Pm        :Mutasyon oranı
%gn        :Jenerasyon sayısı (durdurma kriteri olarak
kullanılacak)
%aralik1   :x'in tanım aralığı için kaç sayı yazılabilir
%aralik2   :ka'nın tanım aralığı için kaç sayı yazılabilir
%bitsay1   :x'in bit sayısı
%bitsay2   :ka'nın bit sayısı
%bitsay1t  :Tüm x(üyelik fonks.sınır) değerleriiçin bit sayısı
%bitsay2t  :Tüm ka(kural ağırlık-irtifa dümeni açı)değerleri için
bit sayısı
%bitsayt   :Tüm x ve ka bitleri yanyana geldiğinde oluşan elemanın
bit sayısı
%x,xd      :Nüfus içindeki bireylerin (üyelik fonksiyon sınır
değerlerini ve kural ağırlık değerlerini içeren) desimal
değerlerini tutan vektör
%binpop    :Nüfus içindeki bireylerin binary değerlerini tutan
vektör
%decpop    :Nüfus içindeki bireylerin desimal karşılıklarını tutan
vektör

%GA parametrelerinin kullanıcıdan alınması

xmin=input('Üyelik fonksiyonu minimum
değeri(Xmin):');fprintf('\n')
xmax=input('Üyelik fonksiyonu maksimum
değeri(Xmax):');fprintf('\n')
has=input('Hassasiyet:');fprintf('\n')
pn=input('Nüfus sayısı:');fprintf('\n')
kamin=input('İrtifa dümeni açısı minimum
değeri(kamin):');fprintf('\n')
kamax=input('İrtifa dümeni açısı maksimum
değeri(kamax):');fprintf('\n')
Pc=input('Çaprazlama oranı:');fprintf('\n')
Pm=input('Mutasyon oranı:');fprintf('\n')
gn=input('Jenerasyon sayısı:');fprintf('\n')

%-------------------------------------------------------------
%hata ve hatanın değişimi için parametre değerlerinin bulunması

aralik1=(xmax-xmin)/has;
bitsay1=length(de2bi(aralik1,'left-msb'));
ebs1=bi2de(ones(1,bitsay1));
yhas1=(xmax-xmin)/ebs1;

%hata ve hatanın değişimi için toplam 13*2=26 tane parametre
```

```
bitsay1t=26*bitsay1;

%kural ağırlık değerleri için parametre değerlerinin bulunması

aralik2=(kamax-kamin)/has;
bitsay2=length(de2bi(aralik2,'left-msb'));
ebs2=bi2de(ones(1,bitsay2));
yhas2=(kamax-kamin)/ebs2;

%25 adet kural ağırlık değeri
bitsay2t=25*bitsay2;

%toplam bit sayısı
bitsayt=bitsay1t+bitsay2t;

%-----------------------------------------------------------------
---------

% crossovera uğrayacak birey sayısının belirlenmesi-----------
crs=pn*Pc;
% crossovera uğrayacak birey sayısı çift yapılıyor-----------
if rem(round(crs),2)==0,
    crs=round(crs),
else
    crs=round(crs)+1;
end
disp(strcat('crossovera uğrayacak birey sayısı:',num2str(crs)));
fprintf('\n')
%-----------------------------------------------------------

%mutasyona uğrayacak bit sayısının belirlenmesi-----------
ms=round(pn*Pm*bitsayt);
disp(strcat('mutasyona uğrayacak bit sayısı:',num2str(ms)))
fprintf('\n')
%-----------------------------------------------------------
% desimal tepe değerlerinin tutulduğu vektörlerin oluşturulması
% pnx1(random)

for i=1:pn,
    x1(i,1)=xmin;
end

a2=[xmin:has:xmax];
x2=randsrc(1,1,a2);
for i=2:pn,
    mm2=randsrc(1,1,a2);
    x2=[x2;mm2];
end

a3=[xmin:has:xmax];
x3=randsrc(1,1,a3);
for i=2:pn,
    mm3=randsrc(1,1,a3);
    x3=[x3;mm3];
end
```

```
a4=[xmin:has:xmax];
x4=randsrc(1,1,a4);
for i=2:pn,
    mm4=randsrc(1,1,a4);
    x4=[x4;mm4];
end

for i=1:pn,
    x5(i,1)=xmax;
end


%tepe değerleri vektörleri matris haline getiriliyor.Sıralama
için.

   x=[x1,x2,x3,x4,x5];

%  x1(i,1)<x2(i,1)<x3(i,1)<x4(i,1)<x5(i,1) olmalı x matrisinin
satırları
%  küçükten büyüğe sıralanıyor.

for i=1:pn,
    y=x(i,:);
    z=sort(y);
    x(i,:)=z;
end

% x matrisinden tek tek tepe değer vektörleri elde ediliyor(pnx1)

x1=x(:,1);x2=x(:,2);x3=x(:,3);x4=x(:,4);x5=x(:,5);


% üyelik fonksiyonu sınır değerleri(desimal random) elde
ediliyor.(pnx1)


a6=[x1(1,1):has:x2(1,1)];
x6=randsrc(1,1,a6);
for i=2:pn,
    a6=[x1(i,1):has:x2(i,1)];
    mm6=randsrc(1,1,a6);
    x6=[x6;mm6];
    end


a7=[x1(1,1):has:x2(1,1)];
x7=randsrc(1,1,a7);
for i=2:pn,
    a7=[x1(i,1):has:x2(i,1)];
    mm7=randsrc(1,1,a7);
    x7=[x7;mm7];
end
```

```
a8=[x2(1,1):has:x3(1,1)];
x8=randsrc(1,1,a8);
for i=2:pn,
    a8=[x2(i,1):has:x3(i,1)];
    mm8=randsrc(1,1,a8);
    x8=[x8;mm8];
end


a9=[x2(1,1):has:x3(1,1)];
x9=randsrc(1,1,a9);
for i=2:pn,
    a9=[x2(i,1):has:x3(i,1)];
    mm9=randsrc(1,1,a9);
    x9=[x9;mm9];
end


a10=[x3(1,1):has:x4(1,1)];
x10=randsrc(1,1,a10);
for i=2:pn,
    a10=[x3(i,1):has:x4(i,1)];
    mm10=randsrc(1,1,a10);
    x10=[x10;mm10];
end


a11=[x3(1,1):has:x4(1,1)];
x11=randsrc(1,1,a11);
for i=2:pn,
    a11=[x3(i,1):has:x4(i,1)];
    mm11=randsrc(1,1,a11);
    x11=[x11;mm11];
end

a12=[x4(1,1):has:xmax];
x12=randsrc(1,1,a12);
for i=2:pn,
    a12=[x4(i,1):has:xmax];
    mm12=randsrc(1,1,a12);
    x12=[x12;mm12];
end


a13=[x4(1,1):has:x5(1,1)];
x13=randsrc(1,1,a13);
for i=2:pn,
    a13=[x4(i,1):has:x5(i,1)];
    mm13=randsrc(1,1,a13);
    x13=[x13;mm13];
end

%hatanın değişimi için  desimal tepe değerlerinin tutulduğu
vektörlerin oluşturulması
% pnx1(random)
```

```
for i=1:pn,
    x1d(i,1)=xmin;
end

a2d=[xmin:has:xmax];
x2d=randsrc(1,1,a2d);
for i=2:pn,
    mm2d=randsrc(1,1,a2d);
    x2d=[x2d;mm2d];
    end

a3d=[xmin:has:xmax];
x3d=randsrc(1,1,a3d);
for i=2:pn,
    mm3d=randsrc(1,1,a3d);
    x3d=[x3d;mm3d];
    end

a4d=[xmin:has:xmax];
x4d=randsrc(1,1,a4d);
for i=2:pn,
    mm4d=randsrc(1,1,a4d);
    x4d=[x4d;mm4d];
end

for i=1:pn,
    x5d(i,1)=xmax;
end


%tepe değerleri vektörleri matris haline getiriliyor. Sıralama
için.

   xd=[x1d,x2d,x3d,x4d,x5d];

%  x1(i,1)<x2(i,1)<x3(i,1)<x4(i,1)<x5(i,1) olmalı x matrisinin
satırları
%  küçükten büyüğe sıralanıyor.

for i=1:pn,
    yd=xd(i,:);
    zd=sort(yd);
    xd(i,:)=zd;
end

% x matrisinden tek tek tepe değer vektörleri elde ediliyor(pnx1)

x1d=xd(:,1);x2d=xd(:,2);x3d=xd(:,3);x4d=xd(:,4);x5d=xd(:,5);

% üyelik fonksiyonu sınır değerleri(desimal random) elde
ediliyor.(pnx1)

a6d=[x1d(1,1):has:x2d(1,1)];
x6d=randsrc(1,1,a6d);
for i=2:pn,
    a6d=[x1d(i,1):has:x2d(i,1)];
```

```
    mm6d=randsrc(1,1,a6d);
    x6d=[x6d;mm6d];
    end


a7d=[x1d(1,1):has:x2d(1,1)];
x7d=randsrc(1,1,a7d);
for i=2:pn,
    a7d=[x1d(i,1):has:x2d(i,1)];
    mm7d=randsrc(1,1,a7d);
    x7d=[x7d;mm7d];
end


a8d=[x2d(1,1):has:x3d(1,1)];
x8d=randsrc(1,1,a8d);
for i=2:pn,
    a8d=[x2d(i,1):has:x3d(i,1)];
    mm8d=randsrc(1,1,a8d);
    x8d=[x8d;mm8d];
end


a9d=[x2d(1,1):has:x3d(1,1)];
x9d=randsrc(1,1,a9d);
for i=2:pn,
    a9d=[x2d(i,1):has:x3d(i,1)];
    mm9d=randsrc(1,1,a9d);
    x9d=[x9d;mm9d];
end


a10d=[x3d(1,1):has:x4d(1,1)];
x10d=randsrc(1,1,a10d);
for i=2:pn,
    a10d=[x3d(i,1):has:x4d(i,1)];
    mm10d=randsrc(1,1,a10d);
    x10d=[x10d;mm10d];
end


a11d=[x3d(1,1):has:x4d(1,1)];
x11d=randsrc(1,1,a11d);
for i=2:pn,
    a11d=[x3d(i,1):has:x4d(i,1)];
    mm11d=randsrc(1,1,a11d);
    x11d=[x11d;mm11d];
end


a12d=[x4d(1,1):has:xmax];
x12d=randsrc(1,1,a12d);
for i=2:pn,
    a12d=[x4d(i,1):has:xmax];
    mm12d=randsrc(1,1,a12d);
    x12d=[x12d;mm12d];
```

```
end


a13d=[x4d(1,1):has:x5d(i,1)];
x13d=randsrc(1,1,a13d);
for i=2:pn,
    a13d=[x4d(i,1):has:x5d(i,1)];
    mm13d=randsrc(1,1,a13d);
    x13d=[x13d;mm13d];
end

%-----kural ağırlık değerlerine ait desimal değerlerin tutulduğu
vektörün oluşturulması
%(pnx1)random

a14=[0:has:kamax];
x14=randsrc(1,1,a14);
for i=2:pn,
    a14=[0:has:kamax];
    mm14=randsrc(1,1,a14);
    x14=[x14;mm14];
end

a15=[0:has:kamax];
x15=randsrc(1,1,a15);
for i=2:pn,
    a15=[0:has:kamax];
    mm15=randsrc(1,1,a15);
    x15=[x15;mm15];
end

a16=[0:has:kamax];
x16=randsrc(1,1,a16);
for i=2:pn,
    a16=[0:has:kamax];
    mm16=randsrc(1,1,a16);
    x16=[x16;mm16];
end



x17=zeros(pn,1);

x18=zeros(pn,1);

a19=[0:has:kamax];
x19=randsrc(1,1,a19);
for i=2:pn,
    a19=[0:has:kamax];
    mm19=randsrc(1,1,a19);
    x19=[x19;mm19];
end

a20=[0:has:kamax];
x20=randsrc(1,1,a20);
for i=2:pn,
```

```
    a20=[0:has:kamax];
    mm20=randsrc(1,1,a20);
    x20=[x20;mm20];
end

a21=[0:has:kamax];
x21=randsrc(1,1,a21);
for i=2:pn,
    a21=[0:has:kamax];
    mm21=randsrc(1,1,a21);
    x21=[x21;mm21];
end

a22=[kamin:has:0];
x22=randsrc(1,1,a22);
for i=2:pn,
    a22=[kamin:has:0];
    mm22=randsrc(1,1,a22);
    x22=[x22;mm22];
end

a23=[kamin:has:0];
x23=randsrc(1,1,a23);
for i=2:pn,
    a23=[kamin:has:0];
    mm23=randsrc(1,1,a23);
    x23=[x23;mm23];
end

a24=[0:has:kamax];
x24=randsrc(1,1,a24);
for i=2:pn,
    a24=[0:has:kamax];
    mm24=randsrc(1,1,a24);
    x24=[x24;mm24];
end

a25=[0:has:kamax];
x25=randsrc(1,1,a25);
for i=2:pn,
    a25=[0:has:kamax];
    mm25=randsrc(1,1,a25);
    x25=[x25;mm25];
end

x26=zeros(pn,1);

a27=[kamin:has:0];
x27=randsrc(1,1,a27);
for i=2:pn,
    a27=[kamin:has:0];
    mm27=randsrc(1,1,a27);
    x27=[x27;mm27];
end

a28=[kamin:has:0];
x28=randsrc(1,1,a28);
```

```
for i=2:pn,
    a28=[kamin:has:0];
    mm28=randsrc(1,1,a28);
    x28=[x28;mm28];
end

a29=[0:has:kamax];
x29=randsrc(1,1,a29);
for i=2:pn,
    a29=[0:has:kamax];
    mm29=randsrc(1,1,a29);
    x29=[x29;mm29];
end

a30=[0:has:kamax];
x30=randsrc(1,1,a30);
for i=2:pn,
    a30=[0:has:kamax];
    mm30=randsrc(1,1,a30);
    x30=[x30;mm30];
end

a31=[kamin:has:0];
x31=randsrc(1,1,a31);
for i=2:pn,
    a31=[kamin:has:0];
    mm31=randsrc(1,1,a31);
    x31=[x31;mm31];
end

a32=[kamin:has:0];
x32=randsrc(1,1,a32);
for i=2:pn,
    a32=[kamin:has:0];
    mm32=randsrc(1,1,a32);
    x32=[x32;mm32];
end

a33=[kamin:has:0];
x33=randsrc(1,1,a33);
for i=2:pn,
    a33=[kamin:has:0];
    mm33=randsrc(1,1,a33);
    x33=[x33;mm33];
end

x34=zeros(pn,1);

x35=zeros(pn,1);

a36=[kamin:has:0];
x36=randsrc(1,1,a36);
for i=2:pn,
    a36=[kamin:has:0];
    mm36=randsrc(1,1,a36);
    x36=[x36;mm36];
end
```

```
a37=[kamin:has:0];
x37=randsrc(1,1,a37);
for i=2:pn,
    a37=[kamin:has:0];
    mm37=randsrc(1,1,a37);
    x37=[x37;mm37];
end

a38=[kamin:has:0];
x38=randsrc(1,1,a38);
for i=2:pn,
    a38=[kamin:has:0];
    mm38=randsrc(1,1,a38);
    x38=[x38;mm38];
end


%binary değerlerin tutulduğu vektörlerin oluşturulması,pnx1(hata)

for i=1:1:pn,
   decpop1(i,1)=round((x1(i,1)-xmin)/yhas1);
end
binpop1=de2bi(decpop1,bitsay1,'left-msb');

for i=1:1:pn,
   decpop2(i,1)=round((x2(i,1)-xmin)/yhas1);
end
binpop2=de2bi(decpop2,bitsay1,'left-msb');

for i=1:1:pn,
   decpop3(i,1)=round((x3(i,1)-xmin)/yhas1);
end
binpop3=de2bi(decpop3,bitsay1,'left-msb');

for i=1:1:pn,
   decpop4(i,1)=round((x4(i,1)-xmin)/yhas1);
end
binpop4=de2bi(decpop4,bitsay1,'left-msb');

for i=1:1:pn,
   decpop5(i,1)=round((x5(i,1)-xmin)/yhas1);
end
binpop5=de2bi(decpop5,bitsay1,'left-msb');


for i=1:1:pn,
   decpop6(i,1)=round((x6(i,1)-xmin)/yhas1);
end
binpop6=de2bi(decpop6,bitsay1,'left-msb');

for i=1:1:pn,
   decpop7(i,1)=round((x7(i,1)-xmin)/yhas1);
end
binpop7=de2bi(decpop7,bitsay1,'left-msb');

for i=1:1:pn,
   decpop8(i,1)=round((x8(i,1)-xmin)/yhas1);
```

```
end
binpop8=de2bi(decpop8,bitsay1,'left-msb');

for i=1:1:pn,
   decpop9(i,1)=round((x9(i,1)-xmin)/yhas1);
end
binpop9=de2bi(decpop9,bitsay1,'left-msb');

for i=1:1:pn,
   decpop10(i,1)=round((x10(i,1)-xmin)/yhas1);
end
binpop10=de2bi(decpop10,bitsay1,'left-msb');

for i=1:1:pn,
   decpop11(i,1)=round((x11(i,1)-xmin)/yhas1);
end
binpop11=de2bi(decpop11,bitsay1,'left-msb');

for i=1:1:pn,
   decpop12(i,1)=round((x12(i,1)-xmin)/yhas1);
end
binpop12=de2bi(decpop12,bitsay1,'left-msb');

for i=1:1:pn,
   decpop13(i,1)=round((x13(i,1)-xmin)/yhas1);
end
binpop13=de2bi(decpop13,bitsay1,'left-msb');

%binary değerlerin tutulduğu vektörlerin
oluşturulması,pnx1(hatanın değişimi)


for i=1:1:pn,
   decpop1d(i,1)=round((x1d(i,1)-xmin)/yhas1);
end
binpop1d=de2bi(decpop1d,bitsay1,'left-msb');

for i=1:1:pn,
   decpop2d(i,1)=round((x2d(i,1)-xmin)/yhas1);
end
binpop2d=de2bi(decpop2d,bitsay1,'left-msb');

for i=1:1:pn,
   decpop3d(i,1)=round((x3d(i,1)-xmin)/yhas1);
end
binpop3d=de2bi(decpop3d,bitsay1,'left-msb');

for i=1:1:pn,
   decpop4d(i,1)=round((x4d(i,1)-xmin)/yhas1);
end
binpop4d=de2bi(decpop4d,bitsay1,'left-msb');

for i=1:1:pn,
   decpop5d(i,1)=round((x5d(i,1)-xmin)/yhas1);
end
binpop5d=de2bi(decpop5d,bitsay1,'left-msb');
```

```
for i=1:1:pn,
   decpop6d(i,1)=round((x6d(i,1)-xmin)/yhas1);
end
binpop6d=de2bi(decpop6d,bitsay1,'left-msb');

for i=1:1:pn,
   decpop7d(i,1)=round((x7d(i,1)-xmin)/yhas1);
end
binpop7d=de2bi(decpop7d,bitsay1,'left-msb');

for i=1:1:pn,
   decpop8d(i,1)=round((x8d(i,1)-xmin)/yhas1);
end
binpop8d=de2bi(decpop8d,bitsay1,'left-msb');

for i=1:1:pn,
   decpop9d(i,1)=round((x9d(i,1)-xmin)/yhas1);
end
binpop9d=de2bi(decpop9d,bitsay1,'left-msb');

for i=1:1:pn,
   decpop10d(i,1)=round((x10d(i,1)-xmin)/yhas1);
end
binpop10d=de2bi(decpop10d,bitsay1,'left-msb');

for i=1:1:pn,
   decpop11d(i,1)=round((x11d(i,1)-xmin)/yhas1);
end
binpop11d=de2bi(decpop11d,bitsay1,'left-msb');

for i=1:1:pn,
   decpop12d(i,1)=round((x12d(i,1)-xmin)/yhas1);
end
binpop12d=de2bi(decpop12d,bitsay1,'left-msb');

for i=1:1:pn,
   decpop13d(i,1)=round((x13d(i,1)-xmin)/yhas1);
end
binpop13d=de2bi(decpop13d,bitsay1,'left-msb');

%*****
for i=1:1:pn,
    decpop14(i,1)=round((x14(i,1)-0)/yhas2);
end
binpop14=de2bi(decpop14,bitsay2,'left-msb');

for i=1:1:pn,
    decpop15(i,1)=round((x15(i,1)-0)/yhas2);
end
binpop15=de2bi(decpop15,bitsay2,'left-msb');

for i=1:1:pn,
    decpop16(i,1)=round((x16(i,1)-0)/yhas2);
end
```

```
binpop16=de2bi(decpop16,bitsay2,'left-msb');

for i=1:1:pn,
    decpop17(i,1)=round((x17(i,1)-kamin)/yhas2);
end
binpop17=de2bi(decpop17,bitsay2,'left-msb');

for i=1:1:pn,
    decpop18(i,1)=round((x18(i,1)-kamin)/yhas2);
end
binpop18=de2bi(decpop18,bitsay2,'left-msb');

for i=1:1:pn,
    decpop19(i,1)=round((x19(i,1)-0)/yhas2);
end
binpop19=de2bi(decpop19,bitsay2,'left-msb');

for i=1:1:pn,
    decpop20(i,1)=round((x20(i,1)-0)/yhas2);
end
binpop20=de2bi(decpop20,bitsay2,'left-msb');

for i=1:1:pn,
    decpop21(i,1)=round((x21(i,1)-0)/yhas2);
end
binpop21=de2bi(decpop21,bitsay2,'left-msb');

for i=1:1:pn,
    decpop22(i,1)=round((x22(i,1)-kamin)/yhas2);
end
binpop22=de2bi(decpop22,bitsay2,'left-msb');

for i=1:1:pn,
    decpop23(i,1)=round((x23(i,1)-kamin)/yhas2);
end
binpop23=de2bi(decpop23,bitsay2,'left-msb');

for i=1:1:pn,
    decpop24(i,1)=round((x24(i,1)-0)/yhas2);
end
binpop24=de2bi(decpop24,bitsay2,'left-msb');

for i=1:1:pn,
    decpop25(i,1)=round((x25(i,1)-0)/yhas2);
end
binpop25=de2bi(decpop25,bitsay2,'left-msb');

for i=1:1:pn,
    decpop26(i,1)=round((x26(i,1)-kamin)/yhas2);
end
binpop26=de2bi(decpop26,bitsay2,'left-msb');

for i=1:1:pn,
    decpop27(i,1)=round((x27(i,1)-kamin)/yhas2);
end
binpop27=de2bi(decpop27,bitsay2,'left-msb');
```

```
for i=1:1:pn,
    decpop28(i,1)=round((x28(i,1)-kamin)/yhas2);
end
binpop28=de2bi(decpop28,bitsay2,'left-msb');

for i=1:1:pn,
    decpop29(i,1)=round((x29(i,1)-0)/yhas2);
end
binpop29=de2bi(decpop29,bitsay2,'left-msb');

for i=1:1:pn,
    decpop30(i,1)=round((x30(i,1)-0)/yhas2);
end
binpop30=de2bi(decpop30,bitsay2,'left-msb');

for i=1:1:pn,
    decpop31(i,1)=round((x31(i,1)-kamin)/yhas2);
end
binpop31=de2bi(decpop31,bitsay2,'left-msb');

for i=1:1:pn,
    decpop32(i,1)=round((x32(i,1)-kamin)/yhas2);
end
binpop32=de2bi(decpop32,bitsay2,'left-msb');

for i=1:1:pn,
    decpop33(i,1)=round((x33(i,1)-kamin)/yhas2);
end
binpop33=de2bi(decpop33,bitsay2,'left-msb');

for i=1:1:pn,
    decpop34(i,1)=round((x34(i,1)-kamin)/yhas2);
end
binpop34=de2bi(decpop34,bitsay2,'left-msb');

for i=1:1:pn,
    decpop35(i,1)=round((x35(i,1)-kamin)/yhas2);
end
binpop35=de2bi(decpop35,bitsay2,'left-msb');

for i=1:1:pn,
    decpop36(i,1)=round((x36(i,1)-kamin)/yhas2);
end
binpop36=de2bi(decpop36,bitsay2,'left-msb');

for i=1:1:pn,
    decpop37(i,1)=round((x37(i,1)-kamin)/yhas2);
end
binpop37=de2bi(decpop37,bitsay2,'left-msb');

for i=1:1:pn,
    decpop38(i,1)=round((x38(i,1)-kamin)/yhas2);
end
binpop38=de2bi(decpop38,bitsay2,'left-msb');
```

```
binpopa=[binpop1 binpop2 binpop3 binpop4 binpop5 binpop6 binpop7
binpop8 binpop9 binpop10 binpop11 binpop12 binpop13];
binpopb=[binpop1d binpop2d binpop3d binpop4d binpop5d binpop6d
binpop7d binpop8d binpop9d binpop10d binpop11d binpop12d
binpop13d];
binpopc=[binpop14 binpop15 binpop16 binpop17 binpop18 binpop19
binpop20 binpop21 binpop22 binpop23 binpop24 binpop25 binpop26];
binpopd=[binpop27 binpop28 binpop29 binpop30 binpop31 binpop32
binpop33 binpop34 binpop35 binpop36 binpop37 binpop38];
binpop=[binpopa binpopb binpopc binpopd];
decpop=bi2de(binpop,'left-msb');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%ctrf3g programı%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%hd      :İstenen irtifa
%hdp     :İstenen irtifanın türevi
%e       :Hata dizisi
%ep      :Hatanın türevi dizisi
%deltae  :İrtifa dümeni açısı
%MHNCKe,MHNKe,MHYe,MHPBe,MHPCBe:Hata Üyelik fonksiyonu sınır değer
%dizileri
%MHDNCKe,MHDNKe,MDHYe,MHDPBe,MHDPCBe:Hatanın değişimi üyelik
fonksiyonu
%sınır değer dizileri
%%%%%%%%%%%%%%%%%%%%
%DESIRED HEIGHT
%%%%%%%%%%%%%%%%%%%%%

hd=6100+20*sin(0.207*time);
hdp=20*0.207*cos(0.207*time);

%hd=12200;
%hdp=0;
%%%%%%%%%%%%%%%%%%
%TRACKiNG ERRORS
%%%%%%%%%%%%%%%%%%
e(i,1)=hd-xx(1);
ep(i,1)=hdp-xx(2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%COMPUTATION OF elevator angle
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('\n')
%e,ep

tdeltaepay(i,1)=0;
tdeltaepayda(i,1)=0;

%%e=HNCK durumu%%%%%%%%%%
bolge=1;
if ((e(i,1)>=x1(i,1))&(e(i,1)<=x6(i,1))) e1(i,1)=1;
    MHNCKe(i,1)=(x6(i,1)-e(i,1))/(x6(i,1)-x1(i,1));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if ((ep(i,1)>=x1d(i,1))&(ep(i,1)<=x6d(i,1)))  e2(i,1)=1;
        MHDNCKep(i,1)=(x6d(i,1)-ep(i,1))/(x6d(i,1)-x1d(i,1));
       % e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x7d(i,1))&(ep(i,1)<=x2d(i,1))) e2(i,1)=2;
        MHDNKep(i,1)=(ep(i,1)-x7d(i,1))/(x2d(i,1)-x7d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x2d(i,1))&(ep(i,1)<=x8d(i,1))) e2(i,1)=2;
        MHDNKep(i,1)=(x8d(i,1)-ep(i,1))/(x8d(i,1)-x2d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x9d(i,1))&(ep(i,1)<=x3d(i,1)))
```

```
        e2(i,1)=3;
        MHDYep(i,1)=(ep(i,1)-x9d(i,1))/(x3d(i,1)-x9d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x3d(i,1))&(ep(i,1)<=x10d(i,1)))
        e2(i,1)=3;
        MHDYep(i,1)=(x10d(i,1)-ep(i,1))/(x10d(i,1)-x3d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x11d(i,1))&(ep(i,1)<=x4d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(ep(i,1)-x11d(i,1))/(x4d(i,1)-x11d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x4d(i,1))&(ep(i,1)<=x12d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(x12d(i,1)-ep(i,1))/(x12d(i,1)-x4d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x13d(i,1))&(ep(i,1)<=x5d(i,1)))
        e2(i,1)=5;
        MHDPCBep(i,1)=(ep(i,1)-x13d(i,1))/(x5d(i,1)-x13d(i,1));
        %e1,e2
        kurallar3g
    end;



end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%----e=HNK(I.BOLGE) durumu--------%%%%%%%%%%%%%%%
bolge=2;
if ((e(i,1)>=x7(i,1))&(e(i,1)<=x2(i,1)))
    e1(i,1)=2;
    MHNKe(i,1)=(e(i,1)-x7(i,1))/(x2(i,1)-x7(i,1));
    if ((ep(i,1)>=x1d(i,1))&(ep(i,1)<=x6d(i,1)))
        e2(i,1)=1;
        MHDNCKep(i,1)=(x6d(i,1)-ep(i,1))/(x6d(i,1)-x1d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x7d(i,1))&(ep(i,1)<=x2d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(ep(i,1)-x7d(i,1))/(x2d(i,1)-x7d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x2d(i,1))&(ep(i,1)<=x8d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(x8d(i,1)-ep(i,1))/(x8d(i,1)-x2d(i,1));
        %e1,e2
        kurallar3g
```

```
    end;
    if ((ep(i,1)>=x9d(i,1))&(ep(i,1)<=x3d(i,1)))
        e2(i,1)=3;
        MHDYep(i,1)=(ep(i,1)-x9d(i,1))/(x3d(i,1)-x9d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x3d(i,1))&(ep(i,1)<=x10d(i,1)))
        e2(i,1)=3;
        MHDYep(i,1)=(x10d(i,1)-ep(i,1))/(x10d(i,1)-x3d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x11d(i,1))&(ep(i,1)<=x4d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(ep(i,1)-x11d(i,1))/(x4d(i,1)-x11d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x4d(i,1))&(ep(i,1)<=x12d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(x12d(i,1)-ep(i,1))/(x12d(i,1)-x4d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x13d(i,1))&(ep(i,1)<=x5d(i,1)))
        e2(i,1)=5;
        MHDPCBep(i,1)=(ep(i,1)-x13d(i,1))/(x5d(i,1)-x13d(i,1));
        %e1,e2
        kurallar3g
    end;


end;


%%%----e=HNK(II.BOLGE) durumu--------%%%%%%%%%%%%%%
bolge=3;
if ((e(i,1)>=x2(i,1))&(e(i,1)<=x8(i,1)))
    e1(i,1)=2;
    MHNKe(i,1)=(x8(i,1)-e(i,1))/(x8(i,1)-x2(i,1));
    if ((ep(i,1)>=x1d(i,1))&(ep(i,1)<=x6d(i,1)))
        e2(i,1)=1;
        MHDNCKep(i,1)=(x6d(i,1)-ep(i,1))/(x6d(i,1)-x1d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x7d(i,1))&(ep(i,1)<=x2d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(ep(i,1)-x7d(i,1))/(x2d(i,1)-x7d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x2d(i,1))&(ep(i,1)<=x8d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(x8d(i,1)-ep(i,1))/(x8d(i,1)-x2d(i,1));
        %e1,e2
        kurallar3g
```

```
    end;
    if ((ep(i,1)>=x9d(i,1))&(ep(i,1)<=x3d(i,1)))
        e2(i,1)=3;
        MHDYep(i,1)=(ep(i,1)-x9d(i,1))/(x3d(i,1)-x9d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x3d(i,1))&(ep(i,1)<=x10d(i,1)))
        e2(i,1)=3;
        MHDYep(i,1)=(x10d(i,1)-ep(i,1))/(x10d(i,1)-x3d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x11d(i,1))&(ep(i,1)<=x4d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(ep(i,1)-x11d(i,1))/(x4d(i,1)-x11d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x4d(i,1))&(ep(i,1)<=x12d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(x12d(i,1)-ep(i,1))/(x12d(i,1)-x4d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x13d(i,1))&(ep(i,1)<=x5d(i,1)))
        e2(i,1)=5;
        MHDPCBep(i,1)=(ep(i,1)-x13d(i,1))/(x5d(i,1)-x13d(i,1));
        %e1,e2
        kurallar3g
    end;


end;


 %%%----HY(I.BOLGE) durumu-----%%%%%%%%%%
 bolge=4;
if ((e(i,1)>=x9(i,1))&(e(i,1)<=x3(i,1)))
    e1(i,1)=3;
    MHYe(i,1)=(e(i,1)-x9(i,1))/(x3(i,1)-x9(i,1));
    if ((ep(i,1)>=x1d(i,1))&(ep(i,1)<=x6d(i,1)))
        e2(i,1)=1;
        MHDNCKep(i,1)=(x6d(i,1)-ep(i,1))/(x6d(i,1)-x1d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x7d(i,1))&(ep(i,1)<=x2d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(ep(i,1)-x7d(i,1))/(x2d(i,1)-x7d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x2d(i,1))&(ep(i,1)<=x8d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(x8d(i,1)-ep(i,1))/(x8d(i,1)-x2d(i,1));
        %e1,e2
        kurallar3g
```

```
    end;
    if ((ep(i,1)>=x9d(i,1))&(ep(i,1)<=x3d(i,1)))
        e2(i,1)=3;
        MHDYep(i,1)=(ep(i,1)-x9d(i,1))/(x3d(i,1)-x9d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x3d(i,1))&(ep(i,1)<=x10d(i,1)))
        e2(i,1)=3;
        MHDYep(i,1)=(x10d(i,1)-ep(i,1))/(x10d(i,1)-x3d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x11d(i,1))&(ep(i,1)<=x4d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(ep(i,1)-x11d(i,1))/(x4d(i,1)-x11d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x4d(i,1))&(ep(i,1)<=x12d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(x12d(i,1)-ep(i,1))/(x12d(i,1)-x4d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x13d(i,1))&(ep(i,1)<=x5d(i,1)))
        e2(i,1)=5;
        MHDPCBep(i,1)=(ep(i,1)-x13d(i,1))/(x5d(i,1)-x13d(i,1));
        %e1,e2
        kurallar3g
    end;


end;

     %%%----HY(II.BOLGE) durumu-----%%%%%%%%%%
 bolge=5;
if ((e(i,1)>=x3(i,1))&(e(i,1)<=x10(i,1)))
    e1(i,1)=3;
    MHYe(i,1)=(x10(i,1)-e(i,1))/(x10(i,1)-x3(i,1));
   if ((ep(i,1)>=x1d(i,1))&(ep(i,1)<=x6d(i,1)))
       e2(i,1)=1;
        MHDNCKep(i,1)=(x6d(i,1)-ep(i,1))/(x6d(i,1)-x1d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x7d(i,1))&(ep(i,1)<=x2d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(ep(i,1)-x7d(i,1))/(x2d(i,1)-x7d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x2d(i,1))&(ep(i,1)<=x8d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(x8d(i,1)-ep(i,1))/(x8d(i,1)-x2d(i,1));
        %e1,e2
        kurallar3g
    end;
```

```
    if ((ep(i,1)>=x9d(i,1))&(ep(i,1)<=x3d(i,1)))
        e2(i,1)=3;
        MHDYep(i,1)=(ep(i,1)-x9d(i,1))/(x3d(i,1)-x9d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x3d(i,1))&(ep(i,1)<=x10d(i,1)))
        e2(i,1)=3;
        MHDYep(i,1)=(x10d(i,1)-ep(i,1))/(x10d(i,1)-x3d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x11d(i,1))&(ep(i,1)<=x4d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(ep(i,1)-x11d(i,1))/(x4d(i,1)-x11d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x4d(i,1))&(ep(i,1)<=x12d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(x12d(i,1)-ep(i,1))/(x12d(i,1)-x4d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x13d(i,1))&(ep(i,1)<=x5d(i,1)))
        e2(i,1)=5;
        MHDPCBep(i,1)=(ep(i,1)-x13d(i,1))/(x5d(i,1)-x13d(i,1));
        %e1,e2
        kurallar3g
    end;


end;

%%-----HPB(I.BOLGE)DURUMU---------%%%%%%%%
bolge=6;
if ((e(i,1)>=x11(i,1))&(e(i,1)<=x4(i,1)))
    e1(i,1)=4;
    MHPBe(i,1)=(e(i,1)-x11(i,1))/(x4(i,1)-x11(i,1));

    if ((ep(i,1)>=x1d(i,1))&(ep(i,1)<=x6d(i,1)))
        e2(i,1)=1;
        MHDNCKep(i,1)=(x6d(i,1)-ep(i,1))/(x6d(i,1)-x1d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x7d(i,1))&(ep(i,1)<=x2d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(ep(i,1)-x7d(i,1))/(x2d(i,1)-x7d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x2d(i,1))&(ep(i,1)<=x8d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(x8d(i,1)-ep(i,1))/(x8d(i,1)-x2d(i,1));
        %e1,e2
        kurallar3g
    end;
```

```
    if ((ep(i,1)>=x9d(i,1))&(ep(i,1)<=x3d(i,1)))
        e2(i,1)=3;
        MHDYep(i,1)=(ep(i,1)-x9d(i,1))/(x3d(i,1)-x9d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x3d(i,1))&(ep(i,1)<=x10d(i,1)))
        e2(i,1)=3;
        MHDYep(i,1)=(x10d(i,1)-ep(i,1))/(x10d(i,1)-x3d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x11d(i,1))&(ep(i,1)<=x4d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(ep(i,1)-x11d(i,1))/(x4d(i,1)-x11d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x4d(i,1))&(ep(i,1)<=x12d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(x12d(i,1)-ep(i,1))/(x12d(i,1)-x4d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x13d(i,1))&(ep(i,1)<=x5d(i,1)))
        e2(i,1)=5;
        MHDPCBep(i,1)=(ep(i,1)-x13d(i,1))/(x5d(i,1)-x13d(i,1));
        %e1,e2
        kurallar3g
    end;


end;

    %%%-----HPB(II.BOLGE) DURUMU------%%%%%%%%
   bolge=7;
if ((e(i,1)>=x4(i,1))&(e(i,1)<=x12(i,1)))
    e1(i,1)=4;
    MHPBe(i,1)=(x12(i,1)-e(i,1))/(x12(i,1)-x4(i,1));
    if ((ep(i,1)>=x1d(i,1))&(ep(i,1)<=x6d(i,1)))
        e2(i,1)=1;
        MHDNCKep(i,1)=(x6d(i,1)-ep(i,1))/(x6d(i,1)-x1d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x7d(i,1))&(ep(i,1)<=x2d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(ep(i,1)-x7d(i,1))/(x2d(i,1)-x7d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x2d(i,1))&(ep(i,1)<=x8d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(x8d(i,1)-ep(i,1))/(x8d(i,1)-x2d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x9d(i,1))&(ep(i,1)<=x3d(i,1)))
```

```
        e2(i,1)=3;
        MHDYep(i,1)=(ep(i,1)-x9d(i,1))/(x3d(i,1)-x9d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x3d(i,1))&(ep(i,1)<=x10d(i,1))) e2(i,1)=3;
        MHDYep(i,1)=(x10d(i,1)-ep(i,1))/(x10d(i,1)-x3d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x11d(i,1))&(ep(i,1)<=x4d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(ep(i,1)-x11d(i,1))/(x4d(i,1)-x11d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x4d(i,1))&(ep(i,1)<=x12d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(x12d(i,1)-ep(i,1))/(x12d(i,1)-x4d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x13d(i,1))&(ep(i,1)<=x5d(i,1)))
        e2(i,1)=5;
        MHDPCBep(i,1)=(ep(i,1)-x13d(i,1))/(x5d(i,1)-x13d(i,1));
        %e1,e2
        kurallar3g
    end;



end;


    %%%%----HPCB durumu--------%%%%%%%%
   bolge=8;
if ((e(i,1)>=x13(i,1))&(e(i,1)<=x5(i,1)))
    e1(i,1)=5;
    MHPCBe(i,1)=(e(i,1)-x13(i,1))/(x5(i,1)-x13(i,1));
    if ((ep(i,1)>=x1d(i,1))&(ep(i,1)<=x6d(i,1)))
        e2(i,1)=1;
        MHDNCKep(i,1)=(x6d(i,1)-ep(i,1))/(x6d(i,1)-x1d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x7d(i,1))&(ep(i,1)<=x2d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(ep(i,1)-x7d(i,1))/(x2d(i,1)-x7d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x2d(i,1))&(ep(i,1)<=x8d(i,1)))
        e2(i,1)=2;
        MHDNKep(i,1)=(x8d(i,1)-ep(i,1))/(x8d(i,1)-x2d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x9d(i,1))&(ep(i,1)<=x3d(i,1)))
```

```
        e2(i,1)=3;
        MHDYep(i,1)=(ep(i,1)-x9d(i,1))/(x3d(i,1)-x9d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x3d(i,1))&(ep(i,1)<=x10d(i,1)))
        e2(i,1)=3;
        MHDYep(i,1)=(x10d(i,1)-ep(i,1))/(x10d(i,1)-x3d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x11d(i,1))&(ep(i,1)<=x4d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(ep(i,1)-x11d(i,1))/(x4d(i,1)-x11d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x4d(i,1))&(ep(i,1)<=x12d(i,1)))
        e2(i,1)=4;
        MHDPBep(i,1)=(x12d(i,1)-ep(i,1))/(x12d(i,1)-x4d(i,1));
        %e1,e2
        kurallar3g
    end;
    if ((ep(i,1)>=x13d(i,1))&(ep(i,1)<=x5d(i,1)))
        e2(i,1)=5;
        MHDPCBep(i,1)=(ep(i,1)-x13d(i,1))/(x5d(i,1)-x13d(i,1));
        %e1,e2
        kurallar3g
    end;


end;

if (tdeltaepay(i,1)==0)&(tdeltaepayda(i,1)==0)
       deltae(i,1)=0;
else
    deltae(i,1)=tdeltaepay(i,1)/tdeltaepayda(i,1);
    deltae(i,1)=0.0174*deltae(i,1);
end;
% deltae
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%-------kurallar3g programı---------%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%e1:Hata bölge dizisi
%e2:Hatanın değişimi bölge dizisi

%%%%%%-----Kural 1------------%%%%%%%%%%%%
if ((e1(i,1)==1)&(e2(i,1)==1)) u(i,1)=x14(i,1);
    if MHNCKe(i,1)<=MHDNCKep(i,1) Mx14(i,1)=MHNCKe(i,1)
      else Mx14(i,1)=MHDNCKep(i,1);
    end;
deltaepay(i,1)=x14(i,1)*Mx14(i,1);
deltaepayda(i,1)=Mx14(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%%-----Kural 2------------%%%%%%%%%%%%
if ((e1(i,1)==1)&(e2(i,1)==2)) u(i,1)=x15(i,1);
    if MHNCKe(i,1)<=MHDNKep(i,1) Mx15(i,1)=MHNCKe(i,1);
    else Mx15(i,1)=MHDNKep(i,1);
    end;
deltaepay(i,1)=x15(i,1)*Mx15(i,1);
deltaepayda(i,1)=Mx15(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%%-----Kural 3------------%%%%%%%%%%%%
if ((e1(i,1)==1)&(e2(i,1)==3)) u(i,1)=x16(i,1);
    if MHNCKe(i,1)<=MHDYep(i,1) Mx16(i,1)=MHNCKe(i,1);
    else Mx16(i,1)=MHDYep(i,1);
    end
deltaepay(i,1)=x16(i,1)*Mx16(i,1);
deltaepayda(i,1)=Mx16(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%%-----Kural 4------------%%%%%%%%%%%%
if ((e1(i,1)==1)&(e2(i,1)==4)) u(i,1)=x17(i,1);
    if MHNCKe(i,1)<=MHDPBep(i,1) Mx17(i,1)=MHNCKe(i,1);
    else Mx17(i,1)=MHDPBep(i,1);
    end
deltaepay(i,1)=x17(i,1)*Mx17(i,1);
deltaepayda(i,1)=Mx17(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%%-----Kural5------------%%%%%%%%%%%%
if ((e1(i,1)==1)&(e2(i,1)==5)) u(i,1)=x18(i,1);
    if MHNCKe(i,1)<=MHDPCBep(i,1) Mx18(i,1)=MHNCKe(i,1);
    else Mx18(i,1)=MHDPCBep(i,1);
    end
deltaepay(i,1)=x18(i,1)*Mx18(i,1);
deltaepayda(i,1)=Mx18(i,1);
```

```
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%-----Kural 6------------%%%%%%%%%%%
if ((e1(i,1)==2)&(e2(i,1)==1)) u(i,1)=x19(i,1);
    if MHNKe(i,1)<=MHDNCKep(i,1) Mx19(i,1)=MHNKe(i,1);
    else Mx19(i,1)=MHDNCKep(i,1);
    end
deltaepay(i,1)=x19(i,1)*Mx19(i,1);
deltaepayda(i,1)=Mx19(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%-----Kural 7------------%%%%%%%%%%%
if ((e1(i,1)==2)&(e2(i,1)==2)) u(i,1)=x20(i,1);
    if MHNKe(i,1)<=MHDNKep(i,1) Mx20(i,1)=MHNKe(i,1);
    else Mx20(i,1)=MHDNKep(i,1);
    end
deltaepay(i,1)=x20(i,1)*Mx20(i,1);
deltaepayda(i,1)=Mx20(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;


%%%%%-----Kural 8------------%%%%%%%%%%%
if ((e1(i,1)==2)&(e2(i,1)==3)) u(i,1)=x21(i,1);
    if MHNKe(i,1)<=MHDYep(i,1) Mx21(i,1)=MHNKe(i,1);
    else Mx21(i,1)=MHDYep(i,1);
    end
deltaepay(i,1)=x21(i,1)*Mx21(i,1);
deltaepayda(i,1)=Mx21(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%-----Kural 9------------%%%%%%%%%%%
if ((e1(i,1)==2)&(e2(i,1)==4)) u(i,1)=x22(i,1);
    if MHNKe(i,1)<=MHDPBep(i,1) Mx22(i,1)=MHNKe(i,1);
    else Mx22(i,1)=MHDPBep(i,1);
    end
deltaepay(i,1)=x22(i,1)*Mx22(i,1);
deltaepayda(i,1)=Mx22(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%-----Kural10------------%%%%%%%%%%%
if ((e1(i,1)==2)&(e2(i,1)==5)) u(i,1)=x23(i,1);
    if MHNKe(i,1)<=MHDPCBep(i,1) Mx23(i,1)=MHNKe(i,1);
    else Mx23(i,1)=MHDPCBep(i,1);
    end
deltaepay(i,1)=x23(i,1)*Mx23(i,1);
deltaepayda(i,1)=Mx23(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
```

```
end;


%%%%%-----Kural 11------------%%%%%%%%%%%%
if ((e1(i,1)==3)&(e2(i,1)==1)) u(i,1)=x24(i,1);
    if MHYe(i,1)<=MHDNCKep(i,1) Mx24(i,1)=MHYe(i,1);
    else Mx24(i,1)=MHDNCKep(i,1);
    end
deltaepay(i,1)=x24(i,1)*Mx24(i,1);
deltaepayda(i,1)=Mx24(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%-----Kural12------------%%%%%%%%%%%%

if ((e1(i,1)==3)&(e2(i,1)==2)) u(i,1)=x25(i,1);
    if MHYe(i,1)<=MHDNKep(i,1) Mx25(i,1)=MHYe(i,1);
    else Mx25(i,1)=MHDNKep(i,1);
    end
deltaepay(i,1)=x25(i,1)*Mx25(i,1);
deltaepayda(i,1)=Mx25(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%-----Kural13------------%%%%%%%%%%%%
if ((e1(i,1)==3)&(e2(i,1)==3)) u(i,1)=x26(i,1);
    if MHYe(i,1)<=MHDYep(i,1) Mx26(i,1)=MHYe(i,1);
    else Mx26(i,1)=MHDYep(i,1);
    end
deltaepay(i,1)=x26(i,1)*Mx26(i,1);
deltaepayda(i,1)=Mx26(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%-----Kural14------------%%%%%%%%%%%%
if ((e1(i,1)==3)&(e2(i,1)==4)) u(i,1)=x27(i,1);
    if MHYe(i,1)<=MHDPBep(i,1) Mx27(i,1)=MHYe(i,1);
    else Mx27(i,1)=MHDPBep(i,1);
    end
deltaepay(i,1)=x27(i,1)*Mx27(i,1);
deltaepayda(i,1)=Mx27(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%-----Kural15------------%%%%%%%%%%%%
if ((e1(i,1)==3)&(e2(i,1)==5)) u(i,1)=x28(i,1);
    if MHYe(i,1)<=MHDPCBep(i,1) Mx28(i,1)=MHYe(i,1);
    else Mx28(i,1)=MHDPCBep(i,1);
    end
deltaepay(i,1)=x28(i,1)*Mx28(i,1);
deltaepayda(i,1)=Mx28(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
```

```
end;

%%%%%%-----Kural16------------%%%%%%%%%%%%
if ((e1(i,1)==4)&(e2(i,1)==1)) u(i,1)=x29(i,1);
    if MHPBe(i,1)<=MHDNCKep(i,1) Mx29(i,1)=MHPBe(i,1);
    else Mx29(i,1)=MHDNCKep(i,1);
    end
deltaepay(i,1)=x29(i,1)*Mx29(i,1);
deltaepayda(i,1)=Mx29(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%%-----Kural17------------%%%%%%%%%%%%
if ((e1(i,1)==4)&(e2(i,1)==2)) u(i,1)=x30(i,1);
    if MHPBe(i,1)<=MHDNKep(i,1) Mx30(i,1)=MHPBe(i,1);
    else Mx30(i,1)=MHDNKep(i,1);
    end
deltaepay(i,1)=x30(i,1)*Mx30(i,1);
deltaepayda(i,1)=Mx30(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%%-----Kural18------------%%%%%%%%%%%%
if ((e1(i,1)==4)&(e2(i,1)==3)) u(i,1)=x31(i,1);
    if MHPBe(i,1)<=MHDYep(i,1) Mx31(i,1)=MHPBe(i,1);
    else Mx31(i,1)=MHDYep(i,1);
    end
deltaepay(i,1)=x31(i,1)*Mx31(i,1);
deltaepayda(i,1)=Mx31(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%%-----Kural 19------------%%%%%%%%%%%%
if ((e1(i,1)==4)&(e2(i,1)==4)) u(i,1)=x32(i,1);
    if MHPBe(i,1)<=MHDPBep(i,1) Mx32(i,1)=MHPBe(i,1);
    else Mx32(i,1)=MHDPBep(i,1);
    end
deltaepay(i,1)=x32(i,1)*Mx32(i,1);
deltaepayda(i,1)=Mx32(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);


end;

%%%%%%-----Kural 20------------%%%%%%%%%%%%
if ((e1(i,1)==4)&(e2(i,1)==5)) u(i,1)=x33(i,1);
    if MHPBe(i,1)<=MHDPCBep(i,1) Mx33(i,1)=MHPBe(i,1);
    else Mx33(i,1)=MHDPCBep(i,1);
    end
deltaepay(i,1)=x33(i,1)*Mx33(i,1);
deltaepayda(i,1)=Mx33(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
```

```
end;

%%%%%-----Kural 21------------%%%%%%%%%%%%
if ((e1(i,1)==5)&(e2(i,1)==1)) u(i,1)=x34(i,1);
    if MHPCBe(i,1)<=MHDNCKep(i,1) Mx34(i,1)=MHPCBe(i,1);
    else Mx34(i,1)=MHDNCKep(i,1);
    end
deltaepay(i,1)=x34(i,1)*Mx34(i,1);
deltaepayda(i,1)=Mx34(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;


%%%%%-----Kural 22------------%%%%%%%%%%%%
if ((e1(i,1)==5)&(e2(i,1)==2)) u(i,1)=x35(i,1);
    if MHPCBe(i,1)<=MHDNKep(i,1) Mx35(i,1)=MHPCBe(i,1);
    else Mx35(i,1)=MHDNKep(i,1);
    end
deltaepay(i,1)=x35(i,1)*Mx35(i,1);
deltaepayda(i,1)=Mx35(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%-----Kural23------------%%%%%%%%%%%%
if ((e1(i,1)==5)&(e2(i,1)==3)) u(i,1)=x36(i,1);
    if MHPCBe(i,1)<=MHDYep(i,1) Mx36(i,1)=MHPCBe(i,1);
    else Mx36(i,1)=MHDYep(i,1);
    end
deltaepay(i,1)=x36(i,1)*Mx36(i,1);
deltaepayda(i,1)=Mx36(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%-----Kural 24------------%%%%%%%%%%%%
if ((e1(i,1)==5)&(e2(i,1)==4)) u(i,1)=x37(i,1);
    if MHPCBe(i,1)<=MHDPBep(i,1) Mx37(i,1)=MHPCBe(i,1);
    else Mx37(i,1)=MHDPBep(i,1);
    end
deltaepay(i,1)=x37(i,1)*Mx37(i,1);
deltaepayda(i,1)=Mx37(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;

%%%%%-----Kural 25------------%%%%%%%%%%%%
if ((e1(i,1)==5)&(e2(i,1)==5)) u(i,1)=x38(i,1);
    if MHPCBe(i,1)<=MHDPCBep(i,1) Mx38(i,1)=MHPCBe(i,1);
    else Mx38(i,1)=MHDPCBep(i,1);
    end
deltaepay(i,1)=x38(i,1)*Mx38(i,1);
deltaepayda(i,1)=Mx38(i,1);
tdeltaepay(i,1)=tdeltaepay(i,1)+deltaepay(i,1);
tdeltaepayda(i,1)=tdeltaepayda(i,1)+deltaepayda(i,1);
end;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%%%%%%%%%% runkutf3g programı %%%%%%%%%%

% ts  :SAMPLE PERIOD
% xx  :STATE VECTOR
% nx  :NUMBER OF STATES
% xxp :DERIVATIVE OF STATE VECTOR
%ctr
  Armf3g
 for j=1:nx
   tempxx(j)=xx(j);
   tempxxp(j)=xxp(j);
 end
  for j=1:nx
     xx(j)=xx(j)+.5*ts*xxp(j);
  end
  time=time+.5*ts;
  %ctr
  Armf3g
  for j=1:nx
     tempxxp(j)=tempxxp(j)+2*xxp(j);
     xx(j)=tempxx(j)+.5*ts*xxp(j);
  end
  %ctr
  Armf3g
  for j=1:nx
    tempxxp(j)=tempxxp(j)+2*xxp(j);
    xx(j)=tempxx(j)+ts*xxp(j);
  end
  time=time+.5*ts;
  %ctr
  Armf3g
  for j=1:nx
    xx(j)=tempxx(j)+ts*(tempxxp(j)+xxp(j))/6;
  end
```

```
%%%%%%%%%%%%%%%%%%%%%
% armf3g programı %
%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%aircraft dynamics%%%%%%%%%
% STATE EQUATİONS FOR (H(S)/DELTAEc(S)) %
 %%%%%%%%%%%%%%%%%
 %%%%SECOND FLIGHT CONDITION (6100m u0=250)%%%%%%%%%

xxp(1)= xx(2);
xxp(2)= xx(3);
xxp(3)= xx(4);
xxp(4)= xx(5)-803.82*deltae(i,1);
xxp(5)= xx(6)-87127.1646*deltae(i,1);
xxp(6)= -166.6*xx(3)-180.02*xx(4)-131.01*xx(5)-
21.47*xx(6)+1975617.26*deltae(i,1);

%%%% FLIGHT CONDITION (0 m(SEA LEVEL) KULLANMIYORUZ%%%%%%%%%

%xxp(1)= xx(2);
%xxp(2)= xx(3);
%xxp(3)= xx(4);
%xxp(4)= xx(5)-195.1*deltae(i,1);
%xxp(5)= xx(6)+3996.79*deltae(i,1);
%xxp(6)= -58.4*xx(3)-103.8*xx(4)-118.9*xx(5)-20.9*xx(6)-
61554.92*deltae(i,1);

%%%%THIRD FLIGHT CONDITION (12200 m)%%%%%%%%%%

%xxp(1)= xx(2);
%xxp(2)= xx(3);
%xxp(3)= xx(4);
%xxp(4)= xx(5)-525.5*deltae(i,1);
%xxp(5)= xx(6)-47334.85*deltae(i,1);
%xxp(6)= -85.7*xx(3)-92.7*xx(4)-115.95*xx(5)-
20.7*xx(6)+1031988.11*deltae(i,1);

%%%%FIRST FLIGHT CONDITION (6100 uo158 m)%%%%%%%%%%

%xxp(1)= xx(2);
%xxp(2)= xx(3);
%xxp(3)= xx(4);
%xxp(4)= xx(5)-505.7*deltae(i,1);
%xxp(5)= xx(6)-24052.17*deltae(i,1);
%xxp(6)= -113*xx(3)-114.3*xx(4)-119.45*xx(5)-
20.9*xx(6)+556103.13*deltae(i,1);
```